

SWE645 – HW3: Build survey form using Angular and Spring Boot and setup Kubernetes cluster using Rancher

Team Member: Aniket Pandey, Siddhanth Kalyanpur, Tanush Singh

Step 1: Write POST and GET api using Spring Tool Suite and connect to SQL

- Following are the two APIs

```
SurveyApplication.java  SurveyService.java  SurveyController.java  application.properties  Survey.java
1  package com.survey.app.controller;
2
3  import java.util.List;
16
17  @RestController
18  @RequestMapping("/api/v1.0")
19  public class SurveyController {
20
21      private SurveyInterface surveyService;
22
23      public SurveyController(SurveyInterface surveyInterface) {
24          super();
25          this.surveyService = surveyInterface;
26      }
27
28      @PostMapping("/surveys")
29      public ResponseEntity<Survey> saveSurvey(@RequestBody Survey survey) {
30          return new ResponseEntity<Survey>(surveyService.saveSurvey(survey), HttpStatus.CREATED);
31      }
32
33      @GetMapping("/surveys")
34      public List<Survey> getAllSurveys(){
35          return surveyService.getAllSurveys();
36      }
37
38  }
39
```

- GET <https://{baseurl}/api/v1.0/surveys>
- POST **Error! Hyperlink reference not valid.**

Body

```
{
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@mail.com",
  "address": "9155 Walnut St",
  "city": "Bentonville",
  "state": "AR",
  "zip": "22031",
  "telephone": "8188884114",
  "dateOfSurvey": "04-28-2022",
  "recommendation": "Very Likely",
  "interest": "Friends",
  "likings": "Students"
}
```

- The Data Model looks like the following fields

```

SurveyApplication.java  SurveyService.java  SurveyController.java  application.properties  Survey.java
1 package com.survey.app.model;
2
3 import javax.persistence.Column;
4
11 @Data
12 @Entity
13 @Table(name="survey")
14 public class Survey {
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     private long id;
18
19     @Column(name = "first_name", nullable = false)
20     private String firstName;
21
22     @Column(name = "last_name", nullable = false)
23     private String lastName;
24
25     @Column(name = "email", nullable = false)
26     private String email;
27
28     @Column(name = "address", nullable = false)
29     private String address;
30
31     @Column(name = "city", nullable = false)
32     private String city;
33
34     @Column(name = "state", nullable = false)
35     private String state;
36
37     @Column(name = "zip", nullable = false)
38     private String zip;
39
40     @Column(name = "telephone", nullable = false)
41     private String telephone;
42
43     @Column(name = "survey_date", nullable = false)
44     private String dateOfSurvey;
45
46     @Column(name = "recommendation")
47     private String recommendation;
48
49     @Column(name = "interest")
50     private String interest;
51
52     @Column(name = "likings")
53     private String likings;
54
55

```

- Write the API and create MySQL database using AWS RDS
- Copy the database endpoint, port number, username and password
- Using JDBC connect to database and write the following lines in application.properties

```

SurveyApplication.java  SurveyService.java  SurveyController.java  *application.properties  Survey.java
1 spring.datasource.url=jdbc:mysql://survey-database.cxltcu4iezvy.us-east-1.rds.amazonaws.com:3306/survey
2 spring.datasource.username=#####
3 spring.datasource.password=#####
4
5 # Hibernate properties
6 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
7
8 # Create, Create-Drop
9 spring.jpa.hibernate.ddl-auto=update

