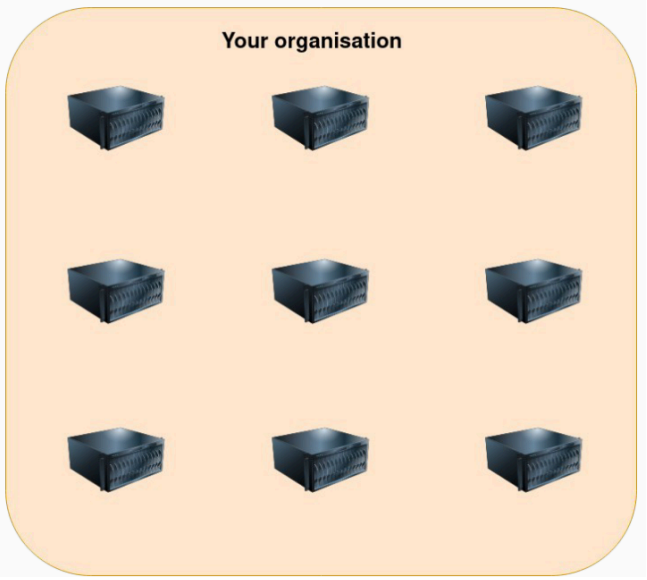
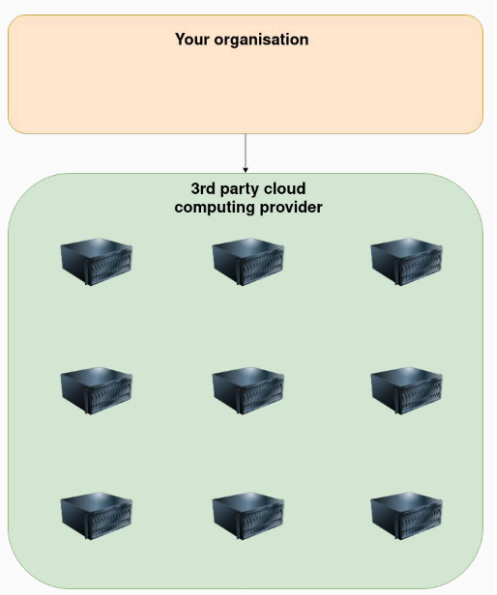
08. Cloud Computing & Software Testing

Cloud Computing

Cloud computing refers to services that allow users to buy compute or storage resources on-demand which can be accessed via the internet. Cloud computing is now the dominant technology for deploying things in the real world. Many systems run 100% on cloud systems (e.g. Spotify). Even big companies use the cloud to scale up and down quickly (e.g. Netflix and Meta).



The normal way of doing things



The cloud computing way of doing things

Advantages of cloud computing

- Quicker to get started
- Have the ability to scale up and down quickly
- Often cheaper (except at really large scales)
- Cattle not pets approach to managing servers: we can add and remove machines quickly, and adding a machine doesn't imply a commitments on our part.

- Programmatic control: cloud providers often provide APIs.

Disadvantages of cloud computing

- You are entirely at the mercy of a big company.
- May be expensive at enormous scales (few organizations are this big).
- You don’t have access to the physical hardware (pulling the power cable is never an option).

Monorepos

A monorepo is a single repository containing multiple distinct projects, with well-defined relationships.

Infrastructure-as-Code (IaC)

Describing your infrastructure in the form of source code.

What is IaC?

Infrastructure-as-Code (IaC) refers to using code written in forma languages to specify and provision computing infrastructure.

Examples of IaC tools outside the cloud: Packer, Vagrant, Docker, Kubernetes.

There are also IaC specifically geared towards infrastructure running remotely, particularly in the cloud.

Advantages of IaC

- The setup of the infrastructure becomes source code, and can be checked into version control just like any other code.
- Serves as an authoritative source of truth.
- Everyone with access to the code can see the configuration.
- Automates the (often complex) process of setting up and provisioning the development and operations environment.

Major IaC Platforms

Some commonly used IaC platforms used in networked and cloud environments.

AWS CloudFormation

It is an Infrastructure-as-Code system built into AWS, integrates tightly with AWS and works really well with their services. The downside is that is doesn't work with any other cloud provides, if you commit to CloudFormation, you might be committing to sticking with AWS forever.

Terraform

Made by HashiCorp (the same people that made Vagrant).

It is vendor neutral and the source is available, though it is not longer open source. It works with AWS, Azure, Google Cloud, OpenStack and more.

