PIPLINE PORJECT :

<https://www.jenkins.io/doc/book/pipeline/syntax/>

In freestyle project all the configuration we do it in UI like General configuration like parameters , Source Code Management, Build Triggers, Build Environment,build steps , Post-build Actions.

In pipeline project apart from few setting all other configuration we will have to write it in pipeline file.

Ex-

|  |
| --- |
| pipeline {  agent any  parameters  {  string(name: 'myParameter', defaultValue: 'myVal', description: 'Enter Parameter value?')  }  stages {  stage('Build') {  steps {  echo 'Building..'  echo "Running ${env.BUILD\_ID} on ${env.JENKINS\_URL}"  }  }  stage('Test') {  steps {  echo 'Testing..'  echo "${params.myParameter} is value retrieved!"  }  }  stage('Deploy') {  steps {  echo 'Deploying....'  }  }  }  } |

>A Jenkinsfile can be written using two types of syntax - Declarative and Scripted.

|  |  |
| --- | --- |
| Jenkinsfile (Declarative Pipeline) | Jenkinsfile (Scripted Pipeline) |
| *pipeline {*  *agent any*  *options {*  *skipStagesAfterUnstable()*  *}*  *stages {*  *stage('Build') {*  *steps {*  *sh 'make'*  *}*  *}*  *stage('Test'){*  *steps {*  *sh 'make check'*  *junit 'reports/\*\*/\*.xml'*  *}*  *}*  *stage('Deploy') {*  *steps {*  *sh 'make publish'*  *}*  *}*  *}*  *}* | *node {*  *stage('Build') {*  *sh 'make'*  *}*  *stage('Test') {*  *sh 'make check'*  *junit 'reports/\*\*/\*.xml'*  *}*  *if (currentBuild.currentResult == 'SUCCESS') {*  *stage('Deploy') {*  *sh 'make publish'*  *}*  *}*  *}* |

**Jenkins Pipeline: running external programs with sh or bat**

From within a Jenkins pipeline you can any external program. If your pipeline will run on Unix/Linux you need to use the sh command. If your pipeline will run on MS Windows you'll need to use the bat command.

**Groovy(**the separate, non-bash interpreter that parses Jenkinsfiles) language is used to write the pipeline syntax which is java based.

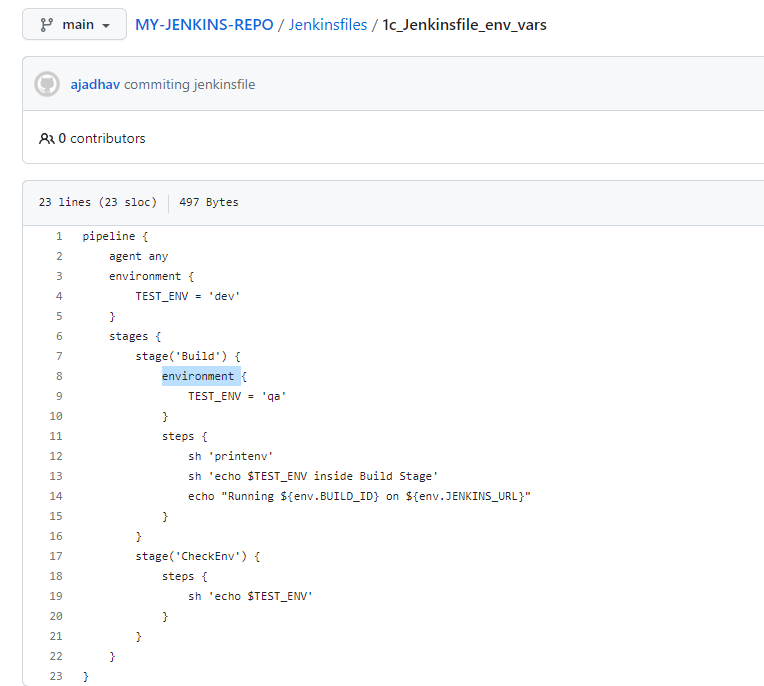
In Groovy, but not in bash, strings must use triple-quotes to span multiple lines.

