

SWE 645
Component-Based Software Development
Prof. Vinod Dubey

Assignment 3

Team Members:

Venkat Ganta : G01321889
Maitreyi Pitale : G01326362
Gowtham Reddy Palnati : G01321927
Deepika Tinnavali : G013343932

Documentation :

Technologies used for implementation:

1. Angular Cli
2. JPA, Spring Boot Framework
3. Amazon RDS
4. GCP- Google Cloud Platform Kubernetes

IDE's used for implementation:

1. Visual Studio Code - Angular - For the frontend part
2. Windows Powershell - Docker part
3. Eclipse - Java Developers Version - For the backend part
4. My SQL workbench - Database - For testing before creating instance in Amazon RDS

Instructions:

Angular

1. To install angular in all of your systems, use "npm install -g @angular/cli."
2. Open the "front-end" folder in the editor, ideally Visual Studio Code, after extracting the zip file supplied.
3. To execute the project at localhost:4200, enter the command "ng serve." Following that, navigate to "http://localhost:4200/".
4. The user may access the application's homepage, which features two links. "Survey form" to view the form and fill out the survey, and "View list of all surveys completed to date" to see the survey forms that have been completed.
5. The user may access the survey form, fill it out according to the fields listed, and submit it.
6. Following the submission of the form, you will be able to examine the GIDs of the users who have filled the form and then you can view the form responses that have been recorded by clicking on the respective GID.

Jpa

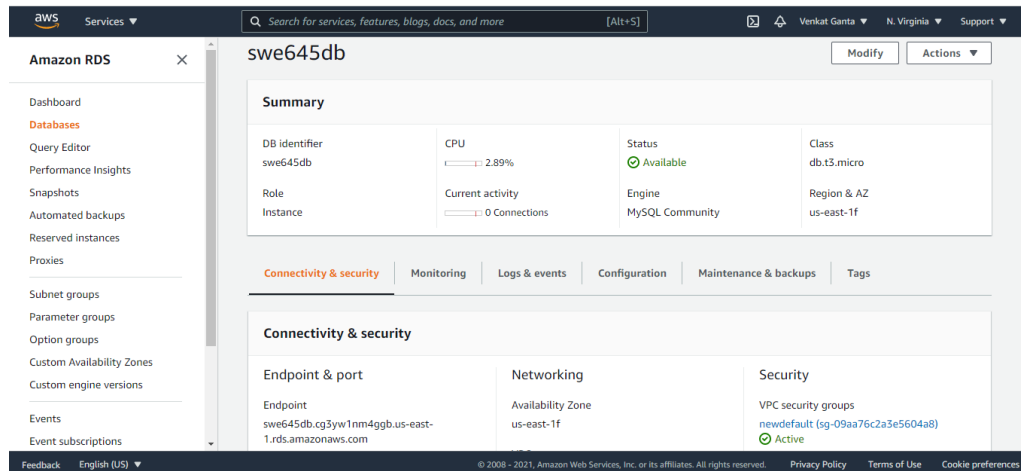
1. To add the JDK 1.8 path to the SDK path, open the JPA application in the Eclipse Enterprise free edition and go to project structures. (Because we only worked on JDK 1.8, the project only takes that version.)
2. JDK may be downloaded from the Oracle website at ["https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html."](https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html)
3. The user will be able to launch the project after adding the SDK to the project's folder. Start the StudentSurveyApp project.
4. Then, navigate to ["http://localhost:8080/api/students"](http://localhost:8080/api/students) to examine the data that has been added to the database.
5. The StudentSurveyApp will have application.properties file where we specified the Amazon RDS endpoint for the database connection
6. As this is a springboot application it uses the Tomcat server within it and make connection to the database with specified endpoint. URL: swe645db.cg3yw1nm4ggb.us-east-1.rds.amazonaws.com

The steps used to complete the project are outlined below.

