

SWE 645 Assignment 3

Team Members:

AKSHITA SRIKANTH - G01393896
ANUSHA JAGADISH - G01394233
AKASH PONNAM - G01378189
SHISHWAN GANDE - G01383809

CREATING AWS RDS CONNECTION

- Sign in to the AWS console and go to the RDS service page.
- Select the option to create a new Database.
- Choose the Standard Create method to build the database.
- Select MySQL as the database engine.

The screenshot shows the 'Create database' wizard on the AWS RDS service page. In the 'Choose a database creation method' section, the 'Standard create' option is selected. Below it, the 'Engine options' section shows various database engines: Aurora (MySQL Compatible), Aurora (PostgreSQL Compatible), MySQL (selected), MariaDB, PostgreSQL, and Oracle. The MySQL icon features a globe and a flame.

- Pick the db.t2.micro tier for the database instance computing capacity and storage allocation.
- Default storage configurations.
- Click the free tier as templates.

The screenshot shows the 'Create database' wizard continuing through the steps. In the 'Templates' section, the 'Free tier' template is selected. This section includes a note about using the RDS Free Tier for development and testing. Below it is the 'Availability and durability' section, which is currently collapsed.

The screenshot shows the 'Launch DB Instance' wizard in the AWS RDS console. The configuration steps are as follows:

- DB subnet group:** Set to 'default-vpc-08418619ff1bf1c6f' (6 Subnets, 6 Availability Zones).
- Public access:** Set to 'Yes'.
- VPC security group (firewall):** Set to 'Choose existing' (Choose existing VPC security groups) and selected 'default'.
- Existing VPC security groups:** Shows 'Choose one or more options' and 'default'.
- Availability Zone:** Set to 'No preference'.

- Name the database, create a username and password to access it.

The screenshot shows the 'Launch DB Instance' wizard with the following configurations:

- DB instance identifier:** Set to 'survey'.
- Credentials Settings:**
 - Master username:** Set to 'root'.
 - Manage master credentials in AWS Secrets Manager:** Unchecked.
 - If you manage the master user credentials in Secrets Manager, some RDS features aren't supported.** A note states: 'Amazon RDS can generate a password for you, or you can specify your own password.'
 - Auto generate a password:** Unchecked.
 - Master password:** Set to '*****'.
 - Confirm master password:** Set to '*****'.
- MySQL Feature Summary:**
 - MySQL is described as the most popular open source database in the world.
 - MySQL on RDS offers rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.
 - Supports database size up to 64 TiB.
 - Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
 - Supports automated backup and point-in-time recovery.
 - Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.

- Database will be created
- Database Endpoint : survey.cquygbzzx1tg.us-east-1.rds.amazonaws.com

Screenshot of the AWS RDS console showing the details of a database named "survey".

Summary

DB identifier	CPU	Status	Class
survey	-	Starting	db.t2.micro
Role	Current activity	Engine	Region & AZ
Instance		MySQL Community	us-east-1c

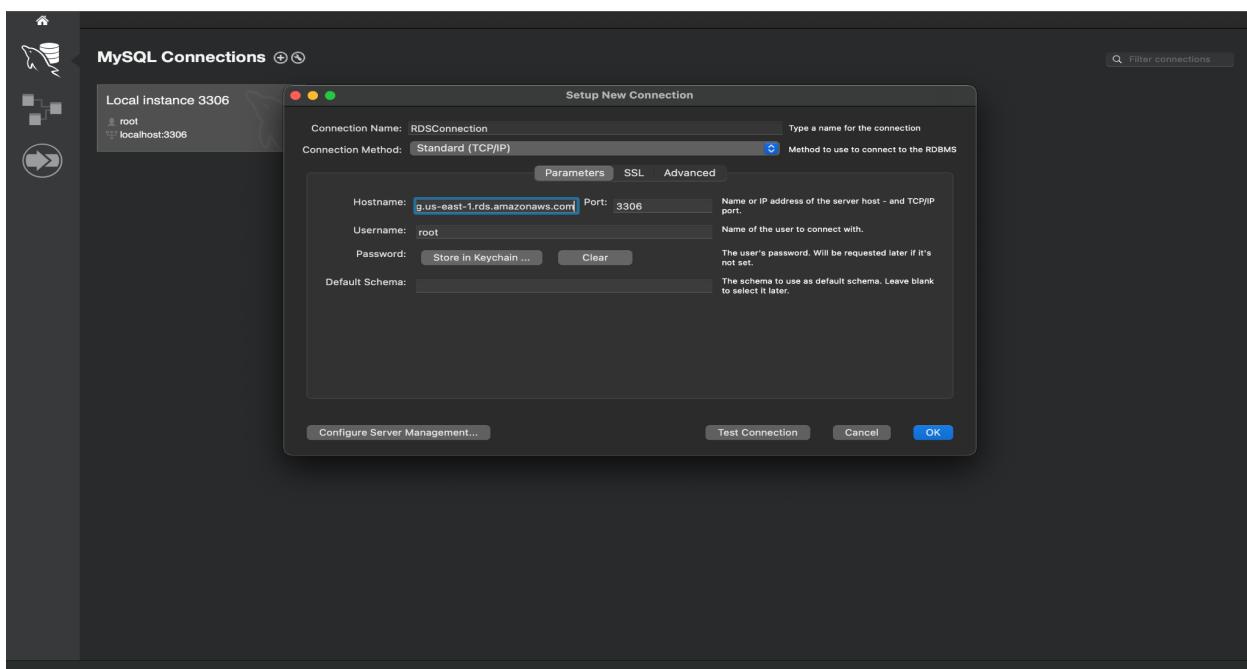
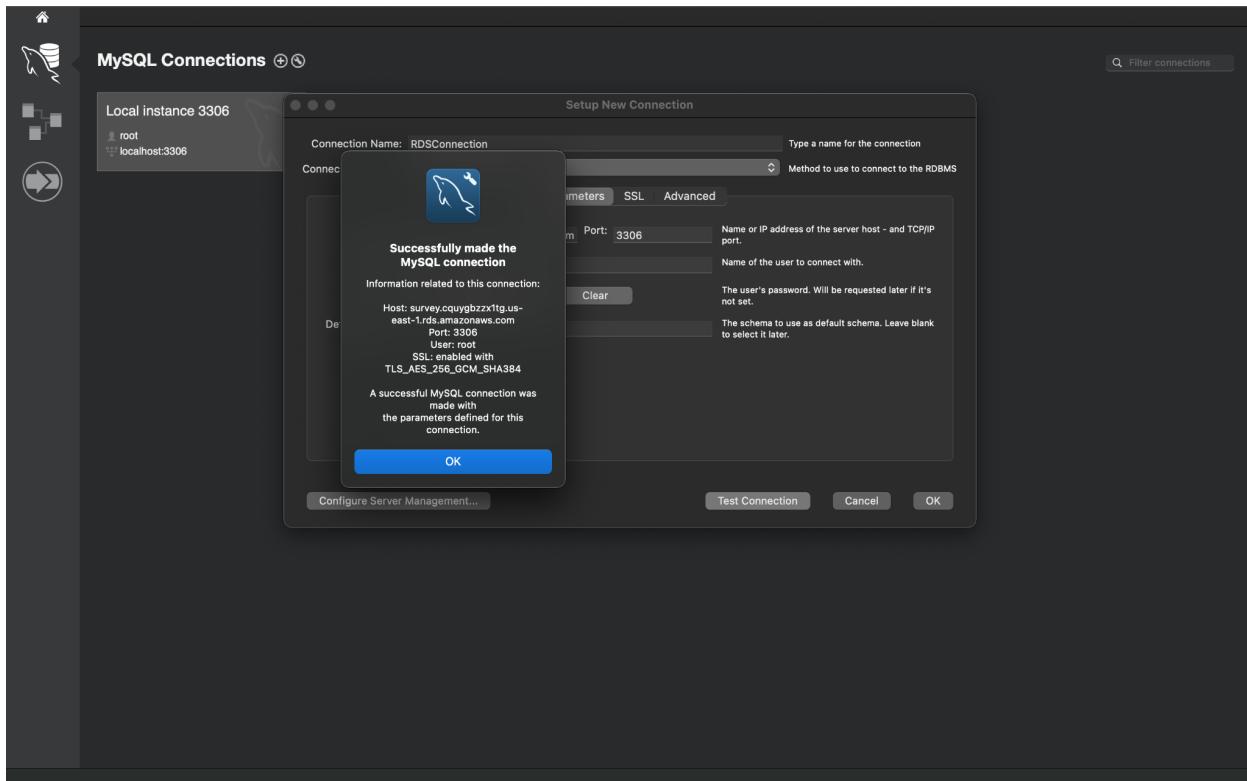
Connectivity & security

Endpoint & port	Networking	Security
Endpoint survey.cquygbzzx1tg.us-east-1.rds.amazonaws.com	Availability Zone us-east-1c VPC vpc-08418619ff1bf1c6f	VPC security groups default (sg-095bc18b4205f024f) Active
Port 3306	Subnet group default-vpc-08418619ff1bf1c6f Subnets subnet-0a2f8da17608780d1 subnet-09047ccb05246fd64 subnet-047f75181ca07a15a subnet-0677efb3181711d00	Publicly accessible Yes Certificate authority Info rds-ca-2019 Certificate authority date August 22, 2024, 13:08 (UTC-04:00) DB instance certificate expiration

CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

CONNECTING AWS MYSQL IN MYSQL WORKBENCH

- Setup a New Connection by clicking + icon.
- Give a name in the connection name tab.
- Give the AWS MySQL endpoint in the Hostname tab.
- Give the username and password which was given while creating the database in AWS.
- Click on Test connection and apply OK.



- Connection will be established to MySQL.
- Create a Database with name surveydb with the command
CREATE DATABASE surveydb;

The screenshot shows the MySQL Workbench interface. The left sidebar has sections for MANAGEMENT (Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, Data Import/Restore), INSTANCE (Startup / Shutdown, Server Logs, Options File), and PERFORMANCE (Dashboard, Performance Reports, Performance Schema Setup). The main area has tabs for Administration and Schemas. The Schemas tab is active, showing a query editor with the following SQL code:

```

1 create database surveydb;
2
3 show databases;

```

The results grid shows the following databases:

Database
information_schema
mysql
performance_schema
surveydb
sys

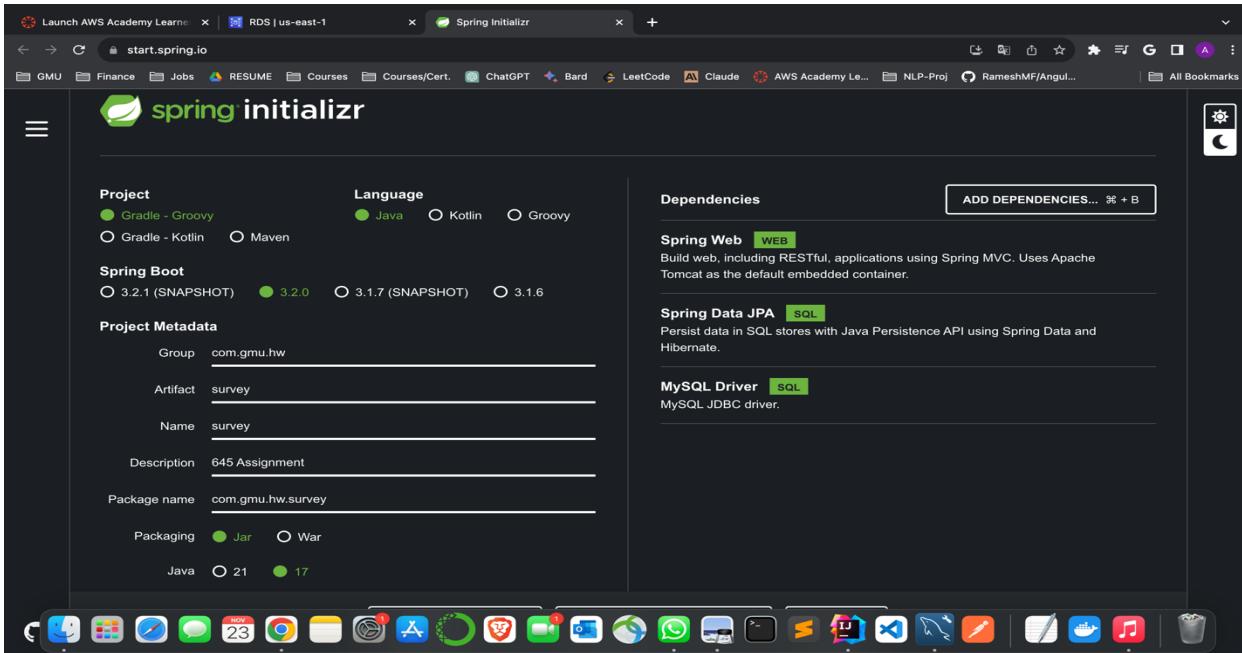
The Action Output section shows the execution history:

Action	Time	Response	Duration / Fetch Time
show databases	15:31:21	5 row(s) returned	0.01 sec / 0.000009...
drop database surveydb	15:31:42	1 row(s) affected	0.046 sec
show databases	15:31:45	4 row(s) returned	0.0085 sec / 0.00001...
create database surveydb	15:32:03	1 row(s) affected	0.021 sec
show databases	15:32:13	5 row(s) returned	0.012 sec / 0.00001...

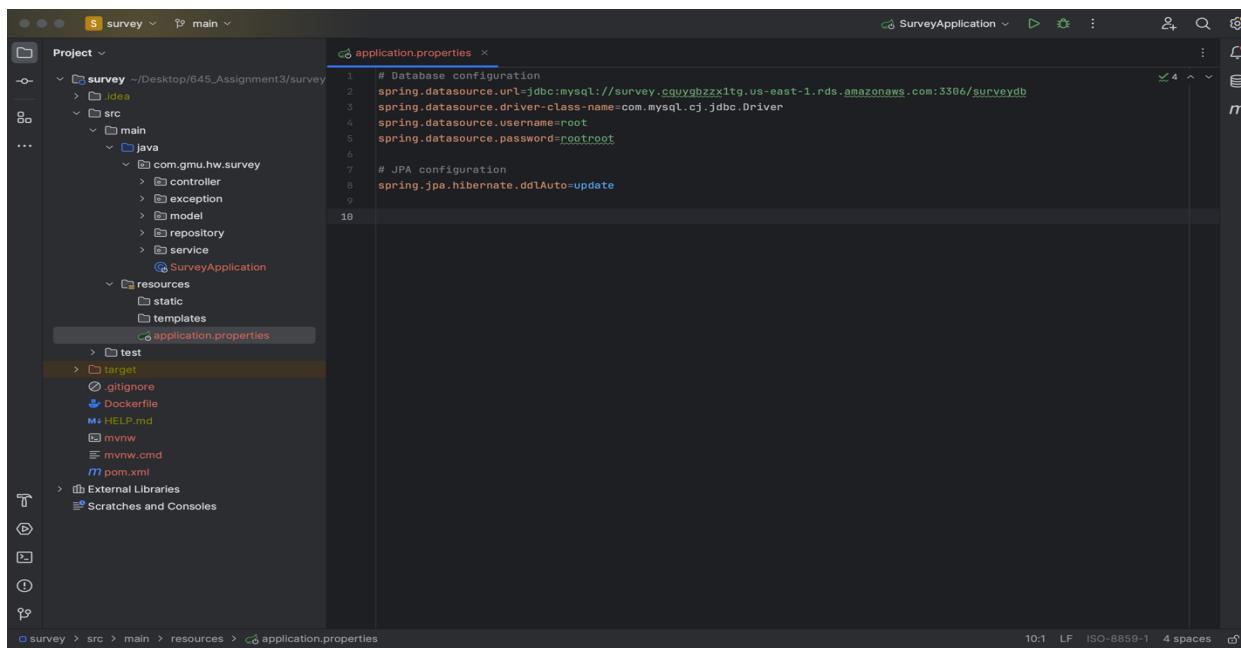
At the bottom, it says "Query Completed".

CREATING SPRING BOOT APPLICATION

- Navigate to <https://start.spring.io/> and download spring project with the below specs and click generate.



- Navigate to the application.properties file and add the below details.



- Create SurveyModel which contains all the specifications of the table

The screenshot shows the IntelliJ IDEA interface with the SurveyModel.java file open in the editor. The code defines a JPA entity named SurveyModel with fields for id, firstName, lastName, streetAddress, city, and state.

```
1 package com.gmu.hw.survey.model;
2
3 import jakarta.persistence.*;
4
5 import java.util.Date;
6
7
8 @Entity
9 @Table(name = "surveys")
10 public class SurveyModel {
11
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     private Long id;
15
16     @Column(name = "first_name", nullable = false)
17     private String firstName;
18
19     @Column(name = "last_name", nullable = false)
20     private String lastName;
21
22     @Column(name = "street_address", nullable = false)
23     private String streetAddress;
24
25     @Column(name = "city", nullable = false)
26     private String city;
27
28     @Column(name = "state", nullable = false)
29     private String state;
30 }
```

The Project tool window on the left shows the project structure, including src, main, test, and various Java packages like com.gmu.hw.survey.controller, com.gmu.hw.survey.exception, and com.gmu.hw.survey.model.

- Create JpaRepository

The screenshot shows the IntelliJ IDEA interface with the SurveyRepository.java file open in the editor. The code defines a JPA repository interface SurveyRepository that extends JpaRepository<SurveyModel, Long>.

```
1 package com.gmu.hw.survey.repository;
2
3 import com.gmu.hw.survey.model.SurveyModel;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7
8 @Repository
9 public interface SurveyRepository extends JpaRepository<SurveyModel, Long> {
10 }
```

The Project tool window on the left shows the project structure, including src, main, test, and various Java packages like com.gmu.hw.survey.controller, com.gmu.hw.survey.exception, and com.gmu.hw.survey.repository.

- Create Survey which extends the JpaRepository

The screenshot shows the IntelliJ IDEA interface with the SurveyService.java file open in the editor. The code defines a SurveyService class with methods for getting all surveys, saving a survey, getting a survey by ID, and deleting a survey. The SurveyRepository dependency is injected via Autowiring.

```
1 package com.gmu.hw.survey.service;
2
3 import ...;
4
5 @Service
6 public class SurveyService {
7
8     @Autowired
9     private SurveyRepository surveyRepository;
10
11     public List<SurveyModel> getAllSurveys() { return surveyRepository.findAll(); }
12
13     public SurveyModel saveSurvey(SurveyModel survey) { return surveyRepository.save(survey); }
14
15     public Optional<SurveyModel> getSurveyById(Long id) { return surveyRepository.findById(id); }
16
17     public void delete(SurveyModel student) { surveyRepository.delete(student); }
18 }
```

- Create an exception class to handle the exceptions

The screenshot shows the IntelliJ IDEA interface with the SurveyException.java file open in the editor. The code defines a SurveyException class that extends RuntimeException and includes annotations for ResponseStatus.

```
1 package com.gmu.hw.survey.exception;
2
3 import org.springframework.http.HttpStatus;
4 import org.springframework.web.bind.annotation.ResponseStatus;
5
6 @ResponseStatus(value = HttpStatus.NOT_FOUND)
7 public class SurveyException extends RuntimeException {
8
9     private static final long serialVersionUID = 1L;
10
11     public SurveyException(String message) { super(message); }
12 }
```

- Create a controller class to handle the REST api calls

The screenshot shows the IntelliJ IDEA interface with the project 'survey' open. The code editor displays the SurveyController.java file, which contains Java code for a Spring Boot application. The code includes imports for various Spring framework classes and annotations like @RestController and @GetMapping. The controller handles requests for '/home', '/all', and '/newStudent'. The project structure on the left shows packages for com.gmu.hw.survey, com.gmu.hw.survey.model, com.gmu.hw.survey.repository, and com.gmu.hw.survey.service. Below the code editor, the terminal window shows the command 'mvnw'.

```
import com.gmu.hw.survey.model.SurveyModel;
import com.gmu.hw.survey.service.SurveyService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

new *
@CrossOrigin(origins = "http://localhost:4200")
@RestController
@RequestMapping("/api/surveys")
public class SurveyController {

    @Autowired
    private SurveyService surveyService;

    new *
    @GetMapping("/home")
    public String home() { return "Campus Survey APP"; }

    new *
    @GetMapping("/all")
    public List<SurveyModel> getAllSurveys() {
        return surveyService.getAllSurveys();
    }

    new *
    @PostMapping("/newStudent")
    public SurveyModel createSurvey(@RequestBody SurveyModel survey) { return surveyService.saveSurvey(survey); }

    new *
    @GetMapping("/student/{id}")
}
```

- Open the Survey Application and run the application

The screenshot shows the IntelliJ IDEA interface with the project 'survey' open. The code editor displays the SurveyApplication.java file, which contains the main method for starting the Spring Boot application. The project structure on the left shows packages for com.gmu.hw.survey, com.gmu.hw.survey.model, com.gmu.hw.survey.repository, and com.gmu.hw.survey.service. Below the code editor, the terminal window shows the output of running 'mvnw'.

```
package com.gmu.hw.survey;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SurveyApplication {

    public static void main(String[] args) {
        SpringApplication.run(SurveyApplication.class, args);
    }
}
```

Terminal Output:

```
2023-11-23T15:36:25.717+05:00 INFO 92267 --- [           main] com.gmu.hw.survey.SurveyApplication      : Starting SurveyApplication using Java 21.0.1 with PID 92267 (v3.1.5)
2023-11-23T15:36:25.719+05:00 INFO 92267 --- [           main] com.gmu.hw.survey.SurveyApplication      : No active profile set, falling back to 1 default profile: "dev"
2023-11-23T15:36:25.988+05:00 INFO 92267 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate  : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2023-11-23T15:36:26.014+05:00 INFO 92267 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate  : Finished Spring Data repository scanning in 23 ms. Found 1 JP
2023-11-23T15:36:26.243+05:00 INFO 92267 --- [           main] o.s.b.embedded.tomcat.TomcatWebServer   : Tomcat initialized with port(s): 8080 (http)
2023-11-23T15:36:26.247+05:00 INFO 92267 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2023-11-23T15:36:26.247+05:00 INFO 92267 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.15]
```

POSTMAN WITHOUT DOCKER

- Creating a new survey using the POST method.

