CAMBRIDGE SEMANTICS

Anzo® 6.0 Deployment Guide

Last Updated: 2/14/2024

Online documentation is available at docs.cambridgesemantics.com

Table of Contents

About This Doc

This guide includes planning guidance and deployment instructions for Anzo and each of the knowledge graph platform components.

> **Tip**
> You can view the contents of this guide as well as release notes, end-user, and administration documentation online at docs.cambridgesemantics.com. You can also find PDF versions of the end-user and administration documentation here.

The following list introduces the sections in this guide.

- Anzo Platform Overview: Introduces the components in the Anzo platform and describes how they work together to enable users to create knowledge graphs from various enterprise data sources.
- Platform Requirements: Provides Information about compatibility between components as well as requirements and recommendations to help you plan and provision the platform infrastructure.
- Deploying the Platform Components: Provides instructions for installing, upgrading, and uninstalling each of the platform components.

Anzo Platform Overview

Anzo is a complete knowledge graph platform built on AnzoGraph, a high-performance graph OLAP engine. Anzo is an open overlay platform that leverages standards, including the W3C's RDF, OWL, SKOS, and SPARQL standards, to combine structured and unstructured enterprise metadata and data into knowledge graphs that can be explored, transformed, analyzed, and visualized with the included Hi-Res Analytics dashboard tool or third-party BI applications. The Anzo platform integrates with your existing governance and security control policies and includes APIs for integration with other processes. The topics in this section introduce you to the platform and each of its components.

In this section:

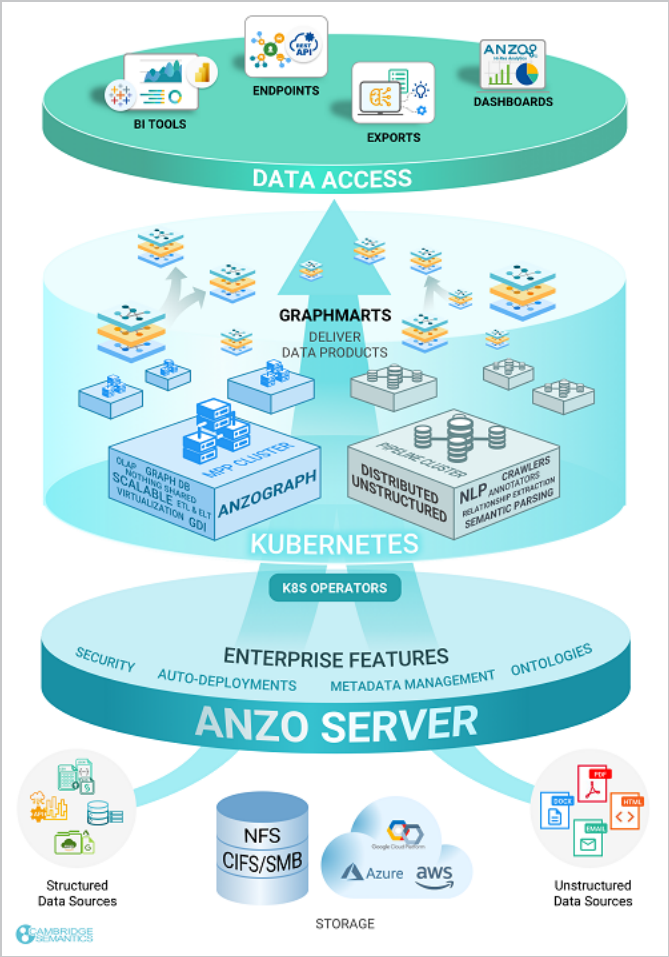
Introduction to the Platform Components

The Anzo knowledge graph platform connects multiple components that enable you to ingest, transform, store, explore, and analyze various types of data. When determining which components to deploy in your platform, the primary distinction is made between structured and semi-structured (relational databases and flat files) and unstructured (documents, text snippets, web pages, emails, etc.) data sources. The list below introduces each of the components in the platform. An environment that includes all of these components could process both structured and unstructured data. Note that each of the platform components is deployed on separate instances or clusters in the same network.

- **Anzo Server**: This required component is the administrative layer that helps organize all of the platform assets. It connects and manages the other components and provides the Anzo application, Administration, and Hi-Res Analytics user interfaces. Anzo manages all of the onboarded data and metadata and provides access control over the other components and artifacts in the system.
- **AnzoGraph**: This required component is Anzo's in-memory graph OLAP engine. AnzoGraph stores all of your graphmarts and includes the Graph Data Interface (GDI), which is used to ingest (or virtualize) and transform all of the structured (and semi-structured) data that is onboarded to the platform.
- **Distributed Unstructured**: This optional component is a cluster of worker nodes that process unstructured documents (like PDFs, text snippets, emails, and knowledgebases) and convert them to the graph data model.
- **Elasticsearch**: This optional component supports the creation, storage, and search of indexes for both structured and unstructured data. Elasticsearch is required for onboarding unstructured data, and it is optional for structured data, depending on whether you want to be able to index and search your knowledge graphs.
- **Shared File System**: The required shared file storage system is a critical part of the platform. The Anzo server and any AnzoGraph, Anzo Distributed Unstructured, and Elasticsearch servers need access to read and write shared files.

Platform Component Details

The diagram below shows an overview of the platform components, their features, and how they work together. Details about the image and the components listed above are provided in the sections below the diagram.



- Structured Data Sources
- Storage
- Unstructured Data Sources
- Anzo Server
- Kubernetes
- AnzoGraph

- Distributed Unstructured
- Graphmarts
- Data Access

Structured Data Sources

Anzo supports ingesting data from structured (relational databases) and semi-structured (flat files) data sources. Anzo connects directly to database sources via ODBC and JDBC drivers and supports loading data directly from CSV, JSON, XML, SAS, and Parquet files. Ingesting structured and semi-structured data sources is automated using AnzoGraph's Graph Data Interface (GDI). The GDI also supports ingesting or virtualizing data via manually written SPARQL queries.

Storage

The Anzo server and all installed platform components need to have read and write access to at least one shared file storage system. Anzo supports connections to NFS, Hadoop Distributed File Systems (HDFS), File Transfer Protocol (FTP or FTPS) systems, Google Cloud Platform (GCP) storage, and Amazon Simple Cloud Storage Service (S3). For the best support and performance for reading and writing files, Cambridge Semantics strongly recommends that you create an NFS and mount it to all of the servers in the platform. If you plan to set up Kubernetes (K8s) integration for dynamic deployments of Anzo components, an NFS is required. For more information, see Shared File Storage Requirements.

Unstructured Data Sources

Unstructured data sources such as documents, PDFs, text snippets, web pages, emails, and content from knowledgebases are ingested using configurable, scalable pipelines. The pipelines generate a graph model for the unstructured text and extracted metadata, and they connect related entities so that the data can be fully integrated into the knowledge graph. The pipelines also build an Elasticsearch index that can be used for fully-integrated queries that search both free-text and semantic relationships within the knowledge graph. More information about unstructured data processing is included in Distributed Unstructured below.

Anzo Server

The Anzo server connects all of the components and provides the user interfaces. Since AnzoGraph is stateless, Anzo manages updates to all of the data that is onboarded. It also manages all data models and other metadata such as data source configuration details, dataset catalog entries, registries, and access control definitions. For more information about how graph data is stored between Anzo and AnzoGraph see Graph Storage Concepts in the Getting Started Guide.

Kubernetes

The AnzoGraph, Anzo Unstructured, and Elasticsearch platform components can be deployed on "static" clusters, where the software is installed on pre-configured hardware, VMs, or cloud instances, or they can be deployed dynamically in a Kubernetes (K8s) cluster. If you choose to configure the K8s infrastructure, Anzo can launch components on-demand and then deprovision the resources when they are not in use. For more information about K8s integration with Anzo, see Kubernetes Concepts.

AnzoGraph

AnzoGraph is Anzo's massively parallel processing (MPP) graph OLAP engine. To provide the highest performance possible, AnzoGraph stores all graph data and performs all analytic operations entirely in memory. You can scale AnzoGraph to run in environments ranging from a single server to tens or even hundreds of servers in a cluster. AnzoGraph also includes advanced analytic functions, such as the analytics that are run when datasets and graphmarts are profiled. And it includes the Graph Data Interface (GDI) plugin, which is used to ingest (or virtualize) and transform all of the structured (and semi-structured) data that is onboarded to Anzo. For more information about AnzoGraph, see AnzoGraph Architecture.

Distributed Unstructured

An Anzo Distributed Unstructured (DU) cluster consists of one leader instance and one or more worker instances. When a user runs an unstructured pipeline, Anzo sends the requests to the leader instance. The leader queues the requests and distributes them to the worker instances to

process in parallel. In order to onboard unstructured data, an Anzo DU cluster and Elasticsearch are required components. For more information about DU and unstructured data processing, see Distributed Unstructured Overview.

Graphmarts

Whether data is ingested with the GDI or unstructured pipelines, it is converted from its original format to a new format that describes the data as a graph model. This format, Resource Description Framework (RDF), simplifies access to complex data and flexibly accommodates new data sources and use cases. The RDF data is added to graphmarts and loaded to AnzoGraph for further transformation and analytics. Graphmarts are collections data products or knowledge graphs that users can blend and enhance. Any subset of data can be combined in a graphmart for analysis. For more information about graphmarts, see Graphmart Concepts in the Getting Started Guide.

Data Access

Users have several options for accessing and analyzing knowledge graphs. Anzo's Hi-Res Analytics application enables users to create dashboards for exploring and visualizing the data without needing to have specialized query knowledge. And, in line with Anzo's open standard architecture, graphmarts can be accessed using modern application program interfaces (APIs) like the Anzo REST API as well as SPARQL-compliant query endpoints. Anzo also offers standards-compliant Open Data Protocol (OData)-based endpoints as part of its Data on Demand service. The Data on Demand service provides access to data from business intelligence tools.

See Platform Requirements for requirements and recommendations for each of the platform components.

Distributed Unstructured Overview

One of Anzo's differentiators as a leading enterprise knowledge graph and data integration platform is its treatment of unstructured data as a first-class citizen in the knowledge graph. Anzo onboards unstructured data—sources that contain text, such as PDFs, text messages, or text snippets embedded in structured data—directly into the knowledge graph using configurable, scalable pipelines. The pipelines generate a graph model for the unstructured text and extracted metadata, and they create connections between related entities so that the data can be fully integrated into the knowledge graph. In addition, the pipelines build an Elasticsearch index that can be used for highly performant, fully-integrated queries that search both free-text and semantic relationships within the knowledge graph.

The following sections provide an overview of the key features of Anzo's unstructured data integration capabilities.

- Support for Crawling a Variety of Sources
- Text Processing and Annotation
- Text Indexing and Searching
- Scalability and Progress Tracking

Support for Crawling a Variety of Sources

Unstructured pipelines can process unstructured text from a large variety of data sources and formats. Configurable crawlers determine what unstructured text a pipeline will process. Crawlers can extract text from a variety of file formats, including PDFs, emails, HTML files, and Microsoft Word documents.

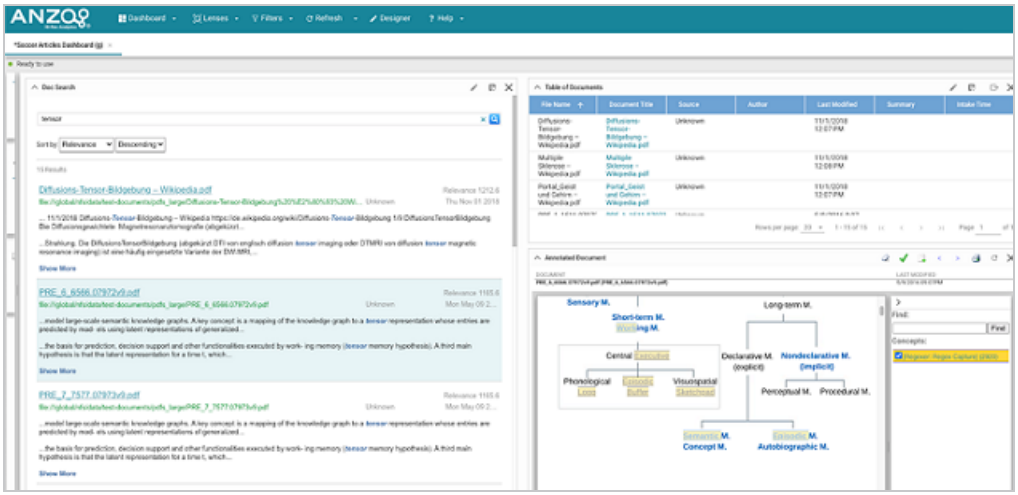
Unstructured pipelines can also be configured to crawl the knowledge graph itself for content to index and annotate—whether the graph contains free-text directly or references to document locations. When combined with Anzo's data virtualization capabilities, this presents a flexible and powerful framework to rapidly process unstructured data and bring it into a knowledge graph from practically any source or repository in a modern data ecosystem.

Text Processing and Annotation

As a baseline, unstructured pipelines extract basic metadata about each document that they process, such as file location, file size, title, author, etc. The metadata is stored within the knowledge graph according to a standardized graph model. The pipelines generate HTML versions of the documents that can be rendered in a browser, and references to the document's original binary are maintained in the graph. With this integration, unstructured content and its associated metadata can be connected and queried alongside any other information stored in the knowledge graph.

Beyond this baseline processing capability, Anzo enables more advanced annotation of unstructured text. Based on pattern matching and taxonomies or dictionaries of terms that already exist in the knowledge graph, annotators pull out facts or references in the text as annotations. The unstructured text and extracted annotations are also added to the knowledge graph, where they are described by a model (ontology) that is dynamically generated by the pipeline. Additionally, unstructured pipelines align the annotations to the source text and include highlights of the annotated text in the HTML version of the document. Once in the knowledge graph, the unstructured annotation data can easily be discovered, explored, and connected with basic document data as well as any other enterprise data in the graph.

The image below shows an HTML rendering of a document and its highlighted annotations in a Hi-Res Analytics dashboard:



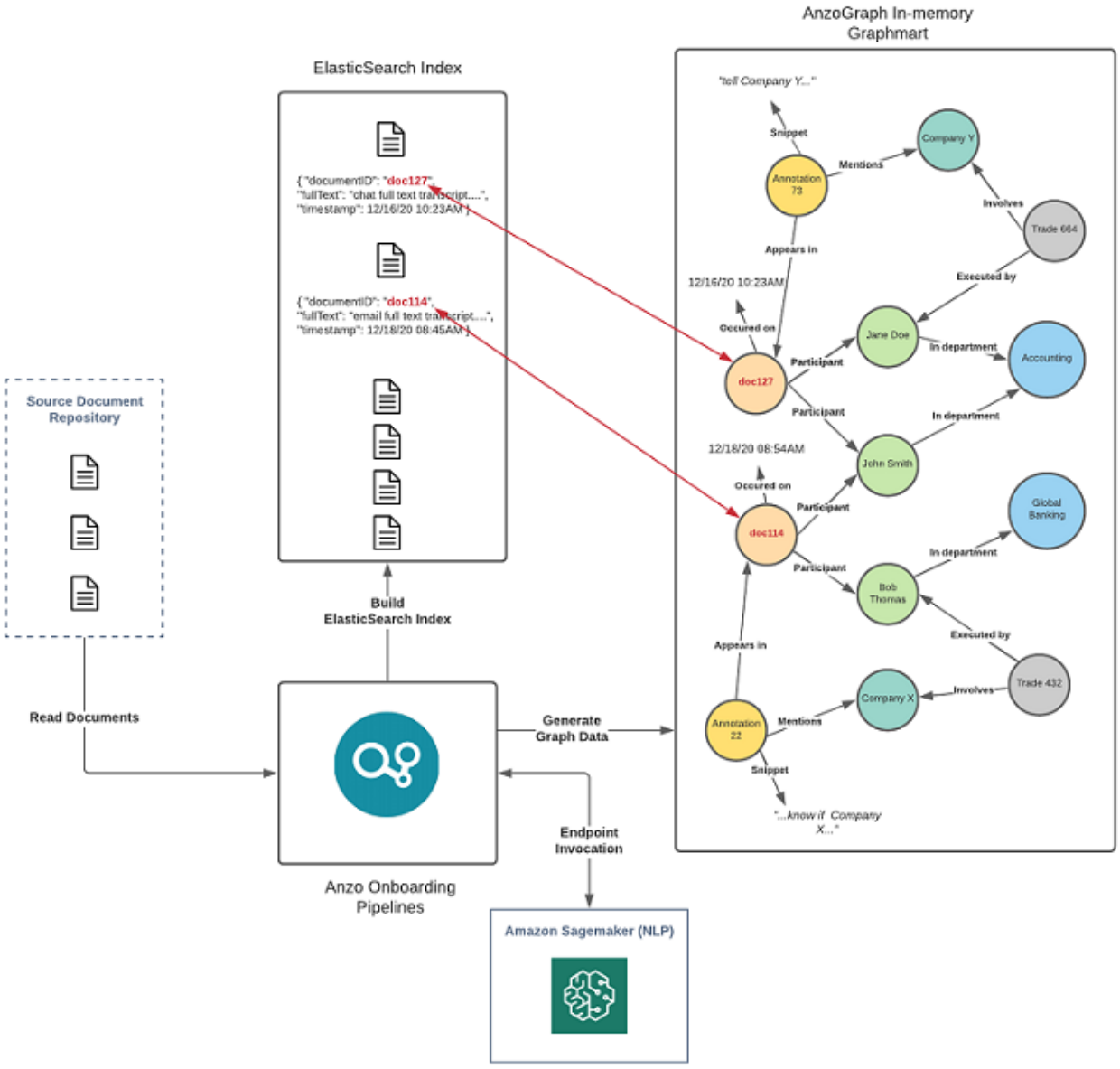
Tip

Unstructured pipelines also offer a flexible and agnostic extension framework to support integration with external NLP engines, such as Amazon Sagemaker, spaCy NER, and Amazon Comprehend, that can provide domain-specific or ML-driven text processing capabilities. Anzo's pipelines provide unstructured plain text to the external components and then bring their output back into the knowledge graph, dynamically generating a graph model and connecting the extracted annotations to the document metadata and related entities. This can serve not only as an effective way to integrate state-of-the-art NLP insights with related data, but also as a flexible and transparent paradigm for validation and analysis of ML-driven NLP development.

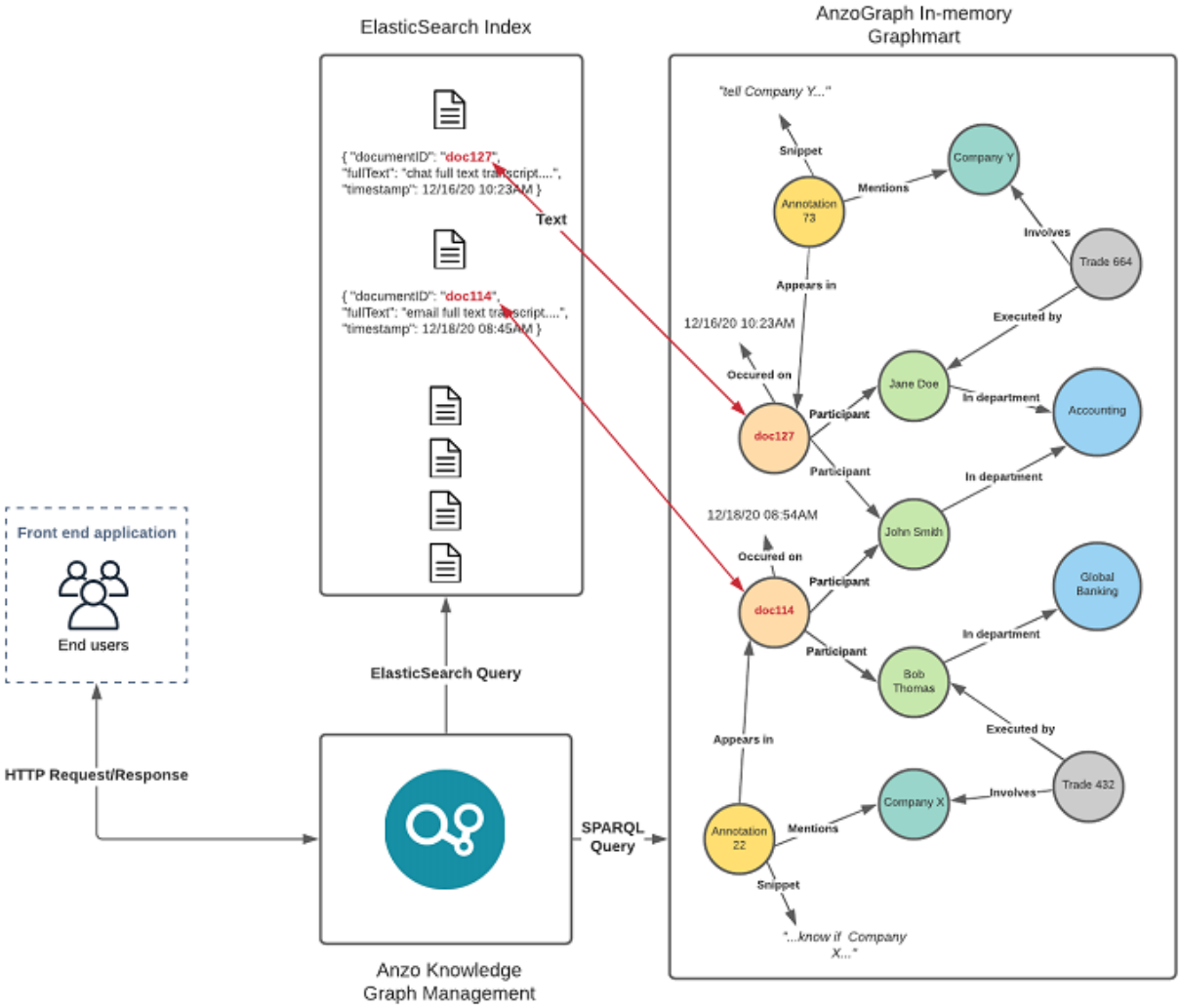
Text Indexing and Searching

Unstructured pipelines create an Elasticsearch index of all unstructured files onboarded to Anzo. The indexes contain references to URIs of related entities in the knowledge graph so that the indexed data can be joined directly against the rich and highly connected graph. When coupled with AnzoGraph's native Elasticsearch SPARQL extension, users can seamlessly execute queries that combine scalable, performant free-text search with complex semantic queries against the graph. This integration can serve as a strong and flexible foundation for advanced, complex modern search applications.

The diagram below shows an overview of the Elasticsearch integration during pipeline processing:



The following diagram shows an overview of Elasticsearch integration during querying and analysis:

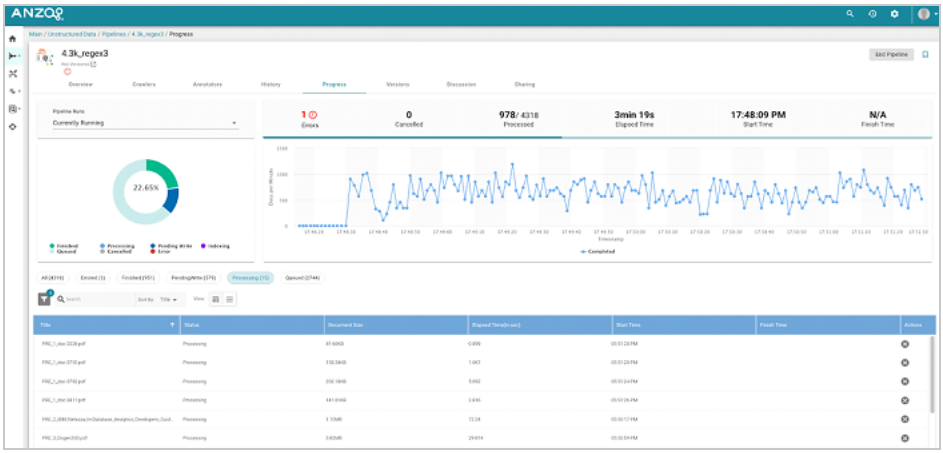


Scalability and Progress Tracking

Unstructured pipelines run using a highly distributed and performant microservice cluster built using Akka. Worker nodes, which perform text processing in parallel, can be scaled up to increase the processing throughput of the pipeline. With this parallelization and scalability, pipelines are capable of processing tens of thousands of unstructured documents per minute. The pipeline processing services can be deployed statically on standard hardware or cloud instances, or they can be spun up dynamically using Anzo's native Kubernetes integration.

To track the progress of unstructured pipelines, Anzo offers a user interface that reports status information about each document as well as any issues encountered. The user interface also shows global statistics about a given pipeline run, including overall processing throughput, percentage complete, and time elapsed. This reporting module gives administrators a centralized view of progress and an easy way to oversee the pipeline as it operates.

The image below shows the unstructured pipeline reporting interface:

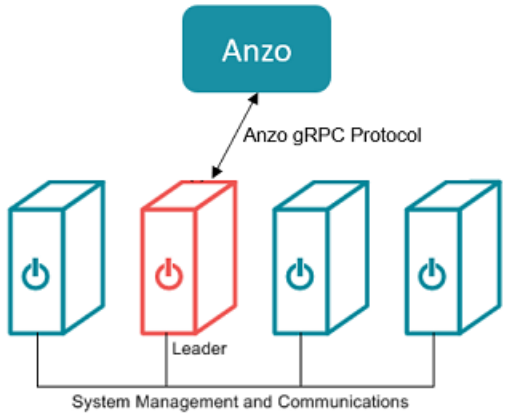


For more information about unstructured pipeline processing and the resulting artifacts, see Unstructured Onboarding Process Overview in the User Guide.

AnzoGraph Architecture

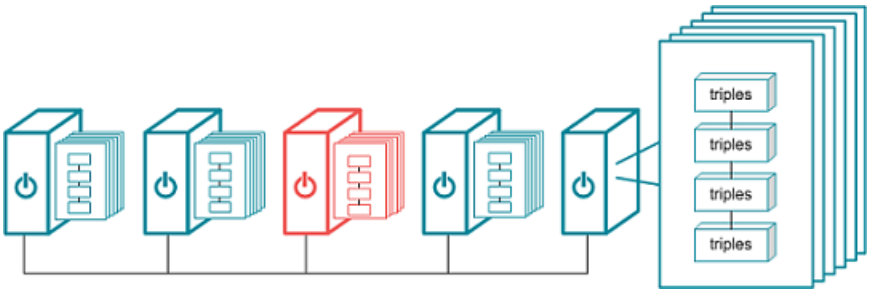
AnzoGraph uses massively parallel processing (MPP) to perform analytic operations on graph data. AnzoGraph distributes data across cluster nodes and processes queries in parallel on all nodes. You can scale AnzoGraph to run in environments ranging from a single server to multiple servers in a cluster in on-premises or cloud environments.

Though all servers in an AnzoGraph cluster store the system metadata and have the ability to perform leader operations, one server acts as the leader for the cluster. All client applications should connect to this server.



In-Memory Data Storage Architecture

To provide the highest performance possible, AnzoGraph stores all graph data and performs all analytic operations entirely in memory. At startup, AnzoGraph sets the number of shards (called "slices" in AnzoGraph) per node to the number of cores on a single server. To utilize massively parallel processing of queries, AnzoGraph distributes (as evenly as possible) the data into memory across all of the slices. When data is loaded, AnzoGraph hashes on subjects to determine how the data is distributed. Distributing on subject allows the database to avoid distributing data over the network under certain conditions. Every slice contains several blocks that store the triples.

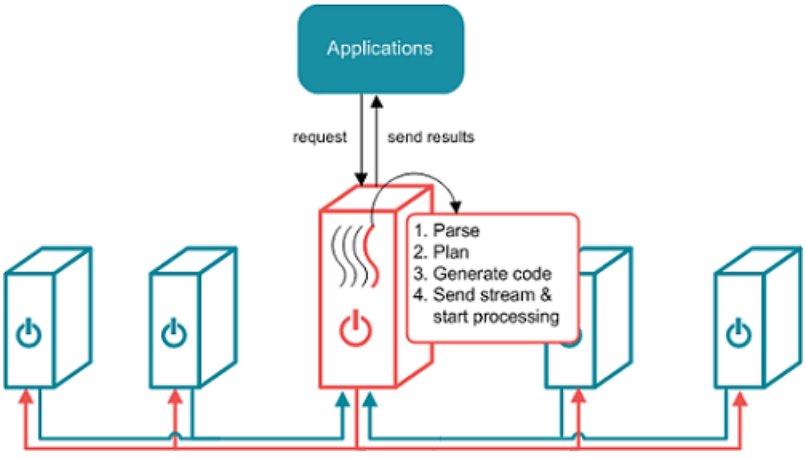


Note

When installed in a cluster, AnzoGraph requires that all servers have equivalent hardware, software, and quality of service.

Leader and Query Processing

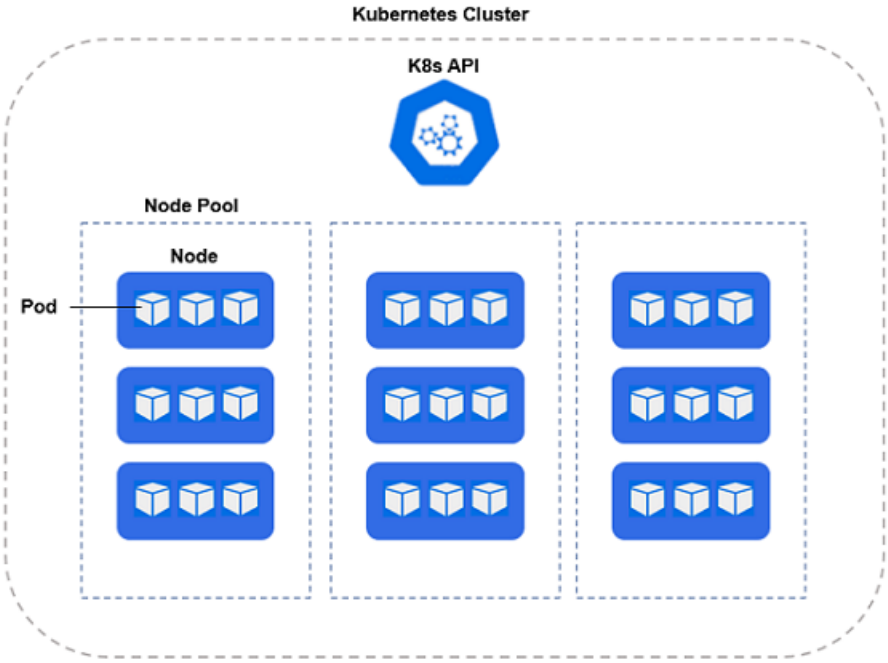
When an application sends a request, the leader node dedicates a thread to process the request. All other threads remain ready for subsequent requests. The leader routes the query through parsing and planning. The planner determines the steps that the query requires, for example, whether a hash join, merge join, or an aggregation step is needed. The planner passes the final query plan to the code generator, which assembles the groups of steps into segments. The code generator then packages all of the segments for the query into a stream. The leader sends the stream to all of the nodes in the cluster and to its own slices. The nodes process the stream in parallel; each node dedicates a thread to process each segment. The nodes then return the results to the leader to send to the application.



Kubernetes Concepts

To set up the Kubernetes (K8s) infrastructure needed to integrate with Anzo, you use scripts that are supplied by Cambridge Semantics and the API for your preferred cloud service provider (CSP) to deploy a K8s cluster. The cluster includes a K8s API server, which manages all communication for the cluster.

In the cluster, you create a number of node pools or node groups. A node pool or node group is a group of nodes within a cluster that all have the same configuration. Different node pools are designed based on machine types and specific properties to be set on each node. The nodes are tuned to host a particular type of pod. A pod is an instance of an application, i.e., a container of images. The diagram below shows a high level view of a K8s cluster:



Node pools can be configured so that they are static or autoscaling. In static node pools, the nodes are deployed in the K8s cluster and remain provisioned even if they do not run an application. If a node pool is configured with an autoscaler, nodes are not deployed unless resources are requested. When the resources are no longer in use, the autoscaler deprovisions the nodes.

For more information about node pools and other requirements, see Anzo Kubernetes Requirements.

Platform Requirements

The topics in this section provide information about the compatibility between components, the platform-level requirements that are shared between components, the component-specific requirements for the Anzo server, AnzoGraph, Distributed Unstructured, and Elasticsearch host servers, and the Kubernetes requirements for the optional dynamic deployment infrastructure.

In this section:

Provisioning Multiple Platform Environments

Cambridge Semantics recommends that you create separate development, staging, potentially user acceptance testing, and production environments. Separating environments is essential for promoting organized development, safeguarding data, and minimizing disruptions in data processing solutions.

When migrating artifacts between environments, administrators can perform a bulk export (and import) by assembling a migration package that includes any number and type of artifacts and their related entities. The export configuration is maintained at the package level and applied to all of the contained artifacts, which means the configuration can be reused as artifacts are added to or removed from the package. For more information about migrating artifacts, see Migration Packages in the Administration Guide.

Tip

Administrators can also set a version environment tag that is displayed at the top of the Anzo application and that Anzo adds to archived versions of entities so that users can distinguish between artifacts from different environments. For more information, see Versioning in the Administration Guide.

Component Compatibility Matrix

This topic shows the compatibility between Anzo and AnzoGraph versions as well as the supported operating systems and Java versions for each of the Anzo platform components.

- Anzo and AnzoGraph Version Compatibility
- Operating System Compatibility
- Java Compatibility

Anzo and AnzoGraph Version Compatibility

Release	AnzoGraph < 2.5	AnzoGraph 2.5.x	AnzoGraph 3.1.x
Anzo 6.0	✗	✗	✓

Operating System Compatibility

The requirements below apply to **static** installations of Anzo, AnzoGraph, and Anzo Distributed Unstructured (DU). Containerized deployments do not depend on the operating system of the host server since the required software is included in the images.

Release	EL 7.9	EL 8.x	EL 9.3+
Anzo 6.0	✓	✓	✓
AnzoGraph 3.1.x	✗	✗	✓
Anzo DU 6.0	✓	✓	✓

Java Compatibility

The requirements below apply to **static** installations of Anzo, AnzoGraph, and Anzo Distributed Unstructured (DU). Containerized deployments do not depend on host server software since the required software is included in the images. Java OpenJDK and JVM environments are supported.

> **Note**
> Java 21 is included in the Anzo installer, and Java 17 is included in the Anzo DU installer. If a Java 17 environment is not already installed the DU host servers, the packaged OpenJDK 17 will be installed. You do not need to install Java on the host servers for those components. The AnzoGraph installer does not include Java, and instructions for installing the appropriate version are included in the pre-installation instructions.

Release	Java 8	Java 17	Java 21
Anzo 6.0	✗	✗	✓
AnzoGraph 3.1.x	✗	✗	✓
Anzo DU 6.0	✗	✓	✗

Shared Requirements

This section provides details about the platform-level requirements. Putting these requirements in place ensures seamless interoperability between all of the Anzo components.

In this section:

Shared File Storage Requirements

The Anzo server and all AnzoGraph, Anzo Distributed Unstructured, and Elasticsearch servers need to have read and write access to a storage system that can be shared between the components. Cambridge Semantics strongly recommends that you deploy an NFS and then mount it to Anzo and each of the AnzoGraph, Anzo DU, and Elasticsearch servers that make up the Anzo platform. Mounted network file systems offer the best support and performance for reading and writing files.

Important
If you plan to set up Kubernetes (K8s) integration for dynamic deployments of Anzo components, an NFS is required. Other file and object stores are not supported for K8s deployments at this time.

Though users can connect to source files that are stored on systems such as Hadoop Distributed File Systems (HDFS), File Transfer Protocol (FTP or FTPS) systems, Google Cloud Platform (GCP) storage, and Amazon Simple Cloud Storage Service (S3), Anzo uploads a copy of those files to a graphmart staging area on the configured NFS so that the files can be used by all of the platform components. The NFS Guidelines section below describes the NFS requirements.

NFS Guidelines

This section describes the key recommendations to follow when creating an NFS for the Anzo platform:

- Use NFS Version 4 or later.
- Provision SSD disk types for the best performance.
- When determining the size of the NFS, consider your workload and use cases. There needs to be enough storage space available for any source data files, exported RDF data files, Elasticsearch indexes, and any other files that you plan to store on the NFS. In addition, consider that cloud-based NFS servers often have better performance if you over-provision

resources. When using a cloud-based VM for your NFS, it can be beneficial to provision more CPU, disk space, and RAM than required to store your artifacts.

- For integration between Anzo applications, the Anzo service account must have read and write access to the NFS. In addition, it is important to set the Anzo account User ID (UID) and Group ID (GID) to **1000** so that the owner of files that are written to the shared file store is UID 1000. For more information about the user account requirements, see Service User Account Requirements.

  Note

  If you are unable to map the Anzo service account UID and GID to 1000, you can modify **anonuid** and **anongid** in the NFS server export table to map all requests to 1000. To do so, add the following line to `/etc/exports` on the NFS server:

  ```
  <mount_point> *(insecure,rw,sync,no_root_squash) x.x.x.x(rw,all_
  squash,anonuid=1000,anongid=1000
  ```

  For example:

  ```
  /global/nfs/data *(insecure,rw,sync,no_root_squash) x.x.x.x(rw,all_
  squash,anonuid=1000,anongid=1000)
  ```

Service User Account Requirements

For consistent and appropriate access management across current and future Anzo platform environments, it is important for the IT organization to create an enterprise-level, LDAP-managed Anzo service user account. The service account should be used when installing and running the Anzo server and each of the components in the platform (AnzoGraph, Elasticsearch, and Anzo Distributed Unstructured).

Account Access Requirements

The service account should not have root user privileges but does need the following access:

- The account must have read and write permissions for the Anzo platform installation directories.
- The account must have a home directory on the Anzo and AnzoGraph host servers.
- The account must have read and write access to the shared file store, such as the NFS mount location, where all Anzo components will read and write files during the data onboarding processes. For more information about the shared file system requirements, see Shared File Storage Requirements.

  Important
  Set the Anzo account User ID (UID) and Group ID (GID) to **1000**. For integration between Anzo components, it is important that the owner of files that are written to the shared file store is UID 1000, especially if you are considering Kubernetes-based deployments of Anzo components.

Note on Using Temporary Local User Accounts

If necessary, you can create a temporary user account on the platform component host servers. Note, however, that using local accounts can cause issues when migrating Anzo or integrating with a central LDAP server. Any local user account must also meet the requirements in Account Access Requirements above.

Anzo Server Requirements

The topics in this section provide requirements and recommendations for the Anzo host server.

In this section:

Anzo Hardware, Software, and Firewall Requirements

This page provides important guidelines to follow when choosing the hardware and software for Anzo host servers.

Hardware Requirements

The following guidelines apply to individual Anzo servers within production and development environments. Your Cambridge Semantics Customer Success manager can help you identify an overall platform deployment configuration that is appropriate for your solution and use cases.

Production Environments

Component	Minimum	Recommended	Description
RAM	64 GB	**128+ GB**	The Anzo system data source is a disk-based graph store (called a journal or volume). When the system source is queried, Anzo swaps the data from disk to memory on demand. Choosing a host server with more RAM increases the performance of system queries because the OS can store the journal data in its file cache, avoiding the need for Anzo to swap data from disk to memory. In addition, RAM is required to hold intermediate results for join queries.

Component	Minimum	Recommended	Description
Disk Space: Anzo Install Path	100 GB	**500+ GB**	The Anzo server installation disk needs to have enough space to store the Anzo system data source, log files, plugins, and the Anzo client.
Disk Space: Shared File System	500 GB	**1+ TB**	The shared file system stores all of the files that are shared between Anzo and all AnzoGraph, Anzo Distributed Unstructured, and Elasticsearch servers. For more information, see Shared File Storage Requirements.
vCPU	16	**32**	Once you provision sufficient RAM, performance depends on CPU capabilities. Keep in mind that you are provisioning for both a production database and a busy application server. A greater number of cores and high clock speed can make a dramatic difference in performance when there are many concurrent Anzo users.
Architecture	64-bit	**64-bit**	Anzo is supported only on 64-bit architecture.

Development Environments

Component	Minimum	Recommended	Description
RAM	32 GB	**64+ GB**	These RAM guidelines assume that the development environment is intended to host smaller data volumes than the production environment and support

Component	Minimum	Recommended	Description
			one or two Anzo users at a time. For development environments with large data volumes and multiple concurrent users, increase the RAM amount.
Disk Space: Anzo Install Path	100 GB	**500+ GB**	The Anzo server installation disk needs to have enough space to store the Anzo system data source, log files, plugins, and the Anzo client.
Disk Space: Shared File System	500 GB	**1+ TB**	Typically the development environment mounts the same shared file system as the production environment.
vCPU	8	**16**	Like the RAM guidelines, these vCPU guidelines assume that the development environment is intended to host smaller data volumes than the production environment and support one or two Anzo users at a time. For development environments with large data volumes and multiple concurrent users, increase the number of vCPU.
Architecture	64-bit	**64-bit**	Anzo is supported only on 64-bit architecture.

Software Requirements

This section lists the software requirements for Anzo servers and client workstations. It also includes important service account information and lists the supported single sign-on providers.

Note

Do not run any other software, including anti-virus software, on the same server as Anzo. Additional software may be run in a development environment with the expectation of lowered Anzo performance. Cambridge Semantics strongly recommends that you do not run additional software on the Anzo server in a production environment.

Component	Minimum	Recommended	Guidelines
Operating System: Anzo Server	RHEL/CentOS 7.9	**RHEL/Rocky 9.3**	Anzo is supported on Enterprise Linux 7.9 – 9.3 operating systems.
Java: Anzo Server	Java 21	**OpenJDK 21 (included in the installer)**	Anzo requires Java version 21. OpenJDK and JVM environments are supported. OpenJDK 21 is included in the Anzo installer and does not need to be pre-installed on the host server.
Web Browser: Client Workstation	Firefox 62+ Chrome 74+ Safari 12+ Chromium-Based	**Chrome 100+**	Use the latest versions of web browsers, especially if you are using a Chromium-based browser, as some older versions will not work with the Anzo user interface components.
Enterprise-Level Anzo Service User Account	N/A	N/A	It is important to work with your IT organization to create an Anzo service user account at the enterprise level. The user account needs to be associated with a central directory server (LDAP) so that it is available across Anzo

Component	Minimum	Recommended	Guidelines
			environments and is managed in accordance with the permissions policies of your company. For more information, see Service User Account Requirements.

Supported Single Sign-On Providers

Anzo supports the following single sign-on (SSO) protocols:

- Basic SSO
- Facebook OAuth
- JSON Web Tokens (JWT)
- Kerberos
- OpenID Connect (OIDC)
- Security Assertion Markup Language (SAML)
- Spring Security OAuth2

Firewall Requirements

The table below lists the TCP ports to open on the Anzo host.

Port	Description	Access Needed...
61616	Anzo port used by the software development kit (SDK) and command line interface (CLI)	Between Anzo and clients.
61617	Anzo SSL port used by the SDK and CLI	Between Anzo and clients.
8022	Anzo SSH service port	Between Anzo and clients.

