HashiCorp

Terraform

Write, Plan, and Create Infrastructure as Code

Agenda

- Infrastructure-as-Code
- Introduction to Terraform
- Code example
- Terraform VS CloudFormation
- Conclusion

Problem stating

Manual creation of Infrastructure

- High cost
- Human errors
- No agility
- Hard maintenance

https://www.firstaidforfree.com/wp-content/uploads/2017/01/manual-handling.jpg

Problem solution

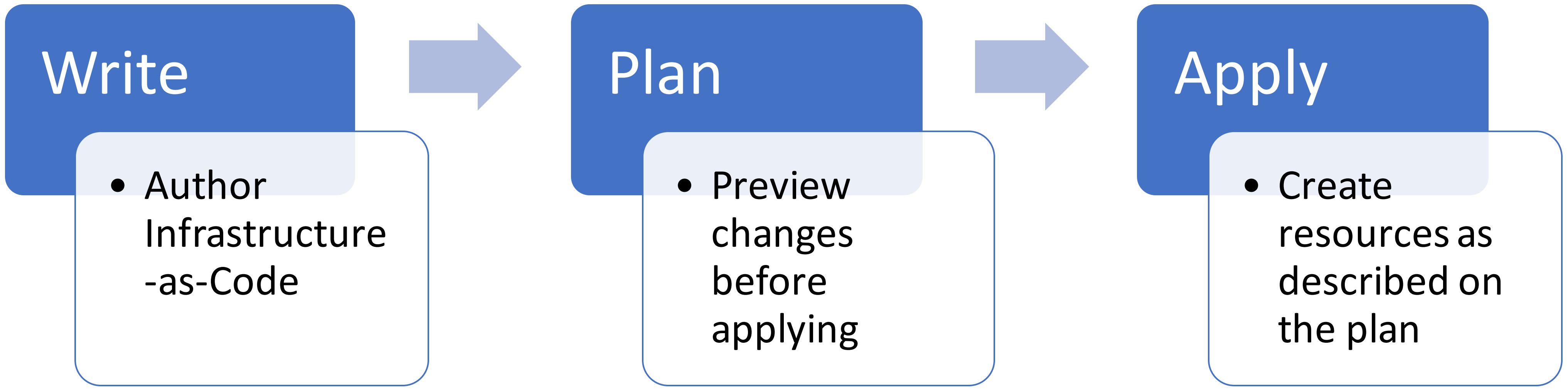
Infrastructure as code is defining infrastructure using **configuration files** (code), rather than manual provisioning and maintenance.

https://www.ranorex.com/test-automation-tools/

+ repeatable process
+ user-friendly language

What is Terraform?

Terraform is a tool for provisioning and maintaining infrastructure efficiently by writing, planning and creating Infrastructure-as-Code.


Write
• Author Infrastructure -as-Code
Plan
• Preview changes before applying
Apply
• Create resources as described on the plan

Key Concepts

Providers

aws

Google Cloud

Azure

Resources

VM

Popularity

Terraform is **hyped**.

http://rabbisacks.org/wp-content/uploads/2016/03/enthusiasm-POST.jpg

Maturity

- Current version: v0.11.13
- Thousands of contributors, big community
- Fast paced development

