120A3050 – Shivani Pandeti

120A3051 - Shreya Idate

Batch: E3

Experiment No: 1

AIM: a) Introduction to DevOps

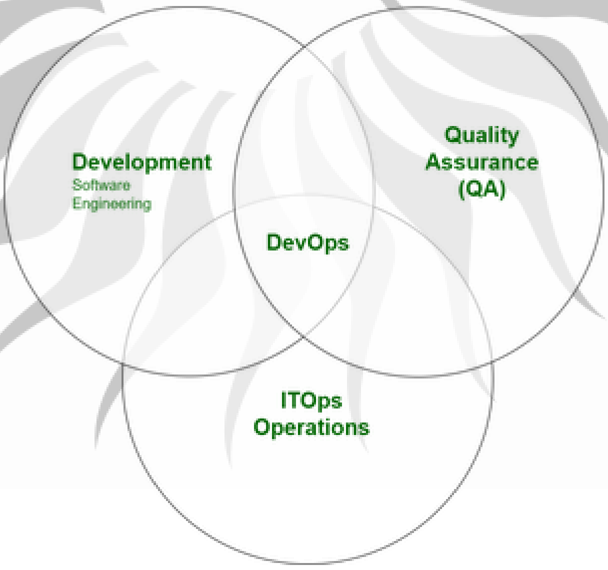
b) Case study on DevOps use cases & Real- life applications

THEORY: a) Introduction to DevOps

- What is DevOps?

DevOps stands for development and operations. It's a practice that aims at merging development, quality assurance, and operations (deployment and integration) into a single, continuous set of processes. This methodology is a natural extension of Agile and continuous delivery approaches. By adopting DevOps companies gain three core advantages that cover technical, business, and cultural aspects of development:

1. Higher speed and quality of product releases
2. Faster responsiveness to customer needs
3. Better working environment



- Need of DevOps

1. **Shorter Development Cycles, Faster Innovation:** When development and operations teams are in separate silos, it's usually difficult to tell if an application is ready for operations. When development teams simply turn over an application, the operations' cycle times are extended needlessly. With a combined development and operations team, applications are ready for use much more quickly. This

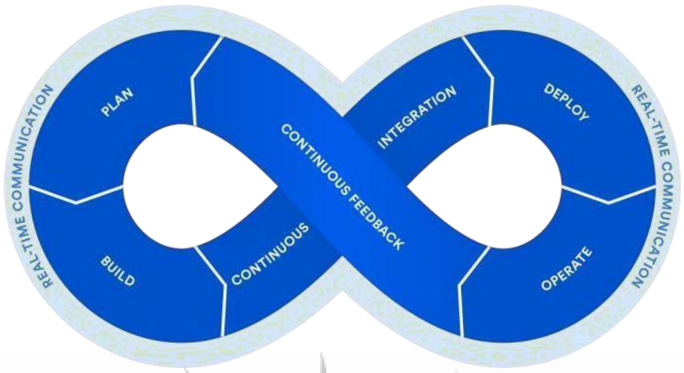
is important, since companies succeed based on their ability to innovate faster than their competitors do.

2. **Reduced Deployment Failures, Rollbacks, and Time to Recover:** Part of the reason teams experience deployment failures is due to programming defects. The shorter development cycles with DevOps promote more frequent code releases. This, in turn, makes it easier to spot code defects. Therefore, teams can reduce the number of deployment failures using agile programming principles that call for collaboration and modular programming. Rollbacks are similarly easier to manage because, when necessary, only some modules are affected.
   Time to recover is an important issue, because some failure has to be expected. But recovery is much faster when the development and operations teams have been working together, exchanging ideas and accounting for both teams' challenges during development.
3. **Improved Communication and Collaboration:** DevOps improves the software development culture. Combined teams are happier and more productive. The culture becomes focused on performance rather than individual goals. When the teams trust each other, they can experiment and innovate more effectively. The teams can focus on getting the product to market or into production, and their KPIs should be structured accordingly.
4. **Increased Efficiencies:** Increased efficiency helps to speed the development process and make it less prone to error. There are ways to automate DevOps tasks. Continuous integration servers automate the process of testing code, reducing the amount of manual work required. This means that software engineers can focus on completing tasks that can't be automated.
5. **Reduced Costs and IT Headcount:** All of the DevOps benefits translate to reduced overall costs and IT headcount requirements. According to Kevin Murphy from Red Hat, DevOps development teams require 35 percent less IT staff and 30 percent lower IT costs.

