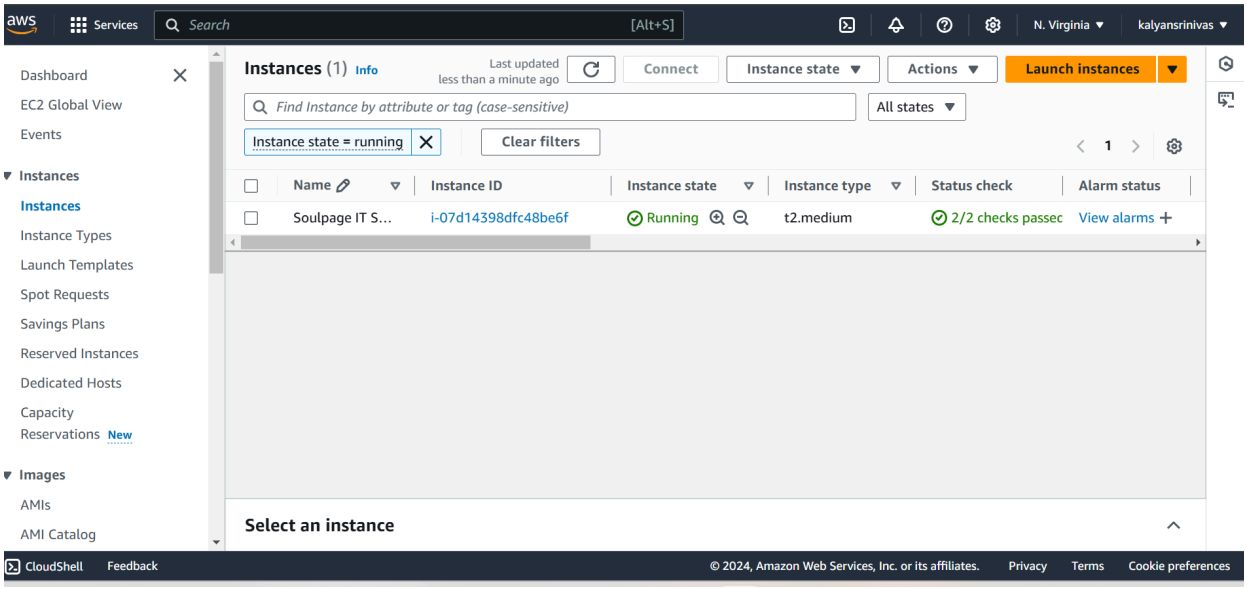
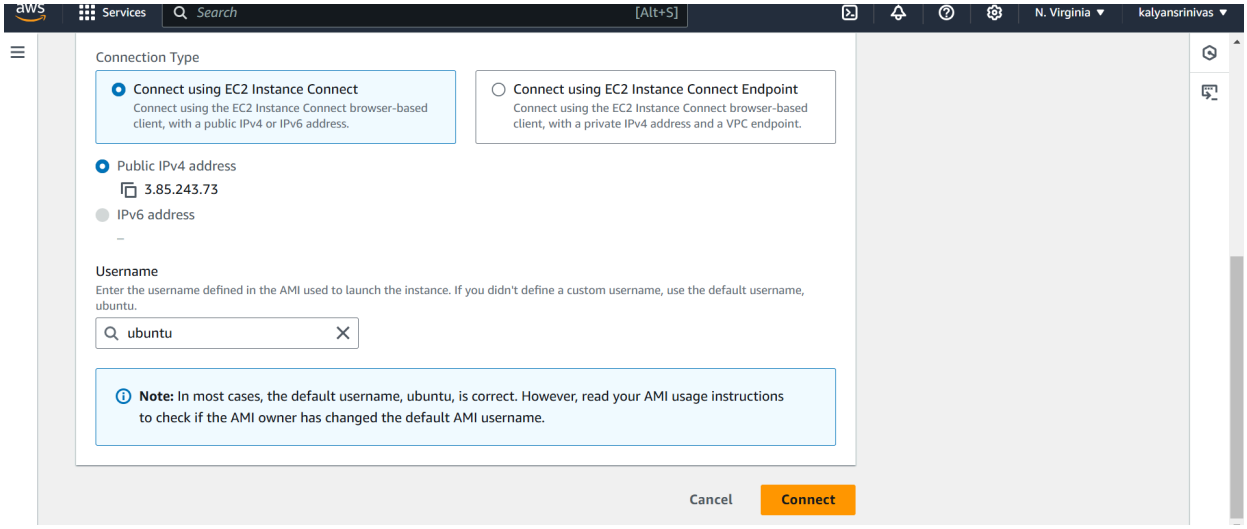
# Soulpage IT Solutions Tasks