Use cases

MULTI-CLOUD DEPLOYMENT

MULTI-TIER APPLICATIONS

DISPOSABLE ENVIRONMENTS

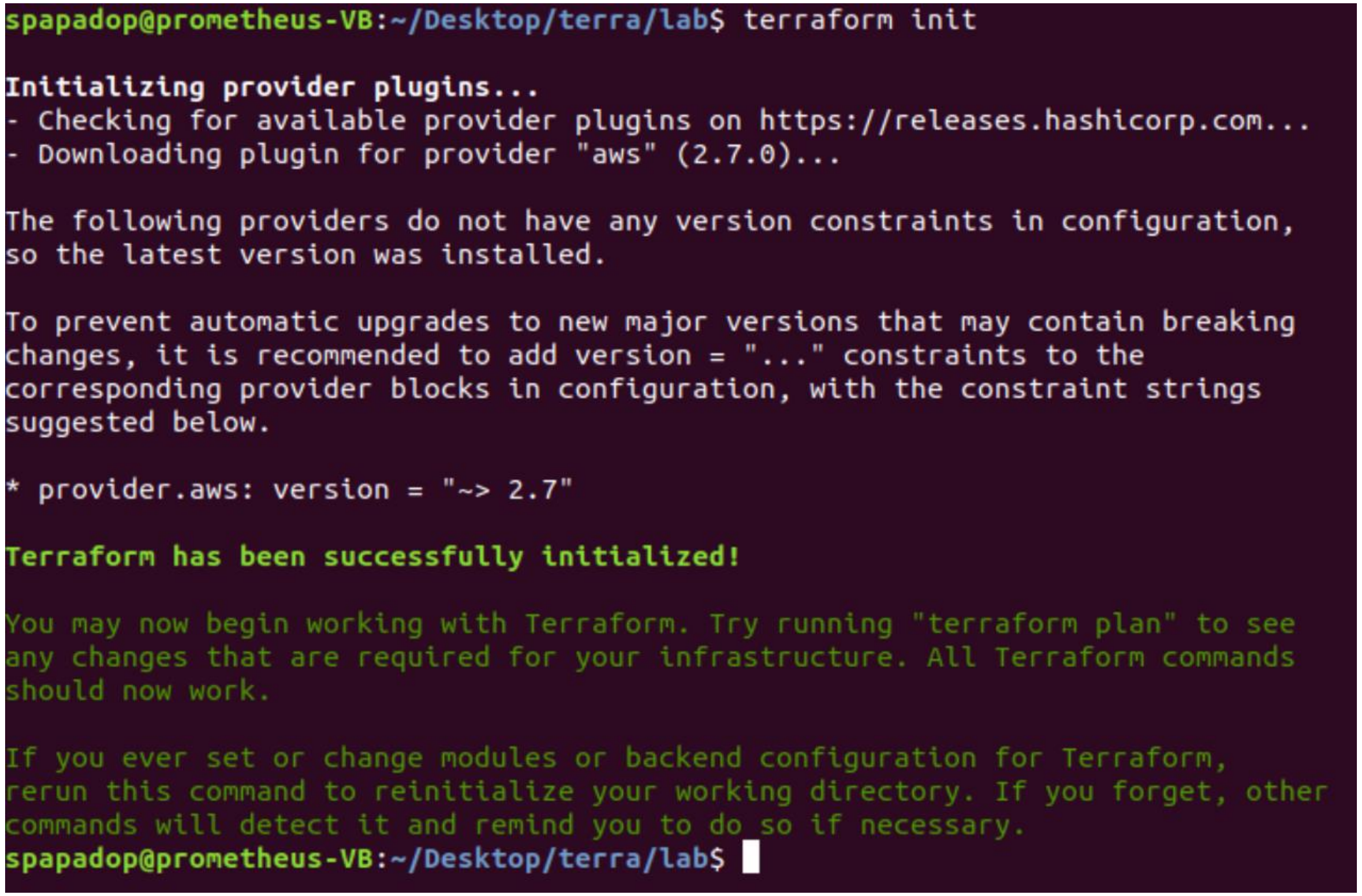
Linus Torvalds – “Talk is cheap. Show me the code.”

```
provider "aws" {
   access_key = "kdsU3jds92l...1DdsA"
   secret_key = "Ddsawd39dds...k34kD"
   region = "eu-west-1"
}

resource "aws_instance" "my_instance_name" {
   ami = "ami-76d6f519"
   instance_type = "t2.micro"
   count = 5
}
```

Linus Torvalds – “Talk is cheap. Show me the code.”

```
provider "aws" {
   access_key = "kdsU3jds92l...1DdsA"
   secret_key = "Ddsawd39dds...k34kD"
   region = "eu-west-1"
}

resource "aws_instance" "my_instance_name" {
   ami = "ami-76d6f519"
   instance_type = "t2.micro"
   count = 5
}
```

```
$ terraform init
```

