Infrastructure as Code

What is IaC, Provision and Configuration Management tools

Technical Trainers

SoftUni Team

Software University

SoftUni

Software University

https://softuni.bg

sli.do

#Dev-Ops

Table of Content

1. Infrastructure as Code

2. Terraform

3. Configuration Management Tools

Infrastructure as Code

Automating Infrastructure Management Using Code

What is IaC?

- **Infrastructure as Code** (**IaC**) is the managing and provisioning of **infrastructure through code** instead of through manual processes
  - As VMs, networks, OS servers, storage, etc.
- **IaC** involves
  - Writing **code** to define the desired state of an **infrastructure environment**
  - Using **tools** to **automatically** deploy and configure the environment based on the code

IaC Configuration Files

- **IaC** is a form of configuration management that **codifies infrastructure resources** into text files

- **Configuration files** are created with your infrastructure specifications

  - Should be **version controlled** and **tested** (unit, integration, ... tests)

  - Ensure that you provision the **same environment every time**

  - Allow you to divide your infrastructure into **modular components** and combine them through automation

  - Should contain always **up-to-date infrastructure documentation**

What Do You Need for IaC?

- **Remote accessible hosting** or **IaaS cloud hosting platform**
  - IaC tools connect and modify remote host
  - IaaS cloud hosting platforms have an **API** for modification of infrastructure resources
- **Provisioning tool**
  - Automates the infrastructure deploy and management
- **Configuration management tool**
  - Manages infrastructure state
- **Version control system**
  - Stores text files used by the CM platform