- DevOps Principles
    1. Culture
    2. Constant collaboration and communication
    3. Gradual Changes.
    4. Shared end-to-end responsibility
    5. Early problem-solving.
    6. Automation of processes
    7. Sharing.

    - DevOps Practices

DevOps requires a delivery cycle that comprises planning, development, testing, deployment, release, and monitoring with active cooperation between different members of a team.



To break down the process even more, let‘s have a look at the core practices that constitute the DevOps:

1. Agile planning**:** In contrast to traditional approaches of project management, agile planning organizes work in short iterations to increase the number of releases. This means that the team has only high-level objectives outlined, while making detailed planning for two iterations in advance. This allows for flexibility and pivots once the ideas are tested on an early product increment.

2. Continuous development**:** The concept of continuous ―everything‖ embraces continuous or iterative software development, meaning that all the development work is divided into small portions for better and faster production. Engineers commit code in small chunks multiple times a day for it to be easily tested.

3. Continuous automated testing**:** A quality assurance team sets committed code testing using automation tools like Selenium, Ranorex, UFT, etc. If bugs and vulnerabilities are revealed, they are sent back to the engineering team. This stage also entails version control to detect integration problems in advance. A Version Control System (VCS) allows developers to record changes in the files and share them with other members of the team, regardless of their location.

4. Continuous integration and continuous delivery (CI/CD)**:** The code that passes automated tests is integrated in a single, shared repository on a server. Frequent code submissions prevent a so-called ―integration hell‖ when the differences between individual code branches and the mainline code become so drastic over time that integration takes more than actual coding. Continuous delivery, detailed in our dedicated article, is an approach that merges development, testing, and deployment operations into a streamlined process as it heavily relies on automation. This stage enables the automatic delivery of code updates into a production environment.

5. Continuous deployment**:** At this stage, the code is deployed to run in production on a public server. Code must be deployed in a way that doesn‘t affect already functioning features and can be available for a large number of users. Frequent deployment allows for a ―fail fast‖ approach, meaning that the new features are tested and verified early. There are various automated tools that help engineers deploy a product increment. The most popular are Chef, Puppet, Azure Resource Manager, and Google Cloud Deployment Manager.

6. Continuous monitoring**:** The final stage of the DevOps lifecycle is oriented to the assessment of the whole cycle. The goal of monitoring is detecting the problematic areas of a process and analyzing

the feedback from the team and users to report existing inaccuracies and improve the product‘s functioning.

- DevOps Engineer role and responsibilities

The main function of a DevOps engineer is to introduce the continuous delivery and continuous integration workflow, which requires the understanding of the mentioned tools and the knowledge of several programming languages. Depending on the organization, job descriptions differ. Smaller businesses look for engineers with broader skillsets and responsibilities. For example, the job description may require product building along with the developers. Larger companies may look for an engineer for a specific stage of the DevOps lifecycle that will work with a certain automation tool.

b) Case study on DevOps use cases & Real- life applications

- Name of the Organization implementing DevOps:

Nordstorm Inc. an American luxury department store chain headquartered in Seattle, Washington and founded by John W. Nordstrom and Carl F. Walling in 1901. Nordstrom's key business drivers are - Win over customers through enhanced service and shopping retail experience, retaining old customers and acquiring new customers and better the quality of shopping that they deliver to their customers through all their platforms.

