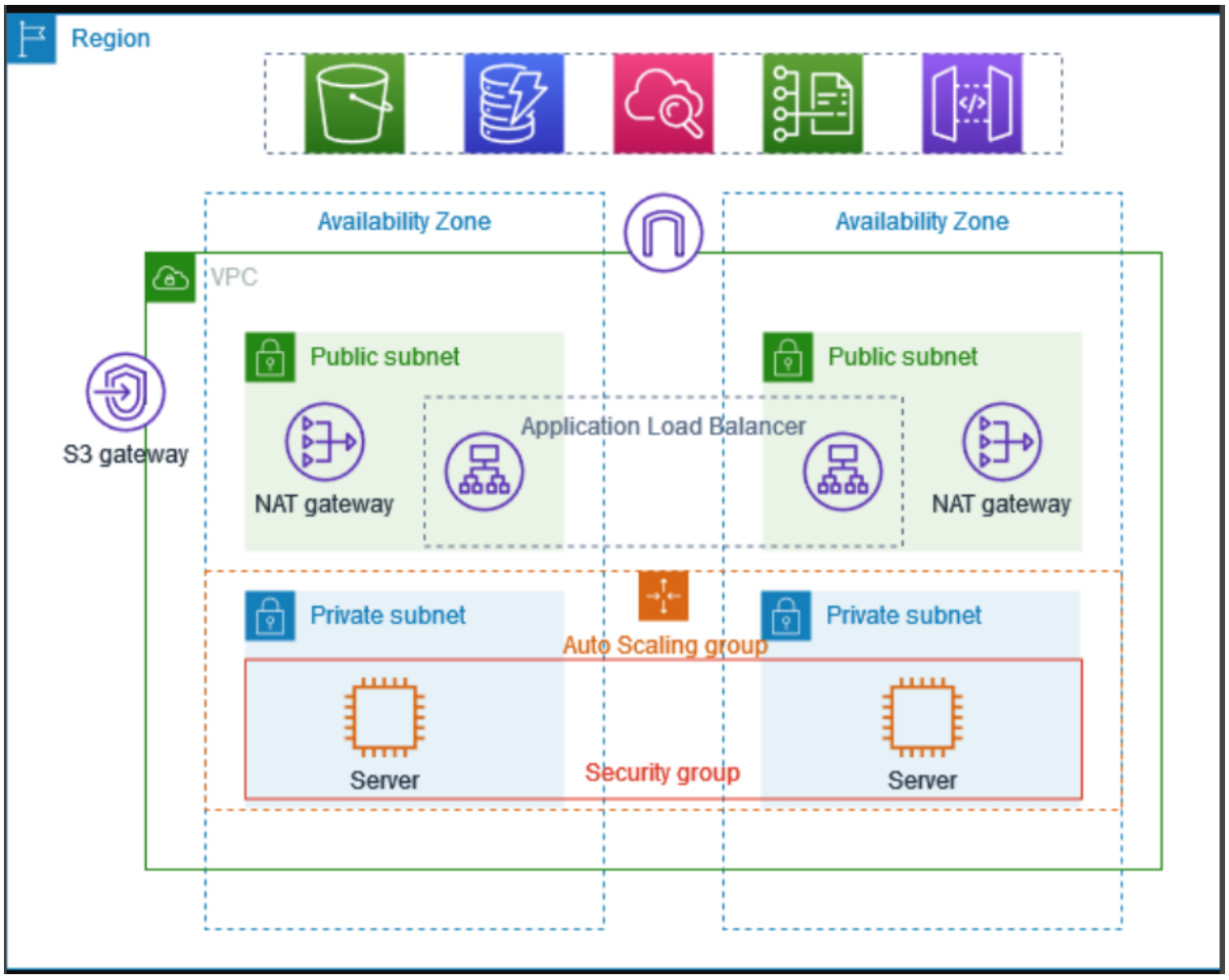
Application deployment within a VPC(Virtual Private Cloud)

Pre-reqs :

- **All you need to have is an AWS account(Free tier)**

VPC Public Private subnet Architecture with a Load Balancer & Auto-Scaling group :



