WHITE PAPER

GOOGLE ANTHOS EXCELS ON SUPERMICRO® SUPERBLADE®

Simplify hybrid and multi-cloud environments with GDC Virtual on SuperBlade

Supermicro SuperBlade with AMD EPYC™ processors powers Google Anthos deployments on-premises

Table of Contents

Executive Summary

The transformative impact of the cloud on businesses has prompted a rapid migration and adoption of cloud-first strategies. As a result, many enterprises are looking into application modernization strategies to meet customer expectations, keep business operations agile, and accelerate innovation. Cloud-native application development empowers enterprises to capitalize on the full power of the cloud by delivering faster time to market, improved efficiency, increased scalability, and better consumer experiences while optimizing cost compared to legacy, on-premise development infrastructures. A cloud-native approach accelerates the application development lifecycle by consuming independent components called microservices, which break large monolithic applications into smaller components.

Cloud-native technologies empower organizations to build and run scalable applications in modern, dynamic public, private, and hybrid cloud environments. Modern cloud-native applications evolve rapidly; leveraging a hybrid cloud platform helps integrate variants of applications across different cloud platforms and/or on-prem environments.

Google Anthos™, now known as Google Distributed Cloud Virtual (GDC Virtual), is a hybrid cloud management platform that manages containers across multiple cloud platforms and legacy VM workloads.

Businesses can leverage the benefits of GDC Virtual by deploying it on a self-managed Supermicro® SuperBlade® environment to directly run cloud-native applications with CPU, GPU, and other hardware resources.

Drivers of On-Premises Cloud

The cloud has recently become popular for deploying new and existing applications. The simplicity and ease of using a service can offer significant time and cost optimization compared to building your own infrastructure. Many organizations are now adopting cloud-first strategies to leverage these advantages.

Cloud-native approaches are also increasing in popularity and encompass a broadening array of use cases from refactoring existing applications and/or creating container-based microservices architectures. The cloud can even run legacy applications while offering cost savings compared to on-prem deployment models.

Expanding cloud use cases empowers cloud-first organizations to run most of their application landscapes in the cloud, with fewer non-migratable exceptions. However, there is no one simple "lift-and-shift strategy" that meets the needs of every app. Thus, no large organization currently relies solely on modern, cloud-based applications; they must work the cloud into and around existing systems and applications, some of which do not support immediate migration to the cloud. These applications are often the mission-critical systems at the heart of an organization's commercial operations.

The key drivers behind deciding to deploy an on-premises cloud may include:

- **Data security, compliance, and data sovereignty requirements:** Data sovereignty, security, and/or similar restrictions in place because of laws, security policies, or compliance rules may prohibit an application from running in the public cloud. This prescription often extends to Personally Identifiable Information (PII), medical, and/or other sensitive data.

- **Monolithic application design:** Some legacy application architectures don't align with cloud computing pricing models and the resulting costs and timelines of refactoring prevent organizations from migrating to the cloud.

- **Demand for very low networking latency:** Highly transactional. Low-latency application systems (e.g., banking or transportation) take an unacceptable performance hit if they are too far away from their users, data, or the next-hop data processor in the application flow.

- **Protecting legacy infrastructure investments: Enterprises often look to optimize costs by** leveraging their existing data center investments in servers, networking equipment, and storage devices. Migrating from CapEx to OpEx is simply not viable when the application is economically best served on existing on-premises infrastructure.

A recent IDC survey[1] reveals that half of the spending on server and storage infrastructure supports on-prem deployments. As shown in Figure 1, 71% of respondents expected to partially or fully move their workloads from a public cloud to a dedicated

[1] https://www.supermicro.com/white_paper/IDC_On-Prem_Cloud_Success_Stories.pdf

IT environment in the next two years. IDC expects these investments to continue to grow at a compound annual growth rate of 2.9% for the next five years and reach $77.5 billion in 2026. Organizations keep investing in on-premises IT infrastructure because of control, cost, and predictability.

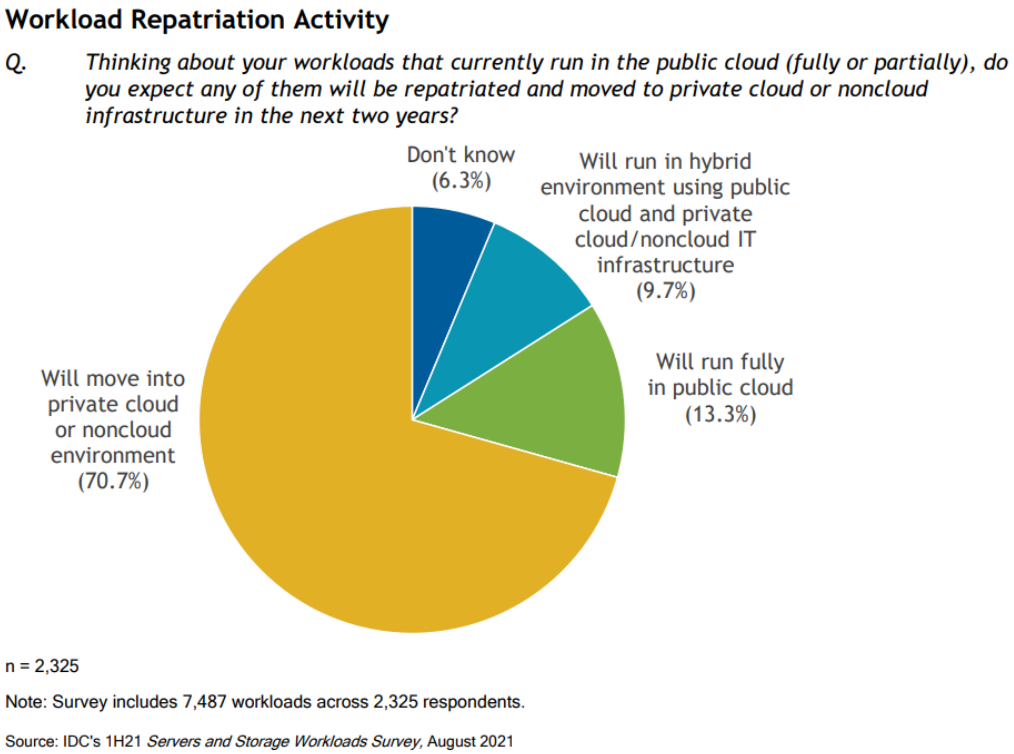


Figure 1 – Workload Repatriation Activity.

One application modernization strategy leverages on-premises cloud solutions such as GDC Virtual to extend cloud capabilities and services to an on-premises data center. This allows the application to leverage many of the advantages of cloud computing while maintaining consistent operations across locations.

GDC Virtual is especially interesting to organizations already using Google Cloud Platform while seeking a seamless solution for integrating their remaining on-premises applications into their cloud operations framework. GDC Virtual empowers organizations to derive the benefits of cloud-based architectures while significantly reducing the costs associated with a DIY approach across operations, lifecycle management, monitoring, and visibility—all traditionally difficult aspects of IT operations.

GDC Virtual can deploy both traditional and cloud-native applications. Figure 2 shows a single GDC Virtual cluster supporting deployments across multiple cloud platforms, such as Amazon AWS, Google Cloud, and Microsoft Azure. GDC Virtual also includes a bare metal deployment option that delivers many cloud benefits to self-managed Supermicro SuperBlade servers.

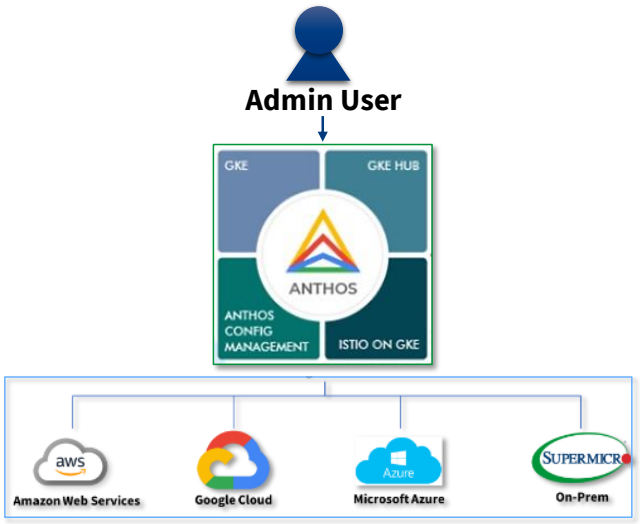


Figure 2 – Sample Google GDC Virtual Deployment.

Supermicro SuperBlade Product Overview

Supermicro SuperBlade servers deliver advanced Bare-Metal-as-a-Service (BMaaS) solutions for the global multi-cloud (hybrid, private), AI inference, visual computing, Data Center, Enterprise, Big Data, and HPC markets. SuperBlade is an ideal platform to deploy Google Anthos for bare metal deployments. The following sections below form a solution stack that allows you to deploy a cloud-managed platform using Supermicro SuperBlade servers powered by Google Anthos. IT administrators can follow the procedure outlined in this section to deploy Google Anthos on bare metal on their on-prem environments.