Port	Description	Access Needed...
8945	Anzo Administration service port	Between Anzo and clients.
8946	Anzo Administration service SSL port	Between Anzo and clients.
80	Application HTTP port	Between Anzo and clients.
443	Application HTTPS port.	Between Anzo and clients.
3389	LDAP port	Between Anzo and the LDAP server.
2551 (optional)	For Anzo Unstructured, this is the port for communication between the Distributed Unstructured (DU) leader node and the worker nodes.	Between the DU leader node (typically installed on the Anzo server) and the DU worker nodes.
9393 (optional)	Optional Java Management Extensions (JMX) port. Enable this port if you want to connect to Anzo from a JMX client.	Between Anzo and the JMX client.
9394 (optional)	Optional JMX SSL port. Enable this port if you want to make a secure connection to Anzo from a JMX client.	Between Anzo and the JMX client.
5700	The Anzo protocol (gRPC) port for secure communication between AnzoGraph and Anzo For more information about the communication between Anzo and AnzoGraph, see Firewall Requirements in AnzoGraph Server Requirements.	Between Anzo and the AnzoGraph leader server.

Port	Description	Access Needed...
5600	AnzoGraph's SSL system management port	Between Anzo and the AnzoGraph leader server.

For instructions on installing Anzo, see Installing Anzo.

Licensing Requirements

When the Anzo server is initially installed, a server ID is generated based on a number of system properties, including the user account that runs the installation script. The Anzo server license is tied to that server ID. If Anzo is re-installed (for instance, during an upgrade) by a different user account, a new server ID is generated and the existing license becomes invalid for the current installation. Whenever you upgrade or re-install Anzo, it is important to use the same user account that was used for the initial installation.

If Anzo is inadvertently updated by the wrong user account, see Restoring the Server ID in the Administration Guide for instructions on correcting the issue.

Important

Cambridge Semantics strongly recommends that you do not change the user running Anzo after installation. If it is absolutely necessary, the license can be changed so that it is associated with the new server ID, and Anzo can be restarted once the license is updated. However, using a new server ID resets (or regenerates from non-customer-specific templates) all previously configured OSGI properties to their default values. Changing the user should only be attempted if there is a complete record of all of the customized OSGI properties and their values as well as a thorough change log so that the configuration can be restored if necessary.

For information about the user account requirements, see Service User Account Requirements.

AnzoGraph Requirements

The topics in this section provide requirements and recommendations for AnzoGraph host servers.

In this section:

AnzoGraph Hardware, Software, and Firewall Requirements

This topic lists the minimum requirements and recommendations to follow for setting up static AnzoGraph host servers and cluster environments.

Hardware Requirements

The following guidelines apply to individual AnzoGraph servers. Your Cambridge Semantics Customer Success manager can help you identify an overall AnzoGraph deployment configuration that is appropriate for your solution and use cases.

Component	Minimum	Recommended	Guidelines
RAM	16 GB (small-scale testing only)	**200+ GB**	AnzoGraph needs enough RAM to store data, intermediate query results, and run the server processes. Cambridge Semantics recommends that you allocate 3 to 4 times as much RAM as the planned data size. Do not overcommit RAM on a VM or on the hypervisor/container host. **Tip** For more information about determining the server and cluster size that is ideal for hosting AnzoGraph, see Sizing Guidelines for In-Memory

Component	Minimum	Recommended	Guidelines
			Storage.
Disk Space (Install Path)	40 GB HDD	**200+ GB SSD**	AnzoGraph requires 30 GB for internal requirements. By default, up to 20 GB of disk space (in `install_path/spill`) is used for logging historical system information for analysis. The amount of additional disk space required for any load file staging, data persistence, or logs depends on the size of the data to be loaded.
vCPU	2	**32** or **64**	Once you provision sufficient RAM and a high-performing I/O subsystem, performance depends on CPU capabilities. A greater number of cores can make a dramatic difference in the performance of file loads and concurrent queries. **Note** Intel processors are preferred, but AnzoGraph is supported on newer Epyc AMD processors. AnzoGraph does not run on older AMD processors.
Networking	10gbE	**20+gbE**	Not applicable for single server installations. Since AnzoGraph is high performance computing (HPC)

Component	Minimum	Recommended	Guidelines
			Massively Parallel Processing (MPP) OLAP engine, inter-cluster communications bandwidth dramatically affects performance. AnzoGraph clusters require optimal network bandwidth. **Important** All servers in a cluster must be in the same network. Make sure that all instances are in the same VLAN, security group, or placement group. In a switched network, make sure that all NICs link to the same Top Of Rack or Full-Crossbar Modular switch. If possible, enable SR-IOV and other HW acceleration methods and dedicated layer 2 networking that guarantees bandwidth.
Shared File System	N/A	N/A	The Anzo file store (shared file system) must be accessible from each AnzoGraph server in the cluster. For more information about the shared file system, see Shared File Storage Requirements.

Software Requirements

The table below lists the software requirements for AnzoGraph servers. Instructions for installing each of the required software components are included in the AnzoGraph installation instructions. See Installing AnzoGraph for more information.

Component	Requirement	Description
Operating System	**RHEL/Rocky Linux 9.3+**	**AnzoGraph 3.1 is not supported on Enterprise Linux 7 or 8.**
Glibc-devel Library	**Installed on all host servers**	For compiling queries, AnzoGraph requires the latest version of the `glibc-devel` library for your operating system.
GNU Binutils	**Installed on all host servers**	To compile and link programs, AnzoGraph requires the latest version of the `binutils` package for your operating system.
OpenJDK or GraalVM	**Version 21 installed on all host servers**	AnzoGraph uses a Java client interface to access data sources. A Java 21 environment is required for using the Java client. AnzoGraph supports OpenJDK 21 and GraalVM 21.
Enterprise-Level Anzo Service User Account	**Created**	It is important to work with your IT organization to create an Anzo service user account at the enterprise level. The service user account needs to be associated with a central directory server (LDAP) so that it is available across Anzo environments and is managed in accordance with the permissions policies of your company. For more information, see Service User Account Requirements.

Optional Software

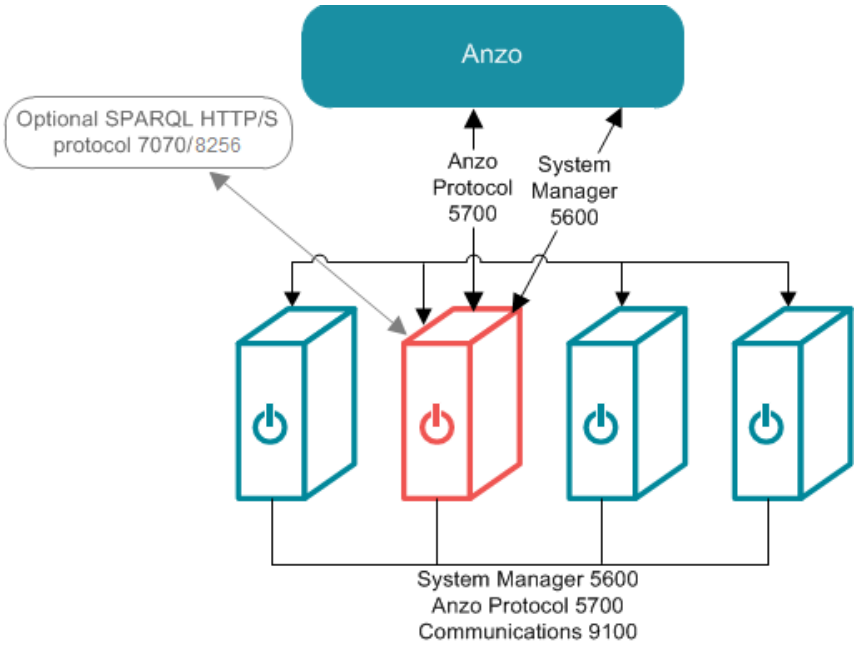
Program	Description
vim	Editor for creating or changing files.
sudo	Enables users to run programs with alternate security privileges.
net-tools	Networking utilities.
psutil	Python system and process utilities for retrieving information on running processes and system usage.
tuned	Linux system service to apply tuning.
wget	Utility for downloading files over a network.

Firewall Requirements

AnzoGraph servers communicate via TCP/IP sockets. AnzoGraph communicates with Anzo via the secure, encrypted, gRPC-based Anzo protocol. Since AnzoGraph is SPARQL-compliant, you also have the option to use standard SPARQL HTTP/S protocol for communication.

> **Important**
> For AnzoGraph clusters, all servers in the cluster must be in the same network. Make sure that all instances are in the same VLAN, security group, or placement group.

Open the TCP ports listed in the table below. This image shows a visual representation of the communication ports:



Port	Description	Required Access
5700	The Anzo protocol (gRPC) port for secure communication between AnzoGraph and Anzo.	The list below describes the access that is needed on port 5700: • Between Anzo and the AnzoGraph leader server. • Between all AnzoGraph servers in the cluster. • Available for AnzoGraph on single node

Port	Description	Required Access
		installations.
5600	AnzoGraph's SSL system management port.	The list below describes the access that is needed on port 5600: • Between Anzo and the AnzoGraph leader server. • Between all AnzoGraph servers in the cluster. • Available for AnzoGraph on single node installations.
9100	AnzoGraph's internal fabric communications port.	The list below describes the access that is needed on port 9100: • Between all AnzoGraph servers in a cluster.

Port	Description	Required Access
		• Available for AnzoGraph on single node installations.
7070 (optional)	Optional SPARQL service HTTP port to enable if you want to give external applications access to AnzoGraph over HTTP.	Between applications and the AnzoGraph leader server.
8256 (optional)	Optional SPARQL service HTTPS port to enable if you want to give external applications SSL access to AnzoGraph and/or use the command line interface, azgi.	Between applications and the AnzoGraph leader server.

Notes on Clusters and Virtual Environments

AnzoGraph requires that all elements of the infrastructure provide the same quality of service (QoS). Do not run AnzoGraph on the same server as any other software, including anti-virus software, except when in single-server mode and with an expectation of lowered performance. Providing the same QoS is especially important when using AnzoGraph in a clustered configuration. If any of the servers in the cluster perform additional processing, the cluster becomes unbalanced and may perform poorly. A single poor performing server degrades the other servers to the same performance level. **All nodes require the same hardware specification and configuration**. Also use static IP addresses or make sure that DHCP leases are persistent.

To ensure the maximum and most reliable QoS for CPU, memory, and network bandwidth, do not co-locate other virtual machines or containers (such as Docker containers) on the same hypervisor or container host. For hypervisor-managed VMs, configure the hypervisor to reserve the available memory for the AnzoGraph server. For clusters, make sure there is enough physical RAM to support all of the AnzoGraph servers, and reserve the memory via the hypervisor.

In addition, running memory compacting services such as Kernel Same-page Merging (KSM) impacts CPU QoS significantly and does not benefit AnzoGraph. Live migrations also impact the performance of VMs while they get migrated. While live migration can provide value for planned host maintenance, AnzoGraph performance may be impacted if live migrations occur frequently. For more information about Kernel Same-page Merging, see https://en.wikipedia.org/wiki/Kernel_same-page_merging.

Tip

Advanced configurations may benefit from CPU pinning on the hypervisor host and disabling CPU hyper-threading. For more information about CPU pinning, see https://en.wikipedia.org/wiki/Processor_affinity. For information about hyper-threading, see https://en.wikipedia.org/wiki/Hyper-threading.

Cambridge Semantics can provide benchmarks to establish relative cluster performance metrics and validate the environment.

For instructions on installing AnzoGraph, see Deploying a Static AnzoGraph Cluster.

Sizing Guidelines for In-Memory Storage

This topic provides guidance on determining the server and cluster size that is ideal for hosting AnzoGraph, depending on the characteristics of your data.

- Memory Sizing Guidelines
- Analyzing Data Characteristics in Load Files

Memory Sizing Guidelines

Since AnzoGraph is a high-performance, in-memory database, it is important to consider the amount of memory needed to store the data that you plan to load. Estimating the amount of memory your workload requires can help you decide what size server to use and whether to use multiple servers. The sections below describe the key points to consider about memory usage and AnzoGraph.

- Data at rest should remain below 50% of the total memory
- AnzoGraph reserves 20% of the memory for the OS
- Memory usage can be high during loads
- Memory usage depends on data characteristics

Data at rest should remain below 50% of the total memory

The data loaded into memory should not consume more than 50% of the total available memory on the instance or across a cluster. **Ideally, the data at rest should use only 25%-30% of the available memory** because query processing and intermediate results can temporarily consume a very large amount of RAM.

AnzoGraph reserves 20% of the memory for the OS

To avoid unexpected shutdowns by the Linux operating system, the default AnzoGraph configuration leaves 20% of memory available for the OS; AnzoGraph will not use more than 80% of the total available memory. Account for this memory buffer in sizing calculations.

Memory usage can be high during loads

During the load streaming process, before duplicates are pruned and triples are moved to their final storage blocks, memory usage temporarily increases and potentially doubles, particularly if the data includes many string values.

Memory usage depends on data characteristics

Memory usage varies significantly depending on the makeup of the data, such as the data types and sizes of literal values, and the complexity of the queries that you run. Triple storage ranges anywhere from 12 bytes per triple to 1 megabyte for a triple that stores pages of text from an unstructured document. For example:

- Triples with integer objects like the following example require about 16 bytes to store in memory.

```
<http://csi.com/resource/person1> <http://csi.com/resource/age> 50
```

- Triples made up of URIs like the following example require about 18 bytes to store in memory.

```
<http://csi.com/resource/person1> <http://csi.com/resource/friend>
<http://csi.com/resource/person100>
```

- Triples with user-defined data types (UDTs) like the following example also require about 18 bytes to store in memory.

```
<http://csi.com/resource/person1> <http://csi.com/resource/height>
"5'8""^^height
```

- Triples with dateTime values like the following example require about 20 bytes to store in memory.

```
<http://www.wikidata.org/entity/Q65949130>
<http://www.wikidata.org/prop/direct/P585>
"1995-01-01T00:00:00Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> .
```

- Triples with long strings like the following example require about 700 bytes to store in memory.

```
<http://dbpedia.org/resource/Keanu_Reeves>
<http://dbpedia.org/ontology/abstract> "Keanu Charles Reeves
```

```
(/keɪˈɑːnuː/ kay-AH-noo; born September 2, 1964) is a Canadian actor, producer,
director and musician.
Reeves is best known for his acting career, beginning in 1985 and spanning more
than three decades.
He gained fame for his starring role performances in several blockbuster films
including comedies
from the Bill and Ted franchise (1989–1991), action thrillers Point Break
(1991) and Speed (1994),
and the science fiction-action trilogy The Matrix (1999–2003). He has also
appeared in dramatic
films such as Dangerous Liaisons (1988), My Own Private Idaho (1991), and
Little Buddha (1993),
as well as the romantic horror Bram Stoker's Dracula (1992)."
```

The table below provides estimates for the number of triples that you can load and query with commonly configured amounts of available RAM. The table also lists the number of triples that could be stored if a data set comprised the example triples above.

Note

The examples below show the number of triples at rest and consider that the data should not consume more than 50% of the available RAM.

RAM	General Estimate	Examples
16 GB	Up to about 100 million triples	Considering that the data at rest should use less than 8 GB RAM, a server with 16 GB total RAM could store: • About 12 million 700-byte triples like the Keanu Reeves example above. • About 475 million 18-byte URI triples like the example above.
32 GB	Up to about 200 million triples	Considering that the data at rest should use less than 16 GB RAM, a server with 32 GB total RAM could store:

RAM	General Estimate	Examples
		• About 24 million 700-byte triples like the Keanu Reeves example above. • About 850 million 20-byte triples like the dateTime example above.
64 GB	Up to about 400 million triples	Considering that the data at rest should use less than 32 GB RAM, a server with 64 GB total RAM could store: • About 48 million 700-byte triples like the Keanu Reeves example above. • About 1.7 billion 20-byte triples.
128 GB	Up to about 800 million triples	Considering that the data at rest should use less than 64 GB RAM, a server with 128 GB total RAM could store: • About 96 million 700-byte triples like the Keanu Reeves example above. • About 3.4 billion 20-byte triples.
256 GB	Up to about 1.5 billion triples	Considering that the data at rest should use less than 128 GB RAM, a server with 256 GB total RAM could store: • About 192 million 700-byte triples like the Keanu Reeves example above. • About 6.8 billion 20-byte triples.
512 GB	Up to about 3 billion triples	Considering that the data at rest should use less than 256 GB RAM, a server with 512 GB total RAM could store: • About 390 million 700-byte triples like the Keanu Reeves

RAM	General Estimate	Examples
		example above. • About 13 billion 20-byte triples.

Analyzing Data Characteristics in Load Files

AnzoGraph enables you to perform pre-load analysis on file-based linked data sets without actually loading the data into memory. You can use this method to run statistical queries, such as counting the number of triples or returning a list of the unique subjects and predicates. Performing a "dry run" of a data load enables you to analyze data set characteristics to help with tasks such as memory sizing. Since the data remains on disk, you can use this method to capture statistics about a large data set without having to deploy an AnzoGraph cluster that has enough memory to store all of the data.

Important Considerations for Analyzing Load Files

- Since AnzoGraph scans the files on disk, queries run much slower than they do when run against data in memory. Consider performance when deciding how many files to query at once and how complex to make the queries.
- Though the pre-load feature does not use memory for storing data, queries that you run against files do consume memory. The server must have sufficient memory available to use for these intermediate query results.
- Unlike loads into the database, pre-load analysis does not prune duplicate triples. Statistics returned for load file queries may differ somewhat from the statistics returned after the data is loaded.

Analysis Query Syntax

Use the following query syntax to analyze load files:

```
SELECT <expression>
FROM EXTERNAL <URI>
```

```
[ FROM EXTERNAL <URI> ]
WHERE { <triple_patterns> }
```

SELECT <expression>

The SELECT clause specifies an expression that returns statistical results such as a count of the total number of triples or the number of distinct predicates. Queries that return values for a specific property may return an error.

FROM EXTERNAL <URI>

The URI in the FROM clause specifies the location of the load file or directory of files. For example, this URI specifies a single file:

```
<file:/anzoshare/data/load/values.ttl>
```

This example specifies a directory of files:

```
<dir:/anzoshare/data/BNorthwind/rdf.ttl.gz>
```

For example, the following query analyzes the files in the rdf.ttl.gz directory for an FLDS. The query counts the total number of triples in the files:

```
SELECT (count (*) as ?triples)
FROM EXTERNAL <dir:/nfs/data/store/LoadGHIB_f5886/rdf.ttl.gz>
WHERE { ?s ?p ?o . }
```

```
triples
-----------
143704445
1 rows
```

Assessing Memory Requirements Based on File Analysis

Although the memory required to load and perform queries on specific data sets will vary based on the size and type of data contained in a data set as well as the type of queries run, you can still obtain a reasonable estimate for the amount of memory you will need to store data set by using the equation below:

```
total_triples x avg_triple_size + total_chars = size_estimate(bytes)
```

Follow the steps below to calculate the values to use in the equation:

1. Count the total number of triples in the files
2. Determine the average triple size
3. Count the number of characters for all strings
4. Calculate the size estimate

Count the total number of triples in the files

As shown in the example above, the following query counts the total number of triples in FLDS load files:

```
SELECT (count (*) as ?triples)
FROM EXTERNAL <dir:/nfs/data/store/LoadGHIB_f5886/rdf.ttl.gz>
WHERE { ?s ?p ?o . }
```

```
triples
-----------
143704445
1 rows
```

Determine the average triple size

The Memory usage depends on data characteristics section above shows some example triples and their estimated size. If you are familiar with the data in the files, you may be able to determine the average size based on the examples. Otherwise, Cambridge Semantics recommends using 30 bytes as the average triple size.

Count the number of characters for all strings

For ASCII characters, AnzoGraph uses about 1-byte of memory to store each character. Counting the number of characters in the load files provides a good estimate of the number of bytes required to store the strings in your data.

```
SELECT (SUM(IF(DATATYPE(?o)=<http://www.w3.org/2001/XMLSchema#string>,
       (STRLEN(?o)),0)) as ?char_count)
FROM EXTERNAL <URI>
WHERE {?s ?p ?o}
```

For example, the following query returns the number of characters in the strings for the FLDS referenced above:

```
SELECT (SUM(IF(DATATYPE(?o)=<http://www.w3.org/2001/XMLSchema#string>,
       (STRLEN(?o)),0)) as ?char_count)
FROM EXTERNAL <dir:/nfs/data/store/LoadGHIB_f5886/rdf.ttl.gz>
WHERE {?s ?p ?o}
```

```
char_count
------------
684348190
1 rows
```

Calculate the size estimate

Once you have counted the triples, determined the average triple size, and counted the characters, use the formula below to estimate the amount of memory needed to store the data at rest:

```
total_triples x avg_triple_size + total_chars = size_estimate(bytes)
```

For example:

```
143,704,445 x 30 + 684,348,190 = 4,995,481,540 bytes
```

This example FLDS requires roughly 5 GB of memory to store the data.

Distributed Unstructured and Elasticsearch Requirements

The Anzo Distributed Unstructured (DU) infrastructure is highly customizable and scalable. The number, size, and configuration of the servers in the environment depends on your unstructured data size, pipeline workload, and performance expectations. DU requires two programs that are installed separately from Anzo.

- A Distributed Unstructured cluster for processing the incoming data.
- Elasticsearch for indexing and searching unstructured document contents.

This section describes DU concepts and includes requirements and recommendations for both applications.

In this section:

DU Cluster Requirements

An Anzo Distributed Unstructured (DU) cluster consists of one leader instance and one or more worker instances. Cambridge Semantics provides an installation script for installing the DU software.

- The **leader** instance is a lightweight program. It is typically installed on the Anzo host server but can be installed on a dedicated server and then connected to Anzo.
- The **worker** instances require significant resources to process the unstructured documents and are typically installed on dedicated servers.

Consider the size of your unstructured data workload when deploying worker host servers. Each worker instance can have multiple server instances to process documents. The table below lists the requirements for DU worker servers.

> **Note**
> Do not run any other software, including anti-virus software, on the DU worker servers. Additional programs running on the worker nodes may severely impact the performance of unstructured pipelines.

Component	Requirement
Operating System	RHEL/CentOS 7.9, RHEL/Rocky 8, or RHEL/Rocky 9
CPU	8+ CPU The more CPU you provision, the more parallelism and higher throughput you can achieve. Anzo DU processes *N* documents in parallel, where *N* is the total number of worker cores in the cluster (minus 1-2 CPU per node for management processes). Since the nature of unstructured documents varies greatly from case to case and the number of annotations per document can vary significantly, Cambridge Semantics recommends that you start with at

Component	Requirement
	least 16 CPU per worker node. If you are deploying servers in a cloud environment, choose compute optimized machines that can be scaled to add CPU if needed.
RAM	16+ GB Unless you plan to process excessively large or complex documents, such as documents with many graphics, you do not need to provision a significant amount of RAM. Typical installations deploy about 2 GB RAM per CPU.
Disk Space	10+ GB
Java 17	Java 17 is included in the Anzo DU installer. Java does not need to be pre-installed on the DU host servers.
Ports	When worker nodes are installed on separate servers, port **2551** needs to be open on those servers to allow communication back to the leader.
File System	The Anzo file store (shared file system) must be accessible from each DU server in the cluster. For more information about the shared file system, see Shared File Storage Requirements.

For instructions on installing Anzo DU, see Installing the Distributed Unstructured Cluster.

Elasticsearch Requirements

Anzo Unstructured uses the Elasticsearch engine to build an index after an unstructured pipeline runs and for running searches on unstructured data that is onboarded to Anzo. When choosing an Elasticsearch host server, consider the following information:

- Generating the index is a lightweight operation compared to document search operations. If you have a light unstructured data workload and do not perform text searches on large amounts of data, installing an Elasticsearch engine on the Anzo host server might be sufficient.
- If you onboard a large number of unstructured documents and plan to perform text searches across a large amount of data, Cambridge Semantics recommends that you install Elasticsearch on a dedicated server.

The table below list the Elasticsearch server requirements:

Component	Requirement
Elasticsearch Version	Versions 7.10.2 – 7.17.13 are supported.
Java	Elasticsearch requires Java 11 or later. The software includes an embedded JDK.
CPU	8+ cores
RAM	64+ GB
Disk Space	100+ GB
Ports	By default, the port range for Elasticsearch requests (http.port) is **9200-9300**. If port 9200 is not available when Elasticsearch is started, Elasticsearch tries 9201 and so on until it finds an accessible port. The Anzo server and the AnzoGraph leader server need to be able to access Elasticsearch on the

Component	Requirement
	HTTP request port that Elasticsearch uses.
File System	The Anzo file store (shared file system) must be accessible from each Elasticsearch server. For more information about the shared file system, see Shared File Storage Requirements.

For instructions on installing Elasticsearch, see Installing Elasticsearch.

Kubernetes Requirements

Anzo integrates with Amazon Elastic Kubernetes Service (EKS), Google Kubernetes Engine (GKE), and Azure Kubernetes Service (AKS) services to offer Kubernetes-based, dynamic deployments of AnzoGraph, Anzo Unstructured with Anzo Agent, and Elasticsearch.

The Kubernetes (K8s) integration automates the provisioning and deprovisioning of the resources and applications that support onboarding and accessing data in Anzo. In a K8s-based environment, Anzo users can activate pre-configured environments on-demand without needing specific technical, cloud platform, or infrastructure deployment skills. In addition, right-sized clusters are automatically created and deleted, avoiding the need to keep instances running indefinitely and reducing the overall cost of maintaining the applications.

The topics in this section provide an overview of K8s concepts, general requirements for integrating K8s with Anzo, and guidance on choosing the compute instances that are ideal for hosting the Anzo applications. For deployment instructions, see Setting up K8s Infrastructure for Dynamic Deployments.

In this section:

Anzo Kubernetes Requirements

This section gives an overview of the general infrastructure requirements for Anzo Kubernetes (K8s) integration. Additional software, network infrastructure, and permission-related requirements are included in the deployment instructions for each of the cloud service providers (see Setting up K8s Infrastructure for Dynamic Deployments).

- Supported Kubernetes Versions
- File Storage Requirements
- Node Pool Requirements
- Container Registry Requirements

Supported Kubernetes Versions

The latest stable Kubernetes versions are supported for all cloud service providers.

File Storage Requirements

A network file system (NFS) is required for shared file storage between Anzo and the dynamic applications. You are required to create the file system. However, Anzo automatically mounts the NFS to the nodes when AnzoGraph, Anzo Unstructured, or Elasticsearch pods are deployed. See Shared File Storage Requirements for more information.

Node Pool Requirements

There are three types of node pools or node groups that you are required to configure for integration with Anzo. In addition to the scripts for creating and configuring the K8s cluster, Cambridge Semantics supplies configuration files to use as templates for defining the policies for each type of node pool. The node pools can be configured as static or autoscaling.

Operator Node Pool

An Operator node pool is tuned to run operator pods. Operator pods manage the application pods and control the K8s resources of the applications that are deployed in the node pools. There is one operator for each application: AnzoGraph, Elasticsearch, and Anzo Agent and Anzo

Unstructured. Anzo deploys and manages the operator pods. With the help of the operators, Anzo orchestrates the provisioning and deprovisioning of the application nodes and pods. Since the operators in the Operator node pool are required to be active at all times, operator pods are designed to be very small and use very few resources. They can be deployed on standard, small-sized cloud instances.

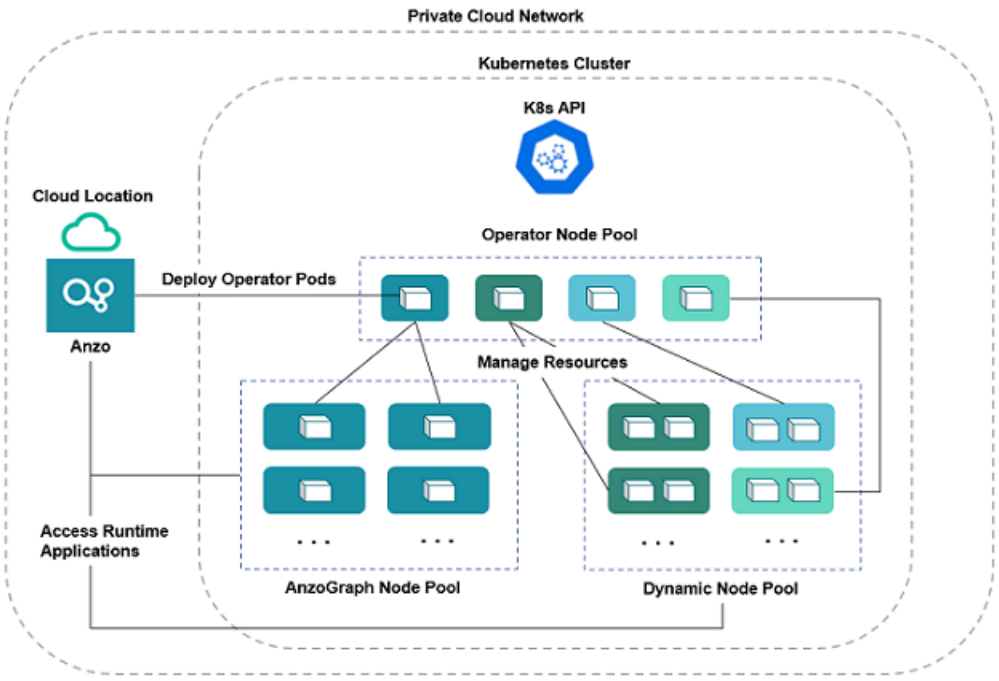
AnzoGraph Node Pool

An AnzoGraph node pool is tuned to run AnzoGraph pods. AnzoGraph node pools are typically configured to auto-scale so that nodes are not deployed unless a user requests an AnzoGraph environment for loading a graphmart or running queries against the data in a graphmart.

Dynamic Node Pool

The Dynamic node pool is tuned to run Elasticsearch, Anzo Agent, and Anzo Unstructured (AU) pods. Dynamic node pools are also typically configured to auto-scale so that nodes are not deployed unless a user requests an environment for running a structured or unstructured pipeline.

The diagram below shows the K8s cluster architecture with the required node pools.



Note

For Amazon EKS deployments, there is a fourth type of required node group. The additional type, called a Common node group, is tuned to run K8s service pods, such as Cluster Autoscalers and Load Balancers.

For guidance on choosing the instance types and sizes for the nodes in the required node pools, see Compute Resource Planning.

Container Registry Requirements

You are not required to set up an internal container registry for Anzo and K8s integration. However, if your K8s cluster will not have outbound internet access for retrieving container images from the Cambridge Semantics repository, you will need to create a container registry through your Cloud Service Provider.

Compute Resource Planning

This section provides guidance on choosing the instance types for the nodes in your node pools.

- Operator Nodes
- AnzoGraph Nodes
- Dynamic Nodes

Operator Nodes

The operator pods are very small. Each operator requires 0.5 CPU. The table below lists the recommended instance types and sizes for a single operator. If you plan to co-locate operators on a single instance, increase CPU accordingly. For example, an instance with 4 CPU can run up to 7 operators (3.5 CPU for operator pods and 0.5 CPU for the auxiliary service).

CSP	Suggested Instance Type	vCPU	RAM	Disk
AWS	m5.large	2	8 GiB	50 GB
GCP	n1-standard-1	1	3.75 GiB	50 GB
Azure	Standard_DS2_v2	2	7 GiB	50 GB

Note
For Amazon EKS deployments, the Suggested Instance Type for Operator nodes is also recommended for nodes in the Common node group. The Common group runs K8s service pods, such as Cluster Autoscalers and Load Balancers, which are very small and require few resources.

AnzoGraph Nodes

Since AnzoGraph is a high-performance, in-memory database, RAM is generally the most critical resource to consider when determining the overall size and number of nodes to use for AnzoGraph environments. Consider the size of the data that you plan to load and then multiply that size by 3 or 4 to determine the total memory requirement. Query processing and intermediate results can temporarily consume a very large amount of memory. For more information about AnzoGraph sizing guidelines, see Sizing Guidelines for In-Memory Storage.

Also, unlike Anzo Unstructured, for example, where leader and worker pods can be colocated on the same node, Cambridge Semantics recommends that only one AnzoGraph pod is run per node. The table below shows a range of cloud instances to choose from that are ideal for running AnzoGraph pods.

CSP	Suggested Instance Range	vCPU Range	RAM Range	Disk
AWS	m5.4xlarge – m5.16xlarge	8 – 64	32 GiB – 256 GiB	100 GB
GCP	n1-standard-8 – n1-standard-64	8 – 64	30 GiB – 240 GiB	100 GB
Azure	DSv2 and DSv3 series	8 – 64	28 GiB – 256 GiB	100 GB

Dynamic Nodes

Nodes in the Dynamic node pool need to be sized to run Anzo Agent pods. An Anzo Agent is a scaled down version of the Anzo server that coordinates the sending of documents to the Anzo Unstructured (AU) worker nodes. Anzo Agent pods require more resources than AU leader and worker and Elasticsearch pods. Each unstructured pipeline deploys a single Anzo Agent pod, and the pod needs to have enough resources to coordinate the pipeline. Anzo Agent pods are typically deployed as one pod per node, while the AU worker and Elasticsearch nodes run multiple pods per

node. The table below lists the recommended instance types and sizes for running the Anzo Agent pods. The recommended instances are also sufficient for running multiple AU and Elasticsearch pods.

CSP	Suggested Instance Type	vCPU	RAM	Disk
AWS	m5.2xlarge	8	32 GiB	100 GB
GCP	n1-standard-8	8	30 GiB	100 GB
Azure	Standard_D8s_v3	8	32 GiB	100 GB

For instructions on setting up the K8s infrastructure, see Setting up K8s Infrastructure for Dynamic Deployments.

Deploying the Platform Components

This section includes instructions for installing the Anzo server and each of the platform components on non-Kubernetes-based static clusters. It also includes instructions on integrating Anzo with Amazon Elastic Kubernetes Service (EKS), Google Kubernetes Engine (GKE), and Azure Kubernetes Service (AKS) for dynamic, Kubernetes-based deployments of components.

Tip

Cambridge Semantics recommends that you start with Deploying and Mounting the Shared File System since the shared storage connection is configured during the Anzo server installation.

In this section:

Deploying and Mounting the Shared File System

Before installing the Anzo server, Cambridge Semantics recommends that you create an NFS for shared file storage. Then mount the NFS in the same location on the Anzo host server and each of the host servers for AnzoGraph, Anzo Distributed Unstructured, and Elasticsearch. During the installation of Anzo, you are prompted to configure the location of the shared data directory. This shared data directory becomes the default File Store in Anzo.

When creating the NFS, follow the NFS Guidelines in Shared File Storage Requirements.

Deploying the Anzo Server

The topics in this section provide instructions for installing, upgrading, and uninstalling the Anzo server as well as tips for securing the environment. For information on server requirements, see Anzo Server Requirements.

In this section:

Installing Anzo

This topic provides instructions for completing the pre-installation requirements and installing and configuring the Anzo server.

1. Meet the Pre-Installation Requirements
2. Install and Configure the Anzo Server
3. (Optional) Route Anzo Ports to the Default HTTP/S Ports
4. Configure and Start the Anzo Service

Meet the Pre-Installation Requirements

Before installing Anzo, make sure that the following requirements are met:

- **Make sure that the host server meets the requirements** in Anzo Hardware, Software, and Firewall Requirements.
- **Confirm that the Anzo service user account is created**. It is important to work with your IT organization to ensure that an Anzo service user account is created at the enterprise level. The user account needs to be associated with a central directory server (LDAP) so that it is available for installing and running Anzo components across environments. For more information, see Service User Account Requirements.
- **Check that the shared file system is created and mounted**. Before installing the Anzo server, Cambridge Semantics recommends that you create an NFS for shared file storage. Then mount the NFS in the same location on the Anzo host server and each of the host servers for AnzoGraph, Anzo Distributed Unstructured, and Elasticsearch. At the end of the Anzo installation during the initial configuration of the server, you specify the location of the shared file system. This shared data directory becomes the default File Store in Anzo. It is not required, but if the shared storage system is not yet created, you may want to set it up before installing Anzo. For information, see Shared File Storage Requirements.

> **Tip**
> If the shared data directory does not exist at the time of initial configuration, you can still complete the configuration. Anzo will create the directory that is specified in the Anzo Shared Data Directory setting. The shared file system will need to be mounted in the specified location or configured as a new file connection before you can onboard data from files, run unstructured pipelines, or export data from graphmarts.

Install and Configure the Anzo Server

Follow the instructions below to install Anzo.

1. Download the installer to the Anzo host server. The installer is an interactive shell script that prompts you to set configuration options for your deployment.
2. If necessary, run the following command to become the Anzo service user:

   ```
   # su <name>
   ```

   Where <name> is the name of the service user. For example:

   ```
   # su anzo
   ```

3. Change directories to the location where you copied the installer and run the following command to make the Anzo installation script executable:

   ```
   chmod +x <script_name>
   ```

   For example:

   ```
   chmod +x Anzo_unix_java21_6_0_0_r202401151700.sh
   ```

4. Run the following command to start the installation wizard:

   ```
   ./<script_name>
   ```

   For example:

   ```
   ./Anzo_unix_java21_6_0_0_r202401151700.sh
   ```

   The script unpacks the JRE and then waits for input before starting the installation.

5. Press **Enter** to start the installation.
6. Review the software license agreement. Press **Enter** to scroll through the terms. At the end of the agreement, type **1** to accept the terms or type **2** to disagree and stop the installation.

   The installer prompts you to specify the components to install:

   ```
   Which components should be installed?
   1: Server [*1]
   2: Client [*2]
   (To show the description of a component, please enter one of *1, *2)
   Please enter a comma-separated list of the selected values or [Enter] for the
   default selection:
   [1,2]
   ```

7. In a comma-separated list, specify the components to install. Item 1 is the Anzo server and item 2 is the Anzo Admin command line client.

   The installer prompts you to specify the Anzo server installation location:

   ```
   Where should the Anzo Server be installed?
   [/opt/Anzo]
   ```

8. Specify the path and directory for the installation. Press **Enter** to accept the default installation path or type an alternate path and then press **Enter**.

   Next, the installer asks if you want to create symlinks:

   ```
   Create symlinks?
   Yes [y, Enter], No [n]
   ```

9. Indicate whether you want the installer to create symlinks. Press **Enter** for yes or type **n** and press **Enter** for no.
10. If you chose to let the installer create symlinks, specify the directory to create the symlinks in. Press **Enter** to accept the default path or type an alternate path and then press **Enter**.

    The installer prompts you to configure the maximum amount of memory that the server can use:

    ```
    Choose the maximum memory that the server can use
    Please enter the maximum amount of RAM memory that the Anzo server may use.
    ```

```
The minimum amount currently supported is 1024 MB. 69995 MB are available.
Maximum Memory in MB
[17499]
```

11. Specify the maximum amount of memory (in MB) that the server can use and then press **Enter**. The installation wizard lists the total RAM available. To meet the minimum memory requirement, the wizard chooses 1/4 of the total memory as the default value. Cambridge Semantics recommends that you allocate at least 1/2 of the total memory to Anzo.

    The wizard installs the components that you selected and then asks if you want to install the optional backend user management components, LDAP and/or Keycloak.

```
Choice of Optional User Management Backends
Include Embedded Ldap Components:
Yes [y, Enter], No [n]
```

12. If you want to install the embedded LDAP components, press **Enter**. If you do not want to install LDAP, type **n** and then press **Enter**.

    The wizard asks if you want to install Keycloak components.

```
Include Embedded Keycloak Components:
Yes [y, Enter], No [n]
```

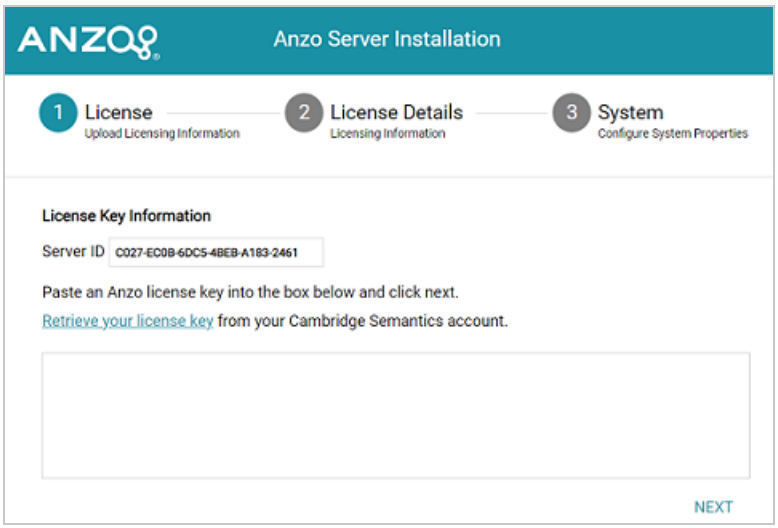
13. If you want to install the embedded Keycloak components, press **Enter**. If you do not want to install Keycloak, type **n** and then press **Enter**.

    The wizard asks if you want to start the Anzo services.

14. Press **Enter** to start the Anzo services. When prompted, open a browser and go to the following URL to open the license administration wizard.

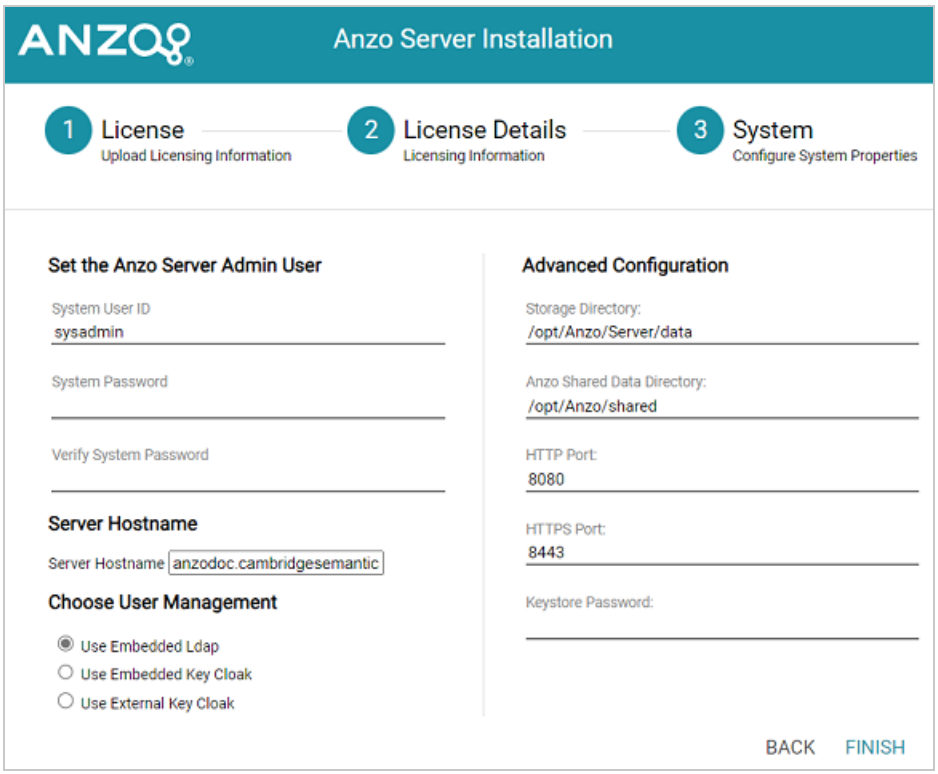
```
http://<hostname>:8945/
```

    Where <hostname> is the host server DNS name or IP address. The License Key Information screen is displayed. For example:



15. Paste your license key into the box and then click **Next**. If necessary, you can obtain the license key by clicking **Retrieve your license key** and logging in to your Cambridge Semantics account.

