😊 Creating Deployment groups for Auto Scaling and Load Balancing

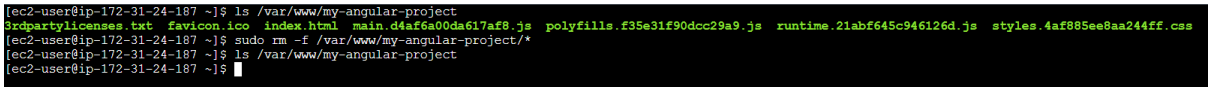
The task is about deploying an application using AWS services for high availability and reliability. Initially, you create an EC2 Auto Scaling group and an Application Load Balancer (ALB) to handle traffic and ensure your application remains available even during high demand. You start by creating an Amazon Machine Image (AMI) from your current EC2 instance, then use it to set up an Auto Scaling group through a launch template. Afterward, you configure the ALB to distribute traffic across instances.

Next, you update your CodeDeploy setup to work with the new Auto Scaling group and ALB, ensuring deployments can be managed efficiently. Finally, you update your deployment pipeline to reflect these changes. The end goal is to automate and scale your application deployment, enhancing its availability and reliability.

😄 To begin with the Lab:

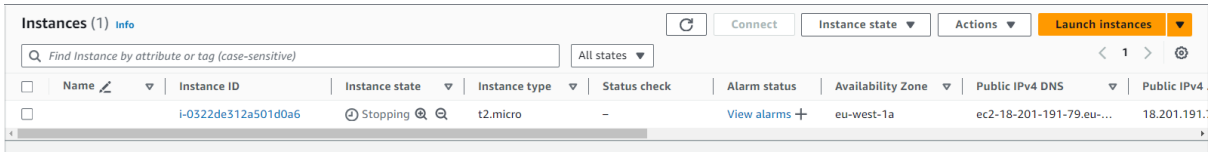
1. In the previous labs, we used AWS CodeDeploy to deploy to a single EC2 instance. But in real life, you primarily use EC2 auto-scaling groups and elastic load balancers to achieve high availability and reliability on AWS. So, in this lecture, you will learn to create deployment groups for them and update your pipelines accordingly.
2. To begin with, we will create an application load balancer and an EC2 auto-scaling group for our sample application. Now we need an EC2 launch template to create an auto-scaling group.
3. Therefore, we will first create an Amazon machine image, AMI, from our current EC2 instance to use in our launch template. But CodeDeploy raises errors if it finds files in the deployment folder that weren't deployed by the same deployment group.
4. So, let's reconnect to our EC2 instance to clean up. If you remember, we deploy to the '/var/www/my-angular-project' folder. As you see, it contains files from our previous deployments. They were also deployed by CodeDeploy but by our existing deployment group. Then, let's use the Linux 'rm' command with 'sudo' and the '-f' option to forcefully remove files and then append a slash followed by an asterisk to the folder path to select all files for deletion.

   ls /var/www/my-angular-project
   sudo rm -f /var/www/my-angular-project

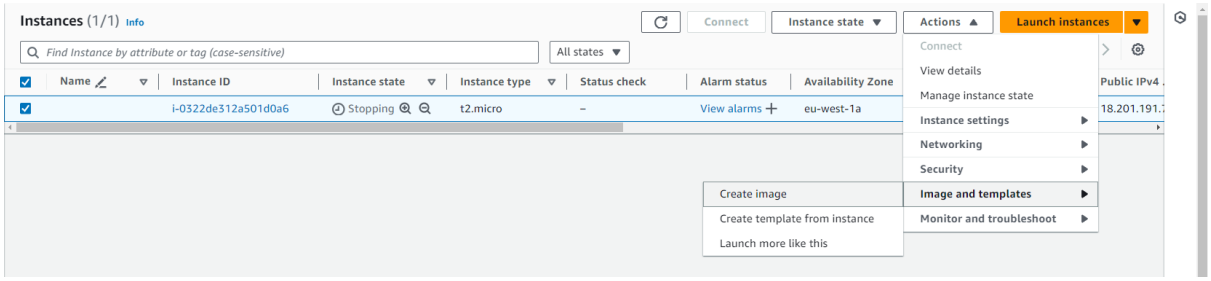


😁 Step 1: Creating AMI and Launch template

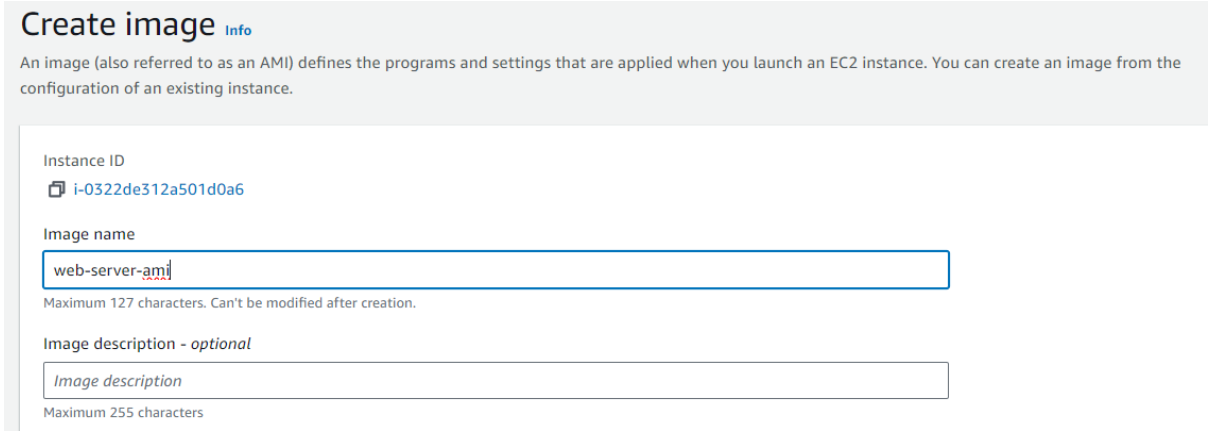
5. Now go back to EC2 and stop your instance. Below you can see that your instance has been stopped.



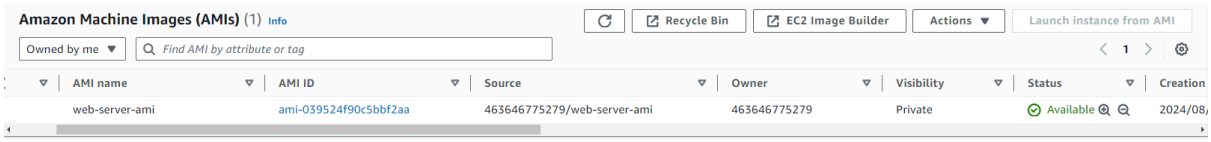
6. Now you need to select your instance and click on Actions then you need to choose Image and template and click on Create Image.

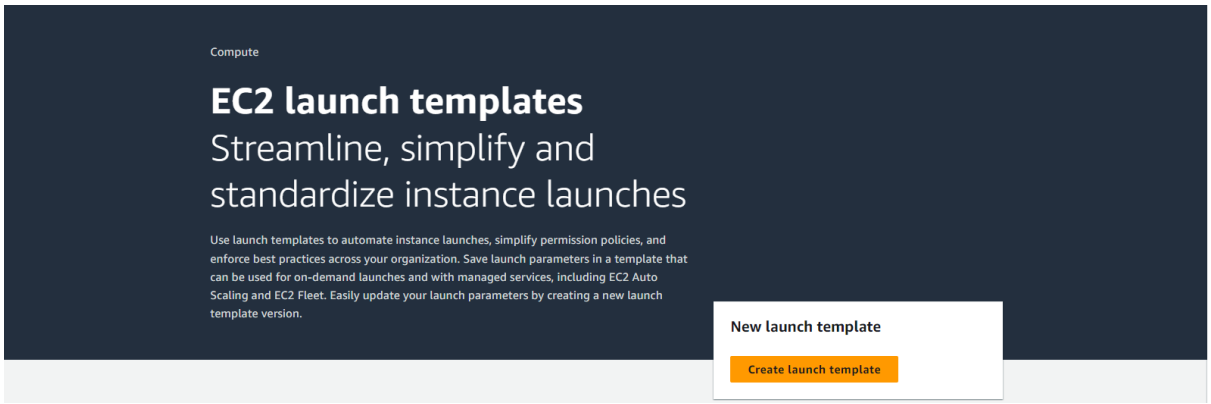


7. Now here just give your image a name and keep everything to default then create your image.

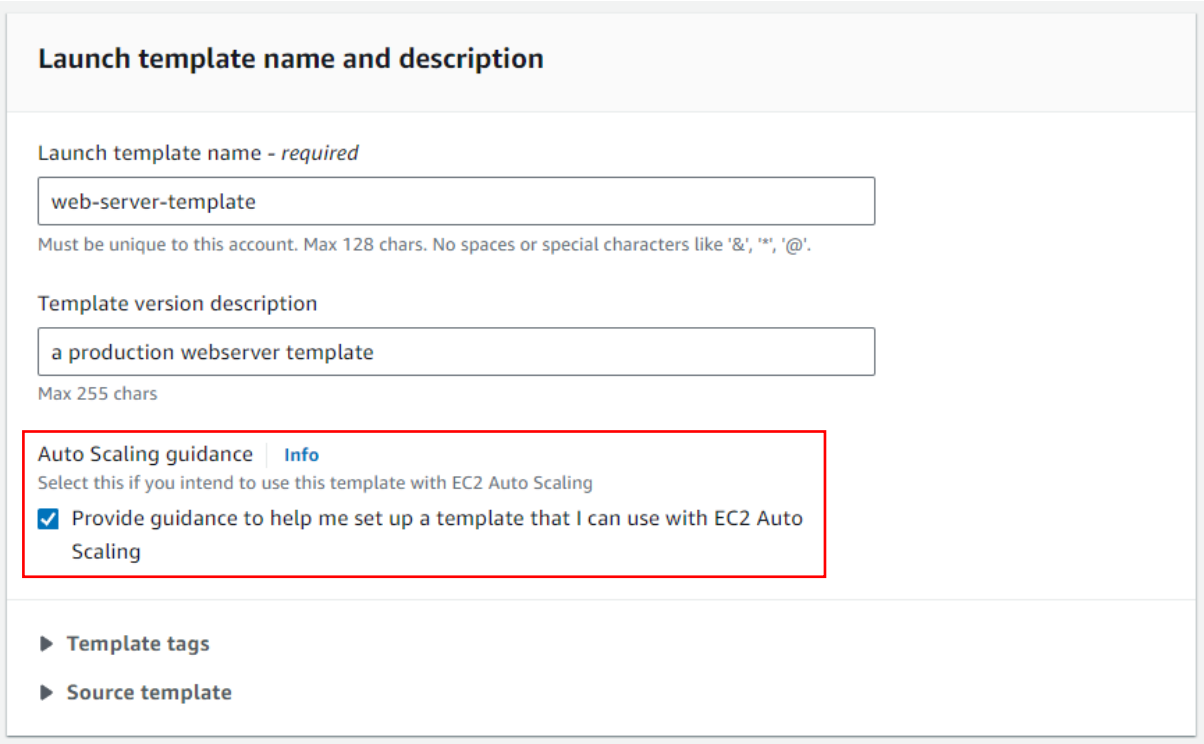


8. Now come to AMI and check whether it is available or not. When it is available you need to go to the Lauch template section in EC2 and click to create a Lauch template for auto-scaling.

