Proyecto DevOps: CI/CD con Docker, Kubernetes (AKS), Helm y Terraform

Este repositorio contiene los pasos, archivos y buenas prácticas seguidas para desplegar una aplicación sí es satisfactorio el escaneo de calidad en el código de SonarQube, usando Docker, AKS, Helm, Terraform, integrando DevOps y automatización. Primer filtro para empleo como DevOps.

Índice

Estructura del Proyecto

```
docker-init-demos/
│
├── environment/
│   ├── deployment.yml          # Manifiesto de Deployment de Kubernetes
│   ├── ingress.yml             # Manifiesto de Ingress para exponer la app
│   └── service.yml             # Manifiesto de Service para el
ClusterIP/LoadBalancer
│
├── helm/
│   └── docker-getting-started/ # Helm Chart para desplegar la app
│
├── terraform-aks/              # Código Terraform para levantar AKS
│
├── node/
│   └── sample/
│       └── Dockerfile          # Dockerfile de la aplicación
│
└── README.md                   # (Este archivo)
```

Requisitos Previos

- Azure CLI (`az`)
- Sonarqube
- Terraform (`terraform`)
- Docker
- kubectl
- Helm
- Cuenta en DockerHub
- Cuenta en Azure y permisos suficientes para crear recursos

---

1. Preparación del Entorno

Instalar Terraform en WSL/Ubuntu:

```
sudo apt-get update && sudo apt-get install -y gnupg software-properties-common curl
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
sudo apt-get update && sudo apt-get install terraform
```

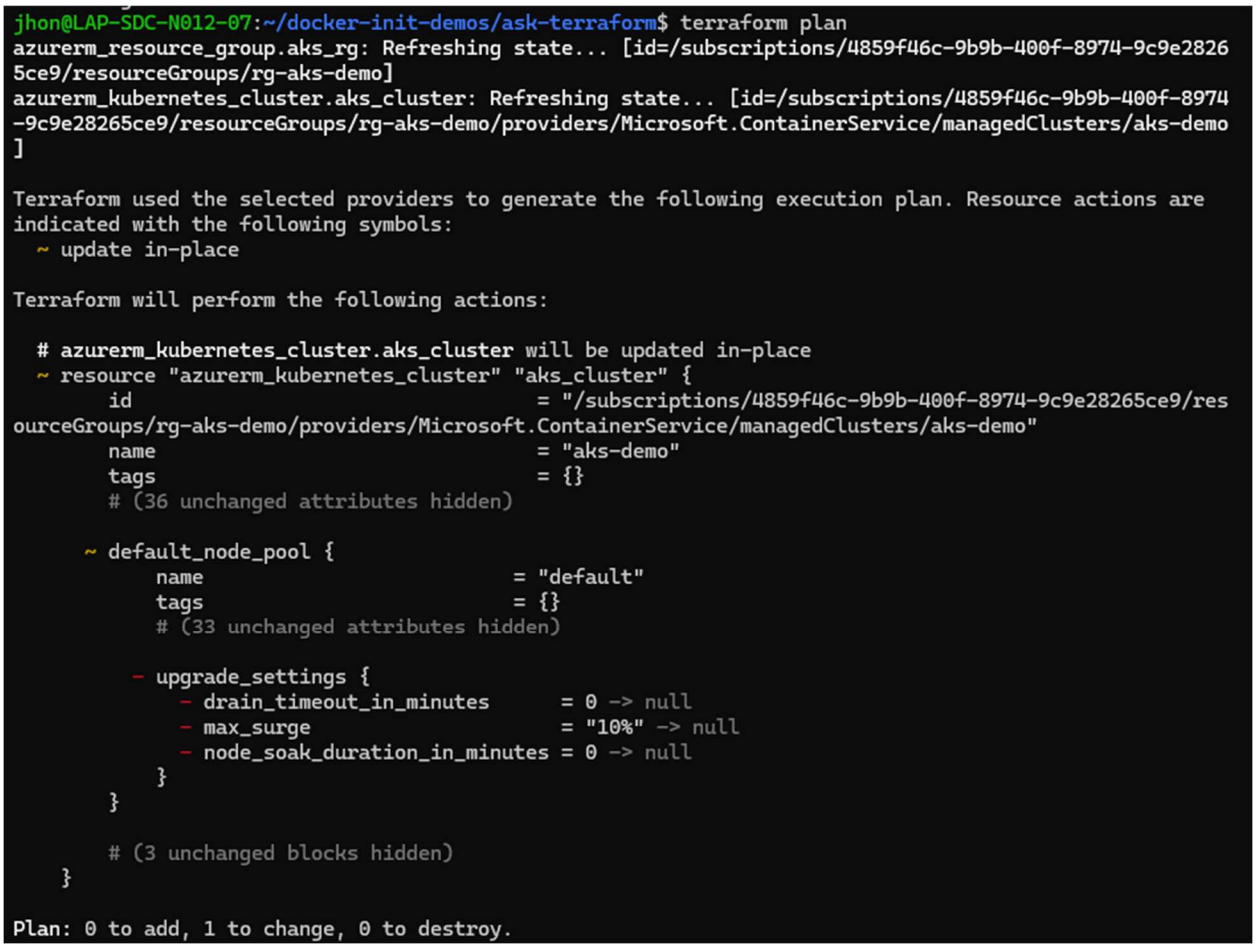
Clonar repositorio:

```
git clone https://github.com/jhonbilly/docker-init-demos

cd docker-init-demos
```

