Managing Resources Using Terraform

Document Level Classification

200

Introduction

We are migrating TFE workload to HCP Terraform, it is a work in process; in the meantime you can continue using TFE until you get notified that your Organization has been migrated into a HCP Project. Questions/ Concerns please reach out in #cloud-ops-support-ext

HashiCorp Terraform is an infrastructure as code (IaC) tool that lets you define cloud resources in human-readable configuration files that you can version, reuse, and share. Terraform uses providers to allow it to work

with multiple platforms or services. You can use terraform to provision and configure resources outside of your Substrate cluster. For example, AWS resources (Amazon RDS, Security Groups, etc.).

Typically, terraform code lives alongside your application and associated infrastructure configurations in version control. To learn more about terraform and how to use it, use the tutorials.

Using terraform with Amazon S3 and DynamoDB backends

Your Substrate infrastructure includes an Amazon S3 bucket (<AWS_ACCOUNT_ID>-tfstate) and DynamoDB table (<AWS_ACCOUNT_ID>-tfstate) that you can use as a backend for storing terraform remote state. For example:

Example: terraform backend configuration

Given the following terraform configuration file (`main.tf`), with the "<>" placeholders updated:

```
provider "aws" {
  region = "us-east-1"
}

terraform {
  backend "s3" {
    bucket         = "<AWS_ACCOUNT_ID>-tfstate"
    region         = "us-east-1"
    key            = "<SUBSTRATE_CLUSTER>/<SUBSTRATE_NAMESPACE>/<APPLIC
    dynamodb_table = "<AWS_ACCOUNT_ID>-tfstate"
  }
}

data "aws_caller_identity" "this" {}
data "aws_region" "this" {}

output "account_id" {
```

```
  description = "Current AWS Account ID"
  value = data.aws_caller_identity.this.account_id
}

output "region" {
  description = "Current AWS Region"
  value = data.aws_region.this.name
}
```

You can run the following commands in an authenticated terminal session to store and manage terraform state in your Substrate account:

```
# Verify your credentials
aws sts get-caller-identity

# Initialize
terraform init

# Apply changes. This should prompt you for confirmation, then display
terraform apply
```

Considerations when using an Amazon S3 backend:

- DO use a unique *backend key* for each application.
- DO use the *default* workspace, as opposed to using a `terraform workspace <name>`.
- DO use a different root-module per-environment. For example, your version control repository may contain a *dev* root-module in `src/terraform/environments/dev`, a *live* root-module in `src/terraform/environments/live`, and child-modules with reusable code in `src/terraform/modules/`.
- DO reference secrets (Amazon RDS passwords, etc.) using the Vault provider. Storing these secrets in terraform state is not secure.

Using HCP Terraform - Migration in Progress

EPIC is moving the TFE workload to HCP Terraform, it provides essentially the same features but has some differences to be considered. The migration has just started and it is planned to be completed before the end of this 2024. Please refer to the HCP user guide and the Data Migration Tracking page to get additional details about the migration. Any question related please ask in #cloud-ops-support-ext.

Using Terraform Enterprise (TFE) - On Deprecation Path

TFE provides additional capabilities (audit logging, remote state, policy-as-code, modules hosting, pull-request integration, etc.) when compared to using terraform (open-source) with Amazon S3 backend. When your Substrate infrastructure is provisioned, your team is also provided with an *Organization* within Epic Games' Terraform Enterprise. TFE is on deprecation path so at the beginning of 2025 you should be using HCP Terraform instead.

For more information, or to get started with TFE, review the TFE User-Guide.