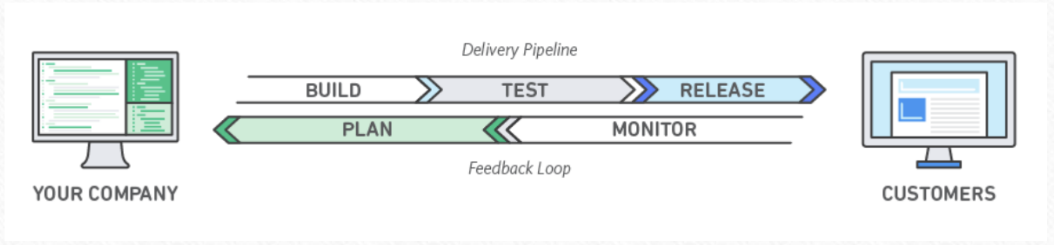
DevOps

BY: CHHAVI, CHHAVI; PEARSON, JACOB MICHEAL

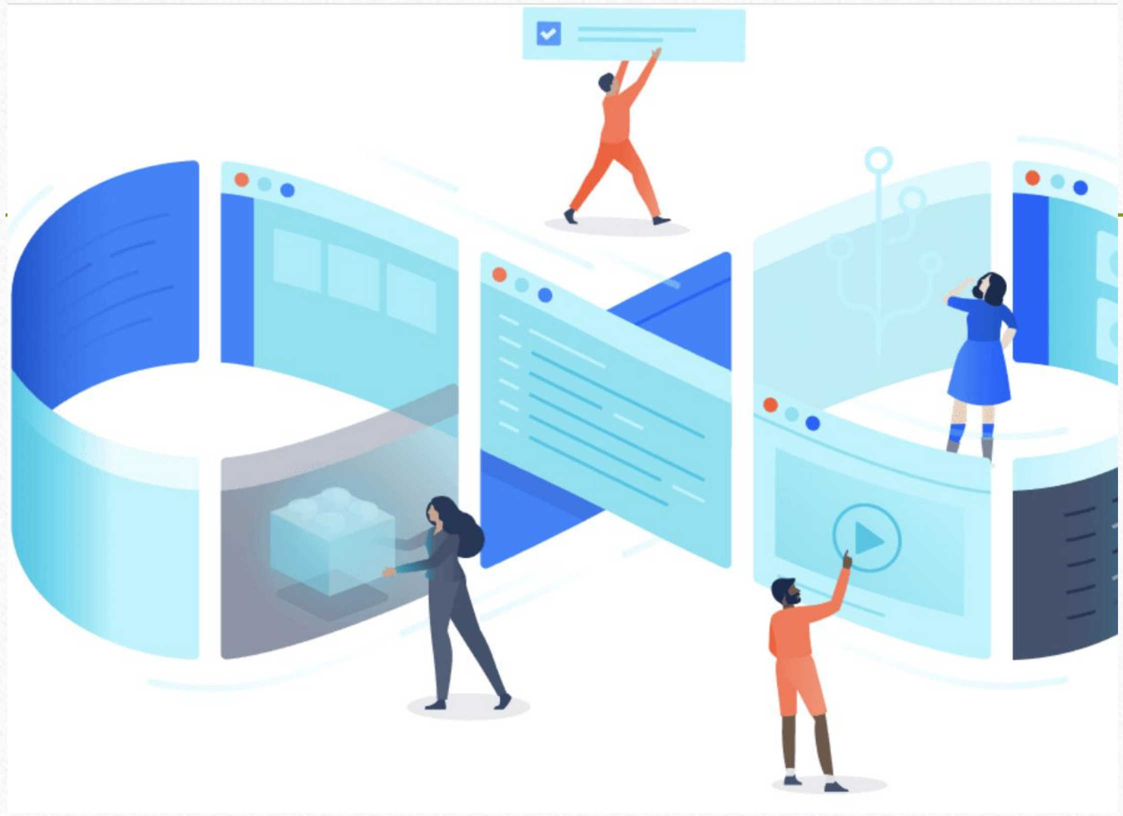
What is DevOps?



- 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. This speed enables organizations to better serve their customers and compete more effectively in the market.

DevOps Practices

DevOps practices focus on improving collaboration, communication, and automation between development (Dev) and operations (Ops) teams to increase the efficiency, reliability, and speed of software delivery.



