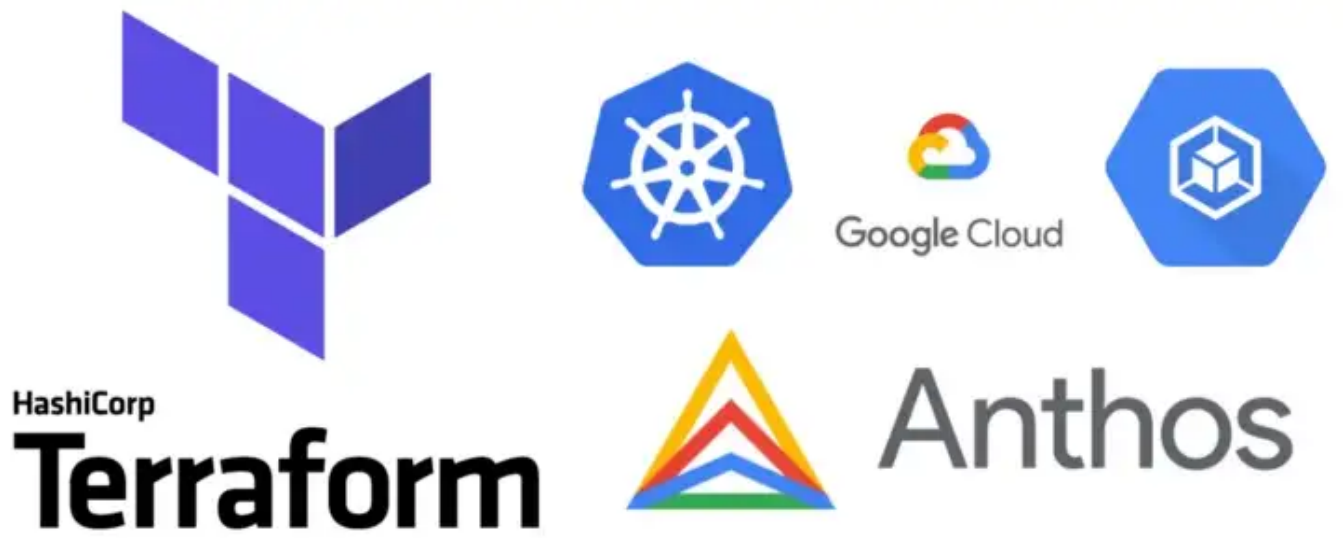
Pritee Dharme . Follow
Jan 21, 2022 · 13 min read · Listen

Terraform Module for Kubernetes Cluster with Google Anthos

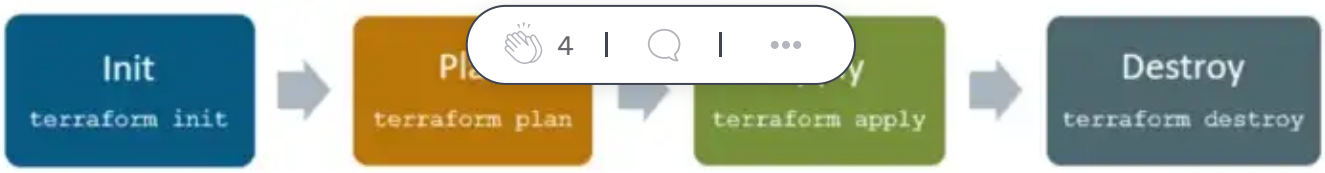


What is Terraform?

Terraform is an open-source infrastructure as code software tool. Infrastructure as Code (IaC) is a widespread terminology among DevOps professionals and a key DevOps practice in the industry. It is the process of managing and provisioning the complete IT infrastructure (comprises both physical and virtual machines) using machine-readable definition files. It helps in automating the complete data center by using programming scripts.

