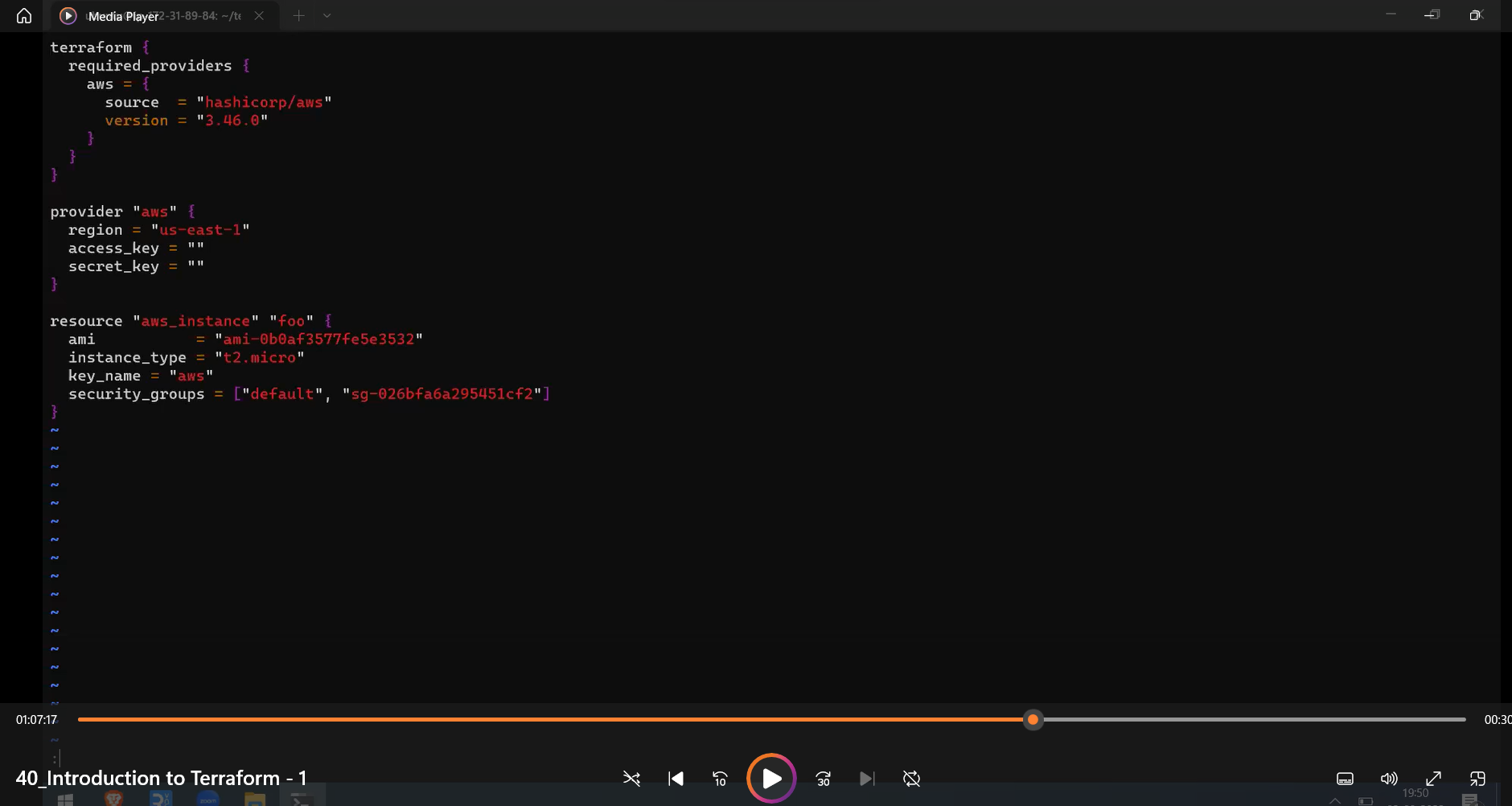


Terraform file always ends with .tf as extention.



