

CASE STUDY -CREATING AN ARCHITECTURE USING TERRAFORM ON AWS

You work as a DevOps Engineer in leading Software Company. You have been asked to build an infrastructure safely and efficiently.

The company Requirements:

1. Use AWS cloud Provider and the software to be installed is Apache2
2. Use Ubuntu AMI

The company wants the Architecture to have the following services:

1. Create a template with a VPC, 2 subnets and 1 instance in each subnet
2. Attach Security groups, internet gateway and network interface to the instance

Launch an instance – terraform01

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. C following the simple steps below.

Name and tags [Info](#)

Name

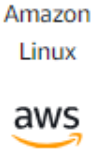
▼ Application and OS Images (Amazon Machine Image) [Info](#)

Select Ubuntu image


Search our full catalog including 1000s of application and OS images

Recents

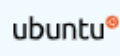
Quick Start




Amazon Linux




macOS




Ubuntu




Windows



Red Hat



SUSE L



Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type
ami-01811d4912b4ccb26 (64-bit (x86)) / ami-0f0a689dea27d9dc8 (64-bit (Arm))

Free tier eligible

▼ Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the key pair before you launch the instance.

Key pair name - *required*

Ruchii-kp

ENI

Additional charges apply when outside of free tier allowance

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to your instance.

☐ Create security group

☒ Select existing security group

Common security groups Info

Select security groups

default sg-05f30ea3543a0ed7a X
VPC: vpc-05adb8e0b5c08af51

Comp group

Security groups that you add or remove here will be added to or removed from all your network interfaces.

Successfully created instance

Instances (1) Info					Last updated less than a minute ago	Refresh	Conn
<input type="text" value="Find Instance by attribute or tag (case-sensitive)"/>							
<input type="checkbox"/>	Name ✎	Instance ID	Instance state ▼	Instance type			
<input type="checkbox"/>	Terraform01	i-02ed5128da7a408af	Running ⓘ 🔍	t2.micro			


Connect to the instance

[EC2](#) > [Instances](#) > [i-02ed5128da7a408af](#) > [Connect to instance](#)

Connect to instance [Info](#)

Connect to your instance i-02ed5128da7a408af (Terraform01) using any of these options

[EC2 Instance Connect](#) | [Session Manager](#) | [SSH client](#) | [EC2 serial console](#)

 **All ports are open to all IPv4 addresses in your security group**
All ports are currently open to all IPv4 addresses, indicated by **All** and **0.0.0.0/0** in the inbound

Update the machine

```
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
ubuntu@ip-172-31-16-86:~$ sudo apt-get update
```

i-02ed5128da7a408af (Terraform01)
PublicIPs: 54.255.53.1 PrivateIPs: 172.31.16.86

Create a shell script for installing terraform

```
ubuntu@ip-172-31-16-86:~$ sudo nano terraform_install.sh  
ubuntu@ip-172-31-16-86:~$
```

i-02ed5128da7a408af (Terraform01)

Copy the following commands for installing terraform

Download and add the GPG key for the HashiCorp repository

```
wget -O- https://apt.releases.hashicorp.com/gpg |  
sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
```

Add the HashiCorp repository to your sources list

```
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]  
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee  
/etc/apt/sources.list.d/hashicorp.list
```

Update the package list and install Terraform

```
sudo apt update && sudo apt install terraform -y
```

```
# Download and add the GPG key for the HashiCorp repository  
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg  
  
# Add the HashiCorp repository to your sources list  
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list  
  
# Update the package list and install Terraform  
sudo apt update && sudo apt install terraform -y
```

Execute the shell script with bash command

```
ubuntu@ip-172-31-16-86:~$ sudo bash terraform_install.sh
```

i-02ed5128da7a408af (Terraform01)

