Install Jenkins on AWS EC2

Jenkins is a self-contained Java-based program, ready to run out-of-the-box, with packages for Windows, Mac OS X and other Unix-like operating systems. As an extensible automation server, Jenkins can be used as a simple CI server or turned into the continuous delivery hub for any project.

Prerequisites

1. EC2 Instance
   - With Internet Access
   - Security Group with Port 8080 open for internet
2. Java 11 should be installed

Install Jenkins

You can install jenkins using the rpm or by setting up the repo. We will set up the repo so that we can update it easily in the future.

1. Get the latest version of jenkins from https://pkg.jenkins.io/redhat-stable/ and install
2. sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
3. sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
4. amazon-linux-extras install epel
5. amazon-linux-extras install java-openjdk11
6.
7. #on RedHat/CentOs
8. #yum install epel-release # repository that provides 'daemonize'
9. #yum install java-11-openjdk-devel
   #yum install jenkins

Start Jenkins

```
# Start jenkins service
service jenkins start

# Setup Jenkins to start at boot,
chkconfig jenkins on
```

Accessing Jenkins

By default jenkins runs at port 8080, You can access jenkins at
http://YOUR-SERVER-PUBLIC-IP:8080

Configure Jenkins

- The default Username is admin
- Grab the default password
- Password Location:/var/lib/jenkins/secrets/initialAdminPassword

- Skip Plugin Installation; *We can do it later*
- Change admin password
  - Admin > Configure > Password
- Configure java path
  - Manage Jenkins > Global Tool Configuration > JDK
- Create another admin user id

Test Jenkins Jobs

1. Create “new item”
2. Enter an item name – My-First-Project
   - Chose Freestyle project
3. Under the Build section Execute shell: echo "Welcome to Jenkins Demo"
4. Save your job
5. Build job
6. Check "console output"

Install & configure Maven build tool on Jenkins

Maven is a code build tool which used to convert your code to an artifact. this is a widely used plugin to build in continuous integration

Prerequisites

1. Jenkins server

Install Maven on Jenkins

1. Download maven packages https://maven.apache.org/download.cgi onto Jenkins server. In this case, I am using /opt/maven as my installation directory

- Link : https://maven.apache.org/download.cgi
- # Creating maven directory under /opt
- mkdir /opt/maven
- cd /opt/maven
- # downloading maven version 3.6.0
- wget http://mirrors.estointernet.in/apache/maven/maven-3/3.6.1/binaries/apache-maven-3.6.1-bin.tar.gz
  tar -xvzf apache-maven-3.6.1-bin.tar.gz

1. Setup M2_HOME and M2 paths in .bash_profile of the user and add these to the path variable
2. vi ~/.bash_profile
3. M2_HOME=/opt/maven/apache-maven-3.6.1
4. M2=$M2_HOME/bin
   PATH=<Existing_PATH>:$M2_HOME:$M2

Checkpoint

1. logoff and login to check maven version

```
mvn --version
```

So far we have completed the installation of maven software to support maven plugin on the jenkins console. Let's jump onto Jenkins to complete the remaining steps.

Setup maven on Jenkins console

1. Install maven plugin without restart

- Manage Jenkins > Jenkins Plugins > available > Maven Invoker
- Manage Jenkins > Jenkins Plugins > available > Maven Integration

2. Configure maven path

- Manage Jenkins > Global Tool Configuration > Maven

Configure Git pulgin on Jenkins

Git is one of the most popular tools for version control system. you can pull code from git repositories using jenkins if you use github plugin.

Prerequisites

1. Jenkins server

Install Git on Jenkins server

1. Install git packages on jenkins server

```
yum install git -y
```

Setup Git on jenkins console

- Install git plugin without restart
    - Manage Jenkins > Jenkins Plugins > available > github
- Configure git path
    - Manage Jenkins > Global Tool Configuration > git

Ansible integration with Jenkins

Prerequisites:

1. Ansible server
2. Jenkins Server

Part-01 Integration Setps

Install "publish Over SSH"

- Manage Jenkins > Manage Plugins > Available > Publish over SSH

Enable connection between Ansible and Jenkins

- Manage Jenkins > Configure System > Publish Over SSH > SSH Servers
    - SSH Servers:
        - Hostname:<ServerIP>
        - username: ansadm
        - password: *******

