Lab: Cloud Provisioning

Authors: Olivia Manz and Samuel Roland in **Group D**

See the instructions if necessary.

Lab convention for naming cloud resources

We will use the following **lab naming convention:** the name of every cloud resource starts with the group, followed by our last name and finnaly the type of object

```
GrD-RolandManz-xxx
```

Part 1 - CloudFormation

Task 1.2: Launch a stack

To create the stack, we will choose a existing subnet with Auto-assign public IPv4 address=yes. For the VPC, we choose the VPC with the previous subnet assigned. In our case, we choose:

- subnet id: subnet-032ade9da5a97829a
- vpc: vpc-037435ee28ff909f8

> What tags did CloudFormation add to the EC2 Instance? Which tag is used by CloudFormation to uniquely identify the resource?

Tags of EC2 instance

Key	Value
aws:cloudformation:stack-name	GrD-RolandManz-master
aws:cloudformation:stack-id	arn:aws:cloudformation:us-east-1:067128346383:stack/GrD-RolandManz-master/b4c61160-bf25-11f0-a8ce-129715c41913
Name	LAMP-WebServer
CloudFormationTest	true
aws:cloudformation:logical-id	WebServer

`aws:cloudformation:stack-id` is used by CloudFormation to uniquely identify the resource

> How did CloudFormation name the Security Group and what tags did it add?

The Security Group name is `sg-040d15266e4fbd60b (GrD-RolandManz-master-WebServerSecurityGroup-kIRPaFqz3wei)`

Security group tags:

Key	Value
aws:cloudformation:stack-id	arn:aws:cloudformation:us-east-1:067128346383:stack/GrD-RolandManz-master/b4c61160-bf25-11f0-a8ce-129715c41913

Key	Value
aws:cloudformation:stack-name	GrD-RolandManz-master
CloudFormationTest	true
aws:cloudformation:logical-id	WebServerSecurityGroup

> The template specified an output parameter *WebsiteURL*. What is its value?

In our case the value is `44.211.131.166`. It's the public IPv4 address of EC2 instance created.

> The LAMP Stack template by AWS makes things appear easier than they are. They use an ugly hack to make the selection of the instance type easy for the user of the template, but it makes the template difficult to maintain. Explain the hack.

To make it easier to select the instance type and create a dropdown list, the template hardcodes the possible instance types:

```
  InstanceType:
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - t2.small
      - t2.medium
      - t3.micro
      - t3.small
      - t3.medium
```

It's the same for the `osVersion` and `ImageId` parameters. However, if AWS adds new instance types, the template must be updated manually, it's not dynamic. Moreover, some instance types are not available in all regions. Here, the possible choices are hardcoded to avoid incompatible configurations.

Part 2 - Terraform

```
> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.100.0...
- Installed hashicorp/aws v5.100.0 (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.
```

> What files were created in the terraform directory? Make sure to look also at hidden files and directories (ls -a). What are they used for?

```
> tree -a
.
├── main.tf
├── .terraform
│   └── providers
│       └── registry.terraform.io
│           └── hashicorp
│               └── aws
│                   └── 5.100.0
│                       └── linux_amd64
│                           ├── LICENSE.txt
│                           └── terraform-provider-aws_v5.100.0_x5
└── .terraform.lock.hcl
```

Given this required plugin for AWS mentionned in `main.tf`, terraform needs to pick a version that respects the version constraints `"~> 5.0"` and download this plugin. In the Operators section in Version constraints docs for the configuration language HCL, we find that `~>` *Allows only the right-most version component to increment.* This means that we can have any `5.x` version matching this pattern.

```
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
```

First, let's take about the `.terraform.lock.hcl` which is another lock file (like `package-lock.json`) used to pin the versions of required plugins.

```
provider "registry.terraform.io/hashicorp/aws" {
  version     = "5.100.0"
  constraints = "~> 5.0"
  hashes = [
    "h1:edXOJWE4ORX8Fm+dpVpICzMZJat4AX0VRCAy/xkcOc0=",
    "zh:054b8dd49f0549c9a7cc27d159e45327b7b65cf404da5e5a20da154b90b8a644",
...
```

As we can see it has taken the `5.100.0` version for the `hashicorp/aws` plugin, on the default registry. According to the section New provider package checksums, the `hashes` key contain hashes of the plugin to verify its integrity.

We can also see executable binary of the downloaded plugin under `.terraform/providers/registry.terraform.io/hashicorp/aws/5.100.0/linux_amd64/terraform-provider-aws_v5.100.0_x5`.

