Reference Architecture for Google Anthos with Lenovo ThinkAgile VX

Last update: **22 March 2023**

Describes the business case for modern micro-services, containers, and multi-cloud

Provides an overview of Google Kubernetes Engine (GKE) on-premises with Anthos

Describes the architecture and implementation of Anthos solution on ThinkAgile VX hyperconverged infrastructure

Provides Anthos use-cases and examples including DevOps, Service management, Hybrid and Multi-cloud

David West
Chandrakandh Mouleeswaran
Xiaotong Jiang
Markesha Parker

Table of Contents

1 Introduction

This document describes the reference architecture for Google Cloud's Anthos Hybrid Cloud solution based on the Lenovo ThinkAgile VX VMware vSAN certified platform. The document provides a technical overview of Google Kubernetes Engine (GKE) On-prem, which is a containerized workload orchestration software. We will cover the functional aspects of Anthos core components including the Kubernetes, Istio service mesh, Anthos config management, Hybrid and multi-cloud management, and Google cloud marketplace. We will also provide an architecture overview and implementation of Anthos on top of Lenovo ThinkAgile VX hyperconverged infrastructure (HCI) platform. In addition, this document provides various example customer use cases for Anthos, including Continuous Integration/Continuous Delivery (CI/CD), Micro-services and Service Mesh, Hybrid Cloud and Multi-cloud management, and Anthos Config Management.

The reference architecture is intended for IT decision makers, infrastructure and application architects looking to plan and implement hybrid cloud and leverage Google Kubernetes Engine container platform to build modern applications on their on-prem data centers and implement a hybrid cloud with Google Cloud Connect. Knowledge of containers, Kubernetes, cloud, and data center infrastructure architecture will be helpful.

This reference architecture covers the following products:

- Google Kubernetes Engine (GKE) On-prem (Anthos) version 1.12.0-gke.446
- Kubernetes version v1.23.5-gke.1504
- VMware vSphere ESXi 7.0.3 , vCenter 7.0.3
- VMware vSAN 7.0.3
- Seesaw bundled VM based network load balancer

Lenovo has certified the ThinkAgile VX solution as an Anthos Ready virtualized platform, with bare-metal certification coming soon. Lenovo successfully deployed the Anthos version 1.12 solution in a 3 node ThinkAgile VX VMware 7 environment, to validate and complete the certification. The Anthos Ready platform ensures Lenovo servers support the latest versions and have been tested for Anthos clusters on VMware, as part of Google Distributed Cloud Virtual (GDC Virtual). The validation includes deployment and scaling of a cluster along with interaction of GDC Virtual with the overall Lenovo compute, network, and storage solution. Finally, it validates cluster deployment on-premises and exposes a workload through one of the supported load balancer options on GDC Virtual clusters. With this, Lenovo demonstrates a strong understanding of Anthos architectures, features, and the hybrid cloud benefits available to customers from Anthos.

https://cloud.google.com/anthos/docs/resources/partner-platforms

In addition, the ThinkAgile VX has been certified for Intel Select Solution for Anthos.

https://www.intel.com/content/www/us/en/products/solutions/select-solutions/cloud/google-cloud-anthos.html

This document provides an overview of the business problem that is addressed by Anthos and the business value that is provided by the various Anthos components. A description of customer requirements is followed by an architectural overview of the solution and the logical components. The operational model describes the architecture for deploying Anthos on ThinkAgile VX platform, deployment considerations, network architecture, and other requirements. The last section provides the Bill of Materials for the hardware configurations for the Lenovo ThinkAgile VX certified nodes and appliances and networking hardware that is used in the solution.

2 Business problem and business value

2.1 Business problem

Technology is one of the primary drivers of innovation for businesses in every industry today. Highly competitive and disruptive companies have successfully adopted new and emerging technologies and modern development practices including Cloud, Intelligent automation/DevOps, Artificial Intelligence (AI), Machine Learning (ML), Big Data and Analytics, and so forth. These new technologies are helping companies bring innovative and superior products and services quickly to the market, and deliver an outstanding customer experience by harnessing the power of data and extracting insights critical to their business, customer, and competition.

In order to take advantage of these emerging technologies, many companies are going through a modernization phase to transform their legacy IT systems, processes, as well as culture to become lean and agile and to build the right kind of technology capabilities while controlling costs.

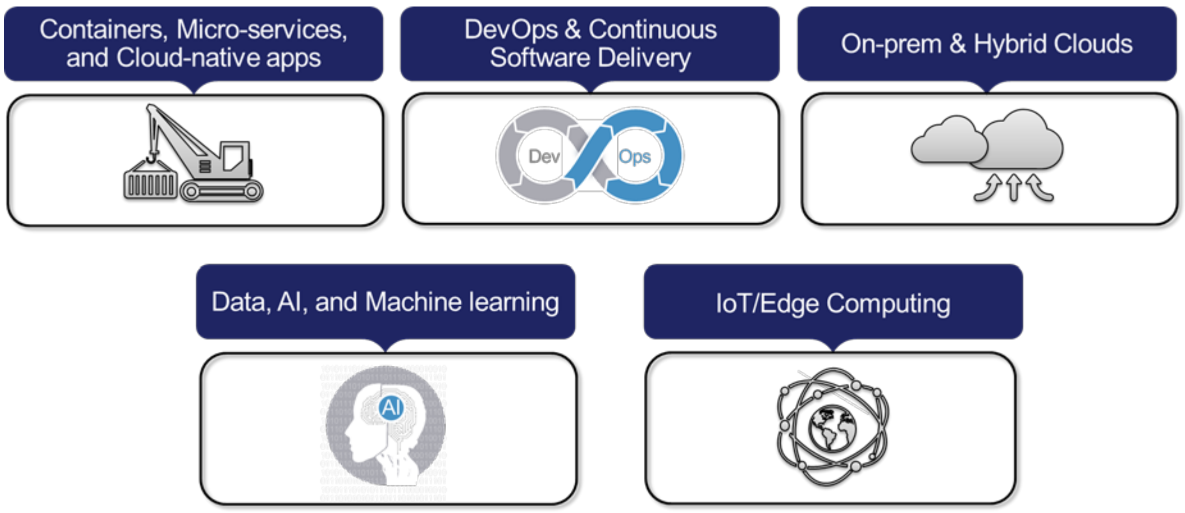

Containers, Micro-services, and Cloud-native apps
DevOps & Continuous Software Delivery
Dev
Ops
On-prem & Hybrid Clouds
Data, AI, and Machine learning
AI
IoT/Edge Computing

Figure 1: Key technology trends

With the increasing cost and competitive pressures, organizations are being forced to rethink their business strategy. Below are some of the common concerns you come across in every business these days:

- How can they continue to stay innovative and relevant in the marketplace while controlling costs
- How can they improve the profitability of the products and services in a crowded market
- How to improve customer satisfaction by quickly delivering new products, features, and better service
- How to establish market leadership by bringing innovative products before competition
- How to build an agile workforce that can react quickly to customer requests and market trends
- How to withstand being disrupted by new market entrants

While some of the business problems above require non-technical solutions, the key technical challenges can be addressed by the following capabilities:

- Creation of a modern application development environment to take advantage of containers, micro-services, and Kubernetes.

- Create a hybrid cloud environment to provide the most flexible IT infrastructure and services to users.
- Provide a single administrative control plane for centralized policy and security across clouds.
- Enable access to a broad eco-system of applications from the cloud marketplace for on-demand consumption.

2.2 Business Value

Google Cloud's Anthos is a modern application development and hybrid cloud technology platform from Google. Anthos enables deployment of some of the key public cloud capabilities in customers' own on-prem data centers. One of the core components of Anthos is the on-prem version of the popular Google Kubernetes Engine container orchestrator, which enables development of modern applications based on micro-services architecture. Customers have the flexibility to develop and test their workloads on-prem and then decide where they want to deploy them – either on-prem or in the public cloud. Multiple different cloud providers today support running Kubernetes clusters including Google compute platform, AWS, or Azure. In addition to the Kubernetes engine, Anthos includes other software capabilities – open source Istio service mesh, Anthos config management, GKE connect for hybrid connectivity and centralized management, multi-cluster management, Google application marketplace, and so forth.

One of the key on-prem components required to deploy Anthos is the infrastructure platform. Lenovo has partnered with Google to certify the Lenovo ThinkAgile VX platform for Anthos. VX provides VMware vSAN certified hyperconverged infrastructure (HCI) servers with a rich set of configurable options depending upon the application workload and business needs. Anthos works directly on the ThinkAgile VX platform without any modifications. Together with Lenovo ThinkAgile VX hyperconverged infrastructure, Anthos provides a turnkey on-prem cloud solution and enables management of Kubernetes container engine clusters from a central control plane.

3 Requirements

This chapter describes the functional and non-functional requirements for this reference architecture.

3.1 Introduction

The following section describes some background on emerging customer requirements around new technologies. This will help setup the discussion for the rest of the reference architecture.

3.1.1 Modern application development

Many companies still have legacy software applications that are not easy to change for a variety of reasons, both technical and business. In order to move fast in a rapidly changing market landscape, companies need to be able to quickly prototype, test, and deploy their new ideas. Modern techniques for software development have emerged in the recent years.

Software that follows a “monolithic” architecture is designed with simplicity and manageability on mind, and typically has a single code-base that included all functional modules such as the user interface tier, business logic, data access, authentication/authorization, etc., within a single application unit. While software best practices are followed in writing the code, e.g. modularity and loose-coupling to allow various functional modules to be independently written and maintained, the application development, integration testing, production deployment, and release lifecycle has to be managed with tight coordination across all development teams because all functional modules had to be combined to produce a single application.

Monolithic applications have several drawbacks:

- Fixes to one or more modules or new features will force the full program to be rebuilt, tested, and released.
- Bugs in any part of the code could affect the entire application
- You cannot independently scale or provide high-availability to different functional units

Micro-services is the new architecture paradigm that addresses some of the issues in monolithic application design. In this design the code is broken down into smaller functional modules, which are deployed as independent applications, exposing a service API. This architecture preserves the benefits of loose-coupling and modularity, while also detaching the modules such that they act as independent services. In addition, micro-services architecture enables smaller functional modules to be developed by independent teams and deployed anywhere- on on-prem data center infrastructure, or on the cloud. For instance, the checkout function on an e-commerce website can be implemented as a micro-service and can be shared by multiple product lines on the same site. Figure 2 illustrates such an example for an n-tier traditional business application broken down into several micro-services interacting with each other.

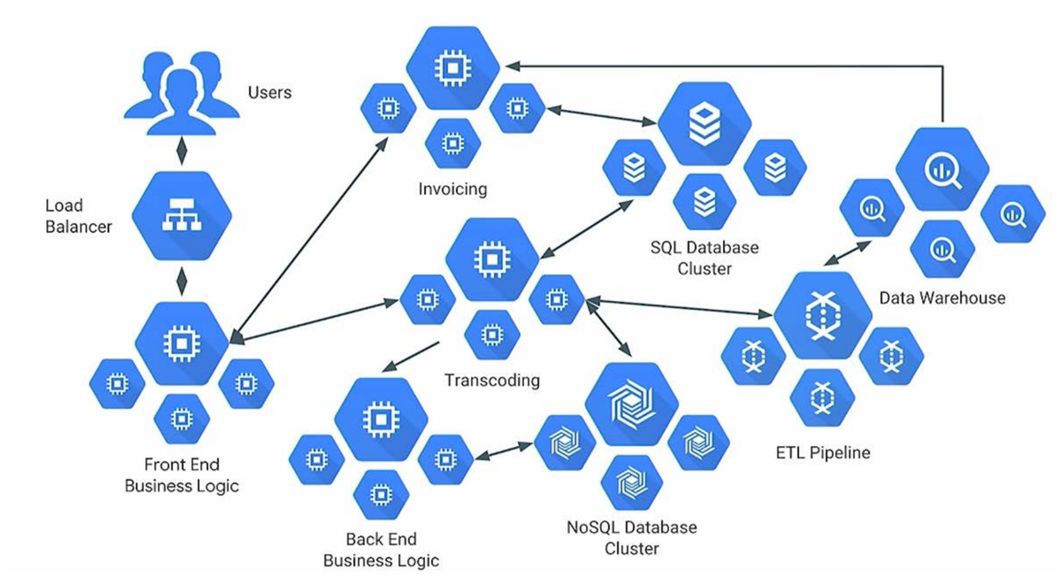


Figure 2: Micro-services architecture for an example e-commerce application

3.1.2 Containers

Containers are a new way of running traditional applications and micro-services at-scale, and managing their lifecycle with an orchestration engine such as Kubernetes. Containers provide a lightweight packaging of the application and its required runtime libraries into an image, and run this image on top of traditional operating systems, either on baremetal hardware or on virtualized infrastructure. Figure 3 shows the new architectural layers from hardware up to the applications. Docker is an open source project, which provides the user runtime libraries and tools to build, deploy, and manage containers. Kubernetes orchestrates containers at-scale on a cluster of machines running Docker.

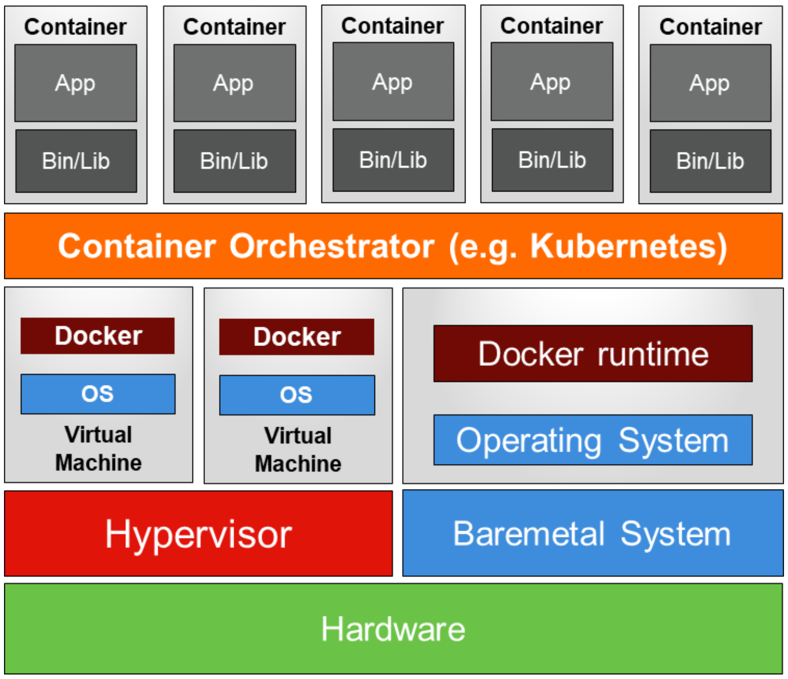


Figure 3: System architecture for containerized applications

3.1.3 DevOps

DevOps combines people, processes, and technology into a practice of delivering software at high velocity through end-to-end automation. The methodology integrates infrastructure, operations, application development, and software tools required to implement a continuous integration and continuous delivery (CI/CD) pipeline for code. Successful DevOps implementation requires a shift from legacy IT to modern IT infrastructure that is flexible, scalable, supports modern technologies, and cloud-like consumption and self-service.

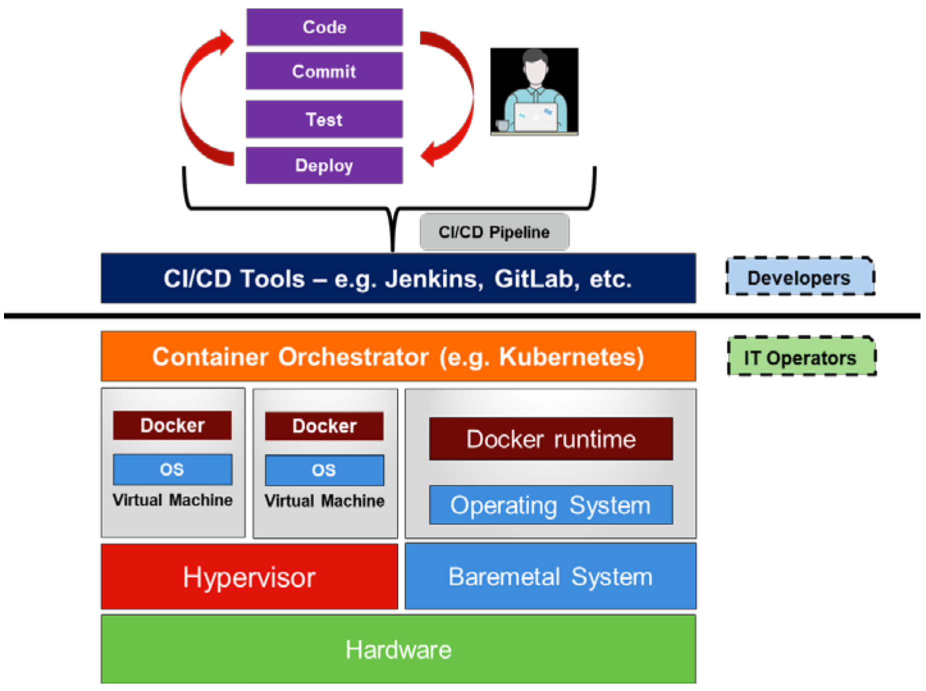


Figure 4: CI/CD Pipeline Architecture

3.1.4 Hybrid cloud

Hybrid cloud provides a flexible IT environment to users to run their workloads either on-premises or on the public cloud on-demand. Configuring a true hybrid cloud requires capabilities at various layers of the stack from hardware up to the applications and administrative services. Consequently, implementing and operating a hybrid cloud has been complex and costly for many organizations

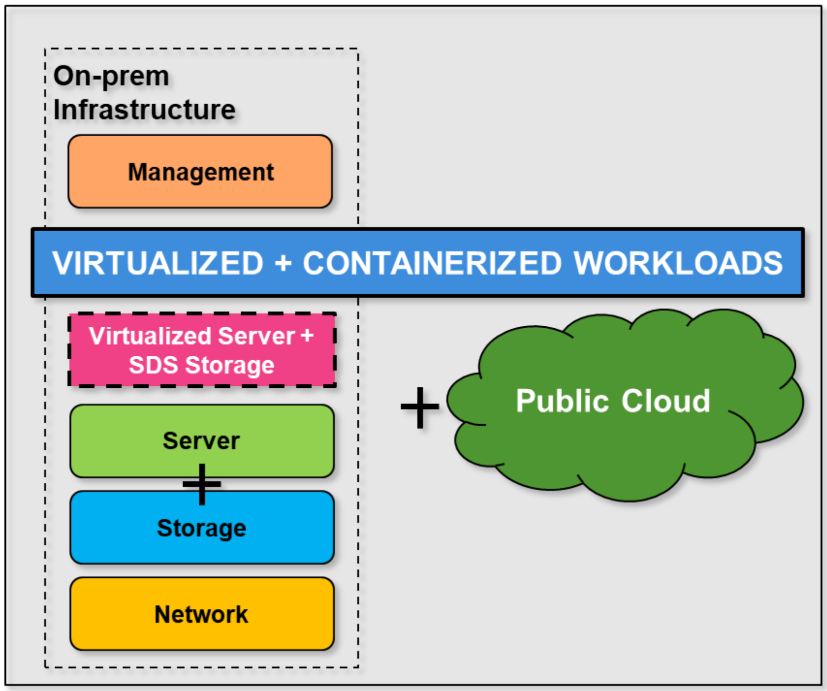

On-prem Infrastructure
Management
VIRTUALIZED + CONTAINERIZED WORKLOADS
Virtualized Server + SDS Storage
Server
+
Storage
Network
+
Public Cloud

Figure 5: Modern Hybrid Cloud Architecture

As modern application development accelerates along with the proliferation of data, hybrid cloud becomes a necessary capability to have for any company, small or large. In addition, new applications will be "cloud-native" by design, which enables them to run on the cloud, allowing for greater flexibility, security, availability, scalability, and other advantages provided by the cloud. Figure 5 illustrates how applications now share a common architecture spanning on-prem and public cloud. Hybrid cloud provides the necessary bridging between on-prem and cloud.

3.2 Functional Requirements

The following section describes the functional requirements that are needed for implementing a modern application development and hybrid cloud platform.

Table 1: Functional requirements

Requirement name	**Description**
Containerization	Solution provides the compute, storage, and network to support containerized workloads
Monitoring, event and capacity management	Monitors the health of the cloud infrastructure, collection and management of exception events, and capacity planning
Self-service automation	Solution provides on boarding, provisioning, and management of services and containers from a service catalog
On-prem cloud administration	Provides capabilities to administer a cloud environment, such as adding storage or computational resources in the cloud pool or defining new segregated networks

Image management	Provides capabilities to create containers, establish version control, search for and compare images, and delete images from the container registries
Service management	Provides capabilities to create services, establish version control, search for services, and delete services from the service templates catalog repositories
Access and authorization Controls	Provides the capabilities to create users and groups and to establish authorization to certain features in the cloud, such as tenant cloud administration, service developer, and user service requester
Container Migration	Migrate container images between private and public clouds.
Centralize Configuration Management	Provide a central management repository for multiple cloud configurations
Catalog Management	Maintain common catalog for templates across clouds.

Following table provides the various components of Anthos and how they address the functional requirements.

Table 2: Anthos solution components

Core Anthos Core Function	Public Cloud Component	On-Premises Component
Google Kubernetes Engine (GKE) for container orchestration	Managed Kubernetes on Google Compute Platform (GCP)	GKE on-prem version 1.12
Multicluster Management	Via GCP console and control plane	Via GCP console and control plane
Configuration Management	Anthos Config Management (1.0)	Anthos Config Management (1.12)
VM migration to containers	Migrate for Anthos (Beta)	N/A
Service Mesh	Istio on GKE Traffic Director	Istio OSS (1.1.7)
Logging & Monitoring	Stackdriver Logging, Stackdriver Monitoring, alerting	Stackdriver for system components
Container Marketplace	Kubernetes Applications in GCP Marketplace	Kubernetes Applications in GCP Marketplace

3.3 Non-functional requirements

Table 3: Non-functional requirements

Requirement	Description
Scalability	Solution components scale for growth
Load balancing	Workload is distributed evenly across servers
Fault tolerance	Single component error will not lead to whole system unavailability
Physical footprint	Compact solution
Ease of installation	Reduced complexity for solution deployment
Ease of management/operations	Reduced complexity for solution management
Flexibility	Solution supports variable deployment methodologies
Security	Solution provides means to secure customer infrastructure
High performance	Solution components are high-performance

4 Architectural overview

This chapter gives an architectural overview of Anthos components. Figure 6 gives a high-level overview of the multi-cloud architecture of Google cloud platform (GCP).

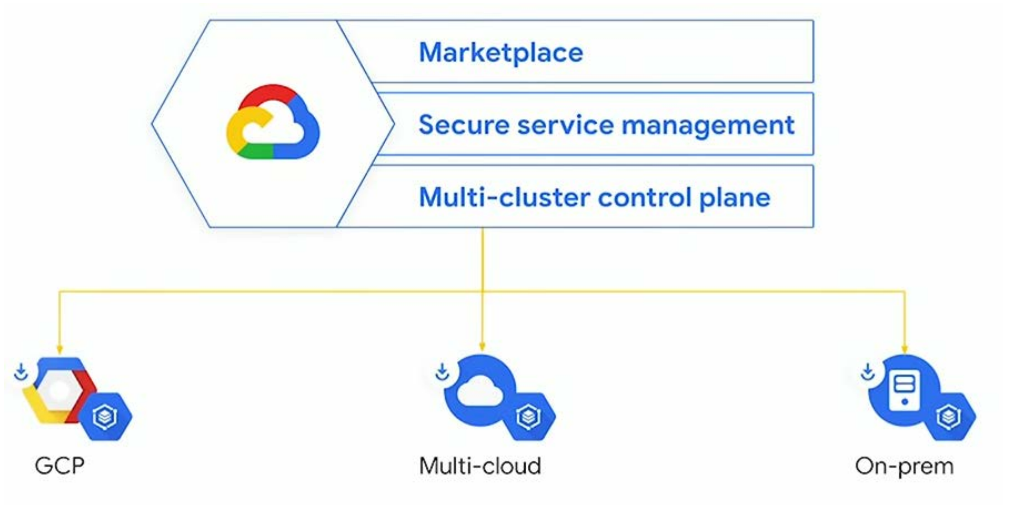


Figure 6: Anthos multi-cloud architecture

In this architecture, the Google cloud platform console provides a single control plane for managing Kubernetes clusters deployed in multiple locations – on the Google public cloud, on the on-prem data center, or other cloud provide such as AWS. In this sense, the on-prem GKE cluster is essentially an extension of the public cloud. GCP provides the centralized configuration and security management across the clusters, and the services running in different clusters can be managed and connected through the Istio service mesh. This centralized control provides a consistent mechanism to manage distributed Kubernetes clusters, configuration policy, and security. In addition, the Google container marketplace is available to deploy workloads to any of the clusters managed from the control plane.

