# DevOps Assignment

**Set-1**

**1.Explain importance of Agile software development.**

1. Agile is an iterative software development approach that emphasizes collaboration and rapid delivery, while DevOps is a software delivery approach that breaks down silos between development and operations. Both are complementary practices that can improve efficiency and predictability in software development.

**Values of Agile Methodology:**

1)Individuals and Interactions over Processes and Tools.

2)Working Software over Comprehensive Documentation.

3)Customer Collaboration over Contract Negotiation.

4)Responding to Change over Following a Plan.

**Principles of the Agile Methodologies:**

The Agile Alliance defines twelve lightness principles for those who need to attain agility:

1) Our highest priority is to satisfy the client through early and continuous delivery of valuable computer software.

2) Welcome dynamic necessities, even late in development. Agile processes harness modification for the customer’s competitive advantage.

3) Deliver operating computer software often, from a pair of weeks to a couple of months, with a preference to the shorter timescale.

4) Business individuals and developers should work along daily throughout the project.

5) The build comes around actuated people. offer them the setting and support they have, and trust them to urge the task done.

6) The foremost economical and effective methodology of conveyancing info to and among a development team is face-to-face speech.

7) Working with computer software is the primary life of progress.

8) Agile processes promote property development. The sponsors, developers, and users will be able to maintain a relentless pace indefinitely.

9) Continuous attention to technical excellence and smart style enhances nimbleness.

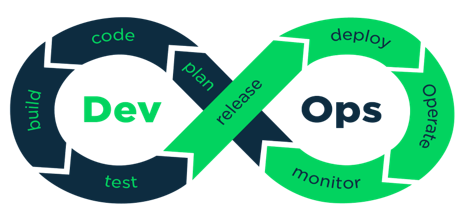
10) Simplicity—the art of maximizing the number of work not done—is essential.

11) The most effective architectures, necessities, and styles emerge from self–organizing groups.

12) At regular intervals, the team reflects on a way to become simpler, then tunes and adjusts its behavior consequently.

**2. Explain DevOps architecture and its features with a neat sketch.**

A) **DevOps**: DevOps is a set of practices that brings together the development and operations team to enable the continuous delivery oh high quality software



* **Plan**– In DevOps planning plays an important role. In this stage, all the requirements of the project and everything regarding the project like time for each stage, cost. etc are discussed. This will help everyone in teams to get a brief idea about the [DevOps project](https://www.edureka.co/blog/devops-projects/).
* **Code** – In this Stage the code is written over here according to the client’s requirements. Here the code is divided into small codes called Units. This is done to get a clear picture of the code. For example, if the team is doing a project on an online -Ekart application then the login part is divided as one unit, after login the page which shows all the categories is divided as another unit, user profile as another unit, etc.
* **Build** – In this stage Building of the units is done. Some of the examples of the tools used are maven, Gradle.
* **Test** – Testing of all units is done in this stage. So we will get to know where exactly the code is having bugs and if there are mistakes found it is returned. Some of the examples of the tools used are Selenium, PYtest
* **Release**– In this stage, all the units of the codes are integrated. That means in this step we will be creating a connection between the development team and the operation team to implement Continuous Integration and Continuous Deployment. An example of the tool used is Jenkins.
* **Deploy** – In this stage, the code is deployed on the client’s environment. Some of the examples of the tools used are AWS, Docker.
* **Operate** – Operations are performed on the code if required. Some of the examples of the tools used are Kubernetes, open shift.
* **Monitor** – In this stage monitoring of the application is done over here in the client’s environment. Some of the examples of the tools used are Nagios, elastic stack.

**Features of Devops:**

 **Continuous Integration (CI)**: Automates the process of integrating code changes into a shared repository frequently (several times a day). This allows for early detection of errors and avoids integration problems later in the process.

 **Continuous Delivery (CD)**: Automates the deployment of code to various environments (e.g., staging, production), making it faster and more reliable to release updates to production.

 **Infrastructure as Code (IaC)**: Automates the provisioning and management of infrastructure using code (e.g., using tools like Terraform or CloudFormation), allowing for consistent and scalable.

**3. Describe various features and capabilities in agile.**

A)**Various features of Agile:**

 **Continuous Integration (CI)**: Automates the process of integrating code changes into a shared repository frequently (several times a day). This allows for early detection of errors and avoids integration problems later in the process.