DevOps Tools

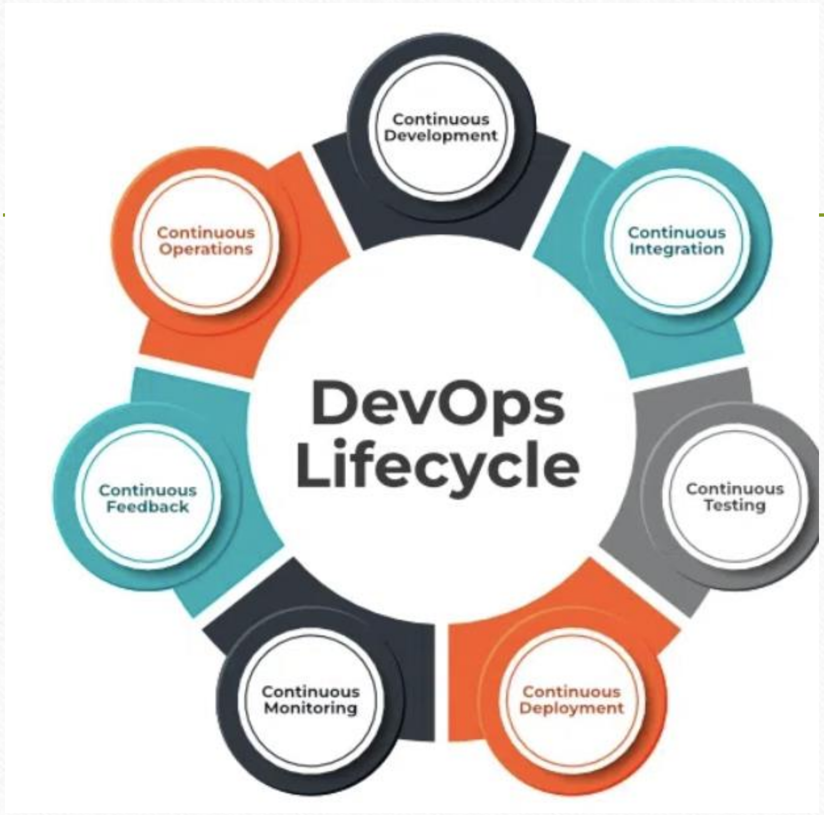
- The DevOps model relies on effective tooling to help teams rapidly and reliably deploy and innovate for their customers. These tools automate manual tasks, help teams manage complex environments at scale, and keep engineers in control of the high velocity that is enabled by DevOps. AWS provides services that are designed for DevOps and that are built first for use with the AWS cloud. These services help you use the DevOps practices described above.

SOME DevOps TOOLS

- **Git**- Git is the commonly used tool in DevOps and the clear winner because of powerful features like branching and merging, enabling seamless collaboration and version management in complex projects. It’s a free, open-source version control system that is easy to get started with a minimal footprint and fast performance.
- **GitHub**- GitHub is the default and most broadly used code repository management system. It provides an easy way to manage distributed version control projects along with many more features and functionalities such as feature requests, task management, CI/CD, wikis, and more to enable developers.
- **AWS CloudFormation**- AWS CloudFormation lets you model, provision, and manage AWS and third-party resources with infrastructure as code principles. Provides native integrations with other AWS services to build a robust infrastructure management pipeline. Here you can see a detailed CloudFormation vs. Terraform comparison.
- **AWS CDK**- The AWS Cloud Development Kit(CDK) allows you to define cloud application resources and infrastructure components using programming languages. It enables developers to use the same language for building applications and infrastructure with the same language they are familiar with.

