Firist we need a machine to start out project on , so I decide to run on vm in vmworkstation with centos OS

Project Tasks

1. Set Up the Project

• Clone the project repository.

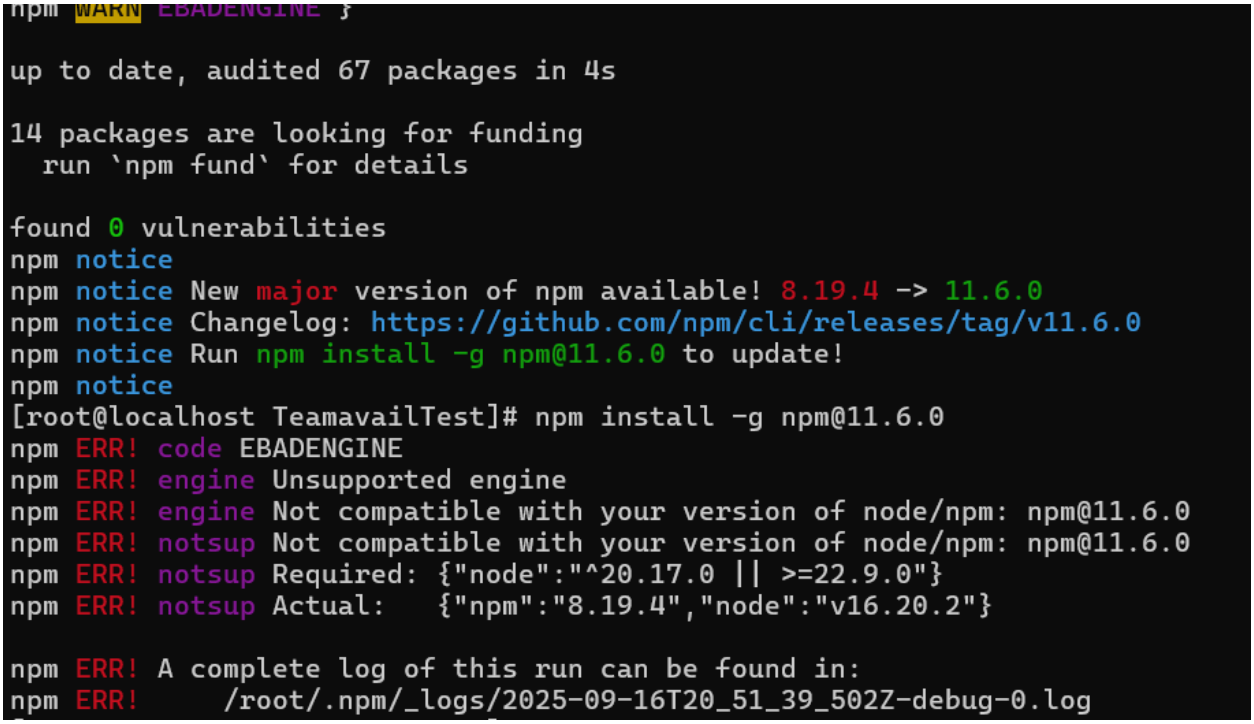
```
[root@localhost ~]# git clone https://github.com/ge0rgeK/TeamavailTest.git
Cloning into 'TeamavailTest'...
remote: Enumerating objects: 705, done.
remote: Counting objects: 100% (44/44), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 705 (delta 22), reused 13 (delta 13), pack-reused 661 (from 2)
Receiving objects: 100% (705/705), 720.66 KiB | 1.63 MiB/s, done.
Resolving deltas: 100% (129/129), done.
```

• Create a .gitignore file if missing.

```
[root@localhost TeamavailTest]# ls
input  node_modules  output  package.json  package-lock.json  public  server.js
[root@localhost TeamavailTest]# touch .gitignore
[root@localhost TeamavailTest]# ls
input  node_modules  output  package.json  package-lock.json  public  server.js
[root@localhost TeamavailTest]# ls l-a
ls: cannot access 'l-a': No such file or directory
[root@localhost TeamavailTest]# ls -la
total 56
drwxr-xr-x.  7 root root   176 Sep 16 23:46 .
dr-xr-x---. 20 root root  4096 Sep 16 23:44 ..
-rw-r--r--.  1 root root  8196 Sep 16 23:44 .DS_Store
drwxr-xr-x.  8 root root   163 Sep 16 23:44 .git
-rw-r--r--.  1 root root     0 Sep 16 23:46 .gitignore
drwxr-xr-x.  2 root root    65 Sep 16 23:44 input
drwxr-xr-x. 68 root root  4096 Sep 16 23:44 node_modules
drwxr-xr-x.  2 root root    23 Sep 16 23:44 output
-rw-r--r--.  1 root root   304 Sep 16 23:44 package.json
-rw-r--r--.  1 root root 28604 Sep 16 23:44 package-lock.json
drwxr-xr-x.  2 root root    76 Sep 16 23:44 public
-rw-r--r--.  1 root root  1106 Sep 16 23:44 server.js
[root@localhost TeamavailTest]#
```

• Install the required dependencies locally.

