**AWS Production Project Implementation**

* Users will access the application throw the **internet gateway**
* To the **application load balancers** then with reach the application

**Overview of the project**

* The VPC has a public subnet and private subnet in two availability zone
* Each public subnet contains a **NAT gateway** and **application load balancers**
* When an application needs to access any data from the internet a private IP address is transferred to the NAT then the NAT GATEWAY Makes the IP address then it will reach the internet
* It provides security to the application

**Auto-scaling group :**

* It will scale the application according to the number of requests coming to the application

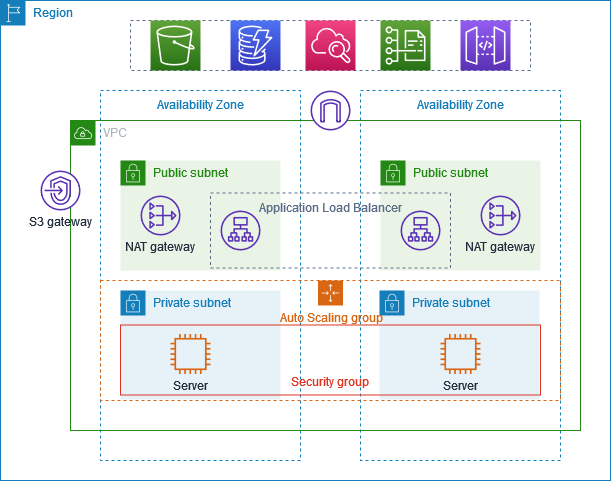
**Load balancer :**

* It will distribute the request application based on the server performances
* **Routing mechanism**

1. **Path-based routing**
2. **Host-based routing**

**Bastioner hosting or jump server :**

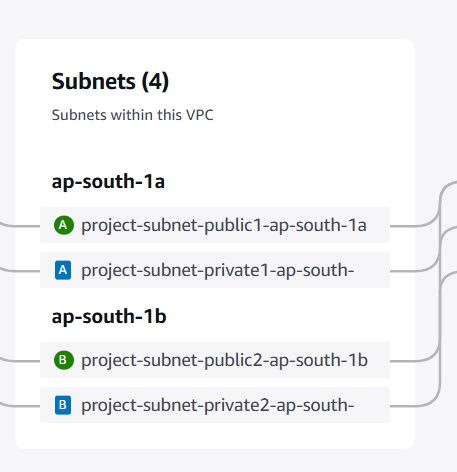
* The application is created in a **private subnet**
* So we do not **SHH** into these instances directly because we do not have a public IP address
* **the advantage of bastion hosting**
* they will proper logging mechanism



**1. Create the VPC**

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. On the dashboard, choose **Create VPC**.
3. For **Resources to create**, choose **VPC and more**

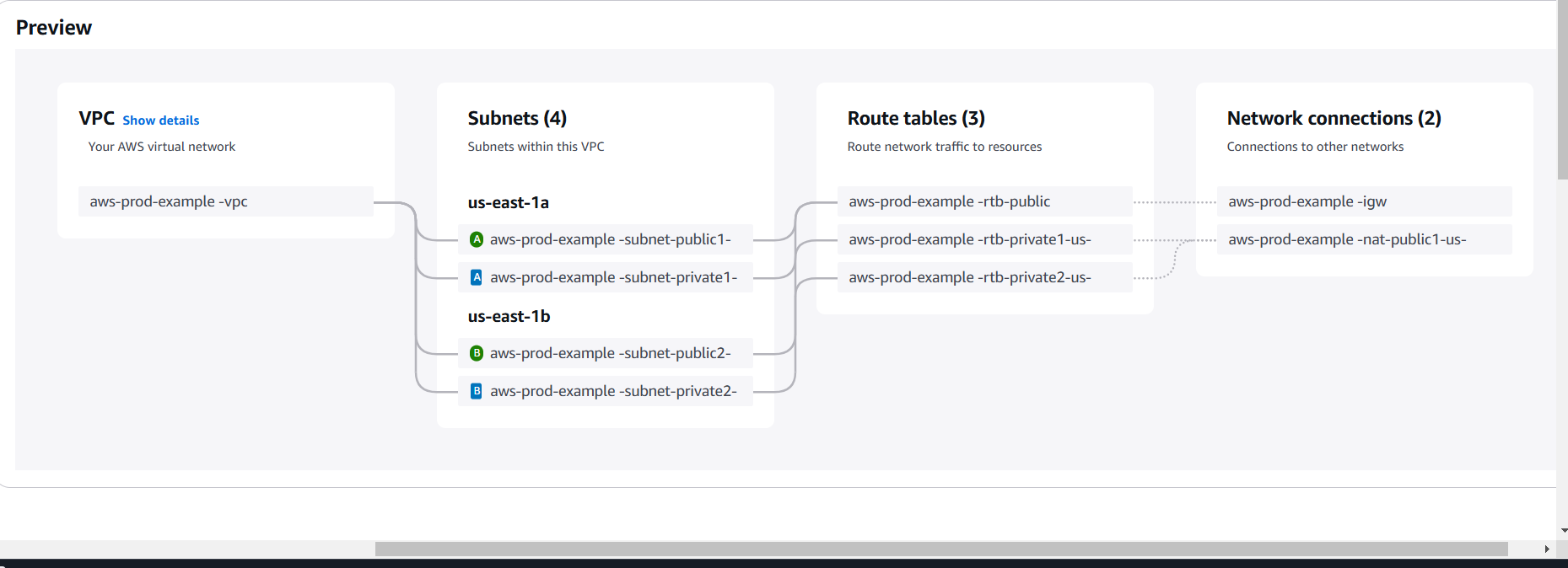
we choose VPC and more it will create 4 subnets in two availability zone