16. The wizard displays your license details. Review the details and then click **Next**. The wizard displays the System Configuration screen:



17. On the left side of the screen in the **System Password** and **Verify System Password** fields, specify the password to use for the system administrator, **sysadmin**.

> **Important**
>
> **Do not change the System User ID**. It must be **sysadmin**. The sysadmin user account has permission to access all features in the main Anzo application as well as administrative functions in the Administration application. In addition, the sysadmin user has read and write access to all of the artifacts (Data Sources, Models, Pipelines, etc.) that are created by all Anzo users. For more information about the account, see System Administrator in the Administration Guide.

18. Next, review the **Server Hostname** that is selected for this server. If the field is blank, you can type the name. If multiple DNS entries are defined for the server, clicking the field presents a drop-down list where you can select the name for Anzo to use. If you plan to use Keycloak and/or set up single sign-on authentication, this is the name that those clients will use to redirect to Anzo.

19. Below the server hostname are the user management options. The options that are displayed depend on whether you installed the embedded LDAP or Keycloak components.
    - Use Embedded Ldap
    - Use Embedded Key Cloak
    - Use External Key Cloak

20. On the right side of the screen under **Advanced Configuration**, configure the following settings as needed:
    - **Storage Directory**: This setting configures the location where system data, like the binary store and system journal or volume, is stored. The default location is `<install_path>/Server/data`. You can specify an alternate location by typing a new path and directory.
    - **Anzo Shared Data Directory**: This setting specifies the base location of the shared file storage (as described in Shared File Storage Requirements). The default value is

`/opt/Anzo/shared`. Change the default value to the correct location for your shared directory. For example, `/opt/anzoshare/data`. If the specified location does not exist, Anzo will create it. Later you can mount the directory to the specified location or configure another location as a new File Connection (see Connecting to a File Store in the Administration Guide for information).

- **HTTP Port**: This setting specifies the HTTP port for Anzo. The default port is 8080.
- **HTTPS Port**: This setting specifies the HTTPS port for Anzo. The default port is 8443.
- **Keystore Password**: This setting specifies the custom password to use for the Anzo key and trust stores. The password can be changed in the future. See Regenerate the Internal Server Secret in the Administration Guide for information.

21. Click **Finish**. The wizard configures and restarts the server. The process may take several minutes. Once the server is running, the browser displays the Anzo application login screen.

(Optional) Route Anzo Ports to the Default HTTP/S Ports

If you have a load balancer that reroutes traffic on the server and you want users to be able to access Anzo over HTTP/S without having to specify a port (8080 or 8443) in the connection URL, you can configure the firewall to forward HTTP requests to port 8080 and HTTPS requests to port 8443 automatically. This section provides instructions for rerouting ports via the iptables or firewalld interfaces.

> **Note**
> Root user privileges are required to complete this task.

Re-route ports using the iptables interface

Run the following commands to route the Anzo ports via the iptables interface:

```
# iptables -A PREROUTING -t nat -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 8080
# iptables -A PREROUTING -t nat -i eth0 -p tcp --dport 443 -j REDIRECT --to-port 8443
# iptables-save > /etc/sysconfig/iptables
```

Re-route ports using the firewalld interface

Run the following commands to route the Anzo ports via the firewalld interface:

```
# firewall-cmd --permanent --add-forward-port=port=443:proto=tcp:toport=8443
# firewall-cmd --permanent --add-forward-port=port=80:proto=tcp:toport=8080
# firewall-cmd --reload
```

Configure and Start the Anzo Service

The last step in the configuration is to implement the Anzo systemd service. It is important to set up the service so that the server starts automatically as the Anzo service user. In addition, the service is configured to tune user resource limits (ulimits) for the Anzo process. Follow the instructions below to implement and start the service.

> **Note**
> Root user privileges are required to complete this task.

1. If Anzo is running, run the following command to stop the server:

   ```
   ./opt/Anzo/Server/AnzoServer -stop
   ```

2. Create a file called **anzo-server.service** in the `/usr/lib/systemd/system` directory. For example:

   ```
   # vi /usr/lib/systemd/system/anzo-server.service
   ```

3. Add the following contents to anzo-server.service. Placeholder values are shown in **bold**:

   ```
   [Unit]
   Description=Service for Anzo server.
   After=syslog.target network.target local-fs.target remote-fs.target nss-
   lookup.target

   [Service]
   Type=simple
   RemainAfterExit=yes
   LimitCPU=infinity
   LimitNOFILE=65536
   LimitAS=infinity
   LimitNPROC=65536
   LimitMEMLOCK=infinity
   LimitLOCKS=infinity
   ```

```
LimitFSIZE=infinity
WorkingDirectory=/<install_path>
UMask=0007
ExecStart=/<install_path>/Server/AnzoServer start
ExecStop=/<install_path>/Server/AnzoServer stop
User=<service_user_name>
Group=<service_user_name>

[Install]
WantedBy=default.target
```

Where **<install_path>** is the Anzo installation path and directory and **<service_user_name>** is the name of the Anzo service user. For example:

```
[Unit]
Description=Service for Anzo server.
After=syslog.target network.target local-fs.target remote-fs.target nss-
lookup.target

[Service]
Type=simple
RemainAfterExit=yes
LimitCPU=infinity
LimitNOFILE=65536
LimitAS=infinity
LimitNPROC=65536
LimitMEMLOCK=infinity
LimitLOCKS=infinity
LimitFSIZE=infinity
WorkingDirectory=/opt/Anzo
UMask=0007
ExecStart=/opt/Anzo/Server/AnzoServer start
ExecStop=/opt/Anzo/Server/AnzoServer stop
User=anzo
Group=anzo

[Install]
WantedBy=default.target
```

4. Save and close the file, and then run the following commands to start and enable the new service:

```
# systemctl start anzo-server.service
```

```
# systemctl enable anzo-server.service
```

The client displays a message such as the following:

```
Created symlink from /etc/systemd/system/default.target.wants/anzo-
server.service to
/usr/lib/systemd/system/anzo-server.service.
```

Once the service is enabled, the Anzo server services are started. The server can be stopped and started with the following systemctl commands: `sudo systemctl stop anzo-server` and `sudo systemctl start anzo-server`.

Once the all of the Anzo services are running, the Server Settings screen in the Admin application is displayed. In the application, you can change settings as needed, add connections to other platform components like AnzoGraph and Elasticsearch, and manage user accounts. See Getting Started with Admin after a New Installation in the Administration Guide for information on completing the important administrative tasks.

Note

For tips on strengthening the security of Anzo environments after installation, see Securing an Anzo Environment.

Securing an Anzo Environment

This topic lists the recommended procedures to follow to strengthen the security of Anzo environments.

- Set Up Firewall Rules
- Review the Default Access Policies
- Replace the Self-Signed Certificate with a Trusted Certificate
- Use Query Contexts to Store Sensitive Information

Set Up Firewall Rules

In order to protect the environment from malicious systems and prevent man-in-the-middle attacks or leaking of data source credentials, firewall rules should be configured for the Anzo network. Rules should allow outbound connections only to trusted data sources and services. For information about the ports that need to be opened for inbound and outbound connections to support normal operations, see Firewall Requirements in Anzo Requirements.

Review the Default Access Policies

Default Access Policies are the security policies that are applied by default to the artifacts that are stored in Anzo. Artifacts are all of the objects that are created when connections to data sources and applications are made and when data is onboarded to Anzo. For example, when users connect to a database or a file source, those connections are stored as artifacts, and when the data from a data source is ingested, the resulting schema, model, graphmart, and any generated datasets are also artifacts. All artifacts of the same type are stored in a particular registry, and each registry has a Default Access Policy associated with it. The Default Access Policies control the base permissions to assign to an artifact when it is created—before permission inheritance and user-configured sharing is applied. For more information about Default Access Policies, permission inheritance, and sharing, see Artifact Access Control Concepts in the Administration Guide.

By default, most Default Access Policies give the creator of an artifact Admin rights to that artifact, meaning the creator can view, modify, and delete that artifact. In addition, the Everyone role (i.e. all authenticated users) is given View permissions for the artifacts, meaning all authenticated users can see that an artifact exists but they cannot modify or delete it. After installation and before new artifacts are created, Cambridge Semantics recommends that you review the Default Access Policy configuration for your server and make any desired modifications. For more details and instructions, see Managing Default Access Policies in the Administration Guide.

Replace the Self-Signed Certificate with a Trusted Certificate

Anzo installations include a self-signed certificate that can be replaced with your own trusted file. For instructions on replacing the default certificate, see Replacing the Anzo Certificate in the Administration Guide.

Use Query Contexts to Store Sensitive Information

When you connect to data sources with manually written Graph Data Interface (GDI) queries (see Onboarding or Virtualizing Data with SPARQL Queries in the User Guide), you may be required to include sensitive connection and authorization information such as keys, tokens, and user credentials. When configuring data layers or steps, Cambridge Semantics strongly recommends that you store all sensitive connection and authorization values in a Query Context and then refer only to the context keys in GDI queries. Values in Query Contexts are abstracted from the requests that are sent to the data source and AnzoGraph. Any values that are specified directly in a query are transmitted as part of the request. For details about Query Contexts, see Using Query Contexts in the User Guide.

Upgrading Anzo

Before you upgrade Anzo, Cambridge Semantics recommends that you make a backup copy of the current Anzo installation in case you have issues and need to revert to the original version. There are three commonly used methods for backing up Anzo:

- Some users choose to make a copy of the Anzo system volume or journal, `<install_path>/Server/data/journal/anzo.jnl`. If you keep a copy of anzo.jnl, you can restore the original Anzo version by reinstalling that release and then copying the backed up journal file into the installation.
- Some users choose to copy or create a tarball of the entire Anzo installation directory, `<install_path>/Anzo`. A backup of the directory can be large, however, and you might want to remove log files to reduce the overall size of the directory before copying or compressing it. If you keep a copy of `<install_path>/Anzo`, you can restore that version by uninstalling the new version and moving the backed up directory to the original installation location.
- Some users choose to take a snapshot of the application disk.

Follow the instructions below to upgrade Anzo.

Important
Complete the steps below as the Anzo service user. When Anzo is initially installed, a server ID is generated based on a number of system properties, including the user account that runs the installation script. The Anzo server license is tied to that server ID. If Anzo is re-installed (for instance, during an upgrade) by a different user account, a new server ID is generated and the existing license will no longer be valid for the installation.

1. Stop Anzo if it is running. Then copy the new Anzo installation script to the server and run the following command to make the script executable:

```
chmod +x <file_name>
```

2. Run the following command to start the installation wizard and perform the upgrade:

```
./<file_name>
```

   The wizard unpacks the JRE and then waits for input before starting the upgrade.

3. Press **Enter** to start the upgrade. The wizard detects the existing installation and asks if you want to update it.

4. Press **Enter** to update the existing installation.

5. Review the software license agreement. Press **Enter** to scroll through the terms. At the end of the agreement, type **1** and press **Enter** to accept the terms or type **2** and press **Enter** to disagree and stop the update.

   The installer prompts you to specify the components to upgrade:

```
Which components should be installed?
1: Server [*1]
2: Client [*2]
(To show the description of a component, please enter one of *1, *2)
Please enter a comma-separated list of the selected values or [Enter] for the
default selection:
[1,2]
```

6. In a comma-separated list, specify the components to upgrade. Item 1 is the Anzo server and item 2 is the Anzo Admin command line client.

   > **Note**
   > If you exclude a component that is currently installed, that component will not be upgraded. The existing component will not be removed from the server.

7. Specify the maximum amount of memory (in MB) that the server can use and then press **Enter**. The wizard lists the amount of memory you have dedicated to the existing Anzo installation. You can type a different value if necessary, and then press **Enter**. The wizard starts the upgrade and then asks if you want to start the server automatically when the upgrade completes.

8. Press **Enter** to start Anzo when the upgrade completes. If you do not want to start the server, type **n** and then press **Enter**. The setup wizard completes the upgrade process.

> **Note**
> During the upgrade, experimental features that were enabled in the previous version are reset to disabled in the new version. For more information and instructions on re-enabling features, contact your Cambridge Semantics Customer Success representative.

Uninstalling Anzo

Before you uninstall Anzo, you may want to make a backup copy of the current Anzo installation. There are three commonly used methods for backing up Anzo:

- Some users choose to make a copy of the Anzo system volume or journal, `<install_path>/Server/data/journal/anzo.jnl`. If you keep a copy of anzo.jnl, you can restore the original Anzo version by reinstalling that release and then copying the backed up journal file into the installation.
- Some users choose to copy or create a tarball of the entire Anzo installation directory, `<install_path>/Anzo`. A backup of the directory can be large, however, and you might want to remove log files to reduce the overall size of the directory before copying or compressing it. If you keep a copy of `<install_path>/Anzo`, you can restore that version by uninstalling the new version and moving the backed up directory to the original installation location.
- Some users choose to take a snapshot of the application disk.

Follow the steps below to uninstall Anzo.

> **Important**
> Complete the steps below as the Anzo service user.

1. Run the following command to begin the uninstall process:

   ```
   ./<install_path>/uninstall
   ```

2. Press **Enter** to confirm that you want to uninstall Anzo. The wizard asks if you want to clear the Anzo installation directory and user and configuration files.
3. Press **Enter** if you want the wizard to remove the entire Anzo installation directory as well as all configuration and user files. Type **n** and then press **Enter** if you do not want the wizard to remove the installation directory.

The wizard uninstalls Anzo.

Deploying a Static AnzoGraph Cluster

The topics in this section provide instructions for installing, upgrading, and uninstalling AnzoGraph as well as tips for securing AnzoGraph environments. For information about AnzoGraph requirements, see AnzoGraph Requirements.

For instructions on setting up the Kubernetes infrastructure so that AnzoGraph clusters can be launched on-demand, see Setting up K8s Infrastructure for Dynamic Deployments.

In this section:

Installing AnzoGraph

The topics in this section guide you through installing AnzoGraph on a single server or cluster. If you are installing AnzoGraph for the first time on a new server, make sure that you complete each of the procedures in this section. Perform the prerequisite configuration, install the AnzoGraph software, and then complete the post-installation configuration. The last topic includes recommendations to follow to strengthen the security of the environment after AnzoGraph is installed and configured.

- Complete the Pre-Installation Requirements
- Install AnzoGraph
- Complete the Post-Installation Configuration
- Securing an AnzoGraph Environment

Complete the Pre-Installation Requirements

This page describes the installation requirements and other important information to know before you install AnzoGraph. The list below summarizes the requirements and recommendations:

- Make sure that each host server meets the requirements in AnzoGraph Hardware, Software, and Firewall Requirements.
- Certain software packages are required to be installed before the AnzoGraph installation. The installer will not run until these prerequisites are installed. See Prerequisite Software for details and instructions.
- Additional dependencies are required to be installed to support extensions like Apache Arrow integration and the Data Science functions that are used for data profiling. However, Cambridge Semantics recommends that you deploy these dependencies after AnzoGraph is installed because the installation includes a `.repo` file that can aid you in the installing the packages. See Post-Installation C++ Dependencies for details.
- When the installer is run with elevated privileges (`sudo` mode), the installer can complete the installation as well as some important post-installation configuration so that AnzoGraph is

running when the installation is finished. See Installation Modes and User Accounts for details.

- If you are installing AnzoGraph in a clustered setup, make note of the IP addresses for each of the servers in the cluster. The installation wizard prompts you to enter the IP addresses during the installation. In addition, choose one server to be the leader server.

Prerequisite Software

The following software must be installed on the host servers before AnzoGraph is installed.

- Java 21 Virtual Environment
- GNU C Devel Library
- GNU Binutils Library

Java 21 Virtual Environment

AnzoGraph requires a Java 21 virtual environment. OpenJDK 21 and GraalVM 21 are supported. For example, you can run the following command to install OpenJDK 21. **Install the JVM on all servers in the cluster.**

```
sudo dnf install java-21-openjdk
```

> **Tip**
> You do not need to set the `$JAVA_HOME` environment variable on the servers. The AnzoGraph system management daemon (azgmgrd) references `$JAVA_HOME`, and it is set when systemd services are configured as part of the installation.

GNU C Devel Library

AnzoGraph requires the latest version of the GNU C `glibc-devel` library for your operating system. Run the following command to install glibc-devel. **Install the library on all servers in the cluster.**

```
sudo dnf install glibc-devel
```

GNU Binutils Library

AnzoGraph requires the GNU `binutils` library. Run the following command to install binutils. **Install the library on all servers in the cluster.**

```
sudo dnf install binutils
```

Post-Installation C++ Dependencies

Additional libraries are required to be installed on all servers to support the C++ extensions that AnzoGraph offers. Though you can install the C++ dependencies before you install AnzoGraph, if you wait until after the installation you can use the included `csi-obs-cambridgesemantics-udxcontrib.repo` file to enable the Cambridge Semantics repository and install the C++ dependencies with or without internet access. For more information, see Installing the C++ Dependencies in the post-installation instructions.

Installation Modes and User Accounts

There are two modes in which you can run the installer, **root (sudo)** or **non-root (current user)**. This section describes both modes and the user account and file ownership implications for each mode.

Mode	Description
Sudo Mode	**Running the installer in sudo mode is the preferred method of installation.** In sudo mode, the installer prompts you to enter the Anzo service user name. Systemd units for the system management daemon (`azgmgrd`) and database (`anzograph`) processes are created in `/etc/systemd/system`. The units start AnzoGraph as the specified user, and file system permissions for the `anzograph` directory and any files that AnzoGraph writes are based on the same user. The services also configure the appropriate resource limits (ulimits) for AnzoGraph and set `$JAVA_HOME` for your OpenJDK installation.
Non-Root Mode	When running the installer as a non-root user, the installer does not create users and file system permissions are based on the user account that performs the installation. If you choose to run the installer in non-root mode, it is important to install and run AnzoGraph with the Anzo service account (see Service User Account Requirements).

Mode	Description
	Example systemd units, in the `<install_path>/examples` directory, are provided as a template for you to configure and enable manually as described in Non-Root Installs: Configuring and Starting the AnzoGraph Services.

Once the prerequisites are in place, proceed to Install AnzoGraph for instructions on installing the software.

Install AnzoGraph

Follow the appropriate instructions below to install AnzoGraph on a single server or cluster. Make sure that each host server meets the requirements in AnzoGraph Hardware, Software, and Firewall Requirements and that you have installed any prerequisite software (see Complete the Pre-Installation Requirements).

> **Note**
> To streamline the installation and configuration of AnzoGraph, Cambridge Semantics recommends that you run the installer with **sudo** privileges. In Sudo Mode, the installer can configure and start systemd services for the AnzoGraph processes. The services configure the appropriate resource limits (ulimits) and set `$JAVA_HOME` to point to the OpenJDK installation. During the installation, you are prompted to provide the Anzo service user name. The provided name is used as the User value in the service units, and file system permissions for the installation directory and AnzoGraph-generated files are also based on the specified user.

- Installing AnzoGraph on a Single Server
- Installing AnzoGraph on a Cluster

Installing AnzoGraph on a Single Server

Follow the steps below to install AnzoGraph on a single server.

1. Download the installer to the AnzoGraph host server. The installer is an interactive shell script that prompts you to choose configuration options for the deployment.
2. Change directories to the location where you copied the script and run the following command to make the script executable:

   ```
   chmod +x <script_name>
   ```

   For example:

   ```
   chmod +x anzograph_linux_3_1_0_r202311010839.sh
   ```

3. Run the following command to start the installation wizard:

   ```
   sudo ./<script_name>.sh
   ```

   The installer first verifies that the prerequisites listed in Prerequisite Software are installed and presents a message if any are missing. For example, on a server where OpenJDK and binutils are installed but the glibc-devel package is missing, the installer displays the following message and the installation is canceled. In this case, the user would follow the instructions in GNU C Devel Library and then restart the installation.

   ```
   Starting Installer ...
   Prerequisite software packages

   The following packages are missing on your system and are required to
   be installed before proceeding with the AnzoGraph installation:
   - glibc-devel

   Canceling the installation, install the missing software, and then run
   the installer again.
   ```

   If the prerequisite software is installed, but the C++ dependencies are not, the installer presents an informational message to let you know the dependencies are required but it is recommended that you install them after the AnzoGraph installation. For example:

   ```
   Starting Installer ...
   Prerequisite software packages

   INFO: The following C++ dependencies are also required and missing from your
   ```

```
system. However, Cambridge Semantics recommends that you install them after
the AnzoGraph installation is complete. Follow the Post-Installation
instructions in the online documentation.
libarchive
libarmadillo12
libboost_filesystem1_80_0
libboost_iostreams1_80_0
libboost_system1_80_0
libflatbuffers2
libhdfs3
libnfs13
libserd-0-0
libsmb2
shadow-utils
```

If all dependencies are installed or you proceed with the installation after getting the informational message, the license agreement is displayed. Press **Enter** to scroll through the text. At the end you are prompted to accept the agreement.

```
Starting Installer ...
Please read the following License Agreement. You must accept the terms of
this agreement before continuing with the installation.

ANZOGRAPH(R) DB
END USER LICENSE AGREEMENT
...
I accept the agreement
Yes [1], No [2]
```

4. Enter **1** to accept the license agreement and continue the installation. The installer prompts you to specify the directory where AnzoGraph should be installed.

```
Where should AnzoGraph DB be installed?
[/opt/cambridgesemantics]
```

> **Note**
> A number of subdirectories and an uninstall script are created inside the directory that you specify. One subdirectory is named `anzograph` and includes the installation files. Because an anzograph directory will be created, you may not want to specify

`/opt/anzograph` as the install location because that will result in an `/opt/anzograph/anzograph` directory.

5. Press **Enter** to select the default installation directory or specify an alternate location and then press **Enter**.

   Next, the wizard prompts you to specify the installation type: single standalone server, leader server, or compute server:

   ```
   Type of server being installed.
   Server Installation Type
   Standalone [1, Enter], Cluster Leader [2], Cluster Compute/Worker [3]
   ```

6. Press **Enter** to accept the default option, **Standalone (1)**.

   The next prompt asks you to create the username for the administrator. This username will be specified when creating the connection between Anzo and AnzoGraph.

   ```
   Set up the AnzoGraph Admin user.
   AnzoGraph DB Admin user
   [admin]
   ```

7. Press **Enter** to accept the default username, **admin**, or type another username (case-sensitive) and press **Enter**. The installer asks you to create a password for the Admin user.

   ```
   AnzoGraph DB Admin password
   ```

8. Enter the case-sensitive password for the Admin user and then press **Enter**. It is recommended that you create a non-trivial secure password that includes some combination of upper and lower case letters and digits. For security purposes, the installer does not echo the characters that you type.

   Important
   There is no limitation placed on the length of the password. However, some special characters, such as `$` and `*`, are treated as parameters in bash. When typing the password, avoid special characters. For more information, see Quoting in the Bash

Reference Manual.

Next, the installer asks if the installation will be used with Anzo:

```
Is this AnzoGraph DB installation intended for use with Anzo?
Yes [y, Enter], No [n]
```

9. Press **Enter** for **Yes**. Answering yes configures AnzoGraph to use the settings that are optimal for Anzo. Answering no would configure the settings that are optimal for AnzoGraph standalone use without Anzo.

   The installer asks if you want the admin user to be able to run queries.

```
Do you want to use admin user to run queries?
Yes [y, Enter], No [n]
```

10. Press **Enter** for **Yes**. Answering yes is required for use with Anzo.

    Next, the installer prompts you to specify the Anzo service user name. The default name is `anzo`.

```
Setup Anzograph service user name
AnzoGraph service user
[anzo]
```

11. Press **Enter** to accept the default name, `anzo`, or type an alternate name and then press **Enter** if a different name is used for the account.

    Next, the installer asks if you want to add a specific configuration setting to the configuration file:

```
Extra configuration settings for server
...
WARNING: Typically, additional settings should only be added
after consultation with Cambridge Semantics to address any
specific requirements for AnzoGraph deployment in your environment
or use for a specific application.
[]
```

12. Press **Enter** if you do not have a setting to add. If you have a custom `setting=value` to add, enter it in the prompt, and then press **Enter**.

    The installer asks if you want to start the system management service, `azgmgrd`, after the installation is complete.

    ```
    Start azgmgrd service?
    Do you want to start azgmgrd systemd service?
    Yes [y], No [n, Enter]
    ```

13. Type **y** (Yes) and press **Enter**.

    The installer asks if you want to start the database service, `anzograph`, after the installation is complete.

    ```
    Start AnzoGraph service?
    Do you want to start AnzoGraph systemd services?
    Yes [y], No [n, Enter]
    ```

14. Type **y** (Yes) and press **Enter**.

    The installer begins installing AnzoGraph based on your responses. When it is finished, the following message is displayed.

    ```
    Setup has finished installing AnzoGraph DB on your computer.
    Finishing installation...
    ```

Now that AnzoGraph is installed and running, proceed to Complete the Post-Installation Configuration to complete the post-installation requirements.

Installing AnzoGraph on a Cluster

Follow the steps below to install AnzoGraph on multiple servers in a cluster. There are two steps in the process:

1. Install AnzoGraph on the Compute Servers
2. Install AnzoGraph on the Leader Server

Install AnzoGraph on the Compute Servers

Follow the instructions below to install AnzoGraph on each compute server.

1. Download the installer to each of the host servers. The installer is an interactive shell script that prompts you to choose configuration options for the deployment.
2. Change directories to the location where you copied the script and run the following command to make the script executable:

   ```
   chmod +x <script_name>
   ```

   For example:

   ```
   chmod +x anzograph_linux_3_1_0_r202311010839.sh
   ```

3. Run the following command to start the installation wizard:

   ```
   sudo ./<script_name>.sh
   ```

   The installer first verifies that the prerequisites listed in Prerequisite Software are installed and presents a message if any are missing. For example, on a server where OpenJDK and OpenLDAP are installed but the glibc-devel package is missing, the installer displays the following message and the installation is canceled. In this case, the user would follow the instructions in GNU C Devel Library and then restart the installation.

   ```
   Starting Installer ...
   Prerequisite software packages

   The following packages are missing on your system and are required to
   be installed before proceeding with the AnzoGraph installation:
   - glibc-devel

   Canceling the installation, install the missing software, and then run
   the installer again.
   ```

   If the prerequisite software is installed, but the C++ dependencies are not, the installer presents an informational message to let you know the dependencies are required but it is recommended that you install them after the AnzoGraph installation. For example:

   ```
   Starting Installer ...
   Prerequisite software packages

   INFO: The following C++ dependencies are also required and missing from your
   ```

```
system. However, Cambridge Semantics recommends that you install them after
the AnzoGraph installation is complete. Follow the Post-Installation
instructions in the online documentation.
libarmadillo12
libboost_filesystem1_80_0
libboost_iostreams1_80_0
libboost_system1_80_0
libgrpc++1
libflatbuffers2
libhdfs3
libnfs13
libserd-0-0
libsmb2
```

If all dependencies are installed or you proceed with the installation after getting the informational message, the license agreement is displayed. Press **Enter** to scroll through the text. At the end you are prompted to accept the agreement.

```
Starting Installer ...
Please read the following License Agreement. You must accept the terms of
this agreement before continuing with the installation.

ANZOGRAPH(R) DB
END USER LICENSE AGREEMENT
...
I accept the agreement
Yes [1], No [2]
```

4. Enter **1** to accept the license agreement and continue the installation.

   The installer prompts you to specify the directory where AnzoGraph should be installed.

```
Where should AnzoGraph DB be installed?
[/opt/cambridgesemantics]
```

> **Note**
> A number of subdirectories and an uninstall script are created inside the directory that you specify. One subdirectory is named `anzograph` and includes the installation files. Because an anzograph directory will be created, you may not want to specify

`/opt/anzograph` as the install location because that will result in an `/opt/anzograph/anzograph` directory.

5. Press **Enter** to select the default installation directory or specify an alternate location and then press **Enter**. **The installation path must be the same on all servers in the cluster**.

   Next, the wizard prompts you to specify the installation type: single standalone server, leader server, or compute server:

   ```
   Type of server being installed.
   Server Installation Type
   Standalone [1, Enter], Cluster Leader [2], Cluster Compute/Worker [3]
   ```

6. At the server installation type prompt, type **3** (**Cluster Compute/Worker**) and press **Enter**.

   Next, the installer prompts you to specify the service user name. The default name is `anzograph`.

   ```
   Setup Anzograph service user name
   AnzoGraph service user
   [anzograph]
   ```

7. Type the name of the Anzo service user. Typically it is **anzo**. Then press **Enter**.

   Next, the wizard prompts you to specify the IP addresses for each of the servers in the cluster:

   ```
   IP Address of nodes in cluster.
   Comma separated list of Cluster Nodes' IP Addresses. Leader node address is
   always first. Order must be the same on all nodes in cluster.
   [127.0.0.1]
   ```

8. Type a comma-separated list of the IP addresses for each server in the cluster. Type the leader server IP address first, followed by each compute IP address. For example, on a cluster with 4 servers where 192.168.2.1 is the leader server:

   ```
   192.168.2.1,192.168.2.2,192.168.2.3,192.168.2.4
   ```

> **Important**
> Make sure that you enter this value exactly the same, with IP addresses in the same order, during the installation on each server.

9. After listing the IP addresses for the cluster, press **Enter**.

   The installer asks if you want to start the system management service, `azgmgrd`. This service runs on all servers in the cluster and manages communication between the servers.

   ```
   Start azgmgrd service?
   Do you want to start azgmgrd systemd service?
   Yes [y], No [n, Enter]
   ```

10. Type **y** (Yes) and press **Enter**.

    The installer begins installing AnzoGraph based on your responses. When it is finished, the following message is displayed.

    ```
    Setup has finished installing AnzoGraph DB on your computer.
    Finishing installation...
    ```

Repeat the steps above to install AnzoGraph on each compute server. When the installation is complete and azgmgrd is running on all compute servers, proceed to Install AnzoGraph on the Leader Server below.

Install AnzoGraph on the Leader Server

Follow the instructions below to install AnzoGraph on the leader server.

1. Download the installer to the host server.
2. Change directories to the location where you copied the script and run the following command to make the script executable:

   ```
   chmod +x <script_name>
   ```

   For example:

   ```
   chmod +x anzograph_linux_3_1_0_r202311010839.sh
   ```

3. Run the following command to start the installation wizard:

```
sudo ./<script_name>.sh
```

The installer first verifies that the prerequisites listed in Prerequisite Software are installed and presents a message if any are missing. For example, on a server where OpenJDK and OpenLDAP are installed but the glibc-devel package is missing, the installer displays the following message and the installation is canceled. In this case, the user would follow the instructions in GNU C Devel Library and then restart the installation.

```
Starting Installer ...
Prerequisite software packages

The following packages are missing on your system and are required to
be installed before proceeding with the AnzoGraph installation:
- glibc-devel

Canceling the installation, install the missing software, and then run
the installer again.
```

If the prerequisite software is installed, but the C++ dependencies are not, the installer presents an informational message to let you know the dependencies are required but it is recommended that you install them after the AnzoGraph installation. For example:

```
Starting Installer ...
Prerequisite software packages

INFO: The following C++ dependencies are also required and missing from your
system. However, Cambridge Semantics recommends that you install them after
the AnzoGraph installation is complete. Follow the Post-Installation
instructions in the online documentation.
libarmadillo12
libboost_filesystem1_80_0
libboost_iostreams1_80_0
libboost_system1_80_0
libgrpc++1
libflatbuffers2
libhdfs3
libnfs13
```

```
libserd-0-0
libsmb2
```

If all dependencies are installed or you proceed with the installation after getting the informational message, the license agreement is displayed. Press **Enter** to scroll through the text. At the end you are prompted to accept the agreement.

```
Starting Installer ...
Please read the following License Agreement. You must accept the terms of
this agreement before continuing with the installation.

ANZOGRAPH(R) DB
END USER LICENSE AGREEMENT
...
I accept the agreement
Yes [1], No [2]
```

4. Enter **1** to accept the license agreement and continue the installation. The installer prompts you to specify the directory where AnzoGraph should be installed.

```
Where should AnzoGraph DB be installed?
[/opt/cambridgesemantics]
```

5. Press **Enter** to select the default installation directory or specify an alternate location and then press **Enter**. **The installation path must be the same on all servers in the cluster**.

   Next, the wizard prompts you to specify the installation type: single standalone server, leader server, or compute server:

```
Type of server being installed.
Server Installation Type
Standalone [1, Enter], Cluster Leader [2], Cluster Compute/Worker [3]
```

6. At the server installation type prompt, type **2** (**Cluster Leader**) and press **Enter**.

   The next prompt asks you to create the username for the administrator. This username will be specified when creating the connection between Anzo and AnzoGraph.

```
Set up the AnzoGraph Admin user.
AnzoGraph DB Admin user
[admin]
```

7. Press **Enter** to accept the default username, **admin**, or type another username (case-sensitive) and press **Enter**. The installer asks you to create a password for the Admin user.

```
AnzoGraph DB Admin password
```

8. Enter the case-sensitive password for the Admin user and then press **Enter**. It is recommended that you create a non-trivial secure password that includes some combination of upper and lower case letters and digits. For security purposes, the installer does not echo the characters that you type.

> **Important**
> There is no limitation placed on the length of the password. However, some special characters, such as `$` and `*`, are treated as parameters in bash. When typing the password, avoid special characters. For more information, see Quoting in the Bash Reference Manual.

Next, the installer asks if the installation will be used with Anzo:

```
Is this AnzoGraph DB installation intended for use with Anzo?
Yes [y, Enter], No [n]
```

9. Press **Enter** for **Yes**. Answering yes configures AnzoGraph to use the settings that are optimal for Anzo. Answering no would configure the settings that are optimal for AnzoGraph standalone use without Anzo.

The installer asks if you want the admin user to be able to run queries.

```
Do you want to use admin user to run queries?
Yes [y, Enter], No [n]
```

10. Press **Enter** for **Yes**. Answering yes is required for use with Anzo.

Next, the installer prompts you to specify the Anzo service user name. The default name is `anzo`.

```
Setup Anzograph service user name
AnzoGraph service user
[anzo]
```

11. Press **Enter** to accept the default name, `anzo`, or type an alternate name and then press **Enter** if a different name is used for the account.

    Next, the wizard prompts you to specify the IP addresses for each of the servers in the cluster:

    ```
    IP Address of nodes in cluster.
    Comma separated list of Cluster Nodes' IP Addresses. Leader node address is
    always first. Order must be the same on all nodes in cluster.
    [127.0.0.1]
    ```

12. Type a comma-separated list of the IP addresses for each server in the cluster. Type the leader server IP address first, followed by each compute IP address. For example, on a cluster with 4 servers where 192.168.2.1 is the leader server:

    ```
    192.168.2.1,192.168.2.2,192.168.2.3,192.168.2.4
    ```

    Important

    Make sure that you enter this value exactly the same, with IP addresses in the same order, during the installation on each server.

13. After specifying the IP addresses for the cluster, press **Enter**.

    Next, the installer asks if you want to add a specific configuration setting to the configuration file:

    ```
    Extra configuration settings for server
    ...
    WARNING: Typically, additional settings should only be added
    after consultation with Cambridge Semantics to address any
    specific requirements for AnzoGraph deployment in your environment
    or use for a specific application.
    []
    ```

14. Press **Enter** if you do not have a setting to add. If you have a custom `setting=value` to add, enter it in the prompt, and then press **Enter**.

    The installer asks if you want to start the system management service, `azgmgrd`, after the installation is complete.

```
Start azgmgrd service?
Do you want to start azgmgrd systemd service?
Yes [y], No [n, Enter]
```

15. Type **y** (Yes) and press **Enter**.

    The installer asks if you want to start the database service, `anzograph`, after the installation is complete.

```
Start AnzoGraph service?
Do you want to start AnzoGraph systemd services?
Yes [y], No [n, Enter]
```

16. Type **y** (Yes) and press **Enter**.

    The installer begins installing AnzoGraph based on your responses. When it is finished, the following message is displayed.

```
Setup has finished installing AnzoGraph DB on your computer.
Finishing installation...
```

Now that AnzoGraph is installed and the cluster is running, proceed to Complete the Post-Installation Configuration to complete the post-installation requirements.

Complete the Post-Installation Configuration

This topic provides instructions for completing the required and optional post-installation configuration of AnzoGraph.

- Adding Drivers for Custom Database Sources
- Installing the C++ Dependencies
- Optimizing the Linux Kernel Configuration for AnzoGraph
- Non-Root Installs: Configuring and Starting the AnzoGraph Services

Adding Drivers for Custom Database Sources

AnzoGraph uses the Graph Data Interface (GDI) Java plugin to connect directly to data sources. The GDI plugin is included in the AnzoGraph installation. Also included in the installation are JDBC drivers for the following databases:

- Databricks
- H2
- IBM DB2
- Microsoft SQL Server
- MariaDB
- Oracle
- PostgreSQL
- SAP Sybase (jTDS)
- Snowflake

To extend the GDI to access custom database sources, JDBC drivers can be added to AnzoGraph. To add a driver, follow the steps below.

1. Copy the .jar file to the `<install_path>/lib/udx` directory on the **leader server**.
2. Restart the database by running the following command. When the database is restarted, the leader broadcasts any new .jar files to the compute servers.

```
sudo systemctl restart anzograph
```

Tip

The `<install_path>/lib/udx` directory on the leader node is a user-managed directory rather than an AnzoGraph-managed directory like `<install_path>/bin` or `<install_path>/internal`. Users can place JDBC drivers and Java or C++ extensions in the `lib/udx` directory any time. Each time the database is restarted, AnzoGraph scans that directory, saves a copy of its contents to the `<install_path>/internal/extensions` directory, and then broadcasts the `internal/extensions` contents from the leader node to the compute nodes. Each restart clears `internal/extensions` and AnzoGraph rescans `lib/udx` to reload `internal/extensions` with the latest plugins.

Installing the C++ Dependencies

Dependencies are required to be installed on all servers in the cluster to support the C++ extensions that AnzoGraph offers, including the remote read (load) and write service, the Data Science functions, and the integration with Apache Arrow. The installer provides a .repo file to aid you in configuring the Cambridge Semantics repository and installing the required software packages with or without internet access.

> **Note**
> The ability to write to the `/etc/yum.repos.d` directory requires root access permissions. See Adding, Enabling, and Disabling a YUM Repository for more information on defining and using yum repositories.

```
libarchive
libarmadillo12
libboost_filesystem1_80_0
libboost_iostreams1_80_0
libboost_system1_80_0
libflatbuffers2
libhdfs3
libnfs13
libserd-0-0
libsmb2
shadow-utils
```

- Install Dependencies via Internet Access to the Cambridge Semantics Repository
- Install Dependencies without Internet Access via the Repository Mirror (tarball)

Install Dependencies via Internet Access to the Cambridge Semantics Repository

Follow the steps below if the AnzoGraph servers have external internet access and you want to install the dependencies directly from the Cambridge Semantics repository.

1. Copy the **csi-obs-cambridgesemantics-udxcontrib.repo** file from the `<install_path>/pre-req/yum.repos.d` directory to the `/etc/yum.repos.d` directory. For example, the following command copies the file from the default installation path to

`/etc/yum.repos.d`:

```
sudo cp /opt/cambridgesemantics/pre-req/yum.repos.d/csi-obs-cambridgesemantics-udxcontrib.repo /etc/yum.repos.d
```

2. Next, run the following command to enable the repository and install the required packages:

   ```
   sudo dnf install --enablerepo=crb $(cat <install_path>/pre-req/rh9-anzograph-requirements.txt)
   ```

   For example, on a server where AnzoGraph is installed in the default location:

   ```
   sudo dnf install --enablerepo=crb $(cat /opt/cambridgesemantics/pre-req/rh9-anzograph-requirements.txt)
   ```

3. Repeat these steps on all servers in the cluster.

Install Dependencies without Internet Access via the Repository Mirror (tarball)

Follow the steps below if the AnzoGraph servers do not have external internet access and you want to install the dependencies from the mirrored Cambridge Semantics repository. The steps below give instructions for copying the repository to each AnzoGraph host server and configuring the `.repo` file accordingly. You can also chose to set up the mirror on a remote server that each of the AnzoGraph servers can access.

1. From a computer that does have internet access, download the dependency tarball, **csi-obs-cambridgesemantics-udxcontrib.rocky9.tar.xz**, from the following Cambridge Semantics Google Cloud Storage location: https://storage.googleapis.com/csi-anzograph/udx/csi-os-contrib/rocky9/2023-03/20230321945/csi-obs-cambridgesemantics-udxcontrib.rocky9.tar.xz.

   You can run the following cURL command to download the tarball:

   ```
   curl -OL https://storage.googleapis.com/csi-anzograph/udx/csi-os-contrib/rocky9/2023-03/20230321945/csi-obs-cambridgesemantics-udxcontrib.rocky9.tar.xz(.sha512)
   ```

2. Also from the computer that has internet access, download the **repomd.xml.key** from the following Cambridge Semantics Google Cloud Storage location: https://storage.googleapis.com/csi-rpmmd-pd/CambridgeSemantics:/UDXContrib/ubi-9/repodata/repomd.xml.key.

You can run the following cURL command to download the file:

```
curl -OL https://storage.googleapis.com/csi-rpmmd-
pd/CambridgeSemantics:/UDXContrib/ubi-9/repodata/repomd.xml.key
```

3. On each of the AnzoGraph servers, create a directory called `/tmp/repo`.
4. Copy **csi-obs-cambridgesemantics-udxcontrib.rocky9.tar.xz** to the `/tmp/repo` directory on each server.
5. Then run the following command to unpack the tarball in the `/tmp/repo` directory:

   ```
   tar -xvf csi-obs-cambridgesemantics-udxcontrib.rocky9.tar.xz
   ```

   The files are unpacked into subdirectories under `/tmp/repo/dl/rocky9/csi-obs-cambridgesemantics-udxcontrib`.
6. Next, copy the **repomd.xml.key** file to the `/tmp/repo/dl/rocky9/csi-obs-cambridgesemantics-udxcontrib` directory on each of the AnzoGraph servers.
7. Now, open the **csi-obs-cambridgesemantics-udxcontrib.repo** file in the `<install_path>/examples/yum.repos.d` directory. The contents of the file are shown below:

   ```
   [csi-obs-cambridgesemantics-udxcontrib]
   name=Contrib directory for CambridgeSemantics AnzoGraph UDX dependencies
   baseurl=https://storage.googleapis.com/csi-rpmmd-
   pd/CambridgeSemantics:/UDXContrib/ubi-9
   gpgkey=https://storage.googleapis.com/csi-rpmmd-
   pd/CambridgeSemantics:/UDXContrib/ubi-9/repodata/repomd.xml.key
   gpgcheck=1
   enabled=1
   ```
8. Edit the **csi-obs-cambridgesemantics-udxcontrib.repo** file contents to replace the **baseurl** and **gpgkey** values so that they point to the repo files that you unpacked in the `/tmp/repo` directory. In addition, change the **gpgcheck** and **enabled** values from 1 to **0**. The contents of the updated file are shown below:

   ```
   [csi-obs-cambridgesemantics-udxcontrib]
   name=Contrib directory for CambridgeSemantics AnzoGraph UDX dependencies
   baseurl=file:///tmp/repo/dl/rocky9/csi-obs-cambridgesemantics-udxcontrib
   gpgkey=file:///tmp/repo/dl/rocky9/csi-obs-cambridgesemantics-
   ```

```
udxcontrib/repomd.xml.key
gpgcheck=0
enabled=0
```

9. Save and close the file.
10. Copy **csi-obs-cambridgesemantics-udxcontrib.repo** from `<install_path>/pre-req/yum.repos.d` to the `/etc/yum.repos.d` directory. For example, the following command copies the file from the default installation path to `/etc/yum.repos.d`:

    ```
    sudo cp /opt/cambridgesemantics/pre-req/yum.repos.d/csi-obs-cambridgesemantics-udxcontrib.repo /etc/yum.repos.d
    ```

11. Next, run the following command to enable the repository and install the required packages:

    ```
    sudo dnf install --enablerepo=crb $(cat <install_path>/pre-req/rh9-anzograph-requirements.txt)
    ```

    For example, on a server where AnzoGraph is installed in the default location:

    ```
    sudo dnf install --enablerepo=crb $(cat /opt/cambridgesemantics/pre-req/rh9-anzograph-requirements.txt)
    ```

Repeat the steps above as needed to install the dependencies on all servers in the cluster.

