Terraform

Hva, hvorfor og hvordan

Workshop @ Amedia

2024-05-28

Agenda

- Hvordan lager vi ressurser i skyen?
- Hva er Terraform?
- Hvorfor er det lurt å bruke det?
- Hvordan fungerer det egentlig?
- Vanlige operasjoner
- Spesialoperasjoner

HashiCorp
Terraform

Hvordan lager vi ressurser i skyen?

ClickOps™

Den raskeste veien til mål

video

CLI

Presist

```
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'
```

CLI

Presist

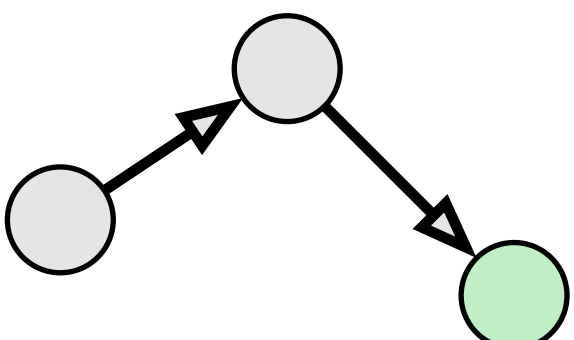
```
gcloud --project='amedia-adp-test' pubsub \
  topics create 'my-topic' \
  --message-retention-duration=1d

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

create-infrastructure.sh

Fellestrekk

