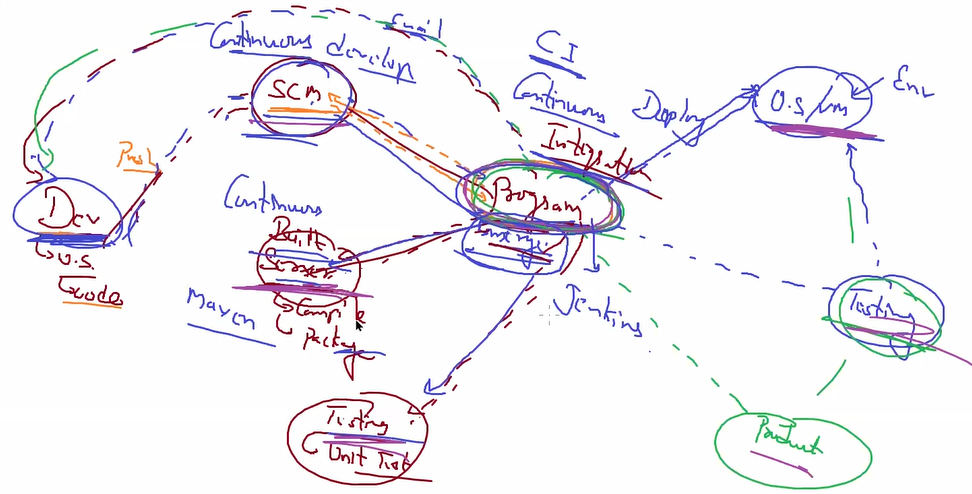
**Session 01 – Intro**

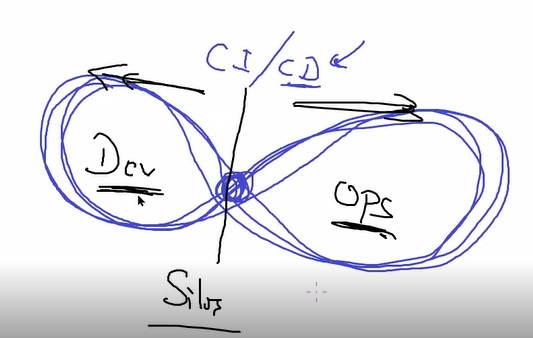
* In our manual process dev will push the code in different branch.
* If code is working fine then we will merge it, solve merge conflicts and then code go to compile/ Build and we make a package of it and after this we share to production world.
  + While deploying we may face many challenges, and may found error, we have to give this code back to the developer and he again work on it.
  + Here we find 2 issues.

1. Developer may forget what he wants to do in that part because of 4-5 days are gone.
2. Our final product comes very late in the production environment.

* Here we don’t want to remove any important step.
  + We want to remove that part which makes this process slow, which can be got by automation.
  + We will make this entire N-to-N pipeline automated.
* This program will fetch code from GitHub, if some error give feedback to dev and again fetch.
  + Then it will build and create package, after this it will test.
  + If all things are fine deploying to the OS, again test & give feedback to OS (team), then release to the production environment.
  + This program is integrating everything.
  + It is known as Jenkins.
    - Jenkins is a **Continuous Integration** tool.



* + In manual world dev team don’t have idea about operation team and vice versa.



* + - **Infinity cycle**.
    - But Jenkins will do this entire connectivity, which make tasks easy.

