

Sebastian Lopez Garcia A00377582

Introduction to Terraform Lab with Kubernetes

In this lab, we'll work with **Terraform** for infrastructure-as-code management, using **kubectl** to interface with a Kubernetes cluster. The main objective will be to deploy a **pod with Nginx** and visualize it through **Lens Desktop**, a graphical tool that facilitates the administration of Kubernetes clusters.

az-cli:

```
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
```

```
sudo apt-get update
```

```
sudo apt-get install apt-transport-https ca-certificates curl gnupg lsb-release
```

```
sudo mkdir -p /etc/apt/keyrings
```

```
curl -sLS https://packages.microsoft.com/keys/microsoft.asc |
```

```
  gpg --dearmor | sudo tee /etc/apt/keyrings/microsoft.gpg > /dev/null
```

```
sudo chmod go+r /etc/apt/keyrings/microsoft.gpg
```

```
AZ_DIST=$(lsb_release -cs)
```

```
echo "Types: deb
```

```
URIs: https://packages.microsoft.com/repos/azure-cli/
```

```
Suites: ${AZ_DIST}
```

```
Components: main
```

```
Architectures: $(dpkg --print-architecture)
```

```
Signed-by: /etc/apt/keyrings/microsoft.gpg" | sudo tee /etc/apt/sources.list.d/azure-cli.sources
```

```
sudo apt-get update
```

```
sudo apt-get install azure-cli
```

It's critical to authenticate to **Microsoft Azure** to manage resources from Terraform and Kubernetes. We'll use the az login command from the **Azure CLI** to authenticate our session and gain access to the subscription where we'll deploy our environment.

After these steps, we would already have Az-cli:

```
Ubuntu 22.04.5 LTS
papita@papita:~$ az version
{
  "azure-cli": "2.69.0",
  "azure-cli-core": "2.69.0",
  "azure-cli-telemetry": "1.1.0",
  "extensions": {}
}
papita@papita:~$ 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=84b07795-8ddb-461a-bbee-02f9e1bf7b46&response_type=code&redirect_uri=https://localhost%3A2917&scope=https%3A%2F%2Fmanagement.core.windows.net%2F%2F.default+offline_access+openid+profile&state=SkH0roVlUwzTbxWt6code_challenge=EPXqKrtWVLAGn6NrBPHW_UN5fLiRtxZ1YrewTyyttA&code_challenge_method=S256&nonce=5abdb0bd69945000d0dd74872d65898b6690872bf100352ce13e711f05e791e56client_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
---  -
[1] * Azure for Students  [redacted]  Universidad Icesi

The default is marked with an *; the default tenant is 'Universidad Icesi' and subscription is 'Azure for Students' (45cf82e3-a6b2-4aa8-a135-26393913d494).

Select a subscription and tenant (Type a number or Enter for no changes): 1

Tenant: Universidad Icesi
Subscription: Azure for Students

[Announcements]
With the new Azure CLI login experience, you can select the subscription you want to use more easily. Learn more about it and its configuration at https://go.microsoft.com/fwlink/?linkid=2271236

If you encounter any problem, please open an issue at https://aka.ms/azclibug

[Warning] The login output has been updated. Please be aware that it no longer displays the full list of available subscriptions by default.

papita@papita:~$ az account set --subscription [redacted]
papita@papita:~$
```

Azure Provider

```
provider "azurerm" {
  features {}
}
```

This block defines the Azure Resource Manager provider ('azurerm'). The 'features {}' block is required but may be empty.

Resource Group

```
resource "azurerm_resource_group" "labTerraform" {
  name      = "labs_plataformas_rg"
  location  = "East US"
}
```

This block creates a resource group in Azure called 'labs_plataformas_rg' in the 'East US' location.

Cluster de Kubernetes

```
resource "azurerm_kubernetes_cluster" "sl-aks1" {
  name                = "sl-aks1"
  location            = azurerm_resource_group.labTerraform.location
  resource_group_name = azurerm_resource_group.labTerraform.name
  dns_prefix          = "sl1"

  default_node_pool {
    name       = "default"
    node_count = 1
    vm_size    = "Standard_D2_v2"
  }

  identity {
    type = "SystemAssigned"
  }

  tags = {
    Environment = "testings"
  }
}
```

This block creates a Kubernetes cluster on Azure (AKS) named 'sl-aks1'. Here are the details:

- 'location' and 'resource_group_name' refer to the previously created resource group.
- 'dns_prefix' is the DNS prefix for the cluster.
- 'default_node_pool' defines the default node pool with a single node ('node_count = 1') and a VM size 'Standard_D2_v2'.
- 'identity' states that the cluster will use a system-managed identity ('SystemAssigned').
- 'tags' adds a tag to identify the environment as "testings".

