POWERING
POTENTIAL

Introduction to DevOps

DEVOPS

CONTENTS

- What is DevOps?
- DevOps as a culture
- CI/CD
- Measurement
- DevOps tooling and infastructure

OBJECTIVES

INTRODUCTION TO DEVOPS

Understand DevOps and the CI/CD pipeline

› What is DevOps and why do we use it in software development?

Describe DevOps as a culture

› How is a DevOps team organised?

Explore the different tools used in DevOps

› What kind of tools are there and why do we use them?

POWERING POTENTIAL

What is DevOps?

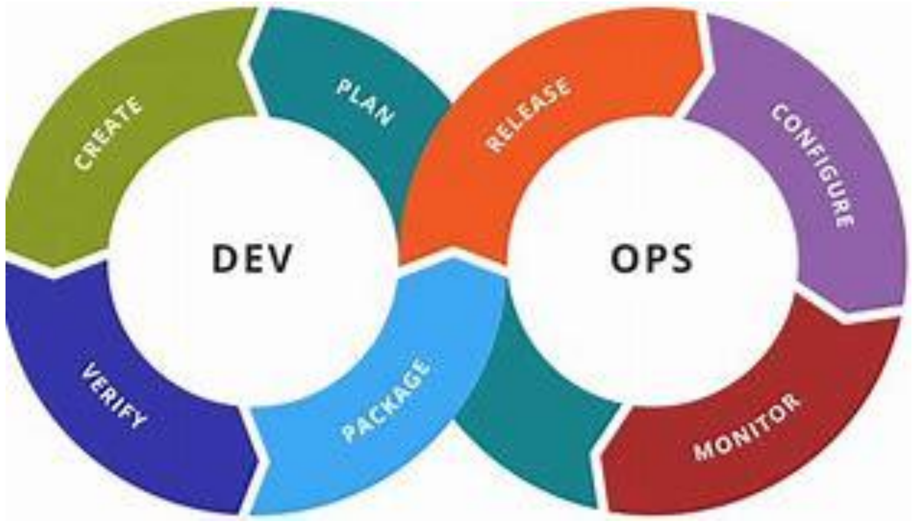
DevOps:

Introduction to DevOps

What is DevOps?

DevOps is best described as a cultural approach to software development with a particular philosophy designed to achieve the following:

- **Increased collaboration**
- **Shared responsibility**
- **Reduction in silos**
- **Autonomous teams**
- **Increase in quality**
- **Valuing feedback**
- **Increase in automation**



How Things Used to be Done:

Traditionally, software companies are structured as separate teams for development, quality assurance, security, and operations.

These teams tend to have varying and sometimes conflicting goals. When paired with poor communication this can result in work that is out of sync with other parts of the organisation. These isolated teams are referred to as silos.

As a result, this structure regularly results in slower releases, wasted time (and therefore money), and blame cultures, where production problems become the fault of another team.

POWERING
POTENTIAL

DevOps as a culture

DevOps as a culture– Module 3: DevOps

How DevOps Changes Things Up…

DevOps is based on Agile project management, an approach to projects that promotes:

- **Individuals and interactions over processes and tools**
- **Working software over comprehensive documentation**
- **Customer collaboration over contract negotiation**
- **Responding to change over following a plan**

Agile encourages flexible teamwork, celebrates achievements, promotes a productive work culture and bridge the gap between developers and customers.

