

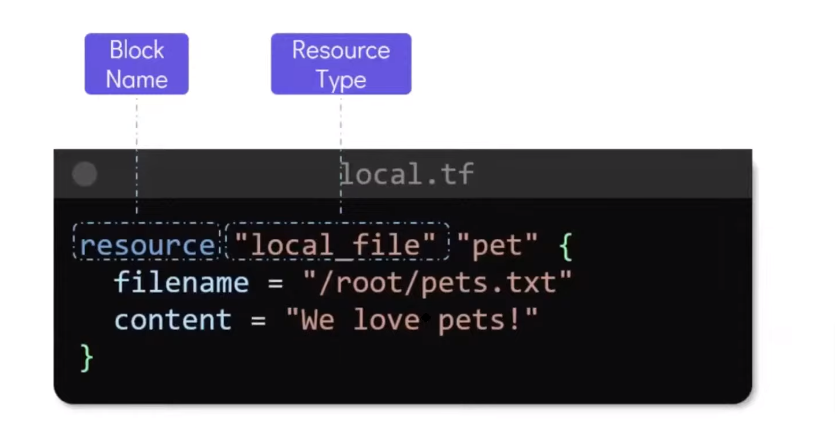
# Blocks and Arguments

A block is a container for other content and An argument assigns a value to a particular name:

filename = “/home/ubuntu/abc123.txt”

The identifier before the equals sign is the argument name, and the expression after the equals

sign is the argument's value.



## 1.EXAMPLE in local.tf file

## resource “local\_file” “devops”{

## filename - “/home/ubuntu/Terraform/terraform\_local

## /devops\_automated.txt”

## content - “I want to become a DevOps Engineer who knows Terraform”

## }

Resource block: block name used to mention the type of the block. The resource block expects

two labels, which are local\_file and “pet” in the example above. A particular block type may have

any number of required labels, or it may require none.

resource “<provider>\_stresource type> “<resource name>“

Argument1 =““

Argument2=““

}

Local = provider, file = type, “pet” name of the resource.

Then we have arguments, filename, content, etc

We can have multiple resources I

## resource “random\_string” “rand-str” {

## length = 16

## special = true

## override\_special = “#$%8°0-\_=+1 <>?”

## }

## output “rand-str” {

## value = random\_string.rand-str[\*].result

## }

devops\_automated.txt

local.tf

terraform.tfstate

Terraform with Docker

Terraform needs to be told which provider to be used in the automation, hence we need to give

the provider name with source and version.

For Docker, we can use this block of code in your main.tf

**Terraform block**

## terraform {

## required\_providers {

## docker = {

## source = “kreuzwerker/docker”

## version = “~> 2.21.0”

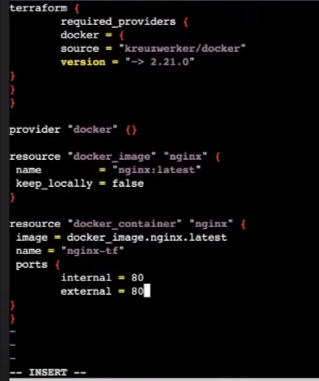
## }

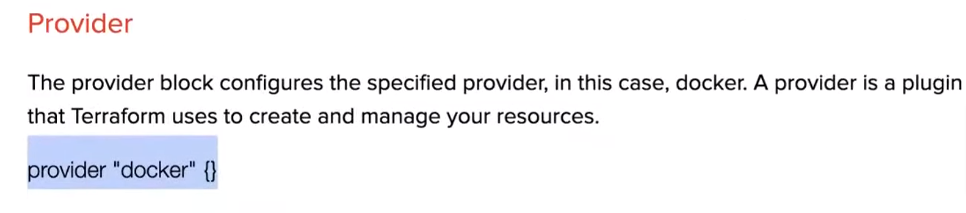
## }

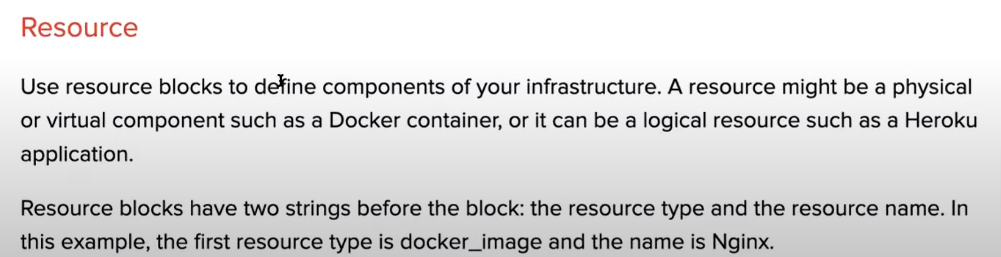
## }

**Note:** kreuzwerker/docker, is shorthand for registry.terraform.io/kreuzwerker/docker.

**2.EXAMPLE**







## terraform {

## required\_providers {

## docker = {

## source = “kreuzwerker/docker”

## version = “~> 2.21.0”

## }

## }

## }

## provider “docker” {}

## resource “docker\_image” “nginx” {

## name = “nginx:latest”

## keep\_locally = false

## }

## resource “docker\_container” “nginx” {

## image = docker\_image.nginx.latest

## name = “nginx-tf”

## ports {

## internal = 80

## external = 80

## }

## }

TO RUN:

**$ terraform init**

**$ terraform validate**

**$ terraform plan**

**$ terraform apply**

creates an EC2 instance with a Security Group and creates a S3 Bucket

https://www.easydeploy.io/blog/terraform-state-file/

## ################### VARIABLES ##############################

## variable “name” {

## type    = string

## default = “Terraform-state”

## }

## variable “access\_key” {

## type = string

## }

## variable “secret\_key” {

## type = string

## }

## variable “region” {

## type = string

## default = “us-east-1”

## }

## ################### PROVIDER ##############################

## provider “aws” {

## access\_key = var.access\_key

## secret\_key = var.secret\_key

## region     = var.region

## }

## ################### EC2 INSTANCE ###########################

## resource “aws\_instance” “test” {

## ami                         = “ami-052efd3df9dad4825”

## instance\_type               = “t2.micro”

## associate\_public\_ip\_address = true

## security\_groups             = [aws\_security\_group.test.name]

## tags = {

## Name = var.name

## }

## }

## ################### SECURITY GROUP ##########################

## resource “aws\_security\_group” “test” {

## name        = var.name

## description = “Allow TLS inbound traffic”

## ingress {

## description = “allow access to web”

## from\_port   = 80

## to\_port     = 80

## protocol    = “tcp”

## cidr\_blocks = [“0.0.0.0/0”]

## }

## egress {

## from\_port   = 0

## to\_port     = 0

## protocol    = “-1”

## cidr\_blocks = [“0.0.0.0/0”]

## }

## tags = {

## Name = var.name

## }

## }

## ################### S3 BUCKET ##############################

## resource “aws\_s3\_bucket” “b” {

## bucket = lower(“${var.name}-test-bucket-state-file”)

## force-destroy = true

## }

## resource “aws\_s3\_bucket\_acl” “acl” {

## bucket = aws\_s3\_bucket.b.id

## acl    = “private”

## }

## ################### OUTPUTS ##############################

## output “IpAddress” {

## value = aws\_instance.test.public\_ip

## }

## output “BucketName” {

## value = aws\_s3\_bucket.b.bucket

## }

The above code creates an EC2 instance with a Security Group and creates a S3 Bucket also.

Run **terraform apply** command to create resources on our AWS Cloud.