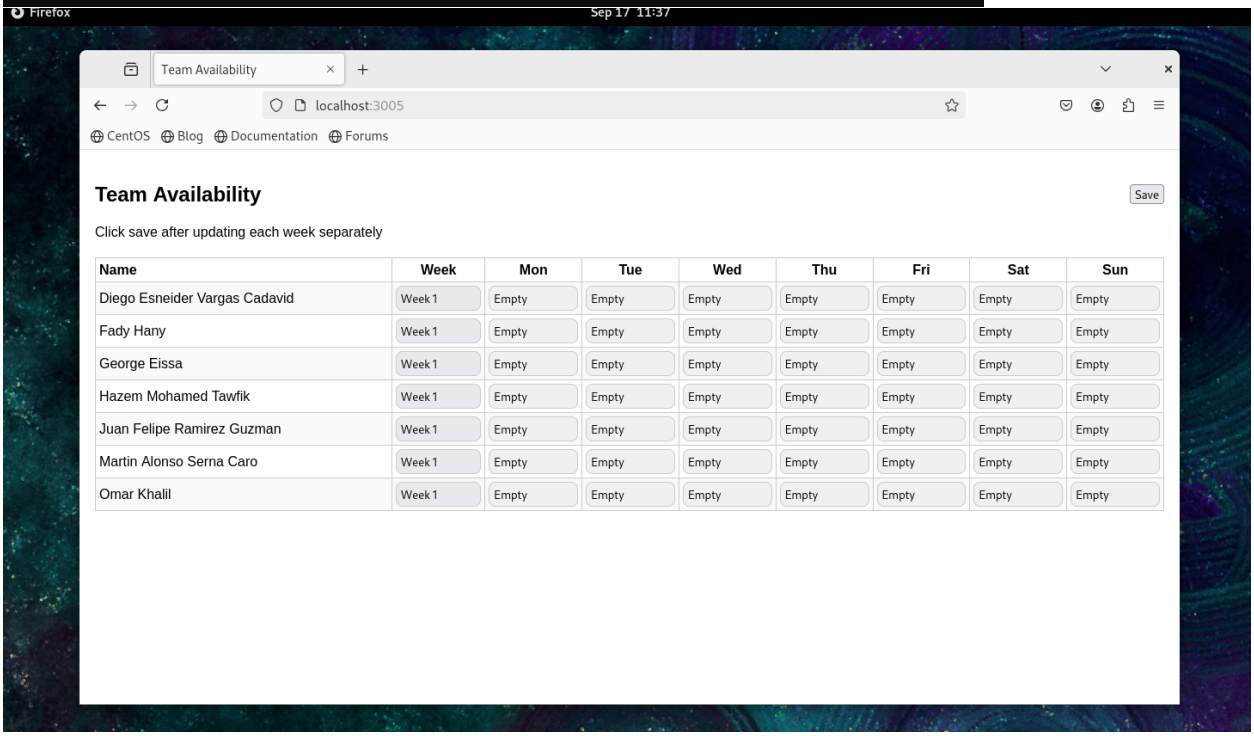
In this step after I run the command (npm install) I faced that I have too old versions of npm and node



And to solve it we need to do those steps

yum remove -y nodejs npm
curl -fsSL https://rpm.nodesource.com/setup_20.x | bash -
yum install -y nodejs
and then try again (npm install)

```
Running transaction
  Preparing        :
  Upgrading        : nodejs-docs-2:20.19.5-1nodesource.noarch
  Running scriptlet: nodejs-2:20.19.5-1nodesource.x86_64
  Installing       : nodejs-2:20.19.5-1nodesource.x86_64
  Cleanup          : nodejs-docs-1:16.20.2-8.el9.noarch
  Running scriptlet: nodejs-docs-1:16.20.2-8.el9.noarch
  Verifying        : nodejs-2:20.19.5-1nodesource.x86_64
  Verifying        : nodejs-docs-2:20.19.5-1nodesource.noarch
  Verifying        : nodejs-docs-1:16.20.2-8.el9.noarch

Upgraded:
  nodejs-docs-2:20.19.5-1nodesource.noarch
Installed:
  nodejs-2:20.19.5-1nodesource.x86_64

Complete!
[root@localhost TeamavailTest]# npm install

up to date, audited 67 packages in 969ms

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

found 0 vulnerabilities
[root@localhost TeamavailTest]#
```



2. Write a Bash Script (ci.sh) This script should:

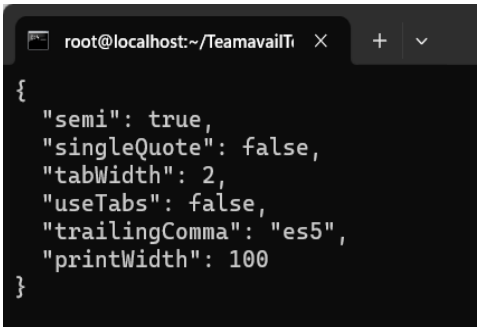
- Run code formatting and linting.

In this step we need some tools for formatting , the tool we will use it Prettier cuz its works with js, html, CSS

npm install --save-dev prettier

```
[root@localhost TeamavailTest]# npm install --save-dev prettier

added 1 package, and audited 68 packages in 3s

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

found 0 vulnerabilities
```

Adding file .prettierrc يحدد إزاي Prettier يظبط التنسيق في كل الملفات (.js, .html, .css, .json إلخ).



Adding this 2 lines in package.json in "scripts":

"format": "prettier --write \"**/*.{js,html,css,json}\"" >> to edit

"format:check": "prettier --check \"**/*.{js,html,css,json}\"" >> to check

```
{
  "name": "version-1",
  "version": "1.0.0",
  "main": "script.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js",
    "format": "prettier --write \"**/*.{js,html,css,json}\"",
    "format:check": "prettier --check \"**/*.{js,html,css,json}\""
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "express": "^5.1.0"
  },
  "devDependencies": {
    "prettier": "^3.6.2"
  }
}
```

Now u can run check, edit if u want:

npm run format >> to edit on all the project

npx prettier --write public/script.js >>to edit to a specific files

```
[root@localhost TeamavailTest]# npm run format:check

> version-1@1.0.0 format:check
> prettier --check "**/*.{js,html,css,json}"

Checking formatting...
[warn] input/selection.json
[warn] input/status.json
[warn] public/index.html
[warn] public/script.js
[warn] public/styles.css
[warn] server.js
[warn] Code style issues found in 6 files. Run Prettier with --write to fix.
```

Now for linting , the tool we will use is ESLint for js, Stylelint for css, HTMLHint for html

npm install --save-dev eslint stylelint stylelint-config-standard htmlhint

npx eslint –init

```
[root@localhost TeamavailTest]# npx eslint --init
You can also run this command directly using 'npm init @eslint/config@latest'.
Need to install the following packages:
@eslint/create-config@1.10.0
Ok to proceed? (y) y


> version-1@1.0.0 npx
> create-config

@eslint/create-config: v1.10.0

✔What do you want to lint? · javascript
✔How would you like to use ESLint? · problems
✔What type of modules does your project use? · commonjs
✔Which framework does your project use? · none
✔Does your project use TypeScript? · No / Yes
✔Where does your code run? · browser
The config that you've selected requires the following dependencies:
```

```
[root@localhost TeamavailTest]# npm install --save-dev eslint stylelint stylelint-config-standard htmlhint

added 198 packages, changed 1 package, and audited 266 packages in 27s

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

found 0 vulnerabilities
[root@localhost TeamavailTest]#
```

Then adding those files

```
[root@localhost TeamavailTest]# cat .stylelintrc.json
{
  "extends": "stylelint-config-standard"
}

[root@localhost TeamavailTest]# cat .htmlhintrc
{
  "tagname-lowercase": true,
  "attr-lowercase": true,
  "attr-value-double-quotes": true,
  "doctype-first": false,
  "id-unique": true,
  "head-script-disabled": false
}

[root@localhost TeamavailTest]#
```