## Task -1

Step 1: *Set up a virtual machine or a cloud instance (AWS, Azure, or any other preferred provider ) also install the necessary tools and dependencies (Nodejs, Python, Docker, etc …)*

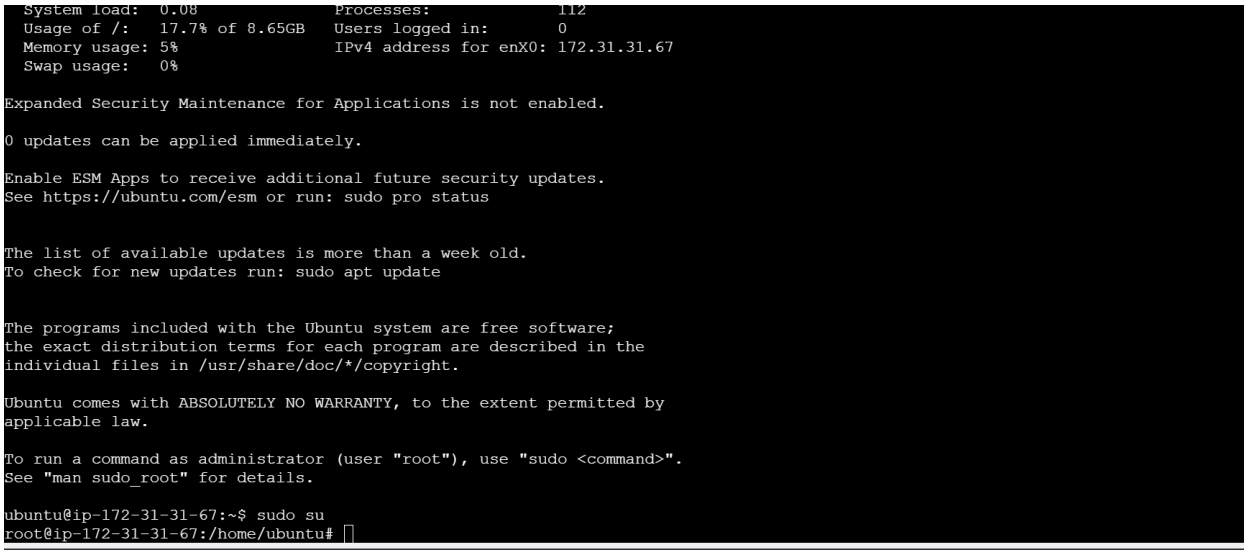
*I have used cloud instance AWS (EC2 instance ) and installed all the necessary tools and dependencies (Nodejs, Python, Docker, etc …)*