Task 2.1: Create a Terraform project and provision an EC2 instance

> After you modified the AMI, copy the output of the command plan into the report and explain it.

```
> terraform plan

aws_instance.app_server: Refreshing state... [id=i-00a5ed928f8423d85]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # aws_instance.app_server must be replaced
-/+ resource "aws_instance" "app_server" {
      ~ ami                                  = "ami-0149b2da6ceec4bb0" -> "ami-08c40ec9ead489470" # forces replacement
      ~ arn                                  = "arn:aws:ec2:us-east-1:067128346383:instance/i-00a5ed928f8423d85" ->
(known after apply)
      ~ associate_public_ip_address          = true -> (known after apply)
      ~ availability_zone                    = "us-east-1a" -> (known after apply)
      ~ cpu_core_count                       = 1 -> (known after apply)
      ~ cpu_threads_per_core                 = 1 -> (known after apply)
      ~ disable_api_stop                     = false -> (known after apply)
      ~ disable_api_termination              = false -> (known after apply)
      ~ ebs_optimized                        = false -> (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      - hibernation                          = false -> null
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      ~ id                                   = "i-00a5ed928f8423d85" -> (known after apply)
      ~ instance_initiated_shutdown_behavior = "stop" -> (known after apply)
      + instance_lifecycle                   = (known after apply)
      ~ instance_state                       = "running" -> (known after apply)
      ~ ipv6_address_count                   = 0 -> (known after apply)
      ~ ipv6_addresses                       = [] -> (known after apply)
      + key_name                             = (known after apply)
      ~ monitoring                           = false -> (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      ~ placement_partition_number           = 0 -> (known after apply)
      ~ primary_network_interface_id         = "eni-0f1d26471015d8861" -> (known after apply)
      ~ private_dns                          = "ip-172-31-24-165.ec2.internal" -> (known after apply)
      ~ private_ip                           = "172.31.24.165" -> (known after apply)
      ~ public_dns                           = "ec2-3-80-50-129.compute-1.amazonaws.com" -> (known after apply)
      ~ public_ip                            = "3.80.50.129" -> (known after apply)
      ~ secondary_private_ips                = [] -> (known after apply)
      ~ security_groups                      = [
          - "default",
        ] -> (known after apply)
      + spot_instance_request_id             = (known after apply)
      ~ subnet_id                            = "subnet-092be4f8b98648a36" -> (known after apply)
        tags                                 = {
            "Course" = "TSM-CloudSys"
            "Group"  = "D"
            "Lab"    = "Terraform"
            "Name"   = "GrD-RolandManz-SecurityGroup"
            "Year"   = "2025"
        }
      ~ tenancy                              = "default" -> (known after apply)
      + user_data                            = (known after apply)
      + user_data_base64                     = (known after apply)
      ~ vpc_security_group_ids               = [
          - "sg-095efe384cb72cc19",
        ] -> (known after apply)
        # (5 unchanged attributes hidden)

      ~ capacity_reservation_specification (known after apply)
      - capacity_reservation_specification {
          - capacity_reservation_preference = "open" -> null
        }

      ~ cpu_options (known after apply)
      - cpu_options {
          - core_count       = 1 -> null
          - threads_per_core = 1 -> null
            # (1 unchanged attribute hidden)
        }

      - credit_specification {
          - cpu_credits = "standard" -> null
        }

      ~ ebs_block_device (known after apply)

      ~ enclave_options (known after apply)
      - enclave_options {
          - enabled = false -> null
        }

      ~ ephemeral_block_device (known after apply)
      ~ instance_market_options (known after apply)

      ~ maintenance_options (known after apply)
      - maintenance_options {
          - auto_recovery = "default" -> null
        }

      ~ metadata_options (known after apply)
      - metadata_options {
          - http_endpoint               = "enabled" -> null
          - http_protocol_ipv6          = "disabled" -> null
          - http_put_response_hop_limit = 1 -> null
          - http_tokens                 = "optional" -> null
          - instance_metadata_tags      = "disabled" -> null
        }

      ~ network_interface (known after apply)

      ~ private_dns_name_options (known after apply)
      - private_dns_name_options {
          - enable_resource_name_dns_a_record    = false -> null
          - enable_resource_name_dns_aaaa_record = false -> null
          - hostname_type                        = "ip-name" -> null
        }

      ~ root_block_device (known after apply)
      - root_block_device {
          - delete_on_termination = true -> null
          - device_name           = "/dev/sda1" -> null
          - encrypted             = false -> null
          - iops                  = 100 -> null
          - tags                  = {} -> null
          - tags_all              = {} -> null
          - throughput            = 0 -> null
          - volume_id             = "vol-098147677725880e0" -> null
          - volume_size           = 8 -> null
          - volume_type           = "gp2" -> null
            # (1 unchanged attribute hidden)
        }
    }

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

As `Plan: 1 to add, 0 to change, 1 to destroy.` and `~ ami = "ami-0149b2da6ceec4bb0" -> "ami-08c40ec9ead489470" # forces replacement` are clearly indicating, the instance has to be destroyed and recreated. A lot of dynamic values (see lines with `(known after apply)`) are marked as `~` (modified) even though we didn't make changes manually.

Only the `tags` section is displayed and will not change. Some other attributes like `instance_type` are probably hidden inside the `# (5 unchanged attributes hidden)`.

Some dynamic parts will obviously need to change, like these networking lines. The public floating IP is changing each time we are renting it.

