

## Kubernetes pull docker image using jenkins

- Requirements:
- java executable program
- machine with jenkins & docker
- and machine with kubernetes
- kubernetes connected nodes

### Step1:

First we create java runnable programs and pull in to your git  
And create pom.xml file to create war or jar file  
i get it from my old git java file. Here link for this

<https://github.com/cmmani/java-app>

### Step2:

Here our project start

first we install docker from jenkins machine or any bare machine.  
But i install docker in jenkins machine.

### Step3:


create new jenkins directory to create war file in local jenkins machine





## Create jenkins in freestyle project:


› All ›


**Enter an item name**  
  
» Required field


**Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**GitHub Organization**  
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

**Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.

## Get a java application from git and add credential:

**Source Code Management**

☐ None

☒ Git

Repositories

Repository URL

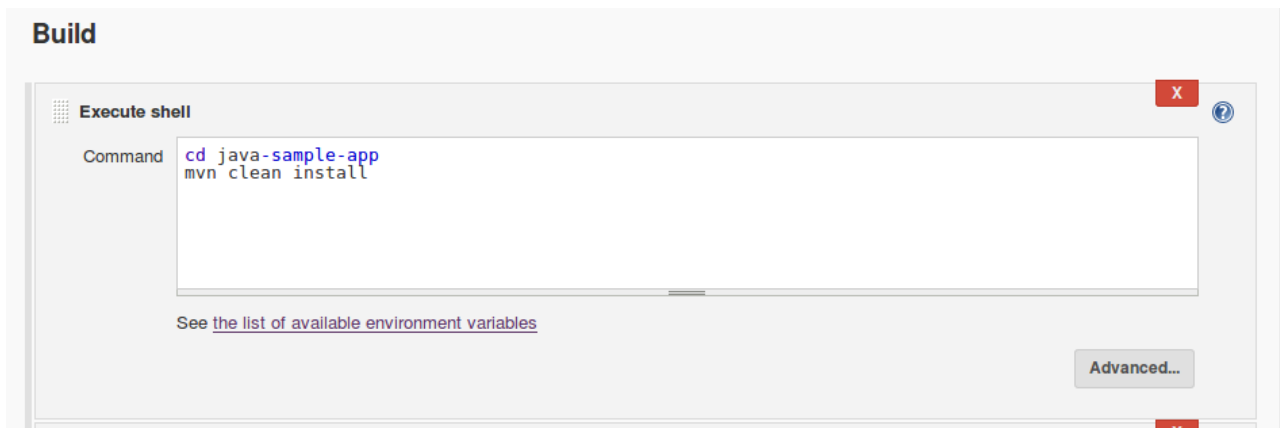
Credentials

Branches to build

Branch Specifier (blank for 'any')

Repository browser

Additional Behaviours



#### Step4:

Then build now the project to deploy War file in jenkins workspace

#### Step5:

Go to the jenkins path

```
$ cd /var/lib/jenkins/workspace/[define your jenkins projectname]/target$ ls
```

```
$ ls
```

You can see the war file in that folder

**JAVA\_APP.war**

#### Step 6:

Create docker tomcat container to deploy war file

```
$ vi Dockerfile
```

```
FROM tomcat
```

here you run a tomcat container;

And tomcat need authentication to enter in to the app folder

So we need to set root password and username for tomcat

```
target>>
```

```
$ vi contexts.xml
```

```
<Context antiResourceLocking="false" privileged="true" >
```

```
  <!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"
```

```
    allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->
```

```
  <Manager sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|Number|String)|org\.apache\.catalina\.filters\.CsrfPreventionFilter|$LruCache(?:\s1)?|java\.util\.(?:Linked)?HashMap"/>
```

```
</Context>
```

```
$ vi tomcat-user.xml
```

```
<tomcat-users xmlns="http://tomcat.apache.org/xml"
```

```
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
  xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
```

