

Lab Exam

Q1 – AWS IAM Setup Using AWS CLI and Console Verification (10 marks)

Step 1: Create IAM Group

aws iam create-group --group-name SoftwareEngineering

```
● @23-22411-029-sudo →/workspaces/Exam (main) $ aws iam create-group --group-name SoftwareEngineering
{
  "Group": {
    "Path": "/",
    "GroupName": "SoftwareEngineering",
    "GroupId": "AGPARPRNTUQVX3N3WLDNR",
    "Arn": "arn:aws:iam::102101132331:group/SoftwareEngineering",
    "CreateDate": "2026-01-19T07:34:38+00:00"
  }
}
```

aws iam get-group --group-name SoftwareEngineering

```
● @23-22411-029-sudo →/workspaces/Exam (main) $ aws iam get-group --group-name SoftwareEngineering
{
  "Users": [],
  "Group": {
    "Path": "/",
    "GroupName": "SoftwareEngineering",
    "GroupId": "AGPARPRNTUQVX3N3WLDNR",
    "Arn": "arn:aws:iam::102101132331:group/SoftwareEngineering",
    "CreateDate": "2026-01-19T07:34:38+00:00"
  }
}
```

Step 2: Create IAM User Kainat

aws iam create-user --user-name Kainat

```
● @23-22411-029-sudo →/workspaces/Exam (main) $ aws iam create-user --user-name Kainat
{
  "User": {
    "Path": "/",
    "UserName": "Kainat",
    "UserId": "AIDARPRNTUQVTG4TFM6RH",
    "Arn": "arn:aws:iam::102101132331:user/Kainat",
    "CreateDate": "2026-01-19T07:36:13+00:00"
  }
}
```

aws iam get-user --user-name Kainat

```

@23-22411-029-sudo →/workspaces/Exam (main) $ aws iam get-user --user-name Kainat
{
  "User": {
    "Path": "/",
    "UserName": "Kainat",
    "UserId": "AIDARPRNTUQVTG4TFM6RH",
    "Arn": "arn:aws:iam::102101132331:user/Kainat",
    "CreateDate": "2026-01-19T07:36:13+00:00"
  }
}

```

Step 3: Add User to Group

```

aws iam add-user-to-group \
  --user-name Kainat \
  --group-name SoftwareEngineering

```

```

@23-22411-029-sudo →/workspaces/Exam (main) $ aws iam add-user-to-group \
  --user-name Kainat \
  --group-name SoftwareEngineering
@23-22411-029-sudo →/workspaces/Exam (main) $

```

```

aws iam get-group --group-name SoftwareEngineering

```

```

@23-22411-029-sudo →/workspaces/Exam (main) $ aws iam get-group --group-name SoftwareEngineering
{
  "Users": [
    {
      "Path": "/",
      "UserName": "Kainat",
      "UserId": "AIDARPRNTUQVTG4TFM6RH",
      "Arn": "arn:aws:iam::102101132331:user/Kainat",
      "CreateDate": "2026-01-19T07:36:13+00:00"
    }
  ],
  "Group": {
    "Path": "/",
    "GroupName": "SoftwareEngineering",
    "GroupId": "AGPARPRNTUQVX3N3WLDNRN",
    "Arn": "arn:aws:iam::102101132331:group/SoftwareEngineering",
    "CreateDate": "2026-01-19T07:34:38+00:00"
  }
}
@23-22411-029-sudo →/workspaces/Exam (main) $

```

Step 4: Find AdministratorAccess Policy

```

aws iam list-policies --scope AWS | grep AdministratorAccess

