Terraform best practices with examples and arguments

by Anton Babenko

September 2018

Agenda

- Introduction
- Why Terraform?
- ~~Terraform basics~~ (www.terraform.io/docs)
- Terraform Modules
- Q&A
- Bar/coffee talk

Anton Babenko

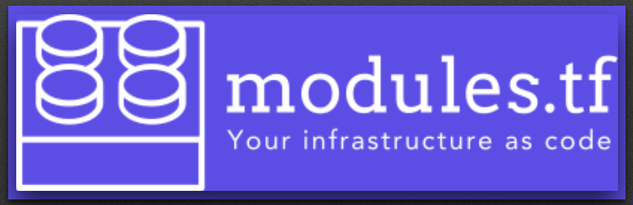
Terraform AWS fanatic

Organise {HashiCorp, AWS, DevOps} User Groups in Norway

DevOpsDays Oslo (29-30th October 2018)

I 💚 open-source — github.com/antonbabenko

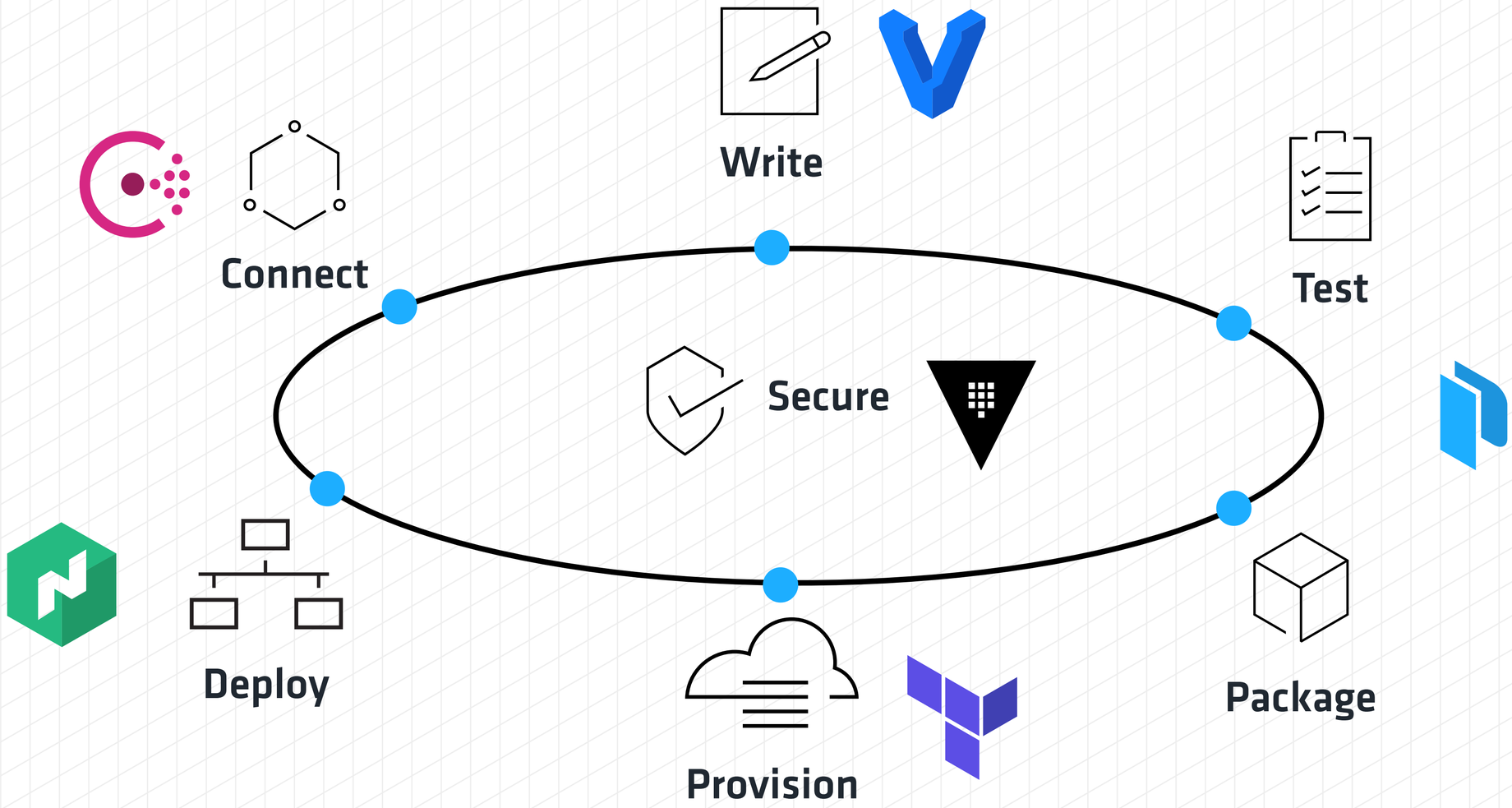
- terraform-community-modules + terraform-aws-modules
- antonbabenko/terrapin — Terraform modules generator
- antonbabenko/pre-commit-terraform — make your configurations nicer
- antonbabenko/modules.tf-lambda — from visual diagram to Terraform
- antonbabenko/terraform-best-practices