```
      ~ private_dns                          = "ip-172-31-24-165.ec2.internal" -> (known after apply)
      ~ private_ip                           = "172.31.24.165" -> (known after apply)
      ~ public_dns                           = "ec2-3-80-50-129.compute-1.amazonaws.com" -> (known after apply)
      ~ public_ip                            = "3.80.50.129" -> (known after apply)
```

> How did Terraform implement the change of AMI?

As said above, there is no way we can change the base image of the VM without deleting the old VM and creating a fresh one. This is what Terraform has planned to do. Terraform respects the "immutable infrastructure" principle: instead of updating, it replaces the instance.

Running `terraform apply` again is successfully recreating another instance.

```
...
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_instance.app_server: Destroying... [id=i-00a5ed928f8423d85]
aws_instance.app_server: Still destroying... [id=i-00a5ed928f8423d85, 00m10s elapsed]
aws_instance.app_server: Still destroying... [id=i-00a5ed928f8423d85, 00m20s elapsed]
aws_instance.app_server: Still destroying... [id=i-00a5ed928f8423d85, 00m30s elapsed]
aws_instance.app_server: Still destroying... [id=i-00a5ed928f8423d85, 00m40s elapsed]
aws_instance.app_server: Still destroying... [id=i-00a5ed928f8423d85, 00m50s elapsed]
aws_instance.app_server: Destruction complete after 52s
aws_instance.app_server: Creating...
aws_instance.app_server: Still creating... [00m10s elapsed]
aws_instance.app_server: Creation complete after 15s [id=i-02d36c9be25327b56]

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

Task 2.2: Develop and test Terraform configurations by using LocalStack

The caller identity has changed after defining the alias.

```
> aws sts get-caller-identity
{
    "UserId": "AKIAIOSFODNN7EXAMPLE",
    "Account": "000000000000",
    "Arn": "arn:aws:iam::000000000000:root"
}
```

Now, we see many more regions listed (41 instead of 18): previously, only the regions supported by our account were displayed, whereas now all AWS regions are shown, including those where our account has not yet opted in. For example, regions like ap-east-1 or eu-south-2 are included even though they are not opted-in for our account.

```
> aws ec2 describe-regions
{
    "Regions": [
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "af-south-1",
            "Endpoint": "ec2.af-south-1.amazonaws.com"
        },
        {
            "OptInStatus": "not-opted-in",
            "RegionName": "ap-east-1",
            "Endpoint": "ec2.ap-east-1.amazonaws.com"
        },
        {
            "OptInStatus": "not-opted-in",
            "RegionName": "ap-east-2",
            "Endpoint": "ec2.ap-east-2.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "ap-northeast-1",
            "Endpoint": "ec2.ap-northeast-1.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "ap-northeast-2",
            "Endpoint": "ec2.ap-northeast-2.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "ap-northeast-3",
            "Endpoint": "ec2.ap-northeast-3.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "ap-south-1",
            "Endpoint": "ec2.ap-south-1.amazonaws.com"
        },
        {
            "OptInStatus": "not-opted-in",
            "RegionName": "ap-south-2",
            "Endpoint": "ec2.ap-south-2.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "ap-southeast-1",
            "Endpoint": "ec2.ap-southeast-1.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "ap-southeast-2",
            "Endpoint": "ec2.ap-southeast-2.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "ap-southeast-3",
            "Endpoint": "ec2.ap-southeast-3.amazonaws.com"
        },
        {
            "OptInStatus": "not-opted-in",
            "RegionName": "ap-southeast-4",
            "Endpoint": "ec2.ap-southeast-4.amazonaws.com"
        },
        {
            "OptInStatus": "not-opted-in",
            "RegionName": "ap-southeast-5",
            "Endpoint": "ec2.ap-southeast-5.amazonaws.com"
        },
        {
            "OptInStatus": "not-opted-in",
            "RegionName": "ap-southeast-6",
            "Endpoint": "ec2.ap-southeast-6.amazonaws.com"
        },
        {
            "OptInStatus": "not-opted-in",
            "RegionName": "ap-southeast-7",
            "Endpoint": "ec2.ap-southeast-7.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "ca-central-1",
            "Endpoint": "ec2.ca-central-1.amazonaws.com"
        },
        {
            "OptInStatus": "not-opted-in",
            "RegionName": "ca-west-1",
            "Endpoint": "ec2.ca-west-1.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "eu-central-1",
            "Endpoint": "ec2.eu-central-1.amazonaws.com"
        },
        {
            "OptInStatus": "not-opted-in",
            "RegionName": "eu-central-2",
            "Endpoint": "ec2.eu-central-2.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "eu-north-1",
            "Endpoint": "ec2.eu-north-1.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "eu-south-1",
            "Endpoint": "ec2.eu-south-1.amazonaws.com"
        },
        {
            "OptInStatus": "not-opted-in",
            "RegionName": "eu-south-2",
            "Endpoint": "ec2.eu-south-2.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "eu-west-1",
            "Endpoint": "ec2.eu-west-1.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "eu-west-2",
            "Endpoint": "ec2.eu-west-2.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "eu-west-3",
            "Endpoint": "ec2.eu-west-3.amazonaws.com"
        },
        {
            "OptInStatus": "not-opted-in",
            "RegionName": "il-central-1",
            "Endpoint": "ec2.il-central-1.amazonaws.com"
        },
        {
            "OptInStatus": "not-opted-in",
            "RegionName": "me-central-1",
            "Endpoint": "ec2.me-central-1.amazonaws.com"
        },
        {
            "OptInStatus": "not-opted-in",
            "RegionName": "me-south-1",
            "Endpoint": "ec2.me-south-1.amazonaws.com"
        },
        {
            "OptInStatus": "not-opted-in",
            "RegionName": "mx-central-1",
            "Endpoint": "ec2.mx-central-1.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "sa-east-1",
            "Endpoint": "ec2.sa-east-1.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "us-east-1",
            "Endpoint": "ec2.us-east-1.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "us-east-2",
            "Endpoint": "ec2.us-east-2.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "us-west-1",
            "Endpoint": "ec2.us-west-1.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "us-west-2",
            "Endpoint": "ec2.us-west-2.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "us-gov-east-1",
            "Endpoint": "ec2.us-gov-east-1.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "us-gov-west-1",
            "Endpoint": "ec2.us-gov-west-1.amazonaws.com"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "cn-north-1",
            "Endpoint": "ec2.cn-north-1.amazonaws.com.cn"
        },
        {
            "OptInStatus": "opt-in-not-required",
            "RegionName": "cn-northwest-1",
            "Endpoint": "ec2.cn-northwest-1.amazonaws.com.cn"
        }
    ]
}
```

