

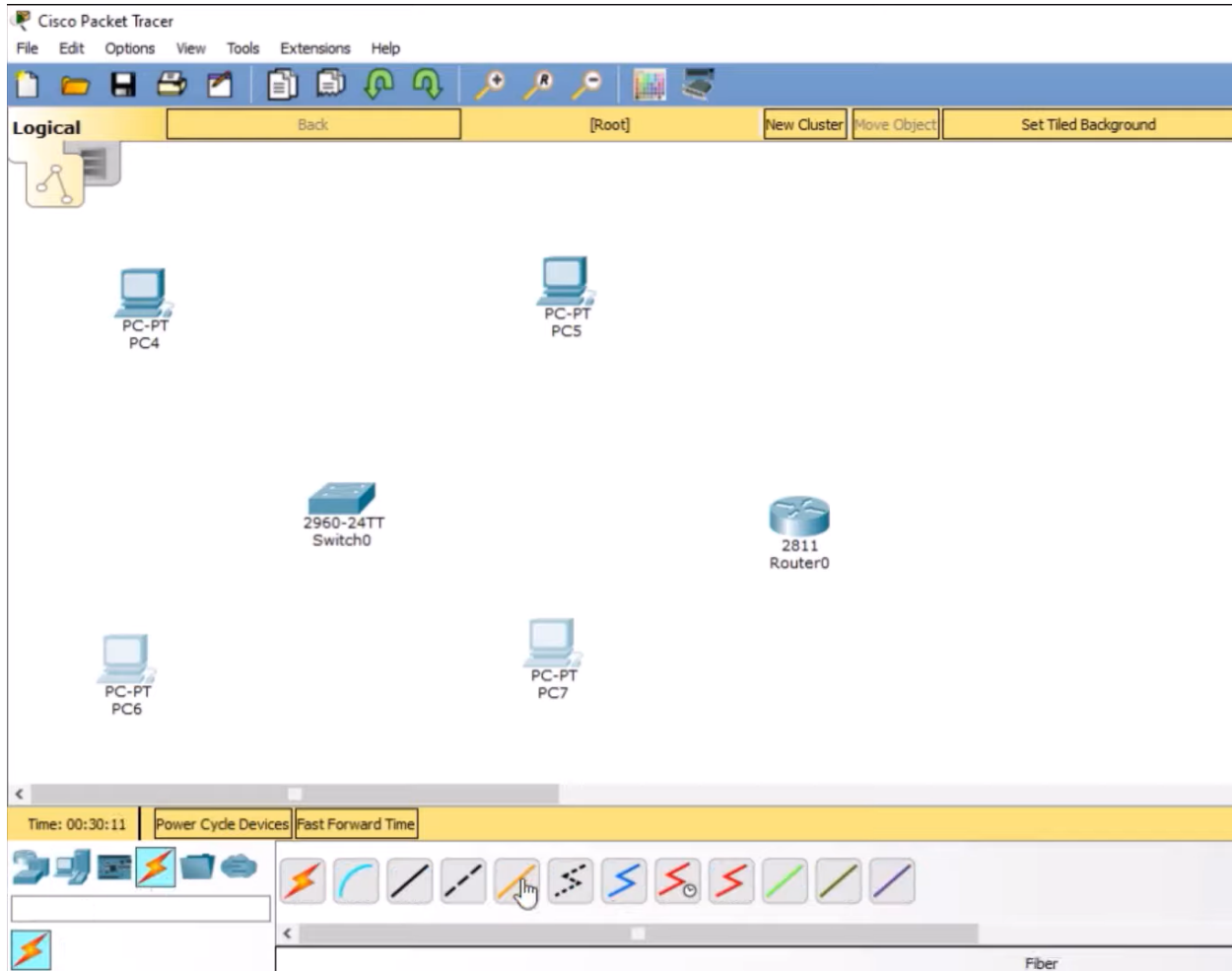
- Cloud computing

# <sup>1</sup> Cisco simulator - VLAN design, Routing, Subnetting, Gateway configuration

ans.

## 1. Add Devices:

- Drag and drop a switch (e.g., 2960) and several PCs onto the workspace.



## 2. Create VLANs:

---

- Enter the following commands to create VLANs:

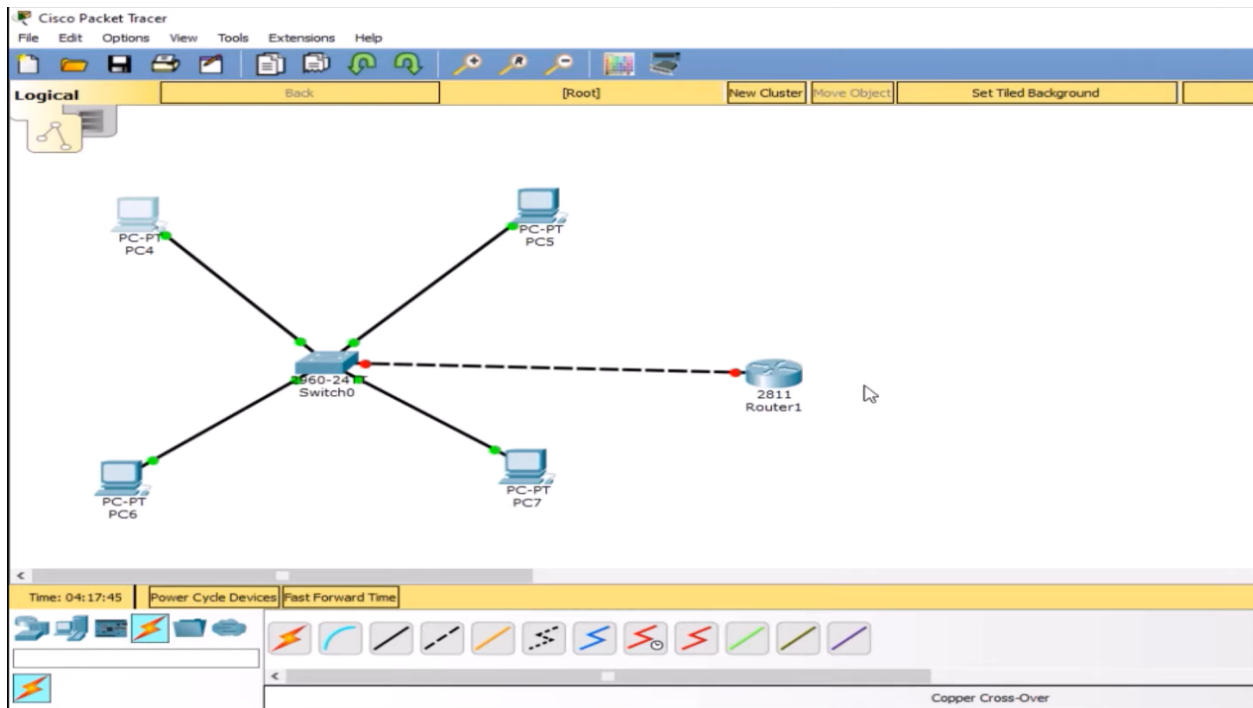
```
#enable
configure terminal
vlan 10
name VLAN10
exit
vlan 20
name VLAN20
exit#
```

### 3. Assign Ports to VLANs:

- Assign specific ports to each VLAN.

```
#interface range fa0/1 - 2
switchport mode access
switchport access vlan 10
exit
```

```
interface range fa0/3 - 4
switchport mode access
switchport access vlan 20
exit#
```



#### 4. Configure Router Interfaces:

- Access the router's CLI and configure sub-interfaces for each VLAN:

```
enable
configure terminal
interface gig0/0.10
encapsulation dot1Q 10
ip address 192.168.10.1 255.255.255.0
exit
```

```
interface gig0/0.20
encapsulation dot1Q 20
ip address 192.168.20.1 255.255.255.0
exit
```

```
interface gig0/0
no shutdown
```

##### 1. Assign IP Addresses:

- For PCs in VLAN 10 (e.g., PC1 and PC2): PC1: IP Address: 192.168.10.2  
Subnet Mask: 255.255.255.0
- Default Gateway: 192.168.10.1

##### 2. For PCs in VLAN 20 (e.g., PC3 and PC4): PC3: IP Address: 192.168.10.3

- Subnet Mask: 255.255.255.0
- Default Gateway: 192.168.10.1

```
C:\>exit

C:\>ping 192.168.2.20

Pinging 192.168.2.20 with 32 bytes of data:

Reply from 192.168.2.20: bytes=32 time<1ms TTL=127
Reply from 192.168.2.20: bytes=32 time<1ms TTL=127
Reply from 192.168.2.20: bytes=32 time<1ms TTL=127
Reply from 192.168.2.20: bytes=32 time=1ms TTL=127

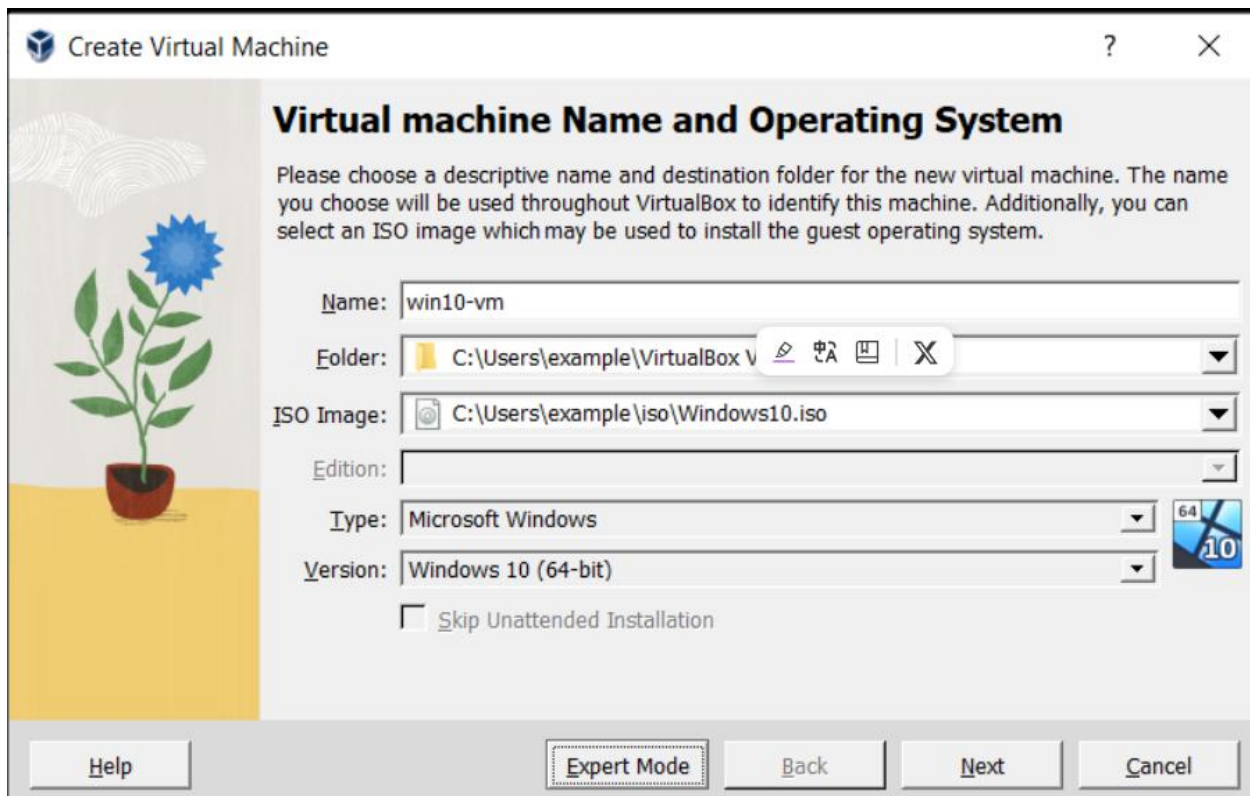
Ping statistics for 192.168.2.20:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>
```