**Session 2 – Installation//plugin//demo**

* Jenkins is build on top of **JAVA**.
  + JAVA is supported by almost all OS.
  + We only require jdk in a system.
* In our setup we are going to use redhat VM built on top of oracle virtualbox.
  + Download software from drive & copy in the rhel.
    - Rpm -ivh jdk-….rpm
    - Rpm -ivh jen…
  + It will generate random password for Jenkins dashboard.
    - Default username is admin.
    - Copy it from /var/lib/Jenkins/secrets/initialAdminPassword
  + You can install plugin at starting (not recommanded).
  + Change admin password.
    - Click on admin 🡪 manage users 🡪 admin.
* Jenkins is a simple tool.
  + - It can’t do anything much but you can use plugins.
    - Suppose Jenkins want to integrate to GitHub, so for this we can download GitHub **plugin**.
    - Almost for everything we have plugin available in the Jenkins.
      * Plugin mean we are giving some extra knowledge to Jenkins with that specific tools & os command.
  + For doing anything in Jenkins we have to create a job.
    - * Eg, Go to GitHub 🡪 Download the code 🡪 show me output.
    - Job is also known as **item**.
    - If you want to execute job in Jenkins it is known as **Build**.
    - Jenkins will download data in their program, in their own OS.
    - For every job, Jenkins provides separate workspace.
      * Your data will be stored per job secured workspace.
* Jenkins have WebUI, CLI.
  + Nowadays also supporting As code (Pipeline as code).
    - It is something like cloud formation in AWS.
    - After knowing all things in AWS, we can use cloud formation to automate these things.
* Systemctl restart Jenkins
  + Netstat -tnlp | grep java
    - Jenkins is working on top of java.
  + Here you can see Jenkins is working on port 8080.
* To connect with Jenkins, you have to type this thing in chrome tab.
  + Localhost:8080
  + Final moto of Jenkins to create a Job for a Project.
* In our first task we are going to use date command using job.
  + For this you have to use **build shell**.
  + After creating a job you have to build it for running.
  + At left side down you can see history of jobs.
    - You can check console output from it.
* For changing something in the job you have to click on configure.
* You can also run your job from dashboard home.
  + At dashboard we have special symbol to check job status.
    - **Shiny** – working very good.
    - **Shiny + cloud** – working good, but job has some failure.
    - **Cloudy** – job has lots of failures.
  + It is for **visualize** because in real world we have multiple jobs.
* Jenkins with GitHub
  + Go to dashboard 🡪 manage Jenkins 🡪 available 🡪 search for github 🡪 download (GitHub plugin)
  + Create a repository & initialize it in GitHub.
    - Write basic rhel command in one file (script).
  + Create a Job.
    - In source code management you can option for Git.
      * This option comes from plugin.
    - Add your repository URL & branch name.
      * If URL does not exist it will give you error.
    - Click on execute shell.
      * Bash ./mytest.sh
    - Here you can see if you change your code & run job it will take some mini seconds time, but if you run job without changing anything then it will run faster.
  + Whenever you run any job with GitHub plugin, they will check for any change or not.
    - If there is no change, Jenkins will not download same code again.

**Session 03 – triggers//build periodically**

* In Jenkins we are creating jobs.
* Here we can do one thing.
  + We can tell Jenkins keep on checking GitHub account, whenever new code come up automatically **trigger** this job.
  + And that job will create a copy of GitHub code.
  + You can also connect to Jenkins dashboard from windows, but for this you have to disable firewall.
* Create a new GitHub repository and initialize it.
  + Create some html code in your base OS and push to github.
  + Install httpd in rhel and make it permanent enable.
* Create a new Job.
  + Add GitHub URL.
  + Build execute shell.
    - Copy code in your document root of rhel.
    - Sudo Cp -f \* /var/www/html
* But here we also have to build it, manually.
* Here we can use triggers.
  + Suppose some interval of time you create a backup.
  + For this we can use trigger **build periodically**.
  + Format for schedule.
    - Min hr date month day
    - Suppose you want every night 11:50
    - 50 23 \* \* \*
  + After setting trigger job will automatically run after particular interval of time.
* But it will impact a performance.
  + Every minute it is going to run.
  + Here we want some intelligence.
  + Jenkins will only go there only when something has been changed.
  + For this we can use **GitHub trigger**.

**Session 04 – Poll SCM // Github hook trigger // trigger builds remotely**

* Connect from public world to local vm using ngrok.
* We want as soon as something changed in GitHub, Jenkins will download it automatically.
* We have one build trigger – **Poll SCM**.
  + They will go to SCM for particular interval of time if something changed then only it will download code and run something for you.
  + But here we also we are wasting our resources.
  + We are not downloading and running job but every second we are going to GitHub which will waste resources.
* Instead of going manually we can ask GitHub to come to Jenkins when something is changed.
  + For this we can use GitHub events.
  + GitHub has concept called hooks.
  + For this we have to use **GitHub hook trigger with GitSCM pooling**.
  + You have to give IP & API of Jenkins which can listen GitHub.
  + For public IP we have to create tunnel.
    - ./ngrok http 8080
  + Set this inside GitHub webhooks.
    - <https://ip/github-webhook>/
  + Now when you change something in the code it will automatically trigger Jenkins.
* We can use one more trigger, **trigger builds remotely**.
  + It will give you one **API** when someone hit on that URL then it will run a job for you.
  + For this you have to set a token (password).
  + Now you will get one URL (API), you can do now whatever you want to do.
    - [https://admin:<os\_pass>@ip/job/<job\_no>/build?token=<token](https://admin:%3cos_pass%3e@ip/job/%3cjob_no%3e/build?token=%3ctoken)>

