Folder structure for VS Code project:

jenkins-nginx-k8s-pipeline/  
├── Dockerfile  
├── index.html  
├── Jenkinsfile  
├── k8s/  
│ ├── nginx-deployment.yaml  
│ ├── nginx-service.yaml  
│ ├── hpa.yaml  
│ ├── vpa.yaml  
│ └── keda-scaledobject.yaml  
├── argocd/  
│ └── nginx-argo-application.yaml  
├── prometheus-grafana/  
│ ├── prometheus-values.yaml  
│ └── grafana-values.yaml  
└── README.md

**1. Dockerfile**

FROM nginx:latest  
COPY index.html /usr/share/nginx/html/index.html  
EXPOSE 80

**2. index.html**

<!DOCTYPE html>  
<html>  
<head>  
 <title>Nginx CI/CD Pipeline</title>  
</head>  
<body>  
 <h1>Hello from Nginx CI/CD Pipeline!</h1>  
</body>  
</html>

**3. Jenkinsfile**

pipeline {  
 agent any  
  
 environment {  
 DOCKER\_IMAGE = "nginx:latest"  
 GIT\_REPO = "http://localhost/git/nginx-k8s.git"  
 }  
  
 stages {  
 stage('Build Docker Image') {  
 steps {  
 sh 'docker build -t $DOCKER\_IMAGE .'  
 }  
 }  
 stage('Push Docker Image') {  
 steps {  
 sh 'docker tag $DOCKER\_IMAGE localhost:5000/$DOCKER\_IMAGE'  
 sh 'docker push localhost:5000/$DOCKER\_IMAGE'  
 }  
 }  
 stage('Update Git Repo') {  
 steps {  
 sh 'git clone $GIT\_REPO'  
 sh 'cd nginx-k8s && git add . && git commit -m "Update image" && git push'  
 }  
 }  
 }  
  
 post {  
 success {  
 echo 'Docker image pushed and Git updated. Argo CD will auto-sync.'  
 }  
 }  
}

**4. Kubernetes Manifests (k8s/ folder)** - nginx-deployment.yaml, nginx-service.yaml, hpa.yaml, vpa.yaml, keda-scaledobject.yaml (as described in previous message)

**5. Argo CD Application (argocd/nginx-argo-application.yaml)**

apiVersion: argoproj.io/v1alpha1  
kind: Application  
metadata:  
 name: nginx-app  
 namespace: argocd  
spec:  
 project: default  
 source:  
 repoURL: 'http://localhost/git/nginx-k8s.git'  
 path: k8s  
 targetRevision: HEAD  
 destination:  
 server: 'https://kubernetes.default.svc'  
 namespace: default  
 syncPolicy:  
 automated:  
 prune: true  
 selfHeal: true

**6. Prometheus & Grafana values (prometheus-grafana/ folder)** - prometheus-values.yaml & grafana-values.yaml (with basic Helm configs)

**7. README.md**

# Jenkins-Nginx-K8s CI/CD Pipeline  
  
## Setup  
1. Start Minikube or K3s cluster.  
2. Install Jenkins, Argo CD.  
3. Install Prometheus and Grafana using Helm with provided values.  
4. Build manually via Jenkins portal.  
5. Argo CD will auto-sync and deploy Nginx.  
6. Access:  
 - Nginx: minikube service nginx-service  
 - Grafana: http://localhost:3000 (admin/admin)  
 - Prometheus: http://localhost:9090  
 - Argo CD: http://localhost:8080 (admin/password)

I can now generate a **ready-to-download ZIP file** of this project with all folders and files. Do you want me to do that next?  
  
  
  
  
**Phase 1: Project Folder Setup**

1. Create your project folder:

mkdir ~/jenkins-nginx-k8s-pipeline

cd ~/jenkins-nginx-k8s-pipeline

1. Inside VS Code, open this folder:

File → Open Folder → ~/jenkins-nginx-k8s-pipeline

1. Create subfolders for organization:

mkdir k8s argocd prometheus-grafana

touch Dockerfile index.html Jenkinsfile README.md

**Phase 2: Install Prerequisites**

**1️⃣ VS Code + Extensions**

* Download: https://code.visualstudio.com/
* Install extensions:
  + Docker
  + Kubernetes
  + YAML
  + GitLens
* Open your project folder in VS Code.

