Terraform Backend

Terraform

Agenda

- Manage Terraform State
- Terraform Backend
- Azure Backend

Manage Terraform State

Terraform

Terraform State

- **Terraform State** is a critical concept in Terraform
- Represents a **snapshot of your infrastructure** at a specific point in time
- Terraform uses this state file to **keep track of resources** you've deployed to your providers
- With it ensures that it knows the current state of your infrastructure
- Terraform State if commonly named as **tfstate**

Terraform State: Work Collaboratively

- Until now, every time you had run your Terraform code you created a local **tfstate**
- For testing purpose this is enough but when working in production and with a team is not possible to keep this strategy
- You need to store your **tfstate** on a centralized and secure place to not lose track of your resources
- This is one of the most important task you need to plan, design and implement before starting using Terraform

Terraform Backend

Terraform

Terraform Backend

- Terraform Backend is the solution to define where you want to store your state
- By default, Terraform uses the local backend
- For your projects you should use a cloud-based Backend
- You can configure your Backend using HCP Terraform (Hashicorp Cloud Platform) or using a storage related resource in any of the traditional cloud providers: Azure, AWS and GCP
- Additionally, you can use other type of backends like Kubernetes

Terraform Backend: How to create?

- Knowing that you always need a **tfstate** to manage Terraform resources, you need to create the place where you want to store it before
- The backend configuration is done on a **backend** block inside the **terraform** block
- There are usually 3 strategies: Manual, using other IaC technology or using local backend and manual upload

Terraform Backend: Auth

- Backends store state in a remote service, which allows multiple people to access it
- Accessing remote state generally requires access credentials, since state data contains extremely sensitive information
- Recommend using environment variables to supply credentials and other sensitive data
- If you use **-backend-config** or hardcode these values directly in your configuration, Terraform will include these values in both the .terraform subdirectory and in plan files
- Ideally, you should select a process with no sensitive data needed

How to create? Manually

- This is the more traditional way
- You can use web portal to do it
- If you want to automatize some steps you can use cloud provider CLI
- This approach allow you to create a script that can manage the creation of the resources
- Pros
  - Easy and faster to implement
- Cons
  - Not repeatable
  - Error prone
  - Security concerns

How to create? Other Tech

- All cloud providers have their own IaC technology
- Neither of them uses the concept of a state managed on an external file
- You can create the code to manage needed resources and execute them being part of your pipeline
- Pros
  - Using IaC approach
  - Repeatable
  - Fully automatized
- Cons
  - Need to learn another technologies
  - Can be more trickier to maintain.

How to create? Local and then upload

- You may execute your code to generate the state locally (can be on an CI/CD agent)
- Then you can upload the state to previously created storage resources
- Next time you already have your state centralized so you can follow same Terraform process
- Pros
  - Total usage of Terraform
  - Usage of same processes
- Cons
  - Two steps process
  - Need a local execution

Azure Backend

Terraform

Azure Backend

- Stores the state as a Blob with the given Key within the Blob Container within the Blob Storage Account
- This backend supports state locking and consistency checking with Azure Blob Storage native capabilities

Azure Backend: Authentication

- The azurerm backend supports 3 methods of authenticating to the storage account:
  - Access Key (default)
  - Azure Active Directory
  - SAS Token
- The Access Key method can be used directly, by specifying the access key, or in combination with an Azure AD principal (e.g. user, service principal or managed identity)
- The Azure Active Directory method can only be used in combination with an Azure AD principal. To use the Azure Active Directory method you must set the **use_azuread_auth** variable to true in your backend configuration.
- The SAS Token method can only be used directly. You must generate a SAS Token for your state file blob and pass it to the backend config.

Authentication: Preferable Auth Mechanism

- Service Principal or User Assigned Managed Identity via OIDC with Azure AD (Workload identity federation)
  - **use_azuread_auth = true, use_oidc = true**
- Service Principal or User Assigned Managed Identity via OIDCwith Access Key (Workload identity federation)
  - **use_oidc = true**
- Managed Identity Principal
  - **use_azuread_auth = true, use_msi = true**

Azurerm Backend Example

```
terraform {
  backend "azurerm" {
    resource_group_name  = "StorageAccount-ResourceGroup"
    storage_account_name = "abcd1234"
    container_name       = "tfstate"
    key                  = "prod.terraform.tfstate"
    use_oidc             = true
    client_id            = "00000000-0000-0000-0000-000000000000"
    subscription_id      = "00000000-0000-0000-0000-000000000000"
    tenant_id            = "00000000-0000-0000-0000-000000000000"
    use_azuread_auth     = true
  }
}
```

Demo – Using Azure Backend

Terraform

Lab 03 – Store your state in Azure

Terraform