WELCOME

Sinhgad Institutes

Elective – IV (Introduction to DevOps)

[414445]

Mrs. A.P. Kulkarni

Assistant Professor

Dept. Of Information Technology,

Sinhgad Institute of Technology, Lonavala

apk.sit@sinhgad.edu

Cell. +91 9767009703

Sinhgad Institutes

B.E. [Information Technology]

Elective – IV (Introduction to DevOps)

[414445]

UNIT No.: 1

Introduction to DevOps and the Culture

(06 Hrs)

Unit I : Introduction to DevOps and the Culture

Sinhgad Institutes

Contents: What is DevOps? Role of DevOps Engineer, Developer responsibility, Introduction to Continuous Integration and Continuous Delivery Policies, DevOps Culture: Dilution of barriers in IT departments, Process automation, Agile Practices, Reason for adopting DevOps, What and Who Are Involved in DevOps? Changing the Coordination, Introduction to DevOps pipeline phases , Defining the Development Pipeline, Centralizing the Building Server, Monitoring Best Practices, Best Practices for Operations.

Unit Objectives

1. Understand basics Information about DevOps.
2. Introduction to Continuous Integration and Continuous Delivery Policies.
3. Introduction to DevOps pipeline phases

Unit outcomes: On completion the students will be able to

1. Understand what DevOps & Role of DevOps Engineer.
2. Gain knowledge of Continuous Integration and Continuous Delivery Policies.
3. Understand different pipeline phases of DevOps

Mapping of Course Outcomes for Unit I: CO1

Teaching Plan

Sinhgad Institutes

Lect. No.	Contents	Reference /Text Books
1	What is DevOps? Role of DevOps Engineer, Developer responsibility,	T1:Chapter 1
2	Introduction to Continuous Integration and Continuous Delivery Policies, DevOps Culture: Dilution of barriers in IT departments,	T1:Chapter 1
3	Process automation, Agile Practices	T1:Chapter 1
4	Reason for adopting DevOps, What and Who Are Involved in DevOps? Changing the Coordination,	T1:Chapter 1
5	Introduction to DevOps pipeline phases, Defining the Development Pipeline	T1:Chapter 1
6	Centralizing the Building Server, Monitoring Best Practices, Best Practices for Operations.	T1:Chapter 1

Sinhgad Institutes

Lecture 1

What is DevOps? Role of DevOps Engineer, Developer responsibility

What is DevOps?

Sinhgad Institutes

DevOps ("development" and "operations") is the combination of practices and tools designed to increase an organization's ability to deliver applications and services faster than traditional software development processes.

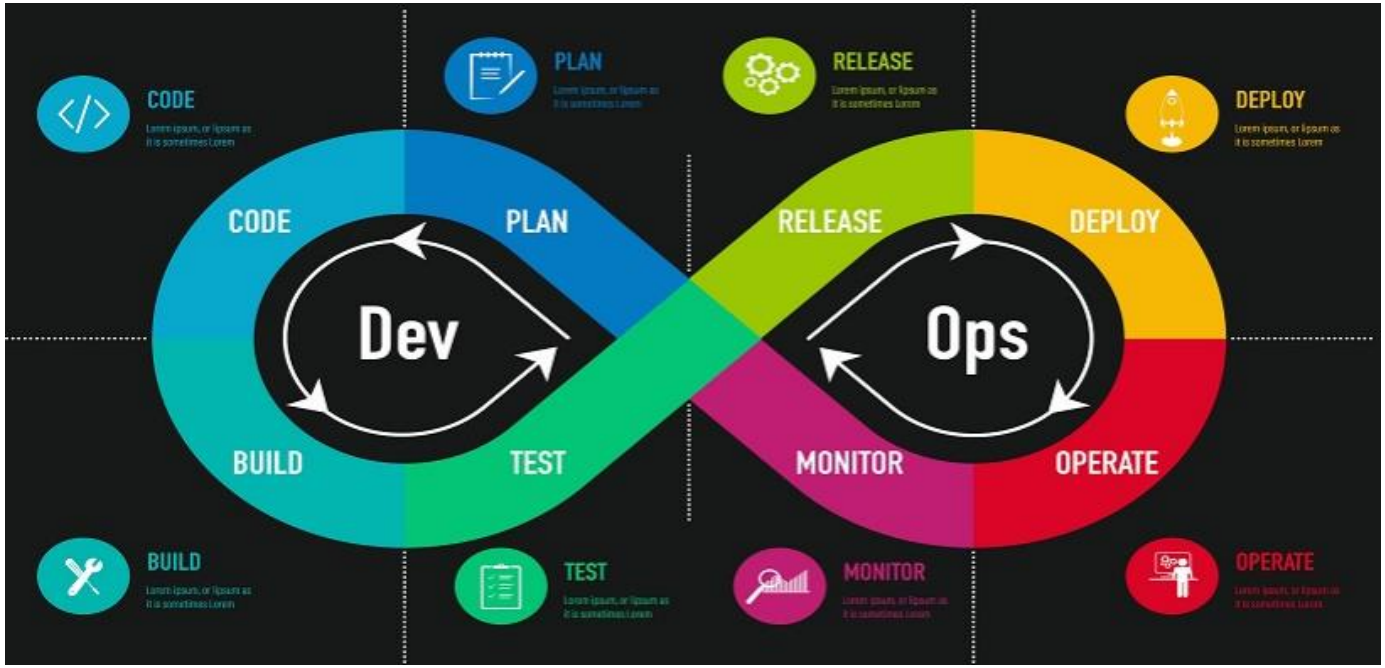
DevOps is basically a combination of two words- **Dev**elopment and **Op**eration**s**. DevOps is a culture that implements the technology in order to promote collaboration between the developer team and the operations team to deploy code to production faster in an automated and repeatable way.

What is DevOps?

Sinhgad Institutes

DevOps is a collaborative approach that combines software development (Dev) and IT operations (Ops), enabling organizations to deliver high-quality software faster and more efficiently.

By emphasizing automation, continuous integration, and continuous delivery, DevOps fosters a culture of collaboration, agility, and rapid innovation, transforming the way software is developed, tested, and deployed.



DevOps enablers include:

- Agile and lean software development practices
- Agile and lean service management practices
- Virtualized and cloud infrastructure from internal and external providers
- Treating infrastructure as code
- Data center automation and configuration management tools
- Monitoring and self-healing technologies

The goal of DevOps is to increase an organization's speed when it comes to delivering applications and services. Many companies have successfully implemented DevOps to enhance their user experience including Amazon, Netflix, etc.

Facebook's mobile app which is updated every two weeks effectively tells users you can have what you want and you can have it. Now ever wondered how Facebook was able to do social smoothing? It's the DevOps philosophy that helps Facebook ensure that apps aren't outdated and that users get the best experience on Facebook. Facebook accomplishes this true code ownership model that makes its developers responsible that includes testing and supporting through production and delivery for each kernel of code. They write and update their true policies like this but Facebook has developed a DevOps culture and has successfully accelerated its development lifecycle.

- DevOps word in itself is a combination of two words one is Development and other is Operations. It is neither an application nor a tool; instead, it is just a culture to promote development and Operation process collaboratively. As a result of DevOps implementation, the speed to deliver applications and services has increased.

Introduction

Sinhgad Institutes

- DevOps enables organizations to serve their customers strongly and better in the market. In other words, we can say that DevOps is the process of alignment of IT and development operations with better and improved communication.
- In DevOps, the operations team have a complete idea of the progress of development. Operations team and development team work together to develop a monitoring plan that caters to the current business, and IT needs.

