Open in app

Adilson Cesar

Follow 165 Followers About

Creating AKS Cluster in 5min using terraform

Adilson Cesar May 18, 2020 · 7 min read ★

Pre-requisites

Install Terraform

Follow the instructions here to install Terraform. When you're done, you should be able to run the terraform command:

Install Terraform

To use Terraform you will need to install it. HashiCorp distributes Terraform as a binary package. You can also install…

learn.hashicorp.com

```
# terraform
Usage: terraform [-version] [-help] <command> [args](...)
```

Create your Azure Service Principal

Firstly, login to the Azure CLI using:

```
# az login
You have logged in. Now let us find all the subscriptions to which
you have access...
[
  {
    "cloudName": "AzureCloud",
    "id": "xxxxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "isDefault": true,
    "name": "PlayGround",
    "state": "Enabled",
    "tenantId": "xxxxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "user": {
      "name": "adilsonbna@outlook.com",
      "type": "user"
    }
  }
```

Probably, You should have more than one subscription! However, you can specify the subscription to use via the following command:

```
# az account set --subscription="SUBSCRIPTION_ID"
```

We can now create the *Service Principal* which will have permissions to manage resources in the specified subscription using the following command:

```
# az ad sp create-for-rbac --role="Contributor" --
scopes="/subscriptions/SUBSCRIPTION_ID"

Creating a role assignment under the scope of
"/subscriptions/xxxxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
  Retrying role assignment creation: 1/36
{
  "appId": "xxxxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    tenant": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}
```

These values map to the Terraform variables like so:

> **`appId`** *is the* **`CLIENT_ID`** *defined above.*
>
> **`password`** *is the* **`CLIENT_SECRET`** *defined above.*
>
> **`tenant`** *is the* **`TENANT_ID`** *defined above.*

```
# az login --service-principal -u CLIENT_ID -p CLIENT_SECRET --
tenant TENANT_ID

[
  {
    "cloudName": "AzureCloud",
    "id": "xxxxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "isDefault": true,
    "name": "PlayGround",
    "state": "Enabled",
    "tenantId": "xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "user": {
      "name": "xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
      "type": "servicePrincipal"
    }
  }
]
```

Once logged in as the service principal type changed to **servicePrincipal.** So, we should be able to list the VM sizes by specifying an Azure region, for example here we use the `East US` region:

```
# az vm list-sizes --location eastus
```

All set! Let's deploy it.

Configuring the Service Principal in Terraform

The following Provider block can be specified where `2.5.0` is the version of the Azure Provider that you'd like to use:

```
    version = "=2.5.0"
    features {}
  }
  EOF
```

At this point running either **`terraform init`**, **`terraform plan`** or **`terraform apply`** should allow terraform to run using the service principal to authenticate.

```
# terraform init

Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
Downloading plugin for provider "azurerm" (hashicorp/azurerm)
2.5.0...Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform
plan" to see any changes that are required for your infrastructure.
All Terraform commands should now work.

If you ever set or change modules or backend configuration for
Terraform, rerun this command to reinitialize your working
directory. If you forget, other commands will detect it and remind
you to do so if necessary.

# terraform plan
Refreshing Terraform state in-memory prior to plan...The refreshed
state will be used to calculate this plan, but will not be
persisted to local or remote state storage.
-----------------------------------------------------------------------
No changes. Infrastructure is up-to-date.This means that Terraform
did not detect any differences between your configuration and real
physical resources that exist. As a result, no actions need to be
performed.
```

Creating terraform modules

`main.tf`, `variables.tf`, `outputs.tf`. These are the recommended filenames for a minimal module, even if they're empty. `main.tf` should be the primary entrypoint. For a simple module, this may be where all the resources are created. For a complex module, resource creation may be split into multiple files but any nested module calls should be in the main file. `variables.tf` and `outputs.tf` should contain the declarations for variables and outputs, respectively.

```
├── README.md
├── main.tf
├── variables.tf
├── outputs.tf
```

