=====================

**Getting Inside the virtual machine with EC2 and VPC(Virtual private cloud)**

VPC create a virtual network have private ip , free service can have max of 5 vpc in a acc.

**Key structures**

**Security group -** defines allowed incoming and outgoing Ip add and ports. mini firewall

**VPC uses routing table -** each vpc have 1 routing table

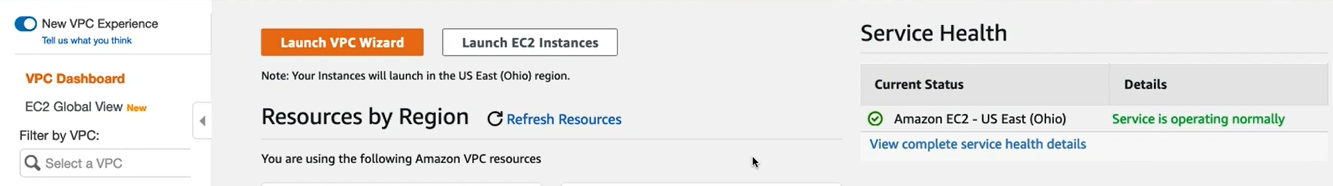
each vpc has 1 Network access control list - acts a ip filtering table and superpowered security groups that apply rules to the entire vpc

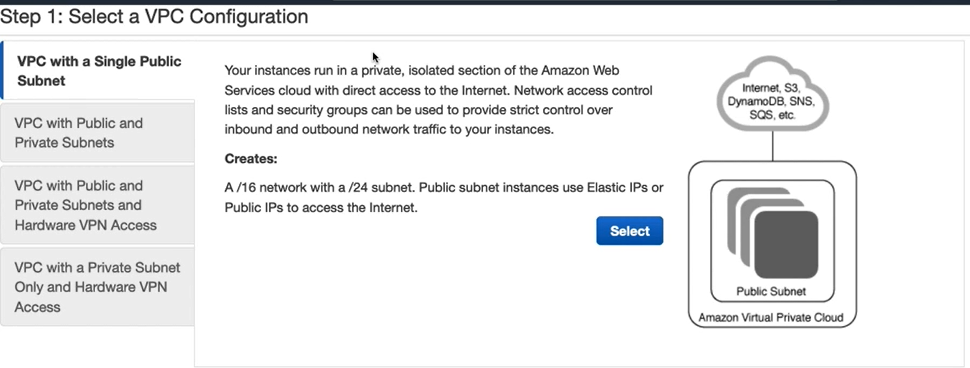
============================

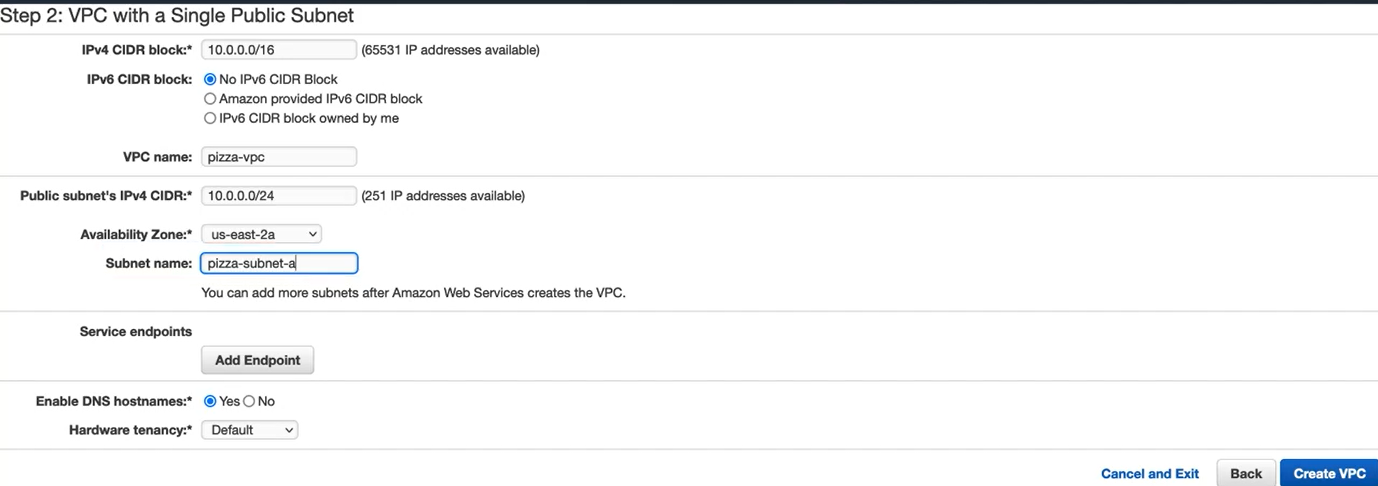
**Create a VPC**

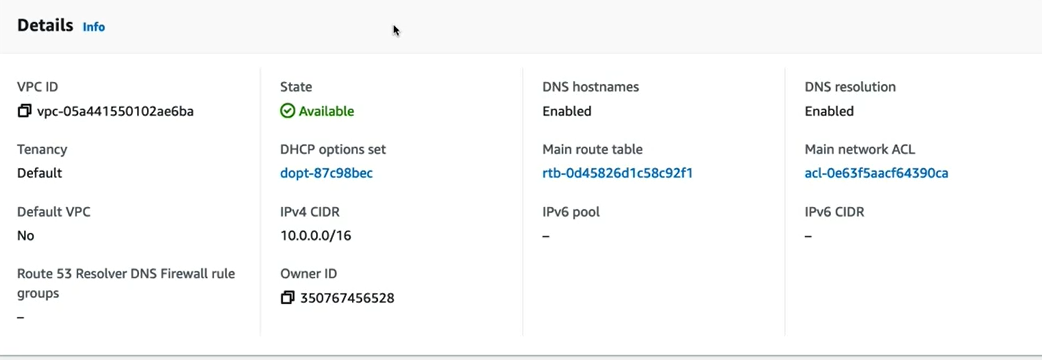
launch VPC wwizard

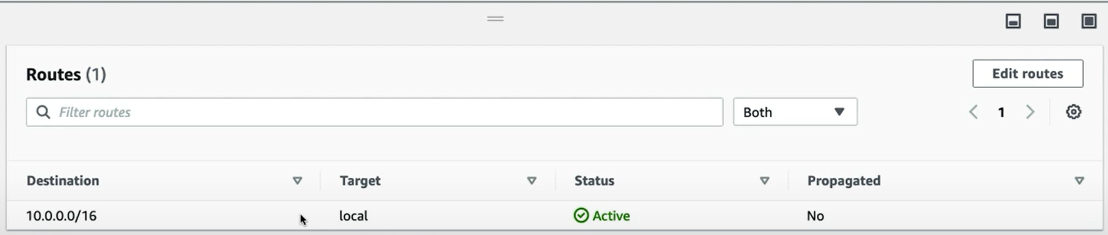
edit routing tables.