DevOps Cultural Philosophy

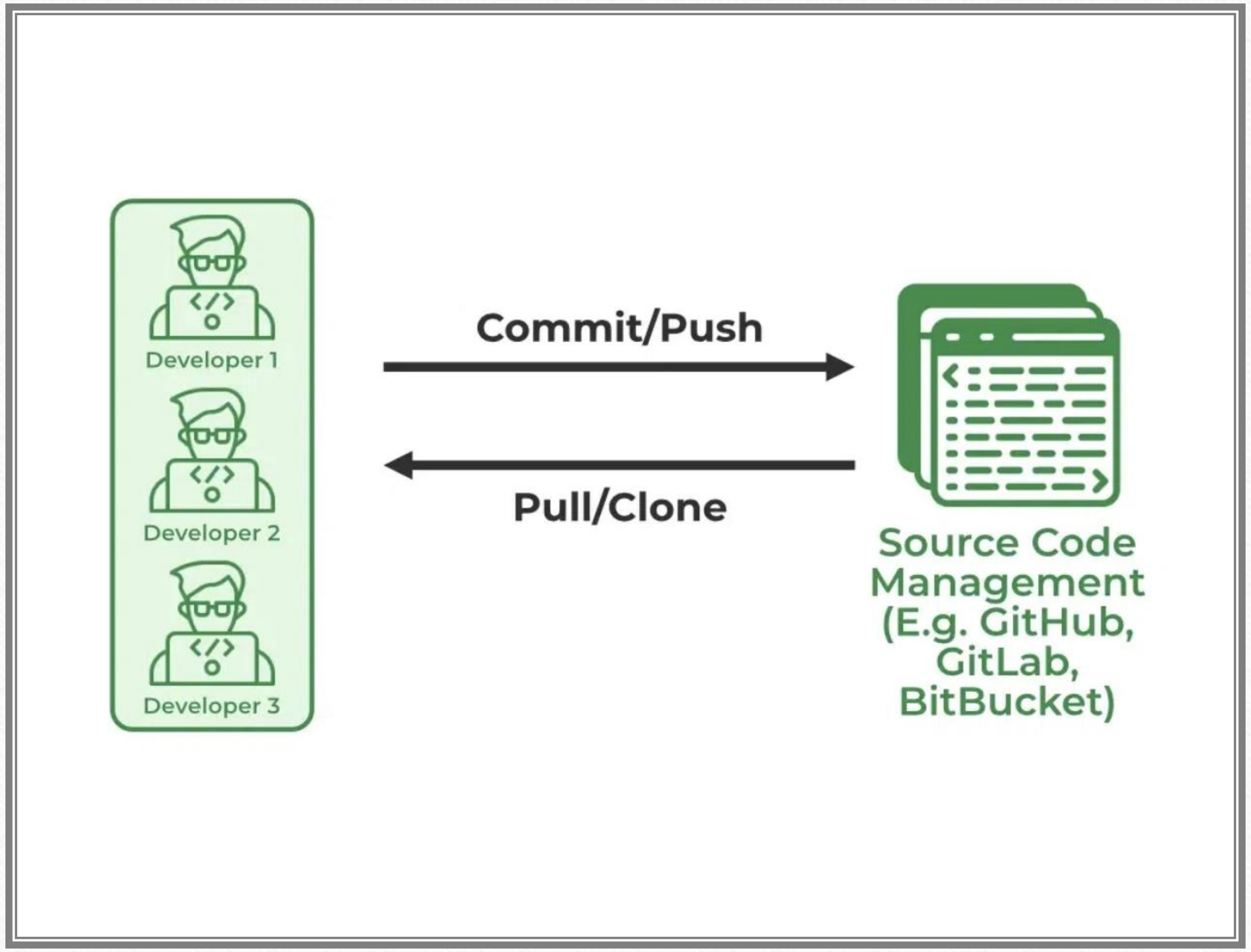
- Transitioning to DevOps requires a change in culture and mindset. At its simplest, DevOps is about removing the barriers between two traditionally siloed teams, development and operations. In some organizations, there may not even be separate development and operations teams; engineers may do both. With DevOps, the two teams work together to optimize both the productivity of developers and the reliability of operations. They strive to communicate frequently, increase efficiencies, and improve the quality of services they provide to customers. They take full ownership for their services, often beyond where their stated roles or titles have traditionally been scoped by thinking about the end customer’s needs and how they can contribute to solving those needs. Quality assurance and security teams may also become tightly integrated with these teams. Organizations using a DevOps model, regardless of their organizational structure, have teams that view the entire development and infrastructure lifecycle as part of their responsibilities.



DevOps Lifecycle

- DevOps Lifecycle is the set of phases that includes DevOps for taking part in Development and Operation group duties for quicker software program delivery. DevOps follows positive techniques that consist of code, building, testing, releasing, deploying, operating, displaying, and planning. DevOps lifecycle follows a range of phases such as non-stop development, non-stop integration, non-stop testing, non-stop monitoring, and non-stop feedback.

