Section 1- Jenkins Continuous deployment

Section 2:- Jenkins Pipelines

Section 3: AWS IAM service

Section 4: AWS cli quickstart

Section 5:- AWS lambda quickstart

Section 6- Ansible configuration management

On linux we configured jenkins

Section 1- Jenkins Continuous deployment

Jenkins is a tool that will help us through CD pipeline and all these tools it will have certain plugin

Need to install and it will immediately start working with these tool

When we install jenkins it will come up on apache tomcat server. We can access it on port 8080

Diagram

Description automatically generated

yum install java-1.8.0-openjdk

sudo amazon-linux-extras install java-openjdk11

Yum install maven

mvn archetype:generate -DgroupId=com.bharath -DartifactId=java-project -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

mvn archetype:generate -DgroupId=com.bharath -DartifactId=java-web-project -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false

**Setup jenkins**

**Objective:**

Install jenkins on linux instance that is up and running on ec2

Note:

Jenkins internally uses java to work

**Install jenkins on ec2**

**Jenkins run on port 8080 because it uses apache server internally**

Using these 2 command latest jenkins will be install

1-wget -O /etc/yum.repos.d/jenkins.repo <http://pkg.jenkins-ci.org/redhat/jenkins.repo>

2-rpm --import <https://jenkins-ci.org/redhat/jenkins-ci.org.key>

3-yum install jenkins

4-service jenkins start

Graphical user interface, application, Word

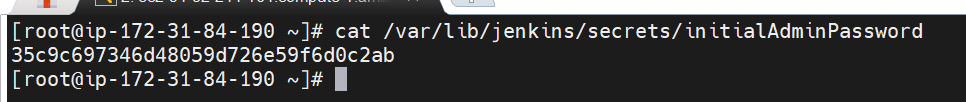
Description automatically generated

Initially when we login the default p/w is store under this location

Graphical user interface, text, application

Description automatically generated

5- cat /var/lib/jenkins/secrets/initialAdminPassword



Default p/w :- 35c9c697346d48059d726e59f6d0c2ab

Graphical user interface, diagram, text, application

Description automatically generated

Click on installed suggested plugin that will install all the plugin will be using later on.

Graphical user interface, application

Description automatically generated

Jenkins store al the working information under

yum remove jenkins

**rm -rf /var/lib/jenkins**

So next time if we remove jenkins before removing that we need to remove this folder.

Otherwise nest time when we install jenkins it will take same old setting from here(**/var/lib/jenkins)**

**Later on we are configuring java path and maven path**

**Java Path:**

/lib/jvm/java-1.8.0-openjdk-1.8.0.242.b08-0.amzn2.0.1.x86\_64

**Maven Path:**

/usr/share/maven

**Git Path:**

/usr/bin/git

Later we configure these path in jenkins so jenkins know which path we need to consider

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Jenkins Dashboard**

<http://ec2-54-82-244-164.compute-1.amazonaws.com:8080/manage>

1. We can create **new jenkins job** by clicking on **new item**
2. we can create user or developer and give them certain access
3. Build history: this will show history of all the job that is run and whether they have failed, succeeded etc
4. Manage jenkins : this is where will spend most of the time

Graphical user interface, text, application, email

Description automatically generated

Global tool configuration : where we will configure maven , java path

**sudo /usr/sbin/alternatives --config java**

Graphical user interface, text

Description automatically generated

Manage jenkins / global tool configuration/

**Configured java,maven and git**

/lib/jvm/java-1.8.0-openjdk-1.8.0.312.b07-1.amzn2.0.2.x86\_64

/usr/bin/git

/usr/share/maven

Managed jenkins / configured system

Graphical user interface, text, application, email

Description automatically generated

/var/lib/jenkins :--- this is where jenkins store all the info when we running the job

Graphical user interface, application

Description automatically generated

**Under build executor status : we can check build is success or not**

**Configure root user permissions**

Before create new job we and execute it we have to make sure jenkins have all required permission to run the job

