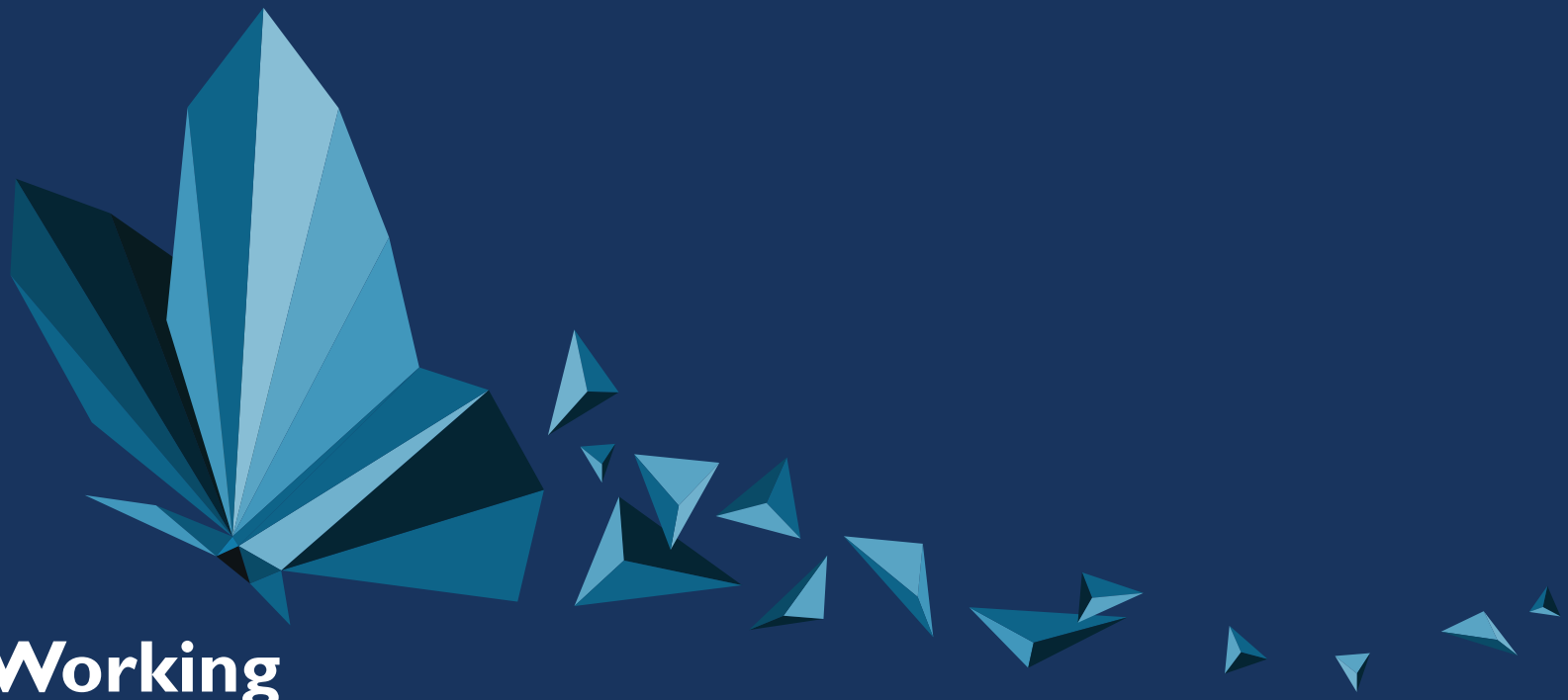


DevOps Introduction: Working with CI CD and Continuous monitoring Tools

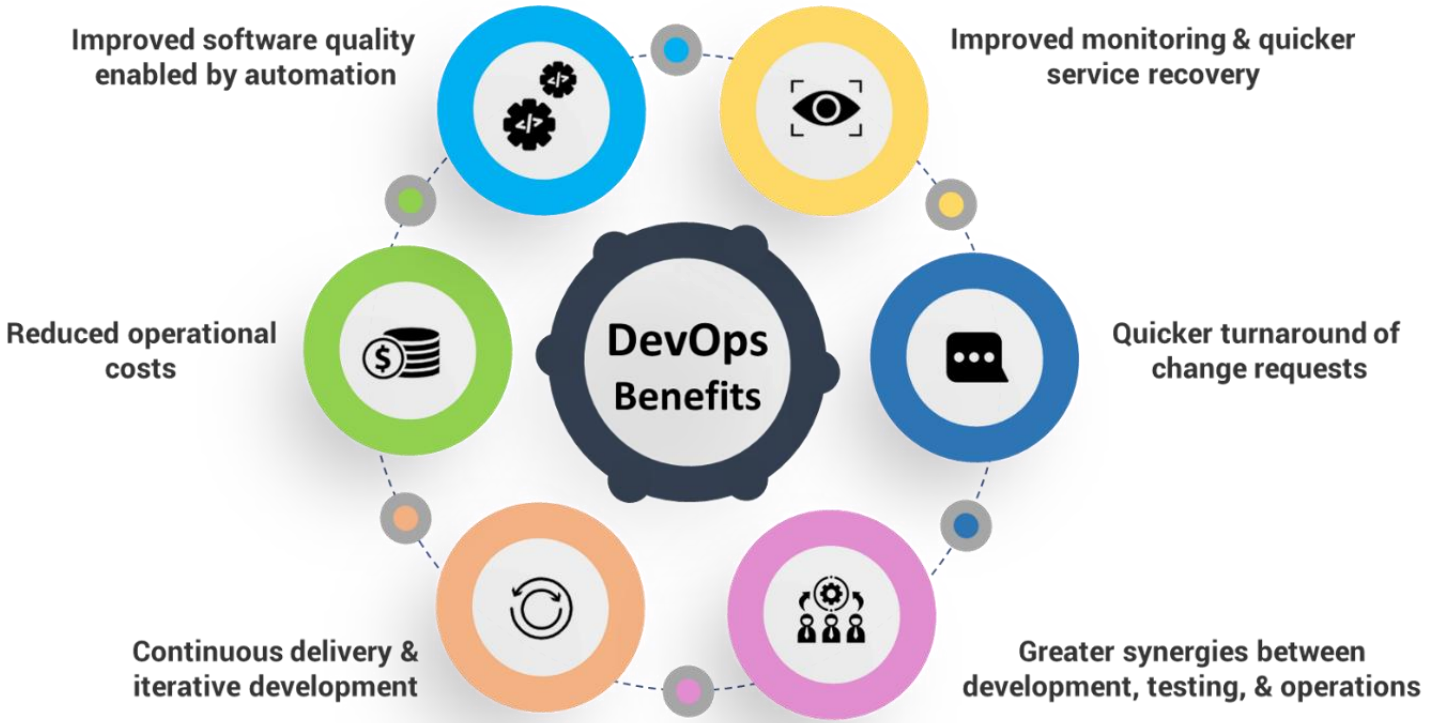
Version 1.0

Module 1: Introduction to DevOps Principles and Practices

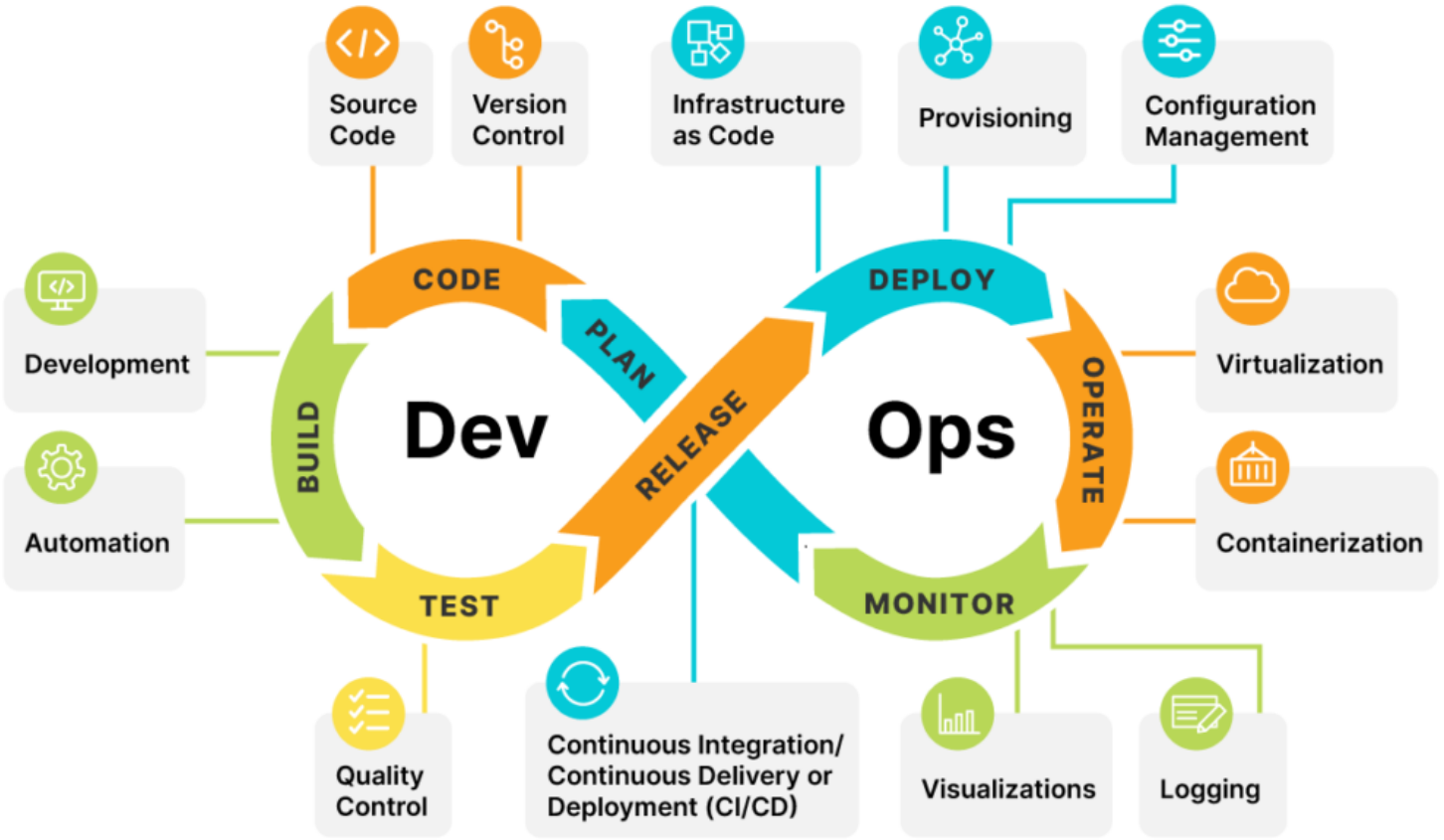
What is DevOps?

