Jenkins master slave configuration

PREPARED BY,
T.AJAY
T.SUSMITH

Contents:-

Method-1

- In this method create one master instance and five slave instances.
- In master generate one key by using command
  - ssh-keygen –t rsa
- Copy the public key id_rsa.pub.
- Paste this key in all slave instaneces using below commands
  - cd .ssh/
  - vi authorised_keys
  - Press (shift+A) paste the key and press esc and type :wq! And enter
- Now install java on all instances.
- Install Jenkins on master node and browse the public ip along with port number 8080.

Creating nodes in Jenkins:

- Goto manage Jenkins➔nodes➔give name and select perminent node→click on create node
  - No of executors=10
  - Remote root directory:/home/ec2-user
  - Labels: slave1(provide label name)
  - Usage: use this node as much as possible
  - Launch method :launch agent via ssh
  - Host: provide ipv4 private ip of your instance
- In credentials you nee to create your credentials click on add select Jenkins kind→ssh username and private key, username➔ec2-user, key→copy your pem file and paste here. click on create and select that created credentials.
  - Host key verification strategy : manually trusted key verification strategy
  - Click on save
  - Now your node created successfully and your agent connected and online.
  - Follow the above process for all slaves create nodes in every slave.

- While implementing jobs you need to select restrict where the project run like slave1,2 etc.
- Install nginx on slave1
- Install httpd on slave2
- Install sonar qube on slave3
- Install tomcat on slave4
- After installing servers browse the public along with port numbers.

Method-2

- Step-1: Clone, build and deploy any python application.
  - Create one job and choose free style project.
  - Select restrict where this project can be run and give label expression.
  - Choose the source code management as git and provide repository url and the branch name.
- Add the build step as execute shell and add the commands to run the python application
- After build get success browse the public ip along with port number.
- Step-2: Clone, build and deploy any python application using bash script.
  - Create one job and choose free style project.
  - Select restrict where this project can be run and give label expression.
  - Choose the source code management as git and provide repository url and the branch name.
- select build step and choose execute shell and add commands to execute the bash script.
- After build get success browse the public ip along with port number.

- Step-3: Clone, build and deploy any python application using terraform scripts.
  - Create one repository in github and add all terraform scripts to run python application with .tf extension.
  - Create one job and choose free style project.
  - Select restrict where this project can be run and give label expression.
  - Choose the source code management as git and provide repository url and the branch name.
- select build step and choose execute shell and add commands to execute the bash script.
- After build get success browse the public ip along with port number.
- Step-4: Clone, build and deploy any java application .
- in this step we are going to deploy java application in the tomcat server. Previously we are installed tomcat server in that server we are going to deploy our java application
- Click on manager app you will get an error it specifies that you are not authorized to view this page.so in order to view the page we need to modify some permissions.
- first you connect to the instance via shell and go in to that apache tomcat server .
- modify the following files . by commenting the valve class name .
- in the tomcat-users.xml add the user name and password .
- Now browse the public ip along with port number 8080.
- Enter your username and password.

- Now your successfully enter into the manager App console.

- Create one job and choose free style project.
- Select restrict where this project can be run and give label expression.
- Choose the source code management as git and provide repository url and the branch name.
- We need to convert our pom.xml into war or jar file using build tool.
- Now choose the build step as invoke top level maven targets select maven version and choose the goal as clean and package .
- Select one more build step as execute shell and add commands to place the .WAR folder into the webapps directory.

- Refresh the page now you can see webaapp-2 folder is added into the tomcat server .
- Click on webapp-2 then you can see your java application.

- Step-5: Clone, build and deploy any java application through bash script.
    - Create one repository in github and write the bash script to run the java application.
    - Create one job and choose free style project.
    - Select restrict where this project can be run and give label expression.
    - Choose the source code management as git and provide repository url and the branch name.
    - select build step and choose execute shell and add commands to execute the bashscripts.
    - save the job and click on build now then your job created successfully.
    - browse the public ip along with the port number 8080 .
    - click on 01-maven-webapp then your page will appear.

Method-03

- Creating master-slave configuration AJAY creates one master and Susmith creates five slave nodes and Ajay will build the jobs in all slave nodes in same AWS account.
  - In this method Susmith create one master instance and Ajay Creates five slave instances.
  - In master generate one key by using command
    - ssh-keygen –t rsa
  - Copy the public key id_rsa.pub.
  - Paste this key in all slave instaneces using below commands
    - cd .ssh/
    - vi authorised_keys
    - Press (shift+A) paste the key and press esc and type :wq! And enter
  - Now install java on all instances.
  - Install Jenkins on master node and browse the public ip along with port number 8080.

- After installing Jenkins create nodes for all slaves like we are done in method-1
- After creating nodes we install different servers in slave instances.
- Install nginx on slave1
- Install httpd on slave2
- Install sonar qube on slave3
- Install tomcat on slave4
- After installing servers browse the public along with port numbers

Method -4

- Creating master-slave configuration AJAY creates one master and Susmith creates five slave nodes and Ajay build the jobs in all slaves using different AWS accounts.
  - In this method create AJAY one master instance and SUSMITH creates five slave instances.
  - In master generate one key by using command
    - ssh-keygen –t rsa
  - Copy the public key id_rsa.pub.
  - Paste this key in all slave instaneces using below commands
    - cd .ssh/
    - vi authorised_keys
    - Press (shift+A) paste the key and press esc and type :wq! And enter
  - Now install java on all instances.
  - Install Jenkins on master node and browse the public ip along with port number 8080.

- After installing Jenkins create nodes for all slaves like we are done in method-1
- After creating nodes we install different servers in slave instances.
- Install nginx on slave1
- Install httpd on slave2
- Install sonar qube on slave3
- Install tomcat on slave4
- After installing servers browse the public along with port numbers

спасибо
谢谢
GRACIAS
THANK YOU
ありがとうございました
MERCI
DANKE
धन्यवाद
شُكراً
OBRIGADO