Software Continuous Delivery in Distributed Systems by DevOps

Software Continuous Delivery

«I use the term “software delivery” to indicate the steps from a developer finishing work on a new feature, to that feature being used in production»
https://martinfowler.com/delivery.html

Continuous delivery (**CD**) is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time and, when releasing the software, without doing so manually.[1][2] It aims at *integrating*, building, testing, and releasing software with greater speed and frequency.

https://en.wikipedia.org/wiki/Continuous_delivery

Distributed Systems

A *distributed system* is a system whose components are located on different networked computers, which communicate and coordinate their actions by passing messages to one another from any system.

- On Premise
- On Cloud
- On Edge

https://en.wikipedia.org/wiki/Distributed_computing

DevOps

DevOps is a set of practices that combines software development (*Dev*) and IT operations (*Ops*). It aims to shorten the systems development life cycle and provide continuous delivery with high software quality.[1] DevOps is complementary with Agile software development; several DevOps aspects came from the Agile methodology.

https://en.wikipedia.org/wiki/DevOps

Contents

- DevOps
  a. DevOps definition, core concepts and history
  b. DevOps and Network and Services Management
- Machines, VMs, microVMs, Containers (services, networks and applications)
  a. Intro to Network and Services Management
  b. Intro to Linux: network, users, software install ...
  c. From Machines to VMs to Containers (Scalability -> Cloud)
  d. Docker as the container platform
     i. Building docker images
     ii. Services in a single host: docker-compose
     iii. Docker containers networking
- Cloud computing
  a. Definition, main concepts and main public clouds: AWS, Azure, Google Cloud
  b. Provisioning and Configuration
     i. Need to manage hundreds of machines and virtual machines: Ansible and Terraform
  c. Orquestating machines and containers
     i. Need to manage hundreds of containers: Swarm and Kubernetes
- Monitoring and Observability of services
- CI/CD: from development to deployment, the DevOps culture
  a. Jenkins as the CI automation tool
  b. CD automation tools

Where and how?

All the classes will be in the laboratory **Fuenlabrada Lab III** - **L3.208**

The host operating system will be Linux (Ubuntu)

Best effort supporting the student laptops

All the software used will be Open Source except the services running in the Cloud

All the software generated will be Open Source

The public cloud used will be AWS (free account/tier, industry leader)