

# **CSCI 5401 Serverless Data Processing**

## **Assignment-2**

**Instructor:** Dr. Saurabh Dey

**Date of Submission:** 10 June 2020

**Submitted By:**

Kase Raviteja (B00823644)



**DALHOUSIE  
UNIVERSITY**

Faculty of Computer Science  
Dalhousie University  
Halifax, Nova Scotia

## Part A

a) Docker container and MySQL setup in CloudSQL

CloudSQL setup in GCP:[1]

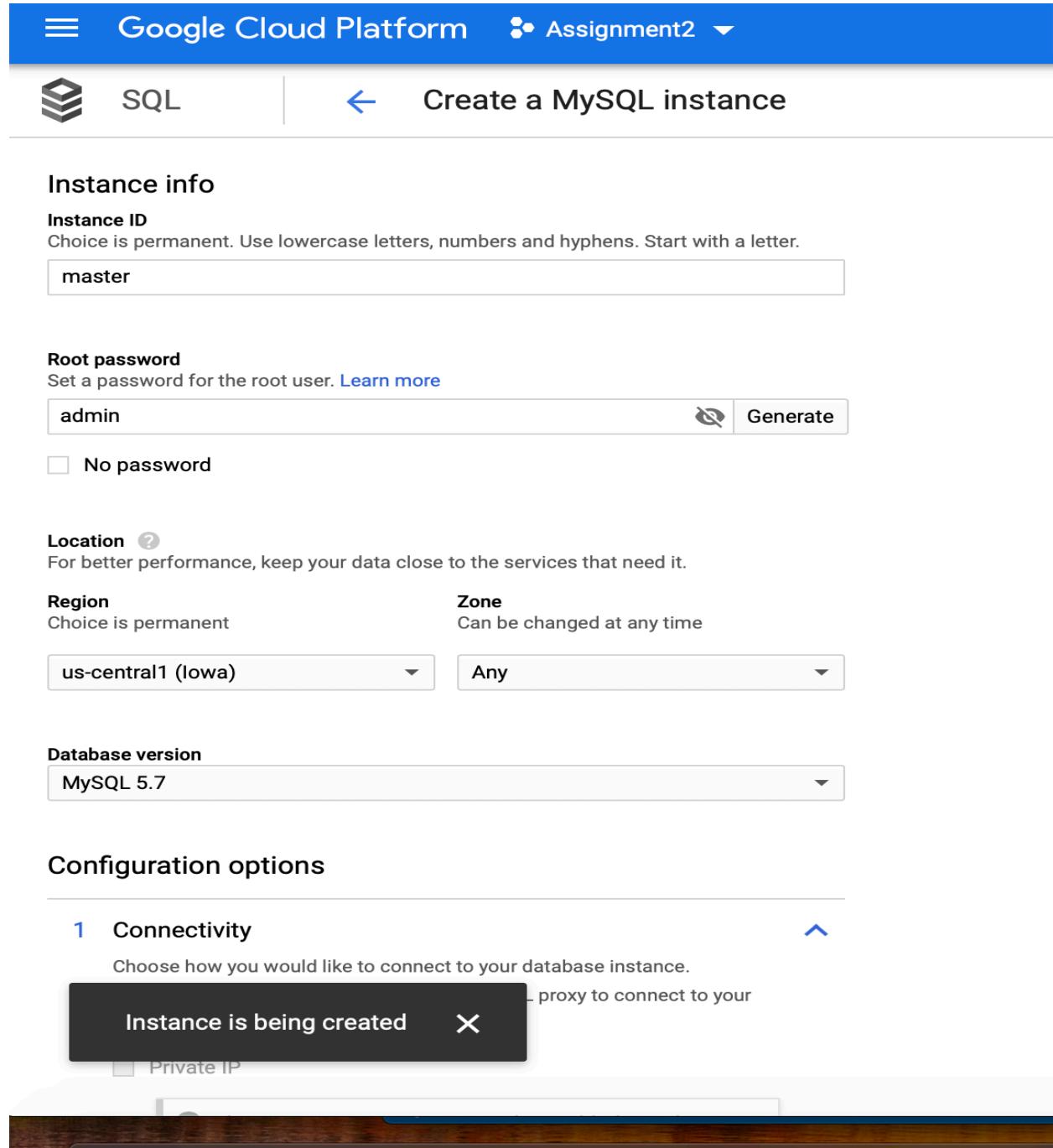


Figure 1: CloudSQL setup

DB creation in CloudSQL

The screenshot shows the MySQL Workbench interface. On the left, there's a sidebar with various options like Overview, Connections, Users, Databases (which is selected and highlighted in blue), Backups, Replicas, and Operations. The main panel is titled "Databases" and shows the "master" instance. It displays a list of MySQL databases, including "information\_schema", "mysql", "performance\_schema", and "sys". A new database, "a2", has just been created and is listed at the top of the user databases. The "a2" row is highlighted with a red border. A "Create database" button is visible above the table.

Name	Character set	Collation	Type
information_schema	utf8	utf8_general_ci	System
a2	utf8	utf8_general_ci	User
mysql	utf8	utf8_general_ci	System
performance_schema	utf8	utf8_general_ci	System
sys	utf8	utf8_general_ci	User

Figure 2:a2 database created

CloudSQL accessed in MySQL workbench:

The screenshot shows the MySQL Workbench "Connections" page. It lists several connections: "Local instance 3306 (auto saved)" (root, localhost:3306), "AWS\_server" (myuser, 3.16.55.248:3306), "Cloud" (user, 54.162.179.234:3306), and "RDS" (admin, kase5410a1.cpxstvfam35l.us-east-1....). One connection, "Google Cloud Sql (auto saved)" (admin, 35.238.119.237:3306), is highlighted with a blue border.

