Introduction to Terraform/OpenTofu and their packaging in pkgsrc

Leonardo Taccari
<leot@NetBSD.org>

The NetBSD Foundation

EuroBSDCon 2025 - NetBSD Summit
September 26th 2025
Zagreb, Croatia

Outline

Infrastructure as Code (IaC), Terraform and OpenTofu

Infrastructure as Code (IaC)

The process of managing and provisioning an organization's IT infrastructure using machine-readable configuration files, rather than employing physical hardware configuration or interactive configuration tools. [1]

[1] From NIST SP 800-172.

Terraform

Terraform [2] *is an open-source infrastructure as code software tool that provides a consistent CLI workflow to manage hundreds of cloud services. Terraform codifies cloud APIs into declarative configuration files.* [3]

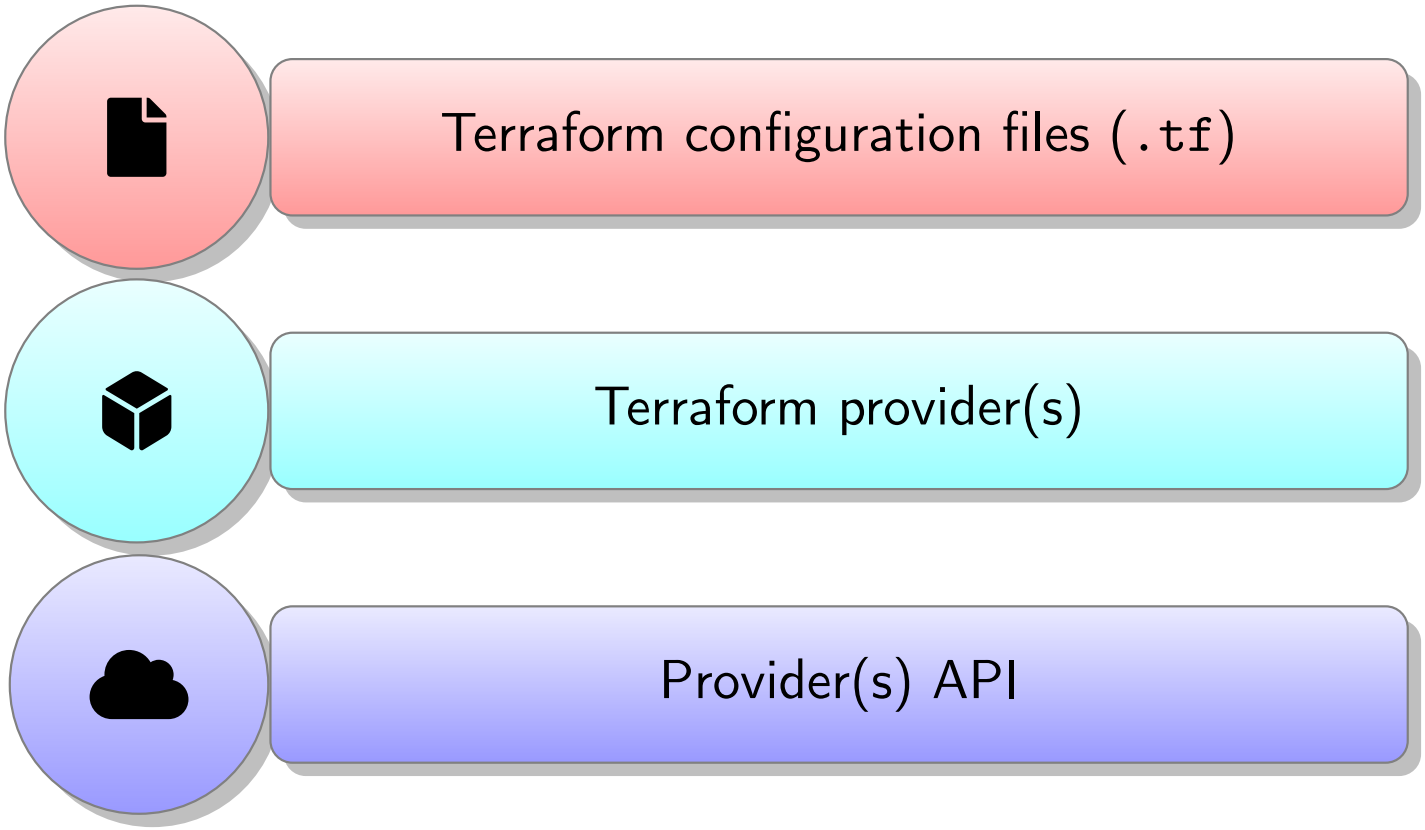
[2] From now on, at least for introduction, everything that we say about Terraform is valid also for OpenTofu.

