

PROJECT

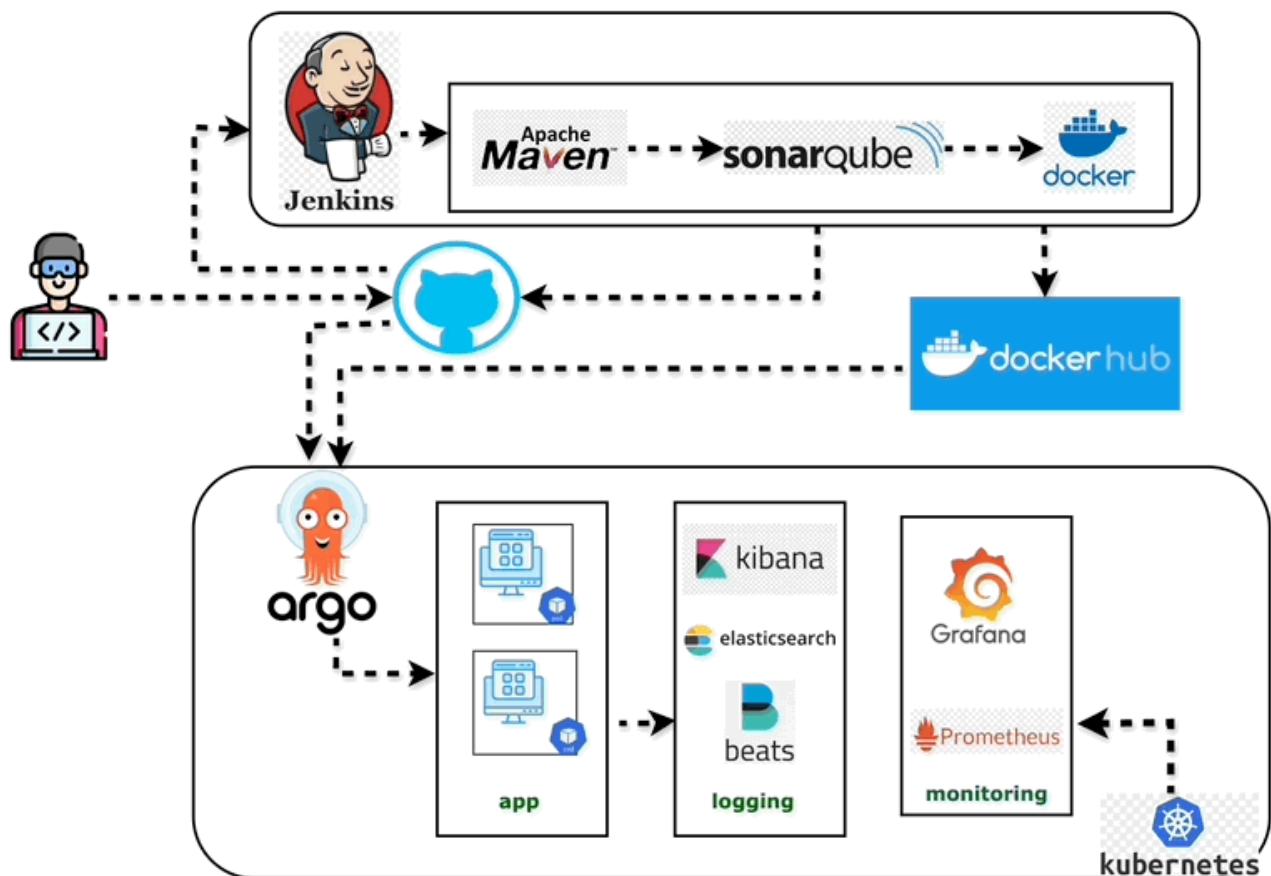
ARIR86 - Delivery Management, Devops et Pipeline

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:

0. Install the DevOps tools on your VM via the script in this guide:

<https://github.com/abdelbaki-bouzaienne/devops-demo-project/blob/main/Preparation-script-devops-tools.md>

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/>

The screenshot shows the 'Unlock Jenkins' step of the Jenkins setup wizard. At the top left, it says 'Getting Started'. The main title is 'Unlock Jenkins'. Below it, a message states: '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'. It then asks the user to 'Please copy the password from either location and paste it below.' A text input field is provided for this purpose, labeled 'Administrator password'. In the bottom right corner of the page, there is a blue 'Continue' button.

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

X

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.

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:

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

Getting Started

Jenkins is ready!

You have skipped the **setup of an admin user**.

To log in, use the username: "admin" and the administrator password you used to access the setup wizard.

Your Jenkins setup is complete.

[Start using Jenkins](#)

Jenkins 2.452.2

The screenshot shows the Jenkins dashboard. At the top, there's a navigation bar with the Jenkins logo, a search bar, and a user dropdown for 'admin'. Below the header, the page title is 'Welcome to Jenkins!'. A sub-header says 'This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.' There are two main sections: 'Start building your software project' (with a 'Create a job' button) and 'Set up a distributed build' (with links for 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'). On the left, there are two dropdown menus: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing '1 Idle' and '2 Idle'). The bottom right corner of the dashboard shows 'REST API' and 'Jenkins 2.452.2'.

5. Install below suggested plugins and restart Jenkins container:

Plugins

Search available plugins

Install

Name	Released
Pipeline: Stage View 2.34	11 mo ago
User Interface	
Pipeline Stage View Plugin.	
Docker Pipeline 580.vc0c340686b_54	4 mo 23 days ago
pipeline DevOps Deployment docker	
Build and use Docker containers from pipelines.	
SonarQube Scanner 2.17.2	7 mo 26 days ago
External Site/Tool Integrations Build Reports	
This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.	
JavaMail API 1.6.2-10	4 mo 21 days ago
Library plugins (for use by other plugins)	
This plugin provides the JavaMail API for other plugins.	
Pipeline: REST API 2.34	11 mo ago
User Interface	

Plugins

SSH Build Agents	Success
Matrix Authorization Strategy	Success
PAM Authentication	Success
LDAP	Success
Email Extension	Success
Mailer	Success
Theme Manager	Success
Dark Theme	Success
Loading plugin extensions	Success
Pipeline: REST API	Success
Pipeline: Stage View	Success
Authentication Tokens API	Success
Docker Commons	Success
Docker Pipeline	Success
SonarQube Scanner	Success
Loading plugin extensions	Success

→ Go back to the top page
(you can start using the installed plugins right away)

→ Restart Jenkins when installation is complete and no jobs are running

REST API Jenkins 2.452.2

6. Open SonarQube using below URL and create token:

<http://localhost:9000>

username: admin

password: admin

Tokens

If you want to enforce security by not providing credentials of a real SonarQube user to run your code scan or to invoke web services, you can provide a User Token as a replacement of the user login. This will increase the security of your installation by not letting your analysis user's password going through your network.

