INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

Examinations Control Office

Examination	B TECH VI SEMESTER END EXAMINATIONS REGULAR JUNE 2025 REG UG20		
Month & Year	1-Jun	**Date**	20/06/2025
Course Name	DEVOPS		
Course Code	ACSC42	**E-Code**	6775

Instructions to Evaluators

- Evaluators should spend at least 3-5 minutes on one answer booklet during the evaluation.
- Evaluators should cross check that marks are allotted for all the attempted questions.
- The marks should be assigned fairly according to the mark distribution specified in the scheme of evaluation.
- For questions that were attempted incorrectly, evaluators are required to award zero marks.
- The evaluator must give a proper justification in case of any mistakes identified in the marks provided.

START WRITING FROM HERE

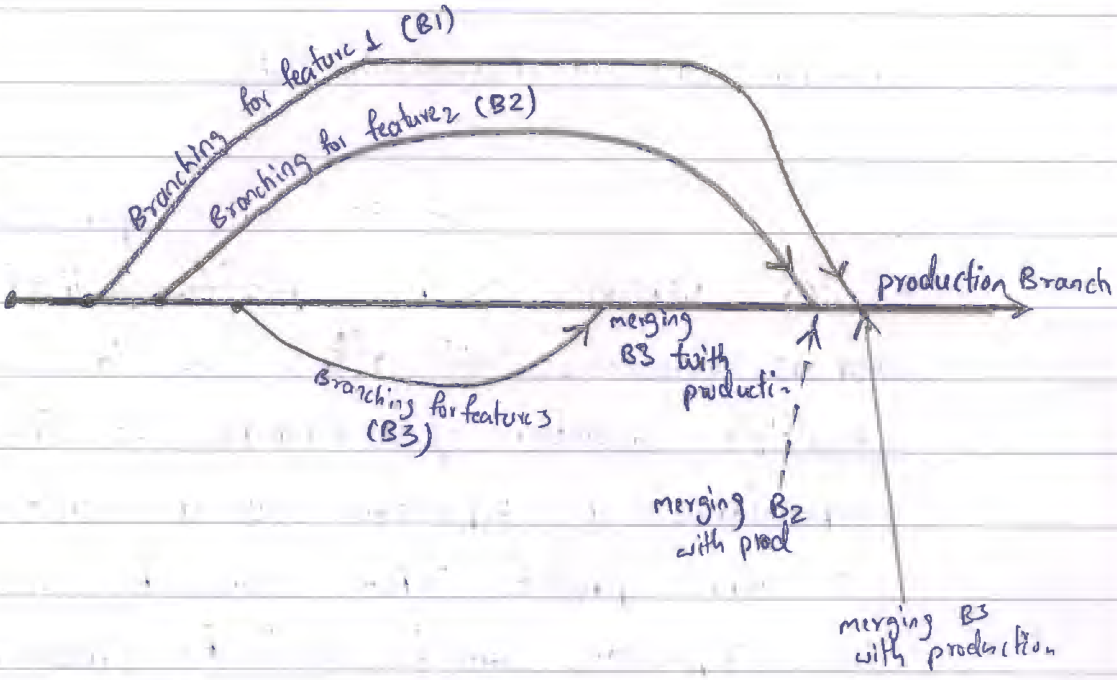
Q.No.

1.a **Branching and Merging in Git :-**

Git is an open source, distributed version control platform that provides various services. Git allows as to create multiple number of branches for the same repository. A repository is a code base for a given project. There can be many branches in a repository which will be maintained by different teams.

Suppose let us consider a product which is on production. And the company wants to add multiple independent features to that product. Here the company will create one new branch for each feature. As each feature is independent and doest rely on other new features. Here each branch is tested and build. Then the owner (author) of the repository will have the authority to merge these branches with the production branch. And the author is responible for accepting the pull request from the branch and only then it is safe to delete the feature branch

Q.No.


Branching for feature 1 (B1)
Branching for feature2 (B2)
production Branch
merging B3 with production
Branching for feature 3 (B3)
merging B2 with prod
merging B3 with production

Affter mergin, the production branch will have the new features of the Branches.