Instalación Sonarqube

```
apt-get update
apt-get install unzip software-properties-common wget default-jdk

apt-get install postgresql postgresql-contrib
su - postgres
psql

CREATE USER sonarqube WITH PASSWORD 'Test-dummy';
CREATE DATABASE sonarqube OWNER sonarqube;
GRANT ALL PRIVILEGES ON DATABASE sonarqube TO sonarqube;
\q

mkdir /downloads/sonarqube -p
cd /downloads/sonarqube
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-7.9.1.zip
unzip sonarqube-7.9.1.zip
mv sonarqube-7.9.1 /opt/sonarqube

adduser --system --no-create-home --group --disabled-login sonarqube
chown -R sonarqube:sonarqube /opt/sonarqube

vi /opt/sonarqube/bin/linux-x86-64/sonar.sh
RUN_AS_USER=sonarqube

vi /opt/sonarqube/conf/sonar.properties
sonar.jdbc.username=sonarqube
sonar.jdbc.password=Test-dummy
sonar.jdbc.url=jdbc:postgresql://localhost/sonarqube
sonar.web.javaAdditionalOpts=-server
sonar.web.host=0.0.0.0

vi /etc/security/limits.d/99-sonarqube.conf
sonarqube   -   nofile   65536
sonarqube   -   nproc    4096

vi /etc/sysctl.conf
vm.max_map_count=262144
fs.file-max=65536
reboot
/opt/sonarqube/bin/linux-x86-64/sonar.sh start
```

2. Infraestructura como Código (IaC) con Terraform



```
jhon@LAP-SDC-N012-07:~/docker-init-demos/ask-terraform$ terraform plan
azurerm_resource_group.aks_rg: Refreshing state... [id=/subscriptions/4859f46c-9b9b-400f-8974-9c9e2826
5ce9/resourceGroups/rg-aks-demo]
azurerm_kubernetes_cluster.aks_cluster: Refreshing state... [id=/subscriptions/4859f46c-9b9b-400f-8974
-9c9e28265ce9/resourceGroups/rg-aks-demo/providers/Microsoft.ContainerService/managedClusters/aks-demo
]

Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # azurerm_kubernetes_cluster.aks_cluster will be updated in-place
  ~ resource "azurerm_kubernetes_cluster" "aks_cluster" {
        id                                  = "/subscriptions/4859f46c-9b9b-400f-8974-9c9e28265ce9/res
ourceGroups/rg-aks-demo/providers/Microsoft.ContainerService/managedClusters/aks-demo"
        name                                = "aks-demo"
        tags                                = {}
        # (36 unchanged attributes hidden)

      ~ default_node_pool {
            name                          = "default"
            tags                          = {}
            # (33 unchanged attributes hidden)

          - upgrade_settings {
              - drain_timeout_in_minutes      = 0 -> null
              - max_surge                     = "10%" -> null
              - node_soak_duration_in_minutes = 0 -> null
            }
        }

        # (3 unchanged blocks hidden)
    }

Plan: 0 to add, 1 to change, 0 to destroy.
```

