Infrastructure as Code & Provisioning

Software Engineering II
WS 2020/21

Enterprise Platform and Integration Concepts

Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

IT Infrastructure

Information Technology you need to run an application

- First 3 OSI model layers are primary components
- **The physical building** computers are located in
- **Physical equipment**, bare-metal servers & network switches
- **Connection topology** & load balancers
- **Virtual Machines**

Possibly (depending on who you ask) also:

- Hypervisors and operating systems
- Application runtimes & dependencies

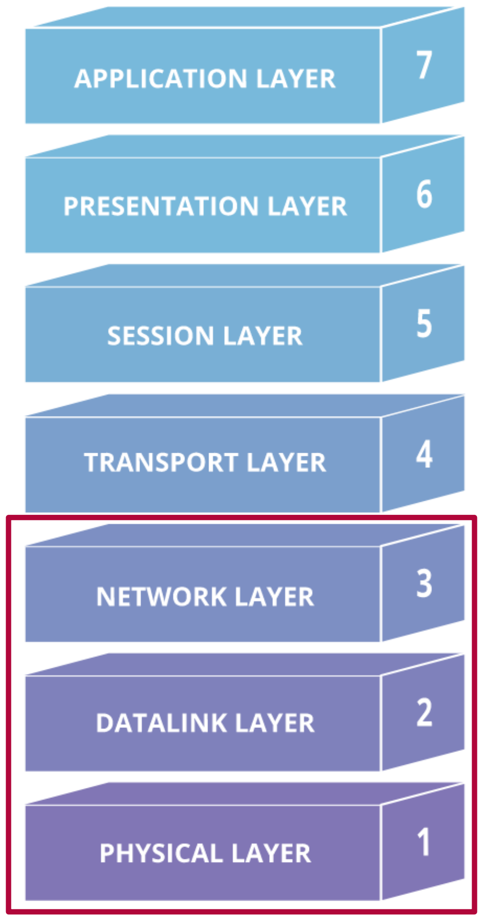


Image: https://www.cloudflare.com/learning/ddos/glossary/open-systems-interconnection-model-osi/

IT Infrastructure as Code Definitions

“IT abstraction where professionals **provision** and **manage** a technology stack **with software**, rather than setting up hardware systems”

— *https://www.techopedia.com/definition/33553/infrastructure-as-code-iac*

“managing and provisioning computer data centers through **machine-readable definition files**, rather than physical hardware configuration or **interactive configuration tools**”

— *Amazon Web Services in Action. Wittig, Michael. 2016. Manning Press.*

IaC Related Terms

Software-Defined Environment

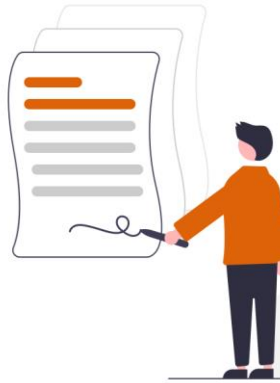
- **Software-Defined Infrastructure**: used mostly synonymously to IaC
- **Software-Defined Networking** (SDN): programmable, centralized control of network traffic without manual access to physical components

Infrastructure-as-a-Service (IaaS)

- **Renting** servers, storage, network on a pay-per-use pricing model
- The IaaS provider (most likely) makes use of IaC tools

Cloud computing

- Someone else's computer
- They set it up the way you want to (defined by code)
- Providers may make use of IaC & SDN concepts and tools

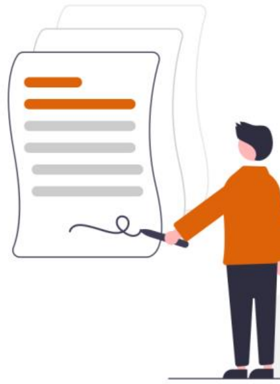


IaC Related Terms

Provisioning

- Process of setting up IT infrastructure
- Not the same as configuration, both are steps in **deployment process**
- Configuration is the next step after provisioning

- The term "provisioning" can mean:
  - Server provisioning
  - Network provisioning
  - User provisioning
  - Service provisioning



