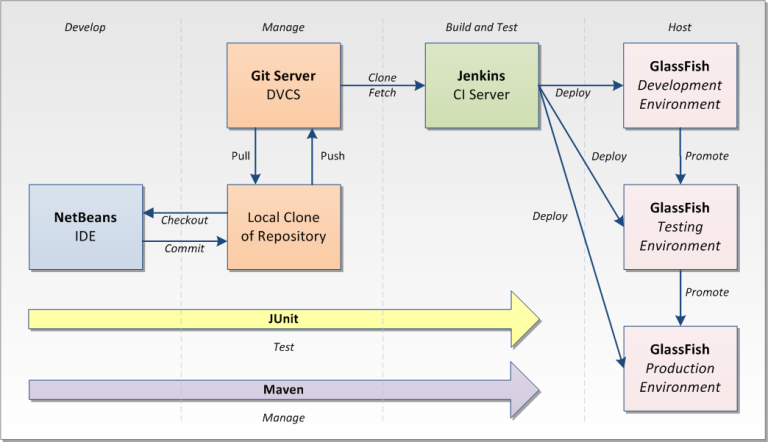
[Jenkins](javascript:void(0);)[Scores](javascript:void(0))

**Objectives**

* Demonstrate setting up Jenkins to get code from GitLab and build the maven project
  + About Jenkins, build automation, Jenkins installation and user setup, plugins for GitLab and Maven, configure GitLab with API authentication token, configure GitLab SSL Certificate, configure git.exe path and maven path, setup maven project, build project
    - Jenkins - https://jenkins.io/doc/
    - Jenkins Java - https://jenkins.io/solutions/java/

**About Jenkins**

* Open source automation server
* Automate build and deployement of a project
* Refer diagram below which has a sample continuous integration server setup:
  + In our scenario it is git bash client instead of NetBeans IDE
  + Git Server is the gitlab server https://code.cognizant.com
  + Ideally in real time environment Jenkins should be a separate server. For our learning purpose, we will use the local desktop as build server environment.
  + In our case it is "tomcat" server instead of "GlassFish"

SME to explain the diagram in detail.  
  
[ *Picture Courtesy*:  <https://programmaticponderings.com/2013/11/13/building-a-deployment-pipeline-using-git-maven-jenkins-and-glassfish-part-2-of-2/>]  


**Jenkins Installation**

Follow steps below to install Jenkins:

**Downloads**

* SME to download following files from SharePoint and share them with the learners:
* SME to download the following files from the SharePoint path GenCSharePath > GenC Shared Content > OBL - Java FSE > software:
  + jenkins.war
  + jenkins-plugins.zip

**Start Jenkins**

* In command prompt go to the folder where jenkins.war is downloaded and execute the below command to start Jenkins

java -jar jenkins.war

* On execution of the above command the log rolls. Look for the generated admin password in the log. copy the password and have it future reference
* Open http://localhost:8080 in browser
* This will open the screen to enter the admin password.
* Get the password from where you had stored or get the password from initialAdminPassword file in "C:\Users\<YOUR\_EMP\_ID>\.jenkins\secrets" folder.
* Enter the admin password
* Select the "Select plugins to install" option
* Due to proxy issues plugin installation will be done later, so select the option "None", so that plugins installation will not be initiated by Jenkins.
* Click "Install" button, which will not initiate plugin installation and will display a form for user details
* Enter employee id in username, fill up other details and click "Save and Continue"
* In the next screen verify the URL and click "Save and Finish" button
* It should display the message "Jenkins is ready"
* Select "Start using Jenkins", which will open the dashboard.
* Sign out once and check if login is working.
* Logout and close the browser window, we  will login after installation of plugin.

**Plugin installation (without internet access)**

* Stop the jenkins server by pressing Ctrl+C in the command prompt window where jenkins server is running
* Get the jenkins-plugins.zip file from the SME. The zip file is available in the same folder as the jenkins.war file
* In Windows File Explorer go to folder D:\Users\<EMP\_ID>\.jenkins and delete the plugins folder
* Copy the plugins folder from jenkins-plugins.zip file and paste the contents in the .jenkins folder
* Start jenkins server by running "java -jar jenkins.war" command in command prompt
* Login into jenkins in browser using your employee id and password
* Subsequent hands on will have steps to configure git and maven to build the project

**Configure Maven and Git path**

* Login into Jenkins
* Go to "Manage Jenkins" > "Global Tool Configuration"
* Git Path
  + Go to "Git" section
  + Click "Add Git"
  + Give "Name" as "Default"
  + Open windows explorer and find out where Git is installed. Check in "Program Files" folder of C drive.
  + "Path to Git executable" should look something like "C:\Program Files\Git\bin\git.exe".
* Maven Path
  + Go to "Maven" section
  + In "Name" provide "Maven" along with version number
  + In "MAVEN\_HOME" provide the root path of maven. (Example: D:\apache-maven-3.5.2)
* Click "Save" to save the configuration

**Build spring-learn project in Jenkins**  
  
**Configure spring-learn project build**

* Login into Jenkins
* Click "New Item"
* Enter item name as "spring-learn-build"
* Click "Maven Project"
* Click "OK"
* In "Source Code Management" section select Git
* In "Repository URL" provide your spring-learn project URL with ".git" suffixed
* Click "Add" > "Jenkins"
* Select "Kind" as "Username with password"
* Provide "Username" as employee id
* Provide "Password" as network password
* Provide "ID" as "gitlab-credentials"
* Provide "Description" as "gitlab-credentials"
* Click "Add"
* In Credentials drop down select "[EMP\_ID]/\*\*\*\*\* (git-credentials)"
* If git configuration give SSL certificate error, execute the below command in GitBash

git config --global http.sslVerify false

* Reopen the spring-learn-build configuration to check if the SSL certificate error is gone
* In Build section in "Goals and options" provide value as "clean package"
* Click "Save"

**Executing the build**

* Now in right hand side of the Jenkins dashboard the "spring-learn-build" project will be listed
* Click on "spring-learn-build"
* Click "Build Now" in the left hand side menu
* In the left hand side menu bottom, there will be a section "Build History"
* In "Build History" section there will be a blinking icon
* Click on the blinking icon to view the build progress log
* Check if the build is successful and the JAR file is generated.
* If there are any test cases that fail, make changes to have successful test execution and make the project build status as green.
* In windows explorer go to C:\Users\[EMP\_ID]\.jenkins\workspace\spring-learn-build to see the code download from git and build output available in target folder.