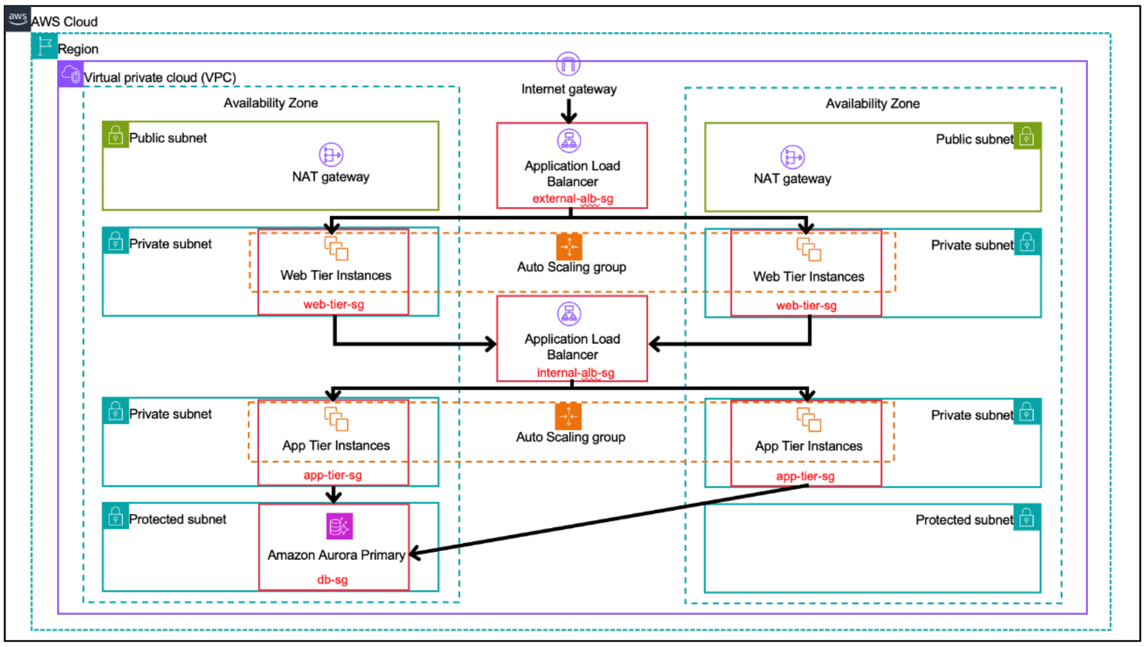
# AWS Workshop: Highly available & scalable three-tier application deployment on AWS

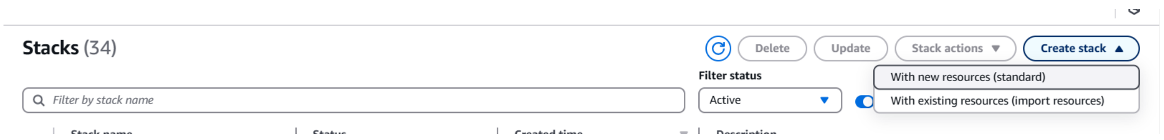
## Table of Contents

## AWS Architecture



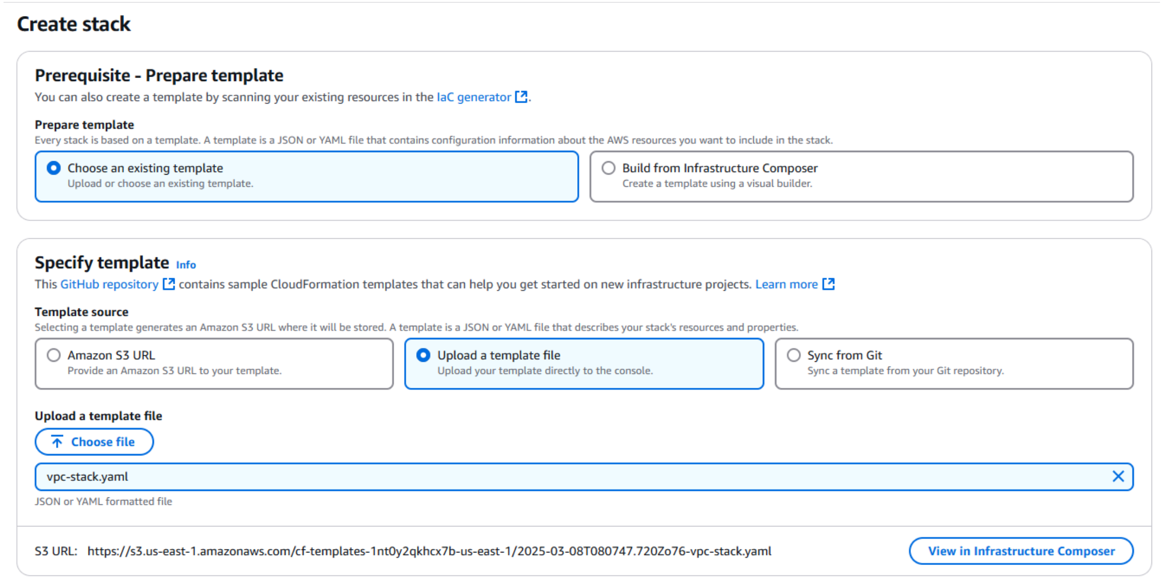
## Step 1: Setting up networking & IAM roles as pre-requisite