Approaches to IaC
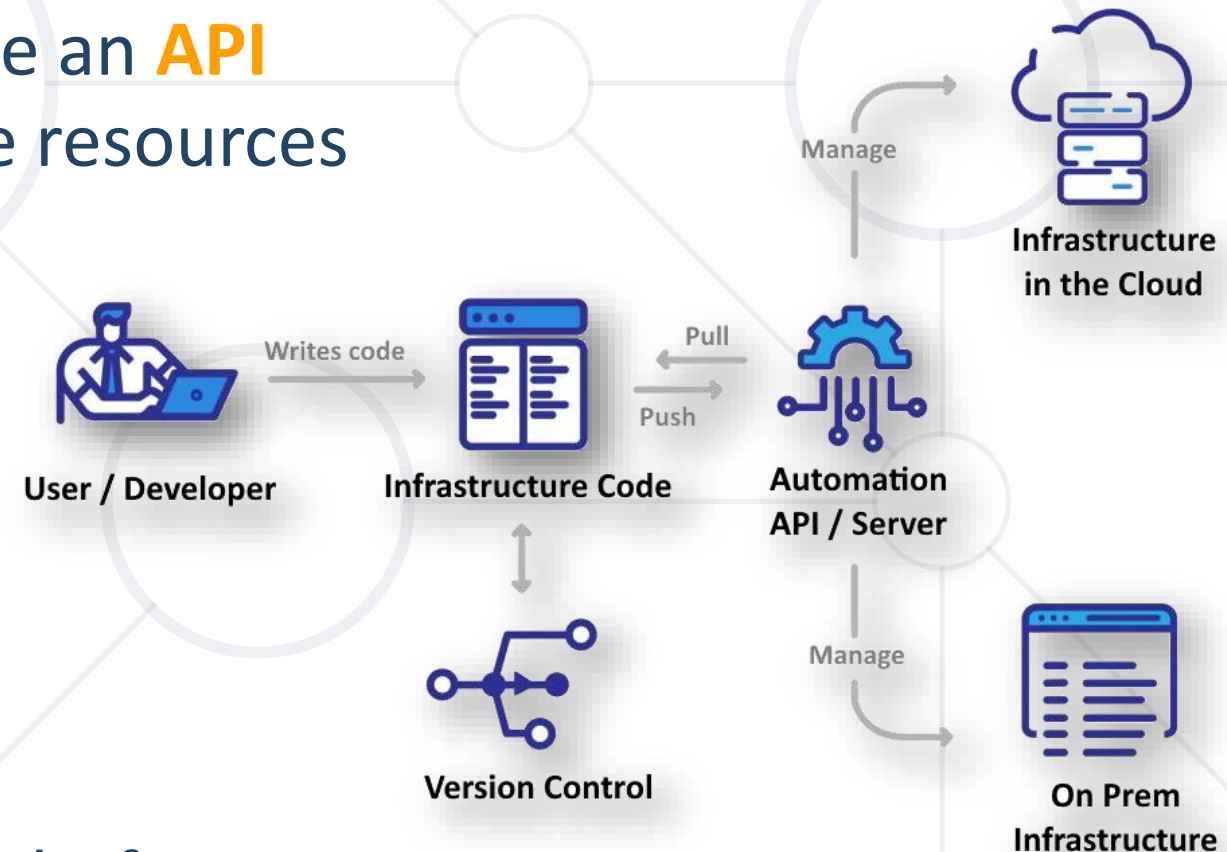
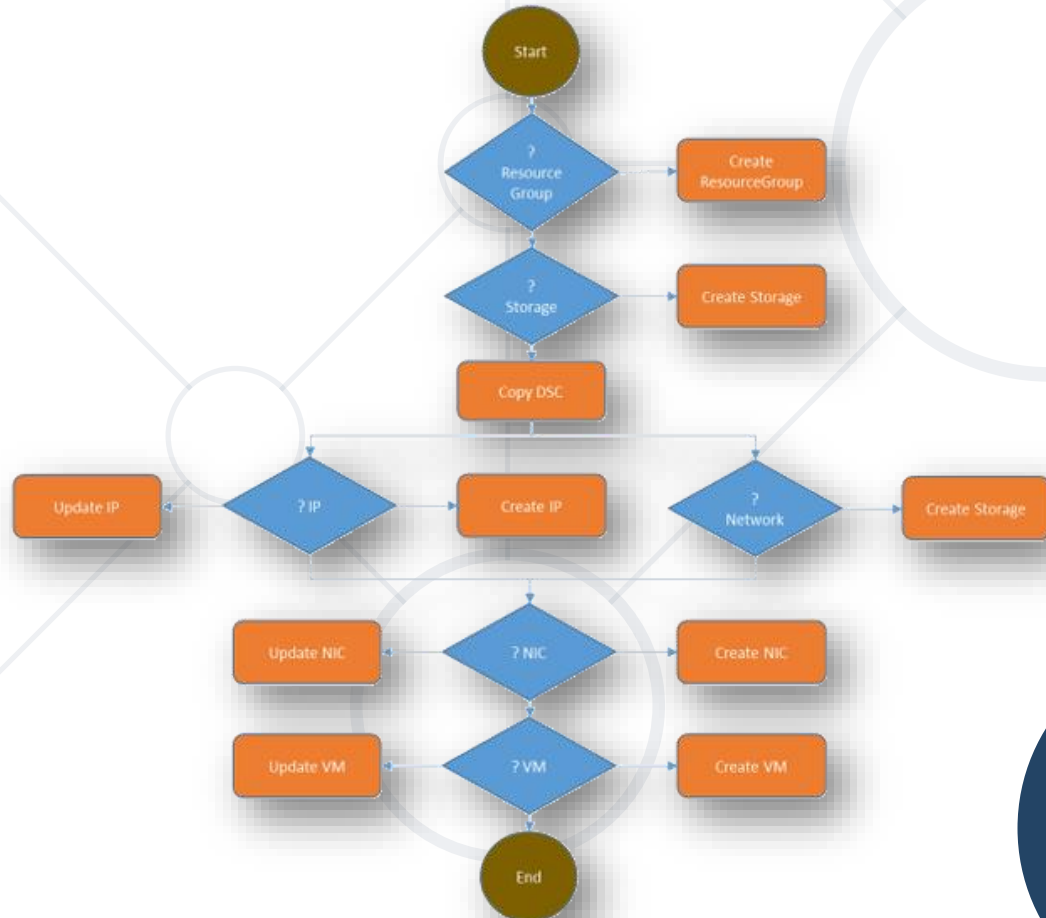- **Imperative approach**

