DEVOPS AND CI/ CD - AN INTRODUCTION

PROGRAMMING APPLICATIONS AND FRAMEWORKS (IT3030)

LEARNING OUTCOMES

- After completing this lecture, you will be able to,
  - Describe the failures in the traditional software deployment methods, and the reasons for the rise of DevOps.
  - Describe the main components of DevOps.
  - Describe what CI/ CD is and how its application is beneficial in modern software development.
  - Describe the importance of Test Automation in DevOps.
  - Apply the concepts learnt in solving real-world problems.

CONTENTS

- Deploying an app
  - Typical Deployment Scenario
  - Failures
- DevOps
  - Core DevOps principles
  - Continuous Integration (CI)
  - Continuous Delivery and Continuous Deployment (CD)
  - Infrastructure Automation
  - Software Testing and Test Automation
- Wrap Up
- Summary

JOURNEY SO FAR

- What was discussed so far,
  - Software Frameworks
  - Version controlling with Git
  - Git Workflows
  - Web application architectures
  - REST APIs
  - An overview on Frontend development

DEPLOYING AN APP

- We have discussed how we engineer a web application so far.
- However, engineering the application is not enough.

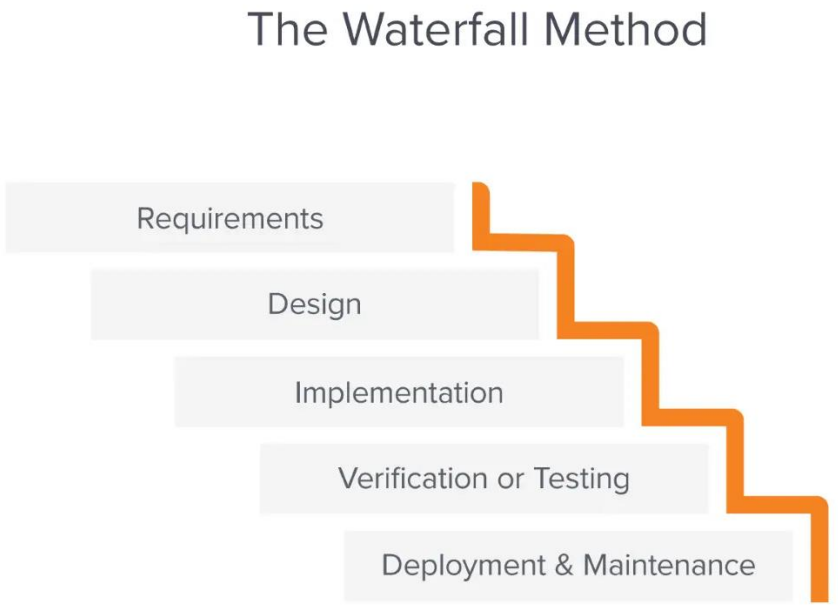
- It must be **deployed**!
  - Why? So that actual users can use the application.

DEPLOYING AN APP

- Application Deployment, also known as Software Deployment, is the process of installing, configuring, updating, and enabling one application or suite of applications that make a software system **available for use.** (Source)

TYPICAL DEPLOYMENT SCENARIO

- How were deployments handled traditionally?
  - Waterfall model was the most common SDLC model used back in the day.
    - SDLC - Software Development Life Cycle
  - In Waterfall model, each stage is carefully planned and executed sequentially by a dedicated team.
    - E.g.: Business stakeholders, Business Analysts, Architects, Developers, Quality Assurance, IT Operations