- Entra a la carpeta `/terraform-aks` y configura tu `main.tf` similar a esto:

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

resource "azurerm_resource_group" "aks_rg" {
  name     = "rg-aks-demo"
  location = "eastus"
}

resource "azurerm_kubernetes_cluster" "aks_cluster" {
  name                = "aks-demo"
  location            = azurerm_resource_group.aks_rg.location
  resource_group_name = azurerm_resource_group.aks_rg.name
  dns_prefix          = "aks-demo"
  default_node_pool {
    name       = "default"
    node_count = 1
    vm_size    = "Standard_B2s"
```

```
  }
  identity {
    type = "SystemAssigned"
  }
}
```

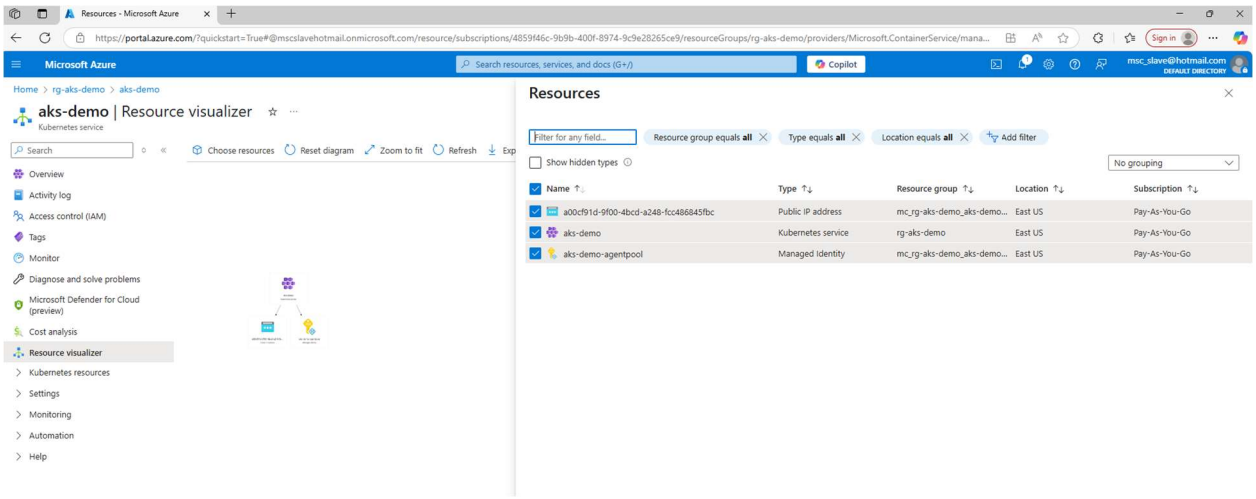
- Desplegar la infraestructura:

```
cd terraform-aks
terraform init
terraform apply
```

---

3. Conexión al Cluster AKS

```
az aks get-credentials --admin --name aks-demo --resource-group rg-aks-demo
```



5. Despliegue con Helm y configuración de Ingress

Instalar Helm si es necesario:

```
sudo snap install helm --classic
```

Crear o actualizar el Helm Chart (dentro de ``):

- Editar `values.yaml`:

```yaml
replicaCount: 1
image:
  repository: jhonbilly/docker-getting-started
  tag: "89"
  pullPolicy: Always
service:
  type: ClusterIP
  port: 8080
ingress:
  enabled: true
  className: nginx
  annotations: {}
  hosts:
    - host: ""    # Deja vacío si usarás la IP pública
      paths:
        - path: /
          pathType: Prefix
```

Aplicar el chart:

```
helm upgrade --install getting-started ./helm/docker-getting-started
```

Instalar Ingress Controller

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.9.6/deploy/static/provider/cloud/deploy.yaml
```

6. Validación y Verificación

Verifica recursos:

```
kubectl get all
kubectl get svc -A
kubectl get ingress
kubectl describe ingress nginx
kubectl logs -n ingress-nginx deployment/ingress-nginx-controller
```

Prueba desde tu navegador:
Abre la IP pública del servicio de tipo LoadBalancer (del controlador Ingress)
o ejecuta:

`curl http:// 52.255.211.254`

7. Buenas Prácticas y Versionamiento

- Excluye archivos generados de Terraform y builds con `.gitignore`:

  ```
  .terraform/
  *.tfstate
  terraform.tfstate.backup
  terraform-provider-*
  ```

- Sube todos los archivos de infraestructura, Helm y manifiestos a tu repo.
- Añade documentación de cada paso.