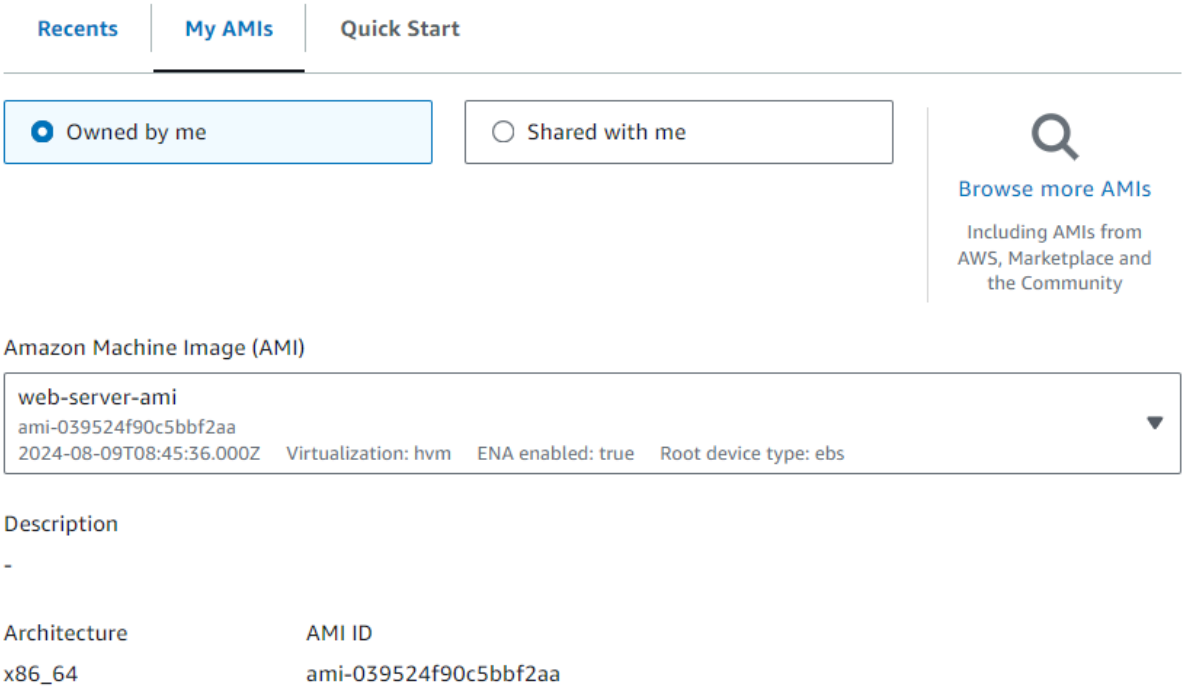




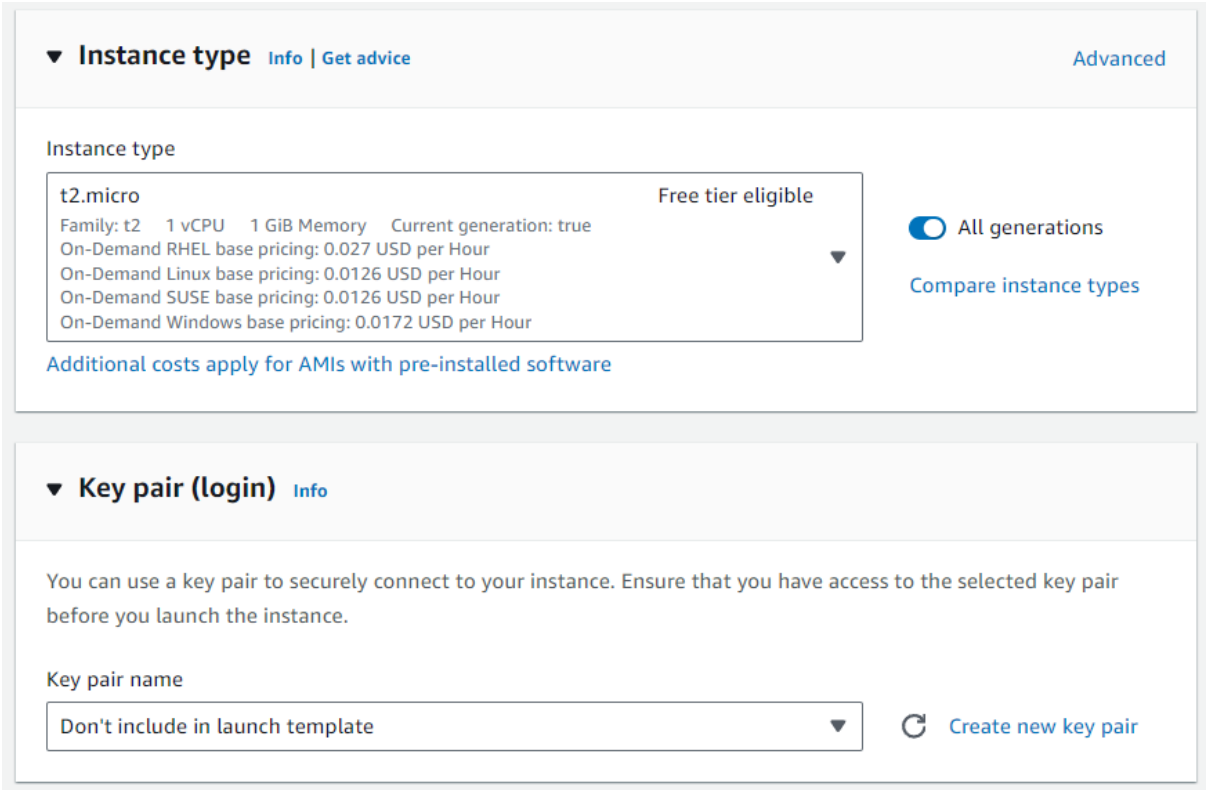
9. First you need to give a name to your launch template then enable auto-scaling guidance.



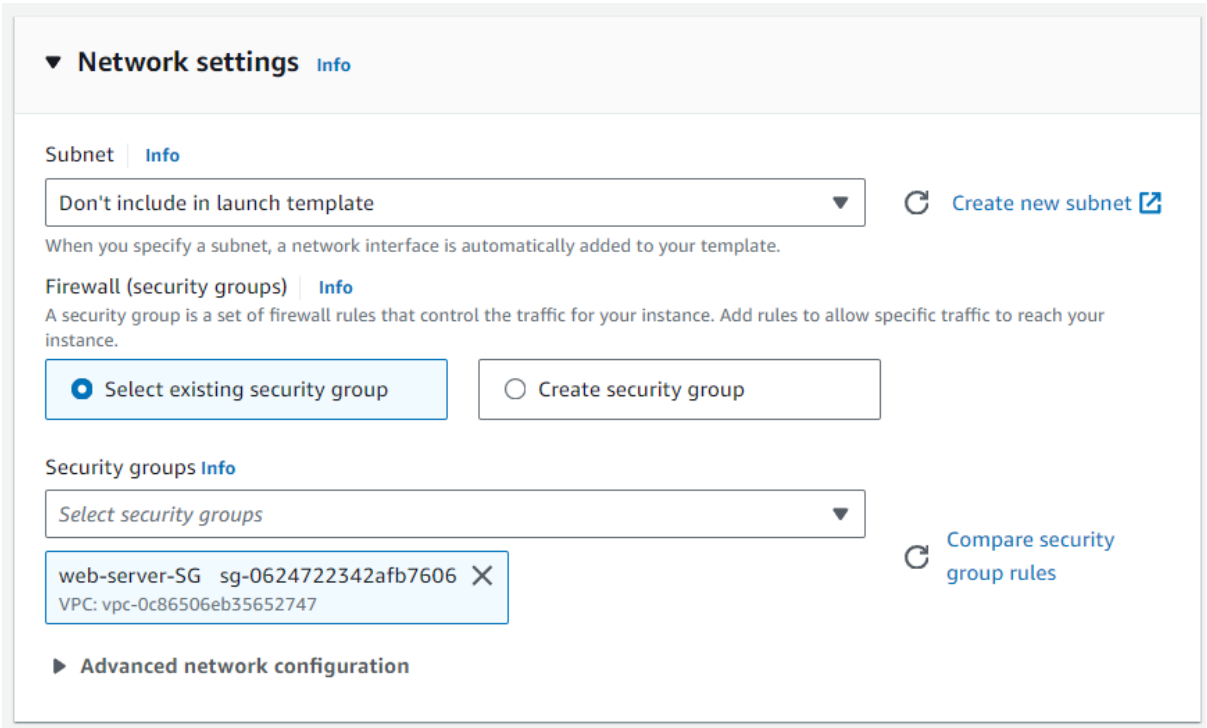
10. Then in the application and OS images section choose My AMI and then choose your web server AMI that you just created.



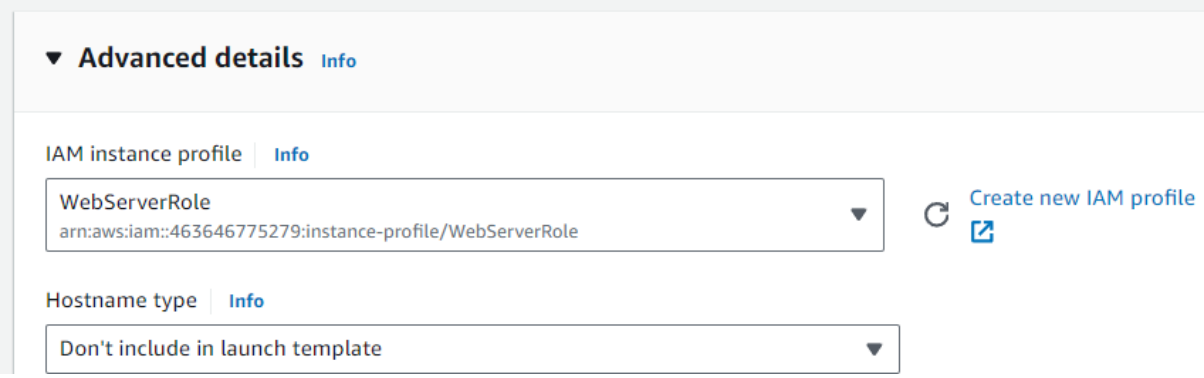
11. After that in the instance type choose t2.micro and in the key pair it is up to you to include a key pair or not.



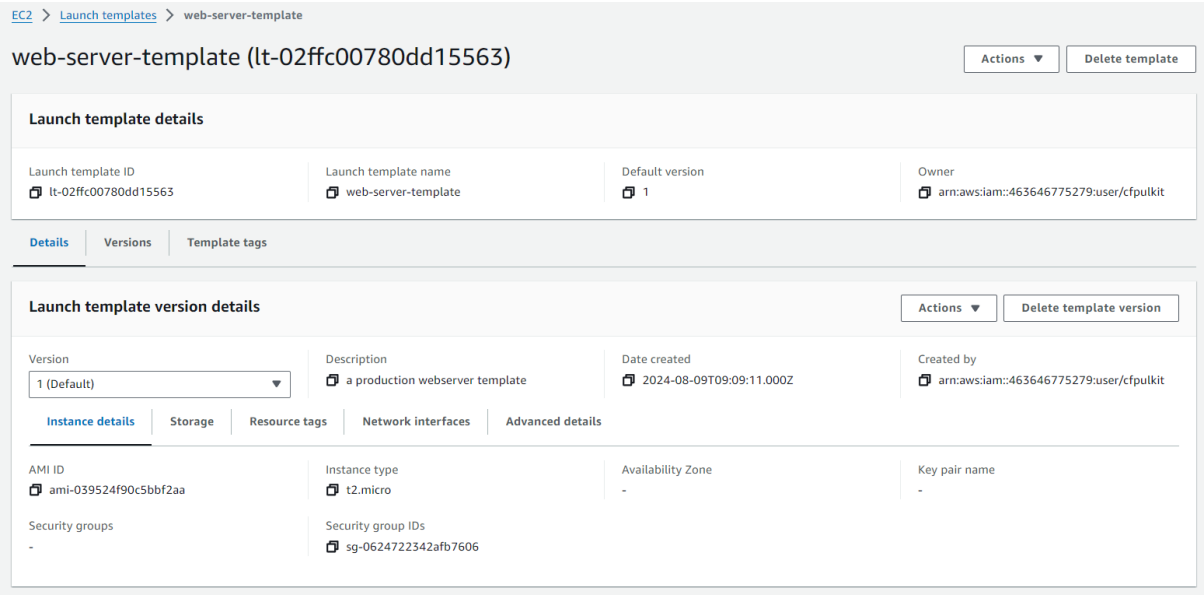
12. Then in the Network settings you just need to choose your existing security group.