- **So, the idea here is we are going to create a VPC(Highly secured isolated virtual private cloud environment) which comes with a size of 65,536 IPs. In which we are going to create a public subnet and a private subnet (Subnet is nothing but a list of IP ranges).**
- **We are going to deploy our application in an EC2 instance which is inside a private subnet. Since, security is our primary concern we won't be having a public IP enabled for this instance.**

- **In the public subnet we will be having an EC2 instance, which is called jump server/Bastion server.**
- **We will login to that Jump server using its Public IP and then from there I will SSH to the private IP of my instance which is inside the private subnet where my application will be deployed.**
- **There are more components that are used in the architecture: Internet Gateway, NAT Gateway, Application Load Balance(ALB), Auto Scaling group(ASG), Security Group(SG) & NACL(Network Access Control List).**
- **Internet Gateway : Through which the user will access the application.**
- **NAT Gateway : If my application inside the private subnet wants to access or download a few packages from the internet, then NAT Gateway helps me in accessing the internet and at the same time masks the IP address of my private instance. So that the IP address won't be exposed to hackers.**
- **ALB : distributes the incoming traffic/load across the instances.**
- **ASG : This is used to scale in(remove)/scale out(add) EC2 instances based on the Demand/Load.**
- **SG : Acts as a Firewall at the instance level.**
- **NACL : Acts as a Firewall at the subnet level.**
- **Target Groups : Within the ALB there will be a target group, which will route the requests to individual targets(EC2) using the protocol & port.**
- **Route Table : It displays where the traffic is directed.**
- **Things to note, private subnets will only have the EC2 instance where the application is hosted. Public subnet will have jump server, ALB & NAT gateway**

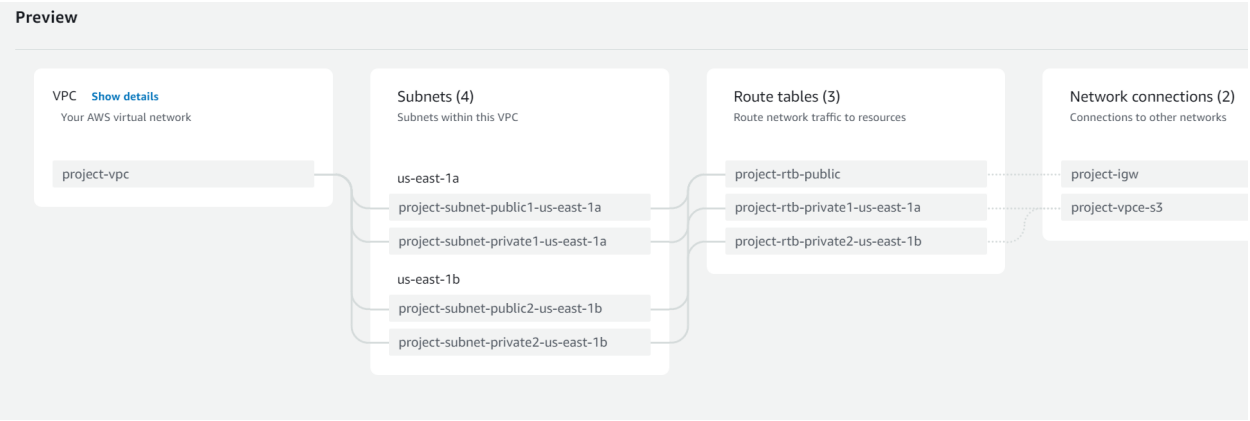
Note :

1. **For High Availability, we are doing this deployment on multiple AZs(Availability Zones).**
2. **I am launching my EC2's within the Auto-scaling group as I want to scale in/scale out the EC2 instances to handle the incoming load.**
3. **VPC is spread across multiple AZs but public & private subnets are restricted to a specific AZ.**

Steps to configure :

Create VPC

1. **Login to AWS management console and look for VPC in the search.**
2. **Now click on create VPC.**
3. **Select VPC and more, just have a look at the preview, You would notice a private & public subnet in each AZ, rtb(route table).**



4. Next, Select Number of Availability Zones (AZs) as 2 or 3.
5. Select Number of public subnets and private subnets as 2.
6. In order to access internet from the private subnet, select NAT gateways as 1 per AZ.\
7. Mark the VPC endpoints as None (As it is Not required).
8. Look at the preview and click “Create VPC”.