Source: https://business.adobe.com/blog/basics/waterfall

TYPICAL DEPLOYMENT SCENARIO

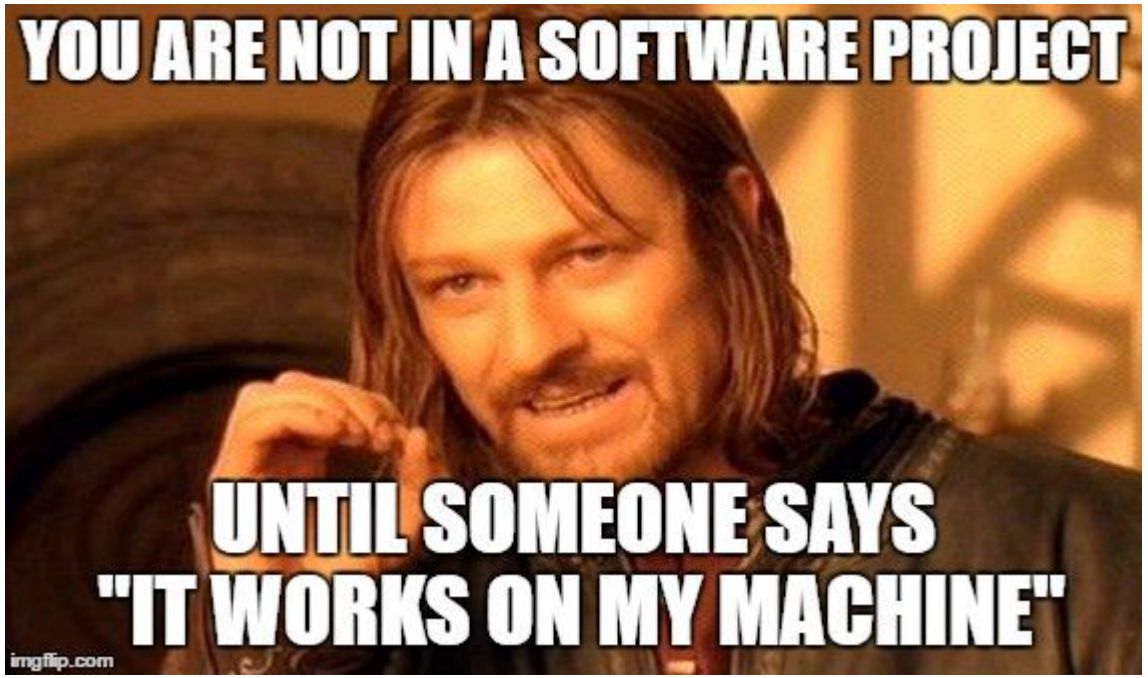
- A release would be carried out every few months.
- First, the developers would develop the solution.
- Then the QA team would start release testing the application leading to the release.

TYPICAL DEPLOYMENT SCENARIO

- The QA team would then discover bugs.
  - QA team would frantically report the bugs.
  - Developers would frantically fix the bugs.
- Now the QA team has to retest the fixes.
  - This is on top of the functionality remaining to be tested!
- Typically, the developers and/ or QA teams would get overwhelmed with the workload as the release date becomes closer.

TYPICAL DEPLOYMENT SCENARIO

- Then the Developers, QA and IT Operations teams would recommend to push ahead the release date given the problems.
  - Business stakeholders would not agree!
  - They dictate that release be carried out as planned as they have their own strategic goals to meet.
- IT Ops would also run into various problems deploying the application.
  - Famous example: **It works on my machine!**



Source: https://elbruno.com/you-are-not-in-a-software-project-until-someone-says-it-works-on-my-machine/

TYPICAL DEPLOYMENT SCENARIO

- Finally, the release would be carried out as planned even with the issues.
  - It will be rolled back in the next couple of hours after doing everything to make the application work fails.
    - Redeploy again with several components disabled.
  - Fixes will be carried out for some time after the release to stabilize the product.
  - The product is actually usable only after few days/ weeks after the release.
- **Rinse and repeat** for the next release!

TYPICAL DEPLOYMENT SCENARIO: FAILURES

- Each team (Dev/ QA/ IT Ops) worked separately and had competing objectives ("siloed").
- Each team had their own goals.
  - Resulted in botched releases and unhappy customers.
- Lots of firefighting/ finger pointing - Very stressful environment!