```

- The api can be tested using Postman
- GET API

GET EC2-Save-API

Survey / EC2-Save-API

GET http://18.233.148.169:30759/api/v1.0/surveys

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Headers 6 hidden

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (5) Test Results

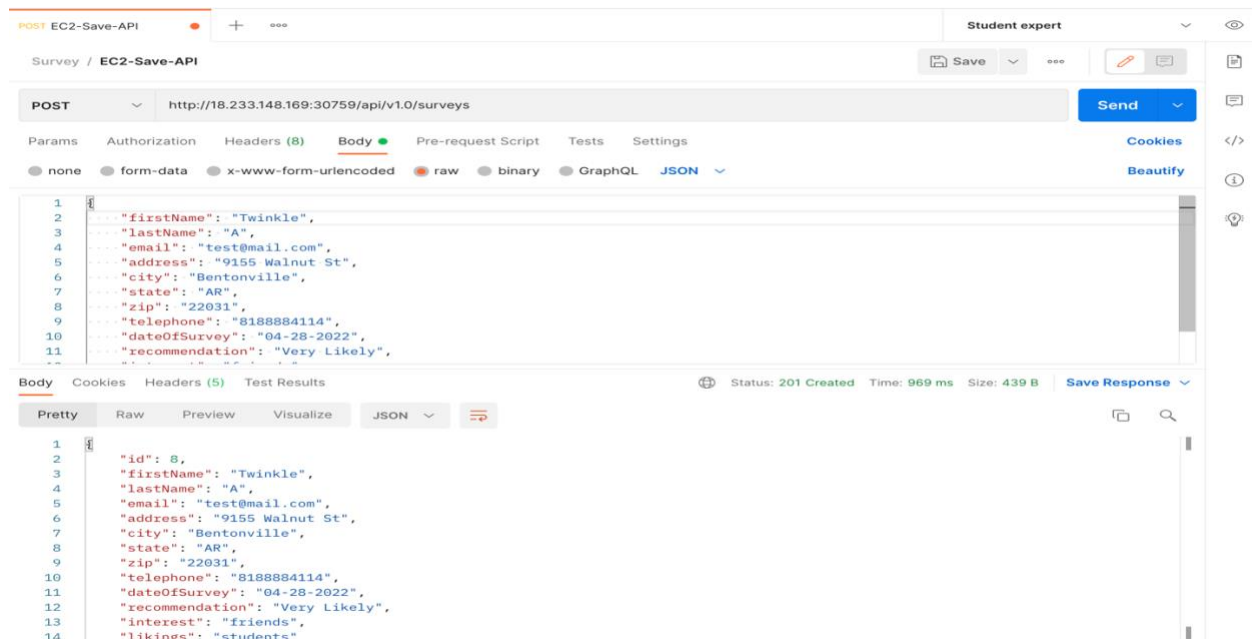
Status: 200 OK Time: 53 ms Size: 2.03 KB

```

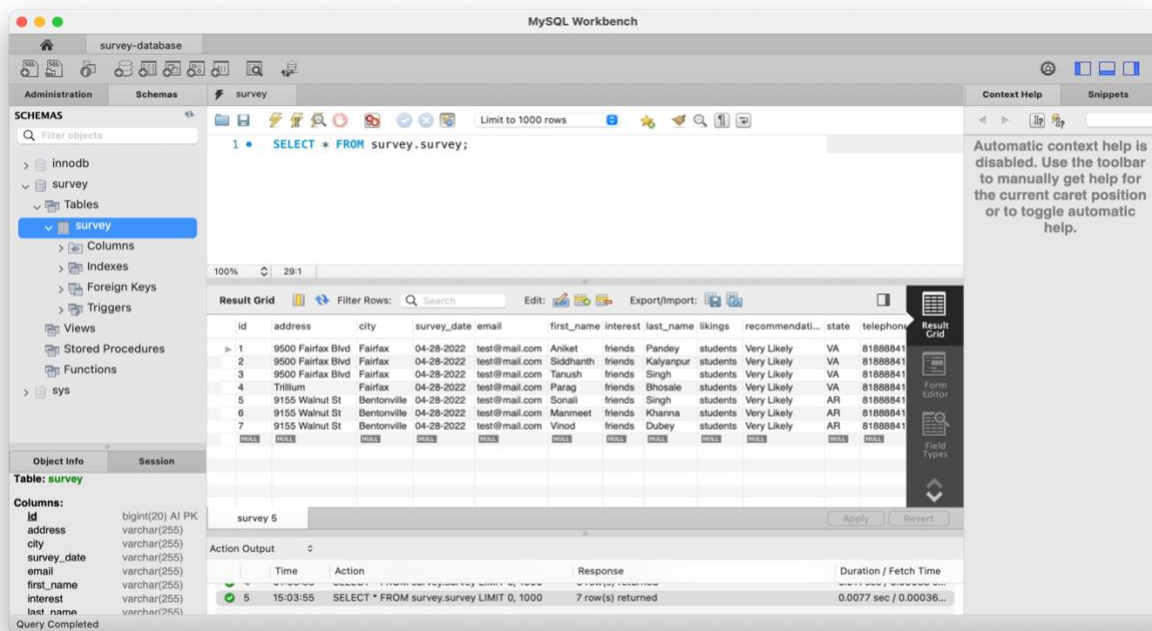
1 {
2   "id": 1,
3   "firstName": "Aniket",
4   "lastName": "Pandey",
5   "email": "test@mail.com",
6   "address": "9500 Fairfax Blvd",
7   "city": "Fairfax",
8   "state": "VA",
9   "zip": "22031",
10  "telephone": "8188884114",
11  "dateOfSurvey": "04-28-2022",
12  "recommendation": "Very Likely",
13  "interest": "Friends",
14  "likings": "students"
15 },
16 {
17   "id": 2,