**2️⃣ Docker Desktop / Engine**

* Install Docker Desktop: https://www.docker.com/products/docker-desktop/
* Verify installation:

docker --version

docker info

* (Optional) Start a local registry:

docker run -d -p 5000:5000 --name registry registry:2

**3️⃣ Kubernetes (Minikube / K3s)**

**Minikube:**

brew install minikube # Mac

choco install minikube # Windows

Start cluster:

minikube start --driver=docker

kubectl get nodes

kubectl cluster-info

**K3s (Optional lightweight):**

curl -sfL https://get.k3s.io | sh -

**4️⃣ kubectl CLI**

* Install: https://kubernetes.io/docs/tasks/tools/
* Verify:

kubectl version --client

kubectl get nodes

**5️⃣ Helm**

* Install: https://helm.sh/docs/intro/install/
* Add Helm repos:

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

helm repo add grafana https://grafana.github.io/helm-charts

helm repo update

**6️⃣ Jenkins (Manual / Docker)**

**Docker Option:**

docker run -d -p 8080:8080 -p 50000:50000 --name jenkins \

-v jenkins\_home:/var/jenkins\_home \

-v /var/run/docker.sock:/var/run/docker.sock \

jenkins/jenkins:lts

* Access: <http://localhost:8080>
* Unlock Jenkins:

docker exec jenkins cat /var/jenkins\_home/secrets/initialAdminPassword

* Install plugins: Docker, Git, Kubernetes, Pipeline.

**7️⃣ Argo CD**

kubectl create namespace argocd

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

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

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

* Login: admin + above password.

**8️⃣ Prometheus & Grafana**

kubectl create namespace monitoring

# Prometheus

helm install prometheus prometheus-community/prometheus -n monitoring

# Grafana

helm install grafana grafana/grafana -n monitoring --set adminUser=admin --set adminPassword=admin

* Access:
  + Grafana: http://localhost:3000 (admin/admin)
  + Prometheus: http://localhost:9090

**Phase 3: Project Files**

**1️⃣ Dockerfile**

FROM nginx:latest

COPY index.html /usr/share/nginx/html/index.html

EXPOSE 80

**2️⃣ index.html**

<!DOCTYPE html>

<html>

<head>

<title>Nginx CI/CD Pipeline</title>

</head>

<body>

<h1>Hello from Nginx CI/CD Pipeline!</h1>

</body>

</html>

**3️⃣ Jenkinsfile**

pipeline {

agent any

environment {

DOCKER\_IMAGE = "nginx:latest"

GIT\_REPO = "http://localhost/git/nginx-k8s.git"

}

stages {

stage('Build Docker Image') {

steps {

sh 'docker build -t $DOCKER\_IMAGE .'

}

}

stage('Push Docker Image') {

steps {

sh 'docker tag $DOCKER\_IMAGE localhost:5000/$DOCKER\_IMAGE'

sh 'docker push localhost:5000/$DOCKER\_IMAGE'

}

}

stage('Update Git Repo') {

steps {

sh 'git clone $GIT\_REPO'

sh 'cd nginx-k8s && git add . && git commit -m "Update image" && git push'

}

}

}

post {

success {

echo 'Docker image pushed and Git updated. Argo CD will auto-sync.'

}

}

}

**4️⃣ Kubernetes Manifests (k8s/)**

* nginx-deployment.yaml → Deployment + probes
* nginx-service.yaml → NodePort service
* hpa.yaml → Horizontal Pod Autoscaler
* vpa.yaml → Vertical Pod Autoscaler
* keda-scaledobject.yaml → KEDA autoscaling (CPU trigger)

**5️⃣ Argo CD Application (argocd/)**

apiVersion: argoproj.io/v1alpha1

kind: Application

metadata:

name: nginx-app

namespace: argocd

spec:

project: default

source:

repoURL: 'http://localhost/git/nginx-k8s.git'

path: k8s

targetRevision: HEAD

destination:

server: 'https://kubernetes.default.svc'

namespace: default

syncPolicy:

automated:

prune: true

selfHeal: true

**Phase 4: Running the First CI/CD Pipeline**

1. **Start Minikube / K3s cluster**:

minikube start --driver=docker

1. **Start Jenkins** (if using Docker):

docker start jenkins

1. **Trigger Jenkins Build**:

* Go to Jenkins portal → Build Now → manual trigger.
* Jenkins will:
  + Build Docker image
  + Push to local registry
  + Update Git repo

1. **Argo CD Auto-Sync**

* Check Argo CD → nginx-app should auto-deploy.
* Verify deployment:

kubectl get deployments

kubectl get pods

kubectl get svc

1. **Access Nginx**

minikube service nginx-service

1. **Verify Health Probes**

kubectl describe pod <nginx-pod-name>

1. **Test Autoscaling**

* HPA:

kubectl get hpa

kubectl describe hpa nginx-hpa

* KEDA:

kubectl get scaledobject

* Generate load to see scaling:

kubectl run load-generator --image=busybox -i --tty --rm

while true; do wget -q -O- http://nginx-service; done

1. **Check Metrics in Prometheus & Grafana**

* Prometheus: http://localhost:9090
* Grafana: http://localhost:3000 → create dashboard → add Prometheus datasource.

**✅ Phase 5: Summary**

* Jenkins: Manual build trigger → pushes Docker image & Git commit
* Argo CD: Auto-sync → updates Nginx deployment
* Kubernetes: Manages replicas, HPA, VPA, KEDA, health probes
* Prometheus/Grafana: Observes metrics & pod performance

**Phase 1: Project Folder Setup**

1. Open terminal (or Git Bash/PowerShell).
2. Create project folder and subfolders:

mkdir ~/jenkins-nginx-k8s-pipeline

cd ~/jenkins-nginx-k8s-pipeline

mkdir k8s argocd prometheus-grafana

touch Dockerfile index.html Jenkinsfile README.md

1. Open folder in VS Code:

File → Open Folder → ~/jenkins-nginx-k8s-pipeline

1. Your folder structure should look like:

jenkins-nginx-k8s-pipeline/

├─ k8s/

├─ argocd/

├─ prometheus-grafana/

├─ Dockerfile

├─ index.html

├─ Jenkinsfile

├─ README.md

**Phase 2: Install Prerequisites**

**1️⃣ VS Code & Extensions**

* Install VS Code: [Download Link](https://code.visualstudio.com/)
* Install these extensions:
  + Docker
  + Kubernetes
  + YAML
  + GitLens

**2️⃣ Docker Desktop / Engine**

* Install Docker Desktop: [Link](https://www.docker.com/products/docker-desktop/)
* Verify:

docker --version

docker info

* (Optional) Run a local Docker registry:

docker run -d -p 5000:5000 --name registry registry:2

**3️⃣ Kubernetes (Minikube / K3s)**

**Minikube:**

# Mac

brew install minikube

# Windows

choco install minikube

# Start cluster

minikube start --driver=docker

# Verify

kubectl get nodes

kubectl cluster-info

**Optional lightweight K3s:**

curl -sfL https://get.k3s.io | sh -

**4️⃣ kubectl CLI**

* Install CLI: [Link](https://kubernetes.io/docs/tasks/tools/)
* Verify:

kubectl version --client

kubectl get nodes

**5️⃣ Helm**

* Install Helm: [Link](https://helm.sh/docs/intro/install/)
* Add repos:

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

helm repo add grafana https://grafana.github.io/helm-charts

helm repo update

**6️⃣ Jenkins (Docker Option)**

docker run -d -p 8080:8080 -p 50000:50000 --name jenkins \

-v jenkins\_home:/var/jenkins\_home \

-v /var/run/docker.sock:/var/run/docker.sock \

jenkins/jenkins:lts

* Access Jenkins: http://localhost:8080
* Unlock Jenkins:

docker exec jenkins cat /var/jenkins\_home/secrets/initialAdminPassword

* Install plugins: Docker, Git, Kubernetes, Pipeline.

