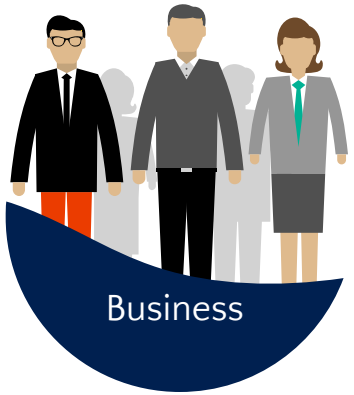
UNIT I-DevOps

INT 331

WHAT IS DEVOPS?

"DevOps is the next step in the evolution of Agile and ALM"



Agile:
How do I **develop** the "**right**" software?

DevOps:
How do I **deliver** software **faster**?

ALM:
How do I **develop** software with **quality**?

WHAT IS DEVOPS?

"The seven habits of effective DevOps"

Microsoft Development Division

DELIVERON

What does DevOps look like?

WHAT DOES DEVOPS LOOK LIKE?

The shift to DevOps

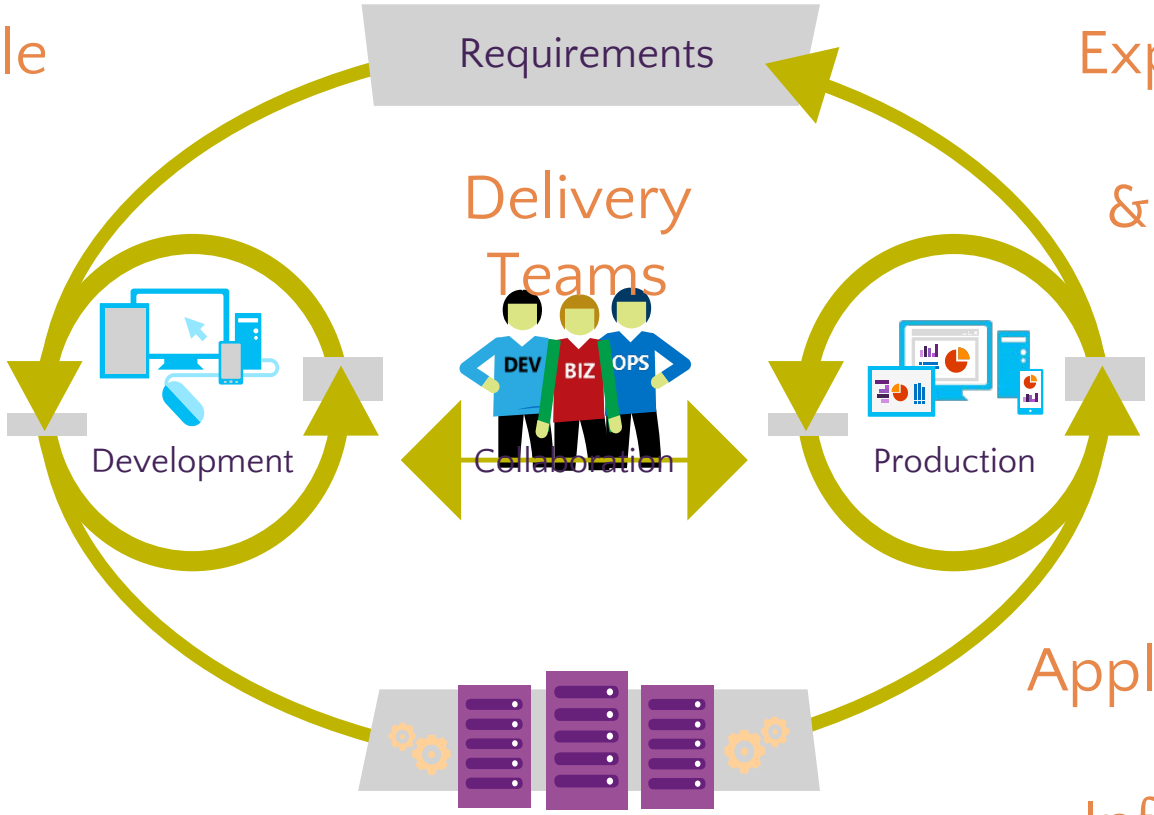
OLD WORLD	NEW WORLD
Focus on **planning**	Focus on **delivering**
Compete, not collaborate	**Collaborate** to win
Static hierarchies	Fluent and **flexible** teams
Individual **productivity**	Collective **value** creation
Efficiency of process	**Effectiveness** of outcomes
Assumptions, not data	Experiment, **learn** and respond
Estimating performance	**Measuring** performance

WHAT DOES DEVOPS LOOK LIKE?