The deployment of Anthos requires a VMware vSAN cluster, which provides the compute and storage virtualization. The GKE on-prem clusters from Anthos will be deployed as virtual machines running on top of the vSAN cluster. Hence, the master and worker nodes that are part of the GKE on-prem clusters are implemented as virtual machines instead of physical hosts. This simplifies the Anthos deployments as well because you do not need dedicated hosts for implementing GKE clusters. Instead, multiple GKE clusters can be installed on the same vSAN cluster.

Most of this document refers to a VMware vSAN configuration. However, as an alternate storage option, GKE on-prem clusters for Anthos can also be deployed in a traditional SAN environment. SAN volumes are mapped to all VMware cluster nodes as usual, where VMware data stores reside. The Anthos deployment uses this storage for all volumes and persistent storage created on the GKE cluster nodes. Both of these storage options have been validated on the Lenovo ThinkAgile VX VMware certified platform.

See Figure 7 for the architecture of the GKE on-prem clusters when deployed on VMware vSAN. There are three main components:

Admin workstation – this is the VM that acts as the deployment host for all GKE clusters on-prem. You would login to the admin workstation to kick-off the Anthos cluster deployments and configuration.

Admin GKE cluster – Every Anthos deployment requires a admin GKE cluster to be deployed first. This cluster acts as the central control plane for all user GKE clusters and acts as the connector between the on-prem clusters and Google compute platform.

User GKE cluster – You can deploy one or more user GKE clusters once the admin workstation and admin cluster are installed. The user GKE clusters execute the user container workloads. These clusters can be managed from the GCP console once the clusters are installed and registered with GCP.

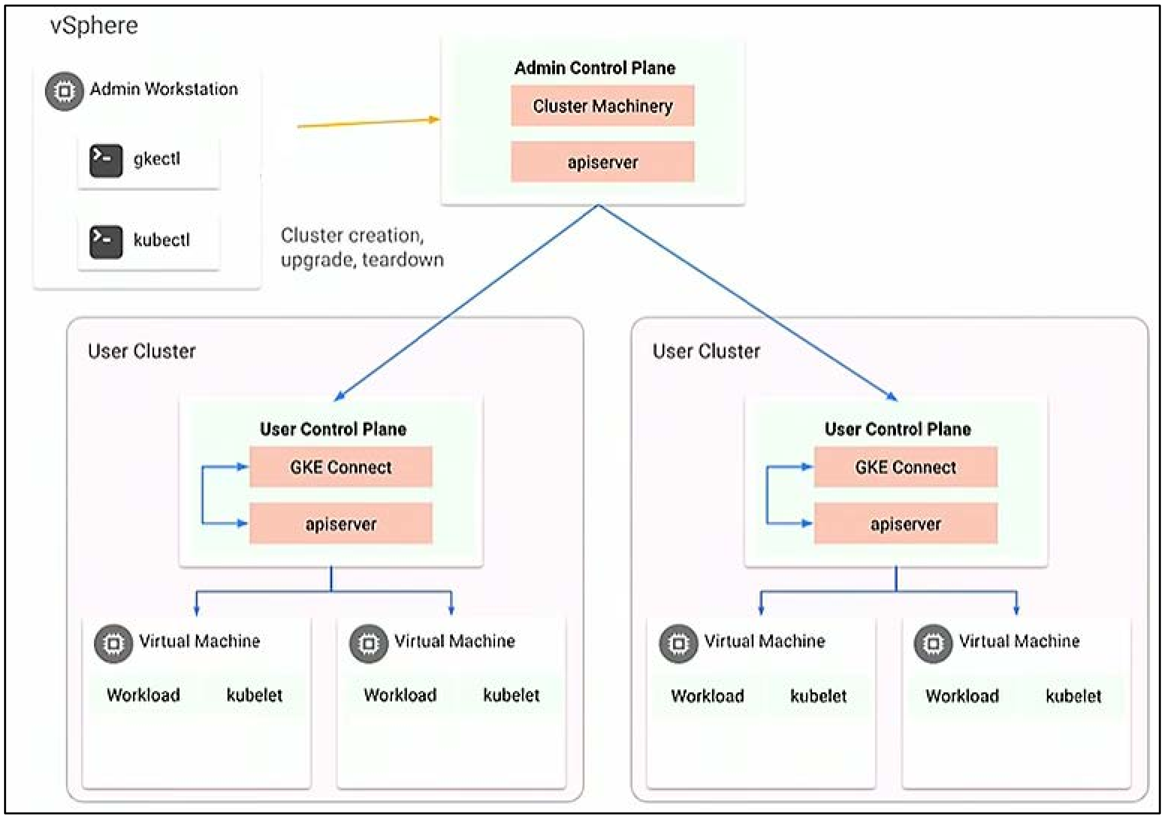


Figure 7: Anthos deployment architecture on VMware

For more information on the GKE on-prem, see Google documentation below:

https://cloud.google.com/anthos/docs/concepts/overview

5 Component model

Anthos multi-cloud architecture consists of a set of core software components that run on the Google cloud platform (GCP) and the Kubernetes deployments running in other clouds including on-prem, AWS, and Google public cloud. Together, the components provide all the services required to orchestrate container workloads across the different clouds as well as provide the common policy framework, centralized configuration management, security, service management, and access to the container marketplace.

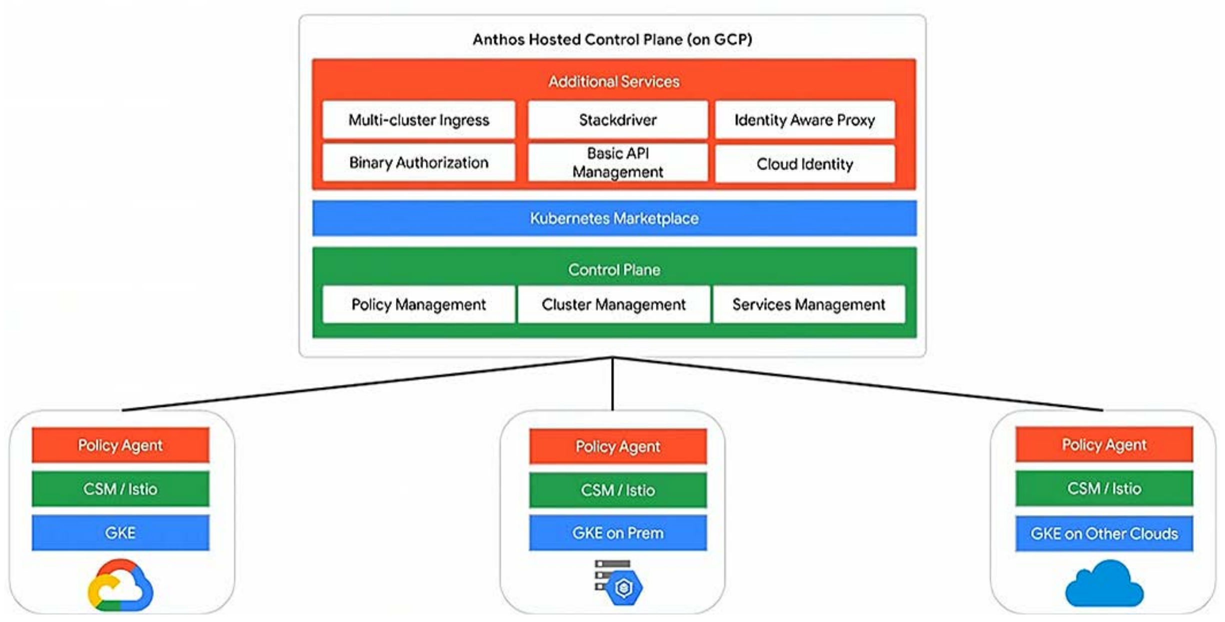


Figure 8: Anthos core multi-cloud management components

As shown in Figure 8, the centralized control plane is hosted on GCP, which provides the common UI and the core services required to connect and operate Kubernetes clusters. Below is a brief description of these components:

Multi-cluster Ingress

If you have an application running on multiple Google Kubernetes Engine clusters located in different regions, then you can route traffic to a cluster in the region closest to the user by configuring the multi-cluster ingress. The applications that need the multi-cluster ingress should be configured identically in all Kubernetes clusters with regards to the deployment configuration including the namespace, name, port number, etc.

More information on multi-cluster ingress configuration can be found here:

https://cloud.google.com/kubernetes-engine/docs/how-to/multi-cluster-ingress

Google Cloud's operations suite (formerly Stackdriver)

Google Cloud's operations suite provides powerful monitoring, logging, and diagnostics. It equips you with insight into the health, performance, and availability of cloud-powered applications, enabling you to find and fix issues faster. It is integrated with Google Cloud Platform, Amazon Web Services, and popular open source packages.

Google Cloud's operations suite combines metrics, logs, and metadata from all of your cloud accounts and projects into a single comprehensive view of your environment, enabling rapid drill-down and root cause analysis. It gives you access to logs, metrics, traces, and other signals from your infrastructure platform(s), virtual machines, containers, middleware, and application tier, so that you can track issues all the way from your end user to your backend services and infrastructure. Integration with popular services like PagerDuty and Slack provide for rapid incident response.

More information about Google Cloud's operations suite can be found here:

https://cloud.google.com/products/operations

Cloud identity and Identity Aware Proxy

Cloud identity aware proxy (IAP) provides unified access control to the workloads running on the Google cloud. With IAP, you can use centralized authentication an authorization to secure users and applications, running inside VMs and containers. Hence, IAP simplifies management of user and service identities as well as access to the workloads and resources across multiple Kubernetes clusters.

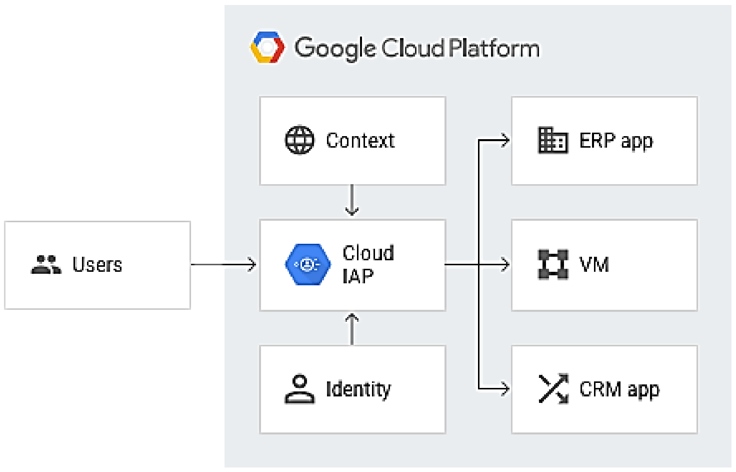


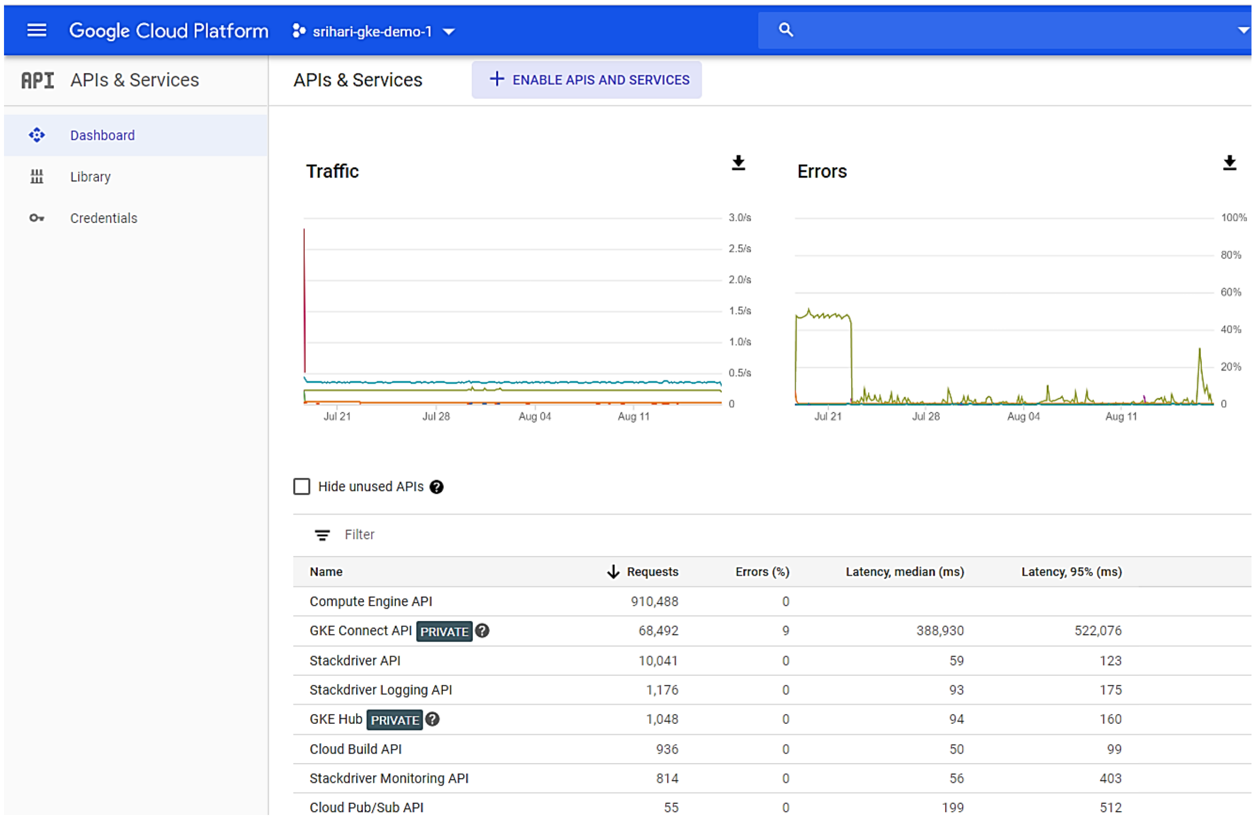
Figure 9: Google cloud identity aware proxy

More information on IAP can be found here:

https://cloud.google.com/iap/

API Management

The API management enables the cloud administrators control access to various service APIs by service agents, users, and tools. Only authorized users (or service accounts) will have access to the secured APIs. For example, in order to connect and manage the Anthos on-prem cluster via the Google cloud console, the user account should be authorized for the GKE connect API. In addition to the access control to APIs, you can also monitor the API access via the GCP dashboard for the respective APIs. Some of the monitored metrics include traffic (calls/sec), errors, latency, etc.



Name	Requests	Errors (%)	Latency, median (ms)	Latency, 95% (ms)
Compute Engine API	910,488	0		
GKE Connect API PRIVATE	68,492	9	388,930	522,076
Stackdriver API	10,041	0	59	123
Stackdriver Logging API	1,176	0	93	175
GKE Hub PRIVATE	1,048	0	94	160
Cloud Build API	936	0	50	99
Stackdriver Monitoring API	814	0	56	403
Cloud Pub/Sub API	55	0	199	512

Figure 10: Google cloud platform APIs & Services dashboard

Service mesh (Istio)

As described previously, Anthos as a developer platform enables modern application development with micro-services architecture. With micro-services, the traditional monolithic applications can be broken down into smaller and more manageable functional modules and deployed as self-contained services in the cloud. Micro-services expose APIs that other services can access. A service mesh essentially enables connectivity among the micro-services distributed across clouds.

Istio is an open source project developed by Google, IBM and others. Istio provides a scalable service mesh implementation for connecting applications running on Kubernetes. Developers do not need to modify their code to use Istio. With simple descriptive annotations as a YAML file, you can specify the service-mesh rules and apply them to running container workloads. Istio will then apply the rules to the deployed applications and start managing the security, configuration, traffic policies, etc., as defined in the configuration rules.

Istio provides the following functionality:

- Automatic load balancing for HTTP, gRPC, WebSocket, MongoDB, and TCP traffic.
- Fine-grained control of traffic behavior with rich routing rules, retries, failovers, and fault injection.
- A configurable policy layer and API supporting access controls, rate limits, and quotas.
- Automatic metrics, logs, and traces for all traffic within a cluster, including cluster ingress, and egress.
- Secure service-to-service communication in a cluster with strong identity based authentication and authorization.

You can fine more detailed information about Istio here:

https://cloud.google.com/istio/docs/istio-on-gke/overview

Anthos config management

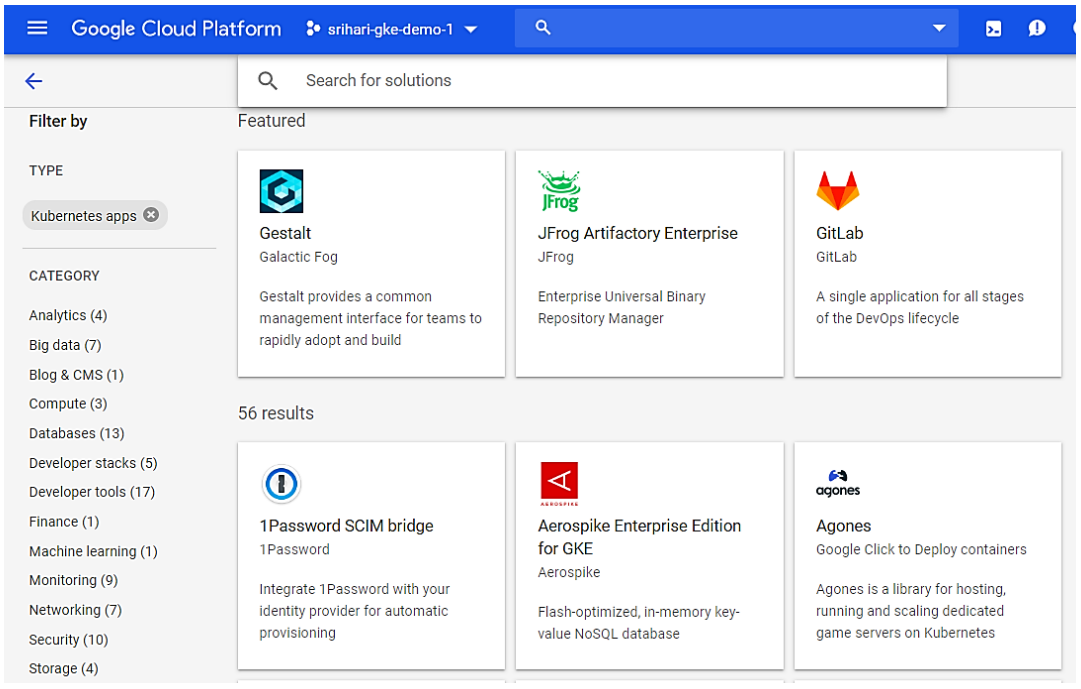
This is one of the core components of Anthos. When deploying and managing GKE clusters in multiple locations, it becomes difficult to keep all clusters in sync with respect to their configuration, security policies (RBAC), resource configurations, namespaces, and so forth. As people start using these clusters and start making configuration changes, over time you will run into "configuration drift", which results in different clusters behaving differently when the same application is deployed in different places. The Anthos Config management enables centralized configuration management via descriptive templates maintained as code in a repository. This makes it easy to ensure that you see consistent behaviour across the clusters and any deviations can be easily rectified by reverting the changes to the last known good state.

More information on config management can be found here:

https://cloud.google.com/anthos-config-management/docs/

GCP marketplace

The GCP container marketplace provides access to a large ecosystem of open source and commercial container application images that can be deployed on the GKE clusters running anywhere. With the marketplace, customers can utilize pre-created containers for common applications such as databases or web servers without having to create them on their own. Figure 11 shows the screenshot of the GCP Kubernetes apps marketplace.



Gestalt
Galactic Fog
Gestalt provides a common management interface for teams to rapidly adopt and build

JFrog Artifactory Enterprise
JFrog
Enterprise Universal Binary Repository Manager

GitLab
GitLab
A single application for all stages of the DevOps lifecycle

1Password SCIM bridge
1Password
Integrate 1Password with your identity provider for automatic provisioning

Aerospike Enterprise Edition for GKE
Aerospike
Flash-optimized, in-memory key-value NoSQL database

Agones
Google Click to Deploy containers
Agones is a library for hosting, running and scaling dedicated game servers on Kubernetes

Figure 11: Google cloud Kubernetes app marketplace

6 Operational model

As described in the previous chapters, Anthos GKE On-Prem is deployed on VMware vSAN hyperconverged infrastructure. In this reference architecture, we deployed Anthos on top of the Lenovo ThinkAgile VX vSAN platform.

Figure 12 shows Anthos GKE On-Prem deployment Architecture with the ThinkAgile VX vSAN certified nodes. The on-prem deployment consists of a single vSAN cluster with four or more servers. Each system runs the VMware vSphere 6.5U3 hypervisor host operating system. The hosts are managed via a vCenter 6.5 virtual appliance. The shared vSAN cluster provides the persistent storage via the VMFS distributed file system. The Anthos deployment consists of an Admin workstation VM, an admin GKE cluster, and one or more user GKE clusters, all implemented as virtual machines. More details on the system requirements, hardware options, and deployment steps are described in the following sections.

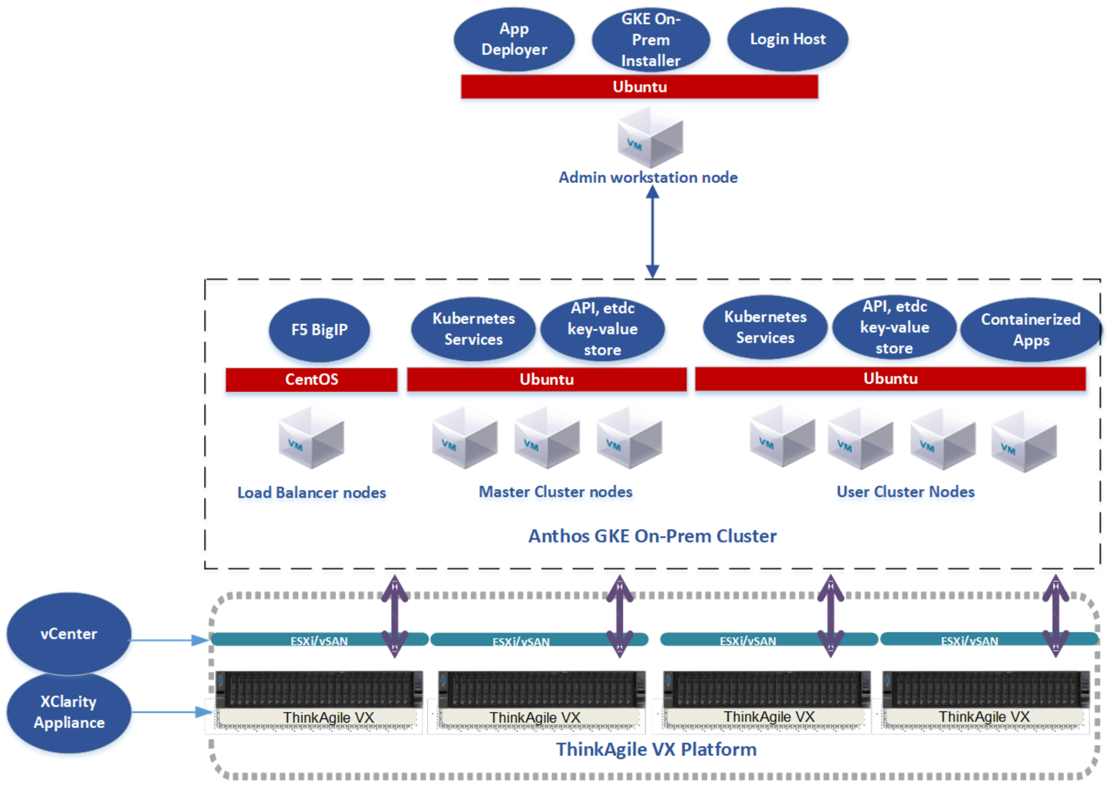


Figure 12: GKE on-prem clusters architecture on ThinkAgile VX

6.1 Hardware components

In this section we will describe the various hardware components and options to implement the GKE on-prem clusters as part of Anthos deployment.