DevOps, short for Development and Operations, is a set of practices and cultural philosophies. That aims to bridge the gap between software development and IT Operation. DevOps brings together development, testing, deployment, and operations under one umbrella. That enables organizations to achieve continuous integration and continuous delivery (CI/CD) and maintain a rapid development pace.

Why DevOps? – Delivery challenges



DevOps Lifecycle



DevOps Lifecycle

- **Plan:** During this phase, teams define project goals, gather requirements, and create a roadmap for development. Key activities in the planning phase contain requirement gathering, scope definition, project estimation, and establishing project milestones and timelines.
- **Code:** The developers will write the code and prepare it for the next phase during the coding stage. Developers will write code in accordance with the specifications outlined in the planning phase and will ensure that the code is created with the project's operations in mind.

DevOps Lifecycle

- **Build:** Code will be introduced to the project during the construction phase, and if necessary, the project will be rebuilt to accommodate the new code. This can be accomplished in a variety of ways, although GitHub or a comparable version control site is frequently used.
- **Release:** The release phase occurs when the code has been verified as ready for deployment and a last check for production readiness has been performed. The project will subsequently enter the deployment phase if it satisfies all requirements and has been thoroughly inspected for bugs and other problems.

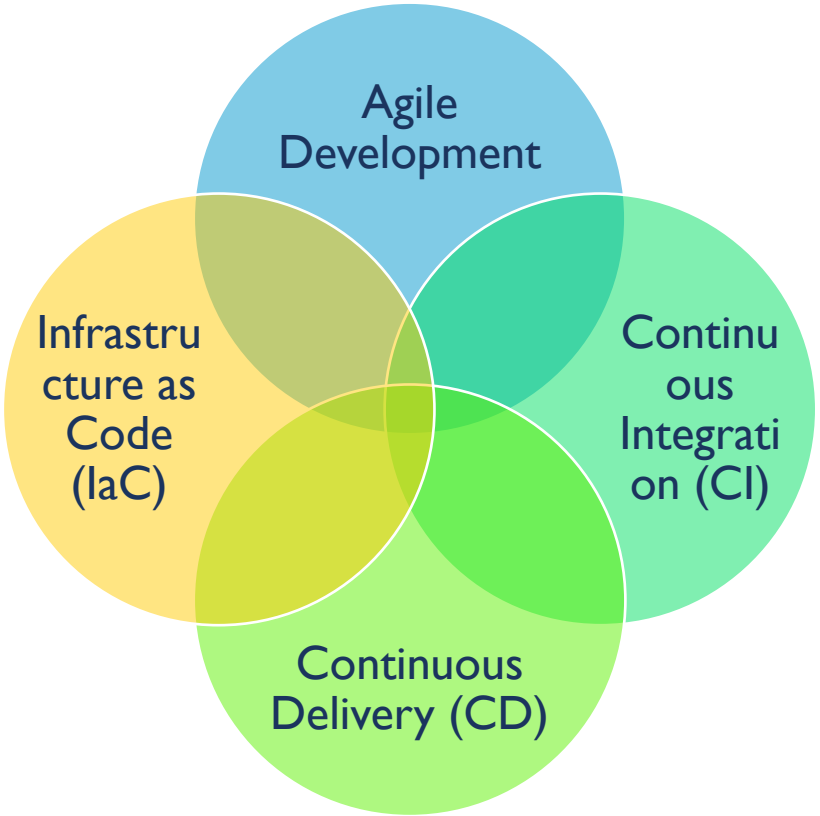
DevOps Lifecycle

- **Test:** The testing phase is essential to ensure the quality and functionality of the software being developed. Rigorous testing is performed to identify and rectify any bugs, errors, or issues. This phase includes various testing types such as unit testing, integration testing, and system testing.
- **Deploy:** The deployment phase involves releasing the software to the production environment for end-users to access and utilize. During this phase, deployment processes are automated to ensure consistency and reliability. Key activities in the deployment phase include environment setup, release coordination, deployment automation, and monitoring the deployment process.

DevOps Lifecycle

- **Operate:** The operation phase focuses on the ongoing management and maintenance of the software in the production environment. This phase involves monitoring the software's performance, addressing user feedback, and handling any issues or incidents that arise. Continuous monitoring and feedback loops are established to ensure optimal software performance and user satisfaction.
- **Monitor:** During the monitoring phase, product usage, as well as any feedback, issues, or possibilities for improvement, are recognized and documented.

Key Concepts of DevOps



Key Concepts of DevOps

Agile Development

- Agile Development has become a cornerstone of successful software delivery. That enables organizations to respond quickly to changing market needs.
- Agile Development focuses on iterative and incremental development. By embracing agile principles, organizations experience enhanced flexibility, improved transparency, and increased customer satisfaction.

Key Concepts of DevOps

Continuous Integration (CI)

- Continuous Integration (CI) is a vital DevOps practice that aims to merge code changes from multiple developers into shared storage regularly. CI promotes early and frequent code integration. Even automated testing and immediate feedback increase their popularity.
- By implementing CI, organizations reduce integration issues. With CI detecting defects earlier in the development process becomes easy. That improves collaboration among development teams.

Key Concepts of DevOps

Continuous Delivery (CD)

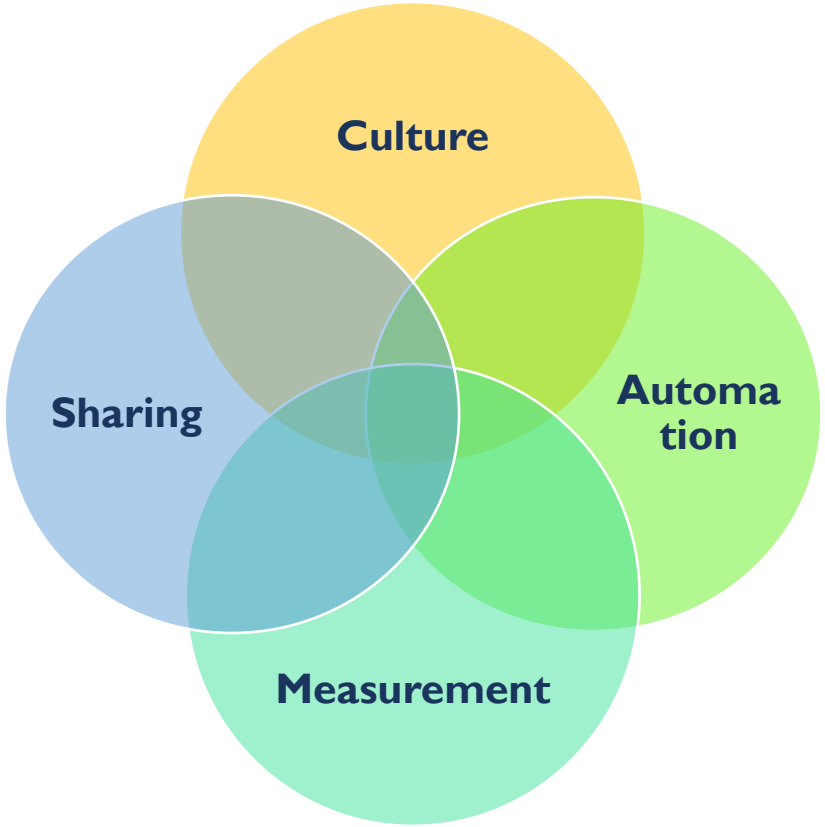
Continuous Delivery (CD) goes hand in hand with CI, enabling organizations to deliver software in a reliable and automated manner. The CD focuses on automating the entire release process, from building and testing to deployment and production. With CD, organizations gain the ability to release software updates swiftly. With minimal manual intervention, and maintain a continuous flow of valuable features to end-users.

Key Concepts of DevOps

Infrastructure as Code (IaC)

A revolutionary concept in DevOps is Infrastructure as Code (IaC). IaC allows organizations to automate infrastructure setup and management, enabling reproducibility, scalability, and consistency across environments. While leveraging IaC tools and practices, organizations gain better control over their infrastructure, reduce human error, and achieve greater efficiency in software deployment.

