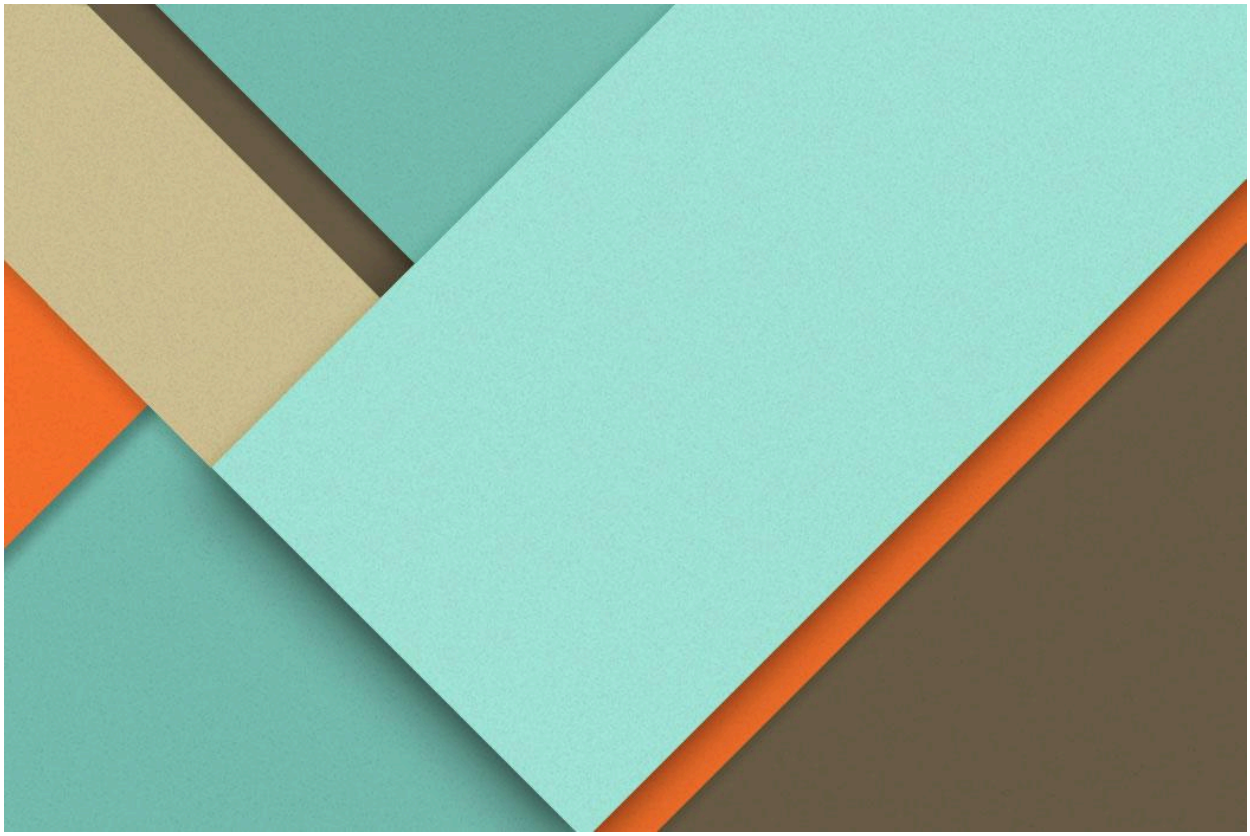

SaaranInfosphere

Overview

This document outlines the deployment steps for the **SaaranInfosphere** application on a Linux-based virtual machine (VM), without the use of Docker. The setup includes:

- Installing Node.js and NGINX
- Configuring SSL/TLS
- Deploying an Angular frontend
- Securing the application using HTTPS
- Automating deployments using Jenkins

Prerequisites

- A Linux VM (Ubuntu 20.04+ recommended)
- Root or sudo privileges
- Valid domain name (e.g., `saaraninfosphere.com`)
- SSL certificates (from Let's Encrypt or another Certificate Authority)
- Pre-built Angular application
- Jenkins installed with relevant plugins

