DEPARTMENT OF
INFORMATION TECHNOLOGY

DevOps Lab

LABORATORY MANUAL

SIES GRADUATE SCHOOL OF TECHNOLOGY
PLOT 1-C, D & E, SECTOR V, NERUL, NAVI MUMBAI-400 706

VISION OF THE INSTITUTE

To be a center of excellence in Engineering Education committed towards Socio-Economic Advancement of the country.

MISSION OF THE INSTITUTE

1. To impart advanced knowledge in Engineering and Technology.
2. To transform young minds towards professional competence by inculcating values and developing skills.
3. To promote research and ensure continuous value addition among students and employees.
4. To strengthen association with industry, research organizations and alumni to enhance knowledge on current technologies.
5. To promote next generation technocracy and nurture entrepreneurial culture for social-economic growth.

VISION OF THE DEPARTMENT

To develop IT professionals for accomplishment of industrial & societal needs through quality education.

MISSION OF THE DEPARTMENT

1. To impart advanced knowledge and develop skills in Information Technology and allied fields.
2. To enhance professional competence by inculcating values and ethics.
3. To upgrade technical skills and also encourage research culture.
4. To extend industry and alumni association for knowledge enhancement.
5. To nurture entrepreneurial talent and contribute towards socio-economic growth.

COURSE OUTCOMES

At the end of the course the student should be able to:

1. Summarize the importance of DevOps tools used in software development life cycle
2. Summarize the importance of Jenkins to Build, Deploy and Test Software Applications
3. Examine the different Version Control strategies
4. Analyze & Illustrate the Containerization of OS images and deployment of applications over Docker
5. Summarize the importance of Software Configuration Management in DevOps
6. Synthesize the provisioning using Chef/Puppet/Ansible or Saltstack

Course Articulation Matrix: Mapping of Course Outcomes (COs) with Program Outcomes (POs)

Course Outcomes (COs) / Program Outcomes (POs)	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO 2
1. Summarize the importance of DevOps tools used in software development life cycle	2	3	3	2	3	2	2	3	3	3	2	2	3	2
2. Summarize the importance of Jenkins to Build, Deploy and Test Software Applications	2	3	3	2	3	2	2	3	3	3	2	2	2	2
3. Examine the different Version Control strategies	2	3	3	2	3	2	2	3	3	3	2	2	3	2
4. Analyze & Illustrate the Containerization of OS images and deployment of applications over Docker	2	3	3	2	3	2	2	3	3	3	2	2	3	2
5. Summarize the importance of Software Configuration Management in DevOps	2	3	3	2	3	2	2	3	3	3	2	2	2	2
6. Synthesize the provisioning using Chef/Puppet/Ansible or Saltstack	2	3	3	2	3	2	2	3	3	3	2	2	2	2

CODE OF CONDUCT FOR THE LABORATORIES

- All the students must observe the Dress Code while in the laboratory.
- Sandals or open-toed shoes are NOT allowed.
- Food, drinks and smoking are NOT allowed.
- All bags must be left at the indicated place.
- The lab timetable must be strictly followed.
- Be PUNCTUAL for your laboratory sessions.
- Program must be executed within the given time.
- Noise must be kept to minimum.
- Workspace must be kept clean and tidy all the time.
- Handle the systems and interfacing kits with care.
- All students are liable for any damage to the accessories due to negligence.
- All interfacing kits connecting cables must be RETURNED to the lab supervisor.
- Students are strictly PROHIBITED from taking out any items from the laboratory.
- Students are NOT allowed to work in the laboratory without the Lab supervisor.
- USB Ports have been disabled. If you want to use USB drive consult lab supervisor.
- Report immediately to the Lab Supervisor of any malfunction of the accessories..

Before leaving the lab

- Place the chairs properly.
- Turn off the system properly.
- Turn off the monitor.

Please check the laboratory notice board regularly for updates.

End semester lab examination, conducted as per the UOM regulations, carries 25 marks of ORAL.

At the end of each laboratory session you must obtain the signature of the teacher along with the marks for the session out of 05 on the lab notebook and 05 for journal.

