**Actividad Terraform - Azure – Lens**
**Plataformas II**
**Juan Yustes A00380718**

Primero instalamos el azure con el terraform en el WSL.

```
yus@Yus:~$ az version
{
  "azure-cli": "2.69.0",
  "azure-cli-core": "2.69.0",
  "azure-cli-telemetry": "1.1.0",
  "extensions": {}
}
yus@Yus:~$ terraform version
Terraform v1.11.0
on linux_amd64
yus@Yus:~$
```

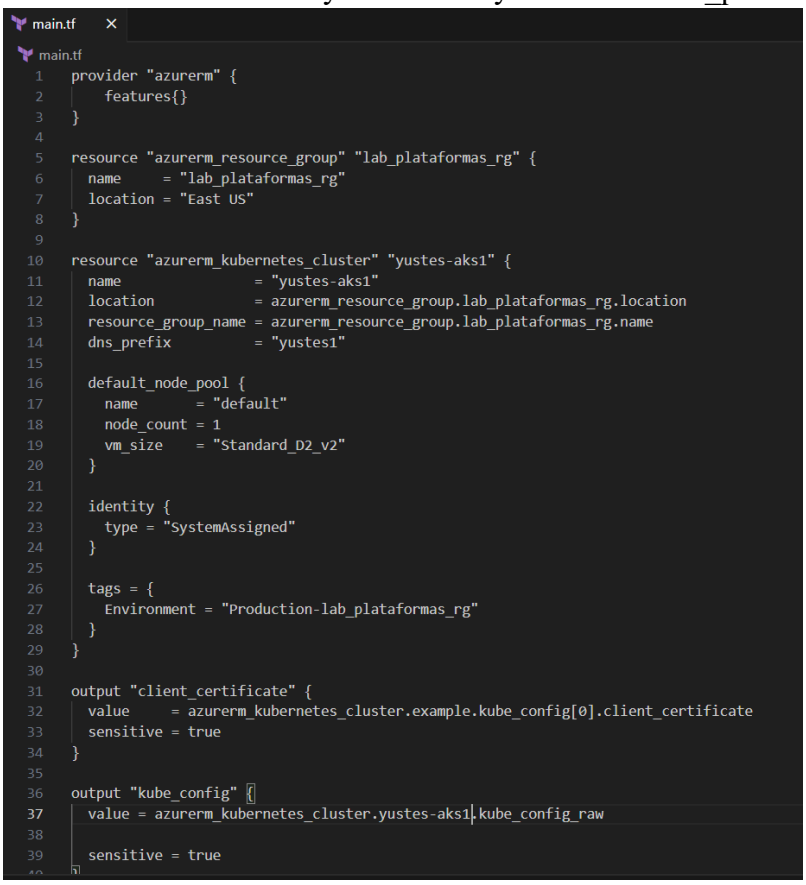
Ya con el AZ instalado, ingresamos con las credenciales estudiantiles.

```
yus@Yus:~$ az login
A web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue the login in the web browser. If no
web browser is available or if the web browser fails to open, use device code flow with `az login --use-device-code`.
gio: https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize?client_id=04b07795-8ddb-461a-bbee-02f9e1bf7b46&response_type=code&redirect_uri=ht
tp%3A%2F%2Flocalhost%3A40403&scope=https%3A%2F%2Fmanagement.core.windows.net%2F%2F.default+offline_access+openid+profile&state=GuAmwHlpOLxVbBdz&code_challen
ge=sByg1oLOF3H3T14N9x6lo_zn5-bBbdn8uqjqv6spwLA&code_challenge_method=S256&nonce=f0dd32c4eec765f49b7243361abe24a76a6efc040ca45a6bd7f1854acc7ad62a&client_info
=1&claims=%7B%22access_token%22%3A+%7B%22xms_cc%22%3A+%7B%22values%22%3A+%5B%22CP1%22%5D%7D%7D%7D&prompt=select_account: Operation not supported

Retrieving tenants and subscriptions for the selection...

[Tenant and subscription selection]

No     Subscription name    Subscription ID                       Tenant
-----  -------------------  ------------------------------------  ----------------
```

