

Desafío N° 13

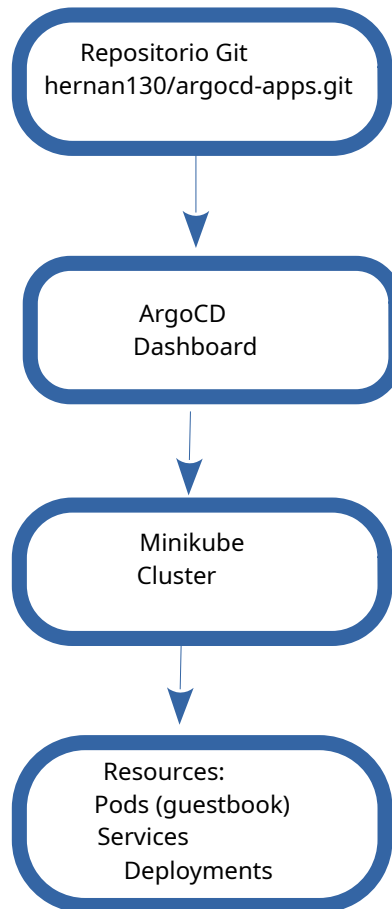
Hernan Andres Acosta-993394

Implementación de ArgoCD en Minikube

Fecha limite de entrega: 12/05/2025

Introducción

Este informe documenta la implementación de ArgoCD en un entorno local de Kubernetes usando Minikube, siguiendo los principios de GitOps para gestionar despliegues de aplicaciones de manera declarativa.



Requisitos:

Instalación de Minikube y kubectl

```
herman@andres:~$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

herman@andres:~$ kubectl version
Client Version: v1.32.3
Kustomize Version: v5.5.0
Server Version: v1.32.0
```

Despliegue de ArgoCD en Minikube

Creación del namespace argocd:

```
kubectl create namespace argocd
```

Despliegue de ArgoCD:

```
kubectl apply -n argocd -f
```

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

Configuración de ArgoCD

Acceso al Dashboard:

```
kubectl port-forward svc/argocd-server -n argocd 8080:443
```

Credenciales iniciales:

Usuario: admin

Contraseña:

```
kubectl get secret argocd-initial-admin-secret -n argocd -o jsonpath="{.data.password}" |  
base64 -decode
```

Conexión con Repositorio Git

Se configuró una aplicación en ArgoCD usando un repositorio Git personal (hernan130/argocd-apps):

Aplicación "guestbook" (Manifiestos YAML)

CLI:

```
argocd app create guestbook \
```

```
--repo https://github.com/hernan130/argocd-apps.git \
```

```
--path guestbook \
```

```
--dest-server https://kubernetes.default.svc \
```

```
--dest-namespace default
```

Login en ArgoCD Se realizó la autenticación en la interfaz de ArgoCD ejecutando:

```
bash Copiar Editar argocd login localhost:8080 --username admin --password <mi-password>
```

Sincronización:

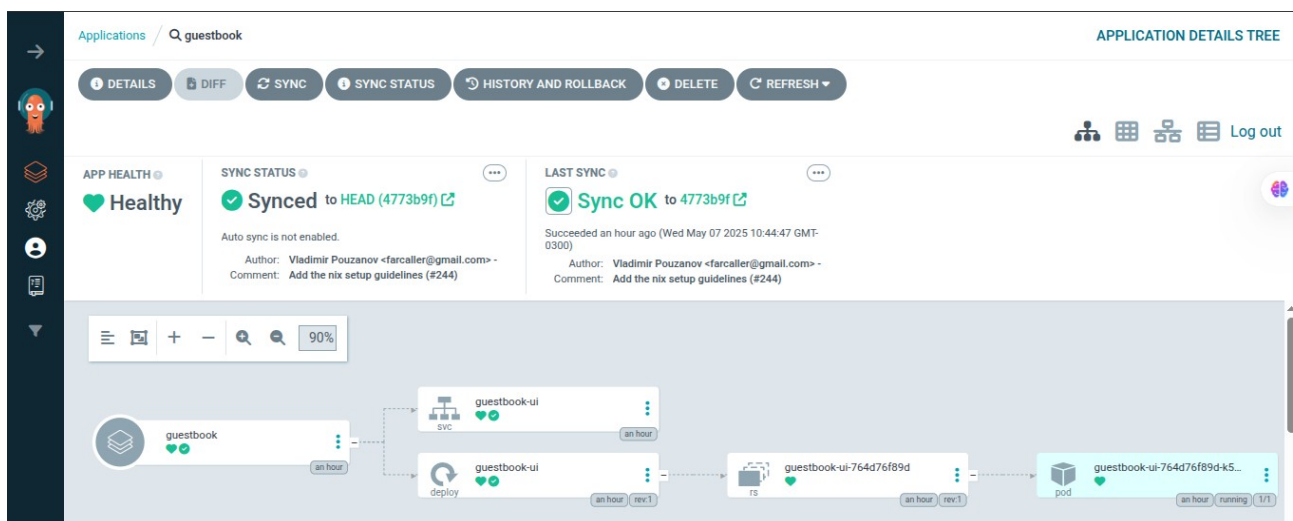
```
argocd app sync guestbook
```

Aplicación "helm-app" (Chart de Helm)

Configurada desde la UI de ArgoCD

Evidencia de Funcionamiento

Aplicación "guestbook" en estado Synced y Healthy:



→

Applications / helm-guestbook

APPLICATION DETAILS TREE

DETAILS

DIFF

SYNC

SYNC STATUS

HISTORY AND ROLLBACK

DELETE

REFRESH

Log out

APP HEALTH

Healthy

SYNC STATUS

Synced to HEAD (4773b9f)

Auto sync is not enabled.
Author: Vladimir Pouzanov <farcaller@gmail.com> -
Comment: Add the nix setup guidelines (#244)

LAST SYNC

Sync OK to 4773b9f

Succeeded 2 hours ago (Wed May 07 2025 10:16:02 GMT-0300)
Author: Vladimir Pouzanov <farcaller@gmail.com> -
Comment: Add the nix setup guidelines (#244)

90%

helm-guestbook

helm-guestbook

helm-guestbook

helm-guestbook-57c97698c4

helm-guestbook-57c97698c4

kubernetes

default

Buscar

+

🔔

Cargas de trabajo

Cargas de trabajo

Cron Jobs

Daemon Sets

Deployments

Jobs

Pods

Replica Sets

Controladores de Replicación

Stateful Sets

Service

Ingresses

Ingress Classes

Replica Sets

Despliegues

| Nombre | Imágenes | Etiquetas | Pods | Fecha de creación |
|----------------|--|--|-------|-------------------|
| guestbook-ui | gcr.io/heptio-images/ks-guestbook-demo:0.2 | | 1 / 1 | an hour ago |
| helm-guestbook | gcr.io/heptio-images/ks-guestbook-demo:0.1 | app: helm-guestbook chart: helm-guestbook-0.1.0 heritage: Helm | 1 / 1 | an hour ago |

Pods

Cargas de trabajo > Pods

Deployments

Jobs

Pods

Replica Sets

Controladores de Replicación

Stateful Sets

Service

Ingresses

Ingress Classes

Services

Configuración y Almacenamiento

Config Maps

Persistent Volume Claims

Pods

| Nombre | Imágenes | Etiquetas | Nodo | Estado | Reinicios | Utilización de CPU (núcleos) | Utilización de memoria (octetos) | Fecha de creación |
|---------------------------------|--|---|----------|---------|-----------|------------------------------|----------------------------------|-------------------|
| guestbook-ui-764d76f89d-k55bz | gcr.io/heptio-images/ks-guestbook-demo:0.2 | app: guestbook-ui pod-template-hash: 764d76f89d | minikube | Running | 0 | - | - | 2 hours ago |
| helm-guestbook-57c97698c4-rz5wv | gcr.io/heptio-images/ks-guestbook-demo:0.1 | app: helm-guestbook pod-template-hash: 57c97698c4 release: helm-guestbook | minikube | Running | 0 | - | - | 2 hours ago |

| Service | | | | | | | |
|--------------------------------|--|--|--|--|--|--|--|
| Replica Sets | | | | | | | |
| Controladores de Replicación | | | | | | | |
| Stateful Sets | | | | | | | |
| Service | | | | | | | |
| Ingresses | | | | | | | |
| Ingress Classes | | | | | | | |
| Services | | | | | | | |
| Configuración y Almacenamiento | | | | | | | |

| Servicios | | | | | | | |
|------------------|--|-----------|---------------|---|--------------------|-------------------|--|
| Nombre | Etiquetas | Tipo | IP cluster | Endpoints Internos | Endpoints Externos | Fecha de creación | |
| ● guestbook-ui | - | ClusterIP | 10.100.39.153 | guestbook-ui:80 TCP guestbook-ui:0 TCP | - | 2 hours ago | |
| ● helm-guestbook | app: helm-guestbook chart: helm-guestbook-0.1.0 heritage: Helm | ClusterIP | 10.105.35.175 | helm-guestbook:80 TCP helm-guestbook:0 TCP | - | 2 hours ago | |