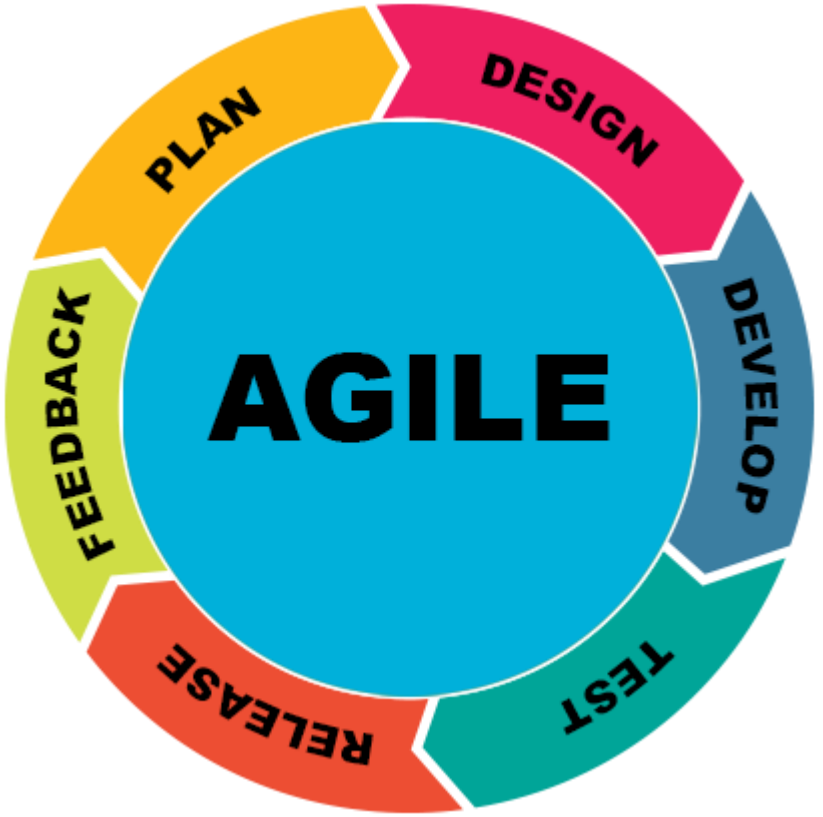
Source: https://resources.biginterview.com/behavioral-interviews/behavioral-interview-questions-conflict/

DEVOPS

- Somewhere between 2007-2009 different communities started raising concerns about the level of dysfunction in doing releases this manner.
- There had to be a better way of doing things!
  - Gradually a movement appeared against the traditional mindset of keeping different stakeholders separated in different silos.
- This is the origin of **"DevOps"**.
- [YouTube] What is DevOps?

DEVOPS

- DevOps stands for "Development-Operations".
- The concept behind DevOps is simple.
  - Bring the Developers, QA, IT Operations (IT Ops) teams together to work collaboratively on planning, building, testing and releasing software with shared responsibility for successful business outcomes.
  - Based on **Agile approach** to software development!
  - DevOps is not just a method for Software Development. It's a **culture**.

