



College Dublin  
Computing • IT • Business

# Cloud Services and Integration

CA

Practical Assessment with Report

On

**Building A Cloud Based Microservices System**

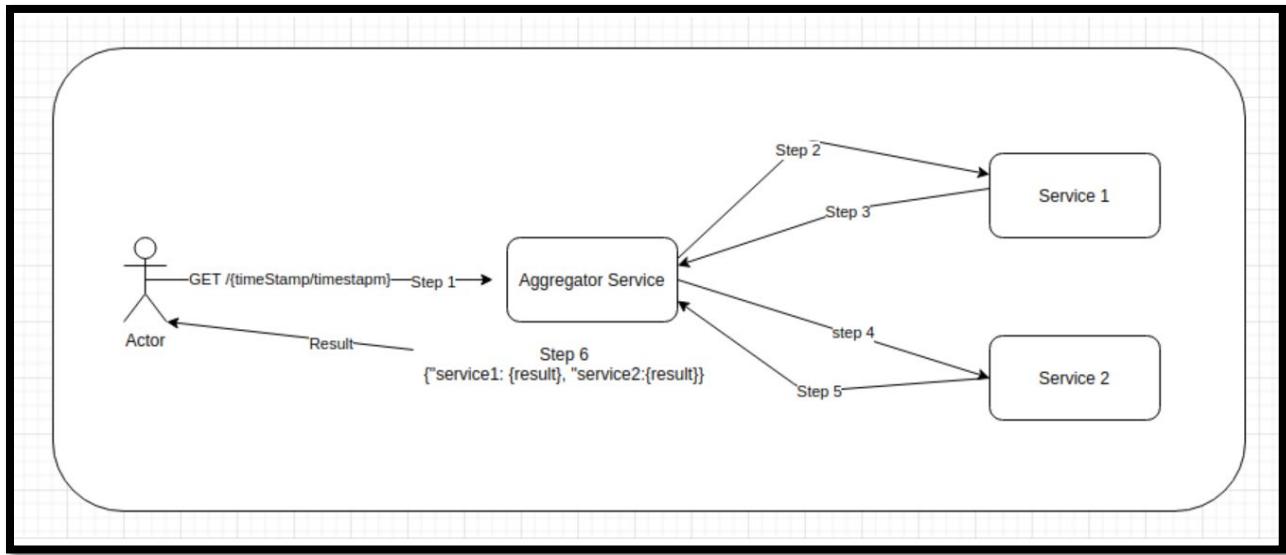
**Submitted by:** Akshay Jadhav (2020175)

**Submitted to:** Prof.David Gonzalez

## 1. Introduction

We are developing the microservices architecture which contains the 3 services like India-time service, US time service, and aggregator service.

The India service and US time service will convert the Unix timestamp into the required timezone (Indian and US time). The aggregator service will aggregate the two services and show the results as shown in the below diagram.



- **Why used Java?**

Chosen to use java for this microservices architecture development because especially everybody is aware of this language for a long time.

I personally find java code is easy to read and maintain as compared to other languages like Python, .NET, etc.

Spring Boot and Spark Java are famous Java microservices frameworks available for developing all types of solutions which provides higher functionality.

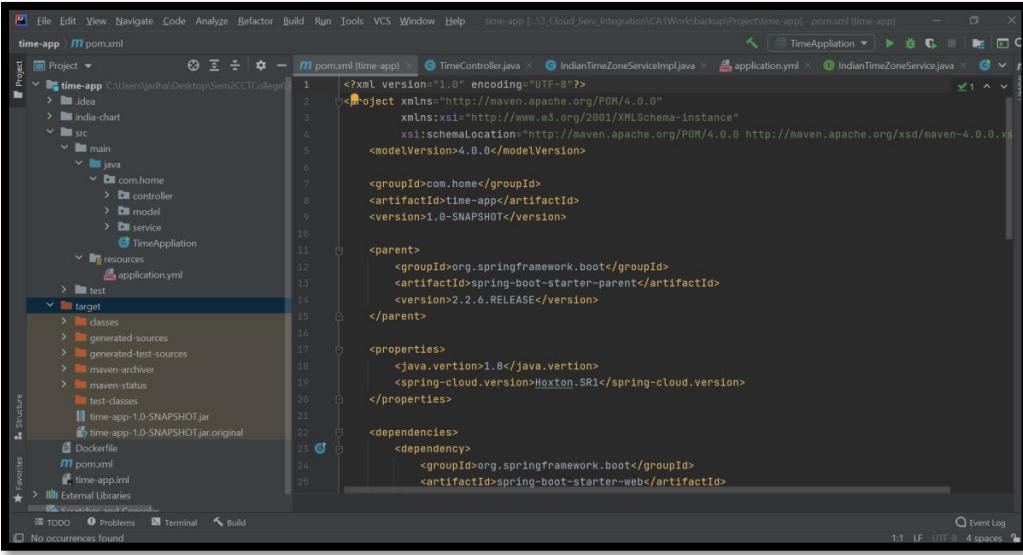
For development, I have used IntelliJ IDE because is easy to use and way faster than eclipse with options of more plugins.

## 2. Building Microservices

As per the requirements I have developed three microservices, out of which two microservices are for the two time zones (India & US time zone) and one is aggregator service which will merge the two time zone services.

The aggregator would be receiving GET request in “**GET /<timestamp>/time**” format.

- Created the spring boot maven project time-app (India-time service).
- Pom.xml file with all the properties and spring boot dependency.



```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

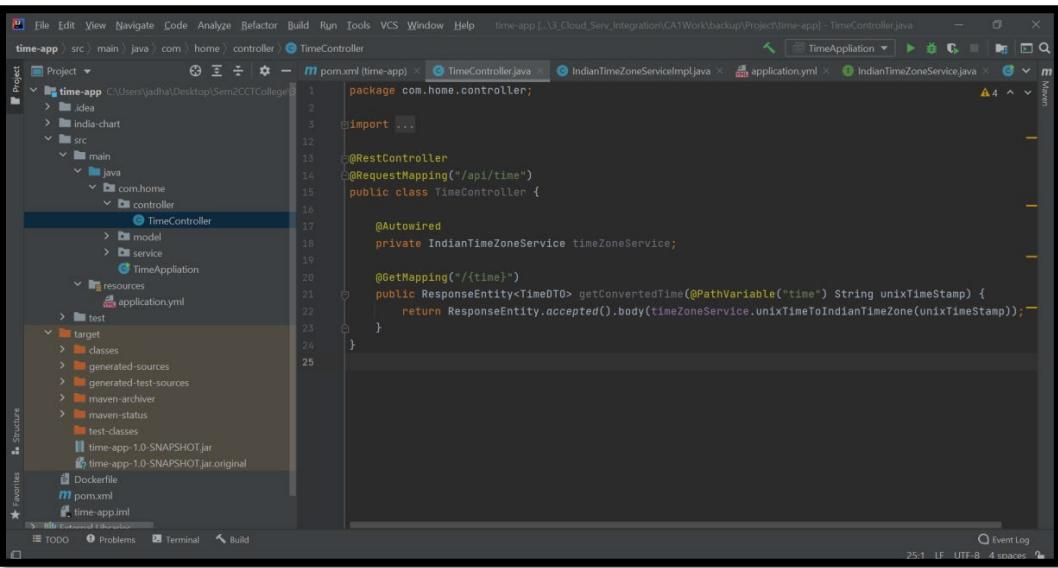
    <groupId>com.home</groupId>
    <artifactId>time-app</artifactId>
    <version>1.0-SNAPSHOT</version>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.6.RELEASE</version>
    </parent>

    <properties>
        <java.version>1.8</java.version>
        <spring-cloud.version>Hoxton.SR1</spring-cloud.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
    </dependencies>
</project>
```

- Here, this is our REST controller class in that we actually creating an indian time API.
- Getting request from outside.



```
package com.home.controller;

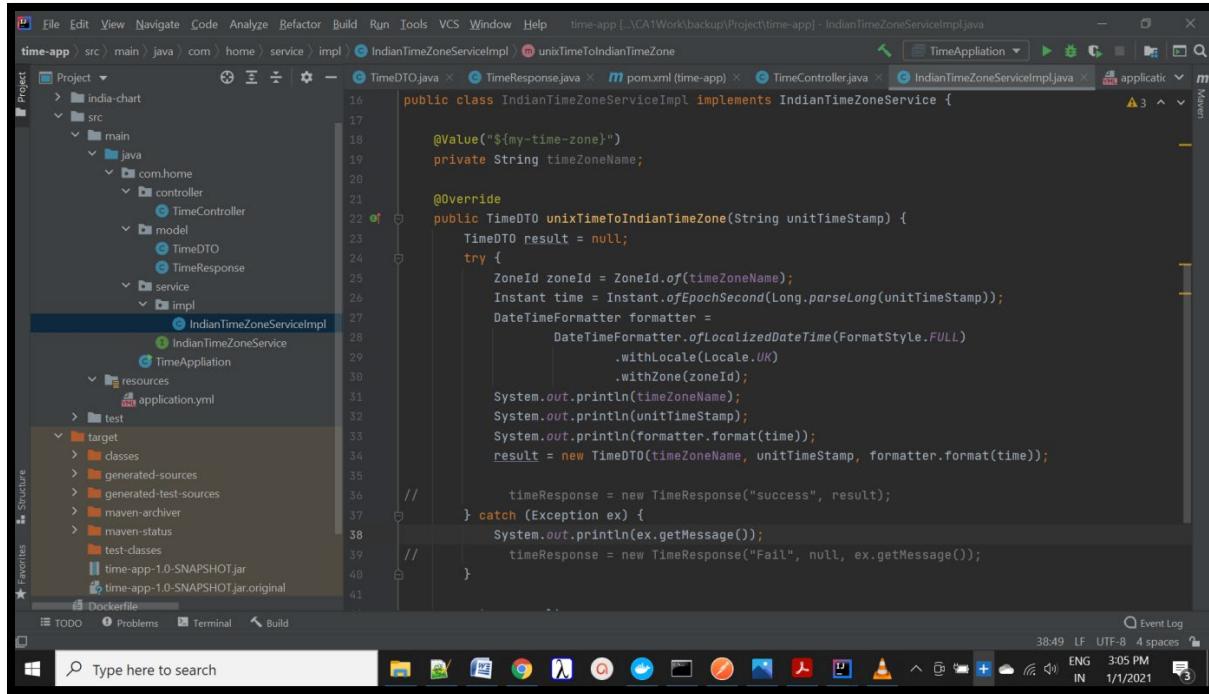
import ...

@RestController
@RequestMapping("/api/time")
public class TimeController {

    @Autowired
    private IndianTimeZoneService timeZoneService;

    @GetMapping("/{time}")
    public ResponseEntity<TimeDTO> getConvertedTime(@PathVariable("time") String unixTimeStamp) {
        return ResponseEntity.accepted().body(timeZoneService.unixTimeToIndianTimeZone(unixTimeStamp));
    }
}
```

