1. Create one vpc in N.virginia region.

Go to <https://console.aws.amazon.com/vpc/>

Ensure the region at the top-right is **N. Virginia (us-east-1)**.

Click **"Create VPC"**.

Select **"VPC only"**.

Provide:

* 1. **Name tag**: MyVPC
  2. **IPv4 CIDR block**: 10.0.0.0/16

Click **Create VPC**.

A screenshot of a computer

AI-generated content may be incorrect.

1. Create two subnets. One Public subnet and one private subnet

Go to **VPC Dashboard** → **Subnets** → **Create Subnet**.

Select your **VPC** (e.g., MyVPC).

Create two subnets:

**Public Subnet**

* **Name tag**: PublicSubnet
* **Availability Zone**: us-east-1a (or any AZ you prefer)
* **IPv4 CIDR Block**: 10.0.0.0/25

**Private Subnet**

* **Name tag**: PrivateSubnet
* **Availability Zone**: us-east-1a (or a different AZ for redundancy)
* **IPv4 CIDR Block**: 10.0.0.128/25

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Provide the IGW to the vpc.

Go to **VPC Dashboard** → **Internet Gateways**

Click **“Create internet gateway”**

* **Name**: MyIGW000

Click **Create**

After creation, click **Actions → Attach to VPC**

* Select your VPC

Done

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Create One public RT and one private RT.

**Public Route Table**

Go to **VPC → Route Tables → Create Route Table**

* + Name: PublicRT
  + VPC: Select your VPC

After creating, select the route table → **Routes → Edit routes**

* + Add Route:
    - Destination: 0.0.0.0/0
    - Target: **Internet Gateway** (select your IGW)

Go to **Subnet Associations**

* + Associate this RT with your **Public Subnet** (10.0.0.0/25)

**Private Route Table**

Repeat the above to create a second route table:

* + Name: PrivateRT
  + VPC: same as before

Don’t add any routes yet (no internet access by default)

-Associate this route table with your **Private Subnet** (10.0.0.128/25)

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Deploy NAT gateway on public subnet and attach the NAT gatewat to private subnet.

**Step 1: Create Elastic IP**

* Go to **EC2 Dashboard → Elastic IPs → Allocate Elastic IP**
* Click **Allocate**
* Note the EIP for the next step

**Step 2: Create NAT Gateway**

* Go to **VPC → NAT Gateways → Create NAT Gateway**
* **Subnet**: Choose the **Public Subnet** (10.0.0.0/25)
* **Elastic IP**: Choose the one you just allocated
* **Name**: MyNATGateway
* Click **Create**

⚠️ Wait for the NAT Gateway to reach available status.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 3: Update Private Route Table**

* Go to **VPC → Route Tables**
* Select your **PrivateRT**
* Click **Routes → Edit Routes → Add route**
  + Destination: 0.0.0.0/0
  + Target: **NAT Gateway** → choose MyNATGateway
* Save changes

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Create Two instances,one in public subnet and one in private subnet.

**🔹 Public EC2 Instance**

Go to **EC2 → Launch Instance**

Name: PublicInstance

AMI: **Amazon Linux 2**

Instance type: t2.micro

Key pair: Select MyKeyPair

Network settings:

* + VPC: vpc-xxxxxxxx
  + Subnet: subnet-public
  + Auto-assign Public IP: **Enable**
  + Security Group: allow SSH (port 22) and optionally ICMP (ping)

Launch

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**🔸 Private EC2 Instance**

Repeat steps for the second instance

Name: PrivateInstance

Same AMI, instance type, and key pair

Network settings:

* + Subnet: subnet-private
  + Auto-assign Public IP: **Disable**
  + Security Group: allow SSH **from the public instance's private IP**

Launch

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Deploy Apache server on both the ec2 instances with sample index.html file.

ssh -i <your-key.pem> ec2-user@<Public-IP>

Install Apache Web Server

sudo yum update -y

sudo yum install httpd -y

sudo systemctl enable httpd

sudo systemctl start httpd

**Create a Sample index.html File**

echo "<h1>Hello from Apache on $(hostname)</h1>" | sudo tee /var/www/html/index.html

A screenshot of a computer screen

AI-generated content may be incorrect.

