SWE 645 Homework / Assignment 3

Author of document: Qasim Shahid
Date Due: 4/16/2025
Note: THIS DOCUMENT DOES NOT CONTAIN API TESTING. PLEASE WATCH THE VIDEO INCLUDED WITH THIS SUBMISSION TO SEE A FULL TEST USING POSTMAN. THIS DOCUMENT ONLY COVERS HOW TO SET THIS UP!

GitHub Repository: https://github.com/qasimshahid/qshahid-swe645-asst3/
Note: This repo is currently public but will be made private after this assignment is graded.
Another note: I ran out of budget on my AWS Learner Lab account, so this is being hosted on my own AWS account.

Website link (this will be available via Elastic IP and will be accessible since I will keep it up on my own AWS account until it is graded).

- http://98.82.89.116:30007/api/surveys/version
- (This is the link for an HTML page that shows the version of the app so that you can just see that it is up and running)

Other endpoints and what HTML method to use to access them
(Please replace any parameters in {} with your own specified parameters, such as ID of surveys)

- To get a survey by ID
  - GET
  - http://98.82.89.116:30007/api/surveys/{id}
- To get all surveys
  - GET
  - http://98.82.89.116:30007/api/surveys
- To delete a survey by ID
  - DELETE
  - http://98.82.89.116:30007/api/surveys/{id}
- To post a new survey
  - POST (Must send a valid survey in HTML Request Body via JSON, please see the README.md of the GitHub repository for examples.)
  - http://98.82.89.116:30007/api/surveys
- To update an existing survey by ID
  - PUT (Must send a valid survey in HTML Request Body via JSON, please see the README.md of the GitHub repository for examples.)
  - http://98.82.89.116:30007/api/surveys/{id}

In this homework assignment, we are tasked to build a CI/CD pipeline using a very simple REST API for doing CRUD operations on survey data. We containerize the Spring Web REST API using Docker and then deploy it to a Kubernetes cluster, running at least 3 replicas. We use Rancher as a Kubernetes management platform to manage our cluster.

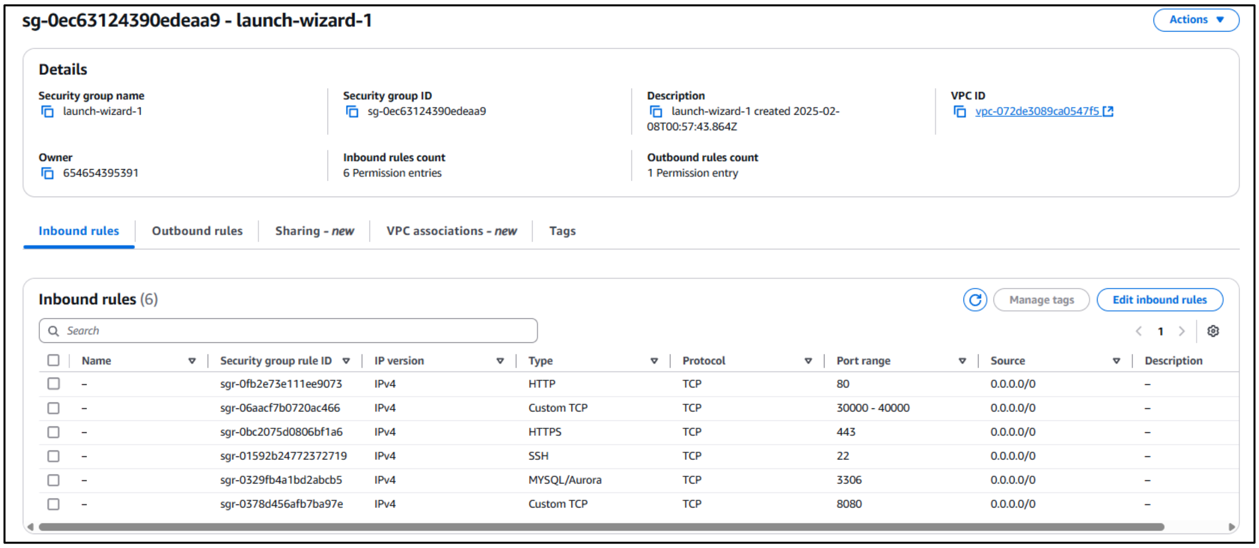
The Jenkins pipeline is triggered upon a push to the main branch of the GitHub repository. The pipeline does the following:

1. Get the most recent source code by cloning the repository
2. Build the jar file using the repository we cloned.
3. Build the docker image defined in the Dockerfile using docker build
4. Push the image we just built to Docker Hub (this requires an account, by default, the source code assumes mine is being used with username qshahid)
5. Deploys the latest container image for our app the Kubernetes cluster, creating a Deployment with 3 replicas (basically deploying the container to 3 pods). The app is exposed externally via a NodePort service, with port 30007 mapped to the container's port 8080, allowing external access at http://<public-cluster-ip>:30007/api/surveys/version

Please make sure you have a Docker Hub account, an AWS account, and a GitHub account. Please also clone/fork my GitHub repository to your own account so you can configure the webhook that allows us to automatically build and deploy whenever we push to the main branch. Please also make this GitHub repository public as that is what I used. Continue reading to see all the other things that are required to make this work.

AWS Security Group Setup

Please duplicate / copy the following security group before doing anything since we will be assigning this security group to multiple things. This will make sure you are able to run without having port forwarding issues.



sg-0ec63124390edeaa9 - launch-wizard-1

Details

Security group name: launch-wizard-1

Security group ID: sg-0ec63124390edeaa9

Description: launch-wizard-1 created 2025-02-08T00:57:43.864Z

VPC ID: vpc-072de3089ca0547f5

Owner: 654654395391

Inbound rules count: 6 Permission entries

Outbound rules count: 1 Permission entry

Inbound rules (6)

□	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
□	–	sgr-0fb2e73e111ee9073	IPv4	HTTP	TCP	80	0.0.0.0/0	–
□	–	sgr-06aacf7b0720ac466	IPv4	Custom TCP	TCP	30000 - 40000	0.0.0.0/0	–
□	–	sgr-0bc2075d0806bf1a6	IPv4	HTTPS	TCP	443	0.0.0.0/0	–
□	–	sgr-01592b24772372719	IPv4	SSH	TCP	22	0.0.0.0/0	–
□	–	sgr-0329fb4a1bd2abcb5	IPv4	MYSQL/Aurora	TCP	3306	0.0.0.0/0	–
□	–	sgr-0378d456afb7ba97e	IPv4	Custom TCP	TCP	8080	0.0.0.0/0	–

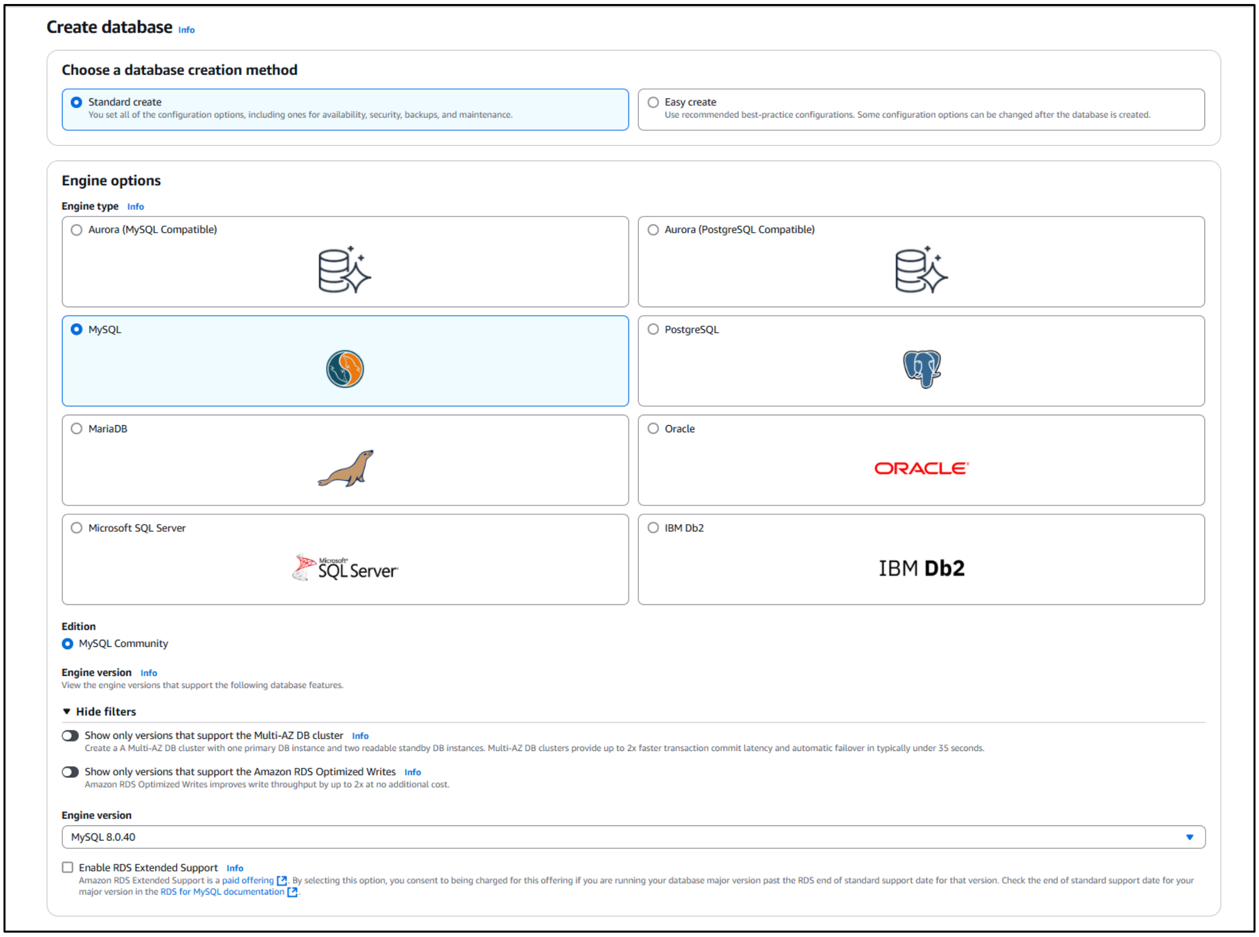
AWS Database Setup

Since we need to store our data somewhere, we need to create a free-tier MySQL database (or Amazon RDS) database. We need to make it publicly available so that our API can access it and we also need to create some credentials that we will remember, which will also be used in the deployment process so we can access the database.

Please also install MySQL Workbench on your computer so you can validate API operations: https://www.mysql.com/products/workbench/

Go to Amazon RDS and create a new database. Copy these settings:

Use MySQL



Create database Info

Choose a database creation method

● Standard create
You set all of the configuration options, including ones for availability, security, backups, and maintenance.

○ Easy create
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type Info

○ Aurora (MySQL Compatible)

○ Aurora (PostgreSQL Compatible)

● MySQL

○ PostgreSQL

○ MariaDB

○ Oracle

○ Microsoft SQL Server

○ IBM Db2

Edition

● MySQL Community

Engine version Info
View the engine versions that support the following database features.

▼ **Hide filters**

Show only versions that support the Multi-AZ DB cluster Info
Create a A Multi-AZ DB cluster with one primary DB instance and two readable standby DB instances. Multi-AZ DB clusters provide up to 2x faster transaction commit latency and automatic failover in typically under 35 seconds.

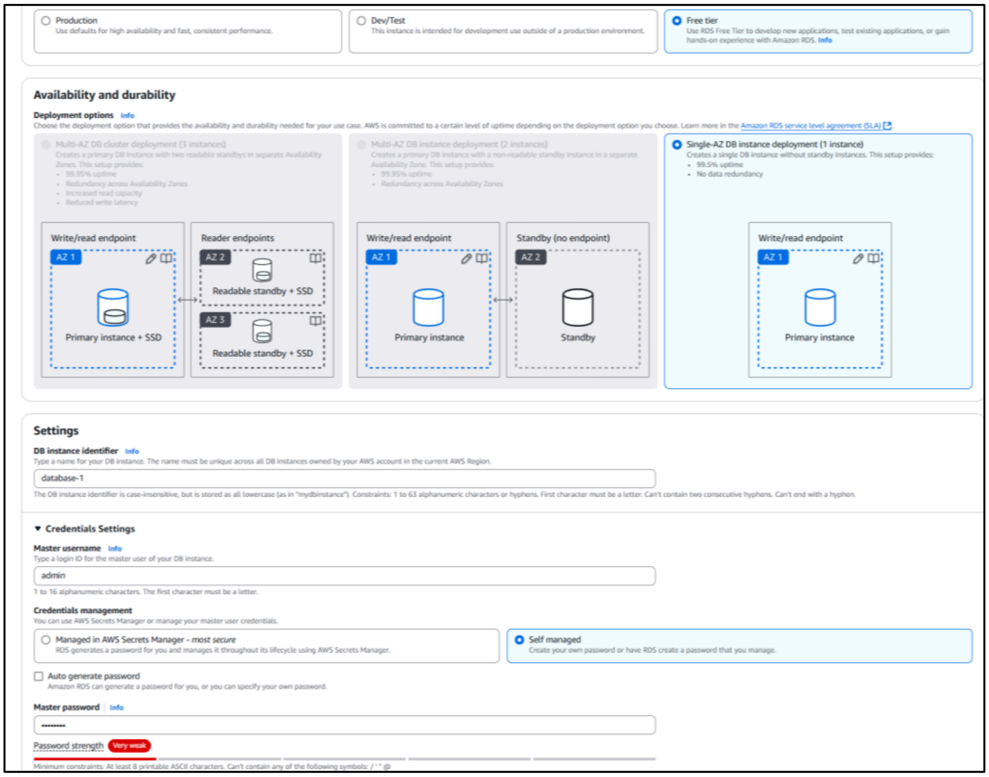
Show only versions that support the Amazon RDS Optimized Writes Info
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

Engine version

MySQL 8.0.40 ▼

□ Enable RDS Extended Support Info
Amazon RDS Extended Support is a paid offering. By selecting this option, you consent to being charged for this offering if you are running your database major version past the RDS end of standard support date for that version. Check the end of standard support date for your major version in the RDS for MySQL documentation.

Next, make sure to choose the free tier DB. Make a self-managed username and password. Note this down since we will need this to connect.



○ Production
Use defaults for high availability and fast, consistent performance.

○ Dev/Test
This instance is intended for development use outside of a production environment.

● Free tier
Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS. Info

Availability and durability

Deployment options Info
Choose the deployment option that provides the availability and durability needed for your use case. AWS is committed to a certain level of uptime depending on the deployment option you choose. Learn more in the Amazon RDS service level agreement (SLA).

○ Multi-AZ DB cluster deployment (3 instances)
Creates a primary DB instance with two readable standbys in separate Availability Zones. This setup provides:
- 99.95% uptime
- Redundancy across Availability Zones
- Increased read capacity
- Reduced write latency

○ Multi-AZ DB instance deployment (2 instances)
Creates a primary DB instance with a non-readable standby instance in a separate Availability Zone. This setup provides:
- 99.95% uptime
- Redundancy across Availability Zones

● Single-AZ DB instance deployment (1 instance)
Creates a single DB instance without standby instances. This setup provides:
- 99.5% uptime
- No data redundancy

Settings

DB instance identifier Info
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.
database-1
The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 63 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

▼ **Credentials Settings**

Master username Info
Type a login ID for the master user of your DB instance.
admin
1 to 16 alphanumeric characters. The first character must be a letter.

Credentials management
You can use AWS Secrets Manager or manage your master user credentials.

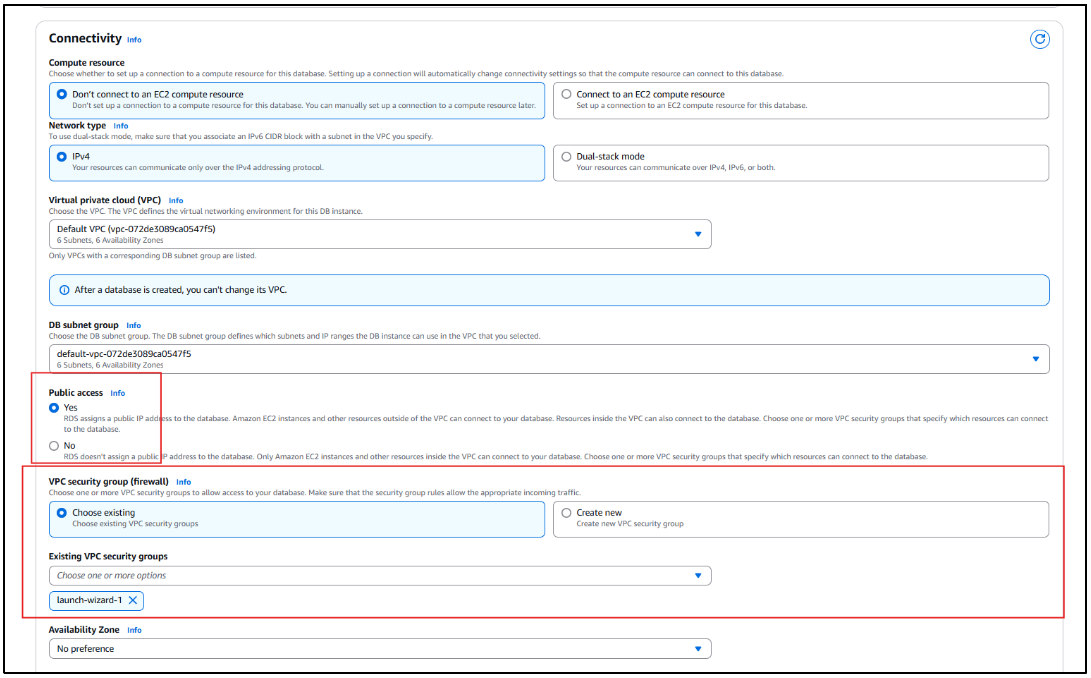
○ Managed in AWS Secrets Manager - *most secure*
RDS generates a password for you and manages it throughout its lifecycle using AWS Secrets Manager.

● Self managed
Create your own password or have RDS create a password that you manage.

□ Auto generate password
Amazon RDS can generate a password for you, or you can specify your own password.

Master password | Info
•••••••••
Password strength Very weak
Minimum constraints: At least 8 printable ASCII characters. Can't contain any of the following symbols: / ' " @

Then, there will be a bunch of options. Scroll down to connectivity and copy the following:



Connectivity Info

Compute resource
Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

● Don't connect to an EC2 compute resource
Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.