Outputs

```

output "client_certificate" {
  value      = azurerm_kubernetes_cluster.sl-aks1.kube_config[0].client_certificate
  sensitive = true
}

output "kube_config" {
  value = azurerm_kubernetes_cluster.sl-aks1.kube_config_raw
  sensitive = true
}

```

These output blocks expose sensitive information from the Kubernetes cluster:

- 'client_certificate' exposes the cluster client certificate.
- 'kube_config' exposes the complete configuration of the cluster in RAW format.

Both values are marked as 'sensitive = true' to prevent them from being displayed in Terraform's output logs.

The next step is to prepare Terraform to deploy our infrastructure. First, we use the **terraform init** command, which is responsible for initializing the work environment. This includes offloading the necessary providers and configuring the backend where the state of the infrastructure will be stored. Without this step, we wouldn't be able to execute any actions with Terraform.

```

6 resource "azurerm_resource_group" "labTerraform" {
  name = "labTerraform"
}

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
• papita@papita:~/terraform$ terraform fmt
• papita@papita:~/terraform$ terraform init
  Initializing the backend...
  Initializing provider plugins...
    - Reusing previous version of hashicorp/azurerm from the dependency lock file
    - Installing hashicorp/azurerm v4.21.1...
    - Installed hashicorp/azurerm v4.21.1 (signed by HashiCorp)
  Terraform has made some changes to the provider dependency selections recorded
  in the .terraform.lock.hcl file. Review those changes and commit them to your
  version control system if they represent changes you intended to make.

  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.

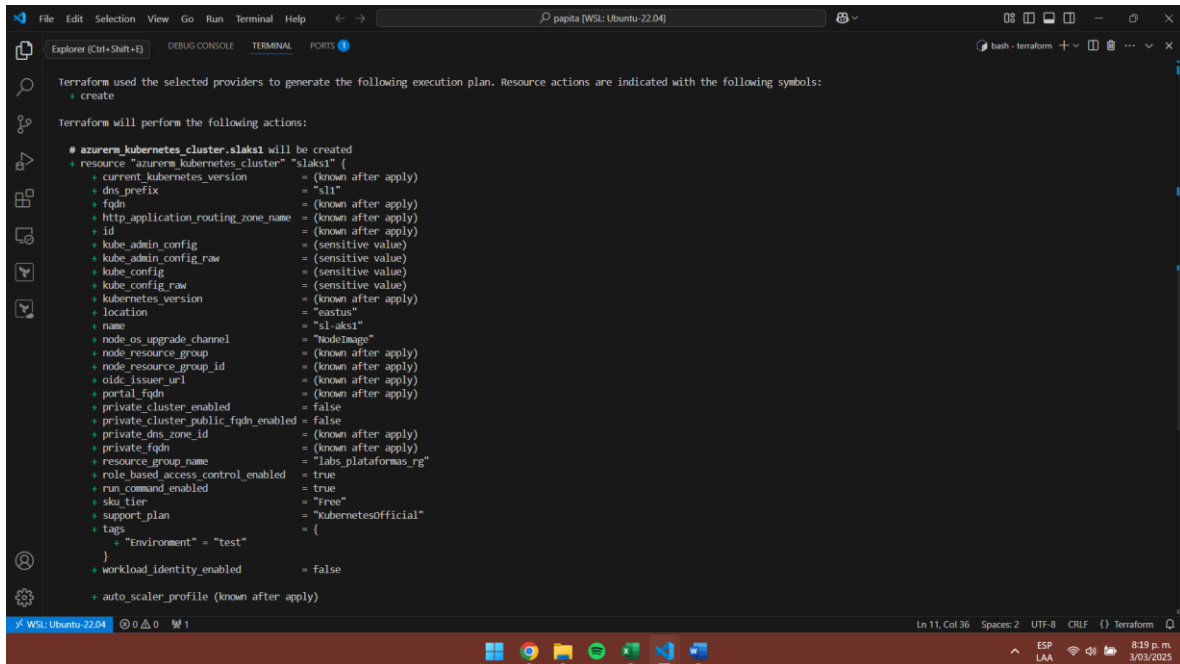
  If you ever set or change modules or backend configuration for Terraform,
  rerun this command to reinitialize your working directory. If you forget, other
  commands will detect it and remind you to do so if necessary.
• papita@papita:~/terraform$ terraform plan

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.slaks1 will be created
+ resource "azurerm_kubernetes_cluster" "slaks1" {
  + current_kubernetes_version = (known after apply)
  + dns_prefix                 = "sl1"
}