Session 05 –

NEW

* Kubectl cluster-info
* We have to connect K8s from Jenkins.
  + We have to download **Kubernetes plugin** for this.
  + For this we have to provide basic details to Jenkins, like IP, Port, User, Password.
* Suppose your developer has created a JAVA file and uploaded in the GitHUb.
  + We use maven to compile/build a package.
  + We create a image from it and push to Docker Hub.
* We have to provide config file to Jenkins plugin.
  + **Cd /etc/Kubernetes**
  + We want to transfer **admin.conf** file.
  + But we have to give permission so we can transfer it to windows using winSCP.
    - You have to change IP of cluster in this file.
* If you check we have **different** network card in Minikube VM.
  + We cant change minikube VM n/w card.
  + So we can **add** a new network card in Jenkins VM.
* Go to Jenkins 🡪 manage Jenkins 🡪 manage nodes and clouds 🡪 Configure clouds.
  + Select Kubernetes.
  + Kubernetes cloud details
    - Add credentials.
      * Select secret file option.
    - Add this credential and click on test.
    - In Jenkins URL they are asking you for Jenkins IP.
      * Here k8s VM will work as a Jenkins agent and current VM is Jenkins master.
      * Jenkins agent has to know where his master is.
      * So here we have to give current VM IP where Jenkins is running.
    - In pod label give some labels.
  + When Jenkins contact to Kubernetes they download default image from hub, launch a pod.
    - This image contains **Jenkins agent**.
    - Now Kubernetes becomes a worker node.
      * You can provide some extra things or change default image using **pod template**.
    - You have to give pod name and labels in pod template.
* Create a new Job 🡪 select on **restrict where this project can be run.** 
  + Give your label name here.
  + For simplicity we have used **date** command in build shell.
* Here we are using a port so Kubernetes pod can connect to this and using this pod it can connect to Jenkins.
  + For this you have to go inside Jenkins 🡪manage Jenkins 🡪 global security
  + Here you have to add a port number.
  + This port will work as a proxy and create a tunnel for you.
  + Many times we use port number 50000.
* You have to write this port number with IP address in Jenkins tunnel configure clouds.

Session 11 –

* Kubectl cluster-info
* We have to connect K8s from Jenkins.
  + We have to download **Kubernetes plugin** for this.
  + For this we have to provide basic details to Jenkins, like IP, Port, User, Password.
* Suppose your developer has created a JAVA file and uploaded in the GitHUb.
  + We use maven to compile/build a package.
  + We create a image from it and push to Docker Hub.
* We have to provide config file to Jenkins plugin.
  + **Cd /etc/Kubernetes**
  + We want to transfer **admin.conf** file.
  + But we have to give permission so we can transfer it to windows using winSCP.
    - You have to change IP of cluster in this file.
* If you check we have **different** network card in Minikube VM.
  + We cant change minikube VM n/w card.
  + So we can **add** a new network card in Jenkins VM.
* Go to Jenkins 🡪 manage Jenkins 🡪 manage nodes and clouds 🡪 Configure clouds.
  + Select Kubernetes.
  + Kubernetes cloud details
    - Add credentials.
      * Select secret file option.
    - Add this credential and click on test.
    - In Jenkins URL they are asking you for Jenkins IP.
      * Here k8s VM will work as a Jenkins agent and current VM is Jenkins master.
      * Jenkins agent has to know where his master is.
      * So here we have to give current VM IP where Jenkins is running.
    - In pod label give some labels.
  + When Jenkins contact to Kubernetes they download default image from hub, launch a pod.
    - This image contains **Jenkins agent**.
    - Now Kubernetes becomes a worker node.
      * You can provide some extra things or change default image using **pod template**.
    - You have to give pod name and labels in pod template.
* Create a new Job 🡪 select on **restrict where this project can be run.** 
  + Give your label name here.
  + For simplicity we have used **date** command in build shell.
* Here we are using a port so Kubernetes pod can connect to this and using this pod it can connect to Jenkins.
  + For this you have to go inside Jenkins 🡪manage Jenkins 🡪 global security
  + Here you have to add a port number.
  + This port will work as a proxy and create a tunnel for you.
  + Many times we use port number 50000.
* You have to write this port number with IP address in Jenkins tunnel configure clouds.
* **Session 01 – Intro** 
  + Jenkins intro
  + CI tool
  + Infinity cylcle
* **Session 02 – installation//plugin//demo** 
  + Java – JDK
  + Rhel installation
  + Plugins
    - GitHub
  + Job
  + Build
  + Shiny-cloudy ball for visualization
  + 2 demos
  + Build shell
  + Command
    - Rpm -ivh jdk-….rpm
    - Rpm -ivh jen…
    - Systemctl restart Jenkins
* **Session 03 – triggers//** 
  + Trigger
    - Build periodically
    - Min hr date month day
* **Session 04 –** **Poll SCM // Github hook trigger // trigger builds remotely**
  + Poll SCM
    - interval of time
  + GitHub hook trigger with GitSCM pooling
    - GitHub webhooks
    - <https://ip/github-webhook>/
  + Trigger builds remotely
    - API
    - [https://admin:<os\_pass>@ip/job/<job\_no>/build?token=<token](https://admin:%3cos_pass%3e@ip/job/%3cjob_no%3e/build?token=%3ctoken)>
* Session 05 –