Managing Conflicts :-

Conflicts some times occur when you are pushing /merging the branches. The best practices for managing conflict in Git is to ensure the files are properly imported. Add all related modules are push to Git properly. In case of a conflict, Git guardian will show the part of code that are leading to merge conflict and we should resolve the conflict. Only then the Branche can be merged with production Branch.

Q.No.

1.b Struggling with maintaining consistency in infrastructure configurations accross multiple environments :

Insonsistancy in maintaining of infrastructure configurations accross multiple environments could lead to issues like System downtime, Accessability ~~probl~~ problems, protability issues, performance issues and Bad user experience etc. Implementing "Infrastructure as Code" (IaC) will solve this ~~incon~~ inconsistency challenge.

As our issue is raised on multiple environments usage of Infrastructure as code will help maintain the consistency in infrastructure configurations accross the multiple environment. which directly resolve the problems with downtime, Responce time, multiple environment issue, scalability issue, and performance issue To implement Infrastructure as code, the developer team should have good knowledge about how to code, that can be modulised, resusable, efficient. So they should SOLID principles. And implement performance techniques like caching and implement user-experience techniques like

Pre-fetch and re-fetch. This is a method used by many product to make user feel like the data is loaded instantaneously. But in reality the all data that user could see is pre-loaded when an application is opened, and the data is refreshed and stored when user ~~of~~ opens the perticular page. This gives an illusion that the data is loaded instaneously. Using this technique The problem of inconsistency in infrastructure configuration across multiple environments could be solved using the Infrastructure as code (IaC) techniques.

Q.No.

2.a. Subversion (SVN) and Git :-

i) Key architectural differences :-

- Subversion (SVN) system is a premitive file based system.
  - Stores the each version as a folder.
  - limited offline control.
  - lacks abilities like conflict resolution and rollbacks.
- Git is an open-source, version-control software.
  - Each version can be stored as a new Branch
  - unlimited offline repository control
  - Have special abilities ~~for~~ like conflict resolution and instant rollbacks.

ii) Impact on teams :-

Using Subversion (SVN) or Git will greatly impact the way teams use these tools for version control and collaboration.

a) While Using SVN :-

→ ~~As~~ SVN is a primitive

Q.No.

file based version contol system. Users or teams may face access or deadlock issues while trying to access a perticulal version.

⇒ SVN is a centerlized system which could be vanarable to the attacker, if not maintained properly the data could be corrupted or trashed.

b.) while using Git :-

→ Git is a distributed system, so the owner has authority to grant or revoke access to perticular person in a team.

→ Git is highly secure and need proper authentication.

- Hence considering the collaboration between the team members, using git will improve and simplifies the complexity of code sharing and code tracking.

Q.No.

2.b **Bug hunting and reverting changes:-**

Considering the case that a critical bug was pushed to main-branch in the latest commit, assuming main-branch as a production branch.

The first thing I do after knowing the bug in the production, I will roll back the production to the previous commit. As it is certain that the bug was introduced in the latest commit. Git simplifies the ~~task~~ task, like activity monitoring and changes monitoring. After reverting the change back to the previous commit, the production will continue to serve the previous version. Now our next task is to locate the module that caused the major Bug. Then we need to fix it. And run the tests to be ~~certain~~ certain the bug has been resolved and every thing ~~work~~ works as expected. Next step is damage control. major Bugs as these could cause huge monetary damages. Go through the system logs and

Q.No.

Try to rollback the changes made in the database by the Bug. When you are fully certain there aren't any issues with the product, push to the main-branch.

It is also very easy to track changes in the each commit. Git maintain a record of each and change that a person has made, which makes it easy to locate and revert the change.

Q.No.

3.a Key components and Architecture of Docker :-

We can consider docker as a tool that simplifies the portability issues. While developing a project we will use many package and finally build the final code. That code is needed to be runned on various devices and different operating system. It is difficult to re-configure the project for each device, there is where docker comes in, ~~doc~~ using docker we can create an docker container which has an docker Image, that can be easly used on different devices with out any additional configuration.

