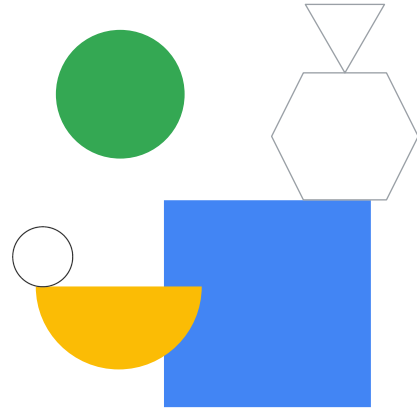


# Adding and Configuring User Clusters



Welcome to Adding and Configuring User Clusters.

# Learning objectives



## Creation

Build and create Anthos user clusters on bare metal. User clusters exist to launch, run, and monitor user workloads.



## Authentication

Learn how to authenticate to Anthos clusters on bare metal by using Google Cloud or third-party providers such as Okta or Active Directory, and set up authorization in your clusters.



## Deployment

Deploy applications on your user clusters and expose them outside of the cluster to your private network or the internet using Kubernetes constructs and the bundled load balancer.




## Storage

Study the different ways of configuring storage and launching stateful workloads in Anthos clusters on bare metal.


Google Cloud

In this module, you learn how to:

- Build bare metal user clusters.
- Authenticate to those clusters using Google Cloud and other identity providers.
- Deploy and expose applications on your user clusters.
- Provision storage for use with stateful workloads on your cluster.




# Today's agenda


- 01 Building the user cluster
  - 02 Enabling authentication
  - 03 Deploying applications
  - 04 Configuring storage
- 

Google Cloud

Here is our agenda for the module.



# Today's agenda



01 [Building the user cluster](#)

---

02 [Enabling authentication](#)

---

03 [Deploying applications](#)

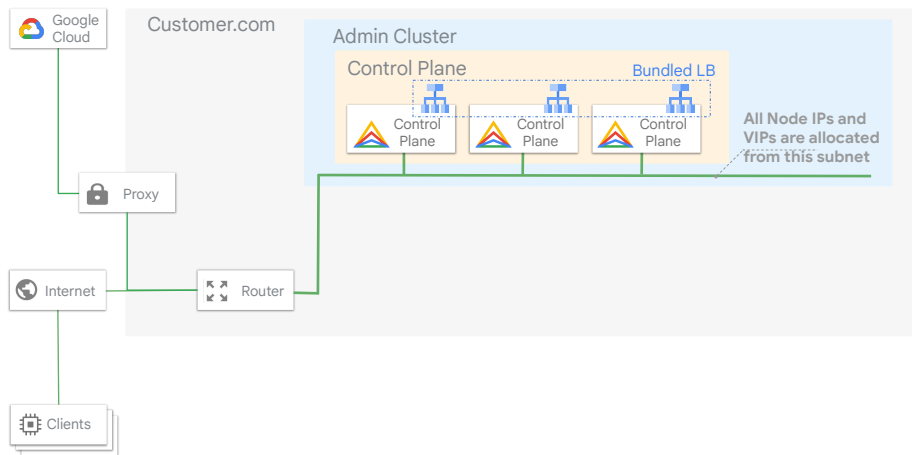
---

04 [Configuring storage](#)

---

We start with the user cluster creation process.

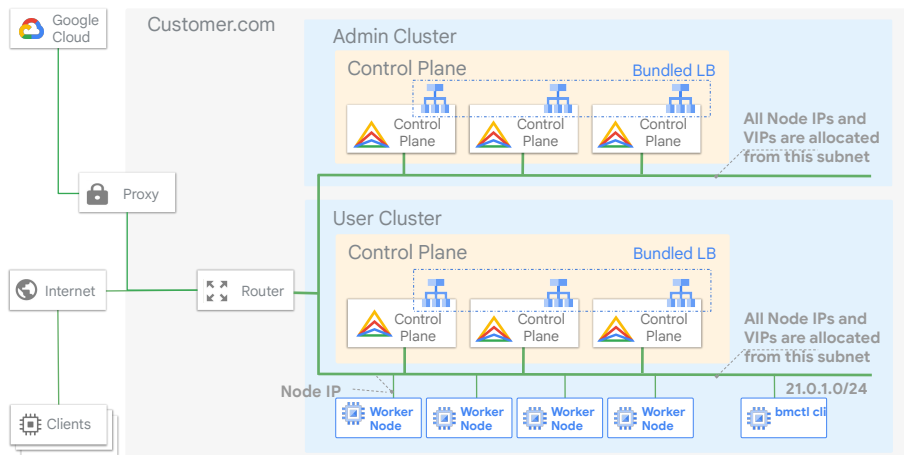
## So far, we've talked about creating admin clusters...



Google Cloud

In the last module, you learned to build an admin cluster.

## Let's talk about creating user clusters to launch workloads



Google Cloud

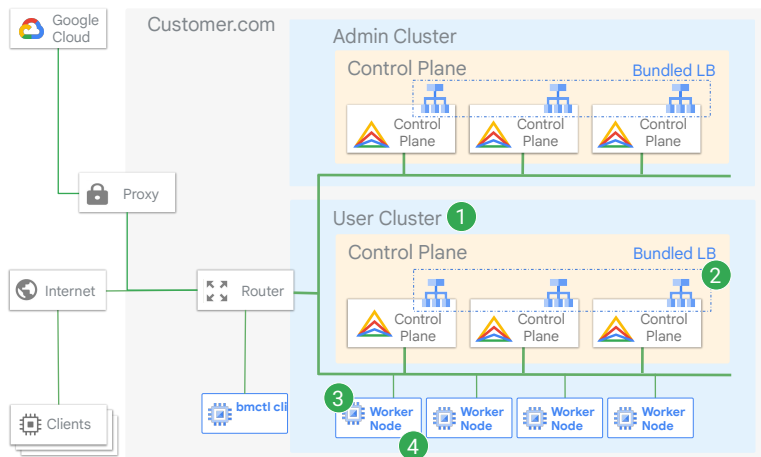
We're going to add a user cluster to that architecture.

What are the differences between a  
**user cluster** and an **admin cluster**?

Google Cloud

We should first make sure we are clear on the differences between user and admin clusters.

## There are four main differences between user and admin clusters



- 1 User clusters run user workloads, and admin clusters manage user clusters.
- 2 User clusters have an Ingress VIP shared by all services for ingress traffic and a range of VIPs for the data plane load balancer.
- 3 User clusters have one or multiple node pools where they can run their user workloads.
- 4 User clusters can enable application logging to observe their workloads.

Google Cloud

There are four key differences between user and admin clusters.

User clusters run application workloads deployed by "users", where users are typically developers or SREs. Admin clusters run the Anthos administrative software, but don't run production applications.

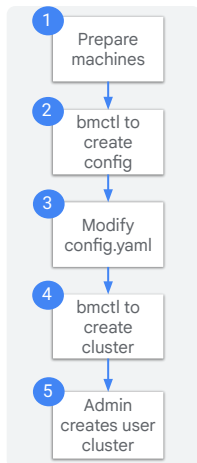
User clusters have an Ingress VIP shared by all services for ingress traffic, and a range of virtual IPs, or VIPs, that can be assigned to LoadBalancer services.

User clusters have one or more worker node pools, where admin clusters typically don't have non-control plane nodes.

User clusters can collect application logs and forward them to Google Cloud. Google recommends not collecting application logs from admin clusters.



## Installation flow with bmctl: commands



1. Distribute the SSH keys so that both the admin workstation and the admin cluster can access the user cluster nodes.
2. Use bmctl to create a user cluster configuration:

```
bmctl create config -c $C1 --project $P
```

3. Modify the auto-generated configuration file to reflect your configuration preferences.
  4. Use bmctl to trigger the user cluster creation:
- ```
bmctl create cluster -c $C1 --kubeconfig $ADMIN_KCF
```
5. The admin cluster runs pods to perform preflight checks and to bootstrap the user cluster.

Google Cloud

The process for creating a user cluster is quite similar to the process for creating an admin cluster. First, you need to establish network connectivity to the cluster servers, and ensure the admin software can SSH into those servers. Then, you:

- Use the bmctl utility to create a cluster configuration file.
- Modify the config file as appropriate for your installation.
- And use bmctl to start creation of the cluster based on the config file.
- The cluster running the admin components (in this case, the admin cluster instead of the kind cluster) does the work to create the user cluster on the specified servers.

## (1) Setup work to configure your cluster: Operations

After you create the cluster configuration file with `bmctl`, modify the file to your required settings:

1. Create an `ssh-key`, distribute it across the cluster nodes, and reference it in the config file.
2. Enable or disable `audit logging` and `application logging` as appropriate to collect more data from your user cluster and your workloads.

```
sshPrivateKeyPath: /root/.ssh/id_rsa
```

```
clusterOperations:  
  disableCloudAuditLogging: false  
  enableApplication: true
```

Google Cloud

Let's look at the modifications you will need to make to the generated config file prior to cluster creation. You'll need to enter the path to the SSH key that will be used to connect to the user cluster servers. You will also want to configure whether Cloud Audit Logging is enabled, and whether you wish to include application logs.