Now adding those lines

"lint:js": "eslint . --ext .js",

"lint:css": "stylelint \"**/*.css\"",

"lint:html": "htmlhint \"**/*.html\"",

"lint": "npm run lint:js && npm run lint:css && npm run lint:html"

- **Run tests.**

We have integration test and unit test

In this step we will use unittest on js and the tool we will use is jest and supertest

npm install --save-dev jest supertest

now adding those 2 lines in package.js

"test:unit": "jest *.unit.test.js",

"test:integration": "jest *.integration.test.js"

I faced a problem that there is a proves uses port 3000 and when I keill it it recreated again so I decided to see the pid and see what make it run automatically and then disable it

ps -fp PID

systemctl list-units | grep my_node_app

systemctl status my_node_app

systemctl stop my_node_app

now after we solve it lets complete our steps

create this file server.unit.test.js for unittest

```
const fs = require('fs');
const path = require('path');
const { saveHistoryToFile, readHistoryFromFile } = require('./server');

const tempFile = path.join(__dirname, 'tempHistory.json');

afterAll(() => {
  if (fs.existsSync(tempFile)) fs.unlinkSync(tempFile);
});

test('saveHistoryToFile should save data correctly', () => {
  const data = { a: 1 };
  const result = saveHistoryToFile(tempFile, data);
  expect(result).toBe(true);
  expect(fs.existsSync(tempFile)).toBe(true);
});

test('readHistoryFromFile should read saved data', () => {
  const data = readHistoryFromFile(tempFile);
  expect(data).toEqual({ a: 1 });
});
```

And this file server.integration.test.js for integration test

```
const request = require('supertest');
const { app } = require('./server');

describe('Integration tests for Express app', () => {
  test('GET / should return index.html', async () => {
    const res = await request(app).get('/');
    expect([200, 404]).toContain(res.statusCode);
  });

  test('POST /save-history should save data and return 200', async () => {
    const mockData = { test: 'value' };
    const res = await request(app)
      .post('/save-history')
      .send(mockData)
      .set('Accept', 'application/json');
    expect([200, 500]).toContain(res.statusCode);
  });

  test('GET unknown route should return 404', async () => {
```

```
    const res = await request(app).get('/unknown');
    expect(res.statusCode).toBe(404);
  });
});
```

And now we need to edit server.js to be compatible with unittest & integration test

```
const express = require("express");
const fs = require("fs");
const path = require("path");
const bodyParser = require("body-parser");

const app = express();
const PORT = 3000;

// Middleware
app.use(bodyParser.json());

// Serve static frontend
app.use(express.static(path.join(__dirname, "public")));
app.use("/input", express.static(path.join(__dirname, "input")));
app.use("/output", express.static(path.join(__dirname, "output")));

// ============================
// Functions for Unit Testing
// ============================

/**
 * Save history object to file
 * @param {string} filePath
 * @param {Object} data
 */
function saveHistoryToFile(filePath, data) {
  const json = JSON.stringify(data, null, 2);
  fs.writeFileSync(filePath, json, "utf8");
  return true;
}

/**
 * Read history object from file
 * @param {string} filePath
 * @returns {Object}
 */
function readHistoryFromFile(filePath) {
  if (!fs.existsSync(filePath)) return {};
```

```
  const raw = fs.readFileSync(filePath, "utf8");
  return JSON.parse(raw);
}

// ============================
// Routes
// ============================

app.post("/save-history", (req, res) => {
  try {
    const historyPath = path.join(__dirname, "output", "history.json");
    saveHistoryToFile(historyPath, req.body);
    res.status(200).send("Saved");
  } catch (err) {
    console.error(err);
    res.status(500).send("Failed to save history.json");
  }
});

// Start server only if running directly
if (require.main === module) {
  app.listen(PORT, () => {
    console.log(`Server running at http://localhost:${PORT}`);
  });
}

// ============================
// Export for Testing
// ============================

module.exports = {
  app,                    // for integration tests
  saveHistoryToFile,      // for unit tests
  readHistoryFromFile,    // for unit tests
};
```

Now when u run npm run test:unit & npm run test:integration

```
[root@localhost TeamavailTest]# npm run test:integration

> version-1@1.0.0 test:integration
> jest *.integration.test.js

 PASS  ./server.integration.test.js
  Integration tests for Express app
    ✓ GET / should return index.html (68 ms)
    ✓ POST /save-history should save data and return 200 (23 ms)
    ✓ GET unknown route should return 404 (8 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        0.513 s, estimated 1 s
Ran all test suites matching server.integration.test.js.
[root@localhost TeamavailTest]# npm run test:unit

> version-1@1.0.0 test:unit
> jest *.unit.test.js

 PASS  ./server.unit.test.js
  ✓ saveHistoryToFile should save data correctly (3 ms)
  ✓ readHistoryFromFile should read saved data (1 ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.371 s, estimated 1 s
Ran all test suites matching server.unit.test.js.
[root@localhost TeamavailTest]# ^C
[root@localhost TeamavailTest]#
```

- Build a Docker image of the application.
- Start the application using Docker Compose.

When I start to write ci.sh and run it I faced a problem that in npm run lint:js I see

```
> version-1@1.0.0 lint:js
> eslint . --ext .js


/root/Building-a-CI-CD-Pipeline-for-the-Availability-
Tracker/server.integration.test.js
   4:1  error  'describe' is not defined  no-undef
   5:3  error  'test' is not defined      no-undef
   7:5  error  'expect' is not defined    no-undef
```

```
  10:3  error  'test' is not defined      no-undef
  16:5  error  'expect' is not defined    no-undef
  19:3  error  'test' is not defined      no-undef
  21:5  error  'expect' is not defined    no-undef

/root/Building-a-CI-CD-Pipeline-for-the-Availability-Tracker/server.unit.test.js
   7:1  error  'afterAll' is not defined  no-undef
  11:1  error  'test' is not defined      no-undef
  14:3  error  'expect' is not defined    no-undef
  15:3  error  'expect' is not defined    no-undef
  18:1  error  'test' is not defined      no-undef
  20:3  error  'expect' is not defined    no-undef

✖ 13 problems (13 errors, 0 warnings)
```

And this because of the eslint see the test files as a read files and u need to tell him those files for jest test by adding this section in eslint.config

```
{
    files: ["**/*.test.js", "**/*.spec.js"],
    languageOptions: {
      globals: {
        ...globals.node,
        ...globals.jest,
      },
    },
  },
```