1 ls -la

2 cd terraform-code

3 ls -la

4 cd hello\_terraform

7 ls -a

8 vi main.tf

9 terraform validate

10 terraform init

11 terraform validate

13 terraform fmt

14 terraform plan

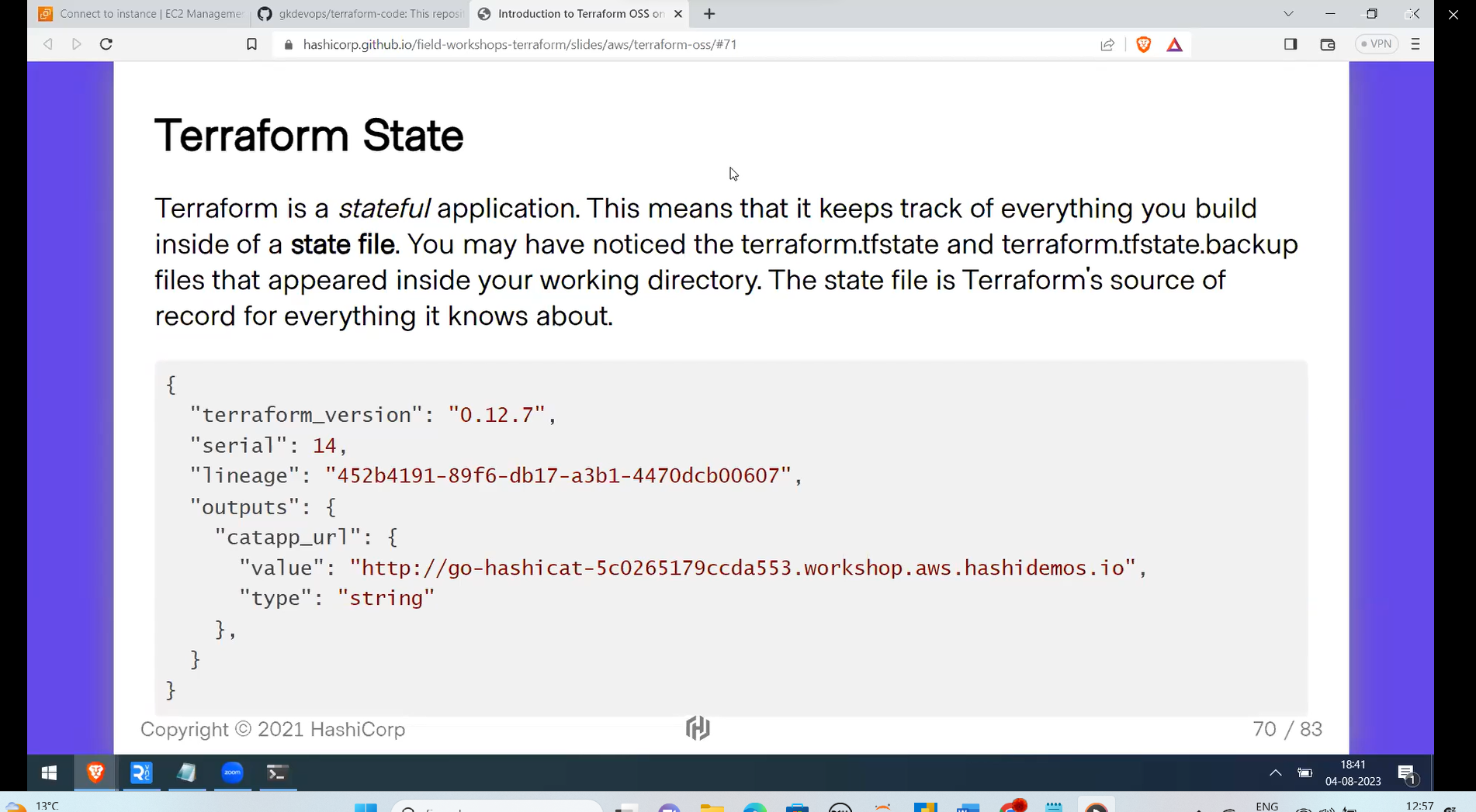
15 terraform plan -out terraform\_plan

16 terraform apply

