**CI/CD with Jenkins using Pipelines and Docker**

* **Introduction of jenkins:**
* **What is Jenkins:**
* Jenkins is an open-source continuous integration and continuous delivery tool written in java.
* It's an automation server used to build and deliver software projects.
* Jenkins was forked from another project called Hudson, after a dispute with oracle.
* A major benefit of using Jenkins is that it has a lot of plugins available.
* There is easier to use CI software available, but Jenkins is open source, free and still very popular.
* **Continuous integration:**
* Continuous integration is the practice, in software engineering, of merging all developer working copies to a shared mainline several times a day.
* **Continuous delivery:**
* Continuous delivery is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time.
* **Benefits:**
* Jenkins provide a feedback loop back to the developer to fix build errors.
* Jenkins can publish every build of your software.
* **CI/CD within the SDLC:**
* **Jenkins installation:**
* Open digital ocean
* Click create droplet
* Select ubuntu
* Click on $20/month
* Select data center region
* Add your SSH keys
* Choose a hostname
* click create
* Open terminal
* Ssh root@<IP ADDRESS>
* yes
* Enter password
* Install jenkins
* Wget <url>
* Bash install\_jenkins.sh
* Copy the ip address:8080
* Jenkins page will be opened
* Click install suggested plugins
* Create first admin user
* Click finish
* Click start using jenkins
* **Introduction of docker:**
* What is docker
* Docker is the most popular container software
* Docker engine: it is docker runtime
* Software to make run docker images
* Docker hub: online service to store and fetch docker images
* Also allow you to build docker images online
* Docker provides isolation-you ship a binary with all the dependencies.
* Docker makes development teams able to ship faster
* Docker uses Linux containers for operating system level isolation
* **Why use docker for Jenkins:**
* Everyone will have the same container image of jenkins.
* Regardless of your operating system
* You can run docker on an ubuntu on DigitalOcean,but also on a Mac, on windows, etc.
* You can easily upgrade Jenkins by pulling the latest container image.
* you can modify the Jenkins container image by creating your own docker file.
* Or modify the docker file I have for Jenkins in my GitHub repository.
* **What is Node.js**
* Node.js is a JavaScript runtime environment which is:
* Open source
* Cross platform
* Used to execute JavaScript code server side instead of client side
* Has an event-driven architecture capable of asynchronous I/O

This makes node.js capable of responding very fast to requests.

* **Why node.js**
* Easy to understand
* It doesn't need to be compiled-so building the app doesn’t take a lot of memory
* **Build and deploy first application (using Node.JS):**
* Open jenkins
* Manage jenkins
* Manage plugins
* Available
* Search bar-node js plugin
* Install without restart
* Create new job:
* Click new item
* Enter job name
* Click freestyle
* Ok
* Select git
* Enter repository <URL>
* If node.js not installed –go to manage jenkins-global tool configuration-select node js-install automatically-apply-save.
* Under build environment –select provide node and npm/bin folder to path
* Under build
* Click execute shell-npm install
* Apply
* Save
* Build now.
* **Building node.js application with Docker:**
* Manage jenkins
* Manage plugins
* Available
* In search bar-docker-cloudbees docker build and publish plugin
* Install without restart
* Ok
* Go to job
* Configure
* Under build-select docker build and publish-give docker hub repository path
* Open docker hub account
* Create repository
* Add docker hub credentials under registry credentials
* Docker installation-select docker
* Save
* Build now
* **Infrastructure as code and automation:**
* Jenkins allows you to use a web UI to input all the build parameters.
* This leads:
* No proper audit trail
* No easy history of changes
* Segregation between Jenkins admins and developers

Users mean developers will have to contact a Jenkins administrator to make changes

Long lead times for changes

* difficult to backup and restore.
* **Jenkins for DSL**:
* The Jenkins job DSL is a plugin that allows you to define jobs in a programmatic form with minimal effort.
* DSL stands for Domain Specific Language

