Terraform

Hva, hvorfor og hvordan

Workshop @ Amedia

2024-06-12

Innhold

- Slik håndterer vi infrastruktur i skyen
- Dette er Terraform
- Derfor er det lurt å bruke Terraform
- Slik fungerer det (*ish*)
- Noen vanlige operasjoner
- Håndtering av «drift»
- `amedia-apps-*` - vi må rydde litt
- Annet

HashiCorp
Terraform

Slik håndterer vi infrastruktur i skyen

ClickOps™

Den raskeste veien til mål

video

CLI

Presist, repeterbart

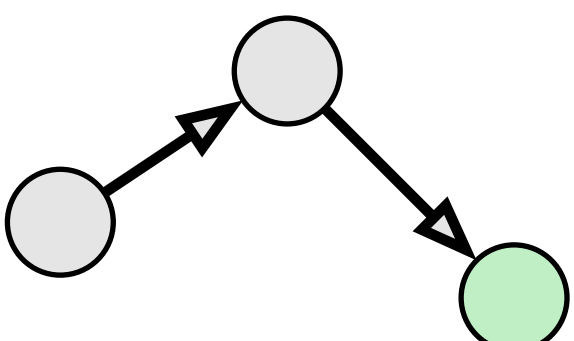
```
gcloud --project='amedia-adp-test' pubsub \
  topics create 'my-topic' \
  --message-retention-duration=1d

gcloud --project='amedia-adp-test' pubsub \
  subscriptions create 'my-subscription' \
  --topic='my-topic'
```

create-infrastructure.sh

Fellestrekk

ClickOps og CLI

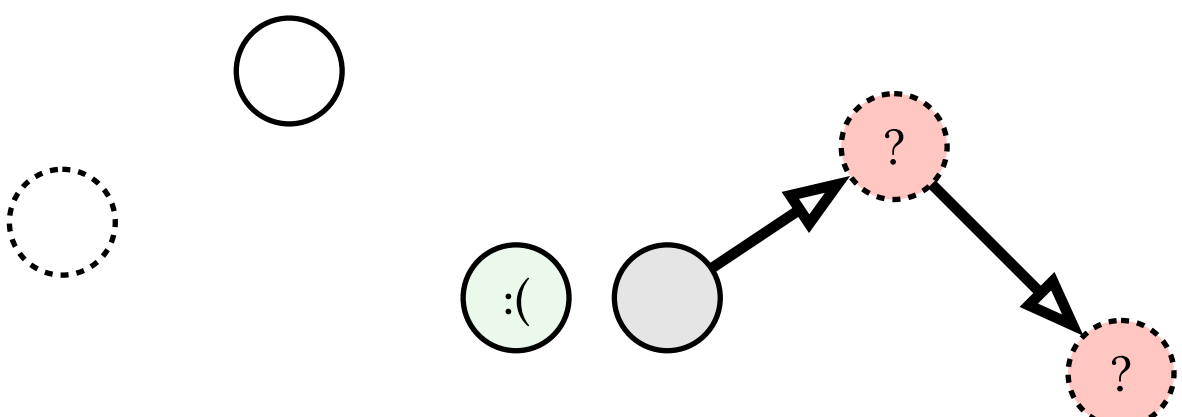
- Sekvens av steg som forhåpentligvis tar deg til mål
- Beskriver handlinger, ikke tilstand
- Er ikke «idempotent»



Fellestrekk

ClickOps og CLI

- Sekvens av steg som forhåpentligvis tar deg til mål
- Beskriver handlinger, ikke tilstand
- Er ikke «idempotent»