17 vi main.tf

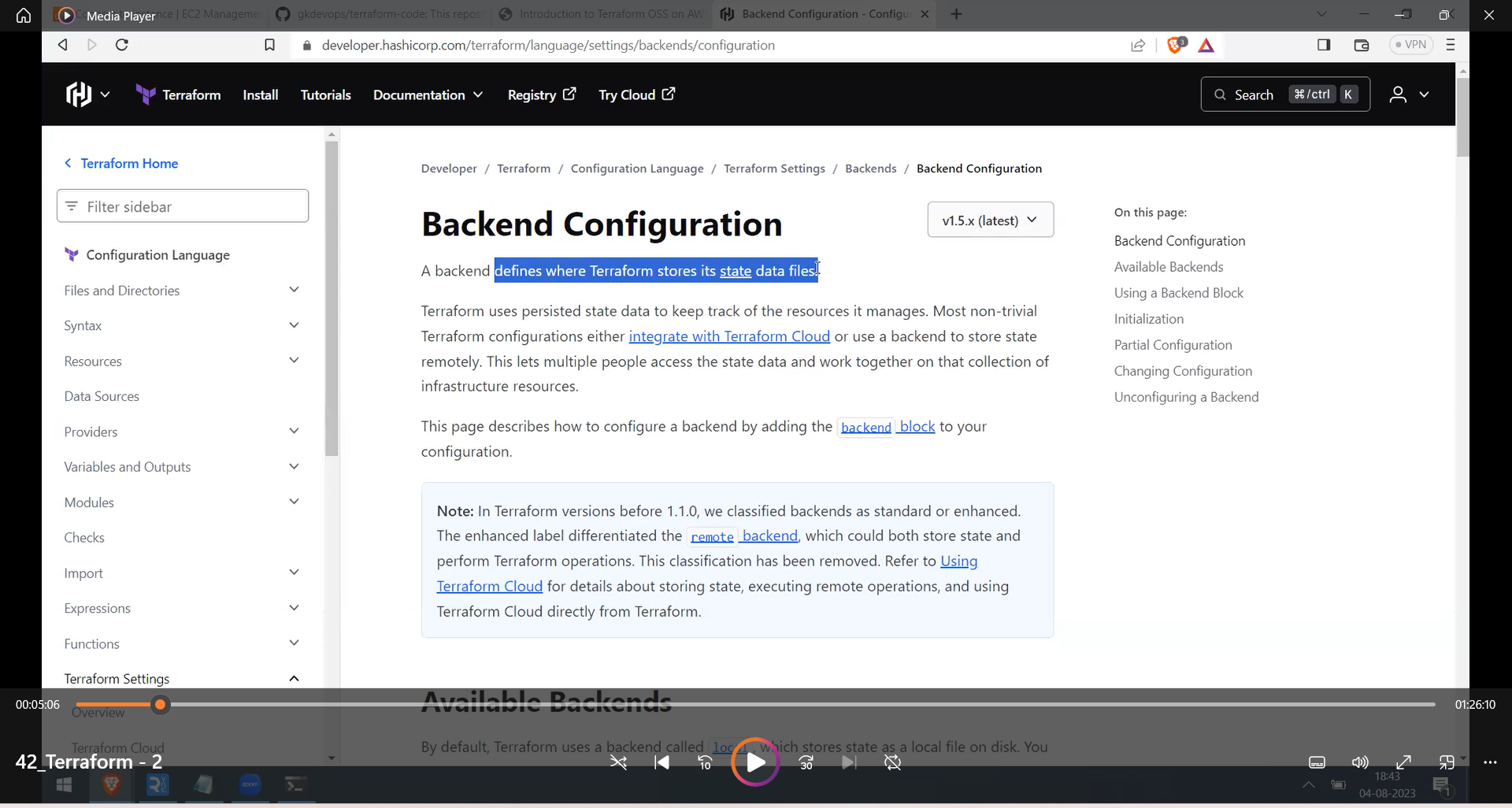
18 terraform apply

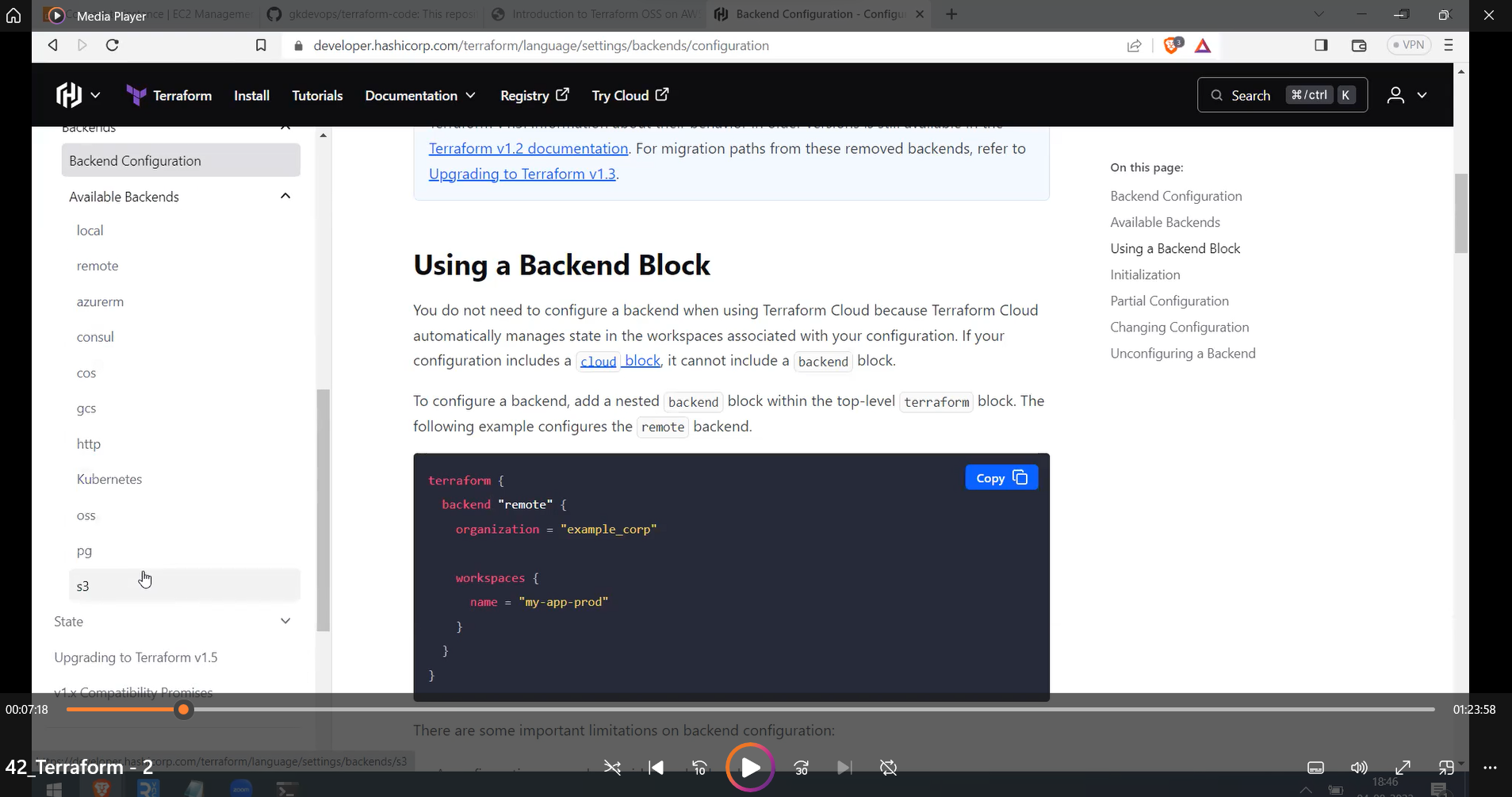
19 history



