

PROJECT

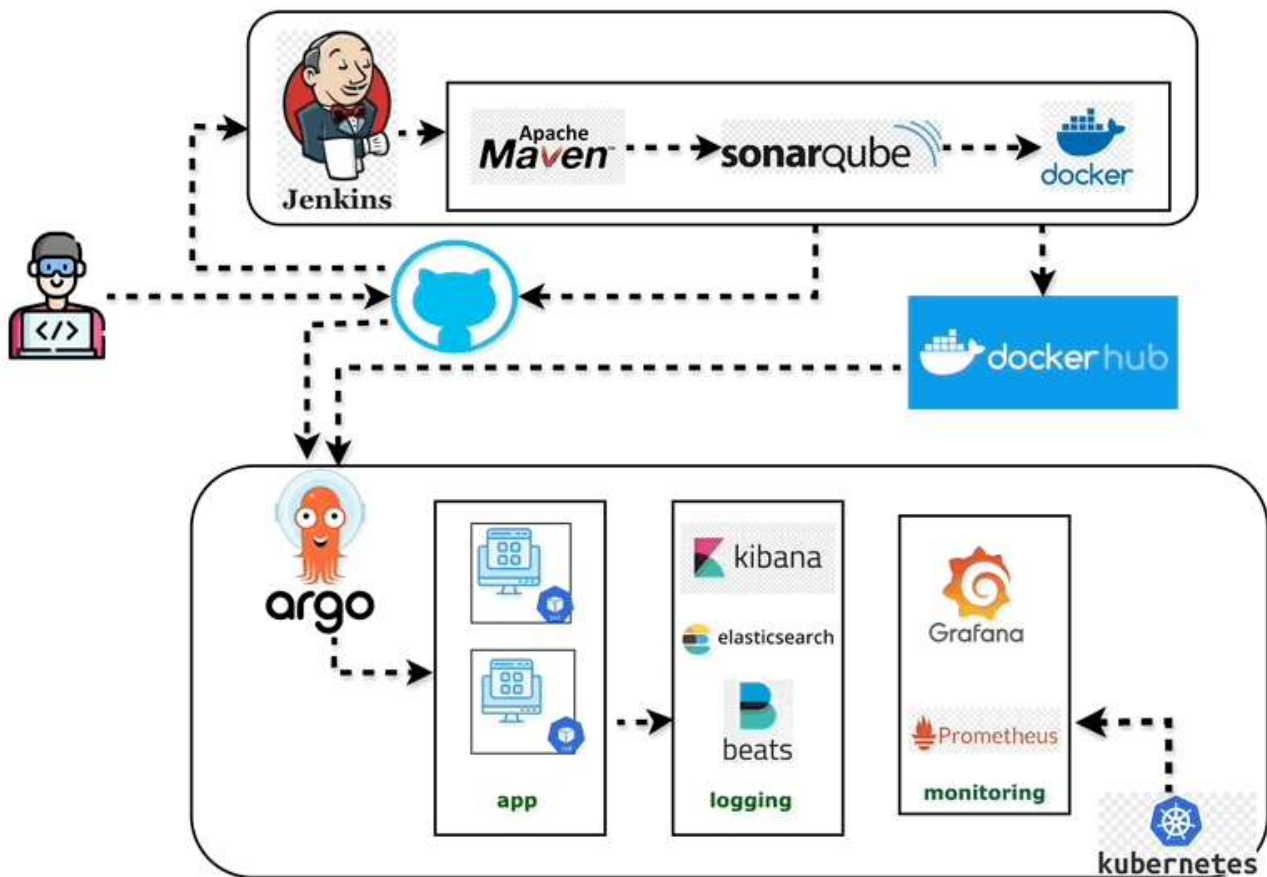
ARIR86 - Delivery Management, Devops et Pipeline

RS1-RS2-RS3

Créer un Pipeline CI/CD DevOps Localement

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

Enseignants: M. BOUZAYEN Abdelbaki and M. MLAYAH Najmeddine



Objectifs généraux :

- Construire un pipeline CI/CD complet dans un environnement local, en intégrant divers outils DevOps pour automatiser les processus de développement et de déploiement des applications.

- Comprendre et utiliser les principaux outils DevOps tels que GitHub, Jenkins, Maven, SonarQube, Docker, DockerHub, ArgoCD, Helm, Kubernetes, Prometheus, Grafana, Filebeat, OpenSearch et Kibana, en les configurant pour qu'ils fonctionnent ensemble de manière transparente au sein d'un pipeline CI/CD.

Présentation du projet

Ce projet implémente un pipeline DevOps CI/CD complet dans un environnement local. La partie CI est implémentée avec Docker et Docker Compose, offrant des environnements isolés pour Jenkins, SonarQube et d'autres outils nécessaires. DockerHub sert de registre de conteneurs pour la gestion des images Docker. Pour l'analyse de la qualité du code et la gestion des builds, Maven et SonarQube sont intégrés à Jenkins, garantissant un processus d'intégration continue fluide et efficace.

La partie CD est gérée par Kubernetes exécuté dans Minikube, permettant une orchestration locale des conteneurs. Le déploiement sur Kubernetes est automatisé grâce à ArgoCD, ce qui facilite les déploiements et les retours en arrière. La surveillance et la journalisation sont assurées par Prometheus, Grafana, Filebeat, OpenSearch et Kibana, tous installés via des charts Helm pour une configuration simplifiée. Cette configuration offre des capacités complètes d'observabilité et de journalisation, idéales pour tester de nouvelles fonctionnalités et des correctifs.

Prérequis :

Avant de commencer ce projet DevOps CI/CD, assurez-vous que les outils suivants sont installés et correctement configurés :

- **Docker** : Pour la conteneurisation.
- **Docker Compose** : Pour gérer des applications multi-conteneurs.
- **DockerHub** : pour stocker et gérer les images Docker.
- **kubectl** : pour interagir avec le cluster Kubernetes.
- **Minikube** : Pour mettre en place un environnement Kubernetes local.
- **Helm** : pour gérer les applications Kubernetes.
- **Git** : Pour le contrôle de version.
- **ArgoCD CLI** : (facultatif) pour gérer les déploiements via la ligne de commande.
- **Compte GitHub** : pour la gestion du référentiel.

Étapes de mise en œuvre :

