Terraform for Beginners

Traditional IT & Challenges

- Slow Deployment
- Scaling Up/Down Difficult
- Expensive
- Limited Automation
- Human Error
- Wasted Resources
- Inconsistency

Terraform Deploys To

- OnPrem, VMware, AWS, GCP, Azure

Phases

- Init → Plan → Apply → Destroy
- Acts on Resources

HCL Basics

```
<block> <parameters> {
   key1 = value1
   key2 = value2
}
```

block name — resource type

```
resource "local_file" "pet" {
   filename = "/root/pets.txt"
   content = "We love pets"
}
```

```
aws-ec2.tf

resource "aws_instance" "webserver" {
   ami = "ami-0c..."
   instance_type = "t2.micro"
}
```

Terraform Basics

Terraform Providers

- In terraform registry
- Official Providers, Partner Provider, Community Provider
- Organizational Namespace / Type

Hostname / optional

hashicorp / local

Configuration Directory

-local.tf - local config by default

Using Input Variables

```
variable "filename" {
    default = "/root/pets.txt"
}

resource "local_file" "pet" {
    filename = var.filename
}
```

optional →

Variables Block

- default parameter
- type - optional - string, number, boolean, any
- description - optional - what it's used for

additional types -

list ["cat", "dog"] var.prefix[0]

map pet1 = cat, pet2 = dog var.file-content["statement1"]

object - varied types allowed, similar to C# object

tuple - immutable list

terraform apply -var "filename=/root/file.txt"

or export TP_VAR_filename="/root/file.txt"

or .tfvars file - filename = "/root/file.txt"

or .tfvars.json or *.auto.tfvars or *.auto.tfvars.json

or -var-file variables.tfvars

precedence - 4) env vars, 3) terraform.tfvars 2) *.auto 1) -var -var-file

Resource Attributes

Creation of a resource will return an id
${random_pet.my-pet.id}
resource type resource name attribute

Resource Dependencies

```
depends_on = [
    random_pet.my-pet
]
```

← explicit dependency

Output Variables

```
output "pet-name" {
  value       = random_pet.my-pet.id
  description = "blah"
}
```

terraform output Command to display output vars

Terraform State

- .tfstate file contains info about everything that was created
- By default stored in configuration directory
- Contains sensitive data
- Do not store .tfstate in Git
- Put it in S3 / TCloud / Google Cloud

Terraform Show -json Terraform plan
Terraform providers Terraform fmt
- mirror from one dir → another
Terraform output
- variable name for only specific ones
Terraform refresh - sync state file
Terraform graph - generates graph file → visualization sw
terraform graph | dot -Tsvg > graph.svg
Terraform validate - Shows errors in config

Mutable is Immutable

Immutable - unchanged

- no in-place updates
- old remains if new fails
- easier to roll-back or forward

Life Cycle Rules

- rules go inside resource block

```
lifecycle {
    create_before_destroy = true
}
```

ignore_changes - accepts list i.e. tags

prevent_destroy - prevents destruction

Data Sources

- Read attributes from files

```
data "local_file" "dog" {
    filename = "/root/dog.txt"
}
```

can look at content / content_base64

Meta-Arguments

- Multiple of same files

Count

- in resource block

```
variable "filename" {
    default = ["pets.txt", "dogs.txt", "cats.txt"]
}
```

```
for_each
resource "local_file" "pet" {
    filename = each.value
    for_each = toset(var.filename)
}
```

filename = var.filename

count = 3

count = length(var.filename)

variable "filename"

default = ["pets.txt", dogs_file]

type = list(string) or set(string)

Version Constraint

```
terraform {
    required_providers {
        local {
            source = "hc/local"
            version = "1.4.0"
        }
    }
}
```

incremental

version = "!= 2.0.0" or <] - ~>

Terraform & AWS

aws <command> <subcommand> [options & parameters]

```
resource "aws_iam_user" "admin_user" {
    name = "lucy"
    tags = {
      Description = "Technical Team Leader"
      }
}
provider "aws" {
   region = ""
   access_key = "" "" ← no bueno
   secret_key = "" ← since version controlled
}
```

Create credentials in .aws/credentials file
or export as variables

```
resource "aws_iam_policy" "adminUsers" {
   name = "AdminUsers"
   policy = <<EOF
    json here or file ("admin-policy.json)
   EOF
}

resource "aws_iam_user_policy_attachment" "lucy-admin-access" {
   user = aws_iam_user.admin-user.name
   policy_arn = aws_iam_policy.adminUser.arn
}
```

```
resource "aws_s3_bucket" "finance" {
    bucket = "finance-21092020"
    tags = {
        Description = "Finance and Payroll"
    }
}

resource "aws_s3_bucket" "finance-2020" {
    content = "/root/finance/finance-2020.doc"
    key = "finance-2020.doc"
    bucket = aws_s3_bucket.id

data "aws_iam_group" "finance-data" {
    group_name = "finance-analytics"
}

resource "aws_s3_bucket_policy" "finance-policy" {
    bucket = aws_s3_bucket.finance.id
    policy = file("file.json")
}

resource "aws_dynamodb_table" "cars" {
    name = "cars"
    hash_key = "VIN"
    billing_mode = "PAY_PER_REQUEST"
    attribute {
        name = "VIN"
        type = "S"
    }
}
```

Remote State in Terraform

- Mapping configuration to Real World
- Tracking Metadata
- Performance in large envs
- Collaboration
- Automatic loading/reloading state file
- State locking and securing

```
terraform {
  backend "s3" {
    bucket = "  "
    key    = "  "
             "  "
    region =
    dynamo-table = "state-locking"
```

```
terraform   list  - lists resources
    state   mv    - move items
            pull  - display remote state in json
            rm    - remove items
            show  - show details on items
```

Introduction to AWS EC2
T2 - General Purpose is most used

```
resource "aws_instance" "webserver" {
  ami = "ami-..."
  instance_type = "t2.micro"
  user_data = "install instructions via file or EOF
}
resource "aws_key_pair" "web" {
    key_name
    public_key = file("/root/.ssh/web.pub")
}
resource "aws_security_group" "ssh_access" {
  name = "ssh-access"
  description = "Allow Connections"
  ingress {
    from_port = 22
    to_port = 22
    protocal = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
provisioner "remote-exec" {
    inline ["sudo apt update",
            "sudo apt upgrade"]
```

vpc-security-group-ids

on-failure inside provisioner block to handle case where it fails

```
resource "aws_eip" "eip" elastic ip /static
```

Use template tools to create custom AMIs w/ your software installed

Terraform taint
mark a resource as "jacked up" so it will be recreated by terraform apply
terraform untaint to cancel it.

export TF_LOG = TRACE
export TF_LOG_PATH = /tmp/log.log
unset VAR_NAME

INFO
WARNING
ERROR
DEBUG
TRACE

Terraform Import
Import existing infra not managed by terraform
terraform import <resource_type>.<resource_name> <attributes>
- Create empty resource block in .tf file
- define values from console or tfstate file
- terraform plan will show if all is good

Terraform Modules

```
module "dev-webserver" {
  source = "../aws-instance"
}
```

you can specify variables here too, ie
dev, qa, prod, stage, etc instances
There are many modules in Terraform Registry

Terraform - More Functions

numeric — min, max, ceil, floor
string — split, lower, upper, title, substr,
collection — length, index, element, contains
Maps — keys, values, lookup → (can provide default if ONLY)
Equality operators are supported, <, >, ==, !=

Terraform Workspaces

terraform workspace new name
terraform workspace list
    * is currently selected