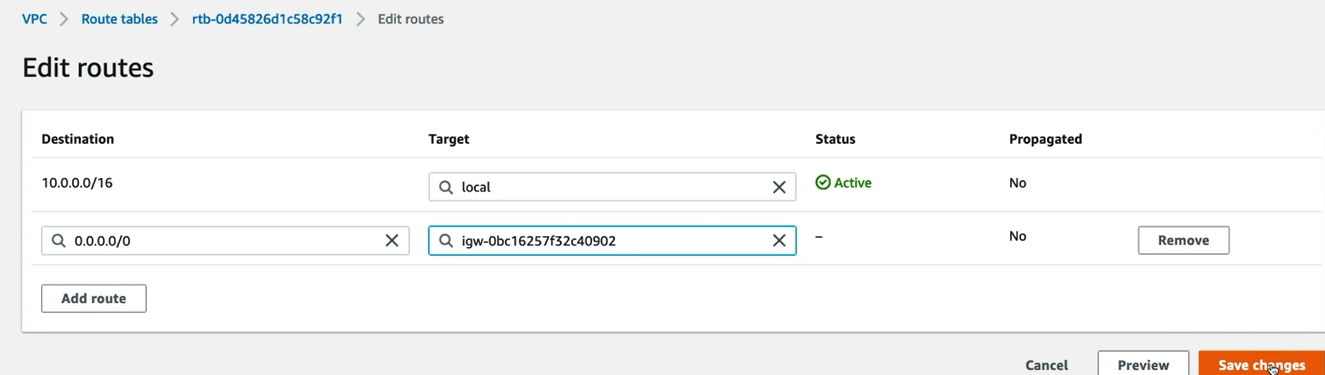










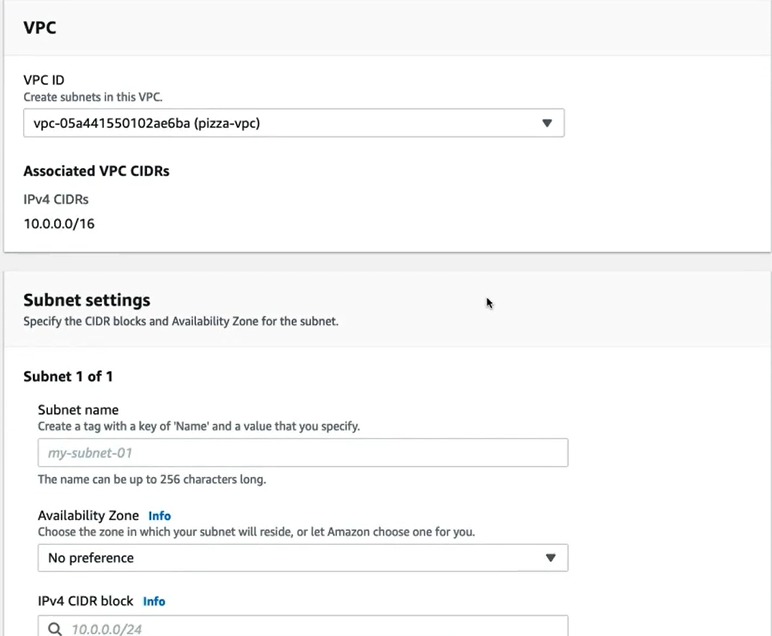


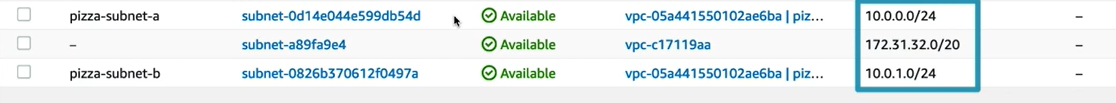
=========================

**Create a public subnet for scaling**

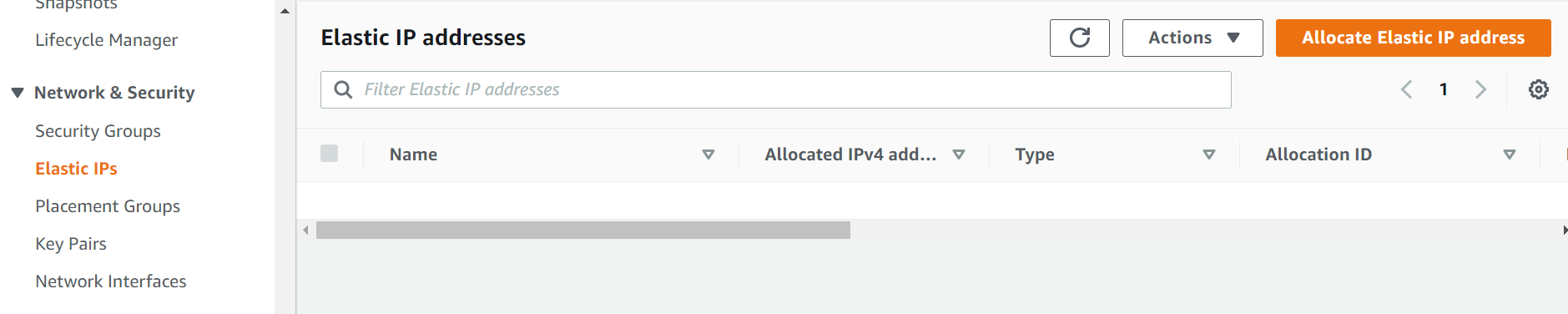
ssubnet can only exist in a single availability zone need to create more subnets to deploy into different availability zones.

Give subnet a name

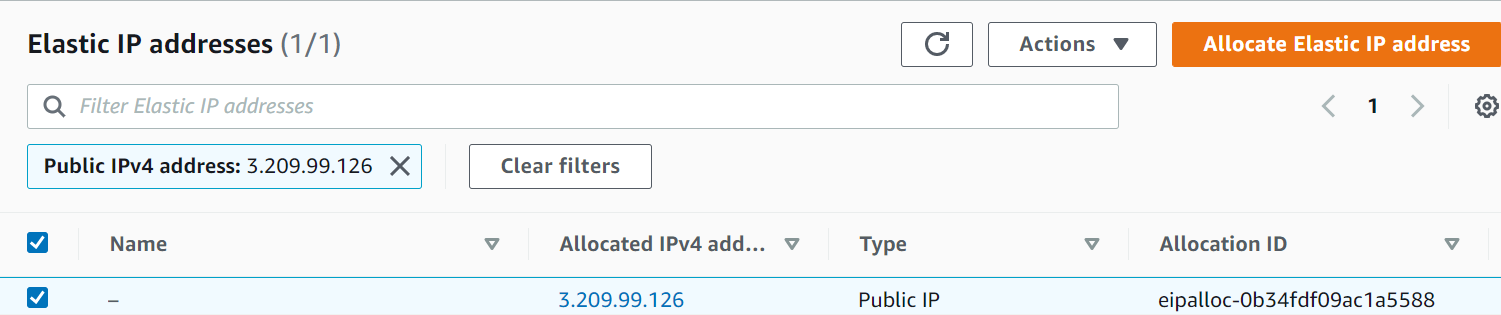


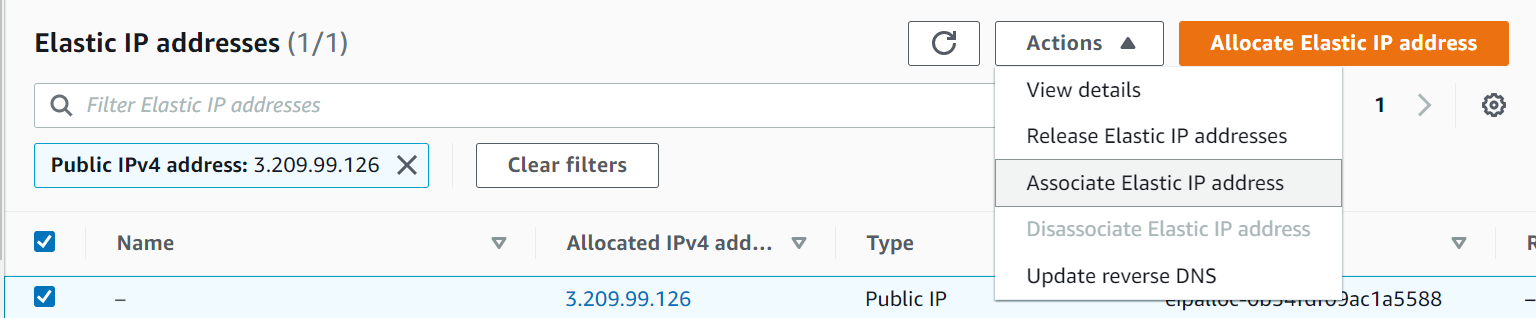


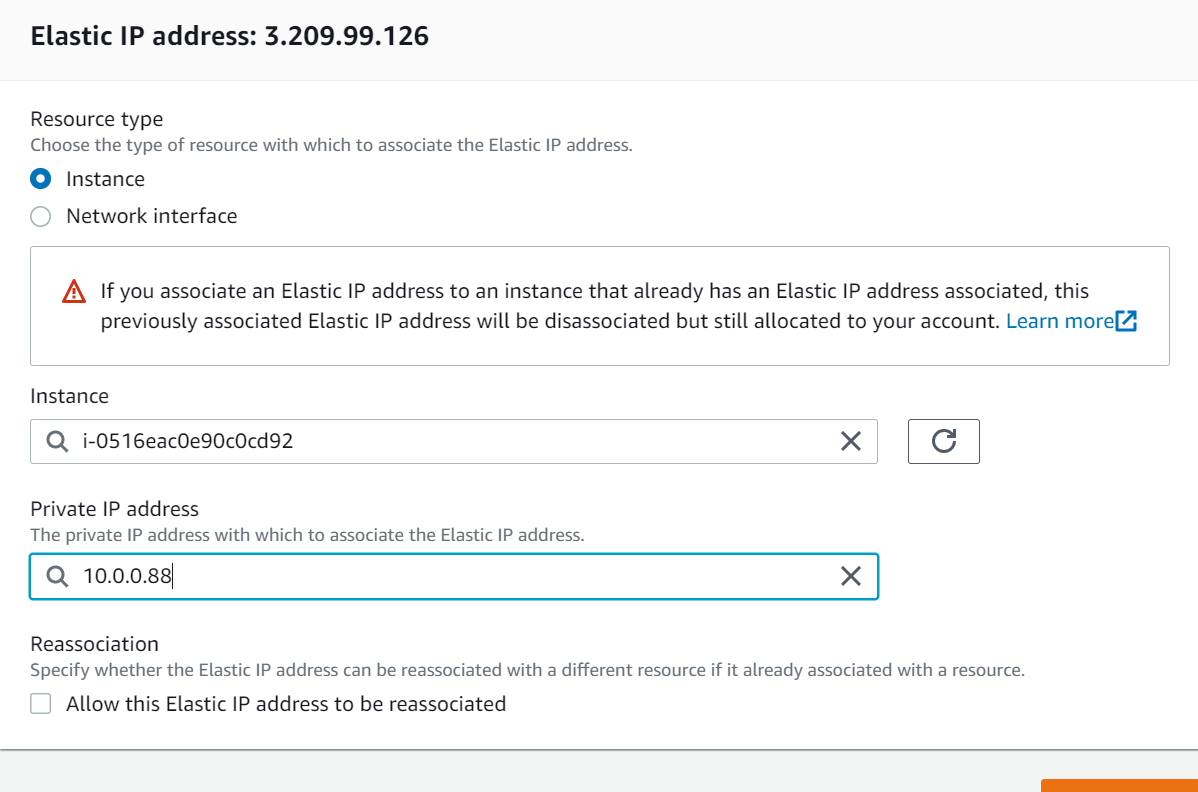
**Code for mig : https://github.com/ryanmurakami/pizza-luvrs**

**create and assign a public ip address**  
Elastic Ip on EC2 dashboard  
  
public ip address that are created, destroyed and assigned independently.  
  