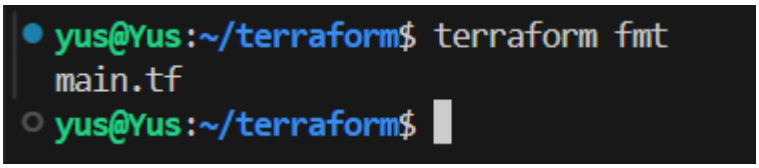
Y ahora ponemos como predeterminado la cuenta asociada de AZ, luego creamos una carpeta para el terraform y ahí codificar.

```
yus@Yus:~$ mkdir terraform
yus@Yus:~$ cd terraform/
yus@Yus:~/terraform$ code .
Installing VS Code Server for Linux x64 (e54c774e0add60467559eb0d1e229c6452cf8447)
Downloading: 100%
Unpacking: 100%
Unpacked 2064 files and folders to /home/yus/.vscode-server/bin/e54c774e0add60467559eb0d1e229c6452cf8447.
Looking for compatibility check script at /home/yus/.vscode-server/bin/e54c774e0add60467559eb0d1e229c6452cf8447/bin/helpers/check-requirements.sh
Running compatibility check script
Compatibility check successful (0)
yus@Yus:~/terraform$
```

Dentro de la carpeta de terraform creamos el main.tf que va a ser el archivo con el cual vamos a montar la infraestructura en azure, personalizándolo según las necesidades, en este caso se llamó el cluster “yustes-aks1” y el recurso “lab_plataformas_rg”




Ya con el main, hecho usamos fmt para el formato.



E inicializamos el .tf

```
yus@Yus:~/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!
```

Hacemos terraform plan para revisar el estado y vemos que hay un error ya que hace falta agregar la suscripción.

```
yus@Yus:~/terraform$ terraform plan

Planning failed. Terraform encountered an error while generating this plan.


Error: `subscription_id` is a required provider property when performing a plan/apply operation

  with provider["registry.terraform.io/hashicorp/azurerm"],
  on main.tf line 1, in provider "azurerm":
   1: provider "azurerm" {
```

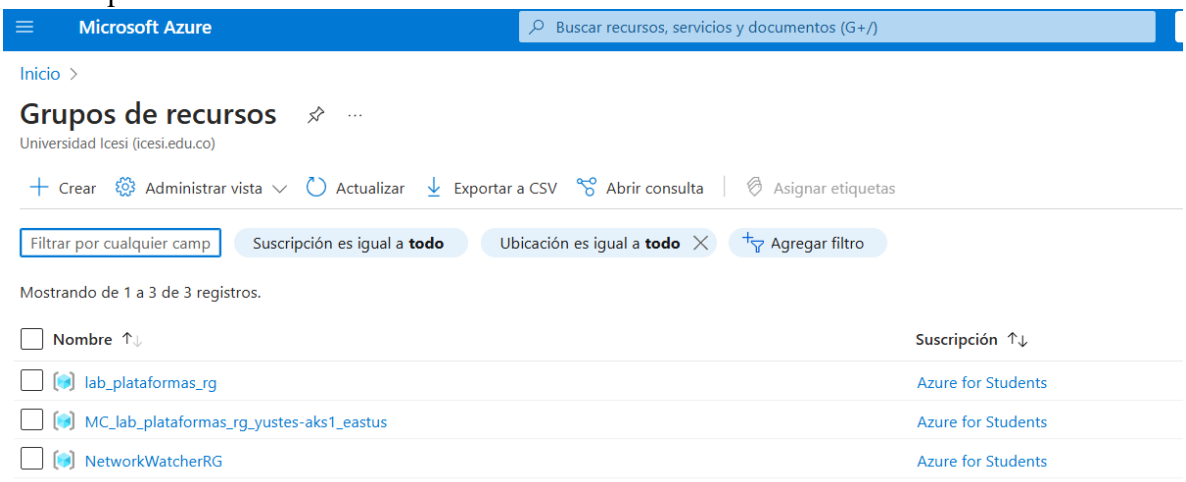
```
provider "azurerm" {
  features {}
  subscription_id = "3f
}
```

Ahora corroboramos de que funcione y ahora sí damos apply para que ejecute.