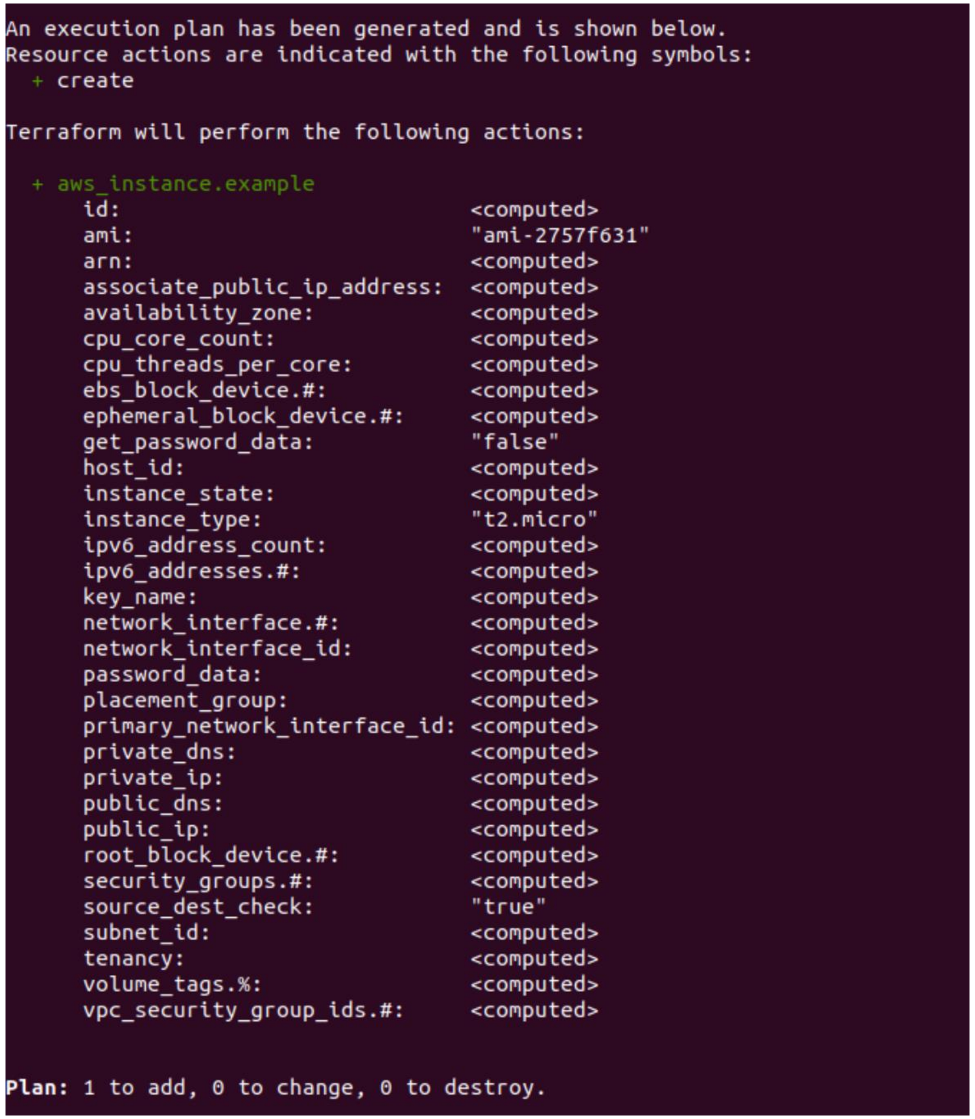


```
spapadop@prometheus-VB:~/Desktop/terra/lab$ terraform init

Initializing provider plugins...
- Checking for available provider plugins on https://releases.hashicorp.com...
- Downloading plugin for provider "aws" (2.7.0)...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.aws: version = "~> 2.7"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
spapadop@prometheus-VB:~/Desktop/terra/lab$
```

Linus Torvalds – “Talk is cheap. Show me the code.”

```
provider "aws" {
   access_key = "kdsU3jds92l...1DdsA"
   secret_key = "Ddsawd39dds...k34kD"
   region = "eu-west-1"
}

resource "aws_instance" "my_instance_name" {
   ami = "ami-76d6f519"
   instance_type = "t2.micro"
   count = 5
}
```

```
$ terraform plan
```

