Assignment No. 2

Q1. What is the use of SSH?

Ans:- The **SSH (Secure Shell)** is an access credential that is used in the SSH Protocol. In other words, it is a cryptographic network protocol that is used for transferring encrypted data over the network. The port number of SSH is 22. It allow users to connect with server, without having to remember or enter password for each system. It always comes in key pairs:

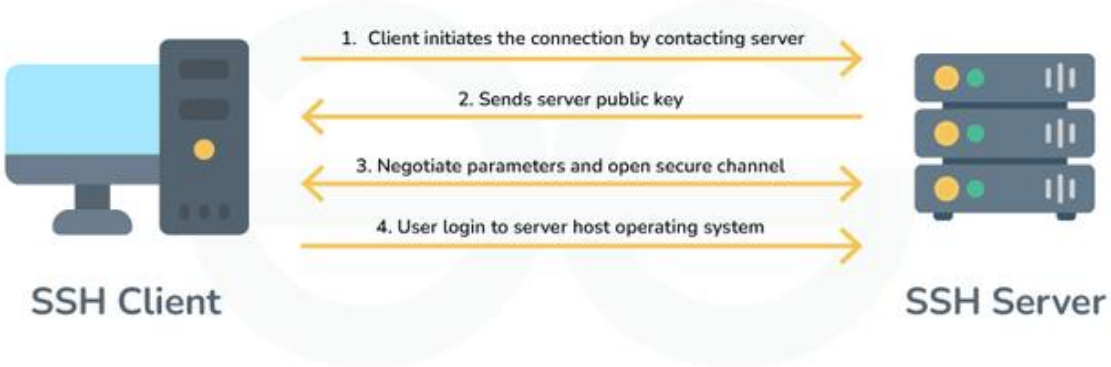
- **Public key –** Everyone can see it, no need to protect it. (for encryption function).
- **Private key –** Stays in computer, must be protected. (for decryption function).

Secure Shell or SSH, is a protocol that allows you to connect securely to another computer over an unsecured network. It developed in 1995. SSH was designed to replace older methods like Telnet, which transmitted data in plain text.

Imagine a system administrator working from home who needs to manage a remote server at a company data centre. Without SSH, they would have to worry about their login credentials being intercepted, leaving the server vulnerable to hackers. Instead of it after using SSH, the administrator establishes a secure connection that encrypts all data sent over the internet. They can now log in with their username and a private key, allowing them to safely execute commands on the server, transfer files, and make necessary updates, all of these without the risk of spying eyes watching their actions. This secure access is essential for maintaining the integrity of sensitive information of the company. SSH (Secure Shell) is an access credential that is used in the SSH Protocol. In other words, it is a cryptographic network protocol that is used for transferring encrypted data over the network.

Features of SSH

- **Encryption:** Encrypted data is exchanged between the server and client, which ensures confidentiality and prevents unauthorized attacks on the system.
- **Authentication:** For authentication, SSH uses public and private key pairs which provide more security than traditional password authentication.
- **Data Integrity:** SSH provides Data Integrity of the message exchanged during the communication.
- **Tunnelling:** Through SSH we can create secure tunnels for forwarding network connections over encrypted channels.



Q2. What is the Test Driven Development(TDD) ? Explain what are different Categories of Tests.

Ans:- Test-driven development (TDD) is a method of coding in which you first write a test and it fails, then write the code to pass the test of development, and clean up the code. This process recycled for one new feature or change. In other methods in which you write either all the code or all the tests first, TDD will combine and write tests and code together into one.

TDD is a software development methodology where developers write automated tests for a feature or function before writing the actual code, ensuring that the code meets specific requirements and is thoroughly tested throughout the development process

Test-Driven Development (TDD) is a method in software development where the focus is on writing tests before writing the actual code for a feature. This approach uses short development cycles that repeat to ensure quality and correctness.

Advantages of Test Driven Development (TDD)

- Unit test provides constant feedback about the functions.
- Quality of design increases which further helps in proper maintenance.
- Test driven development act as a safety net against the bugs.
- TDD ensures that your application actually meets requirements defined for it.
- TDD have very short development lifecycle.

Categories of Tests

Unit Tests - Test individual components or functions in isolation.

Integration Tests - Test interactions between multiple components or modules.

Functional Tests - Validate the application's functionality against requirements.

End-to-End (E2E) Tests - Test the entire system workflow from start to finish.

Performance Tests - Measure speed, scalability, and stability under load.

Regression Tests - Ensure new changes don't break existing features.

Security Tests - Check for vulnerabilities and threats.

Usability Tests - Assess the user-friendliness of the application.

Acceptance Tests - Validate the application meets business and customer needs.

