Terraform

Getting started with Terraform

–

Vidar Waagbø

github.com/vidarw

Hva er Terraform

Verktøy for provisjonering av infrastruktur

Kodebasert - Infrastructure-as-Code

Modulbasert

- Cloud (AWS, Azure og GCP)
- Datacenter (VMware, Azure Stack)
- Software (Docker, New Relic Grafana)

Infrastructure-as-Code 

Forenkler oppsett

- Ulike miljø
- Flere regioner
- Disaster recovery

Versjonskontroll

- Historikk
- Dokumentasjon
- DevOps/CI/CD

Hvorfor Terraform? 

Plattformagnostisk

Enkel og tydelig kode

OpenSource

Modent for bruk i Azure, AWS og GCP

Ready? 

Kommandolinjen 💻

Installasjon: https://www.terraform.io/downloads.html

Tre grunnleggende kommandoer man forholder seg til i hverdagen:

Klargjør Terraform:

```
terraform init
```

Planlegg endringer:

```
terraform plan -out=plan.tfplan
```

Oppdater infrastruktur:

```
terraform apply plan.tfplan
```

Filer 🗁

`terraform init` og `terraform plan` scanner automatisk etter *.*tf*-filer i mappen hvor du kjører Terraform og man kan skrive kode på kryss og tvers i en mappe. *.*tf*-filer skrives som standard i HCL.

`terraform apply` lagrer Terraform tilstanden på infrastrukturen din i en *.*tfstate*-fil. Denne er viktig for å holde oversikt over tingenes tilstand!

Små prosjekter kan sjekke *.*tfstate* inn i versjonskontroll. Større prosjekter bør benytte et `backend` for å håndtere state.

Provider - Velg din sky

Providere er navnet på infrastrukturleverandørene i Terraform. Som regel betyr dette *AWS*, *Azure* eller *GCP*, men kan også være on-premise løsninger som *vSphere*.

`terraform init` klargjør prosjektet ditt for nye providere og må kjøres på nytt dersom man konfigurerer opp flere providere eller endrer på providerkonfigurasjonen.

```
provider "azurerm" {
  subscription_id = "xxx-xxx-xxx"
  client_id       = "xxx-xxx-xxx"
  client_secret   = "xxxxxxxxx=="
}
```

Ressurs 

`resource` definerer en ressurs som skal provisjoneres av Terraform.

Terraform håndterer avhengigheter automatisk og sørger for at ressursene opprettes i riktig rekkefølge.

```
resource "azurerm_resource_group" "app_rg" {
  name     = "app-rg"
  location = "West Europe"
}

resource "azurerm_storage_account" "app_storage" {
  name                     = "storageaccountname"
  resource_group_name      = "${azurerm_resource_group.app_rg.r
  location                 = "${azurerm_resource_group.app_rg.l
  account_tier             = "Standard"
  account_replication_type = "GRS"
}
```

Data Source 💾

`data` henter ut eksisterende infrastruktur som en *read-only* kilde. Objekter hentet inn med `data` blir ikke berørt av `terraform apply`. Når man refererer til en datakilde må man introdusere `${data.}` i selectoren.

```
data "azurerm_resource_group" "existing_rg" {
  name = "existing-rg"
}

resource "azurerm_resource_group" "new_rg" {
  name     = "production"
  location = "${data.azurerm_resource_group.existing_rg.locatic
}
```

Variabler

Inputvariabler `variable` defineres som andre elementer i *.*tf*-filene. Det er vanlig å trekke ut variabler til en fil kalt *variables.tf*.

Variabeldata lagres i *.*tfvars*-filer og lastes inn som et parameter i `terraform plan` steget `terraform plan -var-file="./myfile.tfvars"` .

Variabler kan også sendes inn gjennom kommandolinjen `terraform plan -var="app_region=West US"`

Komplett liste over forskjellige variabeltyper: https://www.terraform.io/intro/getting-started/variables.html

variables.tf:

```
variable "app_region" {
  default = "North Europe"
}
```

utvikling.tfvars:

```
app_region="West Europe"
```

main.tf:

```
resource "azurerm_resource_group" "new_rg" {
  name     = "new-rg"
  location = "${var.app_region}"
}
```

Moduler

Moduler er en samling med resources. Sikrer gjenbruk og konsistent kvalitet Kan hentes ut fra versjonskontroll eller lagres loktalt.

OBS! Dersom man trekker en lokal konfigurasjon ut i en modul vil terraform slette gamle ressurser og bytte nye, men mindre man manuelt manipulerer statefilen.

https://www.terraform.io/docs/modules/usage.html

```
module "my_module" {
  source    = "./modules/my_module"
  variable1 = "value1"
  variable2 = "value2"
}
```

Statehåndtering

Remote state gjennom backend:

https://www.terraform.io/docs/state/index.html

Forslag til oppgaver

Begynn med å provisjonere noe enkelt:
Eksperimenter med f.eks. en storage account.
Sjekk ut funksjonene `taint` og `destroy`.

Forsøk å definere infrastrukturen til ditt siste prosjekt i Terraform.
F.eks. lag en modul som inneholder alle typiske ressurser for en CMS site.
App Service, Storage, Database.

Ressurser

https://www.terraform.io/intro/getting-started/install.html
https://www.terraform.io/docs/providers/azurerm/index.html
https://github.com/vidarw/terraform-azurerm-blank