## (2) Setup work to configure your cluster: Type and size

1. Specify the cluster **type**. Options include:
  - admin
  - user
  - hybrid
  - standalone
2. Specify the addresses that will be used by your user cluster's control plane **nodes**.
  - Add a single machine for dev purposes.
  - Add at least three machines for high availability in production.
3. Add one or multiple **NodePools**.  
Useful for:
  - Having different configuration of CPU and memory.
  - Deploying different types of workloads.

```
spec:  
  type: user
```

```
controlPlane:  
  nodePoolSpec:  
    nodes:  
      - 10.200.0.3
```

```
NodePool:  
  nodePoolSpec:  
    nodes:  
      - 10.200.0.5
```

Google Cloud

You will need to specify the cluster type as user, specify the IP addresses of the control plane nodes, and specify the IP addresses for machines in one or more node pools. For production systems, Google recommends at least three control plane nodes, which offers high availability. Clusters might have multiple node pools, with each offering a different hardware configuration suited to different types of workloads.


### (3) Setup work to configure your cluster: Load balancer

1. Specify type of **load balancer**. Options include:
  - bundled
  - manual
2. Set up **control plane VIP**.  
This IP is used by the Kubernetes API server.
3. Set up **ingress VIP**.  
VIP shared by all services for ingress traffic.
4. Set up one or multiple **address pools**:
  - IP addresses that can be assigned to Kubernetes LoadBalancer Services running on the cluster.
  - Address pools examples include:
    - Public IPs accessible from the internet
    - Private IPs in the customer's network accessible outside of the cluster
5. (Optional) Set up the load balancer on a separate **NodePool**.


```
loadBalancer:
  mode: bundled
...
vips:
  controlPlaneVIP: 10.200.0.99
  ingressVIP: 10.200.0.100
addressPools:
- name: pool1
  addresses:
    - 10.200.0.100-10.200.0.200
...
controlPlane
nodePoolSpec:
  nodes:
    - 10.200.0.3
```

Google Cloud

For user clusters, like admin clusters, you will need to specify the details related to load balancing in your config file. You set the mode, and for bundled load balancing, you designate the VIPs used for Kubernetes API server, the cluster ingress, and LoadBalancer services on the cluster. You can designate more than one address pool for those services, allowing you to choose from private and public IPs. And, as with admin clusters, the load balancing software can run on your control plane nodes or on a separate load balancer node pool, which you would need to specify in your config file.



# Today's agenda



01 Building the user cluster

---

02 [Enabling authentication](#)

---

03 Deploying applications

---

04 Configuring storage

---

Google Cloud

To manage user clusters, an operator must log in to the cluster. Let's explore how Anthos enables authenticating in different environments.

## There are two ways to work with your bare metal cluster

01

### Via Google Cloud

- Works with registered clusters.
- Works for administration via the Console or with `kubectl` via the Connect gateway.
- Authentication is done with one of the following:
  - Google Identity
  - Token
  - Basic Authentication
  - External identity provider

02

### Via a direct connection

- Use standard management tools that connect directly to the Kubernetes API server.
- Requires network connectivity from client to the API Server.
- Authentication is handled using:
  - OpenID Connect
  - X509 client certificates
  - Static passwords

Google Cloud

There are at least two ways you can connect to and use an Anthos cluster.

You can work in the Google Cloud Console, which then forwards commands to registered clusters via the Anthos Connect agent. You must configure the Console to authenticate to the cluster on your behalf using a Google Account, a Kubernetes Service Account token, Basic Authentication, or an external identity provider.

You can also connect directly to the Kubernetes API server. You need to have network connectivity to the server, and the server will need to be able to authenticate you using a token, certificate, or password.

# How do you access your bare metal clusters via Google Cloud?

Google Cloud

So, how does managing a bare metal cluster via Google Cloud tools work?

To access registered clusters via the console, you must log in to the cluster

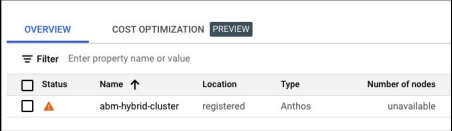
Google Cloud

Obviously, users must be logged in to access the Google Console. But to access the bare metal Anthos cluster, they need to explicitly log in to the cluster as well.




## To access registered clusters via the console, you must log in to the cluster

- You can see registered bare metal clusters in the GKE Clusters page of the Console.



The screenshot shows the Google Cloud console interface for GKE Clusters. At the top, there are tabs for 'OVERVIEW', 'COST OPTIMIZATION', and 'PREVIEW'. Below the tabs is a search bar labeled 'Filter' with the placeholder text 'Enter property name or value'. The main content is a table with the following columns: 'Status', 'Name', 'Location', 'Type', and 'Number of nodes'. There is one row in the table representing a cluster named 'abm-hybrid-cluster' located in 'registered' with 'Anthos' type and 'unavailable' number of nodes. The 'Status' column for this cluster shows a yellow triangle icon.

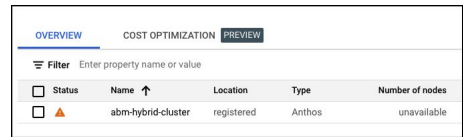
| OVERVIEW                                                                                                   |                    |            |        |                 |
|------------------------------------------------------------------------------------------------------------|--------------------|------------|--------|-----------------|
| COST OPTIMIZATION                                                                                          |                    |            |        |                 |
| PREVIEW                                                                                                    |                    |            |        |                 |
| Filter Enter property name or value                                                                        |                    |            |        |                 |
| <input type="checkbox"/> Status                                                                            | Name ↑             | Location   | Type   | Number of nodes |
| <input type="checkbox"/>  | abm-hybrid-cluster | registered | Anthos | unavailable     |

Google Cloud

When an operator goes to the GKE Clusters page in the Console, they will see the registered cluster displayed in the cluster listing. However, the cluster will show that the operator hasn't yet logged in to the cluster.

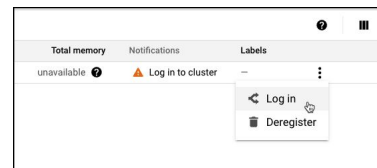
## To access registered clusters via the console, you must log in to the cluster

- You can see registered bare metal clusters in the GKE Clusters page of the Console.
- You must configure the console to talk to the bare metal cluster as a specific user.
  - “Log in” option is available in the console.
  - Each user goes through the login process individually.
  - The identity is used only for that particular user.



The screenshot shows the Google Cloud Console interface for GKE clusters. At the top, there are tabs for 'OVERVIEW', 'COST OPTIMIZATION', and 'PREVIEW'. Below the tabs is a filter bar with the text 'Filter Enter property name or value'. The main content is a table with the following columns: Status, Name, Location, Type, and Number of nodes. There is one row in the table with the following data: Status is 'unavailable' (indicated by a yellow triangle icon), Name is 'abm-hybrid-cluster', Location is 'registered', Type is 'Anthos', and Number of nodes is 'unavailable'.

| Status      | Name               | Location   | Type   | Number of nodes |
|-------------|--------------------|------------|--------|-----------------|
| unavailable | abm-hybrid-cluster | registered | Anthos | unavailable     |

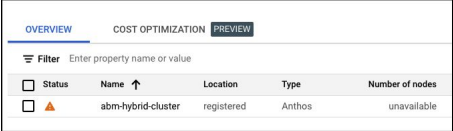


Google Cloud

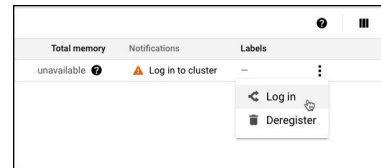
The operator can choose the Log in option from the three-dot menu. This login is user-specific; every user must configure this for themselves.

## To access registered clusters via the console, you must log in to the cluster

- You can see registered bare metal clusters in the GKE Clusters page of the Console.
- You must configure the console to talk to the bare metal cluster as a specific user.
  - “Log in” option is available in the console.
  - Each user goes through the login process individually.
  - The identity is used only for that particular user.
- Actions initiated via the Console trigger API requests to the bare metal cluster.
  - These requests must be submitted in the context of a user or service account.
  - This account must have role bindings on the cluster that allow the requested actions.



| OVERVIEW COST OPTIMIZATION PREVIEW  |                    |            |        |                 |
|-------------------------------------|--------------------|------------|--------|-----------------|
| Filter Enter property name or value |                    |            |        |                 |
| <input type="checkbox"/> Status     | Name ↑             | Location   | Type   | Number of nodes |
| <input type="checkbox"/>            | abm-hybrid-cluster | registered | Anthos | unavailable     |



Google Cloud

The identity used to log in here is the identity which is used to submit requests to the API server on your cluster. This identity must have the necessary role assignments on the cluster itself in order for Console-initiated operations to succeed.