Now we have ci.sh

```
#!/bin/bash
set -e

# ==============================
# Helper functions
# ==============================

function command_exists() {
  command -v "$1" >/dev/null 2>&1
}

function install_npm_package() {
  PACKAGE=$1
  if ! npm list -g "$PACKAGE" >/dev/null 2>&1; then
```

```
    echo "Installing $PACKAGE globally..."
    npm install -g "$PACKAGE"
  else
    echo "$PACKAGE is already installed globally."
  fi
}

function install_docker() {
  if ! command_exists docker; then
    echo "Docker not found. Installing..."
    # Example for Ubuntu
    curl -fsSL https://get.docker.com -o get-docker.sh
    sh get-docker.sh
    rm get-docker.sh
    sudo usermod -aG docker "$USER"
  else
    echo "Docker already installed."
  fi
}

function install_docker_compose() {
  if ! command_exists docker-compose; then
    echo "Docker Compose not found. Installing..."
    sudo curl -L
"https://github.com/docker/compose/releases/download/v2.28.0/docker-compose-
$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
    sudo chmod +x /usr/local/bin/docker-compose
  else
    echo "Docker Compose already installed."
  fi
}

# ==============================
# Step 1: Install tools if missing (idempotent)
# ==============================

echo "Checking required tools..."

# Install global tools if missing
install_npm_package "eslint"
install_npm_package "prettier"
install_npm_package "stylelint"
install_npm_package "htmlhint"
install_docker
install_docker_compose
```

```
# Install dev dependencies if missing
if [ ! -d "node_modules" ]; then
  echo "Installing npm dependencies..."
  npm install
else
  echo "Dependencies already installed."
fi

# Initialize ESLint config if missing
if [ ! -f ".eslintrc.js" ] && [ ! -f ".eslintrc.json" ] && [ ! -f
"eslint.config.mjs" ]; then
  echo "Initializing ESLint configuration..."
  npx eslint --init --yes
else
  echo "ESLint configuration already exists."
fi

# Install Jest and Supertest if missing
if ! npm list jest >/dev/null 2>&1; then
  echo "Installing Jest and Supertest..."
  npm install --save-dev jest supertest
else
  echo "Jest and Supertest already installed."
fi

echo "All required tools are installed."

# ==============================
# Step 2: Code formatting and linting (idempotent)
# ==============================

echo "Running code quality checks..."

# Run format check (read-only, idempotent)
echo "Checking code formatting with Prettier..."
npm run format:check || echo "Format issues found - continuing with other
checks..."

# Run linting checks
echo "Running ESLint..."
npm run lint:js || echo "ESLint issues found - continuing with other checks..."

echo "Running Stylelint..."
```

```
npm run lint:css || echo "Stylelint issues found - continuing with other
checks..."

echo "Running HTMLHint..."
npm run lint:html || echo "HTMLHint issues found - continuing with other
checks..."

echo "All code quality checks passed."

# ==============================
# Step 3: Run tests (idempotent)
# ==============================

echo "Running Unit and Integration Tests in parallel..."
(npm run test:unit || echo "Unit tests failed - continuing with other checks...")
&
(npm run test:integration || echo "Integration tests failed - continuing with
other checks...") &
wait

echo "All tests completed."

# ==============================
# Step 4: Build Docker image
# ==============================

IMAGE_NAME="teamavail:latest"

echo "Building Docker image..."
docker build -t $IMAGE_NAME . || echo "Docker image build failed - continuing
with other steps..."

# ==============================
# Step 5: Start application with Docker Compose
# ==============================

echo "Starting application with Docker Compose..."
docker-compose up -d || echo "Docker Compose start failed - continuing with other
steps..."

echo "CI script completed successfully!"
```

3. Dockerize the App

- Write a Dockerfile to build the application image.
- Use best practices for image building (e.g., smaller base images, clean, layers).

To make the requirements and fit it to have a too small image as we can get , we need to first not take everything in the image like the input will be taken because if we want to change it like names and increase names or decrease it so we will have a flexibility to do this , and second we will use multistage build.

```
# Builder: install prod deps
FROM node:20-alpine AS builder
ENV NODE_ENV=production
WORKDIR /app
COPY package*.json ./
RUN npm ci --omit=dev && npm cache clean --force
COPY --chown=node:node server.js ./
COPY --chown=node:node public/ ./public/
COPY --chown=node:node input/ ./input/
COPY --chown=node:node output/ ./output/
# Runtime: node alpine (with sh)
FROM node:20-alpine
WORKDIR /app
COPY --from=builder /app /app
ENV PORT=3000 NODE_ENV=production
EXPOSE 3000
CMD ["node", "server.js"]
```

4. Use Docker Compose

- Create a docker-compose.yml file.
- Include the app and any required services like Redis or PostgreSQL.
- Configure volumes and ports properly.

Now in this step we want to compose the app and redis image to be connected with each other , the plan is we want to to make the app save its data in redis image , but the problem is the app is save data as a .jeson file and this not compatible with redis , so we decide to make the app save the output as .jsen and then synck and change the format of the output to redis

```
version: "3.9"

services:
  app:
    image: teamavail:latest
```

```
    container_name: availability-tracker
    environment:
      - NODE_ENV=production
      - PORT=3000
    ports:
      - "3000:3000"
    volumes:
      - ./output:/app/output
    depends_on:
      - redis
    restart: unless-stopped

  redis:
    image: redis:7-alpine
    container_name: availability-redis
    ports:
      - "6379:6379"
    volumes:
      - ./redis_volume:/data
    command: ["redis-server", "--appendonly", "yes"]
    restart: unless-stopped

  history-sync:
    image: redis:7-alpine
    container_name: availability-history-sync
    depends_on:
      - app
      - redis
    volumes:
      - ./output:/app/output
    entrypoint: ["sh", "-c"]
    command: >
      "mkdir -p /app/output; \
      sleep 2; \
      r=\$(redis-cli -h redis -p 6379 GET history); \
      if [ -n \"$r\" ]; then \
        printf \"%s\" \"$r\" > /app/output/history.json; \
      else \
        if [ -f /app/output/history.json ]; then \
          cat /app/output/history.json | redis-cli -h redis -p 6379 -x SET
history >/dev/null; \
        fi; \
      fi; \
      prev_r=\"\$(printf %s \"$r\" | sha256sum | awk '{print $1}')\"; \
      if [ -f /app/output/history.json ]; then \
```

```
        prev_f=\"\$(sha256sum /app/output/history.json | awk '{print $1}')\"; \
      else \
        prev_f=\"\"; \
      fi; \
      while :; do \
        r=\$(redis-cli -h redis -p 6379 GET history); \
        rh=\"\$(printf %s \"$r\" | sha256sum | awk '{print $1}')\"; \
        if [ -n \"$r\" ] && [ \"$rh\" != \"$prev_r\" ]; then \
          printf \"%s\" \"$r\" > /app/output/history.json; \
          prev_r=\"$rh\"; \
          prev_f=\"\$(sha256sum /app/output/history.json | awk '{print $1}')\"; \
        fi; \
        if [ -f /app/output/history.json ]; then \
          fh=\"\$(sha256sum /app/output/history.json | awk '{print $1}')\"; \
          if [ \"$fh\" != \"$prev_f\" ]; then \
            cat /app/output/history.json | redis-cli -h redis -p 6379 -x SET
history >/dev/null; \
            prev_f=\"$fh\"; \
            prev_r=\"\$(printf %s \"\$(cat /app/output/history.json)\" |
sha256sum | awk '{print $1}')\"; \
          fi; \
        fi; \
        sleep 2; \
      done"
    restart: unless-stopped

volumes:
  redis-data:
```