A computer screen shot of text

AI-generated content may be incorrect.

Test from Browser or Public Instance

<http://44.222.223.141/>

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen shot of a black screen

AI-generated content may be incorrect.

1. Create one application load balancer and attach the load balancer to both the ec2 instances.

**Step 1: Go to EC2 → Load Balancers**

* Open the AWS Management Console
* Navigate to **EC2 → Load Balancers**
* Click on **“Create Load Balancer”**

**Step 2: Choose Load Balancer Type**

* Select **"Application Load Balancer"**
* Click **"Create"**

**Step 3: Configure Load Balancer**

* **Name:** my-app-lb
* **Scheme:** Internet-facing
* **IP address type:** IPv4
* **Listeners:** HTTP (Port 80)
* **Availability Zones:** Select your VPC and both subnets (public & private)

✅ Ensure both EC2 subnets are selected.

**Step 4: Configure Security Groups**

* Create or select a **Security Group** that allows:
  + **Inbound HTTP (port 80) from anywhere (0.0.0.0/0)**
  + Optional: Allow your IP for SSH (port 22) to troubleshoot later

**Step 5: Configure Target Group**

* **Name:** my-target-group
* **Target type:** Instance
* **Protocol:** HTTP
* **Port:** 80
* **Health checks:**
  + **Path:** /
  + Leave rest as default

Click **Next: Register Targets**

**Step 6: Register EC2 Targets**

* Select **both** EC2 instances (public and private)
* Click **"Add to registered"**
* Click **"Next"**

**Step 7: Review & Create**

* Review settings
* Click **"Create Load Balancer"**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Store Application load balancer logs to s3.

**🧱 Step 1: Create or Choose an S3 Bucket**

1. Go to the **S3** service in the AWS console.
2. Create a new bucket (e.g., alb-logs-yourname) or use an existing one.
3. Make sure the bucket is in the **same region** as your ALB (e.g., us-east-1)

A screenshot of a computer

AI-generated content may be incorrect.

**Step 2: Add Bucket Policy to Allow ALB Logging**

You must add a policy to your bucket to **allow ALB to write logs**.

Go to **S3 > Your Bucket > Permissions > Bucket policy**

{

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

"Statement": [

{

"Sid": "AWSALBLoggingPermissions",

"Effect": "Allow",

"Principal": {

"Service": "logdelivery.elasticloadbalancing.amazonaws.com"

},

"Action": "s3:PutObject",

"Resource": "arn:aws:s3:::alb-logs-nithin/AWSLogs/743316906818/\*"

}

]

}

A screenshot of a computer

AI-generated content may be incorrect.

**Step 3: Enable Access Logging on the ALB**

Go to **EC2 > Load Balancers**.

Select your **Application Load Balancer**.

Click on the **Attributes** tab.

Under **Access logs**, click **Edit**.

Set:

* + **Enable access logs**: ✅ Enabled
  + **S3 location**: alb-logs-yourname
  + **Prefix (optional)**: e.g., logs/

Save changes.

A screenshot of a computer

AI-generated content may be incorrect.

Confirm Logs Appear

A screenshot of a computer

AI-generated content may be incorrect.

10) Store the vpc flow logs to cloudwtach group.

**Create IAM Role for VPC Flow Logs**

VPC Flow Logs need permission to publish to CloudWatch.

Go to **IAM > Roles**.

Click **Create role**.

Select **EC2** as the trusted entity.

Attach the **CloudWatchLogsFullAccess** policy

A screenshot of a computer

AI-generated content may be incorrect.

**Create the VPC Flow Log**

Go to **VPC > Your VPCs**.

Select your VPC (e.g., vpc-xxxxxxx).

Click **Actions > Create flow log**.

