*DevOps*

**DevOps**

(Tutorial – Channel Name: Logiclabstech recordings on Google Drive)

**First Published On: 24th Jun 2021**

**Last Updated On: 24th Jun 2021**

**DevOps**

|  |
| --- |
| **Agenda** |

|  |  |
| --- | --- |
| **CI CD Pipeline**              **Installation**   1. **Install Git** 2. **Install Java**   [**https://www.oracle.com/in/java/technologies/javase-downloads.html**](https://www.oracle.com/in/java/technologies/javase-downloads.html)              **Installing Maven**        **Install Jenkins**  **NOTE: Jenkins works only with JRE 8 or 11**  **So follow the above java installation steps for JDK 11 instead of JDK 16**  <https://www.jenkins.io/>          **To check errors/info of your installation**  C:\Program Files\Jenkins        **First JOB/Build**                **Installing Plugins**              **Build Maven Job**  <https://github.com/technicalguftgu/time-tracker>        **Build Maven Job using Jenkins**            **Source Code Polling – Periodically**        **Source Code Polling – Only when changes are made**    **Upstream Jobs**      **Upstream Jobs**    **Views**      **User Management**              **Master Slave**      **-----**    **-----**    **-----**    **Assigning dedicated Slave to the Job** | **CI CD Pipeline**   * Developer writes a code * After each small chunk of code, he pushes the code to Version Control Repository * As soon as the new code is pushed, CI Server will pull the changes and Build, Test and Deploy * If at any stage there is a problem, the server will notify it to the developer. * Consider an example, if Pipeline automation was not there, developer would write 50k lines of code and then push it to Version Control System * Once the code is pushed, testing team will integrate and test which may take 1 or 2 weeks * Once the testing team finds errors, they notify developer. **But there is a problem**, it will be very hard to detect the issue in 50k lines of code. Also he might be working on some other project. * To solve such issues, **Pipeline** concept is introduced * The steps are in sequential manner; hence it is called as pipeline * At each stage, feedback is sent to the developer (error/success) * Continuous Integration Tool * Alternate tool – Bamboo, Travis CI, Buildbot, but all are paid * CICD – Continuous Integration Continuous Delivery * Delivery – to send your product to client so that they can use their tem to deploy and run the product * Deployment – directly deploy the product to client environment   **Installation**  **First JOB/Build**   * Create a new Job or Build * Choose Freestyle project * For testing, we will use Windows batch command. * Once done, go back to dashboard, and on the FirstJob, click the dropdown and select Build Now. * Successful build will show a green tick * Observe a weather indicator, the Sun means that the job has executed successfully * Go to the job by clicking it and click on the green tick to observe the output of command * In case if the job fails, it will show red cross and also the weather icon will change * In case if the job now is corrected and executes successfully, the green tick will again appear but the weather will be different as the job was errored previously. * Install Maven Integration plugin * Green Balls is optional just used to show the UI (not supported in newer versions) * A new option will be available while creating a new Item * Let’s add the necessary paths to the Jenkins to let Jenkins know which tools are sitting where. * Mention the Path where the Maven is installed\_   **Build Maven Job**   * Fork the project given in the link * Clone it in your local directory * *mvn clean package* to build the project using maven   **Build Maven Job using Jenkins**   * Let’s do the above same thing using Jenkins * *clean package* command to let Jenkins know this command to use when triggered.   **Source Code Polling – Periodically**   * The stars mean that first star is every minute, second is every hour, third is day, fourth is month and last star is day of the week * So five stars (\* \* \* \* \*) mean every minute * Observe that the jobs are running every minute * But this is not productive, as the job will run every minute even if the code is not updated.   **Source Code Polling – Only when changes are made**   * In this case, the job will be polled every minute but will trigger only when there are changes in the code.   **Upstream Jobs**   * Used when second job needs to be triggered on first job completes * In this case the first job is responsible to invoke the second job   **Upstream Jobs**   * In this case, we will configure in second job to let know that it needs to be triggered when first job is completed. * By Default, the user gets all the admin privileges * To manage access and permissions, install this plugin * Restart Jenkins just by typing /restart * Next, login again and goto Configure Global Security option. * Enable Role-Based Strategy   **Master Slave**   * You can have multiple slaves connected to Master * This is done so that the jobs are distributed among all the nodes (slaves). * The jobs are assigned to nodes and master in random order. * Assign a root directory where the Slave will be stored * If an another new slave is created, then another directory should be created * Click the “?” icon and download the agent.jar to a specified folder * Give command java –jar <<downloaded agent path>> * Observe that the new slave is created and waiting for new jobs/builds.   **Assigning dedicated Slave to the Job** |