Agenda: DevOps Introduction

- Traditional Software Development Life Cycle
- Waterfall Model
- About Agile Methodology.
- What is DevOps?
- DevOps Practices?
- The Challenge
- Benefits of DevOps over Traditional IT
- DevOps Tools
- What is CI and CD?
- DevOps as a profession – DevOps Engineer

Traditional Software Development Life Cycle

The **developers** create applications and the **operations teams** deploy them to an infrastructure they manage.

Responsibility of Developers

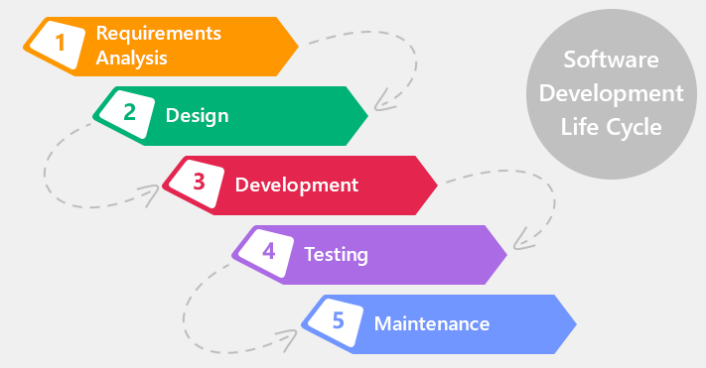
1. Develop Software Applications
2. New Features Implementation
3. Collaborate with other Developers in Team.
4. Maintain Source Repos and deal with versions.
5. Pass on the code to the operations team.

Responsibility of IT Operations

1. IT Operations determine how the software and hardware are managed.
2. Plan and Provide the required IT Infrastructure for Testing and Production of Applications.
3. Deploy the Application and Database.
4. Validate and Monitor performance.

Waterfall Model

In the below diagram you will see the phases it will involve:



Requirement Gathering stage	*During this phase, detailed requirements of the software system to be developed are gathered from client*
Design Stage	Plan the programming language, for Example Java, PHP, .net or database like MSSQL, Oracle, MySQL, etc. Or other high-level technical details of the project
Development Stage	After design stage, it is built stage, that is nothing but coding the software
Test Stage	In this phase, you test the software to verify that it is built as per the specifications given by the client.
Deployment stage	Deploy the application in the respective environment.
Maintenance stage	Once your system is ready to use, you may later require change the code as per customer request

How the traditional Systems worked:

Tasks would be divided into different groups based on specialization

1. Group to write specification
2. Group that Develop application.
3. Group that Test the application.
4. Group to configure and manage VM.
5. Group to that hands over VM to another group to install database.
6. and so on…

A system / process is created for each action and each group operations in isolation from others. Groups communicate with each other in a very formal way, such as using ticketing system.

Drawback:

- This requires handoffs from one group to another. This can introduce significant delays, inconsistencies and inaccuracies.
- Hard to accurately define requirements, which can change over time. After delivery requires change requests and additional cost.
- Lack of a common approach among the groups contributes to the problems of long build times and errors.

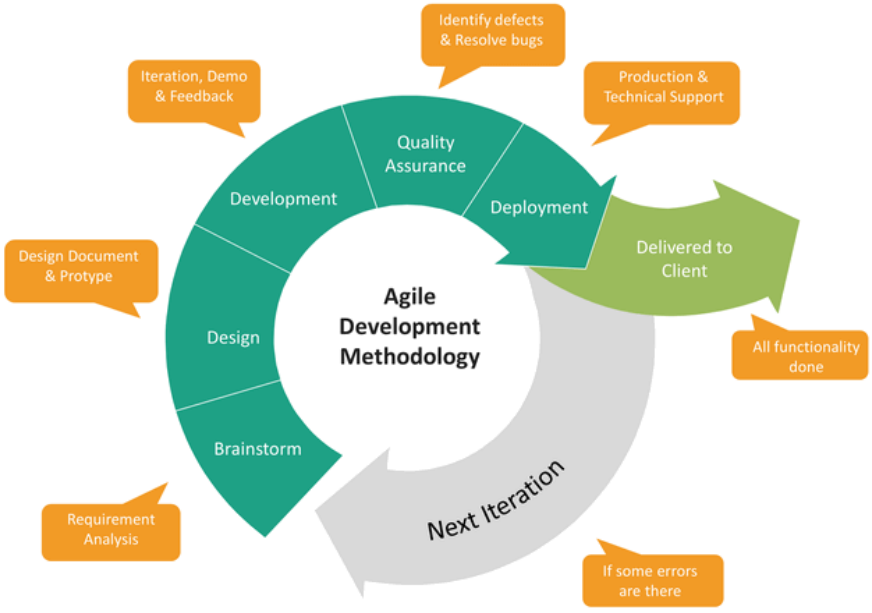
And blame game begins…

What is Agile Methodology

Agile is a process by which a team can manage a project by **breaking** it up into **several stages** and involving constant **collaboration** with stakeholders and **continuous** improvement and iteration at every stage.

