Install and configure apache/httpd

Install and configure nginx - configure it to run as reverse proxy to apache

Webpage should say

'Hi! i am devops ninja'

Install and configure 'ntp' - with singapore time zone

Install Tomcat version 8 (a brief explaination about the it's directories in doc)

Install java version 8 with home directory set as an environment variable

Install 'build essentials' (mention in doc that why do we need it )

Install logrotate and rotate tomcat's catalina.out log as:

rotate the log file after 500kb

keep only last 5 files

Make a directory as '/ninja' having one file inside with name 'devops' and share it via nfs & then mount the same on '/mnt'

Install git (a brief explaination about - what it is and why do we need it in doc)

After installing above check the respective logs if everything is installed and running (mention the log files name in doc)

Also mention other files got created with software installation

**=================================================================**

**Install and configure apache/httpd**

**Install and configure nginx - configure it to run as reverse proxy to apache**

* **Install apache2**

sudo apt-get update

sudo apt-get install apache2

* **Make index file**

sudo echo “Hi! I am devops ninja” >> /var/www/html/index.html

* **start apache**

sudo service apache2 start

sudo service apache2 status

* **Verify in browser**

Hit Cloud VM Public ip

curl –I localhost

* **By default nginx and apache2 both listen to port 80 so now we will make apache listen to port 8080**

edit ports.conf to change port 80 to 8080

sudo nano /etc/apache2/ports.conf

80->8080

443->8443

443->8443

Also edit /etc/apache2/sites-enabled/000-default.conf to change port and default document root which currently is /var/www/html to /var/www/html/apache

* **Restart apache and re-verify in browser**

sudo systemctl restart apache2

curl –I localhost

curl –I localhost:8080

* **Install nginx**

sudo apt-get install apache2

* **Similarly change nginx root location and create different index.html**

Now you can check that on:

Cloud VM public IP nginx html will appear and for

Cloud VM public IP:8080 apache html will appear

* **Now edit default file location{} to make nginx act as proxy server for apache and redirect all requests to apache**

sudo nano /etc/nginx/sites-available/default

location / {

proxy\_pass [http://127.0.0.1:8080](http://127.0.0.1:8080/);

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

Sudo systenctl restart nginx

* **Now hit public in of cloud VM in browser, if all went well you will get apache’s html instead on nginx’s, although on port 80 nginx was listning.**

**Install and configure 'ntp' - with singapore time zone**

Prerequisite: **tzdata** should be installed

sudo apt-get install ntp

sntp –version

sudo timedaectl set-timezone Asia/Singapore

sudo service ntp restart

Install build essentials:

sudo apt-get install build-essential

**Install logrotate and rotate tomcat's catalina.out log as:**

**rotate the log file after 500kb**

**keep only last 5 files**

Logrotate:

sudo apt-get install logrotate

Logrotate --version

* Create a configuration file in the following path: /etc/logrotate.d/ . For example: /etc/logrotate.d/tomcat
* Insert the following to the above file:  
   /opt/tomcat/apache-tomcat-8.5.42/logs/catalina.out {  
   copytruncate  
   daily  
   rotate 5  
   compress  
   missingok  
   size=500K  
   }

The following parameters are used for:

* copytruncate – truncates the original log file in place after creating a copy, instead of moving the old log file and optionally creating a new one.
* daily – rotate the log every day.
* rotate – how many rotation logs to keep.
* compress – If this value is written the log will be compressed.
* missingok – If the log file is missing, go on to the next one without issuing an error message.
* size #M – specify the size of the file before rotation.

**Install Tomcat version 8 (a brief explaination about the it's directories in doc)**

**Install java version 8 with home directory set as an environment variable**

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

Prerequisites:

EC2 instance (JAVA\_HOME set)

-Make dir in opt

cd /opt

mkdir tomcat

cd tomcat

-Download tomcat, unzip:

wget <http://apachemirror.wuchna.com/tomcat/tomcat-8/v8.5.42/bin/apache-tomcat-8.5.42.tar.gz>

tar -xvzf apache-tomcat-8.5.42.tar.gz

cd apache-tomcat-8.5.42

-Give the executable permission to startup.sh and shutdown.sh file of tomcat

chmod +x /opt/apache-tomcat-8.5.42/bin/startup.sh shutdown.sh

Start the tomcat server

sh startup.sh

Check the tomcat server in the browser:

[http://ip:8080](http://ip:8080/)

=> Directory Structure of Tomcat:

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

bin - startup, shutdown and other executables files

lib - jar files that make up the tomcat server classes and Jar files that contain classes shared between the Tomcat servers.

conf - configurations files included server.xml, server-user-xml etc. to configure Tomcat

logs - log files.

webapps - Any .war files placed here will be automatically serverd in the webpage.

* The description below uses the variable name $CATALINA\_BASE to refer the base directory against which most relative paths are resolved. If you have not configured Tomcat for multiple instances by setting a CATALINA\_BASE directory, then $CATALINA\_BASE will be set to the value of $CATALINA\_HOME, the directory into which you have installed Tomcat.

In order to be executed, a web application must be deployed on a servlet container. This is true even during development. We will describe using Tomcat to provide the execution environment. A web application can be deployed in Tomcat by one of the following approaches:

Copy unpacked directory hierarchy into a subdirectory in directory $CATALINA\_BASE/webapps/. Tomcat will assign a context path to your application based on the subdirectory name you choose. We will use this technique in the build.xml file that we construct, because it is the quickest and easiest approach during development. Be sure to restart Tomcat after installing or updating your application.