0. Installez sur votre VM les outils DevOps via le script dans ce guide:

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

1. Fork et clonez le dépôt Git ci-dessous

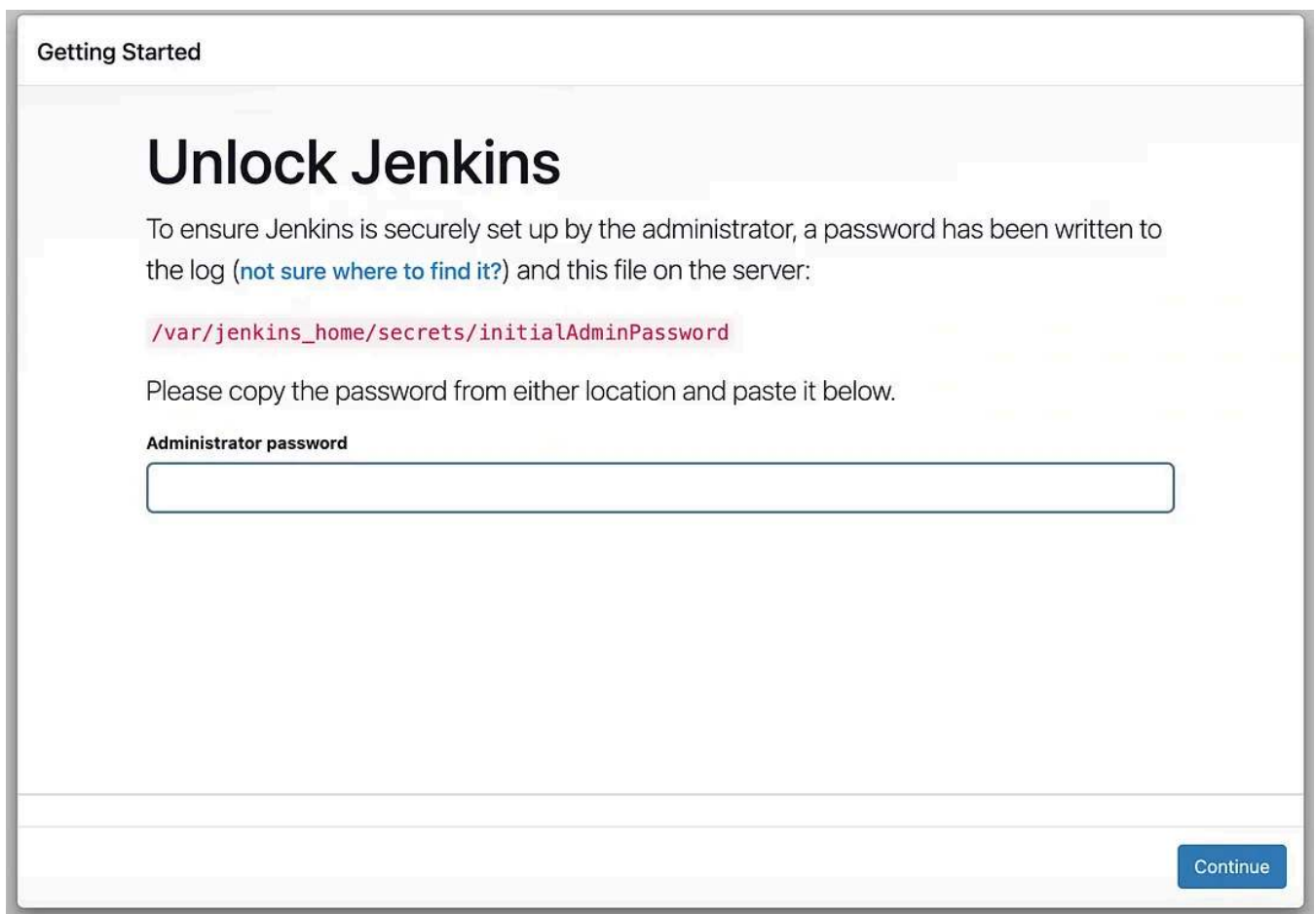
<https://github.com/abdelbaki-bouzaienne/devops-demo-project>

2. Ouvrez le terminal, accédez au répertoire du projet et utilisez docker-compose pour afficher l'environnement CI :

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

3. Ouvrez l'URL ci-dessous pour accéder à Jenkins :

<http://localhost:8010/>

The image shows the Jenkins 'Getting Started' page. At the top, it says 'Getting Started'. Below that is a large heading 'Unlock Jenkins'. The text explains that a password has been written to the log and a file on the server. The file path is shown in red: `/var/jenkins_home/secrets/initialAdminPassword`. It asks the user to copy the password from either location and paste it below. There is a text input field labeled 'Administrator password'. At the bottom right, there is a blue 'Continue' button.

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

4. Accédez au mot de passe Jenkins, connectez-vous et installez les plugins suggérés :

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

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](#)

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:

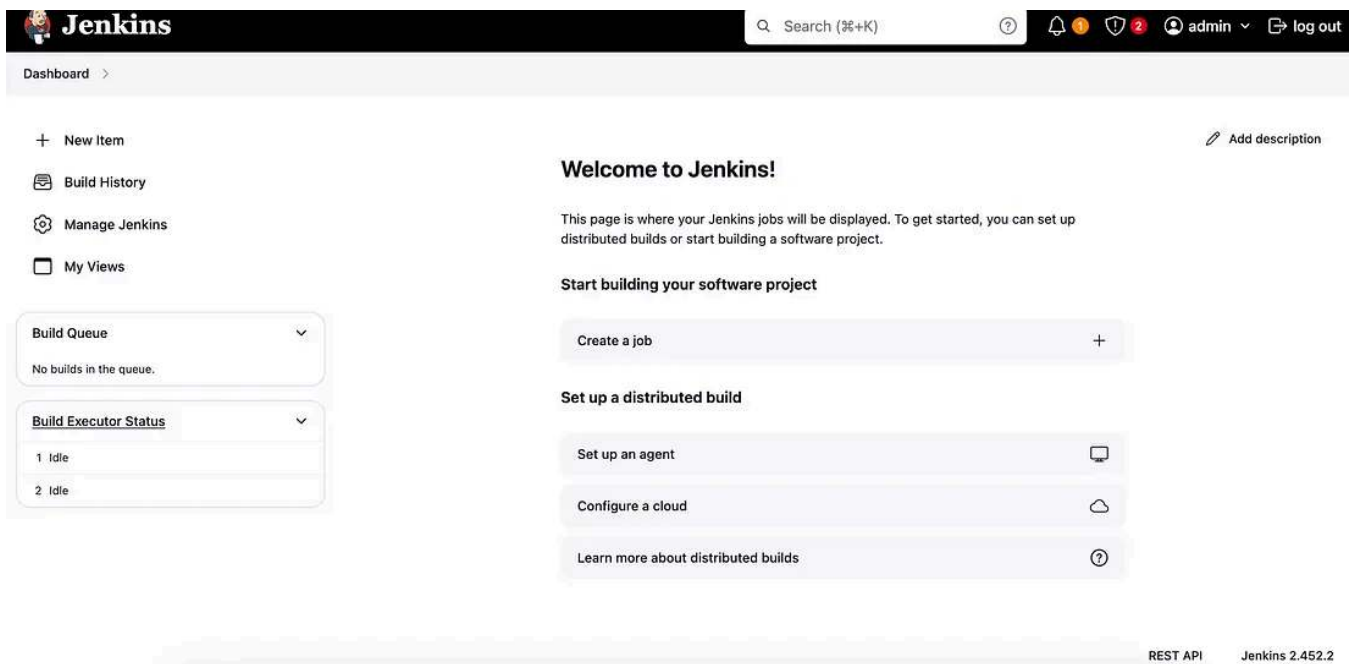
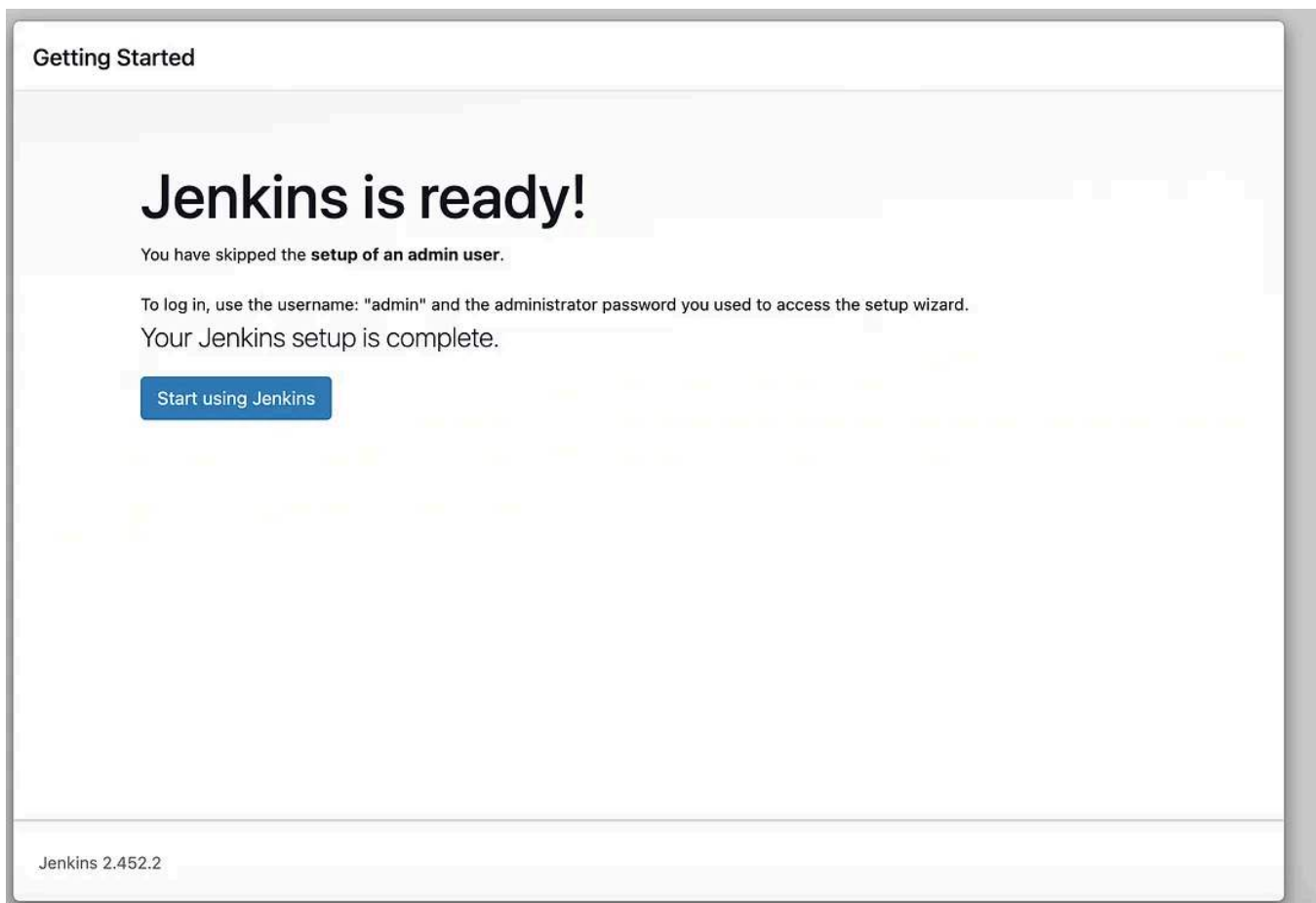
The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

