Version control

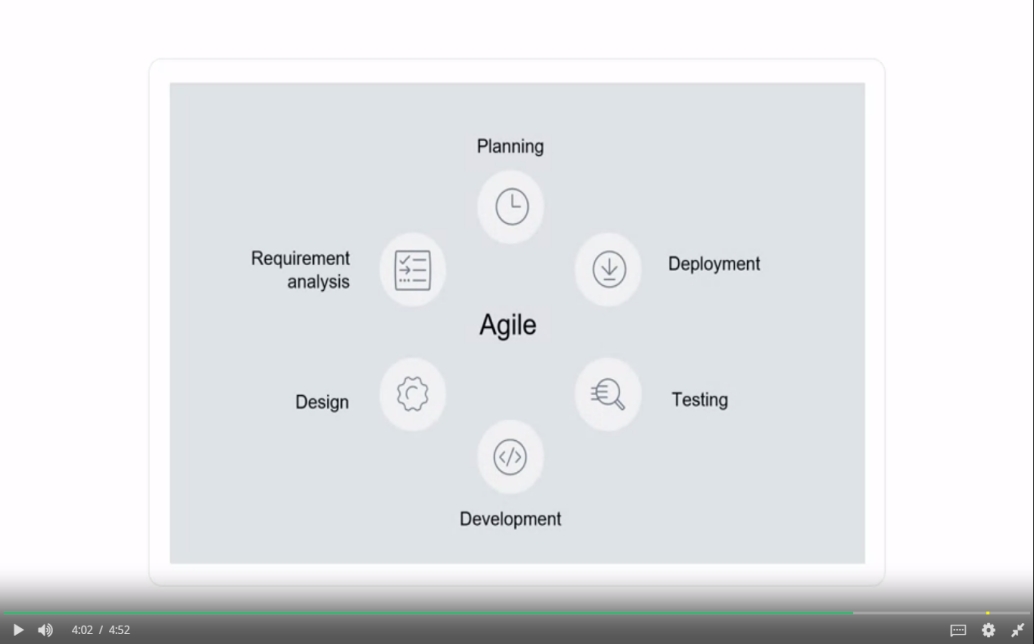
**Version control** is a system that records all changes and modifications to files for tracking purposes. Developers also use the term source control or source code management.

The primary goal of any version control system is to keep track of changes.

There are many benefits associated with version control.

These include; revision history, identity, collaboration, automation, and efficiency

Revision history provides a record of all changes in a project.allows teams to see not only when the changes occurred, but also who made the changes



There are many different version control systems available. For example, Subversion, Perforce, AWS Code Commit, Mercurial, and Git to name a few.

Version control systems can be split into two types or categories. Centralized version control systems and distributed version control systems.

**Centralized version control systems**, or CVCS for short, contain a server and a client. Developers working on projects using a centralized system need to pull down the code from the server to their local machine. This gives the user their own working copy of the code base.After making changes to the code, the developer needs to push the changes to the central server so that other developers can see them. he server holds the full history of changes,

Advantage of CVC: easier to learn than DVC, They also give more access controls to users

Disadvantage of CVC: they can be slower given that you need to establish a connection to the server to perform any actions

**Distributed version control systems** or DVCS for short, in this system every user is essentially a server and not a client. This means that every time you pull down code from the distributed model, you have the entire history of changes on your local system.

Advantage of CVC: It works as if you are actually connected to the server directly but on your own local machine.You only ever need to connect to the server to pull down the latest changes or to push your own changes.Speed and performance are also better than its CVCS.

# A history of version control

version control systems were created before the Internet!

One of the first significant Version Control Systems was the Concurrent Versions System (CVS). It was first developed in 1986 by Walter F. Tichy at Purdue University and released publicly in 1990.

it has some significant flaws .CVS does not include integrity checks which means your data can become corrupted.

The main successor to CVS was Subversion (SVN).

CollabNet developed Subversion in 2000 and solved many of the issues present in CVS.it included integrity checks in its design. However Subversion used a centralized VCS model.If the server were down or slow, this would impede development.

In 2005, two new projects were started to develop distributed version control systems; Mercurial and Git. Both projects were created in response to an event involving the Linux kernel development.

**Mercurial** was developed by Olivia Mackall.It is developed as a high-performance distributed VCS

**Git** was developed by Linus Torvalds to host the Linux kernel’s source code. Like Mercurial, it is a distributed VCS. Its first public release came in 2007.