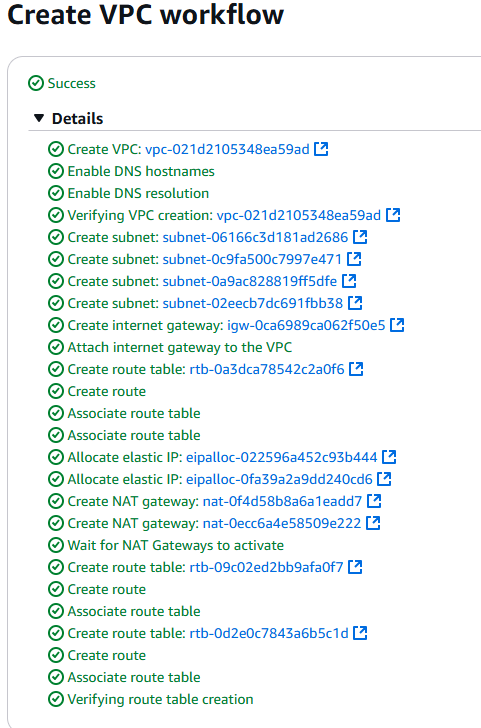
Route tables

* It will define how to route the traffic within the subnet
* It two public subnets is connected to the one route table
* It destination as an internet gateway
* The private sub-addresses are connected to the two different route table
* This are connected to the vice endpoint

1. **Configure the VPC**
   1. For **Name tag auto-generation**, enter a name for the VPC.
   2. For **IPv4 CIDR block**, you can keep the default suggestion, or alternatively you can enter the CIDR block required by your application or network.
   3. If your application communicates by using IPv6 addresses, choose **IPv6 CIDR block**, **Amazon-provided IPv6 CIDR block**.
2. **Configure the subnets**
   1. For **Number of Availability Zones**, choose **2**, so that you can launch instances in multiple Availability Zones to improve resiliency.
   2. For **Number of public subnets**, choose **2**.
   3. For **Number of private subnets**, choose **2**.
   4. You can keep the default CIDR block for the public subnet, or alternatively you can expand **Customize subnet CIDR blocks** and enter a CIDR block. For more information, see [Subnet CIDR blocks](https://docs.aws.amazon.com/vpc/latest/userguide/subnet-sizing.html).
3. For **NAT gateways**, choose **1 per AZ** to improve resiliency.
4. If your application communicates by using IPv6 addresses, for **Egress only internet gateway**, choose **Yes**.
5. For **VPC endpoints**, if your instances must access an S3 bucket, keep the **S3 Gateway** default. Otherwise, instances in your private subnet can't access Amazon S3. There is no cost for this option, so you can keep the default if you might use an S3 bucket in the future. If you choose **None**, you can always add a gateway VPC endpoint later on.
6. For **DNS options**, clear **Enable DNS hostnames**.
7. Choose **Create VPC**.