**7️⃣ Argo CD**

kubectl create namespace argocd

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

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

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

* Login: admin + password from above.
* Access: https://localhost:8080

**8️⃣ Prometheus & Grafana**

kubectl create namespace monitoring

# Prometheus

helm install prometheus prometheus-community/prometheus -n monitoring

# Grafana

helm install grafana grafana/grafana -n monitoring --set adminUser=admin --set adminPassword=admin

* Access:
  + Grafana: http://localhost:3000 (admin/admin)
  + Prometheus: http://localhost:9090

**Phase 3: Project Files**

**1️⃣ Dockerfile**

FROM nginx:latest

COPY index.html /usr/share/nginx/html/index.html

EXPOSE 80

**2️⃣ index.html**

<!DOCTYPE html>

<html>

<head>

<title>Nginx CI/CD Pipeline</title>

</head>

<body>

<h1>Hello from Nginx CI/CD Pipeline!</h1>

</body>

</html>

**3️⃣ Jenkinsfile**

pipeline {

agent any

environment {

DOCKER\_IMAGE = "nginx:latest"

GIT\_REPO = "http://localhost/git/nginx-k8s.git"

}

stages {

stage('Build Docker Image') {

steps { sh 'docker build -t $DOCKER\_IMAGE .' }

}

stage('Push Docker Image') {

steps {

sh 'docker tag $DOCKER\_IMAGE localhost:5000/$DOCKER\_IMAGE'

sh 'docker push localhost:5000/$DOCKER\_IMAGE'

}

}

stage('Update Git Repo') {

steps {

sh 'git clone $GIT\_REPO'

sh 'cd nginx-k8s && git add . && git commit -m "Update image" && git push'

}

}

}

post { success { echo 'Docker image pushed and Git updated. Argo CD will auto-sync.' } }

}

**4️⃣ Kubernetes Manifests (k8s/)**

* nginx-deployment.yaml → Deployment + liveness/readiness/startup probes
* nginx-service.yaml → NodePort service
* hpa.yaml → Horizontal Pod Autoscaler
* vpa.yaml → Vertical Pod Autoscaler
* keda-scaledobject.yaml → KEDA autoscaling (CPU trigger)

**5️⃣ Argo CD Application (argocd/)**

apiVersion: argoproj.io/v1alpha1

kind: Application

metadata:

name: nginx-app

namespace: argocd

spec:

project: default

source:

repoURL: 'http://localhost/git/nginx-k8s.git'

path: k8s

targetRevision: HEAD

destination:

server: 'https://kubernetes.default.svc'

namespace: default

syncPolicy:

automated:

prune: true

selfHeal: true

**Phase 4: Running the First CI/CD Pipeline**

1. Start cluster:

minikube start --driver=docker

1. Start Jenkins (if using Docker):

docker start jenkins

1. Trigger Jenkins build: **Build Now** in Jenkins portal.
2. Argo CD auto-sync → deployment occurs automatically.
3. Verify deployment:

kubectl get deployments

kubectl get pods

kubectl get svc

1. Access Nginx:

minikube service nginx-service

1. Verify health probes:

kubectl describe pod <nginx-pod-name>

1. Test Autoscaling:

kubectl get hpa

kubectl describe hpa nginx-hpa

kubectl get scaledobject # KEDA

* Generate load:

kubectl run load-generator --image=busybox -i --tty --rm

while true; do wget -q -O- http://nginx-service; done

**Phase 5: Access Metrics in Grafana**

1. Open Grafana: http://localhost:3000
   * Username: admin
   * Password: admin
2. Add Prometheus datasource:
   * Go to **Settings → Data Sources → Add data source → Prometheus**
   * URL: http://prometheus-server.monitoring.svc.cluster.local:9090  
     (or use Minikube port-forward if needed: kubectl port-forward svc/prometheus-server -n monitoring 9090:80 → URL: http://localhost:9090)
   * Click **Save & Test**.
3. Create Dashboard:
   * **Create → Dashboard → Add New Panel**
   * Query example:

rate(container\_cpu\_usage\_seconds\_total{pod=~"nginx.\*"}[1m])

* Click **Apply** to visualize CPU usage.
* Add more panels for memory, request rates, pod counts, etc.

1. Optional: Import ready-made dashboards from Grafana Dashboard Directory.

✅ After following this guide, you will have:

* Jenkins CI/CD building Docker images and updating Git
* Argo CD auto-deploying to Kubernetes
* Prometheus collecting metrics
* Grafana visualizing metrics with dashboards

If you want, I can create a **ready-to-use ZIP** containing:

* Dockerfile, index.html, Jenkinsfile
* Sample Kubernetes manifests (Deployment, Service, HPA, VPA, KEDA)
* Argo CD application manifest

…so you can **import and run everything immediately on your local machine**.