With out applying the apply command the state file cannot create.

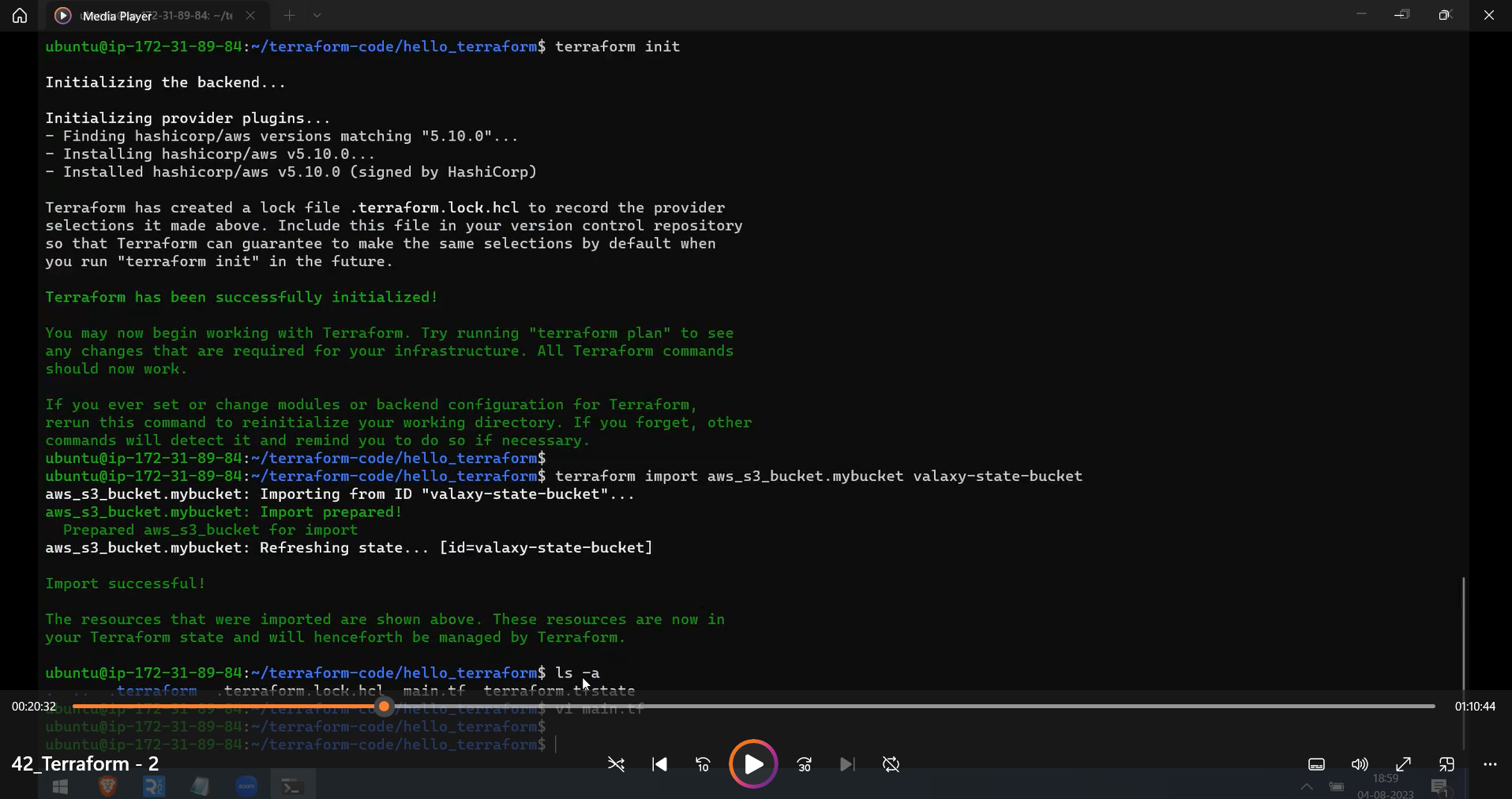
State file wants to store in backend configuration.