1. Go to **CloudFormation** service.
2. Click on **Create a stack With new resources.**



Stacks (34)

3. Download the ‘CFT link’ to deploy the pre-requisite for the LAB
4. Choose **Upload a template file** and upload the file downloaded.



Create stack

Prerequisite - Prepare template

You can also create a template by scanning your existing resources in the IaC generator.

Prepare template

Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

(●) Choose an existing template — Upload or choose an existing template.

( ) Build from Infrastructure Composer — Create a template using a visual builder.

Specify template Info

This GitHub repository contains sample CloudFormation templates that can help you get started on new infrastructure projects. Learn more

Template source

Selecting a template generates an Amazon S3 URL where it will be stored. A template is a JSON or YAML file that describes your stack's resources and properties.

( ) Amazon S3 URL — Provide an Amazon S3 URL to your template.

(●) Upload a template file — Upload your template directly to the console.

( ) Sync from Git — Sync a template from your Git repository.

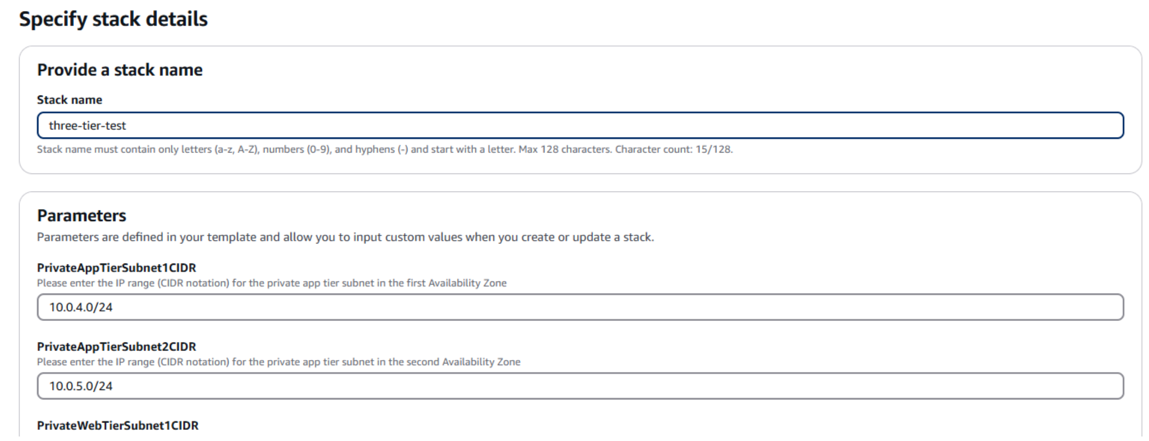
Upload a template file

vpc-stack.yaml

JSON or YAML formatted file

S3 URL: https://s3.us-east-1.amazonaws.com/cf-templates-1nt0y2qkhcx7b-us-east-1/2025-03-08T080747.720Zo76-vpc-stack.yaml

5. You can leave all parameters with default values.



Specify stack details

Provide a stack name

Stack name: three-tier-test

Stack name must contain only letters (a-z, A-Z), numbers (0-9), and hyphens (-) and start with a letter. Max 128 characters. Character count: 15/128.