The screenshot shows the Postman application interface. In the top navigation bar, 'Explore' is selected. Below it, 'My Workspace' contains a collection named '645'. A 'New Request' dialog is open, showing a POST method directed at <http://localhost:8080/api/surveys/newStudent>. The 'Body' tab is active, displaying a JSON payload:

```
1 {  
2   ...  
3     "id": 1,  
4     "firstName": "Alice",  
5     "lastName": "Smith",  
6     "streetAddress": "123 Oak St",  
7     "city": "Hometown",  
8     "state": "CA",  
9     "zip": "12345",  
10    "telephoneNumber": "555-123-4567",  
11    "email": "alice.smith@example.com",  
12    "dateOfSurvey": "2023-03-15T08:00:00.000+00:00",  
13    "likedAboutCampus": "location",  
14    "interestSource": "friends",  
15    "likelihoodToRecommend": "Very Likely",  
16    "additionalComments": "The campus's proximity to the beach is amazing. I heard about this university from some friends who spoke highly of it."  
17 }  
18  
19 {  
20   ...  
21     "id": 2,  
22     "firstName": "Charlie",  
23     "lastName": "Davis",  
24     "streetAddress": "456 Maple Ave",  
25 }  
26  
27 }
```

The status bar at the bottom indicates a successful response: Status: 200 OK, Time: 251 ms, Size: 726 B.

- Retrieving all the surveys from the GET method.

The screenshot shows the Postman application interface. In the top navigation bar, 'Explore' is selected. Below it, 'My Workspace' contains a collection named '645'. A 'New Request' dialog is open, showing a GET method directed at <http://localhost:8080/api/surveys/all>. The 'Params' tab is active, showing a table for 'Query Params' with one row: 'Key' (empty) and 'Value' (empty). The 'Body' tab is active, displaying a JSON response:

```
1 {  
2   ...  
3     "id": 1,  
4     "firstName": "Alice",  
5     "lastName": "Smith",  
6     "streetAddress": "123 Oak St",  
7     "city": "Hometown",  
8     "state": "CA",  
9     "zip": "12345",  
10    "telephoneNumber": "555-123-4567",  
11    "email": "alice.smith@example.com",  
12    "dateOfSurvey": "2023-03-15T08:00:00.000+00:00",  
13    "likedAboutCampus": "location",  
14    "interestSource": "friends",  
15    "likelihoodToRecommend": "Very Likely",  
16    "additionalComments": "The campus's proximity to the beach is amazing. I heard about this university from some friends who spoke highly of it."  
17 },  
18 {  
19   ...  
20     "id": 2,  
21     "firstName": "Charlie",  
22     "lastName": "Davis",  
23     "streetAddress": "456 Maple Ave",  
24 }  
25  
26 }
```

The status bar at the bottom indicates a successful response: Status: 200 OK, Time: 182 ms, Size: 1.88 KB.

- Updating a particular survey details using the UPDATE method.

The screenshot shows the Postman application interface. The top navigation bar includes 'Home', 'Workspaces', 'API Network', 'Explore', 'Search Postman', 'Invite', 'Upgrade', and environment dropdowns. The left sidebar, titled 'My Workspace', lists environments: 'Collections' (empty), '642', and '645'. Below these are buttons for 'GET New Request', 'POST New Request', 'PUT New Request' (selected), and 'DEL New Request'. The main workspace shows a 'PUT' request to 'http://localhost:8080/api/surveys/oldStudent/2'. The 'Body' tab is selected, displaying a JSON object representing a survey record. The 'Pretty' tab in the preview section shows the same JSON with line breaks and indentations. The preview also includes a note about additional comments.

```
PUT http://localhost:8080/api/surveys/oldStudent/2

{
  "id": 2,
  "firstName": "Charlie",
  "lastName": "Davis",
  "streetAddress": "456 Maple Ave",
  "city": "Cityville",
  "state": "CA",
  "zip": "54321",
  "telephoneNumber": "555-987-6543",
  "email": "charlie.davis@example.com",
  "dateOfSurvey": "2023-03-20T10:00:00.000+00:00",
  "likedAboutCampus": "Sports",
  "interestSource": "Television",
  "likelihoodToRecommend": "Likely",
  "additionalComments": "The friendly and welcoming atmosphere on campus caught my attention. I first learned about the"
}
```

- Getting a particular survey using the survey id through the GET method.

The screenshot shows the Postman application interface. At the top, there are tabs for Home, Workspaces, API Network, and Explore. The search bar contains "Search Postman". On the right side, there are buttons for Invite, Settings, Help, and Upgrade.

The left sidebar has sections for Collections, Environments, and History. Under Collections, there are two items: "642" and "645". Under "645", there are five requests listed: "GET New Request", "POST New Request", "PUT New Request", "DEL New Request", and "GET New Request".

The main workspace shows a "New Request" dialog for a "GET" method to "http://localhost:8080/api/surveys/student/2". The "Params" tab is selected, showing one query parameter "Key" with a value "Value". Other tabs include Authorization, Headers (6), Body, Pre-request Script, Tests, and Settings. A "Cookies" section is also present.

Below the request dialog, the response pane displays the following JSON data:

```
1 [object Object]
2 {
3   "id": 2,
4   "firstName": "Charlie",
5   "lastName": "Davis",
6   "streetAddress": "456 Maple Ave",
7   "city": "Cityville",
8   "state": "CA",
9   "zip": "54321",
10  "telephoneNumber": "555-987-6543",
11  "email": "charlie.davis@example.com",
12  "dateOfSurvey": "2023-03-20T10:00:00.000+00:00",
13  "likedAboutCampus": "Sports",
14  "interestSource": "Television",
15  "likelihoodToRecommend": "Likely",
16  "additionalComments": "The friendly and welcoming atmosphere on campus caught my attention. I first learned about the university from a TV advertisement."
}
```

At the bottom, there are buttons for Postbot, Runner, Start Proxy, Cookies, Trash, and a Find and replace console.

- Deleting a survey based on the survey id with DELETE method.

The screenshot shows the Postman interface. In the top navigation bar, 'Explore' is selected. Below it, 'My Workspace' contains a collection named '642' which includes a '645 / New Request' item. The main workspace shows a '645 / New Request' entry with a 'DELETE' method and the URL `http://localhost:8080/api/surveys/remove/1`. The 'Headers' tab is active, showing a single header 'Content-Type: application/json'. The 'Body' tab contains the following JSON payload:

```

1. "Deleted": true
2. ...
3. ...

```

The response panel shows a successful 200 OK status with a response body containing the same JSON object. The bottom of the screen displays various Postman tools and settings.

- Checking if data is inserted into the database.

The screenshot shows the MySQL Workbench interface. The left sidebar has sections for 'Administration', 'Management', 'INSTANCE', and 'PERFORMANCE'. The main area is titled 'Query 1' and contains the following SQL code:

```

1. create database surveydb;
2. 
3. show databases;
4. 
5. use surveydb;
6. 
7. show tables;
8. 
9. select * from surveys;

```

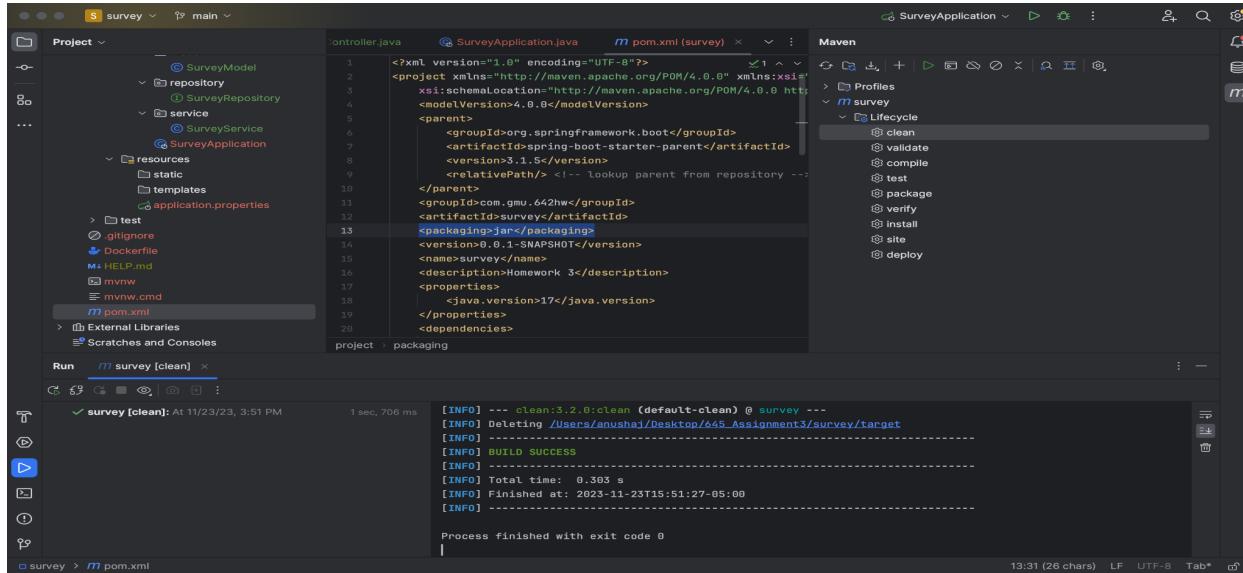
The results grid shows the following data:

id	additional_comments	city	dos	email	first_name	interest_sour...	last_name	liked_abo...
2	The friendly and welcoming atmosphere on cam...	Cityville	2023-03-20 06:00:00.000000	charlie.davis@example.com	Charlie	Television	Davis	Sports
3	I'm a big sports fan, and the university's strong...	Smalltown	2023-03-25 08:00:00.000000	emily.wilson@example.com	Emily	Internet	Wilson	Sports
4								
5								

At the bottom, the status bar shows 'Query Completed' and the command 'select * from surveys LIMIT 0, 1000'.

CREATING JAR

- Add <packaging> tag with in the pom.xml

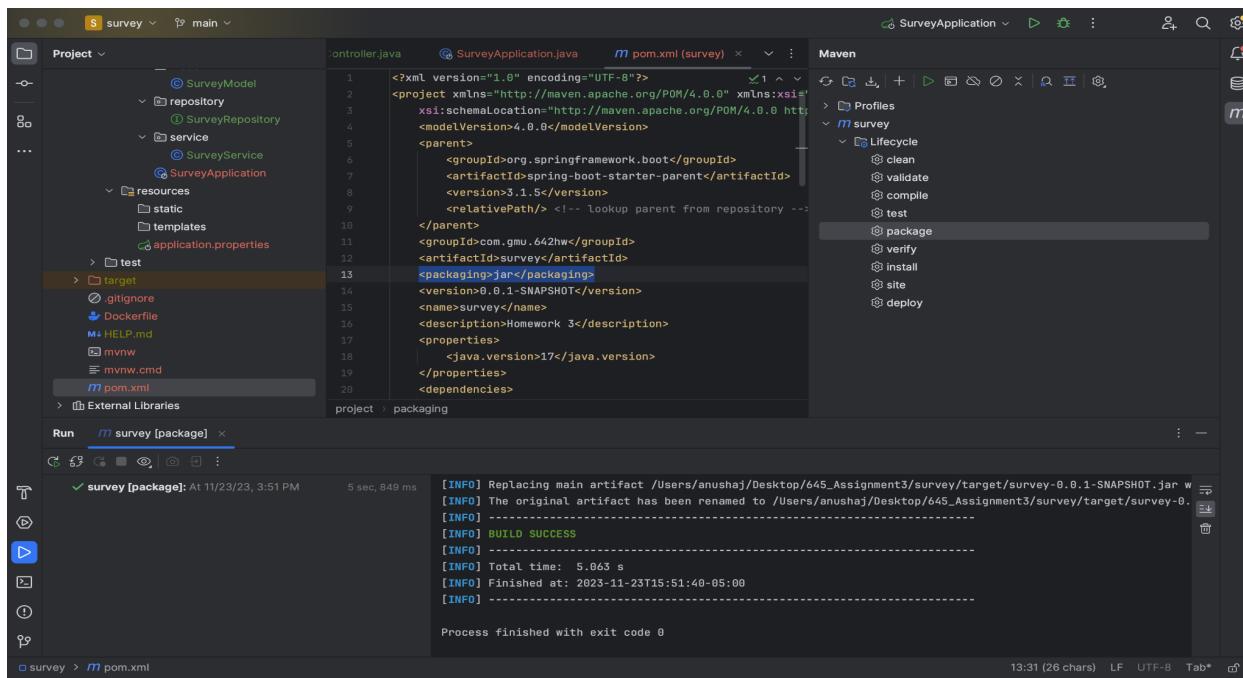


The screenshot shows the IntelliJ IDEA interface with the Maven tool window open. The 'Lifecycle' section is expanded, and the 'clean' option is selected. The 'Run' tab at the bottom has 'survey [clean]' selected. The output window shows the execution of the 'clean' command, which successfully deletes the target directory and ends with a BUILD SUCCESS message.

```
[INFO] --- clean:3.2.0:clean (default-clean) @ survey ---
[INFO] Deleting /Users/anusha/Desktop/645_Assignment3/survey/target
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 0.303 s
[INFO] Finished at: 2023-11-23T15:51:27-05:00
[INFO]

Process finished with exit code 0
```

