Infra As Code

DevSecOps

Agenda

- Infra as Code Principles
- Terraform
- Security Analysis

Infra As Code Principles

Secure DevOps

What is Infra As Code (IaC)?

- Infrastructure as code is the approach to defining your infrastructure through source code that can then be treated just like any software system
- Infrastructure can be computing (like VMs), networking, security and any cloud managed service and resource (like Kubernetes clusters, serverless, etc.)
- This code (as any type of code) must be kept in source control to allow auditability, versioning all full integration with CI/CD
- Natural practice with cloud computing but can be use on several on-prem virtual environments

IaC: Benefits

- Faster and easier way to provisioning, validate and reconfigure your infra
- Help on configuration drift (consistency)
- Control cost on dynamic environments
- Full integration with source control
- Versioned together with source code (and pipelines)
- Serves as infrastructure live documentation using declarative configuration
- Easy and recommended integration with CI/CD process, adding additional layer of security
- Allow you to test your infra definition

IaC: Declarative configuration

- Declarative configuration allow to define desired state on a more human-readable style
- You define what you want to achieve at the end
- How to implement your configuration is not your concern. Let the tooling do that for you
- Opposite of imperative configuration like scripting where you need to define all the steps
- Your configuration is idempotent, means you may ask to get your desired state as much you need and at the end you get always the same outcome
- With imperative configuration you may get the same but you need to do it by yourself

IaC: Tooling

	Pros	Cons
Proprietary	Always updated with last features Direct support from provider	Limited to one Provider You may need to learn several tools
Provider-agnostic	Better on hybrid environments Bigger Communities	Feature parity Changing Provider is not only a configuration task

