Introduction Terraform

Terraform Infrastructure as Code

Syntax name: HCL (HashiCorp Configuration Language) with file extension .tf, .hcl, .tfvars

Programming like syntax but different paradigm

Codify workflow to create/change/configure infrastructure

Why terraform

- Supports multi-cloud & hybrid infrastructure
- Migrate from other cloud providers
- Increase provisioning speed
- Improve efficiency
- Reduce risk

HCL language overview

- **Terraform block** for provider and state backend configuration, see more
- **Provider block** to configure cloud or service provider, see more
- **Resource block** to configure the resource to particular provider
- **DataTypes**
    - String
    - Number
    - Boolean
    - List
    - Set
    - Map
    - Object
    - Tuple
- **Functions** https://developer.hashicorp.com/terraform/language/functions
- **Expressions**

Check out https://developer.hashicorp.com/terraform/language

HCL language overview

```
resource "aws_instance" "example" {
  ami = "abc123"
  network_interface {
    # ...
  }
}
```

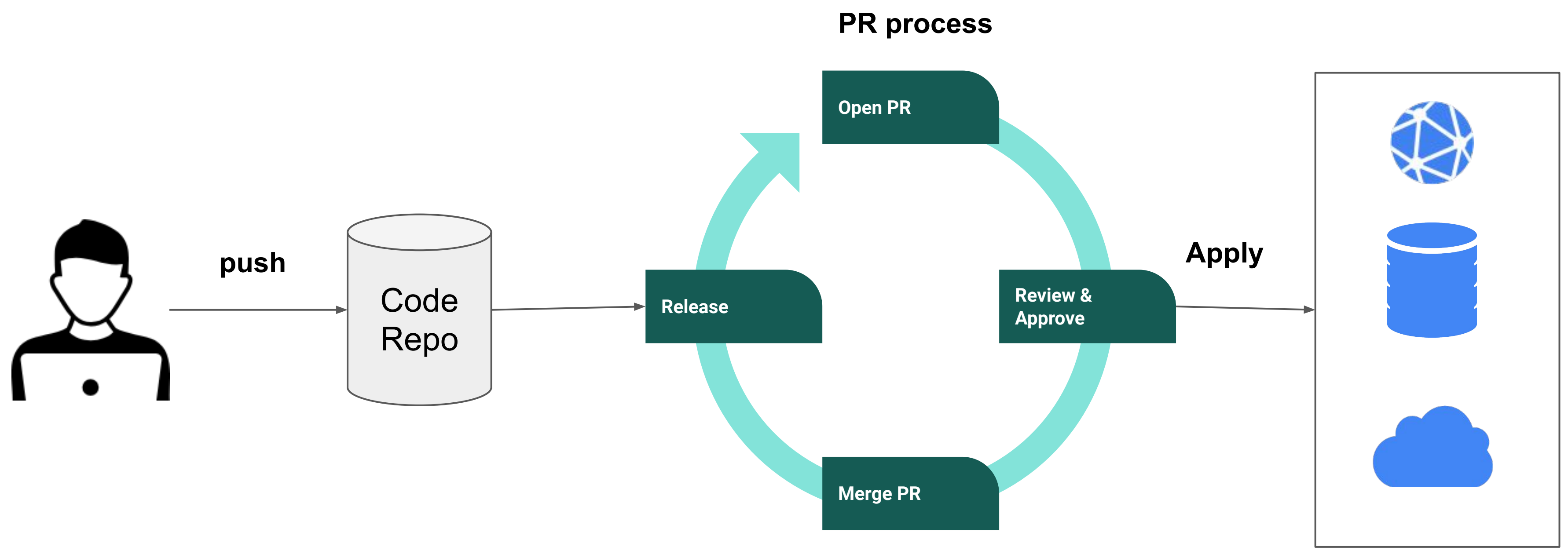
resource "aws_instance"	resource provided by provider
"example"	arbitrary name of resource
ami = "abc123"	resource argument

Code convention https://developer.hashicorp.com/terraform/language/syntax/style.
Run `terraform fmt` to auto format

Terraform CLI

- “terraform init”: For downloading providers and initializing terraform state
- “terraform plan”: For verifying the plan change of infrastructure
- “terraform apply”: For applying the change to target infrastructure

Development workflow


PR process
push
Code Repo
Open PR
Release
Review & Approve
Merge PR
Apply