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

PrivateAppTierSubnet1CIDR

Please enter the IP range (CIDR notation) for the private app tier subnet in the first Availability Zone

10.0.4.0/24

PrivateAppTierSubnet2CIDR

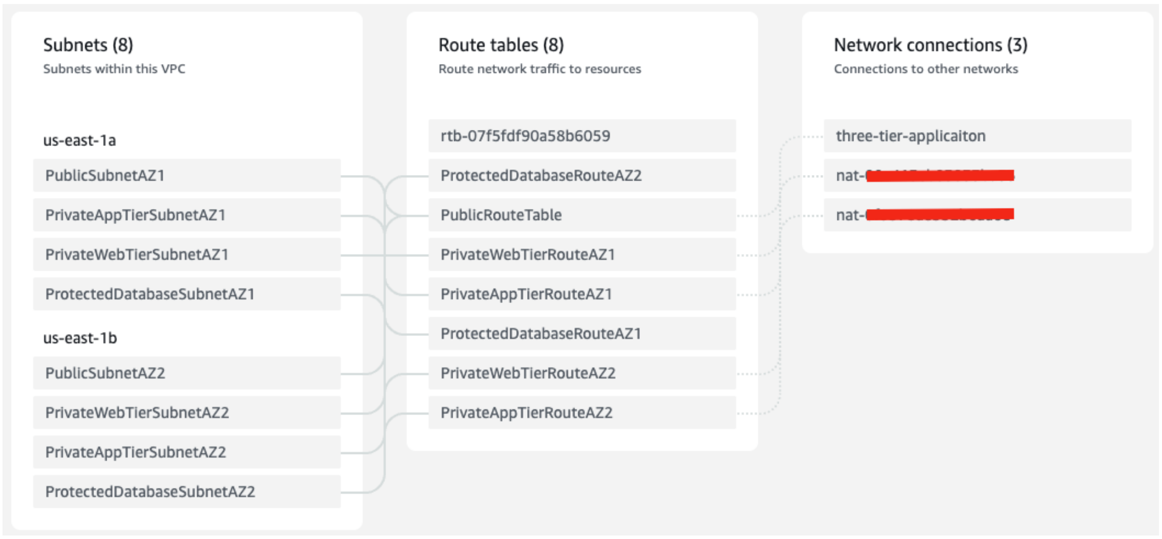
Please enter the IP range (CIDR notation) for the private app tier subnet in the second Availability Zone

10.0.5.0/24

PrivateWebTierSubnet1CIDR

6. Click on **Next** on **Step 2 (Specify stack details)** and **Step 3 (Configure stack options)**.
7. Finally, under review section click on **Submit**.
8. This stack creates the following resources:
   - **VPC** with **2 public**, **4 private**, and **2 protected subnets**. Two public subnets would be connected to a common route table, having network connections to

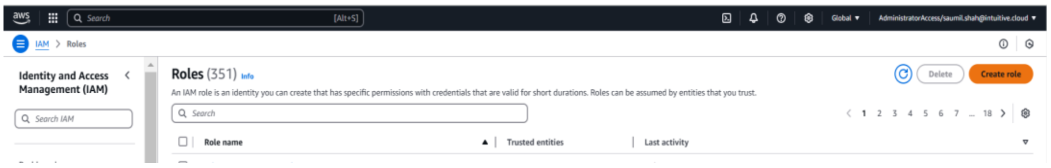
the **internet gateway**. Four private subnet will have 4 separate route tables, each route table will have network connects to to the **NAT gateway**. Protected subnets will have no path to the NAT gateways. Two private subnets will be used for frontend (web tier) logic and the other two private subnets for backend (app tier) logic. Protected subnets will have our RDS database.



- Security groups namely
    1. WebTierSecurityGroup – To be used for all WebTier resources [EC2, ELB]
    2. AppTierSecurityGroup – To be used for AppTier resources- [EC2]
    3. DatabaseSecurityGroup- To be used for DatabaseTier [RDS]

