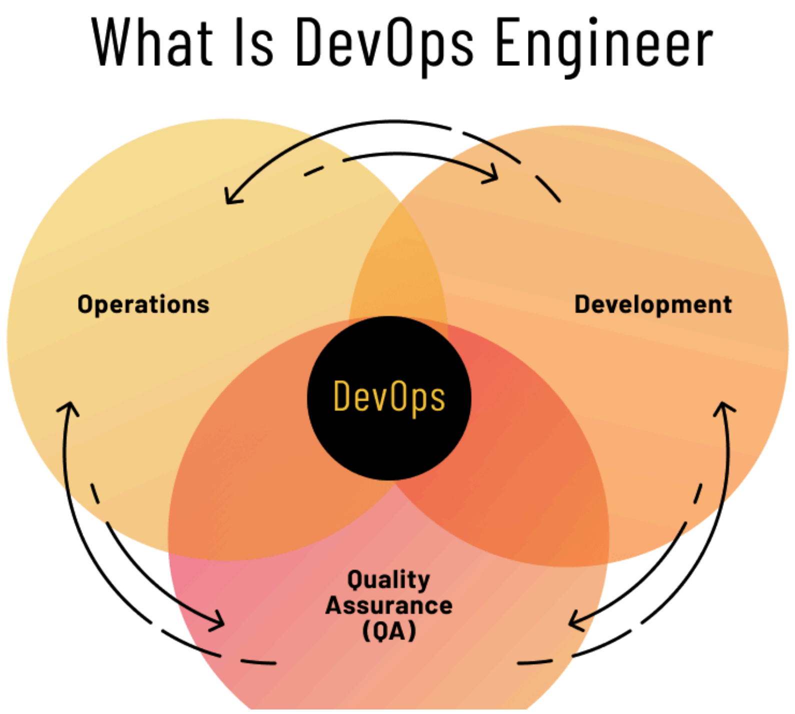
EXPERIMENT NO.1

AIM:

To understand devops ,principles ,practices and DevOps engineer’s role and responsibilities

THEORY:

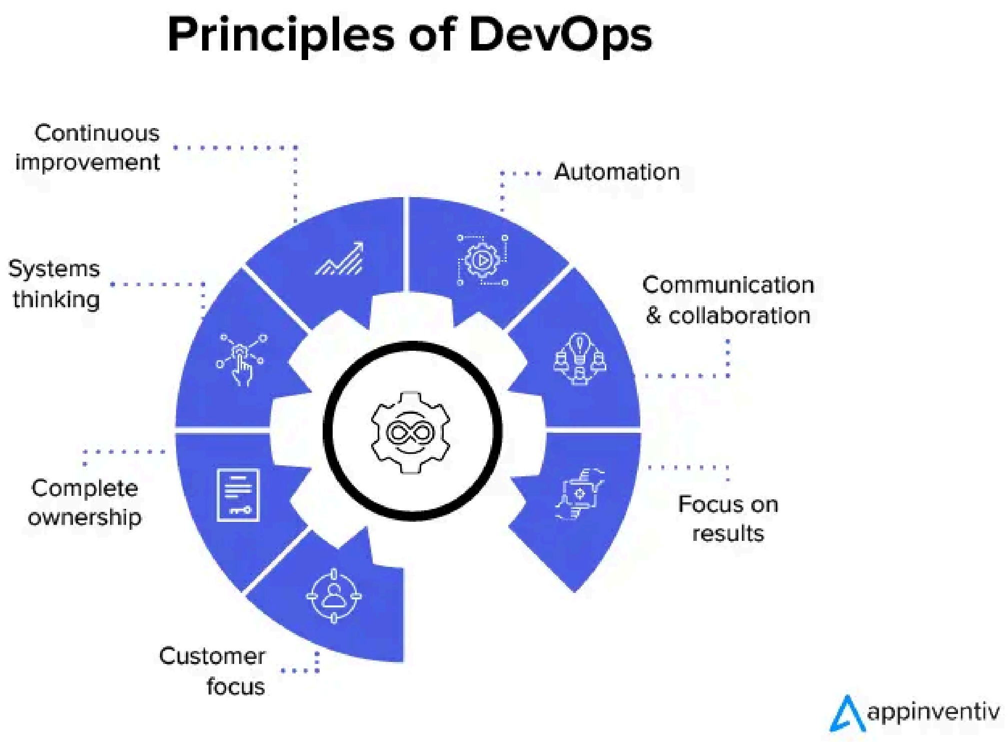
DevOps Definition:
DevOps is a collaborative and integrated set of practices, principles, and cultural philosophies that aim to streamline and enhance the entire software development lifecycle (SDLC). It brings together development (Dev) and operations (Ops) teams, fostering a culture of collaboration, communication, and continuous improvement. DevOps aims to automate processes, eliminate silos between different functional areas, and deliver high-quality software more efficiently.