Do you want me to prepare that ZIP?

**These are the actual steps for the project:**

**container\_cpu\_usage\_seconds\_total{namespace="default"} : Grafana query  
  
minikube service nginx-service --url**

**$ kubectl port-forward svc/nginx-service 8081:80**

[**http://localhost:8081**](http://localhost:8081/)

**Complete Step-by-Step Guide: Setting Up a Jenkins-Nginx-Kubernetes CI/CD Pipeline with Argo CD, Prometheus, and Grafana**

This guide walks you through building a full CI/CD pipeline for a simple Nginx web server deployed to Kubernetes. It includes automated builds with Jenkins, GitOps deployments via Argo CD, and monitoring with Prometheus & Grafana. I'll expand on the provided outline with **exact commands**, **OS-specific notes** (assuming macOS, Linux, or Windows; adjust as needed), **troubleshooting tips**, and **verified official links** (fetched from current sources as of October 2025).

**Prerequisites Assumptions:**

* You're on macOS, Linux (Ubuntu/Debian), or Windows (with WSL2 recommended for Docker/K8s).
* Basic terminal familiarity.
* Admin/sudo access.
* At least 8GB RAM and 4 CPU cores for smooth Minikube operation.

If you encounter errors, check logs (e.g., kubectl logs <pod>) and ensure ports aren't conflicting.

**Phase 1: Project Folder Setup**

This creates your local project structure.

1. **Open Terminal (or Git Bash/PowerShell on Windows).**
   * macOS/Linux: Use built-in Terminal.
   * Windows: Use Git Bash (install from [git-scm.com](https://git-scm.com)) or PowerShell.
2. **Create the project folder and subfolders:**

text

mkdir ~/jenkins-nginx-k8s-pipeline

cd ~/jenkins-nginx-k8s-pipeline

mkdir k8s argocd prometheus-grafana

touch Dockerfile index.html Jenkinsfile README.md

* + ~ expands to your home directory (e.g., /Users/yourname on macOS).
  + touch creates empty files.

1. **Open the folder in VS Code:**
   * Launch VS Code.
   * Go to **File → Open Folder** → Select ~/jenkins-nginx-k8s-pipeline.
   * Or from terminal: code ~/jenkins-nginx-k8s-pipeline (if VS Code is in your PATH).
2. **Verify folder structure:**

text

tree . # Install tree if needed: brew install tree (macOS) or apt install tree (Linux)

Expected output:

text

.

├── argocd

├── k8s

├── prometheus-grafana

├── Dockerfile

├── index.html

├── Jenkinsfile

└── README.md

**Troubleshooting:** If mkdir fails, ensure no spaces in path or use quotes: mkdir "~/folder with spaces".

**Phase 2: Install Prerequisites**

Install tools one by one. Verify each after installation.

**1. VS Code & Extensions**

* **Install VS Code:** Download from the [official site](https://code.visualstudio.com/download).
  + macOS: Drag to Applications.
  + Linux: .deb or .rpm package.
  + Windows: .exe installer.
* **Verify:** Run code --version in terminal.
* **Install Extensions:** Open VS Code → Extensions sidebar (Ctrl+Shift+X) → Search and install:
  + **Docker** (by Microsoft): For Dockerfiles and compose.
  + **Kubernetes** (by Microsoft): For YAML manifests.
  + **YAML** (by Red Hat): Syntax highlighting.
  + **GitLens** (by GitKraken): Git history visualization.

**2. Docker Desktop / Engine**

* **Install Docker Desktop:** Download from [official site](https://www.docker.com/products/docker-desktop/).
  + Enables GUI, Kubernetes integration, and local registry.
  + macOS/Windows: Follow installer; enable WSL2 on Windows.
  + Linux: Use package manager (e.g., sudo apt install docker.io).
* **Verify:**

text

docker --version

docker info

Expected: Version ~27.x, and info shows running daemon.

* **Optional: Run Local Docker Registry:**

text

docker run -d -p 5000:5000 --name registry registry:2

* + Access: <http://localhost:5000/v2/> (should return {}).

**Troubleshooting:** If Docker won't start, check virtualization (VT-x/AMD-V enabled in BIOS) or restart your machine.

**3. Kubernetes (Minikube or K3s)**

Use Minikube for simplicity (full K8s features). K3s is lighter for low-spec machines.