  - Tell the system **how** to do something every step of the way

  - Defines the **specific commands** to be executed in a **specific order** for the **desired configuration**
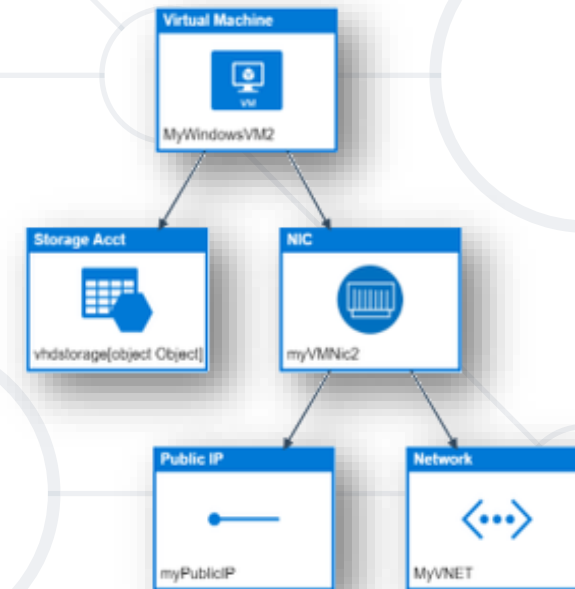
- **Declarative approach**

  - Tell the system **what** you want and let it figure out how to do it

  - Defines the **desired state** of the **system** – resources, their properties and an IaC tool for configuration

Imperative vs Declarative Approach

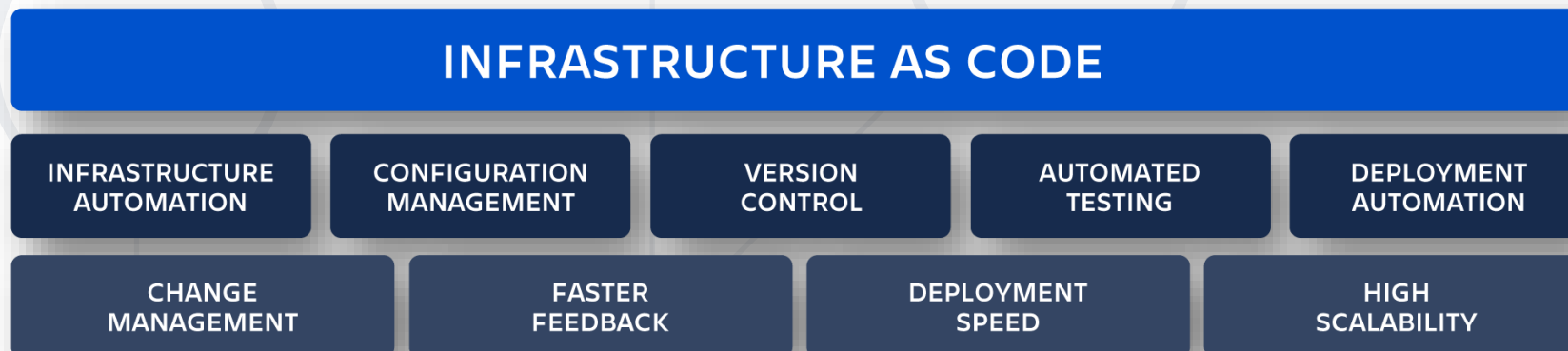- Imperative Approach

- Declarative Approach

IaC Tools

- The primary goal of **IaC tools** is to **bring the infrastructure component** to the **desired state** declared by the user

- **IaC tools** fall into two categories

  - **Infrastructure provisioning tools**

    - Create infrastructure components

  - **Configurations management tools**

    - Configure provisioned servers

Infrastructure Provisioning Tools

- **Infrastructure provisioning**

  - Create infrastructure resources like virtual servers, storage, networking, cloud managed services, etc.