**Assign Root User and Permissions:**

**Step 1** vi /etc/sysconfig/jenkins ---- run in ec2 and change jenkins user as root user

Text

Description automatically generated

**Step 2** We need to change the ownership of certain folder

Root user will have permission on those folder

sudo chown -R root:root /var/lib/jenkins

sudo chown -R root:root /var/cache/jenkins

sudo chown -R root:root /var/log/jenkins

vim /etc/sysconfig/jenkins   
# make sure that $JENKINS\_USER="jenkins"

chown -R jenkins:jenkins /var/lib/jenkins  
chown -R jenkins:jenkins /var/cache/jenkins  
chown -R jenkins:jenkins /var/log/jenkins  
systemctl restart jenkins

<https://stackoverflow.com/questions/11880070/how-to-run-a-script-as-root-in-jenkins>

Graphical user interface, text, application

Description automatically generated

**First Job**

**Will create a shell script which will execute through jenkins job**

Text

Description automatically generated

Click new item

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

First job is saved

Graphical user interface, application

Description automatically generated

Click on workspace , no workspace , run a build jenkins will create workspace

Graphical user interface, text, application, chat or text message

Description automatically generated

**Build maven project from jenkins**

Text

Description automatically generated

**Click on new item**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Work with git repo**

**New item/git/**

Text

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

**Build Triggers**

These jenkins job executed when we hit build now button but in real time environment all these to be automated.

That is where build trigger will be very helpful

Click on configure

Graphical user interface, application

Description automatically generated

Graphical user interface, text

Description automatically generated with medium confidence

Build periodically : like cron job

Graphical user interface, text, application, Teams

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Poll continuously :--- it could be github**

**Configure security**

**Objective :**

How to create more user and how to grant permission to them

**First way**

Go to manage jenkins - user- create user

Graphical user interface, application

Description automatically generated

**Second way**

Manage jenkins - configure global security

Graphical user interface, text, application, email

Description automatically generated

''

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Give permission to bob from admin console

Graphical user interface

Description automatically generated

Now login as bob and we see for bob only limited things

A picture containing shape

Description automatically generated

**Understanding delpoyment**

Diagram

Description automatically generated

**Install tomcat on ec2 machine, when we click on build, it will build project and deploy war in tomcat.**

**Tomcat installation and configuration**

**OBjective:**

**Create a new instance and give name tomcat**

**Automate build and deployment process**

Tomcat Installation and Integration:

yum install tomcat

yum install tomcat-webapps tomcat-admin-webapps

Service tomcat start

Graphical user interface, text, application, website

Description automatically generated

**This is coming because we have also installed webapps**

In new aws instance we installed apache tomcat

**Configure User**

Graphical user interface, text, application, website

Description automatically generated

**We need to click manage app if we want to add user**

vi /usr/share/tomcat/conf/tomcat-users.xml

Uncomment Admin roles and user and Add:

<user username="deployer" password="deployer" roles="manager-script" />

Text

Description automatically generated

Service tomcat restart

Graphical user interface, application, table, Excel

Description automatically generated

Now we have access to complete administration page

**When we authenticate deployer will be using in jenkins**

Deploy war/ear to a container:

\*\*/java-web-project.war

**Install Deployment Plugin**

**Before that Jenkins will build and deploy war to tomcat need to install plugin called deploy to container**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

**Continuous integration in action**

**Objective :**

Create a new jenkin job that will pull maven project from the git repo it will build it and also it will push to tomcat

Instance running separately

At the end we will see java webapp installed here

Step 1

Configure github

Graphical user interface, application

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated with low confidence

Table

Description automatically generated

**Click on java web-app**

Graphical user interface, text, application, Word

Description automatically generated

**Webhooks Introduction**

So far we did build manually

We need something on github that will jenkins and update has happen (push has happen)

For that github provide concept or tool called webhooks

Graphical user interface, text, application, email

Description automatically generated

**Github webhooks in action**

Every repository has it own web hook.

**Step 1**

Graphical user interface, text, application, email

Description automatically generated

**Step 2 in jenkins**

**Now we have webhook**

Graphical user interface, application

Description automatically generated

**Step 3 code commit**

Graphical user interface, text, application, chat or text message

Description automatically generated

**As soon as we commit build will trigger**

Using hook jenkins will came to know commit happens and it will start build , pull the code from github

**Master and Agent**

**Objective :**

Jenkins master slave architecture

So far we install jenkins in single aws instance and create job and it works great but

What if load increases and we want to run job in particular os that is where jenkins master/slave come in picture.