> When switching from AWS services to LocalStack, why do you need to do an init again? Think about the role of the Terraform state file.

The `tflocal init` command doesn't seem to do anything on the files of the `terraform` folder. This is probably because the state file needs to be reset to avoid considering the AWS infrastructure state as the localstack state. It may also be that since the provider configuration in main.tf still points to the default AWS region without explicitly specifying LocalStack endpoints, Terraform does not detect a change that would require updating the .terraform folder or the lock file. In other words, tflocal might redirect API calls internally without modifying the local Terraform configuration files.

> Show in the report how you verified that the instance was "created" in LocalStack. There is obviously the command `aws ec2 describe-instances` who was previously returning

```
$ aws ec2 describe-instances
{
   "Reservations": []
}
```

It now returns:

```
Reservations": [
       {
           "ReservationId": "r-0aba407d07197ce52",
           "OwnerId": "352909266144",
           "Groups": [],
           "Instances": [
           ...
```

The same applies to the command `aws ec2 describe-volumes`. Previously:

```
$ aws ec2 describe-volumes
{
    "Volumes": []
}
```

It now returns:

```
{
    "Volumes": [
        {
            "AvailabilityZoneId": "use1-az4",
            "Iops": 3000,
            "VolumeType": "gp3",
            "MultiAttachEnabled": false,
            "Throughput": 125,
            "Operator": {
            ...
```

To prove that something fake is running in LocalStack, we can list used services before and after.

```
> localstack status services
┏━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━┓
┃ Service                  ┃ Status      ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━┩
│ acm                      │ ✔ available │
│ apigateway               │ ✔ available │
│ cloudformation           │ ✔ available │
│ cloudwatch               │ ✔ available │
│ config                   │ ✔ available │
│ dynamodb                 │ ✔ available │
│ dynamodbstreams          │ ✔ available │
│ ec2                      │ ✔ available │
│ es                       │ ✔ available │
│ events                   │ ✔ available │
│ firehose                 │ ✔ available │
│ iam                      │ ✔ available │
│ kinesis                  │ ✔ available │
...
```

```
> tflocal init
...
> tflocal plan
...
> tflocal apply

...

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_instance.app_server: Creating...
aws_instance.app_server: Still creating... [00m10s elapsed]
aws_instance.app_server: Creation complete after 10s [id=i-e094d19a69786eec3]

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

The `iam` and `ec2` services are now “running”.

```
> localstack status services
┏━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━┓
┃ Service                  ┃ Status      ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━┩
│ acm                      │ ✔ available │
│ apigateway               │ ✔ available │
│ cloudformation           │ ✔ available │
│ cloudwatch               │ ✔ available │
│ config                   │ ✔ available │
│ dynamodb                 │ ✔ available │
│ dynamodbstreams          │ ✔ available │
│ ec2                      │ ✔ running   │
│ es                       │ ✔ available │
│ events                   │ ✔ available │
│ firehose                 │ ✔ available │
│ iam                      │ ✔ running   │
...
```

Task 2.3: Add a security group

> Modify the Terraform configuration file to add a security group. The security group shall allow incoming traffic for SSH and HTTPS from any IP source address, and allow any outgoing traffic.

We first need to setup a VPC with a name: `main`

```
resource "aws_vpc" "main" {
  cidr_block = "10.0.0.0/16"
}
```

Then, we need to provision a security group based on documentation of `Resource: aws_security_group`, also named `main`.

