https://github.com/matrix0415/terraform-tutorial-in-ithome-cloudedge-summit

Step into Terraform

Terraform experiences from KKStream

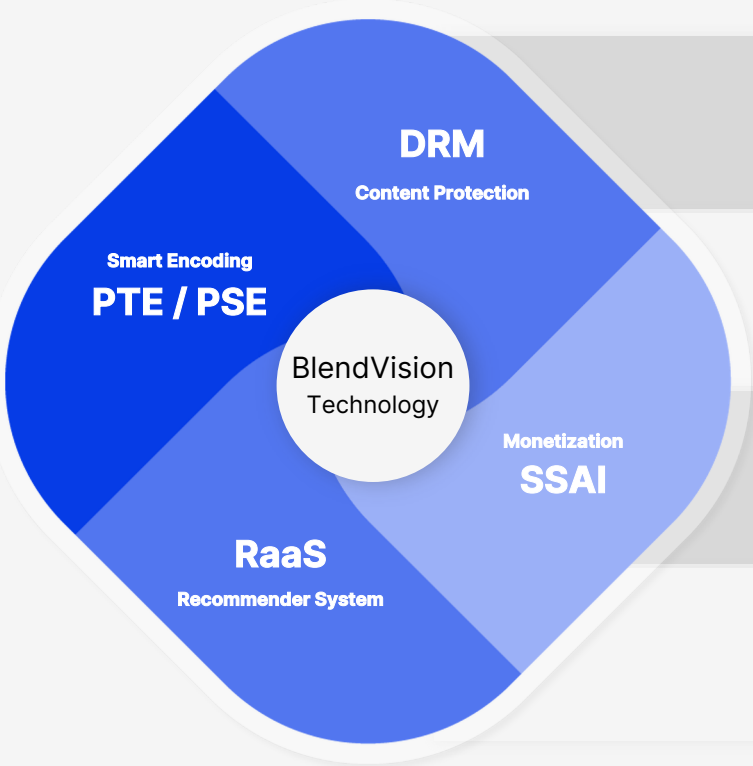
KKstream

Date 2021.11.16

Presenter HsuanKu


KKstream
Enabling seamless online
video experience since 2016
Tomorrow Together KDDI
J:COM
TELASA
uP!!!
FEELCYCLE
中華電信 Chunghwa Telecom
Foxconn 鴻海科技集團
TVBS
giloo 紀貴前信
ADVANTECH
Amway
KKTIX
KKTV
BlendVision
© 2021 KKStream. All Rights Reserved.

KKStream – BlendVision Products & Solution



BlendVision
KALEIDO

Over-the-Top Platform
Build your own branded online video platform with everything you need.

BlendVision
MOMENT

Live Streaming Platform
Live stream the whole world to anyone anywhere in seamless fashion.

BlendVision
LOOM

Video Encoding & Delivery
Encode and deliver beautiful video at unbelievably minimal bandwidth.

BlendVision
ENTERPRISE

Professional Consultant Service
Custimizable E2E streaming solution.

KKStream – Technology research

- Streaming technologies
  - BlendVision PTE : Per-title encoding
  - BlendVision PSE : Perceptual streaming technology
  - BlendVision DRM : Digital-right management
  - BlendVision SSAI : Server-side AD. Insertion

- Data technologies
  - BlendVision Recommender system

About me

- 古宣佑
- KKStream Technology Research Center
- Architect
- hsuanku@kkstream.com

Remember – AWS Credit

- Credit: USD 25
- Survey: https://amazonmr.au1.qualtrics.com/jfe/form/SV_ahKlyIL3fO3EJdH
- Q3 fill in: **11/16 Cloud Summit**

- Submit the survey **today (11/16)**
- Receive the credit in your email, submit the credit code to your AWS account.

Agenda

Terraform introduction

Feature 1: Remote backend

Feature 2: Environment consistency

Feature 3: Multi-clouds/regions

Feature 4: Modules

Premium Terraform services

Summary

Q & A

Terraform introduction

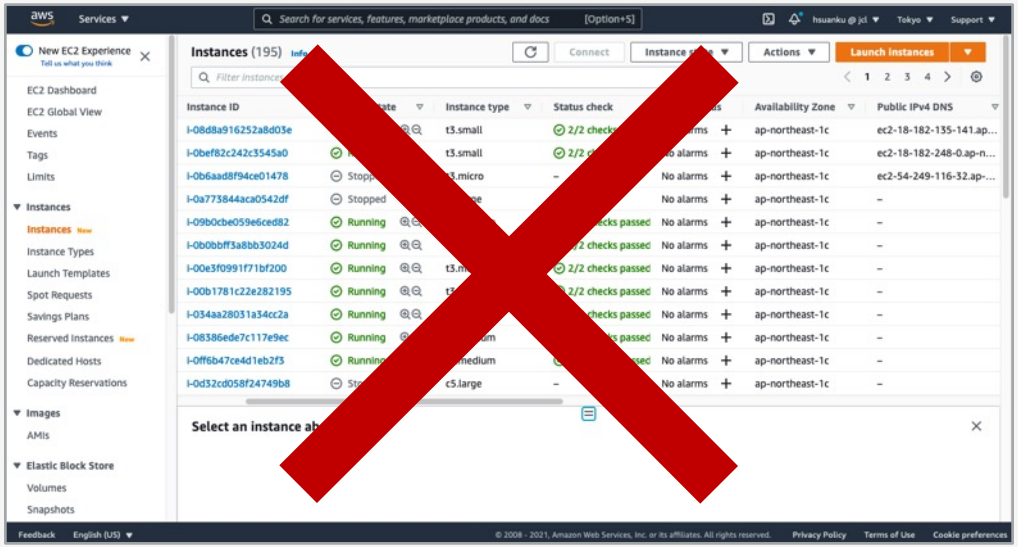
Infrastructure as Code

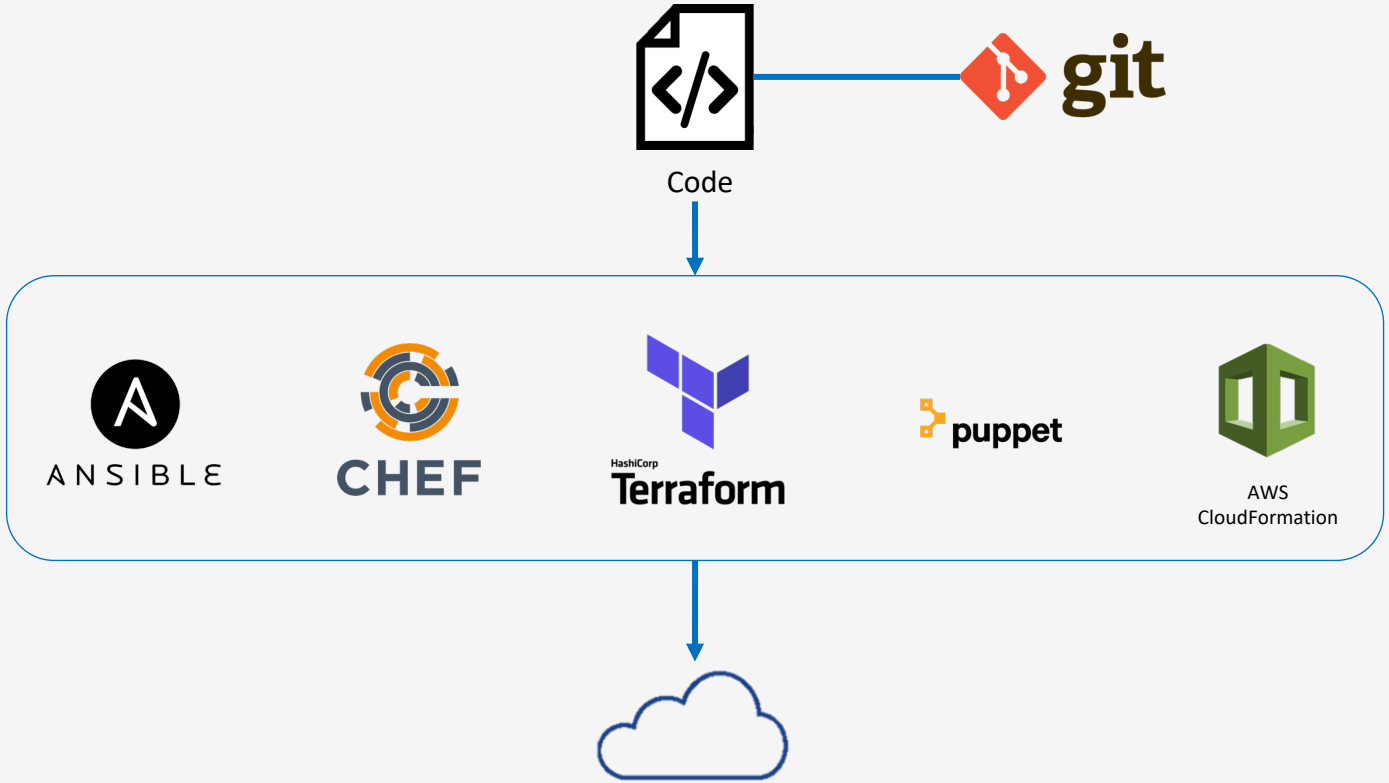
Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