13. In the end expand the advanced details option and choose the IAM role that you had created in the previous lab to attach it to your instance.
14. After that just launch your template.

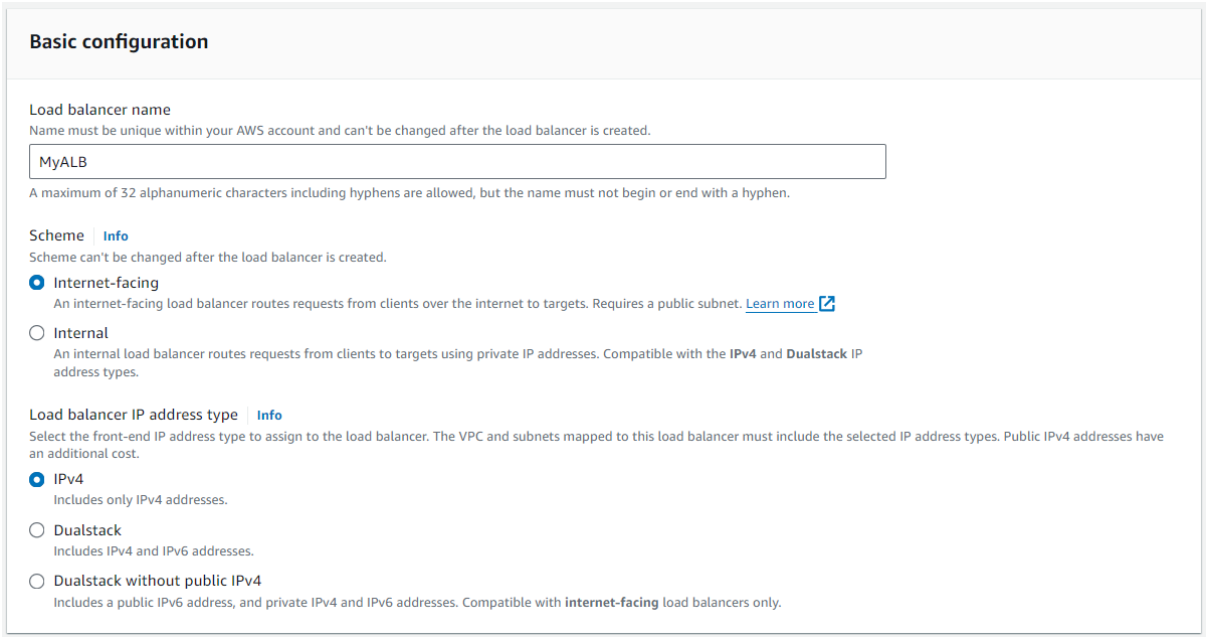


15. Below you can see your launch template.

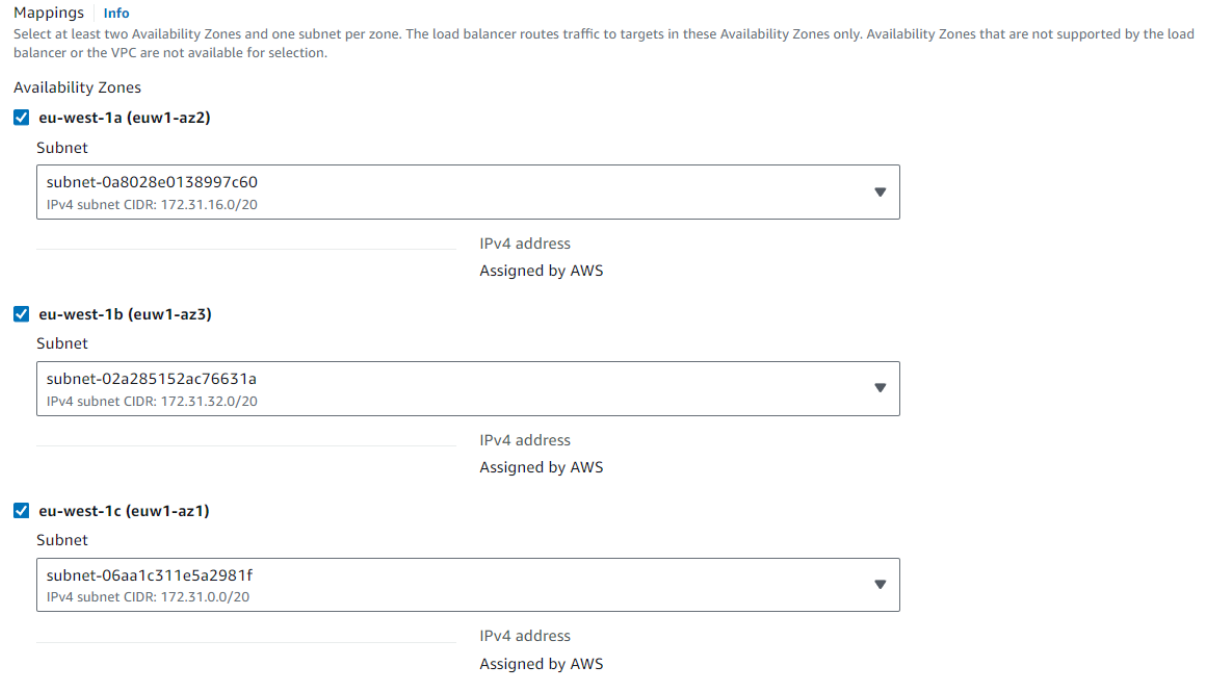


😁 Step 2: Creating Application Load Balancer

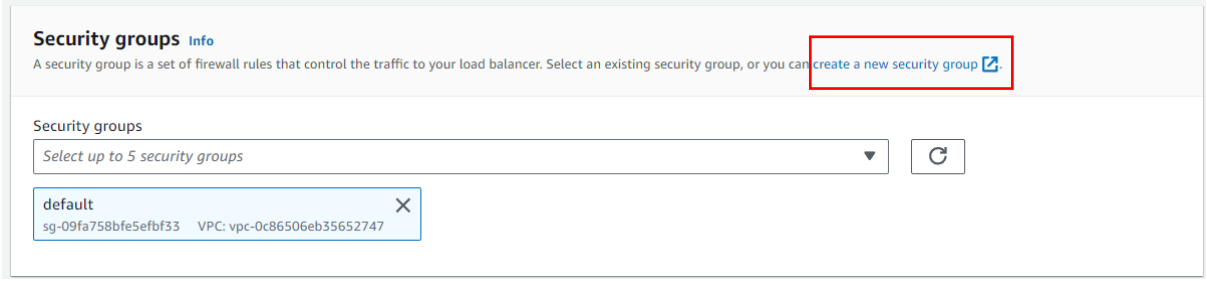
1. Now go to Load Balancer in EC2 and click on Create Load Balancer. Then choose to create an **Application Load Balancer**.
2. Now give the load balancer a name and keep other things to default and scroll down.



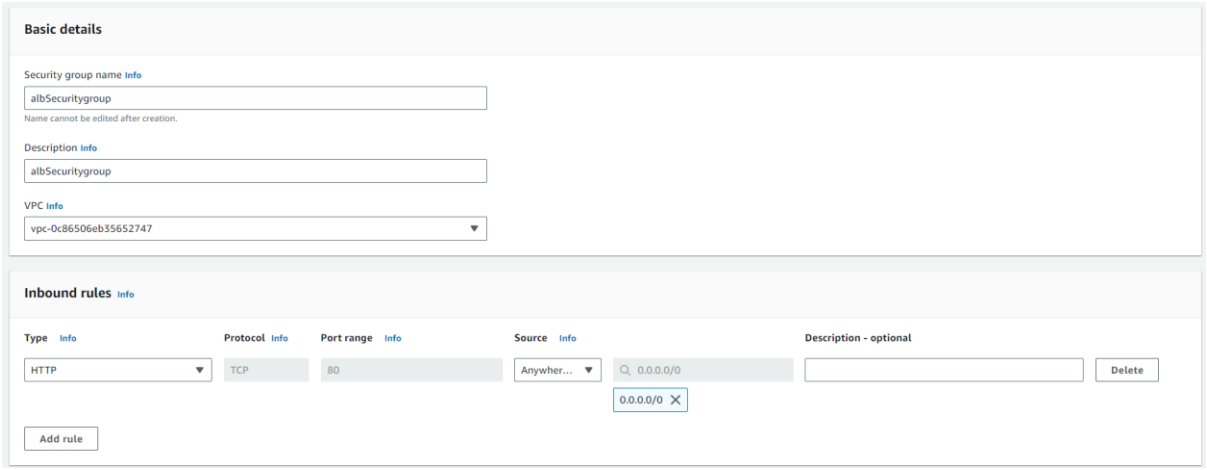
3. Then in the network mappings you need to choose all the Availability Zones so that the traffic can be balanced easily.



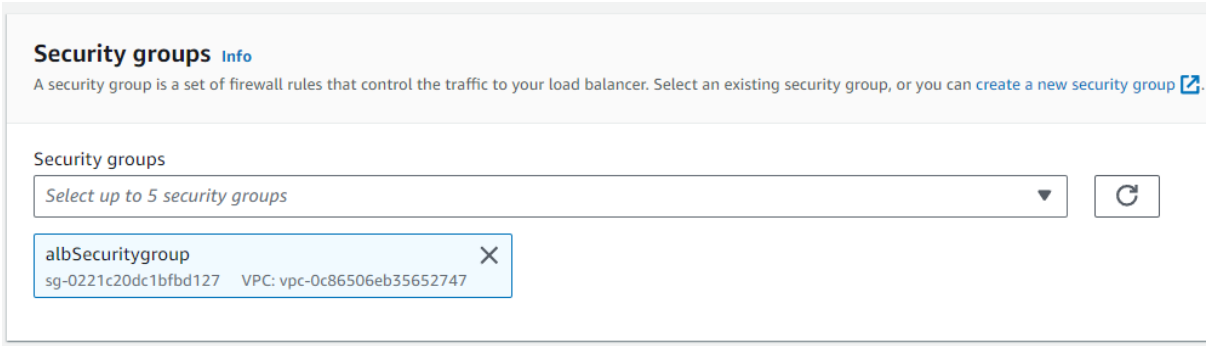
4. Now for the security group click on create a new security group.