*Click on Connect*

Step 2 : *After connecting to the AWS CLI the first command to type is sudo su*

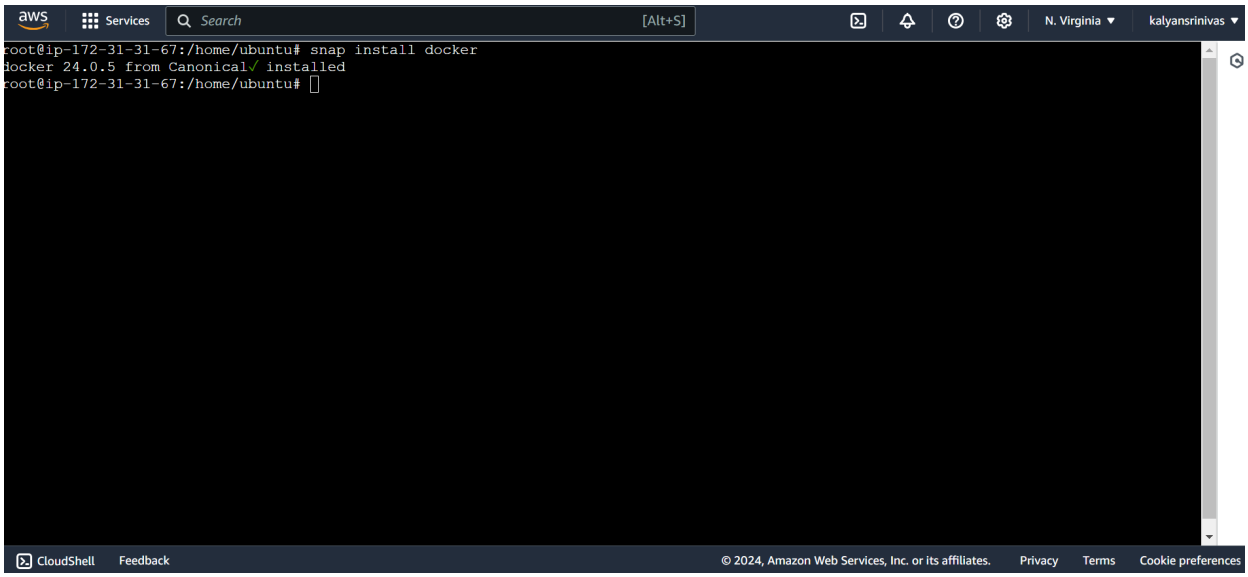



*Use the sudo apt update and sudo apt upgrade command to update and upgrade the dependencies.*

Step 3 :  Install  Nodejs , Python and Docker

Install  Docker

Snap install docker is the command used to install docker .




Next, download and install Nodejs

Use sudo apt install -y nodejs  command to install Nodejs

```
root@ip-172-31-31-67:/home/ubuntu# sudo apt install -y nodejs
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages will be REMOVED:
  nsolid
The following NEW packages will be installed:
  nodejs
0 upgraded, 1 newly installed, 1 to remove and 0 not upgraded.
Need to get 31.8 MB of archives.
After this operation, 24.9 MB of additional disk space will be used.
Get:1 https://deb.nodesource.com/node_20.x nodistro/main amd64 nodejs amd64 20.18.0-1nodesource1 [3
Fetched 31.8 MB in 0s (66.2 MB/s)
(Reading database ... 106504 files and directories currently installed.)
Removing nsolid (20.18.0-ns5.4.0) ...
Selecting previously unselected package nodejs.
(Reading database ... 101199 files and directories currently installed.)
Preparing to unpack .../nodejs_20.18.0-1nodesource1_amd64.deb ...
Unpacking nodejs (20.18.0-1nodesource1) ...
Setting up nodejs (20.18.0-1nodesource1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
```

```
Pending kernel upgrade!
Running kernel version:
  6.8.0-1016-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.8.0-1017-aws.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

Restarting services...

Service restarts being deferred:
 systemctl restart unattended-upgrades.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-31-67:/home/ubuntu#
```

Finally, install Python

Use sudo apt install -y python3 command to install Python

```
root@ip-172-31-31-67:/home/ubuntu# sudo apt install -y python3
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.12.3-0ubuntu2).
python3 set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@ip-172-31-31-67:/home/ubuntu#
```

Step4 : Nextjs Deployment

Create a New Next.js Project:

bash
Copy code

```
mkdir my-nextjs-app
cd my-nextjs-app
npx create-next-app@latest .
```

```
root@ip-172-31-31-67:/home/ubuntu/my-nextjs-app# npx create-next-app@latest .
Need to install the following packages:
create-next-app@15.0.2
Ok to proceed? (y) y

√ Would you like to use TypeScript? … No / Yes
√ Would you like to use ESLint? … No / Yes
√ Would you like to use Tailwind CSS? … No / Yes
√ Would you like your code inside a `src/` directory? … No / Yes
√ Would you like to use App Router? (recommended) … No / Yes
√ Would you like to use Turbopack for next dev? … No / Yes
√ Would you like to customize the import alias (@/* by default)? … No / Yes
Creating a new Next.js app in /home/ubuntu/my-nextjs-app.

Using npm.

Initializing project with template: app-tw


Installing dependencies:
- react
- react-dom
- next
```

```
Installing devDependencies:
- typescript
- @types/node
- @types/react
- @types/react-dom
- postcss
- tailwindcss
- eslint
- eslint-config-next

npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a
ood and tested way to coalesce async requests by a key value, which is much more comprehensive and powerful.
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated rimraf@3.0.2: Rimraf versions prior to v4 are no longer supported
npm warn deprecated @humanwhocodes/object-schema@2.0.3: Use @eslint/object-schema instead
npm warn deprecated @humanwhocodes/config-array@0.13.0: Use @eslint/config-array instead
npm warn deprecated eslint@8.57.1: This version is no longer supported. Please see https://eslint.org/version-support for other optic
s.

added 369 packages, and audited 370 packages in 32s

137 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
Initialized a git repository.

Success! Created my-nextjs-app at /home/ubuntu/my-nextjs-app
```

- The setup wizard will prompt you to name the project and install initial dependencies.