Test the connection "Test Connection"

Deploy on a docker container using Jenkins

Deploy_on_Container

Pre-requisites

- Jenkins server
- Docker-host server
- Dockerfile under */home/dockeradmin* in user home directory on docker host
- # Pull tomcat latest image from dockerhub
- From tomcat
- # Maintainer
- MAINTAINER "mohit1talmale"
- # copy war file on to container
  COPY ./webapp.war /usr/local/tomcat/webapps

Integration between Docker-host and Jenkins

Install "publish Over SSH"

- Manage Jenkins > Manage Plugins > Available > Publish over SSH

Enable connection between Docker-host and Jenkins

- Manage Jenkins > Configure System > Publish Over SSH > SSH Servers
  - SSH Servers: - Name: docker-host
    - Hostname:<ServerIP>
    - username: dockeradmin
    - Advanced > chose Use password authentication, or use a different key
    - password: *******

Steps to create "Deploy_on_Container" Jenkin job

From Jenkins home page select "New Item"

- Enter an item name: Deploy_on_Container
  - Copy from: Deploy_on_Docker_Host
- *Source Code Management:*
  - Repository: https://github.com/yankils/hello-world.git
  - Branches to build : */master
- *Poll SCM* : - * * * *
- *Build:*
  - Root POM:pom.xml
  - Goals and options: clean install package
- *Post-build Actions*
  - Send build artifacts over SSH
    - *SSH Publishers*
    - SSH Server Name: docker-host
    - Transfers > Transfer set
      - Source files: webapp/target/*.war
      - Remove prefix: webapp/target
      - Remote directory: //home//ansadmin or .
      - Exec command:
      - cd /home/dockeradmin;
      - docker build -t simple-devops-image .;
        docker run -d --name simple-devops-container -p 8080:8080 simple-devops-image;

Save and run the job now.

Deploy on a docker container using Ansible

Jenkins Job name: Deploy_on_Container_using_ansible

Pre-requisites

1. Jenkins server
2. Docker-host server
3. Ansible server
4. Dockerfile under */opt/docker* on Ansible server

```
5.  # Pull tomcat latest image from dockerhub
6.  From tomcat
7.  # Maintainer
8.  MAINTAINER "AR Shankar"
9.
10. # copy war file on to container
    COPY ./webapp.war /usr/local/tomcat/webapps
```

11. Create create-docker-image.yml unser */opt/docker* on Ansible server

```
12. ---
13. - hosts: all
14.   #ansadmin doesn't need root access to create an image
15.   become: true
16.
17.   tasks:
18.   - name: building docker image
19.     command: "docker build -t simple-devops-image ."
20.     args:
        chdir: /opt/docker
```

21. Create create-docker-image.yml under */opt/docker* on Ansible server

```
22. ---
23. - hosts: all
24.   become: ture
25.
26.   tasks:
27.   - name: creating docker image using docker command
28.     command: docker run -d --name simple-devops-container -p 8080:8080 simple-devops-image
```

Integration between Ansible-control-node and Jenkins

Install "publish Over SSH"

- Manage Jenkins > Manage Plugins > Available > Publish over SSH

Enable connection between Ansible-control-node and Jenkins

- Manage Jenkins > Configure System > Publish Over SSH > SSH Servers
    - SSH Servers: - Name: ansible-server
        - Hostname:<ServerIP>
        - username: ansadmin
        - Advanced > chose Use password authentication, or use a different key
        - password: *******

Steps to create "Deploy_on_Container_using_ansible" Jenkin job

From Jenkins home page select "New Item"

- Enter an item name: Deploy_on_Container_using_ansible
    - Copy from: Deploy_on_Container
- *Source Code Management:*
    - Repository: https://github.com/yankils/hello-world.git
    - Branches to build : */master
- *Poll SCM* : - * * * *
- *Build:*
    - Root POM:pom.xml
    - Goals and options: clean install package
- *Post-build Actions*
    - Send build artifacts over SSH
        - *SSH Publishers*
        - SSH Server Name: ansible-server
        - Transfers > Transfer set
            - Source files: webapp/target/*.war
            - Remove prefix: webapp/target
            - Remote directory: //opt//docker
            - Exec command:

```
ansible-playbook -i /opt/docker/hosts /opt/docker/create-docker-image.yml;
```

Save and run the job now.