Continuous Development



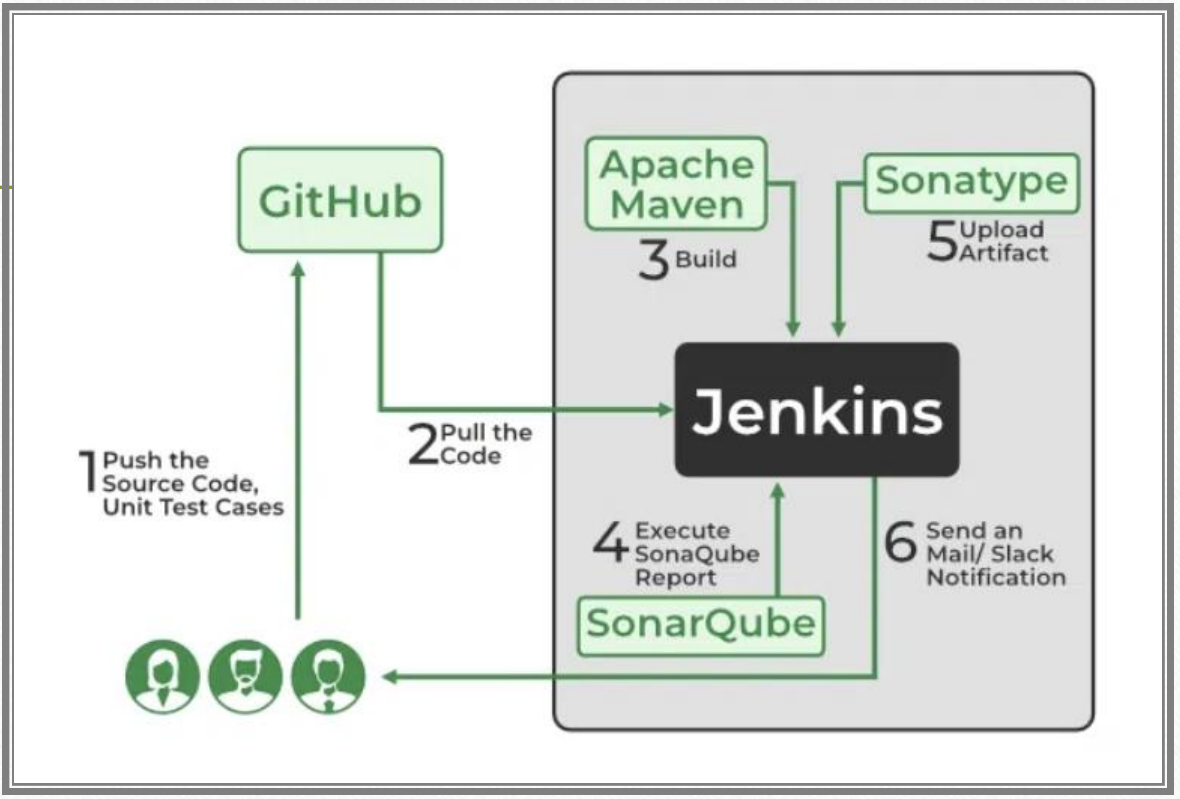
In Continuous Development code is written in small, continuous bits rather than all at once, Continuous Development is important in DevOps because this improves efficiency every time a piece of code is created, it is tested, built, and deployed into production. Continuous Development raises the standard of the code and streamlines the process of repairing flaws, vulnerabilities, and defects. It facilitates developers' ability to concentrate on creating high-quality code.

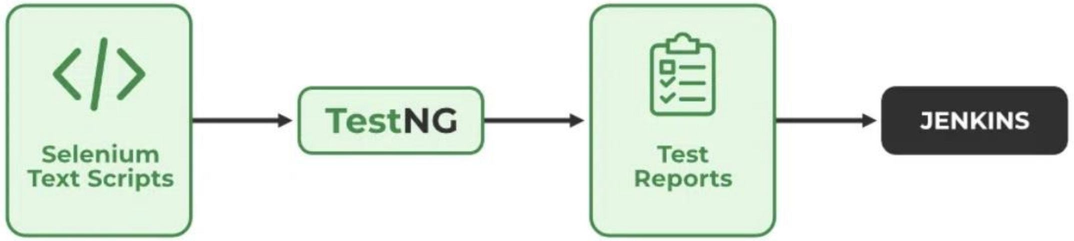
.

Continuous Integration

Continuous Integration can be explained mainly in 4 stages in DevOps.

