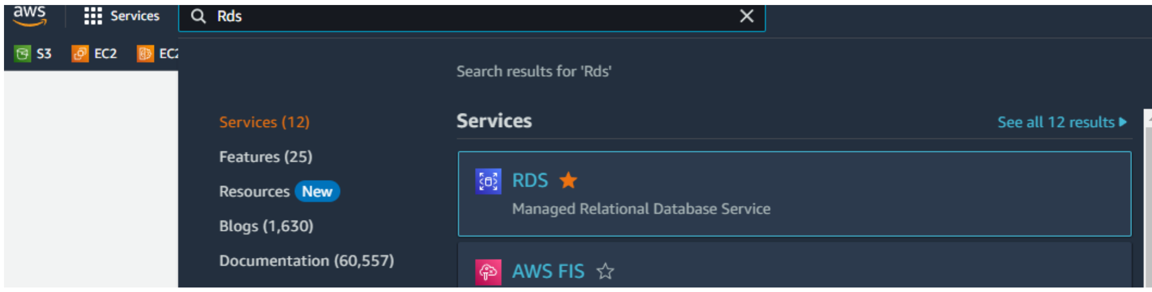
Setup/Installation Instructions

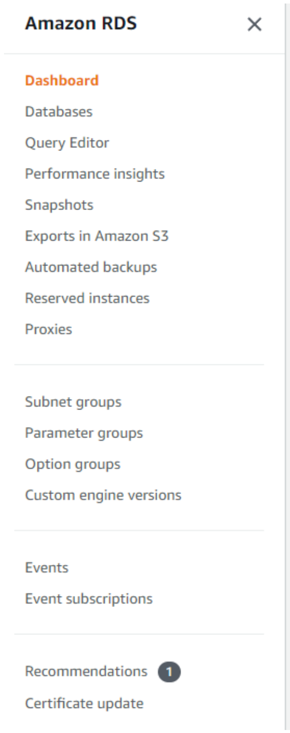
Note: These instructions assume that the reader has implemented the steps from HW2, found here: https://github.com/yang9501/SWE645HW2

I. RDS Setup

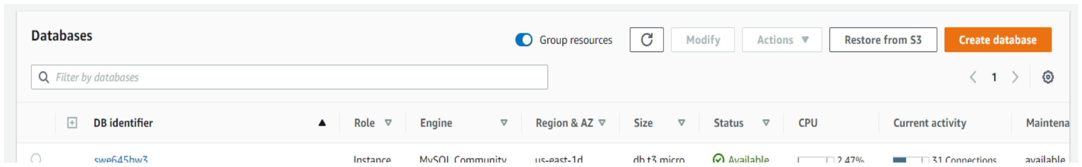
A. Enter the AWS Management Console and enter the RDS Services page

1. 

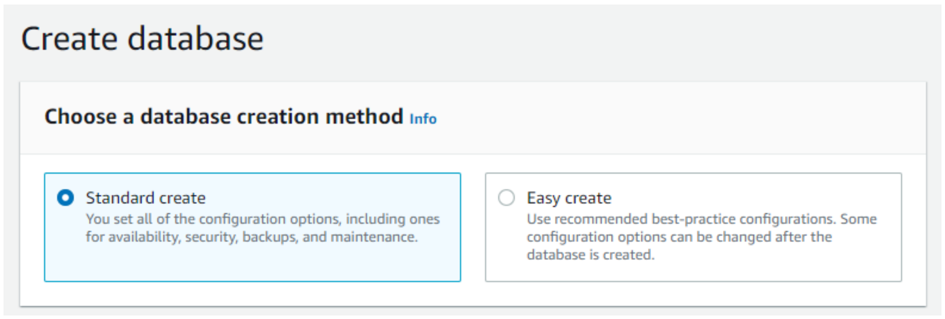
B. In the Left side menu, click the Databases menu item

1. 

C. Click the Create Database button

1. 

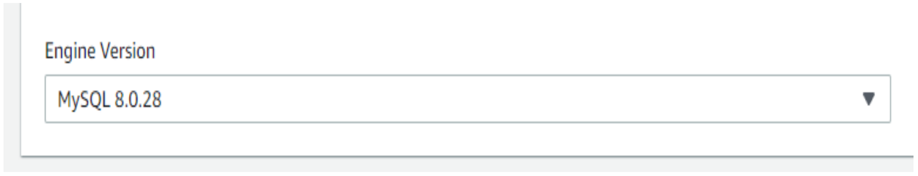
D. Within the Create Database page, select the following options

1. Keep the Standard Create option
   a) 
   Create database

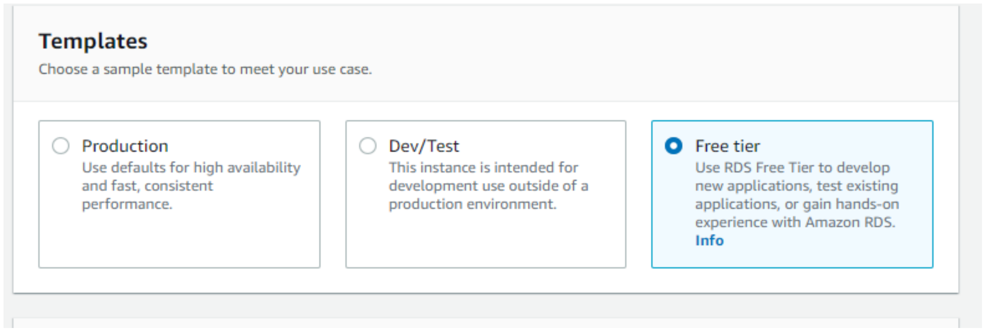
   Choose a database creation method Info

   ● 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.

2. Select the version of your MySQL Community implementation
   a) 
   Engine Version

   MySQL 8.0.28

3. Select Free Tier
   a) 
   Templates

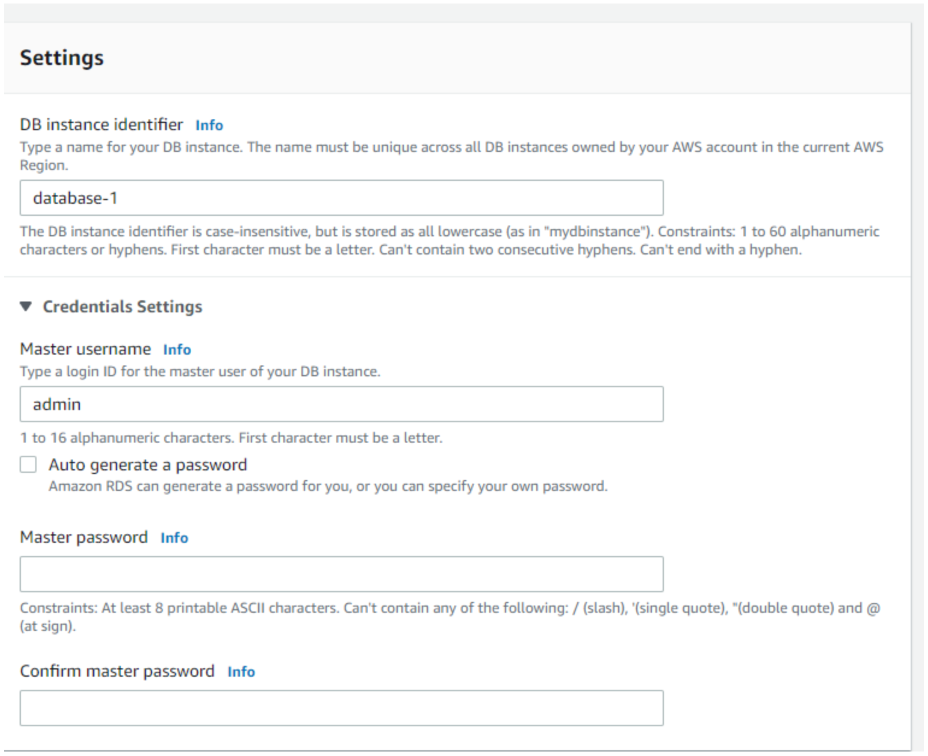
   Choose a sample template to meet your use case.

   ○ 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

4. Name your Database and create an admin user and password



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 60 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. First character must be a letter.

