Anaplan

Repeating yourself or don’t… What?

Brian Menges

Brian Menges

Principal Engineer

Anaplan

Hobbyist brewer

DevOps Advocate

Open Source Contributor

Foster volunteer for animal rescues

Husband and father of two

LinkedIn: https://linkedin.com/in/mengesb

GitHub: https://github.com/mengesb

Twitter: https://twitter.com/mengesb

Plan your next great move

We're changing the way the world's businesses make decisions. We need your help.

Find your fit at **Anaplan.com/careers**

Anaplan

What we'll cover

Iteration in Terraform

- 0.11
- 0.12
- 0.13

Numeric vs Named Indexes

- `count`
- `for`
- `for_each`

Terraform Syntax

- `length()`
- `flatten()`
- `locals`

Tools

- terraform console
- terraform output
- terraform state
- terratest
- jq

Terraform 0.11 Iteration

count

Reference: https://www.terraform.io/docs/configuration-0-11/resources.html#count

```
variable "repos" {
  type        = "list"
  description = "GitHub Repositories"
}

variable "repo_private" {
  type        = "map"
  description = "Map of GitHub Repositories that are private or public (true/false)"
}

resource "github_repository" "repos" {
  count   = "${length(var.repos)}"
  name    = "${var.repos[count.index]}"
  private = "${contains(key(var.repo_private), var.repos[count.index]) ? element(var.repo_private,
var.repos[count.index]) : "false"}"
}
```

Splat notation

Reference: https://www.terraform.io/docs/configuration-0-11/resources.html#using-variables-with-count

```
output "repos" {
  value = "${join(", ", github_repository.repos.*.name)}"
}
```

Terraform 0.12 Iteration

Anaplan

HCL2 introduced

- Double quotes reduced
- Interpolation => Functions
- Types are meaningful
- Modules can have specific provider configurations
- Other improvements
- https://www.terraform.io/docs/configuration/syntax.html
- https://www.hashicorp.com/resources/introducing-terraform-0-12

Proper typing and better expressions

- `number`, `string`, `bool`
- `list()`, `set()`, `map()`, `object()`, `tuple()`
- `null` value means `null`

for_each

```
variable repos {
  type        = map(object({
    name    = string
    private = bool
  }))
  description = “GitHub Repositories”
}

resource github_repository repos {
  for_each = var.repos

  name    = each.value.name == null ? each.key : each.value.name
  private = each.value.private == null ? false : each.value.private
}
```

for

```
locals {
  # Gen object to assign a team/perm to all repositories
  # if var.teams[t.name].global == true
  global_team_repositories = flatten([
    for r in github_repository.repos : [
      for t in github_team.teams : {
        permission = var.teams[t.name].permission == null ? var.default_team_permission :
contains(local.repository_permissions, var.teams[t.name].permission) ? var.teams[t.name].permission :
var.default_team_permission
        repository = r.name
        team_id    = t.id
        team_name  = t.name
      } if (lookup(var.teams, t.name, null) == null ? false : var.teams[t.name].global) == true
    ]
  ])
}

resource github_team_repository global_team_repository {
  for_each   = { for x in local.global_team_repositories : "${x.team_name}:${x.repository}" => x }
  team_id    = each.value.team_id
  repository = each.value.repository
  permission = each.value.permission
}
```

dynamic

```
resource github_repository repos {
  for_each = var.repos

  name    = each.value.name
  private = each.value.private

  dynamic template {
    # NOTE: Only one template{} block allowed
    for_each = each.value.template == null ? [] : [0]

    content {
      owner      = each.value.template.owner
      repository = each.value.template.repository
    }
  }
}
```

dynamic

```
variable ami_owners {
  type        = list(string)
  description = "List of AMI owners to limit search. At least 1 value must be specified. Valid values: an AWS account ID, self (the current
account), or an AWS owner alias (e.g. amazon, aws-marketaplace, microsoft)"
  default     = ["amazon"]
}

variable ami_filters {
  type.       = map(list(string))
  description = "One or more name/[value] pairs to filter"

  default = {
    name                = ["amzn2-ami-hvm*”]
    owner-alias         = ["amazon”]
    root-device-type    = ["ebs”]
    state               = ["available”]
    virtualization-type = ["hvm”]
  }
}

data aws_ami ami {
  most_recent = true
  owners      = var.ami_owners

  dynamic filter {
    for_each = var.ami_filters

    content {
      name   = filter.key
      values = filter.value
    }
  }
}
```