Unified Backlog

Operational Deliverables

Production Experimentation & Monitoring



Feedback Loops

Application-driven Infrastructure

DELIVERON

WHAT DOES DEVOPS LOOK LIKE?

What does it mean for me?

Business Teams

Tech Debt Matters

Learn from Customers

Software is never done

Developers

You build it, you run it

Code for operations

Testing is for everyone

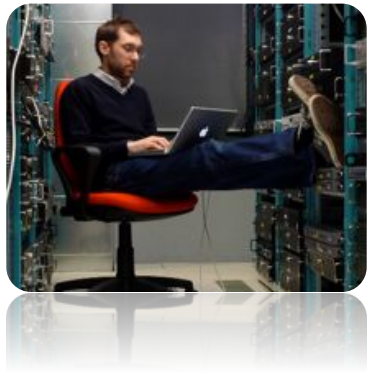
Testers

Automation is a must

Test quality not just quantity

Test data must be part of the strategy

Operations



Apps drive infrastructure

Scripting is tool of choice

We own customer experience too

DevOps

- DevOps is the **combination of cultural philosophies, practices, and tools** that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes

Industry Importance of DevOps

- DevOps **helps in automating the manufacturing processes and ensuring robustness in operations**.

- With DevOps ensuring continuous integration of activities, it helps business owners and decision makers to focus on developing new products as the daily disturbances of production are taken over by the existing teams.

- **Robustness** means the quality or condition of being strong and in good condition.

Industry Importance of DevOps

- Maximizes Efficiency with Automation

- The late DevOps authority Robert Stroud said DevOps is all about "fueling business transformation"

  that involves people, process and culture change. The most effective strategies for DevOps transformation focus on structural improvements that build community.

  A successful DevOps initiative requires a culture—or mindset—change that brings greater collaboration between multiple teams—product, engineering, security, IT, operations and so on—as well as automation to better achieve business goals.

- By managing engineering processes end to end, DevOps emphasizes deploying software more often, in a reliable and secure way through automation.

- End-to-end describes a process that **takes a system or service from beginning to end and delivers a complete functional solution**, usually without needing to obtain anything from a third party.

Industry Importance of DevOps

- Optimizes the Entire Business

- System architect Patrick Debois, best known as the creator of the DevOps movement, says the biggest advantage of DevOps is the vision it provides.

It forces organizations to "optimize for the whole system," not just IT siloes, to improve the business as a whole.

In other words, be more adaptive and data-driven for alignment with customer and business needs.

Industry Importance of DevOps

- Improves Speed and Stability of Software Development and Deployment
- A multi-year analysis in the annual Accelerate State of DevOps Report has found that top-performing DevOps organizations do far better on software development/deployment speed and stability, and also achieve the key operational requirement of ensuring that their product or service is available to end users..

Industry Importance of DevOps

Gets You to Focus on What Matters Most: People

- People, not tools, are the most important component of a DevOps initiative.
- Key roleplayers (i.e., humans) can greatly increase your odds of success, such as a **DevOps evangelist,** a persuasive leader who can explain the business benefits brought by the greater agility of
- DevOps practices and eradicate misconceptions and fears. And since automated systems are crucial to DevOps success, an automation specialist can develop strategies for continuous integration and deployment, ensuring that production and pre-production systems are fully software-defined, flexible, adaptable and highly available.

Challenges of DevOps

- There are many challenges in a DevOps initiative. Your organization must reimagine its structure to improve the way things get done. Companies often underestimate the amount of work required in a DevOps transformation, though. According to a recent Gartner study, 75% of DevOps initiatives through 2020 will fail to meet their goals due to issues around organizational learning and change.
- “Organizational learning and change are key to allowing DevOps to flourish. In other words, people-related factors tend to be the greatest challenges — not technology, ” says Gartner senior analyst George Spafford.
- Choosing the Right Metrics is Hard
- Enterprises transitioning to DevOps practices need to use metrics to recognize progress, document success, and uncover areas that need improvement,
- Forrester notes. For example, an acceleration in deployment velocity without a corresponding improvement in quality is not a success. An effective DevOps effort needs metrics that drive smart automation decisions—and yet organizations often struggle with DevOps metrics.
- So where to start? Find metrics that align with velocity and throughput success.