Optional Extensions

Instead of using ci.sh we can use Jenkins , u just need to install it as an image and run it

U need to make custom image for Jenkins

Here I faced a problem when I run
docker run -d -p 9090:8080 -p 50000:50000 -v jenkins_home:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock jenkins-custom:latest
it said u run out of space , so I decide to expand the size in vm then resize it

```
[root@localhost ~]# sudo fdisk /dev/sda

Welcome to fdisk (util-linux 2.37.4).
```

```
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

This disk is currently in use - repartitioning is probably a bad idea.
It's recommended to umount all file systems, and swapoff all swap
partitions on this disk.


Command (m for help): p

Disk /dev/sda: 40 GiB, 42949672960 bytes, 83886080 sectors
Disk model: VMware Virtual S
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xa228537f

Device     Boot   Start      End  Sectors Size Id Type
/dev/sda1  *       2048  2099199  2097152   1G 83 Linux
/dev/sda2       2099200 41943039 39843840  19G 8e Linux LVM

Command (m for help): d
Partition number (1,2, default 2): 2

Partition 2 has been deleted.

Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (2-4, default 2): 2
First sector (2099200-83886079, default 2099200):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2099200-83886079, default
83886079):

Created a new partition 2 of type 'Linux' and of size 39 GiB.
Partition #2 contains a LVM2_member signature.

Do you want to remove the signature? [Y]es/[N]o: N

Command (m for help): t
Partition number (1,2, default 2): 2
Hex code or alias (type L to list all): L
```

```
00 Empty            24 NEC DOS          81 Minix / old Lin  bf Solaris
01 FAT12            27 Hidden NTFS Win  82 Linux swap / So  c1 DRDOS/sec (FAT-
02 XENIX root       39 Plan 9           83 Linux            c4 DRDOS/sec (FAT-
03 XENIX usr        3c PartitionMagic   84 OS/2 hidden or   c6 DRDOS/sec (FAT-
04 FAT16 <32M       40 Venix 80286      85 Linux extended   c7 Syrinx
05 Extended         41 PPC PReP Boot    86 NTFS volume set  da Non-FS data
06 FAT16            42 SFS              87 NTFS volume set  db CP/M / CTOS / .
07 HPFS/NTFS/exFAT  4d QNX4.x           88 Linux plaintext  de Dell Utility
08 AIX              4e QNX4.x 2nd part  8e Linux LVM        df BootIt
09 AIX bootable     4f QNX4.x 3rd part  93 Amoeba           e1 DOS access
0a OS/2 Boot Manag  50 OnTrack DM       94 Amoeba BBT       e3 DOS R/O
0b W95 FAT32        51 OnTrack DM6 Aux  9f BSD/OS           e4 SpeedStor
0c W95 FAT32 (LBA)  52 CP/M             a0 IBM Thinkpad hi  ea Linux extended
0e W95 FAT16 (LBA)  53 OnTrack DM6 Aux  a5 FreeBSD          eb BeOS fs
0f W95 Ext'd (LBA)  54 OnTrackDM6       a6 OpenBSD          ee GPT
10 OPUS             55 EZ-Drive         a7 NeXTSTEP         ef EFI (FAT-12/16/
11 Hidden FAT12     56 Golden Bow       a8 Darwin UFS       f0 Linux/PA-RISC b
12 Compaq diagnost  5c Priam Edisk      a9 NetBSD           f1 SpeedStor
14 Hidden FAT16 <3  61 SpeedStor        ab Darwin boot      f4 SpeedStor
16 Hidden FAT16     63 GNU HURD or Sys  af HFS / HFS+       f2 DOS secondary
17 Hidden HPFS/NTF  64 Novell Netware   b7 BSDI fs          fb VMware VMFS
18 AST SmartSleep   65 Novell Netware   b8 BSDI swap        fc VMware VMKCORE
1b Hidden W95 FAT3  70 DiskSecure Mult  bb Boot Wizard hid  fd Linux raid auto
1c Hidden W95 FAT3  75 PC/IX            bc Acronis FAT32 L  fe LANstep
1e Hidden W95 FAT1  80 Old Minix        be Solaris boot     ff BBT

Aliases:
   linux          - 83
   swap           - 82
   extended       - 05
   uefi           - EF
   raid           - FD
   lvm            - 8E
   linuxex        - 85
Hex code or alias (type L to list all): 8E

Changed type of partition 'Linux' to 'Linux LVM'.

Command (m for help): w
The partition table has been altered.
Syncing disks.

[root@localhost ~]# sudo partprobe /dev/sda
[root@localhost ~]# lsblk
```

```
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda           8:0    0    40G  0 disk
├─sda1        8:1    0     1G  0 part /boot
└─sda2        8:2    0    39G  0 part
  ├─cs-root 253:0    0    17G  0 lvm  /
  └─cs-swap 253:1    0     2G  0 lvm  [SWAP]
sr0          11:0    1 157.8M  0 rom
sr1          11:1    1  10.6G  0 rom
[root@localhost ~]# sudo pvresize /dev/sda2
  Physical volume "/dev/sda2" changed
  1 physical volume(s) resized or updated / 0 physical volume(s) not resized
[root@localhost ~]# sudo lvextend -l +100%FREE /dev/mapper/cs-root
  Size of logical volume cs/root changed from <17.00 GiB (4351 extents) to <37.00
GiB (9471 extents).
  Logical volume cs/root successfully resized.
[root@localhost ~]# sudo xfs_growfs /
meta-data=/dev/mapper/cs-root    isize=512    agcount=4, agsize=1113856 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=1 inobtcount=1 nrext64=0
data     =                       bsize=4096   blocks=4455424, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=16384, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 4455424 to 9698304
[root@localhost ~]# df -h /
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/cs-root   37G   17G   21G  46% /
[root@localhost ~]#
```