You can describe jobs using a Groovy based Language

Groovy as a scripting language for java platform

* It's similar to java but simpler, because its much more dynamic
* This plugin is easy to manage jobs.
* **Jenkins job DSL with node.js application:**
* Note: install DSL plugin (important)
* Manage jenkins
* Manage plugins
* Available
* Job DSL plugin
* Install without restart
* create new item
* enter anme
* free style project
* Ok
* select git
* Enter repository <URL>
* Under bulid –select process job DSLs
* Look on filesystem-repository url after job <url>
* Save
* Build now
* Build failure(script not yet approved for use)
* manage jenkins
* Inprocess script approval
* Click approve
* Build now.
* **Jenkins job DSL with docker build and publish:**
* Open jenkins
* Manage jenkins
* Git repository <URL>
* Under build click execute shell
* Enter repository name
* Add docker hub credentials
* Apply
* Save
* Build now(build fail)
* Manage Jenkins
* Inprocess script approval
* Click approve
* Build now.
* **Jenkins pipelines:**
* Jenkins pipelines allow you to write the Jenkins build steps in code

1. Build steps allow you to write build(compile), test, deploy in code.
2. Code means you can put this code in version control

* **Jenkins pipelines vs job DSL:**
* They both have the capability to write all your CI/CD in code
* The difference is in implementation in Jenkins
* Jenkins job DSLs creates new jobs based on the code you write
* The Jenkins pipelines is a job type, you can create a Jenkins pipeline job that will handle the build/test/deployment of one project.
* **Jenkins pipelines with NodeJS and Docker:**
* click new item
* Enter job name
* Select pipeline
* Ok
* Select pipeline script from SCM
* Select git
* Enter repository <URL>
* Enter script path
* Apply
* Save
* Build now
* Too see the output open console output.
* **Build test and run everything in docker container:**
* Create new jenkins file in github
* Manage jenkins
* Global tool configuration
* Click node.js installation
* Click new item
* Enter job name
* Pipeline
* Ok
* Select pipeline script from SCM
* Select git
* Enter repository <URL>
* Enter script path
* Apply
* Save
* Build now
* Too see the output open console output.
* **Email integration:**
* The goal is to alert the developer of a broken build as soon as possible.
* **Email integration using Jenkins pipelines:**
* manage Jenkins
* Manage plugins
* Available
* Email extension plugin (this plugin is a replacement for Jenkins email publisher)
* Insall without restart
* Go back to Jenkins home page
* **Configure plugin:** (Mailtrap)
* Manage jenkins
* Configure system
* Extend email notification
* Enter SMTP server
* Advance
* Use SMTP authentication
* Enter username and password.
* New item
* Enter job name
* Click pipeline
* Ok
* Under pipeline
* Select pipeline script from SCM
* Select git
* Copy and paste repository <URL>
* Script path
* Poll scm (H/5 \*\*\*\*)
* Apply
* Save
* Build now(test failed)

open mailtrap

Click jenkins inbox

Click build failure.