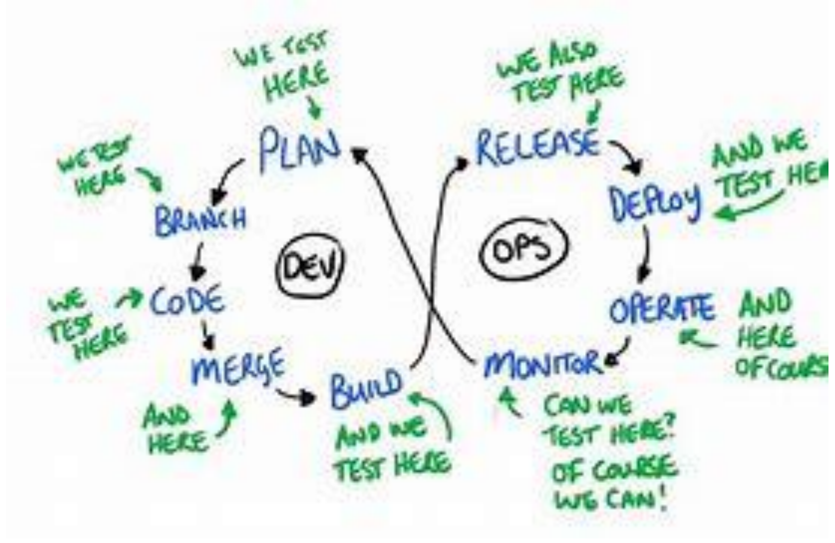
It bring together product owners, Devs and Testers to work together with the same goals in mind.

How DevOps Changes Things Up…

Within software companies, there has historically been friction between the developers and operations teams resulting in:

- **Slower release times**
- **Team-members unable to focus on core competencies**
- **General frustration within the organisation.**

Adoption of the DevOps methodology requires the dismantlement of silos. Devs and Ops are encouraged to break down their silos and start collaborating.

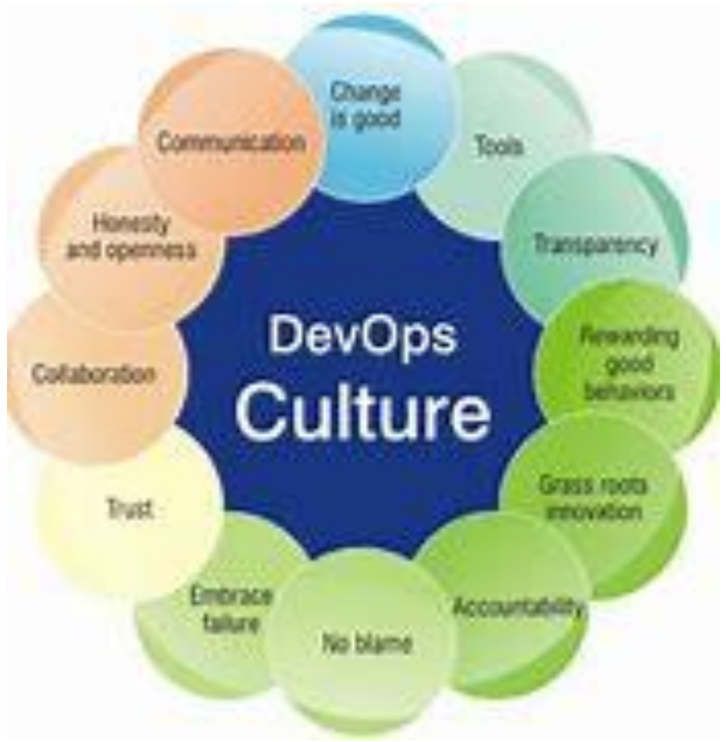


DevOps as a culture

A company culture refers to the shared values, behaviours and beliefs that characterize an organisation.

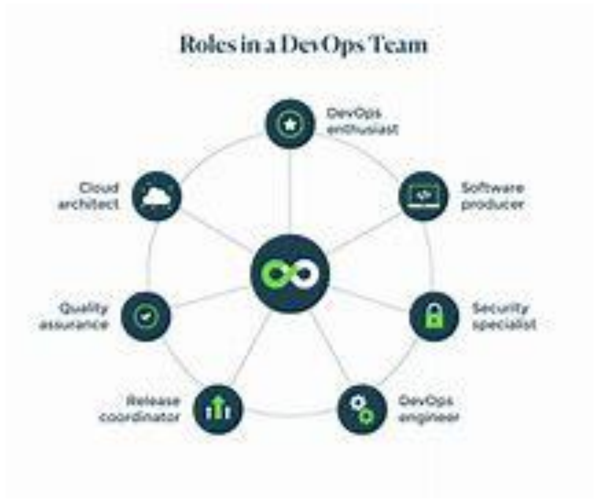
DevOps as a culture encompasses the following:

- **Collaboration and communication**
- **Honesty/openness and trust**
- **Open to change**
- **Embrace failure and 'no blame' culture**
- **Innovation**
- **Accountability and recognising achievements**



A DevOps team...

