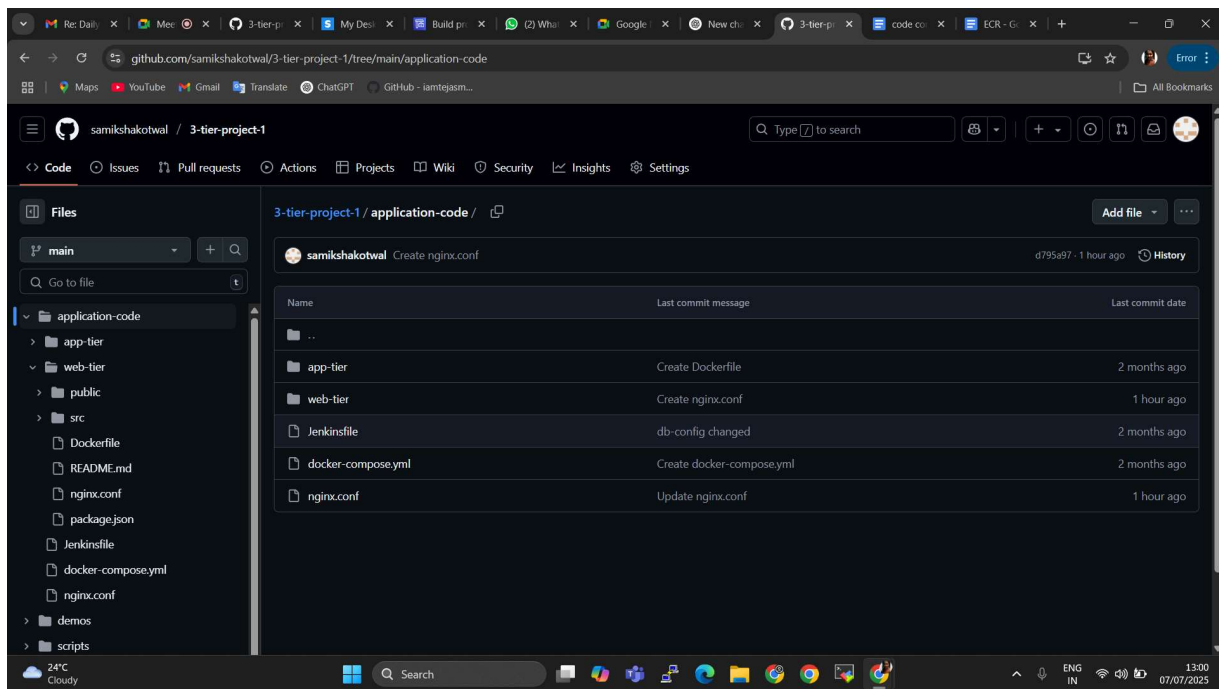


# Code Commit (build project)

## 1. Create Project Files

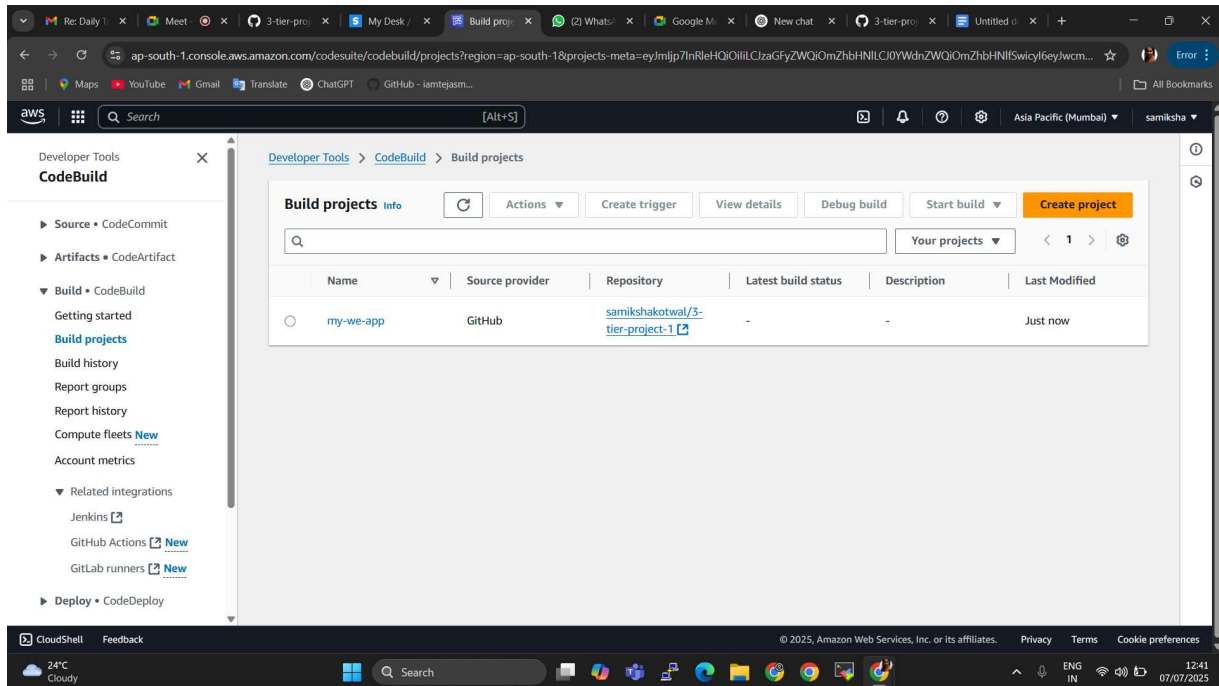
- <https://github.com/samikshakotwal/3-tier-project-1.git>



## 2. Create a CodeBuild Project

- Go to AWS Console > CodeBuild > Create Build Project.
- Project Name: **my-web-app**
- Source Provider: AWS CodeCommit
- Repository: Choose your repo
  - Choose: **GitHub**
  - Click "Connect to GitHub"
- (Login and authorize AWS)
- Environment:
  - OS: Amazon Linux 2
  - Runtime: Choose your runtime (e.g., [Node.js](#))
  - Privileged: Checked if using Docker

- Service Role: Create new or choose existing IAM role
- Buildspec: Use buildspec.yml from source
- Artifacts: No artifacts or S3 (optional)
- Click “**Create Build Project**”.



### 3. Create **buildspec.yml** in Github

```
version: 0.2
phases:
  install:
    commands:
      - echo "Installing frontend and backend via Docker builds..."
      - sudo yum install nginx -y
      - docker build -t frontend-image ./application-code/web-tier
      - docker build -t backend-image ./application-code/app-tier
  build:
    commands:
      - echo "Running Docker containers..."
      - docker run -d --name frontend-container -p 81:80 frontend-image
      - docker run -d --name backend-container -p 4000:4000 backend-image
      - echo "Docker containers started successfully."
artifacts:
```

