Emerging and Prominent Tech in Dev Ops

PowerPoint Presentation in Software Management and Testing
LCA

Group Members and their contributions

1. Krishnaraj Thadesar - PA20
2. Parth Zarekar - PA15
3. Mayur Behere - PA03
4. Nishad Wanjari - PA09

Contents

Introduction to DevOps and Continuous Integration

Krishnaraj Thadesar

What are Dev Ops?

DevOps is a set of practices that combines software development (Dev) and IT operations (Ops) to create a more collaborative and efficient workflow. By breaking down silos between development and operations teams, DevOps aims to improve communication, speed up software delivery, and increase the overall quality of software products. In this presentation, we will explore some of the key principles of DevOps, including Continuous Integration, Continuous Delivery, Infrastructure as Code, and Monitoring and Logging.

What is continuous integration?

Continuous Integration is a software development practice that involves frequently integrating small code changes into a shared repository. The main goal of CI is to detect and address issues early in the development process. CI ensures that each change made by a developer is built, tested, and integrated into the codebase. This approach leads to more reliable software, as well as faster and more efficient development cycles.

Why is it important in DevOps?

In the past, developers on a team might work in isolation for an extended period of time and only merge their changes to the master branch once their work was completed. This made merging code changes difficult and time-consuming, and also resulted in bugs accumulating for a long time without correction. These factors made it harder to deliver updates to customers quickly.

The benefits of continuous integration

Early detection of bugs: With CI, developers can catch bugs early in the development cycle, making it easier to fix them.

Faster feedback: Since CI provides feedback quickly, developers can make changes and test them faster, leading to shorter development cycles.

Easier collaboration: CI promotes better collaboration between team members, allowing them to share code changes and resolve conflicts more easily.

Setting up a continuous integration pipeline - Key Components

Source Control Management (SCM): SCM systems are used to manage the source code and track changes made to it. Popular SCM systems include Git, Subversion, and Mercurial.

Build Automation: Build automation tools are used to automate the process of building the software. Tools like Jenkins and Travis CI can be used to automatically compile, test, and package the software.

Test Automation: Test automation tools are used to automate the process of running tests. Tools like JUnit and Selenium can be used to automatically run unit tests and functional tests.

Continuous Deployment: Continuous Deployment involves automatically deploying the software to production servers after it has been built and tested. Tools like Ansible and Puppet can be used to automate the deployment process.

Best practices for continuous integration

Use a reliable SCM system: Use a reliable SCM system to manage the source code and track changes. This will make it easier to collaborate and manage changes.

Automate the build process: Automating the build process can save time and ensure consistency. It also reduces the risk of human error.

Run tests frequently: Running tests frequently can catch bugs early in the development cycle, making them easier to fix.

Continuous Integration Tools

Jenkins: Jenkins is an open-source CI tool that is widely used. It is highly configurable and can be customized to suit the needs of the team.

Travis CI: Travis CI is a cloud-based CI tool that is easy to set up and use. It is particularly suited for open-source projects.

CircleCI: CircleCI is a cloud-based CI tool that is known for its ease of use and speed. It is particularly suited for small to medium-sized teams.

GitLab CI: GitLab CI is a part of the GitLab platform and provides continuous integration and deployment features. It

Continuous Development

Nishad Wanjari

Continuous Delivery (CD)

- Continuous Delivery is a software development practice where code changes are automatically built, tested, and deployed to production.
- DevOps is a culture and set of practices that aim to bridge the gap between development and operations teams for faster and more efficient software delivery.

The Continuous Delivery Pipeline

- The Continuous Delivery pipeline is a series of automated steps that code changes go through from development to production.
- The pipeline consists of Continuous Integration (CI), Automated Testing, and Continuous Deployment (CD).

- Continuous Delivery pipeline can be customized based on the needs of the organization, such as adding additional testing stages or integrating with other tools and services.

- The goal of the Continuous Delivery pipeline is to enable software teams to deliver high-quality software faster and more efficiently.

Benefits of Continuous Delivery and DevOps

- Faster time-to-market: Continuous Delivery and DevOps enable software teams to release new features and updates quickly and frequently, reducing the time it takes to get new products to market.
- Improved software quality: Automated testing and code reviews ensure that software is tested and validated before it is released to production, leading to higher-quality software with fewer bugs.

- Increased customer satisfaction: Faster delivery of high-quality software leads to greater customer satisfaction and improved user experience.
- Reduced costs and risks: Automation reduces the risk of human error and enables teams to catch and fix issues earlier in the development process, resulting in lower costs and fewer defects in production.

Infrastructure as Code

Parth Zarekar

What is Infrastructure as Code (IAC)?

