VOIDA 포팅 매뉴얼

노션 페이지 바로가기

Frontend 테스트 명령어

앱 실행

> 앱 실행을 위해서는 메인 페이지에서 파일을 다운 받아야 합니다.

- **개발 서버 실행(웹 실행)**

```
npm run dev
```

- **프로덕션 빌드 및 개발 서버 실행 (앱 실행)**

```
npm run build
npm start
```

Backend 테스트 명령어

```
gradlew.bat build
```

---

1. 사용 도구

- 이슈 관리: JIRA
- 형상 관리: GitLab
- 커뮤니케이션: MatterMost, Notion, Discord
- 디자인: Figma
- 영상 포트폴리오: Adobe premiere pro, Movavi Video Editor 25
- CI/CD: Jenkins, Nginx, S3, CloudFront, Docker

---

2. 개발 환경

- 서버 OS : Ubuntu 22.04 LTS
- IDE : VSCode, IntelliJ IDEA
- SSH 접속

- DB : MySQL, Redis, MongoDB
- Front-end: Node.js v22.17.0, TypeScript, Electron v37.3.0
- Back-end: JVM 17, SpringBoot 3.4.5, Python 3.9, FastAPI

---

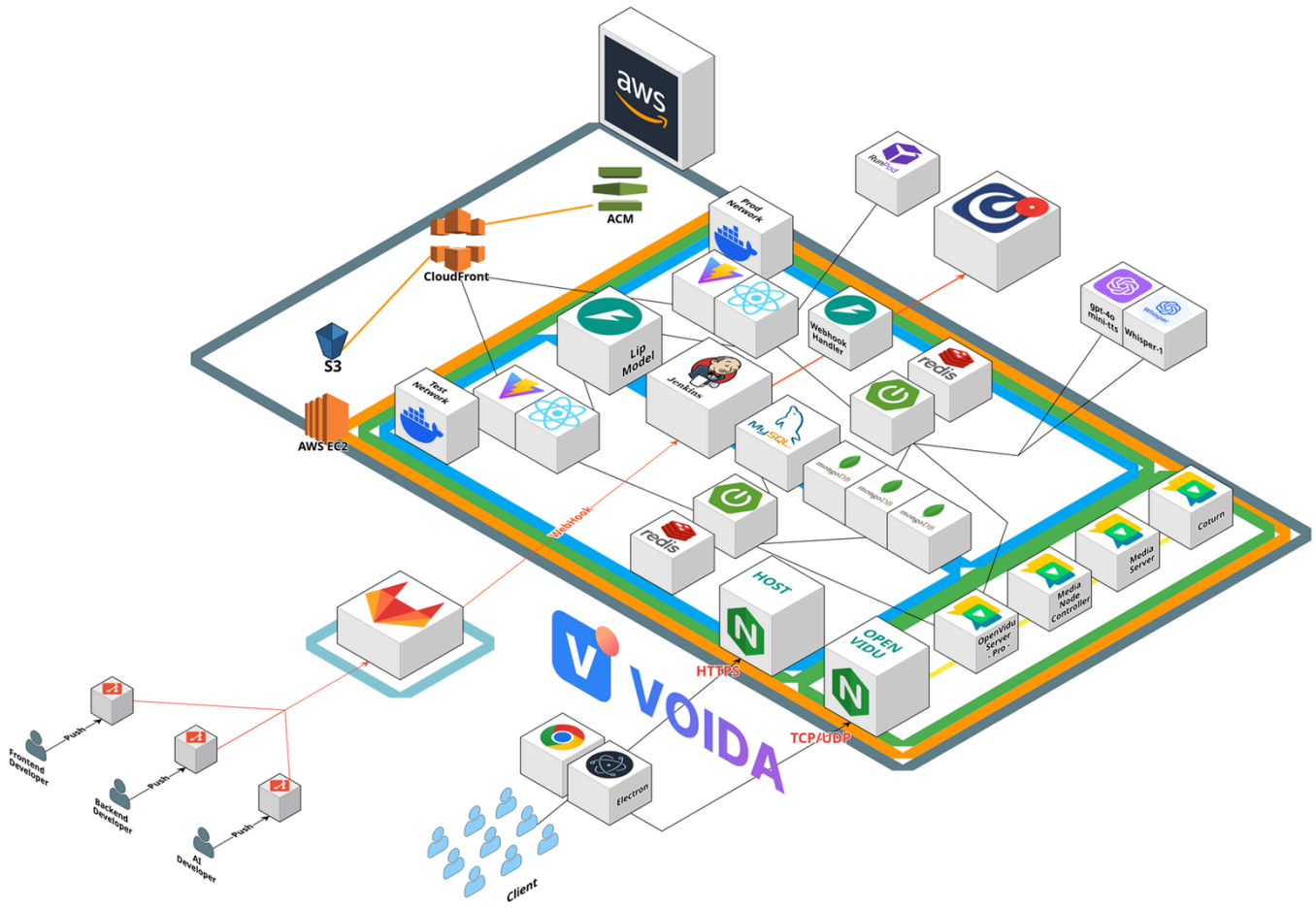
3. DB

3.1 ERD

- https://www.erdcloud.com/d/9NgtvEQNntKNKBXi9

---

4. 시스템 아키텍처 및 기술 스택



4.1 프론트엔드

- 주요 기술: React, TypeScript, electron
- 상태 관리: Zustand
- API 통신: Axios
- 웹소켓 통신: stompjs, sockjs, openvidu
- 스타일링: Emotion CSS, lucide-react
- 코드 컨벤션 : ESLint, Prettier

4.2 백엔드

- 주요 기술 : Java17, Python 3.9, FastAPI, SpringBoot, STOMP, JPA, JWT, SMTP, OAuth2, Gradle, OpenVidu, STT, TTS, Lip Model
- 인증 및 토큰 관리 : JWT, OAuth2
- 웹소켓 : WebSocket, STOMP, OpenVidu
- ORM : JPA