A DevOps team must be cross-functional and self-organising.



- **Teams should self-organise to tackle tasks**
- **A cross-functional team should have DevOps, developer, operations, security, UX, testing and cloud skills.**
- **Collaborative tools should be used**
- **Communication should be open and transparent.**
- **Shared responsibility between all members**
- **The team should have a unified objective**

CI/CD

DevOps:

Introduction to DevOps

CI/CD

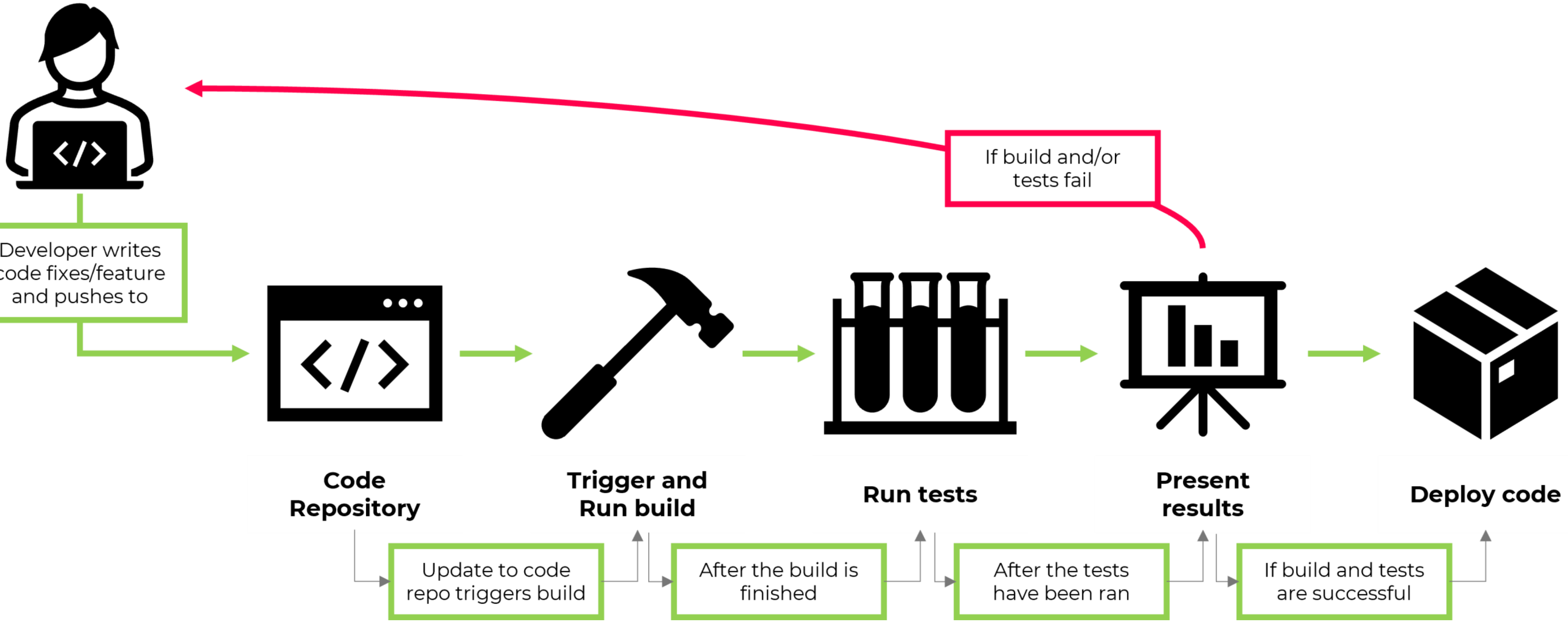
Continuous Integration (CI) is the automated integration of code from many contributors into a single software project.

