SCALING TERRAFORM AS A SERVICE

EFFICIENT INFRASTRUCTURE MANAGEMENT AT SCALE

SEAN DUXBURY

- Cloud Tech Lead at IAG
- Trying to keep user data from the darkweb
- Automates to avoid doing hard work

Linkedin

INITIAL SCALING ISSUES WITH TERRAFORM

- Poor redundancy. If the state file becomes corrupted everything is lost
- Takes to long to run everything
- Issues with breaking up core infra into layers.
  - Need to persist values from one layer to another
  - Need to manage dependencies of one layer to the next

```
.
└── infra/
    ├── 0-bootstrap.tf
    ├── 1-org.tf
    ├── 2-environments-dev.tf
    ├── 2-environments-prod.tf
    ├── 3-networks-dev.tf
    ├── 3-networks-prod.tf
    ├── 4-team-a.tf
    └── 4-team-b.tf
```

INCOMES TERRAGRUNT

- Benefits:
  - Passing variables between steps
  - Dependency tracking across layers
  - Multiple state files to limit blast radius

```
.
└── infra/
    ├── 0-bootstrap/
    │   └── main.tf
    ├── 1-org/
    │   └── orgpolicy.tf
    ├── 2-environments/
    │   ├── dev/
    │   │   └── vpc.tf
    │   └── prod/
    │       └── vpc.tf
    ├── 3-networks/
    │   ├── dev/
    │   │   └── network.tf
    │   └── prod/
    │       └── network.tf
    └── 4-teams/
        ├── team-a/
        │   └── project.tf
        └── team-b/
            └── project.tf
```

TERRAGRUNT PAIN BEGINS

- Challenges:
  - As more teams onboarded, use of jinja templates to abstract boilerplate
  - This causes cognitive complexity to skyrocket
  - For TG to keep all the layers in memory as needed, cicd memory requirements start exploding
  - Still takes a long time to run, so we enable parallelism on TG
  - TG starts slamming the cloud api’s and hits quota limits
  - TG is still single threaded so even in parallel mode runner cpu and memory requirements go crazy
  - Still validating all infra every run. This ensures no drift, but means if there is an issue with any one section then the process fails and no-one can push to prod.

```
.
└── infra/
    ├── 0-bootstrap/
    │   └── main.tf
    ├── 1-org/
    │   └── orgpolicy.tf
    ├── 2-environments/
    │   ├── dev/
    │   │   └── vpc.tf
    │   └── prod/
    │       └── vpc.tf
    ├── 3-networks/
    │   ├── dev/
    │   │   └── network.tf
    │   └── prod/
    │       └── network.tf
    └── 4-teams/
        ├── team-a/
        │   └── project.tf
        └── team-b/
            └── project.tf
```

TRUE PARALLELISM – BACK TO TERRAFORM

- Solution:
  - Each section must be able to run independently. This required accessing outputs from previous layers stored in a bucket
  - 4-teams was split into a new repo
  - All the templates was hidden behind 1 single `provisioner` module. This means that new teams only need to manage a single tf module.
  - Extra features are then loaded in as yaml config in the same level
  - Now each team can run in a github actions matrix job.
    With a concurrency limits (avoid api quotas, and multiple runs overwriting each other)

```
.
└──core infra/
        ├── 0-bootstrap/
        │       └── main.tf
        ├── 1-org/
        │       └── orgpolicy.tf
        ├── 2-environments/
        │       ├── dev/
        │       │       └── vpc.tf
        │       └── prod/
        │               └── vpc.tf
        ├── 3-networks/
                ├── dev/
                │       └── network.tf
                └── prod/
                        └── network.tf
```

```
.
└──4-teams/
        ├── team-a/
        │       └── project.tf
        │       └── iam.yaml
        │       └── firewall.yaml
        │
        └── team-b/
                └── project.tf
                └── iam.yaml
                └── firewall.yaml
```

Cloud Bucket

0-bootstrap-outputs.json
1-org-outputs.json
2-environments-dev-outputs.json
2-environments-prod-outputs.json
3-networks-dev-outputs.json
3-networks-prod-outputs.json
4-team-a-outputs.json
4-team-b-outputs.json

SUMMARY

- Benefits:
  - Runtime from ~45mins down to ~2 mins (for a single team project)
    Down to ~15mins for top to bottom
  - Able to run on scaled out runners that don't need specialty memory requirements. (32gig+)
    Which also saves $
  - A failure in one space is unseen by other runs
  - Has now scaled to 250+ team projects and counting.
  - More and more features are loaded into the front-door module. But this complexity is hidden from the end users.
  - Separate repo for the 4-teams uses a less privileged SA to further limit blast radius.

- Considerations:
  - There is now not true interlevel dependency. This can mean that changes at a higher level will only be picked up on subsequent runs. Given things like vpc's, and networks don't change id's if ever this hasn't cause much problems.

```
.
└──4-teams/
       ├── team-a/
       │      └── project.tf
       │      └── iam.yaml
       │      └── firewall.yaml
       │
       └── team-b/
              └── project.tf
              └── iam.yaml
              └── firewall.yaml
```

LESSONS LEARNED

- Don’t be afraid to “kill your baby”. Just because it was my idea doesn’t mean we should hang on to it.
- Keep iterating as you grow, something that works early on, might not work as you scale.
- Ripping terragrunt out requires a lot of manual state manipulation. (Moved blocks make this more bearable)
- Keeping things independent allows for much better scaling

SEAN DUXBURY

Linkedin