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

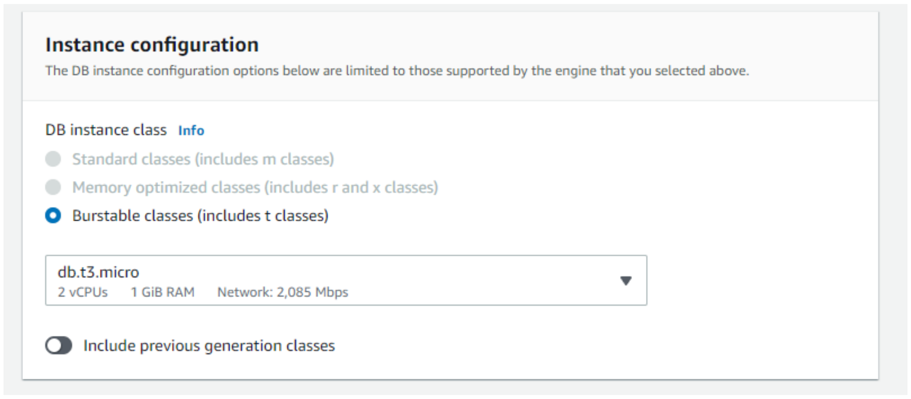
Master password Info

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), '(single quote), "(double quote) and @ (at sign).

Confirm master password Info

a)

5. Use the t3.micro instance



Instance configuration
The DB instance configuration options below are limited to those supported by the engine that you selected above.

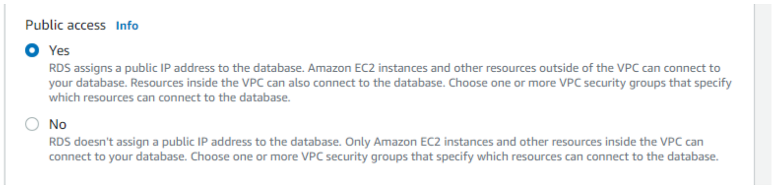
DB instance class Info
- Standard classes (includes m classes)
- Memory optimized classes (includes r and x classes)
- [✓ Burstable classes (includes t classes)]

db.t3.micro
2 vCPUs 1 GiB RAM Network: 2,085 Mbps

Include previous generation classes

a)

6. Turn on Public Access



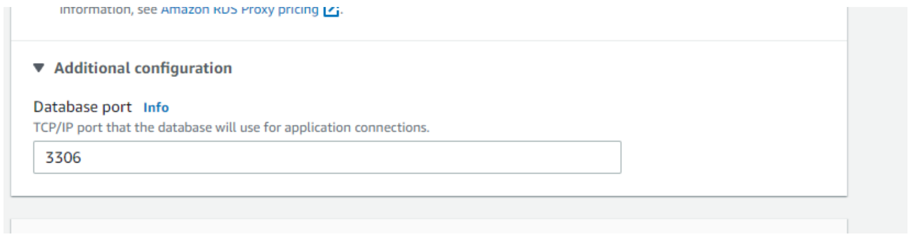
Public access Info

[✓ 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.

a)

7. Ensure that the Database Port is 3306



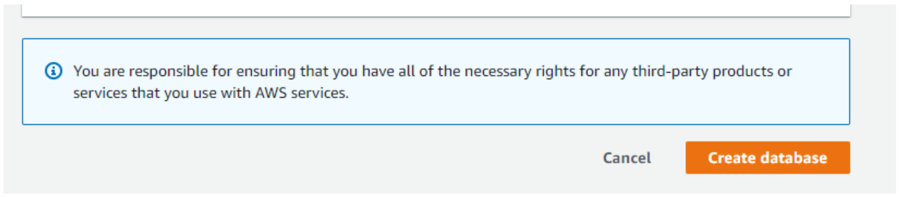
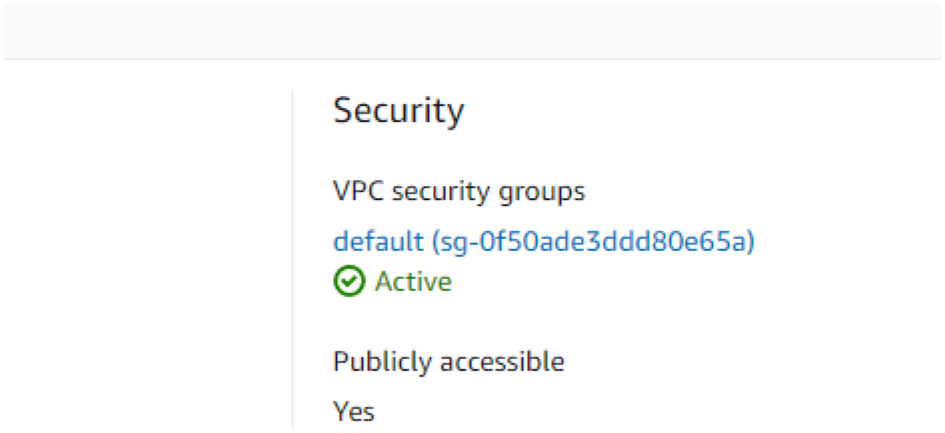
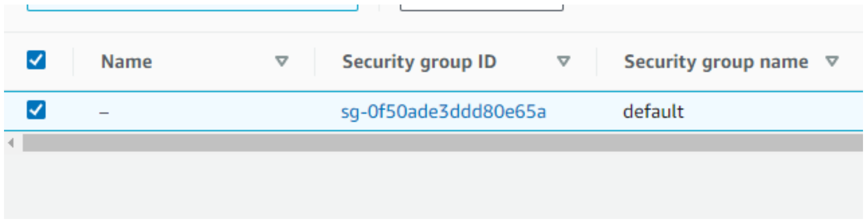
information, see Amazon RDS Proxy pricing

▼ **Additional configuration**

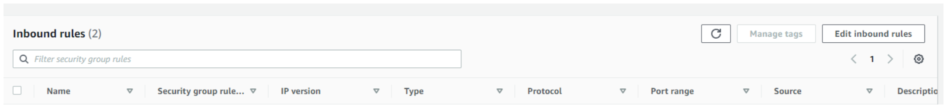
Database port Info
TCP/IP port that the database will use for application connections.

3306

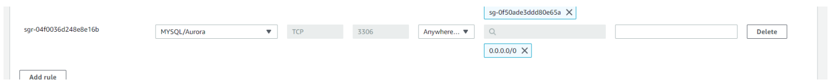
a)

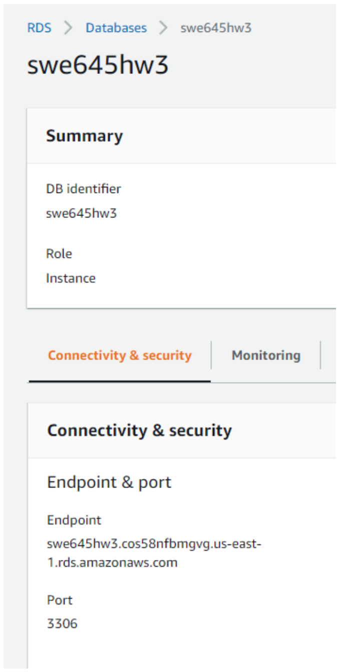
8. Click Create Database
    a) 
    You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services.
    Cancel | Create database
9. After your new database has finished creating, enter the Security Groups page to modify inbound rules
    a) 
    Security
    VPC security groups
    default (sg-0f50ade3ddd80e65a)
    Active
    Publicly accessible
    Yes
10. Click the default security group name to edit it
    a) 

	Name	Security group ID	Security group name
✓	–	sg-0f50ade3ddd80e65a	default

11. Click the Edit inbound rules button
    a) 
    Inbound rules (2) | Manage tags | Edit inbound rules
    Filter security group rules

	Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Descriptio

12. Ensure that there is a rule with the following attributes
    a) Type: MySQL/Aurora
    b) Protocol: TCP
    c) Port Range: 3306
    d) Source: Anywhere IPv4
    e) 
    sgr-04f0036d248e8e16b | MYSQL/Aurora | TCP | 3306 | Anywhere... | sg-0f50ade3ddd80e65a, 0.0.0.0/0 | Delete
    Add rule
13. Navigate back to your RDS instance and make note of your endpoint

a) 
RDS > Databases > swe645hw3

swe645hw3

Summary

DB identifier
swe645hw3

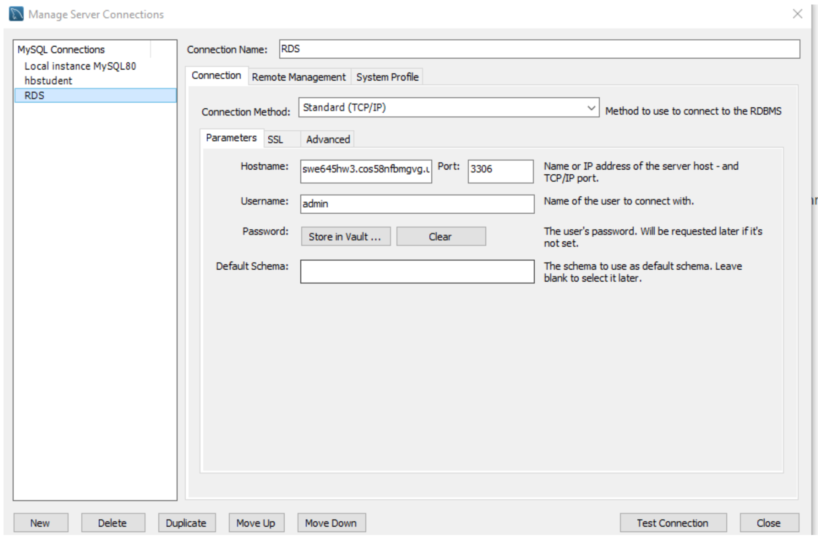
Role
Instance

Connectivity & security

Endpoint & port

Endpoint
swe645hw3.cos58nfbmgvg.us-east-1.rds.amazonaws.com

Port
3306

14. Open MySQL Workbench and create a new connection with the following information:
    a) Hostname: swe645hw3.xxxxxxxxxxxxxxxxx.us-east-1.rds.amazonaws.com (your RDS endpoint)
    b) Port: 3306
    c) Username: (your admin RDS user name)
    d) Password: (your admin RDS password)
    e) 

Manage Server Connections

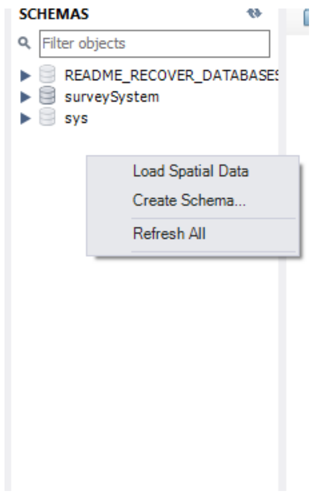
MySQL Connections
Local instance MySQL80
hbstudent
RDS

Connection Name: RDS

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

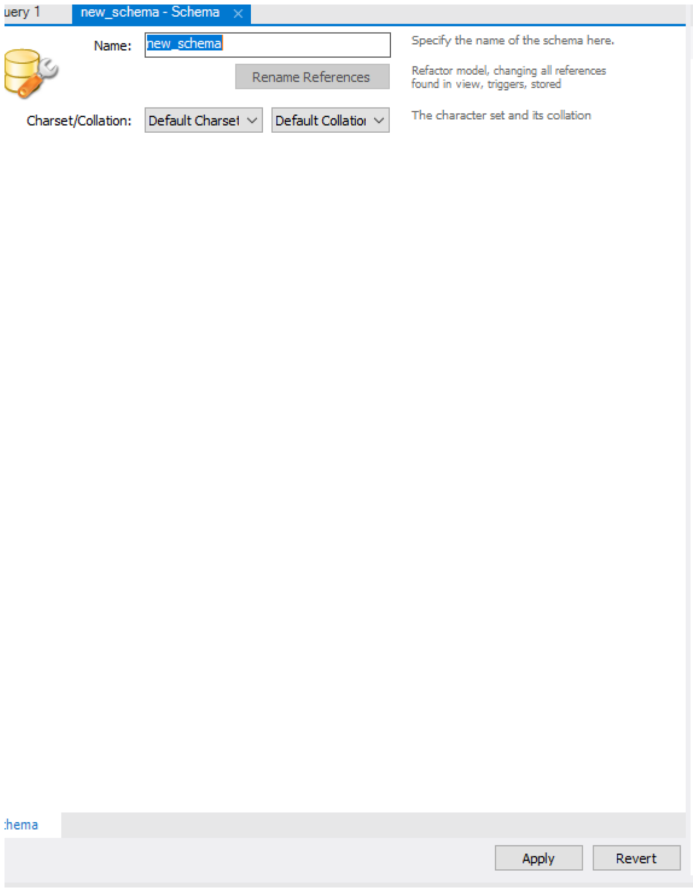
Hostname: swe645hw3.cos58nfbmgvg.u Port: 3306 — Name or IP address of the server host - and TCP/IP port.
Username: admin — Name of the user to connect with.
Password: Store in Vault ... Clear — The user's password. Will be requested later if it's not set.
Default Schema: — The schema to use as default schema. Leave blank to select it later.

15. Once connected, right click anywhere in the whitespace of the left sidebar and select “Create Schema”



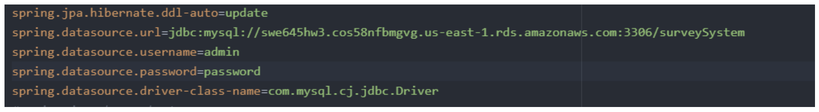
a)

16. Name it whatever you want for use with your backend and save it. In my case I used ‘surveySystem’


uery 1
new_schema - Schema
Name:
new_schema
Specify the name of the schema here.
Rename References
Refactor model, changing all references found in view, triggers, stored
Charset/Collation:
Default Charset
Default Collation
The character set and its collation
chema
Apply
Revert

a)

17. Modify your Spring Boot ‘application.properties’ file to point to your RDS instance

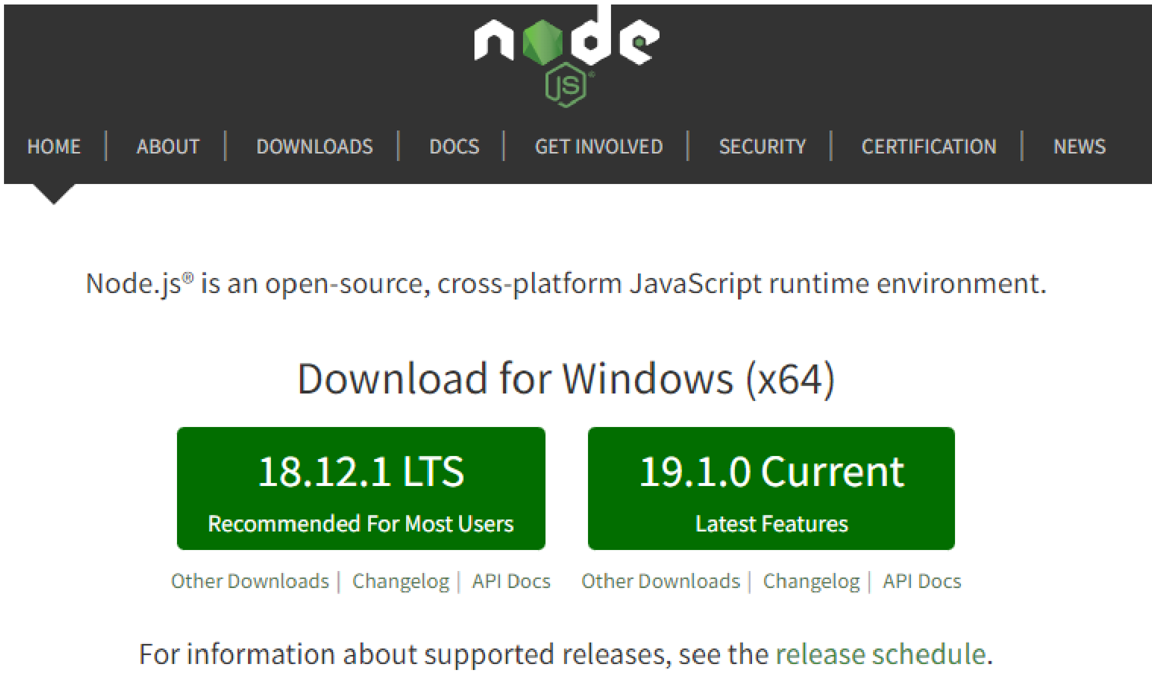


a)

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://swe645hw3.cos58nfbmgvg.us-east-1.rds.amazonaws.com:3306/surveySystem
spring.datasource.username=admin
spring.datasource.password=password
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

II. Frontend

A. Navigate to nodejs.org in your web browser and download the latest version of node and install it


HOME
ABOUT
DOWNLOADS
DOCS
GET INVOLVED
SECURITY
CERTIFICATION
NEWS

Node.js® is an open-source, cross-platform JavaScript runtime environment.