Optimizing the Linux Kernel Configuration for AnzoGraph

To streamline the configuration of the operating system for peak AnzoGraph performance, the installer includes a **tuned** profile that you can activate. Tuned is a daemon program that uses the `udev` device monitor to statically and dynamically tune operating system settings based on the specified profile. It is highly recommended that you activate the AnzoGraph tuned profile.

> **Tip**
> Tuned and Performance Tuning with Tuned and Tuned-ADM in the RedHat Performance Tuning Guide provide an overview of the tuned daemon and more information on using the tuned service to improve the performance of specific workloads.

Activating the Tuned Profile

The profile, called **azg**, is in the `<install_path>/examples/tuned-profile` directory and consists of two files: `tuned.conf` and `additional-tuneables.sh`. For details about the files, see Tuned Profile Reference below.

To activate the azg profile, follow the steps below. Complete these steps on all servers in the cluster:

1. **If you ran the installer in sudo mode, you can skip this step**. The installer copied the tuned profile to the `etc/tuned` directory but it did not automatically activate the profile. If you ran the installer as a non-root user, copy the `azg` directory from `<install_path>/examples/tuned-profile` to the `/etc/tuned` directory. For example, the following command copies azg from the default installation path to `/etc/tuned`:

   ```
   sudo cp -r /opt/cambridgesemantics/examples/azg /etc/tuned
   ```

2. Next, tuned is installed by default with RHEL 9. If you are using Rocky9, you may need to install it. You can run the following command to install the program:

   ```
   sudo dnf install -y tuned
   ```

3. Run the following command to activate the azg profile:

   ```
   sudo tuned-adm profile azg
   ```

The host servers are now configured to use the tuned profile that is optimal for AnzoGraph.

> **Tip**
> To disable tuned profiles, you can run `sudo tuned-adm off`. After running the command, no tuned profiles will be active.

Tuned Profile Reference

This section describes the tuned profile files and the kernel configuration changes that they apply.

tuned.conf

The table below describes the Linux kernel configuration settings that are modified by `tuned.conf`.

Setting	Description	AZG Profile Change
vm.dirty_ratio	Specifies the percentage of system memory that can be occupied by "dirty" data before flushing the cache to disk. Dirty data are pages in memory that have been updated and do not match what is stored on disk.	Reduces `vm.dirty_ratio` to `2%` to increase the frequency with which the system cache is flushed.
vm.swappiness	Controls the tendency of the kernel to move processes out of physical memory and onto the swap disk. A value of `0` means the kernel avoids swapping processes out of physical memory for as long as possible. A value of `100` tells the kernel to aggressively swap processes out of physical memory to the swap disk.	Sets `vm.swappiness` to `30`.
vm.max_map_count	Sets the limit on the maximum number of memory map areas a process can use. Since AnzoGraph is memory intensive, it may reach the default maximum map count of `65535` and be shut down by the operating system.	Increases `vm.max_map_count` to `2097152`.
net.ipv4.tcp_rmem	Controls the size of the receive buffer for TCP connections. It sets the minimum, default, and maximum sizes of the buffer in bytes.	Sets `tcp_rmem` to `"4096 87380 16777216"`.
net.ipv4.tcp_	Controls the size of the send buffer for TCP	Sets `tcp_wmem` to

Setting	Description	AZG Profile Change
wmem	connections. It sets the minimum, default, and maximum sizes of the buffer in bytes.	`"4096 16384 16777216"`.
net.ipv4.udp_mem	Controls the amount of memory that can be allocated for the kernel's UDP buffer. It sets the minimum, default, and maximum sizes of the buffer in bytes.	Sets `udp_mem` to `"3145728 4194304 16777216"`.
transparent_ hugepages	Controls whether Transparent Huge Pages (THP) is enabled or disabled system-wide. When THP is enabled system-wide, it can dramatically degrade AnzoGraph performance.	Disables THP by setting `transparent_ hugepages` to `never`.

additional-tunables.sh

The `additional-tuneables.sh` script is called by `tuned.conf` and configures the following Linux kernel configuration settings so that they are optimal for AnzoGraph.

Setting	Description	AZG Profile Change
overcommit_ memory	Controls whether obvious overcommits of the address space are allowed.	Sets `overcommit_memory` to `0` to ensure that very large overcommits are not allowed but some overcommits can be used to reduce swap usage.
overcommit_ratio	Controls the percentage of memory that is allowed to be used for overcommits.	Sets `overcommit_ratio` to `50%`.

Setting	Description	AZG Profile Change
transparent_ hugepage/defrag	Though the tuned profile disables Transparent Huge Pages (THP) system-wide, this setting controls whether huge pages can still be enabled on a per process basis (inside MADV_HUGEPAGE madvise regions).	Sets `transparent_ hugepage/defrag` to `madvise` so that the kernel only assigns huge pages to individual process memory regions that are specified with the `madvise()` system call.
tcp_timestamps	Controls whether TCP timestamps are enabled or disabled.	Sets `tcp_timestamps` to `0`, which disables TCP timestamps in order to reduce performance spikes related to timestamp generation.

Non-Root Installs: Configuring and Starting the AnzoGraph Services

> **Note**
> **If the installer was run in sudo mode**, the installer automatically created AnzoGraph systemd services in the `/etc/systemd/system` directory. If AnzoGraph is already running, see Connecting to AnzoGraph in the Administration Guide for next steps.

If the installer was run as the Anzo service user and not with sudo privileges, the last step in the post-installation configuration is to implement the AnzoGraph systemd services and start the database. It is important to set up the services to run as the Anzo service user so that AnzoGraph can access the data on the shared file system. In addition, the services are configured to tune user resource limits (ulimits) as well as set `$JAVA_HOME` so that AnzoGraph can find the OpenJDK installation.

The service files are included in the `<install_path>/examples/systemd-services` directory. Follow the instructions below to configure and start the services.

1. Configure the System Management Service
2. Configure the Database Service on the Leader Server (and Single-Servers)

Configure the System Management Service

The system management daemon, **azgmgrd**, is a very lightweight program that runs on all AnzoGraph servers and manages communication between the system manager and the database as well as between the nodes in a cluster. Follow the steps below to configure and start the service that runs the azgmgrd process.

1. Open the **azgmgrd.service** file in the `<install_path>/examples/systemd-services` directory. The contents of the file are shown below.

   > **Note**
   > The following contents are from an installation that used the default installation path, `/opt/cambridgesematics`. The contents of your file may differ. Also, note the **User=anzograph** value shown in bold below. The value needs to be edited to replace **anzograph** with the Anzo service user name.

```
[Unit]
Description=AnzoGraph communication service
# depends on NetworkManager-wait-online.service enabled
Wants=network-online.target
After=network-online.target

[Service]
Type=forking
# The PID file is optional, but recommended in the manpage
# "so that systemd can identify the main process of the daemon"
#PIDFile=/var/run/azgmgrd.pid
WorkingDirectory=/opt/cambridgesemantics/anzograph
StandardOutput=syslog
StandardError=syslog
LimitCPU=infinity
LimitNOFILE=65536
LimitAS=infinity
LimitNPROC=infinity
```

```
LimitMEMLOCK=infinity
LimitLOCKS=infinity
LimitFSIZE=infinity
User=anzograph
UMask=007

Environment=PATH=$PATH:/opt/cambridgesemantics/anzograph/bin:/opt/cambridgesema
ntics/anzograph/tools/bin
Environment=JAVA_HOME=/usr/lib/jvm/java-21-openjdk-21.0.1.0.12-3.el9.x86_64
Environment=UDX_LOGS=/opt/cambridgesemantics/anzograph/internal/logs
Environment=HYPER_
PATH=/opt/cambridgesemantics/anzograph/vendor/com.tableau/hyper/lib/hyper
ExecStart=/opt/cambridgesemantics/anzograph/bin/azgmgrd
/opt/cambridgesemantics/anzograph

CPUAccounting=false
MemoryAccounting=false
[Install]
WantedBy=multi-user.target
Alias=sbxmgrd.service
```

2. In the following line of the file, replace **anzograph** with the name of the Anzo service user.

   ```
   User=anzograph
   ```

   For example, if the name of the service user is **anzo**, the line is changed to the following value:

   ```
   User=anzo
   ```

3. Save and close the file.
4. Copy **azgmgrd.service** from the `<install_path>/examples/systemd-services` directory to the `/usr/lib/systemd/system` directory. For example, the following command copies azgmgrd.service from the default installation path to `/usr/lib/systemd/system`:

   ```
   sudo cp /opt/cambridgesemantics/examples/systemd-services/azgmgrd.service
   /usr/lib/systemd/system
   ```

5. Run the following commands to start and enable the service:

```
sudo systemctl start azgmgrd.service
```

```
sudo systemctl enable azgmgrd.service
```

6. Repeat this process on all servers in the cluster.

> **Tip**
> The azgmgrd deamon must be running to start the database, but it typically does not need to be restarted unless you are upgrading AnzoGraph or the host servers are rebooted. It does not need to be stopped and started each time the database is restarted.

Configure the Database Service on the Leader Server (and Single-Servers)

The **anzograph** service runs the database process. This service is configured to run after azgmgrd is started. Starting the database is done only on the leader server. The leader connects to the system managers on the compute servers and starts the database across the cluster.

1. Open the **anzograph.service** file in the `<install_path>/examples/systemd-services` directory. The contents of the file are shown below.

   > **Note**
   > The following contents are from an installation that used the default installation path, `/opt/cambridgesematics`. The contents of your file may differ. Also, note the **User=anzograph** value shown in bold below. The value needs to be edited to replace **anzograph** with the Anzo service user name.

```
[Unit]
Description=AnzoGraph database service
After=azgmgrd.service
Wants=azgmgrd.service

[Service]
Type=oneshot
# The PID file is optional, but recommended in the manpage
```

```
# "so that systemd can identify the main process of the daemon"
#PIDFile=/var/run/azg.pid
WorkingDirectory=/opt/cambridgesemantics/anzograph
StandardOutput=syslog
StandardError=syslog
User=anzograph
UMask=027
RemainAfterExit=yes

Environment=PATH=$PATH:/opt/cambridgesemantics/anzograph/bin:/opt/cambridgesema
ntics/anzograph/tools/bin
ExecStart=/opt/cambridgesemantics/anzograph/bin/azgctl -start
ExecStop=/opt/cambridgesemantics/anzograph/bin/azgctl -stop

[Install]
WantedBy=multi-user.target
Alias=gqe.service
```

2. In the following line of the file, replace **anzograph** with the name of the Anzo service user.

   ```
   User=anzograph
   ```

   For example, if the name of the service user is **anzo**, the line is changed to the following value:

   ```
   User=anzo
   ```

3. Save and close the file.
4. Copy **anzograph.service** from the `<install_path>/examples/systemd-services` directory to the `/usr/lib/systemd/system` directory. For example, the following command copies anzograph.service from the default installation path to `/usr/lib/systemd/system`:

   ```
   sudo cp /opt/cambridgesemantics/examples/systemd-services/anzograph.service
   /usr/lib/systemd/system
   ```

5. Run the following commands to start and enable the new service:

   ```
   sudo systemctl start anzograph.service
   ```

   ```
   sudo systemctl enable anzograph.service
   ```

Once the services are in place and enabled, AnzoGraph should be running. To stop and start the database from the command line, run the following systemctl commands on the leader node (You do not need to stop and start azgmgrd.):

```
sudo systemctl stop anzograph
```

```
sudo systemctl start anzograph
```

For instructions on configuring the connection to AnzoGraph in Anzo, see Connecting to AnzoGraph in the Administration Guide.

> **Tip**
> See Securing an AnzoGraph Environment for recommendations to follow for securing AnzoGraph environments.

Securing an AnzoGraph Environment

This topic lists the recommended procedures to follow to strengthen the security of AnzoGraph 3.1 environments.

- Set Up Firewall Rules
- Replace the Default Self-Signed Certificates with Trusted Certificates
- Configure File Access Policies

Set Up Firewall Rules

In order to protect the environment from malicious systems and prevent man-in-the-middle attacks or leaking of data source credentials, firewall rules should be configured for the AnzoGraph cluster network. Rules should allow outbound connections only to trusted data sources and services. For information about the ports that need to be opened for inbound and outbound connections to support normal operations, see Firewall Requirements.

Replace the Default Self-Signed Certificates with Trusted Certificates

AnzoGraph installations include self-signed certificates, `serv.crt` and `ca.crt`, and private and public keys, `serv.key` `serv.pub.key`, in the `<install_path>/config` directory. The certificates and keys are required for encrypted communication over gRPC protocol. You can follow the steps below to replace the default certificates and keys with your own trusted files.

Important

Your certificates must meet the following requirements:

- All servers in the cluster must use the same certificates and keys.
- The DNS in the certificates must be `localhost`.
- Your certificates and keys must use the same file names as the default files that you are replacing.
- The public key should be generated from the new private key.

1. On the leader server, run the following commands to stop the database and the system manager, azgmgrd:

   ```
   sudo systemctl stop anzograph
   ```

   ```
   sudo systemctl stop azgmgrd
   ```

2. On the leader server, open the `<install_path>/config/settings.conf` file for editing.
3. Uncomment the `use_custom_ssl_files=false` line and change the value to **true**.
4. Save and close settings.conf.
5. On each server in the cluster, replace the `serv.crt`, `ca.crt`, `serv.key`, and `serv.pub.key` files in the `<install_path>/config` directory with your files. Make sure that the new files have the same file names as the default files.

> **Important**
> Anzo also needs to trust the new certificates. Make sure you have **Trust All TLS Certificates** enabled on the AnzoGraph connection or make sure Anzo's trust store has either the certificate for the CA that signed the certificate or the certificate itself. See Adding a Certificate to the Anzo Trust Store in the Administration Guide for instructions.

6. Restart AnzoGraph with the following commands. Run the first command on all servers in the cluster. Then run the second command on the leader server.

```
sudo systemctl start azgmgrd
```

```
sudo systemctl start anzograph
```

Configure File Access Policies

AnzoGraph offers configuration options for ensuring that only certain files or directories on the server are accessible during the execution of a query. These configuration settings specify patterns that are used to determine whether a directory or file is accessible. When AnzoGraph receives a request that includes a path to a file or directory, it checks that path against the allowed and denied access patterns. If the specified file or directory matches one of the allowed access patterns and it is not matched to a deny pattern, the query is executed. If the specified path is matched to a denied pattern or is not matched to any of the allowed patterns, the query is aborted and AnzoGraph returns an access denied error message. For details and configuration instructions, see Managing AnzoGraph File Access Policies in the Administration Guide.

Upgrading AnzoGraph

A key area of growth in AnzoGraph is the development and support of custom, user-managed extensions, such as the Graph Data Interface for virtualization and Elasticsearch support. Most AnzoGraph releases include revisions to the API and prepackaged extensions. Because of the frequency of AnzoGraph updates and because the extensions directory (`<install_path>/lib/udx`) is user-managed rather than AnzoGraph- or installer-controlled, you must uninstall the existing version of AnzoGraph and then install the new version. **In-place upgrades are not supported.**

Since AnzoGraph is stateless when used with Anzo and Anzo manages all of your data, removing the existing installation does not impact Anzo or your graphmarts. Follow the instructions in Uninstalling AnzoGraph to back up any custom files and remove the installation directory before deploying a new version.

Uninstalling AnzoGraph

This topic provides instructions for uninstalling AnzoGraph. On clusters, complete steps 2 through 4 below on each server in the cluster.

1. Make sure the database and system management daemon processes are stopped. Run the following commands to stop the services. On a cluster, run these commands on the leader server:

   ```
   sudo systemctl stop anzograph
   ```

   ```
   sudo systemctl stop azgmgrd
   ```

2. Next, if you have custom files, such as certificates in `<install_path>/config` or JDBC drivers in the `<install_path>/lib/udx` directory, make a backup copy of those files on the leader node. Make sure that you choose a backup location that is outside of the AnzoGraph installation path. If you install a new version of AnzoGraph, you can place the custom files back into the appropriate directories.

   > **Note**
   > If you have modified the settings file, `<install_path>/config/settings.conf`, Cambridge Semantics recommends that you make a backup copy of the file on the leader server so that you can refer to it when configuring the new deployment. As a best practice, however, do not overwrite `settings.conf` in the new version of AnzoGraph with the backup copy from the previous version. Instead, Cambridge Semantics recommends that you apply all changes to the new file. Since new releases may add or remove settings or change the default value of certain settings, it is important to use the version of the file that was installed with the release.

3. Run the following command to begin the uninstall process:

   ```
   sudo ./<install_path>/uninstall
   ```

   The script asks if you want to proceed:

```
Do you want to proceed with AnzoGraph DB installation?
OK [o, Enter], Cancel [c]
```

4. Press **Enter** to confirm that you want to uninstall AnzoGraph.

   The wizard asks if you want to clear the installation directory and user and configuration files:

```
Are you sure you want to completely remove AnzoGraph DB and all of its
components?
Yes [y, Enter], No [n]
```

5. Cambridge Semantics recommends that you remove all installation and configuration files. Press **Enter** to remove the entire installation directory as well as all configuration and user files.

The wizard uninstalls AnzoGraph. If you plan to install a new version, see Install AnzoGraph for installation instructions.

Deploying a Static Distributed Unstructured Cluster

This section provides instructions for deploying a static Distributed Unstructured (DU) cluster. For information about DU requirements, see Distributed Unstructured and Elasticsearch Requirements.

> **Tip**
> For instructions on setting up the Kubernetes infrastructure so that DU clusters can be launched on-demand, see Setting up K8s Infrastructure for Dynamic Deployments.

In this section:

Installing the Distributed Unstructured Cluster

This topic provides instructions for deploying a Distributed Unstructured (DU) cluster. See DU Cluster Requirements for details about server requirements.

Important
Since the DU cluster will access the shared file store, it is important to install and run the software with the same service account that runs Anzo. For more information, see Service User Account Requirements.

1. Deploy the Leader Node
2. Deploy the Worker Nodes
3. Configure and Start the DU Services

Deploy the Leader Node

Follow the instructions below to deploy the DU leader node. The leader software is typically installed on the Anzo host server since it is a lightweight program that distributes requests to the worker instances for processing.

1. If you are installing the leader software on a dedicated server, make sure that server has access to the shared file system.
2. Copy the Anzo DU installation script to the leader server and then run the following command to make the script executable:

   ```
   chmod +x <script_name>
   ```

3. If necessary, run the following command to become the Anzo service user:

   ```
   su <name>
   ```

   Where <name> is the name of the service user. For example:

   ```
   su anzo
   ```

4. Run the following command to start the installation wizard:

```
./<script_name>
```

   The script unpacks the JRE and then waits for input before starting the installation.

5. Press **Enter** to start the installation. The software license agreement is presented.

6. Review the software license agreement. Press **Enter** to scroll through the terms. At the end of the agreement, type **1** to accept the terms or type **2** to disagree and stop the installation.

   When the agreement is accepted, the installer prompts you to specify the components to install:

```
Which components should be installed?
1: Leader [*1]
2: Worker [*2]
(To show the description of a component, please enter one of *1, *2)
Please enter a comma-separated list of the selected values or [Enter] for the
default selection:
[1,2]
```

7. At the components prompt, type **1** (Leader) and then press **Enter**.

   The installer prompts you to specify the installation path:

```
Where should the Anzo Unstructured be installed?
[/opt/AnzoDU]
```

8. Specify the path and directory to install Anzo DU. Press **Enter** to accept the default installation path or type an alternate path and then press **Enter**.

   Next, the installer prompts for the hostname of this leader instance. It defaults to the IP address of the server:

```
Set the hostname for this node.
Enter the HostName/Address for this node.
Hostname/Address
[10.100.0.11]
```

9. Press **Enter** to accept the default value. If necessary, type a different IP address, and then press **Enter**.

The installer then prompts for any additional leader node hostnames. Typically there is one leader node and this value is specified as the same IP address as the previous step.

```
Configure leader hostnames
Please enter the hostnames or addresses for the leader nodes. Each entry
comma separated.
[10.100.0.11]
```

10. If you set up additional leader nodes for redundancy, enter a comma separated list of the IP addresses for the alternate nodes. Otherwise, accept the default value and press **Enter**.

    Next, the installer prompts you to specify the maximum amount of memory that this leader instance can use. The installer lists the total RAM available and chooses 1/2 of the total memory as the default value.

    ```
    Choose the maximum memory that the node can use
    Please enter the maximum amount of RAM memory that the node may use.
    The minimum amount currently supported is 1024 MB. 29995 MB is available.
    Maximum Memory in MB
    [14998]
    ```

11. Specify the maximum amount of memory (in MB) that this leader instance can use. Press **Enter** to accept the default value or specify an alternate value and then press **Enter**.

12. The installation of the Anzo DU leader software begins and is configured according to the values that you specified. Proceed to Deploy the Worker Nodes to install the Worker instances.

Deploy the Worker Nodes

Follow the instructions below to deploy the Anzo Distributed Unstructured (DU) worker nodes.

1. Make sure that the worker host servers have access to the Anzo shared file system and meet the requirements in DU Cluster Requirements.

2. Copy the Anzo DU installation script to each of the worker servers and then run the following command to make the script executable:

    ```
    chmod +x <script_name>
    ```

3. If necessary, run the following command to become the Anzo service user:

```
su <name>
```

   Where <name> is the name of the service user. For example:

```
su anzo
```

4. Run the following command to start the installation wizard:

```
./<script_name>
```

   The script unpacks the JRE and then waits for input before starting the installation.

5. Press **Enter** to start the installation. The software license agreement is presented.

6. Review the software license agreement. Press **Enter** to scroll through the terms. At the end of the agreement, type **1** to accept the terms or type **2** to disagree and stop the installation.

   When the agreement is accepted, the installer prompts you to specify the components to install:

```
Which components should be installed?
1: Leader [*1]
2: Worker [*2]
(To show the description of a component, please enter one of *1, *2)
Please enter a comma-separated list of the selected values or [Enter] for the
default selection:
[1,2]
```

7. At the components prompt, type **2** (Worker) and then press **Enter**.

   The installer prompts you to specify the installation path:

```
Where should the Anzo Unstructured be installed?
[/opt/AnzoDU]
```

8. Specify the path and directory to install Anzo DU. Press **Enter** to accept the default installation path or type an alternate path and then press **Enter**.

   Next, the installer prompts for the hostname of this worker instance. It defaults to the IP address of the server:

```
Set the hostname for this node.
Enter the HostName/Address for this node.
Hostname/Address
[10.100.0.12]
```

9. Press **Enter** to accept the default value. If necessary, type a different IP address, and then press **Enter**.

   The installer then prompts you to specify the maximum number of service instances for this worker node. Each service instance processes one unstructured document at a time. The default value is 4 instances.

```
Choose the maximum number of service instances and worker port
Please enter the maximum number of service instances to use.
The minimum amount currently supported is 1.
Maximum Service Instances
[4]
```

10. Press **Enter** to accept the default number of maximum service instances or specify another value and then press **Enter**.

    The installer now prompts you to specify the port to use for this worker. The default port is **2552**.

```
Worker Port
[2552]
```

11. Specify the port to use for this worker. Press **Enter** to accept the default value or type a different port and then press **Enter**.

    Next, the installer prompts you to specify the hostname(s) of the leader node(s).

```
Configure leader hostnames
Please enter the hostnames or addresses for the leader nodes. Each entry
comma separated.
[]
```

12. Specify the IP address for the leader instance that you deployed in Deploy the Leader Node above. If you deployed multiple leader nodes, specify each leader's IP address in a comma separated list.

The installer now prompts you to specify the maximum amount of memory that this worker instance can use. The installer lists the total RAM available and chooses 1/2 of the total memory as the default value.

```
Choose the maximum memory that the node can use
Please enter the maximum amount of RAM memory that the node may use.
The minimum amount currently supported is 1024 MB. 29995 MB is available.
Maximum Memory in MB
[14998]
```

13. Specify the maximum amount of memory (in MB) that this worker instance can use. Press **Enter** to accept the default value or specify an alternate value and then press **Enter**.

    The installation of the Anzo DU worker software begins and is configured according to the values that you specified.

14. Repeat the steps above for each worker instance in the cluster.

Once the leader and all of the worker nodes are installed, proceed to Configure and Start the DU Services.

Note

If you upgraded the DU software, make sure that you restart the leader and worker applications.

In addition, restart the **Anzo Server Akka Cluster Integration** and **Anzo Unstructured Distributed** services. To restart these services:

1. In the Administration application, expand the **Servers** menu and click **Advanced Configuration**.
2. On the Advanced Configuration screen, click the **I understand and accept the risk** button to view the Anzo bundles.
3. In the **Search** field at the top of the screen, start typing the name of the service that you want to restart. When the service appears in the list onscreen, click the service name to view the details.

4. At the top of the screen, click **Stop Bundle**. Then click **Start Bundle** when the start option becomes available.

Configure and Start the DU Services

Once the cluster is installed, Cambridge Semantics recommends that you set up leader and worker services to ensure that applications run as the Anzo service user and can access the data that other platform components write to the shared file system. Follow the instructions below to configure the services.

Note
Root user privileges are required to complete these tasks.

1. Configure and Start the Leader Service
2. Configure and Start the Worker Service

Configure and Start the Leader Service

Follow the instructions below to create and start the leader service.

1. On the leader server, create a file called **anzo-du-leader.service** in the `/usr/lib/systemd/system` directory. For example:

```
# vi /usr/lib/systemd/system/anzo-du-leader.service
```

2. Add the following contents to anzo-du-leader.service. Placeholder values are shown in **bold**:

```
[Unit]
Description=Service for Distributed Unstructured Leader
After=syslog.target network.target local-fs.target remote-fs.target nss-
lookup.target

[Service]
Type=forking
RemainAfterExit=yes
LimitCPU=infinity
LimitNOFILE=65536
```

```
LimitAS=infinity
LimitNPROC=65536
LimitMEMLOCK=infinity
LimitLOCKS=infinity
LimitFSIZE=infinity
ExecStart=/install_path/leader start
ExecStop=/install_path/leader stop
User=service_user_name
Group=service_user_name

[Install]
WantedBy=default.target
```

Where **install_path** is the Anzo DU installation path and directory and **service_user_name** is the name of the Anzo service user. For example:

```
[Unit]
Description=Service for Distributed Unstructured Leader
After=syslog.target network.target local-fs.target remote-fs.target nss-
lookup.target

[Service]
Type=forking
RemainAfterExit=yes
LimitCPU=infinity
LimitNOFILE=65536
LimitAS=infinity
LimitNPROC=65536
LimitMEMLOCK=infinity
LimitLOCKS=infinity
LimitFSIZE=infinity
ExecStart=/opt/AnzoDU/leader start
ExecStop=/opt/AnzoDU/leader stop
User=anzo
Group=anzo

[Install]
WantedBy=default.target
```

3. Save and close the file, and then run the following commands to start and enable the new service:

```
# systemctl start anzo-du-leader.service
```

```
# systemctl enable anzo-du-leader.service
```

Once the service is enabled, the leader should be running. Any time you start and stop the leader, run the following systemctl commands: `sudo systemctl stop anzo-du-leader` and `sudo systemctl start anzo-du-leader`.

Configure and Start the Worker Service

Follow the instructions below to create and start the worker service. Complete the steps below on each worker node in the cluster.

1. Create a file called **anzo-du-worker.service** in the `/usr/lib/systemd/system` directory. For example:

```
# vi /usr/lib/systemd/system/anzo-du-worker.service
```

2. Add the following contents to anzo-du-worker.service. Placeholder values are shown in **bold**:

```
[Unit]
Description=Service for Distributed Unstructured Worker
After=syslog.target network.target local-fs.target remote-fs.target nss-
lookup.target

[Service]
Type=forking
RemainAfterExit=yes
LimitCPU=infinity
LimitNOFILE=65536
LimitAS=infinity
LimitNPROC=65536
LimitMEMLOCK=infinity
LimitLOCKS=infinity
LimitFSIZE=infinity
ExecStart=/install_path/worker start
ExecStop=/install_path/worker stop
User=service_user_name
Group=service_user_name

[Install]
WantedBy=default.target
```

Where **install_path** is the Anzo DU installation path and directory and **service_user_name** is the name of the Anzo service user. For example:

```
[Unit]
Description=Service for Distributed Unstructured Worker
After=syslog.target network.target local-fs.target remote-fs.target nss-
lookup.target

[Service]
Type=forking
RemainAfterExit=yes
LimitCPU=infinity
LimitNOFILE=65536
LimitAS=infinity
LimitNPROC=65536
LimitMEMLOCK=infinity
LimitLOCKS=infinity
LimitFSIZE=infinity
ExecStart=/opt/AnzoDU/worker start
ExecStop=/opt/AnzoDU/worker stop
User=anzo
Group=anzo

[Install]
WantedBy=default.target
```

3. Save and close the file, and then run the following commands to start and enable the new service:

```
# systemctl start anzo-du-worker.service
```

```
# systemctl enable anzo-du-worker.service
```

4. Repeat the steps above for each worker server.

Once the service is enabled, the worker should be running. Any time you start and stop a worker, run the following systemctl commands: `sudo systemctl stop anzo-du-worker` and `sudo systemctl start anzo-du-worker`.

Important

Any time the AU leader instance is restarted, the **Anzo Server Akka Cluster Integration** and **Anzo Unstructured Distributed** services must be restarted in Anzo. To restart a service:

1. In the Administration application, expand the **Servers** menu and click **Advanced Configuration**.
2. On the Advanced Configuration screen, click the **I understand and accept the risk** button to view the Anzo bundles.
3. In the **Search** field at the top of the screen, start typing the name of the service that you want to restart. When the service appears in the list onscreen, click the service name to view the details.
4. At the top of the screen, click **Stop Bundle**. Then click **Start Bundle** when the start option becomes available.

For next steps in setting up the unstructured environment, see Installing Elasticsearch. For instructions on connecting the DU cluster to Anzo, see Connect to a Distributed Unstructured Cluster in the Administration Guide.

Installing Elasticsearch

This topic provides instructions for deploying Elasticsearch for use in the Anzo platform.

Important

Elasticsearch cannot be run as the root user and must have read and write access to the Anzo file store. Therefore, it is important to install and run Elasticsearch as the Anzo service user, otherwise unstructured pipelines will fail due to permissions errors. For more information, see Service User Account Requirements.

1. Make sure that the Elasticsearch host server has access to the Anzo shared file system and meets the requirements in Elasticsearch Requirements.
2. Download a supported Elasticsearch version from the Elasticsearch Past Releases website. Docker images are also available from the Docker @ Elastic website.

   Note

   Anzo supports Elasticsearch Versions 7.10.2 – 7.17.3.

3. Follow the appropriate version of the Elasticsearch Guide to install and configure the software.
4. As part of the Elasticsearch configuration, Elastic recommends that you modify the following Linux kernel configuration settings:
   - **vm.swappiness**: Controls the tendency of the kernel to move processes out of physical memory and onto the swap disk. Elastic recommends that you set this value to **1**.
   - **vm.max_map_count**: Sets the limit on the maximum number of memory map areas a process can use. Elastic recommends that you set this value to **262144**.

   You have two options for configuring the values:

   1. You can update the `/etc/sysctl.conf` file to include the following contents:

```
# For more information, see sysctl.conf(5) and sysctl.d(5).
vm.swappiness = 1
vm.max_map_count = 262144
```

Important

With this method, you must reboot the system to apply the configuration changes after `sysctl.conf` is updated.

2. You can run the following sysctl commands to configure the settings:

```
# sysctl -e vm.swappiness=1
```

```
# sysctl -e vm.max_map_count=262144
```

5. Next, configure Elasticsearch to save snapshots to the Anzo shared file system.
   - For a mounted file system, such as NFS, uncomment the Path setting, **path.repo** (or **path.data** in some versions), in `<elasticsearch_install_path>/config/elasticsearch.yml` and specify the path and directory for the mounted file system:

     ```
     path.repo: /<path>/<directory>
     ```

     For example:

     ```
     path.repo: /opt/anzoshare
     ```

   - For S3, see S3 Repository Plugin in the Elasticsearch documentation for information about installing the S3 repository plugin. Then see Client Settings for instructions on configuring the S3 client.
   - For HDFS, see Hadoop HDFS Repository Plugin in the Elasticsearch documentation for information about installing the HDFS repository plugin. Then see Hadoop Security for information about configuring Kerberos authentication.
6. Configure the amount of memory that Elasticsearch can use. By default, Elasticsearch is configured to use a maximum heap size of 1 GB. Cambridge Semantics recommends that you increase the amount to 50% of the memory that is available on the server. To change the

configuration, open the `<elasticsearch_install_path>/config/jvm.options` file in an editor. At the top of the file, modify the **Xms** and **Xmx** values to replace the **1** with the new value. For example:

```
# Xms represents the initial size of total heap space
# Xmx represents the maximum size of total heap space

-Xms15g
-Xmx15g
```

7. If you want to secure the Elasticsearch instance, follow the instructions in Configuring security in Elasticsearch in the Elasticsearch documentation.

   Important
   If you set up SSL authentication with a trusted certificate, make sure that you add the certificate to the Anzo trust store. For instructions, see See Adding a Certificate to the Anzo Trust Store in the Administration Guide.

8. When the configuration is complete, see Configuring the Elasticsearch Service below for instructions on configuring Elasticsearch to start automatically as the Anzo user.

Configuring the Elasticsearch Service

Cambridge Semantics recommends that you configure an Elasticsearch service for starting Elasticsearch automatically as the Anzo service user. Follow the instructions below to implement the service.

Note
Root user privileges are required to complete this task.

1. Create a file called `elasticsearch.service` in the `/usr/lib/systemd/system` directory. For example:

```
# vi /usr/lib/systemd/system/elasticsearch.service
```

2. Add the following contents to `elasticsearch.service`. The text below includes placeholder **<elasticsearch_install_path>**, **<anzo_service_user>**, and **<anzo_service_group>** values. Replace the placeholders with the appropriate values for your Elasticsearch installation location as well as the user and group name for your Anzo service user account.

```
[Unit]
Description=Elasticsearch
Documentation=https://www.elastic.co
Wants=network-online.target
After=network-online.target

[Service]
Type=forking
RuntimeDirectory=elasticsearch
# Use the following setting to specify an alternate Java JVM if not using the
# embedded JVM in elasticsearch/jdk.
# Environment=ES_JAVA_HOME=<java_install_path>
Environment=ES_HOME=<elasticsearch_install_path>
Environment=ES_PATH_CONF=<elasticsearch_install_path>/config

User=<anzo_service_user>
Group=<anzo_service_group>

ExecStart=<elasticsearch_install_path>/bin/elasticsearch --daemonize

# Specifies the maximum file descriptor number that can be opened by this
process
LimitNOFILE=65535

# Specifies the maximum number of processes
LimitNPROC=4096

# Specifies the maximum size of virtual memory
LimitAS=infinity

# Specifies the maximum file size
LimitFSIZE=infinity

# Max Locked Memory
LimitMEMLOCK=infinity
```

```
# Disable timeout logic and wait until process is stopped
TimeoutStopSec=0

# SIGTERM signal is used to stop the Java process
KillSignal=SIGTERM

# Send the signal only to the JVM rather than its control group
KillMode=process

# Java process is never killed
SendSIGKILL=no

# When a JVM receives a SIGTERM signal it exits with code 143
SuccessExitStatus=143

# Allow a slow startup before the systemd notifier module kicks in to extend
the timeout
TimeoutStartSec=75

[Install]
WantedBy=multi-user.target
```

The following example shows a completed elasticsearch.service file:

```
[Unit]
Description=Elasticsearch
Documentation=https://www.elastic.co
Wants=network-online.target
After=network-online.target

[Service]
Type=forking
RuntimeDirectory=elasticsearch
# Use the following setting to specify an alternate Java JVM if not using the
# embedded JVM in elasticsearch/jdk.
# Environment=ES_JAVA_HOME=<java_install_path>
Environment=ES_HOME=/opt/elasticsearch
Environment=ES_PATH_CONF=/opt/elasticsearch/config

User=anzo
Group=anzo
```

```
ExecStart=/opt/elasticsearch/bin/elasticsearch --daemonize

# Specifies the maximum file descriptor number that can be opened by this
process
LimitNOFILE=65535

# Specifies the maximum number of processes
LimitNPROC=4096

# Specifies the maximum size of virtual memory
LimitAS=infinity

# Specifies the maximum file size
LimitFSIZE=infinity

# Max Locked Memory
LimitMEMLOCK=infinity

# Disable timeout logic and wait until process is stopped
TimeoutStopSec=0

# SIGTERM signal is used to stop the Java process
KillSignal=SIGTERM

# Send the signal only to the JVM rather than its control group
KillMode=process

# Java process is never killed
SendSIGKILL=no

# When a JVM receives a SIGTERM signal it exits with code 143
SuccessExitStatus=143

# Allow a slow startup before the systemd notifier module kicks in to extend
the timeout
TimeoutStartSec=75

[Install]
WantedBy=multi-user.target
```

3. Save and close the file, and then run the following commands to start and enable the new service:

```
# systemctl enable elasticsearch.service
```

```
# systemctl status elasticsearch.service
```

```
# systemctl start elasticsearch.service
```

Once the service is in place, Elasticsearch should be stopped and started via systemctl. For example, `systemctl stop elasticsearch` and `systemctl start elasticsearch`.

Once this Elasticsearch instance is configured and running, follow the instructions in Connecting to Elasticsearch in the Administration Guide to connect Anzo to this instance.

Upgrading the Distributed Unstructured Software

The steps to upgrade the Anzo Distributed Unstructured (DU) software are the same as the installation instructions in Installing the Distributed Unstructured Cluster. When you update the existing installation, each prompt defaults to the value that is specified for the current deployment. You can press **Enter** through the prompts to retain the existing settings. The last step in the process, however, asks if you want to overwrite files in the `<AnzoDU_install_path>/etc` directory that have been modified. Cambridge Semantics recommends that you choose **ya (Yes To All)** to overwrite all files in that directory so that important options from the version you are upgrading to are deployed to your environment. If you have customized files in the etc directory, create a backup copy of the directory before starting the upgrade so that you can refer to the backup files when customizing the new version.

Important

When upgrading the DU software, the leader and worker applications must be upgraded at the same time using the same installer so that the software versions are identical across the cluster. You cannot upgrade the worker nodes without upgrading the leader and vice versa.

After the upgrade, make sure that you restart the leader and worker applications as well as the following Anzo services:

- Anzo Server Akka Cluster Integration
- Anzo Unstructured Distributed

Setting up K8s Infrastructure for Dynamic Deployments

To get started on setting up the Kubernetes (K8s) infrastructure, see the deployment instructions for your cloud service provider. For information about K8s requirements, see Kubernetes Requirements.

In this section:

Amazon EKS Deployments

The topics in this section guide you through the process of deploying all of the Amazon Elastic Kubernetes Service (EKS) infrastructure that is required to support dynamic deployments of Anzo components. The topics provide instructions for setting up a workstation to use for deploying the K8s infrastructure, performing the prerequisite tasks before deploying the EKS cluster, creating the EKS cluster, and creating the required node groups.

In this section:

Setting Up a Workstation

This topic provides the requirements and instructions to follow for configuring a workstation to use for creating and managing the EKS infrastructure. The workstation needs to be able to connect to the AWS API. It also needs to have the required AWS and Kubernetes (K8s) software packages as well as the deployment scripts and configuration files supplied by Cambridge Semantics. This workstation will be used to connect to the AWS API and provision the K8s cluster and node groups.

> **Note**
> You can use the Anzo server as the workstation if the network routing and security policies permit the Anzo server to access the AWS and K8s APIs. When deciding whether to use the Anzo server as the K8s workstation, consider whether Anzo may be migrated to a different server or VPC in the future.

- Review the Requirements and Install the Software
- Download the Cluster Creation Scripts and Configuration Files

Review the Requirements and Install the Software

Component	Requirement
Operating System	The operating system for the workstation must be **RHEL/CentOS 7.8 or later**.
Networking	The workstation should be in the same VPC as the EKS cluster. If it is not in the same VPC, make sure that it is on a network that is routable from the cluster's VPC.
Software	• **AWS-CLI Version 2** is recommended. Version 1.16.156 or later is supported. For instructions, see Install AWS-CLI below. • **EKSCTL Version 0.40.0 or later** is required. For instructions, see Install EKSCTL below. • **Kubectl**: Cambridge Semantics recommends that you use the same kubectl version as the EKS cluster version. For instructions, see Install Kubectl below.
CSI EKSCTL Package	Cambridge Semantics provides **eksctl** scripts and configuration files to use for provisioning the EKS cluster and node groups. Download the files to the workstation. See Download the Cluster Creation Scripts and Configuration Files for more information about the eksctl package.

Install AWS-CLI

AWS CLI is the AWS command line interface. Version 2 is recommended. Follow the instructions below to install the latest aws-cli version 2 package. For more information, see Installing, Updating, and Uninstalling the AWS CLI Version 2 on Linux in the AWS CLI documentation.

1. Run the following command to download the latest aws-cli package to the current directory:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
```

2. Run the following command to unzip the package:

```
unzip awscliv2.zip
```

3. Then run the following command to run the install program. By default, the files are all installed to `/usr/local/aws-cli`, and a symbolic link is created in `/usr/local/bin`.