SUPERMICRO AMD SUPERBLADE

8U SuperBlade Enclosure

Supermicro's high performance, density optimized, and energy-efficient SuperBlade® can significantly reduce initial capital and operational expenses.

The Supermicro SuperBlade is powered by AMD EPYC™ Processors
Supermicro's new-generation blade portfolio helps optimize the TCO of key data center components, such as cooling, power efficiency, node density, and networking management. Supermicro SuperBlade, powered by 3rd Gen AMD EPYC processors, is built for the most demanding workloads that require high CPU density and the fastest networking available today in a trusted platform that meets enterprise customer demands for on-prem private/hybrid cloud deployments.

The Supermicro SuperBlade comes in an 8U enclosure that accommodates up to 20 hot-pluggable single socket nodes and delivers high performance with AMD EPYC 7003 Series Processors with AMD 3D V-Cache™ technology**,** DDR4 3200MHz memory, and fast PCIe® Gen4 I/O. Supermicro offers three SuperBlade models powered by AMD EPYC processors: a SAS model, a SATA model, and a GPU-accelerated model, all of which can be mixed in a single 8U enclosure. The 8U

SUPERMICRO AMD SUPERBLADE MODELS

SBA-4114S-C2N/T2N
OCP 3.0 Mezzanine Cards

SBA-4119SG
GPU /PCIe Cards

SuperBlade can support up to 40 single-width GPUs or 20 double-width GPUs. SuperBlade SAS/SATA models support AIOM for front I/O, which extends the Open Compute Project 3.0 specification to support a wide range of networking options in a small form factor. The 8U SuperBlade also provides customers with advanced networking options, such as 200G HDR InfiniBand and 25G Ethernet switches.

Each SuperBlade enclosure contains at least one Chassis Management Module (CMM), which allows administrators to remotely manage and monitor server blades, power supplies, cooling fans, and networking switches. SuperCloud Composer (SCC) is a composable cloud management platform that provides a unified dashboard for administering software-defined data centers. SCC can orchestrate cloud workloads via the streamlined industry-standard Redfish API. SCC can also monitor and manage a broad portfolio of multi-generation Supermicro servers from a single pane of glass, including SuperBlade.

The AMD EPYC 7003 Series Processors with AMD 3D V-Cache technology shown below are built around the "Zen3" core and contain up to 64 cores per socket, delivering breakthrough per-core performance with 3X the L3 Cache (768 MBs per socket) of general purpose AMD EPYC 7003 CPUs. The Supermicro SuperBlade, powered with AMD EPYC 7003 processors, enables exceptional HPC performance thanks to high frequencies, core counts, high memory bandwidth and capacity, and 768MB of L3 cache.

Processor	Cores	TDP (watts)	Base Freq/Max Boost Freq	Total L3 Cache	DDR Channels
7373X	16	240	3.05 GHz / 3.8 GHz	768MB	8
7473X	24	240	2.8 GHz / 3.7 GHz	768MB	8
7573X	32	280	2.8 GHz / 3.6 GHz	768MB	8
7773X	64	280	2.2 GHz / 3.5 GHz	768MB	8

Supermicro GPU SuperBlade SBA-4119SG supports 3rd Gen AMD 7003 Series EPYC with 3D V-Cache technology processors and the AMD Instinct™ MI210 accelerators. These GPU-accelerated SuperBlade servers are ideal for running AI inference, visual computing, and HPC workloads. SuperBlade systems help organizations reduce time-to-solution for a wide range of applications, add advanced security features, and allow all workloads to run either on-prem or in a public or private cloud. Supermicro SuperBlade offers high density, excellent performance, high power efficiency, and low Total Cost of Ownership (TCO).

Solution Deployment Overview

GDC Virtual on bare metal brings Google Kubernetes Engine (GKE) to your on-premises data centers and allows you to build cloud-native workloads and manage legacy VMs. You can also leverage GDC Virtual to create, manage, and upgrade Kubernetes clusters in your data center. The deployment described in this white paper uses GDC Virtual 1.13.1 on bare metal.

GDC Virtual on bare metal supports various cluster types:

- **Admin Cluster:** For cluster administration only.
- **User Cluster:** For running workloads only.
- **Hybrid Cluster:** For administration and running workloads, and can also manage other user clusters.
- **Standalone Cluster:** For administration and running workloads, but cannot manage other user clusters.

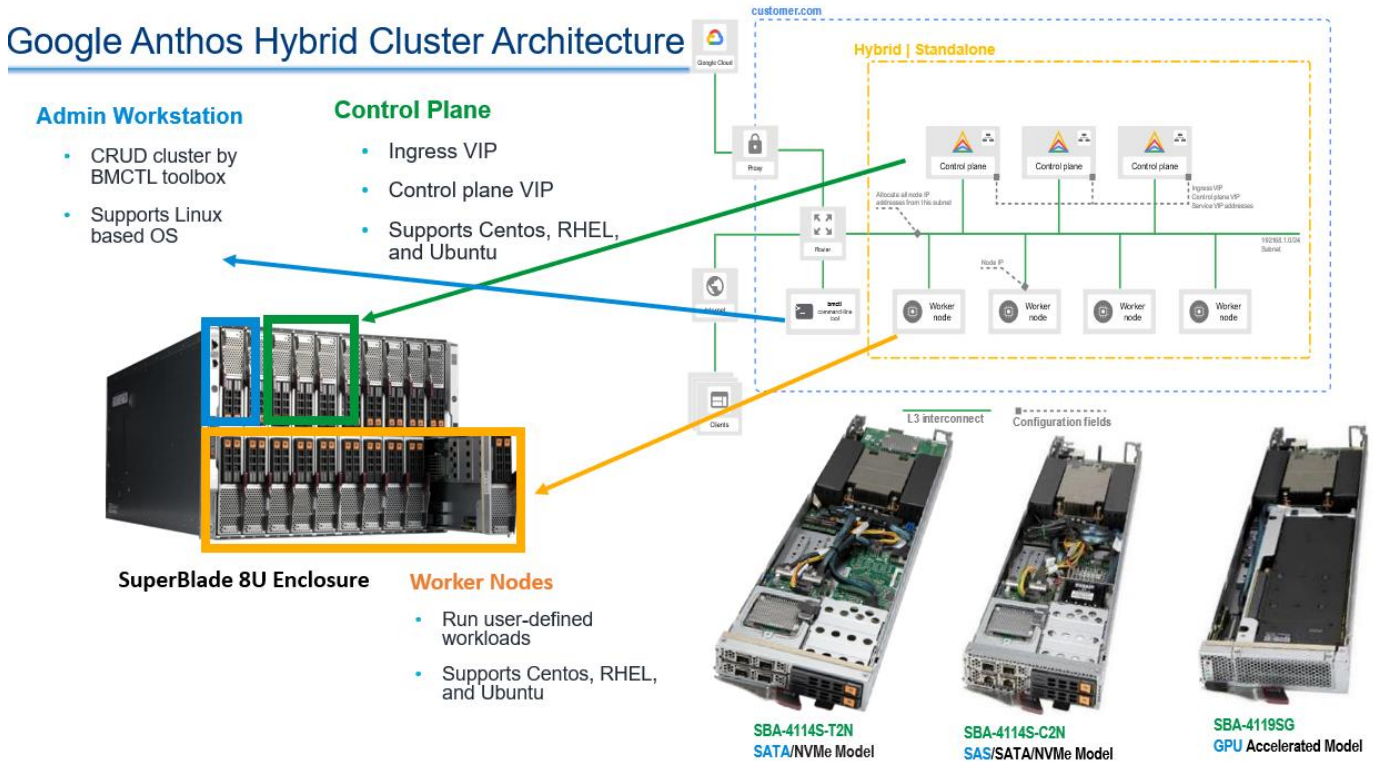


Figure 3 – Google GDC Virtual Cluster Architecture deployment on Supermicro SuperBlade.

Figure 3 shows a typical GDC Virtual hybrid cluster architecture deployment on a Supermicro SuperBlade platform where customers can choose SAS, SATA, and GPU accelerated SuperBlade nodes in an 8U enclosure. SuperBlade offers a "private cloud in a box feature" where you can mix and match CPU-only and GPU accelerated nodes in the same 8U enclosure. Customers can use SuperBlade to deploy the admin workstation that hosts command-line interface (CLI) tools and configuration files to provision clusters during installation and CLI tools for interacting with provisioned clusters post-installation. SuperBlade can also host the Control Plane node, including the Kubernetes API server, etcd storage, and other controllers. You can host the worker nodes that run the actual cloud native applications on CPU- and/or GPU-powered SuperBlade as needed for your workload(s).

