The road to NRK's private Terraform registry

Who am I?

- Stig Otnes Kolstad (stigok)
- Member of Hackeriet since ~2015 (?)
- Working in NRK with platform operations and SRE tasks
  - Go, Python, Terraform, Kubernetes

How we work together in NRK's platform team

- Establish best practices
- Configure and run shared infrastructure
  - Network
  - Monitoring
  - Terraform modules
- Voluntary *Admin groups* that plan around and evolve specific areas of the technology landscape
  - Kubernetes
  - Infrastructure as Code (IaC)
  - ... and more

What is Terraform?

- “A tool to codify cloud APIs into declarative config files” (ref)
- Provisions resources through Terraform *providers* (API clients)
- This helps to keep our infrastructure configuration
  - Reuseable
  - Backed up
  - Tracked with commit- and changelogs
  - GitOps’ed (in some cases, hopefully more)

HashiCorp Configuration Language

```
$ cat <<EOF > main.tf

provider "kubernetes" {
  config_path    = "~/.kube/config"
  config_context = "my-context"
}

resource "kubernetes_namespace" "example" {
  metadata {
    name = "my-first-namespace"
  }
}

EOF
```

Terraform CLI demonstration

```
$ terraform init > /dev/null
$ terraform apply
[...]
Terraform will perform the following actions:
  # kubernetes_namespace.example will be created
  + resource "kubernetes_namespace" "example" {
      + id = (known after apply)

      + metadata {
          + generation       = (known after apply)
          + name             = "my-first-namespace"
          + resource_version = (known after apply)
          + uid              = (known after apply)
        }
    }

Plan: 1 to add, 0 to change, 0 to destroy.
Do you want to perform these actions? (yes/NO)
```

Why are we using it?

- Reduce point-and-click to a minimum
- Track, review and validate changes (Git, GitHub and Actions)
  - Automated checks
    - Sanity checks using built-in Terraform commands
    - Scanning IaC config using Trivy
- Backed up infrastructure configuration
- Abstractions through modules

My Terraform glossary

- Terraform
  - **Provider:** a binary file used to talk to external sources (e.g. API's, local programs, self-contained logic)
  - **Module:** a collection of one or more Terraform configuration files, often parameterised and some times reusable
  - **Registry:** a package library holding versioned modules and providers
  - **Configuration repo:** a Git repository containing configuration we actually run `terraform apply` on.
  - **Module repo:** a Git repository containing code for one or more modules to be included/referenced by a configuration repository

In the Beginning

- Terraform mono-repo that held **all** shared infrastructure
- Grew bigger every day
  - increasing complexity
  - time to apply
  - insecurity
  - raised bar of entry
- A need for separation of concern (in terms of people, teams and code responsibility)
  1) New configuration repos
  2) New module repos for sharing modules between them

In a configuration repository

```
module "my_module_instance" {
  source     = "git@github.com/nrkno/mymodule"
  some_param = “foo”
  team       = “my-team”
}
```

- The ground is moving under our feet
  - What version does person A have locally, and what does B have?
  - Is there a newer version around?
  - Where and how do I find out?
  - What has changed since last time I updated, and is it safe to upgrade?
- We need to reference specific versions

How do we fix this?

- Module versioning (module repos)
  - Enforce repository rules (branch restrictions)
    - No direct push to main branch
    - Require reviewers for pull requests
    - Perform automated code checks
  - Actions to run after merge
    - Automatic release of new versions
- Shared workflows

GitHub Actions reusable workflows

- Workflow to validate Terraform configuration (link)
  - `terraform fmt && terraform init && terraform validate`
  - Trivy vulnerability scan (link)
  - Generate Terraform documentation
- Workflow for validating conventional commit messages and automatic release of new versions (link)
  - Conventional commits
  - Semantic versioning (SemVer)
  - with “latest” tag updates (v1.2.5 == v1.2 == v1)
  - Creates a *release* in GitHub with a changelog attached

In a configuration repository v2

```
module "my_module_instance" {
  source     = "git@github.com/nrkno/mymodule?ref=1.2.3"
  some_param = "foo"
  team       = "my-team"
}
```

- ~~The ground is moving under our feet~~
  - ~~What version does person A have locally, and what does B have?~~
  - Is there a newer version around?
  - Where and how do I find out?
  - ~~What has changed since last time I updated, and is it safe to upgrade?~~
- ~~We need to reference specific versions~~

Dependabot

- Deeply integrated in GitHub
- Supports many different package managers and ecosystems
- Automatically creates pull requests with version bumps
- Terraform support since June 10th 2021
  - Lockfiles (for providers) introduced in Terraform v0.14 (Dec 2020)
  - Modules from a registry always referenced via a version number

A private Terraform registry?

- We don’t want to publish all our modules
- Are there any private Terraform registries available?
- The Terraform Registry Protocol is actually pretty simple...
- Should we create our own?
- Can we use GitHub as a backend?
- Can we use custom authentication?