- Emphasizes constantly adaptive planning, and early delivery with continual improvement.
- Development methods are based on releases and iterations.
- At the end of each iteration, there should be tested working code.
- Focused on these shorter-term outcomes.

There are no surprises. **Continuous collaboration is key**, both among team members and with project stakeholders, to make fully-informed decisions.



Principles of Agile Development

1. Satisfy the customer
2. Welcome changing requirements
3. Deliver working software frequently
4. Work together throughout the project
5. Build projects around motivated individuals
6. Use face-to-face conversation
7. Measure progress through working software
8. Agile processes promote sustainable development
9. Continuous attention to technical excellence and good design
10. Simplicity--the art of minimizing the amount of work not done
11. Use self-organizing teams
12. Reflect on how to become more effective

Mentoring Team Members on Agile Practices

- Many teams hire external Agile coaches or mentors.
- Agile coaches have teaching and mentoring skills.
- Agile coaches tend to be both trainers and consultants.
- Some coaches are technical experts.
- Some coaches are focused on Agile processes.

Enabling In-Team and Cross-Team Collaboration

- **Cultural changes** – more open work spaces, meeting etiquette, outsourcing, better communication.
- **Cross-functional teams** – collaboration with others, diversity of opinion, rewarding collective behavior.
- **Collaboration tooling** – Skype, Slack, Teams, Google Meet.

What is DevOps (Development and Operations)**?**

DevOps Lessons from Formula 1

https://www.devopsgroup.com/blog/devops-lessons-formula-1-part-2/

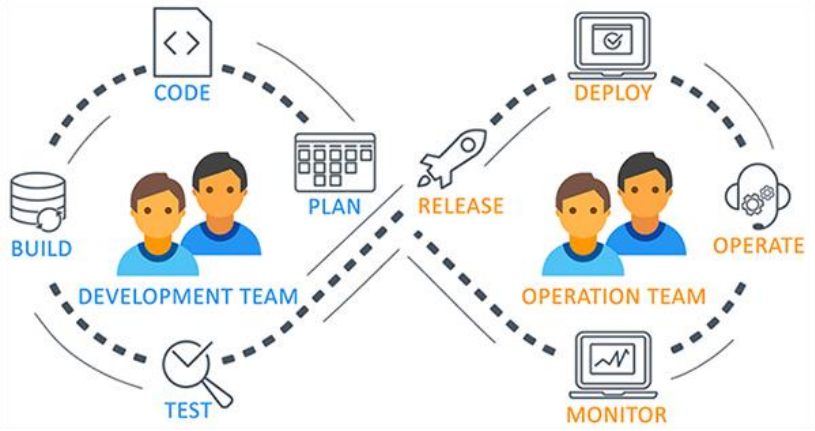
The word "DevOps" was coined in 2009 by Patrick Debois, who became one of its gurus.

What is DevOps?

- DevOps is an Idea / thinking to work in a certain way that could solve the problem of Developers and IT Operation teams within the organization.
- **At Microsoft:** DevOps is the union of **people, process, and products** to enable continuous delivery of value to **our end users**.
- **DevOps is a continuous journey**

- The contraction of **"Dev" and "Ops"** refers to replacing siloed Development and Operations to create multidisciplinary teams that now work together with shared and efficient practices and tools. Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle.
- DevOps is a **culture** and **technical** movement that emphasizes using of **various Automation tools** for collaboration and communication of both software developers and other information-technology (IT) professionals.

How DevOps Works?

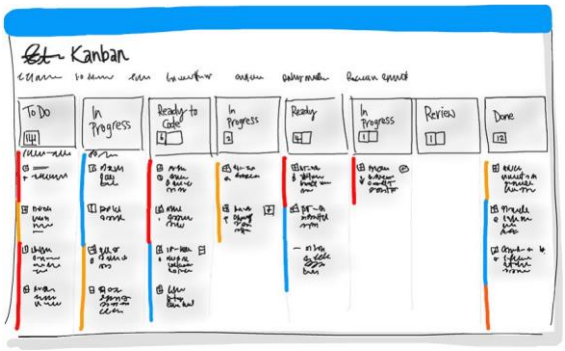


Starting from design and development to testing **automation** and from **continuous integration** to **continuous delivery**, the team works **together** to achieve the desired goal. People having both development and operations skill sets work **together** and use various tools for CI-CD and Monitoring to respond **quickly** to customers need and fix issues and bugs.