files:

- appspec.yml
- scripts/\*
- application-code/web-tier/Dockerfile
- application-code/app-tier/Dockerfile

```

1  version: 0.2
2  phases:
3    install:
4      commands:
5        - echo "Installing frontend and backend via Docker builds..."
6        - sudo yum install nginx -y
7        - docker build -t frontend-image ./application-code/web-tier
8        - docker build -t backend-image ./application-code/app-tier
9
10   build:
11     commands:
12       - echo "Running Docker containers..."
13       - docker run -d --name frontend-container -p 81.80 frontend-image
14       - docker run -d --name backend-container -p 4000:4000 backend-image
15       - echo "Docker containers started successfully."
16
17   artifacts:
18     files:
19       - appspec.yml
20       - scripts/*
21       - application-code/web-tier/Dockerfile
22       - application-code/app-tier/Dockerfile
  
```

#### 4. Go to application-code/web-tier/

- Create **Dockerfile**

```

# Stage 1: Build the React app
FROM public.ecr.aws/docker/library/node:18 AS builder
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
# Stage 2: Serve with NGINX
FROM nginx:stable-alpine
# Remove default config
RUN rm /etc/nginx/nginx.conf
  
```

```
# copy our custom config
COPY nginx.conf /etc/nginx
# copy build output to NGINX web root
COPY --from=builder /app/build /usr/share/nginx/html
# Expose port
EXPOSE 80
```

