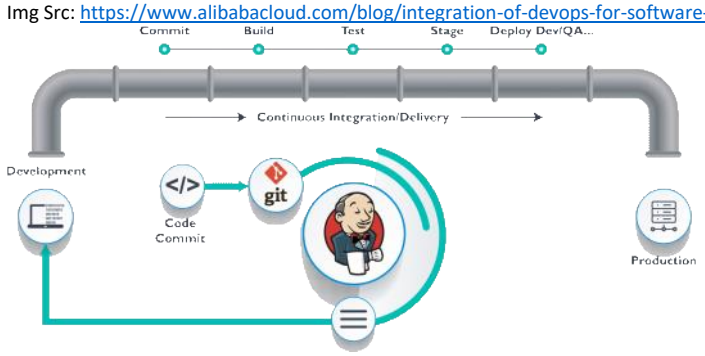
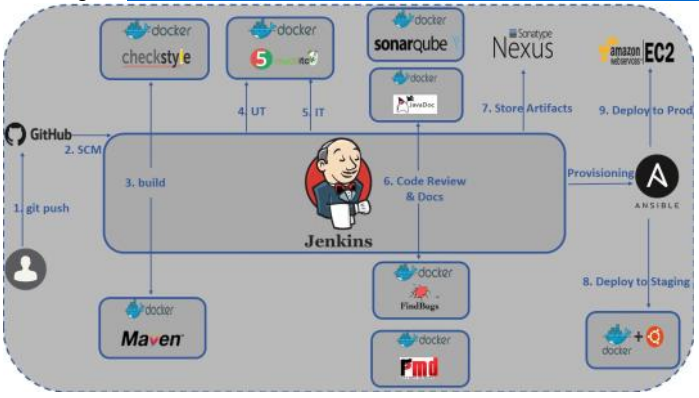
Jenkins-Basics

- **Introduction**
  - Documentation : Jenkins User Documentation
  - Jenkin - Open source automation server.
  - Large plug-in support
  - Simple Jenkins workflow. **Important**: This is one of the many implementations of Jenkins.
    Img Src: https://www.alibabacloud.com/blog/integration-of-devops-for-software-development-%26-delivery_596392



    Alt link (advanced implementation example with scripting) : Pipeline as a Code using Jenkins 2 | by Mayank Patel | Medium
  - Another view from a tool interlock perspective
    - Img src: https://medium.com/@ahmed24khaled/building-e2e-devops-pipeline-a0f7804390f7



- **Terminology**

Workspace	- Working directory for Jenkins.
Master Node / Agent	- Primary Server that hosts Jenkins application
Slave Node / Agent	- Target Server on which the actions as part of the automation that need to be done. • For e.g. These could be the Dev / Prod servers. • Will require agents running and connected to the master. • Ports need to be opened. • If no slave is provided, the master's workspace is used for the deployment. • For Plug-ins that have direct reference and connectivity to target hosts, Slave nodes might not be required (for e.g. Ansible or Terraform), but for a traditional web or application deployments the slave nodes and configuration is needed.
Jobs	- Are the actions that need to be performed within Jenkins
Build	- The trigger of jobs or a pipeline is referenced in Jenkins as build (doesn't necessarily have to be the build of code)
Pipeline	Series of steps in Jenkins. Can have parameters as well as conditional executions. Further Read - Pipeline Syntax (jenkins.io) & https://www.jenkins.io/doc/book/pipeline/#pipeline-concepts
Git hook	- Feature at Jenkins end, that polls git at regular intervals for changes and triggers the pipeline/ job accordingly.
Web Hook	- Feature in Github, that sends push notifications to a designated URL , like a jenkins master url, (with secret if required).
Jenkinsfile	- a text file that can contain instructions for the pipeline.