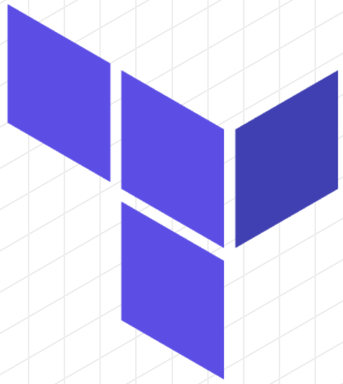
twitter.com/antonbabenko linkedin.com/in/antonbabenko medium.com/@anton.babenko


HashiConf '18
October 22-24, 2018 | San Francisco
www.hashiconf.com

Application Delivery with HashiCorp



https://www.hashicorp.com/resources/application-delivery-hashicorp



HashiCorp
Terraform

Write, plan, and create infrastructure as code

www.terraform.io

```hcl
variable "aws_region" {
  description = "Region where resources should be created"
  default     = "eu-west-1"
}

provider "aws" {
  region = "${var.aws_region}"
}

resource "aws_s3_bucket" "this" {
  bucket = "my-bucket-${random_pet.bucket.id}"
}

resource "random_pet" "bucket" {
  keepers = {
    aws_region = "${var.aws_region}"
  }

  length = 1
}

output "this_s3_bucket_id" {
  description = "ID of S3 bucket"
  value       = "${aws_s3_bucket.this.id}"
}
```

```
$ terraform init

Initializing provider plugins...
- Checking for available provider plugins on https://releases.hashicorp.com...
- Downloading plugin for provider "aws" (1.10.0)...
- Downloading plugin for provider "random" (1.1.0)...

Terraform has been successfully initialized!
```

```
$ terraform apply

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  + aws_s3_bucket.this
      id:                   <computed>
      acl:                  "private"
      bucket:               "my-bucket-${random_pet.bucket.id}"

  + random_pet.bucket
      id:                   <computed>
      keepers.%:            "1"
      keepers.aws_region:   "eu-west-1"
      length:               "1"

Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

random_pet.bucket: Creating...
  keepers.%:          "" => "1"
  keepers.aws_region: "" => "eu-west-1"
  length:             "" => "1"
random_pet.bucket: Creation complete after 0s (ID: seasnail)
aws_s3_bucket.this: Creating...
  acl:                "" => "private"
  arn:                "" => "<computed>"
  bucket:             "" => "my-bucket-seasnail"
aws_s3_bucket.this: Creation complete after 6s (ID: my-bucket-seasnail)

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

this_s3_bucket_id = my-bucket-seasnail
```

Why Terraform?