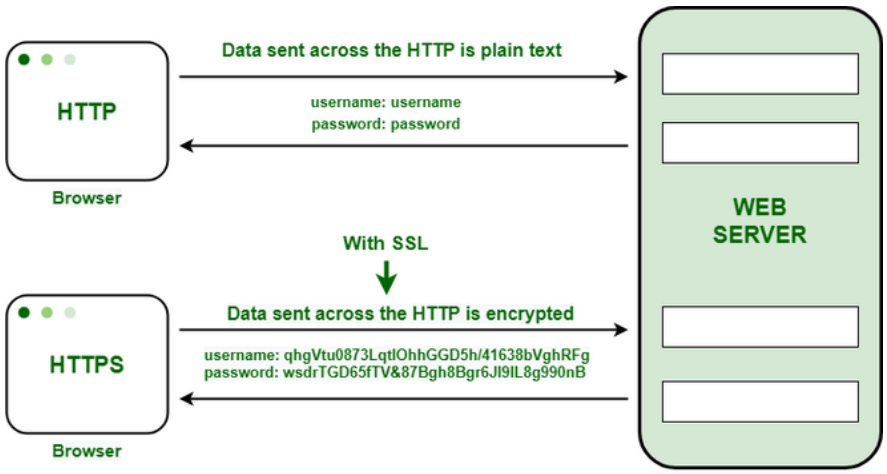
Q3) What is Hypertext Transfer Protocol Secure (HTTPS)?

Ans:- HTTPS stands for Hypertext Transfer Protocol Secure. It is the most common protocol for sending data between a web browser and a website. Hypertext Transfer Protocol Secure is a protocol that is used to communicate between the user browser and the website. It also helps in the transfer of data. It is the secure variant of HTTP. To make the data transfer more secure, it is encrypted. Encryption is required to ensure security while transmitting sensitive information like passwords, contact information, etc.

HTTPS establishes the communication between the browser and the web server. It uses the Secure Socket Layer (SSL) and Transport Layer Security (TLS) protocol for establishing communication. The new version of SSL is TLS(Transport Layer Security).

HTTPS uses the conventional HTTP protocol and adds a layer of SSL/TLS over it. The workflow of HTTP and HTTPS remains the same, the browsers and servers still communicate with each other using the HTTP protocol. However, this is done over a secure SSL connection. The SSL connection is responsible for the encryption and decryption of the data that is being exchanged to ensure data safety.

SSL:- The main responsibility of SSL is to ensure that the data transfer between the communicating systems is secure and reliable. It is the standard security technology that is used for encryption and decryption of data during the transmission of requests. As discussed earlier, HTT PS is basically the same old HTTP but with SSL. For establishing a secure communication link between the communicating devices, SSL uses a digital certificate called SSL certificate.



Advantages of HTTPS :-

- The main advantage of HTTPS is that it provides high security to users.
- Data and information are protected. So, it ensures data protection.
- SSL technology in HTTPS protects the data from third-party or hackers. And this technology builds trust for the users who are using it.
- It helps users by performing banking transactions.

Q4) What Is The Continuous Testing Process?

Ans:- Continuous testing is a process of automated testing done on software continuously as soon as a piece of code is delivered by the developers. This testing is done at every stage starting from the initial stages of development until the deployment of software. Continuous Testing is a software testing approach where automated tests are executed as part of the software delivery pipeline to ensure that the code meets quality standards at every stage of development. It integrates testing into the Continuous Integration (CI) and Continuous Delivery (CD) processes, allowing for faster feedback and identifying defects early in the development lifecycle.

Key Steps in Continuous Testing Process

- **Test Automation** : - Automate test cases for different types of testing (unit, integration, functional, etc.). Use frameworks like Selenium, JUnit, TestNG, or tools like Jenkins and CircleCI.
- **Integrate with CI/CD Pipeline** :- Trigger tests automatically with every code change, build, or deployment. This ensures immediate feedback on code quality.
- **Comprehensive Test Coverage :-** Cover all aspects of the application, including functionality, performance, security, and usability.
- **Parallel and Scalable Testing** :- Execute tests in parallel to reduce the time required for validation. Use cloud-based platforms or containerized environments for scalability.
- **Immediate Feedback Loop** :- Provide developers with real-time feedback on test results. This helps in identifying and fixing issues promptly.
- **Metrics and Reporting** :- Track key metrics like test pass rates, code coverage, and defect density. Use dashboards for visibility into the testing process.
- **Continuous Improvement** :- Refine test cases, update scripts, and improve coverage based on feedback and defect analysis.

Benefits of Implementing Continuous Testing

- More frequent releases and delivery of software.
- Risks are potentially reduced by performing testing from the early stages of development.
- Lower costs by identifying bugs at the initial stages, which saves the time and cost of changing later on.
- Higher product quality because of frequent testing.
- Easy implementation.
- The testing process may be made simpler, faster, and more dependable with the help of solutions that facilitate continuous testing.
- accelerate delivery to production and release more quickly.

Methodologies

- Test Automation
- Continuous Integration (CI)
- Continuous Delivery (CD)
- Shift-Left Testing

Q5) What Is Infrastructure As Code (Iac)?

Ans:- Infrastructure as Code (IaC) is a method of managing and provisioning IT infrastructure using code, rather than manual configuration. It allows teams to automate the setup and management of their infrastructure, making it more efficient and consistent. This is particularly useful in the DevOps environment, where teams are constantly updating and deploying software.

Features of IaC :-

Automation: IAC automates the provisioning and configuration of infrastructure, reducing manual errors and saving time.

Repeatability: IAC scripts can be used repeatedly, making it easy to recreate the same infrastructure in multiple environments.

Version Control: IAC code is stored in version control systems like Git, which makes it easy to track changes, revert to previous versions, and collaborate with others.

Scalability: IAC makes it easy to scale infrastructure up or down, adding or removing resources as needed.

Transparency: IAC makes the infrastructure transparent and understandable, as the code defines the infrastructure components and their relationships.

Improved Security: IAC helps ensure that infrastructure is configured consistently and securely, reducing the risk of security vulnerabilities.