If we do not want this then for every linux command we need to sh



>Enviromental variable declaration at pipeline level or at stage level .

environment{} directive



Our declared env variables can be accessed with **$** but if we want to access the env variables provided by jenkins we need to use **$(env.VARIABLE\_NAME)**

**>Timeout and Retry in JenkinsFile**

retry() and timeout() are the available functions that can be used for this functionality.



**>TO ACCESS** J**ENKINS CREDENTIALS IN JENKINS FILE**

[**https://www.jenkins.io/doc/book/using/using-credentials/**](https://www.jenkins.io/doc/book/using/using-credentials/)

-credentials() function is available.

-credentials can be used by Pipeline projects to interact with these 3rd party applications

Jenkins can store the following types of credentials:

* **Secret text** - a token such as an API token (e.g. a GitHub personal access token),
* **Username and password** - which could be handled as separate components or as a colon separated string in the format username:password (read more about this in [Handling credentials](https://www.jenkins.io/doc/book/pipeline/jenkinsfile#handling-credentials)),
* **Secret file** - which is essentially secret content in a file,
* **SSH Username with private key** - an [SSH public/private key pair](http://www.snailbook.com/protocols.html),
* **Certificate** - a [PKCS#12 certificate file](https://tools.ietf.org/html/rfc7292) and optional password, or
* **Docker Host Certificate Authentication** credentials.



-As of now we have Jenkins server on EC2 instance on one AWS account.

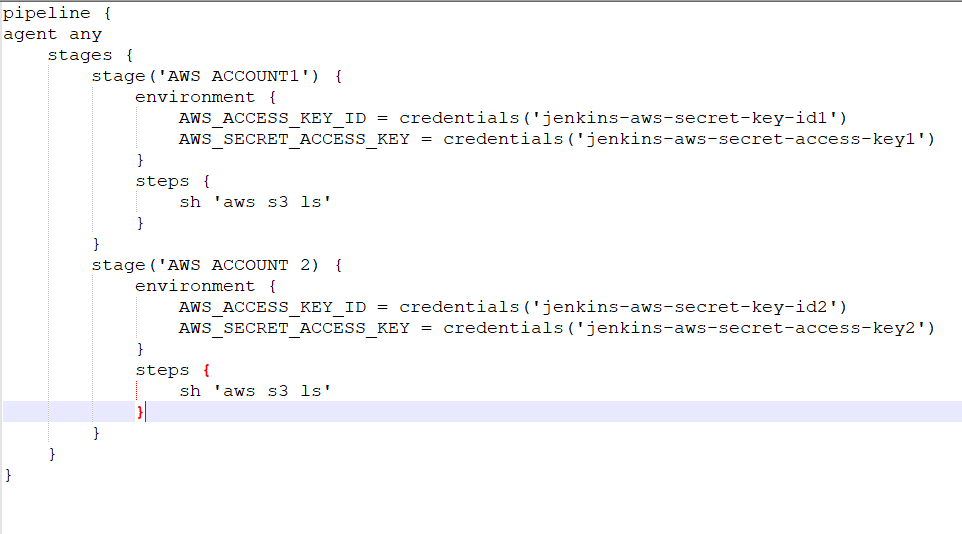
-We can have a single Jenkins job doing deployment in multiple machines in multiple account.

- to do so we need access key and access id of each account.

-These details can be stores in Jenkins credentials and fetched from there in jenkinsfile.

-The testing below we are doing on the EC2 machine which has IAM role attached and hence AWS commands will work in Jenkins file if we write it without AWS access key and access id as well.