The installation prerequisites are:

- **Supermicro SuperBlade:** The deployment described in this white paper requires five (5) SuperBlade servers. Table 1 lists the minimum and recommended requirements. This example uses the SBA-4119SG as a worker node. Table 2 lists the worker node hardware specifications.
- **GCP project:** The sample project described in this whitepaper has billing enabled on the GCP Platform.
- **Linux Administration Workstation:** A workstation used to deploy the GDC Virtual cluster. This example uses Ubuntu® 20.04.
- **Network:** The Admin Workstation and cluster worker nodes must be able to access Google Cloud APIs (e.g., *.googleapis.com).
- **Worker Node Operating System** – GDC Virtual supports supports CentOS (8.2, 8.3, 8.4, 8.5), RHEL (8.2, 8.3, 8.4, 8.5, 8.6), and Ubuntu (18.04, 20.04)

Resource	Minimum	Recommended
CPUs / vCPUs	4 cores	8 cores
RAM	16GB	32GB
Storage	128GB	256GB

Table 1 - Minimum and recommended worker node hardware requirements

WORKER NODE Sample Config			
Part Type	**Part #**	**Part Description**	**Qty**
System	SBA-4119SG	Supermicro SuperBlade (AMD Powered Single-Socket GPU accelerated model)	1
Processor	PSE-MLN7473X-0507	3rd Gen AMD EPYC 7003 Series with AMD 3D V-Cache technology Processor (7473X - 24 Cores, 48T, 2.8GHz, 240W TDP)	1
GPU	GPU-AMDMI210-001	AMD Instinct MI210, 64GB, 300W HBM2e PCIe Gen4	1
Memory	MEM-DR464L-SL02-ER32	64GB DDR4-3200 2Rx4 LP (16Gb) ECC RDIMM	8
Storage	HDS-IMN0-SSDPELKX020T8	2 TB M.2 NVMe SSD	1

Table 2 – A typical worker node deployed to support AI Inference workload on a GPU accelerated SuperBlade (SBA-4119SG)

Worker Node Prerequisites

The worker nodes must meet the following prerequisites:

- **Operating System:** See Select your operating system. Ubuntu 18.04 requires Linux kernel 5.4 or above.
- **Hardware Components:** See Table 1.

- **Connectivity:** Layer 3 to all other node machines.
- **NTP:** Connected to NTP services.
- **Package Manager:** apt or dnf can access the network.
- **API Access:** *.googleapis.com.
- **Load Balancers:** Must be in the same Layer 2 subnet.
- **Disable ufw:** GDC Virtual requires disabling `ufw` on Ubuntu nodes by executing the command `sudo systemctl stop ufw; sudo systemctl disable ufw` on each Ubuntu node.
- **RHEL:** Register RHEL nodes with RedHat and disable `firewalld` by executing the command `sudo systemctl stop firewalld; sudo systemctl disable firewalld`.
- **Disable SELinux:** Disable SELinux on RHEL nodes. Enabling SELinux may cause container creation/run failure. In RHEL: 8, edit `/etc/selinux/config` file by changing `SELINUX=enforcing` to `SELINUX=disabled`, as shown in Figure 4. Next, execute the command `sudo reboot` to complete the configuration. Verify that executing `getenforce` shows `Disable` after reboot.

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disable
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Figure 4 – Disabling SELinux in RHEL 8

You must configure the SSH server to grant keyless login by the Administrator Workstation to install the required packages. To do this:

- Set `PermitRootLogin yes` in `/etc/ssh/sshd_config`, as shown in Figure 5.
- Restart the SSH service.
  ```
  sudo systemctl restart ssh
  ```
- Retype the password for enabling `PermitRootLogin`
  ```
  sudo passwd
  ```

```
#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes
```

Figure 5 – Granting root login permission

Admin Workstation Prerequisites

The Admin Workstation creates, manages, and upgrades clusters. Table 3a lists the minimum and recommended Admin Workstation hardware requirements, and Table 3b lists the SuperBlade sample workstation configuration.

Resource	Minimum	Recommended
CPUs / vCPUs	2 cores	4 cores
RAM	Ubuntu: 4 GB CentOS/RHEL: 6 GB	Ubuntu: 8 GB CentOS/RHEL: 12 GB
Storage	128 GB	256 GB

Table 3a - Minimum and recommended Admin Workstation hardware requirements

ADMIN WORKSTATION Sample Config			
Part Type	**Part #**	**Part Description**	**Qty**
System	SBA-4114S-T2N	Supermicro SuperBlade (AMD Powered Single-Socket SATA model)	1
Processor	PSE-MLN7473X-0507	3rd Gen AMD EPYC 7003 Series with AMD 3D V-Cache technology Processor (7473X - 24 Cores, 48T, 2.8GHz, 240W TDP)	1
Memory	MEM-DR464L-SL02-ER32	64GB DDR4-3200 2Rx4 LP (16Gb) ECC RDIMM	8
Storage	HDS-IMN0-SSDPELKX020T8	2 TB M.2 NVMe SSD	2
	HDS-TUN0-KCD6XLUL1T92	1.92 TB U.2 PCIe4 NVMe SSD	2

Table 3b – A typical admin node deployment using AMD powered SuperBlade

The Admin Workstation requires access to both Google Cloud and all cluster nodes to install and run `bmctl` for cluster deployments:

A. Operating System: Verify that your operating system meets the requirements described in Select the operating system. (https://cloud.google.com/anthos/clusters/docs/bare-metal/1.13/installing/os-reqs).

B. Docker Installation: The Administrator Workstation requires Docker for kubectl in in order to manage containers.

```
# sudo apt update
# sudo apt install docker.io
# sudo groupadd docker
# sudo apt install kubectl
# sudo usermod -aG docker $USER
# newgrp docker // log out and log in to SSH to enable passwordless Docker
```

C. Google Cloud CLI Installation: Install google-cloud-cli to communicate with Google Cloud APIs for project registration.

```
# sudo apt-get install apt-transport-https ca-certificates gnupg
# echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg] https://packages.cloud.google.com/apt
cloud-sdk main" | sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list
# curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key --keyring
/usr/share/keyrings/cloud.google.gpg add -
# sudo apt-get update && sudo apt-get install google-cloud-cli
```

D. Google Cloud Configuration: Login to Google Cloud and configure your project information.

- Execute `gcloud auth application-default login`, copy the URL to your browser to grant the permissions as shown in Figure 6, and then enter the token on the Administrator Workstation.

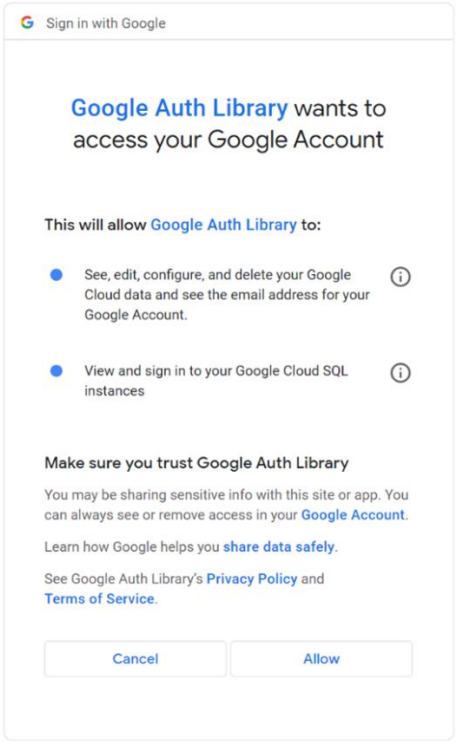


Figure 6 – Granting permissions to Google Cloud

- Set the Google Cloud project quota.

```
#gcloud auth application-default set-quota-project your_project_id
```

E. Create Project: Download bmctl to create a project configuration file. The bmctl version must be consistent with the GDC Virtual cluster. This example uses Google Anthos 1.13.1 on bare metal.

```
# mkdir ~/project_home
# cd ~/project_home
# gsutil cp gs://anthos-baremetal-release/bmctl/1.13.1/linux-amd64/bmctl .
# chmod a+x bmctl
# ./bmctl create config -c your_project_id --enable-apis    --create-service-accounts --project-
id=your_project_id
```

F. Keyless SSH Setting: GDC Virtual on bare metal configures the worker node environment via keyless SSH. Set up keyless SSH before creating the cluster. Replace the sample ID addresses below with your actual IP addresses.

```
# ssh-keygen -t rsa -f ~/.ssh/anthos-certification.key -C "Anthos Certification"
# ssh-copy-id -i ~/.ssh/anthos-certification.key.pub root@172.17.43.169
# ssh-copy-id -i ~/.ssh/anthos-certification.key.pub root@172.17.43.65
# ssh-copy-id -i ~/.ssh/anthos-certification.key.pub root@172.17.43.191
```

```
# ssh-copy-id -i ~/.ssh/anthos-certification.key.pub root@172.17.43.55
# ssh-copy-id -i ~/.ssh/anthos-certification.key.pub root@172.17.43.63
```

Network Prerequisites

GDC Virtual clusters on bare metal retrieve cluster components from the Google Cloud Container Registry and are also registered with Google Cloud Connect Agent. If your Administrator Workstation/worker node network environment is behind a proxy server, then your proxy server must allow the specific connections shown in Table 4.