 **Continuous Delivery (CD)**: Automates the deployment of code to various environments (e.g., staging, production), making it faster and more reliable to release updates to production.

 **Infrastructure as Code (IaC)**: Automates the provisioning and management of infrastructure using code (e.g., using tools like Terraform or CloudFormation), allowing for consistent and scalable environments.

 **Automated Testing**: Every code change triggers automated tests, ensuring high-quality code and reducing the chances of defects making it to production.

**Agile Capabilities:**

**1. Agile Mindset:**

The Agile Mindset refers to the attitudes, values, and principles that underpin the Agile methodology. It is about fostering a culture of collaboration, flexibility, continuous learning, and delivering value. The Agile Mindset encourages teams to embrace change, respond to feedback, and keep the focus on customer needs.

### 2. ****Agile Methodologies****:

**Agile methodologies** are frameworks or practices that implement Agile principles in different ways. These methodologies provide structured approaches to implement Agile processes, tailored to specific team needs or projects.

### 3. ****Agile Practices****:

**Agile practices** are specific techniques and activities used to implement Agile methodologies. These practices help teams deliver high-quality software with a focus on collaboration, communication, and customer value.

### 4. ****Agile Tools****:

**Agile tools** are software applications designed to support and facilitate the implementation of Agile practices. These tools help with project management, collaboration, tracking progress, and automating tasks.

### 5. ****Continuous Delivery (CD)****:

**Continuous Delivery (CD)** is the practice of keeping your codebase deployable at any moment. It builds upon continuous integration (CI) by automating the release process to the point where software can be released to production at any time, without any manual intervention.

6. **Continuous Improvement in Agile Methodologies**:

**Continuous Improvement** in Agile refers to the iterative process of refining and optimizing both the product and the development process. It's built into the Agile framework and focuses on regular reflection and feedback to improve how teams work and deliver value.

# Set-2

**1) What is SDLC? Explain various phases involved in SDLC.**

Software Development Life Cycle (SDLC) is a structured process that helps developers create software. It's also known as the software development process.

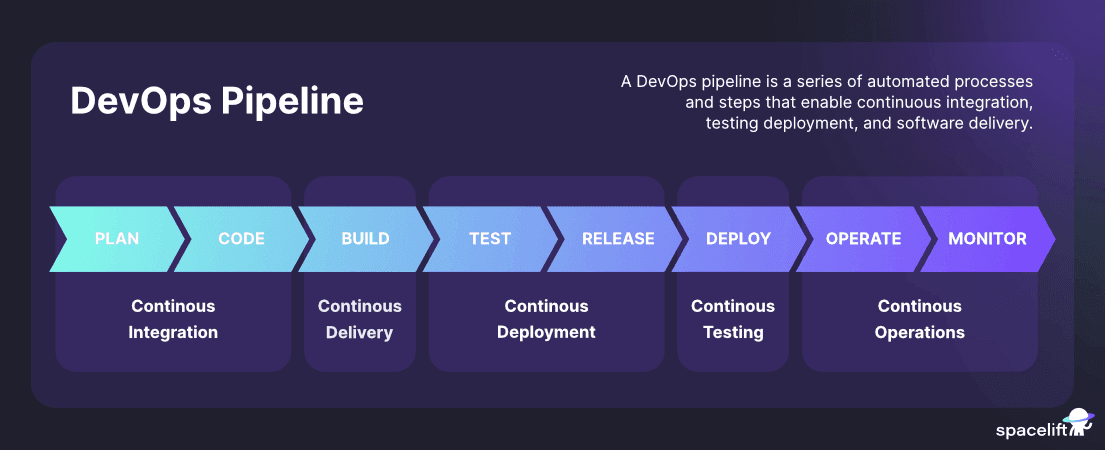


The common phases of the SDLC are:

1. **Planning**: This initial phase involves defining the scope and purpose of the project. Key activities include feasibility studies, resource allocation, and project scheduling. Stakeholders identify requirements and establish a project plan.
2. **Requirements Gathering and Analysis**: In this phase, detailed requirements are collected from stakeholders. This includes functional and non-functional requirements. Analysts document these requirements and create a Software Requirements Specification (SRS) document.
3. **Design**: The design phase involves creating the architecture of the software system. This includes high-level design (HLD) and low-level design (LLD). The design phase outlines how the software will be structured and how components will interact.
4. **Implementation (or Coding)**: During this phase, developers write the actual code based on the design specifications. This phase may involve unit testing to ensure that individual components function correctly.
5. **Testing**: After implementation, the software undergoes rigorous testing to identify and fix defects. Various testing methods (e.g., unit testing, integration testing, system testing, acceptance testing) are employed to ensure the software meets the specified requirements.
6. **Deployment**: Once testing is complete and the software is deemed ready, it is deployed to the production environment. This phase may involve user training and documentation to help users adapt to the new system.
7. **Maintenance**: After deployment, the software enters the maintenance phase, where it is monitored for issues, and updates or enhancements are made as needed. This phase ensures the software remains functional and relevant over time.
8. **Evaluation (optional)**: Some models include an evaluation phase where the project is reviewed to assess its success and gather lessons learned for future projects.

**2. Explain briefly about various stages involved in the DevOps pipeline.**

The DevOps pipeline is a set of automated processes that enable development and operations teams to collaborate effectively throughout the software development lifecycle. It integrates various stages to ensure continuous integration, continuous delivery (CI/CD), and continuous deployment of software.



**Plan**

Planning is the first step of every application development process. It includes identifying the project requirements, finding the resources needed, setting goals, and defining the end-to-end scope of the project. Project management tools such as Jira or Asana are examples of tools used at this stage.

**Code**

Usually, the most critical part of the whole software development and DevOps pipeline process is writing the code for the application. This includes developing, reviewing, and storing the source code in a version control system such as GitHub, BitBucket, or GitLab.

**Build**

After the new code has been stored and integrated with the rest of the codebase, it’s time to build all the necessary artifacts and compile the source code to deployable components. To achieve this outcome, build automation and CI/CD tools are used, such as Jenkins, GitHub Actions, GitLab CI/CD, CircleCI, and more. (Check out [CircleCi vs. Jenkins](https://spacelift.io/blog/circleci-vs-jenkins)).

**Test**

Testing is an integral stage of the DevOps pipeline to ensure software quality. Here, teams set up various types of automated and manual tests to validate reliability, functionality, and quality. The tools vary depending on the code and infrastructure used, but a few examples are Selenium, k6, and TestRail.

**Release**

After storing, packaging, and testing our code, the next step is to deploy a new software release to staging and production environments. As discussed, this step could be entirely automated or require human approval. Typically, the release process is enabled by CI/CD tools that deploy to cloud or on-premises environments, container and orchestration systems such as Docker or Kubernetes, and progressive delivery tools such as [ArgoCD](https://spacelift.io/blog/argocd).

**Monitor**

Our job isn’t done when deploying the new code, as we have to monitor and maintain the applications and infrastructure in production continuously. Leveraging our end-to-end monitoring solution, teams gather feedback, analyze metrics, and use them to improve the apps and environments. A few examples of tools that facilitate this stage are Datadog, Prometheus, Splunk, and ELK stack, among others.