Set the following:

* + **Filter**: All, Reject, or Accept (choose based on what traffic you want to log).
  + **Destination**: Send to CloudWatch Logs.
  + **Log group**: Choose your log group (vpc-flow-logs-group) or create a new one.
  + **IAM Role**: Select VPCFlowLogsRole.

Click **Create flow log**.

A screenshot of a computer

AI-generated content may be incorrect.

11)Create Monitoring Dashboards to monitor cpu utilization and to monitor apache service.

**Create a CloudWatch Dashboard for CPU Utilization**

**Step-by-Step:**

**Go to AWS Console** → **CloudWatch** → **Dashboards**.

Click **Create dashboard**.

Enter a name like: EC2-Monitoring-Dashboard → Click **Create dashboard**.

Choose **"Line" widget** → Click **Next**.

Under **"Metrics"**:

* + Navigate to:  
    EC2 > Per-Instance Metrics  
    Choose the **InstanceId** of the EC2 you want to monitor.
  + Select the **CPUUtilization** metric.

Click **Create widget**.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A white background with black and blue text

AI-generated content may be incorrect.

Monitor Apache Service via Custom Metric

Write the bash script

#!/bin/bash

# Check if httpd process is running using ps -ef

if ps -ef | grep -v grep | grep httpd > /dev/null

then

STATUS=1

else

STATUS=0

fi

# AWS region where CloudWatch metrics will be pushed

REGION="us-east-1" # change this to your AWS region

# Push metric to CloudWatch

aws cloudwatch put-metric-data \

--metric-name HttpdServiceStatus \

--namespace Custom/Apache \

--value $STATUS \

--region $REGION

echo "Httpd service status ($STATUS) pushed to CloudWatch in namespace Custom/Apache"

THEN RUN:

chmod +x httpd\_monitor.sh

./httpd\_monitor.sh

crontab -e

\*/5 \* \* \* \* /path/to/httpd\_monitor.sh

A computer screen with white text

AI-generated content may be incorrect.

**Verify Metrics in CloudWatch Console**

* Go to the **AWS Management Console** → **CloudWatch** → **Metrics**.
* Look for your custom namespace: Custom/Apache.
* You should see the metric named HttpdServiceStatus.
* Check if the metric values are showing correctly (1 for running, 0 for stopped).

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**📈 Graph Behavior**

Once active:

* When Apache is running, graph line stays at **1**.
* If the service goes down, it drops to **0**.
* You’ll visually see "uptime vs. downtime" over time.

A screenshot of a computer

AI-generated content may be incorrect.

