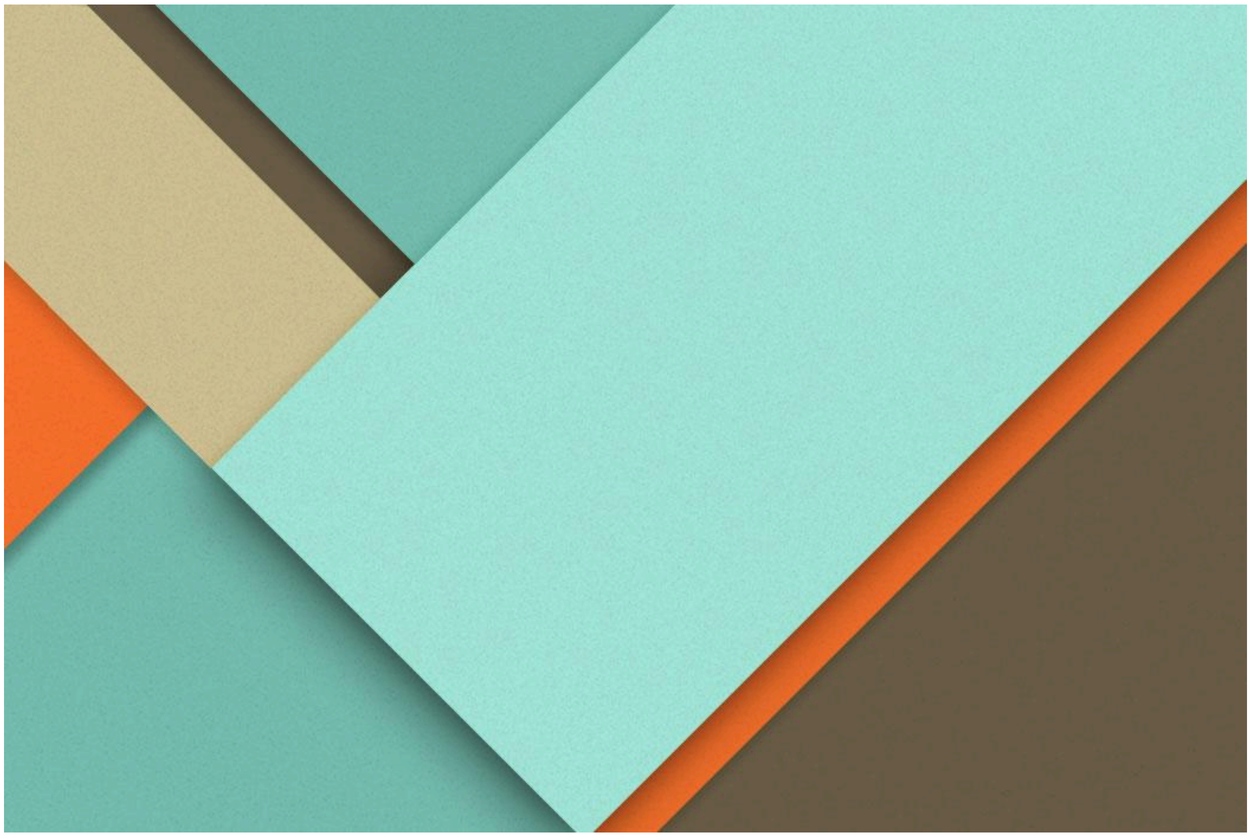

Swimmersweb

1. Overview

This document provides step-by-step instructions for deploying the **Swimmers Web** application, including:

- System prerequisites
- Docker build processes for **frontend** and **backend**
- NGINX reverse proxy and SSL setup
- Jenkins CI/CD automation

2. Prerequisites

Ensure the following components are available and configured:

- Ubuntu Linux-based virtual machine (VM)
- Docker and Docker Compose installed
- Registered domain (e.g., `swimmersweb.com`)
- SSL Certificates:
  - `swim.cert` – Domain SSL Certificate
  - `swim.key` – SSL Private Key
  - `lets-encrypt-r3.pem` – Intermediate Certificate (Let's Encrypt)
- Jenkins CI server with necessary plugins installed

3. System Setup

3.1 Install Required Packages

```
sudo apt update
sudo apt install -y docker.io nginx nano
sudo systemctl enable docker
sudo usermod -aG docker $USER
```

4. SSL Configuration

4.1 Create SSL Directory

```
sudo mkdir -p /etc/nginx/ssl
```

4.2 Place Certificate Files

Place the following files under /etc/nginx/ssl/:

swim.cert

swim.key

lets-encrypt-r3.pem

4.3 Create Full Chain Certificate

```
sudo bash -c "cat /etc/nginx/ssl/swim.cert /etc/nginx/ssl/lets-encrypt-r3.pem > /etc/nginx/ssl/swim.fullchain.cert"
```

