Introduction to Devops

1. DevOps is a philosophy for efficiently developing, deploying, and operating high-quality software.
2. It's not a single definition but rather a philosophy that allows each company to implement it in their own way.
3. DevOps aims to break down silos between development and operations teams to promote collaboration.
4. The main goal is to remove inefficiencies and promote higher quality in the software development process.
5. Modern software systems are complex, and any change introduced can lead to issues or disruptions.
6. Developers implement changes, while operations ensure stability, which can create conflicts.
7. DevOps promotes a culture of collaboration, automation, and measurement.
8. Users expect software to be reliable and of high quality.
9. DevOps helps identify constraints and areas for improvement in the development, deployment, and operations pipeline.
10. DevOps is not a product, brand, job title, or team, but an evolving philosophy.
11. The three main tenets of DevOps are **culture, automation, and measurement.**

▼ Why Culture is Important to DevOps

- DevOps culture is a key tenet of the philosophy, and it's important to understand what it means.
- Culture refers to the beliefs, values, and behaviors of a group, and a company's culture is its vision, values, and habits.
- Cultural changes are necessary to support DevOps and its tenets of automation and measurement.
- DevOps culture values automation and measurement, as they remove constraints and provide necessary data.

- Traditional company structures have separate teams or silos, with conflicting goals and incentives.
- Silos hinder collaboration and can lead to missed goals and inefficiencies.
- Cultural change requires breaking down silos and promoting collaboration and shared responsibility.
- Cross-functional teams, involving developers, QA, security, and operations engineers, can improve product stability, security, and quality.
- Introducing a separate DevOps team often creates another silo and doesn't solve the cultural challenges.
- A blameless culture is important, recognizing that failures happen and focusing on learning and improvement instead of blame.
- Collaboration, transparency, and empathy are key in a DevOps culture.
- Cultural change may be challenging but necessary for success, as a culture that supports employees leads to better outcomes.
- Automation is another important tenet of DevOps and helps remove constraints and improve efficiency.
- Automation should be embraced and applied where it makes sense in the development, deployment, and operations pipeline.

▼ Why Automation is Important to DevOps

- Definition of automation: Technique, method, or system of operating or controlling a process by highly automatic means, reducing human intervention to a minimum.
- Importance of automation in DevOps:
  - Reduces potential for human error in tasks, especially mundane and repetitive ones.
  - Provides consistency, predictability, scalability, and quality.
  - Helps remove or reduce constraints in the development, deployment, and operations pipeline.
  - Enables faster and more frequent deployments, improving efficiency.

    - Facilitates continuous integration (CI) and continuous delivery (CD) practices.
- Automation in the development pipeline:
    - Code is automatically built and tested after being committed to a shared repository (continuous integration).
    - Successful builds result in the creation of an artifact for predictable software installation.
- Automation in the deployment pipeline:
    - Various tasks are automated to enable push-button deployments.
    - Continuous delivery (CD) allows for selecting software versions and environments for deployment.
- Automation in the operations pipeline:
    - Infrastructure as Code (IaC) enables specifying and configuring servers and infrastructure through code.
    - Tools like Chef, Ansible, Puppet, and SaltStack are commonly used.
    - Infrastructure as Code allows for easy recreation of environments and consistent configurations.
    - Automation of log aggregation, log management, and monitoring improves visibility and decision-making.
- Suggestions for starting with automation:
    - Begin with a simple task like running a continuous integration server (e.g., Jenkins) to compile code and report any compilation issues.
    - Consider installing dependencies and running a linter for non-compiled languages.
- Automation improves the efficiency, speed, scalability, and quality of the pipeline.
- Metrics to quantify the success of DevOps efforts will be discussed in the next lecture.

▼ Why Metrics are Important to DevOps

- Metrics play an important role in quantifying the success of DevOps practices.
- Different roles in the team will have different metrics to help them perform their job effectively.
- Specific metrics related to DevOps practices include:
  - **Frequency of deployments:** Measures how often software is deployed, indicating the efficiency of the development, deployment, and operations pipeline.
  - **Mean time to recovery (MTTR):** Measures the average time taken to resolve problems or issues that negatively impact end users in the production environment. The goal is to decrease MTTR over time.
  - **Mean time to discovery (MTTD):** Measures how quickly problems or failures are identified in the production environment, helping to identify potential bottlenecks in the resolution process.
  - **System availability:** Determines the uptime of each system component and the overall uptime percentage, enabling the identification of areas that require attention.
  - **Service performance:** Tracks the performance of systems and checks if they meet desired thresholds, such as response times for APIs or web page loading times.
  - **Customer complaints:** Monitors user feedback and complaints to identify recurring issues and incorporate preventative measures into the pipeline.
  - **Lead time:** Measures the time taken to go from a feature request to its release, emphasizing the importance of delivering features quickly without compromising quality.
- These metrics should be considered in the context of the entire system and used as a starting point for improvement, rather than being used to place blame or set arbitrary goals.
- The lecture hints at future discussions on the business value of DevOps, including improved lead times, enhanced stability and quality, and reduced operational costs.

