```
pipeline {
    agent { label 'build-server-1' }

    triggers {
        cron('H 2 * * *') // Daily at ~2AM
    }

    parameters {
        choice(name: 'FMEFLOW_SOURCE_ENV', choices: [
            'Dev (http://<DevFlow>.compute.amazonaws.com)',
            'Prod (http://<ProdFlow>.compute.amazonaws.com)'
        ], description: 'Select source FME Flow environment')

        choice(name: 'FMEFLOW_TARGET_ENV', choices: [
            'Prod (http://<ProdFlow>.compute.amazonaws.com)',
            'Dev (http://<DevFlow>.compute.amazonaws.com)'
        ], description: 'Select target FME Flow environment')
    }

    environment {
        SOURCE_CONFIG = "${WORKSPACE}/source.yaml"
        TARGET_CONFIG = "${WORKSPACE}/target.yaml"
        PASSWORD_FILE = "${WORKSPACE}/fmeflow_pass.txt"
    }

    stages {
        stage('Resolve Environment URLs') {
            steps {
                script {
                    def envMap = [
                        'Dev'  : 'http://<DevFlow>.compute.amazonaws.com',
                        'Prod' : 'http://<ProdFlow>.compute.amazonaws.com'
                    ]
                    def sourceLabel = params.FMEFLOW_SOURCE_ENV.tokenize(' ')[0]
                    def targetLabel = params.FMEFLOW_TARGET_ENV.tokenize(' ')[0]

                    env.FMEFLOW_SOURCE_URL = envMap[sourceLabel]
                    env.FMEFLOW_TARGET_URL = envMap[targetLabel]

                    echo " Source URL: ${env.FMEFLOW_SOURCE_URL}"
                    echo " Target URL: ${env.FMEFLOW_TARGET_URL}"
                }
            }
        }
```

```
    stage('Login to Source FME Flow') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'fmeflow-login-source',
usernameVariable: 'FME_USER', passwordVariable: 'FME_PASS')]) {
          script {
            writeFile file: env.PASSWORD_FILE, text: FME_PASS
            sh """
            fmeflow login ${env.FMEFLOW_SOURCE_URL} \
              --user ${FME_USER} \
              --password-file ${PASSWORD_FILE} \
              --config ${env.SOURCE_CONFIG}
            """
            sh "rm -f ${env.PASSWORD_FILE}"
          }
        }
      }
    }

    stage('Find and Download Updated Projects') {
      steps {
        script {
          def json = sh(script: "fmeflow projects --owner admin --output json --config
${env.SOURCE_CONFIG}", returnStdout: true).trim()
          def projects = readJSON text: json

          def sdf = new java.text.SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSSX")
          sdf.setTimeZone(TimeZone.getTimeZone("UTC"))
          def now = new Date()
          def updatedProjects = []

          projects.items.each { project ->
            def updated = sdf.parse(project.lastUpdated)
            def age = now.time - updated.time
            if (age < (24 * 60 * 60 * 1000)) {
              echo " Project '${project.name}' was updated in the last 24 hours."
              updatedProjects << project.name
              sh "fmeflow projects download --name '${project.name}' -f
'${project.name}.fsproject' --config ${env.SOURCE_CONFIG}"
            }
          }

          if (updatedProjects.isEmpty()) {
            echo " No updated projects found in the last 24 hours."
```

```
            }

            writeFile file: 'updatedProjects.txt', text: updatedProjects.join('\n')
        }
    }
}

stage('Login to Target FME Flow') {
    steps {
        withCredentials([usernamePassword(credentialsId: 'fmeflow-login-target',
usernameVariable: 'FME_USER', passwordVariable: 'FME_PASS')]) {
            script {
                writeFile file: env.PASSWORD_FILE, text: FME_PASS
                sh """
                fmeflow login ${env.FMEFLOW_TARGET_URL} \
                  --user ${FME_USER} \
                  --password-file ${PASSWORD_FILE} \
                  --config ${env.TARGET_CONFIG}
                """
                sh "rm -f ${env.PASSWORD_FILE}"
            }
        }
    }
}

 stage('Upload Projects to Target') {
steps {
    script {
        def updatedProjects = readFile('updatedProjects.txt').split('\n')
        updatedProjects.each { name ->
            if (name?.trim()) {
                def file = "${name}.fsproject"
                echo " Uploading ${file} to target instance..."

                // Extract token from target.yaml
                def token = sh(
                    script: "grep '^token:' ${env.TARGET_CONFIG} | cut -d ':' -f2 | xargs",
                    returnStdout: true
                ).trim()

                if (!token) {
                    error " Failed to extract token from ${env.TARGET_CONFIG}"
                }
```

```
                    // Upload the project and capture the Location header
                    def uploadResponse = sh(
                        script: """
                        curl -s -D - -o /dev/null -X POST \\

"${env.FMEFLOW_TARGET_URL}/fmeapiv4/projects/imports/upload?skipPreview=true" \\
                          -H 'accept: */*' \\
                          -H "Authorization: fmetoken token=${token}" \\
                          -H 'Content-Type: multipart/form-data' \\
                          -F 'file=@${file}'
                        """,
                        returnStdout: true
                    )

                    // Wait 15 seconds before triggering the import
                    sh "sleep 15"

                    // Extract the ID from the Location header
                    def locationLine = uploadResponse.readLines().find {
it.toLowerCase().startsWith('location:') }
                    def importId = locationLine?.tokenize('/')?.last()?.trim()

                    if (!importId) {
                        error " Failed to extract import ID from response headers."
                    }

                    echo " Import ID received: ${importId}"

                    // Run the import with the extracted ID
                    sh """
                    curl -s -X POST \\
                      "${env.FMEFLOW_TARGET_URL}/fmeapiv4/projects/imports/${importId}/run" \\
                      -H 'accept: */*' \\
                      -H "Authorization: fmetoken token=${token}" \\
                      -H 'Content-Type: application/json' \\
                      -d '{
                        "fallbackOwnerID": "<input Here>",
                        "overwrite": true,
                        "pauseNotifications": true,
                        "disableItems": true,
                        "notification": {
                          "type": "TOPIC",
                          "successTopic": "string",
                          "failureTopic": "string"
```

```
                    }
                  }'
                """
            }
        }
    }
}

    post {
        success {
            echo ' Project sync completed successfully.'
        }
        failure {
            echo ' Project sync failed. Check logs for details.'
        }
        cleanup {
            sh 'rm -f *.fsproject updatedProjects.txt ${SOURCE_CONFIG} ${TARGET_CONFIG}
${PASSWORD_FILE}'
        }
    }
}
```