Challenges of DevOps

- Limited Funds
- DevOps initiatives face other obstacles as well. Given the significant organizational and IT changes involved—with previously siloed teams joining forces, changing job roles, and encountering other transitions—adjustments will take time. According to a survey of IT executives from software company Pensa, the top challenges to DevOps success are:
- Limited budgets (cited by 19.7% of respondents)
- Legacy systems (17.2%)
- Application complexity (12.8%)
- Difficulty managing multiple environments (11.3%)
- Company culture (9.4%)
- Complexity

The Future of DevOps

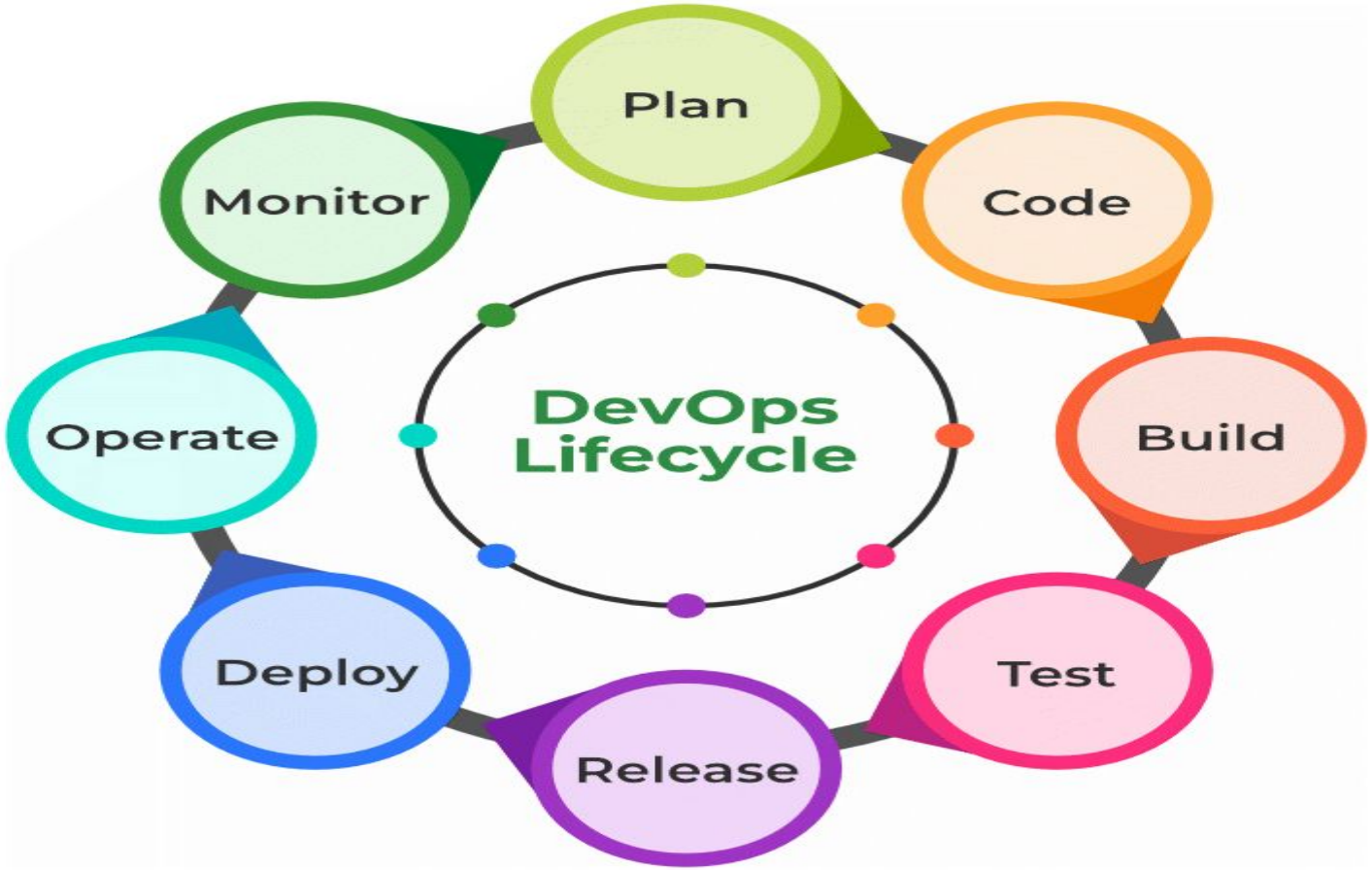
- The future of DevOps will likely bring changes in tooling and organizational strategies, but its core mission will remain the same
- Automation Will Play a Major Role
- Automation will continue to play a major role in DevOps transformation, and artificial intelligence for IT operations—AIOps—will help organizations achieve their DevOps goals. The core elements of AIOps—machine learning, performance baselining, anomaly detection, automated root cause analysis (RCA) and predictive insights—work together to accelerate routine operational tasks. This emerging technology, which can transform how IT operations teams manage alerts and resolve issues, will be a crucial component of the future of DevOps.
- AIOps Will Make Service Uptime Easier to Achieve
- In addition to using data science and computational techniques to automate mundane tasks, AIOps also ingests metrics and uses inference models to pull actionable insights from data, notes data science architect Jiayi Hoffman. AIOps' automation capabilities can make service uptime much easier to achieve, from monitoring to alerting to remediation. And AIOps is a boon for DevOps teams, who can use AIOps tools for real-time analysis of event streams, proactive detection to reduce downtime, improved collaboration, faster deployments, and more.
- Will Sharpen Focus on Cloud Optimization
- The future of DevOps will also bring a greater focus on optimizing the use of cloud technologies. The centralized nature of the cloud provides DevOps automation with a standard platform for testing, deployment, and production notes Deloitte Consulting analyst David Linthicum.
- And regardless of what advanced technologies the future brings, organizations will need to realize that DevOps is all about the journey and that the organization's DevOps-related goals and expectations will evolve over time.