Then u can docker run -d -p 9090:8080 -p 50000:50000 -v jenkins_home:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock jenkins-custom:latest
but u need to write and build the custom Jenkins , and this custom image to install all requirement u need to run ur pipline

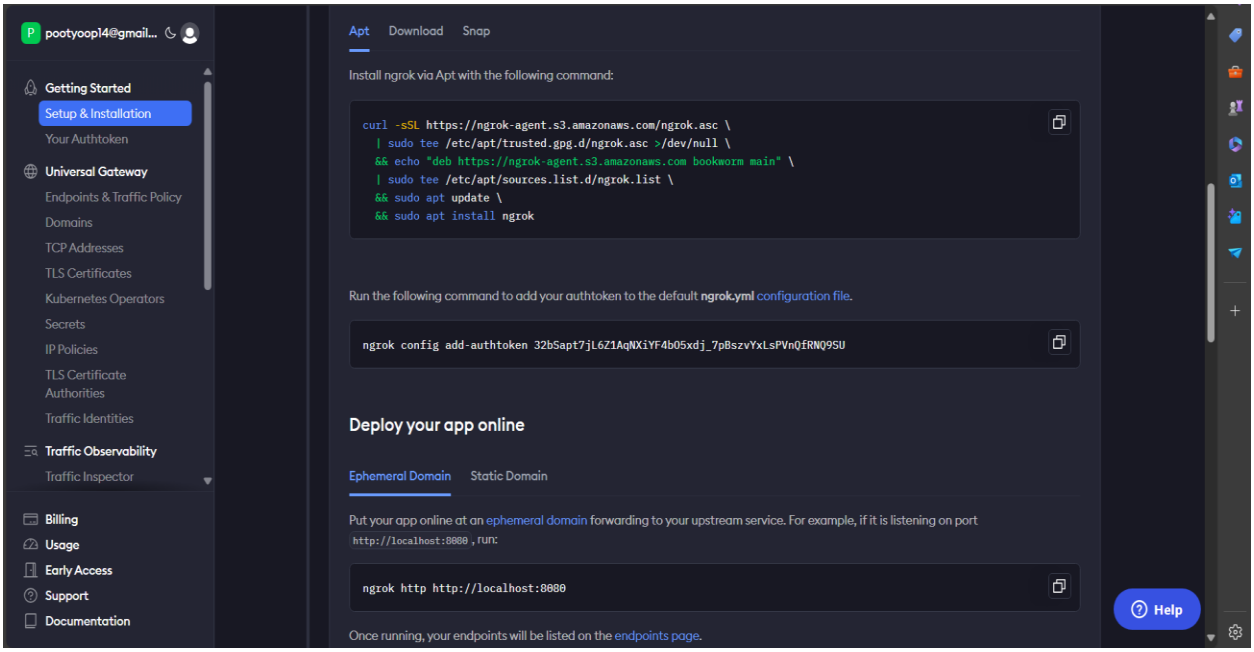
```
FROM jenkins/jenkins:lts

USER root

# Install dependencies
RUN apt-get update && apt-get install -y \
    curl \
    unzip \
```

```
    python3 \
    python3-pip \
    software-properties-common \
    gnupg \
    sshpass \
    git \
    wget \
    ca-certificates \
    apt-transport-https \
    lsb-release \
    && rm -rf /var/lib/apt/lists/*

# Install AWS CLI v2
RUN curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip" \
    && unzip awscliv2.zip \
    && ./aws/install \
    && rm -rf awscliv2.zip aws

# Install Terraform (download binary directly)
RUN curl -fsSL
https://releases.hashicorp.com/terraform/1.9.5/terraform_1.9.5_linux_amd64.zip -o
terraform.zip \
    && unzip terraform.zip \
    && mv terraform /usr/local/bin/ \
    && rm terraform.zip

# Install Ansible (via apt for stability)
RUN apt-get update && apt-get install -y ansible \
    && rm -rf /var/lib/apt/lists/*

# Install Docker CLI (download binary)
RUN curl -fsSL https://download.docker.com/linux/static/stable/x86_64/docker-
25.0.3.tgz -o docker.tgz \
    && tar xzvf docker.tgz --strip 1 -C /usr/local/bin docker/docker \
    && rm docker.tgz

# Install Node.js 20 (for CI pipeline)
RUN curl -fsSL https://deb.nodesource.com/setup_20.x | bash - \
    && apt-get install -y nodejs

# Install Docker Compose (for CI pipeline)
RUN curl -L "https://github.com/docker/compose/releases/download/v2.28.0/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose \
    && chmod +x /usr/local/bin/docker-compose
```

