



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

Step into Terraform

Terraform experiences from KKStream

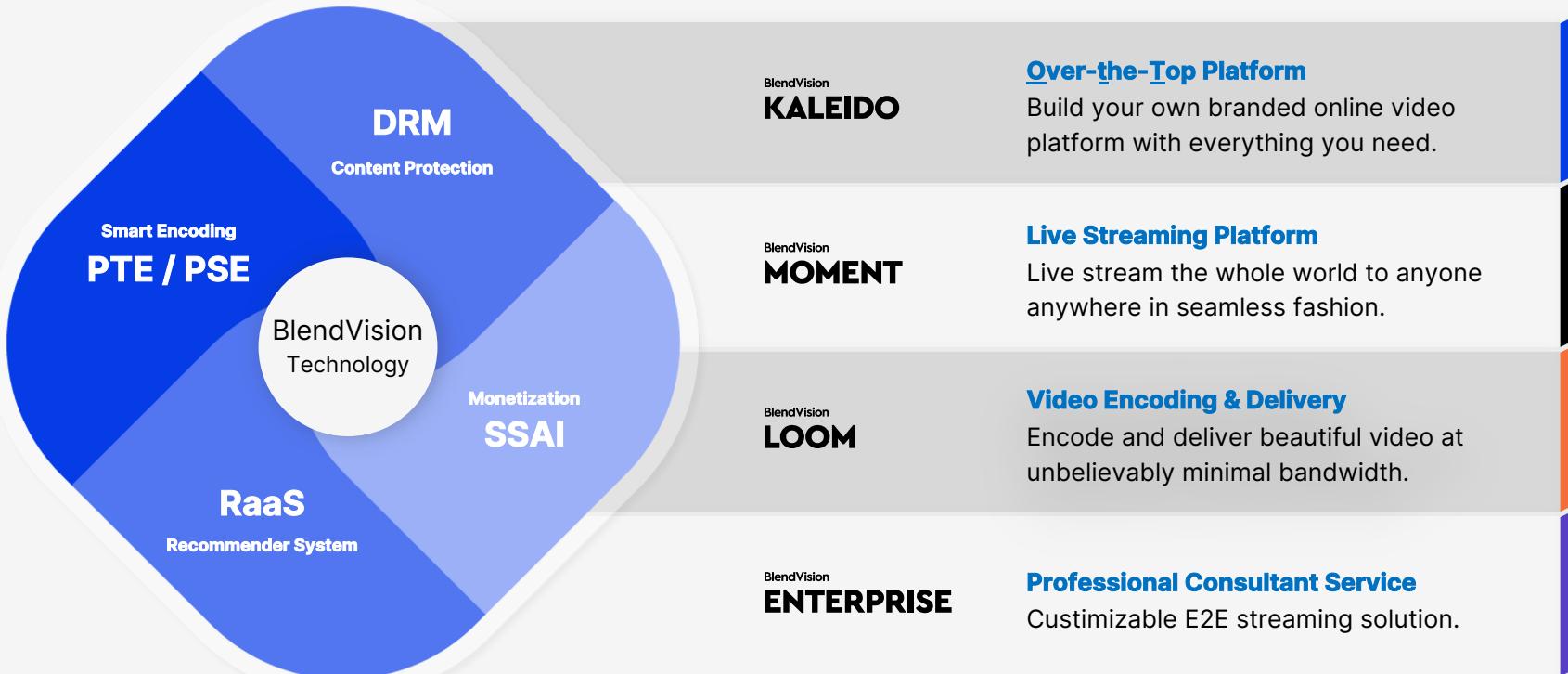


KKstream

Enabling **seamless** online
video experience since 2016



KKStream – BlendVision Products & 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