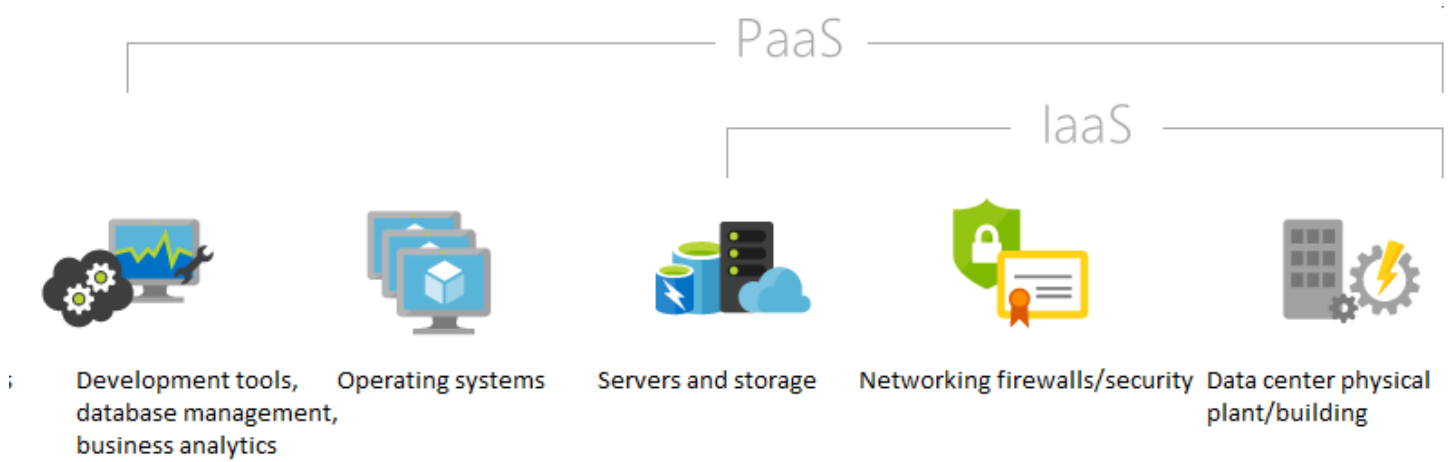
Understanding the pillars of DevOps



When to Adopt

- For eCommerce and other web sites projects (amazon,flickr,groupon etc)

- Cloud platform (IaaS and PaaS)



CI/CD - Introduction

CI/CD - Introduction

- CI/CD is a method that allows DevOps teams to deliver code updates frequently, reliably, and quickly using continuous integration (CI) and continuous delivery CD) practices. CI/CD emphasizes automation throughout the development lifecycle (Buid, Test, Deploy). By replacing the manual efforts of traditional development, code releases can happen more frequently, and with less bugs and security vulnerabilities.

What Is CI (Continuous Integration)?

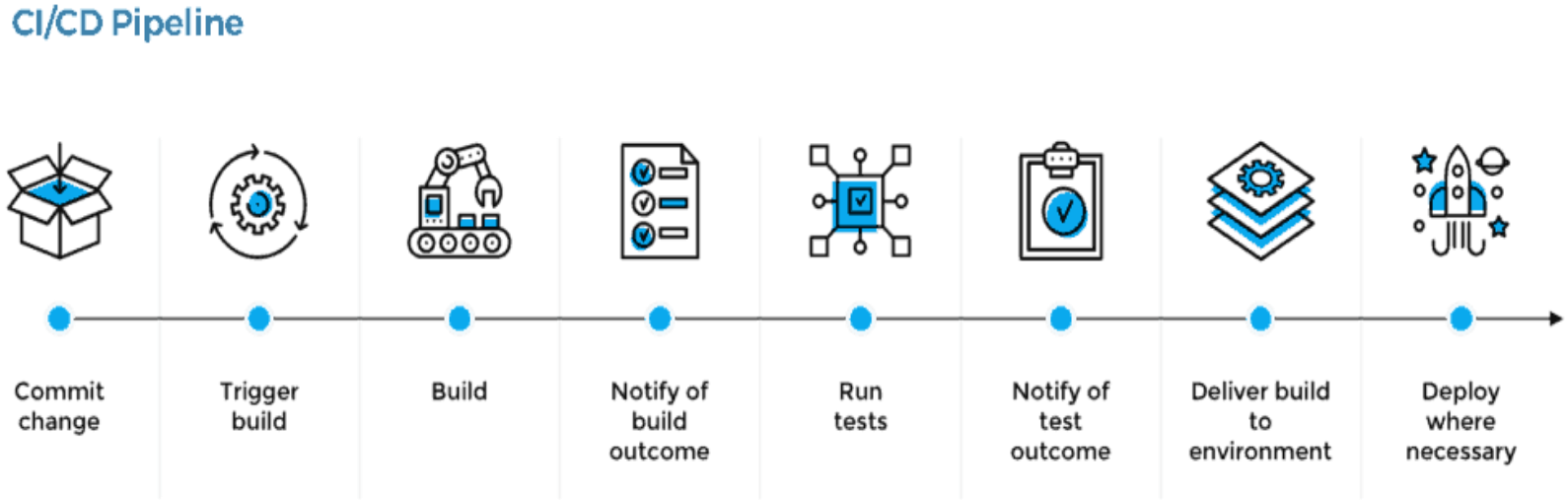
- Continuous integration integrates code changes into a shared repository of source code, and it usually automatically tests these changes as soon as they are committed. Continuous integration is performed as your developers write code, rather than afterward.

What Is CD (Continuous Delivery)?

- Continuous delivery is an extension of continuous integration in which your team automatically deploys new code to a repository (such as GitHub). It can then be deployed to production at any time based on your needs and the needs of your clients. The added automation used by continuous delivery greatly reduces the amount of effort required to deploy code. With continuous delivery, you can respond more quickly to anything from changes in the market to security flaws.

The CI/CD Pipeline

The CI/CD pipeline is the set of processes that drive your preferred combination of continuous integration, delivery and deployment, and it forms the base of your DevOps team's operations. In general, the head of your DevOps team is the one in charge of ensuring that the pipeline is working properly.



Module 2:Continuous Integration and Continuous Deployment (CI/CD) Concepts

ntuc LearningHub

Continuous Deployment (CI/CD)

Continuous Deployment (CI/CD)

- CI/CD is a method that allows DevOps teams to deliver code updates frequently, reliably, and quickly using continuous integration (CI) and continuous delivery CD) practices. CI/CD emphasizes automation throughout the development lifecycle (Buid, Test, Deploy). By replacing the manual efforts of traditional development, code releases can happen more frequently, and with less bugs and security vulnerabilities.

What Is CI (Continuous Integration)?

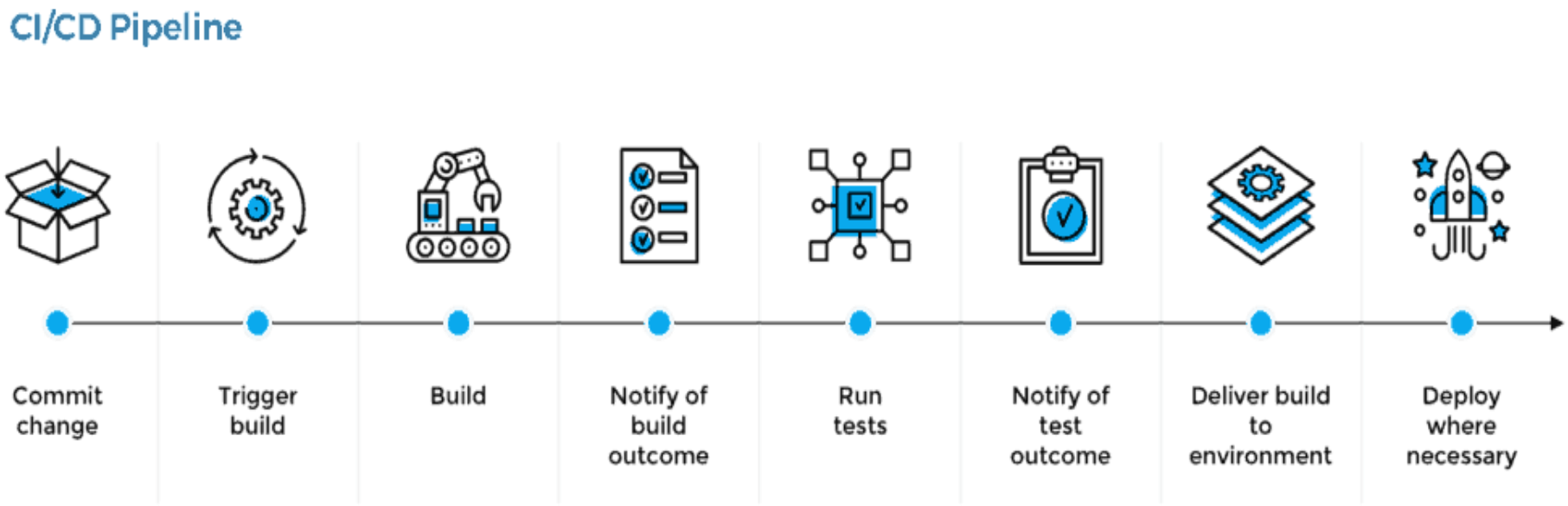
- Continuous integration integrates code changes into a shared repository of source code, and it usually automatically tests these changes as soon as they are committed. Continuous integration is performed as your developers write code, rather than afterward.

What Is CD (Continuous Delivery)?

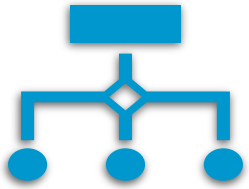
- Continuous delivery is an extension of continuous integration in which your team automatically deploys new code to a repository (such as GitHub). It can then be deployed to production at any time based on your needs and the needs of your clients. The added automation used by continuous delivery greatly reduces the amount of effort required to deploy code. With continuous delivery, you can respond more quickly to anything from changes in the market to security flaws.

The CI/CD Pipeline

- The CI/CD pipeline is the set of processes that drive your preferred combination of continuous integration, delivery and deployment, and it forms the base of your DevOps team's operations. In general, the head of your DevOps team is the one in charge of ensuring that the pipeline is working properly.

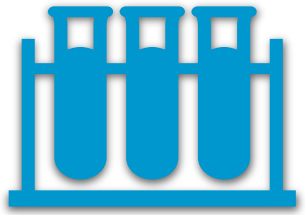


Components of a CI/CD Pipeline



Build:

In this stage, you pull your source code from a repository and build its components into a binary artifact. Your chosen integrated development environment (IDE) may help your developers automate this process.