Terraform Lifecycle

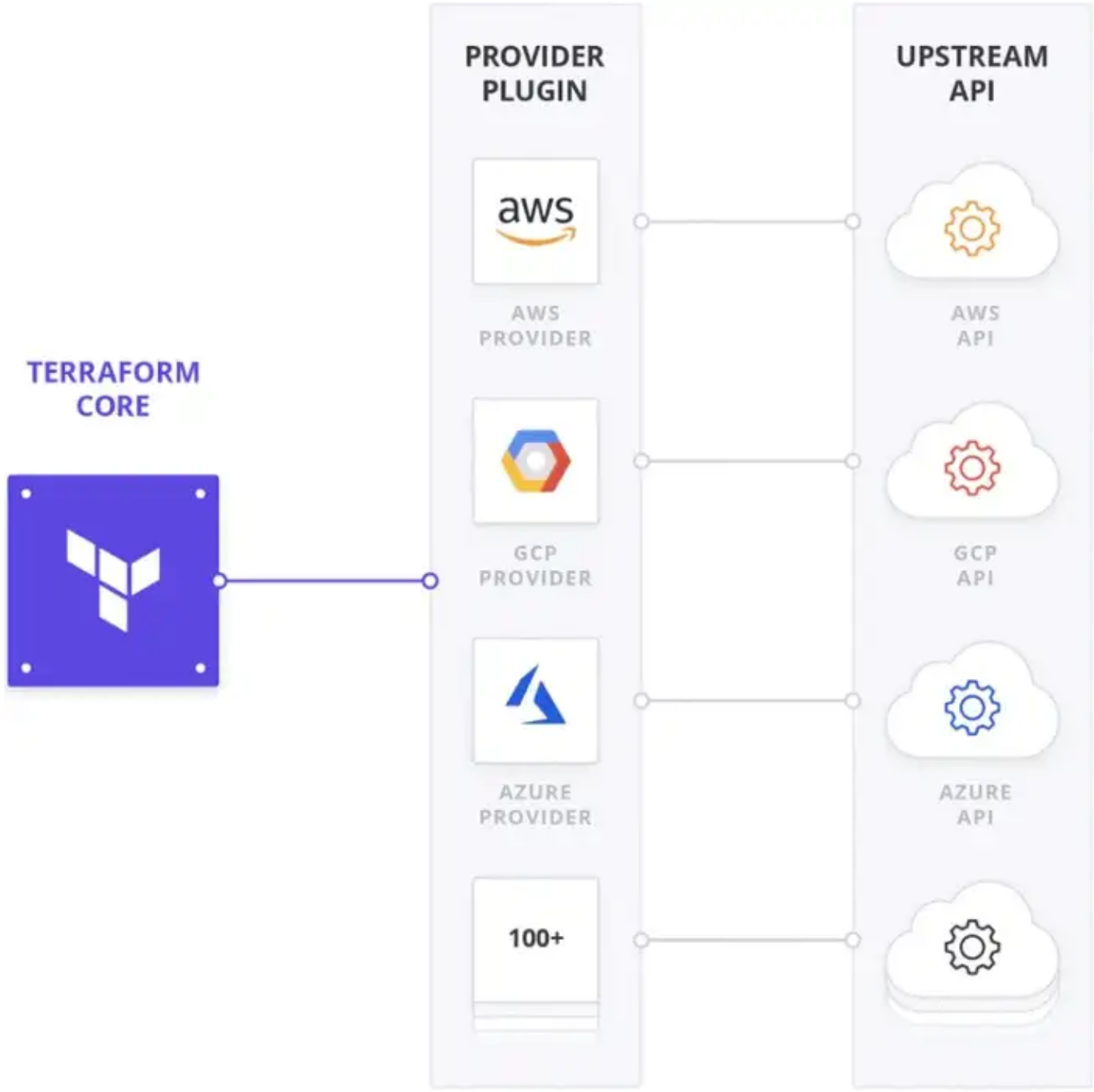
Terraform lifecycle consists of — **init**, **plan**, **apply**, and **destroy**.



1. **Terraform init** initializes the (local) Terraform environment. Usually executed only once per session.
2. **Terraform plan** compares the Terraform state with the as-is state in the cloud, build and display an
   execution plan. This does not change the deployment (read-only).
3. **Terraform apply** executes the plan. This potentially changes the deployment.
4. **Terraform destroy** deletes all resources that are governed by this specific terraform environment.