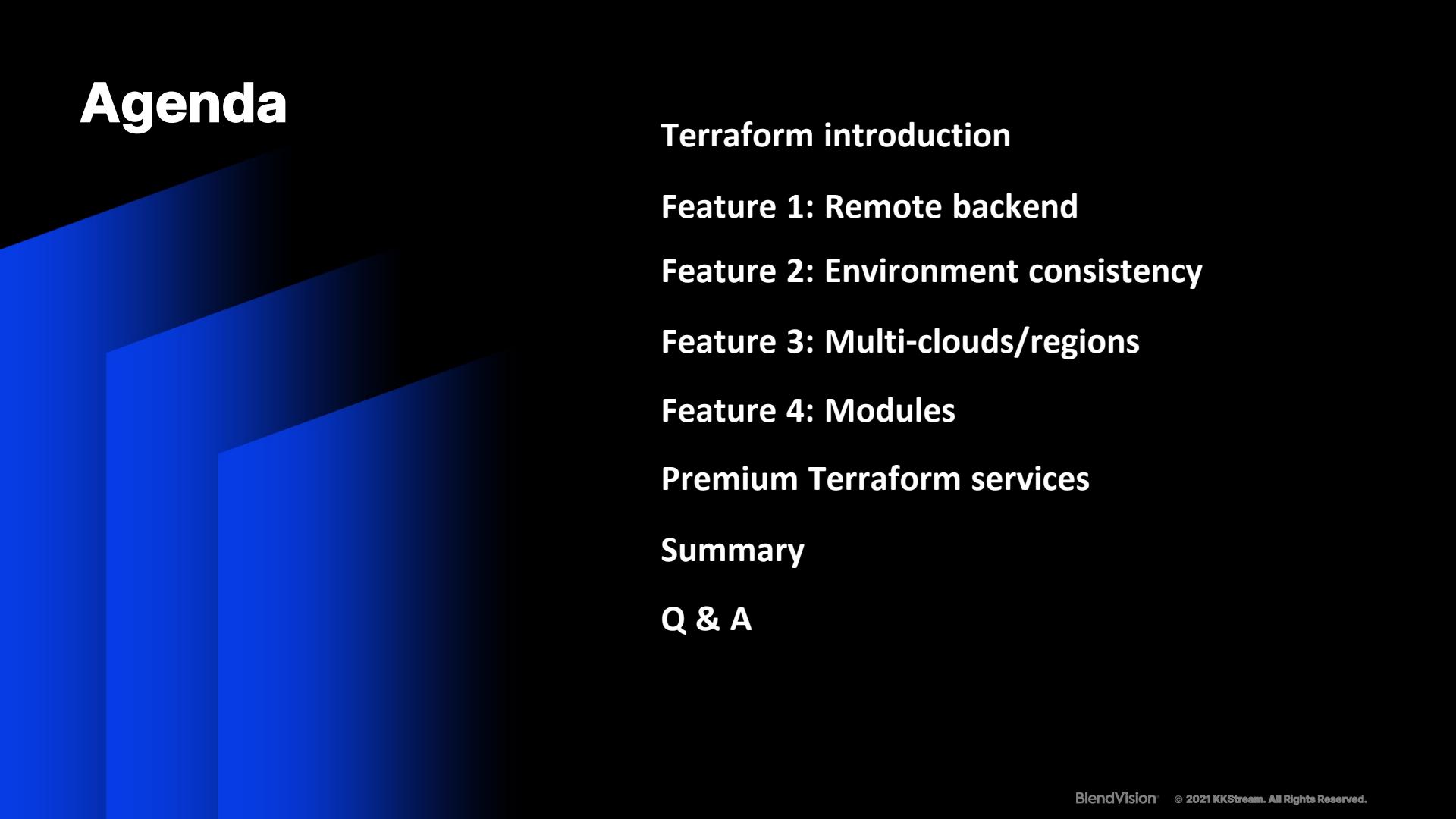
- Credit: USD 25
- Survey: https://amazonmr.au1.qualtrics.com/jfe/form/SV_ahKlyIL3fO3EJdH
- Q3 fill in: **11/16 Cloud Summit**

3. 歡迎留言告訴我們您們希望未來聽到的內容，或其他寶貴意見



- 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



WIKIPEDIA
The Free Encyclopedia

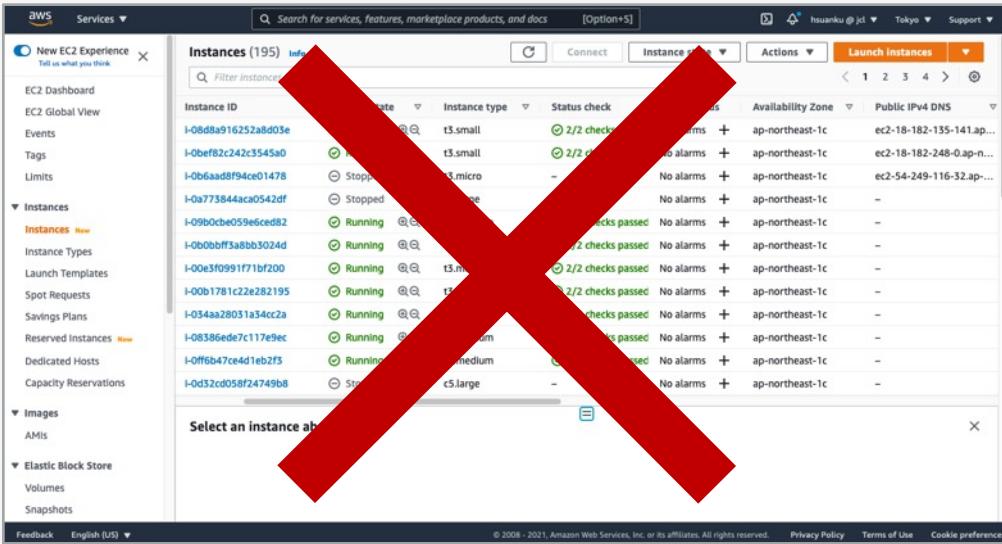
Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