9. Create an **IAM instance profile** for EC2

    1. Open AWS Console and go to the **IAM** service.
    2. Click on **Roles**, then click **Create role.**

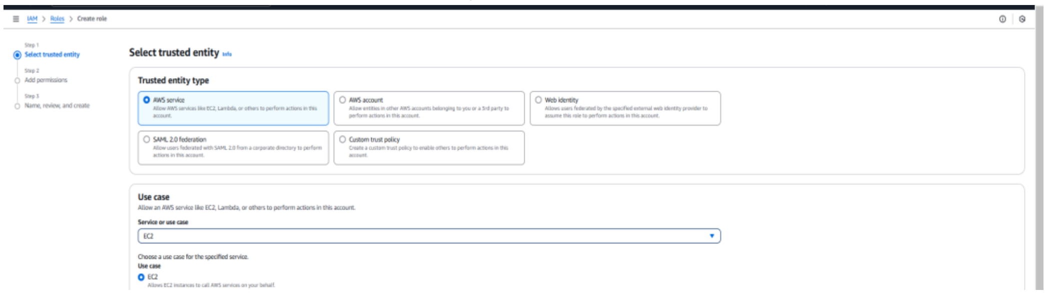


Roles (351)

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

| Role name | Trusted entities | Last activity |
|---|---|---|

    3. Select AWS service as the **trusted entity** and choose **EC2**. **Click Next.**



Select trusted entity

Trusted entity type

- [✓ AWS service] Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account: Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity: Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation: Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy: Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

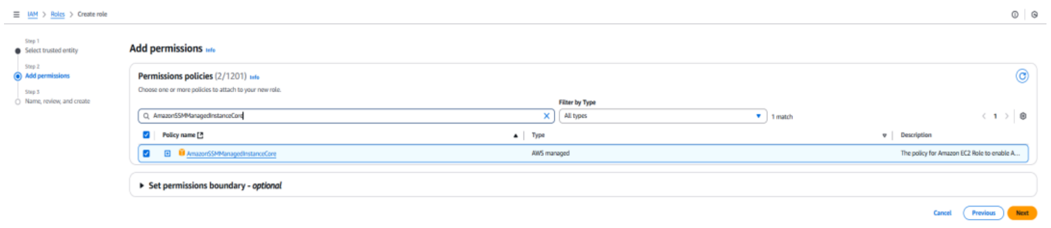
Service or use case: EC2

Choose a use case for the specified service.

Use case

- [✓ EC2] Allows EC2 instances to call AWS services on your behalf.

    4. Search for and attach the **'AmazonS3FullAccess'** and **'AmazonSSMManagedInstanceCore'** policies. **Click Next.**



**Permissions policies** (2/1201)

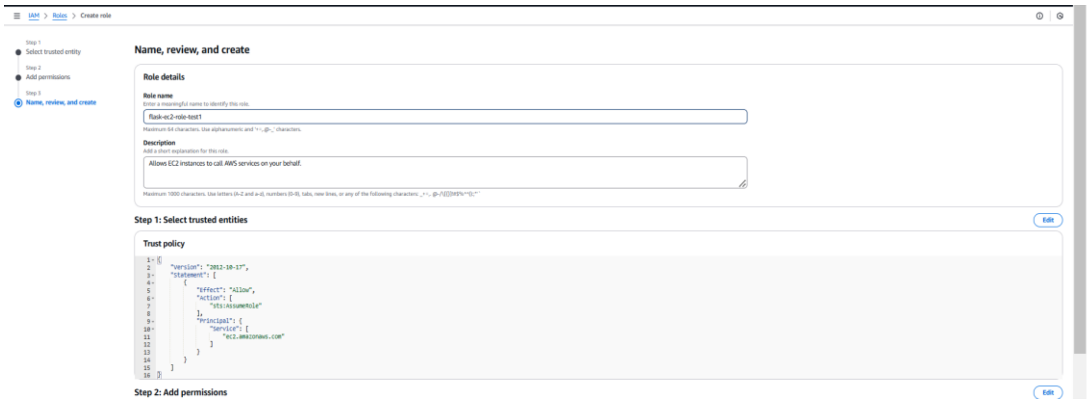
Choose one or more policies to attach to your new role.

Search: AmazonSSMManagedInstanceCore

Filter by Type: All types — 1 match

| | Policy name | Type | Description |
|---|---|---|---|
| ✓ | AmazonSSMManagedInstanceCore | AWS managed | The policy for Amazon EC2 Role to enable A... |

5. Enter the role name as flask-ec2-role-<your_user_id> and click **Create role.**



**Role details**

Role name

Enter a meaningful name to identify this role.

flask-ec2-role-test1

Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.

Description

Add a short explanation for this role.