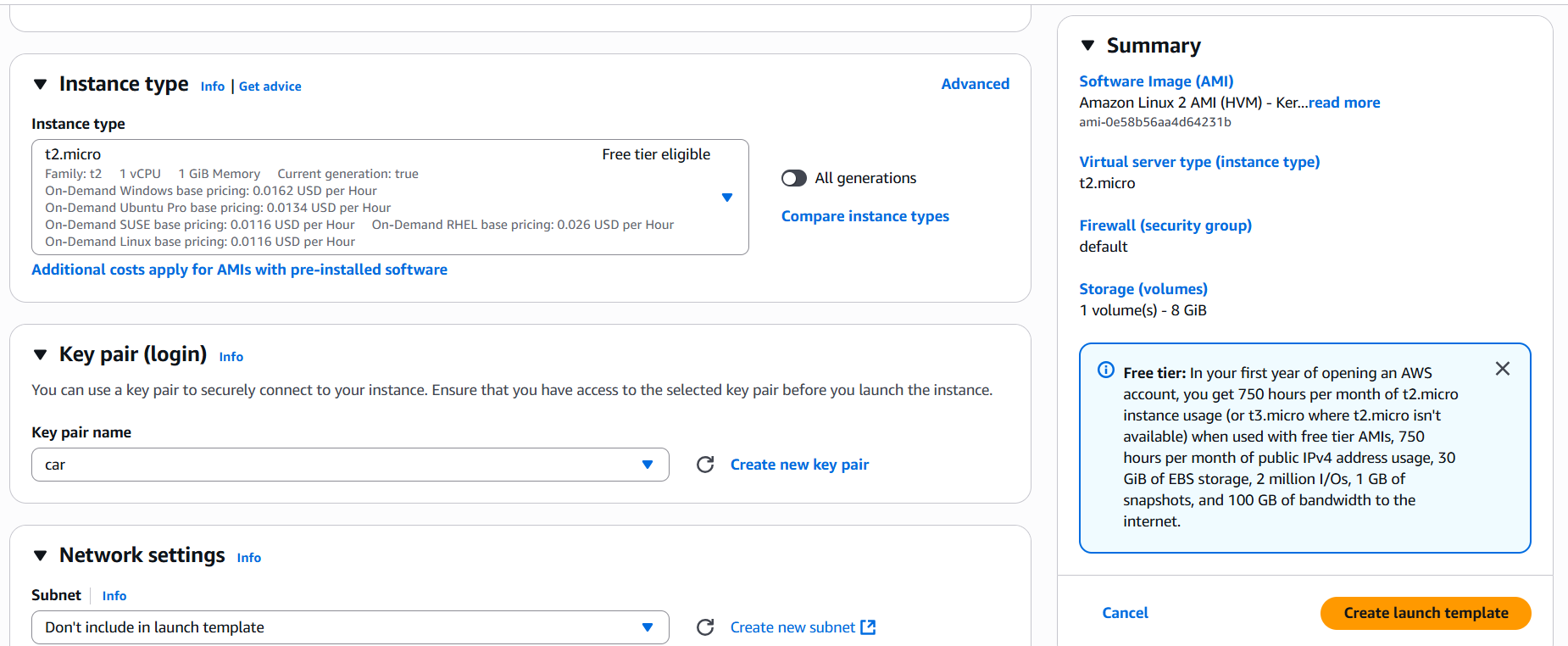
12) CPU utilizationis more than 70% then it should triggere Autoscaling and launch new instance.

**Step 1: Create a Launch Template**

This defines how your EC2 instances will be launched (AMI, instance type, key pair, etc.)

**📌 Do this via AWS Console:**

1. Go to **EC2 Console** → **Launch Templates**.
2. Click **Create launch template**.
3. Enter:
   * **Name**: MyTemplate
   * **AMI**: Choose Amazon Linux or the one you use
   * **Instance Type**: Example: t2.micro
   * **Key pair**: Select your key
   * **Security Group**: Allow port 80 or 22 as needed
4. Click **Create launch template**



A white box with black text

AI-generated content may be incorrect.

**Step 2: Create an Auto Scaling Group**

This is the group that automatically manages EC2 instances.

**📌 Via Console:**

1. Go to **EC2 Console** → **Auto Scaling Groups**
2. Click **Create Auto Scaling group**
3. Choose:
   * **Launch template**: Select the one created above (MyTemplate)
   * **Group name**: MyASG
4. Choose:
   * **VPC & Subnets**: Select 2 subnets in different AZs
   * **Load Balancer (optional)**: Skip for now unless using ALB
5. Set:
   * **Desired capacity**: 2
   * **Minimum capacity**: 1
   * **Maximum capacity**: 3
6. Click **Skip to review** → then **Create Auto Scaling group**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 3: Create a Scaling Policy (Scale Out)**

Now configure what happens when CPU > 70%.

**📌 In Auto Scaling Group:**

1. Go to your Auto Scaling Group (MyASG)
2. Click **Automatic scaling** → **Add policy**
3. Select **Target tracking** OR **Step scaling** → Choose **Step scaling**
4. Set:
   * **Policy type**: Step scaling
   * **CloudWatch Alarm**: Create a new one
     + Metric: EC2 -> CPU Utilization
     + Instance group dimension: Auto Scaling Group Name
     + Threshold: **Greater than 70%**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer error

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.