https://en.wikipedia.org/wiki/Infrastructure_as_code

Infrastructure as code (IaC) tools allow you to manage infrastructure with configuration files rather than through a graphical user interface. IaC allows you to build, change, and manage your infrastructure in a safe, consistent, and repeatable way by defining resource configurations that you can version, reuse, and share.

https://learn.hashicorp.com/tutorials/terraform/infrastructure-as-code





Advantages to adopt Terraform, engineering viewpoint

- Infrastructure as Code to reveal **the final state** for your infrastructure.

  → Precisely control the infrastructure as expectation.

- Reproducible infrastructure environments

  → Reduce cost for managing infrastructure consistency.

- Multi-clouds support

  → Easily integrate with different clouds in one project.

- Free

Advantages to adopt Terraform, business viewpoint

- **Free**

- Final state infrastructure as Code

  → Stability increase for complex services.

- Reproduceable infrastructure environments

  → Increase services delivery quality.

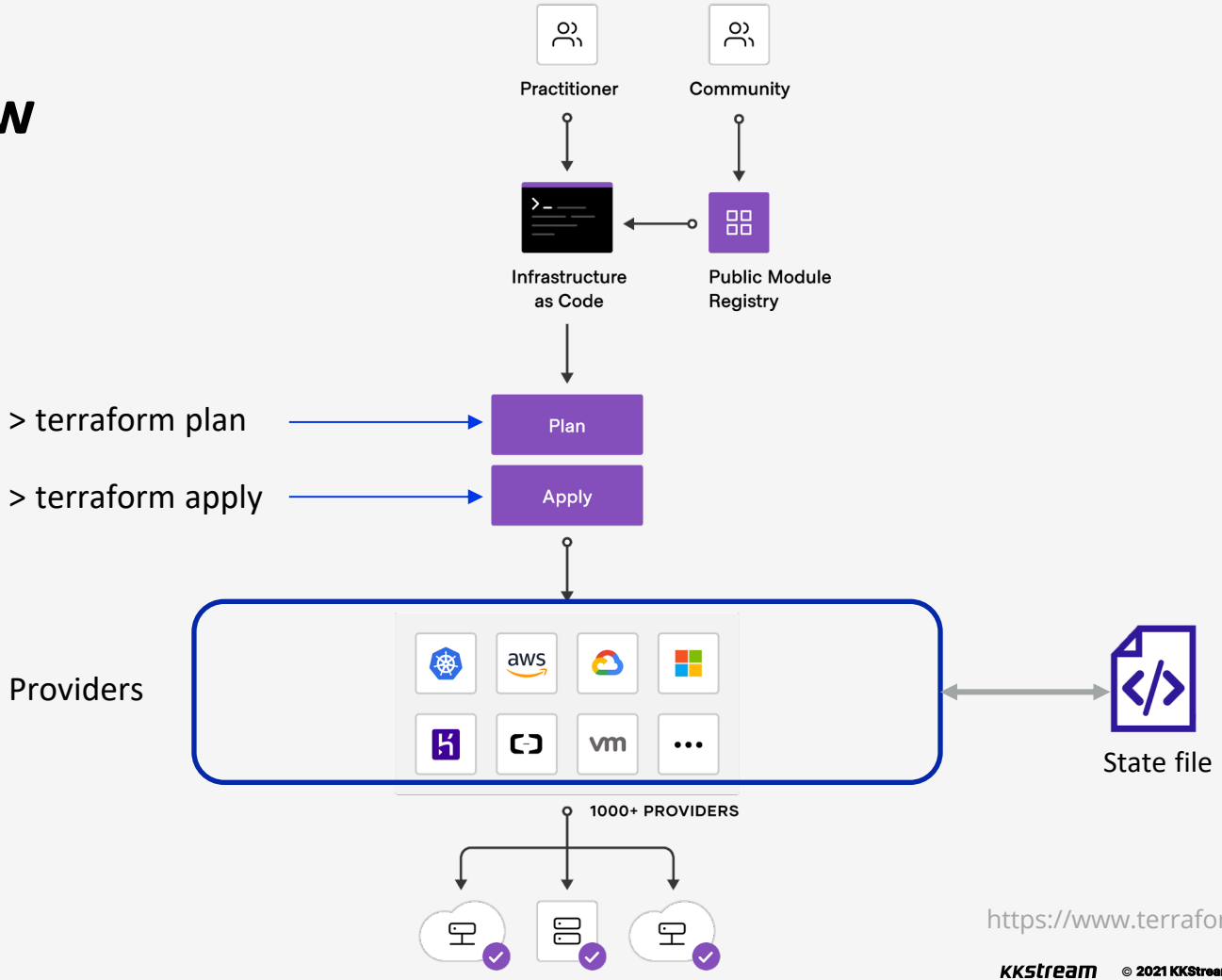
  → Reduce unknow risks for new components.

- Multi-clouds support

  → Agility support variety business requirements.

more can be found in Hasicorp site: https://www.hashicorp.com/solutions/infrastructure-provisioning

Overview



https://www.terraform.io/

Terraform install

5 mins

- Terraform: https://learn.hashicorp.com/tutorials/terraform/install-cli
- TFSwitch: https://tfswitch.warrensbox.com/

Terraform install

Mac OS

> brew tap hashicorp/tap

> brew install hashicorp/tap/terraform

> brew update

> brew upgrade hashicorp/tap/terraform

> terraform version

```
Terraform v1.0.5
on darwin_amd64
+ provider registry.terraform.io/hashicorp/aws v3.61.0
+ provider registry.terraform.io/hashicorp/external v2.1.0
+ provider registry.terraform.io/hashicorp/local v2.1.0
+ provider registry.terraform.io/hashicorp/null v3.1.0
+ provider registry.terraform.io/hashicorp/random v3.1.0
+ provider registry.terraform.io/hashicorp/template v2.2.0
+ provider registry.terraform.io/innovationnorway/tfvars v0.0.1
```

https://learn.hashicorp.com/tutorials/terraform/install-cli

Terraform install

Ubuntu

> sudo apt-get update && sudo apt-get install -y gnupg software-properties-common curl

> curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -

> sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) main"

> sudo apt-get update && sudo apt-get install terraform

> terraform version

```
Terraform v1.0.5
on darwin_amd64
+ provider registry.terraform.io/hashicorp/aws v3.61.0
+ provider registry.terraform.io/hashicorp/external v2.1.0
+ provider registry.terraform.io/hashicorp/local v2.1.0
+ provider registry.terraform.io/hashicorp/null v3.1.0
+ provider registry.terraform.io/hashicorp/random v3.1.0
+ provider registry.terraform.io/hashicorp/template v2.2.0
+ provider registry.terraform.io/innovationnorway/tfvars v0.0.1
```

https://learn.hashicorp.com/tutorials/terraform/install-cli

Terraform install

Windows (Manual installation)

- Download from
  https://www.terraform.io/downloads.html

- Add binary path into system `PATH`
  https://stackoverflow.com/questions/1618280/where-can-i-set-path-to-make-exe-on-windows