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](#)



5. Installez les plugins suggérés ci-dessous et redémarrez le conteneur Jenkins :

Jenkins

Search (#+K)

Dashboard > Manage Jenkins > Plugins

Plugins

Search available plugins

Install

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

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

Dashboard > Manage Jenkins > Plugins

Plugins

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

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. Ouvrez SonarQube en utilisant l'URL ci-dessous et créez un jeton :

<http://localhost:9000>

nom d'utilisateur : admin

mot de passe : admin

sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministration

?

Search for projects...

A

A

Administrator

ProfileSecurityNotificationsProjects

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

Namejenkins

TypeUser Token

Expires in30 days

Generate

Name	Type	Project	Last use	Created	Expiration
No tokens					

7. Connectez-vous à votre compte GitHub et générez un jeton d'accès personnel :

GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens

Tokens (classic)

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:pgp_key, admin:org, admin:org_hook, 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

Last used within the last week

Delete

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.

TermsPrivacySecurityStatusDocsContactManage cookiesDo not share my personal information

GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens

Tokens (classic)

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

Jenkins

What's this token for?

Expiration

30 days

 The token will expire on Mon, Nov 11 2024

Select scopes

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

☒ repo

Full control of private repositories

☒ repo:status

Access commit status

☒ repo_deployment

Access deployment status

☒ public_repo

Access public repositories

☒ repo:invite

Access repository invitations

☒ security_events

Read and write security events

☒ workflow

Update GitHub Action workflows

☒ write:packages

Upload packages to GitHub Package Registry

☒ read:packages

Download packages from GitHub Package Registry

☒ delete:packages

Delete packages from GitHub Package Registry

☒ admin:org

Full control of orgs and teams, read and write org projects

☒ write:org

Read and write org and team membership, read and write org projects

☒ read:org

Read org and team membership, read org projects

☒ manage_runners:org

Manage org runners and runner groups

☒ admin:public_key

Full control of user public keys

GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens

Tokens (classic)

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

Jenkins

What's this token for?

Expiration

30 days

The token will expire on Mon, Nov 11 2024

Select scopes

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

☒ repo

Full control of private repositories

☒ repo:status

Access commit status

☒ repo_deployment

Access deployment status

☒ public_repo

Access public repositories

☒ repo:invite

Access repository invitations

☒ security_events

Read and write security events

☒ workflow

Update GitHub Action workflows

☒ write:packages

Upload packages to GitHub Package Registry

☒ read:packages

Download packages from GitHub Package Registry

☒ delete:packages

Delete packages from GitHub Package Registry

☒ admin:org

Full control of orgs and teams, read and write org projects

☒ write:org

Read and write org and team membership, read and write org projects

☒ read:org

Read org and team membership, read org projects

☒ manage_runners:org

Manage org runners and runner groups

☒ admin:public_key

Full control of user public keys

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

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind

Secret text

Scope

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

Secret

.....

ID

github

Description

Create

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

» 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

Choose 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/Jenkins/Jenkinsfile

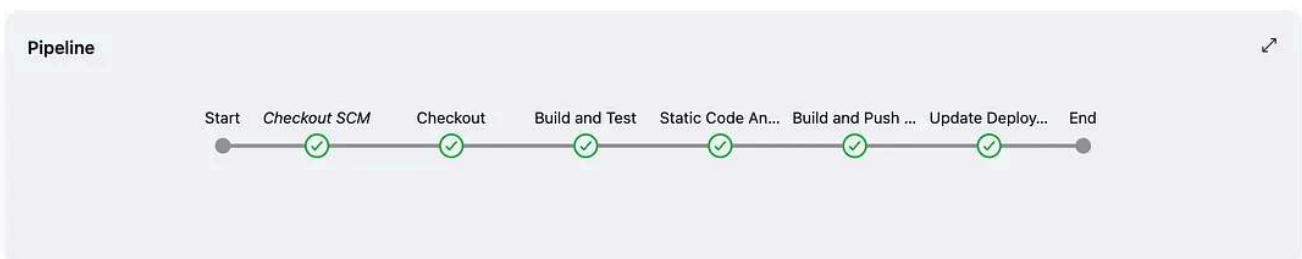
☒ Lightweight checkout ?

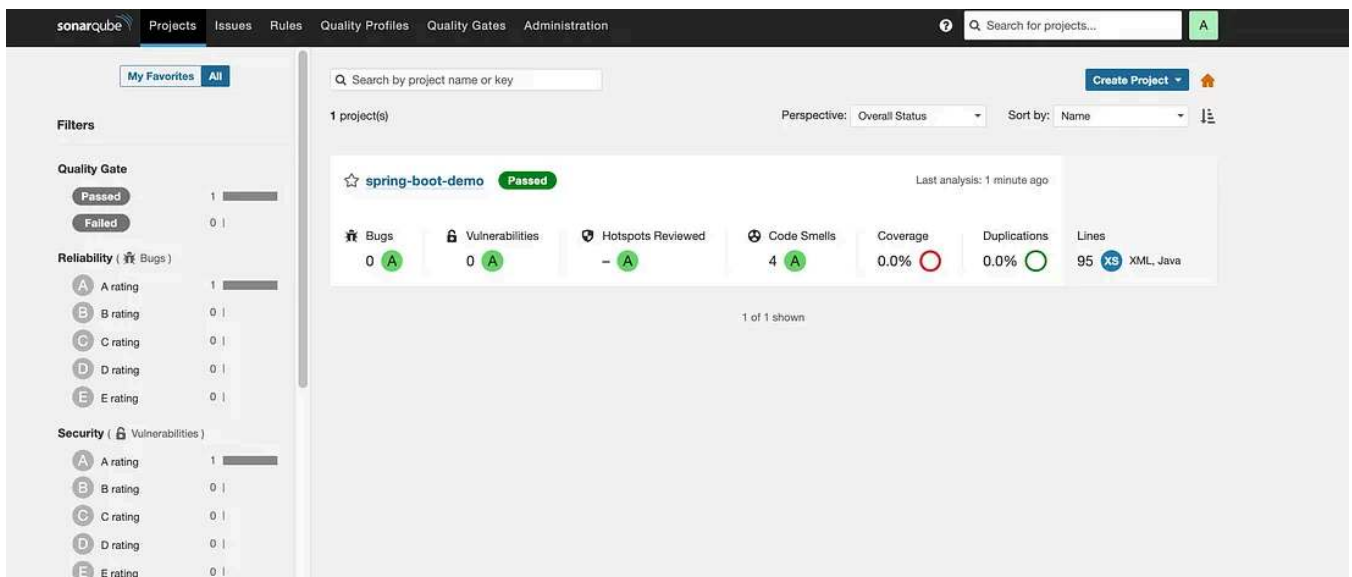
[Pipeline Syntax](#)

