L'Art de ne plus écrire de Terraform dans vos Workflows DataOps

Google BigQuery ❤️ Terraform

ata Days Lille - 2025

> *“Make analytics flow from left to right, from Dev to Ops, as quickly as possible.”*

David O’Keeffe
Solutions Architect @ Databricks

Qui suis-je ?

[sfeir]

Samuel
Boisgontier

Senior
Data Engineer

boisgontier.s@sfeir.com

Google Cloud Certified
Professional
DATA ENGINEER

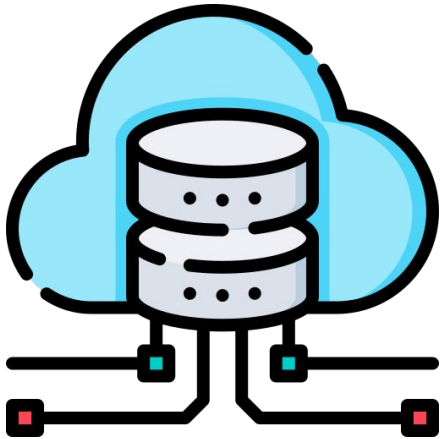
LEROY MERLIN

DATA

BigQuery

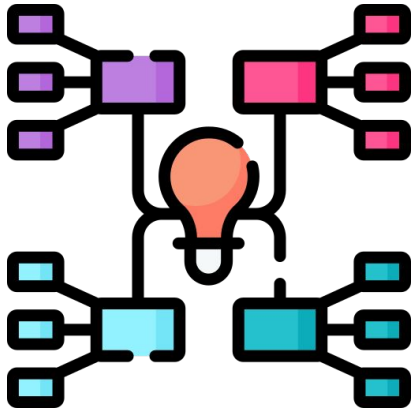
Qu’est-ce que c’est ?

- Data Warehouse
- Serverless
- Pétaoctet
- SQL



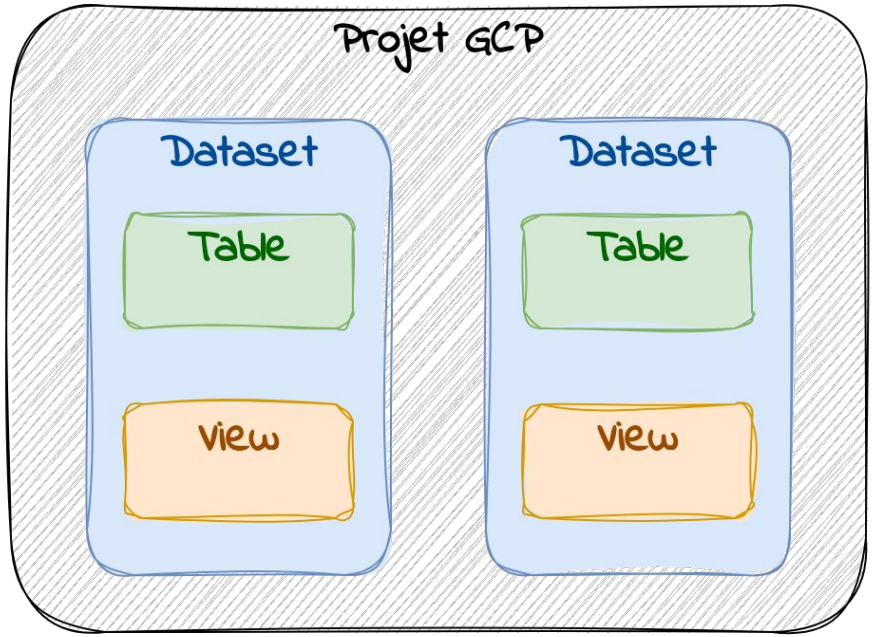
Fonctionnement

- Stockage ✂ Calcul
- Map Reduce
- 1 Colonne 〓 1 Fichier

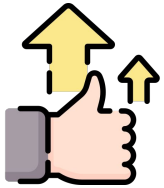


Modélisation

- Dataset
- Accès
- Alimentation

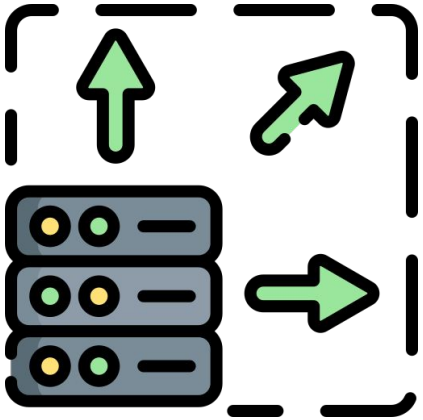


Avantages



- Scalabilité
- Performance Vs Volume
- Coût à l'usage

- Machine Learning
- Sécurité
- Connectivité



Inconvénients

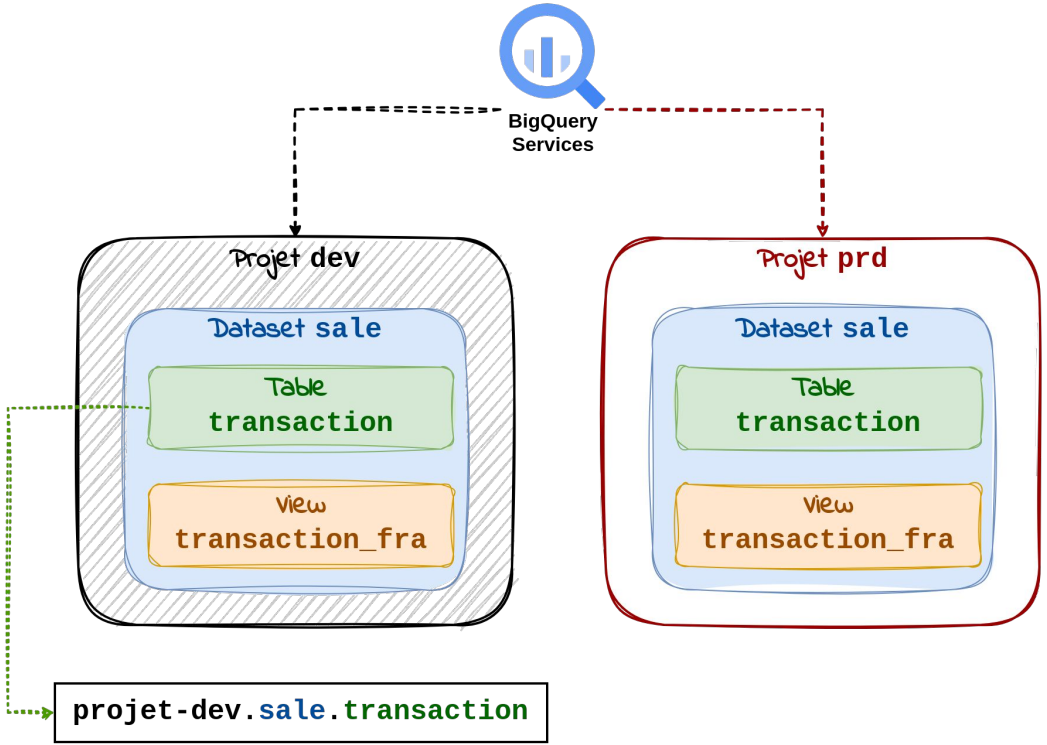
- Coût
- Vendor Lock In
- Modification en temps réel
- Optimisation complexe

