Contents

Lab 4: Amazon Load Balancing

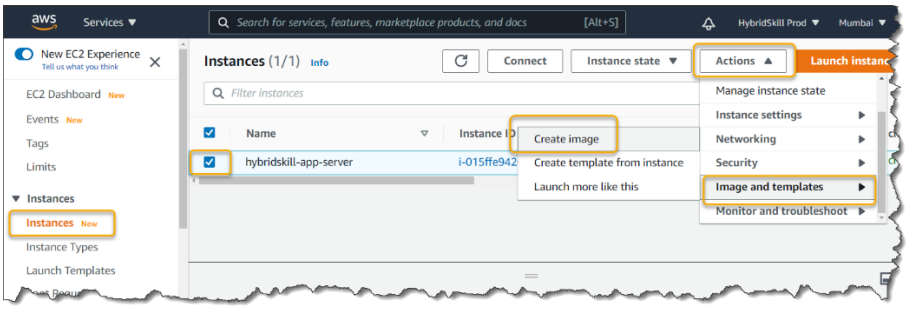
In this lab we will explore amazon elastic load balancing features

Task Breakdown

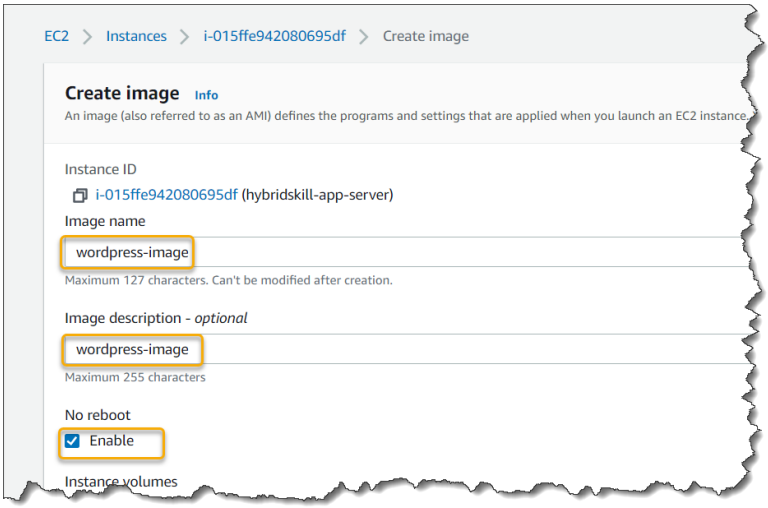
- Take an AMI of your wordpress instance
- Launch another instance using the AMI
- Create a load balancer and add both the instances under it.
- Test if load balancing is working successfully.

Task 1: Take an AMI of your Wordpress instance

1. On the Main **EC2 Dashboard**, on the left-hand side of the screen, select **Instances,** Select your wordpress instance. Next click **Actions** and then select **Image** and then click **Create Image.**



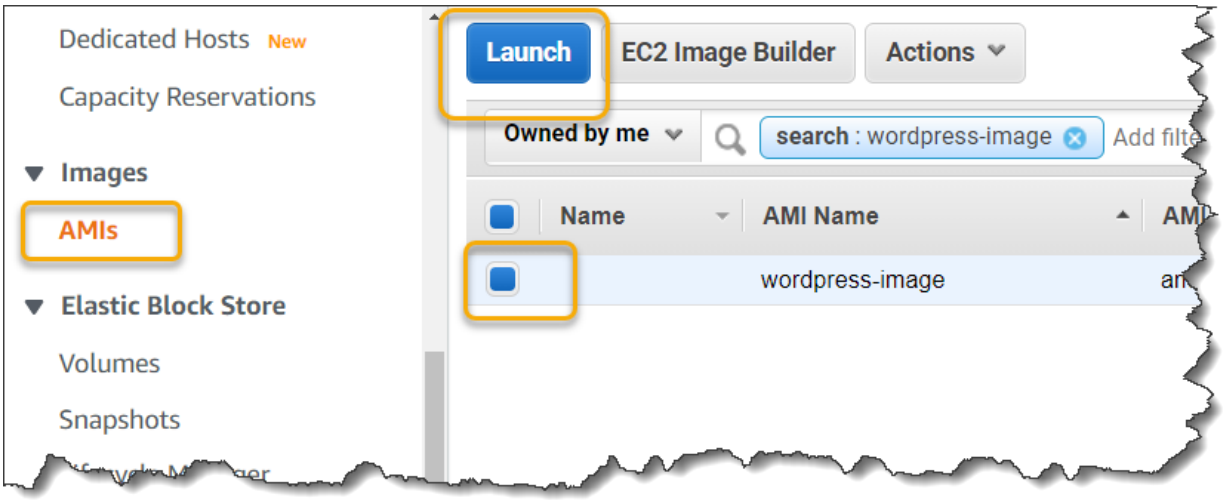
2. Enter a **wordpress-image** as the **Image name**, enter a short **Image description**, Select the **No reboot** option and finally click **Create Image**.