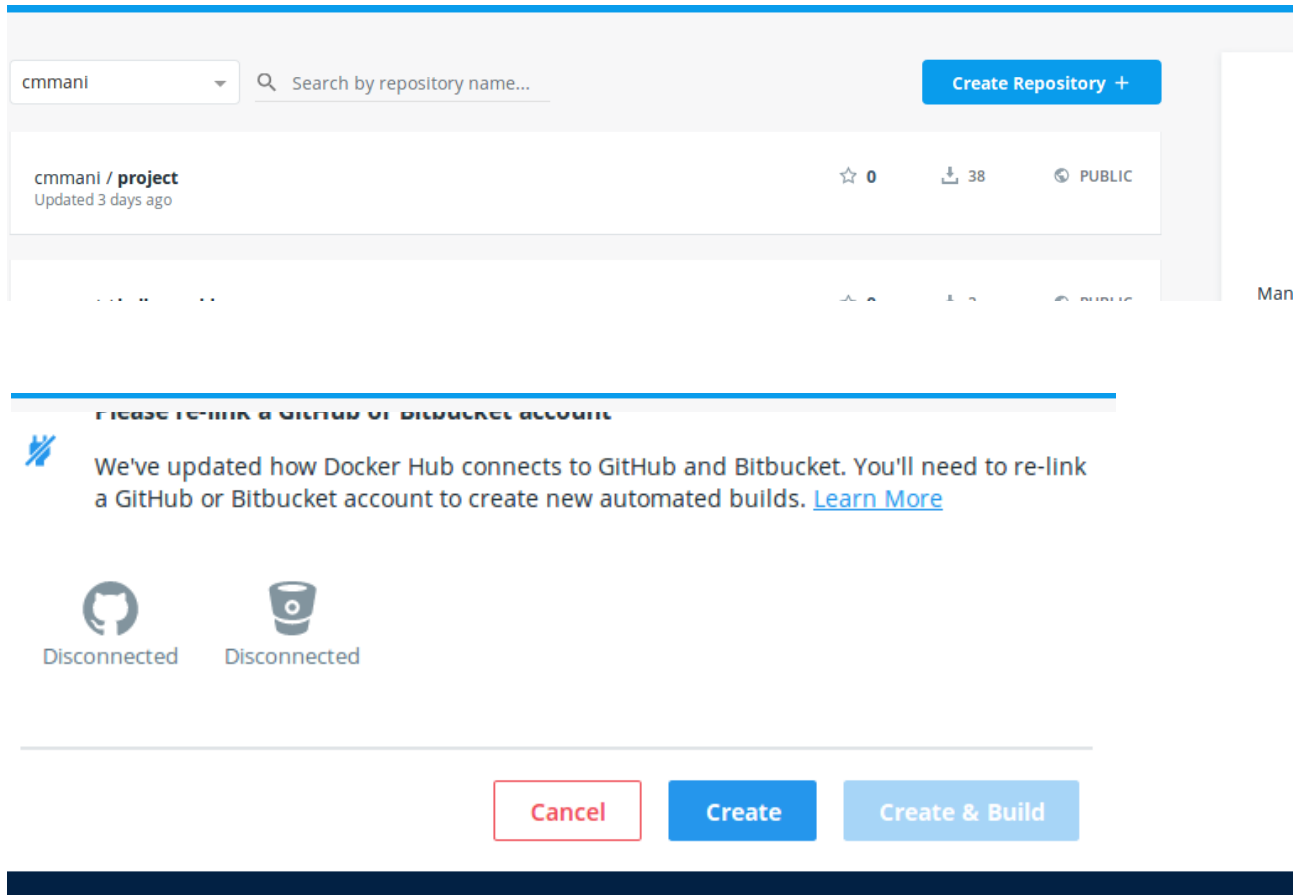
```
  version="1.0">
```

```
  <user username="admin" password="secret" roles="manager-gui"/>
```

</tomcat-users>

## Step 6:

Create docker repo in your docker hub account



\$ vi Dockerfile

FROM tomcat

COPY tomcat-users.xml /usr/local/tomcat/conf/tomcat-users.xml

COPY context.xml /usr/local/tomcat/webapps/manager/META-INF/context.xml

COPY java-sample-app-1.0.0.war /usr/local/tomcat/webapps/

```
[root@jenkins docker]# ls
context.xml  Dockerfile  java-sample-app-1.0.0.war  tomcat-users.xml
[root@jenkins docker]#
```

Then pull three file into Git in the same GIT repo

<https://github.com/cmmani/java-app>

(In this repo had that files)

## Step 7:

GO to jenkins :

Manage jenkins >>>> Manage plugin >>>

Go to availables in plugin

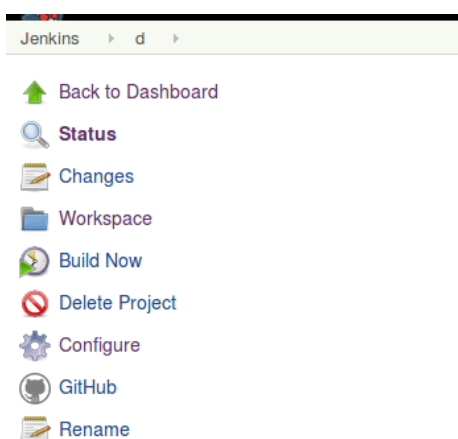
Search plugin Docker and Publish

And install the plugin **Docker and publish**

Then go to jenkinsproject >> configure

And go to build environment

The screenshot shows the 'Docker Build and Publish' configuration page in Jenkins. It includes fields for 'Repository Name' (cmmani/project), 'Tag' (\${BUILD\_NUMBER}), 'Docker Host URI', 'Server credentials' (set to - none -), 'Docker registry URL', and 'Registry credentials' (set to cmmani/\*\*\*\*\* (this is docker)). There are 'Add' buttons for credentials and an 'Advanced...' button at the bottom right. An 'Add build step' button is visible at the bottom left.

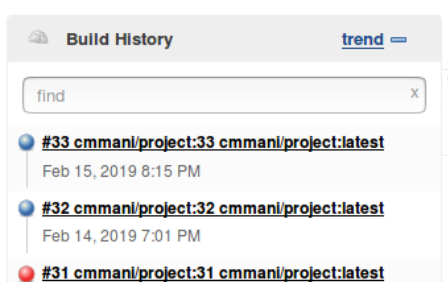


Here we build our jenkins project;

Now our file pull in to the docker hub.

Next we have to pull image run in kubernetes

My idea is to run the kubernetes by connect with SSH and run as a downstream project.



## Kubernetes work:

Requirement:

- kubernetes master
- and its node

Create Yaml file to run docker pull image.

