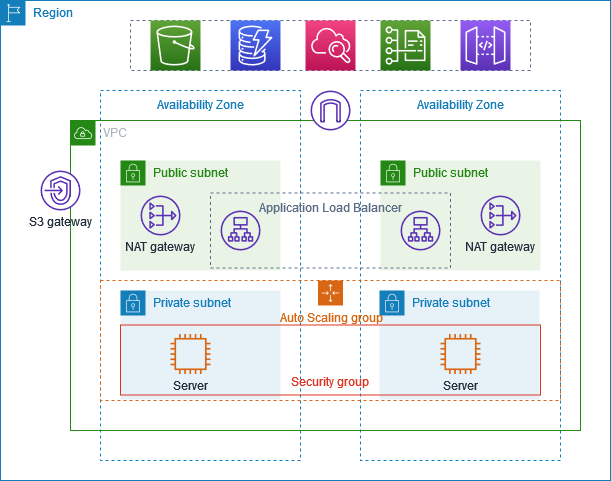
**AWS – Project Using VPC**

**🚀 AWS Cloud Infrastructure Project: EC2, VPC, Auto Scaling, Load Balancer & Bastion Host 🌐**

🏗️ Architecture Diagram:



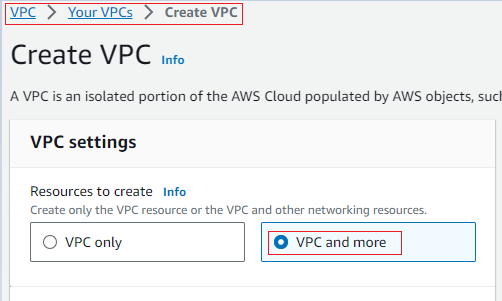
**Key Components in the architecture**

To achieve the project’s objectives, we will explore and implement the following key components:

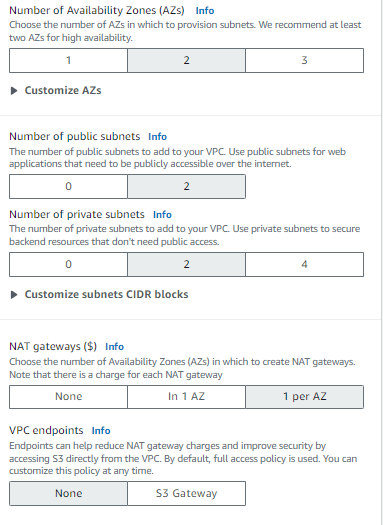
1. **Amazon Virtual Private Cloud (Amazon VPC):** A logically isolated section of the AWS cloud where we will define our network and deploy our resources.
2. **Public Subnet:** A subnet within the VPC where resources that require direct internet access will reside. This includes components like load balancers and web servers.
3. **Private Subnet:** A subnet that is not directly accessible from the internet. Resources that store sensitive data or backend servers will be placed here, enhancing security.
4. **Network Address Translation (NAT) Gateway:** A mechanism used to allow resources in the private subnet to access the internet while maintaining a secure barrier from inbound connections.
5. **Security Groups and Network ACLs:** Security measures that control inbound and outbound traffic at the instance and subnet levels, bolstering the overall security posture.
6. **Route Tables:** Configuration tables that determine the routing of network traffic within the VPC.
7. **Auto Scaling Group (ASG):** Automatically adjusts the number of EC2 instances to meet demand, ensuring high availability and scalability for applications.
8. **Launch Configuration:** Defines the blueprint for EC2 instances in an Auto Scaling group, specifying the AMI, instance type, and other configuration details.
9. **Target Group:**A logical group of instances or resources that receive traffic from a load balancer, allowing for fine-grained routing and load balancing decisions based on health and routing rules.
10. **Load Balancer:**Distributes incoming network traffic across multiple servers or resources to ensure high availability, improve application performance, and prevent overloading of individual servers.
11. **Application Load Balancer (ALB):** Distributes incoming application traffic across multiple targets, such as EC2 instances, based on defined rules and supports advanced routing features.
12. **Amazon EC2 (Elastic Compute Cloud):** Provides resizable compute capacity in the cloud, allowing you to run virtual servers (EC2 instances) for various applications, services, and workloads.