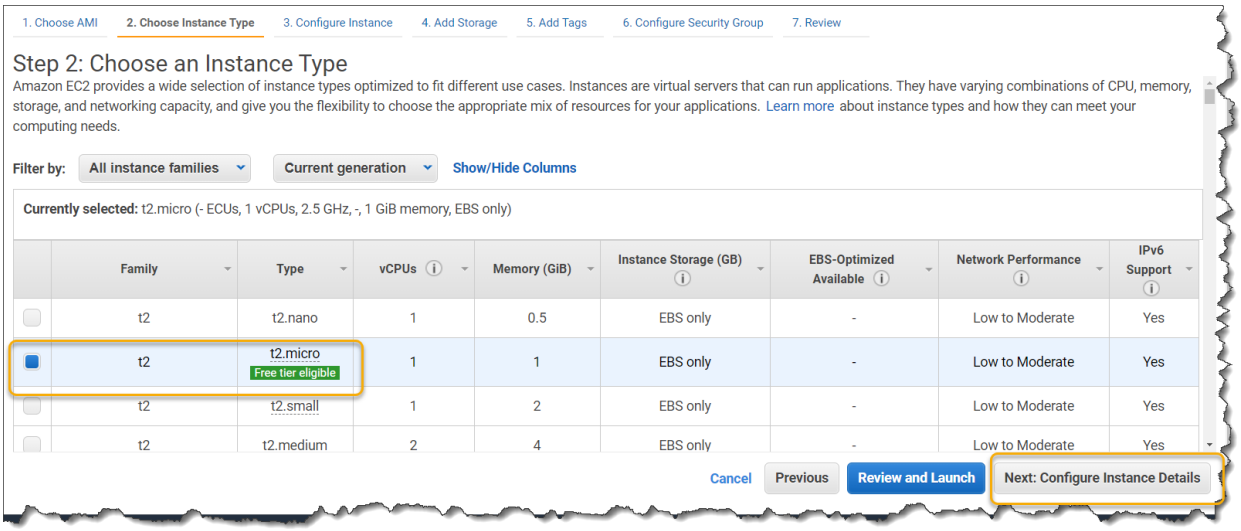
3. Your AMI should be successfully created.

Task 2: Launch another instance using the AMI

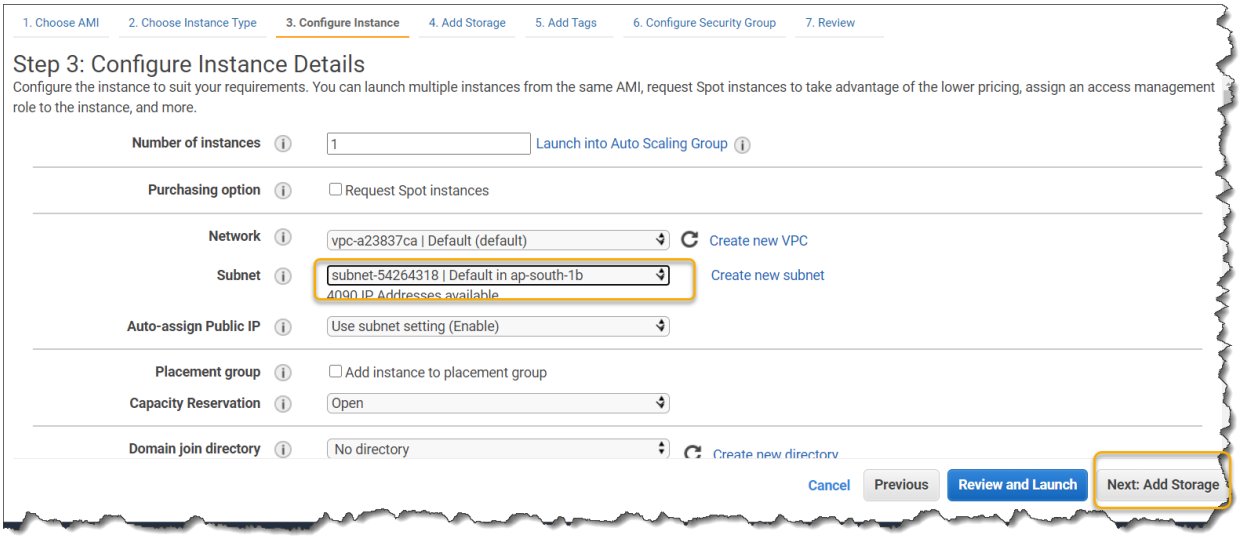
1. Click **AMIs**, **select** your **wordpress-AMI** you created earlier and click **Launch**



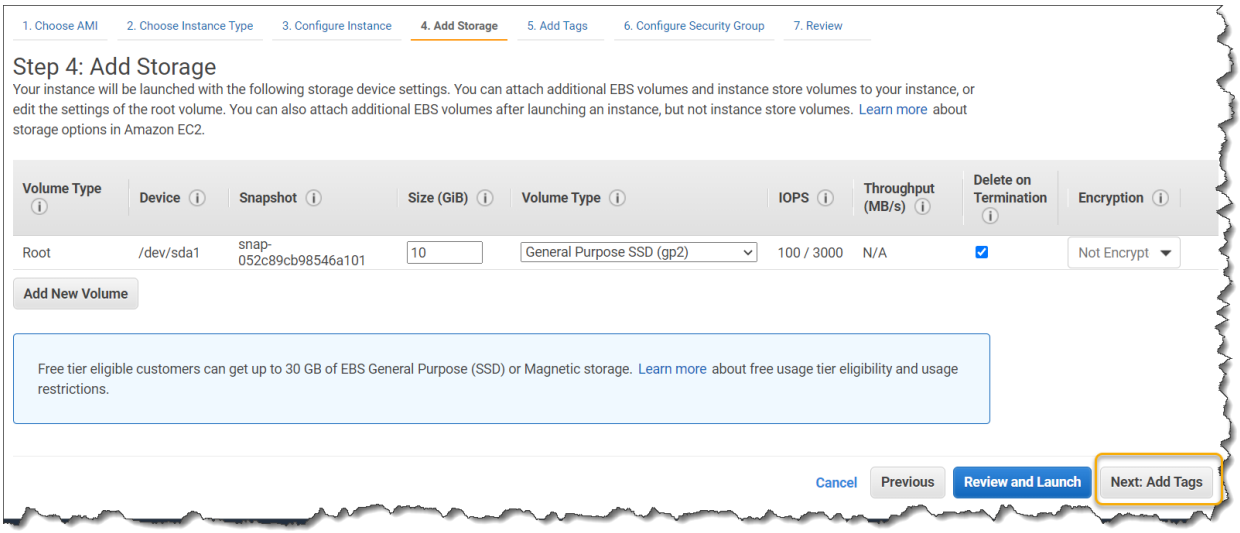
2. As before select **t2.micro** as the **Instance Type** and Click **Next: Configure Instance Details**