[3] From old Terraform homepage: https://www.terraform.io/.

Terraform

- Infrastructure as Code (IaC) tool
- Declarative Language (HCL: HashiCorp Configuration Language)
- Cloud-agnostic
- Thousands of providers and modules available
- Encourage infrastructure changes in two steps:
  - planning phase
  - deployment of resources

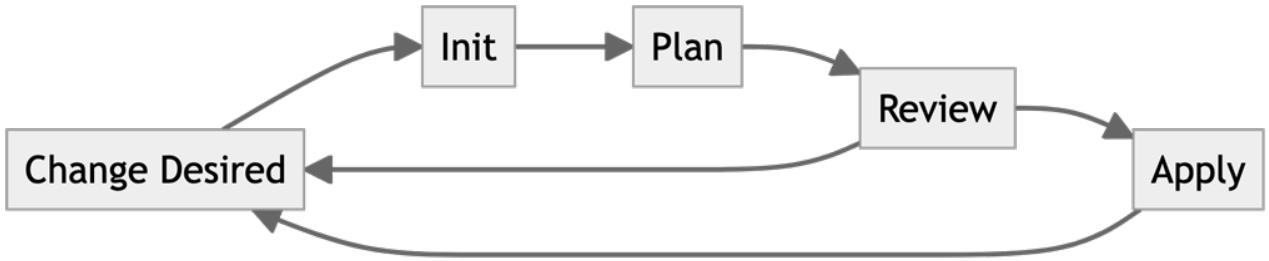
Terraform: how it works?



Terraform: glossary

provider	plugin that supplies a collection of resources and data sources for managing infrastructure with a particular vendor
resource	represents an actual piece of infrastructure
data source	similar to resource but can only be used to read data
module	collection of `.tf` files kept together in a directory
state	map real world resources to configuration, keeping track of metadata and to improve performance in large infrastructures

Terraform: typical workflow



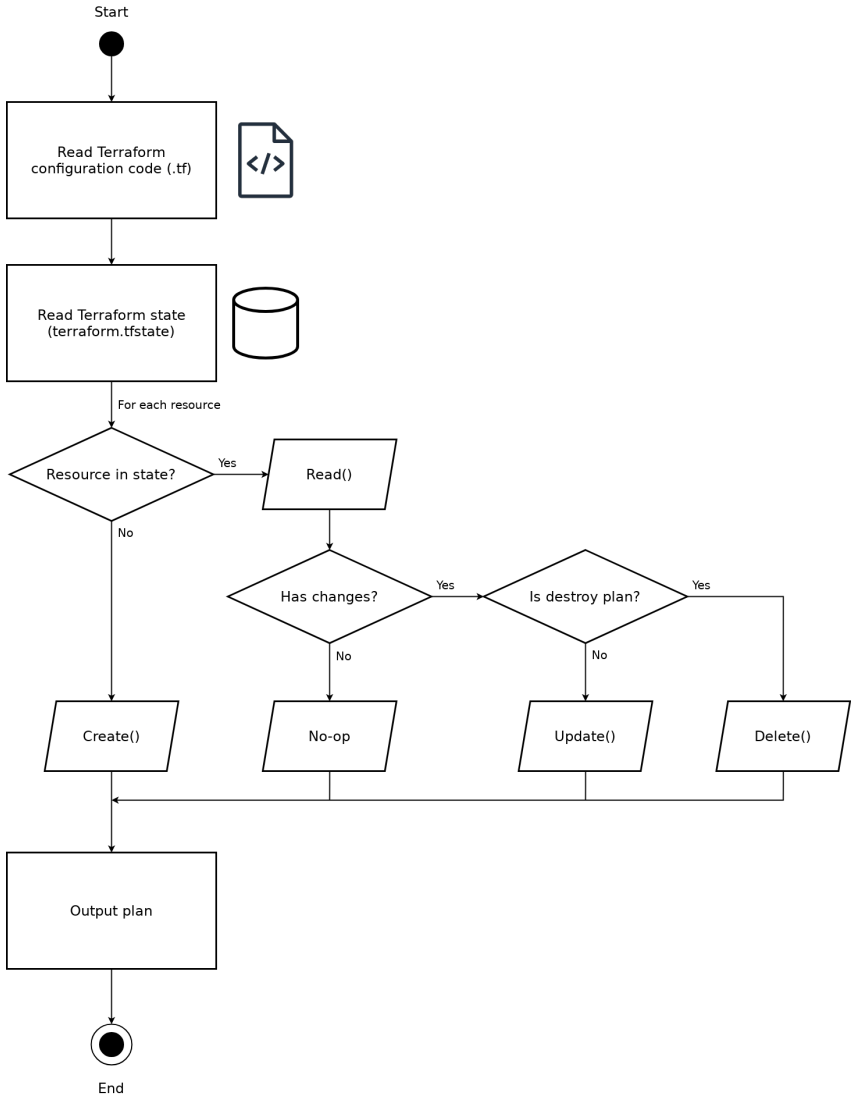
Terraform development workflow from Terraform in Depth by Robert Hafner.