Generate Tokens

Name	Type	Expires in
jenkins	User Token	30 days

Generated Tokens

Name	Type	Project	Last use	Created	Expiration
No tokens					

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

Personal access tokens (classic)

Generate new token ▾

Tokens you have generated that can be used to access the [GitHub API](#).

My Mac Token — admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, Last used within the last week [Delete](#)
 admin:public_key, admin:repo_hook, admin:ssh_signing_key, audit_log, codespace, copilot, delete:packages,
 delete_repo, gist, notifications, project, repo, user, workflow, write:discussion, write:packages
 ⚠️ This token has no expiration date.

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

© 2024 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact](#) [Manage cookies](#) [Do not share my personal information](#)

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

What's this token for?

Expiration *
 The token will expire on Mon, Nov 11 2024

Select scopes
 Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events
<input checked="" type="checkbox"/> workflow	Update GitHub Action workflows
<input checked="" type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input checked="" type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input checked="" type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> manage_runners:org	Manage org runners and runner groups
<input checked="" type="checkbox"/> admin:public_key	Full control of user public keys

The screenshot shows the GitHub 'Personal access tokens' page under the 'Tokens (classic)' tab. A new token is being created for 'Jenkins'. The token is set to expire in 30 days (Nov 11, 2024). The 'Select scopes' section lists various GitHub permissions. Several scopes are checked, including 'repo', 'workflow', 'write:packages', 'delete:packages', 'admin:org', and 'admin:public_key'. Other scopes like 'repo:status', 'repo_deployment', 'public_repo', 'repo:invite', 'security_events', 'read:packages', 'read:org', and 'manage_runners:org' are also listed.

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

The screenshot shows the Jenkins 'Global credentials (unrestricted)' page. A new credential is being created with the following details:

- Kind:** Secret text
- Scope:** Global (Jenkins, nodes, items, all child items, etc)
- Secret:** (Redacted)
- ID:** github
- Description:** (Redacted)

A blue 'Create' button is at the bottom left.

New credentials

Kind

Username with password

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

Username ?
deepakkr35

Treat username as secret ?

Password ?

ID ?
dockerhub

Create

Global credentials (unrestricted)

[+ Add Credentials](#)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
github	github	Secret text	
dockerhub	deepakkr35/*****	Username with password	
sonarqube	sonarqube	Secret text	

Icon: S M L

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.

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') ?

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

/main

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add ▾

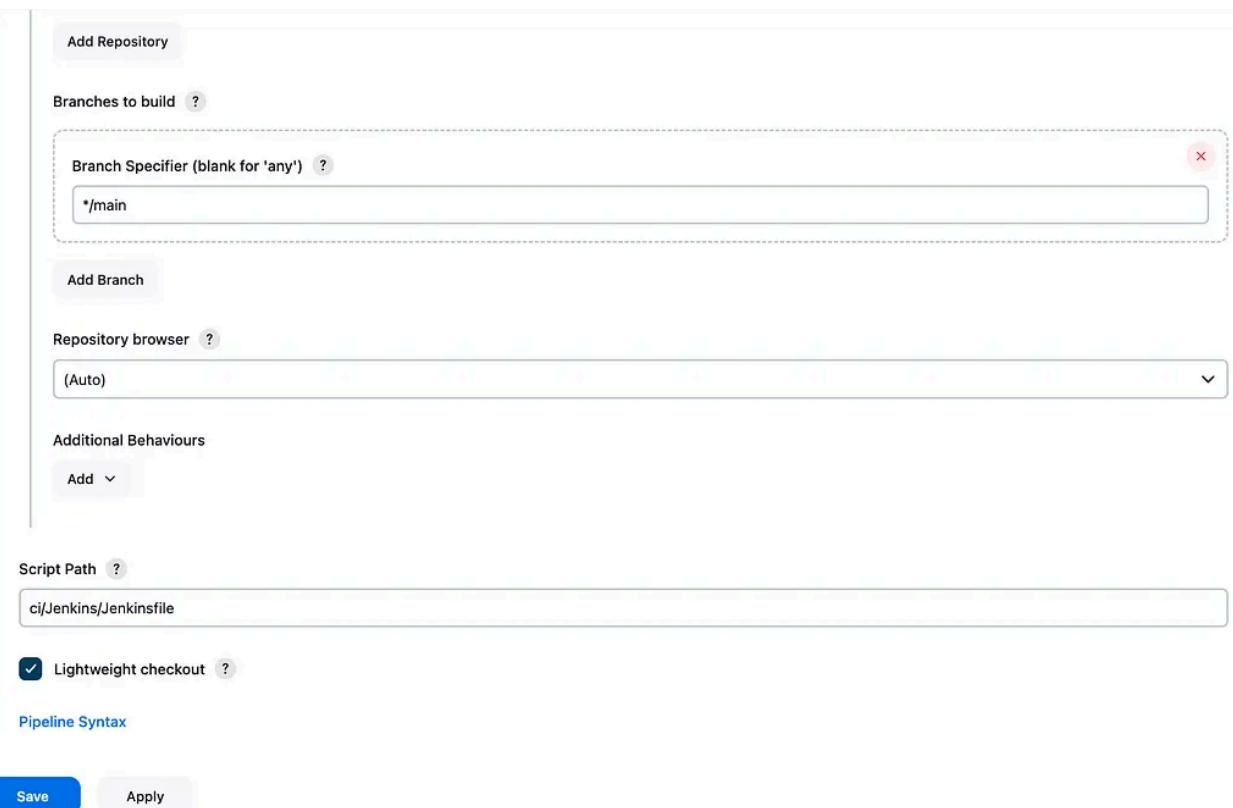
Script Path ?

ci/Jenkinsfile

Lightweight checkout ?