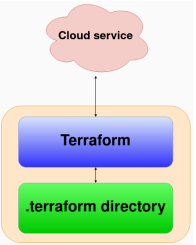
Terraform example

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "→ 3.27"
    }
  }

  required_version = ">= 0.14.9"
}

provider "aws" {
  profile = "default"
  region  = "us-west-2"
}

resource "aws_instance" "app_server" {
  ami           = "ami-830c94e3"
  instance_type = "t2.micro"
```

```
    tags = {
      Name = "ExampleAppServerInstance"
    }
}
```

The .terraform directory

The current state of the system and the providers are stored in a local database, which is stored in the **~/.terraform** directory. This presents a challenge when using Terraform from multiple machines, or with multiple users.



Terraform Cloud

To get around this problem, HashiCorp provide a cloud service to host people's .terraform directory in a centralized location. This service is called **Terraform Cloud**. This is free for up to five users at a time, but you will need to pay to scale beyond that.

Puppet

Uses a client-server architecture, a server controls agents running on target machines according to code/scripts written by the user.

Ansible

Similar to Puppet (used to provision existing machines over a networks), but uses a client application only. This connects to target machines using SSH.

Measurement

We need to keep track of the state of our cloud instances somehow.

Why do we need to measure?

- Performance monitoring
- Empirical evidence that changes are working (is the code running faster?)
- Detect growth and usage of the system
- Identify performance bottlenecks (which could indicate bugs)
- Profile the system

Measurement Tools

Widely used systems for measuring and monitoring systems in production

Nagios Core

A monitoring platform intended to be used in networked environments.

Prometheus

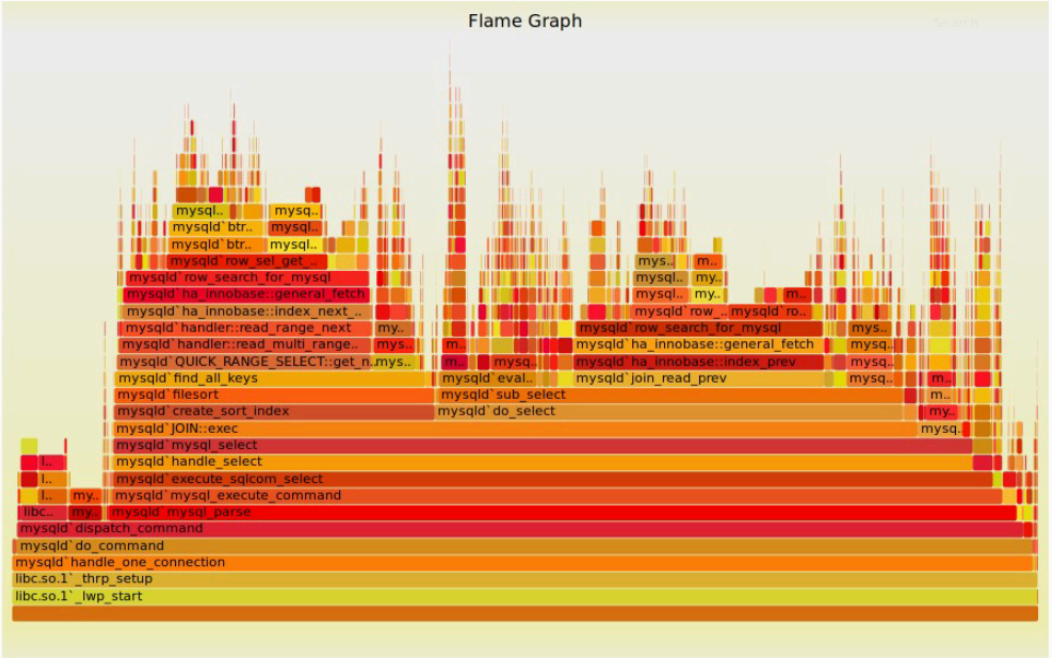
A programme for building a database of the state of the system, which can be queried at any time.

Grafana

Used to graph data provided by other applications. Often combined with Prometheus to build a full monitoring system.

Profiling

Show which functions were called in which order, and how much CPU time was spent in each function. This is very useful for identifying performance bottlenecks, or for understanding how code written by other people is structured.



Software Testing

Classifying testing by degree of internal knowledge

- **Black box testing:** not knowing anything about how the software works.
- **Grey box testing:** knowing some information about how the software works.
- **White box testing:** having detailed knowledge about how the software works.

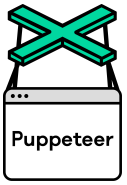
Classifying testing by level

- **Unit testing:** testing simple units of code (functions, classes, etc) at a time.
- **Integration testing:** testing multiple units of code at a time.
- **System testing:** testing the entire system.
- **Acceptance testing:** giving the system to users and allowing them to test it and give feedback.
- **A/B testing:** giving slightly different systems to different users to see which results in the most positive feedback.

Lots of tests can be automated and added to the monorepo.

Puppeteer

A Node application that automates the testing of web apps. Runs Chrome in the background (in headless mode) and allows the user to control it via JavaScript.



JUnit

A widely used testing library for Java and other JVM languages.

Other types of testing

- **Fuzzing:** feeding random input to a system to see if it breaks things.
- **Regression testing:** running tests to detect old bugs to see if they have accidentally been re-introduced.
- **Non-functional testing:** testing the non-functional parts of a system, e.g. speed, startup time, usability, aesthetics.
- **Smoke testing:** basic sanity checks before more detailed testing, e.g. verify the system doesn't crash when it starts up.