* **Slack integration:**
* Slack is a chat and collaboration tool, comparable with Atlassian’s HipChat.
* It's a new and popular communication tool, that is much better than using skype or older enterprise chat tools for collaboration.
* Slack is better than its competitors, because it allows you to integrate all your tools within slack, to avoid switching between apps.
* It can give you a Realtime “war room” to collaborate on problems.
* This is also called “ChatOps”.
* chatOps is a collaboration mode that connects people, tools, processes, and automation in a transparent workflow.
* Slack integration (Jenkins with slack)
* Open slack account
* Create team
* Open jenkins
* Manage jenkins
* Manage plugins
* Available
* Slack
* Download –slack notification plugin
* Install without restart
* Go to slack
* Click on custom apps and integration
* Sign in
* Webhook
* Add configuration
* Post to channel-alerts
* Add incoming webhook integration
* Copy link under example
* Go to Jenkins
* Manage Jenkins
* Configure system
* Global slack notifier settings
* Click add
* Add credentials
* Global credentials
* Secret test
* id,description-slack-alerts
* copy webhook url –paste in secret(after service we need to copy)
* Add
* Integration token credentials id-slack alerts
* Go to webhook-copy URL before service
* Paste it in the - base url
* Team subdomain-after http link in browser
* Channel-#alerts
* Test connection(success)
* Save
* Let's add new job in Jenkins –new item
* Enter job name
* Pipeline
* Ok.
* Pipeline
* Pipeline script from SCM
* Select git
* Copy and paste git URL
* Apply
* Save
* Build now(build failure)
* Open webhook and solve.
* **GitHub integration with a Gradle and maven project:**
* Go to jenkins
* Manage jenkins
* Managed plugins
* install giuhub branch source plugin
* Install without restart
* Go to home page
* New item
* Enter job name
* Select GitHub organization
* Ok
* Add jenkins
* Enter username and password(note generate github token)
* Add(github repo scan everyday)
* Save.
* **Bitbucket integration:**
* Go to jenkins
* Manage jenkins
* Managed plugins
* install bitbucket branch source plugin
* Install without restart
* Go to home page
* New item
* Enter job name
* Select bitbucket team project
* Ok
* Add credentials(go to bitbucket create app password)
* Paste in jenkins
* Add.
* **Jfrog integration:**
* In the previous demos i have always submitted the docker image to the docker registry (docker hub)
* this resulting image is in jenkins called the artifact
* Its the resulting binary from a build
* It can be a docker image,or a .jar file or a .tgz or zip file really anything
* these artigfacts you want to store somewhere
* Jfrog artifactory is a product that can store for you the artifact resulting from a build.
* Jfrog integration is done using a jfrog plugin
* The jfrog plugin allows you to add extra steps to your jenkinsfile.
* Practical
* Download jfrog free source application
* Go to jenkins
* Manage jenkins
* Managed plugins
* install artifactory plugin
* Install without restart
* Go to home page
* Go to jfrog account
* Enter username and password
* Create gradle repository
* Go to group and create a group
* Save
* Go to users
* Create user
* Save
* Add permissions
* Save
* **Configure**
* Go to manage jenkins
* Configure system
* Add jenkins credentials
* Username and password
* Use API key as password
* Enter server ID
* Url
* Save
* Go to manage jenkins-global tool configuration-add gradle name:gradle-install automatically-save
* New item
* Enter job name
* Pipeline
* Ok
* Pipeline
* Pipeline script from SCM
* Copy and paste repository URL
* Enter script path
* Save
* Build now.
* **API integration**:
* Sometimes you want to integrate an API, but there is no plugin available.
* Another solution is to use functionality in the jenkins pipelines to do http requests.
* Http request plugin
* Practical:
* Go to jenkins
* Manage jenkins
* Managed plugins
* install http request plugin
* Install without restart
* Go to home page
* Go to bitbucket
* Under access management
* Click Oauth
* Add consumer
* Give callback url
* Under pll request –click read
* Save
* Click on consumer name
* Take url, key and secret key
* Go to credentials
* Add
* Paste here username and password(above i copied)
* Go to jenkins
* New item
* Enter job name
* Pipeline
* Ok
* Pipeline
* Pipeline script from SCM
* Select git
* Copy and paste git URL
* Apply
* Save
* Build now(failed)
* manage jenkins
* Inprocess script approval
* Click approve
* Build now (success)
* Go to jenkins
* Configure
* Click pipeline syntax
* Snippet generator
* Perform an HTTP request and return a response object
* Url(google.com)
* Yes
* Generate pipeline script
* Advanced
* Content form-application type
* Http mode-post
* Save.
* **SonarQube integration:**
* SonarQube continuously inspects your software project on code quality.
* It can report on:
* Bugs(code issues)
* Vulnerability(security issues)
* Code smells(maintainability related issues)
* Technical debt(estimated time required to fix)
* Code coverage(test coverage of code)
* Sonarqube integrate with jenkins:
* Go to jenkins
* Manage jenkins
* Managed plugins
* install sonarqube scanner for jenkins plugin
* Install without restart.
* Go to sonarqube
* Administrator
* Security
* Enter token name
* Generate
* Go to jenkins
* Manage jenkins
* Global tool configuration
* Sonarqube for scanner
* Click install automatically
* Save
* Credentials
* Add credentials
* Ok
* New item
* Enter job name
* Pipeline
* Ok
* Pipeline
* Pipeline script from SCM
* Select git
* Copy and paste git URL
* Apply
* Save
* Build now.
* **Jenkins slave:**
* **Benefits**
* Reduced cost
* Slaves are easily replaceable
* Important points about slaves:
* Don't manually install tools on the slaves
* Use plugins to provide tools
* **Jenkins slave using SSH:**
* Manage jenkins
* Manage nodes
* Click new node-enter name –select permanent agent-ok-executors(2)-remote root directory give-description(this is new jenkins node)-give label-launch agent slave via ssh-go to digital ocean-create droplet(like instance)-select ubuntu-select datacenter region-user data-copy the script if it is available in github repo-ssh-keygen –t rsa(we will get ssh key)cat mykey.pub-copy key and paste-new ssh key-paste key-copy ip address under host-add creditals jenkins-select ssh-private key paste here-username-mykey-mykey-ok—advanced-port(2222)-save-launch agent.
* **Jenkins slave using jnlp:(java network launch protocol)**
* Manage jenkins
* Manage nodes
* Click new node-enter name –select permanent agent-ok-executors(2)-remote root directory enter-label-save
* Go to ubuntu server-install java-java –version(to check version)-copy node link and paste in ubuntu server-enter-go to jenkins-refersh-it will be connected.
* **Blue ocean**:
* Blue ocean is a new frontend for jenkins