Create & Configure ASG

- Go to the EC2 dashboard and Click on the Auto scaling group which is in the bottom left corner. Click Create Auto scaling group.
- Enter a ASG name and create a launch template(this is same as creating an EC2 instance by selecting instance type, instance AMI), In the Network settings select ‘Create security group’, add security group name, description, select the VPC that we created. In the inbound rules open port 22 for ssh and port 8000 - this is the port on which my splunk Enterprise application runs.
- Finally Click on create launch template.
- Then go back to the ASG tab and select the template we created and click next.
- In the next tab select the VPC and select Availability Zones and subnets as private subnet 1 and 2. Click Next
- Then click Next. In “Configure group size and scaling policies” section mark desired as 2, min as 1, max as 4. This is because I want 2 instances in each AZ as default and if the load increases/decreases, add/remove the instances based on this configuration. Click Next.
- Go to the last tab and click “Create Auto Scaling group”.

Create & Configure ALB

- Go to the EC2 dashboard and Click on the Load balancers, which is in the bottom left corner. Click Create Load balancers and select Application Load Balancer.

- ➢ Enter a ALB name and select the VPC, mark both the AZs and select the public subnets from the dropdown.
- ➢ Select the Security Group which we created earlier. Then create a target group with port 80 and HTTP as protocol. Click Next.(Note :Open port 80 which is an HTTP protocol port in the existing security group).
- ➢ Later on, mark those instances which ASG created. Update the port as 8000(Application running port). Click on include and Finish.
- ➢ Go back to the ALB tab and refresh. Select the Target group that we just created and lastly click on create load balancer.

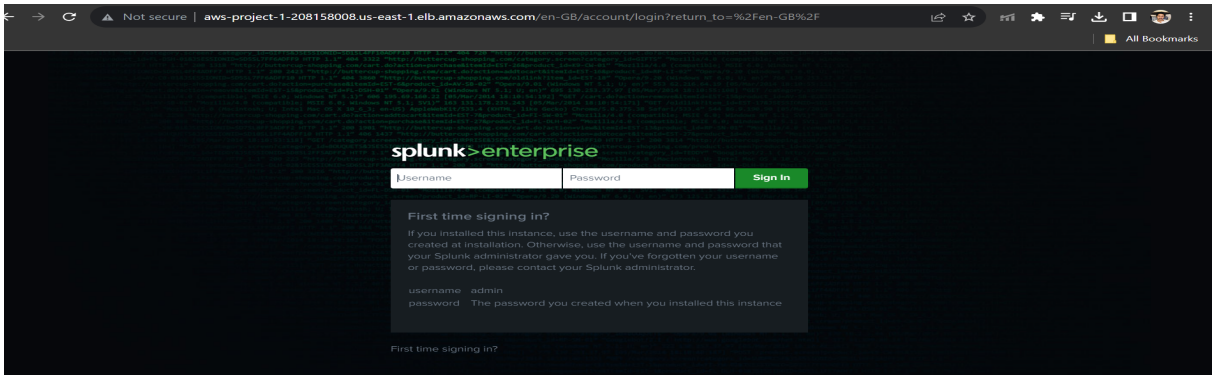
Create & configure Jump Server

1. Go to ec2 dashboard and click on launch instance.
2. Name it as Bastion/Jump server.
3. Select OS as ubuntu, instance is the free tier instance and the existing key value pair.
4. In the network settings, select the VPC we created and subnet as either public1/public2.
5. Enable the Auto assign public IP.
6. Click on Launch Instance.

Now, Copy the key value pair in your local machine to the Jump server, as you need it to SSH to the EC2 instances which are private subnets. You need to use Private IPs for instances in private subnet.

Splunk Installation:

1. SSH to the any one of the instance in the private subnet from the jump server.
2. Install splunk using the below wget command.
3. wget -O splunk-9.1.1-64e843ea36b1-Linux-x86_64.tgz "https://download.splunk.com/products/splunk/releases/9.1.1/linux/splunk-9.1.1-64e843ea36b1-Linux-x86_64.tgz"
4. Post downloading the tar file untar it. Just do tar -xvzf <filename>
5. Now start splunk. Go inside splunk/bin directory and do ./splunk start.
6. Now, lets test the UI is working or not using the Load balancer DNS.



7. **As you can see the Application's UI is up and running. Repeat the same steps in the other node as well.**