AY: 2024/2025

PROJECT

ARIR86 - Delivery Management, Devops et Pipeline

RS1-RS2-RS3

Teachers: Mr BOUZAYEN Abdelbaki and Mr MLAYAH Najmeddine

Building a DevOps CI/CD Pipeline Locally

GitHub, Jenkins, Maven, SonarQube, Docker, DockerHub, ArgoCD, Helm, Kubernetes, Prometheus, Grafana, Filebeat, OpenSearch, and Kibana

General Objectives:

- Build a complete CI/CD pipeline in a local environment, integrating various DevOps tools to automate the development and deployment processes of applications.

- Understand and use key DevOps tools such as GitHub, Jenkins, Maven, SonarQube, Docker, DockerHub, ArgoCD, Helm, Kubernetes, Prometheus, Grafana, Filebeat, OpenSearch, and Kibana, configuring them to work together seamlessly
within a CI/CD pipeline.

Project Introduction:

This project implements a complete DevOps CI/CD pipeline in a local environment. The CI part is implemented using Docker and Docker Compose, providing isolated environments for Jenkins, SonarQube, and other necessary tools. DockerHub is used as the container registry for managing Docker images. For code quality analysis and build management, Maven and SonarQube are integrated with Jenkins, ensuring a smooth and efficient continuous integration process.
The CD part is managed with Kubernetes running in Minikube, allowing for local orchestration of containers. Deployment to Kubernetes is automated using ArgoCD, which facilitates smooth rollouts and rollbacks. Monitoring and logging are handled using Prometheus, Grafana, Filebeat, OpenSearch, and Kibana, all installed via Helm charts for ease of configuration. This setup provides comprehensive observability and logging capabilities, making it ideal for testing new features and fixes.

Prerequisites:

Before getting started with this DevOps CI/CD project, ensure the following tools are installed and properly configured:

- **Docker**: For containerization.
- **Docker Compose**: To manage multi-container applications.
- **DockerHub**: For storing and managing Docker images.
- **kubectl**: For interacting with the Kubernetes cluster.
- **Minikube**: To set up a local Kubernetes environment.
- **Helm**: For managing Kubernetes applications.
- **Git**: For version control.
- **ArgoCD CLI**: (Optional) For managing deployments through the command line.
- **GitHub Account**: For repository management.

Implementation steps:

1. **Fork and Clone below Git repository:**
   https://github.com/abdelbaki-bouzaienne/devops-demo-project

2. **Open terminal, cd to project directory and use docker-compose to bring up CI environment:**

```
cd devops-demo-project/ci
docker-compose up -d
```

3. **Open below URL to access Jenkins:**
http://localhost:8010/

4. **Access Jenkins password, login and install suggested plugins:**

```
docker ps
docker exec -it jenkins bin/bash
cat /var/jenkins_home/secrets/initialAdminPassword
```

Getting Started
Unlock Jenkins
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:
/var/jenkins_home/secrets/initialAdminPassword
Please copy the password from either location and paste it below.
Administrator password
Continue

Getting Started
Customize Jenkins
Plugins extend Jenkins with additional features to support many different needs.
Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.
Jenkins 2.452.2

Getting Started
Username
Password
Confirm password
Full name
E-mail address
Jenkins 2.452.2
Skip and continue as admin
Save and Continue

Getting Started
Instance Configuration
Jenkins URL:
http://localhost:8010/
The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.
The proposed default value shown is not saved yet and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.
Jenkins 2.452.2
Not now
Save and Finish

5. Install below suggested plugins and restart Jenkins container:

6. **Open SonarQube using below URL and create token:**

http://localcalhost:9000

username: admin

password: admin

7. Login to your GitHub account and generate personal access token:

8. Add credentials for GitHub and SonarQube using secret text and DockerHub using username with password:

9. Login to Jenkins and create new item:

Enter an item name
demo-java-app-ci
» Required field
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
OK
Multibranch Pipeline