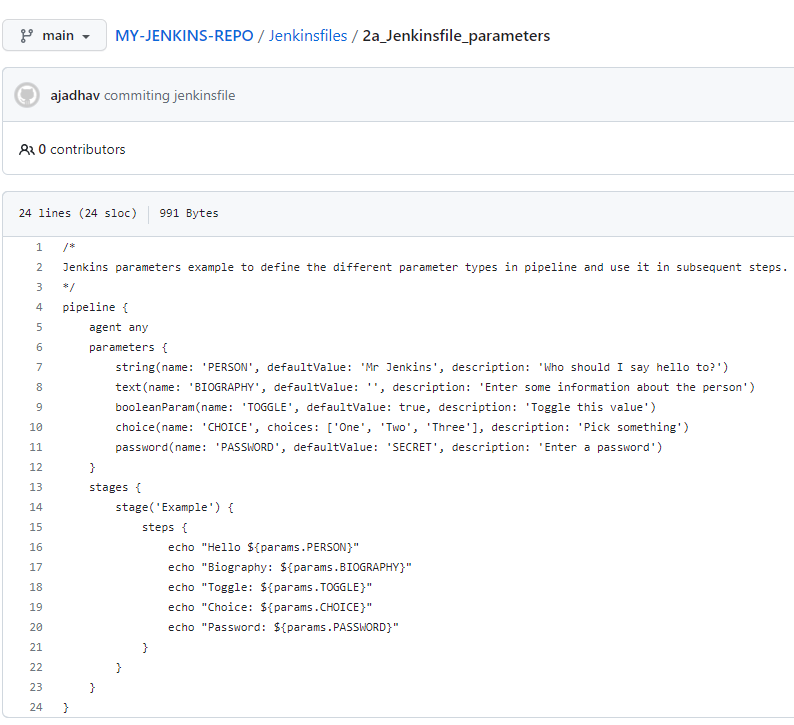




>**parameters{} directive**

We can define diff parameters in pipeline and can be used in different stages with **$(params.PARAMATER\_NAME)** syntax

-first time when we execute the pipeline job has to checkout the code and then it will know that this job needs parameter and In next run “Build Now” button will be changed to “Build with Parameters” and then we can pass the parameter values.



**>post section in pipeline**

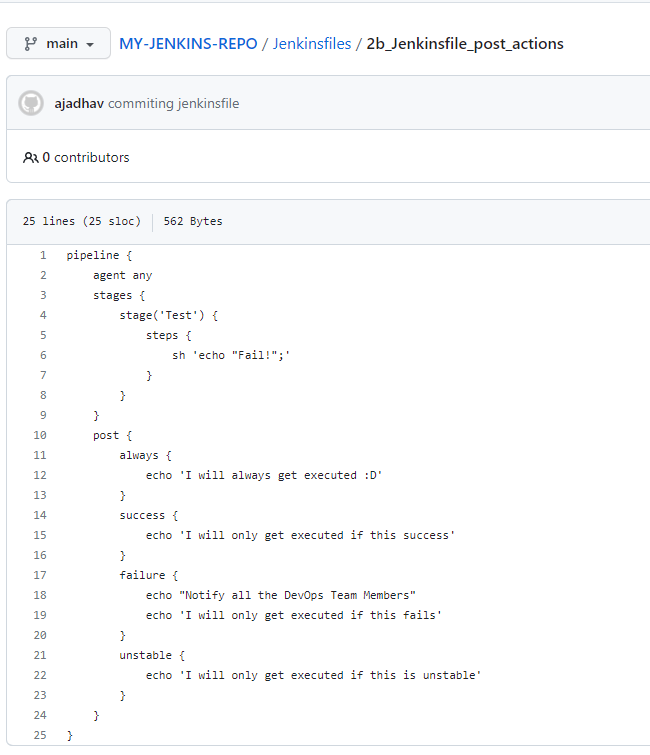
-The post section defines one or more additional steps that are run upon the completion of a Pipeline’s or stage’s run (depending on the location of the post section within the Pipeline).

-post can support any of the following post-condition blocks

always, changed, fixed, regression, aborted, failure, success, unstable, unsuccessful, and cleanup.