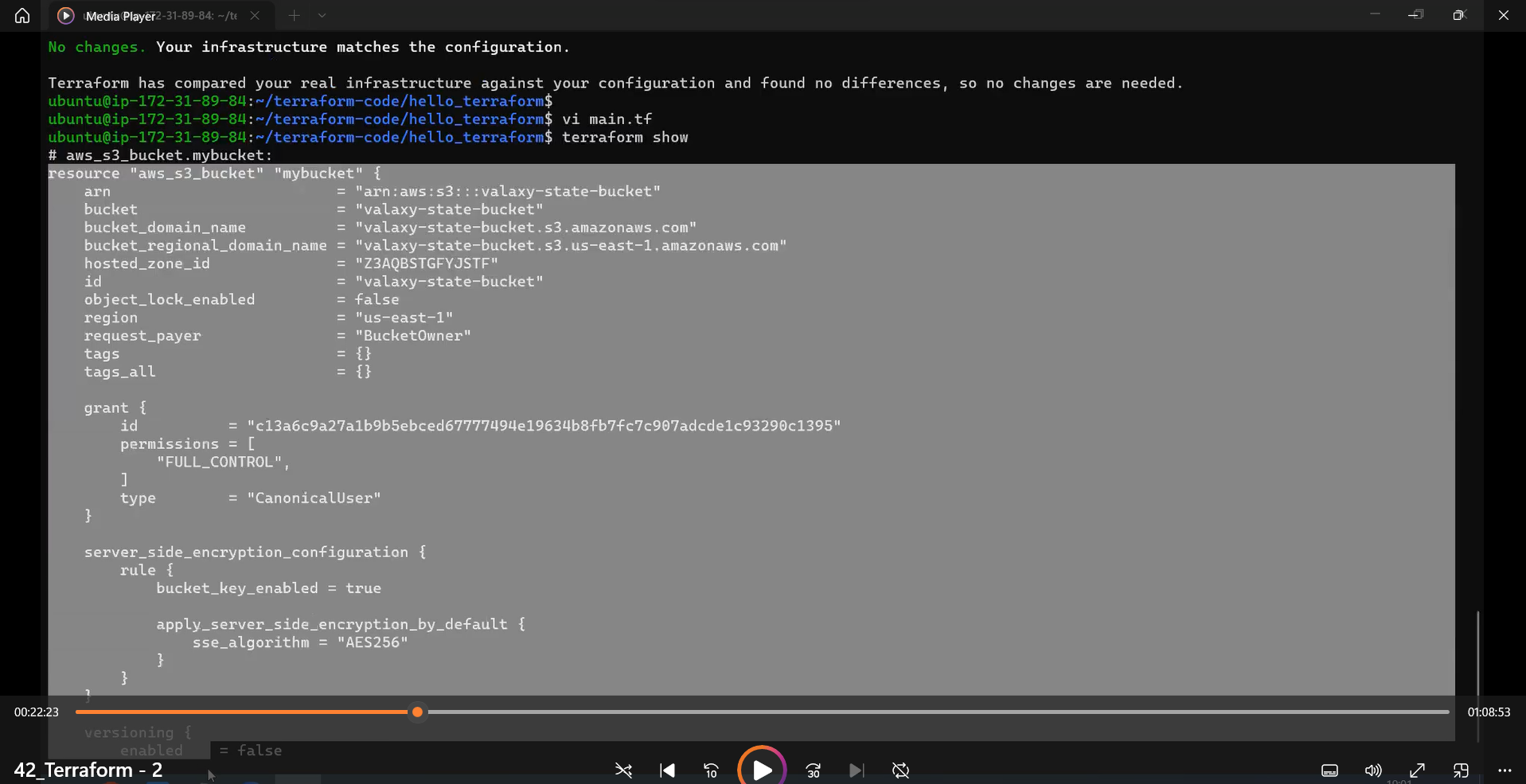


For aws -s3,azure- azurerm

Along with store of the state file, wants to lock the file with dynamodb table permissions



Terraform show – it will generate the code automatically from state file



terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "5.25.0"

}

}

}

provider "aws" {

region = "us-east-1"

access\_key = "AKIARL5M64ETEPGPIJX4"

secret\_key = "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"

}

/\*

resource "aws\_instance" "foo" {

ami = "ami-0b0af3577fe5e3532"

instance\_type = "t2.micro"

key\_name = "aws"

security\_groups = ["default", "sg-026bfa6a295451cf2"]

}

\*/

resource "aws\_instance" "foo" {

ami = "ami-0b0ea68c435eb488d" # us-east-1

instance\_type = "t2.micro"

for importing the bucket:

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "5.25.0"

}

}

}

provider "aws" {

region = "us-east-1"

access\_key = "AKIARL5M64ETEPGPIJX4"

secret\_key = "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"

}

/\*

resource "aws\_instance" "foo" {

ami = "ami-0b0af3577fe5e3532"

instance\_type = "t2.micro"

key\_name = "aws"

security\_groups = ["default", "sg-026bfa6a295451cf2"]

}

\*/

/\*

resource "aws\_instance" "foo" {

ami = "ami-0b0ea68c435eb488d" # us-east-1

instance\_type = "t2.micro"

}

\*/

import {

to = aws\_s3\_bucket.mybucket

id = "mybucketterraform"

}

For import the bucket creating file:

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "5.25.0"

