Maven –

Apache Maven is a software project management and build management tool for Java Frameworks.

1. Why Maven?

* Central repository to get dependencies
* Maintaining common structure across the organization
* Flexibility in Integrating with CI tools
* Plugins for Test framework execution.

1. Install Maven
2. Set System variables to recognize Maven
3. Understanding Maven terminologies

**Artifact**: An artifact is a file, usually a JAR, that gets deployed to a Maven repository.

**GroupId**: groupId will identify your project uniquely across all projects,

**archetype:generate** ; Generates a new project from an archetype

1. Creating Maven Project

mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

1. Integrate Maven with Eclipse
2. Maven Phases for Test Automation
3. Clean, compile, test
4. Understanding POM.xml file dependencies..
5. How to find the dependencies of the Softwares?
6. Importance of Maven Sure fire plugin
7. Running Tests with Surefireplugin
8. Integrating Testng into Maven
9. Testng xml files configuration in POM file
10. Setting up Profiles in POM.xml file
11. Running selected Tests only with Maven commands
12. Maven Sure fire reports

JENKINS

1. Importance of Jenkins
2. Jenkins Installation
   1. Java -jar Jenkins.war → Jenkins will be installed → Goto browser and navigate to localhost:8080
      1. Enter password
      2. Install plugins
      3. Copy and paste HOME locations for Java and Maven as well as their PATH locations
3. Configuring Global settings in Jenkins
4. Understanding Jenkins Workspace
5. Configuring Jenkins Job parameters
6. Post build action plugins
7. TestNG Jenkins plugin to generate Reports
8. Scheduling Jenkins builds