1. Getting the Source Code from SCM
2. Building the code
3. Code quality review
4. Storing the build artifacts


GitHub
Apache Maven
3 Build
Sonatype
5 Upload Artifact
Jenkins
2 Pull the Code
1 Push the Source Code, Unit Test Cases
4 Execute SonaQube Report
6 Send an Mail/ Slack Notification
SonarQube

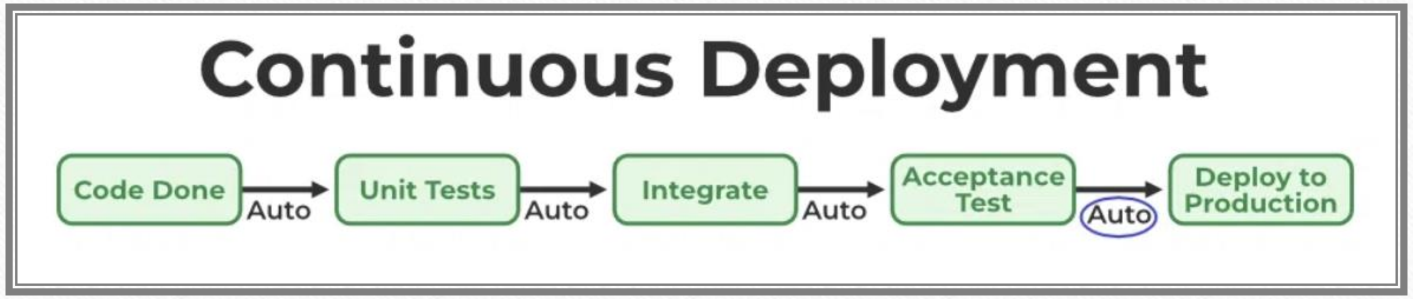


Continuous Testing

- Any firm can deploy continuous testing with the use of the agile and DevOps methodologies. Depending on our needs, we can perform continuous testing using automation testing tools such as Testsigma, Selenium, LambdaTest, etc. With these tools, we can test our code and prevent problems and code smells, as well as test more quickly and intelligently. With the aid of a continuous integration platform like Jenkins, the entire process can be automated, which is another added benefit.

Continuous Deployment

- It is the process of automatically deploying an application into the production environment when it has completed testing and the build stages. Here, we'll automate everything from obtaining the application's source code to deploying it.



Continuous Monitoring

DevOps lifecycle is incomplete if there was no Continuous Monitoring. Continuous Monitoring can be achieved with the help of Prometheus and Grafana we can continuously monitor and can get notified before anything goes wrong with the help of Prometheus we can gather many performance measures, including CPU and memory utilization, network traffic, application response times, error rates, and others. Grafana makes it possible to visually represent and keep track of data from time series, such as CPU and memory utilization.

Continuous Feedback

- Once the application is released into the market the end users will use the application and they will give us feedback about the performance of the application and any glitches affecting the user experience after getting multiple feedback from the end users' the DevOps team will analyze the feedbacks given by end users and they will reach out to the developer team tries to rectify the mistakes they are performed in that piece of code by this we can reduce the errors or bugs that which we are currently developing and can produce much more effective results for the end users also we reduce any unnecessary steps to deploy the application. Continuous Feedback can increase the performance of the application and reduce bugs in the code making it smooth for end users to use the application.

Continuous Operations

- We will sustain the higher application uptime by implementing continuous operation, which will assist us to cut down on the maintenance downtime that will negatively impact end users' experiences. More output, lower manufacturing costs, and quality control are benefits of continuous operations.

Importance of DevOps

1) Shorter Development Cycles, Faster Innovation- Streamlined workflows enable rapid development, testing, and release, accelerating innovation and time-to-market.
2) Reduced Deployment Failures, Rollbacks, and Time to Recover- Automated testing and early issue detection reduce deployment failures, making it easier to rollback and quickly recover from issues.
3) Improved Communication and Collaboration- Cross-functional teams and shared goals break down silos, fostering a collaborative environment
4) Increased Efficiencies- Automation of repetitive tasks and optimization of processes improve productivity and reduce manual work, allowing teams to focus on high-value tasks.

BENEFITS OF DevOps

1) Collaboration and Trust- Building a culture of shared responsibility, transparency, and faster feedback is the foundation of every high-performing DevOps team.
2) Release faster and work smarter- Speed is everything. Teams that practice DevOps release deliverables more frequently, with higher quality and stability.
3) Accelerate time-to-resolution- The team with the fastest feedback loop is the team that thrives.
4) Better manage unplanned work- Unplanned work is a reality that every team faces–a reality that most often impacts team productivity.

DevOps CHALLENGES

1) Challenges with Team Maturity and Competence- The level of maturity and competence a software engineering team has with the Software Development Life Cycle is directly related to their ability to be able to adapt to the DevOps transformation of these processes.
2) Challenges with Monitoring the overall DevOps process- One of the most common problems with DevOps is the challenge in holistically monitoring the entire process. DevOps consists of several moving parts and each of these have different metrics to judge their effectiveness.
3) CI/CD Performance Issues- A suboptimal implementation of the CI/CD pipeline leads to recurring issues such as slow page loads for websites, delayed responses from servers, and poor memory optimization that hampers the overall performance of the application.