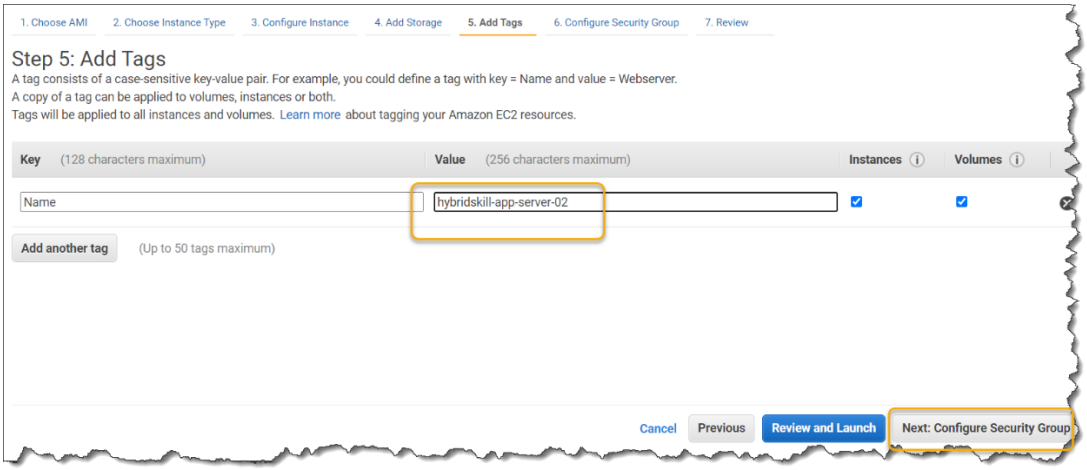
3. For the **Subnet**, select the other AZ this time (**subnet-xxxx ap-south-1b** in my case). Leave all options as default and click **Next: Add storage.**



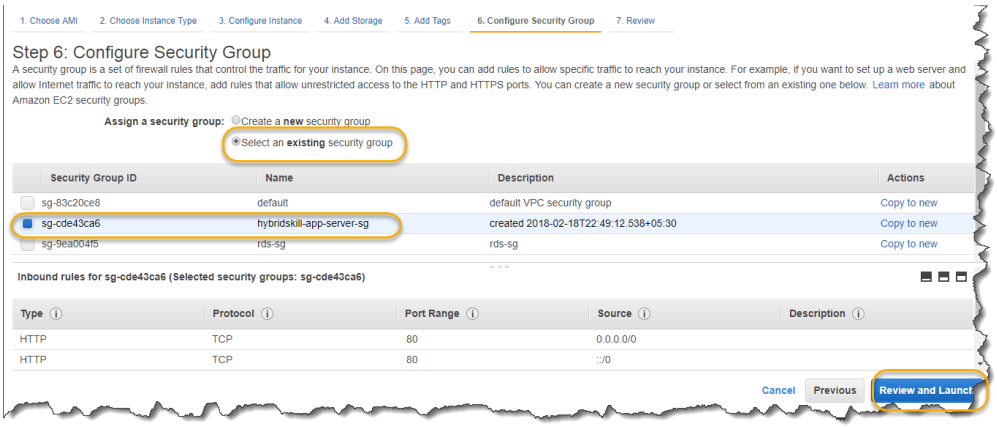
4. Leave all options as default and click **Next: Add Tags.**



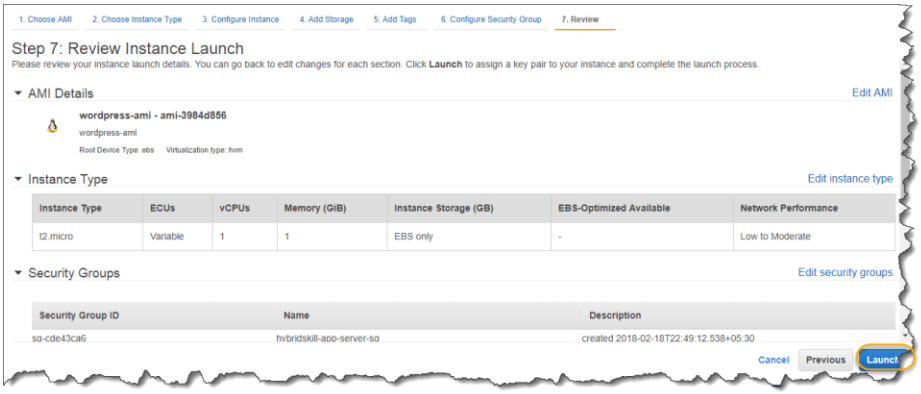
5. This time give **hybridskill-app-server-02** as the name of the instance and click **Next: Configure Security Group**