## You can log in to the cluster using various credentials

### Log in to cluster

Choose the method you want to use for authentication to the cluster

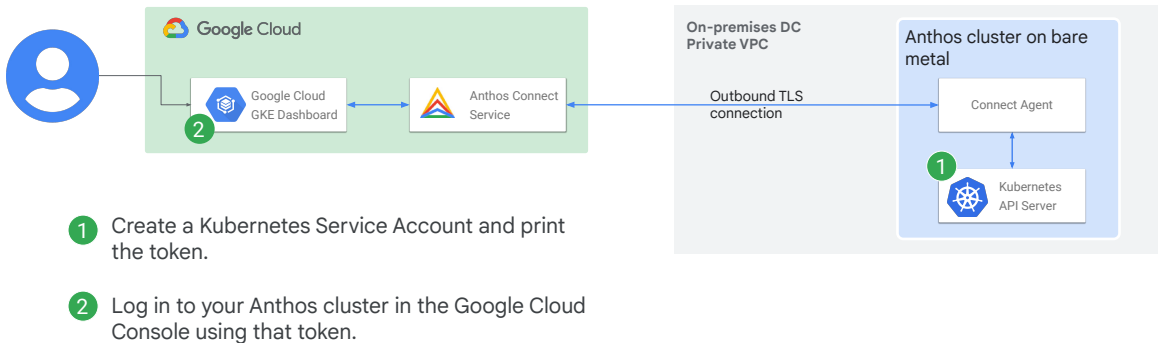
- ☒ Use your Google identity to log-in
- ☐ Token
- ☐ Basic authentication
- ☐ Authenticate with Identity Provider configured for the cluster

[LOGIN](#) [CLOSE](#)

Google Cloud

To log in, the user can use several different types of credentials: Google Accounts, Kubernetes Service Account tokens, basic username/password combinations, or credentials associated with a third-party Identity Provider.

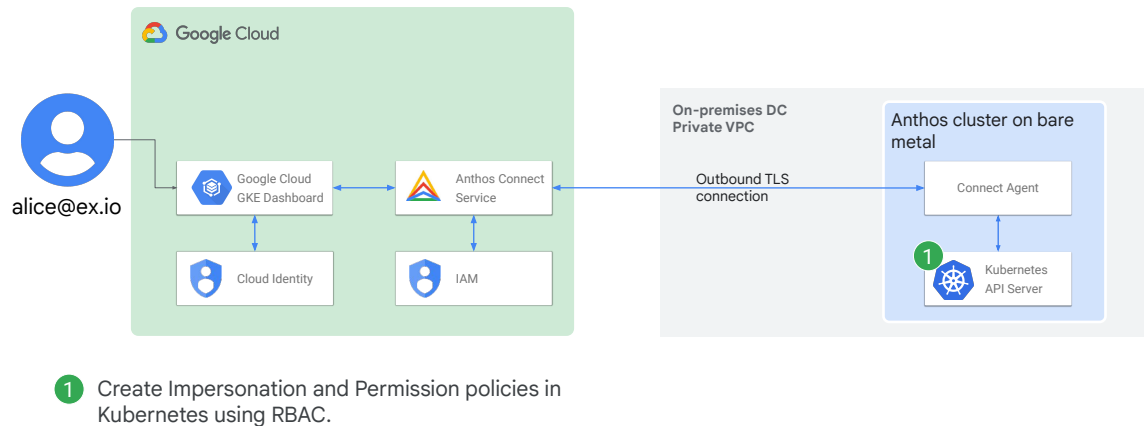
Over the last several years, token-based authentication has been the most common approach



Google Cloud

Over the last few years, the most common way to log in to a cluster was using tokens. A cluster administrator would create a Kubernetes Service Account, or KSA, for each operator, then get the KSA bearer tokens and distribute them. The operator would log in to the cluster by providing a KSA bearer token, and the service would then submit all API requests in the context of that KSA. The cluster administrator would need to make sure that each KSA has all the RBAC assignments necessary.

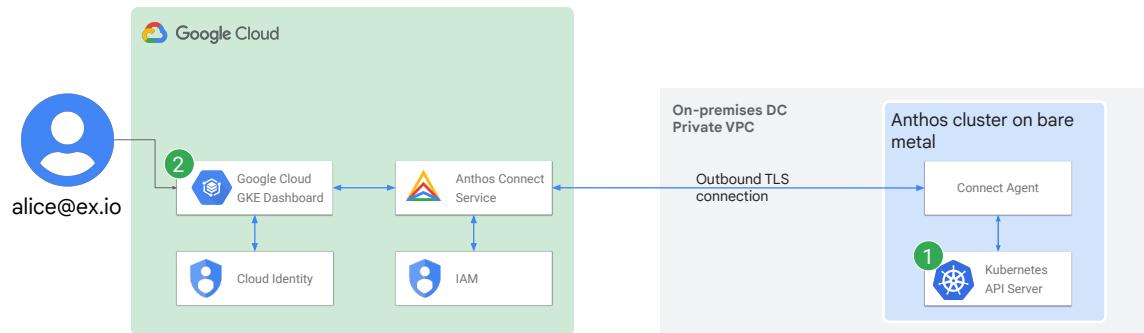
## Google now recommends using your Google identity to authenticate to bare metal clusters



Google Cloud

Today, Google recommends logging in to Anthos clusters using a Google Identity instead of a KSA token. This reduces the complexity of managing identities across clusters. The administrator would start by creating policies on the cluster that allow the connect agent to impersonate the operator, in this case Alice. We're assuming that the cluster administrator has already configured RBAC to allow Alice's user the permissions she needs on the cluster.

## Google now recommends using your Google identity to authenticate to bare metal clusters



- 2 Log in to your Anthos cluster in the Google Cloud Console. After logging in, you can view and edit your Anthos cluster.

Google Cloud

Once this is done, Alice logs in to the cluster selecting the option to use her Google identity. Operations she initiates will flow through the Connect tunnel, and the Connect Agent will submit them impersonating her.

## Operators can also do kubectl-based management via the Anthos Connect gateway

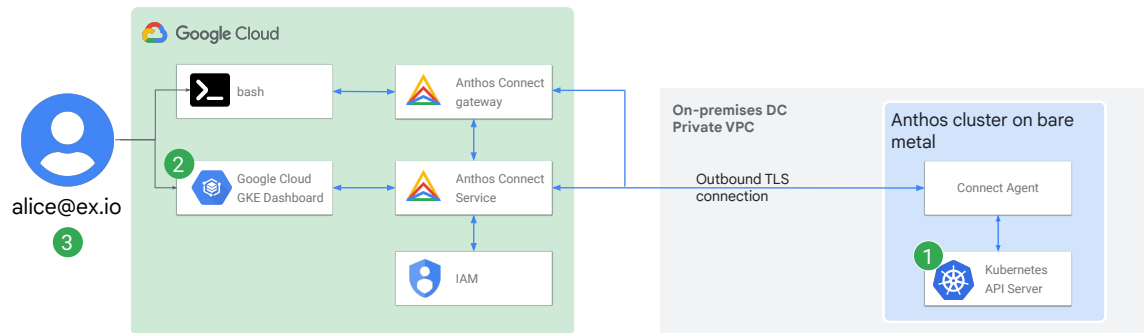
- Create a kubeconfig file that points to the gateway service.
- Kubectl connects to the gateway, which in turn connects to the target API server.
- Provides the following benefits:
  - Easy to discover clusters in your fleet.
  - Allows you to connect to servers running in various environments without direct network connectivity.
  - Allows you to authenticate using your Google credentials.
  - Makes specifying RBAC easier across an entire fleet of clusters.

Google Cloud

Now, using the GKE Console to do cluster administration is nice, but what if you're a hardcore command-line user? Well, Google provides the Anthos Connect gateway which allows users to use kubectl to manage clusters even when they don't have direct network access to the cluster. Operators point kubectl to the gateway, which forwards requests to the target API server. This feature makes it easy for operators to discover all the clusters in a fleet, allows connecting to all using your Google credentials without special cluster configuration, and makes setting up RBAC across clusters much easier.