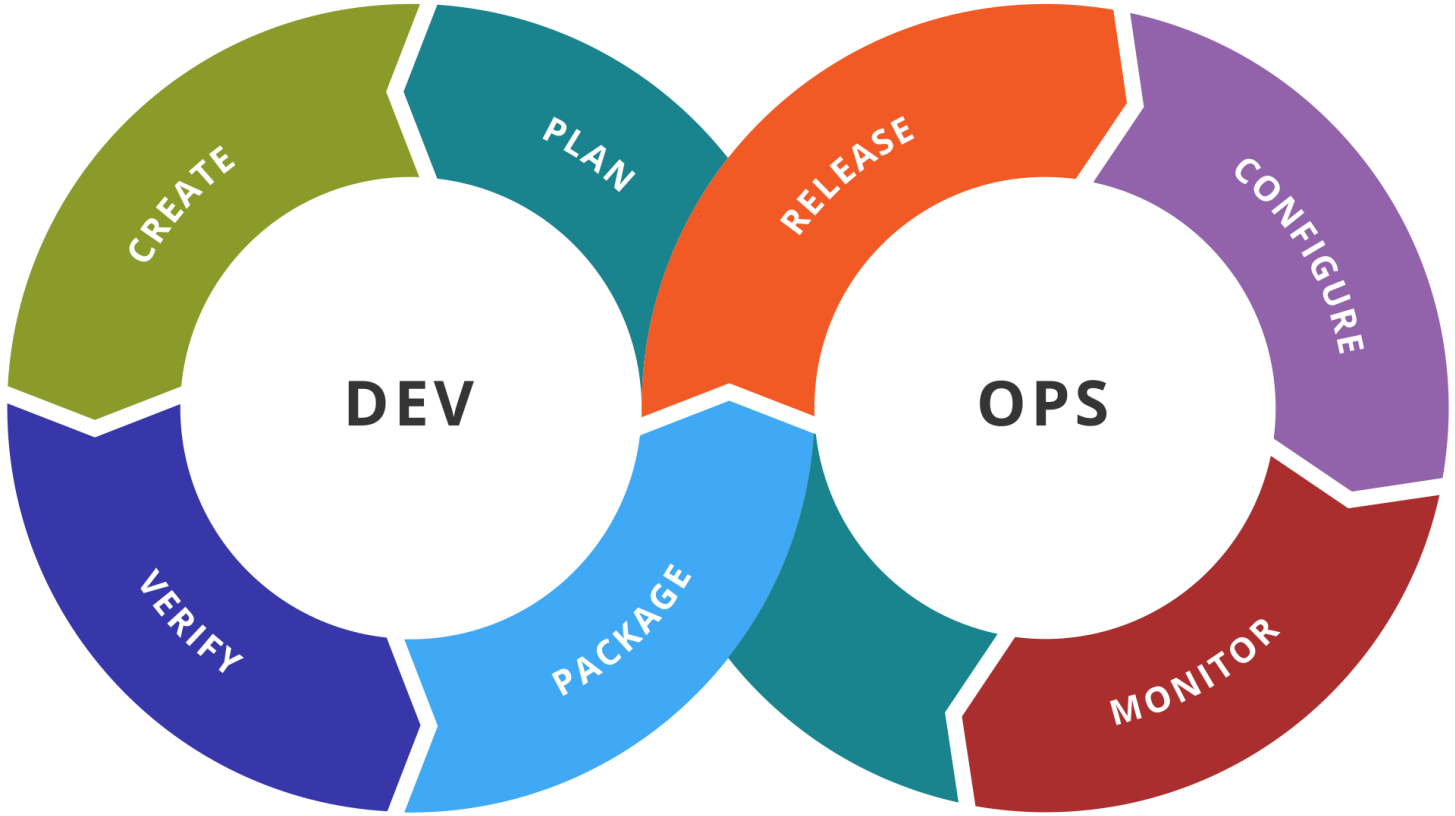


Source: https://technology.berkeley.edu/tpo/agile

DEVOPS

- DevOps is a combination of software development (dev) and operations (ops). It is defined as a software engineering methodology which aims to integrate the work of development teams and operations teams by facilitating a culture of collaboration and shared responsibility. (Source)