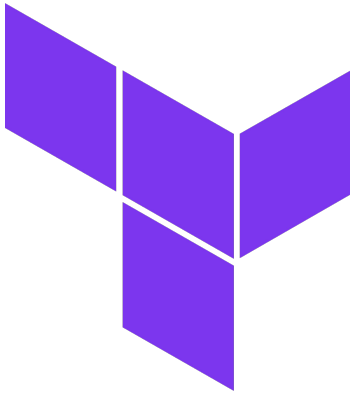
Cas d'usage

- Données de vente
- 2 Environnements

Objectif

↳Déploiement en 1 clic

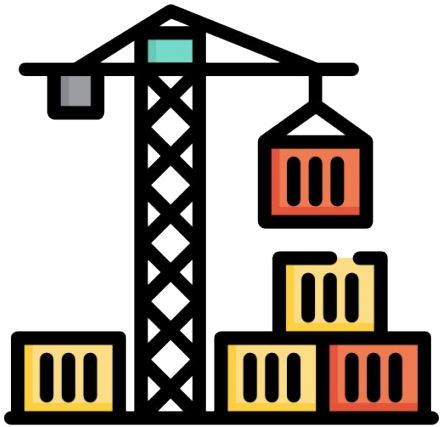




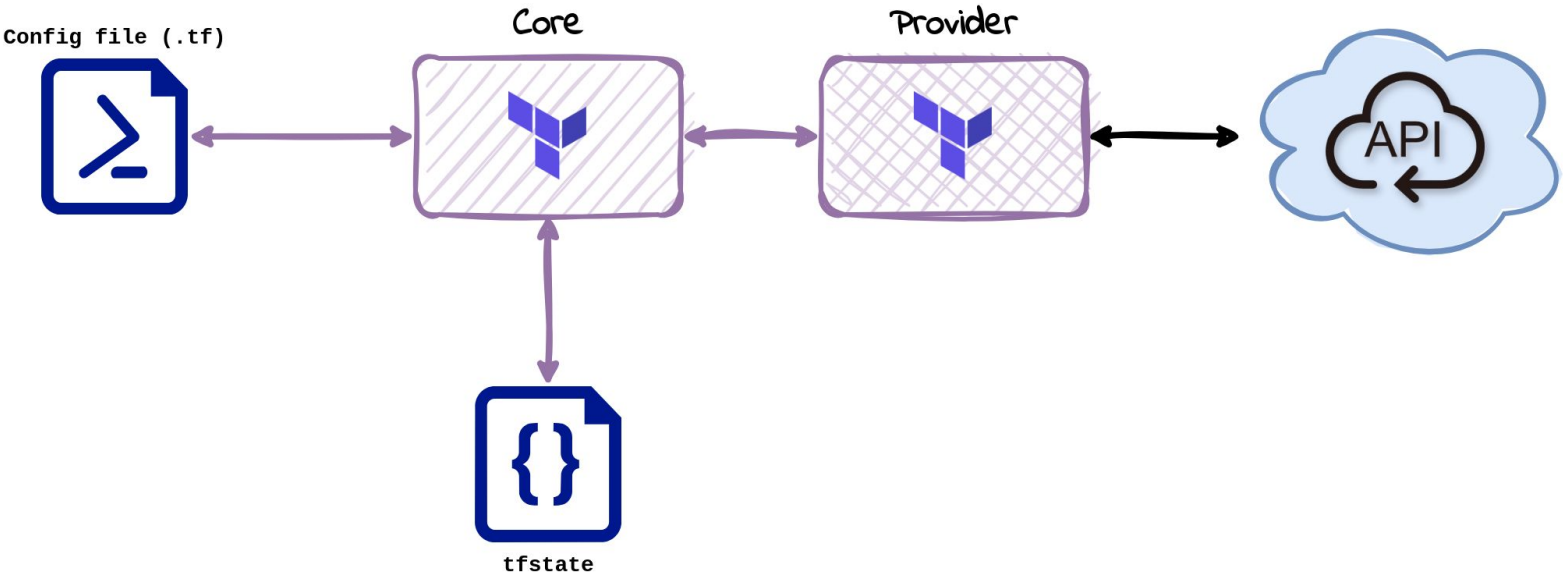
Terraform

Qu’est-ce que c’est ?

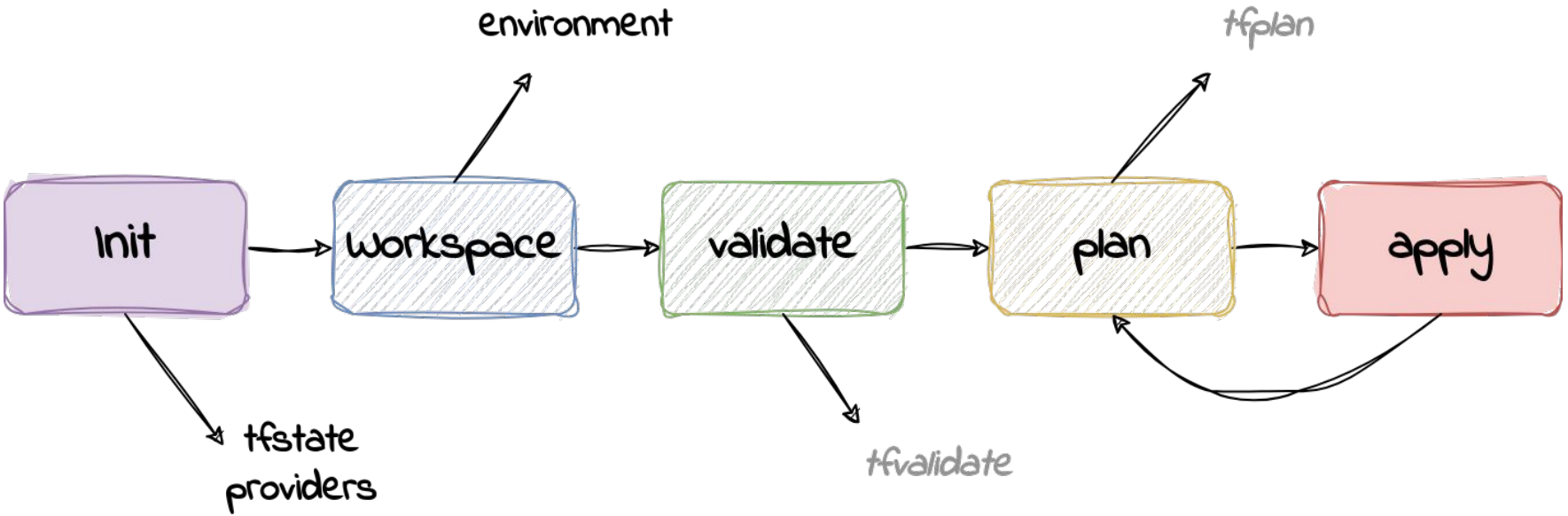
- Hashicorp
- Infrastructure as Code
- Déclaratif



