**PropPulse CI/CD deployment (GitHub, Jenkins, DockerHub & Minikube)**

|  |  |
| --- | --- |
| **Version** | 1.0.1 |
| **Prepared by** | Rimah Houssameldine |
| **Audience** | ParkInnovation Team |
| **Date** | 23/02/2024 |

Table of Contents

[**A.** **Introduction:** 2](#_Toc159956415)

[1. Requirements: 2](#_Toc159956416)

[**B.** **Steps for deploying PopPulse** 3](#_Toc159956417)

[1. Create GitHub Repository 3](#_Toc159956418)

[2. Push local files to the created repository 4](#_Toc159956419)

[3. Create a DockerHub repository 5](#_Toc159956420)

[4. Build docker images 6](#_Toc159956421)

[5. Build a pipeline on jenkins 8](#_Toc159956422)

[6. Add the required credentials 9](#_Toc159956423)

[7. Create DB, FE & BE docker files 11](#_Toc159956424)

[8. Create a database directory 13](#_Toc159956425)

[9. Jenkins Pipeline execution 14](#_Toc159956426)

[10. Apply Kubernetes Configuration for Account Resources 15](#_Toc159956427)

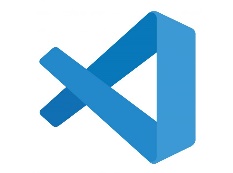
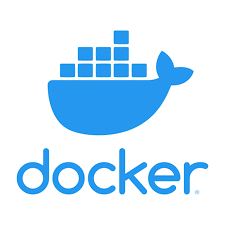
[11. Creating a Production Environment 19](#_Toc159956428)

## 

## **Introduction:**

This guide walks through deploying a PHP application using GitHub, DockerHub, Jenkins, and Minikube. It covers setting up a GitHub repository, configuring DockerHub, implementing Jenkins for automation, and utilizing Minikube for local Kubernetes deployment. By following this documentation, users can establish a seamless CI/CD pipeline for efficient development, testing, and deployment of their PHP application.

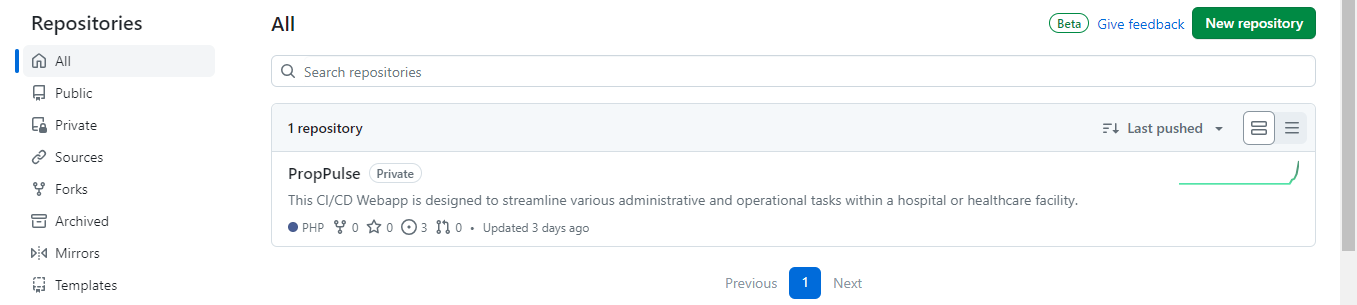
### Requirements:

* + - * PC (Desktop, Laptop)
      * Git, GitHub, Docker engine, DockerHub, Jenkins & Minikube
      * VS Code
      * Browser and Internet

## **Steps for deploying PopPulse**

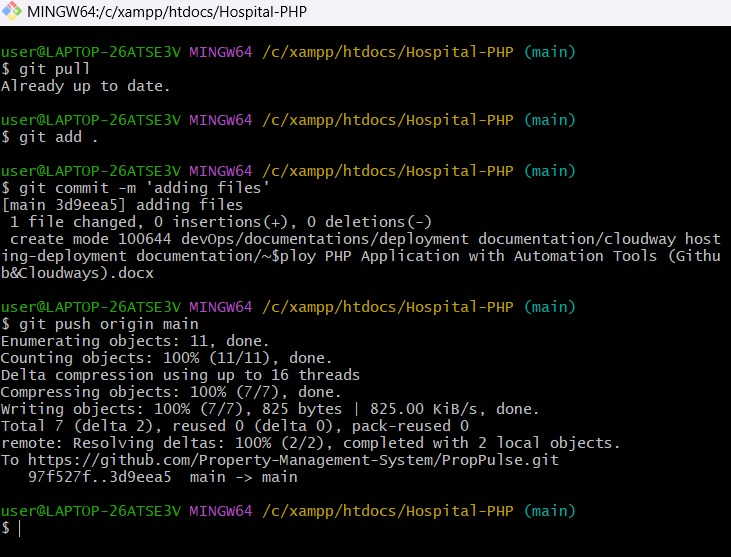
### Create GitHub Repository

To create a new GitHub repository, navigate to your GitHub account, click on the "+" icon in the top-right corner, and select "New repository", then follow the prompts to configure repository settings and create it.