Address
*.gcr.io
accounts.google.com
cloudresourcemanager.googleapis.com
compute.googleapis.com
connectgateway.googleapis.com
dl.fedoraproject.org
download.docker.com
gkeconnect.googleapis.com
gkehub.googleapis.com
gkeonprem.googleapis.com
gkeonprem.mtls.googleapis.com
iam.googleapis.com
iamcredentials.googleapis.com
logging.googleapis.com
monitoring.googleapis.com
packages.cloud.google.com
oauth2.googleapis.com
opsconfigmonitoring.googleapis.com
securetoken.googleapis.com
servicecontrol.googleapis.com
serviceusage.googleapis.com
stackdriver.googleapis.com
storage.googleapis.com
sts.googleapis.com
www.googleapis.com

Table 4 - Proxy allowed list requirements

Figure 7 shows the logical configuration of a solution stack with a non-HA control plane, which includes 1 control plane and 4 worker nodes to run user-defined workloads. Conversely, creating a hybrid cluster with a HA control plane requires at least 3 control plane nodes.

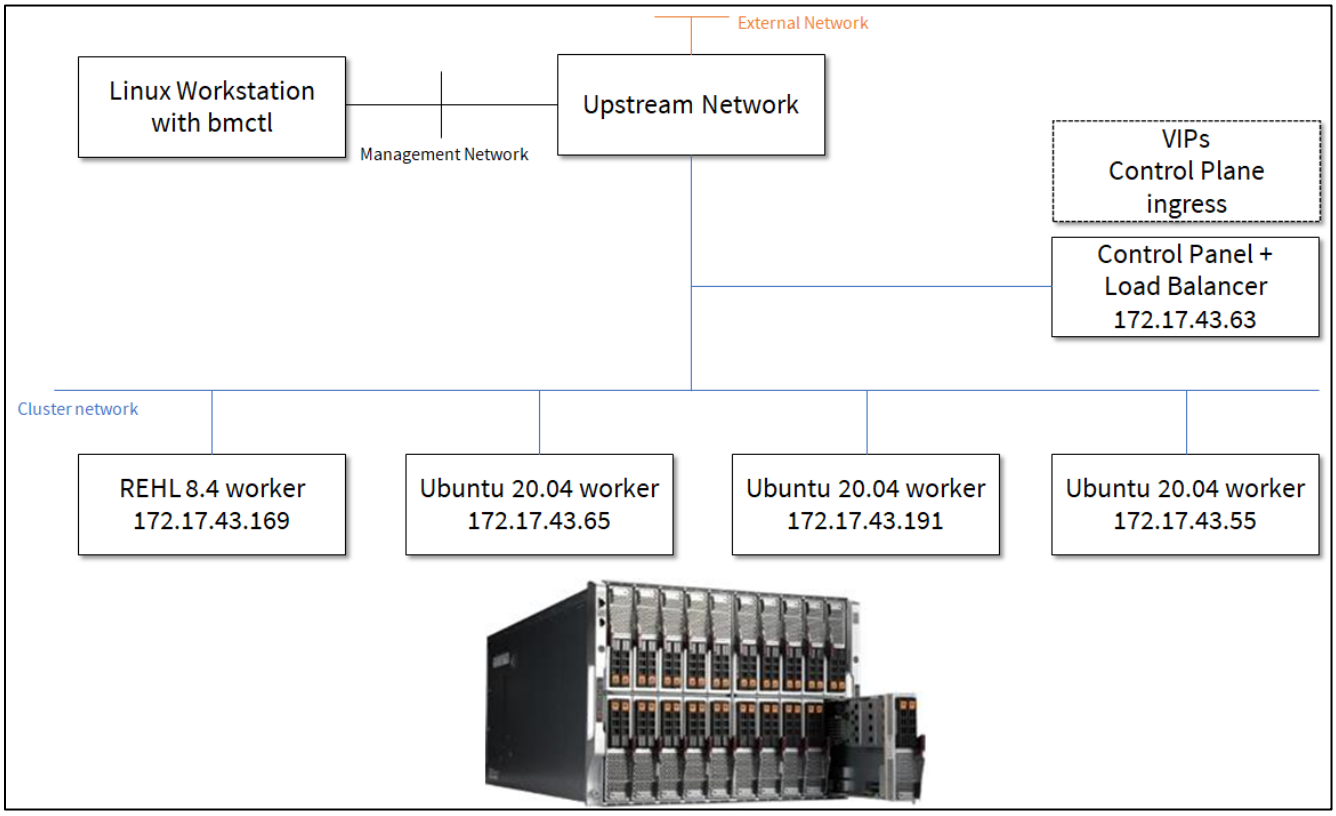


Figure 7 - Non-HA control plane

Worker nodes only require Layer 3 connectivity, but control plane load balancers require Layer 2 connectivity. Load balancer nodes can be either the control plane nodes or a dedicated set of nodes. Load balancer nodes must meet the following requirements:

- All load balance worker nodes for a given cluster must be in the same Layer 2 domain.
- All virtual IP addresses are in the load balancer machine subnet and routable to the subnet's gateway.
- Users are responsible for allowing the ingress of load balancer traffic.

Cluster Deployment

Verify that all prerequisites have been completed.

Edit the configuration file for deployment. This file was created by `bmctl` in the previous section E and saved as `~/project_home/bmctl-workspace/your_project_id/your_project_id.yml`. Open the file and then edit it as follows:

A. Locate the SSH key path: Set the key path for keyless SSH for environment preparation.

```
gcrKeyPath: bmctl-workspace/.sa-keys/your_project_id-anthos-baremetal-gcr.json
sshPrivateKeyPath: /home/smci/.ssh/anthos-certification.key
gkeConnectAgentServiceAccountKeyPath: bmctl-workspace/.sa-keys/your_project_id-anthos-baremetal-connect.json
gkeConnectRegisterServiceAccountKeyPath: bmctl-workspace/.sa-keys/your_project_id-anthos-baremetal-register.json
cloudOperationsServiceAccountKeyPath: bmctl-workspace/.sa-keys/your_project_id-anthos-baremetal-cloud-ops.json
```

B. Meta Data Setting: Set the name of your hybrid cluster.

```
metadata:
 name: your_project_id
 namespace: cluster_your_project_id
```

C. Cluster Type: Cluster types include admin, user, hybrid, and standalone. This example deploys a hybrid cluster.

```
type: hybrid
```

D. Version of Google Anthos on bare metal: The cluster version must equal bmctl.

```
anthosBareMetalVersion: 1.13.1
```

E. Google Kubernetes Engine (GKE) Connect: Set the project ID for Google Kubernetes Engine (GKE).

```
gkeConnect:
  projectID: your_project_id
```

F. Control Plane Node Pool: Set the IP addresses of the control plane node pool. This example only deploys one control plane node. If you are deploying HA, then verify Layer 2 connectivity.

```
controlPlane:
 nodePoolSpec:
  nodes:
  # Control plane node pools. Typically, this is either a single machine.
  # or 3 machines if using a high availability deployment.
  - address: 172.17.43.63
```

G. Cluster Networking: Specify the IP ranges for Kubernetes pods and services.

```
clusterNetwork:
 # Pods specify the IP ranges from which pod networks are allocated.
 pods:
  cidrBlocks:
  - 192.168.0.0/16
```

```
  # Services specify the network ranges from which service virtual IPs are allocated.
  # This can be any RFC1918 range that does not conflict with any other IP range
  # in the cluster and node pool resources.
  services:
   cidrBlocks:
   - 10.96.0.0/20
```

H. Load Balancer: Load balancers can be 'bundled' or 'manual'. This example uses 'bundled' load balancers. The control plane Virtual IP address(es) must not be in the address pool, but the ingress Virtual IP address must be in the address pool.

```
loadBalancer:
  # Load balancer mode can be 'bundled' or 'manual'.
  # In 'bundled' mode a load balancer will be installed on load balancer nodes during cluster creation.
  # In 'manual' mode the cluster relies on a manually-configured external load balancer.
  mode: bundled
  # Load balancer port configuration
  ports:
   # Specifies the port the load balancer serves the Kubernetes control plane on.
   # In 'manual' mode the external load balancer must be listening on this port.
   controlPlaneLBPort: 443
  # There are two load balancer virtual IP (VIP) addresses: one for the control plane
  # and one for the L7 Ingress service. The VIPs must be in the same subnet as the load balancer nodes.
  # These IP addresses do not correspond to physical network interfaces.
  vips:
   # ControlPlaneVIP specifies the VIP to connect to the Kubernetes API server.
   # This address must not be in the address pools below.
   controlPlaneVIP: 172.17.43.221
   # IngressVIP specifies the VIP shared by all services for ingress traffic.
   # Allowed only in non-admin clusters.
   # This address must be in the address pools below.
   ingressVIP: 172.17.43.222
  # AddressPools is a list of non-overlapping IP ranges for the data plane load balancer.
  # All addresses must be in the same subnet as the load balancer nodes.
  # Address pool configuration is only valid for 'bundled' LB mode in non-admin clusters.
```

```
  addressPools:
  - name: pool1
    addresses:
  #   # Each address must be either in the CIDR form (1.2.3.0/24)
  #   # or range form (1.2.3.1-1.2.3.5).
    - 172.17.43.222-172.17.43.225
  # A load balancer node pool can be configured to specify nodes used for load balancing.
  # These nodes are part of the Kubernetes cluster and run regular workloads as well as load balancers.
  # If the node pool config is absent then the control plane nodes are used.
  # Node pool configuration is only valid for 'bundled' LB mode.
  # nodePoolSpec:
  #  nodes:
  #  - address: <Machine 1 IP>
```