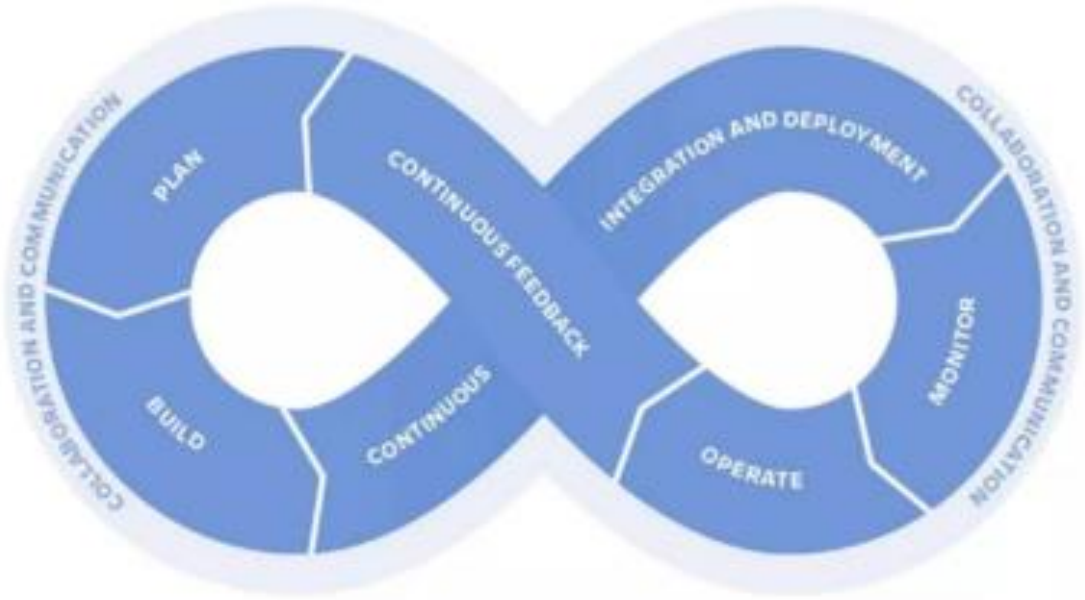
History of DevOps

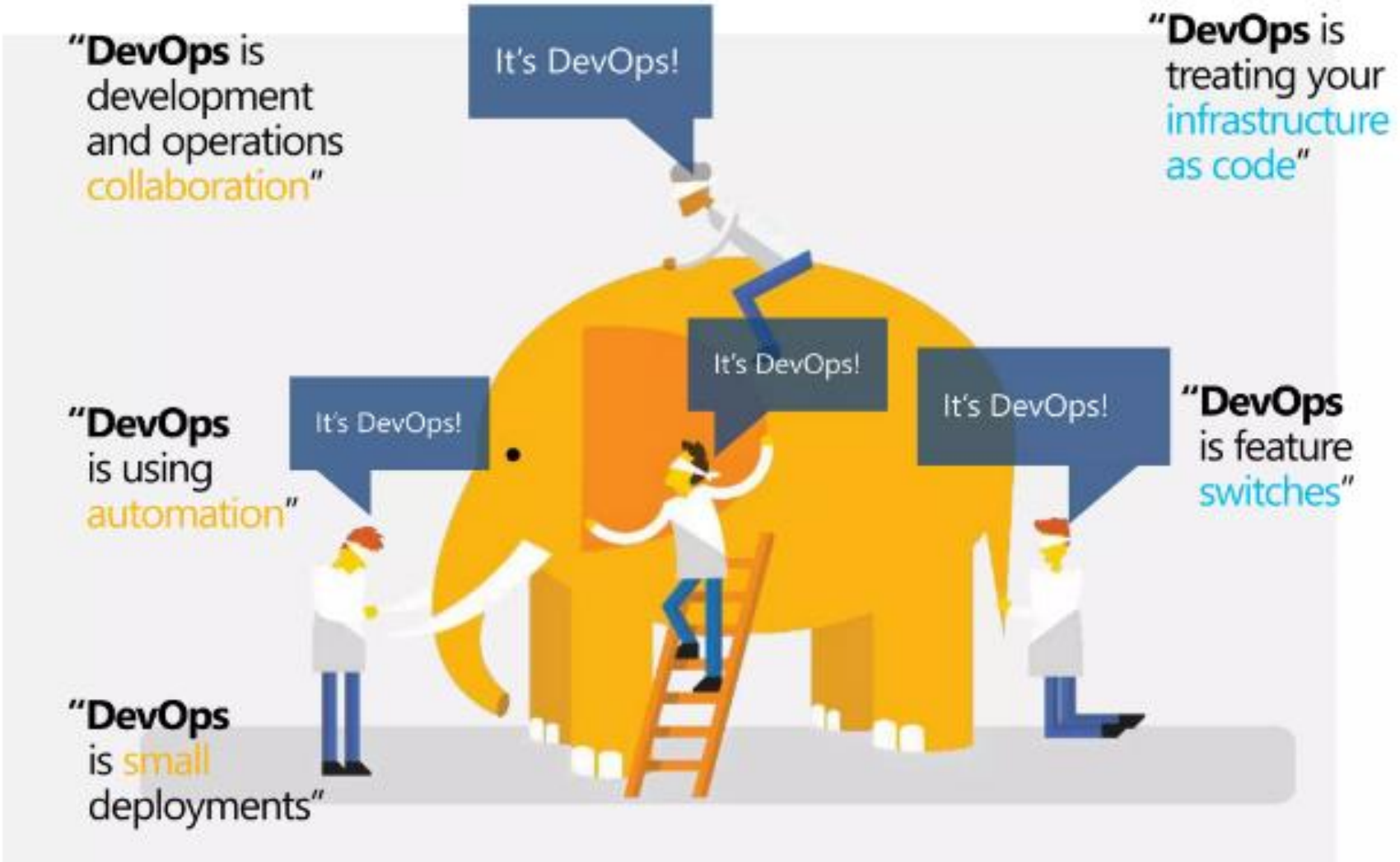
Sinhgad Institutes

- Patrick Debois, a Belgian consultant, project manager, and agile practitioner is one among the initiators of DevOps.
- A presentation on "10+ Deploys per Day: Dev and Ops Cooperation at Flickr" helped in bring out the ideas for DevOps and resolve the conflict of " It's not my code, it's your machines! "
- DevOps blends lean thinking with agile philosophy.

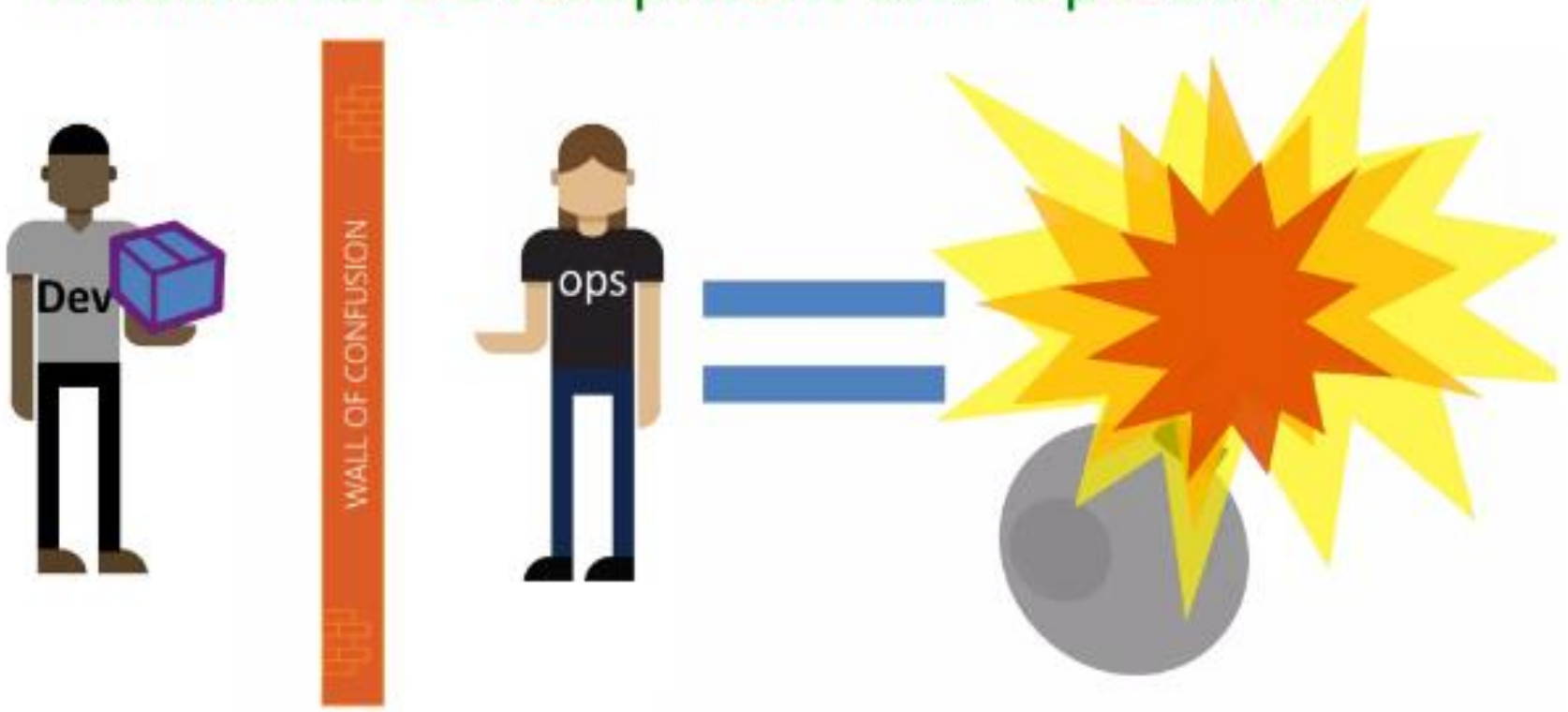
Definition

DevOps is a set of practices and tools designed to shorten the life cycle of a software development process.

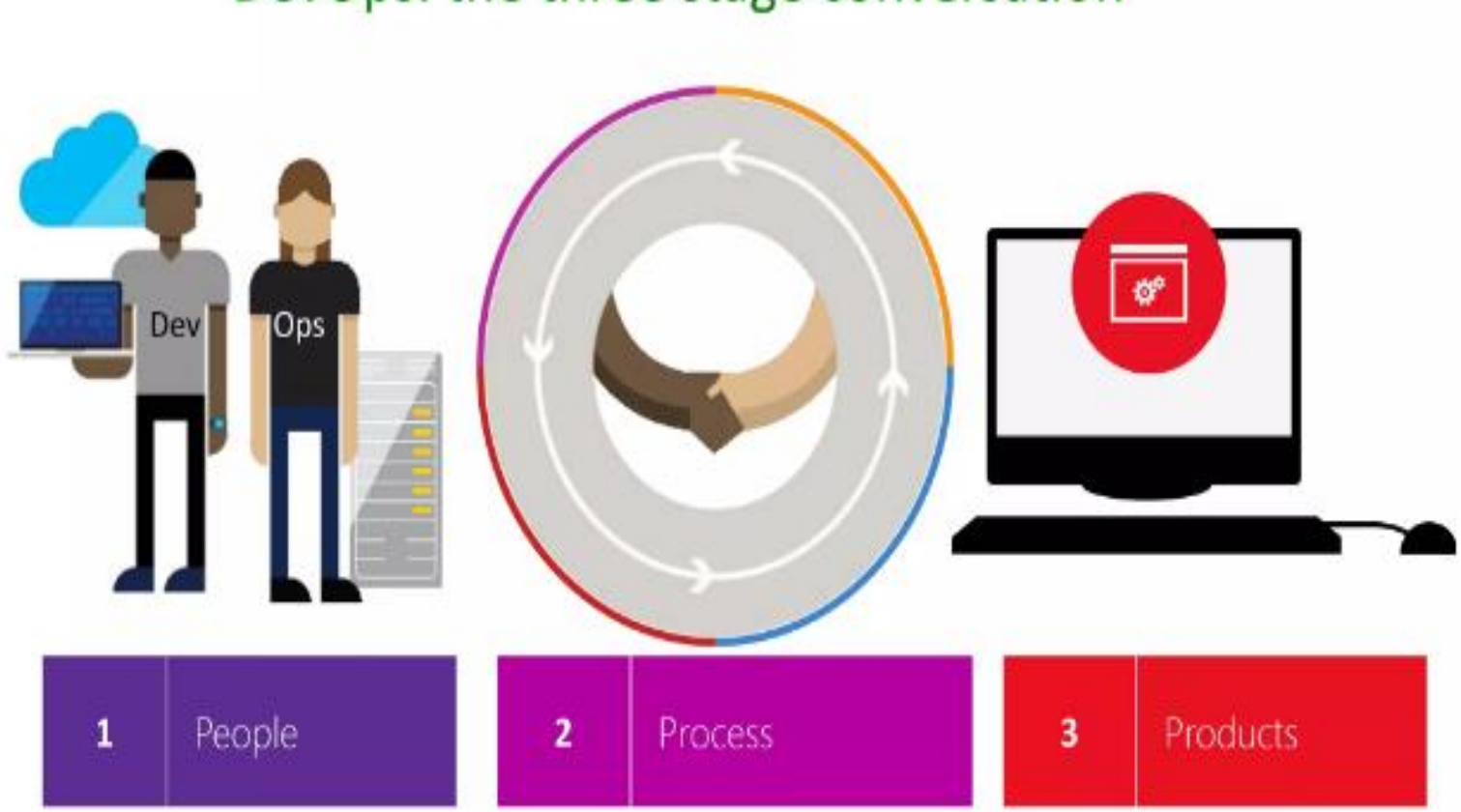



"DevOps is development and operations collaboration"
It's DevOps!
"DevOps is treating your infrastructure as code"
It's DevOps!
It's DevOps!
It's DevOps!
"DevOps is using automation"
"DevOps is feature switches"
"DevOps is small deployments"