- Primary goal

  - **Keep the infrastructure in its desired state** and reproduce or update it

- Tools

  - Terraform, AWS Cloudformation, Azure Resource Manager (ARM) Templates, Pulumi

- They can also **trigger CM tools**

Configuration Management Tools

- **Configuration management**
  - Configuring infrastructure resources
    - e.g., configuring a server with required applications or configuring a firewall device
- Primary goal
  - **Configure the server**
- Tools
  - Ansible, Chef, Puppet, SaltStack, etc.
- In **cloud environments**, tools use an **API-based dynamic inventory** to get the server details

IaC Benefits for DevOps

- **IaC** is an important part of implementing **DevOps practices**
  - **Version control**, **test** and **deploy** of infrastructure code changes
  - **Improved collaboration** – Ops team can participate in writing IaC templates together with Dev team, as IaC uses simple, text-based files
  - **Automation** of creation and management of infrastructure resources
  - **Consistency and reliability across environments** is achieved as IaC generates the same environment every time

INFRASTRUCTURE AS CODE				
INFRASTRUCTURE AUTOMATION	CONFIGURATION MANAGEMENT	VERSION CONTROL	AUTOMATED TESTING	DEPLOYMENT AUTOMATION
CHANGE MANAGEMENT	FASTER FEEDBACK	DEPLOYMENT SPEED		HIGH SCALABILITY

Terraform

IaC Tool for Infrastructure Provisioning Automation

Terraform Overview

- Open-source **IaC** tool
  - Used for **provisioning**, **managing** and **deploying** infrastructure resource
  - Written in Golang
- Allows managing infrastructure for applications across **multiple cloud providers** – AWS, Azure, GCP, etc.
  - Through their application programming interfaces (**APIs**)
- Uses **declarative syntax** – you define desired infrastructure state, Terraform figures out the best way to achieve it

Terraform Workflow

- To **deploy infrastructure with Terraform**

  - Scope – **identify the infrastructure** for your project

  - Author – define **infrastructure** in **configuration files**

  - Initialize – **install the plugins** Terraform needs to manage the infrastructure