Terraform 0.13 Iteration

Anaplan

Terraform 0.13 iteration

- Modules now more resource-like
- `depends_on` now blocks on a whole module resource
- `for_each` for module resources

Terraform module

```
#
# Terraform:: terraform-github-repository
# Plan:: main.tf
#

resource github_repository r {
  lifecycle {
    ignore_changes = [auto_init, default_branch]
  }

  allow_merge_commit     = var.allow_merge_commit
  allow_rebase_merge     = var.allow_rebase_merge
  allow_squash_merge     = var.allow_squash_merge
  archived               = var.archived
  auto_init              = var.auto_init
  delete_branch_on_merge = var.delete_branch_on_merge
  description            = var.description
  gitignore_template     = var.gitignore_template
  has_downloads          = var.has_downloads
  has_issues             = var.has_issues
  has_projects           = var.has_projects
  has_wiki               = var.has_wiki
  homepage_url           = var.homepage_url
  is_template            = var.is_template
  license_template       = var.license_template
```

```
  name                   = var.name
  private                = var.private
  topics                 = var.topics

  dynamic template {
    for_each = var.template == null ? [] : [0]

    content {
      owner      = var.template.owner
      repository = var.template.repository
    }
  }
}
```

Module iteration

```
#
# Terraform:: examples/foreach-example
# Plan:: main.tf
#

module repository {
  source = "../../"

  for_each = var.repositories

  allow_merge_commit     = each.value["allow_merge_commit"]
  allow_rebase_merge     = each.value["allow_rebase_merge"]
  allow_squash_merge     = each.value["allow_squash_merge"]
  delete_branch_on_merge = each.value["delete_branch_on_merge"]
  description            = each.value["description"]
  has_issues             = each.value["has_issues"]
  has_projects           = each.value["has_projects"]
  has_wiki               = each.value["has_wiki"]
  homepage_url           = each.value["homepage_url"]
  is_template            = each.value["is_template"]
  name                   = each.key
  private                = each.value["private"]
}
```

Numeric vs Named Indexes

Examples

`count` index

```
$ terraform state list
github_repository.repos[0]
github_repository.repos[1]
github_repository.repos[2]
...
github_repository.repos[N]
```

`for_each` index

```
$ terraform state list
github_repository.repos["first_repository"]
github_repository.repos["second_repository"]
github_repository.repos["third_repository"]
...
github_repository.repos["Nth_repository"]
```

Benefits and trade-off

- Indexes `change` with count
- Depending on your provider's behavior on certain field changes, reshuffling of indexes can produce -/+ actions (destroy/create), or change in-place settings resulting in unwanted updates
- Relationships between a count of one resource type, and a count of another are very difficult

```
$ terraform state list
github_repository.repos[0]
github_repository.repos[1]
github_repository.repos[2]
...
github_repository.repos[N]
$ # CHANGE: remove repository that is index 1
$ terraform apply
Terraform v0.11.14
Configuring remote state backend...
Initializing Terraform configuration...
YYYY/MM/DD HH:mm:ss [DEBUG] Using modified User-Agent: Terraform/0.11.14 TFE/YYYYMM-#
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be persisted to local or remote state storage.

github_repository.repository[0]: Refreshing state... [id=first_repository]
github_repository.repository[1]: Refreshing state... [id=second_repository]
github_repository.repository[2]: Refreshing state... [id=third_repository]
...
github_repository.repository[N]: Refreshing state... [id=Nth_repository]
```

Example continued

```
An execution plan has been generated and is shown below.
Resource actions are indicated with the following
symbols:
   ~ update
   - destroy

Terraform will perform the following actions:

# github_repository.repository[1] will be updated ~
resource "github_repository" "repository" {
    allow_merge_commit = true
    allow_rebase_merge = true
    allow_squash_merge = true
    archived = false
    auto_init = false
    default_branch = "master"
    description = "The second repository" ~> "The
third repository"
  ~ etag = "W/\"_____________________________________\""
~> (computed)
  ~ full_name = "ORG/second_repository" ~>
"ORG/third_repository"
  ~ git_clone_url =
"git://github.com/ORG/second_repository.git" ~>
"git://github.com/ORG/third_repository.git"
    has_downloads = false
    has_issues = false
    has_projects = false
    has_wiki = false
  ~ html_url =
"https://github.com/ORG/second_repository" ~>
"https://github.com/ORG/third_repository"
  ~ http_clone_url =
"https://github.com/ORG/second_repository.git" ~>
"https://github.com/ORG/third_repository.git"
  ~ id = "second_repository" ~>
"third_repository"
  ~ name = "second_repository" ~>
"third_repository"
    private = true
  ~ ssh_clone_url =
"git@github.com:ORG/second_repository.git" ~>
"git@github.com:ORG/third_repository.git"
  ~ svn_url =
"https://github.com/ORG/second_repository" ~>
"https://github.com/ORG/third_repository"
    topics = []
}
```

