FOOTASYLUM

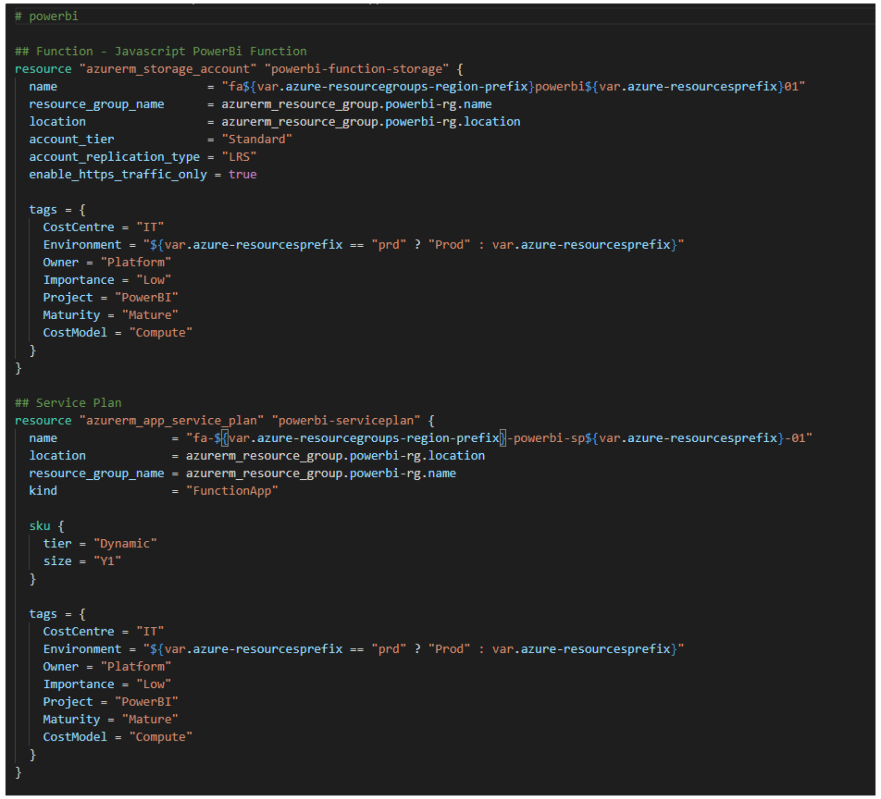
# Terraform

Monorepo vs Multirepo

## Contents

## What is Terraform?

- Open Source Tool for Creating and Maintaining Infrastructure
- Uses Declarative Definitions written in HCL (HashiCorp Configuration Language)
- Has multiple providers (Azure, AWS, Datadog etc.)



```
# powerbi

## Function - Javascript PowerBi Function
resource "azurerm_storage_account" "powerbi-function-storage" {
  name                      = "fa${var.azure-resourcegroups-region-prefix}powerbi${var.azure-resourcesprefix}01"
  resource_group_name       = azurerm_resource_group.powerbi-rg.name
  location                  = azurerm_resource_group.powerbi-rg.location
  account_tier              = "Standard"
  account_replication_type = "LRS"
  enable_https_traffic_only = true

  tags = {
    CostCentre = "IT"
    Environment = "${var.azure-resourcesprefix == "prd" ? "Prod" : var.azure-resourcesprefix}"
    Owner = "Platform"
    Importance = "Low"
    Project = "PowerBI"
    Maturity = "Mature"
    CostModel = "Compute"
  }
}

## Service Plan
resource "azurerm_app_service_plan" "powerbi-serviceplan" {
  name                = "fa-${var.azure-resourcegroups-region-prefix}-powerbi-sp${var.azure-resourcesprefix}-01"
  location            = azurerm_resource_group.powerbi-rg.location
  resource_group_name = azurerm_resource_group.powerbi-rg.name
  kind                = "FunctionApp"

  sku {
    tier = "Dynamic"
    size = "Y1"
  }

  tags = {
    CostCentre = "IT"
    Environment = "${var.azure-resourcesprefix == "prd" ? "Prod" : var.azure-resourcesprefix}"
    Owner = "Platform"
    Importance = "Low"
    Project = "PowerBI"
    Maturity = "Mature"
    CostModel = "Compute"
  }
}
```

FOOTASYLUM

## History of Terraform at Footasylum

- Introduced by Platform team
- Was a single Monorepo with small number of services
- Fast become difficult to manage
- Modules helped separate concerns but Platform was still a “gatekeeper”
- Teams now empowered to write their own terraform files
- Working towards standardizing the approach