Docker Engine :-

Docker engine is a ~~of~~ software application that can be downloaded in to your system, which provides CLI (command line Interface) to control / Manage the docker on your computer.

Docker Hub :-

Docker hub is a web-platform where you store your docker repositories which containe Containers and Images.

Containers :-

Container is a box, which can have docker images stored in it. the are used to run / manage / control images.

Docker Images :-

Docker images are binary files created by docker engine that consists your whole project encoded in it

Network and Volumes :-

The terms network and Volume used in the docker refers to a network of images called volumes which are stored on docker hub.

3.b Migrating from monolithic application to micro-service architecture :-

Monolithic application is an application that has only a single block interacting the user and the database. It don't have backend server api's to access data, it directly performs calculations and operates databse. and in micro-service architecture, there a many service blocks, and each service block will perform only one simple task.

So Inorder to migrate a monolithic application into to a microservice architecture appl-ication, using docker containers the following steps are to be done.

1, Writing Microservices

2, DB management using an ORM.

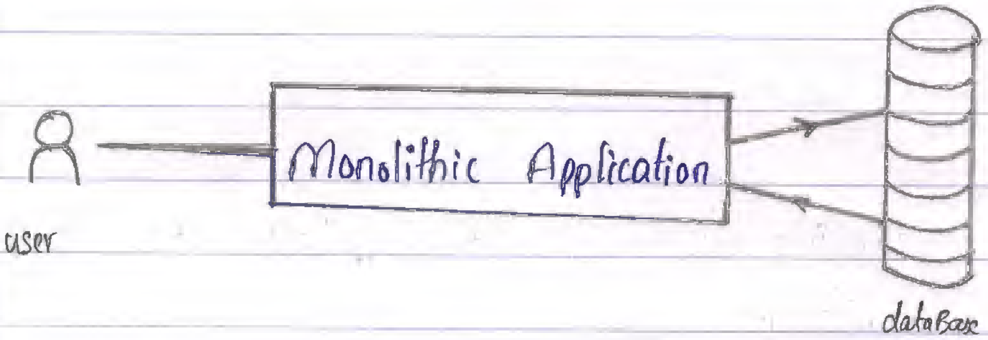
3, Testing Microservices

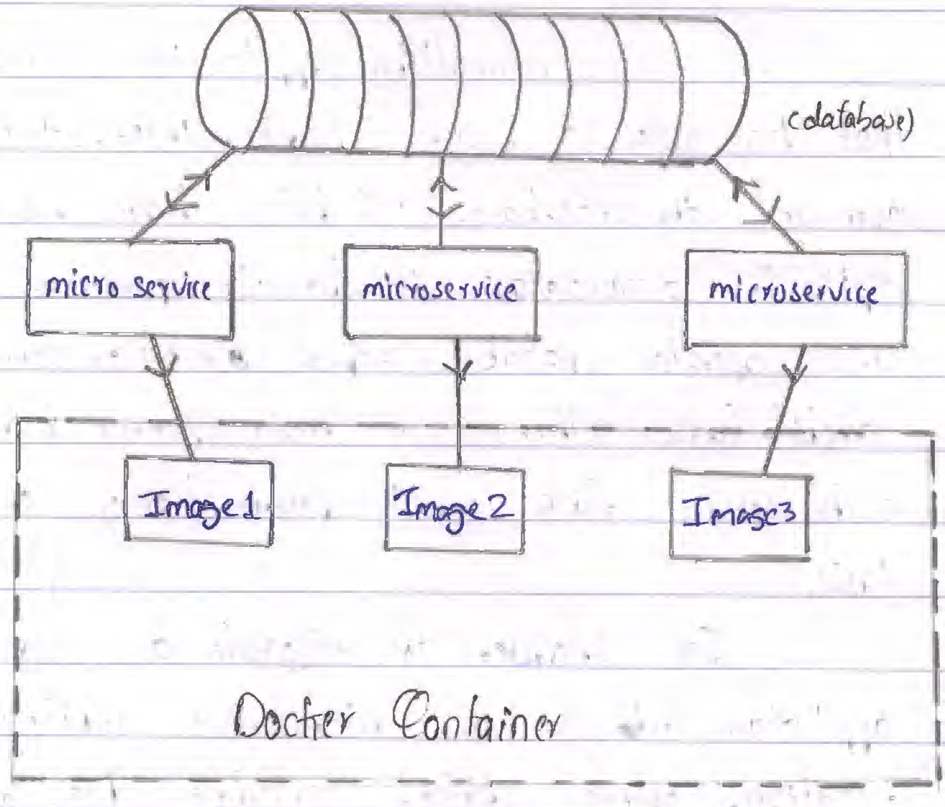
4, Converting MicroServices into Images

5, Managin docker Containers to run the created Images

6, Deploying.

Q.No.


Monolithic Application
user
dataBase


(database)
micro service
microservice
microservice
Image1
Image2
Image3
Docker Container

Q.No.

7.a. Manual Test plan for a Complex web Application :-

Let us consider a complex web application, ProTrack · a fullstack web application used for project management. It has role based access.

Test plan :-

1. Identify the user types
2. Identify the access of each user type
3. Prepare a Test case chart and include the above access and fail cases and pass cases.
4. Include edge cases.

Now let us see the use case for the manual testing. There are total of 5-types of users { User, admin, super-admin, org_manager, org-user }. these are their access

	Create_Org	Create prj	allow_user	manage prj	add user	manage user
user	F	T	F	T	F	F
admin	T	F	F	F	F	T
super admin	T	T	T	T	T	T
org_manager	F	T	T	T	T	T
org user	F	F	F	F	F	F