2. Virtual box-based Web Server creation, Images/Snapshots access webpage from 2nd VM on another subnet work

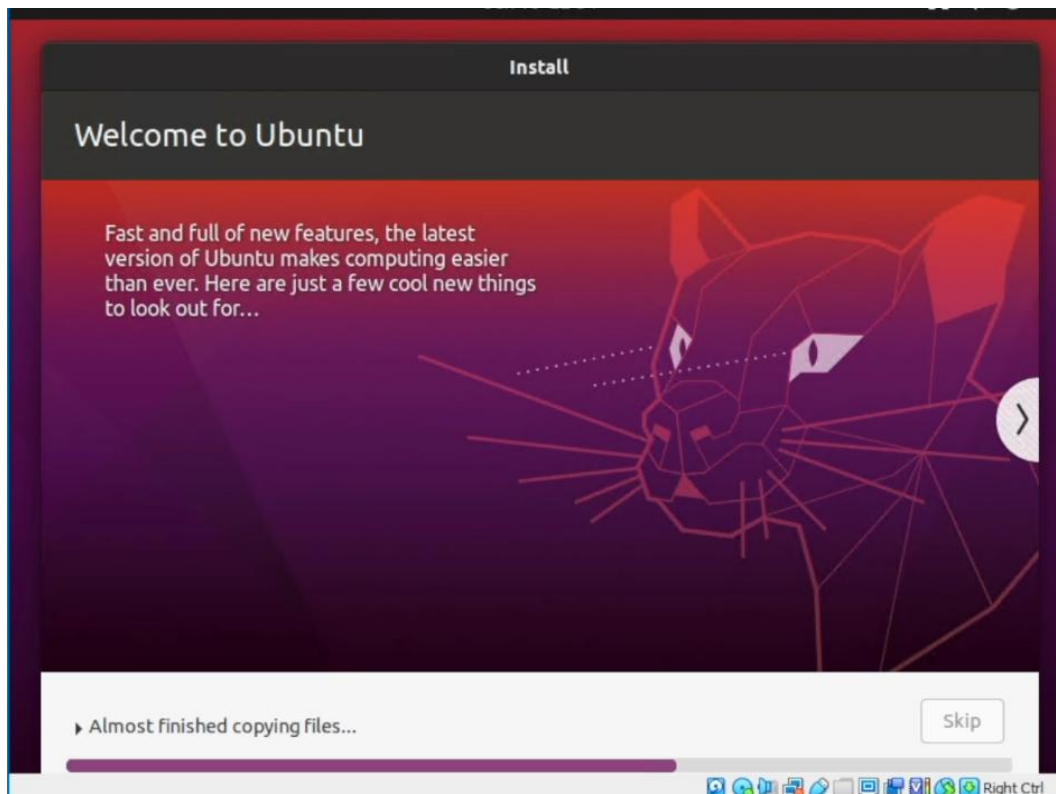
Ans.

## Create a New Virtual Machine



1. Select your new VM, then click on Settings.
2. Go to the Storage tab:
  - Click on the empty CD/DVD icon under Storage Devices.
  - On the right side, click on the CD icon and select "Choose a disk file".
  - Browse to your downloaded Ubuntu Server ISO and select it.