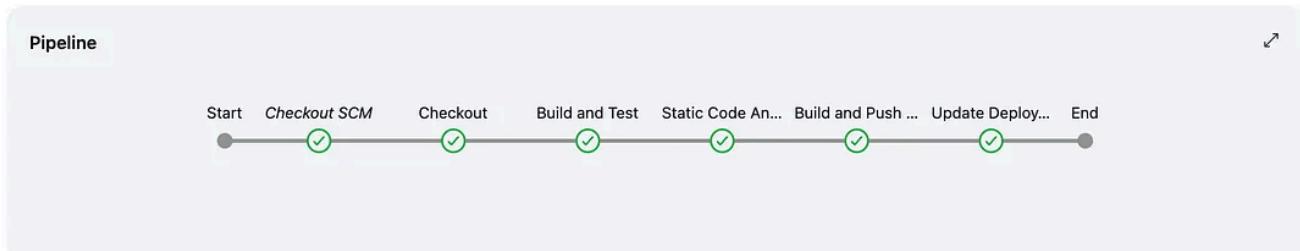
Pipeline Syntax

Save Apply



10. Build the pipeline with V1.0 parameter:

Build #1



Filters

Quality Gate

- Passed: 1
- Failed: 0

Reliability (Bug) Rating

- A rating: 1
- B rating: 0
- C rating: 0
- D rating: 0
- E rating: 0

Security (Vulnerability) Rating

- A rating: 1
- B rating: 0
- C rating: 0
- D rating: 0
- E rating: 0

1 project(s)

Perspective: Overall Status | Sort by: Name

Last analysis: 1 minute ago

Bugs: 0 A | Vulnerabilities: 0 A | Hotspots Reviewed: 1 A | Code Smells: 4 A | Coverage: 0.0% | Duplications: 0.0% | Lines: 95 XS XML, Java

1 of 1 shown

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

```

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

```

deepakkr35 / Repositories / demo-java-app / Tags

Using 0 of 1 private repositories

General Tags Builds Collaborators Webhooks Settings

Sort by Newest Filter Tags

TAG	Digest	OS/ARCH	Last pull	Compressed Size
V1.0	b705597095b2	linux/arm64/v8	---	249.12 MB

12. Start Kubernetes 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 Prometheus 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

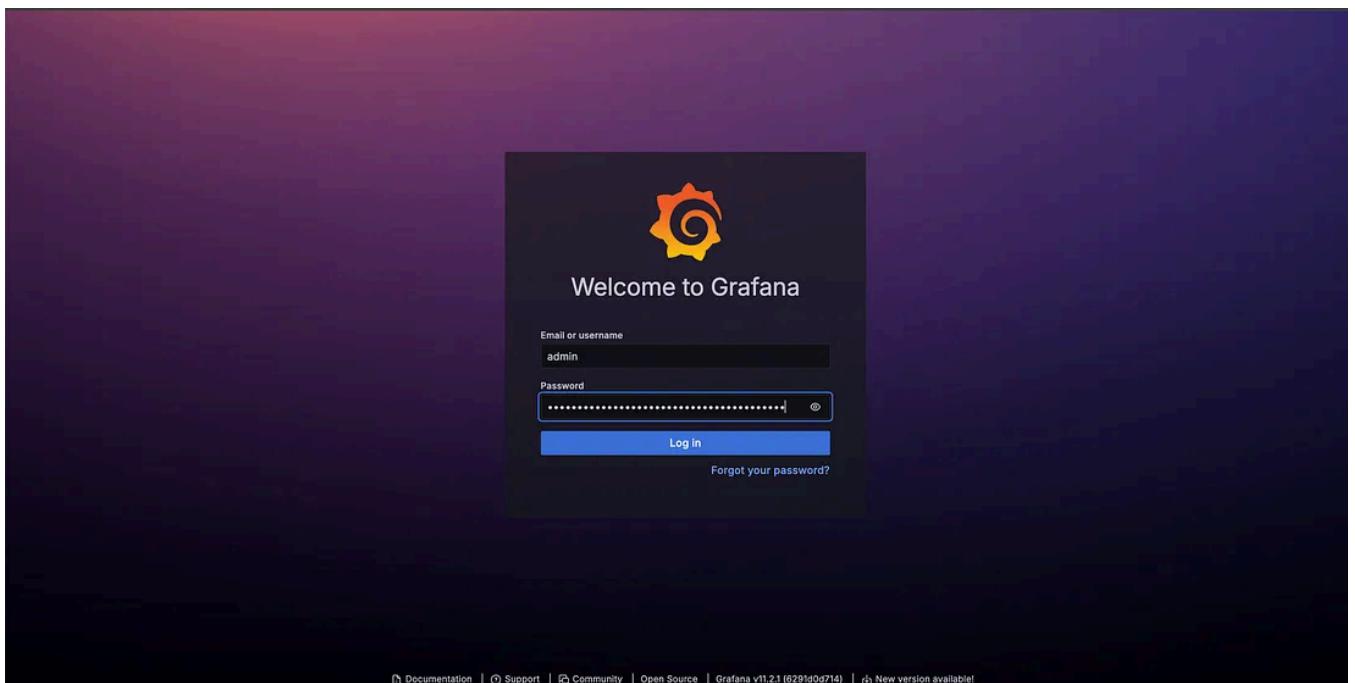
```

<http://localhost:9030/>

The screenshot shows the Prometheus UI interface. At the top, there's a navigation bar with links for Prometheus, Alerts, Graph, Status, and Help. Below the navigation is a toolbar with checkboxes for "Use local time", "Enable query history", "Enable autocomplete" (which is checked), "Enable highlighting" (which is checked), and "Enable linter". A search bar labeled "Expression (press Shift+Enter for newlines)" is followed by an "Execute" button. Below the search bar, there are tabs for "Table" and "Graph", with "Graph" being the active tab. A timestamp "Evaluation time" is shown between two arrows. A message "No data queried yet" is displayed. In the bottom right corner, there are "Add Panel" and "Remove Panel" buttons.

