Terraform in CI/CD

Terraform

Agenda

- Code Validation and Scanning
- Disposable Environments
- Configuration Drift

Code Validation and Scanning

Terraform

Code Validation and Scanning

- Now that your infra is code, you can use some additional tooling to validate your code
- To validate code syntax and provider you can use terraform validate command
- To scan your code and get some security validation you may rely on some external tooling

Code Validation and Scanning

- Now that your infra is code, you can use some additional tooling to validate your code
- To validate code syntax and provider you can use terraform validate command
- To scan your code and get some security validation you may rely on some external tooling

IaC Scanning Tools

terrascan™
by tenable

snyk

checkov
by bridgecrew

trivy

Checkov

- Support several IaC tooling like Terraform, ARM Templates, Bicep, CloudFormation, Helm Charts, etc.
- Checks against security best practices, like security best practices and compliance standards (CIS, NIIST, Well-Architected Frameworks)
- Checks against potential misconfigurations, such as overly permissive security group rules, weak encryption settings, public exposure of sensitive information
- Easy integration with CI/CD pipelines
- Allow development of custom checks

Checkov in CI/CD

- Following Git best practices, you should have Pull Requests enabled on your repo
- Ideally, you should run this validation on your PR pipeline
- Checkov outcome can be a great automatic checker if your PR can progress or not
- Be careful on the approach since you may make your infra too restrictive and increase your costs

Demo – Checkov

Terraform

Disposable Environments

Terraform

Provisioning Environments

- When you develop IaC your main goal is to provision your environment(s)
- Following a traditional approach, you have several (at least 2) environments until you reach production
- Each environment have a specific purpose, and it’s use during the timeline that you define to validate that purpose
- Development environment is used for developers test new code
- QA environment to run automated and manual testing
- Staging to have an environment close to production to make final tests

Environments Purpose

- Development environment is used for developers test new code
- QA environment to run automated and manual testing
- Staging to have an environment close to production to make final tests

Development Lifecycle

- Depending the way you define your development lifecycle, these environment may be running 24x7 and being used for much smaller period
- This can cause you impact on costs but even on security because you have a bigger attack surface
- On cloud platforms, depending on your infra, you may turn off all or parts of the environments
- In a more modern approach you may use the concept of disposable environments

Disposable Environments Lifecycle

- This concept means you create the environment only when needed and delete after you finish the process you want to run
- For instance, you create an environment close to production to run automated load testing and delete after the process finished
- With this approach you optimize your costs, shrink your attack surface and can have bigger environments and more similar with production
- On another perspective, you can have several development environments, allowing everyone on the team to test their code without impacting the others

Disaster Recovery

- Disaster Recovery environments are in place to be used on last resort
- Depending on the solution, DR environments can have a pre-defined SLA to be running
- On the cases that you have alignment with the infra provisioning time and application deployment time, you can use IaC as an effective and cost optimized way

Configuration Drift

Terraform

Infrastructure Provisioning

- As already stated, you develop your IaC to provision your infrastructure
- But you only do that the first time you run your code and any time you change that code
- All the other time you run your IaC is to validate that you have the expected infrastructure to deploy your solution
- This is the way IaC solve the configuration drift issue

Configuration Drift

- Using that approach you can always have sure that you find the infra you need to run properly your applications
- This implementation is out-of-the-box available in any IaC tool due to idempotence property
- Add this validation to a CI/CD pipeline, you can have a quality gate that fail or force the sync between your code and your infra

Lab 07 – Terraform and CI/CD

Terraform