

DEVOPS

QUESTION BANK

UNIT 1-INTRODUCTION TO DEVOPS

PART-A

1. What is the core principle of DevOps?
2. Name three major cloud service providers in the industry.
3. What role do version control systems play in software development?
4. Explain the difference between Git and GitHub.
5. How does DevOps integrate with cloud services like AWS, GCP, and Azure?
6. What are some common features offered by cloud service providers like AWS, GCP, and Azure?
7. Describe the benefits of using Git for version control.
8. How does GitHub facilitate collaboration among software development teams?
9. What is Infrastructure as Code (IaC), and how does it relate to DevOps?
10. Briefly explain the concepts of Continuous Integration (CI) and Continuous Deployment (CD) in the context of DevOps.

PART-B

1. Explain devops lifecycle in detail?
2. Explain in detail about Amazon Web Services?
3. What is GCP? Explain there services in detail.
4. Explain about Azure in detail?
5. What is Git? Explain its benefits in detail?
6. Explain github along with the important terms used by the developers?

UNIT 2-COMPILE AND BUILD USING MAVEN AND GRADLE

PART-A

1. What is Maven and what is its primary purpose in software development?
2. How do you install Maven on your local machine?
3. What does POM stand for in Maven, and what is its significance?
4. Can you name some of the build phases in Maven's lifecycle?
5. Explain the purpose of Maven profiles and how they are used.
6. Differentiate between local, central, and global Maven repositories.
7. What are Maven plugins, and how do they enhance the build process?
8. Describe the process of creating and building artifacts using Maven.
9. How does Maven handle dependency management in a project?
10. What are the steps to install Gradle, and how does it differ from Maven in terms of build management?

PART-B

1. Describe the purpose and importance of POM files in Maven. How do they define a project's configuration?
2. Can you explain the Maven Build lifecycle in detail, including each phase's purpose and the order in which they execute?
3. What are Maven profiles, and how can they be utilized to manage different configurations for various environments or purposes?
4. Explain the concept of Maven repositories, including the distinctions between local, central, and global repositories.
5. How do Maven plugins enhance the functionality of Maven? Provide examples of commonly used Maven plugins and their respective purposes.
6. Compare and contrast the dependency management mechanisms of Maven and Gradle.

UNIT-3 CONTINUOUS INTEGRATION USING JENKINS

PART-A

1. What is Jenkins, and what role does it play in the software development process?
2. Explain the architecture of Jenkins and its key components.
3. How do you install and configure Jenkins on a server?
4. Describe the process of creating a Jenkins job and its significance in a CI/CD pipeline.
5. What are some common configurations you can apply to a Jenkins job, and how do they affect the build process?
6. What is the purpose of Jenkins plugins, and how do you add them to a Jenkins instance?
7. Can you name some commonly used Jenkins plugins and explain their functionalities?
8. How do you configure Jenkins to work with Java, Git, and Maven for building projects?
9. What is the concept of a Jenkins build, and how does it relate to the execution of a job?
10. Explain the concept of Jenkins workspace and its role in the build process.

PART-B

1. What are the steps involved in installing and configuring Jenkins on a server?
2. Can you explain the architecture of Jenkins and how it facilitates continuous integration?
3. Walk me through the process of creating a Jenkins job from scratch.
4. What are the key configurations that can be applied to a Jenkins job, and how do they impact the build process?
5. How do you introduce plugins to Jenkins, and why are they important in extending Jenkins' functionality?
6. Can you provide an overview of commonly used Jenkins plugins such as Git Plugin, Parameter Plugin, HTML Publisher, Copy Artifact, and Extended Choice Parameters?
7. What steps are involved in configuring Jenkins to work seamlessly with Java, Git, and Maven?

8. Describe the process of creating a Jenkins build and how it interacts with the configured job settings.
9. Explain the concept of Jenkins workspace and its significance in the build process.
10. How do you ensure that Jenkins is effectively integrated with version control systems like Git for continuous integration purposes?

UNIT-4 CONFIGURATION MANAGEMENT USING ANSIBLE

PART-A

1. What is Ansible and how does it differ from other configuration management tools?
2. Explain the basic steps involved in installing Ansible on a system.
3. What is the purpose of Ansible master/slave configuration, and how does it facilitate automation?
4. Describe the role of YAML in Ansible and provide examples of basic YAML syntax.
5. How do Ansible modules contribute to the automation process?
6. What are Ansible Inventory files, and how do they help in organizing hosts and groups?
7. Define Ansible playbooks and explain their role in orchestrating tasks across multiple hosts.
8. What are Ansible roles, and how do they aid in modularizing configurations for reuse and maintainability?
9. How can ad-hoc commands be used in Ansible for quick management tasks?
10. Discuss the benefits of using Ansible for configuration management in comparison to traditional methods.

PART-B

1. How does Ansible handle configuration management, and what makes it a popular choice for automation tasks?
2. Can you provide a step-by-step guide to installing Ansible on various operating systems?
3. What is the significance of master/slave configuration in Ansible, and how does it impact the automation process?
4. Explain the YAML basics essential for writing Ansible playbooks, including key-value pairs and indentation rules.
5. How do Ansible inventory files assist in organizing hosts and groups, and what are the best practices for managing them?
6. Discuss the role of Ansible playbooks in automating complex tasks across multiple hosts, including the structure and syntax used within playbooks.

UNIT-5 BUILDING DEVOPS PIPELINES USING AZURE

PART-A

1. What are the steps involved in creating a GitHub account?
2. How do you create a new repository on GitHub, and what are the essential settings to configure during this process?
3. Explain the process of creating an Azure organization and its significance in managing projects and resources.
4. What are the key components of a pipeline in Azure DevOps, and how do they contribute to the automation process?
5. Can you describe the steps required to build a sample code using Azure Pipelines?
6. What is the purpose of the azure-pipelines.yaml file, and how does it influence the configuration of a pipeline?
7. How do you modify the azure-pipelines.yaml file to customize the build and deployment process for your project?
8. What are the different stages that can be defined within an Azure Pipeline, and how do they help in organizing and executing tasks?
9. Explain the role of triggers in Azure Pipelines and how they can be configured to initiate pipeline runs automatically.
10. How do you integrate GitHub repositories with Azure Pipelines to trigger builds upon code commits or pull requests?

PART-B

1. What are the prerequisites for creating a GitHub account, and what information is typically required during the registration process?
2. Can you explain the steps involved in creating a new repository on GitHub, including the options available for configuring repository settings?
3. How does creating an Azure organization differ from creating a project, and what are the benefits of organizing resources within an organization?
4. Walk me through the process of creating a new pipeline in Azure DevOps, highlighting the different types of pipelines available and their use cases.
5. What steps are involved in building a sample code using Azure Pipelines, and how can you customize the build process to suit the requirements of your project?
6. Discuss the structure and key components of the azure-pipelines.yaml file, and provide examples of how it can be modified to define various stages and tasks within a pipeline.