Key Principles of DevOps:

- **Collaboration** : Collaborative efforts ensure shared responsibilities, improved communication, and a unified focus on delivering value to end-users.
- **Automation** : Automation is a fundamental DevOps principle that involves the use of tools and scripts to automate repetitive and manual tasks across the software development lifecycle.
- **Continuous Integration (CI)** : CI is a practice where developers regularly integrate their code into a shared repository.

- **Continuous Delivery/Deployment (CD)** : Continuous Delivery (CD) involves ensuring that the application is always in a deployable state, with the capability to release changes at any time.
- **Infrastructure as Code (IaC) :** IaC involves managing and provisioning infrastructure using code and automation. Infrastructure components, such as servers and networks, are defined and configured through code, ensuring consistency and enabling rapid and reproducible deployments.
- **Monitoring and Feedback:** Continuous monitoring of applications and infrastructure provides real-time feedback on performance, user experience, and potential issues.



Practices :

1. **Agile project management**
Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Agile teams focus on delivering work in smaller increments, instead of waiting for a single massive release date. Requirements, plans, and results are evaluated continuously, allowing teams to respond to feedback and pivot as necessary.

The following are key concepts for agile project management:

Start with a workflow that includes four phases: to do, in progress, code review, and done. Read more about workflows.
Teams need to break large-scale projects into smaller tasks and respond to changes in needs or scope as they make progress. Read more about how to use epics, stories, and themes to scope and structure work.
How do you plan, track, and measure the incremental work? Scrum and kanban are core frameworks for teams practicing agile methodology.

2. **Shift left with CI/CD**
When teams "shift left", they bring testing into their code development processes early. Instead of sending multiple changes to a separate test or QA team, a variety of tests are performed throughout the coding process so that developers can fix bugs or improve code quality while they work on the relevant section of the codebase. The practice of continuous integration and continuous delivery (CI/CD), and deployment underpins the ability to shift left. Read more about CI/CD.

3. **Build with the right tools**
A DevOps toolchain requires the right tools for each phase of the DevOps lifecycle, with key capabilities to improve software quality and speed of delivery. Read more about how to choose DevOps tools and review functionality for each phase of the DevOps lifecycle.

4. **Implement automation**
Continuous integration and delivery allows developers to merge code regularly into the main repository. Instead of manually checking code, CI/CD automates this process, from batching in a specified window to frequent commits. In addition to CI/CD, automated testing is essential to successful DevOps practices. Automated tests might include end-to-end testing, unit tests, integration tests, and performance tests. Read more about incorporating automation into your software development processes. Read more about automation.

5. **Monitor the DevOps pipeline and applications**
It's important to monitor the DevOps pipeline so a broken build or failed test doesn't cause unnecessary delays. Automation improves the speed of development tremendously, but if there is a failure in an automated process and nobody knows about it, you're better off doing the work manually. In a similar vein, it's important to monitor production applications in order to identify failures or performance deficiencies, before you hear about them from your customers.

6. **Observability**
As the industry moved away from monolithic, on-premise systems and applications to cloud-native, microservice-based applications, monitoring is now considerably more complex. As a result, there is an increasing interest in observability. It is often said that the three pillars of observability are logs, traces, and metrics. Logs are generated by most systems components and applications and consist of time-series data about the functioning of the system or application. Traces track the flow of logic within the

application. Metrics include CPU/RAM reservation or usage, disk space, network connectivity, and more. Observability simply means using all three of these sources of information in aggregate to make discoveries and predictions about the functioning of a complex system, which would otherwise be difficult to achieve. Read more about observability.

7. **Gather continuous feedback**
Continuous feedback ensures team members have all the information needed to do their jobs on a timely basis. From the development perspective this entails that the team is alerted to any pipeline failures immediately. It also means that clear, thorough code test results are made available to the developers as quickly as possible. From the product management perspective the team is made aware of any production failures or performance deficiencies, or reported bugs. In the past there was widespread belief that a development team could only optimize for speed or quality. Continuous feedback is one of the elements of DevOps that makes it possible to have both.