Dette er Terraform

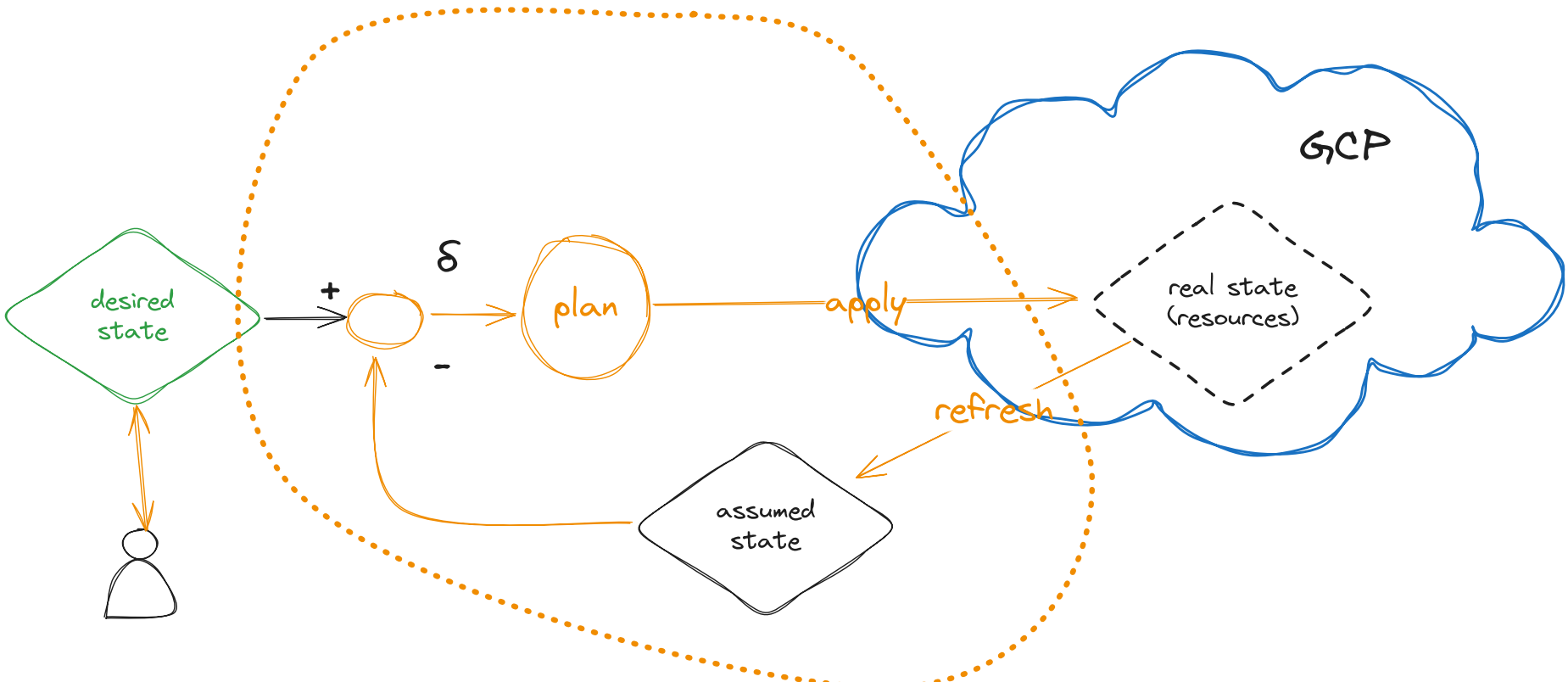
infrastruktur som kode
(infrastrucure as code, IaC)

Man beskriver ønsket tilstand

Jeg vil ha:

- Et pubsub topic som heter `my-fancy-topic`
- En subscription som heter `my-fancy-topic-subscription`
  - ack-deadline på 20 sekunder
  - pusher til et endepunkt

```
resource "google_pubsub_topic" "the-topic" {
  name = "my-fancy-topic"
}

resource "google_pubsub_subscription" "the-subscription" {
  name = "my-fancy-topic-subscription"
  topic = google_pubsub_topic.the-topic.name
  ack_deadline_seconds = 20
  push_config {
    push_endpoint = "https://example.com/notify"
  }
}
```


GCP
desired state
+
δ
-
plan
apply
real state (resources)
refresh
assumed state

Derfor er det lurt å bruke Terraform

Å bruke Terraform er lurt fordi

- koden *er* infrastrukturen $\Longrightarrow$ «dokumentasjonen» vedlikeholdes automatisk
- historikk ved hjelp av `git`
- tjenestene blir mer reproduserbare

En tjeneste består av både kode *og* infrastruktur

Slik fungerer det (*ish*)