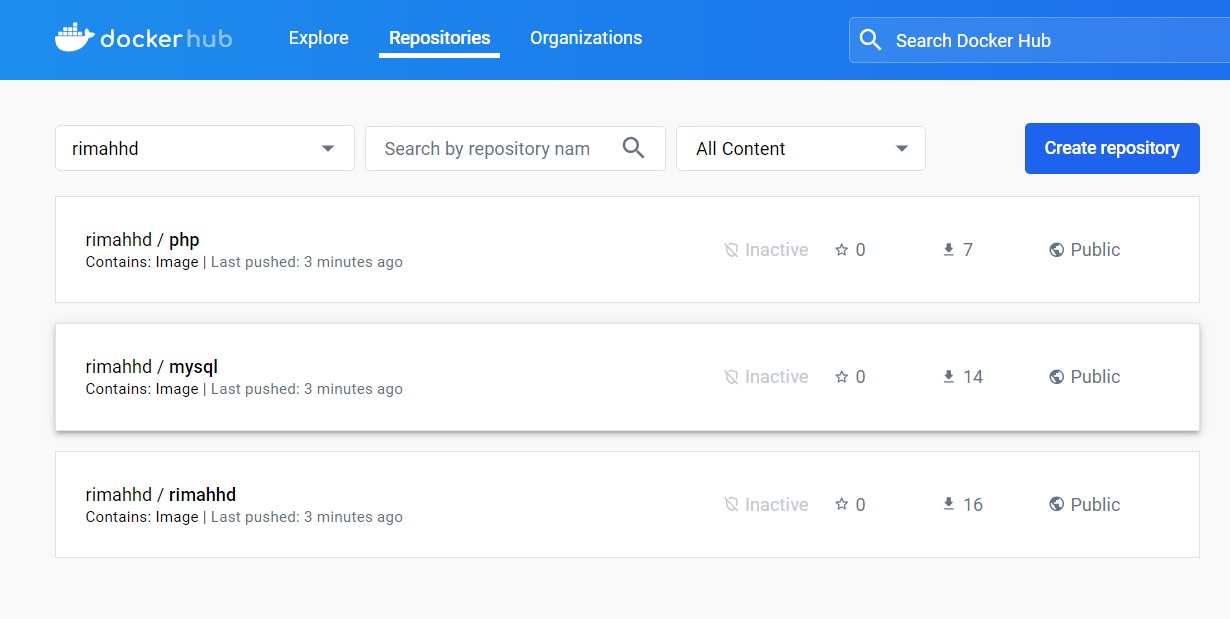
### Push local files to the created repository

To push local files to a GitHub repository, first ensure you have initialized a Git repository locally using **git init**. Then, add your files using **git add .**, commit them using **git commit -m "Your commit message"**, and finally, push them to the GitHub repository using **git push origin master** (replace "master" with your branch name if applicable). Make sure you have the appropriate permissions to push to the repository and that you have configured the correct remote URL for your GitHub repository.



### Create a DockerHub repository

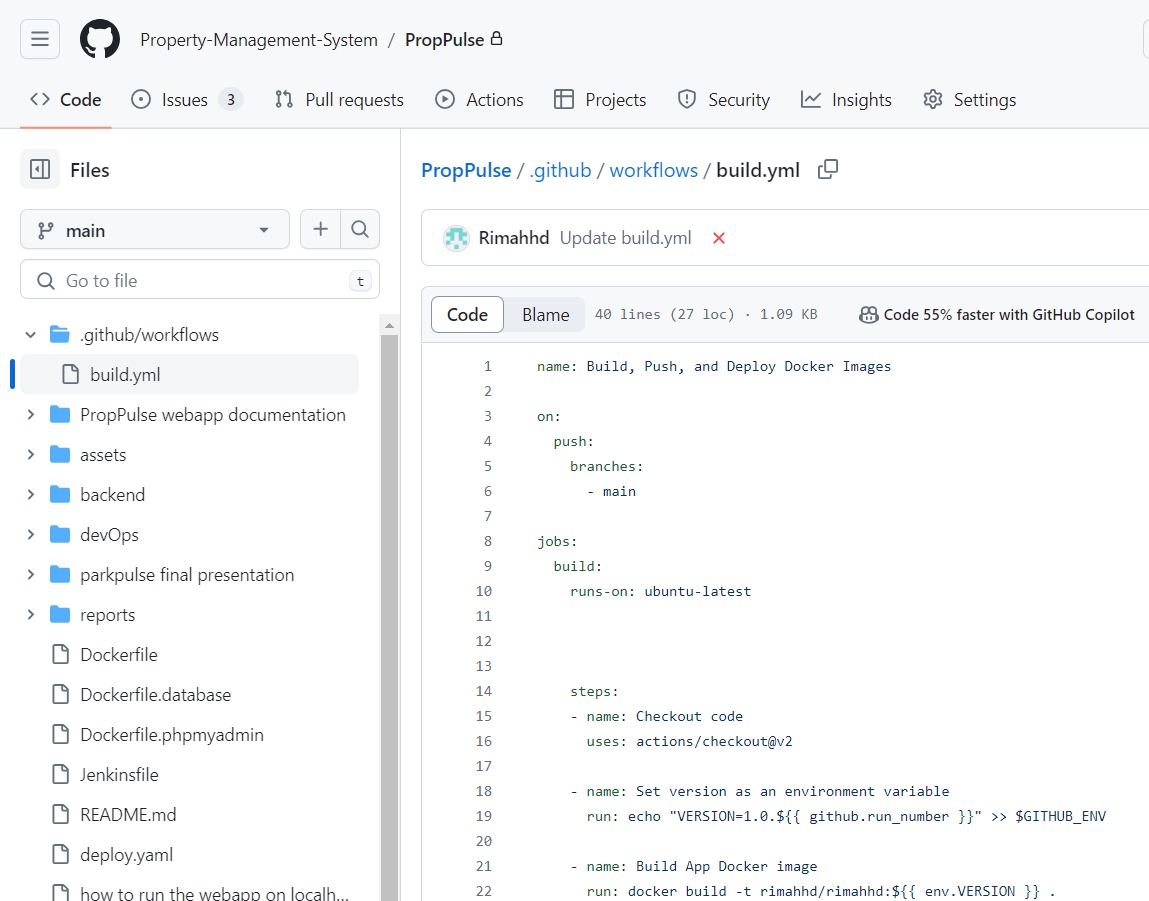
To create a DockerHub repository, log in to your DockerHub account, navigate to the dashboard, and click on the "Create Repository" button. Then, specify the repository name, description, visibility (public or private), and any other desired settings. Finally, click "Create" to finalize the creation of your DockerHub repository.