```

- POST API



- The Database record can be viewed using MySQLWorkbench



- Now right click on the pom.xml file and build inside the Goal on the next step on build write clean build and the continue with the JAR packaging. The JAR file will be present inside target folder by default.

Step 2: Build Docker image and containerize survey api using Docker

- Write Dockerfile as below

```
Dockerfile X
infra > Dockerfile > ...
1 FROM openjdk:17-jdk
2
3 LABEL author.name="Aniket Pandey"
4
5 EXPOSE 8080
6
7 ARG JAR_FILE=survey/target/survey-0.0.1-SNAPSHOT.jar
8
9 COPY ${JAR_FILE} .
10
11 CMD [ "java", "-jar", "/survey-0.0.1-SNAPSHOT.jar"]
```

- Build docker image with the below command. My Dockerfile is inside infra folder and therefore I've specified path as infra/Dockerfile. As a standard practice all the infrastructure setup related files should be kept inside infra folder for easy segregation.
docker build -f infra/Dockerfile --tag survey-api:amd64-v1.0 .
- Tag the docker image and push it on dockerhub using the below command
docker tag survey-api:amd64-v1.0 aniket414/survey-api:amd64-v1.0
docker push aniket414/survey-api:amd64-v1.0

Step 3: Create three EC2 instance, two for deployment and the other for Rancher setup

- Select the Ubuntu ami from AWS marketplace and create one instances for Rancher and select Amazon Linux 2 ami and create instance for deployment of rest api and ui.
- Allow inbound traffic using security group from following ports

Inbound rules (9)								Manage tags	Edit inbound rules
Filter security group rules								< 1 >	
Security group rule...	IP version	Type	Protocol	Port range	Source				
sgr-0225625f9a78f1e54	IPv4	HTTP	TCP	80	0.0.0.0/0				
sgr-04640b85a1c4f2a9e	IPv6	Custom TCP	TCP	8080	::/0				
sgr-06d9bba398e7fabd4	IPv4	All traffic	All	All	0.0.0.0/0				
sgr-024dd4f67097e3ba3	IPv4	HTTPS	TCP	443	0.0.0.0/0				
sgr-0246fd54e55a3a1e6	IPv6	HTTP	TCP	80	::/0				
sgr-02bd1b51428929...	IPv6	HTTPS	TCP	443	::/0				
sgr-021b81b98f23fd7f6	IPv4	SSH	TCP	22	0.0.0.0/0				
sgr-0365140e4d19cf91b	IPv4	Custom TCP	TCP	8080	0.0.0.0/0				
sgr-0d96147c2f4eb03f3	IPv6	All traffic	All	All	::/0				

Step 4: Setup Rancher on one of the instance

- SSH into the instance using pem file
`ssh -i key-value.pem ubuntu@public-dns`
- Install docker after updating
`sudo apt-get update`
`sudo apt install docker.io`
- Install Rancher using the following command
`sudo docker run --privileged -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher`
- After some time open instance public dns which will display Rancher login page where we'll setup username and password
- Login inside and click on create cluster
- Choose custom option from the list of available of available choices and click next
- Enter the cluster name and leave everything as default and click on next
- In the next section select control plane, etcd, and worker and copy the command generated

Cluster Options

Customize Node Run Command

Editing node options will update the command you will run on your existing machines

1

Node Options

Choose what roles the node will have in the cluster.

