**Tools installation Guide on UBUNTU**

Java JDK:

sudo apt update

sudo apt install openjdk-11-jdk

java –version

GIT and MAVEN:

sudo apt-get install -y git maven

Git : git --version

Maven : mvn –version

git config --global user.name "sacsarkh"

git config --global user.email [sachindeshmukh486@gmail.com](mailto:sachindeshmukh486@gmail.com)

git remote add origin <https://github.com/sacsarkh/sachi-hello-world.git>

git push -u origin –all or specific branch

Jenkins:

Open Jenkins website (<https://jenkins.io/download/>)

Go to Long Term Support

Select Generic Java Package (.war)

wget <https://get.jenkins.io/war-stable/2.277.2/jenkins.war>

nohup java -jar jenkins.war &

**Master and slave configuration:**

Download slave.jar in slave machineSSS

sudo wget <http://172.31.41.7:8080/jnlpJars/slave.jar>

ls –l

sudo chmod u+x slave.jar

TOMCAT 9:

sudo apt-get update

sudo apt-get install -y tomcat9

sudo apt-get install -y tomcat9-admin

**setting up path and adding user**

cd /etc/tomcat9/

ls

You will find the file tomcat-users.xml

Open the file -- sudo vim tomcat-users.xml

In the end we need to add one statement

<user username="training" password="sunilsunil"

roles="manager-script,manager-status,manager-gui"/>

save and quit

type :wq

sudo service tomcat9 restart

**Password less communication:**

sudo passwd Ubuntu

enter password

cd /etc/ssh

ls

sudo vim sshd\_config

Go to insert mode )

change password authentication to yes

Save and quit :wq

sudo service ssh restart

Main machine type = ssh-keygen (to generate the keys)

ssh-copy-id ubuntu@172.31.1.107

ssh [ubuntu@172.31.1.107](mailto:ubuntu@172.31.1.107)

Docker :

sudo su –

go to - get.docker.com

curl -fsSL https://get.docker.com -o get-docker.sh

sh get-docker.sh

docker –version

Ansible :

Ubuntu machines default come with Python3

sudo passwd Ubuntu

( lets give the password as ubuntu only )

$ sudo vim /etc/ssh/sshd\_config

change PasswordAuthentication yes

Save and QUIT

$ sudo service ssh restart

$ exit

sudo apt-get install software-properties-common

sudo apt-add-repository ppa:ansible/ansible

sudo apt-get update

sudo apt-get install -y ansible

ansible –version

SONAR Qube:

We can install help from google either wise install Docker on machine and then run sonarqube container.

----------------------------------------------------------------------------------------------------------------------------

Use apt-get to install the required packages.

* apt-get update
* apt-get install unzip software-properties-common wget default-jdk

Install the PostgreSQL database service.

* apt-get install postgresql postgresql-contrib

Access the Postgres database service command-line.

* su - postgres
* psql

Create a Postgres user named sonarqube,Create a Postgres database named sonarqube. Give the PostgreSQL user named sonarqube permission over the database named sonarqube

* CREATE USER sonarqube WITH PASSWORD 'password';
* CREATE DATABASE sonarqube OWNER sonarqube;
* GRANT ALL PRIVILEGES ON DATABASE sonarqube TO sonarqube;
* \q

Download the Sonarqube package and move it to the OPT directory.

* mkdir /downloads/sonarqube -p
* cd /downloads/sonarqube
* wget <https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-7.9.1.zip>
* unzip sonarqube-7.9.1.zip
* mv sonarqube-7.9.1 /opt/sonarqube

Create a new Linux account named sonarqube, Set the correct file permission on the sonarqube directory.

* adduser --system --no-create-home --group --disabled-login sonarqube
* chown -R sonarqube:sonarqube /opt/sonarqube

Edit the sonar.sh configuration file.

* vi /opt/sonarqube/bin/linux-x86-64/sonar.sh

Configure the following options:

* RUN\_AS\_USER=sonarqube

Edit the sonar.properties configuration file.

* vi /opt/sonarqube/conf/sonar.properties

Configure the following options:

sonar.jdbc.username=sonarqube

sonar.jdbc.password=password

sonar.jdbc.url=jdbc:postgresql://localhost/sonarqube

sonar.web.javaAdditionalOpts=-server

sonar.web.host=0.0.0.0

Create a Linux configuration file named 99-sonarqube.conf

* vi /etc/security/limits.d/99-sonarqube.conf

Here is the content of the 99-sonarqube.conf file.

sonarqube - nofile 65536

sonarqube - nproc 4096

Edit the sysctl.conf configuration file.