Lab Reports

- Note that, although students are encouraged to collaborate during lab, each must individually prepare a report and submit.
- They must be organized, neat and legible.
- Your report should be complete, thorough and understandable.
- You should include a well-drawn and labeled engineering schematic for each circuit investigated.
- Your reports should follow the prescribed format, to give your report structure and to make sure that you address all important points.
- Graphics requiring- drawn straight lines should be done with a straight edge. Well drawn free-hand sketches are permissible for schematics.
- Space must be provided in the flow of your discussion for any tables or figures. Do not collect figures and drawings in a single appendix at the end of the report.
- Reports should be submitted within one week after completing a scheduled lab session.

Presentation

- Experimental facts should always be given in the past tense.
- Discussions or remarks about the presentation of data should mainly be in the present tense.
- Discussion of results can be in both the present and past tenses, shifting back and forth from experimental facts to the presentation.
- Any specific conclusions or deductions should be expressed in the past tense.

Report Format:

Lab write-up should consist of the following sections:

Aim:

Instructions used:

Theory :

Diagram(if any):

Algorithm:

Result:

Conclusions:

Experiment No 1

Aim:-Study of Devops Tools.

Theory:-

1. Nagios:-Nagios now known as Nagios Core, is a free and open-source computer-software application that monitors systems, networks and infrastructure.
Nagios offers monitoring and alerting services for servers, switches, applications and services. It alerts users when things go wrong and alerts them a second time when the problem has been resolved.
Nagios Core is open source software licensed under the GNU GPL V2.
Currently it provides:

- Monitoring of network services (SMTP, POP3, HTTP, NNTP, ICMP, SNMP, FTP, SSH)
- Monitoring of host resources (processor load, disk usage, system logs) on a majority of network operating systems, including Microsoft Windows, using monitoring agents.

2. Chef:-Chef is a powerful automation tool that can deploy, repair and update and also manage server and application to any environment
Chef is a Configuration management tool that manages the infrastructure by writing code rather than using a manual process so that it can be automated, tested and deployed very easily. Chef has Client-server architecture and it supports multiple platforms like Windows, Ubuntu, Centos, and Solaris etc. It can also be integrated with cloud platform like AWS, Google Cloud Platform, and Open Stack etc. Before getting into Chef deeply let us understand Configuration Management.

Benefits of Chef

Accelerating software delivery, when your infrastructure is automated all the software requirements like testing, creating new environments for software deployments etc. becomes faster.

Increased service Resiliency, by making the infrastructure automated it monitors for bugs and errors before they occur it can also recover from errors more quickly.

3. SaltStack:-Salt (sometimes referred to as SaltStack) is Python-based, open-source software for event-driven IT automation, remote task execution, and configuration management. Supporting the "Infrastructure as Code" approach to data center system and network deployment and management, configuration automation, SecOps orchestration, vulnerability remediation, and hybrid cloud control. Salt is most often compared to tools from BMC, Computer Associates, IBM, Puppet, Chef, and Ansible.

Remote execution

Salt originated from the need for high-speed data collection and task execution for data center systems administrators managing massive infrastructure scale and resulting complexity. The author of Salt, Thomas S. Hatch, had previously created several utilities for IT teams to solve the problem of systems management at scale, but found these and other open source solutions to be lacking.

Configuration management

In late May 2011 initial progress was made toward the delivery of configuration management built on the Salt remote execution engine.[4] This configuration management system stores all configuration (state) data inside an easily understood data structure that leverages YAML

4. Ansible:-Ansible is an open-source software provisioning, configuration management, and application-deployment tool. It runs on many Unix-like systems, and can configure both Unix-like systems as well as Microsoft Windows. It includes its own declarative language to describe system configuration.

Ansible was written by Michael DeHaan and acquired by Red Hat in 2015. Ansible is agentless, temporarily connecting remotely via SSH or remote PowerShell to do its tasks

Design goals

The design goals of Ansible include

- Minimal in nature. Management systems should not impose additional dependencies on the environment.
- Consistent. With Ansible one should be able to create consistent environments.

5. Puppet:-Puppet is an open-core software configuration management tool. It runs on many Unix-like systems as well as on Microsoft Windows, and includes its own declarative language to describe system configuration. It is written in C++, Clojure and Ruby, with its free-software version released under the Apache License 2.0 (the GNU General Public License (GPL) until version 2.7.0.

Puppet is designed to manage the configuration of Unix-like and Microsoft Windows systems declaratively. The user describes system resources and their state, either using Puppet's declarative language or a Ruby DSL (domain-specific language). This information is stored in files called "Puppet manifests".

