Infrastructure as Code (IaC)

CMXXXX
Lecture XX - IaC part 1
Dom Routley

Objectives

- Know what is IaC
- Know why we use IaC

IaC part 1

What is IaC

Definition

- Infrastructure as Code
- Defined by Microsoft as "*...the management of infrastructure in a descriptive model...*"

Infrastructure ...

- Networks
- Virtual machines (Instances)
- Load balancers
- Connection topology

… as Code

```
resource "openstack_compute_instance_v2" "myWebServer" {
  image_name      = "Ubuntu Server 18.04.1"
  flavor_name     = "m1.micro"
  name            = "Webserver"
  key_pair        = "myKeyPair"
  network {
    name = "myNetwork"
  }
}
```

A more precise definition

Infrastructure as Code is is the method of managing and provisioning cloud infrastructure through descriptive definition files.

Idempotence

- In cloud operations, idempotence is a property of a function/computer operation that can be applied multiple times without the result changing.
- It is vital for the correct implementation of IaC that the output infrastructure that is created is always going to be the same.

Cloudformation

- AWS Cloudformation is the IaC syntax used by AWS
- Cloudformation is not a standard, it is a (free open source) product
- Openstack’s IaC is closely resembles Cloudformation

IaC software tools

Cloudformation is complex to write and syntactically picky. It is human readable, but only just. It is good for storage and referral for infrastructure, but not so good if you want to write some.

This is where IaC software tools come in. They convert much easier to write code into Cloudformation (or other such IaC code) and deploy it onto a cloud.

Some IaC software that will be covered by this module:

- Ansible
- Chef
- Puppet
- Terraform

Management vs provisioning

IaC is sometimes taken to mean the files that just make up the (virtualized) hardware in the infrastructure, with the files that provision the instances (run commands in the OS) just called the provisioning files and not included in IaC.

In this module we will assume that both provisioning files AND hardware infrastructure management files are included under IaC.

It is however important to make the distinction as we will be using two different pieces of software, (one for the hardware infrastructure management and one for the provisioning of the instances), in the coursework.

Why we use IaC

Reasons to use IaC

- Remove snowflake environments
- Easy creation of production-like environments for testing
- Easy validation of infrastructure
- Infrastructure can be managed with code version control software

Snowflake environments

“Infrastructure as Code evolved to solve the problem of environment drift in the release pipeline. Without IaC, teams must maintain the settings of individual deployment environments. Over time, each environment becomes a snowflake, that is, a unique configuration that cannot be reproduced automatically. Inconsistency among environments leads to issues during deployments. With snowflakes, administration and maintenance of infrastructure involves manual processes which were hard to track and contributed to errors.”

- Sam Guckenheimer (Microsoft Azure DevOps Team)

Testing environments

Using Infrastructure as Code allows us to generate as many identical environments as we want at once.

This means that you can generate multiple testing environments from the same code as the production live environment and be confident that if something works in testing then it will also work in production.

Infrastructure validation

Using syntax checking and other methods of checking code files, it is possible to validate changes you are going to make to an infrastructure before you have made the change.

Version control

Usually environments are not version controlled.

However as IaC stores the entire environment in plain text files, it is easy to version control using existing version control software.