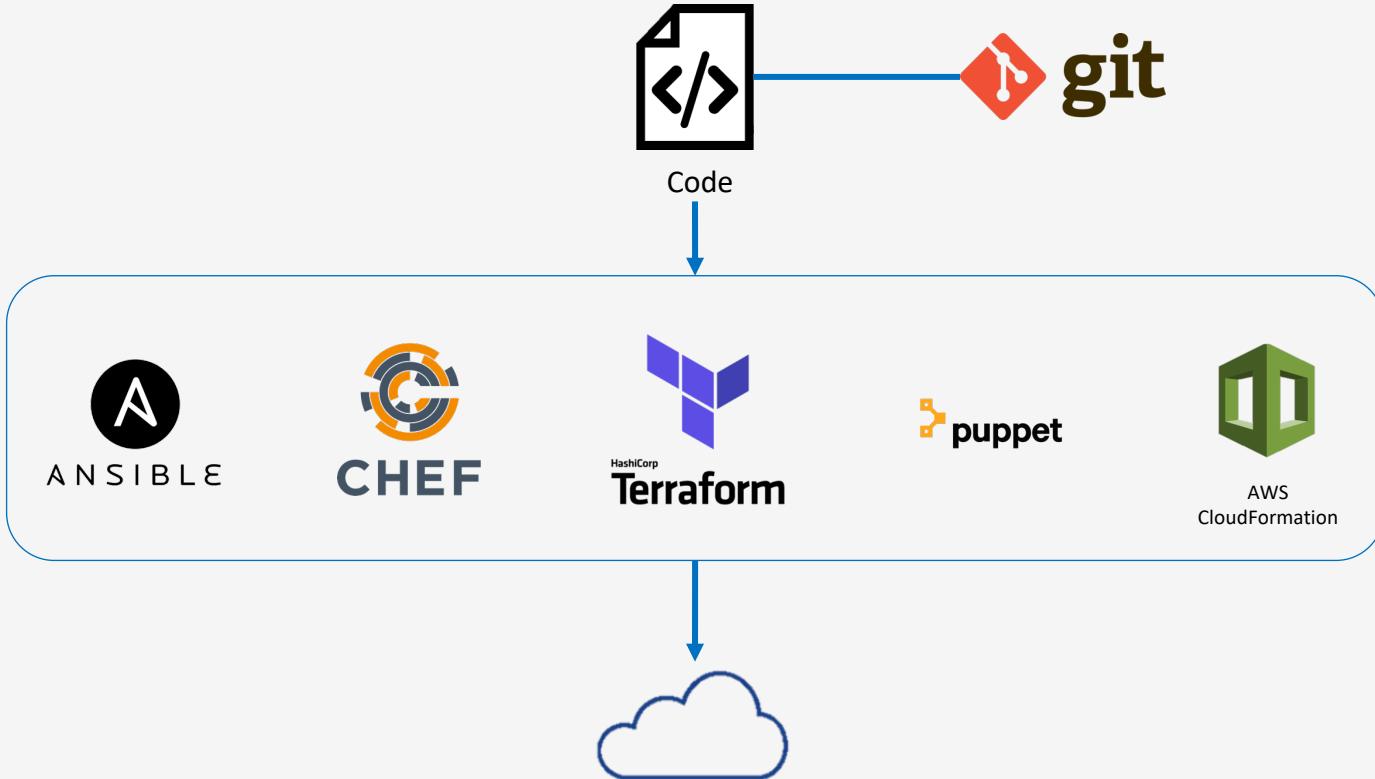
https://en.wikipedia.org/wiki/Infrastructure_as_code

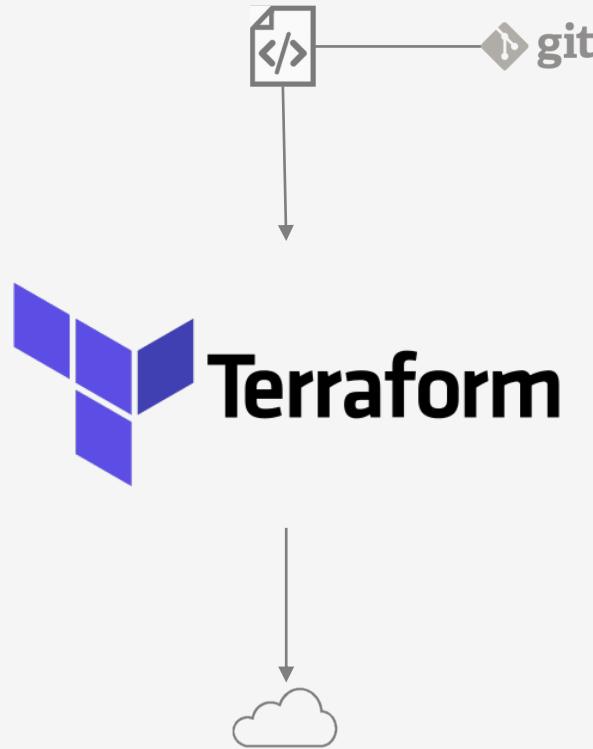


Infrastructure as code (IaC) tools allow you to manage infrastructure with configuration files rather than through a graphical user interface. IaC allows you to build, change, and manage your infrastructure in a safe, consistent, and repeatable way by defining resource configurations that you can version,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