**Step1: Create a VPC**

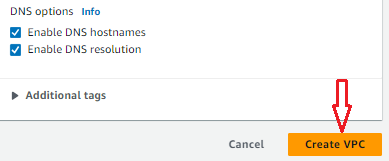
* Navigate to the VPC pannel in AWS console, and click on create VPC with option as VPC and More.



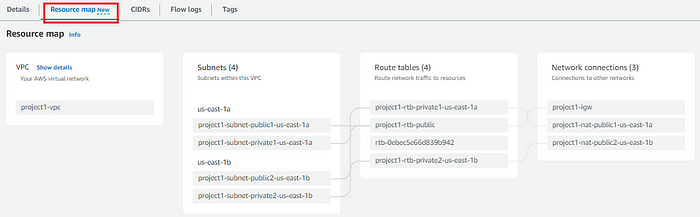
* Choose a VPC CIDR to create or we can leave the default which console displays.
* Since we need to create 2 public subnets, 2 private subnets lets choose Number of Availability Zones (AZs) as 2, Number of public subnets as 2, Number of private subnets as 2, NAT gateways in per AZ, & VPC endpoints as none.



* Enable options Enable DNS hostnames & Enable DNS resolution and then click on Create VPC.



* Now this will create a series of resources VPC and its components.
* Once the creation is complete click on the created VPC choose the resource map where we can see the VPC resource flow.



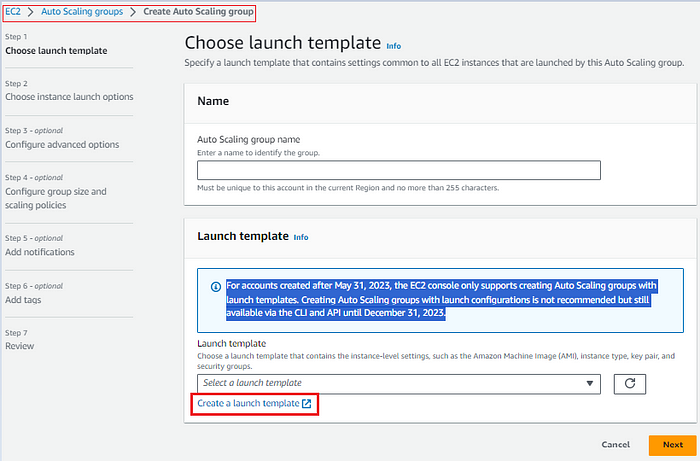
With this we have completed creating VPC

**Step2: Creating a Auto Scaling Group.**

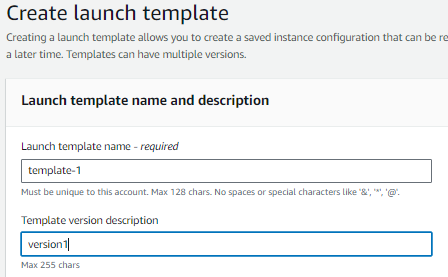
Creation of a Auto Scaling Group as 2 parts, one is creating a launch template and second is to create a Auto Scaling Group by itself.

**Action 1: Creating a Launch template.**

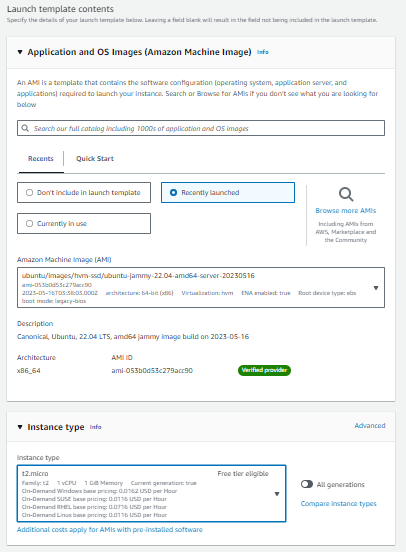
* Navigate to EC2 in left hand side scroll until Auto Scaling Group click on create.
* provide your Auto Scaling Group a name, click on create template where it will redirect you to a new page to create the launch template.