init initializes a new or existing Terraform working directory by creating initial files, loading any remote state, downloading providers and modules

plan generates a speculative execution plan, showing all the actions (create, updates (in place), destroy) Terraform will do

apply creates or updates infrastructure according to Terraform configuration files

Terraform: flowchart of Terraform execution plan



Flowchart of Terraform execution plan from Terraform in Action by Scott Winkler.

Simple example: private S3 bucket

`terraform` block defines the required Terraform version and required providers:

```
terraform {
  required_version = "~> 1.6"

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# ...continue...
```

Simple example: private S3 bucket (cont.)

Each `provider` should be configured. In this case we would like to use the `eu-south-1` (Milan) AWS region for all the resources that we are managing:

```
# ...continued...

provider "aws" {
  region = "eu-south-1"
}

# ...continue...
```

Simple example: private S3 bucket (cont.)

Via several `resources` - all exposed by the `aws` provider - we desire a private S3 bucket:

```
# ...continued...

resource "aws_s3_bucket" "this" {
  bucket = "simple-example"

  tags = {
    Name        = "Simple Example Bucket"
    Environment = "development"
  }
}

resource "aws_s3_bucket_acl" "this" {
  bucket = aws_s3_bucket.this.id
  acl    = "private"
}

resource "aws_s3_bucket_public_access_block" "this" {
  bucket = aws_s3_bucket.this.id

  block_public_acls       = true
  block_public_policy     = true
  ignore_public_acls      = true
  restrict_public_buckets = true
}
```

Simple example: private S3 bucket (init)

- This Terraform configuration file is saved in its own directory `simple-example` as `main.tf`.
- If we move to that directory we can start the workflow described above.
- Let's start with `init`!

```
$ tofu init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.80.0...
- Installed hashicorp/aws v5.80.0 (unauthenticated)

[...]

OpenTofu has been successfully initialized!

You may now begin working with OpenTofu. Try running "tofu plan" to see
any changes that are required for your infrastructure. All OpenTofu commands
should now work.

If you ever set or change modules or backend configuration for OpenTofu,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Simple example: private S3 bucket (plan)

```
$ tofu plan

OpenTofu used the selected providers to generate the following
execution plan. Resource actions are indicated with the following
symbols:

  + create

OpenTofu will perform the following actions:

  # aws_s3_bucket.this will be created
  + resource "aws_s3_bucket" "this" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "simple-example"
      [...]
    }

  # aws_s3_bucket_acl.this will be created
  + resource "aws_s3_bucket_acl" "this" {
      + acl    = "private"
      + bucket = (known after apply)
      + id     = (known after apply)
    }

  [...continue...]
```

Simple example: private S3 bucket (plan)

```
  [...continued...]
  # aws_s3_bucket_public_access_block.this will be created
  + resource "aws_s3_bucket_public_access_block" "this" {
      + block_public_acls       = true
      + block_public_policy     = true
      + bucket                  = (known after apply)
      + id                      = (known after apply)
      + ignore_public_acls      = true
      + restrict_public_buckets = true
    }

Plan: 3 to add, 0 to change, 0 to destroy.