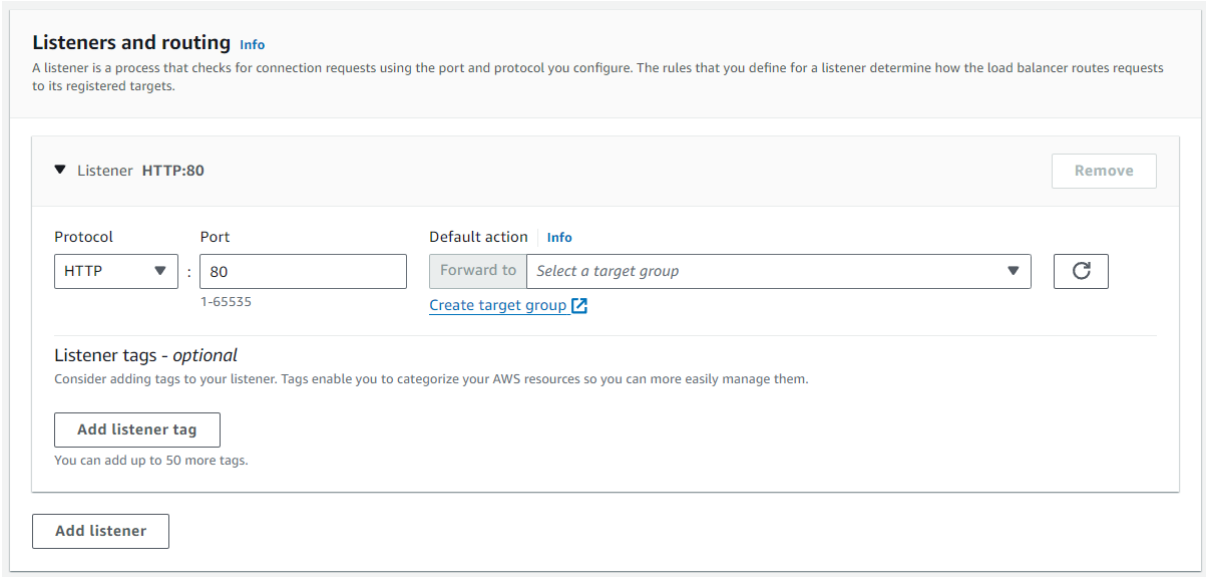
5. To create a new security group, you just need to give it a name and description then in the inbound rules you need to open Port 80 HTTP from everywhere and create your security group.



6. Now come back to ALB and choose your security group, also remove the default SG.

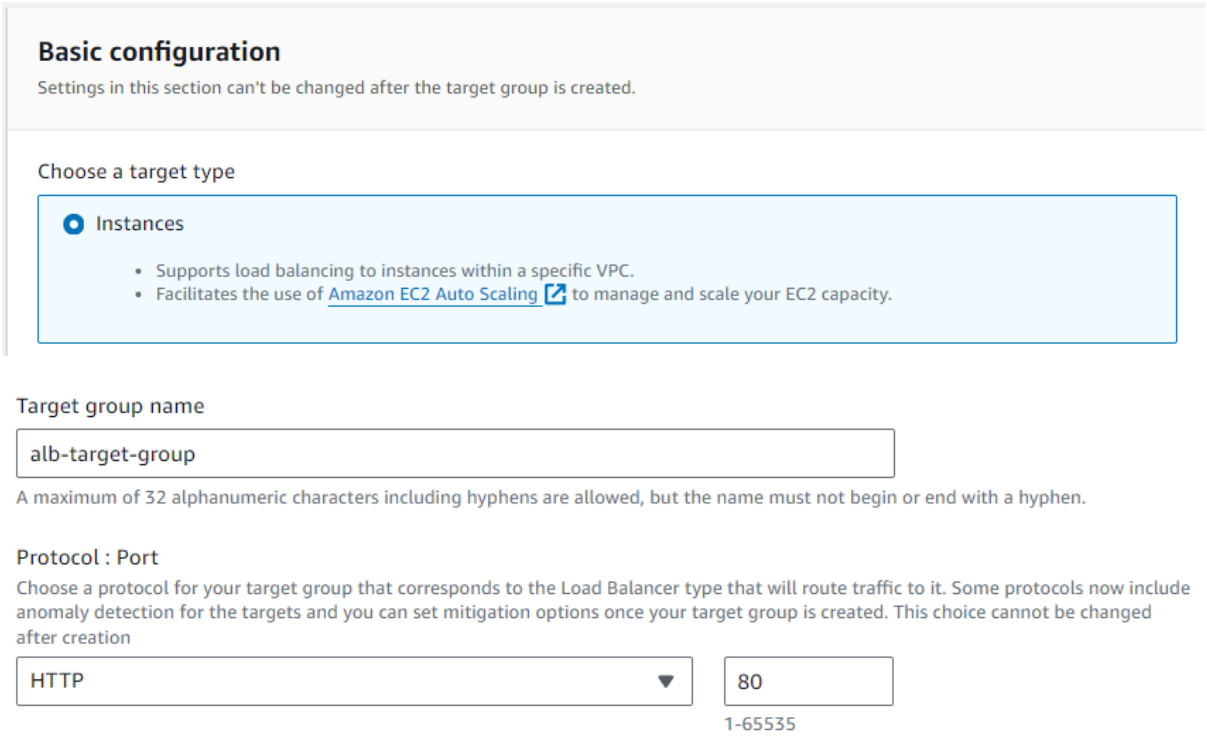


7. Then we also need a target group for our ALB. So, click on create target group.

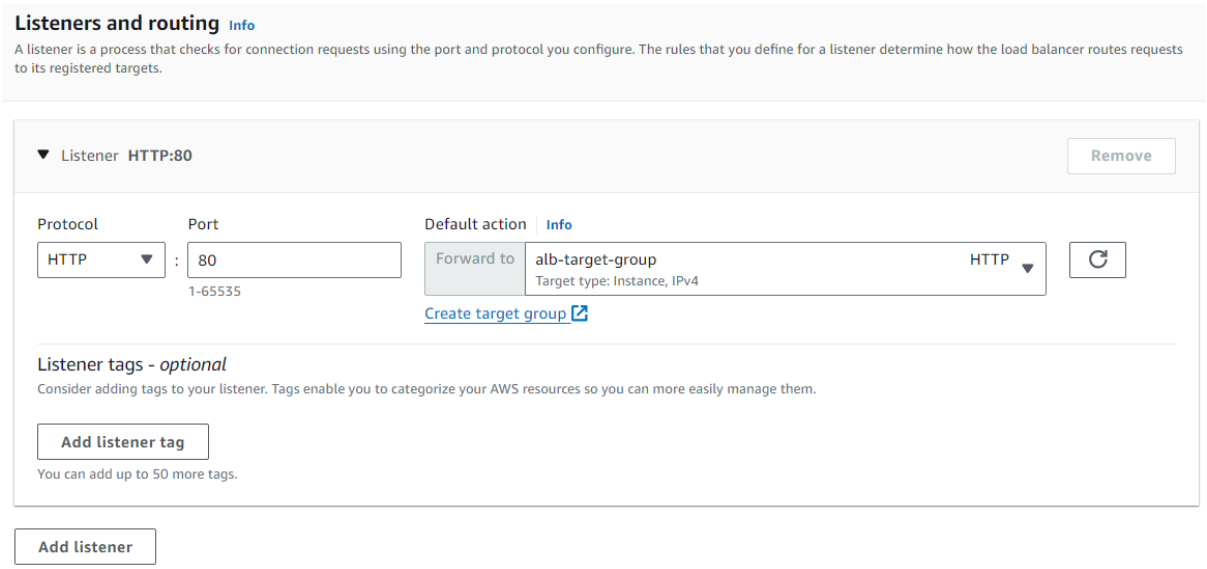


8. To create a target group first in the target type, choose Instacnes then scroll down, and you need to give a name to your target group. Leave the rest of the settings to default and move to the next page.

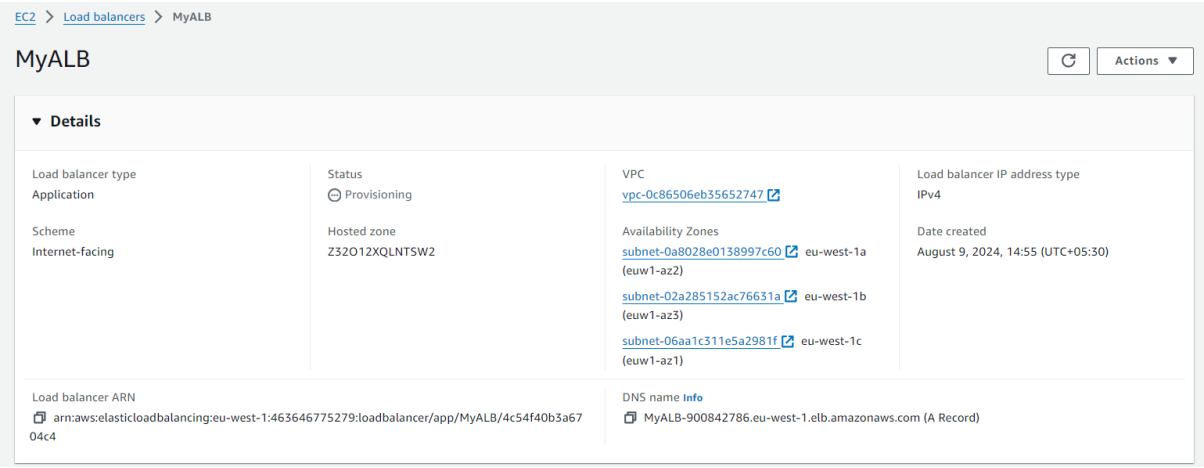
9. After that you don’t need to choose any target now so, just click on create target group.