### Build docker images

To build three Docker images for frontend, backend, and database using GitHub workflows, you'll need to set up a workflow file (e.g., **.github/workflows/docker-build.yml**) in your repository.

Within this file, define jobs for building each image using Docker build commands.



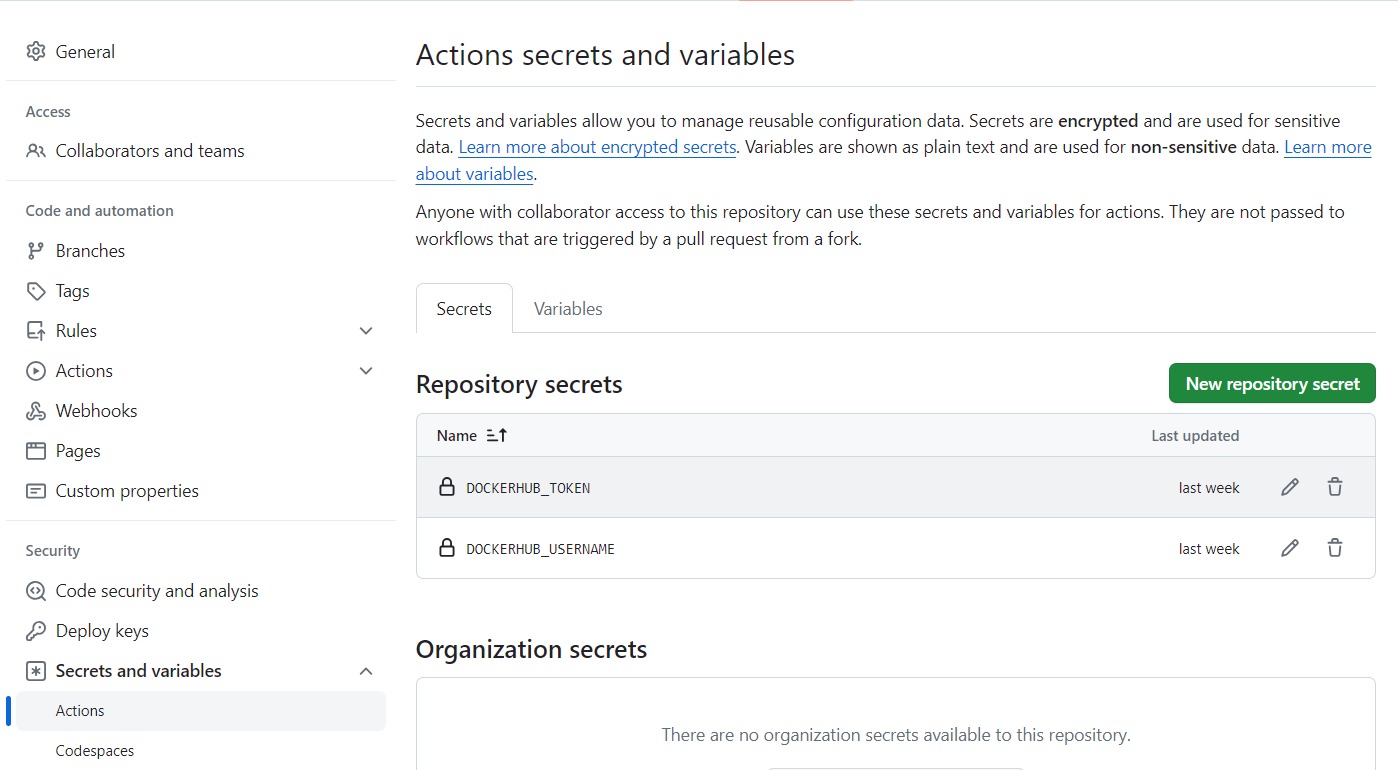
yamlCopy

your database code

Commit this workflow file to your repository, and each time you push changes to the specified branch, GitHub Actions will automatically trigger the workflow to build your Docker images. Make sure to authenticate with DockerHub in your workflow if your images are pushed to DockerHub.

Note on how to create secrets:

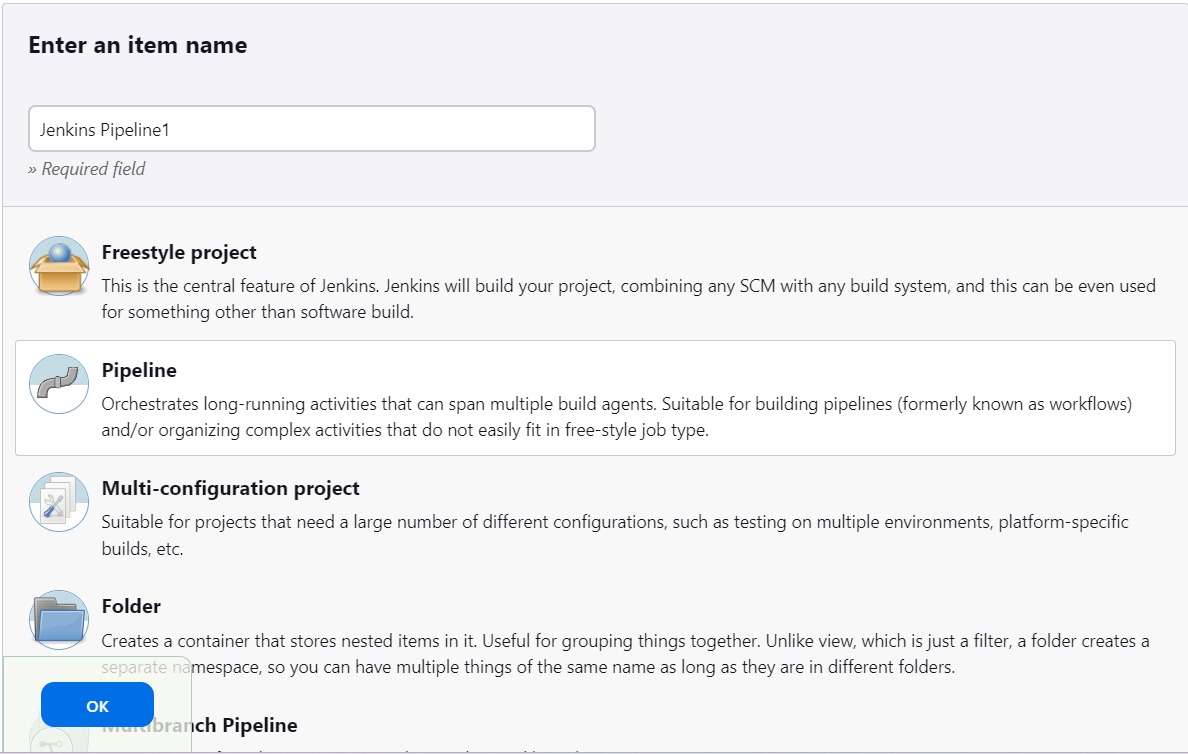
To create GitHub secrets, go to your repository's settings, navigate to the "Secrets" section, and click "New repository secret", then add the secret name and value, and save it.



