



Azure WorkShop

To start we need to install terraform and azure CLI

```
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$ az version
{
  "azure-cli": "2.69.0",
  "azure-cli-core": "2.69.0",
  "azure-cli-telemetry": "1.1.0",
  "extensions": {}
}
```

```
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$ terraform version
Terraform v1.11.0
on linux_amd64
+ provider registry.terraform.io/hashicorp/azurerm v4.21.1
```

Once installed we proceed to log with the university credentials with `az login`

With that we can start the project

```
code .
```

This is necessary to start our terra environment.

In this environment we will create a `main.tf` file that is the one that contains the whole infrastructure that will be deployed in azure later.

This file has the following structure

```
provider "azurerm" {
```

```
  subscription_id = "xxxx-xxxx-xxxx-xxxx-61a3bf3b5c6f"
```

```
  features {}
```

```
}
```

```
resource "azurerm_resource_group" "example" {
```

```
  name = "example-resources"
```

```
  location = "West Europe"
```

```
}
```

```
resource "azurerm_kubernetes_cluster" "example" {
```

```
  name = "example-aks1"
```

```
  location = azurerm_resource_group.example.location
```

```
  resource_group_name = azurerm_resource_group.example.name
```

```
dns_prefix      = "exampleaks1"
```

```
default_node_pool {
```

```
name      = "default"
```

```
node_count = 1
```

```
vm_size   = "Standard_D2_v2"
```

```
}
```

```
identity {
```

```
type = "SystemAssigned"
```

```
}
```

```
tags = {
```

```
Environment = "Production"
```

```
}
```

```
}
```

```
output "client_certificate" {
```

```
value    = azurerm_kubernetes_cluster.example.kube_config[0].client_certificate
```

```
sensitive = true
```

```
}
```

```
output "kube_config" {
```

```
value = azurerm_kubernetes_cluster.example.kube_config_raw
```

```
sensitive = true
```

```
}
```

And with that we can start the deployment

```
• david@david-Aspire-A315-576:~/Documentos/Plataformas/Terraform$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/azurerm...
- Installing hashicorp/azurerm v4.21.1...
- Installed hashicorp/azurerm v4.21.1 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
david@david-Aspire-A315-576:~/Documentos/Plataformas/Terraform$ terraform validate
Success! The configuration is valid.
```

tdavidvergara (Hace 17 minutos) Lín. 40, Col. 2 (899 seleccionada) Espacios: 4 UTF-8 LF {} Terraform Go Live Prettier

```

• david@david-Aspire-A315-576:~/Documentos/Plataformas/Terraform$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
+ create

Terraform will perform the following actions:

# azure_rm_kubernetes_cluster.example will be created
+ resource "azure_rm_kubernetes_cluster" "example" {
+   current_kubernetes_version = (known after apply)
+   dns_prefix                 = "exampleaks1"
+   fqdn                      = (known after apply)
+   http_application_routing_zone_name = (known after apply)
+   id                        = (known after apply)
+   kube_admin_config         = (sensitive value)
+   kube_admin_config_raw     = (sensitive value)
+   kube_config               = (sensitive value)
+   kube_config_raw           = (sensitive value)
+   kubernetes_version        = (known after apply)
+   location                  = "westeurope"
+   name                      = "example-aks1"
+   node_os_upgrade_channel    = "NodeImage"
+   node_resource_group       = (known after apply)
+   node_resource_group_id    = (known after apply)

```

```

you run "terraform apply" now.
• david@david-Aspire-A315-576:~/Documentos/Plataformas/Terraform$ terraform apply




Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
+ create

Terraform will perform the following actions:

# azure_rm_kubernetes_cluster.example will be created
+ resource "azure_rm_kubernetes_cluster" "example" {
+   current_kubernetes_version = (known after apply)
+   dns_prefix                 = "exampleaks1"
+   fqdn                      = (known after apply)
+   http_application_routing_zone_name = (known after apply)
+   id                        = (known after apply)
+   kube_admin_config         = (sensitive value)
+   kube_admin_config_raw     = (sensitive value)
+   kube_config               = (sensitive value)
+   kube_config_raw           = (sensitive value)
+   kubernetes_version        = (known after apply)
+   location                  = "westeurope"
+   name                      = "example-aks1"
+   node_os_upgrade_channel    = "NodeImage"
+   node_resource_group       = (known after apply)
+   node_resource_group_id    = (known after apply)

```

After all that we see that the infrastructure has been deployed

<input type="checkbox"/> Nombre ↑	Suscripción	Ubicación
<input type="checkbox"/>  example-resources	... Azure for Students	West Europe
<input type="checkbox"/>  MC_example-resources_example-aks1_westeurope	... Azure for Students	West Europe
<input type="checkbox"/>  NetworkWatcherRG	... Azure for Students	West Europe

We now need to connect to our cluster by using `az aks get-credentials --resource-group example-resources --name example-aks1 --overwrite-existing`

Inicio > Grupos de recursos > example-resources >

example-aks1

Servicio de Kubernetes

Buscar

«

+

Crear

✈

Conectar

▶

Inicio

□

Detener

🗑

Eliminar

Información general

Registro de actividad

Control de acceso (IAM)

