Building GCP with Terraform

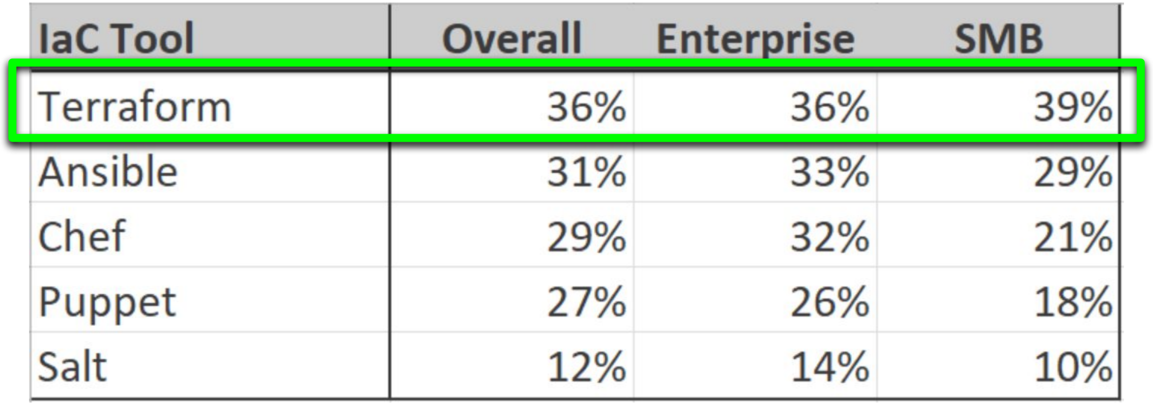
Greg Horie

Overview

- What is IaC?
- Declarative vs. Procedural
- What is Terraform?
- Installation
- Basics Workflow
- HCL
- Providers (Plugins)
- State
- Resources, Inputs
- Terraform Cloud ?

What is IaC?

- IaC = Infrastructure-as-Code
- High-level coding languages used to automate IT infrastructure builds.
- **IaC Tool Adoption**



IaC Tool	Overall	Enterprise	SMB
Terraform	36%	36%	39%
Ansible	31%	33%	29%
Chef	29%	32%	21%
Puppet	27%	26%	18%
Salt	12%	14%	10%

* Taken from Flexera 2021 State of Cloud report.

- Not an apples-for-apples comparison.
- In practice, these tools are often used together to fully-automate builds.

Procedural vs. Declarative

- IaC tools can be categorized as Procedural or Declarative.
    - Based on how the tool applies changes to the underlying target systems.
- **Procedural**
    - Think script.
    - Step-by-step tasks to automate your build.
    - Ansible & Chef are procedural - Code step-by-step tasks to achieve a desired end state.
- **Declarative**
    - Think HTML.
    - Code says what is desired, not the steps.
    - Hides the process and reveals relationships.
    - Terraform & Puppet are declarative - Code specifies your desired end state.

What is Terraform?

- Terraform is an open source IaC tool.
- Enables you to build, change, and version cloud and on-prem resources.
- Original author - Mitchell Hashimoto.
    - Maintained by Hashicorp.
- Enabled through the use of "Providers".
    - Providers = plugins which are an abstraction over an API.
    - Many providers are available through the public Terraform Registry.
    - Support for all major cloud providers.
- Modules are also available to further abstract provider interactions.
    - Public modules are available and you can also build your own.

Terraform Installation

- https://learn.hashicorp.com/tutorials/terraform/install-cli

Basic Workflow

1. Code
   - Write your infrastructure configuration.
2. Initialize
   - Initialized providers and modules.
   - **`terraform init`**
3. Plan
   - Displays the difference between desired and current state.
   - Creates a plan to reconcile this gap.
   - **`terraform plan`**
4. Apply
   - Apply the changes to achieve the desired state.
   - **`terraform apply`**

Quick Demo

- Using Terraform's local provider:

  https://registry.terraform.io/providers/hashicorp/local

- For command line notes, see:

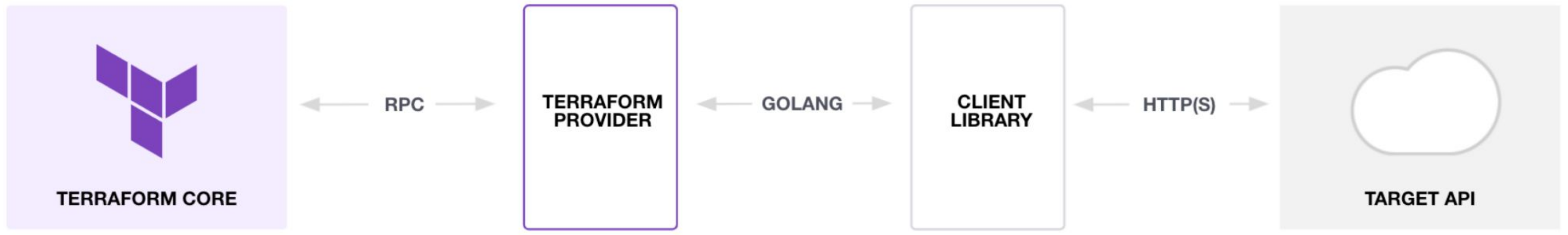
  https://github.com/netserf/netsig-presentation-building-gcp-with-terraform/terraform-cheat-sheet.txt

HCL

- HCL = Hashicorp Configuration Language
- Toolkit for creating structured configuration languages
- Goal is to be both human- and machine-friendly

Providers

- Provider = Terraform plugin that typically integrates with an external API.
- Acts as a translation layer that allows Terraform to communicate and build external resources - VMs, databases, services, etc.



State

- State
- Resources, Inputs
-

Resources

Variables

Modules

GCP Build Network

- For command line notes, see:

  https://github.com/netserf/netsig-presentation-building-gcp-with-terraform/terraform-cheat-sheet.txt

GCP Build VMs

- For command line notes, see:

  https://github.com/netserf/netsig-presentation-building-gcp-with-terraform/terraform-cheat-sheet.txt

Summary

Possible Future Discussions

- Go deeper with GCP networking
  - IPv6, DNS, Load Balancing, VPNs
- Building AWS with Terraform
- Kubernetes
- Current Monitoring Tools
  - e.g. Prometheus, Elastic Stack, etc.
- CI / CD
  - GitHub Actions
- Other ideas welcome!

VicPiMakers and Others Slack

- Please let us know if you want an invite to this Slack group

Backup Slides