▼ How DevOps can Improve Lead Time

- Lead time in the context of software development and deployment is the total time it takes for an idea or feature request to go from request to being released.
- DevOps aims to balance the potential instabilities caused by constant change introduced by developers with the need for stable operations.
- Mistakes are inevitable, but catching them early in the development process is more cost-effective than catching them in production.
- Acme Products Unlimited (APU), a fictitious company, had infrequent deployments and faced issues with site stability and manual source-based deployments.
- APU lacked a collaborative culture and automation in their processes.
- To improve lead time, APU adopted a DevOps philosophy and implemented the following changes:
  - Measuring and establishing baselines for lead time, uptime, deployment frequency, mean time to detection (MTTD), and mean time to recovery (MTTR).
  - Forming autonomous cross-functional teams comprising developers, QA, security, and operations engineers.
  - Implementing a fully automated continuous integration and continuous delivery (CI/CD) pipeline using tools like Jenkins.
  - Using feature toggles to ensure that unreleased features don't impact production code.
  - Adopting a blue-green deployment model and an elastic infrastructure in the cloud.
- APU's efforts resulted in multiple deployments per day, improved stability, reduced downtime, and better lead time.
- To improve lead time, it's essential to review the entire development, deployment, and operations pipeline, identify constraints, and plan ways to remove or reduce those constraints.
- Automation plays a crucial role in achieving faster lead time and a more consistent process.

▼ How DevOps can Improve Stability

Key points from the lecture on improving system stability through DevOps:

- DevOps addresses the tension between development-driven change and the desire for stability in operations.
- The fictitious company, Acme Products Unlimited (APU), experienced stability issues before adopting DevOps practices.
- Changes that had the most impact on stability in APU's scenario were:
    - Cultural change: Operations engineers became part of cross-functional teams, collaborating with developers, QA, and security engineers, leading to a more secure and stable product.
    - Use of metrics: Baseline measurements allowed engineers to identify if changes were impacting system performance and quickly address any issues.
    - Automation: Continuous integration processes helped identify and prevent bugs from reaching production, contributing to stability. Elastic infrastructure scaled automatically to meet traffic demands. Immutable server models ensured stable deployments, and a blue-green deployment model facilitated smooth transitions between changes.
- DevOps improves stability by emphasizing the integration of quality into software, automating infrastructure management, and making deployments predictable and consistent.
- Little changes implemented collectively in a DevOps approach contribute to stability.

In the next lecture, the focus will be on how DevOps can lead to reduced operational costs in the context of the APU scenario.

▼ How DevOps can Reduce Operational Costs

Key points from the lecture on reducing operational costs through DevOps:

- Downtime can be costly, and the impact varies depending on the company and the nature of the services. The availability of services can affect the bottom line, and public perception of the product can also be influenced by outages and security issues.

- Acme Products Unlimited (APU) improved operational costs through various measures:
    - They achieved a more stable product by improving code quality and operations.
    - Implementing faster lead times allowed them to deliver features to users more quickly, reducing the risk of losing customers to competitors.
    - The reduction of unplanned work freed up engineers' time to focus on value-added tasks, improving efficiency and productivity.
    - By reducing stress and addressing firefighting scenarios, APU likely experienced lower turnover rates and improved morale among their engineering team.
- These factors combined lead to improved operational costs in terms of efficiency, productivity, and employee satisfaction.
- The next lecture will provide real-world examples of companies known for their successful DevOps practices.

In the next lecture, you'll have the opportunity to explore actual case studies of companies that have implemented effective DevOps practices.

▼ SUMMARY

- DevOps is a philosophy of efficient development, deployment, and operation of high-quality software.
- DevOps is not bound to specific technologies and allows companies to adopt practices that work best for them.
- Continuous Integration (CI) is a practice where developers commit code to a shared repository, which is automatically built and tested.
- Jenkins is a popular CI server that automates the build and test process for each code commit.
- Continuous Delivery (CD) is the practice of being able to deploy code with the push of a button.
- CD uses the artifacts created by CI to ensure consistency throughout the deployment process.

- CI and CD are commonly used practices in DevOps and contribute to high-quality software and pipeline efficiency.
- DevOps is more than just a buzzword; it involves culture, automation, and measurement.
- A successful DevOps strategy can lead to improved lead time, stability, and reduced operational costs.
- Real companies like Etsy, Netflix, and Amazon have successfully adopted DevOps practices.
- Future courses may cover CI and CD in more depth for practical implementation.