- Infrastructure as Code (Iac) is the practice of managing and provisioning infrastructure through code.
- Iac uses machine-readable files to define automate the creation, modification, and deletion of infrastructure resources.
- Unlike traditional infrastructure management, which involves manual processes and configuration changes, IaC enables organizations to manage infrastructure in a repeatable, consistent, and scalable way.

Benefits of IaC

- **Improved Scalability:** IaC enables organizations to scale infrastructure resources up or down based on demand, reducing the time and cost of manually provisioning new resources.
- **Reduced risk of human error:** IaC minimizes the risk of human error by automating infrastructure changes and reducing the need for manual intervention.

- **Increased automation and efficiency:** IaC enables organizations to automate the creation, configuration, and management of infrastructure resources, reducing the time and effort required for these tasks.
- **Improved collaboration and version control:** IaC enables teams to collaborate on infrastructure changes and version control, ensuring that all changes are tracked, reviewed, and approved before being deployed.

Working of IaC

- Define infrastructure using code: The first step is to define your infrastructure resources, such as servers, storage, networks, and other components, using code. This code is written in a language that is specific to the IaC tool you are using, such as Terraform, CloudFormation, or Ansible.
- Store code in version control: The next step is to store your infrastructure code in version control, such as Git. This makes it easy to track changes, collaborate with others, and roll back changes if necessary.
- Update and maintain infrastructure: As your infrastructure needs change over time, you can update and maintain your infrastructure code to reflect those changes. This involves making changes to the code and then using automation to deploy those changes to your target environment.

Challenges and Considerations

- **Learning curve for new tools and processes:** IaC requires new skills and processes that may take time to learn and implement.
- **Security considerations:** IaC introduces new security risks that need to be managed, such as managing secrets and securing access to infrastructure resources.
- **Integration with existing infrastructure:** IaC may need to integrate with existing infrastructure and tools, which may require additional work to configure and maintain.
- **Cultural resistance to change:** IaC may face resistance from teams that are used to managing infrastructure manually or using traditional tools.

So What’s Monitoring and Logging?

- Monitoring and logging are crucial aspects of DevOps practices
- They help teams to identify and fix issues quickly, reduce downtime, and improve system reliability
- In this presentation, we'll discuss the importance of monitoring and logging in DevOps and how to implement them effectively

Monitoring and Logging

Mayur Behere

Monitoring

- Monitoring involves collecting and analyzing metrics and events to ensure that the system is performing as expected
- It helps teams to detect and diagnose issues quickly and respond to them before they cause downtime or other problems
- Examples of monitoring tools include Prometheus, Nagios, and Zabbix
- Monitoring can be done at different levels, including infrastructure, application, and business

Logging

- Logging involves capturing and storing data about the system's activities, such as events, errors, and user actions
- It helps teams to troubleshoot issues, analyze system behavior, and track user activity
- Examples of logging tools include Elasticsearch, Logstash, and Kibana
- Logging can be used for various purposes, such as security auditing, compliance, and performance analysis

DevOps Monitoring and Logging Best Practices

- Use automation to streamline the monitoring and logging processes
- Define clear metrics and logging policies that align with business goals and objectives
- Monitor the system proactively, instead of reacting to issues
- Use dashboards and alerts to quickly identify and respond to issues
- Regularly review and analyze the data to identify areas for improvement

Why Monitoring and logging?

- Effective monitoring and logging are crucial for maintaining system health and reliability in DevOps.
- Proactive monitoring and logging, clear policies, automation, and data analysis are essential best practices.
- The continuous improvement of monitoring and logging practices is necessary for the system's ongoing success.
- By implementing these best practices, DevOps teams can identify and fix issues quickly, reduce downtime, and improve overall system performance.

Conclusion

- Emerging technologies like AI, ML, and Serverless Computing are improving the DevOps process.
- These technologies have the potential to accelerate software development, reduce errors and downtime, and improve system reliability and performance.
- Careful evaluation and implementation are necessary to ensure success.
- Overall, the adoption of these technologies has the potential to transform the software development process.

References

Chat OpenAI >< :)

Humble, J., & Farley, D. (2010). Continuous delivery: Reliable software releases through build, test, and deployment automation. Addison-Wesley Professional.

Hassan, S., Zhang, K., & Mockus, A. (2017). Achieving scalability and elasticity in continuous delivery pipelines. IEEE Transactions on Software Engineering, 44(5), 441-459.

Hashicorp. (n.d.). Infrastructure as code. Retrieved from What is infrastructure as code and why is it important?

New Relic. (n.d.). Monitoring and observability. Retrieved from https://newrelic.com/products/monitoring-and-observability