Download for Windows (x64)

18.12.1 LTS
Recommended For Most Users
19.1.0 Current
Latest Features
Other Downloads | Changelog | API Docs
Other Downloads | Changelog | API Docs

For information about supported releases, see the release schedule.

1.

B. Open the command prompt and enter the command: “npm install -g @angular/cli”

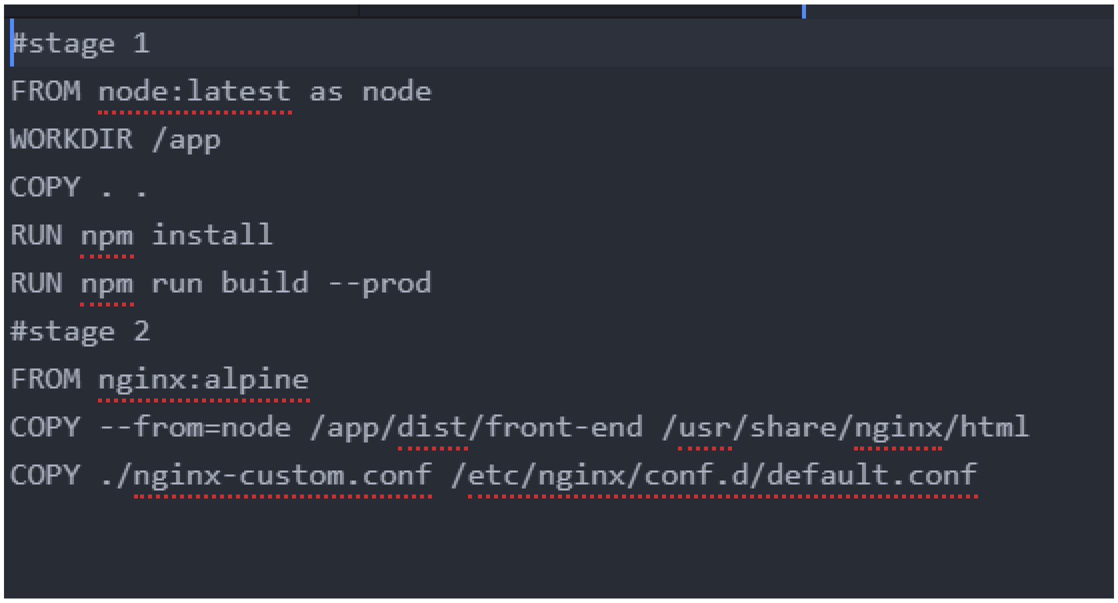
C. Verify that angular is installed by using the command: “ng -version”

D. Create a new angular project with the command : “ng new ‘projectname’”

E. You can run the project using “ng serve” and view it on http://localhost:4200

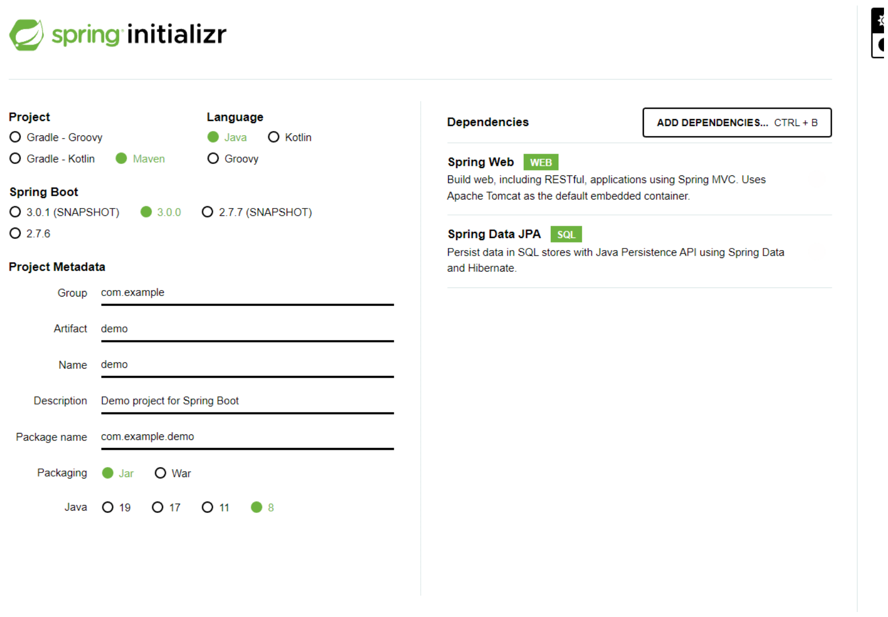
F. Implement your angular frontend service

G. Create a docker file in the root directory of your front end directory

1. 

```
#stage 1
FROM node:latest as node
WORKDIR /app
COPY . .
RUN npm install
RUN npm run build --prod
#stage 2
FROM nginx:alpine
COPY --from=node /app/dist/front-end /usr/share/nginx/html
COPY ./nginx-custom.conf /etc/nginx/conf.d/default.conf
```

III. Backend

A. In your web browser, navigate to https://start.spring.io/ and fill out the fields to start your Spring Boot project
   1. Project: Maven
   2. Language: Java
   3. Spring Boot: 3.0.0
   4. Packaging: Jar
   5. Java: 11
   6. Dependencies:
      a) Spring Web
      b) Spring Data JPA
   7. 

spring initializr

Project
Gradle - Groovy Gradle - Kotlin [✓ Maven]

Language
[✓ Java] Kotlin Groovy

Spring Boot
3.0.1 (SNAPSHOT) [✓ 3.0.0] 2.7.7 (SNAPSHOT) 2.7.6

Project Metadata

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot
Package name	com.example.demo
Packaging	[✓ Jar] War
Java	19 17 11 [✓ 8]

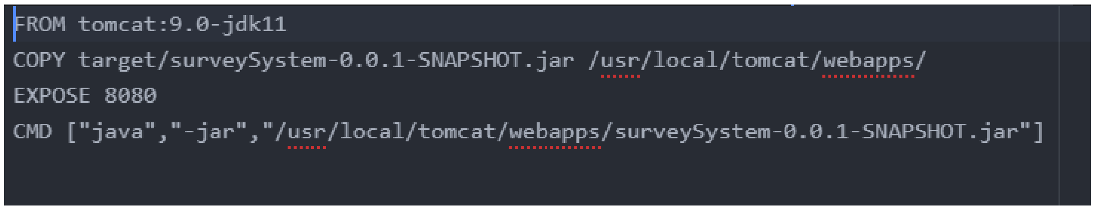
Dependencies ADD DEPENDENCIES... CTRL + B