```
resource "aws_security_group" "main" {
  name        = "GrD-RolandManz-SecurityGroup"
  description = "Allow incoming traffic for SSH and HTTPS from any IP source address, and allow any outgoing traffic."
  tags = {
    Name   = "GrD-RolandManz-SecurityGroup"
    Course = "TSM-CloudSys"
    Year   = "2025"
    Lab    = "Terraform"
    Group  = "D"
  }
}
```

Under the `aws_instance` resource we need to reference the new security group, by adding this line.

```
  # Use the security group created below
  vpc_security_group_ids = [aws_security_group.main.id]
```

We added a first egress rule (traffic from inside the VPC to other external servers) to implement the “allow any outgoing traffic”.

We can reuse these 2 rules from the doc page to allow for IPv4 and IPv6. We need to reference the `security_group_id` so Terraform know where to apply the rule.

```
resource "aws_vpc_security_group_egress_rule" "allow_all_traffic_ipv4" {
  security_group_id = aws_security_group.main.id
  cidr_ipv4         = "0.0.0.0/0"
  ip_protocol       = "-1" # semantically equivalent to all ports
}

resource "aws_vpc_security_group_egress_rule" "allow_all_traffic_ipv6" {
  security_group_id = aws_security_group.main.id
  cidr_ipv6         = "::/0"
  ip_protocol       = "-1" # semantically equivalent to all ports
}
```

To allow SSH and HTTPS, we just need to authorize the ports 443 and 22 with an ingress rules (incoming traffic).

Here as we want a single port 443, we have to specify a range of 1 element 443 -> 443 via `from_port` and `to_port`. (Source). We want to authorize any source IP, therefore we define `cidr_ipv4         = "0.0.0.0/0"`.

```
resource "aws_vpc_security_group_ingress_rule" "allow_incoming_https" {
  security_group_id = aws_security_group.main.id
  cidr_ipv4         = "0.0.0.0/0"
  from_port         = 443
  ip_protocol       = "tcp"
  to_port           = 443
}
```

And we can do the same for port 22

```
resource "aws_vpc_security_group_ingress_rule" "allow_incoming_ssh" {
  security_group_id = aws_security_group.main.id
  cidr_ipv4         = "0.0.0.0/0"
  from_port         = 22
  ip_protocol       = "tcp"
  to_port           = 22
}
```

To add the required output, we just need to add these 2 items

```
output "instance_public_ip" {
  description = "Private IP address of the EC2 instance"
  value       = aws_instance.app_server.private_ip
}

output "instance_id" {
  description = "EC2 instance ID"
  value       = aws_instance.app_server.id
}
```

Result example

```
instance_id = "i-3089edcd0bca9fe5a"
instance_public_ip = "10.188.234.187"
```

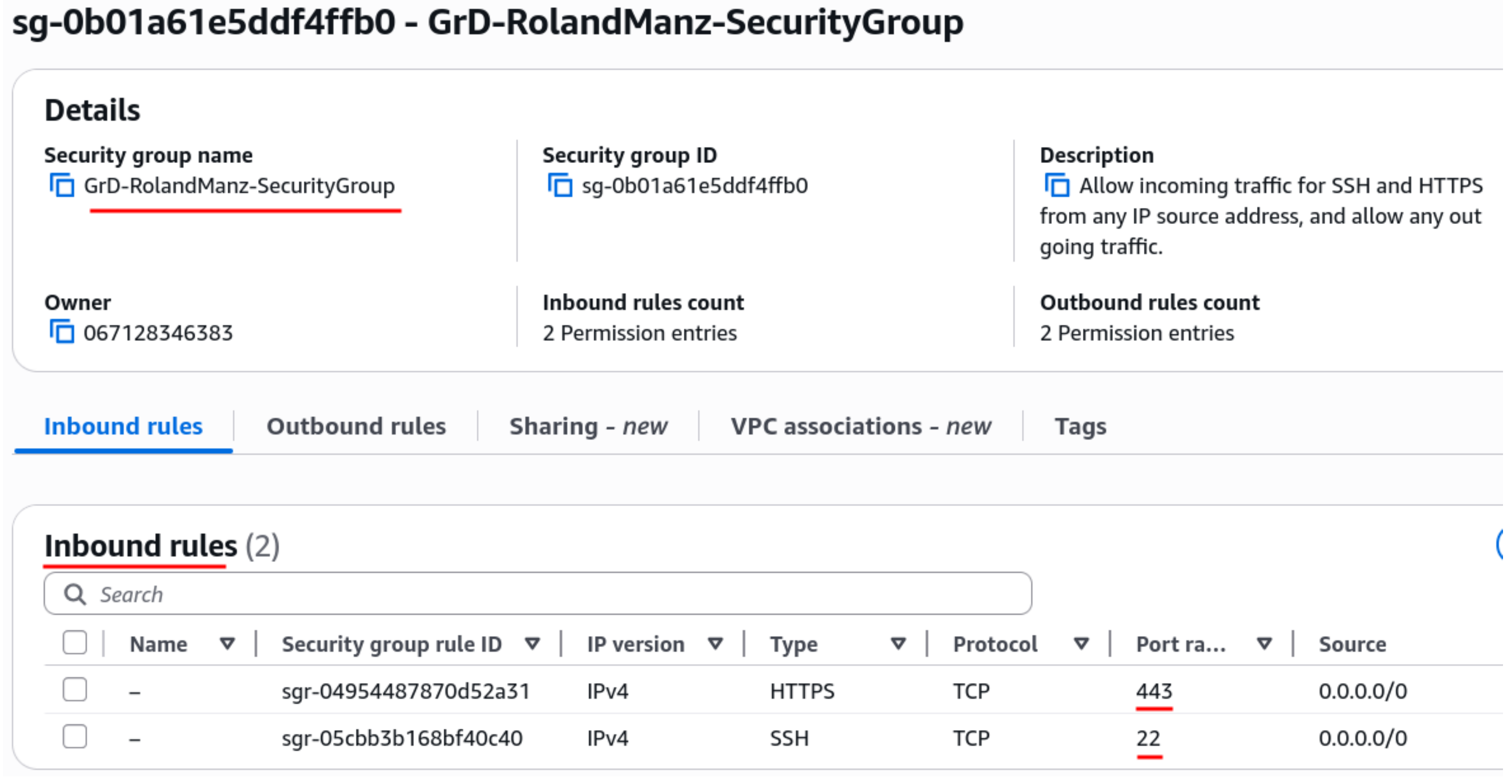
Here is the output resulted from `terraform apply`

