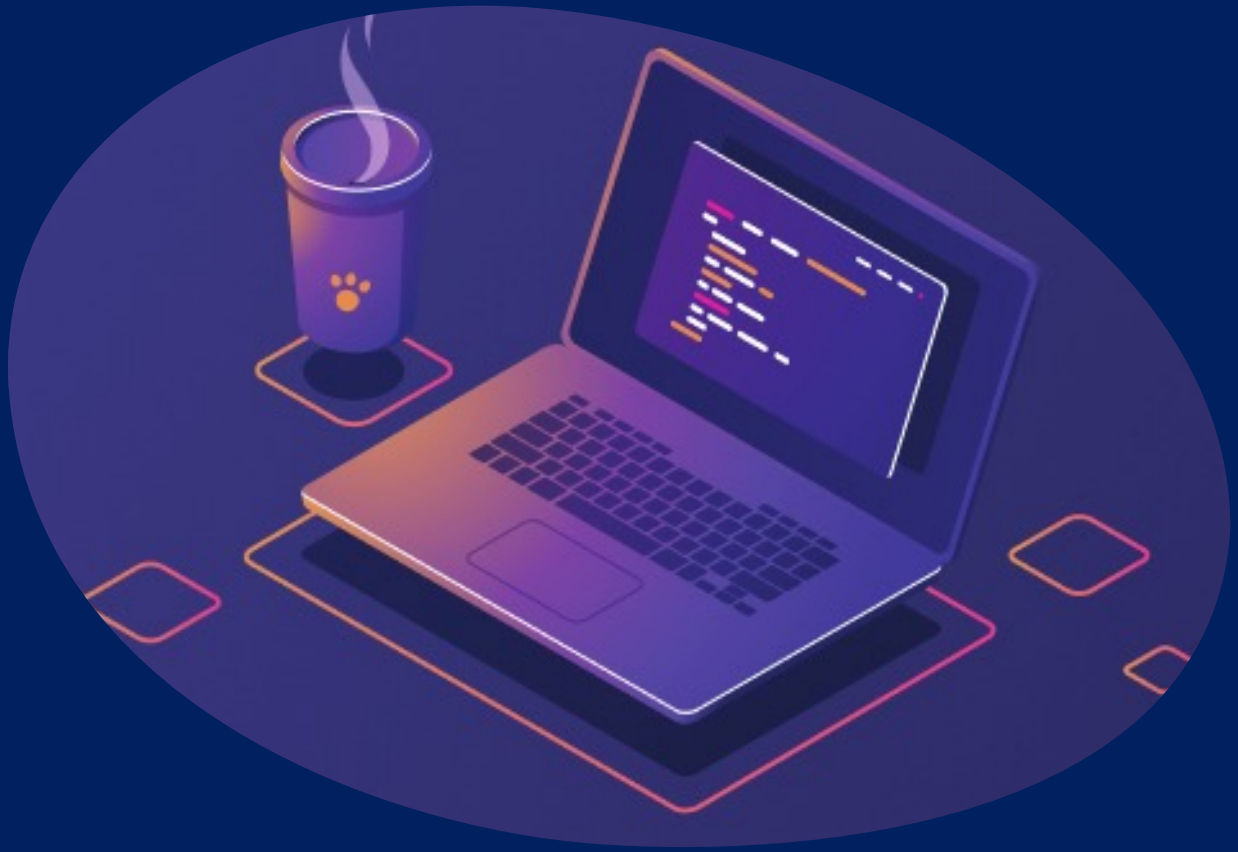

Week 7 : SOFTWARE DEVELOPMENT TOOLS AND ENVIRONMENTS

Parallel steps in a stage

Parallel steps in a stage

```
pipeline {
    agent any
    stages {
        stage('pre -build') {
            steps {
                bat 'echo Pre-build'
            }
        }
        stage('build') {
            steps {
                bat 'echo Build in progress.'
            }
        }
        stage('Unit tests') {
            steps {
                bat 'echo Running unit tests'
            }
        }
        stage('deploy') {
            steps {
                bat 'echo Deploying build'
            }
        }
        stage('Regression tests') {
            steps {
                parallel{
                        stage('chrome') {
                            steps {
                                bat 'echo Running E2E tests on chrome'
                            }
                        }
                        stage('firefox') {
                            steps {
                                bat 'echo Running E2E tests on chrome'
                            }
                        }
                        stage('safari') {
                            steps {
                                bat 'echo Running E2E tests on chrome'
                            }
                        }
                }
            }
        }
        stage('Release to prod') {
            steps {
                bat 'echo Releasing to prod'
            }
        }
    }
 post {
        always {
            echo 'Cleanup after everything!'
        }
    }
}
```

