COMP1531

🌏 Projects - Introduction

Lecture 7.3

Author(s): Hayden Smith

(Download as PDF)

In This Lecture

- **Why? 🤔**
  - The purpose of most software is for people to use it, and for that to happen we need processes to make it available for usage

- **What? 📰**
  - Deployment history
  - Continuous Delivery
  - Continuous Deployment
  - DevOps

Deployment

Activities relating to making a **software system available for use**

Many diagrams in this lecture are sourced from both *gitlab* and *atlassian*

Basic Deployment

What is one of the most basic forms of deployment anyone can do? Deploying to your **CSE Account**

Every CSE student has a **public_html** folder that is exposed to the internet.

Basic Deployment

What is one of the most basic forms of deployment anyone can do? Deploying to your **CSE Account**

Every CSE student has a **public_html** folder that is exposed to the internet.

Let's try this out together!

🏛️ History Of Deployment

Historically, deployment was much less frequent and much more physical.

Code would be worked on for days at a time without being tested, and deployed sometimes years apart. This was largely due to software historically being a physical asset (e.g. a CD).

💨 Something Changed

However, that isn't the case anymore. Two major changes have occurred over the last 10-15 years:

- Increased prevalence of web-based apps (no local installs)
- Improvement to internet connectivity, speed, bandwidth

These changes (and more) have allowed for the pushing of updated software to users to be substantially more frequent. E.G. Think about how quickly Facebook is updated.

A movement from software as an asset, to software as a service, has catalysed this transition

👨‍🍳 Software As A Service (SaaS)

Service V Asset

A simple case study can be found in Microsoft's movement of Windows from shipping a product, to shipping a service.

☁️ Cloud Services

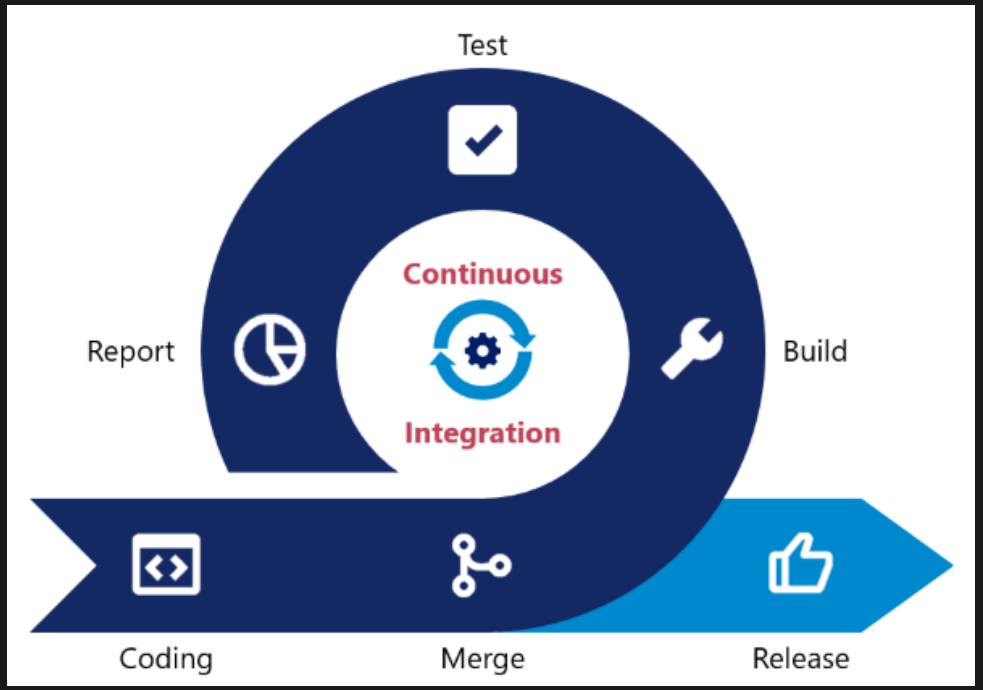
Numerous cloud services offer the ability to "easily" deploy your web applications.