5. 서버 설정

- **백엔드 backend-core**
  - `application.yaml`

```
server:
  port: 26281

swagger:
  url: https://api.voida.site

spring:
  application:
    name: backend-core-prod

  servlet:
    multipart:
      max-file-size: 10MB
      max-request-size: 10MB

  # MySQL
  datasource:
    url: jdbc:mysql://mysql/voida_prod
    username: your_mysql_username
    password: your_mysql_password
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    hibernate:
      ddl-auto: none
      format-sql: true
    show-sql: true
    properties:
      hibernate:
        format_sql: true
  data:
    redis:
      host: your_redis_host
      port: your_redis_port
      password: your_redis_password
    mongodb:
      uri: mongodb://your_mongodb_id:your_mongodb_password@mongo-primary,mongo-secondary,mong
o-backup/voida_prod?authSource=admin
      auto-index-creation: true


cloud:
  aws:
    credentials:
      access-key: your_aws_access_key
      secret-key: your_aws_secret_key
    region:
      static: your_aws_region
    s3:
      bucket: your_aws_s3_bucket
mail:
  host: smtp.gmail.com
  port: 587
  username: your_gmail_username
  password: your_gmail_app_password
  properties:
    mail:
      smtp:
```

```
        auth: true
        starttls:
          enable: true
  security:
    oauth2:
      client:
        registration:
          google:
            client-id: your_oauth2_google_client_id
            client-secret: your_oauth2_client_secret
            scope:
              - email
            redirect-uri: "{baseUrl}/login/oauth2/code/{registrationId}"

jwt:
  secret: your_jwt_secret_key
  expire-time:
    access: 15m
    refresh: 14d


security: # 보안 관련 설정
  whitelist:
    GET:
      - /swagger-ui/**
      - /v3/api-docs/**
      - /swagger-resources/**
      - /webjars/**
      - /v1/auth/random-nickname
      - /oauth2/authorization/**
      - /login/oauth2/code/**
      - /v1/auth/reissue
      - /ws/**
      - /v1/releases/**
    POST:
      - /v1/auth/sign-in
      - /v1/auth/reissue
      - /v1/auth/email-code
      - /v1/auth/verify-email
      - /v1/auth/check-nickname
      - /v1/auth/sign-up
      - /v1/auth/reset-password
      - /v1/auth/check-email

cors:
  allowed-origins:
    - https://www.voida.site
    - https://api.voida.site
    - http://localhost:5173
    - http://127.0.0.1:5173
    - file://
  allowed-methods:
    - GET
    - POST
    - PUT
    - DELETE
    - PATCH
    - OPTIONS
  allowed-headers:
    # - "*"
    - Authorization
```

```
      - Content-Type
      - Accept
    allow-credentials: true
    exposed-headers:
      - Set-Cookie
      - Authorization
    max-age: 3600

  oauth:
    authorization-endpoint: /oauth2/authorization
    redirection-endpoint: /login/oauth2/code/*

    client-endpoint: https://app.voida.site/#/oauth/success # 프론트의 콜백 주소로 리턴
    client-endpoint-link: https://app.voida.site/#/mypage # 프론트의 마이페이지로 리턴

voida:
  main-url: https://www.voida.site/
  social:
    sign-up-expire-time: 20m


ai:
  openai:
    api-key: your_gms_api_key
    audio:
      speech:
        options:
          model: gpt-4o-mini-tts
        endpoint: https://gms.ssafy.io/gmsapi/api.openai.com/v1/audio/speech
      transcriptions:
        options:
          model: whisper-1
        endpoint: https://gms.ssafy.io/gmsapi/api.openai.com/v1/audio/transcriptions

openvidu:
  url: https://rtc.voida.site:20443
  secret: your_openvidu_secret_key
  session:
    prefix: live_prod_
  websocket:
    scheme: wss
    host: rtc.voida.site:20443
    path: /openvidu/ws
```

- **백엔드 backend-lip-model**
  - `.env`

```
# JWT
JWT_SECRET_KEY=your_spring-server_jwt_secret_key # spring 서버의 시크릿키와 동일하게 설정

# CORS
ALLOWED_ORIGINS='https://lip.voida.site,https://app.voida.site,http://localhost:5173,http://127.0.0.1:5173,file://'

# GMS
OPENAI_API_KEY=your_gms_api_key
OPENAI_BASE_URL=https://gms.ssafy.io/gmsapi/api.openai.com/v1
```

- **프론트엔드 front-web**
  - `package.json`

```
{
  "name": "frontend-web",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "tsc -b && vite build",
    "lint": "eslint .",
    "preview": "vite preview"
  },
  "dependencies": {
    "@emotion/react": "^11.14.0",
    "@emotion/styled": "^11.14.1",
    "axios": "^1.11.0",
    "lucide-react": "^0.535.0",
    "react": "^19.1.0",
    "react-dom": "^19.1.0",
    "react-router-dom": "^7.7.1"
  },
  "devDependencies": {
    "@eslint/js": "^9.30.1",
    "@types/node": "^24.1.0",
    "@types/react": "^19.1.8",
    "@types/react-dom": "^19.1.6",
    "@vitejs/plugin-react": "^4.6.0",
    "eslint": "^9.30.1",
    "eslint-plugin-react-hooks": "^5.2.0",
    "eslint-plugin-react-refresh": "^0.4.20",
    "globals": "^16.3.0",
    "typescript": "~5.8.3",
    "typescript-eslint": "^8.35.1",
    "vite": "^7.0.4"
  }
}
```