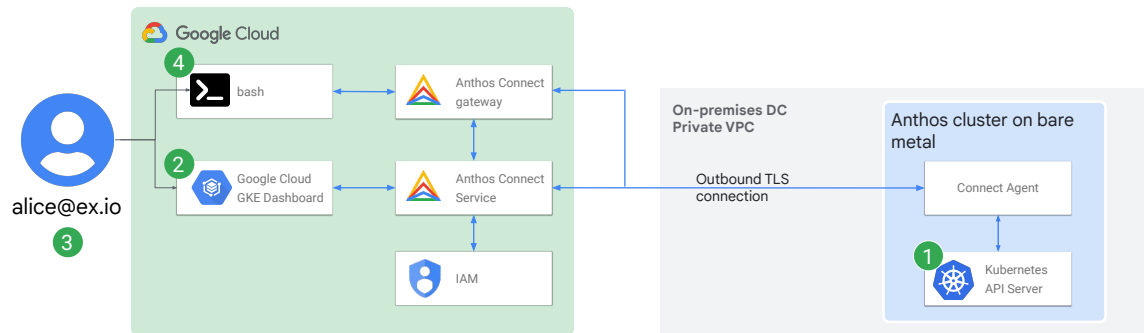
## Using the Connect gateway: Assign roles



- 3 Enable APIs and grant the `gkehub.gatewayAdmin` and `gkehub.viewer` IAM roles to Alice.

To enable use of the gateway, a project administrator assigns operators specific roles in IAM. Users will need the gateway admin and viewer roles.

## Using the Connect gateway: Getting a kubeconfig file



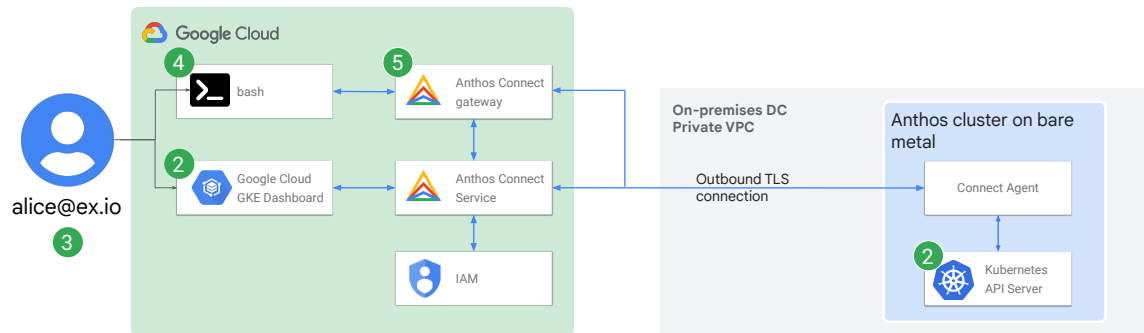
- 4 Download the kubeconfig file for the Connect gateway:

```
gcloud container hub memberships get-credentials user-cluster
```

Google Cloud

The operator, in this case Alice, generates a kubeconfig file using the `gcloud container hub memberships get-credentials` command.

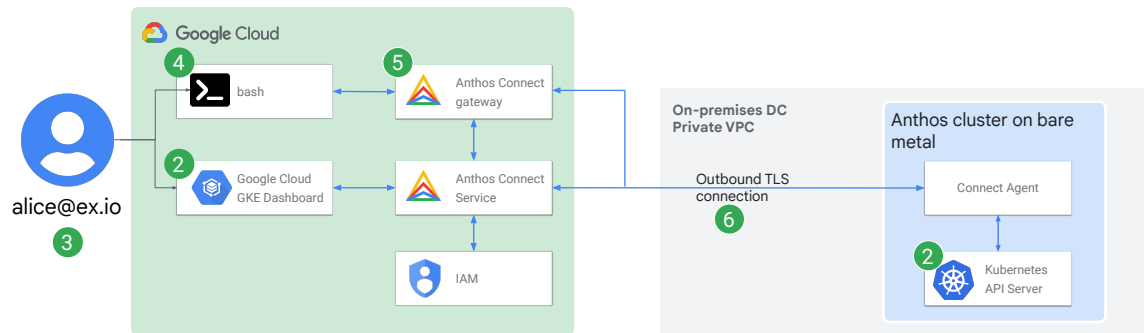
## Using the Connect gateway



- 5 Use the new kubeconfig file with kubectl, connect to the gateway, and initiate an operation.

Alice would then use the new kubeconfig file with kubectl, which will connect to the gateway instead of directly to the target cluster. Once connected, Alice can initiate an operation with kubectl.

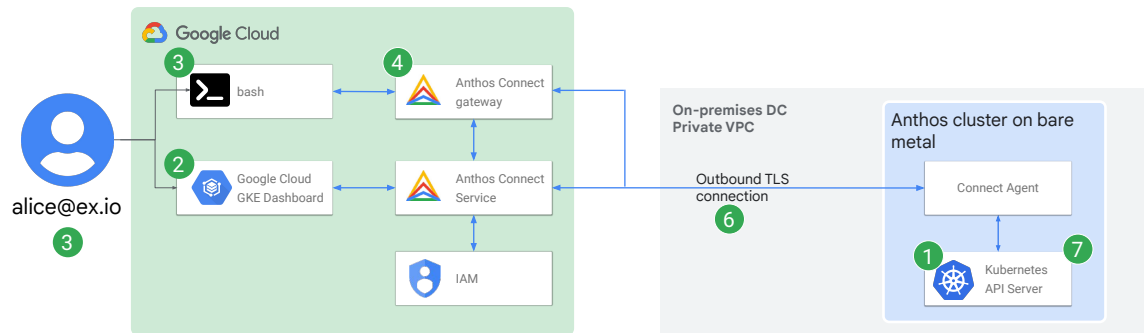
## Using the Connect gateway



- 6 The Connect gateway forwards the request, along with identity information, via the Connect Agent to the API server.

The Connect gateway will forward the requests via the agent on the cluster to the API server.

## Using the Connect gateway



- 7 RBAC on the server allows the Connect Agent to impersonate a user or service account, so the commands are run using the Google identity of the user.

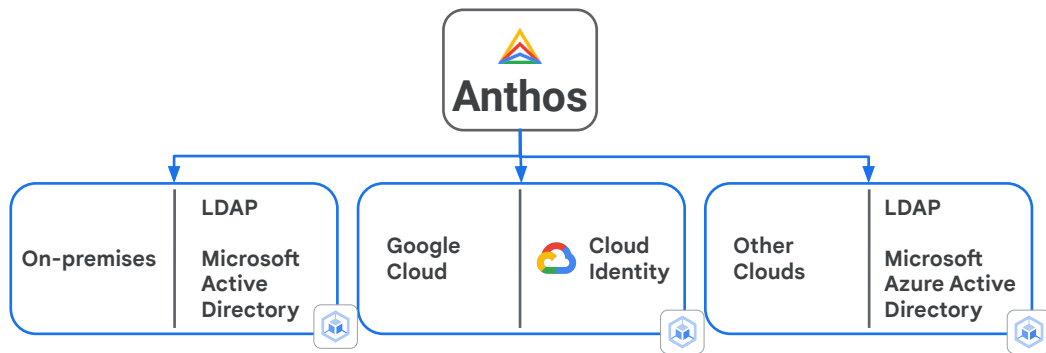
We're assuming here that the administrator had previously configured the impersonation policy as noted before, which allows the Agent to impersonate the user. Assuming other RBAC assignments have granted the user appropriate permissions, the command executes successfully.

For direct connections, Anthos  
clusters on bare metal provides the  
**Anthos Identity Service.**

Google Cloud

For operators who want to connect directly to the cluster, instead of going through the Anthos Connect tunnel, Anthos helps with cluster authentication by providing the Anthos Identity Service.

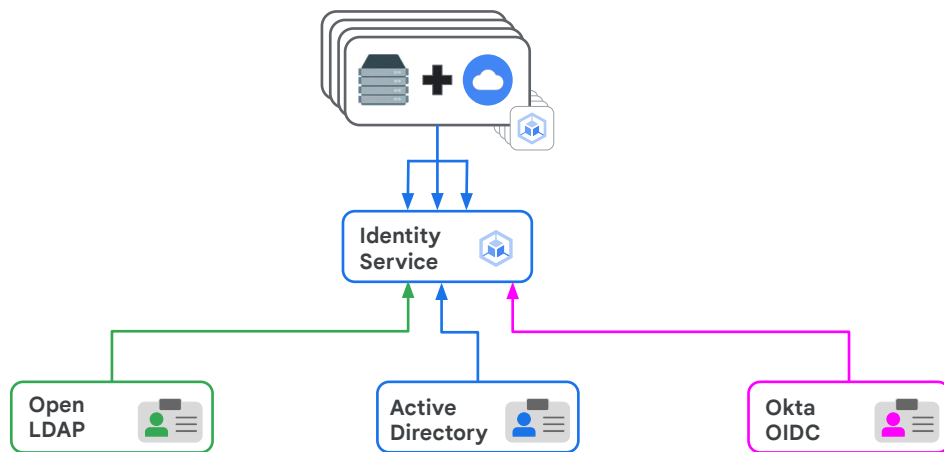
Different environments typically use different authentication solutions



Google Cloud

The challenge when authenticating directly to multiple Anthos clusters in multiple operating environments is that each environment might use different identity management solutions. On-premises clusters might use LDAP servers or on-premises Active Directory for authentication. GKE servers obviously use Cloud Identity and Google Accounts. Clusters running other clouds may use LDAP or Azure AD or Okta. This would make configuring clusters across environments quite complex.