Plus 100+ more providers

HashiCorp
Terraform

Write, plan, and create infrastructure as code

www.terraform.io

Why Terraform and not AWS CloudFormation, Azure ARM, Google Cloud Deployment Manager?

- Terraform manages 100+ providers, has easier syntax (HCL), has native support for modules and remote states, has teamwork related features, is an open-source project.
- Provides a high-level abstraction of infrastructure (IaC)
- Allows for composition and combination
- Supports parallel management of resources (graph, fast)
- Separates planning from execution (dry-run)

Terraform's Goals

- Unify the view of resources using infrastructure as code
- Support the modern data center (IaaS, PaaS, SaaS)
- Expose a way for individuals and teams to safely and predictably change infrastructure
- Provide a workflow that is technology agnostic
- Manage anything with an API

Terraform — is a universal tool to manage anything that has an API

- GSuite resources
- Dropbox user files
- New Relic alerts
- Datadog users, monitors
- Jira issues
- See All Terraform providers

What are the tools/solutions out there?

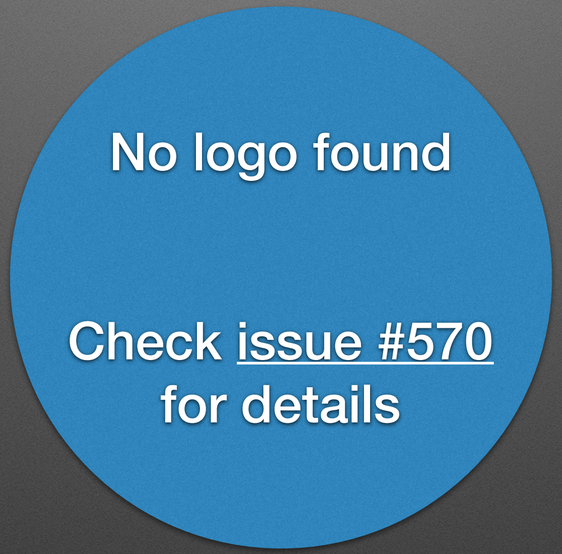
- Terraform Registry (https://registry.terraform.io/) — collection of public Terraform modules for common infrastructure configurations for any provider.
- Terraform linter to detect errors that can not be detected by `terraform plan` — https://github.com/wata727/tflint
- Terraform version manager — https://github.com/kamatama41/tfenv
- A web dashboard to inspect Terraform States — https://github.com/camptocamp/terraboard
- Jsonnet — The data templating language — http://jsonnet.org

Atlantis — Start working on Terraform as a team

A unified workflow for collaborating on Terraform through GitHub, GitLab and Bitbucket

https://www.runatlantis.io

Terragrunt



Thin wrapper for Terraform that provides extra tools for working with multiple Terraform modules

https://github.com/gruntwork-io/terragrunt/

Terraform Learning Path

- ✓ Terraform basics
- Modules
- Code structures
- Integrations (CI/CD), lifecycles
- Tricks&tips

Terraform Modules

Types of Terraform modules

Resource modules (terraform-aws-modules, for eg):