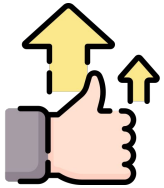
Fonctionnement



Workflow


environment
tfplan
Init
workspace
validate
plan
apply
tfstate
providers
tfvalidate

Avantages

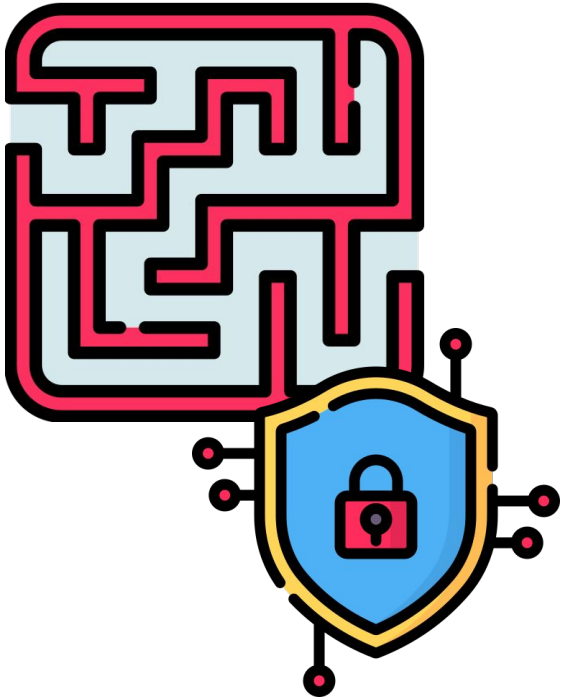


- Multi-Cloud
- Code Déclaratif
- Suivi d’état

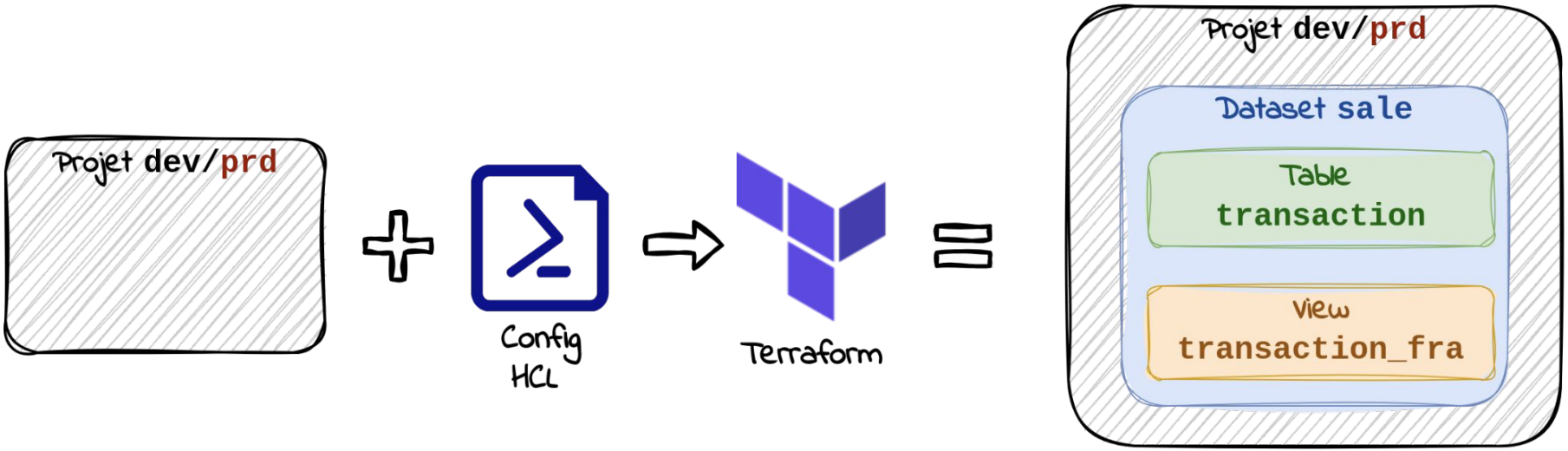
- **Modulaire**
- Réutilisable
- Standardisation

Inconvénients

- Nouveau langage (HCL)
- Dépendances aux Providers
- Gestion des secrets



Cas d'usage


Projet dev/prd
Config
HCL
Terraform
Projet dev/prd
Dataset sale
Table
transaction
View
transaction_fra

Disclaimer

- **BigQuery** n'est qu'**un exemple**
- **Stratégie** globale et générale

1 Contrainte ⇒ Terraform

Initialisation

Machine locale

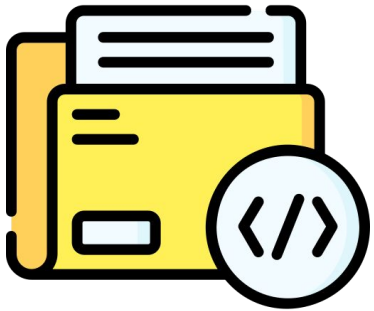
- Créer un repo Git
- Installer Terraform
- Installer Google Cloud SDK