Terraform terminologies

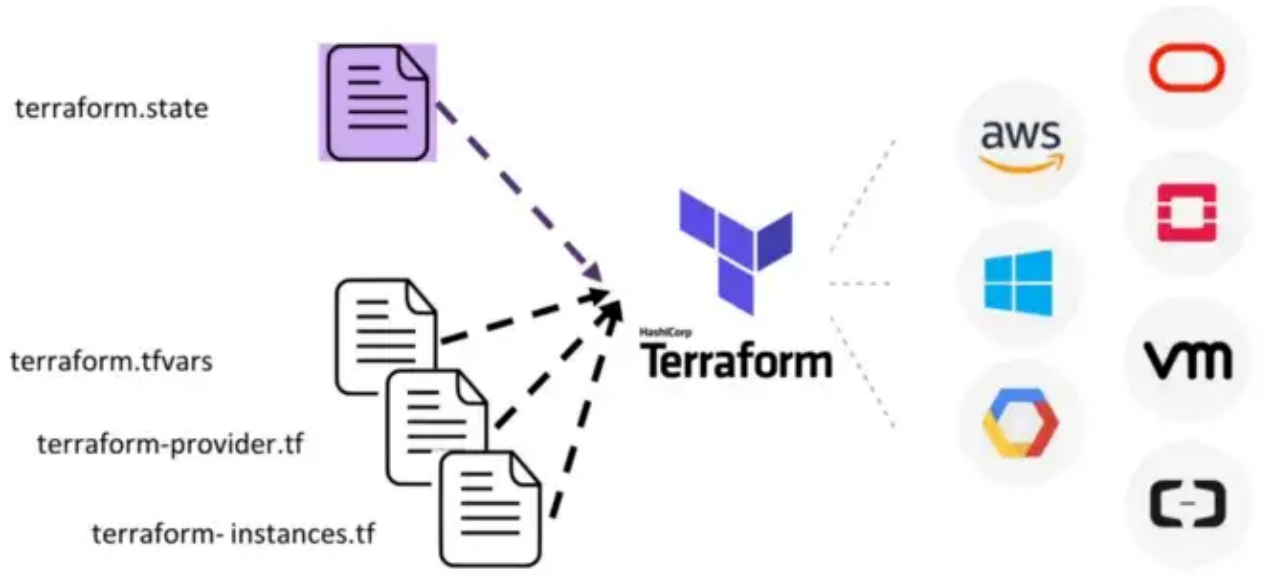
1. **Provider**: Terraform relies on plugins called “providers” to interact with cloud providers, SaaS providers, and other APIs. Terraform configurations must declare which providers they require so that Terraform can install and use them. Terraform supports more than 100 providers.
2. **Plan**: It is one of the stages in the Terraform lifecycle where it determines what needs to be created, updated, or destroyed to move from the real/current state of the infrastructure to the desired state.
3. **Module**: Any set of Terraform configuration files in a folder is a *module*. Every Terraform configuration has at least one module, known as its root module.
4. **Variables:** Reusability is one of the major benefits of Infrastructure as Code. In Terraform, we can use variables to make our configurations more dynamic. This means we are no longer hard coding every value into the configuration.
5. **State**: Terraform records information about what infrastructure is created in a Terraform *state* file. With the state file, Terraform is able to find the resources it created previously, supposed to manage and update them accordingly.
6. **Apply**: Apply is one of the stages in the Terraform lifecycle where it applies the changes real/current state of the infrastructure in order to achieve the desired state.



For installing terraform refer: Install Terraform | Terraform — HashiCorp Learn

Terraform Configuration Files

Configuration files are a set of files used to describe infrastructure in Terraform and have the file extensions **.tf** and **.tf.json**. Terraform uses a declarative model for defining infrastructure. Configuration files let you write a configuration that declares your desired state. Configuration files are made up of resources with settings and values representing the desired state of your infrastructure.



A Terraform configuration is made up of one or more files in a directory, provider binaries, plan files, and state files once Terraform has run the configuration.

1. Configuration file (*.tf files): Here we declare the provider and resources to be deployed along with the type of resource and all resources specific settings

2. Variable declaration file (variables.tf or variables.tf.json): Here we declare the input variables required to provision resources

3. Variable definition files (terraform.tfvars): Here we assign values to the input variables

4. State file (terraform.tfstate): a state file is created once after Terraform is run. It stores state about our managed infrastructure.

What is Google Anthos?

Google Cloud Anthos is a hybrid, cloud-agnostic container environment.

Google Cloud launched the Anthos platform in 2019, promisws customers a way to run Kubernetes workloads on-premises, in the Google Cloud, and, crucially, in other major public clouds including Amazon Web Services (AWS) and Microsoft Azure. Google Anthos helps in avoiding vendor locking.

Anthos enables us to run Kubernetes clusters anywhere, in both cloud and on-premises environments. We get consistent managed Kubernetes experience with simple installs as well as upgrades validated by Google. Anthos can run on our existing virtualized infrastructure and bare metal servers without a hypervisor

layer. Anthos simplifies your application stack, reduces the costs associated with licensing a hypervisor, and decreases time spent learning new skills.