### Build a pipeline on jenkins

To build a Jenkins pipeline, follow these steps:

1. Install Jenkins
2. Install Pipeline plugins
3. Create a Jenkinsfile
4. Define stages and steps
5. Configure Jenkins job
6. Run the job
7. Monitor pipeline execution
8. Iterate and improve

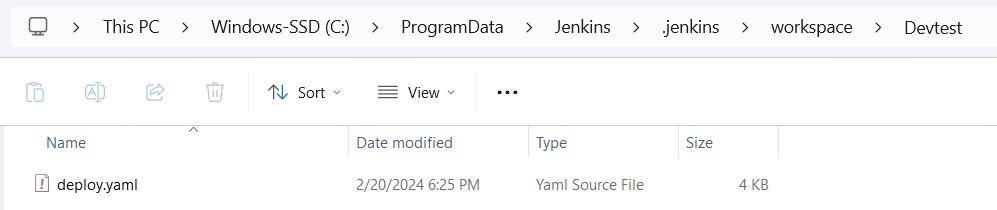


### Add the required credentials

To add credentials in Jenkins:

1. Navigate to the Jenkins dashboard and click on "Manage Jenkins".
2. Select "Manage Credentials" from the options.
3. Click on the "Global credentials" domain (or any other domain where you want to store your credentials).
4. Click on "Add Credentials" on the left sidebar.
5. Choose the type of credential you want to add (e.g., Username with password, SSH username with private key, Secret text, etc.).
6. Fill in the required fields with your credential information.
7. Optionally, provide an ID and description for the credential.
8. Click "OK" to save the credential.