Figure 3: CloudSQL accessed in MySQL

APIs are developed using Spring Boot[2]. Docker images are built locally[3] and tagged to Google Container Registry[4].

### b)Registration Docker image:

```
Last login: Wed Jun 10 15:45:16 on ttys004
/Users/ravitejakase/.zshenv:4: command not found: 'export
ravitejakase@RAVITEJAs ~ % cd Desktop/assignment2
ravitejakase@RAVITEJAs assignment2 % cd registration
ravitejakase@RAVITEJAs registration % gradle clean build

BUILD SUCCESSFUL in 9s
10 actionable tasks: 10 executed
ravitejakase@RAVITEJAs registration % cd build/libs
ravitejakase@RAVITEJAs libs % ls
registration-0.0.1-SNAPSHOT.jar
ravitejakase@RAVITEJAs libs % cd ..
ravitejakase@RAVITEJAs build % cd ..
ravitejakase@RAVITEJAs registration % docker build -t registration .
Sending build context to Docker daemon    254MB
Step 1/6 : FROM openjdk:8-jdk-alpine
--> a3562aa0b991
Step 2/6 : VOLUME /tmp
--> Using cache
--> 1e61d366e687
Step 3/6 : ARG JAR_FILE=../build/libs/*
--> Using cache
--> 6ef72f3769a
Step 4/6 : COPY ${JARFILE} registration-0.0.1-SNAPSHOT.jar
--> ef0e05cf68995
Step 5/6 : EXPOSE 9010
--> Running in cfe1639ca91e
Removing intermediate container cfe1639ca91e
--> 6d06bdd8c3da
Step 6/6 : CMD ["java", "-jar", "registration-0.0.1-SNAPSHOT.jar"]
--> Running in 58bf33b6e47
Removing intermediate container 58bf33b6e47
--> 8ed1fe22f9b8
Successfully built 8ed1fe22f9b8
Successfully tagged registration:latest
ravitejakase@RAVITEJAs registration % docker tag registration gcr.io/assignment2-279422/registration
ravitejakase@RAVITEJAs registration % docker push gcr.io/assignment2-279422/registration
[The push refers to repository [gcr.io/assignment2-279422/registration]
c244c4963673: Pushed
cea9fe1ebef5: Layer already exists
9b9b7f3d56a0: Layer already exists
f1b5933fe4b5: Layer already exists
latest: digest: sha256:c4ffa84c3343f03307b4c47a62b4353fae3ab2ee8ed5bc9fa5bfc9bf07ab size: 1159
ravitejakase@RAVITEJAs registration % ]
```

Figure 4:Registration Docker image

### Registration Code snippet:

@CrossOrigin

@RestController

```
public class RegistrationController {
```

```
    @Autowired
```

```
    private RegistrationService service;
```

```
    @PostMapping("/registerUser")
```

```
    public User registerUser(@RequestBody User user) throws Exception {
```

```
        String tempEmailId = user.getEmailId();
```

```
        if (tempEmailId != null && !"".equals(tempEmailId)) {
```

```
            User userObjUser = service.fetchUserByEmailId(tempEmailId);
```

```
            if (userObjUser != null) {
```

```
                throw new Exception("User already exists");
```

```

    }

}

User userObjUser = null;

userObjUser = service.saveUser(user);

return userObjUser;

}

}

```

### User Registered in Database:

	id	email_id	password	topic	user_name
	5	hay@gmail.com	1234567	AZURE	hayati
	7	ravit@gmail.com	1234567	GCP	ravi
	8	uvan@gmail.com	12345678	GCP	uvan
	HULL	NULL	NULL	NULL	HULL

Figure 5:User registered in Database

### c)Login Docker image:

```

Build an image from a Dockerfile
[ravitejakase@RAVITEJAs login % docker build -t login .
Sending build context to Docker daemon 268.1MB
Step 1/6 : FROM openjdk:8-jdk-alpine
--> a3b62aa0b991
Step 2/6 : VOLUME /tmp
--> Using cache
--> 1e61d366e687
Step 3/6 : ARG JARFILE=./build/libs/*
--> Using cache
--> 6ef72f33769a
Step 4/6 : COPY ${JARFILE} login-0.0.1-SNAPSHOT.jar
--> Using cache
--> 25677ac69478
Step 5/6 : EXPOSE 9010
--> Using cache
--> 170a32749495
Step 6/6 : CMD ["java","-jar","login-0.0.1-SNAPSHOT.jar"]
--> Using cache
--> 8338f420ba39
Successfully built 8338f420ba39
Successfully tagged login:latest
ravitejakase@RAVITEJAs login % docker tag login gcr.io/assignment2-279422/login
ravitejakase@RAVITEJAs login % docker push gcr.io/assignment2-279422/login
[The push refers to repository [gcr.io/assignment2-279422/login]
7e1c818c72d4: Pushed
ceaf9e1ebef5: Layer already exists
9b9b7f3d56a0: Layer already exists
f1b5933fe4b5: Layer already exists
latest: digest: sha256:854d83d29f05a01887966aa3939cd416d710d5125605266e7aaa4e586147e837 size: 1159
ravitejakase@RAVITEJAs login %

```

Figure 6:Login Docker image

### Login Code snippet and Change state to Online:

```
@CrossOrigin
@RestController
public class LoginController {

    @Autowired
    private LoginService service;

    @Autowired
    private UserStatusService userStatusService;

    @PostMapping("/login")
    public User loginUser(@RequestBody User user) throws Exception {
        String tempEmailId = user.getEmailId();
        String tempPassword = user.getPassword();
        User userObjUser = null;
        if (tempEmailId != null && tempPassword != null) {
            userObjUser = service.fetchUserByEmailIdAndPassword(tempEmailId,
tempPassword);
        }
        if (userObjUser == null) {
            throw new Exception("Bad Credentials");
        }
        Date date = new Date();
        long time = date.getTime();
        Timestamp ts = new Timestamp(time);
        UserStatus statusObj = userStatusService.fetchUserByEmailId(tempEmailId);
        if (statusObj == null) {
            UserStatus sUserStatus = new UserStatus();
            sUserStatus.setEmailId(tempEmailId);
            sUserStatus.setStatus("online");
            sUserStatus.setTimestamp(ts);
            userStatusService.saveUserStatus(sUserStatus);
        }
    }
}
```

```
        } else {  
            userStatusService.updateStatus(tempEmailId, "online", ts);  
        }  
        return userObjUser;  
    }  
}
```

d) State changed to Online in Database:

The screenshot shows the MySQL Workbench interface with the database 'a2' selected. The 'Tables' section is open, and the 'user\_status' table is selected. The table has four columns: id, email\_id, status, and timestamp. There are five rows of data. The last row, where id=9 and email\_id=uwan@gmail.com, is highlighted with a red box.

	id	email_id	status	timestamp
▶	2	teja@gmail.com	offline	2020-06-07 03:50:04
▶	4	ravi@gmail.com	online	2020-06-07 03:50:17
▶	6	hav@gmail.com	online	2020-06-07 03:51:10
▶	9	uwan@gmail.com	online	2020-06-10 18:21:13

*Figure 7: State changed to Online*

e) Database contains only two tables and DB scripts are pasted in the submission folder.

f) Stateinfo Docker image:

```
2020-06-10 15:55:16.458 INFO 35932 --- [extShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.

BUILD SUCCESSFUL in 16s
12 actionable tasks: 12 executed
ravitejakase@RAVITEJAs stateInfo % cd build/libs
ravitejakase@RAVITEJAs libs % ls
stateInfo-0.0.1-SNAPSHOT.jar
ravitejakase@RAVITEJAs build % cd ..
ravitejakase@RAVITEJAs build % cd ..
ravitejakase@RAVITEJAs stateInfo % docker build -t stateinfo .
Sending build context to Docker daemon 254.3MB 1kB
Step 1/6 : FROM openjdk:8-jdk-alpine
--> a3562aa0b991
Step 2/6 : VOLUME /tmp
--> Using cache
--> 396dd34016109
Step 3/6 : ARG JAR_FILE=./build/libs/*
--> Using cache
--> 6ef72f33769a
Step 4/6 : COPY ${JAR_FILE} stateInfo-0.0.1-SNAPSHOT.jar
--> 396dd40077e1
Step 5/6 : EXPOSE 9030
--> Running in 5db634016109
Removing intermediate container 5db634016109
> 014a70cd1f50
Step 6/6 : CMD ["java", "-jar", "stateInfo-0.0.1-SNAPSHOT.jar"]
--> Running in 014a70cd1f50
Removing intermediate container 014a70cd1f50
--> c0c5ddb5566d
Successfully built c0c5ddb5566d
Successfully tagged stateinfo:latest
ravitejakase@RAVITEJAs stateInfo % docker tag stateinfo gcr.io/assignment2-279422/stateinfo
ravitejakase@RAVITEJAs stateInfo % docker push gcr.io/assignment2-279422/stateinfo
The push refers to repository [gcr.io/assignment2-279422/stateinfo]
432408d74ca9: Pushed
ceaf9e1ebef5: Layer already exists
9b9b7f3d56a0: Layer already exists
fb1f5933fe4b5: Layer already exists
latest: digest: sha256:41ba02444a20f65b8010c140e7c09bbe2a1b0e595db81da8421ab137055fd379 size: 1159
ravitejakase@RAVITEJAs stateInfo %
```

*Figure 8: Stateinfo docker*

```

@CrossOrigin
@RestController
public class LogoutController {

    @Autowired
    private LoginService service;

    @Autowired
    private UserStatusService userStatusService;

    @PostMapping("/logout/{emailId}")
    public void loginUser(@PathVariable String emailId) {
        Date date = new Date();
        long time = date.getTime();
        Timestamp ts = new Timestamp(time);
        userStatusService.updateStatus(emailId, "offline", ts);
    }

    @GetMapping("/onlineUsers/{emailId}")
    public List<User> fetchAllOnlineUsers(@PathVariable String emailId) {
        List<User> onlineUsers = null;
        onlineUsers = userStatusService.fetchAllStatus("online", emailId);
        return onlineUsers;
    }
}

```

**Stateinfo changed to Offline in Database:**

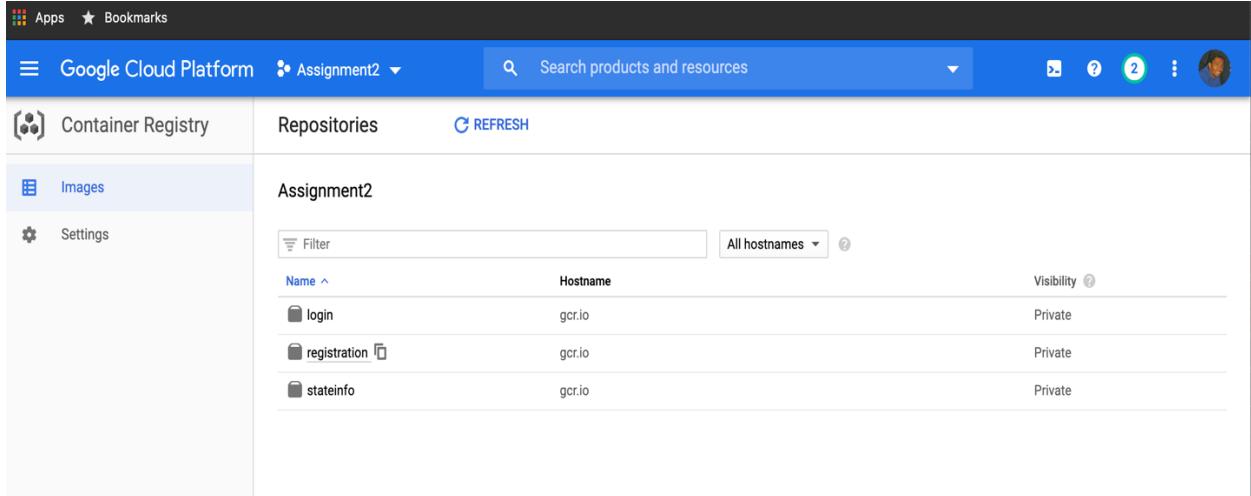
9	uvan@gmail.com	offline	2020-06-10 18:22:54

*Figure 9:Stateinfo changed to Offline in database*

### g) Google Cloud run:

Docker images are tagged in Google Cloud registry, using Google Cloud run images are run in GCP.