Save Apply

10. Build the pipeline with V1.0 parameter:

✓ Build #1



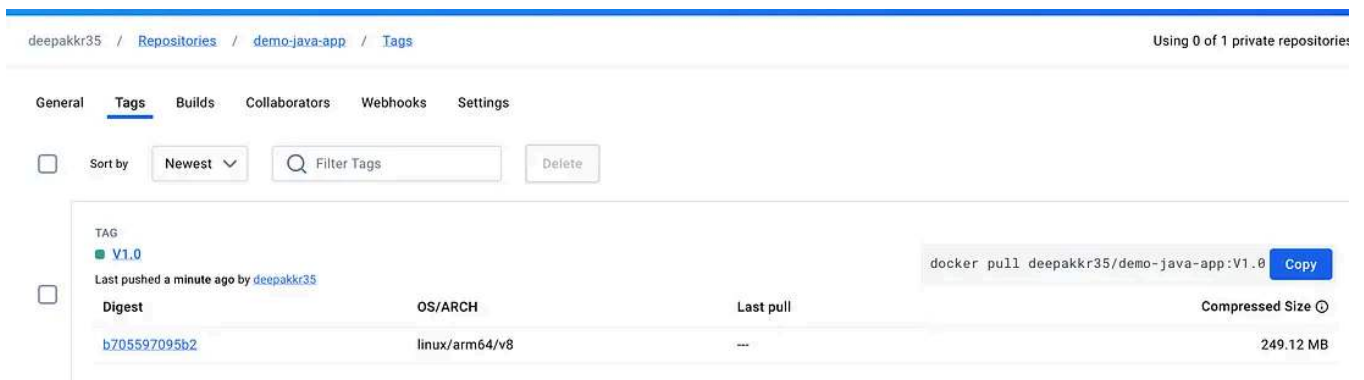


11. Docker image will be created and GitHub helm/app/values.yaml will be updated with appropriate tag:

```

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: V1.0
13

```



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 namespace using Helm: (port-forward is required to access UI, as minikube is running in Docker driver mode)

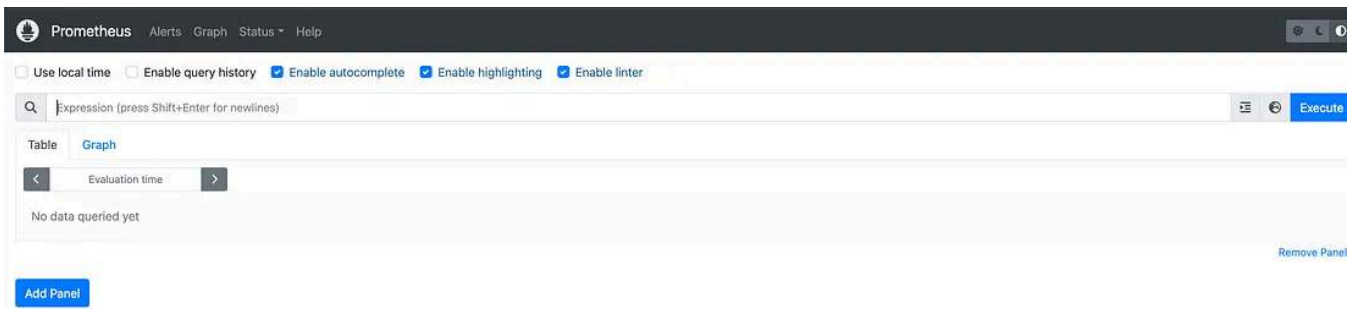
(voir sur teams)

```

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