6. CVS(Concurrent Versions System):-The Concurrent Versions System (CVS), also known as the Concurrent Versioning System, is a free client-server revision control system in the field of software development. A version control system keeps track of all work and all changes in a set of files, and allows several developers (potentially widely separated in space and time) to collaborate.

CVS uses a client–server architecture: a server stores the current version(s) of a project and its history, and clients connect to the server in order to "check out" a complete copy of the project, work on this copy and then later "check in" their changes.

7. RCS(Revision Control System):-RCS is an early version control system (VCS). It is a set of UNIX commands that allow multiple users to develop and maintain program code or documents. With RCS, users can make their own revisions of a document, commit changes, and merge them. RCS was originally developed for programs but is also useful for text documents or configuration files that are frequently revised.

An innovation in RCS is the adoption of *reverse deltas*. Instead of storing every revision in a file like SCCS does with interleaved deltas, RCS stores a set of edit instrutions to go back to an earlier version of the file. Tichy claims that it is faster for most cases because the recent revisions are used more often.

8. Docker:- Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating-system kernel and are thus more lightweight than virtual machines.

The service has both free and premium tiers. The software that hosts the containers is called Docker Engine. It was first started in 2013 and is developed by Docker, Inc.

9. Jenkins:- Jenkins is a free and open source automation server. Jenkins helps to automate the non-human part of the software development process, with continuous integration and facilitating technical aspects of continuous delivery

It is a server-based system that runs in servlet containers such as Apache Tomcat. It supports version control tools, including AccuRev, CVS, Subversion, Git, Mercurial, Perforce, TD/OMS, ClearCase and RTC, and can execute Apache Ant, Apache Maven and sbt based projects as well as arbitrary shell scripts and Windows batch commands.

Conclusion:-

Experiment No. 2

Aim: To perform version control using Git

Theory:

GitHub is a global company that provideshostingforsoftware developmentversion controlusingGit. It is a subsidiary ofMicrosoft, which acquired the company in 2018 for $7.5 billion.It offers all of thedistributed version controlandsource code management(SCM) functionality of Git as well as adding its own features. It providesaccess controland several collaboration features such asbug tracking,feature requests,task management, andwikisfor every project.

GitHub offers plans for free, professional, and enterprise accounts.Free GitHub accounts are commonly used to hostopen sourceprojects.As of January 2019, GitHub offers unlimited private repositories to all plans, including free accounts.As of May 2019, GitHub reports having over 37 million users and more than 100 millionrepositories(including at least 28 million public repositories),making it the largest host ofsource codein the world.

Making a new Git folder

```
it68@puppet:~$ mkdir git
it68@puppet:~$ cd git
```

Setting up local repository

```
it68@puppet:~/git$ git config --global user.name "wolf"
it68@puppet:~/git$ git config --global user.email "karthik.iyer16@siesgst.ac.in"

it68@puppet:~/git$ git config --global list
error: key does not contain a section: list
it68@puppet:~/git$ git config --global --list
user.email=karthik.iyer16@siesgst.ac.in
user.name=wolf
```

Setting up new folder for git project

```
it68@puppet:~/git$ mkdir git-demo-project
it68@puppet:~/git$ git init
Initialized empty Git repository in /home/it68/git/.git/
it68@puppet:~/git$ ls -a
.  ..  .git  git-demo-project
it68@puppet:~/git$ rm -rf .git/
it68@puppet:~/git$ git init
Initialized empty Git repository in /home/it68/git/.git/
it68@puppet:~/git$ ls -al
total 16
drwxr-xr-x  4 it68 it68 4096 Jan 16 09:28 .
drwxr-xr-x 26 it68 it68 4096 Jan 16 09:20 ..
drwxr-xr-x  7 it68 it68 4096 Jan 16 09:28 .git
drwxr-xr-x  2 it68 it68 4096 Jan 16 09:23 git-demo-project
```

Adding files to git

```
it68@puppet:~/git$ cd git-demo-project/
it68@puppet:~/git/git-demo-project$ cd ..
it68@puppet:~/git$ git add .
it68@puppet:~/git$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
it68@puppet:~/git$ gedit
```

The file

```
Open ▾                    First_git_file
                          ~/git/git-demo-project
First doc to upload in git
1. First addition and edit to file
2. checking the new commit
```