6.1.1 VMware vSAN Hyperconverged Infrastructure (HCI)

VMware vSAN is the industry-leading HCI software, which simplifies the deployments of data center infrastructure by combining enterprise servers and storage into a single system and provides a scale-out platform to grow capacity as the application and user needs grow. By pooling capacity from locally attached disks from multiple servers forming the vSAN cluster, the storage is presented as a shared filesystem (VMFS) to all nodes.

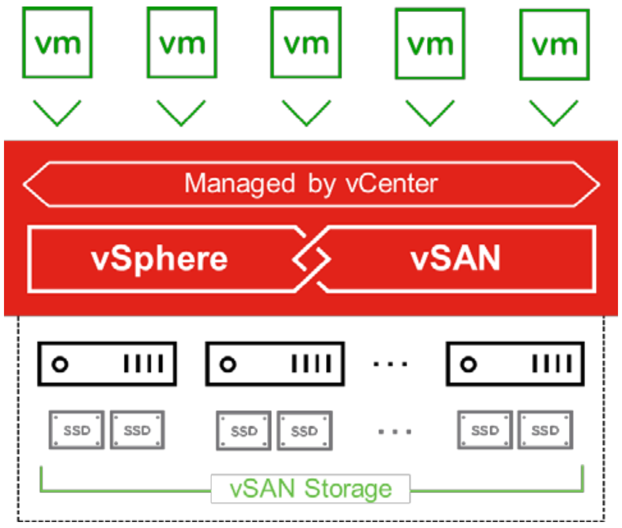


Figure 13: VMware vSAN architecture

In addition to the simplified deployment architecture, vSAN HCI provides several advantages:

- vSAN is built on top of the popular VMWare vSphere (ESXi) hypervisor operating system. Hence, applications that run virtualized on top of vSphere can directly take advantage of vSAN without any modifications or additional software requirements.
- vSAN is managed through the familiar vCenter software, which provides a single-pane-of-glass management to vSphere clusters. Hence, administrators that are already familiar with vCenter do not need to learn a new tool to manage vSAN clusters.
- Health monitoring and lifecycle management of vSAN is built into the vCenter/vSphere.
- All the enterprise-class storage features such as data replication, deduplication, compression, encryption, scaling of storage, etc., are standard.
- vSAN also supports container-native storage interface (CSI) to provide persistent storage for containers running in a Kubernetes environment.

6.1.2 Lenovo ThinkAgile VX Hyperconverged Infrastructure

Lenovo ThinkAgile VX provides certified server hardware for VMware vSAN hyperconverged infrastructure (HCI) solution. ThinkAgile VX series deliver fully validated and integrated Lenovo hardware and firmware that is certified with VMware software and preloaded with the VMware ESXi hypervisor.

The Lenovo ThinkAgile VX Series appliance arrives with the hardware configured, VMware Hyper-Converged Infrastructure (HCI) software preinstalled, and Lenovo professional services to integrate it into your environment. This makes the ThinkAgile VX Series easy to deploy and provides faster time-to-value and reduces your costs.

There are two types of ThinkAgile VX systems – **VX Certified Nodes** and **VX Series appliances**. The VX certified nodes are vSAN certified build-your-own (BYO) servers. They provide all the certified disk and I/O options for vSAN and allow most flexibility of choice for customers to configure the systems. The VX series appliances are also vSAN certified systems.

VX series appliances are optimized around specific workload use cases such as transactional databases, web, storage-rich, high-performance, etc. Hence, the VX series appliances are purpose-built vSAN certified servers. VX series comes in a wide range of platforms and provides the flexibility to configure the system you need to meet any use-case. The appliances are preloaded with VMware ESXi and preconfigured with vSAN along with license and subscriptions. Both all-flash and hybrid platforms are supported.

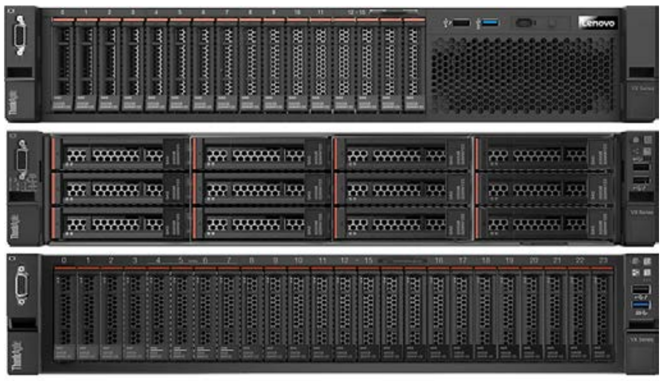


Figure 14: Lenovo ThinkAgile VX 2U Certified Node with 16x SFF (top), 12x LFF (middle), or 24x SFF (bottom) drive bays

You can find more detailed information about the various ThinkAgile VX certified nodes as well as appliances on the Lenovo press website here:

https://lenovopress.com/servers/thinkagile/vx-series

6.1.3 Lenovo ThinkAgile VX with 3rd Generation of Intel Xeon Scalable Processors

ThinkAgile VX servers with 3rd Generation of Intel Xeon Scalable Processors provide up to 40 cores per processor, 4TB memory per server and support for the new PCIe 4.0 standard for I/O, the VX systems offer the ultimate in two-socket performance in a 1U/2U form factors. The 1U servers are suitable for compute heavy and 2U servers are scalable for compute, storage and inference workloads with support for more drives and GPUs.

Table 4: ThinkAgile VX 1U Appliances and Certified Noded

	VX2330	**VX3330**	**VX7330-N**	**VX3331**
VX offering type	Appliance	Appliance	Appliance	Certified Node

	VX2330	**VX3330**	**VX7330-N**	**VX3331**
Target workloads	SMB	Compute heavy	High performance	Compute heavy
Base MTM	7Z62CTO1WW	7Z62CTO2WW	7Z62CTO3WW	7Z62CTO4WW
Base platform	SR630 V2	SR630 V2	SR630 V2	SR630 V2
CPU	1x or 2x Intel Xeon SP Gen 3	1x or 2x Intel Xeon SP Gen 3	1x or 2x Intel Xeon SP Gen 3	1x or 2x Intel Xeon SP Gen 3
Memory	32x DDR4 3200 MHz (4TB maximum)	32x DDR4 3200 MHz (4TB maximum)	32x DDR4 3200 MHz (4TB maximum)	32x DDR4 3200 MHz (4TB maximum)
Drive Bays	4x 3.5" SAS/SATA 4x 3.5" NVMe	12x 2.5" SAS/SATA 12x 2.5" NVMe	12x 2.5" NVMe	12x 2.5" SAS/SATA 12x 2.5" NVMe 4x 3.5" SAS/SATA
Drive configurations	All Flash Hybrid	All Flash Hybrid	All Flash	All Flash Hybrid
Disk Groups	1	1 - 4	1 - 4	1 - 4
GPUs	No support	No support	No support	3x SW GPU 75W each

Table 5: ThinkAgile VX 2U Appliances and Certified Nodes

	VX3530-G	**VX5530**	**VX7530**	**VX7531**
VX offering type	Appliance	Appliance	Appliance	Certified Node
Target workloads	Compute heavy	Storage heavy	High performance	High performance
Base MTM	7Z63CTO2WW	7Z63CTO3WW	7Z63CTO4WW	7Z63CTO5WW
Base platform	SR650 V2	SR650 V2	SR650 V2	SR650 V2

CPU	1x or 2x Intel Xeon SP Gen 3	1x or 2x Intel Xeon SP Gen 3	1x or 2x Intel Xeon SP Gen 3	1x or 2x Intel Xeon SP Gen 3
Memory	32x DDR4 3200 MHz (4TB maximum)	32x DDR4 3200 MHz (4TB maximum)	32x DDR4 3200 MHz (4TB maximum)	32x DDR4 3200 MHz (4TB maximum)
Drive Bays	24 x 2.5" (HS)	16 x 3.5" (HS)	40x 2.5" SAS/SATA 32x 2.5" NVMe	40x 2.5" SAS/SATA 32x 2.5" NVMe 16x 3.5" SAS/SATA
Drive configurations	All Flash Hybrid	All Flash Hybrid	All Flash Hybrid	All Flash Hybrid
Disk groups	1 - 5	1 - 5	1 - 5	1 - 5
GPUs	3x DW GPU 300W each 8x SW GPU 75W each 6x SW GPU 150W each	No support	No support	3x DW GPU 300W each 8x SW GPU 75W each 6x SW GPU 150W each

6.2 Persistent Storage for GKE on-prem Clusters

There are two types of storage consumed by containerized applications – ephemeral (non-persistent) and persistent. As the names suggest, non-persistent storage is created and destroyed along with the container and is only used by applications during their lifetime as a container. Hence, non-persistent storage is used for temporary data. When implementing the Kubernetes Platform, local disk space on the application nodes can be configured and used for the non-persistent storage volumes.

Persistent storage, on the other hand, is used for data that needs to be persisted across container instantiations. An example is a 2 or 3-tier application that has separate containers for the web and business logic tier and the database tier. The web and business logic tier can be scaled out using multiple containers for high availability. The database that is used in the database tier requires persistent storage that is not destroyed.

Kubernetes uses a persistent volume framework that operates on two concepts – persistent storage and persistent volume claim. Persistent storage are the physical storage volumes that are created and managed by the Kubernetes cluster administrator. When an application container requires persistent storage, it would create a persistent volume claim (PVC). The PVC is a unique pointer/handle to a persistent volume on the physical storage, except that PVC is not bound to a physical volume. When a container makes a PVC request, Kubernetes would allocate the physical disk and binds it to the PVC. When the container image is destroyed, the volume bound to the PVC is not destroyed unless you explicitly destroy that volume. In addition, during the lifecycle of the container if it relocates to another physical server in the cluster, the PVC binding will still be maintained. After the container image is destroyed, the PVC is released, but the persisted storage volume is not deleted. The specific persistent storage policy for the volume will determine when the volume gets deleted.

As described previously, the GKE on-prem clusters are installed on a VMware vSAN HCI platform, which provides both the compute and storage capacity for the workloads. Since the GKE clusters are deployed as virtual machines on vSphere, they will have direct access to the vSAN cluster storage or SAN based data stores if a SAN storage environment is used.

As shown in Figure 15, the nodes in the GKE cluster run as virtual machines with the corresponding virtual disks attached to them, which are stored on the shared vSAN datastore. When the Kubernetes pods running within the nodes make persistent volume claim requests, the data is persisted as part of the VM's virtual disks, which are the VMDK files. This makes management of persistent volume stores and PVCs much easier from a Kubernetes administration standpoint.

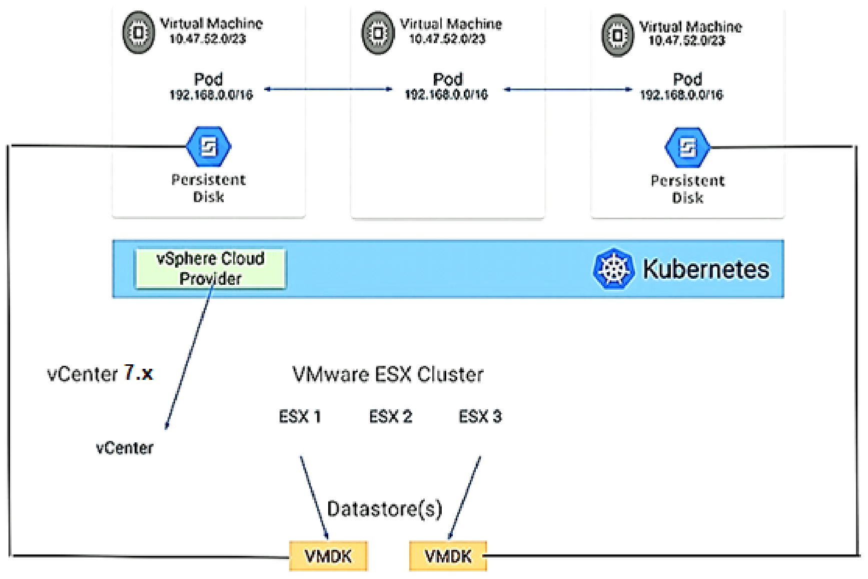
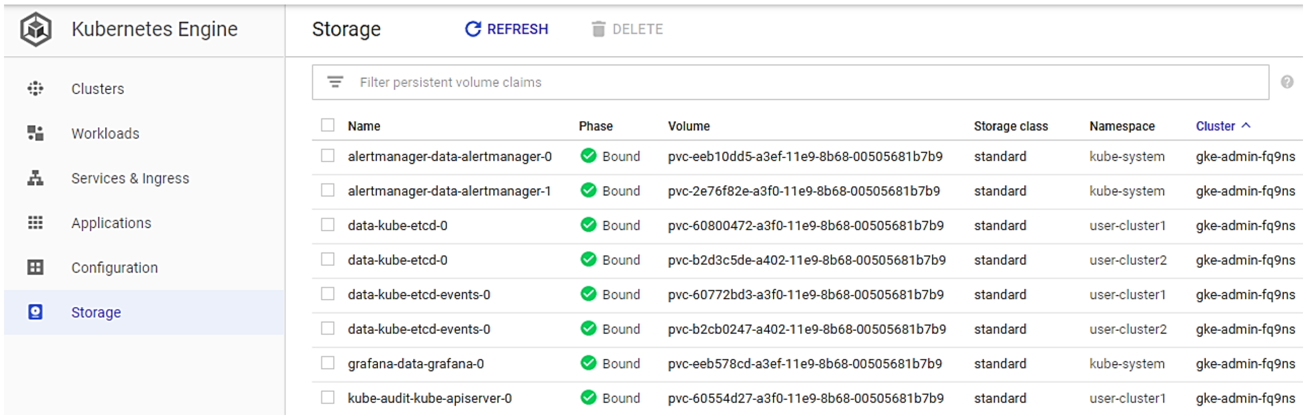


Figure 15: Persistent storage for Kubernetes with VMware vSAN



Name	Phase	Volume	Storage class	Namespace	Cluster
alertmanager-data-alertmanager-0	Bound	pvc-eeb10dd5-a3ef-11e9-8b68-00505681b7b9	standard	kube-system	gke-admin-fq9ns
alertmanager-data-alertmanager-1	Bound	pvc-2e76f82e-a3f0-11e9-8b68-00505681b7b9	standard	kube-system	gke-admin-fq9ns
data-kube-etcd-0	Bound	pvc-60800472-a3f0-11e9-8b68-00505681b7b9	standard	user-cluster1	gke-admin-fq9ns
data-kube-etcd-0	Bound	pvc-b2d3c5de-a402-11e9-8b68-00505681b7b9	standard	user-cluster2	gke-admin-fq9ns
data-kube-etcd-events-0	Bound	pvc-60772bd3-a3f0-11e9-8b68-00505681b7b9	standard	user-cluster1	gke-admin-fq9ns
data-kube-etcd-events-0	Bound	pvc-b2cb0247-a402-11e9-8b68-00505681b7b9	standard	user-cluster2	gke-admin-fq9ns
grafana-data-grafana-0	Bound	pvc-eeb578cd-a3ef-11e9-8b68-00505681b7b9	standard	kube-system	gke-admin-fq9ns
kube-audit-kube-apiserver-0	Bound	pvc-60554d27-a3f0-11e9-8b68-00505681b7b9	standard	user-cluster1	gke-admin-fq9ns

Figure 16: Kubernetes persistent volume claims

6.3 Networking

There are three logical networks defined in this RA:

- **External**: The external network is used for the internet access to the clusters, ingress to the exposed applications (services and routes). Anthos in its current release requires a layer 4 load-balancer to support traffic routing across the internal and external networks, as well as the communication across the on-prem GKE clusters.
- **Internal**: This is the primary, non-routable network used for cluster management and inter-node communication. Domain Name Servers (DNS) and Dynamic Host Configuration Protocol (DHCP) services also reside on this network to provide the functionality necessary for the deployment process and the cluster to work. Communication with the Internet is handled by the F5 gateway, which runs as a separate virtual appliance on the VMware cluster.
- **Out-of-band network**: This is a secured and isolated network used for switch and server hardware management, such as access to the xClarity Controller (XCC) module on the servers and SoL (Serial-over-LAN).

6.3.1 Network redundancy

The Anthos deployment on the ThinkAgile VX vSAN platform uses 10GbE network as the primary fabric for inter-node communication. Two ToR switches are used to provide redundant data layer communication and deliver maximum network availability. The typical deployment architecture for this setup is shown in Figure 17.

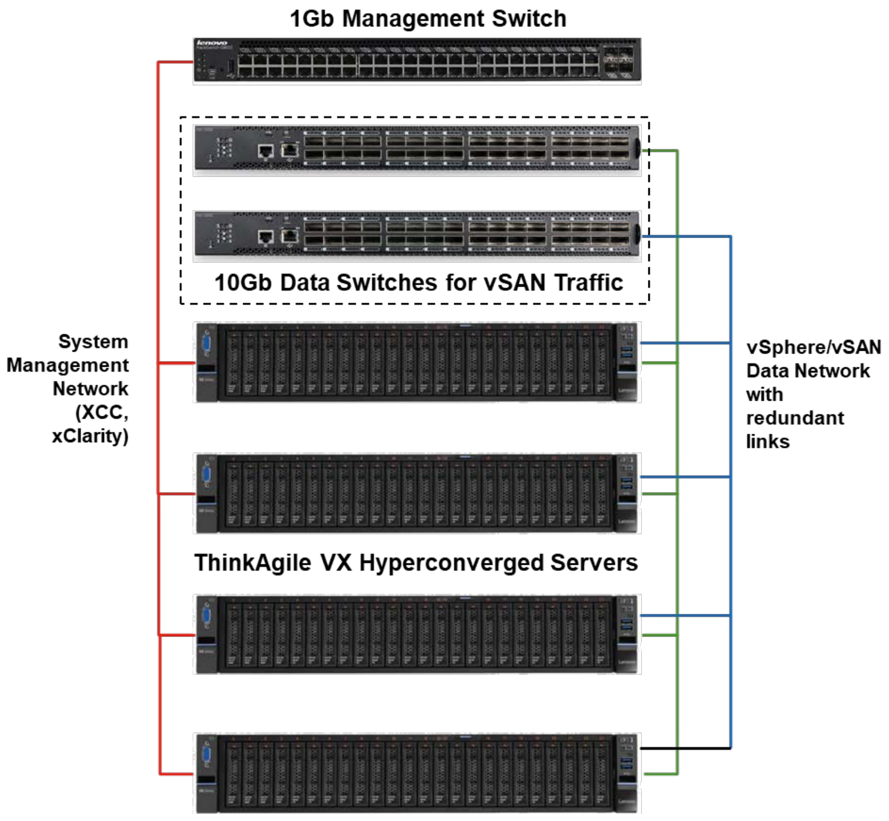

1Gb Management Switch
10Gb Data Switches for vSAN Traffic
System Management Network (XCC, xClarity)
vSphere/vSAN Data Network with redundant links
ThinkAgile VX Hyperconverged Servers

Figure 17: ThinkAgile VX network connectivity

The two primary network fabrics shown in the diagram are the systems management network and the internal data/user network. Typically, 1Gbps Ethernet is sufficient for the systems management network, which provides out-of-band access to the on-board management processors on the servers and network switches. The data/cluster internal fabric is recommended to be 10Gbps Ethernet. This fabric is also recommended to have redundant switches for high-availability of the network fabric. The Lenovo ThinkSystem network switches support the Cloud Network Operating System (CNOS), which provides advanced data center networking features including virtual link aggregation (VLAG).

Figure 18 shows the redundant network architecture and the VLAG configuration.

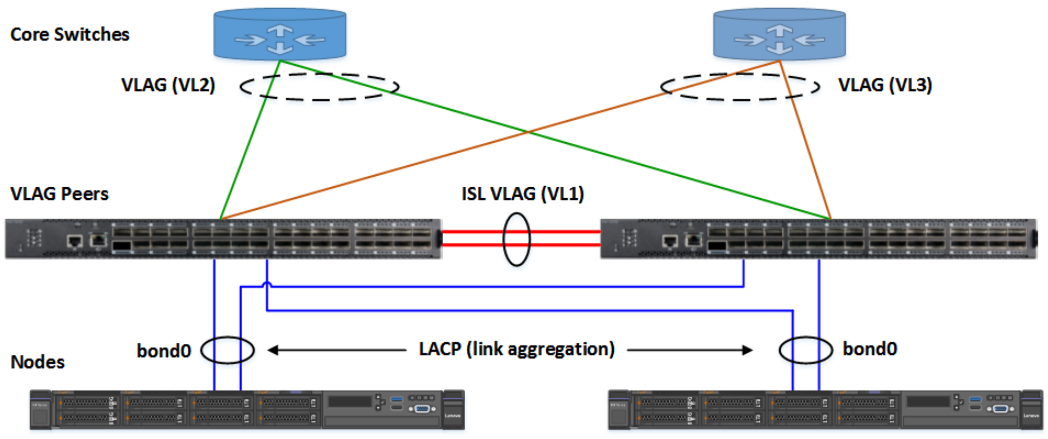


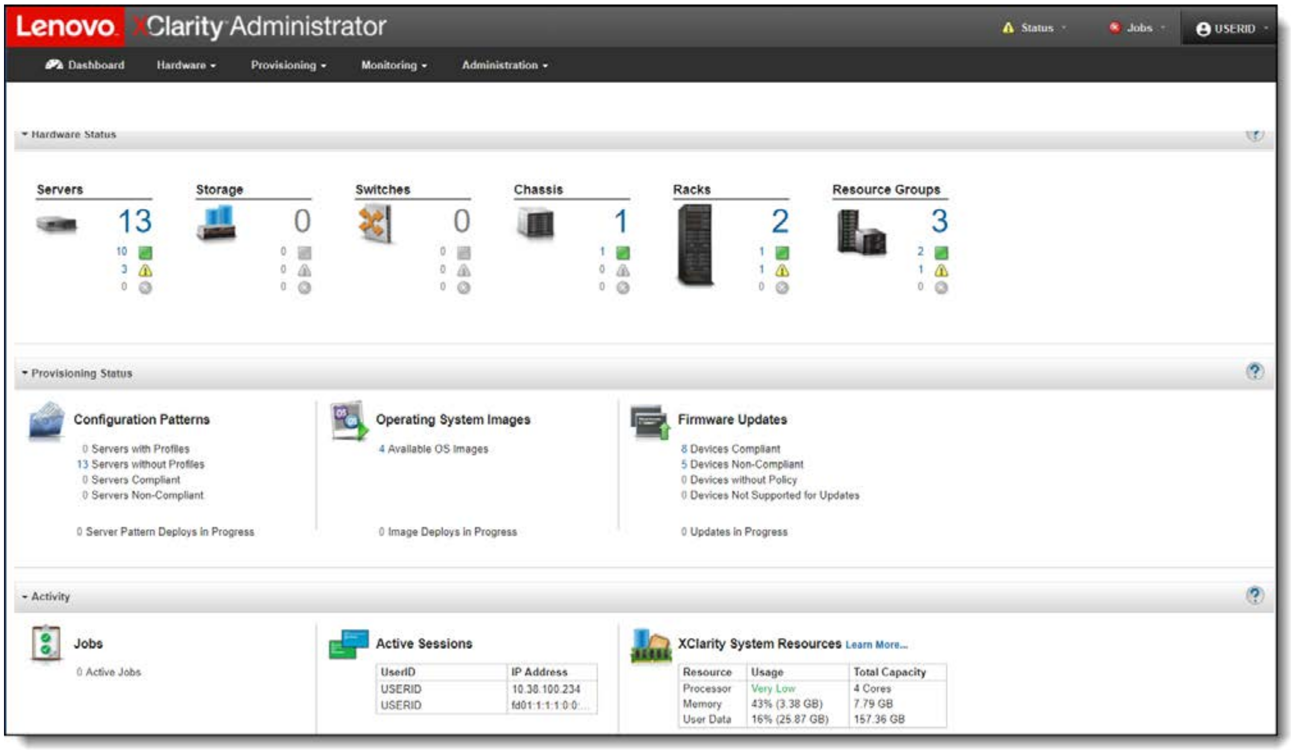
Figure 18: Network fabric redundancy and VLAG

Virtual Link Aggregation Group (VLAG) is a feature allows a pair of leaf switches to work as a single virtual switch. Each of the cluster nodes has a link to each VLAG peer switch for redundancy. This provides high availability (HA) for the nodes using the link aggregation control protocol (LACP) for aggregated bandwidth capacity. Connection to the uplink core network is facilitated by the VLAG peers, which present a logical switch to the uplink network, enabling connectivity with all links active and without a hard requirement for spanning-tree protocol (STP). The link between the two VLAG peers is an inter-switch link (ISL) and provides excellent support of east-west cluster traffic the nodes. The VLAG presents a flexible basis for interconnecting to the uplink/core network, ensures the active usage of all available links, and provides high availability in case of a switch failure or a required maintenance outage.