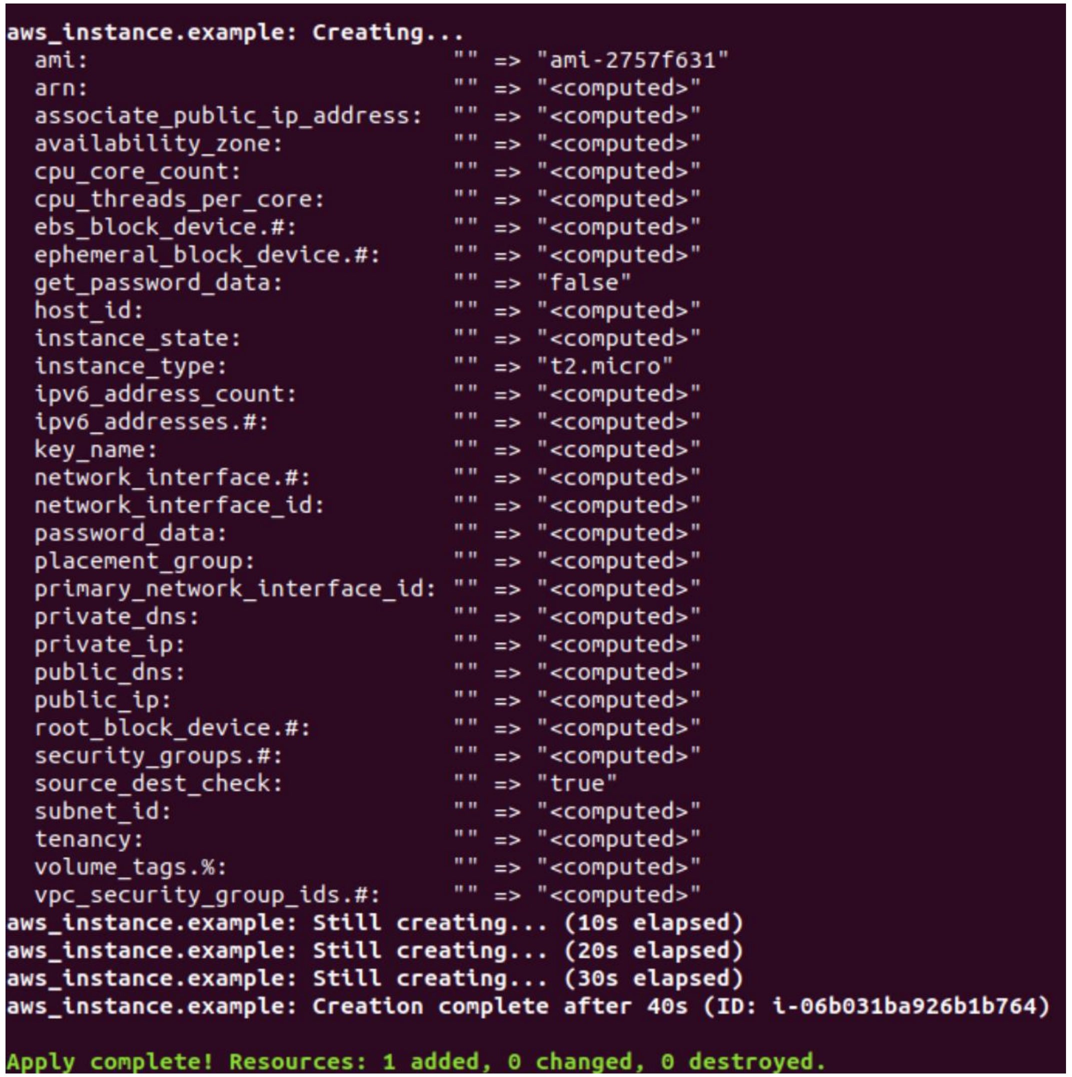


```
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  + aws_instance.example
      id:                           <computed>
      ami:                          "ami-2757f631"
      arn:                          <computed>
      associate_public_ip_address:  <computed>
      availability_zone:            <computed>
      cpu_core_count:               <computed>
      cpu_threads_per_core:         <computed>
      ebs_block_device.#:           <computed>
      ephemeral_block_device.#:     <computed>
      get_password_data:            "false"
      host_id:                      <computed>
      instance_state:               <computed>
      instance_type:                "t2.micro"
      ipv6_address_count:           <computed>
      ipv6_addresses.#:             <computed>
      key_name:                     <computed>
      network_interface.#:          <computed>
      network_interface_id:         <computed>
      password_data:                <computed>
      placement_group:              <computed>
      primary_network_interface_id: <computed>
      private_dns:                  <computed>
      private_ip:                   <computed>
      public_dns:                   <computed>
      public_ip:                    <computed>
      root_block_device.#:          <computed>
      security_groups.#:            <computed>
      source_dest_check:            "true"
      subnet_id:                    <computed>
      tenancy:                      <computed>
      volume_tags.%:                <computed>
      vpc_security_group_ids.#:     <computed>


Plan: 1 to add, 0 to change, 0 to destroy.
```