Stage View

	pre -build	build	Unit tests	deploy	Regression tests	chrome	firefox	safari	Release to prod	Declarative: Post Actions
Average stage times: (Average full run time: ~2s)	320ms	303ms	307ms	317ms	54ms	348ms	394ms	404ms	320ms	49ms
#1 ก.พ. 25 09:22 No Changes	320ms	303ms	307ms	317ms	54ms	348ms	394ms	404ms	320ms	49ms

Environment Variables

Environment Variables

The environment directive specifies a sequence of key-value pairs which will be defined as environment variables for all steps, or stage-specific steps, depending on where the environment directive is located within the Pipeline.

```
 environment {
    fname = 'kamal'
}
```

Scope

- If defined at pipeline level, it will apply to all steps within the Pipeline.

```
pipeline {
    agent any
    environment {
        fname = 'kamal'
    }
    stages {
        stage('Example') {
            steps {
                sh 'echo "HELLO ${fname}"'
            }
        }
    }
}
```

- If defined within a stage, it will only be accessible to steps within the stage.

```
pipeline {
    agent any
    stages {
        stage('one') {
            environment {
                fname = 'kamal'
            }
            steps {
                sh 'echo "HELLO ${fname}"'
            }
        }
        stage('two') {
            steps {
                sh 'echo "Hello ${fname}"'
            }
        }
    }
}
```

This directive supports a special helper method credentials() which can be used to access pre-defined Credentials by their identifier in the Jenkins environment.

If you want to access credentials stored in Jenkins Credential Manager through the pipeline, then you need to use this credentials() method.

And once you put that in the environment variable (say GIT_CREDS), you can access username and password with the names "GIT_CREDS_USR" and "GIT_CREDS_PSW".

```
pipeline {
    agent any
    environment {
        GIT_CREDS = credentials('github')
    }
    stages {
        stage('abc') {
            steps {
                sh 'echo "Git user is $GIT_CREDS_USR"'
                sh 'echo "Git password is $GIT_CREDS_PSW"'
                sh 'echo " Current build number is $BUILD_NUMBER"'
            }
        }
    }
}
```

http://IP_Address:8080/pipeline-syntax/globals#env

A set of environment variables are made available to all Jenkins projects, including Pipelines. The following is a general list of variable names that are available.

BRANCH_NAME
For a multibranch project, this will be set to the name of the branch being built, for example in case you wish to deploy to production from `master` but not from feature branches; if corresponding to some kind of change request, the name is generally arbitrary (refer to `CHANGE_ID` and `CHANGE_TARGET`).

BRANCH_IS_PRIMARY
For a multibranch project, if the SCM source reports that the branch being built is a primary branch, this will be set to `"true"`; else unset. Some SCM sources may report more than one branch as a primary branch while others may not supply this information.

CHANGE_ID
For a multibranch project corresponding to some kind of change request, this will be set to the change ID, such as a pull request number, if supported; else unset.

CHANGE_URL
For a multibranch project corresponding to some kind of change request, this will be set to the change URL, if supported; else unset.

CHANGE_TITLE
For a multibranch project corresponding to some kind of change request, this will be set to the title of the change, if supported; else unset.

CHANGE_AUTHOR
For a multibranch project corresponding to some kind of change request, this will be set to the username of the author of the proposed change, if supported; else unset.

CHANGE_AUTHOR_DISPLAY_NAME
For a multibranch project corresponding to some kind of change request, this will be set to the human name of the author, if supported; else unset.

CHANGE_AUTHOR_EMAIL
For a multibranch project corresponding to some kind of change request, this will be set to the email address of the author, if supported; else unset.

CHANGE_TARGET
For a multibranch project corresponding to some kind of change request, this will be set to the target or base branch to which the change could be merged, if supported; else unset.

CHANGE_BRANCH
For a multibranch project corresponding to some kind of change request, this will be set to the name of the actual head on the source control system which may or may not be different from `BRANCH_NAME`. For example in GitHub or Bitbucket this would have the name of the origin branch whereas `BRANCH_NAME` would be something like `PR-24`.

CHANGE_FORK
For a multibranch project corresponding to some kind of change request, this will be set to the name of the forked repo if the change originates from one; else unset.