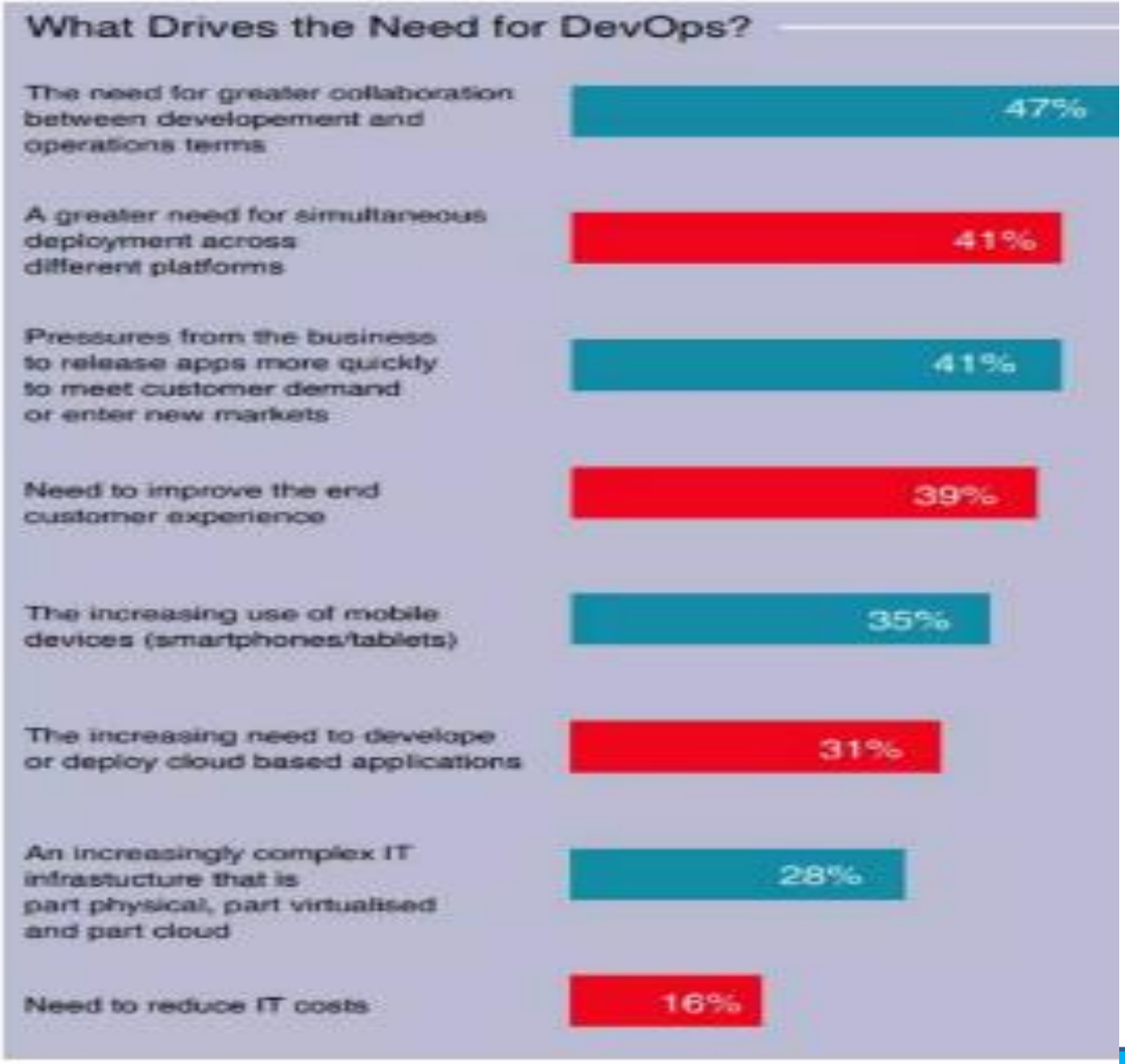
Traditional Development and Operations



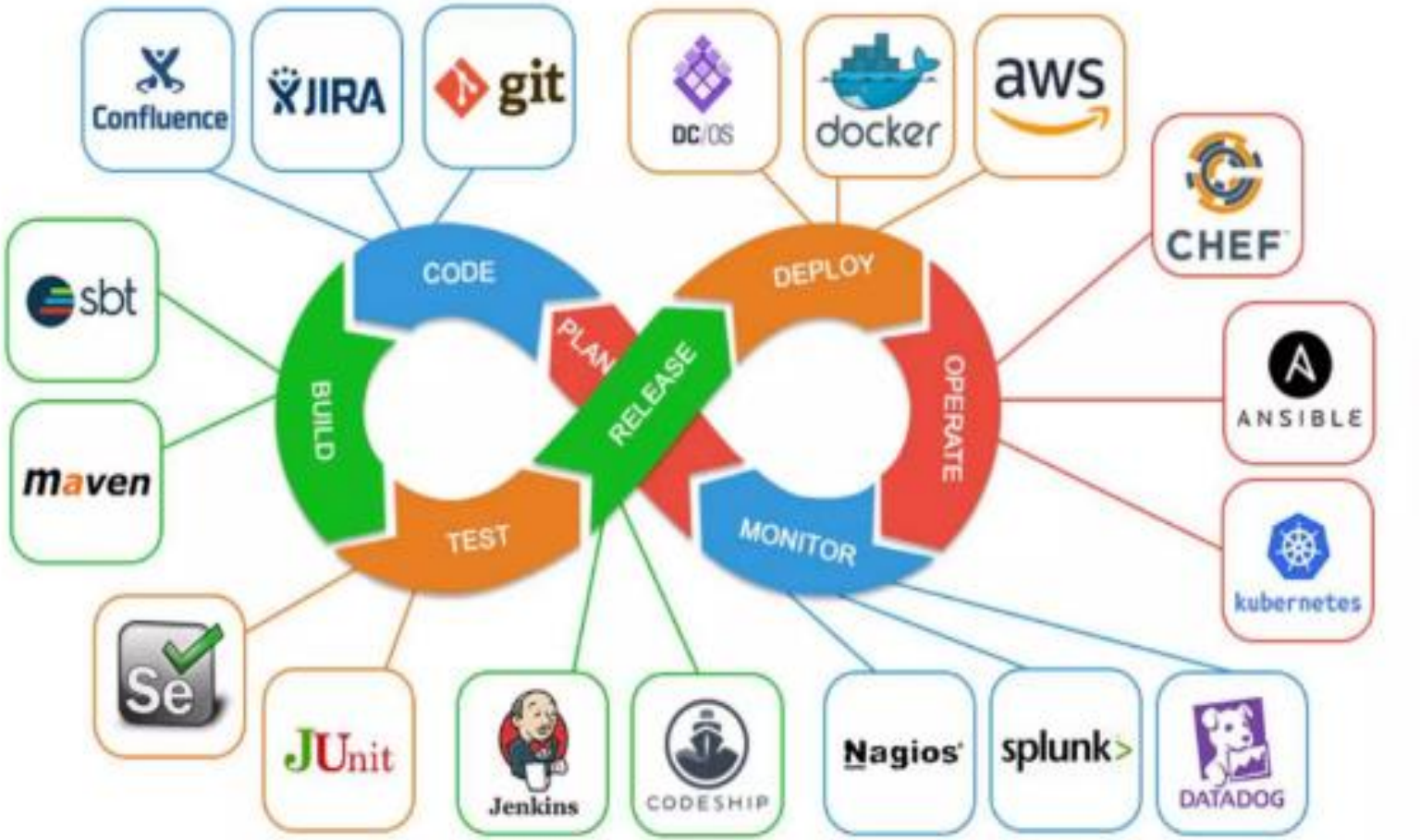
DevOps: the three stage conversation



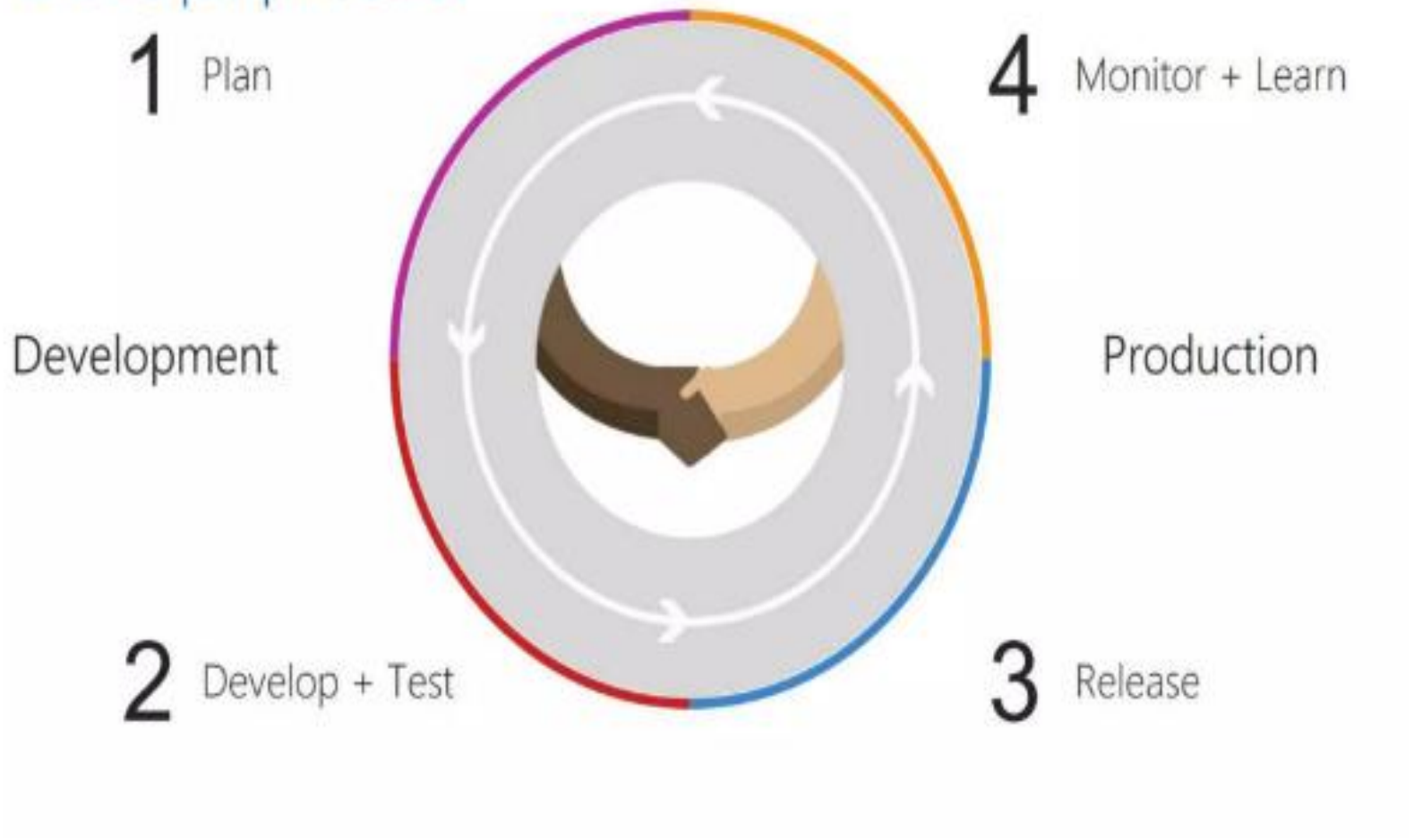