* vi /etc/sysctl.conf

Add the following lines at the end of the sysctl.conf file.

vm.max\_map\_count=262144

fs.file-max=65536

Reboot your computer to enable the new configuration

reboot

Start the Sonarqube service.

* /opt/sonarqube/bin/linux-x86-64/sonar.sh start

Use the following command to monitor the SonarQube log.

* tail -f /opt/sonarqube/logs/sonar.log

for more deatils refer - <https://techexpert.tips/sonarqube/sonarqube-installation-ubuntu-linux/>

Jenkins pipeline integration for sonarqube:-

Install pipeline utility

Sonarqube all plugins

Add token to Jenkins machine

Check webhook config with Jenkins

**Best way to install sonarquebe is below:**

apt install unzip

adduser sonarqube

wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.4.0.54424.zip

unzip \*

chmod -R 755 /home/sonarqube/sonarqube-9.4.0.54424

chown -R sonarqube:sonarqube /home/sonarqube/sonarqube-9.4.0.54424

cd sonarqube-9.4.0.54424/bin/linux-x86-64/

./sonar.sh start

Nexus Repo:

using docker

docker run -d -p 8081:8081 --name nexus sonatype/nexus3

-------------------------------------------------------------------

* apt-get install wget ( install if you dont have wget )
* java -version ( make sure java is installed which should be java 8 or higher version )
* wget <https://download.sonatype.com/nexus/3/latest-unix.tar.gz>
* tar -xvf latest-unix.tar.gz
* cd nexus-3.35.0-02/bin
* ./nexus start ( starts the nexus artifactory )
* ./nexus status ( by this you check the status of nexus artifactory )
* To access this use http://ip\_Address:8081 ( by deafault which will be running on 8081)

intial password will be present in /opt/sonatype-work/nexus3/admin.password

(https://www.howtoforge.com/how-to-install-and-configure-nexus-repository-manager-on-ubuntu-20-04/)

Setup Kubernetes on Amazon EKS

(https://github.com/yankils/Simple-DevOps-Project/blob/master/Kubernetes/kubernetes\_setup\_using\_eksctl.md)

You can follow same procedure in the official AWS document [Getting started with Amazon EKS – eksctl](https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html)

#### Pre-requisites:

* an EC2 Instance
* Install AWSCLI latest verison

1. Setup kubectl  
   a. Download kubectl version 1.21  
   b. Grant execution permissions to kubectl executable  
   c. Move kubectl onto /usr/local/bin  
   d. Test that your kubectl installation was successful
2. curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.21.2/2021-07-05/bin/linux/amd64/kubectl
3. chmod +x ./kubectl
4. mv ./kubectl /usr/local/bin

kubectl version --short --client

1. Setup eksctl  
   a. Download and extract the latest release  
   b. Move the extracted binary to /usr/local/bin  
   c. Test that your eksclt installation was successful
2. curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp
3. sudo mv /tmp/eksctl /usr/local/bin

eksctl version

1. Create an IAM Role and attache it to EC2 instance  
   Note: create IAM user with programmatic access if your bootstrap system is outside of AWS  
   IAM user should have access to  
   IAM  
   EC2  
   CloudFormation  
   Note: Check eksctl documentaiton for [Minimum IAM policies](https://eksctl.io/usage/minimum-iam-policies/)
2. Create your cluster and nodes
3. eksctl create cluster --name cluster-name \
4. --region region-name \
5. --node-type instance-type \
6. --nodes-min 2 \
7. --nodes-max 2 \
8. --zones <AZ-1>,<AZ-2>
9. example:
10. eksctl create cluster --name valaxy-cluster \
11. --region ap-south-1 \

--node-type t2.small \

1. To delete the EKS clsuter

eksctl delete cluster valaxy --region ap-south-1

1. Validate your cluster using by creating by checking nodes and by creating a pod
2. kubectl get nodes

kubectl run tomcat --image=tomcat

#### Deploying Nginx pods on Kubernetes

1. Deploying Nginx Container
2. kubectl create deployment demo-nginx --image=nginx --replicas=2 --port=80
3. # kubectl deployment regapp --image=valaxy/regapp --replicas=2 --port=8080
4. kubectl get all

kubectl get pod

1. Expose the deployment as service. This will create an ELB in front of those 2 containers and allow us to publicly access them.
2. kubectl expose deployment demo-nginx --port=80 --type=LoadBalancer
3. # kubectl expose deployment regapp --port=8080 --type=LoadBalancer

kubectl get services -o wide