Configure

설명

특화 프로젝트 파이프라인

Plain text 미리보기

- [ ] Do not allow concurrent builds
- [ ] Do not allow the pipeline to resume if the controller restarts
- [ ] GitHub project

GitLab Connection

- [ ] Use alternative credential
- [ ] Pipeline speed/durability override
- [ ] Preserve stashes from completed builds
- [ ] Throttle builds

오래된 빌드 삭제

Strategy

Log Rotation

빌드 이력 유지 기간(일)

공백일 경우, [보관할 최대갯수] 만큼 기록됩니다.

보관할 최대갯수

if not empty, only up to this number of build records are kept

15

고급

- [ ] 이 빌드는 매개변수가 있습니다

Build Triggers

- [ ] Build after other projects are built
- [ ] Build periodically
- [ ] Build when a change is pushed to GitLab. GitLab webhook URL: http://j11b103.p.ssafy.io:7070/project/sp-project

Enabled GitLab triggers

Push Events

- [ ] Push Events in case of branch delete
- [ ] Opened Merge Request Events
- [ ] Build only if new commits were pushed to Merge Request
- [ ] Accepted Merge Request Events
- [ ] Closed Merge Request Events

Rebuild open Merge Requests

Never

- [ ] Approved Merge Requests (EE-only)

Comments

Comment (regex) for triggering a build

Jenkins please retry a build

고급

- [ ] GitHub hook trigger for GITScm polling
- [ ] Poll SCM
- [ ] Quiet period
- [ ] 빌드를 원격으로 유발 (예: 스크립트 사용)

Advanced Project Options

고급

Pipeline

Definition

Pipeline script

Script

```
pipeline {
    agent any

    environment {
        GIT_BRANCH = ''
        GIT_URL = 'https://pswlove38:az3udQpBxi9b7RDKhQyH@lab.ssafy.com/s11-ai-image-
    }

    stages {
        stage('Checkout Code') {
            steps {
                script {
                    // 안전한 디렉토리 설정
                    sh 'git config --global --add safe.directory /var/jenkins_home/wo

                    GIT_BRANCH = sh(script: 'git rev-parse --abbrev-ref HEAD', return

                    dir('backend'){
                        // .grable 파일이 있으면 삭제
                        sh'''
                        if [ -d ".gradle" ]; then
                            echo "Deleting .gradle directory..."
                            rm -rf .gradle
                        else
                            echo ".gradle directory not found."
                        fi
                        '''
                    }

                    echo "Checked out branch: ${GIT_BRANCH}"
                    sh "git pull origin ${GIT_BRANCH}"
```

```
                    def changes = sh(script: "git diff --name-only HEAD~1 HEAD", retu
                    echo "Changed files:\n${changes}"

                    CHANGED_BACKEND = changes.split('\n').any { it.startsWith('backen
                    CHANGED_FRONTEND = changes.split('\n').any { it.startsWith('front

                    if (CHANGED_BACKEND) {
                        echo "Backend directory has changes."
                    } else {
                        echo "No changes in Backend directory."
                    }

                    if (CHANGED_FRONTEND) {
                        echo "Frontend directory has changes."
                    } else {
                        echo "No changes in Frontend directory."
                    }
                }
            }
        }

        stage('Build and Dockerize') {
            steps {
                script {
                    def parallelTasks = [:]

                    if (CHANGED_BACKEND) {
                        parallelTasks["Backend Build and Dockerize"] = {
                            echo "Building Backend..."
                            dir('backend'){
                                sh 'pwd'
                                sh "chmod +x gradlew"
                                sh "./gradlew clean bootJar"
                                sh "ls -al"
                            }
                            echo "Backend build completed."

                            echo "Making Docker Image Backend..."
                            dir('backend'){
                                // 새로운 도커 이미지 생성
                                sh "docker build -t spring-image:latest ."
                                sh "docker images"

                                // 동일한 이름의 컨테이너가 이미 존재하면 삭제
                                echo "Checking existing spring containers..."
                                sh """
                                if [ \$(docker ps -a -q -f name=spring-container) ];
                                    docker stop spring-container
                                    docker rm spring-container
                                fi
                                """

                                // 새로운 컨테이너 실행
                                sh "docker run -d --name spring-container --network a
                                sh "docker ps -a | grep spring-container"
                                sh "docker logs spring-container"

                                // 기준 <none> 이미지들 삭제
                                sh "docker image prune -f"
                            }
                            echo "Backend Docker image created & run successfully."
                        }
                    }

                    if (CHANGED_FRONTEND) {
                        parallelTasks["Frontend Build and Dockerize"] = {
                            // 프론트엔드 도커 이미지 생성
                            echo "Making Docker Image Frontend..."
                            dir('frontend'){
                                // 프론트 도커 이미지 생성
                                sh "docker build -t react-image:latest ."

                                // 동일한 이름의 컨테이너가 이미 존재하면 삭제
                                echo "Checking existing react containers..."
                                sh """
                                if [ \$(docker ps -a -q -f name=react-container) ]; t
                                    docker stop react-container
                                    docker rm react-container
                                fi
                                """
                                // 기존에 있던 이미지 삭제
                                sh "docker image prune -f"

                                // 디버깅을 위한 로그
                                sh "docker images"

                                // 리액트 컨테이너 실행
                                sh "docker run -d -p 3000:80 --name react-container -

                                // 컨테이너 로그 확인
                                sh "docker logs react-container"
                            }
                            echo "Frontend Docker image created successfully."
                        }
                    }

                    if (parallelTasks) {
                        parallel parallelTasks
                    } else {
                        echo "No changes detected in Backend or Frontend. Skipping bu
                    }
                }
            }
        }
    }

    post {
        success {
            script {
                echo "Pipeline completed successfully."
                if (CHANGED_BACKEND) {
                    echo "Backend build and Docker image creation completed successfu
                } else {
```

```
                    echo "No changes in Backend, no build or Docker image creation pe
                }

                if (CHANGED_FRONTEND) {
                    echo "Frontend build and Docker image creation completed successf
                } else {
                    echo "No changes in Frontend, no build or Docker image creation p
                }
            }
        }
        failure {
            script {
                echo "Pipeline failed. Please check the logs for more details."
            }
        }
    }
}

```

Use Groovy Sandbox

Pipeline Syntax

REST API Jenkins 2.462.3