```

```
@23-22411-029-sudo →/workspaces/Exam (main) $ aws iam list-policies --scope AWS | grep AdministratorAccess
  "PolicyName": "AdministratorAccess",
  "Arn": "arn:aws:iam::aws:policy/AdministratorAccess",
  "PolicyName": "AdministratorAccess-Amplify",
  "Arn": "arn:aws:iam::aws:policy/AdministratorAccess-Amplify",
  "PolicyName": "AWSAuditManagerAdministratorAccess",
  "Arn": "arn:aws:iam::aws:policy/AWSAuditManagerAdministratorAccess",
  "PolicyName": "AdministratorAccess-AWSElasticBeanstalk",
  "Arn": "arn:aws:iam::aws:policy/AdministratorAccess-AWSElasticBeanstalk",
  "PolicyName": "AWSManagementConsoleAdministratorAccess",
  "Arn": "arn:aws:iam::aws:policy/job-function/AWSManagementConsoleAdministratorAccess",
@23-22411-029-sudo →/workspaces/Exam (main) $
```

Step 5: Attach Policy to Group

```
aws iam attach-group-policy \
  --group-name SoftwareEngineering \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

```
@23-22411-029-sudo →/workspaces/Exam (main) $ aws iam attach-group-policy \
  --group-name SoftwareEngineering \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
@23-22411-029-sudo →/workspaces/Exam (main) $
```

```
aws iam list-attached-group-policies \
  --group-name SoftwareEngineering
```

```
@23-22411-029-sudo →/workspaces/Exam (main) $ aws iam list-attached-group-policies \
  --group-name SoftwareEngineering
{
  "AttachedPolicies": [
    {
      "PolicyName": "AdministratorAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
    }
  ]
}
@23-22411-029-sudo →/workspaces/Exam (main) $
```

Step 6: Console Verification

Go to **IAM Console**

- q1_console_group.png

User groups (2) [Info](#)

A user group is a collection of IAM users. Use groups to specify permissions for a collection of users.

<input type="checkbox"/>	Group name	Users	Permissions	Creation time
<input type="checkbox"/>	developers	1	Defined	Yesterday
<input type="checkbox"/>	SoftwareEngineering	1	Defined	8 minutes ago

- q1_console_user_in_group.png

Users (2) [Info](#)

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

<input type="checkbox"/>	User name	Path	Group	Last activity	MFA	Password age	Console
<input type="checkbox"/>	Kainat	/	1	-	-	-	-
<input type="checkbox"/>	loadbalancer	/users/	1	8 minutes ago	-	Yesterday	-

- q1_console_group_policy.png

The screenshot shows the AWS IAM console interface. On the left is a navigation sidebar with 'Identity and Access Management (IAM)' selected. The main content area shows the details for the 'SoftwareEngineering' user group. The 'Summary' section displays the group name, creation time, and ARN. Below this, the 'Users (1)' tab is active, showing a table of users in the group. The table has columns for 'User name', 'Groups', 'Last activity', and 'Creation time'. One user, 'Kainat', is listed with 'None' for groups and '8 minutes ago' for creation time. The 'Users in this group (1)' section title and the table itself are highlighted with a green rectangular box.

<input type="checkbox"/>	User name	Groups	Last activity	Creation time
<input type="checkbox"/>	Kainat	None	None	8 minutes ago

— TERRAFORM (30 MARKS)

Step 1: Folder Structure

mkdir terraform
cd terraform

```
@23-22411-029-sudo →/workspaces/Exam (main) $ mkdir terraform
cd terraform
@23-22411-029-sudo →/workspaces/Exam/terraform (main) $
```

Files you WILL create:

main.tf
variables.tf
outputs.tf
entry-script.sh
terraform.tfvars

```
• @23-22411-029-sudo →/workspaces/Exam (main) $ mkdir terraform
  cd terraform
• @23-22411-029-sudo →/workspaces/Exam/terraform (main) $ touch main.tf
• @23-22411-029-sudo →/workspaces/Exam/terraform (main) $ touch variables.tf
• @23-22411-029-sudo →/workspaces/Exam/terraform (main) $ touch outputs.tf
• @23-22411-029-sudo →/workspaces/Exam/terraform (main) $ touch entry-script.sh
• @23-22411-029-sudo →/workspaces/Exam/terraform (main) $ touch terraform.tfvars
• @23-22411-029-sudo →/workspaces/Exam/terraform (main) $ ls
  entry-script.sh main.tf outputs.tf terraform.tfvars variables.tf
• @23-22411-029-sudo →/workspaces/Exam/terraform (main) $
```

Step 2: Provider