Source: https://www.redhat.com/en/topics/automation/what-is-provisioning

Infrastructure as Code
Core Idea

Treat IT infrastructure configuration as software

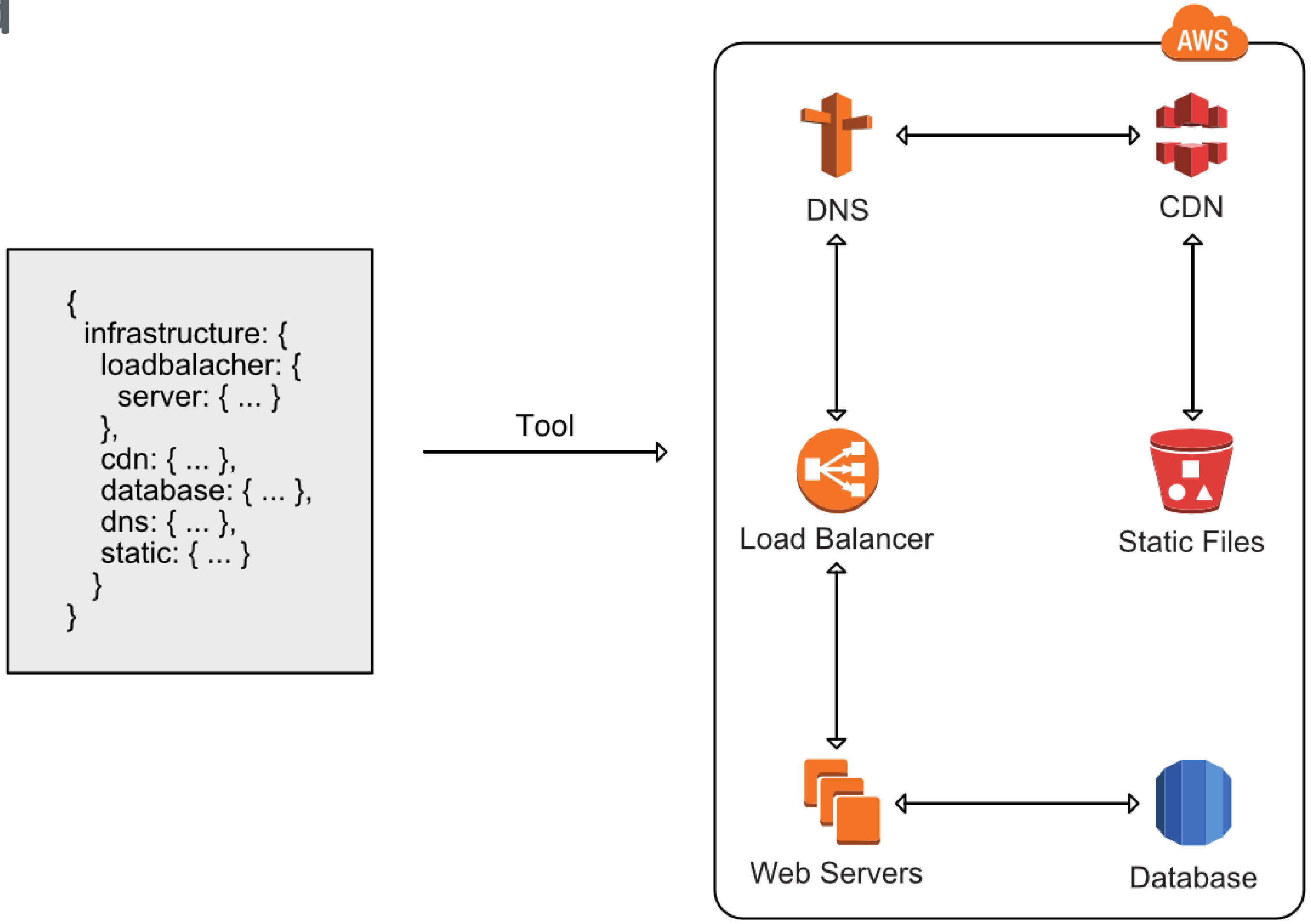
- Write code to define, deploy & destroy basic infrastructure
- Make changes to code, then apply them to systems