Google Anthos Architecture

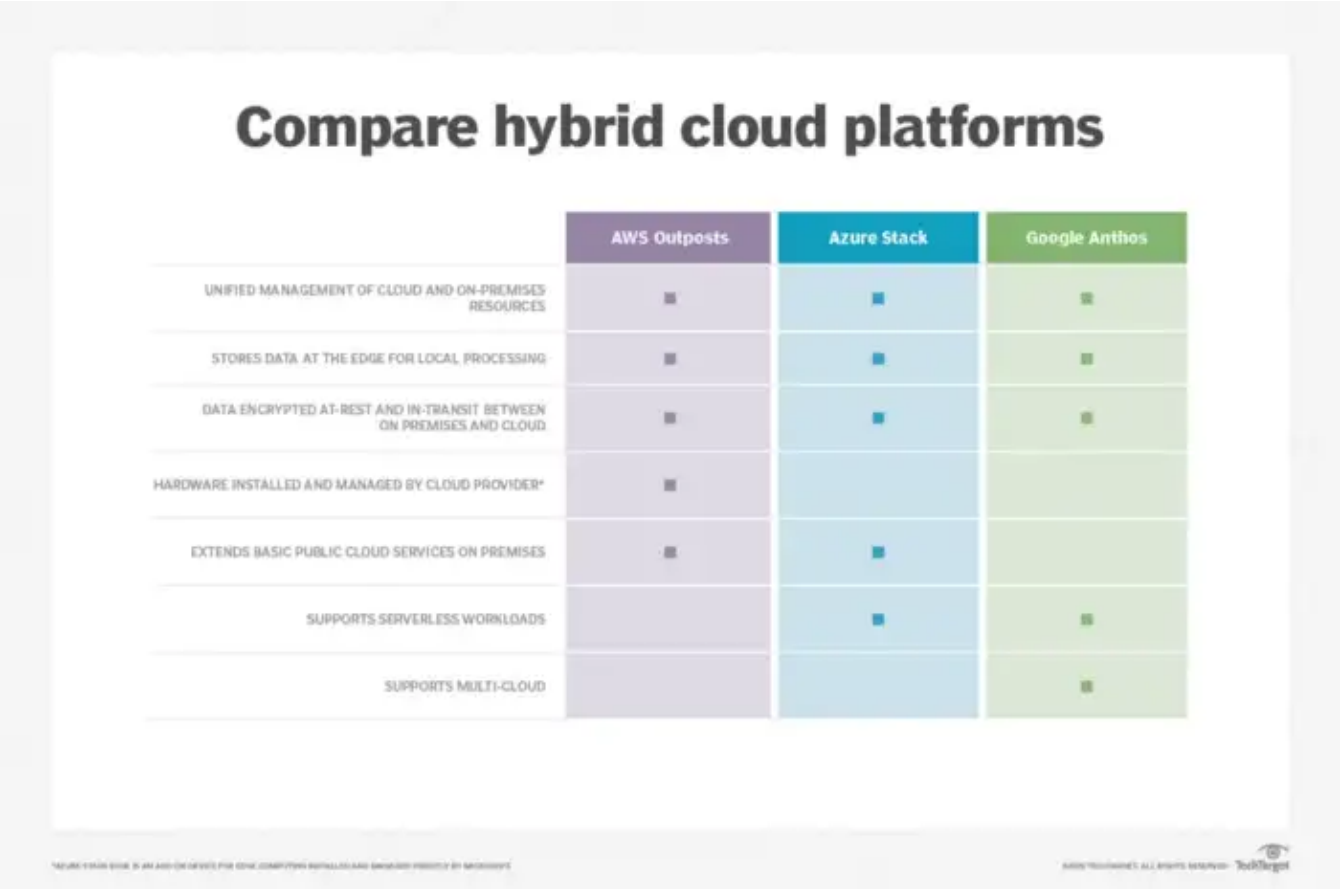
Kubernetes is great at managing a cluster of hosting resources, but it requires additional features to handle multiple autonomous resource pools, such as those hosted across multiple domains on premises or in various clouds. Without tools to manage multiple resource pools, IT teams would struggle with cooperative missions like cloud bursting or backup.

IT teams can use Anthos for federated operations management. It's a technique that accommodates application deployments across multiple resource pools, while still maintaining the identity and practices associated with each individual resource type. While Anthos is most often discussed for hybrid and multi-cloud applications, it can also support federated operations for VMs as well as containers. This makes it an almost-universal management framework.