Configure Environment Variables:

- If your application requires environment variables, create a `.env.local` file in the project root and add them there:

```
NEXT_PUBLIC_API_URL=http://example.com/api
```

Build and Start the Next.js Application:

```
npm run build
```

```
aws  Services  Search [Alt+S]  N. Virginia ▾  kalyansrinivas ▾
root@ip-172-31-31-67:/home/ubuntu/my-nextjs-app# vi .env.local
root@ip-172-31-31-67:/home/ubuntu/my-nextjs-app# npm run build

> my-nextjs-app@0.1.0 build
> next build

Attention: Next.js now collects completely anonymous telemetry regarding usage.
This information is used to shape Next.js' roadmap and prioritize features.
You can learn more, including how to opt-out if you'd not like to participate in this anonymous program, by visiting the following URL
:
https://nextjs.org/telemetry

   ▲ Next.js 15.0.2
   - Environments: .env.local

   Creating an optimized production build ...
 ✓ Compiled successfully
 ✓ Linting and checking validity of types
 ✓ Collecting page data
 ✓ Generating static pages (5/5)
 ✓ Collecting build traces
 ✓ Finalizing page optimization

Route (app)                              Size     First Load JS
┌ ○ /                                    5.46 kB         105 kB
└ ○ /_not-found                          897 B           101 kB
+ First Load JS shared by all            99.7 kB
  ├ chunks/215-f207ea7968f9b6d8.js       45.2 kB
  ├ chunks/4bd1b696-23516f99b565b560.js  52.6 kB
```

```
Route (app)                              Size     First Load JS
┌ ○ /                                    5.46 kB         105 kB
└ ○ /_not-found                          897 B           101 kB
+ First Load JS shared by all            99.7 kB
  ├ chunks/215-f207ea7968f9b6d8.js       45.2 kB
  ├ chunks/4bd1b696-23516f99b565b560.js  52.6 kB
  └ other shared chunks (total)          1.88 kB


○  (Static)  prerendered as static content

root@ip-172-31-31-67:/home/ubuntu/my-nextjs-app#
```

## npm start

```
○  (Static)  prerendered as static content

root@ip-172-31-31-67:/home/ubuntu/my-nextjs-app# npm start

> my-nextjs-app@0.1.0 start
> next start

   ▲ Next.js 15.0.2
   - Local:        http://localhost:3000

 ✓ Starting...
 ✓ Ready in 495ms
```

By default, Next.js runs on port 3000. Access it via `http://your_instance_ip:3000`.

Step 5 :  Django Setup and Deployment

*Create a New Django Project:*

*mkdir my-django-app*
*cd my-django-app*
*python3 -m venv venv*
*source venv/bin/activate*
*pip install django*

```
root@ip-172-31-31-67:/home/ubuntu/my-nextjs-app/my-django-app# source venv/bin/activate
(venv) root@ip-172-31-31-67:/home/ubuntu/my-nextjs-app/my-django-app# pip install django
Collecting django
  Downloading Django-5.1.2-py3-none-any.whl.metadata (4.2 kB)
Collecting asgiref<4,>=3.8.1 (from django)
  Downloading asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse>=0.3.1 (from django)
  Downloading sqlparse-0.5.1-py3-none-any.whl.metadata (3.9 kB)
Downloading Django-5.1.2-py3-none-any.whl (8.3 MB)
   ---------------------------------------- 8.3/8.3 MB 73.3 MB/s eta 0:00:00
Downloading asgiref-3.8.1-py3-none-any.whl (23 kB)
Downloading sqlparse-0.5.1-py3-none-any.whl (44 kB)
   ---------------------------------------- 44.2/44.2 kB 6.2 MB/s eta 0:00:00
Installing collected packages: sqlparse, asgiref, django
Successfully installed asgiref-3.8.1 django-5.1.2 sqlparse-0.5.1
(venv) root@ip-172-31-31-67:/home/ubuntu/my-nextjs-app/my-django-app# django-admin startproject myproject .
(venv) root@ip-172-31-31-67:/home/ubuntu/my-nextjs-app/my-django-app#
```