Applications of IaC :-

- **Cloud computing:** IAC is widely used in cloud computing, where it can be used to provision and configure cloud resources, such as virtual machines, storage, and databases.
- **DevOps:** IAC is a key component of DevOps, where it is used to automate the deployment and management of infrastructure and applications.
- **Continuous integration and delivery (CI/CD):** IAC is used in CI/CD pipelines to automate the deployment and configuration of infrastructure and applications.
- **Networking:** IAC can be used to automate the deployment and management of networks, including creating and managing subnets, security groups, and firewalls.
- **Web application deployment:** IAC can be used to automate the deployment and management of web applications, including specifying the web server, application server, and load balancer.
- **Database deployment:** IAC can be used to automate the deployment and management of databases, including specifying the database engine, creating tables, and configuring users.
- **Big data:** IAC can be used to automate the deployment and management of big data infrastructure, including setting up clusters and configuring data processing frameworks such as Apache Hadoop or Apache Spark.

Q6) Explain the concept of branching and merging in Git.

Ans:- Branching and merging are two of the key concepts in Git that enable teams to collaborate on projects and manage code changes. In this blog, we'll discuss Git branching and merging, their benefits, and how to use them effectively.

Git Branching
Git branching is a technique in Git that allows developers to create multiple versions of their codebase, known as branches. Each branch is essentially a snapshot of the project's codebase at a specific point in time. Developers can create branches to experiment with new features or ideas without affecting the main codebase.

Creating a Branch
To create a new branch in Git, use the git branch command followed by the name of the new branch:

$ git branch new-branch

This creates a new branch named "new-branch." You can confirm that the new branch was created by running the ‘git branch’ command without any arguments:

```
$ git branch
  main
* new-branch
```

The ‘git branch’ command shows you all of the branches in your repository. The asterisk indicates the current branch.

Merge:-
To merge a branch into the current branch, use the git merge command followed by the name of the branch:

```
$ git checkout main
$ git merge new-branch
```

In some cases, Git may not be able to automatically merge the changes from two branches. This can happen when the same lines of code are changed in different ways on each branch. When this happens, Git will pause the merge and alert you to the conflict.
To resolve a merge conflict, you'll need to manually edit the code to merge the changes. Git will mark the conflicting lines of code with special markers that look like this:
<<<<<<< HEAD

This is the code from the current branch.
This is the code from the branch you're merging.
>>>>>>> new-branch

Q7) What is CAMS? Explain in detail.

The CAMS Model was created by Damon Edwards and John Willis. CAMS stands for Culture, Automation, Measurement, and Sharing. The CAMS model is a set of values used by many DevOps engineers.

Culture

Even though we use some pretty advanced technology and tools in DevOps, at the core of what we're trying to solve are problems related to people and business. Culture is defined by the interaction of people and groups and is driven by behaviour. Substantial communication improvement can result when there is a mutual understanding of others and their goals and responsibilities.

Traditional information technology business models split developers and operations into two distinct groups. They essentially spoke different languages as developers were tasked with innovating, creating and "moving fast and breaking things" whereas operations staff were in charge of maintaining stable environments and infrastructure. These competing goals often caused friction as each group accused the other of preventing them from doing their jobs. Changing the business culture by sharing responsibility and getting teams on the same page is a major goal of DevOps.

Automation

Automation can save time, effort, and money, just like culture, it truly focuses on people and processes and not just tools. The impact of implementing infrastructure as code as well as using continuous integration and continuous delivery pipelines can be magnified after understanding an organization's culture and goals. It helps to think of automation as an accelerator that enhance the benefits of DevOps as a whole.

Measurement

Using measurements will help in determining if progress is being made in the intended direction. There are two major bumps that may occur when using metrics. First, make sure the metrics you are the correct ones. Secondly, incentivize the right metrics. DevOps practices encourage you to "see the forest from the trees" by taking a look at the entire operation and assessing it as a whole and not just focusing on small parts. Major metrics include (but are certainly not limited to) income, costs, revenue, mean time to recovery, mean time between failures, and employee satisfaction.

Sharing

DevOps processes, similar to agile and scrum, place a very high premium on transparency and openness. Spreading knowledge helps to tighten feedback loops and enables the organization to continuously improve. This collective intelligence makes the team a more efficient unit and allows it to become greater than just the sum of its parts.

Q8) What's the difference between HTTP and HTTPS ?

HTTP	HTTPS
It stands for Hyper Text Transfer Protocol	It stands for Hyper Text Transfer Protocol Secure
They do not encrypt the text	They encrypt the code so that no one can access it
It does not require SSL at Transport Layer	They use Secure Socket Layer to encrypt the code.
Sender and receiver can understand the code . Even everybody who see the text can understand	Here sender and receiver can understand the code they can decipher the code and fetch the message
They do not require TLS or SSL	Here security is provided by TLS(Transport layer Security) and SSL(secure socket layer)
It is default protocol used at port no 80	It is not default protocol.
Here handshake schema is followed between sender and receiver and data transfer takes place	In this first secure connection is established to make sure data is transferred to right receiver then both sender and receiver agree upon cipher(decoding technique) then message is transferred.
It used port no 80	It uses port no 443
URL begins with http://	URL begins with https://
It is unsecure	It is safe transfer protocol
It does not require any validation	It requires validation like domain verification
It has simple address bar	It has green colored address bar that show that it is secure
It can be hacked	It cannot be attacked by hackers

Q9) What are types of Configuration management Tool. Specify names and their working?