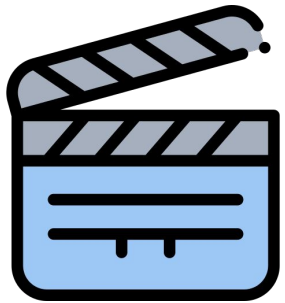
Serveur

- Nouveau projet **Google Cloud**
- Configurer **Google Cloud SDK**

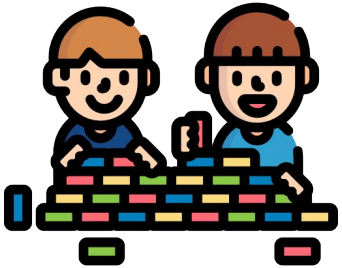
Code

- Backend **Terraform**
- Provider **Google**
- **Versions**



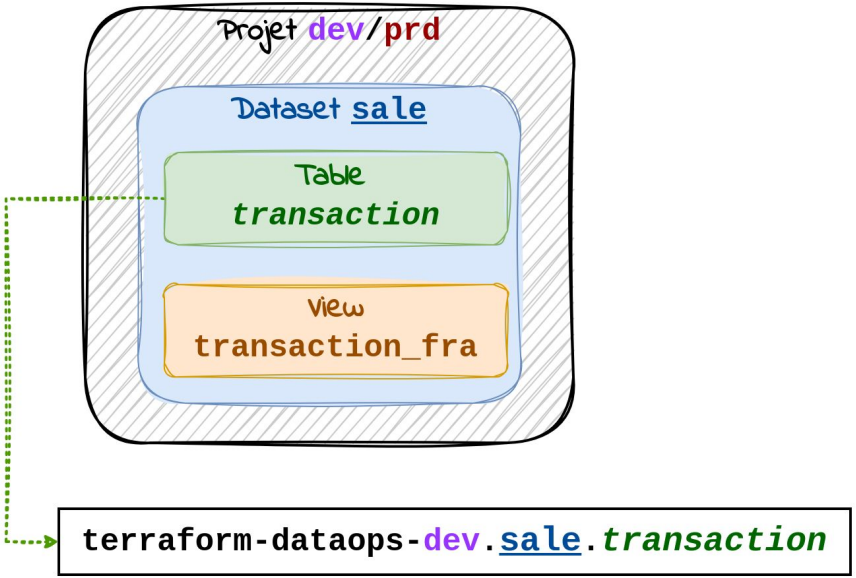


Demo



Premier Pas

Cas d'usage


Projet dev/prd
Dataset sale
Table
transaction
View
transaction_fra
terraform-dataops-dev.sale.transaction

Dataset

dataset.tf

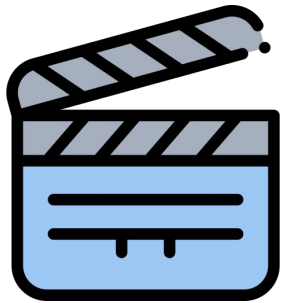
Table

```
resource "google_bigquery_table" "transaction" {

  project    = "terraform-dataops-dev"
  dataset_id = "sale"
  table_id   = "transaction"

}
```

table.tf

Clé Statique

```
google_bigquery_table.transaction
```



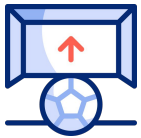
Demo

Problèmes

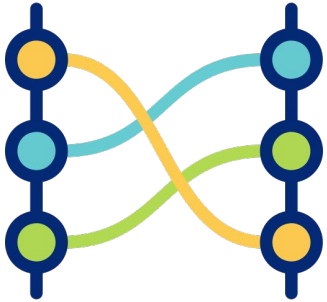
- 1 fichier ⇔ Plusieurs langages
- Projet de **prod** ?

Variable, Fichier & Référence

Objectifs



- 1 langage / fichier
- Lier les ressources
- Injecter l'environnement



Création de Variables

```
locals {
  zone = "EU"

  env = terraform.workspace
  dataops_project = "terraform-dataops-${local.env}"
}
```

locals.tf

Utilisation de Variables

```
resource "google_bigquery_dataset" "sale" {

  project    = local.dataops_project
  dataset_id = "sale"
  location   = local.zone

}
```

dataset.tf

Référence & Fichier

```
resource "google_bigquery_table" "transaction" {

  project    = local.dataops_project
  dataset_id = google_bigquery_dataset.sale.dataset_id
  table_id   = "transaction"

  schema = file("../schema/transaction.json")

}
```

table.tf

Template

```
SELECT
  *
FROM
  `terraform-dataops-${env}.sale.transaction`
WHERE
  business_unit_code = 'FRA'
```

query/transaction_fra.sql

```
resource "google_bigquery_table" "transaction_fra" {
  [...]

  view {
    query = templatefile("../query/transaction_fra.sql",
      { env = local.env }
    )
  }

}
```

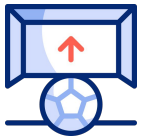
view.tf

Problèmes

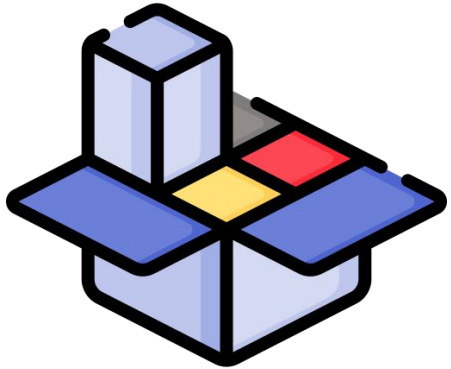
- Valeurs en dures
- Duplication de code

Dictionnaire

Objectifs



- Sortir les valeurs en dures
- Unifier dans un objet commun



Mapping

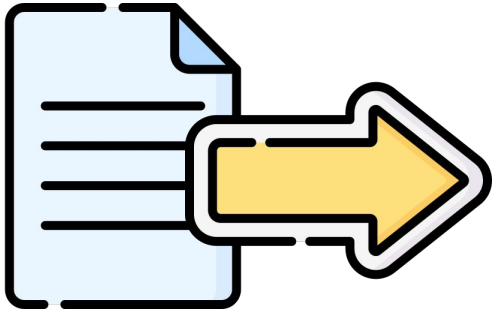
```
locals {
  dataset_sale = {
    project        = local.dataops_project
    dataset_id     = "sale"
    location       = local.zone
  }
}
```

```
resource "google_bigquery_dataset" "sale" {

 project        = local.dataset_sale.project
 dataset_id     = local.dataset_sale.dataset_id
 location       = local.dataset_sale.location

}
```

dataset.tf

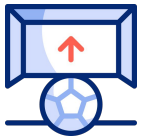
Problèmes

- Duplication de code

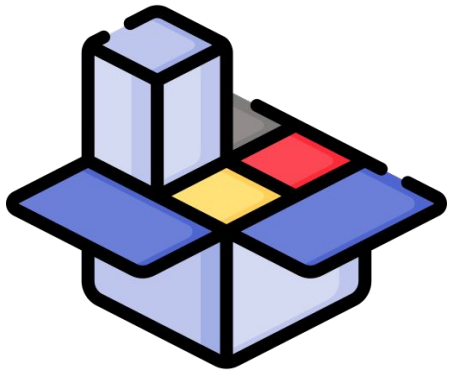


Injection de **Fichiers**

Objectifs



- **Mettre** les valeurs dans un fichier
- **Injecter** le fichier dans la ressource
- 1 type de ressource ⇐ N fichiers