Finally, add a deploy.yaml file

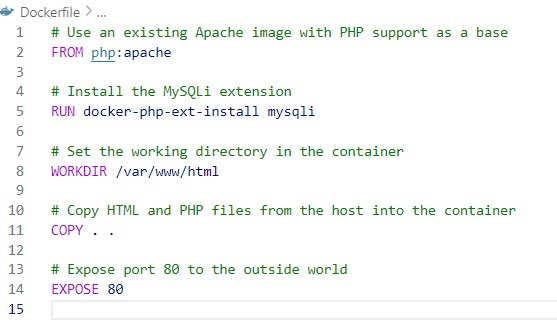
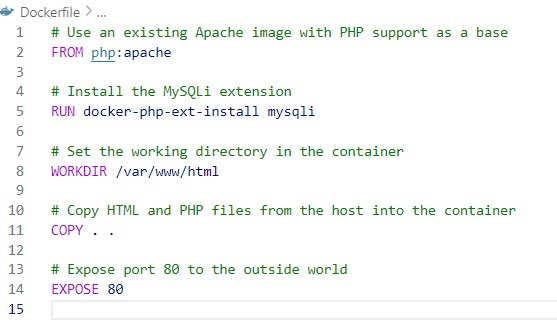
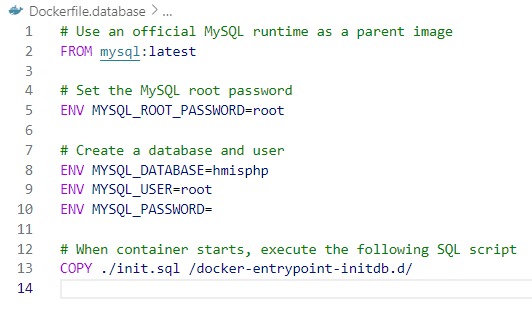


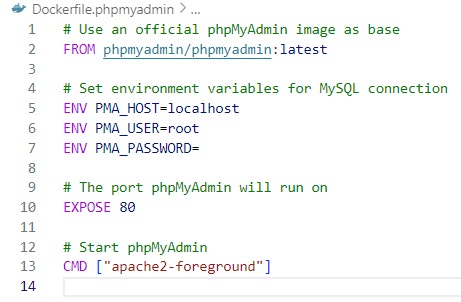
The provided YAML file defines Kubernetes resources for deploying a web application, MySQL database, and PHPMyAdmin:

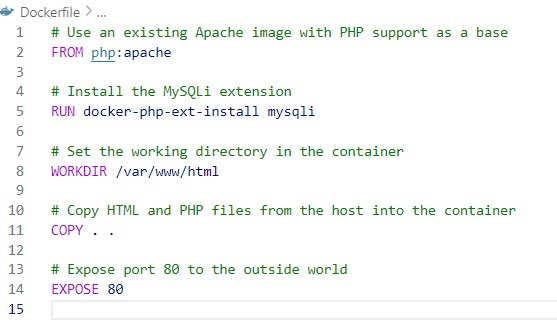
* **Web Application**:
  + Exposed through a NodePort service on port 80.
  + Deployed using a single replica Deployment.
* **MySQL Database**:
  + Exposed internally using a ClusterIP service on port 3306.
  + Managed by a StatefulSet for persistent storage.
* **PHPMyAdmin**:
  + Exposed through a NodePort service on port 8080.
  + Deployed using a single replica Deployment to manage the MySQL database.

These resources are configured with appropriate labels, ports, and environment variables for inter-component communication and functionality. Adjustments may be necessary based on specific deployment requirements and configurations.

### Create DB, FE & BE docker files

Top of Form





Start Minikube

To start Minikube using Docker engine, follow these steps:

1. **Install Docker**: If you haven't already, download and install Docker from the official Docker website according to your operating system.
2. **Install Minikube**: Download and install Minikube by following the instructions provided in the Minikube documentation, ensuring that you have the latest version installed.
3. **Start Minikube**: Open a terminal or command prompt and run the following command to start Minikube with the Docker driver:

sqlCopy code

minikube start --driver=docker

This command initializes Minikube with Docker as the driver for managing the Kubernetes cluster.