Copy the web application archive file into directory $CATALINA\_BASE/webapps/. When Tomcat is started, it will automatically expand the web application archive file into its unpacked form, and execute the application that way. This approach would typically be used to install an additional application, provided by a third party vendor or by your internal development staff, into an existing Tomcat installation. NOTE - If you use this approach, and wish to update your application later, you must both replace the web application archive file AND delete the expanded directory that Tomcat created, and then restart Tomcat, in order to reflect your changes.

================================================================

**Install 'build essentials' (mention in doc that why do we need it )**

The build-essentials package is a reference for all the packages needed to compile a Debian package. It

generally includes the GCC/g++ compilers and libraries and some other utilities

sudo apt-get update

sudo apt-get install build-essential

================================================================

**Install logrotate and rotate tomcat's catalina.out log as:**

**rotate the log file after 500kb**

**keep only last 5 files**

Logrotate:

sudo apt-get install logrotate

Logrotate --version

Create a configuration file in the following path: /etc/logrotate.d/ . For example: /etc/logrotate.d/tomcat

Insert the following to the above file:

/opt/tomcat/apache-tomcat-8.5.42/logs/catalina.out {

copytruncate

daily

rotate 5

compress

missingok

size=500K

}

================================================================

**Install git (a brief explaination about - what it is and why do we need it in doc)**

apt-get install git-core

-check version

git --version

-Configure git settings for root user:

git config --global user.name "abhishek.sachan"

git config --global user.email "abhishek.sachan@mygurukulam.org"

-Verify changes done:

cat ~/.gitconfig

**Version control Syatem:**

A version control system (VCS) allows you to track the history of a collection of files. It supports creating different versions of this collection. Each version captures a snapshot of the files at a certain point in time and the VCS allows you to switch between these versions. These versions are stored in a specific place, typically called a repository.

**Globalised and localised VCS:**

A localized version control system keeps local copies of the files. This approach can be as simple as creating a manual copy of the relevant files.

A centralized version control system provides a server software component which stores and manages the different versions of the files. A developer can copy (checkout) a certain version from the central sever onto their individual computer.

Both approaches have the drawback that they have one single point of failure. In a localized version control systems it is the individual computer and in a centralized version control systems it is the server machine. Both system makes it also harder to work in parallel on different features

**Distributed VCS:**

In a distributed version control system each user has a complete local copy of a repository on his individual computer. The user can copy an existing repository. This copying process is typically called cloning and the resulting repository can be referred to as a clone.

Every clone contains the full history of the collection of files and a cloned repository has the same functionality as the original repository.

Every repository can exchange versions of the files with other repositories by transporting these changes. This is typically done via a repository running on a server which is, unlike the local machine of a developer, always online. Typically, there is a central server for keeping a repository but each cloned repository is a full copy of this repository. The decision which of the copies is considered to be the central server repository is pure convention.

**GIT is a distribute VCS**

================================================================

**After installing above check the respective logs if everything is installed and running (mention the log files name in doc)**

**Also mention other files got created with software installation**

Go to /var/log directory using the following cd command:

# cd /var/log

To list files use the following ls command:

# ls

To view a common log file called /var/log/messages use any one of the following command:

# less /var/log/messages

# more -f /var/log/messages

# cat /var/log/messages

# tail -f /var/log/messages

# grep -i error /var/log/messages

Common Linux log files names and usage

/var/log/messages : General message and system related stuff

/var/log/auth.log : Authenication logs

/var/log/kern.log : Kernel logs

/var/log/cron.log : Crond logs (cron job)

/var/log/maillog : Mail server logs

/var/log/qmail/ : Qmail log directory (more files inside this directory)

/var/log/httpd/ : Apache access and error logs directory

/var/log/lighttpd/ : Lighttpd access and error logs directory

/var/log/boot.log : System boot log

/var/log/mysqld.log : MySQL database server log file

/var/log/secure or /var/log/auth.log : Authentication log

/var/log/utmp or /var/log/wtmp : Login records file

/var/log/yum.log : Yum command log file.

=================================================================

**Make a directory as '/ninja' having one file inside with name 'devops' and share it via nfs & then mount the same on '/mnt'**

**install nfs at SERVER:-**

yum install nfs-utils

mkdir ninja/

cd ninja

touch testfile

cd ..

chmod -R 755 ninja/

chown nfsnobody:nfsnobody ninja/

systemctl enable rpcbind

systemctl enable nfs-server

systemctl enable nfs-lock

systemctl enable nfs-idmap

systemctl start rpcbind

systemctl start nfs-server

systemctl start nfs-lock

systemctl start nfs-idmap

vim /etc/exports

/ninja \*(rw,sync,no\_root\_squash,no\_all\_squash)

systemctl restart nfs-server

**NOTE:**

The permissions “rw,sync,no\_subtree\_check” permissions defined in this file mean that the client(s) can perform:

rw: read and write operations

sync: write any change to the disc before applying it

no\_subtree\_check: prevent subtree checking

**install nfs at CLIENT:-**

yum install nfs-utils

mkdir -p sharedninja

mount -t nfs serverIP:/ninja /mnt

df -HT

vim /etc/fstab:-

ServerIP:/ninja /mnt/nfs/ninja nfs defaults 0 0