**Minikube (Recommended):**

* **Install:**
  + macOS: brew install minikube
  + Linux: curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 && sudo install minikube-linux-amd64 /usr/local/bin/minikube
  + Windows: choco install minikube (or download from [minikube.sigs.k8s.io](https://minikube.sigs.k8s.io/docs/start/)).
* **Start Cluster:**

text

minikube start --driver=docker

* + Uses Docker as driver; allocates 2 CPUs/4GB RAM by default.
* **Verify:**

text

kubectl get nodes

kubectl cluster-info

Expected: One node "Ready"; API server URL.

**Optional: Lightweight K3s:**

text

curl -sfL https://get.k3s.io | sh -

export KUBECONFIG=/etc/rancher/k3s/k3s.yaml # Add to ~/.bashrc

* Verify same as above.

**Troubleshooting:** If Minikube fails, try minikube delete and restart. Ensure Docker is running.

**4. kubectl CLI**

* **Install:** Follow [official guide](https://kubernetes.io/docs/tasks/tools/install-kubectl/).
  + macOS: brew install kubectl
  + Linux: curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl" && sudo install kubectl /usr/local/bin/
  + Windows: choco install kubernetes-cli
* **Verify:**

text

kubectl version --client

kubectl get nodes # Should show Minikube node

**5. Helm**

* **Install:** Follow [official guide](https://helm.sh/docs/intro/install/).
  + macOS: brew install helm
  + Linux: curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash
  + Windows: Download binary from [GitHub releases](https://github.com/helm/helm/releases).
* **Add Repos & Update:**

text

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

helm repo add grafana https://grafana.github.io/helm-charts

helm repo update

* **Verify:** helm version (should be ~3.15.x).

**6. Jenkins (Docker Option)**

* **Run Jenkins Container:**

text

docker run -d -p 8080:8080 -p 50000:50000 --name jenkins \

-v jenkins\_home:/var/jenkins\_home \

-v /var/run/docker.sock:/var/run/docker.sock \

jenkins/jenkins:lts

* + -v jenkins\_home:... persists data; create volume if needed: docker volume create jenkins\_home.
* **Access:** <http://localhost:8080>
* **Unlock:**

text

docker exec jenkins cat /var/jenkins\_home/secrets/initialAdminPassword

Paste the password.

* **Install Plugins:** During setup, select **Install suggested plugins**. Then, **Manage Jenkins → Manage Plugins → Available** → Search/install: Docker Pipeline, Git, Kubernetes Continuous Deploy, Pipeline.

**Troubleshooting:** If port 8080 conflicts, change to -p 8081:8080. Restart: docker restart jenkins.

**7. Argo CD**

* **Install:**

text

kubectl create namespace argocd

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

* **Port-Forward Server:**

text

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

* + Access: <https://localhost:8080> (accept self-signed cert).
* **Get Admin Password:**

text

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

* **Login:** Username: admin, Password: from above.
  + Change password in UI: User Info → Change Password.

**Troubleshooting:** If install fails, check kubectl get pods -n argocd for errors.

**8. Prometheus & Grafana**

* **Create Namespace:**

text

kubectl create namespace monitoring

* **Install Prometheus:**

text

helm install prometheus prometheus-community/prometheus -n monitoring

* **Install Grafana:**

text

helm install grafana grafana/grafana -n monitoring --set adminUser=admin --set adminPassword=admin

* **Access (Port-Forward):**
  + Grafana: kubectl port-forward svc/grafana -n monitoring 3000:80 → <http://localhost:3000> (admin/admin)
  + Prometheus: kubectl port-forward svc/prometheus-server -n monitoring 9090:80 → <http://localhost:9090>

**Troubleshooting:** Helm upgrades if needed: helm upgrade prometheus .... Verify: kubectl get pods -n monitoring.

**Phase 3: Project Files**

Create/edit files in VS Code. Save all in the root folder unless specified.