○ Connect to an EC2 compute resource
Set up a connection to an EC2 compute resource for this database.

Network type Info
To use dual-stack mode, make sure that you associate an IPv6 CIDR block with a subnet in the VPC you specify.

● IPv4
Your resources can communicate only over the IPv4 addressing protocol.

○ Dual-stack mode
Your resources can communicate over IPv4, IPv6, or both.

Virtual private cloud (VPC) Info
Choose the VPC. The VPC defines the virtual networking environment for this DB instance.
Default VPC (vpc-072de3089ca0547f5)
6 Subnets, 6 Availability Zones
Only VPCs with a corresponding DB subnet group are listed.

ⓘ After a database is created, you can't change its VPC.

DB subnet group Info
Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.
default-vpc-072de3089ca0547f5
6 Subnets, 6 Availability Zones

Public access Info

● Yes
RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

○ No
RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

VPC security group (firewall) Info
Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

● Choose existing
Choose existing VPC security groups

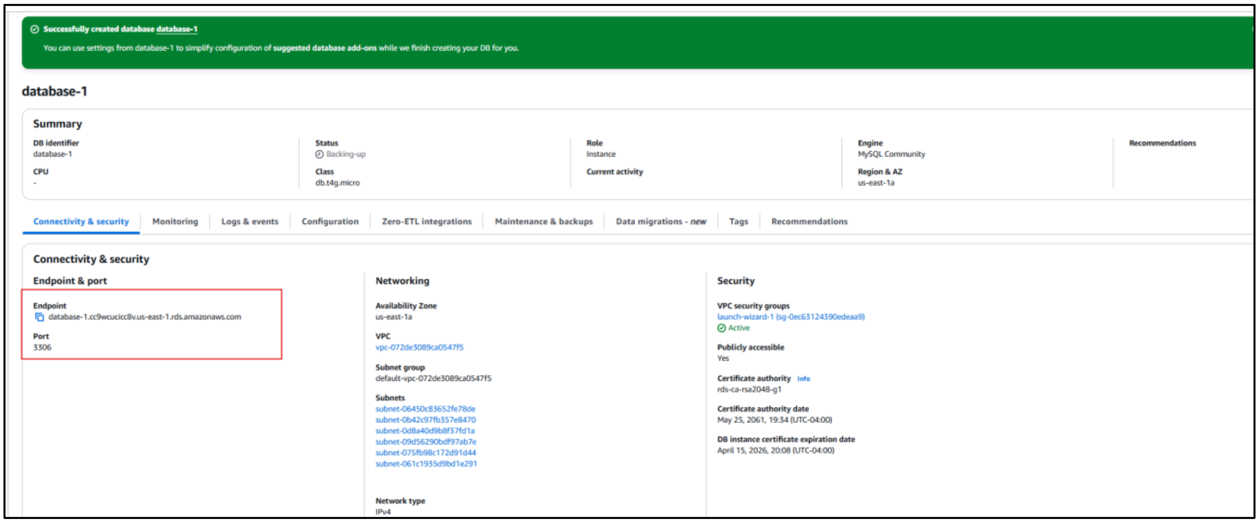
○ Create new
Create new VPC security group

Existing VPC security groups
Choose one or more options
launch-wizard-1 ✕

Availability Zone Info
No preference

After this, click “Create database”

Now, open MySQL Workbench. Try to connect to your database using the IP, 3306 port, and your username and password that you set.



Successfully created database database-1

You can use settings from database-1 to simplify configuration of **suggested database add-ons** while we finish creating your DB for you.

database-1

Summary

DB identifier
database-1

CPU
-

Status
Backing-up

Class
db.t4g.micro

Role
Instance

Current activity

Engine
MySQL Community

Region & AZ
us-east-1a

Recommendations

Connectivity & security

Endpoint & port

Endpoint
database-1.cc9wcucicc8v.us-east-1.rds.amazonaws.com

Port
3306

Networking

Availability Zone
us-east-1a

VPC
vpc-072de3089ca0547f5

Subnet group
default-vpc-072de3089ca0547f5

Subnets
subnet-06450c83652fe78de
subnet-0b42c97fb357e8470
subnet-0d8a40d9b8f37fd1a
subnet-09d56290bdf97ab7e
subnet-075fb98c172d91d44
subnet-061c1935d9bd1e291

Network type
IPv4

Security

VPC security groups
launch-wizard-1 (sg-0ec63124390edeaa9)
Active

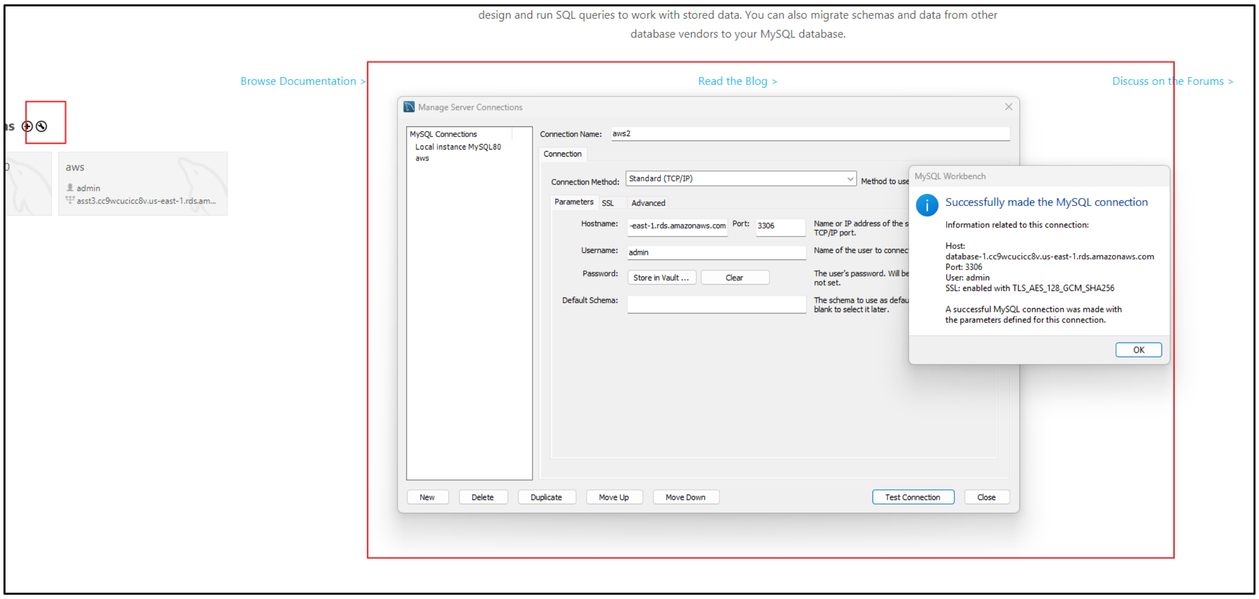
Publicly accessible
Yes

Certificate authority Info
rds-ca-rsa2048-g1

Certificate authority date
May 25, 2061, 19:34 (UTC-04:00)

DB instance certificate expiration date
April 15, 2026, 20:08 (UTC-04:00)

You should see this if you did everything correctly. Create the connection and connect to the DB. We need to execute one command in it.



design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

Connection Name: aws2

Connection Method: Standard (TCP/IP) Method to use

Hostname: -east-1.rds.amazonaws.com Port: 3306 Name or IP address of the s TCP/IP port.

Username: admin Name of the user to connec

Password: The user's password. Will be not set.

Default Schema: The schema to use as defau blank to select it later.

MySQL Workbench

Successfully made the MySQL connection

Information related to this connection:

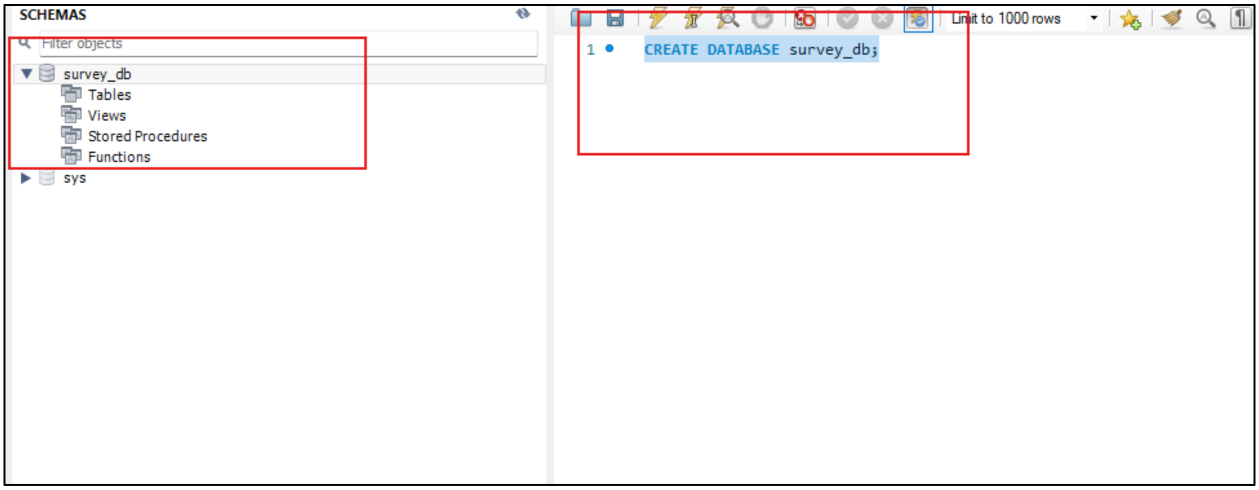
Host:
database-1.cc9wcucicc8v.us-east-1.rds.amazonaws.com
Port: 3306
User: admin
SSL: enabled with TLS_AES_128_GCM_SHA256

A successful MySQL connection was made with
the parameters defined for this connection.

OK

Once connected, please go to the GitHub repository and go to the createDb.sql file and copy the command to create the database. This is the only SQL command we need to run.

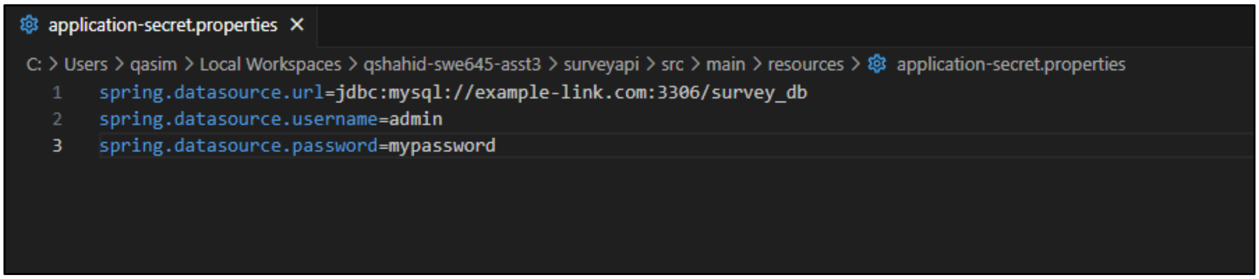
“CREATE DATABASE survey_db;”



```sql
CREATE DATABASE survey_db;
```

Now, you are done with database setup. Note down the AWS DB link and your credentials. We will be injecting these credentials into our Jenkins instance by creating a new credential that uses a secret file. This will be “application-secret.properties” so that our API can connect to the database.

Create a file exactly like this but use your own DB link and credentials. Note that I specify the database I want to connect to, survey_db, and you should do this as well. We created this database in the previous step.


application-secret.properties ✕
C: > Users > qasim > Local Workspaces > qshahid-swe645-asst3 > surveyapi > src > main > resources > application-secret.properties

```properties
spring.datasource.url=jdbc:mysql://example-link.com:3306/survey_db
spring.datasource.username=admin
spring.datasource.password=mypassword
```

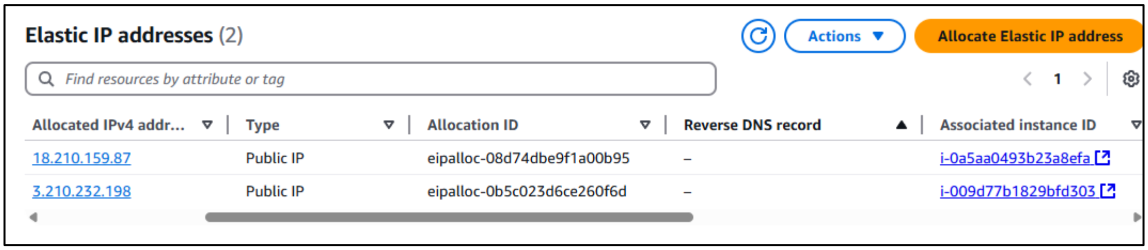
Have it be the same name as well, application-secret.properties. Like I said, we will give this file to Jenkins.

Once the file has been created, save it somewhere where you will remember.

AWS EC2 Setup

This assignment makes extensive use of AWS. Please keep that in mind since this might cost you money.

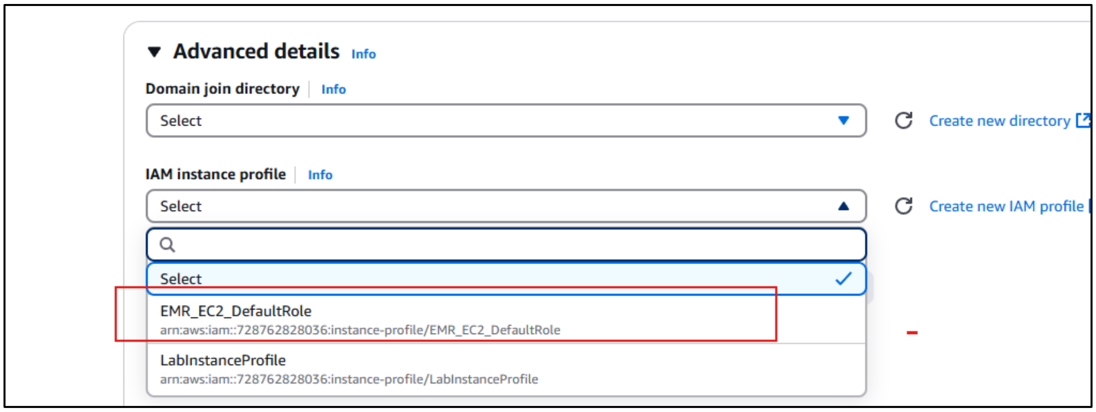
- 2 EC2 machines:
  - 1 t2.large machine, which will host Rancher
  - 1 t2.large machine, which will host Jenkins
  - Please use Ubuntu on all of these machines and configure them to have 30 GB of storage.
  - From this point on, I will refer to them as the Rancher instance and the Jenkins instance.

- Elastic IPs
  - Please attach an elastic IP to both of these machines as it makes it easier to work with.
  - 

Elastic IP addresses (2)

Actions ▼
Allocate Elastic IP address
Find resources by attribute or tag
1

Allocated IPv4 addr...	Type	Allocation ID	Reverse DNS record	Associated instance ID
18.210.159.87	Public IP	eipalloc-08d74dbe9f1a00b95	–	i-0a5aa0493b23a8efa
3.210.232.198	Public IP	eipalloc-0b5c023d6ce260f6d	–	i-009d77b1829bfd303

- Please add this setting in “Advanced” when creating your instances (if you are using AWS Learner Lab, which I am not using):
  - 

▼ Advanced details Info

Domain join directory | Info

Select

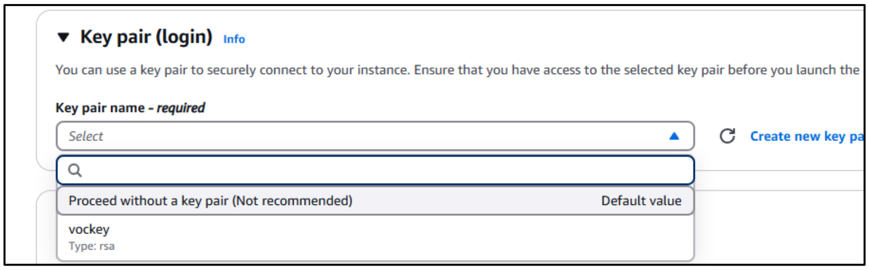
IAM instance profile | Info

Select

Select ✓

EMR_EC2_DefaultRole
arn:aws:iam::728762828036:instance-profile/EMR_EC2_DefaultRole

LabInstanceProfile
arn:aws:iam::728762828036:instance-profile/LabInstanceProfile

- Please proceed without a key, as we will be using Instance Connect.
  - 

▼ Key pair (login) Info

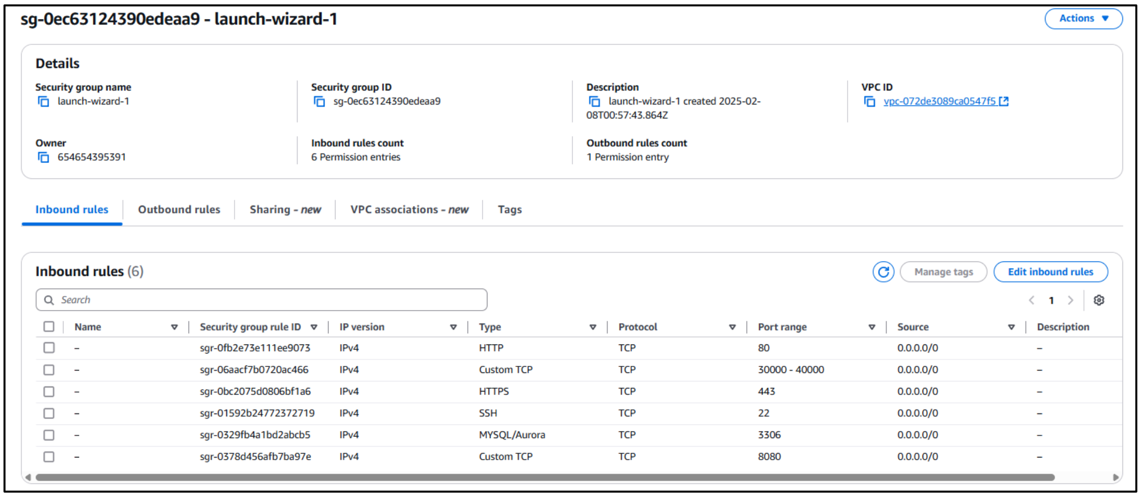
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the

Key pair name - *required*

Select

Proceed without a key pair (Not recommended) Default value

vockey
Type: rsa

- Please create the following security group and apply it to both the Rancher instance and the Jenkins instance.
  - 

    sg-0ec63124390edeaa9 - launch-wizard-1

    Details

Security group name	Security group ID	Description	VPC ID
launch-wizard-1	sg-0ec63124390edeaa9	launch-wizard-1 created 2025-02-08T00:57:43.864Z	vpc-072de3089ca0547f5
Owner	**Inbound rules count**	**Outbound rules count**	
654654395391	6 Permission entries	1 Permission entry	

    Inbound rules (6)

□	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
□	–	sgr-0fb2e73e111ee9073	IPv4	HTTP	TCP	80	0.0.0.0/0	–
□	–	sgr-06aacf7b0720ac466	IPv4	Custom TCP	TCP	30000 - 40000	0.0.0.0/0	–
□	–	sgr-0bc2075d0806bf1a6	IPv4	HTTPS	TCP	443	0.0.0.0/0	–
□	–	sgr-01592b24772372719	IPv4	SSH	TCP	22	0.0.0.0/0	–
□	–	sgr-0329fb4a1bd2abcb5	IPv4	MYSQL/Aurora	TCP	3306	0.0.0.0/0	–
□	–	sgr-0378d456afb7ba97e	IPv4	Custom TCP	TCP	8080	0.0.0.0/0	–
  - Leave outbound rules as the default.