DevOps Lifecycle

- DevOps defines an agile relationship between operations and Development. It is a process that is practiced by the development team and operational engineers together from beginning to the final stage of the product.

DevOps Lifecycle : DevOps lifecycle is the methodology where professional development teams come together to bring products to market more efficiently and quickly.

The structure of the DevOps lifecycle consists of Plan, Code, Building, Test, Releasing, Deploying, Operating, and Monitoring.



DevOps Lifecycle in Detail.

- **Plan:** Determining the commercial needs and gathering the opinions of end-user by professionals in this level of the DevOps lifecycle.
- **Code:** At this level, the code for the same is developed and in order to simplify the design, the team of developers uses tools and extensions that take care of security problems.
- **Build:** After the coding part, programmers use various tools for the submission of the code to the common code source.
- **Test:** This level is very important to assure software integrity. Various sorts of tests are done such as user acceptability testing, safety testing, speed testing, and many more.
- **Release:** At this level, everything is ready to be deployed in the operational environment.
- **Deploy:** In this level, Infrastructure-as-Code assists in creating the operational infrastructure and subsequently publishes the build using various DevOps lifecycle tools.
- **Operate:** At this level, the available version is ready for users to use. Here, the department looks after the server configuration and deployment.
- **Monitor:** The observation is done at this level that depends on the data which is gathered from consumer behavior, the efficiency of applications, and from various other sources.