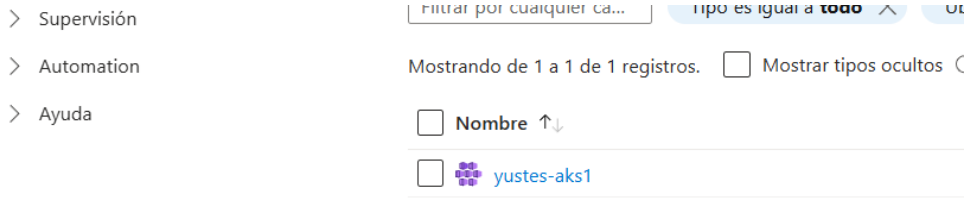
```
yus@Yus:~/terraform$ terraform plan

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:

  # azurerm_kubernetes_cluster.yustes-aks1 will be created
  + resource "azurerm_kubernetes_cluster" "yustes-aks1" {
      + current_kubernetes_version          = (known after apply)
      + dns_prefix                          = "yustes1"
      + 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)
```

```
yus@Yus:~/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:

  # azurerm_kubernetes_cluster.yustes-aks1 will be created
  + resource "azurerm_kubernetes_cluster" "yustes-aks1" {
      + current_kubernetes_version          = (known after apply)
      + dns_prefix                          = "yustes1"
      + fqdn                                = (known after apply)
```

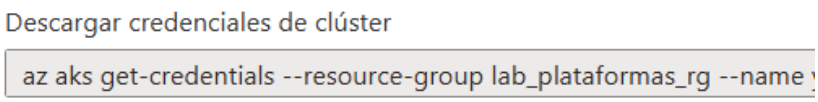
Vemos que se creó de forma efectiva el recurso en Azure.




Entramos al recurso, vemos que está nuestro cluster y entramos para conectarnos al cluster.




Usamos el comando dado para conectarnos a nuestro propio cluster desde el WSL.




```
yus@Yus:~/terraform$ az aks get-credentials --resource-group lab_plataformas_rg --nam
Merged "yustes-aks1" as current context in /home/yus/.kube/config
yus@Yus:~/terraform$
```

Ya con el cluster conectado, cambiamos el contexto del kubectl para que se asocie al cluster.

```
yus@Yus:~/terraform$ kubectl config current-context
yustes-aks1
yus@Yus:~/terraform$ kubectl config get-contexts
CURRENT   NAME          CLUSTER       AUTHINFO                                NAMESPACE
*         yustes-aks1   yustes-aks1   clusterUser_lab_plataformas_rg_yustes-aks1
```