Anthos architecture includes:

- Kubernetes, hosted in the data center, in Google Kubernetes Engine (GKE) in Google Cloud or in another public cloud.
- Anthos Config Management, for each cloud or domain.
- Istio service mesh for policy control, even if the applications don't require it.
- Google Operations for monitoring.
- Any additional Anthos tools desired from the Google Cloud Platform (GCP) Marketplace.

Google Connect for Anthos is the central networking element that links all these components. GKE is primarily managed through the Google Cloud Console, so a small GKE element is normally the logical center of Anthos. The rest of the hosting can be on any cloud or in any data center. You can also host GKE Anthos management in some data centers.

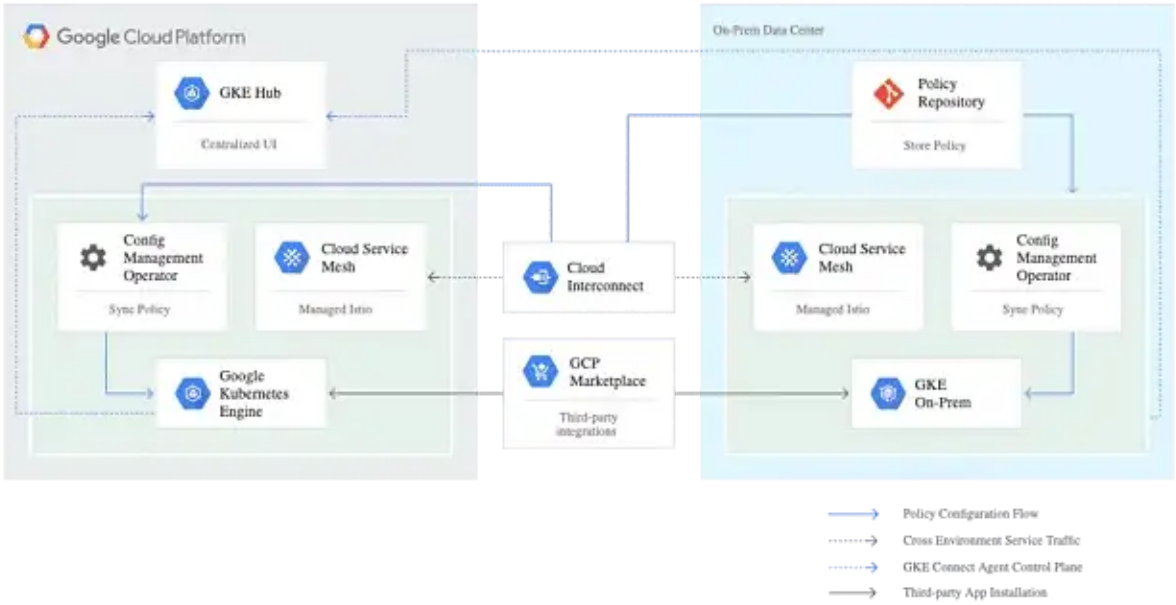


Cloud provider services that compete with Anthos are AWS Outposts and Azure Arc. The former represents a fairly basic hybrid cloud strategy of extending applications to on-premises servers managed by AWS.

Azure Arc is similar in architecture to Anthos, but it's not as tightly linked to Kubernetes and cloud-native behavior. That makes Anthos, overall, the most advanced hybrid-and-multi-cloud orchestration and management tool for those prepared to do more cloud-native development.

We can use identity sameness and namespace conventions to manipulate clusters, resources and services and create relationships between them. This facilitates the creation of simple policy sets to govern behavior. Without this feature, IT teams might have to define central policies in cluster- or cloud-specific ways, which would make Anthos little more than a repository for a diverging set of cluster policies.

Functionally, Anthos creates a GKE-hosted control plane that extends across all the connected resources and through which policies are exchanged. Within each cluster, an Anthos Config Management instance — and Config Sync for non-GKE clusters — provides the policy control resource, and the policies are stored in a central Git repository. All the standard Kubernetes policies are supported and can be applied as central policies via the single control plane to all clusters.



Although Anthos adheres to the Kubernetes open-source standards and is therefore able to work with virtually all Kubernetes clusters from different providers, there are a number of advantages that Anthos offers with its own Google Kubernetes Engine (GKE).

When new versions of Kubernetes are released, with new features, optimizations or security patches, they need to be rolled out to all active clusters. The pace at which this happens varies per provider. A Kubernetes update must also be able to deal with the provider's own implementations, must be tested properly because no clusters may fail due to an incorrect software update.