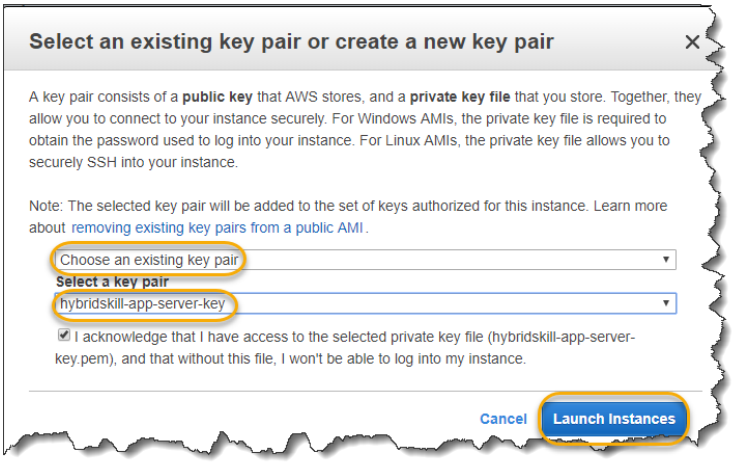
6. Click **Select an existing security group,** select your **hybridskill-app-server-sg** as the name of your security group and click **Review and Launch.**



7. Review your settings and click **Launch.**



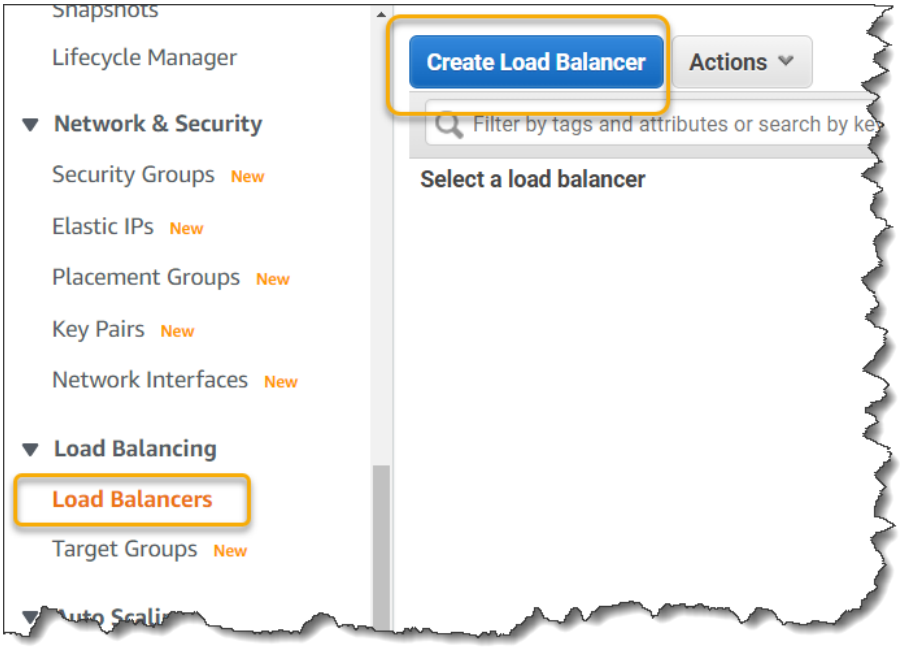
8. Select **Choose an existing key-pair** select the key you created earlier and Click **Launch Instances**



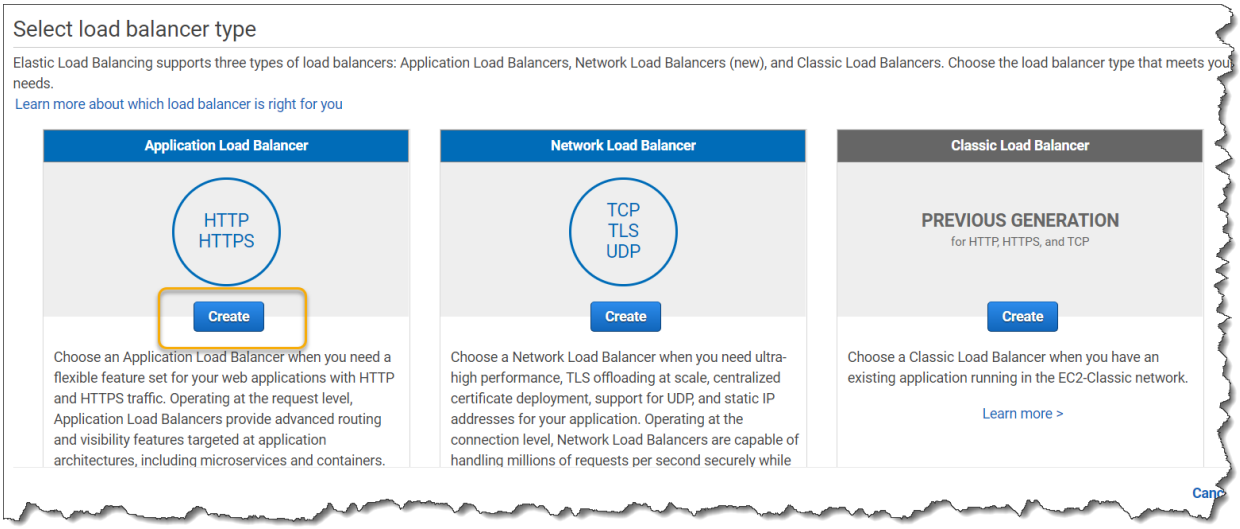
9. Your instance should be launching now.

Task 3: Create a load balancer and add both the instances under it.