  - Plan – **preview the changes** Terraform will make to match your configuration

  - Apply – Terraform **provisions the infrastructure** and **updates state file**
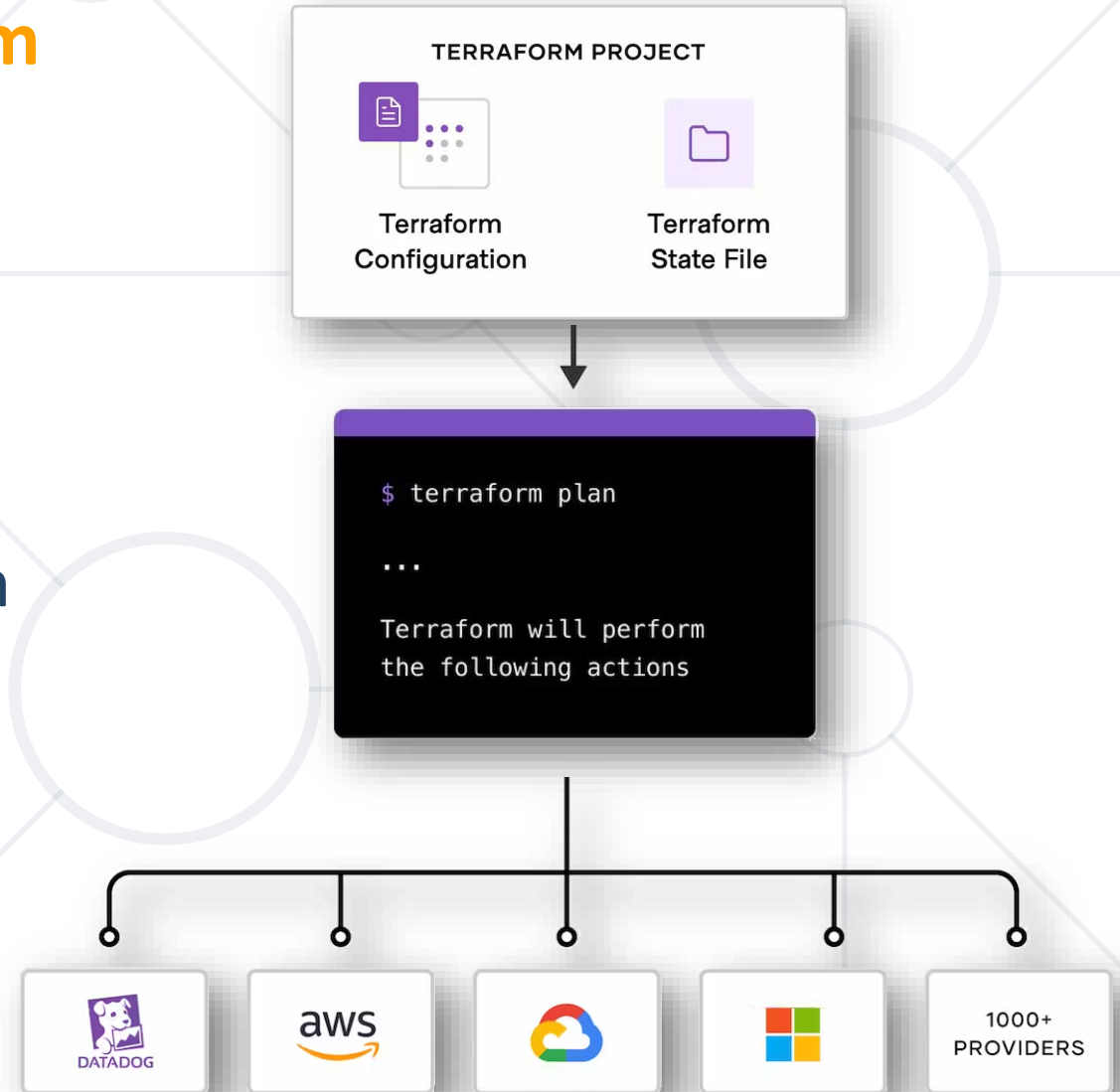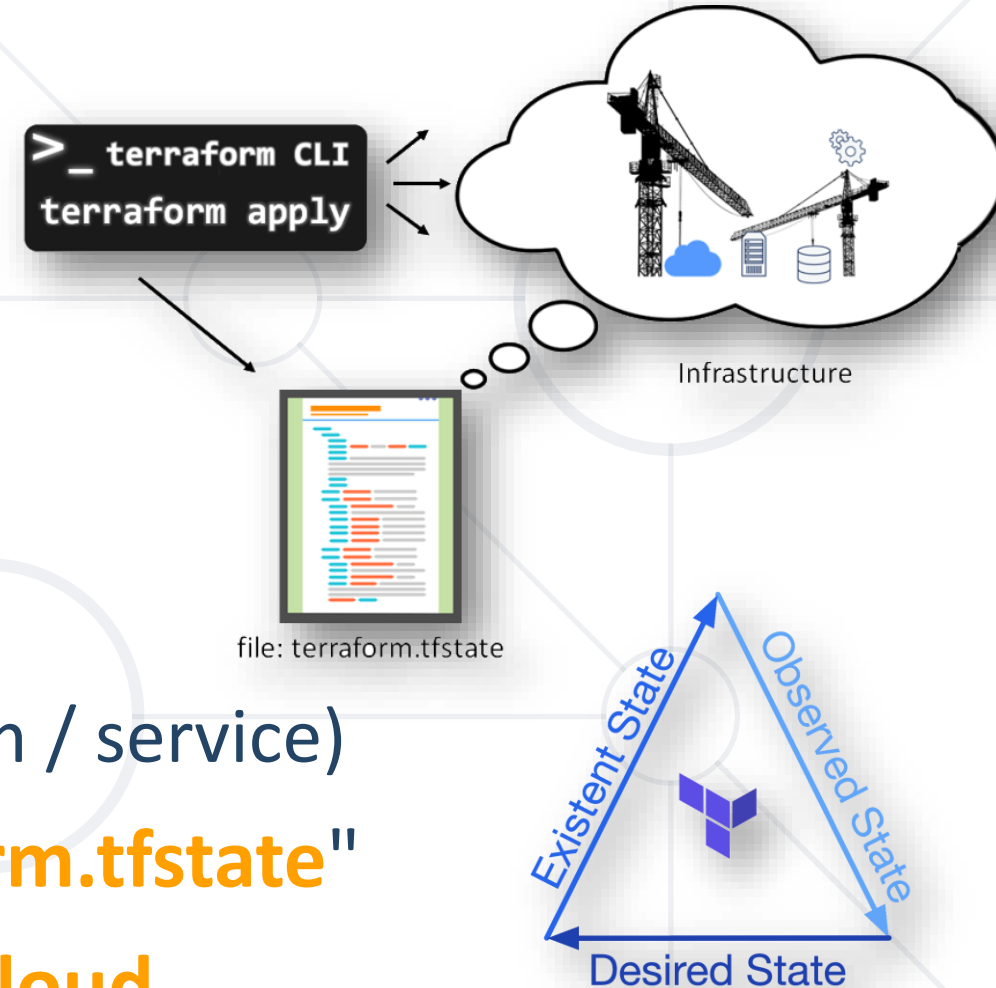
Terraform Configuration File

