

Write Terraform script to create highly available infrastructure in AWS. The infra should have 1 vpc, 3 subnets setup in 3 different az and 2 instances setup in 2 different subnets

Step1: Creating the Roles for Terraform instance:

Home

Sign out

Select trusted entity

Step 1
Select trusted entity

Step 2
Add components

Step 3
Allow access and controls

Select trusted entity

☒ AWS service
Allow or restrict access to EC2 instances in relation to performance actions in this account.

☐ AWS account
Restrict access to other AWS accounts belonging to you as a third party to perform actions in this account.

☐ AWS identity
Allow users or identities in the specified external ID identity provider to assume this role to perform actions in this account.

☐ IAM L2 federation
Allow user federated with IAM L2 from a corporate directory to perform actions in this account.

☐ Custom trust policy
Have custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS account to call EC2 APIs, or allow to perform actions in this account.

Service or API role

EC2

Choose a use case for the specified service.
Use case

☒ EC2
Allow EC2 instances to call AWS services on your behalf.

☐ EC2 Role to AWS Systems Manager
Allow EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.

☐ EC2 Spot Fleet Role
Allow EC2 Spot Fleet to request and terminate Spot instances on your behalf.

☐ EC2 - Spot Fleet Auto Scaling
Allow Amazon EC2 to launch and manage EC2 Spot Fleets on your behalf.

☐ EC2 - Spot Fleet Tagging
Allow EC2 Spot fleet instances to attach tags to the launched instances on your behalf.

☐ EC2 - Instance Profiles
Allow EC2 Spot instances to launch and manage spot instances on your behalf.

☐ EC2 - Spot Fleet
Allow EC2 Spot fleet to launch and manage spot fleet instances on your behalf.

☐ EC2 - Spot Fleet Instance Management
Allow EC2 Scheduled instances to manage instance on your behalf.

[illegible][illegible]

[IAM](#) > **Roles**

Roles (4) [Add](#)

Delete

Create role

In IAM, roles are identities you can create that have specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

(1)

	Role name	Trusted entities	Last activity
<input type="checkbox"/>	AWS_s3dm	AWS Service s3	-
<input type="checkbox"/>	AWSServiceReflexTestLaunchWarning	AWS Service cloudwatchlogs	32 days ago
<input type="checkbox"/>	AWSServiceReflexSupport	AWS Service support	ServiceLinker -
<input type="checkbox"/>	AWSServiceReflexTrustAdminRole	AWS Service trustadvisor	ServiceLinker -
<input type="checkbox"/>	ControlPolicyRole	AWS Service cloudpolicy	-
<input type="checkbox"/>	Key_to_deploy_s3	AWS Service s3	7 days ago

Step 2 : Creating the Ec2 Instance with New Role.

[Launch an instance](#)

Launch an instance

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

Summary

Number of instances
1

Software image (AMI)
Canonical, Ubuntu, 22.04 LTS, _...read more
ami-03f487675d343977f

Virtual server type (Instance type)
t2.micro

Firewall (security group)
New security group

Storage (volume)
1 volume(s) - 8 GB

Free tier! In your first year includes 750 hours of t2.micro or t3.micro in the Regions in which t2.micro is available. Instance stops on free tier AMIs per month, 30 GB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Cancel
Launch instance
Review commands

Application and OS Images (Amazon Machine Image)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or browse for AMIs if you don't see what you are looking for below.

Quick Start

Amazon Linux

macOS

Ubuntu

Windows

Red Hat

SUSE Linux

Browse more AMIs including AMIs from AWS, marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type
ami-03f487675d343977f (64-bit x86_64) | ami-07f865d343977f (ARM 64-bit ARMv8)

Virtualization: hvm
Encryption: true
Root device type: ebs

Free tier eligible

Description

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image built on 2023-12-07

Architecture: ARM ID: ami-03f487675d343977f

Advanced details

Info

Domain join directory

Info

Select

Create new directory

IAM instance profile

Info

us-east-1-admin

Create new IAM profile

Hostname type

Info

IP name

DNS Hostname

Info

☒ Enable IP name (IPv4 (A record)) DNS requests

☒ Enable resource-based IPv4 (A record) DNS requests

☐ Enable resource-based IPv6 (AAAA record) DNS requests

Instance auto-recovery

Info

Select

Shutdown behavior

Info

Stop

Stop - Hibernate behavior

Info

Select

Termination protection

Info

Select

Stop protection

Info

Summary

Number of instances

Info

1

Software Image (AMI)

Canonical, Ubuntu, 22.04 LTS, ...read more

ami-0848787554340779

Virtual server type (instance type)

t2.micro

Firewall (security group)

launch-wizard-1

Storage (volumes)

1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) Instance usage on free tier: AMIs per month, 30 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GiB of bandwidth to the internet.