Injection

```
locals {
  tables = list[ "contenu des fichiers" ]
}


resource "google_bigquery_table" "tables" {
  for_each = local.tables

  project    = each.value.project
  dataset_id = each.value.dataset_id
  table_id   = each.value.table_id
}
```

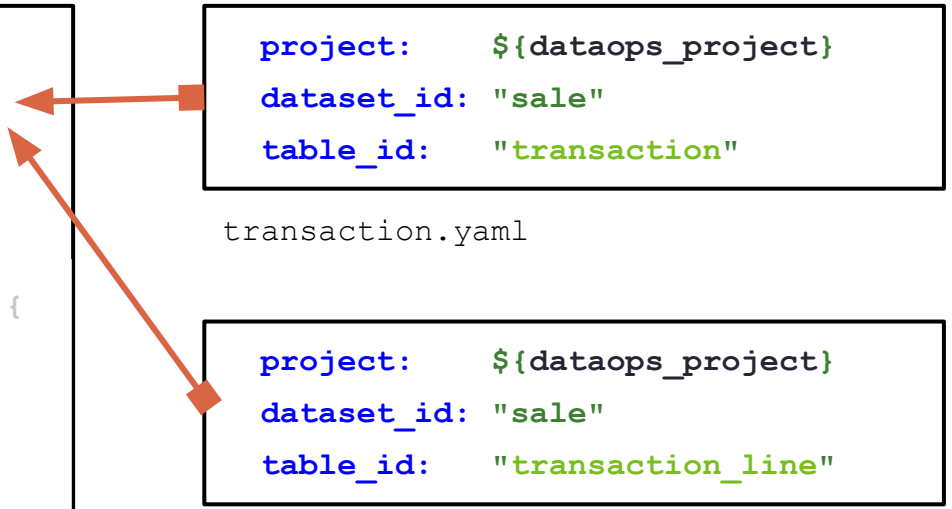
tables.tf

```
project:    ${dataops_project}
dataset_id: "sale"
table_id:   "transaction"
```

transaction.yaml

```
project:    ${dataops_project}
dataset_id: "sale"
table_id:   "transaction_line"
```

transaction_line.yaml



Boucle simplifiée

```
locals {

  tables = {
    for table_file in table_files :
      table_key => content[table_file]
  }

}
```

tables.tf

Boucle réel

```
locals {

  tables = {
    for file in fileset(".", "tables/*.yaml") :
      trimsuffix(basename(file), ".yaml")  #key
       => yamldecode(templatefile(file, local.context))
  }

}
```

tables.tf

Clé Dynamique

```
google_bigquery_table.tables['transaction']
```

```
google_bigquery_table.tables['transaction_line']
```

Référence

```
[...]

dataset_key: sale
```

transaction.yaml

```
resource "google_bigquery_table" "tables" {
  for_each = local.tables

  project    = google_bigquery_dataset.datasets[each.value.dataset_key].project
  dataset_id = google_bigquery_dataset.datasets[each.value.dataset_key].dataset_id
}
```

tables.tf

Optionalité

```
resource "google_bigquery_table" "tables" {
  for_each = local.tables

  [...]

  description         = lookup(each.value, "description", null)
  deletion_protection = lookup(each.value, "deletion_protection", true)
}
```

tables.tf

Liste

```
resource "google_bigquery_table" "tables" {
  for_each = local.tables

  [...]

  clustering = toset(each.value.clustering)
}
```

tables.tf

```
[...]

clustering:
 - business_unit_code
 - store_code
```

transaction.yaml

Dictionnaire

```
resource "google_bigquery_table" "tables" {
  for_each = local.tables

  [...]

  labels = each.value.labels
}
```

tables.tf

```
[...]

labels:
  env: ${env}
  product: retail
```

transaction.yaml

Fichier Statique

```
resource "google_bigquery_table" "tables" {
  for_each = local.tables

  [...]

  schema = file(each.value.schema_file)
}
```

tables.tf

```
[...]

schema_file: tables/schema/transaction.json
```

transaction.yaml

Fichier Dynamique

```
resource "google_bigquery_table" "views" {
  for_each = local.views

  [...]

  query = templatefile(
    each.value.query_file,
    { env = local.env }
  )
}
```

views.tf

```
[...]

query_file: views/query/transaction_fra.sql
```

transaction_fra.yaml

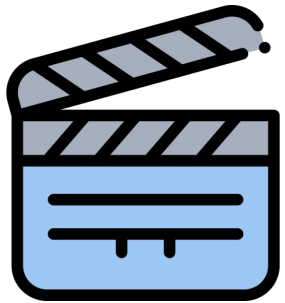
Bloc Optionnel

```
resource "google_bigquery_table" "tables" {
  for_each = local.tables

  [...]

  dynamic "time_partitioning" {
    for_each = lookup(each.value, "time_partitioning", null)
               != null
               ? [each.value.time_partitioning]
               : []
    content {
      type  = time_partitioning.value.type
      field = lookup(time_partitioning.value, "field", null)
    }

}
```