Allocate Elastic Ip add and select allocate







**Updating and deploying to an EC2 Instance**

On the instance run the below

curl -sL https://rpm.nodesource.com/setup\_16.x | sudo bash -

**to install node js**

sudo yum install -y nodejs

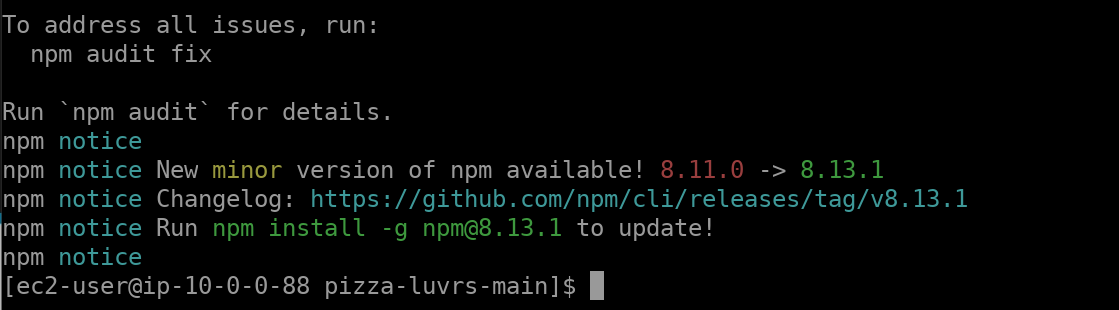
**Transfer demo app code to EC2 Instance**

**scp -r -i <pem\_file> <local\_code>** [**ec2-user@<ec2.ip>:home/ec2-user**](mailto:ec2-user@%3cec2.ip%3e:home/ec2-user)

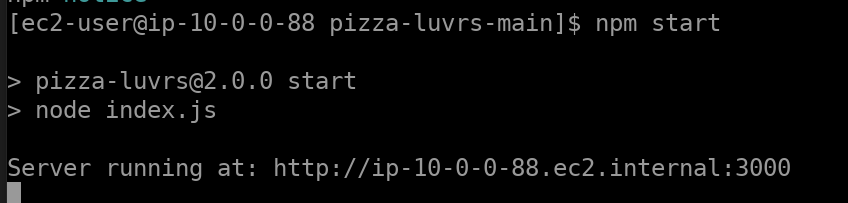
**Copy files either SCp or WInscp tool**