------------------------------------------------------------

Note: You didn't use the -out option to save this plan, so OpenTofu
can't guarantee to take exactly these actions if you run "tofu apply"
now.
```

Simple example: private S3 bucket (apply)

```
$ tofu apply

OpenTofu used the selected providers to generate the following
execution plan. Resource actions are indicated with the following
symbols:

  + create

OpenTofu will perform the following actions:

  [...same output of plan...]

Plan: 3 to add, 0 to change, 0 to destroy.

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

  Enter a value: yes

aws_s3_bucket.this: Creating...
aws_s3_bucket.this: Creation complete after 2s [id=simple-example]
aws_s3_bucket_public_access_block.this: Creating...
aws_s3_bucket_acl.this: Creating...
aws_s3_bucket_acl.this: Creation complete after 0s [id=simple-example,private]
aws_s3_bucket_public_access_block.this: Creation complete after 0s [id=simple-example]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
```

Module example: private S3 bucket

- In Terraform often we reuse reusable module like LEGO building blocks

```
terraform {
  required_version = "~> 1.6"

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

provider "aws" {
  region = "eu-central-1"
}

module "s3_bucket" {
  source  = "terraform-aws-modules/s3-bucket/aws"
  version = "~> 4.0"

  bucket = "module-example"
  acl    = "private"

  tags = {
    Name        = "Module Example Bucket"
    Environment = "development"
  }
}
```

Module example: private S3 bucket (init)

```
$ tofu init

Initializing the backend...
Initializing modules...
Downloading registry.opentofu.org/terraform-aws-modules/s3-bucket/aws 4.2.2 for s3_bucket...
- s3_bucket in .terraform/modules/s3_bucket

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0, >= 5.70.0"...
- Installing hashicorp/aws v5.80.0...
- Installed hashicorp/aws v5.80.0 (unauthenticated)

OpenTofu has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that OpenTofu can guarantee to make the same selections by default when
you run "tofu init" in the future.

OpenTofu has been successfully initialized!

You may now begin working with OpenTofu. Try running "tofu plan" to see
any changes that are required for your infrastructure. All OpenTofu commands
should now work.

If you ever set or change modules or backend configuration for OpenTofu,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Module example: private S3 bucket (plan)

```
$ tofu plan
module.s3_bucket.data.aws_canonical_user_id.this[0]: Reading...
module.s3_bucket.data.aws_caller_identity.current: Reading...
module.s3_bucket.data.aws_region.current: Reading...
module.s3_bucket.data.aws_partition.current: Reading...
module.s3_bucket.data.aws_partition.current: Read complete after 0s [id=aws]
module.s3_bucket.data.aws_region.current: Read complete after 0s [id=eu-central-1]
module.s3_bucket.data.aws_caller_identity.current: Read complete after 0s [id=378032484863]
module.s3_bucket.data.aws_canonical_user_id.this[0]: Read complete after 0s [id=88d0aeb4141d5d787d74abc1e5

OpenTofu used the selected providers to generate the following execution plan. Resource actions are indica
  + create

OpenTofu will perform the following actions:

  # module.s3_bucket.aws_s3_bucket.this[0] will be created
  + resource "aws_s3_bucket" "this" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "module-example"
      [...]
      + tags                        = {
          + "Environment" = "development"
          + "Name"        = "Module Example Bucket"
        }
      [...]
    }

  [...continue...]
```

Module example: private S3 bucket (plan)

```
  [...continued...]

  # module.s3_bucket.aws_s3_bucket_acl.this[0] will be created
  + resource "aws_s3_bucket_acl" "this" {
      + acl    = "private"
      + bucket = (known after apply)
      + id     = (known after apply)
    }

  # module.s3_bucket.aws_s3_bucket_public_access_block.this[0] will be created
  + resource "aws_s3_bucket_public_access_block" "this" {
      + block_public_acls       = true
      + block_public_policy     = true
      + bucket                  = (known after apply)
      + id                      = (known after apply)
      + ignore_public_acls      = true
      + restrict_public_buckets = true
    }

Plan: 3 to add, 0 to change, 0 to destroy.

------------------------------------------------------------
```

OpenTofu

- On August 2023 HashiCorp changed license from Mozilla Public License v2.0 to Business Source License v1.1
- Terraform v1.5.x was the latest version under MPL v2.0
- OpenTofu was forked from Terraform to keep its development open-source

Why packaging is important!