Cancel

Launch instance

Review commands

Step 3: Updating the Instance and installation Terraform:

```
root@ip-172-31-41-35: /home/ubuntu
root@ip-172-31-41-35:/home/ubuntu# terraform --version
Terraform v1.7.4
on linux_amd64
root@ip-172-31-41-35:/home/ubuntu#
```

Step 4: Create a new .tf file in a new directory & wirte the Terraform code:

```
root@ip-172-31-41-35: /home/ubuntu/Task
root@ip-172-31-41-35:/home/ubuntu/Task# cat terraform.tf
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.38.0"
    }
  }
}
provider "aws" {
  region = "ap-south-1"
}

# Create VPC
resource "aws_vpc" "terra_vpc" {
  cidr_block = "10.100.0.0/16"
}

# Create three subnets (one per AZ)
resource "aws_subnet" "subnet_a" {
  vpc_id = aws_vpc.terra_vpc.id
  cidr_block = "10.100.1.0/24"
  availability_zone = "ap-south-1a"
}

resource "aws_subnet" "subnet_b" {
  vpc_id = aws_vpc.terra_vpc.id
  cidr_block = "10.100.2.0/24"
  availability_zone = "ap-south-1b"
}

resource "aws_subnet" "subnet_c" {
  vpc_id = aws_vpc.terra_vpc.id
  cidr_block = "10.100.3.0/24"
  availability_zone = "ap-south-1c"
}

# Create EC2 instances
resource "aws_instance" "instance_a" {
  ami = "ami-03f4878755434977f"
  instance_type = "t2.micro"
  subnet_id = aws_subnet.subnet_a.id
}

resource "aws_instance" "instance_b" {
  ami = "ami-03f4878755434977f"
  instance_type = "t2.micro"
  subnet_id = aws_subnet.subnet_b.id
}
root@ip-172-31-41-35:/home/ubuntu/Task#
```

Step 5: Run the terraform init command inside the directory:

```
root@ip-172-31-41-35: /home/ubuntu/Task
root@ip-172-31-41-35:/home/ubuntu/Task# terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.38.0"...
- Installing hashicorp/aws v5.38.0...
- Installed hashicorp/aws v5.38.0 (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.
root@ip-172-31-41-35:/home/ubuntu/Task#
```