```
> terraform apply

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.app_server will be created
  + resource "aws_instance" "app_server" {
      + ami                                  = "ami-08c40ec9ead489470"
      + 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)
      + enable_primary_ipv6                  = (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                                 = {
          + "Course" = "TSM-CloudSys"
          + "Group"  = "D"
          + "Lab"    = "Terraform"
          + "Name"   = "GrD-RolandManz-Instance"
          + "Year"   = "2025"
        }
      + tags_all                             = {
          + "Course" = "TSM-CloudSys"
          + "Group"  = "D"
          + "Lab"    = "Terraform"
          + "Name"   = "GrD-RolandManz-Instance"
          + "Year"   = "2025"
        }
      + 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)

      + capacity_reservation_specification (known after apply)

      + cpu_options (known after apply)

      + ebs_block_device (known after apply)

      + 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)

      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

  # aws_security_group.main will be created
  + resource "aws_security_group" "main" {
      + arn                    = (known after apply)
      + description            = "Allow incoming traffic for SSH and HTTPS from any IP source address, and allow any
outgoing traffic."
      + egress                 = (known after apply)
      + id                     = (known after apply)
      + ingress                = (known after apply)
      + name                   = "GrD-RolandManz-SecurityGroup"
      + name_prefix            = (known after apply)
      + owner_id               = (known after apply)
      + revoke_rules_on_delete = false
      + tags                   = {
          + "Course" = "TSM-CloudSys"
          + "Group"  = "D"
          + "Lab"    = "Terraform"
          + "Name"   = "GrD-RolandManz-SecurityGroup"
          + "Year"   = "2025"
        }
      + tags_all               = {
          + "Course" = "TSM-CloudSys"
          + "Group"  = "D"
          + "Lab"    = "Terraform"
          + "Name"   = "GrD-RolandManz-SecurityGroup"
          + "Year"   = "2025"
        }
      + vpc_id                 = (known after apply)
    }

  # aws_vpc.main will be created
  + resource "aws_vpc" "main" {
      + arn                                  = (known after apply)
      + cidr_block                           = "10.0.0.0/16"
      + 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)
    }

  # aws_vpc_security_group_egress_rule.allow_all_traffic_ipv4 will be created
  + resource "aws_vpc_security_group_egress_rule" "allow_all_traffic_ipv4" {
      + arn                    = (known after apply)
      + cidr_ipv4              = "0.0.0.0/0"
      + id                     = (known after apply)
      + ip_protocol            = "-1"
      + security_group_id      = (known after apply)
      + security_group_rule_id = (known after apply)
      + tags_all               = {}
    }

  # aws_vpc_security_group_egress_rule.allow_all_traffic_ipv6 will be created
  + resource "aws_vpc_security_group_egress_rule" "allow_all_traffic_ipv6" {
      + arn                    = (known after apply)
      + cidr_ipv6              = "::/0"
      + id                     = (known after apply)
      + ip_protocol            = "-1"
      + security_group_id      = (known after apply)
      + security_group_rule_id = (known after apply)
      + tags_all               = {}
    }

  # aws_vpc_security_group_ingress_rule.allow_incoming_https will be created
  + resource "aws_vpc_security_group_ingress_rule" "allow_incoming_https" {
      + arn                    = (known after apply)
      + cidr_ipv4              = "0.0.0.0/0"
      + from_port              = 443
      + id                     = (known after apply)
      + ip_protocol            = "tcp"
      + security_group_id      = (known after apply)
      + security_group_rule_id = (known after apply)
      + tags_all               = {}
      + to_port                = 443
    }