- Then it will call our service to perform main logic by using date time formatter.



```

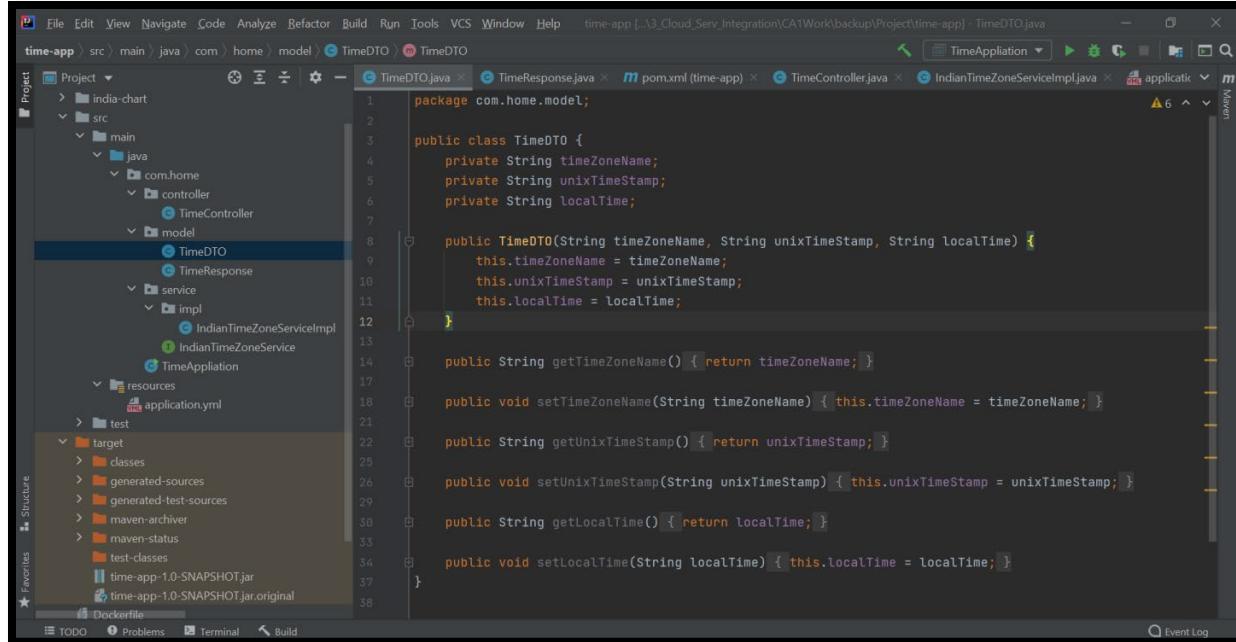
public class IndianTimeZoneServiceImpl implements IndianTimeZoneService {

    @Value("${my-time-zone}")
    private String timeZoneName;

    @Override
    public TimeDTO unixTimeToIndianTimeZone(String unitTimeStamp) {
        TimeDTO result = null;
        try {
            ZoneId zoneId = ZoneId.of(timeZoneName);
            Instant time = Instant.ofEpochSecond(Long.parseLong(unitTimeStamp));
            DateTimeFormatter formatter =
                DateTimeFormatter.ofLocalizedDateTime(FormatStyle.FULL)
                    .withLocale(Locale.UK)
                    .withZone(zoneId);
            System.out.println(timeZoneName);
            System.out.println(unitTimeStamp);
            System.out.println(formatter.format(time));
            result = new TimeDTO(timeZoneName, unitTimeStamp, formatter.format(time));
        } catch (Exception ex) {
            System.out.println(ex.getMessage());
            result = new TimeResponse("Fail", null, ex.getMessage());
        }
    }
}

```

- We are passing some values to outside .



```

package com.home.model;

public class TimeDTO {
    private String timeZoneName;
    private String unixTimeStamp;
    private String localTime;

    public TimeDTO(String timeZoneName, String unixTimeStamp, String localTime) {
        this.timeZoneName = timeZoneName;
        this.unixTimeStamp = unixTimeStamp;
        this.localTime = localTime;
    }

    public String getTimeZoneName() { return timeZoneName; }

    public void setTimeZoneName(String timeZoneName) { this.timeZoneName = timeZoneName; }

    public String getUnixTimeStamp() { return unixTimeStamp; }

    public void setUnixTimeStamp(String unixTimeStamp) { this.unixTimeStamp = unixTimeStamp; }

    public String getLocalTime() { return localTime; }

    public void setLocalTime(String localTime) { this.localTime = localTime; }
}

```

- This is the data we are actually passing outside to user like status, result and error and forward this response to controller class in the end.

The screenshot shows the IntelliJ IDEA interface with the project 'time-app' open. The left sidebar displays the project structure, including packages like 'india-chart', 'src' (containing 'main' and 'test'), and 'resources'. The main editor window shows the 'TimeResponse.java' file:

```
1 package com.home.model;
2
3 public class TimeResponse {
4     private String status;
5     private TimeDTO result;
6     private String error;
7
8     public TimeResponse(String status, TimeDTO result, String error) {
9         this.status = status;
10        this.result = result;
11        this.error = error;
12    }
13
14    public TimeResponse(String status, TimeDTO result) {
15        this.status = status;
16        this.result = result;
17    }
18
19    public String getStatus() {
20        return status;
21    }
22
23    public void setStatus(String status) {
24        this.status = status;
25    }
26}
```

- In spring boot there is a feature that we can store some parameters in yml file as shown below.
- Here we mentioned the time zone name.

The screenshot shows the IntelliJ IDEA interface with the project 'time-app' open. The left sidebar displays the project structure, including packages like 'india-chart', 'src' (containing 'main' and 'test'), and 'resources'. The main editor window shows the 'application.yml' file:

```
1 server:
2   port: 8660
3
4   my-time-zone: Asia/Calcutta
```

- Same as for US time service given below.
- Interface for both services

The screenshot shows two side-by-side Java code editors in an IDE.

**Left Editor (time-app) - IndianTimezoneService.java:**

```

package com.home.service;

import com.home.model.TimeDTO;

public interface IndianTimezoneService {
    TimeDTO unixTimeToIndianTimeZone(String unixTimeStamp);
}

```

**Right Editor (time-app-3) - NotIndianTimezoneService.java:**

```

package com.home.service;

import com.home.model.TimeDTO;

public interface NotIndianTimezoneService {
    TimeDTO unixTimeToIndianTimeZone(String unixTimeStamp);
}

```

- US time service is exactly same as Indian time service except application.yml
- Below two are the snapshots of US time zone service.

The screenshot shows a Java code editor in an IDE displaying the `TimeController.java` file.

```

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api/time")
public class TimeController {

    @Autowired
    private NotIndianTimezoneService timeZoneService;

    @GetMapping("/{time}")
    public ResponseEntity<TimeDTO> getConvertedTime(@PathVariable("time") String unixTimeStamp) {
        return ResponseEntity.accepted().body(timeZoneService.unixTimeToIndianTimeZone(unixTimeStamp));
    }
}

```

```

server:
  port: 8661
my-time-zone: America/Bahia

```

- Created aggregator service which will merge the response of two time zone services.
- Interface for aggregator.

```

package com.home.service;

import com.home.model.AggregatorResponse;

public interface AggregatorService {
    AggregatorResponse getLocalTimes(String unitTimeStamp);
}

```

- This is the main business logic, in that we are validating the input strings using regex validator (regular expression).
- We are getting the response from the service in format of time response class and created the list of that objects.