tables.tf

```
[...]

time_partitioning:
  type:  DAY
  field: date
```

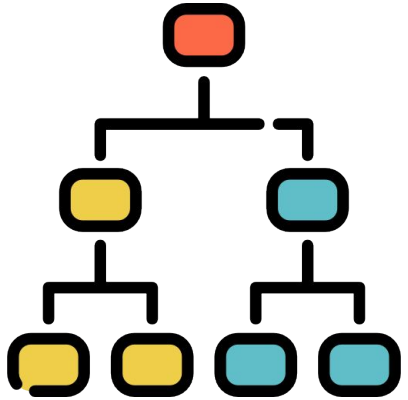
transaction.yaml



Demo

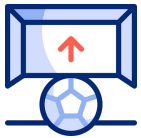
Problèmes

- Fichiers au même niveau



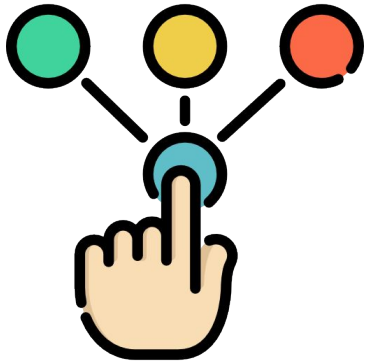
Hiérarchie

Objectifs

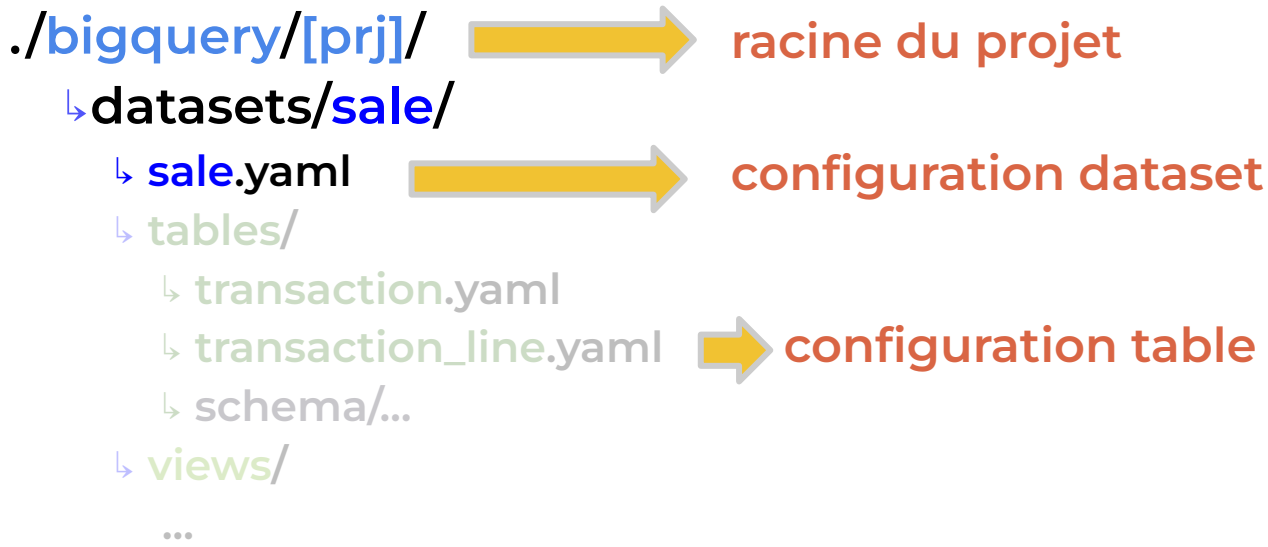


- **Hiérarchiser** les fichiers
- Générer une **clé unique**

⇒ Utiliser une **arborescence**



Arborescence



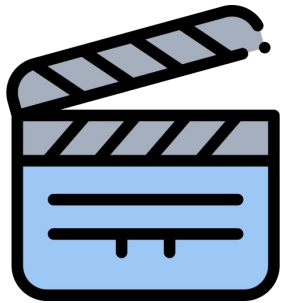
Clés

./bigquery/[prj]/
↳datasets/sale/
↳ sale.yaml → [prj]_sale
↳ tables/
↳ transaction.yaml
↳ transaction_line.yaml → [prj]_sale_transaction_line
↳ schema/...
↳ views/
...

Boucle

```
locals {

  tables = {
    for file in fileset("../bigquery", "*/datasets/*/tables/*.yaml") :
     < [prj]_sale_transaction >  # generated key
      => yamldecode(templatefile(file, local.context))
  }

}
```

tables.tf



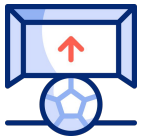
Demo

Améliorations

- Modulariser en un seul package “Bigquery”

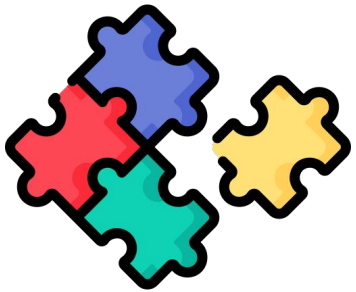
Modules

Objectifs



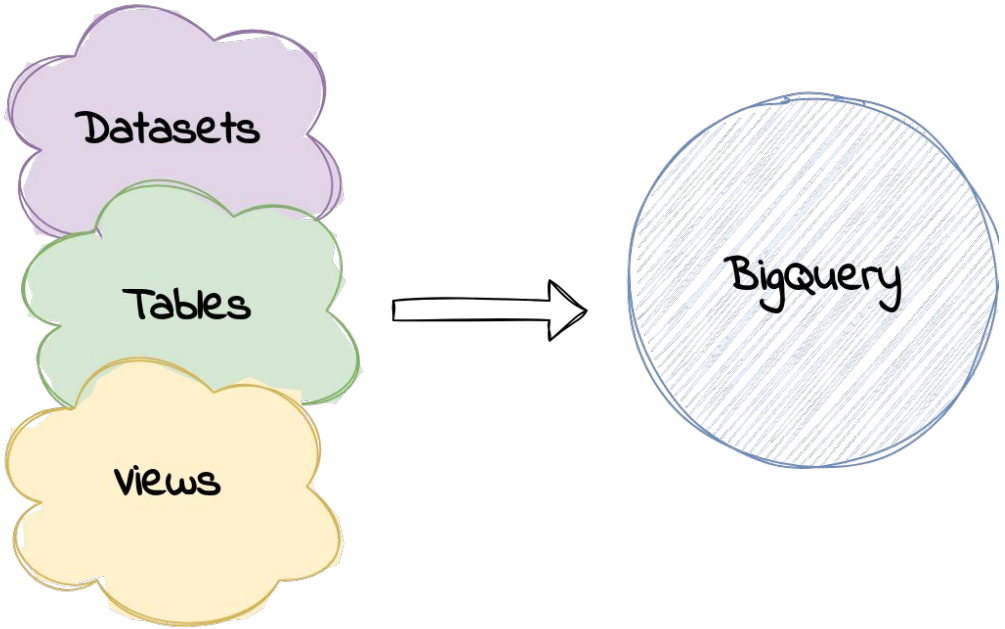
Abstraction

- concepts **réutilisables**
- **packager** des ressources
- **simplification** du code
- plug & play



⇒ **Ecrire** du code **Terraform** **une seule fois**

Abstraction



Arborescence

./terraform/
- ↳ bigquery.tf → Déclaration du module
- ↳ modules/bigquery/
  - ↳ variables.tf → Variables d'entrées
  - ↳ outputs.tf → Variables de sorties
  - ↳ provider.tf → Déclaration du provider
  - ↳ tables.tf
  - ↳ datasets.tf
  - ↳ views.tf

Contexte

```
locals {

context = {
   env              = local.env
   location         = local.zone
   dataops_project  = local.dataops_project
   exposed_project  = local.exposed_project
 }

}
```

locals.tf

Déclaration locale

```
module "bigquery" {
  source = "./modules/bigquery"

  # inputs
  configuration_folder = "../bigquery"
  context              = local.context
}
```