  # aws_vpc_security_group_ingress_rule.allow_incoming_ssh will be created
  + resource "aws_vpc_security_group_ingress_rule" "allow_incoming_ssh" {
      + arn                    = (known after apply)
      + cidr_ipv4              = "0.0.0.0/0"
      + from_port              = 22
      + id                     = (known after apply)
      + ip_protocol            = "tcp"
      + security_group_id      = (known after apply)
      + security_group_rule_id = (known after apply)
      + tags_all               = {}
      + to_port                = 22
    }

Plan: 7 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + instance_id        = (known after apply)
  + instance_public_ip = (known after apply)

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.main: Creating...
aws_security_group.main: Creating...
aws_vpc.main: Creation complete after 5s [id=vpc-082fe8826e70c91ed]
aws_security_group.main: Creation complete after 6s [id=sg-0b01a61e5ddf4ffb0]
aws_vpc_security_group_egress_rule.allow_all_traffic_ipv6: Creating...
aws_vpc_security_group_ingress_rule.allow_incoming_https: Creating...
aws_vpc_security_group_egress_rule.allow_all_traffic_ipv4: Creating...
aws_vpc_security_group_ingress_rule.allow_incoming_ssh: Creating...
aws_instance.app_server: Creating...
aws_vpc_security_group_egress_rule.allow_all_traffic_ipv4: Creation complete after 0s [id=sgr-0dcf75732315524d2]
aws_vpc_security_group_ingress_rule.allow_incoming_ssh: Creation complete after 0s [id=sgr-05cbb3b168bf40c40]
aws_vpc_security_group_egress_rule.allow_all_traffic_ipv6: Creation complete after 1s [id=sgr-0c16a4ed43524d28c]
aws_vpc_security_group_ingress_rule.allow_incoming_https: Creation complete after 1s [id=sgr-04954487870d52a31]
aws_instance.app_server: Still creating... [00m10s elapsed]
aws_instance.app_server: Creation complete after 16s [id=i-02cb690fb56ebbcfd]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:

instance_id = "i-02cb690fb56ebbcfd"
instance_public_ip = "172.31.28.85"
```

The final `main.tf` is here

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }

  required_version = ">= 1.2.0"
}

provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "app_server" {
  ami           = "ami-08c40ec9ead489470" # Ubuntu Server 22.04 LTS (HVM), SSD Volume Type
  instance_type = "t2.micro"

  tags = {
    Name   = "GrD-RolandManz-Instance"
    Course = "TSM-CloudSys"
    Year   = "2025"
    Lab    = "Terraform"
    Group  = "D"
  }
  # Use the security group created below
  vpc_security_group_ids = [aws_security_group.main.id]
}

resource "aws_vpc" "main" {
  cidr_block = "10.0.0.0/16"
}

resource "aws_security_group" "main" {
  name        = "GrD-RolandManz-SecurityGroup"
  description = "Allow incoming traffic for SSH and HTTPS from any IP source address, and allow any outgoing traffic."
  tags = {
    Name   = "GrD-RolandManz-SecurityGroup"
    Course = "TSM-CloudSys"
    Year   = "2025"
    Lab    = "Terraform"
    Group  = "D"
  }
}

resource "aws_vpc_security_group_egress_rule" "allow_all_traffic_ipv4" {
  security_group_id = aws_security_group.main.id
  cidr_ipv4         = "0.0.0.0/0"
  ip_protocol       = "-1" # semantically equivalent to all ports
}

resource "aws_vpc_security_group_egress_rule" "allow_all_traffic_ipv6" {
  security_group_id = aws_security_group.main.id
  cidr_ipv6         = "::/0"
  ip_protocol       = "-1" # semantically equivalent to all ports
}

resource "aws_vpc_security_group_ingress_rule" "allow_incoming_https" {
  security_group_id = aws_security_group.main.id
  cidr_ipv4         = "0.0.0.0/0"
  from_port         = 443
  ip_protocol       = "tcp"
  to_port           = 443
}

resource "aws_vpc_security_group_ingress_rule" "allow_incoming_ssh" {
  security_group_id = aws_security_group.main.id
  cidr_ipv4         = "0.0.0.0/0"
  from_port         = 22
  ip_protocol       = "tcp"
  to_port           = 22
}

output "instance_public_ip" {
  description = "Private IP address of the EC2 instance"
  value       = aws_instance.app_server.private_ip
}

output "instance_id" {
  description = "EC2 instance ID"
  value       = aws_instance.app_server.id
}
```

> Copy a screenshot of the AWS Management Console showing the details of the security group into the report.