```
• @23-22411-029-sudo →/workspaces/Exam/terraform (main) $ cat ~/.aws/config
[default]
region = ap-south-1
output = json
• @23-22411-029-sudo →/workspaces/Exam/terraform (main) $
```

Step 1 – Configure AWS provider

Edit `main.tf` to configure the AWS provider:

Add to `main.tf`:

```
GNU nano 7.2 main.tf *
cidr_block      = var.subnet_cidr_block
availability_zone = var.availability_zone
tags = { Name = "${var.env_prefix}-subnet-1" }
}
resource "aws_internet_gateway" "myapp_igw" {
  vpc_id = aws_vpc.myapp_vpc.id
  tags   = { Name = "${var.env_prefix}-igw" }
}

resource "aws_route_table" "myapp_rt" {
  vpc_id = aws_vpc.myapp_vpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.myapp_igw.id
  }
  tags = { Name = "${var.env_prefix}-rt" }
}

resource "aws_route_table_association" "subnet_assoc" {
  subnet_id      = aws_subnet.myapp_subnet.id
  route_table_id = aws_route_table.myapp_rt.id
}
```

Step 5 – Discover public IP & define local /32

Add to `main.tf`:

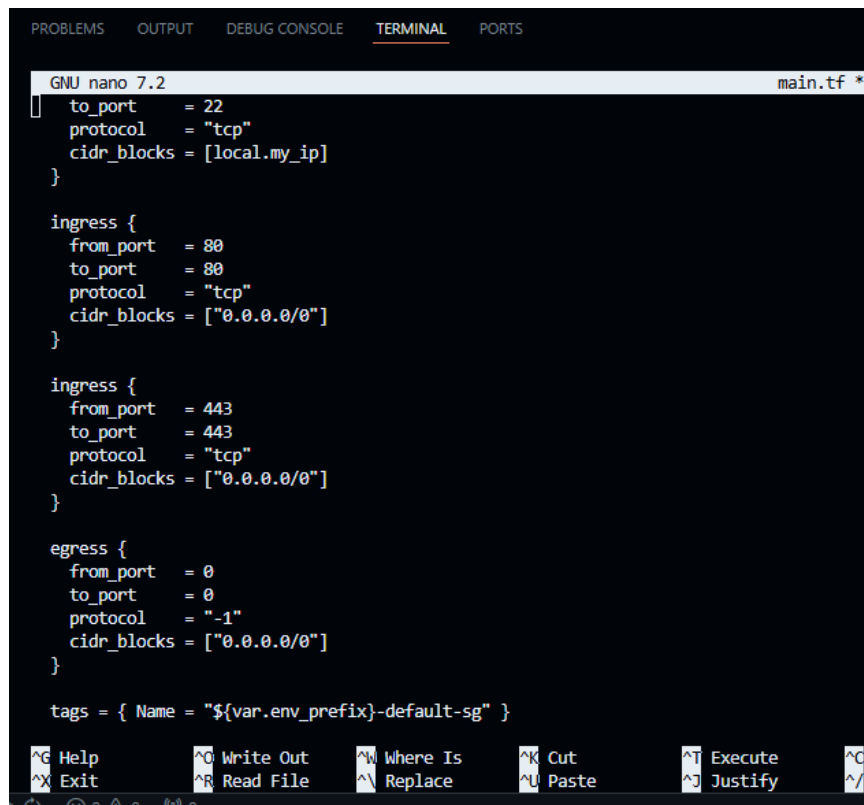
```
resource "aws_route_table_association" "subnet_assoc" {
  subnet_id      = aws_subnet.myapp_subnet.id
  route_table_id = aws_route_table.myapp_rt.id
}
data "http" "my_ip" {
  url = "https://icanhazip.com"
}