```
sudo ./aws/install
```

Install EKSCTL

EKSCTL is the AWS EKS command line interface. Version 0.40.0 or later is required. Follow the instructions below to download the eksctl package and place it in the `/usr/local/bin` directory. For more information, see Installing eksctl in the Amazon EKS documentation.

1. Run the following command to download the eksctl package to the `/tmp` directory:

```
curl --silent --location
"https://github.com/weaveworks/eksctl/releases/download/<tag>/eksctl_$(uname -
s)_amd64.tar.gz" | tar xz -C /tmp
```

   Where <tag> is the release that you want to download. For example:

```
curl --silent --location
"https://github.com/weaveworks/eksctl/releases/download/0.40.0/eksctl_$(uname -
s)_amd64.tar.gz" | tar xz -C /tmp
```

2. Then run the following command to move eksctl to the `/usr/local/bin` directory:

```
sudo mv /tmp/eksctl /usr/local/bin
```

Install Kubectl

Follow the instructions below to install kubectl on your workstation. Cambridge Semantics recommends that you install the same version of kubectl as the K8s cluster API. For more information, see Install and Set Up kubectl on Linux in the Kubernetes documentation.

1. Run the following cURL command to download the kubectl binary:

```
curl -LO https://dl.k8s.io/release/<version>/bin/linux/amd64/kubectl
```

Where <version> is the version of kubectl to install. For example, the following command downloads version 1.19.12:

```
curl -LO https://dl.k8s.io/release/v1.19.12/bin/linux/amd64/kubectl
```

2. Run the following command to make the binary executable:

```
chmod +x ./kubectl
```

3. Run the following command to move the binary to your PATH:

```
sudo mv ./kubectl /usr/local/bin/kubectl
```

4. To confirm that the binary is installed and that you can run kubectl commands, run the following command to display the client version:

```
kubectl version --client
```

The command returns the following type of information. For example:

```
Client Version: version.Info{Major:"1", Minor:"19", GitVersion:"v1.19.12",
GitCommit:"f3abc15296f3a3f54e4ee42e830c61047b13895f",
GitTreeState:"clean", BuildDate:"2021-06-16T13:21:12Z", GoVersion:"go1.13.15",
Compiler:"gc", Platform:"linux/amd64"}
```

Download the Cluster Creation Scripts and Configuration Files

The Cambridge Semantics GitHub repository, k8s-genesis (https://github.com/cambridgesemantics/k8s-genesis.git), includes all of the files that are needed to manage the configuration, creation, and deletion of the EKS cluster and node groups.

You can clone the repository to any location on the workstation or download the k8s-genesis package as a ZIP file, copy the file to the workstation, and extract the contents. The k8s-genesis directory includes three subdirectories (one for each supported Cloud Service Provider), the license information, and a readme file:

```
k8s-genesis
├── aws
├── azure
├── gcp
├── LICENSE
└── README.md
```

Navigate to `/aws/k8s/eksctl`. The **eksctl** directory contains all of the EKS cluster and node group configuration files. You can remove all other directories from the workstation. The eksctl files and subdirectories are shown below:

```
eksctl
├── aws_cli_common.sh
├── common.sh
├── conf.d
│   ├── iam_serviceaccounts.yaml
│   ├── k8s_cluster.conf
│   ├── nodepool_anzograph.yaml
│   ├── nodepool_common.yaml
│   ├── nodepool_dynamic.yaml
│   ├── nodepool_operator.yaml
│   └── nodepool.yaml
├── create_k8s.sh
├── create_nodepools.sh
├── delete_k8s.sh
├── delete_nodepools.sh
├── README.md
├── reference
│   ├── ca_autodiscover-patch-file.yaml
│   ├── ca_autodiscover.yaml
│   ├── cluster-autoscaler-policy.json
│   ├── nodepool_anzograph_tuner.yaml
│   ├── nodepool_dynamic_tuner.yaml
│   ├── versions
│   └── warm_ip_target.yaml
└── sample_use_cases
    ├── 1_existing_vpc_private_cluster
        │   └── k8s_cluster.conf
        ├── 2_new_vpc_public_cluster
        │   └── k8s_cluster.conf
        └── 3_nat_ha_private_cluster
                └── k8s_cluster.conf
```

The following list gives an overview of the files. Subsequent topics describe the files in more detail.

- The **aws-cli-common.sh** and **common.sh** scripts are used by the **create*.sh** and **delete*.sh** scripts during EKS cluster and node group creation and deletion.

- The **conf.d** directory contains the configuration files that supply the specifications to follow when creating the EKS cluster and node groups.
  - **iam_serviceaccounts.yaml**: Supplies optional IAM roles for Service Account specifications for use as part of cluster creation if you would like to assign permissions for the applications that run on EKS.
  - **k8s_cluster.conf**: Supplies the specifications for the EKS cluster.
  - **nodepool_anzograph.yaml**: Supplies the specifications for the AnzoGraph node group.
  - **nodepool_common.yaml**: Supplies the specifications for the Common node group.
  - **nodepool_dynamic.yaml**: Supplies the specifications for the Dynamic node group.
  - **nodepool_operator.yaml**: Supplies the specifications for the Operator node group.
  - **nodepool.yaml**: This file is supplied as a reference. It contains the superset of node group parameters and includes comments that provide additional information.
- The **create_k8s.sh** script is used to deploy the EKS cluster.
- The **create_nodepools.sh** script is used to deploy node groups in the EKS cluster.
- The **delete_k8s.sh** script is used to delete the EKS cluster.
- The **delete_nodepools.sh** script is used to remove node groups from the EKS cluster.
- The **reference** directory contains crucial files that are referenced by the cluster and node group creation scripts. The files in the directory should not be edited, and the **reference** directory must exist on the workstation at the same level as the **create*.sh** and **delete*.sh** scripts.
- The **sample_use_cases** directory contains sample EKS cluster configuration files that you can refer to or use as a template for configuring your EKS cluster depending on your use case:
  - The **k8s_cluster.conf** file in the **1_existing_vpc_private_cluster** directory is a sample file for a use case where you want to deploy the EKS cluster in an existing VPC that does not have public internet access.

- The **k8s_cluster.conf** file in the **2_new_vpc_public_cluster** directory is a sample file for a use case where you want to deploy the EKS cluster into a new VPC with public internet access that is restricted to specific IP ranges.
- The **k8s_cluster.conf** file in the **3_nat_ha_private_cluster** directory is a sample file for a use case where you want to create a private EKS cluster in an existing VPC and deploy highly available NAT gateways.

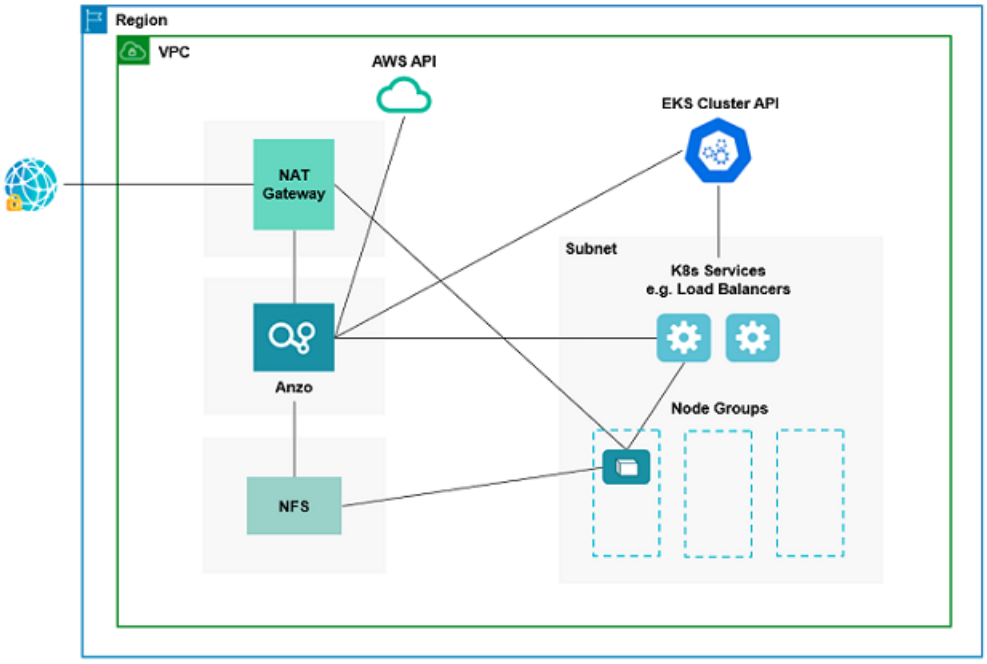
Once the workstation is configured, see Planning the Anzo and EKS Network Architecture to review information about the network architecture that the eksctl scripts create. And see Creating and Assigning IAM Policies for instructions on creating the IAM policies that are needed for assigning permissions to create and use the EKS cluster.

Planning the Anzo and EKS Network Architecture

This topic describes the network architecture that supports the Anzo and EKS integration.

> **Note**
> When you deploy the K8s infrastructure, Cambridge Semantics strongly recommends that you create the EKS cluster in the same VPC as Anzo. If you create the cluster in a new VPC, you must configure the new VPC to be routable from the Anzo VPC.

The diagram below shows the typical network components that are employed when an EKS cluster is integrated with Anzo. Most of the network resources shown in the diagram are automatically deployed (and the appropriate routing is configured) according to the values that you supply in the cluster and node group .conf files in the **eksctl** package on the workstation.



In the diagram, there are two components that you deploy before configuring and creating the K8s resources:

- **Anzo**: Since the Anzo server is typically deployed before the K8s components, you specify the Anzo VPC ID when creating the EKS cluster, ensuring that Anzo and all of the EKS cluster components are in the same network and can talk to each other. Also, make sure that Anzo has access to the AWS and EKS APIs.
- **NFS**: You are required to create a network file system (NFS). However, Anzo automatically mounts the NFS to the nodes when AnzoGraph, Anzo Unstructured, and Elasticsearch pods are deployed so that all of the applications can share files. See Shared File Storage Requirements for more information. The NFS does not need to have its own subnet but it can.

The rest of the components in the diagram are automatically provisioned, depending on your specifications, when the EKS cluster and node groups are created. The eksctl scripts can be used to create NAT gateways and subnets for outbound internet access, such as for pulling container images from the Cambridge Semantics repository. In addition, the scripts create a subnet for the K8s services and node groups and configure the routing so that Anzo can communicate with the K8s services and the services can talk to the pods that are deployed in the node groups.

Tip

When considering the network requirements of your organization and planning how to integrate the new K8s infrastructure in accordance with those requirements, it may help to consider the following types of use cases. Cambridge Semantics supplies sample cluster configuration files in the `eksctl/sample_use_cases` directory that are tailored for each of these use cases:

- **Deploy a private EKS cluster in an existing VPC (i.e., the same VPC as Anzo)**

  In this use case, the EKS cluster is deployed in a private subnet in your existing VPC. And a new (or existing, if you have one) NAT gateway is used to enable access to external services that are outside of the VPC. The control plane security group is configured to allow access only from certain CIDRs, and communication through VPN can be enabled to allow a virtual private gateway to automatically propagate routes to the route tables.

- **Deploy a public EKS cluster in a new VPC**

  In this use case, a new VPC is created with the specified CIDR. A new NAT gateway is deployed to provide outbound connectivity for the cluster nodes. Public and private subnets are also created, and public access is restricted to specific IP ranges. The new VPC will need to be configured so that it is routable from Anzo.

- **Deploy a private, highly available EKS cluster in an existing VPC**

  In this use case (like the first case listed above) a private EKS cluster is deployed in an existing VPC. In addition, NAT gateways are created in each of the cluster's Availability Zones, making the cluster highly available.

For a summary of the files in the eksctl directory, see Download the Cluster Creation Scripts and Configuration Files. Specifics about the parameters in the sample files are included in Creating the EKS Cluster.

To get started on creating the EKS infrastructure, see Creating and Assigning IAM Policies for instructions on creating the IAM policies that are needed for assigning permissions to create and use the EKS cluster.

Creating and Assigning IAM Policies

There are two custom Identity and Access Management (IAM) policies that need to be created in AWS to grant the necessary permissions to the following two types of EKS users:

1. The first type of user is the user who accesses AWS services to set up the K8s infrastructure, i.e., the user who configures, creates, and maintains the EKS cluster and node groups. This policy is called the **EKS Cluster Admin**.
2. The second type of user is the user who connects to the EKS cluster and deploys the dynamic Anzo applications. Typically this user is Anzo. Since Anzo communicates to the K8s services that provision the applications, the Anzo service account needs to be granted certain privileges. This user role is called the **EKS Cluster Developer**.

> **Note**
> The enterprise-level Anzo service account is a requirement for the Anzo installation and is typically in place before Anzo is installed. For more information, see Service User Account Requirements.

This topic provides instructions for creating the two policies and gives guidance on attaching the policies to the appropriate users or roles.

- Create and Assign the EKS Cluster Admin Policy
- Create and Assign the EKS Cluster Developer Policy

Create and Assign the EKS Cluster Admin Policy

The following IAM policy applies the minimum permissions needed for an EKS cluster administrator who will create and manage the cluster and node groups. Follow the steps below to create the policy and attach it to the appropriate principal.

1. Refer to Creating IAM Policies in the AWS documentation to create the following policy using your preferred method. You can save the contents below as a JSON file on your workstation and use the AWS CLI to create the policy, or you can paste the contents on the JSON tab if

you use the IAM console.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPermissions",
      "Effect": "Allow",
      "Action": [
        "iam:GetInstanceProfile",
        "iam:CreateInstanceProfile",
        "iam:AddRoleToInstanceProfile",
        "iam:RemoveRoleFromInstanceProfile",
        "iam:DeleteInstanceProfile",
        "iam:GetRole",
        "iam:CreateRole",
        "iam:TagRole",
        "iam:PassRole",
        "iam:GetRolePolicy",
        "iam:AttachRolePolicy",
        "iam:PutRolePolicy",
        "iam:DetachRolePolicy",
        "iam:DeleteRolePolicy",
        "iam:UntagRole",
        "iam:DeleteRole"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ComputeAndEKS",
      "Effect": "Allow",
      "Action": [
        "autoscaling:*",
        "cloudformation:*",
        "elasticloadbalancing:*",
        "ec2:*",
        "eks:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ECRPushPull",
```

```
      "Effect": "Allow",
      "Action": [
        "ecr:CompleteLayerUpload",
        "ecr:DescribeImages",
        "ecr:GetAuthorizationToken",
        "ecr:DescribeRepositories",
        "ecr:UploadLayerPart",
        "ecr:InitiateLayerUpload",
        "ecr:BatchCheckLayerAvailability",
        "ecr:PutImage"
      ],
      "Resource": "*"
    }
  ]
}
```

2. Once the policy has been created, attach the policy to any principal that will be used to configure, create, and maintain the EKS cluster and node groups. For instructions on attaching policies, see Adding and removing IAM identity permissions in the AWS Identity and Access Management User Guide.

Create and Assign the EKS Cluster Developer Policy

The following IAM policy applies the minimum permissions needed for an EKS cluster developer. Follow the steps below to create the policy and attach it to the Anzo service account.

1. Refer to Creating IAM Policies in the AWS Identity and Access Management User Guide to create the following policy using your preferred method. You can save the contents below as a JSON file on your workstation and use the AWS CLI to create the policy, or you can paste the contents on the JSON tab if you use the IAM console.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Compute",
      "Effect": "Allow",
      "Action": [
        "ec2:*",
        "elasticloadbalancing:*",
```

```
        "autoscaling:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Pricing",
      "Effect": "Allow",
      "Action": [
        "pricing:GetProducts"
      ],
      "Resource": "*"
    },
    {
      "Sid": "EKSListAndDescribe",
      "Effect": "Allow",
      "Action": [
        "eks:ListUpdates",
        "eks:DescribeCluster",
        "eks:DescribeNodegroup", //Needed for GovCloud only
        "eks:ListClusters",
        "eks:ListNodegroups", //Needed for GovCloud only
        "eks:ListTagsForResource" //Needed for GovCloud only
      ],
      "Resource": "arn:aws:eks:*:*:cluster/*"
    },
    {
      "Sid": "ECRPull",
      "Effect": "Allow",
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:GetAuthorizationToken",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ],
      "Resource": "*"
    }
  ]
}
```

2. Once the policy has been created, attach the policy to the Anzo service user so that Anzo has permission to connect to the EKS services and deploy application pods. For instructions on

attaching policies, see Adding and removing IAM identity permissions in the AWS Identity and Access Management User Guide.

Once the IAM policies are in place and attached to principals, proceed to Creating the EKS Cluster for instructions on configuring and creating the cluster.

Creating the EKS Cluster

Follow the instructions below to define the EKS cluster resource requirements and then create the cluster based on your specifications.

- Define the EKS Cluster Requirements
- Example Cluster Configuration File
- (Optional) Define the IAM Role for K8s Service Accounts
- Create the EKS Cluster

Define the EKS Cluster Requirements

The first step in creating the K8s cluster is to define the infrastructure specifications. The configuration file to use for defining the specifications is called **k8s_cluster.conf**. Multiple sample k8s_cluster.conf files are included in the **eksctl** directory. Any of them can be copied and used as templates, or the files can be edited directly.

Sample k8s_cluster.conf Files

To help guide you in choosing the appropriate template for your use case, this section describes each of the sample files. Details about the parameters in the sample files are included in Cluster Parameters below.

eksctl/conf.d/k8s_cluster.conf

This file is a non-specific use case. It includes sample values for all of the available cluster parameters.

eksctl/sample_use_cases/1_existing_vpc_private_cluster/k8s_cluster.conf

This file includes sample values for a use case where:

- The EKS cluster will be deployed in a new private subnet in an existing VPC. You specify the existing VPC ID in the `VPC_ID` parameter.
- A NAT gateway is deployed to enable access to external services. If your VPC has an existing NAT gateway that you want to use, you can specify the CIDR for the existing gateway in the `NAT_SUBNET_CIDRS` parameter.
- The control plane security group is configured to allow access only from certain CIDRs. Those CIDRs are specified in the `ALLOW_NETWORK_CIDRS` parameter.
- Communication through your VPN can be enabled and routes can automatically be propagated to the route tables by including `ENABLE_ROUTE_PROPAGATION=true`.

eksctl/sample_use_cases/2_new_vpc_public_cluster/k8s_cluster.conf

This file includes sample values for a use case where:

- A new VPC will be created and the EKS cluster will be deployed into it. You specify the CIDR for the new VPC in the `VPC_CIDR` parameter.
- A NAT gateway is deployed to enable outbound connectivity for the cluster nodes.
- Public and private subnets will be created in the new VPC based on the CIDRs specified in the `PUBLIC_SUBNET_CIDRS` and `PRIVATE_SUBNET_CIDRS` parameters.
- Public access can be restricted to certain IP ranges by specifying the allowed CIDRs in the `ALLOW_NETWORK_CIDRS` parameter.
- Since a new VPC is created (rather than creating the cluster in the same VPC as Anzo) the new VPC must be configured to allow access from Anzo.

eksctl/sample_use_cases/3_nat_ha_private_cluster/k8s_cluster.conf

Like the **1_existing_vpc_private_cluster** sample file described above, this file includes sample values for a use case where:

- The EKS cluster will be deployed in a new private subnet in an existing VPC. You specify the existing VPC ID in the `VPC_ID` parameter.

- Multiple NAT gateways will be created, making the cluster highly available. One NAT gateway is deployed in each Availability Zone specified in the `AvailabilityZones` parameter. And a CIDR for each gateway needs to be specified in the `NAT_SUBNET_CIDRS` parameter. In addition, `VPC_NAT_MODE="HighlyAvailable"`.
- The control plane security group is configured to allow access only from certain CIDRs. Those CIDRs are specified in the `ALLOW_NETWORK_CIDRS` parameter.
- Communication through VPN is enabled to automatically propagate routes to the route tables by including `ENABLE_ROUTE_PROPAGATION=true`.

Cluster Parameters

The contents of `k8s_cluster.conf` are shown below. Descriptions of the cluster parameters follow the contents.

```
# AWS Configuration parameters
REGION="<region>"
AvailabilityZones="<zones>"
TAGS="<tags>"

# Networking configuration
VPC_ID="<vpc-id>"
VPC_CIDR="<vpc-cidr>"
NAT_SUBNET_CIDRS="<nat-subnet-cidr>"
PUBLIC_SUBNET_CIDRS="<public-subnet-cidr>"
PRIVATE_SUBNET_CIDRS="<private-subnet-cidr>"
VPC_NAT_MODE="<nat-mode>"
WARM_IP_TARGET="<warm-ip-target>"
PUBLIC_ACCESS_CIDRS="<public-access-cidrs>"
ALLOW_NETWORK_CIDRS="<allow-network-cidrs>"
ENABLE_ROUTE_PROPAGATION=<enable-route-propagation>

# EKS control plane configuration
CLUSTER_NAME="<name>"
CLUSTER_VERSION="<version>"
ENABLE_PRIVATE_ACCESS=<resources-vpc-config endpointPrivateAccess>
ENABLE_PUBLIC_ACCESS=<resources-vpc-config endpointPublicAccess>
CNI_VERSION="<cni-version>"

# Logging types: ["api","audit","authenticator","controllerManager","scheduler"]
ENABLE_LOGGING_TYPES="<logging-types>"
```

```
DISABLE_LOGGING_TYPES="<logging-types>"

# Common parameters
WAIT_DURATION=<wait-duration>
WAIT_INTERVAL=<wait-interval>
STACK_CREATION_TIMEOUT="<timeout>"
```

Parameter	Description
REGION	The AWS region for the EKS cluster. For example, **us-east-1**.
AvailabilityZones	A space-separated list of each of the Availability Zones in which you want to make the EKS cluster highly available. To ensure that the AWS EKS service can maintain high availability, you can list up to three Availability Zones. For example, **us-east-1a us-east-1b**.
TAGS	A comma-separated list of any labels that you want to add to the EKS cluster resources. Tags are optional key/value pairs that you define for categorizing resources.
VPC_ID	The ID of the VPC to provision the cluster into. Typically this value is the ID for the VPC that Anzo is deployed in. For example, **vpc-0dd06b24c819ec3e5**. **Note** If you want eksctl to create a new VPC, you can leave this value blank. However, after deploying the EKS cluster, you must configure the new VPC to make it routable from the Anzo VPC.
VPC_CIDR	The CIDR block to use for the VPC. For example, **10.107.0.0/16**. **Note** Supply this value even if VPC_ID is not set and a new VPC will be created.

Parameter	Description
NAT_SUBNET_CIDRS	A space-separated list of the CIDR blocks for the public subnets that will be used by the NAT gateway. For example, **10.107.0.0/24 10.107.5.0/24**. **Note** The number of CIDR blocks should equal the number of specified AvailabilityZones if you want the NAT gateway to be highly available.
PUBLIC_SUBNET_CIDRS	A space-separated list of the CIDR blocks for the public subnets. For example, **10.107.1.0/24 10.107.2.0/24**. For a private cluster, leave this value blank.
PRIVATE_SUBNET_CIDRS	A space-separated list of the CIDR blocks for the private subnets. For example, **10.107.3.0/24 10.107.4.0/24**.
VPC_NAT_MODE	The NAT mode for the VPC. Valid values are "HighlyAvailable," "Single," or "Disable." When this value is **HighlyAvailable** and multiple Availability Zones are specified in AvailabilityZones, a NAT gateway is deployed in each zone.
WARM_IP_TARGET	Specifies the "warm pool" or number of free IP addresses to keep available for pod assignment on each node so that there is less time spent waiting for IP addresses to be assigned when a pod is scheduled. Cambridge Semantics recommends that you set this value to **8**.
PUBLIC_ACCESS_CIDRS	A comma-separated list of the CIDR blocks that can access the K8s API server over the public endpoint.
ALLOW_NETWORK_CIDRS	A comma-separated list of the CIDR blocks that can access the K8s API over port 443.

Parameter	Description
ENABLE_ROUTE_PROPAGATION	Indicates whether to allow the virtual private gateway to automatically propagate routes to the route tables. This feature is useful when the cluster subnets need access to intranet/VPN routes.
CLUSTER_NAME	Name to give the EKS cluster. For example, **csi-k8s-cluster**.
CLUSTER_VERSION	The Kubernetes version of the EKS cluster.
ENABLE_PRIVATE_ACCESS	Indicates whether to enable private (VPC-only) access to the EKS cluster endpoint. This parameter accepts a "true" or "false" value and maps to the EKS `--resources-vpc-config endpointPrivateAccess` option. The default value in k8s_cluster.conf is **true**.
ENABLE_PUBLIC_ACCESS	Whether to enable public access to the EKS cluster endpoint. This parameter accepts a "true" or "false" value and maps to the EKS `--resources-vpc-config endpointPublicAccess` option. The default value in k8s_cluster.conf is **false**.
CNI_VERSION	An optional property that specifies the version of the VPC CNI plugin to use for pod networking.
ENABLE_LOGGING_TYPES	A comma-separated list of the logging types to enable for the cluster. Valid values are **api**, **audit**, **authenticator**, **controllerManager**, and **scheduler**. For information about the types, see Amazon EKS Control Plane Logging in the EKS documentation. The default value in k8s_cluster.conf is **api,audit** for Kubernetes API logging and Audit logs, which provide a record of the users, administrators, or system components that have affected the cluster.
DISABLE_LOGGING_TYPES	A comma-separated list of the logging types to disable for the cluster. Valid values are **api**, **audit**, **authenticator**, **controllerManager**, and

Parameter	Description
	scheduler. The default value in k8s_cluster.conf is **controllerManager,scheduler**, which disables the Kubernetes Controller Manager daemon as well as the Kubernetes Scheduler.
WAIT_DURATION	The number of seconds to wait before timing out during cluster resource creation. For example, **1200** means the creation of a resource will time out if it is not finished in 20 minutes.
WAIT_INTERVAL	The number of seconds to wait before polling for resource state information. The default value in k8s_cluster.conf is **10** seconds.
STACK_ CREATION_ TIMEOUT	The number of minutes to wait for stack creation to complete before aborting the creation.

Example Cluster Configuration File

An example completed k8s_cluster.conf file is shown below.

```
# AWS Configuration parameters
REGION="us-east-1"
AvailabilityZones="us-east-1a us-east-1b"
TAGS="Description=EKS Cluster"

# Networking configuration
VPC_ID="vpc-0dd06b24c819ec3e5"
VPC_CIDR="10.107.0.0/16"
NAT_SUBNET_CIDRS="10.107.0.0/24 10.107.5.0/24"
PUBLIC_SUBNET_CIDRS="10.107.1.0/24 10.107.2.0/24"
PRIVATE_SUBNET_CIDRS="10.107.3.0/24 10.107.4.0/24"
VPC_NAT_MODE="HighlyAvailable"
WARM_IP_TARGET="8"
PUBLIC_ACCESS_CIDRS="1.2.3.4/32,1.1.1.1/32"
ALLOW_NETWORK_CIDRS="10.108.0.0/16 10.109.0.0/16"
ENABLE_ROUTE_PROPAGATION=true

# EKS control plane configuration
```

```
CLUSTER_NAME="csi-k8s-cluster"
CLUSTER_VERSION="1.19"
ENABLE_PRIVATE_ACCESS=True
ENABLE_PUBLIC_ACCESS=False
CNI_VERSION="1.7.5"
# Logging types: ["api","audit","authenticator","controllerManager","scheduler"]
ENABLE_LOGGING_TYPES="api,audit"
DISABLE_LOGGING_TYPES="controllerManager,scheduler"

# Common parameters
WAIT_DURATION=1200
WAIT_INTERVAL=10
STACK_CREATION_TIMEOUT="30m"
```

(Optional) Define the IAM Role for K8s Service Accounts

For fine-grained permission management of the applications that run in the EKS cluster, you can associate an IAM role with a Kubernetes (K8s) Service Account. The Service Account can then be used to grant permissions to the pods in the cluster so that the container applications can use an AWS SDK or AWS CLI to make API requests to AWS services like S3 or Amazon RDS. For details, see IAM Roles for Service Accounts in the Amazon EKS documentation.

If you want to create a new IAM role with associated K8s Service Accounts during EKS cluster creation, you can define the Service Account requirements in the **iam_serviceaccounts.yaml** file in the `conf.d` directory. When you create the cluster, there is a prompt that asks if you want to update IAM properties for the cluster. Responding **y** (yes) creates the account based on the specifications in iam_serviceaccounts.yaml. The contents of the file are shown below. Descriptions of the parameters follow the contents.

```
            apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: <eks-cluster-name>
  region: <cluster-region>
iam:
  withOIDC: true
  serviceAccounts:
  - metadata:
      name: <service-account-name>
      namespace: <namespace>
      labels: {<label-name>: "<value>"}
```

```
    attachPolicyARNs:
    - "<arn>"
    tags:
      <tag-name>: "<value>"
  - metadata:
      name: <service-account-name>
      namespace: <namespace>
      labels: {<label-name>: "<value>"}
    attachPolicyARNs:
    - "<arn>"
    tags:
      <tag-name>: "<value>"
    wellKnownPolicies:
      <policy>: <enable-policy>
    roleName: <role-name>
    roleOnly: <role-only>
```

Parameter	Description
apiVersion	The version of the schema for this object.
kind	The schema for this object.
name	The name of the EKS cluster (CLUSTER_NAME) to create the Service Accounts for. For example, **csi-k8s-cluster**.
region	The region that the EKS cluster is deployed in (REGION). For example, **us-east-1**.
withOIDC	Indicates whether to enable the IAM OpenID Connect Provider (OIDC) as well as IRSA for the Amazon CNI plugin. This value must be **true**. Amazon requires OIDC to use IAM roles for Service Accounts.
serviceAccounts	There are multiple `- metadata` sequences under serviceAccounts. Each sequence supplies the metadata for one Service Account. You can include any number of metadata sequences to create multiple Service Accounts.

Parameter	Description

	`- metadata:`
	`name: <service-account-name>`
	`namespace: <namespace>`
	`labels: {<label-name>: "<value>"}`
	`attachPolicyARNs:`
	`- "<arn>"`
	`tags:`
	`<tag-name>: "<value>"`

name	The name to use for the Service Account.
namespace	The namespace to create the Service Account in. If the namespace you specify does not exist, a new namespace is created. If namespace is not specified, **default** is used.
labels	An optional list of labels to add to the Service Account.
attachPolicyARNs	A list of the Amazon Resource Names (ARN) for the IAM policies to attach to the Service Account.
tags	An optional list of tags to add to the Service Account.
wellKnownPolicies	A list of any common AWS IAM policies that you want to attach to the Service Accounts, such as imageBuilder, autoScaler, awsLoadBalancerController, or certManager. For a complete list of the supported well-known policies, see the eksctl Config File Schema.
roleName	The name for the new Service Account IAM Role.
roleOnly	Indicates whether to annotate the Service Accounts with the ARN of the new IAM Role (eks.amazonaws.com/role-arn). Cambridge Semantics

Parameter	Description
	recommends that you set this value to **true**.

Example IAM Role for Service Accounts Configuration File

An example completed iam_serviceaccounts.yaml file is shown below. This example creates a role called S3ReadRole with one Service Account that gives AnzoGraph containers read-only access to Amazon S3.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: csi-k8s-cluster
  region: us-east-1
iam:
  withOIDC: true
  serviceAccounts:
  - metadata:
      name: s3-reader
      namespace: anzograph
      labels: {app: "database"}
    attachPolicyARNs:
    - "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
    tags:
      Team: "AnzoGraph Deployment"
    wellKnownPolicies:
      autoScaler: true
    roleName: S3ReadRole
    roleOnly: true
```

Create the EKS Cluster

After defining the cluster requirements, run the **create_k8s.sh** script in the `eksctl` directory to create the cluster.

Note

The `create_k8s.sh` script references the files in the `eksctl/reference` directory. If you customized the directory structure on the workstation, ensure that the **reference** directory is available at the same level as `create_k8s.sh` before creating the cluster.

Run the script with the following command. The arguments are described below.

```
./create_k8s.sh -c <config_file_name> [ -d <config_file_directory> ] [ -f | --force ] [ -h | --help ]
```

Argument	Description
-c <config_ file_name>	This is a **required** argument that specifies the name of the configuration file that supplies the cluster requirements. For example, **-c k8s_cluster.conf**.
-d <config_ file_ directory>	This is an **optional** argument that specifies the path and directory name for the configuration file specified for the -c argument. If you are using the original `eksctl` directory file structure and the configuration file is in the `conf.d` directory, you do not need to specify the -d argument. If you created a separate directory structure for different Anzo environments, include the -d option. For example, **-d /eksctl/env1/conf**.
-f | --force	This is an **optional** argument that controls whether the script prompts for confirmation before proceeding with each stage involved in creating the cluster. If **-f** (**--force**) is specified, the script assumes the answer is "yes" to all prompts and does not display them.
-h | --help	This argument is an **optional** flag that you can specify to display the help from the create_k8s.sh script.

For example, the following command runs the create_k8s script, using k8s_cluster.conf as input to the script. Since k8s_cluster.conf is in the conf.d directory, the -d argument is excluded:

```
./create_k8s.sh -c k8s_cluster.conf
```

The script validates that the required software packages, such as the aws-cli, eksctl, and kubectl, are installed and that the versions are compatible with the script. It also displays an overview of the deployment details based on the values in the configuration file.

The script then prompts you to proceed with deploying each component of the EKS cluster infrastructure. Type **y** (yes) and press **Enter** to proceed with each step in creating the specified network, cluster, Internet gateway, NAT gateway, route table, and security group resources. All resources are created according to the specifications in the configuration file. Once the cluster resources are deployed, the script asks whether you would like to update IAM properties for the cluster. Continue to Configuring Cluster IAM Properties below for background information and details on configuring IAM properties.

Configuring Cluster IAM Properties

At the final stage of EKS cluster creation, the last few prompts are related to IAM properties.

First, you are asked about IAM roles for K8s Service Accounts. If you want to create Service Accounts, as described in (Optional) Define the IAM Role for K8s Service Accounts, answer **y** (yes) to the prompt **Do you want to update IAM properties for cluster?** Service Accounts will be created according to the specifications in iam_serviceaccounts.yaml. If you do not want to create Service Accounts, answer **n** (no).

The last prompt is related to IAM identity mapping for the EKS cluster. Only the IAM entity that created the cluster has **system:masters** permission for the cluster and its K8s services. To grant additional AWS users or roles the ability to interact with the cluster, IAM identity mapping must be performed by adding the **aws-auth** ConfigMap to the EKS cluster configuration (see Managing Users or IAM Roles for your Cluster in the Amazon EKS documentation).

To aid you in updating the ConfigMap so that additional users can access the cluster, the create_k8s.sh script includes prompts that ask for the required ConfigMap information. If you want to update the ConfigMap, answer **y** (yes) to the **Do you want to add IAM users to control access to cluster** prompt. The script prompts for the following values, which will be used to update `mapRoles` and/or `mapUsers` in aws-auth ConfigMap:

- **Account ID**: The AWS account ID where the EKS cluster is deployed.
- **User Name**: The username within Kubernetes to map to the IAM role. For example, **admin**.
- **RBAC Group**: The Kubernetes group to map the IAM role to. For example, **system:masters**.
- **Service Name**: This value must be **emr-containers**.
- **Namespace**: The namespace to create RBAC resources in.
- **User or Role ARN**: The Amazon Resource Name for the IAM role or user to create. For example, **arn:aws:iam::105333188789:role/admin**.

When cluster creation is complete, proceed to Creating the Required Node Groups to add the required node groups to the cluster.

Creating the Required Node Groups

This topic provides instructions for creating the four types of required node groups:

- The **Common** node group for running K8s services such as the Cluster Autoscaler and Load Balancers.
- The **Operator** node group for running the AnzoGraph, Anzo Agent with Anzo Unstructured (AU) and Elasticsearch operator pods.
- The **AnzoGraph** node group for running AnzoGraph application pods.
- The **Dynamic** node group for running Anzo Agent with AU and Elasticsearch application pods.

> **Tip**
> For more information about the node groups, see Anzo Kubernetes Requirements.

- Define the Node Group Requirements
- Example Configuration Files
- Create the Node Groups

Define the Node Group Requirements

Before creating the node groups, configure the infrastructure requirements for each type of group. The **nodepool_*.yaml** object files in the `eksctl/conf.d` directory are sample configuration files that you can use as templates, or you can edit the files directly:

- **nodepool_common.yaml** defines the requirements for the Common node group.
- **nodepool_operator.yaml** defines the requirements for the Operator node group.
- **nodepool_anzograph.yaml** defines the requirements for the AnzoGraph node group.
- **nodepool_dynamic.yaml** defines the requirements for the Dynamic node group.

Each type of node group configuration file contains the following parameters. Descriptions of the parameters and guidance on specifying the appropriate values for each type of node group are provided below.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: <eks-cluster-name>
  region: <cluster-region>
  tags:
    <metadata-tag-name>: "<value>"
managedNodeGroups:
  - name: <node-prefix>
    amiFamily: <ami-type>
    labels:
      <label-name>: '<value>'
    instanceType: <instance-type>
    desiredCapacity: <desired-capacity>
    availabilityZones:
    - <zones>
    minSize: <min-size>
    maxSize: <max-size>
    volumeSize: <volume-size>
    maxPodsPerNode: <max-pods>
    iam:
      attachPolicyARNs:
      - <arns>
      withAddonPolicies:
```

```
      autoScaler: <auto-scaler>
      imageBuilder: <image-builder>
      efs: <efs>
      cloudWatch: <cloud-watch>
  volumeType: <volume-type>
  privateNetworking: <private-networking>
  ssh:
    allow: <allow-ssh>
    publicKeyName: <public-key-name>
  taints:
    '<taint-name>': '<taint-value>'
  tags:
    '<tag-name>': '<tag-value>'
  asgMetricsCollection:
    - granularity: <granularity>
      metrics:
      - <metric-name>
```

apiVersion

The version of the schema for this object.

kind

The schema for this object.

name

The name of the EKS cluster that hosts the node group. For example, **csi-k8s-cluster**.

region

The region that the EKS cluster is deployed in. For example, **us-east-1**.

tags

A list of any custom tags to add to the AWS resources that are created by eksctl.

name

The prefix to add to the names of the nodes that are deployed in this node group.

Node Group Type	Sample name Value
Common	common
Operator	operator
AnzoGraph	anzograph
Dynamic	dynamic

amiFamily

The EKS-optimized Amazon Machine Image (AMI) type to use when deploying nodes in the node group. Cambridge Semantics recommends that you specify **AmazonLinux2**.

labels

A space-separated list of key/value pairs that define the type of pods that can be placed on the nodes in this node group. One label, `cambridgesemantics.com/node-purpose`, is **required** for each type of node group. The node-purpose label indicates that the purpose of the nodes in the groups are to host operator, anzograph, dynamic, or common pods. Labels are used to attract pods to nodes, while "taints" (described in taints below) are used to repel other types of pods from being placed in this node group. The table below lists the required labels for each node group.

Node Group Type	Required nodeGroups labels Value
Common	cambridgesemantics.com/node-purpose: 'common' deploy-ca: 'true' cluster-autoscaler-version: '<version>'
Operator	cambridgesemantics.com/node-purpose: 'operator'
AnzoGraph	cambridgesemantics.com/node-purpose: 'anzograph'

Node Group Type	Required nodeGroups labels Value
Dynamic	cambridgesemantics.com/node-purpose: 'dynamic'

Note

The additional Common node group label **deploy-ca: 'true'** identifies this group as the node group to host the Cluster Autoscaler (CA) service. The related **cluster-autoscaler-version** label identifies the CA version. The version that you specify must have the same major and minor version as the Kubernetes version for the EKS cluster (CLUSTER_VERSION). For example, if the cluster version is 1.24, the CA version must be 1.24.*n*, where *n* is a valid CA patch release number, such as 1.24.1. To view the CA releases for your Kubernetes version, see Cluster Autoscaler Releases on GitHub.

instanceType

The EC2 instance type to use for the nodes in the node group.

Node Group Type	Sample instanceType Value
Common	m5.large
Operator	m5.large
AnzoGraph	m5.8xlarge
Dynamic	m5.2xlarge

Tip

For more guidance on determining the instance types to use for nodes in the required node groups, see Compute Resource Planning.

desiredCapacity

The number of nodes to deploy when this node group is created. This value must be set to at least **1**. When you create the node group, at least one node in the group needs to be deployed as well. However, if **minSize** is **0** and the **autoScaler** addon is enabled, the autoscaler will deprovision this node because it is not in use.

availabilityZones

A list of the Availability Zones to make this node group available to.

minSize

The minimum number of nodes for the node group. If you set the minimum size to **0**, nodes will not be provisioned unless a pod is scheduled for deployment in that group.

maxSize

The maximum number of nodes that can be deployed in the node group.

volumeSize

The size (in GB) of the EBS volume to add to the nodes in this node group.

maxPodsPerNode

The maximum number of pods that can be hosted on a node in this node group.In addition to Anzo application pods, this limit also needs to account for K8s service pods and helper pods. Cambridge Semantics recommends that you set this value to at least **16** for all node group types.

attachPolicyARNs

A list of the Amazon Resource Names (ARN) for the IAM policies to attach to the node group. These policies apply at the node level. Include the default node policies as well as any other policies that you want to add. For example:

```
attachPolicyARNs:
- arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
- arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
- arn:aws:iam::aws:policy/AmazonS3FullAccess
```

autoScaler

Indicates whether to add an autoscaler to this node group. Cambridge Semantics recommends that you set this value to **true**.

imageBuilder

Indicates whether to allow this node group to access the full Elastic Container Registry (ECR). Cambridge Semantics recommends that you set this value to **true**.

efs

Indicates whether to enable access to the persistent volume, Elastic File System (EFS).

cloudWatch

Indicates whether to enable the CloudWatch service, which performs control plane logging when the node group is created.

volumeType

The type of EBS volume to use for the nodes in this node group.

privateNetworking

Indicates whether to isolate the node group from the public internet. Cambridge Semantics recommends that you set this value to **true**.

allow

Indicates whether to allow SSH access to the nodes in this node group.

publicKeyName

The public key name in EC2 to add to the nodes in this node group. If allow is false, this value is ignored.

taints

This parameter defines the type of pods that are allowed to be placed in this node group. When a pod is scheduled for deployment, the scheduler relies on this value to determine whether the pod belongs in this group. If a pod has a **toleration** that is not compatible with this **taint**, the pod is rejected from the group. The following recommended values specify that pods must be operator pods to be deployed in the Operator node group; they must be anzograph pods to be deployed in the AnzoGraph node group; and they must be dynamic pods to be deployed in the Dynamic node group. The **NoSchedule** value means a toleration is required and pods without a toleration will not be allowed in the group.

Node Group Type	Recommended taints Value
Operator	'cambridgesemantics.com/dedicated': 'operator:NoSchedule'
AnzoGraph	'cambridgesemantics.com/dedicated': 'anzograph:NoSchedule'
Dynamic	'cambridgesemantics.com/dedicated': 'dynamic:NoSchedule'

tags

The list of key:value pairs to add to the nodes in this node group. For autoscaling to work, the list of tags must include the namespaced version of the label and taint definitions.

Node Group	Recommended tags Value
Common	'k8s.io/cluster-autoscaler/node-template/label/cambridgesemantics.com/node-purpose': 'common'

Node Group	Recommended tags Value
Operator	'k8s.io/cluster-autoscaler/node-template/label/cambridgesemantics.com/node-purpose': 'operator' 'k8s.io/cluster-autoscaler/node-template/taint/cambridgesemantics.com/dedicated': 'operator:NoSchedule' 'cambridgesemantics.com/node-purpose': 'operator'
AnzoGraph	'k8s.io/cluster-autoscaler/node-template/label/cambridgesemantics.com/node-purpose': 'anzograph' 'k8s.io/cluster-autoscaler/node-template/taint/cambridgesemantics.com/dedicated': 'anzograph:NoSchedule' 'cambridgesemantics.com/node-purpose': 'anzograph'
Dynamic	'k8s.io/cluster-autoscaler/node-template/label/cambridgesemantics.com/node-purpose': 'dynamic' 'k8s.io/cluster-autoscaler/node-template/taint/cambridgesemantics.com/dedicated': 'dynamic:NoSchedule' 'cambridgesemantics.com/node-purpose': 'dynamic'

Tip

You can also augment the required tags with any custom tags that you want to include. For information about tagging, see Tagging your Amazon EKS Resources in the Amazon EKS documentation.

asgMetricsCollection

If cloudWatch is enabled, this parameter configures the specific Auto Scaling Group (ASG) metrics to capture as well as the frequency with which to capture the metrics.

granularity

This property is a required property that specifies the frequency with which Amazon EC2 Auto Scaling sends aggregated data to CloudWatch. The only valid value is **1Minute**.

metrics

This property lists the specific group-level metrics to collect. If **granularity** is specified but **metrics** is omitted, all of the metrics are enabled. For more information and a list of valid values, see AutoScalingGroup MetricsCollection in the AWS CloudFormation documentation.

Example Configuration Files

Example completed configuration files for each type of node group are shown below.

Common Node Group

The example below shows a completed nodepool_common.yaml file.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: csi-k8s-cluster
  region: us-east-1
  tags:
    description: "K8s cluster Common node group"
managedNodeGroups:
  - name: common
    amiFamily: AmazonLinux2
    labels:
      cambridgesemantics.com/node-purpose: 'common'
      deploy-ca: 'true'
      cluster-autoscaler-version: '1.24.1'
    instanceType: m5.large
    desiredCapacity: 1
    availabilityZones:
    - us-east-1a
    minSize: 0
    maxSize: 4
    volumeSize: 50
    maxPodsPerNode: 16
    iam:
      attachPolicyARNs:
      - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
      - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
      - arn:aws:iam::aws:policy/AmazonS3FullAccess
      withAddonPolicies:
        autoScaler: true
```

```
        imageBuilder: true
        efs: true
        CloudWatch: true
    volumeType: gp2
    privateNetworking: true
    ssh:
      allow: true
      publicKeyName: common-keypair
    tags:
      'k8s.io/cluster-autoscaler/node-template/label/cambridgesemantics.com/node-
purpose': 'common'
    asgMetricsCollection:
      - granularity: 1Minute
        metrics:
        - GroupPendingInstances
        - GroupInServiceInstances
        - GroupTerminatingInstances
        - GroupInServiceCapacity
```