After copying files  
install node dependencies

🡪Npm install

****

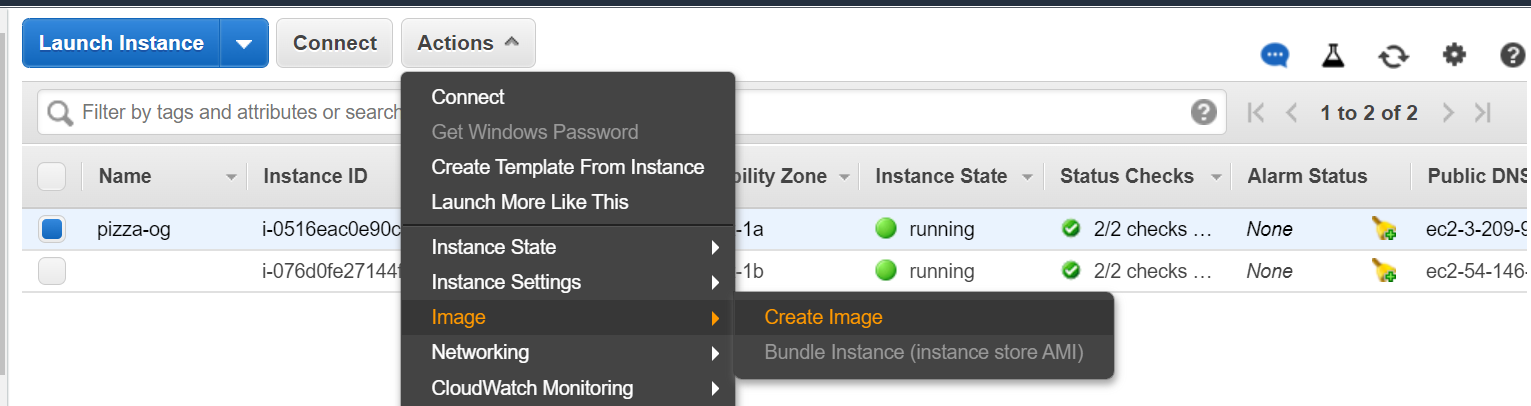
To start application 🡪NPM start

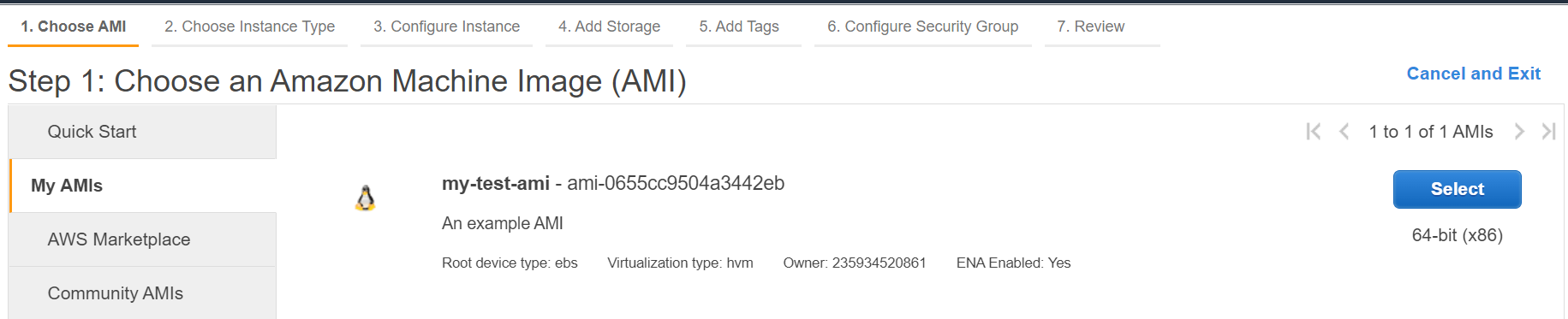


On chrome use elastic Ip to the application to load



**Create AMI**

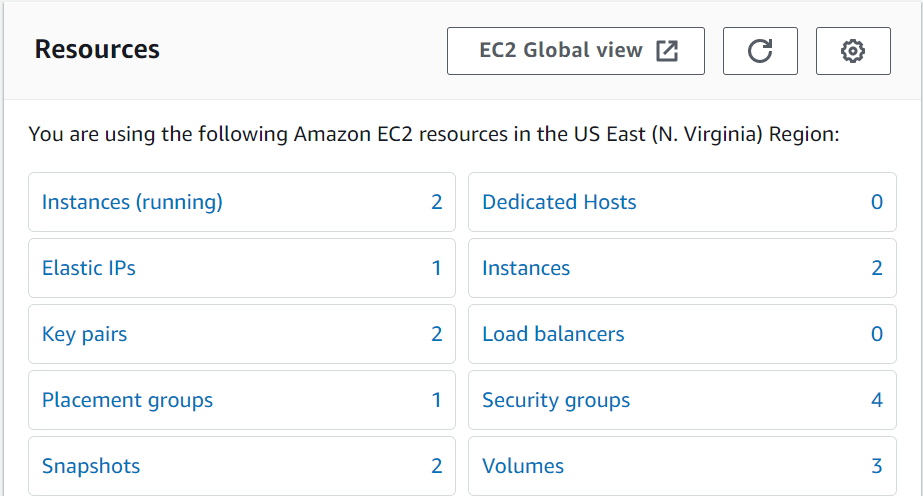
**Select instance and action** ****

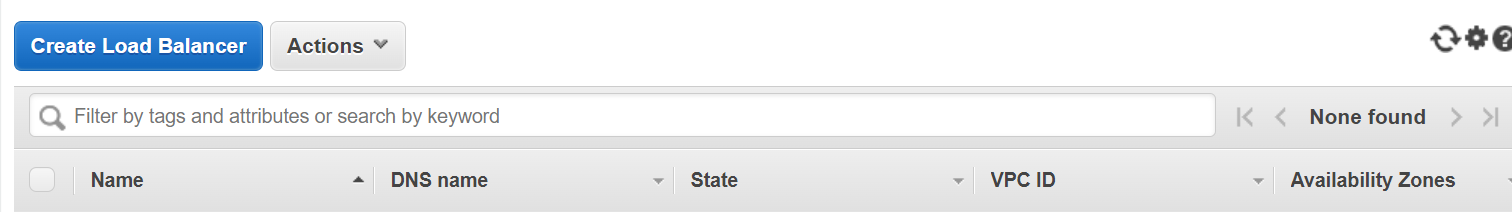
Launch instance  
****

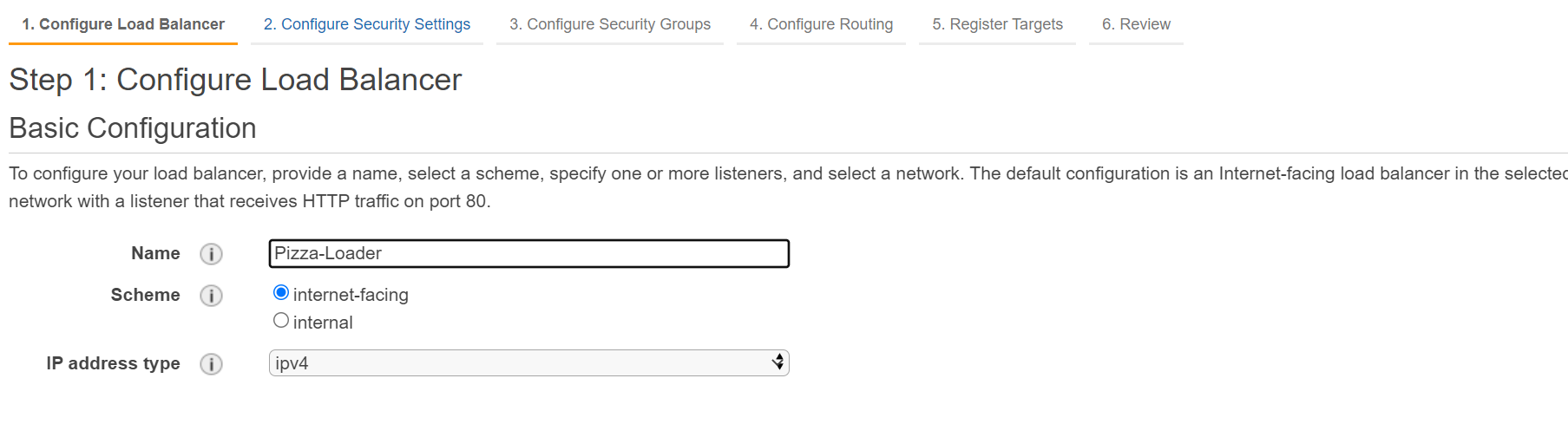
**Create LB**

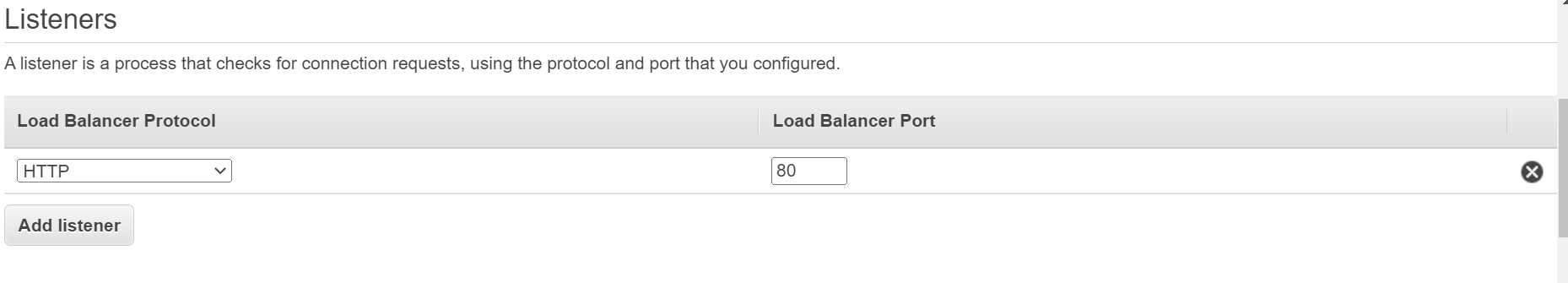
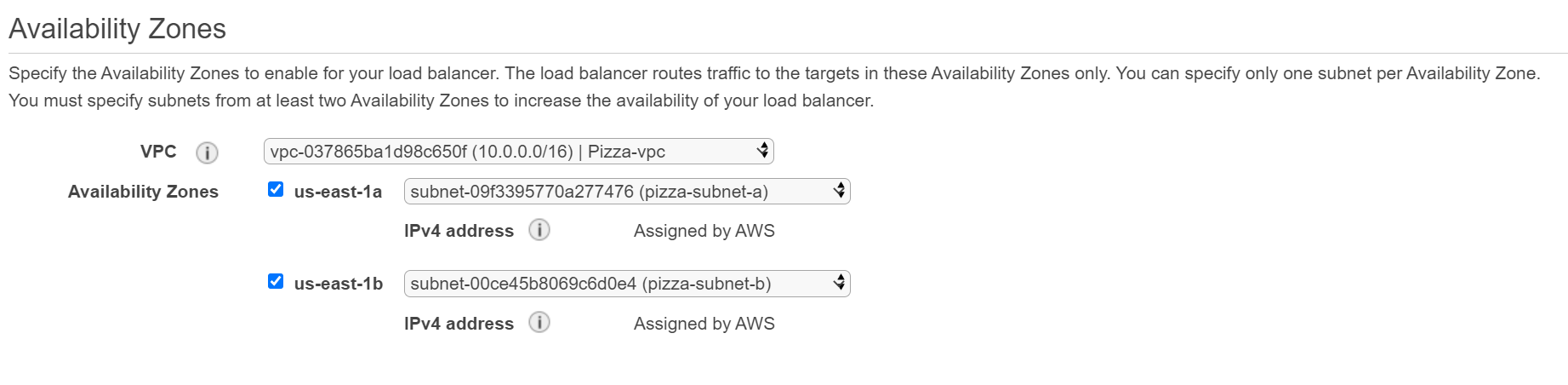
Click LB on resources section

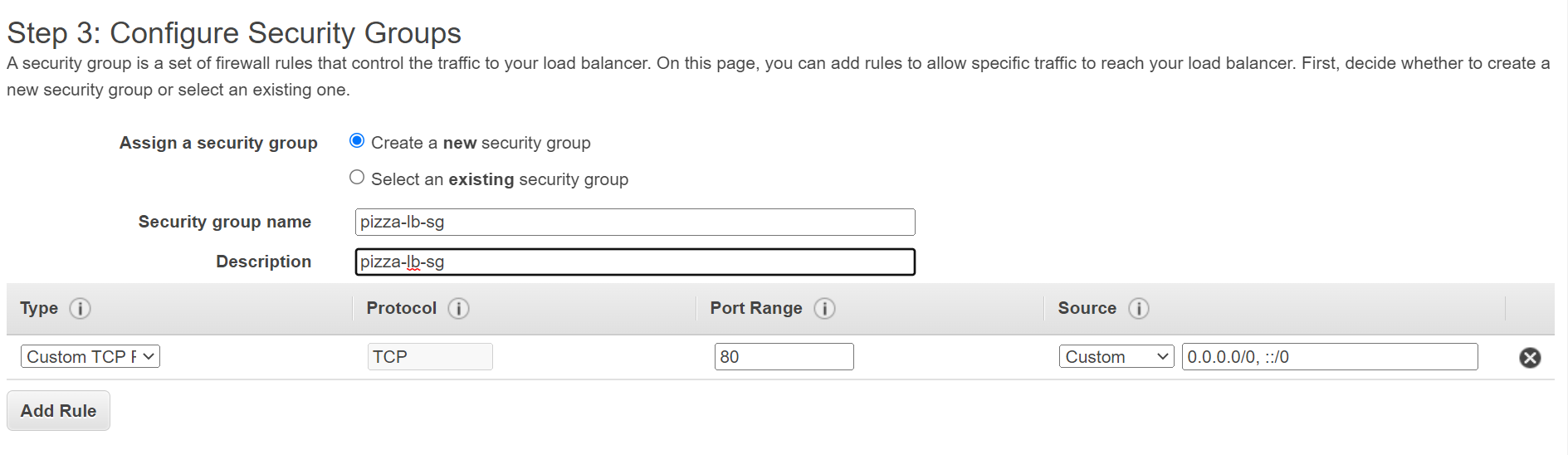
Select type : Https/Https, TCS/TLS/UDP, IP and classic