Committing changes

```
it68@puppet:~/git$ git commit -m "first commit"
On branch master

Initial commit

Untracked files:
        git-demo-project/
```

```
it68@puppet:~/git$ cd git-demo-project/
it68@puppet:~/git/git-demo-project$ git commit -m "first commit"
On branch master

Initial commit

Untracked files:
        ./

nothing added to commit but untracked files present
```

```
it68@puppet:~/git/git-demo-project$ cd ..
it68@puppet:~/git$ git add .
it68@puppet:~/git$ cd git-demo-project/
it68@puppet:~/git/git-demo-project$ git add .
it68@puppet:~/git/git-demo-project$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   First_git_file
```

Detecting changed files

```
it68@puppet:~/git/git-demo-project$ gedit First_git_file
it68@puppet:~/git/git-demo-project$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   First_git_file

no changes added to commit (use "git add" and/or "git commit -a")
```

First commit and express commit

```
it68@puppet:~/git/git-demo-project$ touch teststatus
it68@puppet:~/git/git-demo-project$ git add teststatus
it68@puppet:~/git/git-demo-project$ git add First_git_file
it68@puppet:~/git/git-demo-project$ git commit -am "express commit"
[master ff29011] express commit
 2 files changed, 2 insertions(+)
 create mode 100644 git-demo-project/teststatus
it68@puppet:~/git/git-demo-project$ git log
commit ff29011aec7c69120e67609ce1282aa0d96e59a3 (HEAD -> master)
Author: wolf <karthik.iyer16@siesgst.ac.in>
Date:   Thu Jan 16 09:36:16 2020 +0530

    express commit

commit 4d3d00189fab28c703a964c8d985199ba5737ea8
Author: wolf <karthik.iyer16@siesgst.ac.in>
Date:   Thu Jan 16 09:33:14 2020 +0530

    first commit
```

Adding remote repository

```
it68@puppet:~/git/git-demo-project$ git remote add origin https://github.com/The
LoneW01f/MyRepository.git
it68@puppet:~/git/git-demo-project$ git push -u origin master
Username for 'https://github.com': TheLoneW01f
Password for 'https://TheLoneW01f@github.com':
Counting objects: 9, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (9/9), 689 bytes | 344.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0)
To https://github.com/TheLoneW01f/MyRepository.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Pulling changes from remote repository

6 lines (5 sloc) | 200 Bytes Raw Blame History

```
First doc to upload in git
1. First addition and edit to file
2. checking the new commit

Now editing from the remote repository...
1. First addition in git editor (must say im not a big fan of this)
```

```
it68@puppet:~/git/git-demo-project$ git pull
Username for 'https://github.com': TheLoneW01f
Password for 'https://TheLoneW01f@github.com':
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://github.com/TheLoneW01f/MyRepository
   ff29011..1105fc7  master     -> origin/master
Updating ff29011..1105fc7
Fast-forward
 git-demo-project/First_git_file | 2 ++
 1 file changed, 2 insertions(+)
```

Reading Log of commits

```
it68@puppet:~/git/git-demo-project$ git log --oneline origin/master
1105fc7 (HEAD -> master, origin/master) Update First_git_file
ff29011 express commit
4d3d001 first commit
```

Trying fetch

```
First doc to upload in git
1. First addition and edit to file
2. checking the new commit

Now editing from the remote repository...
1. First addition in git editor (must say im not a big fan of this)

Trying out the fetch command....Doesnt seem much different from pull
```

```
it68@puppet:~/git/git-demo-project$ git fetch
Username for 'https://github.com': TheLoneW01f
Password for 'https://TheLoneW01f@github.com':
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://github.com/TheLoneW01f/MyRepository
   1105fc7..fd65eb9  master     -> origin/master
```

```
it68@puppet:~/git/git-demo-project$ git merge origin/master
Updating 1105fc7..fd65eb9
Fast-forward
 git-demo-project/First_git_file | 2 ++
 1 file changed, 2 insertions(+)
it68@puppet:~/git/git-demo-project$ cat First_git_file
First doc to upload in git
1. First addition and edit to file
2. checking the new commit

Now editing from the remote repository...
1. First addition in git editor (must say im not a big fan of this)

Trying out the fetch command....Doesnt seem much different from pull
```

Postlab (RCS)

Configuration

