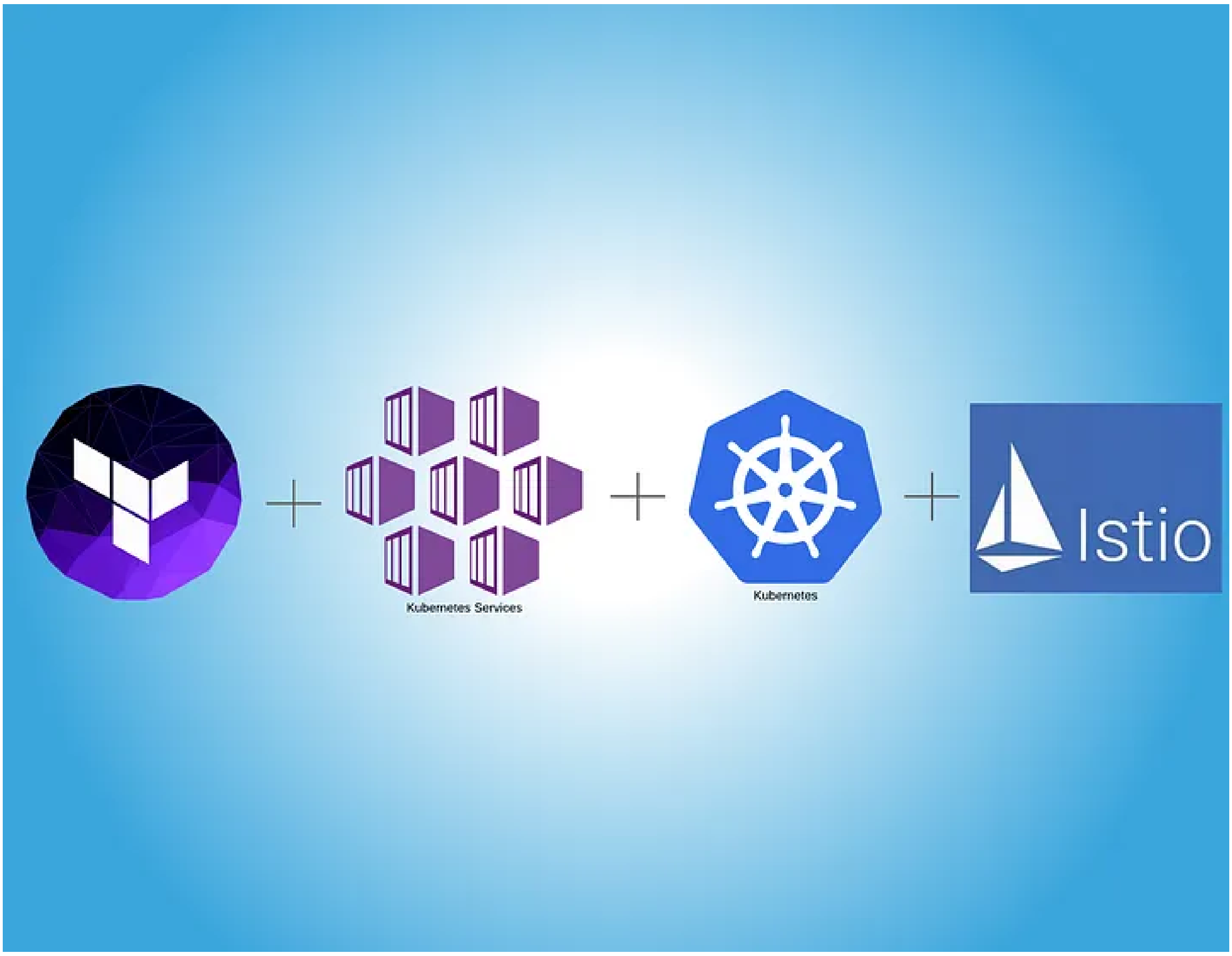
Vipin Agarwal Follow

Sep 6, 2020 · 5 min read · Listen

Save

Install Istio on Azure Kubernetes cluster using Terraform



In this article we will explore how to set-up your kubernetes cluster in Azure and install Istio (service-mesh) using Terraform in a single deployment.