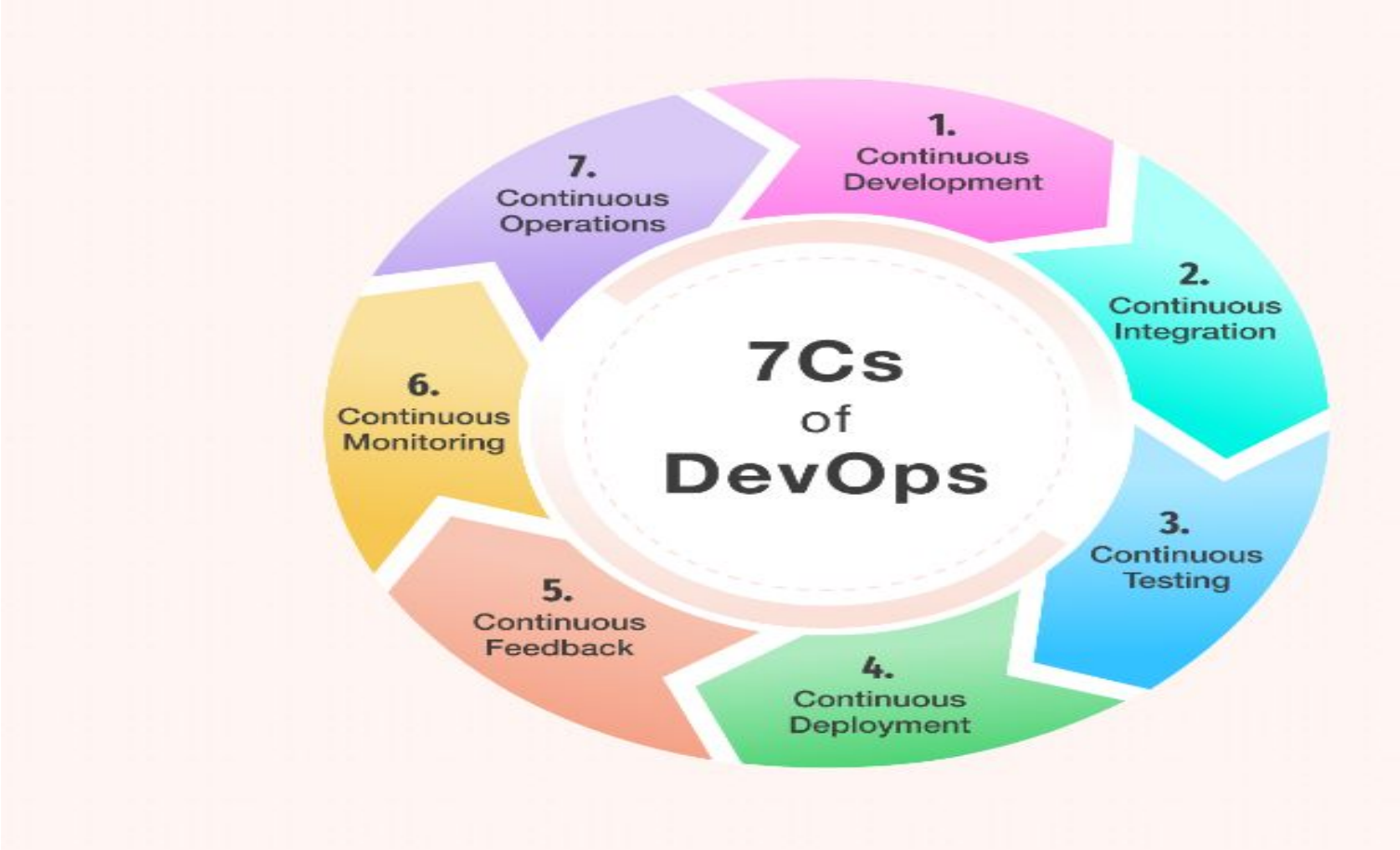
Important Links

- https://www.youtube.com/watch?v=to5H5iPZmD4
- https://www.youtube.com/watch?v=dJAxKm8La88

DevOps lifecycle phases: the 7Cs of DevOps lifecycle

- As we have discussed in Lifecycle that everything is continuous in DevOps – from **planning to monitoring**.
- So let's break down the entire **lifecycle into seven phases** where continuity is at its core.
- Any phase in the lifecycle can iterate throughout the projects multiple times until it's finished.

7 C’s of DevOps



1. Continuous Development

This phase plays a pivotal role in delineating the vision for the entire software development cycle.

It primarily focuses on project planning and coding.

During this phase, project requirements are gathered and discussed with stakeholders.

Moreover, the product backlog is also maintained based on customer feedback which is broken down into smaller releases and milestones for continuous software development.

Once the team agrees upon the business needs, the development team starts coding for the desired requirements.

It’s a continuous process where developers are required to code whenever any changes occur in the project requirement or in case of any performance issues.

2. Continuous integration

- Continuous integration is the most crucial phase in the entire DevOps lifecycle.

- In this phase, **updated code** or add-on functionalities and features are developed and integrated into existing code. Furthermore, bugs are detected and identified in the code during this phase at every step through **unit testing**, and then the source code is modified accordingly.

- This step makes integration a continuous approach where code is tested at **every commit**. Moreover, the tests needed are also planned in this phase.

CASE STUDY

- Let’s take the example of Docusign, which developed e-signature technology back in 2003.
- It helps its clients automate the process of preparing, signing, and managing agreements.
- Their development teams used to follow Agile methodology for years to collect customer feedback and make small and quick releases.
- But, they lacked collaboration between the development and operations team, which led them to many failures.
- Moreover, their business was solely based on the transaction of signatures and approvals.
- So, the biggest challenge for their business was continuous integration and delivery.
- A single mistake could cause a serious problem and ruin the entire operation process. Hence, the organization decided to move to DevOps. DocuSign implemented a tool – mock for their internal API to speed up the product development and delivery.

- This tool helped the organization in integrating critical functionalities such as incident management. This tool also makes the testing with simulation simple.

Tools Used in Continuous integration

- Jenkin, Bamboo, GitLab CI, Buddy, TeamCity, Travis, and CircleCI are a few DevOps tools used to make the project workflow smooth and more productive.

- For example, Jenkin (open-source tool) is used widely to automate builds and tests.
- CircleCI and Buddy, on the other hand, are commercial tools.

Well, whatever tools you select for continuous integration, pick the one that can fit your business and project requirements.

Continuous Testing

- Some teams carry out the continuous testing phase before the integration occurs, while others do it after the integration.
- Quality analysts continuously test the software for bugs and issues during this stage using **Docker containers**.
- In case of a bug or an error, the code is sent back to the integration phase for modification.
- Automation testing also reduces the time and effort to deliver quality results.
- Teams use tools like ***Selenium*** at this stage. Moreover, continuous testing enhances the test evaluation report and minimizes the provisioning and maintenance cost of the test environments

Tools Used in Continuous Testing

- JUnit, Selenium, TestNG, and TestSigma are a few DevOps tools for continuous testing.