Step 6: Run the terraform plan command inside the directory:

```
root@ip-172-31-41-35:/home/ubuntu/Task# terraform plan

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.instance_a will be created
+ resource "aws_instance" "instance_a" {
  + ami                     = "ami-03f4878755434977f"
  + arn                     = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone       = (known after apply)
  + cpu_core_count          = (known after apply)
  + cpu_threads_per_core    = (known after apply)
  + disable_api_stop        = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized           = (known after apply)
  + get_password_data       = false
  + host_id                 = (known after apply)
  + host_resource_group_arn = (known after apply)
  + iam_instance_profile    = (known after apply)
  + id                      = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_lifecycle      = (known after apply)
  + instance_state          = (known after apply)
  + instance_type           = "t2.micro"
  + ipv6_address_count      = (known after apply)
  + ipv6_addresses          = (known after apply)
  + key_name                = (known after apply)
  + monitoring              = (known after apply)
  + outpost_arn             = (known after apply)
  + password_data           = (known after apply)
  + placement_group         = (known after apply)
  + placement_partition_number = (known after apply)
  + primary_network_interface_id = (known after apply)
  + private_dns             = (known after apply)
  + private_ip              = (known after apply)
  + public_dns              = (known after apply)
  + public_ip               = (known after apply)
  + secondary_private_ips   = (known after apply)
  + security_groups         = (known after apply)
  + source_dest_check       = true
  + spot_instance_request_id = (known after apply)
  + subnet_id               = (known after apply)
  + tags_all                = (known after apply)
  + tenancy                 = (known after apply)
  + user_data               = (known after apply)
  + user_data_base64        = (known after apply)
  + user_data_replace_on_change = false
  + vpc_security_group_ids  = (known after apply)
}

# aws_instance.instance_b will be created
+ resource "aws_instance" "instance_b" {
  + ami                     = "ami-03f4878755434977f"
  + arn                     = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone       = (known after apply)
  + cpu_core_count          = (known after apply)
```

```
+ enable_dns64              = false
+ enable_resource_name_dns_a_record_on_launch = false
+ enable_resource_name_dns_aaaa_record_on_launch = false
+ id                        = (known after apply)
+ ipv6_cidr_block_association_id = (known after apply)
+ ipv6_native               = false
+ map_public_ip_on_launch  = false
+ owner_id                 = (known after apply)
+ private_dns_hostname_type_on_launch = (known after apply)
+ tags_all                 = (known after apply)
+ vpc_id                   = (known after apply)
}

# aws_subnet.subnet_c will be created
+ resource "aws_subnet" "subnet_c" {
  + arn                     = (known after apply)
  + assign_ipv6_address_on_creation = false
  + availability_zone       = "ap-south-1c"
  + availability_zone_id    = (known after apply)
  + cidr_block              = "10.100.3.0/24"
  + enable_dns64            = false
  + enable_resource_name_dns_a_record_on_launch = false
  + enable_resource_name_dns_aaaa_record_on_launch = false
  + id                      = (known after apply)
  + ipv6_cidr_block_association_id = (known after apply)
  + ipv6_native             = false
  + map_public_ip_on_launch = false
  + owner_id               = (known after apply)
  + private_dns_hostname_type_on_launch = (known after apply)
  + tags_all               = (known after apply)
  + vpc_id                 = (known after apply)
}

# aws_vpc.terra_vpc will be created
+ resource "aws_vpc" "terra_vpc" {
  + arn                     = (known after apply)
  + cidr_block              = "10.100.0.0/16"
  + default_network_acl_id  = (known after apply)
  + 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_all               = (known after apply)
}

Plan: 6 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 these actions if you run "terraform apply" now.
root@ip-172-31-41-35:/home/ubuntu/Task#
```

Step 7: Run the Terraform apply command to create the new VPN , Subnets & Instances.

```
root@ip-172-31-41-35: /home/ubuntu/Task
+ enable_resource_name_dns_a_record_on_launch = false
+ enable_resource_name_dns_aaaa_record_on_launch = false
+ id = (known after apply)
+ ipv6_cidr_block_association_id = (known after apply)
+ ipv6_native = false
+ map_public_ip_on_launch = false
+ owner_id = (known after apply)
+ private_dns_hostname_type_on_launch = (known after apply)
+ tags_all = (known after apply)
+ vpc_id = (known after apply)
}

# aws_vpc.terra_vpc will be created
+ resource "aws_vpc" "terra_vpc" {
+   arn = (known after apply)
+   cidr_block = "10.100.0.0/16"
+   default_network_acl_id = (known after apply)
+   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_all = (known after apply)
}

Plan: 6 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_vpc.terra_vpc: Creating...
aws_vpc.terra_vpc: Creation complete after 1s [id=vpc-0f226d1464c48aaef]
aws_subnet.subnet_a: Creating...
aws_subnet.subnet_b: Creating...
aws_subnet.subnet_c: Creating...
aws_subnet.subnet_a: Creation complete after 1s [id=subnet-0c28007bfe8c4ce4c]
aws_subnet.subnet_b: Creation complete after 1s [id=subnet-09d344a9769d3f1eb]
aws_subnet.subnet_c: Creation complete after 1s [id=subnet-09d344a9769d3f1eb]
aws_instance.instance_a: Creating...
aws_instance.instance_a: Still creating... [10s elapsed]
aws_instance.instance_a: Still creating... [10s elapsed]
aws_instance.instance_b: Still creating... [20s elapsed]
aws_instance.instance_a: Still creating... [20s elapsed]
aws_instance.instance_b: Creation complete after 21s [id=i-0e522717198e2e6ee]
aws_instance.instance_b: Still creating... [30s elapsed]
aws_instance.instance_b: Creation complete after 31s [id=i-0ae92148ee2bf6420]

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.
root@ip-172-31-41-35: /home/ubuntu/Task#
```

New VPC created.

Your VPCs (2) info

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set	Main route table	Main network ACL	Tenancy	Default VPC
-	vpc-0a545112a0095c1	Available	172.31.0.0/16	-	dhcp-025a0f4e10c35c18	rtb-0b05f42007b60278	acl-0a8445a319c5900a	Default	Yes
-	vpc-0f226d1464c48aaef	Available	10.100.0.0/16	-	dhcp-025a0f4e10c35c18	rtb-0c3a0c3b40509002	acl-0a8445a319c5900a	Default	No

Three new subnets created in three different AZ.

