**Create an AKS Cluster in Azure with Terraform**

**Introduction**

In this lab, using the Azure portal, you will configure the Cloud Shell and download and run the lab setup script. Next, you will import the resource group. Then, you will add your AKS, variable, and outputs to the configuration. Lastly, you will deploy your Kubernetes cluster resources and verify that the cluster is up and healthy.

**Solution**

**Set Up the Cloud Shell and Lab Environment**

1. In the Azure portal, click on the Cloud Shell icon (>\_) at the top of the page, to the right of the search bar.
2. Select **Bash**.
3. Click **Show advanced settings**.
4. For the **Cloud Shell region**, select the same region as your resource group location (This will be noted above, in the portal).
5. For **Storage account**, choose **Use existing**.
6. Under **File share**, select **Create new** and type in the name of *terraform*.
7. Click **Create storage**. Your Cloud Shell should begin to configure.ss

**Import the Resource Group**

1. Run terraform init to initialize the working directory.
2. Run az group list to look up the subscription ID.
3. Copy the subscription ID to your clipboard. It should be located on the top line after "id":. Make sure to copy all of the characters in between the quotation marks.
4. Run the following command, making sure to paste in your copied subscription ID to replace <SUBSCRIPTION\_ID>:

terraform import azurerm\_resource\_group.k8s <SUBSCRIPTION\_ID>

**Note:** It may take a minute to import your resource.

1. Run vim providers.tf to edit the file.

**Note:** When copying and pasting code into Vim from the lab guide, first enter :set paste (and then i to enter insert mode) to avoid adding unnecessary spaces and hashes. To save and quit the file, press **Escape** followed by :wq. To exit the file *without* saving, press **Escape** followed by :q!.

1. Delete the comment hashes (#) in front of name and location.
2. Replace the placeholder <RESOURCE\_GROUP\_NAME> next to name.
   * Copy the resource group name located at the top left of the Azure portal, under **Home**.
   * Paste it into the file, to replace <RESOURCE\_GROUP\_NAME> making sure not to replace the quotation marks.
3. Replace the placeholder <RESOURCE\_GROUP\_LOCATION> next to location.
   * Copy the resource group location listed to the right of **Location** in the Azure portal. (If you hover over it, a copy icon should appear that you can click to copy it to your clipboard.)
   * Paste it into the file, to replace <RESOURCE\_GROUP\_LOCATION> making sure not to replace the quotation marks.
4. Type **Esc** followed by *:wq* to save and quit the file.
5. Run the following command to create an SSH key:

ssh-keygen -m PEM -t rsa -b 4096

1. Hit Enter to keep the defaults.
2. Hit Enter to leave the passphrase empty.
3. Hit Enter again to create your key pair.

**Add the AKS Config, Variables, and Outputs to the Configuration**

1. Run vim aks.tf to create your first configuration file.
2. Enter the following configuration:

resource "azurerm\_kubernetes\_cluster" "k8s" {

name = var.cluster\_name

location = azurerm\_resource\_group.k8s.location

resource\_group\_name = azurerm\_resource\_group.k8s.name

dns\_prefix = var.dns\_prefix

linux\_profile {

admin\_username = "ubuntu"

ssh\_key {

key\_data = file(var.ssh\_public\_key)

}

}

default\_node\_pool {

name = "agentpool"

node\_count = var.agent\_count

vm\_size = "Standard\_D2s\_v3"

os\_disk\_size\_gb = 30

}

service\_principal {

client\_id = var.aks\_service\_principal\_app\_id

client\_secret = var.aks\_service\_principal\_client\_secret

}

network\_profile {

load\_balancer\_sku = "Standard"

network\_plugin = "kubenet"

}

tags = {

Environment = "Development"

}

}

1. Type **Esc** followed by *:wq* to save and quit the file.
2. Run vim variables.tf to create your next configuration file.
3. Enter the following configuration. Be sure to replace <YOUR\_RESOURCE\_GROUP\_LOCATION> with the location of your resource group, and replace <SERVICE\_PRINCIPAL\_APP\_ID> and <SERVICE\_PRINCIPAL\_CLIENT\_SECRET> with the service principal IDs generated for this lab, which can be found in the lab credentials section.

variable "resource\_group\_location" {

default = "<YOUR\_RESOURCE\_GROUP\_LOCATION>"

}

variable "agent\_count" {

default = 3

}

variable "ssh\_public\_key" {

default = "~/.ssh/id\_rsa.pub"

}

variable "dns\_prefix" {

default = "k8sguru"

}

variable cluster\_name {

default = "k8sguru"

}

variable aks\_service\_principal\_app\_id {

default = "<SERVICE\_PRINCIPAL\_APP\_ID>"

}

variable aks\_service\_principal\_client\_secret {

default = "<SERVICE\_PRINCIPAL\_CLIENT\_SECRET>"

}

1. Type **Esc** followed by *:wq* to save and quit the file.
2. Run vim output.tf to create your final configuration file.
3. Enter the following configuration:

output "resource\_group\_name" {

value = azurerm\_resource\_group.k8s.name

}

output "client\_key" {

value = azurerm\_kubernetes\_cluster.k8s.kube\_config.0.client\_key

}

output "client\_certificate" {

value = azurerm\_kubernetes\_cluster.k8s.kube\_config.0.client\_certificate

}

output "cluster\_ca\_certificate" {

value = azurerm\_kubernetes\_cluster.k8s.kube\_config.0.cluster\_ca\_certificate

}

output "cluster\_username" {

value = azurerm\_kubernetes\_cluster.k8s.kube\_config.0.username

}

output "cluster\_password" {

value = azurerm\_kubernetes\_cluster.k8s.kube\_config.0.password

}

output "kube\_config" {

value = azurerm\_kubernetes\_cluster.k8s.kube\_config\_raw

sensitive = true

}

output "host" {

value = azurerm\_kubernetes\_cluster.k8s.kube\_config.0.host

}

1. Type **Esc** followed by *:wq* to save and quit the file.

**Deploy and Verify the Kubernetes Cluster is Running**

1. Run terraform fmt to check the formatting of your configuration files.

Your aks.tf, output.tf, providers.tf, and variables.tf files should be listed.

1. Run terraform validate to validate the code in your configuration files.

You should see a message confirming that your configuration is valid.

1. Run terraform plan -out aks.tfplan to create your execution plan.
2. Run terraform apply aks.tfplan to execute your execution plan.

**Note**: It may take a couple of minutes to deploy your resources.

You will see a big block of text appear, which should mean that your cluster deployed successfully. You can scroll up to view the **Apply complete** message in green to confirm.

1. Scrolling down from the **Apply complete** message, you can view the client\_certificate, client\_key, cluster\_ca\_certificate, cluster\_password, and cluster\_username. Lastly, you should see the host address, kube\_config, and resource\_group\_name.
2. Run the following command to move your kube\_config to a different file:

echo "$(terraform output kube\_config)" > ./azurek8s

1. Run cat ./azurek8s to check the file.

You should see EOT at the beginning and end of the file, which will need to be removed.

1. Run vim ./azurek8s to edit the file.
2. Delete the <<EOT at the beginning and the EOT at the end of the file.
3. Type **Esc** followed by *:wq* to save and quit the file.
4. Run export KUBECONFIG=./azurek8s to create your environment variable.
5. Run kubectl get nodes to check if your nodes are running and healthy.

You should see your 3 nodes returned with a STATUS of Ready.