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

B. Implement your Spring Boot backend service

C. Create and configure a Dockerfile in the root directory of the project

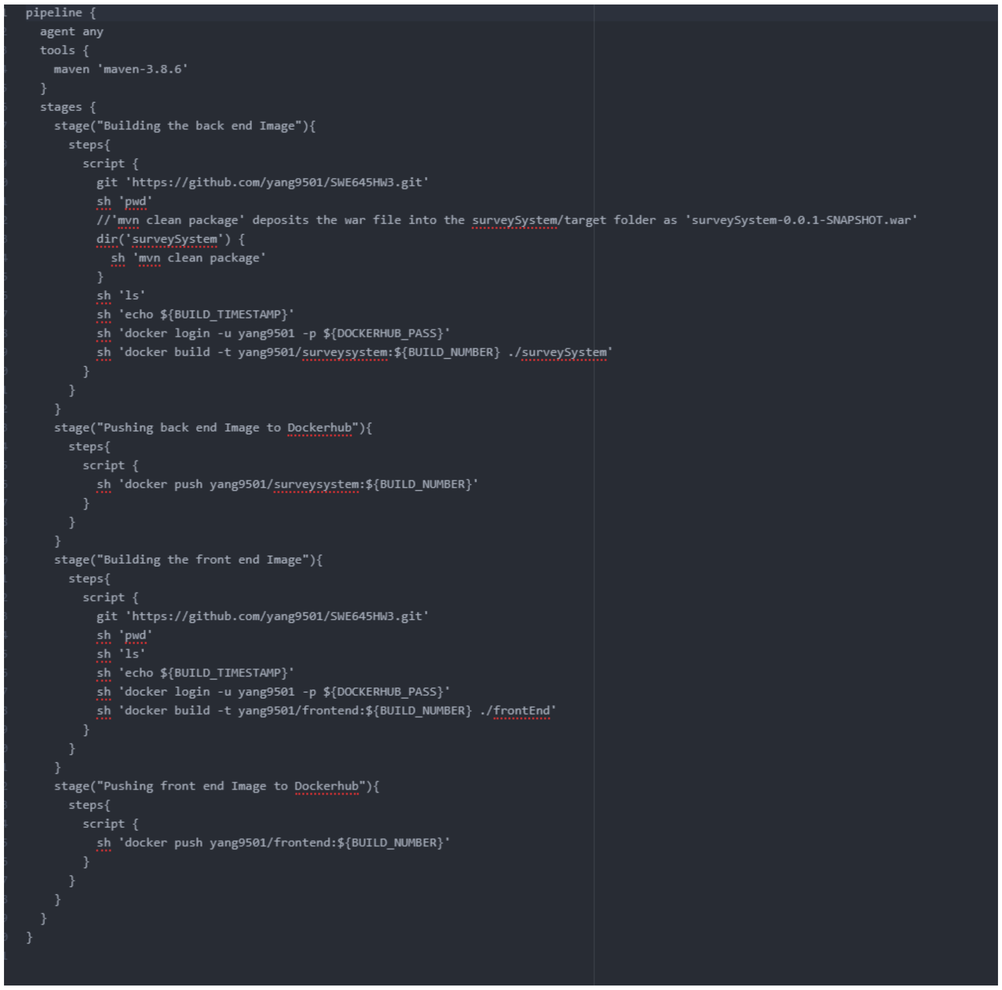
1. 

```
FROM tomcat:9.0-jdk11
COPY target/surveySystem-0.0.1-SNAPSHOT.jar /usr/local/tomcat/webapps/
EXPOSE 8080
CMD ["java","-jar","/usr/local/tomcat/webapps/surveySystem-0.0.1-SNAPSHOT.jar"]
```

IV. Jenkinsfile

A. Modify Jenkins to use the build number as the docker image tag

B. Create the Jenkinsfile

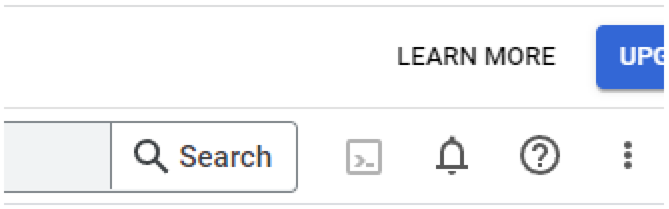
C. 

```
pipeline {
  agent any
  tools {
    maven 'maven-3.8.6'
  }
  stages {
    stage("Building the back end Image"){
      steps{
        script {
          git 'https://github.com/yang9501/SWE645HW3.git'
          sh 'pwd'
          //'mvn clean package' deposits the war file into the surveySystem/target folder as 'surveySystem-0.0.1-SNAPSHOT.war'
          dir('surveySystem') {
            sh 'mvn clean package'
          }
          sh 'ls'
          sh 'echo ${BUILD_TIMESTAMP}'
          sh 'docker login -u yang9501 -p ${DOCKERHUB_PASS}'
          sh 'docker build -t yang9501/surveysystem:${BUILD_NUMBER} ./surveySystem'
        }
      }
    }
    stage("Pushing back end Image to Dockerhub"){
      steps{
        script {
          sh 'docker push yang9501/surveysystem:${BUILD_NUMBER}'
        }
      }
    }
    stage("Building the front end Image"){
      steps{
        script {
          git 'https://github.com/yang9501/SWE645HW3.git'
          sh 'pwd'
          sh 'ls'
          sh 'echo ${BUILD_TIMESTAMP}'
          sh 'docker login -u yang9501 -p ${DOCKERHUB_PASS}'
          sh 'docker build -t yang9501/frontend:${BUILD_NUMBER} ./frontEnd'
        }
      }
    }
    stage("Pushing front end Image to Dockerhub"){
      steps{
        script {
          sh 'docker push yang9501/frontend:${BUILD_NUMBER}'
        }
      }
    }
  }
}
```

V. EXTRA CREDIT: ArgoCD

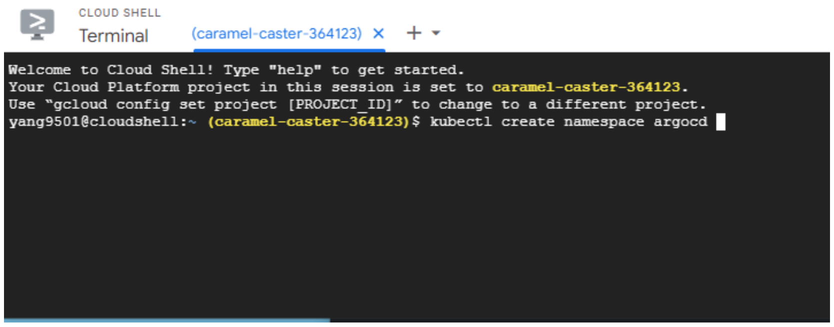
A. Install ArgoCD on the cluster host

1. Enter the cloud shell terminal



a)

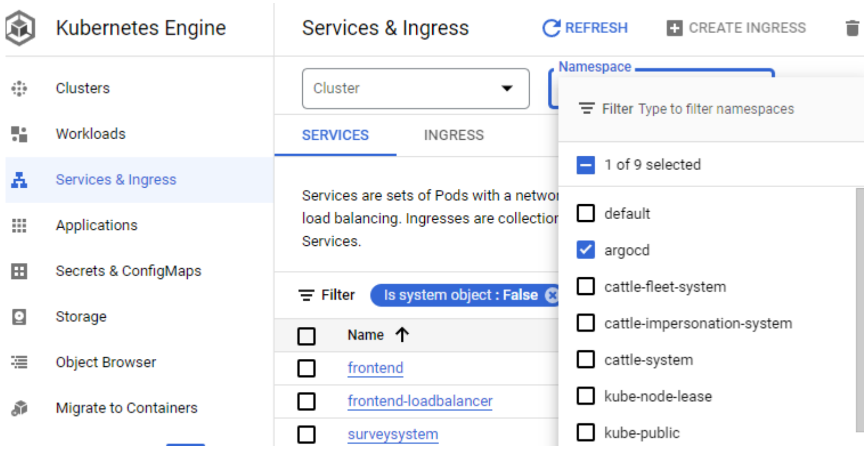
2. Enter the following command:
    a) kubectl create namespace argocd