```



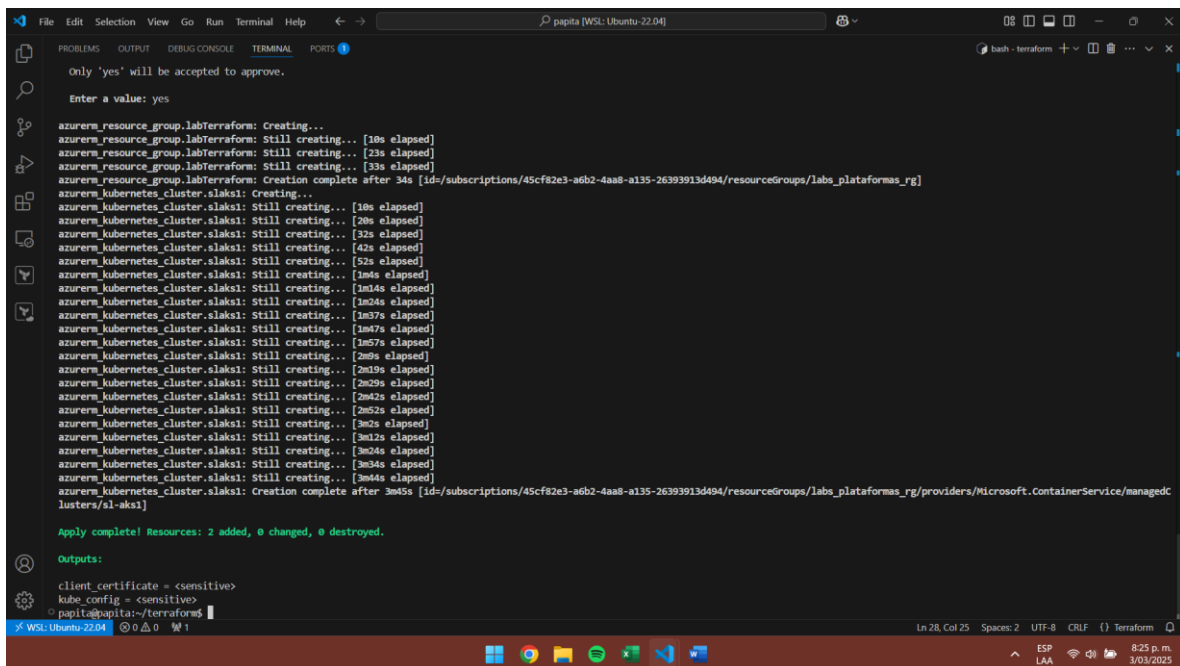
The screenshot shows a VS Code terminal window with the Terraform execution plan output. The terminal title is 'papita [WSL: Ubuntu-22.04]'. The output indicates that Terraform will create an 'azure_rm_kubernetes_cluster.slaks1' resource. The plan lists various attributes for this resource, including 'current_kubernetes_version', 'dns_prefix', 'fqdn', 'http_application_routing_zone_name', 'id', 'kube_admin_config', 'kube_admin_config_raw', 'kube_config', 'kube_config_raw', 'kubernetes_version', 'location', 'name', 'node_os_upgrade_channel', 'node_resource_group', 'node_resource_group_id', 'oidc_issuer_url', 'portal_fqdn', 'private_cluster_enabled', 'private_cluster_public_fqdn_enabled', 'private_dns_zone_id', 'private_fqdn', 'resource_group_name', 'role_based_access_control_enabled', 'run_command_enabled', 'sku_tier', 'support_plan', 'tags', and 'workload_identity_enabled'. The 'tags' block is set to 'Environment = test'. The 'auto_scaler_profile' is also listed as 'known after 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.slaks1 will be created
+ resource "azure_rm_kubernetes_cluster" "slaks1" {
  + current_kubernetes_version = (known after apply)
  + dns_prefix                 = "sl1"
  + 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                  = "eastus"
  + name                      = "sl-aks1"
  + node_os_upgrade_channel   = "nodeImage"
  + node_resource_group       = (known after apply)
  + node_resource_group_id    = (known after apply)
  + oidc_issuer_url           = (known after apply)
  + portal_fqdn               = (known after apply)
  + private_cluster_enabled   = false
  + private_cluster_public_fqdn_enabled = false
  + private_dns_zone_id       = (known after apply)
  + private_fqdn              = (known after apply)
  + resource_group_name       = "labs_plataformas_rg"
  + role_based_access_control_enabled = true
  + run_command_enabled       = true
  + sku_tier                   = "free"
  + support_plan               = "kubernetesofficial"
  + tags                      = {
    + "Environment" = "test"
  }
  + workload_identity_enabled = false
  + auto_scaler_profile      = (known after apply)
}
```

Finally, we apply the changes with **terraform apply**, which is responsible for creating or modifying the infrastructure as defined in the Terraform files. This command generates an execution plan, and upon confirmation, proceeds to deploy the resources to Azure. With this, our infrastructure is ready for use.



The screenshot shows a VS Code terminal window with the Terraform apply output. The terminal title is 'papita [WSL: Ubuntu-22.04]'. The output shows the 'terraform apply' command being executed. It prompts for confirmation, and the user enters 'yes'. The output shows the creation of the 'azure_rm_kubernetes_cluster.slaks1' resource. The status of the resource is 'Still creating...' with a progress bar. The output shows the resource being created successfully. The final output shows the resource being created successfully.