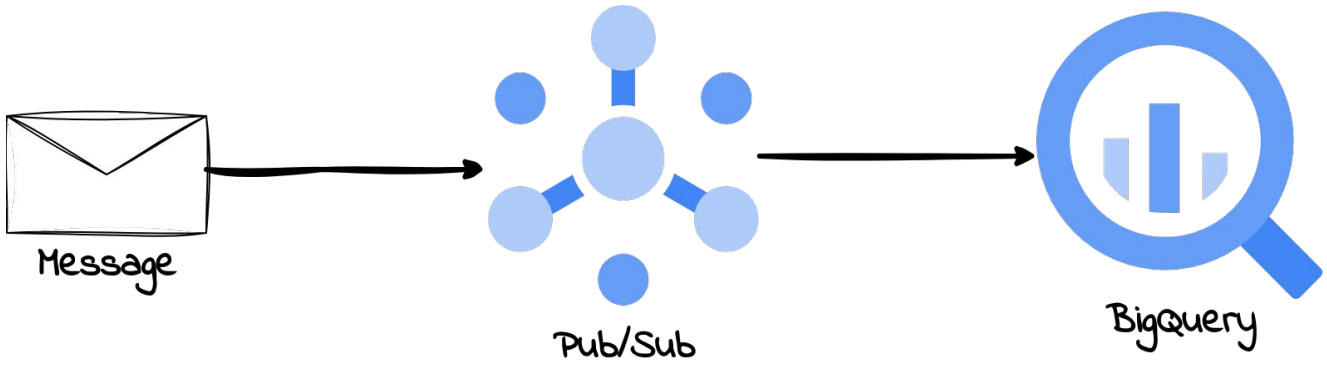
bigquery.tf

Déclaration distante

```
module "bigquery" {
  source = "git@github.com:<org>/<repo>.git//bigquery?ref=v1.2.3"

  # inputs
  configuration_folder = "../bigquery"
  context              = local.context
}
```

bigquery.tf

Référence inter-module



Référence inter-module

```hcl
variable "bigquery_tables" {
  description = "Bigquery table ressources"
  type = map(any)
}
```

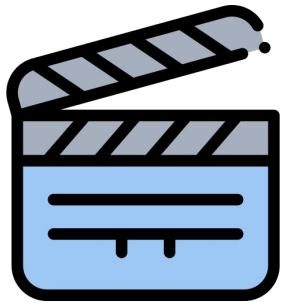
pubsub/variables.tf

```hcl
output "tables" {
  description = "Bigquery table ressources"
  value = google_bigquery_table.tables
}
```

bigquery/outputs.tf

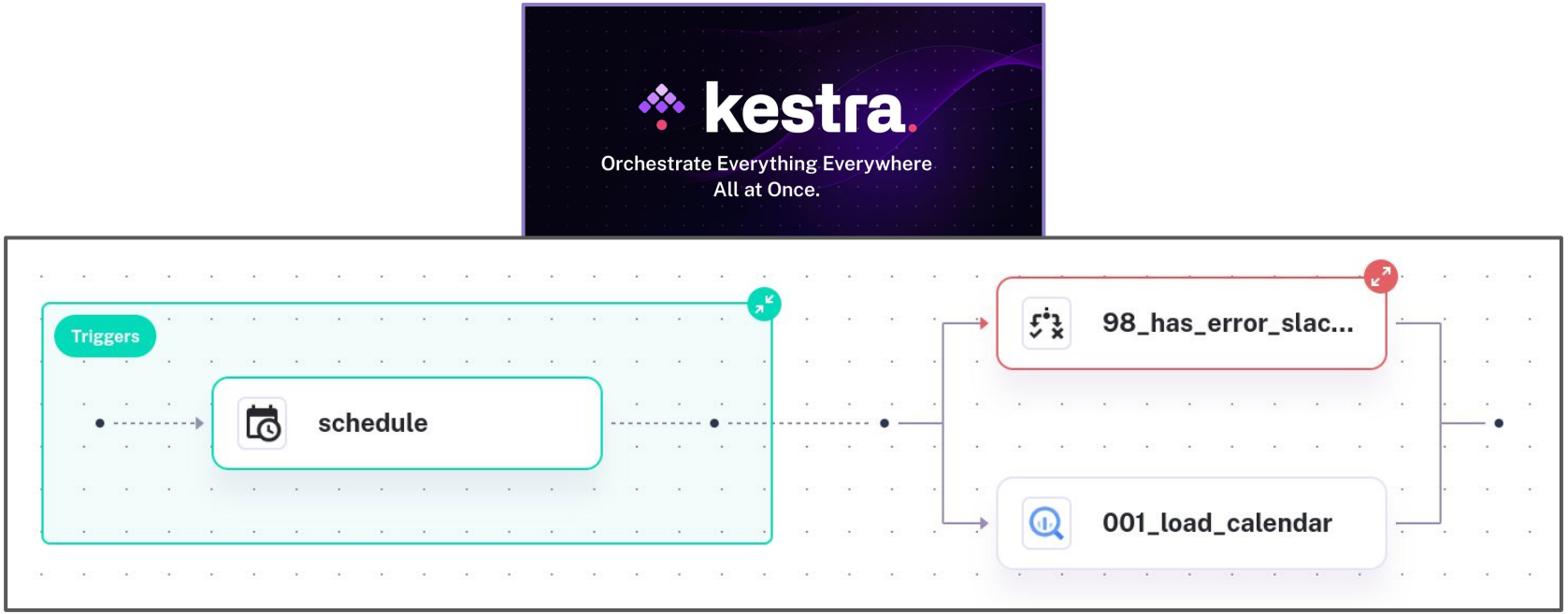
```hcl
resource "google_pubsub_subscription" "subscriptions" {
  for_each = local.subscriptions
  [...]

  dynamic "bigquery_config" {
    for_each = [...]
    content {
      table = var.bigquery_tables[bigquery_config.value.table_key].table_id
    }
  }
}
```