DEVOPS STAGES



Source: https://en.wikipedia.org/wiki/DevOps_toolchain

CORE DEVOPS PRINCIPLES

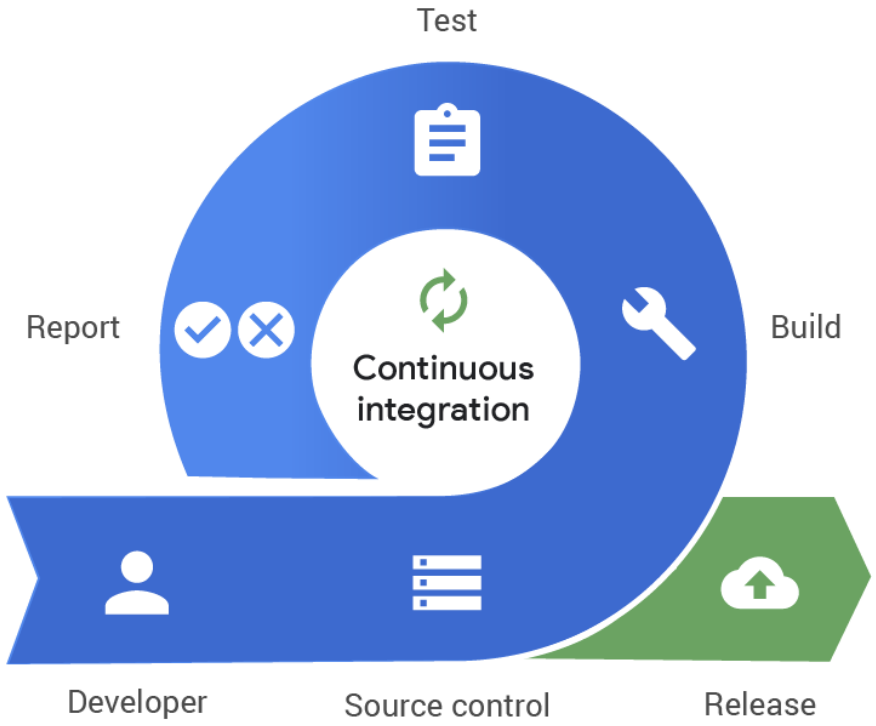
- Automation of the software development lifecycle
  - Automating the SDLC as much as possible increases productivity by reducing the introduction of human errors and freeing up humans from repetitive tasks.
  - Mainly consists of,
    - Continuous Integration (CI)
    - Continuous Delivery and Continuous Deployment (CD)
    - Infrastructure Automation (Automated provisioning of Environments and other resources)
    - Test Automation - **Shift Left** Testing (Read this)

CORE DEVOPS PRINCIPLES

- Collaboration and communication
  - Developers, QA, stakeholders, IT Ops should communicate clearly to collaborate efficiently. At the end of the day, it is everyone's responsibility to deliver software which satisfies business requirements.
- Continuous improvement
  - It is the practice of focusing on experimentation, minimizing waste, and optimizing for speed, cost, and ease of delivery.
- Customer-centric action
  - DevOps teams use short feedback loops with customers and end users to develop products and services centered around user needs.

CONTINUOUS INTEGRATION (CI)

- Continuous integration is the practice of,
  - Frequently pushing all code changes into a designated main branch of a code repository,
  - automatically kicking off a build whenever there is a new change(s),
  - And running automatic tests against the build to verify the change.
    - Developers need to create the tests too if they don't exist already.



Source: https://www.pagerduty.com/resources/learn/what-is-continuous-integration/

CONTINUOUS INTEGRATION (CI)

- By merging changes frequently and triggering automatic testing and validation processes,
  - the issues can be discovered early and fixed then and there.
  - Integration challenges that can happen when waiting for release day to merge changes into the release branch can be avoided.
  - Quality of the product can be verified from the early days of development rather than near the release date (Shift-left testing).

CONTINUOUS DELIVERY (CD)

- **Continuous delivery** is an extension of continuous integration.
- It is the deployment of all built and automatically tested deliverables (basically executables) to a further testing and/or production environment after the CI stage via an automated process.
- This **takes away the stress on a team** to prepare for a delivery for days.