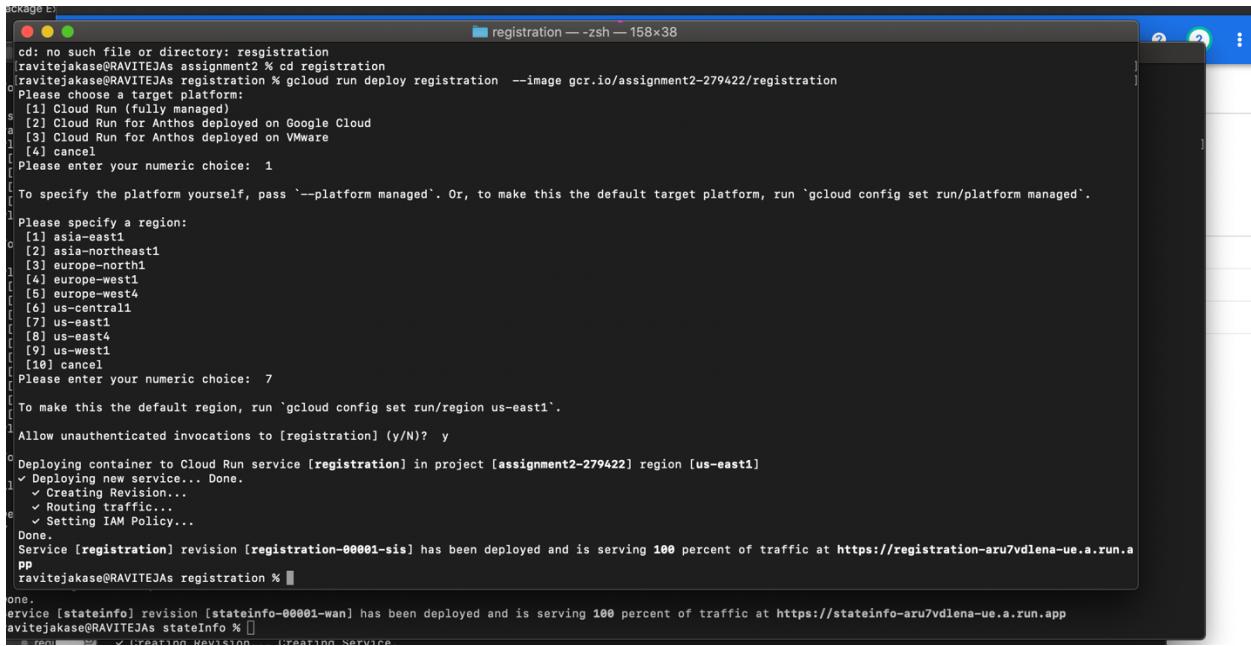
### Images in Google Cloud Registry[4]:



The screenshot shows the Google Cloud Platform Container Registry interface. The left sidebar has 'Container Registry' selected under 'Images'. The main area shows a table of registered images in the 'Assignment2' repository. The columns are 'Name', 'Hostname', and 'Visibility'. There are three entries: 'login' with hostname 'gcr.io' and visibility 'Private', 'registration' with hostname 'gcr.io' and visibility 'Private', and 'stateinfo' with hostname 'gcr.io' and visibility 'Private'. A 'Filter' input field and a 'REFRESH' button are also visible.

Figure 10:Containers registered in GCR

### Registration container running using Google Cloud Run [5]:



```
cd: no such file or directory: registration
ravitejakase@RAVITEJAs assignment2 % cd registration
ravitejakase@RAVITEJAs registration % gcloud run deploy registration --image gcr.io/assignment2-279422/registration
Please choose a target platform:
[1] Cloud Run (fully managed)
[2] Cloud Run for Anthos deployed on Google Cloud
[3] Cloud Run for Anthos deployed on VMware
[4] cancel
Please enter your numeric choice: 1

To specify the platform yourself, pass `--platform managed`. Or, to make this the default target platform, run `gcloud config set run/platform managed`.

Please specify a region:
[1] asia-east1
[2] asia-northeast1
[3] europe-north1
[4] europe-west1
[5] europe-west4
[6] us-central1
[7] us-east1
[8] us-east4
[9] us-west1
[10] cancel
Please enter your numeric choice: 7

To make this the default region, run `gcloud config set run/region us-east1`.
Allow unauthenticated invocations to [registration] (y/N)? y

Deploying container to Cloud Run service [registration] in project [assignment2-279422] region [us-east1]
Creating Revision...
  ✓ Routing traffic...
  ✓ Setting IAM Policy...
Done.
Service [registration] revision [registration-00001-sis] has been deployed and is serving 100 percent of traffic at https://registration-aru7vdlena-ue.a.run.app
pp
ravitejakase@RAVITEJAs registration %

one.
service [stateinfo] revision [stateinfo-00001-wan] has been deployed and is serving 100 percent of traffic at https://stateinfo-aru7vdlena-ue.a.run.app
avitejakase@RAVITEJAs stateInfo %
```