Need for DevOps



DevOps Life Cycles

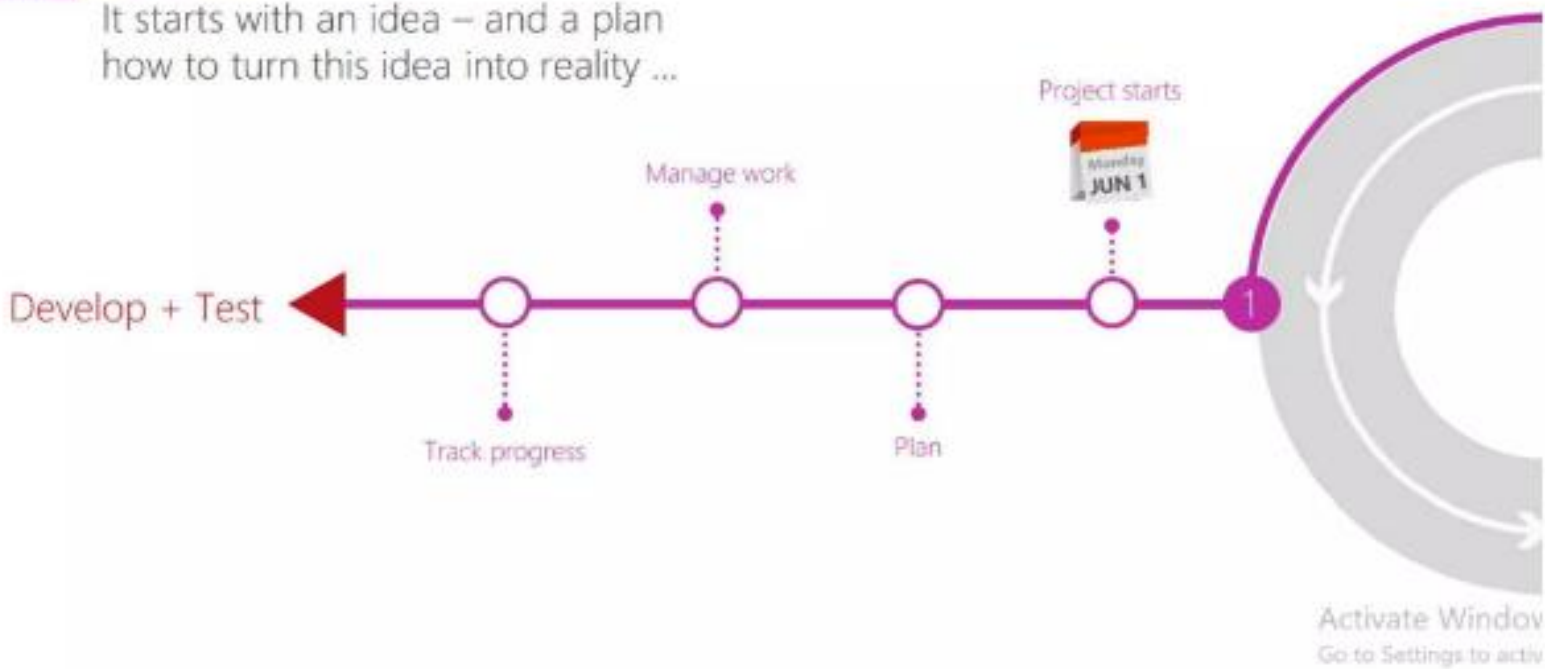


DevOps process



Plan

It starts with an idea – and a plan
how to turn this idea into reality ...