Types of Configuration Management Tools

Configuration Management (CM) tools are used to automate the management, configuration, and deployment of systems, ensuring consistency and efficiency in IT operations. These tools can be categorized based on their approach and purpose. Below are the major types along with examples and their workings.

1. Infrastructure as Code (IaC) Tools

These tools manage and provision infrastructure through code, enabling repeatability and version control. Examples and Their Working

Terraform (by HashiCorp)

Type: Declarative

Working: Users define the desired state of infrastructure (e.g., servers, networks) in configuration files. Terraform creates, updates, or destroys resources to match the desired state using providers like AWS, Azure, or GCP.

Use Case: Provisioning virtual machines, load balancers, and databases in the cloud.

Pulumi
Type: Declarative/Imperative
Working: Similar to Terraform, but allows using general-purpose programming languages like Python or JavaScript to define infrastructure.
Use Case: Advanced configurations where logic-based infrastructure is needed.

2. Configuration Orchestration Tools
These tools automate the process of configuring systems and deploying software. Examples and Their Working
Ansible
Type: Agentless, Declarative
Working: Uses YAML-based playbooks to define configurations. It connects to servers via SSH and executes tasks without requiring any agent on the target system.
Use Case: Managing configurations, deploying applications, and automating tasks on multiple servers.
Chef
Type: Agent-based, Imperative
Working: Users write configuration recipes in Ruby that define the desired system state. The Chef server communicates with client agents on nodes to apply the configurations.
Use Case: Managing complex infrastructure with dependencies.
Puppet
Type: Agent-based, Declarative
Working: Uses a master-agent model where the master server pushes configurations defined in manifests (written in Puppet DSL) to agents installed on target systems.
Use Case: Large-scale IT environments requiring consistent configurations.

3. Version Control Systems (VCS)
These tools track changes to configuration files and scripts, ensuring versioning and collaboration. Examples and Their Working
Git
Type: Distributed
Working: Tracks changes to files, enabling teams to collaborate on infrastructure code and configurations. It supports branching, merging, and rollback to previous versions.
Use Case: Storing and managing IaC files like Terraform configurations or Ansible playbooks.
Subversion (SVN)
Type: Centralized
Working: Maintains a central repository to track changes. Teams commit changes to this repository for collaboration.
Use Case: Legacy systems where centralized versioning is preferred.

4. Container Orchestration Tools
These tools manage containerized applications and their configurations.
Examples and Their Working
Kubernetes
Type: Declarative

Working: Automates the deployment, scaling, and management of containerized applications. Configuration files define the desired state of pods, services, and other resources.
Use Case: Managing large-scale containerized microservices architectures.

Docker Compose

Type: Declarative
Working: Defines and runs multi-container Docker applications using YAML files. Users can specify networks, volumes, and dependencies.
Use Case: Local development and small-scale container orchestration.

5. Build and Release Management Tools

These tools focus on automating the build, test, and deployment process.
Examples and Their Working

Jenkins

Type: Continuous Integration/Continuous Deployment (CI/CD) Tool
Working: Automates building, testing, and deploying code. Users define pipelines using declarative or scripted methods.
Use Case: CI/CD pipelines for infrastructure and application delivery.

Azure DevOps

Type: Cloud-Based CI/CD
Working: Combines version control, build pipelines, and release management. Configurations for pipelines and releases are stored as code.
Use Case: Managing DevOps workflows in Azure environments.

6. Cloud-Specific CM Tools

These are tailored for managing cloud-based infrastructure and configurations.
Examples and Their Working

AWS CloudFormation

Type: Declarative
Working: Users define AWS infrastructure resources in JSON or YAML templates. CloudFormation creates and manages these resources based on the template.
Use Case: Automating AWS resource provisioning.

Google Cloud Deployment Manager

Type: Declarative
Working: Uses YAML or Python templates to define resources. Deployment Manager provisions these resources on Google Cloud.
Use Case: Managing Google Cloud infrastructure.

Q10) What is git Cheat Sheet?

Ans:- Git Cheat Sheet is a comprehensive quick guide for learning Git concepts, from very basic to advanced levels. By this Git Cheat Sheet, our aim is to provide a handy reference tool for both beginners and experienced developers/DevOps engineers. This Git Cheat Sheet not only makes it easier for newcomers to get started but also serves as a refresher for experienced professionals.

In this Git Cheat Sheet, we've covered all the basics to advanced Git commands that the developers required during the development and deployment process. Moreover, it is well-structured and categorized according to different use cases. It includes Git and GitHub, Git download, Git Configuration & Setup, Git commands, Git bash, Creating and Getting Git Projects, Git Snapshots, Branching and Merging in Git, Sharing and Updating in Git, Git Comparison, Managing History in Git, and more.

A "Git cheat sheet" is a quick reference guide that lists the most commonly used Git commands, allowing developers to easily look up and recall how to perform various actions within a Git repository without having to memorize every detail, essentially acting as a handy reminder for common Git operations like creating branches, committing changes, pulling updates, or pushing to a remote server.

Key points about Git cheat sheets:

Function:

They provide a concise summary of essential Git commands, including their syntax and usage scenarios.

Benefit:

Saves time by allowing developers to quickly access necessary commands without needing to constantly consult documentation.