locals {
  my_ip = "${chomp(data.http.my_ip.body)}/32"
}

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Lo
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/_ Go
```

Step 6 – Default Security Group

Add to `main.tf`:



The screenshot shows a terminal window with the nano 7.2 editor open to a file named main.tf. The code defines a security group with three ingress rules and one egress rule. The first ingress rule allows traffic on port 22 (SSH) from local.my_ip. The second ingress rule allows traffic on port 80 (HTTP) from 0.0.0.0/0. The third ingress rule allows traffic on port 443 (HTTPS) from 0.0.0.0/0. The egress rule allows traffic on port -1 (all ports) to 0.0.0.0/0. The security group is tagged with the name "\${var.env_prefix}-default-sg".

```
GNU nano 7.2 main.tf *
to_port = 22
protocol = "tcp"
cidr_blocks = [local.my_ip]
}

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

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

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

tags = { Name = "${var.env_prefix}-default-sg" }
```

Step 7 – Key Pair

Add to `main.tf`:

```
tags = { Name = "${var.env_prefix}-default-sg" }
}
resource "aws_key_pair" "serverkey" {
  key_name = "serverkey"
  public_key = file("~/ssh/id_ed25519.pub")
}
```

Step 8 – EC2 Instance

Add to `main.tf`:


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

GNU nano 7.2 main.tf

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

tags = { Name = "${var.env_prefix}-default-sg" }
}
resource "aws_key_pair" "serverkey" {
  key_name   = "serverkey"
  public_key = file("~/ssh/id_ed25519.pub")
}
resource "aws_instance" "web" {
  ami              = "ami-0c02fb55956c7d316"
  instance_type    = var.instance_type
  subnet_id        = aws_subnet.myapp_subnet.id
  vpc_security_group_ids = [aws_security_group.default_sg.id]
  associate_public_ip_address = true
  key_name          = aws_key_pair.serverkey.key_name
  availability_zone = var.availability_zone
  user_data         = file("entry-script.sh")

  tags = { Name = "${var.env_prefix}-ec2-instance" }
}
}
```

– Run Terraform

```
@23-22411-029-sudo →/workspaces/Exam/terraform (main) $ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Finding latest version of hashicorp/http...
- Installing hashicorp/aws v6.28.0...
- Installed hashicorp/aws v6.28.0 (signed by HashiCorp)
- Installing hashicorp/http v3.5.0...
- Installed hashicorp/http v3.5.0 (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.
@23-22411-029-sudo →/workspaces/Exam/terraform (main) $
```

terraform plan

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
@23-22411-029-sudo →/workspaces/Exam/terraform (main) $ terraform plan

+ enclave_options (known after apply)
+ ephemeral_block_device (known after apply)
+ instance_market_options (known after apply)
+ maintenance_options (known after apply)
+ metadata_options (known after apply)
+ network_interface (known after apply)
+ primary_network_interface (known after apply)
+ private_dns_name_options (known after apply)
+ root_block_device (known after apply)
}

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

Changes to Outputs:
+ ec2_instance_id = (known after apply)
+ ec2_public_ip   = (known after apply)

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if y
@23-22411-029-sudo →/workspaces/Exam/terraform (main) $
```

```
@23-22411-029-sudo →/workspaces/Exam/terraform (main) $ ssh-keygen -t ed25519 -f ~/.ssh/id_ed25519 -N ""
Generating public/private ed25519 key pair.
Created directory '/home/codespace/.ssh'.
Your identification has been saved in /home/codespace/.ssh/id_ed25519
Your public key has been saved in /home/codespace/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:ahgyFkYrFqTsgFral0TXz0q17c5X3uH7MbI9o2RFnzE codespace@codespaces-bdb421
The key's randomart image is:
+--[ED25519 256]--+
|.O.
|+...
|+++ + E
|+O... + . =
|..+... .S O.
| O O.O... O.
| ...OO. O *.+.
| O O.. ... + =++
|. .. O+. O.O*
+----[SHA256]-----+
@23-22411-029-sudo →/workspaces/Exam/terraform (main) $
```

terraform apply -auto-approve