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to caramel-caster-364123.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
yang9501@cloudshell:~ (caramel-caster-364123)$ kubectl create namespace argocd
```

    b)

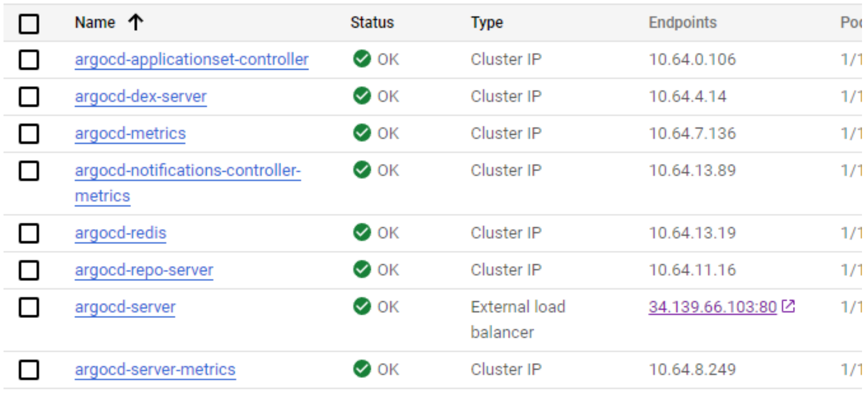
3. Enter the following command:
    a) kubectl apply -n argocd –f https://raw.githubusercontent.com/argoproj/argocd/stable/manifests/install.yaml
4. Create a loadbalancer so that the ArgoCD UI is accessible
    a) kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'
5. View the initial password by entering the command
    a) kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d; echo
6. Locate the ArgoCD UI by entering your cluster management console and selecting the Services and Ingress menu option and selecting the 'argocd' namespace checkbox:



Services are sets of Pods with a netwo[rk]... load balancing. Ingresses are collection... Services.

    a)

7. View the 'argocd-server' service and click the associated Endpoint link

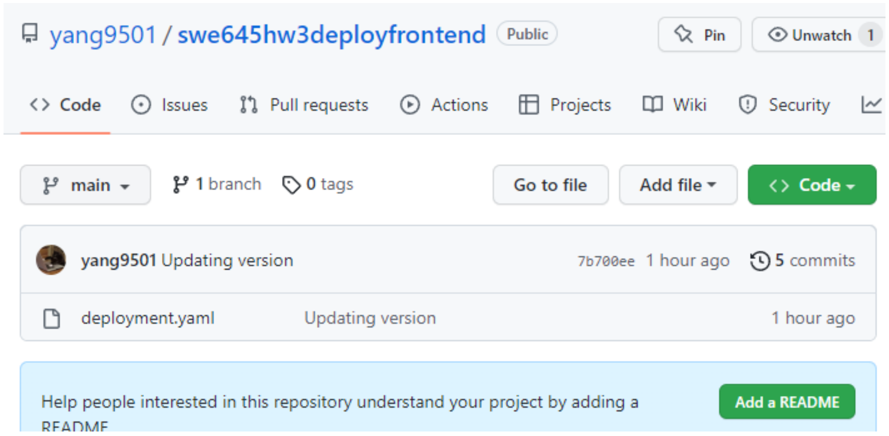


	Name ↑	Status	Type	Endpoints	Po
☐	argocd-applicationset-controller	✓ OK	Cluster IP	10.64.0.106	1/1
☐	argocd-dex-server	✓ OK	Cluster IP	10.64.4.14	1/1
☐	argocd-metrics	✓ OK	Cluster IP	10.64.7.136	1/1
☐	argocd-notifications-controller-metrics	✓ OK	Cluster IP	10.64.13.89	1/1
☐	argocd-redis	✓ OK	Cluster IP	10.64.13.19	1/1
☐	argocd-repo-server	✓ OK	Cluster IP	10.64.11.16	1/1
☐	argocd-server	✓ OK	External load balancer	34.139.66.103:80	1/1
☐	argocd-server-metrics	✓ OK	Cluster IP	10.64.8.249	1/1

a)

8. You'll be taken to the ArgoCD UI login page
   a) Enter the following credentials:
   b) User: admin
   c) Password: The string from step 5a.

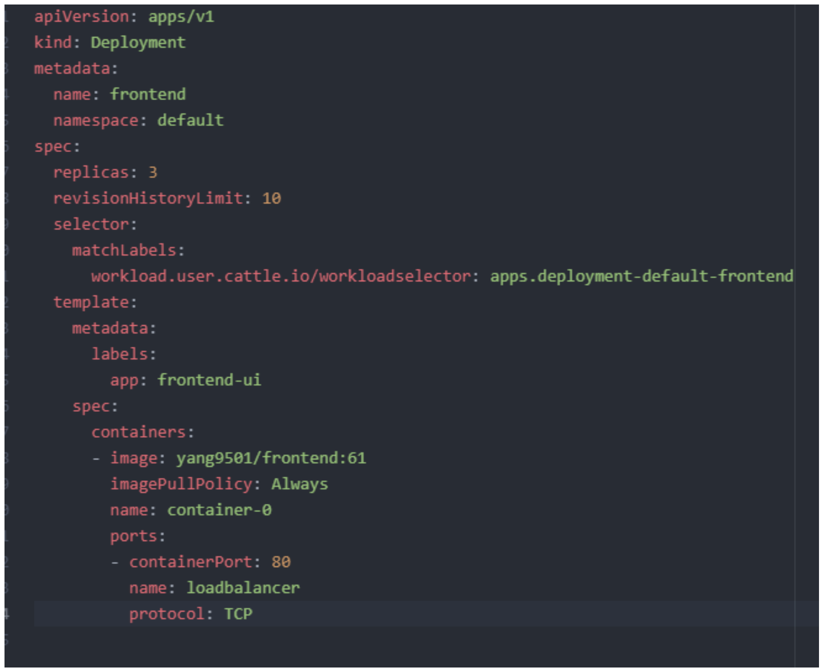
B. Create deployment repos and deployment.yamls for each of the services
   1. Front end (https://github.com/yang9501/swe645hw3deployfrontend)
      a) No other files are necessary besides the deployment.yaml file



yang9501 Updating version 7b700ee 1 hour ago 5 commits

deployment.yaml Updating version 1 hour ago

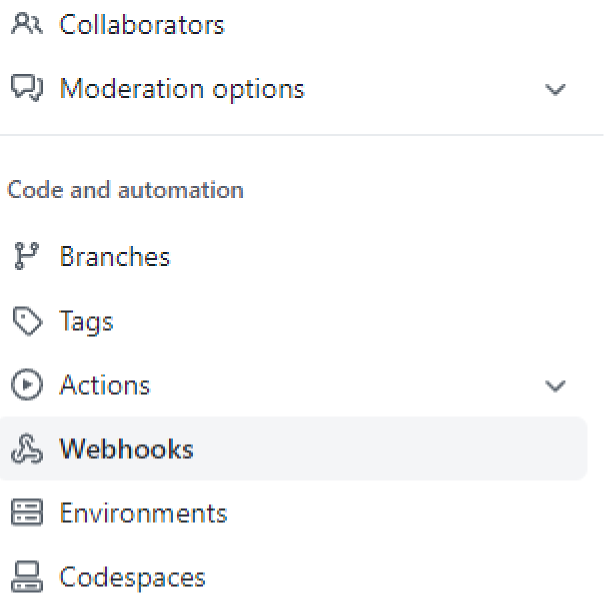
Help people interested in this repository understand your project by adding a README

b) 

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  namespace: default
spec:
  replicas: 3
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      workload.user.cattle.io/workloadselector: apps.deployment-default-frontend
  template:
    metadata:
      labels:
        app: frontend-ui
    spec:
      containers:
      - image: yang9501/frontend:61
        imagePullPolicy: Always
        name: container-0
        ports:
        - containerPort: 80
          name: loadbalancer
          protocol: TCP
```

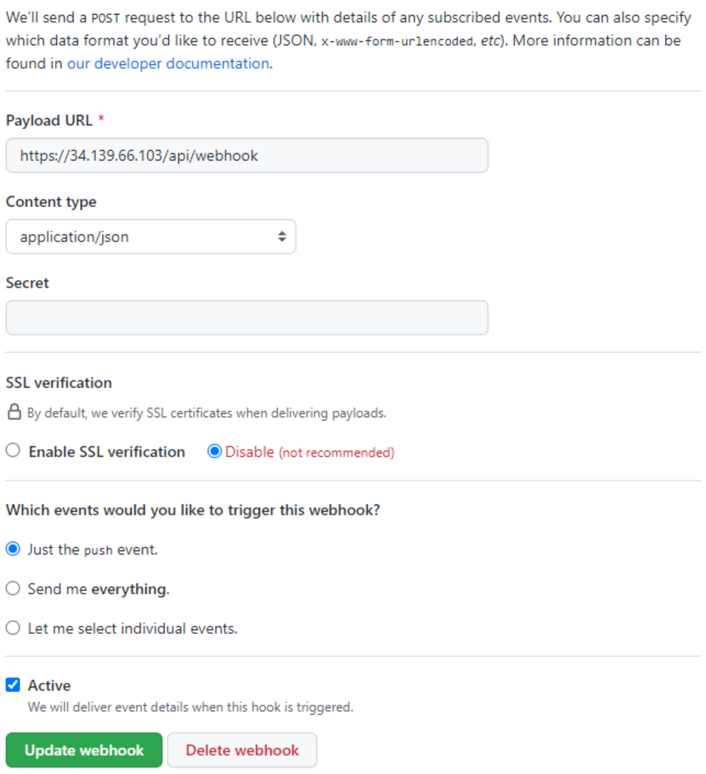
c) Create a Github webhook to let ArgoCD know when the deployment file is updated

(1) Within the Settings tab of your repository, select the Webhooks menu option

(2) 

(3) Click the Add webhook button

(4) Use the loadbalancer IP address as the url + "/api/webhook/". Ensure the content type is json and SSL verification is disabled



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 *

https://34.139.66.103/api/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**.