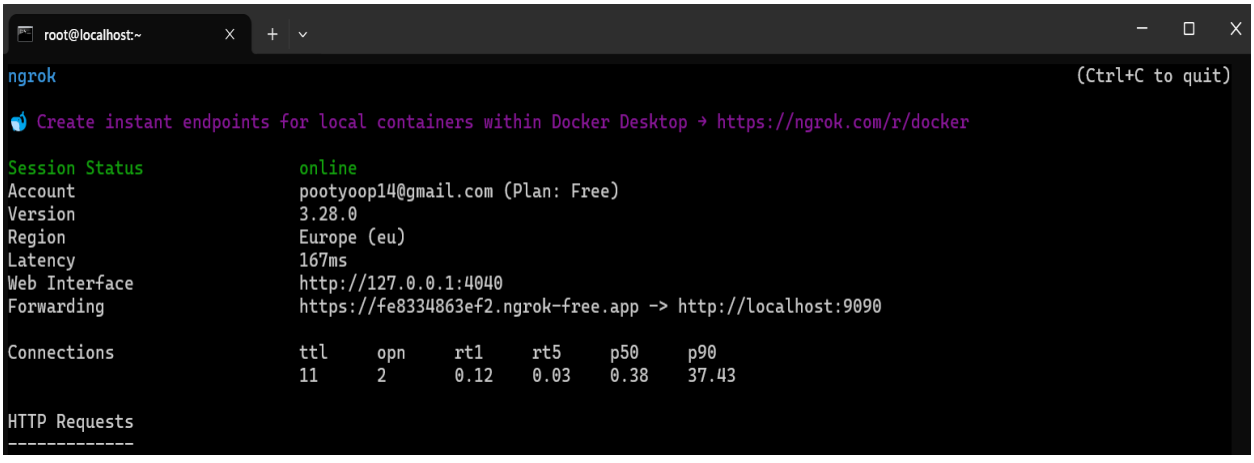
```
# Create docker group and add jenkins user to it
RUN groupadd docker && usermod -aG docker jenkins

# Switch to jenkins user for npm global installs
USER jenkins

# Configure npm to use a global directory accessible to jenkins user
RUN mkdir -p /var/jenkins_home/.npm-global \
    && npm config set prefix '/var/jenkins_home/.npm-global'

# Add npm global bin to PATH for jenkins user
ENV PATH="/var/jenkins_home/.npm-global/bin:$PATH"

# Install global npm packages (for CI pipeline)
RUN npm install -g eslint prettier stylelint htmlhint jest supertest

# Install Jenkins plugins
RUN jenkins-plugin-cli --plugins \
    workflow-aggregator \
    git \
    docker-workflow \
    docker-plugin \
    blueocean \
    pipeline-stage-view \
    build-timeout \
    credentials-binding \
    timestamper \
    ws-cleanup \
    ant \
    gradle \
    workflow-job \
    ssh-slaves \
    matrix-auth \
    pam-auth \
    ldap \
    email-ext \
    mailer \
    aws-credentials \
    terraform \
    ansible

# Set environment variables
ENV JAVA_OPTS="-Djenkins.install.runSetupWizard=false"
ENV JENKINS_OPTS="--httpPort=8080"
```

```
# Expose ports
EXPOSE 8080 50000

# Health check
HEALTHCHECK --interval=30s --timeout=10s --start-period=60s --retries=3 \
    CMD curl -f http://localhost:8080/login || exit 1
```

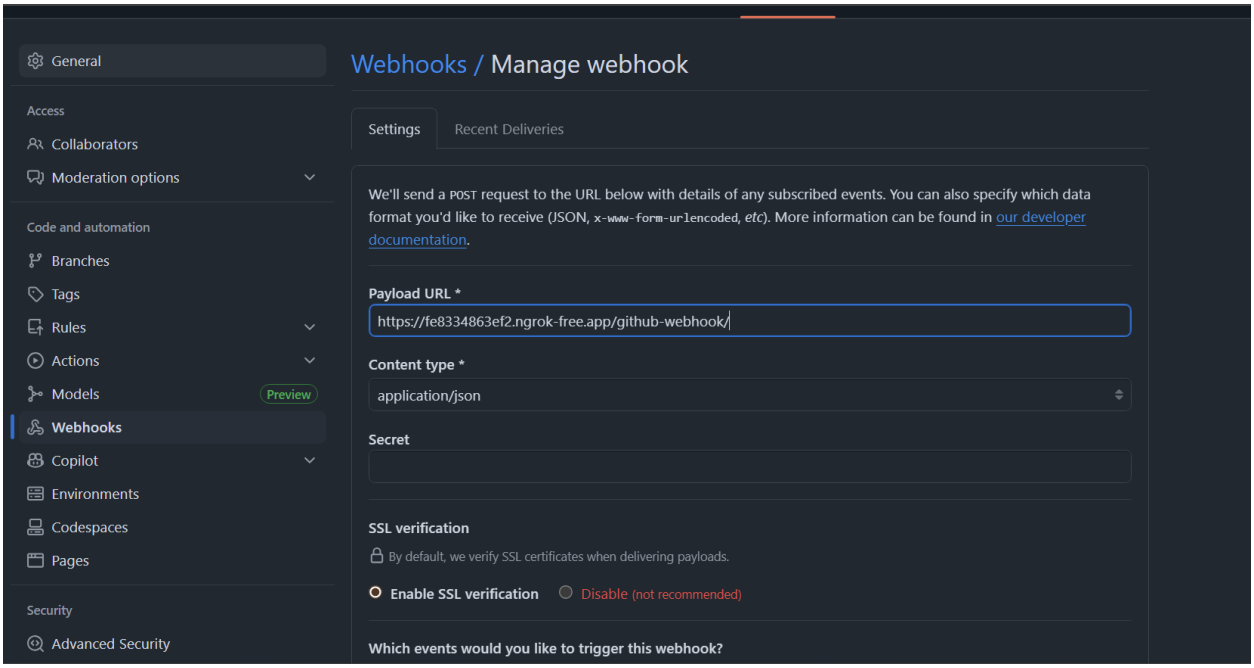
and to connect between github and Jenkins , u can use webhook and cuz we are in lockal environment so u can use ngrok to make it publick and github see u