Pipeline
Definition
Pipeline script from SCM
SCM
Git
Repositories
Repository URL
https://github.com/deepakkr35/devops-demo-project.git
Credentials
- none -
+ Add
Advanced
Add Repository
Branches to build
Branch Specifier (blank for 'any')

10. Build the pipeline with V1.0 parameter:

11. Docker image will be created and GitHub helm/app/values.yaml will be updated with appropriate tag:

```
7     image:
8       githubID: deepakkr35
9       repository: demo-java-app
10      pullPolicy: IfNotPresent
11      # Overrides the image tag whose default is the chart appVersion.
    -   tag:
12  +   tag: V1.0
13
```

12. **Start Kubernets cluster (Minikube) in docker driver mode:**

```
minikube start --driver=docker --cpus 4 --memory 10240
```

```
😄  minikube v1.33.1 on Darwin 14.6.1 (arm64)
✨  Using the docker driver based on user configuration
📌  Using Docker Desktop driver with root privileges
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.44 ...
🔥  Creating docker container (CPUs=4, Memory=10240MB) ...
🐳  Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

13. **Create namespaces for installing application, logging and monitoring:**

```
kubectl create ns app
kubectl create ns logging
kubectl create ns monitoring
```

14. **Edit app namespace and add following label for ArgoCD:**

```
kubectl edit ns app
```

```
labels:
  kubernetes.io/metadata.name: app
  argocd.argoproj.io/managed-by: argocd
```

15. **Install Graphana and Promotheus in monitoring namespce using Helm:** (port-forward is required to access UI, as minikube is running in Docker driver mode)

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-
helm install prometheus prometheus-community/prometheus -n monitoring
kubectl port-forward service/prometheus-server 9030:80 -n monitoring
```

http://localhost:9030/

```
helm repo add grafana https://grafana.github.io/helm-charts
helm install grafana grafana/grafana -n monitoring
kubectl get secret --namespace monitoring grafana -o jsonpath="{.data.admin-pas
kubectl port-forward service/grafana 9040:80 -n monitoring
```

http://localhost:9040/

16. Configure Graphana to use Promoetheus as data source and import required dashhboards:

Search or jump to...
Home › Dashboards › Import dashboard
Home
Starred
Dashboards
Explore
Alerting
Connections
Add new connection
Data sources
Administration
Import dashboard
Import dashboard from file or Grafana.com
Upload dashboard JSON file
Drag and drop here or click to browse
Accepted file types: .json, .txt
Find and import dashboards for common applications at grafana.com/dashboards
315
Load
Import via dashboard JSON model
{
"title": "Example - Repeating Dictionary variables",
"uid": "_0HnEoN4z",
"panels": [...]
...
}
Load
Cancel

Search or jump to...
Home › Dashboards › Import dashboard
Home
Starred
Dashboards
Explore
Alerting
Connections
Add new connection
Data sources
Administration
Import dashboard
Import dashboard from file or Grafana.com
Importing dashboard from Grafana.com
Published by
Instrumentisto Team
Updated on
2023-03-21 00:40:51
Options
Name
Kubernetes cluster monitoring (via Prometheus)
Folder
Dashboards
Unique identifier (UID)
The unique identifier (UID) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.
Change uid
Prometheus
prometheus
Import
Cancel

17. Install Elasticsearch, Kibana and Filebeat in logging namespace using Helm and port forward to access Elasticsearch and Kibana UIs:

```
cd devops-demo-project/helm
helm install elasticsearch ./logging/elasticsearch
helm install kibana ./logging/kibana
helm install filebeat ./logging/filebeat

kubectl port-forward service/es-master-service 9010:9200 -n logging
kubectl port-forward service/kibana-service 9020:5601 -n logging
```

http://localhost:9010/

localhost:9010

Pretty-print


```
{
  "name" : "es-6b48497db7-gmgsf",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "XkQAwLSsSYGT5iM9XpdDgg",
  "version" : {
    "number" : "7.16.2",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "2b937c44140b6559905130a8650c64dbd0879cfb",
    "build_date" : "2021-12-18T19:42:46.604893745Z",
    "build_snapshot" : false,
    "lucene_version" : "8.10.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```


http://localhost:9020/

18. Install ArgoCD in Kubernetes cluster:

```
cd devops-demo-project/cd/argocd
curl -sL https://github.com/operator-framework/operator-lifecycle-manager/relea
kubectl get csv -n operators
kubectl create ns argocd
kubectl apply -f argocd-basic.yaml
echo $(kubectl get secret example-argocd-cluster -n argocd -o jsonpath="{.data.
minikube service example-argocd-server -n argocd
```

```
|-----------|-----------------------|-------------|-------------|
| NAMESPACE |         NAME          | TARGET PORT |     URL     |
|-----------|-----------------------|-------------|-------------|
| argocd    | example-argocd-server |             | No node port |
|-----------|-----------------------|-------------|-------------|
😿  service argocd/example-argocd-server has no node port
❗  Services [argocd/example-argocd-server] have type "ClusterIP" not meant to be exposed, however for local development minikube allows you to access this !
🏃  Starting tunnel for service example-argocd-server.
|-----------|-----------------------|-------------|------------------------|
| NAMESPACE |         NAME          | TARGET PORT |          URL           |
|-----------|-----------------------|-------------|------------------------|
| argocd    | example-argocd-server |             | http://127.0.0.1:49176 |
|           |                       |             | http://127.0.0.1:49177 |
|-----------|-----------------------|-------------|------------------------|
[argocd example-argocd-server  http://127.0.0.1:49176
http://127.0.0.1:49177]
❗  Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

19. Open ArgoCD URL and create application to deploy, using password retrieved earlier:

Let's get stuff deployed!
argo
Username
admin
Password
SIGN IN

argo
v2.12.1+26b2039
Applications
Settings
User Info
Documentation
CREATE
CANCEL
EDIT AS YAML
GENERAL
Application Name
demo-java-app
Project Name
default
SYNC POLICY
Automatic
PRUNE RESOURCES
SELF HEAL
SET DELETION FINALIZER
SYNC OPTIONS
SKIP SCHEMA VALIDATION
PRUNE LAST
RESPECT IGNORE DIFFERENCES
AUTO-CREATE NAMESPACE
APPLY OUT OF SYNC ONLY
SERVER-SIDE APPLY

argo
v2.12.1+26b2039
Applications
Settings
User Info
Documentation
+ NEW APP
CREATE
CANCEL
SOURCE
Repository URL
https://github.com/deepakkr35/devops-demo-project.git
GIT
Revision
HEAD
Branches
Path
helm/app
DESTINATION
Cluster URL
https://kubernetes.default.svc
URL
Namespace
app
Helm
HELM

20. port-forward app to access through UI:

```
kubectl port-forward service/demo-java-app-service 8080:80 -n app
```

http://localhost:8080/

21. **Check application logs rolling in Kibana:**

22. **Make code changes and commit (for testing CI/CD pipeline):**

demo-java-app/src/main/java/com/deepak/demo/DemoApplication.java

```
model.addAttribute( s: "title",  o: "This is a Spring Boot demo app version v2.0");
```

demo-java-app/src/main/resources/templates/index.html

```
    background: linear-gradient(135deg, #6e45e2, #88d3ce);
```

23. Build pipeline in Jenkins:

24. Once CI pipeline is complete, Dockerhub and GitHub will be update:

```
@@ -9,7 +9,7 @@ image:
9       repository: demo-java-app
10      pullPolicy: IfNotPresent
11      # Overrides the image tag whose default is the chart appVersion.
-       tag: V1.0
12  +   tag: V2.0
13
14    appName: demo-java-app
```

25. ArgoCD will deploy application changes:

26. **Access updated application:**

27. **Optional step** (Cleanup of resources)

```
cd devops-demo-project/ci
docker-compose down

minikube stop
minikube delete
```