I create

**\$ vi index.yml**

```
kind: Deployment
metadata:
  name: angular-deployment
spec:
  selector:
    matchLabels:
      app: angular
  replicas: 2
  template:
    metadata:
      labels:
        app: angular
    spec:
      containers:
        - name: angular
          image: cmmani/project:latest
          ports:
            - containerPort: 8080
---
networking/service/#defining-a-service
kind: Service
apiVersion: v1
metadata:
  name: angular-service
spec:
  selector:
    app: angular
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
      nodePort: 31001
  type: NodePort
```

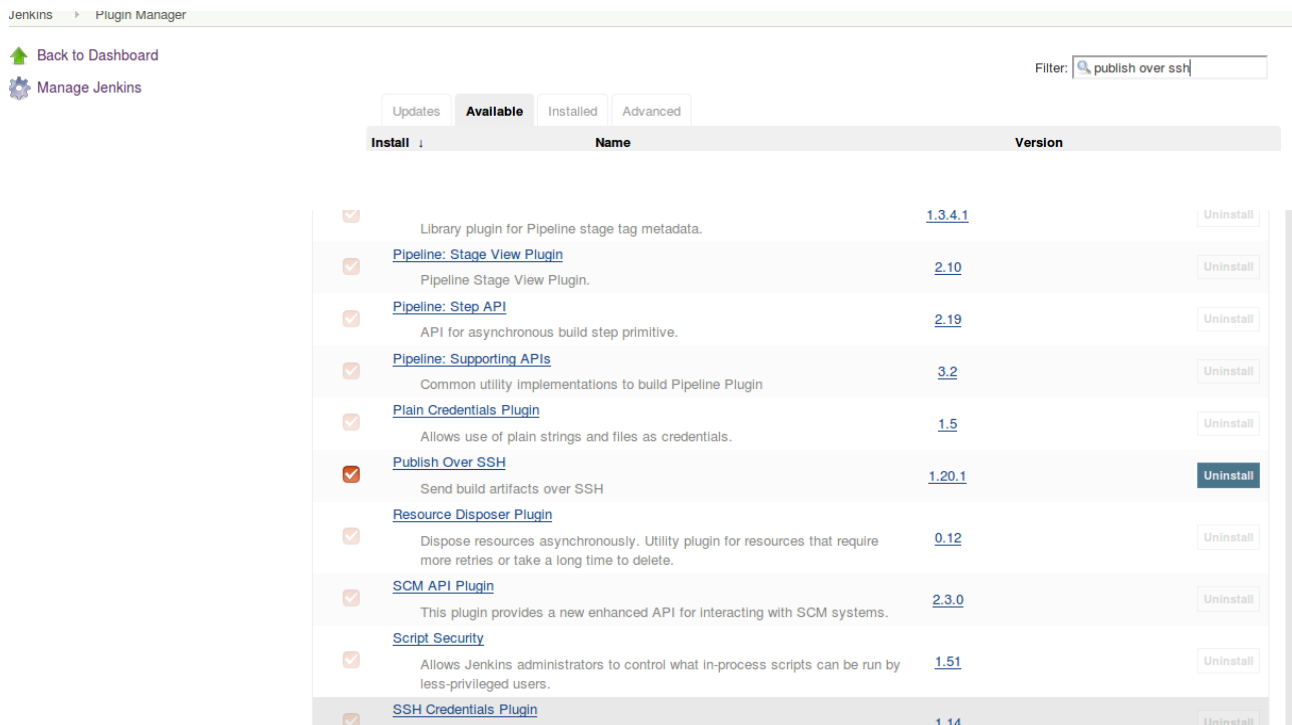
Those above file is in /root path in kubernetes master