```
Only 'yes' will be accepted to approve.

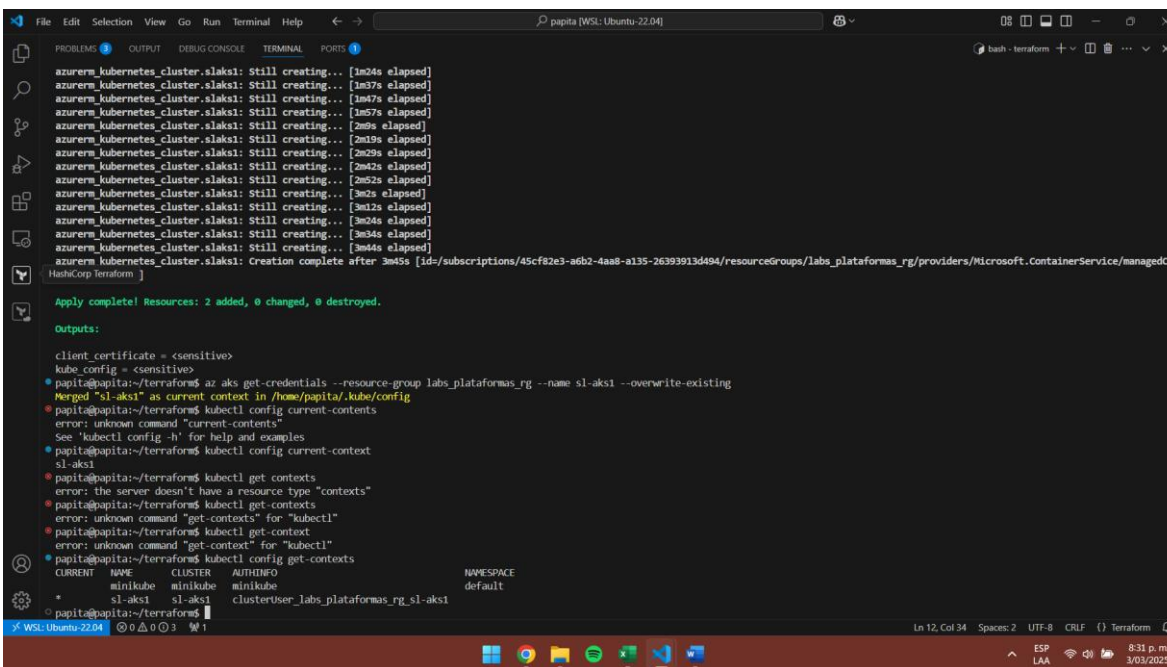
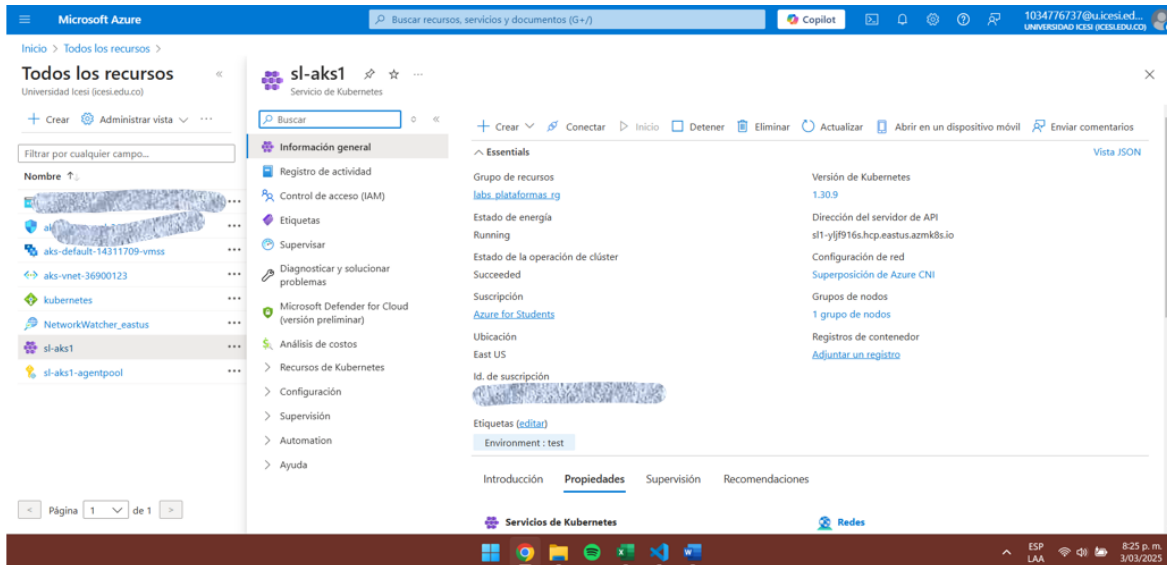
Enter a value: yes