- **프론트엔드 front-app**
  - `package.json`

```
{
  "name": "voida",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "electron:build": "esbuild electron/main.ts --bundle --platform=node --format=cjs --external:electro
n --outfile=electron/dist/main.cjs && esbuild electron/preload.ts electron/overlayWindow.ts --bundle -
-platform=node --format=cjs --external:electron --outdir=electron/dist",
    "electron": "cross-env ELECTRON_DEV=true electron electron/dist/main.cjs",
    "start": "concurrently \"npm run dev\" \"wait-on http://localhost:5173 && npm run electron:build && n
pm run electron\"",
    "build": "vite build",
    "dist": "npm run build && npm run electron:build && electron-builder",
```

```
    "lint": "eslint ."
  },
  "main": "electron/dist/main.cjs",
  "build": {
    "appId": "com.ssafy.voida",
    "productName": "VOIDA",
    "files": [
      "dist/**",
      "electron/dist/**"
    ],
    "win": {
      "icon": "electron/assets/voida-favicon.ico",
      "target": "nsis"
    },
    "asarUnpack": [
      "electron/dist/preload.js"
    ]
  },
  "dependencies": {
    "@emotion/react": "^11.14.0",
    "@emotion/styled": "^11.14.1",
    "@stomp/stompjs": "^7.1.1",
    "axios": "^1.10.0",
    "canvas-confetti": "^1.9.3",
    "cross-env": "^10.0.0",
    "emoji-picker-react": "^4.13.2",
    "lucide-react": "^0.526.0",
    "openvidu-browser": "^2.31.0",
    "react": "^19.1.0",
    "react-dom": "^19.1.0",
    "react-icons": "^5.5.0",
    "react-router-dom": "^7.7.0",
    "sockjs-client": "^1.6.1",
    "zustand": "^5.0.6"
  },
  "devDependencies": {
    "@eslint/js": "^9.30.1",
    "@types/canvas-confetti": "^1.9.0",
    "@types/node": "^24.1.0",
    "@types/react": "^19.1.8",
    "@types/react-dom": "^19.1.6",
    "@types/sockjs-client": "^1.5.4",
    "@vitejs/plugin-react": "^4.6.0",
    "concurrently": "^9.2.0",
    "electron": "^37.3.0",
    "electron-builder": "^26.0.12",
    "esbuild": "^0.25.8",
    "eslint": "^9.30.1",
    "eslint-plugin-react-hooks": "^5.2.0",
    "eslint-plugin-react-refresh": "^0.4.20",
    "express": "^5.1.0",
    "globals": "^16.3.0",
    "typescript": "~5.8.3",
    "typescript-eslint": "^8.35.1",
    "vite": "^7.0.4",
    "wait-on": "^8.0.4"
  }
}
```

- `.env`

```
VITE_JSON_SERVER=http://www.voida.site:3003
VITE_SPRING_API_URL=https://api.test.voida.site
VITE_FAST_API_URL=https://lip.voida.site
VITE_CDN_URL=https://media.voida.site
VITE_WEB_URL=https://www.test.voida.site/
```