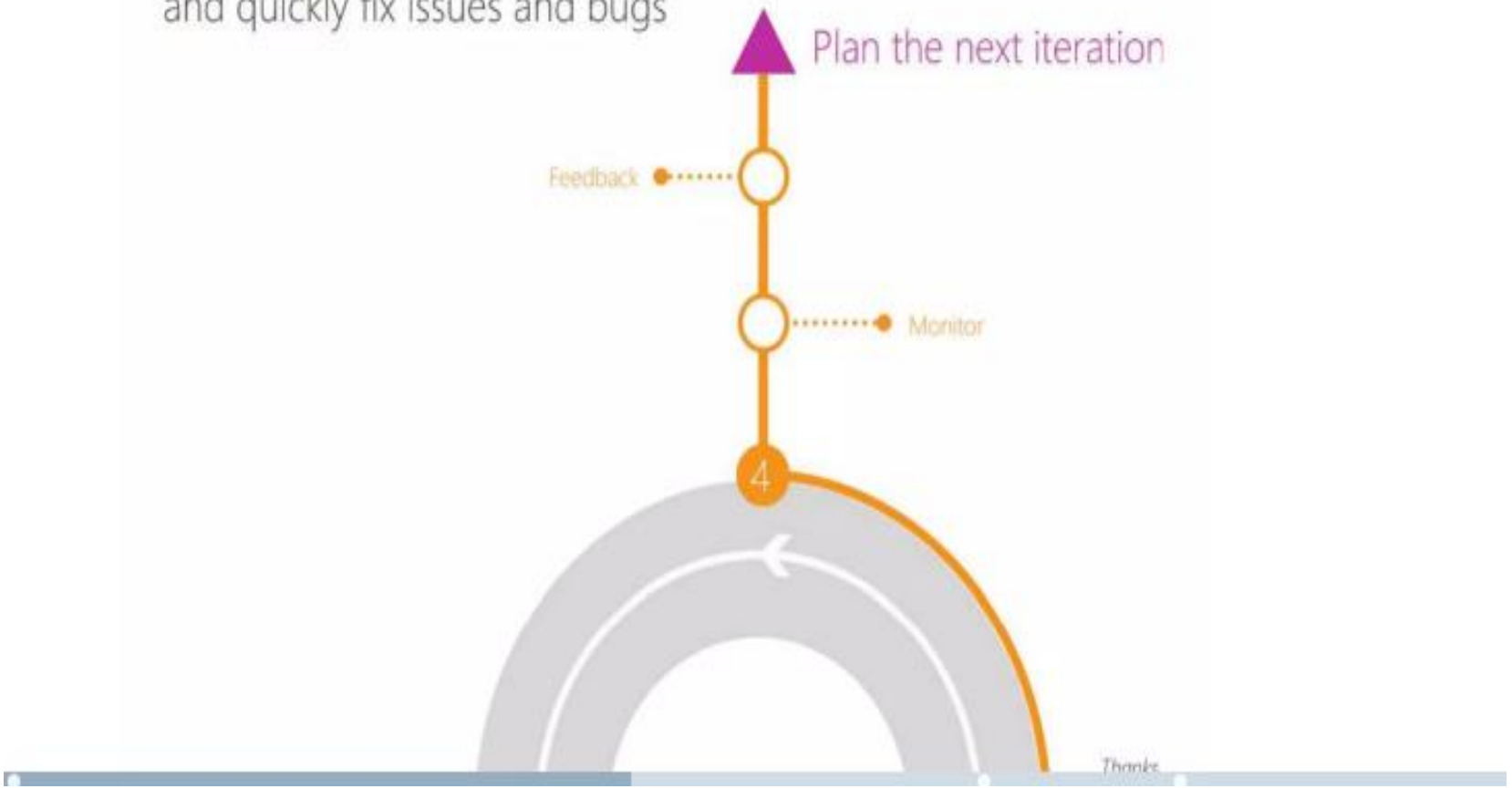


Develop + Test

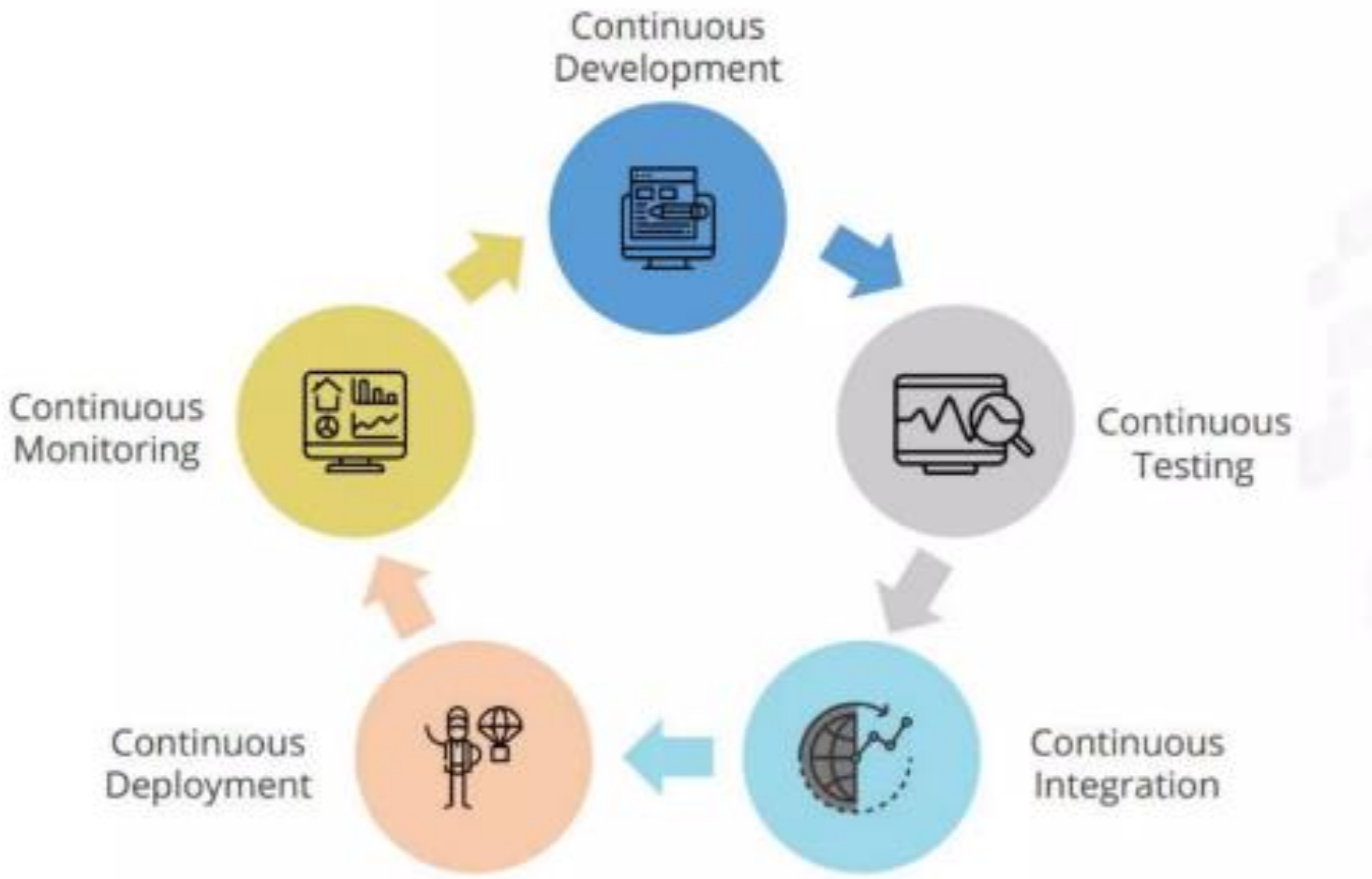
Once the iteration starts, developers turn great ideas into features ...

Monitor + Learn

Learn and understand how users use your app, how it reacts and quickly fix issues and bugs



DevOps Architecture



Role of DevOps Engineer

Sinhgad Institutes

- The primary role of a DevOps Engineer is to introduce methodologies to balance needs throughout the software development life cycle, processes, and tools, from coding to development to maintenance and updates.
- They monitor health and track everything happening in all system parts during the software lifecycle.
- DevOps engineers build, test and maintain infrastructure and tools so that software can be developed and released.
- DevOps (development operations) is a series of practices and processes that help organizations speed up and automate aspects of developing, testing, releasing and updating software.

- DevOps Engineer has to investigate and resolve technical issues, provide level 2 technical support, perform root cause analysis for production errors, build tools to improve customer experience and develop software to integrate with internal back-end systems.
- A DevOps Engineer should be skilled, knowledgeable, and capable of automating the entire DevOps pipeline, including application performance monitoring, CI/CD cycles, infrastructure and configuration management, and many others.
- They must deeply understand the DevOps toolset, coding, and scripting.

Developer responsibility

Sinhgad Institutes

- DevOps Engineer responsibilities include deploying product updates, identifying production issues and implementing integrations that meet customer needs.
- A DevOps engineer works with diverse teams and departments to create and implement software systems. People who work in DevOps are experienced IT professionals who collaborate with software developers, quality assurance professionals, and IT staff to manage code releases.

Developer responsibility

Sinhgad Institutes

DevOps engineers need to be able to multitask, demonstrate flexibility, and deal with many different situations at a time. Specifically, a DevOps engineer's responsibilities include:

- **Documentation:** Writes specifications and documentation for the server-side features.
- **Systems analysis**: Analyzes the technology currently being used and develops plans and processes for improvement and expansion. The DevOps engineer provides support for urgent analytic needs.
- **Development:** Develops, codes, builds, installs, configures, and maintains IT solutions.

Developer responsibility

Sinhgad Institutes

- **Project planning**: Participates in project planning meetings to share their knowledge of system options, risk, impact, and costs vs. benefits. In addition, DevOps engineers communicate operational requirements and development forecasts.
- **Testing:** Tests code, processes, and deployments to identify ways to streamline and minimize errors.
- **Deployment:** Uses configuration management software to automatically deploy updates and fixes into the production environment.

Developer responsibility

Sinhgad Institutes

- Maintenance and troubleshooting: Performs routine application maintenance to ensure the production environment runs smoothly. Develops maintenance requirements and procedures.
- Performance management: Recommends performance enhancements by performing gap analysis, identifying alternative solutions, and assisting with modifications.
- Management: Depending on the size of the organization, the DevOps engineer may also be responsible for managing a team of DevOps engineers.

Lecture 2

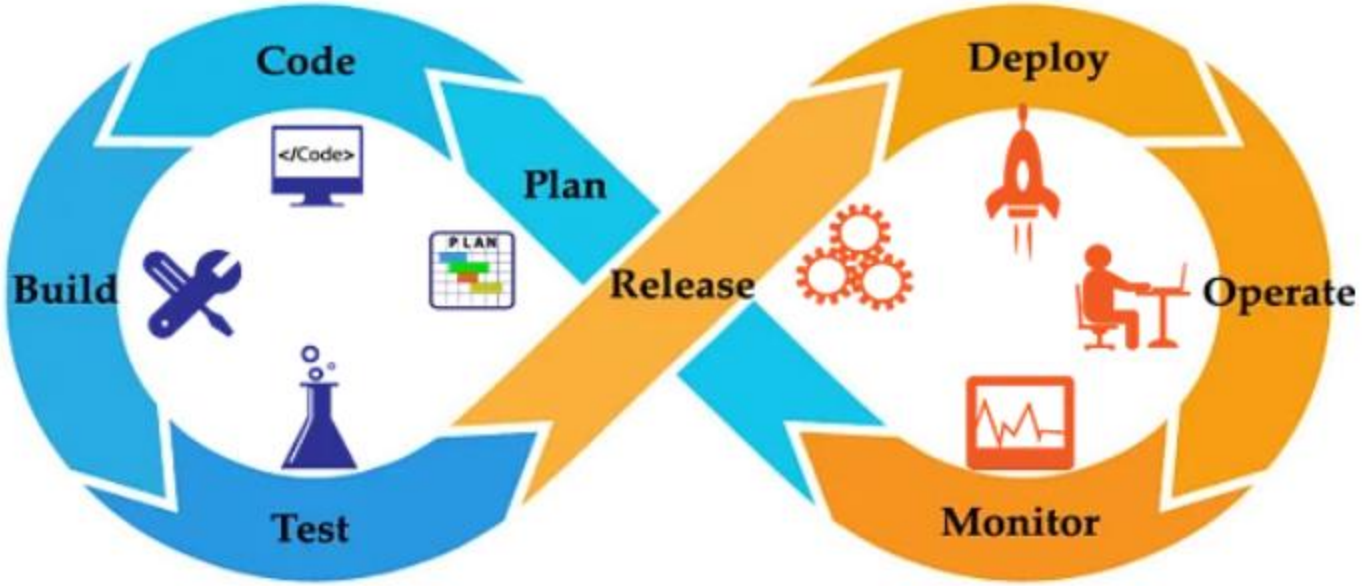
Introduction to Continuous Integration and Continuous Delivery Policies, DevOps Culture: Dilution of barriers in IT departments,

What is Continuous Integration?

Sinhgad Institutes

Continuous Integration (CI) is a DevOps software development practice that enables the developers to merge their code changes in the central repository. That way, automated builds and tests can be run. The amendments by the developers are validated by creating a built and running an automated test against them.

In the case of Continuous Integration, a tremendous amount of emphasis is placed on testing automation to check on the application. This is to know if it is broken whenever new commits are integrated into the main branch.



What is Continuous Integration?

Sinhgad Institutes

- **Continuous integration** is a practice that encourages developers to integrate their code into a main branch of a shared repository early and often.
- Instead of building out features in isolation and integrating them at the end of a development cycle, code is integrated with the shared repository by each developer multiple times throughout the day.
- The idea is to minimize the cost of integration by making it an early consideration.
- Developers can discover conflicts at the boundaries between new and existing code early, while conflicts are still relatively easy to reconcile.
- Once the conflict is resolved, work can continue with confidence that the new code honors the requirements of the existing codebase.