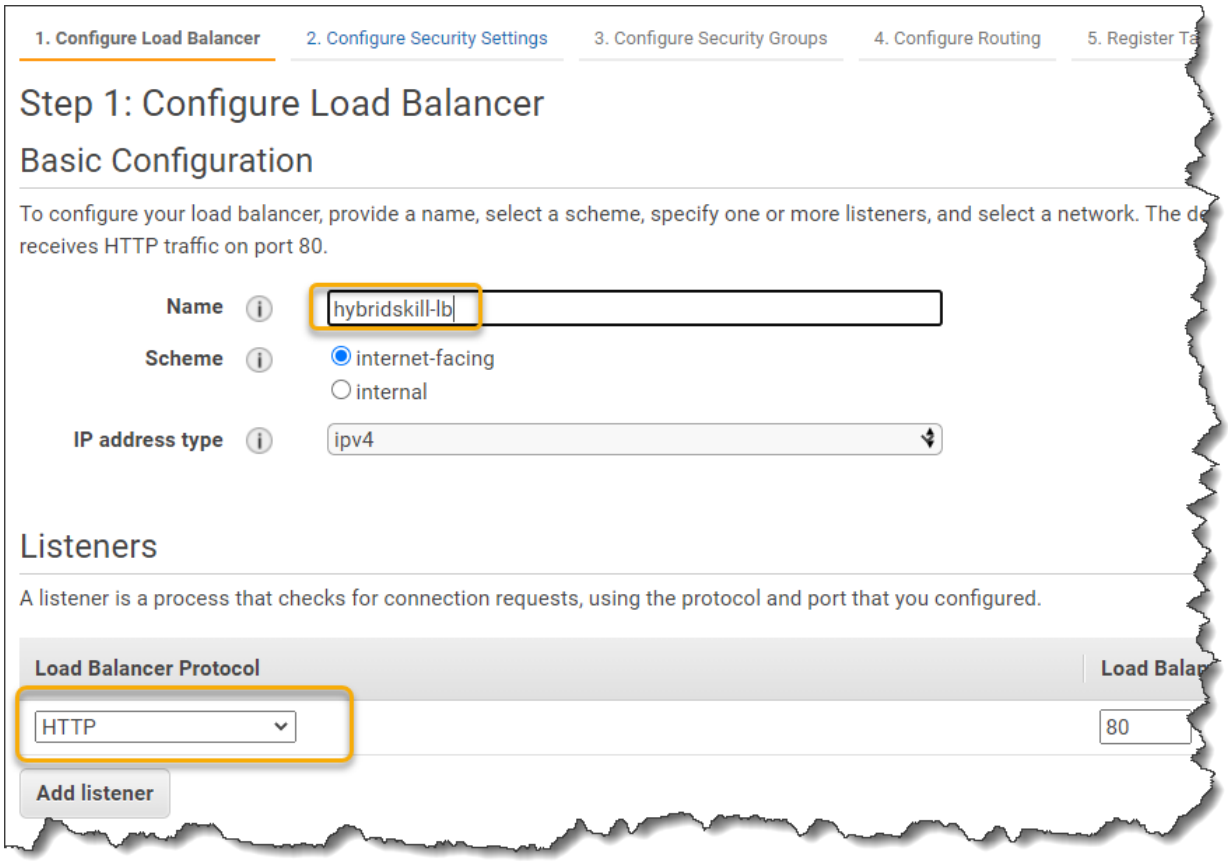
1. On the main EC2 instance dashboard under the **LOAD BALANCING**, click **Load Balancers** and then click **Create Load Balancer**



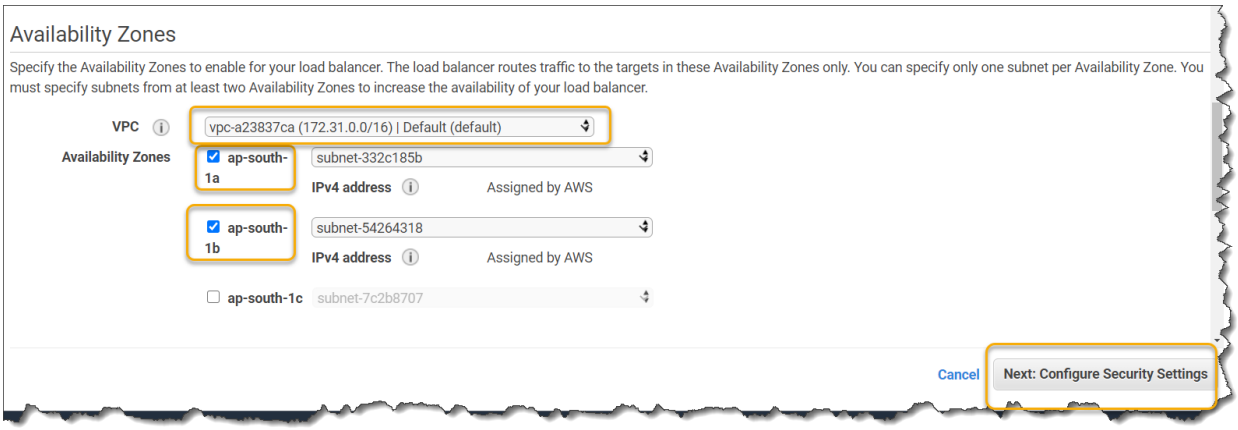
2. Select **Application Load Balancer** as your load balancer type and click **Create.**