- To **create an infrastructure**, a **Terraform Configuration file** (**.tf**) should be executed

- Executed with the help of **Terraform CLI** or other executors

- Written in HashiCorp Configuration Language (**HCL**) or **JSON syntax**

```
provider "aws" {
  region = "us-west-2"
}

resource "aws_s3_bucket" "bucket_name" {
  bucket = "bucket-name"
}
```

Terraform Configuration File (HCL)

terraform CLI

Terraform CLI

Infrastructure Automation/Provisioning with Terraform

Terraform State File

- Terraform stores **state** about managed infrastructure and configuration

- **State** allows us to have a point-in-time **view of our infrastructure** and **compare**

  - **Desired state** (our code)

  - **Perceived state** (the state file)

  - **Reality** (the resources within the platform / service)

- This state is stored in a **local file** "**terraform.tfstate**"

- State file is recommended to be **kept in cloud**

- State file format is **JSON**, but should not be edited directly

file: terraform.tfstate

Infrastructure

>_ terraform CLI
terraform apply

Existent State

Observed State

Desired State

18

Live Demo

Installing Terraform

Live Demo

Terraform and Docker:

Provision a NGINX Server

Configuration Management Tools

Ansible

- Open-source infrastructure **automation tool**
  - Written in Python
- Focuses on **security** and **reliability**
  - Uses OpenSSH
- Easy to read and write
  - Uses **YAML**
  - Structured
- **Agentless**
  - No agents, repositories, etc.

Ansible – Key Features

- Powerful tool for **managing Infrastructure as Code**
- **Declarative**
- **Idempotent**
  - Run an operation multiple times, without changing the initial state of the application
- Three major **use cases**
  - Inventory (Provision)
  - Configuration management
  - Application deployment

Puppet

- **Configuration management** tool for **servers**
  - Ensures all systems are **configured** to the **desired states**
- Also used as a **deployment** tool
- Uses **server-agent model**
- Configurations are written in Puppet code
  - Ruby DSL
- Open Source and Enterprise