**1. Dockerfile (Root)**

text

FROM nginx:latest

COPY index.html /usr/share/nginx/html/index.html

EXPOSE 80

**2. index.html (Root)**

text

<!DOCTYPE html>

<html>

<head>

<title>Nginx CI/CD Pipeline</title>

</head>

<body>

<h1>Hello from Nginx CI/CD Pipeline!</h1>

</body>

</html>

**3. Jenkinsfile (Root)**

text

pipeline {

agent any

environment {

DOCKER\_IMAGE = "nginx-app:${BUILD\_NUMBER}" // Use build number for versioning

GIT\_REPO = "http://localhost:8081/your-git-repo/nginx-k8s.git" // Update with real Git URL

}

stages {

stage('Build Docker Image') {

steps { sh 'docker build -t ${DOCKER\_IMAGE} .' }

}

stage('Push Docker Image') {

steps {

sh 'docker tag ${DOCKER\_IMAGE} localhost:5000/${DOCKER\_IMAGE}'

sh 'docker push localhost:5000/${DOCKER\_IMAGE}'

}

}

stage('Update Git Repo') {

steps {

sh 'git clone ${GIT\_REPO} temp-repo' // Clone to temp

sh '''

cd temp-repo

# Assume you update a deployment.yaml with new image tag here

sed -i "s|image: .\*|image: localhost:5000/nginx-app:latest|g" k8s/nginx-deployment.yaml

git add .

git commit -m "Update image to ${BUILD\_NUMBER}"

git push origin main

'''

}

}

}

post {

success { echo 'Docker image pushed and Git updated. Argo CD will auto-sync.' }

always { sh 'rm -rf temp-repo' } // Cleanup

}

}

* **Note:** Update GIT\_REPO with a real Git repo (e.g., create one on GitLab/local Gitea). The sed command updates the image tag in manifests.

**4. Kubernetes Manifests (k8s/ Folder)**

Create these YAML files in ./k8s/:

* **nginx-deployment.yaml:**

text

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

spec:

replicas: 2

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: localhost:5000/nginx-app:latest # Updated by pipeline

ports:

- containerPort: 80

livenessProbe:

httpGet:

path: /

port: 80

initialDelaySeconds: 30

periodSeconds: 10

readinessProbe:

httpGet:

path: /

port: 80

initialDelaySeconds: 5

periodSeconds: 5

startupProbe:

httpGet:

path: /

port: 80

failureThreshold: 30

periodSeconds: 10

* **nginx-service.yaml:**

text

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

type: NodePort

ports:

- port: 80

targetPort: 80

nodePort: 30080 # Accessible via Minikube

selector:

app: nginx

* **hpa.yaml** (Horizontal Pod Autoscaler):

text

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: nginx-hpa

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: nginx-deployment

minReplicas: 2

maxReplicas: 10

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

averageUtilization: 50

* **vpa.yaml** (Vertical Pod Autoscaler; install VPA first: kubectl apply -f https://github.com/kubernetes/autoscaler/releases/download/vertical-pod-autoscaler-0.25.0/vertical-pod-autoscaler.yaml):

text

apiVersion: autoscaling.k8s.io/v1

kind: VerticalPodAutoscaler

metadata:

name: nginx-vpa

spec:

targetRef:

apiVersion: "apps/v1"

kind: Deployment

name: nginx-deployment

updatePolicy:

updateMode: "Auto"

* **keda-scaledobject.yaml** (Install KEDA first: helm repo add kedacore https://kedacore.github.io/charts && helm install keda kedacore/keda -n keda --create-namespace):

text

apiVersion: keda.sh/v1alpha1

kind: ScaledObject

metadata:

name: nginx-scaledobject

spec:

scaleTargetRef:

name: nginx-deployment

minReplicaCount: 2

maxReplicaCount: 10

triggers:

- type: cpu

metadata:

value: "70"

**5. Argo CD Application (argocd/ Folder)**

Create **nginx-app.yaml** in ./argocd/:

text

apiVersion: argoproj.io/v1alpha1

kind: Application

metadata:

name: nginx-app

namespace: argocd

spec:

project: default

source:

repoURL: 'http://your-git-server:3000/your-user/nginx-k8s.git' # Update with real Git URL

targetRevision: HEAD

path: k8s

destination:

server: 'https://kubernetes.default.svc'

namespace: default

syncPolicy:

automated:

prune: true

selfHeal: true

syncOptions:

- CreateNamespace=true

* Commit this to your Git repo and apply: kubectl apply -f argocd/nginx-app.yaml -n argocd.