Allows EC2 instances to call AWS services on your behalf.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,.@-/\[{}]!#$%^*();:"'

**Step 1: Select trusted entities**

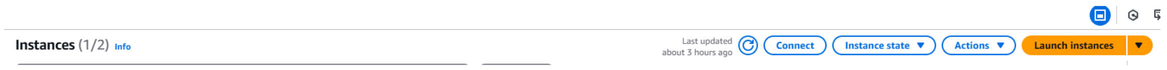
Trust policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

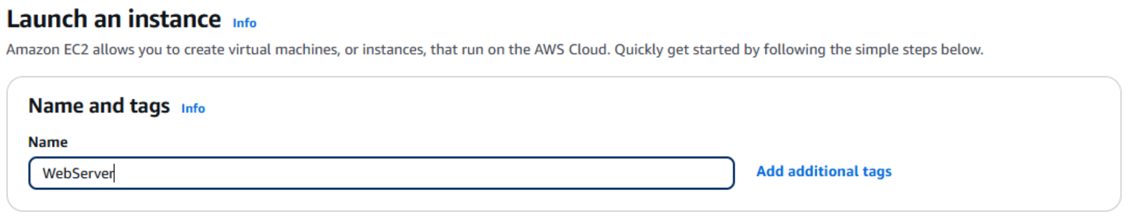
**Step 2: Add permissions**

## Step 2: Creating a Web Server using EC2 Instance

1. Open the Amazon EC2. From the EC2 console dashboard, in the Launch instance pane, choose **Launch instance**.



2. Under Name and tags, for name enter Websever.



Launch an instance Info

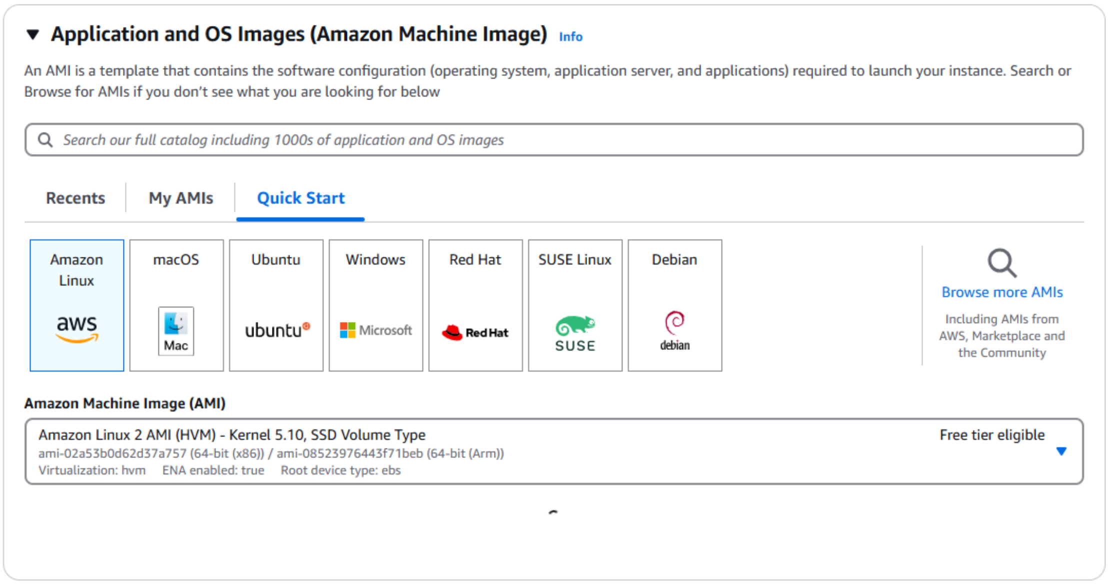
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: WebServer

Add additional tags

3. Under Application and OS Images (Amazon Machine Image). Choose **Quick Start** and then choose the operating system (OS) for your instance. From Amazon Machine Image (AMI), select **Amazon Linux 2AMI**.