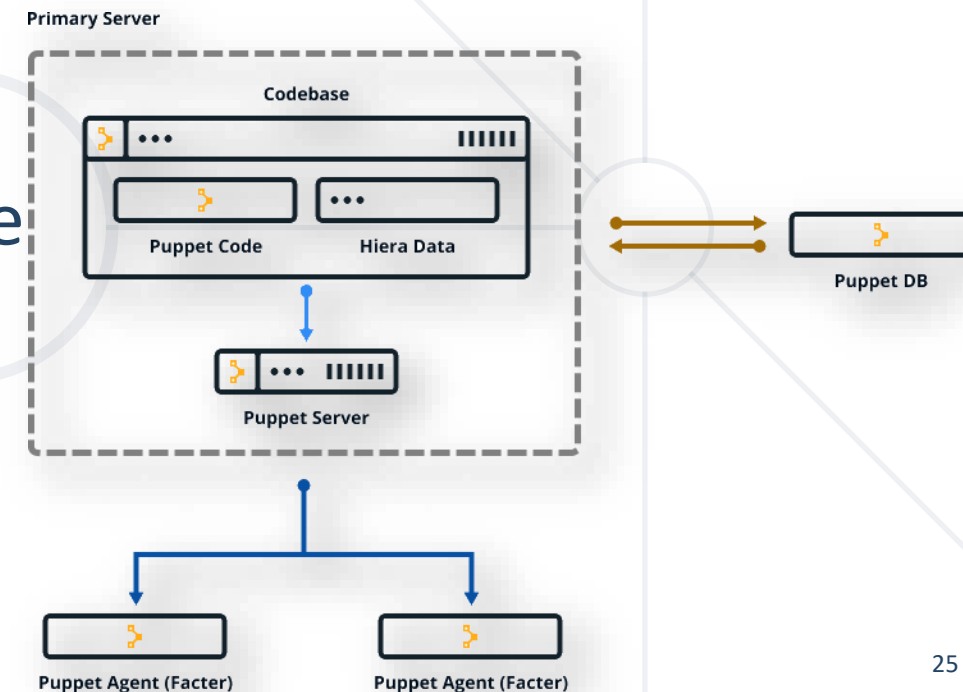# Puppet – Platform

- **Puppet Server**
  - Controls configuration for one or more managed nodes
  - Communicate via HTTPS with the agents
  - Has a built-in certificate authority
  - Runs an agent to configure itself
- **Puppet Agent**
  - **Facter** → gather information about a node
  - **Hiera** → separate the data from the code
- **Puppet DB**
 - Stores facts, catalog, reports, etc.

SaltStack

- **Management** tool
  - Used for **configuration management**, data-driven orchestration and remote execution
- Two **operation modes**
  - With agents (minions)
  - Agent-less
- Management instructions in **YAML**

Salt Master

- **Salt master**
  - The machine that **controls** the infrastructure and dictates policies for the servers it manages
  - Operates as
    - A **repository** for **configuration data**
    - A **control center**
      - Initiates **remote commands**
      - Ensures the **state** of other machines