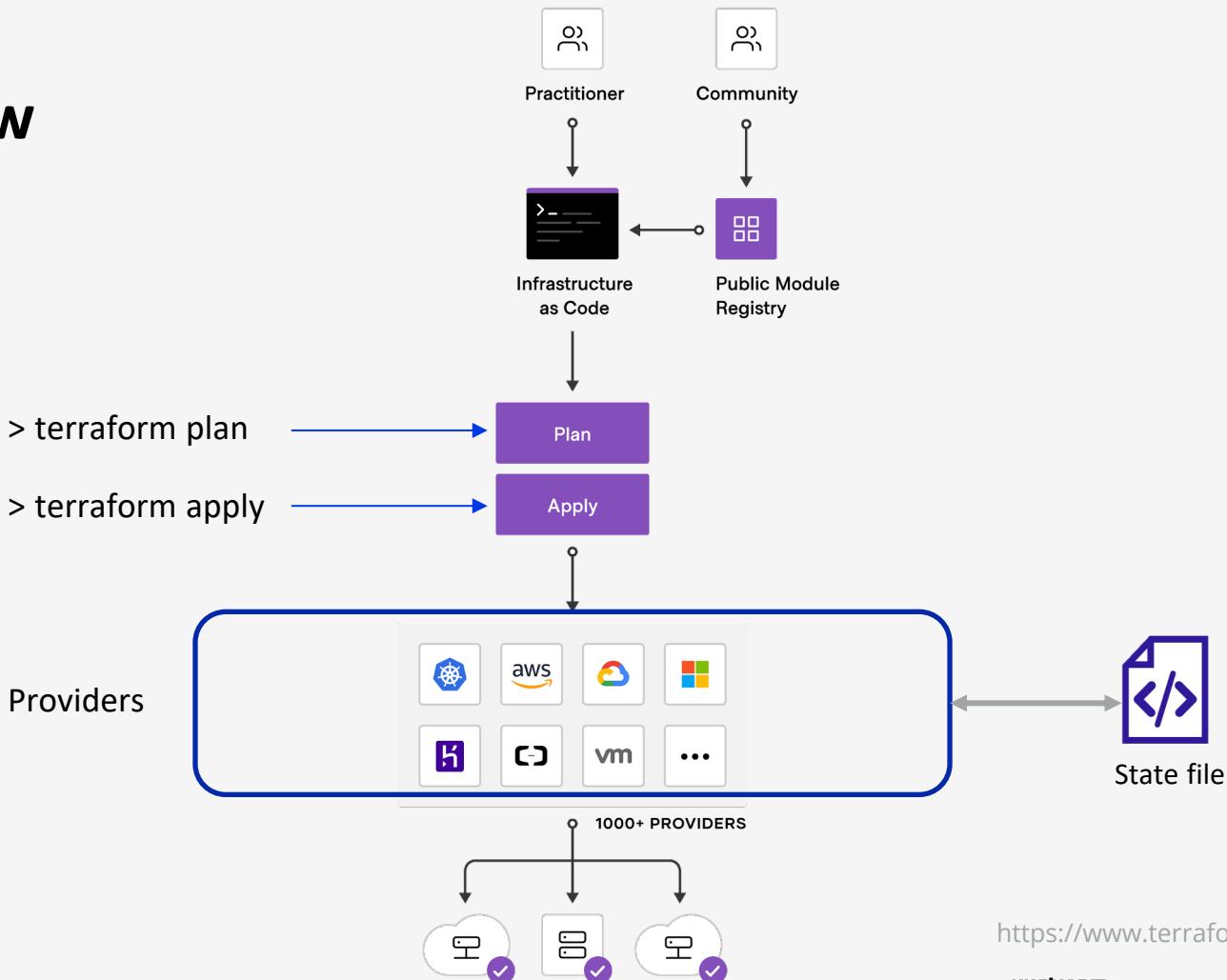
Advantages to adopt Terraform, business viewpoint

- 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 Hashicorp site: <https://www.hashicorp.com/solutions/infrastructure-provisioning>



Overview



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

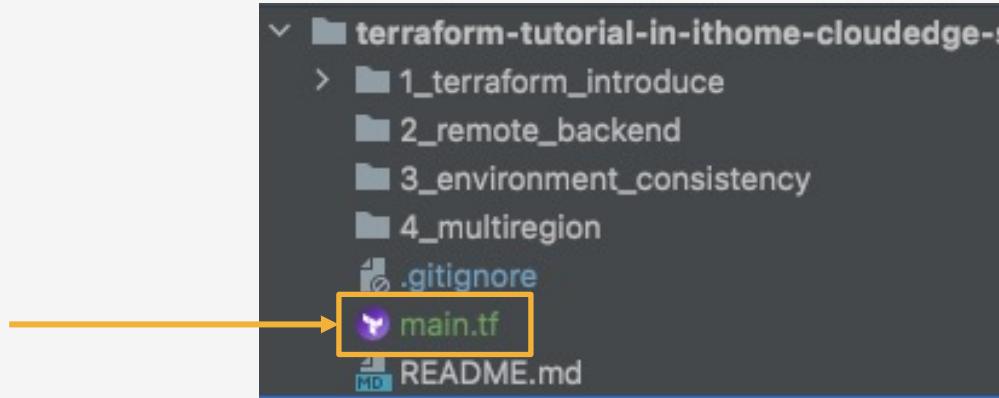
1. **aws configure** → Set AWS credential in local
2. **terraform init** → Initialize Terraform project: download providers, sync state file
3. **terraform plan** → Diff resources
4. **terraform apply** → Apply diff