After license change HashiCorp also changed terms of use of Terraform Registry, the default service used to distribute providers and modules, in particular under point '2. SERVICES CONTENT.':

> *You may download providers, modules, policy libraries and/or other Services or Content from this website solely for use with, or in support of, HashiCorp Terraform.*

- Services could disappear
- Packaging software in pkgsrc avoid such single point of failure
- (Both Terraform and Terraform providers has never been pre-built by HashiCorp for NetBSD, so packaging them was also the only way to use Terraform!)

Terraform/OpenTofu in pkgsrc

Terraform packages in pkgsrc I

- In main pkgsrc:
  - `net/terraform` 0.12.31
  - `net/opentofu` 1.6.2
  - Several `net/terraform-provider-*` for `net/terraform`

Terraform packages in pkgsrc II

- In pkgsrc-wip:
  - `wip/terraform012` 0.12.31
  - `wip/terraform013` 0.13.7
  - `wip/terraform014` 0.14.11
  - `wip/terraform015` 0.15.5
  - `wip/terraform11` 1.1.9
  - `wip/terraform13` 1.3.10
  - `wip/terraform15` 1.5.7 (last MPL v2.0 version)
  - `wip/opentofu` 1.10.6
  - Several `wip/terraform-provider-*` most of them for Terraform >= 0.13 and OpenTofu
  - `wip/terraform/version.mk` that defines the various Terraform/OpenTofu versions (maybe we can get rid of it and directly define them in corresponding packages?)
  - `wip/terraform/provider.mk` used by `wip/terraform-provider-*` to more easily package them

Terraform packages in pkgsrc III

- Terraform 0.12.x search for providers under `${PREFIX}/bin` out of the box
- Terraform 0.13.x and newer and OpenTofu are patched in order to search for providers under `${PREFIX}/share/terraform/plugis`
- `wip/terraform/provider.mk`'s `TERRAFORM_PROVIDER_LEGACY_INSTALL` if defined add symlinks under `${PREFIX}/bin` so that the provider can be used by Terraform 0.12.x too
- We can package multiple majors because providers always contains a version number as part of the path, e.g.:

```
$ pkg_info -L terraform-provider-aws3
Information for terraform-provider-aws3-3.76.1:

Files:
/usr/pkg/bin/terraform-provider-aws_v3.76.1
/usr/pkg/share/terraform/plugins/registry.opentofu.org/hashicorp/aws/3.76.1/netbsd_amd64/
    terraform-provider-aws
/usr/pkg/share/terraform/plugins/registry.terraform.io/hashicorp/aws/3.76.1/netbsd_amd64/
    terraform-provider-aws
```

Migration path from Terraform to OpenTofu

- Packages in pkgsrc-wip are intended to be imported to pkgsrc
- Permits to upgrade from Terraform 0.12.x to OpenTofu 1.10.6
- Shortest upgrade path is: Terraform 0.12 → Terraform 0.13 → Terraform 0.14 → Terraform 1.x → OpenTofu 1.10.6 [4]

[4] Older OpenTofu documentation suggests to first update to OpenTofu 1.6.x and then newer OpenTofu.

Testing Terraform to OpenTofu migration

- OK, we have packaged several Terraform-s, OpenTofu-s and providers... How to test upgrades?
- Possibly without being worried to destroy resources and cloud bills?
- `net/py-moto` `moto_server` to the rescue! (it can mock ups a lot of AWS services, perfect for such purposes and requirements!)
- Actual tests written with `devel/py-cram` so we have both some notes and tests that checks expected output and exit status