*django-admin startproject myproject .*

*Install Additional Dependencies:*

*If you need packages like `djangorestframework`, you can install them using:*

*pip install djangorestframework*

```
(venv) root@ip-172-31-31-67:/home/ubuntu/my-nextjs-app/my-django-app# pip install djangorestframework
Collecting djangorestframework
  Downloading djangorestframework-3.15.2-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: django>=4.2 in ./venv/lib/python3.12/site-packages (from djangorestframework) (5.1.2)
Requirement already satisfied: asgiref<4,>=3.8.1 in ./venv/lib/python3.12/site-packages (from django>=4.2->djangorestframework) (3.8.1
)
Requirement already satisfied: sqlparse>=0.3.1 in ./venv/lib/python3.12/site-packages (from django>=4.2->djangorestframework) (0.5.1)
Downloading djangorestframework-3.15.2-py3-none-any.whl (1.1 MB)
   ---------------------------------------- 1.1/1.1 MB 23.2 MB/s eta 0:00:00
Installing collected packages: djangorestframework
Successfully installed djangorestframework-3.15.2
(venv) root@ip-172-31-31-67:/home/ubuntu/my-nextjs-app/my-django-app#
```

*Configure Environment Variables and Database Settings:*

- *In `myproject/settings.py`, configure the database, allowed hosts, and other settings.*
- *If using environment variables, create a `.env` file and load it in settings*

*SECRET_KEY=your_secret_key*

*DATABASE_URL=postgres://user:password@localhost:5432/mydatabase*

*Apply Migrations and Create Superuser:*

*python manage.py migrate*
*python manage.py createsuperuser*

*Run the Django Application:*
*bash*

*Copy code*

```
python manage.py runserver 0.0.0.0:8000
```

- 
    - *The application is now accessible at* `http://your_instance_ip:8000`*.*
- *Deploy with Gunicorn and Nginx (Optional for Production):*

*Install Gunicorn:*
*bash*
*Copy code*

```
pip install gunicorn
```

```
python: can't open file '/home/ubuntu/my-nextjs-app/my-django-app/myproject/manage.py': [Errno 2] No such fil
(venv) root@ip-172-31-31-67:/home/ubuntu/my-nextjs-app/my-django-app/myproject# pip install gunicorn
Collecting gunicorn
  Downloading gunicorn-23.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from gunicorn)
  Downloading packaging-24.1-py3-none-any.whl.metadata (3.2 kB)
Downloading gunicorn-23.0.0-py3-none-any.whl (85 kB)
   ---------------------------------------- 85.0/85.0 kB 4.6 MB/s eta 0:00:00
Downloading packaging-24.1-py3-none-any.whl (53 kB)
   ---------------------------------------- 54.0/54.0 kB 8.0 MB/s eta 0:00:00
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-23.0.0 packaging-24.1
(venv) root@ip-172-31-31-67:/home/ubuntu/my-nextjs-app/my-django-app/myproject#
```

- 

*Run Gunicorn:*
*bash*
*Copy code*

```
gunicorn --workers 3 myproject.wsgi:application --bind 0.0.0.0:8000
```

- 
- *Optionally configure Nginx to reverse proxy to Gunicorn for production-grade deployments.*

*5. Monitoring Setup (Optional)*

- *Install and Configure Prometheus and Grafana for self-hosted monitoring, or:*
- *Use New Relic or Datadog for cloud-based monitoring, with alerting for critical metrics (CPU, memory, response time, error rates).*

Task 2 :

*Step 1: Configure Jenkins*

1. *Install Jenkins: Ensure Jenkins is installed and accessible.*
2. *Install Necessary Plugins:*
    - *Install plugins like Git (for source control), Pipeline, Docker (for containerized deployments), and any cloud-specific plugins (like AWS or Azure if applicable).*