Figure 11:Registration container running using Google Cloud Run

## Login container running using Google Cloud Run [5]:

The screenshot shows a terminal window titled "STORE IT in Backend database (Change +)" with the command "ravitejakase@RAVITEJAs login % gcloud run deploy login \ --image gcr.io/assignment2-279422/login". The terminal lists options for target platform selection and region specification. It then shows the deployment process, including creating a revision, routing traffic, and setting IAM policies. Finally, it confirms the service has been deployed and is serving 100% traffic at a specific URL.

```
ravitejakase@RAVITEJAs login % gcloud run deploy login \
[--image gcr.io/assignment2-279422/login]
Please choose a target platform:
[1] Cloud Run (fully managed)
[2] Cloud Run for Anthos deployed on Google Cloud
[3] Cloud Run for Anthos deployed on VMware
[4] cancel
Please enter your numeric choice: 1

To specify the platform yourself, pass `--platform managed`. Or, to make this the default target platform, run `gcloud config set run/platform managed`.

Please specify a region:
[1] asia-east1
[2] asia-northeast1
[3] europe-north1
[4] europe-west1
[5] europe-west4
[6] us-central1
[7] us-east1
[8] us-east4
[9] us-west1
[10] cancel
Please enter your numeric choice: 7

To make this the default region, run `gcloud config set run/region us-east1`.

Allow unauthenticated invocations to [login] (y/N)? y

Deploying container to Cloud Run service [login] in project [assignment2-279422] region [us-east1]
✓ Deploying new service... Done.
  ✓ Creating Revision... Creating Service.
  ✓ Routing traffic...
  ✓ Setting IAM Policy...
Done.
Service [login] revision [login-00001-hut] has been deployed and is serving 100 percent of traffic at https://login-aru7vdlena-ue.a.run.app
ravitejakase@RAVITEJAs login %
```

Figure 12:Login container running using Google Cloud Run

## Stateinfo container running using Google Cloud Run[5]:

The screenshot shows a terminal window titled "Google Cloud Platform - Assignment2" with the command "ravitejakase@RAVITEJAs stateInfo % gcloud run deploy stateinfo --image gcr.io/assignment2-279422/stateinfo". The terminal lists options for target platform selection and region specification. It then shows the deployment process, including creating a revision, routing traffic, and setting IAM policies. Finally, it confirms the service has been deployed and is serving 100% traffic at a specific URL.

```
To search the help text of gcloud commands, run:
  gcloud help -- SEARCH_TERMS
zsh: command not found: --SEARCH_TERMS
ravitejakase@RAVITEJAs stateInfo % gcloud run deploy stateinfo --image gcr.io/assignment2-279422/stateinfo
Please choose a target platform:
[1] Cloud Run (fully managed)
[2] Cloud Run for Anthos deployed on Google Cloud
[3] Cloud Run for Anthos deployed on VMware
[4] cancel
Please enter your numeric choice: 1

To specify the platform yourself, pass `--platform managed`. Or, to make this the default target platform, run `gcloud config set run/platform managed`.

Please specify a region:
[1] asia-east1
[2] asia-northeast1
[3] europe-north1
[4] europe-west1
[5] europe-west4
[6] us-central1
[7] us-east1
[8] us-east4
[9] us-west1
[10] cancel
Please enter your numeric choice: 7

To make this the default region, run `gcloud config set run/region us-east1`.

Allow unauthenticated invocations to [stateinfo] (y/N)? y

Deploying container to Cloud Run service [stateinfo] in project [assignment2-279422] region [us-east1]
✓ Deploying new service... Done.
  ✓ Creating Revision... Revision deployment finished. Waiting for health check
  : to begin.
  ✓ Routing traffic...
  ✓ Setting IAM Policy...
Done.
Service [stateinfo] revision [stateinfo-00001-wan] has been deployed and is serving 100 percent of traffic at https://stateinfo-aru7vdlena-ue.a.run.app
ravitejakase@RAVITEJAs stateInfo %
```

Figure 13:Stateinfo container running using Google Cloud Run

## Containers running in Google Cloud run[5]:

The screenshot shows the Google Cloud Platform Cloud Run Services page. At the top, there's a navigation bar with 'Google Cloud Platform' and 'Assignment2'. A search bar says 'Search products and resources'. On the right, there are icons for help, notifications (2), and user profile.

The main area has tabs for 'Cloud Run' (selected), 'Services', 'CREATE SERVICE', 'MANAGE CUSTOM DOMAINS', 'COPY', 'DELETE', and 'HIDE INFO PANEL'.

A message states: 'Each Cloud Run service has a unique endpoint and auto-scales deployed containers. [Learn more](#)'.

A 'Filter services' dropdown is open, showing options like 'Name ↑', 'Req/sec', 'Region', 'Authentication', 'Connectivity', and 'Last deployed'.

The table lists three services:

	Name	Req/sec	Region	Authentication	Connectivity	Last deployed
<input type="checkbox"/>	login	0	us-east1	Allow unauthenticated	External	10 Jun 2020, 15:04:09
<input type="checkbox"/>	registration	0	us-east1	Allow unauthenticated	External	10 Jun 2020, 15:09:13
<input type="checkbox"/>	stateinfo	0	us-east1	Allow unauthenticated	External	10 Jun 2020, 15:07:15

To the right, a sidebar shows 'No services selected' with tabs for 'PERMISSIONS' (selected) and 'LABELS'. A message says: 'Please select at least one resource.'

Figure 14:Containers running in Google Cloud run

h) Webpages:

Front end is built in angular. Code is placed in the submission folder. Please install node modules by running “npm install” command for node modules .

**Registration:**

The screenshot shows a registration form. At the top, there are two buttons: 'Register' and 'Login'. The background is blue.