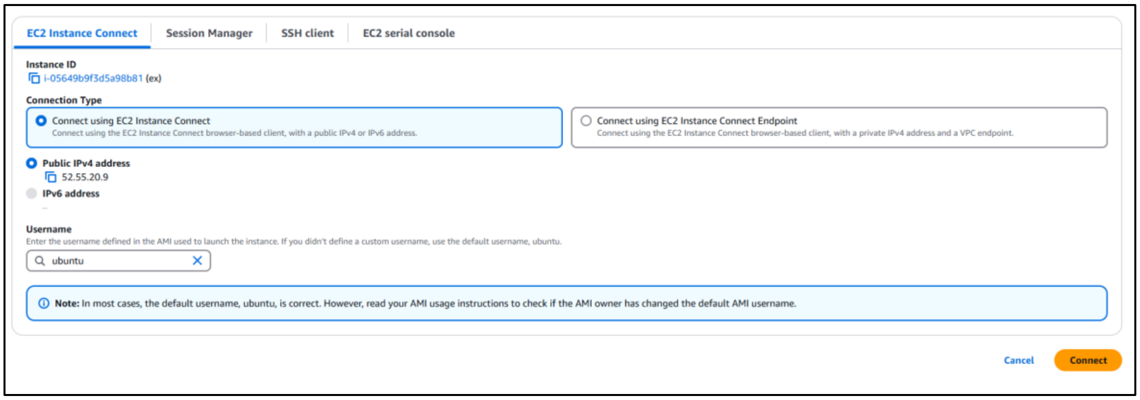
- After you do all this, you should be ready to proceed to the next step of setting up your EC2 machines. In particular, we need to install some dependencies.

For both machines, you need to execute the following commands. These will install some dependencies, such as Docker, kubectl, and generally just update the machines.

```
sudo su
sudo apt-get update -y
sudo apt-get upgrade -y

curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

sudo snap install kubectl --classic
```

Setup Rancher Instance and Cluster

In order to get Rancher on your t2.large instance, do the following.
Login to the instance using instance connect:



Instance ID
i-05649b9f3d5a98b81 (ex)

Connection Type

● Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.

○ Connect using EC2 Instance Connect Endpoint
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

● Public IPv4 address
52.55.20.9

○ IPv6 address
–

Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ubuntu.
ubuntu

Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Run all the commands mentioned at the end of the previous section.

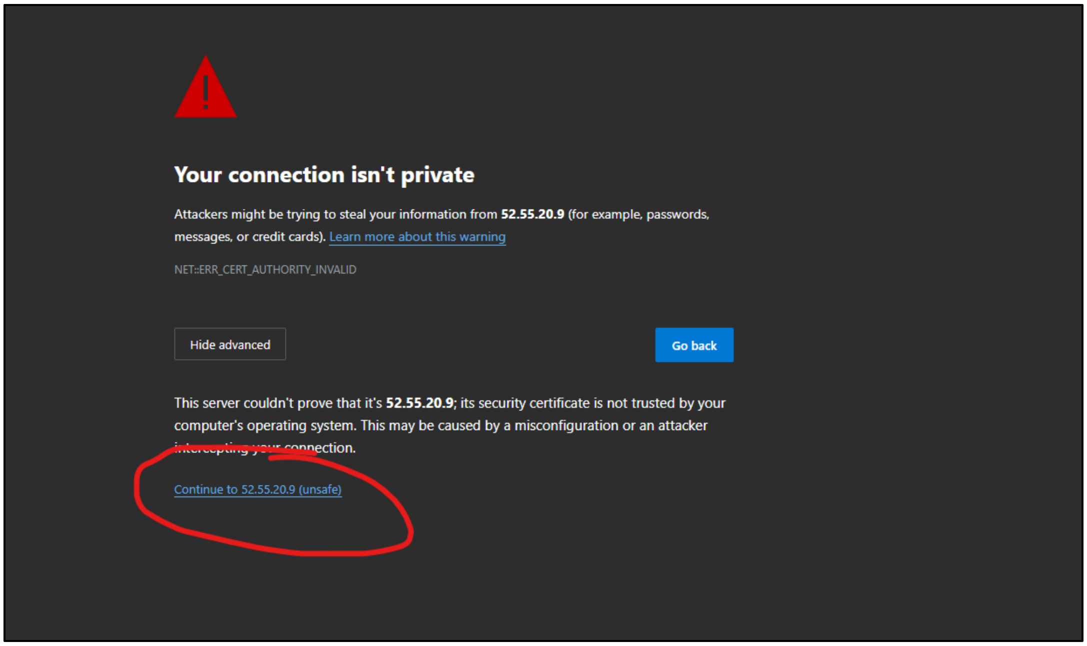
```
sudo su
sudo apt-get update -y
sudo apt-get upgrade -y

curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

sudo snap install kubectl –classic
```

Then, to use Rancher, run the following command:

```
sudo docker run --privileged -d --restart=unless-stopped -p 80:80 -p 443:443
rancher/rancher
```

After that, go to the following link to access Rancher:
http://{ec2-public-ip}:80
For example,
http://52.55.20.9:80

It will tell you the site is not secure, but just proceed past the warnings.



Your connection isn't private

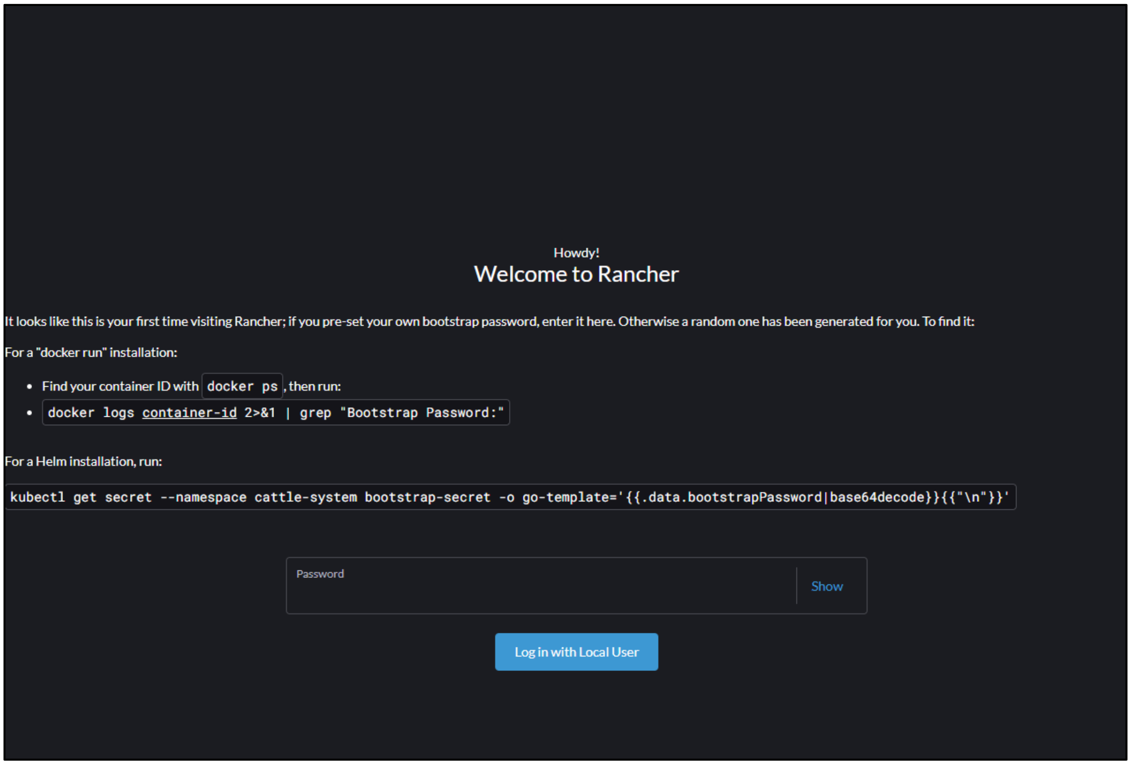
Attackers might be trying to steal your information from **52.55.20.9** (for example, passwords, messages, or credit cards). Learn more about this warning

NET::ERR_CERT_AUTHORITY_INVALID

This server couldn't prove that it's **52.55.20.9**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

Continue to 52.55.20.9 (unsafe)

After that, you will be met by this UI and login page from Rancher.



Howdy!

Welcome to Rancher

It looks like this is your first time visiting Rancher; if you pre-set your own bootstrap password, enter it here. Otherwise a random one has been generated for you. To find it:

For a "docker run" installation:

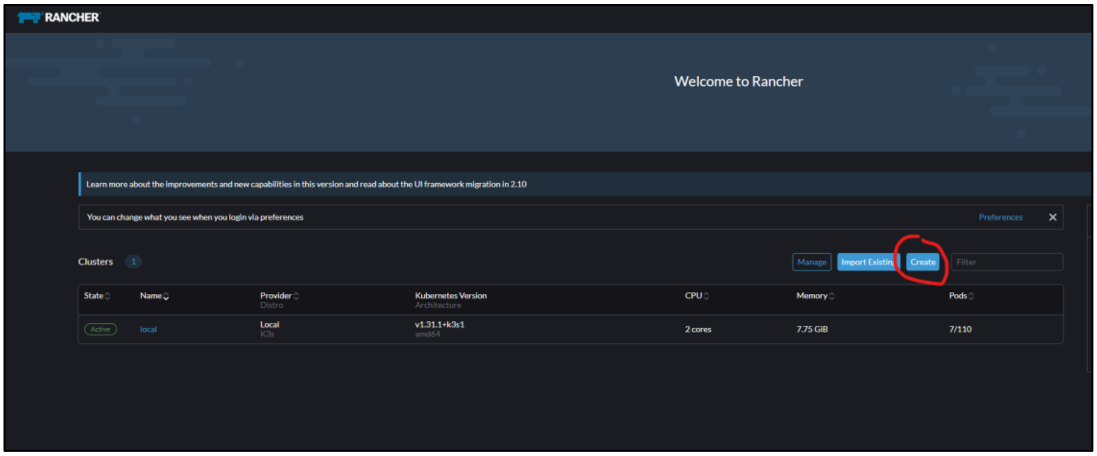
- Find your container ID with `docker ps`, then run:
- `docker logs container-id 2>&1 | grep "Bootstrap Password:"`

For a Helm installation, run:

```
kubectl get secret --namespace cattle-system bootstrap-secret -o go-template='{{.data.bootstrapPassword|base64decode}}{{"\n"}}'
```

Password

Follow the instructions and login to Rancher with the bootstrap password. Create your own password so that you remember the login. The username is always “admin”. For me, I always use “temppass3$”



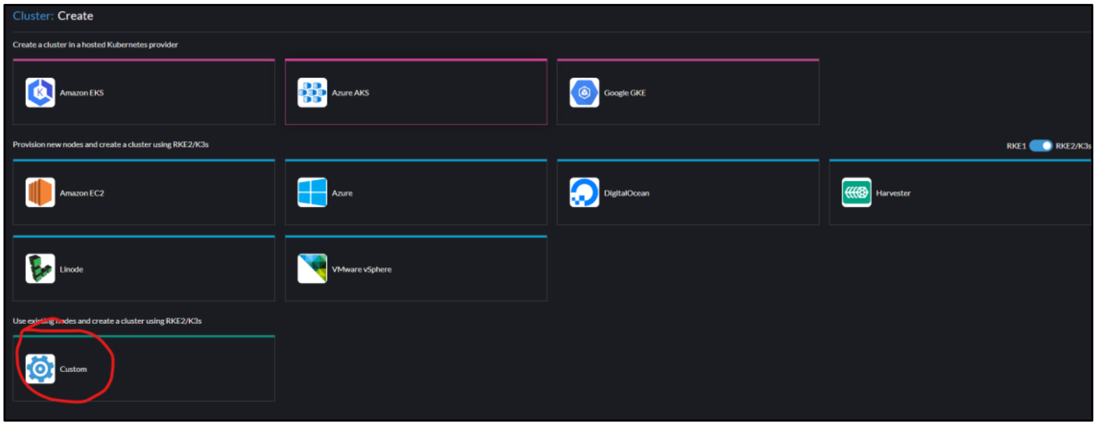
Learn more about the improvements and new capabilities in this version and read about the UI framework migration in 2.10

You can change what you see when you login via preferences

Clusters 1

State	Name	Provider Distro	Kubernetes Version Architecture	CPU	Memory	Pods
Active	local	Local K3s	v1.31.1+k3s1 amd64	2 cores	7.75 GiB	7/110

Let’s create our new cluster. Use RKE2/3 to create a custom cluster.

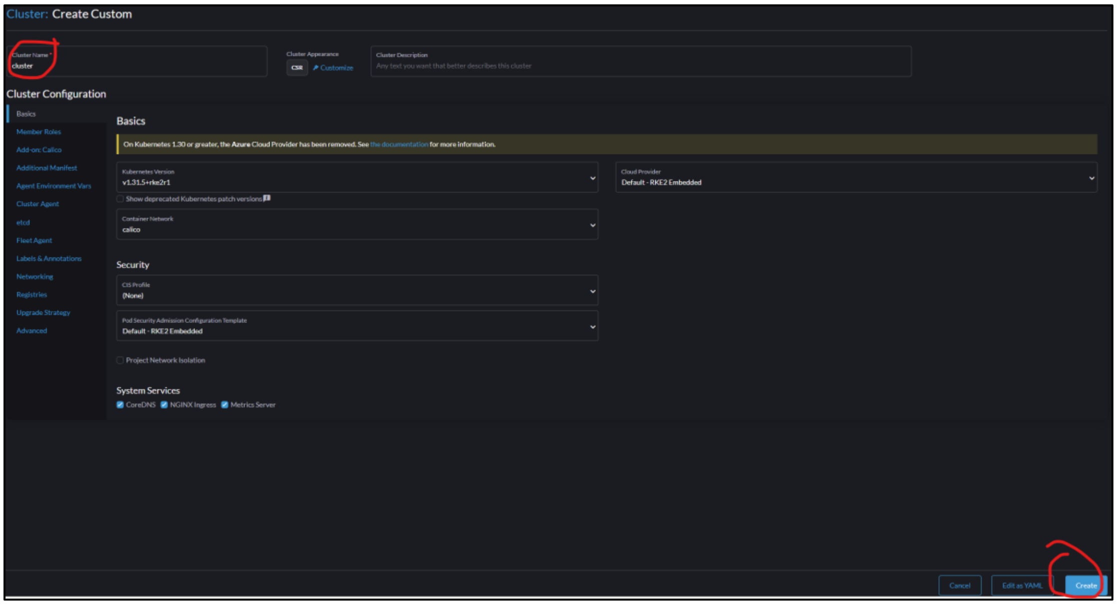


Create a cluster in a hosted Kubernetes provider

Provision new nodes and create a cluster using RKE2/K3s

Use existing nodes and create a cluster using RKE2/K3s

You don’t need to do much. Just name it “cluster” and press create.



Cluster Name * cluster

Cluster Appearance CSR Customize

Cluster Description Any text you want that better describes this cluster

On Kubernetes 1.30 or greater, the Azure Cloud Provider has been removed. See the documentation for more information.

Kubernetes Version v1.31.5+rke2r1

Cloud Provider Default - RKE2 Embedded

□ Show deprecated Kubernetes patch versions

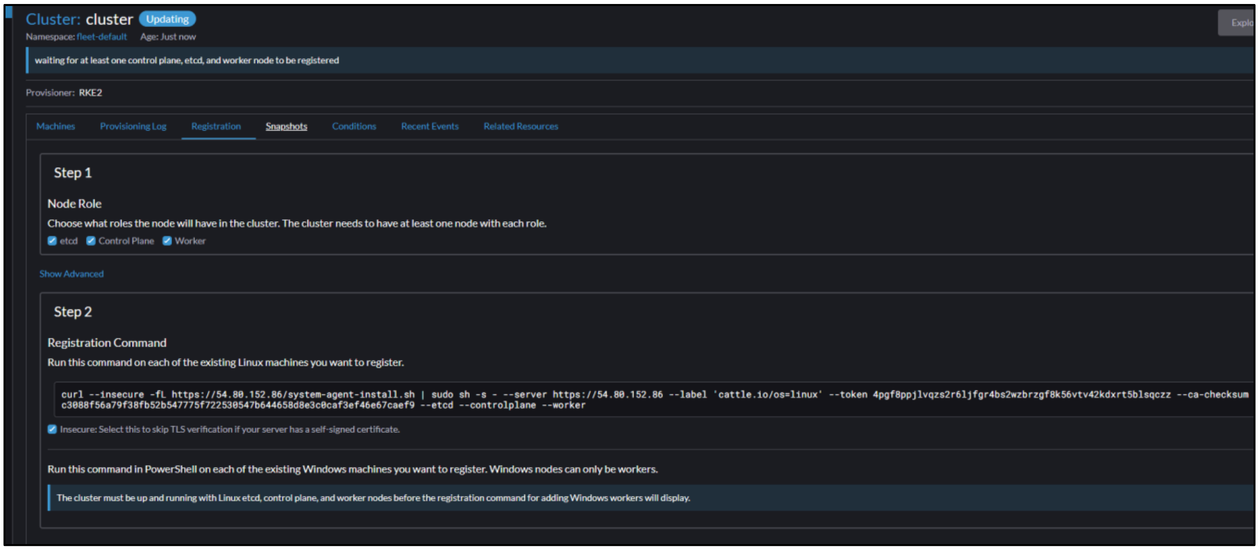
Container Network calico

CIS Profile (None)

Pod Security Admission Configuration Template Default - RKE2 Embedded

□ Project Network Isolation

☑ CoreDNS ☑ NGINX Ingress ☑ Metrics Server



Cluster: cluster Updating

Namespace: fleet-default Age: Just now

waiting for at least one control plane, etcd, and worker node to be registered

Provisioner: RKE2

Step 1

Node Role

Choose what roles the node will have in the cluster. The cluster needs to have at least one node with each role.