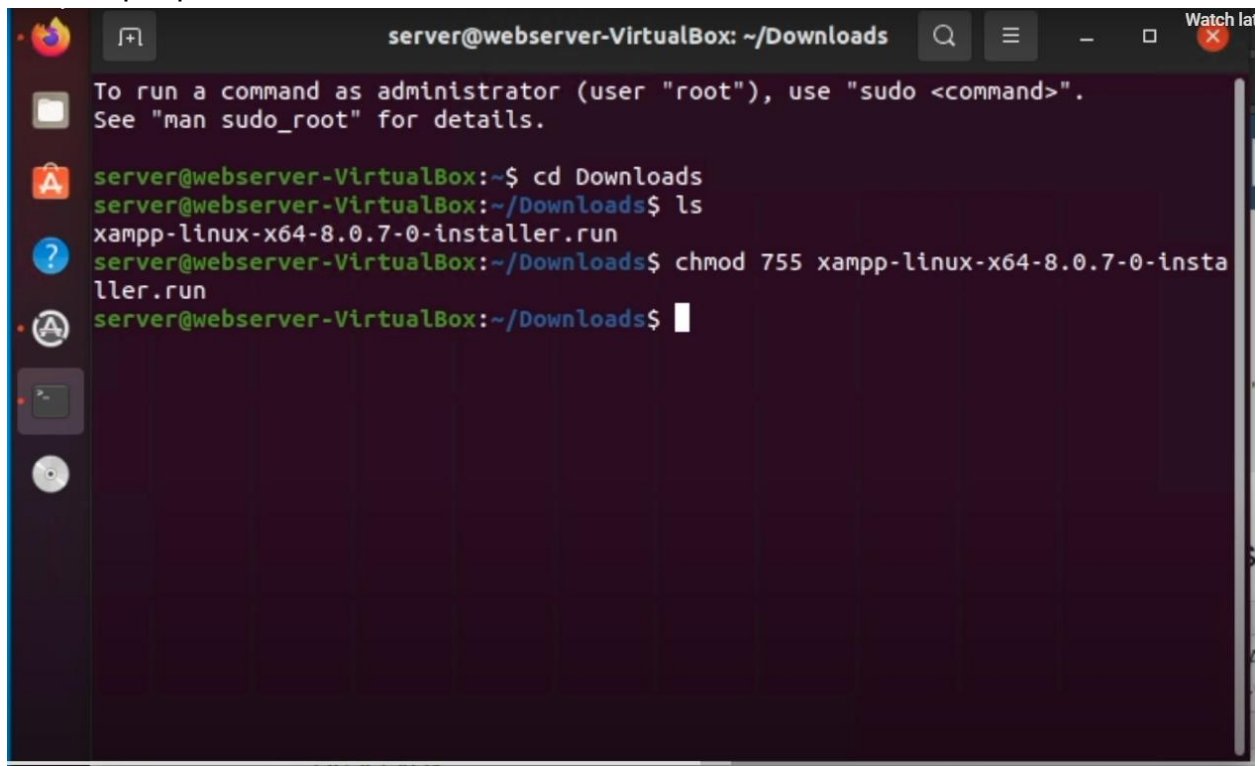
### Install Ubuntu Server



### Access Your Web Server

1. After installation, reboot your VM.
2. Log in using your credentials.
3. Update your package list:

```
#sudo apt update
```

A terminal window titled 'server@webserver-VirtualBox: ~/Downloads' with standard window controls. The terminal shows a message about running commands as administrator, followed by a series of commands: 'cd Downloads', 'ls' (listing 'xampp-linux-x64-8.0.7-0-installer.run'), and 'chmod 755 xampp-linux-x64-8.0.7-0-installer.run'. The prompt is currently at the end of the last command.

```
server@webserver-VirtualBox: ~/Downloads
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

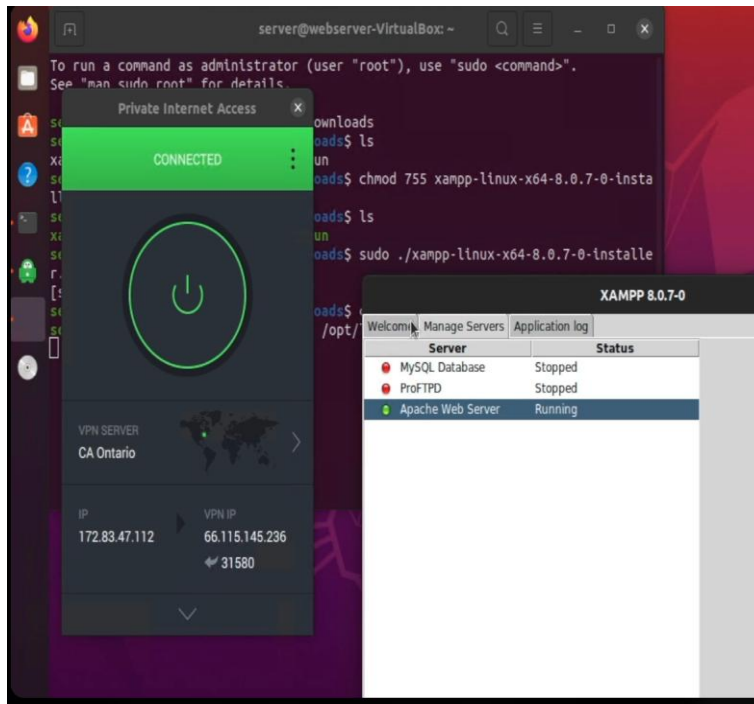
server@webserver-VirtualBox:~$ cd Downloads
server@webserver-VirtualBox:~/Downloads$ ls
xampp-linux-x64-8.0.7-0-installer.run
server@webserver-VirtualBox:~/Downloads$ chmod 755 xampp-linux-x64-8.0.7-0-installer.run
server@webserver-VirtualBox:~/Downloads$
```

Install a web server package (e.g., Apache):

```
#sudo apt install apache2
```

```
#sudo systemctl start apache2
```

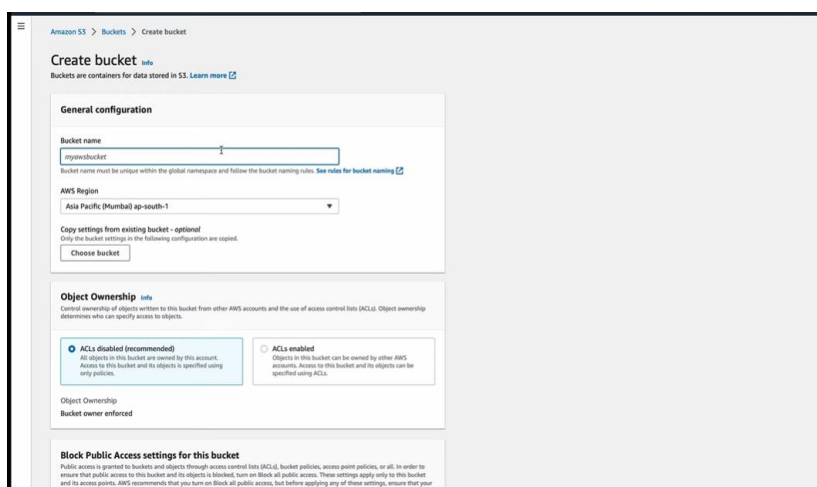
```
#sudo systemctl enable apache2
```



## 3.EC2 AWS-S3 bucket based static web pages

Create a Bucket:

- Click on Create bucket.
- Enter a unique bucket name
- Choose a region close to your target audience to minimize latency.
- Keep the default settings for Block Public Access unless you need to allow public access for your website.
- Click Create bucket to finalize.