Eksempel: Et pubsub topic og en subscription

```
resource "google_pubsub_topic" "the-topic" {
  name = "my-fancy-topic"
}

resource "google_pubsub_subscription" "the-subscription" {
  name = "my-fancy-topic-subscription"
  topic = google_pubsub_topic.the-topic.name
  ack_deadline_seconds = 20
  push_config {
    push_endpoint = "https://my-endpoint.example.com/notify"
  }
  description = "Subscribes to id ${google_pubsub_topic.the-topic.id}"
}
```

Eksempel: Et pubsub topic og en subscription

```
resource "google_pubsub_topic" "the-topic" {
  name = "my-fancy-topic"
}
```

ressurstype, navn (scopet), argument

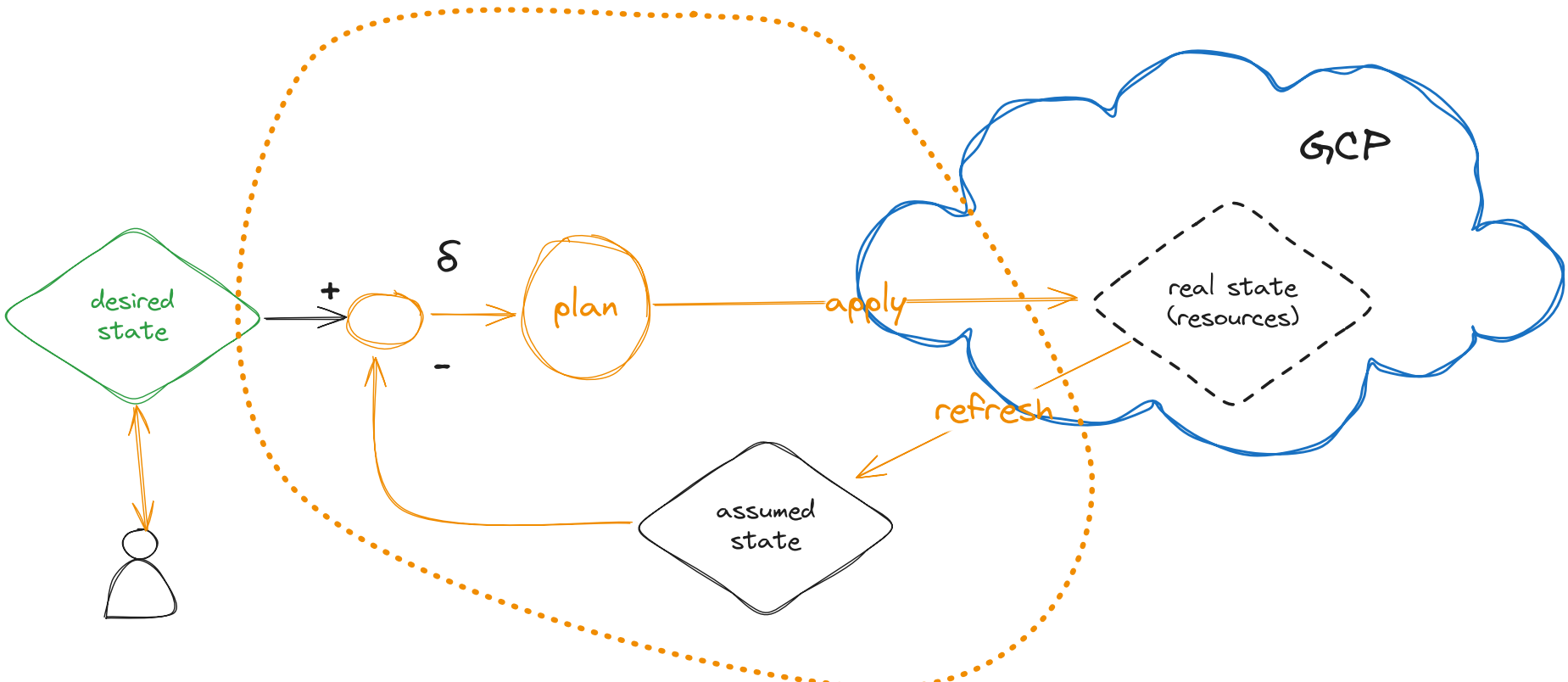
Eksempel: Et pubsub topic og en subscription

```
resource "google_pubsub_topic" "the-topic" {
  name = "my-fancy-topic"
}

resource "google_pubsub_subscription" "the-subscription" {

  topic = google_pubsub_topic.the-topic.name




  description = "Subscribes to id ${google_pubsub_topic.the-topic.id}"
}
```