- Then run mvn clean and mvn package

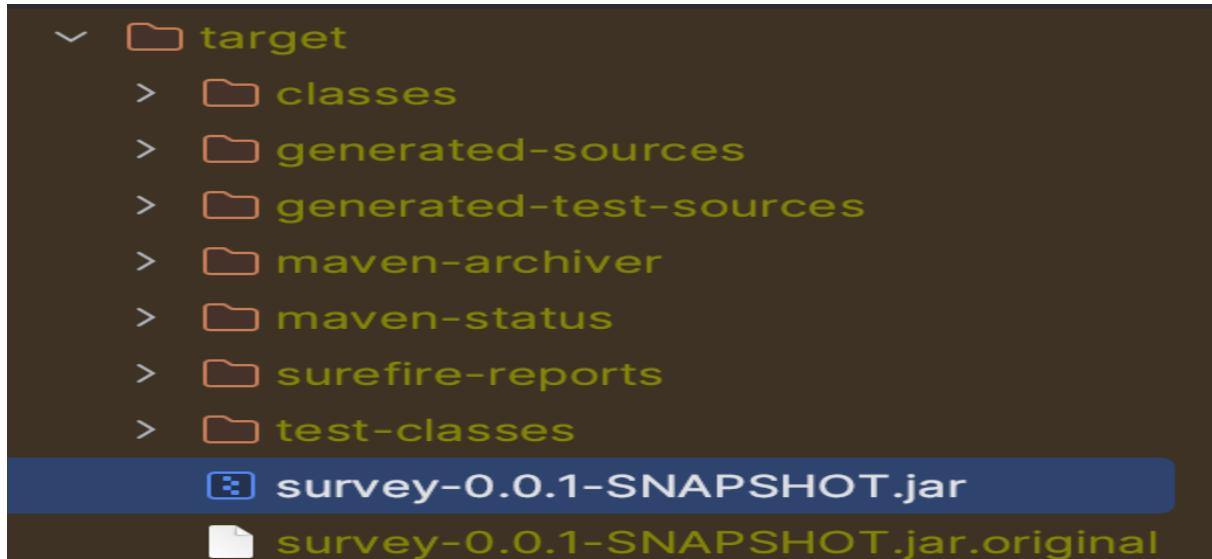


The screenshot shows the IntelliJ IDEA interface with the Maven tool window open. The 'Lifecycle' section is expanded, and the 'package' option is selected. The 'Run' tab at the bottom has 'survey [package]' selected. The output window shows the execution of the 'package' command, which replaces the main artifact and ends with a BUILD SUCCESS message.

```
[INFO] Replacing main artifact /Users/anusha/Desktop/645_Assignment3/survey/target/survey-0.0.1-SNAPSHOT.jar with /Users/anusha/Desktop/645_Assignment3/survey/target/survey-0.0.1-SNAPSHOT.jar
[INFO] The original artifact has been renamed to /Users/anusha/Desktop/645_Assignment3/survey/target/survey-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 5.063 s
[INFO] Finished at: 2023-11-23T15:51:40-05:00
[INFO]

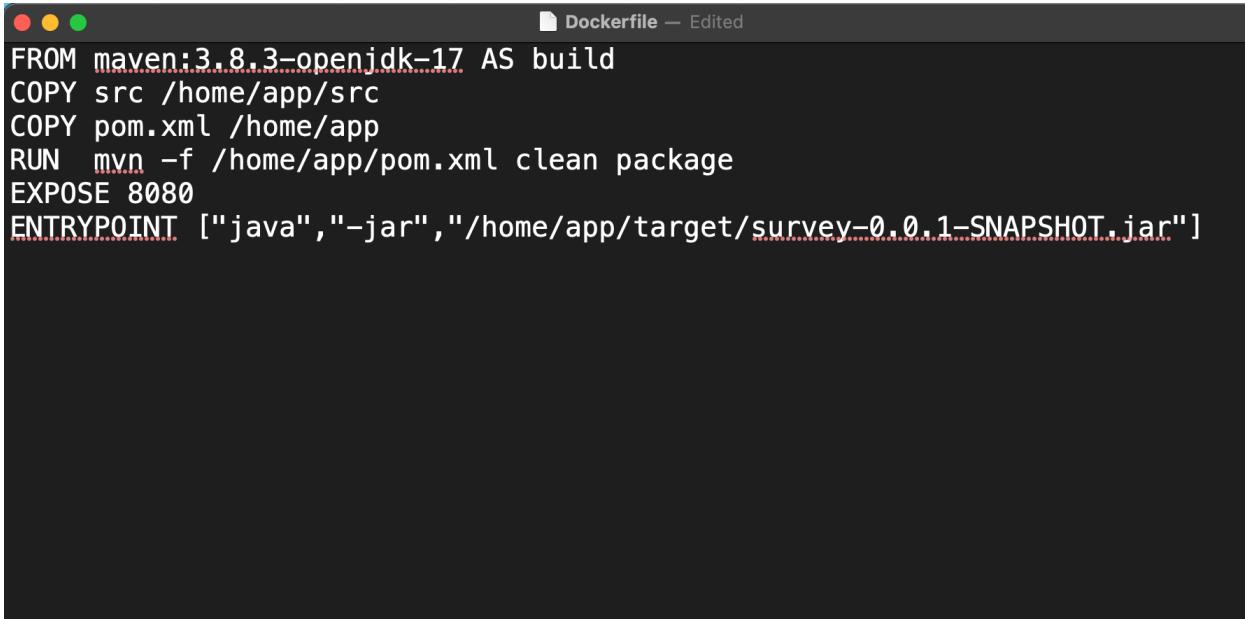
Process finished with exit code 0
```

- A jar file will be created in the target folder



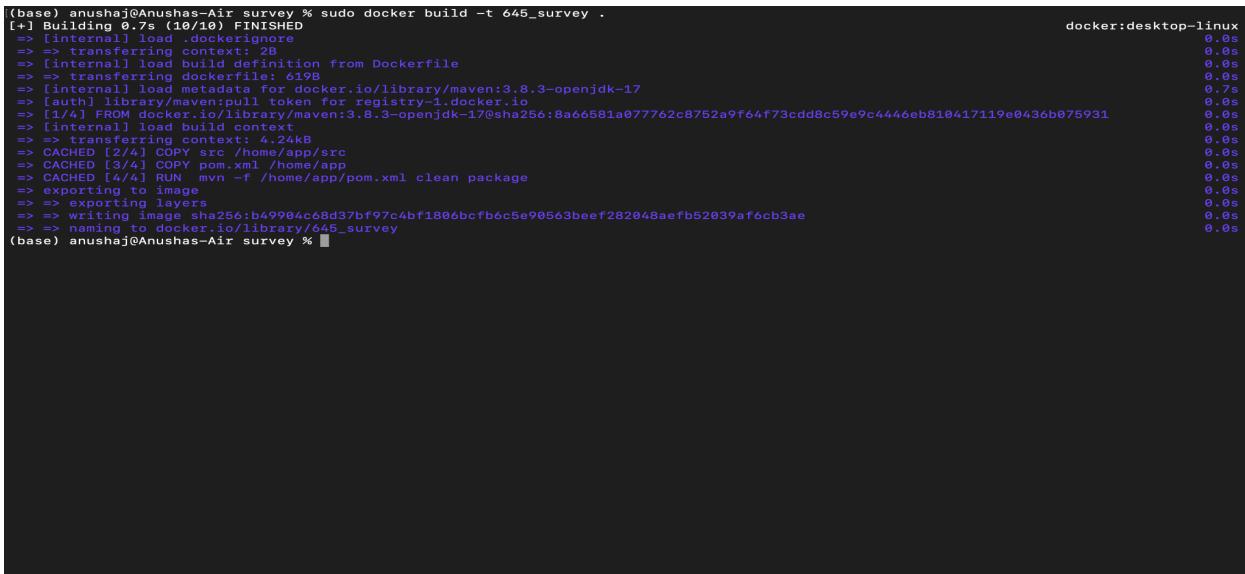
DOCKER

- Create a file named Dockerfile



```
FROM maven:3.8.3-openjdk-17 AS build
COPY src /home/app/src
COPY pom.xml /home/app
RUN mvn -f /home/app/pom.xml clean package
EXPOSE 8080
ENTRYPOINT ["java","-jar","/home/app/target/survey-0.0.1-SNAPSHOT.jar"]
```

- In the terminal in survey path, enter the commands
sudo docker build -t 645_survey .



```
(base) anushaj@Anushas-Air survey % sudo docker build -t 645_survey .
[+] Building 0.7s (10/10) FINISHED
=> [internal] load .dockerignore
=> [internal] transfer context: 2B
=> [internal] load metadata for docker.io/library/maven:3.8.3-openjdk-17
=> [auth] library/maven:pull token for registry-1.docker.io
=> [1/4] FROM docker.io/library/maven:3.8.3-openjdk-17@sha256:8a66581a077762c8752a9f64f73cdd8c59e9c4446eb810417119e0436b075931
=> [internal] load build context
=> [internal] transfer context: 4.24kB
=> CACHED [2/4] COPY src /home/app/src
=> CACHED [3/4] COPY pom.xml /home/app
=> CACHED [4/4] RUN mvn -f /home/app/pom.xml clean package
=> writing image sha256:b49904c68d37bf97c4bf1806bcfb6c5e90563beef282048aefb52039af6cb3ae
=> naming to docker.io/library/645_survey
(base) anushaj@Anushas-Air survey %
```

- docker run -it -p 8182:8080 645_survey.

```
>>> => exporting layers
=>>> writing image sha256:b49904c68d37bf97c4bf1806bcfb6c5e90563beef282048aefb52039af6cb3ae
=>>> naming to docker.io/library/645_survey
(base) anusha@Anushas-Air survey % docker run -it -p 8182:8080 645_survey

:: Spring Boot :: (v3.1.5)

2023-11-23T20:59:42.417Z INFO 1 --- [           main] com.gmu.hw.survey.SurveyApplication : Starting SurveyApplication v0.0.1-SNAPSHOT us
ing Java 17.0.1 with PID 1 (/home/app/target/survey-0.0.1-SNAPSHOT.jar started by root in /)
2023-11-23T20:59:42.419Z INFO 1 --- [           main] com.gmu.hw.survey.SurveyApplication : No active profile set, falling back to 1 defa
ult profile: "default"
2023-11-23T20:59:43.201Z INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in
DEFAULT mode.
2023-11-23T20:59:43.241Z INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 3
2 ms. Found 1 JPA repository interfaces.
2023-11-23T20:59:43.578Z INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2023-11-23T20:59:43.584Z INFO 1 --- [           main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-11-23T20:59:43.584Z INFO 1 --- [           main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.
15]
2023-11-23T20:59:43.626Z INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplicationCo
ntext
2023-11-23T20:59:43.626Z INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization co
mpleted in 1163 ms
2023-11-23T20:59:43.729Z INFO 1 --- [           main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [na
me: default]
2023-11-23T20:59:43.760Z INFO 1 --- [           main] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.2.13.
Final
2023-11-23T20:59:43.762Z INFO 1 --- [           main] org.hibernate.cfg.Environment : HHH000406: Using bytecode reflection optimize
r
2023-11-23T20:59:43.898Z INFO 1 --- [           main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class t
ransformer
2023-11-23T20:59:43.928Z INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2023-11-23T20:59:44.320Z INFO 1 --- [           main] com.zaxxer.hikari.HikariPool : HikariPool-1 - Added connection com.mysql.cj.
jdbc.ConnectionImpl@4e868be5
2023-11-23T20:59:44.322Z INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2023-11-23T20:59:44.837Z INFO 1 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hi
bernate.transaction.jta.platform' to enable JTA platform integration)
2023-11-23T20:59:44.886Z INFO 1 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for pers
```

POSTMAN WITH DOCKER

- Retrieving all the surveys from the GET method.