5. Docker Setup

5.1 Frontend Dockerfile

```
# Stage 1: Build Angular App
FROM node:18-alpine AS builder
WORKDIR /app
COPY package.json package-lock.json ./
RUN npm install
COPY . .
RUN npm run build --prod
# Stage 2: Serve with NGINX
FROM nginx:1.20-alpine
COPY --from=builder /app/dist/swim-frontend/browser /usr/share/nginx/html
COPY nginx.conf /etc/nginx/conf.d/default.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

5.2 Docker NGINX File (nginx.conf)

```
server {
  listen 80;
  root /usr/share/nginx/html;
  index index.html index.htm;

  location / {
    try_files $uri $uri/ /index.html;
  }

  error_page 404 /index.html;
}
```

5.3 Backend Dockerfile

```
# Build Stage
FROM maven:3.8.6-eclipse-temurin-17 AS build
WORKDIR /opt/app
COPY ./ /opt/app
RUN mvn clean install -DskipTests
# Final Stage
FROM eclipse-temurin:17-jdk-jammy
WORKDIR /opt/app
COPY --from=build /opt/app/target/*.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "app.jar"]
```

6. NGINX Reverse Proxy & SSL Termination

6.1 NGINX Configuration with SSL

Path: `/etc/nginx/nginx.conf`

```
events {
    worker_connections 1024;
}

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

    upstream ui {
        server 34.130.201.147:8081;
        server 34.130.230.176:8081;
    }

    upstream backend {
        server 34.130.201.147:8072;
        server 34.130.230.176:8072;
    }

    sendfile on;
    keepalive_timeout 65;
    client_max_body_size 20M;

    gzip on;
    gzip_types text/plain text/css application/json application/javascript text/xml
application/xml application/xml+rss text/javascript;

    server {
        listen 80;
        server_name swimmersweb.com;
        return 301 https://$host$request_uri;
    }

    server {
        listen 443 ssl http2;
        server_name swimmersweb.com;

        ssl_certificate /etc/nginx/ssl/swim.fullchain.cert;
        ssl_certificate_key /etc/nginx/ssl/swim.key;

        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_ciphers
'TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:ECDHE-ECDSA-AES128-GCM-SHA25
6:ECDHE-RSA-AES128-GCM-SHA256';
        ssl_session_cache shared:SSL:10m;
        ssl_session_timeout 1h;
        ssl_session_tickets off;

        ssl_stapling on;
        ssl_stapling_verify on;
        resolver 1.1.1.1 8.8.8.8 valid=300s;

        add_header Strict-Transport-Security "max-age=31536000; includeSubDomains;
preload" always;
```

```
add_header X-Frame-Options "SAMEORIGIN" always;
add_header X-XSS-Protection "1; mode=block" always;
add_header X-Content-Type-Options "nosniff" always;
add_header Referrer-Policy "strict-origin-when-cross-origin" always;
add_header Permissions-Policy "geolocation=(), microphone=()" always;

location = /status {
    access_log off;
    return 200 'OK';
    add_header Content-Type text/plain;
}

location / {
    proxy_pass http://ui;
    proxy_http_version 1.1;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}

location /api/ {
    proxy_pass http://backend;
    proxy_http_version 1.1;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
```

```
            add_header Access-Control-Allow-Origin * always;
            add_header Access-Control-Allow-Methods "GET, POST, PUT, DELETE, OPTIONS"
always;
            add_header Access-Control-Allow-Headers "Origin, Content-Type, Accept,
Authorization" always;
            add_header Access-Control-Max-Age 3600 always;

            if ($request_method = OPTIONS) {
                add_header Content-Length 0;
                return 204;
            }
        }
    }
}
```

7. Reload NGINX and Validate Configuration

```
sudo nginx -t
sudo systemctl reload nginx
```

8. SSL Certificate Validation

8.1 Validate Certificate Chain

```
openssl s_client -connect swimmersweb.com:443 -servername swimmersweb.com
-showcerts
```

8.2 Verify Certificate Trust Chain

# Option A

```
openssl verify -CAfile /etc/nginx/ssl/lets-encrypt-r3.pem /etc/nginx/ssl/swim.cert
```

# Option B

```
openssl verify -CAfile /etc/nginx/ssl/swim.fullchain.cert /etc/nginx/ssl/swim.cert
```

8.3 Validate Certificate and Key Pair