Built from the ground up for jenkins pipeline

Provided as plugin

Still compatible with freestyle projects

Should eventually replace the normal jenkins UI over time.

* **New features:**
* Sophisticated visualizations of the pipeline
* A pipeline editor
* Personalization
* more precision to quickly find what's wrong during an intervention
* Native integration for branch and pull requests
* **Blue ocean installation:**
* Go to jenkins
* Manage jenkins
* Manage plugins
* Available
* Install blue ocean plugin
* Install without restart
* Ok
* Open blue ocean on the above scroll bar
* **SSH agent-how to use ssh agent:**
* Click add credentials
* Download ssh agent plugin
* Create new job
* Add git repository url
* Save
* Build now
* **Security best practices:**
* Keep your Jenkins up to date
* Use firewall rules
* Always upgrade to the latest version
* Use the long-term support edition if you want to be on a stable version
* Also read the changelog
* Always up to date plugins
* Configure authentication and authorization
* Don't use weak passwords and logins
* **Authentication:**
* The process or action of verifying the identity of a user and process.
* Basically, verifying the credentials of the user: the username and password.
* **Authorization:**

**Meaning of authorization:**

Authorization in Jenkins refers to the process of controlling access to Jenkins resources, such as jobs, views, and nodes, based on user permissions.

* **Few options:**
* Anyone can do anything (not recommended)
* Anyone who is logged in can do anything
* Matrix based authorization (just a big table to give users access)
* Role strategy plugin.
* Practical:
* Go to manage jenkins
* Configure global security
* Add options(which we want)
* **Authentication:**
* **Onelogin.com**
* OneLogin handle the authentication process.
* It also provide multi factor authentication
* create first onelogin account
* add apps
* SAML test connector
* display name(jenkins)-if u need jenkins logo you can upload
* Save
* Go to configuration
* Audience (http://ip adress:8080/securityrealm/finishlogin)
* Recepient (http://ip adress:8080/securityrealm/finishlogin)
* Acs URL validator(^http:\/\/ipaddress:8080\/)
* Acs consumer URL (http://ip adress:8080/securityrealm/finishlogin)
* Save
* More actions
* SAML metadata
* Open text edit-copy text
* Open jenkins
* Manage jenkins
* Manage plugins
* Install SAML plugin
* Install without restart
* Manage jenkins
* Global tool configuration
* Select SAML
* Paste copied text here
* save
* Login to onelogin.