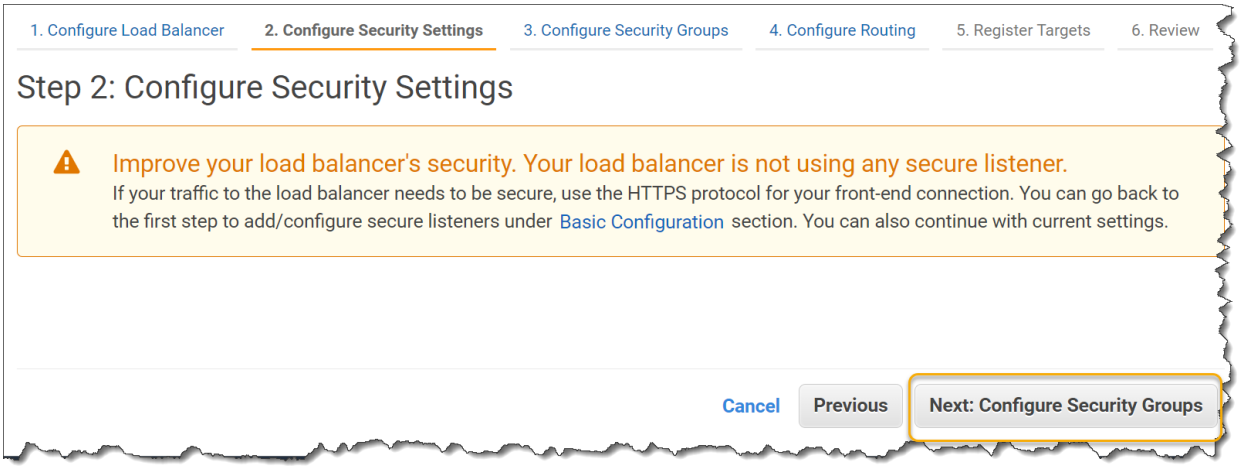
3. Enter **hybridskill-lb** as the **Load Balancer name.** Leave the **Load balancer protocol** as **HTTP**



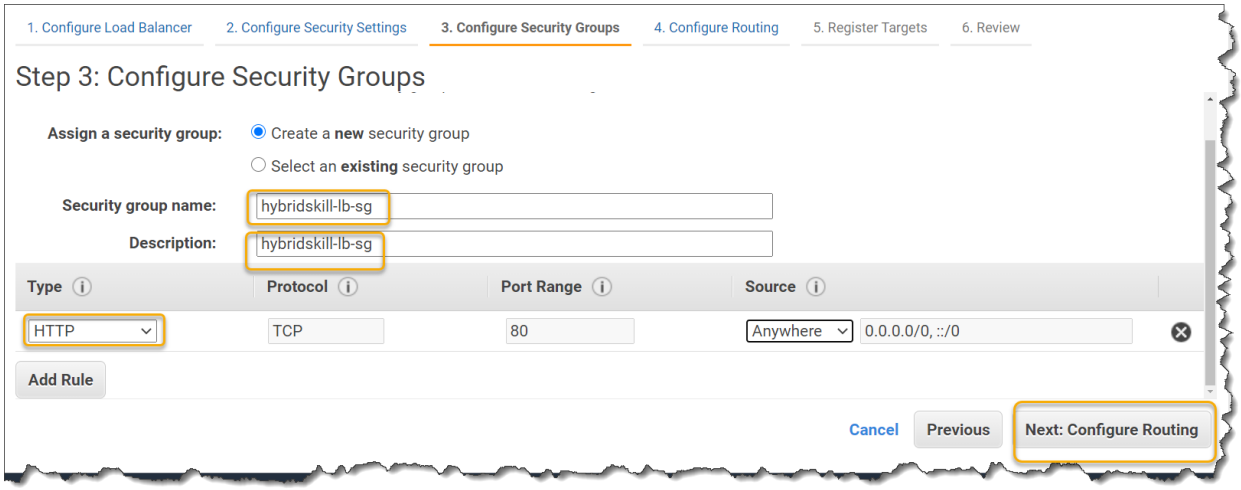
4. Select the **Default VPC** and **Availability zones a** and **b** leave all options as default and **click Next: Configure Security Settings**



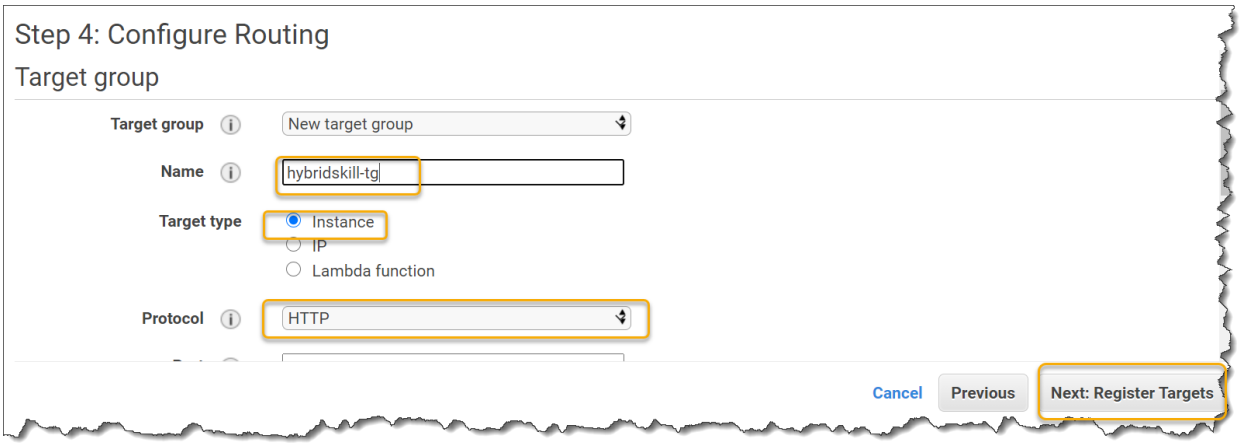
5. Click **Next: Configure Security Groups.**



