

Desafío 13

Este nuevo desafío lo inicias instalando **Minikube** y **kubectl** para administrar el cluster seguidamente abrimos la aplicación de **Docker** ya que **Minikube** lo usa para gestionar los cluster.

```
Windows PowerShell
PS C:\Users\Michael> kubectl version
Client Version: v1.30.2
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Unable to connect to the server: dial tcp 127.0.0.1:6443: connectex: No connection could be made because the target machine actively refused it.
PS C:\Users\Michael> minikube version
minikube version: v1.34.0
commit: 210b148df93a80eb872ecbeb7e35281b3c582c61
PS C:\Users\Michael> |
```

Seguido de esto ejecutamos en el powershell **Minikube** con el comando **Minikube start** y creamos el namespace de **ArgoCD** con el comando **kubectl create namespace argocd** luego de eso desplegamos los recursos de **ArgoCD** en **Minikube** con el comando **kubectl apply -n argocd -f**

<https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml>

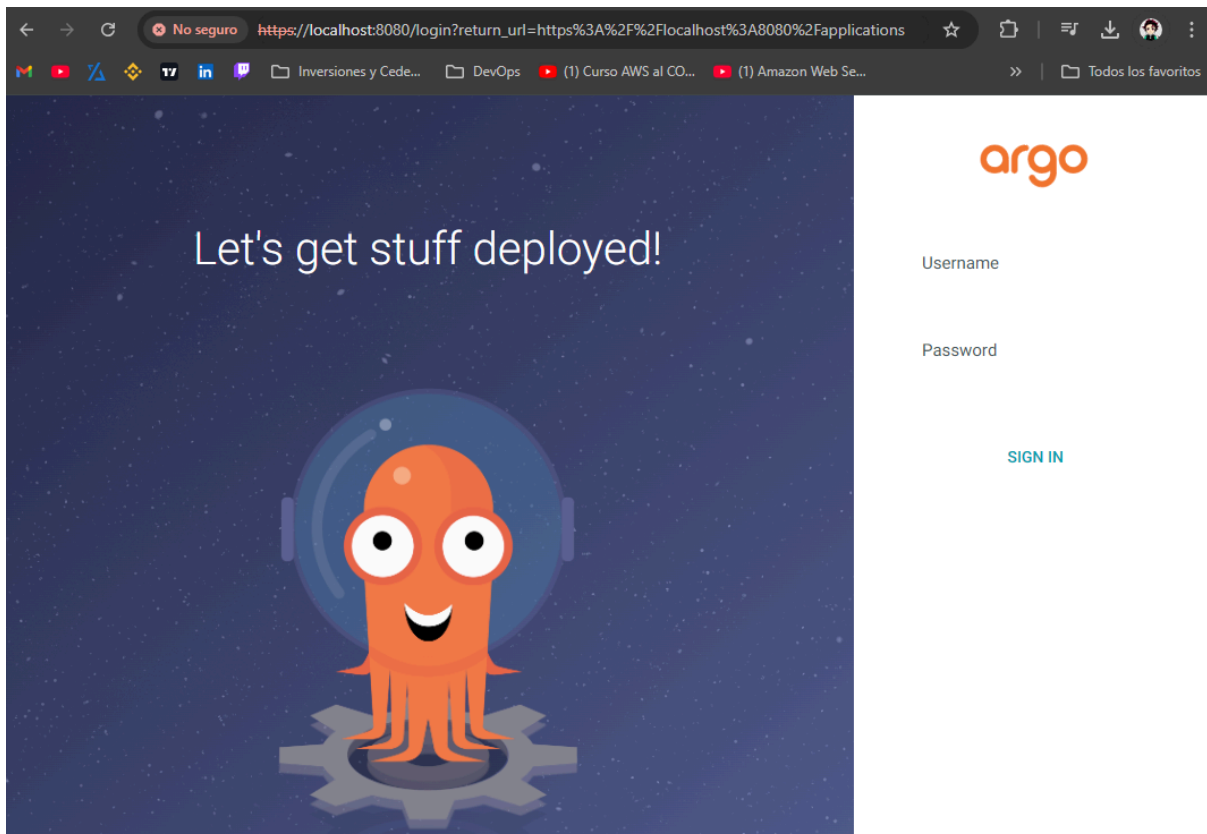
```
PS C:\Users\Michael> minikube start
minikube v1.34.0 en Microsoft Windows 11 Pro 10.0.22631.4169 Build 22631.4169
Controlador docker seleccionado automáticamente
Using Docker Desktop driver with root privileges
Starting "minikube" primary control-plane node in "minikube" cluster
Pulling base image v0.0.45 ...
Descargando Kubernetes v1.31.0 ...
> gcr.io/k8s-minikube/kicbase...: 487.90 MiB / 487.90 MiB 100.00% 10.84 M
> preloaded-images-k8s-v18-v1...: 326.69 MiB / 326.69 MiB 100.00% 5.10 Mi
Creating docker container (CPUs=2, Memory=4000MB) ...
Failing to connect to https://registry.k8s.io/ from both inside the minikube container and host machine
To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/referen
ce/networking/proxy/
Preparando Kubernetes v1.31.0 en Docker 27.2.0...
  Generando certificados y llaves
  Iniciando plano de control
  Configurando reglas RBAC...
Configurando CNI bridge CNI ...
Verifying Kubernetes components...
  Using image gcr.io/k8s-minikube/storage-provisioner:v5
Complementos habilitados: storage-provisioner, default-storageclass
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

PS C:\Users\Michael> kubectl create namespace argocd
namespace/argocd created