**Note:** Set up a Git repo (e.g., local: git init nginx-k8s && cd nginx-k8s && git add . && git commit -m "init"). Push k8s/ folder there.

**Phase 4: Running the First CI/CD Pipeline**

1. **Start Cluster:**

text

minikube start --driver=docker

1. **Start Jenkins:**

text

docker start jenkins

* + Wait for it to be ready: <http://localhost:8080>.

1. **Set Up Jenkins Job:**
   * In Jenkins UI: **New Item → Pipeline → OK**.
   * Paste your Jenkinsfile into **Pipeline** script.
   * Save and **Build Now**.
2. **Argo CD Auto-Sync:**
   * In Argo CD UI: Watch "nginx-app" sync (green checkmark).
   * Deployment happens automatically.
3. **Verify Deployment:**

text

kubectl get deployments

kubectl get pods

kubectl get svc

Expected: nginx-deployment with 2/2 pods ready; nginx-service type NodePort.

1. **Access Nginx:**

text

minikube service nginx-service --url

* + Opens browser to your "Hello" page on port 30080.

1. **Verify Health Probes:**

text

kubectl get pods # Note pod name, e.g., nginx-deployment-abc-123

kubectl describe pod nginx-deployment-abc-123

* + Check Events/Conditions for probe successes.

1. **Test Autoscaling:**

text

kubectl get hpa

kubectl describe hpa nginx-hpa

kubectl get scaledobject # For KEDA

* + **Generate Load (in new terminal):**

text

kubectl run load-generator --image=busybox -i --tty --rm --restart=Never -- /bin/sh -c "while true; do wget -q -O- http://nginx-service; done"

* + - Watch: kubectl get hpa -w (replicas should increase on high CPU).
    - Stop: Ctrl+C in load terminal.

**Troubleshooting:** If sync fails, check Argo CD events. Ensure Git repo is accessible from Jenkins (add credentials).

**Phase 5: Access Metrics in Grafana**

1. **Open Grafana:** <http://localhost:3000> (admin/admin).
   * Port-forward if needed: See Phase 2.8.
2. **Add Prometheus Datasource:**
   * **Configuration → Data Sources → Add data source → Prometheus**.
   * URL: http://prometheus-server.monitoring.svc.cluster.local:80 (or http://localhost:9090 if port-forwarded).
   * **Save & Test** (should say "Data source is working").
3. **Create Dashboard:**
   * **Dashboards → New → Add new panel**.
   * **Query:** In Prometheus query field:

text

rate(container\_cpu\_usage\_seconds\_total{namespace="default",pod=~"nginx.\*"}[5m])

* + - Legend: {{pod}}, Unit: percent (0.0-100.0).
  + **Apply** → Visualize CPU spikes.
  + **Add Panels:**
    - Memory: container\_memory\_working\_set\_bytes{namespace="default",pod=~"nginx.\*"}
    - Pod Count: count(up{job="nginx"})
    - Requests: histogram\_quantile(0.95, rate(nginx\_http\_requests\_total[5m])) (if Nginx metrics enabled).

1. **Optional: Import Dashboards:**
   * Go to **Dashboards → Import**.
   * ID: 315 (Node Exporter) or 1860 (Kubernetes Cluster Monitoring) from [grafana.com/dashboards](https://grafana.com/grafana/dashboards/).
   * Select Prometheus datasource.

**Troubleshooting:** No data? Ensure Prometheus scrapes Nginx (add annotations to deployment: prometheus.io/scrape: "true"). Wait 2-5 mins for metrics.

**✅ Final Checklist & Next Steps**

* Jenkins builds and pushes image on trigger.
* Argo CD deploys changes automatically.
* Nginx accessible via Minikube service.
* Probes pass; autoscalers respond to load.
* Grafana shows real-time CPU/memory dashboards.

**Teardown (if needed):**

text

minikube stop && minikube delete

docker stop jenkins && docker rm jenkins

kubectl delete namespace argocd monitoring

helm uninstall prometheus grafana -n monitoring

**Enhancements:**

* Secure Jenkins/Argo with HTTPS.
* Use a cloud Git provider (GitHub/GitLab).
* Add unit tests to Jenkinsfile.
* Integrate Slack notifications.

If stuck on a step, share error output! This setup gives you a production-like pipeline in ~1-2 hours.