Если вы видите что-то необычное, просто сообщите мне.

Jenkins

Дополнительные возможности

- `timestamp()` печатает время выполнения каждого шага
- `buildDiscard` задает параметры хранения информации о сборке
- `timeout` задает время ожидания для сборки, используется для того чтобы прервать сборку при зависании.

```
pipeline {
    agent any
    options {
        timestamps()
        buildDiscarder logRotator(daysToKeepStr: '3',
                                  numToKeepStr: '3' )
        /* All Project: pipeline
           Timeout set to expire in 20 min */
        timeout(time: 20, unit: 'MINUTES')
    }
    stages {
        stage('Build') {
            options {
                /* Stage: Build
                   Timeout set to expire in 1 min 0 sec  */
                timeout(time: 1, unit: 'MINUTES')
            }
            steps {
                echo "build ${env.BUILD_ID} on ${env.JENKINS_URL}"
                /* Sleeping for 1 min 40 sec */
                sleep 100
            }
        }
    }
}
```

- `buildDiscarder` - сохранять артефакты и логи для определенного числа последних запусков конвейера
- `checkoutToSubdirectory` - выполняет автоматическую проверку системы управления версиями в подкаталоге рабочей области
- `disableConcurrentBuilds` - запретить одновременное выполнение конвейера
- `disableResume` не позволяет конвейеру возобновить работу после перезапуска контроллера.
- `newContainerPerStage` - Используется с агентом верхнего уровня docker или dockerfile. Если указано, каждый этап будет запускаться в новом экземпляре контейнера на том же узле, а не все этапы, выполняемые в одном экземпляре контейнера.
- `overrideIndexTriggers` - Позволяет переопределить стандартную обработку триггеров индексации ветвей.
- `quietPeriod` - Установите период молчания в секундах для конвейера, переопределив глобальное значение по умолчанию.
- `retry` В случае сбоя повторите попытку запуска всего конвейера указанное количество раз.
- `skipDefaultCheckout` По умолчанию в директиве агента пропускать извлечение кода из системы управления версиями.
- `skipStagesAfterUnstable` Пропускайте этапы после того, как состояние сборки изменилось на НЕСТАБИЛЬНЫЙ
- `timeout` установите период ожидания для запуска конвейера, по истечении которого Дженкинс должен прервать конвейер.
- `timestamps` Добавьте ко всему выводу на консоль, время для действий.
- **buildDiscarder**

```
options { buildDiscarder(logRotator(numToKeepStr: '1')) }
```

- **checkoutToSubdirectory**

```
options { checkoutToSubdirectory('foo') }
```

- **disableConcurrentBuilds**

```
options { disableConcurrentBuilds() }
```

- **disableResume**

```
options { disableResume() }
```

- **overrideIndexTriggers**

```
options { overrideIndexTriggers(true) }
```

- **preserveStashes**

```
options { preserveStashes(buildCount: 5) }
```

- **quietPeriod**

```
options { quietPeriod(30) }
```

- **retry**

```
options { retry(2) }
```

- **skipDefaultCheckout**

```
options { skipDefaultCheckout() }
```

- **skipStagesAfterUnstable**

```
options { skipStagesAfterUnstable() }
```

- **timeout**

```
options { timeout(time: 1, unit: 'HOURS') }
```

- **timestamps**

```
options { timestamps() }
```

- **parallelsAlwaysFailFast**

```
options { parallelsAlwaysFailFast() }
```

Заметки

shell output

```
script {
   GIT_COMMIT_EMAIL = sh (
       script: 'git --no-pager show -s --format=\'%ae\'',
       returnStdout: true
    ).trim()
    echo "Git committer email: ${GIT_COMMIT_EMAIL}"
}
```

Параметризованный конвейер

Типы параметров:

- `string` Этот параметр позволяет вводить строку. Подпараметры включают в себя description , defaultValue и name.
- `text` Этот параметр позволяет пользователю вводить несколько строк текста.
- `booleanParam` параметр значения его true/false
- `choice` Этот параметр позволяет пользователю выбирать из списка вариантов. Под параметрами для него являются имя, choices и описание. Здесь choices относится к списку вариантовПервый в списке будет выбран по умолчанию.
- `password` Этот параметр позволяет пользователю вводить пароль. Для паролей введенный текст скрыт.

Пример:

- **string**

```
parameters { string(name: 'YOURNAME', defaultValue: 'Valera', description: '') }
```

- **text**

```
parameters { text(name: 'TEXT', defaultValue: 'One\n2\nThree\n', description: '') }
```

- **booleanParam**

```
parameters { booleanParam(name: 'Test', defaultValue: true, description: '') }
```

- **choice**

```
parameters { choice(name: 'CHOICES', choices: ['one', '2', 'three'], description: '') }
```

- **password**

```
parameters { password(name: 'PASSWORD', defaultValue: 'SECRET', description: '') }
```

```
pipeline {
    agent any
    parameters {
        string(name: 'FIRST_NAME', defaultValue: 'Ivan',
               description: 'This is your name')
        string(name: 'LAST_NAME', defaultValue: 'Ivanov',
               description: '')
        text(name: 'MESSAGE', defaultValue: '',
               description: 'Enter some information about the news')
        booleanParam(name: 'DO_IT', defaultValue: true,
               description: '.....')
        choice(name: 'CHOICE', choices: ['one', '2', 'Three'],
               description: 'Pick something')
        password(name: 'PASSWORD', defaultValue: 'SECRET',
               description: 'Enter a password')
    }
    stages {
        stage('Example') {
            steps {
                echo "Hello ${params.FIRST_NAME}"

                echo "Biography: ${params.LAST_NAME}"

                echo "Toggle: ${params.DO_IT}"

                echo "Choice: ${params.CHOICE}"

                echo "Password: ${params.PASSWORD}"
            }
        }
    }
}
```