What is Continuous Integration?

Sinhgad Institutes

- The end goal of continuous integration is to make integration a simple, repeatable process that is part of the everyday development workflow in order to reduce integration costs and respond to defects early.
- Working to make sure the system is robust, automated, and fast while cultivating a team culture that encourages frequent iteration and responsiveness to build issues is fundamental to CI success.

What is Continuous Delivery?

Sinhgad Institutes

- Continuous Delivery (CD) is a DevOps practice that refers to the building, testing, and delivering improvements to the software code.
- The phase is referred to as the extension of the Continuous Integration phase to make sure that new changes can be released to the customers quickly in a substantial manner.
- This can be simplified as, though you have automated testing, the release process is also automated, and any deployment can occur at any time with just one click of a button.

What is Continuous Delivery?

- Continuous Delivery gives you the power to decide whether to make the releases daily, weekly, or whenever the business requires it.
- The maximum benefits of Continuous Delivery can only be yielded if they release small batches, which are easy to troubleshoot if any glitch occurs.

What is Continuous Delivery?

Sinhgad Institutes

- **Continuous delivery** is an extension of continuous integration.
- It focuses on automating the software delivery process so that teams can easily and confidently deploy their code to production at any time.
- By ensuring that the codebase is always in a deployable state, releasing software becomes an unremarkable event, without any complicated rituals.
- Teams can be confident that they can release whenever they need to without complex coordination or late-stage testing.
- As with continuous integration, continuous delivery is a practice that requires a mixture of technical and organizational improvements to be effective.

What is Continuous Delivery?

Sinhgad Institutes

- Continuous delivery is compelling because it automates the steps between checking code into the repository and deciding on whether to release well-tested, functional builds to your production infrastructure.

- The steps that help assert the quality and correctness of the code are automated, but the final decision about what to release is left in the hands of the organization for maximum flexibility.

What is DevOps culture?

Sinhgad Institutes

- At its essence, a DevOps culture involves closer collaboration and a shared responsibility between development and operations for the products they create and maintain.
- This helps companies align their people, processes, and tools toward a more unified customer focus.
- It involves cultivating multidisciplinary teams who take accountability for the entire lifecycle of a product.
- DevOps teams work autonomously and embrace a software engineering culture, workflow, and toolset that elevates operational requirements to the same level of importance as architecture, design and development.
- The understanding that developers who build it, also run it, brings developers closer to the user, with a greater understanding of user requirements and needs.
- With operations teams more involved in the development process, they can add maintenance requirements and customer needs for a better product.

What is DevOps culture?

Sinhgad Institutes

At its essence, a DevOps culture involves closer collaboration and a shared responsibility between development and operations for the products they create and maintain.

This helps companies align their people, processes, and tools toward a more unified customer focus.

It involves cultivating multidisciplinary teams who take accountability for the entire lifecycle of a product.

DevOps teams work autonomously and embrace a software engineering culture, workflow, and toolset that elevates operational requirements to the same level of importance as architecture, design and development.

The understanding that developers who build it, also run it, brings developers closer to the user, with a greater understanding of user requirements and needs. With operations teams more involved in the development process, they can add maintenance requirements and customer needs for a better product.

What are the benefits of DevOps culture?

Sinhgad Institutes

- The most obvious and impactful benefit of embracing a DevOps culture is streamlined, frequent, and high-quality software releases.
- This not only increases company performance, but also employee satisfaction.
- A DevOps culture fosters high levels of trust and collaboration, results in higher quality decision making, and even higher levels of job satisfaction, according to the book "Accelerate: Building and Scaling High Performing Technology Organizations."
- Embracing a DevOps culture is key to building a high-performing engineering organization without sacrificing employee contentment. It's a win-win.
- For an engineer there is nothing like the feeling of frequently and easily deploying and running stable, high-performing software that makes its users happy, and for executives the improved business outcomes are a hit.

9 DevOps Barriers

Sinhgad Institutes

1.Culture: Many understand the importance of culture when it comes to DevOps. Few IT executives understand HOW to cultivate a successful DevOps culture.

2.Test Automation: Continuous testing a vital part of the DevOps model but often gets forgotten in the organized chaos. Implement AlertOps and set triggers to automatically kick-off tests

3.Legacy systems: Introducing new tools or systems into existing ones can be tough. Implement AlertOps, with an open API, for easy integration into all your systems allowing for enhanced visibility and incident response automation.

4.Application Complexity: Few IT executives account for application architecture changes during deployment processes.

5.No Plan: Many executives fail to develop a DevOps strategy or a set of goals to strive toward

9 DevOps Barriers

Sinhgad Institutes

6. Managing Environments: DevOps operations become complex when there is a lack of standardization and automation. Implement AlertOps into your DevOps model to automate processes such as alert escalation and streamline real-time collaboration

7. Skillset: Finding talented developers and ITOps professionals, capable of working together to achieve common goals, is a major hurdle for many organizations.

8.Budget: DevOps ultimately aims to reduce an organization's incident response cost, yet only a few take the time to develop a DevOps budget

9.Tools: Fragmented tool adoption limits the overall performance of a DevOps team. Implement AlertOps to easily integrate into processes and ensure teams don't have to work with cookie-cutter solutions.

Lecture 3

Process automation, Agile Practices

Process automation

Sinhgad Institutes

DevOps automation is the addition of technology that performs tasks with reduced human assistance to processes that facilitate feedback loops between operations and development teams so that iterative updates can be deployed faster to applications in production.

Automation is the use of technology to perform tasks with reduced human assistance. Automation helps you accelerate processes and scale environments, as well as build continuous integration, continuous delivery, and continuous deployment (CI/CD) workflows. There are many kinds of automation, including IT automation, business automation, robotic process automation, industrial automation, artificial intelligence, machine learning, and deep learning.

Process automation

Sinhgad Institutes

Examples of automation in DevOps include: Infrastructure-as-Code tools can automatically configure software environments based on configuration management files created beforehand. Release automation suites can build, test and deploy new versions of an application.

Why is automation important in DevOps?

Alongside automation, DevOps is able to revolutionize software development. DevOps practice is often very dependent on automation in order to make fast and frequent delivery. Automation in DevOps can allow more speed, accuracy, consistency, reliability in products.

Agile Practices

Sinhgad Institutes

What is agile?

Agile is an iterative approach to project management and software development that focuses on collaboration, customer feedback, and rapid releases. It arose in the early 2000s from the software development industry, helping development teams react and adapt to changing market conditions and customer demands.

In an agile approach, some planning and design is done upfront, but the development proceeds in small batches and involves close collaboration with stakeholders. Changes are incorporated continuously and a usable version of a product is often released quicker compared to products developed through the waterfall methodology. This provides many benefits, with arguably the most important being that if software doesn't meet the needs or expectations of the customer, it can be remediated in real-time.

Difference between Agile and DevOps

Sinhgad Institutes

Agile: Agile program advancement comprises different approaches to computer program improvement beneath which prerequisites and arrangements advance through the collaborative exertion of self-organizing and cross-functional groups and their customer/end client.
Basically, it breaks the product into two pieces and then integrates them for final testing. There are many ways through which it can be implemented, such as Kanban, XP, Scrum and many more.

DevOps: DevOps could be a set of hones that combines program improvement and information-technology operations which points to abbreviating the framework's advancement life cycle and giving nonstop conveyance with tall program quality.
Facebook's mobile app which is updated every two weeks effectively tells users you can have what you want and you can have it. Now ever wondered how Facebook was able to do social smoothing? It's the DevOps philosophy that helps facebook and sure that apps aren't outdated and that users get the best experience on Facebook. Facebook accomplishes this true code ownership model that makes its developers responsible that includes testing and supporting through production and delivery for each kernel of code. They write and update their true policies like this but Facebook has developed a DevOps culture and has successfully accelerated its development lifecycle.

When do agile and DevOps work together?

Sinhgad Institutes

DevOps can be thought of as an evolution of agile practices, or as a missing piece of agile. It's an effort to take the innovations of the agile approach and apply them to operations processes. At the same time, it's a missing piece of agile, because certain principles of agile are only realized in their most complete form when DevOps practices are employed. For example, there are multiple references to continuous delivery of software in agile documents, but because delivery pipelines encompass operations concerns, continuous delivery is usually regarded as a DevOps practice. Amplifying feedback loops requires improved communication across and between teams. Agile, specifically scrum, helps facilitate this communication through its various ceremonies such as daily standups, planning meetings, and retrospectives.

Agile Practices

Sinhgad Institutes

Ultimately the goals of agile and DevOps are the same: to improve the speed and quality of software development, and it makes very little sense to talk about one without the other. Many teams have found agile methodologies help them tremendously, while others have struggled to realize the benefits promised by an agile approach. This might be for any number of reasons, including teams not fully understanding or correctly implementing agile practices. It may also be that incorporating a DevOps approach will help fill the gaps for organizations that struggle with agile and help them have the success they were hoping for.

Lecture 4

Reason for adopting DevOps, What and Who Are Involved in DevOps? Changing the Coordination,

Reason for adopting DevOps

Sinhgad Institutes

Adopting DevOps practices improves the quality of your software when deploying new features and allows you to make changes rapidly. Continuous Integration and Continuous Deployment (CI/CD) is the practice where you make incremental changes to your code and quickly merge to the source code.

8 Reasons You Should Adopt DevOps

1. Efficient Development and Deployment Planning

2. Continuous Improvement and Software Delivery

3. Improves Software Security

4. Improves Customer Experiences

5. Better Collaboration Among Teams

6. DevOps Practices Create More Time to Innovate

7. DevOps Enhances Decision-Making

8. DevOps Encourages Higher Trust and Better

1. Efficient Development and Deployment Planning

Sinhgad Institutes

One of the biggest headaches of any IT Manager is managing cross-functional teams to develop and deploy software in good time. DevOps practices help you plan in advance how both teams will work cohesively to deliver products, meet customer needs and stay competitive.