```
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-password}" | base64 --decode > grafana-admin-password.txt
kubectl port-forward service/grafana 9040:80 -n monitoring
```

<http://localhost:9040/>



16. Configure Graphana to use Prometheus as data source and import required dashhhboards:

Welcome to Grafana

Basic

The steps below will guide you to quickly finish setting up your Grafana installation.

TUTORIAL DATA SOURCE AND DASHBOARDS

Grafana fundamentals

Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.

DATA SOURCES

Add your first data source

DASHBOARDS

Create your first dashboard

Learn how in the docs [Remove this panel](#)

Dashboards

Starred dashboards

Recently viewed dashboards

Latest from the blog

Oct 11 Key Prometheus concepts every Grafana user should know

Prometheus has become an essential technology in the world of monitoring and observability. I've been aware of its importance for some time, but as a performance engineer, my experience with Prometheus had been limited to using it to store some metrics and visualize them in Grafana. Being a Grafanista, I felt I should dig deeper into Prometheus, knowing it had much more to offer than just being a place to throw performance test results.

Oct 10

Home > Connections > Data sources > prometheus

Name: prometheus Default:

Before you can use the Prometheus data source, you must configure it below or in the config file. For detailed instructions, [view the documentation](#).

Fields marked with * are required

Connection

Prometheus server URL:

Authentication

Authentication methods

Choose an authentication method to access the data source

No Authentication

TLS settings

Additional security measures that can be applied on top of authentication

Add self-signed certificate TLS Client Authentication Skip TLS certificate validation

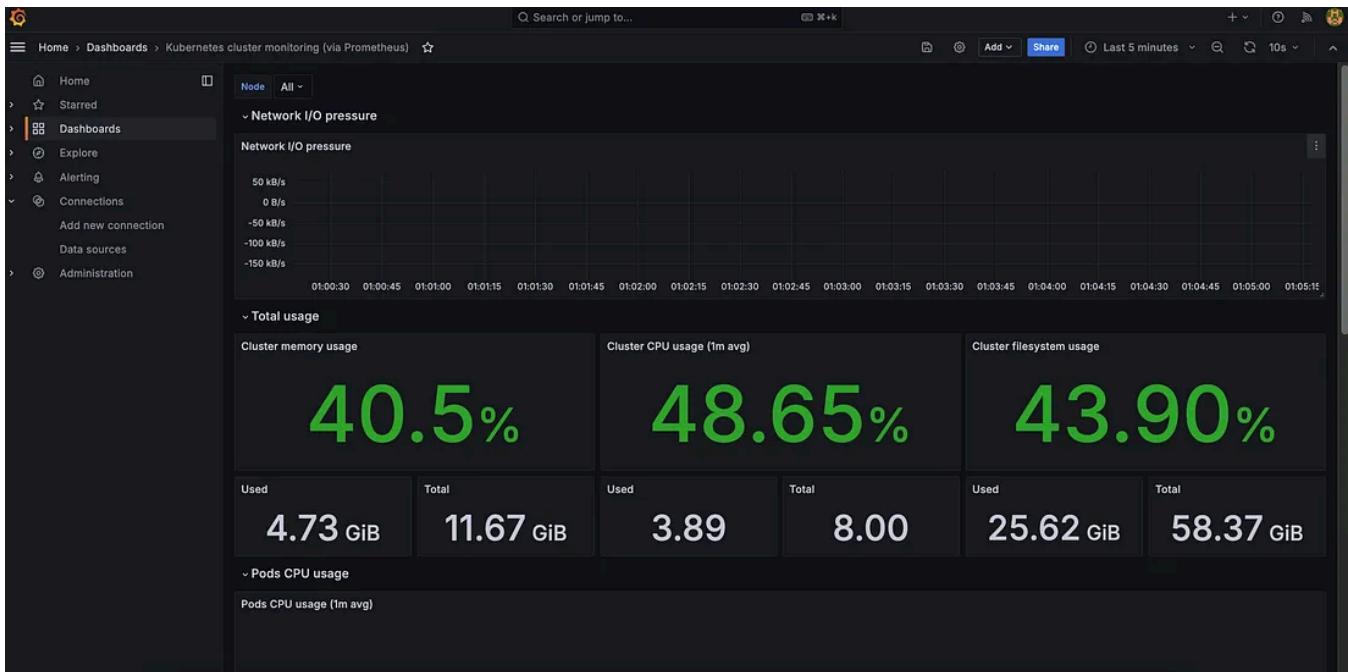
HTTP headers

The screenshot shows the 'Import dashboard' screen in Grafana. On the left, a sidebar menu includes 'Home', 'Starred', 'Dashboards' (which is selected), 'Explore', 'Alerting', 'Connections', 'Data sources', and 'Administration'. The main area has a title 'Import dashboard' and a sub-instruction 'Import dashboard from file or Grafana.com'. Below this is a large text input field labeled 'Upload dashboard JSON file' with a placeholder 'Drag and drop here or click to browse' and a note 'Accepted file types: .json, .txt'. At the bottom of this field is a 'Load' button. Below the input field is another section titled 'Import via dashboard JSON model' containing a JSON snippet:

```
{
  "title": "Example - Repeating Dictionary variables",
  "uid": "...OhInEoN4z",
  "panels": [...]
}
```

At the bottom of this section are two buttons: 'Load' (highlighted in blue) and 'Cancel'.

This screenshot shows the 'Importing dashboard from Grafana.com' screen. The left sidebar is identical to the previous one. The main area starts with the title 'Import dashboard' and the import instruction. It then displays information about the imported dashboard: 'Published by Instrumentisto Team' and 'Updated on 2023-03-21 00:40:51'. Below this is a 'Options' section with fields for 'Name' (set to 'Kubernetes cluster monitoring (via Prometheus)') and 'Folder' (set to 'Dashboards'). A note about 'Unique identifier (UID)' explains that it's used for identifying dashboards across multiple installations. There is a 'Change uid' button. Below this is a 'Prometheus' dropdown menu containing the entry 'prometheus'. At the bottom are 'Import' and 'Cancel' buttons.

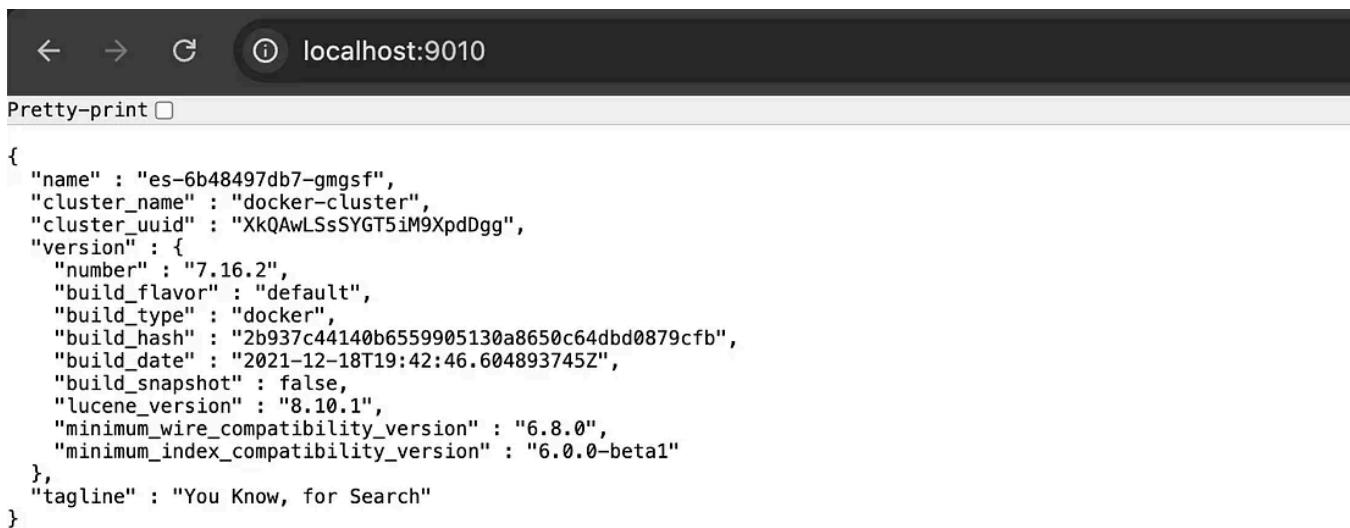


17. Install Elasticsearch, Kibana and Filebeat in logging namespace using Helm and port forward to access Elasticsearch and Kibana UIs:

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

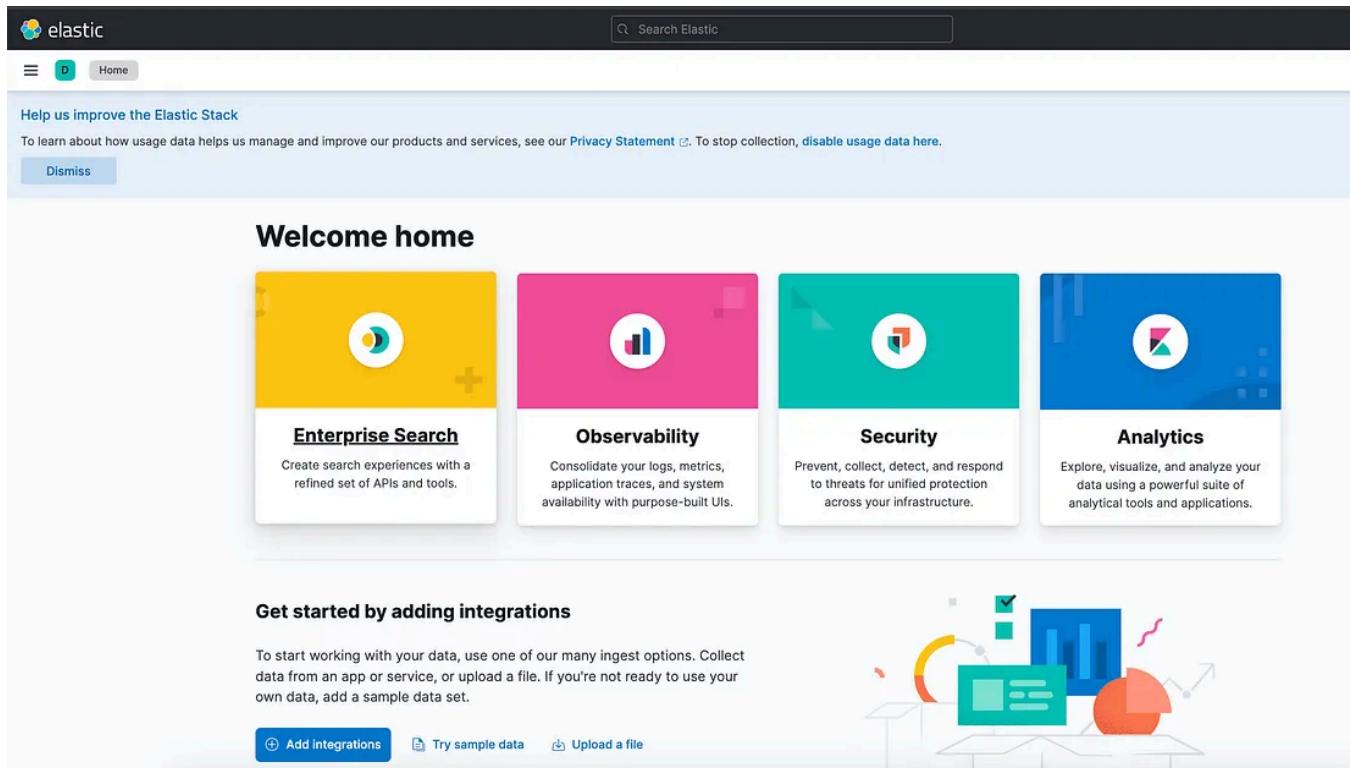
kubectl port-forward service/es-master-service 9010:9200 -n logging
kubectl port-forward service/kibana-service 9020:5601 -n logging
```