Amazon RDS –

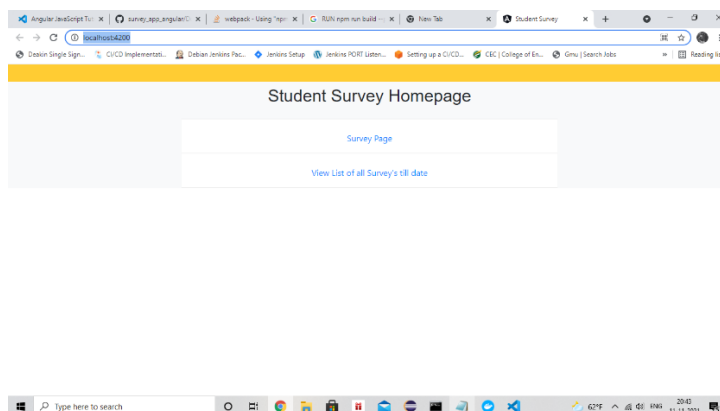
Amazon RDS Endpoint: swe645db.cg3yw1nm4ggb.us-east-1.rds.amazonaws.com

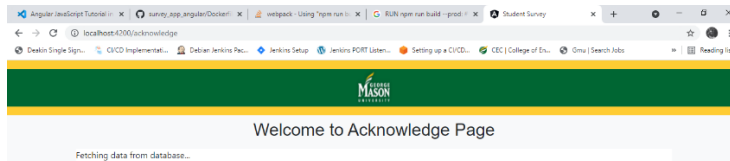
1. Log in to the AWS console using your credentials and pick Amazon RDS in the console.
2. Following that, create a new database by selecting mysql and providing the assign4 user name and password that will be required when connecting to the database.
3. Enable public access and choose the security group that will give access to all inbound and outgoing rules.
4. Don't forget to add a name to the database once you've finished configuring it. (The instance name is not the name, and it will not be displayed if no name is specified for the database.)
5. The database instance takes some time to construct, but once completed, you can see that the database is ready.
6. Check to see if the database is operational.



Front end of an Angular application –

1. We used Angular's reusable components to develop four components: the homepage component, the read-survey component, the survey component, and the acknowledgment component.
2. The homepage component includes links to the survey form and a list of all completed surveys.
3. The survey component includes a form that the user will fill out, as well as form validation for the replies.
4. After submitting the form, the user will be directed to the acknowledgement page (component), where he'd be able to view the ids of the forms that have been stored in the database.





Back end using Eclipse IDE for Java Developers -

1. In Eclipse IDE for Java Developers, start a new project (you can use IntelliJ too.).
2. We insert the database connection credentials, jdbc driver connection information in lieu of the driver, and the endpoint url in the database base url section in the application properties file.
3. Following that, we must enter the database's assign4 username and password, which we specified when we created the Amazon RDS DB.
4. Endpoint URL: "jdbc:mysql://swe645db.cg3yw1nm4ggb.us-east-1.rds.amazonaws.com:3306/swe645?useSSL=false"
5. Now we must construct an entity class with an @entity tag that specifies how our attributes will appear in the database and we make sure that the column names here and the column name in the database are similar.
6. @column(name = "table name") is used to provide the column names. By right-clicking on the file, we can also immediately import the getters and setters from eclipse.

We next build a controller class StudentController in which we write a RESTful webservice using the @restcontroller and @requestmapping("api") tags, which will be the url that appears after the default url.

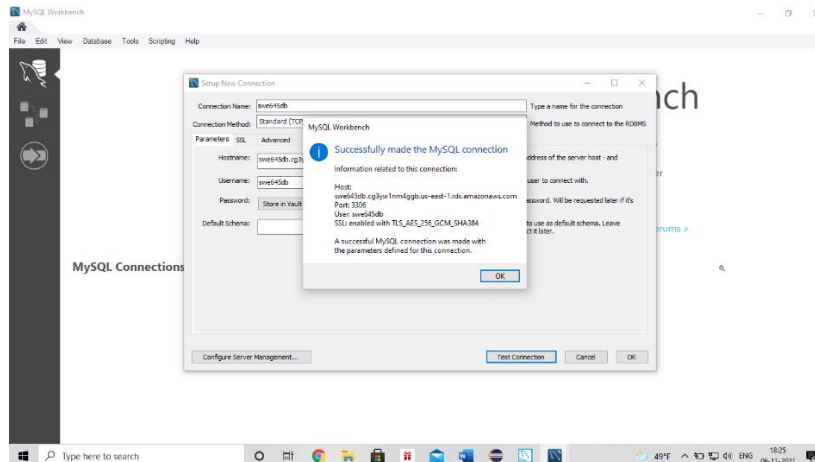
1. We use the service which we wrote to save the form data and also a get function to get the form data by using the get function written in the student service. We use the @Autowire tag for the repository to call the student save function and return the service using the functions.