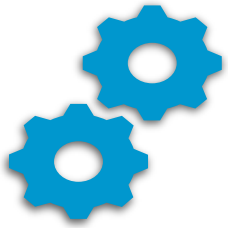
Test:

With the CI/CD pipeline, you want to employ as much continuous testing as possible. Unit Testing helps to verify that new features are working as intended, and most of your testing should be this type. Regression testing makes sure that new additions to your code won't break your existing infrastructure.

Components of a CI/CD Pipeline

Delivery:

After your tests are done, your developers' code should go to a staging environment. This enables you to perform A/B tests and find lingering problems, as well as letting your QA team know what they need to look at.



Deploy:

Once your build has passed all automated testing, it can be deployed in production. With continuous delivery, the build is sent to a human for manual approval, while with continuous deployment the build is deployed automatically.

Time for Practical Implementation

Module 3: Setting Up CI/CD Pipelines

ntuc LearningHub

Automation testing

- By automating software testing, organizations can remove redundancy, create a more unified approach among teams, and facilitate more efficient development.

What is test automation?

- Software testing is a vital process in the development of software. However, manual testing processes create considerable difficulty in collaborating and communicating feedback for DevOps and quality assurance (QA) teams, resulting in slower release cycles. Test automation, or automated QA testing, involves automatically reviewing, validating, and analyzing a software product and using the results from these assessments to improve software quality — ensuring more consistent and unified code and optimizing product functionality and user experience. Plus, it makes for happier developers.

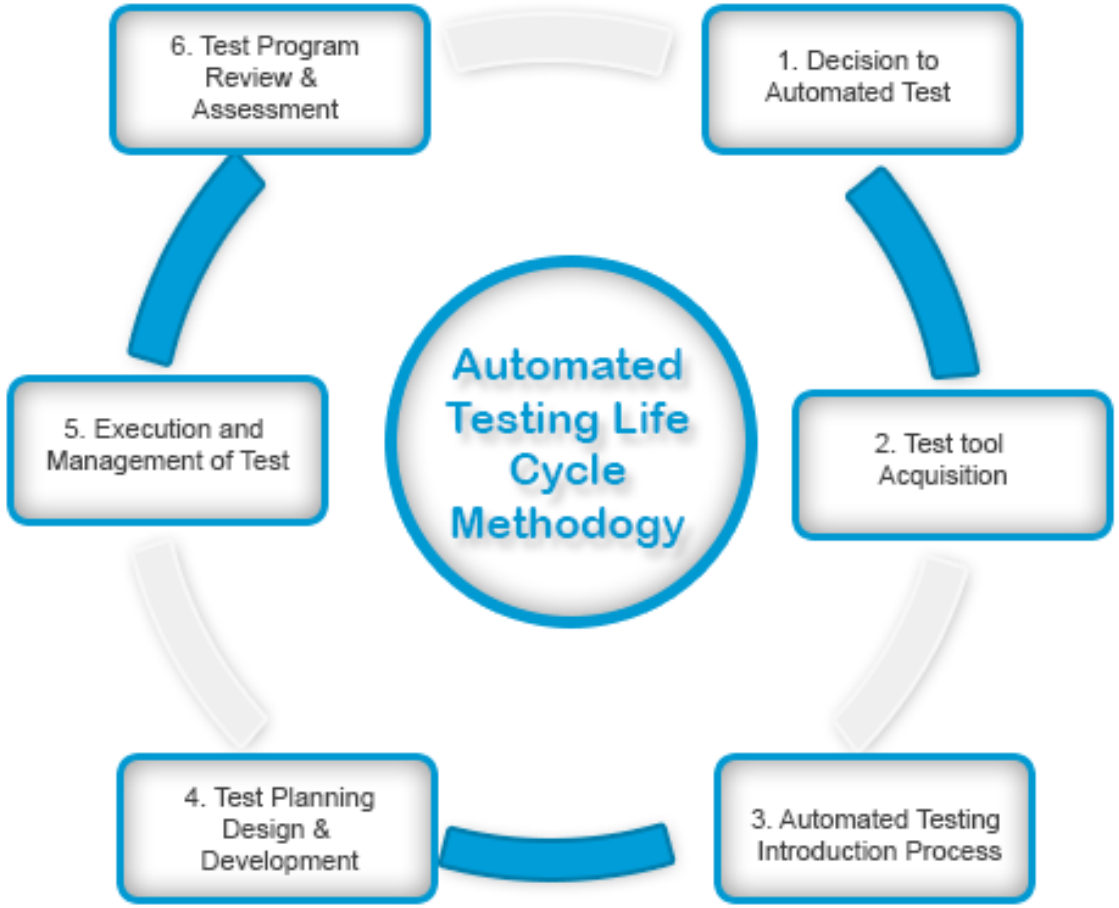
Automated testing in DevOps

- DevOps involves the software development workflows that accelerate the build, test, configuration, deployment, and release of software products. This approach helps teams build applications much faster. Because continuous testing is an integral part of continuous integration and continuous delivery (CI/CD) practices, adopting automated testing makes CI/CD more efficient and enables teams to release software more frequently.

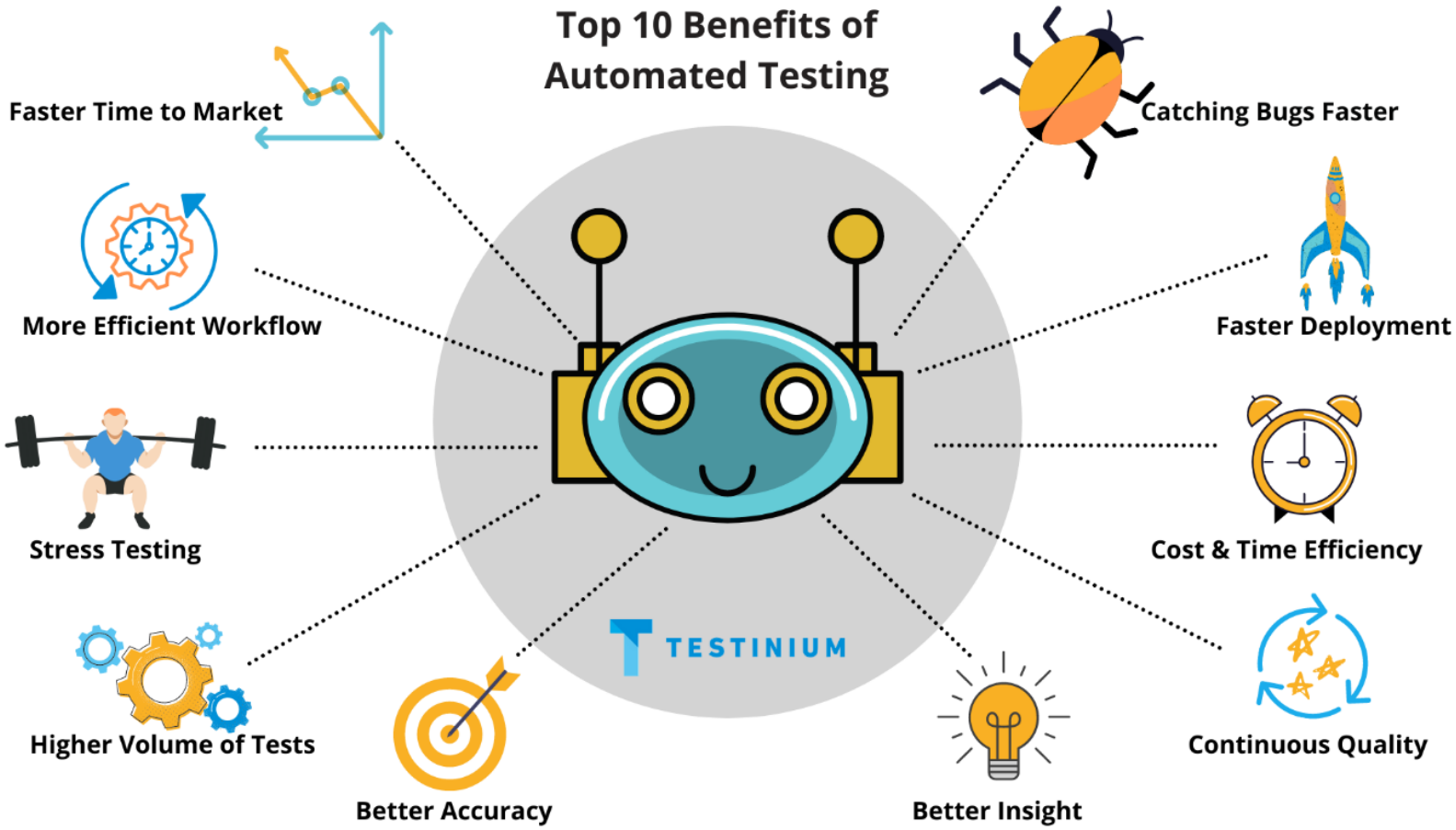
Automated testing in DevOps

- Quality assurance engineers should focus on developing automated integration and end-to-end tests while developers perform unit tests for each block of code they build. These tests should be executed early enough in the DevOps CI/CD pipeline to ensure each component works as expected. Additionally, product managers should perform functional testing (e.g., the black-box method) to ensure the optimal user experience

Test Automation Stage(Lifecycle)



Benefits of Automated Testing



Reference Link: Git Pipeline

- https://docs.gitlab.com/ee/ci/environments/
- **https://docs.gitlab.com/ee/ci/pipelines/pipeline_architectures.html**

Time for Practical Implementation

Module 4: Continuous Monitoring Tools and Practices

Contiguous Monitoring

Contiguous Monitoring

- DevOps is identifying threats to the security and compliance rules of a software development cycle and architecture. Also known as continuous control monitoring or CCM, this automated procedure can be extended to detect similar inconsistencies in IT infrastructures.