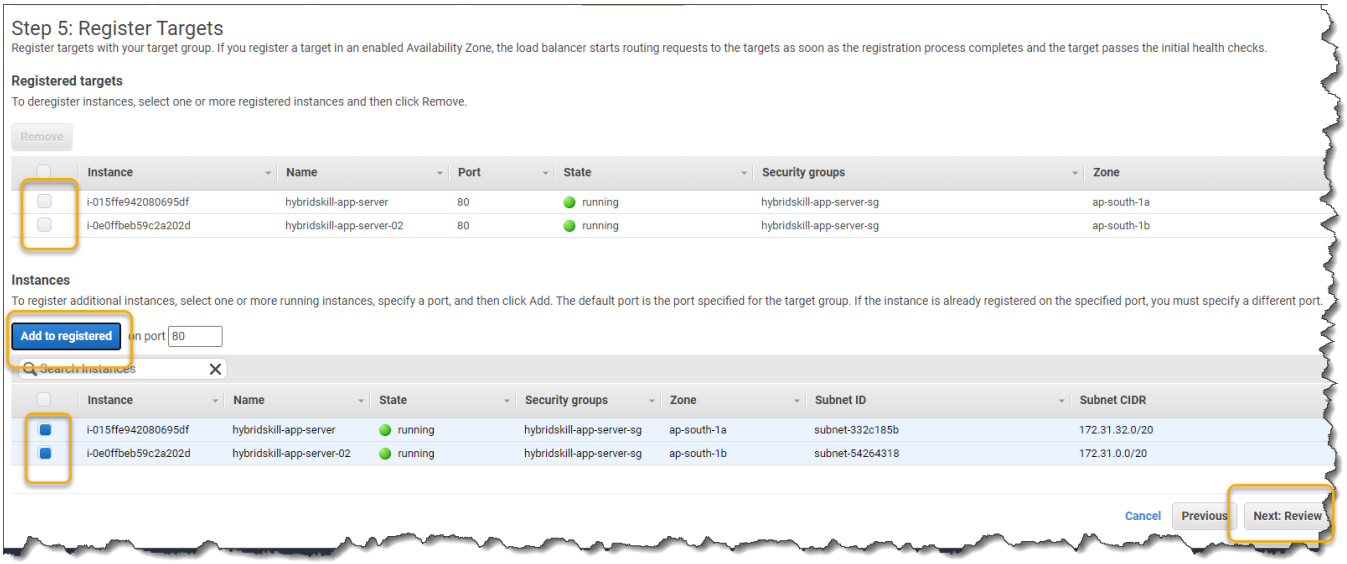
6. Select **Create a new security group.** Enter **hybridskill-lb-sg** as the **Security group name,** enter a short **Description.** Select **HTTP** select the Source as **Anywhere**. And **Select Next: Configure Routing**



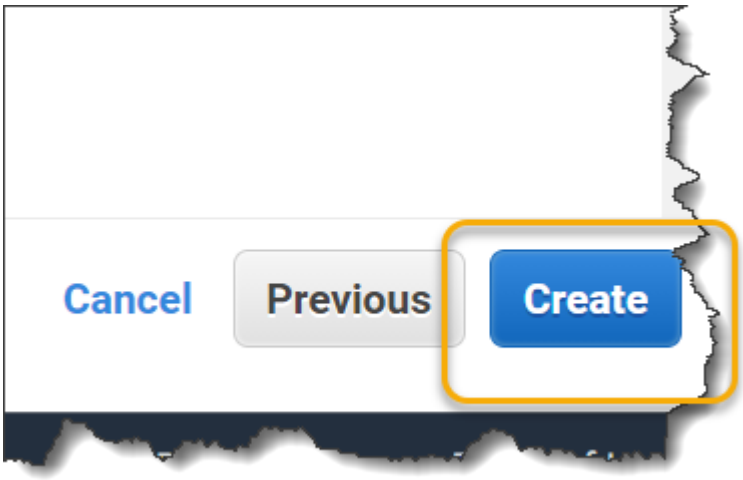
7. Select **New target group,** select the **Target type** as **Instance** and the **Protocol** as **HTTP.** Leave other options as default and click **Next: Register targets** .



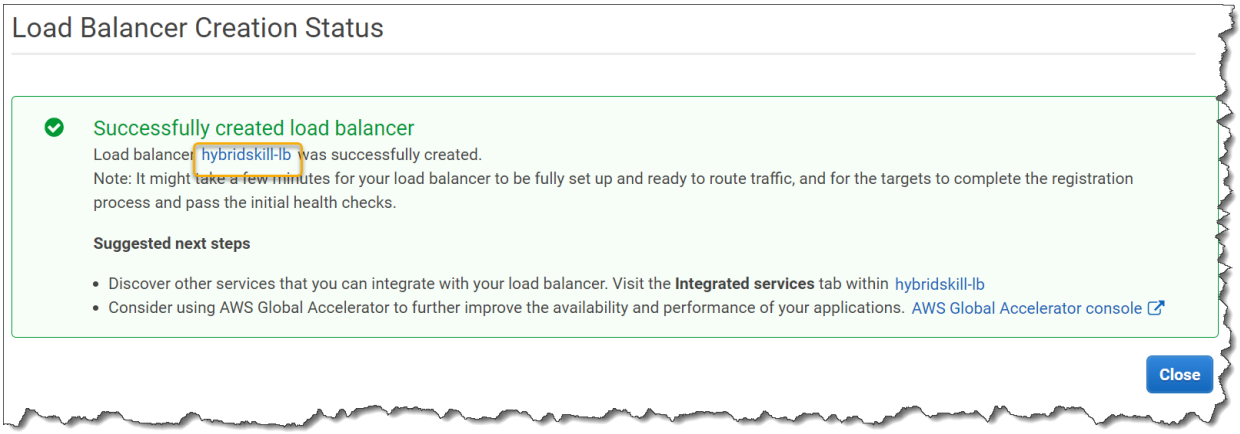
8. **Select** both your EC2 instances you created earlier and click **Add to registered.** Click **Next:Review**