> terraform version

```
Terraform v1.0.5
on darwin_amd64
+ provider registry.terraform.io/hashicorp/aws v3.61.0
+ provider registry.terraform.io/hashicorp/external v2.1.0
+ provider registry.terraform.io/hashicorp/local v2.1.0
+ provider registry.terraform.io/hashicorp/null v3.1.0
+ provider registry.terraform.io/hashicorp/random v3.1.0
+ provider registry.terraform.io/hashicorp/template v2.2.0
+ provider registry.terraform.io/innovationnorway/tfvars v0.0.1
```

https://learn.hashicorp.com/tutorials/terraform/install-cli

Quick start

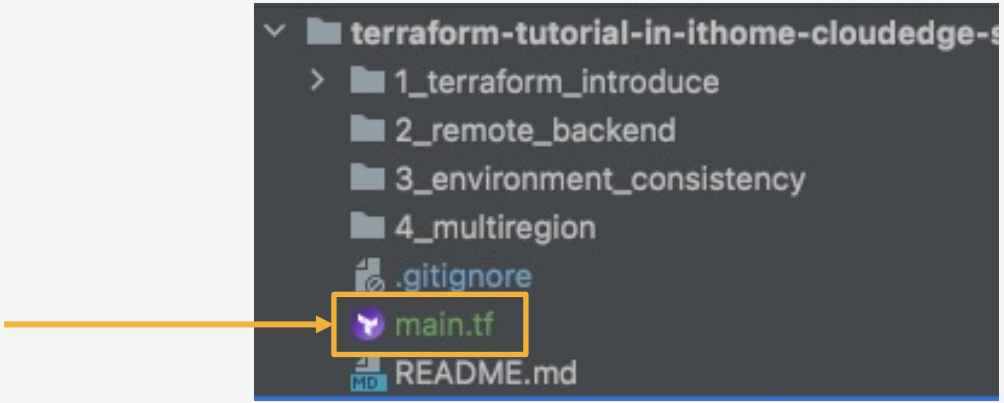
Terraform steps

1. aws configure → Set AWS credential in local
2. terraform init → Initialize Terraform project: download providers, sync state file
3. terraform plan → Diff resources
4. terraform apply → Apply diff

https://github.com/matrix0415/terraform-tutorial-in-ithome-cloudedge-summit

Quick start

Before start



Put your practice file `main.tf` in **root folder**.

Quick start

Start an EC2 instance

```
1  provider "aws" {
2    region = "ap-northeast-1"
3  }
4
5  resource "aws_instance" "instance" {
6    ami           = "ami-036d0684fc96830ca"
7    instance_type = "t3.micro"
8  }
9
10 output "id" {
11   value = aws_instance.instance.id
12 }
```

> terraform init

> terraform plan -out .plan

> terraform apply .plan

id = "your-aws-instance-id"

Quick start

Start an ec2 instance

> terraform plan -out .plan

```
  # aws_instance.test will be created
  + resource "aws_instance" "test" {
      + ami                                  = "ami-036d0684fc96830ca"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
```

~

```
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }
    }

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

Changes to Outputs:
  + id = (known after apply)
```

> terraform apply .plan

```
aws_instance.test: Creating...
aws_instance.test: Still creating... [10s elapsed]
aws_instance.test: Creation complete after 15s [id=i-0ee445ae118232f4e]

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

Outputs:

id = "i-0ee445ae118232f4e"
```

Quick start

Add a key for the instance

```
provider "aws" {
  region = "ap-northeast-1"
}

resource "aws_instance" "instance" {
  ami           = "ami-03600684fc96830ca"
  instance_type = "t3.micro"
  key_name      = aws_key_pair.deployer.key_name
}

resource "aws_key_pair" "deployer" {
  key_name   = "deployer-key"
  public_key = "ssh-rsa ..."
}

output "id" {
  value = aws_instance.instance.id
}
```

> terraform plan -out .plan

> terraform apply .plan

id = "your-aws-instance-id"

Quick start

Add a key for the instance

> terraform plan -out .plan

```
  # aws_instance.test must be replaced
-/+ resource "aws_instance" "test" {
      ~ arn                                  = "arn:aws:ec2:ap-northeast-1:
      ~ associate_public_ip_address          = true -> (known after apply)
      ~ availability_zone                    = "ap-northeast-1a" -> (known
      ~ cpu_core_count                       = 1 -> (known after apply)
      ~ cpu_threads_per_core                 = 2 -> (known after apply)

~

  # aws_key_pair.deployer will be created
  + resource "aws_key_pair" "deployer" {
      + arn         = (known after apply)
      + fingerprint = (known after apply)
      + id          = (known after apply)
      + key_name    = "deployer-key"
      + key_pair_id = (known after apply)
      + public_key  = "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCu0QOlwx92aaSF
GG6j7eTV4c6toHiyVIna1eC7gWQemf3eCmIJIdVJXzHLnUdnADFYsAIQ0UYeb3KD9psPF5rY8QI
t+hFCNviYOHIIBzIY+/jKIgyazX7+8Bewv0fIqr3IQJWnasCsBruChw0JZMrxwGjxRgvUfaRHxY
      + tags_all    = (known after apply)
    }

Plan: 2 to add, 0 to change, 1 to destroy.
```

> terraform apply .plan

```
aws_instance.test: Destroying... [id=i-0ee445ae118232f4e]
aws_instance.test: Still destroying... [id=i-0ee445ae118232f4e, 10s elapsed
aws_instance.test: Still destroying... [id=i-0ee445ae118232f4e, 20s elapsed
aws_instance.test: Still destroying... [id=i-0ee445ae118232f4e, 30s elapsed
aws_instance.test: Still destroying... [id=i-0ee445ae118232f4e, 40s elapsed
aws_instance.test: Still destroying... [id=i-0ee445ae118232f4e, 50s elapsed
aws_instance.test: Destruction complete after 52s
aws_key_pair.deployer: Creating...
aws_key_pair.deployer: Creation complete after 0s [id=deployer-key]
aws_instance.test: Creating...
aws_instance.test: Still creating... [10s elapsed]
aws_instance.test: Creation complete after 15s [id=i-0be9d910aaf312117]

Apply complete! Resources: 2 added, 0 changed, 1 destroyed.

Outputs:

id = "i-0be9d910aaf312117"
```

Quick start

Add a security group for the instance

```
...

resource "aws_instance" "instance" {
  ami                    = "ami-036d0684fc96830ca"
  instance_type          = "t3.micro"
  key_name               = aws_key_pair.deployer.key_name
  vpc_security_group_ids = [aws_security_group.sg.id]
}

...

resource "aws_security_group" "sg" {
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

...
```

> terraform plan -out .plan

> terraform apply .plan

id = "your-aws-instance-id"

Quick start

Add a security group for the instance

> terraform plan -out .plan

```
 # aws_instance.test must be replaced
-/+ resource "aws_instance" "test" {
      ~ arn                                  = "arn:aws:ec2:ap-northeast-1:
      ~ associate_public_ip_address          = true -> (known after apply)
      ~ availability_zone                    = "ap-northeast-1a" -> (known
      ~ cpu_core_count                       = 1 -> (known after apply)

      ~ vpc_security_group_ids               = [
          - "sg-0dba0a1a57a2195e8",
        ] -> (known after apply)

  # aws_security_group.test will be created
  + resource "aws_security_group" "test" {
      + arn                    = (known after apply)
      + description            = "Managed by Terraform"
      + egress                 = (known after apply)

      + vpc_id                 = (known after apply)
    }

Plan: 2 to add, 0 to change, 1 to destroy.
```