The screenshot shows the IntelliJ IDEA interface with the file `AggregatorServiceImpl.java` open. The code implements the `AggregatorService` interface. It includes a regular expression for validating Unix timestamps and logic for calling two service endpoints based on the timestamp format. The code is annotated with `@Configuration`, `@Autowired`, and `@Override`. The IntelliJ UI includes toolbars, a project tree, and a status bar at the bottom.

```
public class AggregatorServiceImpl implements AggregatorService {
    private final String reg = "^\d+";
    @Autowired
    private TimeClient timeClient;
    @Override
    public AggregatorResponse getLocalTimes(String unixTimeStamp) {
        AggregatorResponse aggregatorResponse;
        if(Pattern.matches(reg, unixTimeStamp)) {
            List<TimeDTO> resultTimes = new ArrayList<>();
            resultTimes.add(timeClient.callTimeService1(unixTimeStamp)); //adding the service results and calling service1
            resultTimes.add(timeClient.callTimeService2(unixTimeStamp)); //adding the service results and calling service2
            aggregatorResponse = new AggregatorResponse(resultTimes, status: "Success", error: null);
        } else {
            aggregatorResponse = new AggregatorResponse( times: null, status: "Invalid Unix Time Stamp", error: "Please enter correct 'Unix Time Stamp' format");
        }
        return aggregatorResponse;
    }
}
```

- Here we need to pass the response time and it will automatically convert that result to this object.

The screenshot shows the IntelliJ IDEA interface with the file `TimeClient.java` open. This class contains methods for calling two service endpoints using `RestTemplate`. It uses `@Configuration` and `@Autowired` annotations. The IntelliJ UI includes toolbars, a project tree, and a status bar at the bottom.

```
@Configuration
public class TimeClient {
    @Autowired
    private RestTemplate restTemplate;
    public TimeDTO callTimeService1(String unixTime) {
        HttpHeaders headers = new HttpHeaders();
        headers.setAccept(Arrays.asList(MediaType.APPLICATION_JSON));
        HttpEntity<String> entity = new HttpEntity<>(headers);
        return restTemplate.exchange(url: "http://192.168.0.129:8660/api/time/" + unixTime, HttpMethod.GET, entity, TimeDTO.class);
    }
    public TimeDTO callTimeService2(String unixTime) {
        HttpHeaders headers = new HttpHeaders();
        headers.setAccept(Arrays.asList(MediaType.APPLICATION_JSON));
        HttpEntity<String> entity = new HttpEntity<>(headers);
        return restTemplate.exchange(url: "http://192.168.0.129:8661/api/time/" + unixTime, HttpMethod.GET, entity, TimeDTO.class);
    }
}
```

- Started all three services one by one.

```

Run: TimeApplication x
2021-01-01 15:36:23.953 INFO 2196 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.33]
2021-01-01 15:36:24.219 INFO 2196 --- [main] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplicationContext
2021-01-01 15:36:24.219 INFO 2196 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization com
2021-01-01 15:36:24.568 INFO 2196 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskE
2021-01-01 15:36:24.848 INFO 2196 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8660 (http) with co
2021-01-01 15:36:24.852 INFO 2196 --- [main] com.home.TimeApplication : Started TimeApplication in 5.918 seconds (JVM r
2021-01-01 15:38:45.326 INFO 2196 --- [nio-8660-exec-1] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring DispatcherServlet 'dispat
2021-01-01 15:38:45.331 INFO 2196 --- [nio-8660-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2021-01-01 15:38:45.409 INFO 2196 --- [nio-8660-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 78 ms

Run: Asia/Calcutta x
2021-01-01 15:37:04.482 INFO 17360 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTask
2021-01-01 15:37:04.749 INFO 17360 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8661 (http) with c
2021-01-01 15:37:04.753 INFO 17360 --- [main] com.home.TimeApplication : Started TimeApplication in 5.374 seconds (JVM r
2021-01-01 15:38:45.940 INFO 17360 --- [nio-8661-exec-1] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring DispatcherServlet 'dispat
2021-01-01 15:38:45.943 INFO 17360 --- [nio-8661-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2021-01-01 15:38:45.970 INFO 17360 --- [nio-8661-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 27 ms

Run: America/Bahia x
2021-01-01 15:37:04.482 INFO 17360 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTask
2021-01-01 15:37:04.749 INFO 17360 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8661 (http) with c
2021-01-01 15:37:04.753 INFO 17360 --- [main] com.home.TimeApplication : Started TimeApplication in 5.374 seconds (JVM r
2021-01-01 15:38:45.940 INFO 17360 --- [nio-8661-exec-1] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring DispatcherServlet 'dispat
2021-01-01 15:38:45.943 INFO 17360 --- [nio-8661-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2021-01-01 15:38:45.970 INFO 17360 --- [nio-8661-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 27 ms

Run: AggregatorApplication x
2021-01-01 15:37:59.976 INFO 17928 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.3
2021-01-01 15:38:00.162 INFO 17928 --- [main] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplicationCo
2021-01-01 15:38:00.163 INFO 17928 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization co
2021-01-01 15:38:00.509 INFO 17928 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTask
2021-01-01 15:38:00.765 INFO 17928 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8670 (http) with c
2021-01-01 15:38:00.769 INFO 17928 --- [main] com.home.AggregatorApplication : Started AggregatorApplication in 4.928 second
2021-01-01 15:38:44.761 INFO 17928 --- [nio-8670-exec-1] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring DispatcherServlet 'dispat
2021-01-01 15:38:44.761 INFO 17928 --- [nio-8670-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2021-01-01 15:38:44.771 INFO 17928 --- [nio-8670-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 9 ms

```

Build completed successfully with 1 warning in 6 sec, 550 ms (8 minutes ago)

Build completed successfully with 1 warning in 7 sec, 400 ms (8 minutes ago)

Build completed successfully with 1 warning in 6 sec, 127 ms (8 minutes ago)

## Output:

- The output below shows the converted Indian time zone and the US time zone (different formats) with status and error details.

Postman

My Workspace

Untitled Request

GET http://192.168.0.129:8670/701226600/time

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

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

2
  "times": [
    {
      "timeZoneName": "Asia/Calcutta",
      "unixTimeStamp": "701226600",
      "localTime": "Sunday, 22 March 1992 06:40:00 o'clock IST"
    },
    {
      "timeZoneName": "America/Bahia",
      "unixTimeStamp": "701226600",
      "localTime": "21/03/92 22:10"
    }
  ],
  "status": "Success",
  "error": null

```

### 3. Docker Activities:

We use Docker containers to create, deploy, and execute different applications. These containers are nothing but the packaged application with all required libraries and dependencies to run on any platform.

So basically our application starts and stops from Docker without any manual intervention. Everything will be taken care of by Docker itself.

- Dockerfile for all three services.

The screenshot shows three Dockerfiles in the IntelliJ IDEA interface:

- time-app/Dockerfile**:

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
EXPOSE 8668
ARG JAR_FILE=target/*.jar
ADD ${JARFILE} app.jar
ENTRYPOINT ["java", "-jar", "/app.jar", "$@"]
```
- time-app-3/Dockerfile**:

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
EXPOSE 8661
ARG JARFILE=target/*.jar
ADD ${JARFILE} app.jar
ENTRYPOINT ["java", "-jar", "/app.jar", "$@"]
```
- time-aggregator/Dockerfile**:

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
EXPOSE 8670
ARG JARFILE=target/*.jar
ADD ${JARFILE} app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

- Build the indian time service project to create docker image.

The screenshot shows the Maven Lifecycle in the IntelliJ IDEA interface for the **time-app** project:

- Maven** tool window: Shows the **Lifecycle** section with the **install** goal selected.
- Run** tool window: Shows the command **time-app [install]** and its output:

```
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 10.315 s
[INFO] Finished at: 2021-01-01T17:00:39Z
[INFO] -----
```

Process finished with exit code 0

- Docker build command for building docker image.

The screenshot shows the IntelliJ IDEA interface with the project 'time-app' open. The Dockerfile tab is selected, displaying the following content:

```

FROM openjdk:8-jdk-alpine
VOLUME /tmp
EXPOSE 8660
ARG JAR_FILE=target/*.jar
ADD ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "/app.jar", "$@"]

```

The terminal tab shows the command being run and its output:

```

C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-app>docker build -t india-service .
[+] Building 0.6s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 228B
=> [internal] load .dockerrcignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/openjdk:8-jdk-alpine
=> [1/2] FROM docker.io/library/openjdk:8-jdk-alpine
=> [internal] load build context
=> => transferring context: 19..34MB
=> => CACHED [2/2] ADD target/*.jar app.jar
=> => exporting layers
=> => writing image sha256:eb44da377bd5269429f8e5d0bc1f7759a1c46421d92fe63c52978d5c5264c1c9
=> => naming to docker.io/library/india-service

```

The terminal also displays a warning about Dockerfile detection.

- Run the docker container in background mode (-d) and publish the port number on host (-p).