## Step1:

Then I connect kubernetes master with ssh plugin

Managejenkins>>>manage plugins

### Publish Over SSH

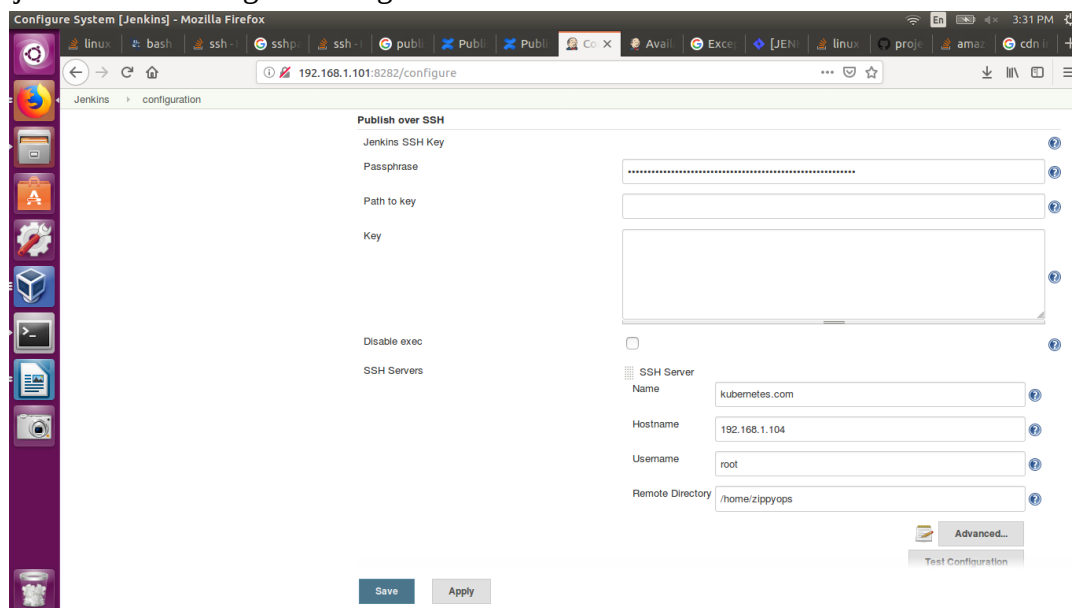


The screenshot shows the Jenkins Plugin Manager interface. The 'Available' tab is selected, and a search filter 'publish over ssh' is applied. The following table lists the available plugins:

Install	Name	Version	Action
<input checked="" type="checkbox"/>	Library plugin for Pipeline stage tag metadata.	1.3.4.1	Uninstall
<input checked="" type="checkbox"/>	<a href="#">Pipeline: Stage View Plugin</a>	2.10	Uninstall
<input checked="" type="checkbox"/>	Pipeline Stage View Plugin.		
<input checked="" type="checkbox"/>	<a href="#">Pipeline: Step API</a>	2.19	Uninstall
<input checked="" type="checkbox"/>	API for asynchronous build step primitive.		
<input checked="" type="checkbox"/>	<a href="#">Pipeline: Supporting APIs</a>	3.2	Uninstall
<input checked="" type="checkbox"/>	Common utility implementations to build Pipeline Plugin		
<input checked="" type="checkbox"/>	<a href="#">Plain Credentials Plugin</a>	1.5	Uninstall
<input checked="" type="checkbox"/>	Allows use of plain strings and files as credentials.		
<input checked="" type="checkbox"/>	<a href="#">Publish Over SSH</a>	1.20.1	Uninstall
<input checked="" type="checkbox"/>	Send build artifacts over SSH		
<input checked="" type="checkbox"/>	<a href="#">Resource Disposer Plugin</a>	0.12	Uninstall
<input checked="" type="checkbox"/>	Dispose resources asynchronously. Utility plugin for resources that require more retries or take a long time to delete.		
<input checked="" type="checkbox"/>	<a href="#">SCM API Plugin</a>	2.3.0	Uninstall
<input checked="" type="checkbox"/>	This plugin provides a new enhanced API for interacting with SCM systems.		
<input checked="" type="checkbox"/>	<a href="#">Script Security</a>	1.51	Uninstall
<input checked="" type="checkbox"/>	Allows Jenkins administrators to control what in-process scripts can be run by less-privileged users.		
<input checked="" type="checkbox"/>	<a href="#">SSH Credentials Plugin</a>	1.14	Uninstall

## Step2:

Manage jenkins >>> configure management



The screenshot shows the Jenkins 'Configure System' page for the 'Publish over SSH' plugin. The 'SSH Servers' section is expanded, showing configuration for 'kubernetes.com'.

**Publish over SSH**

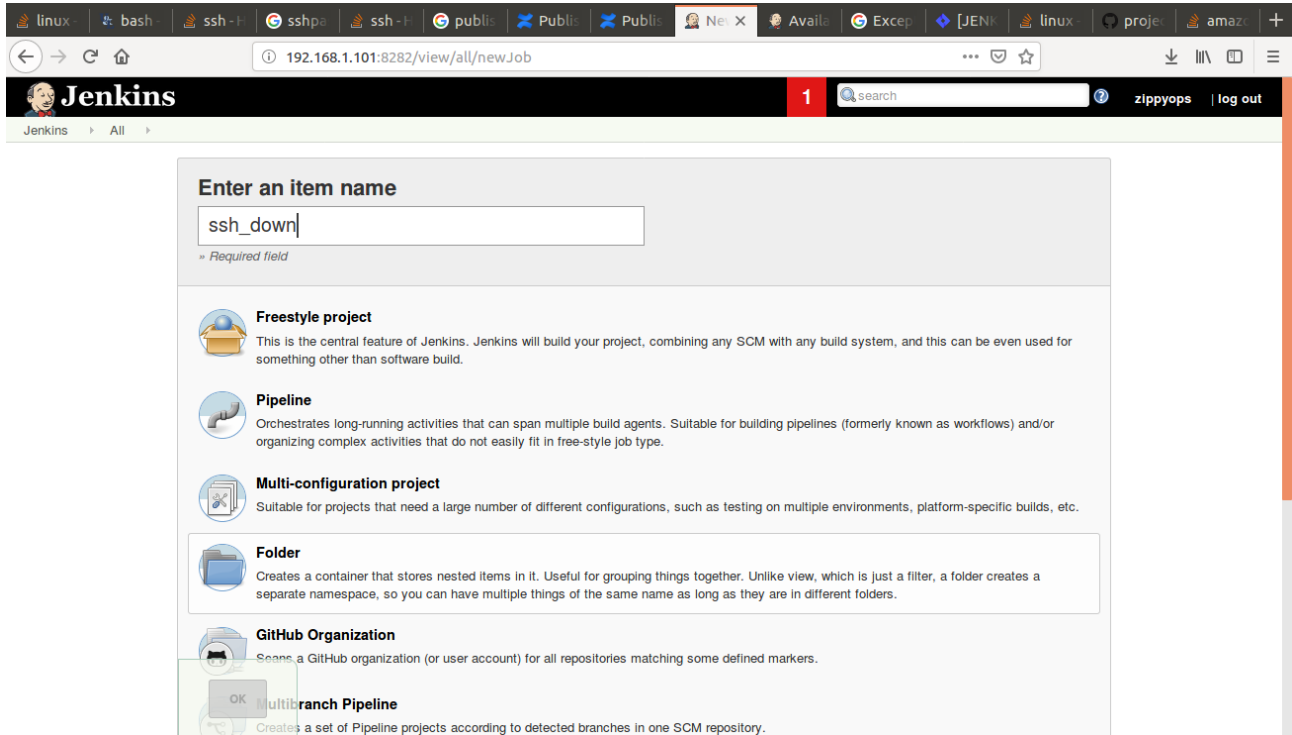
- Jenkins SSH Key
  - Passphrase: [Redacted]
  - Path to key: [Redacted]
  - Key: [Redacted]
- Disable exec: ☐
- SSH Servers
  - Name: kubernetes.com
  - Hostname: 192.168.1.104
  - Username: root
  - Remote Directory: /home/zippyops