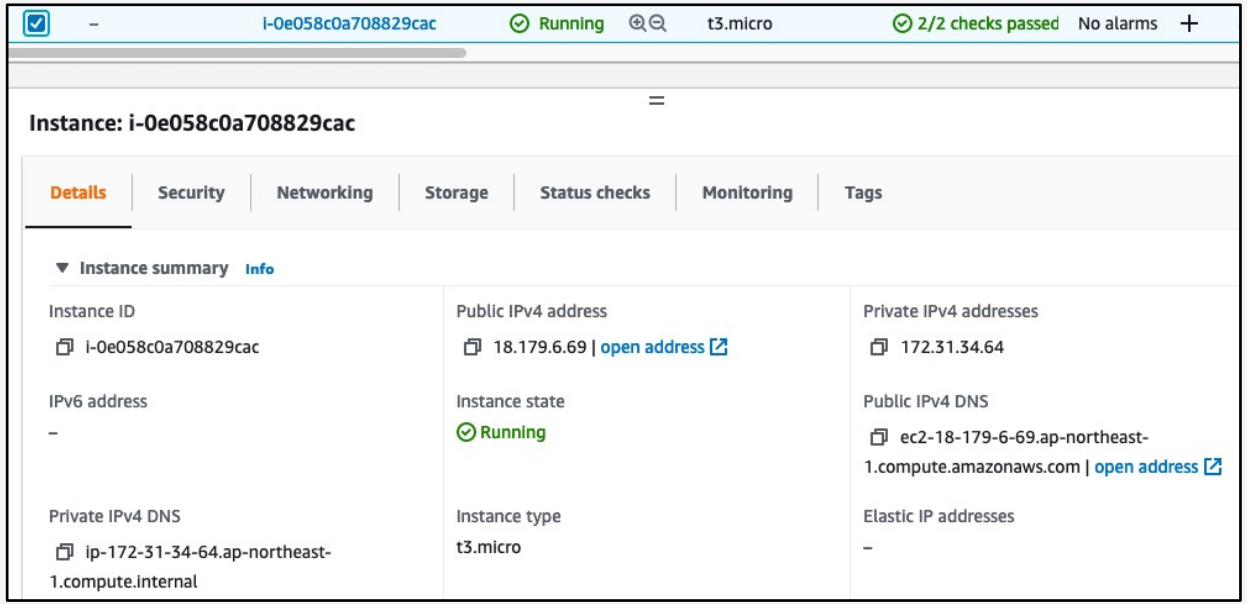
> terraform apply .plan

```
aws_instance.test: Destroying... [id=i-0ee445ae118232f4e]
aws_instance.test: Still destroying... [id=i-0ee445ae118232f4e, 10s elapsed
aws_instance.test: Still destroying... [id=i-0ee445ae118232f4e, 20s elapsed
aws_instance.test: Still destroying... [id=i-0ee445ae118232f4e, 30s elapsed
aws_instance.test: Still destroying... [id=i-0ee445ae118232f4e, 40s elapsed
aws_instance.test: Still destroying... [id=i-0ee445ae118232f4e, 50s elapsed
aws_instance.test: Destruction complete after 52s
aws_key_pair.deployer: Creating...
aws_key_pair.deployer: Creation complete after 0s [id=deployer-key]
aws_instance.test: Creating...
aws_instance.test: Still creating... [10s elapsed]
aws_instance.test: Creation complete after 15s [id=i-0be9d910aaf312117]

Apply complete! Resources: 2 added, 0 changed, 1 destroyed.

Outputs:

id = "i-0be9d910aaf312117"
```

Quick start

Login into the instance

> ssh -i `your-private-location` ubuntu@`your-public-ipv4-address`



Quiz

After “terraform apply”, show the public IPv4 address. (5 mins)

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

Outputs:

id = "i-0e058c0a708829cac"
ip = "18.179.6.69"
```

Reference: https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance

Quiz

After “terraform apply”, show the public IPv4 address. (5 mins)

```
...