ClickOps og gcloud

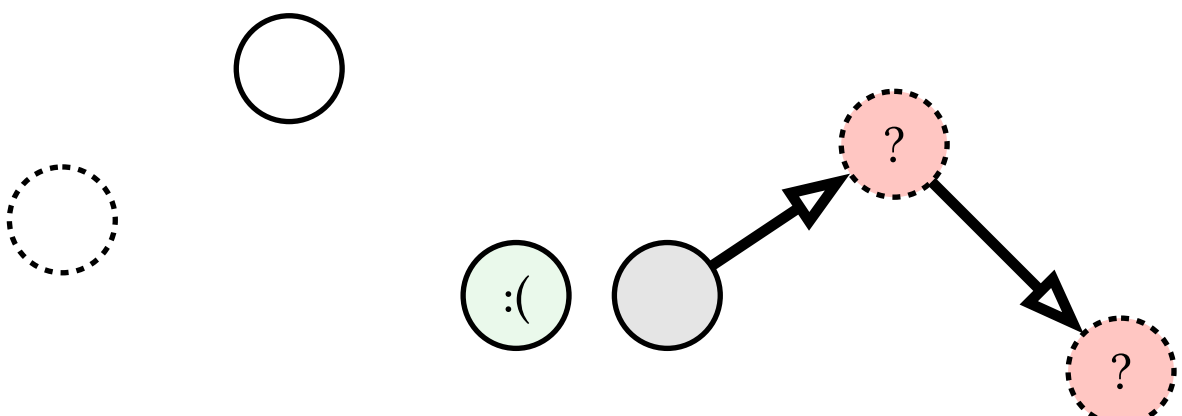
- Sekvens av steg som forhåpentligvis tar deg til mål
- Beskriver handlinger, ikke tilstand
- Er ikke «idempotent»



Fellestrekk

ClickOps og gcloud

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



Hva er Terraform?

infrastruktur som kode
(infrastrucure as code, IaC)

Ønsket tilstand

Jeg vil ha:

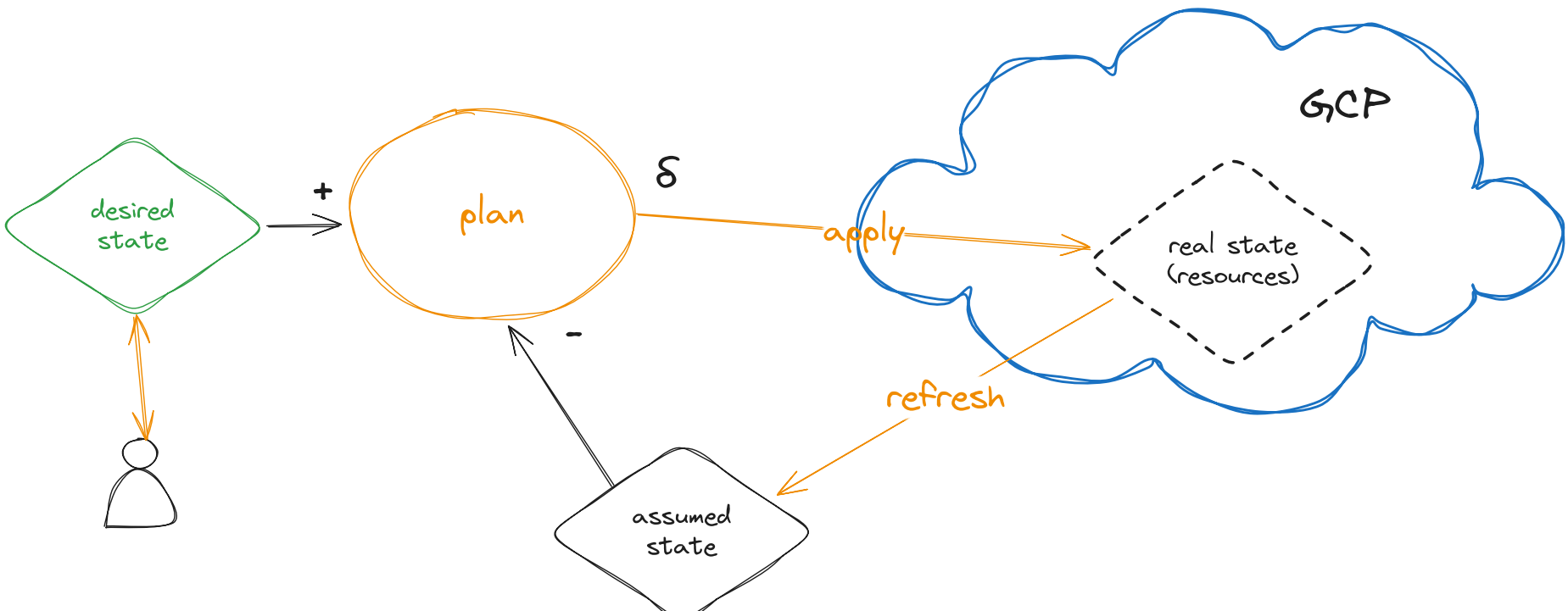
- Et pubsub topic som heter `my-fancy-topic`
- En subscription som heter `my-fancy-topic-subscription`
  - ack-deadline på 20 sekunder
  - pusher til et endepunkt

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://my-endpoint.example.com/notify"
  }
}
```