Examples include:

- Amazon Web Services
- Google App Engine
- Heroku

♻️ Recap: Continuous Integration

Continuous integration: Practice of automating the integration of code changes from multiple contributors into a single software project.



👽 Modern Deployment With CI

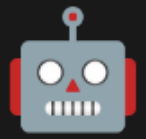
To achieve rapid deployment cycles, modern deployment isn't as simple as pushing code. Rather, a heavily integrated and automated approach is preferred. This new part of the pipeline we will explore is called **Continunous Delivery**.

Source: Gitlab

🤖 Continuous Delivery

Continuous delivery: Allows accepted code changes to be deployed to customers quickly and sustainably.

This involves the **automation of the release process such that releases can be done in a "button push"**.



Continuous Delivery

Many companies will have a daily or weekly deployment or "ship".

To "ship" code there is a "sign off" process (human intervention) to deploy the code.

🚗 Release Methods

Continuous delivery is often concerned with more than just going from "your computer" to a "production environment".

We can have 0 or more interim stages of release.

🚗 Release Methods

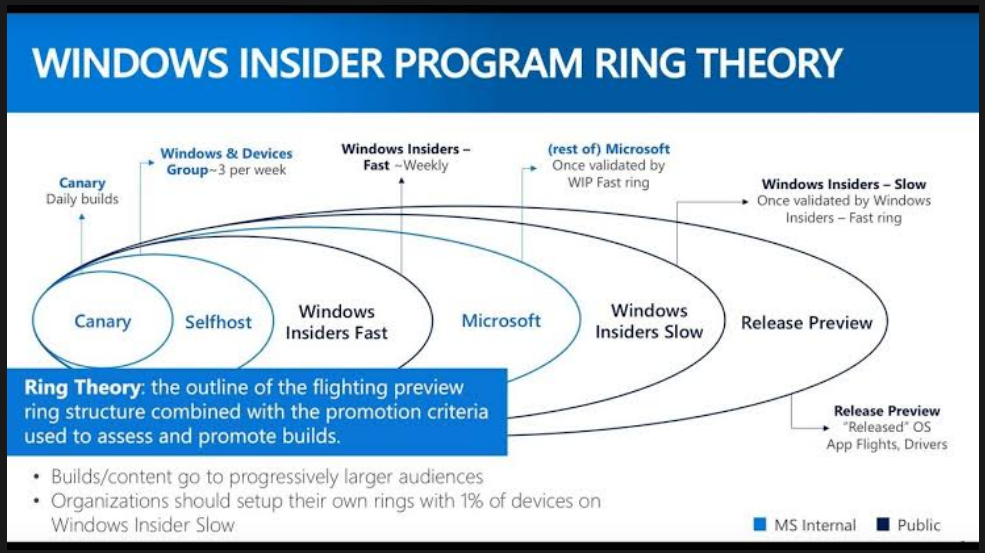
For example, you might have 3 core tiers:

- **dev**: Released often, available to developers to see their changes in deployment
- **test/staging**: As close to release as possible, ideally identical to prod
- **prod**: Released to customers, ideally as quickly as possible

As you move down the stages, things tend to be more stable.