1. **Verify Minikube**: Once Minikube has started successfully, verify its status by running:

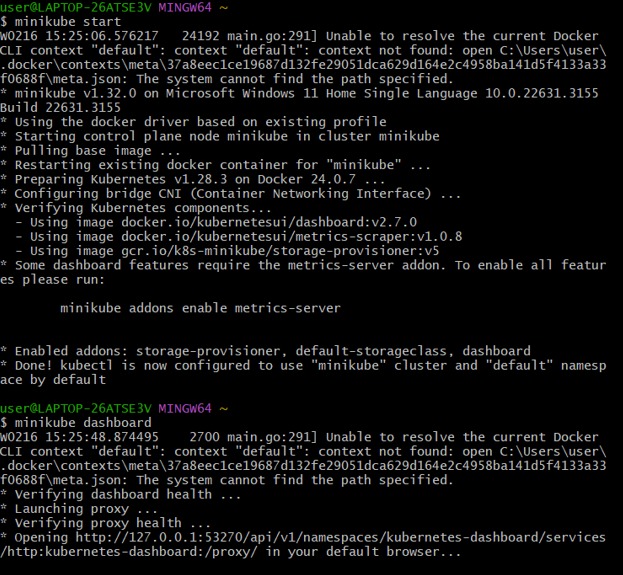
luaCopy code

minikube status

You should see the status of Minikube as "Running".

1. **Manage your Kubernetes cluster**: You can now use Minikube to manage your Kubernetes cluster locally. You can interact with the cluster using kubectl, the Kubernetes command-line tool.

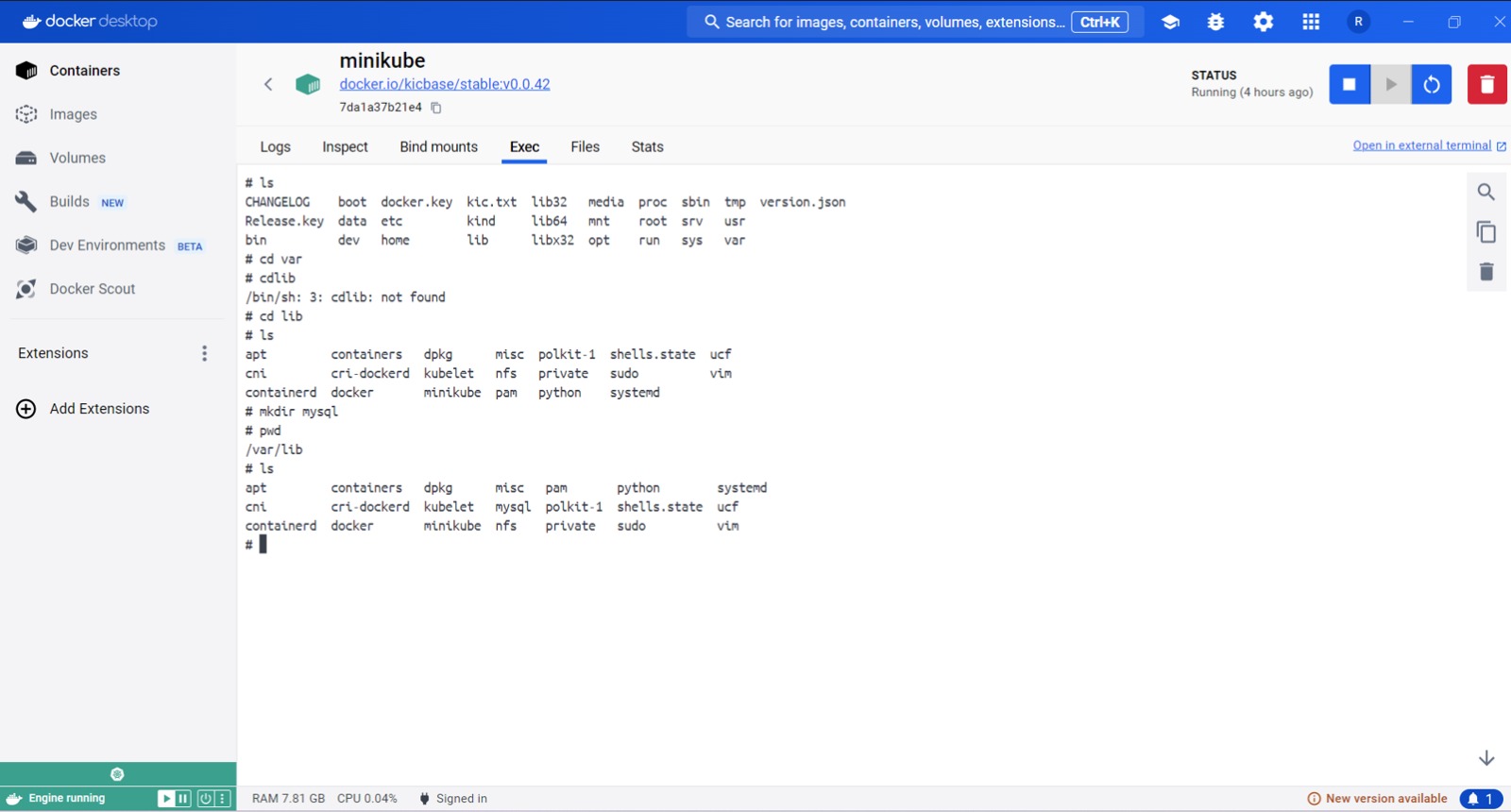
By following these steps, you can start Minikube using Docker as its driver, enabling you to run a Kubernetes cluster locally for development and testing purposes.