Testing Terraform to OpenTofu migration: checking binaries and their versions I

Listing 1: `requirements.t` cram test

```
This test Terraform upgrades in pkgsrc and goes from Terraform 0.12 to
latest supported OpenTofu.

First we check that we have all needed Terraform versions...

We set CHECKPOINT_DISABLE variable because otherwise Terraform phone
home and check if there are new versions available via
hashicorp/go-checkpoint:

  $ export CHECKPOINT_DISABLE=yes

We check all Terraform versions:

  $ terraform012 version
  Terraform v0.12.* (glob)
  $ terraform013 version
  Terraform v0.13.* (glob)
  $ terraform014 version
  Terraform v0.14.* (glob)
  $ terraform15 version
  Terraform v1.5.* (glob)
  on * (glob)

In pkgsrc-wip we have also packaged other intermediate versions.
Maybe we should test them too.
```

Testing Terraform to OpenTofu migration: checking binaries and their versions II

```
26
27  We also check that we have OpenTofu installed:
28
29    $ tofu version
30    OpenTofu v1.* (glob)
31    on * (glob)
```

To run the tests:

```
$ cram requirements.t
.
# Ran 1 tests, 0 skipped, 0 failed.
```

Testing Terraform to OpenTofu migration: checking upgrade I

- We need some Terraform code to create several AWS resources
- AWS VPC ends up in needing around 30 resources, good enough!

Testing Terraform to OpenTofu migration: checking upgrade II

Listing 2: `versions.tf`

```
terraform {
  required_version = ">= 0.12.31"

  required_providers {
    aws = ">= 3.28, <= 4"
  }
}
```

Listing 3: `main.tf`

```
provider "aws" {
  access_key                  = "test"
  secret_key                  = "test"
  region                      = "us-east-1"
  skip_credentials_validation = true
  skip_metadata_api_check     = true
  skip_requesting_account_id  = true

  endpoints {
    ec2 = "http://localhost:5000"
    iam = "http://localhost:5000"
  }
}

```

Testing Terraform to OpenTofu migration: checking upgrade III

```
15 module "vpc" {
16   source  = "terraform-aws-modules/vpc/aws"
17   version = "3.7.0"
18
19   name = "testing-aws-vpc"
20   cidr = "10.0.0.0/16"
21
22   azs             = ["us-east-1a", "us-east-1b", "us-east-1c"]
23   private_subnets = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
24   public_subnets  = ["10.0.101.0/24", "10.0.102.0/24", "10.0.103.0/24"]
25
26   enable_dns_hostnames = true
27   enable_nat_gateway   = true
28
29   tags = {
30     Terraform = "true"
31   }
32 }
```

Testing Terraform to OpenTofu migration: checking upgrade IV

Listing 4: `upgrades-aws-vpc.t` cram test

```
1  This initialize and test the complete upgrade path from Terraform 0.12
2  to OpenTofu for aws-vpc module.
3  Requires that moto_server is running and listening to localhost:5000.
4
5  Copy the entire module to a temporary directory so we do not need to
6  worry and clean .terraform and terraform.tfstate files:
7
8    $ cp -r "${TESTDIR}/aws-vpc" "${CRAMTMP}/aws-vpc"
9    $ cd "${CRAMTMP}/aws-vpc"
10
11 Set CHECKPOINT_DISABLE environment variable so Terraform do not phone
12 home:
13
14   $ export CHECKPOINT_DISABLE=yes
15
16
17 Initialize the project with Terraform 0.12.x and create the resources:
18
19   $ terraform012 init >/dev/null
20   $ terraform012 plan -detailed-exitcode >/dev/null
21   [2]
22   $ terraform012 apply -auto-approve >/dev/null
23
24
25 Upgrade to Terraform 0.13.x by following
```

Testing Terraform to OpenTofu migration: checking upgrade V

```
26 <https://developer.hashicorp.com/terraform/language/v1.1.x/upgrade-guides
       /0-13>...
27
28 We ensure that there are no pending changes to do:
29
30   $ terraform012 plan -detailed-exitcode >/dev/null
31
32 Run 0.13upgrade command so that provider requirements are rewritten to have
33 explicit source locations:
34
35   $ terraform013 0.13upgrade -yes >/dev/null
36
37 Rewrite providers in the state too and re-init:
38
39   $ terraform013 state replace-provider -auto-approve -- -/aws registry.
         terraform.io/hashicorp/aws >/dev/null
40   $ terraform013 state replace-provider -auto-approve -- -/random registry.
         terraform.io/hashicorp/random >/dev/null
41   $ terraform013 init >/dev/null
42
43 Run apply to complete the upgrade
44
45   $ terraform013 apply -auto-approve >/dev/null
46
47
48 Upgrade to Terraform 0.14.x by following
49 <https://developer.hashicorp.com/terraform/language/v1.1.x/upgrade-guides
       /0-14>...
```