General Salt Architecture
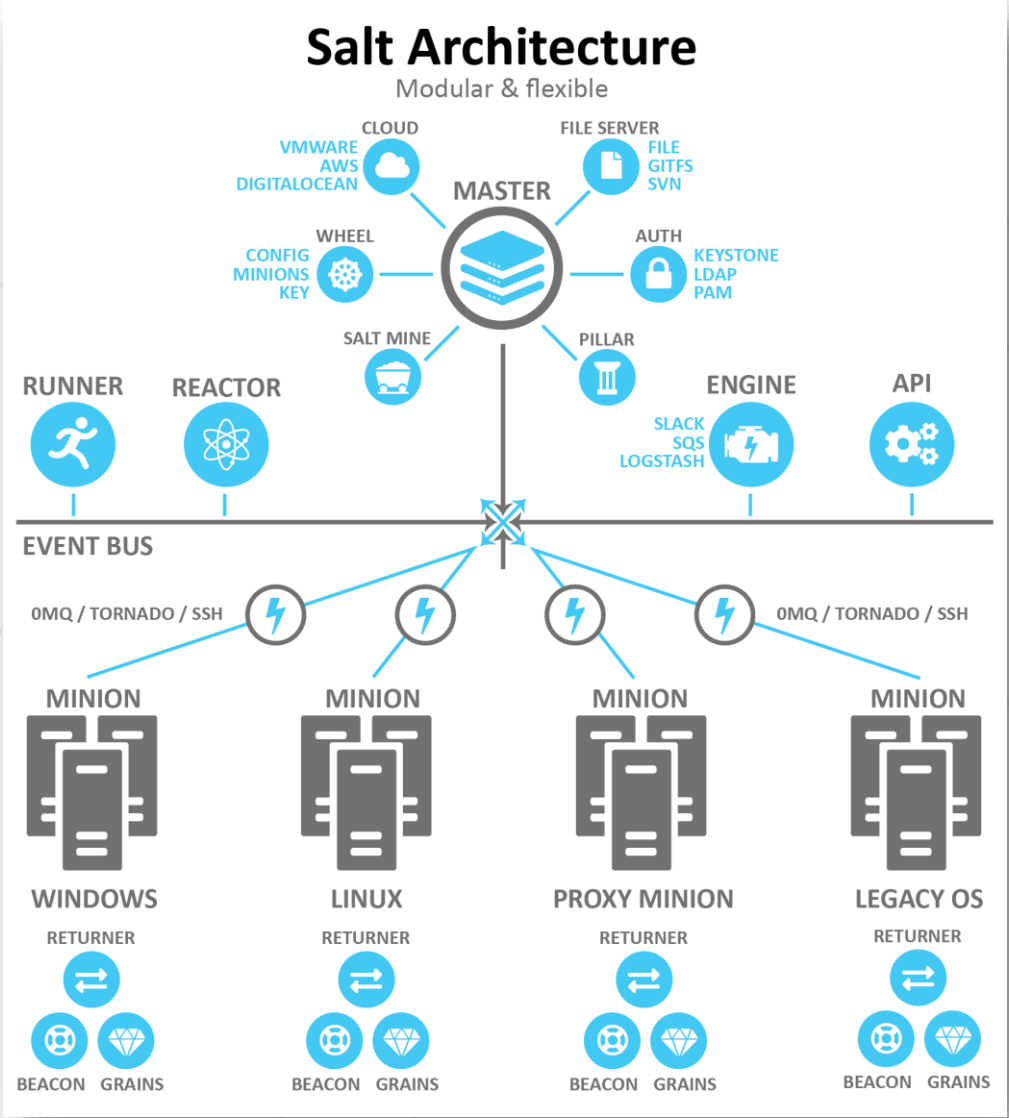
Chef

- **Configuration Management** tool
  - Written in Ruby and Erlang
  - Uses pure-Ruby DSL
- Works with system configuration "**recipes**"
- Used for **configuring** and **maintaining** servers
- Can be integrated with **cloud-based platforms** to automatically provision and configure new machines
- **Chef Infra** → **configure** and **manage** infrastructure

Chef Infra

- Policy-based **configuration management** tool
  - Define and enforce desired state of systems
- Uses the **master-agent** model
- "**Recipes**" are contained in "**cookbooks**"
  - Manage configuration, software installations and system updates

Summary

- **Infrastructure as Code** (**IaC**) uses DevOps practices and versioning with a descriptive model to **define** and **deploy infrastructure**

- **Terraform** is an IaC **provisioning** tool used to create infrastructure

- **Ansible**, **Puppet**, **Salt** and **Chef** are **configuration management** tools used to configure provisioned servers

Questions?
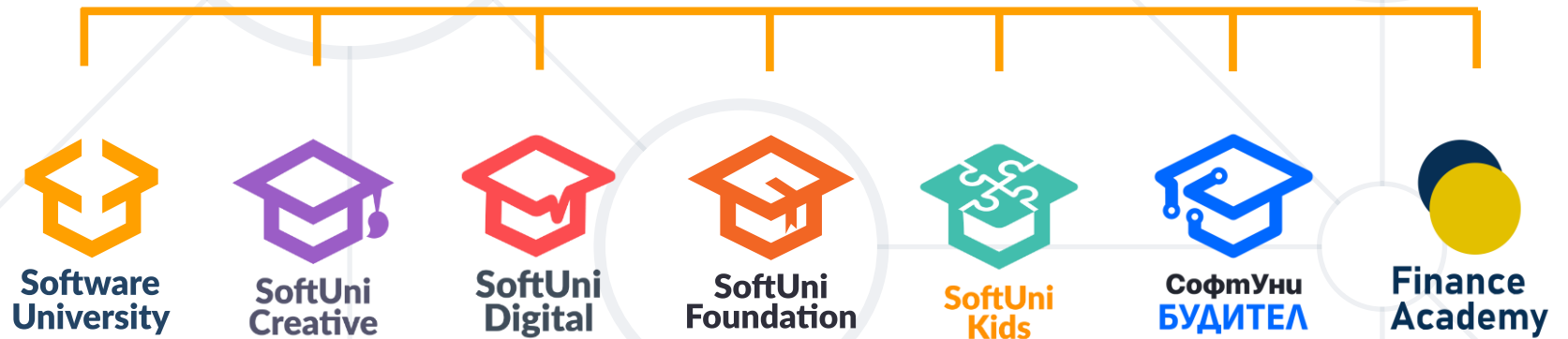
SoftUni Diamond Partners

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers

  - softuni.bg, about.softuni.bg

- Software University Foundation

  - softuni.foundation

- Software University @ Facebook

  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**

- Unauthorized copy, reproduction or use is illegal

- © SoftUni – https://about.softuni.bg/

- © Software University – https://softuni.bg