

# Terraform

Hva, hvorfor og hvordan

Workshop @ Amedia

2024-05-28

# 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»
- Infrastrukturen i amedia-adp-\*
- 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'
```

# CLI

*Presist, repeterbart*

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

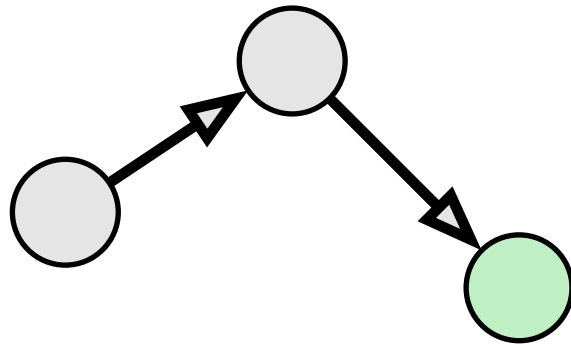
```
gcloud --project='amedia-adp-test' pubsub \  
  subscriptions create 'my-subscription' \  
  --topic='my-topic'
```

[create-infrastructure.sh](#)

# Fellestrekke

*ClickOps og CLI*

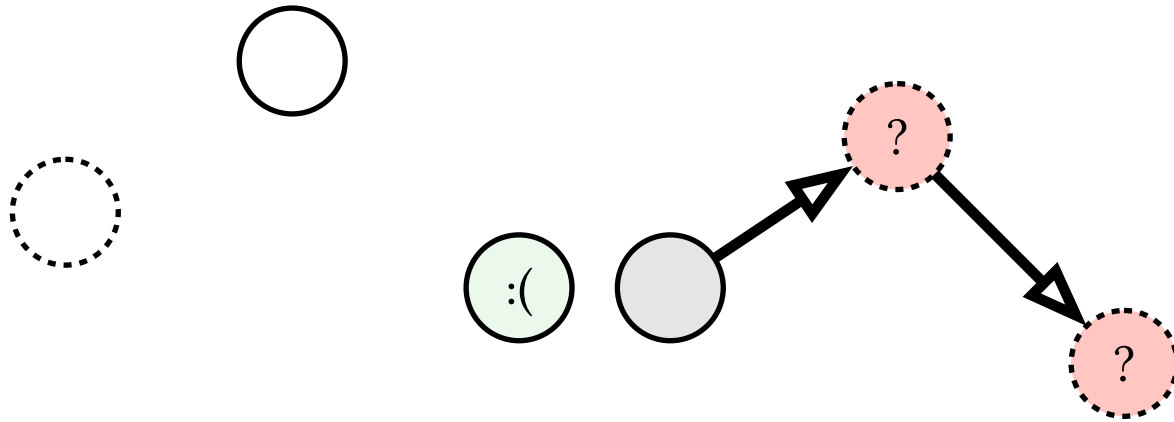
- Sekvens av steg som forhåpentligvis tar deg til mål
- Beskriver handlinger, ikke tilstand
- Er ikke «idempotent»



# Fellestrekke

*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  
*(infrastructure as code, IaC)*

## Man beskriver ønsket tilstand

Jeg vil ha:

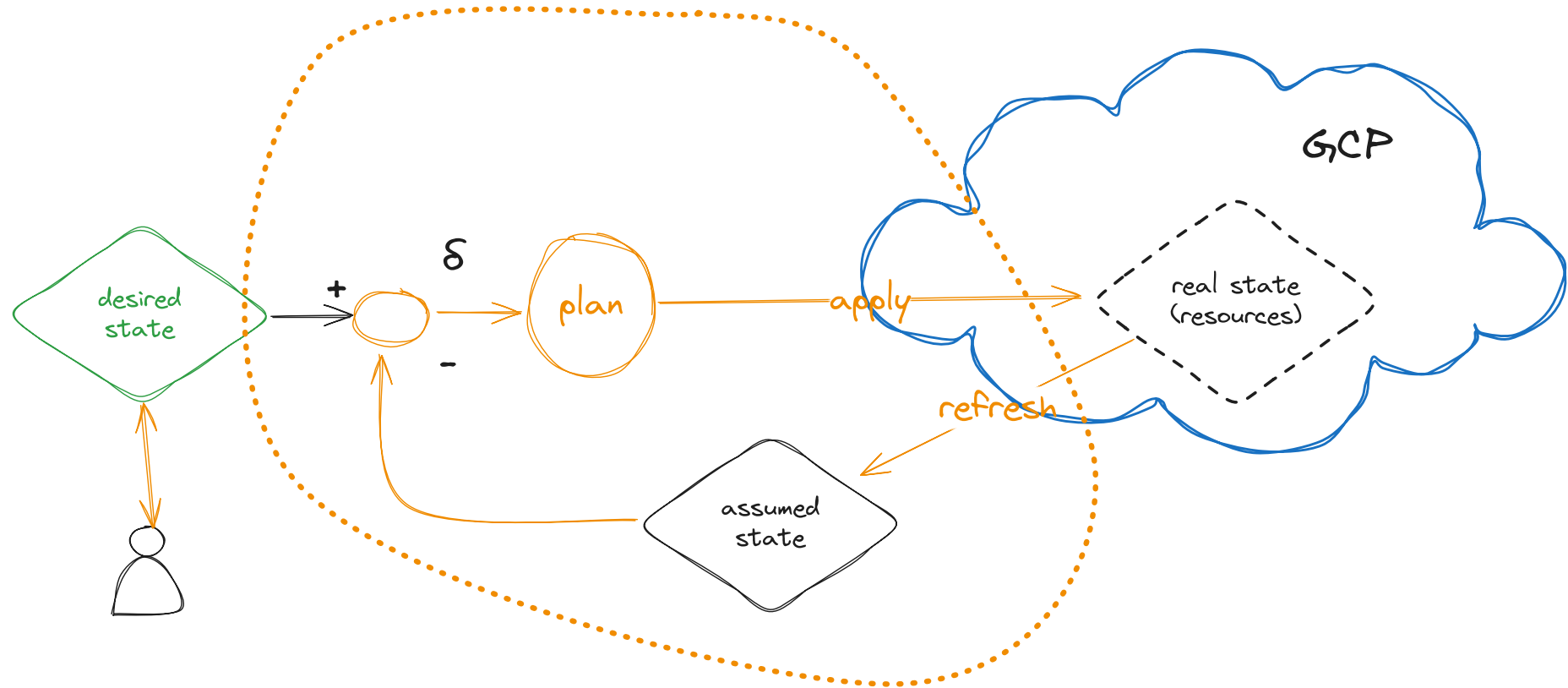
- Et pubsub topic som heter my-fancy-topic
- En subscription som heter my-fancy-topic-subscription
  - ack-deadline på 20 sekunder
  - pusher til et endepunkt

## 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"  
  }  
}
```