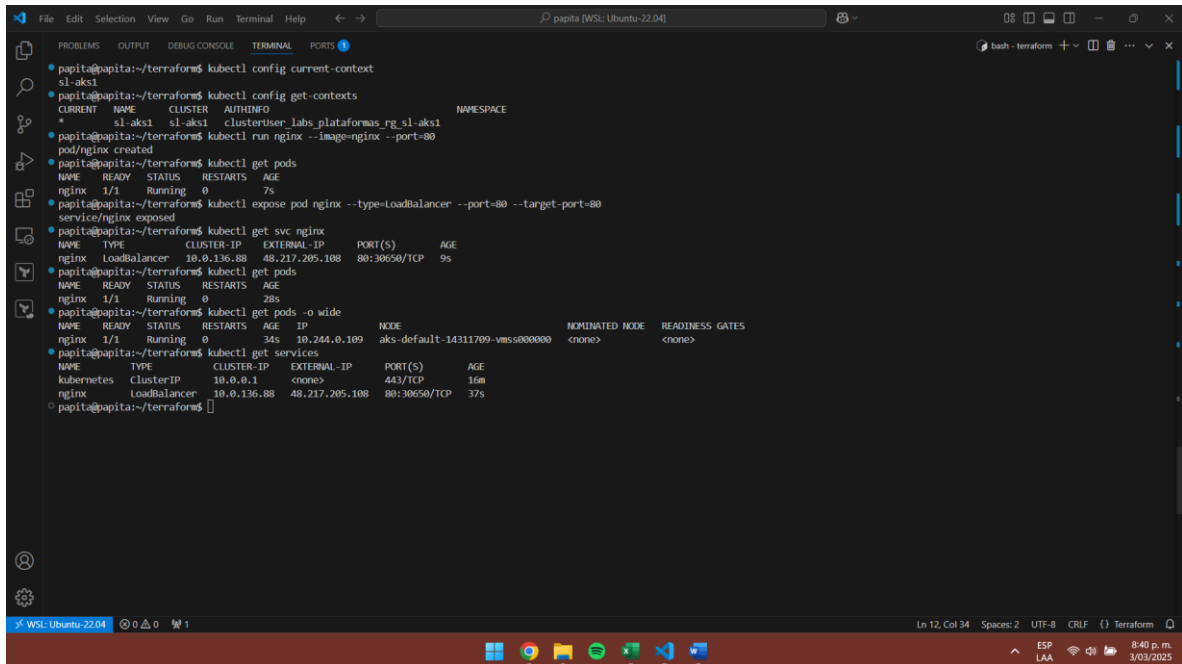
azure_rm_resource_group.labTerraform: Creating...
azure_rm_resource_group.labTerraform: Still creating... [10s elapsed]
azure_rm_resource_group.labTerraform: Still creating... [23s elapsed]
azure_rm_resource_group.labTerraform: Still creating... [33s elapsed]
azure_rm_resource_group.labTerraform: Creation complete after 34s [id=/subscriptions/45cf82e3-a6b2-4aa8-a135-26393913d494/resourceGroups/labs_plataformas_rg]
azure_rm_kubernetes_cluster.slaks1: Creating...
azure_rm_kubernetes_cluster.slaks1: Still creating... [10s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [20s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [32s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [42s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [52s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [1m4s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [1m44s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [1m24s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [1m37s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [1m47s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [1m57s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [2m5s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [2m19s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [2m39s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [2m42s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [2m52s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [3m2s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [3m12s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [3m24s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [3m34s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still creating... [3m44s elapsed]
azure_rm_kubernetes_cluster.slaks1: Creation complete after 3m45s [id=/subscriptions/45cf82e3-a6b2-4aa8-a135-26393913d494/resourceGroups/labs_plataformas_rg/providers/Microsoft.ContainerService/managedClusters/sl-aks1]