We can have any number of slave and one master where jenkins is installed and this guy will push the job on those

Respective machine

We can install one more ec2 instance and we can manually tell jenkins run job in that particular instance

Create a ec2 agent instance

Configure ssh communication

Configure a node/agent on master.

If there is too much job in master automatically it will push job to one of the agent

Diagram

Description automatically generated

**Generate the key on master**

A picture containing map

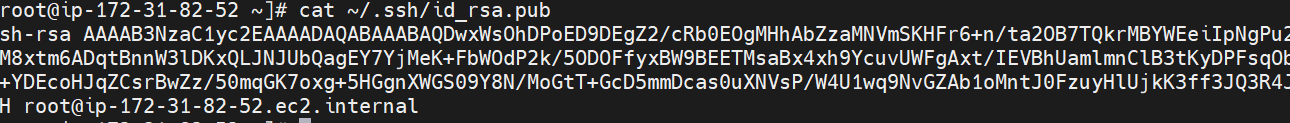
Description automatically generated

**Use linux box**

**Ssh-keygen**

Text

Description automatically generated



**Private key will use in jenkins while configuring the agent and public key will go to authorization of node.**

**Launch the agent instance**

**Objective :**

Will create a agent launching a new aws instance where we will taking the public key and configuring them in the agent

And also will install java

**Note :**

Whatever process we will follow here, we can process with same no of step as many agent we want to create

**Jenking will spread the load or job across these agent**

Graphical user interface

Description automatically generated with medium confidence

This is where public key live to communicate with other machine

Text

Description automatically generated

**First step on agent**

Paste the public key in authorized which we generated in linux(master) and it can talk with slave

Graphical user interface, text, application

Description automatically generated

**Step 2**

**Install java on this machine because** for jenkins to be installed it need java

Yum install java

In this lecture we launch aws instance mark as jenkins agent grab the public key that we have generated in master and copied to the

Authorized keys. So that master and slave can communicate with each other

**Create a node using the agent**

**Objective :**

To tell master node or jenkins about the slave node or agent that we have created

Click on manage jenkins - manage node and cloud

Graphical user interface

Description automatically generated with low confidence

Right now there is only one node

**To add a agent node click on new node left**

Text, application

Description automatically generated with medium confidence

Section 2 :**Jenkins Pipelines**

**A pipeline is combination of multiple stages**

Timeline

Description automatically generated

**Jenknsfile :**

Within the stage multiple step is there to perform

**First pipeline (testing default pipeline)**

Graphical user interface, text, application, email

Description automatically generated

**Click on pipeline and choose hello world**

Graphical user interface, text, application, Teams

Description automatically generated

Pipeline is always the root element

**Agent any** : this particular job can run on any agent

The child element of pipeline is **stages** and we can have multiple **stage and stage** can have multiple **steps**

**Save it** and click on build now

Pipeline output

Graphical user interface, text

Description automatically generated with medium confidence

Diagram

Description automatically generated

**Multi stage pipeline**

Graphical user interface

Description automatically generated with medium confidence

**Output :**

Graphical user interface, application

Description automatically generated

**Reading environment variable**

We can access jenkins environment variable from within the job very easily.

Jenkins by default exposes all the env variable under this html

Graphical user interface, text, application, email

Description automatically generated

**All these variables we can get within the jenkins**

Graphical user interface, application

Description automatically generated

**Output**

Graphical user interface, application

Description automatically generated

**Pass parameter to the pipeline**

**Objective :**

How to pass parameter to the pipeline

Graphical user interface, application, Teams

Description automatically generated

If we checked this box value will be true ( set by default)

**Now let see how to get parameter value in pipeline.**

Graphical user interface, text, application

Description automatically generated

After giving parameter value will get **build with paramter**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, application

Description automatically generated

**Requesting User Input**

**Objective :**

If we have to do manual approval then we can ask for input

Graphical user interface, application

Description automatically generated

Once it reaches approve deployment stage it will ask

Graphical user interface, application, Word

Description automatically generated

Click on proceed.