PublicIPs: 54.255.53.1 PrivateIPs: 172.31.16.86

Terraform installed successfully.

```
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

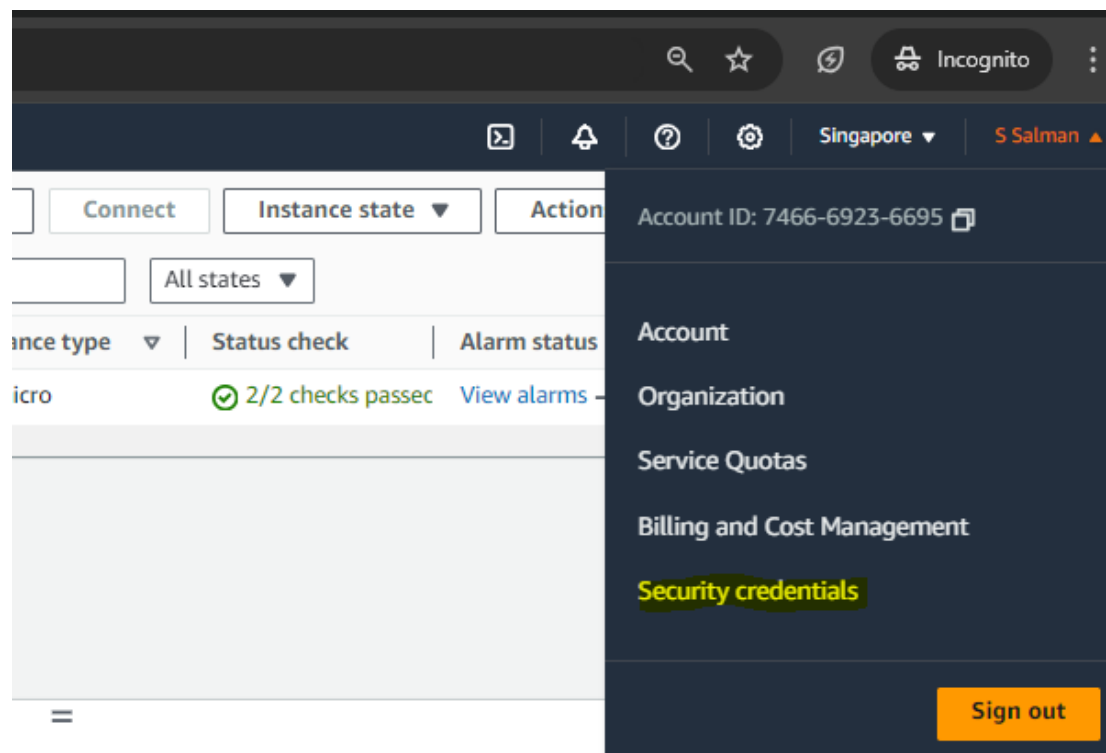
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-16-86:~$
```

Check the version and confirm terraform installation

```
ubuntu@ip-172-31-16-86:~$ terraform --version
Terraform v1.9.5
on linux_amd64
ubuntu@ip-172-31-16-86:~$
```

Now to give the aws provider in the terraform script we need to generate access key go to security credentials



Click on create access key

Access key ID	Created on	Access key last used	Region last used	Service last used	Status
No access keys					

As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials.[Learn more](#)

Create access key

Continue to create access key?

☒ I understand creating a root access key is not a best practice, but I still want to create one.



Cancel

Create access key

Access key and secret access key is generated

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make inactive.

Access key	Secret access key
 AKIA23WHUMHLYC5ZGTUB	 ***** Show

Create main.tf file for writing script

```
ubuntu@ip-172-31-16-86:~$ sudo nano main.tf
ubuntu@ip-172-31-16-86:~$
```

TERRAFORM SCRIPT:

Give the aws provider

```
provider "aws" {  
    region = "ap-southeast-1"  
    access_key = "AKIA23WHUMHLYC5ZGTUB"  
    secret_key = "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"  
}
```

Create a VPC

```
# Create a VPC  
  
resource "aws_vpc" "testvpc" {  
    cidr_block = "10.0.0.0/16"  
  
    tags = {  
        Name = "testvpc"  
    }  
}
```

Create a public subnet

```
# Create a Public Subnet  
  
resource "aws_subnet" "testsbnt1" {  
    vpc_id = aws_vpc.testvpc.id  
    cidr_block = "10.0.1.0/24"  
    map_public_ip_on_launch = "true"  
    availability_zone = "ap-southeast-1a"  
  
    tags = {  
        Name = "testsbnt1"  
    }  
}
```

Create a private subnet

```
# Create a Private Subnet  
  
resource "aws_subnet" "testsbnt2" {  
    vpc_id = aws_vpc.testvpc.id  
    cidr_block = "10.0.2.0/24"  
    map_public_ip_on_launch = "false"  
    availability_zone = "ap-southeast-1b"  
  
    tags = {  
        Name = "testsbnt2"  
    }  
}
```

Create an Internet gateway

```
# Create an Internet Gateway

resource "aws_internet_gateway" "testigw" {
  vpc_id = aws_vpc.testvpc.id
  tags = {
    Name = "testigw"
  }
}
```

Create a route table for public subnet

```
# Create a Route Table for Public Subnet

resource "aws_route_table" "testrtbl" {
  vpc_id = aws_vpc.testvpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.testigw.id
  }

  route {
    ipv6_cidr_block = ":::/0"
    gateway_id = aws_internet_gateway.testigw.id
  }
}
```