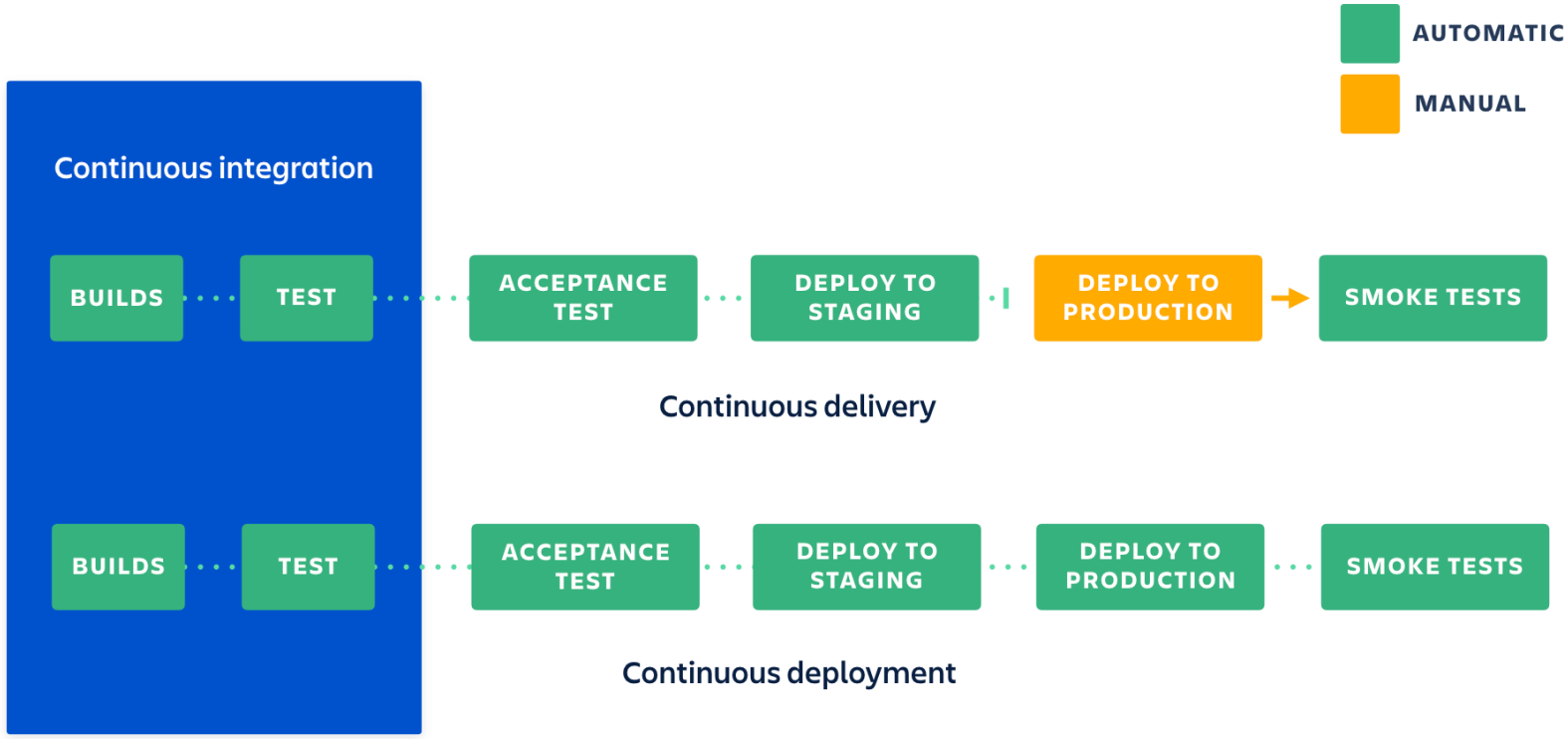
CONTINUOUS DELIVERY (CD)

- This deployment process is triggered manually. Once the process is triggered, the deployment happens automatically.
- The user has the control over when the automatic deployment should be triggered with just a click of a button.
- **Feature flags** are used to disable/ hide incomplete features from affecting customers in production.

CONTINUOUS DEPLOYMENT (CD)

- **Continuous Deployment** is a further extension of Continuous Delivery.
- With Continuous Deployment, every change that passes all stages of the production pipeline is released to the customers.
- There is **no human intervention**, and only a failed test will prevent a new change to be deployed to production.
- Feature flags are an inherent part in Continuous Deployments.

CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY VS. CONTINUOUS DEPLOYMENT