```

<http://localhost:9030/>



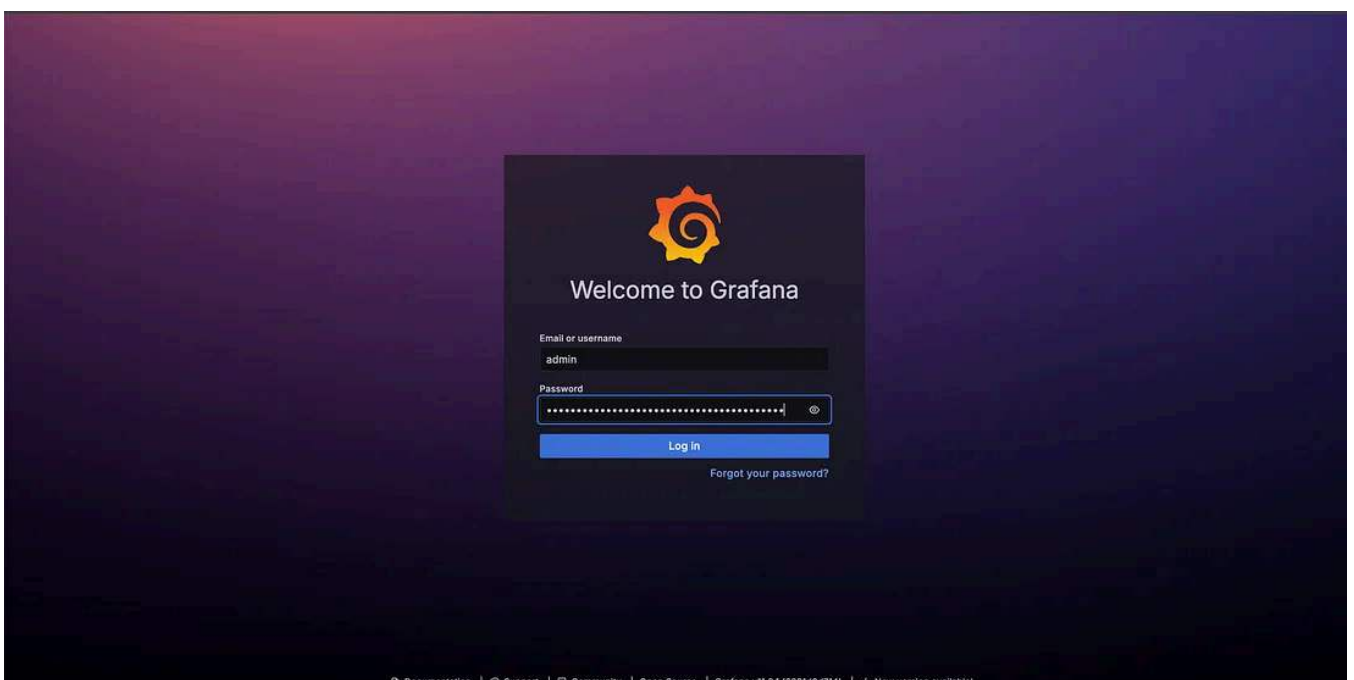
(voir sur teams)

```
helm repo add grafana https://grafana.github.io/helm-charts
helm install grafana grafana/grafana -n monitoring

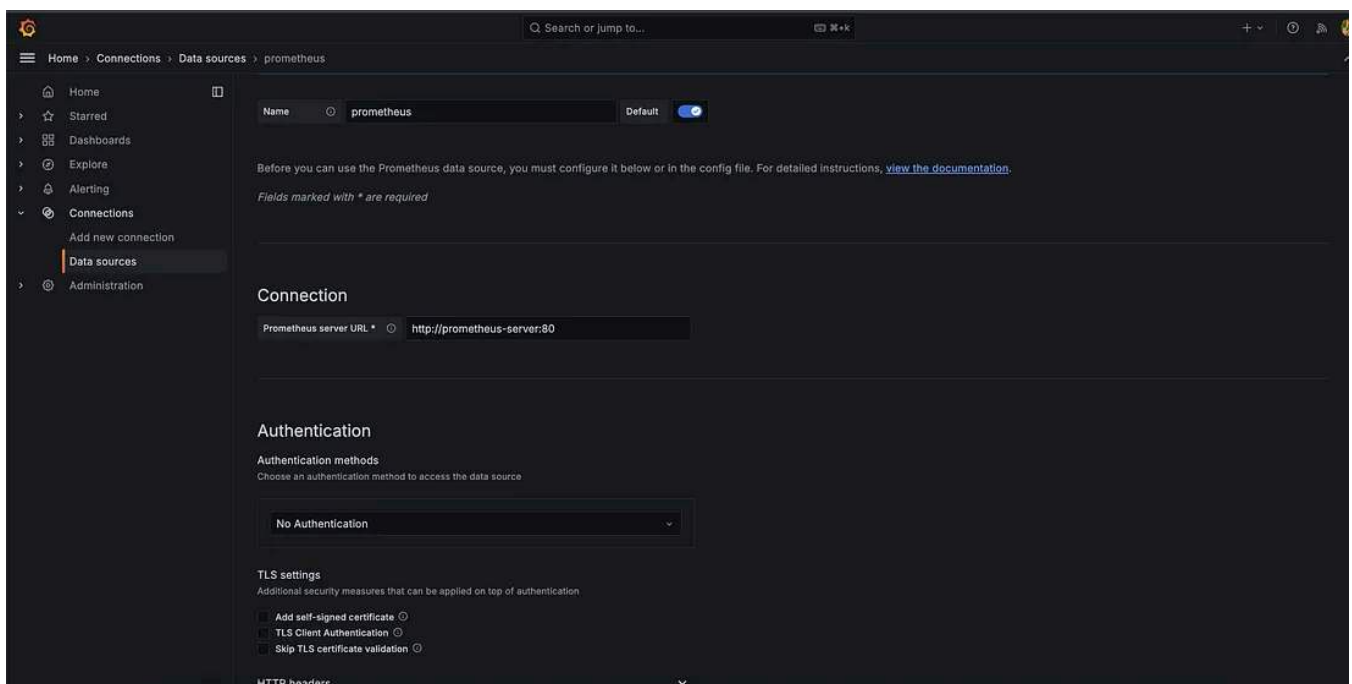
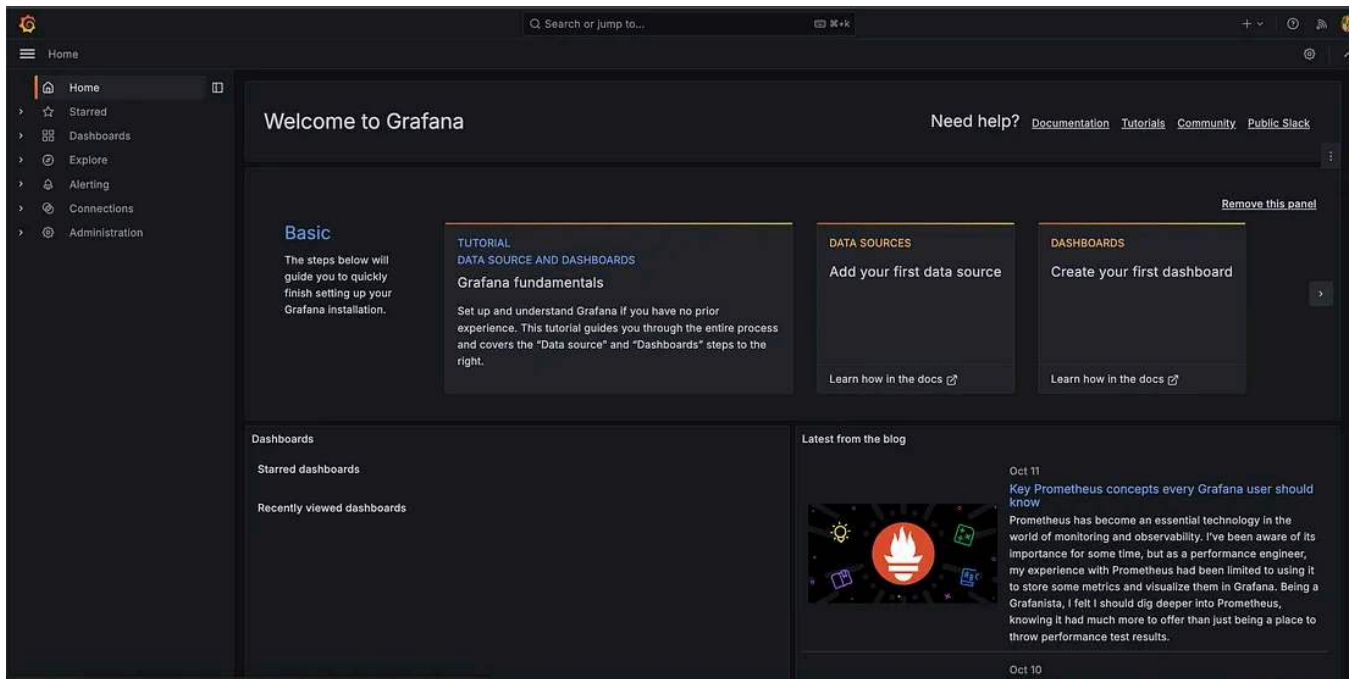
kubectl get secret --namespace monitoring grafana -o jsonpath="{.data.admin-password}" | base64
--decode ; echo

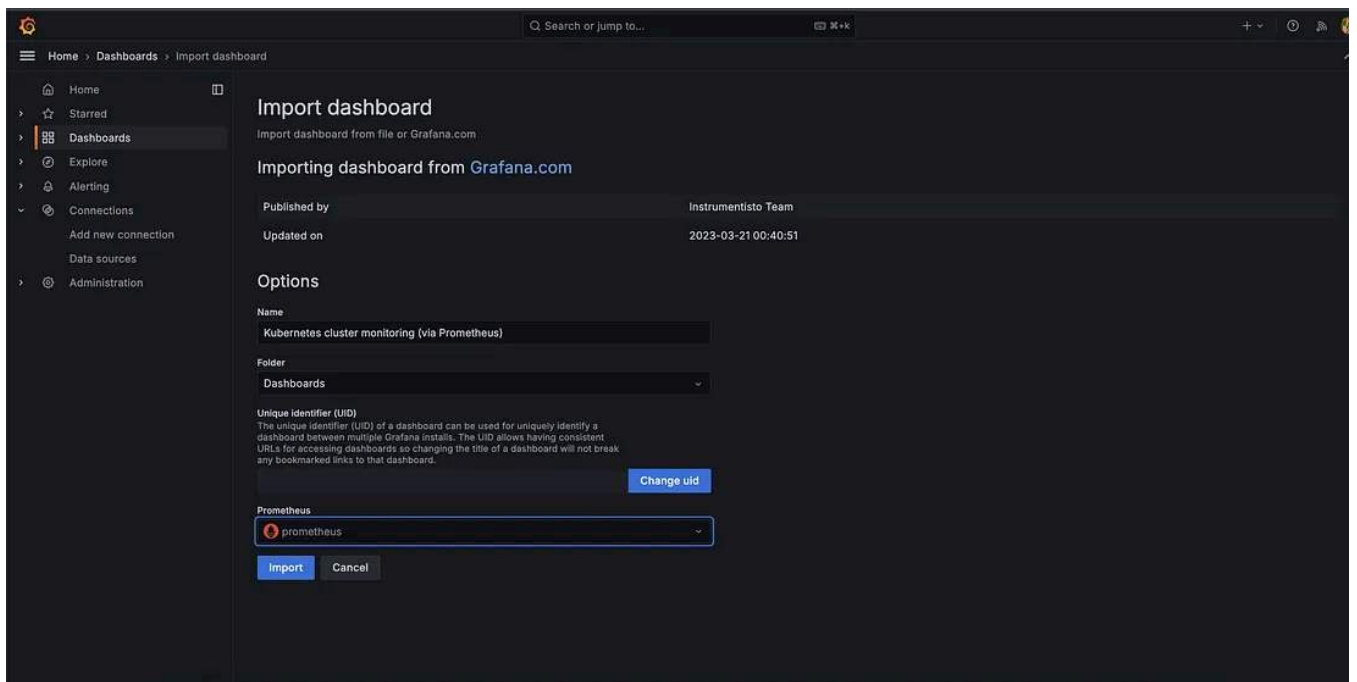
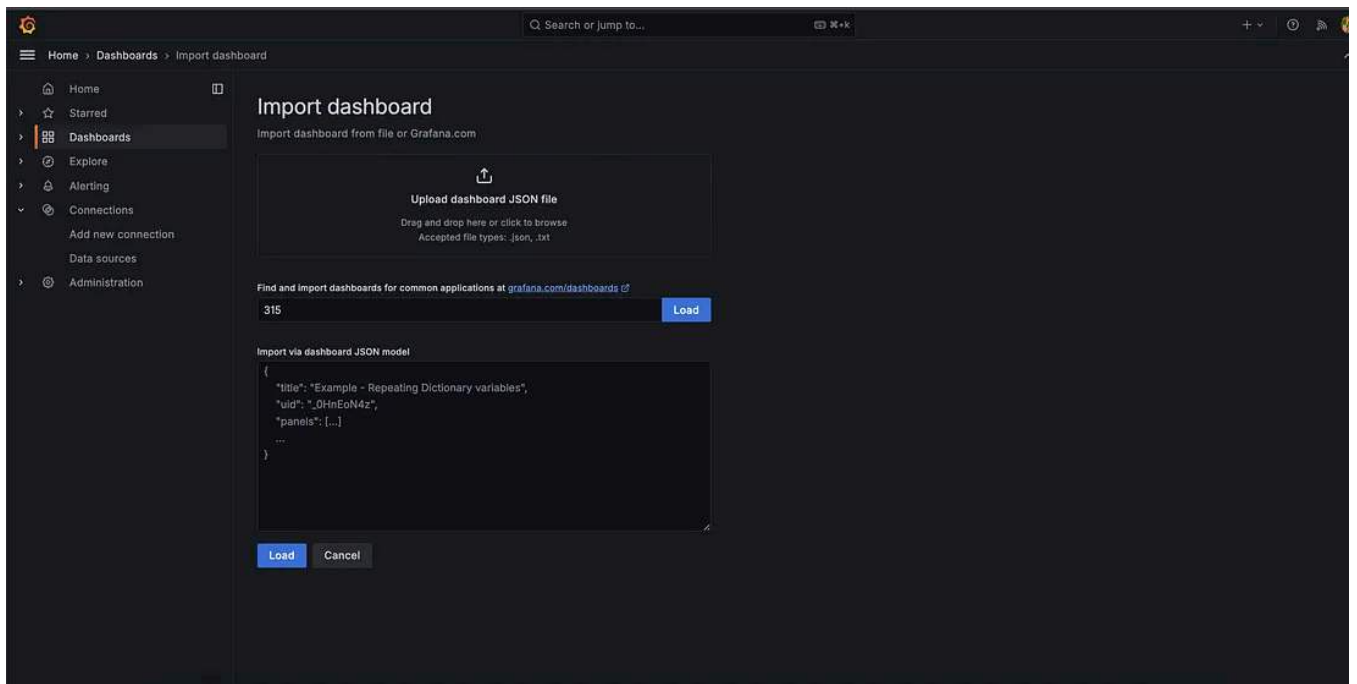
kubectl port-forward service/grafana 9040:80 -n monitoring
```

