Terraforming GCP

Taking the first steps

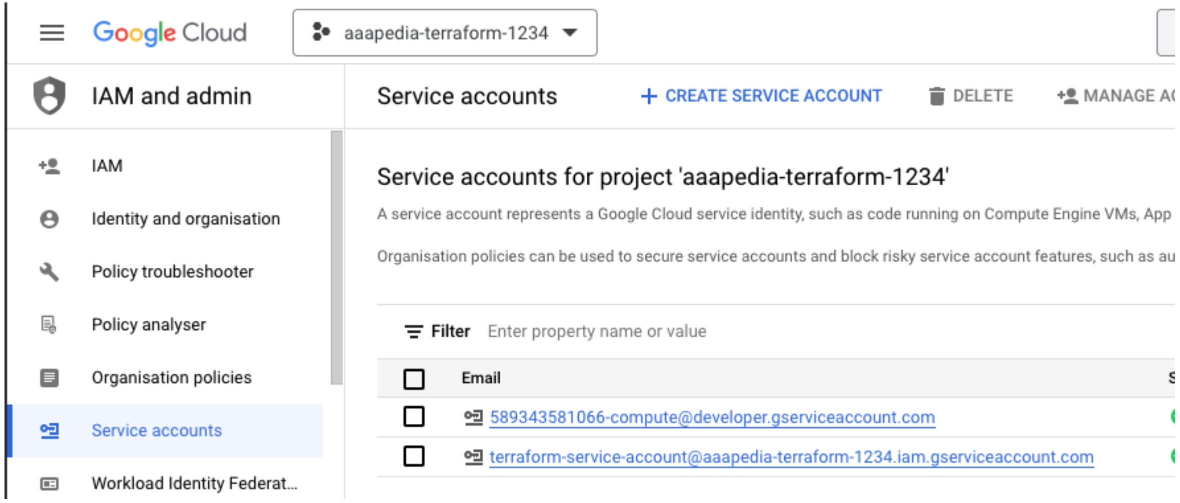
What we talk about today

- What is Terraform? Basic commands and definitions.
- Exercise: Installing and basic commands
- Terraform details
- Exercise: Web Server, accessible from the internet
- Variables
- Exercise: Flexible web server
- Structuring terraform projects
- Closing Remarks
- 3ap projects & further resources

What is needed to terraform GCP?

1. Google Account / Membership in Google Organisation
2. Service Account (organisation wide or per account)

- Sufficient rights
- Authentication (json-key ⚠️ / access for personal account)



Service accounts for project 'aaapedia-terraform-1234'

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App

Organisation policies can be used to secure service accounts and block risky service account features, such as au

	Email
☐	589343581066-compute@developer.gserviceaccount.com
☐	terraform-service-account@aaapedia-terraform-1234.iam.gserviceaccount.com

What is Terraform

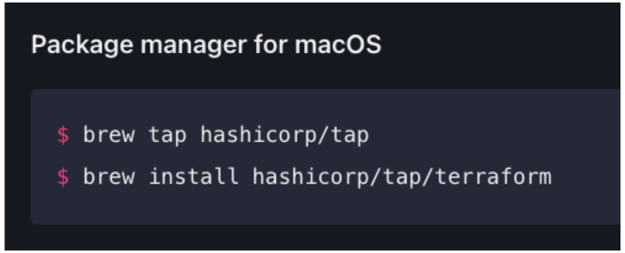
- https://www.terraform.io
- CLI tool with files & declarative language → Infrastructure as Code
  - Supports variables, conditions, loops, references, dependencies, …
  - Human & machine readable files (*.tf, *.tfvars, *.tfplan, …)
- Keeps track of changes in local or remote state
- Multitude of plugins to interact with various environments
- Infrastructure management without scripts, clicks or other hacks

Description of a desired infrastructure state. Allows for quick setup & teardown of changes and new environments. Usually. 🤫

Installing terraform

https://developer.hashicorp.com/terraform/downloads

MacOS



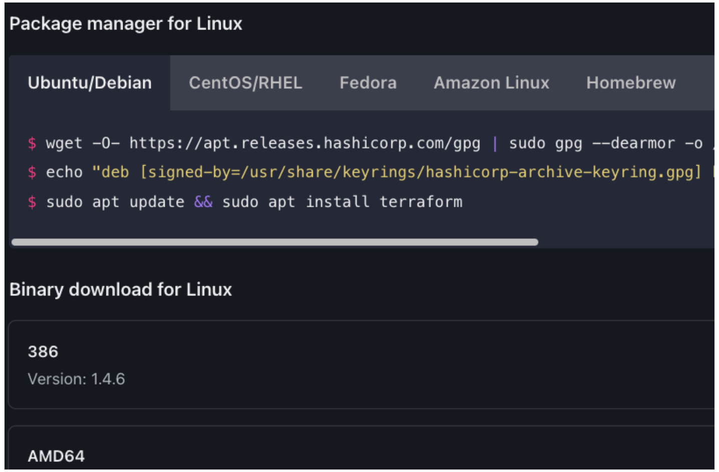
Package manager for macOS

```
$ brew tap hashicorp/tap
$ brew install hashicorp/tap/terraform
```

Windows

Please download from the site.