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

Hvorfor bruke Terraform?

Hvorfor bruke Terraform?

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

Hvorfor bruke Terraform?

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

En tjeneste består av både kode *og* infrastruktur

Hvordan fungerer Terraform?

Eksempel

```
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

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

ressurstype, navn (scopet), argument

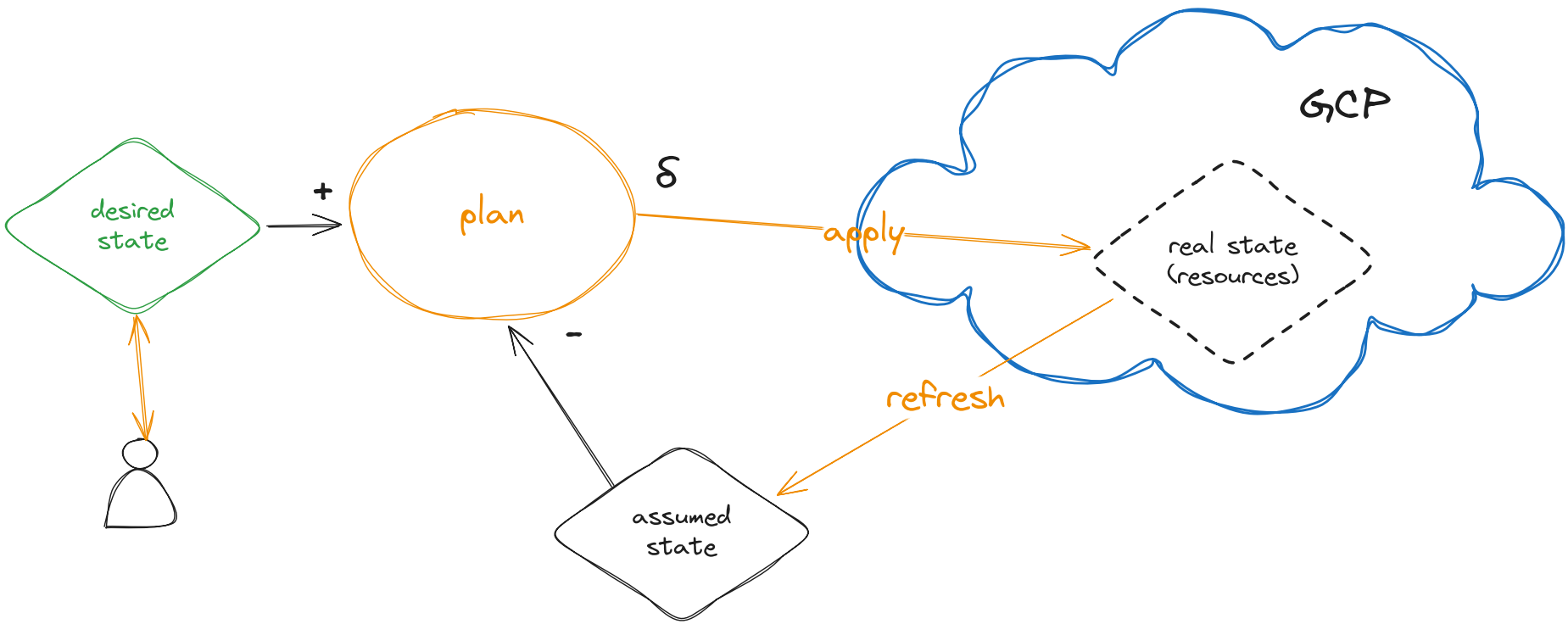
Eksempel

```
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

```
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}"
}
```


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

I grove trekk

- utvikleren endrer på en **konfigurasjon** (f.eks. legger til ressurser i `main.tf`)