output "ip" {
    value = aws_instance.instance.public_ip
}
```

> terraform plan -out .plan

> terraform apply .plan

id = "your-aws-instance-id"

ip = "18.179.6.69"

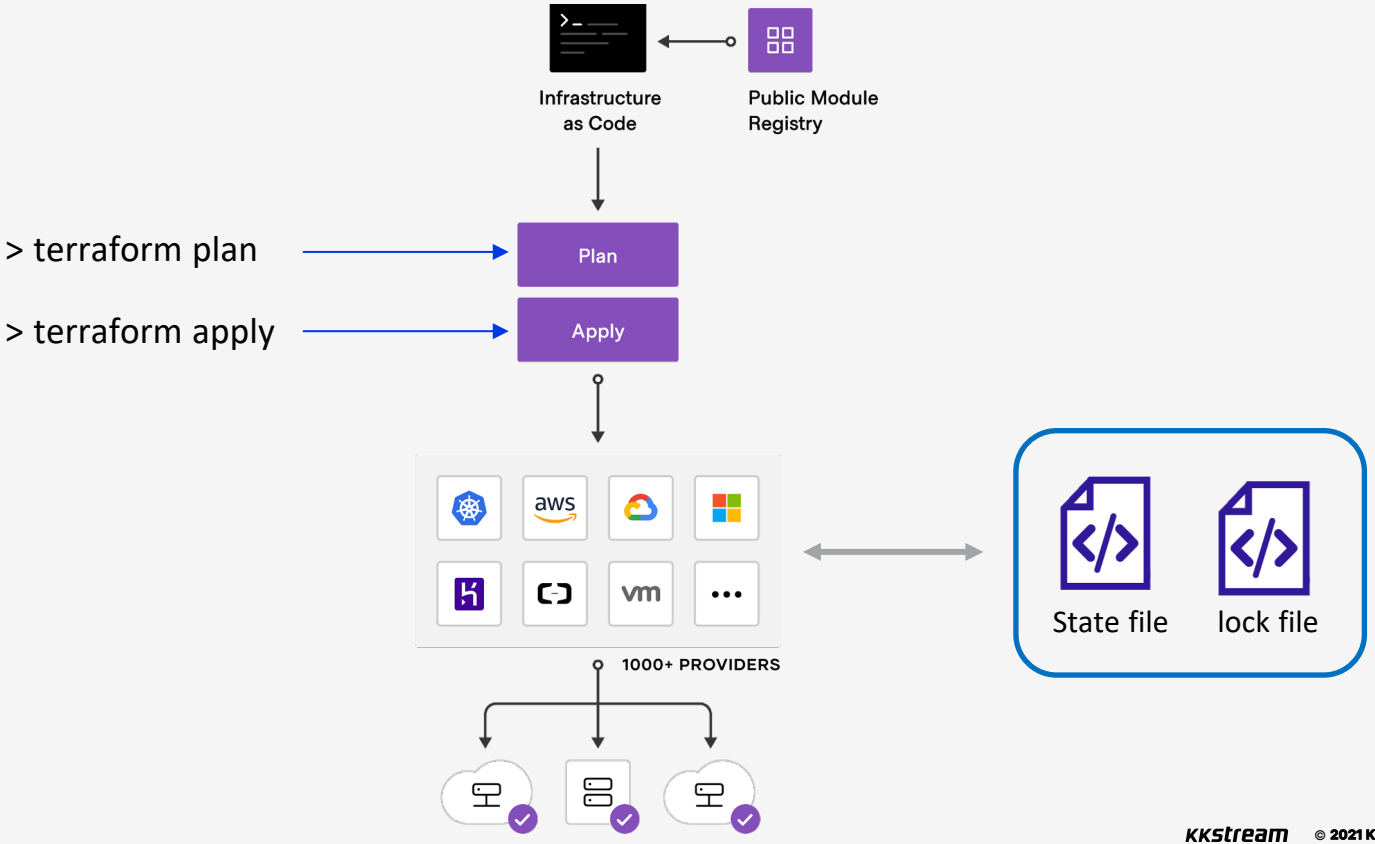
Summary

- Infrastructure as Code
- Terraform advantages in engineering side / business side
- Terraform entry tutorial
    - terraform init
    - terraform plan
    - terraform apply

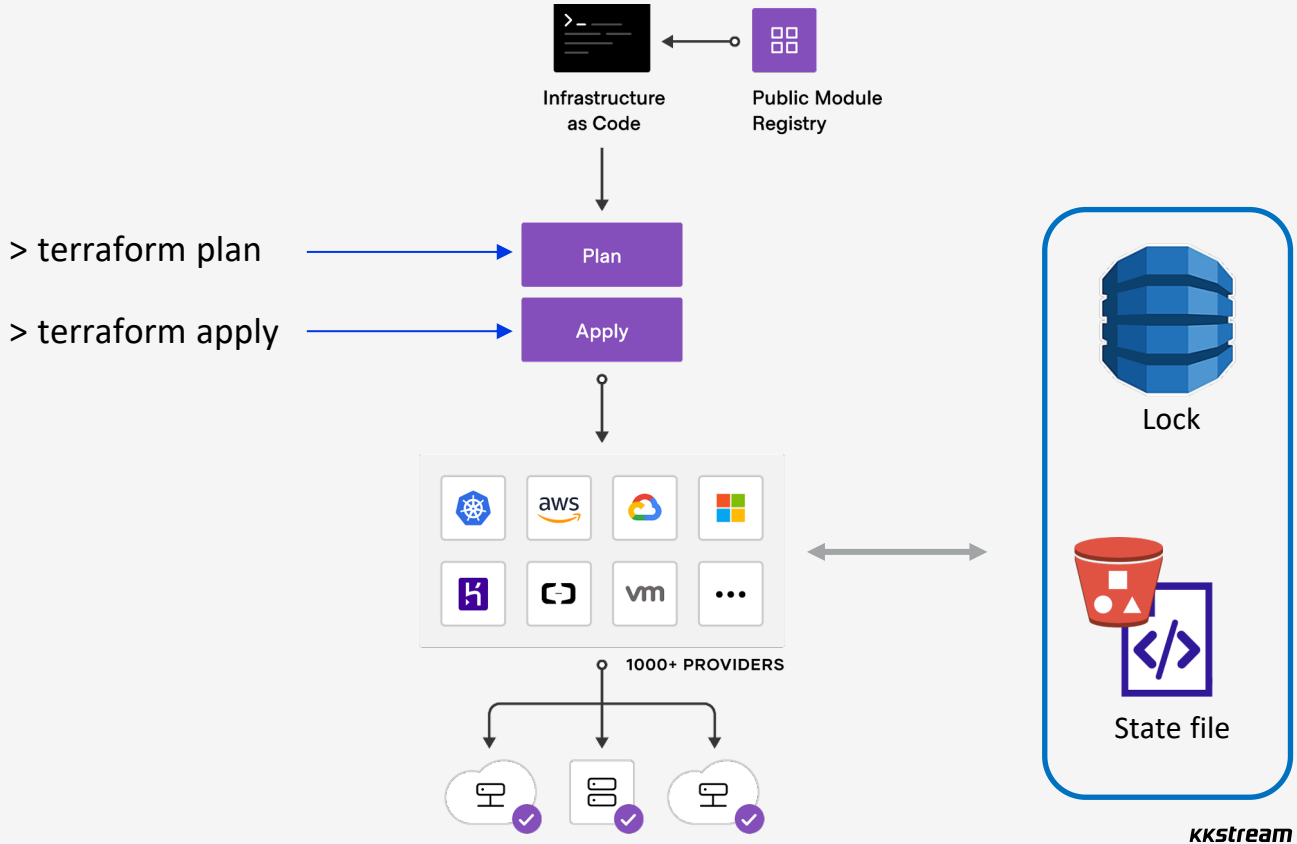
Feature 1

Remote backend

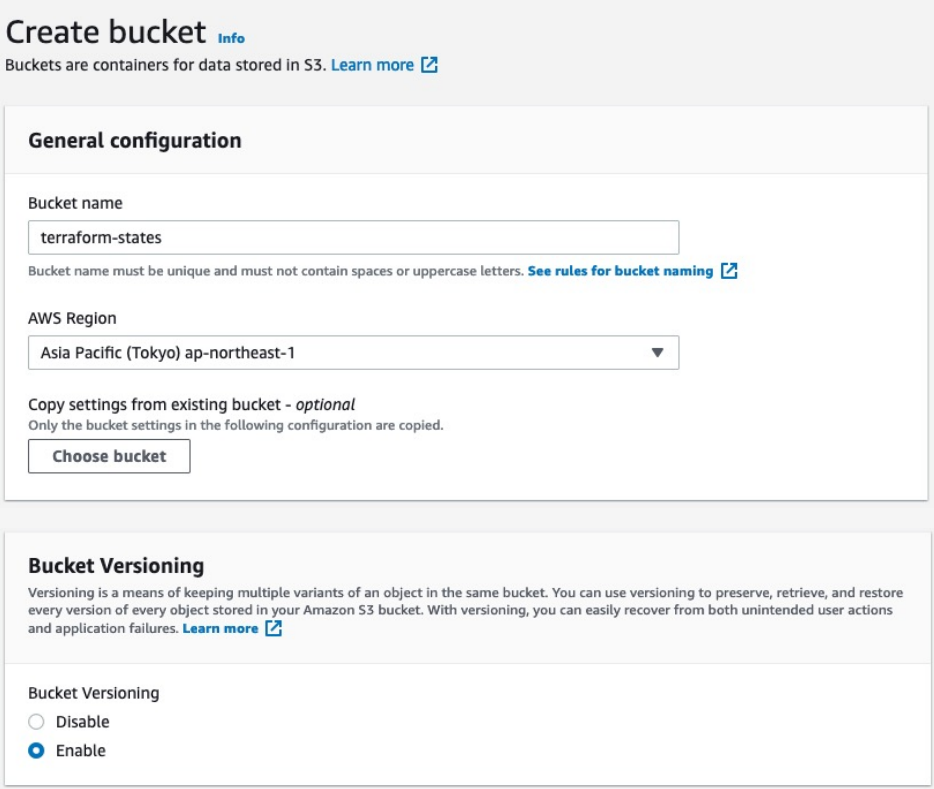
Local mode



S3 mode



Create S3 bucket

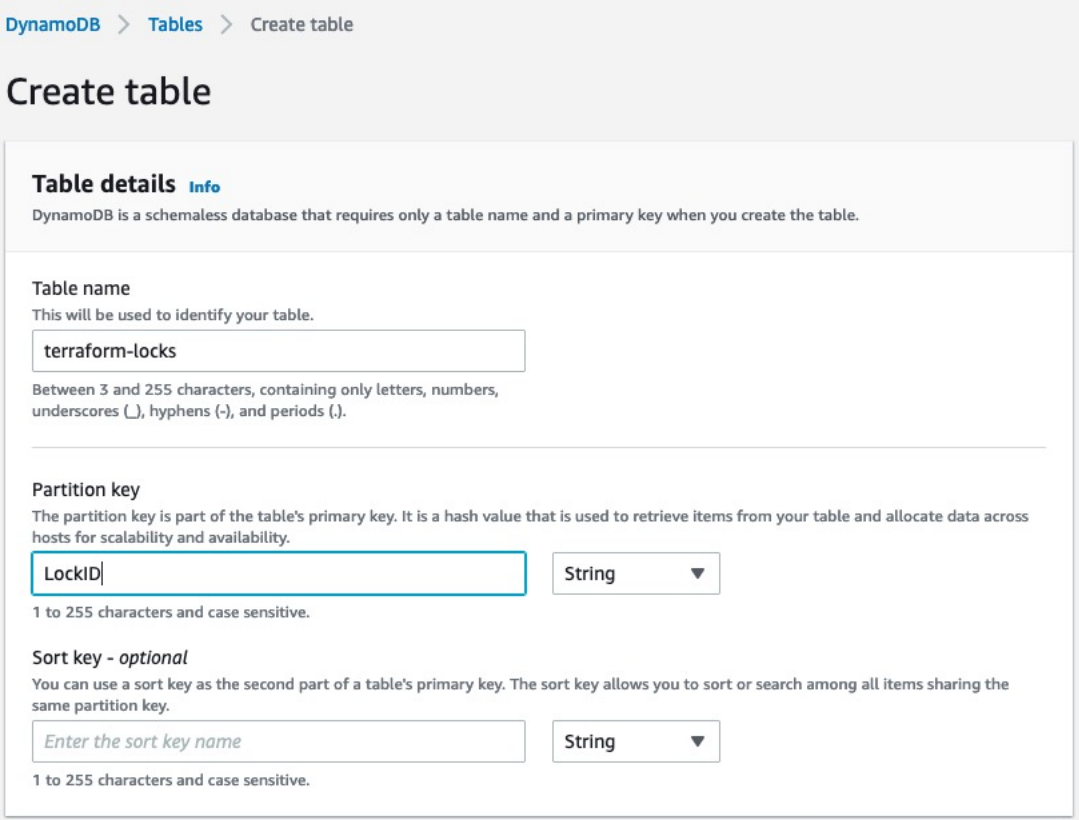


- Create a S3 bucket in Tokyo

  (Bucket name must be a global unique one)

- Enable bucket versioning

Create DynamoDB



- Create a DynamoDB table.
- Partition Key must be `LockID` in string type.
- In real environment, use on-demand capacity mode for saving cost.

Change Terraform backend

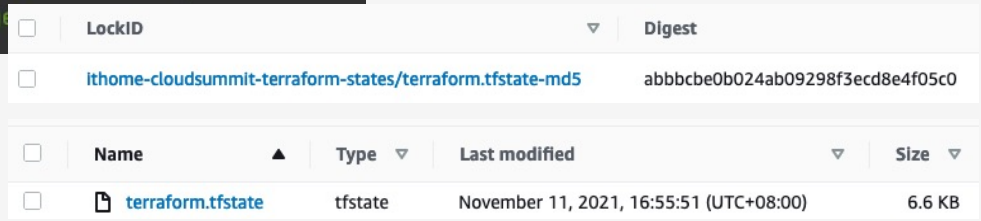
```
terraform {
    backend "s3" {
        region         = "ap-northeast-1"
        bucket         = "your-bucket-name"
        key            = "terraform.tfstate"
        dynamodb_table = "your-table-name"
    }
}