Etiquetas

Supervisar

Diagnosticar y solucionar problemas

Microsoft Defender for Cloud (versión preliminar)

Análisis de costos

Recursos de Kubernetes

Configuración

Supervisión

Automation

Ayuda

Essentials

Grupo de recursos : example-resources

Estado de energía : Running

Estado de la operac... : Succeeded

Suscripción : [Azure for Students](#)

Ubicación : West Europe

Id. de suscripción : 7f837881-7308-4c8b-84ef-61a3bf3b5c6f

Etiquetas (editar) : Environment : Production

Introducción

Propiedades

Supervisión

Recomendación

Servicios de Kubernetes

Tipo de cifrado

Cifrado en reposo con una clave administrada por la plataforma

Grupos de nodos virtuales

No habilitado

Grupos de nodos

Conectarse a example-aks1

Cloud Shell

CLI de Azure

Comando de ejecución

Conéctese al clúster mediante las herramientas de línea de comandos para interactuar directamente con el clúster mediante kubectl, la herramienta de línea de comandos para Kubernetes. Kubectl está disponible en Azure Cloud Shell de forma predeterminada y también se puede instalar localmente.

Establecer contexto de clúster

1. Abrir Cloud Shell

Ejecuta automáticamente los siguientes comandos

Establecer la suscripción del clúster

az account set --subscription 7f837881-7308-4c8b-84ef-61a3bf3b5c6f

Descargar credenciales de clúster

az aks get-credentials --resource-group example-resources --name example-aks1 --...

Comandos de muestra

Una vez que haya ejecutado el comando anterior para conectarse al clúster, puede ejecutar los comandos de kubectl. Aquí tiene algunos ejemplos de comandos útiles que puede probar.

Cerrar

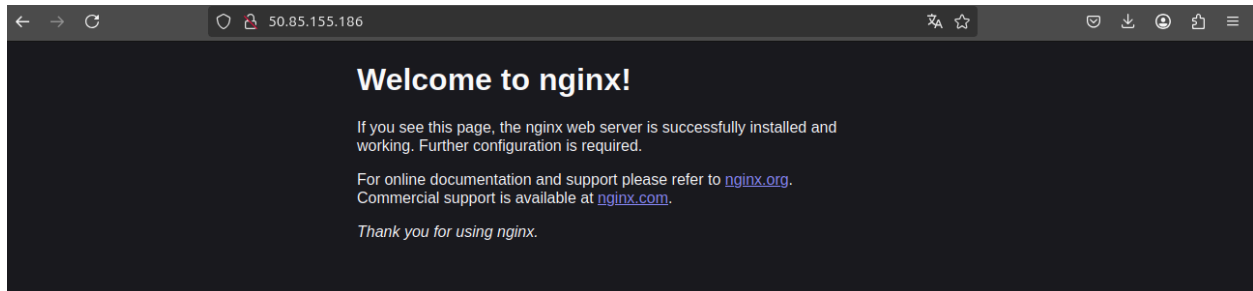
Now we need to proceed to switch contexts

```
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$ az aks get-credentials --resource-group example-resources --name example-aks1 --overwrite-existing
Merged "example-aks1" as current context in /home/david/.kube/config
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$ kubectl --context
error: flag needs an argument: --context
See 'kubectl --help' for usage.
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$ kubectl config current-context
example-aks1
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$ kubectl config get-context
error: unknown command "get-context"
See 'kubectl config -h' for help and examples
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$ kubectl config get-contexts
CURRENT  NAME      CLUSTER      AUTHINFO      NAMESPACE
*         example-aks1  example-aks1  clusterUser_example-resources_example-aks1  default
minikube  minikube    minikube      minikube      default
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$ kubectl config use-context example-aks1
Switched to context "example-aks1".
```

And we now need to add some file that will allow us to expose the service with the pod

```
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$ ^C
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$ nano nginx-pod.yaml
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$ kubectl get svc nginx-service
Error from server (NotFound): services "nginx-service" not found
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$ kubectl apply -f nginx-pod.yaml
pod/nginx-pod created
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$ nano nginx-service.yaml
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$ kubectl apply -f nginx-service.yaml
service/nginx-service created
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$ kubectl get svc nginx-service
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
nginx-service  LoadBalancer  10.0.89.42      <pending>        80:30941/TCP  10s
david@david-Aspire-A315-57G:~/Documentos/Plataformas/Terraform$ kubectl get svc nginx-service
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
nginx-service  LoadBalancer  10.0.89.42      50.85.155.186    80:30941/TCP  31s
```

Here we can verify that the service is working



Now i need to download Lens, go to [add kubeconfig by pasting](#)

Use this distribution

```
apiVersion: v1
kind: Config
clusters:
- name: "WSL Cluster"
  cluster:
    server: http://localhost:8001
users:
- name: nouser
contexts:
- name: "WSL Cluster"
  context:
    cluster: "WSL Cluster"
    user: nouser
current-context: "WSL Cluster"
preferences: {}
```

And the len will connect to our cluster