Project Structure

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

Deployment Steps

1. Install Required Packages

Install Node.js v18

curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -

sudo apt-get install -y nodejs

Verify installation

node -v

npm -v

Install NGINX

sudo apt update

sudo apt install nginx -y

Install Nano

sudo apt install nano -y

2. Configure SSL Certificates

Create SSL Directory

sudo mkdir -p /etc/nginx/ssl

Place Certificate Files

Copy the following to `/etc/nginx/ssl/`:

- `saran.cert` – Domain certificate
- `saran.key` – Private key
- `lets-encrypt-r3.pem` – Intermediate certificate

Create Full Chain Certificate

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

3. Configure NGINX

Edit `/etc/nginx/nginx.conf` and use the configuration provided in the Appendix A section.

Test and reload NGINX

sudo nginx -t

sudo systemctl reload nginx

4. Validate SSL Configuration

Test Certificate Chain

openssl s_client -connect saaraninfosphere.com:443 -servername saaraninfosphere.com -showcerts

Verify Certificate Validity

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

Verify Certificate-Key Match

openssl rsa -noout -modulus -in /etc/nginx/ssl/saran.key | openssl md5

openssl x509 -noout -modulus -in /etc/nginx/ssl/saran.fullchain.cert | openssl md5

Jenkins CI/CD Pipeline

Purpose

Automates the build and deployment of the SaaranInfosphere Angular UI to the production VM.

Pipeline Steps

- Clone source code from GitHub
- Install Node dependencies
- Build Angular project
- Retrieve SSL certs from remote server
- Deploy the build to `/usr/share/nginx/html`
- Reload NGINX

Plugins Required

- NodeJS Plugin
- Git Plugin
- SSH Agent Plugin
- Pipeline Plugin

Credentials Configuration

- `gcp-ssh-key`: SSH key to access VM
- `github-creds`: GitHub token or credentials

Jenkins Declarative Pipeline

```
pipeline {
    agent any

    tools {
        nodejs 'Node18' // Node.js from Global Tool Configuration
    }

    environment {
        REPO_URL       = "https://github.com/data-on-disk/saaraninfosphere-main-ui.git"
        SSH_USER       = "ashis"
        VM1_IP         = "34.141.111.192"
        CREDENTIALS_ID = "gcp-ssh-key"
        BUILD_DIR      = "dist/main-site/browser"
        REMOTE_PATH    = "/usr/share/nginx/html"
    }

    stages {

        stage('Clone Repository') {
            steps {
```

```
            git credentialsId: 'github-creds', url: "${REPO_URL}", branch: 'prod'
        }
    }

    stage('Install Dependencies & Build') {
        steps {
            sh 'npm install'
            sh 'npm run build'
        }
    }

    stage('Fetch SSL Certificates from Remote VM') {
        steps {
            sshagent(credentials: [CREDENTIALS_ID]) {
                sh '''
                    echo "Fetching existing SSL certs from remote VM..."
                    rm -rf ssl
                    mkdir -p ssl
                    scp -o StrictHostKeyChecking=no
${SSH_USER}@${VM1_IP}:/etc/nginx/ssl/main.fullchain.cert ssl/
                    scp -o StrictHostKeyChecking=no
${SSH_USER}@${VM1_IP}:/etc/nginx/ssl/main.key ssl/
                '''
            }
```

```
        }
    }
    stage('Deploy Build to Remote VM') {
        steps {
            sshagent(credentials: [CREDENTIALS_ID]) {
                sh '''
                    echo "Transferring build files to VM..."
                    TMP_DIR="/tmp/deploy-$$"
                    ssh -o StrictHostKeyChecking=no ${SSH_USER}@${VM1_IP} "mkdir -p
$TMP_DIR"
                    scp -o StrictHostKeyChecking=no -r ${BUILD_DIR}/*
${SSH_USER}@${VM1_IP}:$TMP_DIR/
                    echo "Deploying files to NGINX directory and reloading..."
                    ssh -o StrictHostKeyChecking=no ${SSH_USER}@${VM1_IP} "
                        sudo cp -r $TMP_DIR/* ${REMOTE_PATH}/ &&
                        rm -rf $TMP_DIR &&
                        sudo systemctl reload nginx
                    "
                '''
            }
        }
    }
  }
}
```