# Register

Fields:

- userName
- EmailId
- Password
- Select Topic
- 

Figure 15:Registration

**Login:**

The screenshot shows a blue header bar with two buttons: "Register" and "Login". Below the header is a section titled "Login". It contains fields for "EmailId" and "Password", each with an input box. At the bottom are two buttons: "Login" and "Register".

*Figure 16:Login*

i) Test cases:

Test Case Id	Test Case Scenario	Test steps	Test Data	Expected Results	Actual Result	Pass/Fail
TU01	Registration	Click on Register	Enter User name, Email, Password, Select topic from drop down	After entering the details, and click on register, if registration is successful should navigate to Login page	As expected	Pass
TU02	Login	Click on Login	Enter email and password	After click of login button, user state should be changed to online	As expected	Pass
TU03	Fetch online users	After clicking Login	No Input	Should fetch all other online users	As expected	Pass
T004	Logout	Click on Logout	No Input	User state must be	As expected	Pass

				changed to Offline		
--	--	--	--	-----------------------	--	--

### TU01 (Registration)

---

[Register](#)   
 [Login](#)

## Register

userName   
 EmailId   
 Password   
 Select Topic   
[Register](#) [Cancel](#)

Figure 17:Registration

### TU02 (Login):

[Register](#)   
 [Login](#)

## Login

EmailId   
 Password   
[Login](#) [Register](#)

Figure 18:Login

### TU03 (Fetch Online Users):



Hi, uvan you are logged in

[Logout](#)

Here are the Other Users who are online

- tej
- hayati

Figure 19: Fetch Online Users

#### TU04 (Logout):

9	uvan@gmail.com	offline	2020-06-10 18:22:54	
---	----------------	---------	---------------------	--

Figure 20: After Clicking Logout User state changed to offline

#### j) Summary:

##### Google Cloud run:[7]

Google Cloud runs abstracts the infrastructure management and auto-scaling and it enables us to run containers written in any language or to run containers present in Google Cloud registry. Using Google Cloud run app development and deploying is easier.

##### Google Container Registry:[8]

GCR provides the functionality to place all docker images. From GCR, images can be run using GKE or the in-app engine or in Google Cloud run. Google Cloud runs directly works with the Docker command line. Managing container images is easy and it can be done through the console.

##### Docker Containers:[9]

Docker containers are stand-alone packages, which can be built locally and run anywhere irrespective of hardware specifications. Docker containers follow works as built once and run anywhere. Docker isolates software requirements and ensures it runs even though there are differences concerning operating systems.

##### Usage of technologies in the application built:

Dockerfiles are created for each API. Docker images are built locally. Such built images are tagged to the Google Cloud Registry. To access G cloud locally, CloudSDK is installed and configured to the project as default. Tagged images are pushed to the Google Cloud Registry. Using Google Cloud run, container images are run from the registered images in the Google Container registry. Endpoints from Google Cloud are utilized in the angular application developed.

## Part B

### a) Chatbot for Delivery and Take away are created:

#### Delivery

The screenshot shows the configuration for the 'DeliveryFood' chatbot. Under 'Sample utterances', there are two examples: 'e.g. I would like to book a flight.' and 'I want to place an order for pickup'. Under 'Lambda initialization and validation', there is an unchecked checkbox for 'Initialization and validation code hook'.

Figure 21:Delivery Food Chatbot

The screenshot shows the configuration for delivery slots. It lists five slots: 'PizzaTypes' (Priority 1, RequiredName checked, Slot type Pizza..., Version 2, Prompt 'What do you want?', Settings gear and delete), 'Quantity' (Priority 2, RequiredName checked, Slot type AMAZ..., Version Built-in, Prompt 'How many?', Settings gear and delete), 'Address' (Priority 3, RequiredName checked, Slot type AMAZ..., Version Built-in, Prompt 'Where do you want?', Settings gear and delete), 'DeliveryDate' (Priority 4, RequiredName checked, Slot type AMAZ..., Version Built-in, Prompt 'When do you want?', Settings gear and delete), and 'DeliveryTime' (Priority 5, RequiredName checked, Slot type AMAZ..., Version Built-in, Prompt 'At what time do you want?', Settings gear and delete).

Figure 22:Delivery details

The screenshot shows the configuration for a confirmation prompt. It includes a checked checkbox for 'Confirmation prompt', a 'Confirm' section with the prompt 'You have ordered {Quantity} {PizzaTypes}. You', and a 'Cancel (if the user says "no")' section with the prompt 'Okay. Your order won't be placed'.

Figure 23:Confirmation Prompt

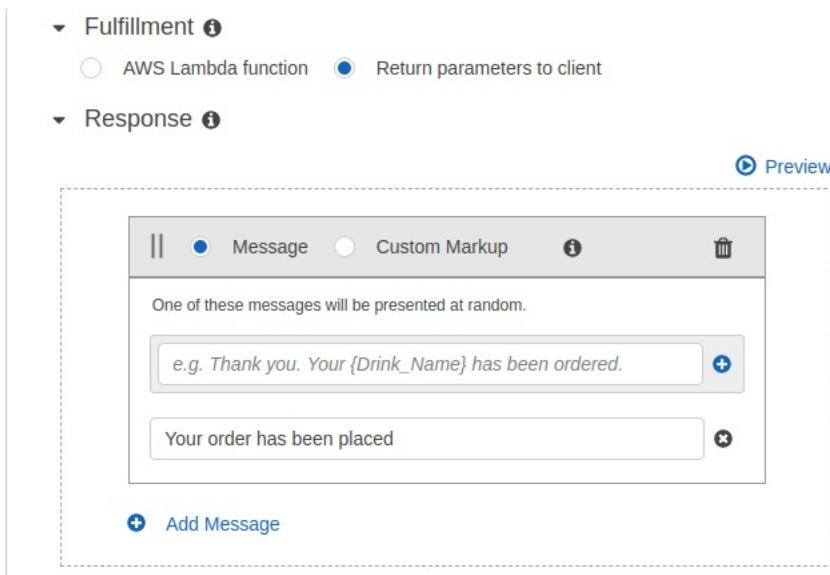


Figure 24:Response builder

## Pickup

PickupFood Latest ▾

### Sample utterances

e.g. I would like to book a flight.	<input type="button" value="+"/>
I want to place an order for delivery	<input type="button" value="x"/>