Here Docker is taken by default, so no need to mention –driver in the CLI

### Create a database directory

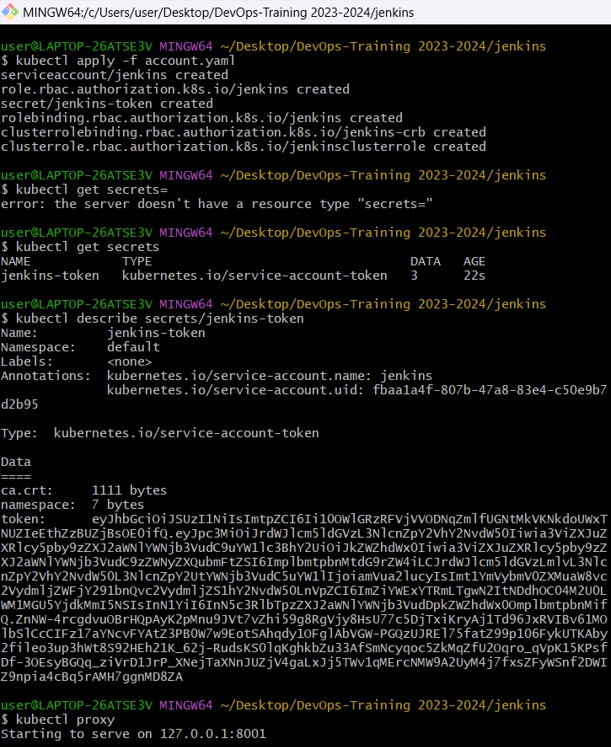
* Navigate to minikube container on docker desktop
* Navigate to exec



* Create a mysql directory, as appears in the above image

### Jenkins Pipeline execution

### Apply Kubernetes Configuration for Account Resources



This command applies the configuration defined in the "account.yaml" file to Kubernetes, creating or updating resources related to user accounts according to the specifications provided in the YAML file.

apiVersion: v1

kind: ServiceAccount

metadata:

name: jenkins

namespace: default

---

kind: Role

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: jenkins

namespace: default

rules:

- apiGroups: [""]

resources: ["pods","services"]

verbs: ["create","delete","get","list","patch","update","watch"]

- apiGroups: ["apps"]

resources: ["deployments"]

verbs: ["create","delete","get","list","patch","update","watch"]

- apiGroups: [""]

resources: ["pods/exec"]

verbs: ["create","delete","get","list","patch","update","watch"]

- apiGroups: [""]

resources: ["pods/log"]

verbs: ["get","list","watch"]

- apiGroups: [""]

resources: ["secrets"]

verbs: ["get"]

- apiGroups: [""]

resources: ["persistentvolumeclaims"]

verbs: ["create","delete","get","list","patch","update","watch"]

---

apiVersion: v1

kind: Secret

metadata:

name: jenkins-token

annotations:

kubernetes.io/service-account.name: jenkins

type: kubernetes.io/service-account-token

---

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

name: jenkins

namespace: default

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: Role

name: jenkins

subjects:

- kind: ServiceAccount

name: jenkins

---

# Allows jenkins to create persistent volumes

# This cluster role binding allows anyone in the "manager" group to read secrets in any namespace.

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: jenkins-crb

subjects:

- kind: ServiceAccount

namespace: default

name: jenkins

roleRef:

kind: ClusterRole

name: jenkinsclusterrole

apiGroup: rbac.authorization.k8s.io

---

kind: ClusterRole

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: jenkinsclusterrole

rules:

- apiGroups: [""]