Example continued

```
# github_repository.repository[2] will be destroyed
- resource "github_repository" "repository" { -
  - allow_merge_commit = true -> null
  - allow_rebase_merge = true -> null
  - allow_squash_merge = true -> null
  - archived = false -> null
  - auto_init = false -> null
  - default_branch = "master" -> null
  - description = "The third repository" -> null
  - etag = "W/\"____________________________________\""
-> null
  - full_name = "ORG/third_repository" -> null
  - git_clone_url =
"git://github.com/ORG/third_repository.git" -> null
  - has_downloads = false -> null
  - has_issues = false -> null
  - has_projects = false -> null
  - has_wiki = false -> null
  - html_url =
"https://github.com/ORG/third_repository" -> null
  - http_clone_url =
"https://github.com/ORG/third_repository.git" ->
null
  - id = "third_repository" -> null
  - name = "third_repository" -> null
  - private = true -> null
  - ssh_clone_url =
"git@github.com:ORG/third_repository.git" -> null
  - svn_url =
"https://github.com/ORG/third_repository" -> null
  - topics = [] -> null
}

Plan: 0 to add, 1 to change, 1 to destroy.
```

Benefits and trade-off continued

- Indexes don’t re-shuffle with `for_each`
- Relationships between resources are easier to maintain

```
$ terraform state list
github_repository.repos["first_repository”]
github_repository.repos["second_repository"]
github_repository.repos["third_repository"]
...
github_repository.repos["Nth_repository"]
github_team.team["First"]
github_team.team["Second"]
github_team.team["Third"]
...
$ # CHANGE: remove index "second_repository"
$ terraform apply
Terraform v0.12.24
Configuring remote state backend... Initializing
 Terraform configuration...
YYYY/MM/DD HH:mm:ss [DEBUG] Using modified User-Agent:
Terraform/0.12.24 TFE/YYYYMM-#
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan,
but will not be persisted to local or remote state
storage.

github_team.team["First"]: Refreshing state... [id=###]
github_team.team["Second"]: Refreshing state... [id=###]
github_team.team["Third"]: Refreshing state... [id=###]
...
```

```
github_repository.repository["first_repository"]: Refreshing
state... [id=first_repository]
github_repository.repository["second_repository"]: Refreshing
state... [id=second_repository]
github_repository.repository["third_repository"]: Refreshing
state... [id=third_repository]
...
github_repository.repository["Nth_repository"]: Refreshing
state... [id=Nth_repository]
github_team_repository.global_team_repository["First:first_re
pository"]: Refreshing state... [id=###:first_repository]
github_team_repository.global_team_repository["Second:first_r
epository"]: Refreshing state... [id=###:first_repository]
github_team_repository.global_team_repository["Third:first_re
pository"]: Refreshing state... [id=###:first_repository]
github_team_repository.global_team_repository["First:second_r
epository"]: Refreshing state... [id=###:second_repository]
github_team_repository.global_team_repository["Second:second_
repository"]: Refreshing state... [id=###:second_repository]
github_team_repository.global_team_repository["Third:second_r
epository"]: Refreshing state... [id=###:second_repository]
...
```

Example continued

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

Terraform will perform the following actions:

  # github_repository.repository["second_repository"] will be destroyed
  - resource "github_repository" "repository" { - allow_merge_commit = true -> null
  - allow_rebase_merge = true -> null
  - allow_squash_merge = true -> null
  - archived = false -> null
  - auto_init = false -> null
  - default_branch = "master" -> null
  - description = "The third repository" -> null
  - etag = "W/\"____________________________________\"" -> null
  - full_name = "ORG/second_repository" -> null
  - git_clone_url = "git://github.com/ORG/second_repository.git" -> null
  - has_downloads = false -> null
  - has_issues = false -> null
  - has_projects = false -> null
  - has_wiki = false -> null
  - html_url = "https://github.com/ORG/second_repository" -> null
  - http_clone_url = "https://github.com/ORG/second_repository.git" -> null
  - id = "second_repository" -> null
  - name = "second_repository" -> null
  - private = true -> null
  - ssh_clone_url = "git@github.com:ORG/second_repository.git" -> null
  - svn_url = "https://github.com/ORG/second_repository" -> null
  - topics = [] -> null
}
```

Example continued

