Terraform Null Commands

Provisioner Local_Exec

```
terraform {
  required_version = "> 0.8.0"
}

resource "null_resource" "health_check" {

 provisioner "local-exec" {

    command = "aws --version && terraform -v && java -version"
  }

}
```

null_resource

The `null_resource` resource implements the standard resource lifecycle but takes no further action.
The `triggers` argument allows specifying an arbitrary set of values that, when changed, will cause the resource to be replaced.

Example Usage

```
resource "aws_instance" "cluster" {
  count = 3

  # ...
}

# The primary use-case for the null resource is as a do-nothing container for
# arbitrary actions taken by a provisioner.
#
```

```
# In this example, three EC2 instances are created and then a null_resource instance
# is used to gather data about all three and execute a single action that affects
# them all. Due to the triggers map, the null_resource will be replaced each time
# the instance ids change, and thus the remote-exec provisioner will be re-run.
resource "null_resource" "cluster" {
  # Changes to any instance of the cluster requires re-provisioning
  triggers = {
    cluster_instance_ids = join(",", aws_instance.cluster.*.id)
  }

  # Bootstrap script can run on any instance of the cluster
  # So we just choose the first in this case
  connection {
    host = element(aws_instance.cluster.*.public_ip, 0)
  }

  provisioner "remote-exec" {
    # Bootstrap script called with private_ip of each node in the clutser
    inline = [
      "bootstrap-cluster.sh ${join(" ", aws_instance.cluster.*.private_ip)}",
    ]
  }
}
```

A few more commands to run

Terraform -help

fmt

validate

show

Terraform validate -json

What is JSON used for?

JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications (e.g., sending some data from the server to the client, so it can be displayed on a web page, or vice versa).

```
Forex@LAPTOP-4C614INS MINGW64 ~/OneDrive/Desktop/python_folder (master)
$ terraform validate -json
{
  "format_version": "0.1",
  "valid": true,
  "error_count": 0,
  "warning_count": 0,
  "diagnostics": []
}
```

Terraform validate --no-color - If supplied, the output will be colourless.

Terraform version

```
Forex@LAPTOP-4C614INS MINGW64 ~/OneDrive/Desktop/python_folder (master)
$ terraform version
Terraform v1.0.11
on windows_amd64
+ provider registry.terraform.io/hashicorp/aws v4.8.0
+ provider registry.terraform.io/hashicorp/null v3.1.1

Your version of Terraform is out of date! The latest version
is 1.1.7. You can update by downloading from https://www.terraform.io/downloads.html
```

Version control systems that supports terraform

Github

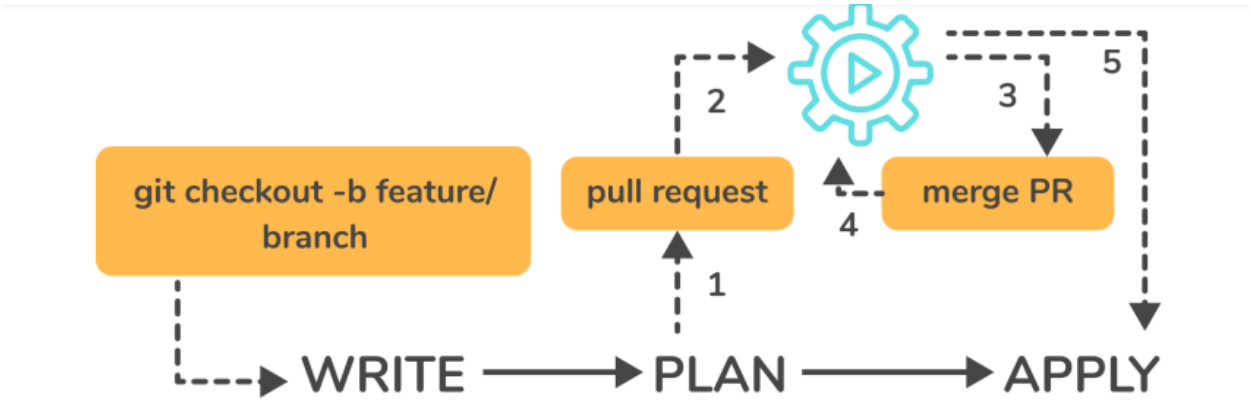
Gitlab EE Gitlab enterprise – you pay

Gitlab CE community (free)

Remember

Git checkout -b hotfix

Workflow Core in Terraform



In Terraform, a backend has a variety of responsibilities:

- Carry out operations (e.g. plan, apply)
- To save workspace-defined variables
- To save state

Child Modules in terraform

In a module call block, the optional providers meta-argument specifies which provider configurations from the parent module will be available inside the child module.

Parent Module

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "4.8.0"
    }
  }
}

provider "aws" {
  # Configuration options
}
```

Child Module

The default "aws" configuration is used for AWS resources in the root

module where no explicit provider instance is selected.

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "4.8.0"
    }
  }
}

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

An alternate configuration is also defined for a different

region, using the alias "usw2".

```
provider "aws" {
  alias  = "usw2"
  region = "us-west-2"
}
```

Naming of resources

For example

aws_vpc

```
  terraform
  {
                    required_providers {
                      aws = {
                        source  = "hashicorp/aws"
                        version = "~> 3.0"
                      }
                    }
                  }
```

```
# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
}

# Create a VPC
resource "aws_vpc" "my-vpc" {
  cidr_block = "10.0.0.0/16"
  tags = {
    Name = "Demo VPC"
  }
}
```