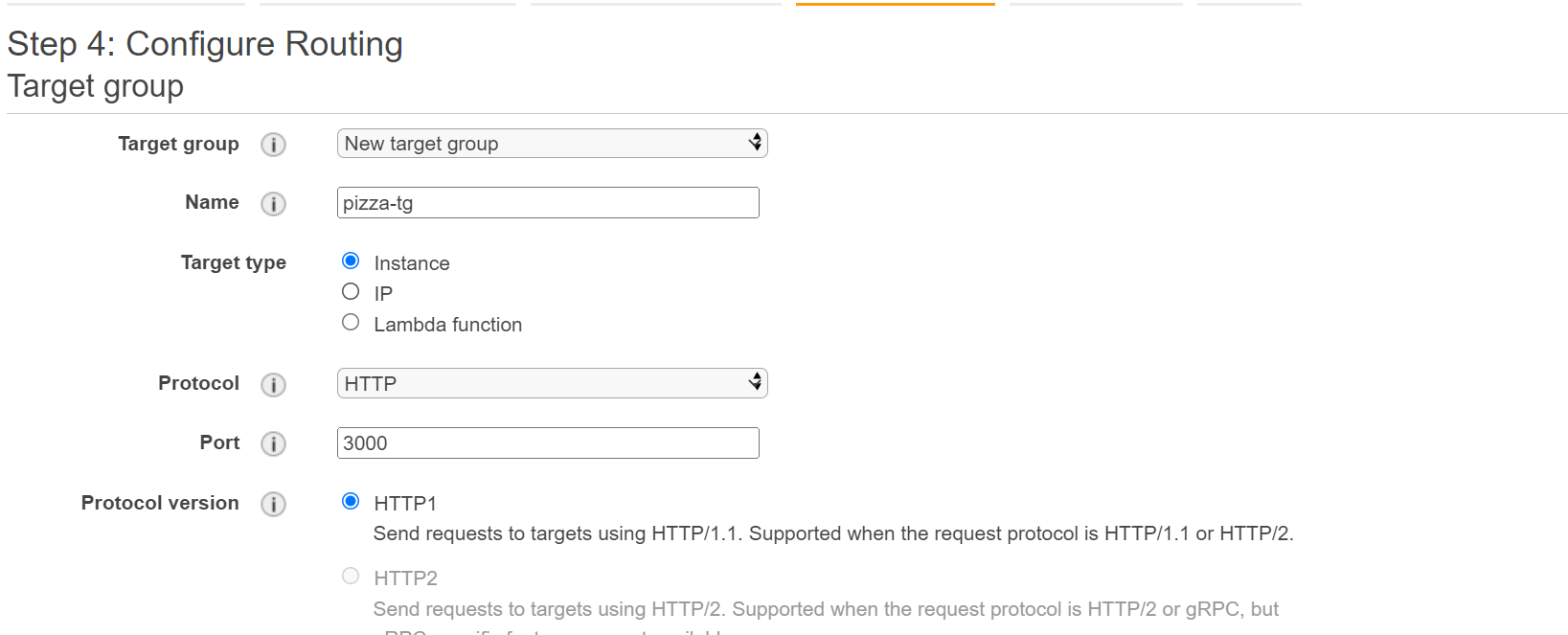


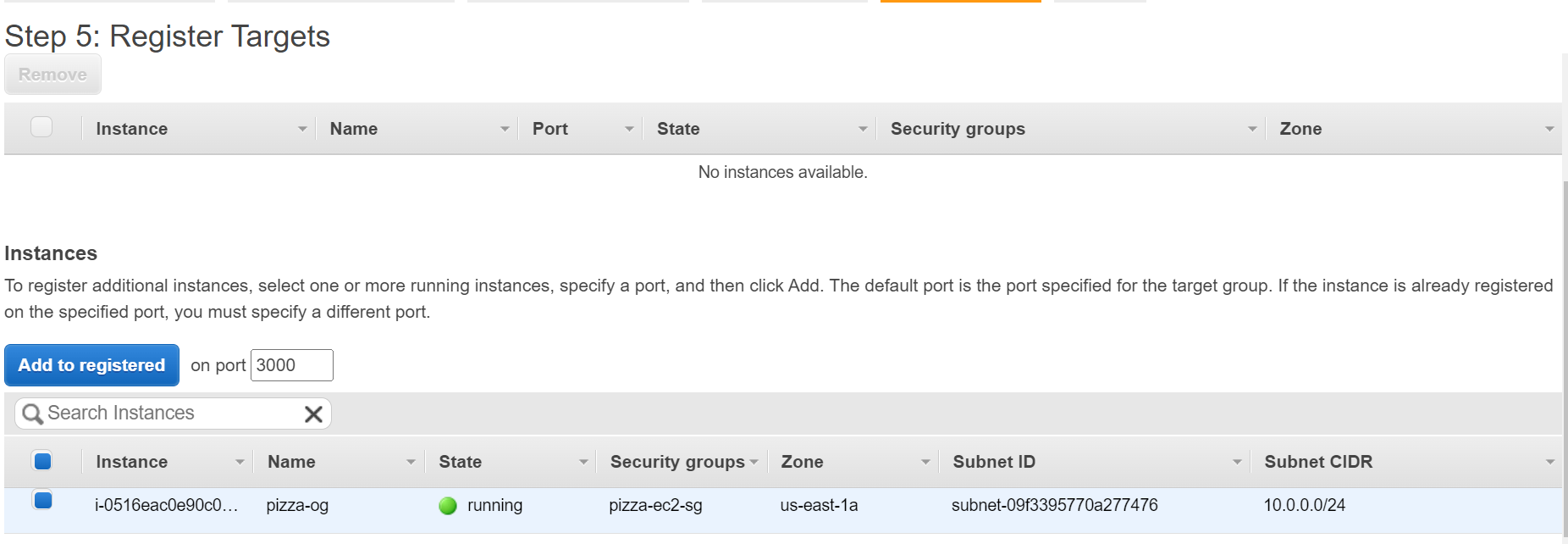




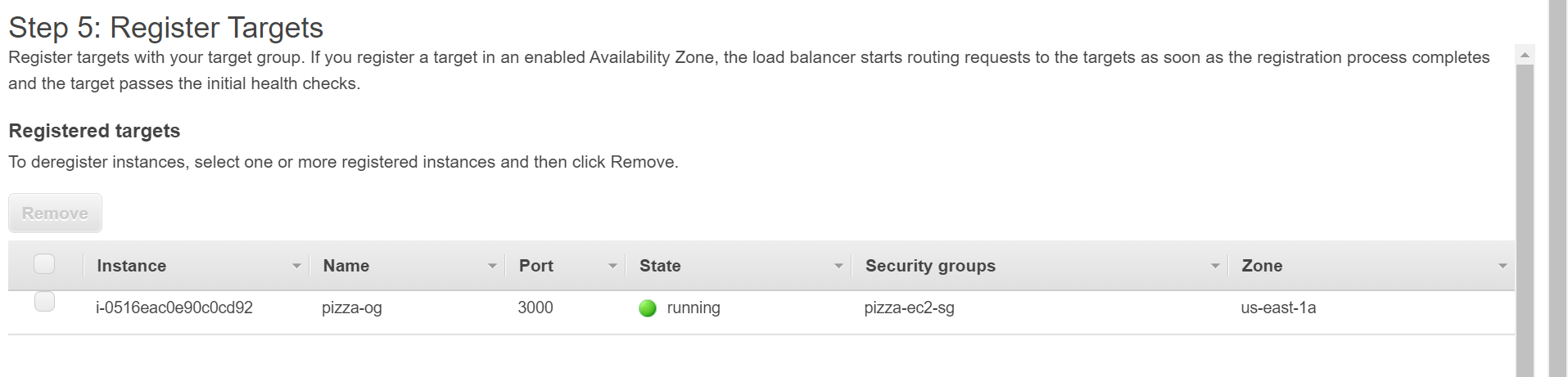
 





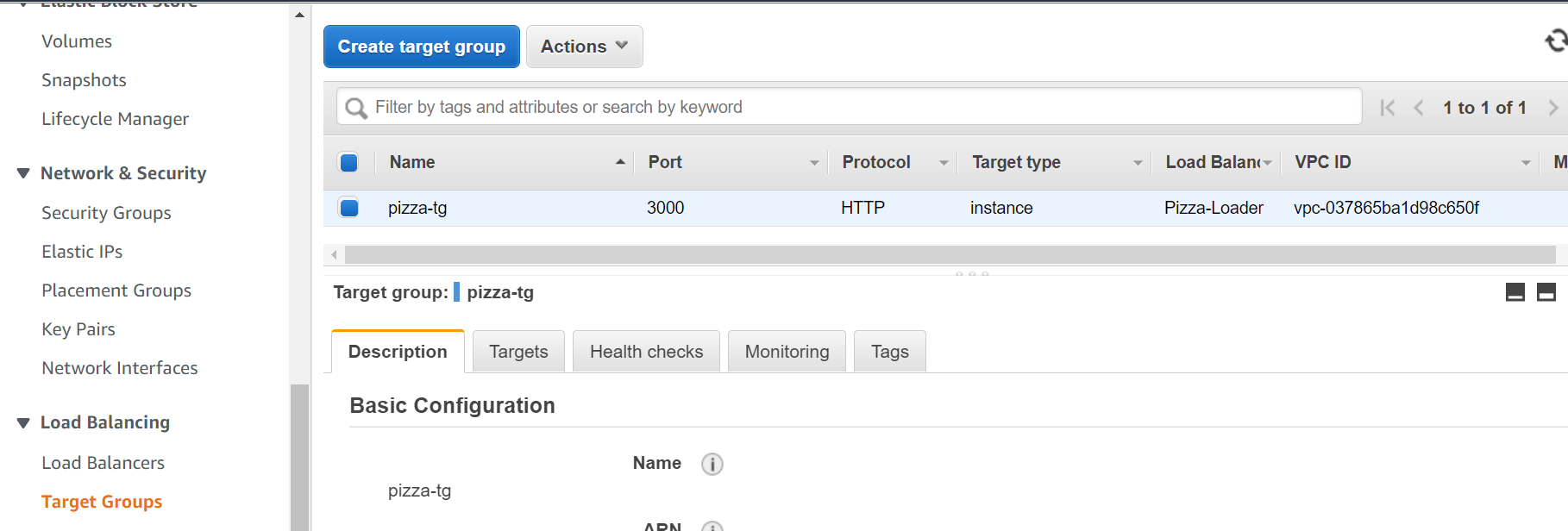


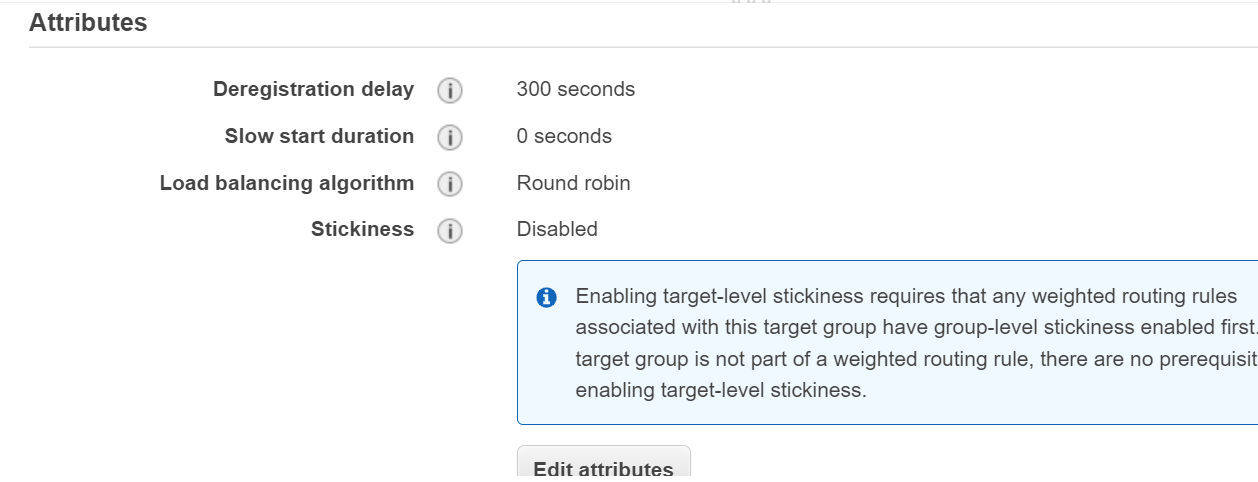
Select and add to register

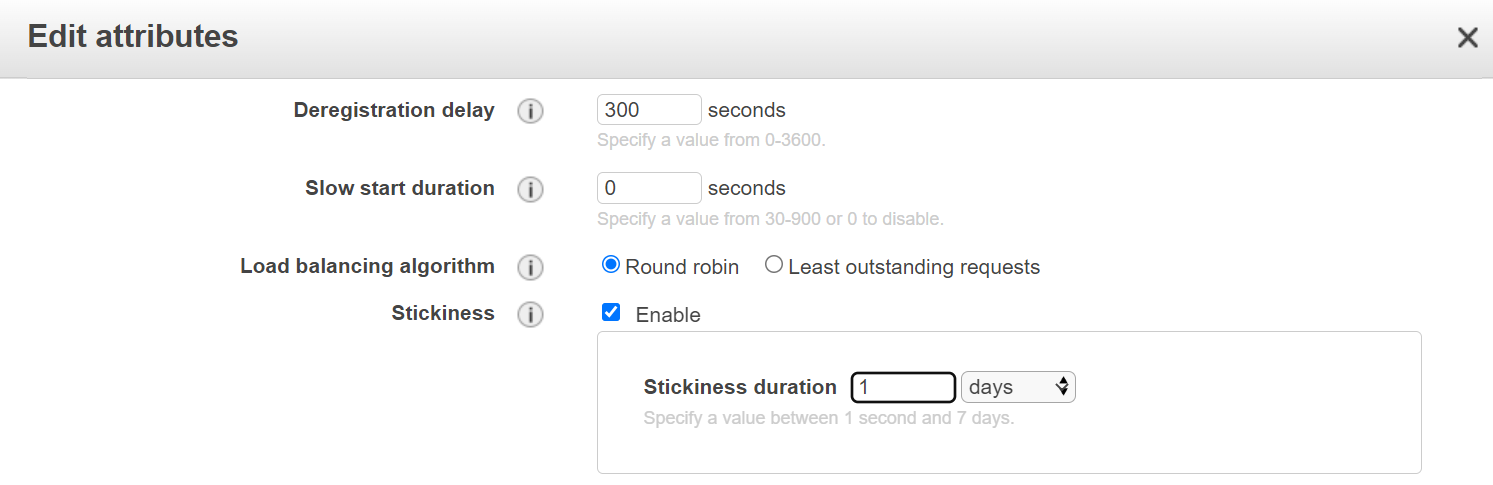


**Enable instance stickiness on load balancer**

* we need to ensure that iif a user logs into out app in on one EC2 instance that they continue to connect to that same instance with subsequent req.
* The load balancer will use a cookie to keep track of which users should be sent to which instances. Sessions stickiness in configures in the target group