Testing Terraform to OpenTofu migration: checking upgrade VI

```
50
51  We ensure that there are no pending changes to do:
52
53    $ terraform013 plan -detailed-exitcode >/dev/null
54
55  Rewrite providers in the state too and re-init:
56
57    $ terraform014 init >/dev/null
58
59  Run apply to complete the upgrade
60
61    $ terraform014 apply -auto-approve >/dev/null
62
63
64  Upgrade to Terraform 1.5 by following
65  <https://developer.hashicorp.com/terraform/language/v1.5/upgrade-guides/>...
66
67  We ensure that there are no pending changes to do:
68
69    $ terraform014 plan -detailed-exitcode >/dev/null
70
71  Rewrite providers in the state too and re-init:
72
73    $ terraform15 init >/dev/null
74
75  Run apply to complete the upgrade
76
77    $ terraform15 apply -auto-approve >/dev/null
```

Testing Terraform to OpenTofu migration: checking upgrade VII

```
78
79
80  Upgrade to OpenTofu by following
81  <https://opentofu.org/docs/intro/migration/migration-guide/>...
82
83  We ensure that there are no pending changes to do:
84
85    $ terraform15 plan -detailed-exitcode >/dev/null
86
87  Re-init:
88
89    $ tofu init >/dev/null
90
91  Run apply to complete the upgrade:
92
93    $ tofu apply -auto-approve >/dev/null
```

- We need to run `moto_server` in a terminal
- In another terminal we can then run the tests:

```
$ cram upgrades-aws-vpc.t
.
# Ran 1 tests, 0 skipped, 0 failed.
```

Terraform configuration to forbid provider registries

To only honor providers provided by `terraform-provider-*` packages:

Listing 5: `~/.terraformrc`

```
provider_installation {
  filesystem_mirror {
    path = "/usr/pkg/share/terraform/plugins"
    include = ["*/*/*"]
  }
  direct {
    exclude = ["*/*/*"]
  }
}
```

Possible open questions

- Should we also version OpenTofu? (e.g. `opentofu16`, `opentofu110`)
- Is it okay that `terraform-provider-*` does not depends on any Terraform/OpenTofu packages?

Thanks

- Particular thank you: <bsiegert> and <riastradh>

Conclusions

- We have introduced Infrastructure as Code (IaC), Terraform and OpenTofu
- We reflected on why packaging is important
- We have seen how Terraform/OpenTofu and Terraform providers are packaged in pkgsrc
- We have seen how a complete upgrade path from Terraform 0.12.31 to OpenTofu 1.10.6 works

References I

Ron Ross et al.
Enhanced security requirements for protecting controlled unclassified information: A supplement to nist special publication 800-171.
Technical Report NIST Special Publication (SP) 800-172, National Institute of Standards and Technology, Gaithersburg, MD, 2021.

HashiCorp.
Terraform.
https://www.terraform.io/, a.

OpenTofu.
Opentofu.
https://opentofu.org/.

References II

Anton Babenko.
Terraform best practices.
https://www.terraform-best-practices.com/.

Steve Pulec.
Moto - mock aws services.
https://github.com/getmoto/moto.

bitheap.
Cram: It's test time.
https://bitheap.org/cram/.

HashiCorp.
Terms of use - terraform registry.
https://registry.terraform.io/terms, b.

References III

Robert Hafner.
Terraform in Depth - Infrastructure as Code with Terraform and OpenTofu.
Manning Pubblications Co., 2025.
ISBN 9781633438002.

Yevgeniy Brikman.
Terraform: Up and Running - Writing Infrastructure as Code.
O'Reilly Media, 2022.
ISBN 9781098116743.

Scott Winkler.
Terraform in Action.
Manning Pubblications Co., 2021.
ISBN 9781617296895.

Questions?