Linux



Package manager for Linux

```
$ wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o
$ echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]
$ sudo apt update && sudo apt install terraform
```

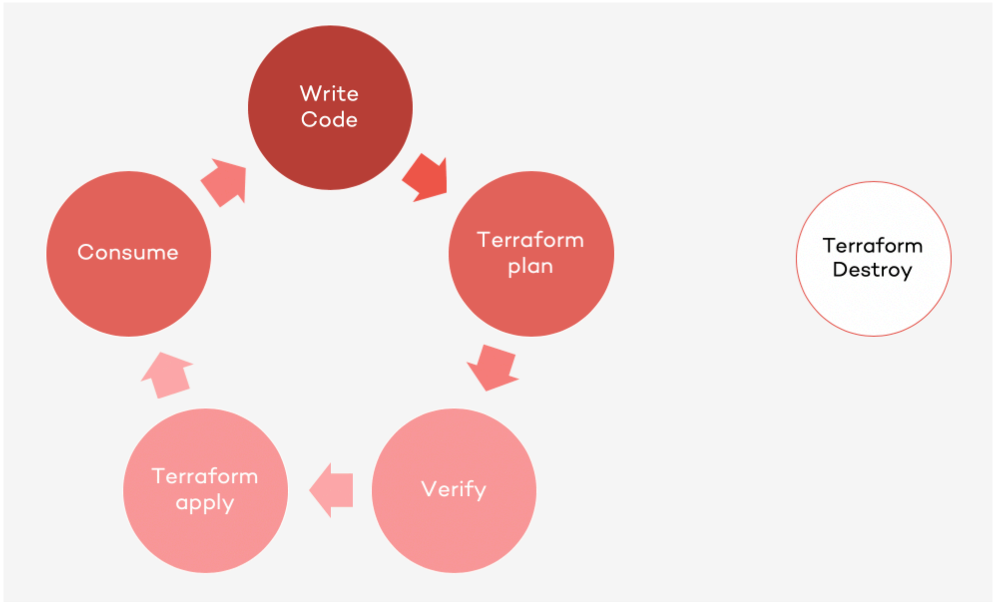
Binary download for Linux

386
Version: 1.4.6

AMD64

Verify with terraform -v

The Terraform cycle



https://www.thorsten-hans.com/terraform-the-definitive-guide-for-azure-enthusiasts/

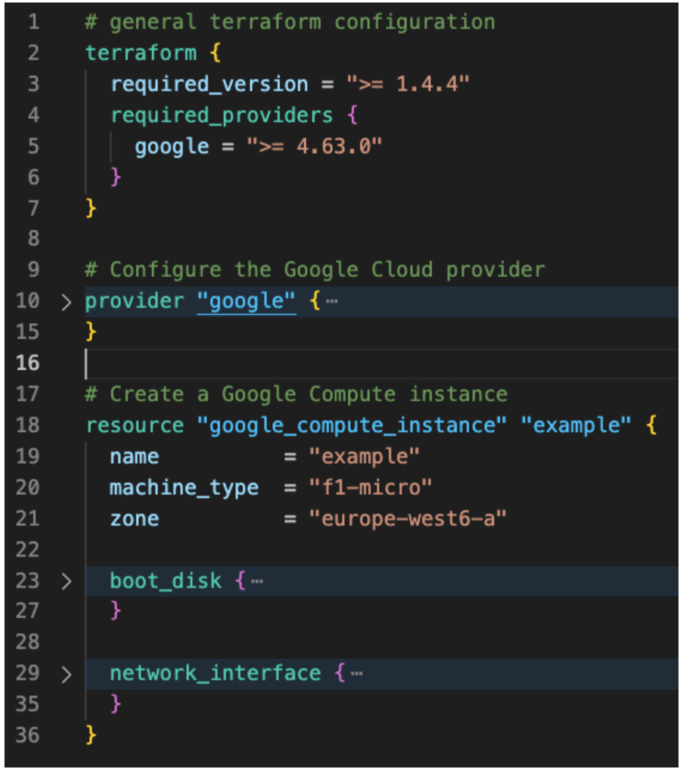
Common terraform … commands:

… **init** - Initialize a directory & state
… **plan** - Check for changes to be made
… **apply** - Execute (planned) changes

… **workspace new|select** - Current context
… **import** - Import existing infrastructure
… **show** - Show current state
… **fmt** - Format terraform files
… **validate** - Validate current definition
… **destroy** - Delete resources

Terraform files and components

- At least a *.tf file, usually main.tf
- Contains definitions for
    - **TF Metadata:** Version compatibility & provider requirements (optional, but highly recommended)
    - **Providers:** “Plugins” to use (GCP, Azure, K8s, …)
    - **Resources:** Elements to create
    - **Data:** Information to read from infrastructure
    - **Locals:** (Dynamic) self defined values
    - **Variables:** Dynamic arguments
    - **Outputs:** Returned values of operation
    - **Modules:** Groups of definitions
    - …
- Terraform combines all *.tf files



```
# general terraform configuration
terraform {
  required_version = ">= 1.4.4"
  required_providers {
    google = ">= 4.63.0"
  }
}

# Configure the Google Cloud provider
provider "google" {…
}

# Create a Google Compute instance
resource "google_compute_instance" "example" {
  name          = "example"
  machine_type  = "f1-micro"
  zone          = "europe-west6-a"

  boot_disk {…
  }

  network_interface {…
  }
}
```

Your first GCP VM

Let’s get your hands dirty!