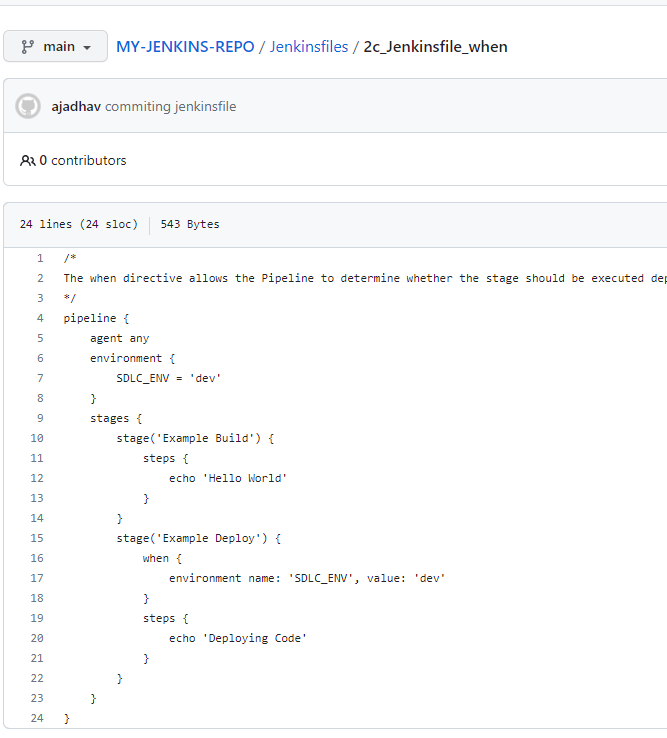
-Conventionally, the post section should be placed at the end of the Pipeline.

-Post-condition blocks contain steps the same as the steps section.



**>when directive**

The when directive allows the Pipeline to determine whether the stage should be executed depending on the given condition. The when directive must contain at least one condition



**>Using Docker with Pipeline**

[**https://www.jenkins.io/doc/book/pipeline/docker/**](https://www.jenkins.io/doc/book/pipeline/docker/)

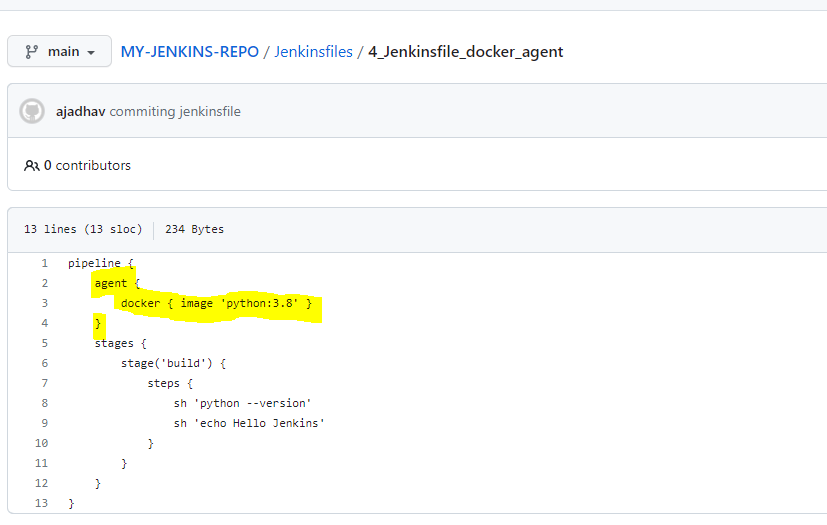
-Many organizations use Docker to unify their build and test environments across machines, and to provide an efficient mechanism for deploying applications.

-Starting with Pipeline versions 2.5 and higher, Pipeline has built-in support for interacting with Docker from within a Jenkinsfile.

- Pipeline is designed to easily use Docker images as the execution environment for a single Stage or the entire Pipeline.

-Meaning that a user can define the tools required for their Pipeline, without having to manually configure agents.

- Practically any tool which can be packaged in a Docker container,can be used with ease by making only minor edits to a Jenkinsfile.



-When the Pipeline executes, Jenkins will automatically start the specified container and execute the defined steps within it.

-Meaning all the steps mentioned in jenkinsfile will be executed in docker container , In above case we are using docker container with python image hence python commands can be executed and will be executed inside the container.

**Specifying a Docker Label**

By default, Pipeline assumes that any configured agent is capable of running Docker-based Pipelines. For Jenkins environments which have macOS, Windows, or other agents, which are unable to run the Docker daemon, this default setting may be problematic. Pipeline provides a global option in the Manage Jenkins page, and on the Folder level, for specifying which agents (by Label) to use for running Docker-based Pipelines.

As of now pipeline is failing with below error need to understand if we need to install docker tool to make it work.

We installed DOCKER, DOCKER PIPELINE plugins in Jenkins.