I. **Cluster Operations:** Configuration for logs and metrics.

```
clusterOperations:
  # Cloud project for logs and metrics.
  projectID: your_project_id
  # Cloud location for logs and metrics.
  location: us-central1
  # Whether collection of application logs/metrics should be enabled (in addition to
  # collection of system logs/metrics which correspond to system components such as
  # Kubernetes control plane or cluster management agents).
  # enableApplication: false
```

J. **Worker Nodes:** Set the pool of worker nodes.

```
# Node pools for worker nodes
apiVersion: baremetal.cluster.gke.io/v1
kind: NodePool
metadata:
  name: node-pool-1
  namespace: cluster_your_project_id
spec:
  clusterName: your_project_id
```

```
nodes:
- address: 172.17.43.169
- address: 172.17.43.191
- address: 172.17.43.55
- address: 172.17.43.65
```

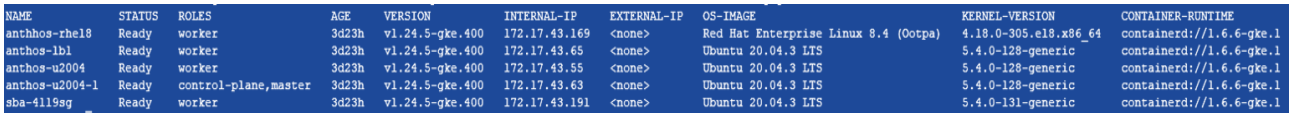
Begin deploying the GDC Virtual on the configured bare metal cluster.

A. Create Cluster: Use `bmctl` to create the cluster with the configuration file you edited above. This action will take several minutes.

```
./bmctl create cluster -c your_project_id
```

B. Verify Cluster: Print the node status, as shown in Figure 8.

```
# kubectl --kubeconfig ~/project_home/bmctl-workspace/your_project_id/your_project_id-kubeconfig
get nodes -o wide
```



NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	KERNEL-VERSION	CONTAINER-RUNTIME
anthhos-rhel8	Ready	worker	3d23h	v1.24.5-gke.400	172.17.43.169	<none>	Red Hat Enterprise Linux 8.4 (Ootpa)	4.18.0-305.el8.x86_64	containerd://1.6.6-gke.1
anthos-1b1	Ready	worker	3d23h	v1.24.5-gke.400	172.17.43.65	<none>	Ubuntu 20.04.3 LTS	5.4.0-128-generic	containerd://1.6.6-gke.1
anthos-u2004	Ready	worker	3d23h	v1.24.5-gke.400	172.17.43.55	<none>	Ubuntu 20.04.3 LTS	5.4.0-128-generic	containerd://1.6.6-gke.1
anthos-u2004-1	Ready	control-plane,master	3d23h	v1.24.5-gke.400	172.17.43.63	<none>	Ubuntu 20.04.3 LTS	5.4.0-128-generic	containerd://1.6.6-gke.1
sba-4119sg	Ready	worker	3d23h	v1.24.5-gke.400	172.17.43.191	<none>	Ubuntu 20.04.3 LTS	5.4.0-131-generic	containerd://1.6.6-gke.1

Figure 8 – Output of Node Status

Login to Google Kubernetes Engine Cluster

Google Kubernetes Engine manages the cluster on the Google Cloud Platform. You can use GKE to deploy workloads and use additional applications. You can authenticate with GKE clusters via Google Identity, OIDC identity provider, or bearer tokens. This example uses bearer tokens. Obtain the required bearer tokens by creating a Kubernetes service account (KSA) in the cluster and then using its bearer token to log in.

A. Grant roles for Google Cloud console: You must grant `roles/container.viewer` and `roles/gkehub.viewer` to interact with connected clusters.

```
# gcloud projects add-iam-policy-binding your_project_id --
member=user:your_persional_email@email.address --role=roles/container.viewer
# gcloud projects add-iam-policy-binding your_project_id --member=user:your_persional_email@
email.address --role=roles/gkehub.viewer
```

B. Create and apply cloud-console-reader RBAC role: Create and apply the cloud-console-reader ClusterRole to view your cluster's resources in the Google Cloud console.

```
# cat <<EOF > cloud-console-reader.yaml
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: cloud-console-reader
  rules:
  - apiGroups: [""]
    resources: ["nodes", "persistentvolumes", "pods"]
    verbs: ["get", "list", "watch"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["storageclasses"]
    verbs: ["get", "list", "watch"]
EOF
# kubectl apply -f cloud-console-reader.yaml --kubeconfig ~/project_home/bmctl-workspace/
your_project_id /your_project_id-kubeconfig
```

C. Service Account Setting: A bearer token is like a password. Create your own account and restrict operations by the RBAC roles held by the KSA. Please replace user_account with your user account.

```
# kubectl create serviceaccount user_account --kubeconfig ~/project_home/bmctl-
workspace/your_project_id/your_project_id-kubeconfig
# kubectl create clusterrolebinding user_account --clusterrole view --serviceaccount
default:user_account --kubeconfig ~/project_home/bmctl-workspace/your_project_id/your_project_id-
kubeconfig
# kubectl create clusterrolebinding user_accountcloud --clusterrole cloud-console-reader --
serviceaccount default:user_account  --kubeconfig ~/project_home/bmctl-
workspace/your_project_id/your_project_id-kubeconfig
# kubectl create clusterrolebinding user_accountcluster --clusterrole cluster-admin --serviceaccount
default:user_account   --kubeconfig ~/project_home/bmctl-
workspace/your_project_id/your_project_id-kubeconfig
```

D. Get the Bear Token: To acquire the bearer token to login to GKE:

```
# kubectl apply --kubeconfig ~/project_home/bmctl-workspace/your_project_id/your_project_id-kubeconfig -f - << __EOF__
apiVersion: v1
kind: Secret
metadata:
 name: "user_account-token"
 annotations:
  kubernetes.io/service-account.name: "user_account"
type: kubernetes.io/service-account-token
__EOF__
# until [[ $(kubectl get --kubeconfig ~/project_home/bmctl-workspace/your_project_id/your_project_id-kubeconfig -o=jsonpath="{.data.token}" "secret/user_account-token") ]]; do
 echo "waiting for token..." >&2;
 sleep 1;
done
# kubectl get secret user_account-token --kubeconfig ~/project_home/bmctl-workspace/your_project_id/your_project_id-kubeconfig -o jsonpath='{$.data.token}' | base64 –decode
```

E. **Login via Bear Token:** Click **Log in**, then select **Token** to login, as shown in Figure 9. Copy and paste the generated token in the field, and then click **LOGIN**, as shown in Figure 10.