Figure 25:Pickup Food

### Lambda initialization and validation

Initialization and validation code hook

### Slots

Priority	RequiredName	Slot type	Version	Prompt	Settings
	e.g. Location	e.g. A...		e.g. What city?	<input type="button" value="+"/>
1.	✓	PickupTime	AMAZ...	Built-in When are you co	<input type="button" value="⚙"/> <input type="button" value="x"/>
2.	✓	PizzaTypes	Pizza...	2 ▾ What do you war	<input type="button" value="⚙"/> <input type="button" value="x"/>
3.	✓	Quantity	AMAZ...	Built-in How many?	<input type="button" value="⚙"/> <input type="button" value="x"/>

Figure 26:Pickup times

▼ Confirmation prompt ⓘ

Confirmation prompt

Confirm

You have ordered {Quantity} {PizzaTypes} and ⚙️

Cancel (if the user says "no")

Okay,your order won't be placed ⚙️

▼ Fulfillment ⓘ

AWS Lambda function  Return parameters to client

▼ Response ⓘ

Figure 27:Confirmation Prompt

▼ Response ⓘ

▶ Preview

|| ● Message ● Custom Markup  ⓘ ✖

One of these messages will be presented at random.

e.g. Thank you. Your {Drink\_Name} has been ordered. + ⚙️

Your order has been placed successfully ✖

+ ⚙️ Add Message

Figure 28:Ordered Message

## b) Pizza types

Edit slot type

PizzaTypes Latest ▾  
e.g. Available car types

Slot Resolution

Expand Values ⓘ  Restrict to Slot values and Synonyms ⓘ

Value ⓘ

e.g. Small	Enter Synonym
Press Tab to add a synonym	
Pepperoni Pizza	Enter Synonym <span style="border: 1px solid #ccc; padding: 2px;">✖</span>
Veg Pizza	Enter Synonym <span style="border: 1px solid #ccc; padding: 2px;">✖</span>
Cheese Pizza	Enter Synonym <span style="border: 1px solid #ccc; padding: 2px;">✖</span>

Cancel Save slot type Add slot to intent

Figure 29:Pizza types

c) Chatbot accepts delivery or pickup based on the user input.

**d)Delivery Testing:**

> Test bot (Latest)  Ready. Build complete.