6.3.2 Systems management

The Lenovo XClarity Administrator software provides centralized resource management that reduces management complexity, speeds up response, and enhances the availability of Lenovo® server systems and solutions.

The Lenovo XClarity Administrator provides agent-free hardware management for Lenovo's ThinkSystem® rack servers, System x® rack servers, and Flex System™ compute nodes and components, including the Chassis Management Module (CMM) and Flex System I/O modules. Figure 19 shows the Lenovo XClarity administrator interface, in which Flex System components and rack servers are managed and are seen on the dashboard. Lenovo XClarity Administrator is a virtual appliance that is quickly imported into a virtualized environment server configuration.



Servers	Storage	Switches	Chassis	Racks	Resource Groups
13	0	0	1	2	3
10 🟩	0	0	1 🟩	1 🟩	2 🟩
3 ⚠	0 ⚠	0 ⚠	0 ⚠	1 ⚠	1 ⚠
0 ●	0 ●	0 ●	0 ●	0 ●	0 ●

Configuration Patterns

0 Servers with Profiles
13 Servers without Profiles
0 Servers Compliant
0 Servers Non-Compliant

0 Server Pattern Deploys in Progress

Operating System Images

4 Available OS Images

0 Image Deploys in Progress

Firmware Updates

8 Devices Compliant
5 Devices Non-Compliant
0 Devices without Policy
0 Devices Not Supported for Updates

0 Updates in Progress

Jobs

0 Active Jobs

Active Sessions

UserID	IP Address
USERID	10.38.100.234
USERID	fd01:1:1:1:0:0:...

XClarity System Resources Learn More...

Resource	Usage	Total Capacity
Processor	Very Low	4 Cores
Memory	43% (3.38 GB)	7.79 GB
User Data	16% (25.87 GB)	157.36 GB

Figure 19. Lenovo XClarity Administrator Dashboard

For more information, see: [Lenovo XClarity Administrator Product Guide]()

6.4 Deployment of Anthos GKE On-prem Clusters

6.4.1 New features in Anthos 1.12.0

For a detailed list of release notes for Google Cloud's Anthos versions, see the following URL:

https://cloud.google.com/anthos/gke/docs/on-prem/release-notes

Some of the key new features since earlier Anthos 1.x releases are the following.

- **Workload Identity for GKE on-prem**
  Workload Identity enables secure access to Google Cloud services from within GKE. Workloads running on GKE must authenticate to use Google Cloud APIs such as the Compute APIs, Storage and Database APIs, or Machine Learning APIs. With Workload Identity, you configure a Kubernetes service account (KSA) to act as a Google service account (GSA). Any workload running as the KSA automatically authenticates as the GSA when accessing Google Cloud APIs.
  More information about Workload Identity can be found here:
  https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity
- **Support for vSAN datastore as storage for admin and user clusters**
  Includes support for datastores on vSAN or standard SAN based datastores
- **Support for Seesaw loadbalancer**
  Starting with Anthos 12.0 release, GKE on-prem provides and manages the Seesaw load balancer, which is an open source alternative to the F5 BIGIP loadbalancer.
  https://github.com/google/seesaw
- **Support for up to 100 nodes per user cluster**
- **Windows Server container support**
  This allows organizations to run Windows-based apps more efficiently in your data centers without having to perform application rewrites. Use Windows containers alongside Linux containers for your container workloads.
- **Automated cluster scaling is available**
  With cluster autoscaling, it's possible to horizontally scale node pools per workload demand. When demand is higher, the autoscaler adds nodes to the pool. When demand is low, the autoscaler removes nodes from the pool to a minimum size that you designate. This can increase the availability of your workloads while lowering costs.
- **There is another bundled load balancer option available**
  MetalLB is now available as another bundled software load balancer in addition to Seesaw.
- **User cluster control-plane node and admin cluster add-on node auto sizing is supported**
- **Use the Cloud console to administer Anthos clusters**
  Create, update, and delete Anthos on VMware user clusters.
- **NOTE: vSphere versions lower than v7.0 Update 2 are deprecated in Kubernetes 1.24**

VMware's General Support for vSphere 6.7 will end on October 15, 2022. Customers are recommended to upgrade vSphere (both ESXi and vCenter) to version 7.0 Update 2 or above.

- **Starting with Anthos v 1.7.0, Anthos clusters supported on vSphere 7.0**
  Deploy Anthos clusters in v7.0 environments in addition to vSphere 6.5 and 6.7
- **Deploy GKE clusters in separate vSphere clusters**
  With this feature, it's possible to deploy the admin cluster in one vSphere cluster, and a user cluster in a separate vSphere cluster.
- **Kubernetes service account (KSA) Signing Key rotation**
  This is now supported on user clusters as well as the admin cluster.
- **Storing credentials for user clusters as Kubernetes Secrets**
  Users can prepare credentials for the user cluster and store them as Kubernetes Secrets in the admin cluster before a user cluster is created. When creating a user cluster, the prepared credentials will be used.
- **User cluster Nodepools**
  A node pool is a group of nodes within a cluster that all have the same configuration. This feature lets users create multiple node pools in a cluster and update them as needed. You can manage a pool of nodes separately without affecting any of the other nodes in each cluster.
  https://cloud.google.com/anthos/gke/docs/on-prem/how-to/managing-node-pools

6.4.2 Deployment pre-requisites

In order to perform initial configuration and installation of the Anthos GKE On-Prem clusters, you need to apply for a Google cloud platform (GCP) account and create a GCP project. There is also a free trial account available to test a deployment in a lab setting. After an account is established, you can download the Anthos related deployment images as the VMware OVA templates. In addition to the Anthos images, you also need to download some of the Google tools necessary for the deployment automation and cluster administration tasks. Specifically, you would need to download and install the Google cloud SDK, govc utility (which is a CLI to interact with vCenter), and Terraform, which is the deployment automation tool for the base VM images for Anthos and works with vCenter APIs.

More detailed instructions to prepare for Anthos deployment can be found here:

https://cloud.google.com/gke-on-prem/docs/how-to/installation/getting-started

6.4.3 Deployment considerations

The hardware required for the GKE on-prem clusters will depend upon the specific workload and user requirements. Hence, the sizing of the cluster will vary based on the types of workloads deployed, performance and scalability requirements, number of container images expected to run, the deployment type – test and development, staging, and production, etc. There two levels of sizing that need to be performed for Anthos. Since GKE on-prem runs as a virtual machine cluster, the actual container pods of Kubernetes execute inside the virtual machines. Hence, you need to determine the right size of the virtual machines used for worker nodes of Kubernetes such as the number of vCPUs, vRAM, virtual disk, etc. For example, if you anticipate a production deployment of Kubernetes and the workloads are typical enterprise applications with multiple tiers such as web, business logic, database, etc., then you may need to choose the worker nodes with a good number of vCPUs and virtual memory. Whereas for a test/dev type environment your worker VMs could small.

The second tier of sizing is the physical hardware sizing. Anthos allows you to deploy multiple user clusters on top of the same VMware vSAN cluster. Hence, you need to determine how many GKE clusters you would install, and how many worker VMs in each cluster, the individual resource requirements as the vCPUs and vRAM, and then aggregate the total to determine what kind of physical resources you would need to implement the clusters. This translates to a number of vSAN servers with specific physical CPU cores, core speed, physical memory, and disk.

In this reference architecture, we provided three recommended hardware configurations based on a rough workload profile estimate:

i. An entry configuration for test and development environments
ii. A mid-range configuration for common web and other lightweight workloads
iii. A high-performance configuration for production, mission-critical workloads such as large-scale micro-services, transactional databases, etc.

These configuration bill of materials are provided in the appendix.

6.4.4 Anthos GKE On-Prem Configuration

As described in previous sections, the GKE on-prem deployment consists of three types of virtual machines:

Admin workstation: Used for the rest of the cluster deployment. This is a Google provided VM template.

Admin cluster: Provides the administrative control plane for all on-prem clusters and for connectivity to Google cloud.

User clusters: Kubernetes clusters for running user workloads.

Table lists the minimum hardware requirements for these different virtual machines.

Table 6: Admin cluster minimum requirements

Name	Specifications	Purpose
Admin cluster master	• 4 vCPU • 16384 MB RAM • 40 GB hard disk space	Runs the admin control plane.
Add-ons VMs	Two VMs running with the following specifications: • 4 vCPU • 16384 MB RAM • 40 GB hard disk space	Run the admin control plane's add-ons.
User cluster	• 4 vCPU	Each user cluster has its own control plane. User control

master	• 8192 MB RAM • 40 GB hard disk space	plane VMs run in the admin cluster. You can choose to create one or three user control planes. If you choose to create three user control planes, GKE On-Prem creates three VMs—one for each control plane—with these specifications.

Table 7: User clusters minimum requirements

Name	Specifications	Purpose
User cluster worker nodes	• 4 vCPU • 8192 MB RAM • 40 GB hard disk space	A user cluster "node" (also called a "machine") is a virtual machine where workloads run. When you create a user cluster, you decide how many nodes it should run. The configuration required for each node depends on the workloads you run. For information on the maximum number of clusters and nodes you can create, see Quotas and limits. You can add or remove VMs from an existing user cluster. See Resizing a Cluster.

See the following Google document for more detailed hardware requirements for Anthos:

https://cloud.google.com/gke-on-prem/docs/how-to/installation/requirements

6.4.5 Anthos Deployment Example

In this deployment example, we start with a minimal hardware configuration for demonstration with 3 servers and 2 switches. Together with the admin workstation, GKE admin and user clusters, and the F5 BIG IP load-balancer appliance, we deploy 9 virtual machines.

The hardware consists of:

- 3x Lenovo ThinkAgile VX servers

The 9 VM nodes are as follows:

- 1 admin workstation node
- 1 load balancer nodes
- 3 admin cluster nodes (1 admin master node, 2 admin add-on nodes)
- 4 user cluster nodes (1 user master node, 3 user worker nodes)

The resource configuration of the various VMs is summarized in table.

Table 8. Node Configuration

Node	CPU	Memory	Hard Disk	Network Adapter
Admin workstation	4 vCPU(s)	16GB	50GB	1 VMXNET3 NIC

Load balancer	4 vCPU(s)	8GB	40TB	1 VMXNET3 NIC
Admin cluster	4 vCPU(s)	16GB	50GB	1 VMXNET3 NIC
User cluster	4 vCPU(s)	8GB	40GB	1 VMXNET3 NIC

The cluster administrator can scale up the clusters later by adding more user nodes or create additional user clusters based on requirements.

The high-level cluster architecture for this example implementation is shown in Figure 20.

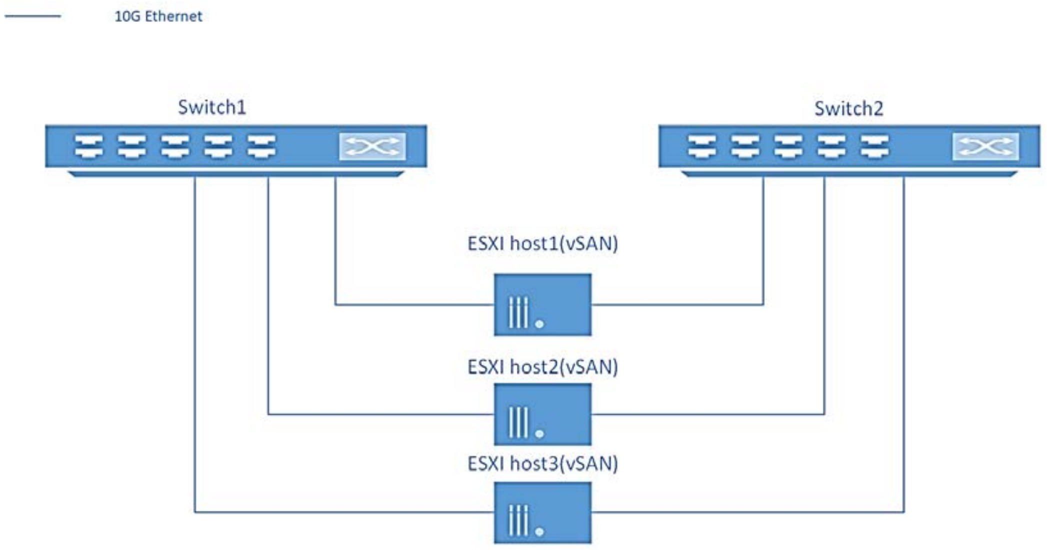


Figure 20: 3-node ESXi cluster topology

6.2.1 High-level Deployment Steps

The installation uses the Google Cloud CLI and GKEADM tool to setup the installation environment and deploy the admin workstation image. All subsequent setup tasks are completed from this admin workstation VM.

Below are the high-level steps to deploy the Anthos solution.

1. On the VX7531 nodes, change UEFI processor mwait setting to **enabled**. By default, it is disabled. It must be in custom mode to allow this setting change. This is a VMware requirement to allow the VMware CLS system VMs to start.
2. Setup the VMware vSphere platform. This includes loading ESXi on the VX7531 certified nodes, and a vSphere vCenter server which runs on a VM now as of version 7.0.
3. Setup the vSAN data stores or SAN based shared storage accessible by all ESXi hosts
4. Plan IP addresses and networking. Either from a DHCP server scope, or a list of allowed static IP addresses. Provide a DNS server and default gateway.
5. Configure vSphere resource levels such as the virtual data center, resource pool, network and data store.

6. Setup Google Cloud resources, including install the Google Cloud CLI, create a Google Cloud project, and create Google Cloud service accounts with permissions.

Note: Google's online documentation walks you through these steps and includes the Linux commands and cluster config file templates needed to accomplish each one. See this link for the latest install details: https://cloud.google.com/anthos/clusters/docs/on-prem/latest/how-to/install-overview

6. Create the admin workstation. This is a Google provided VMware template that creates a VM to use for the remaining deployment steps.
7. Create the admin cluster load balancer VM. Google includes a bundled VM-based load balancer called Seesaw, which works fine.
8. Create an admin cluster. This cluster runs the Kubernetes control plane for itself and any associated user clusters.
9. Create the user cluster load balancer VM. Use the Seesaw one.
10. Create user clusters. These clusters are made up of worker nodes and run the workload containers.
11. Deploy a sample workload on a user cluster. This includes deploying an application, creating a service and an ingress for it.

6.2.2 Production Anthos GKE On-Prem Topology

For a production level GKE on-prem implementation you need to consider system reliability, availability, performance, and scalability requirements. Since technically there is no limit for scaling the GKE clusters, you will be only limited by the underlying infrastructure capabilities. VMware vSAN can scale up to 64 physical hosts in a single vSAN cluster. This scale provides a lot of capacity, both from the compute and storage perspective. The sweet spot production cluster deployments are between 4 and 16 nodes. When designing the vSAN clusters for production environments you should follow the best practices for data redundancy, availability, and disaster recovery.

See the following vSAN design guide for additional detail:

https://storagehub.vmware.com/t/vmware-vsan/vmware-r-vsan-tm-design-and-sizing-guide-2/version-6-5/

To deliver high performance for mission-critical workloads, consider vSAN All-flash node configurations which use solid state disks (SSDs) to deliver high IOPS and throughput. The high-performance configuration BOM in the appendix is based on all-flash vSAN.

Figure 21 shows the network architecture for a production level vSAN cluster for Anthos deployment. As you will see in the picture, the data and user network fabric uses redundant 10Gbps Ethernet switches. The nodes each have two 10Gbps ports connected into the fabric for redundancy as well as aggregation of the ports to deliver 20Gbps bandwidth. The aggregation configuration can be done in vSphere. The switches also have an ISL VLAG across them, which essentially makes the two switches act as a single logical switch for the downstream links. If you lose one of the switches, the nodes will still have the other port active without any disruption to the network traffic. In addition, there is a 1Gbps switch used for out-of-band management access to the hardware for functions such as remote power management, event/alerts, firmware updates, etc. In some production environments, it’s also common to separate the in-band (operating system) management traffic from the out-of-band management traffic, which requires an additional switch.

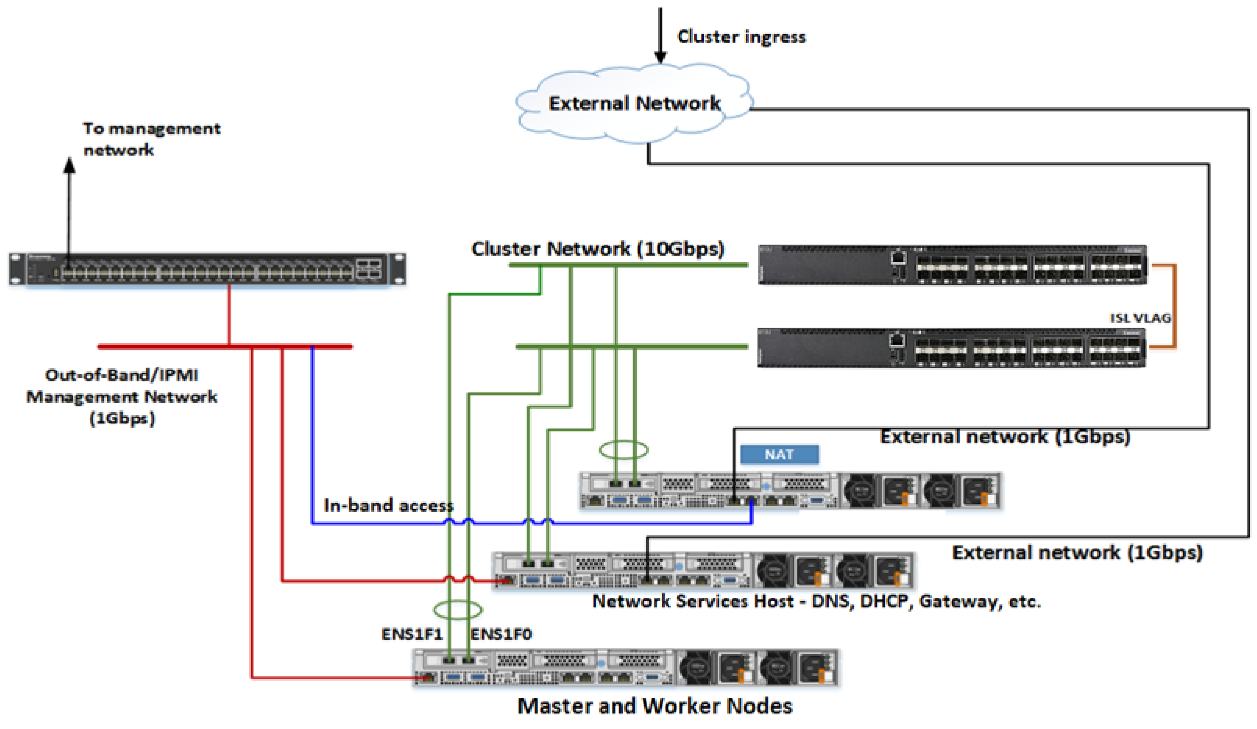


Figure 21: Production Anthos cluster network topology

6.4.6 Logical network architecture

From a logical architecture perspective, the network is segmented into different traffic groups (see Figure 22):

External network:

As the name suggests, this network connects the GKE on-prem clusters into the customer’s campus network as well as the internet. The external network traffic is only allowed to access services inside the GKE on-prem clusters via the ingress routing provided by the F5 BIG IP load-balancer. In addition, you will need a gateway host that provides NAT based access to the GKE cluster nodes to access the internet during the initial deployment phase and optionally to provide access to the internet to running pods. This will be a unidirectional link. When the on-prem GKE clusters are connected to the GCP console, a TLS connection is established from the GCP to the admin Kubernetes cluster through the gateway.

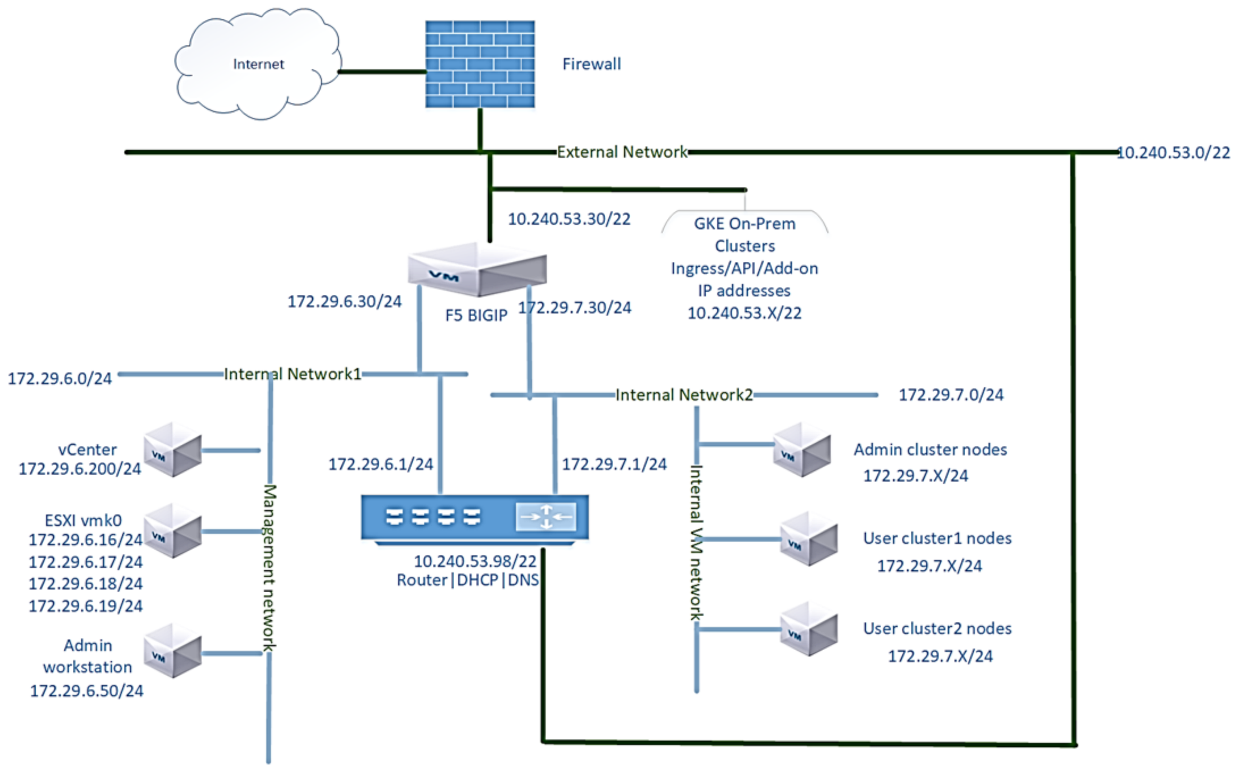

Internet
Firewall
External Network
10.240.53.0/22
10.240.53.30/22
GKE On-Prem Clusters Ingress/API/Add-on IP addresses 10.240.53.X/22
172.29.6.30/24
F5 BIGIP
172.29.7.30/24
172.29.6.0/24
Internal Network1
Internal Network2
172.29.7.0/24
vCenter 172.29.6.200/24
172.29.6.1/24
172.29.7.1/24
Admin cluster nodes 172.29.7.X/24
ESXI vmk0 172.29.6.16/24 172.29.6.17/24 172.29.6.18/24 172.29.6.19/24
Management network
10.240.53.98/22 Router|DHCP|DNS
Internal VM network
User cluster1 nodes 172.29.7.X/24
Admin workstation 172.29.6.50/24
User cluster2 nodes 172.29.7.X/24

Figure 22: Logical network architecture and network segmentation

Internal VM network:

This is a private network segment used for the management network across the GKE cluster virtual machines (admin and worker nodes). The Kubernetes cluster management and API traffic is accessible over their network segment. The IP addresses for this segment can be assigned statically or via a DHCP server running on the network. It's recommended to isolate the VM network segment for each of the user clusters into its own VLAN on the virtual switch so that the traffic is isolated across the different clusters. The deployment definition file for the user cluster should specify the IP addresses used for various API end-points.

