

Documentation SWE645 HW3

Group Members :

Calvin Johnson Joseph - G01369895
Nikhil Billakanti - G01393314
Yaswanth Devarapalli - G01390147
NagendraBabu Pulipati – G01372668

Set-Up Instructions:

Initially created an empty repository in GitHub and pushed the files into the repository using IntelliJ also pushed the Dockerfile and Jenkinfile into the repo.

Git link- <https://github.com/calvinjohnson747/swe645hw3>

Amazon RDS:

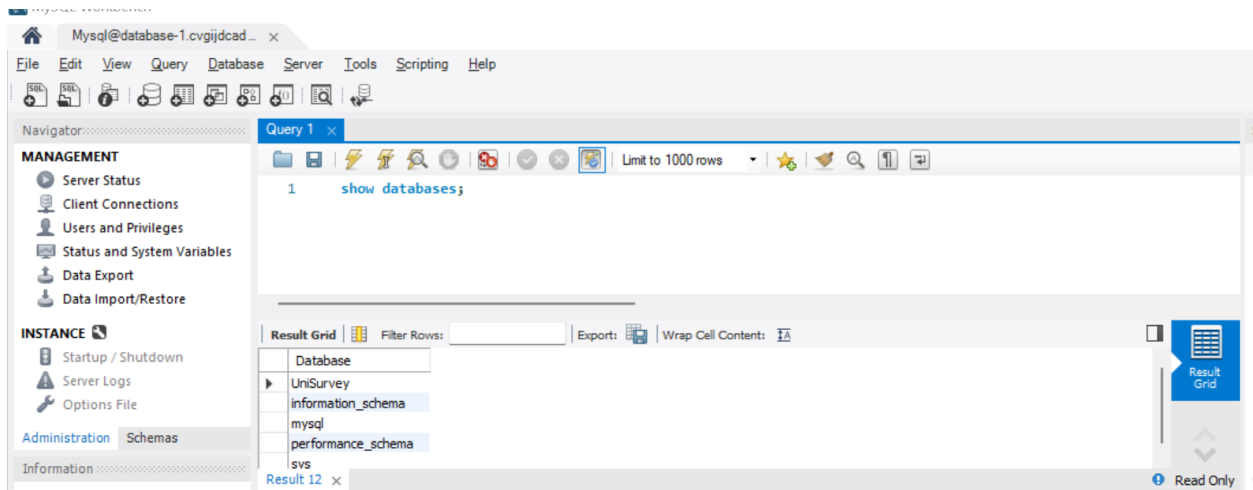
Database with name 'database-1' created and connection to it has been established to access the database.

The screenshot displays the Amazon RDS console interface. On the left is a navigation sidebar with options like Dashboard, Databases, Query Editor, Performance insights, Snapshots, Exports in Amazon S3, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, and Custom engine versions. The main content area is titled 'Amazon RDS' and shows the configuration for 'database-1'. It includes a 'Summary' section with details like DB identifier, CPU usage (4.67%), Status (Available), Class (db.t2.micro), Role, Current activity (32 Connections), Engine (MySQL Community), and Region & AZ (us-east-1c). Below this is a 'Connectivity & security' section with tabs for Connectivity & security, Monitoring, Logs & events, Configuration, Maintenance & backups, and Tags. The 'Connectivity & security' tab is active, showing details for Endpoint & port, Networking, and Security.

Summary			
DB identifier	CPU	Status	Class
database-1	4.67%	Available	db.t2.micro
Role	Current activity	Engine	Region & AZ
Instance	32 Connections	MySQL Community	us-east-1c

Connectivity & security		
Endpoint & port	Networking	Security
Endpoint	Availability Zone	VPC security groups
database-1.cvrijdcadaoe.us-east-1.rds.amazonaws.com	us-east-1c	database-1 (sg-0a2e3506715c04505)
	VPC	Active