Linus Torvalds – “Talk is cheap. Show me the code.”

```
provider "aws" {
   access_key = "kdsU3jds92l...1DdsA"
   secret_key = "Ddsawd39dds...k34kD"
   region = "eu-west-1"
}

resource "aws_instance" "my_instance_name" {
   ami = "ami-76d6f519"
   instance_type = "t2.micro"
   count = 5
}
```

```
$ terraform apply
```

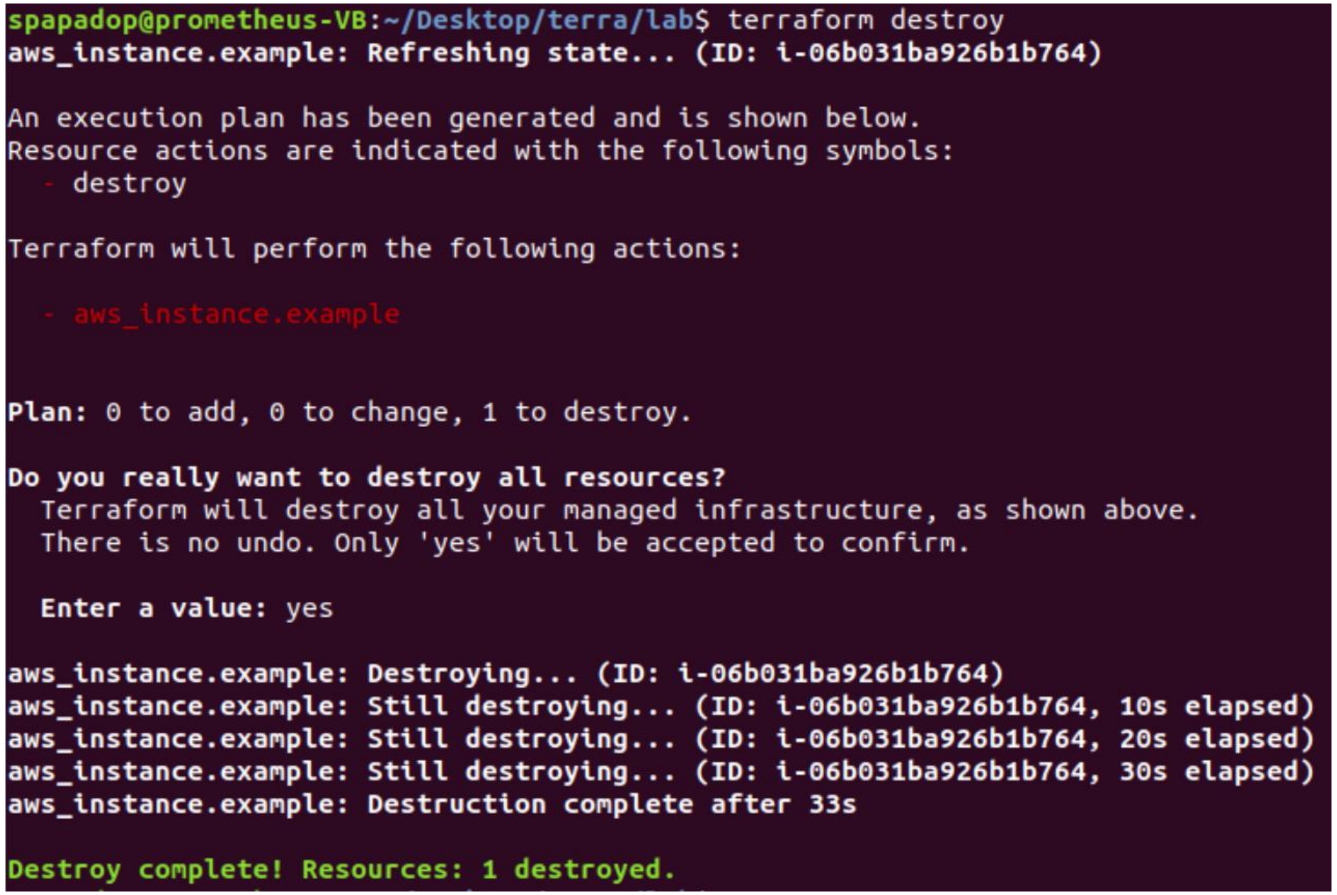


```
aws_instance.example: Creating...
  ami:                          "" => "ami-2757f631"
  arn:                          "" => "<computed>"
  associate_public_ip_address:  "" => "<computed>"
  availability_zone:            "" => "<computed>"
  cpu_core_count:               "" => "<computed>"
  cpu_threads_per_core:         "" => "<computed>"
  ebs_block_device.#:           "" => "<computed>"
  ephemeral_block_device.#:     "" => "<computed>"
  get_password_data:            "" => "false"
  host_id:                      "" => "<computed>"
  instance_state:               "" => "<computed>"
  instance_type:                "" => "t2.micro"
  ipv6_address_count:           "" => "<computed>"
  ipv6_addresses.#:             "" => "<computed>"
  key_name:                     "" => "<computed>"
  network_interface.#:          "" => "<computed>"
  network_interface_id:         "" => "<computed>"
  password_data:                "" => "<computed>"
  placement_group:              "" => "<computed>"
  primary_network_interface_id: "" => "<computed>"
  private_dns:                  "" => "<computed>"
  private_ip:                   "" => "<computed>"
  public_dns:                   "" => "<computed>"
  public_ip:                    "" => "<computed>"