sg-0b01a61e5ddf4ffb0 - GrD-RolandManz-SecurityGroup

Details

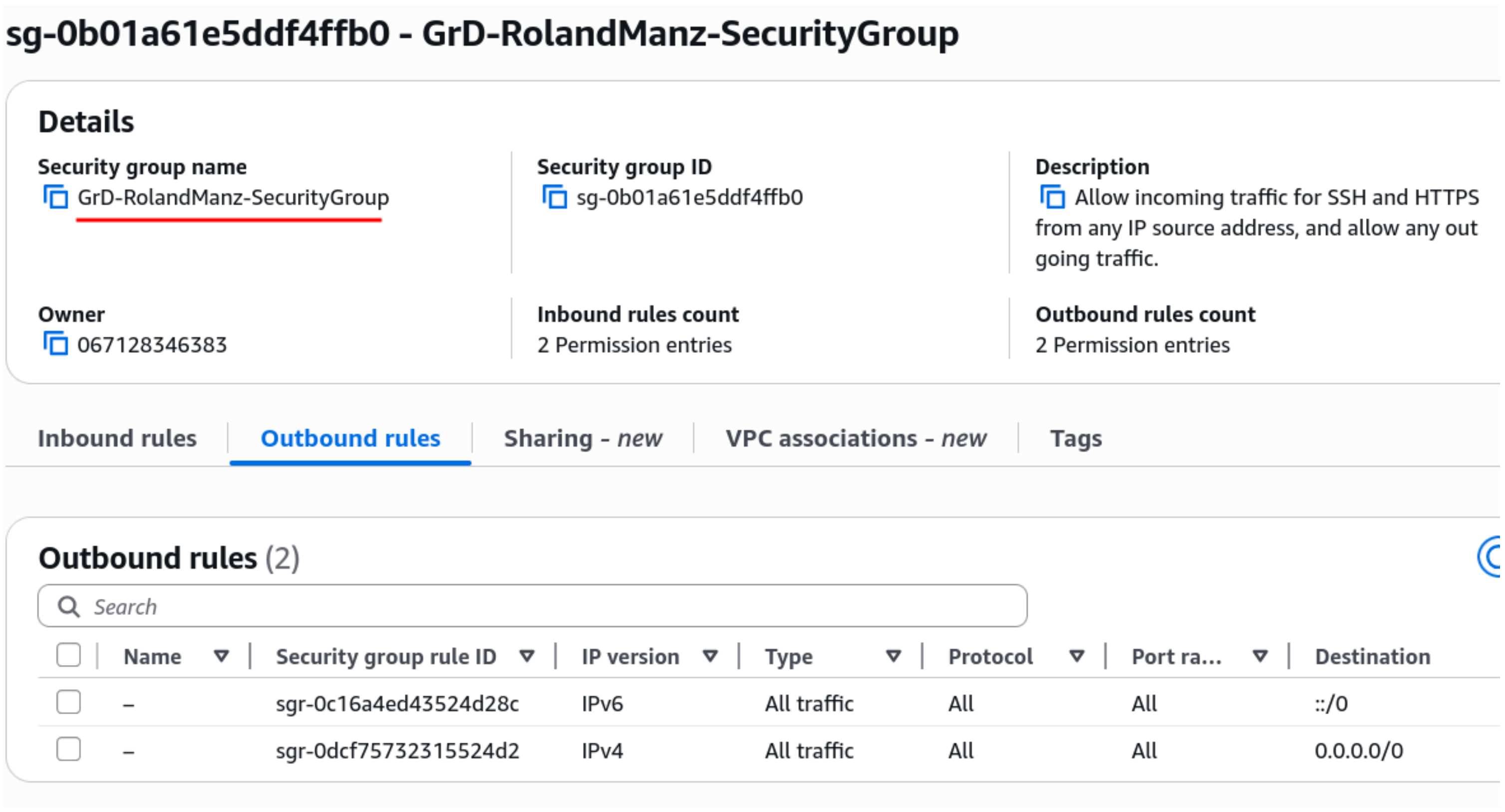
Security group name	Security group ID	Description
GrD-RolandManz-SecurityGroup	sg-0b01a61e5ddf4ffb0	Allow incoming traffic for SSH and HTTPS from any IP source address, and allow any out going traffic.
Owner	**Inbound rules count**	**Outbound rules count**
067128346383	2 Permission entries	2 Permission entries

Inbound rules | Outbound rules | Sharing - *new* | VPC associations - *new* | Tags

Inbound rules (2)

Search

Name	Security group rule ID	IP version	Type	Protocol	Port ra...	Source
–	sgr-04954487870d52a31	IPv4	HTTPS	TCP	443	0.0.0.0/0
–	sgr-05cbb3b168bf40c40	IPv4	SSH	TCP	22	0.0.0.0/0



sg-0b01a61e5ddf4ffb0 - GrD-RolandManz-SecurityGroup

Details

Security group name	Security group ID	Description
GrD-RolandManz-SecurityGroup	sg-0b01a61e5ddf4ffb0	Allow incoming traffic for SSH and HTTPS from any IP source address, and allow any out going traffic.
Owner	**Inbound rules count**	**Outbound rules count**
067128346383	2 Permission entries	2 Permission entries

Inbound rules | Outbound rules | Sharing - *new* | VPC associations - *new* | Tags

Outbound rules (2)

Search

Name	Security group rule ID	IP version	Type	Protocol	Port ra...	Destination
–	sgr-0c16a4ed43524d28c	IPv6	All traffic	All	All	::/0
–	sgr-0dcf75732315524d2	IPv4	All traffic	All	All	0.0.0.0/0