}

}

}

provider "aws" {

region = "us-east-1"

access\_key = "AKIARL5M64ETEPGPIJX4"

secret\_key = "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"

}

/\*

resource "aws\_instance" "foo" {

ami = "ami-0b0af3577fe5e3532"

instance\_type = "t2.micro"

key\_name = "aws"

security\_groups = ["default", "sg-026bfa6a295451cf2"]

}

\*/

/\*

resource "aws\_instance" "foo" {

ami = "ami-0b0ea68c435eb488d" # us-east-1

instance\_type = "t2.micro"

}

\*/

/\*

import {

to = aws\_s3\_bucket.mybucket

id = "mybucketterraform"

}

\*/

To create vpc :

terraform {

backend "s3" {

bucket = "mybucketterraform"

key = "dev/network/terraform.tfstate"

region = "us-east-1"

dynamodb\_table = "terraforms-state-lock"

profile = "aws-prod-account"

}

}

# Configure the AWS Provider

provider "aws" {

profile = "aws-prod-account"

region = "us-east-1"

}

# define a variable

variable "vpc\_cidr" {

type = string

default = "10.0.0.0/16"

validation {

error\_message = "the CIDR provided is invalid."

condition = can(cidrhost(var.vpc\_cidr, 0))

}

}

# create a aws resource

resource "aws\_vpc" "main" {

cidr\_block = var.vpc\_cidr

instance\_tenancy = "default"

tags = {

Name = "Terraform\_VPC"

}

}

# Query all avilable Availibility Zone.

data "aws\_availability\_zones" "available" {}

# Print the value of the variables

output "aws\_availability\_zones" {

value = data.aws\_availability\_zones.available

}