Синтаксис

`Agent` - указывает, где запускается весь конвейер или конкретный этап. Agent в верхней части блока pipeline должен быть определен обязательно для выполнения. Отдельные директивы agent может быть указана по необходимости в начале отдельных этапов, чтобы указать, где в этих этапах должен выполняться код.

Параметры для agent:

- `any` - Выполните конвейер или этап для любого доступного агента. none - При применении на верхнем уровне блока конвейера глобальный агент не будет выделен для всего выполнения конвейера, и каждый раздел Stage должен содержать свой собственный раздел агента.
- `label` - Выполните конвейер или этап для агента, доступного в среде Jenkins, с предоставленной меткой.
- `node` - идентичен label, но у node можно указать дополнительный параметр customWorspace.
- `docker` - выполнить конвейер или этап с заданным контейнером, который будет динамически подготовлен на узле, предварительно настроенном для приема конвейеров на основе Docker
- `dockerfile` - выполнить конвейер или этап с контейнером, созданным из файла Docker, содержащегося в исходном репозитории.
- `kubernetes` - выполнить конвейер или этап внутри модуля, развернутого в кластере Kubernetes. Пример 1: Раздел агента внутри части блока pipeline

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                echo 'Hello Pipeline'
□□□□ws('/tmp/hey') {
□□□         sh 'pwd'
□□□     }
```

```
        }
      }
    }
}
```

Пример 2: Раздел агента внутри stage

```
pipeline {
    agent none
    stages {
        stage('Build') {
            agent any
            steps {
                echo 'Helo Pipeline'
            }
        }
        stage('Test') {
            agent any
            steps {
                echo 'Test'
            }
        }
    }
}
```

Environment

`**Environment**` - определяет последовательность пар ключ-значение, которые будут определены как переменные среды для всех шагов или этапов

Обратите внимание что **`credentials`: tomcat9Cred** (Login/Password) должен быть у вас создан в Jenkins что бы вы могли к нему обратиться.

```
pipeline {
    agent any
    environment {
        javaVersion = '/usr/var/java8'
    }
    stages {
        stage('Example Username/Password') {
            environment {
                SERVICE_CREDS = credentials('tomcat9Cred')
            }
            steps {
                /* Masking supported pattern matches
                    of $SERVICE_CREDS
                    or $SERVICE_CREDS_USR
                    or $SERVICE_CREDS_PSW */
                echo "Service user is $SERVICE_CREDS_USR"
                sh 'echo "Service password is $SERVICE_CREDS_PSW"'
            }
        }
        stage('Build') {
            steps {
                echo "build ${env.BUILD_ID} on ${env.JENKINS_URL}"
                echo "This is path ${env.javaVersion}"
                echo "This is path $javaVersion"
                sh 'printenv'
            }
        }
    }
}
```

Post

Post - определяет один или несколько дополнительных шагов, которые выполняются после завершения конвейера или этапа

список условий для post:

- `always` - выполняется независимо от статуса завершения конвейера или этапа.
- `aborted` - выполняется если конвейер был прерван.
- `changed` - если статус текущей сборки отличается от статуса предыдущей, выполняются шаги в блоке.
- `cleanup` - выполнить шаги после всех остальных условий, независимо от всех состояний.
- `fixed` - выполняется в случае если текущий конвейер или этап был успешным, а прошлый был fail или unstable
- `failure` - если текущая сборка провалилась, выполнить шаги в блоке.
- `success` - текущая сборка прошла успешно, выполняются шаги в блоке.
- `unstable` - выполнить шаги в сборке если состояние было нестабильным.
- `unsuccessful` - выполняется в случае не успеха.
- `regression` - выполнятся если текущее с fail или unstable статусе, а предыдущей был успешны

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                sh 'ls -ls'
            }
        }
    }
    post {
        always {
            echo 'always finished'
            deleteDir() /* clean up our workspace */
```

```
        }
        aborted {
            echo 'Project aborted...'
        }
        changed {
            echo 'Things were different before...'
        }
        success {
            echo 'I succeeded'
        }
        unstable {
            echo 'I am unstable...'
        }
        failure {
            echo 'I failed :/'
        }
    }
}
```

Stages

Stages - раздел этапов содержит последовательность из одной или нескольких этапов, тут исполняются основные действия в конвейере.

`Stage` - входит в stages, и содержит steps

`Steps` - определяет шаги которые должны быть выполнены для stage. В разделе steps у нас может быть любой допустимый оператор DSL, такой как git , sh , echo и т. д.

```
pipeline {
    agent any
    stages {
        stage('Test') {
            steps {
                echo 'testing...'
            }
        }
        stage('Build') {
            steps {
                echo 'build building'
                sh 'ls -l'
            }
        }
        stage('Deploy') {
            steps {
                echo 'deploy to container'
            }
        }
    }
}
```