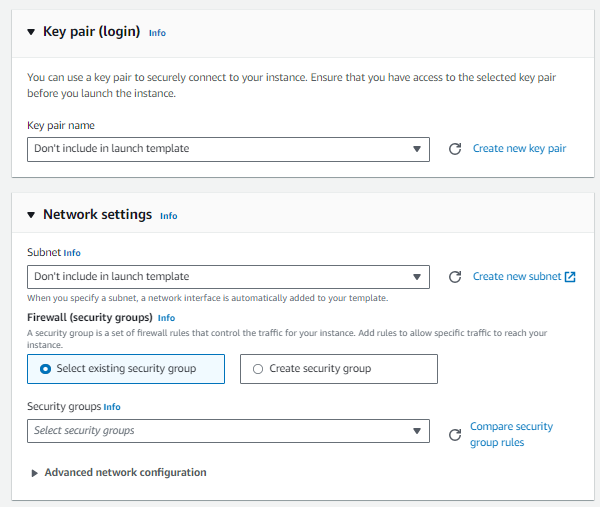
* provide your template with a name followed by description,



* Furthermore choose a AMI for the template to use to launch instances, here we choose Ubuntu, next we will select the instance type here we are selecting t2.micro for this project since it comes with free tier.



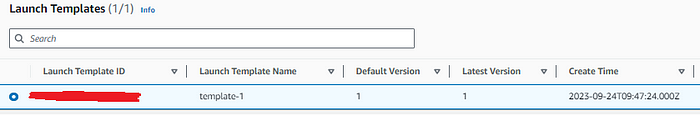
* then in next step choose the key pair for instances to connect, lets also choose Don’t include in launch template for subnets, will choose a security group for our launch template



* Rest all we will keep default as same and click on the create template version which will create our launch template.

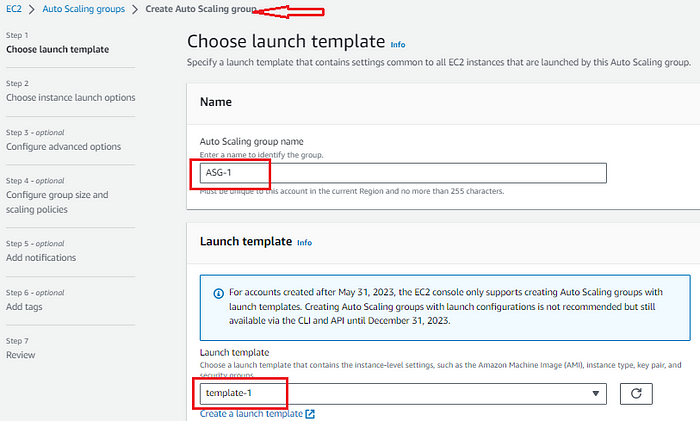
https://miro.medium.com/v2/resize:fit:263/1*IfoRfvCmSaivJSYnGpUR6A.png

* Once you create the template, the template what we have created will be displayed in the launch template console.

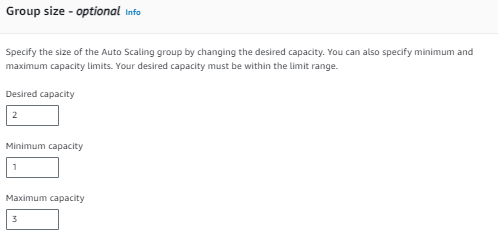


**Action 2: Create a Auto Scaling Group**

* Next we are creating the Auto Scaling Group, to do so click on the auto scaling group and click on create choose a name of your choice, under launch template choose the launch template you have created earlier.



* Choose the VPC along with the Availability zones and subnets which we have created earlier, make user to choose the subnets that are private as we are keeping our app in private subnets, choose next. We are leaving **Configure advanced options*— optional***as it is by keeping all fields as defaults, click on next it is always a best practice choose the group size according to requirement, for this activity we will choose the desired capacity, minimum capacity and maximum capacity as 1.



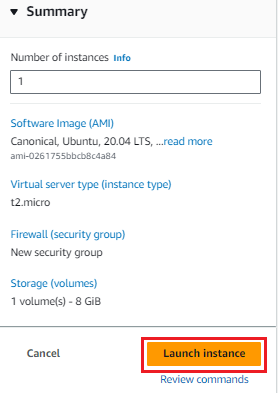
* leaving rest all default we will click on next, next add the tags is required, pass onto the next section 7, review all the options we have chosen and click on Create Auto Scaling Group. This will create us the Auto Scaling Group.

https://miro.medium.com/v2/resize:fit:549/1*_rdZFkmAdiLACGFlFK3OWw.png

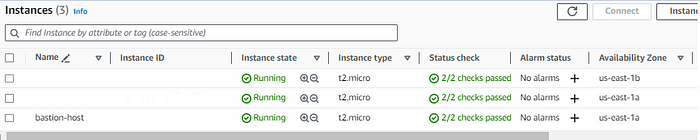
* Since we are creating our applications servers on private servers we will not be able to connect this servers directly in order to connect them we need a bastion server/jump host server.