## The Anthos Identity Service enables Kubernetes authentication using multiple backend services

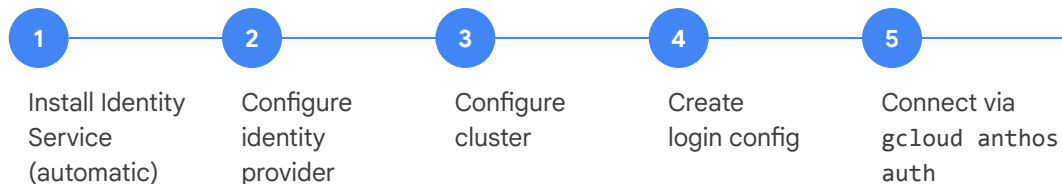


Google Cloud

The Anthos Identity Service streamlines the process. Each cluster will run the Identity service, which can talk to various backend Identity Providers, or IdPs. So, administrators have a single, relatively consistent process for setting up clusters using multiple IdPs.



## Anthos Identity Service setup process



Google Cloud

Setting up and using the Identity service is a relatively straightforward process. The software is automatically installed on Anthos clusters on bare metal. The cluster administrator then configures the IdP to talk to the service, configures the cluster to talk to the IdP, and creates a login config file that will be used by operators. The operators then generate an auth token that is stored in their kubeconfig files by using a `gcloud` command.

## Configuring the provider

- The identity service offers two types of providers:
  - OpenID Connector (OIDC) providers (works for all Anthos cluster types)
  - LDAP providers (only works on VMware and bare metal clusters)
- For both types of providers, follow vendor-specific configuration instructions to set up the Identity Provider (IdP) to work with your clusters.
  - Works with on-prem Active Directory, Azure AD, Okta, OpenLDAP, etc.
- When the IdP is configured, it provides some information you use to configure the clusters.

Google Cloud

There are two types of providers used by the Identity Service. An OIDC provider can talk to any OIDC compliant IdP, such as Azure AD, Okta, Google, etc. LDAP providers talk to, well, LDAP servers. The setup is similar across vendors, but will differ slightly so you will need to look at the vendor-specific documentation. Once the IdP is configured, you will take some information from it to configure your cluster.

## Configuring the clusters

- You can configure individual clusters or an entire fleet of clusters.
  - Fleet setup allows you to perform a single process to enable all the clusters in the fleet to use the same authentication scheme.
- To configure a single cluster, modify the custom ClientConfigs resource:

```
kubectl --kubeconfig=KUBECONFIG_PATH edit ClientConfigs default -n kube-public
```

- Fleet configuration is done via the Features section of the Anthos dashboard.
- Configurations details will differ by IdP vendor, but typically will require:
  - clientId
  - clientSecret
  - URIs
  - Scopes and claims

Google Cloud

To configure clusters, you can set them up individually or as a fleet. At the time this video was recorded, the fleet configuration was in preview. It makes configuring a whole set of clusters much easier if they all need to use the same authentication setup. To configure an individual cluster, you manually edit the ClientConfigs object in the cluster control plane. To configure a fleet, you use the GUI found in the Anthos dashboard. While configuration details vary by vendor, you generally will specify a clientId, clientSecret, some URIs, and some scopes and claims.

## Creating the login config

- To connect directly to clusters and authenticate using the Anthos Identity Service, operators need a special configuration file.
- Administrators can generate that configuration file and share it with operators:

```
gcloud anthos create-login-config --kubeconfig=KUBECONFIG
```

Google Cloud

Once the provider and cluster have been configured, the administrator generates a login config file that is used by the gcloud command to generate a token and update the kubeconfig file. To generate the login config, you use gcloud anthos create-log-config. This requires that the anthos-auth component be added to your Google SDK installation. You then need to find a secure distribution method to get this file to the operators who will use it.

# Connecting to the cluster


1. Operators run the following command to initiate the authentication flow:

```
gcloud anthos auth login \  
--cluster [CLUSTER_NAME] \  
--user [USER_NAME] \  
--login-config [AUTH_CONFIG_FILE_PATH] \  
--kubeconfig [CLUSTER_KUBECONFIG]
```


2. You are prompted in a web browser window to log in using your credentials from the configured identity provider.
3. A token is generated and added to your kubeconfig file, and you can now connect to the cluster. (The cluster will trust the token and can request additional IdP information as needed.)
4. You must make role assignments on the cluster to allow the authenticated user to perform operations.

Google Cloud

Finally, for an operator to connect and authenticate to the cluster, they start by running the `gcloud anthos auth login` command. They specify the cluster they want to connect to, the name of the user they wish to authenticate as, the location of the login config, and the kubeconfig file they want to have updated with the generated token. A web page will be opened in your local browser, presenting the login page of the appropriate IdP. Once you successfully enter your credentials and are authenticated by the IdP, a token is generated and stored in your kubeconfig file. Now, you can connect to the cluster, and operations are executed in the context of the user account you logged in with. Keep in mind that, as always, RBAC on the cluster must allow that user to perform the desired operations.



# Today's agenda



01 Building the user cluster

---

02 Enabling authentication

---

03 [Deploying applications](#)

---

04 Configuring storage

---

Google Cloud

All right, so you've built your user cluster and you have an authenticated connect to your cluster. Now, you want to deploy workloads to that cluster.

# Deploying applications on Anthos clusters on bare metal



- You can run applications using regular Kubernetes constructs such as Deployments.
- There are two options to expose the applications outside of the cluster:
  - Using Kubernetes-native constructs with:
    - Kubernetes Services type LoadBalancer
    - Kubernetes Ingress resource
  - Using Anthos Service Mesh

Google Cloud

Deploying applications on your bare metal clusters is almost identical to deploying applications on any Kubernetes cluster. You can create Deployments and StatefulSets, which create running workloads. And you can expose your workloads using native Kubernetes constructs like services and ingresses, or you can leverage Anthos Service Mesh.

# Deploying applications on Anthos clusters on bare metal



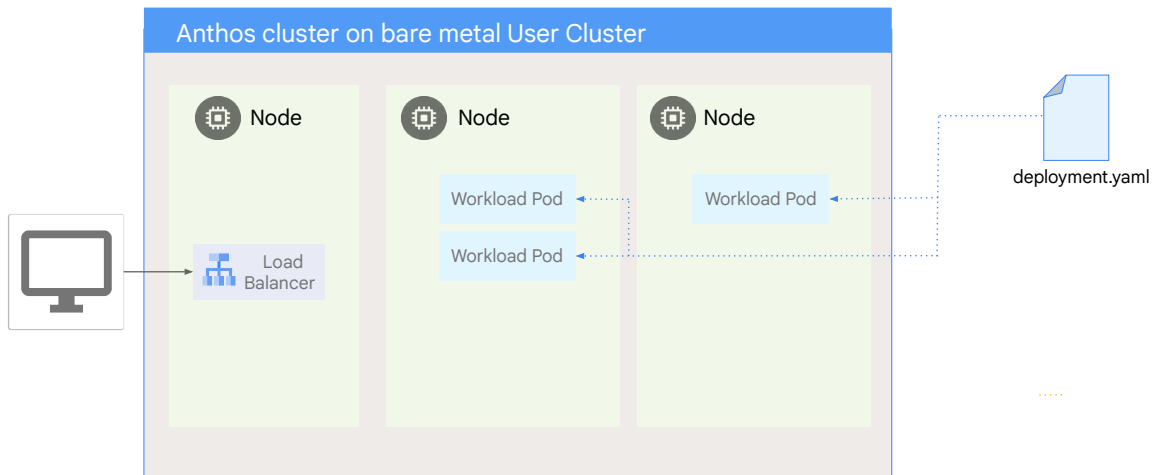
- You can run applications using regular Kubernetes constructs such as Deployments.
- There are two options to expose the applications outside of the cluster:
  - **Using Kubernetes-native constructs** with:
    - Kubernetes Services type LoadBalancer
    - Kubernetes Ingress resource
  - Using Anthos Service Mesh

Google Cloud

Let's look into how exposing workloads to clients outside the cluster works when using Kubernetes services and ingresses.