10. Now come back to ALB and choose your target group. After that go ahead and create your load balancer.

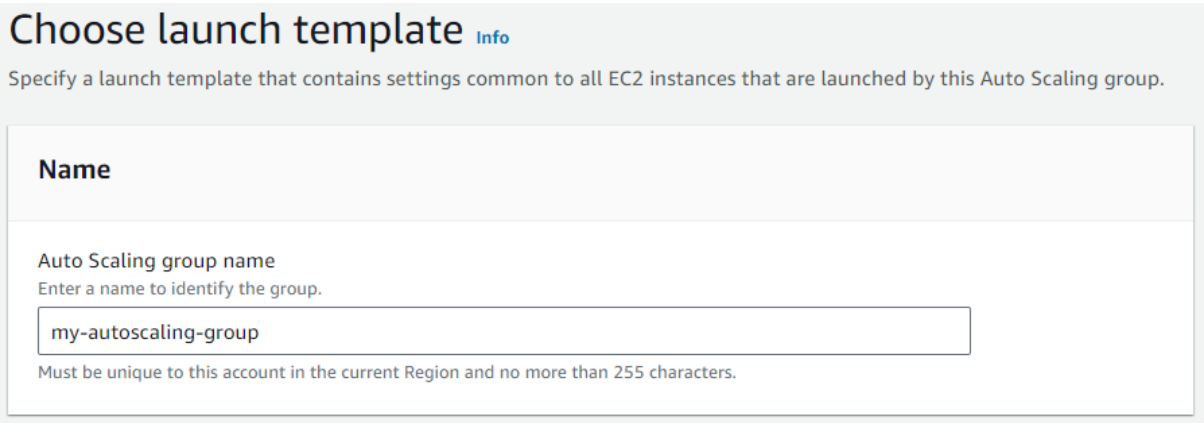


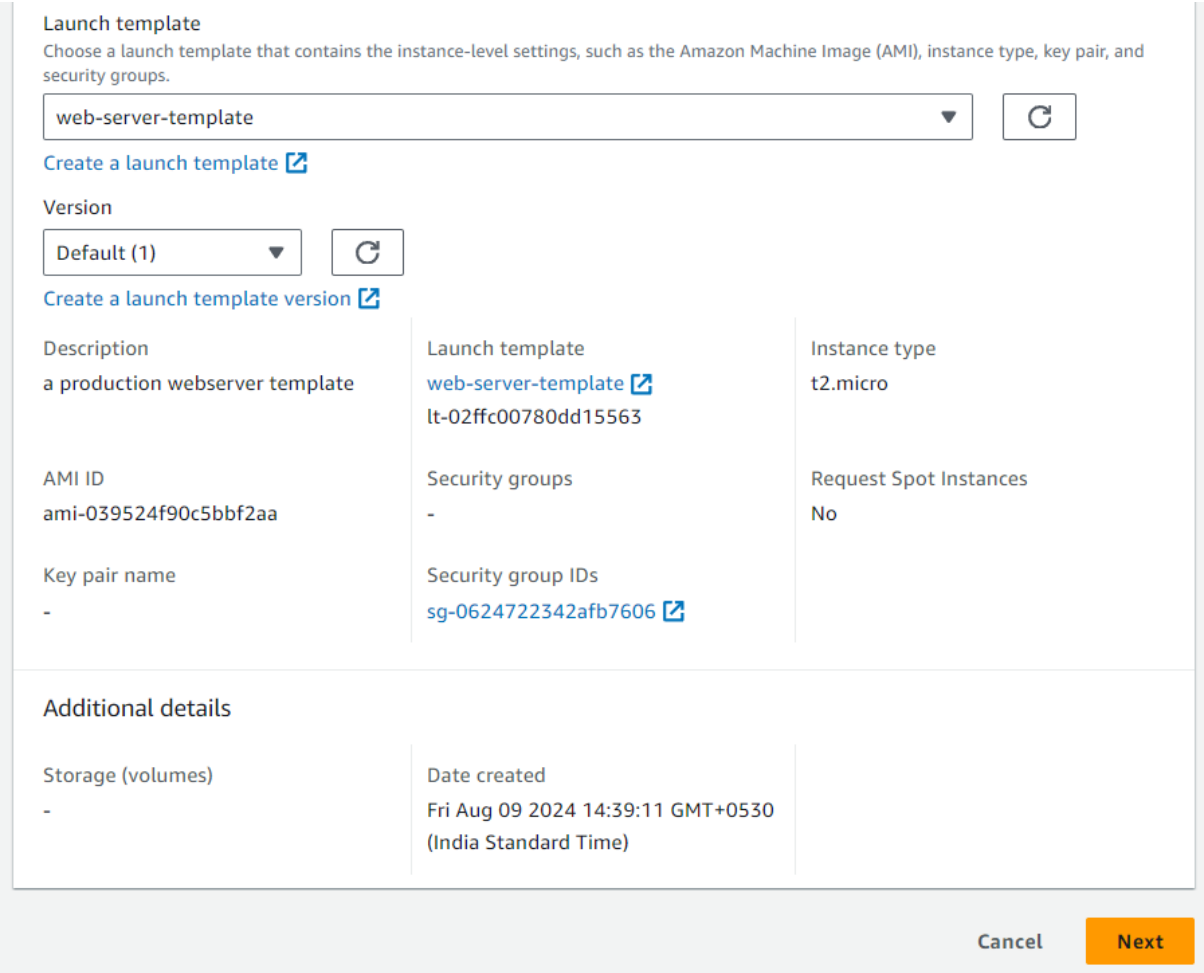
11. Below you can see that our load balancer is in provisioned state.



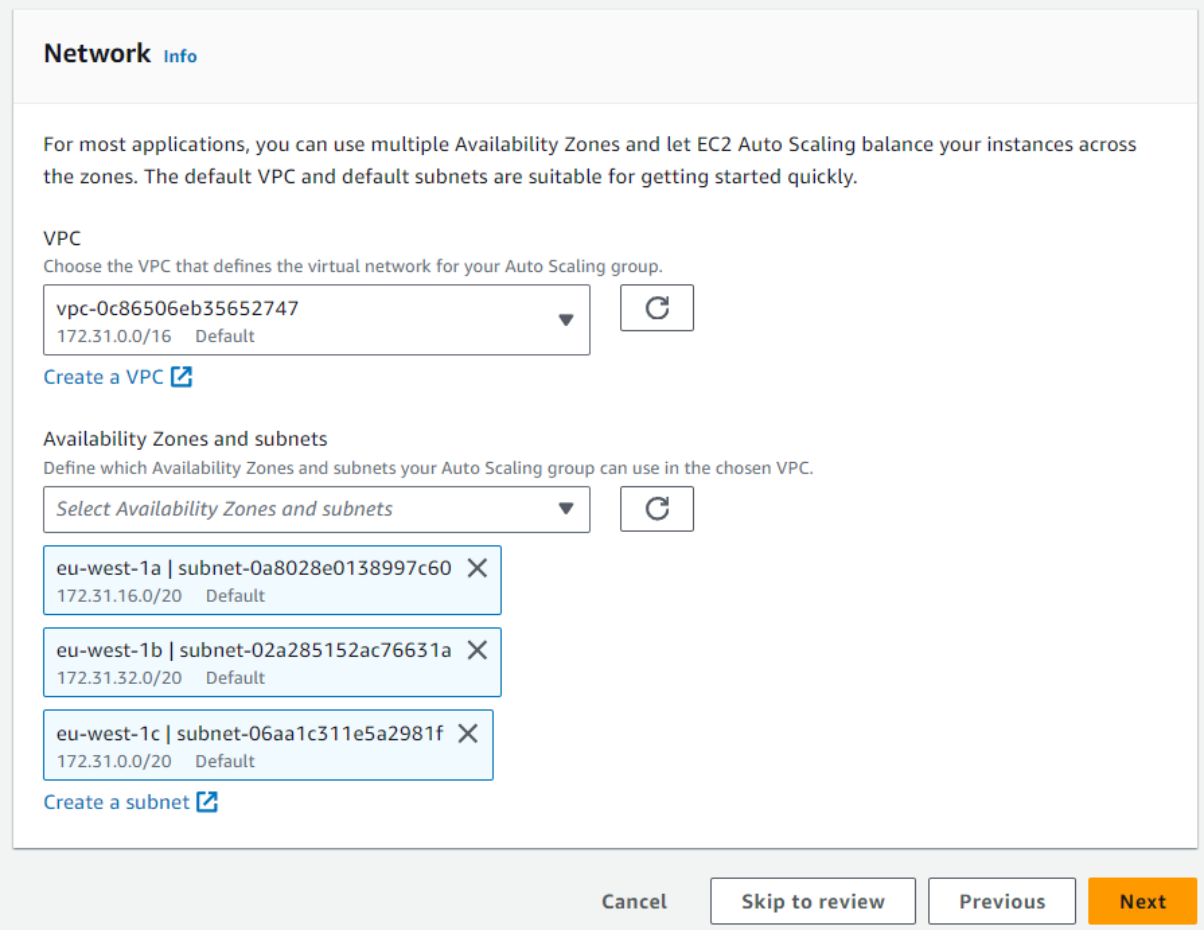
😁 Step 3: Creating Auto Scaling Group

1. Now you need to go to the Auto Scaling group and click to create one. Now on step 1 you need to give a name to your auto scaling group. Then you need to choose your template and move forward.