...
```

> terraform init

Change Terraform backend

> terraform init

```
Initializing the backend...
Do you want to copy existing state to the new backend?
  Pre-existing state was found while migrating the previous "local" backend to the
  newly configured "s3" backend. No existing state was found in the newly
  configured "s3" backend. Do you want to copy this state to the new "s3"
  backend? Enter "yes" to copy and "no" to start with an empty state.

  Enter a value: yes


Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration change
```



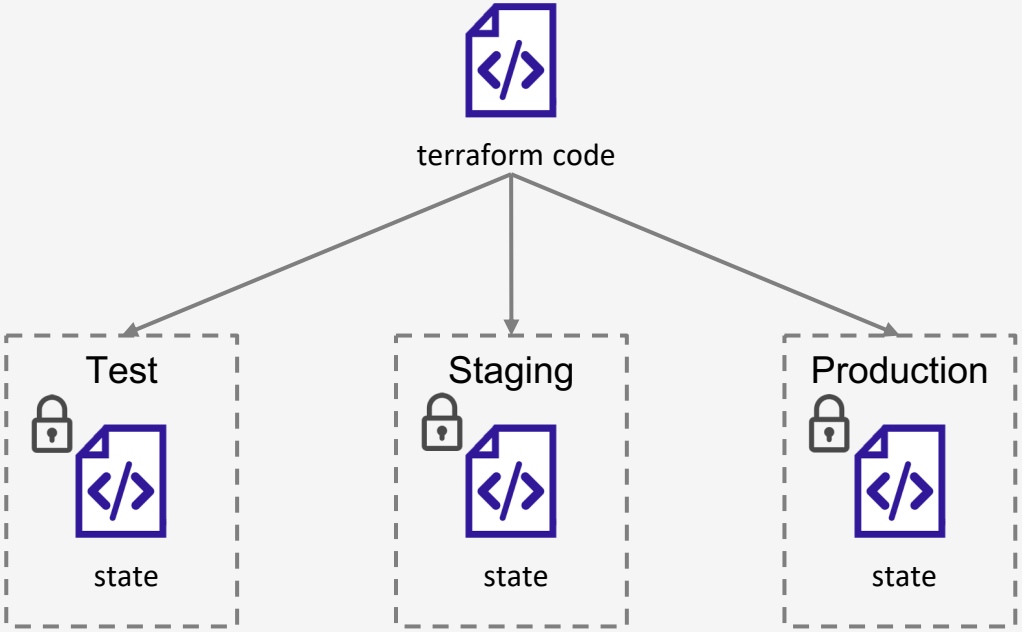
Summary

- From local to S3 Backend

Feature 2

Environment consistency

Workspace



Workspace

- terraform workspace list
- terraform workspace new
- terraform workspace select
- terraform workspace show
- terraform workspace delete

Change code to deploy on different workspace

```
...

variable "instance_type" {
  type    = string
  default = "t3.micro"
}

resource "aws_instance" "instance" {
  ami                    = "ami-036d0684fc96830ca"
  instance_type          = var.instance_type
  key_name               = aws_key_pair.deployer.key_name
  vpc_security_group_ids = [aws_security_group.sg.id]
  tags = {
    Name = terraform.workspace
  }
}

resource "aws_key_pair" "deployer" {
  key_name   = "deployer-key-${terraform.workspace}"
  public_key = "ssh-rsa ..."
}

resource "aws_security_group" "sg" {
  name = terraform.workspace
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

...
```

terraform.workspace

> terraform plan -out .plan -var-file default.tfvars

> terraform apply .plan

id = "your-aws-instance-id"

ip = "18.179.6.69"

.tfvars

main.tf

```
...

variable "instance_type" {
  type    = string
  default = "t3.micro"
}

resource "aws_instance" "instance" {
  ami                    = "ami-036d0684fc96830ca"
  instance_type          = var.instance_type
  key_name               = aws_key_pair.deployer.key_name
  vpc_security_group_ids = [aws_security_group.sg.id]
  tags = {
    Name = terraform.workspace
  }
}

...
```

default.tfvars

```
instance_type = "t3.micro"
```

production.tfvars

```
instance_type = "t3.small"
```

New workspace, Plan, Apply

> terraform workspace new production

```
Created and switched to workspace "production"!

You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "terraform plan" Terraform will not see any existing state
for this configuration.
```

> terraform plan -out .plan -var-file production.tfvars

> terraform apply .plan

id = "your-aws-instance-id"

ip = "18.179.6.69"

Destroy

```
          - to_port                   = 22
        },
      ] -> null
    - name                   = "terraform-20211111085136359800000001" -> null
    - name_prefix            = "terraform-" -> null
    - owner_id               = "641891449893" -> null
    - revoke_rules_on_delete = false -> null
    - tags                   = {} -> null
    - tags_all               = {} -> null
    - vpc_id                 = "vpc-02158dd19612a74f6" -> null
  }

Plan: 0 to add, 0 to change, 3 to destroy.

Changes to Outputs:
  - id = "i-0fe8bc47b7f3445b6" -> null
  - ip = "13.230.152.178" -> null

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
```

> terraform destroy -var-file production.tfvars

> terraform workspace select default

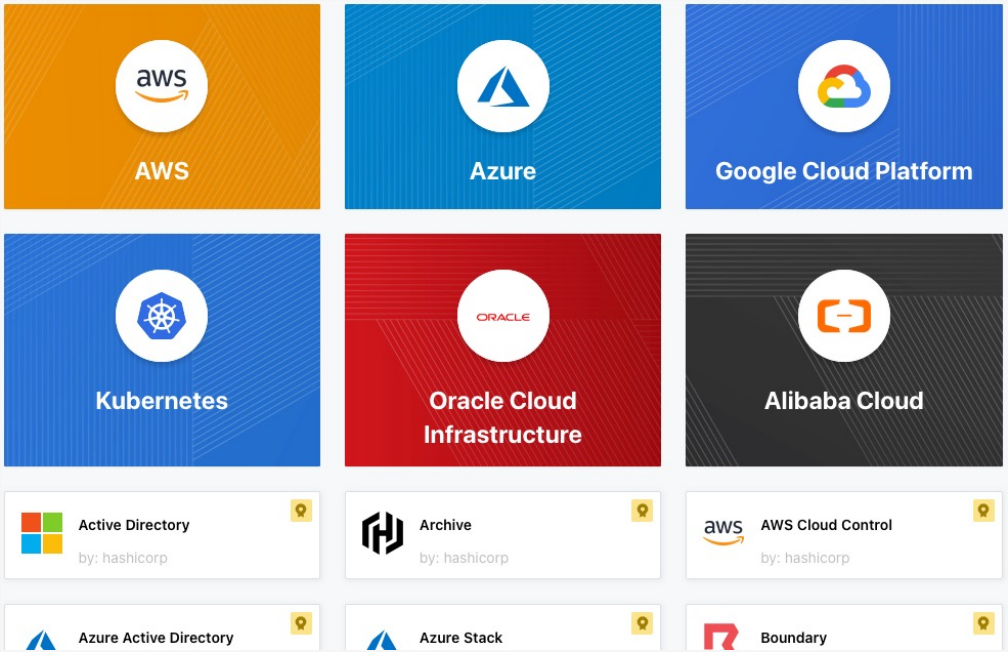
Summary

- Workspace
- terraform.workspace variable
- .tfvars
- New workspace, Plan, Apply
- Destroy

Feature 3

Multi-cloud/region in practice

Terraform providers registry



https://registry.terraform.io/browse/providers

Multiple providers