Q.No.

Based on the table above, we can write a Test table like

Testcourid	Sceanario	UsrType	expected	actual	state
1.	create_org	User	F	F	P
2.	Create org	admin	T	T	P
3	Create Org	Manage.	F	F	P

Manually open the site and login as different roles and validate the test can table, to find any bugs.

Q.No.

7b. Comprensive Testplan for a new feature =

To develop a comprehensive test plan to ensure full coverage of a new feature in an application, It is mandatory to include edge cases and potential failure points to make shure there arent and bugs and it ir read to be produced.

Firstly we should read the requirement of the projects new feature to understand it functionality. It helps to find the edge case while testing. Perform unit testing, integration testing, performance testing with the test cases included with the edge cases. Also use Top-down and ~~Bed~~ Bottom-up approches to make shure there aren't any bugs and unexpected behaviour of the feature.

Making shure the feature meets the requirements and work without any issue is our main priority to develop a comprehensive test plank to ensure the full coverage of the new features in an application

6.a Amazon Elastic Computing is a service provided by Amazon Web Services (AWS), Which will provide a virtual ~~of~~ machine on AWS cloud. There are types of EC2 service, like reseverd instance, on-demand instance, spot instance. It is important to choose the most-sutable package model based on the usage, If not we will be seeing unexpected large number on our Monthly billing,

→ EC2 Reserved :- this instance is not scalable and sutable for mini-server where load is not expected

⇒ On-demand Instance : Here it is scalable based on the traffic the memory and bandwidth increases

→ Spot-on Instance -

Here the ~~serv~~ server is already scaled with top hardware, this is efficient if you app expects continous traffic.

Q.No.

6b. Advantages of deploying in Docker Container:-

→ Docker container are easily portable, no-need install packages and do the setup

→ Easy access with simplicity.

→ Easy application packaging

→ Simple dependecy management, as the docker image also have all needed packages

→ Easy to deploy accross different environment, like windows, linux and MacOS.

Q.No.

Q.No.

Q.No.

Q.No.

Q.No.

Q.No.

Q.No.	

Q.No.	

Q.No.

Q.No.	

Q.No.	

Q.No.

Q.No.

Q.No.	

Q.No.

Q.No.

ROUGH WORK

Content written here will not be considered for valuation