8. **Change the culture**
DevOps requires collaboration, transparency, trust, and empathy. If your organization is one of the rare ones where these qualities are already established, it should be fairly easy for your teams to adopt DevOps practices. If not, some effort will be required to develop these qualities. The most common organizational structures are siloed, meaning different teams have separate areas of ownership and responsibility and there is minimal cross-team communication or collaboration. For DevOps to succeed, these barriers must be eliminated by adopting the "you build it, you run it" practice. This doesn’t mean there aren’t people or teams who specialize, only that the lines of communication and collaboration between teams are open and used. Read more on building your team culture.

Honing your DevOps practices is an ongoing journey. Focus on the people and processes as you start your DevOps transformation, and incorporate advanced tooling, integration, and feature functionality as you become a mature team.

If you’re just starting your journey to DevOps, learn best practices with our Beginner's guide to DevOps. To put DevOps into practice, we recommend trying Open DevOps, which provides everything teams need to develop and operate software. Teams can build the DevOps toolchain they want, thanks to integrations with leading vendors and marketplace apps. Try it now.

● ***Role of a DevOps Engineer:***

A DevOps engineer plays a crucial role in bridging the gap between software development and IT operations. Their responsibilities revolve around implementing and maintaining the tools, practices, and cultural aspects necessary to achieve continuous integration, continuous delivery, and overall automation in the software development lifecycle. The role is multidisciplinary, requiring expertise in both development and operations, as well as a deep understanding of the DevOps philosophy.
Key Responsibilities:
Automation and Scripting:

Description: DevOps engineers are responsible for automating manual, repetitive tasks throughout the software development lifecycle. This includes building and deploying code, configuring infrastructure, and managing environments.

Infrastructure as Code (IaC):

Description: DevOps engineers leverage IaC principles to define and manage infrastructure using code. They use tools like Terraform, Ansible, or Puppet to provision and configure servers and other infrastructure components.

Continuous Integration/Continuous Deployment (CI/CD):

Description: DevOps engineers implement and maintain CI/CD pipelines, ensuring the automated building, testing, and deployment of code changes. They configure tools like Jenkins, GitLab CI, or Travis CI to orchestrate these pipelines.

Monitoring and Logging:

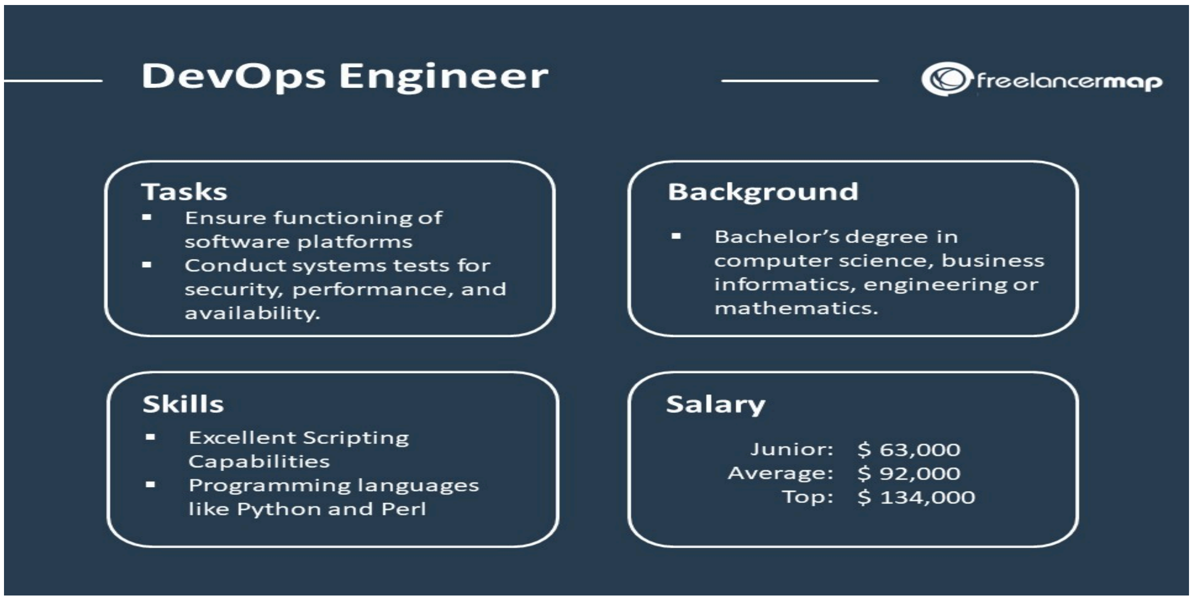
Description: DevOps engineers set up monitoring and logging systems to track the performance and health of applications and infrastructure. They use tools like Prometheus, ELK Stack (Elasticsearch, Logstash, Kibana), or Grafana.