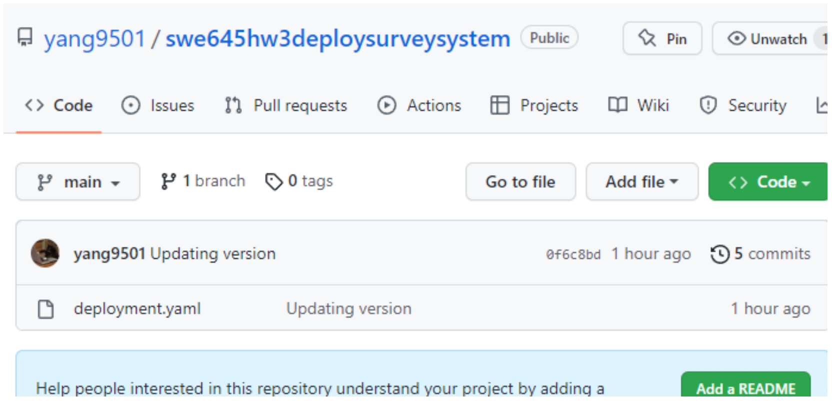
Let me select individual events.

[✓ Active]

We will deliver event details when this hook is triggered.

(5)

2. Backend (https://github.com/yang9501/swe645hw3deploysurveysystem)
    a) No other files are necessary besides the deployment.yaml file