If we combine Kubernetes clusters from different providers, it's possible the performance will vary, that certain functionalities will not work properly, or that clusters will still be vulnerable due to missing patches.

By using a managed Kubernetes engine, we don't have to perform all those updates and tests ourselves, but this is done for us. The entire maintenance of the cluster is done by the provider or in this case by Google.

As part of Anthos, GKE on-premise has been introduced. GKE on-premise allows us to roll out GKE in our data center. There are also specific versions of GKE on-premise for AWS and Azure, making it possible to roll out Anthos at AWS and Azure. This is done automatically with the right configuration, Anthos can talk directly to the IaaS-APIs of AWS and Azure to build a GKE in their cloud environments. When choosing to work with GKE in all data centers and clouds, Google Anthos can guarantee that the Kubernetes version and therefore, the experience is the same

everywhere. Google ensures flawless updates for GKE that can be rolled out across all GKE clusters at the touch of a button.

Centralized and easier management of our Kubernetes clusters is one of the possibilities of Google Anthos. If companies want to simplify their Kubernetes management and configuration, Anthos is definitely a good option. These can be GKE clusters, but also from other suppliers.

For the configuration of the Kubernetes clusters, Google has developed Anthos Config Management. With Anthos Config Management, we control the configuration of the clusters. We can centrally manage the configurations of all Anthos clusters from a Git repository (this can be Github or Gitlab). Once the configurations are modified, they can be rolled out directly to all connected clusters. We can also configure it to disable local control of the Kubernetes clusters themselves. This way, the configuration from the Google Cloud console is always leading, and we can be sure that all our clusters are configured in the same way.

Configuring the clusters includes standard things like quotas for the amount of compute and memory. Also policies and compliance can be configured. For example, who has access to a cluster? Tailor-made compliance rules can also be applied. If a configuration violates the compliance rules, the configuration will simply be rejected or automatically reset.

Configurations for Kubernetes are extensive and can therefore quickly become complex. Anthos helps by testing the configurations for possible problems. This allows us to minimize errors.

With Anthos Service Mesh, Google takes over some more complicated tasks. With this tool, We can arrange things like security and monitoring between the different containers. The Anthos Service Mesh uses the open-source project Istio. Istio is an open-source service layer that is linked to a container (also called a sidecar) to monitor it. The Service Mesh operates independently of the container but is close to it. All traffic also passes through the Service Mesh, creating telemetry data for logging and monitoring the container. This data can be analyzed from the Google Cloud console.

Especially in the case of malfunctions or parts that do not want to work smoothly, all this telemetry and log data can quickly locate the problem and help apply the solution more quickly.

With all this log information, the Service Mesh can also map out exactly which containers communicate with each other and over which ports. These ports can be opened or closed. It is even possible to implement a Zero Trust Security Model, so that good behavior can be recorded once and everything else is blocked.

Google Anthos Pricing

Anthos charges for both pay-as-you-go and subscriptions. Prices are listed in U.S. dollars (USD). If you pay in a currency other than USD, the prices listed in your currency on Cloud Platform SKUs apply. A bill is sent out at the end of each billing cycle, listing previous usage and charges.

Anthos on public cloud	Pay-as-you-go (hourly)	Pay-as-you-go (monthly)[M]	Subscription (monthly)[M]
Anthos (Google Cloud) [GC]	$0.01096 /vCPU	$8 / vCPU	$6 / vCPU
Anthos (AWS) [AWS]	$0.01096 /vCPU	$8 / vCPU	$6 / vCPU
Anthos (attached clusters - multi-cloud)	$0.01096 /vCPU	$8 / vCPU	$6 / vCPU
Anthos on-premises	**Pay-as-you-go (hourly)**	**Pay-as-you-go (monthly)[M]**	**Subscription (monthly)[M]**
Anthos (on-premises - VMware)	$0.03288 / vCPU	$24 / vCPU	$18 / vCPU
Anthos (on-premises - bare metal)	$0.03288 / vCPU	$24 / vCPU	$18 / vCPU

M - Estimated monthly price based on 730 hours in a month.

GC - Anthos on Google Cloud pricing does not include charges for Google Cloud resources such as Compute Engine, Cloud Load Balancing, and Cloud Storage.

AWS - Anthos on AWS pricing does not include any costs associated with AWS resources such as EC2, ELB, and S3. The customer is responsible for any charges for their AWS resources.