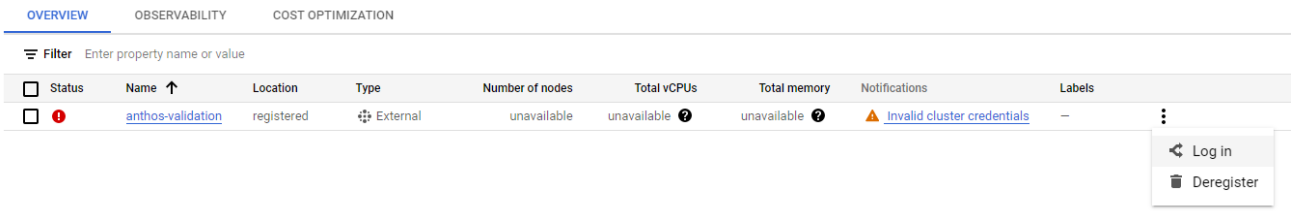


Figure 9 – Log in to GKE Cluster

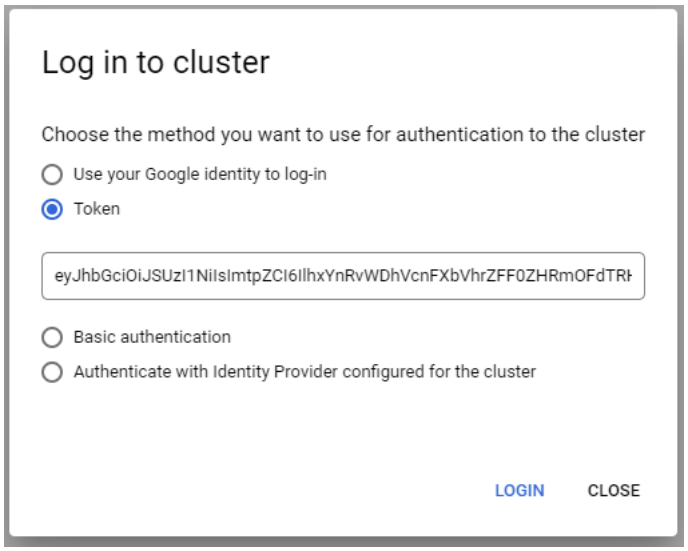


Figure 10 – Paste the token and LOGIN

F. Verify the Cluster Details: You can verify the cluster details as shown in Figure 11 after a successful installation.

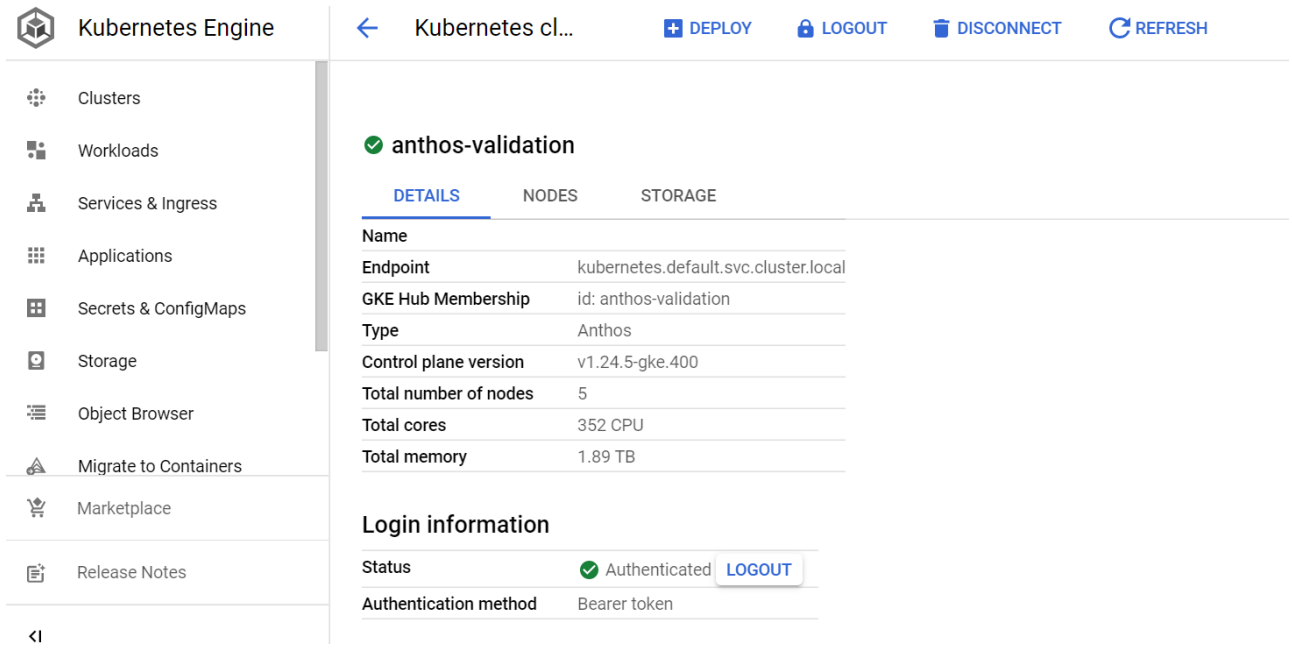


Figure 11 – Viewing cluster details to verify successful installation

Workload Examples

You can quickly deploy various workloads (AI inference, visual computing, 5G/Edge, or any other cloud-native application) in your on-prem environment using Supermicro SuperBlade servers with GDC Virtual via the GKE console. AMD GPU accelerated SuperBlade can be used to perform performance intensive tasks such as AI inference and large scale data processing. GKE offers GPU-specific features, such as time-sharing and multi-instance GPUs, that can improve the efficiency with which your workloads use the GPU resources on your nodes. Local users can visit services via exposed cluster ports. Google Container Registry also provides different images to help rapid deployment and securely manage private images. GDC Virtual on bare metal clusters sends regular health messages to update the health status on Google Kubernetes Engine. Google Cloud Console can display detailed information and perform further operations, such as deploying and deleting workloads.

NGINX is a well-known, free, and open-source web server that can also function as a reverse proxy, load balancer, and HTTP cache. Figure 12 shows a sample bare metal deployment of NGINX on SuperBlade servers powered by GDC Virtual.

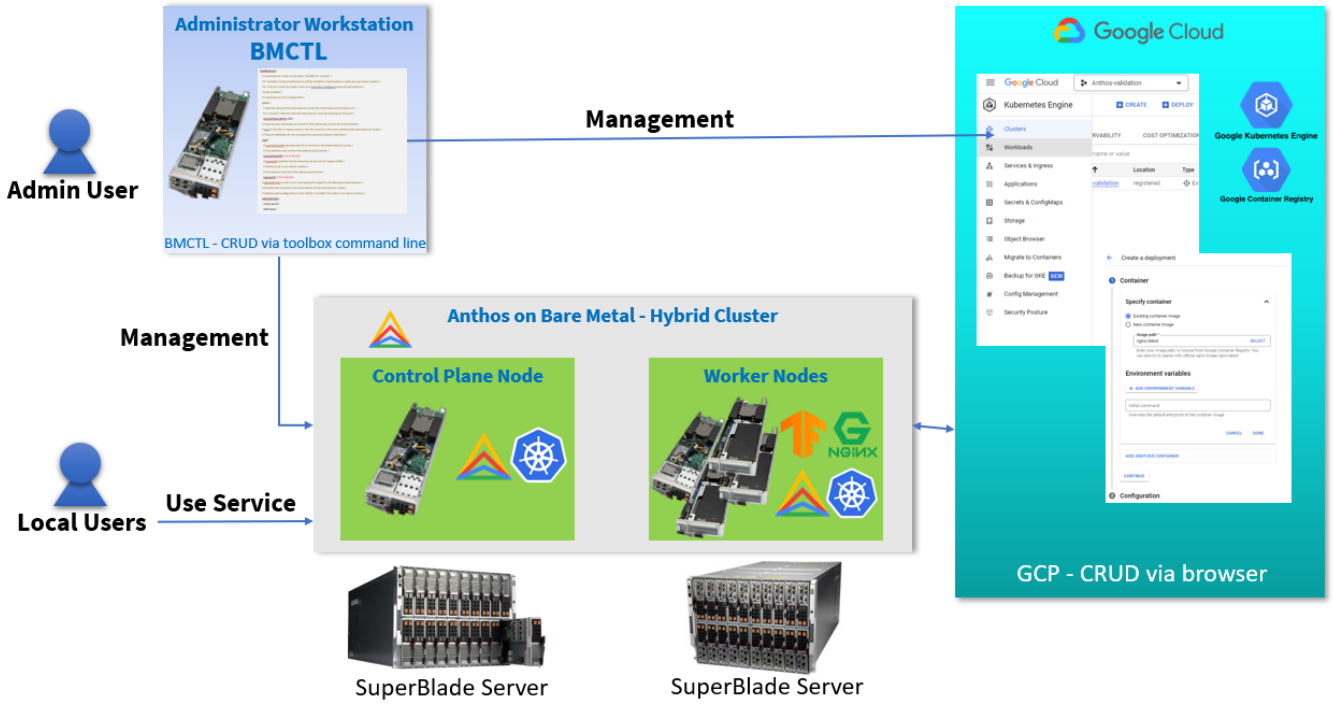


Figure 12 – Sample workload deployment of GDC Virtual on SuperBlade bare metal

To deploy NGINX:

A. **Workload Overview:** In GKE, click **Workloads** to see a workload overview, as shown in Figure 13.

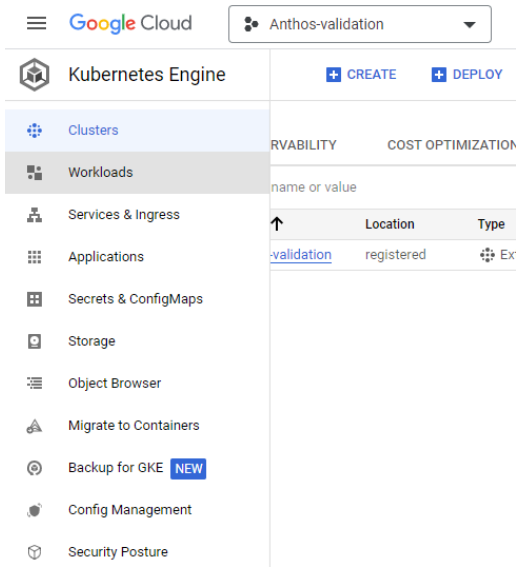


Figure 13 – Click Workloads to see workload details

B. Deploy Workloads: Click **DEPLOY** to deploy workloads, as shown in Figure 14.

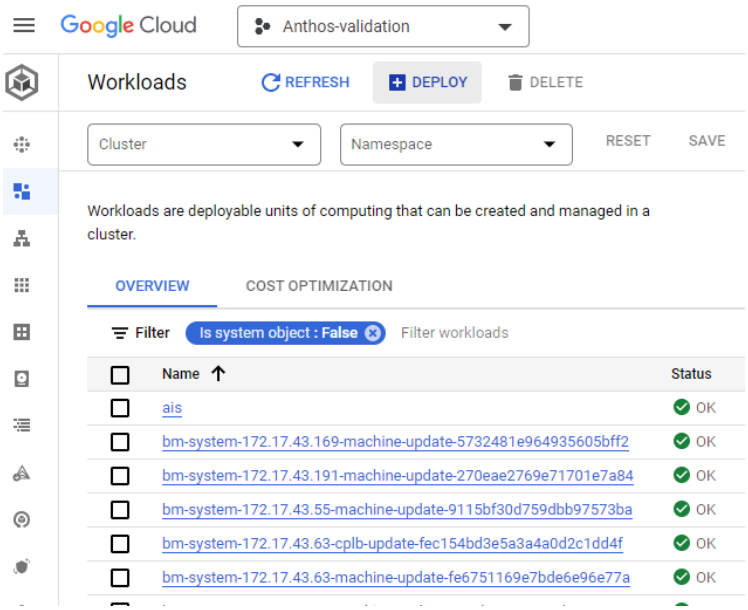


Figure 14 – Click DEPLOY to deploy workloads.