Terminology before we begin-

- Azure Kubernetes Service (AKS) is managed kubernetes cluster as a service from Azure.
- Istio is an open-source service mesh that provides a key set of functionality across the microservices in a Kubernetes cluster.
- Terraform is an open-source tool for building your azure resources using Infrastructure as a code

In this article, you learn how to create/configure resources:

✔ Deploy AKS cluster in Azure using terraform
✔ Install Istio on AKS using terraform
✔ Creating kubernetes secret via terraform
✔ Validate Istio installation
✔ Access some the add-ons on Istio (like Kiali and Grafana)
✔ Install sample application on AKS again using terraform
✔ Cleaning-up resources in Azure

Before you begin:
Download and install the Istio istioctl client binary, Terraform, Azurecli, Kubernetes Yaml provider (this is optional for installing application in aks)

Create AKS cluster in Azure

The following code snippet create *myAKSCluster* with one node, this will take several minutes, part of the script will create resource group, vnet, subnet, azure ead group for managed identity (you can use service principal instead of azure ead group).

```
resource "azurerm_resource_group" "rg" {
  name     = "aks-resource-group"
  location = "eastus"
}

resource "azurerm_virtual_network" "vnet" {
  name                = "aks-vnet"
  resource_group_name = azurerm_resource_group.rg.name
  location            = azurerm_resource_group.rg.location
  address_space       = ["192.168.0.0/16"]
}

resource "azurerm_subnet" "subnet" {
  name                 = "aks-subnet"
  resource_group_name  = azurerm_resource_group.rg.name
  virtual_network_name = azurerm_virtual_network.vnet.name
  address_prefixes     = ["192.168.1.0/24"]
  service_endpoints    = ["Microsoft.ContainerRegistry"]
}

resource "azuread_group" "aks-admin-group" {
  name = "AKS-Aadmins"
}

resource "azurerm_kubernetes_cluster" "aks" {
  name                = "myAKSCluster"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  dns_prefix          = "someapplication"
  default_node_pool {
    name                  = "default"
    vnet_subnet_id        = azurerm_subnet.subnet.id
    type                  = "VirtualMachineScaleSets"
    availability_zones    = ["1", "2", "3"]
    enable_auto_scaling   = true
    enable_node_public_ip = false
    max_count             = 3
    min_count             = 1
    os_disk_size_gb       = 256
    vm_size               = "Standard_D2_v2"
  }
  role_based_access_control {
    enabled = true
    azure_active_directory {
      managed                = true
      admin_group_object_ids = [azuread_group.aks-admin-group.id]
    }
  }
  }
  identity {
    type = "SystemAssigned"
  }
  network_profile {
    network_plugin    = "azure"
    network_policy    = "azure"
    load_balancer_sku = "Standard"
  }

  addon_profile {
    aci_connector_linux {
      enabled = false
    }

    azure_policy {
      enabled = true
    }

    http_application_routing {
      enabled = false
    }

    kube_dashboard {
      enabled = true
    }
  }
}
```

```
##################Install Istio (Service Mesh) #######################################
resource "random_password" "password" {
  length           = 16
  special          = true
  override_special = "_%@"
}

data "azurerm_subscription" "current" {
}

resource "local_file" "kube_config" {
  content    = azurerm_kubernetes_cluster.aks.kube_admin_config_raw
  filename   = ".kube/config"
}


resource "null_resource" "set-kube-config" {
  triggers = {
    always_run = "${timestamp()}"
  }

  provisioner "local-exec" {
    command = "az aks get-credentials -n ${azurerm_kubernetes_cluster.aks.name} -g ${az
  }
  depends_on = [local_file.kube_config]
}


resource "kubernetes_namespace" "istio_system" {
  provider = kubernetes.local
  metadata {
    name = "istio-system"
  }
}

resource "kubernetes_secret" "grafana" {
  provider = kubernetes.local
  metadata {
    name      = "grafana"
    namespace = "istio-system"
    labels = {
      app = "grafana"
    }
  }
  data = {
    username   = "admin"
    passphrase = random_password.password.result
  }
  }
  type        = "Opaque"
  depends_on = [kubernetes_namespace.istio_system]
}

resource "kubernetes_secret" "kiali" {
  provider = kubernetes.local
  metadata {
    name      = "kiali"
    namespace = "istio-system"
```

```
apiVersion: install.istio.io/v1alpha2
kind: IstioControlPlane
spec:
  # Use the default profile as the base
  # More details at: https://istio.io/docs/setup/additional-setup/config-profiles/
  profile: default
  values:
    global:
      # Ensure that the Istio pods are only scheduled to run on Linux nodes
      defaultNodeSelector:
        beta.kubernetes.io/os: linux
      # Enable mutual TLS for the control plane
      controlPlaneSecurityEnabled: true
      mtls:
        # Require all service to service communication to have mtls
        enabled: false
    grafana:
      # Enable Grafana deployment for analytics and monitoring dashboards
      enabled: ${enableGrafana}
      security:
        # Enable authentication for Grafana
        enabled: true
    kiali:
      # Enable the Kiali deployment for a service mesh observability dashboard
      enabled: ${enableKiali}
    tracing:
      # Enable the Jaeger deployment for tracing
      enabled: ${enableTracing}
```