Core to the strategy of Continuous Integration is the CI Pipeline. This allows developers to integrate newly-generated code easily and frequently.

A continuous integration pipeline should:

- Maintain a single source code repository for a project
- Automate build processes
- Automate testing of new builds
- Inform developers of test failures with detailed logs
- Encourage smaller, frequent deployments of code

CI/CD



Developer writes code fixes/feature and pushes to
Code Repository
Trigger and Run build
Run tests
Present results
Deploy code
If build and/or tests fail
Update to code repo triggers build
After the build is finished
After the tests have been ran
If build and tests are successful

CI/CD

Benefits:

- **Scaling**
- **Feedback Loop**
- **Communication**

Challenges:

- **Installation and adoption**
- **Learning curve**

Task: CI pipeline

Outcome:

Research different products that can be used at each stage of the CI Pipeline, to create your own CI Pipeline.

Steps:

Whilst choosing these tools, ensure you note down answers to the below questions:

Why have you chosen the product?

How much does using this product cost?

What are the alternatives to using this product?

CD

Continuous Deployment/Delivery (CD)

As new code passes acceptance tests, it is automatically integrated into a deployment environment.

Being able to choose a version to deploy with one push of a button requires a good amount of automation.

Two extensions of CI that are commonly employed are Continuous Delivery and Continuous Deployment

CD

Continuous Delivery:

Continuous Delivery requires an automated process in place, such that new releases only require approval via a click of a button to go live. This approach encourages organisations to deploy their applications as early as possible in smaller batches.

Continuous Deployment

Continuous deployment further extends the continuous delivery practice to ensure feature releases are entirely automated, requiring no human interaction for new application features to end up in the customer's hands.

Task: CD

Outcome:

Research 2 companies that use CD and list answers to the following questions:

Steps:

What benefit does working in this way bring the company?

What would be the alternative ways of working?

How does working like this affect the end user?

POWERING
POTENTIAL

Measurement

DevOps:

Measurement

Measurement

Measurement is central to ensuring that a production pipeline is working efficiently. After all, how can you know something is working better if you can't measure it?

Accurate and precise measurements allow us to pinpoint constraints in the pipeline and fix or improve them faster.

Measurements are also important from a cultural standpoint as they can inform teams when they're working more productively and what can be done to improve.

Measurement

The types of metrics we work to measure include:

Frequency of deployments

DevOps pipelines encourage frequent, smaller updates to software, so charting the frequency of deployments is a good indicator of the effectiveness of a pipeline.

Mean time to recovery (MTTR)

This refers to the average time it takes to solve problems that impact the end-user. Common problems include outages, security issues, and severe bugs.

Measurement

Mean time to discovery (MTTD)

This refers to how quickly problems are discovered. This metric is measured from the point of integration into production to the point the problem is identified.

System availability

We want our systems to be available at all times. Knowing the availability of our systems allows us to pinpoint which parts of our infrastructure need attention.

Service performance

Measuring response times per request, CPU load, loading times etc, allows us to see whether our services are running within the desired thresholds.

Task: Measurements

Outcome:

Make a list of what else we might want to measure in DevOps.

Steps:

What other technical/performance aspects should we measure?

What other measurements could help us improve our pipelines?

What non-technical measurements are useful in DevOps?