C. **Select Container Image for Deployment:** Select an image for deployment from the Google Container Registry, and then click **CONTINUE**, as shown in Figure 15.

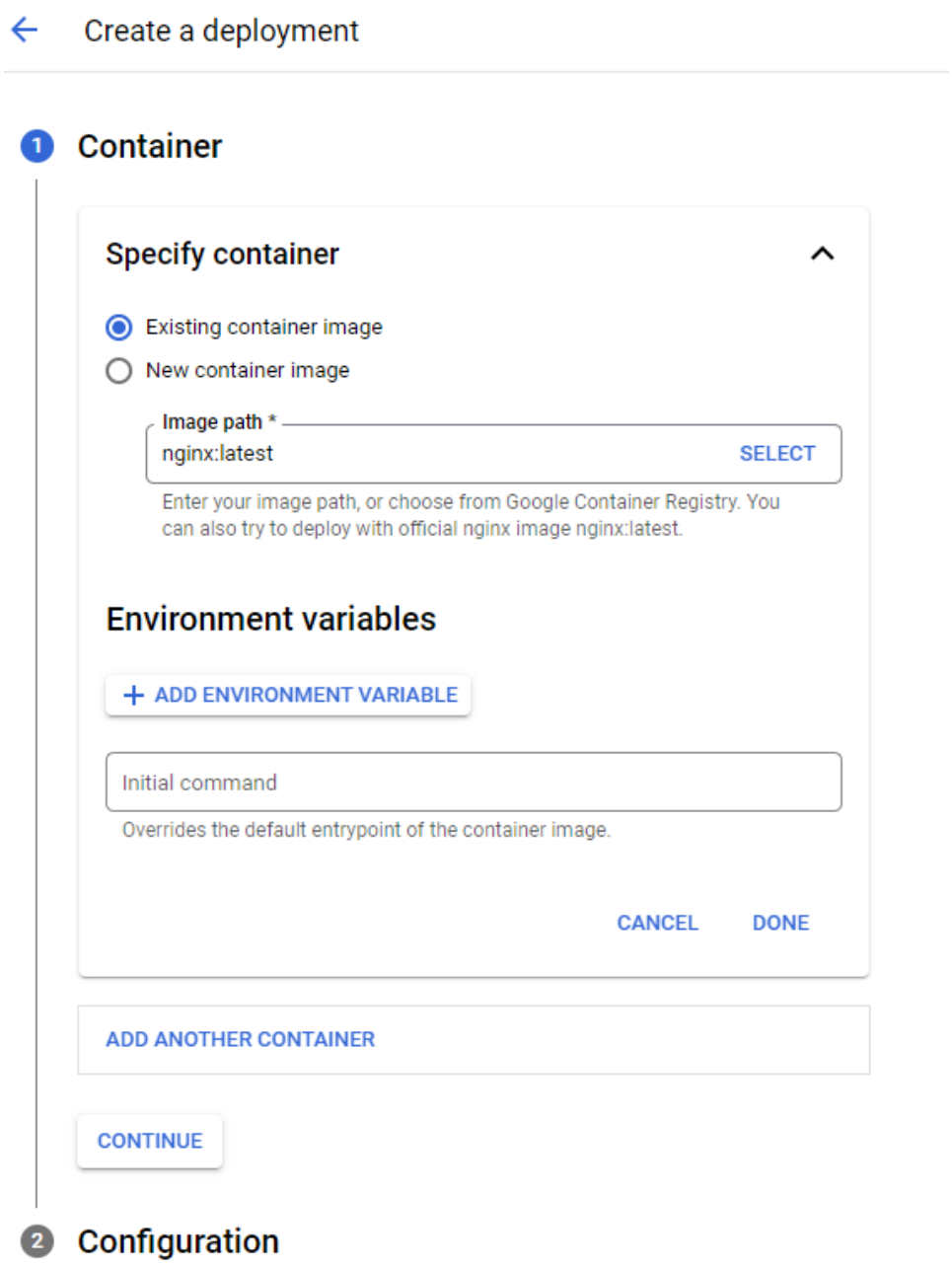


Figure 15 – Selecting a container image for deployment

D. Container Configuration: Set the application name, namespace, and labels, as shown in Figure 16.

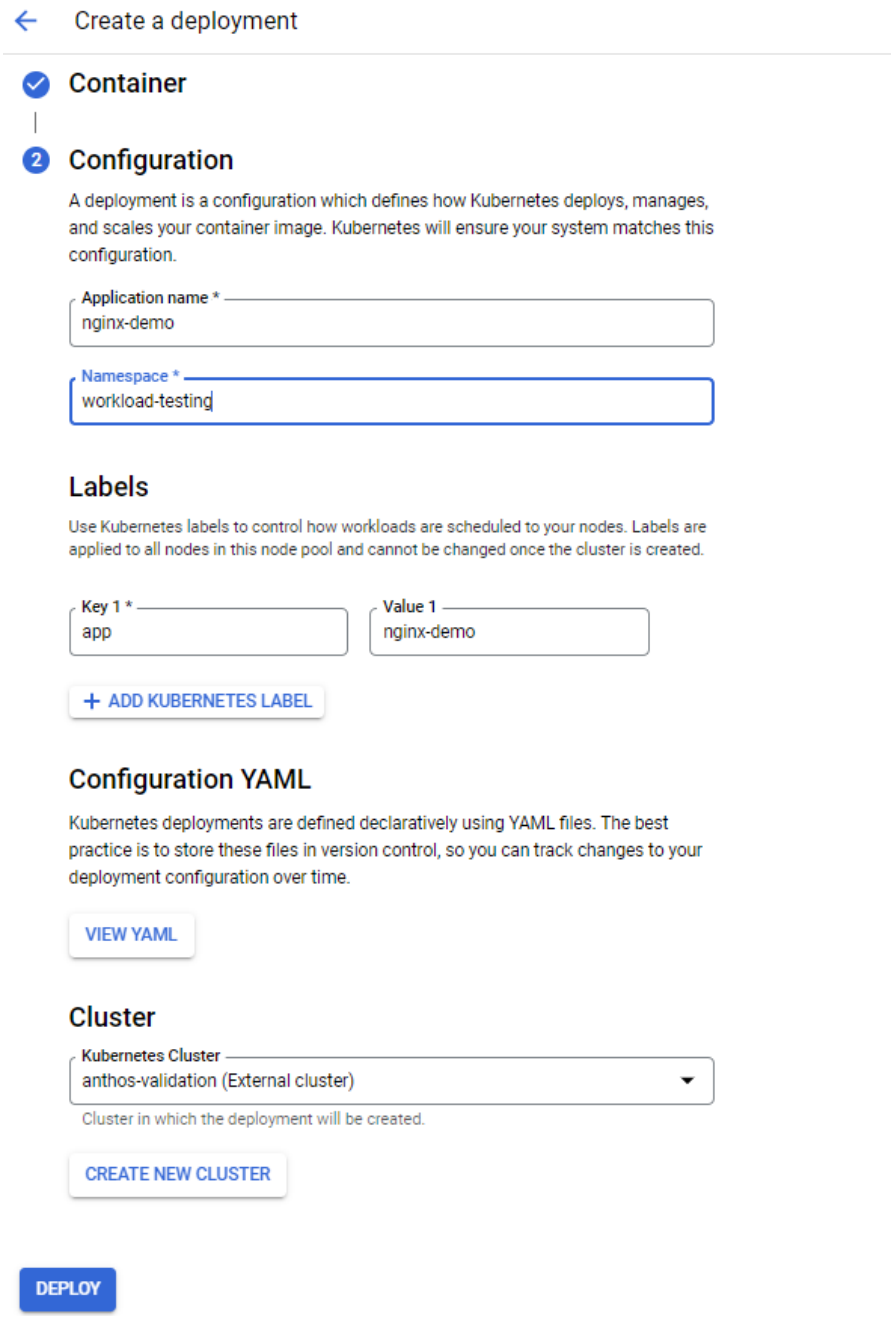


Figure 16 – Container configuration

E. Deployment Result: Verify successful deployment, as shown in Figure 17.

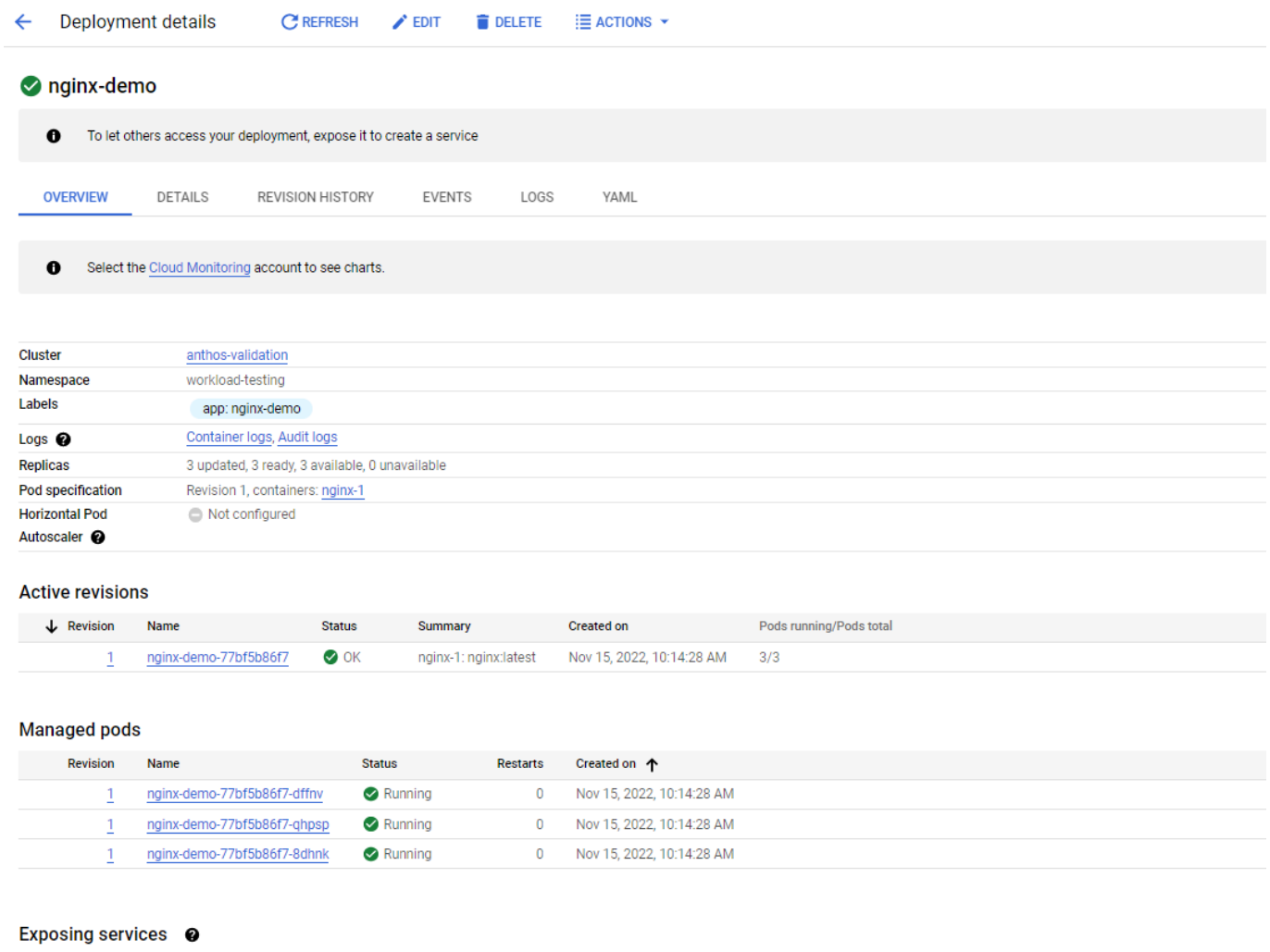


Figure 17 – Deployment Result

F. Show the Status of Container: Execute the command kubectl get pods on the Administrator Workstation to view pod status, as shown in Figure 18.

```
smci@run:~/anthos-validation$ kubectl get pods -n workload-testing -o wide  --kubeconfig ~/anthos-validation/bmctl-workspace/anthos-validation/anthos-validation-kubeconfig
NAME                          READY   STATUS    RESTARTS   AGE     IP              NODE           NOMINATED NODE   READINESS GATES
nginx-demo-77bf5b86f7-8dhnk   1/1     Running   0          6h27m   192.168.8.51    anthhos-rhel8  <none>           <none>
nginx-demo-77bf5b86f7-dffnv   1/1     Running   0          6h27m   192.168.7.238   anthos-u2004   <none>           <none>
nginx-demo-77bf5b86f7-qhpsp   1/1     Running   0          6h27m   192.168.2.204   anthos-lb1     <none>           <none>
```

Figure 18 – Viewing pod status on the Admin Workstation

Solution Benefits:

Deploying GDC Virtual on Supermicro SuperBlade offers the following benefits:

- Hardware agnostic: Customers can leverage existing on-prem SuperBlade servers to drive data center efficiency.
- No hypervisor layer overhead: Deploying GDC Virtual on SuperBlade has reduced complexity.
- Rapid deployment: Both developers and dev-ops teams benefit from increased productivity because GDC Virtual enables rapid cloud native application development and delivery.
- Superior performance: Supermico SuperBlade, powered by AMD EPYC CPUs and/or AMD Instinct accelerators, plus the advanced networking described above, delivers the performance needed to run today's and tomorrow's most demanding workloads on GDC Virtual. A large L3 Cache also reduces memory bandwidth pressure and reduces application latency barriers.
- Easy manageability: SuperBlade CMM manageability coupled with GDC Virtual management enables increased operational efficiency (offers a dashboard with a range of health checks, logging, and monitoring).
- Optimal TCO: Supermicro SuperBlade offers a building block solution with a Resource Saving Architecture that optimizes precious data center resources, such as power, space, and cooling.
- Scalability: SuperBlade enables GDC Virtual to deploy at scale across development, test, and production clusters.

Key Takeaways and Business Values