Source: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

SOME POPULAR TOOLS FOR CI/ CD

Open Source

Jenkins | GoCD | GitLab CI | Drone CI | Spinnaker | Buildbot

SaaS

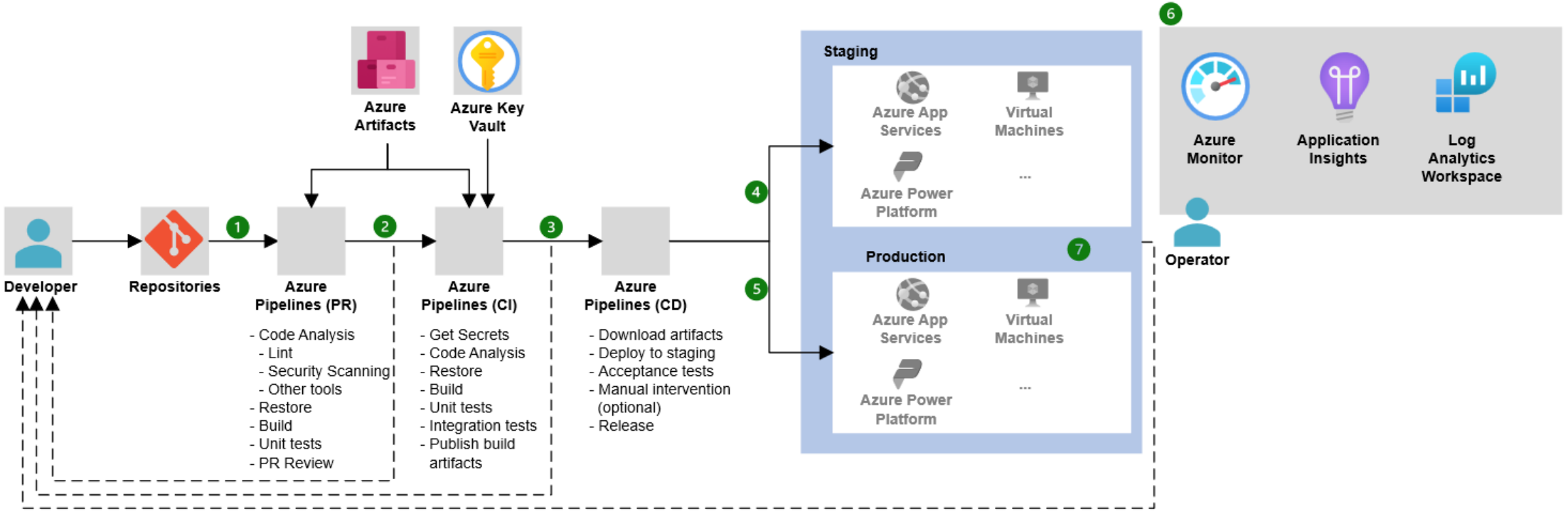
Codeship | Travis CI | TeamCity | CircleCI | GitHub Actions | Semaphore

Cloud Services

Azure DevOps | AWS CodePipeline | Google Cloud Build

Source: https://livebook.manning.com/book/pipeline-as-code/chapter-1/135

EXAMPLE CI/ CD WORKFLOW ON AZURE DEVOPS



Source: https://learn.microsoft.com/en-us/azure/architecture/example-scenario/apps/devops-dotnet-baseline

BENEFITS OF CONTINUOUS INTEGRATION

- Less bugs get shipped to production as regressions are captured early by the automated tests.
- Building the release is easy as all integration issues have been solved early.
- Less **context switching** as developers are alerted as soon as they **break the build** and can work on fixing it before they move to another task.
- Testing costs are reduced drastically – your CI server can run hundreds of tests in the matter of seconds.
- Your QA team spends less time testing and can focus on significant improvements to the quality culture.

BENEFITS OF CONTINUOUS DELIVERY

- The **complexity of deploying software has been taken away**. Your team doesn't have to spend days preparing for a release anymore.
- You can release more often, thus **accelerating the feedback loop** with your customers.

BENEFITS OF CONTINUOUS DEPLOYMENT

- Deployments pipelines are triggered automatically for every change.
- Releases are less risky and easier to fix in case of problem as you deploy small batches of changes.
- Customers see a continuous stream of improvements, and quality increases every day, instead of every month, quarter or year.

CI/ CD BEST PRACTICES

- Self-study activity: Read on best practices here.