PS C:\Users\Michael> kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
customresourcedefinition.apiextensions.k8s.io/applications.argoproj.io unchanged
customresourcedefinition.apiextensions.k8s.io/applicationsets.argoproj.io unchanged
customresourcedefinition.apiextensions.k8s.io/appprojects.argoproj.io unchanged
serviceaccount/argocd-application-controller created
serviceaccount/argocd-applicationset-controller created
serviceaccount/argocd-dex-server created
serviceaccount/argocd-notifications-controller created
serviceaccount/argocd-redis created
serviceaccount/argocd-repo-server created
serviceaccount/argocd-server created
role.rbac.authorization.k8s.io/argocd-application-controller created
role.rbac.authorization.k8s.io/argocd-applicationset-controller created
role.rbac.authorization.k8s.io/argocd-dex-server created
role.rbac.authorization.k8s.io/argocd-notifications-controller created
role.rbac.authorization.k8s.io/argocd-redis created
role.rbac.authorization.k8s.io/argocd-server created
clusterrole.rbac.authorization.k8s.io/argocd-application-controller unchanged
clusterrole.rbac.authorization.k8s.io/argocd-applicationset-controller unchanged
clusterrole.rbac.authorization.k8s.io/argocd-server unchanged
rolebinding.rbac.authorization.k8s.io/argocd-application-controller created
rolebinding.rbac.authorization.k8s.io/argocd-applicationset-controller created
rolebinding.rbac.authorization.k8s.io/argocd-dex-server created
rolebinding.rbac.authorization.k8s.io/argocd-notifications-controller created
rolebinding.rbac.authorization.k8s.io/argocd-redis created
rolebinding.rbac.authorization.k8s.io/argocd-server created
clusterrolebinding.rbac.authorization.k8s.io/argocd-application-controller unchanged
clusterrolebinding.rbac.authorization.k8s.io/argocd-applicationset-controller unchanged
clusterrolebinding.rbac.authorization.k8s.io/argocd-server unchanged
configmap/argocd-cm created
```

Exponemos el servicio de **ArgoCD** para acceder al dashboard con el comando **kubectl port-forward svc/argocd-server -n argocd 8080:443**

```
PS C:\Users\Michael> kubectl port-forward svc/argocd-server -n argocd 8080:443
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
```

Ingresamos a nuestra dirección IP seguido del puerto 8080 <https://localhost:8080>