2. “service.saveStudent(student)” and “return service.getStudentBySID(student_id)” which will trigger the save function to save the student form details and “get method” to get the form details when required.
3. Functions in services will trigger the repository methods which are provided by JpaRepository. JpaRepository will automatically write the database queries respective to the database to push and pull the data from the database
4. We need to mention @Crossorigin because we have to run to servers in the same local machine. In order to allow that we need to have the @Crossorigin tag enabled in the webservice Or else this will result in a CORS error which does not allow our frontend to make the necessary api calls.

```

eclipse-workspace - JPA-mysql/src/main/java/com/springboot/StudentSurveyApp.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
StudentSurveyApp [Java Application]
Spring Boot (v2.4.4)
2021-11-12 13:31:12.510 INFO 8712 --- [main] com.springboot.StudentSurveyApp : Starting StudentSurveyApp using Java 15.0.2 on DESKTOP-TUS083E with PID 8712 (C:\Us
2021-11-12 13:31:12.517 INFO 8712 --- [main] com.springboot.StudentSurveyApp : No active profile set, falling back to default profiles: default
2021-11-12 13:31:13.624 INFO 8712 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2021-11-12 13:31:13.730 INFO 8712 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-11-12 13:31:15.176 INFO 8712 --- [main] org.apache.catalina.core.StandardEngine : Starting service [Tomcat]
2021-11-12 13:31:15.194 INFO 8712 --- [main] o.a.c.c.c.[Tomcat].[/] : Starting Servlet engine: [Apache Tomcat/9.0.44]
2021-11-12 13:31:15.381 INFO 8712 --- [main] w.s.c.ServletWebServerApplicationContext : Initializing Spring embedded WebApplicationContext
2021-11-12 13:31:15.387 INFO 8712 --- [main] org.hibernate.jpa.internal.util.LogHelper : Root WebApplicationContext: initialization completed in 2703 ms
2021-11-12 13:31:15.807 INFO 8712 --- [main] org.hibernate.Version : HH0000412: Hibernate ORM core version 5.4.29.Final
2021-11-12 13:31:15.900 INFO 8712 --- [main] org.hibernate.Version : HCC000001: Hibernate Commons Annotations (5.1.2.Final)
2021-11-12 13:31:16.194 INFO 8712 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2021-11-12 13:31:16.392 INFO 8712 --- [main] org.hibernate.dialect.Dialect : HH0000400: Using dialect: org.hibernate.dialect.MySQL5Dialect
2021-11-12 13:31:17.744 INFO 8712 --- [main] o.h.e.t.j.p.i.1.PlatformInitiator : HH0000400: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta
2021-11-12 13:31:19.621 INFO 8712 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2021-11-12 13:31:19.654 INFO 8712 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be p
2021-11-12 13:31:20.731 WARN 8712 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2021-11-12 13:31:21.023 INFO 8712 --- [nio-8080-exec-1] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-11-12 13:31:22.081 INFO 8712 --- [main] com.springboot.StudentSurveyApp : Started StudentSurveyApp in 10.337 seconds (JVM running for 11.185)
2021-11-12 13:32:17.920 INFO 8712 --- [nio-8080-exec-1] o.a.c.c.c.[Tomcat].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2021-11-12 13:32:17.920 INFO 8712 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2021-11-12 13:32:17.922 INFO 8712 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 2 ms
  
```

My SQL workbench and Amazon RDS Connection:

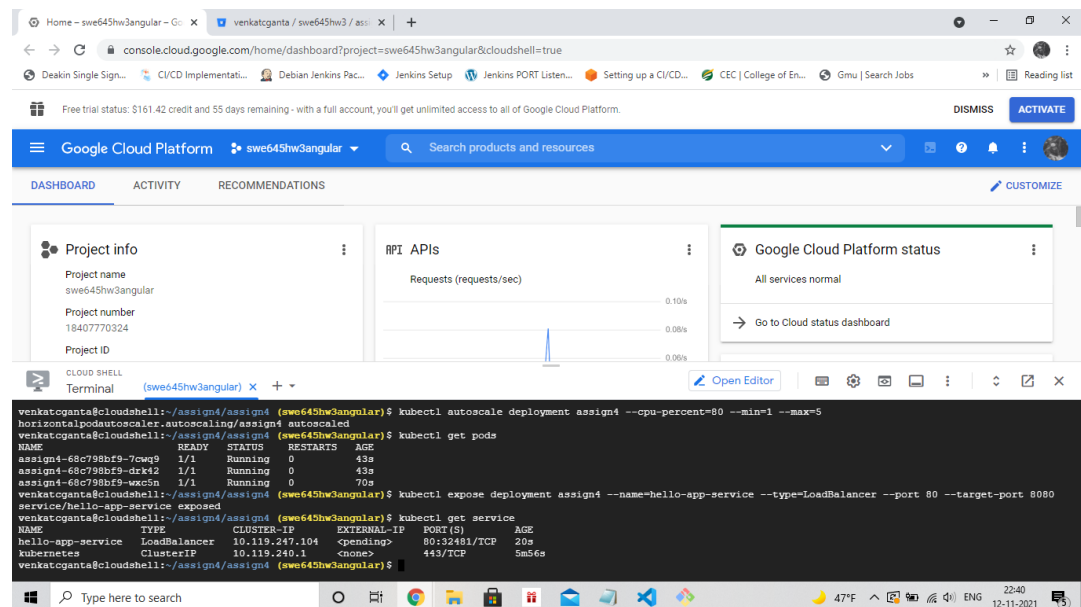


Creating Kubernetes Cluster using Google Cloud Platform:

- 1) **Creating GKE cluster and deploying Angular App with Nginx web server**
Google Kubernetes lets you run containerized applications on Google cloud.

Install gcloud CLI and Configure Dockerize the Project

- 1) To achieve this we created a docker image and container with the help of Dockerfile and command prompt commands for our windows system.
- 2) Make sure you have docker installed on your machine. We used Docker Desktop for Windows. You will also need to create an account on <https://hub.docker.com/> in order to push our docker image and make it available globally.
- 3) In Eclipse, create a file called Dockerfile, which has no extension. Docker requires the files to be called 'Dockerfile'. We specify the base image and the "package.json" file that is to be copied in the specific location of the docker container.
- 4) Build the dynamic web project on eclipse and generate the war file and place it in the same folder as the Dockerfile in your local system to get your war file read by the docker container.
- 5) In the DockerFile, use the FROM command to get the initial base image for the build. We want to run the war file in Tomcat so we should use the nginx image.
- 6) Next, we need to drop the war file in the webapps folder, for that we use this statement in Dockerfile.
- 7) On the command line, use this command: 'docker build --tag assign4 .' (You can use whatever name and tag you want.)
- 8) Verify that the image is properly working by running 'docker run -it -p 8081:8080 assign4:latest' and open a browser at http://localhost:8081/Student_survey/Survey.html to see the containerized angular web application.



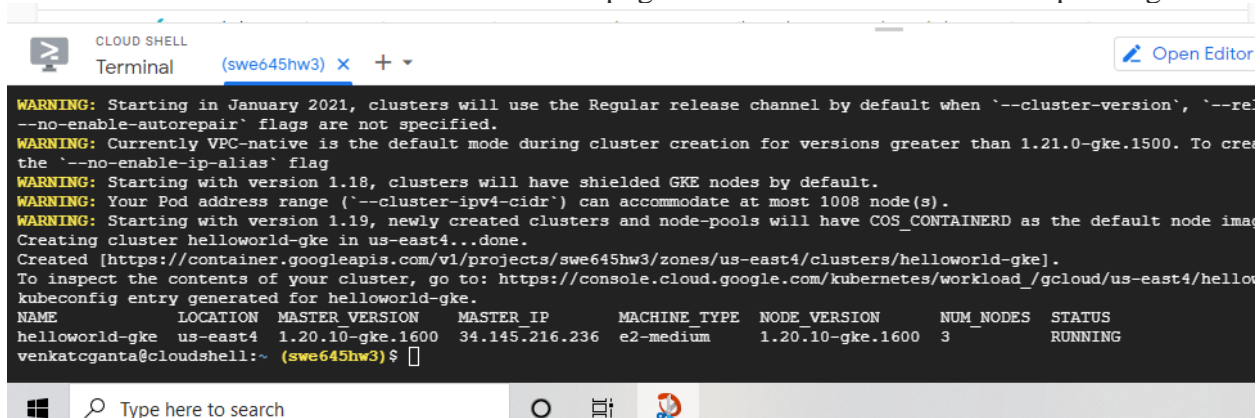
2. Pushing Docker Image To Container Registry

1. You need to push your image to DockerHub in order to access it globally by a few commands in your command prompt.
2. Firstly, you need to create an account in DockerHub with a username and password of your choice.