```
openssl rsa -noout -modulus -in /etc/nginx/ssl/swim.key | openssl md5
openssl x509 -noout -modulus -in /etc/nginx/ssl/swim.fullchain.cert | openssl md5
```

9. Online Tools for SSL Testing

- SSL Shopper - Certificate Checker
- SSL Labs - Server Test
- HSTS Preload List Submission

10. Jenkins CI/CD Pipelines

10.1 Frontend Jenkinsfile

```
pipeline {
    agent any

    environment {
        PROJECT_NAME     = "Swim-UI-Prod"
        REGISTRY         = "35.188.22.165:5000"
        IMAGE_NAME       = "swim-ui-app"
        CONTAINER_NAME   = "swim-ui-container"
        FULL_IMAGE_NAME  = "${REGISTRY}/${IMAGE_NAME}:latest"
        REPO_URL         = "https://github.com/data-on-disk/dod-swim-ui.git"
        APP_PORT         = "80"
        SSH_USER         = "ashis"
        VM1_IP           = "34.130.230.176"
        CREDENTIALS_ID   = "gcp-ssh-key"
    }

    stages {
        stage('Clone Repository') {
            steps {
                git credentialsId: 'github-dod-credentials', url: "${REPO_URL}", branch: 'prod'
    }
  }

  stage('Build Docker Image') {
    steps {
      script {
        sh "docker build -t ${IMAGE_NAME} ."
        sh "docker tag ${IMAGE_NAME} ${FULL_IMAGE_NAME}"
      }
    }
  }

  stage('Push to Private Registry') {
    steps {
      script {
        withCredentials([usernamePassword(credentialsId: 'docker-registry-creds',
usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')]) {
          sh "docker login ${REGISTRY} -u ${DOCKER_USER} -p ${DOCKER_PASS}"
          sh "docker push ${FULL_IMAGE_NAME}"
        }
      }
    }
  }
```

```
        stage('Save Docker Image as TAR') {
            steps {
                sh "docker save -o swim-ui-app.tar ${FULL_IMAGE_NAME}"
            }
        }

        stage('Deploy to VM') {
            steps {
                script {
                    sshagent(credentials: [CREDENTIALS_ID]) {
                        sh """
                        echo "Deploying to ${VM1_IP}"
                        scp -o StrictHostKeyChecking=no swim-ui-app.tar
${SSH_USER}@${VM1_IP}:/tmp/
                        ssh -o StrictHostKeyChecking=no ${SSH_USER}@${VM1_IP} '
                            docker stop ${CONTAINER_NAME} || true
                            docker rm ${CONTAINER_NAME} || true
                            docker rmi ${FULL_IMAGE_NAME} || true
                            docker load -i /tmp/swim-ui-app.tar
                            docker run -d -p 8081:80 --restart=always --name ${CONTAINER_NAME}
${FULL_IMAGE_NAME}
                        '
                        """ } } } } } }
```

10.2 Backend Jenkinsfile

```
pipeline {
    agent any

    tools {
        maven 'Maven 3.8.6'
        jdk 'jdk17'
    }

    environment {
        IMAGE_NAME = 'swim-ui-backend'
        IMAGE_TAG = 'latest'
        REGISTRY = "35.188.22.165:5000"
        FULL_IMAGE_NAME = "${REGISTRY}/${IMAGE_NAME}:${IMAGE_TAG}"
        IMAGE_TAR = 'swim-ui-backend.tar'
        REMOTE_USER = 'ashis'
        REMOTE_HOST = '34.130.230.176'
        REMOTE_IMAGE_PATH = '/home/ashis/swim-ui-backend.tar'
        CONTAINER_NAME = 'swim-ui-backend'
        PORT = '8072'
        PROJECT_NAME = 'Swim-UI-backend'
    }

    stages {
        stage('Verify Tools') {
            steps {
                sh '''
                    java -version
            mvn -version
          '''
        }
      }

      stage('Cleanup Disk Space') {
        steps {
          sh 'docker system prune -af --volumes || true'
        }
      }

      stage('Checkout Code') {
        steps {
          git credentialsId: 'github-dod-credentials', url:
'https://github.com/data-on-disk/dod-swim-backend.git', branch: 'prod'
        }
      }

      stage('Build Application') {
        steps {
          sh 'mvn clean package -DskipTests'
        }
      }

      stage('Build Docker Image') {
        steps {
          sh 'docker build -t $FULL_IMAGE_NAME .'
        }
      }
```