**Derfor er det lurt å bruke Terraform**

## Å bruke Terraform er lurt fordi

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

## Å bruke Terraform er lurt fordi

- koden *er* infrastrukturen  $\implies$  «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}"  
}
```

Slik fungerer det (*ish*)

## Eksempel: Et pubsub topic og en subscription

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

ressurstype, navn (scopet), argument

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" {
```

```
  topic = google_pubsub_topic.the-topic.name
```

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

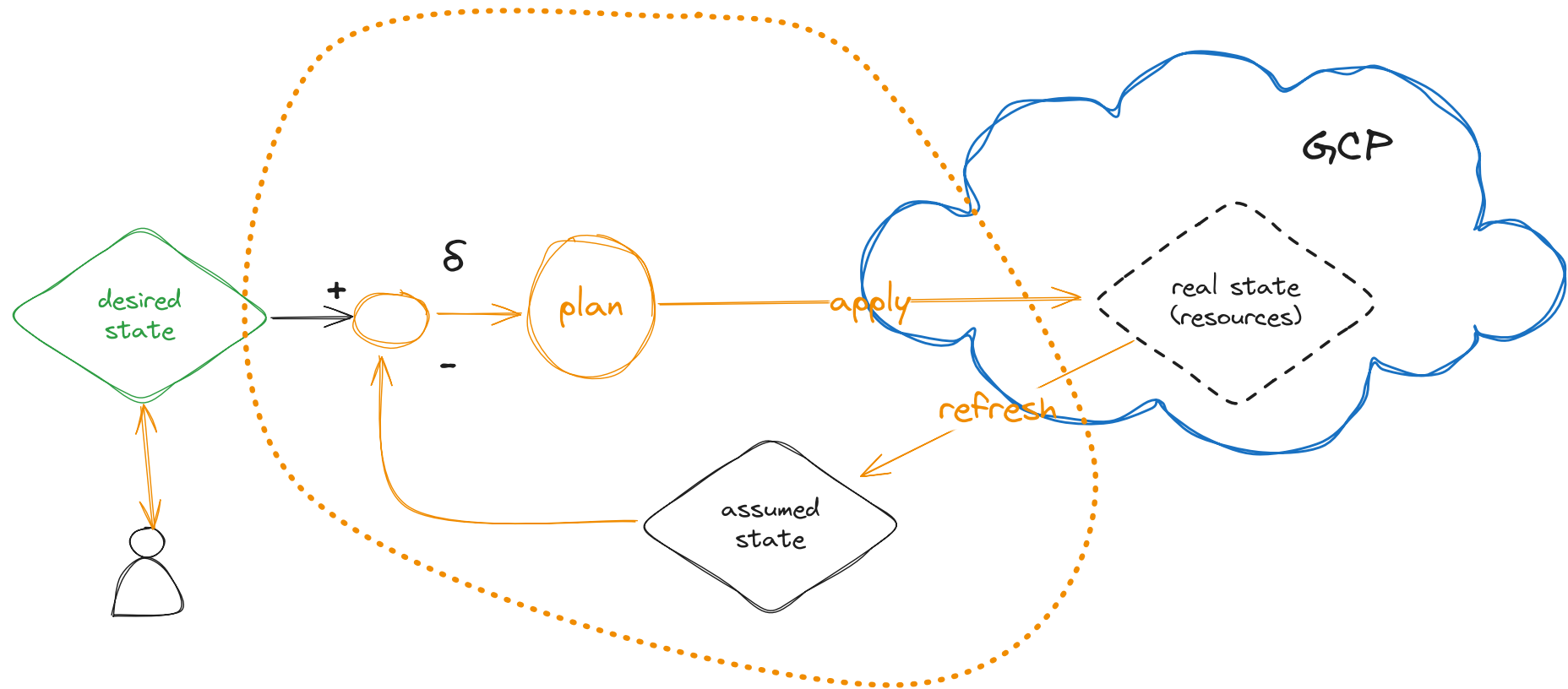
referanse til argument og attributt

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}"  