Associate public route table with public subnet

```
# Associate Public Route Table with Public Subnet

resource "aws_route_table_association" "testassoc1" {
  subnet_id = aws_subnet.testsbnt1.id
  route_table_id = aws_route_table.testrtbl.id
}
```


Create private RT for subnet 2

```
# Create a Private Route Table for Subnet 2

resource "aws_route_table" "testrtb2" {
  vpc_id = aws_vpc.testvpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_nat_gateway.nat.id
  }

  tags = {
    Name = "testrtb2"
  }
}
```

Associate RT with Private subnet

```
# Associate Route Table with Private Subnet

resource "aws_route_table_association" "testassoc2" {
  subnet_id = aws_subnet.testsbnt2.id
  route_table_id = aws_route_table.testrtb2.id
}
```

Assign ENI with IP

```
# Assign ENI with IP

resource "aws_network_interface" "testeni1" {
  subnet_id = aws_subnet.testsbnt1.id
  private_ips = ["10.0.1.10"]
  security_groups = [aws_security_group.testsg.id]
}

resource "aws_network_interface" "testeni2" {
  subnet_id = aws_subnet.testsbnt2.id
  private_ips = ["10.0.2.10"]
  security_groups = [aws_security_group.testsg.id]
}
```

Assign Elastic IP to ENI

```
# Assign Elastic IP to ENI

resource "aws_eip" "testeip1" {
  domain = "vpc"
  network_interface = aws_network_interface.testeni1.id
  associate_with_private_ip = "10.0.1.10"
  depends_on= [aws_internet_gateway.testigw, aws_instance.Instance1]
  tags = {
    Name = "testeip1"
  }
}
```

Create an Elastic IP Address for NAT Gateway

```
# Create an Elastic IP Address for NAT Gateway

resource "aws_eip" "testeip2" {
  domain = "vpc"
  associate_with_private_ip = "10.0.2.10"
  depends_on= [aws_internet_gateway.testigw]
  tags = {
    Name = "testeip2"
  }
}
```

Create a NAT Gateway for VPC

```
# Create a NAT Gateway for VPC

resource "aws_nat_gateway" "nat" {
  allocation_id = aws_eip.testeip2.id
  subnet_id = aws_subnet.testsbnt2.id

  tags = {
    Name = "nat"
  }
}
```

Create a security group

```
# Create a Security Group

resource "aws_security_group" "testsg" {
  description = "Allow limited inbound external traffic"
  vpc_id = aws_vpc.testvpc.id
  name = "testsg"

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

  ingress {
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
    from_port = 80
    to_port = 80
  }

  ingress {
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
    from_port = 443
    to_port = 443
  }

  egress {
    from_port = 0
    to_port = 0
    protocol = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "testsg"
  }
}
```

Create Linux server & Install/Enable Apache2 (Instance 1)

```
# Create Linux Server & Install/Enable Apache2 (Instance 1)

resource "aws_instance" "Instance1" {
  ami = "ami-0d07675d294f17973"
  instance_type = "t2.medium"
  availability_zone = "ap-southeast-1a"
  key_name = "Ruchii-kp"
  network_interface {
    device_index = 0
    network_interface_id = aws_network_interface.testeni1.id
  }
}

user_data = <<-EOF
#!/bin/bash
sudo apt update -y
sudo apt install apache2 -y
sudo systemctl start apache2
sudo systemctl enable apache2
EOF

tags = {
  Name = "Instance1"
}
}
```

Create Linux server & Install/Enable Apache2 (Instance 2)

```
# Create Linux Server & Install/Enable Apache2 Here (Instance 2)

resource "aws_instance" "Instance2" {
  ami = "ami-01811d4912b4ccb26"
  instance_type = "t2.medium"
  availability_zone = "ap-southeast-1b"
  key_name = "Ruchii-kp"
  network_interface {
    device_index = 0
    network_interface_id = aws_network_interface.testeni2.id
  }
}

user_data = <<-EOF
#!/bin/bash
sudo apt update -y
sudo apt install apache2 -y
sudo systemctl start apache2
sudo systemctl enable apache2
EOF

tags = {
  Name = "Instance2"
}
}
```

\$ terraform init

```
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.65.0...
- Installed hashicorp/aws v5.65.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
ubuntu@ip-172-31-16-86:~$
```

i-02ed5128da7a408af (Terraform01)

