Managing GitHub with Terraform

...

Who am I?

What is Terraform?

Developer / Terraform / Intro

https://www.terraform.io/

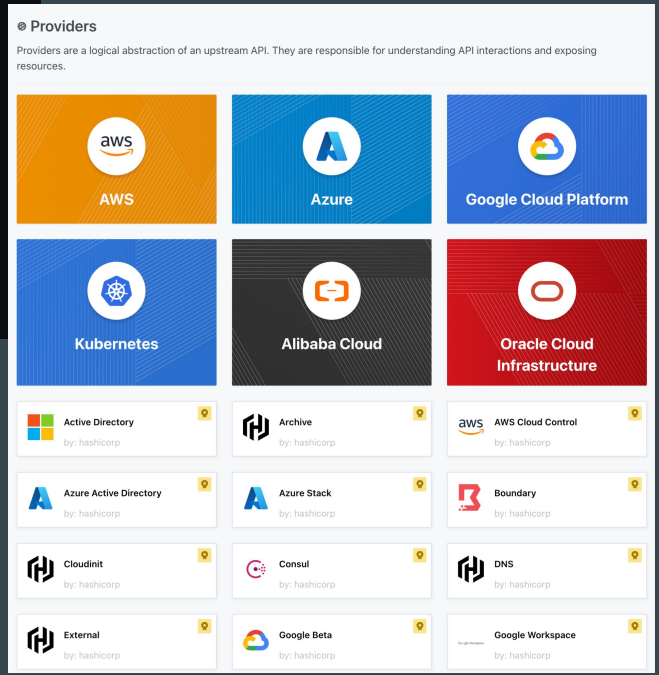
What is Terraform?

v1.5.x (latest)

Terraform is an infrastructure as code tool that lets you build, change, and version cloud and on-prem resources safely and efficiently.

Terraform manages resources through providers
https://registry.terraform.io/browse/providers

HashiCorp Terraform is an infrastructure as code tool that lets you define both cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share. You can then use a consistent workflow to provision and manage all of your infrastructure throughout its lifecycle. Terraform can manage low-level components like compute, storage, and networking resources, as well as high-level components like DNS entries and SaaS features.



Terraform Resources

- Resources are the most important element in the Terraform language.
- Each resource block describes one or more infrastructure objects
  - compute instances
  - virtual networks
  - DNS records.

```
resource "aws_instance" "web" {
  ami           = "ami-a1b2c3d4"
  instance_type = "t2.micro"
}
```

Create an ec2 instance

https://developer.hashicorp.com/terraform/language/resources/syntax

Terraform State

- All terraform provisioned resources are stored in a state file.
- The state file can can be stored in different places, configured as a backend. Some examples include;
  - local file (all in terraform.tfstate locally)
  - S3 backed by dynamodb for state locks
- State files need to be treated as secure as they contain confidential information about your resources.
- As working with terraform requires access to the state file, storing it in a common, secure location is necessary for teams.

Example Configuration

Copy

```
terraform {
  backend "s3" {
    bucket = "mybucket"
    key    = "path/to/my/key"
    region = "us-east-1"
  }
}
```

bucket = s3 bucket name to store the state

key = path in s3 to the state file

region = the region the bucket exists in

dynamodb_table = The table to store the state locks (missing in the above example), doc here

https://developer.hashicorp.com/terraform/language/settings/backends/configuration

Include the Provider

- Terraform providers need to be required..
- Running terraform init with pull the required providers.
  - terraform init also manages provider updates
  - You can specify specific versions using the version constraints
- It's possible for the provider to accept configuration options via parameters.
  - With GitHub it's possible to override the token with a variable.

```
terraform {
  required_providers {
    github = {
      source  = "integrations/github"
      version = "~> 5.0"
    }
  }
}

# Configure the GitHub Provider
provider "github" {}
```

```
provider "github" {
  token = var.token # or `GITHUB_TOKEN`
}
```

https://registry.terraform.io/providers/integrations/github/latest/docs

GitHub Provider

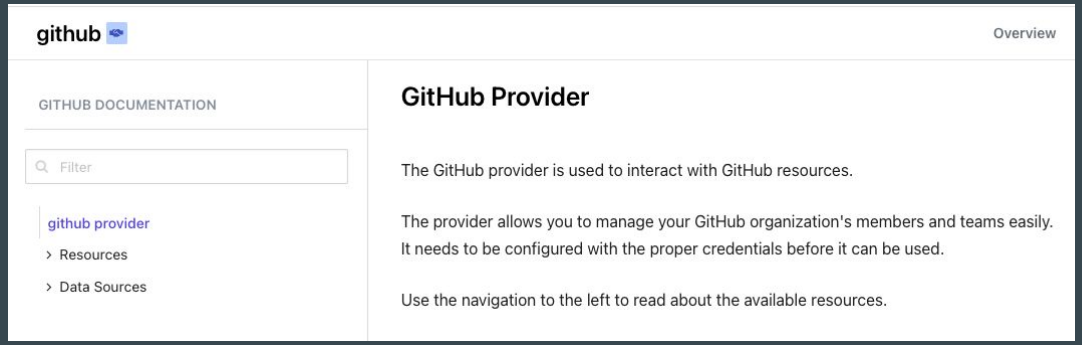
...