Main file:

```
# cat >> main.tf <<'EOF'

provider "azurerm" {
  subscription_id = var.subscription_id
  client_id       = var.client_id
  client_secret   = var.client_secret
  tenant_id       = var.tenant_id

features {}
}

resource "azurerm_resource_group" "k8s" {
  name     = var.resourcename
  location = var.location
}

resource "azurerm_kubernetes_cluster" "k8s" {
  name                = var.clustername
  location            = azurerm_resource_group.k8s.location
  resource_group_name = azurerm_resource_group.k8s.name
  dns_prefix          = var.dnspreffix

default_node_pool {
    name       = "default"
    node_count = var.agentnode
    vm_size    = var.size
  }

service_principal {
    client_id     = var.client_id
    client_secret = var.client_secret
  }
}
EOF
```

Variable file:

```
# cat >> variables.tf <<'EOF'
```

```
variable "clustername" {
  default = "kubernetes-aks1"
}

variable "location" {
  default = "East US"
}

variable "dnspreffix" {
  default = "kubecluster"
}

variable "size" {
  default = "Standard_D2_v2"
}

variable "agentnode" {
  default = "1"
}

variable "subscription_id" {
  default = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}

variable "client_id" {
  default = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}

variable "client_secret" {
  default = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}

variable "tenant_id" {
  default = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}
EOF
```

Output file:

```
# cat >> output.tf <<'EOF'

output "client_certificate" {
  value =
azurerm_kubernetes_cluster.k8s.kube_config.0.client_certificate
}

output "kube_config" {
  value = azurerm_kubernetes_cluster.k8s.kube_config_raw
  value = azurerm_kubernetes_cluster.k8s.fqdn
}
EOF
```

Let's check your code:

```
# terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will
not be persisted to local or remote state storage.
--------------------------------------------------------------------
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

# azurerm_kubernetes_cluster.k8s will be created
  + resource "azurerm_kubernetes_cluster" "k8s" {
      + dns_prefix              = "kubecluster"
      + fqdn                    = (known after apply)
      + id                      = (known after apply)
      + kube_admin_config       = (known after apply)
      + kube_admin_config_raw = (sensitive value)
      + kube_config             = (known after apply)
      + kube_config_raw         = (sensitive value)
      + kubernetes_version      = (known after apply)
      + location                = "eastus"
      + name                    = "kubernetes-aks1"
      + node_resource_group     = (known after apply)
      + private_fqdn            = (known after apply)
      + resource_group_name     = "k8s-resources"

+ addon_profile {
          + aci_connector_linux {
              + enabled     = (known after apply)
              + subnet_name = (known after apply)
            }

+ azure_policy {
              + enabled = (known after apply)
            }

+ http_application_routing {
              + enabled                              = (known after apply)
              + http_application_routing_zone_name = (known after
apply)
            }
```

```
+ oms_agent {
              + enabled                    = (known after apply)
              + log_analytics_workspace_id = (known after apply)
            }
        }

+ default_node_pool {
            + max_pods        = (known after apply)
            + name            = "default"
            + node_count      = 1
            + os_disk_size_gb = (known after apply)
            + type            = "VirtualMachineScaleSets"
            + vm_size         = "Standard_D2_v2"
        }

+ network_profile {
            + dns_service_ip     = (known after apply)
            + docker_bridge_cidr = (known after apply)
            + load_balancer_sku  = (known after apply)
            + network_plugin     = (known after apply)
            + network_policy     = (known after apply)
            + outbound_type      = (known after apply)
            + pod_cidr           = (known after apply)
            + service_cidr       = (known after apply)

+ load_balancer_profile {
              + effective_outbound_ips    = (known after apply)
              + managed_outbound_ip_count = (known after apply)
              + outbound_ip_address_ids   = (known after apply)
              + outbound_ip_prefix_ids    = (known after apply)
            }
        }

+ role_based_access_control {
            + enabled = (known after apply)

+ azure_active_directory {
              + client_app_id     = (known after apply)
              + server_app_id     = (known after apply)
              + server_app_secret = (sensitive value)
              + tenant_id         = (known after apply)
            }
        }

+ service_principal {
            + client_id     = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx"
            + client_secret = (sensitive value)
        }

+ windows_profile {
            + admin_password = (sensitive value)
            + admin_username = (known after apply)
```