```
# github_team_repository.global_team_repository["First:second_repository"] will be destroyed
- resource "github_team_repository" "global_team_repository" {
  - etag = "W/\"_____________________________________\"" -> null
  - id = "###:second_repository" -> null
  - permission = "push" -> null
  - repository = "second_repository" -> null
  - team_id = "###" -> null
}

# github_team_repository.global_team_repository["Second:second_repository"] will be destroyed
-  resource "github_team_repository" "global_team_repository" {
  - etag = "W/\"_____________________________________\"" -> null
  - id = "###:second_repository" -> null
  - permission = "push" -> null
  - repository = "second_repository" -> null
  - team_id = "###" -> null
}

# github_team_repository.global_team_repository["Third:second_repository"] will be destroyed
-  resource "github_team_repository" "global_team_repository" {
  - etag = "W/\"_____________________________________\"" -> null
  - id = "###:second_repository" -> null
  - permission = "push" -> null
  - repository = "second_repository" -> null
  - team_id = "###" -> null
}

Plan: 0 to add, 0 to change, 4 to destroy.
```

Benefits and trade-off continued

Resources generated from resources are easier to manage with named indexes, however complicated to understand sometimes

```
resource github_repository repos {
  for_each = var.repos
  ...
}

resource github_team teams {
  for_each = var.teams
  ...
}

locals {
  # Gen object to assign a team/perm to all repositories
  # if var.teams[t.name].global == true
  global_team_repositories = flatten([
    for r in github_repository.repos : [
      for t in github_team.teams : {
        permission = var.teams[t.name].permission
        repository = r.name
        team_id    = t.id
        team_name  = t.name
      } if (lookup(var.teams, t.name, null) == null ? false : var.teams[t.name].global) == true
    ]
  ])
}

resource github_team_repository global_team_repository {
  for_each   = { for x in local.global_team_repositories : "${x.team_name}:${x.repository}" => x }
  team_id    = each.value.team_id
  repository = each.value.repository
  permission = each.value.permission
}
```

Syntax

length()

```
resource "github_repository" "repos" {
  count    = "${length(var.repos)}"
  name     = "${var.repos[count.index]}"
  private = "${contains(key(var.repo_private), var.repos[count.index]) ? element(var.repo_private,
var.repos[count.index]) : "false"}"
}
```

- Returns a `string` in 0.11.x
- Returns a `number` in 0.12.x and 0.13.x
- Resources based on `length()` using `count` can have an on/off feel, or numeric index iteration

flatten()

```
locals {
  # Gen object to assign a team/perm to all repositories
  # if var.teams[t.name].global == true
  global_team_repositories = flatten([
    for r in github_repository.repos : [
      for t in github_team.team : {
        permission = var.teams[t.name].permission == null ? var.default_team_permission :
contains(local.repository_permissions, var.teams[t.name].permission) ? var.teams[t.name].permission :
var.default_team_permission
        repository = r.name
        team_id    = t.id
        team_name  = t.name
      } if (lookup(var.teams, t.name, null) == null ? false : var.teams[t.name].global) == true
    ]
  ])
}
```

- Useful when generating relationships across multiple data structures

Tools

Tools

terraform console

- Built-into the terraform binary
- Allows you to browse state and test functions
- https://www.terraform.io/docs/commands/console.html

terraform output

- See your output variables
- https://www.terraform.io/docs/commands/output.html

terraform state

- Advanced state management
- https://www.terraform.io/docs/commands/state/index.html

terratest

- Go testing library
- Most useful for module testing, still useful for plan tests
- https://terratest.gruntwork.io/

jq

- Useful for parsing JSON formatted outputs
- Excellent for parsing the terraform state file or terraform outputs
- https://stedolan.github.io/jq/
- https://jqplay.org/

References

- https://learn.hashicorp.com/
- https://registry.terraform.io/
- https://www.hashicorp.com/blog/new-terraform-tutorials-on-count-and-for-each/
- https://www.terraform.io/upgrade-guides/0-13.html
- https://www.terraform.io/docs/configuration
- https://terratest.gruntwork.io/
- https://github.com/mengesb

Thank you!

Q
A