CI/CD AND DevOps

CI/CD is a set of practices and tools that enable the automation of software development, testing, and deployment. The goal of CI/CD is to improve the time-to-release and reliability of software releases. They do so by detecting and fixing bugs early in the development process.

DevOps is a software development methodology that emphasizes collaboration and communication between developers and operations teams. DevOps culture promotes a shared responsibility for the entire software development life cycle, from development to production. This enables teams to work together more effectively, improving the speed and quality of software development and deployment.

DevOps vs CI/CD

CI/CD	**DevOps**
Involves the use of automated testing and continuous integration	Involves the use of infrastructure as code, containerization, and automation of infrastructure provisioning
Focuses on creating a rapid feedback loop for developers	Focuses on creating a culture of collaboration and shared responsibility between development and operations teams
Emphasizes the use of automation to reduce human error and improve consistency	Emphasizes the use of monitoring and logging to identify and resolve issues quickly
Tools include Jenkins, Travis CI, and CircleCI	Tools include Ansible, Puppet, and Chef
Involves the use of version control systems such as Git	Involves the use of agile methodologies and continuous testing
Focuses on streamlining and automating the software release process	Focuses on improving communication and collaboration between development and operations teams to achieve faster and more reliable software releases.

Is CI/CD and agile the same?

CI/CD and Agile are not the same, but they are often used together. Both CI/CD and Agile are used to improve the software development process, but they have different focuses. Agile focuses on delivering working software incrementally, with regular feedback and collaboration, while CI/CD focuses on automating the software build and deployment process. They can be used together to improve the software development process by implementing automation in the development process and delivering working software incrementally and iteratively.

DevOps Practices

The practice of DevOps encourages smoother, continuous communication, collaboration, integration, visibility, and transparency between application development teams (Dev) and their IT operations team (Ops) counterparts.

1) Collaboration- The key premise behind DevOps is collaboration. Development and operations teams coalesce into a functional team that communicates, shares feedback and collaborates throughout the entire development and deployment cycle.
2) Automation- An essential practice of DevOps is to automate as much of the software development lifecycle as possible. This gives developers more time to write code and develop new features.
3) Continuous Improvement- It's the practice of focusing on experimentation, minimizing waste, and optimizing for speed, cost, and ease of delivery.
4) Customer-centric action- DevOps teams use short feedback loops with customers and end users to develop products and services centered around user needs.
5) Create with the end in mind- This principle involves understanding the needs of customers and creating products or services that solve real problems.

Some more DevOps Practices

1. Infrastructure as Code: Treating infrastructure configuration and provisioning as code makes it possible to version, reuse, and automate infrastructure. This enables teams to replicate and scale environments quickly and maintain consistency across development, testing, and production.
2. Monitoring and Logging: Continuous monitoring provides real-time visibility into applications and infrastructure, helping teams detect, diagnose, and resolve issues quickly. Logging further helps analyze system performance and user behaviors, facilitating continuous improvement.
3. Microservices Architecture: Designing applications as a suite of small, loosely coupled services allows teams to develop, test, deploy, and scale each service independently, improving flexibility, reliability, and resilience.
4. Version Control: Using version control for code and infrastructure configurations (often with Git) helps manage changes, track history, collaborate effectively, and roll back to previous versions if necessary.

DevOps Architecture

DevOps Architecture consists of the tools and workflows that allow for efficient software development, deployment, and monitoring. With optimal DevOps architecture the process of creating a program should be a smooth process that works in tandem with the Agile framework to allow for minimal interruptions at any point.

DevOps Architecture(Continued)

Creating a DevOps Architecture plan can seem very similar to creating a CD/CI pipeline, just wider in scope. With DevOps, not only is the coding to deployment pipeline considered, but also the team structure and planning. DevOps integrates many of the topics we've covered in this course to make for a more efficient development process. A complete DevOps Architecture plan can make use of everything from CD/CI and Scrum, to Agile and Kanban boards.

DevOps Infrastructure

DevOps Infrastructure involves making development as smooth as possible by means of automation. This can cover everything from server management to automated deployment and monitoring. While not easy or simple to institute, DevOps infrastructure can allow for massively increased productivity since developers will no longer need to waste time doing tasks that could be automated.