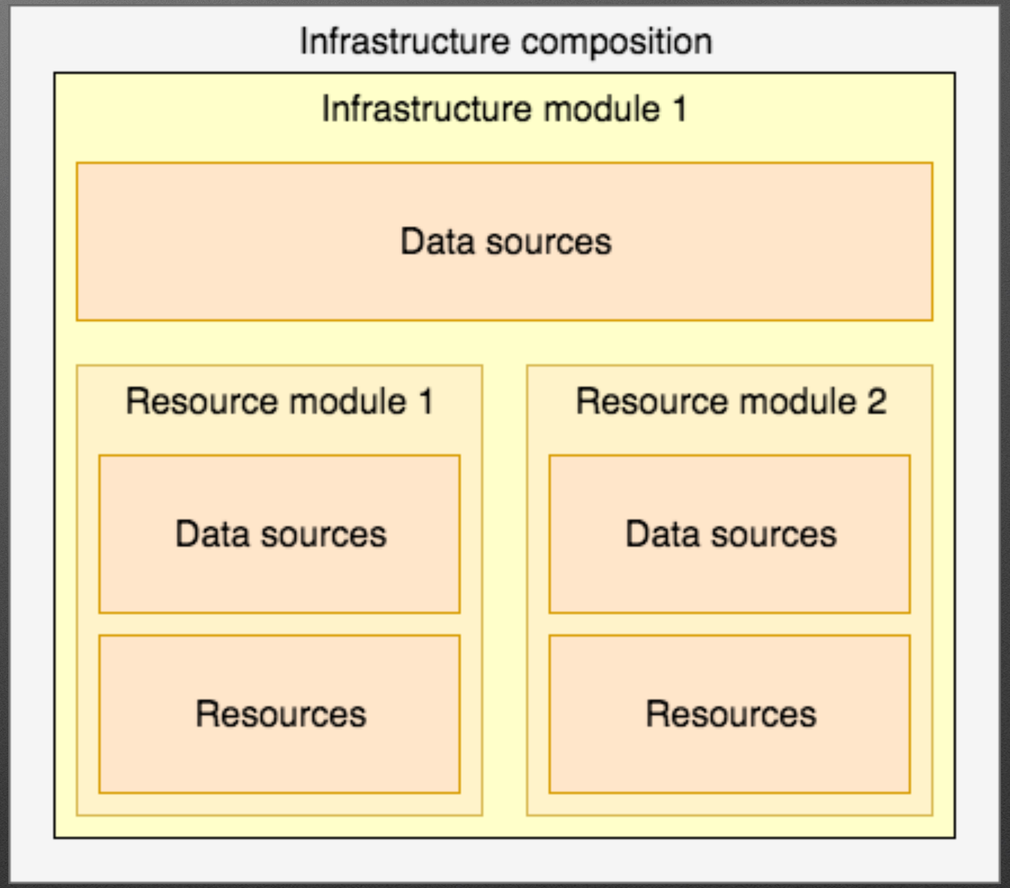
- Create resources (obviously)
- Few relations to other modules (usually)
- Very flexible

Infrastructure modules:

- Use specific version of resource modules
- Company-wide standards (eg, tags and names)
- May use code generators (jsonnet, cookiecutter, etc)

Compositions:

- Use specific version of infrastructure or resource modules
- Provide all the values for region, environment, module, etc
- Terragrunt is awesome



https://www.terraform-best-practices.com/key-concepts

Traits of good Terraform modules

- Clean code
- Feature-rich
- Sane defaults
- Tests
- Examples
- Documentation
- … (secure, versioning, lifecycle-readiness)

Read more: https://medium.com/@anton.babenko/using-terraform-continuously-common-traits-in-modules-8036b71764db

Collection of Terraform AWS modules supported by the community (100+ contributors).

More than 1,5 million downloads since September 2017.

(VPC, Autoscaling, RDS, Security Groups, ELB, ALB, Redshift, SNS, SQS, IAM, EKS, ECS...)

github.com/terraform-aws-modules

registry.terraform.io/modules/terraform-aws-modules

Q: What should we build?

git clone git@github.com:antonbabenko/terraform-best-practices-workshop.git

Read this — https://github.com/antonbabenko/terraform-best-practices-workshop/blob/master/README.md

"Lets host static web-site on AWS S3"

- https://www.terraform.io/docs/providers/aws/r/s3_bucket.html
- https://www.terraform.io/docs/providers/aws/r/s3_bucket_object.html
- 2 separate modules:
  - git clone git@github.com:terraform-aws-modules/terraform-aws-s3-bucket.git
  - git clone git@github.com:terraform-aws-modules/terraform-aws-s3-object.git
- Remember "Traits of good Terraform modules" (Clean code, Feature-rich, Sane defaults, Tests, Examples, Documentation) ?
- Naming and default value copy from the official Terraform docs
- All arguments (variables) & all attributes (outputs)