}
```

Slik fungerer det (*ish*)



Slik fungerer det (*ish*)

## Proessen i grove trekk

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

Slik fungerer det (*ish*)

## Proessen i grove trekk

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

**konfigurasjon** filer som slutter med `.tf`

## Proessen 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 bønne (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



## Proessen 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 bølge (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

## Proessen 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 bønne (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

## Proessen 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 bølge (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 • lage en modul

Med demonstrasjon fra lokal kjøring av Terraform

**Legge til en ressurs**

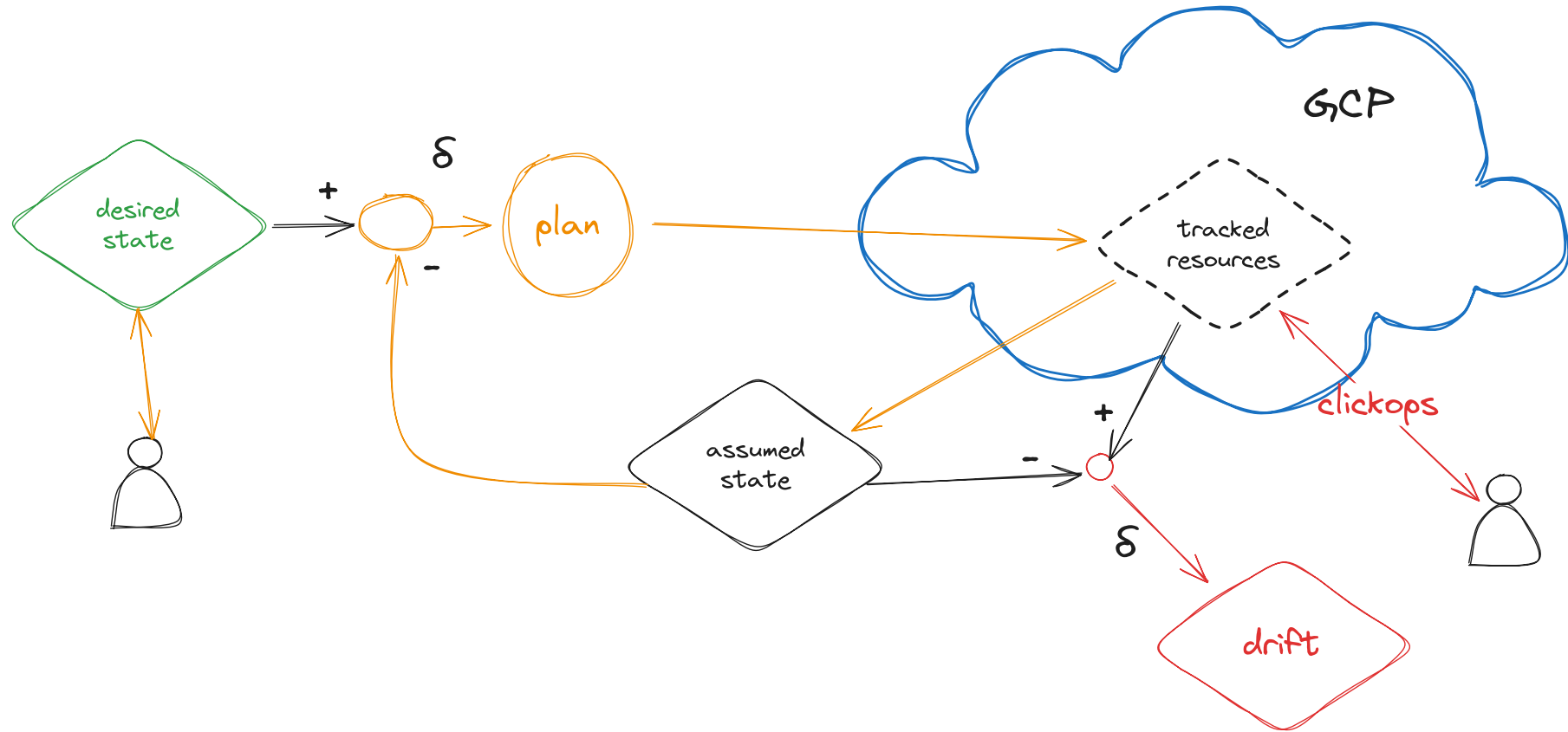
**Bruke count og for\_each**

**Lage en modul**

# Håndtering av «drift»

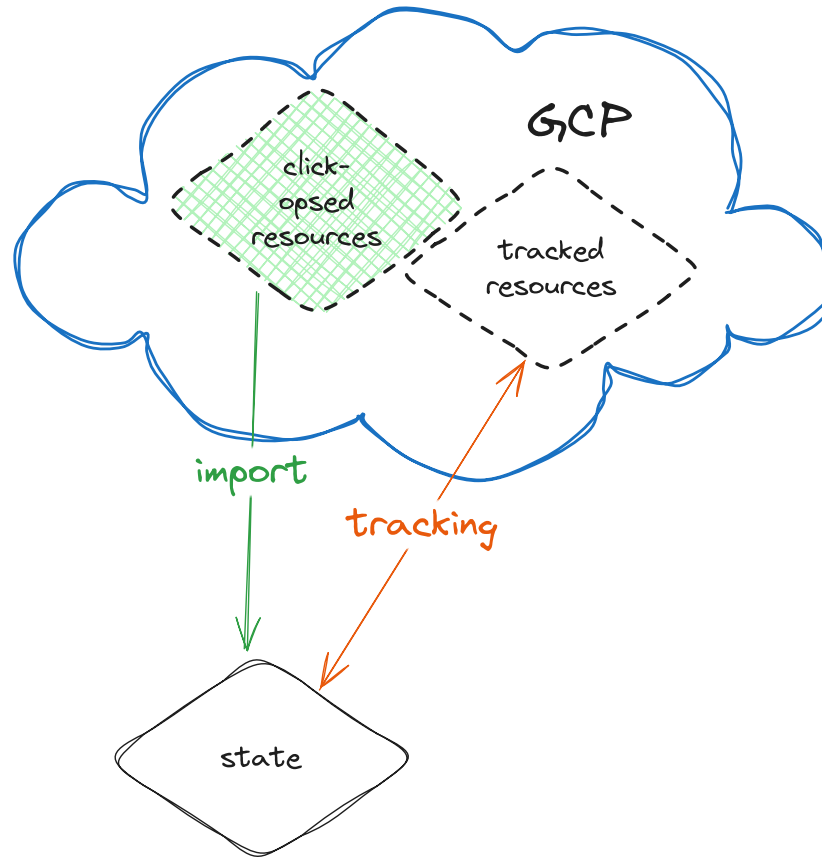
Hva om noen tuller det til med ClickOps™?





## 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.



1. Konfigurer 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

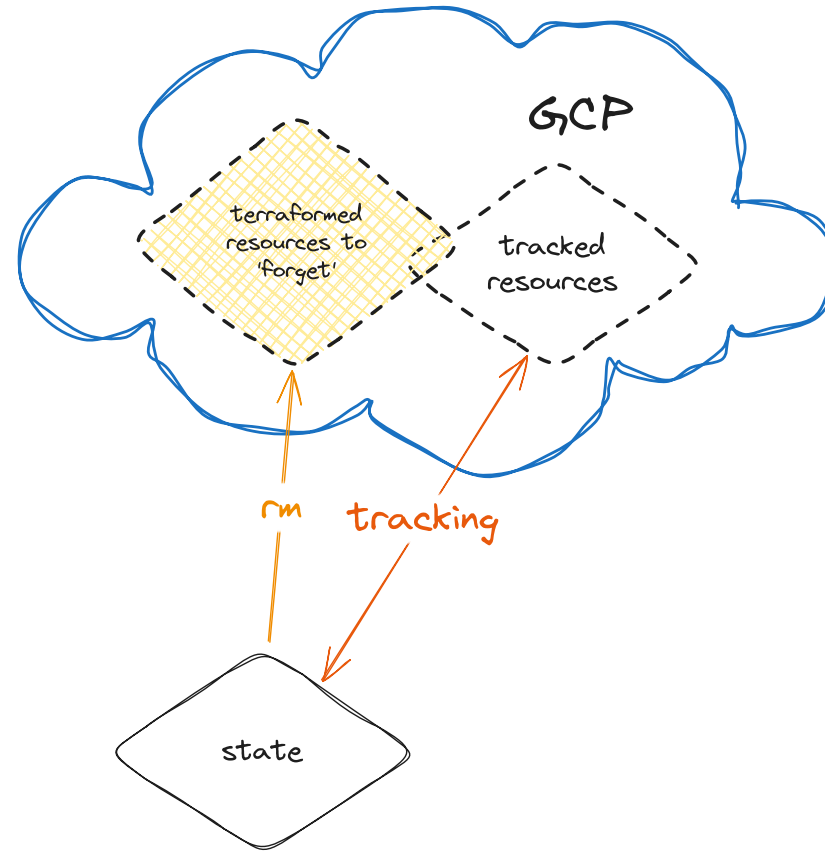
```
terraform plan
```

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.



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
```