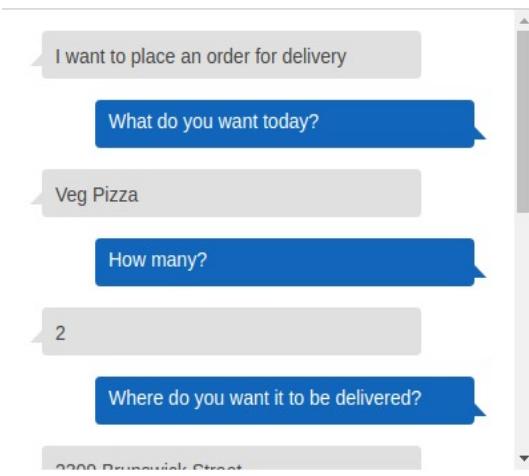


Figure 30:Test-1

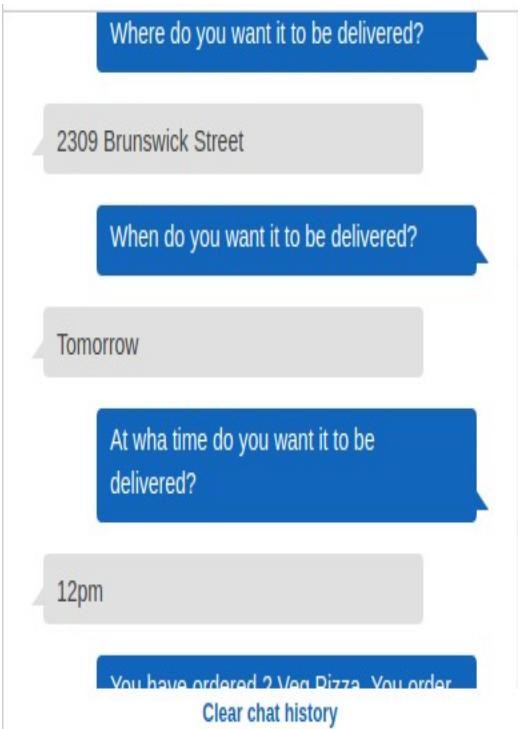


Figure 31:Test-2

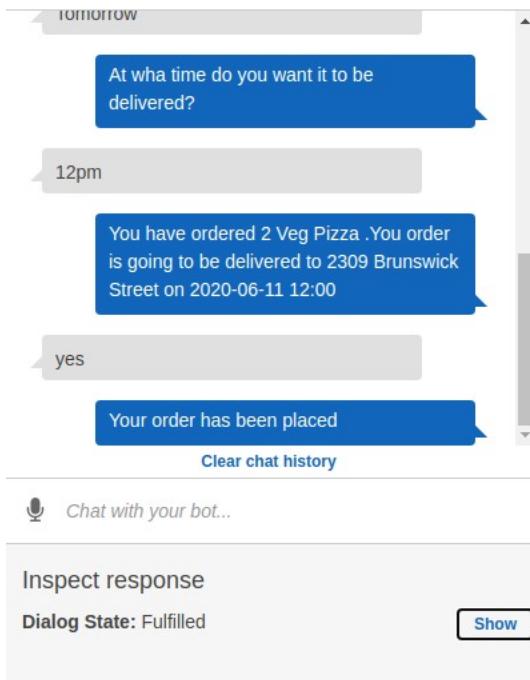


Figure 32:Test-3

### e) Take away testing:

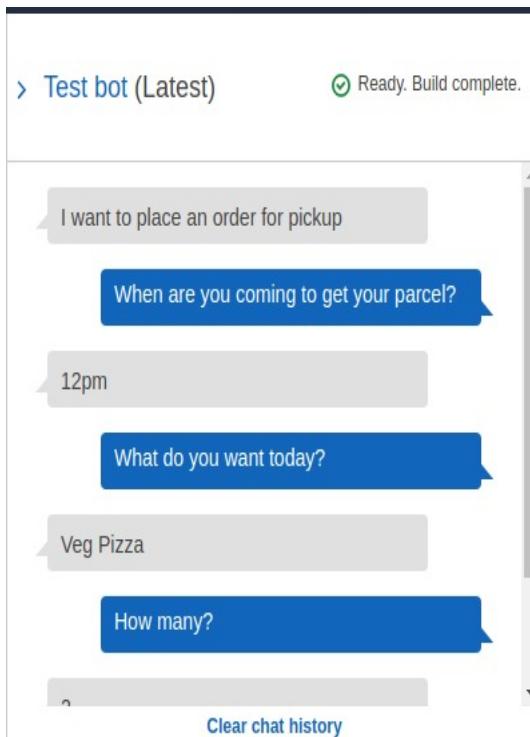


Figure 33:Take away test-1

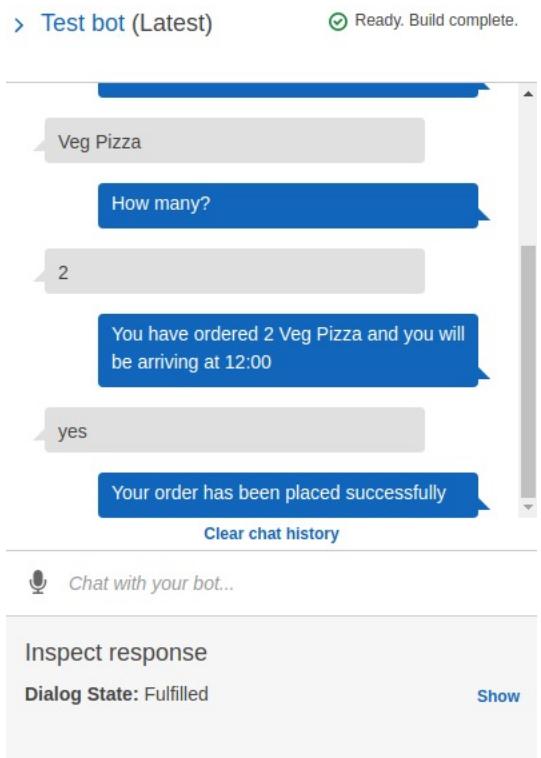


Figure 34:Take away testing -2

### Operation Performed:[10]

Amazon Lex is used to building two chatbots, Pickup and delivery for Pizza orders. For each order type, utterances are created and slots are created. For pickup type pickup time, pizza type and quantities are created. For delivery type pizza type, address, quantity, delivery time and data are created. Confirmation message for an order is created for both the types and similarly response messages are created for final orders. Chatbots are tested successfully.

## References:

- [1] “Quickstart for Cloud SQL for MySQL | Google Cloud,” *Google Cloud*, 2020. [Online]. Available: <https://cloud.google.com/sql/docs/mysql/quickstart>. [Accessed: 03-Jun-2020].
- [2] D. Kabii, “Developing a RESTful API With Spring Boot, Gradle, Groovy, JPA and MySQL in an hour,” *Medium*, 29-Jul-2019. [Online]. Available: <https://medium.com/@david.kabii/developing-a-restful-api-with-spring-boot-gradle-groovy-jpa-and-mysql-in-2-hours-hour-1-a2066a141152>. [Accessed: 04-Jun-2020].
- [3] Sairam Krish, “Docker for Spring boot | Gradle | Java micro service,” *Medium*, 22-May-2018. [Online]. Available: <https://medium.com/@sairamkrish/docker-for-spring-boot-gradle-java-micro-service-done-the-right-way-2f46231dbc06>. [Accessed: 06-Jun-2020].
- [4] “Pushing and pulling images | Container Registry Documentation,” *Google Cloud*, 2020. [Online]. Available: <https://cloud.google.com/container-registry/docs/pushing-and-pulling>. [Accessed: 06-Jun-2020].
- [5] “Quickstart: Build and Deploy | Cloud Run Documentation | Google Cloud,” *Google Cloud*, 2020. [Online]. Available: <https://cloud.google.com/run/docs/quickstarts/build-and-deploy#java>. [Accessed: 06-Jun-2020].
- [6] JavaInUse, “Angular 7 + Spring Boot Login Example,” *YouTube*. 2020.
- [7] “Cloud Run Documentation | Google Cloud,” *Google Cloud*, 2020. [Online]. Available: <https://cloud.google.com/run/docs>. [Accessed: 07-Jun-2020].
- [8] “Container Registry | Google Cloud,” *Google Cloud*, 2020. [Online]. Available: <https://cloud.google.com/container-registry>. [Accessed: 08-Jun-2020].
- [9] “What is a Container? | Docker,” *Docker*, 2013. [Online]. Available: <https://www.docker.com/resources/what-container>. [Accessed: 08-Jun-2020].
- [10] “Module 3 - May 25, 2020 - CSCI5410 - Serverless Data Processing (Sec 1) - 2020 Summer,” *Brightspace.com*, 2020. [Online]. Available: <https://dal.brightspace.com/d2l/le/content/124063/viewContent/1706936/View>. [Accessed: 09-Jun-2020].