☑ etcd ☑ Control Plane ☑ Worker

Step 2

Registration Command

Run this command on each of the existing Linux machines you want to register.

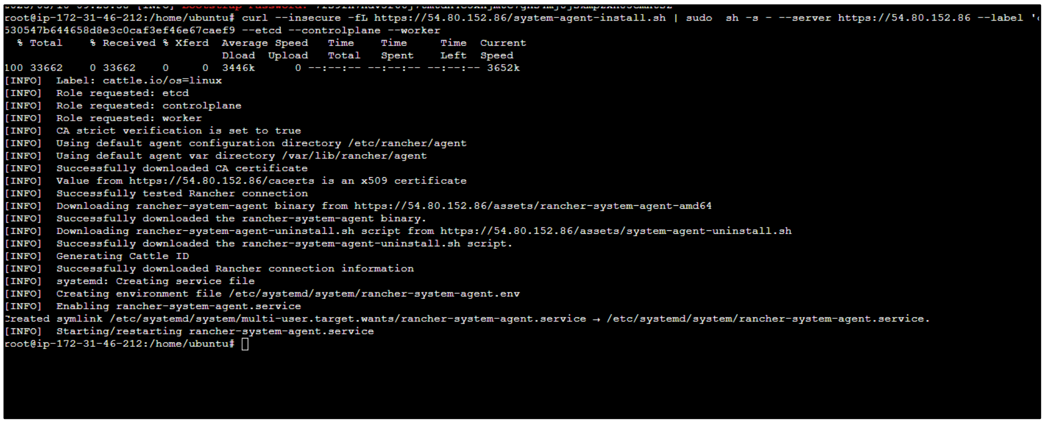
```
curl --insecure -fL https://54.80.152.86/system-agent-install.sh | sudo sh -s - --server https://54.80.152.86 --label 'cattle.io/os=linux' --token 4pgf8ppjlvqzs2r6ljfgr4bs2wzbrzgf8k56vtv42kdxrt5blsqczz --ca-checksum
c3088f56a79f38fb52b547775f722530547b644658d8e3c0caf3ef46e67caef9 --etcd --controlplane --worker
```

☑ Insecure: Select this to skip TLS verification if your server has a self-signed certificate.

Run this command in PowerShell on each of the existing Windows machines you want to register. Windows nodes can only be workers.

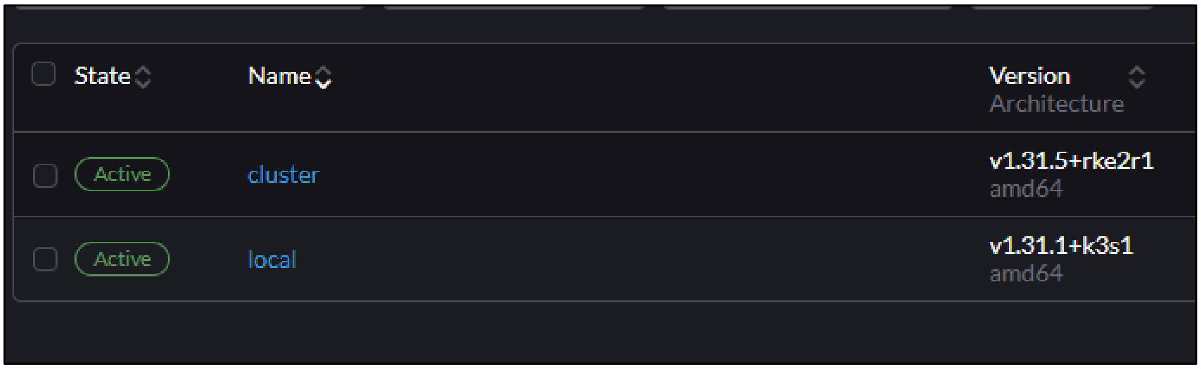
The cluster must be up and running with Linux etcd, control plane, and worker nodes before the registration command for adding Windows workers will display.

Then, you need to register a control plane, etcd, and worker node. You can do these on separate machines, but let’s just do it here on the Rancher instance since this is a large machine and can handle it. Please select the “insecure” option of the registration command or this won’t work. Copy it and run it on your Rancher instance.



```
root@ip-172-31-46-212:/home/ubuntu# curl --insecure -fL https://54.80.152.86/system-agent-install.sh | sudo  sh -s - --server https://54.80.152.86 --label '
530547b644658d8e3c0caf3ef46e67caef9 --etcd --controlplane --worker
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 33662    0 33662    0     0  3446k      0 --:--:-- --:--:-- --:--:-- 3652k
[INFO]  Label: cattle.io/os=linux
[INFO]  Role requested: etcd
[INFO]  Role requested: controlplane
[INFO]  Role requested: worker
[INFO]  CA strict verification is set to true
[INFO]  Using default agent configuration directory /etc/rancher/agent
[INFO]  Using default agent var directory /var/lib/rancher/agent
[INFO]  Successfully downloaded CA certificate
[INFO]  Value from https://54.80.152.86/cacerts is an x509 certificate
[INFO]  Successfully tested Rancher connection
[INFO]  Downloading rancher-system-agent binary from https://54.80.152.86/assets/rancher-system-agent-amd64
[INFO]  Successfully downloaded the rancher-system-agent binary.
[INFO]  Downloading rancher-system-agent-uninstall.sh script from https://54.80.152.86/assets/system-agent-uninstall.sh
[INFO]  Successfully downloaded the rancher-system-agent-uninstall.sh script.
[INFO]  Generating Cattle ID
[INFO]  Successfully downloaded Rancher connection information
[INFO]  systemd: Creating service file
[INFO]  Creating environment file /etc/systemd/system/rancher-system-agent.env
[INFO]  Enabling rancher-system-agent.service
Created symlink /etc/systemd/system/multi-user.target.wants/rancher-system-agent.service → /etc/systemd/system/rancher-system-agent.service.
[INFO]  Starting/restarting rancher-system-agent.service
root@ip-172-31-46-212:/home/ubuntu# 
```

Wait about 5 minutes at this point for everything to be set up properly. During this time, Rancher will show that the cluster is updating. Once 5 minutes has elapsed, you should now see that your new Kubernetes cluster is active.

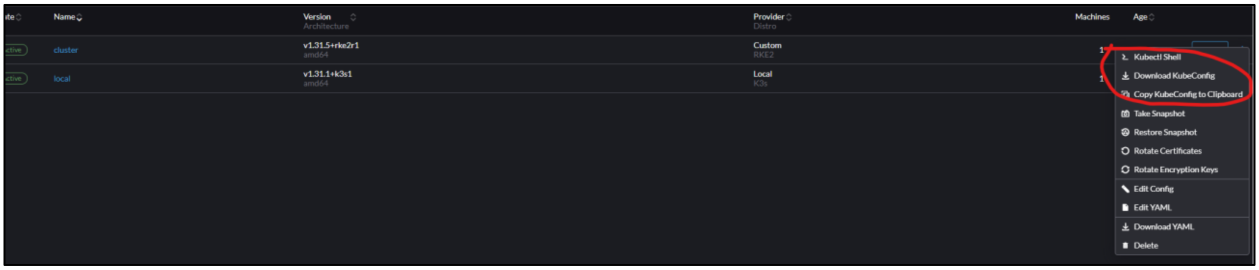


□	State	Name	Version Architecture
□	Active	cluster	v1.31.5+rke2r1 amd64
□	Active	local	v1.31.1+k3s1 amd64

We have our K8s cluster created, named “cluster”, and it has 1 worker node, 1 control plane, and 1 etcd node, all on the same machine as where our cluster is.

The last step that you should do is download the kubeconfig file to your computer, as we will need it here shortly for the Jenkins instance:

This can be done from the “Cluster Management” tab on the left.



ate	Name	Version Architecture	Provider Distro	Machines	Age
ctive	cluster	v1.31.5+rke2r1 amd64	Custom RKE2	1	
ctive	local	v1.31.1+k3s1 amd64	Local K3s	1	

This allows us to connect to our cluster from different machines and instances (like our Jenkins instance that we will make after this) using the kubeconfig file.

Congratulations, at this point, you are done with everything regarding the Rancher instance. Rancher has been setup, and you can access it using the elastic IP we assigned to the instance at http://{ec2-public-ip}:80

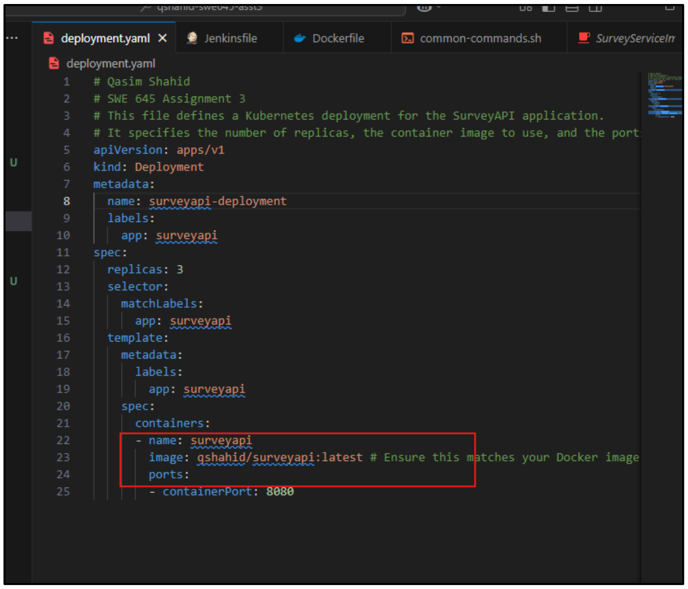
GitHub Repository Setup

GitHub Repository: https://github.com/qasimshahid/qshahid-swe645-asst3/

Please fork this repository or copy all the files from it and add them to a new repository that you created. This has pretty much everything you need, but you do need to make some changes.

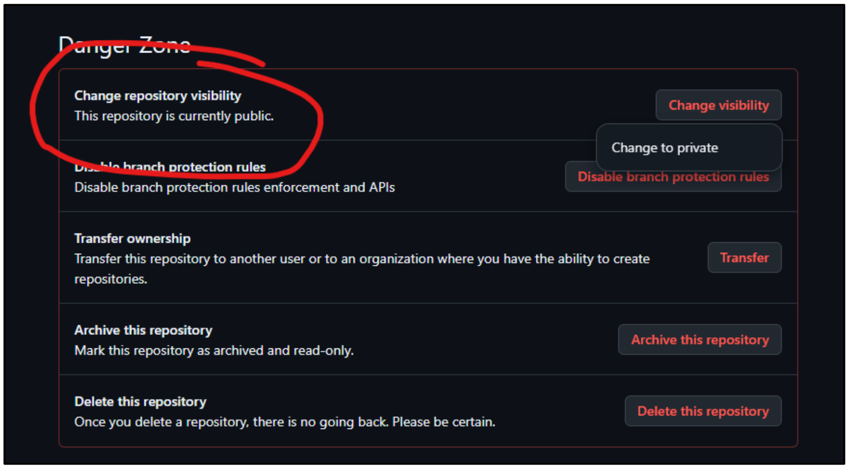
First of all, please fix these settings to be your own. They should utilize the link of your GitHub repository and whatever your username is for Docker Hub and what your image should be called.

For example, here you can see I am using my Docker Hub username. Replace this with your own. Please look through all the files of the GitHub repository and replace it wherever applicable. You need to make sure it is all configured correctly for our pipeline to work without access issues.



```yaml
# Qasim Shahid
# SWE 645 Assignment 3
# This file defines a Kubernetes deployment for the SurveyAPI application.
# It specifies the number of replicas, the container image to use, and the port
apiVersion: apps/v1
kind: Deployment
metadata:
  name: surveyapi-deployment
  labels:
    app: surveyapi
spec:
  replicas: 3
  selector:
    matchLabels:
      app: surveyapi
  template:
    metadata:
      labels:
        app: surveyapi
    spec:
      containers:
      - name: surveyapi
        image: qshahid/surveyapi:latest # Ensure this matches your Docker image
        ports:
        - containerPort: 8080
```

After you do all that, make sure the GitHub repository is public by going into the repository settings:



Danger Zone

Change repository visibility
This repository is currently public.

Disable branch protection rules
Disable branch protection rules enforcement and APIs

Transfer ownership
Transfer this repository to another user or to an organization where you have the ability to create repositories.

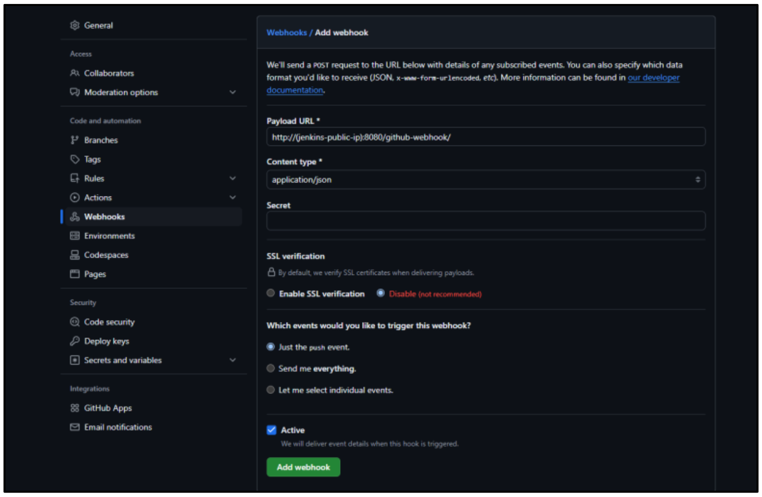
Archive this repository
Mark this repository as archived and read-only.

Delete this repository
Once you delete a repository, there is no going back. Please be certain.

Please also at this point go into the webhook settings and create a new webhook that points to the following link:

http://{jenkins-public-ip}:8080/github-webhook/

Replace {jenkins-public-ip} with the public elastic IP of your Jenkins instance and use the following settings:



Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in our developer documentation.

Payload URL *
http://{jenkins-public-ip}:8080/github-webhook/

Content type *
application/json

Secret

SSL verification
By default, we verify SSL certificates when delivering payloads.
○ Enable SSL verification ● Disable (not recommended)

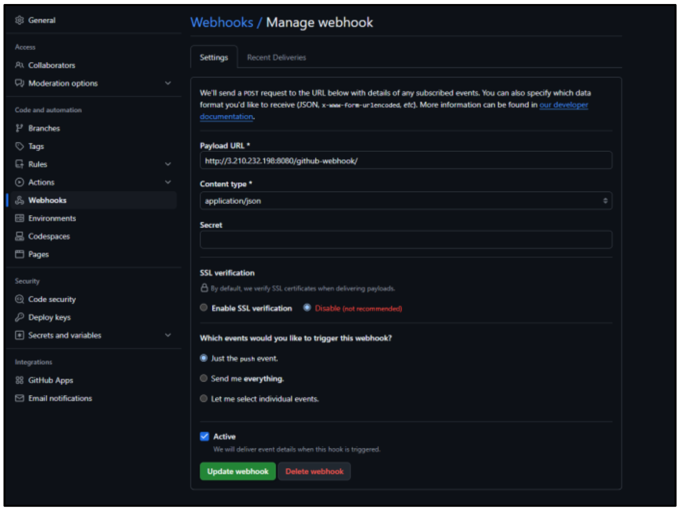
Which events would you like to trigger this webhook?
● Just the push event.
○ Send me everything.
○ Let me select individual events.

☑ Active
We will deliver event details when this hook is triggered.

Add webhook

Add this webhook.

Example:



Webhooks / Manage webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in our developer documentation.

Payload URL *
http://3.210.232.198:8080/github-webhook/

Content type *
application/json

Secret

SSL verification
By default, we verify SSL certificates when delivering payloads.
○ Enable SSL verification ● Disable (not recommended)

Which events would you like to trigger this webhook?
● Just the push event.
○ Send me everything.
○ Let me select individual events.

☑ Active
We will deliver event details when this hook is triggered.

Update webhook Delete webhook

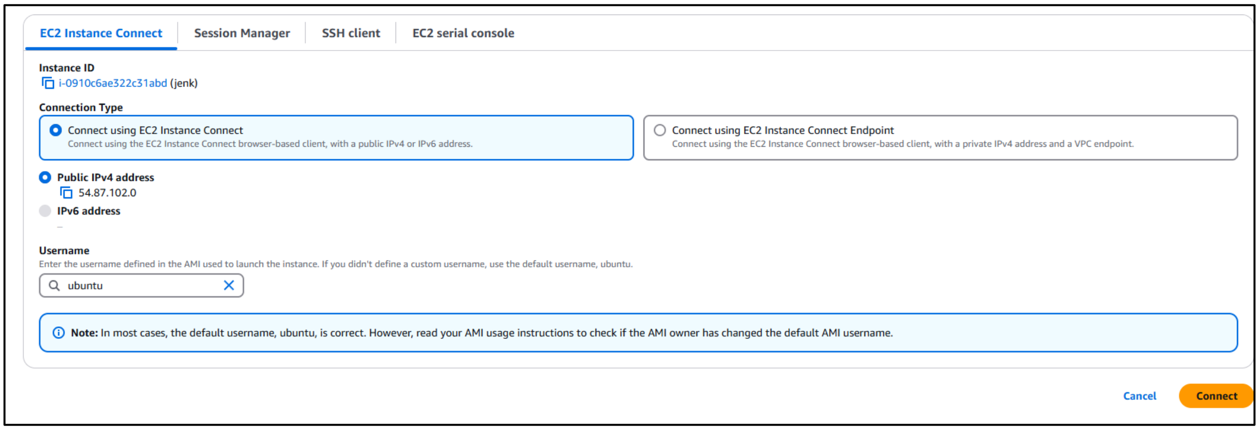
This webhook will allow us to automatically build and deploy our application whenever we push to main branch on our GitHub repository. GitHub will recognize our push event and send a POST request to our Jenkins server, which will then kick off the pipeline.

Congratulations, at this point, you are done with setting up your GitHub repo. Push all the code changes and modifications you made and save all of these setting.