referanse til argument og attributt

Eksempel: Et pubsub topic og en subscription

```
resource "google_pubsub_topic" "the-topic" {
  name = "my-fancy-topic"
}

resource "google_pubsub_subscription" "the-subscription" {
  name = "my-fancy-topic-subscription"
  topic = google_pubsub_topic.the-topic.name
  ack_deadline_seconds = 20
  push_config {
    push_endpoint = "https://my-endpoint.example.com/notify"
  }
  description = "Subscribes to id ${google_pubsub_topic.the-topic.id}"
}
```


GCP
desired state
+
δ
-
plan
apply
real state (resources)
refresh
assumed state

Prosessen i grove trekk

- utvikleren endrer på en **konfigurasjon** (f.eks. legger til ressurser i `main.tf`)
- terraform sammenlikner konfigurasjonen med en **tilstand** og lager en **plan**
- terraform får tilgang til skyen ved hjelp av en **provider**
- ved hjelp av providerens API-er gjør terraform endringer i ressurser
- tilstanden er nå oppdatert slik at den stemmer med konfigurasjonen

konfigurasjon filer som slutter med `.tf`

tilstand vanligvis `default.tfstate` – lagret lokalt eller i en bøtte (en **backend**) – beskriver hvilke ressurser terraform tracker og tilstanden til disse ressursene – holdes i synk ved hver `terraform plan/refresh`

plan en sekvens av handlinger som utgjør en diff, og fører til at tilstanden er slik konfigurasjonen tilsier

provider en plugin som beskriver hvilke ressurser som er tilgjengelige, og hvordan de konfigureres

Noen vanlige operasjoner

legge til en ressurs

`count` og `for_each`

Med demonstrasjon fra lokal kjøring av Terraform

```
git clone https://github.com/glennib/terraform-workshop.git
cd terraform
```

Her er det mulig å kjøre litt lokalt i et prosjekt du selv har tilgang til. Bytt ut `amedia-adp-test` med et annet testprosjekt.

```
sed -i 's/amedia-adp-test/YOUR-PROJECT-HERE/' main.tf
gcloud auth application-default login
terraform init
```