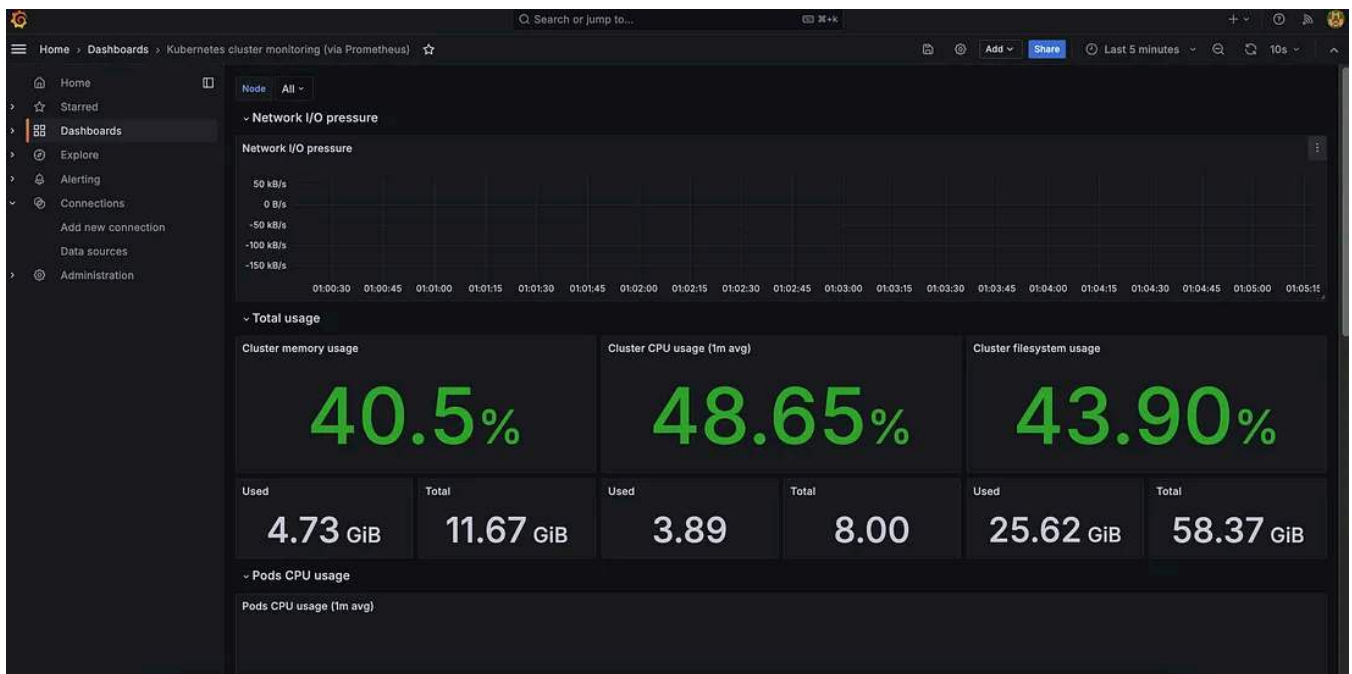
<http://localhost:9040/>



16. Configure Grafana to use Prometheus as data source and import required dashboards:







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

(Voir sur teams)