If you are a new Anthos customer, you can try Anthos on Google Cloud for free up to $800 worth of usage, or for a maximum of 30 days, whichever comes earlier. During the trial, you are billed for the applicable fees and then credited at the same time for those fees up to $800. You are still billed for applicable infrastructure usage during the trial. If you currently have an Anthos subscription, then this trial is not available to you.

Now, let us see how we can automate Anthos using Terraform

First step would be installing the required software i.e. Terraform, kpt, and kustomize.

```
wget -q
https://releases.hashicorp.com/terraform/${TERRAFORM_VERSION}/terraf
```

```
orm_${TERRAFORM_VERSION}_linux_amd64.zip
unzip terraform_${TERRAFORM_VERSION}_linux_amd64.zip
chmod +x terraform
sudo mv -f terraform /usr/local/bin
rm -rf terraform_${TERRAFORM_VERSION}_linux_amd64.zip

curl -o kpt "https://storage.googleapis.com/kpt-
dev/latest/linux_amd64/kpt"
chmod +x kpt
mv ./kpt $WORK_DIR/bin

curl -o install_kustomize.sh
"https://raw.githubusercontent.com/kubernetes-
sigs/kustomize/master/hack/install_kustomize.sh"
chmod +x install_kustomize.sh && ./install_kustomize.sh
mv ./kustomize $WORK_DIR/bin
```

Next step would be creating base Terraform resources

1. variables.tf

```
variable "project_id" {
  description = "The project ID to host the cluster in"
}

variable "primary_region" {
  description = "The primary region to be used"
}
variable "primary_zones" {
  description = "The primary zones to be used"
}

variable "secondary_region" {
  description = "The secondary region to be used"
}
variable "secondary_zones" {
  description = "The secondary zones to be used"
}
```

2. terraform.tfvars

```
primary_region      = "us-central1"
primary_zones      = ["us-central1-a"]
secondary_region      = "us-west1"
secondary_zones      = ["us-west1-b"]
```

3. main.tf

```
provider "google" {
  project = var.project_id
  region  = var.primary_region
}

data "google_client_config" "current" {}

data "google_project" "project" {
  project_id = var.project_id
}

output "project" {
  value = data.google_client_config.current.project
}
```

This file defines the provider and creates data and output that can serve as input for other module values.

4. apis.tf

```
module "project-services" {
  source  = "terraform-google-modules/project-
factory/google//modules/project_services"

  project_id  = data.google_client_config.current.project
  disable_services_on_destroy = false
  activate_apis = [
    "compute.googleapis.com",
    "iam.googleapis.com",
    "container.googleapis.com",
    "cloudresourcemanager.googleapis.com",
    "anthos.googleapis.com",
    "cloudtrace.googleapis.com",
    "meshca.googleapis.com",
    "meshtelemetry.googleapis.com",
    "meshconfig.googleapis.com",
    "iamcredentials.googleapis.com",
    "gkeconnect.googleapis.com",
    "gkehub.googleapis.com",
    "monitoring.googleapis.com",
    "logging.googleapis.com"

  ]
}
```

For using Anthos with terraform we need to enable the following APIs: Compute Engine, Identity and Access Management, GKE, Resource Manager, Anthos, Cloud Trace, Anthos Service Mesh, Connect, Cloud Monitoring, and Cloud Logging.

1. Download all required providers:

```
$ terraform init
```

Testing and reviewing our configuration:

```
$ terraform plan -var project_id=${PROJECT_ID}
```

The output provides feedback about our configuration without applying any changes to your environment. Review the changes to check for any errors.

Applying the configuration:

```
$ terraform apply -var project_id=${PROJECT_ID}
```

The `apis.tf` file enabled our APIs, but no additional resources were defined or created.

Creating our cluster

We will provision the Anthos architecture by defining two clusters: a primary and a secondary cluster.