<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

```
1 provider "aws" {
2   region = "ap-northeast-3"
3 }
4
5 resource "aws_instance" "instance" {
6   ami           = "ami-03608684fc96838ca"
7   instance_type = "t3.micro"
8   key_name      = aws_key_pair.deployer.key_name
9 }
10
11 resource "aws_key_pair" "deployer" {
12   key_name      = "deployer-key"
13   public_key    = "ssh-rsa ..."
14 }
15
16 output "id" {
17   value = aws_instance.instance.id
18 }
```

```
> 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 AAAAB3NzaC1yc2EAAAQABAAQCu0Q0lw92aaSP
G66j7eTV4c6toHiyVIna1eC7gWQemf3eCmIJIdVJXzHLnUdnADFYsAIQ0UYeb3KD9psPF5rY8Qj
t+hFCNviYOHIIIBzIY+/jKIGyzX7+8Bewv0fIqr3IQJWnasCsBruChw0JZMrxwGjxRgvUfaRHx
    + 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-03609584fc96e38ca"  
    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"
    + ingress                             = (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`
```

The screenshot shows the AWS EC2 Instances page with a single instance selected. The instance ID is i-0e058c0a708829cac. The instance is labeled as 'Running' with 2/2 checks passed and no alarms. The instance type is t3.micro. The 'Details' tab is active, showing the following information:

Instance summary		
Info		
Instance ID i-0e058c0a708829cac	Public IPv4 address 18.179.6.69 open address	Private IPv4 addresses 172.31.34.64
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-18-179-6-69.ap-northeast-1.compute.amazonaws.com open address
Private IPv4 DNS ip-172-31-34-64.ap-northeast-1.compute.internal	Instance type t3.micro	Elastic IP addresses -

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"