PublicIPs: 54.255.53.1 PrivateIPs: 172.31.16.86

\$ terraform plan

```
+ default_route_table_id      = (known after apply)
+ default_security_group_id    = (known after apply)
+ dhcp_options_id             = (known after apply)
+ enable_dns_hostnames         = (known after apply)
+ enable_dns_support           = true
+ enable_network_address_usage_metrics = (known after apply)
+ id                           = (known after apply)
+ instance_tenancy             = "default"
+ ipv6_association_id          = (known after apply)
+ ipv6_cidr_block              = (known after apply)
+ ipv6_cidr_block_network_border_group = (known after apply)
+ main_route_table_id         = (known after apply)
+ owner_id                     = (known after apply)
+ tags                         = {
+   + "Name" = "testvpc"
+ }
+ tags_all                     = {
+   + "Name" = "testvpc"
+ }
}
```

Plan: 16 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly

ubuntu@ip-172-31-16-86:~\$

i-02ed5128da7a408af (Terraform01)

\$ terraform apply

```
aws_network_interface.testeni2: Creating...
aws_nat_gateway.nat: Creating...
aws_network_interface.testeni2: Creation complete after 1s [id=eni-01ec06add02403d33]
aws_instance.Instance2: Creating...
aws_nat_gateway.nat: Still creating... [10s elapsed]
aws_instance.Instance2: Still creating... [10s elapsed]
aws_instance.Instance2: Creation complete after 12s [id=i-04133e37e0db5f258]
aws_nat_gateway.nat: Still creating... [20s elapsed]
aws_nat_gateway.nat: Still creating... [30s elapsed]
aws_nat_gateway.nat: Still creating... [40s elapsed]
aws_nat_gateway.nat: Still creating... [50s elapsed]
aws_nat_gateway.nat: Still creating... [1m0s elapsed]
aws_nat_gateway.nat: Still creating... [1m10s elapsed]
aws_nat_gateway.nat: Still creating... [1m20s elapsed]
aws_nat_gateway.nat: Still creating... [1m30s elapsed]
aws_nat_gateway.nat: Still creating... [1m40s elapsed]
aws_nat_gateway.nat: Still creating... [1m50s elapsed]
aws_nat_gateway.nat: Creation complete after 1m54s [id=nat-0a58e3f240a13cfb0]
aws_route_table.testrtb2: Modifying... [id=rtb-002e76786f330f5e0]
aws_route_table.testrtb2: Modifications complete after 1s [id=rtb-002e76786f330f5e0]
aws_route_table_association.testassoc2: Creating...
aws_route_table_association.testassoc2: Creation complete after 0s [id=rtbassoc-0b61de676f1b2e686]

Apply complete! Resources: 5 added, 1 changed, 4 destroyed.
ubuntu@ip-172-31-16-86:~$
```

i-02ed5128da7a408af (Terraform01)

PublicIPs: 54.255.53.1 PrivateIPs: 172.31.16.86

Instance 1 and instance 2 are created newly by the script

<div><div><div></div><div>Find Instance by attribute or tag (case-sensitive)</div></div></div>						<div>All states ▾</div>	
<div><input type="checkbox"/></div>	<div>Name <div></div></div>	<div>Instance ID</div>	<div>Instance state <div></div></div>	<div>Instance type <div></div></div>	<div>Status check</div>	<div>Alarm status</div>	<div>Av</div>
<div><input type="checkbox"/></div>	<div>Instance1</div>	<div>i-0a3b5f3b82b0c7136</div>	<div><div><div></div></div>Running <div><div></div><div></div></div></div>	<div>t2.medium</div>	<div><div><div></div></div>2/2 checks passec</div>	<div><div>View alarms</div> <div>+</div></div>	<div>ap</div>
<div><input type="checkbox"/></div>	<div>Terraform01</div>	<div>i-02ed5128da7a408af</div>	<div><div><div></div></div>Running <div><div></div><div></div></div></div>	<div>t2.micro</div>	<div><div><div></div></div>2/2 checks passec</div>	<div><div>View alarms</div> <div>+</div></div>	<div>ap</div>
<div><input type="checkbox"/></div>	<div>Instance2</div>	<div>i-04133e37e0db5f258</div>	<div><div><div></div></div>Running <div><div></div><div></div></div></div>	<div>t2.medium</div>	<div><div><div></div></div>2/2 checks passec</div>	<div><div>View alarms</div> <div>+</div></div>	<div>ap</div>

Public IP, private IP and elastic IP are addressed in Instance 1

i-0a3b5f3b82b0c7136 (Instance1)

Instance ID

i-0a3b5f3b82b0c7136 (Instance1)