DevOps methods such as **Scrum** and **Kanban** provide practices that define how teams work together. For example, Scrum practices include working in sprints, holding Scrum meetings, and designating time boxes.

These work to reduce blockers quickly and complete work within a fixed period before starting another project.

Kanban manages workflows by visualizing work on a **Kanban Board**, and there's real-time communication of your work's progress.

2. Continuous Improvement and Software Delivery

Sinhgad Institutes

Adopting DevOps practices improves the quality of your software when deploying new features and allows you to make changes rapidly.

Continuous Integration and **Continuous Deployment (CI/CD)** is the practice where you make incremental changes to your code and quickly merge to the source code. This DevOps practice delivers your software to the market faster while addressing customer issues rapidly.

3. Improves Software Security

Sinhgad Institutes

A subset of DevOps is **DevSecOps** which combines **development, security,** and **operations**. Adopting DevOps practices allows you to **build security** into your software continuously.

The IT security team is involved in the software development cycle from the start rather than at the deployment stage. Outdated security practices integrate infrastructure security features independently.

With DevOps, software security is a collaborative practice and is considered the foundation before any product development.

4. Improves Customer Experiences

A major benefit to adopting DevOps practices is it **lowers the failure rates** of new features while **improving recovery time**. The continuous deployment, testing, and feedback loop ensure faster service delivery and happier customers.

By automating the software pipeline, the development teams focus on creating superior products while the operations team can improve business delivery.

5. Better Collaboration Among Teams

Sinhgad Institutes

DevOps practices dismantle silos between teams and improve collaboration.

By adopting methods such as **Scrum** and **Kanban**, teams collaborate more efficiently, friction among colleagues is dealt with quickly, and communication flows seamlessly across the organisation.

Since DevOps discourages hierarchies among teams, everyone is responsible for software quality and speedy delivery. This approach **reduces** cases of **low-effort contributors** and **blame games**.

6. DevOps Practices Create More Time to Innovate

Sinhgad Institutes

DevOps automates **repetitive tasks**, improves **service delivery**, and reduces the **software development cycle**. The methodology frees up time for teams to develop new products and better features.

Unfortunately, inefficient teamwork leads to time wasted fixing errors rather than innovating. Building better products and services leads to a competitive advantage.

Companies that innovate continuously are also better able to react to market changes and take advantage of new opportunities.

7. DevOps Enhances Decision-Making

Sinhgad Institutes

DevOps helps you leverage data to **make better decisions** and **removes hierarchies** that require layers of approvals.

In addition, DevOps practices **reduce time spent** between design and execution. **Scrum** and the **Kanban** approach address blockers as they come, manage workflows efficiently, and tackle projects in sprints.

Fast decision-making based on accurate data is a competitive advantage. Slow decision-making delays go-to-market plans and throttles the implementation of new ideas.

According to a **McKinsey Global Survey**, only **20% of organizations** are good at **decision-making,** demonstrating **room for improvement by adopting DevOps**.

8. DevOps Encourages Higher Trust and Better Collaboration

Sinhgad Institutes

Due to improved collaboration across development and IT operations, teams trust each other more and produce higher quality work. Higher trust leads to improved morale and better productivity.

Conversely, teams working in silos mistrust each other's new suggestions and ideas.

A culture of trust and collaboration creates transparency in workflows and gives visibility into projects. This approach leads to better development and deployment planning.

The net effect is a collaborative environment that works in unity, solves problems faster, and delivers superior products to the market.

What and Who Are Involved in DevOps?

Sinhgad Institutes

A DevOps team includes developers and IT operations working collaboratively throughout the product lifecycle, in order to increase the speed and quality of software deployment. It's a new way of working, a cultural shift, that has significant implications for teams and the organizations they work for.

In addition to a lead change agent to oversee the transformation to DevOps, these roles include release managers to handle adaptive release governance, automation architects, developer-testers, experience assurance experts, security engineers, and utility technology players who understand development and operations. testing on an agile DevOps project is done by the whole team, including developers, testers and operations people. In the DevOps model, testing is done continuously—early, often and even after applications are in production.

Sinhgad Institutes

Lecture 5

Introduction to DevOps pipeline phases, Defining the Development Pipeline

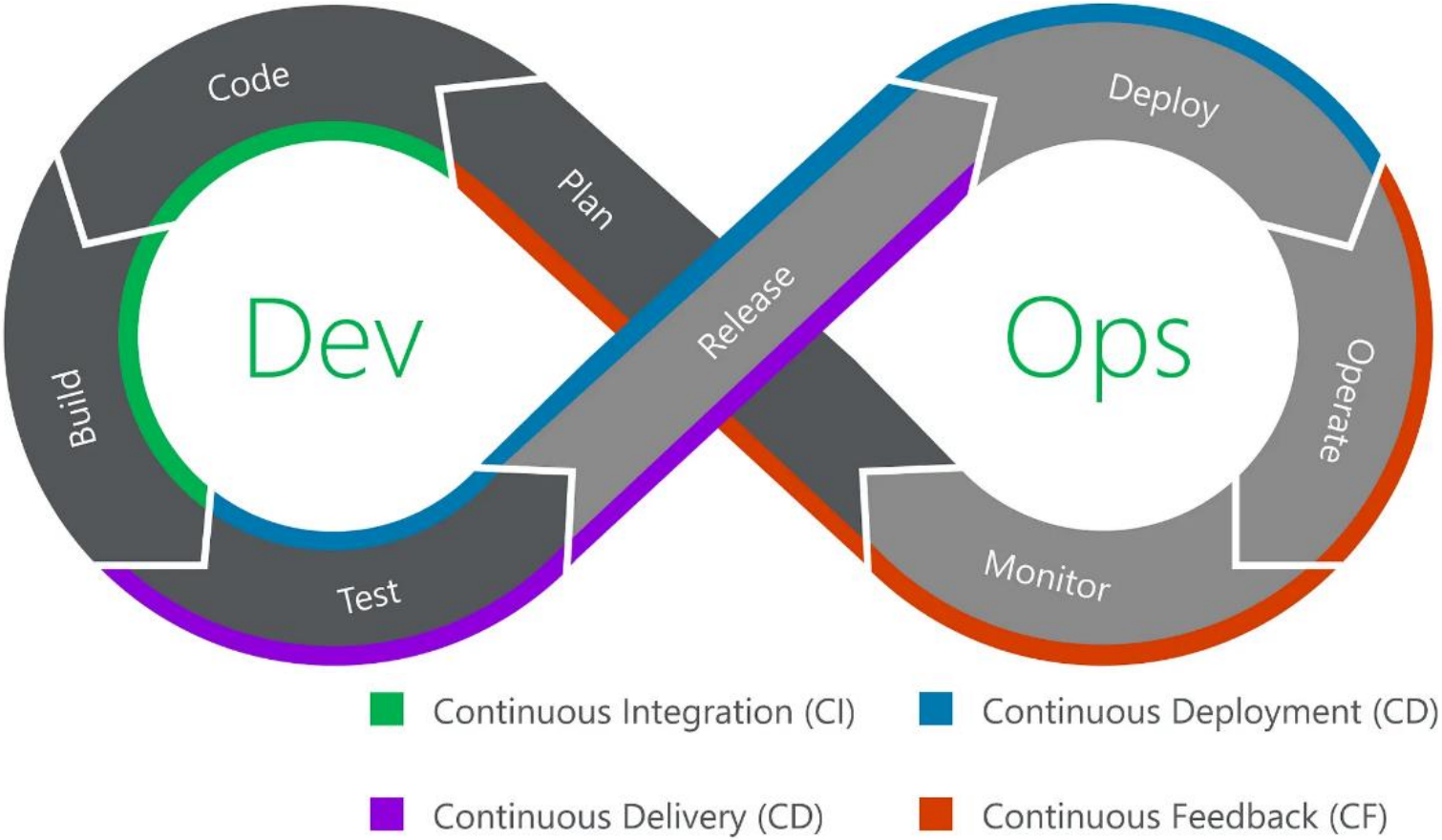
Introduction to DevOps pipeline phases

Sinhgad Institutes

The DevOps pipeline typically has eight stages.

In the Development phase, they are: plan, code, build, and test.

In the Operations phase, the stages are: release, deploy, operate, and monitor



1. Plan

The Plan stage covers everything that happens before the developers start writing code, and it's where a Product Manager or Project Manager earns their keep. Requirements and feedback are gathered from stakeholders and customers and used to build a product roadmap to guide future development. The product roadmap can be recorded and tracked using a ticket management system such as Jira, Azure DevOps or Asana which provide a variety of tools that help track project progress, issues and milestones.

The product roadmap can be broken down into Epics, Features and User Stories, creating a backlog of tasks that lead directly to the customers' requirements. The tasks on the backlog can then be used to plan sprints and allocate tasks to the team to begin development.

2.Code

Sinhgad Institutes

Once the team had grabbed their coffees and had the morning stand-up, the developments can get to work. In addition to the standard toolkit of a software developer, the team has a standard set of plugins installed in their development environments to aid the development process, help enforce consistent code-styling and avoid common security flaws and code anti-patterns.

This helps to teach developers good coding practice while aiding collaboration by providing some consistency to the codebase. These tools also help resolve issues that may fail tests later in the pipeline, resulting in fewer failed builds.

3.Build

Sinhgad Institutes

The Build phase is where DevOps really kicks in. Once a developer has finished a task, they commit their code to a shared code repository. There are many ways this can be done, but typically the developer submits a pull request — a request to merge their new code with the shared codebase. Another developer then reviews the changes they've made, and once they're happy there are no issues, they approve the pull-request. This manual review is supposed to be quick and lightweight, but it's effective at identifying issues early.

Simultaneously, the pull request triggers an automated process which builds the codebase and runs a series of end-to-end, integration and unit tests to identify any regressions. If the build fails, or any of the tests fail, the pull-request fails and the developer is notified to resolve the issue. By continuously checking code changes into a shared repository and running builds and tests, we can minimize integration issues that arise when working on a shared codebase, and highlight breaking bugs early in the development lifecycle.