Jenkins Instance Setup

So, this is the other EC2 machine we created at the start, and this is where we will host our Jenkins server. This will allow us to automate our builds and make a pipeline that gets triggered whenever we push to the main branch of our GitHub repository.

Please connect to the instance using Instance Connect:



Instance ID
i-0910c6ae322c31abd (jenk)

Connection Type

● Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.

○ Connect using EC2 Instance Connect Endpoint
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

● **Public IPv4 address**
54.87.102.0

○ **IPv6 address**
–

Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ubuntu.

ubuntu

Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

After that, please execute these commands if you haven't already. These commands are required for both instances:

```
sudo su
sudo apt-get update -y
sudo apt-get upgrade -y

curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

sudo snap install kubectl --classic
```

After that, please execute the following commands or follow the instructions from the official Jenkins site on how to download for Ubuntu (https://www.jenkins.io/doc/book/installing/linux/):

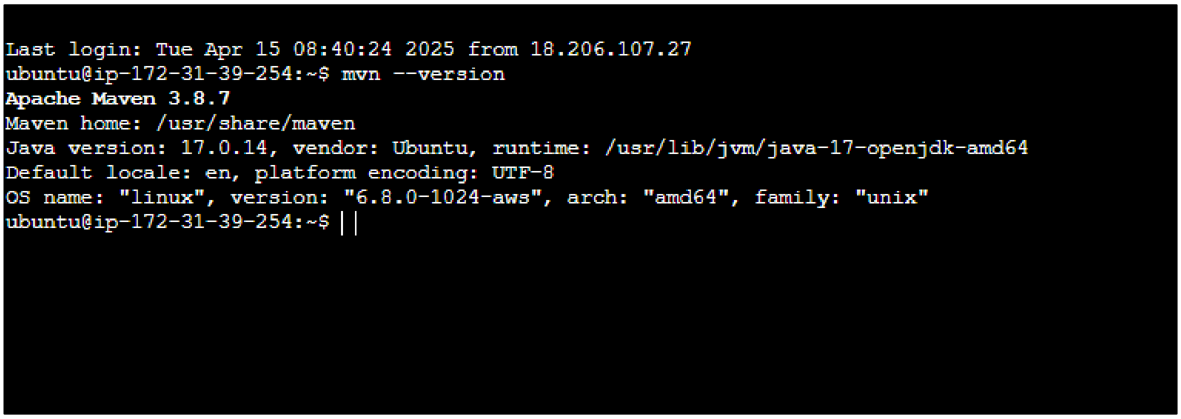
```
sudo apt install fontconfig openjdk-17-jre -y

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins -y
```

We also need to install Maven so we can build our Spring application.

```
sudo apt install maven
sudo apt install openjdk-17-jdk
```

IMPORTANT: Installing Maven might install a different version of Java other than 17. We need to make sure that our machine will be using JDK 17 and not any other JDK version. Please check the java version using:

```
mvn -version
```



```
Last login: Tue Apr 15 08:40:24 2025 from 18.206.107.27
ubuntu@ip-172-31-39-254:~$ mvn --version
Apache Maven 3.8.7
Maven home: /usr/share/maven
Java version: 17.0.14, vendor: Ubuntu, runtime: /usr/lib/jvm/java-17-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.8.0-1024-aws", arch: "amd64", family: "unix"
ubuntu@ip-172-31-39-254:~$ |
```

Make sure it says version 17 or else our pipeline will NOT work.

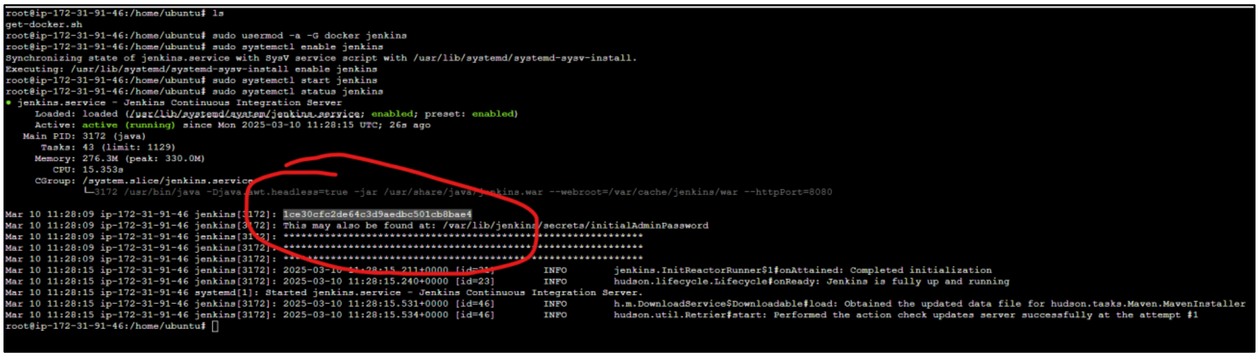
If it doesn't say version 17, use these commands to find the path to JDK 17 which we installed earlier and point your Java to that directory.

```
# IMPORTANT - MAKE SURE JENKINS INSTANCE HAS JAVA 17 SET AS THE DEFAULT JAVA
# VERSION
# You can check the default java version using the following command:
update-java-alternatives --list

# If you have multiple versions of Java installed, you can set the default version using the
following command:
# This will set Java 17 as the default version. Make sure to replace the path with the correct
# one for your system. This is what the path should be if you installed openjdk-17-jdk
sudo update-java-alternatives --set /usr/lib/jvm/java-1.17.0-openjdk-amd64
```

(This command is also SUPER IMPORTANT, make sure to execute this one or else builds might fail. If it does not work after your first build and you get some docker permission issues, execute it again)

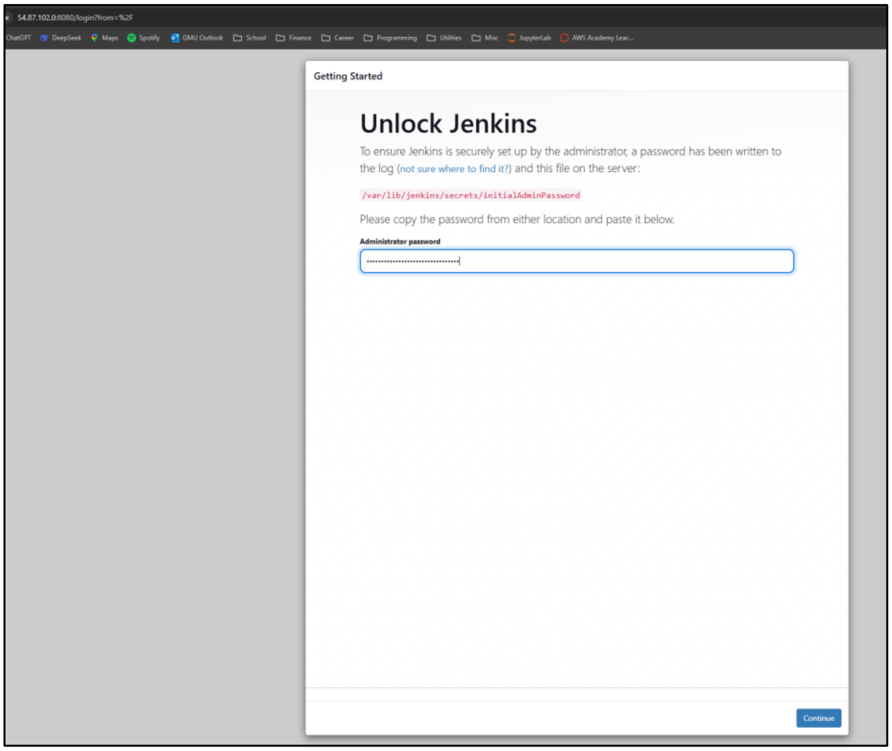
```
sudo usermod -a -G docker jenkins

sudo systemctl enable jenkins
sudo systemctl start jenkins
sudo systemctl status jenkins
```

These commands will install Java and Jenkins. They will also make Docker accessible to Jenkins without needing to do sudo. The last command will also print a password, please store this as we will need it here in a second:



```
root@ip-172-31-91-46:/home/ubuntu# ls
get-docker.sh
root@ip-172-31-91-46:/home/ubuntu# sudo usermod -a -G docker jenkins
root@ip-172-31-91-46:/home/ubuntu# sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
root@ip-172-31-91-46:/home/ubuntu# sudo systemctl start jenkins
root@ip-172-31-91-46:/home/ubuntu# sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
     Active: active (running) since Mon 2025-03-10 11:28:15 UTC; 26s ago
   Main PID: 3172 (java)
      Tasks: 43 (limit: 1129)
     Memory: 276.3M (peak: 330.0M)
        CPU: 15.353s
     CGroup: /system.slice/jenkins.service
             └─3172 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Mar 10 11:28:09 ip-172-31-91-46 jenkins[3172]: 1ce30cfc2de64c3d9aedbc501cb8bae4
Mar 10 11:28:09 ip-172-31-91-46 jenkins[3172]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Mar 10 11:28:09 ip-172-31-91-46 jenkins[3172]: *************************************************************
Mar 10 11:28:09 ip-172-31-91-46 jenkins[3172]: *************************************************************
Mar 10 11:28:09 ip-172-31-91-46 jenkins[3172]: *************************************************************
Mar 10 11:28:15 ip-172-31-91-46 jenkins[3172]: 2025-03-10 11:28:15.211+0000 [id=31]        INFO        jenkins.InitReactorRunner$1#onAttained: Completed initialization
Mar 10 11:28:15 ip-172-31-91-46 jenkins[3172]: 2025-03-10 11:28:15.240+0000 [id=23]        INFO        hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Mar 10 11:28:15 ip-172-31-91-46 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Mar 10 11:28:15 ip-172-31-91-46 jenkins[3172]: 2025-03-10 11:28:15.531+0000 [id=46]        INFO        h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
Mar 10 11:28:15 ip-172-31-91-46 jenkins[3172]: 2025-03-10 11:28:15.534+0000 [id=46]        INFO        hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt #1
root@ip-172-31-91-46:/home/ubuntu# 
```

Now, the server is up. To go to it, use the elastic public IP we created at the start and go to this link:
http://{jenkins-public-ip}:8080
For example:
http://54.87.102.0:8080/



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:

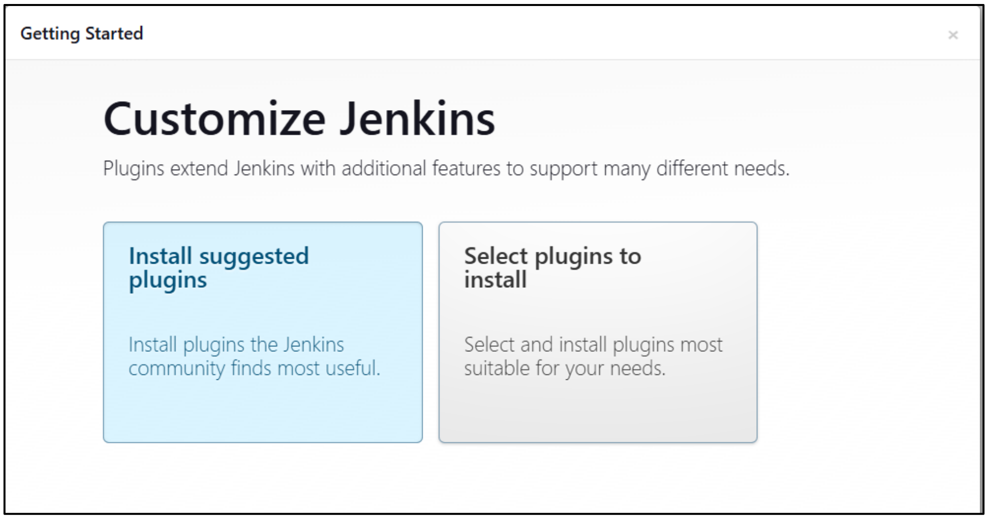
/var/lib/jenkins/secrets/initialAdminPassword

Please copy the password from either location and paste it below.

Administrator password
································

Go the link and Jenkins will ask you for the password from earlier. Paste it in.

Jenkins will now ask you what plugins you want. Select “Install suggested plugins.” We will add some more later on.



Getting Started

Customize Jenkins

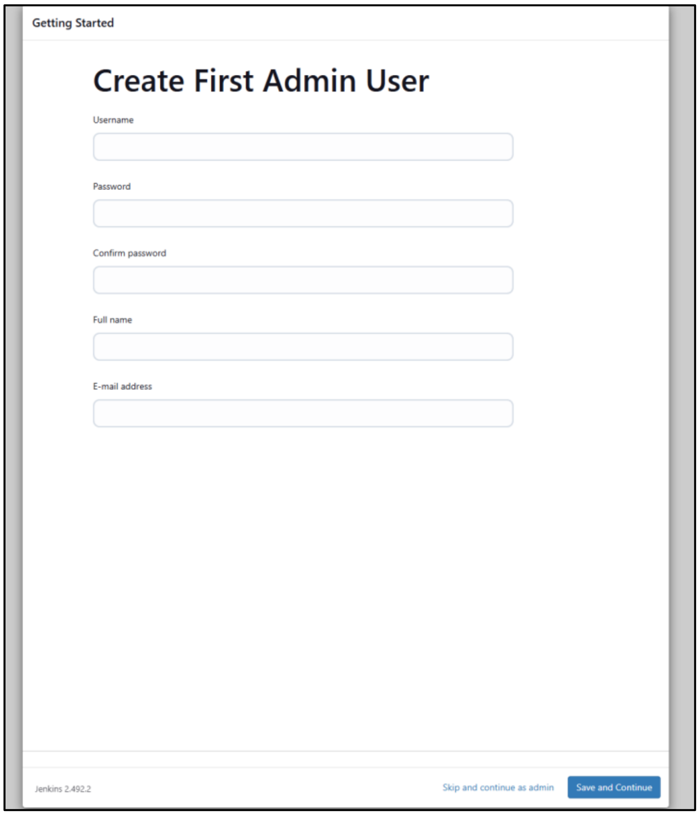
Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.



Getting Started

Create First Admin User

Username

Password

Confirm password

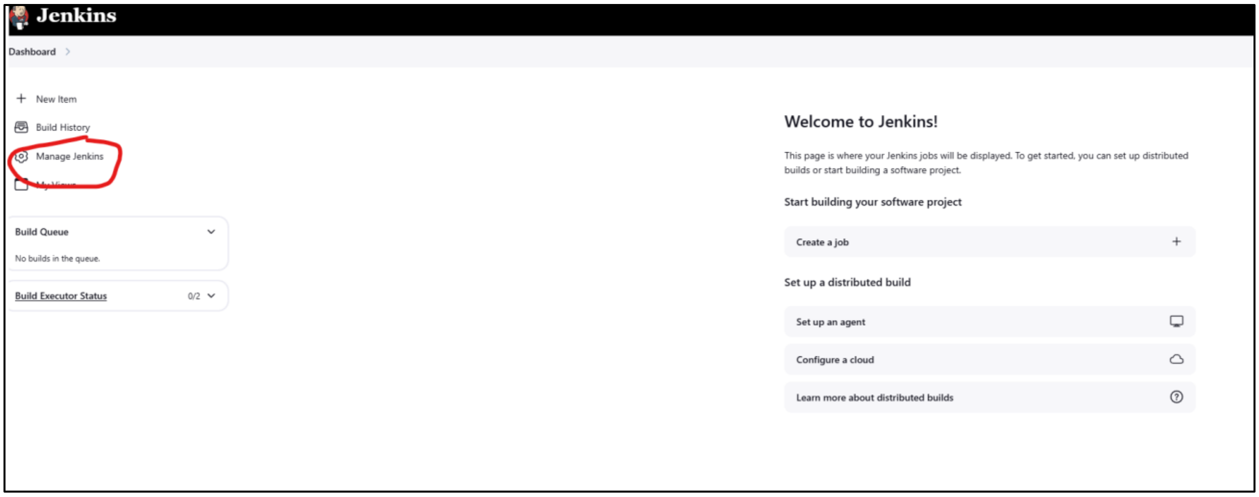
Full name

E-mail address

Jenkins 2.492.2

Then, create a username and password. I use “admin” and “temppass3$” with my GMU email. After this, keep clicking continue until the setup is complete.

We are now done with the initial setup. You should be on this screen:



Welcome to Jenkins!

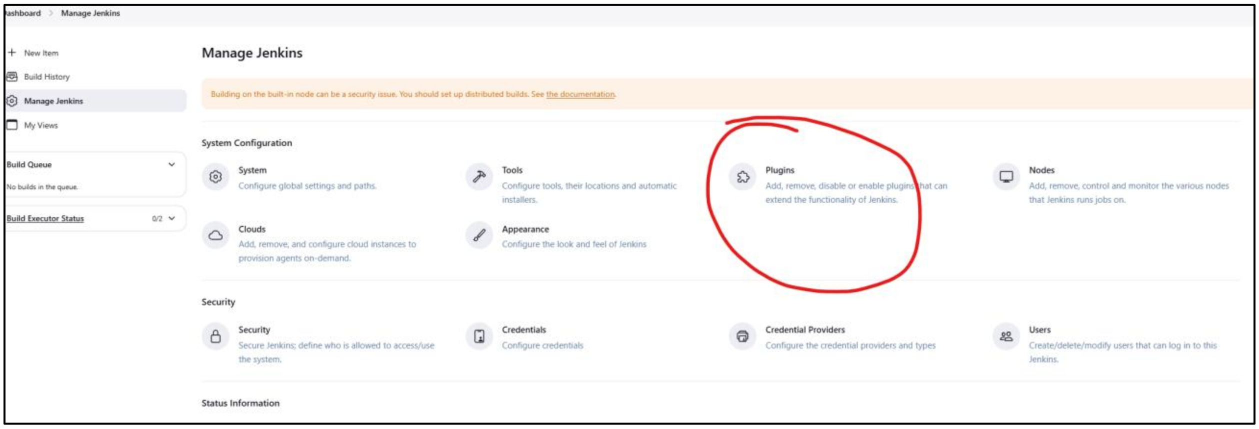
This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Set up a distributed build

No builds in the queue.

Now, go to “Manage Jenkins” so that we can install some plugins.



Manage Jenkins

No builds in the queue.

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

System Configuration

System
Configure global settings and paths.

Tools
Configure tools, their locations and automatic installers.

Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Clouds
Add, remove, and configure cloud instances to provision agents on-demand.

Appearance
Configure the look and feel of Jenkins

Security

Security
Secure Jenkins; define who is allowed to access/use the system.