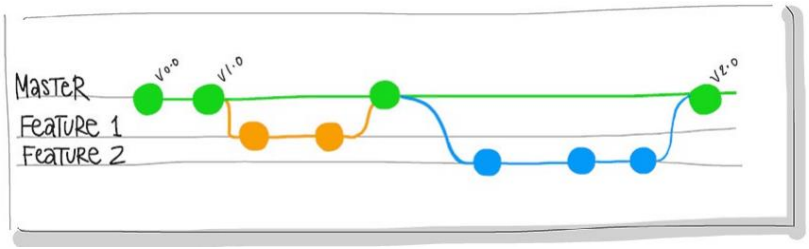
DevOps Practices

When you adopt DevOps practices, you **shorten your cycle** time by working in smaller batches, using more automation, hardening your release pipeline, improving your telemetry, and deploying more frequently.

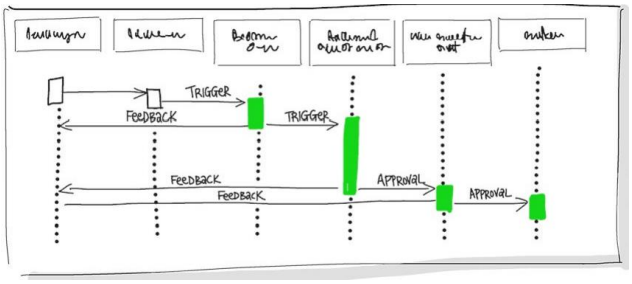
- **Agile planning**. Together, we'll create a backlog of work that everyone on the team and in management can see. We'll **prioritize** the items so we know what we need to work on first. The backlog can include user stories, bugs, and any other information that helps us.



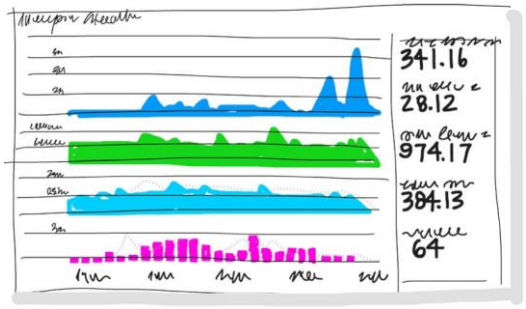
- **Version Control,** Usually With Git, enables teams located anywhere in the world to communicate effectively during daily development activities as well as to integrate with software development tools for monitoring activities such as deployments.



- **Continuous Integration (CI)**. We'll automate how we **build** and **test** our code. We'll run that every time a team member commits changes to version control. This leads to finding defects early. Other benefits include less time wasted on fighting merge issues and rapid feedback for development teams.
- **Continuous Delivery (CD)**. CD is how we test, configure, and deploy from a build to a QA or production environment.



- **Continuous Monitoring**. We'll use telemetry to get information about an application's performance and usage patterns. We can use that information to improve as we iterate.



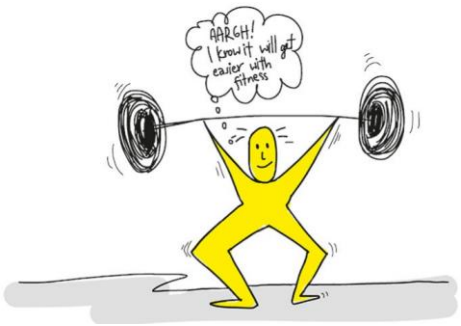
When you adopt DevOps practices, you **shorten your cycle time** by working in smaller batches, **using more automation**, hardening your release pipeline, improving your telemetry, and deploying more frequently.

The Challenge!!!

The biggest challenge is breaking down a process based on decades of rules, regulations, and frustrating areas of comfort: "It is how we have always done it; why change?"

DevOps May Hurt at First

If it hurts, do it more often. Just like going to the gym, adopting new practices is likely to hurt at first. The more often you exercise the new practices, the easier they will become. And just like training at the gym, where you exercise large muscles before small muscles, adopt practices that have the greatest impact first and cross-train to develop synergy.



There are several challenges when creating teams:

- Availability of staff.
- Disruption of current procedures.

