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**	7108

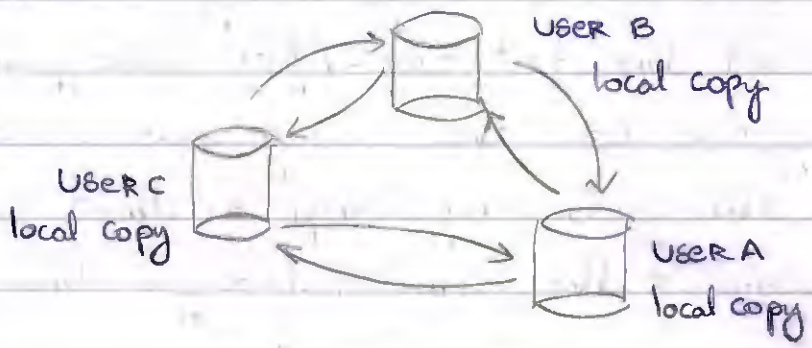
Instructions to Evaluators

- 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) Git is a distributed version control system. There are various git operations we perform like branching and merging, commiting etc. We use git for its distributed version control. Unlike SVN, git isnot a centralised system. Git is a open source software. Git also works offline and each users can have their own copy that they can work on.



Branching in git refers to when in the code repository, we branch out the code into our local copy so that we can review and edit the code. Branching helps to review and edit the code present in the repository without disturbing the pipeline of operations and we can easily edit the code. Using branching, the user can their own local copy on their computer and build new features or operations

Q.No.

and test it. Without branching, if we directly commit the changes, it may crash our application. Therefore branching ensure high uptime. Merging in Git refers to the process where we add/merge our code to the central repository and commit the changes. There are different processes for merging. First, we need to test our code and it is only ready when it passes all the testcases. After successfully passing, we can then merge our code with the central repository so that we can commit the required changes, this is the process of merging in git. To ensure the process stays efficient and smooth, we employ best practices for managing branches and resolving conflicts. To manage the branches, we need to remove all the unnecessary branches and clean up the space. Then we need to define the objective of each branch so it can be organised and ensure the management of branches. We need to bundles the branches together so that they are more

Q.No.

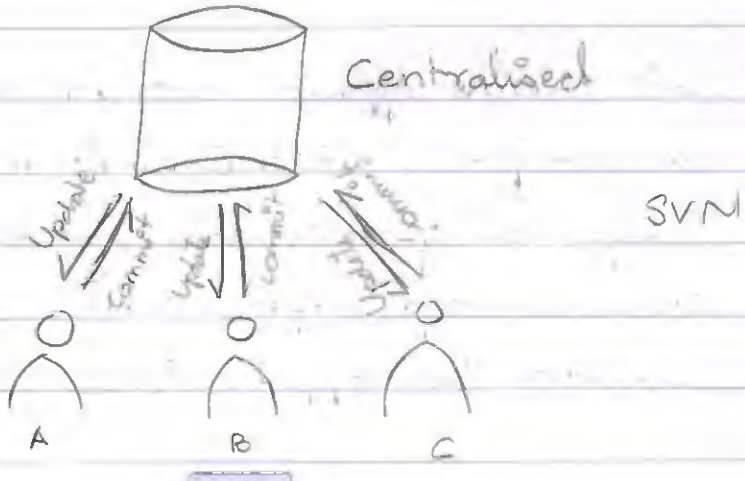
easy to manage. With respect to resolving conflicts, we first need to identify where the conflicts arises and we need to find the cause of the conflict. We then need to diagnose it and resolve the conflict by appropriate measures.

1(b) If my team is struggling in maintaining consistency in infrastructure configurations across multiple environments, I would implement infrastructure as Code (IaC) to address the given challenge. Infrastructure as Code (IaC) is a revolutionary service that provides the customers with the necessary infrastructure in the form of code so that there is no need of developing and building. It provides the necessary infrastructure across different environments. It ensures that the infrastructure stays consistent across multiple environments. There are multiple reasons that may arise for my team which is struggling with maintaining consistency. Building and scaling an application is hard in and on itself but also maintaining consistency across multiple

Q.No.