  - FastAPI → runpod 서버(외부 GPU) 사용 시 `VITE_FAST_API_URL` 항목을 해당 url로 변경

6. 빌드 및 배포

6.1 젠킨스 파이프라인 스크립트

```
pipeline {
    agent any

    environment {
        GIT_REPO_URL = 'https://lab.ssafy.com/s13-webmobile1-sub1/S13P11E107'
        PROJECT_ID = '1050708'
        GIT_CREDENTIALS_ID = 'gitlab-access-token'
        S3_BUCKET = 's3://voida-assets'

        FINAL_BRANCH = "${params.OVERRIDE_BRANCH ?: env.gitlabTargetBranch ?: env.gitlabBranch}"
        FINAL_AUTHOR = "${params.OVERRIDE_USERNAME}"
        FINAL_COMMIT = "${params.OVERRIDE_COMMIT}"
    }

    options {
        skipDefaultCheckout true
    }

    stages {
        stage('Print environment values') {
            steps {
                echo "FINAL_BRANCH: ${env.FINAL_BRANCH}"

                script {
                    // Re-load는 ActionType을 PUSH로 설정
                    env.gitlabActionType = env.gitlabActionType ?: 'PUSH'
                }
                echo "env.gitlabActionType: ${env.gitlabActionType}"
            }
        }

        stage('Checkout & Detect Changed Directories') {
            when {
                expression {
                    return env.FINAL_BRANCH == 'develop' || env.FINAL_BRANCH == 'release'
                }
            }
            steps {
                script {
                    if (!env.FINAL_BRANCH?.trim()) {
                        error('❌ FINAL_BRANCH가 정의되지 않았습니다. 빌드를 종료합니다.')
                    }

                    echo "✅ Checkout to branch: ${env.FINAL_BRANCH}"
```

```
// Checkout
try {
    checkout([$class: 'GitSCM',
        branches: [[name: "*/${env.FINAL_BRANCH}"]],
        userRemoteConfigs: [[
            url: "${env.GIT_REPO_URL}.git",
            credentialsId: "${env.GIT_CREDENTIALS_ID}"
        ]]
    ])
} catch (err) {
    echo "err: ${err}"
}

def changedFiles = ''

if (params.OVERRIDE_ACTION == 'MERGE' || env.gitlabActionType == 'MERGE') {
    echo "🔀 MERGE 이벤트 감지됨"
                def mergeCommit = sh(script: 'git rev-parse HEAD', returnStdout: true).trim()
                def parents = sh(script: "git log -1 --pretty=%P ${mergeCommit}", returnStdout: true).trim().spli
t(' ')

                if (parents.size() >= 1) {
                    def baseCommit = parents[0]
                    changedFiles = sh(
                        script: "git diff --name-only ${baseCommit} ${mergeCommit} || true",
                        returnStdout: true
                    ).trim()
                }

            } else if (params.OVERRIDE_ACTION == 'PUSH' || env.gitlabActionType == 'PUSH') {
                echo "⬇ PUSH 이벤트 감지됨"
                FINAL_COMMIT = env.FINAL_COMMIT ?: sh(script: 'git rev-parse HEAD', returnStdout: true).tri
m()
                echo "🔍 변경 파일 추적 기준 커밋: ${FINAL_COMMIT}"

                changedFiles = sh(
                    script: "git show --name-only --pretty=\"\" ${FINAL_COMMIT} || true",
                    returnStdout: true
                ).trim()
            }

if (!changedFiles) {
    echo "❌ 변경된 파일이 없습니다. 빌드를 건너뜁니다."
    // Merge Request가 승인 됐을 때 Merged에 대한 작업만 수행하게 함
    // 원래는 Merged 작업과 Pushed 작업 2개를 수행함
    SHOULD_SKIP_MESSAGE = true
} else {
    // echo "📂 변경된 파일 목록:\n${changedFiles}"
}

// 변경된 디렉터리 판단
def CHANGED_BACKEND_CORE = changedFiles.split('\n').any { it.startsWith('backend-core/') }
def CHANGED_BACKEND_LIP = changedFiles.split('\n').any { it.startsWith('backend-lip-model/') }
def CHANGED_FRONTEND_WEB = changedFiles.split('\n').any { it.startsWith('frontend-web/') }
def CHANGED_FRONTEND_APP = changedFiles.split('\n').any { it.startsWith('frontend-app/') }

env.CHANGED_BACKEND_CORE = CHANGED_BACKEND_CORE.toString()
env.CHANGED_BACKEND_LIP = CHANGED_BACKEND_LIP.toString()
```

```
            env.CHANGED_FRONTEND_WEB = CHANGED_FRONTEND_WEB.toString()
            env.CHANGED_FRONTEND_APP = CHANGED_FRONTEND_APP.toString()

            echo "CHANGED_BACKEND_CORE: ${env.CHANGED_BACKEND_CORE}"
            echo "CHANGED_BACKEND_LIP: ${env.CHANGED_BACKEND_LIP}"
            echo "CHANGED_FRONTEND_WEB: ${env.CHANGED_FRONTEND_WEB}"
            echo "CHANGED_FRONTEND_APP: ${env.CHANGED_FRONTEND_APP}"
        }
    }
}

stage('Spring Boot Build & Deploy') {
    when {
        expression {
            return (env.FINAL_BRANCH == 'develop' || env.FINAL_BRANCH == 'release') &&
                   env.CHANGED_BACKEND_CORE == 'true'
        }
    }
    steps {
        script {
            def isDev = (env.FINAL_BRANCH == 'develop')
            def yamlCredentialsId = isDev ? 'application-yaml-test' : 'application-yaml-prod'
            def netType = isDev ? 'test' : 'prod'
            def dockerTag = isDev ? 'backend-core-test' : 'backend-core-prod'
            def port = isDev ? '26280' : '26281'

            echo "▶️ Spring Boot application.yaml 설정"

            withCredentials([file(credentialsId: yamlCredentialsId, variable: 'APP_YAML')]) {
                sh """
                    cd backend-core
                    mkdir -p ./src/main/resources   # 디렉터리 없으면 생성
                    cp -f \$APP_YAML ./src/main/resources/application.yaml
                    cat ./src/main/resources/application.yaml
                """
            }

            echo "▶️ ${netType} Server: Spring Boot Build & Deploy 시작"

            withCredentials([usernamePassword(
                credentialsId: 'dockerhub-credentials',
                usernameVariable: 'DOCKERHUB_USERNAME',
                passwordVariable: 'DOCKERHUB_PASSWORD'
            )]) {
                sh """
                    echo "\$DOCKERHUB_PASSWORD" | docker login -u "\$DOCKERHUB_USERNAME" --passw
ord-stdin
                    cd backend-core
                    docker build -f Dockerfile -t \$DOCKERHUB_USERNAME/${dockerTag} .

                    docker stop ${dockerTag} 2>/dev/null || true
                    docker rm ${dockerTag} 2>/dev/null || true
                    docker run -d --name ${dockerTag} -p ${port}:${port} --network ${netType} \$DOCKERHUB
_USERNAME/${dockerTag}:latest
                    docker network connect devops ${dockerTag}
                    docker image prune -f
                """
            }
        }
    }
}
```

```
    }

    stage('Backend Lip Model Build & Deploy') {
      when {
        expression {
          return (env.FINAL_BRANCH == 'develop' || env.FINAL_BRANCH == 'release') &&
              env.CHANGED_BACKEND_LIP == 'true'
        }
      }
      steps {
        script {
          def isDev = (env.FINAL_BRANCH != 'release')
          def envFileCredentialsId = isDev ? 'lip-model-env-test' : 'lip-model-env-prod'
          def netType = isDev ? 'test' : 'prod'
          def dockerTag = isDev ? 'backend-lip-model-test' : 'backend-lip-model-prod'
          def port = isDev ? '16280' : '16281'

          withCredentials([file(credentialsId: envFileCredentialsId, variable: 'ENV_FILE')]) {
            sh """
              cd backend-lip-model
              cp -f \$ENV_FILE ./.env
              cat .env
            """
          }

          echo "▶️ FastAPI Docker Build & Deploy 시작"

          withCredentials([usernamePassword(