DevOps Infrastructure Planning

DevOps Infrastructure is, as previously mentioned, difficult and expensive to implement. That is why the planning stage of the creation of it can also be the most important. During the planning stage the goals and requirements of the infrastructure must be identified. Since the creation of the infrastructure can be time and resource intensive, it is essential for the this to be done carefully and with forethought.

DevOps Infrastructure Planning(cont.)

The planning of DevOps Infrastructure should also take into account the need to scale in the future. This can be accomplished through the analysis of infrastructure data, as well as through the use of tools such as Grafana. The infrastructure should be built to the needs of the future throughout its lifecycle, not just current requirements.

DevOps and Agile

DevOps was the natural continuation of Agile in many ways. While Agile focuses on how a development team works together with stakeholders, DevOps exists to remove any barriers for that team by focusing on communication with the operations team as well. Essentially, a team within a working DevOps architecture should be able to work on a portion of a project from start to finish without needing to wait for oversight or review. This is done through careful planning and communication between the teams involved.

DevOps and Agile(cont.)

The Agile Manifesto explicitly prioritizes individuals and interactions, working software, customer collaboration, and responding to change. These are clearly the same priorities of DevOps but extended beyond the development process and into the management of systems and running applications.

Agile centers the flow of software from ideation to code completion — DevOps extends the focus to delivery and maintenance

Agile emphasizes iterative development and small batches — DevOps focuses more on test and delivery automation

DevOps and CI

Continuous integration is an integral part of DevOps. CI allows for the automated building and testing of code without the need for developer interaction. DevOps' goal is to increase the efficiency of the development process, and automated pipelines such as CI and CD can allow for an incredibly large amount of time to be saved, especially on larger projects.

DevOps and CI(cont.)

Both CI/CD and DevOps focus on automating processes of code integration, thereby speeding up the processes by which an idea (like a new feature, a request for enhancement, or a bug fix) goes from development to deployment in a production environment where it can provide value to the user.

DevSecOps

DevSecOps stands for development, security, and operations. It's an approach to culture, automation, and platform design that integrates security as a shared responsibility throughout the entire IT lifecycle. Effective DevOps ensures rapid and frequent development cycles (sometimes weeks or days), but outdated security practices can undo even the most efficient DevOps initiatives. This is why it is important to consider security at every stage of the DevOps process.

DevSecOps(cont.)

Whether you call it "DevOps" or "DevSecOps," it has always been ideal to include security as an integral part of the entire app life cycle. DevSecOps is about built-in security, not security that functions as a perimeter around apps and data. If security remains at the end of the development pipeline, organizations adopting DevOps can find themselves back to the long development cycles they were trying to avoid in the first place.

DevSecOps(cont.)

A good DevSecOps strategy is to determine risk tolerance and conduct a risk/benefit analysis. What amount of security controls are necessary within a given app? How important is speed to market for different apps? Automating repeated tasks is key to DevSecOps, since running manual security checks in the pipeline can be time intensive.

DevSecOps(cont.)

DevSecOps helps ensure that security is addressed as part of all DevOps practices by integrating security practices and automatically generating security and compliance artifacts throughout the process. This is important for several reasons, including:

- **Reduces vulnerabilities, malicious code, and other security issues** in released software without slowing down code production and releases.
- **Mitigates the potential impact of vulnerability exploitation throughout the application lifecycle**, including when the code is being developed and when the software is executing on dynamic hosting platforms.
- **Addresses the root causes of vulnerabilities to prevent recurrences**, such as strengthening test tools and methodologies in the toolchain, and improving practices for developing code and operating hosting platforms.
- **Reduces friction between the development, operation, and security teams** in order to maintain the speed and agility needed to support the organization's mission while taking advantage of modern and innovative technology.

DevOps with cloud services

When DevOps teams work in the cloud, they enjoy easier access to scalable hardware resources that can help build, test, and deploy new updates and offerings more quickly. The popularity of cloud application delivery has led to widespread adoption of DevOps methods because they are well suited to the rapid, ongoing processes that are a key benefit of cloud operations.