Configure Bucket Policy

## Set Permissions:

- Go to the Permissions tab.
- Click on Bucket Policy and add a policy that allows public access:

"Version": "2012-10-17",

"Statement": [

{

"Sid": "PublicReadGetObject",

"Effect": "Allow",

"Principal": "\*",

"Action": "s3:GetObject",

"Resource": "arn:aws:s3:::example.com/\*"

}

## Upload Files:

```

1  <!DOCTYPE html>
2  <html>
3
4  <head>
5    <title>WELCOME PAGE</title>
6    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
7    <style>
8      .slider {
9        width: 800px;
10       height: 450px;
11       overflow: hidden;
12     }
13
14     .slider img {
15       width: 100%;
16       height: 100%;
17     }
18
19     .welcome-message {
20       text-align: center;
21       font-size: 24px;
22       margin: 20px;
23     }
24   </style>
25   <script>
26     var messages = [
27       "Welcome to my channel IY Studio",
28       "Please like share and subscribe",
29       "Grateful forever A"
30     ];
31     function displayRandomMessage() {
32       var message = messages[Math.floor(Math.random() * messages.length)];
33       alert(message);
34     }
35   </script>
36 </head>
37
38 <body>
39   <div class="container">
40     <div class="slider">
41       
42       
43       
44     </div>
45     <h1 class="welcome-message">WELCOME TO MY WEBSITE</h1>
46     <button class="btn btn-primary" onclick="displayRandomMessage()">SHOW RANDOM MESSAGE</button>
47   </div>
48
49   <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
50   <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
51   <script>
52     var sliderImages = document.querySelectorAll('.slider img');
53     var currentIndex = 0;
54
55     function showNextImage() {
56       sliderImages[currentIndex].style.display = 'none';

```



- After uploading, your static website can be accessed using the endpoint provided in the static website hosting section (<http://example.com.s3-website-us-east-1.amazonaws.com>).

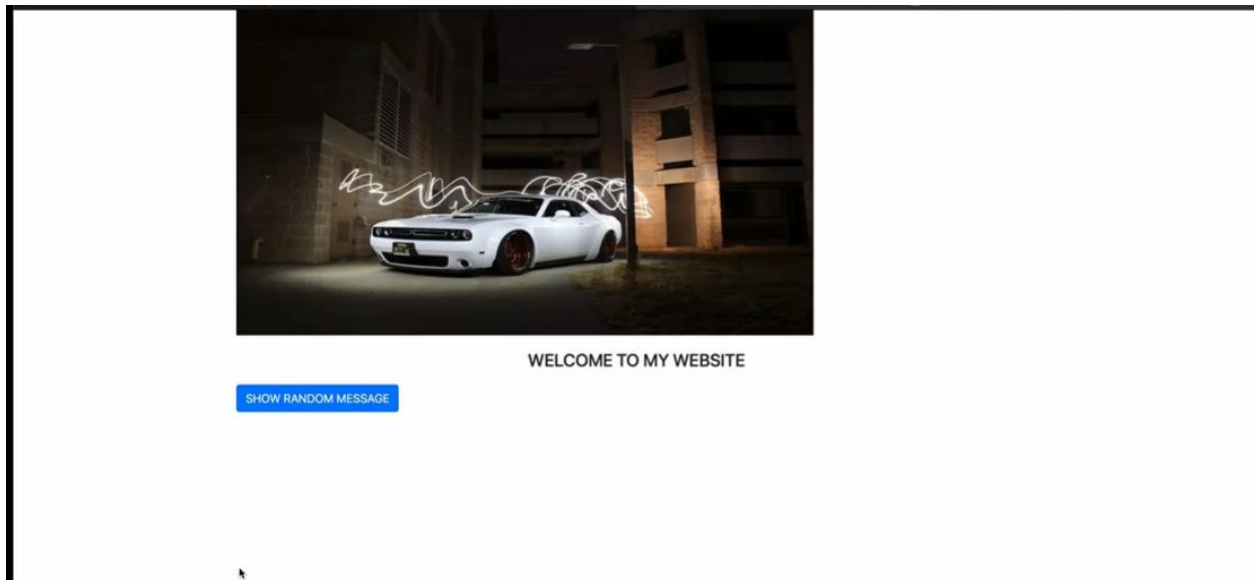
## Set Up an EC2 Instance

If you need server-side processing or other functionalities not supported by S3 alone:

1. Launch an EC2 Instance:
  - Navigate to EC2 in the AWS Management Console.
  - Click on Launch Instance, choose an Amazon Machine Image (AMI), select instance type, configure security groups, and launch
2. Install a Web Server  
If you need dynamic content or server capabilities:

```
sudo apt update
```

```
sudo apt install apache2
```