Node Role

☒ etcd

☒ Control Plane

☒ Worker

Show advanced options

2

Run this command on one or more existing machines already running a supported version of Docker.

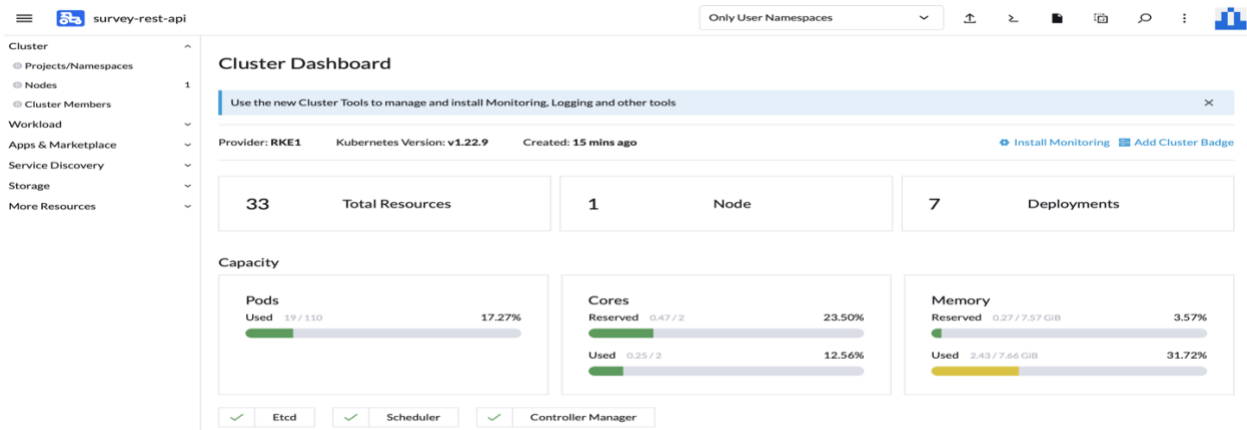
```
sudo docker run -d --privileged --restart=unless-stopped --net=host -v /etc/kubernetes:/etc/kubernetes -v /var/run:/var/run rancher/rancher-agent:v2.6.4 --server https://ec2-54-227-210-92.compute-1.amazonaws.com --token 7msdjxfp7gicfxhhxvp8bt9rwsftq6zbn5t5ttclfmqgfcz4xdjcjv --ca-checksum d331501ac4fac75407cf5f13372487c001b5fb9775d73b6f27a0f663378b812d --etcd --controlplane --worker
```

Done

Step 5: Setting up another instance for api orchestration

- SSH into the instance using pem file
`ssh -i key-value.pem ec2-user@public-dns`
- Install docker after updating
`sudo yum update`
`sudo yum install docker`
`sudo service docker start`

- Run the above copied command and after some time go to the Rancher page and you should see cluster provisioning has started.



- Condition of the cluster is active and healthy

The screenshot shows the 'Clusters' management page in Rancher. It displays a list of clusters with their current status and configuration details.

State	Name	Version	Provider	Machines	Age	Actions
Active	local	v1.22.7+k3s1	Local K3s	1	14 hours	Explore
Active	survey-rest-api	v1.22.9	RKE	1	15 mins	Explore

The screenshot shows the detailed configuration page for the 'survey-rest-api' cluster. It provides step-by-step instructions for node registration and cluster configuration.