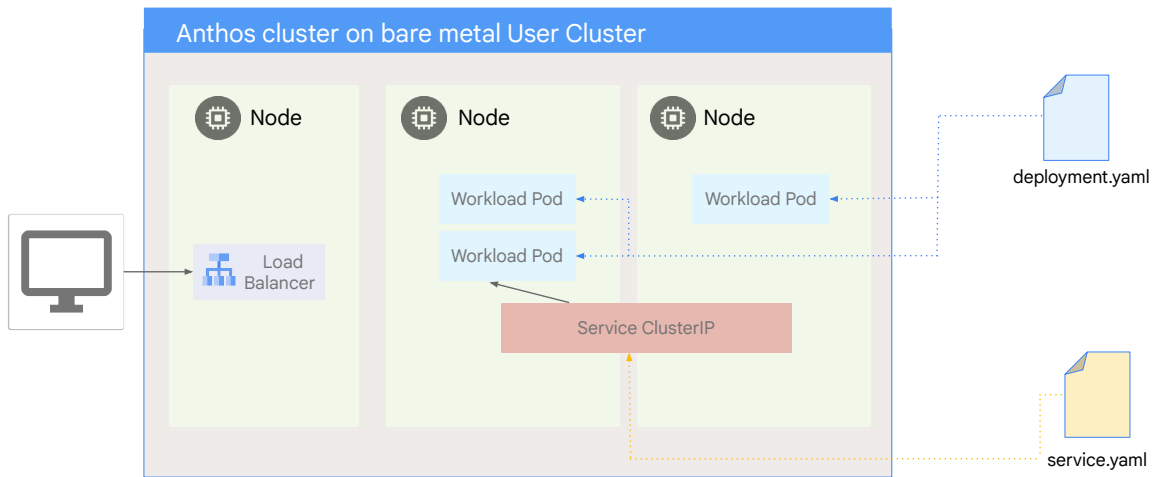
## Accessing workloads through a LoadBalancer Service



Google Cloud

Obviously, you begin by applying a deployment manifest, which results in pods being scheduled on worker nodes.

## Accessing workloads through a LoadBalancer Service



Google Cloud

Then, you apply a resource config that specifies a load balancer service. A Service ClusterIP is created and can be used within the cluster.

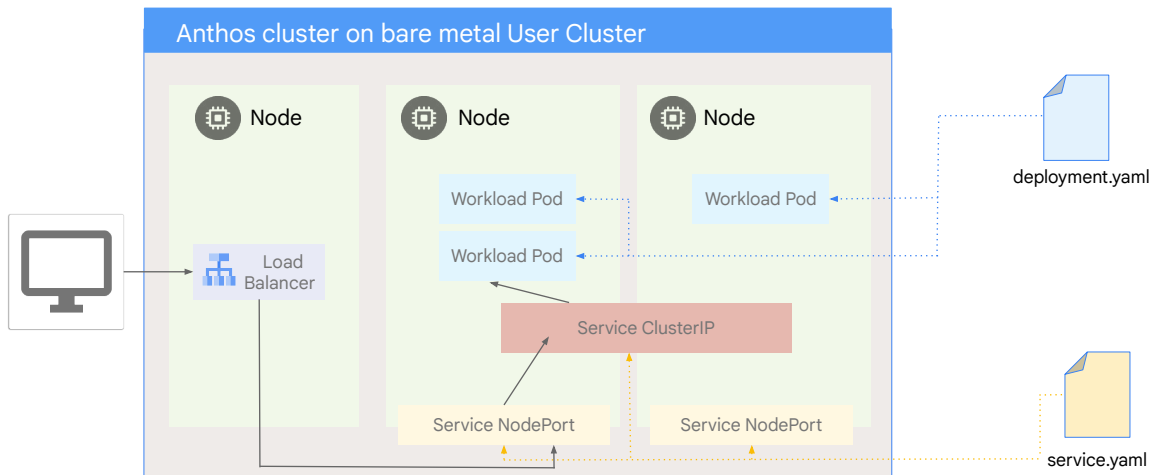
# Accessing workloads through a LoadBalancer Service

The diagram illustrates an Anthos cluster on bare metal User Cluster. It features three nodes, each containing a Service NodePort and a Service ClusterIP. A Load Balancer is shown outside the cluster, connected to the Service ClusterIP. Traffic from the Load Balancer is routed to the Service ClusterIP, which then directs it to the Workload Pods. The diagram also shows deployment.yaml and service.yaml files, indicating the configuration of the Workload Pods and the Service.

Google Cloud

A NodePort is also assigned to the service, and each node is configured to listen on that port and forward to the ClusterIP.

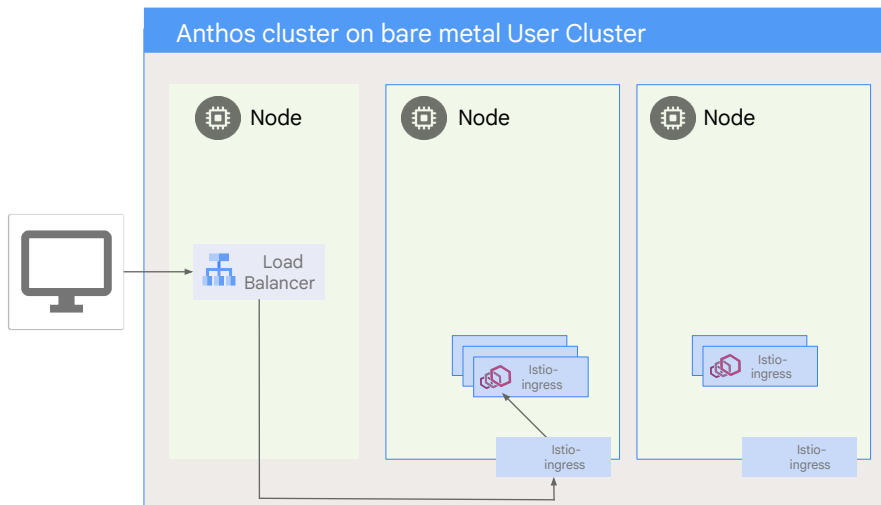
## Accessing workloads through a LoadBalancer Service



Google Cloud

Lastly, the load balancer is configured with a virtual IP assigned to the service, with requests sent to that IP forwarded to one of the nodes on specified NodePort port. With bundled load balancing, this load balancer configuration is done automatically.

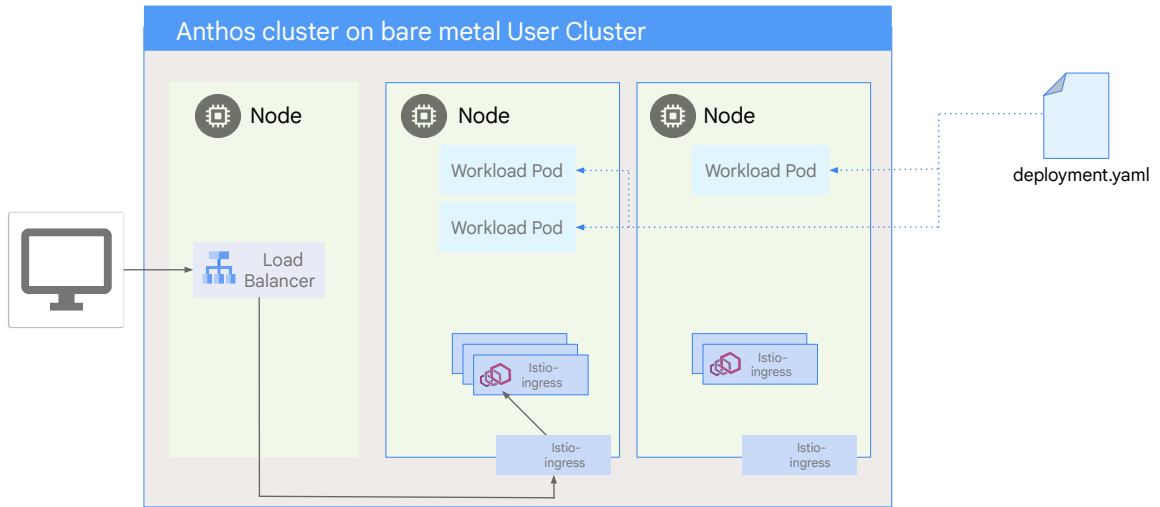
## Accessing workloads through a Kubernetes Ingress