- **Selenium** is the most popular open-source automation testing tool that supports multiple platforms and browsers.

- **TestSigma**, on the other hand, is a unified AI-driven test automation platform that eliminates the technical complexity of test automation through **artificial intelligence**.

Continuous Deployment

- This phase is the crucial and most active one in the DevOps lifecycle, where final code is deployed on production servers.
- The continuous deployment includes configuration management to make the deployment of code on servers accurate and smooth.
- Development teams release the code to servers and schedule the updates for servers, keeping the configurations consistent throughout the production process.
- Containerization tools also help in the deployment process by providing consistency across development, testing, production, and **staging environments**.
- This practice made the continuous delivery of new features in production possible.

Tools Used

- Ansible, Puppet, and Chef are the configuration management tools that make the deployment process smooth and consistent throughout the production process.

- Docker and Vagrant are another DevOps tool used widely for handling the scalability of the continuous deployment process.

- Apart from this, Spinnaker is an open-source continuous delivery platform for releasing the software changes, while ArgoCD is another open-source tool for Kubernetes native CI/CD.

Continuous Feedback

Continuous feedback came into existence to analyze and improve the application code.

During this phase, customer behavior is evaluated regularly on each release to improve future releases and deployments.

Businesses can either opt for a structural or unstructured approach to gather feedback.

In the structural approach, feedback is collected through surveys and questionnaires.

In contrast, the feedback is received through social media platforms in an unstructured approach.

Overall, this phase is quintessential in making continuous delivery possible to introduce a **better version** of the application.

CASE STUDY

- One of the examples of continuous feedback is Tangerine bank.

It's a Canadian bank that embraced continuous feedback to enhance its customers' mobile experience.

After opting for continuous feedback, this Canadian bank collected a considerable amount of valuable feedback within a few weeks, which helped the bank reach the cause of the problem quickly.

Furthermore, this has helped them improve the application as per their customers' needs.

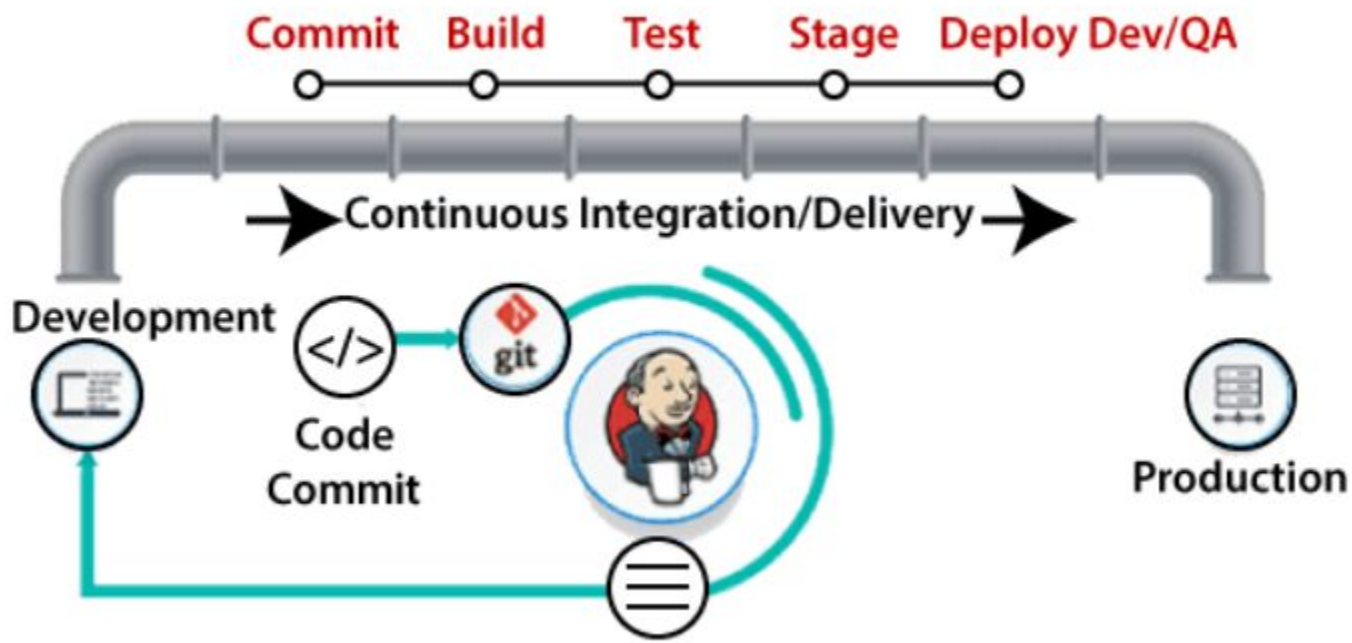
This is how Tangerine bank managed to repurpose the resources and money on other crucial things excellently after adopting DevOps.