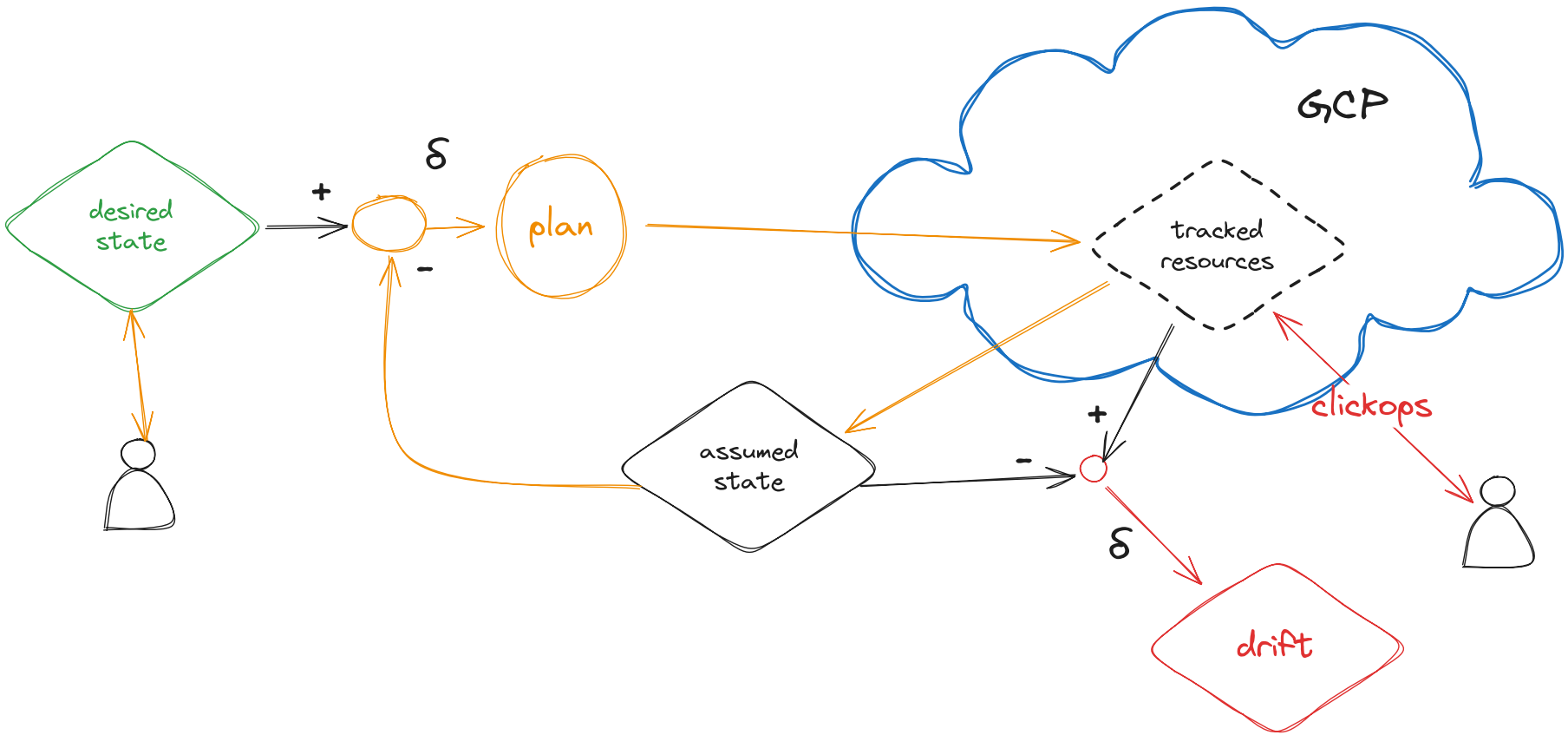
Legge til en ressurs

Bruke `count` og `for_each`

Lage en modul

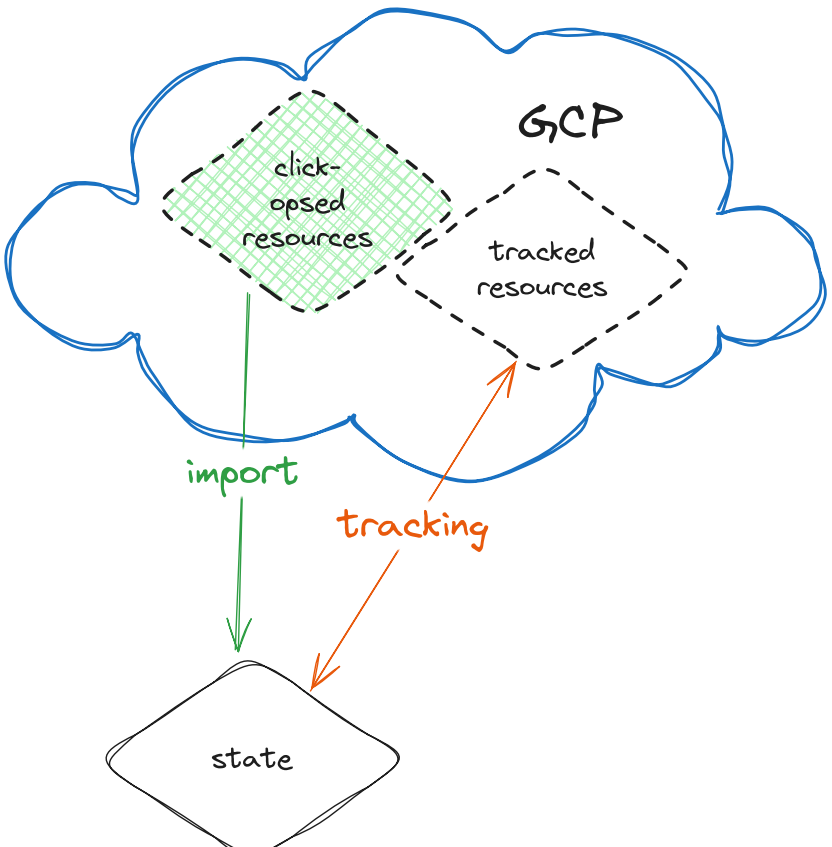
Håndtering av «drift»

Hva om noen tuller det til med ClickOps™?