<http://localhost:9010/>



```
{
  "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/>



The screenshot shows the Elastic Stack landing page. At the top, there's a navigation bar with the Elastic logo, a search bar, and a 'Home' button. A notification bar below it encourages users to help improve the stack and provides a 'Dismiss' button. The main section features a 'Welcome home' heading and four colored cards representing different services: Enterprise Search (yellow), Observability (pink), Security (teal), and Analytics (blue). Each card has an icon and a brief description. Below these cards is a section titled 'Get started by adding integrations' with three buttons: '+ Add integrations', 'Try sample data', and 'Upload a file'. To the right of this text is a decorative graphic of a dashboard with charts and graphs.

The screenshot shows the Elasticsearch interface with a search bar containing 'filebeat-*'. Below the search bar is a 'Change index pattern' dropdown set to 'filebeat-*'. A message box in the center says 'No results match your search criteria' with a magnifying glass icon.

18. Install ArgoCD in Kubernetes cluster:

```
cd devops-demo-project/cd/argocd
curl -sL https://github.com/operator-framework/operator-lifecycle-manager/releases/latest/download/operator-lifecycle-manager.yaml | kubectl apply -f -
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.token}' | base64 --decode) > ./argocd.token
minikube service example-argocd-server -n argocd
```

The terminal output shows the following steps:

- cd devops-demo-project/cd/argocd
- curl -sL https://github.com/operator-framework/operator-lifecycle-manager/releases/latest/download/operator-lifecycle-manager.yaml | kubectl apply -f -