Operator Node Group

The example below shows a completed nodepool_operator.yaml file.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: csi-k8s-cluster
  region: us-east-1
  tags:
    description: "K8s cluster Operator node group"
managedNodeGroups:
  - name: operator
    amiFamily: AmazonLinux2
    labels:
      cambridgesemantics.com/node-purpose: 'operator'
    instanceType: m5.large
    desiredCapacity: 1
    availabilityZones:
    - us-east-1a
    minSize: 0
    maxSize: 5
    volumeSize: 50
    maxPodsPerNode: 16
```

```
    iam:
      attachPolicyARNs:
      - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
      - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
      - arn:aws:iam::aws:policy/AmazonS3FullAccess
      withAddonPolicies:
        autoScaler: true
        imageBuilder: true
        efs: true
        cloudWatch: true
    volumeType: gp2
    privateNetworking: true
    ssh:
      allow: true
      publicKeyName: operator-keypair
    taints:
      'cambridgesemantics.com/dedicated': 'operator:NoSchedule'
    tags:
      'k8s.io/cluster-autoscaler/node-template/label/cambridgesemantics.com/node-
purpose': 'operator'
      'k8s.io/cluster-autoscaler/node-template/taint/cambridgesemantics.com/dedicated':
'operator:NoSchedule'
      'cambridgesemantics.com/node-purpose': 'operator'
    asgMetricsCollection:
      - granularity: 1Minute
        metrics:
        - GroupPendingInstances
        - GroupInServiceInstances
        - GroupTerminatingInstances
        - GroupInServiceCapacity
```

AnzoGraph Node Group

The example below shows a completed nodepool_anzograph.yaml file.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: csi-k8s-cluster
  region: us-east-1
  tags:
    description: "K8s cluster AnzoGraph node group"
managedNodeGroups:
```

```
  - name: anzograph
    amiFamily: AmazonLinux2
    labels:
      cambridgesemantics.com/node-purpose: 'anzograph'
    instanceType: m5.8xlarge
    desiredCapacity: 1
    availabilityZones:
    - us-east-1a
    minSize: 0
    maxSize: 12
    volumeSize: 100
    maxPodsPerNode: 16
    iam:
      attachPolicyARNs:
      - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
      - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
      - arn:aws:iam::aws:policy/AmazonS3FullAccess
      withAddonPolicies:
        autoScaler: true
        imageBuilder: true
        efs: true
        CloudWatch: true
    volumeType: gp2
    privateNetworking: true
    ssh:
      allow: true
      publicKeyName: anzograph-keypair
    taints:
      'cambridgesemantics.com/dedicated': 'anzograph:NoSchedule'
    tags:
      'k8s.io/cluster-autoscaler/node-template/label/cambridgesemantics.com/node-
purpose': 'anzograph'
      'k8s.io/cluster-autoscaler/node-template/taint/cambridgesemantics.com/dedicated':
'anzograph:NoSchedule'
      'cambridgesemantics.com/node-purpose': 'anzograph'
    asgMetricsCollection:
      - granularity: 1Minute
        metrics:
        - GroupPendingInstances
        - GroupInServiceInstances
        - GroupTerminatingInstances
        - GroupInServiceCapacity
```

Dynamic Node Group

The example below shows a completed nodepool_dynamic.yaml file.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: csi-k8s-cluster
  region: us-east-1
  tags:
    description: "K8s cluster Dynamic node group"
nodeGroups:
  - name: dynamic
    amiFamily: AmazonLinux2
    labels:
      cambridgesemantics.com/node-purpose: 'dynamic'
    instanceType: m5.2xlarge
    desiredCapacity: 1
    availabilityZones:
    - us-east-1a
    minSize: 0
    maxSize: 12
    volumeSize: 100
    maxPodsPerNode: 16
    iam:
      attachPolicyARNs:
      - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
      - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
      - arn:aws:iam::aws:policy/AmazonS3FullAccess
      withAddonPolicies:
        autoScaler: true
        imageBuilder: true
        efs: true
        CloudWatch: true
    volumeType: gp2
    privateNetworking: true
    ssh:
      allow: true
      publicKeyName: dynamic-keypair
    taints:
      'cambridgesemantics.com/dedicated': 'dynamic:NoSchedule'
    tags:
      'k8s.io/cluster-autoscaler/node-template/label/cambridgesemantics.com/node-
```

```
purpose': 'dynamic'
      'k8s.io/cluster-autoscaler/node-template/taint/cambridgesemantics.com/dedicated':
'dynamic:NoSchedule'
      'cambridgesemantics.com/node-purpose': 'dynamic'
    asgMetricsCollection:
      - granularity: 1Minute
        metrics:
        - GroupPendingInstances
        - GroupInServiceInstances
        - GroupTerminatingInstances
        - GroupInServiceCapacity
```

Create the Node Groups

After defining the requirements for the node groups, run the **create_nodepools.sh** script in the `eksctl` directory to create each type of node group. Run the script once for each type of group.

> **Note**
> The `create_nodepools.sh` script references the files in the `eksctl/reference` directory. If you customized the directory structure on the workstation, ensure that the **reference** directory is available at the same level as `create_nodepools.sh` before creating the node groups.

Run the script with the following command. The arguments are described below.

```
./create_nodepools.sh -c <config_file_name> [ -d <config_file_directory> ] [ -f | --
force ] [ -h | --help ]
```

> **Important**
> It is important to create the Common node group first. The Cluster Autoscaler and other core cluster services are dependent on the Common node group.

Argument	Description
-c <config_ file_name>	This is a **required** argument that specifies the name of the configuration file (i.e., `nodepool_common.yaml`, `nodepool_operator.yaml`, `nodepool_ anzograph.yaml`, or `nodepool_dynamic.yaml`) that supplies the node group

Argument	Description
	requirements. For example, **-c nodepool_dynamic.yaml**.
-d <config_ file_ directory>	This is an **optional** argument that specifies the path and directory name for the configuration file specified for the -c argument. If you are using the original `eksctl` directory file structure and the configuration file is in the `conf.d` directory, you do not need to specify the -d argument. If you created a separate directory structure for different Anzo environments, include the -d option. For example, **-d /eksctl/env1/conf**.
-f | --force	This is an **optional** argument that controls whether the script prompts for confirmation before proceeding with each stage involved in creating the node group. If **-f** (**--force**) is specified, the script assumes the answer is "yes" to all prompts and does not display them.
-h | --help	This argument is an **optional** flag that you can specify to display the help from the create_nodepools.sh script.

For example, the following command runs the create_nodepools script, using nodepool_ common.yaml as input to the script. Since nodepool_common.yaml is in the conf.d directory, the -d argument is excluded:

```
./create_nodepools.sh -c nodepool_common.yaml
```

The script validates that the required software packages, such as aws-cli, eksctl, and kubectl, are installed and that the versions are compatible with the script. It also displays an overview of the deployment details based on the values in the specified configuration file.

The script then prompts you to proceed with deploying each component of the node group. Type **y** and press **Enter** to proceed with the configuration.

Once the Common, Operator, AnzoGraph, and Dynamic node groups are created, the next step is to create a Cloud Location in Anzo so that Anzo can connect to the EKS cluster and deploy applications. See Connecting to a Cloud Location in the Administration Guide.

Google Kubernetes Engine Deployments

The topics in this section guide you through the process of deploying all of the Google Kubernetes Engine (GKE) infrastructure that is required to support dynamic deployments of Anzo components. The topics provide instructions for setting up a workstation to use for deploying the K8s infrastructure, performing the prerequisite tasks before deploying the GKE cluster, creating the GKE cluster, and creating the required node pools.

In this section:

Setting Up a Workstation

This topic provides the requirements and instructions to follow for configuring a workstation to use for creating and managing the GKE infrastructure. The workstation needs to be able to connect to the Google Cloud API. It also needs to have the required Google Cloud and Kubernetes (K8s) software packages as well as the deployment scripts and configuration files supplied by Cambridge Semantics. This workstation will be used to connect to the Google Cloud API and provision the K8s cluster and node pools.

Note

You can use the Anzo server as the workstation if the network routing and security policies permit the Anzo server to access the Google Cloud and K8s APIs. When deciding whether to use the Anzo server as the K8s workstation, consider whether Anzo may be migrated to a different server or VPC in the future.

- Review the Requirements and Install the Software
- Download the Cluster Creation Scripts and Configuration Files

Review the Requirements and Install the Software

The table below lists the requirements for the K8s workstation.

Component	Requirement
Operating System	The operating system for the workstation must be **RHEL/CentOS 7.8 or higher**.
Networking	The workstation should be in the same VPC network as the GKE cluster. If it is not in the same VPC, make sure that it is on a network that is routable from the cluster's VPC.
Software	• **Kubectl**: Cambridge Semantics recommends that you use the same kubectl version as the GKE cluster version. For instructions, see Install Kubectl below. • **Google Cloud SDK** is required. For installation instructions, see Install the Google Cloud SDK below.
CSI GCLOUD Package	Cambridge Semantics provides **gcloud** scripts and configuration files to use for provisioning the GKE cluster and node pools. Download the files to the workstation. See Download the Cluster Creation Scripts and Configuration Files for more information about the gcloud package.

Install Kubectl

Follow the instructions below to install kubectl on your workstation. Cambridge Semantics recommends that you install the same version of kubectl as the K8s cluster API. For more information, see Install and Set Up kubectl on Linux in the Kubernetes documentation.

1. Run the following cURL command to download the kubectl binary:

```
curl -LO https://dl.k8s.io/release/<version>/bin/linux/amd64/kubectl
```

Where <version> is the version of kubectl to install. For example, the following command downloads version 1.19.12:

```
curl -LO https://dl.k8s.io/release/v1.19.12/bin/linux/amd64/kubectl
```

2. Run the following command to make the binary executable:

```
chmod +x ./kubectl
```

3. Run the following command to move the binary to your PATH:

```
sudo mv ./kubectl /usr/local/bin/kubectl
```

4. To confirm that the binary is installed and that you can run kubectl commands, run the following command to display the client version:

```
kubectl version --client
```

The command returns the following type of information. For example:

```
Client Version: version.Info{Major:"1", Minor:"19", GitVersion:"v1.19.12",
GitCommit:"f3abc15296f3a3f54e4ee42e830c61047b13895f",
GitTreeState:"clean", BuildDate:"2021-06-16T13:21:12Z", GoVersion:"go1.13.15",
Compiler:"gc", Platform:"linux/amd64"}
```

Install the Google Cloud SDK

Follow the instructions below to install the Google Cloud SDK on your workstation.

1. Run the following command to configure access to the Google Cloud repository:

```
sudo tee -a /etc/yum.repos.d/google-cloud-sdk.repo << EOM
[google-cloud-sdk]
name=Google Cloud SDK
baseurl=https://packages.cloud.google.com/yum/repos/cloud-sdk-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOM
```

2. Run the following command to install google-cloud-sdk:

```
sudo yum install google-cloud-sdk
```

The following packages are installed:

```
google-cloud-sdk-app-engine-grpc
google-cloud-sdk-pubsub-emulator
google-cloud-sdk-app-engine-go
google-cloud-sdk-cloud-build-local
google-cloud-sdk-datastore-emulator
google-cloud-sdk-app-engine-python
google-cloud-sdk-cbt
google-cloud-sdk-bigtable-emulator
google-cloud-sdk-datalab
google-cloud-sdk-app-engine-java
```

3. Next, configure the default project and region settings for the Cloud SDK:
   a. Run the following command to set the default project for the GKE cluster:

   ```
   gcloud config set project <project_ID>
   ```

   Where <project_ID> is the Project ID for the project in which the GKE cluster will be provisioned.

   b. If you work with zonal clusters, run the following command to set the default compute zone for the GKE cluster:

   ```
   gcloud config set compute/zone <compute_zone>
   ```

   Where <compute_zone> is the default compute zone for the GKE cluster. For example:

   ```
   gcloud config set compute/zone us-central1-a
   ```

   c. If you work with regional clusters, run the following command to set the default region for the GKE cluster:

   ```
   gcloud config set compute/region <compute_region>
   ```

   Where <compute_region> is the default region for the GKE cluster. For example:

   ```
   gcloud config set compute/region us-east1
   ```

d. To make sure that you are using the latest version of the Cloud SDK, run the following command to check for updates:

```
gcloud components update
```

Download the Cluster Creation Scripts and Configuration Files

The Cambridge Semantics GitHub repository, k8s-genesis (https://github.com/cambridgesemantics/k8s-genesis.git), includes all of the files that are needed to manage the configuration, creation, and deletion of the GKE cluster and node pools.

You can clone the repository to any location on the workstation or download the k8s-genesis package as a ZIP file, copy the file to the workstation, and extract the contents. The k8s-genesis directory includes three subdirectories (one for each supported Cloud Service Provider), the license information, and a readme file:

```
k8s-genesis
├── aws
├── azure
├── gcp
├── LICENSE
└── README.md
```

Navigate to `/gcp/k8s/gcloud`. The **gcloud** directory contains all of the GKE cluster and node pool configuration files. You can remove all other directories from the workstation. The gcloud files and subdirectories are shown below:

```
gcloud
├── common.sh
├── conf.d
│   ├── k8s_cluster.conf
│   ├── nodepool_anzograph.conf
│   ├── nodepool_anzograph_tuner.yaml
│   ├── nodepool_common.conf
│   ├── nodepool.conf
│   ├── nodepool_dynamic.conf
│   ├── nodepool_dynamic_tuner.yaml
│   └── nodepool_operator.conf
├── create_k8s.sh
├── create_nodepools.sh
├── delete_k8s.sh
```

```
├── delete_nodepools.sh
├── gcloud_cli_common.sh
├── README.md
└── sample_use_cases
        ├── 1_usePrivateEndpoint_private_cluster
        │   └── k8s_cluster.conf
        ├── 2_public_cluster
        │   └── k8s_cluster.conf
        ├── 3_useAuthorizedNetworks
        │   └── k8s_cluster.conf
        └── 4_providePublicEndpointAccess
                └── k8s_cluster.conf
```

The following list gives an overview of the files. Subsequent topics describe the files in more detail.

- The **common.sh** and **gcloud_cli_common.sh** scripts are used by the **create*.sh** and **delete*.sh** scripts when the GKE cluster and node pools are created or deleted.
- The **conf.d** directory contains the configuration files that supply the specifications to follow when creating the K8s cluster and node pools.
  - **k8s_cluster.conf**: Supplies the specifications for the GKE cluster.
  - **nodepool_anzograph.conf**: Supplies the specifications for the AnzoGraph node pool.
  - **nodepool_anzograph_tuner.conf**: Supplies the kernel-level tuning and security policies to apply to AnzoGraph runtime environments.
  - **nodepool_common.conf**: Supplies the specifications for a Common node pool. The Common node pool is not required for GKE deployments, and this configuration file is typically not used.
  - **nodepool.conf**: This file is supplied as a reference. It contains the superset of node pool parameters.
  - **nodepool_dynamic.conf**: Supplies the specifications for the Dynamic node pool.
  - **nodepool_dynamic_tuner.conf**: Supplies the kernel-level tuning and security policies to apply to Dynamic runtime environments.
  - **nodepool_operator.conf**: Supplies the specifications for the Operator node pool.

- The **create_k8s.sh** script is used to deploy the GKE cluster.
- The **create_nodepools.sh** script is used to deploy node pools in the GKE cluster.
- The **delete_k8s.sh** script is used to delete the GKE cluster.
- The **delete_nodepools.sh** script is used to remove node pools from the GKE cluster.
- The **sample_use_cases** directory contains sample GKE cluster configuration files that you can refer to or use as a template for configuring your GKE cluster depending on your use case:
  - The **k8s_cluster.conf** file in the **1_usePrivateEndpoint_private_cluster** directory is a sample file for a use case where you want to deploy the GKE cluster in an existing network that does not have public internet access.
  - The **k8s_cluster.conf** file in the **2_public_cluster** directory is a sample file for a use case where you want to deploy the GKE cluster into a new network with public internet access.
  - The **k8s_cluster.conf** file in the **3_useAuthorizedNetworks** directory is a sample file for a use case where you want to deploy the GKE cluster into a private network with master authorized networks.
  - The **k8s_cluster.conf** file in the **4_providePublicEndpointAccess** directory is a sample file for a use case where you want to deploy the GKE cluster into a private network that has public endpoint access enabled.

Once the workstation is configured, see Planning the Anzo and GKE Network Architecture to review information about the network architecture that the gcloud scripts create. And see Creating and Assigning IAM Roles for instructions on creating the IAM roles that are needed for assigning permissions to create and use the GKE cluster.

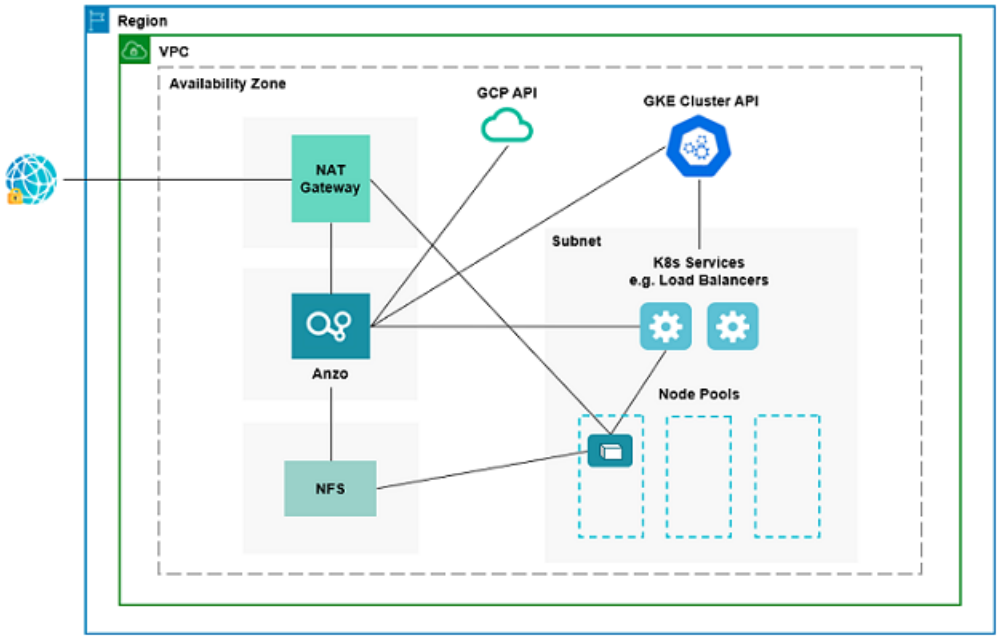
Planning the Anzo and GKE Network Architecture

This topic describes the network architecture that supports the Anzo and GKE integration.

Note

When you deploy the K8s infrastructure, Cambridge Semantics strongly recommends that you create the GKE cluster in the same VPC network as Anzo. If you create the GKE cluster in a new VPC, you must configure the new VPC to be routable from the Anzo VPC.

The diagram below shows the typical network components that are employed when a GKE cluster is integrated with Anzo. Most of the network resources shown in the diagram are automatically deployed (and the appropriate routing is configured) according to the values that you supply in the cluster and node pool .conf files in the **gcloud** package on the workstation.



In the diagram, there are two components that you deploy before configuring and creating the K8s resources:

- **Anzo**: Since the Anzo server is typically deployed before the K8s components, you specify the Anzo network when creating the GKE cluster, ensuring that Anzo and all of the GKE cluster components are in the same network and can talk to each other. Also, make sure that Anzo has access to the GCP and GKE APIs.
- **NFS**: You are required to create a network file system (NFS). However, Anzo automatically mounts the NFS to the nodes when AnzoGraph, Anzo Unstructured, and Elasticsearch pods

are deployed so that all of the applications can share files. See Shared File Storage Requirements for more information. The NFS does not need to have its own subnet but it can.

The rest of the components in the diagram are automatically provisioned, depending on your specifications, when the GKE cluster and node pools are created. The gcloud scripts can be used to create a NAT gateway and subnet for outbound internet access, such as for pulling container images from the Cambridge Semantics repository. In addition, the scripts create a subnet for the K8s services and node pools and configure the routing so that Anzo can communicate with the K8s services and the services can talk to the pods that are deployed in the node pools.

Tip

When considering the network requirements of your organization and planning how to integrate the new K8s infrastructure in accordance with those requirements, it may help to consider the following types of use cases. Cambridge Semantics supplies sample cluster configuration files in the `gcloud/sample_use_cases` directory that are tailored for each of these use cases:

- **Deploy a private GKE cluster in an existing network (i.e., the same network as Anzo)**

  In this use case, the GKE cluster is deployed in a private subnet in your existing network. And a new (or existing, if you have one) NAT gateway is used to enable outbound access to services that are outside of the network. The control plane (master) is configured to allow access only from certain CIDRs.

- **Deploy a public GKE cluster in a new network**

  In this use case, a new network is created with the specified CIDR. A new NAT gateway is deployed to provide outbound connectivity for the cluster nodes. Public and private subnets are also created, and public access is restricted to specific IP ranges. The new network will need to be configured so that it is routable from Anzo.

- **Deploy a private GKE cluster with master authorized networks**

  In this use case (like the first case listed above), a private GKE cluster is deployed in an existing network. Master authorized network IP ranges are specified to limit the access to the public endpoint.

- **Deploy a private GKE cluster with public endpoint access enabled**

  In this use case, a private GKE cluster is deployed but public endpoint access is enabled and not restricted to specific IP ranges.

For a summary of the files in the gcloud directory, see Download the Cluster Creation Scripts and Configuration Files. Specifics about the parameters in the sample files are included in Creating the GKE Cluster.

To get started on creating the GKE infrastructure, see Creating and Assigning IAM Roles for instructions on creating the IAM roles that are needed for assigning permissions to create and use the GKE cluster.

Creating and Assigning IAM Roles

There are two custom Identity and Access Management (IAM) roles that need to be created in Google Cloud to grant the necessary permissions to the following two types of GKE users:

1. The first type of user is the user who sets up the K8s infrastructure, i.e., the user who configures, creates, and maintains the GKE cluster and node pools. This user role is called the **GKE Cluster Admin**.
2. The second type of user is the user who connects to the GKE cluster and deploys the dynamic Anzo applications. Typically this user is Anzo. Since Anzo communicates to the K8s services that provision the applications, the Anzo service account needs to be granted certain privileges. This user role is called the **GKE Cluster Developer**.

> **Note**
> The enterprise-level Anzo service account is a requirement for the Anzo installation and is typically in place before Anzo is installed. For more information, see Service User Account Requirements.

This topic provides instructions for creating the two roles and gives guidance on assigning the roles to the appropriate members or service accounts.

- Create and Assign the GKE Cluster Admin Role
- Create and Assign the GKE Cluster Developer Role

Create and Assign the GKE Cluster Admin Role

To ensure that the GKE cluster creator has all of the permissions needed for creating and managing K8s resources, there are four predefined Google roles in addition to the GKE Cluster Admin custom role that must be applied to the member or service account that will be used when creating the K8s infrastructure. Follow the instructions below to create the custom role and assign all necessary roles to the appropriate member or service account.

> **Note**
> Google Cloud IAM administrator privileges are required to create and assign IAM roles. The steps below give instructions for creating the custom GKE Cluster Admin role from the workstation. For more information about creating roles, including instructions on creating roles from the Cloud Console, see Creating and Managing Custom Roles in the Google Cloud documentation.

1. Create a JSON file on your workstation and copy the following contents to the file. For example, `vi /tmp/gke-cluster-admin.json`. The contents apply the minimum permissions needed for the GKE Cluster Admin.

```
{
  "name": "customClusterAdminRole",
  "title": "Custom Role for GKE Cluster Admin",
  "includedPermissions": [
```

```
      "compute.addresses.create",
      "compute.addresses.delete",
      "compute.addresses.get",
      "compute.addresses.use",
      "compute.firewallPolicies.get",
      "compute.firewalls.get",
      "compute.instanceGroups.get",
      "compute.instanceGroups.list",
      "compute.instances.get",
      "compute.instances.list",
      "compute.networks.create",
      "compute.networks.delete",
      "compute.networks.get",
      "compute.networks.listPeeringRoutes",
      "compute.networks.updatePolicy",
      "compute.networks.use",
      "compute.nodeGroups.get",
      "compute.regionOperations.get",
      "compute.regionOperations.list",
      "compute.regions.get",
      "compute.routers.create",
      "compute.routers.delete",
      "compute.routers.get",
      "compute.routers.update",
      "compute.routers.use",
      "compute.subnetworks.create",
      "compute.subnetworks.delete",
      "compute.subnetworks.get",
      "compute.subnetworks.use",
      "compute.vpnTunnels.get",
      "container.clusters.create",
      "container.clusters.delete",
      "container.clusters.update",
      "container.daemonSets.create",
      "container.daemonSets.delete",
      "container.daemonSets.get",
      "container.daemonSets.getStatus",
      "container.daemonSets.list",
      "container.nodes.list",
      "container.operations.get",
      "container.operations.list",
      "container.podSecurityPolicies.create",
```

```
      "container.podSecurityPolicies.delete",
      "container.podSecurityPolicies.get",
      "container.podSecurityPolicies.list",
      "container.podSecurityPolicies.update",
      "container.roleBindings.create",
      "container.roleBindings.delete",
      "container.roleBindings.get",
      "container.roles.bind",
      "container.roles.create",
      "container.roles.delete",
      "container.roles.get",
      "container.serviceAccounts.create",
      "container.serviceAccounts.delete",
      "container.serviceAccounts.get"
      ],
        "stage": "GA"
}
```

2. Once the file is created, run the following command to create the GKE Cluster Admin role, named **customClusterAdminRole**:

```
gcloud iam roles create <role_name> --project <project_name> --
file=/<path>/<file_name>.json
```

Where <project_name> is the project ID that the GKE cluster will be deployed in. For example:

```
gcloud iam roles create customClusterAdminRole --project cloud-project-1592 --
file=/tmp/gke-cluster-admin.json
```

3. Next, grant the new **customClusterAdminRole** and the following four predefined Compute Engine, Kubernetes Engine, Service Account, and Logging roles to the member or service account that will be used to create the GKE cluster:

- **roles/compute.networkViewer**
- **roles/container.clusterViewer**
- **roles/iam.serviceAccountUser**
- **roles/logging.viewer**

For information about granting roles to a member, see Granting, changing, and revoking access to resources. For information about applying a role to a service account, see Creating and managing service accounts. And for details about the predefined roles, see Predefined Roles in the Google Cloud documentation.

Create and Assign the GKE Cluster Developer Role

The following IAM role applies the minimum permissions needed for the GKE Cluster Developer role. Follow the instructions below to create the role and assign it to the Anzo service account.

> **Note**
> Google Cloud IAM administrator privileges are required to create and assign IAM roles. The steps below give instructions for creating the custom GKE Cluster Developer role from the workstation. For more information about creating roles, including instructions on creating roles from the Cloud Console, see Creating and Managing Custom Roles in the Google Cloud documentation.

1. Create a JSON file on your workstation and copy the following contents to the file. For example, `vi /tmp/gke-cluster-developer.json`.

```
{
  "name": "customClusterDevAnzoRole",
  "title": "Custom Role with Additional permissions required to deploy
resources through Anzo",
  "includedPermissions": [
      "compute.machineTypes.list",
      "storage.buckets.get",
      "storage.buckets.list"
      ],
        "stage": "GA"
}
```

2. Once the file is created, run the following command to create the GKE Cluster Developer role, named **customClusterDevAnzoRole**:

```
gcloud iam roles create <role_name> --project <project_name> --
file=/<path>/<file_name>.json
```

Where <role_ID> is the unique ID to use for the role and <project_name> is the project ID that the GKE cluster will be deployed in. For example:

```
gcloud iam roles create customClusterDevAnzoRole --project cloud-project-1592 --file=/tmp/gke-cluster-developer.json
```

3. Next, grant the new **customClusterDevAnzoRole** and the following three predefined Kubernetes Engine Developer, Kubernetes Engine Service Agent, and Storage Object Viewer roles to the Anzo service account:
   - **roles/container.developer**
   - **roles/container.serviceAgent**
   - **roles/storage.objectViewer**

   For information about applying a role to a service account, see Creating and managing service accounts in the Google Cloud documentation. For details about the predefined roles, see Predefined Roles in the Google Cloud documentation.

Once the IAM roles are in place and users are granted access, proceed to Creating the GKE Cluster for instructions on configuring and creating the cluster.

Creating the GKE Cluster

Follow the instructions below to define the GKE cluster resource requirements and then create the cluster based on your specifications.

- Define the GKE Cluster Requirements
- Example Configuration File
- Create the GKE Cluster

Define the GKE Cluster Requirements

The first step in creating the K8s cluster is to define the infrastructure specifications. The configuration file to use for defining the specifications is called **k8s_cluster.conf**. Multiple sample k8s_cluster.conf files are included in the **gcloud** directory. Any of them can be copied and used as templates, or the files can be edited directly.

Sample k8s_cluster.conf Files

To help guide you in choosing the appropriate template for your use case, this section describes each of the sample files. Details about the parameters in the sample files are included in Cluster Parameters below.

gcloud/conf.d/k8s_cluster.conf

This file is a non-specific use case. It includes sample values for all of the available cluster parameters.

gcloud/sample_use_cases/1_usePrivateEndpoint_private_cluster/k8s_cluster-.conf

This file includes sample values for a use case where:

- The GKE cluster will be deployed in a new private subnet in an existing network. You specify the existing network name in the `GCLOUD_NETWORK` parameter.
- A NAT gateway is deployed with a private endpoint (`GKE_ENABLE_PRIVATE_ENDPOINT=true`, `GKE_ENABLE_PRIVATE_ENDPOINT=true`, `GKE_PRIVATE_ACCESS=true`). There is no client access to the public endpoint.
- Secondary IP ranges are added to the NAT mapping along with the primary IP when `NETWORK_NAT_ALLOW_SUBNET_SECONDARY_IPS=true`. Outbound connectivity is allowed through the NAT gateway but restricted to the IP ranges specified in the `GKE_MASTER_ACCESS_CIDRS` parameter.

gcloud/sample_use_cases/2_public_cluster/k8s_cluster.conf

This file includes sample values for a use case where:

- A new network with public and private subnetworks will be created and the GKE cluster will be deployed into it.
- The cluster is public (`GKE_PRIVATE_ACCESS=false`).

gcloud/sample_use_cases/3_useAuthorizedNetworks/k8s_cluster.conf

This file includes sample values for a use case where:

- The GKE cluster will be deployed in a new or existing network with public and private subnets.
- The `GKE_MASTER_ACCESS_CIDRS` parameter is used to limit the access to the public endpoint.

gcloud/sample_use_cases/4_providePublicEndpointAccess/k8s_cluster.conf

This file includes sample values for a use case where:

- The GKE cluster will be deployed as a private cluster with public endpoint access enabled (`GKE_ENABLE_PRIVATE_ENDPOINT=false`).

Cluster Parameters

The contents of `k8s_cluster.conf` are shown below. Descriptions of the cluster parameters follow the contents.

```
            NETWORK_BGP_ROUTING="<bgp-routing-mode>"
NETWORK_SUBNET_MODE="<subnet-mode>"
NETWORK_ROUTER_NAME="<router>"
NETWORK_ROUTER_MODE="<advertisement-mode>"
NETWORK_ROUTER_ASN=<asn>
NETWORK_ROUTER_DESC="<description>"
NETWORK_NAT_NAME="<nat-name>"
NETWORK_NAT_UDP_IDLE_TIMEOUT="<udp-idle-timeout>"
NETWORK_NAT_ICMP_IDLE_TIMEOUT="<icmp-idle-timeout>"
NETWORK_NAT_TCP_ESTABLISHED_IDLE_TIMEOUT="<tcp-established-idle-timeout>"
NETWORK_NAT_TCP_TRANSITORY_IDLE_TIMEOUT="<tcp-transitory-idle-timeout>"
NETWORK_NAT_ALLOW_SUBNET_SECONDARY_IPS=<allow-subnet-secondary-ips>
K8S_CLUSTER_NAME=${K8S_CLUSTER_NAME:-"<cluster-name>"}
K8S_CLUSTER_PODS_PER_NODE="<default-max-pods-per-node>"
K8S_CLUSTER_ADDONS="<addons>"
GKE_MASTER_VERSION="<cluster-version>"
GKE_PRIVATE_ACCESS=<enable-private-nodes>
GKE_MASTER_NODE_COUNT_PER_LOCATION=<num-nodes>
GKE_NODE_VERSION="<node-version>"
GKE_IMAGE_TYPE="<image-type>"
GKE_MAINTENANCE_WINDOW='<maintenance-window>'
GKE_ENABLE_PRIVATE_ENDPOINT=<enable-private-endpoint>
```

```
GKE_MASTER_ACCESS_CIDRS="<master-authorized-networks>"
K8S_PRIVATE_CIDR="<cluster-ipv4-cidr>"
K8S_SERVICES_CIDR="<services-ipv4-cidr>"
GCLOUD_NODES_CIDR="<create-subnetwork>"
K8S_API_CIDR="<master-ipv4-cidr>"
K8S_HOST_DISK_SIZE='<disk-size>'
K8S_HOST_DISK_TYPE="<disk-type>"
K8S_HOST_MIN_CPU_PLATFORM="<min-cpu-platform>"
K8S_POOL_HOSTS_MAX=<max-nodes-per-pool>
K8S_METADATA="<metadata>"
K8S_MIN_NODES=<min-nodes>
K8S_MAX_NODES=<max-nodes>
GCLOUD_RESOURCE_LABELS='<labels>'
GCLOUD_VM_LABELS=<node-labels>
GCLOUD_VM_TAGS="<tags>"
GCLOUD_VM_MACHINE_TYPE="<machine-type>"
GCLOUD_VM_SSD_COUNT=<local-ssd-count>
GCLOUD_PROJECT_ID=${GCLOUD_PROJECT_ID:-"<project>"}
GCLOUD_NETWORK=${GCLOUD_NETWORK:-"<network>"}
GCLOUD_NODES_SUBNET_SUFFIX="<suffix>"
GCLOUD_CLUSTER_REGION=${GCLOUD_CLUSTER_REGION:-"<region>"}
GCLOUD_NODE_LOCATIONS="<node-locations>"
GCLOUD_NODE_TAINTS='<node-taints>'
GCLOUD_NODE_SCOPE='<scopes>'
```

Parameter	Description
NETWORK_BGP_ROUTING	The mode the Cloud Router will use to advertise BGP routes when the network is created, i.e, whether the cluster is global or regional. This parameter maps to the gcloud Cloud Router `--bgp-routing-mode` option. The default value is **regional**.
NETWORK_SUBNET_MODE	The method to use when subnets are created. Valid values are "auto" or "custom." This parameter maps to the gcloud VPC `--subnet-mode` option. The recommended value is **custom**.
NETWORK_ROUTER_NAME	The name to assign to the Cloud Router. For example, **csi-cloudrouter**.

Parameter	Description
NETWORK_ ROUTER_MODE	The route advertisement mode for the Cloud Router. This parameter maps to the gcloud Cloud Router `--advertisement-mode` option. The recommended value is **custom**.
NETWORK_ ROUTER_ASN	The Border Gateway Protocol (BGP) autonomous system number (ASN). When a router is created, it is assigned an ASN. This parameter maps to the gcloud Cloud Router `--asn` option. Coordinate with your network administrator to determine the number to specify.
NETWORK_ ROUTER_DESC	A description of the Cloud Router. This parameter maps to the gcloud Cloud Router `--description` option. For example, **Cloud router for K8S NAT**.
NETWORK_NAT_ NAME	The name to assign to the NAT gateway. For example, **csi-natgw**.
NETWORK_NAT_ UDP_IDLE_ TIMEOUT	The timeout value for UDP connections to the NAT gateway. This parameter maps to the gcloud NAT router `--udp-idle-timeout` option. The default value in k8s_cluster.conf is **60s** (60 seconds). For information about duration formats, refer to gcloud topic datetimes in the Cloud SDK documentation.
NETWORK_NAT_ ICMP_IDLE_ TIMEOUT	The timeout value for ICMP connections to the NAT gateway. This parameter maps to the gcloud NAT router `--icmp-idle-timeout` option. The default value in k8s_cluster.conf is **60s** (60 seconds).
NETWORK_NAT_ TCP_ ESTABLISHED_ IDLE_TIMEOUT	The timeout value for TCP established connections to the NAT gateway. This parameter maps to the gcloud NAT router `--tcp-established-idle-timeout` option. The default value in k8s_cluster.conf is **60s** (60 seconds).
NETWORK_NAT_	The timeout value to use for TCP transitory connections to the NAT

Parameter	Description
TCP_ TRANSITORY_ IDLE_TIMEOUT	gateway. This parameter maps to the gcloud NAT router `--tcp-transitory-idle-timeout` option. The default value in k8s_ cluster.conf is **60s** (60 seconds).
NETWORK_NAT_ ALLOW_SUBNET_ SECONDARY_IPS	Indicates whether to allow all secondary IP ranges for the GKE cluster to use the NAT gateway. If **true**, the secondary IP ranges for the subnets will have NAT gateway access.
K8S_CLUSTER_ NAME	The name to give to the cluster. For example, **csi-k8s-cluster**.
K8S_CLUSTER_ PODS_PER_NODE	The maximum number of pods that can be hosted on each compute instance. This parameter maps to the gcloud container cluster `--default-max-pods-per-node` option. This value also applies to the node pools in the cluster if the node pool configuration does not specify the maximum number of pods per node. Cambridge Semantics recommends that you set this value to **16**.
K8S_CLUSTER_ ADDONS	A comma-separated list of any additional Kubernetes cluster components to enable for the cluster. This parameter maps to the gcloud container cluster `--addons` option. By default, the k8s_cluster.conf file lists **HttpLoadBalancing** and **HorizontalPodAutoscaling**. Cambridge Semantics recommends that you include both of these components as a best practice.
GKE_MASTER_ VERSION	The Kubernetes version to use for the GKE cluster. This parameter maps to the gcloud container cluster `--cluster-version` option.
GKE_PRIVATE_ ACCESS	Indicates whether the cluster's nodes should have external IP addresses. When GKE_PRIVATE_ACCESS=true, the cluster remains private and nodes are not assigned external IP addresses. This parameter maps to the GKE `--enable-private-nodes` option.

Parameter	Description
GKE_MASTER_ NODE_COUNT_ PER_LOCATION	The number of nodes to create for running the K8s services in the default node pool in each of the cluster's zones. This value must be at least **1**. For high availability, Cambridge Semantics recommends setting this value to **3**. This parameter maps to the gcloud container cluster `--num-nodes` option.
GKE_NODE_ VERSION	The Kubernetes version to use for nodes in the node pools. This parameter maps to the gcloud container cluster `--node-version` option. Cambridge Semantics recommends that you specify the same version as the GKE_MASTER_VERSION.
GKE_IMAGE_TYPE	The base operating system that the nodes in the cluster will run on. This parameter maps to the gcloud container cluster `--image-type` option. This value must be **COS**.
GKE_ MAINTENANCE_ WINDOW	The time of day to start maintenance on this cluster. This parameter maps to the gcloud container cluster `--maintenance-window` option. The time corresponds to the UTC time zone and must be in HH:MM format. The default value in k8s_cluster.conf is **06:00** (6:00 am).
GKE_ENABLE_ PRIVATE_ ENDPOINT	Indicates whether to use a private or public IP address for the master API endpoint. When GKE_ENABLE_PRIVATE_ENDPOINT=true, the IP address for the API endpoint is private. This parameter maps to the GKE `--enable-private-endpoint` option.
GKE_MASTER_ ACCESS_CIDRS	The list of CIDR blocks (up to 50) that are allowed to connect to the GKE cluster over HTTPS. This value should include the Anzo subnet CIDR so that Anzo has access to the GKE cluster. This parameter maps to the gcloud container cluster `--master-authorized-networks` option. For example, **10.128.0.0/9**.
K8S_PRIVATE_	The IP address range (in CIDR notation) for the pods in this cluster. This

Parameter	Description
CIDR	parameter maps to the gcloud container cluster `--cluster-ipv4-cidr` option. For example, **172.16.0.0/20**.
K8S_SERVICES_CIDR	The IP address range for the cluster services. This parameter maps to the gcloud container cluster `--services-ipv4-cidr` option. For example: **172.17.0.0/20**.
GCLOUD_NODES_CIDR	The CIDR for the new subnet that will be created for the K8s cluster. This parameter maps to the `--create-subnetwork` option For example, **192.168.0.0/20**.
K8S_API_CIDR	The IPv4 CIDR range to use for the master network. The range should have a subnet mask of **/28**. This parameter maps to the gcloud container cluster `--master-ipv4-cidr` option. For example, **192.171.0.0/28**.
K8S_HOST_DISK_SIZE	The size of the boot disks on the cluster compute instances. This parameter maps to the gcloud container cluster `--disk-size` option. For example, **50GB**.
K8S_HOST_DISK_TYPE	The type of boot disk to use. This parameter maps to the gcloud container cluster `--disk-type` option. For example, **pd-standard**.
K8S_HOST_MIN_CPU_PLATFORM	The minimum CPU platform to use. This parameter maps to the gcloud container cluster `--min-cpu-platform` option. This value is left blank in the k8s_cluster.conf file.
K8S_POOL_HOSTS_MAX	The maximum number of nodes to allocate for the default initial node pool. This parameter maps to the gcloud container cluster `--max-nodes-per-pool` option. The default value is **1000**, but it can be set as low as 100 for the initial creation.
K8S_METADATA	The compute engine metadata (in the format key=val,key=val) to make

Parameter	Description
	available to the guest operating system running on nodes in the node pools. This parameter maps to the gcloud container cluster `--metadata` option. **Important** Including **disable-legacy-endpoints=true** is required to ensure that legacy metadata APIs are disabled. For more information about the option, see Protecting Cluster Metadata in the GKE documentation.
K8S_MIN_NODES	The minimum number of nodes in the default node pool. This parameter maps to the gcloud container cluster `--min-nodes` option. For example, **1**.
K8S_MAX_NODES	The maximum number of nodes in the default node pool. This parameter maps to the gcloud container cluster `--max-nodes` option. For example, **3**.
GCLOUD_ RESOURCE_ LABELS	A comma-separated list of any labels that you want to apply to the Google Cloud resources in use by the GKE cluster (unrelated to Kubernetes labels).
GCLOUD_VM_ LABELS	A comma-separated list of any Kubernetes labels to apply to nodes in the default node pool. This parameter maps to the gcloud container cluster `--node-labels` option.
GCLOUD_VM_ TAGS	A comma-separated list of strings to add to the instances in the cluster to classify the VMs. This parameter maps to the gcloud container cluster `--tags` option.
GCLOUD_VM_	The machine type to use for the GKE cluster nodes. This parameter maps

Parameter	Description
MACHINE_TYPE	to the gcloud container cluster `--machine-type` option. For example, **n1-standard-1**.
GCLOUD_VM_SSD_COUNT	The number of local SSD disks to add to each node. This parameter maps to the gcloud container cluster `--local-ssd-count` option. For example, specify **0** if you do not want to add SSDs to the nodes.
GCLOUD_PROJECT_ID	The Project ID for the GKE cluster. This parameter maps to the gcloud-wide `--project` option. For example, **cloud-project-1592**.
GCLOUD_NETWORK	The network to provision the GKE cluster in. This value should match the name of the network that Anzo is deployed in. This parameter maps to the gcloud container cluster `--network` option. For example, **devel-network**. **Note** If you want gcloud to create a new network, you can leave this value blank. However, after deploying the GKE cluster, you must configure the new network so that it is routable from the Anzo network.
GCLOUD_NODES_SUBNET_SUFFIX	The suffix to add to the subnetworks. For example, **nodes**.
GCLOUD_CLUSTER_REGION	The compute region for the GKE cluster. This value should match the name of the region that Anzo is deployed in. This parameter maps to the gcloud container cluster `--region` option. For example, **us-central1**.
GCLOUD_NODE_LOCATIONS	A comma-separated list of any zones to replicate the nodes in. This parameter maps to the gcloud container cluster `--node-locations` option. For example, **us-central1-f**.

Parameter	Description
GCLOUD_NODE_TAINTS	A comma-separated list of the Kubernetes taints for the nodes in the default node pool. When a pod is scheduled for deployment, the scheduler relies on this information to find the node pool that the pod belongs in. A pod has a **toleration** that identifies whether it is compatible with a node taint. This parameter maps to the gcloud container cluster `--node-taints` option. For more information, see Controlling Scheduling with Node Taints in the GKE documentation.
GCLOUD_NODE_SCOPE	A comma-separated list of the access scopes the nodes should have. This parameter maps to the gcloud container cluster `--scopes` option. For example, **gke-default**.

Example Configuration File

An example completed k8s_cluster.conf file is shown below.