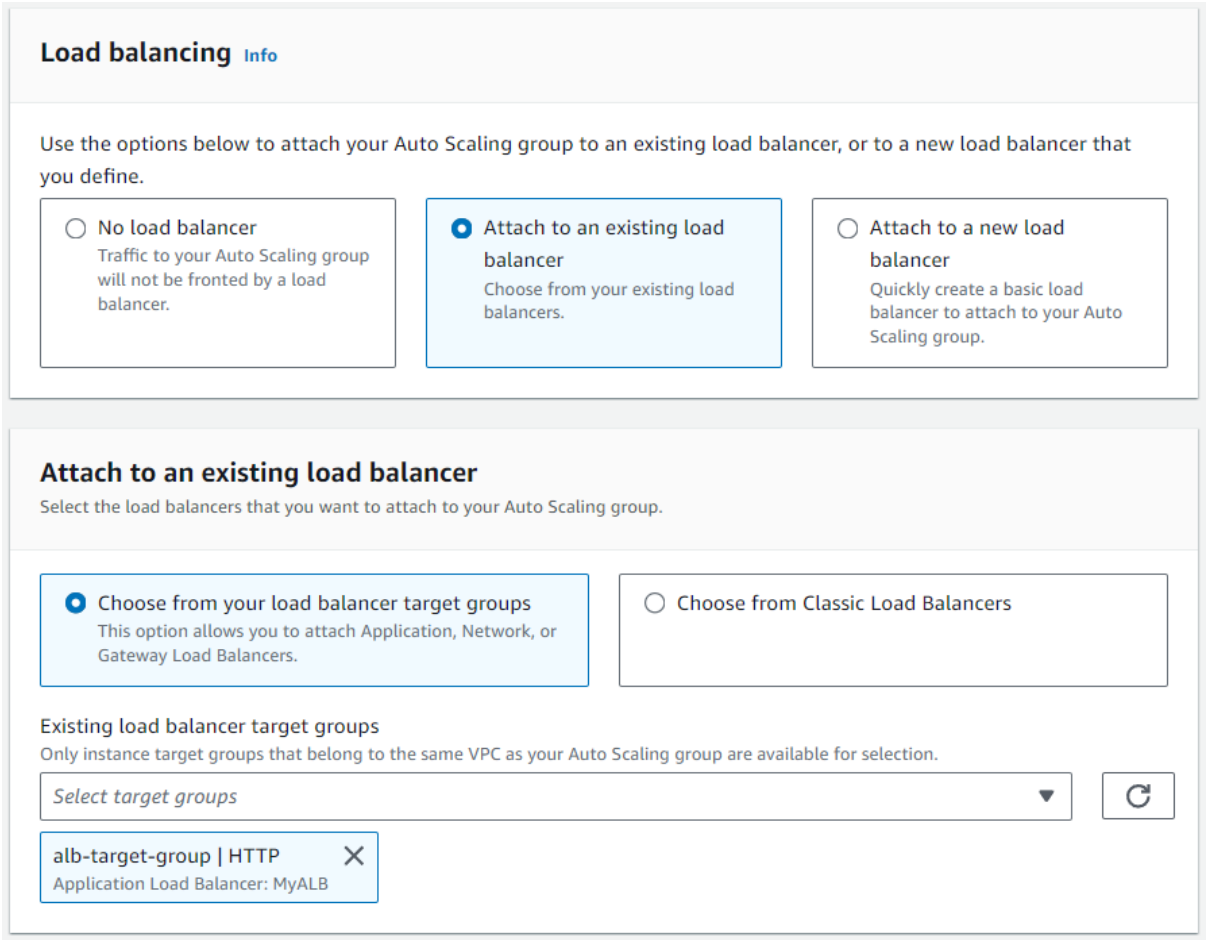




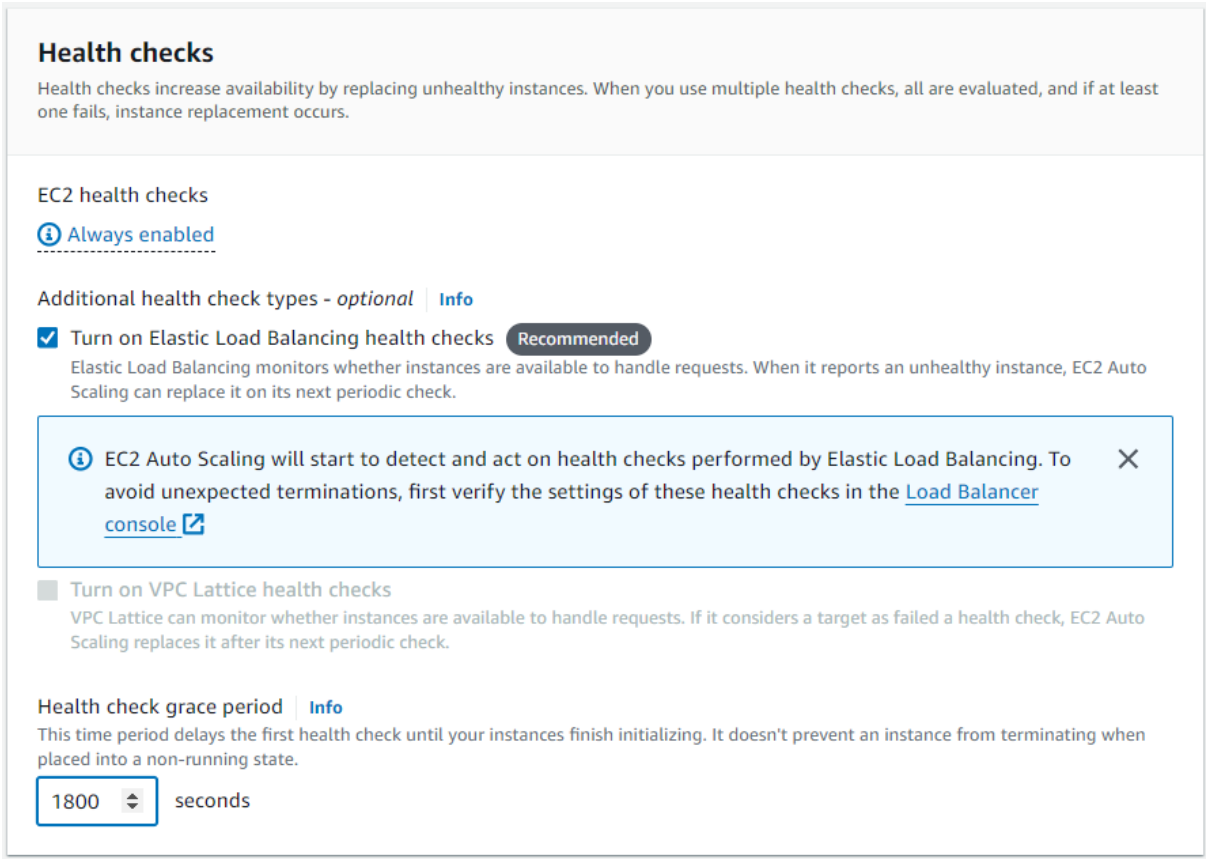
2. now you need to choose all the Availability zones and subnet from your default subnet and click on Next.



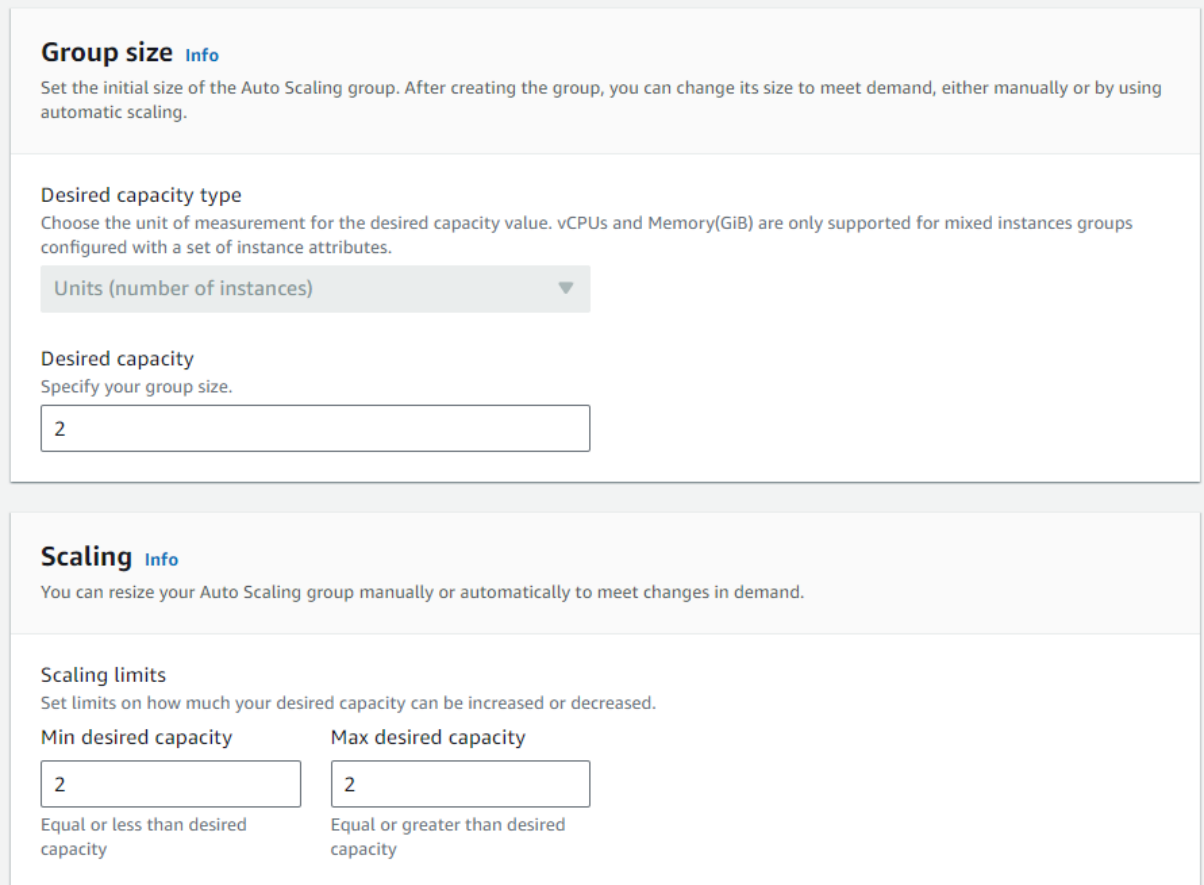
3. Then you need to choose to attach to an existing load balancer, then choose your existing Target groups.



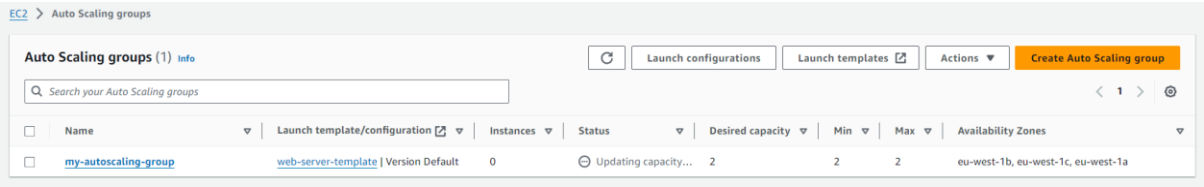
4. After that scroll down to health checks and turn on elastic load balancing health checks and change the health check grace period to 1800 seconds.



5. Now in the group size choose the capacity to 2 in all aspects. Then move to the next page which will be the review page. Now click to create your auto-scaling group.

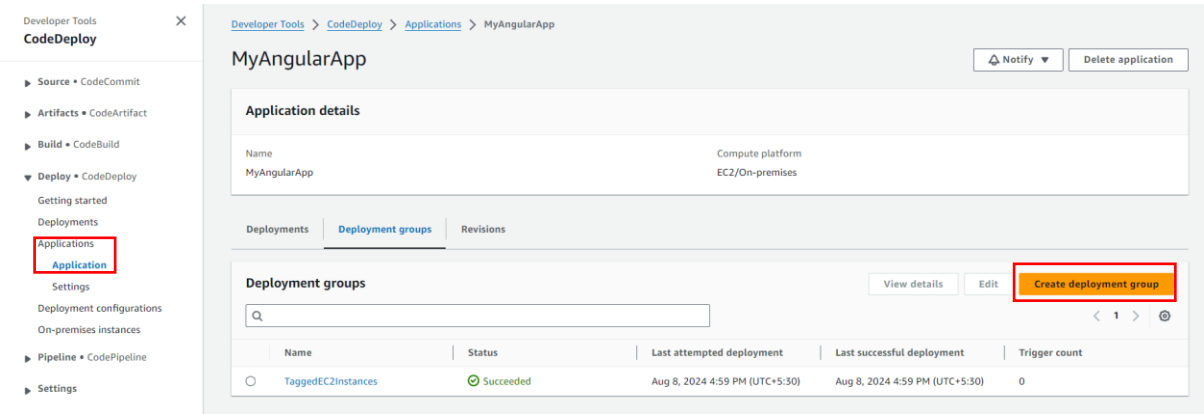


6. Below you can see that our auto-scaling group is updating.

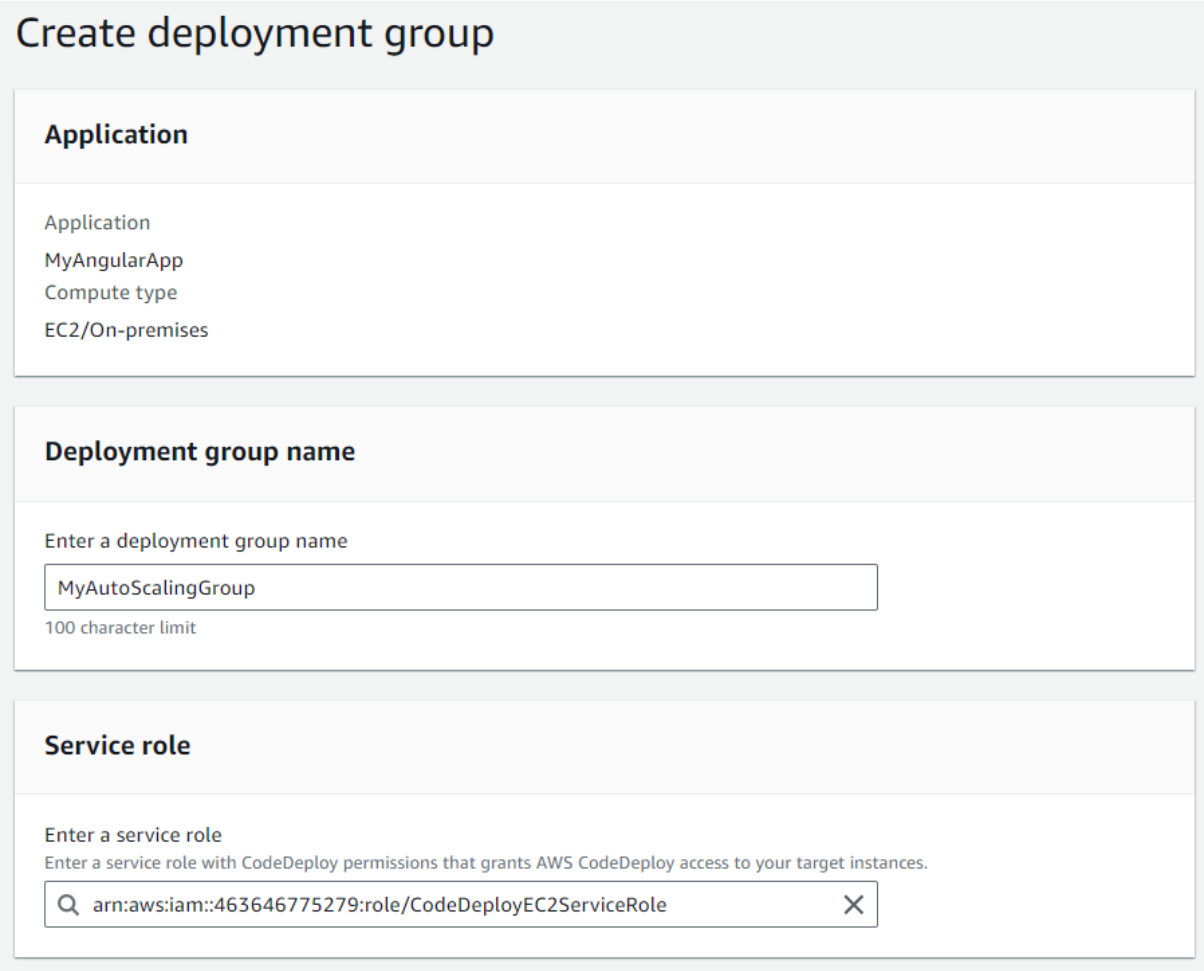


😁 Step 5: Update Deployment Groups

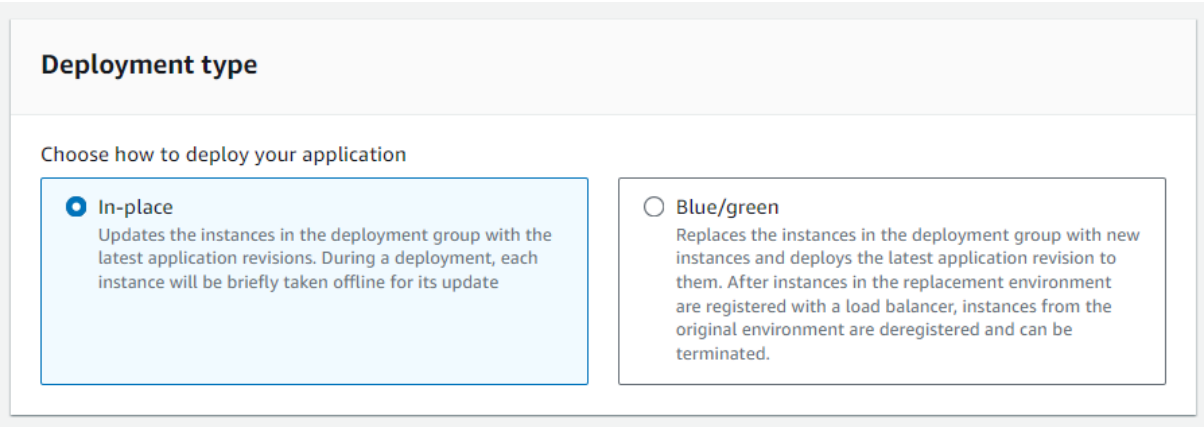
1. Now go to CodeDeploy to update the deployment group. So, first go to applications then inside your application and from here you need to create a deployment group.