pubsub/subscription.tf

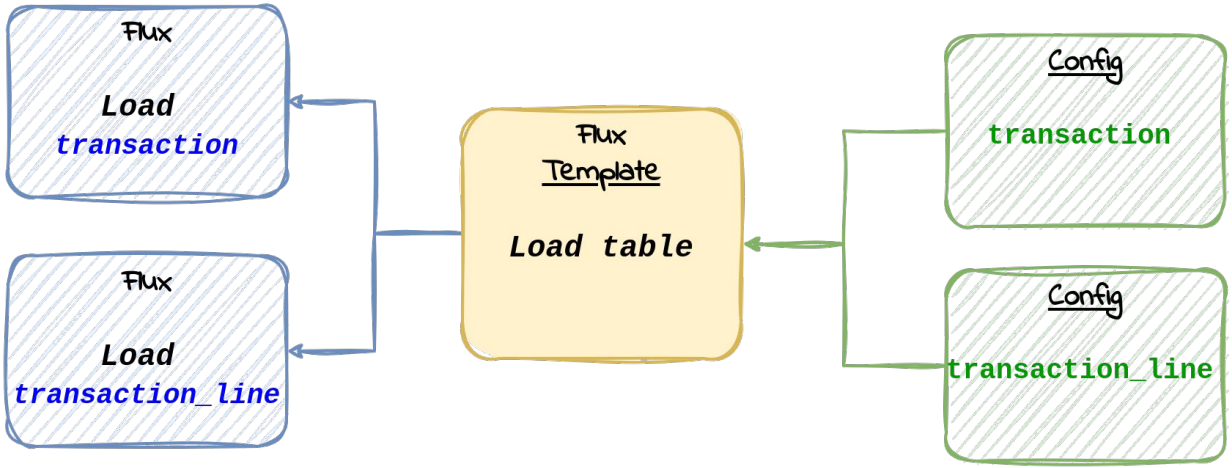


Demo

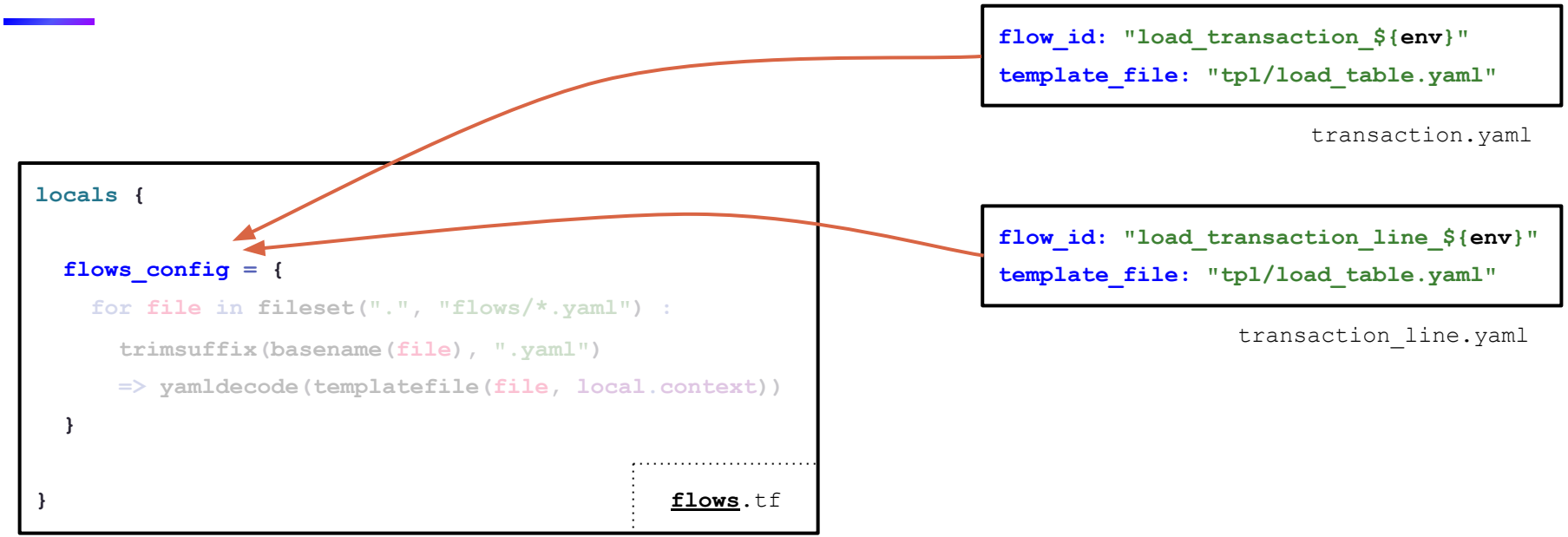
Template de Template



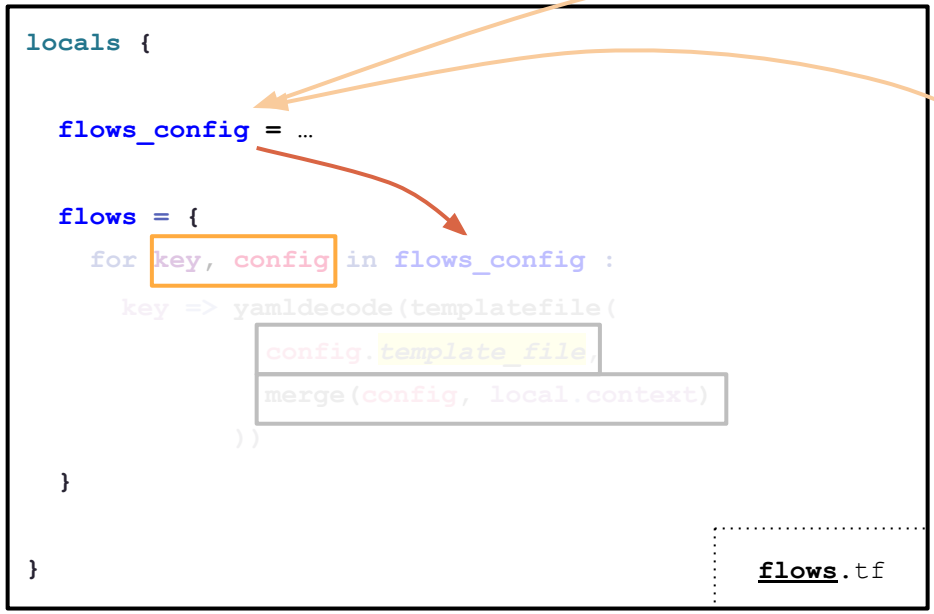
Template de Template



Template de Template



Template de Template




```yaml
flow_id: "load_transaction_${env}"
template_file: "tpl/load_table.yaml"
```

transaction.yaml

```yaml
flow_id: "load_transaction_line_${env}"
template_file: "tpl/load_table.yaml"
```

transaction_line.yaml

```yaml
id: "${flow_id}"
tasks:
  - id: "load_table_${env}"
  [...]
```

tpl/load_table.yaml

Recap

Code Terraform

- Générique
- Dynamique
- Modulaire & Abstrait
- Versionné & Lié

Merci