CI/CD & Jenkins

Dockerisation

* Docker basics
* Docker CLI
* Docker images & dockerfile
* Docker registry
* Docker compose
* Docker security
* Docker in Devops

Simplified loop

* **Develop the code**
  + How do I configure the environment
  + Configure step by step
  + Can we do better
* **Deploy the code to staging** 
  + Work for me problem
* **Deploy the code to production**

Docker使用目的

使用一个容器能把代码在所有开发服务器使用

What is docker?

* Physical computer 🡪 virtual computer = virtualisations
* Pre-configured image
* Resource efficient

Why virtualisations

* Resource utilization

Docker architecture

* Docker daemon
* Client
* CLi
* GUI
* Container
* Image
* Resource
  + Network
  + Data volume

CI/CD

A white paper with text and a diagram

Description automatically generated with medium confidence

A diagram with blue green and purple lines

Description automatically generated

A white background with text and images of papers and arrows

Description automatically generated with medium confidence

([wiki](https://en.wikipedia.org/wiki/Infrastructure_as_code#:~:text=The%20difference%20between%20the%20declarative,be%20changed%20to%20meet%20this.))([Amazon](https://aws.amazon.com/what-is/iac/#:~:text=Infrastructure%20as%20code%20(IaC)%20is,%2C%20database%20connections%2C%20and%20storage.)) Much like software code describes an application and how it works, infrastructure as code (IaC) describes a system architecture and how it works. An infrastructure architecture **contains resources such as servers, networking, operating systems, and storage. IaC controls virtualized resources by treating configuration files like source code files**. You can use it to manage infrastructure in a codified, repeatable way.

The **declarative approach** defines the desired state and the system executes what needs to happen to achieve that desired state. **Imperative** defines specific commands that need to be executed in the appropriate order to end with the desired conclusion.

Microservices ([Amazon](https://aws.amazon.com/microservices/))

Microservices are an architectural and organizational approach to software development where software is composed of small independent services that communicate over well-defined APIs. These services are owned by small, self-contained teams.

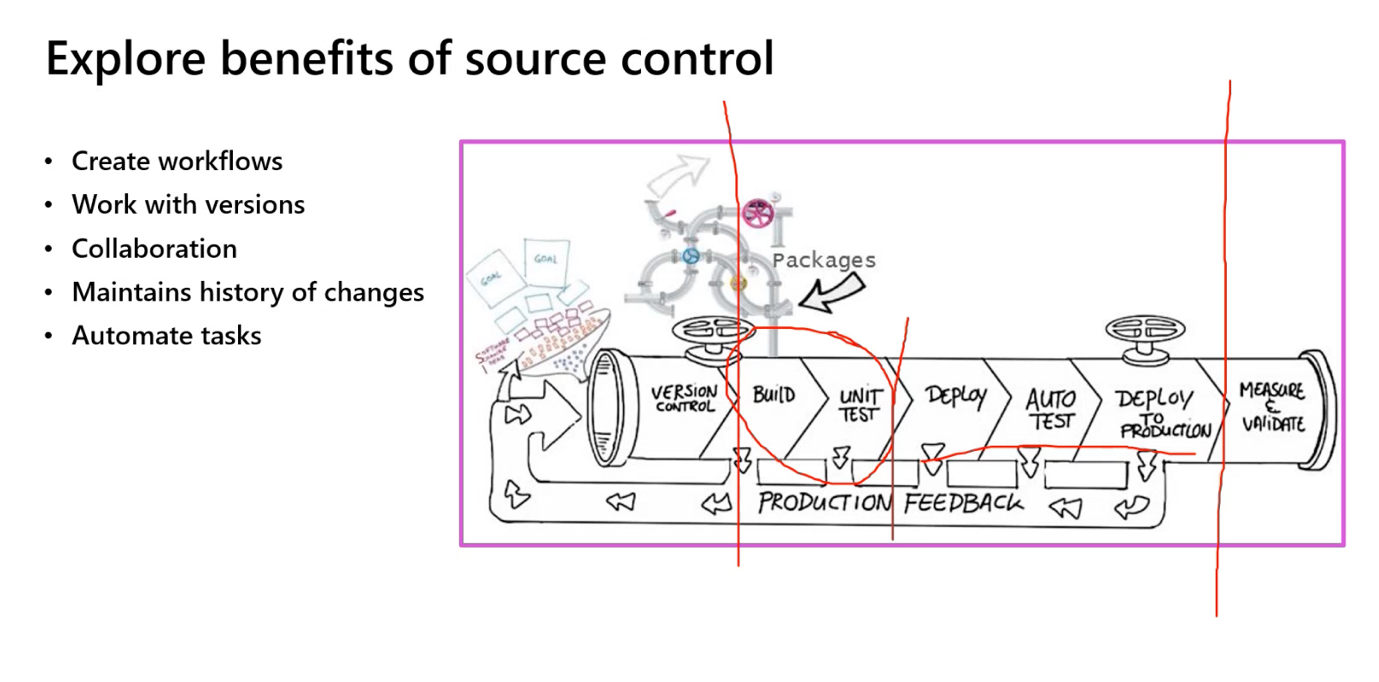
Microservices architectures make applications easier to scale and faster to develop, enabling innovation and accelerating time-to-market for new features.

A drawing of a crane lifting weights

Description automatically generated

A screenshot of a computer

Description automatically generated



CI 解决的是： build、unit test

CD解决的是：deploy、auto test、deploy to production

Types of repo

**Monorepos** – source control pattern where all the source code is kept in a single repo.

**Multiple** **repositories**/ traditional repositories – refer to organing your projects each into their separate repo.

Cloud computing ([Microsoft](https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-cloud-computing#:~:text=Simply%20put%2C%20cloud%20computing%20is,resources%2C%20and%20economies%20of%20scale.))

Simply put, **cloud computing is the delivery of computing services—including servers, storage, databases, networking, software, analytics, and intelligence—over the internet (“the cloud”) to offer faster innovation**, flexible resources, and economies of scale. You typically pay only for cloud services you use, helping you lower your operating costs, run your infrastructure more efficiently, and scale as your business needs change

Azure Devops

Terminology when releases pipeline

**Canary releases**

* Releasing a feature to a limited subset of end users
* How to implement: through The Load Balance
* Canary implement to traffic manager e.g. released only 5% of users

**Blue-green deployment**

* **There are two version of app swapping each other as one feature released**

A/B testing

* A/B testing is an experiment where two or more variants are shown to users at random, and statistical analysis is used to determine which variation performs better for a given conversion goal.

Dark launching

* Deploy some features to specific customers then deploy to all customers
* Pros of Feature Testing
  + Tests how a feature performs under load
  + ﻿﻿Allows for gradual exposure to users
* Cons of Feature Testing
  + Hidden features might still impact the production environment in unforeseen ways
  + ﻿﻿Requires feature toggles