- 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.token}' | base64 --decode) > ./argocd.token
- minikube service example-argocd-server -n argocd

Warnings and notes in the terminal:

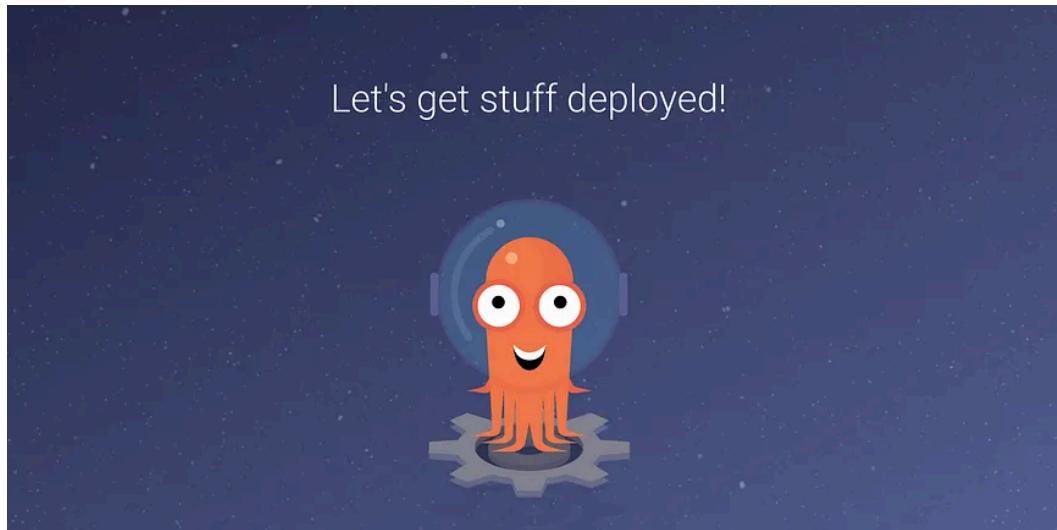
- service argocd/example-argocd-server has no node port
- Services [argocd/example-argocd-server] have type "ClusterIP" not meant to be exposed, however for local development minikube allows you to access this !
- Starting tunnel for service example-argocd-server.

NAMESPACE	NAME	TARGET PORT	URL
argocd	example-argocd-server		No node port

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:



GENERAL

Application Name: demo-java-app

Project Name: default

SYNC POLICY

Automatic

PRUNE RESOURCES

SELF HEAL

SET DELETION FINALIZER

SYNC OPTIONS

SKIP SCHEMA VALIDATION AUTO-CREATE NAMESPACE

PRUNE LAST APPLY OUT OF SYNC ONLY

RESPECT IGNORE DIFFERENCES SERVER-SIDE APPLY

SOURCE

Repository URL: <https://github.com/deepakkr35/devops-demo-project.git>

Revision: HEAD

Path: helm/app

DESTINATION

Cluster URL: <https://kubernetes.default.svc>

Namespace: app

Helm

HELM

demo-java-app

Project: default

Labels:

Status: ○ Progressing ✓ Synced

Repository: [https://github.com/deepakkr35/devops-demo-project...](https://github.com/deepakkr35/devops-demo-project)

Target Revisi...: HEAD

Path: helm/app

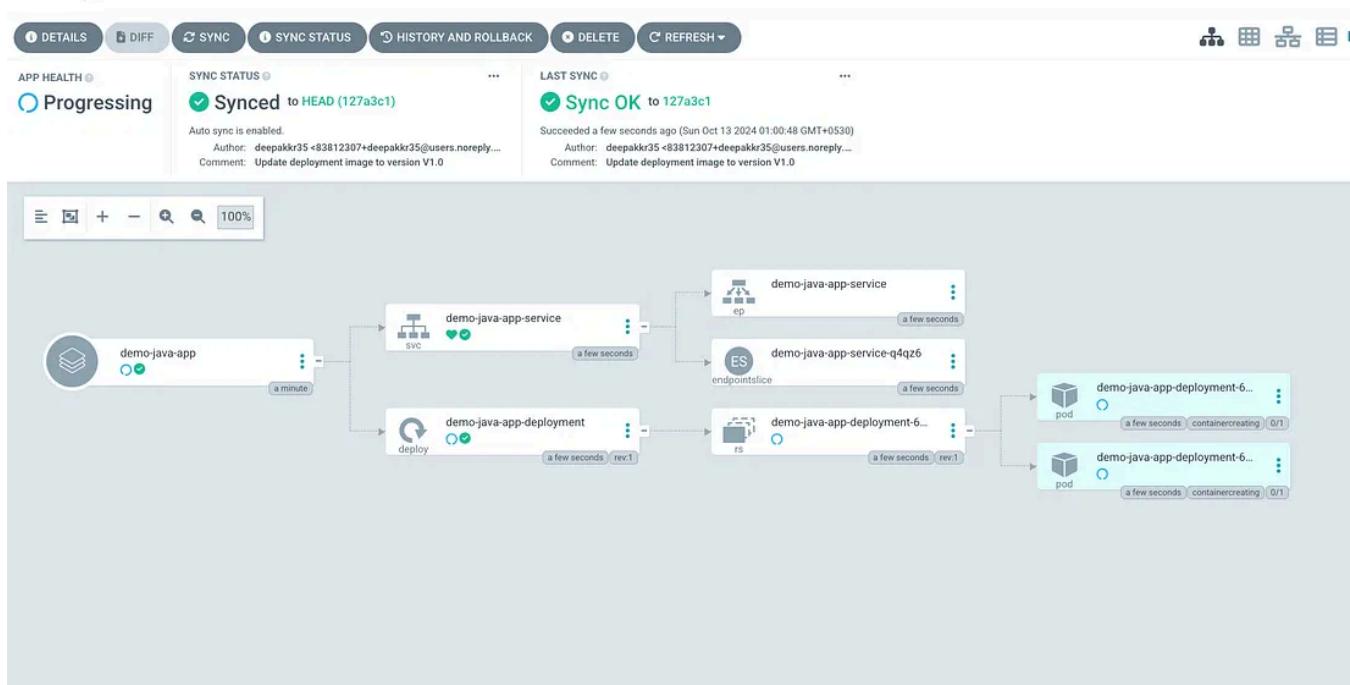
Destination: in-cluster

Namespace: app

Created At: 10/13/2024 01:00:16 (a few seconds ago)

Last Sync: 10/13/2024 01:00:48 (a few seconds ago)

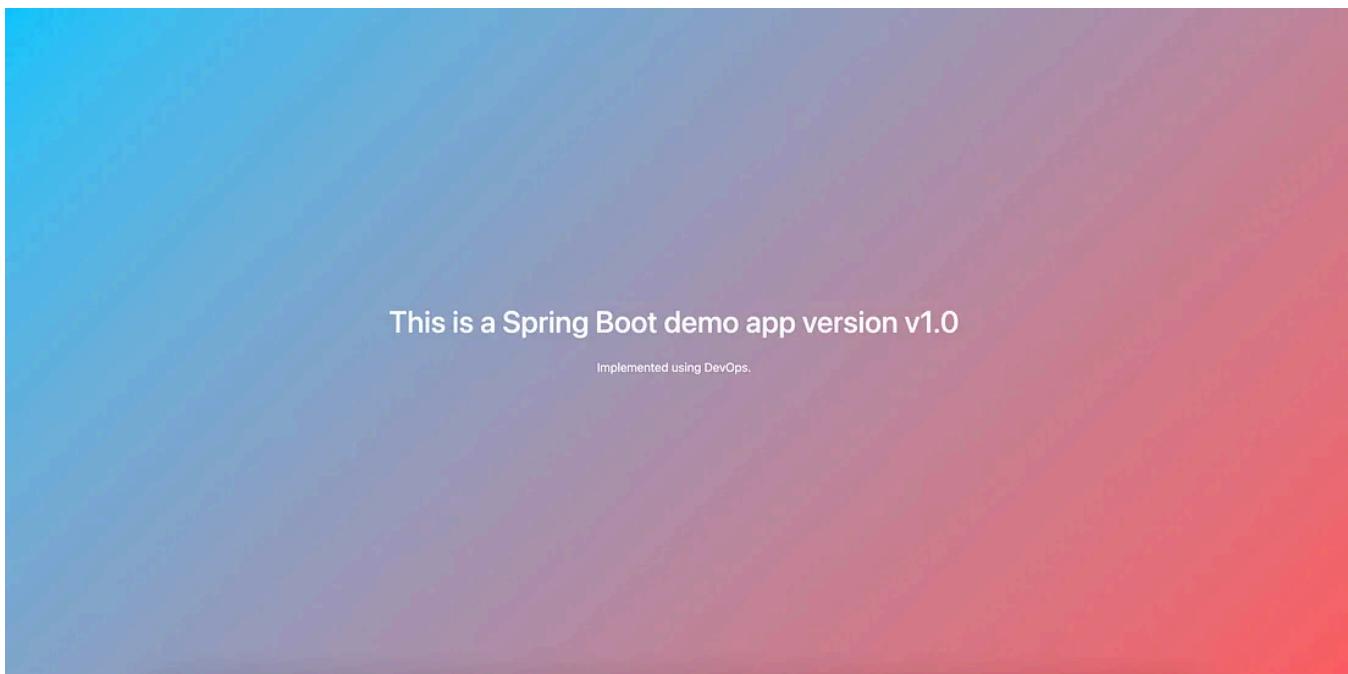
↻ SYNC ⟳ REFRESH ✖ DELETE



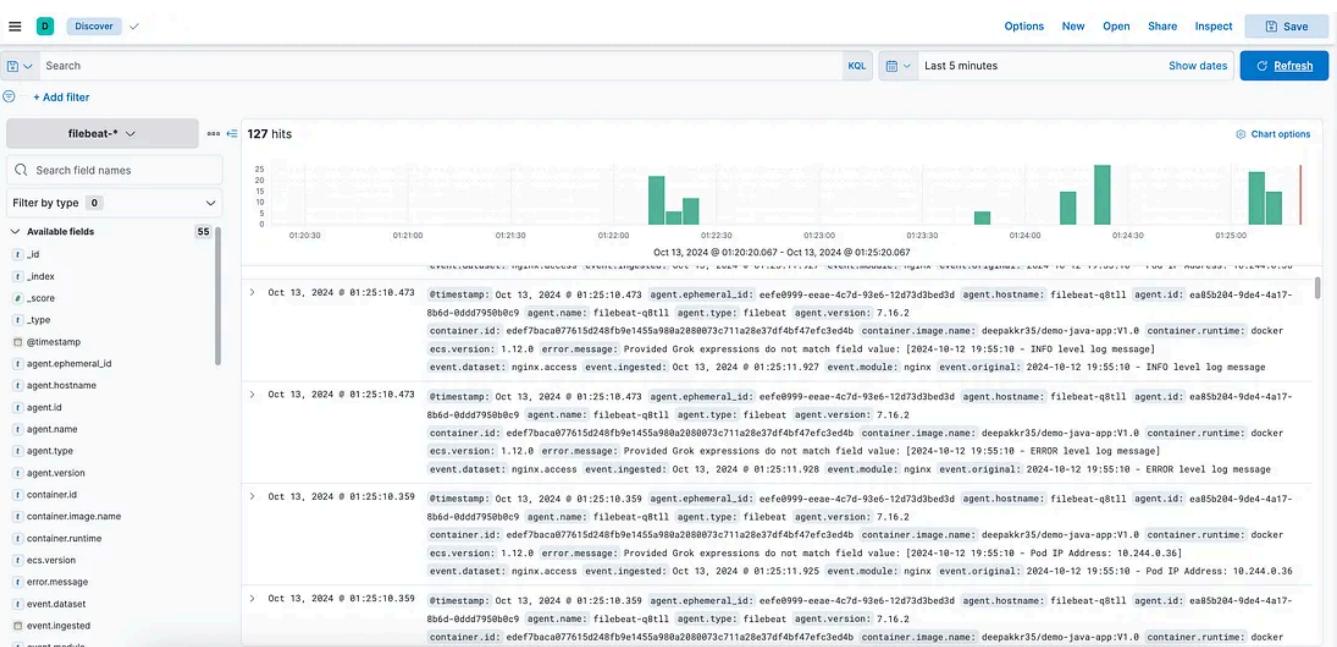
20. port-forward app to access through UI:

```
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("title", "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:

Pipeline demo-java-app-ci

This build requires parameters:

build_version

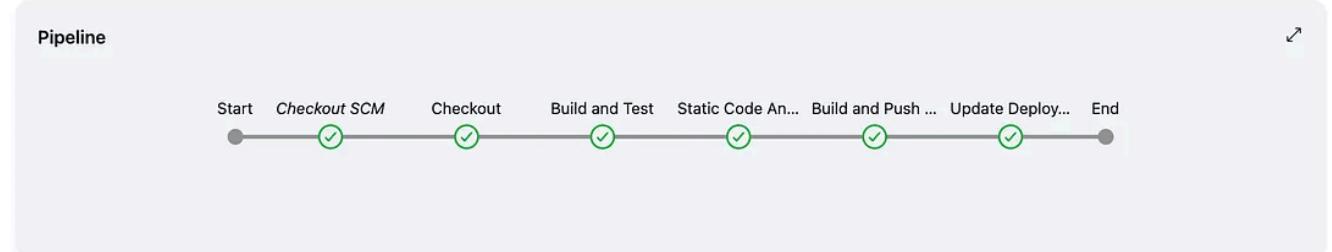
Build version to use for Docker image

V2.0

▷ Build

Cancel

✓ < Build #2



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

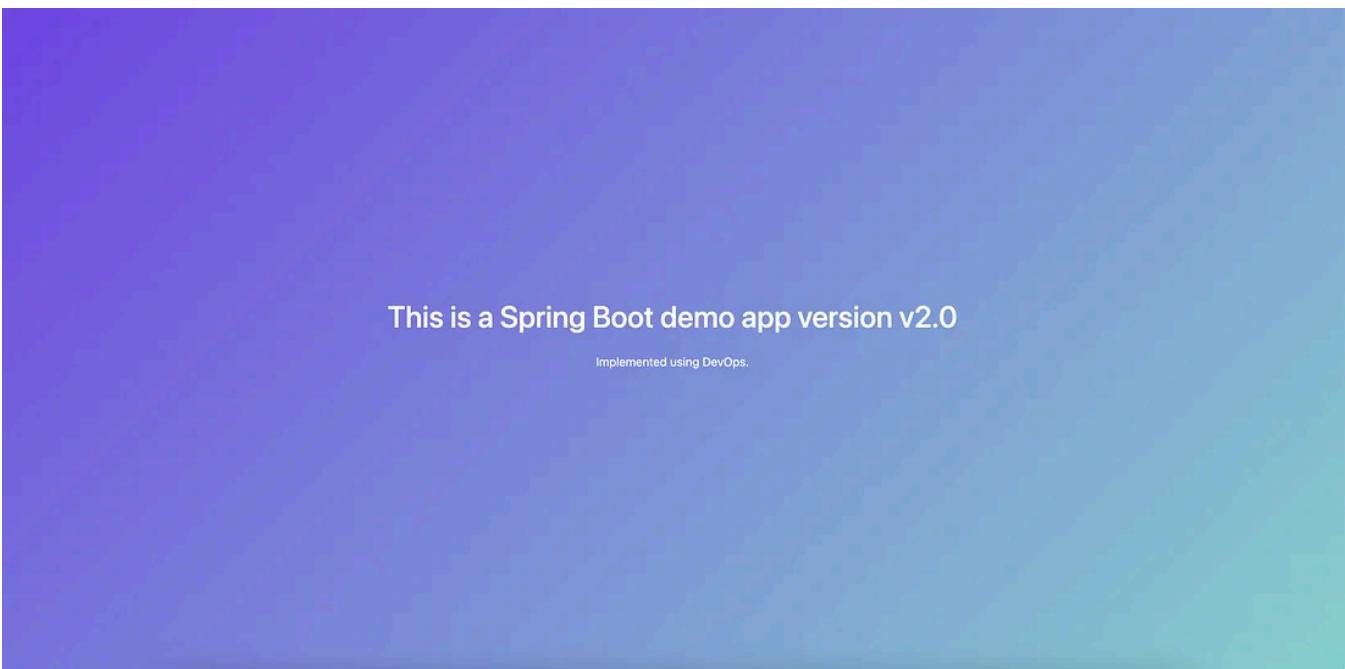
TAG		Last pushed 13 minutes ago by deepakkr35		docker pull deepakkr35/demo-java-app:V2.0	Copy
Digest	OS/ARCH	Last pull	Compressed Size		
306519f70c15	linux/arm64/v8	6 minutes ago	249.12 MB		

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