## Monorepo

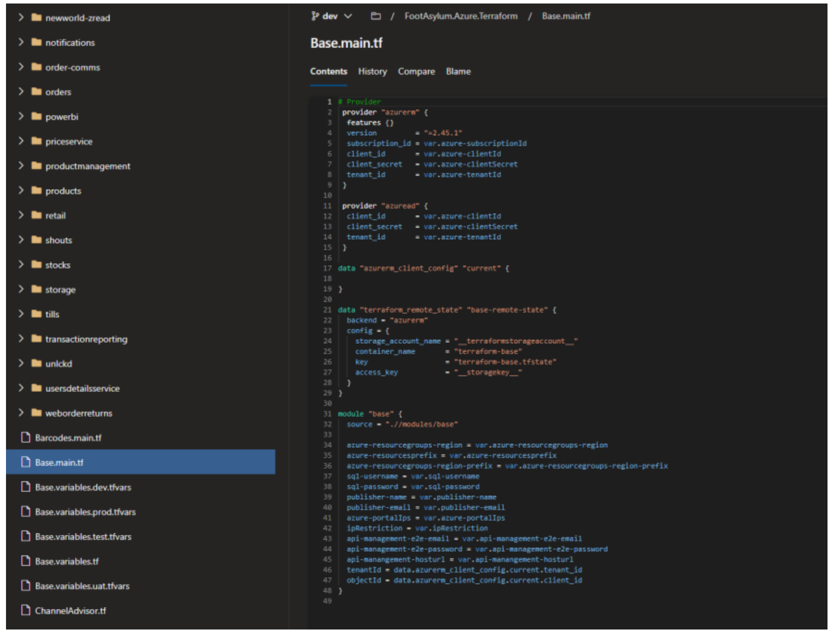
A Single repository with all the files needed to build the infrastructure.

PROS:

- Single source of truth
- Easier to assert quality control

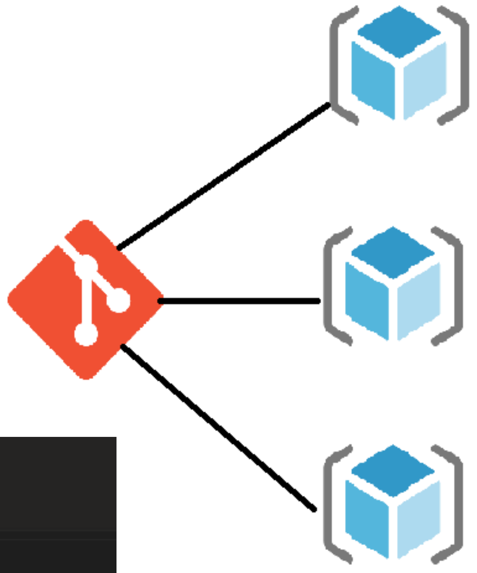
CONS:

- Can block other teams
- Longer build and release lifecycle



Base.main.tf

```
# Provider
provider "azurerm" {
 features {}
 version         = "=2.45.1"
 subscription_id = var.azure-subscriptionId
 client_id       = var.azure-clientId
 client_secret   = var.azure-clientSecret
 tenant_id       = var.azure-tenantId
}

provider "azuread" {
 client_id       = var.azure-clientId
 client_secret   = var.azure-clientSecret
 tenant_id       = var.azure-tenantId
}

data "azurerm_client_config" "current" {

}

data "terraform_remote_state" "base-remote-state" {
  backend = "azurerm"
  config = {
    storage_account_name = "__terraformstorageaccount__"
    container_name       = "terraform-base"
    key                  = "terraform-base.tfstate"
    access_key           = "__storagekey__"
  }
}

module "base" {
  source = ".//modules/base"

  azure-resourcegroups-region = var.azure-resourcegroups-region
  azure-resourcesprefix = var.azure-resourcesprefix
  azure-resourcegroups-region-prefix = var.azure-resourcegroups-region-prefix
  sql-username = var.sql-username
  sql-password = var.sql-password
  publisher-name = var.publisher-name
  publisher-email = var.publisher-email
  azure-portalIps = var.azure-portalIps
  ipRestriction = var.ipRestriction
  api-management-e2e-email = var.api-management-e2e-email
  api-management-e2e-password = var.api-management-e2e-password
  api-manangement-hosturl = var.api-manangement-hosturl
  tenantId = data.azurerm_client_config.current.tenant_id
  objectId = data.azurerm_client_config.current.client_id
}
```



## Multirepo

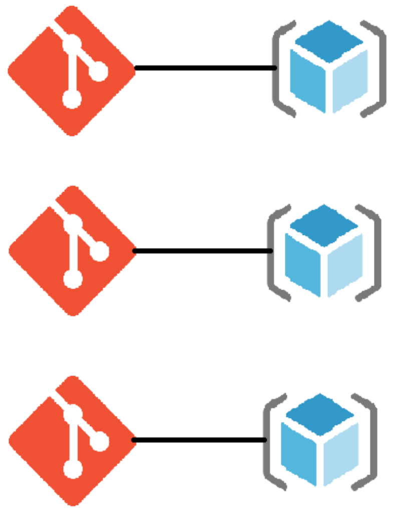
Multiple independent repositories containing the infrastructure just for that service / product