TAG_NAME
For a multibranch project corresponding to some kind of tag, this will be set to the name of the tag being built, if supported; else unset.

TAG_TIMESTAMP
For a multibranch project corresponding to some kind of tag, this will be set to a timestamp of the tag in milliseconds since Unix epoch, if supported; else unset.

TAG_UNIXTIME
For a multibranch project corresponding to some kind of tag, this will be set to a timestamp of the tag in seconds since Unix epoch, if supported; else unset.

TAG_DATE
For a multibranch project corresponding to some kind of tag, this will be set to a timestamp in the format as defined by java.util.Date#toString() (e.g., Wed Jan 1 00:00:00 UTC 2020), if supported; else unset.

JOB_DISPLAY_URL
URL that will redirect to a Job in a preferred user interface

RUN_ARTIFACTS_DISPLAY_URL
URL that will redirect to Artifacts of a Build in a preferred user interface

RUN_CHANGES_DISPLAY_URL
URL that will redirect to Changelog of a Build in a preferred user interface

RUN_TESTS_DISPLAY_URL
URL that will redirect to Test Results of a Build in a preferred user interface

CI
Statically set to the string "true" to indicate a "continuous integration" execution environment.

BUILD_NUMBER
The current build number, such as "153".

BUILD_ID
The current build ID, identical to BUILD_NUMBER for builds created in 1.597+, but a YYYY-MM-DD_hh-mm-ss timestamp for older builds.

BUILD_DISPLAY_NAME
The display name of the current build, which is something like "#153" by default.

JOB_NAME
Name of the project of this build, such as "foo" or "foo/bar".

JOB_BASE_NAME
Short Name of the project of this build stripping off folder paths, such as "foo" for "bar/foo".

BUILD_TAG
String of "jenkins-*${JOB_NAME}-${BUILD_NUMBER}*". All forward slashes ("/") in the JOB_NAME are replaced with dashes ("-"). Convenient to put into a resource file, a jar file, etc for easier identification.

EXECUTOR_NUMBER
The unique number that identifies the current executor (among executors of the same machine) that's carrying out this build. This is the number you see in the "build executor status", except that the number starts from 0, not 1.

NODE_NAME
Name of the agent if the build is on an agent, or "built-in" if run on the built-in node (or "master" until Jenkins 2.306).

NODE_LABELS
Whitespace-separated list of labels that the node is assigned.

WORKSPACE
The absolute path of the directory assigned to the build as a workspace.

WORKSPACE_TMP
A temporary directory near the workspace that will not be browsable and will not interfere with SCM checkouts. May not initially exist, so be sure to create the directory as needed (e.g., `mkdir -p` on Linux). Not defined when the regular workspace is a drive root.

JENKINS_HOME
The absolute path of the directory assigned on the controller file system for Jenkins to store data.

JENKINS_URL
Full URL of Jenkins, like `http://server:port/jenkins/` (note: only available if *Jenkins URL* set in system configuration).

BUILD_URL
Full URL of this build, like `http://server:port/jenkins/job/foo/15/` (*Jenkins URL* must be set).

JOB_URL
Full URL of this job, like `http://server:port/jenkins/job/foo/` (*Jenkins URL* must be set).

Flow Control If Else Try Catch

if..else

```
pipeline {
    agent any
    environment{
        fname = "KAMAL"
    }
    stages {
        stage('one') {
            steps{
                script{
                    if (env.fname == 'KAMAL') {
                        echo 'HELLO KAMAL'
                    }
                    else{
                        echo 'I do not know you!'
                    }
                }
            }

        }
    }
}
```

try...catch

```
pipeline {
    agent any
    environment{
        fname = "KAMAL"
    }
    stages {
        stage('one') {
            steps{
                script{
                    try{
                        sh 'hello'
                    }
                    catch{
                        echo 'Error occured!!!!'
                    }
                }
            }

        }
    }
}
```

Parameters

Parameters

A very rich feature of jenkins is that jobs can be parameterized. In pipelines, we have a directive named as parameters.The values for these user-specified parameters are made available to Pipeline steps via the params object.

```
pipeline {
    agent any
    parameters {
        string(name: 'NAME', defaultValue: 'Uthred', description: 'Enter your name')
        choice(name: 'CITY', choices: ['Bebbanburg', 'Mercia', 'East Anglia'], description: 'Choose your city')
    }
    stages {
        stage('Example') {
            steps {
                echo "Hello ${params.NAME} of ${params.CITY}"
            }
        }
    }
}
```

Parameters

Passing Parameter from one Job to Another