**ASSIGNMENT NO 1**

**Aim : To understand devops principles, practices and devops engineer roles and responsibilities.**

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.

**Lifecycle:**



DevOps is the next evolution of agile methodologies. A cultural shift that brings development and operations teams together. DevOps is a practice that involves a cultural change, new management principles, and technology tools that help to implement best practices.DevOps is the next evolution of agile methodologies. A cultural shift that brings development and operations teams together. DevOps is a practice that involves a cultural change, new management principles, and technology tools that help to implement best practices.

Regardless of the type of DevOps toolchain an organization uses, a DevOps process needs to use the right tools to address the key phases of the DevOps lifecycle:

* Discover
* Plan
* Build
* Test
* Monitor
* Operate
* Continuous feedback

**Discover:**

In the Discover phase, a DevOps team researches and defines the scope of a project. In particular, it involves activities such as user research, establishing goals, and defining success.

Tools like Mural and Miro empower the entire software team to gather ideas and conduct research. Jira Product Discovery organizes this information into actionable inputs and prioritizes actions for development teams. As you’re prioritizing, you’ll also need to keep your backlog of user feedback in mind.

**Plan:**

Taking a page out of the agile handbook, we recommend tools that allow development and operations teams to break work down into smaller, manageable chunks for quicker deployments. This allows you to learn from users sooner and helps with optimizing a product based on the feedback. Look for tools that provide sprint planning, issue tracking, and allow collaboration, such as Jira.

**Build**:

While Puppet and Chef primarily benefit operations, developers use open source tools like Kubernetes and Docker to provision individual development environments. Coding against virtual, disposable replicas of production helps you get more work done.

**Continuous Delivery**:

Continuous integration is the practice of checking in code to a shared repository several times a day, and testing it each time. That way, you automatically detect problems early, fix them when they’re easiest to fix, and roll out new features to your users as early as possible.Look for tools that automatically apply your tests to development branches, and give you the option to push to main when branch builds are successful. Along with that, you get continuous feedback through real-time chat alerts from your team with a simple integration.

**Test**:

Testing tools span many needs and capabilities, including exploratory testing, test management, and orchestration. However, for the DevOps toolchain, automation is an essential function. [Automated testing](https://www.atlassian.com/devops/devops-tools/test-automation) pays off over time by speeding up your development and testing cycles in the long run. And in a DevOps environment, it’s important for another reason: awareness. Test automation can increase software quality and reduce risk by doing it early and often. Development teams can execute automated tests repeatedly, covering several areas such as UI testing, security scanning, or load testing. They also yield reports and trend graphs that help identify risky areas.

**Deploy**:

One of the most stressful parts of shipping software is getting all the change, test, and deployment information for an upcoming release into one place. The last thing anyone needs before a release is a long meeting to report on status.

**Automated deployment**:

There’s no magic recipe for automated deployment that will work for every application and IT environment. But converting operations’ runbook into a cmd-executable script using Ruby or bash is a common way to start.

**Operate**:

The keys to unlocking collaboration between DevOps teams is making sure they’re viewing the same work. What happens when incidents are reported? Are they linked and traceable to software problems? When changes are made, are they linked to releases?

Nothing blocks Dev’s collaboration with Ops more than having incidents and software development projects tracked in different systems. Look for tools that keep [incidents](https://www.atlassian.com/incident-management), [changes](https://www.atlassian.com/itsm/change-management), [problems](https://www.atlassian.com/itsm/problem-management), and software projects on one [platform](https://www.atlassian.com/software/jira/service-management) so you can identify and fix problems faster.

**Observe**:

There are two types of monitoring that should be automated: server monitoring and application performance monitoring.

Manually “topping” a box or hitting your API with a test is fine for spot-checking. But to understand trends and the overall health of your application (and environments), you need software that is listening and recording data 24/7. Ongoing observability is a key capability for successful DevOps teams.

**Continuous Feedback**:

Customers are already telling you whether you’ve built the right thing – you just have to listen. Continuous feedback includes both the culture and processes to collect feedback regularly, and tools to drive insights from the feedback. Continuous feedback practices include collecting and reviewing NPS data, churn surveys, bug reports, support tickets, and even tweets. In a [DevOps culture](https://www.atlassian.com/team-playbook/examples/devops-culture?utm_campaign=service-desk_devops16-blog), everyone on the product team has access to user comments because they help guide everything from release planning to exploratory testing sessions.

**DEVOPS TOOL**



1) Puppet

Puppet is the most widely used DevOps tool. It allows the delivery and release of the technology changes quickly and frequently. It has features of versioning, automated testing, and continuous delivery. It enables to manage entire infrastructure as code without expanding the size of the team.

2) Ansible

Ansible is a leading DevOps tool. Ansible is an open-source IT engine that automates application deployment, cloud provisioning, intra service orchestration, and other IT tools. It makes it easier for DevOps teams to scale automation and speed up productivity. Ansible is easy to deploy because it does not use any agents or custom security infrastructure on the client-side, and by pushing modules to the clients. These modules are executed locally on the client-side, and the output is pushed back to the Ansible server.

3) Docker

Docker is a high-end DevOps tool that allows building, ship, and run distributed applications on multiple systems. It also helps to assemble the apps quickly from the components, and it is typically suitable for container management.

4) Nagios

Nagios is one of the more useful tools for DevOps. It can determine the errors and rectify them with the help of network, infrastructure, server, and log monitoring systems.

5) CHEF

A chef is a useful tool for achieving scale, speed, and consistency. The chef is a cloud-based system and open source technology. This technology uses Ruby encoding to develop essential building blocks such as recipes and cookbooks. The chef is used in infrastructure automation and helps in reducing manual and repetitive tasks for infrastructure management.

6) Jenkins

Jenkins is a DevOps tool for monitoring the execution of repeated tasks. Jenkins is a software that allows continuous integration. Jenkins will be installed on a server where the central build will take place. It helps to integrate project changes more efficiently by finding the issues quickly.

7) Git

Git is an open-source distributed version control system that is freely available for everyone. It is designed to handle minor to major projects with speed and efficiency. It is developed to co-ordinate the work among programmers. The version control allows you to track and work together with your team members at the same workspace. It is used as a critical distributed version-control for the DevOps tool.

8) SALTSTACK

Stackify is a lightweight DevOps tool. It shows real-time error queries, logs, and more directly into the workstation. SALTSTACK is an ideal solution for intelligent orchestration for the software-defined data center.

9) Splunk

Splunk is a tool to make machine data usable, accessible, and valuable to everyone. It delivers operational intelligence to DevOps teams. It helps companies to be more secure, productive, and competitive.

10) Selenium

Selenium is a portable software testing framework for web applications. It provides an easy interface for developing automated tests.

**ROLES AND RESPONSIBILITIES**