Google Cloud

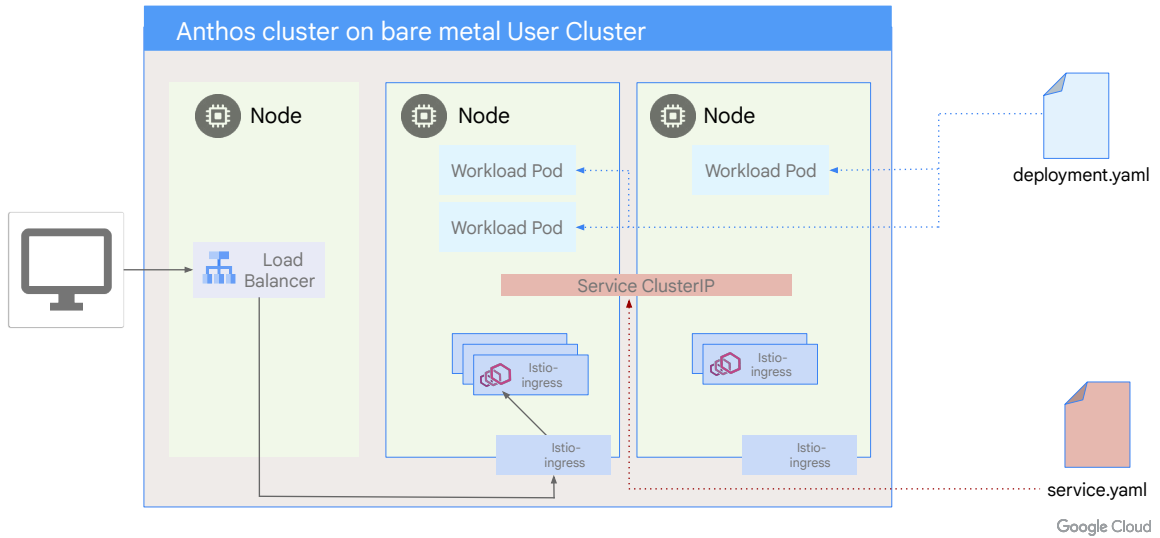
If you want to use a Kubernetes Ingress instead of a standard LoadBalancer service, you can do that on your bare metal cluster. The Ingress implementation on bare metal Anthos clusters is a bit unusual. When you create the user cluster, a subset of Istio functionality is installed. Istiod and the Istio Ingress gateway are installed in the gke-system namespace. The istio-ingress workload is comprised of a deployment istio-proxy which receives and processes incoming connections. The istio-ingress service is a LoadBalancer service; the load balancer is configured with a virtual IP, it forwards the traffic to a nodeport on a worker node, and the worker node forwards the traffic to the istio-proxy pod which handles ingress functionality.

# Accessing workloads through a Kubernetes Ingress



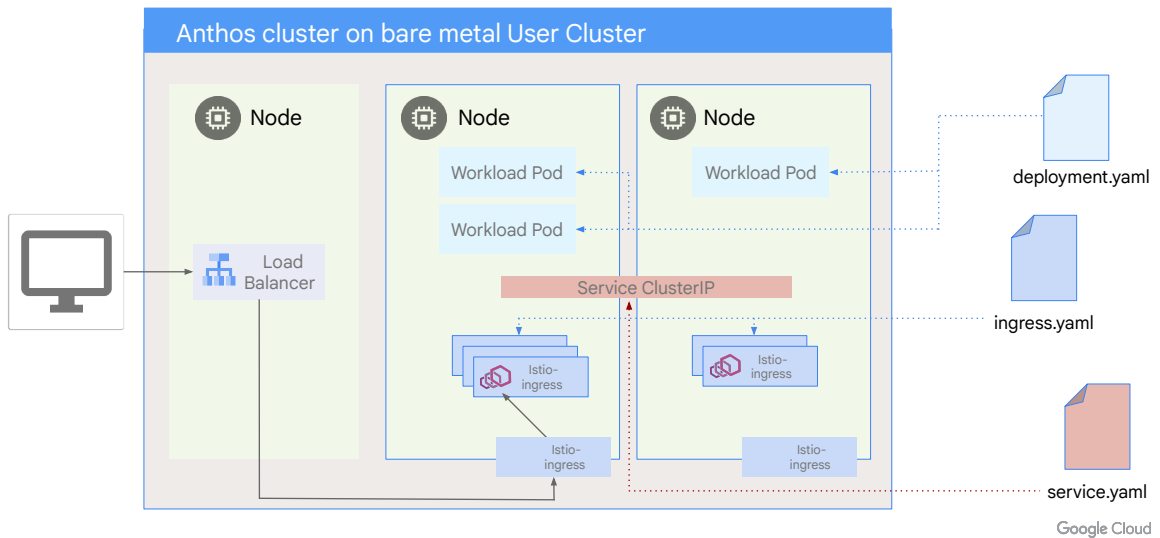
So, you begin by deploying the desired workload.

# Accessing workloads through a Kubernetes Ingress



Then, you deploy a ClusterIP service.

## Accessing workloads through a Kubernetes Ingress



Last, you create the Ingress resource, which configures the istio-proxy pods, so that when traffic flows through the load balancer, to the proxy pods, the proxy connects to the workloads behind the ClusterIP service.



# Deploying applications on Anthos clusters on bare metal

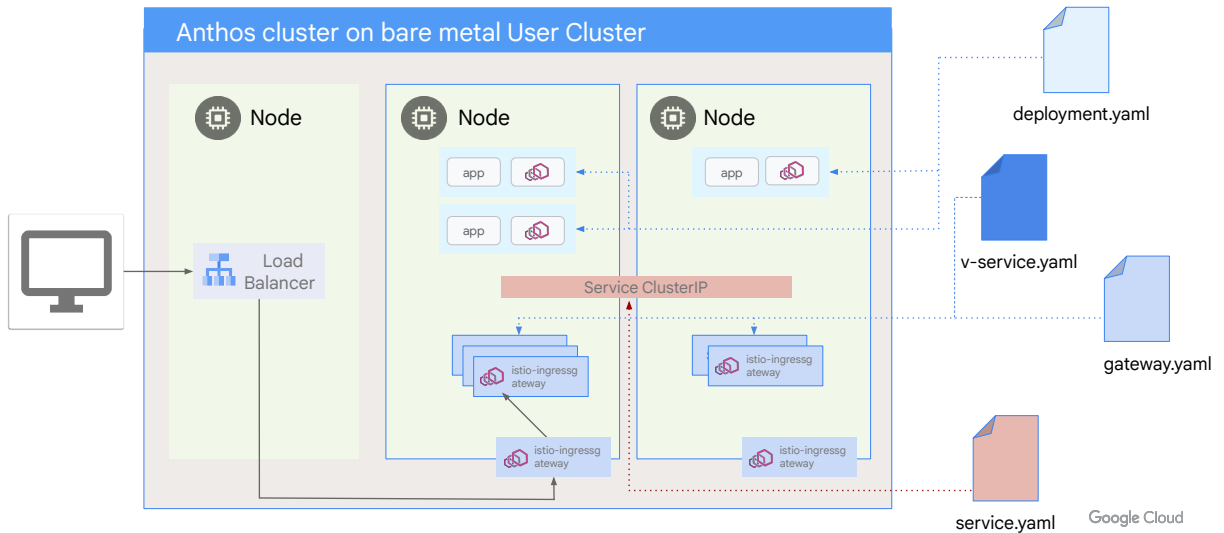


- You can run applications using regular Kubernetes constructs such as Deployments.
- There are two options to expose the applications outside of the cluster:
  - Using Kubernetes native constructs with:
    - Kubernetes Services type LoadBalancer
    - Kubernetes Ingress resource
  - **Using Anthos Service Mesh**

Google Cloud

If you are looking to leverage service mesh features, like build-in mTLS, monitoring, logging, and routing, you might choose to install and use Anthos Service Mesh.

## Accessing services via Anthos Service Mesh Ingress Gateways



With Anthos Service Mesh, you have a LoadBalancer service pointing to a workload comprised of istio-proxy pods, though the name used for both the workload and the service is now istio-ingressgateway. You create deployment and service resources just as you did with the Ingress-based access, but you also create a gateway resource and a virtual service resource, which configure the gateway proxies.

# Anthos Service Mesh installation

1. Download the `asmcli` utility to perform the ASM control plane installation on a cluster.
2. Install ASM the same way you would in Google Cloud.
3. To install a gateway:
  - Create a gateway namespace.
  - Enable sidecar-injection.
  - Use `kubectl` to apply.

```
1 curl
  https://storage.googleapis.com/csm-artifacts/asm/asmcli_1.11 > asmcli
  chmod +x asmcli
```


```
2 ./asmcli install \
  --project_id PROJECT_ID \
  --cluster_name CLUSTER_NAME \
  --cluster_location CLUSTER_LOCATION \
  --fleet_id FLEET_PROJECT_ID \
  --output_dir DIR_PATH \
  --enable_all \
  --ca mesh_ca
  --custom_overlay OVERLAY_FILE
```

```
3 kubectl create namespace $GATEWAY_NAMESPACE
  kubectl label namespace $NAMESPACE \
    istio.io/rev=$(kubectl -n istio-system get pods -l app=istiod -o json | jq \
      -r '.items[0].metadata.labels["istio.io/rev"]') --overwrite
  kubectl apply -n GATEWAY_NAMESPACE -f ./istio-ingressgateway.yaml
```


Google Cloud

Installing Anthos Service Mesh on Anthos Clusters on Bare Metal works the same way as in GKE. First, download the `asmcli` utility to perform the ASM control plane installation on a cluster.

Second, run the installer, choosing among several option for your installation. For example, you can use Google's Mesh CA or your own CA. Third, if you want to install gateways to enable north-south or east-west routing, you do so separately using `kubectl`, as shown in the command at the bottom.