Terraform Registry Protocol - Module Source References

- A module is referenced in the configuration by a URL consisting of hostname/namespace/name/system
  - If **hostname** is missing, registry.terraform.io is implied
  - **Namespace** is typically the owner of the package, or the name of the group responsible for it (e.g. hashicorp, nrkno, cisco)
  - **Name** is the name of the module (e.g. kubernetes-cluster)
  - **System** is used in the official registry to differentiate individual modules targeted at different cloud providers (i.e. systems). This value can be anything.

```
module "my_module_instance" {
  source  = "terraform-registry.nrk.cloud/nrkno/mymodule/generic"
  version = “1.2.3”
}
```

Terraform Registry Protocol - Service Discovery

```
# source = “terraform-registry.nrk.cloud/nrkno/mymodule/generic”


$ curl https://terraform-registry.nrk.cloud/.well-known/terraform.json
{
  "modules.v1": "/v1/modules/"
}
```

Terraform Registry Protocol - List module versions

```
# source = “terraform-registry.nrk.cloud/nrkno/mymodule/generic”


$ curl https://terraform-registry.nrk.cloud/v1/modules/nrkno/mymodule/generic/versions
{
   "modules": [
      {
         "versions": [
            {"version": "1.0.0"},
            {"version": "1.1.0"},
            {"version": "1.2.3"},
            {"version": "2.0.0"},
            {"version": "2.3.4"}
         ]
      }
   ]
}
```

Terraform Registry Protocol - Download module version

```
# source = “terraform-registry.nrk.cloud/nrkno/mymodule/generic”
```

```
$ curl -i https://terraform-registry.nrk.cloud/v1/modules/nrkno/mymodule/generic/2.3.4/download
HTTP/1.1 204 No Content
Content-Length: 0
X-Terraform-Get: git::ssh://git@github.com/nrkno/my-terraform-module.git?ref=2.3.4
```

What will this solve?

- Know when a module has been updated by configuring Dependabot
- What has changed?
  - Read the changelog/release notes in the GitHub release
  - Determine if you need to update at all
  - Know if it requires configuration changes or recreation of resources

Our private Terraform Registry implementation

- Uses GitHub as a backend for modules and their version tags
  - Searches repositories tagged with *terraform-module* in our org
- Clients authenticate using predefined static tokens
  - direnv (.envrc) for loading environment variables (ref)
    - `export TF_TOKEN_terraform__registry_nrk_cloud=my-secret-password`
  - can be used to track repositories depending on a module
- The registry returns git+ssh:// for module URL's, offloading end-user authentication to GitHub's SSH server
- Open source
  - Issues discussing new features are very welcome!

Dependabot configuration

```
version: 2
registries:
  terraform-registry.nrk.cloud:
    type: terraform-registry
    url: https://terraform-registry.nrk.cloud
    token: ${{ secrets.PLATTFORM_TERRAFORM_REGISTRY_NRK_CLOUD }}
updates:
- package-ecosystem: terraform
  directory: "/"
  registries:
    - terraform-registry.nrk.cloud
  schedule:
    interval: daily
    time: "08:00"
    timezone: "Europe/Oslo"
  open-pull-requests-limit: 5
  reviewers:
  - nrkno/plattform
```

In a configuration repository v3

```
module "my_module_instance" {
  source  = "terraform-registry.nrk.cloud/nrkno/myodule/generic"
  version = “1.2.3”

  some_param = "foo"
  team       = "my-team"
}
```

- ~~The ground is moving under our feet~~
  - ~~What version does person A have locally, and what does B have?~~
  - ~~Is there a newer version around?~~ We can make this better
  - ~~Where and how do I find out?~~ We can make this better
  - ~~What has changed since last time I updated, and is it safe to upgrade?~~
- ~~We need to reference specific versions~~

Wishes for the future

- A frontend for the registry with modules, documentation and examples
  - List of all availalable modules
  - Single page documentation and examples
- Improved testing of our modules
  - Automation
  - Actually create and test the infrastructure described
    - How to avoid/minimize spending?
- Automation to improve our daily Terraform workflow
  - Atlantis
  - Terraform Cloud

Thoughts

- Conventional commits drives semantic versioning
- Automatically enforcing a given Git workflow in teams/organisations is not enough. Training, tough reviews and shared understanding of the issues around versioning is a must.
- Not having to publish new versions to a registry is nice, but brings other issues like cache invalidation and new version discovery strategies
- Updating the Terraform documentation is a bit annoying. The GITHUB_TOKEN does not have write permissions to the repository in a shared workflow. However, a deploy key can be configured.

Resources

- https://github.com/nrkno/github-workflow-terraform-config/
- https://github.com/nrkno/github-workflow-semantic-release/
- https://github.com/nrkno/terraform-registry/
- https://developer.hashicorp.com/terraform/internals/module-registry-protocol
- https://stigok.com/

Questions

- Please ask!