Content:

Usually includes basic commands like git init, git add, git commit, git push, git pull, git branch, git checkout, and more depending on complexity.

Assignment No. 3

Q1. Explain Source code Management?

Ans:- Source code management (SCM) is used to track modifications to a source code repository. SCM tracks a running history of changes to a code base and helps resolve conflicts when merging updates from multiple contributors. SCM is also synonymous with Version control.

As software projects grow in lines of code and contributor head count, the costs of communication overhead and management complexity also grow. SCM is a critical tool to alleviate the organizational strain of growing development costs.

The importance of source code management tools:

When multiple developers are working within a shared codebase it is a common occurrence to make edits to a shared piece of code. Separate developers may be working on a seemingly isolated feature; however, this feature may use a shared code module. Therefore developer 1 working on Feature 1 could make some edits and find out later that Developer 2 working on Feature 2 has conflicting edits.

Before the adoption of SCM this was a nightmare scenario. Developers would edit text files directly and move them around to remote locations using FTP or other protocols. Developer 1 would make edits and Developer 2 would unknowingly save over Developer 1's work and wipe out the changes. SCM's role as a protection mechanism against this specific scenario is known as Version Control.

SCM brought version control safeguards to prevent loss of work due to conflict overwriting. These safeguards work by tracking changes from each individual developer and identifying areas of conflict and preventing overwrites. SCM will then communicate these points of conflict back to the developers so that they can safely review and address.

This foundational conflict prevention mechanism has the side effect of providing passive communication for the development team. The team can then monitor and discuss the work in progress that the SCM is monitoring. The SCM tracks an entire history of changes to the code base. This allows developers to examine and review edits that may have introduced bugs or regressions.

The benefits of source code management:

In addition to version control SCM provides a suite of other helpful features to make collaborative code development a more user friendly experience. Once SCM has started tracking all the changes to a project over time, a detailed historical record of the projects life is created. This historical record can then be used to 'undo' changes to the codebase. The SCM can instantly revert the codebase back to a previous point in time. This is extremely valuable for preventing regressions on updates and undoing mistakes.

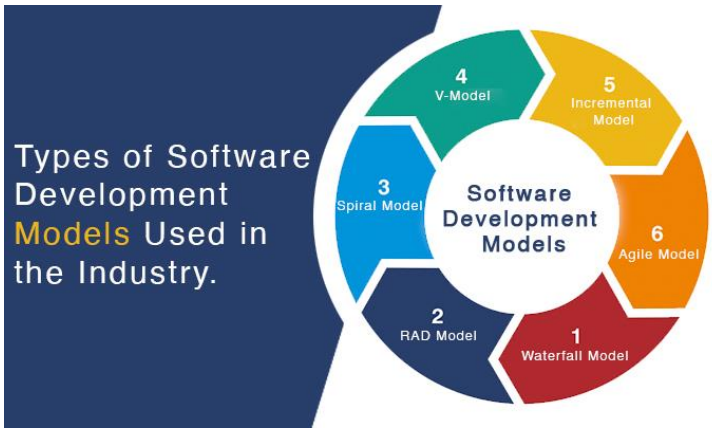
The SCM archive of every change over a project's life time provides valuable record keeping for a project's release version notes. A clean and maintained SCM history log can be used interchangeably as release notes. This offers insight and transparency into the progress of a project that can be shared with end users or non-development teams.

SCM will reduce a team's communication overhead and increase release velocity. Without SCM development is slower because contributors have to take extra effort to plan a non-overlapping sequence of develop for release. With SCM developers can work independently on separate branches of feature development, eventually merging them together.

Q2. What Are Different SDLC Models Available?

Ans:- Software Development life cycle (SDLC) is a spiritual model used in project management that defines the stages include in an information system development project, from an initial feasibility study to the maintenance of the completed application.

There are different software development life cycle models specify and design, which are followed during the software development phase. These models are also called "Software Development Process Models." Each process model follows a series of phase unique to its type to ensure success in the step of software development.



Waterfall Model

The waterfall is a universally accepted SDLC model. In this method, the whole process of software development is divided into various phases. The waterfall model is a continuous software development model in which development is seen as flowing steadily downwards (like a waterfall) through the steps of requirements analysis, design, implementation, testing (validation), integration, and maintenance.

RAD Model

RAD or Rapid Application Development process is an adoption of the waterfall model; it targets developing software in a short period. The RAD model is based on the concept that a better system can be developed in lesser time by using focus groups to gather system requirements.

Spiral Model

The spiral model is a **risk-driven process model**. This SDLC model helps the group to adopt elements of one or more process models like a waterfall, incremental, waterfall, etc. The spiral technique is a combination of rapid prototyping and concurrency in design and development activities.

Each cycle in the spiral begins with the identification of objectives for that cycle, the different alternatives that are possible for achieving the goals, and the constraints that exist. This is the first quadrant of the cycle (upper-left quadrant).

The next step in the cycle is to evaluate these different alternatives based on the objectives and constraints. The focus of evaluation in this step is based on the risk perception for the project.

The next step is to develop strategies that solve uncertainties and risks. This step may involve activities such as benchmarking, simulation, and prototyping.

V-Model

In this type of SDLC model testing and the development, the step is planned in parallel. So, there are verification phases on the side and the validation phase on the other side. V-Model joins by Coding phase.