The screenshot shows the Postman interface with a successful GET request to `http://localhost:8182/api/surveys/all`. The response status is 200 OK, and the response body is a JSON array of survey records:

```
[{"id": 2, "firstName": "Charlie", "lastName": "Davis", "streetAddress": "456 Maple Ave", "city": "Cityville", "state": "CA", "zip": "54321", "telephoneNumber": "555-987-6543", "email": "charlie.davis@example.com", "dateOfSurvey": "2023-03-20T08:00:00.000+00:00", "likedAboutCampus": "Safety", "interestSource": "Television", "likelihoodToRecommend": "Likely", "additionalComments": "The friendly and welcoming atmosphere on campus caught my attention. I first learned about the university from a TV advertisement."}, {"id": 3, "firstName": "Emily", "lastName": "Wilson", "streetAddress": "567 Elm Rd", "city": "Hometown", "state": "CA", "zip": "12345", "telephoneNumber": "555-123-4567", "email": "emily.wilson@example.com", "dateOfSurvey": "2023-03-15T08:00:00.000+00:00", "likedAboutCampus": "Location", "interestSource": "Friends", "likelihoodToRecommend": "Very Likely", "additionalComments": "The campus's proximity to the beach is amazing. I heard about this university from some friends who spoke highly of it."}]
```

- Creating a new survey using the POST method.

The screenshot shows the Postman interface with a successful POST request to `http://localhost:8182/api/surveys/newStudent`. The response status is 200 OK, and the response body is a JSON survey record:

```
{"id": 4, "firstName": "Alice", "lastName": "Smith", "streetAddress": "123 Oak St", "city": "Hometown", "state": "CA", "zip": "12345", "telephoneNumber": "555-123-4567", "email": "alice.smith@example.com", "dateOfSurvey": "2023-03-15T08:00:00.000+00:00", "likedAboutCampus": "Safety", "interestSource": "Friends", "likelihoodToRecommend": "Very Likely", "additionalComments": "The campus's proximity to the beach is amazing. I heard about this university from some friends who spoke highly of it."}
```

- Updating a particular survey details using the UPDATE method.

The screenshot shows the Postman interface with a PUT request to `http://localhost:8182/api/surveys/oldStudent/2`. The request body is a JSON object containing survey details for a student named Charlie Davis. The response status is 200 OK, and the response body is identical to the request body.

```

1   "id": 2,
2   "firstName": "Charlie",
3   "lastName": "Davis",
4   "streetAddress": "456 Maple Ave",
5   "city": "Cityville",
6   "state": "CA",
7   "zip": "54321",
8   "telephoneNumber": "555-123-5678",
9   "email": "charlie.davis@example.com",
10  "dateOfSurvey": "2023-03-20T00:00:00.000+00:00",
11  "likedAboutCampus": "Sports",
12  "interestSource": "Television",
13  "likelihoodToRecommend": "likely",
14  "additionalComments": "The friendly and welcoming atmosphere on campus caught my attention. I first learned about the university from a TV advertisement."
15

```

- Deleting a survey based on the survey id with DELETE method.

The screenshot shows the Postman interface with a DELETE request to `http://localhost:8182/api/surveys/remove/2`. The request body is a JSON object with a single key "Deleted" set to true. The response status is 200 OK, and the response body is identical to the request body.

```

1   "Deleted": true
2
3

```

- Getting a particular survey using the survey id through the GET method.

The screenshot shows the Postman application interface. In the top navigation bar, there are links for Home, Workspaces, API Network, and Explore. A search bar is present at the top right. The main workspace is titled "My Workspace" and contains several requests under "Collections". One collection is expanded, showing a "GET New Request" for the URL `http://localhost:8182/api/surveys/student/3`. The "Params" tab is selected, showing a single query parameter "Key": "Value". The "Body" tab is selected, displaying the JSON response:

```

1  {
2      "id": 3,
3      "firstName": "Emily",
4      "lastName": "Wilson",
5      "streetAddress": "567 Elm Rd",
6      "city": "Smalltown",
7      "state": "FL",
8      "zip": "34567",
9      "telephoneNumber": "555-789-0123",
10     "email": "emily.wilson@example.com",
11     "dateOfSurvey": "2023-03-25T08:00:00.000+00:00",
12     "likedAboutCampus": "Sports",
13     "interestSource": "Internet",
14     "likelihoodToRecommend": "Unlikely",
15     "additionalComments": "I'm a big sports fan, and the university's strong focus on sports programs attracted me. I found out about the university while browsing online."
16

```

The status bar at the bottom indicates a 200 OK response with a time of 150 ms and a size of 750 B.

DOCKER IMAGE

Run the below commands to push the image to docker

- docker tag 645_survey ajagadis/645_survey
- docker push ajagadis/645_survey

```
(base) anushaj@Anushas-Air survey % docker tag 645_survey ajagadis/645_survey
(base) anushaj@Anushas-Air survey % docker push ajagadis/645_survey
Using default tag: latest
The push refers to repository [docker.io/ajagadis/645_survey]
5539dac4206e: Pushed
88515675eec9: Pushed
0a9e5de54a13: Pushed
366e0ac3b92a: Mounted from library/maven
48f9189d20fc: Mounted from library/maven
eaab9d11cc5d: Mounted from library/maven
4da0bd95a2ee: Mounted from library/maven
3ea4fe4d8bedd: Mounted from library/maven
6ba7dd301dbf: Mounted from library/maven
ac6d61109c4: Mounted from library/maven
latest: digest: sha256:9ff6bb8a8b2e528a8bec8ef4f2e216f2aab56618717b6391ebb1dfc8f7107407 size: 2420
(base) anushaj@Anushas-Air survey %
```

The screenshot shows a web browser window with the URL hub.docker.com/repository/docker/ajagadis/645_survey/general. The page displays the details of the Docker repository 'ajagadis / 645_survey'. The 'General' tab is selected. Key information shown includes:

- Description:** A placeholder message stating "This repository does not have a description".
- Last pushed:** A few seconds ago.
- Docker commands:** A text input field containing the command `docker push ajagadis/645_survey :tagname`.
- Tags:** A table showing one tag: `latest` (Image type, pushed a few seconds ago).
- Automated Builds:** A section explaining how to automatically build and tag images using GitHub or Bitbucket. It mentions that this feature is available with Pro, Team, and Business subscriptions.

EC2 INSTANCE

- Create two EC2 instances by choosing Ubuntu from the AMI list by choosing t2.medium from the Instance types available.

The screenshot shows the AWS Management Console with the URL us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:instanceState=running. The left sidebar is open, showing navigation options like EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instances Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups), and CloudShell/Feedback. The main content area is titled "Instances (2) Info" and shows a table with two rows. The columns are Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Z..., and Public. The first row has "instance1" and "i-00bdd5110defff3b" in the first two columns, and the second row has "instance2" and "i-0b41a8968d33bf36e". Both rows show "Running" in the Instance state column and "t2.medium" in the Instance type column. The Status check column shows "2/2 checks passed" with a green icon. The Alarm status column shows "View alarms" with a plus sign. The Availability Z... column shows "us-east-1b" and the Public column shows "ec2-3-". Below the table, a modal window titled "Select an instance" is open, showing the same two instances. At the bottom of the page, there are links for "CloudShell" and "Feedback".

- Now allocate the elastic ip address with the instance.

The screenshot shows the AWS Management Console with the URL us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Addresses. The left sidebar is open, showing the same navigation options as the previous screenshot. The main content area is titled "Elastic IP addresses (2)" and shows a table with two rows. The columns are Name, Allocated IPv4 add..., Type, Allocation ID, and Reverse DNS record. The first row has "-" in all columns, and the second row has "3.215.61.95" in the Allocated IPv4 add... column, "Public IP" in the Type column, "eipalloc-0477e75a3cd53d1b8" in the Allocation ID column, and "-" in the Reverse DNS record column. Above the table, a green notification bar says "Elastic IP address associated successfully. Elastic IP address 34.237.18.250 has been associated with instance i-0b41a8968d33bf36e". Below the table, a callout box says "View IP address usage and recommendations to release unused IPs with Public IP insights." At the bottom of the page, there are links for "CloudShell" and "Feedback".