```

C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-app>docker run -d -p 8660:8660 --name indiaservice india-service
86cf5b76e70a2c26f0df4d165d9313ed8f0ca8d7512a49eff4c851097a4ffa9

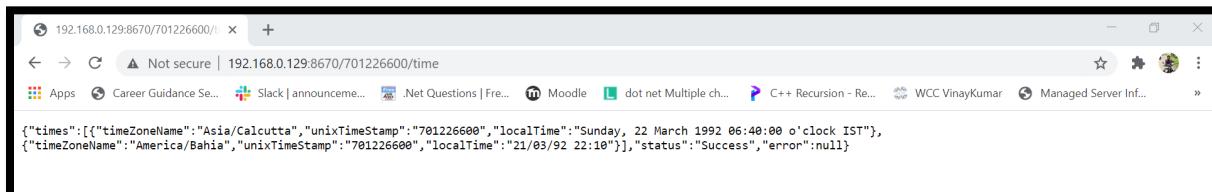
```

- Perform all the same steps for US time service and aggregator service remaining service.
- Now we have all three docker images available up and running.

The screenshot shows the terminal output of the 'docker ps' command, listing three containers:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
1001e0b0b586	india-service2	"java -Djava.security. ..."	40 seconds ago	Up 39 seconds	0.0.0.0:8660->8660/tcp
78826122ea1b	agg-time2	"java -jar /app.jar" ..."	6 minutes ago	Up 6 minutes	0.0.0.0:8670->8670/tcp
f63b669761ce	us-service	"java -jar /app.jar ..." ..."	11 minutes ago	Up 11 minutes	0.0.0.0:8661->8661/tcp

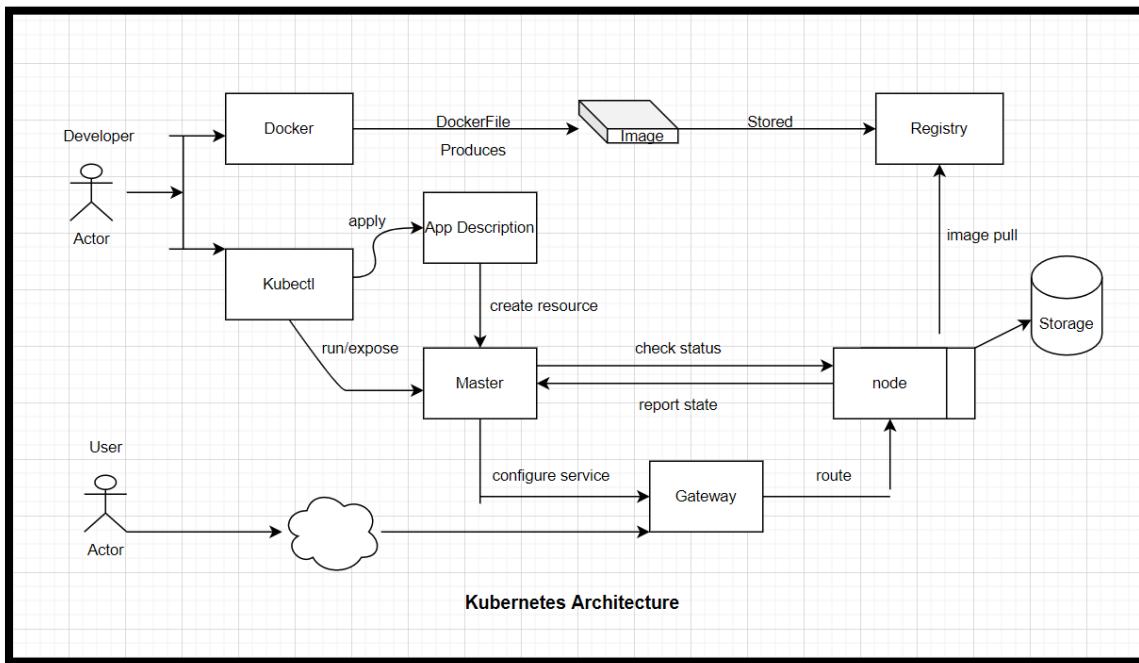
## Output:



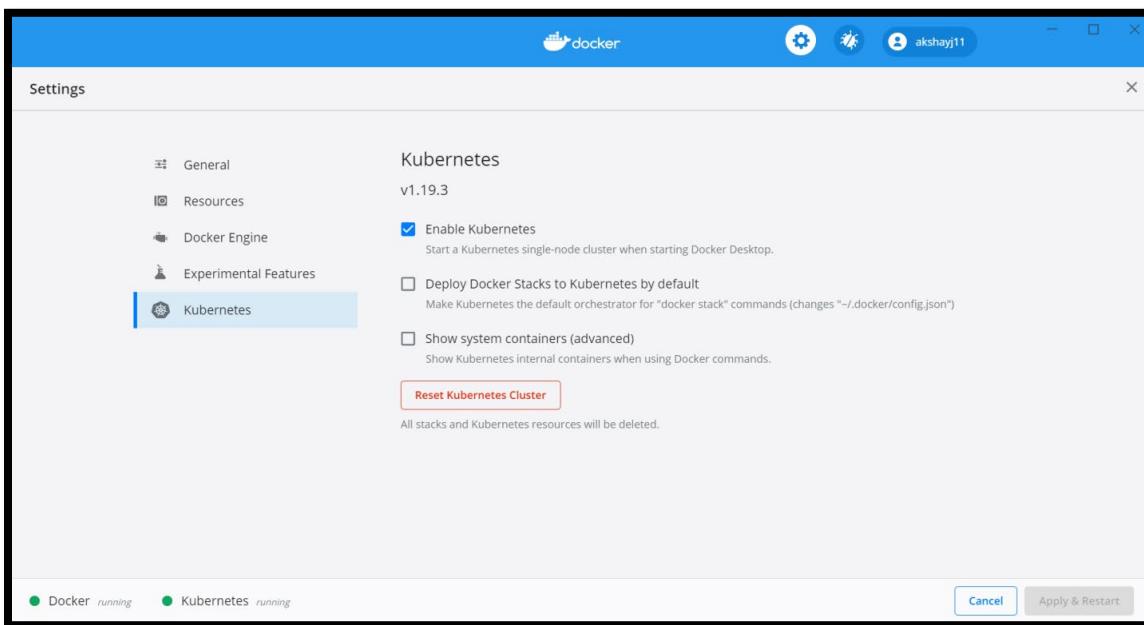
## 4. Kubernetes Activities:

The Kubernetes use over Docker for container orchestration activities. It helps in automating the deployment, scaling, and management of applications. The main parts of Kubernetes are deployment and service YAML files.

- Created the basic Kubernetes working diagram for better understanding.

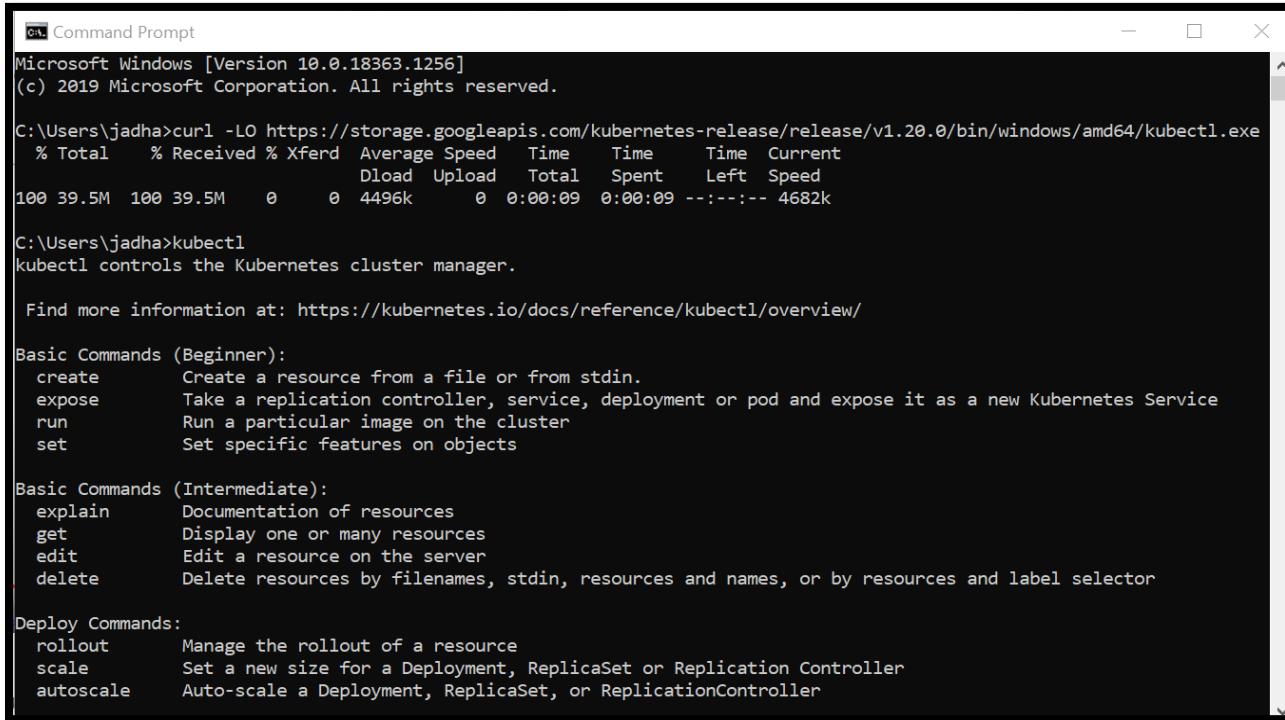


- First enable Kubernetes from Docker desktop.



- **Kubectl and Minikube Setup:**

Minikube will allow us to run a Kuberntes cluster locally we will set up a Kubernetes single node cluster environment locally using minikube and to perform application deployment, inspection, cluster management we will install kubectl command-line tool.



```
Command Prompt
Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\jadha>curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.20.0/bin/windows/amd64/kubectl.exe
  % Total    % Received % Xferd  Average Speed   Time     Time   Current
          Dload  Upload Total   Spent    Left  Speed
100 39.5M  100 39.5M    0      0  4496k      0  0:00:09  0:00:09  ---:--- 4682k

C:\Users\jadha>kubectl
kubectl controls the Kubernetes cluster manager.