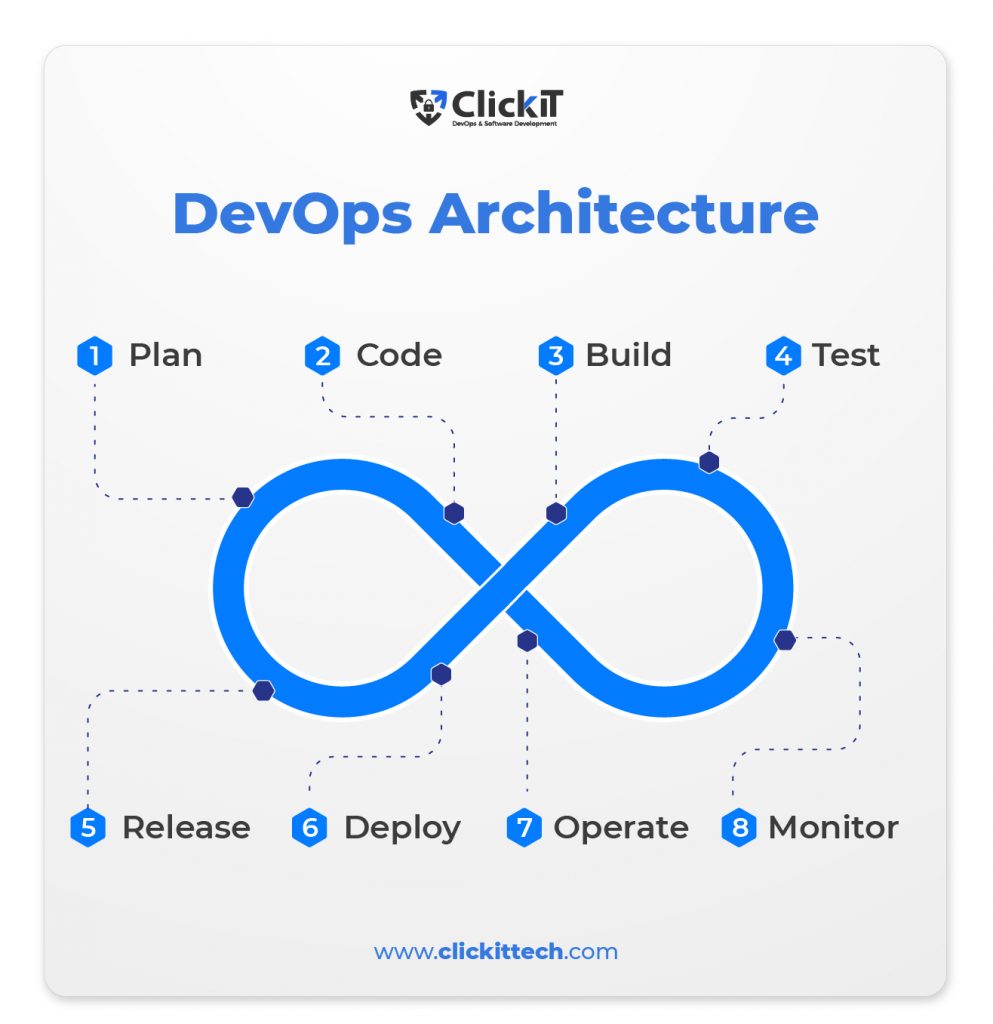
**Operate**

The operate stage ensures our applications and environments are available and up and running at all times with the least possible downtime. DevOps teams leverage [Infrastructure as Code tools](https://spacelift.io/blog/infrastructure-as-code-tools), such as Terraform or Pulumi, configuration management tools, such as Ansible or Puppet, and collaborative infrastructure tools, such as [Spacelift](https://spacelift.io/)

3)**Explain DevOps architecture and its features with a neat sketch.**

**DevOps Architecture:**

DevOps is a software development approach that combines development (Dev) and operations (Ops) to deliver continuous value to customers. It aims to bridge the gap between development, testing, deployment, and operations, making the entire software development lifecycle (SDLC) more efficient, collaborative, and automated.



* **Plan**– In DevOps planning plays an important role. In this stage, all the requirements of the project and everything regarding the project like time for each stage, cost. etc are discussed. This will help everyone in teams to get a brief idea about the [DevOps project](https://www.edureka.co/blog/devops-projects/).
* **Code** – In this Stage the code is written over here according to the client’s requirements. Here the code is divided into small codes called Units. This is done to get a clear picture of the code. For example, if the team is doing a project on an online -Ekart application then the login part is divided as one unit, after login the page which shows all the categories is divided as another unit, user profile as another unit, etc.
* **Build** – In this stage Building of the units is done. Some of the examples of the tools used are maven, Gradle.
* **Test** – Testing of all units is done in this stage. So we will get to know where exactly the code is having bugs and if there are mistakes found it is returned. Some of the examples of the tools used are Selenium, PYtest
* **Release**– In this stage, all the units of the codes are integrated. That means in this step we will be creating a connection between the development team and the operation team to implement Continuous Integration and Continuous Deployment. An example of the tool used is Jenkins.
* **Deploy** – In this stage, the code is deployed on the client’s environment. Some of the examples of the tools used are AWS, Docker.
* **Operate** – Operations are performed on the code if required. Some of the examples of the tools used are Kubernetes, open shift.
* **Monitor** – In this stage monitoring of the application is done over here in the client’s environment. Some of the examples of the tools used are Nagios, elastic stack.