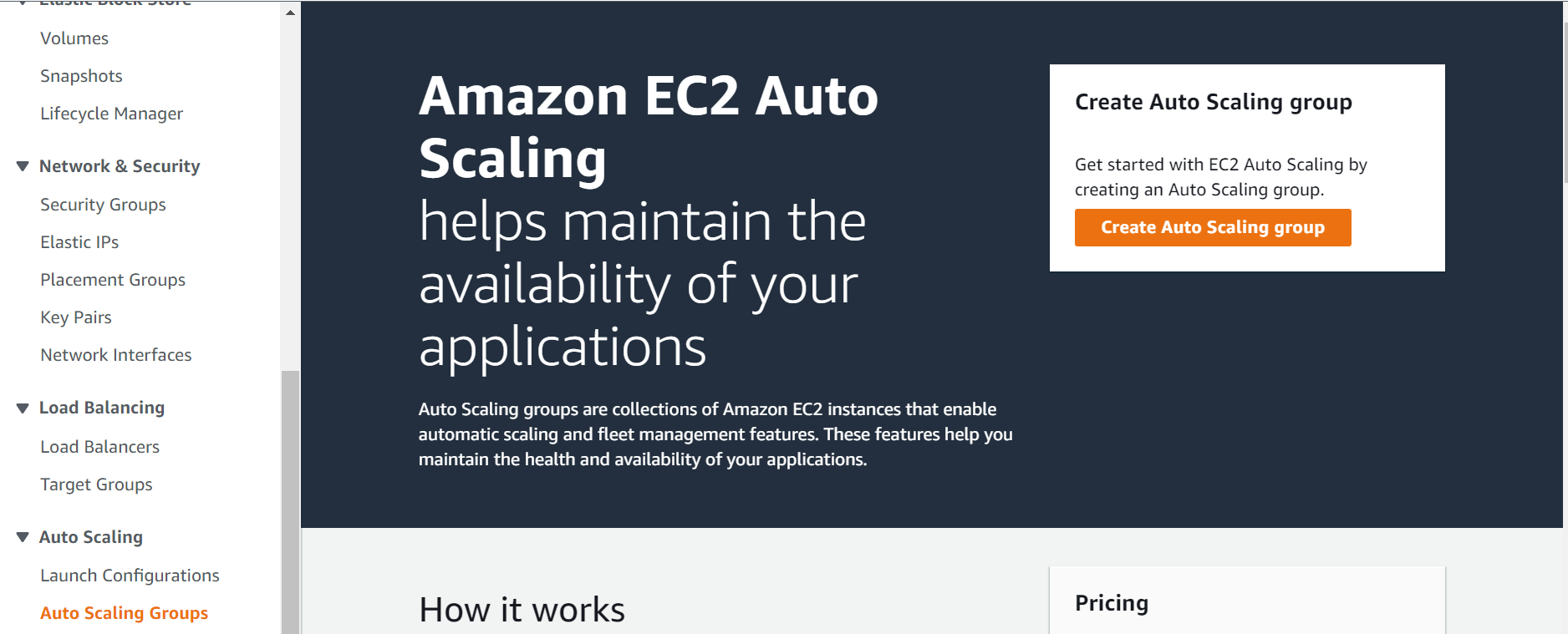


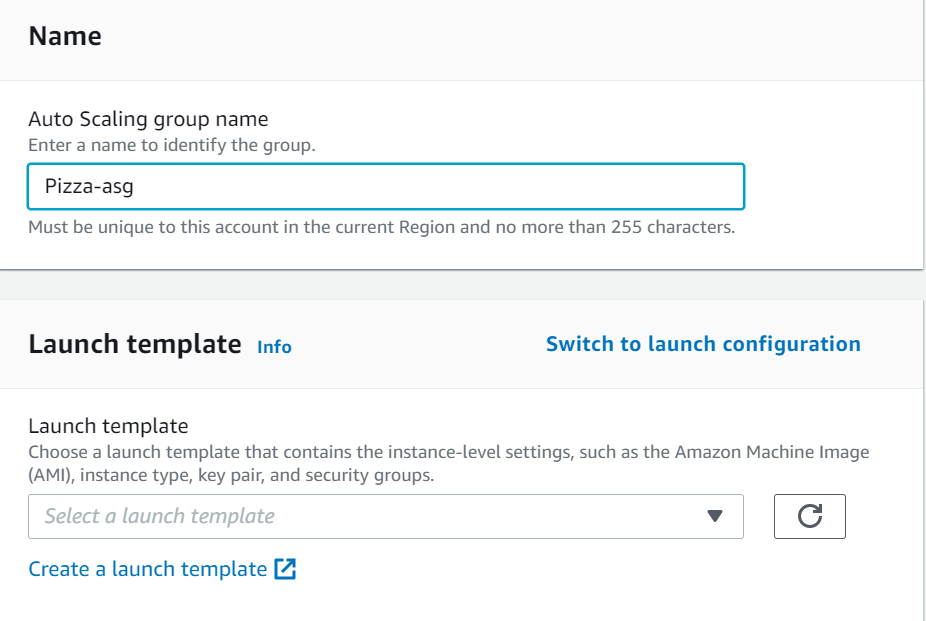


Session stickiness set to 1 day

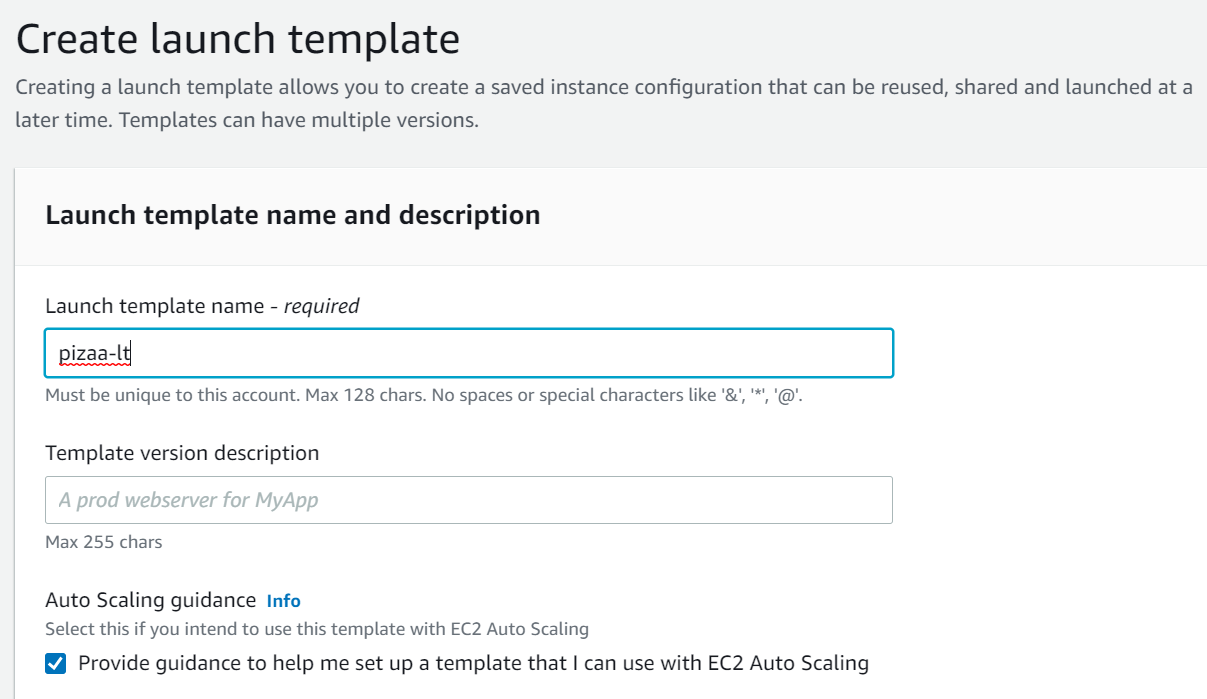
**Creating autoscaling group**

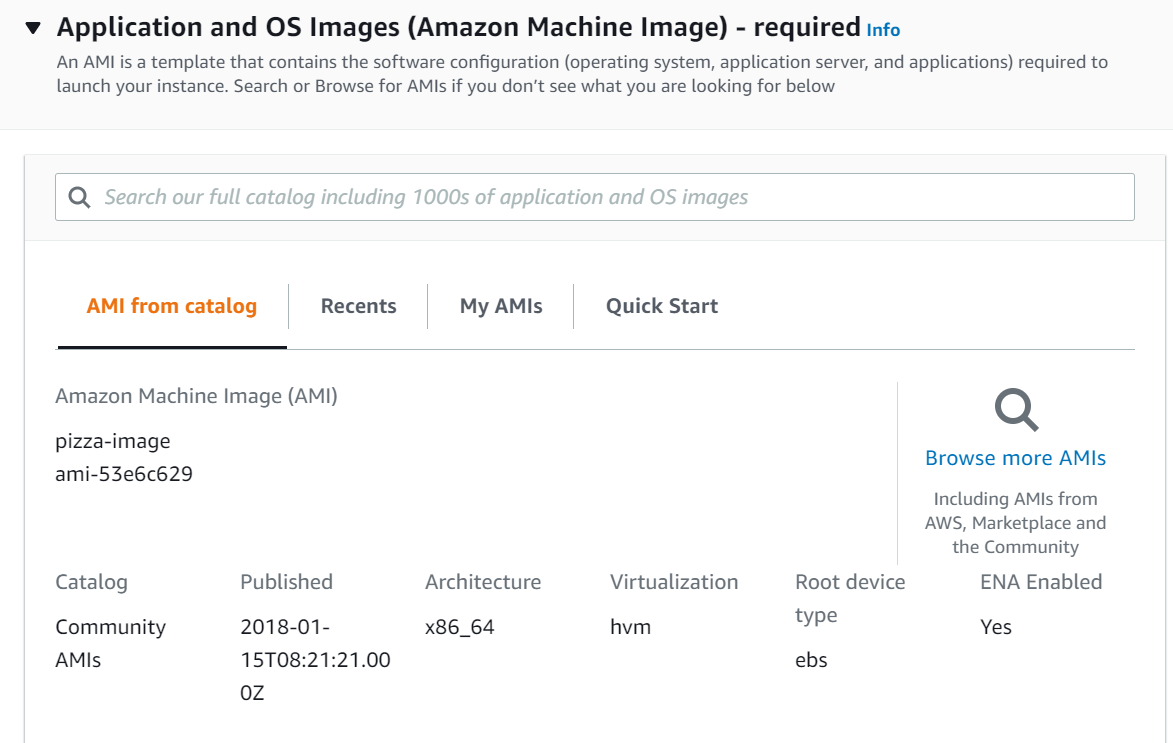