istio-aks.tmpl hosted with ❤ by **GitHub** **view raw**

```
  depends_on = [kubernetes_secret.grafana, kubernetes_secret.kiali, local_file.istio_c
}
```

kubernetes secret, and install Istio using istioctl. In your console you can see the install will deploy number of CRDs.

```
Applying manifests for these components:
- Tracing
- EgressGateway
- NodeAgent
- Grafana
- Policy
- Citadel
- CertManager
- IngressGateway
- Injector
- Prometheus
- PrometheusOperator
- Kiali
- Telemetry
- Galley
- Cni
- Pilot
- Base
- CoreDNS
NodeAgent is waiting on a prerequisite...
Telemetry is waiting on a prerequisite...
Galley is waiting on a prerequisite...
Cni is waiting on a prerequisite...
Grafana is waiting on a prerequisite...
Policy is waiting on a prerequisite...
Citadel is waiting on a prerequisite...
EgressGateway is waiting on a prerequisite...
Tracing is waiting on a prerequisite...
Kiali is waiting on a prerequisite...
PrometheusOperator is waiting on a prerequisite...
IngressGateway is waiting on a prerequisite...
Prometheus is waiting on a prerequisite...
CertManager is waiting on a prerequisite...
Injector is waiting on a prerequisite...
Pilot is waiting on a prerequisite...
Applying manifest for component Base
Waiting for CRDs to be applied.
CRDs applied.
Finished applying manifest for component Base
Prerequisite for Tracing has completed, proceeding with install.
Prerequisite for Injector has completed, proceeding with install.
Prerequisite for Telemetry has completed, proceeding with install.
Prerequisite for Policy has completed, proceeding with install.
Prerequisite for PrometheusOperator has completed, proceeding with
install.
Prerequisite for NodeAgent has completed, proceeding with install.
Prerequisite for IngressGateway has completed, proceeding with
install.
Prerequisite for Kiali has completed, proceeding with install.
Prerequisite for EgressGateway has completed, proceeding with
install.
Prerequisite for Galley has completed, proceeding with install.
Prerequisite for Grafana has completed, proceeding with install.
Prerequisite for Cni has completed, proceeding with install.
Prerequisite for Citadel has completed, proceeding with install.
Applying manifest for component Tracing
Prerequisite for Prometheus has completed, proceeding with install.
Prerequisite for Pilot has completed, proceeding with install.
Prerequisite for CertManager has completed, proceeding with install.
Applying manifest for component Kiali
Applying manifest for component Prometheus
Applying manifest for component IngressGateway
Applying manifest for component Policy
Applying manifest for component Telemetry
Applying manifest for component Citadel
Applying manifest for component Galley
Applying manifest for component Pilot
Applying manifest for component Injector
Applying manifest for component Grafana
Finished applying manifest for component Kiali
Finished applying manifest for component Tracing
Finished applying manifest for component Prometheus
Finished applying manifest for component Citadel
Finished applying manifest for component Policy
Finished applying manifest for component IngressGateway
Finished applying manifest for component Injector
Finished applying manifest for component Galley
Finished applying manifest for component Pilot
Finished applying manifest for component Grafana
Finished applying manifest for component Telemetry

Component IngressGateway installed successfully:
================================================

serviceaccount/istio-ingressgateway-service-account created
deployment.apps/istio-ingressgateway created
gateway.networking.istio.io/ingressgateway created
sidecar.networking.istio.io/default created
poddisruptionbudget.policy/ingressgateway created
horizontalpodautoscaler.autoscaling/istio-ingressgateway created
service/istio-ingressgateway created

...
```

If you can see above CRD’s installation in your console window means you have successfully deployed Istio to your AKS cluster. Let’s move on to validate our Istio installation and access Kiali and Grafana dashboards.