Find more information at: https://kubernetes.io/docs/reference/kubectl/overview/

Basic Commands (Beginner):
  create      Create a resource from a file or from stdin.
  expose      Take a replication controller, service, deployment or pod and expose it as a new Kubernetes Service
  run         Run a particular image on the cluster
  set         Set specific features on objects

Basic Commands (Intermediate):
  explain     Documentation of resources
  get         Display one or many resources
  edit        Edit a resource on the server
  delete      Delete resources by filenames, stdin, resources and names, or by resources and label selector

Deploy Commands:
  rollout     Manage the rollout of a resource
  scale       Set a new size for a Deployment, ReplicaSet or Replication Controller
  autoscale   Auto-scale a Deployment, ReplicaSet, or ReplicationController
```

- Minikube downloaded and installed.



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>minikube version
minikube version: v1.16.0
commit: 9f1e482427589ff8451c4723b6ba53bb9742fbb1

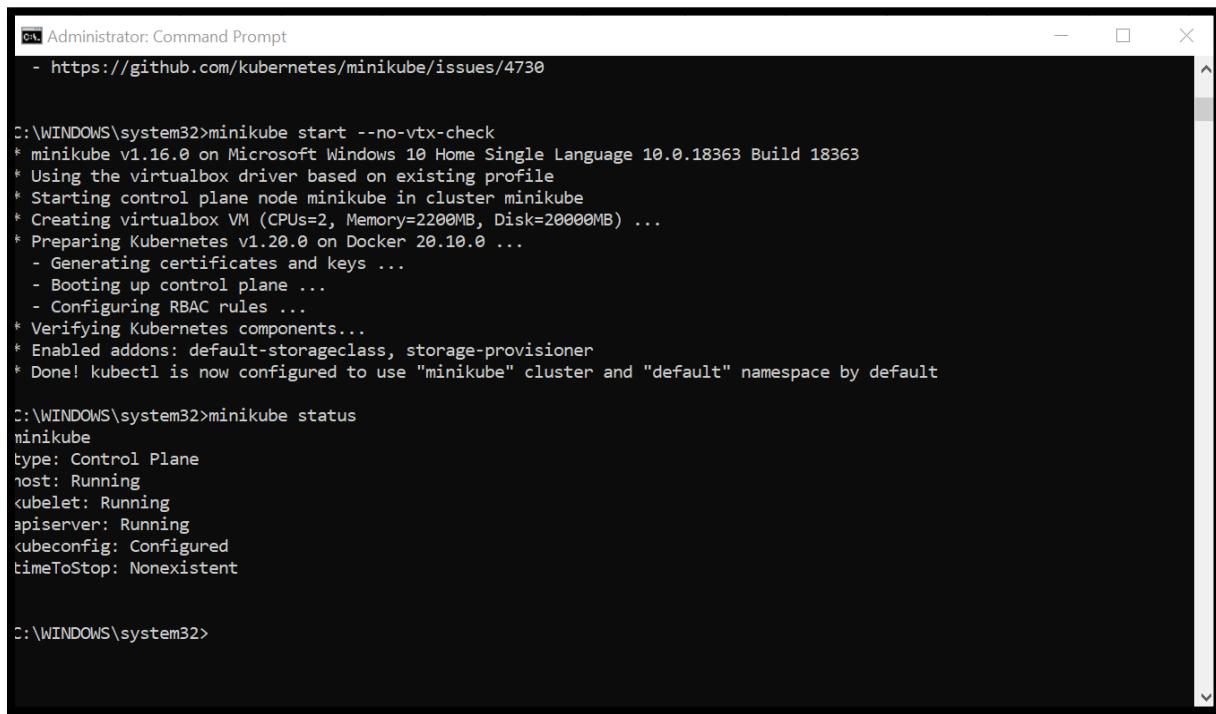
C:\WINDOWS\system32>
```

- Error while starting minikube cluster with “minikube start” command.

```
C:\WINDOWS\system32>minikube start --vm-driver=virtualbox
* minikube v1.16.0 on Microsoft Windows 10 Home Single Language 10.0.18363 Build 18363
* Using the virtualbox driver based on user configuration
* Downloading VM boot image ...
  > minikube-v1.16.0.iso.sha256: 65 B / 65 B [-----] 100.00% ? p/s 0s
  > minikube-v1.16.0.iso: 212.62 MiB / 212.62 MiB [] 100.00% 4.02 MiB p/s 53s
* Starting control plane node minikube in cluster minikube
* Creating virtualbox VM (CPUs=2, Memory=2200MB, Disk=20000MB) ...
! StartHost failed, but will try again: creating host: create: precreate: This computer doesn't have VT-X/AMD-v enabled.
Enabling it in the BIOS is mandatory
* Creating virtualbox VM (CPUs=2, Memory=2200MB, Disk=20000MB) ...
* Failed to start virtualbox VM. Running "minikube delete" may fix it: creating host: create: precreate: This computer d
oesn't have VT-X/AMD-v enabled. Enabling it in the BIOS is mandatory

X Exiting due to HOST_VIRT_UNAVAILABLE: Failed to start host: creating host: create: precreate: This computer doesn't ha
ve VT-X/AMD-v enabled. Enabling it in the BIOS is mandatory
* Suggestion: Virtualization support is disabled on your computer. If you are running minikube within a VM, try '--drive
r=docker'. Otherwise, consult your systems BIOS manual for how to enable virtualization.
* Related issues:
- https://github.com/kubernetes/minikube/issues/3900
- https://github.com/kubernetes/minikube/issues/4730
```

- Instead, I used “minikube start –no-vtx-check” command and started successfully as we can see the virtual machine is getting created as shown below.



```
Administrator: Command Prompt
- https://github.com/kubernetes/minikube/issues/4730

C:\WINDOWS\system32>minikube start --no-vtx-check
* minikube v1.16.0 on Microsoft Windows 10 Home Single Language 10.0.18363 Build 18363
* Using the virtualbox driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Creating virtualbox VM (CPUs=2, Memory=2200MB, Disk=20000MB) ...
* Preparing Kubernetes v1.20.0 on Docker 20.10.0 ...
- Generating certificates and keys ...
- Booting up control plane ...
- Configuring RBAC rules ...
* Verifying Kubernetes components...
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

C:\WINDOWS\system32>minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
cubeconfig: Configured
timeToStop: Nonexistent

C:\WINDOWS\system32>
```

- Checked the minikube status, kubectl version, kubectl nodes and opened minikube dashboard.

```

Administrator: Command Prompt - minikube dashboard
C:\WINDOWS\system32>minikube status
minikube
  type: Control Plane
  host: Running
  kubelet: Running
  apiserver: Running
  kubeconfig: Configured
  timeToStop: Nonexistent

C:\WINDOWS\system32>kubectl version
Client Version: version.Info{Major:"1", Minor:"19", GitVersion:"v1.19.3", GitCommit:"1e11e4a2108024935ecfcb2912226cedeaef99df", GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z", GoVersion:"go1.15.2", Compiler:"gc", Platform:"windows/amd64"}
Server Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.0", GitCommit:"af46c47ce925f4c4ad5cc8d1fc46c7b77d13b38", GitTreeState:"clean", BuildDate:"2020-12-08T17:51:19Z", GoVersion:"go1.15.5", Compiler:"gc", Platform:"linux/amd64"}

C:\WINDOWS\system32>kubectl get nodes
NAME      STATUS    ROLES     AGE   VERSION
minikube  Ready     control-plane,master  37m   v1.20.0

C:\WINDOWS\system32>minikube dashboard
* Enabling dashboard ...
* Verifying dashboard health ...
* Launching proxy ...
* Verifying proxy health ...
* Opening http://127.0.0.1:52807/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in your default browser...

```

- Minikube is working fine as shown below.

Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Created
minikube	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux kubernetes.io/arch: arm64	True	750.00m (37.50%)	0.00m (0.00%)	170.00Mi (7.96%)	170.00Mi (7.96%)	42 minutes ago

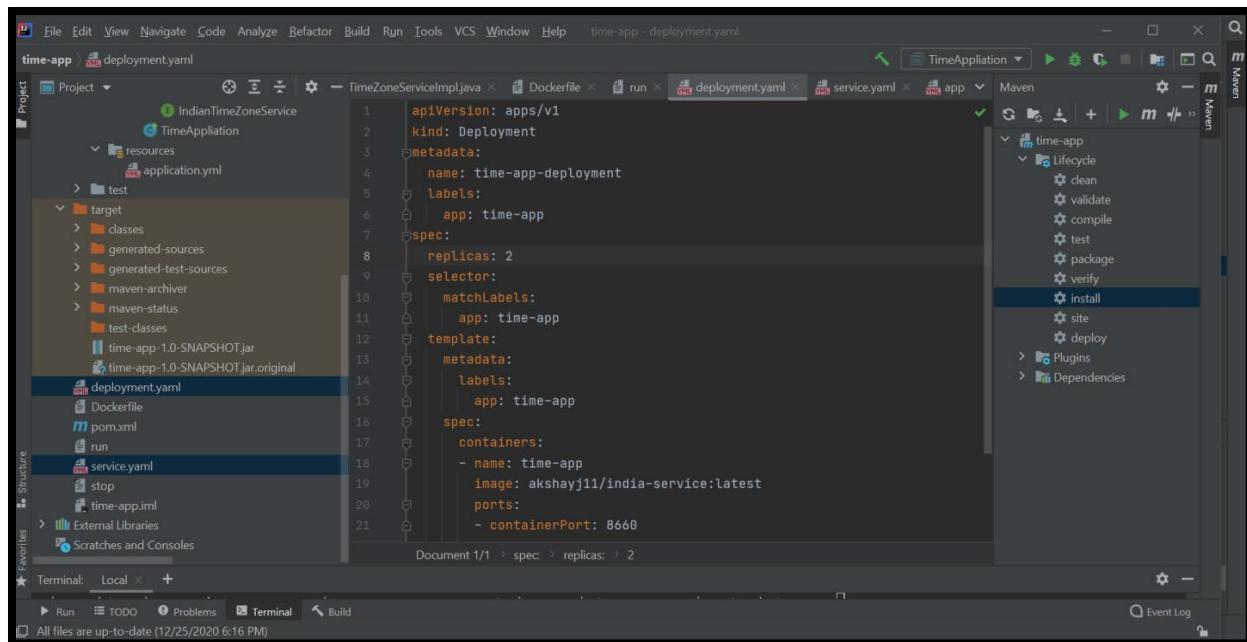
- Pushed the Docker images of all three services to the Docker hub public repository called micro-service.

```
C:\WINDOWS\system32>docker tag eb44da377bd5 akshayj11/micro_services:indiatime
C:\WINDOWS\system32>docker push akshayj11/micro_services:indiatime
The push refers to repository [docker.io/akshayj11/micro_services]
211fb8fb639b: Mounted from akshayj11/india-service
ceaf9e1ebef5: Layer already exists
9b9b7f3d56a0: Layer already exists
f1b5933fe4b5: Layer already exists
indiatime: digest: sha256:1f07e1abc2b2e5fd242839b62381f725585486402a4146045517e16576ad9a1c size: 1159

C:\WINDOWS\system32>docker tag 0bc8dd48d6a4 akshayj11/micro_services:agg-service
C:\WINDOWS\system32>docker push akshayj11/micro_services:agg-service
The push refers to repository [docker.io/akshayj11/micro_services]
81b5bcd26f70: Pushed
ceaf9e1ebef5: Layer already exists
9b9b7f3d56a0: Layer already exists
f1b5933fe4b5: Layer already exists
agg-service: digest: sha256:814a45d89d16225b3b252b81bb5d9b92052d379bc6473a7c3666d12b5996b5c5 size: 1159

C:\WINDOWS\system32>
```