**ON EC2 - Auto scaling**

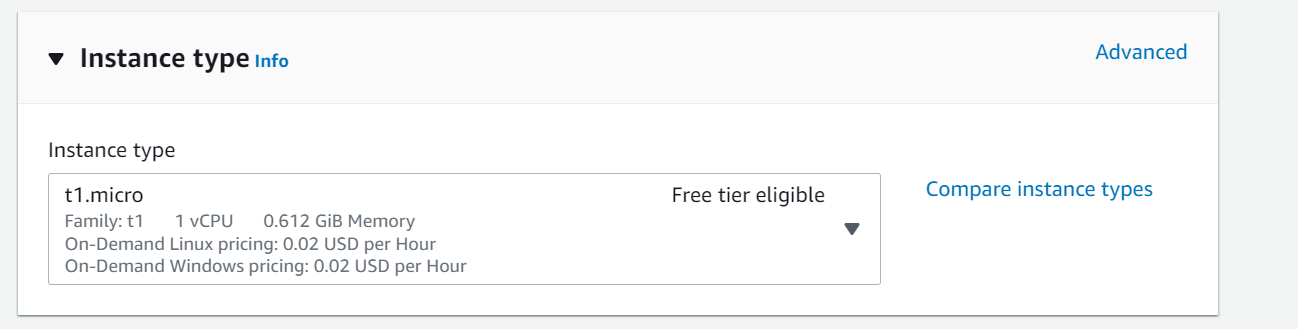
****

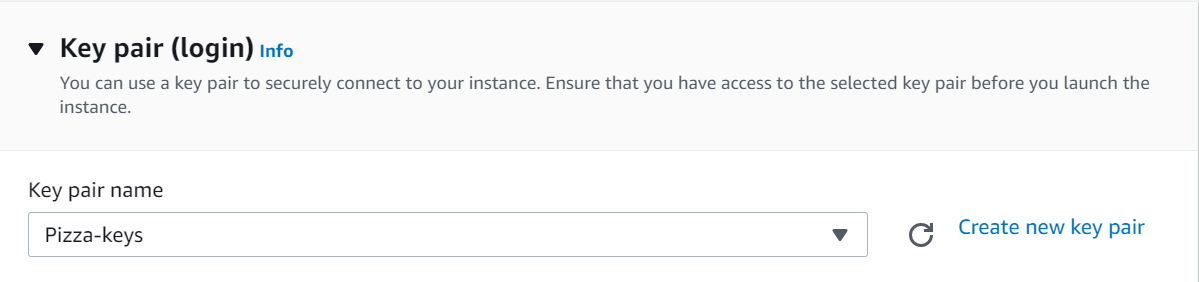
****

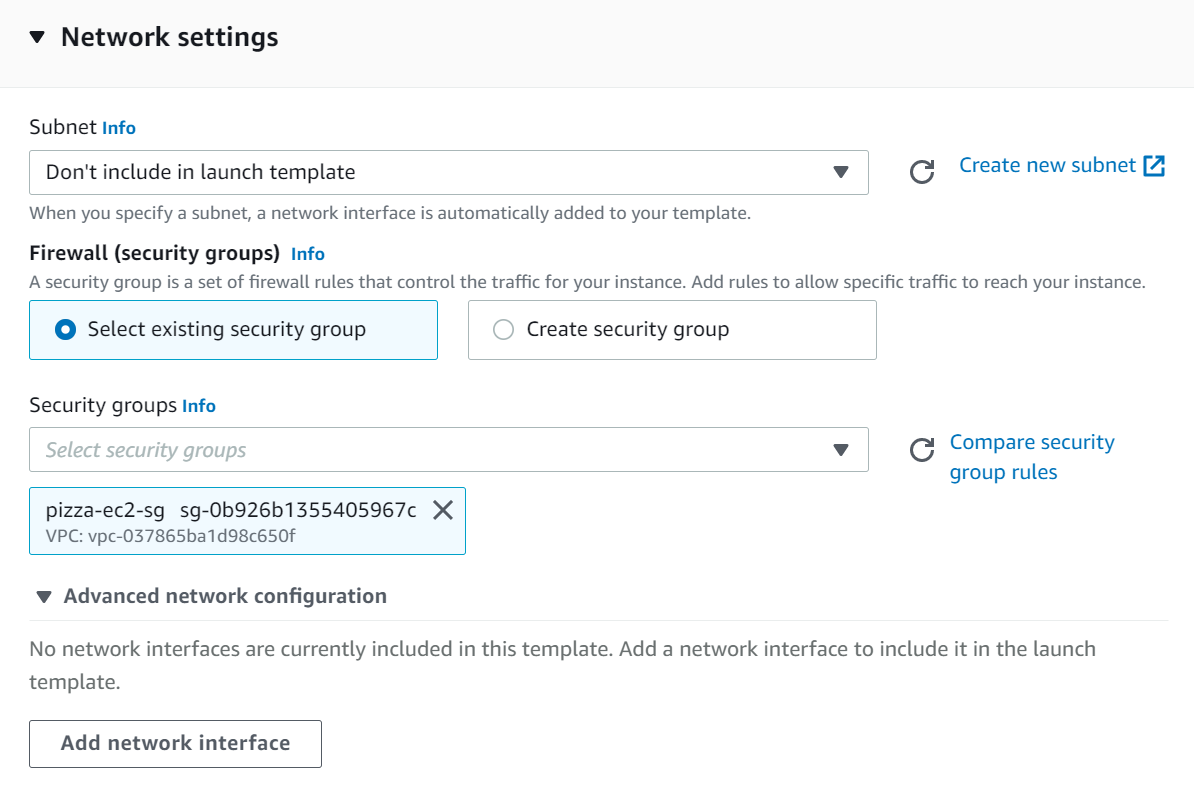
Since there is no existing template create a new template