Internal management network:

This is also a private network segment similar to the VM network segment. This management network is for communication across the vSphere ESXi hosts, vCenter, and the GKE on-prem admin workstation.

More detailed network configuration and F5 BIG IP load-balancer requirements can be found here:

https://cloud.google.com/gke-on-prem/docs/how-to/installation/requirements#f5_big-ip_requirements

7 Deployment Examples and Considerations

This section is used to describe noteworthy deployment considerations. In particular it can be used to describe how features of the hardware is used in the solution deployment or in conjunction with other data center roles. This could include considerations such as: high availability, backup, etc and how the Lenovo/ISV reference architecture enables these capabilities.

This section also may include a high-level overview of the requirements the customers IT environment must address for deploying this reference architecture.

7.1 Anthos hybrid and multi-cloud management

Hybrid cloud enables organizations to take advantage of public cloud capabilities while retaining some part of their application landscape and data to reside within their on-prem data centers. Many organizations are implementing hybrid clouds for a variety of reasons – data privacy and security, service level agreements (SLAs), cost concerns, proprietary applications, user needs, and so forth. With a hybrid cloud approach, companies can selectively migrate workloads and data on-demand to the public cloud providers such as Google cloud platform, AWS, Azure, etc. This is also known as bursting of traffic on-demand to the public cloud.

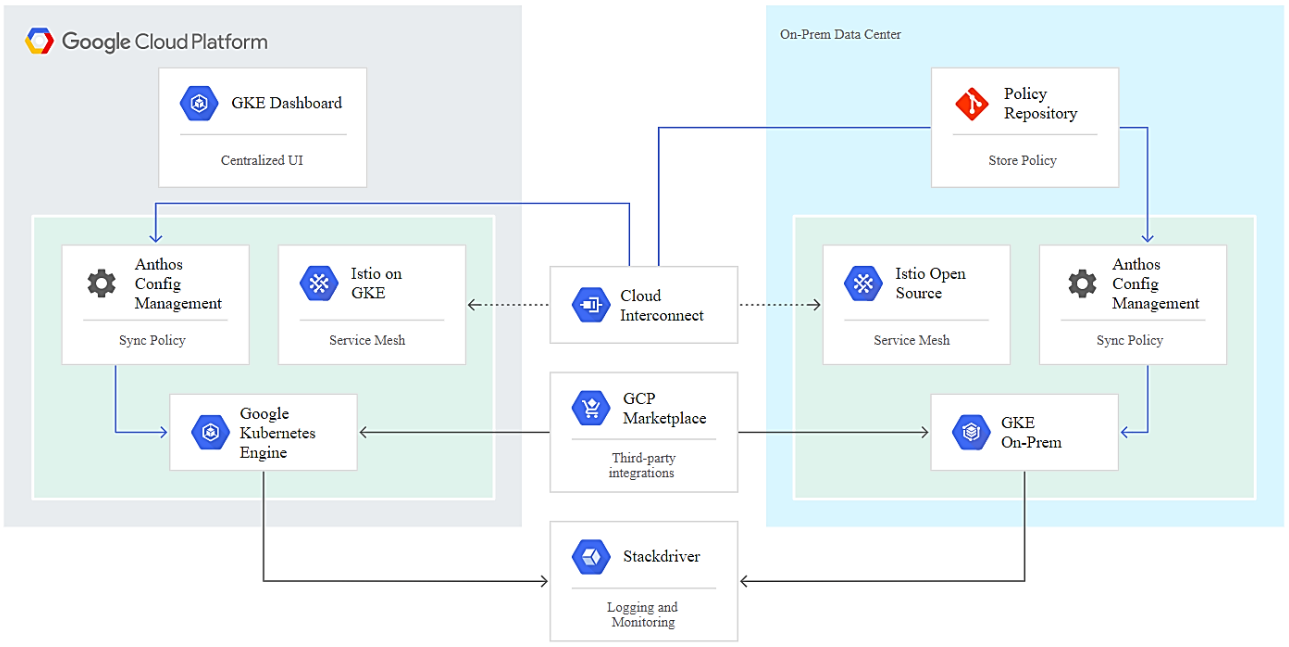


Figure 23: Anthos hybrid cloud architecture components

Hybrid cloud implementations tend to be complex because of the integration required between the on-prem data center and the public cloud data centers, concerns with network/internet security, complex bi-directional traffic routing, data access and provisioning requirements, etc. Hence, hybrid cloud implementations tend to require several third party tools and services depending upon the specific capabilities expected. Google Cloud's Anthos simplifies hybrid cloud by providing the necessary tools and services for a secure and scalable hybrid cloud implementation.

Figure 23 shows the core pieces of the hybrid cloud architecture with google cloud and Anthos. You will notice that the components that are running on the Google public cloud are largely the same components running on the on-prem cloud. Hence, the Anthos clusters running in the on-prem data center are essentially an extension of the public cloud. This is why the Google hybrid cloud approach is different from other vendors. The consistency of the architecture across the public and on-prem clouds simplifies the deployment, and more importantly, customers do not need to do any additional work to implement the hybrid cloud. Soon as the Anthos on-prem cluster is deployed and connected to the Google compute platform, the hybrid cloud is ready for operation.

Google Kubernetes Engine (GKE) is the common denominator between the public and on-prem clouds. Since Anthos is primarily focused on enabling containerized workloads running on top of Kubernetes, the hybrid cloud implemented with Anthos enables cross-cloud orchestration of containers and micro-services across the Kubernetes clusters. The migration of container workloads between on-prem and GKE clusters running in other clouds can be achieved through the public container registry such as Google container registry (GCR), or through your own private registries that are secured with the central identity and access control to only allow authenticated users or service accounts to push or pull container images from the registry. There is no need for converting container images running on-prem to run on the Google managed Kubernetes engine on the public cloud.

7.1.1 Google Kubernetes Engine (GKE)

Kubernetes (popularly known as K8s for short), is an open source project developed by Google to enable orchestration of containerized workloads at-scale. Kubernetes typically runs on a cluster of machines as a distributed system and consists of one or more master nodes and one or more worker nodes. The master nodes run the core Kubernetes services such as the API server, scheduler, cluster configuration database (etcd), authentication/authorization services, virtual networking, etc. The worker nodes execute users containerized applications.

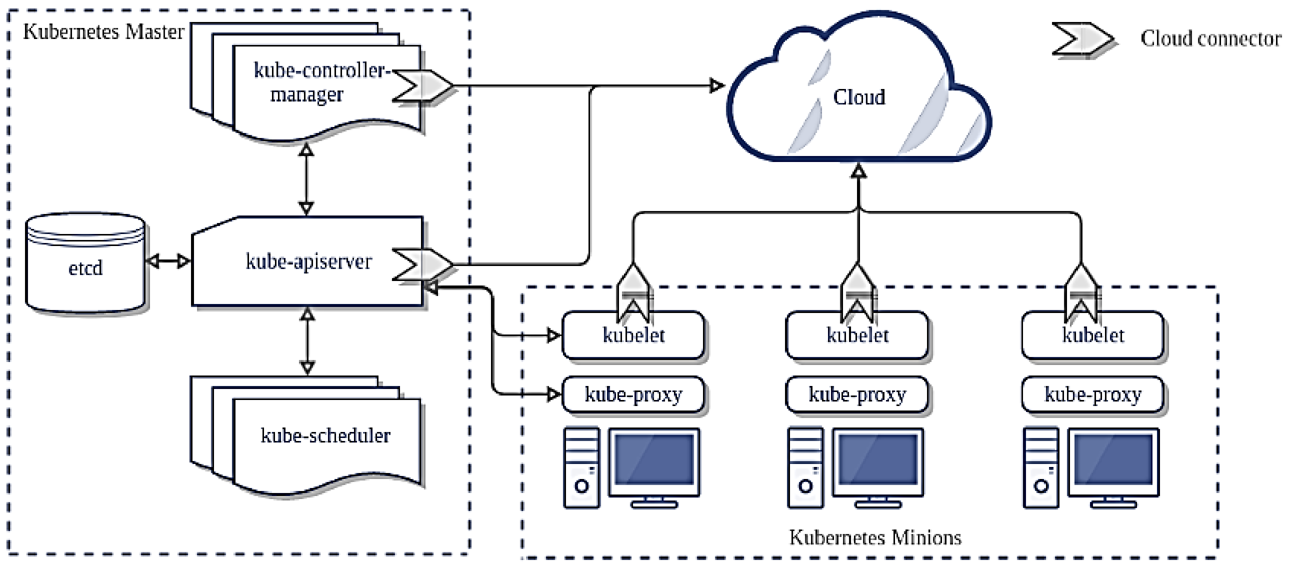


Figure 24: Google Kubernetes service architecture

As shown in Figure 24, the master node(s) run the core cluster management services such as the kube-apiserver, kube-scheduler, and etc. The worker nodes interact with the master nodes via the kubelet service, which is responsible for managing the Kubernetes pods on the local servers. The pods run one or more containers inside. The kube-proxy service provides a simple networking proxy for common ingress traffic into the pods. More complex networking including load-balancers are supported in Kubernetes via built-in as well as open source based projects such as Calico.

For more background on Kubernetes architecture see:

https://kubernetes.io/docs/concepts/architecture/cloud-controller/

7.1.2 Multi-cluster Management

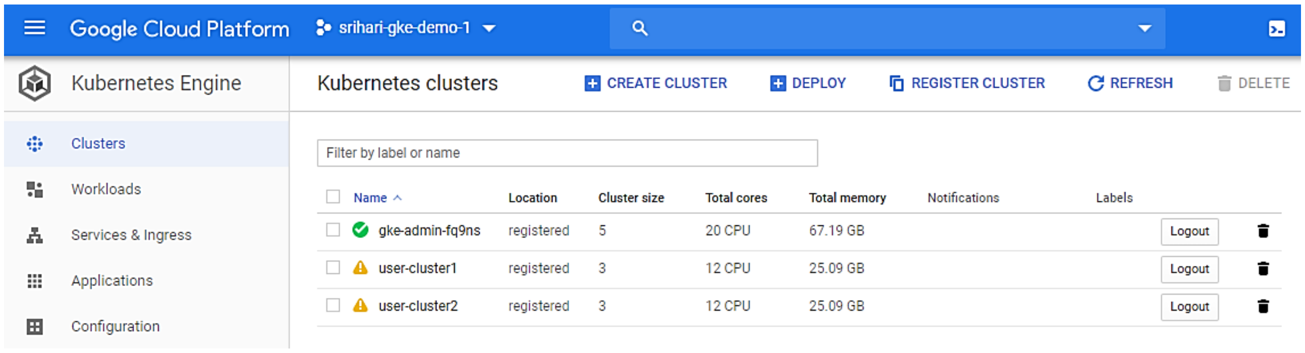
Many enterprises using the cloud today have the need to use multiple solutions, both on-premises and public cloud for various reasons – cost control, specific application workload needs, user demands, SLAs, and so forth. With lack of common standards across various cloud providers, managing multiple clouds requires specific training and administrative skills. With Kubernetes becoming the de facto standard for containerized application orchestration for many customers, it would be beneficial to provide a single control plane for managing Kubernetes clusters anywhere from a single console.

With introduction of Anthos, Google cloud platform also enables managing Kubernetes clusters running anywhere – on Google cloud, on-prem data centers, or other cloud providers such as Amazon AWS from a single place. In addition, the multi-cluster capability also enables configuration management across cloud and on-prem environments as well as workloads running in different environments.

The core components of the multi-cluster management are GKE connection hub (Connect for short), Google cloud platform console, and the Anthos Config management.

7.1.3 Google Cloud Connect

Connect allows you to connect the on-prem Kubernetes clusters as well as Kubernetes clusters running on other public clouds with the Google cloud platform. Connect uses an encrypted connection between the Kubernetes clusters and the Google cloud platform project and enables authorized users to login to clusters, access details about their resources, projects, and clusters, and to manage cluster infrastructure and workloads whether they are running on Google's hardware or elsewhere.



Name	Location	Cluster size	Total cores	Total memory	Notifications	Labels		
gke-admin-fq9ns	registered	5	20 CPU	67.19 GB			Logout	
user-cluster1	registered	3	12 CPU	25.09 GB			Logout	
user-cluster2	registered	3	12 CPU	25.09 GB			Logout	

Figure 25: Google cloud connect for multi-cluster management

GKE Connect Agent installs in your remote cluster

No public IP required for your cluster

Authenticated and encrypted connection from the Kubernetes cluster to GCP

Uses VPC Service Controls to ensure that GCP is an extension of your private cloud

Can traverse NATs and firewalls

User interactions with clusters are visible in Kubernetes Audit Logs

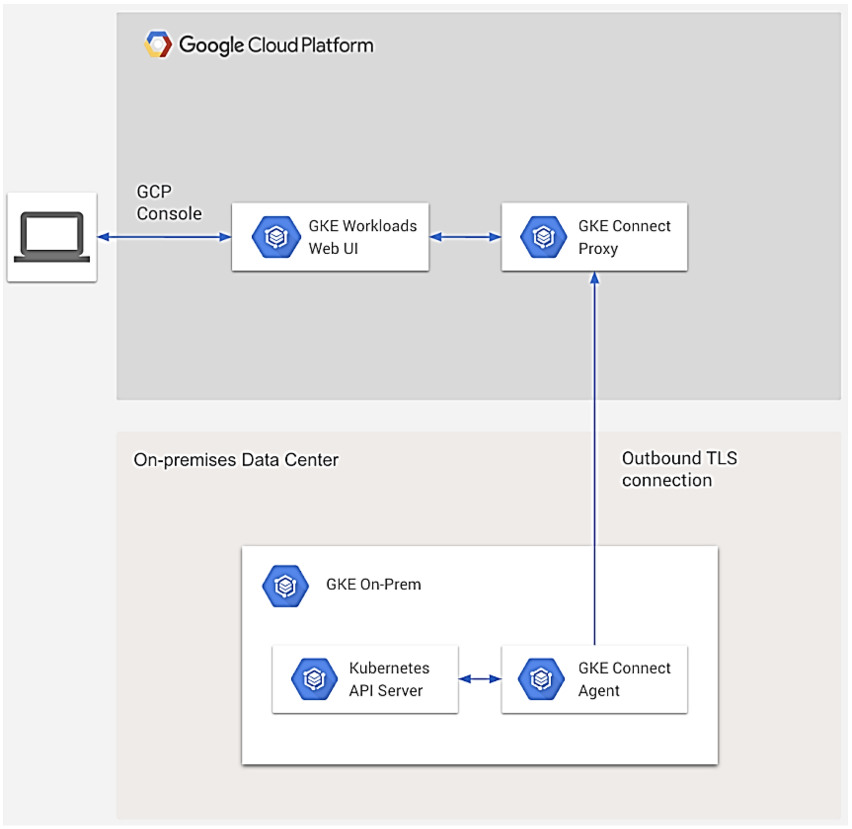


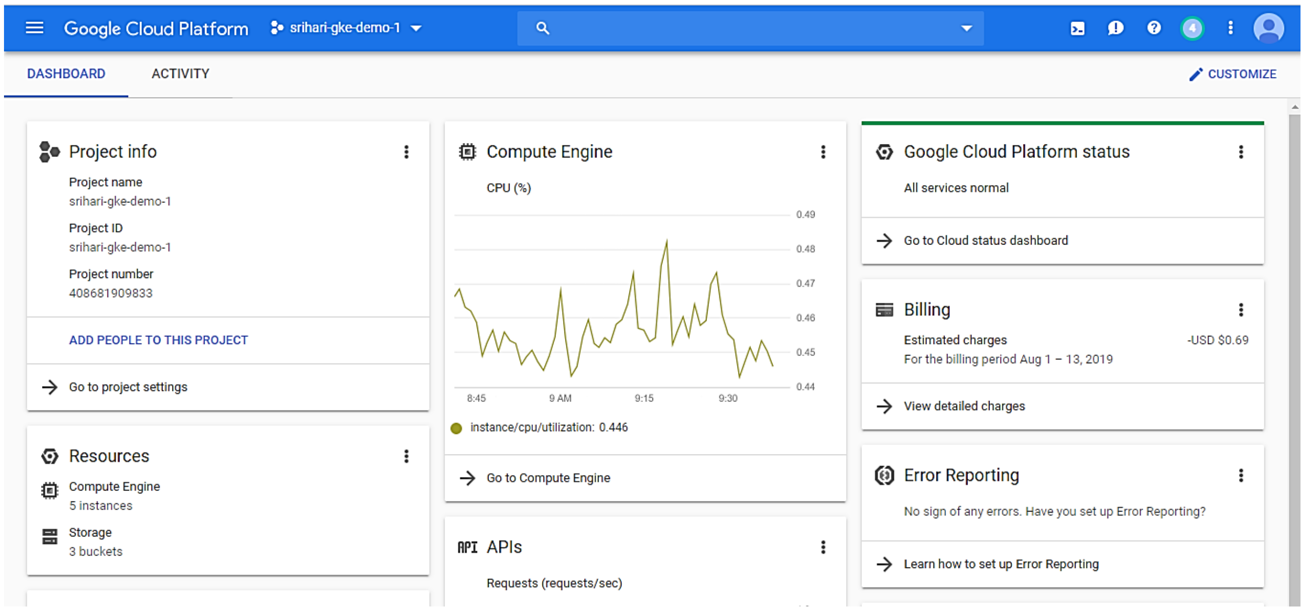
Figure 26: Secure (TLS) based connection to Google cloud platform from on-prem

More information about Google Connect can be found here:

https://cloud.google.com/anthos/multicluster-management/connect/overview

7.1.4 GCP Console

Google cloud platform console acts as the single point of management and monitoring of Kubernetes clusters running in different locations. This is the same Web based UI used to manage all the Google cloud resources such as the compute engine clusters, storage, networking, Kubernetes clusters, and so forth.



Project info

Project name
srihari-gke-demo-1

Project ID
srihari-gke-demo-1

Project number
408681909833

ADD PEOPLE TO THIS PROJECT

Go to project settings

Resources

Compute Engine
5 instances

Storage
3 buckets

Compute Engine

CPU (%)

instance/cpu/utilization: 0.446

Go to Compute Engine

APIs

Requests (requests/sec)

Google Cloud Platform status

All services normal

Go to Cloud status dashboard

Billing

Estimated charges	-USD $0.69
For the billing period Aug 1 – 13, 2019	

View detailed charges

Error Reporting

No sign of any errors. Have you set up Error Reporting?

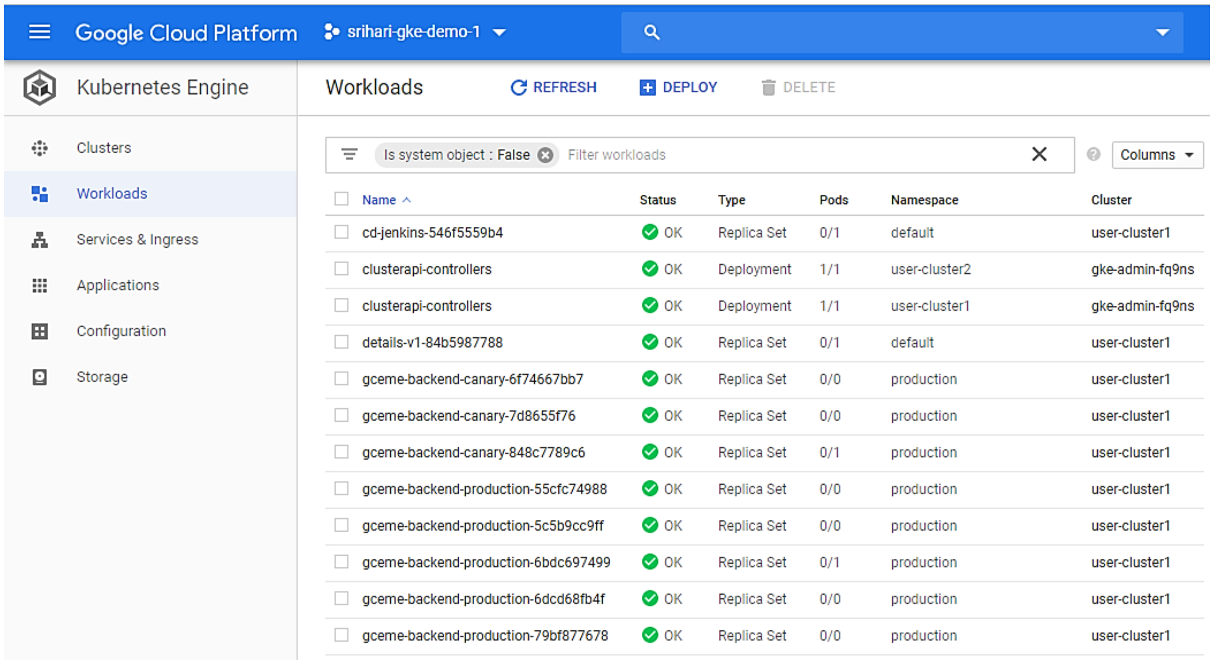
Learn how to set up Error Reporting

Figure 27: Google cloud platform console

7.1.5 Managing Anthos Clusters from GCP

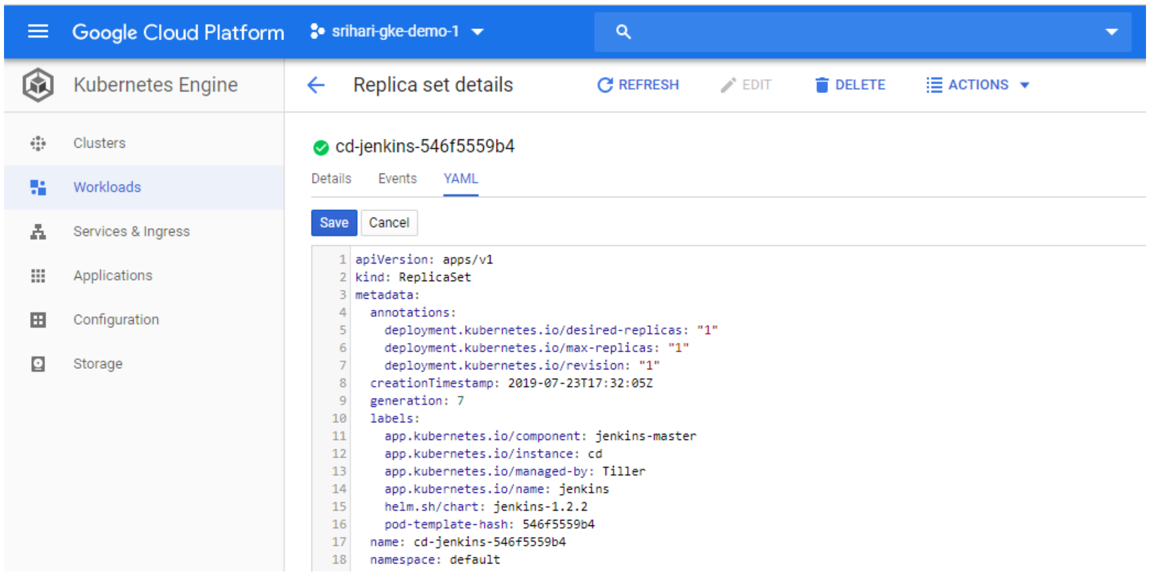
Management of multiple Kubernetes clusters and workloads running across different locations becomes easy with the use of the GCP console. For example, you can use the console to check the health of the running workloads and make any configuration changes to them.