- Created docker deployment YAML file for India – time service.



- Created service YAML file for India-time service with “LoadBalancer” type (later on changed to NodeIP).

```

apiVersion: v1
Kind: Service
metadata:
  name: time-service
spec:
  selector:
    app: time-app
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 8660
      targetPort: 8660

```

- Applied deployment and service YAML in Kubernetes using “kubectl apply -f deployment.yaml” command.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: time-app-deployment
  labels:
    app: time-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: time-app

```

C:\Users\jadha\Desktop\Sem2CCTCollege\3\_Cloud\_Serv\_Integration\CA1Work\Files from fve\Project\time-app>kubectl apply -f deployment.yaml  
deployment.apps/time-app-deployment configured

C:\Users\jadha\Desktop\Sem2CCTCollege\3\_Cloud\_Serv\_Integration\CA1Work\Files from fve\Project\time-app>kubectl get replicsets

NAME	DESIRED	CURRENT	READY	AGE
nginx-depl-5c8bf76b5b	1	1	1	25h
time-app-deployment-8686d8fb45	2	2	2	21h

C:\Users\jadha\Desktop\Sem2CCTCollege\3\_Cloud\_Serv\_Integration\CA1Work\Files from fve\Project\time-app>>kubectl apply -f deployment.yaml

- You can see two replica set means two pods running in Kubernetes for this service.

```

C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\Files from fve\Project\time-app>kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
nginx-depl-5c8bf76b5b-vm5z7   1/1    Running   0          26h
time-app-deployment-8686d8fb45-fj9mk  1/1    Running   0          21h
time-app-deployment-8686d8fb45-vq4cf  1/1    Running   0          21m

C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\Files from fve\Project\time-app>kubectl get replicasets
NAME        DESIRED   CURRENT   READY   AGE
nginx-depl-5c8bf76b5b      1         1         1      26h
time-app-deployment-8686d8fb45  2         2         2      21h

C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\Files from fve\Project\time-app>kubectl get services
NAME        TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
kubernetes  ClusterIP   10.96.0.1    <none>       443/TCP   42h
time-service  LoadBalancer  10.111.27.9  <pending>   8660:31672/TCP  21h

```

- YAML files of both US time and aggregator services.

```

us-time-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: us-time-deployment
  labels:
    app: time-app-3
spec:
  replicas: 2
  selector:
    matchLabels:
      app: time-app-3
  template:
    metadata:
      labels:
        app: time-app-3
    spec:
      containers:
        - name: time-app-3
          image: akshayj11/micro_services:ustime
          ports:
            - containerPort: 8661

```

```

aggregator-time-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: aggregator-time-deployment
  labels:
    app: time-aggregator
spec:
  replicas: 2
  selector:
    matchLabels:
      app: time-aggregator
  template:
    metadata:
      labels:
        app: time-aggregator
    spec:
      containers:
        - name: time-aggregator
          image: akshayj11/micro_services:agg-service
          ports:
            - containerPort: 8670

```

```

us-time-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: us-time-service
spec:
  selector:
    app: time-app-3
  type: ClusterIP
  ports:
    - protocol: TCP
      port: 8661
      targetPort: 8661

```

```

aggregator-time-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: aggregator-time-service
spec:
  selector:
    app: time-aggregator
  type: NodePort
  ports:
    - protocol: TCP
      port: 8670
      targetPort: 8670

```

Applied deployment and service YAML in Kubernetes for both US time and aggregator service using the same command as stated earlier.

All the deployment and services to access these time service has been placed on Kubernetes cluster.

- Check using “kubectl get deployment” and “kubectl get services”.

```
C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-aggregator>kubectl get deployment
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
nginx-depl     1/1     1            1           27h
time-aggregator-deployment 2/2     2            2           42s
time-app-3-deployment 2/2     2            2           7m9s
time-app-deployment 2/2     2            2           22h

C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-aggregator>kubectl get services
NAME          TYPE      CLUSTER-IP    EXTERNAL-IP   PORT(S)      AGE
kubernetes    ClusterIP  10.96.0.1    <none>        443/TCP     44h
time-aggregator-service LoadBalancer 10.100.4.42  <pending>    8670:30405/TCP 52s
time-app-3-service LoadBalancer 10.109.184.203 <pending>    8661:30477/TCP 5m45s
time-service   LoadBalancer 10.111.27.9   <pending>    8660:31672/TCP 22h

C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-aggregator>[
```

- Faced the API server minikube issue then deleted and started minikube again.

```
D:\>minikube delete --purge --all
* Removing C:\Users\jadha\.minikube\machines\minikube ...
* Removed all traces of the "minikube" cluster.
* Successfully deleted all profiles
* Successfully purged minikube directory located at - [C:\Users\jadha\.minikube]

D:\>minikube start --no-vtx-check
* minikube v1.16.0 on Microsoft Windows 10 Home Single Language 10.0.18363 Build 18363
* Automatically selected the docker driver. Other choices: hyperv, virtualbox
* Starting control plane node minikube in cluster minikube
* Downloading Kubernetes v1.20.0 preload ...
  > preloaded-images-k8s-v8-v1....: 360.05 MiB / 491.00 MiB 73.33% 1.88 MiB -
```

We can expose the services type NodePort or Load Balancer by using the minikube tunnel command. It creates the network route from the host to the service which is on the cluster via cluster IP.

- As we can see the External IP status is pending without running the minikube tunnel.

```
C:\> Administrator: Command Prompt
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
* Stopping tunnel for service aggregator-time-service.

C:\WINDOWS\system32>kubectl get svc aggregator-time-service
NAME          TYPE      CLUSTER-IP    EXTERNAL-IP   PORT(S)      AGE
aggregator-time-service NodePort  10.99.205.113  <none>        8670:30809/TCP 27h

C:\WINDOWS\system32>
```

- After running the minikube tunnel in another terminal, we got the external IP of one LoadBalancer type service.

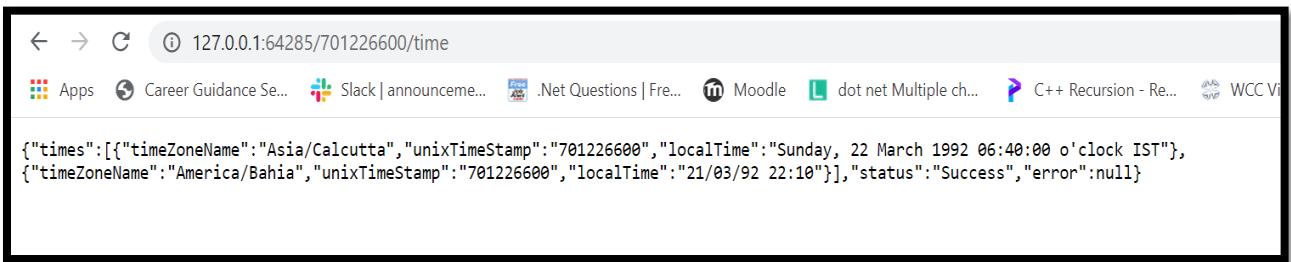
```
C:\Users\jadha>minikube tunnel
* Starting tunnel for service time-app-agg-service.
```

```
C:\WINDOWS\system32>kubectl get svc time-app-agg-service
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
time-app-agg-service   LoadBalancer  10.98.155.86  127.0.0.1   8670:32168/TCP  5m21s
```

- Accessing the time aggregator LoadBalancer type service deployed in kubernetes.
- Use “minikube service time-app-agg-service –url” command.