4.Test

Sinhgad Institutes

Once a build succeeds, it is automatically deployed to a staging environment for deeper, out-of-band testing. The staging environment may be an existing hosting service, or it could be a new environment provisioned as part of the deployment process. This practice of automatically provisioning a new environment at the time of deployment is referred to as Infrastructure-as-Code (IaC) and is a core part of many DevOps pipelines. More on that in a later article.

Once the application is deployed to the test environment, a series of manual and automated tests are performed. Manual testing can be traditional User Acceptance Testing (UAT) where people use the application as the customer would to highlight any issues or refinements that should be addressed before deploying into production.

At the same time, automated tests might run security scanning against the application, check for changes to the infrastructure and compliance with hardening best-practices, test the performance of the application or run load testing. The testing that is performed during this phase is up to the organisation and what is relevant to the application, but this stage can be considered a test-bed that lets you plug in new testing without interrupting the flow of developers or impacting the production environment.

5.Release

Sinhgad Institutes

The Release phase is a milestone in a DevOps pipeline — it's the point at which we say a build is ready for deployment into the production environment. By this stage, each code change has passed a series of manual and automated tests, and the operations team can be confident that breaking issues and regressions are unlikely.

Depending on the DevOps maturity of an organisation, they may choose to automatically deploy any build that makes it to this stage of the pipeline. Developers can use feature flags to turn off new features so they can't be seen by the customers until they are ready for action. This model is considered the nirvana of DevOps and is how organizations manage to deploy multiple releases of their products every day.

Alternatively, an organisation may want to have control over when builds are released to production. They may want to have a regular release schedule or only release new features once a milestone is met. You can add a manual approval process at the release stage which only allows certain people within an organisation to authorize a release into production.

The tooling lets you customize this, it’s up to you how you want to go about things.

6.Deploy

Sinhgad Institutes

Finally, a build is ready for the big time and it is released into production. There are several tools and processes that can automate the release process to make releases reliable with no outage window.

The same Infrastructure-as-Code that built the test environment can be configured to build the production environment. We already know that the test environment was built successfully, so we can rest assured that the production release will go off without a hitch.

A blue-green deployment lets us switch to the new production environment with no outage. Then the new environment is built, it sits alongside the existing production environment. When the new environment is ready, the hosting service points all new requests to the new environment. If at any point, an issue is found with the new build, you can simply tell the hosting service to point requests back to the old environment while you come up with a fix.

7.Operate

Sinhgad Institutes

The new release is now live and being used by the customers.

The operations team is now hard at work, making sure that everything is running smoothly. Based on the configuration of the hosting service, the environment automatically scales with load to handle peaks and troughs in the number of active users.

The organisation has also built a way for their customers to provide feedback on their service, as well as tooling that helps collect and triage this feedback to help shape the future development of the product. This feedback loop is important — nobody knows what they want more than the customer, and the customer is the world's best testing team, donating many more hours to testing the application than the DevOps pipeline ever could. You need to capture this information, it's worth its weight in gold.

8.Monitor

Sinhgad Institutes

The ‘final’ phase of the DevOps cycle is to monitor the environment. this builds on the customer feedback provided in the Operate phase by collecting data and providing analytics on customer behavior, performance, errors and more.

We can also do some introspection and monitor the DevOps pipeline itself, monitoring for potential bottlenecks in the pipeline which are causing frustration or impacting the productivity of the development and operations teams.

All of this information is then fed back to the Product Manager and the development team to close the loop on the process. It would be easy to say this is where the loop starts again, but the reality is that this process is continuous. There is no start or end, just the continuous evolution of a product throughout its lifespan, which only ends when people move on or don’t need it any more.

Continuous Integration

Sinhgad Institutes

One of the biggest difficulties in coordinating a software development team is managing the collaboration of many developers, often in remote locations, on a single codebase. A shared code repository is key to solving this problem, however, there can still be issues in when merging the changes made by multiple people on the same piece of code.

A change made by one developer may impact what somebody else is working on, and the longer that developers wait to integrate their changes back into the shared codebase, the bigger the drift, resulting in more effort and headache in resolving the issues and conflicts. Continuous integration aligns with the Code and Build phases of the DevOps pipeline. It's the practice of regularly merging a developer's code into the centralised codebase and conducting automated testing to ensure that no regressions have been introduced. By merging smaller changes more regularly, these issues become smaller and easier to manage, improving overall productivity and sanity.

Continuous Delivery

Sinhgad Institutes

Continuous Delivery is an extension of Continuous Integration which automates the process of deploying a new build into production.

The goals of Continuous Delivery is to:

1.Perform automated testing on each new build to verify builds that are ready for release into production, and fail those which are not.

2.Manage the automatic provisioning and configuration of deployment environments, as well as testing of these environments for stability, performance and security compliance.

3.Deploy a new release into production when approved and manually triggered by the organisation.

Continuous Delivery aligns with the Test and Release phases of the pipeline, and allows organisations to manually trigger the release of new builds as regularly as they choose.

Continuous Deployment

Sinhgad Institutes

Continuous Deployment is a more advanced version of Continous Delivery (which makes the reuse of the ‘CD’ abbreviation more acceptable). The goals are the same, but the manual step of approving new releases into production is removed. In a Continuous Deployment model, each build which passes all of the checks and balances of the pipeline are automatically deployed into production.

Continuous Feedback

Sinhgad Institutes

CI and CD tend to get the glory when people talk about DevOps, but an equally important factor is Continuous Feedback. The whole point of DevOps is to release new features and fixes as quickly as possible so that the organisation can get feedback from customers, stakeholders and analytics as quickly as possible to make better decisions when designing the next set of changes. The whole point is to achieve a strong Continous Feedback loop to develop a better product.

It's Continuous Feedback that ties the ends of the loop together, feeding back data and analytics from the Operate and Monitor phases back into the Plan phase to do it all over again.

Lecture 6

Centralizing the Building Server, Monitoring Best Practices, Best Practices for Operations.

Centralizing the Building Server

Sinhgad Institutes

Centralizing the building server is crucial for building the correct pipeline. When we design a DevOps architecture, we must think of reducing points of failure. When we adopt a build server, we centralize everything in one server. This means that we use a different software to release our new software. Having only one server, or cluster, for building new software means that there is only one point of failure. Any problem is centralized at only one point. This reduces the cost of maintaining the software and speeds up the release process.

The building server is normally connected with an artifact repository server. The repository server is where the build is stored and saved. It is associated with continuous release. Essentially, with this practice, we build the software every time and release it to a server. This server essentially maintains the different versions of the software we build on the server.

Centralizing the Building Server

Sinhgad Institutes

Normally, we establish a naming policy to maintain the different software versions. This is because we want to identify every version of the software uniquely. With the artifact server, we can easily centralize one point for release of the software. In this way, we can have different versions of the same software, and, if we use Docker, we can have different versions on the same server at the same time. We can also start them at the same time, with some minor adjustments. This allows the QA engineer,

For example, to undertake some regression test, and in case of new errors, identify exactly what version has the bug. This allows the developer to understand exactly what change to the code introduced the error.

Monitoring Best Practices

Sinhgad Institutes

To be effective, monitoring must be combined with some other practice. A log analysis is the most important practice for preventing errors and understanding how the system functions. Some software is required for analyzing the log and making related predictions.

The software most commonly used is ELK (Elasticsearch, Logstash, and Kibana).This ecosystem is helpful because it gives a complete log analysis system, not only providing alerts, but also a graphical representation of the error and the log. Log analysis is very important for improving the quality of software. One important practice we can put in place is to have some software that not only identifies the number of errors but graphs these as well

Having a graphical representation of the errors is important for providing visible feedback about the software, without the necessity of reading a log, in order to understand the status of the software

Monitoring Best Practices

Sinhgad Institutes

Monitoring is the backbone for every DevOps practice, and if we want a very successful journey, we must be sure to have a good monitoring system. At the same time, we must start to monitor not only the production but possibly the canary server. This is because it can reveal an error, and we can solve it before release to production. Monitoring can take two forms. Black-box monitoring tests a piece of code as if it were in a black box. It reveals only the status of the system, to determine whether it is alive. It doesn't really indicate what is happening inside, because the monitoring is external.

An example of black-box software monitoring is Nagios.

The opposite of this is white-box monitoring. This type of monitoring provides a clear picture of the inside of the system, such as, for example, the number of HTTP connections open, the number of errors, etc. Prometheus is an example of white-box monitoring software.

Best Practices for Operations

Sinhgad Institutes

In DevOps, the operations team has a big influence on achieving the best results. The importance of the operations team is strictly connected to the quality of the software and what the customer thinks about the company.

In case of an error, the operations team is the first face of the company. This team is normally delegated to maintain the software in the production environment. The only point of contact with the software is the log. For this reason, some member of the operations team must be included when software is designed, and, more important is the feedback they can provide when software is released for testing. This is because if the log is insufficient, the operations team can't really identify the error, which means more time will be required to fix the issue.

Best Practices for Operations

Sinhgad Institutes

At the same time, the operations team can help to identify common issues and provide documentation to solve them faster. This documentation is actually a step toward resolving the issue. It is used essentially by first-line operations engineers. It is “live,” meaning that it is never closed and must be carefully managed, so that it aligns with the most recent software updates.

The documentation must indicate common errors in the log and show how to solve the root cause(s) of the problem. This documentation should be written with people who don’t know the system in mind and, based on that, must provide specific details about the appropriate steps to be taken. Another operations practice we can put in place is developer on-call. This practice introduces a new figure to the operations world. he developer on-call is essentially a software engineer, working with the operations professionals to resolve errors in production.

Best Practices for Operations

Sinhgad Institutes

This has two principal advantages. The first is a reduction in the time it takes to identify and fix an issue. Because one of the developers works on the issue, he/she can easily identify what's gone wrong and what part of the code creates the issue. This can drive the operations team's efforts to fix it.

The second advantage is improving the level of responsibility. Because the developer works to fix a live issue, he/she understands better what's wrong with the software and thus can improve the way he/she writes the software and the log, because a bad log can result in more work for him/her in future.