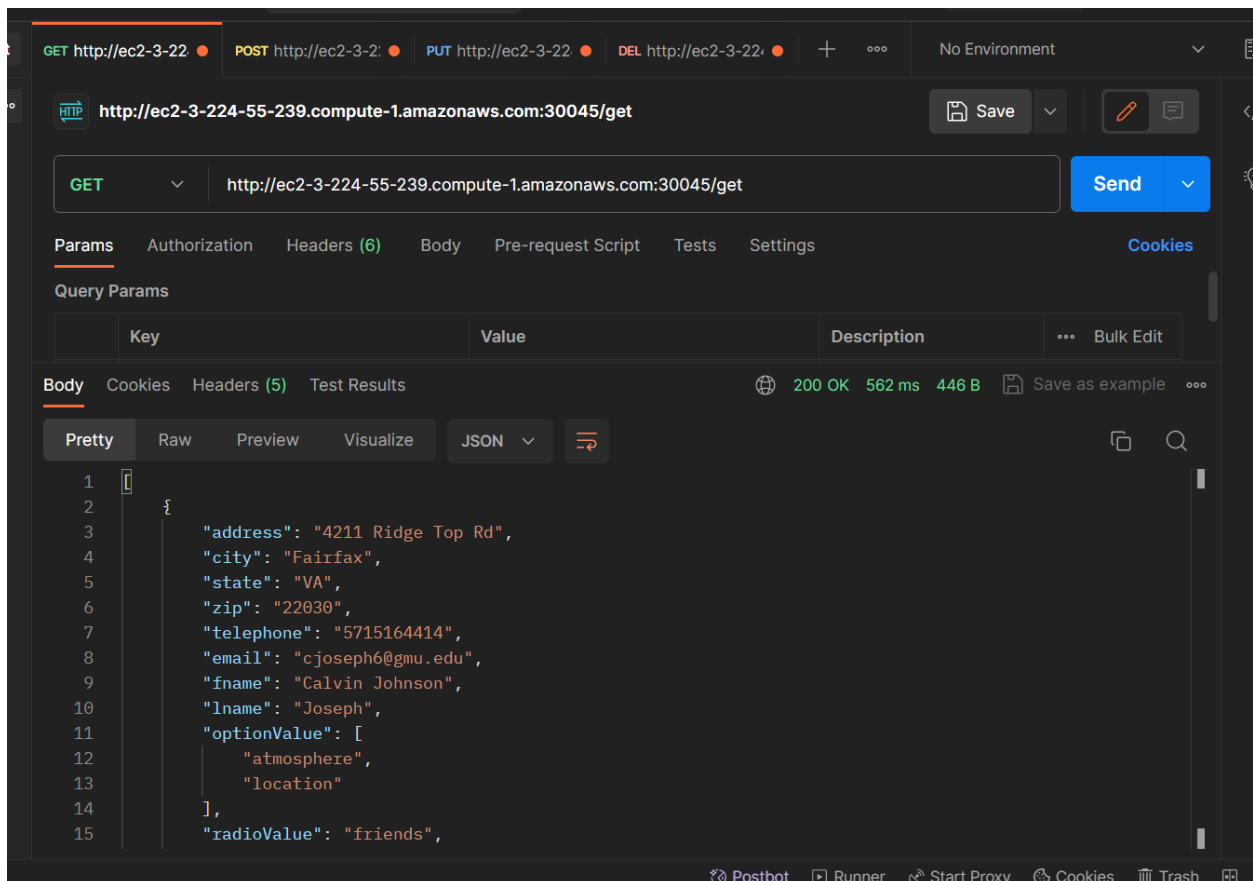
We have named our database 'UniSurvey' , which has a table called 'survey' which stores the survey data that is entered in the survey page.



RESTful Web Service:

We used spring boot for our RESTful Web Service. The web service has all CRUD operations which are verified using 'Postman'. The project can be found in the git repository mentioned above. The project produces a jar file.

Get Request:



Post Request:

We have implemented our POST request using the submit button in our Restful Web Service.

Home

University Survey

First Name *

Last Name *

Street Address *

City *

State *

ZIP *

Put Request:

HTTP <http://ec2-3-224-55-239.compute-1.amazonaws.com:30045/update/3?Fname=Nikhil&Lname=Kumar> Save

PUT <http://ec2-3-224-55-239.compute-1.amazonaws.com:30045/update/3?Fname=Nikhil&Lname=Kumar> Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	Fname	Nikhil			
<input checked="" type="checkbox"/>	Lname	Kumar			
	Key	Value	Description		

Body Cookies Headers (4) Test Results

200 OK 575 ms 123 B Save as example

Pretty Raw Preview Visualize Text

1

Delete Request:

The screenshot shows a REST client interface with a DELETE request to `http://ec2-3-224-55-239.compute-1.amazonaws.com:30045/3`. The interface includes tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, and Settings. The Params tab is active, showing a table with columns Key, Value, and Description. The Body tab is also visible, showing a status of 200 OK, 102 ms, and 123 B.

Key	Value	Description
Key	Value	Description

Containerization:

We containerized our application using docker and pushed it into our docker repository.

The screenshot shows the Docker Hub repository page for `calvinjohnson747/hw3-spring-boot`. The page includes a description, Docker commands, a list of tags, and information about automated builds.

Description
This repository does not have a description
Last pushed: an hour ago

Docker commands
To push a new tag to this repository:
`docker push calvinjohnson747/hw3-spring-boot:tagname`







Tags
This repository contains 8 tag(s).