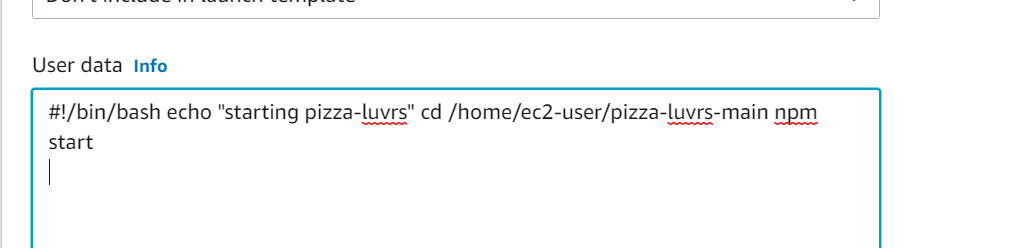


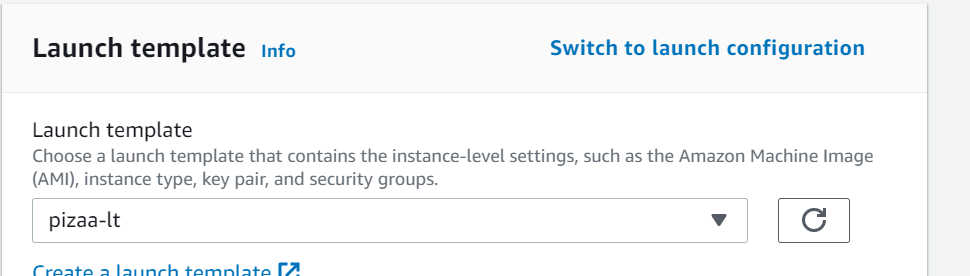


**On the advance**

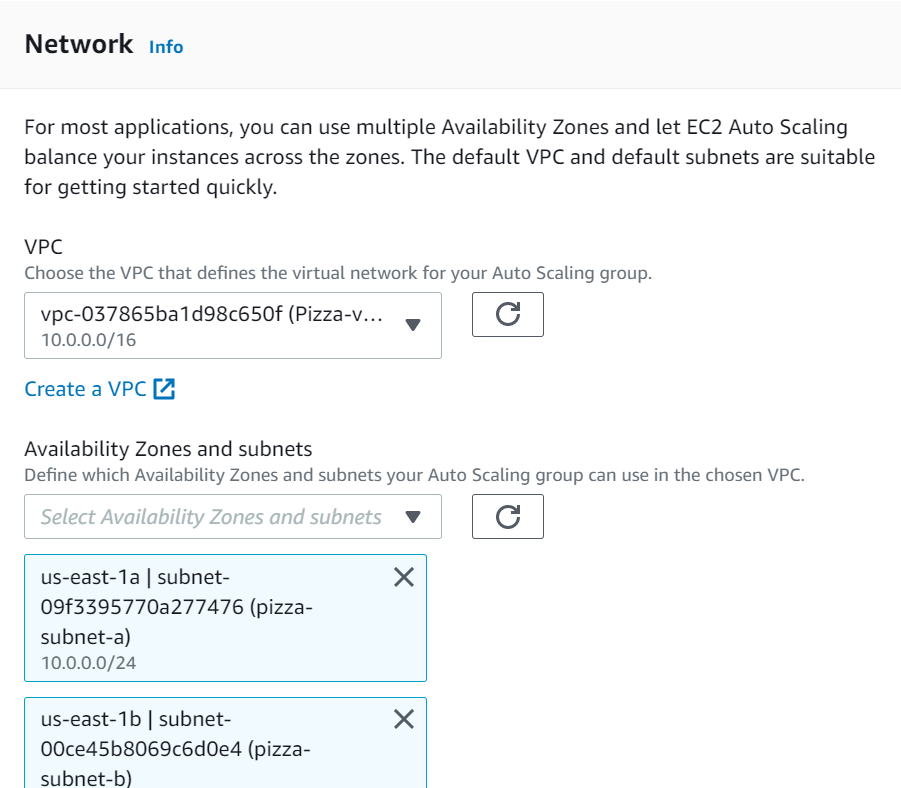
it's not enough for our instances to just be create. we also need to automatically start our node application. The user data section is where we can run scripts to execute at instance startup

**#!/bin/bash echo "starting pizza-luvrs" cd /home/ec2-user/pizza-luvrs-main npm start**

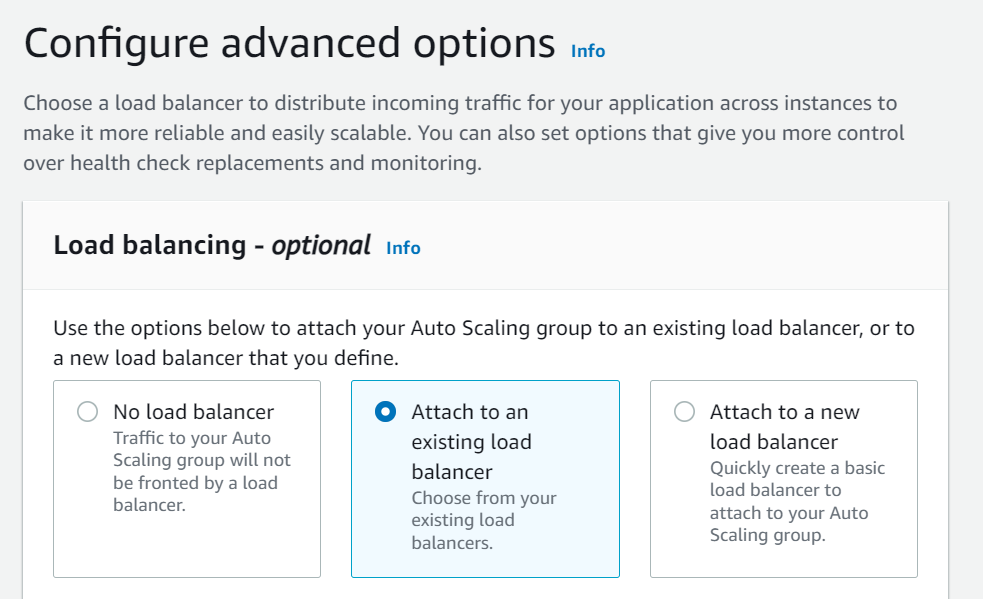
****

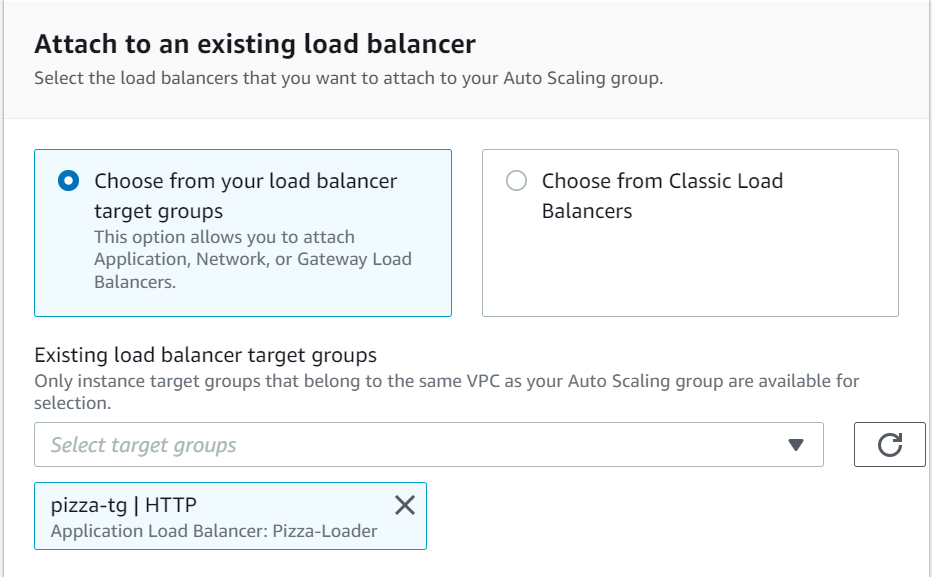
****

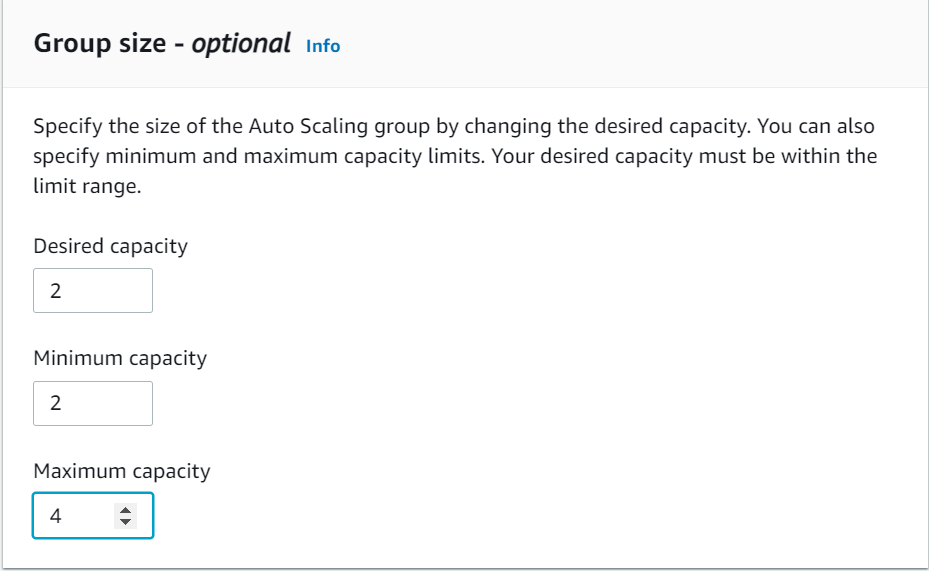
**On next select the VPC and both subnets**

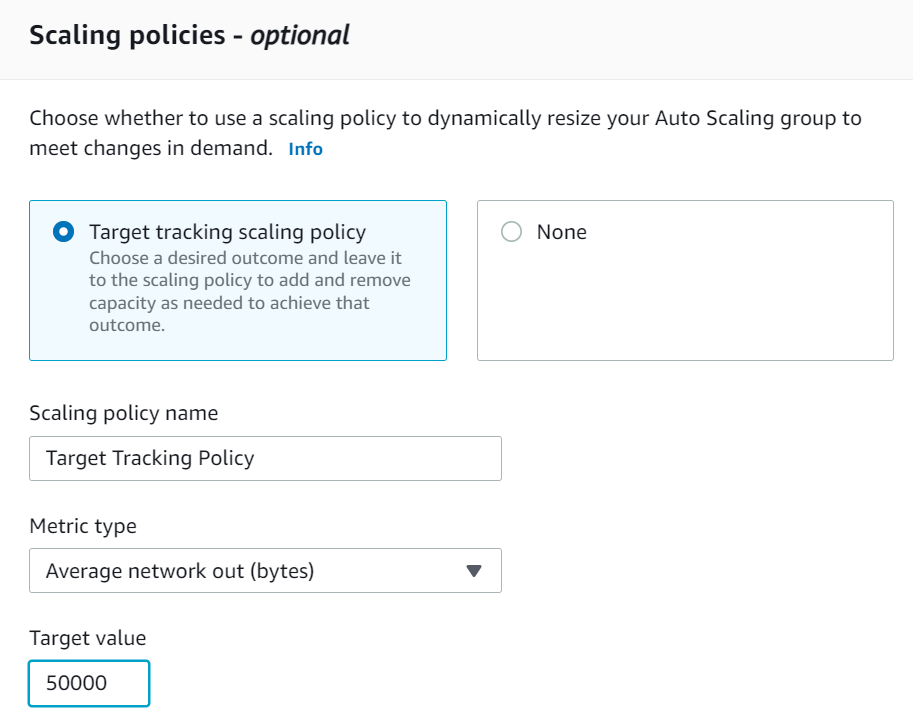
****

**Next**

****

****

****



Crate auto scaling