```
it68@puppet:~/RCS/rcs-5.7$ ./configure
creating cache ./config.cache
checking diff basename... diff
checking for diff... /usr/bin/diff
checking diff success status... 0
checking diff failure status... 1
checking diff trouble status... 2
checking diff options for RCS... -an
checking diff -L... yes
checking diff3 -m... yes
checking for diff3... /usr/bin/diff3
checking for ed... /bin/ed
checking for sendmail... no
checking for mail... no
checking for mailx... no
checking for pic... pic -n
checking for gcc... gcc
checking whether we are using GNU C... yes
checking whether gcc accepts -g... yes
checking for a BSD compatible install... /usr/bin/install -c
checking whether make sets ${MAKE}... yes
updating cache ./config.cache
creating ./config.status
creating Makefile
creating man/Makefile
creating src/Makefile
it68@puppet:~/RCS/rcs-5.7$
```

Compiling the package

Making installation of packages

```
it68@puppet:~/RCS/rcs-5.7$ sudo su
[sudo] password for it68:
root@puppet:/home/it68/RCS/rcs-5.7# make install
cd man && make install
make[1]: Entering directory '/home/it68/RCS/rcs-5.7/man'
./../mkinstalldirs /usr/local/man/man1 /usr/local/man/man5
mkdir /usr/local/man/man1
mkdir /usr/local/man/man5
for m in ci co ident merge rcs rcsclean rcsdiff rcsintro rcsmerge rlog; do \
        /usr/bin/install -c -m 644 ./$m.1 /usr/local/man/man1/$m.1; \
done
{ test -f rcsfile.5 || cd .; } && \
/usr/bin/install -c -m 644 rcsfile.5 /usr/local/man/man5/rcsfile.5
make[1]: Leaving directory '/home/it68/RCS/rcs-5.7/man'