Agile Model

Agile methodology is a practice which promotes continues interaction of development and testing during the SDLC process of any project. In the Agile method, the entire project is divided into small incremental builds. All of these builds are provided in iterations, and each iteration lasts from one to three weeks.

Any agile software phase is characterized in a manner that addresses several key assumptions about the bulk of software projects:

1. It is difficult to think in advance which software requirements will persist and which will change. It is equally difficult to predict how user priorities will change as the project proceeds.
2. For many types of software, design and development are interleaved. That is, both activities should be performed in tandem so that design models are proven as they are created. It is difficult to think about how much design is necessary before construction is used to test the configuration.
3. Analysis, design, development, and testing are not as predictable (from a planning point of view) as we might like.

Prototype Model

The prototyping model starts with the requirements gathering. The developer and the user meet and define the purpose of the software, identify the needs, etc.

A 'quick design' is then created. This design focuses on those aspects of the software that will be visible to the user. It then leads to the development of a prototype. The customer then checks the prototype, and any modifications or changes that are needed are made to the prototype.

Q3. Explain How Devops Is Helpful For Developers?

Ans:- DevOps significantly benefits developers by automating repetitive tasks like building, testing, and deployment, allowing them to focus on writing code, identifying issues early through continuous integration, receiving faster feedback loops, and collaborating more effectively with operations teams, ultimately leading to faster development cycles and higher quality software delivery.

Key ways DevOps helps developers:

Continuous Integration (CI):

Automatically merges code changes into a shared repository, triggering automated builds and tests, which helps catch issues early in the development process and prevents integration problems.

Faster Feedback Loops:

Enables rapid testing and deployment, providing developers with quick feedback on their code changes, allowing them to iterate and fix issues faster.

Reduced Manual Work:

Automates many routine tasks like environment setup, configuration management, and deployment, freeing developers to focus on core development activities.

Improved Collaboration:

Fosters better communication between development and operations teams, enabling faster problem-solving and smoother deployments.

Early Bug Detection:

Through automated testing at every stage of the development lifecycle, developers can identify and fix bugs early on, improving software quality.

Increased Productivity:

By streamlining the development process and minimizing manual interventions, developers can deliver software more efficiently.

Infrastructure as Code (IaC):

Allows developers to manage infrastructure through code, making it easier to provision and scale environments for development and testing.

Monitoring and Logging:

Provides real-time visibility into application performance, allowing developers to quickly identify and address issues in production environments

Q4. Explain how you can set up a Jenkins job?

Ans:- To set up a Jenkins job, you need to access the Jenkins web interface, create a new item, choose the project type (like a freestyle project or a pipeline), configure source code management (like Git), define build steps (commands to compile, test, and deploy your code), and optionally set up triggers for automatic builds based on code changes or a schedule.

Key steps to setting up a Jenkins job:

1. Access Jenkins:

- Open your web browser and navigate to your Jenkins server URL.
- Log in with your credentials if required.

2. Create a New Job:

- On the Jenkins dashboard, click "New Item" in the left menu.
- Enter a descriptive name for your job and select the project type (e.g., "Freestyle project" for basic builds, "Pipeline" for more complex workflows).
- Click "OK" to proceed to the configuration page.

3. Configure Source Code Management:

- In the "General" section, provide a description (optional).
- Navigate to the "Source Code Management" section.
- Select your version control system (e.g., Git, SVN).
- Enter the repository URL and credentials if necessary.

4. Add Build Steps:

- Go to the "Build Steps" section.
- Depending on your project, you might add steps like:
    - Execute shell commands: Run shell scripts to compile your code, run tests, etc.
    - Execute batch command (Windows): Run batch scripts on Windows systems.
    - Invoke Ant: Use Apache Ant build tool.
    - Invoke Maven: Use Apache Maven for Java projects.
    - Build a Pipeline: Define a pipeline using Jenkins pipeline syntax if you need a more complex workflow.

5. Optional: Set Build Triggers:

- Go to the "Build Triggers" section:
- Poll SCM: Automatically trigger builds based on changes in your source code repository at specified intervals.
- Build periodically: Set a cron-like schedule to trigger builds at specific times.
- Webhooks: Configure webhooks from your code repository to trigger builds on commits.

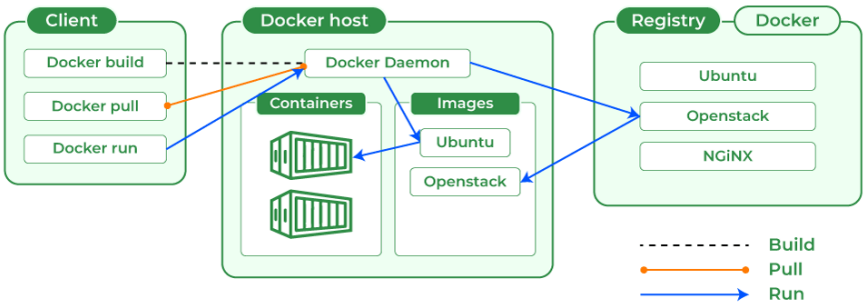
6. Save and Test:

- Click "Save" to apply your changes.
- To test your job, click "Build Now" on the job's page.
- Monitor the build process in the console output to check for errors.

Q5. Explain The Architecture Of Docker?