- We already have engineers who know how to code
- We already have processes for writing & controlling software
- Allow computers to do what they do best (automation) and developers what they do best (coding)

Of course, someone, somewhere needs to build & operate datacenters

Source: https://docs.microsoft.com/en-us/azure/devops/learn/what-is-infrastructure-as-code

Infrastructure as Code
Core Idea



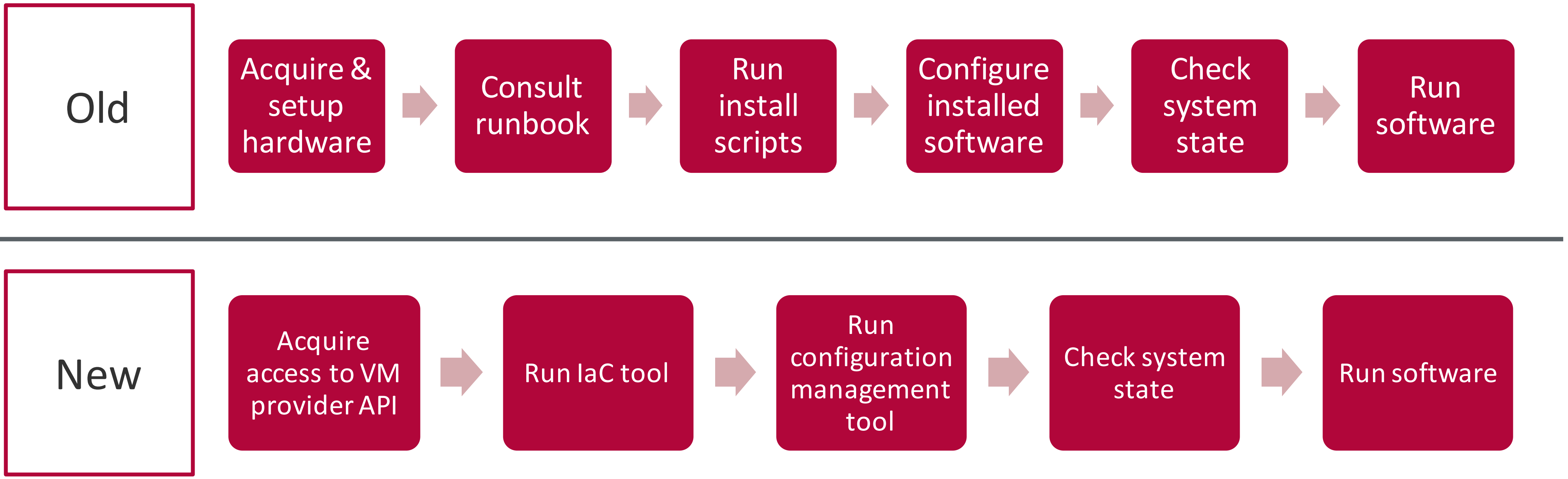
```
{
  infrastructure: {
    loadbalacher: {
      server: { ... }
    },
    cdn: { ... },
    database: { ... },
    dns: { ... },
    static: { ... }
  }
}
```

Image: https://cloudonaut.io/understanding-infrastructure-as-code/

Why Infrastructure as Code?

IT infrastructure is critical.

Without it running, the quality of the application is irrelevant!



Why Infrastructure as Code?

IT infrastructure is critical.

Without it running, the quality of the application is irrelevant!

Problems of manual provisioning

- **Cost & effort** of hiring engineers and maintenance technicians
- **Environment drift & snowflakes**
  - Teams maintain custom deployment infrastructure
  - Over time, teams' environments diverge -> **inconsistency**
- Manual processes may lead to **errors** -> "works on my machine"
- **Scalabilty** and **availability** as manual configuration is slow

Infrastructure as Code Benefits

Self-Service

- Don't solely rely on sysadmin with deployment access
- **Leveling access** to knowledge (contained in tools)
- **Automation**, development teams can kick off own environments
- Motivation to set up different environments and sandboxes

Speed & Safety

- Automated scripts are faster than humans using interactive tools
- Computers tend not to miss/forget steps -> **repeatability**
- Automated scripts vs. "copy/paste the following shell commands"

Further reading: Terraform: Up & Running, 2nd Edition. Yevgeniy Brikman. 2019. O'Reilly Media, Inc.