Scripts

Install_dependencies.sh

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

echo "Updating packages..."
sudo apt update

echo "Installing Node.js v18..."
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs

echo "Installing NGINX and Nano..."
sudo apt install -y nginx nano

echo "Versions:"
node -v
npm -v
nginx -v
```

Create_fullchain.sh

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

CERT_DIR="/etc/nginx/ssl"
```

```
FULLCHAIN="$CERT_DIR/main.fullchain.cert"

echo "Creating fullchain certificate..."
sudo bash -c "cat $CERT_DIR/main.cert $CERT_DIR/lets-encrypt-r3.pem > $FULLCHAIN"

echo "Fullchain cert created at $FULLCHAIN"
```

Validate_ssl.sh

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

CERT_DIR="/etc/nginx/ssl"

echo "Verifying SSL certificate chain..."
openssl verify -CAfile "$CERT_DIR/lets-encrypt-r3.pem" "$CERT_DIR/main.cert"

echo "Verifying key and cert match..."
KEY_HASH=$(openssl rsa -noout -modulus -in "$CERT_DIR/main.key" | openssl md5)
CERT_HASH=$(openssl x509 -noout -modulus -in "$CERT_DIR/main.cert" | openssl md5)

if [ "$KEY_HASH" == "$CERT_HASH" ]; then
    echo "Key and certificate match."
else
    echo " Key and certificate DO NOT match!"
    exit 1
fi
```

Reload_nginx.sh

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

echo "Testing NGINX configuration..."
sudo nginx -t

echo "Reloading NGINX service..."
sudo systemctl reload nginx

echo "NGINX reloaded successfully."
```

Appendix A: Sample NGINX Configuration

```
events {}

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

    access_log  /var/log/nginx/access.log;
    error_log   /var/log/nginx/error.log;

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

    etag off;

    server {
        listen 80;
        server_name saaraninfosphere.com;

        return 301 https://$host$request_uri;
    }

    server {
        listen 443 ssl http2;
        server_name saaraninfosphere.com;
```

```
        root /usr/share/nginx/html;
        index index.html;

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

        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_prefer_server_ciphers on;
        ssl_ciphers
'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHA
CHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:E
CDHE-RSA-AES128-GCM-SHA256:!aNULL:!MD5:!3DES';
        ssl_session_cache shared:SSL:10m;
        ssl_session_timeout 1h;

        add_header Strict-Transport-Security "max-age=63072000; includeSubDomains;
preload" always;
        add_header X-Content-Type-Options nosniff always;
        add_header X-Frame-Options DENY always;
        add_header X-XSS-Protection "1; mode=block" always;
        add_header Referrer-Policy "strict-origin-when-cross-origin" always;
        add_header Permissions-Policy "geolocation=(), microphone=(), camera=()" always;
        add_header Content-Security-Policy "default-src 'self'; script-src 'self';  style-src 'self'
'unsafe-inline'; object-src 'none'; base-uri 'self'; frame-ancestors 'none';" always;

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

        location ~* \.html$ {
            add_header Cache-Control "no-cache, no-store, must-revalidate" always;
```

```
        add_header Pragma "no-cache" always;
        add_header Expires 0 always;
    }

    location ~* \.(?:js|css|woff2?|ttf|svg|eot|ico|jpg|jpeg|png|gif|webp|json)$ {
        access_log off;
        add_header Cache-Control "public, max-age=31536000, immutable" always;
    }

    location ~ /\. {
        deny all;
    }
  }
}
```

Online SSL Testing Tools

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