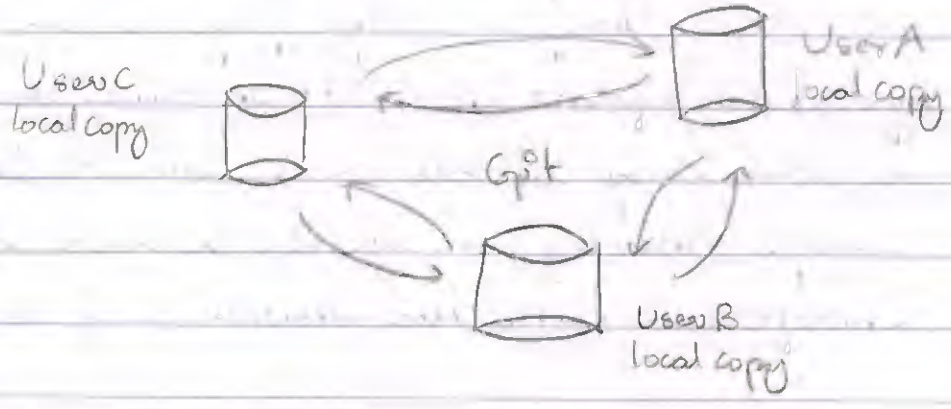
environments is very hard. Therefore, we implement infrastructure as code to address the challenge. Such challenges are caused due to a variety of reasons, including, since the infrastructure configurations are different at different environments, it becomes a challenge. So, we implement infrastructure as code. The infrastire as code (IaC) address this problem by providing our team with the necessary infrastire to build our applications on, without need to build on our own. It provides the infrastructure, in the form of code which, we can then integrate in our application. It also ensures that the infrastructure, it also maintains consistency across multiple different environments. Infrastructure as Code (IaC) allows our team to easily scale, build and maintain our applications with relative ease. It also helps a lot in maintaining the different infrastructure configurations so that those configurations are consistent across the different

Q.No.	
	multiple environments. Therefore, this is how I would implement infrastructure as Code (IaC) to address the given challenge.
2(a)	SVN or SubVersion is a centralised version system, whereas git is a distributed version control system. Both SVN and git are version control systems used for different purposes because of one key fundamental difference. SVN is a centralised version control system. In SVN, the repository is present in a centralised server. The user can't share their own copy as SVN restricts it. The users can therefore only perform limited operations update/commit. In SVN, therefore we can only update and commit the changes directly to the centralised server. Therefore, in SVN the downtime is very high.



Q.No.

SVN doesn't work offline and all the parties should be online for SVN to work. It is also slowers than git which works instantly. Therefore, this is about the architecture of SVN. Git is also a ~~subver~~ version control system but it is distributed. Git is a distributed system where each users can have their own copy of the repository. In git, the users can have a copy and perform different number of operations where they can done. Git is highly used nowadays since SVN is an outdated.



Git works offline and it also faster as compared with SVN. It is also more easy as it encourages cross-collaboration among different users. It also aids to test the application rigorously

Q.No.	
	before pushing and commiting final changes to the main repository. The architecture difference, centralised & distributed enable different teams to use different tools for Version Control and Collaboration. SVN is used to tightly control the development process and isn't very collaborative and used for legacy applications that don't require much changes. Git enables high collaboration among teams as each can have their own copy and perform necessary operations. It is also highly flexible and enables great functionality for version control.
2(b)	If a critical bug was introduced in the latest commit on the main branch, I would use git to identify the commit that introduced the bug. I will then try to revert the changes to maintain the integrity of the application. We first need to employ automated tools that can identify and report the errors like this critical bug to the development team. After releasing the bug and diagnosing

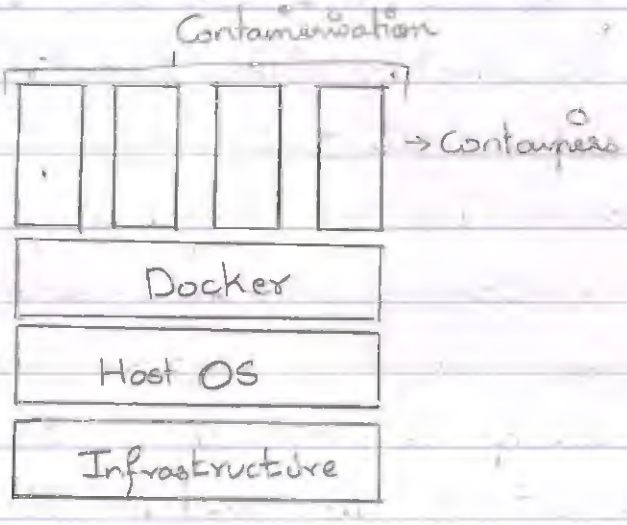
Q.No.

the critical bug, we need to look at how the bug has been introduced and we need to keep track of it in the log files. Since, git is a distributed version control system, we can easily track the commit that introduced the bug since it keeps track of the commits and changes made to the main branch. But before that, we need to shut down the servers before the bug can cause harm or sabotage the other parts of the application. After identifying the commit that introduced the bug, we need to clone all the repository and store in our local. We need to simultaneously work on the new update with rigorous testing that ensure that new bug or any bug is not formed in the code. We need to then isolate the branch and diagnose the bug. We then need to clean the repository and push the repository with the code that we cloned before the branch that introduced a critical bug in the main branch. This

Q.No.

way, we can eliminate the threat of a critical bug and can ensure that there is high uptime. Then we can neutralise the bug thus removing the critical bug. This is how I would use Git to identify the commit that introduced the bug and revert the changes.

3(a) Docker is an open source application that containerise the application and all its dependencies. Docker helps in the containerisation of application into a single unit. It is a lightweight resource and ensure consistency across multiple different environments.