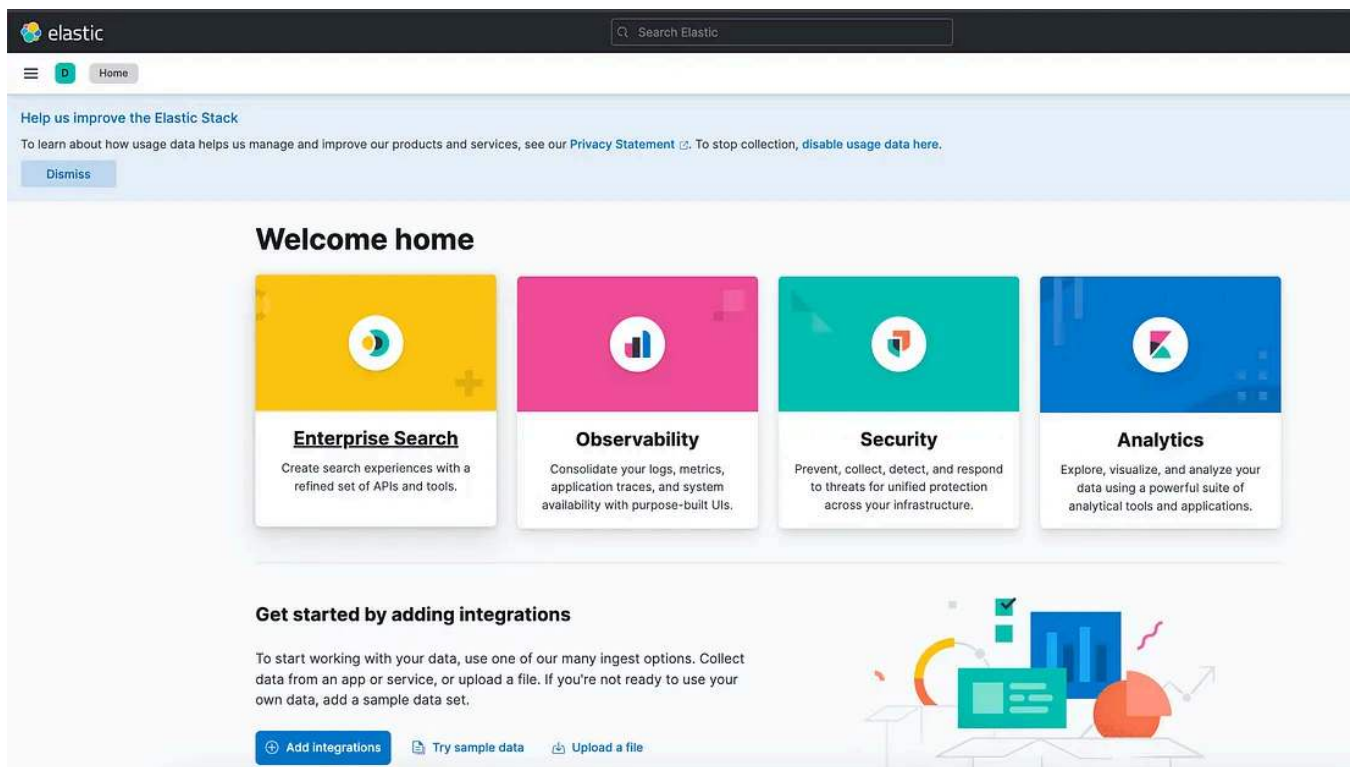
```
cd devops-demo-project/helm
helm install elasticsearch ./logging/elasticsearch
helm install kibana ./logging/kibana
helm install filebeat ./logging/filebeat

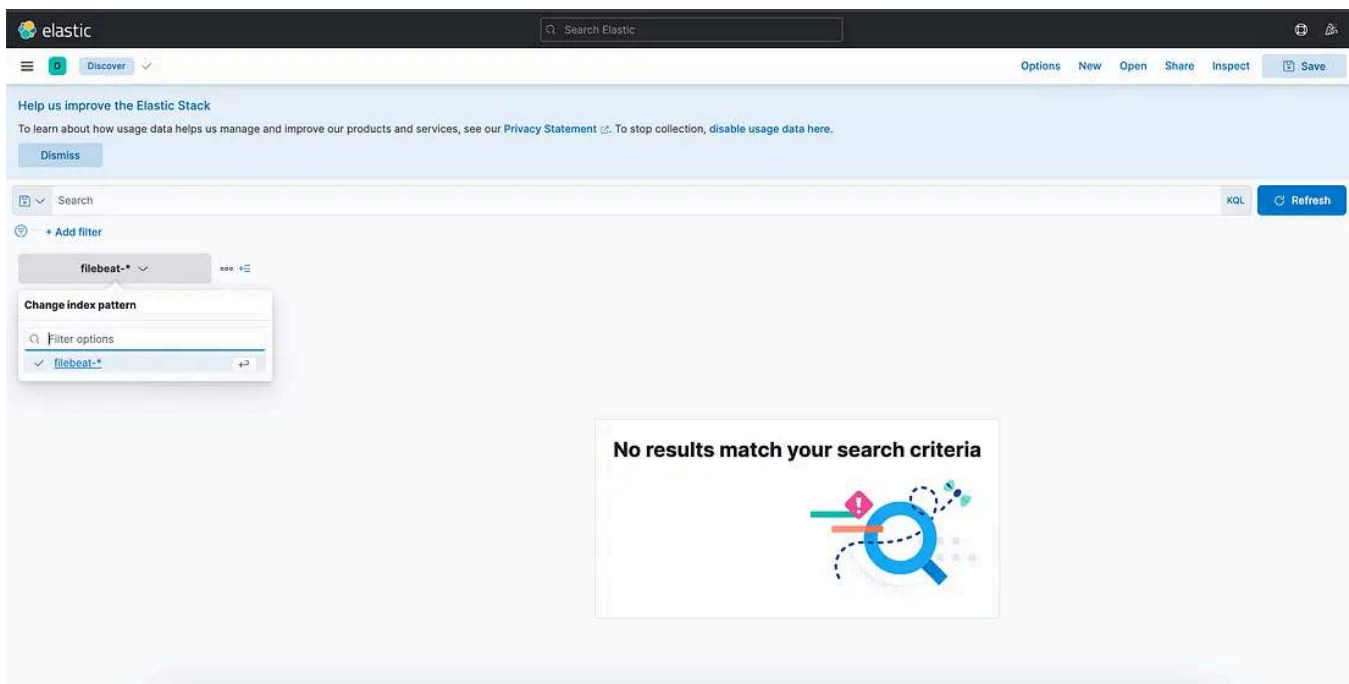
kubectl port-forward service/es-master-service 9010:9200 -n logging
kubectl port-forward service/kibana-service 9020:5601 -n logging
```

<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: (voir teams)

```
cd devops-demo-project/cd/argocd
curl -sL https://github.com/operator-framework/operator-lifecycle-manager/releases/download/v0.28.0/install.sh | bash
-s v0.28.0
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.admin\.password}" | base64 --decode)
minikube service example-argocd-server -n argocd
```



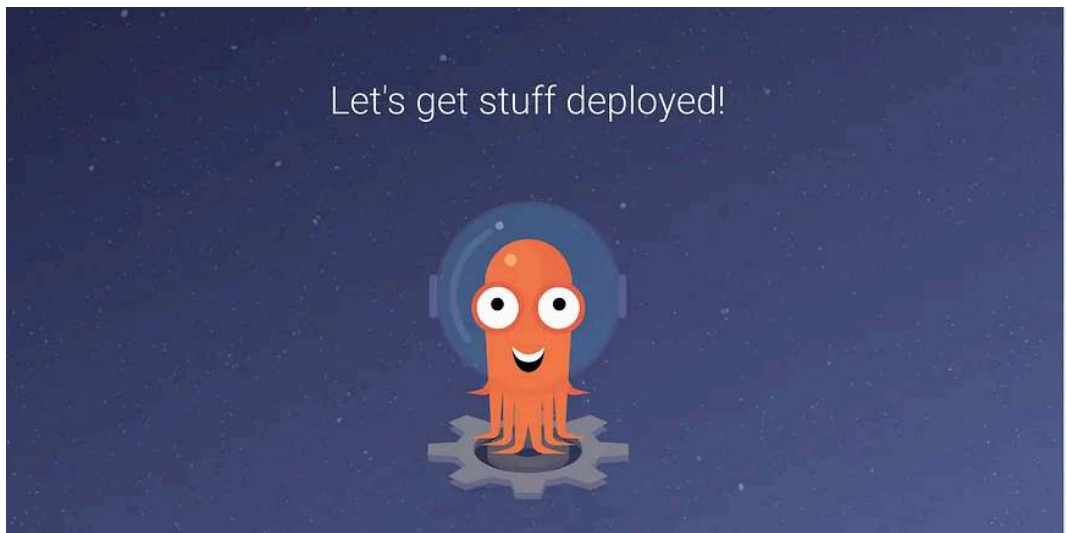
NAMESPACE	NAME	TARGET PORT	URL
argocd	example-argocd-server		No node port

🚨 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:



argo v2.12.1+26b2099

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+26b2099

Applications

+ NEW APP

CREATE CANCEL

SOURCE

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

Revision
HEAD

Path
helm/app

DESTINATION

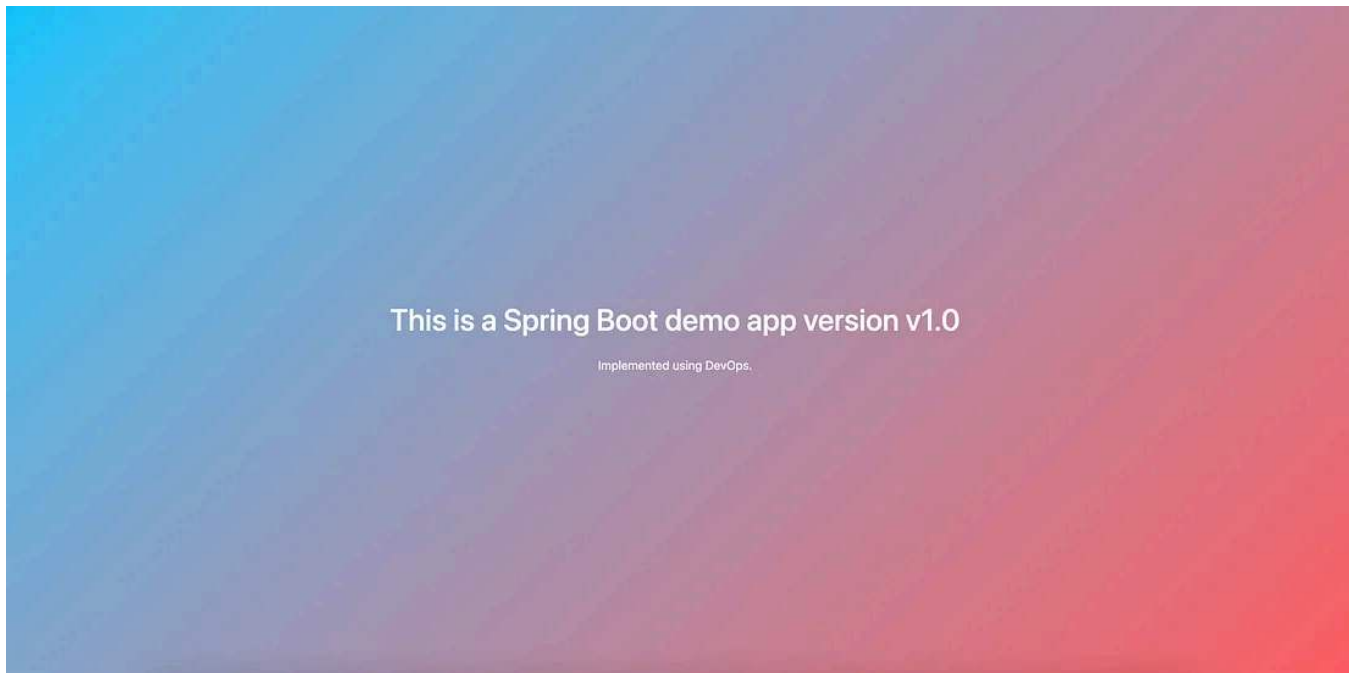
Cluster URL
https://kubernetes.default.svc

Namespace
app

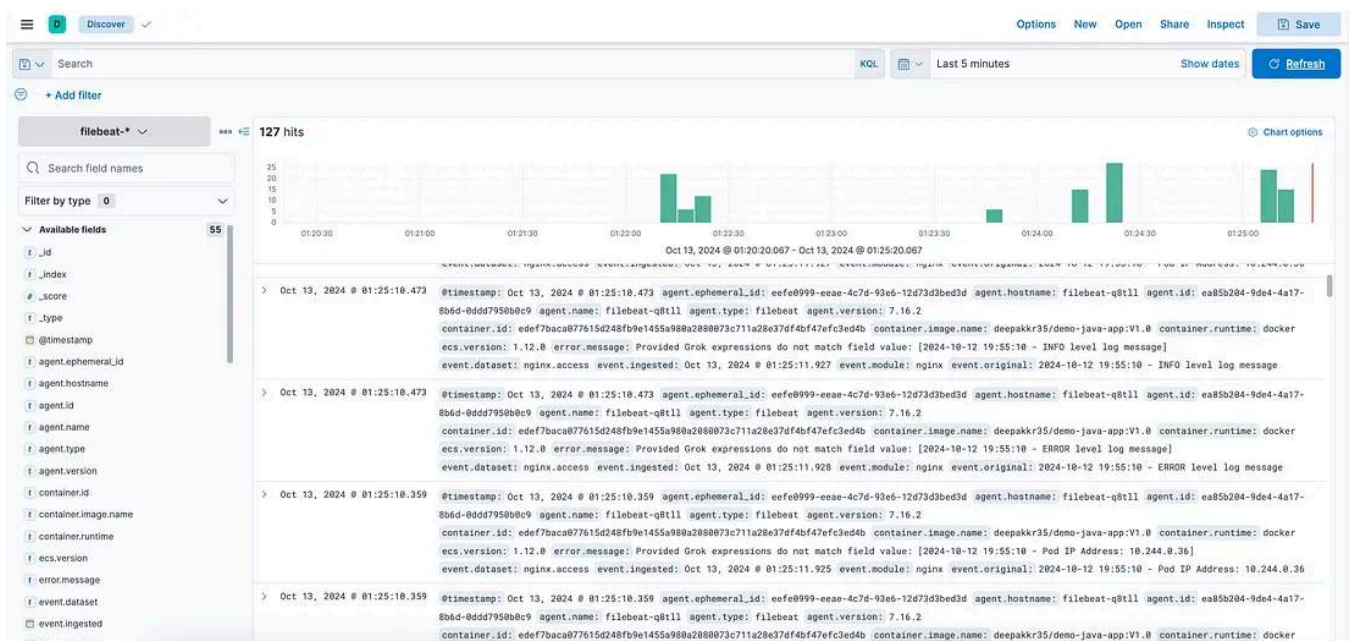
Helm

HELM

<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

23. Build pipeline in Jenkins:

Pipeline demo-java-app-ci

This build requires parameters:

build_version

Build version to use for Docker image

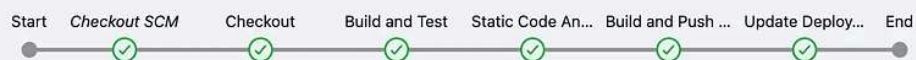
V2.0

▶ Build

Cancel

✓ < Build #2

Pipeline



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

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

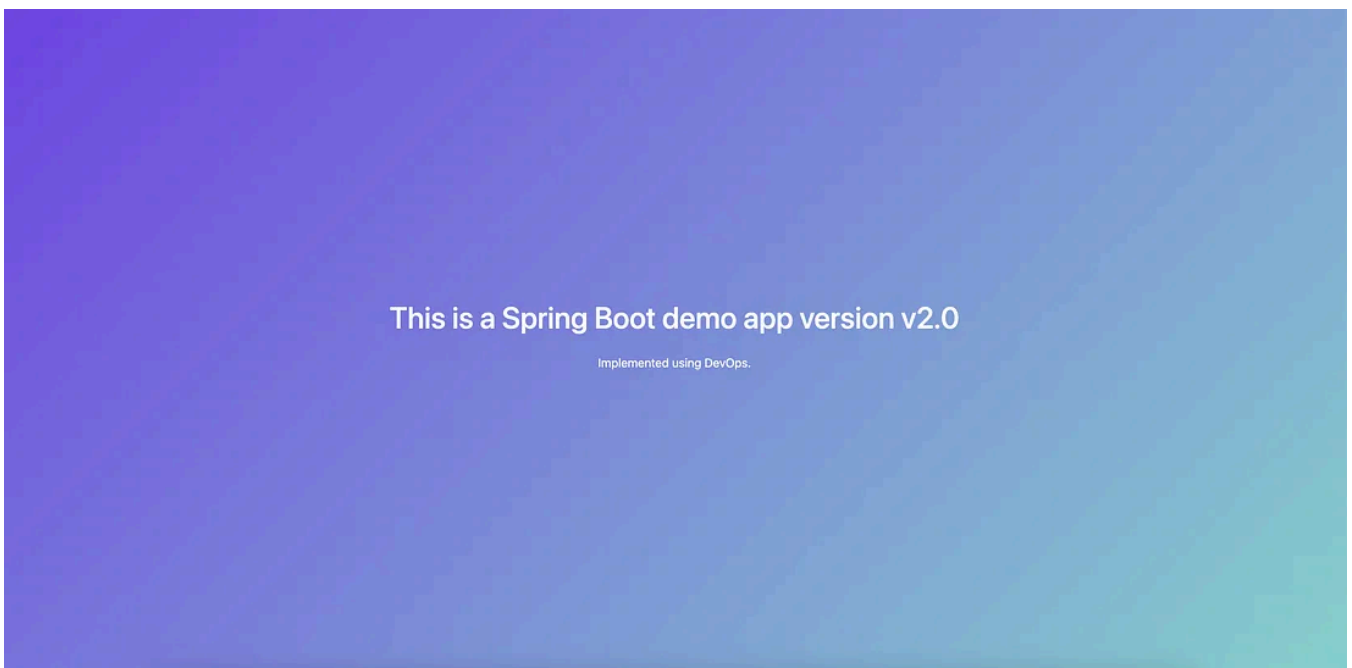
@@ -9,7 +9,7 @@ image:

```
9     repository: demo-java-app
10     pullPolicy: IfNotPresent
11     # Overrides the image tag whose default is the chart appVersion.
12 -   tag: V1.0
13 +   tag: V2.0
14
15     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
```

DevOps

Jenkins

Docker

Argo Cd

Ci Cd Pipeline