Graphical user interface, application

Description automatically generated

**Create a java project pipeline**

**Objective :**

Create pipeline for java maven project

Earlier we trigger java maven project using webhooks now will create a pipeline and within the pipeline building and testing

Of project will happen.

Practical

Step1

Within jenkins create new step :---javaprojectpipeline

Go to build trigger because we want to be trigger from githook which will be fixing in next lecture and check box

**Github hook trigger for gitscm polling**

Graphical user interface, application

Description automatically generated

**Write pipeline and move into github repository.**

Very simple pipeline two stages

Graphical user interface, text, application

Description automatically generated

A picture containing timeline

Description automatically generated

**Imp**

Pipeline code should go into jenkinsfile under project directly

**Test Pipeline**

**Added Jenkinsfile under project**

Graphical user interface, text, application, email

Description automatically generated

**Before configuring the webhook build once**

Graphical user interface, application

Description automatically generated

Due to scm it will pull code from github build it and test it

We manually run it now lets go to github

Note : before configure webhook build once manually

**Webhook added :** if we do any commit pipeline will trigger automatically

Graphical user interface, text, application, email

Description automatically generated

Do changes in file so that webhook triggered pipeline in jenkins

Graphical user interface, application

Description automatically generated

**Create a java web project pipeline**

**Objective**

Build and deploy java web project through pipeline

Create a new pipeline :- javawebpipeline

Graphical user interface, text, application, email

Description automatically generated

Go trigger and check webhook

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, application

Description automatically generated

Click on pipeline syntax to create code for deploy

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Copy and paste in github

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Test Pipeline**

**Start server in ec2 machine :-**

**service tomcat start**

**Before webhook :**

**First build it :** it will build it and deploy to tomcat

Graphical user interface, application

Description automatically generated

Table

Description automatically generated

**Now configure github webhook**

**In webhook :** will put jenkins url which is configure from ec2 machine

Graphical user interface, text, application, email

Description automatically generated

Commit again then pipeline will trigger in github:--

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

Section 3 AWS: IAM service

Once we login as root user we create any number of IAM user and assign roles to user based on that user able to see on console.

We should create IAM user. It will restrict access

**We can create user , group of user, roles to user and group of user and policies ( based on policy user able to access)**

**Creating a IAM user**

**Objective:**

Login aws dashboard but only access to ec2

**IAM:-- manage access to aws resource**

Graphical user interface, text, application, email

Description automatically generated

<https://861782232355.signin.aws.amazon.com/console>

Graphical user interface, application

Description automatically generated

**Sknk@2000**

**Create custom policy**

**Note :**

We can create our own policy

**Go to policies :--- create policy**

Graphical user interface, text, application, email

Description automatically generated

Policy will take effect when request condition met

Create policy and we can attach to user (policy is like more specific)

Graphical user interface, text, application, email

Description automatically generated

**Create and assign role**

Will assign ec2 instance a certain role and that role has certain permission so anybody use instance

will use that role.

**Role can be apply to any service and whater role has permission only that can be apply**

**We gave ec2 :--- s3 full access and rds full access ---create role**

**AWS CLI Quickstart**

We can perform multiple thing on aws service from cli

Diagram

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

**Create Programmatic user**

**Create user and get access key and secret key**

**User : awscliuser**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Connect with ec2 machine :--- yum install awscli**

Text

Description automatically generated

**Successfully configure aws-ec2**

**Run Command**

* Aws help
* Aws service name help (aws ec2 help)

Text

Description automatically generated

**Aws Lambda Quickstart**

A lambda function brings in custom code into the serverless platform where there are other various component can be used.

These component trigger lambda function and when lambda function trigger the handler function which we write will be

Executed

It allow us to create our own custom code that will be trigger when even happens in these other component and in response

It can produce o/p in the process of producing o/p it can handle the event with other component.

**Create and test a lambda.**

Every lambda function need IAM role which will give basic permission.

Basic lambda function allows to write a cloudwatch log

A lambda function will be triggered by event that is happening in other component.

These other component can be s2 ,api gateway and sns.

It will get event inside this will return response.

Section 6- Ansible configuration management

Ansible is provisioning or configuration management tool using it we can push across multiple nodes.