Deploy on a Tomcat server

***Jenkins Job name*: Deploy_on_Tomcat_Server**

Pre-requisites

1. Jenkins server
2. Tomcat Server

Adding Deployment steps

1. Install 'deploy to container' plugin. This plugin needs to deploy on tomcat server.

- Install 'deploy to container' plugin without restart
  - Manage Jenkins > Jenkins Plugins > available > deploy to container

2. Jenkins should need access to the tomcat server to deploy build artifacts. setup credentials to enable this process. use credentials option on Jenkins home page.

- setup credentials
  - credentials > jenkins > Global credentials > add credentials
    - Username : deployer
    - Password : deployer
    - id : deployer
    - Description: user to deploy on tomcat vm

Steps to create "Deploy_on_Tomcat_Server" Jenkin job

From Jenkins home page select "New Item"

- Enter an item name: Deploy_on_Tomcat_Server
    - Copy from: My_First_Maven_Build
- *Source Code Management:*
    - Repository: https://github.com/yankils/hello-world.git
    - Branches to build : */master
- *Poll SCM* : - * * * *
- *Build:*
    - Root POM:pom.xml
    - Goals and options: clean install package
- *Post-build Actions*
    - Deploy war/ear to container
        - WAR/EAR files : **/*.war
        - Containers : Tomcat 8.x
            - Credentials: deployer (user created on above)
            - Tomcat URL : http://<PUBLIC_IP>:8080

Save and run the job now.

Dockerfile

```
#Pull tomcat latest image from dockerhub

From tomcat:latest
# Maintainer
MAINTAINER "mohit1talmale"
# copy war file on to container
COPY ./webapp.war /usr/local/tomcat/webapps
```

Create a First Maven Jenkins job to build hello-world project

Jenkins Job name: **My_First_Maven_Build**

We know how to use work with each and Git, Jenkins independently. What if you want to collaborate these two? that is where Simple DevOps project helps you. Follow the below steps if you are a new guy to DevOps. You love it.

Pre-requisites

1. Jenkins server

Steps to create "My_First_Maven_Build" Jenkin job

1. Login to Jenkins console
2. Create *Jenkins job*, Fill the following details,
   - *Source Code Management:*
     - Repository: https://github.com/yankils/hello-world.git
     - Branches to build : */master
   - *Build:*
     - Root POM:pom.xml
     - Goals and options: clean install package

Create-docker-container.yml

```
# Option-1 : Createting docker container using command module



---
- hosts: all
  become: true
  tasks:
  - name: creating docker image using docker command
    command: docker run -d --name simple-devops-container -p 8080:8080 simple-devops-image

# option-2 : creating docker container using docker_container module
#  tasks:
#  - name: create simple-devops-container
#    docker_container:
#      name: simple-devops-container
#      image: simple-devops-image
#      state: present
#      recreate: yes
#      ports:
#       - "8080:8080"
```

Create-docker-image.yml

# Option-1 : Createting docker image using command module

```
---
- hosts: all
  become: true
  tasks:
  - name: building docker image
    command: docker build -t simple-devops-image .
    args:
      chdir: /opt/docker
# option-2 : creating docker image using docker_image module
#  tasks:
#  - name: building docker image
#    docker_image:
#      build:
#        path: /opt/docker
#      name: simple-devops-image
#     tag: v1
#     source: build
```

Simple docker project

```
---
- hosts: all
  become: true
  tasks:
  - name: stop if we have old docker container
    command: docker stop simple-devops-container
    ignore_errors: yes
  - name: remove stopped docker container
    command: docker rm simple-devops-container
    ignore_errors: yes
  - name: remove current docker image
    command: docker rmi simple-devops-image
    ignore_errors: yes
#    register: result
#    failed_when:
#      - result.rc == 0
#      - '"docker" not in result.stdout'
  - name: building docker image
    command: docker build -t simple-devops-image .
    args:
      chdir: /opt/docker
  - name: creating docker image
    command: docker run -d --name simple-devops-container -p 8080:8080 simple-devops-image
```