8. Pipeline CI/CD en Azure DevOps

```yaml
trigger:
  - main

pool:
  name: wsl-agent

variables:
  dockerRegistryServiceConnection: 'DockerHubServiceConnection'
  imageRepository: 'jhonbilly/docker-getting-started'
  containerRegistry: '$dockerRegistryServiceConnection'
  tag: '$(Build.BuildId)'
  SONAR_HOST_URL: 'http://172.20.38.158:9000'
  SONAR_PROJECT_KEY: 'docker-getting-started'
  SONAR_TOKEN: 'f4926fec98e5dd8491fd55400c0c167711d23aa2'

stages:
  - stage: AnalyzeAndBuild
    jobs:
      # 1. SonarQube Analysis (con continueOnError)
      - job: SonarQubeAnalysis
        displayName: 'SonarQube Analysis'
        continueOnError: true
        steps:
          - script: |
              cd node/
              npm install
            displayName: 'NPM Install'
          - script: |
              export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
              export PATH=$JAVA_HOME/bin:$PATH
              cd node/sample
              sonar-scanner \
                -Dsonar.projectKey=$(SONAR_PROJECT_KEY) \
                -Dsonar.sources=. \
                -Dsonar.host.url=$(SONAR_HOST_URL) \
                -Dsonar.login=$(SONAR_TOKEN)
            displayName: 'Run SonarQube Scanner'

      # 2. Build Job: SOLO si SonarQubeAnalysis fue OK
      - job: BuildApp
        displayName: 'Build app (solo si SonarQube OK)'
        dependsOn: SonarQubeAnalysis
```

```
        condition: succeeded('SonarQubeAnalysis')
        steps:
          - script: |
              echo "Análisis SonarQube exitoso, compilando app..."
              cd node/sample
              npm run build
            displayName: 'NPM Build'

      # 3. Escenario Fallido: SOLO si SonarQubeAnalysis Falla
      - job: SonarQubeFailed
        displayName: 'SonarQube Fallido'
        dependsOn: SonarQubeAnalysis
        condition: failed('SonarQubeAnalysis')
        steps:
          - script: |
              echo "El análisis SonarQube FALLÓ. Revisar calidad de código antes
de continuar."
              exit 1
            displayName: 'Mensaje Fallo SonarQube'

  # (Las demás stages igual que antes: DockerBuildPush, DeployK8s)
  - stage: DockerBuildPush
    dependsOn: AnalyzeAndBuild
    jobs:
      - job: DockerImage
        steps:
          - task: Docker@2
            displayName: Login to registry
            inputs:
              command: login
              containerRegistry: 'DockerHubServiceConnection'
          - task: Docker@2
            inputs:
              containerRegistry: '$(dockerRegistryServiceConnection)'
              repository: '$(imageRepository)'
              command: 'buildAndPush'
              Dockerfile: '**/node/sample/Dockerfile'
              buildContext: '$(Build.SourcesDirectory)/node'
              tags: |
                $(tag)

  - stage: DeployK8s
    dependsOn: DockerBuildPush
    jobs:
      - deployment: DeployApp
```

```
environment: 'dev'
strategy:
  runOnce:
    deploy:
      steps:
        - task: HelmInstaller@1
          inputs:
            helmVersionToInstall: 'latest'
        - task: HelmDeploy@0
          inputs:
            connectionType: 'Kubeconfig'
            kubeconfig: '$(KUBECONFIG)'
            command: 'upgrade'
            chartType: 'FilePath'
            chartPath: 'helm/docker-getting-started'
            releaseName: 'getting-started'
            overrideValues: |
              image.repository=$(containerRegistry)/$(imageRepository)
              image.tag=$(tag)
            install: true
```