Ans:- Docker makes use of a client-server architecture. The Docker client talks with the docker daemon which helps in building, running, and distributing the docker containers. The Docker client runs with the daemon on the same system or we can connect the Docker client with the Docker daemon remotely. With the help of REST API over a UNIX socket or a network, the docker client and daemon interact with each other.

Docker’s architecture allows for efficient container management and deployment. If you’re looking to dive deeper into how Docker fits within the broader DevOps pipeline, the DevOps Engineering – Planning to Production course covers Docker’s architecture and how it integrates into DevOps workflows



Docker Client

With the help of the docker client, the docker users can interact with the docker. The docker command uses the Docker API. The Docker client can communicate with multiple daemons. When a docker client runs any docker command on the docker terminal then the terminal sends instructions to the daemon. The Docker daemon gets those instructions from the docker client withinside the shape of the command and REST API's request.

The main objective of the docker client is to provide a way to direct the pull of images from the docker registry and run them on the docker host. The common commands which are used by clients are docker build, docker pull, and docker run.

Docker Host

A Docker host is a type of machine that is responsible for running more than one container. It comprises the Docker daemon, Images, Containers, Networks, and Storage.

Docker Registry

All the docker images are stored in the docker registry. There is a public registry which is known as a docker hub that can be used by anyone. We can run our private registry also. With the help of docker run or docker pull commands, we can pull the required images from our configured registry. Images are pushed into configured registry with the help of the docker push command.

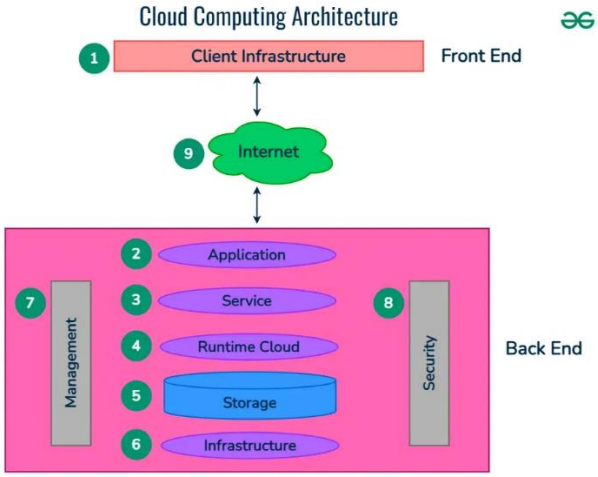
Q6. What Is Cloud Computing?

Cloud Computing means storing and accessing the data and programs on remote servers that are hosted on the internet instead of the computer's hard drive or local server. Cloud computing is also referred to as Internet-based computing, it is a technology where the resource is provided as a service through the Internet to the user. The data that is stored can be files, images, documents, or any other storable document.

The following are some of the Operations that can be performed with Cloud Computing

- Storage, backup, and recovery of data
- Delivery of software on demand
- Development of new applications and services
- Streaming videos and audio

Architecture Of Cloud Computing



1. Front End (User Interaction Enhancement)

The User Interface of Cloud Computing consists of 2 sections of clients. The Thin clients are the ones that use web browsers facilitating portable and lightweight accessibilities and others are known as Fat Clients that use many functionalities for offering a strong user experience.

2. Back-end Platforms (Cloud Computing Engine)

The core of cloud computing is made at back-end platforms with several servers for storage and processing computing. Management of Applications logic is managed through servers and effective data handling is provided by storage. The combination of these platforms at the backend offers the processing power, and capacity to manage and store data behind the cloud.

3. Cloud-Based Delivery and Network

On-demand access to the computer and resources is provided over the Internet, Intranet, and Intercloud. The Internet comes with global accessibility, the Intranet helps in internal communications of the services within the organization and the Intercloud enables interoperability across various cloud services. This dynamic network connectivity ensures an essential component of cloud computing architecture on guaranteeing easy access and data transfer.

Types of Cloud Computing Services?

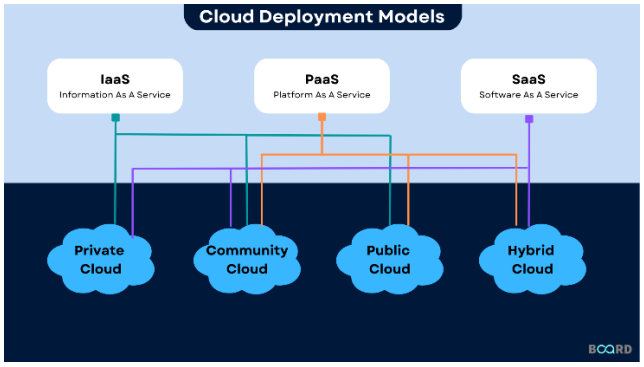
The following are the types of Cloud Computing:

1. Infrastructure as a Service (IaaS)
2. Platform as a Service (PaaS)
3. Software as a Service (SaaS)

Q7. Explain Cloud Deployment Architecture.

Ans:- Cloud Deployment Model functions as a virtual computing environment with a deployment architecture that varies depending on the amount of data you want to store and who has access to the infrastructure.

The cloud deployment model identifies the specific type of cloud environment based on ownership, scale, and access, as well as the cloud's nature and purpose. The location of the servers you're utilizing and who controls them are defined by a cloud deployment model. It specifies how your cloud infrastructure will look, what you can change, and whether you will be given services or will have to create everything yourself. Relationships between the infrastructure and your users are also defined by cloud deployment types.