```
NETWORK_BGP_ROUTING="regional"
NETWORK_SUBNET_MODE="custom"
NETWORK_ROUTER_NAME="csi-cloudrouter"
NETWORK_ROUTER_MODE="custom"
NETWORK_ROUTER_ASN=64512
NETWORK_ROUTER_DESC="Cloud router for K8S NAT."
NETWORK_NAT_NAME="csi-natgw"
NETWORK_NAT_UDP_IDLE_TIMEOUT="60s"
NETWORK_NAT_ICMP_IDLE_TIMEOUT="60s"
NETWORK_NAT_TCP_ESTABLISHED_IDLE_TIMEOUT="60s"
NETWORK_NAT_TCP_TRANSITORY_IDLE_TIMEOUT="60s"
NETWORK_NAT_ALLOW_SUBNET_SECONDARY_IPS=false
K8S_CLUSTER_NAME=${K8S_CLUSTER_NAME:-"csi-k8s-cluster"}
K8S_CLUSTER_PODS_PER_NODE="16"
K8S_CLUSTER_ADDONS="HttpLoadBalancing,HorizontalPodAutoscaling"
GKE_MASTER_VERSION="1.19.9-gke.1900"
GKE_PRIVATE_ACCESS=true
GKE_MASTER_NODE_COUNT_PER_LOCATION=1
GKE_NODE_VERSION="1.19.9-gke.1900"
GKE_IMAGE_TYPE="COS"
GKE_MAINTENANCE_WINDOW='06:00'
GKE_ENABLE_PRIVATE_ENDPOINT=true
```

```
GKE_MASTER_ACCESS_CIDRS="10.128.0.0/9"
K8S_PRIVATE_CIDR="172.16.0.0/20"
K8S_SERVICES_CIDR="172.17.0.0/20"
GCLOUD_NODES_CIDR="192.168.0.0/20"
K8S_API_CIDR="192.171.0.0/28"
K8S_HOST_DISK_SIZE='50GB'
K8S_HOST_DISK_TYPE="pd-standard"
K8S_HOST_MIN_CPU_PLATFORM=""
K8S_POOL_HOSTS_MAX=1000
K8S_METADATA="disable-legacy-endpoints=true"
K8S_MIN_NODES=1
K8S_MAX_NODES=3
GCLOUD_RESOURCE_LABELS='deleteafter=false,owner=user'
GCLOUD_VM_LABELS=description=k8s_cluster
GCLOUD_VM_TAGS="cluster-vm"
GCLOUD_VM_MACHINE_TYPE="n1-standard-1"
GCLOUD_VM_SSD_COUNT=0
GCLOUD_PROJECT_ID=${GCLOUD_PROJECT_ID:-"cloud-project-1592"}
GCLOUD_NETWORK=${GCLOUD_NETWORK:-"devel-network"}
GCLOUD_NODES_SUBNET_SUFFIX="nodes"
GCLOUD_CLUSTER_REGION=${GCLOUD_CLUSTER_REGION:-"us-central1"}
GCLOUD_NODE_LOCATIONS="us-central1-f"
GCLOUD_NODE_TAINTS='key1=val1:NoSchedule,key2=val2:PreferNoSchedule'
GCLOUD_NODE_SCOPE='gke-default'
```

Create the GKE Cluster

After defining the cluster requirements, run the **create_k8s.sh** script in the `gcloud` directory to create the cluster. Run the script with the following command. The arguments are described below.

```
./create_k8s.sh -c <config_file_name> [ -d <config_file_directory> ] [ -f | --force ] [ -h | --help ]
```

Argument	Description
-c <config_file_name>	This is a **required** argument that specifies the name of the configuration file that supplies the cluster requirements. For example, **-c k8s_cluster.conf**.
-d <config_file_directory>	This is an **optional** argument that specifies the path and directory name for the configuration file specified for the -c argument. If you are using the original `gcloud` directory file structure and the configuration file is in the `conf.d`

Argument	Description
	directory, you do not need to specify the -d argument. If you created a separate directory structure for different Anzo environments, include the -d option. For example, **-d /gcloud/env1/conf**.
-f | --force	This is an **optional** argument that controls whether the script prompts for confirmation before proceeding with each stage involved in creating the cluster. If **-f** (**--force**) is specified, the script assumes the answer is "yes" to all prompts and does not display them.
-h | --help	This argument is an **optional** flag that you can specify to display the help from the create_k8s.sh script.

For example, the following command runs the create_k8s script, using k8s_cluster.conf as input to the script. Since k8s_cluster.conf is in the conf.d directory, the -d argument is excluded:

```
./create_k8s.sh -c k8s_cluster.conf
```

The script validates that the required software packages, such as the gcloud sdk and kubectl, are installed and that the versions are compatible with the deployment. It also displays an overview of the deployment details based on the values in the specified configuration file. For example:

```
Operating System   : CentOS Linux
- Google Cloud SDK: 322.0.0
  alpha: 2021.01.05
  beta: 2021.01.05
  bq: 2.0.64
  core: 2021.01.05
  gsutil: 4.57
  kubectl cli version: Client Version: v1.19.12
  valid

Deployment details:
    Project            : cloud-project-1592
    Region             : us-central1
    GKE Cluster        : cloud-k8s-cluster
    GKE Master version : 1.19.9-gke.1900
```

The script then prompts you to proceed with deploying each component of the GKE cluster infrastructure. Type **y** and press **Enter** to proceed with creating the specified network, cluster, cloud router, and NAT gateway components. All components are created according to the specifications in the configuration file.

When cluster creation is complete, proceed to Creating the Required Node Pools to add the required node pools to the cluster.

Creating the Required Node Pools

This topic provides instructions for creating the three types of required node pools:

- The **Operator** node pool for running the AnzoGraph, Anzo Agent with Anzo Unstructured (AU), and Elasticsearch operator pods.
- The **AnzoGraph** node pool for running AnzoGraph application pods.
- The **Dynamic** node pool for running Anzo Agent with AU and Elasticsearch application pods.

> **Tip**
> For more information about the node pools, see Anzo Kubernetes Requirements.

- Define the Node Pool Requirements
- Example Configuration Files
- Create the Node Pools

Define the Node Pool Requirements

Before creating the node pools, configure the infrastructure requirements for each type of pool. The **nodepool_*.conf** files in the `gcloud/conf.d` directory are sample configuration files that you can use as templates, or you can edit the files directly:

- **nodepool_operator.conf** defines the requirements for the Operator node pool.
- **nodepool_anzograph.conf** defines the requirements for the AnzoGraph node pool.
- **nodepool_dynamic.conf** defines the requirements for the Dynamic node pool.

Important

The additional AnzoGraph and Dynamic node pool configuration files, **nodepool_anzograph_tuner.yaml** and **nodepool_dynamic_tuner.yaml**, configure the kernel-level tuning and security policies to apply to AnzoGraph and Dynamic runtime environments. Do not make changes to the files. There is a stage during node pool creation when the script prompts, **Do you want to tune the nodepools?**. It is important to answer **y** (yes) so that the kernel tuning and security policies are applied.

Each type of node pool configuration file contains the following parameters. Descriptions of the parameters and guidance on specifying the appropriate values for each type of node pool are provided below.

```
DOMAIN="<domain>"
KIND="<kind>"
GCLOUD_CLUSTER_REGION=${GCLOUD_CLUSTER_REGION:-"<region>"}
GCLOUD_NODE_TAINTS="<node-taints>"
GCLOUD_PROJECT_ID=${GCLOUD_PROJECT_ID:-"<project>"}
GKE_IMAGE_TYPE="<image-type>"
GKE_NODE_VERSION="<version>"
K8S_CLUSTER_NAME=${K8S_CLUSTER_NAME:-"<cluster>"}
NODE_LABELS="<node-labels>"
MACHINE_TYPES="<machine-type>"
TAGS="<tags>"
METADATA="<metadata>"
MAX_PODS_PER_NODE=<max-pods-per-node>
MAX_NODES=<max-nodes>
MIN_NODES=<min-nodes>
NUM_NODES=<num-nodes>
DISK_SIZE="<disk-size>"
DISK_TYPE="<disk-type>"
```

DOMAIN

The name of the domain that hosts the node pool. This is typically prefaced with the name of the organization.

Node Pool Type	Sample DOMAIN Value
Operator	csi-operator
AnzoGraph	csi-anzograph
Dynamic	csi-dynamic

KIND

This parameter classifies the node pool in terms of kernel tuning and the type of pods that the node pool will host.

Node Pool Type	Required KIND Value
Operator	operator
AnzoGraph	anzograph
Dynamic	dynamic

GCLOUD_CLUSTER_REGION

The compute region for the GKE cluster. This parameter maps to the gcloud container cluster `--region` option. For example, **us-central1**.

GCLOUD_NODE_TAINTS

This parameter configures a node so that the scheduler avoids or prevents using it for hosting certain pods. When a pod is scheduled for deployment, the scheduler relies on this value to determine whether the pod belongs in this pool. If a pod has a **toleration** that is not compatible with this **taint**, the pod is rejected from the pool. The table below lists the recommended values. The `NoSchedule` value means a toleration is required and pods without the appropriate toleration will not be allowed in the pool.

Node Pool Type	Recommended GCLOUD_NODE_TAINTS Value
Operator	cambridgesemantics.com/dedicated=operator:NoSchedule
AnzoGraph	cambridgesemantics.com/dedicated=anzograph:NoSchedule, cloud.google.com/gke-preemptible="false":PreferNoSchedule
Dynamic	cambridgesemantics.com/dedicated=dynamic:NoSchedule, cloud.google.com/gke-preemptible="true":NoSchedule

GCLOUD_PROJECT_ID

The Project ID for the node pool. This parameter maps to the gcloud-wide `--project` option. The value should match the Project ID for the GKE cluster. For example, **cloud-project-1592**.

GKE_IMAGE_TYPE

The base operating system that the nodes in the node pool will run on. This parameter maps to the gcloud container cluster `--image-type` option. This value must be **cos_containerd**.

GKE_NODE_VERSION

The Kubernetes version to use for nodes in the node pool. Cambridge Semantics recommends that you specify the same version as the GKE_MASTER_VERSION. This parameter maps to the gcloud container cluster `--node-version` option.

K8S_CLUSTER_NAME

The name of the GKE cluster to add the node pool to. For example, **csi-k8s-cluster**.

NODE_LABELS

A comma-separated list of key/value pairs that define the type of pods that can be placed on the nodes in this node pool. Labels are used to attract pods to nodes, while "taints" (GCLOUD_NODE_TAINTS) are used to repel other types of pods from being placed in this node pool. One

label, `cambridgesemantics.com/node-purpose`, is **required** for each type of node pool. The node-purpose label indicates that the purpose of the nodes in the pools are to host operator, anzograph, or dynamic pods. The table below lists the required labels for each node pool.

Node Pool Type	Required NODE_LABELS Value
Operator	cambridgesemantics.com/node-purpose=operator
AnzoGraph	cambridgesemantics.com/node-purpose=anzograph
Dynamic	cambridgesemantics.com/node-purpose=dynamic

MACHINE_TYPES

A space-separated list of the machine types that can be used for the nodes in this node pool. This parameter maps to the gcloud container cluster `--machine-type` option. If you list multiple machine types, the node pool creation script prompts you to create multiple node pools of the same KIND, one pool for each machine type.

Node Pool Type	Sample MACHINE_TYPES Value
Operator	n1-standard-1
AnzoGraph	n1-standard-16 n1-standard-32 n1-standard-64
Dynamic	n1-standard-4

Tip
For more guidance on determining the instance types to use for nodes in the required node pools, see Compute Resource Planning.

TAGS

A comma-separated list of strings to add to the instances in the node pool to classify the VMs. This parameter maps to the gcloud container cluster `--tags` option. For example, **csi-anzo**.

METADATA

The compute engine metadata (in the format key=val,key=val) to make available to the guest operating system running on nodes in the node pool. This parameter maps to the gcloud container cluster `--metadata` option.

> **Important**
> Including **disable-legacy-endpoints=true** is required to ensure that legacy metadata APIs are disabled. For more information about the option, see Protecting Cluster Metadata in the GKE documentation.

MAX_PODS_PER_NODE

The maximum number of pods that can be hosted on a node in this node pool. This parameter maps to the gcloud container cluster `--max-pods-per-node` option. In addition to Anzo application pods, this limit also needs to account for K8s service pods and helper pods. Cambridge Semantics recommends that you set this value to at least **16** for all node pool types.

MAX_NODES

The maximum number of nodes in the node pool. This parameter maps to the gcloud container cluster `--max-nodes` option.

Node Pool Type	Sample MAX_NODES Value
Operator	8
AnzoGraph	64
Dynamic	64

MIN_NODES

The minimum number of nodes in the node pool. This parameter maps to the gcloud container cluster `--min-nodes` option. If you set the minimum nodes to **0** for each node pool type, nodes will not be provisioned unless the relevant type of pod is scheduled for deployment.

NUM_NODES

The number of nodes to deploy when the node pool is created. This value must be set to at least **1**. When you create the node pool, at least one node in the pool needs to be deployed as well. However, if the GKE cluster autoscaler addon is enabled, the autoscaler will deprovision this node because it is not in use.

> **Note**
> Depending on the version of gcloud that you are using, you may be able to set NUM_NODES to **0**. Recent versions of gcloud added support for creating node pools without deploying any nodes.

DISK_SIZE

The size of the boot disks on the nodes. This parameter maps to the gcloud container cluster `--disk-size` option.

Node Pool Type	Sample DISK_SIZE Value
Operator	50GB
AnzoGraph	200GB
Dynamic	100GB

DISK_TYPE

The type of boot disk to use. This parameter maps to the gcloud container cluster `--disk-type` option.

Node Pool Type	Sample DISK_TYPE Value
Operator	pd-standard
AnzoGraph	pd-ssd
Dynamic	pd-ssd

Example Configuration Files

Example completed configuration files for each type of node pool are shown below.

Operator Node Pool

The example below shows a configured nodepool_operator.conf file.

```
DOMAIN="csi-operator"
KIND="operator"
GCLOUD_NODE_
TAINTS="cambridgesemantics.com/dedicated=operator:NoSchedule,cloud.google.com/gke-
preemptible="false":NoSchedule"
GCLOUD_CLUSTER_REGION=${GCLOUD_CLUSTER_REGION:-"us-central1"}
GKE_IMAGE_TYPE="cos_containerd"
GKE_NODE_VERSION="1.23.7-gke.1400"
GCLOUD_PROJECT_ID=${GCLOUD_PROJECT_ID:-"cloud-project-1592"}
K8S_CLUSTER_NAME=${K8S_CLUSTER_NAME:-"csi-k8s-cluster"}
NODE_LABELS="cambridgesemantics.com/node-
purpose=operator,cambridgesemantics.com/description=k8snode"
MACHINE_TYPES="n1-standard-1"
TAGS="csi-anzo"
METADATA="disable-legacy-endpoints=true"
MAX_PODS_PER_NODE=16
MAX_NODES=8
MIN_NODES=0
NUM_NODES=1
```

```
DISK_SIZE="50Gb"
DISK_TYPE="pd-standard"
```

AnzoGraph Node Pool

The example below shows a configured nodepool_anzograph.conf file.

```
DOMAIN="csi-anzograph"
KIND="anzograph"
GCLOUD_CLUSTER_REGION=${GCLOUD_CLUSTER_REGION:-"us-central1"}
GCLOUD_NODE_
TAINTS="cambridgesemantics.com/dedicated=anzograph:NoSchedule,cloud.google.com/gke-
preemptible="false":PreferNoSchedule"
GCLOUD_PROJECT_ID=${GCLOUD_PROJECT_ID:-"cloud-project-1592"}
GKE_IMAGE_TYPE="cos_containerd"
GKE_NODE_VERSION="1.23.7-gke.1400"
K8S_CLUSTER_NAME=${K8S_CLUSTER_NAME:-"csi-k8s-cluster"}
NODE_LABELS="cambridgesemantics.com/node-
purpose=anzograph,cambridgesemantics.com/description=k8snode"
MACHINE_TYPES="n1-standard-16 n1-standard-32 n1-standard-64"
TAGS="csi-anzo"
METADATA="disable-legacy-endpoints=true"
MAX_PODS_PER_NODE=16
MAX_NODES=64
MIN_NODES=0
NUM_NODES=1
DISK_SIZE="200Gb"
DISK_TYPE="pd-ssd"
```

Dynamic Node Pool

The example below shows a configured nodepool_dynamic.conf file.

```
DOMAIN="csi-dynamic"
KIND="dynamic"
GCLOUD_CLUSTER_REGION=${GCLOUD_CLUSTER_REGION:-"us-central1"}
GCLOUD_NODE_
TAINTS="cambridgesemantics.com/dedicated=dynamic:NoSchedule,cloud.google.com/gke-
preemptible="false":NoSchedule"
GCLOUD_PROJECT_ID=${GCLOUD_PROJECT_ID:-"cloud-project-1592"}
GKE_IMAGE_TYPE="cos_containerd"
GKE_NODE_VERSION="1.23.7-gke.1400"
K8S_CLUSTER_NAME=${K8S_CLUSTER_NAME:-"csi-k8s-cluster"}
```

```
NODE_LABELS="cambridgesemantics.com/node-
purpose=dynamic,cambridgesemantics.com/description=k8snode"
MACHINE_TYPES="n1-standard-4"
TAGS="csi-anzo"
METADATA="disable-legacy-endpoints=true"
MAX_PODS_PER_NODE=16
MAX_NODES=64
MIN_NODES=0
NUM_NODES=1
DISK_SIZE="100Gb"
DISK_TYPE="pd-ssd"
```

Create the Node Pools

After defining the requirements for the node pools, run the **create_nodepools.sh** script in the `gcloud` directory to create each type of node pool. Run the script with the following command. Run it once for each type of pool. The arguments are described below.

```
./create_nodepools.sh -c <config_file_name> [ -d <config_file_directory> ] [ -f | --
force ] [ -h | --help ]
```

Argument	Description
-c <config_ file_name>	This is a **required** argument that specifies the name of the configuration file (i.e., `nodepool_operator.conf`, `nodepool_anzograph.conf`, or `nodepool_ dynamic.conf`) that supplies the node pool requirements. For example, **-c nodepool_dynamic.conf**.
-d <config_ file_ directory>	This is an **optional** argument that specifies the path and directory name for the configuration file specified for the -c argument. If you are using the original `gcloud` directory file structure and the configuration file is in the `conf.d` directory, you do not need to specify the -d argument. If you created a separate directory structure for different Anzo environments, include the -d option. For example, **-d /gcloud/env1/conf**.
-f | --force	This is an **optional** argument that controls whether the script prompts for confirmation before proceeding with each stage involved in creating the node pool. If **-f** (**--force**) is specified, the script assumes the answer is "yes" to all

Argument	Description
	prompts and does not display them.
-h | --help	This argument is an **optional** flag that you can specify to display the help from the create_nodepools.sh script.

For example, the following command runs the create_nodepools script, using nodepool_operator.conf as input to the script. Since nodepool_operator.conf is in the conf.d directory, the -d argument is excluded:

```
./create_nodepools.sh -c nodepool_operator.conf
```

The script validates that the required software packages are installed and that the versions are compatible with the deployment. It also displays an overview of the deployment details based on the values in the specified configuration file. For example:

```
Operating System   : CentOS Linux
- Google Cloud SDK: 322.0.0
  alpha: 2021.01.05
  beta: 2021.01.05
  bq: 2.0.64
  core: 2021.01.05
  gsutil: 4.57
  kubectl cli version: Client Version: v1.23.9
  valid

Deployment details:
    Project            : cloud-project-1592
    Region             : us-central1
    GKE Cluster        : csi-k8s-cluster
```

The script then prompts you to proceed with deploying each component of the node pool. Type **y** and press **Enter** to proceed with the configuration.

Important

When creating the AnzoGraph and Dynamic node pools, there is a stage when the script prompts, **Do you want to tune the nodepools?**. It is important to answer **y** (yes) so that the kernel tuning and security policies from the related nodepool_*_tuner.yaml file are applied to the node pool configuration.

Once the Operator, AnzoGraph, and Dynamic node pools are created, the next step is to create a Cloud Location in Anzo so that Anzo can connect to the GKE cluster and deploy applications. See Connecting to a Cloud Location in the Administration Guide.

Azure Kubernetes Service Deployments

The topics in this section guide you through the process of deploying all of the Azure Kubernetes Service (AKS) infrastructure that is required to support dynamic deployments of Anzo components. The topics provide instructions for setting up a workstation to use for deploying the K8s infrastructure, performing the prerequisite tasks before deploying the AKS cluster, creating the AKS cluster, and creating the required node pools.

In this section:

Setting Up a Workstation

This topic provides the requirements and instructions to follow for configuring a workstation to use for creating and managing the AKS infrastructure. The workstation needs to be able to connect to the Azure API. It also needs to have the required Azure and Kubernetes (K8s) software packages as well as the deployment scripts and configuration files supplied by Cambridge Semantics. This workstation will be used to connect to the Azure API and provision the K8s cluster and node pools.

> **Note**
> You can use the Anzo server as the workstation if the network routing and security policies permit the Anzo server to access the Azure and K8s APIs. When deciding whether to use the Anzo server as the K8s workstation, consider whether Anzo may be migrated to a different server or VPC in the future.

- Review the Requirements and Install the Software
- Download the Cluster Creation Scripts and Configuration Files

Review the Requirements and Install the Software

Component	Requirement
Operating System	The operating system for the workstation must be **RHEL/CentOS 7.8 or higher**.
Networking	The workstation should be in the same VPC network as the AKS cluster. If it is not in the same VPC, make sure that it is on a network that is routable from the cluster's VPC.
Software	• **Python 3** is required. • **Kubectl**: Cambridge Semantics recommends that you use the same kubectl version as the AKS cluster version. For instructions, see Install Kubectl below. • **Azure CLI Version 2.5.1 or later** is required. For installation instructions, see Install Azure CLI below.
CSI AZ Package	Cambridge Semantics provides **az** scripts and configuration files to use for provisioning the AKS cluster and node pools. Download the files to the workstation. See Download the Cluster Creation Scripts and Configuration Files for more information about the az package.

Install Kubectl

Follow the instructions below to install kubectl on your workstation. Cambridge Semantics recommends that you install the same version of kubectl as the K8s cluster API. For more information, see Install and Set Up kubectl on Linux in the Kubernetes documentation.

1. Run the following cURL command to download the kubectl binary:

```
curl -LO https://dl.k8s.io/release/<version>/bin/linux/amd64/kubectl
```

Where <version> is the version of kubectl to install. For example, the following command downloads version 1.19.12:

```
curl -LO https://dl.k8s.io/release/v1.19.12/bin/linux/amd64/kubectl
```

2. Run the following command to make the binary executable:

```
chmod +x ./kubectl
```

3. Run the following command to move the binary to your PATH:

```
sudo mv ./kubectl /usr/local/bin/kubectl
```

4. To confirm that the binary is installed and that you can run kubectl commands, run the following command to display the client version:

```
kubectl version --client
```

The command returns the following type of information. For example:

```
Client Version: version.Info{Major:"1", Minor:"19", GitVersion:"v1.19.12",
GitCommit:"f3abc15296f3a3f54e4ee42e830c61047b13895f",
GitTreeState:"clean", BuildDate:"2021-06-16T13:21:12Z", GoVersion:"go1.13.15",
Compiler:"gc", Platform:"linux/amd64"}
```

Install Azure CLI

Follow the instructions below to install the Azure CLI on your workstation. These instructions follow the steps in Install the Azure CLI on Linux in the Microsoft Azure CLI documentation.

1. Run the following command to import the Microsoft repository key:

```
sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc
```

2. Run the following command to create the local azure-cli repository information:

```
echo -e "[azure-cli]
name=Azure CLI
baseurl=https://packages.microsoft.com/yumrepos/azure-cli
enabled=1
gpgcheck=1
gpgkey=https://packages.microsoft.com/keys/microsoft.asc" | sudo tee
/etc/yum.repos.d/azure-cli.repo
```

3. Run the following command to install the CLI:

```
sudo yum install azure-cli
```

4. To ensure that the CLI was installed, run the following command to display the CLI version:

```
az version
```

5. Next, run the following command to run the Azure CLI. Follow the prompts to log in to Azure:

```
az login --use-device-code
```

Download the Cluster Creation Scripts and Configuration Files

The Cambridge Semantics GitHub repository, k8s-genesis (https://github.com/cambridgesemantics/k8s-genesis.git), includes all of the files that are needed to manage the configuration, creation, and deletion of the AKS cluster and node pools.

You can clone the repository to any location on the workstation or download the k8s-genesis package as a ZIP file, copy the file to the workstation, and extract the contents. The k8s-genesis directory includes three subdirectories (one for each supported Cloud Service Provider), the license information, and a readme file:

```
k8s-genesis
├── aws
├── azure
├── gcp
├── LICENSE
└── README.md
```

Navigate to `/azure/k8s/az`. The **az** directory contains all of the AKS cluster and node pool configuration files. You can remove all other directories from the workstation. The az files and subdirectories are shown below:

```
az
├── common.sh
├── conf.d
│   ├── k8s_cluster.conf
│   ├── nodepool_anzograph.conf
│   ├── nodepool_common.conf
│   ├── nodepool.conf
```

```
│   ├── nodepool_dynamic.conf
│   └── nodepool_operator.conf
├── create_k8s.sh
├── create_nodepools.sh
├── delete_k8s.sh
├── delete_nodepools.sh
├── exec_samples
│   ├── rbac_aad_group.yaml
│   └── rbac_aad_user.yaml
├── permissions
│   ├── aks_admin_role.json
│   └── cluster_developer_role.json
├── README.md
├── reference
│   ├── nodepool_anzograph_tuner.yaml
│   └── nodepool_dynamic_tuner.yaml
└── sample_use_cases
    ├── 10_useExistingResources
    │   └── k8s_cluster.conf
    ├── 11_useProximityPlacementGroups
    │   └── k8s_cluster.conf
    ├── 1_azureManagedIdentity_private_cluster
    │   └── k8s_cluster.conf
    ├── 2_createServicePrincipal_public_cluster
    │   └── k8s_cluster.conf
    ├── 3_useServicePrincipal
    │   └── k8s_cluster.conf
    ├── 4_userManagedAAD
    │   └── k8s_cluster.conf
    ├── 5_azureManagedAAD
    │   └── k8s_cluster.conf
    ├── 6_attachACR
    │   └── k8s_cluster.conf
    ├── 7_clusterAutoscalerSupport
    │   └── k8s_cluster.conf
    ├── 8_MonitoringEnabled
    │   └── k8s_cluster.conf
    └── 9_RBACSupport
        └── k8s_cluster.conf
```

The following list gives an overview of the files. Subsequent topics describe the files in more detail.

- The **common.sh** script is used by the create and delete cluster and node pool scripts.
- The **conf.d** directory contains the configuration files that are used to supply the specifications to follow when creating the K8s cluster and node pools:
  - **k8s_cluster.conf**: Supplies the specifications for the AKS cluster.
  - **nodepool_anzograph.conf**: Supplies the specifications for the AnzoGraph node pool.
  - **nodepool_common.conf**: Supplies the specifications for a Common node pool. The Common node pool is not required for AKS deployments, and this configuration file is typically not used.
  - **nodepool.conf**: This file is supplied as a reference. It contains the super set of node pool parameters.
  - **nodepool_dynamic.conf**: Supplies the specifications for the Dynamic node pool.
  - **nodepool_operator.conf**: Supplies the specifications for the Operator node pool.
- The **create_k8s.sh** script is used to deploy the AKS cluster, and the **k8s_cluster.conf** file in the conf.d directory is the configuration file that is input to the create_k8s.sh script.
- The **create_nodepools.sh** script is used to deploy the required node pools in the AKS cluster. The **nodepool_*.conf** files in the **conf.d** directory are the configuration files that are input to the create_nodepools.sh script.
- The **delete_k8s.sh** script is used to delete the AKS cluster.
- The **delete_nodepools.sh** script is used to remove node pools from the AKS cluster.
- The **exec_samples** and **permissions** directories contain role definitions and scripts for creating the custom roles that are needed to grant access to the Azure users and groups who will create or use the AKS cluster.
- The **reference** directory contains crucial files that are referenced by the cluster and node pool creation scripts. The files in the directory should not be edited, and the **reference** directory must exist on the workstation at the same level as the **create*.sh** and **delete*.sh** scripts.
- The **sample_use_cases** directory contains sample AKS cluster configuration files that you can refer to or use as a template for configuring your AKS cluster depending on your use

case. There are several files in the directory because there is an example for each type of AKS-supported identity and authentication management option. You can use a combination of settings from different sample files to configure your cluster, but you can only choose one type of authentication mode. For example, you cannot enable Service Principals with Azure Active Directory.

- The **k8s_cluster.conf** file in the **1_azureManagedIdentity_private_cluster** directory is a sample file for a use case where you want to deploy the AKS cluster into a private Virtual Network and let Azure handle identity creation and management. Using an Azure managed identity is recommended.
- The **k8s_cluster.conf** file in the **2_createServicePrincipal_public_cluster** directory is a sample file for a use case where you want to create a new Service Principal to deploy a public AKS cluster. Access to the cluster is limited to certain IP ranges. Managing Service Principals adds more complexity than using an Azure managed identity.
- The **k8s_cluster.conf** file in the **3_useServicePrincipal** directory is a sample file for a use case that is similar to the **2_createServicePrincipal_public_cluster** use case above but uses an existing Service Principal.
- The **k8s_cluster.conf** file in the **4_userManagedAAD** directory is a sample file for a use case where you want to deploy an AKS cluster that connects to your user-managed Azure Active Directory (AAD) server for identity management. You supply the AAD client and server applications and the AAD tenant.
- The **k8s_cluster.conf** file in the **5_azureManagedAAD** directory is a sample file for a use case where you want to deploy an AKS cluster that connects to an Azure-managed Azure Active Directory (AAD) server for identity management. In this case, the AKS resource provider manages the client and server AAD applications.
- The **k8s_cluster.conf** file in the **6_attachACR** directory is a sample file for a use case where you want to deploy an AKS cluster that retrieves images from a private Azure Container Registry.
- The **k8s_cluster.conf** file in the **7_clusterAutoscalerSupport** directory is a sample file for a use case where you want to deploy an AKS cluster that employs the Cluster

Autoscaler service. The autoscaler automatically adds nodes to the node pool when demand increases and then deprovisions the nodes when demand decreases.

- The **k8s_cluster.conf** file in the **8_MonitoringEnabled** directory is a sample file for a use case where you want to deploy an AKS cluster with cluster monitoring enabled.
- The **k8s_cluster.conf** file in the **9_RBACSupport** directory is a sample file for a use case where you want to deploy an AKS cluster with Azure Role-Based Access Control (RBAC). Enabling RBAC allows you to use Azure AD users, groups, or service principals as subjects in Kubernetes RBAC.
- The **k8s_cluster.conf** file in the **10_useExistingResources** directory is a sample file for a use case where you want to deploy the AKS cluster into existing resources, such an existing Virtual Network with existing resource groups and subnetworks.
- The **k8s_cluster.conf** file in the **11_useProximityPlacementGroups** directory is a sample file for a use case where you want to use proximity placement groups for reduced latency. A proximity placement group is a logical grouping used to make sure Azure compute resources are physically located close to each other.

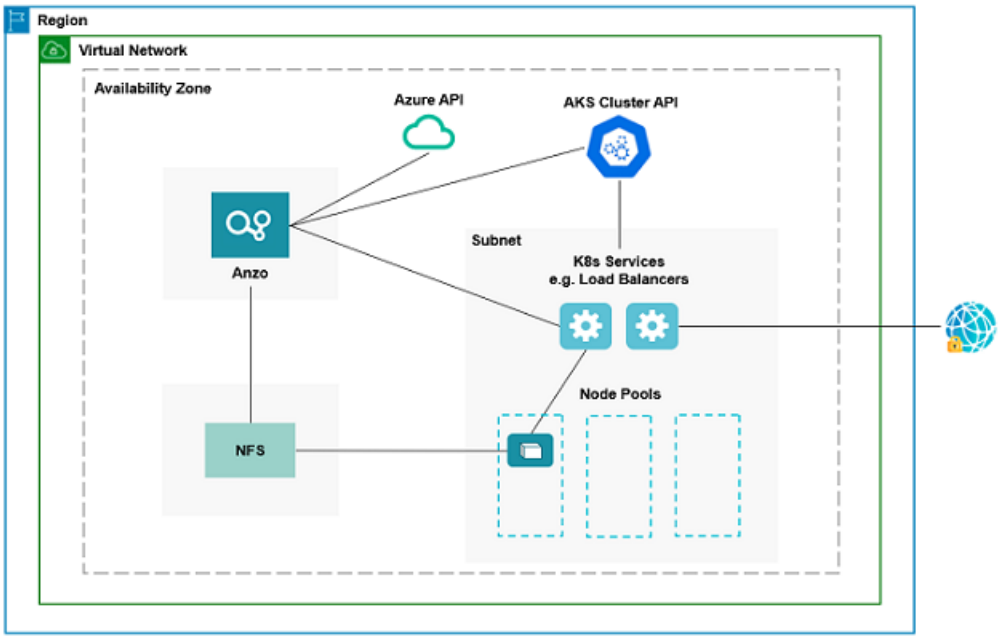
Once the workstation is configured, see Planning the Anzo and AKS Network Architecture to review information about the network architecture that the az scripts create. And see Creating and Assigning IAM Roles for instructions on creating the IAM roles that are needed for assigning permissions to create and use the AKS cluster.

Planning the Anzo and AKS Network Architecture

This topic describes the network architecture that supports the Anzo and AKS integration.

> **Note**
> When you deploy the K8s infrastructure, Cambridge Semantics strongly recommends that you create the AKS cluster in the same Virtual Network as Anzo. If you create the AKS cluster in a new Virtual Network, you must configure the new network to be routable from the Anzo Virtual Network.

The diagram below shows the typical network components that are employed when an AKS cluster is integrated with Anzo. Most of the network resources shown in the diagram are automatically deployed (and the appropriate routing is configured) according to the values that you supply in the cluster and node group .conf files in the **az** package on the workstation.



In the diagram, there are two components that you deploy before configuring and creating the K8s resources:

- **Anzo**: Since the Anzo server is typically deployed before the K8s components, you specify the Anzo network when creating the AKS cluster, ensuring that Anzo and all of the AKS cluster components are in the same network and can talk to each other. Also, make sure that Anzo has access to the Azure and AKS APIs.
- **NFS**: You are required to create a network file system (NFS). However, Anzo automatically mounts the NFS to the nodes when AnzoGraph, Anzo Unstructured, and Elasticsearch pods are deployed so that all of the applications can share files. See Shared File Storage Requirements for more information. The NFS does not need to have its own subnet but it can.

The rest of the components in the diagram are automatically provisioned, depending on your specifications, when the AKS cluster and node pools are created. The az scripts can be used to create a subnet for the K8s services and node pools and configure the routing so that Anzo can

communicate with the K8s services and the services can talk to the pods that are deployed in the node pools. In addition, a Standard Load Balancer can be used to provide outbound internet access, such as for pulling container images from the Cambridge Semantics repository.

Tip

When considering the network requirements of your organization and planning how to integrate the new K8s infrastructure in accordance with those requirements, it may help to consider the following types of use cases. Cambridge Semantics supplies sample cluster configuration files in the `az/sample_use_cases` directory that are tailored for each of these use cases:

- **Deploy a private AKS cluster with Azure Managed Identity**

  In this use case, the AKS cluster is deployed as a private cluster with no public access, and the Azure Managed Identity service is enabled for identity and authorization management. Using Azure Managed Identity is the recommended method to choose for AKS access control.

- **Deploy a public AKS cluster with a new Service Principal**

  In this use case, the AKS cluster is deployed as a public cluster, and a Service Principal is created for managing access control. You are responsible for maintaining the Service Principal to keep the cluster functional.

- **Deploy a public AKS cluster with an existing new Service Principal**

  This use case is similar to the use case described above but uses an existing Service Principal instead of creating a new one.

- **Deploy a private AKS cluster with a user-managed Azure Active Directory server**

  In this use case, the AKS cluster is deployed as a private cluster and a user-managed Azure Active Directory (AAD) server is used for identity and authorization management. In this case, you supply the AAD client and server applications and the AAD tenant.

- **Deploy a private AKS cluster with an Azure-managed AAD server**

  In this use case, the AKS cluster is deployed as a private cluster and an Azure-managed AAD server is used for identity and authorization management. In this case, the AKS resource manager manages the AAD client and server applications.

- **Deploy an AKS cluster and access a private Azure Container Registry**

  In this use case, an AKS cluster is deployed and accesses images that are maintained in a private Azure Container Registry.

- **Deploy an AKS cluster that Auto Scales on Demand**

  In this use case, an AKS cluster is deployed and the Cluster Autoscaler service is enabled. The Cluster Autoscaler automatically adds nodes to the node pool when demand increases and deprovisions nodes when demand decreases.

- **Deploy an AKS with the Monitoring Addon**

  In this use case, an AKS cluster is deployed and the Log Analytics monitoring service is enabled.

- **Deploy an AKS cluster with RBAC enabled**

  In this use case, an AKS cluster is deployed and Role-Based Access Control (RBAC) is enabled. RBAC manages Kubernetes user identities and credentials. RBAC can be enabled in conjunction with other authorization modes, such as Azure Managed Identity or AAD.

- **Deploy an AKS cluster using existing resources**

  In this use case, an AKS cluster is deployed without creating new network components. The cluster is deployed into an existing Virtual Network and uses existing resource groups and subnetworks.

- **Deploy an AKS cluster with Proximity Placement Groups**

  In this case, an AKS cluster is deployed with specified Proximity Placement Groups to ensure that compute resources are deployed physically close to each other to reduce latency.

For a summary of the files in the az directory, see Download the Cluster Creation Scripts and Configuration Files. Specifics about the parameters in the sample files are included in Creating the AKS Cluster.

To get started on creating the AKS infrastructure, see Creating and Assigning IAM Roles for instructions on creating the IAM roles that are needed for assigning permissions to create and use the AKS cluster.

Creating and Assigning IAM Roles

This topic provides instructions for creating the Identity and Access Management (IAM) roles that are needed to supply the necessary permissions for creating and managing the AKS cluster and using the cluster to deploy applications.

Note
AKS is typically configured to use Azure Active Directory (AD) for user authentication. AKS integration with Azure AD is optional but highly recommended. For more information, see Azure Active Directory Integration in the AKS documentation.

There are two custom roles that need to be created in Azure to grant the necessary permissions to the following two types of AKS users:

1. The first type of user is the user who sets up the K8s infrastructure, i.e., the user who configures, creates, and maintains the AKS cluster and node pools. This policy is called the **AKS Cluster Admin**.
2. The second type of user is the user who connects to the AKS cluster and deploys the dynamic Anzo applications. Typically this user is Anzo. Since Anzo communicates with the K8s services that provision the applications, the Anzo service principal needs to be granted certain

privileges. This user role is called the **AKS Cluster Developer**.

> **Note**
> The enterprise-level Anzo service principal is a requirement for the Anzo installation and is typically in place before Anzo is installed. For more information, see Service User Account Requirements.

This topic provides instructions for creating the two roles and gives guidance on assigning the roles to the appropriate users, groups, or service principals.

- Create and Assign the AKS Cluster Admin Role
- Create and Assign the AKS Cluster Developer Role

Create and Assign the AKS Cluster Admin Role

The following IAM role applies the minimum permissions needed for an AKS Cluster Admin who will create and manage the AKS cluster and node pools. Follow the instructions below to create the role and assign it to the user, group, or service principal that will be used when creating the K8s infrastructure.

> **Note**
> The **az** file package on the workstation includes the configuration file that defines the AKS Cluster Admin role: `az/permissions/aks_admin_role.json`.

1. Open the `az/permissions/aks_admin_role.json` file for editing. At the bottom of the file, replace `<subscription_id>` with the ID for the subscription to attach the new AKS Cluster Admin role to. Then save and close the file. The contents of `aks_admin_role.json` are shown below:

```
{
  "Name": "AKS Cluster Admin",
  "IsCustom": true,
  "Description": "AKS cluster admin role.",
  "Actions": [
      "Microsoft.Resources/subscriptions/resourcegroups/read",
```

```
      "Microsoft.Resources/subscriptions/resourcegroups/write",
      "Microsoft.Resources/subscriptions/resourcegroups/delete",
      "Microsoft.Network/virtualNetworks/read",
      "Microsoft.Network/virtualNetworks/write",
      "Microsoft.Network/virtualNetworks/delete",
      "Microsoft.Network/virtualNetworks/subnets/read",
      "Microsoft.Network/virtualNetworks/subnets/write",
      "Microsoft.Network/virtualNetworks/subnets/delete",
      "Microsoft.Network/virtualNetworks/subnets/join/action",
      "Microsoft.Network/publicIPPrefixes/read",
      "Microsoft.Network/publicIPPrefixes/write",
      "Microsoft.Network/publicIPPrefixes/delete",
      "Microsoft.Network/publicIPPrefixes/join/action",
      "Microsoft.Authorization/roleAssignments/read",
      "Microsoft.Authorization/roleAssignments/write",
      "Microsoft.Authorization/roleAssignments/delete",
      "Microsoft.Resources/deployments/write",
      "Microsoft.ContainerService/managedClusters/read",
      "Microsoft.ContainerService/managedClusters/write",
      "Microsoft.ContainerService/managedClusters/delete",
      "Microsoft.ContainerService/managedClusters/agentPools/read",
      "Microsoft.ContainerService/managedClusters/agentPools/write",
      "Microsoft.ContainerService/managedClusters/agentPools/delete",

"Microsoft.ContainerService/managedClusters/listClusterAdminCredential/action",
      "Microsoft.OperationsManagement/solutions/read",
      "Microsoft.OperationsManagement/solutions/write",
      "Microsoft.OperationalInsights/workspaces/read",
      "Microsoft.OperationalInsights/workspaces/sharedkeys/read",
      "Microsoft.ContainerRegistry/registries/read"
  ],
  "NotActions": [

  ],
  "AssignableScopes": [
    "/subscriptions/<subscription_id>"
  ]
}
```

2. Next, run the following Azure CLI command to create a custom role definition based on aks_admin_role.json. For information about managing role definitions, see az role definition in the Azure CLI documentation.

