**Birla Institute of Technology & Science, Pilani**

**Work Integrated Learning Programmes Division**

**First Semester 2022-2023**

**Comprehensive Exam (EC-3 Regular)**

Course No. : SE ZG583

Course Title : Scalable Services

No of questions: 5

No of pages: 5

Nature of Exam : Open Book

Weightage : 40%

Duration : 2 ½ Hours

Date of Exam : 26/11/2022 (FN)

Note to Students:

1. Please follow all the *Instructions to Candidates* given on the cover page of the answer book.
2. All parts of a question should be answered consecutively. Each answer should start from a fresh page.
3. Assumptions made if any, should be stated clearly at the beginning of your answer.

**Question 1: Set A**

An XYZ organization has been migrating few microservices from the request-reply pattern to event driven architecture. They encountered the following pitfall stated below.

An order service should update the **order db** and publish the “*order created*” event to a message broker in a single lockstep. They are encountering inconsistencies between the orderdb and event consumers.

a) Identify the problem and possible reason for the inconsistencies. [2]

b) Propose reliable solutions (at least two) for the same. Suggest suitable design patterns, technology choices for the same. List out pros and cons for your solutions as well. [6]

**Question 1: Set B**

An XYZ organisation has a recommendation service which uses synchronous request response pattern. The API gateway invokes a product service which in turn calls the recommendation service for recommendations on the products and other related downstream services. The services communicate synchronously.

a) List out the problems they will encounter if recommendation service is unresponsive or if one of the downstream service is unresponsive [2]

b) Propose reliable solutions (at least three) to handle such partial failures. Suggest suitable design patterns, technology choices (if any) for the same. List out pros and cons for your solutions as well. [6]

**Question 2: Set A**

Consider a docker file as below [5]

FROM python:3.8-slim-buster  
COPY . .  
RUN pip install -r requirements.txt  
ENTRYPOINT ["python", "main.py"]

When you run docker build for the first time, it takes longer time.

a) Suggest changes to the above docker file to optimize the docker build.

b) When you run the docker build the second time, it takes considerably lesser time. Elucidate the reason for the same

**Question 2: Set B**

Consider a docker file as below [5]

FROM node:16

WORKDIR /usr/src/app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 8080

CMD [ "node", "server.js" ]

When you run docker build for the first time, it takes longer time.

a) Suggest changes to the above docker file to optimize the docker build.

b) When you run the docker build the second time, it takes considerably lesser time. Elucidate the reason for the same

**Question 3: SET A**  [5]

In an ecommerce application to render the product details page, the product service communicates with the recommendation service and the ads service and sends the response to the frontend. All these services are written in different languages by different teams. External monitoring only tells you the overall response time and number of invocations and no insight into the individual operations.

* Propose a method to follow the course of a request or transaction as it travels through the ecommerce application that is being monitored.
* Track a single interaction on how it is processed across multiple services.

**Question 3: SET B** [5]

Consider an example with 3 microservices that calls to each other in a row. All these services are written in different languages by different teams. When we are calling the other microservices using a REST interface, the other microservice returns a 500 HTTP status code. Details of failure are unknown to us. External monitoring only tells you the overall response time and number of invocations and no insight about the individual operations and failures.

Propose a method to extend your monitoring efforts.

**Question 4 Set A**

Consider a lending library which supplies books on payment to its users. The users login to the system, select and block the book they require. Block the book will reduce the number of copies available by one. Then they proceed to payment. On successful payment, the book is allotted for lending to the user. Because each service has its own database, you need to use a mechanism to maintain data consistency across those databases. Solve this distributed transactions problem using the choreography based saga pattern.

a) Design and model this use case as a collection of sub–transactions. For each sub transaction design a relevant compensating transaction. [2]

b) List down the sequence of steps of a successful saga as it will appear in a Saga log (happy path) [4]

c) List down the sequence of steps of an unsuccessful saga as it will appear in a Saga log (when one of the service fails) [4]

**Question 4 SET B:**

Consider a lending library which supplies books on payment to its users. The users login to the system, select and block the book they require. Block the book will reduce the number of copies available by one. Then they proceed to payment. On successful payment, the book is allotted for lending to the user. Because each service has its own database, you need to use a mechanism to maintain data consistency across those databases. Solve this distributed transactions problem using the orchestration based saga pattern.

a) Design and model this use case as a collection of sub–transactions. For each sub transaction design a relevant compensating transaction. [2]

b) List down the sequence of steps of a successful saga as it will appear in a Saga log (happy path) [4]

c) List down the sequence of steps of an unsuccessful saga as it will appear in a Saga log (when one of the service fails) [4]

**Question 5 Set A**

**Answer the following questions related to Kubernetes**

An image is built from the below docker file and is pushed to Docker hub for global access.

FROM alpine

RUN apk add --update redis

EXPOSE 6379

CMD ["redis-server"]

a) Create a manifest in YAML to create a simple Pod in Kubernetes. [3]

b) Write a manifest in YAML to configure a LivenessProbe to the container (created in step 1). The probe must respond within the 1-second timeout and Kubernetes will call the probe every 10 seconds. [2]

c) Why does Kubernetes allow more than one container in a Pod? Mention one suitable use case of a multi container pod? [2]

d) John and his team used the below Deployment File to deploy application and it works fine. Now they need to update the image to hub/samples/hello-app:2.0. John edited the yaml file to update the image and used kubectl apply –f. The application faced a little downtime. Provide a solution for John and his team to do zero downtime deployment. Briefly explain the steps with relevant manifest file to achieve the same. [5]

apiVersion: apps/v1  
kind: Deployment  
metadata:  
name: hello-app  
namespace: default  
spec:  
replicas: 2  
selector:  
matchLabels:  
app: hello-app  
deployment  
template:  
metadata:  
labels:  
app: hello-app  
spec:  
containers:  
- image: hub/samples/hello-app:1.0  
imagePullPolicy: Always  
name: hello-dep  
ports:  
- containerPort: 8080

**Question 5 Set B**

**Answer the following questions related to Kubernetes**

a) Write a YAML manifest file to create a Pod running a Container that exposes a Port [Use an NGINIX image and expose on port 8080] [3]

b) Configure the maximum memory and CPU requirements for the container. Maximum Memory- 128MB Maximum CPU- 1 [2]

c) In a multicontainer pod, all containers in a Pod are being started in parallel. In our use case, second container is dependent on first container. There is a chance that the second container starts before the first one and the second container will fail. How would you solve this issue? [2]

d) John and his team used the below Deployment File to deploy application and it works fine. Now they need to update the image to hub/samples/hello-app:2.0. John edited the yaml file to update the image and used kubectl apply –f. The application faced a little downtime. Provide a solution for John and his team to do zero downtime deployment. Briefly explain the steps with relevant manifest file to achieve the same. [5]

apiVersion: apps/v1  
kind: Deployment  
metadata:  
name: hello-app  
namespace: default  
spec:  
replicas: 2  
selector:  
matchLabels:  
app: hello-app  
deployment  
template:  
metadata:  
labels:  
app: hello-app  
spec:  
containers:  
- image: hub/samples/hello-app:1.0  
imagePullPolicy: Always  
name: hello-dep  
ports:  
- containerPort: 8080

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*