**Step 3: Create a Bastion host**

* We will now create a bastion host/jump server, in order to create the navigate to the EC2 console and click on Launch Instances → choose a name as bastion host/jump server, choose the OS as required(here we will be choosing ubuntu 20.04), will choose the instance type t2.micro, choose a key pair to login(we will choose the key pair which we have created earlier), choose the VPC which we have created earlier along with the public subnet to create the bastion host, will choose to create a security group and open port 22 inbound to connect SSH, we will leave the storage settings as default, verify the Summary in the right hand side of console and click on launch instances.



* Now let us navigate to the EC2 dash board, where we can see a bastion host is created and ASG as created the instances as per specifications mentioned in our launch configuration and ASG.



* Let us connect to the bastion host and then connect to one of the app server, here we will run the python application, in app server make a file as index.html and copy paste the below content.

If you stuck with the below error, give the command to login in the bastion host

@ WARNING: UNPROTECTED PRIVATE KEY FILE! @

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Permissions 0644 for 'app-login.pem' are too open.

It is required that your private key files are NOT accessible by others.

This private key will be ignored.

Load key "app-login.pem": bad permissions

ubuntu@192.168.158.105: Permission denied (publickey).

Chmod 600 app-login.pem

<!DOCTYPE html>  
<html>  
<body>  
<h1>My sever 1 one </h1>  
<p>My first Paragraph.</p>  
</body>  
</html>

* Save a the file, once it is saved you can read the content with command

cat index.html

* once we have saved the file run the below command to start the application, make sure you have opened the port 8000 to accept the connections.

python3 -m http.server 8000

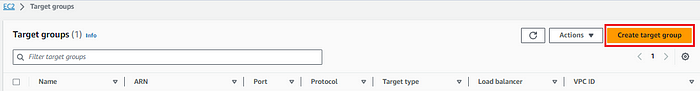
* Similarly login to the other app server and make the configurations as we have made to the first app server.

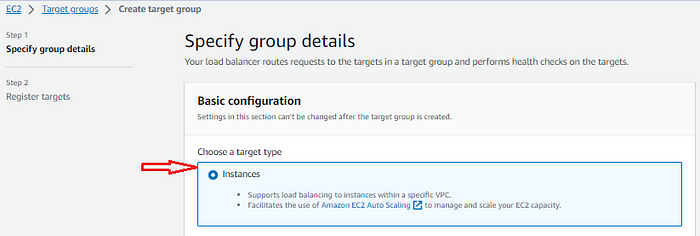
**Step 4: Create a Load Balancer**

* In order to create the a load balancer creation we need to create the target group

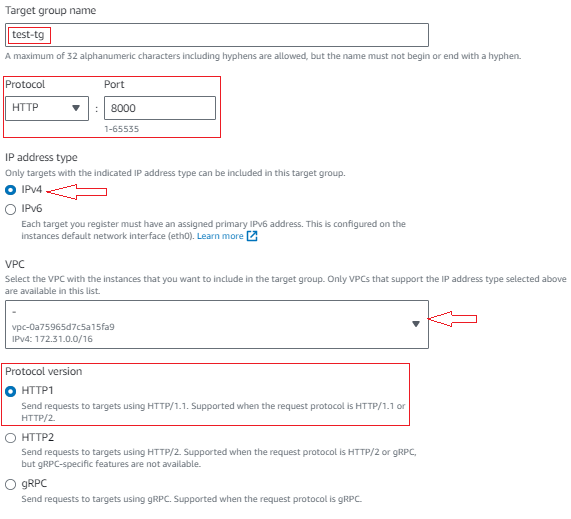
**Action 1: Create a Target group.**

* In the EC2 console navigate to the load balancer and choose target group and click on create target group, choose target type as instances.