**Infrastrukturen i amedia-adp-\***

## **amedia-adp-sources**

Her styres det meste av [amedia/terraform-selvbetjening](#).

Relevante moduler ligger i modules/amedia-adp-sources, mens instansieringen av disse ligger i projects/amedia-adp-sources.

## **amedia-adp-sources**

Her styres det meste av [amedia/terraform-selvbetjening](#).

Relevante moduler ligger i modules/amedia-adp-sources, mens instansieringen av disse ligger i projects/amedia-adp-sources.

## **amedia-adp-{prod, test}**

Denne infrastrukturen styres med en god blanding av Terraform fra [amedia/adp-infrastructure](#), ClickOps™, DataFlow og Airflow.

En stund var tilstanden i Terraform og virkeligheten ute av synk, som har medført mye ClickOps™. Dette skal nå være rettet opp i, og vi har lik struktur som i terraform-selvbetjening.

Et mål er å flytte det som ligger i adp-infrastructure over i terraform-selvbetjening.

## Ymse

- amedia-adp-dbt-\*
- amedia-adp-marts
- amedia-analytics-eu
- amedia-data-restricted

Disse har jeg ikke oversikt over.

# Annet

- terraform fmt -recursive og terraform validate

# Annet

- terraform fmt -recursive og terraform validate
- Kan jeg hoppe over ressurser om noen andre har skapt mye drift?



# 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>`).

# 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?

# 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.



# 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 reproduuserbare og bedre dokumentert.



# 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».



# 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 hvordan vi bruker Terraform i ADP sine prosjekter.