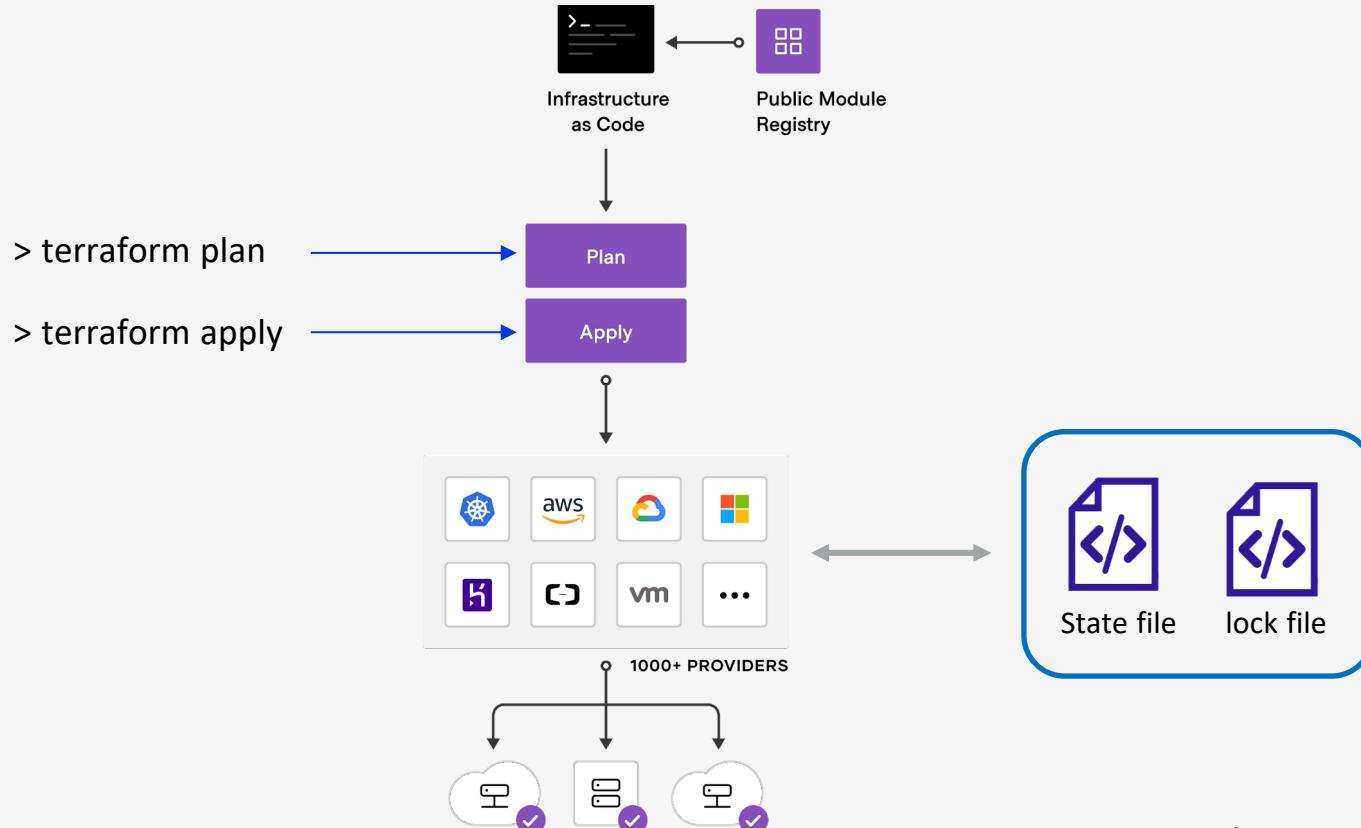
Summary

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

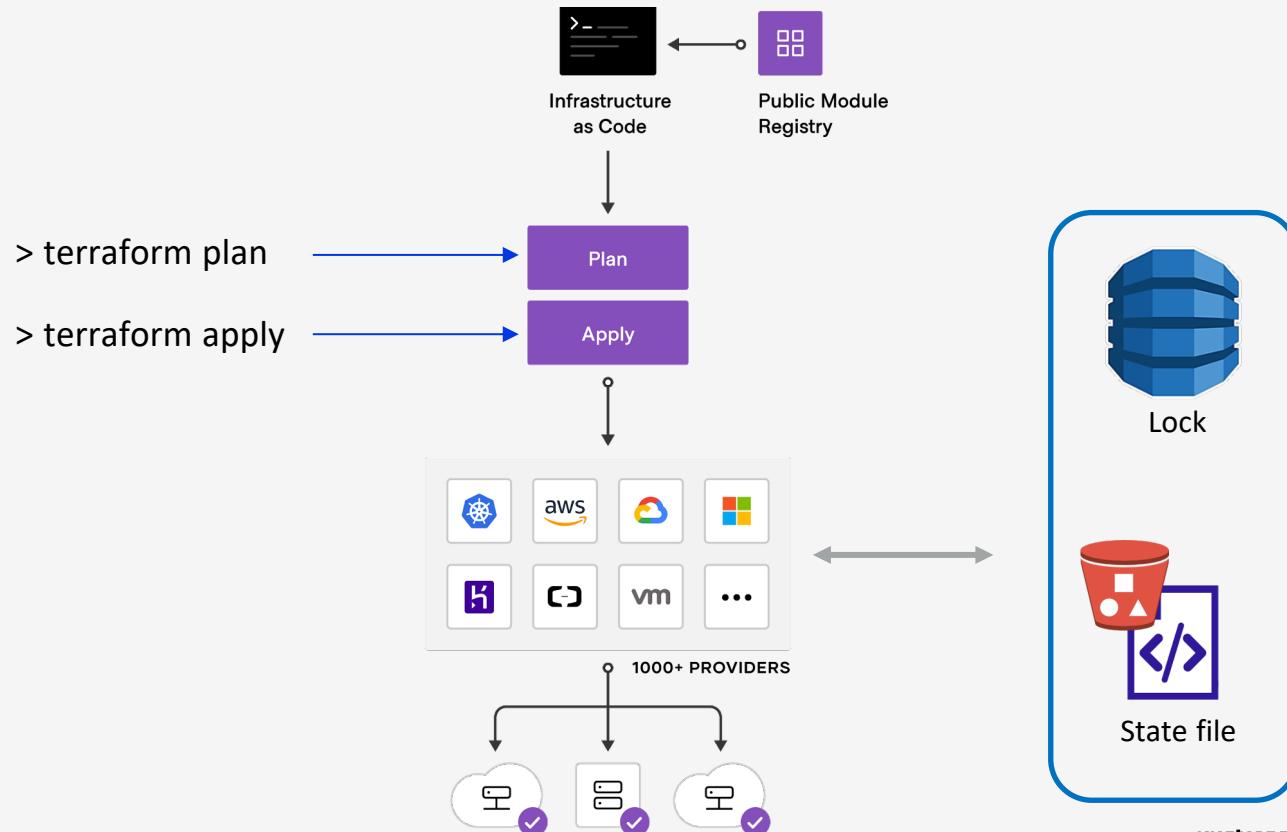
Feature 1

Remote backend

Local mode



S3 mode



Create S3 bucket

Create bucket Info

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

Bucket name
terraform-states
Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

AWS Region
Asia Pacific (Tokyo) ap-northeast-1

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.
[Choose bucket](#)

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning
 Disable
 Enable

- Create a S3 bucket in Tokyo
(Bucket name must be a global unique one)
- Enable bucket versioning

Create DynamoDB

DynamoDB > Tables > Create table

Create table

Table details Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.
 terraform-locks