            credentialsId: 'dockerhub-credentials',
            usernameVariable: 'DOCKERHUB_USERNAME',
            passwordVariable: 'DOCKERHUB_PASSWORD'
          )]) {
            def status = sh(
              script: """
                bash -c '
                  set -o pipefail
                  cd backend-lip-model
                  mkdir -p temp-logs
                  {
                    echo "Downloading model weight file ..."
                    wget -q --show-progress "https://media.voida.site/models/weights/vsr_trlrs2lrs3vo
x2avsp_base.pth" -O models/vsr_trlrs2lrs3vox2avsp_base.pth

                    echo "Downloading model binary file ..."
                    wget -q --show-progress "https://media.voida.site/models/lip-model/model.bin" -O
models/lip-model/model.bin

                    ls models

                    echo "\$DOCKERHUB_PASSWORD" | docker login -u "\$DOCKERHUB_USERNAME"
--password-stdin
                    docker build -f Dockerfile -t \$DOCKERHUB_USERNAME/${dockerTag} .
                    BUILD_STATUS=\${PIPESTATUS[0]}

                    if [ "\$BUILD_STATUS" -ne 0 ]; then
                      echo "❌ Docker build failed with exit code \$BUILD_STATUS"
                      exit \$BUILD_STATUS
                    fi

                    docker stop ${dockerTag} 2>/dev/null || true
```

```
                                docker rm ${dockerTag} 2>/dev/null || true
                                docker run -d --name ${dockerTag} -p ${port}:${port} -e PORT=${port} --network
${netType} \$DOCKERHUB_USERNAME/${dockerTag}:latest
                                docker image prune -f
                            } 2>&1 | tee temp-logs/fastapi-cicd.log
                        '
                    """,
                returnStatus: true)

                if (status != 0) {
                    echo "❌ Lip Model FastAPI 배포 실패"

                    def errorLog = sh(
                        script: "grep 'ERROR\\|error\\|failed' backend-lip-model/temp-logs/fastapi-cicd.log || ec
ho '[ERROR] 로그 없음'",
                        returnStdout: true
                    ).trim()

                    echo "🔥 FastAPI 배포 에러 로그:\n${errorLog}"

                    ERROR_MESSAGE = "#### ❌ [FastAPI 빌드] 중 에러 발생! ####\n${errorLog}"
                    // 이후 stage skip
                    currentBuild.result = 'FAILURE'
                }
            }
        }
    }
}

stage('Frontend Web Build & Deploy') {
    when {
        expression {
            return (env.FINAL_BRANCH == 'develop' || env.FINAL_BRANCH == 'release') &&
                env.CHANGED_FRONTEND_WEB == 'true'
        }
    }
    steps {
        script {
            def isDev = (env.FINAL_BRANCH == 'develop')
            def envFileCredentialsId = isDev ? 'frontend-web-env-test' : 'frontend-web-env-prod'
            def dockerTag = isDev ? 'frontend-web-test' : 'frontend-web-prod'
            def viteDistPath = '/app/dist/.'
            def deployPath = isDev ? '/mnt/test-html' : '/mnt/nginx-html'
            def netType = isDev ? 'test' : 'prod'

            withCredentials([file(credentialsId: envFileCredentialsId, variable: 'ENV_FILE')]) {
                sh """
                    cd frontend-web
                    cp -f \$ENV_FILE ./.env
                    cat .env
                """
            }

            withCredentials([usernamePassword(
                credentialsId: 'dockerhub-credentials',
                usernameVariable: 'DOCKERHUB_USERNAME',
                passwordVariable: 'DOCKERHUB_PASSWORD'
            )]) {
                def status = sh(
                    script: """
```

```
                    bash -c '
                        set -o pipefail
                        cd frontend-web
                        mkdir -p temp-logs

                        echo "\$DOCKERHUB_PASSWORD" | docker login -u "\$DOCKERHUB_USERNAME" --p
assword-stdin

                        docker rm ${dockerTag} 2>/dev/null || true

                        docker build -t \$DOCKERHUB_USERNAME/${dockerTag} . 2>&1 | tee temp-logs/vite-bu
ild.log
                        BUILD_STATUS=\${PIPESTATUS[0]}

                        if [ "\$BUILD_STATUS" -ne 0 ]; then
                            echo "❌ Docker build failed with exit code \$BUILD_STATUS"
                            exit \$BUILD_STATUS  # 실패 exit
                        fi

                        docker create --name ${dockerTag} \$DOCKERHUB_USERNAME/${dockerTag}
                        rm -rf ${deployPath}/.*
                        docker cp ${dockerTag}:${viteDistPath} ${deployPath}
                        docker image prune -f

                        exit 0
                    '
                """,
                returnStatus: true
            )

            if (status != 0) {
                echo "❌ Frontend Web 배포 실패"

                def errorLog = sh(
                    script: "grep 'ERROR\\|error\\|failed' frontend-web/temp-logs/vite-build.log || echo '[ERRO
R] 로그 없음'",
                    returnStdout: true
                ).trim()

                def detailedErrorLog = sh(
                    script: "grep -A 10 'error during build' frontend-web/temp-logs/vite-build.log || echo '[DET
AIL ERROR] 없음'",
                    returnStdout: true
                ).trim()

                echo "🔥 Frontend 배포 에러 로그:\n${errorLog}\n${detailedErrorLog}"

                ERROR_MESSAGE = "#### ❌ [Frontend Web 빌드] 중 에러 발생! ####\n${errorLog}\n${detail
edErrorLog}"
                currentBuild.result = 'FAILURE'
            }
        }
    }
}

stage('Frontend App Build & Deploy') {
    when {
        expression {
            return (env.FINAL_BRANCH == 'develop' || env.FINAL_BRANCH == 'release') &&
```

```
                env.CHANGED_FRONTEND_APP == 'true'
        }
    }
    steps {
        script {
            echo "Frontend App Build & Deploy"
            def isDev = (env.FINAL_BRANCH == 'develop')
            def envFileCredentialsId = isDev ? 'frontend-app-env-test' : 'frontend-app-env-prod'
            def postfix = isDev ? 'Test_' : ''

            withCredentials([file(credentialsId: envFileCredentialsId, variable: 'ENV_FILE')]) {
                sh """
                    cd frontend-app
                    cp \$ENV_FILE ./.env
                    cat .env
                """
            }

            def buildStatus = sh(
                script: """
                  bash -euo pipefail -c '
                    cd frontend-app
                    mkdir -p temp-logs
                    : > temp-logs/electron-build.log

                    echo "== STEP: npm install =="
                    npm install >> temp-logs/electron-build.log 2>&1 || { echo "::FAIL step=npm_install code=11" |
tee -a temp-logs/electron-build.log; exit 11; }

                    echo "== STEP: npm run dist =="
                    if ! npm run dist -- --win >> temp-logs/electron-build.log 2>&1; then
                      echo "❌ Build failed, printing log..."
                      cat temp-logs/electron-build.log
                      exit 1
                    fi

                    echo "::OK all steps" | tee -a temp-logs/electron-build.log