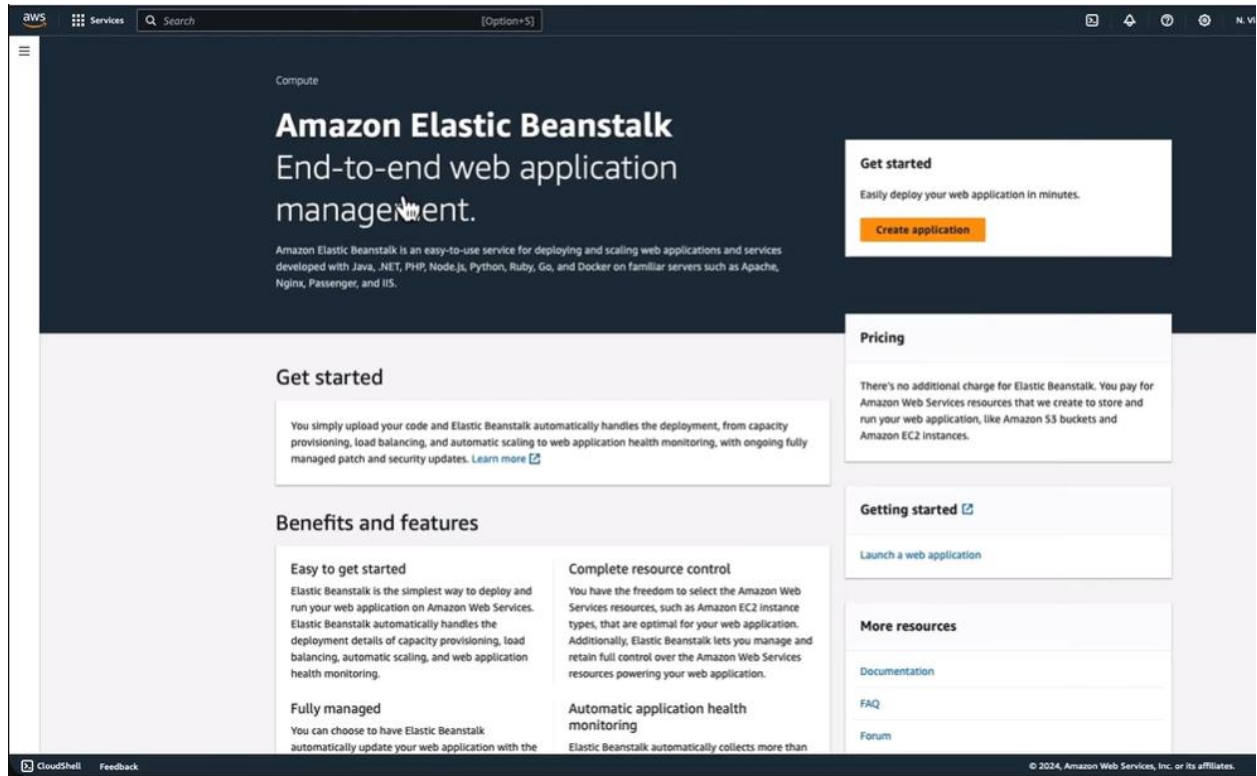
## 4. EC2 AWS - Web application using Beanstalk

### Configure Environment

1. Choose Environment Type:
  - Select Web server environment as the environment tier.
- 2.
3. Select Platform:
  - Choose the platform for your application (e.g., Node.js, Python, Java, etc.).
- 4.

## 5. Configure Service Role:

- Choose an existing service role or create a new one if necessary.
- If creating a new role, navigate to the IAM console to set permissions like `AWSElasticBeanstalkWebTier`, `AWSElasticBeanstalkWorkerTier`, and others as needed [12](#).



## Configure Application Settings

**Configure service access** [Info](#)

**Service access**  
IAM roles, assumed by Elastic Beanstalk as a service role, and EC2 instance profiles allow Elastic Beanstalk to create and manage your environment. Both the IAM role and instance profile must be attached to IAM managed policies that contain the required permissions. [Learn more](#)

**Service role**  
☒ Create and use new service role  
☐ Use an existing service role

**Service role name**  
Enter the name for an IAM role that Elastic Beanstalk will create to assume as a service role. Beanstalk will attach the required managed policies to it.

[View permission details](#)

**EC2 key pair**  
Select an EC2 key pair to securely log in to your EC2 instances. [Learn more](#)

**EC2 instance profile**  
Choose an IAM instance profile with managed policies that allow your EC2 instances to perform required operations.

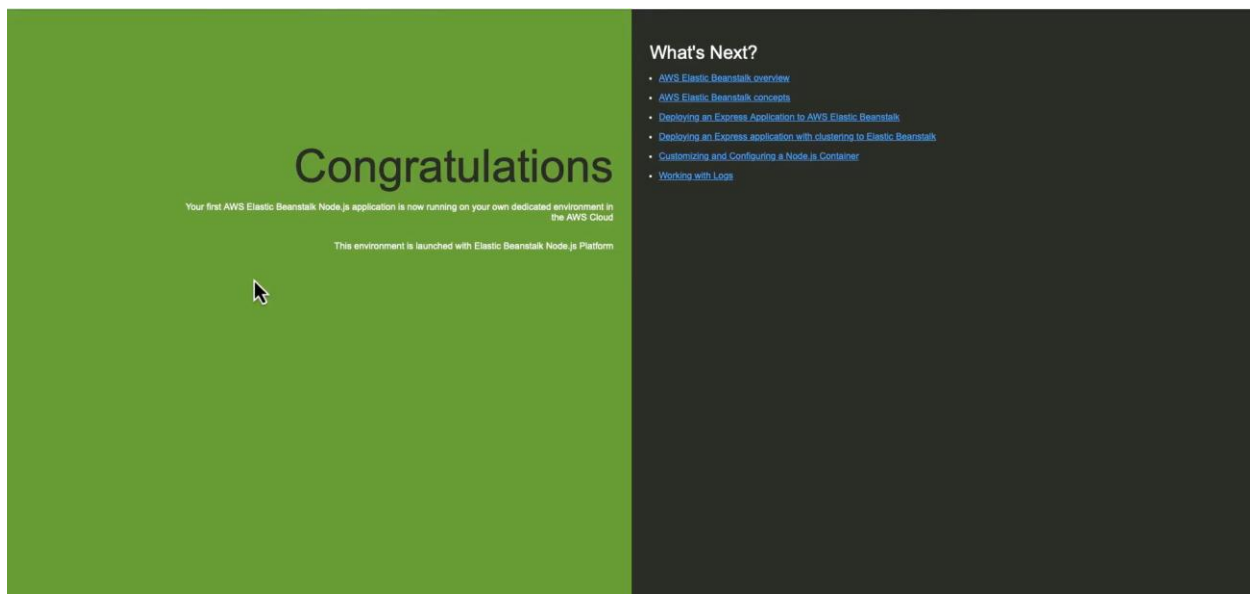
[View permission details](#)