  root_block_device.#:          "" => "<computed>"
  security_groups.#:            "" => "<computed>"
  source_dest_check:            "" => "true"
  subnet_id:                    "" => "<computed>"
  tenancy:                      "" => "<computed>"
  volume_tags.%:                "" => "<computed>"
  vpc_security_group_ids.#:     "" => "<computed>"
aws_instance.example: Still creating... (10s elapsed)
aws_instance.example: Still creating... (20s elapsed)
aws_instance.example: Still creating... (30s elapsed)
aws_instance.example: Creation complete after 40s (ID: i-06b031ba926b1b764)

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Linus Torvalds – “Talk is cheap. Show me the code.”

```
provider "aws" {
   access_key = "kdsU3jds92l...1DdsA"
   secret_key = "Ddsawd39dds...k34kD"
   region = "eu-west-1"
}

resource "aws_instance" "my_instance_name" {
   ami = "ami-76d6f519"
   instance_type = "t2.micro"
   count = 5
}
```

```
$ terraform destroy
```



```
spapadop@prometheus-VB:~/Desktop/terra/lab$ terraform destroy
aws_instance.example: Refreshing state... (ID: i-06b031ba926b1b764)

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  - aws_instance.example


Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.example: Destroying... (ID: i-06b031ba926b1b764)
aws_instance.example: Still destroying... (ID: i-06b031ba926b1b764, 10s elapsed)
aws_instance.example: Still destroying... (ID: i-06b031ba926b1b764, 20s elapsed)
aws_instance.example: Still destroying... (ID: i-06b031ba926b1b764, 30s elapsed)
aws_instance.example: Destruction complete after 33s