```
        stage('Push to Private Registry') {
            steps {
                withCredentials([usernamePassword(credentialsId: 'docker-registry-creds',
usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')]) {
                    sh '''
                        echo "$DOCKER_PASS" | docker login "$REGISTRY" -u "$DOCKER_USER"
--password-stdin
                        docker push "$FULL_IMAGE_NAME"
                    '''
                }
            }
        }

        stage('Save Docker Image to Tar') {
            steps {
                sh 'docker save -o $IMAGE_TAR $FULL_IMAGE_NAME'
            }
        }

        stage('Deploy Docker Image to VM') {
            steps {
                sshagent(credentials: ['gcp-ssh-key']) {
                    sh """
                        scp -o StrictHostKeyChecking=no $IMAGE_TAR
$REMOTE_USER@$REMOTE_HOST:$REMOTE_IMAGE_PATH
                        ssh -o StrictHostKeyChecking=no $REMOTE_USER@$REMOTE_HOST '
                            docker load -i $REMOTE_IMAGE_PATH
                            docker stop $CONTAINER_NAME || true
                            docker rm $CONTAINER_NAME || true
                            docker run -d --name $CONTAINER_NAME -p $PORT:8080
$FUNLL_IMAGE_NAME
                        '
                    """
                }
            }
        }
    }
}
```

11. Project Structure

```
dod-infra/
├── infrastructure/
│   ├── ssl/
│   │   ├── main.cert
│   │   ├── main.key
│   │   └── lets-encrypt-r3.pem
│   │
│   └── nginx/
│       └── nginx.conf
│
├── automation/
│   ├── scripts/
│   │   ├── install_dependencies.sh
│   │   ├── main_fullchain.sh
│   │   ├── validate_ssl.sh
│   │   └── reload_nginx.sh
│   │
│   └── ci-cd/
│       ├── Jenkinsfile-frontend
│         └── Jenkinsfile-backend
│
└── README.md
```

12. Scripts & Config Files Content

12.1 install_dependencies.sh

```
#!/bin/bash

set -e

echo " Installing Docker, NGINX, and Nano..."

# Update package lists

sudo apt update

# Install required packages

sudo apt install -y docker.io nginx nano

# Enable and configure Docker

sudo systemctl enable docker

sudo usermod -aG docker $USER

echo " Dependencies installed. You may need to log out and log in again to use Docker
without sudo."
```

12.2 main_fullchain.sh

```
#!/bin/bash

set -e

SSL_DIR="/etc/nginx/ssl"

CERT_FILE="${SSL_DIR}/main.cert"

INTERMEDIATE_FILE="${SSL_DIR}/lets-encrypt-r3.pem"

FULLCHAIN_FILE="${SSL_DIR}/main.fullchain.cert"

echo "Creating fullchain certificate..."

# Check if required files exist

if [[ ! -f "$CERT_FILE" || ! -f "$INTERMEDIATE_FILE" ]]; then

    echo " Error: Missing certificate or intermediate file."

    exit 1

fi

# Concatenate to fullchain

sudo bash -c "cat $CERT_FILE $INTERMEDIATE_FILE > $FULLCHAIN_FILE"

echo "Fullchain created at: $FULLCHAIN_FILE"
```

12.3 validate_ssl.sh

```
#!/bin/bash

set -e

SSL_DIR="/etc/nginx/ssl"

CERT="${SSL_DIR}/main.cert"

KEY="${SSL_DIR}/main.key"

FULLCHAIN="${SSL_DIR}/main.fullchain.cert"

INTERMEDIATE="${SSL_DIR}/lets-encrypt-r3.pem"

echo "Validating SSL certificates..."

# Check file existence

for f in "$CERT" "$KEY" "$INTERMEDIATE" "$FULLCHAIN"; do

  if [[ ! -f "$f" ]]; then

    echo " Missing file: $f"

    exit 1

  fi

done

# Chain verification

echo "Verifying certificate chain..."

openssl verify -CAfile "$INTERMEDIATE" "$CERT"

openssl verify -CAfile "$FULLCHAIN" "$CERT"

# Cert and key match
```

```
echo "Checking certificate and key match..."

MOD1=$(openssl rsa -noout -modulus -in "$KEY" | openssl md5)

MOD2=$(openssl x509 -noout -modulus -in "$FULLCHAIN" | openssl md5)


if [[ "$MOD1" == "$MOD2" ]]; then

    echo "Certificate and key match."

else

    echo "Certificate and key do NOT match."

    exit 1

fi
```

12.4 reload_nginx.sh

```
#!/bin/bash

set -e

echo "Testing and reloading NGINX..."

# Test configuration

sudo nginx -t

# Reload if successful

sudo systemctl reload nginx

echo "NGINX reloaded successfully."
```