-	subnet-0c28007bfe8c4ce4c	Available	vpc-0f226d1464c48aaef	10.100.3.0/24	-	251	ap-south-1a	ap-s1-az2
-	subnet-09d344a9769d3f1eb	Available	vpc-0f226d1464c48aaef	10.100.2.0/24	-	250	ap-south-1b	ap-s1-az3
-	subnet-09d344a9769d3f1eb	Available	vpc-0f226d1464c48aaef	10.100.1.0/24	-	250	ap-south-1a	ap-s1-az1

Two Instance created in two different subnets.

Instance ID	Description	Platform	Platform	Public IP address	Private IP address	Private IP DNS	IP address	Private IP address	Attachment time	Interface owner
i-0ae92148ee2bf6420	-	Linux	Linux	10.100.1.101	10.100.1.101	-	-	-	Tue Feb 27 2024 12:11	EC2-20240227
i-0e522717198e2e6ee	-	Linux	Linux	10.100.3.101	10.100.3.101	-	-	-	Tue Feb 27 2024 12:11	EC2-20240227

Step 8: Run the terraform destroy command

```
root@ip-172-31-41-35: /home/ubuntu/Task

# aws_vpc.terra_vpc will be destroyed
- resource "aws_vpc" "terra_vpc" {
  - arn = "arn:aws:ec2:ap-south-1:992382583979:vpc/vpc-0f226dl464c48aaef" -> null
  - assign_generated_ipv6_cidr_block = false -> null
  - cidr_block = "10.100.0.0/16" -> null
  - default_network_acl_id = "acl-0dd9e6363b0e69393" -> null
  - default_route_table_id = "rtb-0e2ca528a46690982" -> null
  - default_security_group_id = "sg-07b85d790d75fb96a" -> null
  - dhcp_options_id = "dopt-02fda6f4a10c93e18" -> null
  - enable_dns_hostnames = false -> null
  - enable_dns_support = true -> null
  - enable_network_address_usage_metrics = false -> null
  - id = "vpc-0f226dl464c48aaef" -> null
  - instance_tenancy = "default" -> null
  - ipv6_netmask_length = 0 -> null
  - main_route_table_id = "rtb-0e2ca528a46690982" -> null
  - owner_id = "992382583979" -> null
  - tags = {} -> null
  - tags_all = {} -> null
}

Plan: 0 to add, 0 to change, 6 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.instance_a: Destroying... [id=i-0e522717198e2e6ee]
aws_instance.instance_b: Destroying... [id=i-0ae92148ee2bf6420]
aws_subnet.subnet_c: Destroying... [id=subnet-0c28007bfe8c4ce4c]
aws_subnet.subnet_c: Destruction complete after 0s
aws_instance.instance_a: Still destroying... [id=i-0e522717198e2e6ee, 10s elapsed]
aws_instance.instance_b: Still destroying... [id=i-0ae92148ee2bf6420, 10s elapsed]
aws_instance.instance_a: Still destroying... [id=i-0e522717198e2e6ee, 20s elapsed]
aws_instance.instance_b: Still destroying... [id=i-0ae92148ee2bf6420, 20s elapsed]
aws_instance.instance_a: Still destroying... [id=i-0e522717198e2e6ee, 30s elapsed]
aws_instance.instance_b: Still destroying... [id=i-0ae92148ee2bf6420, 30s elapsed]
aws_instance.instance_b: Destruction complete after 40s
aws_subnet.subnet_b: Destroying... [id=subnet-0c63a190d2b9a651c]
aws_instance.instance_a: Still destroying... [id=i-0e522717198e2e6ee, 40s elapsed]
aws_subnet.subnet_b: Destruction complete after 0s
aws_instance.instance_a: Still destroying... [id=i-0e522717198e2e6ee, 50s elapsed]
aws_instance.instance_a: Destruction complete after 50s
aws_subnet.subnet_a: Destroying... [id=subnet-09d344a9769df4feb]
aws_subnet.subnet_a: Destruction complete after 0s
aws_vpc.terra_vpc: Destroying... [id=vpc-0f226dl464c48aaef]
aws_vpc.terra_vpc: Destruction complete after 1s

Destroy complete! Resources: 6 destroyed.
root@ip-172-31-41-35: /home/ubuntu/Task#
```