                  '
                """,
                  returnStatus: true
                )

            if (buildStatus != 0) {
              def errorLog = sh(
                script: """
                  bash -euo pipefail -c "
                    sed -r 's/\\\\x1b\\\\[[0-9;]*m//g' frontend-app/temp-logs/electron-build.log \
                    | grep -F -i -m 50 -e '[vite:load-fallback]' -e 'Could not load' -e 'Build failed' -e 'npm ERR!' \
                    || echo '[ERROR] 로그 없음'
                  "
                """,
                returnStdout: true
              ).trim()

              def detailedErrorLog = sh(
                script: """
                  bash -euo pipefail -c "
                    sed -r 's/\\\\x1b\\\\[[0-9;]*m//g' frontend-app/temp-logs/electron-build.log \
                    | awk '/error during build:/,0' \
                    | sed -n '1,120p' \
```

```
        || echo '[DETAIL ERROR] 없음'
      "
    """,
    returnStdout: true
  ).trim()

  echo "❌ Desktop App 빌드 실패\\n${errorLog}\\n${detailedErrorLog}"
  archiveArtifacts artifacts: 'frontend-app/temp-logs/electron-build.log', fingerprint: true

  ERROR_MESSAGE = """#### ❌ [Frontend App 빌드] 중 에러 발생! ####
    ${errorLog}

    ${detailedErrorLog}"""

  currentBuild.result = 'FAILURE'
  error("빌드 실패")
}

// 결과물 파일명 변경
def newFileName = "VOIDA_Setup_${postfix}${desktopAppVersion}.exe"
sh """
  bash -euo pipefail -c '
    cd frontend-app/dist
    ls -al

    shopt -s nullglob
    set -- "VOIDA Setup"*.exe
    if [ "\$#" -eq 0 ]; then
      echo "❌ 설치파일(.exe)을 찾지 못했습니다. "
      exit 2
    fi
    originalFile="\$1"
    mv -- "\$originalFile" "${newFileName}"
    echo "✅ 빌드 결과물 파일명: ${newFileName}"
  '
"""

if (env.FINAL_BRANCH == 'develop') {
    echo "== STEP: Desktop App - Web Test Deploy ... =="
    sh """
      bash -euo pipefail -c '
        cd frontend-app/dist
        rm -rf /mnt/electron-web/.*

        cp -r assets /mnt/electron-web/
        cp -r fonts  /mnt/electron-web/
        cp -r logo   /mnt/electron-web/
        cp index.html /mnt/electron-web/
        echo "⚛️ Desktop App - Web Test Deploy Success"
        '
      """
  }

  withCredentials([usernamePassword(credentialsId: 'backend-core-api-credentials',   usernameVa
riable: 'EMAIL',
    passwordVariable: 'PASSWORD')]) {

      def BACKEND_API_BASE = isDev ? "http://backend-core-test:26280" : "http://backend-core-
prod:26281"
      def LOGIN_PATH = "/v1/auth/sign-in"
```

```
                def RELEASE_PATH = "/v1/releases/desktop-apps/versions/${desktopAppVersion}"

                def loginPayload = groovy.json.JsonOutput.toJson([ email: env.EMAIL, password: env.PASS
WORD ])

                def raw = sh(
                  script: """
                    curl -sS -i -w "\\n%{http_code}" \\
                      -X POST "${BACKEND_API_BASE}${LOGIN_PATH}" \\
                      -H "Content-Type: application/json" \\
                      -H "Accept: application/json" \\
                      --data '${loginPayload}'
                  """,
                  returnStdout: true
                ).trim()

                def lines = raw.readLines()
                def code  = lines[-1].toInteger()
                def headPlusBody = lines[0..-2].join('\n')

                def parts = headPlusBody.split("\\r?\\n\\r?\\n", 2)
                def headersText = parts[0]
                def bodyText    = parts.length > 1 ? parts[1] : ""

                def authHeader = headersText.readLines()
                  .find { it.toLowerCase().startsWith('authorization:') }
                def accessToken = authHeader ? authHeader.split(':', 2)[1].trim() : null

                echo "accessToken: ${accessToken}"
                echo "bodyText: ${bodyText}"

                int start = bodyText.indexOf('{')
                int end   = bodyText.lastIndexOf('}')
                if (start < 0 || end < 0) {
                  error "❌ 응답에서 JSON 객체를 찾지 못했습니다. bodyText=${bodyText}"
                }
                def jsonText = bodyText.substring(start, end + 1)

                def parsed = readJSON text: jsonText
                def httpStatus = parsed.httpStatus as String
                def message    = parsed.message  as String
                parsed = null

                if (httpStatus != 'OK') {
                  error "❌ 로그인 실패: httpStatus=${httpStatus}, message=${message ?: jsonText}"
                }
                if (!accessToken) {
                  error "❌ Authorization 헤더 없음"
                }

                def releasePayload = groovy.json.JsonOutput.toJson([
                  url  : (isDev ? "electron-app/VOIDA_Setup_Test_${desktopAppVersion}.exe" : "electron-app/
VOIDA_Setup_${desktopAppVersion}.exe")
                ])

                def raw2 = sh(
                  script: """
                    curl -sS -w "\\n%{http_code}" \\
                      -X POST "${BACKEND_API_BASE}${RELEASE_PATH}" \\
                      -H "Content-Type: application/json" \\
```

```
                -H "authorization: ${accessToken}" \\
                --data '${releasePayload}'
              """,
              returnStdout: true
            ).trim()

            def lines2 = raw2.readLines()
            def code2  = lines2[-1].toInteger()
            def body2  = lines2.size() > 1 ? lines2[0..-2].join('\\n') : ''

            if (code2 == 200) {
              echo "✅ 릴리스 등록 성공 (HTTP ${code2})"
            } else {
              error "❌ 릴리스 등록 실패 (HTTP ${code2})\n${body2}"
              currentBuild.result = 'FAILURE'
            }
        }

        withAWS(credentials: 'aws-s3-credentials', region: 'ap-northeast-2') {
            def s3UploadStatus = sh(
                script: """
                  bash -euo pipefail -c '
                    cd frontend-app/dist
                    aws s3 cp "${newFileName}" "${env.S3_BUCKET}/electron-app/" 2>&1 | tee ../temp-logs/
s3-upload.log
                  '
                """,
                returnStatus: true
            )

            if (s3UploadStatus != 0) {
                def s3ErrorLog = sh(
                    script: """
                        grep 'ERROR\\|error\\|failed' frontend-app/temp-logs/s3-upload.log || echo '[ERROR] S3
업로드 로그 없음'
                    """,
                    returnStdout: true
                ).trim()

                echo "❌ S3 업로드 실패\n${s3ErrorLog}"
                currentBuild.result = 'FAILURE'
                error("S3 업로드 실패")
            }
        }

        echo "✅ Desktop App 빌드 및 S3 업로드 성공"
      }
    }
  }

  post {
    always {
      script {
        if ((env.FINAL_BRANCH == 'develop' || env.FINAL_BRANCH == 'release') && SHOULD_SKIP_MESSA
GE == false) {
          // Jenkins 워크스페이스(빌드 시 다운로드된 파일, 빌드 산출물 등) 삭제
          cleanWs()
        } else {
          echo """
```

```
                ⛔ 해당 JOB은 전송 대상이 아닙니다.
                branch: ${env.FINAL_BRANCH}
                action: ${env.gitlabActionType}
                SHOULD_SKIP_MESSAGE: ${SHOULD_SKIP_MESSAGE}
                """
            }
        }
    }
  }
}
```