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.
 LockID String

1 to 255 characters and case sensitive.

Sort key - optional
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.
 Enter the sort key name String

1 to 255 characters and case sensitive.

- 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 changes.
```

<input type="checkbox"/>	LockID	Digest		
<input type="checkbox"/>	lithome-cloudsummit-terraform-states/terraform.tfstate-md5	abbbcbcbe0b024ab09298f3ecd8e4f05c0		
<input type="checkbox"/>	Name	Type	Last modified	Size
<input type="checkbox"/>	terraform.tfstate	tfstate	November 11, 2021, 16:55:51 (UTC+08:00)	6.6 KB

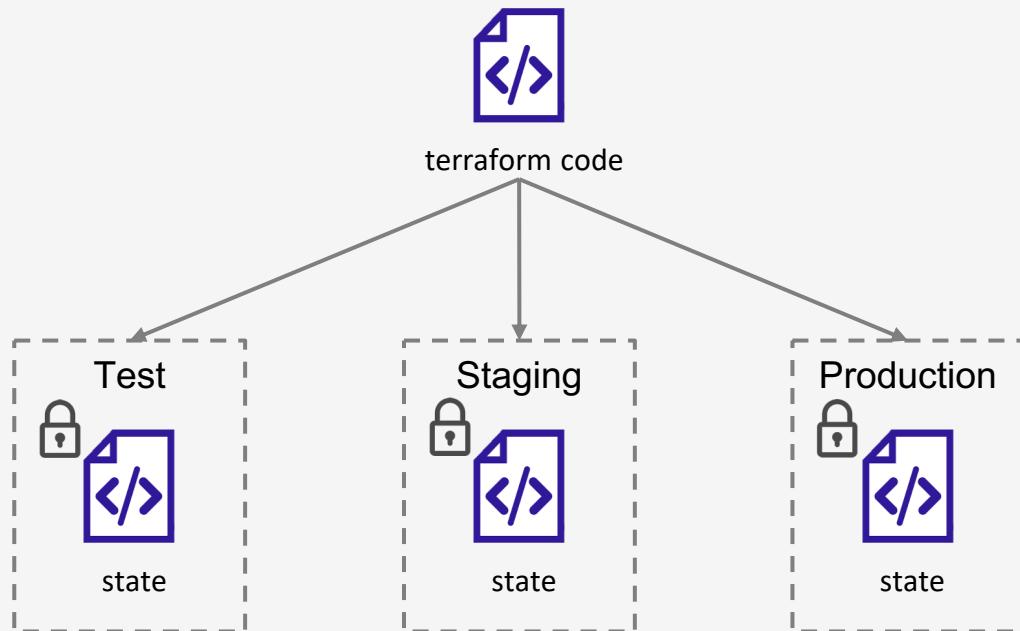
Summary

- 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-036d9684fc96838ca"  
    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