Buttons: Save, Apply, Advanced..., Test Configuration

Create one downstream project in jenkins

Jenkins

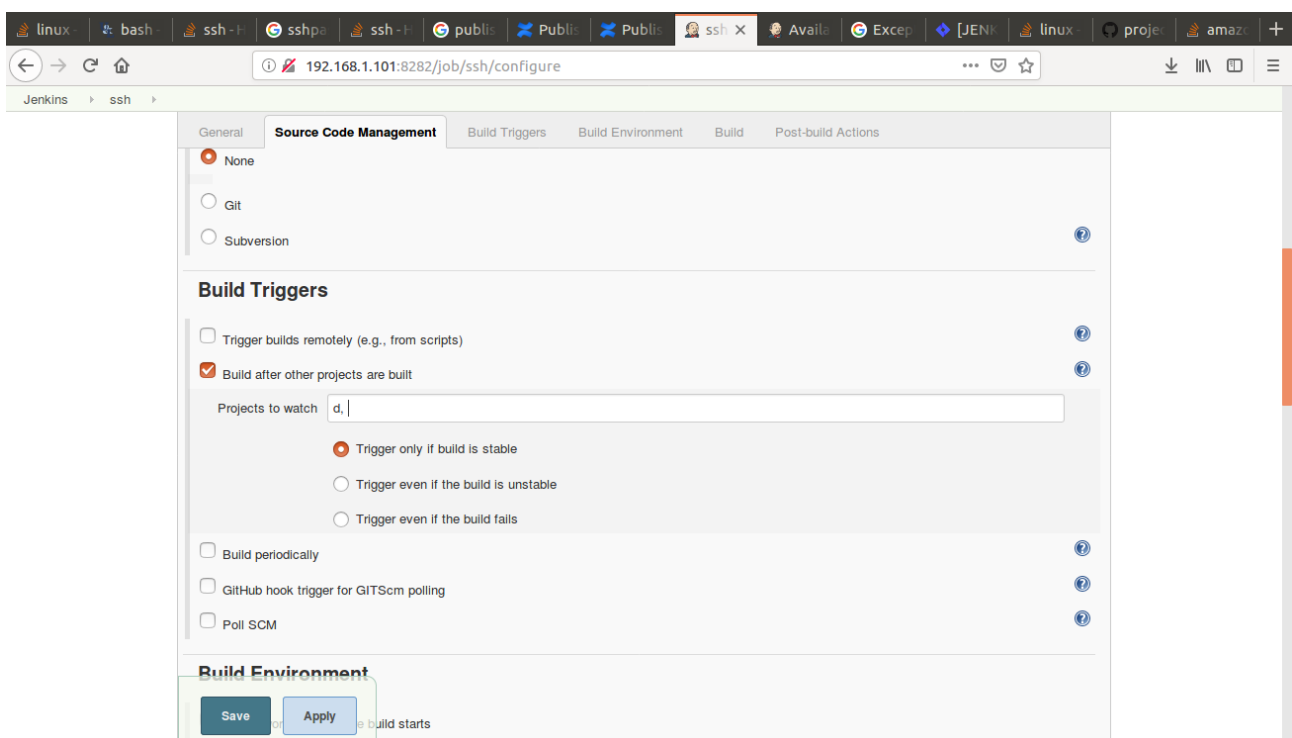
 New Item



The screenshot shows the Jenkins 'New Item' page. The browser address bar displays '192.168.1.101:8282/view/all/newJob'. The Jenkins header includes a search bar and a 'log out' link. The main content area is titled 'Enter an item name' and contains a text input field with 'ssh\_down'. Below the input field, there are several project type options: 'Freestyle project', 'Pipeline', 'Multi-configuration project', 'Folder', 'GitHub Organization', and 'Multibranch Pipeline'. Each option has a brief description. A 'Save' button is visible at the bottom left of the form.