Cluster: survey-rest-api (Active)

Namespace: fleet-default | Age: 16 mins

Provisioner: RKE
Labels: provider:cattle.io:rke

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.

```
sudo docker run -d --privileged --restart=unless-stopped --net=host -v /etc/kubernetes:/etc/kubernetes -v /var/run:/var/run rancher/rancher-agent:v2.6.4 --server https://34.238.244.49 --token l7pdnd2tf7xv2kgn1jqsbpfs5qx4r99n2w6jw115241gfnj9xtn6 -ca-checksum 61423c9bdb67e20f8700b8e87cd742226828401ee469599b1e16ca2c6ff43b6b --etcd --controlplane --worker
```

Step 6: Deploy the survey-api docker image on cluster on setup three pods running at all time

- Enter the cluster just created on Rancher and click on Deployment under the Workload section. Alternatively you can also copy the kubeconfig file and if you have kubectl setup done on your local then paste the kubeconfig file content in .kube/config and you can easily access the cluster and deployment using kubectl from your local.
- Enter the replica set as mentioned in assignment i.e. 3
- Enter the docker image uri
- Setup NodePort as a service

Deployment: survey-deployment Active

Namespace: default Age: 1.1 hours

Detail Config **YAML** ⋮

Namespace
default

Name
survey-deployment

Description
Any text you want that better describes this resource

Replicas
3

General

Health Check

Labels & Annotations

Networking

Node Scheduling

Pod Scheduling

Resources

Scaling and Upgrade Policy

Security Context

Storage

General

Container Name
container-0

Init Container
Standard Container

Image

Container Image
aniket414/survey-api:amd64-v1.0

Pull Policy
Always

Pull Secrets

Ports

Service Type
Node Port

Name
survey-svc

Private Container Port
8080

Protocol
TCP

Listening Port
30759

- Click on done and you should see your pods coming up.

survey-rest-api Only User Namespaces

Cluster Workload CronJobs DaemonSets **Deployments** 1 Jobs 0 StatefulSets 0 Pods 3 Apps & Marketplace Service Discovery Storage More Resources

Deployments

Create

Redeploy Download YAML Delete

State	Name	Image	Endpoints	Ready	Up-to-date	Available	Age	Health
Namespace: default								
Active	survey-deployment	aniket414/survey-api:amd64-v1.0	30759/TCP	3/3	3	3	7 mins	

Deployment: survey-deployment Active

Namespace: default Age: 7 mins

Image: aniket414/survey-api:amd64-v1.0
Annotations: [Show 1 annotation](#)

Pods by State

3 Running

State	Name	Node	Image
Running	survey-deployment-7b4489b958-b5gns	ip-172-31-89-70	aniket414/survey-api:amd64-v1.0
Running	survey-deployment-7b4489b958-jlnsh	ip-172-31-89-70	aniket414/survey-api:amd64-v1.0
Running	survey-deployment-7b4489b958-jwh5m	ip-172-31-89-70	aniket414/survey-api:amd64-v1.0

- You can also verify the status from local using kubectl command
- Following are the screenshot of service, node, deployment, and pods

```

survey — -zsh — 93x24
aniket@Anikets-MacBook-Pro survey % kubectl get nodes
NAME                STATUS    ROLES                  AGE      VERSION
ip-172-31-89-70     Ready    controlplane,etcd,worker 2m51s    v1.22.9
aniket@Anikets-MacBook-Pro survey % kubectl get pods
No resources found in default namespace.
aniket@Anikets-MacBook-Pro survey % kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
survey-deployment-7b4489b958-b5gns  1/1     Running   0           39s
survey-deployment-7b4489b958-jlnsh  1/1     Running   0           39s
survey-deployment-7b4489b958-jwh5m  1/1     Running   0           39s
aniket@Anikets-MacBook-Pro survey % kubectl get deployment
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
survey-deployment   3/3     3             3           50s
aniket@Anikets-MacBook-Pro survey % kubectl get service
NAME                                TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
kubernetes                        ClusterIP   10.43.0.1     <none>        443/TCP          11m
survey-deployment                 ClusterIP   10.43.165.33  <none>        8080/TCP          6m26s
survey-deployment-nodeport        NodePort    10.43.160.199 <none>        8080:30759/TCP   6m26s
aniket@Anikets-MacBook-Pro survey %