```
    },
  ],
  - name = "terraform-202111108513635980000001" -> 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
```

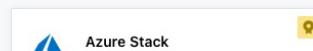
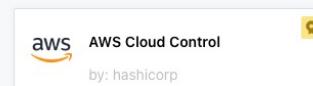
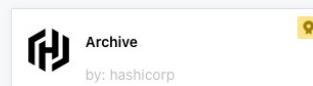
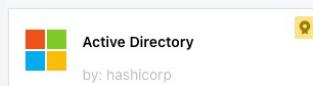
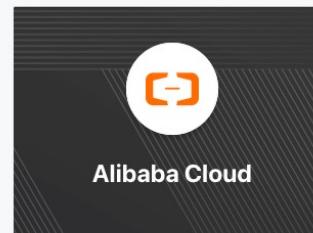
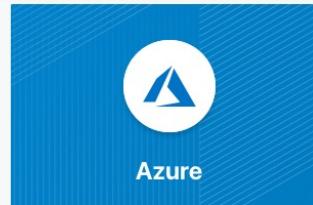
Summary

- 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-036d0604fc96838ca" // 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

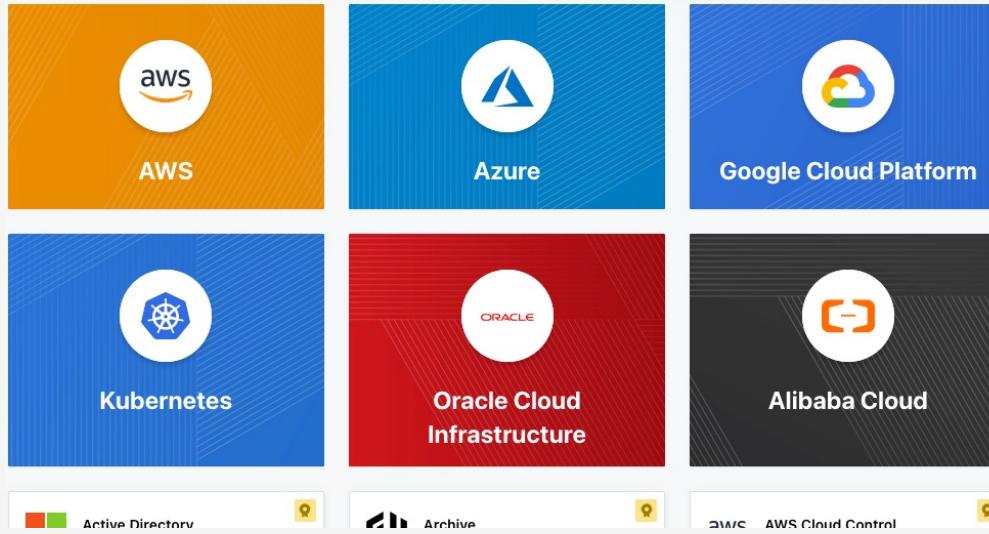
> terraform plan -out .plan

> 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
}

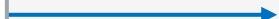
...

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

Modules

Modules are self-contained packages of Terraform configurations that are managed as a group.

terraform-aws-modules / vpc
Terraform module which creates VPC resources on AWS
🕒 10 days ago 14.4M aws provider

terraform-aws-modules / security-group
Terraform module which creates EC2-VPC security groups on AWS
🕒 a month ago 9.6M aws provider

terraform-aws-modules / iam
Terraform module which creates IAM resources on AWS
🕒 a month ago 7.1M aws provider

Terraform module registry

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

terraform-aws-ec2-ssh-auth-iam Public
SSH login without private key. Use aws-ec2-ssh (<https://github.com/widdix/aws-ec2-ssh>) to auto auth through IAM user public key
Shell ⭐ 0 4.8 MIT 1 0 0 Updated yesterday

terraform-aws-ec2-jump Public
Create a jump instance in VPC
HCL ⭐ 0 4.8 MIT 1 0 0 Updated yesterday

terraform-aws-mysql-user-creation Public
Auto create MySQL user
Python ⭐ 0 4.8 MIT 1 0 0 Updated yesterday

terraform-aws-postgresql-user-creation Public
Auto create PostgreSQL user
Python ⭐ 0 4.8 MIT 1 0 0 Updated yesterday

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

 Plan finished 5 days ago Resources: 0 to add, 1 to change, 0 to destroy ▾

 Policy check overridden 5 days ago Policies: 5 passed, 1 soft failed ▾

Queued 5 days ago > Soft failed 5 days ago > Overridden 5 days ago

- ✓ passed sentinel-aws-kks-global/ec2-cannot-use-private-key
- ✓ passed sentinel-aws-trc-smt-global-settings/cannot-destroy-resources
- ✓ passed sentinel-aws-trc-smt-global-settings/cannot-modify-iam-policy-attachment
- ✓ passed sentinel-aws-trc-smt-global-settings/shouldnot-create-resources
- ✓ passed sentinel-aws-trc-smt-global-settings/shouldnot-modify-anything-exclude-policies
- ✗ failed sentinel-aws-trc-smt-global-settings/shouldnot-use-wildcard-to-indicate-permissions

[View Log](#) | [View JSON Data](#)

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



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_GaeG3cOGSvWvPrynl5Jhg

主辦單位 **BROBRIDGE**



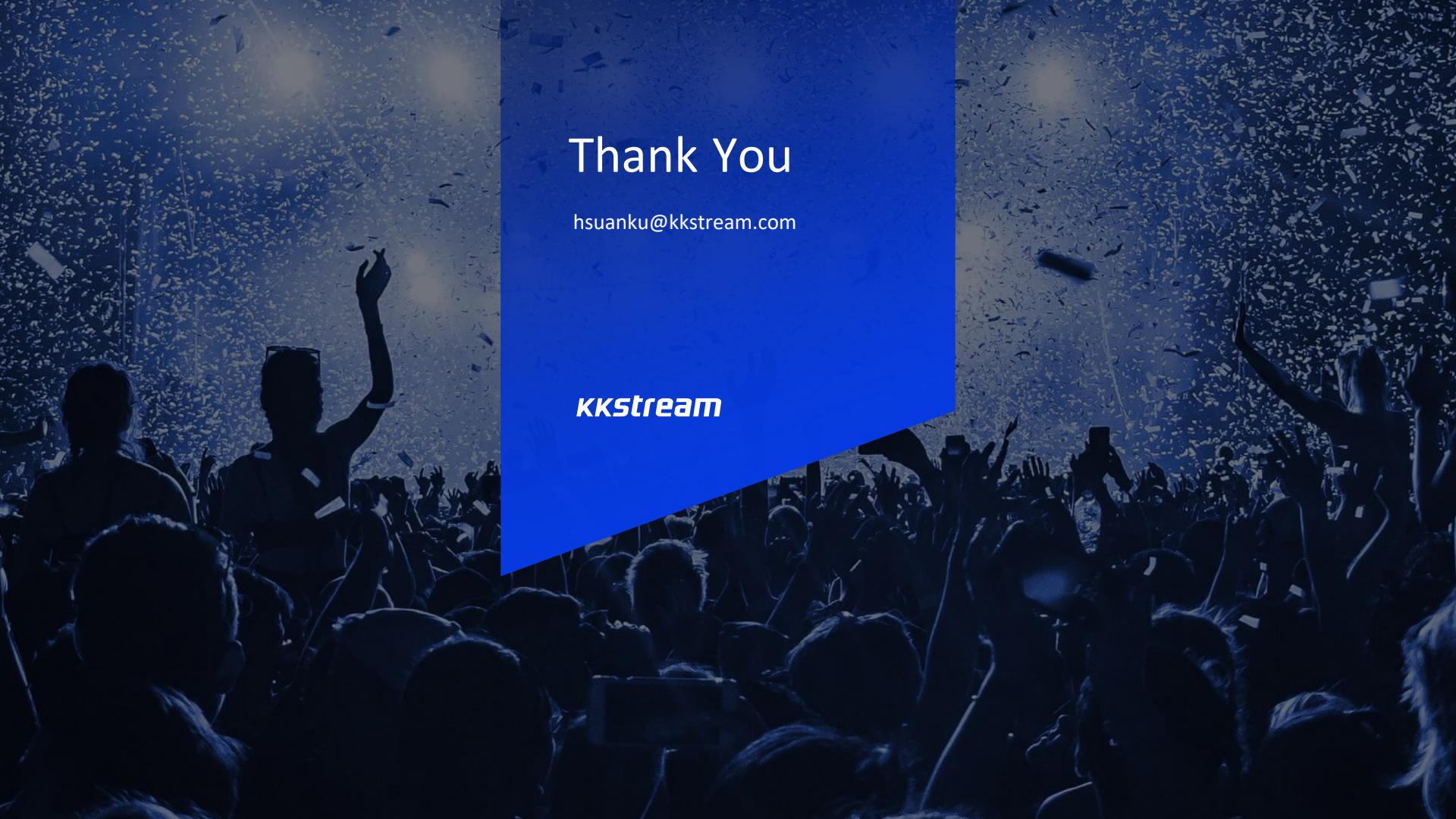
Q & A

Remember – AWS Credit

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

3. 歡迎留言告訴我們您們希望未來聽到的內容，或其他寶貴意見

- 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