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

    parameters {
        string(name: 'FMEFLOW_URL', defaultValue:
'http://<FMEFlowURL>.ca-central-1.elb.amazonaws.com/', description: 'FME Flow URL')
        string(name: 'STANDARD_ENGINE_COUNT', defaultValue: '2', description: 'Number of
Standard Engines')
        string(name: 'DYNAMIC_ENGINE_COUNT', defaultValue: '2', description: 'Number of
Dynamic Engines')
        string(name: 'FIRST_NAME', defaultValue: '<FirstName>', description: 'License request
first name')
        string(name: 'LAST_NAME', defaultValue: '<LastName>', description: 'License request last
name')
        string(name: 'EMAIL', defaultValue: '<EmailAddress>', description: 'License request email')
        string(name: 'COMPANY', defaultValue: '<CompanyName>', description: 'License request
company')
        string(name: 'SERIAL_NUMBER', defaultValue: '<LicenseSerialNumber>', description:
'License serial number')
    }

    environment {
        CLI_BIN = '/usr/local/bin/fmeflow'
        CLI_CONFIG_FILE = "${WORKSPACE}/.fmeflow-cli.yaml"
        PASSWORD_FILE = "${WORKSPACE}/fmeflow_pass.txt"
    }

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

```
    }

    stage('Health Check') {
      steps {
        sh "${CLI_BIN} healthcheck --config ${CLI_CONFIG_FILE}"
      }
    }

    stage('Extract API Token') {
      steps {
        script {
          def token = sh(script: "awk '/^token:/{print \$2}' ${CLI_CONFIG_FILE}",
returnStdout: true).trim()
          env.FMEFLOW_API_TOKEN = token
          echo "Token extracted and stored in environment variable."
        }
      }
    }

    stage('Request License') {
      steps {
        sh """
        ${CLI_BIN} license request \
          --first-name "${params.FIRST_NAME}" \
          --last-name "${params.LAST_NAME}" \
          --email "${params.EMAIL}" \
          --company "${params.COMPANY}" \
          --serial-number "${params.SERIAL_NUMBER}" \
          --config ${CLI_CONFIG_FILE}
        """
      }
    }

    stage('Check License Status') {
      steps {
        script {
          echo "Waiting 30 seconds before checking license status..."
          sleep time: 30, unit: 'SECONDS'
        }
        sh "${CLI_BIN} license request status --config ${CLI_CONFIG_FILE}"
      }
    }

    stage('Scale & Summarize Engine Hosts') {
```

```
        steps {
            script {
                def baseUrl = "${params.FMEFLOW_URL}".replaceAll('/+$', '')
                def token = env.FMEFLOW_API_TOKEN
                def enginesUrl = "${baseUrl}/fmeapiv4/enginehosts?fmetoken=${token}"
                def response = sh(script: "curl -s -X GET '${enginesUrl}'", returnStdout: true).trim()
                def parsed = readJSON text: response
                def engines = parsed.items

                def summary = []

                engines.each { engine ->
                    def name = engine.name
                    def scaleUrl =
"${baseUrl}/fmeapiv4/enginehosts/${name}/engines/scale?fmetoken=${token}"
                    def preCheckUrl =
"${baseUrl}/fmeapiv4/enginehosts/${name}/engines?fmetoken=${token}"

                    def preResponse = sh(script: "curl -s -X GET '${preCheckUrl}'", returnStdout:
true).trim()
                    def preParsed = readJSON text: preResponse
                    def preCount = preParsed.size()

                    ['standard', 'dynamic'].each { forcedType ->
                        def forcedCount = forcedType == 'dynamic' ?
params.DYNAMIC_ENGINE_COUNT : params.STANDARD_ENGINE_COUNT
                        def forcePayload = """{
                          \"type\": \"${forcedType}\",
                          \"engineCount\": ${forcedCount}
                        }"""

                        echo "Scaling ${name} for type '${forcedType}' to ${forcedCount} engines"

                        sh """
                        curl -s -X POST '${scaleUrl}' \
                            -H 'Content-Type: application/json' \
                            -d '${forcePayload}'
                        """

                        summary << [name: name, type: forcedType, from: preCount, to: forcedCount]
                    }
                }

                echo "\n====== Engine Scaling Summary ======"
```

```
                summary.each {
                    echo "✔ ${it.name} (${it.type}) — scaled from ${it.from} to ${it.to} engines"
                }
                echo "=====================================\n"
            }
        }
    }
    stage('Run FME Workspace') {
        steps {
            sh """
            ${CLI_BIN} run \
              --repository Samples \
              --workspace austinApartments.fmw \
              --wait \
              --config ${CLI_CONFIG_FILE}
            """
        }
    }

    stage('Use Token (Optional)') {
        steps {
            echo "Token available as env var (not printed for security)."
        }
    }
  }

  post {
    always {
        sh "rm -f ${PASSWORD_FILE}"
    }
  }
}
```