The screenshot shows a web browser displaying a GitHub repository for '3-tier-project-1'. The file 'nginx.conf' is selected in the 'web-tier' directory. The code content is as follows:

```
1 user nginx;
2 worker_processes auto;
3 error_log /var/log/nginx/error.log notice;
4 pid /run/nginx.pid;
5
6 include /usr/share/nginx/modules/*.conf;
7
8 events {
9     worker_connections 1024;
10 }
11
12 http {
13     log_format main '$remote_addr - $remote_user [$time_local] "$request" '
14                     '$status $body_bytes_sent "$http_referer" '
15                     '"$http_user_agent" "$http_x_forwarded_for"';
16
17     access_log /var/log/nginx/access.log main;
18
19     sendfile        on;
20     tcp_nopush      on;
21     keepalive_timeout 65;
```

- Create **nginx.conf**

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log notice;
pid /run/nginx.pid;

include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

http {
```

```
log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                '$status $body_bytes_sent "$http_referer" '
                '"$http_user_agent" "$http_x_forwarded_for"';

access_log /var/log/nginx/access.log main;

sendfile      on;
tcp_nopush    on;
keepalive_timeout 65;
types_hash_max_size 4096;

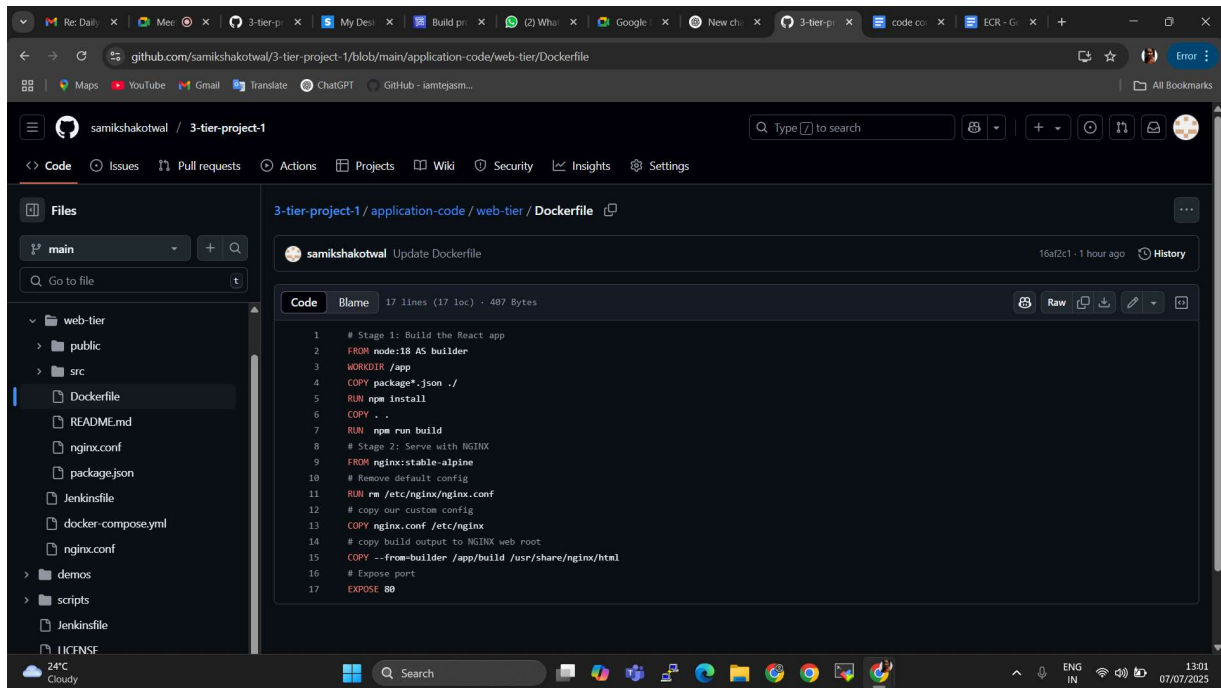
include       /etc/nginx/mime.types;
default_type  application/octet-stream;

include /etc/nginx/conf.d/*.conf;

server {
    listen 80;
    server_name localhost;

    location / {
        proxy_pass http://localhost:81; # Frontend
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    location /api/ {
        proxy_pass http://localhost:4000/; # Backend
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```



## 5. Configure Project

- Then click "Start build"
- Monitor build logs to see progress