Note that the Anthos GKE clusters running in your on-premises data center need to be connected and registered with GCP to be reachable by Google and displayed in GCP Console. The GKE on-prem clusters deployed through Anthos are automatically registered and connected with GCP as part of the setup process.



Name	Status	Type	Pods	Namespace	Cluster
cd-jenkins-546f5559b4	OK	Replica Set	0/1	default	user-cluster1
clusterapi-controllers	OK	Deployment	1/1	user-cluster2	gke-admin-fq9ns
clusterapi-controllers	OK	Deployment	1/1	user-cluster1	gke-admin-fq9ns
details-v1-84b5987788	OK	Replica Set	0/1	default	user-cluster1
gceme-backend-canary-6f74667bb7	OK	Replica Set	0/0	production	user-cluster1
gceme-backend-canary-7d8655f76	OK	Replica Set	0/0	production	user-cluster1
gceme-backend-canary-848c7789c6	OK	Replica Set	0/1	production	user-cluster1
gceme-backend-production-55cfc74988	OK	Replica Set	0/0	production	user-cluster1
gceme-backend-production-5c5b9cc9ff	OK	Replica Set	0/0	production	user-cluster1
gceme-backend-production-6bdc697499	OK	Replica Set	0/1	production	user-cluster1
gceme-backend-production-6dcd68fb4f	OK	Replica Set	0/0	production	user-cluster1
gceme-backend-production-79bf877678	OK	Replica Set	0/0	production	user-cluster1

Figure 28: Managing workloads across GKE clusters from GCP console



```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  annotations:
    deployment.kubernetes.io/desired-replicas: "1"
    deployment.kubernetes.io/max-replicas: "1"
    deployment.kubernetes.io/revision: "1"
  creationTimestamp: 2019-07-23T17:32:05Z
  generation: 7
  labels:
    app.kubernetes.io/component: jenkins-master
    app.kubernetes.io/instance: cd
    app.kubernetes.io/managed-by: Tiller
    app.kubernetes.io/name: jenkins
    helm.sh/chart: jenkins-1.2.2
    pod-template-hash: 546f5559b4
  name: cd-jenkins-546f5559b4
  namespace: default
```

Figure 29: Editing a workload definition from GCP console

More information about the GCP console can be found here:

https://cloud.google.com/anthos/multicluster-management/console/

7.2 DevOps and CI/CD Pipelines

DevOps is one of the primary use-cases for Anthos. Modern software development practice is implemented following Agile/Scrum and DevOps methodologies. We covered DevOps in the previous chapters. In this section we will describe from a high-level how to implement a continuous integration and continuous deployment (CI/CD) pipeline on top of the Anthos Kubernetes clusters.

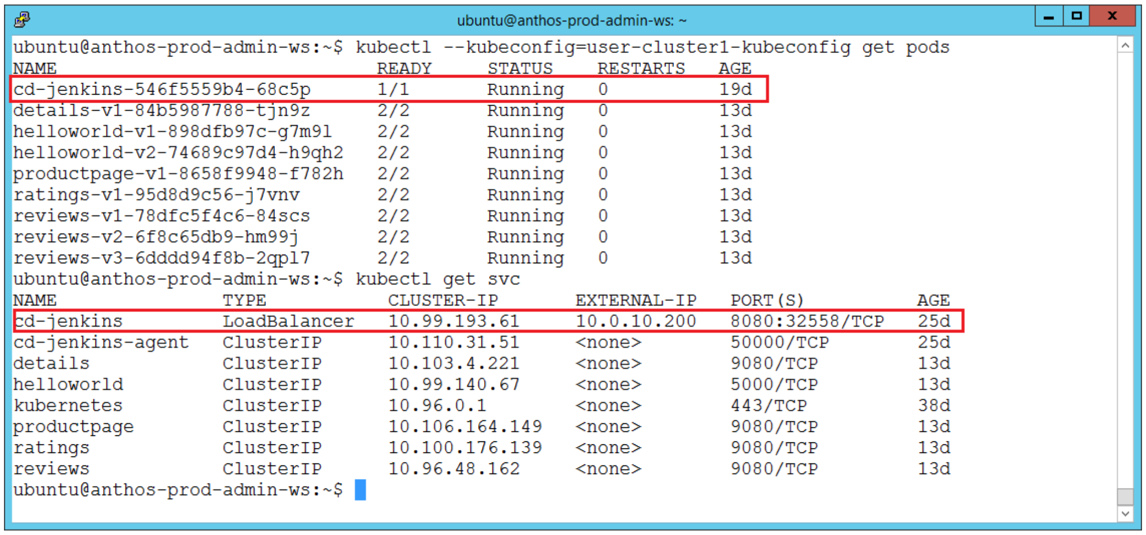
Continuous integration is the process in which code developed by multiple developers concurrently is continuously pulled from the source code repository, integrated, built, and tested.

7.2.1 Jenkins deployment and integration with GKE on-prem

To implement CI/CD pipeline on the GKE cluster on-prem, you can use the popular open source CI/CD tool called Jenkins. There are other popular open source and commercial CI/CD tools available in the market as well, but Jenkins has a broad eco system of open source and commercial plugins for a variety of CI/CD configurations, including integration with Kubernetes and Docker, which makes it a good fit for Anthos.

Jenkins itself can be deployed containerized on top of the GKE cluster on-prem, which makes the deployment quite straightforward. We are not covering Jenkins deployment in detail in this paper. Please see the following tutorial for a step-by-step implementation of Jenkins on Kubernetes.

https://cloud.google.com/solutions/jenkins-on-kubernetes-engine



```
ubuntu@anthos-prod-admin-ws:~$ kubectl --kubeconfig=user-cluster1-kubeconfig get pods
NAME                               READY     STATUS    RESTARTS   AGE
cd-jenkins-546f5559b4-68c5p        1/1       Running   0          19d
details-v1-84b5987788-tjn9z        2/2       Running   0          13d
helloworld-v1-898dfb97c-g7m9l      2/2       Running   0          13d
helloworld-v2-74689c97d4-h9qh2     2/2       Running   0          13d
productpage-v1-8658f9948-f782h     2/2       Running   0          13d
ratings-v1-95d8d9c56-j7vnv         2/2       Running   0          13d
reviews-v1-78dfc5f4c6-84scs        2/2       Running   0          13d
reviews-v2-6f8c65db9-hm99j         2/2       Running   0          13d
reviews-v3-6dddd94f8b-2qpl7        2/2       Running   0          13d
ubuntu@anthos-prod-admin-ws:~$ kubectl get svc
NAME                 TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)          AGE
cd-jenkins           LoadBalancer   10.99.193.61     10.0.10.200    8080:32558/TCP   25d
cd-jenkins-agent     ClusterIP      10.110.31.51     <none>         50000/TCP        25d
details              ClusterIP      10.103.4.221     <none>         9080/TCP         13d
helloworld           ClusterIP      10.99.140.67     <none>         5000/TCP         13d
kubernetes           ClusterIP      10.96.0.1        <none>         443/TCP          38d
productpage          ClusterIP      10.106.164.149   <none>         9080/TCP         13d
ratings              ClusterIP      10.100.176.139   <none>         9080/TCP         13d
reviews              ClusterIP      10.96.48.162     <none>         9080/TCP         13d
ubuntu@anthos-prod-admin-ws:~$
```

Figure 30: Jenkins CI/CD tool deployed as container on GKE on-prem cluster

Once Jenkins has been deployed on the cluster you will see the Jenkins container pods successfully created and running. See Figure 30 for the kubectl commands to check the Jenkins pod status and to access the Jenkins website at that point.

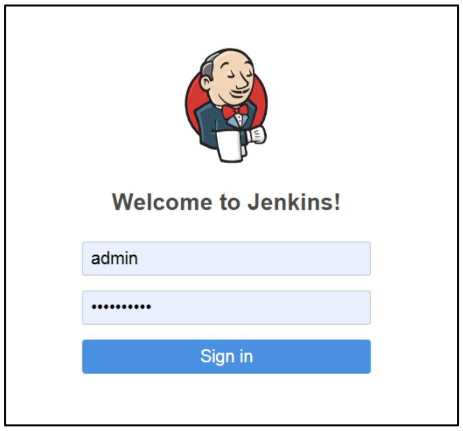
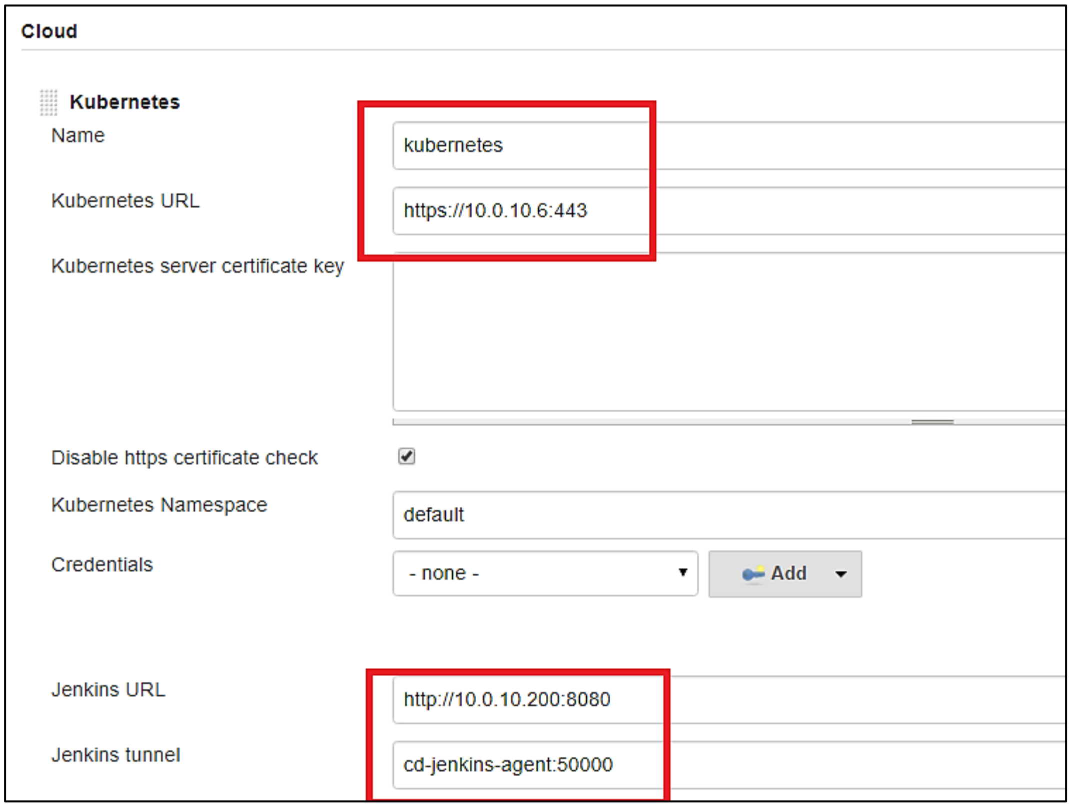


Figure 31: Jenkins master login screen

After installing Jenkins successfully, you need to configure Jenkins and attach the Anthos GKE cluster with the Jenkins master to use to run the CI/CD pipelines. From the Jenkins portal, select “configure Jenkins” and create the “cloud” configuration (Figure 32). This is where you need to specify “kubernetes” as the name of the cloud because this same cloud needs to be specified with the pipeline definition later. Also, you need to specify the URL for the Jenkins master and the Kubernetes agent.



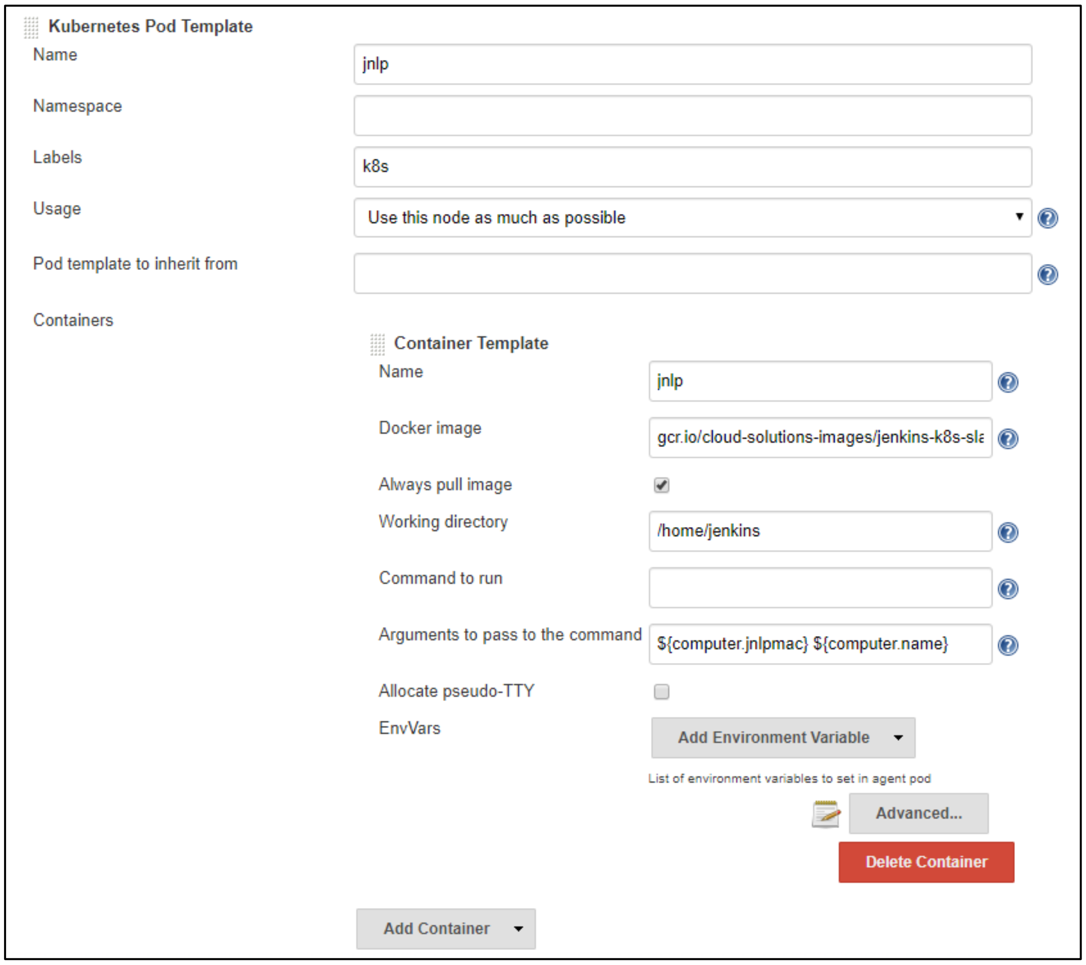
Cloud

Kubernetes

Name	kubernetes
Kubernetes URL	https://10.0.10.6:443
Kubernetes server certificate key	
Disable https certificate check	✓
Kubernetes Namespace	default
Credentials	- none -
Jenkins URL	http://10.0.10.200:8080
Jenkins tunnel	cd-jenkins-agent:50000

Figure 32: Kubernetes cloud definition in Jenkins configuration

In order for the Jenkins master to successfully deploy and run the Jenkins slaves on the Kubernetes cluster you also need to configure the credentials for the Kubernetes cluster in the global Jenkins credentials. You can basically copy-paste the kubeconfig file from the Anthos GKE cluster. See Figure 33.



Kubernetes Pod Template

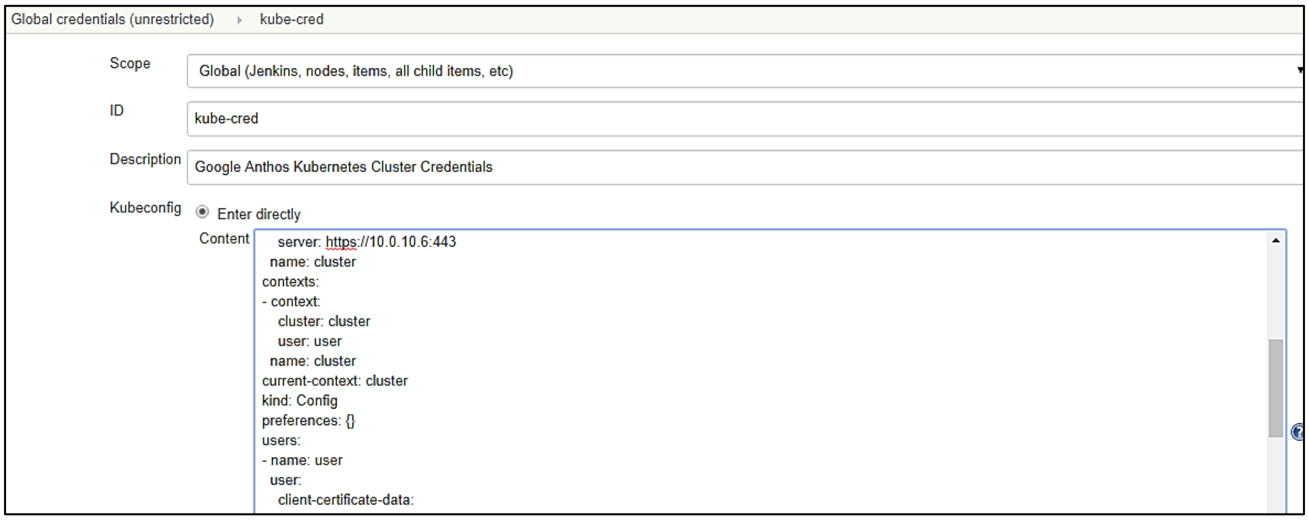
Name	jnlp
Namespace	
Labels	k8s
Usage	Use this node as much as possible
Pod template to inherit from	
Containers	

Container Template

Name	jnlp
Docker image	gcr.io/cloud-solutions-images/jenkins-k8s-sla
Always pull image	✓
Working directory	/home/jenkins
Command to run	
Arguments to pass to the command	${computer.jnlpmac} ${computer.name}
Allocate pseudo-TTY	☐
EnvVars	Add Environment Variable

List of environment variables to set in agent pod

Figure 33: Kubernetes pod template for Jenkins slave



Global credentials (unrestricted) › kube-cred

Scope	Global (Jenkins, nodes, items, all child items, etc)
ID	kube-cred
Description	Google Anthos Kubernetes Cluster Credentials
Kubeconfig	● Enter directly

Content

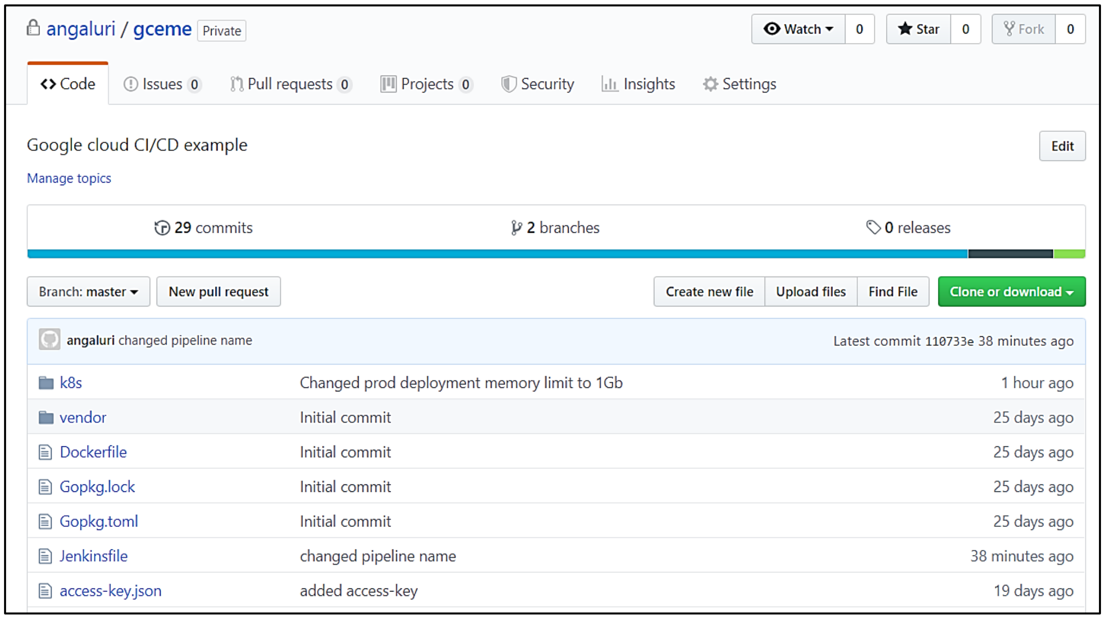
```
    server: https://10.0.10.6:443
  name: cluster
contexts:
- context:
    cluster: cluster
    user: user
  name: cluster
current-context: cluster
kind: Config
preferences: {}
users:
- name: user
  user:
    client-certificate-data:
```

Figure 34: Kubeconfig file for the GKE on-prem cluster with Jenkins

7.2.2 Integrating Jenkins with source code repository

To implement the CI part of the pipeline, you need to integrate the source code repo with Jenkins and configure a pipeline agent that will periodically scan the repos for updates and automatically schedule the builds. In this example, we are using Github. The sample code from the Google CI/CD tutorial below has been cloned into another personal repository on Git.

https://cloud.google.com/solutions/continuous-delivery-jenkins-kubernetes-engine


angaluri / gceme Private

Watch 0 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Security | Insights | Settings

Google cloud CI/CD example

Manage topics

29 commits | 2 branches | 0 releases

Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

angaluri changed pipeline name — Latest commit 110733e 38 minutes ago

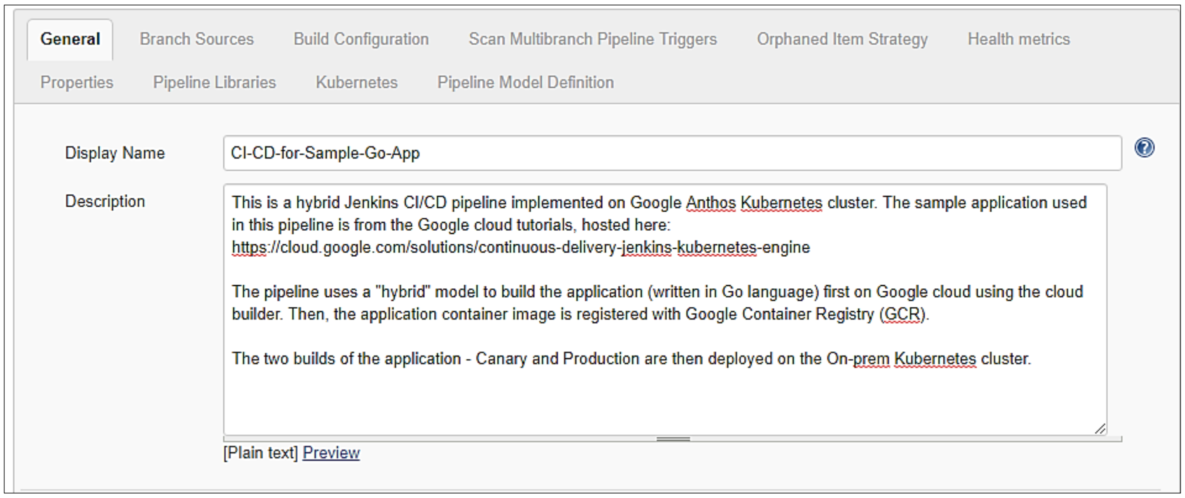
k8s	Changed prod deployment memory limit to 1Gb	1 hour ago
vendor	Initial commit	25 days ago
Dockerfile	Initial commit	25 days ago
Gopkg.lock	Initial commit	25 days ago
Gopkg.toml	Initial commit	25 days ago
Jenkinsfile	changed pipeline name	38 minutes ago
access-key.json	added access-key	19 days ago

Figure 35: Git repository for sample CI/CD application

You will also need to register your Git repository credentials in Jenkins so that the Jenkins build agent can access the repository and checkout the code. In addition, for your personal development workstation to pull and push code to the Git repository you need to register the SSH keys with the Git repo and enable them. See the Git documentation on how to do that.