By using the vpc and more we will find the services that are created by the aws automatical with involvement of manually



1. **Deploy your application**

create a launch template to specify the configuration information needed to launch your ec2 instances by using Amazon ec2 auto-scaling**.**

**Create launch template**

* **Name = aws -prod-exam**
* **Description**
* **Os = ubuntu select free version**
* **Key pair**
* **Security Group** 
  + 1. **Name**
    2. **Description**
    3. **Add vpc**
* **Inbound rule**
* **All port**

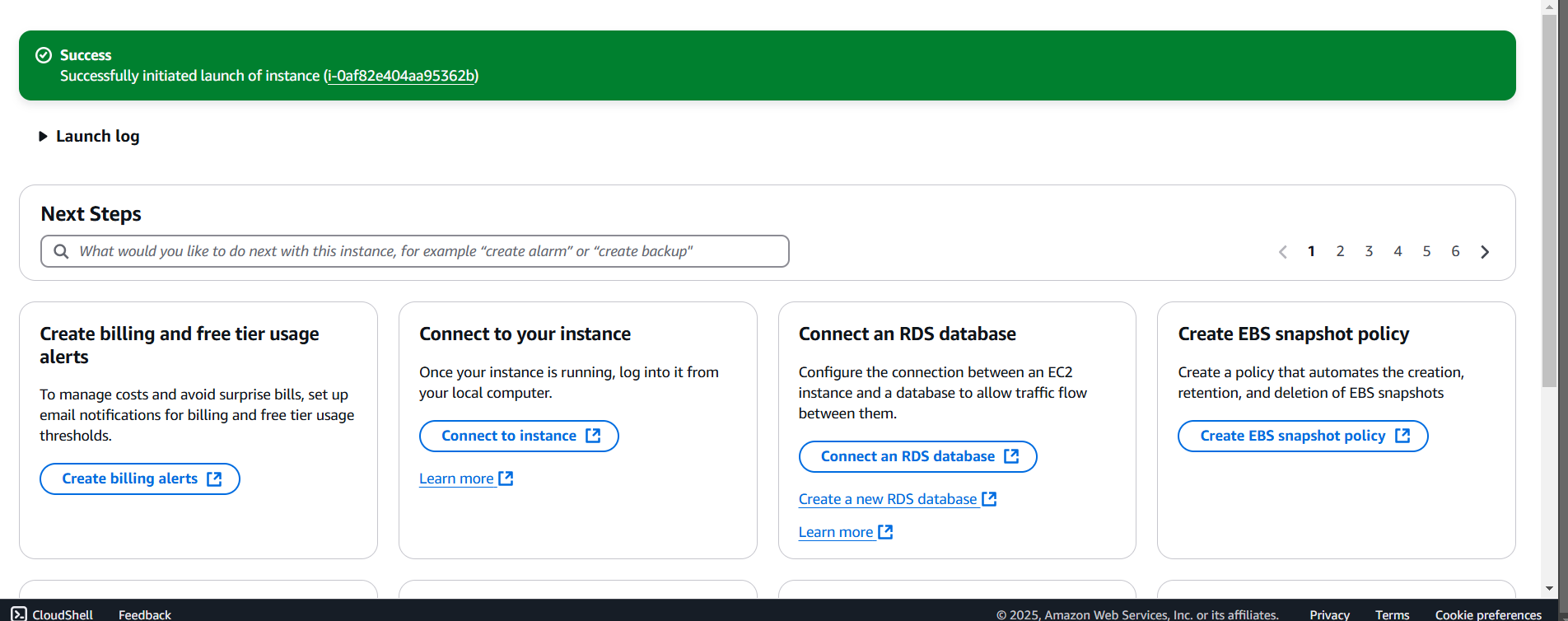
**Security group for application inside the private instances**

* **All port**
* **Anywhere**

**Launch the template**

**After launching two instances in the private subnet then we need to create bashing hosting on the instances**

* **Create a other instance in the same vpc**
* **Enable auto sing ip address**

****

* **Cope the pem file**

**scp -i "C:\Users\gopic\Downloads\autokey12.pem" "C:\Users\gopic\Downloads\autokey12.pem"** [**ubuntu@18.212.85.52:/home/ubuntu/**](mailto:ubuntu@18.212.85.52:/home/ubuntu/)

* **login to jump host instances**
* **install the html in private1 instances**

**go to ec2 and create load balancer**

* **the application load balancer distribute the HTTP and HTTPS traffic across the multiple targets such as ec2 instances**
* **the application load balancer should be in the public subnet**
* **create application**
* **create template for aws**
* **select only public subnets**
* **create template to target group**