```
az role definition create --role-definition cluster-admin-role.json
```

3. Once the role is defined in Azure, run the following command to assign the role to the user, group, or service principal who will create and manage the AKS cluster. For information about managing role assignments, see az role assignment in the Azure CLI documentation.

```
az role assignment create --assignee "<user_group_or_sp_name_or_id>" --role
"<role_name_or_id>"
```

Create and Assign the AKS Cluster Developer Role

The following IAM role applies the minimum permissions needed for the AKS Cluster Developer role. Follow the instructions below to create the role and assign it to the Anzo service account.

> **Note**
> The **az** file package on the workstation includes the configuration file that defines the AKS Cluster Developer role: `az/permissions/cluster_developer_role.json`.

1. Open the `az/permissions/cluster_developer_role.json` file for editing. At the bottom of the file, replace `<subscription_id>` with the ID for the subscription to attach the new AKS Cluster Developer role to. Then save and close the file. The contents of `cluster_developer_role.json` are shown below:

```
{
  "Name": "AKS Cluster Developer",
  "IsCustom": true,
  "Description": "AKS cluster developer role.",
  "Actions": [

"Microsoft.ContainerService/managedClusters/listClusterUserCredential/action"
  ],
  "NotActions": [

  ],
  "AssignableScopes": [
    "/subscriptions/<subscription_id>"
  ]
}
```

2. Next, run the following Azure CLI command to create a custom role definition based on cluster_developer_role.json.

   ```
   az role definition create --role-definition cluster_developer_role.json
   ```

   For more information about managing role definitions in Azure, see az role definition in the Azure CLI documentation.

3. Once the role is defined in Azure, run the following command to assign the role to the Anzo service principal.

   ```
   az role assignment create --assignee "<anzo_sp>" --role "<role_name_or_id>"
   ```

   For more information about managing role assignments in Azure, see az role assignment in the Azure CLI documentation.

Once the IAM roles are in place and users are granted access, proceed to Creating the AKS Cluster for instructions on configuring and creating the cluster.

Creating the AKS Cluster

Follow the instructions below to define the AKS cluster resource requirements and then create the cluster based on your specifications.

- Define the AKS Cluster Requirements
- Example Configuration File
- Create the AKS Cluster

Define the AKS Cluster Requirements

The first step in creating the K8s cluster is to define the infrastructure specifications. The configuration file to use for defining the specifications is called **k8s_cluster.conf**. Multiple sample k8s_cluster.conf files are included in the **az** directory. Any of them can be copied and used as templates, or the files can be edited directly.

Sample k8s_cluster.conf Files

To help guide you in choosing the appropriate template for your use case, this section describes each of the sample files. Details about the parameters in the sample files are included in Cluster Parameters below.

> **Note**
> There are several sample use case files because there is an example for each type of AKS-supported identity and authentication management option. You can use a combination of settings from different sample files to configure your cluster, but you can only choose one type of authentication. For example, you cannot configure Service Principals and enable Azure Active Directory.

az/conf.d/k8s_cluster.conf

This file is a non-specific use case. It includes sample values for all of the available cluster parameters.

az/sample_use_cases/1_azureManagedIdentity_private_cluster/k8s_cluster.conf

This file includes sample values for a use case where:

- A private AKS cluster is deployed (`PRIVATE_CLUSTER="true"`) so that the cluster is only accessible from within the Virtual Network or a connected network.
- Azure Managed Identity is enabled (`ENABLE_MANAGED_IDENTITY="true"`) so that Azure manages identity creation and management. Using Azure Managed Identity is highly recommended.

az/sample_use_cases/2_createServicePrincipal_public_cluster/k8s_cluster.conf

This file includes sample values for a use case where:

- A public AKS cluster is deployed (`PRIVATE_CLUSTER="false"`).
- A Service Principal is created (`SP=${SP:-"<service-principal>"}`) that must be renewed and managed by you.
- Public access to the cluster can be limited to certain IP ranges by specifying the approved ranges in the `API_SERVER_AUTHORIZED_IP_RANGES` parameter.

az/sample_use_cases/3_useServicePrincipal/k8s_cluster.conf

This file includes sample values for a use case where:

- A public AKS cluster is deployed (`PRIVATE_CLUSTER="false"`).
- An existing Service Principal is used for identity and access management. The `SP_ID` and `SP_SECRET` parameters are used to specify the ID and secret for the existing Service Principal.
- Public access to the cluster can be limited to certain IP ranges by specifying the approved ranges in the `API_SERVER_AUTHORIZED_IP_RANGES` parameter.

az/sample_use_cases/4_userManagedAAD/k8s_cluster.conf

This file includes sample values for a use case where:

- A private AKS cluster is deployed (`PRIVATE_CLUSTER="true"`) so that the cluster is only accessible from within the Virtual Network or a connected network.
- An existing Azure Active Directory (AAD) server is used for identity and authorization management. Details about the existing AAD client and server applications as well as the tenet ID need to be specified in the `AAD_CLIENT_APP_ID`, `AAD_SERVER_APP_ID`, `AAD_SERVER_APP_SECRET`, and `AAD_TENANT_ID` parameters.

az/sample_use_cases/5_azureManagedAAD/k8s_cluster.conf

This file includes sample values for a use case where:

- A private AKS cluster is deployed (`PRIVATE_CLUSTER="true"`) so that the cluster is only accessible from within the Virtual Network or a connected network.
- An Azure-managed Active Directory (AAD) server is enabled (`ENABLE_AAD="true"`).
- The AKS resource provider manages the AAD client and server applications.

az/sample_use_cases/6_attachACR/k8s_cluster.conf

This file includes sample values for a use case where:

- A private AKS cluster is deployed (`PRIVATE_CLUSTER="true"`) so that the cluster is only accessible from within the Virtual Network or a connected network.
- The cluster is configured to retrieve images from an existing private Azure Container Registry (ACR) by specifying the name of the ACR in the `ATTACH_ACR` parameter.

az/sample_use_cases/7_clusterAutoscalerSupport/k8s_cluster.conf

This file includes sample values for a use case where:

- A private AKS cluster is deployed (`PRIVATE_CLUSTER="true"`) so that the cluster is only accessible from within the Virtual Network or a connected network.
- The Cluster Autoscaler service is enabled (`ENABLE_CLUSTER_AUTOSCALER="true"`) so that nodes are automatically added to the node pool when demand increases and removed from the node pool when demand decreases.
- The parameter `CLUSTER_AUTOSCALER_PROFILE` parameter is used to configure the autoscaler.

az/sample_use_cases/8_MonitoringEnabled/k8s_cluster.conf

This file includes sample values for a use case where:

- A private AKS cluster is deployed (`PRIVATE_CLUSTER="true"`) so that the cluster is only accessible from within the Virtual Network or a connected network.
- The Monitoring service is enabled (`AKS_ENABLE_ADDONS="monitoring"`) for the cluster.

az/sample_use_cases/9_RBACSupport/k8s_cluster.conf

This file includes sample values for a use case where:

- A private AKS cluster is deployed (`PRIVATE_CLUSTER="true"`) so that the cluster is only accessible from within the Virtual Network or a connected network.
- Azure Role-Based Access Control (RBAC) is enabled (`DISABLE_RBAC="false"`).

az/sample_use_cases/10_useExistingResources/k8s_cluster.conf

This file includes sample values for a use case where:

- A private AKS cluster is deployed (`PRIVATE_CLUSTER="true"`) so that the cluster is only accessible from within the Virtual Network or a connected network.
- You deploy the cluster into your existing Resource Group, Virtual Network, and Subnetwork by specifying the values for those resources in the `RESOURCE_GROUP=${RESOURCE_GROUP:-"<resource-group>"}`, `VNET_NAME=${VNET_NAME:-"<name>"}`, and `SUBNET_NAME="<subnet-name>"` parameters.

az/sample_use_cases/11_useProximityPlacementGroups/k8s_cluster.conf

This file includes sample values for a use case where:

- A private AKS cluster is deployed (`PRIVATE_CLUSTER="true"`) so that the cluster is only accessible from within the Virtual Network or a connected network.
- You define a Proximity Placement Group (PPG) so that Azure deploys compute resources into a logical grouping where they are physically located close to each other to reduce latency. You specify the PPG name and type of group in the `PPG=${PPG:-"<name>"}` and `PPG_TYPE=${PPG_TYPE:-"<type>"}` parameters.

Cluster Parameters

The contents of `k8s_cluster.conf` are shown below. Descriptions of the cluster parameters follow the contents.

```
ENABLE_MANAGED_IDENTITY="<enable-managed-identity>"
SP=${SP:-"<service-principal>"}
```

```
SP_VALIDITY_YEARS="<years>"
SP_ID="<id>"
SP_SECRET="<client-secret>"
RESOURCE_GROUP=${RESOURCE_GROUP:-"<resource-group>"}
RESOURCE_GROUP_TAGS="<tags>"
LOCATION=${LOCATION:-"<location>"}
SUBSCRIPTION_ID="<subscription-id>"
VNET_NAME=${VNET_NAME:-"<name>"}
VNET_CIDR="<vnet-cidr>"
VNET_TAGS="<tags>"
VNET_VM_PROTECTION="<vm-protection>"
SUBNET_NAME="<subnet-name>"
SUBNET_CIDR="<subnet-cidr>"
NODE_ZONES="<zones>"
NODEPOOL_NAME="<name>"
NODEPOOL_TAGS="<tags>"
MACHINE_TYPE="<machine-type>"
K8S_CLUSTER_NAME=${K8S_CLUSTER_NAME:-"<name>"}
K8S_CLUSTER_VERSION=${K8S_CLUSTER_VERSION:-"<kubernetes-version>"}
K8S_CLUSTER_NODE_COUNT="<node-count>"
K8S_NODE_ADMIN_USER="<admin-username>"
AKS_TAGS="<tags>"
AKS_ENABLE_ADDONS="<addons>"
PRIVATE_CLUSTER="<enable-private-cluster>"
LOAD_BALANCER_SKU="<load-balancer-sku>"
LB_BALANCER_IDLE_TIMEOUT=<load-balancer-idle-timeout>
LB_OUTBOUND_IP_PREFIXES="<load-balancer-outbound-ip-prefixes>"
LB_OUTBOUND_IPS="<load-balancer-outbound-ips>"
LB_OUTBOUND_PORTS=<load-balancer-outbound-ports>
LB_MANAGED_OUTBOUND_IP_COUNT=<load-balancer-managed-outbound-ip-count>
VM_SET_TYPE="<vm-set-type>"
NETWORK_PLUGIN="<network-plugin>"
NETWORK_POLICY="<network-policy>"
DOCKER_BRIDGE_ADDRESS="<docker-bridge-address>"
DNS_SERVICE_IP="<dns-service-ip>"
DNS_NAME_PREFIX="<dns-name-prefix>"
SERVICE_CIDR="<service-cidr>"
MIN_NODES="<min-count>"
MAX_NODES="<max-count>"
MAX_PODS_PER_NODE="<max-pods>"
DISK_SIZE="<node-osdisk-size>"
AZURE_CLI_VERSION="<azure-cli-version>"
NODE_OSDISK_TYPE="<node-osdisk-type>"
OS_DISK_ENCRYPTIONSET_ID="<node-osdisk-diskencryptionset-id>"
```

```
ENABLE_CLUSTER_AUTOSCALER="<enable-cluster-autoscaler>"
CLUSTER_AUTOSCALER_PROFILE="<cluster-autoscaler-profile>"
ATTACH_ACR="<attach-acr>"
ENABLE_AAD="<enable-aad>"
AAD_ADMIN_GROUP_OBJECT_IDS="<aad-admin-group-object-ids>"
AAD_CLIENT_APP_ID="<aad-client-app-id>"
AAD_SERVER_APP_ID="<aad-server-app-id>"
AAD_SERVER_APP_SECRET="<aad-server-app-secret>"
AAD_TENANT_ID="<tenant-id>"
ENABLE_POD_SECURITY_POLICY="<enable-pod-security-policy>"
DISABLE_RBAC="<disable-rbac>"
ENABLE_NODE_PUBLIC_IP="<enable-node-public-ip>"
SSH_PUB_KEY_VALUE="<ssh-key-value>"
API_SERVER_AUTHORIZED_IP_RANGES="<api-server-authorized-ip-ranges>"
NODEPOOL_LABELS="<nodepool-labels>"
PPG=${PPG:-"<name>"}
PPG_TYPE=${PPG_TYPE:-"<type>"}
UPTIME_SLA="<uptime-sla>"
OUTBOUND_TYPE="<outbound-type>"
```

ENABLE_MANAGED_IDENTITY

Indicates whether to use a system-assigned managed identity for cluster resource management. When enabled, this identity is used to create the K8s cluster resources. In addition, if Managed Identity is enabled, the Service Principal parameters (SP, SP_VALIDITY_YEARS, SP_ID, and SP_SECRET) are not required.

SP

The Service Principal to use for the AKS cluster. If you want to use an existing Service Principal, specify the name for that principal. If you want to create a new Service Principal, specify a new name, and the new Service Principal will be created when the cluster is created. For example, **aks-service-principal**.

SP_VALIDITY_YEARS

The number of years for which the Service Principal credentials should be valid. For example, **2**.

SP_ID

The ID for the existing Service Principal. Leave this value blank if you chose to create a new principal.

SP_SECRET

The secret for the existing Service Principal. Leave this value blank if you chose to create a new principal.

RESOURCE_GROUP

The name of the Azure Resource Group to allocate the AKS cluster resources to. You can specify the name of an existing group, or you can specify a new name if you want the K8s scripts to create a new Resource Group.

RESOURCE_GROUP_TAGS

A space-separated list of any tags (key=value pairs) to add to the Resource Group.

LOCATION

The Region code for the location where the AKS cluster will be deployed. For example, **eastus**.

SUBSCRIPTION_ID

The ID for your Azure subscription.

VNET_NAME

The name of the Virtual Network to provision the AKS cluster in. This value should match the name of the network that Anzo is deployed in.

VNET_CIDR

The IP address prefix in CIDR format to use for the Virtual Network.

> **Note**
> Supply this value even if VNET_NAME is not set and a new Virtual Network will be created.

VNET_TAGS

A space-separated list of any tags (in key=value format) to add to the Virtual Network.

VNET_VM_PROTECTION

A true or false value that indicates whether to enable VM protection for the subnets in the Virtual Network.

SUBNET_NAME

The name of the new subnetwork to create in the Virtual Network.

SUBNET_CIDR

The IP address prefix in CIDR format for the new subnetwork.

NODE_ZONES

The number of Availability Zones to place the agent nodes in. Valid values are **1**, **2**, or **3**.

NODEPOOL_NAME

The name to give the default node pool that is created in the AKS cluster.

NODEPOOL_TAGS

A space-separated list of any tags (in key=value format) to add to resources in the default node pool.

MACHINE_TYPE

The Virtual Machine Type to use for the nodes in the AKS cluster.

K8S_CLUSTER_NAME

The name to give the AKS cluster.

K8S_CLUSTER_VERSION

The version of Kubernetes to use for creating the cluster.

> **Note**
> Kubernetes versions 1.18 and 1.19 are supported. See the AKS Engine Release Notes for details about the available versions.

K8S_CLUSTER_NODE_COUNT

The number of nodes to deploy in the default node pool.

K8S_NODE_ADMIN_USER

The user account to create on the K8s cluster nodes for SSH access.

AKS_TAGS

A space-separated list of any tags (in key=value format) to add to the cluster.

AKS_ENABLE_ADDONS

A comma-separated list of addons to enable for the AKS cluster. Cambridge Semantics recommends that you include the **monitoring** addon.

PRIVATE_CLUSTER

Indicates whether to make the AKS cluster a private cluster. If the cluster is private, network traffic between the K8s API server and node pools remains on the private network.

> **Tip**
> When deciding whether to configure the cluster as a private cluster, you may want to review the Limitations described in "Create a private Azure Kubernetes Service cluster" in the Azure AKS documentation.

LOAD_BALANCER_SKU

The Azure Load Balancer SKU selection for your cluster. The options are **basic** or **standard**. The standard SKU is recommended for AKS clusters. For information about the SKUs, see Azure Load Balancer SKUs in the Azure documentation.

LB_BALANCER_IDLE_TIMEOUT

This optional parameter specifies the number of minutes to wait before dropping idle connections to the Load Balancer. For example, a value of **5** means that idle connections are dropped after 5 minutes. If this parameter is not specified, the default value is 30 minutes.

> **Tip**
> For more information about configuring the Load Balancer, including details about the idle timeout parameter as well as the outbound IP address and port parameters, see Configure the Public Standard Load Balancer in the Azure AKS documentation.

LB_OUTBOUND_IP_PREFIXES

This optional parameter specifies a comma-separated list of outbound IP prefix resource IDs.

LB_OUTBOUND_IPS

This optional parameter specifies a comma-separated list of outbound IP resource IDs.

LB_OUTBOUND_PORTS

This optional parameter specifies the number of outbound ports to allocate for the Load Balancer. For example, **8000**.

LB_MANAGED_OUTBOUND_IP_COUNT

This optional parameter specifies the number of AKS-managed outbound IP addresses to allocate for the Load Balancer. For example, **10**.

VM_SET_TYPE

The Agent pool VM set type. Valid values are **VirtualMachineScaleSets** or **AvailabilitySet**. Cambridge Semantics recommends that you set this value to **VirtualMachineScaleSets**.

NETWORK_PLUGIN

The type of Kubernetes network plugin to use, i.e. whether to use basic (kubenet) networking or advanced CNI (azure) networking. Valid values are **kubenet** or **azure**.

NETWORK_POLICY

The type of the network policy (Azure Network Policies or Calico Network Policies) to apply to the pods in the AKS cluster. The network policy defines the rules for ingress and egress traffic between pods in the cluster. Valid values are **azure** or **calico**. For information about the policies, see Network Policy Options in AKS in the Azure AKS documentation.

DOCKER_BRIDGE_ADDRESS

The CIDR block to use for the Docker bridge. The Docker bridge is not used by the AKS cluster or pods but does need to be set up since Docker is configured as part of the Kubernetes setup. Choose an address space that does not collide with any other CIDRs on your networks, including the cluster's service CIDR and pod CIDR. For example, **172.17.0.1/16**.

DNS_SERVICE_IP

The IP address to assign to the Kubernetes DNS service.

DNS_NAME_PREFIX

This optional parameter specifies the prefix to use for hostnames that are created for the DNS service. If not specified, a hostname is generated using the managed cluster and resource group names.

SERVICE_CIDR

The IP address range in CIDR notation from which to assign the Kubernetes DNS service IP addresses.

MIN_NODES

The minimum number of nodes in the default node pool.

MAX_NODES

The maximum number of nodes in the default node pool.

MAX_PODS_PER_NODE

The maximum number of pods deployable to a node in the default node pool.

DISK_SIZE

The size in GB of the OS disk for each node in the default node pool.

AZURE_CLI_VERSION

The version of the Azure CLI on the workstation. For example, **2.25.0**.

NODE_OSDISK_TYPE

The type of OS disk to use for machines in the cluster. The options are **Ephemeral** or **Managed**.

OS_DISK_ENCRYPTIONSET_ID

Specifies the Resource ID of the disk encryption set to use for encryption at rest on the agent node OS disk.

ENABLE_CLUSTER_AUTOSCALER

Indicates whether to enable the cluster autoscaler for the default node pool.

CLUSTER_AUTOSCALER_PROFILE

A space-separated list of any key=value pairs to use for configuring the Cluster Autoscaler. For example, **scan-interval=10s scale-down-delay-after-delete=10s**. For information about all of the configuration options, see Using the Autoscaler Profile in the Azure AKS documentation.

ATTACH_ACR

The name or resource ID of the Azure Container Registry to grant the `acrpull` role assignment to.

ENABLE_AAD

Indicates whether to enable managed Azure Active Directory (AAD) for the cluster. When AAD is enabled, the Admin Group Object IDs, AAD Client ID, Server ID, Server Secret, and Tenet ID parameters (AAD_ADMIN_GROUP_OBJECT_IDS, AAD_CLIENT_APP_ID, AAD_SERVER_APP_ID, AAD_SERVER_APP_SECRET, and AAD_TENANT_ID) are not required.

AAD_ADMIN_GROUP_OBJECT_IDS

This parameter specifies the comma-separated list of AAD group object IDs to set as cluster admin.

AAD_CLIENT_APP_ID

The ID of a "Native" type Azure Active Directory client application. This application is for user logins via kubectl.

AAD_SERVER_APP_ID

The ID of a "Web app/API" Azure Active Directory server application. This application represents the managed cluster's API Server (apiserver application).

AAD_SERVER_APP_SECRET

The secret for the Azure Active Directory server application.

AAD_TENANT_ID

The ID of the Azure Active Directory tenant.

ENABLE_POD_SECURITY_POLICY

Indicates whether to enable the pod security policy for the AKS cluster.

> **Note**
> Azure will deprecate this feature in June 2021. For information, see Secure your cluster using pod security policies in Azure Kubernetes Service (AKS) in the Azure AKS documentation.

DISABLE_RBAC

Indicates whether to disable Kubernetes Role-Based Access Control (RBAC).

ENABLE_NODE_PUBLIC_IP

Indicates whether to enable a public IP address for the Virtual Machine Scale Set (VMSS) node.

SSH_PUB_KEY_VALUE

The public key path or key contents to install on the K8s cluster nodes for SSH access. If not specified, the default value is **~\.ssh\id_rsa.pub**.

API_SERVER_AUTHORIZED_IP_RANGES

The list of IP address ranges in CIDR notation that are authorized to access the AKS cluster.

NODEPOOL_LABELS

A space-separated list (in key=value format) of labels to add to the nodes in the default node pool. For information about using labels in Kubernetes clusters, see Labels and Selectors in the Kubernetes documentation.

PPG

This optional parameter specifies the name of the Proximity Placement Group (PPG) to use for the cluster. For information about using proximity placement groups, see Use Proximity Placement Groups in the Azure AKS documentation.

PPG_TYPE

If using a Proximity Placement Group (PPG), this parameter specifies the type of PPG to use. The only valid value is **Standard**.

UPTIME_SLA

Indicates whether to enable a paid managed cluster service with a financially backed SLA.

OUTBOUND_TYPE

Specifies how to configure outbound traffic for the cluster. Valid values are **loadBalancer** and **userDefinedRouting**.

Example Configuration File

An example completed k8s_cluster.conf file is shown below.

```
ENABLE_MANAGED_IDENTITY="true"
#SP=${SP:-"aks-service-principal"}
#SP_VALIDITY_YEARS="2"
#SP_ID="291bba3f-e0a5-47bc-a099-3bdcb2a50a05"
#SP_SECRET="ValidServicePrincipalSecretIfPresent"
RESOURCE_GROUP=${RESOURCE_GROUP:-"aks-resource-group"}
RESOURCE_GROUP_TAGS="description=aks-cluster"
LOCATION=${LOCATION:-"eastus"}
SUBSCRIPTION_ID="ValidSubscriptionId"
```

```
VNET_NAME=${VNET_NAME:-"anzo-vnet"}
VNET_CIDR="20.20.0.0/16"
VNET_TAGS="description=aks-virtual-network"
VNET_VM_PROTECTION="true"
SUBNET_NAME="k8s-subnet"
SUBNET_CIDR="20.20.0.0/19"
#NODE_ZONES=""
NODEPOOL_NAME="defaultpool"
NODEPOOL_TAGS="description=default-nodepool"
MACHINE_TYPE="Standard_DS1_v2"
K8S_CLUSTER_NAME=${K8S_CLUSTER_NAME:-"k8s-cluster"}
K8S_CLUSTER_VERSION=${K8S_CLUSTER_VERSION:-"1.18"}
K8S_CLUSTER_NODE_COUNT="2"
K8S_NODE_ADMIN_USER="azureuser"
AKS_TAGS="description=aks-cluster"
AKS_ENABLE_ADDONS="monitoring"
PRIVATE_CLUSTER="false"
LOAD_BALANCER_SKU="standard"
#LB_BALANCER_IDLE_TIMEOUT=5
#LB_OUTBOUND_IP_PREFIXES="<ip-prefix-resource-id-1,ip-prefix-resource-id-2>"
#LB_OUTBOUND_IPS="<ip-resource-id-1,ip-resource-id-2>"
#LB_OUTBOUND_PORTS=8000
#LB_MANAGED_OUTBOUND_IP_COUNT=10
VM_SET_TYPE="VirtualMachineScaleSets"
NETWORK_PLUGIN="azure"
NETWORK_POLICY="azure"
DOCKER_BRIDGE_ADDRESS="172.17.0.1/16"
DNS_SERVICE_IP="10.0.0.10"
#DNS_NAME_PREFIX="k8stest"
SERVICE_CIDR="10.0.0.0/16"
MIN_NODES="1"
MAX_NODES="8"
MAX_PODS_PER_NODE="16"
DISK_SIZE="100"
AZURE_CLI_VERSION="2.19.1"
NODE_OSDISK_TYPE="Ephemeral"
#OS_DISK_ENCRYPTIONSET_ID=""
ENABLE_CLUSTER_AUTOSCALER="true"
CLUSTER_AUTOSCALER_PROFILE="scan-interval=10s scale-down-delay-after-delete=10s"
ATTACH_ACR="ContainerRegistry"
ENABLE_AAD="true"
AAD_ADMIN_GROUP_OBJECT_IDS="5d24455a-1111-3333-4444-5dv77afa27aed"
#AAD_CLIENT_APP_ID="ValidAADClientAppId"
```

```
#AAD_SERVER_APP_ID="ValidAADServerAppId"
#AAD_SERVER_APP_SECRET="ValidAADServerAppSecret"
#AAD_TENANT_ID="8f70baf1-1f6e-46a2-a1ff-238dac1ebfb7"
ENABLE_POD_SECURITY_POLICY="true"
ENABLE_MANAGED_IDENTITY="false"
DISABLE_RBAC="false"
SSH_PUB_KEY_VALUE=""
API_SERVER_AUTHORIZED_IP_RANGES="10.107.1.0/24"
NODEPOOL_LABELS="description=k8scluster"
#PPG=${PPG:-"csippg"}
#PPG_TYPE=${PPG_TYPE:-"Standard"}
UPTIME_SLA="false"
OUTBOUND_TYPE="loadBalancer"
```

Create the AKS Cluster

After defining the cluster requirements, run the **create_k8s.sh** script in the `az` directory to create the cluster. Run the script with the following command. The arguments are described below.

```
./create_k8s.sh -c <config_file_name> [ -d <config_file_directory> ] [ -f | --force ] [ -h | --help ]
```

Argument	Description
-c <config_file_name>	This is a **required** argument that specifies the name of the configuration file that supplies the cluster requirements. For example, **-c k8s_cluster.conf**.
-d <config_file_directory>	This is an **optional** argument that specifies the path and directory name for the configuration file specified for the -c argument. If you are using the original `az` directory file structure and the configuration file is in the `conf.d` directory, you do not need to specify the -d argument. If you created a separate directory structure for different Anzo environments, include the -d option. For example, **-d /az/env1/conf**.
-f | --force	This is an **optional** argument that controls whether the script prompts for confirmation before proceeding with each stage involved in creating the cluster. If **-f** (**--force**) is specified, the script assumes the answer is "yes" to all prompts and does not display them.

Argument	Description
-h | --help	This argument is an **optional** flag that you can specify to display the help from the create_k8s.sh script.

For example, the following command runs the create_k8s script, using k8s_cluster.conf as input to the script. Since k8s_cluster.conf is in the conf.d directory, the -d argument is excluded:

```
./create_k8s.sh -c k8s_cluster.conf
```

The script validates that the required software packages, such as the Azure CLI and kubectl, are installed and that the versions are compatible with the script. It also displays an overview of the deployment details based on the values in the specified configuration file.

The script then prompts you to proceed with deploying each component of the AKS cluster infrastructure. Type **y** and press **Enter** to proceed with each step in creating the specified Service Principal, Virtual Network, subnetwork, and Load Balancer components. All components are created according to the specifications in the configuration file.

When cluster creation is complete, proceed to Creating the Required Node Pools to add the required node pools to the cluster.

Creating the Required Node Pools

This topic provides instructions for creating the three types of required node pools:

- The **Operator** node pool for running the AnzoGraph, Anzo Agent with Anzo Unstructured (AU), and Elasticsearch operator pods.
- The **AnzoGraph** node pool for running AnzoGraph application pods.
- The **Dynamic** node pool for running Anzo Agent with AU and Elasticsearch application pods.

> **Tip**
> For more information about the node pools, see Anzo Kubernetes Requirements.

- Define the Node Pool Requirements
- Example Configuration Files
- Create the Node Pools

Define the Node Pool Requirements

Before creating the node pools, configure the infrastructure requirements for each type of pool. The **nodepool_*.conf** files in the `az/conf.d` directory are sample configuration files that you can use as templates, or you can edit the files directly:

- **nodepool_operator.conf** defines the requirements for the Operator node pool.
- **nodepool_anzograph.conf** defines the requirements for the AnzoGraph node pool.
- **nodepool_dynamic.conf** defines the requirements for the Dynamic node pool.

Each type of node pool configuration file contains the following parameters. Descriptions of the parameters and guidance on specifying the appropriate values for each type of node pool are provided below.

```
NODEPOOL_NAME="<name>"
KUBERNETES_VERSION="<kubernetes-version>"
DOMAIN="<domain>"
KIND="<kind>"
MACHINE_TYPE="<node-vm-size>"
LOCATION=${LOCATION:-"<location>"}
RESOURCE_GROUP=${RESOURCE_GROUP:-"<resource-group>"}
VNET_NAME=${VNET_NAME:-"<vnet-name>"}
SUBNET_NAME="<name>"
SUBNET_CIDR="<address-prefix>"
K8S_CLUSTER_NAME=${K8S_CLUSTER_NAME:-"<cluster-name>"}
NODE_TAINTS="<node-taints>"
MAX_PODS_PER_NODE=<max-pods>
MAX_NODES=<max-count>
MIN_NODES=<min-count>
NUM_NODES=<node-count>
DISK_SIZE="<node-osdisk-size>"
OS_TYPE="<os-type>"
PRIORITY="<priority>"
ENABLE_CLUSTER_AUTOSCALER=<enable-cluster-autoscaler>
LABELS="<nodepool-labels>"
MODE="<mode>"
```

```
NODE_OSDISK_TYPE="<node-osdisk-type>"
PPG="<name>"
PPG_TYPE=${PPG_TYPE:-"<type>"}
```

NODEPOOL_NAME

The name to give the node pool.

Node Pool Type	Sample NODEPOOL_NAME Value
Operator	csi-operator
AnzoGraph	csi-anzograph
Dynamic	csi-dynamic

KUBERNETES_VERSION

The version of Kubernetes to use for creating the node pool. This value must match the AKS cluster version (K8S_CLUSTER_VERSION). For example, **1.24**.

DOMAIN

The name of the domain that hosts the node pool. This is typically the name or acronym for the organization, such as **csi**.

KIND

This parameter classifies the node pool in terms of kernel tuning and the type of pods that the node pool will host.

Node Pool Type	Required KIND Value
Operator	operator
AnzoGraph	anzograph

Node Pool Type	Required KIND Value
Dynamic	dynamic

MACHINE_TYPE

The Virtual Machine Type to use for the nodes in the node pool.

Node Pool Type	Sample MACHINE_TYPE Value
Operator	Standard_DS2_v2
AnzoGraph	Standard_D16s_v3
Dynamic	Standard_D8s_v3

> **Tip**
> For more guidance on determining the instance types to use for nodes in the required node pools, see Compute Resource Planning.

LOCATION

The Region code for the location of the AKS cluster. For example, **eastus**.

RESOURCE_GROUP

The name of the Azure Resource Group to allocate the node pool's resources to. You can specify the name of an existing group, or you can specify a new name if you want the K8s scripts to create a new Resource Group for the node pool.

VNET_NAME

The name of the Virtual Network that the AKS cluster was deployed in.

SUBNET_NAME

The name of the subnetwork to create.

SUBNET_CIDR

The IP address prefix to use when creating the subnetwork.

K8S_CLUSTER_NAME

The name of the AKS cluster.

NODE_TAINTS

This parameter configures a node so that the scheduler avoids or prevents using it for hosting certain pods. When a pod is scheduled for deployment, the scheduler relies on this value to determine whether the pod belongs in this pool. If a pod has a **toleration** that is not compatible with this **taint**, the pod is rejected from the pool. The table below lists the recommended values. The `NoSchedule` value means a toleration is required and pods without the appropriate toleration will not be allowed in the pool.

Node Pool Type	Recommended NODE_TAINTS Value
Operator	cambridgesemantics.com/dedicated=operator:NoSchedule
AnzoGraph	cambridgesemantics.com/dedicated=anzograph:NoSchedule
Dynamic	cambridgesemantics.com/dedicated=dynamic:NoSchedule

MAX_PODS_PER_NODE

The maximum number of pods that can be hosted on a node in the node pool. In addition to Anzo application pods, this limit also needs to account for K8s service pods and helper pods. Cambridge Semantics recommends that you set this value to at least **16** for all node pool types.

MAX_NODES

The maximum number of nodes that can be deployed in the node pool.

Node Pool Type	Sample MAX_NODES Value
Operator	8
AnzoGraph	16
Dynamic	32

MIN_NODES

The minimum number of nodes to remain deployed in the node pool at all times. If the cluster autoscaler is enabled for the node pool, you can set this value to **1** (the lowest value allowed by AKS). The autoscaler will automatically provision additional nodes if multiple pods are scheduled for deployment.

NUM_NODES

The number of nodes to deploy when this node pool is created. This value must be set to at least **1**. When you create the node pool, at least one node in the pool needs to be deployed as well.

DISK_SIZE

The size in GB of the OS disk for each node in the node pool.

Node Pool Type	Sample DISK_SIZE Value
Operator	50
AnzoGraph	100
Dynamic	100

OS_TYPE

The operating system to use for the nodes in the node pool. Specify **Linux** for each type of node pool.

PRIORITY

Specifies the priority level of the VMs for the nodes in the node pool. Valid values are **Regular** (dedicated) or **Spot** (low-priority or preemptible).

ENABLE_CLUSTER_AUTOSCALER

Indicates whether to enable the cluster autoscaler for the node pool.

LABELS

A space-separated list (in key=value format) of labels that define the type of pods that can be placed on the nodes in this node pool. One label, `cambridgesemantics.com/node-purpose`, is **required** for each type of node pool. The node-purpose label indicates that the purpose of the nodes in the pools are to host operator, anzograph, or dynamic pods. The table below lists the required labels for each node pool.

Node Pool Type	Required NODE_LABELS Value
Operator	cambridgesemantics.com/node-purpose=operator
AnzoGraph	cambridgesemantics.com/node-purpose=anzograph
Dynamic	cambridgesemantics.com/node-purpose=dynamic

For information about using labels in Kubernetes clusters, see Labels and Selectors in the Kubernetes documentation.

MODE

The mode for the node pool. The mode defines the node pool's primary function, i.e., whether it is a **System** node pool or a **User** pool. System node pools serve the primary purpose of hosting critical system pods. User node pools serve the primary purpose of hosting application pods. For the Operator, AnzoGraph, and Dynamic node pools, the mode should be set to **User**. For more information, see System and User Node Pools in the Azure AKS documentation.

NODE_OSDISK_TYPE

The type of OS disk to use for machines in the node pool. The options are **Ephemeral** or **Managed**.

PPG

This optional parameter specifies the name of the Proximity Placement Group (PPG) to use for the node pool. For information about using proximity placement groups, see Use Proximity Placement Groups in the Azure AKS documentation.

PPG_TYPE

If using a Proximity Placement Group (PPG), this parameter specifies the type of PPG to use. The only valid value is **Standard**.

Example Configuration Files

Example completed configuration files for each type of node pool are shown below.

Operator Node Pool

The example below shows a configured nodepool_operator.conf file.

```
NODEPOOL_NAME="csi-operator"
KUBERNETES_VERSION="1.24"
DOMAIN="csi"
KIND="operator"
MACHINE_TYPE="Standard_DS2_v2"
LOCATION=${LOCATION:-"eastus"}
```

```
RESOURCE_GROUP=${RESOURCE_GROUP:-"aks-resource-group"}
VNET_NAME=${VNET_NAME:-"anzo-vnet"}
SUBNET_NAME="k8s-subnet"
SUBNET_CIDR="20.20.2.0/19"
K8S_CLUSTER_NAME=${K8S_CLUSTER_NAME:-"k8s-cluster"}
NODE_TAINTS="cambridgesemantics.com/dedicated=operator:NoSchedule"
MAX_PODS_PER_NODE=16
MAX_NODES=8
MIN_NODES=1
NUM_NODES=1
DISK_SIZE="50"
OS_TYPE="Linux"
PRIORITY="Regular"
ENABLE_CLUSTER_AUTOSCALER=true
LABELS="cambridgesemantics.com/node-purpose=operator"
MODE="User"
NODE_OSDISK_TYPE="Managed"
#PPG="testppg"
#PPG_TYPE=${PPG_TYPE:-"standard"}
```

AnzoGraph Node Pool

The example below shows a configured nodepool_anzograph.conf file.

```
NODEPOOL_NAME="csi-anzograph"
KUBERNETES_VERSION="1.24"
DOMAIN="csi"
KIND="anzograph"
MACHINE_TYPE="Standard_D16s_v3"
LOCATION=${LOCATION:-"eastus"}
RESOURCE_GROUP=${RESOURCE_GROUP:-"aks-resource-group"}
VNET_NAME=${VNET_NAME:-"anzo-vnet"}
SUBNET_NAME="k8s-subnet"
SUBNET_CIDR="20.20.2.0/19"
K8S_CLUSTER_NAME=${K8S_CLUSTER_NAME:-"k8s-cluster"}
NODE_TAINTS="cambridgesemantics.com/dedicated=anzograph:NoSchedule"
MAX_PODS_PER_NODE=16
MAX_NODES=16
MIN_NODES=1
NUM_NODES=1
DISK_SIZE="100"
OS_TYPE="Linux"
PRIORITY="Regular"
```

```
ENABLE_CLUSTER_AUTOSCALER=true
LABELS="cambridgesemantics.com/node-purpose=anzograph"
MODE="User"
NODE_OSDISK_TYPE="Managed"
#PPG="testppg"
#PPG_TYPE=${PPG_TYPE:-"standard"}
```

Dynamic Node Pool

The example below shows a configured nodepool_dynamic.conf file.

```
NODEPOOL_NAME="csi-dynamic"
KUBERNETES_VERSION="1.24"
DOMAIN="csi"
KIND="dynamic"
MACHINE_TYPE="Standard_D8s_v3"
LOCATION=${LOCATION:-"eastus"}
RESOURCE_GROUP=${RESOURCE_GROUP:-"aks-resource-group"}
VNET_NAME=${VNET_NAME:-"anzo-vnet"}
SUBNET_NAME="k8s-subnet"
SUBNET_CIDR="20.20.2.0/19"
K8S_CLUSTER_NAME=${K8S_CLUSTER_NAME:-"k8s-cluster"}
NODE_TAINTS="cambridgesemantics.com/dedicated=dynamic:NoSchedule"
MAX_PODS_PER_NODE=16
MAX_NODES=32
MIN_NODES=1
NUM_NODES=1
DISK_SIZE="100"
OS_TYPE="Linux"
PRIORITY="Regular"
ENABLE_CLUSTER_AUTOSCALER=true
LABELS="cambridgesemantics.com/node-purpose=dynamic"
MODE="User"
NODE_OSDISK_TYPE="Managed"
#PPG="testppg"
#PPG_TYPE=${PPG_TYPE:-"standard"}
```

Create the Node Pools

After defining the requirements for the node pools, run the **create_nodepools.sh** script in the `az` directory to create each type of node pool. Run the script once for each type of pool.

Note

The `create_nodepools.sh` script references the files in the `az/reference` directory. If you customized the directory structure on the workstation, ensure that the **reference** directory is available at the same level as `create_nodepools.sh` before creating the node pools.

Run the script with the following command. The arguments are described below.

```
./create_nodepools.sh -c <config_file_name> [ -d <config_file_directory> ] [ -f | --force ] [ -h | --help ]
```

Argument	Description
-c <config_file_name>	This is a **required** argument that specifies the name of the configuration file (i.e., `nodepool_operator.conf`, `nodepool_anzograph.conf`, or `nodepool_dynamic.conf`) that supplies the node pool requirements. For example, **-c nodepool_dynamic.conf**.
-d <config_file_directory>	This is an **optional** argument that specifies the path and directory name for the configuration file specified for the -c argument. If you are using the original `az` directory file structure and the configuration file is in the `conf.d` directory, you do not need to specify the -d argument. If you created a separate directory structure for different Anzo environments, include the -d option. For example, **-d /az/env1/conf**.
-f | --force	This is an **optional** argument that controls whether the script prompts for confirmation before proceeding with each stage involved in creating the node pool. If **-f** (**--force**) is specified, the script assumes the answer is "yes" to all prompts and does not display them.
-h | --help	This argument is an **optional** flag that you can specify to display the help from the create_nodepools.sh script.

For example, the following command runs the create_nodepools script, using nodepool_operator.conf as input to the script. Since nodepool_operator.conf is in the conf.d directory, the -d argument is excluded:

```
./create_nodepools.sh -c nodepool_operator.conf
```

The script validates that the required software packages, such as the Azure CLI and kubectl, are installed and that the versions are compatible with the script. It also displays an overview of the node pool deployment details based on the values in the specified configuration file.

The script then prompts you to proceed with deploying each component of the node pool. Type **y** and press **Enter** to proceed with the configuration.

Once the Operator, AnzoGraph, and Dynamic node pools are created, the next step is to create a Cloud Location in Anzo so that Anzo can connect to the AKS cluster and deploy applications. See Connecting to a Cloud Location in the Administration Guide.