Parameters	- Inputs that are provided by the person triggering the job (either text , drop down etc), that will be used in Pipeline for customized execution.
	- Example:
	• Declaration:
	parameters {
	choice(name: 'WORKSPACE', choices: ['DEV', 'UAT', 'PRD'] , description:'workspace environment')
	choice(name: 'SERVERTYPE', choices: ['websrvr', 'dbsrvr','natsrvr'], description: 'server group to perform action on')
	}
	• Used in code as
	○ srvrgrp = "${params.SERVERTYPE}_${params.WORKSPACE}"

- **Typical execution sequence** will be
  - Job that references a git repository
  - Jenkins copies the git repo.
  - Executes the steps defined.
    - Build Steps could be
      - Another Job
      - Another Pipeline

        - A Script (or batch file)
    - Post build tasks
        - Can be script or trigger another job.
    - Triggers notification if required.
    - **Note**: As part of the CI/CD setup, Jenkins can auto trigger a pipeline when the git repository is updated (webhooks) or there is an action calling the pipeline thru the Jenkins API
- **Build Steps** (or Pipeline) to be executed can be done in two ways.
    - Using Buildstep plugins (like conditionalBuildStep plugin)
    - Defining the pipeline either in UI or in Jenkinsfile. (Same syntax)
- **Types**:
    - Pipeline there are two types (Pipeline (jenkins.io)):
        - Declarative Pipeline [starts with pipeline {]
        - Scripted Pipeline [Starts with node { stages { stage |
- **Jenkinsfile** :
    - File definition of how the build steps need to be performed. Contains the parameters, build stages & steps in each build stage
    - **Note:** Jenkins has a workspace directory(its working directory) , where the git repositories are pulled and any required files are copied , so care needs to be taken to check the file / folder references as such. Best to get a cleanup job frequently to clean up these folders to avoid space issues.
    - **Jenkins file format:**

```
Global Variable definitions #Optional
pipeline {
  parameters {   }    # Optional
  environment {   } # Optional
  stages {
    stage('Stage Name '){
      steps {  }
        }
      }
    stage('Stage Name '){
      steps {  }
        }
      }
  }
```

    - **Variables**:
        - Variables are defined using **def var1 = <default value>.** Variables are referenced as $ { variable name }.
        - Environment variables are defined inside **environment {}**
            - **Example:**

```
environment {
    AWS_ACCESS_KEY_ID = credentials('aws_access_key')
    AWS_SECRET_ACCESS_KEY = credentials('aws_secret_access_key')
    PATH = "$TF_HOME:$PATH"
  }
```

        - Secrets can be updated in Jenkins in Global configuration. They will be called using credentials function or using the withcredentials function.
        - Parameter variables are referenced as $ { params.<variable_name>}
    - **Scripts** :
        - To enhance the execution, small pieces of code / program calls can be included in the Jenkins file.
        - However, keep in mind that the commands to call are limited by the underlying OS on which the Jenkins server is running on.
            - For ex Commands / Batch files in windows are called using the "bat" command while in Linux it is the "sh".
        - Non Jenkins commands would be going under the script block.
        - Scripts are encoded within {}
        - Exception Handling:
            - Try - catch - finally can also be implemented within the script for error handling.
            - Usage

```
try {
        Steps to execute
} catch(err){
        Steps to execute if error encountered
} finally {
        Steps to execute irrespective of error status
   }
```

        - Common Commands:

currentBuild variable	represents meta data for current build. □ currentBuild.result = 'UNSTABLE' , updates the build stage status as failed based on conditions.
Echo	displays content.
Return	Exits the block
If (condition) {}	Executes steps within the block based on the condition
Input	Accept inputs from user during pipeline execution
dir	Provides working directory for the steps
When	Condition clause for running the steps. Written as the first line in the stage before steps Usage Examples: when { equals expected:'BUILD', actual:"${params.WORKTYPE}"} Or when { expression { $params.WORKTYPE} = "BUILD"}
withCredentials	References the secrets stored in Jenkins (updated manually in Jenkins Config , through UI)
Expression	Represents a string processing (like equal to etc)
Stash	Keep the build in staging (usually done if you want to apply the build post approval or after a checking some conditions).

	released using the "unstash <stash name>"

- Putting it all together (simple example):
  - https://github.com/inugav/TerraformProj_AWS/blob/master/NetworkSetup/Jenkinsfile