Aplicación Helm desplegada:

Verificación en Kubernetes

kubectl get applications -n argocd

```

helm-guestbook-57c97698c4-rz5wv 1/1 Running 0 57m
hernan@andres:~$ kubectl get applications -n argocd
NAME SYNC STATUS HEALTH STATUS
guestbook Synced Healthy
helm-guestbook Synced Healthy
hernan@andres:~$

```

kubectl get pods -n default

```

hernan@andres:~$ kubectl get pods -n default
NAME READY STATUS RESTARTS AGE
guestbook-ui-764d76f89d-k55bz 1/1 Running 0 29m
helm-guestbook-57c97698c4-rz5wv 1/1 Running 0 57m
hernan@andres:~$

```

Código Fuente

Repositorio Git con los manifiestos: github.com/hernan130/argocd-apps.git

Estructura:

```

hernan@andres:~/Documentos/Educacion_It/argocd-example-apps/guestbook$ tree
.
├── guestbook-ui-deployment.yaml
└── guestbook-ui-svc.yaml

1 directory, 2 files
hernan@andres:~/Documentos/Educacion_It/argocd-example-apps/guestbook$

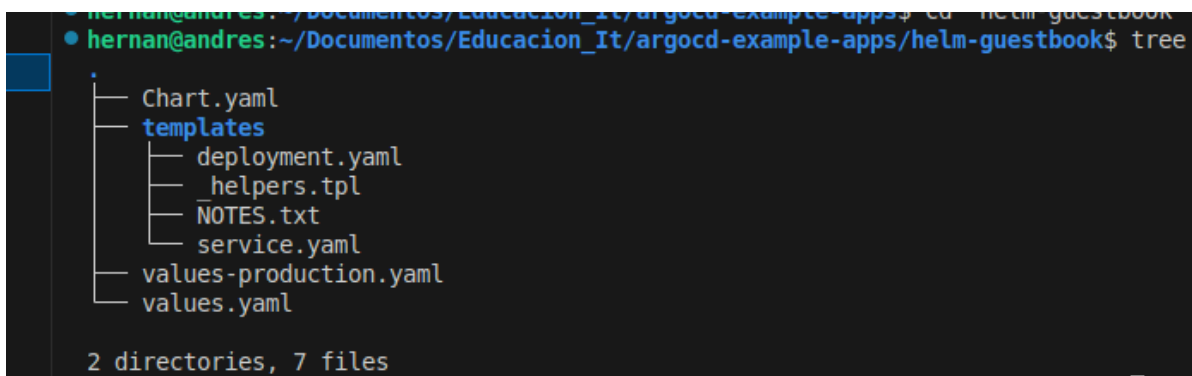
```

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: guestbook-ui
spec:
  replicas: 1
  revisionHistoryLimit: 3
  selector:
    matchLabels:
      app: guestbook-ui
  template:
    metadata:
      labels:
        app: guestbook-ui
    spec:
      containers:
        - image: gcr.io/heptio-images/ks-guestbook-demo:0.2
          name: guestbook-ui
      ports:
        - containerPort: 80
```

svc.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: guestbook-ui
spec:
  ports:
    - port: 80
  targetPort: 80
  selector:
    app: guestbook-ui
```