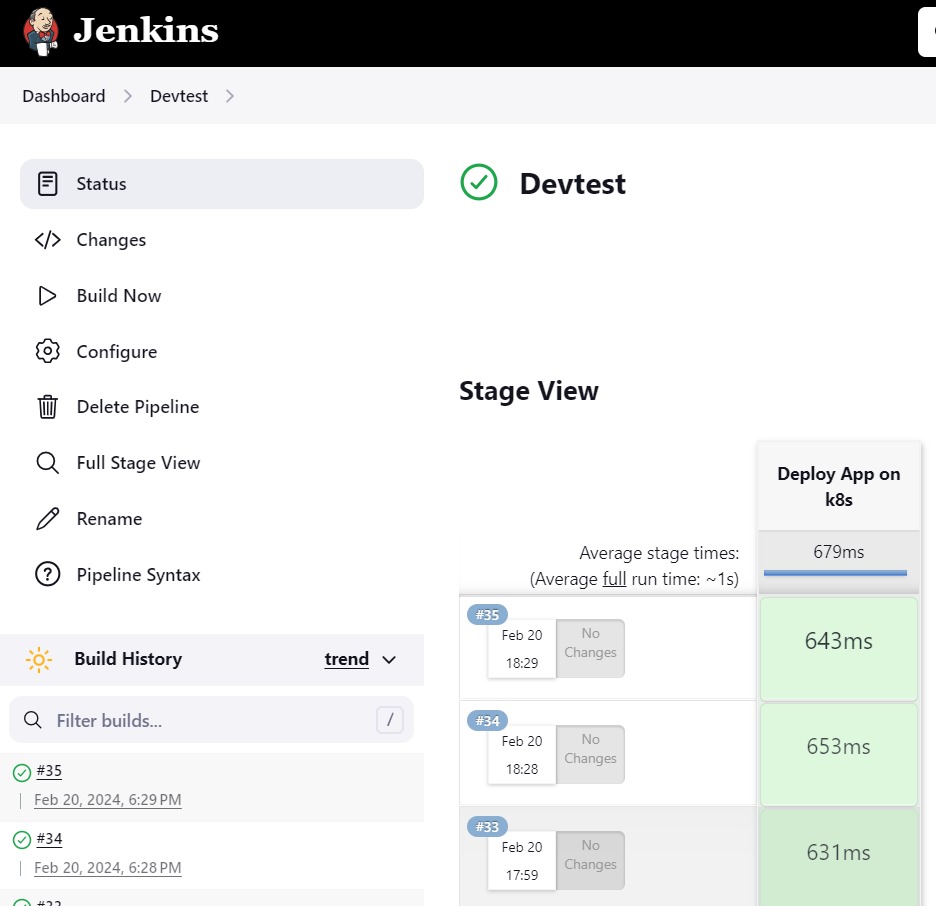
resources: ["persistentvolumes"]

verbs: ["create","delete","get","list","patch","update","watch"]

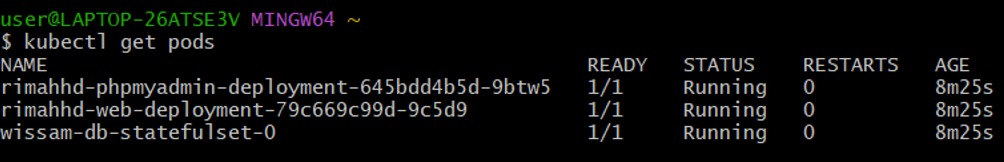
Top of Form

These Kubernetes manifests define resources and permissions necessary for Jenkins to interact with the Kubernetes cluster. It includes a ServiceAccount, Role, Secret, RoleBinding, ClusterRoleBinding, and ClusterRole. These resources grant Jenkins appropriate permissions for managing pods, services, deployments, secrets, and persistent volume claims within the default namespace. Additionally, it allows Jenkins to create and manage persistent volumes across the entire cluster.

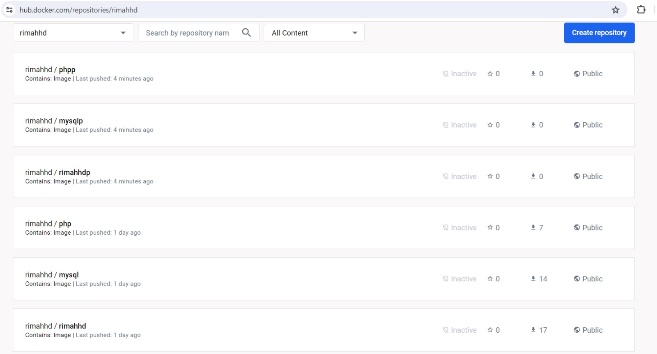
Then run kubectl proxy that initiates a proxy server, facilitating a connection between your local system and the Kubernetes API server. This enables you to interact with the Kubernetes API directly from your local machine, enhancing accessibility and ease of management.

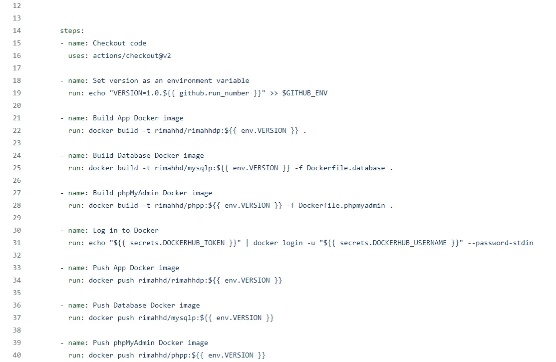


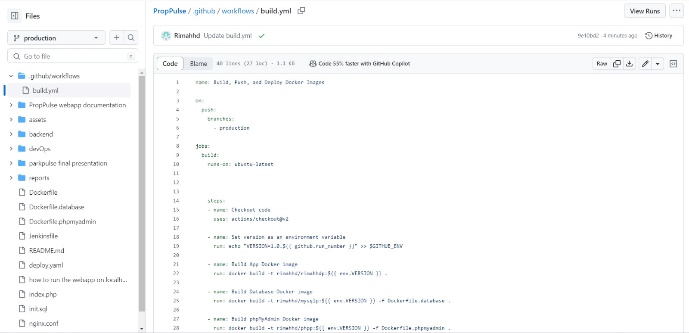
Then wait till the pods finish running



### Creating a Production Environment







* Creating a production branch