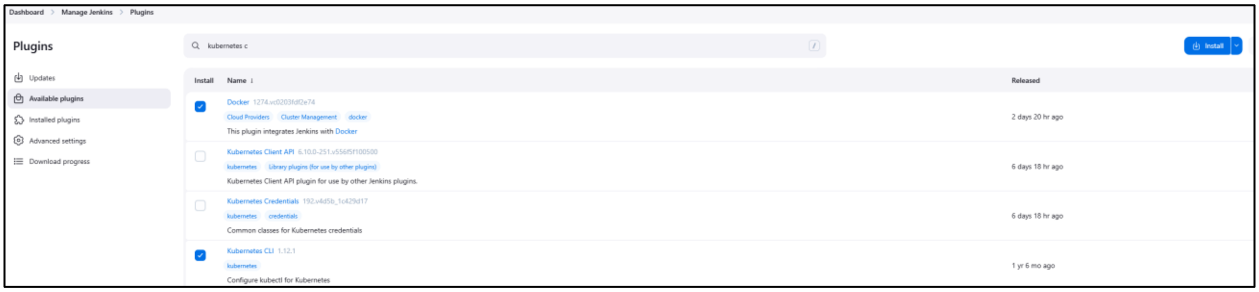
Credentials
Configure credentials

Credential Providers
Configure the credential providers and types

Users
Create/delete/modify users that can log in to this Jenkins.

Status Information

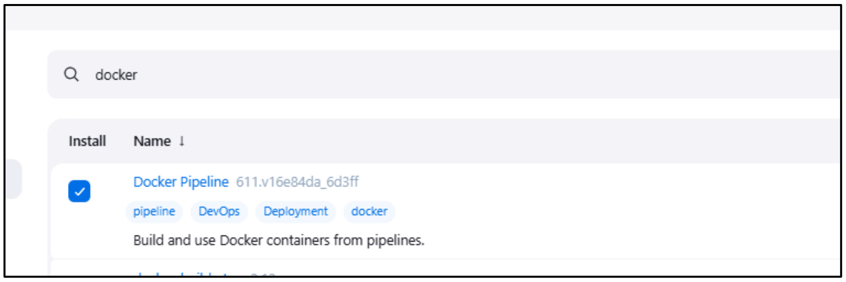
We need to install three plugins:
Docker plugin
Kubernetes CLI plugin
Docker Pipeline plugin



Plugins

kubernetes c

Install	Name ↓	Released
☑	Docker 1274.vc0203fdf2e74 Cloud Providers Cluster Management docker This plugin integrates Jenkins with Docker	2 days 20 hr ago
□	Kubernetes Client API 6.10.0-251.v556f5f100500 kubernetes Library plugins (for use by other plugins) Kubernetes Client API plugin for use by other Jenkins plugins.	6 days 18 hr ago
□	Kubernetes Credentials 192.v4d5b_1c429d17 kubernetes credentials Common classes for Kubernetes credentials	6 days 18 hr ago
☑	Kubernetes CLI 1.12.1 kubernetes Configure kubectl for Kubernetes	1 yr 6 mo ago

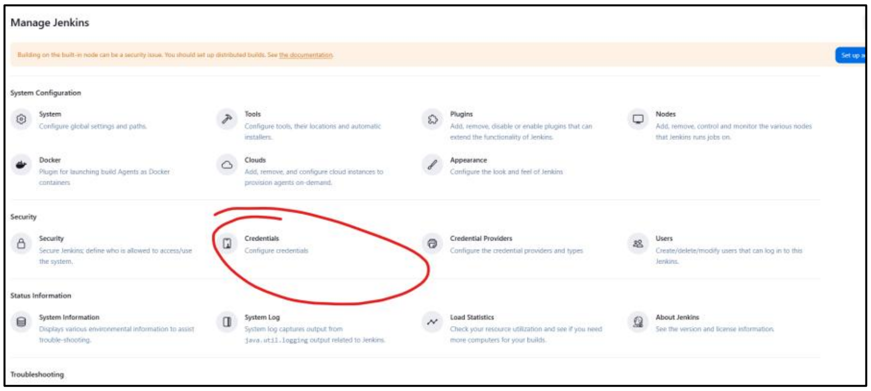


☑ Docker Pipeline 611.v16e84da_6d3ff

Build and use Docker containers from pipelines.

Install these and wait for them to finish downloading. We need these to build our image and deploy to our cluster.

After these are complete, go back to “Manage Jenkins” but this time, go to Credentials:


Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

System Configuration

System – Configure global settings and paths.

Tools – Configure tools, their locations and automatic installers.

Plugins – Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Nodes – Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Docker – Plugin for launching build Agents as Docker containers

Clouds – Add, remove, and configure cloud instances to provision agents on-demand.

Appearance – Configure the look and feel of Jenkins

Security

Security – Secure Jenkins; define who is allowed to access/use the system.

Credentials – Configure credentials

Credential Providers – Configure the credential providers and types

Users – Create/delete/modify users that can log in to this Jenkins.

Status Information

System Information – Displays various environmental information to assist trouble-shooting.

System Log – System log captures output from java.util.logging output related to Jenkins.

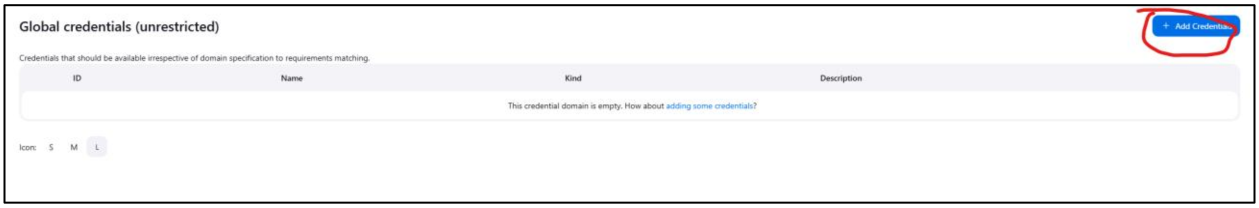
Load Statistics – Check your resource utilization and see if you need more computers for your builds.

About Jenkins – See the version and license information.

Troubleshooting

We will now add our Docker Hub credentials and the kubeconfig file that you should have saved from the Rancher Instance setup.
First, the kubeconfig file:

Add a new credential to System -> Global credentials.

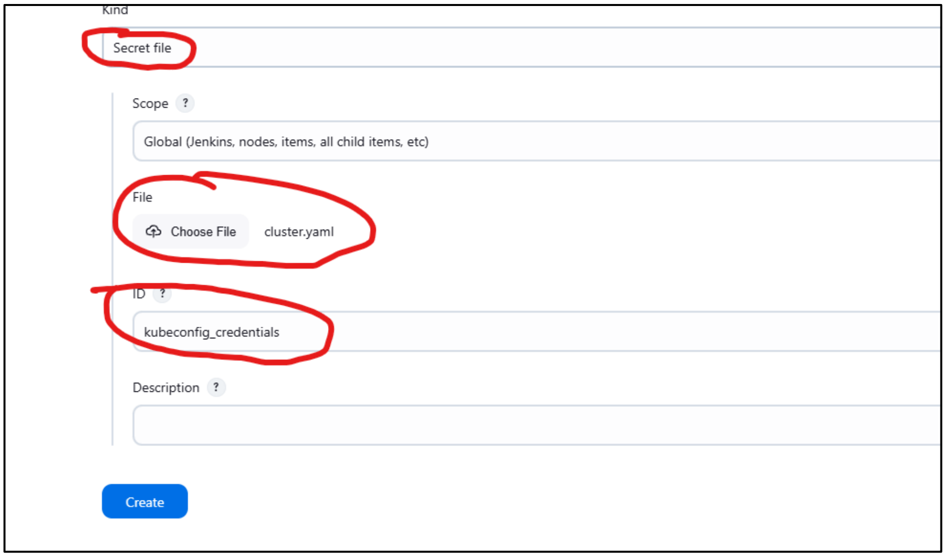

Global credentials (unrestricted)

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

ID	Name	Kind	Description
This credential domain is empty. How about adding some credentials?			

Make it a “Secret file”

Copy these settings:
ID - kubeconfig_credentials
File – Select the .yaml file we downloaded earlier when setting up the Rancher Instance. This was the KubeConfig file of our cluster that we downloaded from Rancher.



Kind

Secret file

Scope ?

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

File

Choose File cluster.yaml

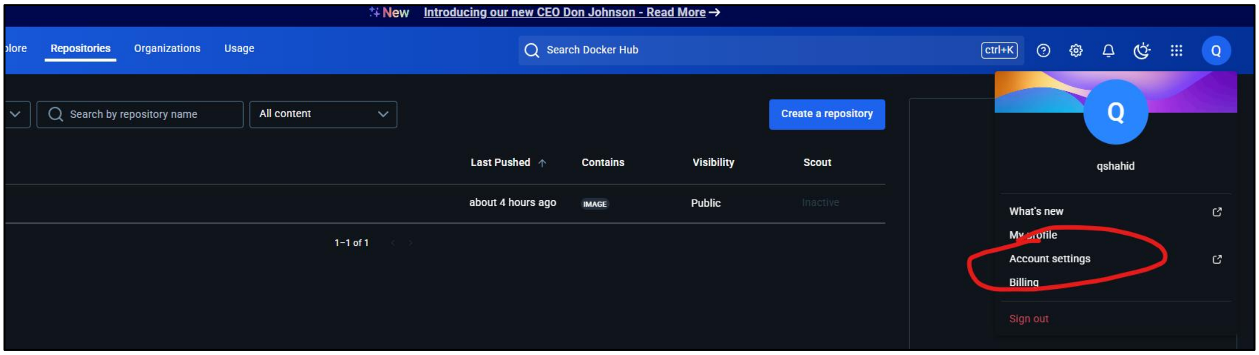
ID ?

kubeconfig_credentials

Description ?

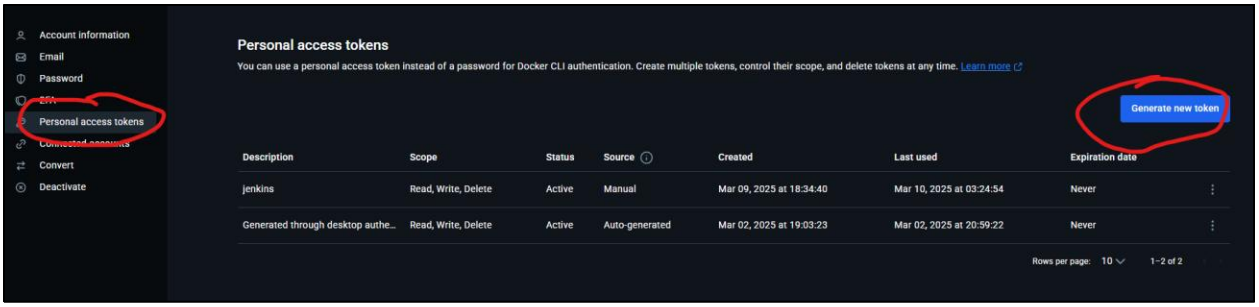
This will allow us to deploy to our cluster even though we are not on the same instance. Jenkins can use Kubernetes CLI with this file to connect to our cluster on the other EC2 instance.
Second, lets add our Docker Hub credentials:

First, you need to go to Docker Hub and login to your account. Go to “Account settings.”



Last Pushed ↑	Contains	Visibility	Scout
about 4 hours ago	IMAGE	Public	Inactive

Go to “Personal access tokens” and create a new token.



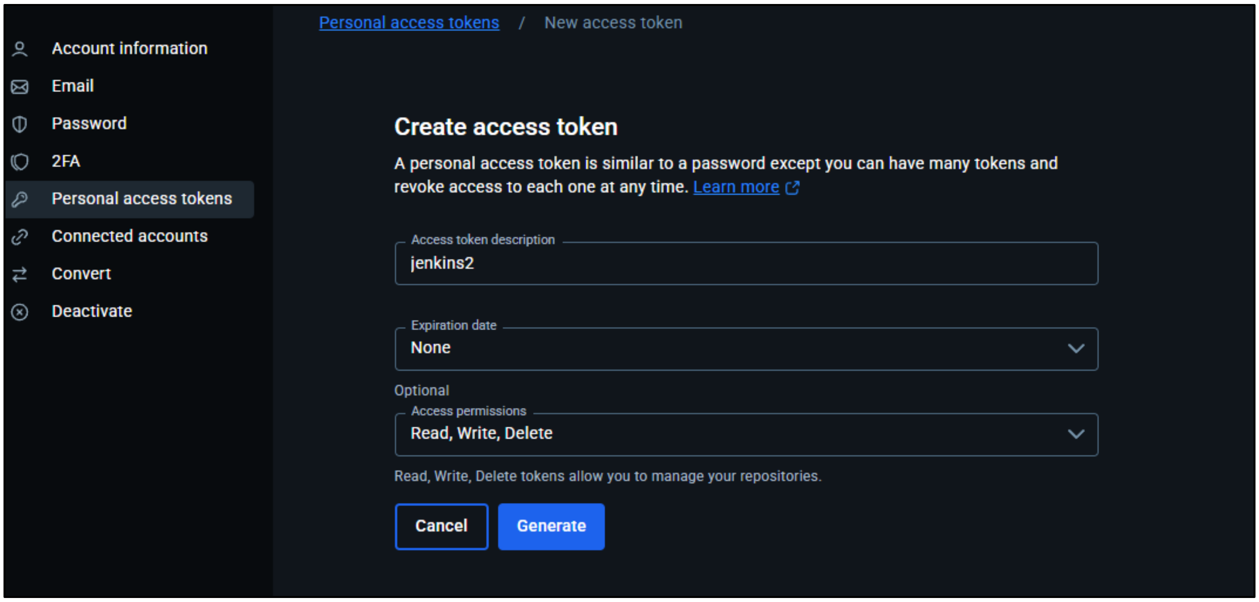
Personal access tokens

You can use a personal access token instead of a password for Docker CLI authentication. Create multiple tokens, control their scope, and delete tokens at any time. Learn more

Description	Scope	Status	Source	Created	Last used	Expiration date
jenkins	Read, Write, Delete	Active	Manual	Mar 09, 2025 at 18:34:40	Mar 10, 2025 at 03:24:54	Never
Generated through desktop authe...	Read, Write, Delete	Active	Auto-generated	Mar 02, 2025 at 19:03:23	Mar 02, 2025 at 20:59:22	Never

Rows per page: 10 1–2 of 2

Copy these settings:
Access permissions – Read, Write, Delete



Create access token

A personal access token is similar to a password except you can have many tokens and revoke access to each one at any time. Learn more

Access token description: jenkins2

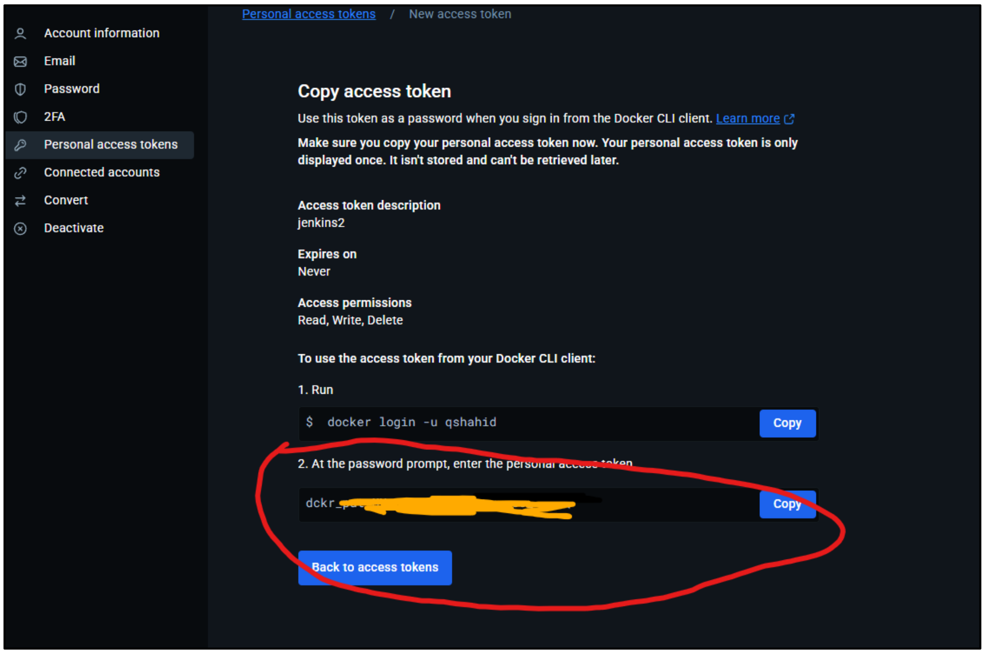
Expiration date: None

Optional

Access permissions: Read, Write, Delete

Read, Write, Delete tokens allow you to manage your repositories.

Click generate. After that, it will give you a password. Copy this and save it somewhere as we will need it here soon. This will only show once, so be careful and make sure to save it somewhere.



Copy access token

Use this token as a password when you sign in from the Docker CLI client. Learn more

Make sure you copy your personal access token now. Your personal access token is only displayed once. It isn't stored and can't be retrieved later.

Access token description
jenkins2

Expires on
Never

Access permissions
Read, Write, Delete

To use the access token from your Docker CLI client:

1. Run

```
$ docker login -u qshahid
```

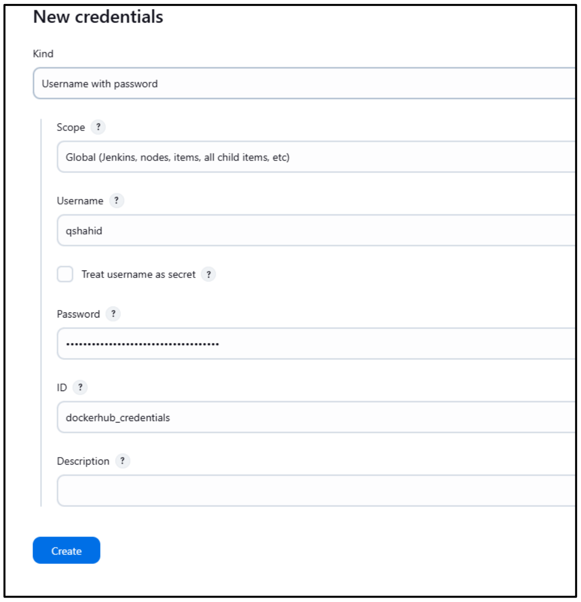
2. At the password prompt, enter the personal access token.

```
dckr_[redacted]
```

Now, go back to Jenkins and create another credential:

Make it as a “Username with password”. Then, add your Docker Hub username and take the password you just copied from the Personal Access Token and paste it into the password field. Then, give it this id:
dockerhub_credentials

Save this credential. Here is an example:



New credentials

Kind

Username with password

Scope ?