```
... PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
CES: STU...
@23-22411-029-sudo →/workspaces/Exam/terraform (main) $ terraform apply -auto-approve

+ maintenance_options (known after apply)
+ metadata_options (known after apply)
+ network_interface (known after apply)
+ primary_network_interface (known after apply)
+ private_dns_name_options (known after apply)
+ root_block_device (known after apply)
}

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

Changes to Outputs:
+ ec2_instance_id = (known after apply)
+ ec2_public_ip   = (known after apply)
aws_instance.web_instance: Creating...
aws_instance.web_instance: Still creating... [00m10s elapsed]
aws_instance.web_instance: Creation complete after 13s [id=i-0c3da813d6ced8733]

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

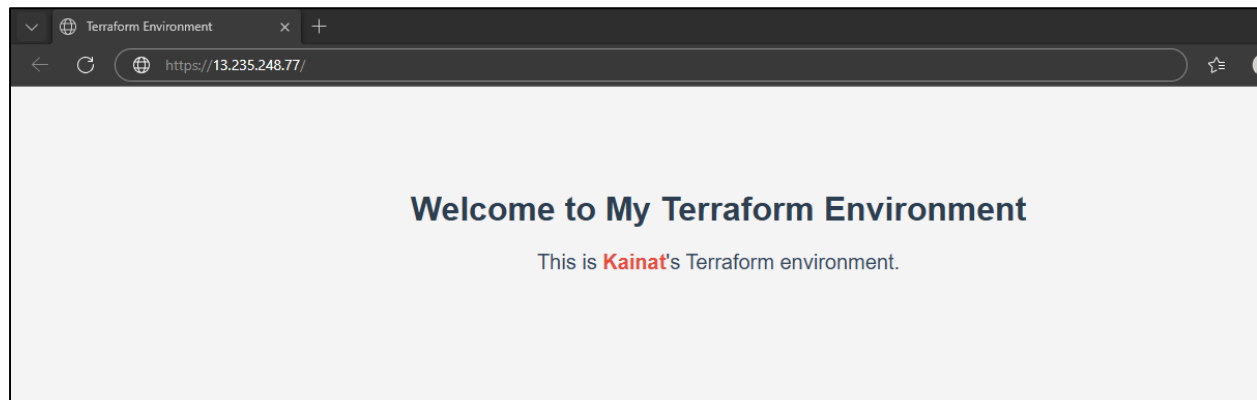
Outputs:

ec2_instance_id = "i-0c3da813d6ced8733"
ec2_public_ip   = "13.235.248.77"
@23-22411-029-sudo →/workspaces/Exam/terraform (main) $
```

terraform output

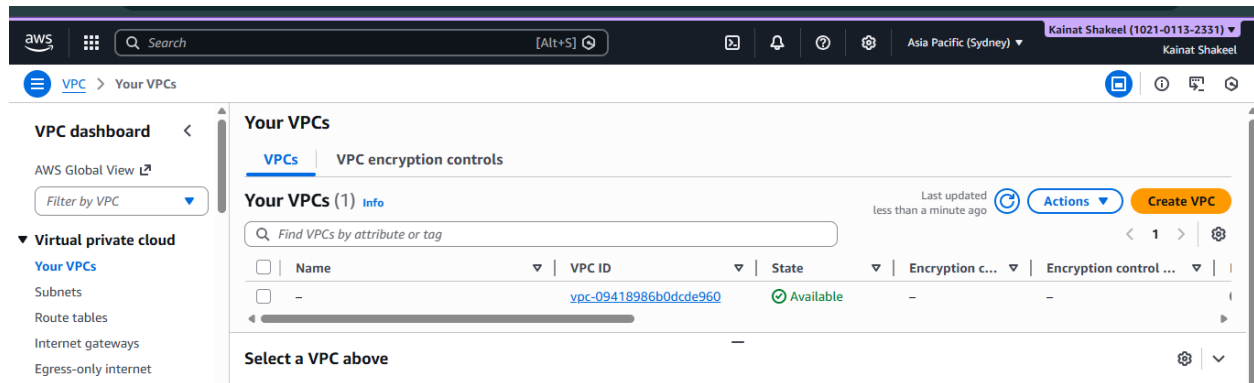
```
● @23-22411-029-sudo →/workspaces/Exam/terraform (main) $ terraform output
ec2_instance_id = "i-0c3da813d6ced8733"
ec2_public_ip   = "13.235.248.77"
@23-22411-029-sudo →/workspaces/Exam/terraform (main) $
```

Apply the Terraform configuration

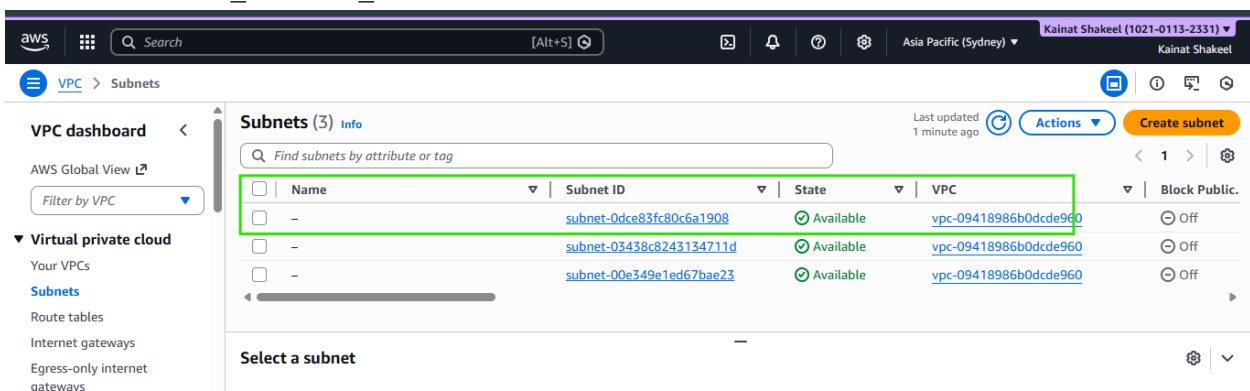


Step 13 – Verify AWS Console

VPC: q2_console_vpc.png



- Subnet: q2_console_subnet.png



- IGW: q2_console_igw.png

aws [Search] [Alt+S] Asia Pacific (Sydney) Kainat Shakeel (1021-0113-2331) Kainat Shakeel

VPC > Internet gateways

Internet gateways (1) Info

Find internet gateways by attribute or tag

<input type="checkbox"/>	Name	Internet gateway ID	State	VPC ID
<input type="checkbox"/>	-	igw-0c9f342a1c669890e	Attached	vpc-09418986b0dcde960

Select an internet gateway above

- **Route Table:** q2_console_route_table.png

aws [Search] [Alt+S] Asia Pacific (Sydney) Kainat Shakeel (1021-0113-2331) Kainat Shakeel

VPC > Route tables

Route tables (1/1) Info

Find route tables by attribute or tag

Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC
Exam	rtb-0ad12994964037594	-	-	Yes	vpc-09418986b0dcde960

rtb-0ad12994964037594 / Exam

- **Security Group:** q2_console_sg.png

aws [Search] [Alt+S] Asia Pacific (Sydney) Kainat Shakeel (1021-0113-2331) Kainat Shakeel

VPC > Security Groups

Security Groups (1) Info

Find security groups by attribute or tag

<input type="checkbox"/>	Name	Security group ID	Security group name	VPC ID
<input type="checkbox"/>	Exam	sg-0d31e0cf78a3fb43d	default	vpc-09418986b0dcde960

DESTROY

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

@23-22411-029-sudo →/workspaces/Exam/terraform (main) $ terraform destroy -auto-approve
}

Plan: 0 to add, 0 to change, 8 to destroy.

Changes to Outputs:
  - ec2_instance_id = "i-0c3da813d6ced8733" -> null
  - ec2_public_ip   = "13.235.248.77" -> null
aws_route_table_association.subnet_assoc: Destroying... [id=rtbassoc-0ca25b417b7bba26a]
aws_instance.web_instance: Destroying... [id=i-0c3da813d6ced8733]
aws_route_table_association.subnet_assoc: Destruction complete after 1s
aws_route_table.myapp_rt: Destroying... [id=rtb-0d1130ae316da95f7]
aws_route_table.myapp_rt: Destruction complete after 0s
aws_internet_gateway.myapp_igw: Destroying... [id=igw-0e99d01944b9e76f4]
aws_instance.web_instance: Still destroying... [id=i-0c3da813d6ced8733, 00m10s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-0e99d01944b9e76f4, 00m10s elapsed]
aws_instance.web_instance: Still destroying... [id=i-0c3da813d6ced8733, 00m20s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-0e99d01944b9e76f4, 00m20s elapsed]
aws_internet_gateway.myapp_igw: Destruction complete after 27s
aws_instance.web_instance: Destruction complete after 30s
aws_subnet.myapp_subnet: Destroying... [id=subnet-016bfe32bc1ebb413]
aws_key_pair.serverkey: Destroying... [id=serverkey]
aws_security_group.default_sg: Destroying... [id=sg-04028c763ba9d230f]
aws_key_pair.serverkey: Destruction complete after 0s
aws_subnet.myapp_subnet: Destruction complete after 0s
aws_security_group.default_sg: Destruction complete after 1s
aws_vpc.myapp_vpc: Destroying... [id=vpc-031bc8db74c63a71a]
aws_vpc.myapp_vpc: Destruction complete after 0s

Destroy complete! Resources: 8 destroyed.
@23-22411-029-sudo →/workspaces/Exam/terraform (main) $
```

