**Continuous Integration (CI):**

Continuous Integration (CI) is the process of automating the build and testing of code every time a team member commits changes to version control/GIT.

**Continuous Delivery (CD)**

Continuous Delivery (CD) is the process to build, test, configure and deploy from a build to a production environment

Classic Diagram is available to explain these terms in below AWS Website

<https://aws.amazon.com/devops/continuous-integration/>

Basics of Deployment:

**What are Application Servers**   
An **application/Web server** is a **server** specifically designed to run applications (or) It is server to host applications.

It handles Http requests and send response calls over Http protocol.

**What are Hosted Servers?**  Hosted Servers are nothing but physical machines where application/Web/Databases servers are hosted

199.323.434.53:8080  
  
- Application server can start at any port in Hosted Server. And can be accessed through  
HostedServerIpAddress: portNumber

We can map HostedServerIpAddress: portNumber to any custom Domain Name so that we can access the application server directly with Domain Name

Dist- ng -build

Mvn clean install

To keep our Developed App running on server and access it from anywhere, we should first build the code so that build files will be generated. These build files should be placed into App/Web Server   
And then we can access the developed App directly on the browser with HostedServerIpAddress :portNumber or with Domain Name if mapped.

Java, Maven & Jenkins setup on AWS Linux EC2 Instance

1. **Create AWS EC2 Instance**
2. **Download MobaXTerm to connect to EC2 Instance**
3. **What is Yum?**  
    **yum** is the primary tool for getting, installing, deleting, querying, and managing software Packages

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/install-software.html>

1. **Download Java**  
   <http://openjdk.java.net/install/>

Install Java in Linux

**yum** install java-1.8.0-openjdk-devel

Java is installed in usr/lib directory

Set Java Home Path in Bash profile  
 JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.252.b09-2.51.amzn1.x86\_64  
 PATH=$PATH:$HOME/bin:$JAVA\_HOME:

1. **Install Maven in Linux**

**wget**  
Wget solely lets you download files from an HTTP/HTTPS or FTP server. You give it a link and it automatically downloads the file where the link points to. It basically helps to download Binaries

wget <downloadlinkofMaven>  
tar xzvf <mavenfoldername>

Set Maven in System variables

JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.252.b09-2.51.amzn1.x86\_64

MAVEN\_HOME=/opt/apache-maven-3.6.3

M2=/opt/apache-maven-3.6.3/bin

PATH=$PATH:$HOME/bin:$JAVA\_HOME:$MAVEN\_HOME:$M2

1. **Install Jenkins**  
   <https://wiki.jenkins.io/display/JENKINS/Installing+Jenkins+on+Red+Hat+distributions>

* sudo wget -O /etc/yum.repos.d/jenkins.repo <http://pkg.jenkins.io/redhat/jenkins.repo>
* sudo rpm --import <http://pkg.jenkins.io/redhat/jenkins.io.key>
* sudo yum install jenkins
* Start Jenkins  
  sudo service jenkins start/stop/restart
* sudo chkconfig jenkins on

1. Create New Jenkin Job and verify War file generation

Deploying App into another Server with Jenkins and Start the Application on Docker Container

Phase 1:

* Spin up New AWS Instance for deploying application  
  Create New User with Password

passwd ec2-user (Make sure you are in root)

* Enable Password Authentication in below File Path  
  vi /etc/ssh/sshd\_config
* Reload Service with service sshd reload
* Add the Server into Jenkins from Manage Jenkins
* Download Publish over SSH Plugin in Jenkins
* Add Docker Server SSH to Jenkins in the job
* Run the Deployment Job to deploy the artifacts into Application Server from Jenkins Server

Phase 2 :

* Install Docker with – yum install docker
* cat /etc/group - to see all groups available for root user
* giving permission access to group  
  **usermod -aG docker ec2-user**
* Login as ec2-user and start docker service  
  sudo service docker start
* Pull Tomcat Image from Docker hub and start container with below command
* docker pull tomcat:latest  
  sudo docker run -d --name tomcat-container -p 8080:8080 tomcat:latest
* Check if Container is up and running with below command  
  docker ps
* Get Access to container with below command and see where tomcat is installed inside the container  
  Use the command docker exec -it <container name> /bin/bash to get a bash shell in the container
* Understand where should Webapp.war file should be placed in Tomcat in Container
* Create Docker file which can pull Tomcat Image and perform necessary configurations to deploy war file
* Create Image from Docker file

docker build -t tomcatimage .