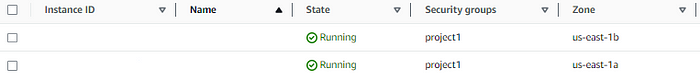




* enter a target group name as per requirement, choose protocol as HTTP & port 8000, IP address type IPV4, select VPC & protocol version as HTTP1 and click on next.

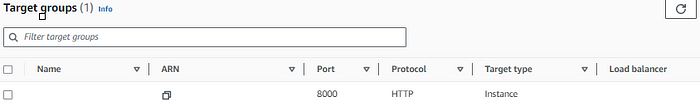


* In next step we will choose the target groups, choose the instances and click on as including as pending below & and click on Create target group.

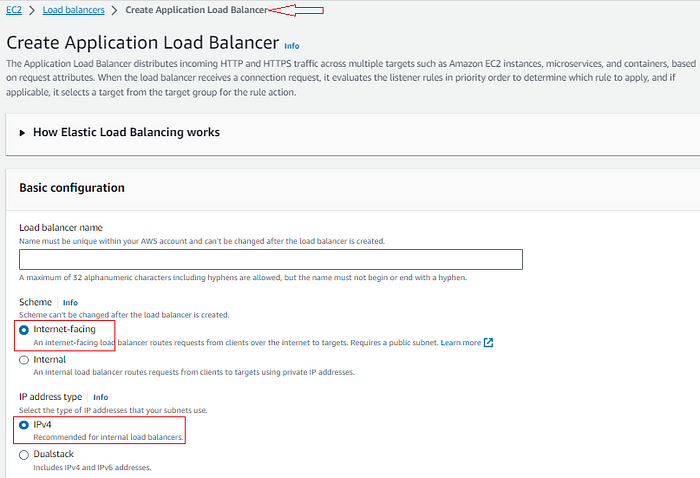


https://miro.medium.com/v2/resize:fit:295/1*Jo1unZU7ryQyW5yZ-n3a5w.png

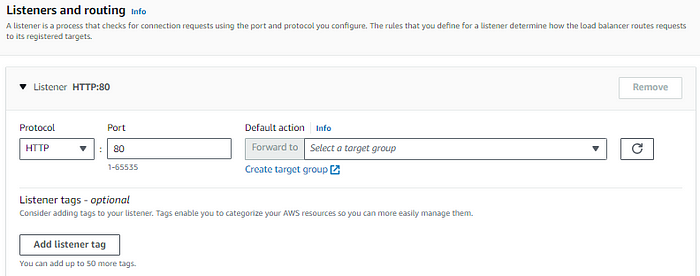
* Once you create a target group it will be displayed,



* Now let us create the ASG, click on create ASG choose the name, choose as the internet facing, choose the IP address type as IPV4.



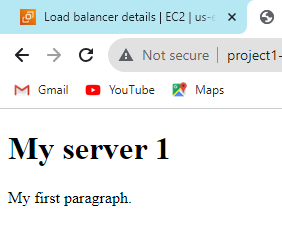
* Next will choose the VPC and the SG for ASG to create. Scrolling down at choose the Listeners and Routing, we will choose protocol as HTTP and port as 80 and lets choose the target group from the dropdown and click on Create load balancer.



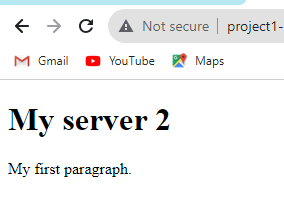
https://miro.medium.com/v2/resize:fit:211/1*nFJDr8sdwhRcBbuUhBFg_A.png

* Once the load balancer is created it is been displayed under load balancer section. Now let us access the application we have created using the load balancer DNS to do so navigate to the load balancer, copy the DNS and paste the same URL in the browser of your choice.
* Navigate to the load balancer that you have created and copy the DNS from the load balancer dashboard and paste the same in the browser.

https://miro.medium.com/v2/resize:fit:197/1*A4pB0L28hKul4nChoGw6Og.png



* start refreshing your browser you can see that the request is passing to alternative servers each time you refresh.



* Also you can stop application in any one server and verify the traffic flow at this moment at each and every time we refresh the browser we will gate a page of bad gateway and browser accessing successfully.

**With this we have successfully completed the activity.**

⚡ Get Started:

Clone this repo and follow the instructions to set up your own scalable AWS infrastructure. If you have any questions or suggestions, feel free to open an issue or reach out!