*Step 2: Set Up Jenkins Pipeline*

1. *Create a Pipeline Job:*
    - *Open Jenkins, create a new Pipeline job, and connect it to your Git repository.*
    - *Configure webhooks (e.g., in GitHub) to trigger Jenkins jobs on every push.*
2. *Pipeline Script:*
    - *In the `Jenkinsfile` (place it in your repo's root), define the stages for testing, building, and deployment. Here's a sample setup:*

```
pipeline {
```

```
    agent any
    environment {
        DOCKER_IMAGE = "your-image-name" // Replace with your Docker
image name
        DOCKER_REGISTRY_CREDENTIALS_ID = 'docker-hub-credentials'
// Jenkins credentials ID
    }
    stages {
        stage('Checkout') {
            steps {
                git branch: 'main', url: 'https://github.com/your-repo-url.git'
            }
        }
        stage('Test') {
            steps {
                sh 'npm run test' // Adjust for testing commands in your project
            }
        }
        stage('Build') {
            steps {
                sh 'npm run build' // Adjust for build commands in your project
                sh 'docker build -t $DOCKER_IMAGE .'
            }
        }
```

```
        stage('Push Image') {
            steps {
                withCredentials([usernamePassword(credentialsId: DOCKER_REGISTRY_CREDENTIALS_ID, passwordVariable: 'DOCKER_PASSWORD', usernameVariable: 'DOCKER_USERNAME')]) {
                    sh "docker login -u $DOCKER_USERNAME -p $DOCKER_PASSWORD"
                    sh "docker push $DOCKER_IMAGE"
                }
            }
        }
        stage('Deploy') {
            steps {
                sh 'kubectl apply -f deployment.yaml' // Adjust for Kubernetes, if using Kubernetes
            }
        }
    }
}
```

***Step 3: Database Backup and Restore Strategy***

1. ***Database Backup**:*

- *Use a cron job or scheduled job in Jenkins to back up the database regularly (e.g., daily or weekly).*
- *For example, to back up a MongoDB database:*

```
# Scheduled in Jenkins or a cron job
TIMESTAMP=$(date +"%F")
BACKUP_DIR="/path/to/backup/$TIMESTAMP"
mkdir -p "$BACKUP_DIR"
mongodump --db your-database-name --out "$BACKUP_DIR"
```

***Database Restore Script**:*

- *Create a restore script to re-import the database in case of data loss.*

```
# Run this script with the backup path as an argument
BACKUP_DIR=$1
mongorestore --db your-database-name
"$BACKUP_DIR/your-database-name"
```

***Automate Backups and Restoration**:*

- *Optionally, use Jenkins jobs for backup and restore scripts to maintain a consistent backup schedule and trigger restores when needed.*

*Task 3 :*

*1. Provision Infrastructure with Terraform or Ansible*

***Using Terraform:***

*Here's a sample Terraform configuration to provision a virtual machine on AWS (can be adapted for Azure, GCP, or other providers):*

```
provider "aws" {
  region = "us-west-2"
}

resource "aws_instance" "next_django_vm" {
  ami           = "ami-0c55b159cbfafe1f0"  # Update with the latest stable AMI for your region
  instance_type = "t2.micro"
  tags = {
    Name = "Next-Django-Server"
  }

  provisioner "remote-exec" {
```

```
    inline = [
      "sudo apt update -y",
      "sudo apt install -y docker.io docker-compose"
    ]
  }
}
```

***Using Ansible:***

*An Ansible playbook can help you provision infrastructure and install Docker and Docker Compose on the virtual machine.*

```
  tasks:- hosts: all
  become: true

    - name: Update apt repository
      apt:
        update_cache: yes

    - name: Install Docker
      apt:
        name: docker.io
        state: present
```

```
  - name: Install Docker Compose
    apt:
      name: docker-compose
      state: present
```

### 2. Dockerize the Next.js and Django Applications

#### Dockerfile for Next.js

```
# Dockerfile for Next.js
FROM node:18-alpine

WORKDIR /app

COPY package.json ./
COPY package-lock.json ./
RUN npm install

COPY . .

RUN npm run build

EXPOSE 3000
CMD ["npm", "start"]
```

*Dockerfile for Django*

```
# Dockerfile for Django
FROM python:3.10

WORKDIR /app

COPY requirements.txt ./
RUN pip install -r requirements.txt

COPY . .

EXPOSE 8000
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```

***3. Docker Compose Setup***

*Create a `docker-compose.yml` file to define and manage both containers, including network configuration to enable communication between Next.js and Django.*

```yaml
version: '3.8'

services:
  nextjs:
    build:
      context: ./path-to-nextjs
      dockerfile: Dockerfile
    ports:
      - "3000:3000"
    depends_on:
      - django
    networks:
      - app-network

  django:
    build:
      context: ./path-to-django
      dockerfile: Dockerfile
    ports:
      - "8000:8000"
    networks:
      - app-network
```

```
networks:
  app-network:
    driver: bridge
```

***4. Run the Setup***

1. ***Terraform/Ansible****: Use the Terraform script to spin up the infrastructure, or run the Ansible playbook to install Docker and Docker Compose on the server.*
2. ***Deploy Docker Containers****: After provisioning, SSH into the server, clone your application repositories, and run* `docker-compose up -d` *to deploy the applications.*