Terraform

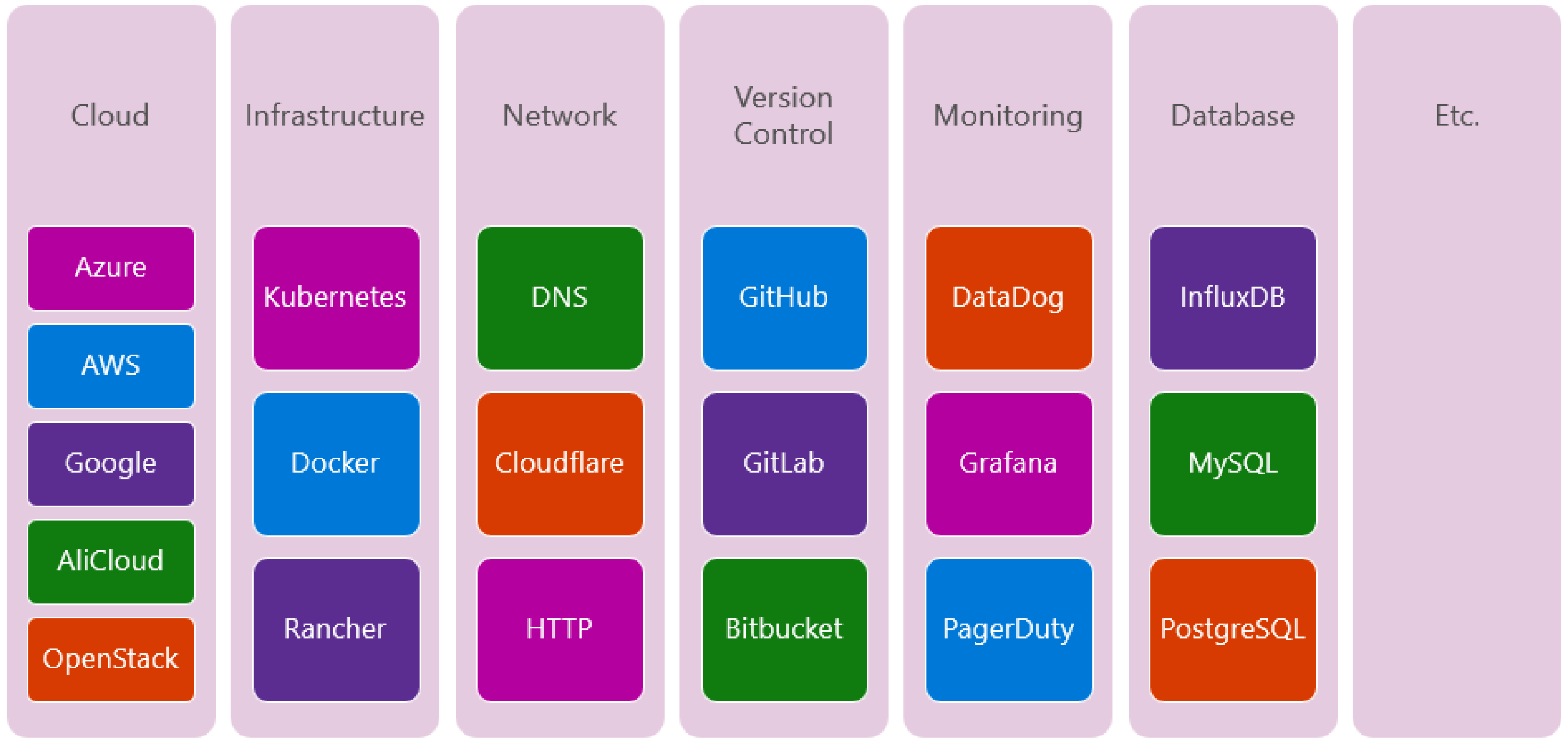
Secure DevOps

What is Terraform

- Multi platform and multi provider IaC tooling from Hashicorp
- Biggest community with a big ecosystem of providers
- Provides a clean and easy way to write and maintain your code
- Uses a proprietary language (HCL) but similar with JSON/YAML

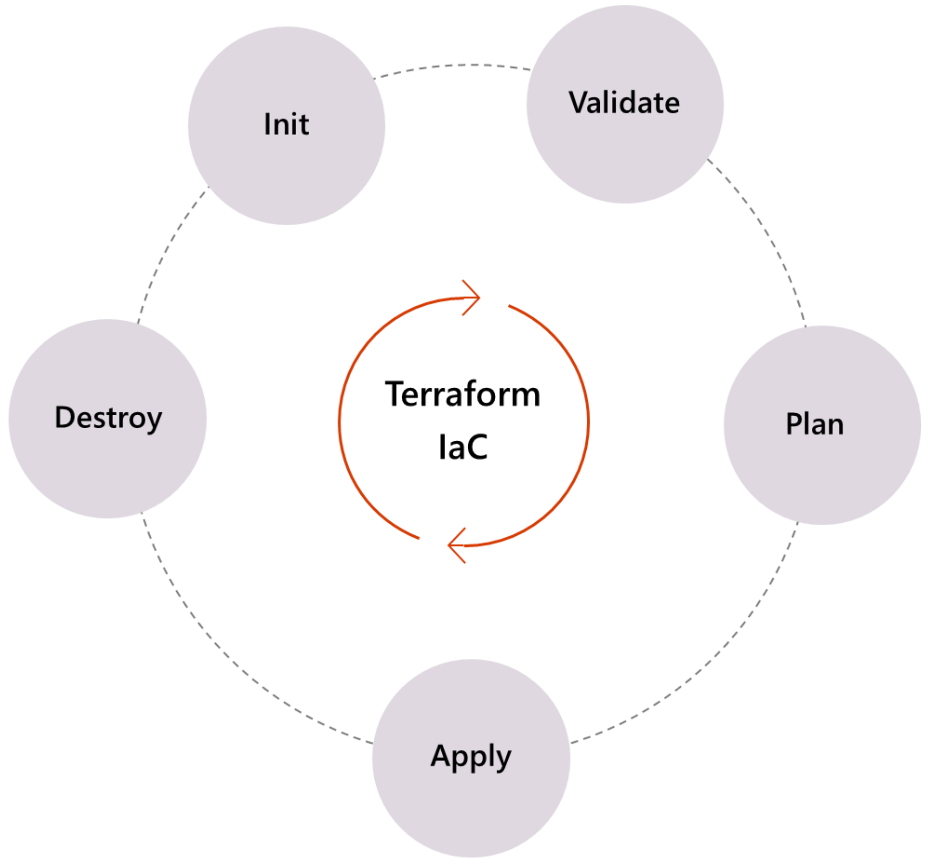
To create resources? Terraform Providers

- Big ecosystem of providers ([Browse Providers | Terraform Registry]())
- Allow to everyone defines your own provider if it not exists