# Set-3

**1)Write the difference between Waterfall and Agile models**.

| **Agile Model** | **Waterfall Model** |
| --- | --- |
| Client input is required throughout the product development. | Client input is required only after completing each phase. |
| Changes can be made at any stage. | Changes cannot be made after the completion of a phase. |
| Coordination among [project teams](https://www.geeksforgeeks.org/software-project-team-organization/) is required to ensure correctness. | Coordination is not needed as one team starts the work after the finish of another team. |
| It is really useful in large and complex projects. | It is mainly used for small [project development](https://www.geeksforgeeks.org/phases-project-management-processes/). |
| The testing part can be started before the development of the entire product. | Testing can only be performed when the complete product is ready. |
| A Small team is sufficient for Agile project management. | It requires a large team. |
| The cost of development is less. | The cost of development is high. |
| It completes the project in comparatively less time. | It takes more time compared to Agile. |
| The Agile Method is known for its flexibility. | The waterfall Method is a structured software development methodology so it is quite rigid. |
| After each sprint/cycle test plan is discussed. | Hardly any test plan is discussed during a cycle. |

**2.Discuss in detail about DevOps eco system.**

The DevOps ecosystem is a software engineering practice that combines development and operations teams to collaborate on the software development process. The goal is to deliver products and services faster and more reliably.

Key aspects of the DevOps ecosystem:

* **Continuous integration (CI):** Automates the process of integrating code changes from multiple developers into a single codebase
* **Continuous delivery (CD**): Automatically builds, tests, and deploys code changes to a production environment
* **Continuous monitoring**: Regularly monitors the application's performance and features to detect issues
* **Automation**: Uses automated tools to build, test, and deploy applications
* **Collaboration**: Encourages collaboration between development and operations teams

**3)List and explain the steps followed for adopting DevOps in IT projects.**

Adopting DevOps in IT projects involves several strategic steps to foster a culture of collaboration, automation, and continuous improvement between development and operations teams. The goal is to improve the speed, quality, and reliability of software delivery. Below are the key steps followed when adopting DevOps:

**1. Assess Current Processes and Set Objectives**

* **Evaluate Existing Practices**: Before introducing DevOps, it's important to understand the existing software development and deployment processes. This includes identifying bottlenecks, inefficiencies, and areas for improvement in both development and operations.

**2. Foster a Collaborative Culture**

* Break Down Silos: DevOps encourages collaboration between development, operations, QA, and other teams. Moving away from the traditional siloed structure is essential for success.

**3. Select and Implement Automation Tools**

* **Automate Repetitive Tasks**: DevOps relies heavily on automation. Start by automating repetitive tasks such as code integration, testing, deployment, and monitoring. Tools like Jenkins, GitLab CI/CD, or CircleCI for Continuous Integration (CI) and Continuous Deployment (CD) are commonly used.

**4. Implement Continuous Integration (CI) and Continuous Delivery (CD)**

* **Continuous Integration**: Developers should frequently commit code to a shared repository. CI tools automatically build and test the code after each commit to ensure that it integrates well with the rest of the codebase.
* **Continuous Delivery**: Once the code passes automated tests, it should be automatically deployed to production or staging environments. This ensures that the code is always in a deployable state and speeds up the release cycle.

5**. Adopt a Microservices Architecture (If Applicable)**

* **Modularize the Application:** A microservices architecture divides an application into smaller, independently deployable services. This improves the scalability, maintainability, and agility of software projects. It also complements DevOps practices by enabling teams to work on different services in parallel without dependencies.

**6. Automate Testing and Monitoring**

* **Automate Testing**: Implement automated testing (unit, integration, performance, etc.) to ensure that code changes do not introduce defects. Tools like Selenium, JUnit, and TestNG can be used for automation.

**7. Establish Continuous Feedback Mechanisms**

* **Feedback Loops**: DevOps emphasizes the importance of feedback. Set up systems for continuous feedback from the testing and production environments. This can include user feedback, performance monitoring, and testing results.

**8. Scale and Improve the DevOps Practices**

* **Iterative Scaling**: Start small, perhaps with a single team or project, and then scale the DevOps practices to the organization over time. This allows for adjustments and learning before full-scale adoption.