GCP
desired state
+
δ
-
plan
tracked resources
assumed state
+
-
clickops
δ
drift

Noen har lagt til en ressurs med ClickOps™

- Dette er helt OK, Terraform tracker ikke ressurser som ikke er i tilstanden, og vil derfor ikke gjøre noe med dem.
- Om man ønsker å tracke dem med Terraform etter at de er opprettet, kan man importere dem.


GCP
click-opsed resources
tracked resources
import
tracking
state

1. Konfigurér ressursblokker som tilsvarer ressursene som allerede eksisterer

   ```
   resource "google_pubsub_topic" "the-topic" { ... }
   ```

2. Importér de eksisterende ressursene

   ```
   terraform import google_pubsub_topic.the-topic projects/<project-id>/
   topics/<topic-id>
   ```

3. Se om konfigurasjonen matcher tilstanden

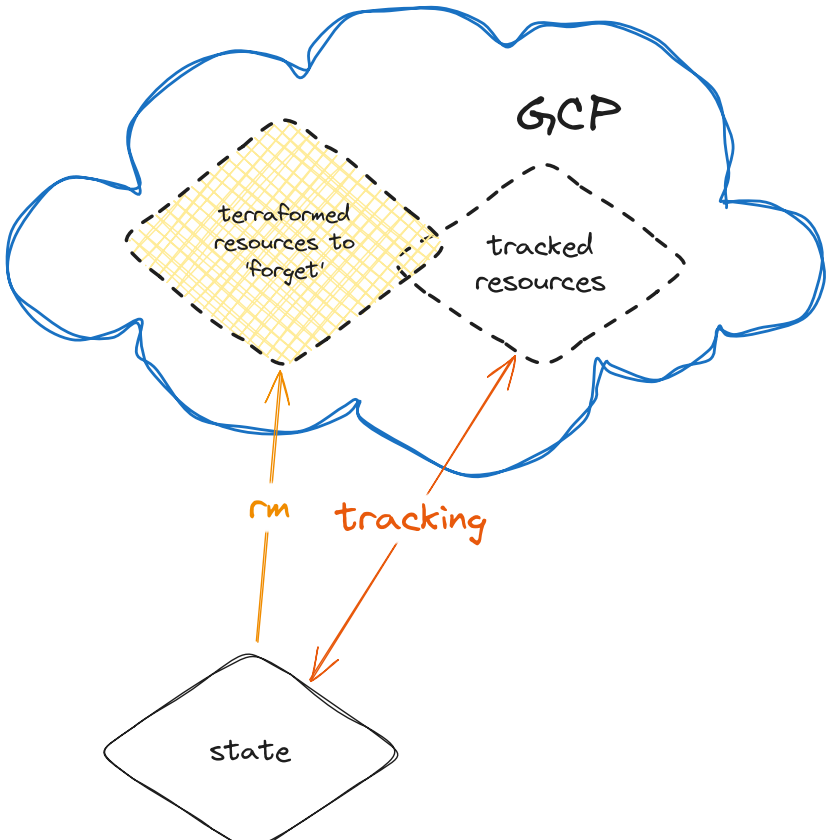
   ```
   terraform plan
   ```

4. Om konfigurasjonen matcher, vil Terraform ikke gjøre noen endringer. Om ikke, må konfigurasjonen tilpasses, eller så må man akseptere at tilstanden endres:

   ```
   terraform apply
   ```

Har man Terraformet noe som man ønsker å håndtere videre med ClickOps™?

- Da må man få Terraform til å «glemme» ressursen, og fjerne den fra konfigurasjonen.