Open in app

```
  # azurerm_resource_group.k8s will be created
    + resource "azurerm_resource_group" "k8s" {
        + id       = (known after apply)
        + location = "eastus"
        + name     = "k8s-resources"
      }

  Plan: 2 to add, 0 to change, 0 to destroy.
  ------------------------------------------------------------------
  Note: You didn't specify an "-out" parameter to save this plan, so
  Terraform can't guarantee that exactly these actions will be
  performed if "terraform apply" is subsequently run.
```

Go ahead apply it!

```
  # terraform apply

  Plan: 2 to add, 0 to change, 0 to destroy.

  Do you want to perform these actions?
    Terraform will perform the actions described above.
    Only 'yes' will be accepted to approve.

  Enter a value: yes
```

In a couple of minutes, you will have your own Kubernetes cluster built.

```
  azurerm_kubernetes_cluster.k8s: Creation complete after 4m17s
  [resources/providers/Microsoft.ContainerService/managedClusters/kube
  rnetes-aks1]

  Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Accessing your cluster

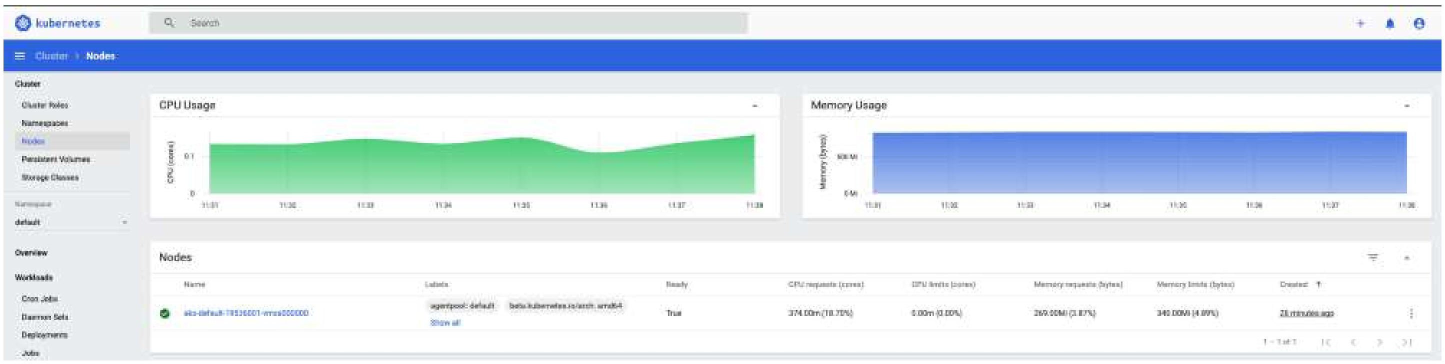
```
  # az aks get-credentials --name kubernetes-aks1 --resource-group
  k8s-resources
  Merged "kubernetes-aks1" as current context in
  /Users/root/.kube/config

  # kubectl get nodes
  NAME                                STATUS   ROLES   AGE    VERSION
```

Web UI (Dashboard)

```
# kubectl proxy
Starting to serve on 127.0.0.1:8001
```

> *Don't forget to get your token from* ***.kube/config.***



Nodes

Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Created
aks-default-19536001-vmss000000	agentpool: default beta.kubernetes.io/arch: amd64 Show all	True	374.00m (18.70%)	0.00m (0.00%)	269.00Mi (3.87%)	340.00Mi (4.89%)	28 minutes ago

1 – 1 of 1

http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/

Destroy Infrastructure

```
# terraform destroy
```

Destroy Infrastructure

You have now seen how to build and change infrastructure. Before moving on to creating multiple resources and showing...

learn.hashicorp.com

Enjoy it! \o/

References:

Kubernetes: Getting Started with Kubernetes provider - Terraform by HashiCorp

Kubernetes (K8S) is an open-source workload scheduler with focus

Terraform by HashiCorp

Deliver infrastructure as code with Terraform Collaborate and share configurations Evolve and version your...

www.terraform.io

Terraform Kubernetes DevOps Iac Azure