Destroy complete! Resources: 1 destroyed.
```

Terraform VS Cloudformation

HashiCorp Terraform

- Cloud agnostic (not 100%)
- Open source
- Language: HCL, JSON

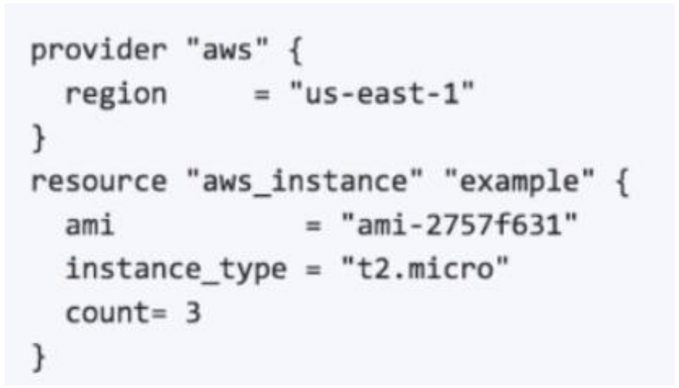
CloudFormation

- Tight integration with AWS
- Property of AWS
- Language: JSON, YAML

Terraform VS Cloudformation

HashiCorp Terraform

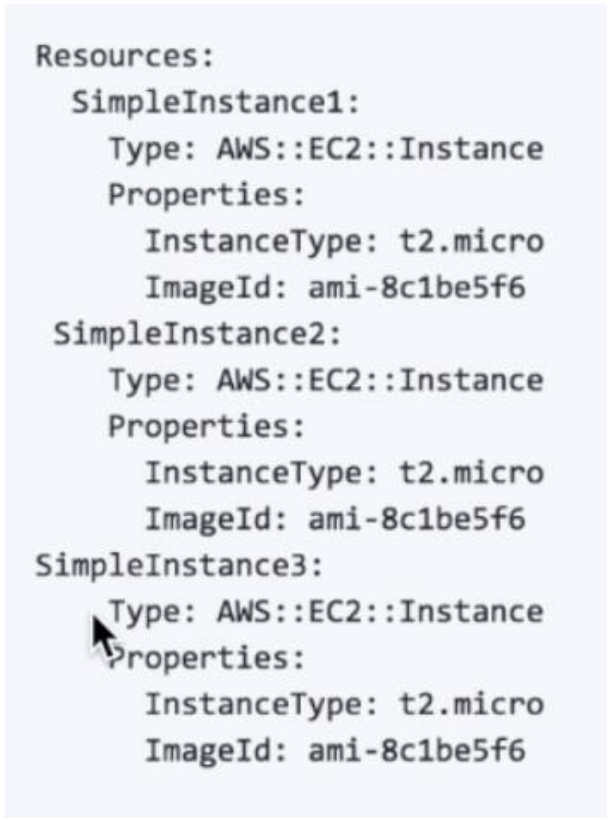
- Simple syntax
  - $ terraform init
  - $ terraform apply
- Support easy multiple configuration



```
provider "aws" {
  region      = "us-east-1"
}
resource "aws_instance" "example" {
  ami           = "ami-2757f631"
  instance_type = "t2.micro"
  count= 3
}
```

CloudFormation

- CLI syntax is verbose
  - aws cloudformation create-stack --stack-name myteststack --template-body file://template.yaml
- Repeated code



```
Resources:
  SimpleInstance1:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: t2.micro
      ImageId: ami-8c1be5f6
 SimpleInstance2:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: t2.micro
      ImageId: ami-8c1be5f6
SimpleInstance3:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: t2.micro
      ImageId: ami-8c1be5f6
```

Terraform VS Cloudformation

HashiCorp Terraform

- GUI on enterprise version
- Managing state file
- Easy working with multiple configuration files

CloudFormation

- GUI support for free
- No need to manage state file
- Complex nested architecture on multiple configuration files

Pros

- Automation the process of creating cloud services.
- Small learning curve.
- Manage a lot of different providers & resources.
- It is opensource, with a big community & many contributors.
- Well documented with a good API.
- Reusable code.

Cons

- Not completely agnostic
- It is opensource, causing delay
- Provider technology knowledge is necessary

Conclusion

- Terraform is an immature, much promising tool.
- Infrastructure-as-Code is here to stay.