Separating Transformation Teams

- For DevOps transformations, the **separate team** should be made up of staff members, all of whom are focused on and measured on the transformation outcomes, and **not involved in the operational day-to-day work**.
- The team might also include some **external experts** that can **fill the knowledge gaps** and help to advise on processes that are new to the existing staff members.
- Ideally the staff members who were recruited for this should already be **well-regarded** throughout the organization and as a group they should **offer a broad knowledge base** so they can think outside the box.

Ideal DevOps team members

- They think there is a **need to change** and have shown an ability to innovate.
- They are **well-respected** and have broad knowledge of the organization and how it operates.
- Ideally, they already **believe** that DevOps practices are what is needed.

Defining Shared Goals

One of the key aims of DevOps is to provide **greater customer value**, so outcomes should have a customer value focus.

Projects must have a clearly-defined set of measurable outcomes, like:

- Reduce the time spent on fixing bugs by 60%.
- Reduce the time spent on unplanned work by 70%.
- Reduce the out-of-hours work required by staff to no more than 10% of total working time.
- Remove all direct patching of production systems.

Measurable goals should have timelines that challenging yet achievable.

Benefits of DevOps over Traditional IT

Speed + Rapid Delivery + Reliability + Scale + Improved Collaboration + Security

With DevOps, teams:

1. **Deploy more frequently**

   In fact, some teams deploy up to dozens of times per day.

   Practices such as monitoring, continuous testing, database change management, and integrating security earlier in the software development process help elite performers deploy more frequently, and with greater predictability and security.

2. **Reduce lead time from commit to deploy**

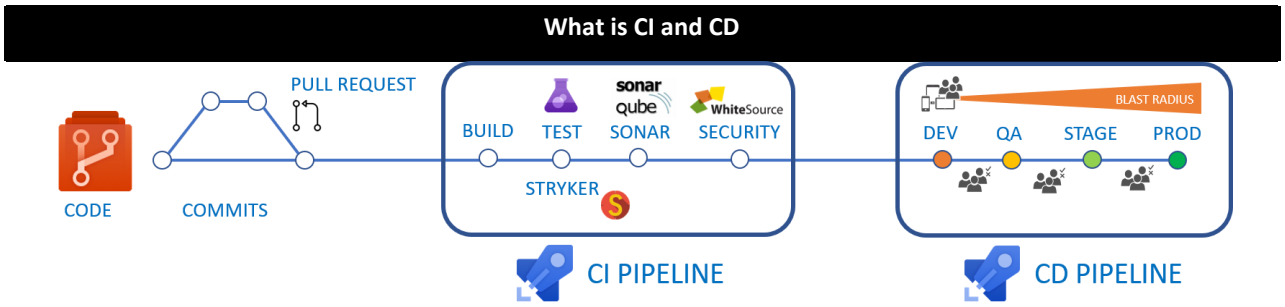
   Lead time is the time it takes for a feature to make it to the customer. By working in smaller batches, automating manual processes, and deploying more frequently, elite performers can achieve in hours or days what once took weeks or even months.

3. **Reduce change failure rate**

   A new feature that fails in production or that causes other features to break can create a lost opportunity between you and your users. As high-performing teams mature, they reduce their change failure rate over time.

4. **Recover from incidents more quickly**

   When incidents do occur, elite performers are able to recover more quickly. Acting on metrics helps elite performers recover more quickly while also deploying more frequently.



- **Continuous integration** is a software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.
- **Continuous delivery** is a software development practice where code changes are automatically built, tested, and prepared for a release to production. It expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage. When continuous

delivery is implemented properly, developers will always have a deployment-ready build artifact that has passed through a standardized test process.

CI and CD enables agile teams to **increase deployment frequency** and **decrease lead time for change**, change-failure rate, and mean time to recovery key performance indicators (KPIs), thereby **improving quality and delivering value faster**. The only prerequisites are a solid development process, a mindset for quality and accountability for features from ideation to deprecation, and a comprehensive pipeline.

DevOps Tools

To expedite and actualize DevOps process apart from culturally accepting it, one also needs various DevOps tools like Puppet, Jenkins, GIT, Chef, Docker, Selenium, Azure/AWS etc to achieve automation at various stages which helps in achieving **Continuous Development, Continuous Integration, Continuous Testing, Continuous Deployment, Continuous Monitoring** to deliver a quality software to the customer at a very fast pace.

Tools Required for

- Version Control System
- Configuration management
- Ticketing System
- Resource Monitoring
- Provisioning