```
C:\WINDOWS\system32>minikube service time-app-agg-service --url
* Starting tunnel for service time-app-agg-service.
|-----|-----|-----|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----|-----|-----|-----|
| default | time-app-agg-service | | http://127.0.0.1:64285 |
|-----|-----|-----|-----|
http://127.0.0.1:64285
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
* Stopping tunnel for service time-app-agg-service.
```

## Output:



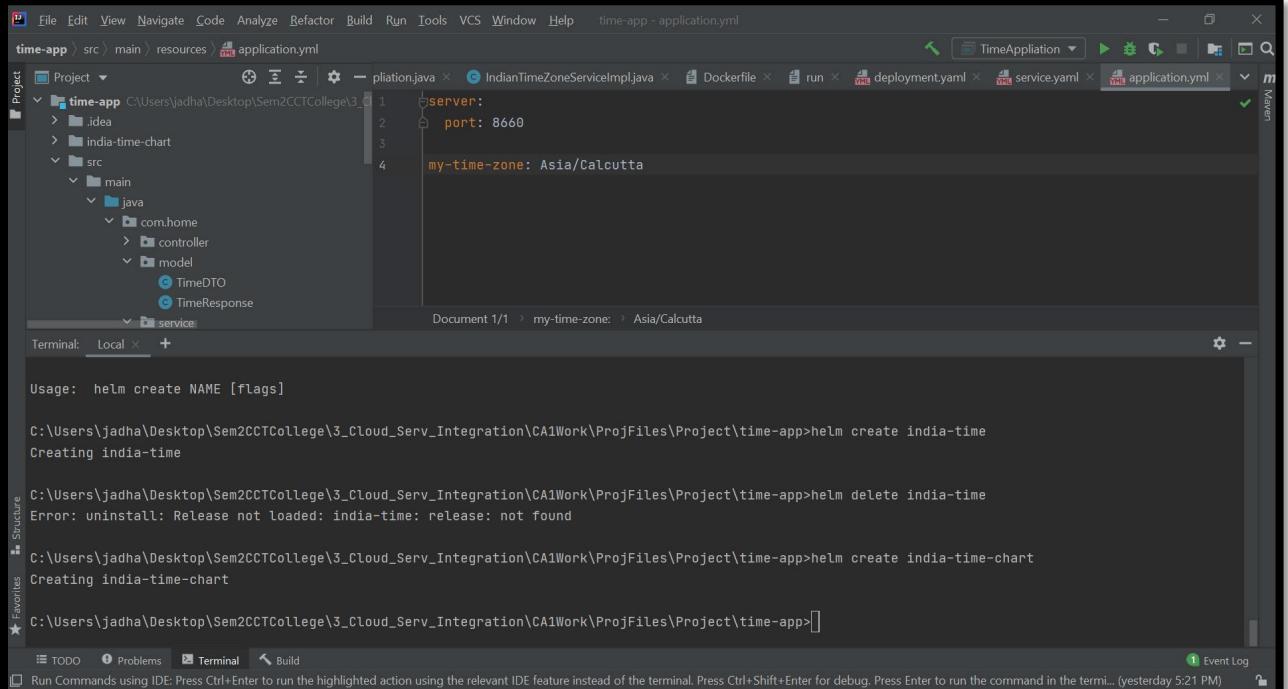
Successfully tested the application in Kubernetes environment.

## 5. Helm Chart

It is a Kubernetes package manager and the packages in that are called a chart. Mostly used to install, upgrade, and define the complex application as well as we can share and publish the helm charts.

- Installed the Helm and added to the path of environment variable.

- Created the helm chart for API service (e.g., India-time service).
- Helm create India-time-chart



The screenshot shows the IntelliJ IDEA interface with the project 'time-app' open. The code editor displays the file 'application.yaml' containing configuration for a service named 'my-time-zone' with port 8660. Below the code editor, a terminal window shows the command-line process of creating a Helm chart:

```

Usage: helm create NAME [flags]

C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-app>helm create india-time
Creating india-time

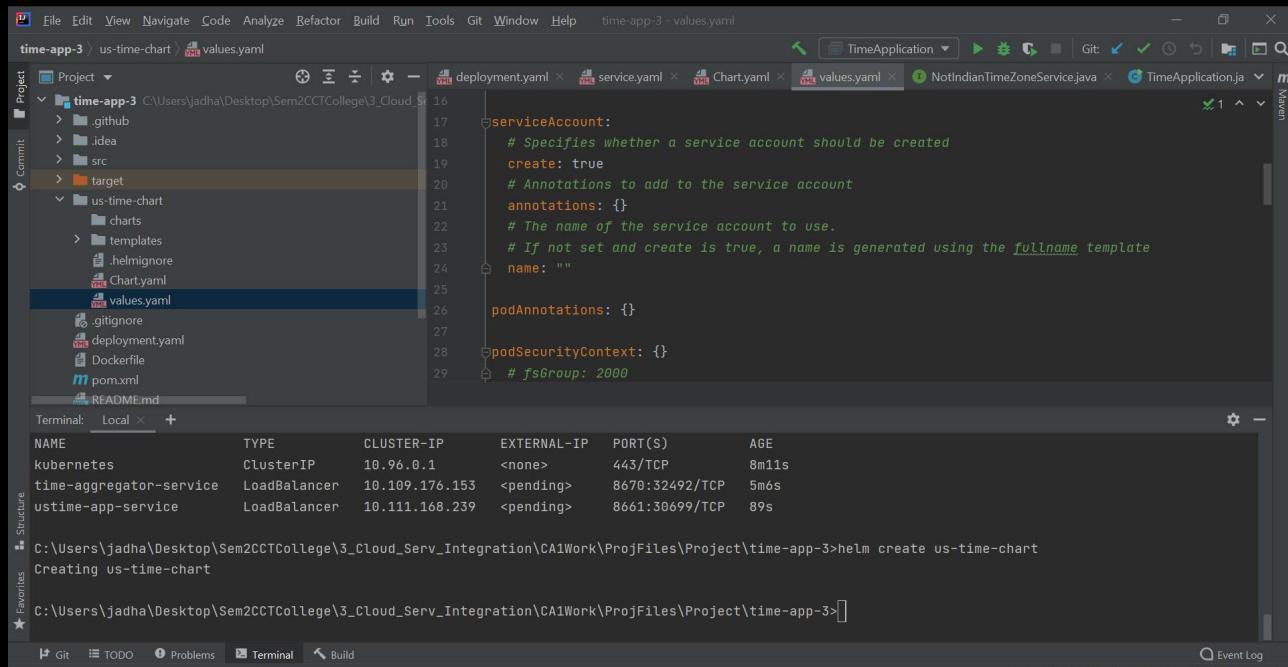
C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-app>helm delete india-time
Error: uninstall: Release not loaded: india-time: release: not found

C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-app>helm create india-time-chart
Creating india-time-chart

C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-app>

```

- Created the Helm chart for time-app-3 (US time service).



The screenshot shows the IntelliJ IDEA interface with the project 'time-app-3' open. The code editor displays the file 'values.yaml' for the 'us-time-chart' chart, which includes configurations for a service account and pod annotations. Below the code editor, a terminal window shows the command-line process of creating a Helm chart:

```

NAME          TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes    ClusterIP   10.96.0.1       <none>        443/TCP     8m11s
time-aggregator-service LoadBalancer  10.109.176.153  <pending>     8670:32492/TCP  5m6s
ustime-app-service   LoadBalancer  10.111.168.239  <pending>     8661:30699/TCP  89s

C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-app-3>helm create us-time-chart
Creating us-time-chart

C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-app-3>

```

- Same for aggregator service.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: time-aggregator
spec:
  replicas: 2
  selector:
    matchLabels:
      app: time-aggregator
  template:
    metadata:
      labels:
        app: time-aggregator
    spec:
      containers:
        - name: time-aggregator
          image: akshayj11/micro_services:agg-service
          ports:
            - containerPort: 8670

```

```

C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-aggregator>kubectl delete deployment time-app-3-deployment

deployment.apps "time-app-3-deployment" deleted

```

- Again, faced the issue of “kubernetes unreachable ...net/http: TLS handshake timeout” as shown below:

```

! Executing "docker container inspect minikube --format={{.State.Status}}" took an unusually long time: 2.1148239s
* Restarting the docker service may improve performance.

C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-app-3>helm delete us-time-chart
Error: Kubernetes cluster unreachable: Get "https://127.0.0.1:49157/version?timeout=32s": net/http: TLS handshake timeout