konfigurasjon filer som slutter med `.tf`

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**

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

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**

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

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

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

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

Vanlige operasjoner

• legge til en ressurs • `count` og `for_each` • lage en modul

Med demonstrasjon fra lokal kjøring av Terraform

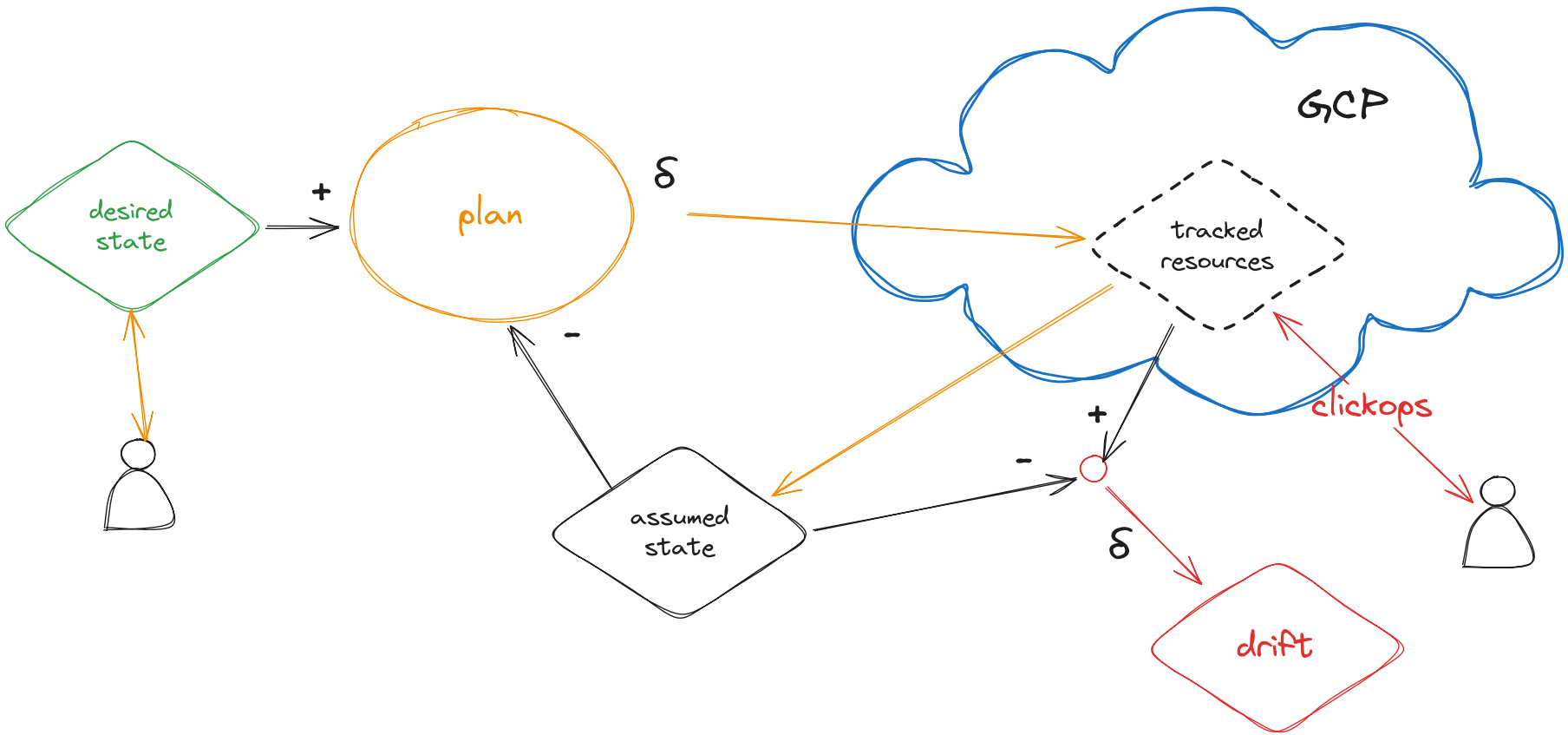
Legge til en ressurs

Bruke `count` og `for_each`

Lage en modul

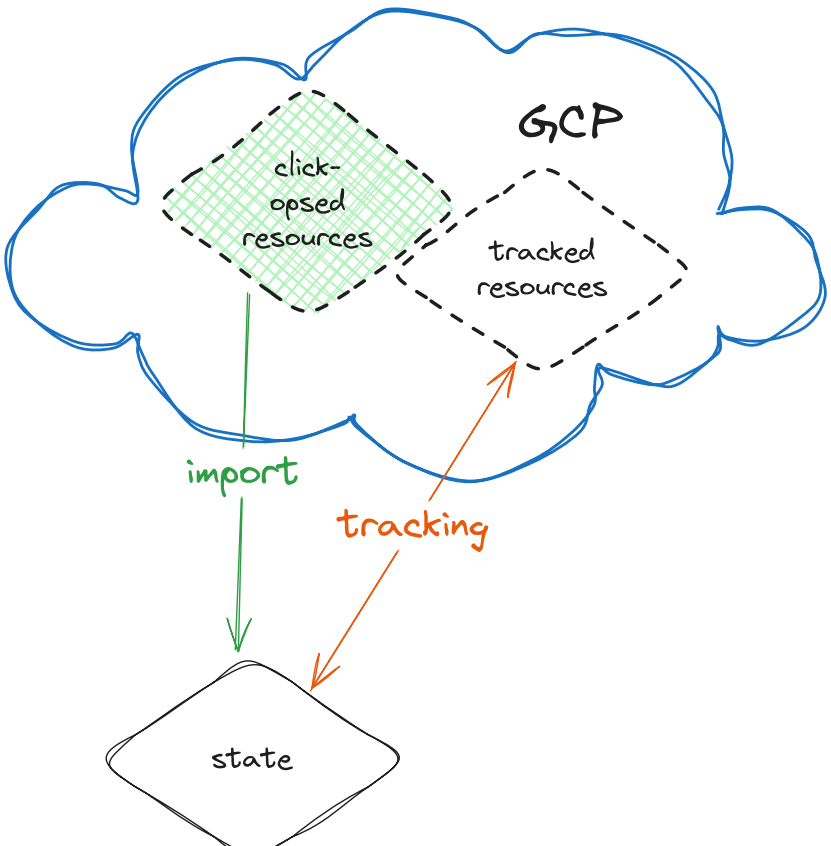