Infrastructure as Code Benefits

Documentation

- **One location** to find entire required IT architecture
- Everyone understands how things work (even if sysadmin is on vacation)
- Define IT architecture for different dev stages

- Declarative vs. Imperative code
  - Declarative: Express end state without control flow
  - Imperative: Statements that explicitly change (a program's) state
- Declarative descriptions allow flexible operations,
  e.g. orchestrating them in the most efficient way

Infrastructure as Code Benefits

Consistency

- Configuration files as **single "sources of truth"**
- **Standardize** components across organization
- **Reusability**: documented, battle-tested modules
- **Idempotency**: guaranteed end configuration, regardless of environment's starting state

Validation

- (Infrastructure) code changes can be **reviewed**
- Code can be automatically **tested**
- **Static analysis** tools for (infrastructure) code

Idempotency can be achieved by configuring target or discard & recreate. Which is faster?

Infrastructure as Code Benefits

Version Control

- History of infrastructure is captured -> **traceability and documentation** of changes
- Roll back changes if errors are detected
- Identify commit that introduced the bug
- Edit the source, not the target

Version control can help in debugging.
Check out `git bisect`.

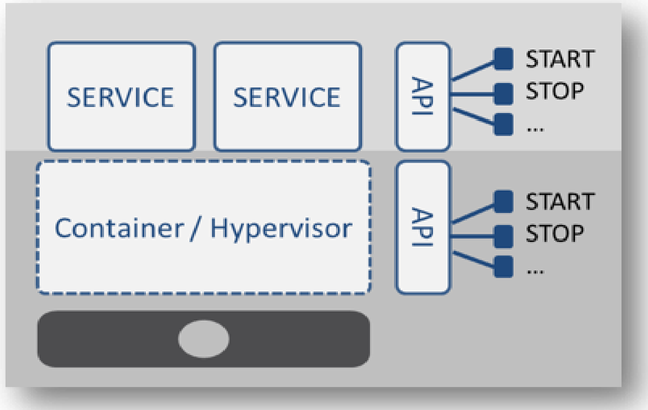
Cost reduction

- Reduced cost of infrastructure management & maintenance
  - Infrastructure providers have economies of scale

Infrastructure as Code Tools

Configuration Management ≠ Provisioning

- **Tools and their capabilities often overlap**
- **Provisioning**: create, modify and/or delete infrastructure using code and/or APIs.
  - Example: Create 100 AWS EC2 instances in a single subnet
- **Configuration management**: Express state of your infrastructure with code and/or APIs.
  - Example: Ensure tmux is installed on all servers



Configuration

Provisioning

Sources: https://www.slideshare.net/CarlosNunez37/configuration-management-and-provisioning-are-different-79287391, https://devops.com/provisioning-vs-configuration/

Infrastructure as Code Tools

An application's requirements defines the required infrastructure.
The type of infrastructure and the application's purpose defines the required automation.

Ad Hoc Scripts

- Write code (using APIs) that sets up infrastructure
- Use any (scripting) language
- Fast to write, fast to change
- Custom code for every task -> inconsistency between scripts
- Great for small projects,
  messy for large amount of servers, databases, etc.