```

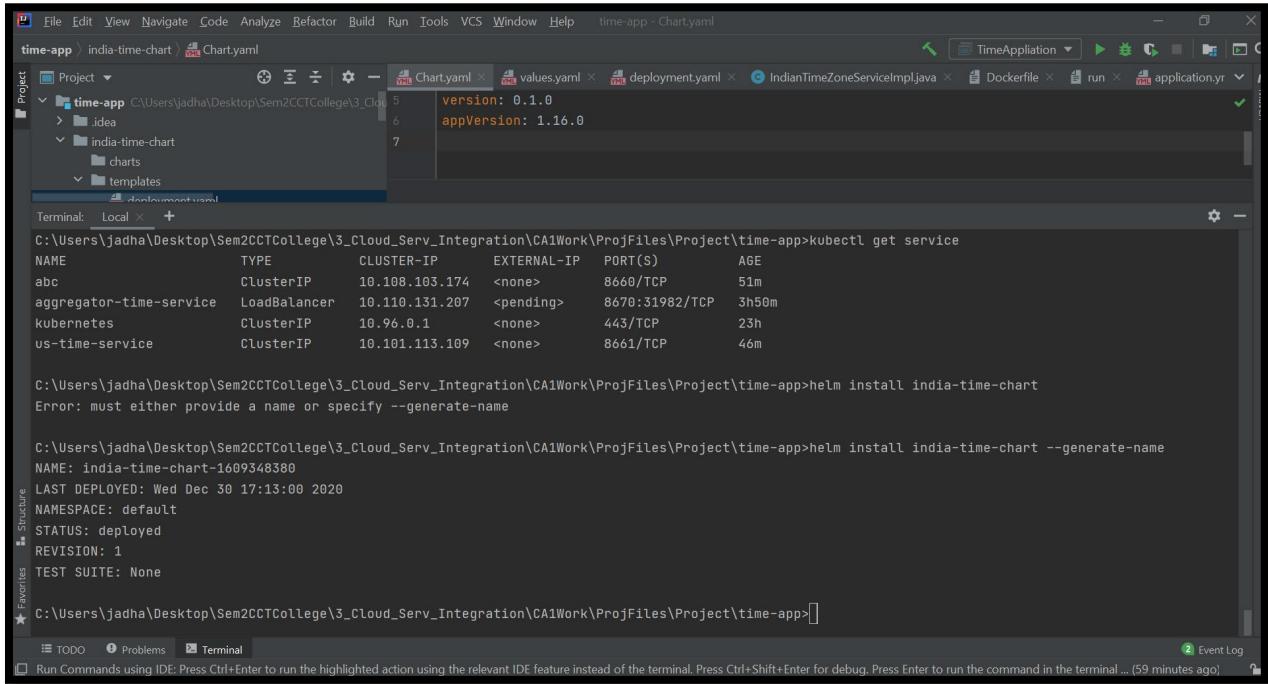
- Again, restarted the minikube.
- Copied all YAML files (deployment and service) in template folder of helm chart.
- Remove the deployment and service YAML from Kubernetes otherwise helm would face an error while recreating these deployments in its process.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: india-time-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: time-app
  template:
    metadata:
      labels:
        app: time-app
    spec:
      containers:

```

- Installed helm india-time-chart using “helm install india-time-chart --generate-name command”.



```

time-app > india-time-chart > Chart.yaml
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help time-app - Chart.yaml
Project > time-app > .idea > india-time-chart > charts > templates > Chart.yaml
version: 0.1.0
appVersion: 1.16.0

Terminal: Local × +
C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-app>kubectl get service
NAME           TYPE     CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
abc            ClusterIP 10.108.103.174 <none>       8660/TCP   51m
aggregator-time-service LoadBalancer 10.110.131.207 <pending>    8670:31982/TCP 3h50m
kubernetes     ClusterIP 10.96.0.1    <none>       443/TCP    23h
us-time-service ClusterIP 10.101.113.109 <none>       8661/TCP   46m

C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-app>helm install india-time-chart
Error: must either provide a name or specify --generate-name

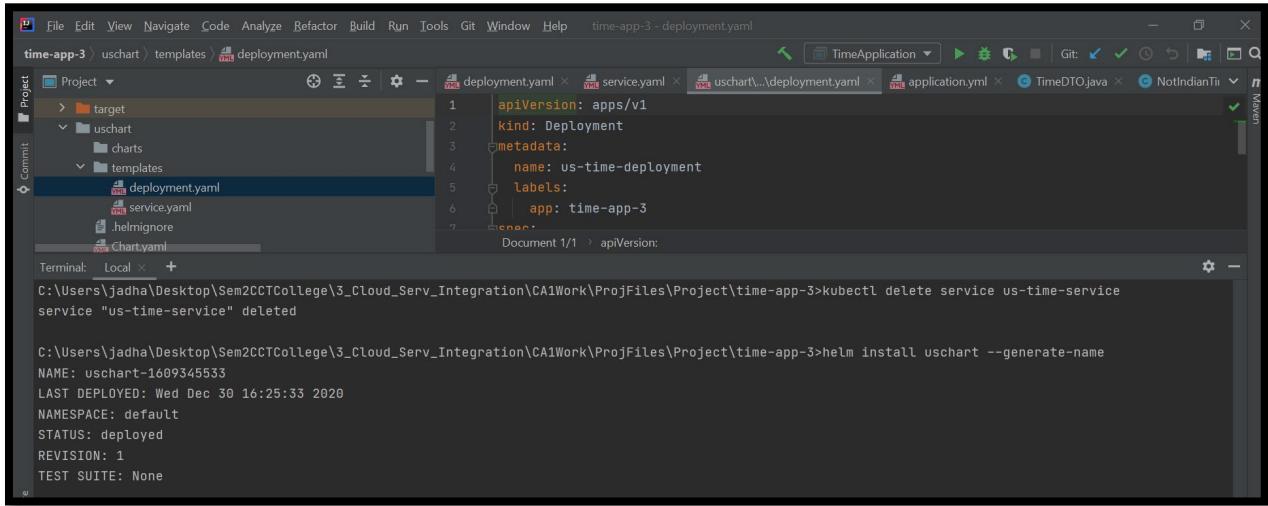
C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-app>helm install india-time-chart --generate-name
NAME: india-time-chart-1609348380
LAST DEPLOYED: Wed Dec 30 17:13:00 2020
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None

C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-app>

```

The screenshot shows the IntelliJ IDEA interface with the terminal tab active. The terminal window displays the command `helm install india-time-chart` followed by an error message: "Error: must either provide a name or specify --generate-name". Below this, the command `helm install india-time-chart --generate-name` is run successfully, resulting in a new chart named `india-time-chart-1609348380` in the `default` namespace, revision 1, and status `deployed`. The project structure on the left shows the `time-app` directory containing `Chart.yaml`, `values.yaml`, `deployment.yaml`, and other files.

- Installed US time chart



```

time-app-3 > uschart > templates > deployment.yaml
File Edit View Navigate Code Analyze Refactor Build Run Tools Git Window Help time-app-3 - deployment.yaml
Project > target > uschart > charts > templates > deployment.yaml > service.yaml > .helmignore > Chart.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: us-time-deployment
  labels:
    app: time-app-3
Document 1/1 > apiVersion:

Terminal: Local × +
C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-app-3>kubectl delete service us-time-service
service "us-time-service" deleted

C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-app-3>helm install uschart --generate-name
NAME: uschart-1609345533
LAST DEPLOYED: Wed Dec 30 16:25:33 2020
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None

```

The screenshot shows the IntelliJ IDEA interface with the terminal tab active. The terminal window displays the command `helm install uschart --generate-name` followed by the creation of a new chart named `uschart-1609345533` in the `default` namespace, revision 1, and status `deployed`. The project structure on the left shows the `time-app-3` directory containing `Chart.yaml`, `service.yaml`, `deployment.yaml`, and other files.

- Installed aggregator time chart.

The screenshot shows the IntelliJ IDEA interface with the 'time-aggregator' project open. The left sidebar displays the project structure, including a 'charts' folder containing 'agg-time-chart' and 'templates' subfolders, along with 'Chart.yaml', 'values.yaml', 'deployment.yaml', and 'service.yaml'. The right pane shows a terminal window with the following command history:

```
C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-aggregator>kubectl delete service aggregator-time-service
service "aggregator-time-service" deleted

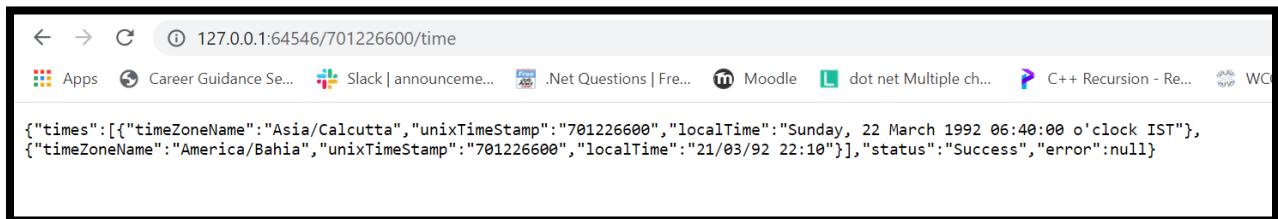
C:\Users\jadha\Desktop\Sem2CCTCollege\3_Cloud_Serv_Integration\CA1Work\ProjFiles\Project\time-aggregator>helm install agg-time-chart --generate-name
NAME: agg-time-chart-1609348568
LAST DEPLOYED: Wed Dec 30 17:16:09 2020
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

- Accessing the time aggregator NodePort type service deployed and controlled by Helm Chart.
- Use “minikube service aggregator-time-service –url” command.

```
C:\WINDOWS\system32>kubectl get svc aggregator-time-service
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)      AGE
aggregator-time-service   NodePort   10.99.205.113 <none>       8670:30809/TCP   27h

C:\WINDOWS\system32>minikube service aggregator-time-service --url
* Starting tunnel for service aggregator-time-service.
|-----|-----|-----|-----|
|  NAMEspace |     NAME    | TARGET PORT |      URL      |
|-----|-----|-----|-----|
|  default  | aggregator-time-service |          | http://127.0.0.1:64546 |
|-----|-----|-----|-----|
http://127.0.0.1:64546
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

## Output:



Successfully tested the application after packaged in Helm chart.

## **Conclusion:**

In this way, I have learned how to build cloud-based microservice, including creating a docker image (containerization), container management using Kubernetes, and packing all Kubernetes services in Helm chart. It gives me a clear understanding of how multiple teams can work on each service and deploy the application faster. It also minimizes the development work and improves code reusability.

This way of developing a solution based on a set of services that communicate through an API. In addition to all this using docker container, Kubernetes, and Helm chart make it easy to deploy, maintain an application with high availability as well as scalability.