* Create and Run Container on created Image   
  docker run -d --name newcontainer -p 8080:8080 tomcatimage
* Access the Application with IPAddress:Port

**SSH Connection to another Linux Server/LocalHost**

* Try SSh from Server X (Application Server) into another Server Y with below command  
  ssh user@<ipaddr>
* Now go to the Server Y and update Password Authentication to Yes and provide password to the ec2-user  
  vi /etc/ssh/sshd\_config
* service sshd reload
* Come back to Server X and Create SSH Keys with below command  
  ssh-keygen
* Give below command to copy Public Key Server to Y through ssh  
  ssh-copy-id user@<target-server> - Authorized\_keys
* You are all Set to connect remotely to Server Y from X  
  ssh user@<ipaddr>

SSH\_Localhost – private and public key -

**How Does Key-based Authentication Work?**

Key-based authentication works by creating a pair of keys: a private key and a public key.

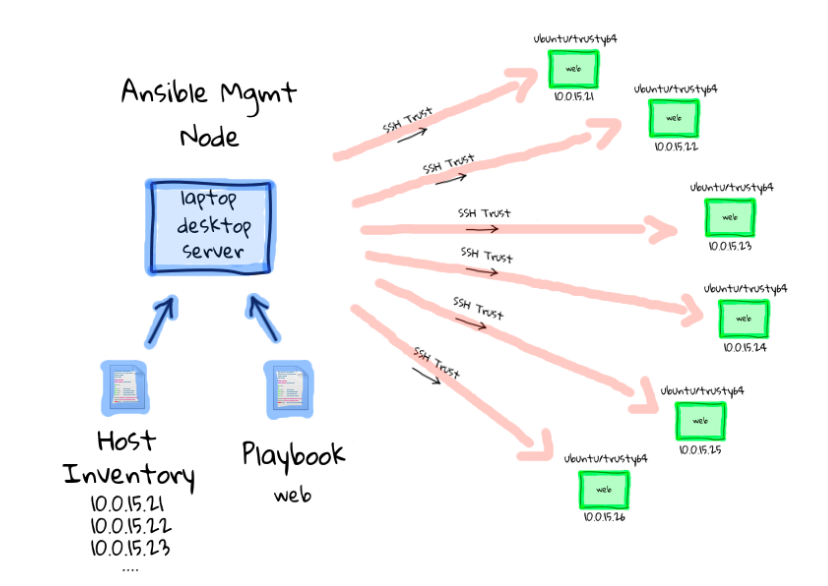
The private key is located on the client machine and is secured and kept secret.

The public key can be given to anyone or placed on any server you wish to access.

When you attempt to connect using a key-pair, the server will use the public key to create a message for the client computer that can only be read with the private key.

What is Ansible?

Ansible is a simple IT automation engine that automates all the commands you can provide on terminal  
it uses a very simple language (YAML, in the form of Ansible Playbooks) that allow you to describe your automation jobs in a way that approaches plain English.  
Ansible works by connecting to your nodes and pushing out small programs, called "Ansible modules" to them  
Ansible then executes these modules (over SSH by default), and removes them when finished.



To Install Ansible

1. We need to first install Python first with below command  
yum install python

2. From Pip Package Manager install Ansible with below command

pip install ansible

3. After installation Check if it is detecting in the system with below command  
ansible –version

4. Add Sudo Access to ec2-user

**echo "ec2-user ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers**

docker stop customcontainer  
docker rm customcontainer

docker rmi customimage

**Create Ansible Playbook with your requirements**

**Create Inventory/hosts file**

**Cross check if you can give SSh connection to the host you are connecting**  
  
**Run playbook with below command**  
ansible-playbook -i <hostsfile> <playbook.yml>

---

- hosts: all

tasks:

- name: copy Dockerfile into QA server

copy:

src: Dockerfile

dest: .

- name: copy war file into Remote machine

copy:

src: webapp.war

dest: .

- name: stop the running container

command: docker stop customcontainer

ignore\_errors: True

- name: remove the stopped container

command: docker rm customcontainer

ignore\_errors: True

- name: remove the container image

command: docker rmi customimage

ignore\_errors: True

- name: create container image

command: sudo docker build -t customimage .

- name: create container

command: sudo docker run -d -p 8080:8080 --name customcontainer customimage

**Summary**:

Using Ansible we can write playbooks which can run automation script with set of commands on any machine.

You can deploy your code into n number of machines by having IP Address list in hosts file and running Ansible deployment playbook yaml file

Greatest advantage of deploying Apps in Docker Containers is that we need not worry about existing software installations and their version compatibility

We can create Deployment environment in docker file so that it creates image out of it and deploy in container in any machine instantly with out worrying about downloading the softwares manually

Deploying Apps using Ansible + Docker on Linux and automating the Continuous Delivery process with Jenkins is the powerful Process followed in Devops World.