Drift

Hva om noen tuller det til med ClickOps™?


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

Drift

- Har noen lagt til en ressurs?
  - 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

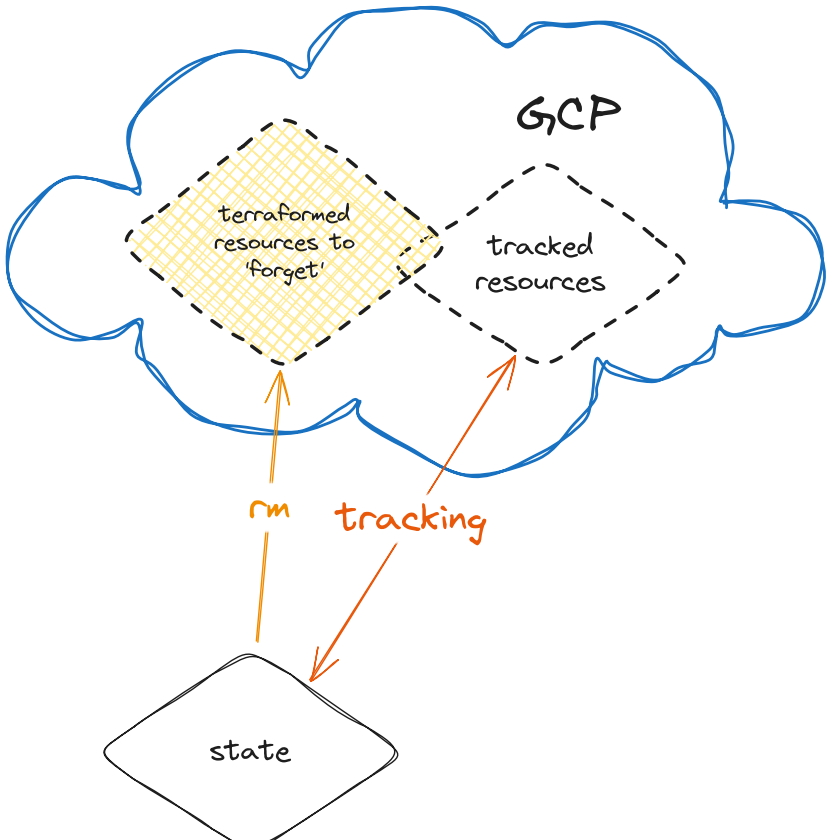
   ```
   terraform plan
   ```

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
   ```

Drift

- 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.

Drift

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

Drift

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

1.
2.

Drift

- Jeg ønsker å endre navn på eller flytte en ressurskonfigurasjon

Drift

- Jeg ønsker å endre navn på eller flytte en ressurskonfigurasjon

1.

2.

3.