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

Outputs:
client_certificate = <sensitive>
kube_config = <sensitive>
papita@papita:~/terraform$
```

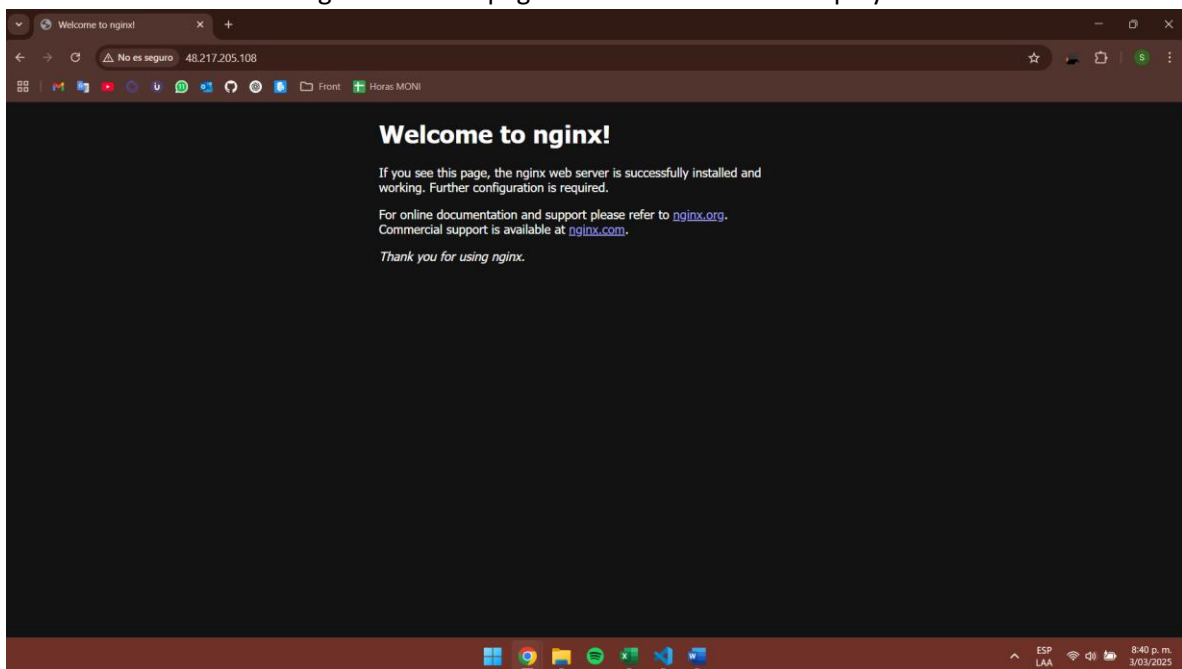
Once we've run **terraform apply**, we can verify that our pod with Nginx is running inside Azure. To do this, we use **kubect**, which allows us to interact with the Kubernetes cluster. With the **kubect** **get pods** command, we can see the status of the pods and make sure they are in the **Running** state. We can also inspect the services with **kubect** **get services**, which will show us if the external IP has been assigned correctly.