## Step2:

And in add build triggers i specified run this project after run the old project

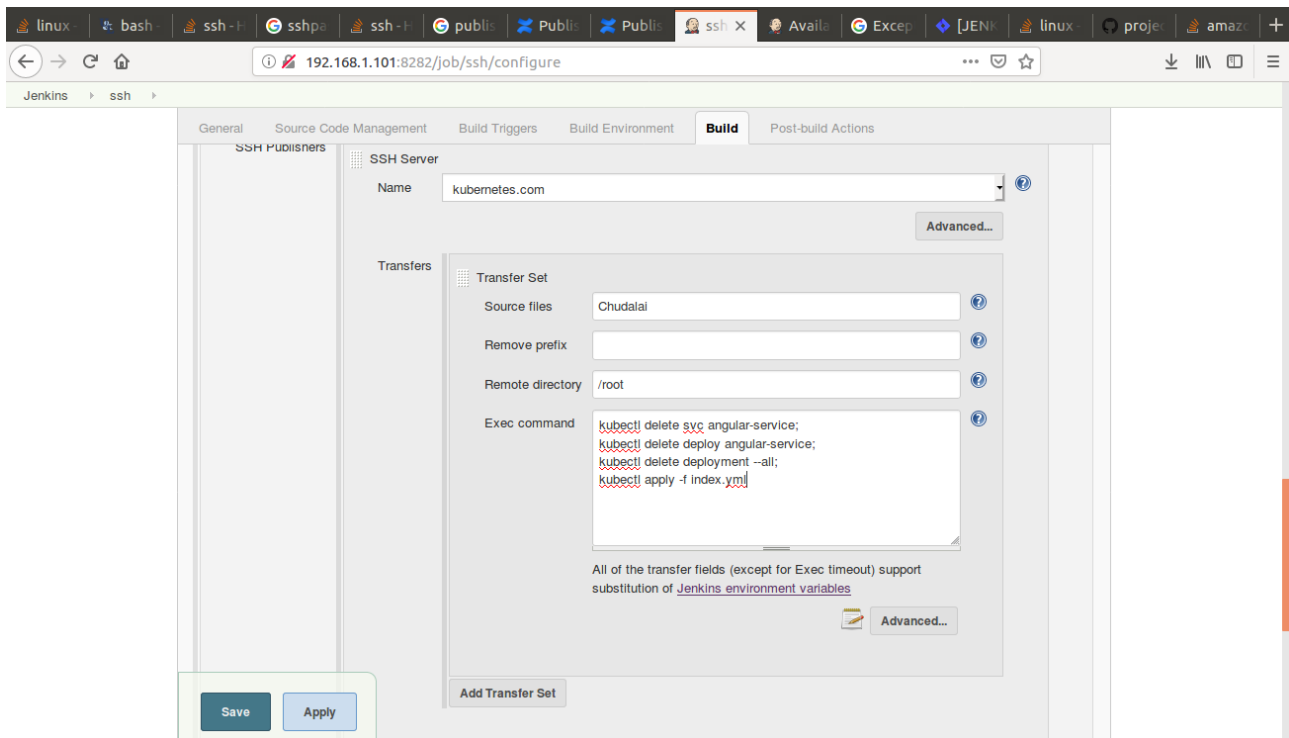


The screenshot shows the Jenkins 'Configure' page for a job named 'ssh'. The browser address bar displays '192.168.1.101:8282/job/ssh/configure'. The page has several tabs: 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The 'Build Triggers' tab is selected. Under 'Source Code Management', 'None' is selected. Under 'Build Triggers', the checkbox 'Build after other projects are built' is checked. The 'Projects to watch' field contains 'd, '. Below this, there are three radio button options: 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. At the bottom, there are 'Save' and 'Apply' buttons.



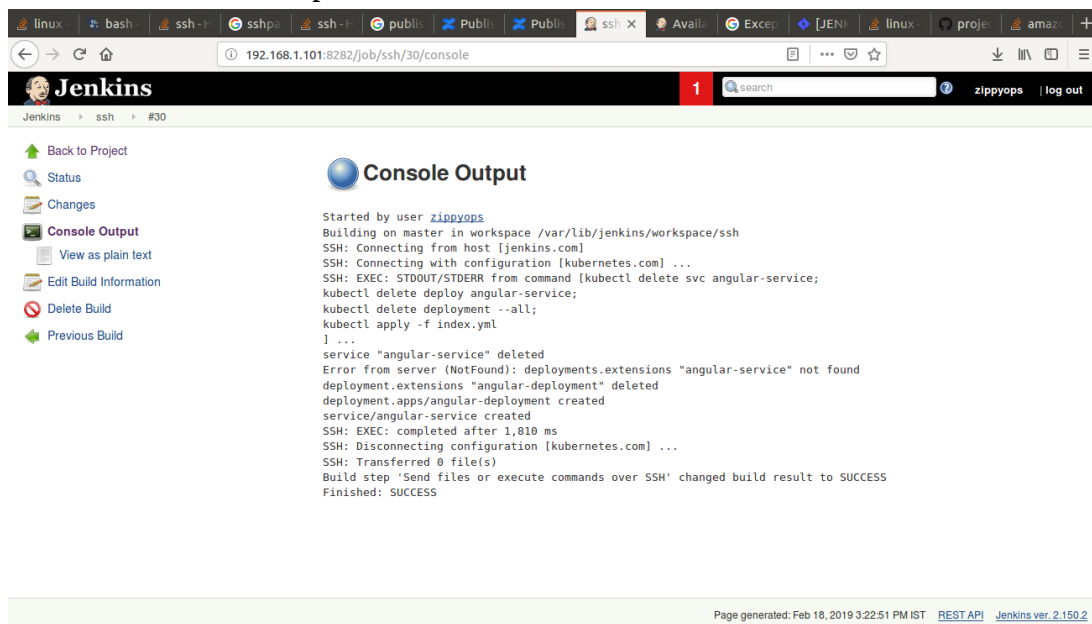
And Exec Command via jenkins

```
kubectl delete svc angular-service;  
kubectl delete deploy angular-service;  
kubectl delete deployment --all;  
kubectl apply -f index.yml
```