- Connect to EC2 instance enter the following commands
 - a) sudo apt-get update

```
...wnloads -- ubuntu@ip-172-31-89-110:~ ssh -i newkey.pem ubuntu@ec2-3-215-61-95.compute-1.amazonaws.com ...s -- ubuntu@ip-172-31-85-117:~ ssh -i newkey.pem ubuntu@ec2-34-237-18-250.compute-1.amazonaws.com
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-89-110:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1200 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [252 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [16.1 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [1168 kB]
Get:15 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [985 kB]
Get:16 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [192 kB]
Get:17 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [11.4 kB]
Get:18 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [1134 kB]
Get:19 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [185 kB]
```

- b) sudo apt install docker.io
- c) docker -v

To deploy Rancher navigate to the rancher website and copy the command:
<https://www.rancher.com/quick-start>

```
...wnloads -- ubuntu@ip-172-31-89-110:~ ssh -i newkey.pem ubuntu@ec2-3-215-61-95.compute-1.amazonaws.com ...s -- ubuntu@ip-172-31-85-117:~ ssh -i newkey.pem ubuntu@ec2-34-237-18-250.compute-1.amazonaws.com
Reading package lists... Done
ubuntu@ip-172-31-89-110:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse
| zfsutils
The following NEW packages will be installed:
bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 62 not upgraded.
Need to get 69.7 MB of archives.
After this operation, 267 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-1ubuntu3 [34.4 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.1.7-0ubuntu1~22.04.1 [4249 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 containerd amd64 1.7.2-0ubuntu1~22.04.1 [36.0 MB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 dns-root-data all 202101101 [5256 B]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dnsmasq-base amd64 2.86-1ubuntu0.3 [354 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 docker.io amd64 24.0.5-0ubuntu1~22.04.1 [28.9 MB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 ubuntu-fan all 0.12.16 [35.2 kB]
Fetched 69.7 MB in 2s (28.9 MB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 64726 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.6-1_amd64.deb ...
Unpacking pigz (2.6-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.7-1ubuntu3_amd64.deb ...
Unpacking bridge-utils (1.7-1ubuntu3) ...
Selecting previously unselected package runc.
Preparing to unpack .../2-runc_1.1.7-0ubuntu1~22.04.1_amd64.deb ...
Unpacking runc (1.1.7-0ubuntu1~22.04.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../3-containerd_1.7.2-0ubuntu1~22.04.1_amd64.deb ...
```

- d) To launch the rancher as a docker container run the command

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

```

...wnloads -- ubuntu@ip-172-31-89-110:~ ssh -i newkey.pem ubuntu@ec2-3-215-61-95.compute-1.amazonaws.com
No user sessions are running outdated binaries.

No VM guests are running hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-89-110:~$ sudo docker run --privileged -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher
Unable to find image 'rancher/rancher:latest' locally
latest: Pulling from rancher/rancher
7d9ba2420ea2: Pull complete
b66830f3e912: Pull complete
ab9de5009c8: Pull complete
c8525db440cb: Pull complete
afa3cdb22dc5: Pull complete
e4c379c4bdab: Pull complete
3e4c3b5c740f: Pull complete
9b0d2d798ac2: Pull complete
ecd29de33a95: Pull complete
cadb2e90dd87: Pull complete
8b7241380918: Pull complete
fc8c791788bb6: Pull complete
9b4a9fda1712: Pull complete
efec57f2f8c5: Pull complete
877b33558361: Pull complete
9b2b25ebc011: Pull complete
ea9b5b75da0f: Pull complete
7e02e1b4edd4: Pull complete
173ee7bee6bf: Pull complete
e5d635e9d931: Pull complete
Digest: sha256:b8e448b6b47c5b0a510e3e4443e761be89846ca5c90f287b1a32c7b69cf97cb80
Status: Downloaded newer image for rancher/rancher:latest
2beead0dcc6d11beef3536b3a6aac1f2ea89e08f75bd956b9a5315hb16edcc55
ubuntu@ip-172-31-89-110:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
NAMES
2beead0dcc6d      rancher/rancher   "/entrypoint.sh"    About a minute ago   Up About a minute   0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0
.0.0.443->443/tcp, :::443->443/tcp   cranky_nightingale
ubuntu@ip-172-31-89-110:~$ docker logs 2beead0dcc6d 2>&1 | grep "Bootstrap Password:"
ubuntu@ip-172-31-89-110:~$ docker logs 2beead0dcc6d 2>&1 | grep "Bootstrap Password:"
ubuntu@ip-172-31-89-110:~$ sudo docker logs 2beead0dcc6d 2>&1 | grep "Bootstrap Password:"
2023/11/26 00:07:32 [INFO] Bootstrap Password: rjjfkxbvphckc6ckjfbjqkxqfbhc7c98k6thnhd6r4pjx8h9zsrs9
ubuntu@ip-172-31-89-110:~$ █

```

- Next step,to get the container id of the Docker run “ sudo docker ps ” command
And then copy the container id to get the bootstrap password of the Rancher
sudo docker logs d17e192c92da 2>&1 | grep "Bootstrap Password:"
- Now, copy the password obtained and login, now we can create a new password and launch the rancher.

CLUSTER CREATION

- In the Rancher dashboard click on create, select custom as the option

The screenshot shows the Rancher dashboard under 'Cluster Management'. On the left, there's a sidebar with 'Clusters' (2), 'Cloud Credentials', 'Drivers', 'RKE1 Configuration', and 'Advanced'. The main area has two sections: 'Provision new nodes and create a cluster using RKE2/K3s' and 'Use existing nodes and create a cluster using RKE2/K3s'. Under the first section, there are icons for Amazon EC2, Azure, DigitalOcean, and Harvester. Under the second section, there is an icon for Linode and one for VMware vSphere. A 'Custom' option is also present. At the top right, there's a toggle switch between 'RKE1' and 'RKE2/K3s'. A 'Cancel' button is at the bottom right.

- Give a name to the cluster and then create

The screenshot shows the 'Cluster: Create Custom' configuration page. The 'Cluster Name' field is set to 'rancher'. The 'Cluster Description' field contains placeholder text: 'Any text you want that better describes this cluster'. The 'Cluster Configuration' section is expanded, showing the 'Basics' tab selected. Under 'Basics', the 'Kubernetes Version' is set to 'v1.26.10+rke2r2', 'Cloud Provider' is 'Default - RKE2 Embedded', and 'Container Network' is 'calico'. There are also checkboxes for 'Show deprecated Kubernetes patch versions' and 'Pod Security Policies have been removed in Kubernetes v1.25, use Pod Security Admission instead.' The 'Security' tab shows a dropdown for 'CIS Profile' set to '(None)'. At the bottom right are 'Cancel', 'Edit as YAML', and a blue 'Create' button.

- After some time the clusters will be active.

Cluster: rancher (Active)
Namespace: fleet-default Age: 7 mins
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://ec2-3-215-61-95.compute-1.amazonaws.com/system-agent-install.sh | sudo sh -s - --server https://ec2-3-215-61-95.compute-1.amazonaws.com --label 'cattle.io/os=linux' --token xlnhrmmngrnkgbv9f5sqbnmwntdbjqc69fffed7421afe5d2e8edbf937de1298317b5529edc3833510738a300dfe01db --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.

- After creation, copy the curl command and run it in the second EC2 instance where rancher has not been deployed.

```
...wnloads -- ubuntu@ip-172-31-89-110:~ ssh -i newkey.pem ubuntu@ec2-3-215-61-95.compute-1.amazonaws.com ...s -- ubuntu@ip-172-31-85-117:~ ssh -i newkey.pem ubuntu@ec2-34-237-18-250.compute-1.amazonaws.com +  
applicable law.  
  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
ubuntu@ip-172-31-85-117:~$ curl --insecure -fL https://ec2-3-215-61-95.compute-1.amazonaws.com/system-agent-install.sh | sudo sh  
s --server https://ec2-3-215-61-95.compute-1.amazonaws.com --label 'cattle.io/os=linux' --token xlnhrmmngrnkgbv9f5sqbnmwntdbjqc69fffed7421afe5d2e8edbf937de1298317b5529edc3833510738a300dfe01db --etcd --controlplane --worker  
% Total    % Received % Xferd  Average Speed   Time     Time      Current  
          Dload  Upload Total Spent   Left Speed  
100 30903    0 30903    0    0  1312k      0  ---:---:---:---:---:--- 1257k  
[INFO] Label: cattle.io/os=linux  
[INFO] Role requested: etcd  
[INFO] Role requested: controlplane  
[INFO] Role requested: worker  
[INFO] Using default agent configuration directory /etc/rancher/agent  
[INFO] Using default agent var directory /var/lib/rancher/agent  
[INFO] Determined CA is necessary to connect to Rancher  
[INFO] Successfully downloaded CA certificate  
[INFO] Value from https://ec2-3-215-61-95.compute-1.amazonaws.com/cacerts is an x509 certificate  
[INFO] Successfully tested Rancher connection  
[INFO] Downloading rancher-system-agent binary from https://ec2-3-215-61-95.compute-1.amazonaws.com/assets/rancher-system-agent-a  
md64  
[INFO] Successfully downloaded the rancher-system-agent binary.  
[INFO] Downloading rancher-system-agent-uninstall.sh script from https://ec2-3-215-61-95.compute-1.amazonaws.com/assets/system-ag  
ent-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-ag  
ent.service.  
[INFO] Starting/restarting rancher-system-agent.service  
ubuntu@ip-172-31-85-117:~$
```

- Now open the cluster, and then open workload tab and open deployment tab, then create a deployment with three replicas and the docker image and add nodeport in networking.

Deployment: Create

Namespace: default

Name: deployment1

Replicas: 3

Deployment Pod container-0 + Add Container

General

Container Name: container-0

Image: Container Image: ajagadis / 645_survey

Pull Policy: Always

Networking

Cancel Edit as YAML Create

Storage

Container Image: ajagadis / 645_survey

Pull Policy: Always

Networking

Service Type: Node Port

Name: nodeport

Private Container Port: 8080

Protocol: TCP

Listening Port: 8080

Add Port or Service

Command

WorkingDir

Arguments

Stdin

Cancel Edit as YAML Create

- The nodeport deployment can be accessed using the below url.
<https://ec2-23-20-134-74.compute-1.amazonaws.com/k8s/clusters/c-m-hrw8c9zn/api/v1/namespaces/default/services/http:deploy:8080/proxy/swe/>

- After some time the deployment will be active.

Deployment: deployment2 (Active)

Namespaces: default Age: 1 mins Pod Restarts: 0

Image: ajagadis/645_survey Ready: 3/3 Up-to-date: 3 Available: 3 Annotations: Show 1 annotation

Pods by State

State	Name	Image	Ready	Restarts	IP	Node	Age
Running	deployment2-6448d4f57d-6rxpg	ajagadis/645_survey	1/1	0	10.42.86.60	ip-172-31-85-117	10 secs
Running	deployment2-6448d4f57d-l2x5v	ajagadis/645_survey	1/1	0	10.42.86.62	ip-172-31-85-117	7 secs
Running	deployment2-6448d4f57d-n7xfs	ajagadis/645_survey	1/1	0	10.42.86.61	ip-172-31-85-117	9 secs

- By following the previous method create one more deployment for load balancer and after activation it can be accessed using the url.
<https://ec2-23-20-134-74.compute-1.amazonaws.com/k8s/clusters/c-m-hrw8c9zn/api/v1/namespaces/default/services/http:deploy-lb:8080/proxy/api/surveys/all>

```
[{"id":3,"firstName":"Emily","lastName":"Wilson","streetAddress":"567 Elm Rd","city":"Smalltown","state":"FL","zip":"34567","telephoneNumber":"555-789-0123","email":"emily.wilson@example.com","dateOfSurvey":"2023-03-25T08:00:00.000Z","likedAboutCampus":"Sports","interestSource":"Elm","likelihoodToRecommend":"Unlikely","additionalComments":"I'm a big sports fan, and the university's strong focus on sports programs attracted me. I found out about the university while browsing online."}, {"id":4,"firstName":"Sarah","lastName":"Smith","streetAddress":"123 Oak St","city":"Hometown","state":"CA","zip":"12345","telephoneNumber":"555-123-4567","email":"sarah.smith@example.com","dateOfSurvey":"2023-03-25T08:00:00.000Z","likedAboutCampus":"Location","interestSource":"Friends","likelihoodToRecommend":"Very Likely","additionalComments":"The campus's proximity to the beach is amazing. I heard about this university from some friends who spoke highly of it."}]
```

JENKINS

- Launch an EC2 instance and pick Ubuntu from the Amazon Machine Images (AMI) catalog
- Choose the t2.medium instance type for computing capacity and memory

The screenshot shows the AWS EC2 Instances details page for an instance named "jenkins". The instance ID is i-0863157e0e12d7cfa. Key details include:

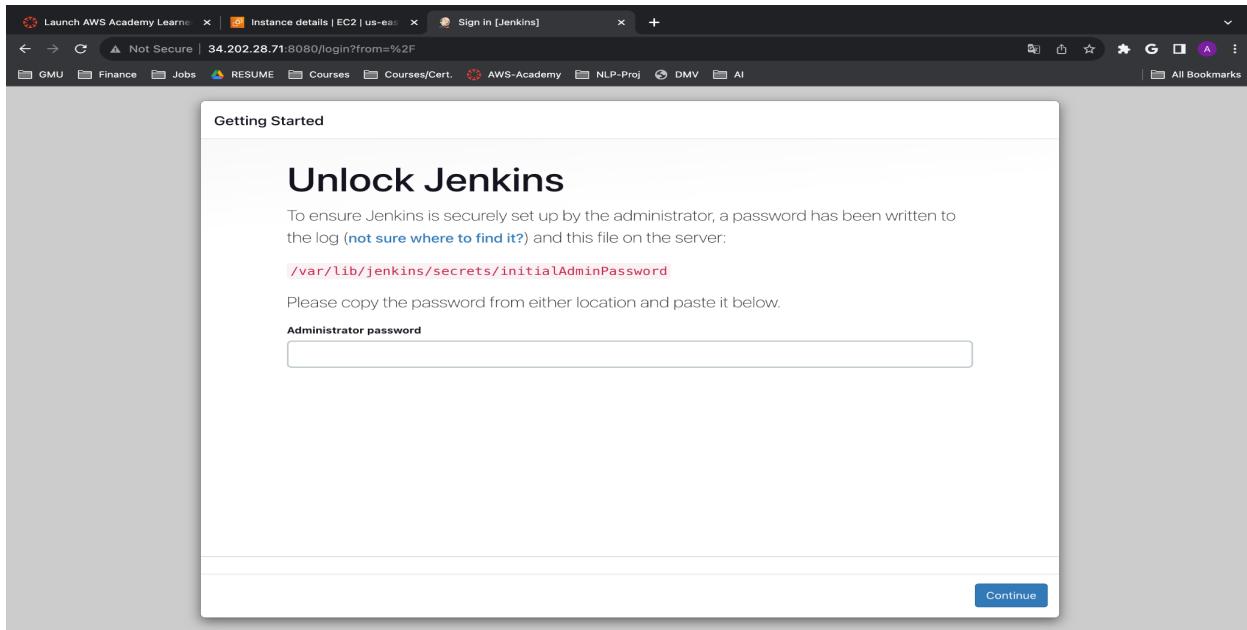
- Public IPv4 address:** 34.202.28.71
- Instance state:** Running
- Private IP4 address:** 172.31.88.199
- Public IPv4 DNS:** ec2-34-202-28-71.compute-1.amazonaws.com
- Hostname type:** ip-172-31-88-199.ec2.internal
- IP name:** ip-172-31-88-199.ec2.internal
- Answer private resource DNS name:** IPv4 (A)
- Instance type:** t2.medium
- VPC ID:** vpc-08418619ff1bf1c6f
- Subnet ID:** subnet-0a2f8da17608780d1
- IAM Role:** -
- IMDSv2:** Required

- Connect to the EC2 instance and enter the following commands given below:
 - a) sudo apt-get update
 - b) sudo apt install docker.io
 - c) Verify that docker is installed by running the command **docker -v**
 - d) curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \ /usr/share/keyrings/jenkins-keyring.asc > /dev/null
 - e) 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
 - f) sudo apt-get update
 - g) sudo apt install openjdk-11-jdk
 - h) sudo apt-get update
 - i) sudo apt-get install jenkins
 - f) Enter the command `systemctl status jenkins` and check the status of Jenkins

```
ubuntu@ip-172-31-88-199:~$ systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2023-11-29 00:58:37 UTC; 11min ago
     Main PID: 5880 (java)
        Tasks: 48 (limit: 4667)
       Memory: 1.2G
          CPU: 46.807s
        CGroup: /system.slice/jenkins.service
            └─5880 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war

Nov 29 00:58:21 ip-172-31-88-199 jenkins[5880]: 21c495183e1a47a5b358f8bfa5ca17b1
Nov 29 00:58:21 ip-172-31-88-199 jenkins[5880]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Nov 29 00:58:21 ip-172-31-88-199 jenkins[5880]: ****
Nov 29 00:58:21 ip-172-31-88-199 jenkins[5880]: ****
Nov 29 00:58:21 ip-172-31-88-199 jenkins[5880]: ****
Nov 29 00:58:37 ip-172-31-88-199 jenkins[5880]: 2023-11-29 00:58:37.226+0000 [id=33]      INFO      jenkins.InitReactorP
Nov 29 00:58:37 ip-172-31-88-199 jenkins[5880]: 2023-11-29 00:58:37.245+0000 [id=24]      INFO      hudson.lifecycle.Lif
Nov 29 00:58:37 ip-172-31-88-199 systemd[1]: Started Jenkins Continuous Integration Server.
Nov 29 00:58:37 ip-172-31-88-199 jenkins[5880]: 2023-11-29 00:58:37.290+0000 [id=49]      INFO      h.m.DownloadService$
Nov 29 00:58:37 ip-172-31-88-199 jenkins[5880]: 2023-11-29 00:58:37.290+0000 [id=49]      INFO      hudson.util.Retrive#
Lines 1-20/20 (END)
```

- Once the Jenkins status is active go to the EC2 instance security and allow inbound traffic on port 8080
- Now, launch the Jenkins instance using the public IPv4 address: <http://18.215.66.252:8080/>.



- Login using the admin password and launch the Jenkins dashboard

The screenshot shows the Jenkins dashboard at the URL 34.202.28.71:8080/computer/new. The main content area features a "Welcome to Jenkins!" message with a sub-instruction: "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." Below this, there's a "Start building your software project" button and a "Create a job" link. On the left, there are navigation links for "New Item", "People", "Build History", "Manage Jenkins", and "My Views". Under "Build Queue", it says "No builds in the queue." Under "Build Executor Status", it lists "1 Idle" and "2 Idle". On the right, there's a "Set up a distributed build" section with links to "Set up an agent" (with a monitor icon), "Configure a cloud" (with a cloud icon), and "Learn more about distributed builds" (with a question mark icon). The bottom right corner shows "REST API" and "Jenkins 2.426.1".

- Open a terminal session on the EC2 server outside Jenkins.
- Enter the commands to install kubectl for Kubernetes management.
 - a) sudo apt install snapd
 - b) sudo snap install kubectl –classic
 - c) Login as Jenkins user : sudo su jenkins
- In Rancher, navigate to your cluster and download the KubeConfig file
- Enter the following command to enable kubectl connection with our rancher cluster.
 - a) Cd /var/lib/jenkins
 - b) > .kube
 - c) > config
 - d) cat > config
 - e) Copy the content from the KubeConfig file and paste it here and do ctrl + d.
 kubectl config current-context

```
apiVersion: v1
kind: Config
clusters:
- name: "rancher"
  cluster:
    server: "https://ec2-3-215-61-95.compute-1.amazonaws.com/k8s/clusters/c-m-bs7h7v5p"
    certificate-authority-data: "LS0tLS1CRUdJt1BDRVJUSUJQ0FURS0tLS0tCk1JSUJ2VENDQ\
      VdpZ0f3SUJBz0lCQURBS0jNz3Foa2pPUFFRREFqKqdNUnd3R2dZRFZRUUtFe5YzVc1aGJxb6oKY\ \
      kdSemRHVnVaWE10YjNkbk1TWxdKQV1EV1FRRECMWt1VzVoYldsmaJHbHpkR1Z1WlhJdFkyRkFNV\ \
      GN3TURrMqpOkekxTKRBZUZ3Mh1Nekv4TwpZd01EQTNNe1JhRncwek16RxhNak13TURBM016UmFNR\ \
      V14SERBYUJnTlZCQQwPUCKuyUiViBuZYYVd0c2FYtjBaVzVsY2kxmdNtY3hKakFrQmd0VkJBTU1IV\ \
      1I1y0GdFGXTnNHWE4wWlc1bGNpMoKWVVBcE56QxdPVFUzTwpVME1Ga3dFd1IIS29aSxpqMENBU\ \
      V1JS29aSxpgMERBUwNEUwdBRVTqNRMQ01LSDhiWaOrb0JjRG5nckI4MDJsaEJhOUFVN205NVFd6\ \
      GluZUh3Mz16RWT2dzNYdnMwZ0h2aFhuMkf1REMvZVhaYktCeW95CnVBBmJWck8ywTZ0Q01FQXdEZ\ \
      11EV1iUwUEFRSC9QVFEEQWdLa01B0EdBMVVkRXdFQi93UUZNUQ1CQWY4d0hRWUQKV1lwT0JCUWUGT\ \
      @VyaHN6eGQuWtc0M1JmYKNEdmVKcnQzYw1sTUfvr0NDcUdTTQ5QkFNQ0EwZ0FNRVVDSVFEVQowd\ \
      WZiVGJPS1JXU1NQc3VnQ2Mzd0tYMVQxeGtMSmh2WEE0cW02TjY1SUfJZ01nWnpvMU1neDAxaWgxa\ \
      WJnQTMT2CnZwR1BwX1UQUhBw1MUEJsRXNiZnRrPQotLS0tLUVORCBDRVJUSUZjQ0FURS0tLS0t"

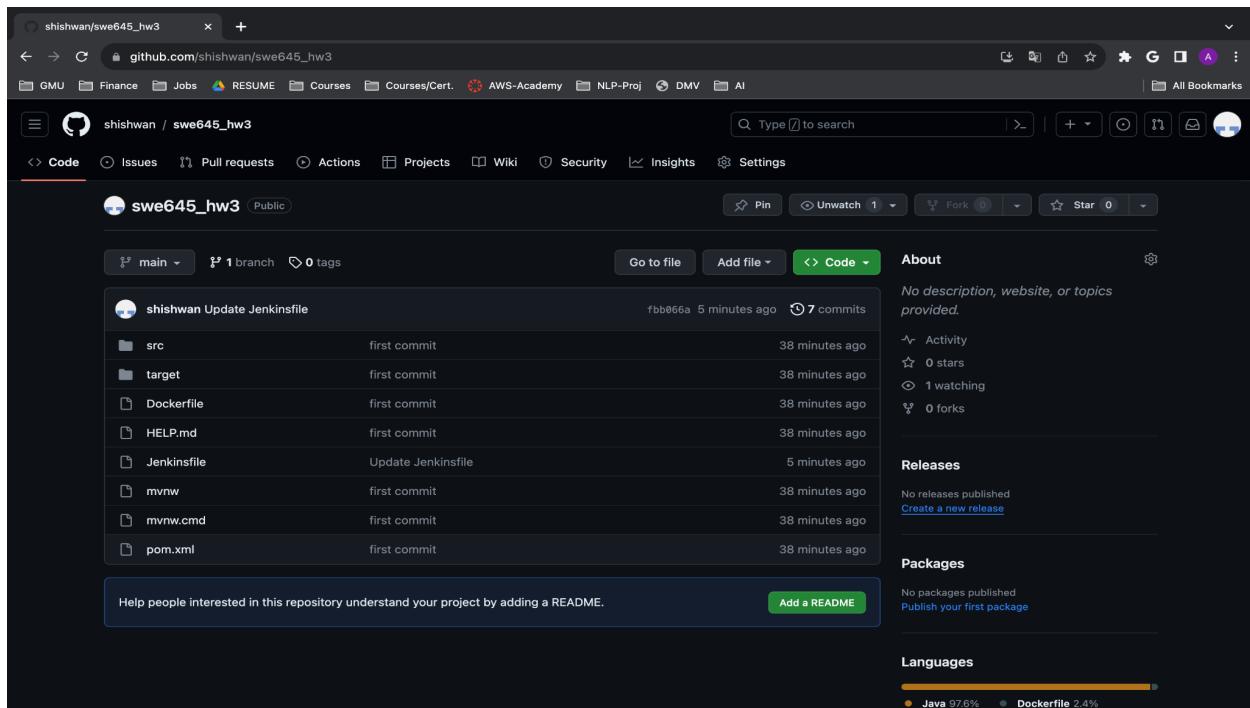
users:
- name: "rancher"
  user:
    token: "kubeconfig-user-cg46f98tlt:xg96rzm2kztz6gfn57bpk5vlftzhbgwt9kk1qlqzd7gmpvb9jx8c7cr"

contexts:
- name: "rancher"
  context:
    user: "rancher"
    cluster: "rancher"

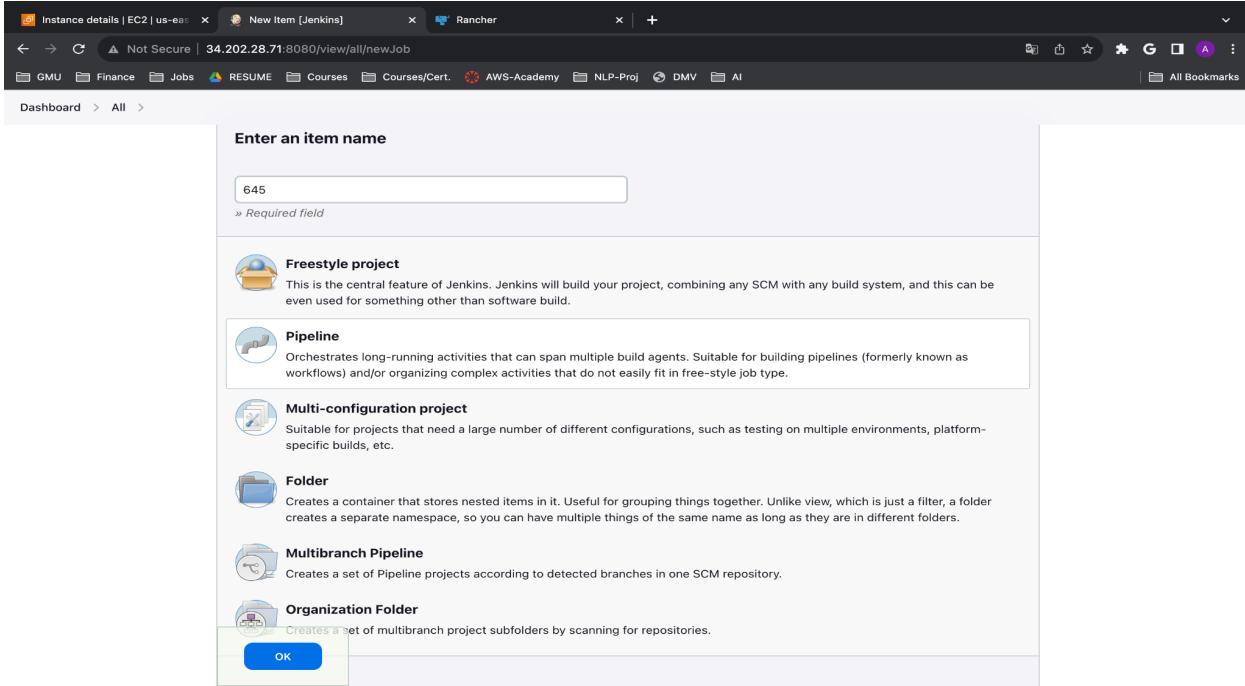
current-context: "rancher"

~
~
~
~
~
```