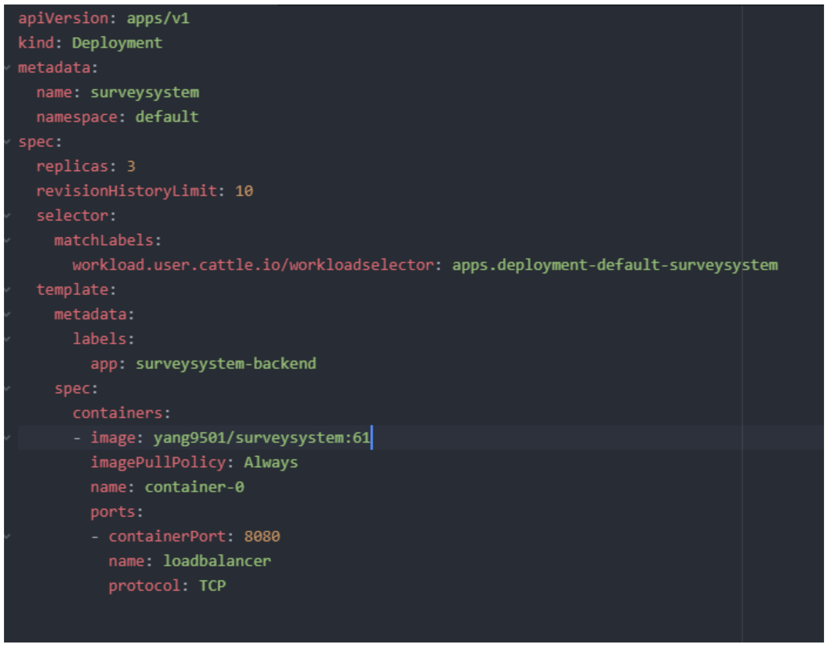


yang9501 / swe645hw3deploysurveysystem Public

yang9501 Updating version 0f6c8bd 1 hour ago 5 commits

deployment.yaml	Updating version	1 hour ago

Help people interested in this repository understand your project by adding a

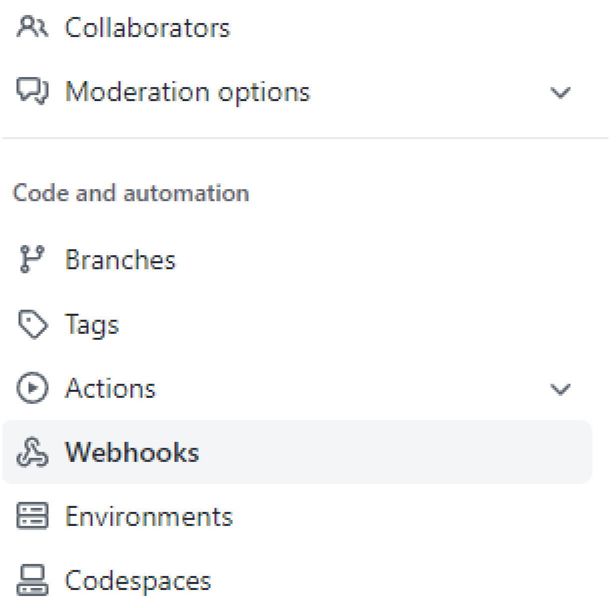


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: surveysystem
  namespace: default
spec:
  replicas: 3
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      workload.user.cattle.io/workloadselector: apps.deployment-default-surveysystem
  template:
    metadata:
      labels:
        app: surveysystem-backend
    spec:
      containers:
      - image: yang9501/surveysystem:61
        imagePullPolicy: Always
        name: container-0
        ports:
        - containerPort: 8080
          name: loadbalancer
          protocol: TCP
```

b)

c) Create a Github webhook to let ArgoCD know when the deployment file is updated

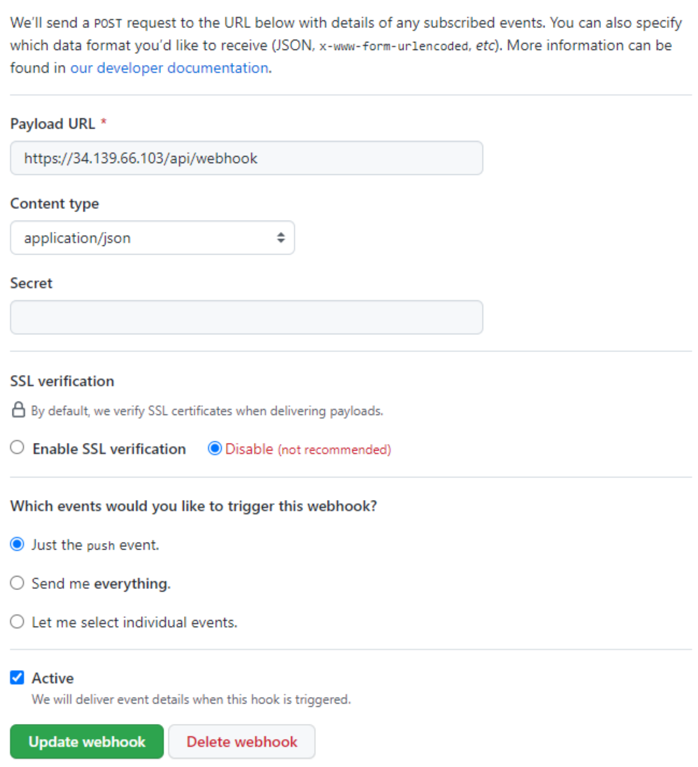
(1) Within the Settings tab of your repository, select the Webhooks menu option



(2)

(3) Click the Add webhook button

(4) Use the loadbalancer IP address as the url + “/api/webhook/”. Ensure the content type is json and SSL verification is disabled



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 *
https://34.139.66.103/api/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.

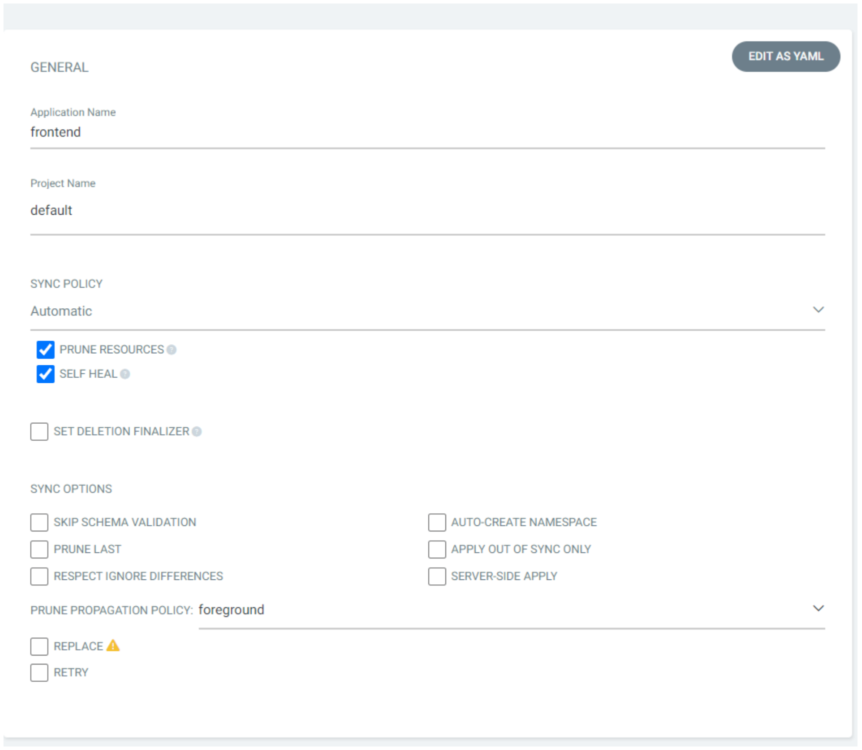
(5)

C. Configure deployments in ArgoCD

1. Create a new front end deployment

a) Set up the deployment as follows:

b) Ensure that the Sync policy is automatic, and Prune Resources and Self Heal are checked



GENERAL

Application Name
frontend

Project Name
default

SYNC POLICY
Automatic

[✓ PRUNE RESOURCES]
[✓ SELF HEAL]

SET DELETION FINALIZER

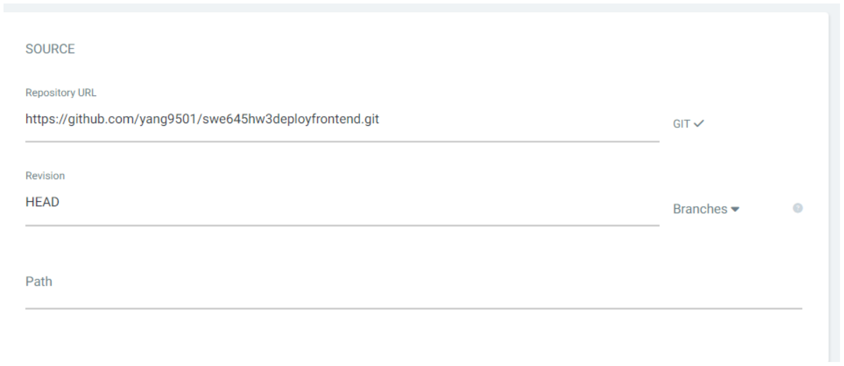
SYNC OPTIONS

SKIP SCHEMA VALIDATION
PRUNE LAST
RESPECT IGNORE DIFFERENCES
AUTO-CREATE NAMESPACE
APPLY OUT OF SYNC ONLY
SERVER-SIDE APPLY

PRUNE PROPAGATION POLICY: foreground

REPLACE ⚠
RETRY

c)


SOURCE

Repository URL

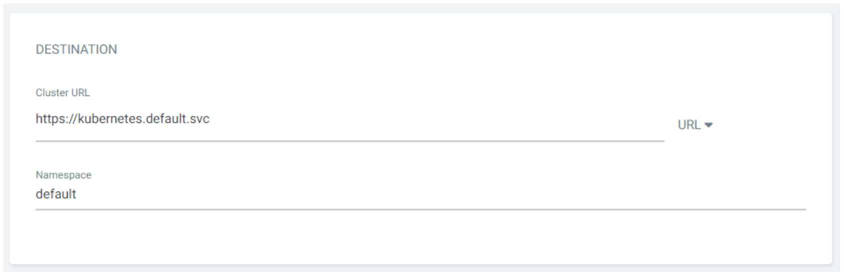
https://github.com/yang9501/swe645hw3deployfrontend.git GIT ✓

Revision

HEAD Branches

Path

d)


DESTINATION

Cluster URL

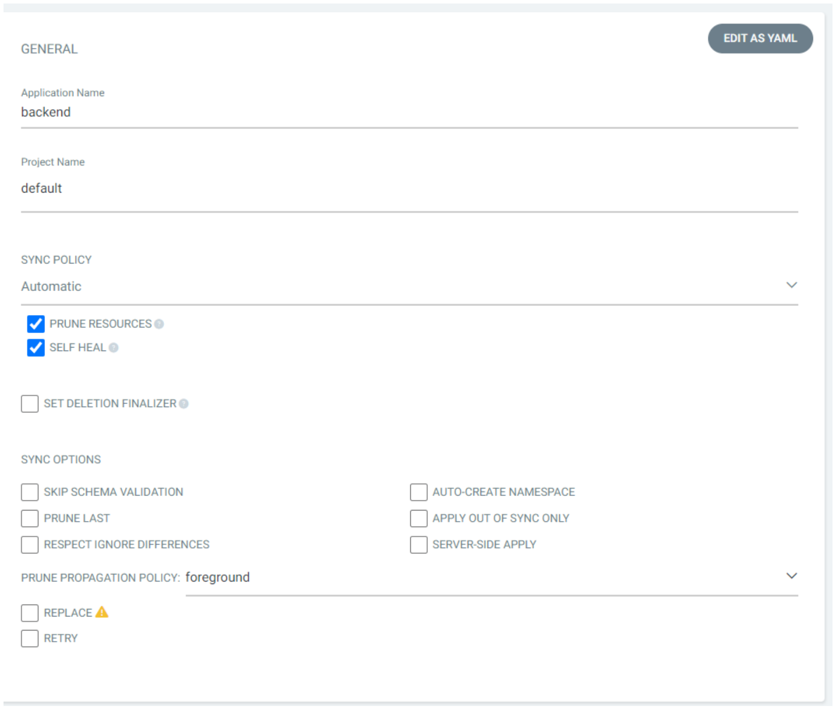
https://kubernetes.default.svc URL

Namespace

default

e)

2. Create the backend deployment
   a) Set up the deployment as follows



GENERAL

Application Name

backend

Project Name

default

SYNC POLICY

Automatic

- [x] PRUNE RESOURCES
- [x] SELF HEAL

- [ ] SET DELETION FINALIZER

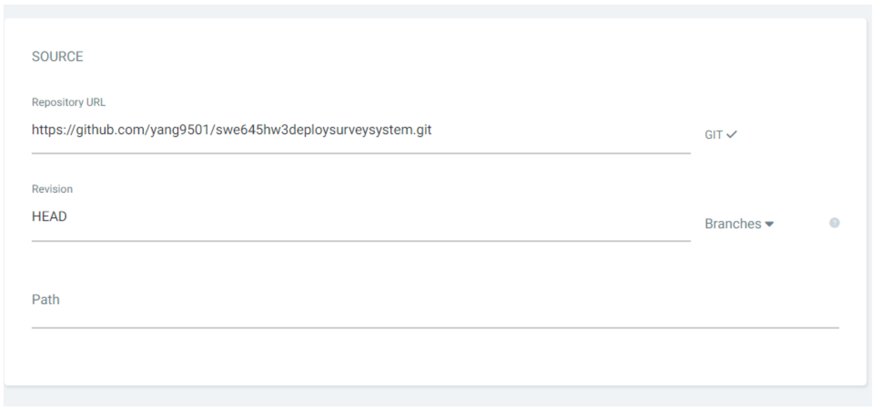
SYNC OPTIONS

- [ ] SKIP SCHEMA VALIDATION
- [ ] AUTO-CREATE NAMESPACE
- [ ] PRUNE LAST
- [ ] APPLY OUT OF SYNC ONLY
- [ ] RESPECT IGNORE DIFFERENCES
- [ ] SERVER-SIDE APPLY

PRUNE PROPAGATION POLICY: foreground

- [ ] REPLACE
- [ ] RETRY

b)



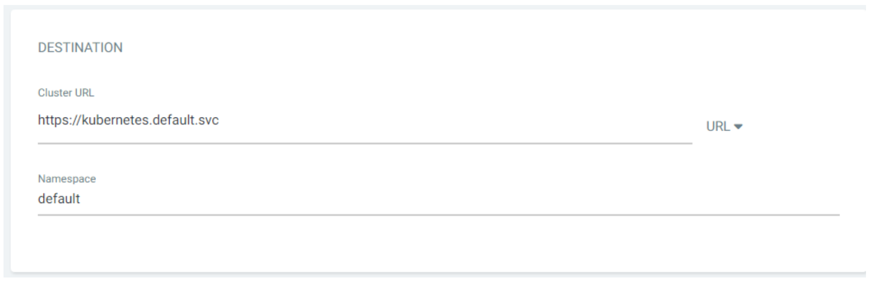
SOURCE

Repository URL
https://github.com/yang9501/swe645hw3deploysurveysystem.git

Revision
HEAD

Path

c)



DESTINATION

Cluster URL
https://kubernetes.default.svc

Namespace
default

d)