- Challenges faced in implementing DevOps:

In 2011, Nordstrom faced a major outage on Nordstrom.com during a period of high traffic that was predicted which had shown the lack of performance at scale the application had. This event brought to light the importance of testing the performance of the code before release.

They followed a traditional methodology of software development with separate teams for development and operations. This forced the developers to spend excessive amounts of time on debugging and fixing environments for production rather than being able to spend time on writing code for new feature sets.

Some of the challenges most IT companies face when implementing DevOps are:

- **Choosing the Right Metrics is Hard:** Enterprises transitioning to DevOps practices need to use metrics to recognize progress, document success, and uncover areas that need improvement, Forrester notes. An effective DevOps effort needs metrics that drive smart automation decisions and yet organizations often struggle with DevOps metrics.

- **Limited Funds:** DevOps initiatives face other obstacles as well. Given the significant organizational and IT changes involved with previously siloed teams joining forces, changing job roles, and encountering other transitions   adjustments will take time.
- **Unrealistic Goals, Bad Metrics Can Wreck DevOps:** DevOps efforts can fail for many reasons, such as setting unrealistic expectations, tracking metrics that don't align with business goals, or implementing a half-baked DevOps effort that embraces agile methodologies while keeping IT ops and engineering/development teams in traditional silos.
- **Overcoming the dev versus ops mentality:** In many organisations, we see the old cliché of developers tossing code over an imaginary wall to a centralised operations team—where developers are trying to innovate and make changes as quickly as possible, and the operations team are trying to maintain high service levels.
  The objectives of these two groups often counter each other, causing friction points and resulting in handovers and increased costs, along with longer feedback loops.
  DevOps is all about integrating teams together and breaking down silos within IT organisations. This journey begins by setting out a vision on how this will work for your organisation.
- **Moving from legacy infrastructure & architecture to microservices:** Older infrastructure and applications with complex architecture stacks can be problematic, even if they have served the company for years.
  Maintaining the status-quo can often lead to stability problems, lack of support and high operational costs—all ultimately resulting in being left behind the competition.
  Using infrastructure-as-code together with a microservices architecture is a huge step towards a future of continuous innovation, which results in directly re-inventing and modernising the entire software development lifecycle and allows the business to quickly adapt to changing markets and customer needs.

- Solution to overcome the challenges:
  - Transitioning from the traditional software development approach to DevOps is not something you can do effectively overnight. Equipping your people with the right tools is essential, but the secret to successful DevOps implementation is creating the right culture in which everyone has their eye on the bigger picture and works together to achieve it. Adopting this mindset and following best practices in creating a delivery pipeline with CI/CD and automated DevSecOps will optimise the massive potential that DevOps can deliver.
  - Understanding the roles and responsibilities of where dev stops and ops currently starts, and how these can best be integrated together, is a great starting point for any company, and it’s often the first hurdle that it needs to overcome as it adopts DevOps practices.
  - Moving towards a more cloud-native ecosystem with microservices architecture can open up the floodgates to faster development and quicker innovation. In addition, it’s vital to have a solid foundation around automation, configuration management and continuous delivery practices to cope with increased operational workload that microservices bring.

- Benefits of implementing the DevOps:

They also were able to adopt the devops culture into their business teams by encouraging constant feedback, quick and continuous improvement. Here are some of the perfect examples of how adopting DevOps helped Nordstorm survive the market:

- Reduced deployment cycles from 3 months to almost 30 minutes.
- More frequent releases with the upgrade from the previous 28 weeks to now 4 weeks.
- Higher quality updates with better tracking of issues at hand.
- Improved CPU Utilisation from 5x to 12x depending on workloads which meant increased Operations efficiency.
- Increase in employee happiness
- Building at scale, complex systems with larger configurations still would work with the
- same resources.
- Better overall planning and goal achievements.

Conclusion: Students successfully studied about DevOps and understood the challenges faced but also the benefits gained while implementing it.