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

Username ?

qshahid

□ Treat username as secret ?

Password ?

··

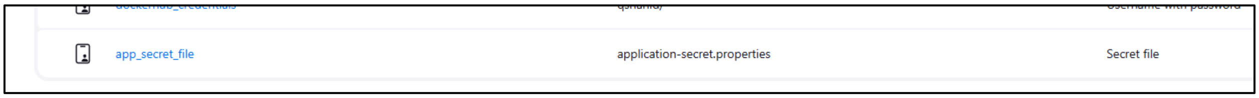
ID ?

dockerhub_credentials

Description ?

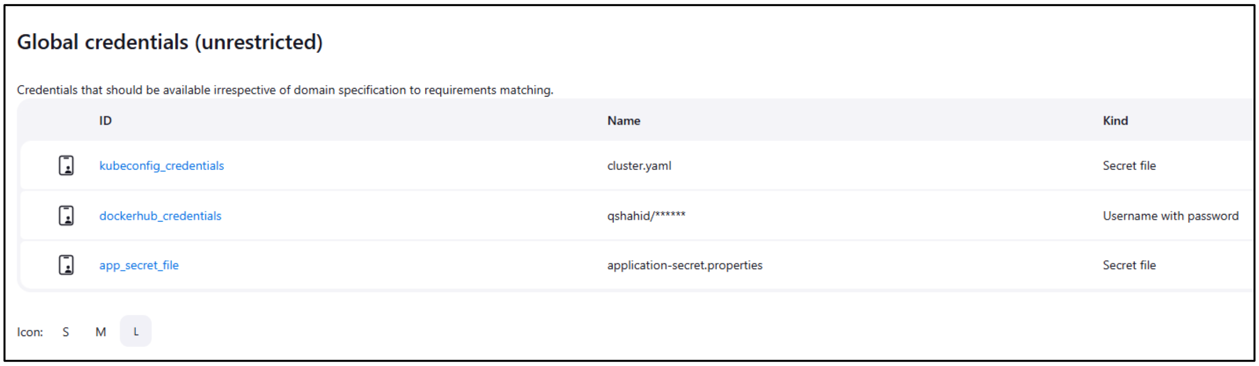
Now, we also need to add our application-secret.properties file to Jenkins. Follow the same steps you did for the kubeconfig file.

Select your application-secret.properties as the file (this has your database connection details) and use the id "app_secret_file".



app_secret_file	application-secret.properties	Secret file

Overall, you should have this setup in your credentials in Jenkins:

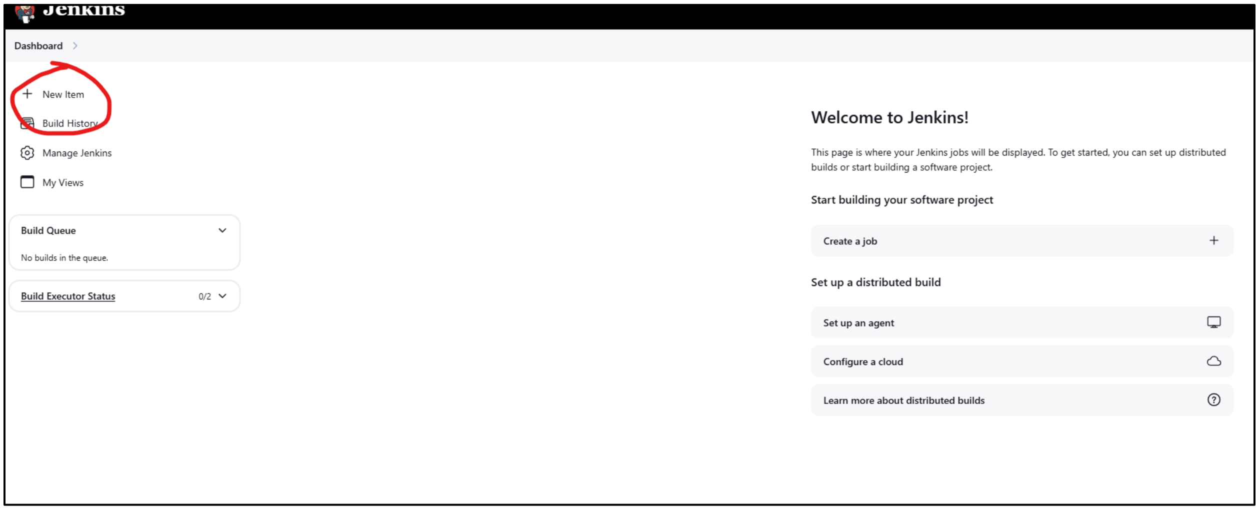


Global credentials (unrestricted)

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

ID	Name	Kind
kubeconfig_credentials	cluster.yaml	Secret file
dockerhub_credentials	qshahid/******	Username with password
app_secret_file	application-secret.properties	Secret file

Finally, we are ready to create our pipeline. Go back to the Jenkins dashboard and create a new item:



No builds in the queue.

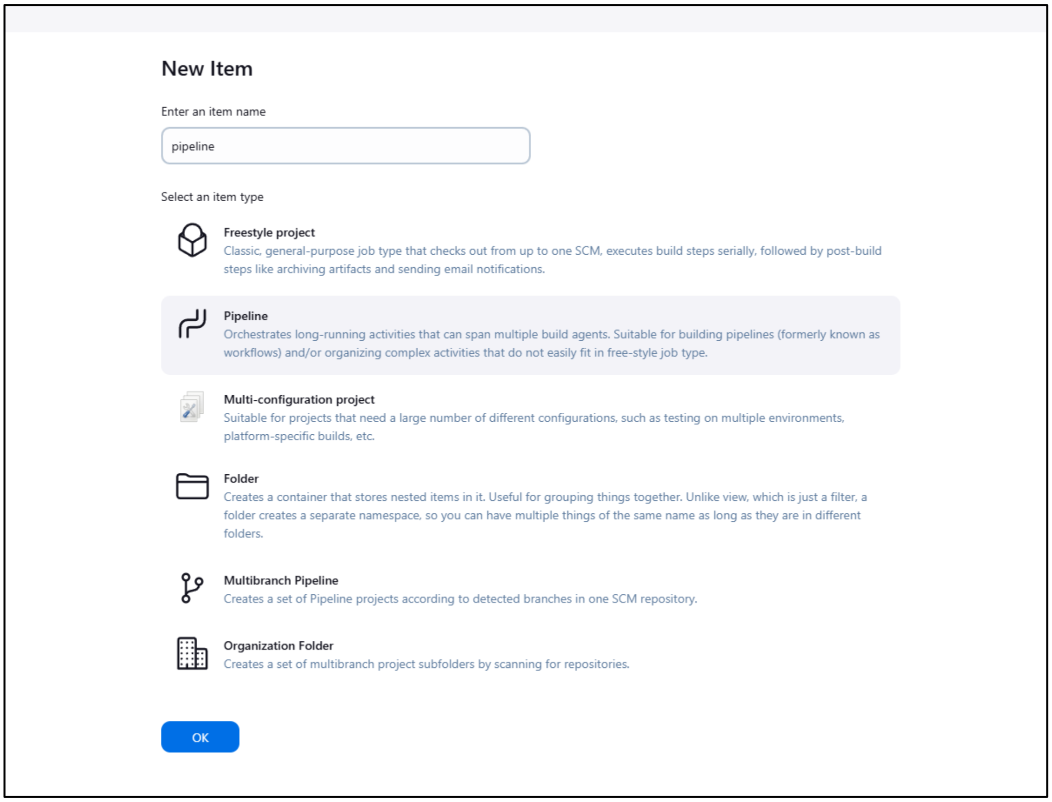
Welcome to Jenkins!

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.

Start building your software project

Set up a distributed build

Create a new pipeline.



New Item

Enter an item name: pipeline

Select an item type

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.

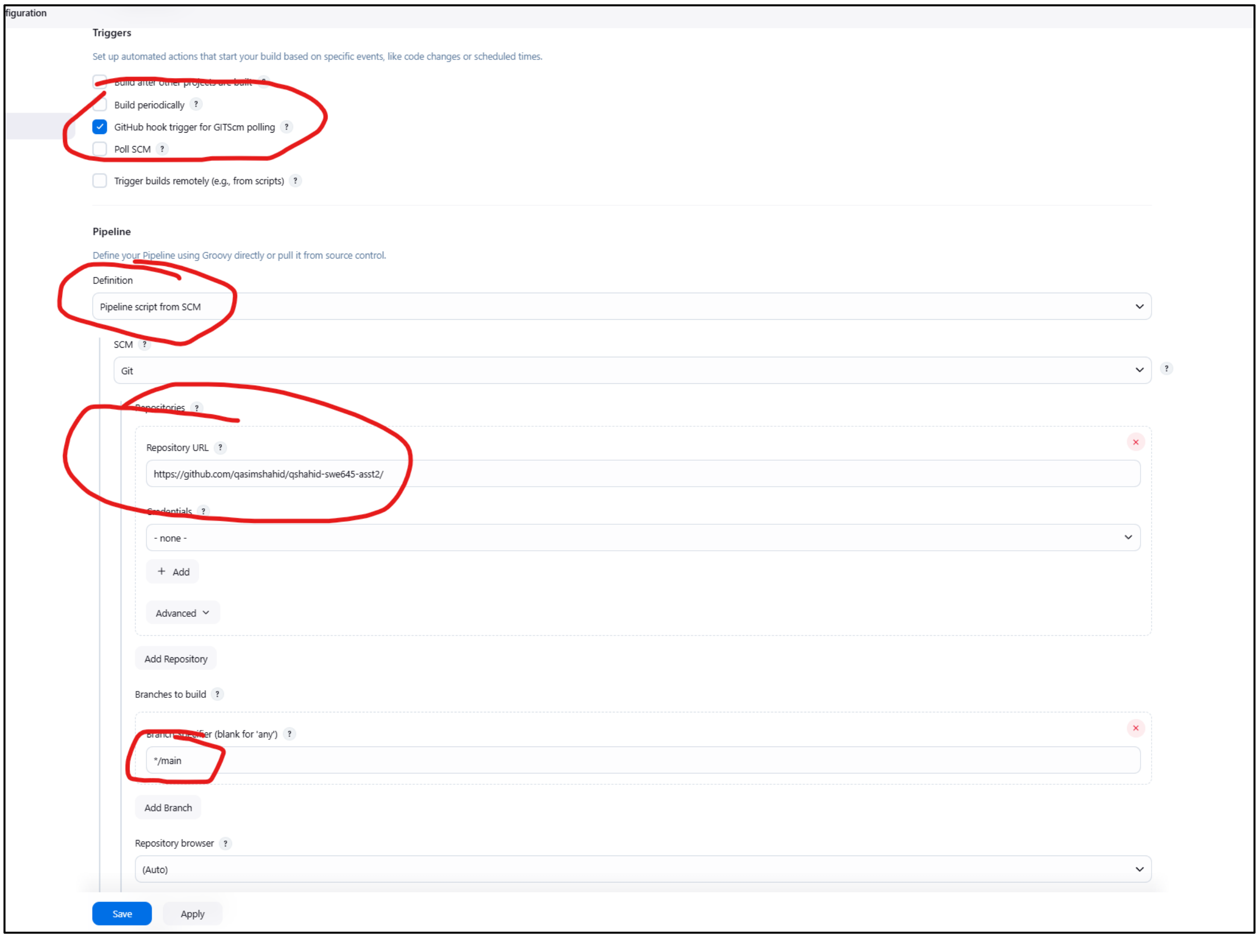
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

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
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

This will give you a bunch of options. We want to copy these settings:



figuration

Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

- □ Build after other projects are built ?
- □ Build periodically ?
- ☑ GitHub hook trigger for GITScm polling ?
- □ Poll SCM ?
- □ Trigger builds remotely (e.g., from scripts) ?

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition: Pipeline script from SCM

SCM ?: Git

Repositories ?

Repository URL ?: https://github.com/qasimshahid/qshahid-swe645-asst2/

Credentials ?: - none -

Branches to build ?

Branch Specifier (blank for 'any') ?: */main

Repository browser ?: (Auto)

Namely, do the following:

Select “GitHub hook trigger for GITScm polling” as a trigger

Pipeline definition should be from “Pipeline script from SCM”. This means it will use our Jenkinsfile in our GitHub repository. By default, the name of the file is Jenkinsfile, so we don’t need to change it.

Add the link to your GitHub repository.

Change the branch to “main”. This tells Jenkins to build the main branch. This is important because it is no longer named “master,” so make sure you enter “main” here.

After all of that, save the pipeline.

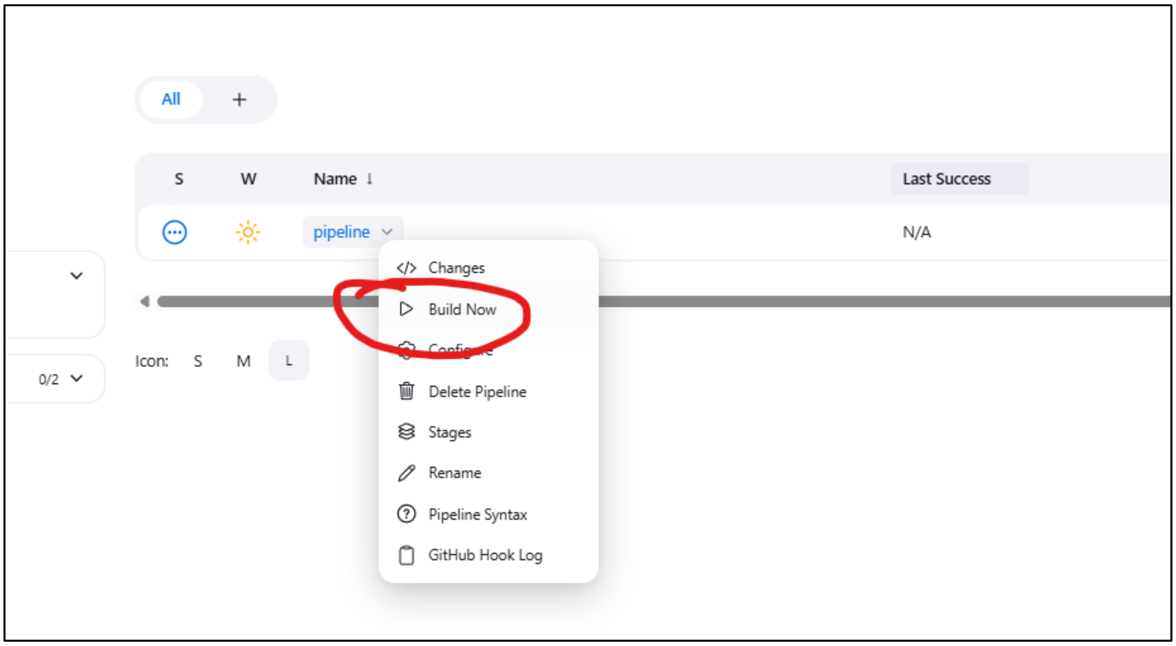
Congratulations, we are ready to do our first build. Everything should be setup properly now.

Running the Application

Assuming you followed all the rest of the steps, you should be ready to build and deploy the app to your cluster.

You can do these two ways.

Manually via Jenkins:



S	W	Name ↓	Last Success
		pipeline	N/A

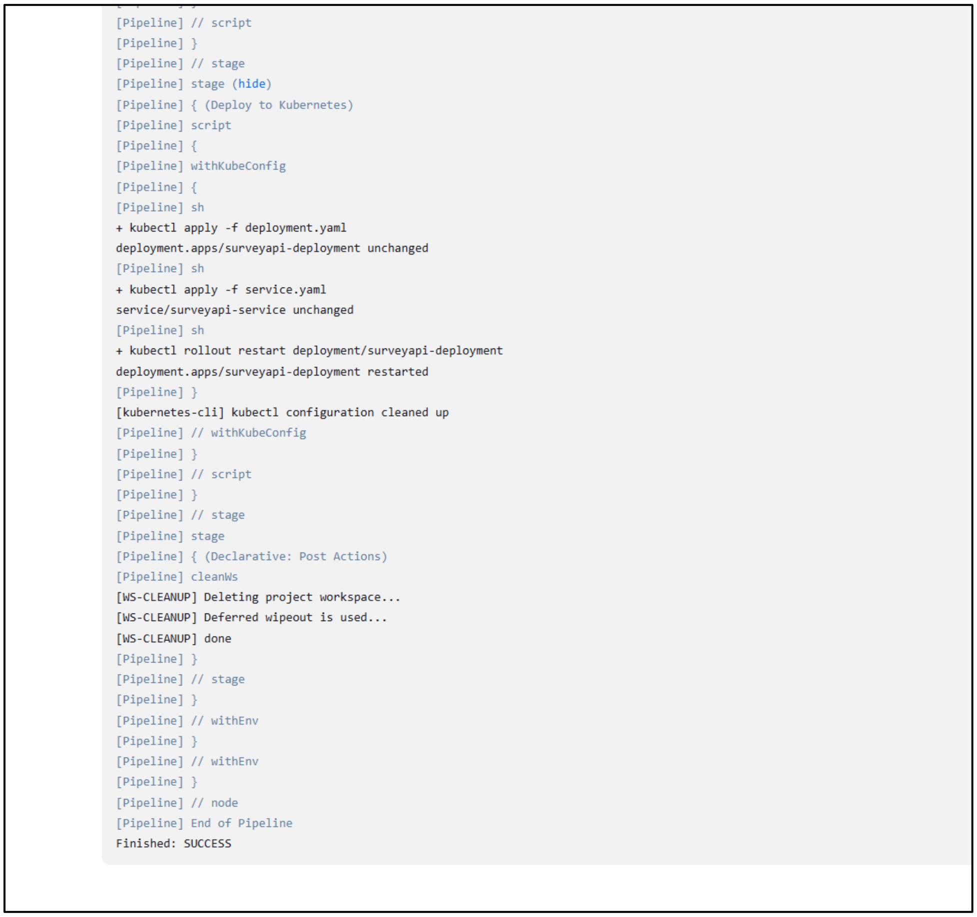
Alternatively, if you were able to correctly setup your webhook from your GitHub repository, then you can simply push updates to your main branch and the pipeline will be triggered automatically.

IF YOU HAVE ANY ISSUES WITH THE DOCKER BUILD STEP:

Execute this command and then restart the Jenkins server.