Question#3

Step 1: Create the Ansible inventory file (hosts)

```
Destroy complete! Resources: 8 destroyed.
@23-22411-029-sudo →/workspaces/Exam/terraform (main) $ mkdir -p ~/Lab_exam/ansible
cd ~/Lab_exam/ansible
@23-22411-029-sudo →~/Lab_exam/ansible $ nano hosts
@23-22411-029-sudo →~/Lab_exam/ansible $
```

```

@23-22411-029-sudo → ~/Lab_exam/ansible $ timeout 5 curl http://https://13.235.248.77/ 2>&
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload  Total   Spent    Left   Speed
100 1802 100 1802    0     0   8352      0 --:--:-- --:--:-- --:--:--  8342
<!DOCTYPE html>
<html>
<head>
  <title>Terraform Environment</title>
  <style>
    <
      body {
        font-family: Arial, sans-serif;
        margin: 50px;
        background-color: #f5f5f5;
      }
      .container {
        background-color: white;
        padding: 30px;
        border-radius: 8px;
        box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    
```

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
GNU nano 7.2 my-playbook.yml *
X-aws-ec2-metadata-token-ttl-seconds: "21600"
register: imds_token

- name: Get public IPv4 using IMDSv2
  uri:
    url: http://169.254.169.254/latest/meta-data/public-ipv4
    method: GET
    headers:
      X-aws-ec2-metadata-token: "{{ imds_token.content }}"
    register: public_ip

- name: Get public hostname using IMDSv2
  uri:
    url: http://169.254.169.254

```

```

ybook.yml
ok: [51.112.52.191]

TASK [Print public IP address] *****
ok: [51.112.52.191] =>
  msg:
    - 'Public IP: 51.112.52.191'
    - 'Public Hostname: '

TASK [Restart httpd service] *****
changed: [51.112.52.191]

TASK [Verify httpd is running] *****
ok: [51.112.52.191]

TASK [Display httpd status] *****
ok: [51.112.52.191] =>
  msg: Apache HTTPD is active

PLAY RECAP *****
51.112.52.191      : ok=14  changed=5  unreachable=0  failed=0  skipped=1  rescued=0  ignored=0

```