IPv6 address

-

Hostname type

IP name: ip-10-0-1-10.ap-southeast-1.compute.internal

Answer private resource DNS name

-

Public IPv4 address

175.41.138.206 | open address

Instance state

Running

Private IP DNS name (IPv4 only)

ip-10-0-1-10.ap-southeast-1.compute.internal

Instance type

t2.medium

Private IPv4 addresses

10.0.1.10

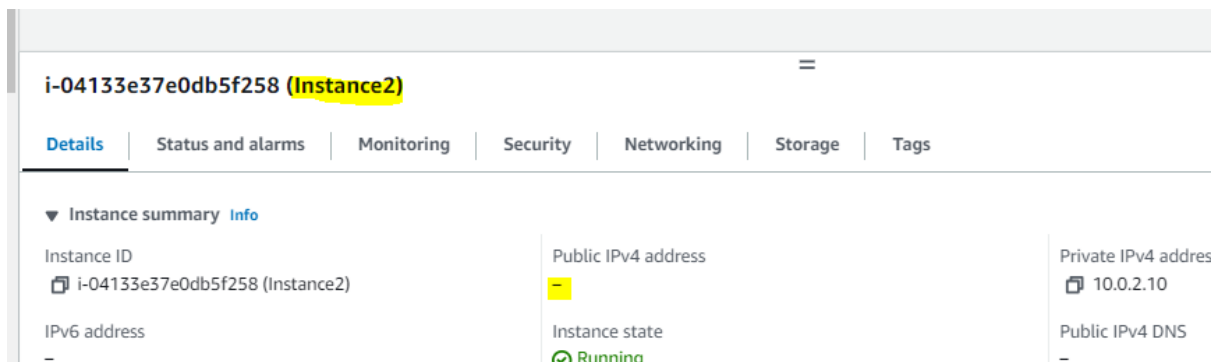
Public IPv4 DNS

-

Elastic IP addresses

175.41.138.206 (testiep1) [Public IP]

You can find public IP in instance2, as we have not written script for it



The screenshot shows the AWS Management Console for an instance named 'i-04133e37e0db5f258 (Instance2)'. The 'Details' tab is selected. The 'Instance summary' section shows the following information:

Instance ID	Public IPv4 address	Private IPv4 address
i-04133e37e0db5f258 (Instance2)	-	10.0.2.10

Below this, the 'Instance state' is shown as 'Running' with a green checkmark icon. The 'IPv6 address' and 'Public IPv4 DNS' fields are both empty, indicated by a '-' sign.

Browse the Public IP of Instance 1 to visit the web page of apache2



At all destroy your terraform script

```
ubuntu@ip-172-31-16-86:~$ terraform destroy
aws_vpc.testvpc: Refreshing state... [id=vpc-0f2467c869eb8a1e4]
aws_security_group.testsg: Refreshing state... [id=sg-07c41f7f104c36051]
aws_subnet.testsbnt2: Refreshing state... [id=subnet-0094d6a5c6db42f03]
aws_subnet.testsbnt1: Refreshing state... [id=subnet-012185220b63fc16d]
aws_internet_gateway.testigw: Refreshing state... [id=igw-0a49b7c893ad0c4]
```

Successfully destroyed

```
aws_internet_gateway.testigw: Destroying... [id=igw-0a49b7c893ad0c4a9]
aws_internet_gateway.testigw: Destruction complete after 1s
aws_network_interface.testeni1: Destruction complete after 1s
aws_security_group.testsg: Destroying... [id=sg-07c41f7f104c36051]
aws_subnet.testsbnt1: Destroying... [id=subnet-012185220b63fc16d]
aws_subnet.testsbnt1: Destruction complete after 0s
aws_security_group.testsg: Destruction complete after 0s
aws_vpc.testvpc: Destroying... [id=vpc-0f2467c869eb8a1e4]
aws_vpc.testvpc: Destruction complete after 1s

Destroy complete! Resources: 16 destroyed.
ubuntu@ip-172-31-16-86:~$
```

i-02ed5128da7a408af (Terraform01)

PublicIP: 54.255.53.1 PrivateIP: 172.31.16.86

you can see instances are terminated

Find Instance by attribute or tag (case-sensitive)						All states ▼
<input type="checkbox"/>	Name ↗	Instance ID	Instance state ▼	Instance type ▼	Status check	
<input type="checkbox"/>	Terraform01	i-02ed5128da7a408af	✔ Running 🔍 🔍	t2.micro	✔ 2/2 checks	
<input type="checkbox"/>	Instance2	i-04133e37e0db5f258	⊖ Terminated 🔍 🔍	t2.medium	-	
<input type="checkbox"/>	Instance1	i-0a3b5f3b82b0c7136	⊖ Terminated 🔍 🔍	t2.medium	-	