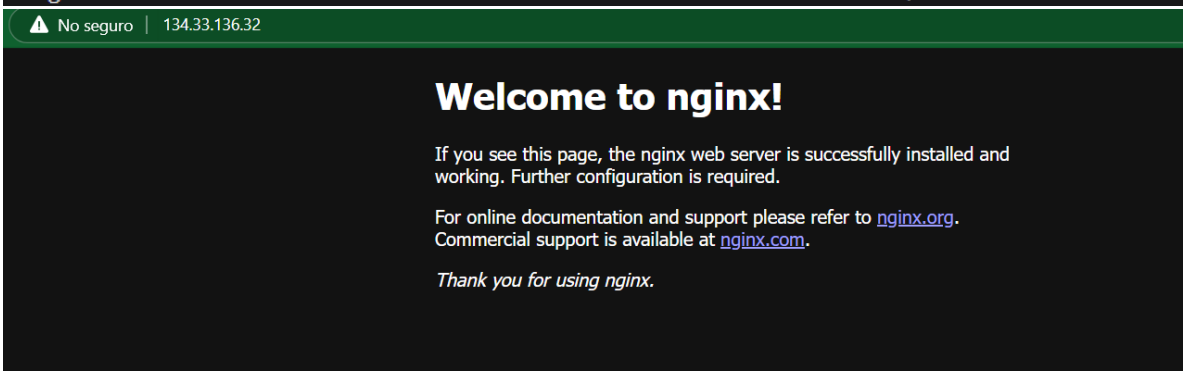
```
yus@Yus:~/terraform$ kubectl config use-context yustes-aks1
Switched to context "yustes-aks1".
```

Ya en el cluster creamos los .yaml necesarios del pod y del servicio para poder exponer el servicio con el pod de nginx y damos apply para que se ejecute.

```
yus@Yus:~/terraform$ nano nginx-pod.yaml
yus@Yus:~/terraform$ kubectl apply -f nginx-pod.yaml
pod/nginx-pod created
yus@Yus:~/terraform$ nano nginx-service.yaml
yus@Yus:~/terraform$ kubectl apply -f nginx-service.yaml
service/nginx-service created
```

Vemos con que IP se expuso y procedemos a revisar que se vea desde esa ip.

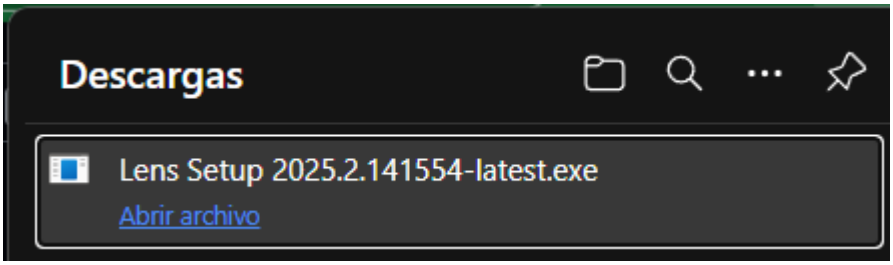
```
yus@Yus:~/terraform$ kubectl get svc nginx-service
NAME            TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)        AGE
nginx-service   LoadBalancer   10.0.105.127   134.33.136.32   80:30122/TCP   5m31s
```



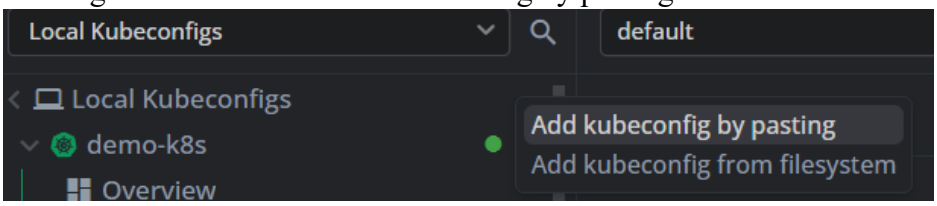

Como ya tenemos el servicio con el pod de Nginx corriendo, ahora usamos el siguiente comando para exponer el acceso al cluster desde la maquina Windows, pues como estábamos accediendo desde WSL al cluster, no podríamos conectar la Windows al mismo.

```
yus@Yus:~/terraform$ kubectl proxy --port 8001 --reject-paths "^/api/./pods/./attach"
Starting to serve on 127.0.0.1:8001
```

Ahora descargamos Lens en su versión de Windows.



Y al ingresar le damos a “add kubeconfig by pasting”.




Y usamos esta declaración para poder acceder al cluster expuesto desde WSL.

```yaml
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: {}
```

Y automáticamente el Lens se conecta a nuestro cluster para monitorear.