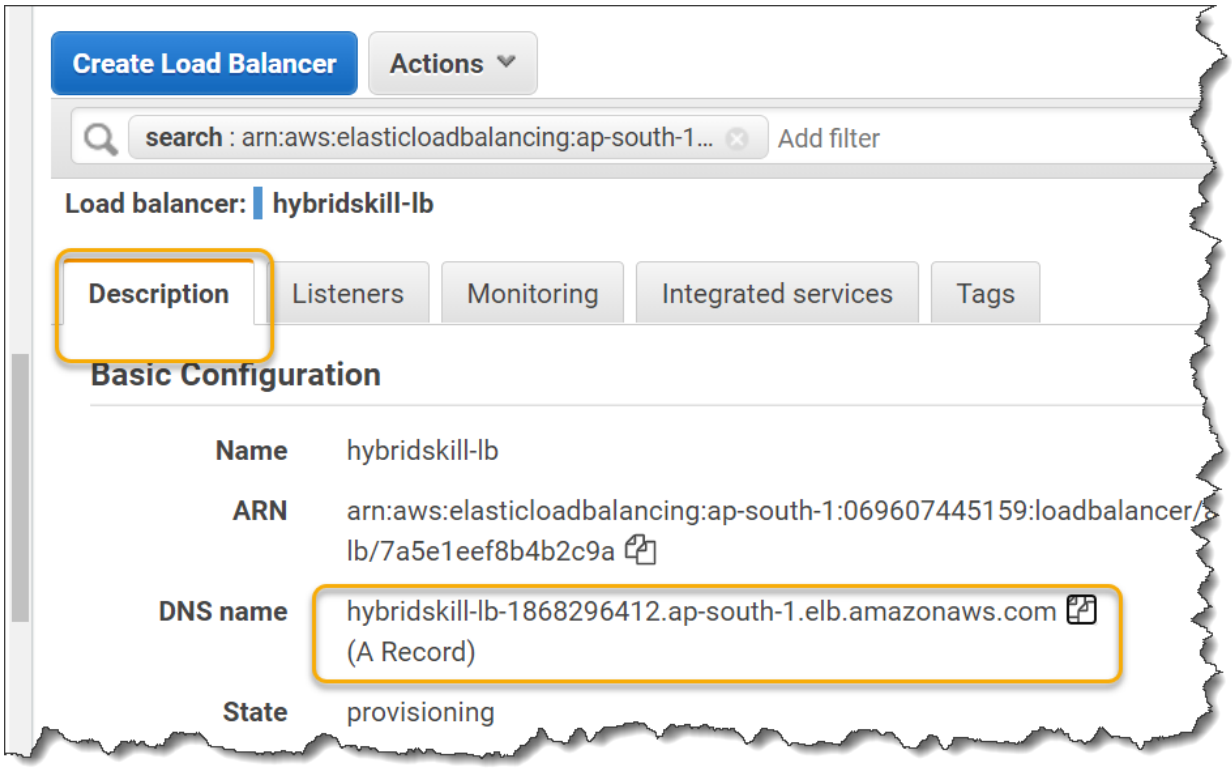
9. Review your settings and click **Create.**



10. Your load balancer has been created. Click **hybridskill-lb** to view more info about your load balancer



11. Under Description copy the DNS name and visit it on a browser



Your wordpress application should be visible from the load balancer DNS

Task 4: Manage ELB using CLI

Now that we have explored ELB through the console, let's do the same through the CLI. Run the following commands on the command line interface that you had setup earlier.

1. Launch box from created AMI without using user-data and Note down the instance ID.

```
aws ec2 run-instances --image-id ami-0e356a61 --count 1 --instance-type t2.micro --
key-name hybridskill-test --security-groups hybridskill-sg-test
```

2. Create a ELB.
    a. Create Subnet for ELB:

```
:~$ aws ec2 create-security-group --group-name wordpressapp-elb-test --description
"created via awscli"
{
    "GroupId": "sg-3bcd6750"
}
```

3. Add ELB ingress rule:

```
:~$ aws ec2 authorize-security-group-ingress --group-name  wordpressapp-elb-test  --
protocol tcp --port 80 --cidr 0.0.0.0/0
```

4. Run below command and get Subnet ID

```
:~$aws ec2 describe-subnets
{
    "Subnets": [
        {
            "AvailabilityZone": "ap-south-1b",
             ..
              ..
            "SubnetId": "subnet-3f312b72",
            "VpcId": "vpc-d6a00fbe",
            ..
        },
        {
            "AvailabilityZone": "ap-south-1a",
             ..
            "SubnetId": "subnet-7e248f16",
            "VpcId": "vpc-d6a00fbe",
            ..
        }
]
}
```

5. Use one of the subnet and create ELB

```
:~$  aws  elb  create-load-balancer  --load-balancer-name  wp-test-elb  --listeners
"Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80"    --subnets
subnet-3f312b72 --security-groups sg-3bcd6750
{
    "DNSName": "wp-test-elb-1143976224.ap-south-1.elb.amazonaws.com"
}
```

6. Now register instance using instance ID under ELB

```
:~$ aws elb register-instances-with-load-balancer  --load-balancer-name wp-test-elb --instances i-04f8608c2e791aa76 i-0ea9e37dd06475610

{
    "Instances": [
        {
            "InstanceId": "i-0ea9e37dd06475610"
        },
        {
            "InstanceId": "i-04f8608c2e791aa76"
        }
    ]
}
```

7. Check Instance status:

```
:~$ aws elb describe-instance-health --load-balancer-name wp-test-elb --instances i-0ea9e37dd06475610

{
    "InstanceStates": [
        {
            "InstanceId": "i-0ea9e37dd06475610",
            "State": "InService",
            "ReasonCode": "N/A",
            "Description": "N/A"
        }
    ]
}
```