Tools Used:

Pendo is a product analytics tool used to collect customer reviews and insights.

Qentelli's TED is another tool used primarily for tracking the entire DevOps process to gather actionable insights for bugs and flaws.

- The code supporting new functionality is continuously integrated with the existing code. Therefore, there is continuous development of software. The updated code needs to be integrated continuously and smoothly with the systems to reflect changes to the end-users.
- Jenkins is a popular tool used in this phase. Whenever there is a change in the Git repository, then Jenkins fetches the updated code and prepares a build of that code, which is an executable file in the form of war or jar. Then this build is forwarded to the test server or the production server.


Commit
Build
Test
Stage
Deploy Dev/QA
Continuous Integration/Delivery
Development
</>
git
Code
Commit
Production

- 3) Continuous Testing
- This phase, where the developed software is continuously testing for bugs. For constant testing, automation testing tools such as **TestNG, JUnit, Selenium**, etc are used. These tools allow QAs to test multiple code-bases thoroughly in parallel to ensure that there is no flaw in the functionality. In this phase, **Docker** Containers can be used for simulating the test environment.

- **Selenium** does the automation testing, and TestNG generates the reports. This entire testing phase can automate with the help of a Continuous Integration tool called **Jenkins**.
- Automation testing saves a lot of time and effort for executing the tests instead of doing this manually. Apart from that, report generation is a big plus. The task of evaluating the test cases that failed in a test suite gets simpler. Also, we can schedule the execution of the test cases at predefined times. After testing, the code is continuously integrated with the existing code.


Selenium
Test Scripts
TestNG
TEST REPORTS
JENKINS

- Some developers carry out the continuous testing phase prior to the continuous integration phase. Based on the updations in the application code, this phase can be repositioned around the continuous integration phase in the DevOps lifecycle. Here, the developed software is continuously tested for bugs. A test environment is simulated with the use of Docker containers.
- Through automated testing, developers save effort and time, usually lost in manual testing. Reports generated by automated testing improve the test evaluation process. Analyzing the failed test-cases becomes easy. After going through a UAT (User Acceptance Testing) process, the resultant testsuite is simpler and bug-free. TestNG, Selenium and JUnit are some of the DevOps tools used for automated testing. These tools can also arrange test-case execution in a preset timeline

- 4) Continuous Monitoring
- Monitoring is a phase that involves all the operational factors of the entire DevOps process, where important information about the use of the software is recorded and carefully processed to find out trends and identify problem areas. Usually, the monitoring is integrated within the operational capabilities of the software application.
- It may occur in the form of documentation files or maybe produce large-scale data about the application parameters when it is in a continuous use position. The system errors such as server not reachable, low memory, etc are resolved in this phase. It maintains the security and availability of the service.

- Monitoring the performance of an application is of key importance for application developers. In this phase, developers record data on the use of application and continuously monitor each functionality.
- “Server not reachable” or “low memory” are some of the common system errors resolved in this phase. Continuous monitoring helps in sustaining the availability of services in the application. It also determines the threats and root causes of recurring system errors. Security issues get resolved and problems are automatically detected and fixed.

- Compared to the software development teams, the IT operations teams are more involved in this phase. Their role is pivotal in supervising user activity, checking the system for unusual behavior, and tracing the presence of bugs.
- Sensu, ELK Stack, NewRelic, Splunk and Nagios are the key DevOps tools used in continuous monitoring. These tools enable complete control in overseeing the performance of the system, the production server, and the application. The operations team can actively engage in increasing the reliability and productivity of the applications with the help of these tools.

- When major issues are detected in this phase, the application is swiftly rerun through all the earlier phases of the DevOps lifecycle. That is how finding a resolution to all sorts of issues becomes faster in this phase
- The last phase of the DevOps life cycle is the shortest phase and the least complicated one. The purpose of continuous operation is to automate the process of releasing the application and the subsequent updates. Development cycles in continuous operations are shorter, allowing developers to ongoing accelerate the time-to-market for the application.