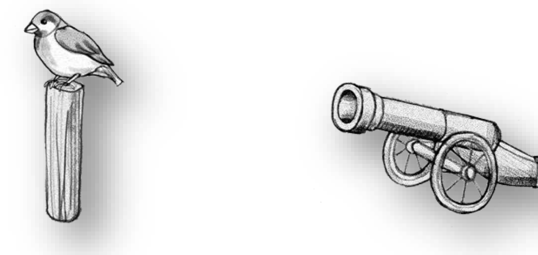


Image: https://www.praxis-breitenberger.de/2016/03/allergien/

Infrastructure as Code Tools

Configuration Management Tools

- Install and manage software on existing servers
- Not only for one but any amount of servers
- Most functionalities are idempotent
- e.g. Chef, Puppet, Ansible

Server Templating Tools

- Create an image as virtual machine or container of the OS, software, and files
- Install the image on all servers using an IaC tool
- e.g. Docker, Packer, Vagrant

Infrastructure as Code Tools

Orchestration Tools

- Manage VMs and containers
  - Deploy them
  - Roll out updates
  - Scale the number of images (auto scaling)
  - Distribute traffic (load balancing)
- e.g. Kubernetes, Amazon ECS, Docker Swarm

Infrastructure as Code Tools

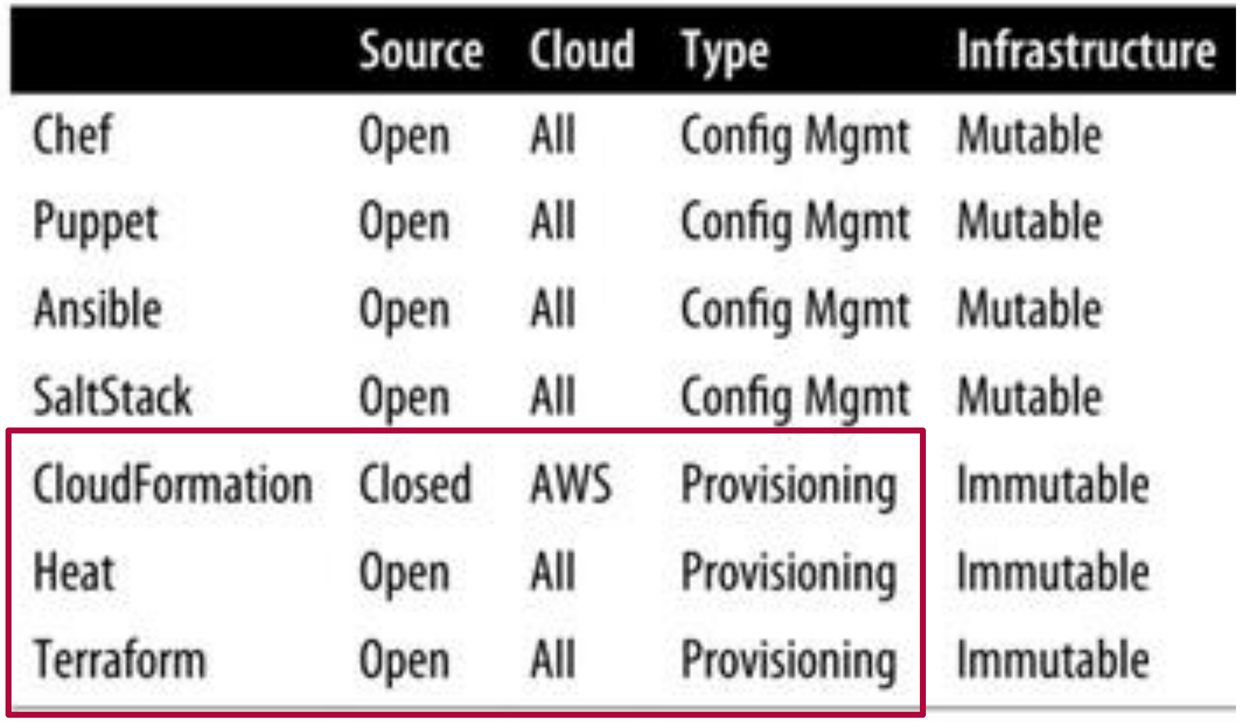
Provisioning Tools

- Create needed IT infrastructure
  (e.g. servers, databases, caches, monitoring)
- e.g. Terraform, Cloud-Formation

Mix and match your set of tools

Infrastructure as Code Tools

Most common ways to use popular IaC tools:



	Source	Cloud	Type	Infrastructure
Chef	Open	All	Config Mgmt	Mutable
Puppet	Open	All	Config Mgmt	Mutable
Ansible	Open	All	Config Mgmt	Mutable
SaltStack	Open	All	Config Mgmt	Mutable
CloudFormation	Closed	AWS	Provisioning	Immutable
Heat	Open	All	Provisioning	Immutable
Terraform	Open	All	Provisioning	Immutable

Types

- **Immutable**: Replace servers with new ones for updates
- **Mutable**: Modification possible after provisioning

This paper considers both config management and provisioning tools as "IaC"

M. Guerriero, M. Garriga, D. A. Tamburri and F. Palomba, "Adoption, Support, and Challenges of Infrastructure-as-Code: Insights from Industry," 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), Cleveland, OH, USA, 2019, pp. 580-589, doi: 10.1109/ICSME.2019.00092

Popular IaC Tools

“ IaC technology ecosystem is currently very **scattered**, **heterogeneous** and **not fully understood**, with no single tool dominating the market ”

—Guerriero et al., 2019

M. Guerriero, M. Garriga, D. A. Tamburri and F. Palomba, "Adoption, Support, and Challenges of Infrastructure-as-Code: Insights from Industry," 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), Cleveland, OH, USA, 2019, pp. 580-589, doi: 10.1109/ICSME.2019.00092

IaC Example: Terraform

- Manage services deployed on different external infrastructure
  - Public and private clouds
  - VMs & containers
  - Network appliances
  - Software/Platform as a service
- Declarative configuration describing final state
- Parallel creation of non-dependent resources
- Modules for reuse and maintainability

IaC Example: Terraform

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 2.70"
    }
  }
}

provider "aws" {
  region = "us-east-1"
}

# # 1. Create vpc
resource "aws_vpc" "dev-vpc" {
  cidr_block = "10.0.0.0/16"
  tags = {
    Name = "dev"
  }
}
```

"Virtual Private Cloud" is Amazon's virtual network service

https://levelup.gitconnected.com/terraform-hello-world-b4985162cff2

IaC Tool Challenges

Testability

- Testing declarative code
- Verify definition syntax

Maintainability

- Specialised domain specific languages
- Possibly lacking IDE support

Portability

- Move definitions between IaC tools

M. Guerriero, M. Garriga, D. A. Tamburri and F. Palomba, "Adoption, Support, and Challenges of Infrastructure-as-Code: Insights from Industry," 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), Cleveland, OH, USA, 2019, pp. 580-589, doi: 10.1109/ICSME.2019.00092

Infrastructure as Code

“IT Infrastructure includes all of the Information Technology but **not** the associated **people, Processes and documentation**

— Information Technology Infrastructure Library's (ITIL) Foundation Course Glossary https://itil.it.utah.edu/downloads/ITILV3_Glossary.pdf”

ITIL v3 is from 2011

- **People** as code?
- **Processes** as code?
- **Documentation** as code?

Summary

Infrastructure as Code & Provisioning

- Definition of IT infrastructure
- Core ideas of Infrastructure of Code (IaC)
- Provisioning
- Related Terms
- Motivation & Benefits
- Different types of tools & challenges
- Future of IaC

Further Reading

- Terraform: Up & Running, 2nd Edition. Yevgeniy Brikman. 2019. O'Reilly Media, Inc.
- Infrastructure as Code. July 2017. AWS Whitepaper
  https://d0.awsstatic.com/whitepapers/DevOps/infrastructure-as-code.pdf
- Peter Souter, "Provisioning vs Configuration Management Deployment vs Orchestration"
  https://archive.fosdem.org/2018/schedule/event/deployment_provisioning_orchestration/
- M. Guerriero, M. Garriga, D. A. Tamburri and F. Palomba, "Adoption, Support, and Challenges of Infrastructure-as-Code: Insights from Industry," 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), Cleveland, OH, USA, 2019, pp. 580-589, doi: 10.1109/ICSME.2019.00092

Acknowledgements

- Special thanks to Wanda Baltzer for work on this content!