- clusters.tf

```
# Primary Cluster
module "primary-cluster" {
  name                    = "primary"
  project_id              = module.project-services.project_id
  source                  = "terraform-google-modules/kubernetes-
engine/google//modules/beta-public-cluster"
  version                 = "13.0.0"
  regional                = false
  region                  = var.primary_region
  network                 = "default"
  subnetwork              = "default"
  ip_range_pods           = ""
```

```
  ip_range_services       = ""
  zones                   = var.primary_zones
  release_channel         = "REGULAR"
  cluster_resource_labels = { "mesh_id" : "proj-
\${data.google_project.project.number}" }
  node_pools = [
    {
      name         = "default-node-pool"
      autoscaling  = false
      auto_upgrade = true

      node_count   = 5
      machine_type = "e2-standard-4"
    },
  ]

}

# Secondary Cluster
module "secondary-cluster" {
  name                    = "secondary"
  project_id              = module.project-services.project_id
  source                  = "terraform-google-modules/kubernetes-
engine/google//modules/beta-public-cluster"
  version                 = "13.0.0"
  regional                = false
  region                  = var.secondary_region
  network                 = "default"
  subnetwork              = "default"
  ip_range_pods           = ""
  ip_range_services       = ""
  zones                   = var.secondary_zones
  release_channel         = "REGULAR"
  cluster_resource_labels = { "mesh_id" : "proj-
\${data.google_project.project.number}" }

  node_pools = [
    {
      name         = "default-node-pool"
      autoscaling  = false
      auto_upgrade = true

      node_count   = 5
      machine_type = "e2-standard-4"
    },
  ]

}
```

- Provision the clusters:

```
$ terraform init
$ terraform plan -var project_id=${PROJECT_ID}
```

```
$ terraform apply -var project_id=${PROJECT_ID}
```

Registering with the project fleet

Fleets are a Google Cloud concept for logically organizing clusters and other resources, that let us use and manage multi-cluster capabilities and apply consistent policies across your systems. Fleets form a crucial part of how enterprise multi-cluster functionality works in Anthos.

To register your clusters, we can use the `hub` submodule.

- hub.tf

```
module "hub-primary" {
  source           = "terraform-google-modules/kubernetes-
engine/google//modules/hub"

  project_id       = data.google_client_config.current.project
  cluster_name     = module.primary-cluster.name
  location         = module.primary-cluster.location
  cluster_endpoint = module.primary-cluster.endpoint
  gke_hub_membership_name = "primary"
  gke_hub_sa_name = "primary"
}

module "hub-secondary" {
  source           = "terraform-google-modules/kubernetes-
engine/google//modules/hub"

  project_id       = data.google_client_config.current.project
  cluster_name     = module.secondary-cluster.name
  location         = module.secondary-cluster.location
  cluster_endpoint = module.secondary-cluster.endpoint
  gke_hub_membership_name = "secondary"
  gke_hub_sa_name = "secondary"
}
```

Applying the configuration

```
$ terraform init
$ terraform plan -var project_id=${PROJECT_ID}
$ terraform apply -var project_id=${PROJECT_ID}
```

Enabling Anthos Service Mesh

Anthos Service Mesh provides traffic management, security, and observability for microservices within GKE. To enable Anthos Service Mesh on a cluster, you can use the `asm` submodule. This module installs and enables Anthos Service Mesh on your clusters.

- asm.tf

```
module "asm-primary" {
  source           = "terraform-google-modules/kubernetes-
engine/google//modules/asm"
  version          = "13.0.0"
  project_id       = data.google_client_config.current.project
  cluster_name     = module.primary-cluster.name
  location         = module.primary-cluster.location
  cluster_endpoint = module.primary-cluster.endpoint

  asm_dir          = "asm-dir-\${module.primary-cluster.name}"

}

module "asm-secondary" {
  source           = "terraform-google-modules/kubernetes-
engine/google//modules/asm"
  version          = "13.0.0"
  project_id       = data.google_client_config.current.project
  cluster_name     = module.secondary-cluster.name
  location         = module.secondary-cluster.location
  cluster_endpoint = module.secondary-cluster.endpoint

  asm_dir = "asm-dir-\${module.secondary-cluster.name}"

}
```

Applying the configuration

```
$ terraform init
$ terraform plan -var project_id=${PROJECT_ID}
$ terraform apply -var project_id=${PROJECT_ID}
```

We can also enable configuration management on the clusters.

Anthos Config Management can sync assets from a Git repository and ensure that the assets are applied to multiple clusters.

We enable Anthos Configuration Management on our clusters and configure it to sync with a public sample repository. In a production environment, we can sync to your own private repository.

For Reference you can use my Github repository ..

GitHub - pritee55/Terraform_Module_Anthos You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or... github.com	

If you like it and want to contact me then below is my linked profile link , connect me there ..

Pritee Dharme - Learner Success Head at MLOPS Summer Program - LinuxWorld Informatics Pvt Ltd |... Hello All ... I am Pritee Dharme and Welcome in my profile. Currently I am Pursuing Computer Science And Engineering... www.linkedin.com	

Thank You So Much For Reading and Supporting ..

See you soon till then stay safe and healthy .. :)

Terraform Google Anthos Kubernetes Google Cloud