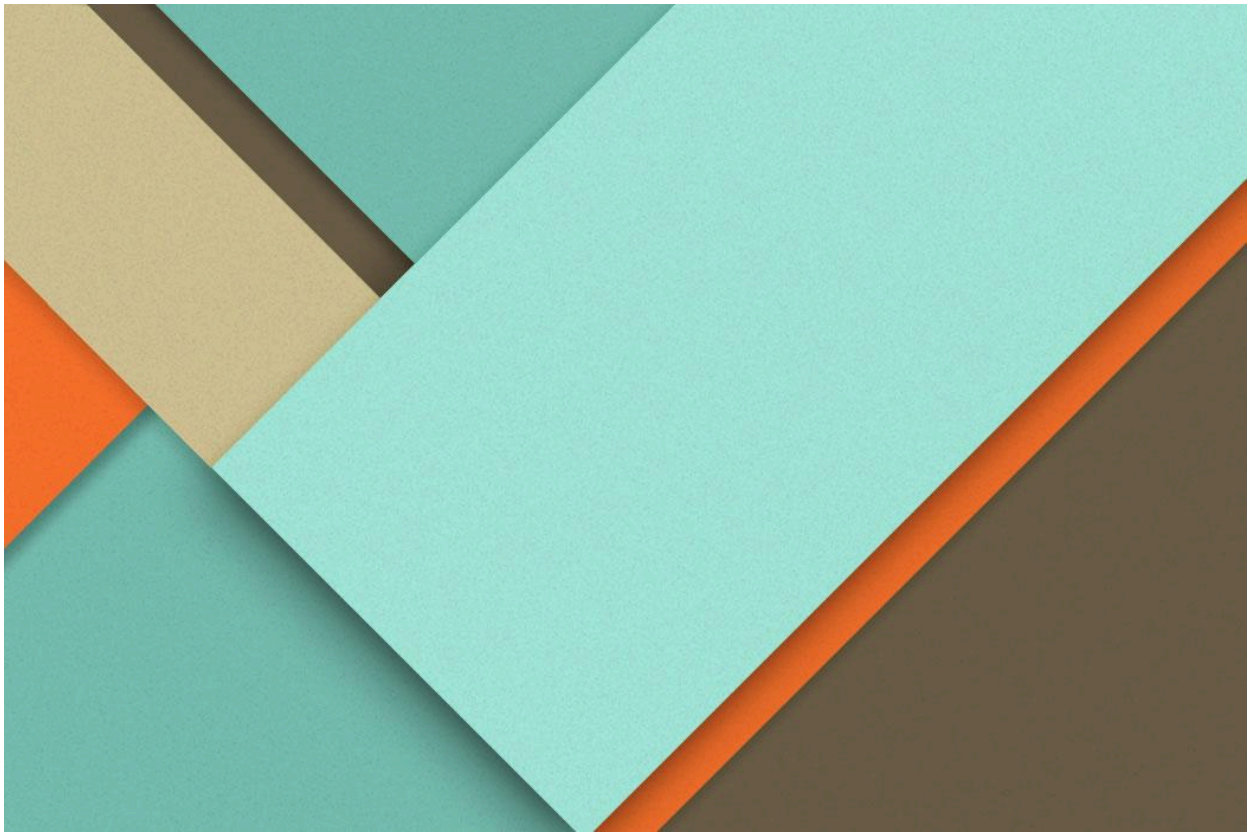

DOD MAIN

Overview

The steps required to deploy the DataOnDisk (DoD) application on a Linux-based virtual machine (VM). The deployment does not use Docker and involves configuring Node.js, serving an Angular frontend using NGINX, and securing the application using SSL/TLS certificates.

Prerequisites

1. A Linux VM (Ubuntu recommended)
2. Root or sudo privileges
3. Domain name (e.g., `dataondisk.com`)
4. SSL certificates (can be from Let's Encrypt or a trusted CA)

1. Install Required Packages

Install Node.js (v18 LTS)

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

sudo apt-get install -y nodejs

Confirm Installation

node -v

npm -v

Install NGINX

sudo apt update

sudo apt install nginx -y

Install Nano Text Editor

sudo apt install nano -y

2. Configure SSL Certificates

Create SSL Directory

sudo mkdir -p /etc/nginx/ssl

Place SSL Files

Ensure the following files are placed in /etc/nginx/ssl/:

main.cert – Your domain certificate
main.key – Your private key
lets-encrypt-r3.pem – Intermediate certificate (if using Let's Encrypt)

Create Full Chain Certificate

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

3. Configure NGINX

Edit or create the NGINX configuration file (e.g., `/etc/nginx/nginx.conf`) and insert the following 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 dataondisk.com;

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

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

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

            client_max_body_size 20M;

            ssl_certificate     /etc/nginx/ssl/main.fullchain.cert;
            ssl_certificate_key /etc/nginx/ssl/main.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_conf_command Ciphersuites
TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_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 1.0.0.1 valid=300s;
        resolver_timeout 5s;

        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 ~* \.html$ {
            add_header Cache-Control "no-cache, no-store, must-revalidate" always;
            add_header Pragma "no-cache" always;
            add_header Expires 0 always;
            add_header Last-Modified "" always;
            etag off;
            try_files $uri =404;
```

```
        }

        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;
            try_files $uri =404;
        }

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

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

4. Reload NGINX and Validate

Test Configuration

sudo nginx -t

Reload NGINX

sudo systemctl reload nginx

5. SSL Certificate Validation

Test the Certificate Chain

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

Verify the Chain

Option A

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

Option B

openssl verify -CAfile /etc/nginx/ssl/main.fullchain.cert /etc/nginx/ssl/main.cert

Verify Certificate and Key Match

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

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

6. Online SSL Testing Tools

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

7. HSTS Preload Submission

https://hstspreload.org/

8. CI/CD Deployment Guide with Jenkins for DataOnDisk (DoD) UI

Jenkins Pipeline Overview

This Jenkins Pipeline:

- Clones the UI repository from GitHub
- Installs Node.js dependencies and builds the Angular project
- Securely retrieves SSL certificates from the remote VM
- Deploys the build to the remote VM (/usr/share/nginx/html)
- Reloads NGINX to reflect changes

Prerequisites

Jenkins Plugins Required:

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

Global Tool Configuration:

- Add Node.js installation labeled as: Node18

Credentials Setup:

- gcp-ssh-key: SSH private key for the remote VM
- github-creds: GitHub personal access token (PAT) or username/password for private repo access

Remote Server:

- NGINX already installed and configured
- Angular app previously deployed at `/usr/share/nginx/html`
- SSL certificates in `/etc/nginx/ssl/`

Jenkins Declarative Pipeline

```
pipeline {
    agent any

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

```
environment {
    REPO_URL      = "https://github.com/data-on-disk/dod-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
              "
            '''
```

```
            }
        }
    }
  }
}
```