5) Continuous Feedback

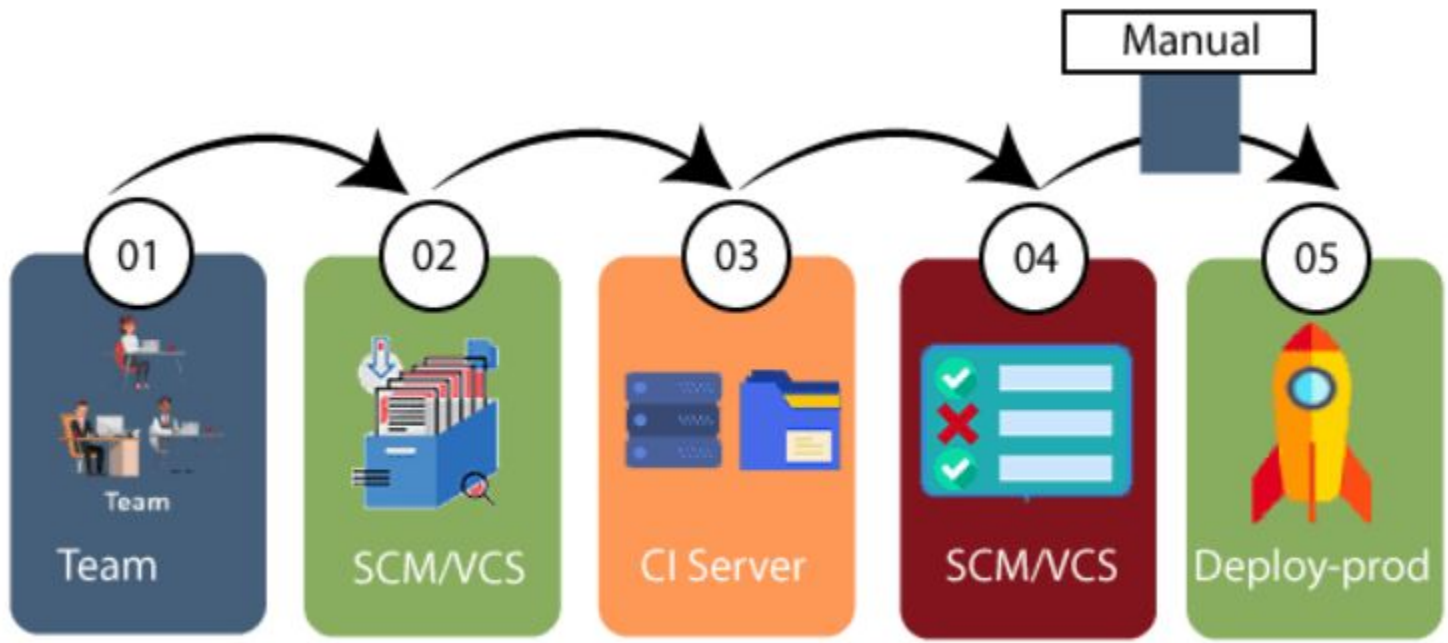
- The application development is consistently improved by analyzing the results from the operations of the **software. This is carried out by placing the critical phase of constant feedback between the operations and the development** of the next version of the current software application.

- The continuity is the essential factor in the DevOps as it removes the unnecessary steps which are required to take a software application from development, using it to find out its issues and then producing a better version. It kills the efficiency that may be possible with the app and reduce the number of interested customers.

- Continuous testing and continuous integration are the two crucial phases that ensure consistent improvements in the application code. Continuous feedback is a peculiar phase where these improvements are analyzed.
- Developers can scale the outcome of these modifications on the final product. Most importantly, customers who tested these applications can share their experiences in this phase. In a majority of cases, this phase of the DevOps lifecycle provides a turning point to the application development process.

- The feedback is assessed promptly and developers begin working on the new changes. Sooner, there is a positive response in customer feedback, which covers the way for releasing new versions of the software application.

- 6) Continuous Deployment
- In this phase, the code is deployed to the production servers. Also, it is essential to ensure that the code is correctly used on all the servers.


Manual
01
02
03
04
05
Team
Team
SCM/VCS
CI Server
SCM/VCS
Deploy-prod

- The new code is deployed continuously, and configuration management tools play an essential role in executing tasks frequently and quickly. Here are some popular tools which are used in this phase, such as **Chef, Puppet, Ansible**, and **SaltStack**.
- Containerization tools are also playing an essential role in the deployment phase. **Vagrant** and **Docker** are popular tools that are used for this purpose. These tools help to produce consistency across development, staging, testing, and production environment. They also help in scaling up and scaling down instances softly.
- Containerization tools help to maintain consistency across the environments where the application is tested, developed, and deployed. There is no chance of errors or failure in the production environment as they package and replicate the same dependencies and packages used in the testing, development, and staging environment. It makes the application easy to run on different computers.

7) Continuous Operations

- All DevOps operations are based on the continuity with complete automation of the release process and allow the organization to accelerate the overall time to market continuingly.

- It is clear from the discussion that continuity is the critical factor in the DevOps in removing steps that often distract the development, take it longer to detect issues and produce a better version of the product after several months. With DevOps, we can make any software product more efficient and increase the overall count of interested customers in your product.