```

Step 7: Create Angular application

- Create a service call and make GET and POST calls to the api using HttpClient
- The url should be the url of the EC2 instance on which the api is deployed. Check the image below with baseUrl.


```

import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Survey } from './survey';

@Injectable({
  providedIn: 'root'
})
export class SurveyService {

  private baseUrl = "http://54.91.112.74:31867/api/v1.0/surveys";

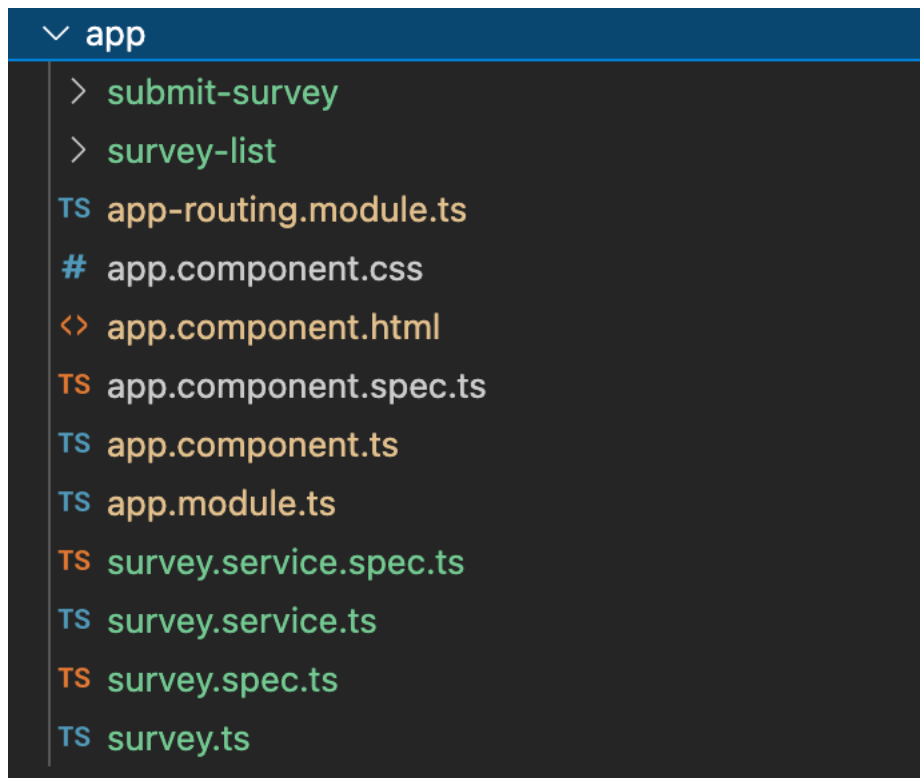
  constructor(private httpClient: HttpClient) { }

  getAllSurveys() : Observable<Survey[]>{
    return this.httpClient.get<Survey[]>(`${this.baseUrl}`)
  }

  submitSurvey(survey: Survey): Observable<Object>{
    return this.httpClient.post(`${this.baseUrl}`, survey);
  }
}

```

- Following is the walk through of Angular project



- Class for data binding: survey.ts
- Service for API calls: survey.service.ts
- Submit form component: submit-survey
- Get list of all the surveys: survey-list

Step 8: Build Docker image and containerize survey ui using Docker

- Write Dockerfile as below

```
FROM node:latest as build

LABEL author.name="Aniket Pandey"

WORKDIR /survey-ui/app

RUN npm install -g @angular/cli

COPY ./survey-ui/package.json .

RUN npm install

COPY ./survey-ui .

RUN ng build

FROM nginx as runtime

COPY --from=build /survey-ui/app/dist/survey-ui /usr/share/nginx/html
```

- Build docker image with the below command. My Dockerfile is inside infra folder and therefore I've specified path as infra/ui.Dockerfile. As a standard practice all the infrastructure setup related files should be kept inside infra folder for easy segregation.
docker build -f infra/ui.Dockerfile --tag survey-ui:amd64-v1.0 .
- Tag the docker image and push it on dockerhub using the below command
docker tag survey-ui:amd64-v1.0 aniket414/survey-ui:amd64-v1.0
docker push aniket414/survey-ui:amd64-v1.0

Step 9: Deploy the survey-ui docker image on cluster on setup three pods running at all time

- Now similarly perform Step 5 and Step 6 as mentioned above for Angular UI deployment.

Learn more about the improvements and new capabilities in this version.