Collaboration and Communication:

Description: DevOps engineers facilitate communication and collaboration between development and operations teams. They act as a bridge, ensuring that both sides understand each other's requirements and constraints.

Overall Contribution to DevOps Culture:

DevOps engineers contribute to the DevOps culture by instilling a mindset of collaboration, automation, and continuous improvement. They facilitate the adoption of practices that break down silos, promote transparency, and streamline the software development lifecycle. Through their technical expertise and focus on automation, DevOps engineers play a pivotal role in realising the benefits of a more agile, efficient, and reliable software delivery process.



DevOps Engineer

freelancermap

Tasks
- Ensure functioning of software platforms
- Conduct systems tests for security, performance, and availability.

Background
- Bachelor's degree in computer science, business informatics, engineering or mathematics.

Skills
- Excellent Scripting Capabilities
- Programming languages like Python and Perl

Salary

Junior:	$ 63,000
Average:	$ 92,000
Top:	$ 134,000

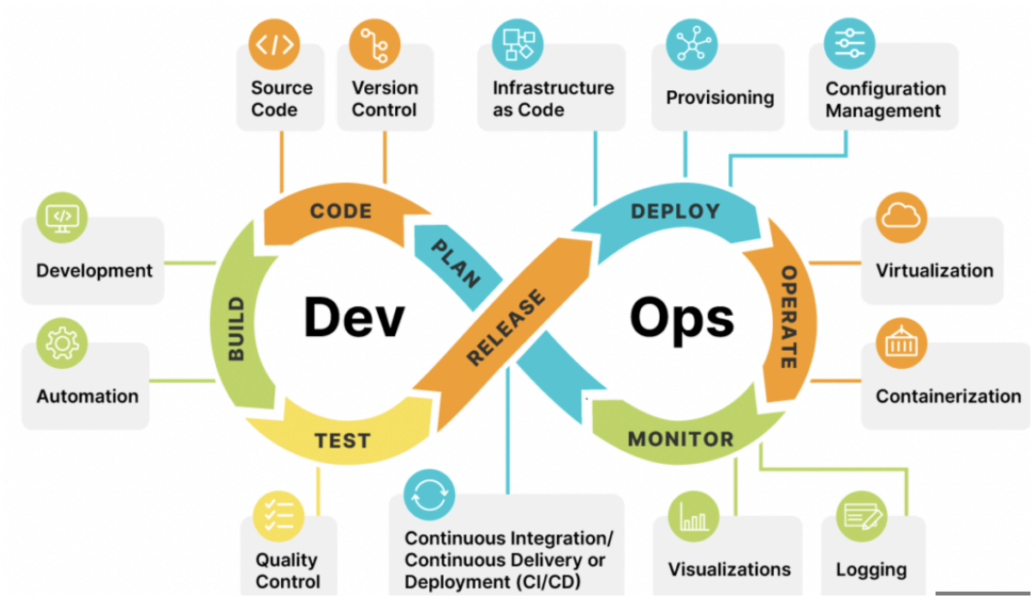
RESPONSIBILITIES :

Some of the core responsibilities of DevOps Engineer include –

- Understanding customer requirements and project KPIs
- Implementing various development, testing, automation tools, and IT infrastructure
- Planning the team structure, activities, and involvement in project management activities.
- Managing stakeholders and external interfaces
- Setting up tools and required infrastructure
- Defining and setting development, test, release, update, and support processes for DevOps operation
- Have the technical skill to review, verify, and validate the software code developed in the project.
- Troubleshooting techniques and fixing the code bugs
- Monitoring the processes during the entire lifecycle for its adherence and updating or creating new processes for improvement and minimizing the wastage
- Encouraging and building automated processes wherever possible
- Identifying and deploying cybersecurity measures by continuously performing vulnerability assessment and risk management
- Incidence management and root cause analysis
- Coordination and communication within the team and with customers
- Selecting and deploying appropriate CI/CD tools
- Strive for continuous improvement and build continuous integration, continuous development, and constant deployment pipeline (CI/CD Pipeline)
- Mentoring and guiding the team members

Monitoring and measuring customer experience and KPIs

- Managing periodic reporting on the progress to the management and the customer



CONCLUSION:

DevOps is a transformative approach emphasising collaboration, automation, and continuous improvement in software development and IT operations. Key principles include collaboration, automation, and practices like Continuous Integration and Infrastructure as Code. DevOps engineers play a pivotal role in implementing and maintaining these practices, fostering a culture of efficiency and rapid response to change. Overall, DevOps represents a cultural shift, promoting collaboration and automation to deliver high-quality software efficiently.