cd src && make install
make[1]: Entering directory '/home/it68/RCS/rcs-5.7/src'
./../mkinstalldirs /usr/local/bin
for p in ci co ident merge rcs rcsclean rcsdiff rcsmerge rlog; do \
  /usr/bin/install -c $p /usr/local/bin/$p; \
done
make[1]: Leaving directory '/home/it68/RCS/rcs-5.7/src'
```

Conclusion:-

Experiment No. 3

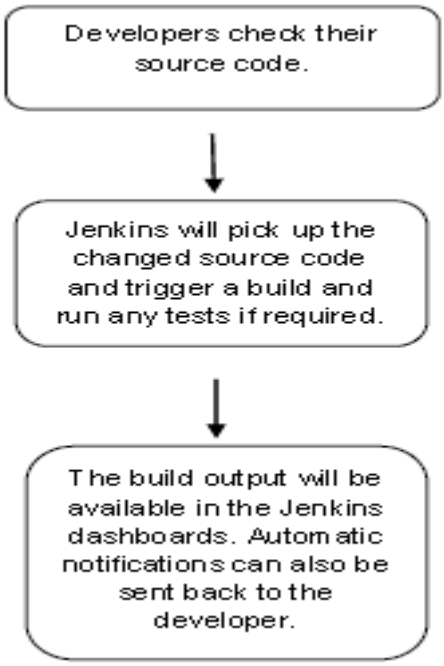
AIM: Study of Jenkins and its installation.

THEORY:

Jenkins is a powerful application that allows continuous integration and continuous delivery of projects, regardless of the platform you are working on. It is a free source that can handle any kind of build or continuous integration. You can integrate Jenkins with a number of testing and deployment technologies. In this tutorial, we would explain how you can use Jenkins to build and test your software projects continuously.

Jenkins is a software that allows **continuous integration**. Jenkins will be installed on a server where the central build will take place. The following flowchart demonstrates a very simple workflow of how Jenkins works.

For example: Maven 2 Project, Git, HTML Publisher, Amazon EC2, etc.



Along with Jenkins, sometimes, one might also see the association of **Hudson**. Hudson is a very popular open-source Java-based continuous integration tool developed by Sun Microsystems which was later acquired by Oracle. After the acquisition of Sun by Oracle, a fork was created from the Hudson source code, which brought about the introduction of Jenkins.

