Argo CD Setup on Kubernetes

**1. Create a Namespace for Argo CD**

kubectl create namespace argocd

This isolates Argo CD resources within their own Kubernetes namespace.

**2. Install Argo CD using Official Manifest**

kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

This installs all the core components including:

* argocd-server
* argocd-repo-server
* argocd-dex-server (for SSO)
* argocd-application-controller

**3. Verify Argo CD Services**

kubectl get svc -n argocd

This lists services and their types. By default, argocd-server is a ClusterIP service.

**4. Expose Argo CD Server via NodePort**

kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "NodePort"}}'

This makes Argo CD externally accessible by assigning a port on each node.

**5. Retrieve Argo CD Initial Admin Password**

kubectl get secret -n argocd argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d && echo

This gives you the default admin password for logging in to the UI or CLI.

**6. Access Argo CD UI**

Use the NodePort:

https://<EC2-public-ip>:<NodePort>

Example from your setup:

https://13.127.192.210:30564

If using port-forwarding instead:

kubectl port-forward svc/argocd-server -n argocd 8888:443

Access it via:

https://localhost:8888

**How Argo CD Works**

Argo CD is a declarative GitOps tool that continuously compares the desired state stored in a Git repository against the live state of your Kubernetes cluster.

**Core Concepts:**

* **Application**: Defines a Git repo, target cluster, and the path to manifests.
* **Sync**: ArgoCD applies manifests from Git to the cluster and ensures they're kept in sync.
* **Diff**: It shows what's changed between Git and the cluster.
* **Hooks & Notifications**: Optional triggers for external systems.

**Workflow Summary:**

1. You define Kubernetes manifests (YAML) in Git.
2. Argo CD monitors the Git repo for changes.
3. When changes are detected, Argo CD syncs the cluster state.
4. Argo CD UI/CLI shows drift status and lets you trigger manual or auto-syncs.

**Issues Encountered and How You Solved Them**

**1. Incorrect Port-Forward Target**

You tried:

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

But the service didn't expose port 8080. Resolution:

kubectl port-forward svc/argocd-server -n argocd 8888:443

**2. Port Already in Use**

You attempted to forward to 8080, but got:

address already in use

Solution: You switched to port 8888 which was free.

**3. Accessing Argo CD Externally via EC2**

Initially, you accessed:

https://13.127.192.210:8888

That didn’t work because port-forwarding binds to localhost only. Solution: You used the NodePort assigned (30564) and successfully reached:

https://13.127.192.210:30564

**4. Docker Compose YAML Misinterpreted by kubectl**

During deployment, you ran:

kubectl apply -f .

Which included docker-compose.yml. It failed because that file lacks apiVersion and kind. Resolution: You corrected this to:

kubectl apply -f k8s/

**Install Kubernetes Dashboard**

**Deploy the Dashboard**

kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml



**Generate Login Token**

kubectl -n kubernetes-dashboard create token admin-user

Port-Forward the Dashboard Service

kubectl port-forward svc/kubernetes-dashboard -n kubernetes-dashboard 8888:443 --address=0.0.0.0 &



**Access at:**

https://<ec2ip>:8888

You’ll use the token from step 2 to log into the UI.

How Argo CD and Kubernetes Dashboard Work Together

| **Component** | **Role** |
| --- | --- |
| **Argo CD** | GitOps controller: syncs manifests from Git to the cluster |
| **Kubernetes Dashboard** | Web UI for viewing & managing cluster resources manually |
| **Git** | Source of truth for desired application state |
| **Argo CD Application** | Defines the repo, path, sync policy, and target namespace |
| **Dashboard Token** | Authenticates access to the Dashboard UI via secure port-forwarding |