Organizations that need to keep some applications on-premises can turn to Supermicro SuperBlade servers running GDC Virtual on bare metal to transform their on-premises data center into a fully managed, fully integrated cloud region. GDC Virtual is a cloud-native application modernization platform that helps organizations move applications from legacy environments to container-based microservice architectures to reap the full benefits of cloud computing for both cloud-native and legacy applications. GDC Virtual offers a ready-to-go, on-premises platform that helps optimize setup and operating costs while extending the power of Google Kubernetes Engine to Supermicro SuperBlade servers, virtualized on-prem environments, and other public clouds. The Supermicro SuperBlade GDC Virtual solution is ideal for cloud-native workload management. Supermicro SuperBlade servers can be used as control panel nodes and worker nodes to create a GDC Virtual hybrid cluster. GDC Virtual on SuperBlade delivers consistent management, robust security features, out-of-the-box observability, and more. Supermicro SuperBlade, powered by 3rd Gen AMD EPYC processors, runs GDC Virtual with the operational efficiency and consistency needed to meet various SLAs and IT initiatives and empower increased productivity, optimized TCO, and scalability in your data center.

BENEFITS OF DEPLOYING GDC VIRTUAL ON SUPERMICRO SUPERBLADE

Improved productivity for development and security

- Faster application development, testing, and deployment. Developers can write once and deploy anywhere.
- Consistent, unified security policy creation and enforcement.

Streamlined Operational Efficiency

- Reduced complexity with simplified management.
- Faster migrations – IT can quickly containerize, lift and shift applications.
- Reduced effort for releases and patching.

Greener environmental impact and lower TCO

- SuperBlade offers a high-density platform with 96% efficient power supplies there by reducing the carbon footprint.
- Cost optimized building block solution for GDC Virtual with lowest TCO.

As a Google Anthos Ready Platform Partner, Supermicro is a GDC Virtual Ready Platform Provider. We offer a robust application modernization strategy with GDC Virtual on SuperBlade. Deploying GDC Virtual on a tested and validated SuperBlade yields faster time-to-market for new features and increased customer revenue by increasing release frequencies compared to legacy approaches. GDC Virtual on bare metal SuperBlade servers helps future-proof applications by placing them adjacent to Google's cloud services, including advanced machine learning and artificial intelligence options. SuperBlade offers high density, high performance, optimum power efficiency, and ideal Total Cost of Ownership (TCO) for fast, power efficient GDC Virtual deployments on bare metal.

SUPERMICRO

As a global leader in high performance, high efficiency server technology and innovation, we develop and provide end-to-end green computing solutions to the data center, cloud computing, enterprise IT, big data, HPC, and embedded markets. Our Building Block Solutions® approach allows us to provide a broad range of SKUs, and enables us to build and deliver application-optimized solutions based upon your requirements.