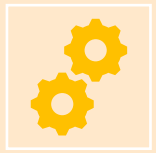
Importance of Continuous Monitoring

Many components of software operations can trigger devastating outcomes like breaches. Continuous monitoring aims to strengthen the transparency of such environments while keeping in place a vigilant system to monitor and resolve said issues.

Continuous monitoring aims to identify performance inconsistencies and error sources. It also resolves these problems using relevant solutions to safeguard the enterprise.

Continuous monitoring assists companies in keeping a tab of their user experience. CM is especially helpful in tracking user feedback after a recent change or update to a software or an application. This helps firms build and strengthen their business strategies.

Monitoring Tools

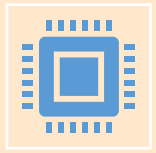


DevOps tools for Infrastructure Monitoring

- Nagios
- Prometheus

DevOps Tools for Application Monitoring

- Datadog
- Splunk



DevOps Tools for Networking Monitoring

- Wireshark
- NMap- Network Mapper

Grafana Practical

- Please start with the **Grafana Practical** for reference you can use the Learner's Guide.

What is the ELK Stack?

- The ELK Stack began as a collection of three open-source products — Elasticsearch, Logstash, and Kibana — all developed, managed and maintained by Elastic. The introduction and subsequent addition of Beats turned the stack into a four-legged project.

elasticsearch

How to use the ELK Stack?

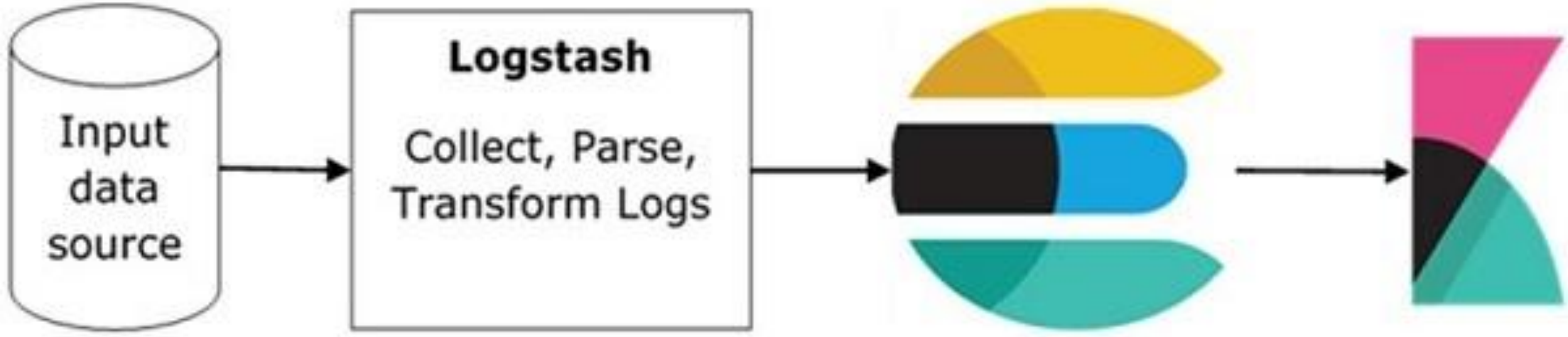
Time for Practical Implementation

Kibana

- Kibana is an open source browser based visualization tool mainly used to analyse large volume of logs in the form of line graph, bar graph, pie charts , heat maps, region maps, coordinate maps, gauge, goals, timelion etc. The visualization makes it easy to predict or to see the changes in trends of errors or other significant events of the input source.Kibana works in sync with Elasticsearch and Logstash which together forms the so called ELK stack.

What is Kibana?

- Kibana is a visualization tool, which accesses the logs from Elasticsearch and is able to display to the user in the form of line graph, bar graph, pie charts etc.

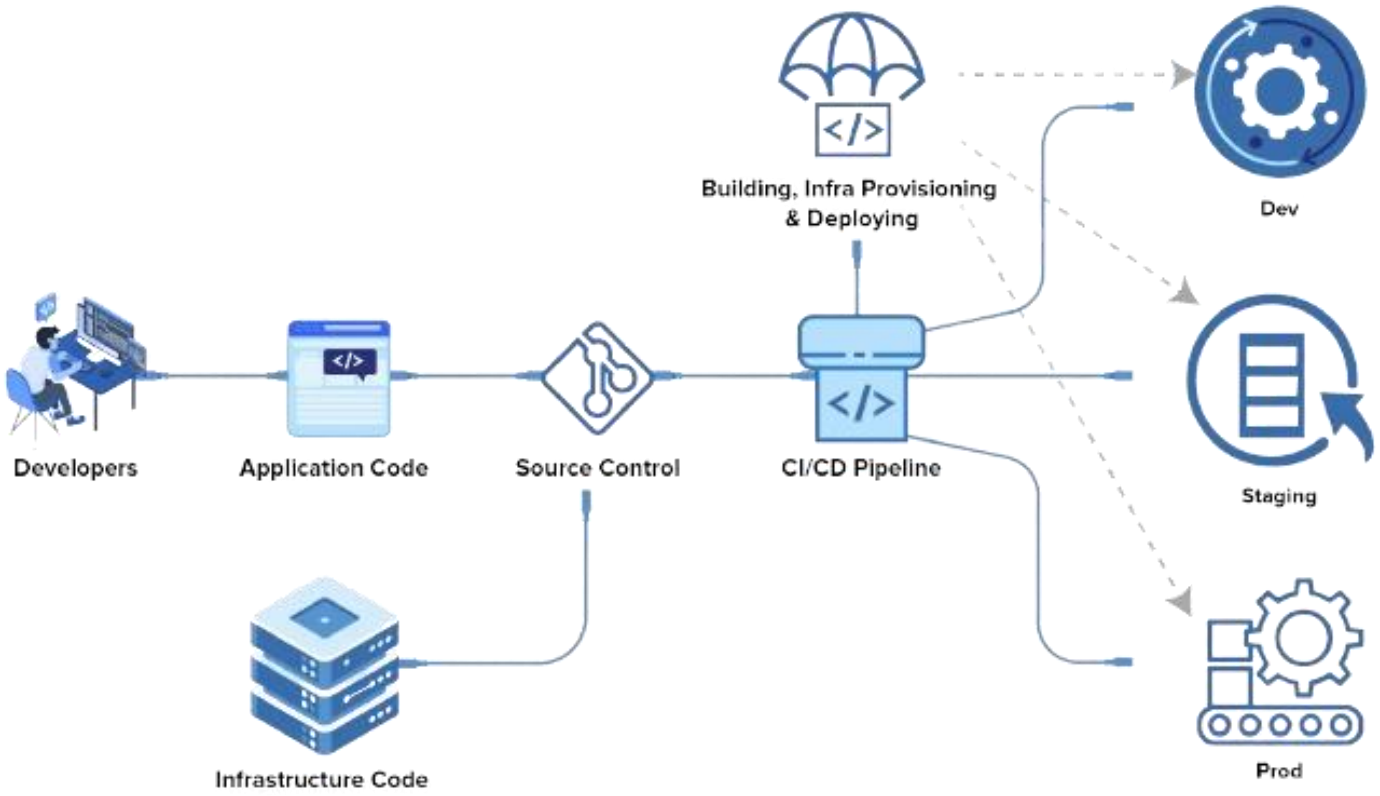


Time for Practical Implementation

Module 5: Automating Deployment with Infrastructure as Code (IaC)

ntuc LearningHub

Introduction to Infrastructure as Code (IaC) principles



Introduction to Infrastructure as Code (IaC) principles

- The Infrastructure as Code (IaC) operations have modified how software developers develop, evaluate, and release applications by increasing the number of development and distribution cycles. To make infrastructure development and configuration more competitive and successful, reducing the costs and effort involved, automation tools that facilitate these activities are essential.

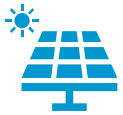
Principles of Infrastructure as Code (IAC)

Easily recreate systems

Disposable Systems

Self-Documentation

Design Is Always Changing



Generic Modules

A single, unified API for automated infrastructure deployment

https://www.xenonstack.com/insights/infrastructure-code-principles

Features of IaC

- Automation: IAC automates the provisioning and configuration of infrastructure, reducing manual errors and saving time.
- Repeatability: IAC scripts can be used repeatedly, making it easy to recreate the same infrastructure in multiple environments.
- Version Control: IAC code is stored in version control systems like Git, which makes it easy to track changes, revert to previous versions, and collaborate with others.
- Scalability: IAC makes it easy to scale infrastructure up or down, adding or removing resources as needed.
- Transparency: IAC makes the infrastructure transparent and understandable, as the code defines the infrastructure components and their relationships.
- Improved Security: IAC helps ensure that infrastructure is configured consistently and securely, reducing the risk of security vulnerabilities

Applications of IaC

Cloud computing

DevOps

Continuous integration and delivery (CI/CD)

Networking

Web application deployment

Database deployment

Big data

Git and Github

Git-Introduction

- Git is a popular version control system.
  It was created by Linus Torvalds in 2005
  and has been maintained by Junio
  Hamano since then.