6.2 Nginx Host Script

```
sudo vi /etc/nginx/sites-enabled/default
```

```
##
# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##

# Default server configuration
#
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/www/html;

    # Add index.php to the list if you are using PHP
```

```
	index index.html index.htm index.nginx-debian.html;

	server_name _;

	location / {
		# First attempt to serve request as file, then
		# as directory, then fall back to displaying a 404.
		try_files $uri $uri/ =404;
	}

	# pass PHP scripts to FastCGI server
	#
	#location ~ \.php$ {
	#	include snippets/fastcgi-php.conf;
	#
	#	# With php-fpm (or other unix sockets):
	#	fastcgi_pass unix:/run/php/php7.4-fpm.sock;
	#	# With php-cgi (or other tcp sockets):
	#	fastcgi_pass 127.0.0.1:9000;
	#}

		# deny access to .htaccess files, if Apache's document root
		# concurs with nginx's one
		#
		#location ~ /\.ht {
		#	deny all;
		#}
	}


	# Virtual Host configuration for example.com
	#
	# You can move that to a different file under sites-available/ and symlink that
	# to sites-enabled/ to enable it.
	#
	#server {
	#	listen 80;
	#	listen [::]:80;
	#
	#	server_name example.com;
	#
	#	root /var/www/example.com;
	#	index index.html;
	#
	#	location / {
	#		try_files $uri $uri/ =404;
	#	}
	#}

	server {

		# SSL configuration
		#
		# listen 443 ssl default_server;
		# listen [::]:443 ssl default_server;
		#
		# Note: You should disable gzip for SSL traffic.
		# See: https://bugs.debian.org/773332
		#
		# Read up on ssl_ciphers to ensure a secure configuration.
```

```
# See: https://bugs.debian.org/765782
#
# Self signed certs generated by the ssl-cert package
# Don't use them in a production server!
#
# include snippets/snakeoil.conf;

root /var/www/html;

# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;
server_name api.voida.site; # managed by Certbot

location / {
        proxy_pass http://localhost:26281;
        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 /ws/ {
        proxy_pass http://localhost:26281;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
}

# pass PHP scripts to FastCGI server
#
#location ~ \.php$ {
#  include snippets/fastcgi-php.conf;
#
#  # With php-fpm (or other unix sockets):
#  fastcgi_pass unix:/run/php/php7.4-fpm.sock;
#  # With php-cgi (or other tcp sockets):
#  fastcgi_pass 127.0.0.1:9000;
#}

# deny access to .htaccess files, if Apache's document root
# concurs with nginx's one
    #
    #location ~ /\.ht {
    #  deny all;
    #}


    listen [::]:443 ssl; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/api.voida.site/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/api.voida.site/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
    if ($host = api.voida.site) {
        return 301 https://$host$request_uri;
    } # managed by Certbot
```

```
    listen 80 ;
    listen [::]:80 ;
    server_name api.voida.site;
    return 404; # managed by Certbot


}

server {

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /usr/share/nginx/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;
    server_name www.voida.site; # managed by Certbot


    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ /index.html;
    }

    # pass PHP scripts to FastCGI server
    #
    #location ~ \.php$ {
    #  include snippets/fastcgi-php.conf;
    #
    #  # With php-fpm (or other unix sockets):
    #  fastcgi_pass unix:/run/php/php7.4-fpm.sock;
    #  # With php-cgi (or other tcp sockets):
    #  fastcgi_pass 127.0.0.1:9000;
    #}

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #  deny all;
    #}


    listen [::]:443 ssl ipv6only=on; # managed by Certbot
```

```
    listen 443 ssl; # managed by Certbot
    ssl_certificate     /etc/letsencrypt/live/www.voida.site/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/www.voida.site/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}
server {
    if ($host = www.voida.site) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80 ;
    listen [::]:80 ;
    server_name www.voida.site;
    return 404; # managed by Certbot

}
server {
    if ($host = i13e107.p.ssafy.io) {
        return 301 https://www.voida.site$request_uri;
    }
    listen 443 ssl;
    listen [::]:443 ssl;

    server_name i13e107.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/i13e107.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/i13e107.p.ssafy.io/privkey.pem; # managed by Certbot
}

server {

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/www/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;
    server_name lip.voida.site; # managed by Certbot

    location / {
            proxy_pass http://localhost:16281;
            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;
    }

client_max_body_size 20M;

# pass PHP scripts to FastCGI server
#
#location ~ \.php$ {
#  include snippets/fastcgi-php.conf;
#
#  # With php-fpm (or other unix sockets):
#  fastcgi_pass unix:/run/php/php7.4-fpm.sock;
#  # With php-cgi (or other tcp sockets):
#  fastcgi_pass 127.0.0.1:9000;
#}

# deny access to .htaccess files, if Apache's document root
# concurs with nginx's one
    #
    #location ~ /\.ht {
    #  deny all;
    #}


    listen [::]:443 ssl; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/lip.voida.site/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/lip.voida.site/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
    if ($host = lip.voida.site) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80 ;
    listen [::]:80 ;
    server_name lip.voida.site;
    return 404; # managed by Certbot


}

server {

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
```

```
# Self signed certs generated by the ssl-cert package
# Don't use them in a production server!
#
# include snippets/snakeoil.conf;

root /var/www/html;

# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;
server_name api.test.voida.site; # managed by Certbot


location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    proxy_pass http://localhost:26280;
            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 /ws/ {
    proxy_pass http://localhost:26280;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
}

# pass PHP scripts to FastCGI server
#
#location ~ \.php$ {
#  include snippets/fastcgi-php.conf;
#
#  # With php-fpm (or other unix sockets):
#  fastcgi_pass unix:/run/php/php7.4-fpm.sock;
#  # With php-cgi (or other tcp sockets):
#  fastcgi_pass 127.0.0.1:9000;
#}

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #  deny all;
    #}


    listen [::]:443 ssl; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/api.test.voida.site/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/api.test.voida.site/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