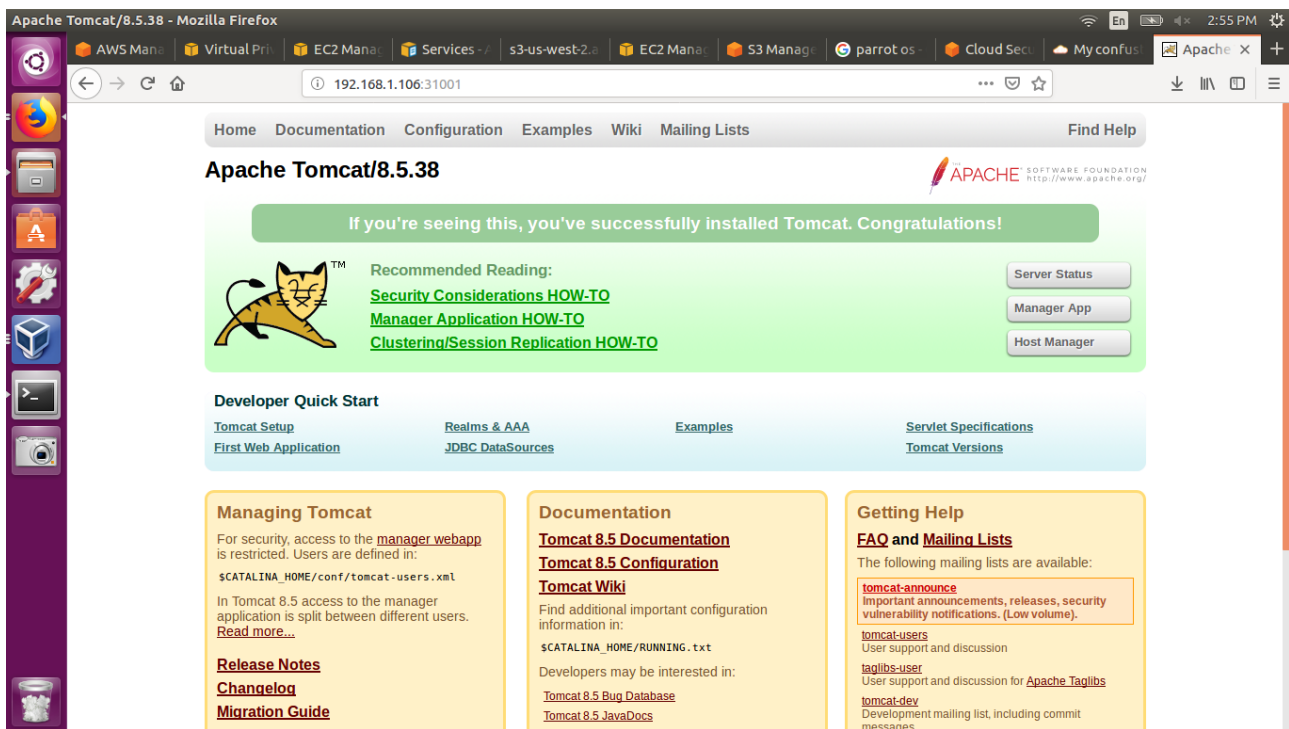


Run the above jenkins project:

You get those success build output



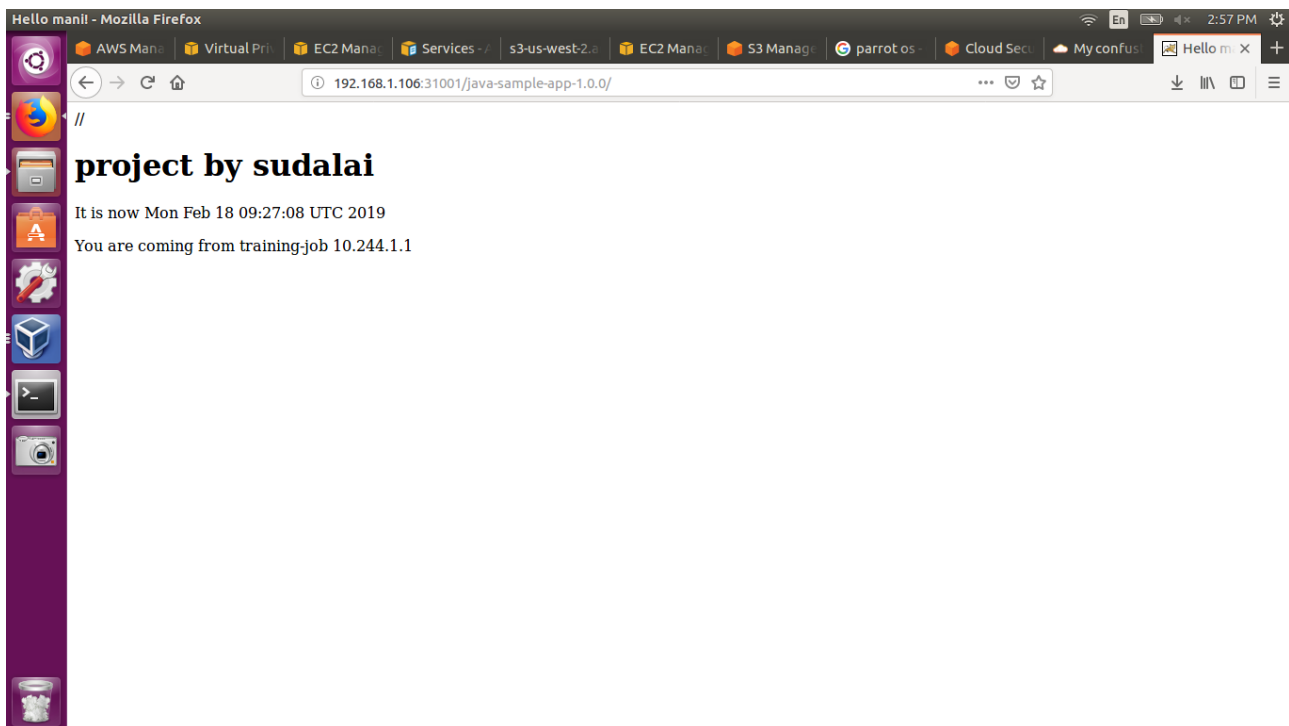
Save and Build the project in local machine by using kubernetes node IP



click manage apps:

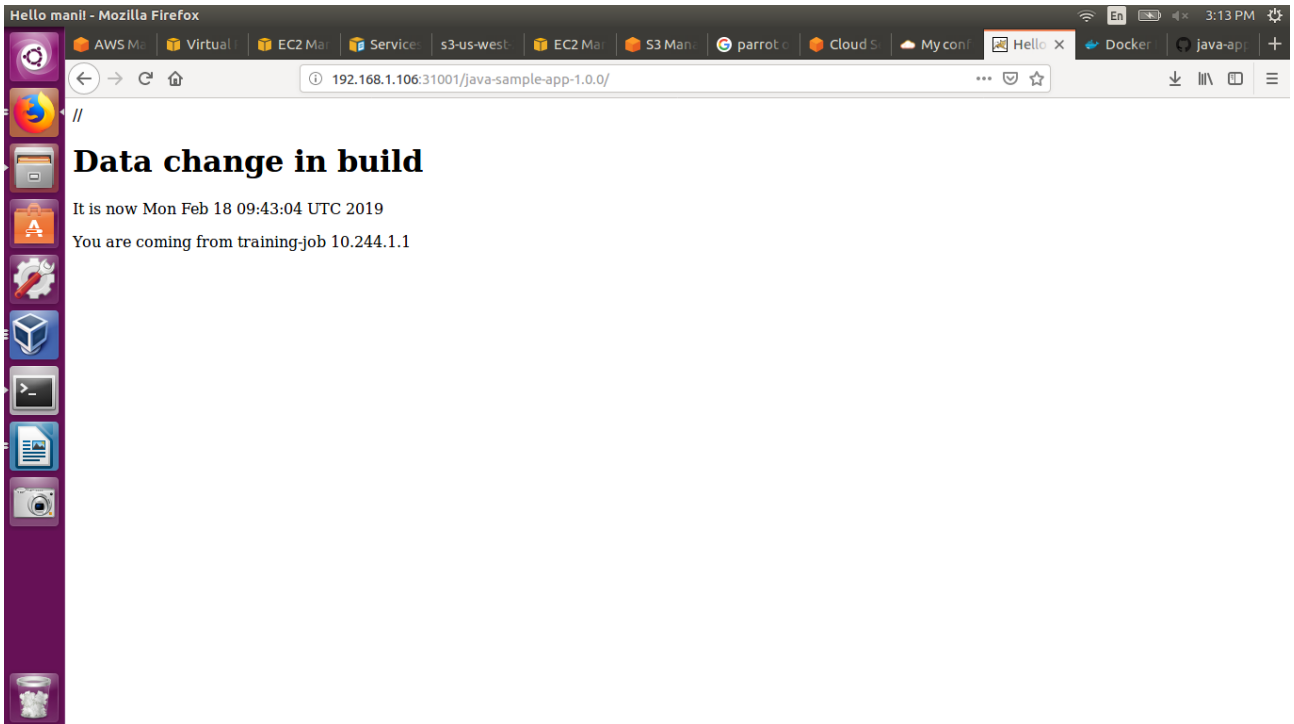
It is asked for password and username

these details are in tomcat\_user.xml file



Hereafter i changed littlebit in java application and now build the D project in one moretime.  
The output will be like

just refresh the page



Hence you got the output of the kubernetes and all files deploy automatic in every kubenodes.