Create Environment:

- Click on Create Environment.
- AWS will begin provisioning resources, including EC2 instances, security groups, and S3 buckets for storage

The screenshot shows the AWS Elastic Beanstalk console during the 'Configure updates, monitoring, and logging' step. On the left, a sidebar lists the steps: Step 1 (Configure environment), Step 2 (Configure service access), Step 3 (optional: Set up networking, database, and tags), Step 4 (optional: Configure instance traffic and scaling), Step 5 (optional: Configure updates, monitoring, and logging), and Step 6 (Review). The main content area is titled 'Configure updates, monitoring, and logging - optional' and includes an 'Info' link. It features a 'Monitoring' section with a 'Health reporting' subsection. Under 'Health reporting', there is a 'System' section with two radio buttons: 'Basic' (selected) and 'Enhanced'. Below this is a 'Health event streaming to CloudWatch Logs' section with a description and a link to 'Amazon CloudWatch Pricing'. Further down is a 'Log streaming' section with a checkbox labeled 'Activated (standard CloudWatch charges apply.)'. Below that is a 'Retention' section with a dropdown menu set to '7'. Finally, there is a 'Lifecycle' section with a dropdown menu set to 'Keep logs after terminating environment'.

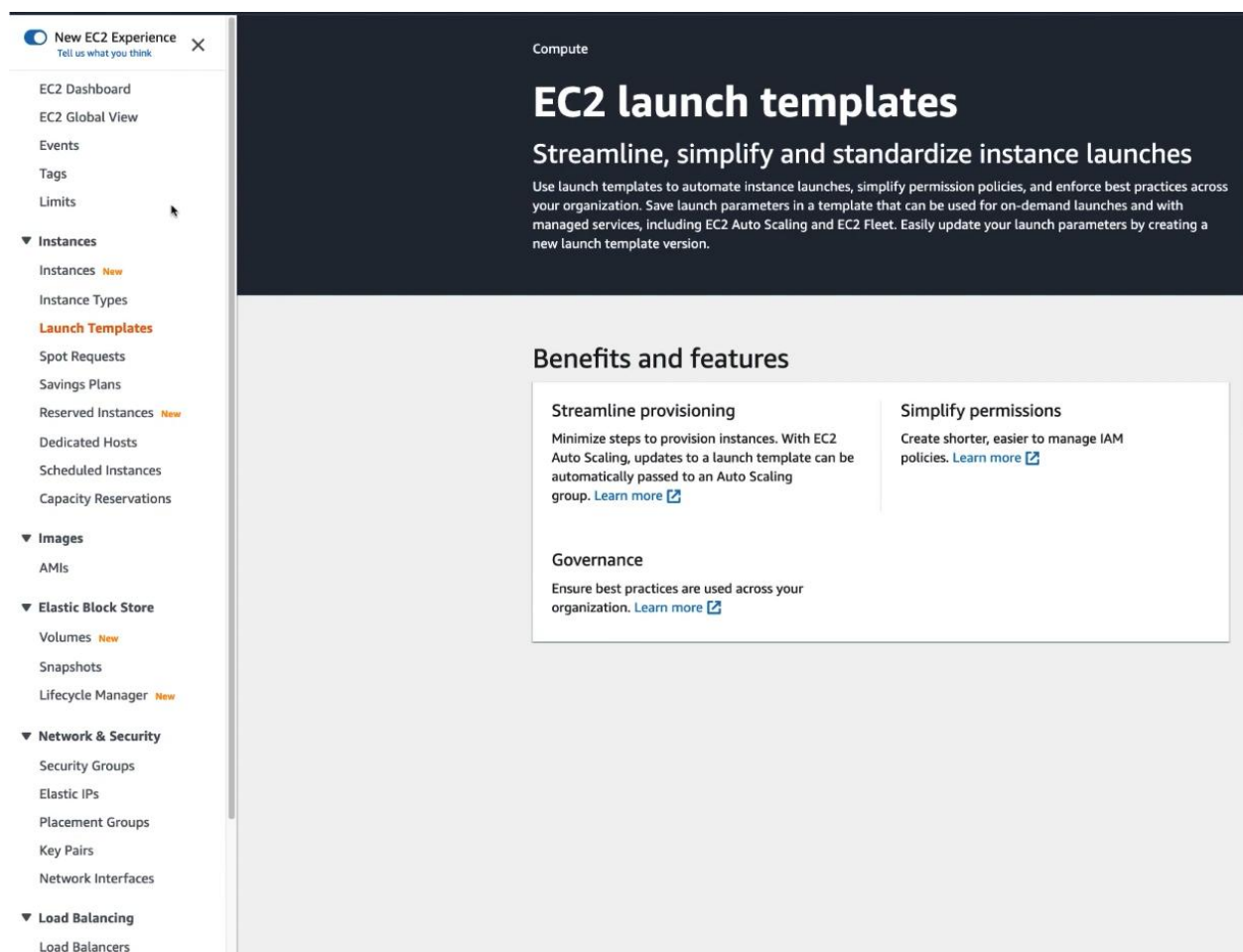
## Deploy Your Application



## 5. AWS Local balancing and auto scaling

## Create a Launch Template

- Click on Launch Templates in the left-hand menu.
- Click on Create launch template.
- Fill in the required details:
  - Name: Provide a name for your template (e.g., `my-launch-template`).
  - Amazon Machine Image (AMI): Select an appropriate AMI (e.g., Amazon Linux 2).
  - Instance Type: Choose an instance type (e.g., `t2.micro` for free tier).
  - Key Pair: Select or create a key pair for SSH access.
  - Security Group: Define security group settings to control inbound/outbound traffic.



## Create an Auto Scaling Group

Navigate to Auto Scaling Groups:

- In the EC2 Dashboard, click on Auto Scaling Groups in the left-hand menu.

Create Auto Scaling Group:

- Click on the Create Auto Scaling group button.
- Enter a name for your Auto Scaling group (e.g., `my-auto-scaling-group`).
- Select the launch template you created earlier.

Configure Network Settings:

- Choose the VPC and subnets where instances will be launched. Select multiple subnets for high availability.
- Click on Next to proceed.

## Set Scaling Policies

Define Capacity Settings:

- Set minimum, maximum, and desired capacity for your instances.
- For example: Minimum: 1

Configure Scaling Policies:

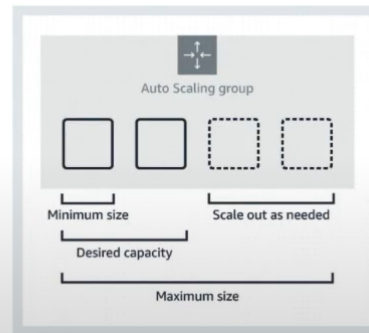
- You can choose between different scaling policies: Target Tracking Scaling
- Set up notifications if desired to alert you of scaling events.

# Amazon EC2 Auto Scaling

## helps maintain the availability of your applications

Auto Scaling groups are collections of Amazon EC2 instances that enable automatic scaling and fleet management features. These features help you maintain the health and availability of your applications.