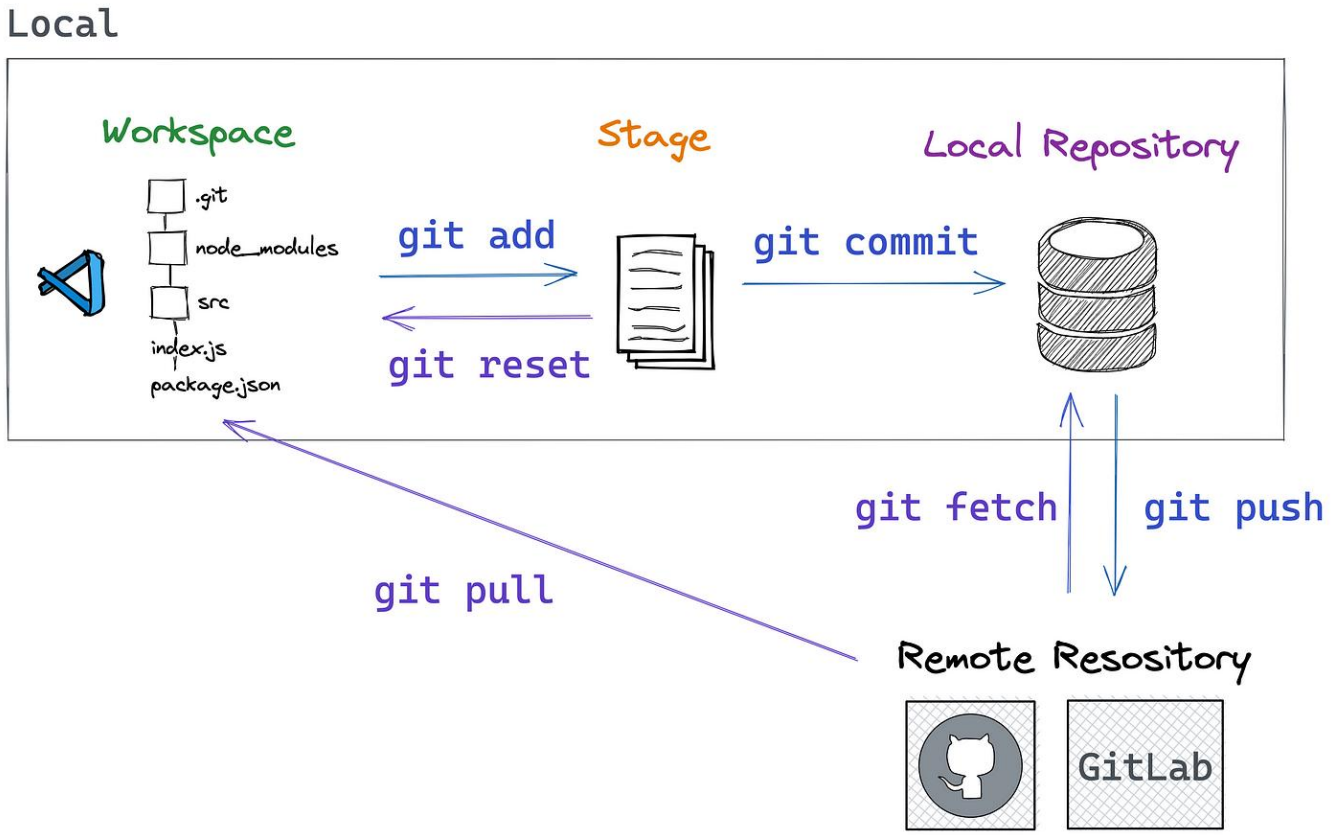
It is used for:

- Tracking code changes
- Tracking who made changes
- Coding collaboration

git

What does Git do?

- Manage projects with **Repositories**
- **Clone** a project to work on a local copy
- Control and track changes with **Staging** and **Committing**
- **Branch** and **Merge** to allow for work on different parts and versions of a project
- **Pull** the latest version of the project to a local copy
- **Push** local updates to the main project



What is GitHub?

- Git is not the same as GitHub.
- GitHub makes tools that use Git.
- GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018.
- In this tutorial, we will focus on using Git with GitHub.

Time for Practical Implementation

Module 6: Security and Compliance in CI/CD Pipelines

ntuc LearningHub

CI/CD Pipeline

- A continuous integration and continuous delivery/deployment (CI/CD) pipeline is a series of steps that software delivery undergoes from code creation to deployment.
- CI/CD encompasses a series of automated processes — from code development to production deployment — that enable frequent and reliable delivery of code changes to the production environment. It forms the backbone of DevOps, a shift in software development that emphasizes collaboration between development and operations teams to ultimately shorten the development lifecycle without compromising software quality.

Integrating security practices into CI/CD pipelines

- it's important to include security checks at various stages to ensure that your code is secure and compliant with security standards. There are a number of measures you can take to secure your CI/CD pipeline.

Planning phase

- This stage involves gathering requirements and consumer input to develop a product roadmap. It also encompasses the best practices and policies for a successful DevOps strategy.
- You should also take advantage of threat modeling to help identify potential areas of attack and take steps to secure your pipeline. In threat modeling, security vulnerabilities are identified, and countermeasures are determined to mitigate them. By applying threat modeling to CI/CD pipelines, you can identify potential attack areas and take measures to secure them.
- Supply-chain Levels for Software Artifacts (SLSA) is also useful during the planning phase. This security framework comprises a checklist of standards and controls to prevent supply-chain attacks, safeguard against integrity challenges, and safeguard software packages and infrastructure in your organization.

Coding phase

- In the coding phase, the developers write the necessary code to build the software. The code must be written by predefined standards and design guidelines.
- You should use source code scanners such as CAST Application Intelligence Platform (AIP) or CodeSecure to detect pieces of code that might be vulnerable to security threats.

Build phase

- During the build phase, the developers are responsible for committing their source code to a shared repository. Once the code changes are checked into the repository, builds are triggered, and automated tests are executed to verify if the builds comply with the requirements.
- Here are some tips for setting up security checks in your CI/CD pipeline:
- Include a static code analysis tool in your build stage to check the code for common security vulnerabilities and compliance issues.

Build phase

- Use Static application security testing (SAST) tools like SonarQube, Veracode, AppScan, or Codacy.
- Use software composition analysis (SCA) tools such as Veracode, Sonatype Nexus Platform, etc.
- Set up security-related test cases based on organization policies. These tests can check for things like cross-site scripting (XSS) and SQL injection flaws.
- Use a code-signing service to sign your code ahead of deploying the code to the production environment. This will help ensure that the code has not been tampered with and that it comes from a trusted source.

Testing phase

- Once a build is successful, the software is tested to detect any potential bugs. If new features are added, a new build is generated and regression testing is performed on the new build to verify if the functional tests succeed.
- At this stage, you should run container scanning tools (e.g., Datadog, Clair, Anchore, and Qualys) or dynamic analysis security testing (DAST) tools (e.g., Netsparker and Acunetix)

Deployment phase

- In this phase, the build is deployed to the production environment.

Monitoring Phase

- During this phase, the build is monitored to ensure that it works as expected. The application deployed in the production environment is observed to evaluate performance and other aspects.

What is Vulnerability Scanning?

- Vulnerability scanning is performed to detect and remediate these vulnerabilities.
- Vulnerability scanning can be done either by the team or by automated software to manage different types of vulnerabilities. Automated vulnerability scanning is different from manual vulnerability scanning, in which a human examines an application or system and searches for vulnerabilities.

CI/CD Secrets Management

- A continuous workflow calls for unique practices when it comes to managing vault secrets and privileged access. Secrets refer to user credentials, keys, passwords, private certificates, and anything else that is essential for software deployment and must be kept secret.
- Secrets can serve as authentication tools to prove the identity of a user or system or authorization, where access to certain corporate resources is granted based on user privileges.
- When working with CI/CD, organizations must use multiple services, environment repositories, cloud providers, and verification providers. Keeping accurate secrets for each team and each service can be a challenge when everyone needs a different level of access.

CI/CD Secrets Management

- Also, part of development is authenticating digital tools to get the most out of them. Any resources you use must have the right access to do their jobs, especially in hybrid cloud environments or automated tools like Kubernetes.
- All this complexity makes secrets management in a CI/CD environment expensive and error-prone if managed improperly. Because secrets are required for any CI/CD working environment, it's important to pick up some best practices.

Module -7
Hands-on Workshop: Building and Managing CI/CD Pipelines

ntuc LearningHub

Applying best practices for testing, deployment, and monitoring

Applying best practices for testing, deployment, and monitoring

- **Prioritize security, testing, and time to release:**

  Before configuring your CI/CD pipeline, prioritize security, testing, and time to release. This determines how you approach each step in your pipeline. For example, if security is a top priority, you can implement more controls on your code before it's deployed.

Applying best practices for testing, deployment, and monitoring

- **Test code in the early stages:**

  You can catch bugs and identify issues at different testing stages. For example, if developers find a problem on their local machine, they can fix issues in the staging environment before moving the code to production.

Applying best practices for testing, deployment, and monitoring

- **Identify tests that can be automated:**

  Any test that doesn't require human interaction should be automated and integrated into the pipeline as a rule of thumb. You should adopt an "automate everything" mindset but also understand not all tests can be automated.

Applying best practices for testing, deployment, and monitoring

- **Commit daily:**

  High-performing teams can commit to their repository once a day. This requires extensive automation to ensure high quality with each new version, but it's well worth the effort.

Applying best practices for testing, deployment, and monitoring

- **Choose tools that support your priorities:**

  When choosing DevOps tools, consider what matters most to you as a developer and business analyst. Do you heavily focus on infrastructure automation? Does an application lifecycle management (ALM) methodology drive your work? Do you manage multiple virtual machines? Consider vendor options designed explicitly for CI/CD operations to address these fundamental concerns.

Applying best practices for testing, deployment, and monitoring

- **See if you're ready to adopt a microservices architecture:**

  Microservices architecture is a way to structure software applications so that each component functions, updates, and scales independently.

Applying best practices for testing, deployment, and monitoring

- **Use on-demand testing environments:**

  The ease of creating a new testing environment at the click of a button helps run tests as early and often as possible. As a result, developers can make the most of their time knowing they'll complete their work quickly.