7.2.3 CI/CD pipeline creation

In Jenkins, create a new multi-branch pipeline project. The Git repository where the code is hosted should be specified with the corresponding credentials. See Figure 37.


General | Branch Sources | Build Configuration | Scan Multibranch Pipeline Triggers | Orphaned Item Strategy | Health metrics | Properties | Pipeline Libraries | Kubernetes | Pipeline Model Definition

Display Name: CI-CD-for-Sample-Go-App

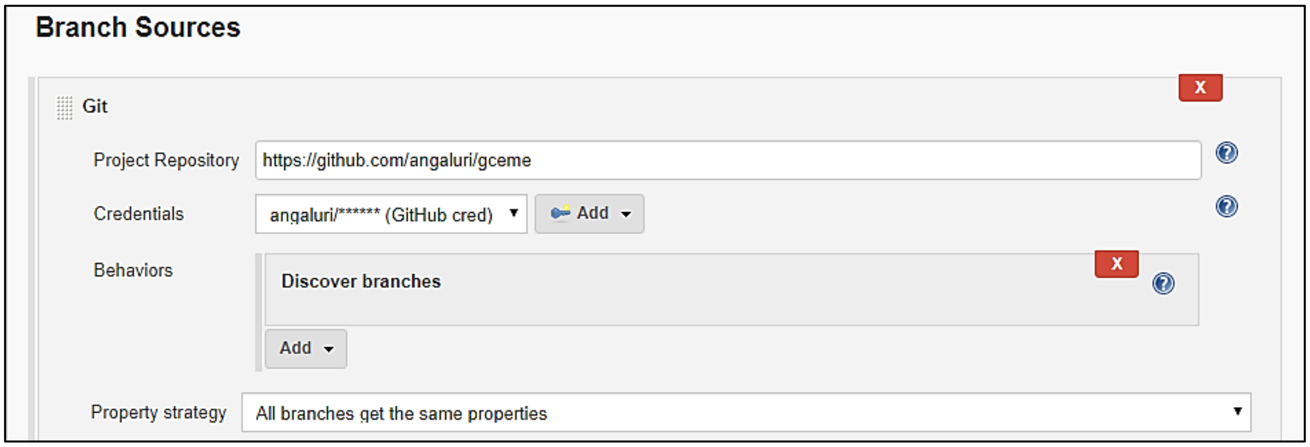
Description: This is a hybrid Jenkins CI/CD pipeline implemented on Google Anthos Kubernetes cluster. The sample application used in this pipeline is from the Google cloud tutorials, hosted here:
https://cloud.google.com/solutions/continuous-delivery-jenkins-kubernetes-engine

The pipeline uses a "hybrid" model to build the application (written in Go language) first on Google cloud using the cloud builder. Then, the application container image is registered with Google Container Registry (GCR).

The two builds of the application - Canary and Production are then deployed on the On-prem Kubernetes cluster.

[Plain text] Preview

Figure 36: Creating a multi-branch pipeline for the sample CI/CD application


Branch Sources

Git

Project Repository: https://github.com/angaluri/gceme

Credentials: angaluri/****** (GitHub cred)

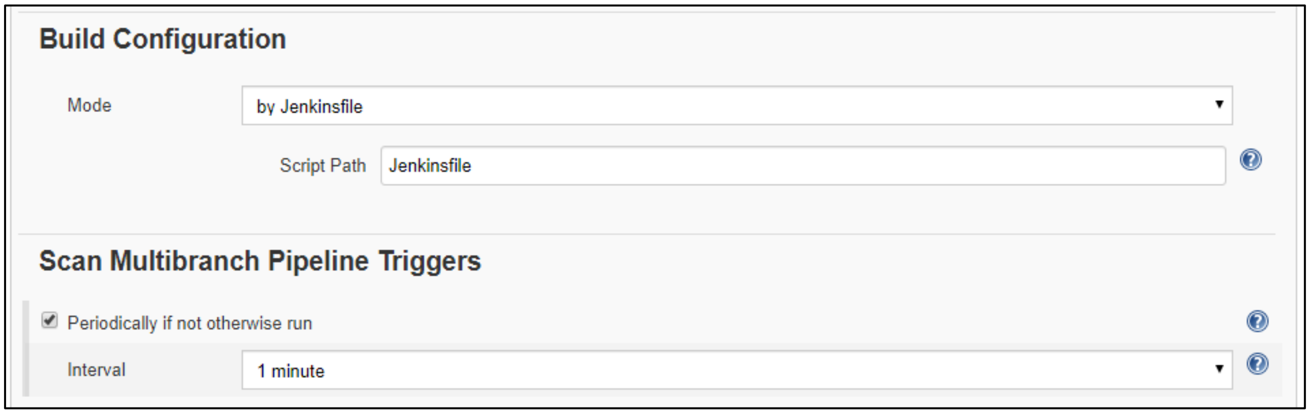
Behaviors: Discover branches

Property strategy: All branches get the same properties

Figure 37: Github repository and credentials for CI/CD pipeline

For the continuous integration setup, we want the source code repo to be scanned periodically for changes. You can also configure build triggers via web hooks within Git configuration such that Git will trigger the pipeline build when new code is committed to the repos. However, typically your Anthos GKE clusters will be behind corporate firewalls, which will prevent the Git web hooks traffic to the Jenkins server. There are ways of working around this issue. See the following article for details on making Git web hooks work with firewalls.

For this article, we will configure a periodic repository scanner in the Jenkins pipeline. See Figure 38 where we specify one minute as the interval for scanning the repositories on Git hub. Every one minute, Jenkins agent will scan the repository for any code updates and then trigger the pipeline build.


Build Configuration

Mode: by Jenkinsfile

Script Path: Jenkinsfile

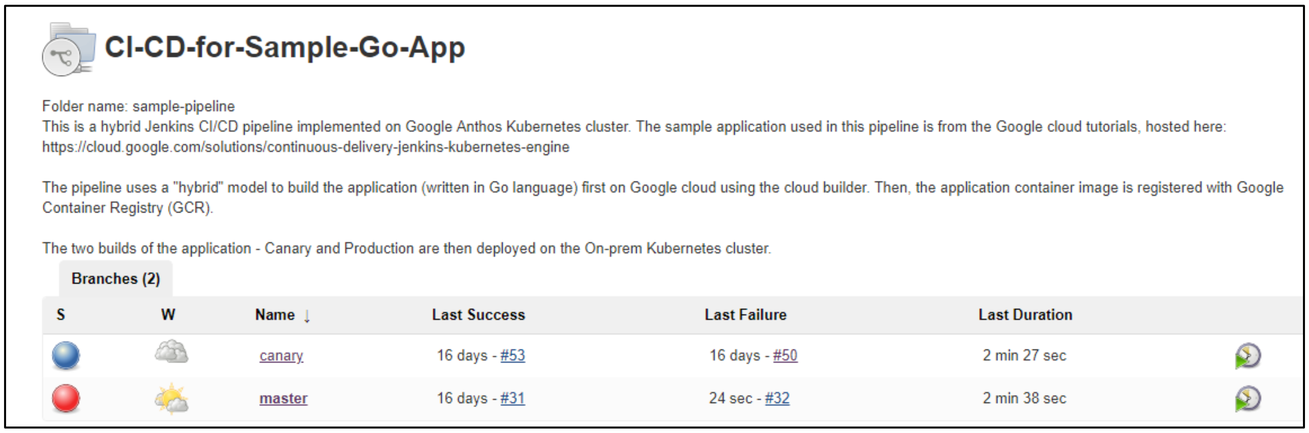
Scan Multibranch Pipeline Triggers

☑ Periodically if not otherwise run

Interval: 1 minute

Figure 38: Pipeline scanner for triggering builds

Once the multi-branch pipeline is created you can see that pipeline in the Jenkins dashboard as in figure. You can see there are two branches detected by Jenkins – Canary and Master. Jenkins will trigger the build on one or both of these branches individually when the code scanner detects the changes.



CI-CD-for-Sample-Go-App

Folder name: sample-pipeline
This is a hybrid Jenkins CI/CD pipeline implemented on Google Anthos Kubernetes cluster. The sample application used in this pipeline is from the Google cloud tutorials, hosted here: https://cloud.google.com/solutions/continuous-delivery-jenkins-kubernetes-engine

The pipeline uses a "hybrid" model to build the application (written in Go language) first on Google cloud using the cloud builder. Then, the application container image is registered with Google Container Registry (GCR).

The two builds of the application - Canary and Production are then deployed on the On-prem Kubernetes cluster.

Branches (2)

S	W	Name ↓	Last Success	Last Failure	Last Duration
🔵		canary	16 days - #53	16 days - #50	2 min 27 sec
🔴		master	16 days - #31	24 sec - #32	2 min 38 sec

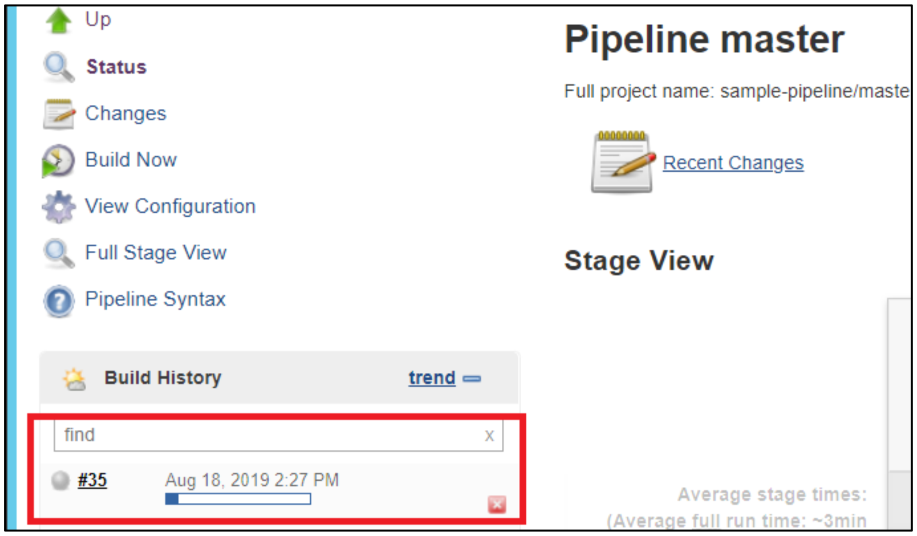
Figure 39: Multi-branch pipeline for the sample CI/CD application

7.2.4 Triggering pipeline builds

When code changes are made and committed to the respective branch, the pipeline build scanner will see the changes and trigger an automatic build. In the output below, we made a change to one of the YAML files in the repository and push the change to the repository origin, which is on Github.

```
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
$git commit -a -m 'Changed prod deployment memory limit to 1Gb'
[master c7dfee6] Changed prod deployment memory limit to 1Gb
 1 file changed, 1 insertion(+), 1 deletion(-)
```

```
$ git push origin master
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 588 bytes | 588.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To git+ssh://github.com/angaluri/gceme.git
   41fb4b2..c7dfee6  master -> master
```



Pipeline master

Full project name: sample-pipeline/maste

Recent Changes

Stage View

Average stage times:
(Average full run time: ~3min

Figure 40: Automatic build trigger after code checkin

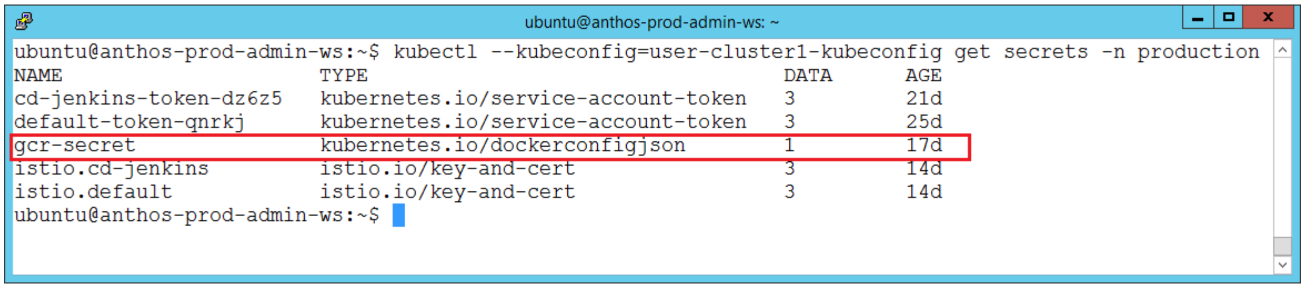
7.2.5 CI/CD build pipeline execution

Jenkins supports multiple scripting and declarative languages to create pipelines. The script is typically specified in a Jenkinsfile and included within the source repo, or provided with the pipeline definition. The various stages of the pipeline and corresponding steps to checkout, build, test, and deploy the code are specified in the Jenkinsfile.

During the init step, various settings for the build are specified, including the project ID for the GCP project where the Anthos on-prem cluster is registered, the application parameters, build tag for the docker image for the application, the IAM cloud service account, and the Jenkins account authorized for the Jenkins slave running on the Kubernetes cluster.

```
def project = 'srihari-gke-demo-1'
def  appName = 'gceme'
def  feSvcName = "${appName}-frontend"
def  imageTag = "gcr.io/${project}/${appName}:${env.BRANCH_NAME}.${env.BUILD_NUMBER}"
def  gcloud_account = 'sa-connect-sri-gke-1@srihari-gke-demo-1.iam.gserviceaccount.com'
def jenkins_account = 'cd-jenkins'
```

The Google cloud service account is used for accessing the Google container registry, where we would push the built docker container images. This account should have the read/write access to the Google cloud storage bucket which is used by GCR as the image repository. The same account should be then specified in the later stage during the deployment of the code to the on-prem Kubernetes cluster. The service account credentials should be registered in the cluster image pull secret store and then used as part of the deployment definition of the pods.



```
ubuntu@anthos-prod-admin-ws:~$ kubectl --kubeconfig=user-cluster1-kubeconfig get secrets -n production
NAME                     TYPE                                  DATA      AGE
cd-jenkins-token-dz6z5   kubernetes.io/service-account-token   3         21d
default-token-qnrkj      kubernetes.io/service-account-token   3         25d
gcr-secret               kubernetes.io/dockerconfigjson        1         17d
istio.cd-jenkins         istio.io/key-and-cert                 3         14d
istio.default            istio.io/key-and-cert                 3         14d
ubuntu@anthos-prod-admin-ws:~$
```

Figure 41: Storing credentials for Google container registry access

```
    spec:
      containers:
      - name: backend
        image: gcr.io/cloud-solutions-images/gceme:1.0.0
        resources:
          limits:
            memory: "1000Mi"
            cpu: "100m"
        imagePullPolicy: Always
        readinessProbe:
          httpGet:
            path: /healthz
            port: 8080
        command: ["sh", "-c", "app -port=8080"]
        ports:
        - name: backend
          containerPort: 8080
      imagePullSecrets:
      - name: gcr-secret
```

Sample output from the pipeline build process is shown below.

The first step is for Jenkins to checkout the source code from Git and extract the Jenkinsfile, which describes the pipeline steps.

```
Setting origin to https://github.com/angaluri/gceme
 > git config remote.origin.url https://github.com/angaluri/gceme # timeout=10
Fetching origin...
Fetching upstream changes from origin
 > git --version # timeout=10
 > git config --get remote.origin.url # timeout=10
using GIT_ASKPASS to set credentials GitHub cred
 > git fetch --tags --progress origin +refs/heads/*:refs/remotes/origin/*
Seen branch in repository origin/canary
Seen branch in repository origin/master
Seen 2 remote branches
Obtained Jenkinsfile from 110733ed0c14d360089a0f892908154853dbad6a
```

Once the Jenkinsfile is obtained, the pipeline build stages will execute. The full Jenkinsfile is posted at the end of this section.

The test phase will execute the unit tests (and any other additional QA to be done) for the code. The following is the console output in Jenkins from the pipeline execution.

```
[Pipeline] { (Test)
[Pipeline] container
[Pipeline] {
[Pipeline] sh
+ pwd
+ ln -s /home/jenkins/workspace/sample-pipeline_master /go/src/sample-app
+ cd /go/src/sample-app
+ go test
PASS
ok  sample-app  0.015s
```

Once the test phase is complete and successful, the next stage is to build the container image for the application and push it to the container registry on GCR.

```
[Pipeline] { (Build and push image with Container Builder)
[Pipeline] container
[Pipeline] {
[Pipeline] sh
+ PYTHONUNBUFFERED=1 gcloud builds submit -t gcr.io/srihari-gke-demo-
1/gceme:master.35 .
Creating temporary tarball archive of 34 file(s) totalling 83.3 KiB before
compression.
Uploading tarball of [.] to [gs://srihari-gke-demo-1_cloudbuild/source/1566138521.78-
fc2917dcd2f34da3b8e792e055650e30.tgz]
```

```
Created [https://cloudbuild.googleapis.com/v1/projects/srihari-gke-demo-1/builds/b56222e6-
4140-40c1-a5b8-61abd8b5acb5].
Logs are available at [https://console.cloud.google.com/gcr/builds/b56222e6-4140-40c1-
a5b8-61abd8b5acb5?project=408681909833].
..
Successfully built 81f4bb7c3874
Successfully tagged gcr.io/srihari-gke-demo-1/gceme:master.35
PUSH
Pushing gcr.io/srihari-gke-demo-1/gceme:master.35
DONE
```

7.2.6 Continuous deployment

Once the code is successfully built and the container image pushed to the registry, the deployment stage kicks-off. In this simple example, we are just pulling down the image from the registry and deploying it to the Anthos Kubernetes on-prem cluster using the kubectl commands. In a real production scenario, there would be multiple additional checks to make sure the code should be updated on the production system through a scheduled maintenance window. On the other hand, with Kubernetes, the advantage is that you can do side-by-side deployments of different versions of code and selectively route traffic across the different deployments to test before replacing the active production instance. For example, in this sample application we have the canary and master branches. When new functions need to be tested first with production level traffic, you route the traffic to the canary deployment. With the Istio service mesh you can also create the traffic routing rules to test blue-green deployments.

```
[Pipeline] { (Deploy Production)
[Pipeline] container
[Pipeline] {
[Pipeline] sh
+ sed -i.bak s#gcr.io/cloud-solutions-images/gceme:1.0.0#gcr.io/srihari-gke-demo-
1/gceme:master.35# ./k8s/production/backend-production.yaml ./k8s/production/frontend-
production.yaml
[Pipeline] sh
+ kubectl --namespace=production apply -f k8s/services/
service/gceme-backend unchanged
service/gceme-frontend unchanged
[Pipeline] sh
+ kubectl --namespace=production apply -f k8s/production/
deployment.extensions/gceme-backend-production configured
deployment.extensions/gceme-frontend-production configured
[Pipeline] sh
+ kubectl --namespace=production get service/gceme-frontend -o
jsonpath={.status.loadBalancer.ingress[0].ip}
+ echo http://10.0.10.205
```

```
def project = 'srihari-gke-demo-1'
def  appName = 'gceme'
def  feSvcName = "${appName}-frontend"
def  imageTag = "gcr.io/${project}/${appName}:${env.BRANCH_NAME}.${env.BUILD_NUMBER}"
def  gcloud_account = 'sa-connect-sri-gke-1@srihari-gke-demo-1.iam.gserviceaccount.com'
def jenkins_account = 'cd-jenkins'
pipeline {
  environment {
    registry = "gcr.io/${project}"
    IMAGETAG = "${imageTag}"
  }
agent {
    kubernetes {
      label 'sample-app'
      defaultContainer 'jnlp'
      yaml """
apiVersion: v1
kind: Pod
metadata:
labels:
  component: ci
spec:
  # Use service account that can deploy to all namespaces
  serviceAccountName: "$jenkins_account"
  containers:
  - name: golang
    image: golang:1.10
    command:
    - cat
    tty: true
  - name: gcloud
    image: gcr.io/cloud-builders/gcloud
    command:
    - cat
    tty: true
  - name: kubectl
    image: gcr.io/cloud-builders/kubectl
    command:
    - cat
    tty: true
"""
}
}
```

```
stages {
  stage('Init') {
    steps {
      container('kubectl') {
        sh "gcloud config set project ${project}"
        sh "gcloud config set account ${gcloud_account}"
        sh "gcloud auth activate-service-account --key-file=connect-key.json ${gcloud_account}"
        sh "gcloud auth configure-docker"
      }
      container('gcloud') {
        sh "gcloud config set project ${project}"
        sh "gcloud config set account ${gcloud_account}"
        sh "gcloud auth configure-docker"
      }
    }
  }
```

```
    stage('Test') {
    steps {
      container('golang') {
        sh """
          ln -s `pwd` /go/src/sample-app
          cd /go/src/sample-app
          go test
        """
      }
    }
  }
  stage('Build and push image with Container Builder') {
    steps {
      container('gcloud') {
        sh "PYTHONUNBUFFERED=1 gcloud builds submit -t ${imageTag} ."
      }
    }
  }
```

```
stage('Build and push image with Container Builder') {
     steps {
       container('gcloud') {
         sh "PYTHONUNBUFFERED=1 gcloud builds submit -t ${imageTag} ."
       }
     }
   }
   stage('Deploy Canary') {
     // Canary branch
     when { branch 'canary' }
     steps {
       container('kubectl') {
         // Change deployed image in canary to the one we just built
         sh("sed -i.bak 's#gcr.io/cloud-solutions-images/gceme:1.0.0#${imageTag}#' ./k8s/canary/*.yaml")
         sh("kubectl --namespace=production apply -f k8s/services/")
         sh("kubectl --namespace=production apply -f k8s/canary/")
         sh("echo http://`kubectl --namespace=production get service/${feSvcName} -o
jsonpath='{.status.loadBalancer.ingress[0].ip}'` > ${feSvcName}")
       }
     }
   }
   stage('Deploy Production') {
     // Production branch
     when { branch 'master' }
     steps{
       container('kubectl') {
       // Change deployed image in canary to the one we just built
         sh("sed -i.bak 's#gcr.io/cloud-solutions-
images/gceme:1.0.0#${imageTag}#' ./k8s/production/*.yaml")
         sh("kubectl --namespace=production apply -f k8s/services/")
         sh("kubectl --namespace=production apply -f k8s/production/")
         //sh("kubectl --namespace=production describe secret gcrregcred")
         sh("echo http://`kubectl --namespace=production get service/${feSvcName} -o
jsonpath='{.status.loadBalancer.ingress[0].ip}'` > ${feSvcName}")
       }
     }
   }
```

7.3 Micro-services development and service mesh

Micro-services is an architecture that structures an application as a collection of services. Advantages of Micro-services are the following:

- Micro-services simplify integration of the businesses, processes, technology, and people by breaking down the monolithic application to a smaller set that can be handled independently.
- They help build an application as a suite of small services, each running in its own process and are independently deployable.
- Micro-services can be written in different programming languages and may use different data storage techniques.
- Micro-services are scalable and flexible, and connected via APIs,
- Leverage many of the reusable tools and solutions in the RESTful and web service ecosystem.
- Micro-service architecture enables the rapid, frequent and reliable delivery of large, complex applications.
- Enables an organization to quickly evolve its technology stack

Micro-services Apps are deployed as a set of containers in Kubernetes cluster. Istio is a service mesh platform to connect micro-services. Istio makes it easy to manage load balancing, service-to-service authentication, monitoring, etc., in services network. Figure 42 shows the official architecture diagram of istio 1.1.