Clusters 3 Import Existing Create

State	Name	Provider	Kubernetes Version	CPU	Memory	Pods
Active	local	k3s	v1.22.7+k3s1	0.1/2 cores	70 MiB/7.67 GiB	5/110
Active	survey-api	rke	v1.22.9	0.47/2 cores	280 MiB/7.57 GiB	13/110
Active	survey-ui	rke	v1.22.9	0.47/2 cores	280 MiB/7.57 GiB	13/110

- The cluster for UI is setup above

- The deployment of 3 pods is done and can be seen as below which are in ready state.

<input type="checkbox"/> State	Name	Image	Endpoints	Ready	Up-to-date	Available	Age	Health
Namespace: default								
<input type="checkbox"/> Active	ui-deployment	aniket414/survey-ui:amd64-v1.0	31222/TCP	3/3	3	3	5 hours	<div><div></div></div>

- After deployment the UI will look like as below

Survey List Submit Survey

GMU Survey Form

- This is the homepage it has two navigation options one to view list of surveys done till date and the other is for submitting new survey

Survey List Submit Survey

GMU Survey Form

List of Survey

First Name	Last Name	Email	Address	City	State	Zip	Telephone	Date	Recommendation	Interest	Likings
Aniket	Pandey	apandey7@gmu.edu	9500 Fairfax Blvd	Fairfax	VA	22031	8888214444	2022-05-04	Very Likely	Internet	Student,Campus,Sports
Sonali	Singh	sonali@mail.com	9450 Fairfax Blvd	Fairfax	VA	22031	8888214444	2022-05-04	Likely	Friends	Student,Campus,Sports,Location
Siddhanth	Kalyanpur	skalyanp@gmu.edu	9450 Fairfax Blvd	Tysons	VA	22031	8888215555	2022-05-05		Television	
Sai	P	sai@mail.com	Test Street	Test	MH	40114	9876543211	2022-05-06			
New	User	newuser@email.com	Street No 1	New York	NY	34012	8888888888	2022-05-05	Unlikely		
Sachin	Tendulkar	sachin_ten@gmail.com	Pali hill	Mumbai	Maharashtra	400212	704532211	2022-05-05	Likely	Others	Sports

- The above image displays list of surveys

New Survey

First Name *

Last Name *

Email *

Street Address *

City *

State *

Zip *

Telephone *

Date of Survey *

05/04/2022

What do you like about campus?

☐ Student

☐ Location

☐ Campus

☐ Atmosphere

☐ Dorm Rooms

☐ Sports

How did you become interested in George Mason University?

☐ Friends

☐ Television

☐ Internet

☐ Others

Would you recommend George Mason University to other prospective students?

Submit

Reset

Cancel

- The above image displays the survey form with all the fields.
- The UI calls the API can also be verified from the browser inspect.

The screenshot shows a web browser window with two main panels. The left panel displays a web page titled "GMU Survey Form" with a green header containing "Survey List" and "Submit Survey". Below the header is a section titled "List of Survey" which contains a table listing survey participants. The right panel shows the Chrome DevTools Network tab, displaying a network request for the survey form.

First Name	Last Name	Email	Address	City	State	Zip	Telephone
Aniket	Pandey	apandey7@gmu.edu	9500 Fairfax Blvd	Fairfax	VA	22031	88882144
Sonali	Singh	sonali@mail.com	9450 Fairfax Blvd	Fairfax	VA	22031	88882144
Siddhanth	Kalyanpur	skalyanp@gmu.edu	9450 Fairfax Blvd	Tysons	VA	22031	88882155
Sai	P	sai@mail.com	Test Street	Test	MH	40114	98765432
New	User	newuser@email.com	Street No 1	New York	NY	34012	88888888
Sachin	Tendulkar	sachin_ten@gmail.com	Pali hill	Mumbai	Maharashtra	400212	70453221

The Network tab on the right shows a request to `http://54.91.112.74:31867/api/v1.0/surveys`. The request method is GET, and the status code is 200. The response headers include `Access-Control-Allow-Origin: *`, `Connection: keep-alive`, `Content-Type: application/json`, `Date: Thu, 05 May 2022 08:45:35 GMT`, and `Keep-Alive: timeout=60`.