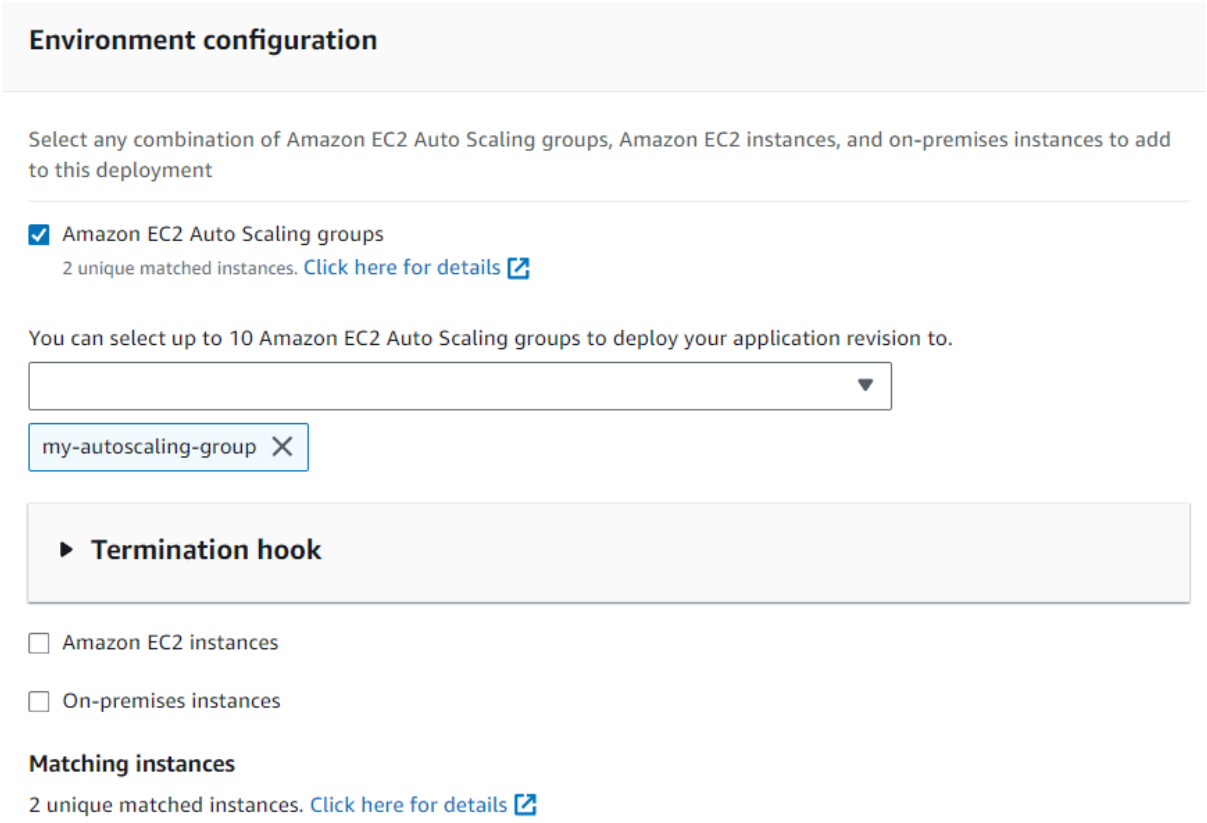
2. Now you need to give this deployment group a name then choose your IAM role which you created for your previous deployment group.



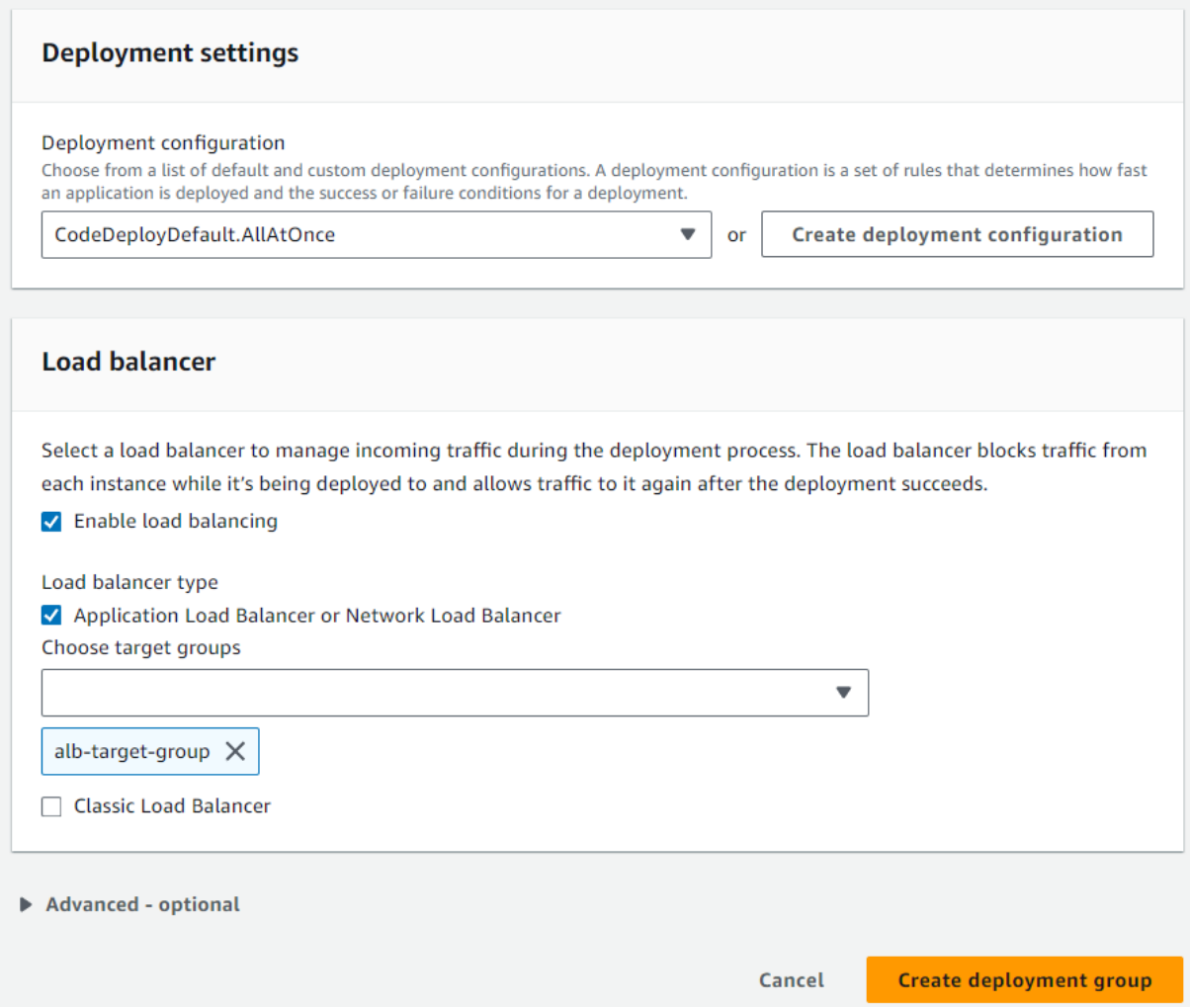
3. Then keep deployment type as In-place.



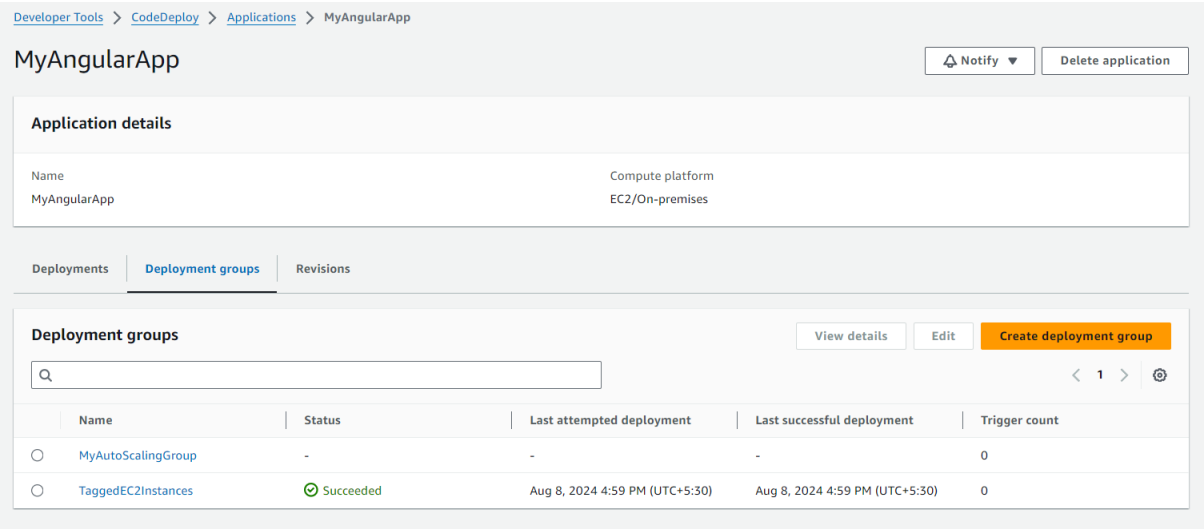
4. After that you have to choose Amazon EC2 Auto Scaling groups.



5. Now in the end you need to keep deployment settings to default and click to enable load balancing then to choose the load balancer type to ALB and choose your target group. After that create your deployment group.