3. Secondly, login to docker using ‘docker login -u ’ enter the password to push the image to the docker hub from your command line.
4. Use “docker push ” to push your image on to the docker hub.
5. Verify that your image is on Docker Hub. Your image is accessible from the internet by logging into docker hub.

3. Creating GKE Cluster ,Configure Kubectl With GKE Cluster

1. I used Google Cloud Platform (GCP) to create a cluster and run the Kubernetes platform Google Kubernetes Engine to deploy the containerized web application.
2. Firstly enable packages Artifact Registry and Kubernetes Engine to get the desired outcome.
3. Activate the Google Cloud Shell and enter the commands locally to get each functionality.
4. Install the kubectl package using the command “gcloud components install kubectl”.
5. Create a repository by exporting the Project_ID and configuring it by running commands as specified in the commands.docx file.
6. Set your project ID for the gcloud command line tool.
7. Create a repository for the GKE cluster using the command “gcloud artifacts repositories create gvm-d-repo --repository-format=docker --location=us-east4 --description=”Docker repository”
8. Download the “Survey” (web application) source code and Dockerfile by running the following commands: Here we give the git repository address where we push all the Dockerfile and Student_survey.war files of our project. Command: git clone <https://bitbucket.org/venkatcganta/swe645-hw2/src/assign4>
9. Build and tag the docker image for the project using the following command docker build -t us-east4-docker.pkg.dev/swe645hw3/swe645hw3repo/assign4:v1



The screenshot shows a Google Cloud Shell terminal window with the title 'Terminal (swe645hw3)'. It displays the output of the 'gcloud clusters create' command. The output includes several warnings about the release channel, VPC-native mode, and node image. It then shows the successful creation of the 'helloworld-gke' cluster in the 'us-east4' region. A table lists the cluster details:

NAME	LOCATION	MASTER_VERSION	MASTER_IP	MACHINE_TYPE	NODE_VERSION	NUM_NODES	STATUS
helloworld-gke	us-east4	1.20.10-gke.1600	34.145.216.236	e2-medium	1.20.10-gke.1600	3	RUNNING

The prompt shows the user 'venkatcganta@cloudshell:~' in the 'swe645hw3' project.

4. Deploy Kubernetes Objects on GKE Cluster & Access the WebApp from the browser

1. Running the container locally using this command “docker run --rm -p 8080:8080 useast4-docker.pkg.dev/swe645hw3/swe645hw3repo/assign4:v1” 11)

2. Push the Docker image to the Artifact registry to save the docker image and make it available all the time.
3. Create a Kubernetes Deployment for your Docker image. 5) Set the baseline number of Deployment replicas to 3.
4. Create a HorizontalPodAutoscaler resource for your Deployment.
5. Now the 3 pods are allocated to your nodes. To see the Pods created, run the following command: `kubectl get pods`. Output Screenshot for nodes and pods running status: Step – Exposing the web application to the internet: Use the “`kubectl expose`” command to generate a Kubernetes Service for assign4 deployment. Run this command to use the `kubectl service “kubectl get service”` you will get the output as follows: You use the External IP which you can access the web application from the bitbucket repository we cloned and the docker image we created and containerized in a Kubernetes engine. This makes our web application available to the Kubernetes engine with the IP of External IP and the endpoint similar to the previous one.
6. For deploying projects we need to create a new app and connect to it using git repository.
7. The projects need to be pushed to the github so that the deployment can be triggered from the yaml files.
8. We need to define a workflow so that the github will build and push the image to github.
9. This can be done by setting the workflow in github.

URL's:

- 1) My Bit Bucket repository Link: <https://bitbucket.org/venkatcganta/swe645hw3/src/assign4/>
- 2) Amazon RDS Endpoint: swe645db.cg3yw1nm4ggb.us-east-1.rds.amazonaws.com
- 3) Kubernetes URL for Angular Application: <http://34.85.225.96/api/students>

References:

<https://medium.com/bb-tutorials-and-thoughts/docker-image-creation-and-management-9d91e4c277b1>
<https://cloud.google.com/kubernetes-engine/docs/tutorials/hello-app#console>
<https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-creds>
<https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html>
<https://eksctl.io/usage/minimum-iam-policies/>
https://argoproj.github.io/argo-cd/getting_started/