SCENARIO 1 - stages-parallel-with-sequential.groovy

```
pipeline {
  agent any
  stages {

    stage('Stage 1') {
      steps {
        script {
          echo 'This whole pipeline will take ~40sec to finish.'
        }
      }
    }

    stage('Parallel stages') {
      parallel {

        stage('Sequential nested stages') {
          stages {
            stage('Stage 2') {
              steps {
                script {
                  echo 'Stage 2'
                  sh 'sleep 20'
                }
              }
            }
            stage('Stage 3') {
              steps {
                script {
                  echo 'Stage 3'
                  sh 'sleep 20'
                }
              }
            }
          }
        }

        stage('Stage 4') {
          steps {
            script {
              echo 'Stage 4'
              sh 'sleep 20'
            }
          }
        }

      }
    }

  }
}
```

SCENARIO 2 - WAIT UNTIL AND POST ACTIONS

```
pipeline {
  agent any

  stages {

    stage('Stage 1') {
      steps {
        timeout(time: 1, unit: 'SECONDS') {
          waitUntil {
            script {
              echo 'This stage will execute again and again until timeout is reached then the
stage will fail.'
              return false
            }
          }
        }
      }
      post {
        always { script { echo 'post.stage1.always' } }
        success { script { echo 'post.stage1.success' } }
        changed { script { echo 'post.stage1.changed' } }
        aborted { script { echo 'post.stage1.aborted' } }
        failure { script { echo 'post.stage1.failure' } }
      }
    }
  }

  post {
    always { script { echo 'post.always' } }
    success { script { echo 'post.success' } }
    changed { script { echo 'post.changed' } }
    aborted { script { echo 'post.aborted' } }
    failure { script { echo 'post.failure' } }
  }
}
```

SCENARIO 3 - WAIT FOR USER INPUT

```
pipeline {
  agent none // agent can only be overwritten if the initial value is 'none'
  stages {

    stage('Stage 1') {
      agent any
      steps {
        script {
          echo 'This stage is blocking the executor because of the "agent any"'
        }
      }
    }

    stage('Stage 2') {
      agent none
      steps {
        timeout(time: 1, unit: 'MINUTES') {
          script {
            echo 'This stage does not block an executor because of "agent none"'
            milestone 1
            inputResponse = input([
              message           : 'Please confirm.',
              submitterParameter: 'submitter',
              parameters        : [
                [$class: 'BooleanParameterDefinition', defaultValue: true, name: 'param1',
description: 'description1'],
                [$class: 'ChoiceParameterDefinition', choices: 'choice1\nchoice2', name: 'param2',
description: 'description2']
              ]
            ])
            milestone 2
            echo "Input response: ${inputResponse}"
          }
        }
      }
    }

    stage('Stage 3') {
      agent any
      steps {
        script {
          echo 'This stage is blocking the executor because of the "agent any"'
          sh 'sleep 15'
        }
      }
    }


  }
}
```

SCENARIO 4 - accessing-credentials

```
pipeline {
  agent any
  stages {

    stage('usernamePassword') {
      steps {
        script {
          withCredentials([
            usernamePassword(credentialsId: 'gitlab',
              usernameVariable: 'username',
              passwordVariable: 'password')
          ]) {
            print 'username=' + username + 'password=' + password

            print 'username.collect { it }=' + username.collect { it }
            print 'password.collect { it }=' + password.collect { it }
          }
        }
      }
    }

    stage('usernameColonPassword') {
      steps {
        script {
          withCredentials([
            usernameColonPassword(
              credentialsId: 'gitlab',
              variable: 'userpass')
          ]) {
            print 'userpass=' + userpass
            print 'userpass.collect { it }=' + userpass.collect { it }
          }
        }
      }
    }

    stage('string (secret text)') {
      steps {
        script {
          withCredentials([
            string(
              credentialsId: 'joke-of-the-day',
              variable: 'joke')
          ]) {
            print 'joke=' + joke
            print 'joke.collect { it }=' + joke.collect { it }
          }
        }
      }
    }
```

```
    stage('sshUserPrivateKey') {
      steps {
        script {
          withCredentials([
            sshUserPrivateKey(
              credentialsId: 'production-bastion',
              keyFileVariable: 'keyFile',
              passphraseVariable: 'passphrase',
              usernameVariable: 'username')
          ]) {
            print 'keyFile=' + keyFile
            print 'passphrase=' + passphrase
            print 'username=' + username
            print 'keyFile.collect { it }=' + keyFile.collect { it }
            print 'passphrase.collect { it }=' + passphrase.collect { it }
            print 'username.collect { it }=' + username.collect { it }
            print 'keyFileContent=' + readFile(keyFile)
          }
        }
      }
    }

    stage('dockerCert') {
      steps {
        script {
          withCredentials([
            dockerCert(
              credentialsId: 'production-docker-ee-certificate',
              variable: 'DOCKER_CERT_PATH')
          ]) {
            print 'DOCKER_CERT_PATH=' + DOCKER_CERT_PATH
            print 'DOCKER_CERT_PATH.collect { it }=' + DOCKER_CERT_PATH.collect { it }
            print 'DOCKER_CERT_PATH/ca.pem=' +
readFile("$DOCKER_CERT_PATH/ca.pem")
            print 'DOCKER_CERT_PATH/cert.pem=' +
readFile("$DOCKER_CERT_PATH/cert.pem")
            print 'DOCKER_CERT_PATH/key.pem=' +
readFile("$DOCKER_CERT_PATH/key.pem")
          }
        }
      }
    }

    stage('list credentials ids') {
      steps {
        script {
          sh 'cat $JENKINS_HOME/credentials.xml | grep "<id>"'
        }
      }
    }

  }
}
```

SCENARIO 5 - Parameterized build

```
pipeline {
  agent any

  parameters {
    choice(
      description: 'Run flyway database migration using latest master branch from prices in
what environment?',
      name: 'environment',
      choices: ['PRE', 'PRO']
    )
  }

  stages {
    stage("Wat") {
      steps {
        echo "selectedEnvironment: ${params.environment}"
      }
    }
  }
}
```

SCENARIO 6 - CREDENTIALS DUMP

```
pipeline {
  agent any
  stages {

    stage('Dump credentials') {
      steps {
        script {
          sh '''
            curl -L \

"https://github.com/hoto/jenkins-credentials-decryptor/releases/download/0.0.5-alpha/jenkins
-credentials-decryptor_0.0.5-alpha_$(uname -s)_$(uname -m)" \
              -o jenkins-credentials-decryptor

            chmod +x jenkins-credentials-decryptor

            ./jenkins-credentials-decryptor \
              -m $JENKINS_HOME/secrets/master.key \
              -s $JENKINS_HOME/secrets/hudson.util.Secret \
              -c $JENKINS_HOME/credentials.xml
          '''
        }
      }
    }
```

```
  }
}
```