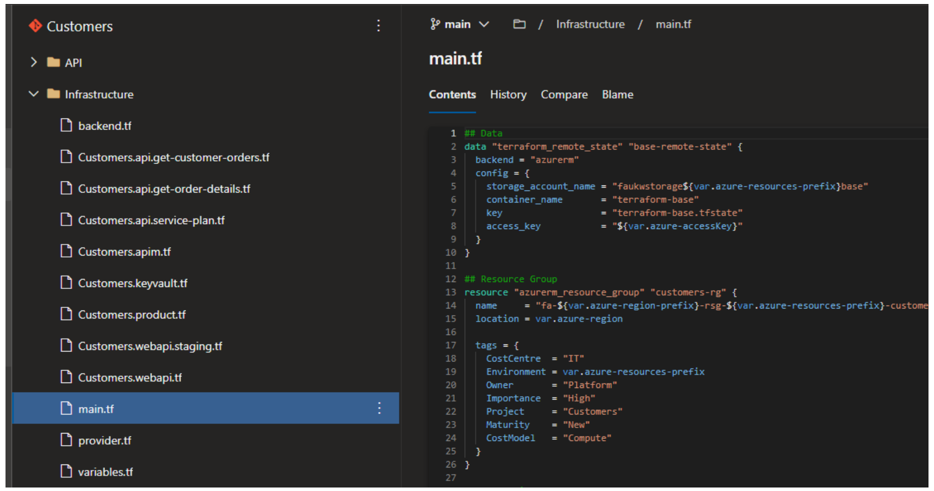
PROS:

- Empowers/Enables teams to achieve
- Removes inter-team dependencies
- Faster build times

CONS:

- Differences configuration / design
- Harder to link dependencies between separate resources





```
## Data
data "terraform_remote_state" "base-remote-state" {
  backend = "azurerm"
  config = {
    storage_account_name = "faukwstorage${var.azure-resources-prefix}base"
    container_name       = "terraform-base"
    key                  = "terraform-base.tfstate"
    access_key           = "${var.azure-accessKey}"
  }
}

## Resource Group
resource "azurerm_resource_group" "customers-rg" {
  name     = "fa-${var.azure-region-prefix}-rsg-${var.azure-resources-prefix}-custome
  location = var.azure-region

  tags = {
    CostCentre  = "IT"
    Environment = var.azure-resources-prefix
    Owner       = "Platform"
    Importance  = "High"
    Project     = "Customers"
    Maturity    = "New"
    CostModel   = "Compute"
  }
}
```

FOOTASYLUM

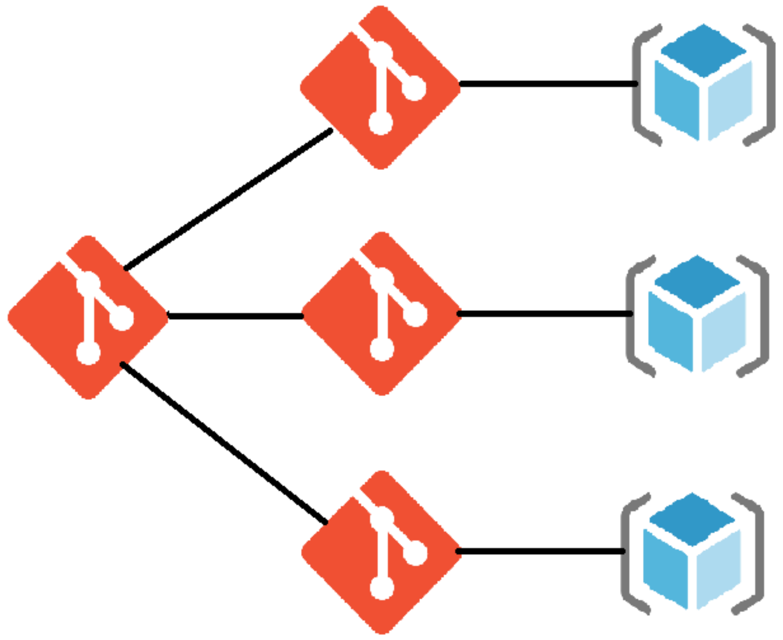
## The future – A blend of the two

Mono repo for shared / consumable modules:

- Like an internal Open Source library for infrastructure
- Modules can be constructed with the approved set up in place
- Everyone can benefit from and contribute to the "baseline"

Multi repo for teams:

- Empowers them to take ownership of their infrastructure
- Standardised modules removes cognitive load from the resources
- Improves or better aligns the over all infrastructure



FOOTASYLUM