A terminal window showing the command `tree` being executed in the directory `~/Documentos/Educacion_It/argocd-example-apps/helm-guestbook`. The output shows a directory structure with a `templates` subdirectory containing `deployment.yaml`, `helpers.tpl`, `NOTES.txt`, and `service.yaml`. Other files in the root directory are `Chart.yaml`, `values-production.yaml`, and `values.yaml`. The terminal also shows the prompt `hernan@andres: ~/Documentos/Educacion_It/argocd-example-apps/helm-guestbook` and the command `cd helm-guestbook`.

```
hernan@andres: ~/Documentos/Educacion_It/argocd-example-apps/helm-guestbook$ cd helm-guestbook
hernan@andres:~/Documentos/Educacion_It/argocd-example-apps/helm-guestbook$ tree
.
├── Chart.yaml
├── templates
│   ├── deployment.yaml
│   ├── helpers.tpl
│   ├── NOTES.txt
│   └── service.yaml
├── values-production.yaml
└── values.yaml

2 directories, 7 files
```

value.yaml

```
# Default values for helm-guestbook.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.
```

replicaCount: 1

image:

repository: gcr.io/heptio-images/ks-guestbook-demo

tag: 0.1

pullPolicy: IfNotPresent

containerPort: 80

service:

type: ClusterIP

port: 80

ingress:

enabled: false

annotations: {}

kubernetes.io/ingress.class: nginx

kubernetes.io/tls-acme: "true"

path: /

hosts:

- chart-example.local

tls: []

- secretName: chart-example-tls

hosts:

- chart-example.local

resources: {}

We usually recommend not to specify default resources and to leave this as a conscious

choice for the user. This also increases chances charts run on environments with little

resources, such as Minikube. If you do want to specify resources, uncomment the following

lines, adjust them as necessary, and remove the curly braces after 'resources:'.

limits:

cpu: 100m

memory: 128Mi

requests:

cpu: 100m

memory: 128Mi

nodeSelector: {}

tolerations: []

affinity: {}

chart.yaml

apiVersion: v2

name: helm-guestbook

description: A Helm chart for Kubernetes

A chart can be either an 'application' or a 'library' chart.

```
#  
# Application charts are a collection of templates that can be packaged into versioned archives  
# to be deployed.  
#  
# Library charts provide useful utilities or functions for the chart developer. They're included as  
# a dependency of application charts to inject those utilities and functions into the rendering  
# pipeline. Library charts do not define any templates and therefore cannot be deployed.  
type: application  
  
# This is the chart version. This version number should be incremented each time you make  
# changes  
# to the chart and its templates, including the app version.  
# Versions are expected to follow Semantic Versioning (https://semver.org/)  
version: 0.1.0  
  
# This is the version number of the application being deployed. This version number should be  
# incremented each time you make changes to the application. Versions are not expected to  
# follow Semantic Versioning. They should reflect the version the application is using.  
appVersion: "1.0"
```

Conclusión

El presente desafío permitió implementar con éxito ArgoCD en un entorno Minikube, estableciendo un flujo de trabajo GitOps completo para la gestión declarativa de aplicaciones Kubernetes. A través de este ejercicio se logró:

Automatización de despliegues: Se configuró ArgoCD para sincronizar automáticamente el estado del cluster con los manifiestos almacenados en el repositorio Git (hernan130/argocd-apps), demostrando el principio fundamental de GitOps: "La verdad está en el repositorio".

Gestión multi-metodología: Se desplegaron aplicaciones usando tanto manifiestos YAML nativos (guestbook) como charts de Helm, validando la flexibilidad de ArgoCD para trabajar con diferentes herramientas de empaquetamiento.

Verificación del ciclo completo: Desde la definición en Git hasta la creación de los recursos en Kubernetes (Pods, Services, Deployments), se comprobó el funcionamiento integral del sistema mediante:

Dashboard de ArgoCD (estado Synced/Healthy)

Comandos kubectl de verificación

Acceso a las aplicaciones desplegadas

Beneficios obtenidos:

Control de versiones: Todo cambio debe pasar por el repositorio Git

Auditoría: Historial completo de despliegues

Recuperación ante fallos: Estado deseado siempre definido en Git

Consistencia: Entornos idénticos entre desarrollo y producción

Lecciones aprendidas:

La importancia de configurar correctamente el port-forward para acceder al dashboard

La necesidad de gestionar cuidadosamente los secrets (como la contraseña inicial de ArgoCD)

La ventaja de usar namespaces dedicados (argocd) para herramientas de administración

Este ejercicio no solo cumplió con los objetivos del desafío, sino que sentó las bases para implementaciones más avanzadas de GitOps en entornos empresariales, demostrando el poder de ArgoCD como herramienta central en la plataforma DevOps.

Hernan Andres Acosta Estudiante DevOps Bootcamp Engineering 07/05/2025