```
...

provider "aws" {
  region = "ap-northeast-1"
}

provider "aws" {
  alias  = "southeast"
  region = "ap-southeast-1"
}

...

resource "aws_instance" "instance_southeast" {
  ami                    = "ami-036d0684fc96830ca" // Ubuntu 20.04 LTS
  instance_type          = var.instance_type
  key_name               = aws_key_pair.deployer_southeast.key_name
  vpc_security_group_ids = [aws_security_group.sg_southeast.id]
  tags = {
    Name = terraform.workspace
  }
  provider = aws.southeast
}

...
```

> terraform init

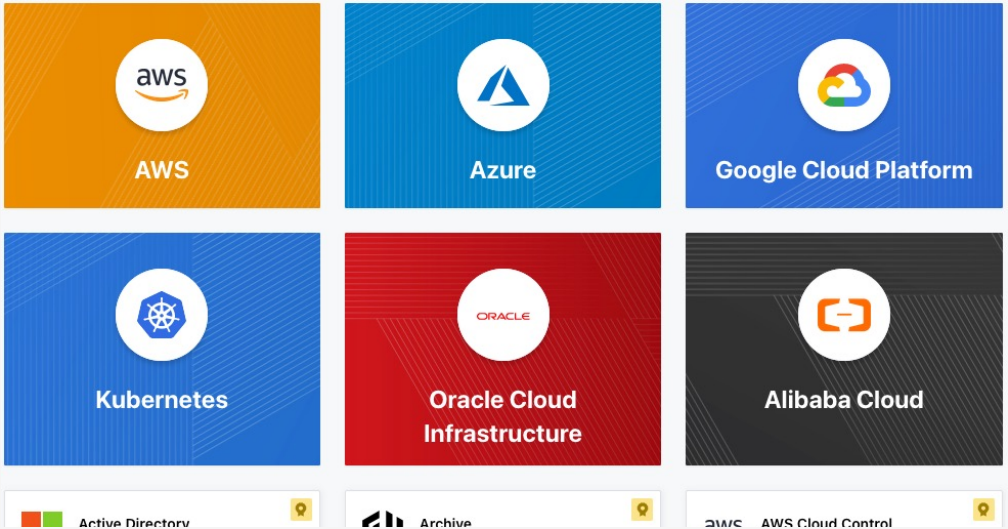
> terraform plan -out .plan

> terraform apply .plan

> terraform destroy

Summary

- With different providers, you can provision in different region.
- Cross service is feasible to integrate. Ex: Aws + Azure



Feature 4

Module

Local module

```
variable "instance_type" {
  type = string
}

variable "public_key" {
  type = string
}

...

resource "aws_instance" "instance" {
  ...
}

resource "aws_key_pair" "deployer" {
  ...
}

resource "aws_security_group" "sg" {
  ...
}

...
```



```
module "ec2" {
  source        = "./ec2"
  instance_type = var.instance_type
  public_key    = var.public_key
}
```

Usage

```
...

// ap-northeast-1
module "ec2" {
  source        = "./ec2"
  ami_id        = "ami-036d0684fc96830ca"
  instance_type = var.instance_type
  public_key    = var.public_key
}


// ap-southeast-1
module "ec2_southeast" {
  source        = "./ec2"
  ami_id        = "ami-0fed77069cd5a6d6c"
  instance_type = var.instance_type
  public_key    = var.public_key
  providers     = {
    aws = aws.southeast
  }
}

...
```

> terraform init

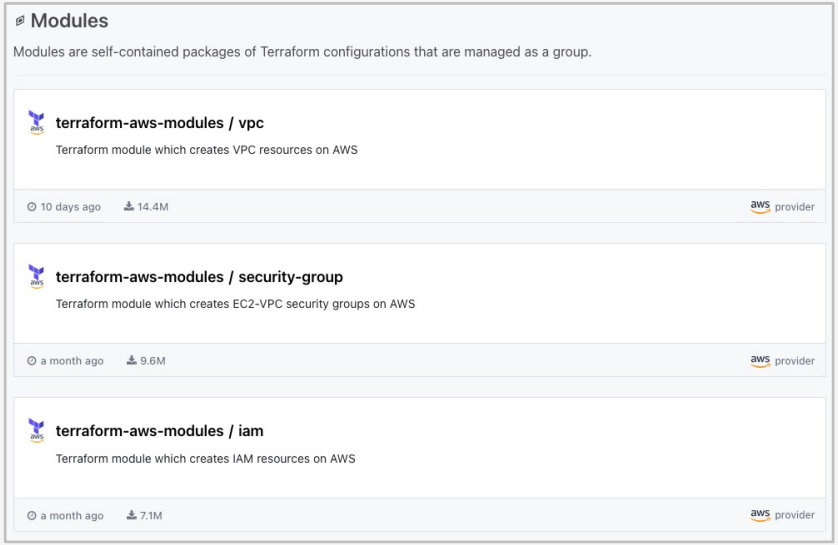
> terraform plan -out .plan

> terraform apply .plan

> ssh ubuntu@`your-public-ipv4-address`

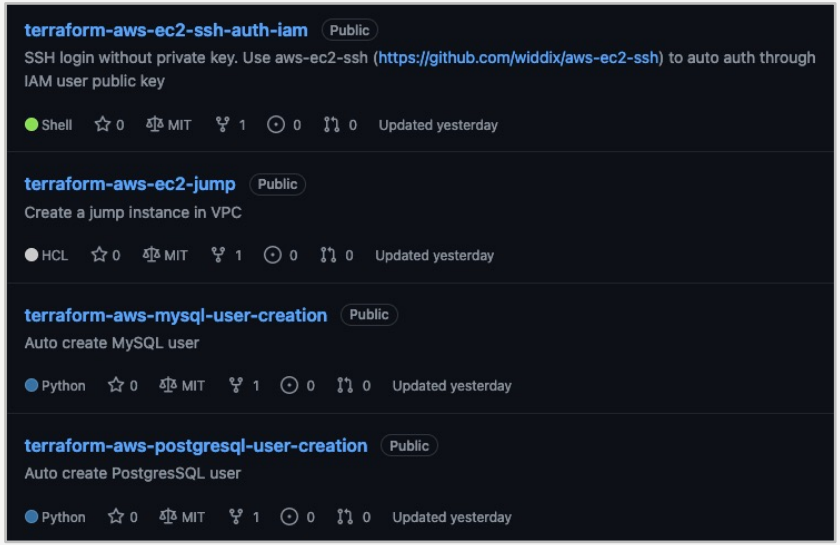
> terraform destroy

Remote module



Terraform module registry

https://registry.terraform.io/browse/modules



KKStream open source modules

https://github.com/kkstream

Introduce KKStream modules

- terraform-aws-ec2-jump

  https://github.com/KKStream/terraform-aws-ec2-jump

  Provision a jump server in VPC.

- terraform-aws-ec2-ssh-auth-iam

  https://github.com/KKStream/terraform-aws-ec2-ssh-auth-iam

  Use `aws-ec2-ssh` package to sync IAM user & authorize user login through IAM user SSH public key.

Usage

```
...

module "ec2" {
  source                      = "git::https://github.com/KKStream
/terraform-aws-ec2-jump?ref=v0.0.1"
  ami                         = "ami-036d0684fc96830ca"
  instance_type               = "t3.micro"
  project_name                = "ithome"
  subnet_id                   = aws_default_subnet.subnet.id
  vpc_id                      = aws_default_vpc.vpc.id
  allow_ssh_access_cidrs      = ["0.0.0.0/0"]
  termination_protection      = false
  enabled_egress_allow_http   = true
  enabled_egress_allow_https  = true
  enabled_egress_allow_ntp    = true
  user_data_base64            =
module.ec2_auth.ec2_user_data_base64
}

module "ec2_auth" {
  source                       = "git::https://github.com
/KKStream/terraform-aws-ec2-ssh-auth-iam?ref=v0.0.1"
  ec2_role_id                  = module.ec2.role_id
  allow_login_iam_group_names = [aws_iam_group.group.name]
}