Some benefits of managing GitHub with Terraform

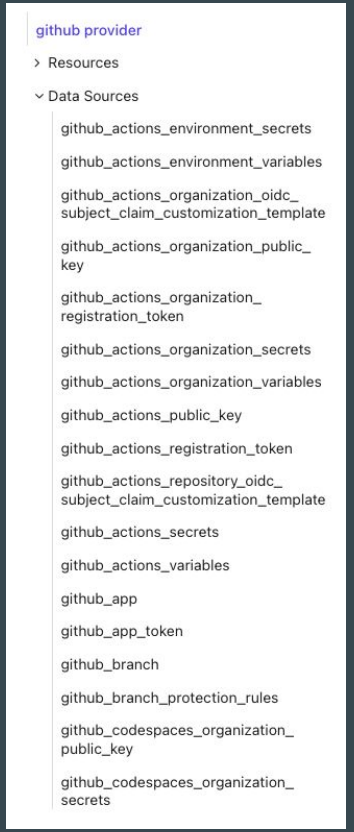
- Automation and Consistency
  - The Terraform code is stored in GitHub and deployed by conventional CI/CD methods.
  - This consistency helps to maintain a reliable and predictable development environment.
- Simplified Resource Management
  - Terraform's declarative language abstracts the complexities of interacting with GitHub's API.
  - This abstraction simplifies the management of repositories, teams and members
- Versioning and Auditing
  - As Terraform code is version-controlled (i.e. stored in GitHub), it facilitates easy tracking of changes made to GitHub resources.
  - This audit trail allows teams to understand who made changes, when they were made, and why they were made, improving transparency and accountability.
- Scalability and Reusability
  - As infrastructure needs grow, Terraform enables the simple scaling of GitHub resources.
  - Terraform goes further and modules allow the creation of reusable and parameterised configurations which reduces duplication of effort and provides some consistency across projects.
- Safe Changes and Rollbacks
  - Terraform's plan and apply workflow lets you preview changes before applying them to GitHub.
  - This feature allows you to assess potential impacts and catch errors before they affect the environment.
  - In case of issues, you can roll back changes to a known working state easily.
- Infrastructure as Code (IaC) Approach
  - The above points are really points that sum up IaC.
  - With Terraform, GitHub resources are defined as code, allowing for version control, collaboration, and consistency across environments.
  - IaC promotes better documentation, repeatability, and reduces the risk of manual configuration errors.

Getting around

- The [GitHub Provider docs]

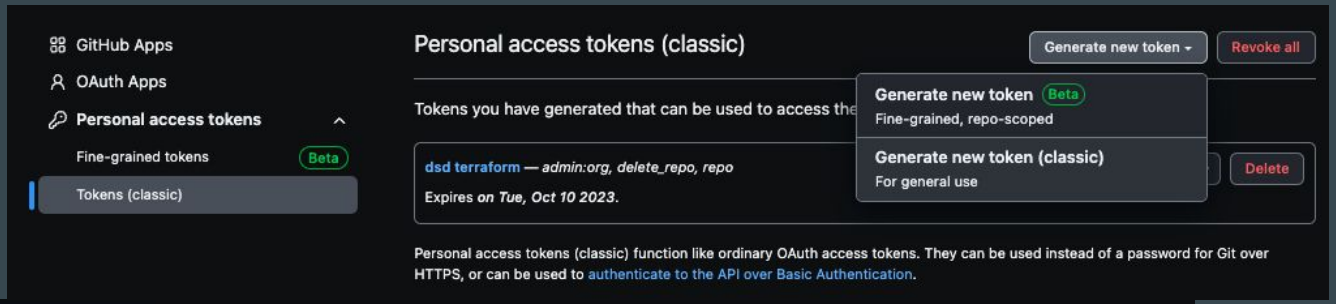


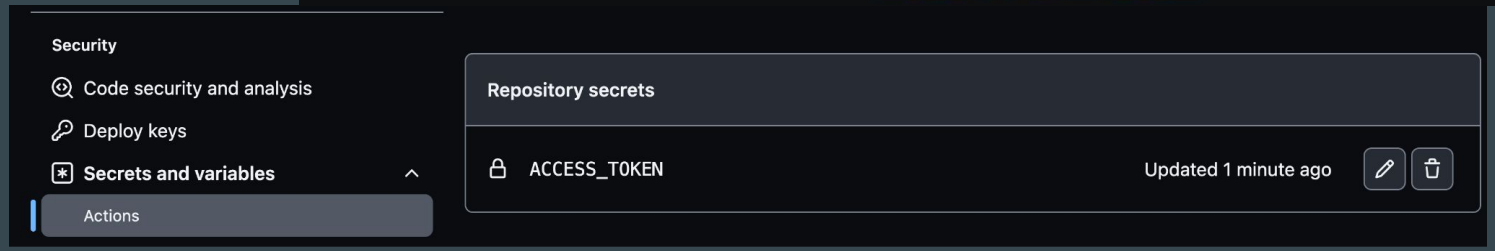
- Documentation is maintained by the provider
- Pretty well maintained



The Setup

- As a service account;
- Generate a classic token
- Stored the token in the repository secrets in GitHub





Code Segue

Some Enterprise

Benefits of SSO for GitHub

- Centralised Authentication
- Centralised user provisioning
- Immediate user deprovisioning
  - When a user gets removed from AD they lose access to the organisation
- Support for MFA
- Can integrate existing systems like Active Directory

How we integrate with AD

- Well we cheat a little,
- We use OneLogin as our SSO provider
  - OneLogin has a SCIM integration with GitHub
  - This is setup via a OneLogin application and a GitHub OAuth application
- Our flow for adding a user to our GitHub organisation
  - Add the user to a specific AD group
  - The user then gets and invite to the GitHub organisation.
  - To accept the invite, the user needs to login to OneLogin
  - Select the GitHub tile (now available in the OneLogin UI)
  - They'll be prompted to sign in to GitHub
  - OneLogin stores this link between its user and GitHub

Some things to think about

- We have a manageable number repositories to have in their own files.
  How do you scale this further while still being able to manage it? Use a DB?

Q&A