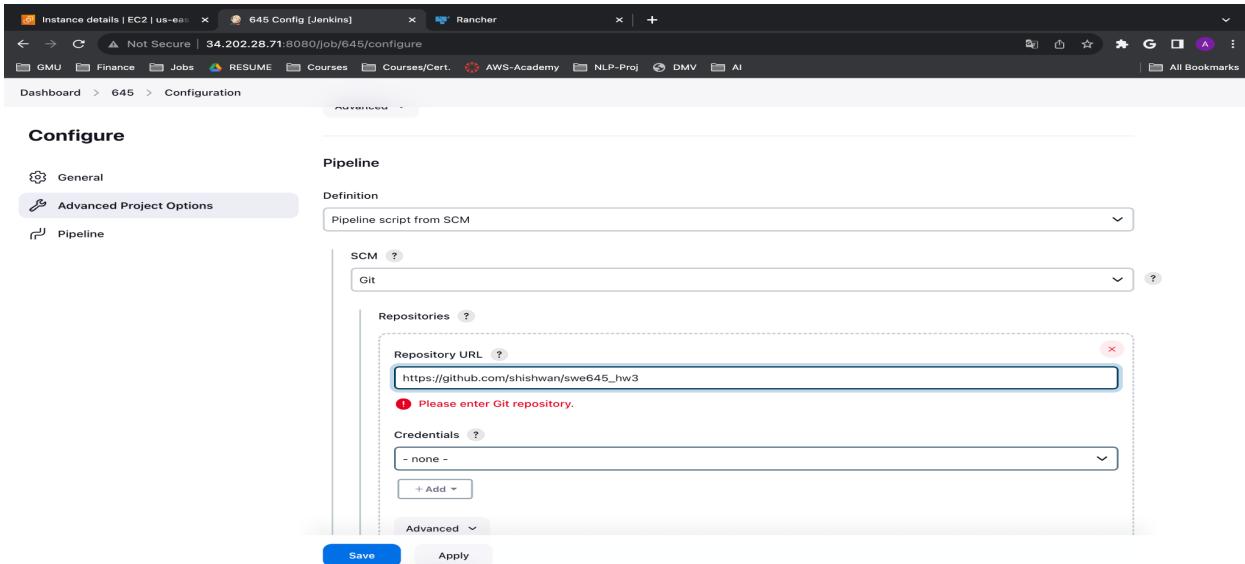
- Push the contents to github https://github.com/a-jagadish/swe_645



- Go to Dashboard and click on a new item, enter a name, select a pipeline and click OK.



- In the pipeline configuration, under pipeline select pipeline script from SCM and git as SCM and provide github repo url.



- Push the code and Jenkinsfile to a GitHub repository.
- Go back to the Jenkins dashboard and click on the pipeline job and click on Build.
- Now to trigger the pipeline.

Status 645

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Rename

Pipeline Syntax

Build History trend ▾

Average stage times: (Average full run time: ~15s)

Declarative: Checkout SCM	Checkout code	BuildJAR	Pushing image to docker	UpdateDeployment
3s	614ms	1s	8s	323ms

#1 Nov 30 20:18 No Changes

Permalinks

[Atom feed for all](#) [Atom feed for failures](#)

REST API Jenkins 2.426.1

- Once the build is successful, we go back to the rancher and open the deployment url and docker. We can confirm that our Jenkins is built successfully by seeing the tags in the docker and the build deployment in the rancher.

ajagadis / **General**

Description
This repository does not have a description

Last pushed: a few seconds ago

Docker commands
To push a new tag to this repository:

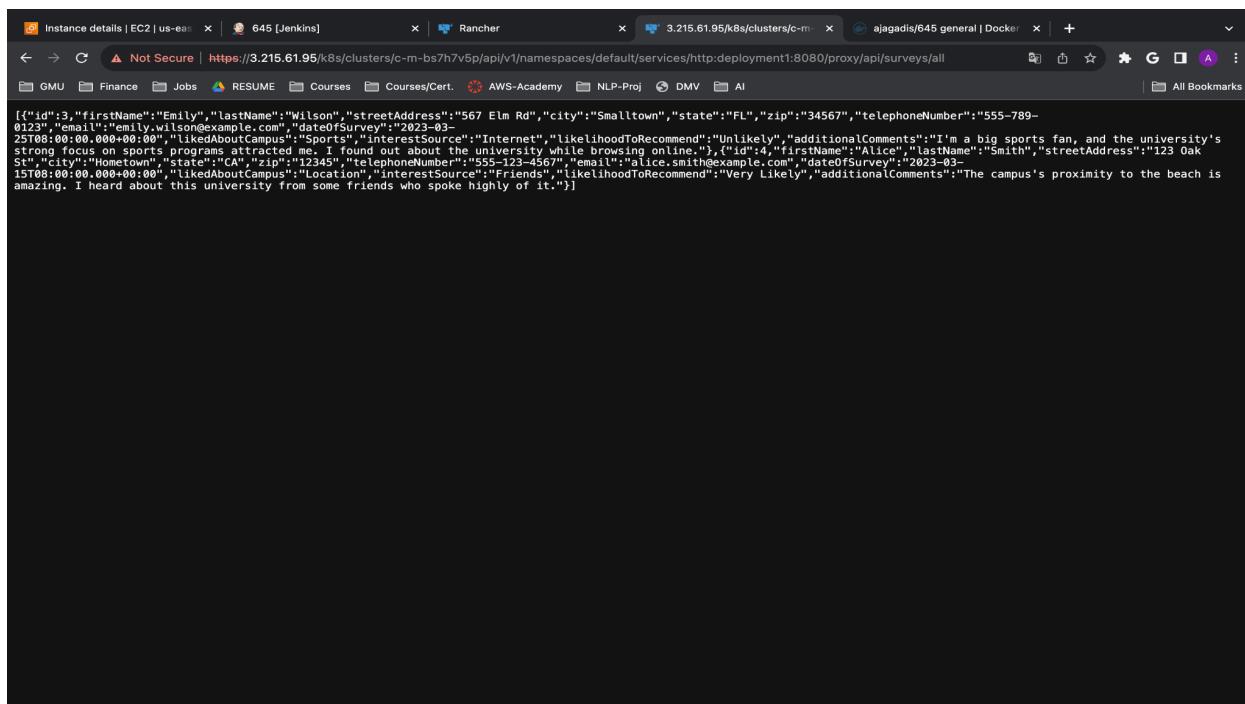
```
docker push ajagadis/645:tagname
```

Tags
This repository is empty. Push some images to it to see them appear here.

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



A screenshot of a web browser window displaying survey data in JSON format. The browser has multiple tabs open, including 'Instance details | EC2 | us-east-1', '645 [Jenkins]', 'Rancher', '3.215.61.95/k8s/clusters/c-m-bs7h7v5p/api/v1/namespaces/default/services/http:deployment1:8080/proxy/api/surveys/all', and 'ajagadis/645 general | Docker'. The main content area shows a single JSON object:

```
[{"id":3,"firstName":"Emily","lastName":"Wilson","streetAddress":"567 Elm Rd","city":"Smalltown","state":"FL","zip":"34567","telephoneNumber":"555-789-0123","email":"emily.wilson@example.com","dateOfSurvey":"2023-03-01","likedAboutCampus":true,"interestSource":"Internet","likelihoodToRecommend":"Unlikely","additionalComments":"I'm a big sports fan, and the university's strong focus on sports programs attracted me. I found out about the university while browsing online."}, {"id":4,"firstName":"Alice","lastName":"Smith","streetAddress":"123 Oak St","city":"Hometown","state":"CA","zip":"12345","telephoneNumber":"555-123-4567","email":"alice.smith@example.com","dateOfSurvey":"2023-03-15T08:00:00.000+00:00","likedAboutCampus":false,"interestSource":"Friends","likelihoodToRecommend":"Very Likely","additionalComments":"The campus's proximity to the beach is amazing. I heard about this university from some friends who spoke highly of it."}]
```