Application and OS Images (Amazon Machine Image) Info

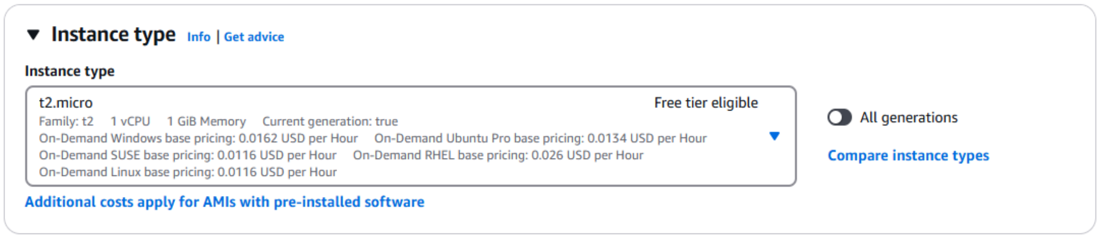
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type — Free tier eligible
ami-02a53b0d62d37a757 (64-bit (x86)) / ami-08523976443f71beb (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

4. Under Instance type, for Instance type, choose **t2.micro**.



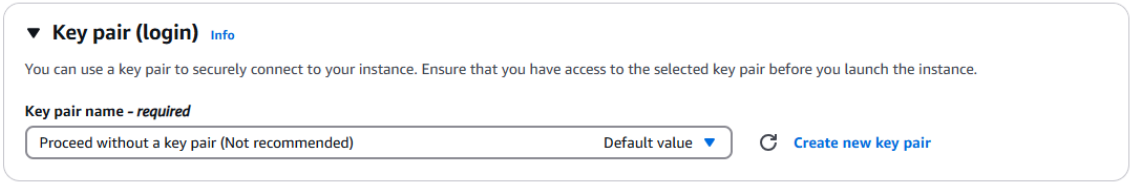
Instance type Info | Get advice

Instance type

t2.micro — Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.0162 USD per Hour On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour On-Demand RHEL base pricing: 0.026 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour

Additional costs apply for AMIs with pre-installed software

5. Under Key pair (login), **Proceed without a key pair**.



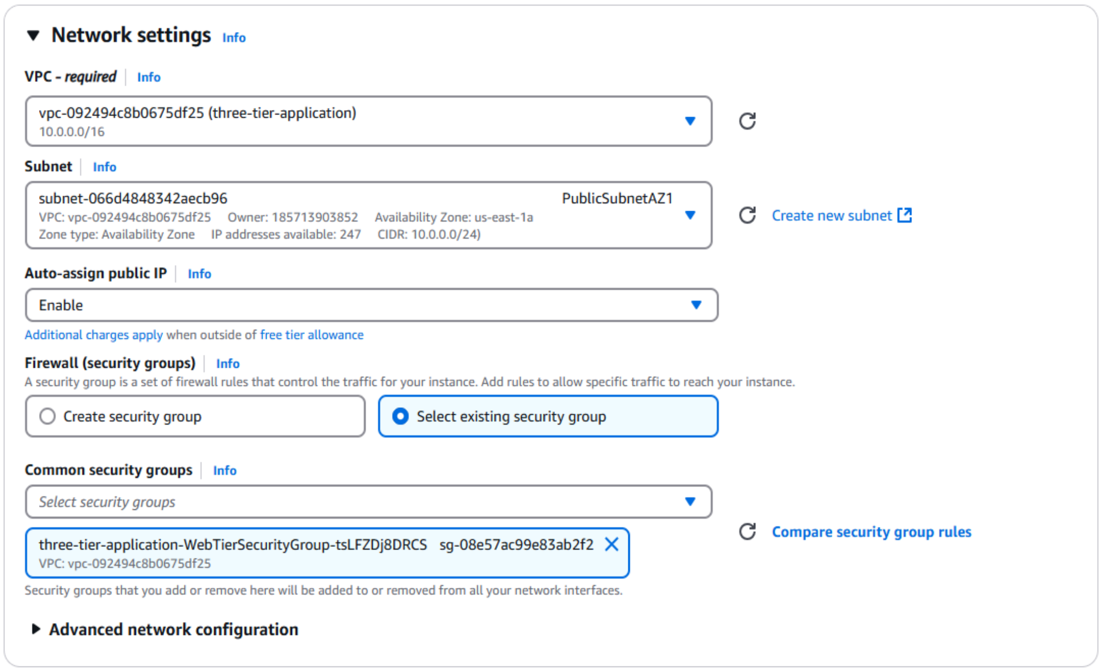
Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