Applying best practices for testing, deployment, and monitoring

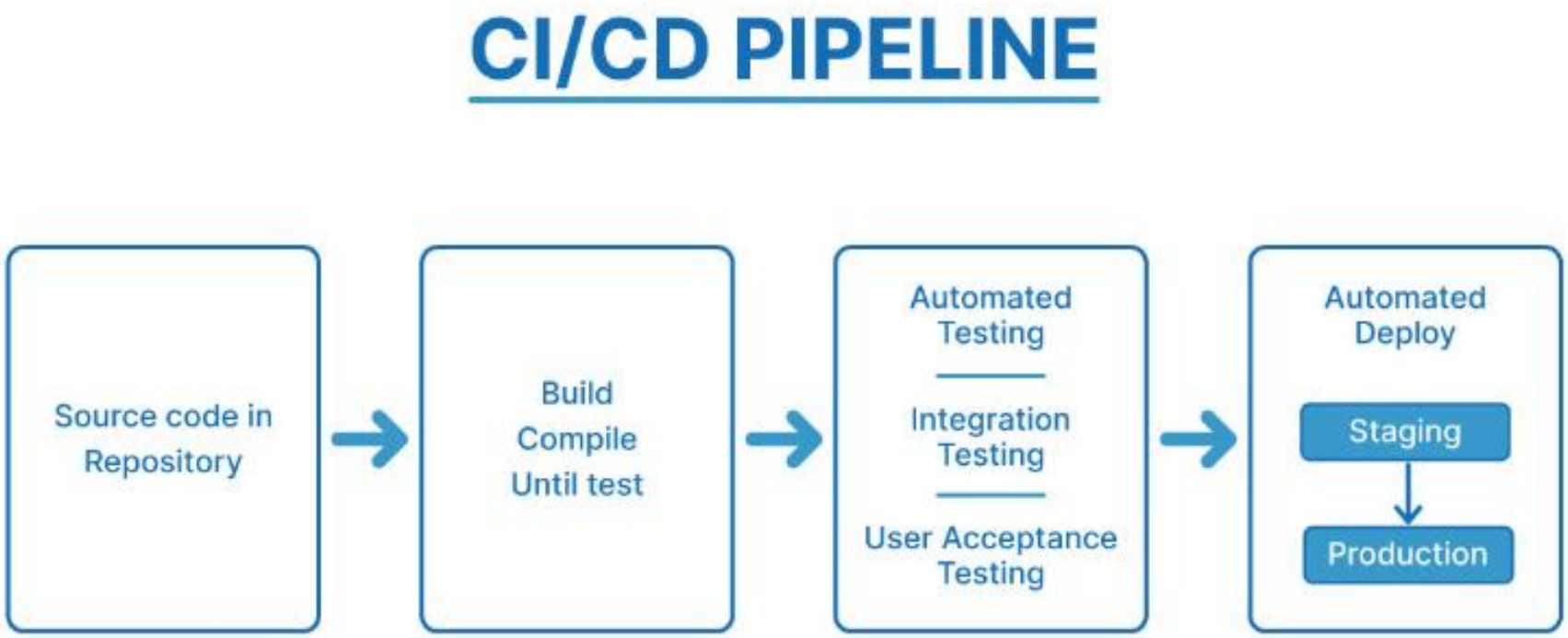
- **Involve the whole team in the CI/CD implementation:**

  Involving each team member in the CI/CD process is critical to CI/CD implementation. CI/CD isn't technical; it's cultural. Make it clear that CI/CD isn't just a technical step but an essential part of the company culture to seek active participation in the process.

Module-8
Use Cases: Successful CI/CD and Continuous Monitoring Implementations

ntuc LearningHub

Analyzing real-world use cases of successful CI/CD and monitoring implementations



CI/CD pipelines real-life example

- There are many big giant companies nowadays implementing DevOps. Netflix and Facebook are the prime examples that have successfully implemented CI/CD pipelines to deliver software updates and features quickly and with high quality. Netflix uses a custom-built platform called Spinnaker to manage its CI/CD pipelines. Spinnaker allows Netflix to deploy code changes to production environments in minutes rather than hours or days.

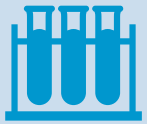
Case Study 1: Accelerating Software Delivery with Continuous Integration

- **Background**:
  - The case study begins with a well-established e-commerce company facing a common challenge in the digital era: the need to innovate and deliver new features and updates to its online store rapidly. In a highly competitive market, speed to market and agility were critical factors that could set them apart from their competitors. However, their existing software development process was plagued by slow manual testing and deployment procedures, leading to delays in delivering new features and bug fixes to customers.

The DevOps Solution: Continuous Integration and Continuous Deployment (CI/CD)

- To address these challenges, the company embraced DevOps principles, focusing initially on implementing Continuous Integration (CI) and Continuous Deployment (CD) pipelines. Here's how they did it:

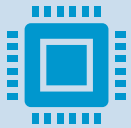
The DevOps Solution: Continuous Integration and Continuous Deployment (CI/CD)



Automated Testing: They automated their testing process, ensuring that every code change was automatically subjected to a battery of unit tests, integration tests, and regression tests. This allowed for quick identification of defects and issues, reducing the chances of bugs making their way into production.

Version Control: They adopted a robust version control system (like Git) to track code changes, enabling developers to collaborate seamlessly and ensuring that the codebase remained stable and maintainable.

The DevOps Solution: Continuous Integration and Continuous Deployment (CI/CD)



CI/CD Pipelines: CI/CD pipelines were implemented to automate the entire software delivery process. Developers would commit their code to the version control repository, triggering the CI/CD pipeline. The pipeline consisted of stages such as building the application, running automated tests, and deploying the application to a staging environment for further testing.

Deployment Automation: The CD part of the pipeline included automated deployment to production once the changes passed all tests in the staging environment. This automated deployment process eliminated the need for manual intervention, reducing the chances of human error.

Results: The implementation of CI/CD pipelines had a profound impact on the e-commerce company's software delivery process:

Faster Time to Market: The time required to deliver new features and updates was drastically reduced. What previously took weeks or even months could now be accomplished in a matter of days or hours.

Higher Quality Software: Automated testing and deployment significantly improved software quality. Bugs and issues were caught early in the development process, reducing post-release bug fixes.

Results: The implementation of CI/CD pipelines had a profound impact on the e-commerce company's software delivery process:

Enhanced Collaboration: The CI/CD pipeline encouraged collaboration among development and operations teams. The pipeline's transparency allowed everyone to see the progress of changes from development to production.

Improved Customer Satisfaction: With quicker releases and a reduced number of bugs, customer satisfaction soared. The e-commerce site provided a smoother, more reliable shopping experience.

Competitive Advantage: The company gained a competitive edge by being able to respond swiftly to market demands, seasonal changes, and emerging trends. They could roll out features in response to customer feedback more rapidly than ever before.

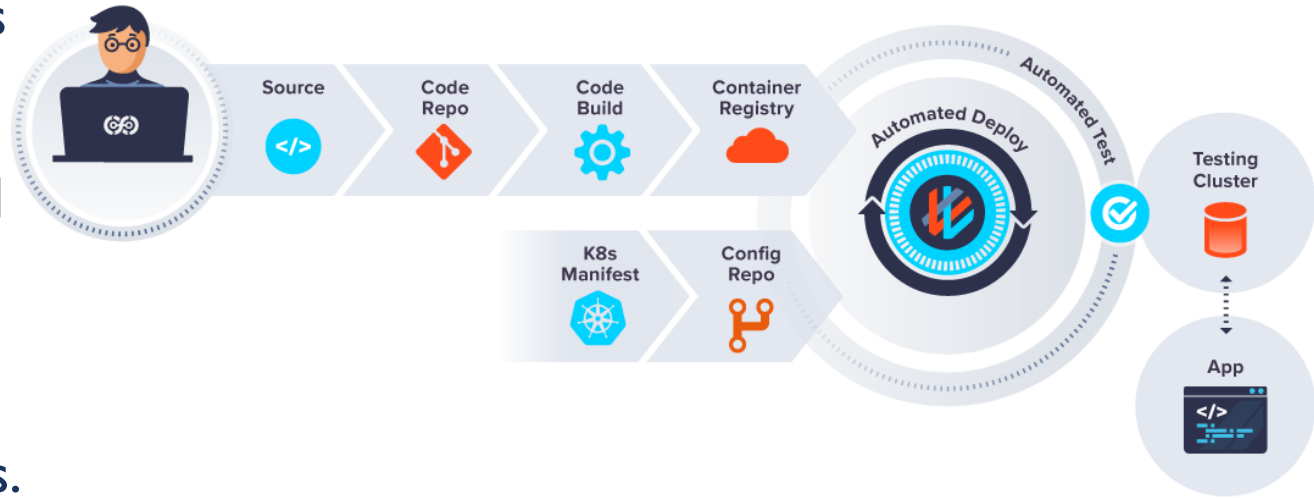
Conclusion:

- This case study illustrates the transformative power of Continuous Integration and Continuous Deployment (CI/CD) in the world of DevOps. By automating testing and deployment processes, the e-commerce company was able to accelerate software delivery, improve software quality, and stay ahead in a competitive market. It underscores the fact that CI/CD is not just a technology but a cultural shift that can revolutionize how software is developed, tested, and delivered. In the fast-paced digital age, organizations that embrace CI/CD are better positioned to meet customer expectations and thrive in the market