Terraform basic workflow

- Init: Initialize a working directory with Terraform configuration files
- Validate: Validates configuration files in a directory without checking remotely
- Plan: It creates an execution plan (aka WhatIf)
- Apply: Deploy the changes required to reach the desired state
- Destroy: Remove the TF manage infrastructure



Security Analysis

Secure DevOps

Shift-left Infra Scanning

- Integrating security checks early in the delivery pipeline allows minimizing the cost of fixing security issues and ensure they do not reach production
- In the context of cloud environments, companies usually describe their infrastructure as code using tools like Terraform or CloudFormation
- Let’s review some tools that allow us to perform static analysis of Terraform code in order to identify cloud security issues and misconfigurations even before they pose an actual security risk

Shift-left Infra Scanning

- In the cloud, misconfigurations will get you hacked well before zero-days do
- According to a report released in 2020, the NSA asserts that misconfiguration of cloud resources is the most prevalent vulnerability in cloud environments
- Looking at a few recent data breaches in AWS
  - The Capital One breach was caused by a vulnerable application exposed to the Internet, along with an overprivileged EC2 instance role
  - The Los Angeles Times website started mining cryptocurrency in your browser due to a world-writable S3 bucket
  - The Magecart group backdoored Twilio's SDK which was hosted on a world-writable S3 bucket

Types of scan

- Static analysis tools for Terraform usually fall into one of two categories. They either scan HCL code directly, or scan the Terraform plan file.
- Scanning the HCL code has the advantage of making the scan faster, stateless, and not requiring any communication with a backend API
- Scanning the Terraform plan makes sure the scan runs after any interpolation, function call, or variable processing in the HCL code
- On the other hand, it requires that we generate the plan before scanning, often assuming that an authenticated communication with the appropriate backend is available
- Typically, tools scanning the HCL code take no more than a few seconds to run and can be used without network connectivity. However, they have a good chance of missing security issues introduced by dynamically evaluated expressions

Key Benefits

- Early Detection: Identifies security vulnerabilities and misconfigurations early in development, preventing them from reaching production.
- Compliance Assurance: Ensures Terraform code complies with industry standards and internal security policies.
- Automated Security Integration: Seamlessly integrates with CI/CD pipelines, automating security checks to maintain a continuous focus on security.
- Actionable Insights: Delivers detailed vulnerability reports, facilitating swift and effective resolution.
- Scalability: Effectively handles increasing project complexity and size, maintaining rigorous security standards without additional manual effort.

Main Features

- Policy Coverage: The tool should offer comprehensive scanning capabilities to detect security vulnerabilities specific to Infrastructure as Code.
- Customizable Security Policies: It must allow users to define and adjust security policies and severity levels to align with specific project needs or compliance requirements.
- Seamless Integration: The analyzer should integrate effortlessly with existing CI/CD tools and version control systems, facilitating a smooth workflow.
- Detailed Reporting: Clear and actionable reports are crucial. The tool should prioritize issues based on severity and provide practical steps for remediation.
- Scanning Customization: Users should be able to tailor the scanning process to focus on particular aspects of the codebase, enabling targeted and efficient security assessments.

IaC Scanning Tools

terrascan™
by tenable

checkov
by bridgecrew

trivy

IaC: Tooling

	# Policies	Terraform Providers	Custom Policies
Checkov	2110	aws, azure, gcp, digitalocean, kubernetes, github, gitlab, ibm, linode, openstack, alicloud	yaml
Trivy	322	aws, azure, gcp, digitalocean, cloudstack, github, oracle, openstack	OPA Rego
Terrascan	790	aws, azure, gcp, digitalocean, kubernetes, docker, github	OPA Rego

IaC Scanning Tools

	Docker Image	IDE Plugin	CI/CD System	Pre-Commit hook
Checkov	2110	VSCode, JetBrains	GitHub Actions, GitLab	Yes
Trivy	322	VSCode, JetBrains	Azure DevOps, GitHub Actions, Buildkite, Dagger, Semaphore, CircleCI, Concourse CI	No
Terrascan	790	VSCode	GitHub Actions, Atlantis	Yes

Demo: Checkov

Secure DevOps

Lab 06: IaC Scanning

Secure DevOps