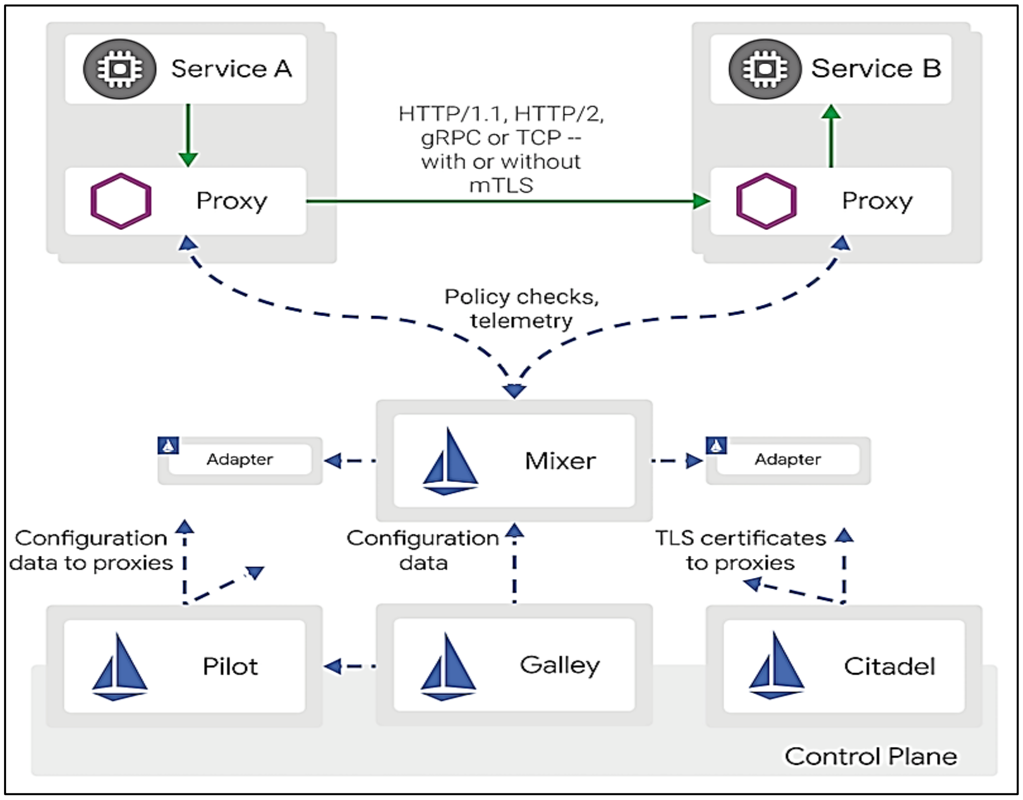


Figure 42: Istio service mesh architecture

Istio deploys a special sidecar proxy throughout application environment that intercepts all network communication between micro-services with few code or no code changes in service code. It reduces the complexity of managing micro-services deployments.

More detailed introduction of Istio components can be found at:

https://istio.io/docs/

Figure 43 shows architecture of Istio deployed on Anthos GKE On-Prem on ThinkAgile VX platform.

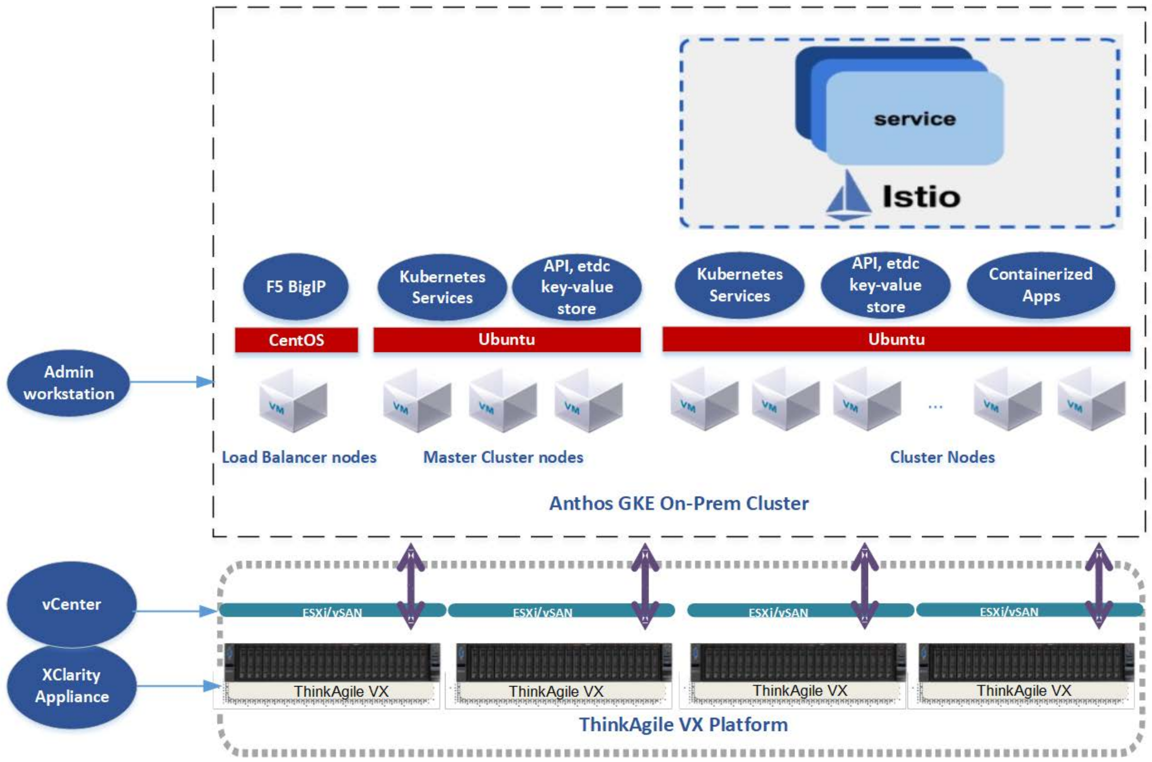


Figure 43: Istio service mesh on GKE on-prem cluster

Istio is installed in user clusters on Anthos GKE On-Prem with Lenovo ThinkAgile VX platform. Users can leverage Istio to deploy applications and provide service to their customers.

8 Appendix: Lenovo Bill of materials

This appendix contains the bill of materials (BOMs) for different configurations of hardware for Anthos on-prem cluster deployments using the Lenovo ThinkAgile VX (vSAN) hyperconverged infrastructure (HCI). Since HCI solution provides shared storage from the locally attached disks on the servers, there is no need for an external storage source. The options for network switching are provided as separate BOM because for each of the system configurations below, you can combine the network switch options with the respective compute nodes to implement the full system.

There are three configurations provided below based on the Anthos deployment use cases for dev/test, QA, and production environments and increasing workload resource requirements in each environment.

The BOM lists in this appendix are not meant to be exhaustive and must always be double-checked with the configuration tools. Any discussion of pricing, support, and maintenance options is outside the scope of this document.

8.1 BOM for compute servers

8.1.1 Entry configuration

The entry ThinkAgile VX Hybrid configuration is meant for small environments such as SMB, dev/test or a proof-of-concept. The configuration consists of the following servers:

- 1x ThinkAgile Enclosure for VX appliance
- 4x ThinkAgile VX3330 compute node with 2x2.4Ghz 16C CPUs + 256GB memory
- (2x 800GB cache SSDs + 4x 1.2TB capacity HDD) per node
- 2x 10Gbps ethernet ports per node
- 2x 480GB M.2. SSDs per node for OS

Table 9: BOM for entry hardware configuration for Anthos

Part number	Product Description	Qty
7Z62CTO2WW	**Server : Lenovo ThinkAgile VX3330 Appliance**	**1**
BJLH	ThinkAgile VX 1U 2.5" Chassis with 8 or 10 Bays	1
B0W3	XClarity Pro	1
BPZQ	VMware HCI Kit 8 Enterprise (per CPU)	1
BN8K	ThinkAgile VX Remote Deployment	1
BB2Z	Intel Xeon Silver 4314 16C 135W 2.4GHz Processor	2
B964	ThinkSystem 32GB TruDDR4 3200 MHz (2Rx4 1.2V) RDIMM	8
5977	Select Storage devices - no configured RAID required	1
AUNM	ThinkSystem 430-16i SAS/SATA 12Gb HBA	1
B5MD	vSAN Hybrid Config	1
B8HU	ThinkSystem 2.5" PM1645a 800GB Mainstream SAS 12Gb Hot Swap SSD	2
AUM1	ThinkSystem 2.5" 1.2TB 10K SAS 12Gb Hot Swap 512n HDD	4
B5XH	ThinkSystem M.2 SATA 2-Bay RAID Enablement Kit	1
BQ1Y	ThinkSystem M.2 5400 PRO 480GB Read Intensive SATA 6Gb NHS SSD	2
BMEY	VMware ESXi 7.0 U3 (Factory Installed)	1

B8MX	ThinkSystem 1U 10x2.5" (6x SAS/SATA 4x AnyBay) Backplane	1
B8N2	ThinkSystem 1U PCIe Gen4 x16/x16 Riser 1	1
B8NC	ThinkSystem 1U LP+LP BF Riser Cage Riser 1	1
B5SZ	ThinkSystem Broadcom 57414 10/25GbE SFP28 2-port OCP Ethernet Adapter	1
BHS9	ThinkSystem 1100W (230V/115V) v2 Platinum Hot-Swap Power Supply	2
6400	2.8m, 13A/100-250V, C13 to C14 Jumper Cord	2
AUPW	ThinkSystem XClarity Controller Standard to Enterprise Upgrade	1
BH9M	ThinkSystem V2/V3 1U Performance Fan Option Kit	8
B8LA	ThinkSystem Toolless Slide Rail Kit v2	1
B0MK	Enable TPM 2.0	1
B97M	ThinkSystem SR630 V2 MB	1
B0ML	Feature Enable TPM on MB	1
9207	VMWare Specify	1
9220	Preload by Hardware Feature Specify	1
B7Y0	Enable IPMI-over-LAN	1
BK14	Low voltage (100V+)	1
B8KM	ThinkSystem 1U 10x2.5" 6 SAS/SATA+4 AnyBay HDD Type Label	1
B8KY	Thinksystem WW Lenovo LPK	1
B265	ThinkAgile VX Pubkit	1
AUTQ	ThinkSystem small Lenovo Label for 24x2.5"/12x3.5"/10x2.5"	1
AWF9	ThinkSystem Response time Service Label LI	1
B8JY	ThinkSystem 1100W Pt Power Rating Label WW	1
BLAJ	ThinkAgile SR630 V2 Agency label	1
BA22	ThinkSystem SR630 V2 MB to 6SATA/SAS and 4 AnyBay BP Cable 440mm	1
BC4T	ThinkSystem 1U SFF Front 2.5" Backplane SAS/SATA Gen3 Cable	1
BA1W	ThinkSystem V3 1U SFF Front BP SATA/SAS Cable	1
B8GY	M.2 Module Cable	1
BMJC	ThinkSystem 8x2.5" BP and 6+4 x2.5" BP Power Cable v2	1
BE0E	N+N Redundancy With Over-Subscription	1
B13M	ThinkAgile EIA Plate	1
BJDQ	ThinkAgile VX3330 Appliance	1
BH9R	10x2.5" Media Bay w/ Cable	1
B955	ThinkSystem 4R ICX CPU HS Clip	2
B978	ThinkSystem SR630/SR850/SR860 V2 Standard Heatsink	2
BHJS	1U MB PSU Airduct for CPU>125W	1
B989	ThinkSystem V2 1U Package	1
B984	ThinkSystem 1U PLV Top Cover Sponge	1
BJDJ	ThinkAgile SR630 v2 Service Label LI	1
AVEN	ThinkSystem 1x1 2.5" HDD Filler	4
B8NM	ThinkSystem 1U MS Air Duct	1
B8NK	ThinkSystem 1U Super Cap Holder Dummy	1
AUWG	Lenovo ThinkSystem 1U VGA Filler	1

7S06CTOEWW	VMware vSAN	1
S6AR	VMware HCI Kit 8 Enterprise (per CPU) w/Lenovo 3Yr S&S	2

8.1.2 Mid-range configuration

The mid-range configuration is intended for many common workload use cases that do not require ultra-high-performance or a lot of system resources. For example, web servers, analytical databases, NoSQL, node.js, network load-balancers, and other containerized workloads. The mid-range configuration could be a good starting configuration for production environments that will grow over time.

The mid-range configuration consists of the following servers:

- 4x ThinkAgile VX3330 1U compute node with 2x3.1Ghz 16C CPUs + 512 GB memory
- (2x 800GB cache SSDs + 8x 3.84TB SATA capacity SSD) per node
- 4x 10Gbps CAT6 Ethernet ports per node
- 2x 480GB M.2. SSDs per node for OS

Table 10: Mid-range hardware BOM for Anthos

Part number	Product Description	Qty
7Z62CTO2WW	**Server : Lenovo ThinkAgile VX3330 Appliance**	**1**
BJLH	ThinkAgile VX 1U 2.5" Chassis with 8 or 10 Bays	1
B0W3	XClarity Pro	1
BPZQ	VMware HCI Kit 6 Enterprise (per CPU)	1
BN8K	ThinkAgile VX Remote Deployment	1
BB2W	Intel Xeon Gold 6346 16C 205W 3.1GHz Processor	2
B964	ThinkSystem 32GB TruDDR4 3200 MHz (2Rx4 1.2V) RDIMM	16
5977	Select Storage devices - no configured RAID required	1
AUNM	ThinkSystem 430-16i SAS/SATA 12Gb HBA	1
B5MC	vSAN All Flash Config	1
B5XH	ThinkSystem M.2 SATA 2-Bay RAID Enablement Kit	1
B919	ThinkSystem M.2 5300 480GB SATA 6Gbps Non-Hot Swap SSD	2
BMEY	VMware ESXi 7.0 U3 (Factory Installed)	1
B8MX	ThinkSystem 1U 10x2.5" (6x SAS/SATA 4x AnyBay) Backplane	1
B8N2	ThinkSystem 1U PCIe Gen4 x16/x16 Riser 1	1
B8NC	ThinkSystem 1U LP+LP BF Riser Cage Riser 1	1
B5SZ	ThinkSystem Broadcom 57414 10/25GbE SFP28 2-port OCP Ethernet Adapter	1
BHS9	ThinkSystem 1100W (230V/115V) v2 Platinum Hot-Swap Power Supply	2
6400	2.8m, 13A/100-250V, C13 to C14 Jumper Cord	2
AUPW	ThinkSystem XClarity Controller Standard to Enterprise Upgrade	1
BH9M	ThinkSystem 1U Performance Fan Option Kit	8
B8LA	ThinkSystem Toolless Slide Rail Kit v2	1
B97M	ThinkSystem SR630 V2 MB	1
B0ML	Feature Enable TPM on MB	1

B8GY	M.2 Module Cable	1
BMJC	ThinkSystem 8x2.5" BP and 6+4 x2.5" BP Power Cable v2	1
BA22	ThinkSystem SR630 V2 MB to 6SATA/SAS and 4 AnyBay BP Cable 440mm	1
BC4T	ThinkSystem 1U SFF Front 2.5" Backplane SAS/SATA Gen3 Cable	1
BA1W	ThinkSystem SR630 V2 SFF Front BP SATA/SAS Cable	1
BJDQ	ThinkAgile VX3330 Appliance	1
B173	Companion Part for XClarity Controller Standard to Enterprise Upgrade in Factory	1
BJDJ	ThinkAgile SR630 v2 Service Label LI	1
B978	ThinkSystem SR630/SR850/SR860 V2 Standard Heatsink	2
B49C	ThinkSystem 2.5" S4510 3.84TB Read Intensive SATA 6Gb HS SSD	8
B8HU	ThinkSystem 2.5" PM1645a 800GB Mainstream SAS 12Gb Hot Swap SSD	2
7S06CTOEWW	VMware vSAN	1
S6AR	VMware HCI Kit 6 Enterprise (per CPU) w/Lenovo 3Yr S&S	2
5641PX3	XClarity Pro, Per Endpoint w/3 Yr SW S&S	1
1340	Lenovo XClarity Pro, Per Managed Endpoint w/3 Yr SW S&S	1
5WS7A94898	Foundation ThinkAgile APP - 3Yr NBD Resp VX3330	1
5MS7A87711	ThinkAgile VX Remote Deployment (up to 4 node cluster)	1
7S06CTOSWW	VMware vSAN for VX	1

8.1.3 High-performance configuration

The high-performance configuration is intended for workload use cases that do require ultra-high-performance or a lot of system resources. For example, highly transactional database workloads, Web serving, high number of containers, storage intensive applications, AI/ML workloads, and other containerized workloads requiring a lot of system resources. This high-performance configuration uses NVMe flash disks to deliver high IOPS and throughput from the vSAN storage.

The high-performance configuration consists of the following servers:

- 4x ThinkAgile VX7530 2U compute node with 2x2.6Ghz 32C CPUs + 1TB memory
- (2x 800GB NVMe cache SSDs + 8x 6.4TB SAS capacity SSD) per node
- 4x 10Gbps CAT6 Ethernet ports per node
- 2x 480GB M.2. SSDs per node for OS

Table 11: High-performance hardware BOM for Anthos

Part number	Product Description	Qty
7Z63CTO4WW	**Server : Lenovo ThinkAgile VX7530 Appliance**	**1**
BJLK	ThinkAgile VX 2U 2.5" Chassis with 8, 16 or 24 Bays	1
B0W3	XClarity Pro	1
BPZQ	VMware HCI Kit 8 Enterprise (per CPU)	1
BN8K	ThinkAgile VX Remote Deployment	1
BB3R	Intel Xeon Platinum 8358 32C 250W 2.6GHz Processor	2

B966	ThinkSystem 64GB TruDDR4 3200 MHz (2Rx4 1.2V) RDIMM	16
5977	Select Storage devices - no configured RAID required	1
BM51	ThinkSystem 440-8i SAS/SATA PCIe Gen4 12Gb HBA	1
B5MC	vSAN All Flash Config	1
BNF1	ThinkSystem 2.5" U.3 7450 MAX 800GB Mixed Use NVMe PCIe 4.0 x4 HS SSD	2
BP3K	ThinkSystem 2.5" PM1655 6.4TB Mixed Use SAS 24Gb HS SSD	8
B5XH	ThinkSystem M.2 SATA 2-Bay RAID Enablement Kit	1
BQ1Y	ThinkSystem M.2 5400 PRO 480GB Read Intensive SATA 6Gb NHS SSD	2
BMEY	VMware ESXi 7.0 U3 (Factory Installed)	1
BH8B	ThinkSystem 2U/4U 8x2.5" AnyBay Backplane	1
B5SZ	ThinkSystem Broadcom 57414 10/25GbE SFP28 2-port OCP Ethernet Adapter	1
B8LJ	ThinkSystem 2U PCIe Gen4 x16/x8/x8 Riser 1 or 2	1
BQ0W	ThinkSystem V2 1100W (230Vac/115Vac) Platinum Hot Swap Power Supply	2
6400	2.8m, 13A/100-250V, C13 to C14 Jumper Cord	2
AUPW	ThinkSystem XClarity Controller Standard to Enterprise Upgrade	1
BH8E	ThinkSystem 2U Performance Fan	6
B8LA	ThinkSystem Toolless Slide Rail Kit v2	1
BMJ7	ThinkSystem 2U EIA Latch Standard (Left) v2	1
B0MK	Enable TPM 2.0	1
B7Y0	Enable IPMI-over-LAN	1
B97L	ThinkSystem SR650 V2 MB	1
B0ML	Feature Enable TPM on MB	1
BK14	Low voltage (100V+)	1
AUTQ	ThinkSystem small Lenovo Label for 24x2.5"/12x3.5"/10x2.5"	1
B7KL	ThinkSystem OCP NIC Label 1-2	1
B8JY	ThinkSystem 1100W Pt Power Rating Label WW	1
AWF9	ThinkSystem Response time Service Label LI	1
B265	ThinkAgile VX Pubkit	1
BLAK	ThinkAgile SR650 V2 Agency label	1
BMPF	ThinkSystem V3 2U Power Cable from MB to Front 2.5" BP v2	1
BH8R	ThinkSystem SR650 V2 MB PCIe 3/4 to Front 8x2.5" BP1 (NVMe4-7)SL	1
BH8M	ThinkSystem SR650 V2 MB PCIe 1/2 to Front 8x2.5" BP1 (NVMe0-3)SL	1
BETS	ThinkSystem V3 2U SFF C0 (RAID) to Front 8x2.5" BP1	1
BMJS	ThinkSystem 2U M.2 Cable v2	1
BE0E	N+N Redundancy With Over-Subscription	1
BBAL	G4 x16/x8/x8 PCIe Riser B8LJ for Riser 1 Placement	1
BJDM	ThinkAgile VX7530 Appliance	1
B13M	ThinkAgile EIA Plate	1
AVEN	ThinkSystem 1x1 2.5" HDD Filler	1
AVEQ	ThinkSystem 8x1 2.5" HDD Filler	2
B955	ThinkSystem 4R ICX CPU HS Clip	2
B977	ThinkSystem SR650 V2 Performance Heatsink	2

B8MP	ThinkSystem 2U MS Air Duct Filler(For 2U Gap)	2
B173	Companion Part for XClarity Controller Standard to Enterprise Upgrade in Factory	1
B986	ThinkSystem HV 2U WW General PKG BOM	1
BHJN	2U MB PSU Airduct	1
BC4X	MS 2FH Riser Filler	1
BHWJ	ThinkSystem 2U MS 3FH Riser1 Cage v2	1
B8MM	ThinkSystem 2U MS 3FH Riser Filler	1
BJDG	ThinkAgile SR650 v2 Service Label LI	1
BMJ8	ThinkSystem 2U EIA Latch with FIO (right) v2	1
7S06CTOEWW	VMware vSAN	1
S6AR	VMware HCI Kit 8 Enterprise (per CPU) w/Lenovo 3Yr S&S	2
5641PX3	XClarity Pro, Per Endpoint w/3 Yr SW S&S	1
1340	Lenovo XClarity Pro, Per Managed Endpoint w/3 Yr SW S&S	1

Resources

1. Google Cloud’s Anthos product page
   https://cloud.google.com/anthos/
2. GKE on-prem product page
   https://cloud.google.com/gke-on-prem/
3. Google Cloud’s Anthos central documentation
   https://cloud.google.com/anthos/clusters/docs/on-prem/latest/how-to/install-overview
4. VMware vSAN technical document repository
   https://storagehub.vmware.com/t/vmware-vsan/
5. Lenovo ThinkAgile VX product page
   https://www.lenovo.com/us/en/data-center/software-defined-infrastructure/ThinkAgile-VX-Series/p/WMD00000340
6. Lenovo Press for ThinkAgile VX series
   https://lenovopress.com/servers/thinkagile/vx-series

Document history

Date	Version No	Changes
8/20/2019		First version of Google Cloud’s Anthos reference architecture on ThinkAgile VX.
8/26/2019		Formatting and name updates.
6/11/2020		Added updates for Anthos version 1.3.0.
03/22/2023		• Added Section 6.1.3 Lenovo ThinkAgile VX with 3rd Generation of Intel Scalable Processors • BOM is updated with ThinkAgile VX (3rd Generation Intel Xeon SP) • Removed references to withdrawn Lenovo ToR switches on Section 6.2.2, 8 • Updated Anthos version to 1.12 • Added high-level Anthos deployment steps • Updated drawings to reflect above changes

Trademarks and special notices