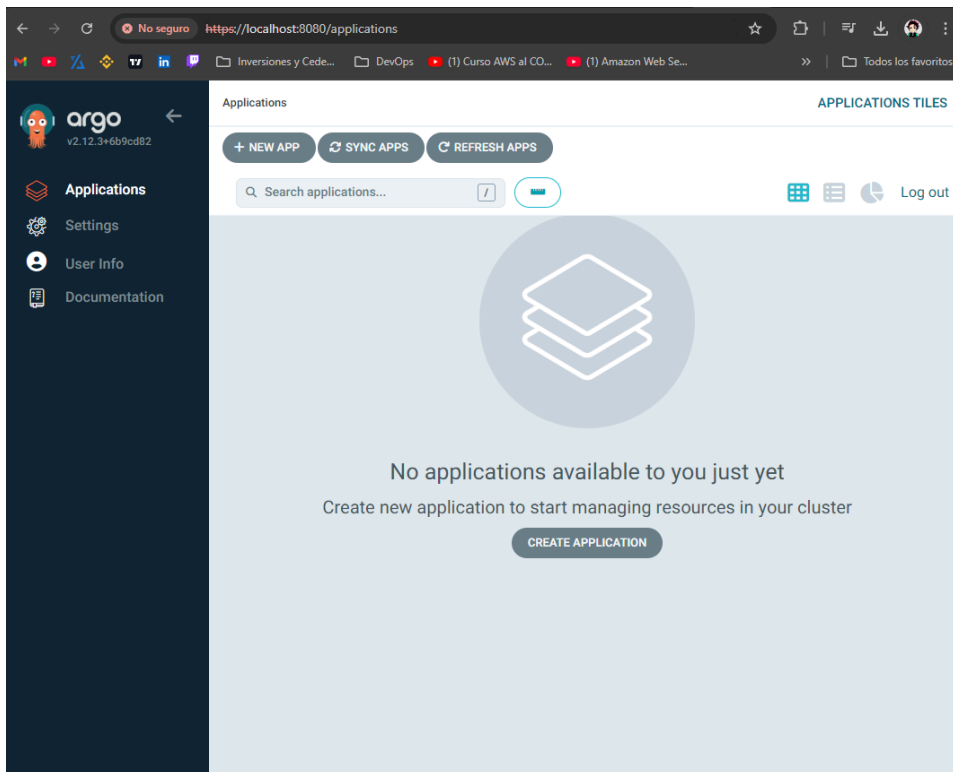
Seguido de esto obtenemos la contraseña del usuario admin para ingresar a **ArgoCD**, primero ingresamos el comando **kubectl get secret argocd-initial-admin-secret -n argocd -o jsonpath="{.data.password}"** para obtener el valor codificado en **base64** de la contraseña.

Luego con el comando

[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String("<your-base64-password>")) la decodificamos. Reemplazamos "**<your-base64-password>**" con la cadena que obtuvimos en el primer paso.

```
PS C:\Users\Michael> kubectl get secret argocd-initial-admin-secret -n argocd -o jsonpath="{.data.password}"
cDktSG5BeXR3Yk5QwThIdA==
PS C:\Users\Michael> [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String("cDktSG5BeXR3Yk5QwThIdA=="))
p9-HnAytwbNPY8Ht
PS C:\Users\Michael>
```

Luego accedemos nuevamente a nuestro <https://localhost:8080> corriendo el comando en powershell **kubectl port-forward svc/argocd-server -n argocd 8080:443** e ingresamos al dashboard con el usuario **admin** y el **password** descifrado en el paso anterior



Para crear nuestra aplicación nos dirigiremos a **Applications -> New App ->** y completamos los campos con los datos solicitados

SOURCE

Repository URL

<https://github.com/Kidbuut/Desafio-12>

GIT ▼

Revision

HEAD

Branch

Path

k8s/chart/nestjs-app

DESTINATION

Cluster URL

<https://kubernetes.default.svc>

URL ▼

Namespace

default

CREATE

CANCEL

PARAMETERS

autoscaling.enabled	false
autoscaling.maxReplicas	100
autoscaling.minReplicas	1
autoscaling.targetCPUUtilizationPercentage	80
env.MONGO_DB_NAME	test
env.MONGO_DB_PASS	mongo123
env.MONGO_DB_URI	mongodb://mongo:mongo123@mongodb-service:27017/test
env.MONGO_DB_USER	mongo
ingress.enabled	false
ingress.hosts[0].host	chart-example.local
mongodb.dbName	test
mongodb.dbPassword	mongo123
mongodb.dbUser	mongo
mongodb.image	mongo
mongodb.replicaCount	1

Una vez tengamos la configuración completa, vemos cómo se despliegan