Proceed without a key pair (Not recommended) — Default value

6. Under Network settings, select **Edit**, under VPC choose **three-tier-application** VPC created in step 1. Under subnet select **PublicSubnetAZ1**. For Auto-assign public IP select **Enable**. Choose Select **existing security group** and choose **WebTierSecurityGroup.**


Network settings Info
VPC - required Info
Subnet Info
Create new subnet
Auto-assign public IP Info
Firewall (security groups) Info
Create security group
Select existing security group
Common security groups Info
Select security groups
Compare security group rules
Advanced network configuration

VPC: vpc-092494c8b0675df25 (three-tier-application) 10.0.0.0/16

Subnet: subnet-066d4848342aecb96 PublicSubnetAZ1
VPC: vpc-092494c8b0675df25 Owner: 185713903852 Availability Zone: us-east-1a
Zone type: Availability Zone IP addresses available: 247 CIDR: 10.0.0.0/24)

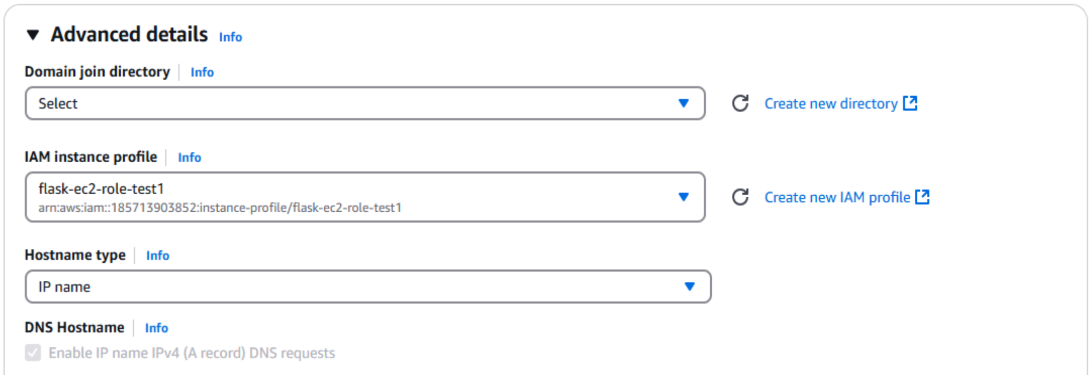
Auto-assign public IP: Enable
Additional charges apply when outside of free tier allowance

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

three-tier-application-WebTierSecurityGroup-tsLFZDj8DRCS sg-08e57ac99e83ab2f2
VPC: vpc-092494c8b0675df25

Security groups that you add or remove here will be added to or removed from all your network interfaces.

7. Under Advanced Details section, for **IAM instance profile** select **flask-ec2-role-<your_user_id>** created in step-1.



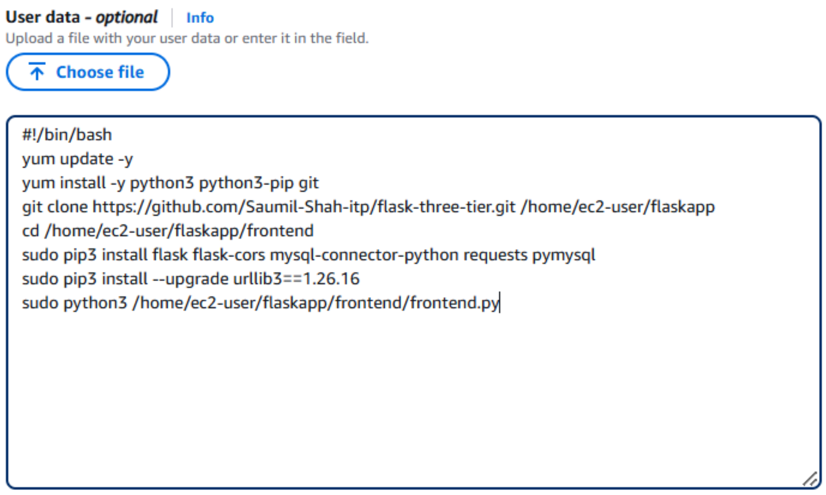
Domain join directory: Select

IAM instance profile: flask-ec2-role-test1
arn:aws:iam::185713903852:instance-profile/flask-ec2-role-test1