# Version control in professional software development

**Workflow**

Workflows are essential to ensure code is managed correctly and reduce mistakes from happening. Different projects will have different workflows.

**Continuous Integration**

Continuous Integration, or CI, is used to automate the integration of code changes from multiple developers into a single main stream.. Continuous Integration, or CI, is used to automate the integration of code changes from multiple developers.

**Continuous Delivery**

Continuous Delivery is an extension of Continuous Integration. Once the changes have been merged into the main stream, a Continuous Delivery system automatically packages the application and prepares it for deployment. This helps avoid human error when packaging the application.

**Continuous Deployment**

Continuous Deployment is an extension of Continuous Delivery. The goal of Continuous Deployment is to deploy and release software to customers frequently and safely. involves automatically deploying to a test (also known as staging) environment first to validate the deployment package and software changes before releasing to customer.

The **revision history** will record the essential data points so any developer or team member can walk through the entire project from start to its current state. Every change that has occurred on the project should be easily accessible either by a simple command or integration into the developer's IDE.

After commit, the developer will push their changes to the repository and create something called a pull request. Developers will then peer review the pull request to approve request changes or decline.

# Staging vs. Production

**Development Environments**

Developers need to set up multiple environments for different ways to test and verify.The main purpose of this flow is to find any potential issues that may arise due to changes or new features being added to the codebase.

**Staging**

The staging environment should mimic your production environment. Staging environments can also be used for testing and verifying new features and allow other teams including QA or stakeholders to see and use those features as a pre-trial. Staging should also cover all areas of the architecture of the application including the database and any other services that may be required. Areas that benefit from staging environments include:

New Features

They allow teams to verify that the feature works, it can be turned on and off via configuration flags and also that it does not break or interfere with existing functionality.

Testing

As the staging environment mimics your production environment, it's also a great place to run tests.The types of testing covered will be Unit testing, Integration testing and performance testing.

Migration

Staging is a perfect place to test and verify data migrations. Snapshots can be taken from production and used to test your migration scripts.something like a migration in production is extremely risky.

Configuration Changes

Having a staging environment will allow you to spot any potential issues or bottlenecks for configuration.

**Productuon**

Production is live. It's out there for people to see and/or interact with. Any issues or problems you may have had should have been caught and fixed in the staging environment.

Downtime

Downtime for any service especially customer facing will most likely be revenue impacting.

Vulnerabilities

Cyber-security should also play a big role in what gets released in production. Any updates to software such as patching or moving to the latest version should be checked and verified.

Reputation

Downtime or issues in production is damaging for a company’s reputation as it does not instill confidence in end users

**UTILIZING THE COMMAND LINE**

By learning just a few commands, you can perform various tasks, such as creating new directories, creating new files, combining directories, copying and moving files around different directories, and searching through files using various criteria. As you become more advanced in using the command line, you will be able to perform tasks such as track software, access and control remote servers, search for files using specific criteria, unzip archives, access software manuals, Install, upgrade, and uninstall software. First, the cd command, which stands for change directory. This is used to point our command line to a specific directory

**Bash commands**

Bash provides a list of commands for navigating through files

| ****Command**** | ****Used for**** |
| --- | --- |
| cd | Change Directory |
| ls | List command used for showing the content of a directory. |
| rm | Remove command used for removing a file or a directory |
| mv | Used to move files or folders to another location |
| touch | Allows creating of a new empty file or to update a timestamp on a file |
| cp | Used to make a copy of a file or folder |
| mkdir | Make a new directory |
| pwd | Print work directory, shows the current location in the shell |
| cat | Allows reading or concatenation of a file |
| less | Displays the contents of a file one page at a time. |
| grep | Global regular expression, allows for searching contents of files or folders |

**Flags**

Every bash command has flags for changing the output of the command itself. For example, the ls command prints out the list of contents inside a directory. If we wanted to show the list in a different view, we simply need to add a flag such as -l.

**Man Pages**

he man page lists all the flags and options that a particular command has to offer.

**Editing**

There are many options for editing files in bash. The most common is usually VI or Vim. VI stands for visual editor. It's used for making edits and changes to a file and saving them.VIM is a better version of VI with some improvements - hence its name: visual editor improved. It uses insert mode, normal mode and command line mode

**My github token**

ghp\_jQqkRhdnxqCVXdXMB6lbQGIGs6BGse0PZNTx