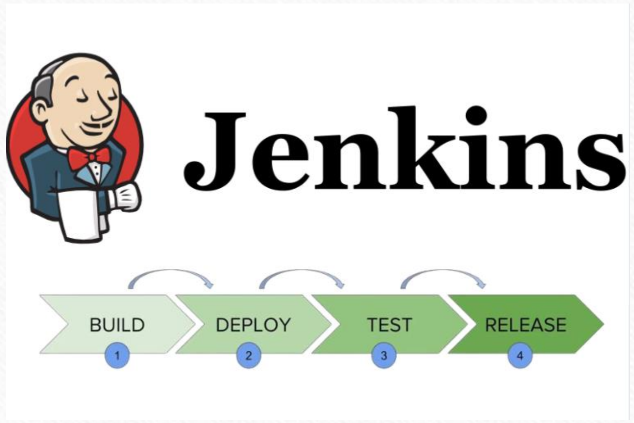
In a traditional application delivery environment, a finished application might be handed off to IT operations for maintenance, with future upgrades managed on a predetermined schedule. With cloud based projects however, this is often not the best course of action.

DevOps with cloud services

In cloud computing the application stack is likely to continue changing after its initial deployment. That dynamism is a benefit because it helps the organization to continue improving its products and services. Those rapid iterations also pose a challenge, and a DevOps framework helps organizations to remain responsive and competitive in a rapidly evolving marketplace. DevOps affinity for cloud services can easily be seen in the major companies that utilize it. Amazon is able to release thousands of updates to its websites per day with the help of DevOps, Netflix can release hundreds as well. This would be impossible without DevOps automating as many steps as possible.

DevOps Related Tools: Jenkins

Jenkins is a tool that is used for automation. It is mainly an open-source server that allows all the developers to build, test and deploy software. Jenkins facilitates the automation of several stages of the software development lifecycle, including application development, testing, and deployment. Operating within servlet containers like Apache Tomcat, the technology is server-based. Continuous delivery (CD) and integration (CI) pipelines can be created and managed with Jenkins. The development, testing, and deployment of software applications are automated using CI/CD pipelines.

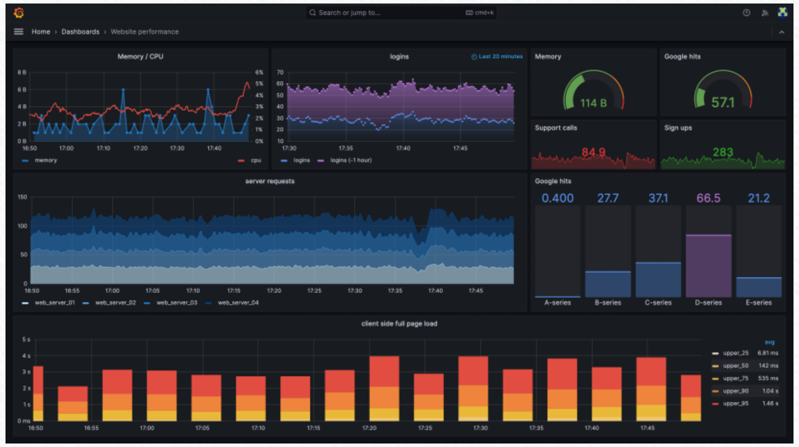


DevOps Related Tools: Terraform

HashiCorp Terraform is an infrastructure as code tool that lets you define both cloud and on-premisis resources in human-readable configuration files that you can version, reuse, and share. You can then use a consistent workflow to provision and manage all of your infrastructure throughout its lifecycle. Terraform can manage low-level components like compute, storage, and networking resources, as well as high-level components like DNS entries and SaaS features.

DevOps Related Tools: Grafana

Grafana is a service that allows data and analytic visualization from nearly any source. It is widely used with DevOps in order to help monitor automated processes and to better understand the requirements of the DevOps project.



Works Cited

- https://aws.amazon.com/devops/what-is-devops/
- https://spacelift.io/blog/devops-tools
- https://www.geeksforgeeks.org/devops-lifecycle/
- https://www.atlassian.com/devops/what-is-devops#:~:text=higher%20quality%20output.-,Automation,errors%20and%20increase%20team%20productivity
- https://shadow-soft.com/content/why-devops-important
- https://www.atlassian.com/devops/what-is-devops/benefits-of-devops
- https://testsigma.com/blog/devops-vs-cicd/

Works Cited

- https://instatus.com/blog/devops-infrastructure
- https://www.atlassian.com/devops/what-is-devops/agile-vs-devops
- https://www.redhat.com/en/topics/devops/what-is-ci-cd
- https://www.redhat.com/en/topics/devops/what-is-devsecops?cicd=32h281b
- https://csrc.nist.gov/projects/devsecops
- https://www.intel.com/content/www/us/en/cloud-computing/devops.html
- https://www.geeksforgeeks.org/what-is-jenkins/
- https://developer.hashicorp.com/terraform/intro
- https://grafana.com/