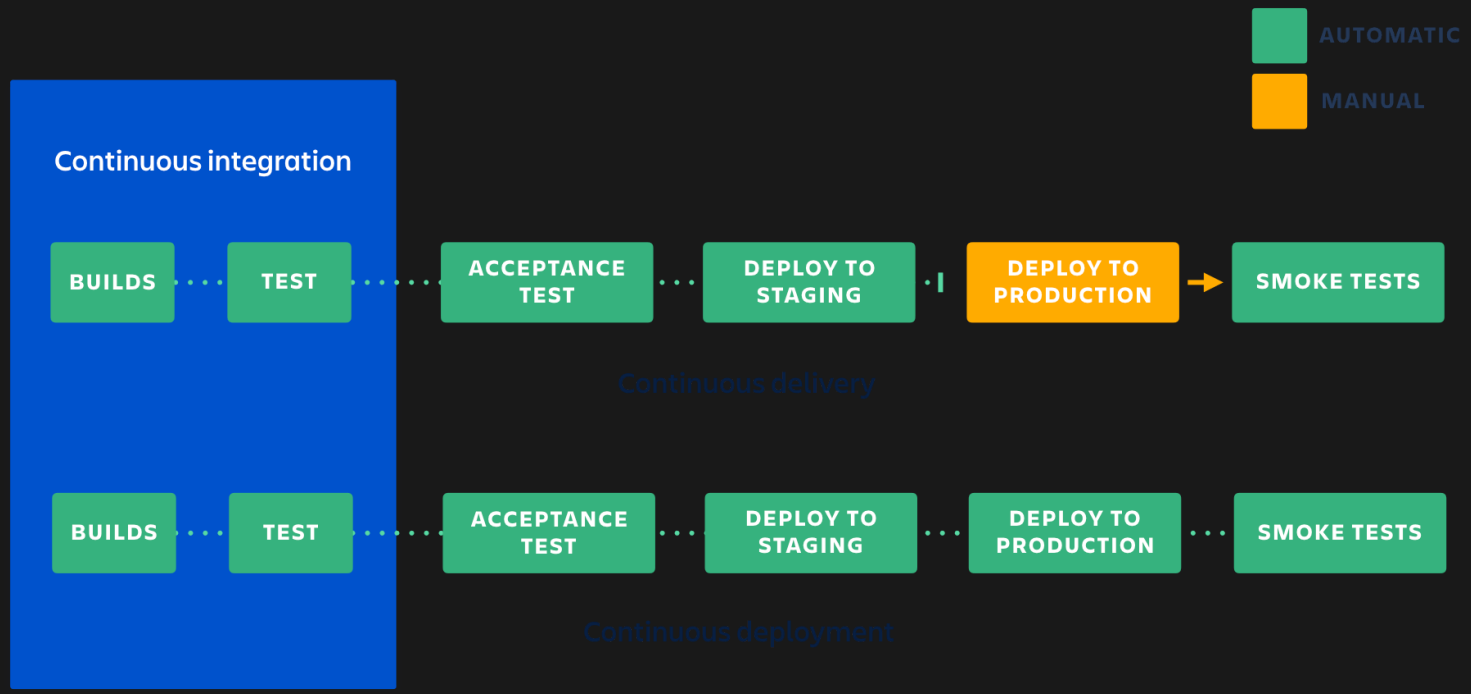
✈️ Flighting

For example, Microsoft historically had a very nuanced release method they referred to as their flighting ring structure.



Continuous Deployment

Continuous Deployment is an extension of Continuous Delivery whereby changes attempt to flight toward production automatically, and the only thing stopping them is a failed test.



Source: Atlassian

 Deploying On Your Own

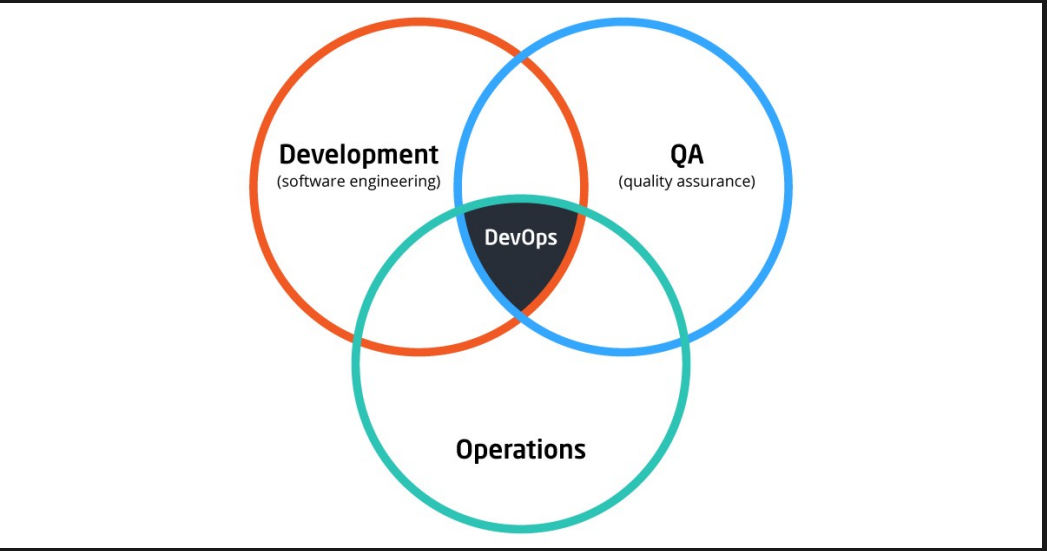
This term COMP1531 has decided to use a free service known as "alwaysdata" to let students deploy their backend to the cloud.