Hostname type: IP name

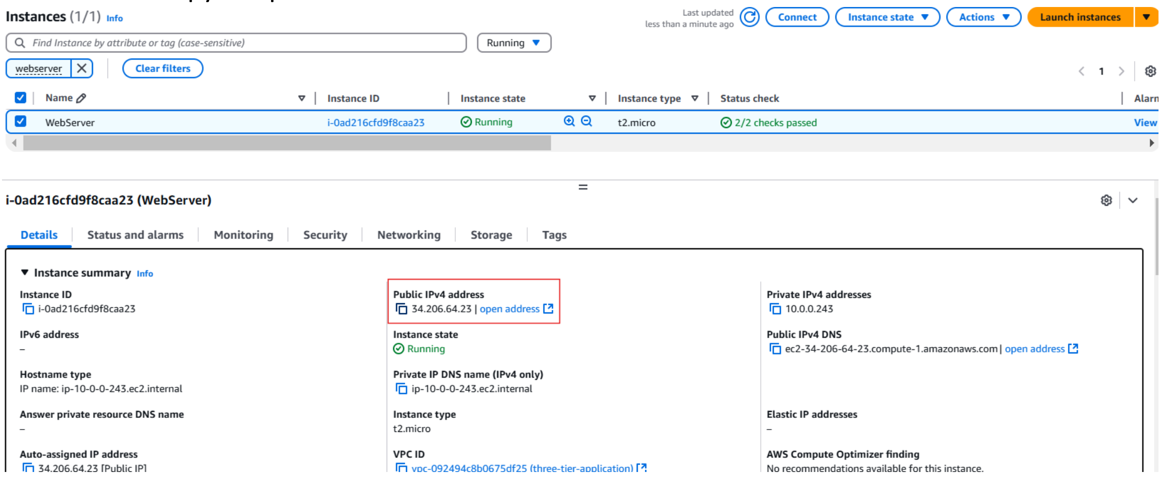
DNS Hostname: Enable IP name IPv4 (A record) DNS requests

8. Scroll down to **User Data** and copy contents from this Link.


User data - optional Info
Upload a file with your user data or enter it in the field.
Choose file

```
#!/bin/bash
yum update -y
yum install -y python3 python3-pip git
git clone https://github.com/Saumil-Shah-itp/flask-three-tier.git /home/ec2-user/flaskapp
cd /home/ec2-user/flaskapp/frontend
sudo pip3 install flask flask-cors mysql-connector-python requests pymysql
sudo pip3 install --upgrade urllib3==1.26.16
sudo python3 /home/ec2-user/flaskapp/frontend/frontend.py
```

9. Click on **Launch Instance**.
10. Once the instance is up and running in **Healthy** state with **2/2 checks passed**. Select the instance and copy the public IP.



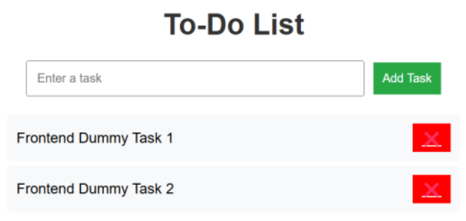
| Name | Instance ID | Instance state | Instance type | Status check | Alarm |
|---|---|---|---|---|---|
| WebServer | i-0ad216cfd9f8caa23 | Running | t2.micro | 2/2 checks passed | View |

i-0ad216cfd9f8caa23 (WebServer)

Instance summary Info

| | | |
|---|---|---|
| Instance ID<br>i-0ad216cfd9f8caa23 | Public IPv4 address<br>34.206.64.23 \| open address | Private IPv4 addresses<br>10.0.0.243 |
| IPv6 address<br>– | Instance state<br>Running | Public IPv4 DNS<br>ec2-34-206-64-23.compute-1.amazonaws.com \| open address |
| Hostname type<br>IP name: ip-10-0-0-243.ec2.internal | Private IP DNS name (IPv4 only)<br>ip-10-0-0-243.ec2.internal | |
| Answer private resource DNS name<br>– | Instance type<br>t2.micro | Elastic IP addresses<br>– |
| Auto-assigned IP address<br>34.206.64.23 [Public IP] | VPC ID<br>vpc-092494c8b0675df25 (three-tier-application) | AWS Compute Optimizer finding<br>No recommendations available for this instance. |

11. Open your browser and search for http://<public_id>:80. The screen below should appear.


To-Do List
Enter a task
Add Task
Frontend Dummy Task 1
Frontend Dummy Task 2