## **Set 4**

**1 Explain the values and principles of Agile model.**

1. Agile is an iterative software development approach that emphasizes collaboration and rapid delivery, while DevOps is a software delivery approach that breaks down silos between development and operations. Both are complementary practices that can improve efficiency and predictability in software development.

**Values of Agile Methodology:**

1)Individuals and Interactions over Processes and Tools.

2)Working Software over Comprehensive Documentation.

3)Customer Collaboration over Contract Negotiation.

4)Responding to Change over Following a Plan.

**Principles of the Agile Methodologies:**

The Agile Alliance defines twelve lightness principles for those who need to attain agility:

1) Our highest priority is to satisfy the client through early and continuous delivery of valuable computer software.

2) Welcome dynamic necessities, even late in development. Agile processes harness modification for the customer’s competitive advantage.

3) Deliver operating computer software often, from a pair of weeks to a couple of months, with a preference to the shorter timescale.

4) Business individuals and developers should work along daily throughout the project.

5) The build comes around actuated people. offer them the setting and support they have, and trust them to urge the task done.

6) The foremost economical and effective methodology of conveyancing info to and among a development team is face-to-face speech.

7) Working with computer software is the primary life of progress.

8) Agile processes promote property development. The sponsors, developers, and users will be able to maintain a relentless pace indefinitely.

9) Continuous attention to technical excellence and smart style enhances nimbleness.

10) Simplicity—the art of maximizing the number of work not done—is essential.

11) The most effective architectures, necessities, and styles emerge from self–organizing groups.

12) At regular intervals, the team reflects on a way to become simpler, then tunes and adjusts its behavior consequently.

**2. Write a short notes on the DevOps Orchestration.**

DevOps Orchestration refers to the integration and automation of various DevOps tools and processes to streamline and synchronize the workflows involved in the software development and delivery pipeline. It focuses on automating and coordinating complex tasks across development, testing, deployment, and operations, ensuring smooth communication and efficient execution. Orchestration tools manage and monitor multiple processes, allowing teams to focus on continuous integration, delivery, and infrastructure management without manual intervention.

Key Aspects of DevOps Orchestration:

1. **Automation of Tasks:**
   * Orchestration automates repetitive and complex tasks such as code integration, testing, deployment, and monitoring. This reduces manual errors and accelerates the delivery pipeline.
2. **Tool Integration:**
   * It connects and integrates various DevOps tools like version control systems (Git), continuous integration (CI) tools (Jenkins, GitLab CI), configuration management tools (Ansible, Puppet), and deployment tools (Docker, Kubernetes). This ensures a seamless flow of data and operations across different stages of the software lifecycle.
3. **Continuous Delivery Pipeline:**
   * Orchestration facilitates the end-to-end automation of the continuous integration and continuous delivery (CI/CD) pipeline. This enables faster, more reliable delivery of software updates by ensuring that each step, from development to production, is automatically triggered and properly managed.
4. **Infrastructure as Code (IaC):**
   * Orchestration supports Infrastructure as Code (IaC) principles, allowing infrastructure and configurations to be defined in code and automatically deployed, reducing manual configuration and ensuring consistency across environments.
5. **Scalability and Flexibility**:
   * DevOps orchestration tools provide flexibility in scaling processes, allowing teams to handle growing workloads or project complexity with minimal adjustments. Automated scaling and load balancing ensure that applications can handle increasing traffic without issues.
6. **Monitoring and Feedback:**
   * Orchestration includes continuous monitoring and feedback loops, ensuring that all processes are functioning as expected. If an error or failure occurs, it triggers corrective actions, such as alerting teams or rolling back changes.

**Popular DevOps Orchestration Tools:**

* Jenkins: Widely used for automating CI/CD pipelines and integrating other tools.
* Ansible: Automates software provisioning, configuration management, and application deployment.
* Kubernetes: Manages containerized applications and automates deployment, scaling, and operations.
* Terraform: A tool for infrastructure provisioning that supports IaC and orchestrates infrastructure management.
* Docker Swarm: A container orchestration tool that helps manage and deploy Docker containers.

**3. What is the difference between Agile and DevOps models?**