Module 9: Future Trends and Innovations in DevOps Practices

Exploring emerging trends in DevOps, including GitOps, NoOps, and AIOps

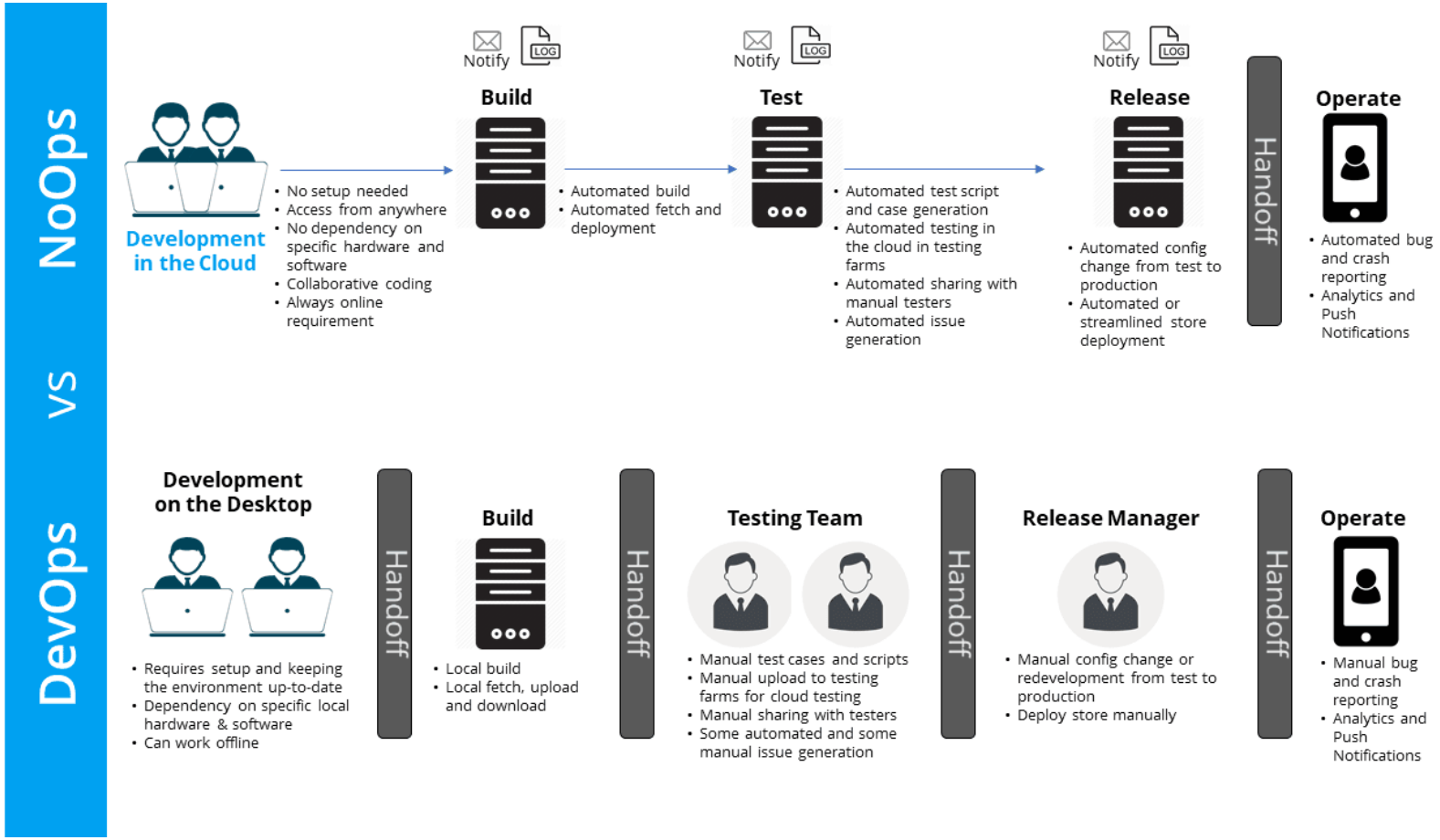
- **GitOps** is a standardized approach to deploying, configuring, monitoring, updating, and managing infrastructure as code. GitOps keeps documentation, code, and any other information related to deployment in a version control system like Git, and makes updates automatically using automated directors.



Exploring emerging trends in DevOps, including GitOps, NoOps, and AIOps

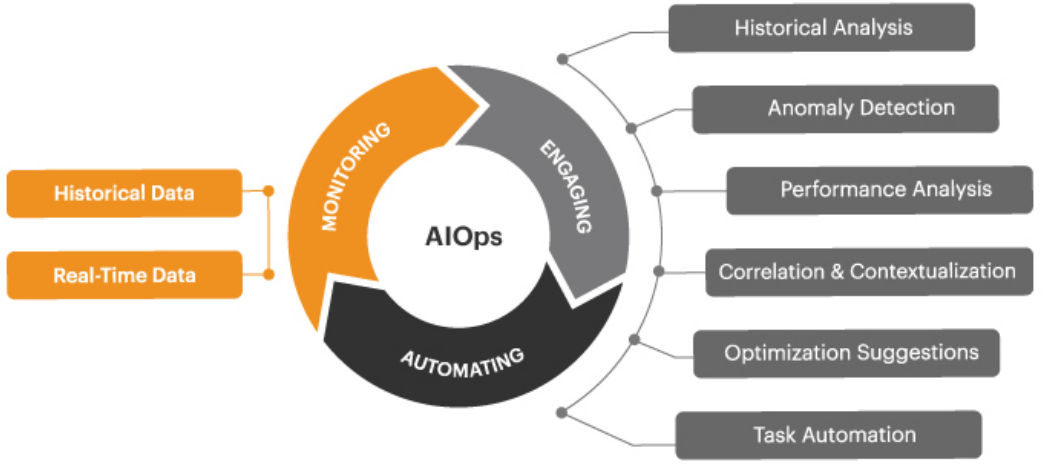
- **NoOps** is the concept that operations can be completely automated, with zero need for software management. NoOps follows a trend that has been in development for over a decade now.

DevOps vs NoOps

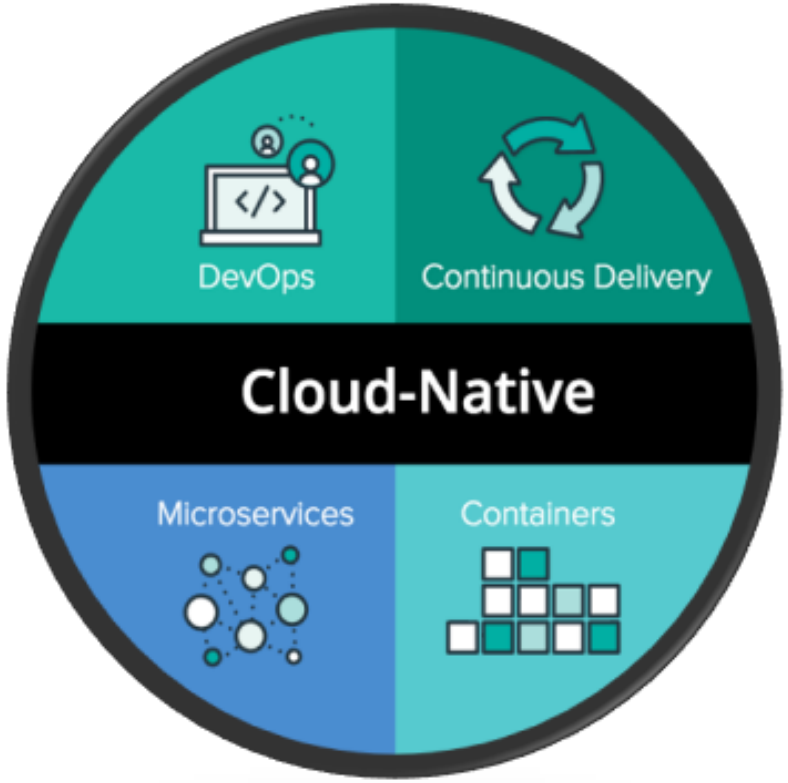


Exploring emerging trends in DevOps, including GitOps, NoOps, and AIOps

- **AIOps** stands for artificial intelligence for IT operations. AIOps is a model that recognizes that advances in big data and machine learning are the keys to building new approaches in the rapidly evolving IT environment. The concept refers to a set of technologies that utilize analytics and machine learning (ML) to enhance and automate IT operations.



Cloud-native application



Cloud-native application

- These applications are run and hosted in the cloud and are designed to capitalize on the inherent characteristics of a cloud computing software delivery model.

- Cloud-native applications use a microservice architecture. This architecture efficiently allocates resources to each service that the application uses, making the application flexible and adaptable to a cloud architecture.

For more course enquiries, visit us at:

Branches / Sales Centres

1. NTUC LHUB @ NTUC Trade Union House, 73 Bras Basah Road, #02-01, S189556
2. NTUC LHUB @ Devan Nair Institute, 80 Jurong East Street 21, #02-03, S609607
3. NTUC LHUB @ Lifelong Learning Institute, 11 Eunos Road 8, #05-01, S40860
4. NTUC LHUB @ Tampines Mall, 4 Tampines Central 5 (L4, outside Long John Silver's), S529510
5. NTUC LHUB @ AMK Hub, 53 Ang Mo Kio Ave 3, #B1-80, S569933
6. NTUC LHUB @ NorthPoint City, North Wing, #B1-39, 930 Yishun Ave 2, S769098
7. NTUC LHUB @ Causeway Point, 1 Woodlands Square, #03-K03, S738099
8. NTUC LHUB @ Compass One, 1 Sengkang Square, #04-23, S545078
9. NTUC LHUB @ Jurong Point 2, 1 Jurong West Central 2, #03-119, S648886

6336 5482

www.ntuclearninghub.com

Training Centres

1. NTUC LHUB Industry Skills Centre @ Benoi, 60 Benoi Road, #01-08, S629906
2. NTUC LHUB @ Aljunied Training Centre, 260 Sims Ave, ECM Building, #02-01, S387604
3. NTUC LHUB @ Tampines Plaza 2, Tampines Plaza, 5 Tampines Central 1, #04-02, S529541

ntuc LearningHub