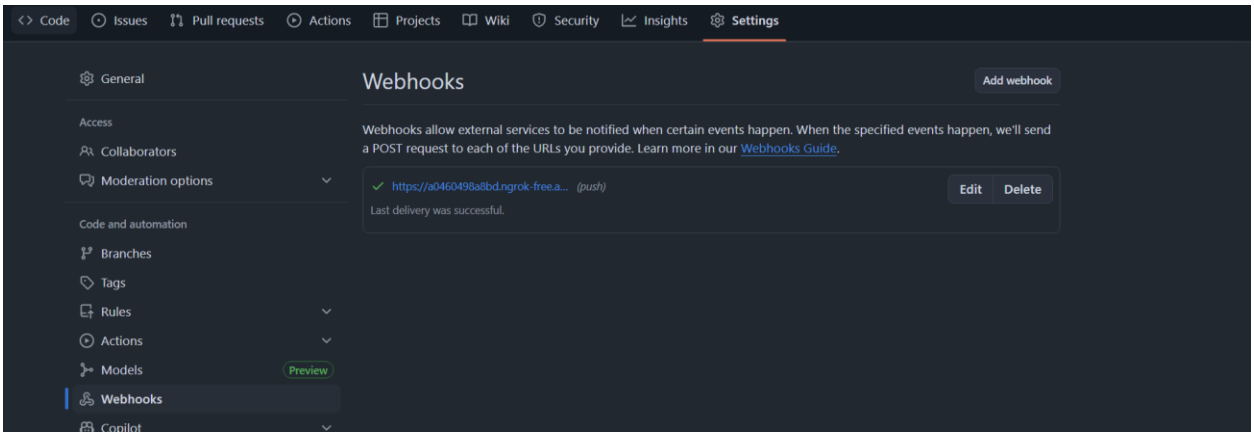


Then run this [root@localhost ~]# ngrok http 9090
and now



Now u can manage webhook in github





Then u need to make a Jenkins pipeline

```
pipeline {
    agent any

    environment {
        IMAGE_NAME = 'teamavail:latest'
        NODE_VERSION = '20'
    }

    stages {
        stage('Checkout') {
            steps {
                checkout scm
                script {
```

```
                    echo "Checked out code from ${env.GIT_URL}"
                }
            }
        }

        stage('Install Dependencies') {
            steps {
                script {
                    echo "Installing Node.js dependencies..."
                    sh '''
                        # Verify tools are available (pre-installed in custom
Jenkins image)
                        echo "Node.js version: $(node --version)"
                        echo "npm version: $(npm --version)"
                        echo "Docker version: $(docker --version)"
                        echo "Docker Compose version: $(docker-compose --
version)"

                        # Install npm dependencies
                        if [ ! -d "node_modules" ]; then
                            echo "Installing npm dependencies..."
                            npm install
                        else
                            echo "Dependencies already installed."
                        fi

                        # Verify global tools are available
                        echo "Global tools available:"
                        for tool in eslint prettier stylelint htmlhint jest
supertest; do
                            if command -v "$tool" >/dev/null 2>&1; then
                                echo "✓ $tool: $($tool --version 2>/dev/null ||
echo 'installed')"
                            else
                                echo "✗ $tool: not found"
                            fi
                        done

                        # Initialize ESLint config if missing
                        if [ ! -f ".eslintrc.js" ] && [ ! -f ".eslintrc.json" ]
&& [ ! -f "eslint.config.mjs" ]; then
                            echo "Initializing ESLint configuration..."
                            npx eslint --init --yes
                        else
                            echo "ESLint configuration already exists."
```

```
                        fi

                        # Install Jest and Supertest if missing
                        if ! npm list jest >/dev/null 2>&1; then
                            echo "Installing Jest and Supertest..."
                            npm install --save-dev jest supertest
                        else
                            echo "Jest and Supertest already installed."
                        fi
                    '''
                }
            }
        }

        stage('Code Quality Checks') {
            parallel {
                stage('Format Check') {
                    steps {
                        script {
                            echo "Checking code formatting with Prettier..."
                            sh 'npm run format:check || echo "Format issues found
- continuing with other checks..."'
                        }
                    }
                }

                stage('ESLint') {
                    steps {
                        script {
                            echo "Running ESLint..."
                            sh 'npm run lint:js || echo "ESLint issues found -
continuing with other checks..."'
                        }
                    }
                }

                stage('Stylelint') {
                    steps {
                        script {
                            echo "Running Stylelint..."
                            sh 'npm run lint:css || echo "Stylelint issues found
- continuing with other checks..."'
                        }
                    }
                }
```

```
                stage('HTMLHint') {
                    steps {
                        script {
                            echo "Running HTMLHint..."
                            sh 'npm run lint:html || echo "HTMLHint issues found
- continuing with other checks..."'
                        }
                    }
                }
            }
        }

        stage('Tests') {
            parallel {
                stage('Unit Tests') {
                    steps {
                        script {
                            echo "Running Unit Tests..."
                            sh 'npm run test:unit || echo "Unit tests failed -
continuing with other checks..."'
                        }
                    }
                }

                stage('Integration Tests') {
                    steps {
                        script {
                            echo "Running Integration Tests..."
                            sh 'npm run test:integration || echo "Integration
tests failed - continuing with other checks..."'
                        }
                    }
                }
            }
        }

        stage('Docker Build') {
            steps {
                script {
                    echo "Building Docker image..."
                    sh '''
                        # Verify Docker tools are available (pre-installed in
custom Jenkins image)
                        echo "Docker version: $(docker --version)"
```

```
                        echo "Docker Compose version: $(docker-compose --
version)"

                        # Build Docker image
                        docker build -t ${IMAGE_NAME} . || echo "Docker image
build failed - continuing with other steps..."
                    '''
                }
            }
        }

        stage('Docker Compose Deploy') {
    steps {
        script {
            echo "Debugging Docker Compose mount before starting..."
            dir("${env.WORKSPACE}") {
                sh '''
                    set -x
                    echo "WORKSPACE: $(pwd)"
                    chown -R 1000:1000
/var/jenkins_home/workspace/AvailabilityTracker/input
                    chmod -R 777
/var/jenkins_home/workspace/AvailabilityTracker/input
                    chmod -R 777
/var/jenkins_home/workspace/AvailabilityTracker/output
                    echo "Listing workspace root:"
                    ls -la

                    echo "Listing input directory:"
                    [ -d input ] && ls -la input || echo "input/ missing"

                    echo "Listing output directory:"
                    [ -d output ] && ls -la output || echo "output/ missing"

                    echo "Checking ownership and permissions:"
                    stat input
                    stat output

                    # Fix permissions if needed
                    chmod -R 777 input output
                    echo "Permissions fixed."

                    # Show mounts in Docker
                    docker info | grep "Docker Root Dir"
```

```
                    # Compose down/up
                    COMPOSE_FILE=$(find . -maxdepth 2 -type f -name docker-
compose.yml | head -n1)
                    if [ -z "$COMPOSE_FILE" ]; then
                        echo "docker-compose.yml not found" >&2
                        exit 1
                    fi
                    COMPOSE_DIR=$(dirname "$COMPOSE_FILE")
                    cd "$COMPOSE_DIR"
                    docker-compose down -v || true
                    docker-compose up -d
                '''
            }
        }
    }
}
    }

    post {
        always {
            script {
                echo "Pipeline completed!"
                // Clean up workspace if needed
                sh '''
                    echo "Cleaning up temporary files..."
                    # Add any cleanup commands here
                '''
            }
        }

        success {
            script {
                echo "Pipeline succeeded! Application is running."
                // You can add notifications here (email, Slack, etc.)
            }
        }

        failure {
            script {
                echo "Pipeline failed! Check the logs for details."
                // You can add failure notifications here
            }
        }

        unstable {
```

```
            script {
                echo "Pipeline is unstable! Some checks failed but pipeline
continued."
            }
        }
    }
}
```

Then u will see the ut app is running