Tag	OS	Type	Pulled	Pushed
20231128-045108	linux/amd64	Image	an hour ago	an hour ago
20231128-043809	linux/amd64	Image	an hour ago	an hour ago
20231117-045345	linux/amd64	Image	11 days ago	11 days ago
20231117-040038	linux/amd64	Image	11 days ago	11 days ago
20231117-035721	linux/amd64	Image	---	11 days ago

Automated Builds
Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.
Available with Pro, Team and Business subscriptions. [Read more about automated builds](#).
[Upgrade](#)

Kubernetes Cluster:








We updated our deployment.yaml to pull the docker image from the docker repository into the cluster.














 Services [Alt+S]      N. Virginia


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-boot-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: spring-boot-app
  template:
    metadata:
      labels:
        app: spring-boot-app
    spec:
      containers:
        - name: spring-boot-container
          image: calvinjohnson747/hw3-spring-boot:v2
          ports:
            - containerPort: 8080
-- INSERT --
```


i-0b76bb93088f61d9b (Clusterv1.4)




PublicIPs: 3.224.55.239 PrivateIPs: 172.31.91.134

 hw3-cluster Only User Namespaces      



Cluster  Workloads  CronJobs 0  DaemonSets 0  Deployments 2  Jobs 0  StatefulSets 0  Pods 4  Apps  Service Discovery  Storage  Policy  More Resources 
















Cluster Tools 

Endpoints:  Annotations: [Show 2 annotations](#)

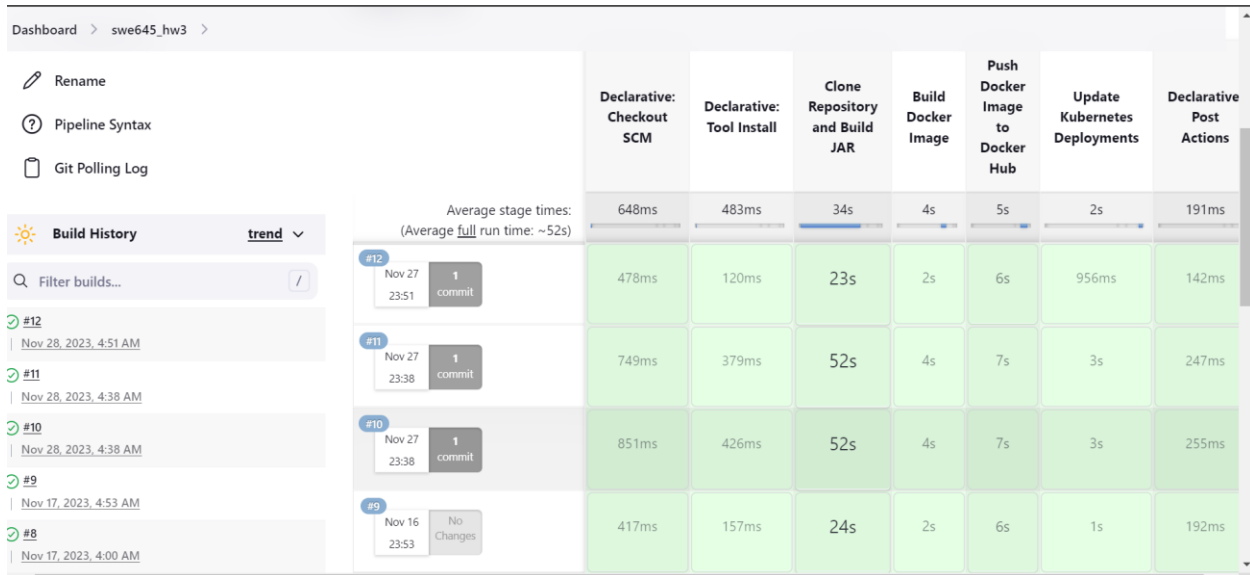
Pods by State  Scale  3 

Pods Services Ingresses Conditions Recent Events Related Resources

 Download YAML  Delete

 State	 Name	 Image	 Ready	 Restarts	 IP	 Node	 Age	
 Running	spring-boot-deployment-787797cb56-cn5pz	calvinjohnson747/hw3-spring-boot:20231128-045108	1/1	1 (18m ago)	10.42.207.95	ip-172-31-91-134	43 mins	
 Running	spring-boot-deployment-787797cb56-f96hj	calvinjohnson747/hw3-spring-boot:20231128-045108	1/1	1 (18m ago)	10.42.207.119	ip-172-31-91-134	43 mins	
 Running	spring-boot-deployment-787797cb56-q5htg	calvinjohnson747/hw3-spring-boot:20231128-045108	1/1	1 (18m ago)	10.42.207.110	ip-172-31-91-134	43 mins	

Jenkins Pipeline



Conclusion:

The application has been successfully hosted in the rancher Kubernetes cluster along with a Jenkins pipeline. All the CRUD operations function as expected, have been shown in Postman and the endpoint produces the desired RESTful application.