Dockers, unlike virtual machines which have their own OS, will make use of the host's

Q.No.

operating system and therefore is lightweight in nature. The architecture of dockers is therefore lightweight, easy to implement and consistent across multiple environments since its dependencies are also containerised into a single unit. The key components of dockers are dockers engine, dockers hub, dockers images, containers, networks and volumes. The dockers engine is the engine that enable the use of scripts to containerise the different elements. It enables to create dockers images and containers. Dockers hub is a place where all the dockers images and files are stationed and it provides access to dockers files. Dockers images is a blueprint that is creates for the containers. Dockers images are designed so that containers can be made. Containers are the units that contain the application and all of its dependencies. It is the central component of the dockers. Networks are the routes or connections that established among the different components of

Q.No.	
	dockers. Volumes indicate the storage of the dockers since dockers is a lightweight architecture and it is consistent across multiple environments. This is about the key components of dockers.
3(b)	Monolithic application is the type of application in which all the infrastructure is built on a single layer. This type of application is devoid of any type of components. Monolithic applications are therefore easier to handle and maintain. It also requires low maintainance and also used in legacy applications, where there are less frequent updates and handles low amount of traffic. But as the team size increase and the functionality of the application increases, we need most robust infrastructure and need a new type of applications to not increase the load on monolithic applications. Therefore, we introduce microservices architecture using dockers containers. The microservices architecture is the type of architecture which breaksdown

Q.No.

the monolith application into smaller and individualised components. In microservices architecture, we define the tasks of each component and break down into smaller, independently deployable containers. In monolith application, the entire application works, and if we remove any component, it will not work. But in microservices architecture, the individual components are independently deployable. This is possible using docker container. Using the docker containers, after dividing and breaking down into small, independent components, we need to containerise each individual component along with dependencies. Thus we need to break them down into smaller, and ready to be deploy independent containers. We then make sure that these containers are compatible with each other and ensure that they work in conjunction with each other. Then we need to test the application and deploy it when ready. These are the steps

Q.No.	
	that I would take for breaking down the monolith in to smaller, independently deployable containers when my team is migrating from existing monolith application to microservices architecture using docker containers.
6(a)	EC2 or Elastic Compute Cloud is a service provided by the Amazon Web Services that provide the cloud services. EC2 is a service when we use the instances and AWS provides the cloud and server application where we have our own virtual machine in the cloud. We can ~~prop purchase~~ purchase instances in the cloud according to our requirements, and AWS provides different instances and they are - EC2 reserved instances, on-demand instance and spot instances. EC2 reserved instances is the type of instance provided by AWS where we purchase a fixed amount of instances. This model has a fixed pricing model and has a constant pricing. It is rigid and isn't reflective of the usage.

Q.No.

Reserved instances are used when we can estimate the demand and the demand is constant throughout the application. It is best used when the demand is relatively stable. On-demand instances are the type of instances provided by AWS where we can't predict the demand of our applications. In on-demand instance, the numbers of requested instances aren't constant and they differ based on the current trends. The pricing of the on-demand instances are more flexible since if there is few demand for our application, the numbers of instances created are less and therefore we can pay less. But it is highly beneficial when we need to scale and maintain our applications over large numbers of people. We can easily buy instances and scale-as-we-go without worrying about the reserved instances. Spot instances, on the other hand are the type of instances where if we reach the limit of our reserved instances, we can buy

Q.No.

the required instances on the spot. This is on the pricier side before the extra credits are costly and this is used where we aren't sure of the scale of our application and if we scale upto our reserved demand or not.

6(b) Docker containers are the units that containerise the application and all of its dependencies. Therefore, different components of the applications are containerised and broken down into deployable, independent containers. Therefore, there are many advantages of deploying applications in docker containers. Docker will help simplify the application packing, dependency management, and deployment across different environments. Docker will help in containerizing the components and then we can individually check if each component is working properly or not. Thus after ensuring that the individual containers are working properly independently of each other, it helps in the

Q.No.	
	application packaging where we can package different, individual components of dockers units. This helps in organising our application for a faster and a smoother deployment. Therefore, it also helps in the dependency management because not all dockers containers work individually so using dockers containers we can reduce the dependency by packaging the each components and their dependencies in containers thus no issue arises. Since the dockers containers are containerised and act as a single unit, it doesn't need to interact with the environment surrounding it and therefore work in its own ecosystem. Therefore, since it depends only on the host OS, it will work wherever we deploy, independently across the different environments.
7(a)	In a complex web application, we need to clearly outline and mention the objectives of creating a manual

Q.No.

test plan to ensure full coverage of the application with all the features. Firstly, we need to go in a phase by phase manner, one feature after an another. We need to check the fundamental or key features of the application and check whether they function properly or not. These key features are the heart and soul of such applications, therefore they must be thoroughly and rigorously tested. After the basic testing is complete, we need to check for edge cases and if it returns any errors or not. These edge cases are critical and can cause errors. Then we assign different teams to different part of the application and gather feedback and make the necessary changes. Therefore, these are the key elements that should be included, to ensure that the plan is thorough and effective.

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