Verify a successful Istio installation

You can check if the Istio installation succeeded using the `verify-install` command which compares the installation on your cluster to a manifest you specify.

```
$ istioctl verify-install -f istio-aks.yaml
```

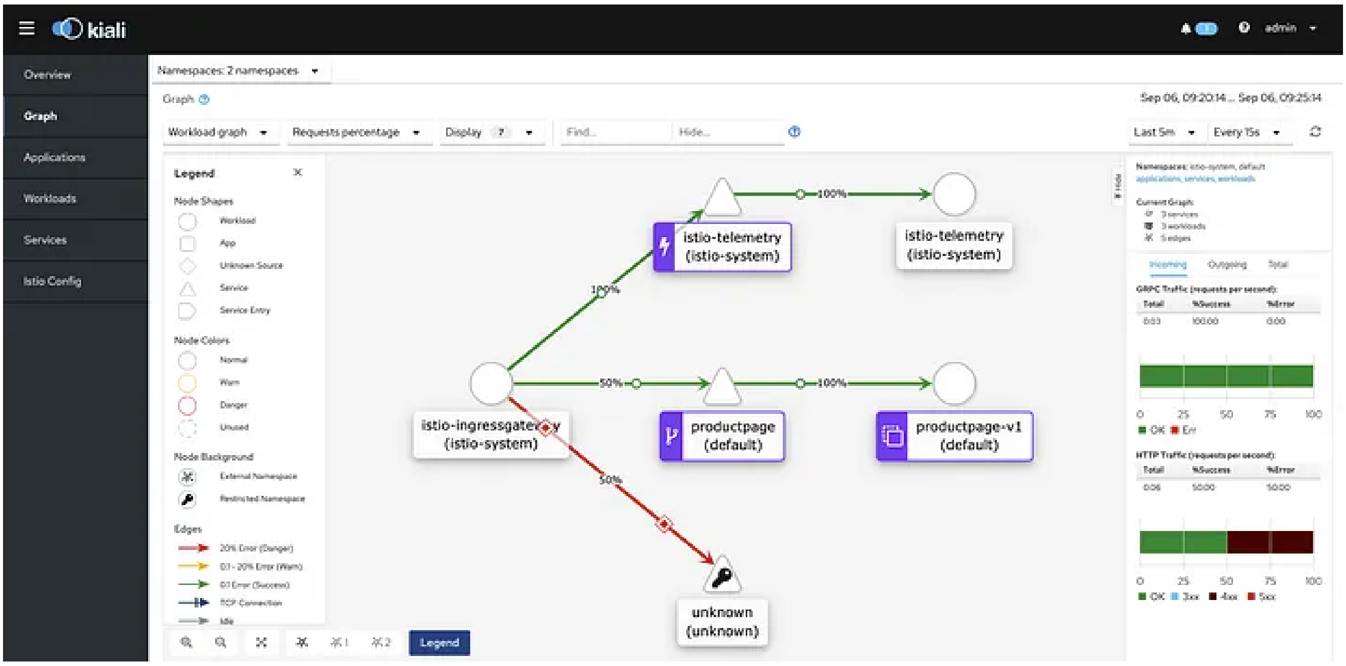
Access Kiali Dashboard

A service mesh observability dashboard is provided by kiali. First get the credentials from your cluster. To get credentials run the below code.

```
$ kubectl get secrets/kiali -n istio-system --template=
{{.data.passphrase}} | base64 -D
```

Then use the username ‘admin’ and password from the above output to access Kiali dashboard. To open the Kiali dashbaord securely as follows:

```
$ istioctl dashboard kiali
```



sample kiali dashboard

Access Grafana Dashboard

he analytics and monitoring dashboards for Istio are provided by Grafana . First get the credentials from your cluster. To get credentials run the below code.

```
$ kubectl get secrets/grafana -n istio-system --template=
{{.data.passphrase}} | base64 -D
```

Then use the username ‘admin’ and password from the above output to access Grafana dashboard. To open the Grafana dashbaord securely as follows:

```
$ istioctl dashboard grafana
```

Some other dashboards that are installed as part of our manifest file.

Jaeger for tracing

```
$ istioctl dashboard jaeger
```

Prometheus for metrics

```
$ istioctl dashboard prometheus
```

Cleanup resources

To clean-up all the resources you have created run

```
$ terraform destroy -auto-approve
```

Closing Remarks 🎉

I hope this article will help you with creating AKS cluster and setting-up Istio via manifest. Please feel free to share your feedback and experience in the comments section.

Kubernetes Istio Azure Kubernetes Service Terraform Iac