Types Of Cloud Deployment Architecture

Public Cloud Deployments

Public servers that are accessible over the internet or through a VPN provider host public cloud installations. Every piece of gear, including networking equipment and VM hosts, is the responsibility of the service's owner. For businesses that do not want to make significant hardware and software investments, this makes operating an IT infrastructure far less burdensome. Popular public cloud systems also simplify the provisioning of services.

Private Cloud Deployments

A business may decide to host its cloud infrastructure on-site or in a data center. In both scenarios, the business typically controls the infrastructure. The company's own workforce creates and maintains the technologies that function on a private cloud. Only authorized users are able to access the resources of the private cloud due to rigorous access controls.

Community Cloud Deployments

A community cloud is basically a multi-tenant hosting platform that usually involves similar industries and complementary businesses. Community cloud deployments enable its users to save money by providing the infrastructure across several businesses. With the exception of locations where shared access has been decided upon and set up, data is still compartmentalized and kept private.

Hybrid Cloud

Similar to one another, cloud solutions incorporate several elements to meet a company's needs. One illustration would be a hybrid cloud that combines the affordability and simplicity of public clouds with the safety of private clouds. Private clouds may be used to store sensitive data while public clouds can be used for user services.

Q8. What Is AWS ? Explain In Brief.

Ans:- AWS stands for Amazon Web Services, It is an expanded cloud computing platform provided by Amazon Company. AWS provides a wide range of services with a pay-as-per-use pricing model over the Internet such as Storage, Computing power, Databases, Machine Learning services, and much more. AWS facilitates for both businesses and individual users with effectively hosting the applications, storing the data securely, and making use of a wide variety of tools and services improving management flexibility for IT resources.

How AWS Works?

AWS comes up with its own network infrastructure on establishing the datacentres in different regions mostly all over the world. Its global Infrastructure acts as a backbone for operations and services provided by AWS. It facilitates the users on creating secure environments using Amazon VPCs (Virtual Private Clouds). Essential services like Amazon EC2 and Amazon S3 for utilizing the compute and storage service with elastic scaling. It supports the dynamic scaling of the applications with the services such as Auto Scaling and Elastic Load Balancing (AWS ELB). It provides a good user-friendly AWS Management Console facilitating seamless configuration and management of AWS services to the Users. Its Architecture ensures high availability , fault tolerance making AWS as a versatile powerful Cloud Computing Platform.

Top AWS Services:-

In the rapid revolution of Cloud Computing, AWS facilitates with wide variety of services respect to the fields and needs. The following are the top AWS services that are in wide usage:

- **Amazon EC2(Elastic Compute Cloud) :** It provides the Scalable computing power via cloud allowing the users to run applications and manage the workloads over their remotely.
- **Amazon S3 (Simple Storage Service):** It offers scalable object Storage as a Service with high durability for storing and retrieving any amount of data.
- **AWS Lambda:** It is a service in Serverless Architecture with Function as a Service facilitating serverless computing i.e., running the code on response to the events, the background environment management of servers is handled by aws automatically. It helps the developers to completely focus on the logic of code build.
- **Amazon RDS (Relational Database Service):** This is an aws service that simplifies the management of database providing high available relational databases in the cloud.
- **Amazon VPC (Virtual Private Cloud):** It enables the users to create isolated networks with option of public and private expose within the AWS cloud, providing safe and adaptable configurations of their resources.

Advantages Of Amazon Web Services

- AWS allows you to easily scale your resources up or down as your needs change, helping you to save money and ensure that your application always has the resources it needs.
- AWS provides a highly reliable and secure infrastructure, with multiple data centers and a commitment to 99.99% availability for many of its services.

Q9. Explain version control system with example.

Ans:- Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done in the code. As we know that a software product is developed in collaboration by a group of developers they might be located at different locations and each one of them contributes to some specific kind of functionality/features. So in order to contribute to the product, they made modifications to the source code(either by adding or removing). A version control system is a kind of software that helps the developer team to efficiently communicate and manage(track) all the changes that have been made to the source code along with the information like who made and what changes have been made. A separate branch is created for every contributor who made the changes and the changes aren't merged into the original source code unless all are analysed as soon as the changes are green signalled they merged to the main source code. It not only keeps source code organized but also improves productivity by making the development process smooth.

Basically Version control system keeps track on changes made on a particular software and take a snapshot of every modification. Let's suppose if a team of developer add some new functionalities in an application and the updated version is not working properly so as the version control system keeps track of our work so with the help of version control system we can omit the new changes and continue with the previous version.

Types of Version Control Systems:

- **Local Version Control Systems**
- **Centralized Version Control Systems**
- **Distributed Version Control Systems**

Local Version Control Systems:

It is one of the simplest forms and has a database that kept all the changes to files under revision control. RCS is one of the most common VCS tools. It keeps patch sets (differences between files) in a special format on disk. By adding up all the patches it can then re-create what any file looked like at any point in time.

Centralized Version Control Systems:

Centralized version control systems contain just one repository globally and every user need to commit for reflecting one's changes in the repository. It is possible for others to see your changes by updating.

Distributed Version Control Systems:

Distributed version control systems contain multiple repositories. Each user has their own repository and working copy. Just committing your changes will not give others access to your changes. This is because commit will reflect those changes in your local repository and you need to push them in order to make them visible on the central repository. Similarly, When you update, you do not get others' changes unless you have first pulled those changes into your repository.