1. Check out repository: https://github.com/3AP-AG/3apedia-terraforming-gcp/
2. Open 1-gcp-vm/main.tf
3. Change name of compute instance to your (unique) name

And only afterwards!:

1. Open terminal in the folder
2. terraform init
3. terraform plan -out=out.tfplan
4. terraform apply “out.tfplan”

terraform destroy

A bit more sophisticated

1. Open 2-web-server/main.tf
2. Change name of compute instance **and firewall** to your name
3. init → plan → apply
4. Access the printed IP in Browser

```
# Output variable: Public IP address
output "public_ip" {
 value = "${google_compute_instance.example.network_interface.0.access_config.0.nat_ip}"
}
```

1. **destroy**

Needs only one firewall, workshop purpose 🤫

Terraform Variables

- How to provide dynamic or confidential information? → Variables!
- Values kept in *.tfvars files (keep out of git ⚠️)
- Has description, defaults, validations, …
- Referenced with var.<name>
- Available on root & module level
- Possible arguments for terraform plan and apply
- Terraform will ask if no value provided or no defaults

```
# an example variable
variable "key" {
   description = "description of the variable"
   sensitive = false # whether to hide value while printing
   default = "" # default value
}
```

Making it flexible

1. Open 3-variable-web-server/main.tf
2. Adapt variable definition to allow for your name
3. Use for compute instance, firewall (and “Hello” statement)
4. Provide my.tfvars file with key=value
5. init → plan -var-file=my.tfvars -out=out.tfplan → apply “out.tfplan”
6. (Access in Browser)
7. (Try without passing tfvars file)

1. **destroy**

Structuring terraform - the minimum

Terraform enforces no structure. Split, as you wish.

Combines everything in current folder, stores state per component

Pass values between components by reference, terraform sorts it

Best practices

- Files
  - **minimum:** main.tf, variables.tf, output.tf
  - **often seen:** providers.tf
  - **as you wish:** context specific, like nginx.tf
- Keep related things together

Structuring terraform - even better

Modules

Virtual “packages” with own scope, main.tf, variables.tf and output.tf

Referenced in root main.tf

```
module "compute_instance" {
 source = "./modules/compute-instance"
 name = var.name # passing down values
}

# Output with module reference
output "public_ip" {
 value = module.compute_instance.public_ip
}
```

Tips & tricks

- Use remote state. Enables collaboration and automation.
- State control via terraform state list|rm
- Use terraform workspaces
    - Separate state per environment
    - Environment TF_WORKSPACE
    - terraform workspace select|new
- terraform import for existing infrastructure

```
terraform {
 backend "gcs" {
   bucket = "google-bucket-1234"
   prefix = "terraform/state/<project>"
 }
}
```

```
➜  terraform workspace list
  default
* dev
  prod
```

- Ask around! More and more work with it.

And there's more to it… 😬

Things to regard on GCP

Soft deletes and unique names

→ Random id suffixes, as things get soft deleted

Sometimes easier / only possible to manually delete

→ TF state cleanup → TF apply

What does the UI / CLI allow?

→ Orientation on needed TF resources & configuration

…

Closing Remarks

- Very powerful. Allows for consistency
  - CI/CD integration
  - Same config on new environment will result in the same
  - Easy to destroy and reapply, if losing stuff is ok
- Yet another tool and language to learn
- Hard to use without knowing, what you want to achieve
- Check, before applying. Reverts are not always easy
- Clean up of state is possible, but annoying

Advice:

- Start small and work in increments (applied state helps to keep overview)
- Structure, but don't overcomplicate
- What belongs into terraform, what belongs elsewhere?
- Think ahead. What’s needed in only certain environments?

3ap projects using terraform

3ap Platform (GCP): [3ap-platform/tree/develop/cloud/terraform]

Conperi (GCP): [conperi-platform/tree/develop/cloud-infrastructure/tf]

Enge (GCP): [enge/tree/develop/cloud/terraform]

OYU (GCP): [oyu-backend/tree/develop/cloud/terraform]

ÖKK (Azure): [oekk-simpla-backend/tree/develop/cloud/terraform]

Foundera (GCP): [foundera/tree/main/cloud/terraform]

Swisscard SCNET (GCP): [swisscard-scnet/tree/develop/cloud/terraform]

…

Further Resources

Official terraform tutorials: https://developer.hashicorp.com/terraform/tutorials

Provider registry: https://registry.terraform.io

GCP tutorials & documentation: https://cloud.google.com/docs/terraform

Brain Snack by Simona B.: brain-snacks/tree/develop/terraform & recording

… the internet 🙏 …

DONE
!
?