```
papita@papita:~/terraform$ kubectl config current-context
sl-aks1
papita@papita:~/terraform$ kubectl config get-contexts
CURRENT   NAME      CLUSTER   AUTHINFO   NAMESPACE
*          sl-aks1   sl-aks1   clusterUser_labo_plataformas_rg_sl-aks1
papita@papita:~/terraform$ kubectl run nginx --image=nginx --port=80
pod/nginx created
papita@papita:~/terraform$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
nginx     1/1     Running   0           7s
papita@papita:~/terraform$ kubectl expose pod nginx --type=loadbalancer --port=80 --target-port=80
service/nginx exposed
papita@papita:~/terraform$ kubectl get svc nginx
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
nginx     loadbalancer 10.0.136.88   48.217.205.108 80:30650/TCP     9s
papita@papita:~/terraform$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
nginx     1/1     Running   0           28s
papita@papita:~/terraform$ kubectl get pods -o wide
NAME      READY   STATUS    RESTARTS   AGE   IP              NODE               NOMINATED NODE   READINESS GATES
nginx     1/1     Running   0           34s   10.244.0.109    aks-default-14311709-vmss000000    <none>            <none>
papita@papita:~/terraform$ kubectl get services
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes  ClusterIP   10.0.0.1     <none>        443/TCP          16m
nginx      loadbalancer 10.0.136.88   48.217.205.108 80:30650/TCP     37s
papita@papita:~/terraform$
```

To access the website served by Nginx, we use the public IP address assigned to the service. This will allow us to view the Nginx welcome page and confirm that the deployment was successful.



Finally, we can use **Lens Desktop** to visually manage our cluster. Lens provides us with a graphical interface where we can see the pods, services, and other resources deployed in Kubernetes. To connect, we add the cluster to Lens using the **kubeconfig** file, which will allow us to monitor the state of the system, review logs, and manage resources in a more intuitive way.

The image is a composite of two screenshots. The top screenshot is a web browser window displaying a tutorial from patrickdap.com. The tutorial explains how to connect to a Kubernetes cluster using OpenLens. It mentions that with port forward enabled and 'exec' capabilities on, you're providing God mode capabilities to your cluster. It then provides a command to run a proxy session on port 8001, overriding the default --reject-paths parameter to allow access to the exec feature. The command is: `kubectl proxy --port 8001 --reject-paths "/^/api/./pods/./attach"`. The bottom screenshot shows the OpenLens desktop application interface. The left sidebar contains a 'Navigator' with sections for 'KUBERNETES CLUSTERS' (Local Kubeconfigs, demo-k8s, sl-aks1) and 'Overview' (Applications, Nodes, Workloads, Config, T1 Network, Storage, Namespaces, Events, Helm, Access Control, Custom Resources). The main panel shows the 'Cluster Overview' for 'sl-aks1'. It displays three circular progress indicators for CPU, Memory, and Pods usage. Below these, there's a 'Warnings' section showing a warning about an IMDS query failure. The bottom status bar indicates 'You're using Lens Personal (for individuals or companies with < \$10M annual revenue or funding)' and 'sl-aks1 (v1.30.9)'.

patrickdap.com/post/openlens-wsl/

Coming from your machine, you're connecting to it.

With port forward enabled and `exec` capabilities on, you're providing God mode capabilities to your cluster to anyone who can find this URL.

So what we have to do instead is not only create the proxy but also allow access to the `exec` feature. To do that, we can use the following command:

```
kubectl proxy --port 8001 --reject-paths "/^/api/./pods/./attach"
```

This will start a proxy session on port `8001` with support for connecting to the running container via `exec` - which is normally in the `--reject-paths` parameter by default but we override to not include it.

Now let's access the cluster from OpenLens.

Connecting to the proxy from OpenLens

Add a new cluster in OpenLens by going to `File > Add Cluster...` (or if it's your first time

UPGRADE

Cluster Overview - sl-aks1

Home

KUBERNETES CLUSTERS

- Local Kubeconfigs
- demo-k8s
- sl-aks1

Overview

- Applications
- Nodes
- Workloads
- Config
- T1 Network
- Storage
- Namespaces
- Events
- Helm
- Access Control
- Custom Resources

CPU

Usage: 0.15

Allocatable Capacity: 1.90

Capacity: 2.00

Memory

Allocatable Capacity: 4.9GiB

Capacity: 6.8GiB

Pods

Usage: 1

Allocatable Capacity: 250

Capacity: 250

Warnings: 1

Message	Object	Type	Age
IMDS query failed, exit code: 28 Connection timed out after 24 second...	aks-default-14311709-vmss000000	Event	29m

Terminal

You're using Lens Personal (for individuals or companies with < \$10M annual revenue or funding)

sl-aks1 (v1.30.9)


```
File Edit Selection View Go Run Terminal Help
capita [WSL: Ubuntu-22.04]

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
} -> null
- idle_timeout_in_minutes = 0 -> null
- managed_outbound_ip_count = 1 -> null
- managed_outbound_ip_prefix_count = 0 -> null
- outbound_ip_address_ids = [] -> null
- outbound_ip_prefix_ids = [] -> null
- outbound_ports_allocated = 0 -> null
}
}
}
# azure_rm_resource_group.labterraform will be destroyed
- resource "azure_rm_resource_group" "labterraform" {
  - id = "/subscriptions/45cf82e3-a6b2-4aa8-a135-26393913d494/resourceGroups/labs_plataformas_rg" -> null
  - location = "eastus" -> null
  - name = "labs_plataformas_rg" -> null
  - tags = {} -> null
  # (1 unchanged attribute hidden)
}

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

Changes to Outputs:
- client_certificate = (sensitive value) -> null
- kube_config = (sensitive value) -> null

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

azure_rm_kubernetes_cluster.slaks1: Destroying... [id=/subscriptions/45cf82e3-a6b2-4aa8-a135-26393913d494/resourceGroups/labs_plataformas_rg/providers/Microsoft.ContainerService/managedClusters/sl-aks1]
azure_rm_kubernetes_cluster.slaks1: Still destroying... [id=/subscriptions/45cf82e3-a6b2-4aa8-a135-26393913d494/resourceGroups/labs_plataformas_rg/providers/Microsoft.ContainerService/managedClusters/sl-aks1, 18s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still destroying... [id=/subscriptions/45cf82e3-a6b2-4aa8-a135-26393913d494/resourceGroups/labs_plataformas_rg/providers/Microsoft.ContainerService/managedClusters/sl-aks1, 23s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still destroying... [id=/subscriptions/45cf82e3-a6b2-4aa8-a135-26393913d494/resourceGroups/labs_plataformas_rg/providers/Microsoft.ContainerService/managedClusters/sl-aks1, 33s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still destroying... [id=/subscriptions/45cf82e3-a6b2-4aa8-a135-26393913d494/resourceGroups/labs_plataformas_rg/providers/Microsoft.ContainerService/managedClusters/sl-aks1, 43s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still destroying... [id=/subscriptions/45cf82e3-a6b2-4aa8-a135-26393913d494/resourceGroups/labs_plataformas_rg/providers/Microsoft.ContainerService/managedClusters/sl-aks1, 57s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still destroying... [id=/subscriptions/45cf82e3-a6b2-4aa8-a135-26393913d494/resourceGroups/labs_plataformas_rg/providers/Microsoft.ContainerService/managedClusters/sl-aks1, 1m7s elapsed]
azure_rm_kubernetes_cluster.slaks1: Still destroying... [id=/subscriptions/45cf82e3-a6b2-4aa8-a135-26393913d494/resourceGroups/labs_plataformas_rg/providers/Microsoft.ContainerService/managedClusters/sl-aks1, 1m17s elapsed]

WSL: Ubuntu-22.04 0 0 0 W 1
Ln 14, Col 16 Spaces: 2 UTF-8 LF ( ) YAML
ESP LAA 8:59 p. m. 3/3/2025
```