POWERING POTENTIAL

DevOps tooling and infrastructure

DevOps:
DevOps tooling and infrastructure

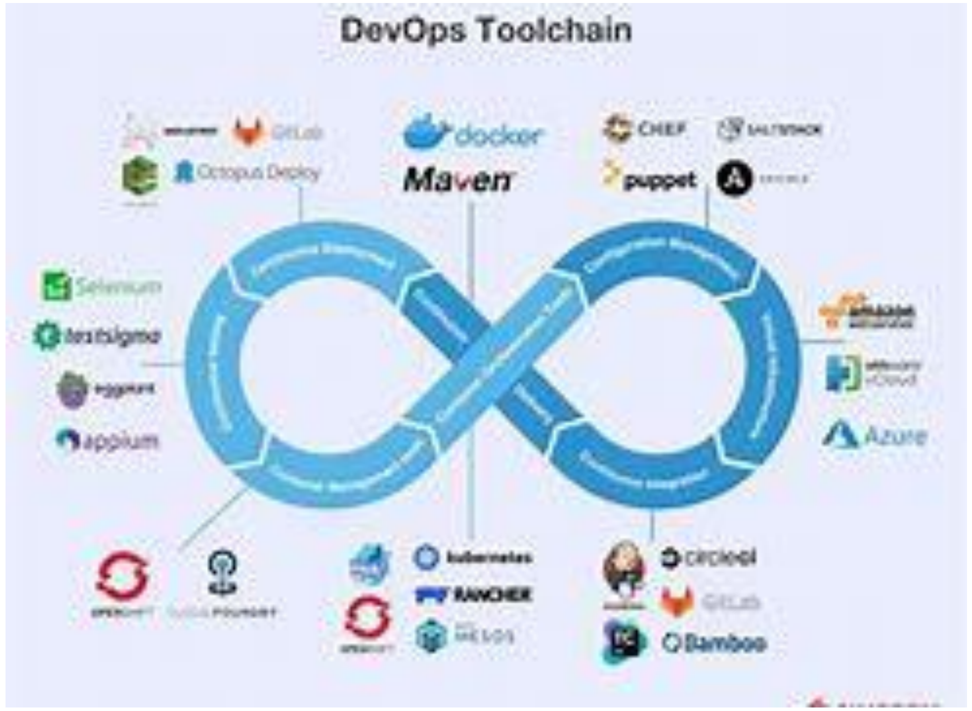
DevOps tooling and infrastructure

There are a number of tools required for DevOps to be effective, here we will briefly introduce some of them.

- Version Control System

A version control system (source control) is a place where developers check in and store their code, as well as make changes to it.

Popular source control tools include Git, Mercurial and Subversion.



DevOps tooling and infrastructure

Build server:

A build server, also known as an automation server, is a tool that compiles code from the source code into an executable code base.

The build server is able initiate a build of the software, run tests, document results, and complete post-build actions. Popular build servers include Jenkins, GitLab, and Bamboo.

Configuration management:

Configuration management focuses on configuring a node in a network or pipeline. This ensures that a given server or environment meets a declared, desired state.

Popular configuration management tools include Ansible, Puppet, Salt, and Chef.

DevOps tooling and infrastructure

Containerisation

Containerisation is becoming a standard practice within the world of development and DevOps. Containerisation allows for a developer to build, test, and deploy application in a way that is resource-independent.

This is because everything the application needs - dependencies, libraries, config files, source code - is bundled together into one container that can be used anywhere. DevOps needs containerisation (and the orchestration of these containers) to efficiently develop and deploy modern applications.

The most popular containerisation tool in the industry is Docker, with Kubernetes being the most popular container orchestration tool out there.

Task: DevOps tooling and infrastructure

Outcome:

Research the many IaC (infrastructure as code), configuration management and container orchestration tools available

Steps:

You will be split into three teams and assigned either Infrastructure as Code, Configuration Management or Container Orchestration as your topic to research. You must research a number of tools (one tool per team member) that can be used for your team's assigned topic.

SUMMARY

DEVOPS

Understand DevOps and the CI/CD pipeline

- What is DevOps and why do we use it in software development?

Describe DevOps as a culture

- How is a DevOps team organised?

Explore the different tools used in DevOps

- What kind of tools are there and why do we use them?

Thank you for listening

Any questions?