    if ($host = api.test.voida.site) {
        return 301 https://$host$request_uri;
```

```
    } # managed by Certbot


    listen 80 ;
    listen [::]:80 ;
    server_name api.test.voida.site;
    return 404; # managed by Certbot


}
server {

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/test/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;
    server_name www.test.voida.site; # managed by Certbot


    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
    }

    # pass PHP scripts to FastCGI server
    #
    #location ~ \.php$ {
    #  include snippets/fastcgi-php.conf;
    #
    #  # With php-fpm (or other unix sockets):
    #  fastcgi_pass unix:/run/php/php7.4-fpm.sock;
    #  # With php-cgi (or other tcp sockets):
    #  fastcgi_pass 127.0.0.1:9000;
    #}

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #  deny all;
    #}
```

```
    listen [::]:443 ssl; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/www.test.voida.site/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/www.test.voida.site/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
    if ($host = www.test.voida.site) {
        return 301 https://$host$request_uri;
    } # managed by Certbot



    listen 80 ;
    listen [::]:80 ;
    server_name www.test.voida.site;
    return 404; # managed by Certbot


}
server {

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/www/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;
    server_name lip.test.voida.site; # managed by Certbot


    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        proxy_pass http://localhost:16280;
                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;

    }

    client_max_body_size 20M;

    # pass PHP scripts to FastCGI server
```

```
    #
    #location ~ \.php$ {
    #  include snippets/fastcgi-php.conf;
    #
    #  # With php-fpm (or other unix sockets):
    #  fastcgi_pass unix:/run/php/php7.4-fpm.sock;
    #  # With php-cgi (or other tcp sockets):
    #  fastcgi_pass 127.0.0.1:9000;
    #}

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #  deny all;
    #}



    listen [::]:443 ssl; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/lip.test.voida.site/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/lip.test.voida.site/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
    if ($host = lip.test.voida.site) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80 ;
    listen [::]:80 ;
    server_name lip.test.voida.site;
    return 404; # managed by Certbot


}


server {
    if ($host = i13e107.p.ssafy.io) {
        return 301 https://app.voida.site$request_uri;
    } # managed by Certbot



    listen 80 ;
    listen [::]:80 ;

    server_name i13e107.p.ssafy.io;
    return 404; # managed by Certbot


}
server {

    # SSL configuration
    #
```

```
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/electron/web;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;
    server_name app.voida.site; # managed by Certbot


    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
    }

    # pass PHP scripts to FastCGI server
    #
    #location ~ \.php$ {
    #  include snippets/fastcgi-php.conf;
    #
    #  # With php-fpm (or other unix sockets):
    #  fastcgi_pass unix:/run/php/php7.4-fpm.sock;
    #  # With php-cgi (or other tcp sockets):
    #  fastcgi_pass 127.0.0.1:9000;
    #}

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
        #location ~ /\.ht {
        #  deny all;
        #}


        listen [::]:443 ssl; # managed by Certbot
        listen 443 ssl; # managed by Certbot
        ssl_certificate /etc/letsencrypt/live/app.voida.site/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/app.voida.site/privkey.pem; # managed by Certbot
        include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

    }
    server {
        if ($host = app.voida.site) {
            return 301 https://$host$request_uri;
        } # managed by Certbot
```

```
    listen 80 ;
    listen [::]:80 ;
    server_name app.voida.site;
    return 404; # managed by Certbot


}

server {

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/www/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;
    server_name jenkins.voida.site; # managed by Certbot


    location / {
            proxy_pass http://localhost:9090;
            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;
    }

    # pass PHP scripts to FastCGI server
    #
    #location ~ \.php$ {
    #  include snippets/fastcgi-php.conf;
    #
    #  # With php-fpm (or other unix sockets):
    #  fastcgi_pass unix:/run/php/php7.4-fpm.sock;
    #  # With php-cgi (or other tcp sockets):
    #  fastcgi_pass 127.0.0.1:9000;
    #}

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #  deny all;
    #}
```

```
    listen [::]:443 ssl; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/jenkins.voida.site/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/jenkins.voida.site/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
    if ($host = jenkins.voida.site) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80 ;
    listen [::]:80 ;
    server_name jenkins.voida.site;
    return 404; # managed by Certbot


}

server {

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/www/html;

# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;
server_name rtc.voida.site; # managed by Certbot

    location / {
      proxy_pass https://localhost:20443;

      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 /openvidu/ws {
          proxy_pass https://localhost:20443/openvidu/ws;

          proxy_http_version 1.1;
    proxy_set_header X-Debug-Version $server_protocol;
```

```
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "Upgrade";
        proxy_set_header Host $host;
        proxy_read_timeout 86400;
}

# pass PHP scripts to FastCGI server
#
#location ~ \.php$ {
#  include snippets/fastcgi-php.conf;
#
#  # With php-fpm (or other unix sockets):
#  fastcgi_pass unix:/run/php/php7.4-fpm.sock;
#  # With php-cgi (or other tcp sockets):
#  fastcgi_pass 127.0.0.1:9000;
#}

# deny access to .htaccess files, if Apache's document root
# concurs with nginx's one
    #
    #location ~ /\.ht {
    #  deny all;
    #}


    listen [::]:443 ssl; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/rtc.voida.site/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/rtc.voida.site/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
    if ($host = rtc.voida.site) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80 ;
    listen [::]:80 ;
    server_name rtc.voida.site;
    return 404; # managed by Certbot


}
```