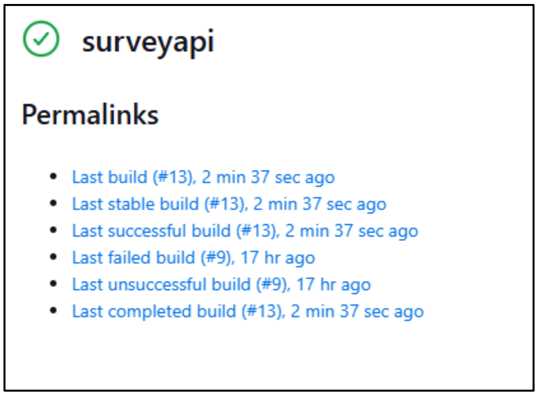
```
sudo usermod -a -G docker Jenkins
sudo reboot
```

Once you either push a change to GitHub or manually trigger the build through Jenkins, you should see this in the build output console:



```
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage (hide)
[Pipeline] { (Deploy to Kubernetes)
[Pipeline] script
[Pipeline] {
[Pipeline] withKubeConfig
[Pipeline] {
[Pipeline] sh
+ kubectl apply -f deployment.yaml
deployment.apps/surveyapi-deployment unchanged
[Pipeline] sh
+ kubectl apply -f service.yaml
service/surveyapi-service unchanged
[Pipeline] sh
+ kubectl rollout restart deployment/surveyapi-deployment
deployment.apps/surveyapi-deployment restarted
[Pipeline] }
[kubernetes-cli] kubectl configuration cleaned up
[Pipeline] // withKubeConfig
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] cleanWs
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] done
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Here you can see our build succeeding (after a couple mistakes I made when recreating this, if you follow the guide, you should not have these.)

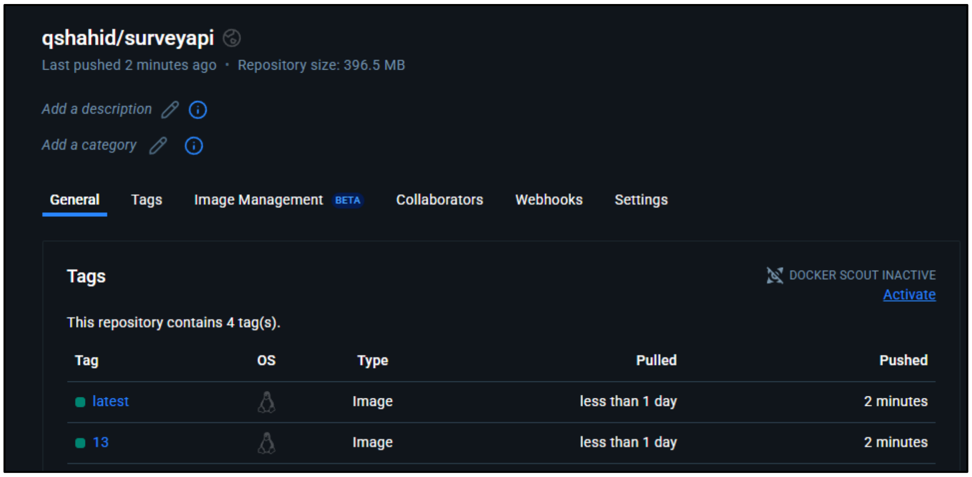


✓ surveyapi

Permalinks

- Last build (#13), 2 min 37 sec ago
- Last stable build (#13), 2 min 37 sec ago
- Last successful build (#13), 2 min 37 sec ago
- Last failed build (#9), 17 hr ago
- Last unsuccessful build (#9), 17 hr ago
- Last completed build (#13), 2 min 37 sec ago

New image successfully pushed to Docker Hub under my account.



qshahid/surveyapi

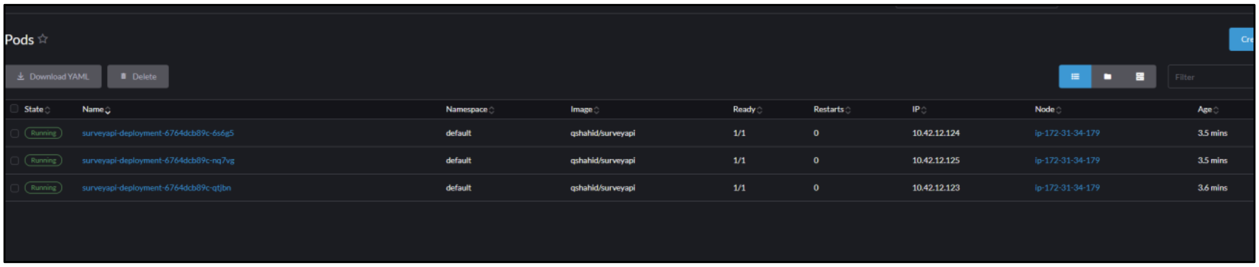
Last pushed 2 minutes ago · Repository size: 396.5 MB

Tags

This repository contains 4 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	less than 1 day	2 minutes
13		Image	less than 1 day	2 minutes

In Rancher, you can see the 3 pods / replicas we created as a part of our deployment.

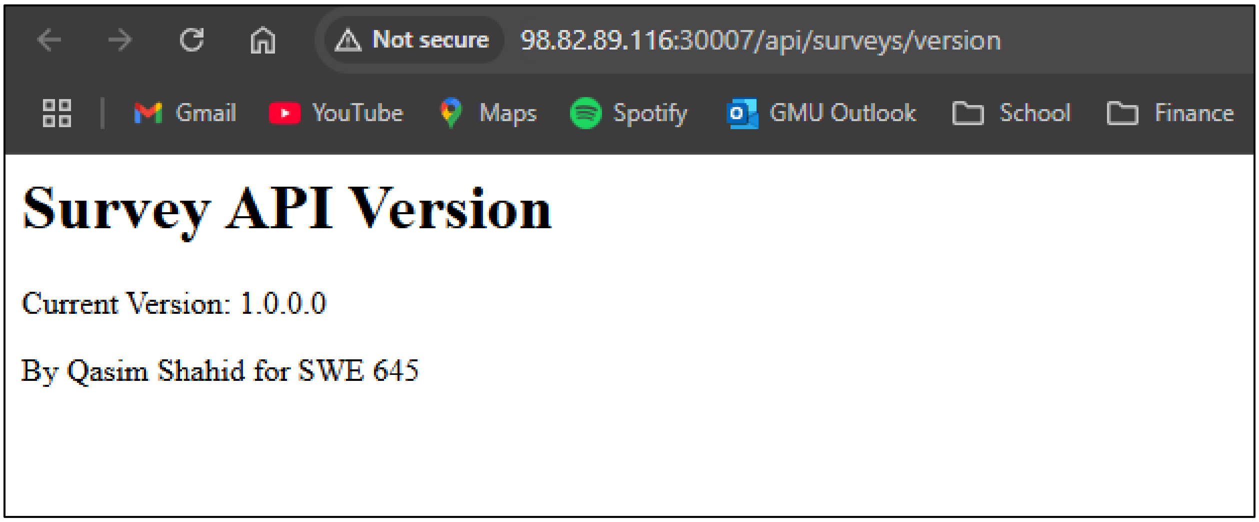


Pods

State	Name	Namespace	Image	Ready	Restarts	IP	Node	Age
Running	surveyapi-deployment-6764dcb89c-6s6g5	default	qshahid/surveyapi	1/1	0	10.42.12.124	ip-172-31-34-179	3.5 mins
Running	surveyapi-deployment-6764dcb89c-nq7vg	default	qshahid/surveyapi	1/1	0	10.42.12.125	ip-172-31-34-179	3.5 mins
Running	surveyapi-deployment-6764dcb89c-qtjbn	default	qshahid/surveyapi	1/1	0	10.42.12.123	ip-172-31-34-179	3.6 mins

You can access these at the Rancher Instance's public elastic IP. The NodePort service I used hosts the API on the 30007 port. So, you can go to:
http://<public_instance_ip>:30007/api/surveys/version
For example:
http://98.82.89.116:30007/api/surveys/version

Here you can see my API responding via some HTML when we ask for the version.

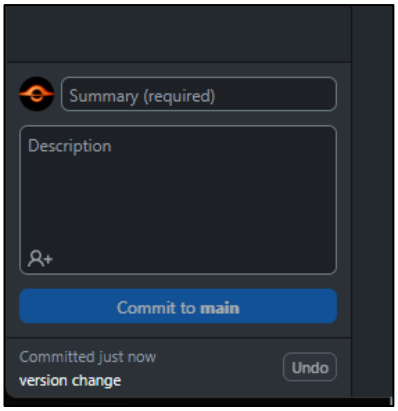


Survey API Version

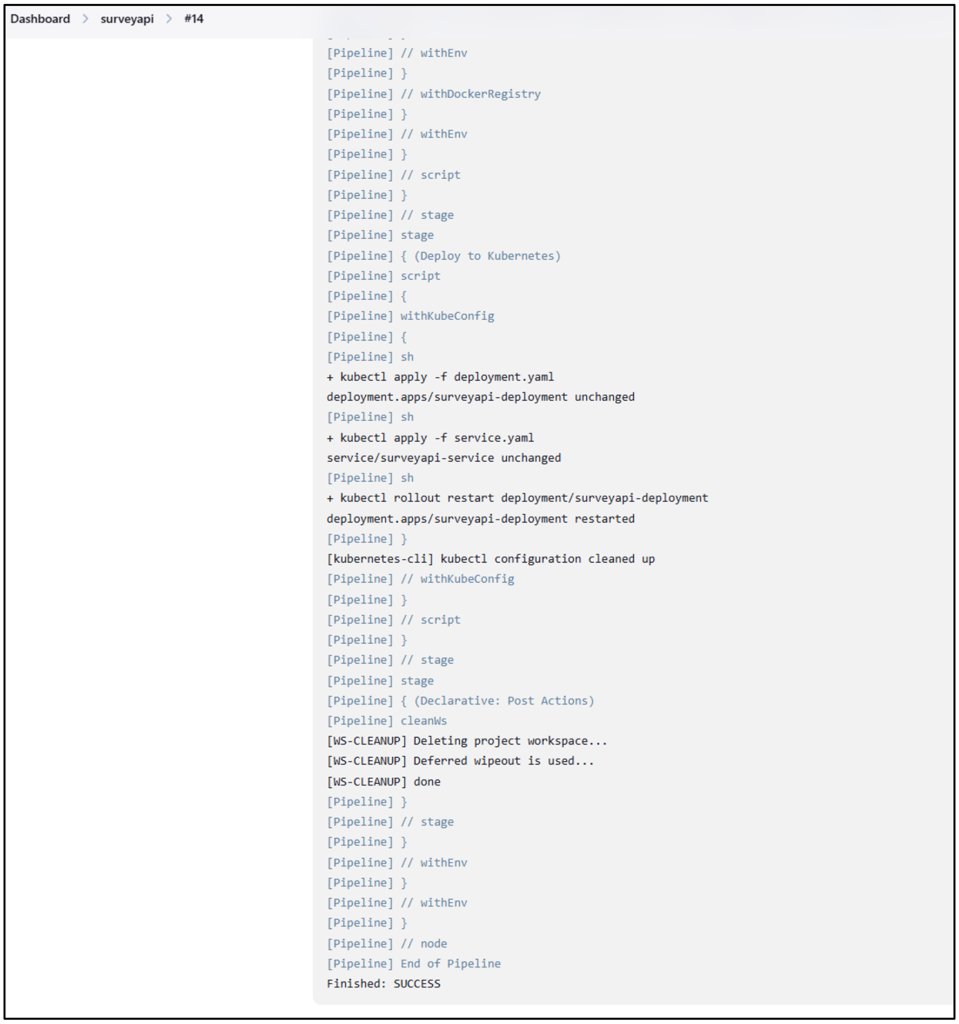
Current Version: 1.0.0.0

By Qasim Shahid for SWE 645

I'll now push a small change to GitHub, let's see what happens. We will change version to 2.0.0.0.

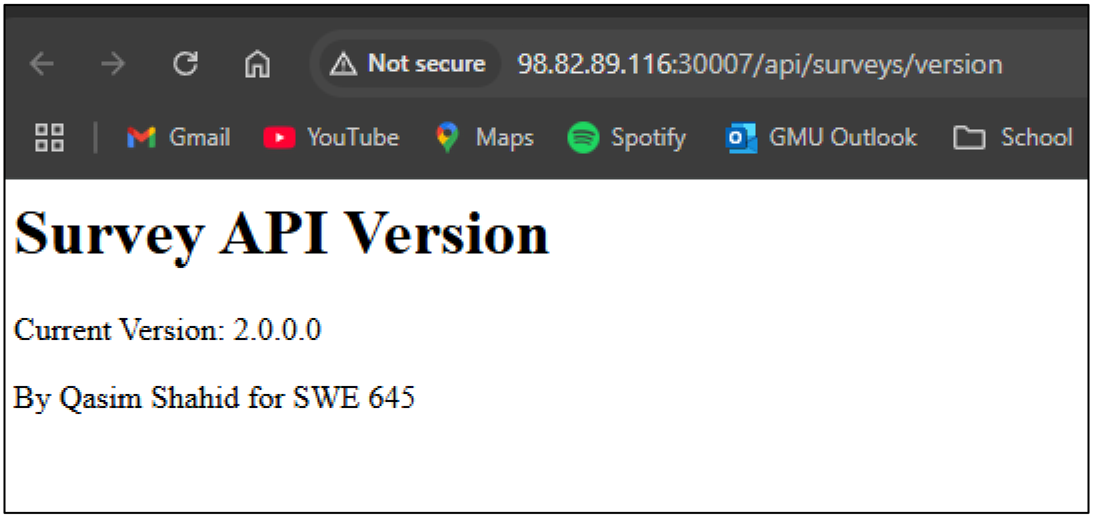


version change



```
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withDockerRegistry
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy to Kubernetes)
[Pipeline] script
[Pipeline] {
[Pipeline] withKubeConfig
[Pipeline] {
[Pipeline] sh
+ kubectl apply -f deployment.yaml
deployment.apps/surveyapi-deployment unchanged
[Pipeline] sh
+ kubectl apply -f service.yaml
service/surveyapi-service unchanged
[Pipeline] sh
+ kubectl rollout restart deployment/surveyapi-deployment
deployment.apps/surveyapi-deployment restarted
[Pipeline] }
[kubernetes-cli] kubectl configuration cleaned up
[Pipeline] // withKubeConfig
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] cleanWs
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] done
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Let's go to http://98.82.89.116:30007/api/surveys/version again and see if our pipeline deployed our change.



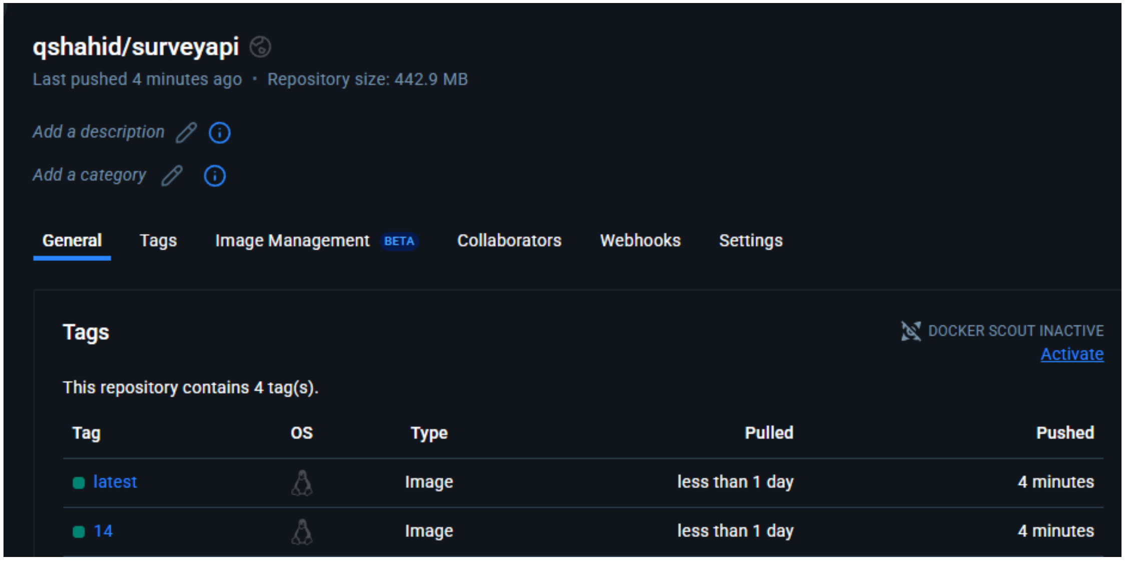
Survey API Version

Current Version: 2.0.0.0

By Qasim Shahid for SWE 645

You can see that it worked. We are now on version 2.0.0.0, which we just pushed. Our pipeline listened to the push event and automatically deployed the change to our cluster.

Successful upload to Docker Hub as well.



qshahid/surveyapi

Last pushed 4 minutes ago · Repository size: 442.9 MB

Add a description

Add a category

Tags

This repository contains 4 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	less than 1 day	4 minutes
14		Image	less than 1 day	4 minutes

NOTE: I won't be testing the API for doing CRUD operations, but this is covered in the demo_video.mp4 file which will be included with the submission. In that, we go through all the CRUD operations on our APIs and make sure they are being persisted in our database.

References:

https://docs.docker.com/security/for-developers/access-tokens/
https://kubernetes.io/
https://plugins.jenkins.io/pipeline-github/
https://www.cprime.com/resources/blog/how-to-integrate-jenkins-github/
https://www.rancher.com/
https://ranchermanager.docs.rancher.com/
https://plugins.jenkins.io/kubernetes-cli/
https://medium.com/@pratik.941/building-rest-api-using-spring-boot-a-comprehensive-guide-3e9b6d7a8951
https://start.spring.io/
https://askubuntu.com/questions/740757/switch-between-multiple-java-versions
https://www.baeldung.com/executable-jar-with-maven#:~:text=To%20build%20a%20jar%2C%20we,%3A8080%2F%20in%20a%20browser.

Lessons Learned:

This assignment, I learned how to deploy REST APIs to a Kubernetes cluster. In the last assignment, we did something very similar for a very simple web server and HTML file, and this assignment, we did basically that but the backend. So combined, these two assignments have taught me how to almost deploy an entire website with a front and back end to Kubernetes. I also learned a lot about how to develop APIs using Spring, and also learned how dependencies are managed with Spring projects (Maven). Overall, I learned a lot about how to create and deploy an API which connects to a database, and then create an automated pipeline that builds and deploys it for me.