### How it works



An Auto Scaling group is a collection of Amazon EC2 instances that are treated as a logical unit. You configure settings for a group and its instances as well as define the group's minimum, maximum, and desired capacity. Setting different minimum and maximum capacity values forms the bounds of the group, which allows the group to scale as the load on your application spikes higher or lower, based on demand. To scale the Auto Scaling group, you can either make manual adjustments to the desired capacity or let Amazon EC2 Auto Scaling automatically add and remove capacity to meet changes in demand.

When launching fleets of instances, you can specify what percentage of your capacity should be fulfilled by On-Demand instances, and what percentage with Spot Instances, to save up to 90% on EC2 costs. Amazon EC2 Auto Scaling lets you provision and balance capacity across Availability Zones to optimize availability. It also provides lifecycle hooks, instance health checks, and scheduled scaling to automate capacity

## Review

my-app, 1 Scaling policy created successfully

EC2 > Auto Scaling groups

### Auto Scaling groups (1/1)

Search your Auto Scaling groups

<input checked="" type="checkbox"/>	Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max
<input checked="" type="checkbox"/>	my-app	my-web-app   Version Latest	0	Updating capacity	2	2	10

<input type="checkbox"/>	Instance ID	Lifecycle	Instance type	Weighted capacity	Launch template/configuration	Availability Zone
<input type="checkbox"/>	i-0377408e4a8af2895	Pending	t2.micro	-	my-web-app   Version 1	us-west-2c
<input type="checkbox"/>	i-0c88f755181246935	Pending	t2.micro	-	my-web-app   Version 1	us-west-2a

### Lifecycle hooks (0)

Filter lifecycle hooks

<input type="checkbox"/>	Name	Lifecycle transition	Default result	Heartbeat timeout (seconds)	Notification ta
No lifecycle hooks are currently configured. Lifecycle hooks help you perform custom actions on instances as they launch and before they terminate.					

Create lifecycle hook

## Configure Load Balancer

## Attach Load Balancer:

- During the Auto Scaling group setup, you can optionally create or attach an Elastic Load Balancer (ELB).
- If creating a new load balancer, follow the prompts to configure it, ensuring it distributes traffic across your EC2 instances effectively.

search : my-app-lb Add filter

Name	DNS name	State	VPC ID	Availability Zones	Type
my-app-lb	my-app-lb-1993936138.us-w...	Active	vpc-d9abefa1	us-west-2c, us-west-2a...	application

Load balancer: my-app-lb

Description Listeners Monitoring Integrated services Tags

**Basic Configuration**

<b>Name</b>	my-app-lb
<b>ARN</b>	arn:aws:elasticloadbalancing:us-west-2:978762696487:loadbalancer/app/my-app-lb/befe4d07d6075fed
<b>DNS name</b>	my-app-lb-1993936138.us-west-2.elb.amazonaws.com (A Record)
<b>State</b>	Active
<b>Type</b>	application
<b>Scheme</b>	internet-facing
<b>IP address type</b>	ipv4

Edit IP address type

## Test Load Balancing (if configured)