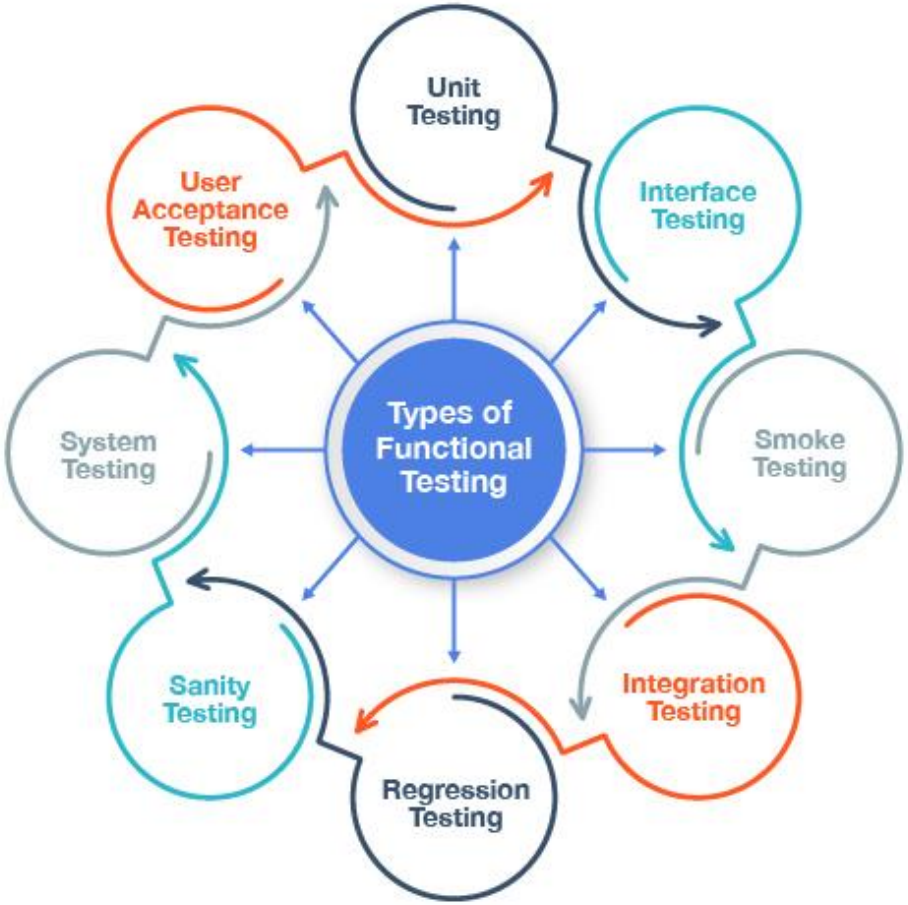
INFRASTRUCTURE AUTOMATION

- Infrastructure automation refers to the practice of using software and tools to automate the provisioning, configuration, management, and operation of IT infrastructure.
- It involves automating various tasks that would traditionally be performed manually, reducing the need for human intervention and increasing efficiency and reliability.
- [YouTube] Infrastructure Automation with Red Hat Ansible

SOFTWARE TESTING

- Software testing can be broadly categorized into two types.
  - Functional tests – verifies if functional requirements are met
  - Non-functional tests – verifies if non-functional requirements are met

FUNCTIONAL TEST TYPES



Source: https://testsigma.com/blog/the-different-software-testing-types-explained/

NON-FUNCTIONAL TEST TYPES

Source: https://testsigma.com/blog/the-different-software-testing-types-explained/

TEST AUTOMATION

- In continuous testing, various types of tests are performed within the CI/CD pipeline. These can include:
  - **Unit testing**, which checks that individual units of code work as expected.
  - **Integration testing**, which verifies how different modules or services within an application work together.
  - **Regression testing**, which is performed after a bug is fixed to ensure that specific bug won't occur again.
- Self study activity: Read this article on Automated software testing.

WRAP UP

- [YouTube] DevOps CI/CD Explained in 100 Seconds
- Try out GitHub Actions: a CI/ CD Platform on GitHub.

SUMMARY

- Deploying an app
  - Typical Deployment Scenario
  - Failures
- DevOps
  - Core DevOps principles
  - Continuous Integration (CI)
  - Continuous Delivery and Continuous Deployment (CD)
  - Infrastructure Automation
  - Software Testing and Test Automation

REFERENCES

1. What is CI/CD?
2. 4 Must-know DevOps principles
3. Manifesto for Agile Software Development
4. How Is Netflix SO GOOD at DevOps?
5. What Is Shift Left Testing?
6. Continuous integration vs. delivery vs. deployment
7. The different types of software testing

THANK YOU

VISHAN.J@SLIIT.LK

NELUM.A@SLIIT.LK