Instructions of how to set this up are found in the project repository for iteration 3 as a pre-recorded video.

👨‍💻 DevOps

A decade ago, you would typically expect people from a multitude of separate roles to work together to coordinate the deployment of work.

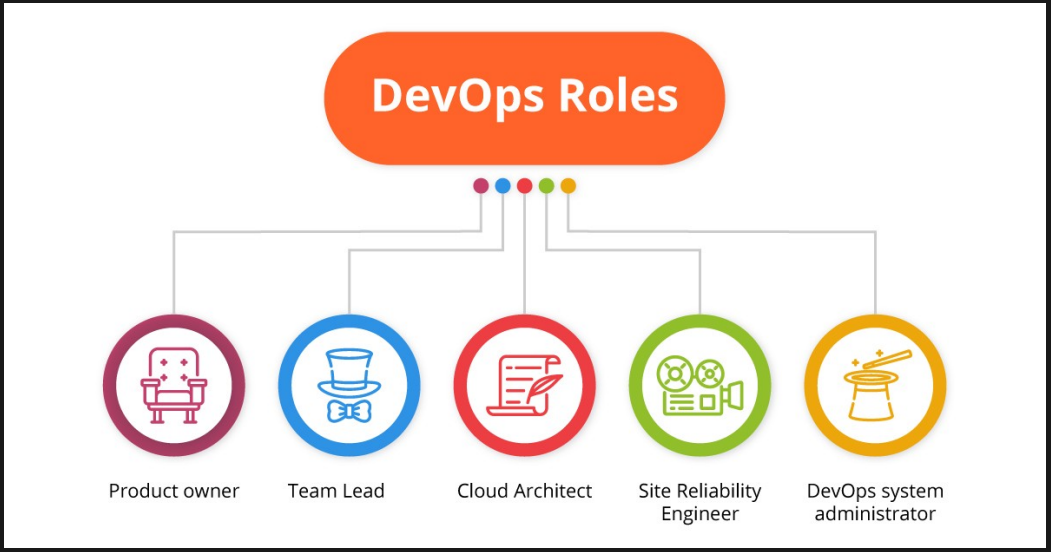
In recent years, DevOps as a role has become a more relevant role that focuses on skill(s) that include an overlap of development, deployment, and quality assurance.



👨‍💻 DevOps

Wikipedia says that *DevOps is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality*

As development teams become less silo'ed, modern DevOps is less a role, and more a series of roles or aspect of a role.



Source & Reading

🔧 Maintenance & Monitoring

Maintenance: After deployment, the use of analytics and monitoring tools to ensure that as the platform is used and remains in a healthy state.

- Monitoring often has two purposes:
  1. *Preserving user experience*: Monitoring errors, warnings, and other issues that affect performance or uptime.
  2. *Enhancing user experience*: Using analytical tools to monitor users or understanding their interactions. Often leads to customer interviews and user stories

🔧 Maintenance & Monitoring

- Health is defined by developers, but often consists of:
  - Monitoring 4XX and 5XX errors
  - Ensuring disk, memory, cpu, and network is not overloaded

Often these aren't actively monitored, but rather monitored with alerts and triggers

📚 Further Reading

- Atlassian Continuous Delivery
- Gitlab Continuous Integration
- Atlassian Delivery VS Deployment

🦻 Feedback

Or go to the form here.