DAY 1: Welcome to Introduction to DevOps

What will you learn?
you will be introduced to DevOps, a culture of collaboration between Development and Operations teams. This allows teams are able to more reliably and rapidly release high quality software to their users.

Through this course, you will learn:

- The stages that changes to software go through to get from a developer's computer to its users
- The role and responsibilities of a traditional Operations team
- An overview of modern infrastructure management
- The difference between a DevOps culture and that of a traditional organization
- The purposes of key practices such as CI/CD, monitoring, and containerization
- The role of scalability, observability, and resiliency in a modern software system

---

What is Deployment?
Deployment is the general process of making a piece of software available to its users. Software developers are able to deploy their software via the internet with greater ease and speed of delivery than ever before. This is the same process for a new application, or a minor patch to fix a bug.

we will cover the most commonly used tools and practices:

- Infrastructure management
- Version control systems
- Testing
- Deployment environments

The Progress Flow:

- Development
- Version Control
- Deployment
- Users

Infrastructure Management
Traditionally, an Operations team is responsible for managing a company's servers.

Servers are computers that run software that can be accessed by the internet. Really the Operations team manages all of an application's **infrastructure**.

Infrastructure is the full set of resources that support the development, testing, and deployment of applications. This consists of:

- Hardware components such as servers, routers, switches and cables.
- Software components such as operating systems, version control systems, and applications.

The responsibility of the Operations team, include:

- Installing and replacing (a.k.a "provisioning") physical components such as servers, switches, and hard drives.
- Performing software/firmware upgrades such as security patches
- Configuring infrastructure such as firewalls, user access, ports
- Monitoring network health and alerting personnel when issues arise

---

Version Control Systems

Version control systems (such as Github) are tools designed to manage different versions of a file or project. They allow you to update and save previous versions of software. Some of the data that is tracked by a version control system includes:

- changed files
- new or deleted files
- renamed or moved files
- the author and date of the change

VCS lowers the risk and impact of bugs or loss of programs. You will be able to compare stable and new versions of a program before pushing a change.

Version control systems (VCS) change how teams work together. This new tech has created branching and merging, which Dev teams use all the time.

- Branching: is the process of creating a copy of the source code (the "trunk").
- Merging is the process of combining the changes in one branch with another.

version control systems are able to synchronize with project management tools.

Testing

Testing is an essential component of the deployment process, this will ensure new

features integrate with existing features.
Different types of tests exist that are used in the various stages of deployment. Such as:

- **Unit test** — evaluates the smallest possible unit of testable code, such as a single function.
- **Integration test** — evaluates how the units of a particular program work with one another.
- **Acceptance test** — evaluates whether the user experience aligns with the business requirements of the software.
- **End-to-end test** — evaluates the application's behavior using production-like infrastructure that includes networking, databases, and calls to external APIs.

Deployment Environments
The **production environment** refers to the infrastructure that supports the complete application used by real users.
an **environment** is the subset of infrastructure resources in a controlled environment.
Each intermediate environment allows developers to test new software, without any major risks.

a common set of environments includes:

- The **local development environment** — where software is first written and tested, typically on a developer's own computer.
- The **integration environment** — where software changes are merged using a version control system.
- The **quality assurance (QA) / testing environment** — where tests are executed to ensure the functionality and usability of each new feature.
- The **staging environment** — where the software can be performance tested in a production-like environment.
- The **production environment** — where software is accessible by real users!

Each of these environments can be viewed as a space that developers can use throughout the entire deployment process.

Using a staging environment is quite similar to using focus groups to test new products.

The movement of software in and out of intermediate environments is a common

source of bugs, particularly when these migrations are performed manually.

Review
we learned about **deployment** — the process of making a piece of software available to its users. The process starts with a developer moving their code into a **version control system**, through a **staging environment**, and ends with them deploying their code to a **production environment**.

This deployment process ensures that new code is shipped quickly, reliably, and with high quality.

#DEV-OPS-CERT-CODE-ACADEMY