GCP
terraformed resources to 'forget'
tracked resources
rm
tracking
state

1. Fjern ressursen fra konfigurasjonen

   ```
   // fjern denne blokken
   resource "google_pubsub_topic" "the-topic" { ... }
   ```

2. «Glem» ressursen fra tilstanden

   ```
   terraform state rm google_pubsub_topic.the-topic
   ```

3. Se om konfigurasjonen matcher tilstanden

   ```
   terraform plan
   ```

   Planen skal ikke vise noen endringer.

Hva om man ønsker å terraforme ressursen, men ikke *alle* attributtene?

1. Oppdater konfigurasjonen med en `lifecycle`-blokk som inneholder `ignore_changes`

```
resource "google_pubsub_topic" "the-topic" {
  name = "my-fancy-topic"
  lifecycle {
    ignore_changes = ["message_retention_duration"]
  }
}
```

2. Sjekk at en ny plan ikke medfører endringer.

Jeg ønsker å endre navn på eller flytte en ressurskonfigurasjon

1. Flytt eller gi nytt navn til ressursen i konfigurasjonen

```
//                    endra fra "the-topic"
resource "google_pubsub_topic" "hot-topic" {
  name = "my-fancy-topic"
}
```

2. Flytt til riktig adresse i tilstanden

```
terraform state mv \
  google_pubsub_topic.the-topic \
  google_pubsub_topic.hot-topic
```

3. Sjekk at en ny plan ikke medfører endringer.

Jeg ønsker å flytte en frittstående ressurs inn i en modul

`terraform state mv` fungerer også mellom, inn, og ut av moduler:

```
terraform state mv \
  google_pubsub_topic.the-topic \
  module.my-fancy-module.google_pubsub_topic.hot-topic
```

`amedia-apps-*` - vi må rydde litt

```
git clone git@github.com:amedia/terraform-selvbetjening.git
cd projects/amedia-apps-test
```

Alt er flatt + mapper er moduler

```
amedia-apps-test
├── aiven
│   ├── ...
├── atlantis.tf
├── eavis.tf
├── einherjer.tf
├── encamp-write-pubsub.tf
├── imported-buckets.tf
├── jotunheim.tf
├── midgard-bucket.tf
├── midgard.tf
└── ...
```

```
// midgard.tf
resource "google_storage_bucket"
"midgard" {
  // ...
}
// einherjer.tf
resource "google_storage_bucket"
"einherjer" {
  // ...
}
```

Alt er flatt + mapper er moduler

```
amedia-apps-test
├── aiven
│   ├── ...
├── atlantis.tf
├── eavis.tf
├── einherjer.tf
├── encamp-write-pubsub.tf
├── imported-buckets.tf
├── jotunheim.tf
├── midgard-bucket.tf
├── midgard.tf
└── ...
```

```
// midgard.tf
resource "google_storage_bucket"
"midgard" {
  // ...
}
// einherjer.tf
resource "google_storage_bucket"
"einherjer" {
  // ...
}
```

Alt er flatt + mapper er moduler

```
.
├── modules
│   ├── midgard
│   │   ├── main.tf
│   │   ├── output.tf
│   │   ├── README.md
│   │   └── variables.tf
│   └── yggdrasil/...
└── projects
    ├── amedia-apps-prod
    │   └── main.tf
    └── amedia-apps-test/...
```

projects/amedia-apps-prod/main.tf:

```
module "midgard" {
  source = "../modules/midgard"
  // parametrize this
}
module "yggdrasil" {
  source = "../modules/yggdrasil"
  // parametrize this
}
// ...
```

Gjør det likt i amedia-apps-test.

Annet

- `terraform fmt -recursive` og `terraform validate`
- Kan jeg hoppe over ressurser om noen andre har skapt mye drift?
  (Ja, se `terraform apply -target=<resource-address>`).
- Hva er Atlantis?
- Alternativer til Terraform?

Oppsummering

- Terraform (og andre *IaC*-verktøy) er en måte å håndtere infrastruktur gjennom å beskrive ønsket tilstand framfor sekvenser av handlinger.
- Bruk av *IaC* gjør at tjenestene våre blir mer reproduserbare og bedre dokumentert.
- Vi har sett på en del vanlige operasjoner og håndtering av «drift».
- Vi har sett på tilstand og gjort litt opprydding i `amedia-apps-test`.