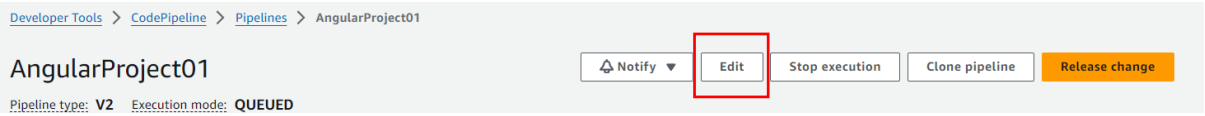


6. Below you can see that our deployment group has been created.

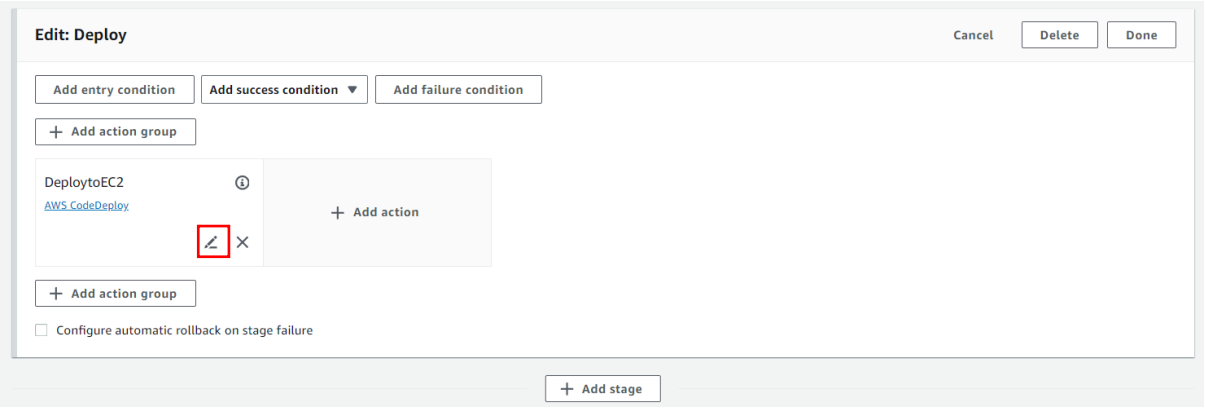


😁 Step 6: Update the Pipeline

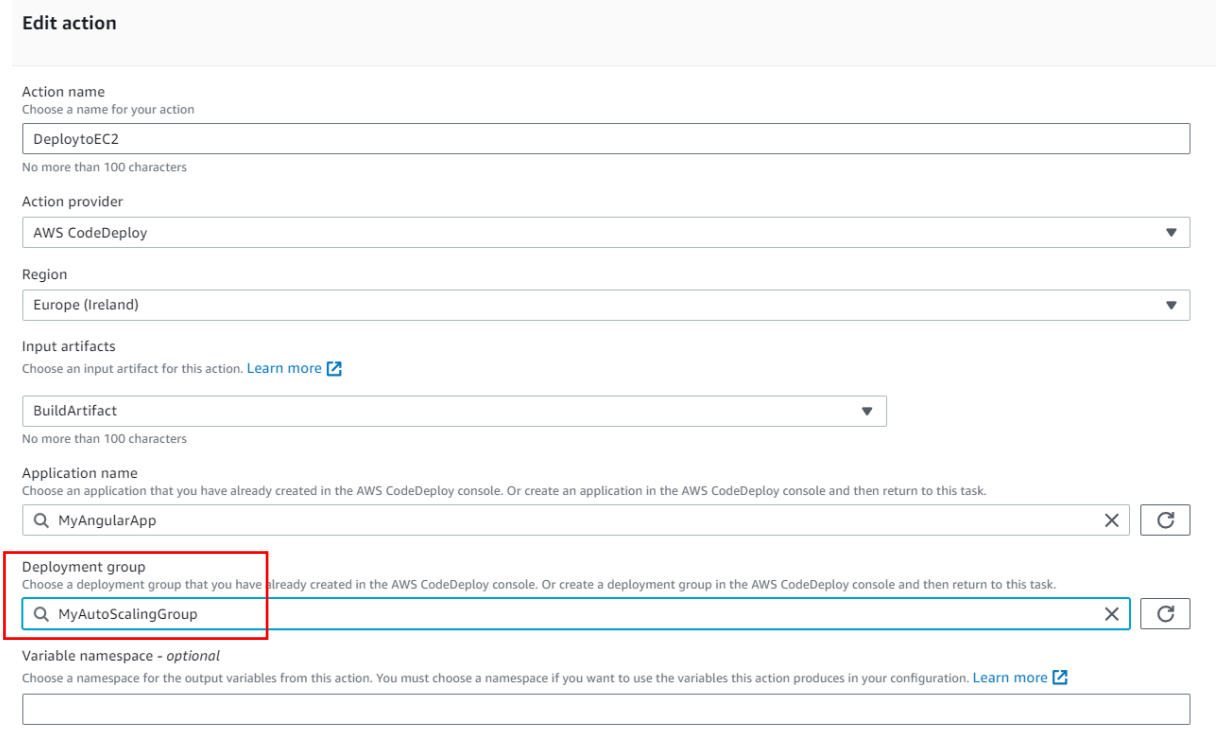
1. Now navigate to the code pipeline and update the deployment stage of our pipeline. Here you need to click on edit.



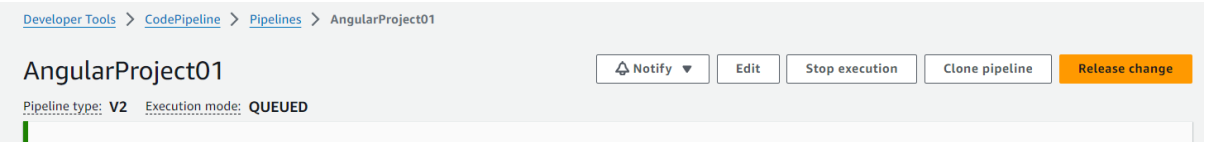
2. Then scroll down to the bottom choose your deploy stage and click on edit. Now we need to edit our deploy stage for that click on the highlighted icon.



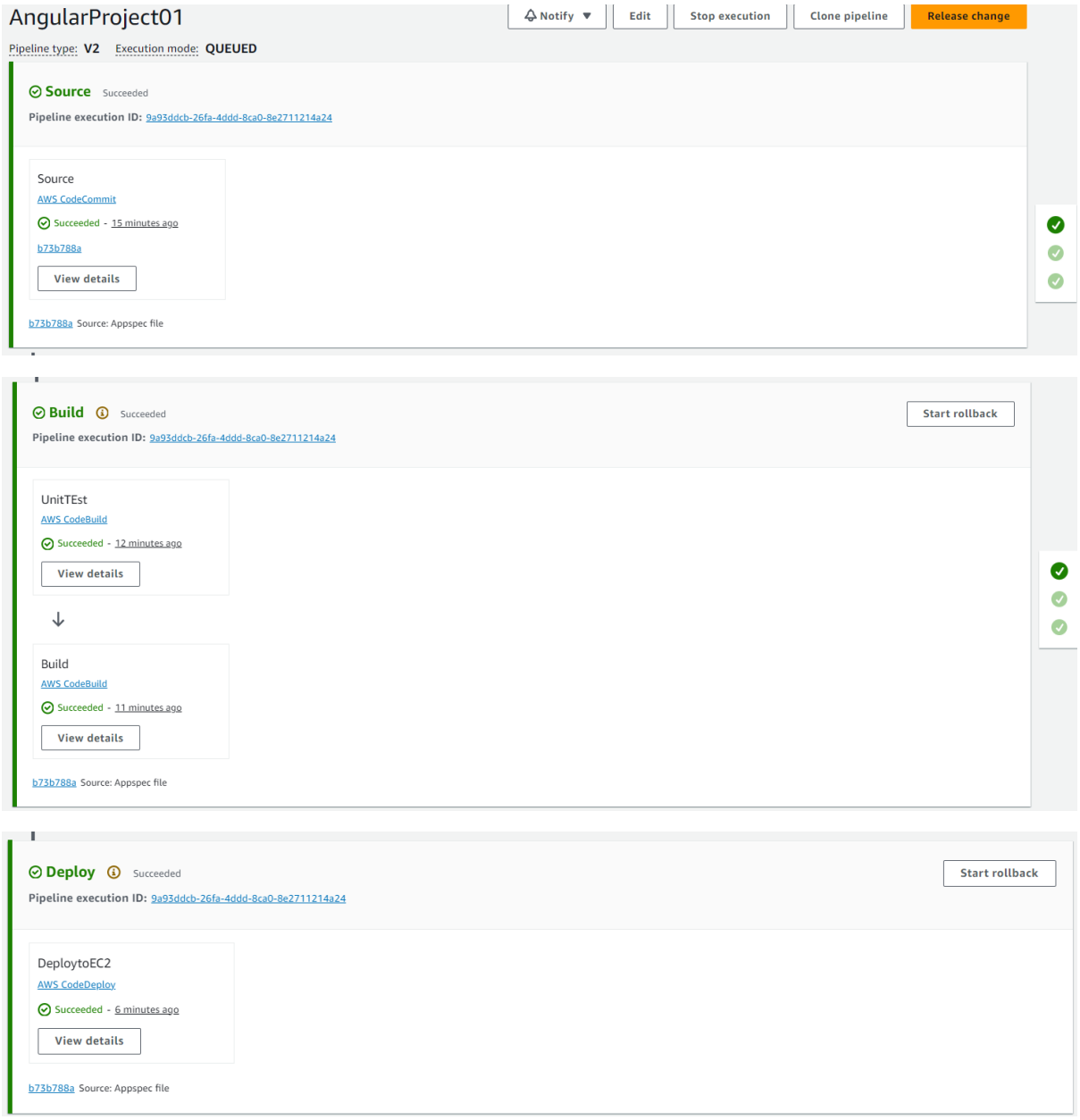
3. Here you just need to change the deployment group as you can see below. Then save your change.



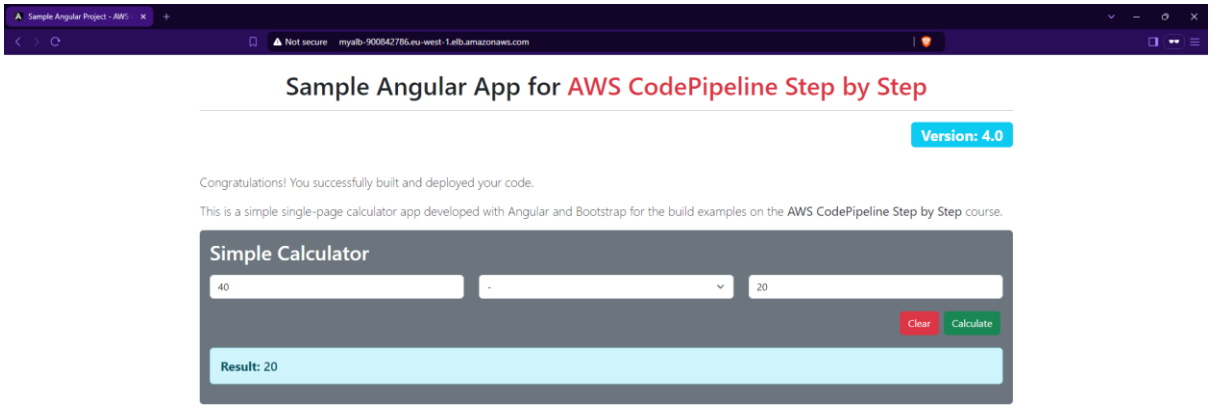
4. Now we are going to release our changes manually. For that, you need to click on Release change.



5. Now you need to wait until your pipeline gets executed. Below you can see that your pipeline has been executed.



6. Now if you want to see details regarding deployment then you can do that yourself and go to the Application load balancer and copy the DNS name.
7. Then paste that DNS name in a new tab, here you can see that your application is online.


Sample Angular Project - AWS
Not secure
myalb-900842786.eu-west-1.elb.amazonaws.com
Sample Angular App for AWS CodePipeline Step by Step
Version: 4.0
Congratulations! You successfully built and deployed your code.
This is a simple single-page calculator app developed with Angular and Bootstrap for the build examples on the AWS CodePipeline Step by Step course.
Simple Calculator
40
-
20
Clear
Calculate
Result: 20