## Today's agenda



01 Building the user cluster

---

02 Enabling authentication

---

03 Deploying applications

---

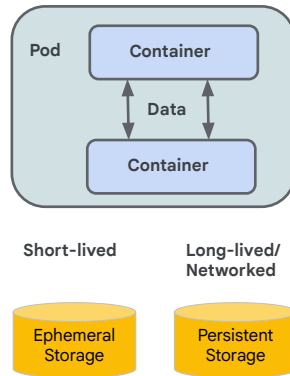
04 [Configuring storage](#)

---

Google Cloud

In Kubernetes, there are a variety of options available for storing and accessing data with your workloads. Let's look at those options and how they work on Anthos clusters on bare metal.

# Storage in Kubernetes



Google Cloud

Pods can use either ephemeral or persistent storage.

# Storage in Kubernetes

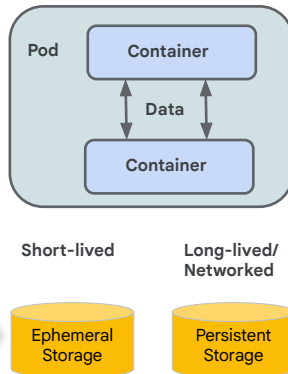
## Ephemeral Storage

Is not persisted after a pod dies.

Uses resources from the node's boot disk and is managed in the same way as CPU and memory.

Examples include:

- emptyDir
- ConfigMap
- Secrets



Google Cloud

Ephemeral volumes are created and deleted along with the pod, so data stored in these volumes doesn't exist beyond the life of the pod. This is useful for caching data, temporary data, or read-only configuration data and keys. Ephemeral volumes include configMaps, secrets, and emptyDir volumes where storage is generally coming from the kubelet base directory on the node. These types of ephemeral storage are all supported on Anthos clusters on bare metal.

# Storage in Kubernetes

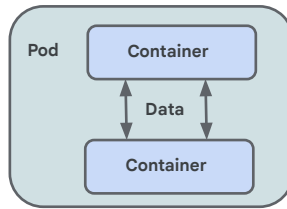
## Ephemeral Storage

Is not persisted after a pod dies.

Uses resources from the node's boot disk and is managed in the same way as CPU and memory.

Examples include:

- emptyDir
- ConfigMap
- Secrets



Short-lived

Long-lived/  
Networked



## Persistent Storage

Persists after a pod dies.

Uses PersistentVolumes and PersistentVolumeClaims to request disk space in the node or in a network file system.

- Use vendor's storage by installing CSI drivers.
- Use Anthos pre-configured Local Persistent Volumes for node storage.

Google Cloud

Persistent storage is provisioned and maintained outside the lifecycle of the pod, so data stored in these volumes can persist after pods are deleted. Persistent volumes are created using different storage classes, implemented by provisioners, and backed by local and network storage solutions. Kubernetes supports a variety of built in storage provisioners as well the Container Storage Interface—an extensible framework that allows storage vendors to create drivers that expose storage product functionality to pods.

# Local persistent volumes on bare metal clusters

## LVP share

- Every node has a section of the file system set aside and shared by all workloads for PVs.
- Anthos configures the file system during cluster creation.
- Workloads using this storage class share capacity and IOPS because the PVs are backed by the same shared file system.

Google Cloud

Anthos clusters on bare metal support two default strategies that allow operators to provision Persistent Volumes backed by the local file system on worker nodes. For LVP share volumes, Anthos sets aside disk space in the file system on every node, and multiple volumes write into the same directory on the same disk, sharing IOPs and storage capacity. The default storage class used to provision the LVP share Persistent Volumes is local-shared.



# Local persistent volumes on bare metal clusters

## LVP share

- Every node has a section of the file system set aside and shared by all workloads for PVs.
- Anthos configures the file system during cluster creation.
- Workloads using this storage class share capacity and IOPS because the PVs are backed by the same shared file system.

## LVP node mounts

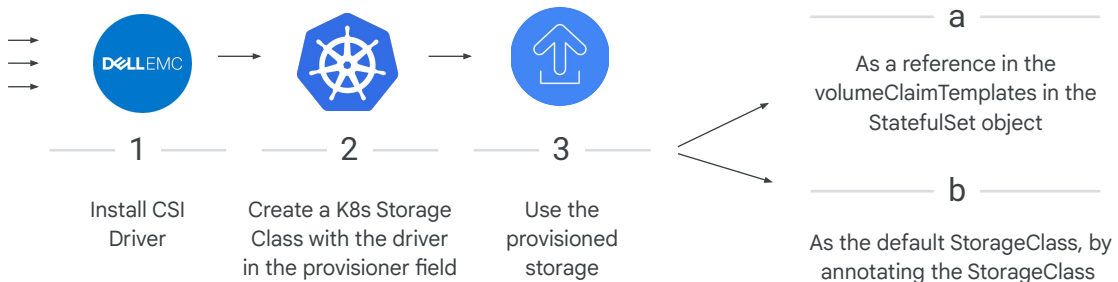
- A node mount and a local PV are created per disk attached to the node and have the same capacity as the disk.
- Additional mounts and PV can be added after cluster creation.
- Offer better isolation than LVP share because your workload has direct access to the disk instead of a shared file system.

Google Cloud

An alternative approach is to configure one Persistent Volume per available data disk on each node. The administrator configures each node, mounting disks on each node under the `/mnt/localpv-disk` directory, and configures the `lvpNodeMounts` resource on the cluster. The default storage class used for these volumes is `local-disks`. Performance will be better since the underlying storage isn't being shared by multiple volumes. You can also add additional mounts after cluster creation, growing your storage as needed.

## Configuring persistent storage CSI drivers

- Anthos clusters on bare metal are compatible with Container Storage Interface (CSI) v1.0 drivers.
- CSI is an open-standard API supported by many major storage vendors that enables Kubernetes to expose arbitrary storage systems to containerized workloads.
- Supports modern features such as snapshots and resizing.



Google Cloud

If you want to provide persistent volumes to your workloads using some other storage solution, often some sort of networked storage, you can use CSI 1.0 drivers on your bare metal clusters. You simply follow the vendor's instructions for installing the driver, create a storage class with the driver in the provisioner field, then reference that storage class volumeClaimTemplate section of your StatefulSet config file. You can also designate the new storage class as the cluster default.

## Anthos Ready storage partner CSI drivers

Google verifies the following requirements:

- Ability to deploy the storage CSI driver and its dependencies, using the Kubernetes framework.
- Core functions that customers require today, including dynamic provisioning of volumes, by using the Kubernetes-native storage APIs.
- The ability to manage storage for Kubernetes scale-up and scale-down scenarios.
- Workload portability with persistent storage for the stateful workloads.



Dell EMC



Hitachi Vantara



Portworx



PureStorage



Robin.io

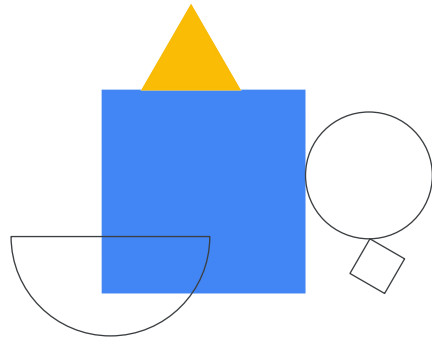
Google Cloud

Google recommends using CSI drivers from Anthos Ready storage partners, including Dell, Hitachi, PureStorage, and others. Check the online documentation for the latest list.

## Lab intro

🕒 30 min

Lab 092: Deploying Workloads  
on Anthos Clusters on Bare Metal



Google Cloud

# Questions and answers



# 1. What component do we use to get cluster credentials so we can access the cluster through Google Cloud?

1

Anthos Identity Service

2

Anthos Connect gateway

3

Anthos Connect Agent

4

Cloud Identity

5

Cloud IAM

# 1. What component do we use to get cluster credentials so we can access the cluster through Google Cloud?

1

Anthos Identity Service

2

Anthos Connect gateway

3

Anthos Connect Agent

4

Cloud Identity

5

Cloud IAM

## 2. Anthos on bare metal installs Anthos Service Mesh automatically.

1 True

2 False



## 2. Anthos on bare metal installs Anthos Service Mesh automatically.

1 True

2 False

Google Cloud

False, it only installs an Ingress controller using the envoy proxy. This Ingress is comprised of a deployment of Envoy proxy pods and a LoadBalancer Service for that istio-ingress deployment on the user cluster's ingress VIP specified on creation.

### 3. I need to install a Container Storage Interface (CSI) plugin if I want to...

1 Use long-term persistence with PersistentVolumes and PersistentVolumeClaims.

2 Use ephemeral persistence with emptyDir and ConfigMaps.

3 Use any kind of storage, both ephemeral and persistent.

### 3. I need to install a Container Storage Interface (CSI) plugin if I want to...

1

Use long-term persistence with PersistentVolumes and PersistentVolumeClaims.

2

Use ephemeral persistence with emptyDir and ConfigMaps.

3

Use any kind of storage, both ephemeral and persistent.