...
```

> terraform init

> terraform plan -out .plan

> terraform apply .plan

(wait for 3 minutes)

> ssh ithome-user@`your-public-ipv4-address`

> terraform destroy

Summary

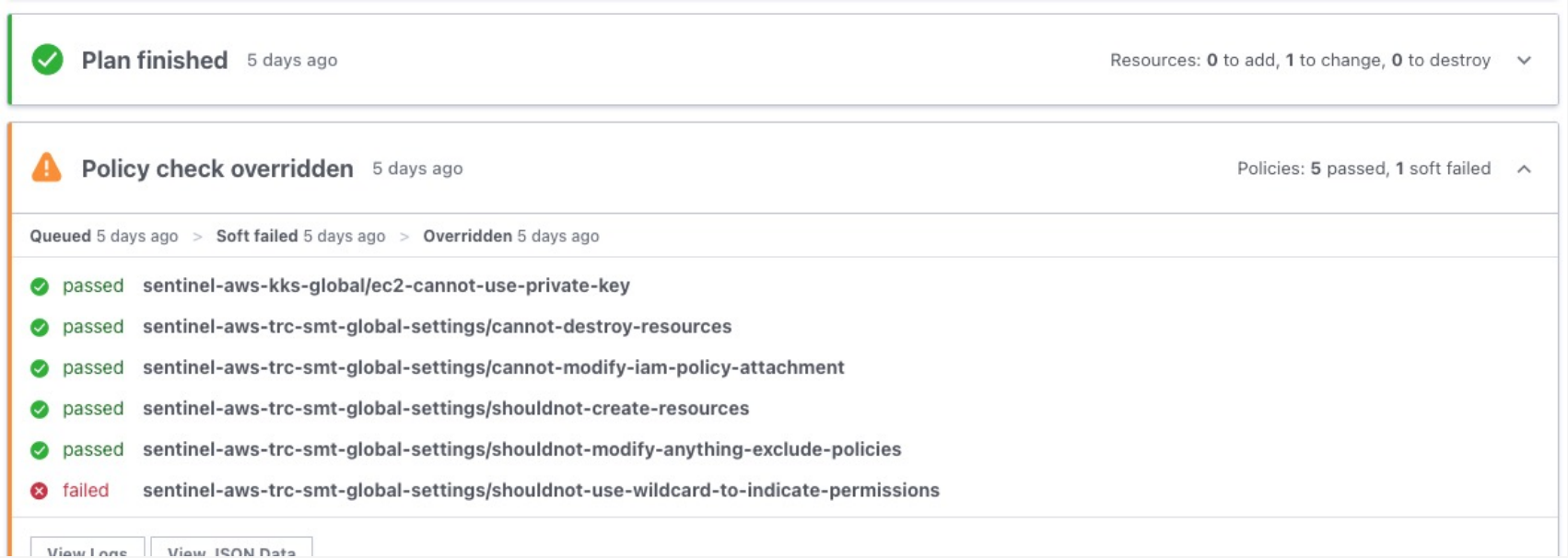
- Infrastructure encapsulation for functions/features/services
- Local module
- Remote module
    - From Git
    - From Terraform module registry

Premium Terraform services

Terraform cloud / Terraform enterprise

- Infrastructure CI / CD automation
- State version control
- Private module registry

Infrastructure governance



Summary

Summary

- Terraform introduction
  Advantages for using Terraform & Basic syntax/manipulation
- Feature 1: Remote backend
  S3 backend → Enable multiple developers to work together.
- Feature 2: Environment consistency
  Duplicate the same infrastructure to different development environment.
- Feature 3: Multi-clouds/regions
  Cross-region/Cross-cloud infrastructure provisioning and management.
- Feature 4: Modules
  Package for Terraform resources → Encapsulate for functions/features/services
- Premium Terraform services
  Advanced infrastructure management

HashiCorp

Terraform 線上研討會

如何以最快的方式 優化雲環境？

特別邀請 KKstream 分享實戰經驗

01 DEC 2021 | 2PM

https://hashicorp.zoom.us/webinar/register/WN_GaeG3cOGSvWvPrynIu5Jhg

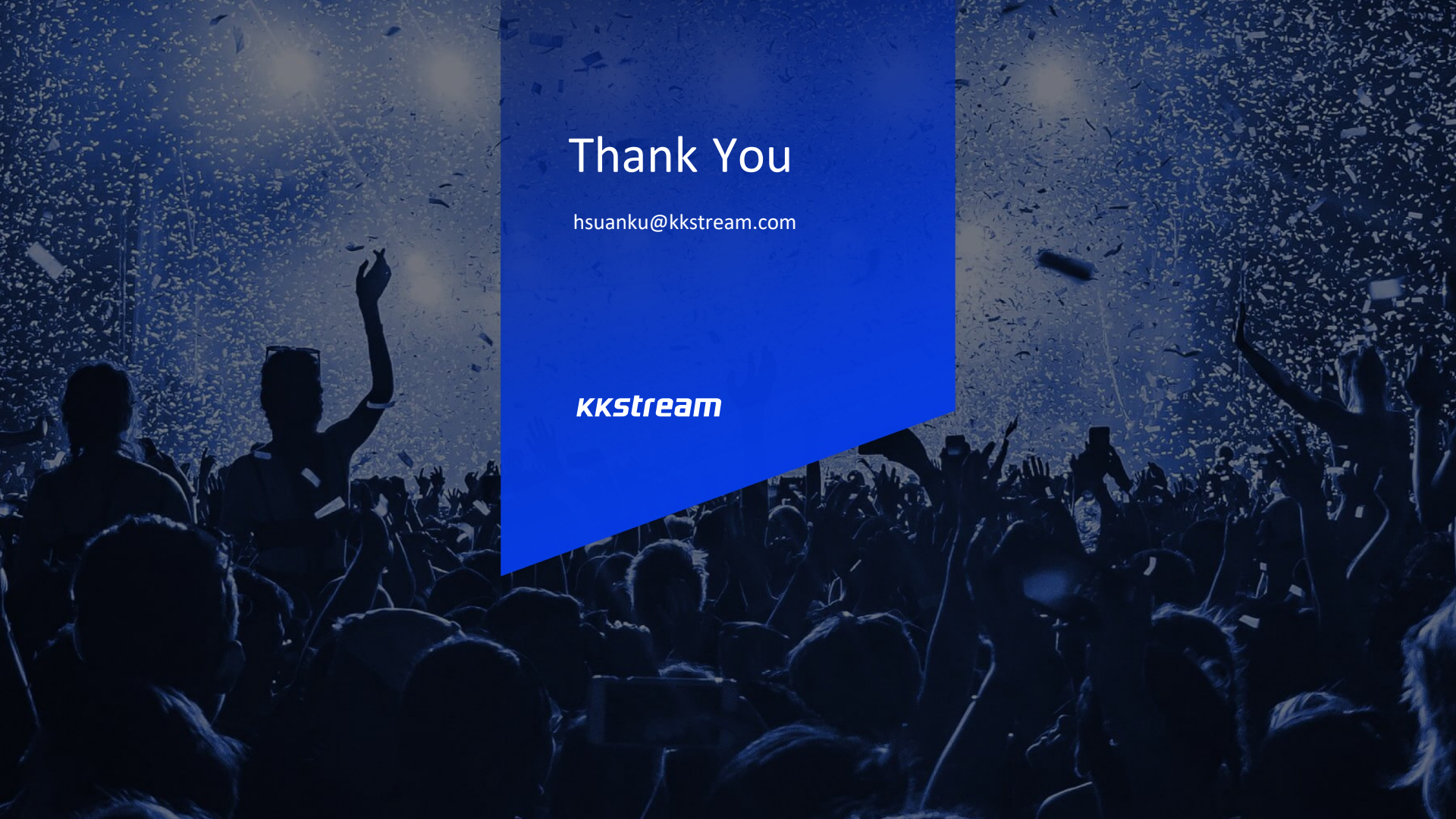
主辦單位 BROBRIDGE

Q & A

Remember – AWS Credit

- Credit: USD 25
- Survey: https://amazonmr.au1.qualtrics.com/jfe/form/SV_ahKlyIL3fO3EJdH
- Q3 fill in: 11/16 Cloud Summit

- Submit the survey today (11/16)
- Receive the credit in your email, submit the credit code to your AWS account.


Thank You
hsuanku@kkstream.com
KKstream