INSTALLATION:

To start Jenkins:

Installing Jenkins on Ubuntu

Before moving on to installing the packages on the server, our system package manager must be updated. Use the following command to ensure your system package manager is up to date:

$ sudo apt update

Install Java

Since Jenkins is written in Java, the first step is to install Java. Install the Java 8 OpenJDK package with the following command:

$ sudo apt install openjdk-8-jdk

The current version of Jenkins doesn't support Java 10 or more yet. If you have multiple java versions installed on your system then make sure java 8 is the default java version.

Install Jenkins

Once the Jenkins repository is enabled and keys and sources are added, update the apt package list:

1. $ sudo apt update

Now, install the latest version of Jenkins by using the following command:

1. $sudo apt install jenkins

Once the installation is completed, Jenkins service will start automatically.

We can verify it with the help of following command:

1. $ systemctl status Jenkins

We should see something like this:

jenkins.service-LSB: Start Jenkins at boot time

Loaded: loaded (/etc/init.d/jenkins;generated)

Active: active (exited) since Wed 2019-07-06 1308 PDT;2min 16s ago

Docs: man:systemd-sysv-generator(8)

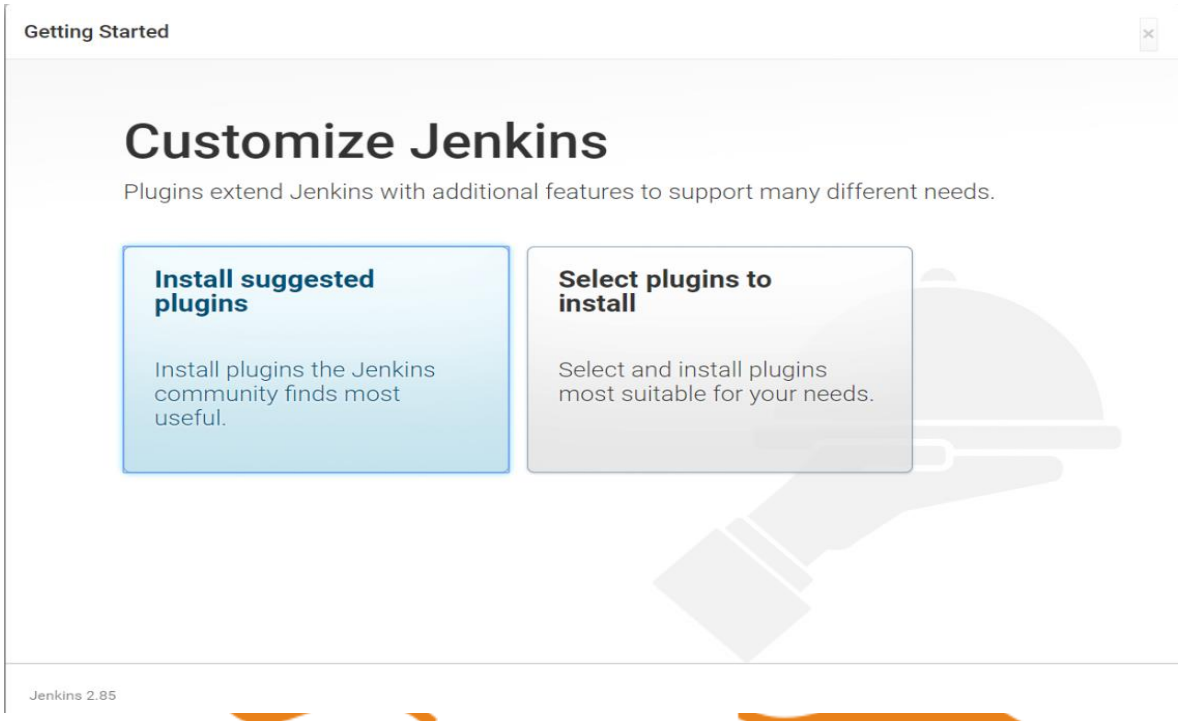
Tasks:0 (limit: 2319)

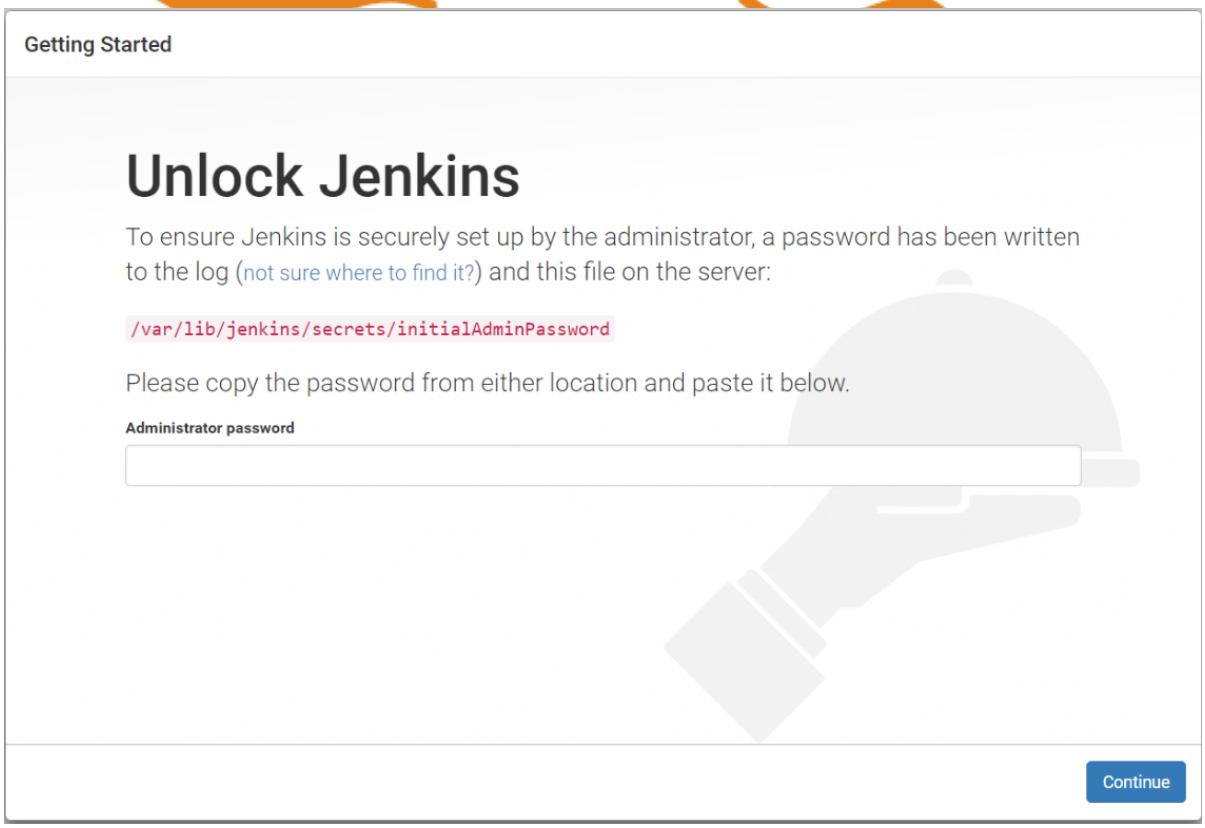
Cgroup: /system.slice/jenkins.service

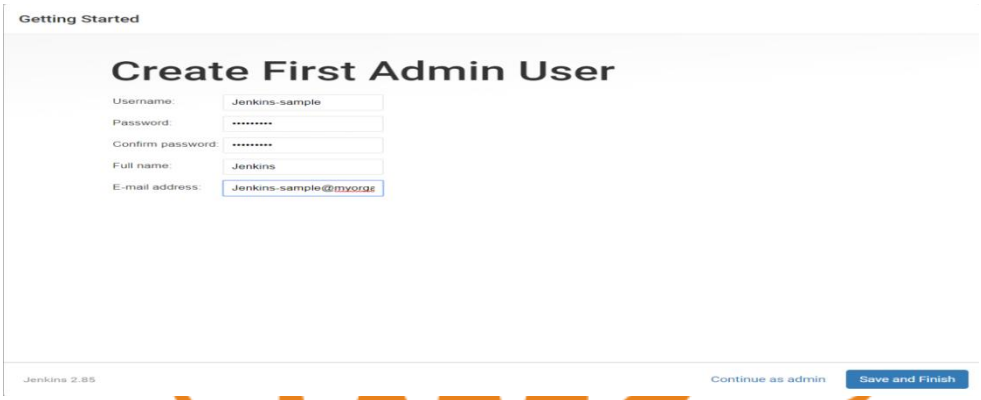
Accessing Jenkins:

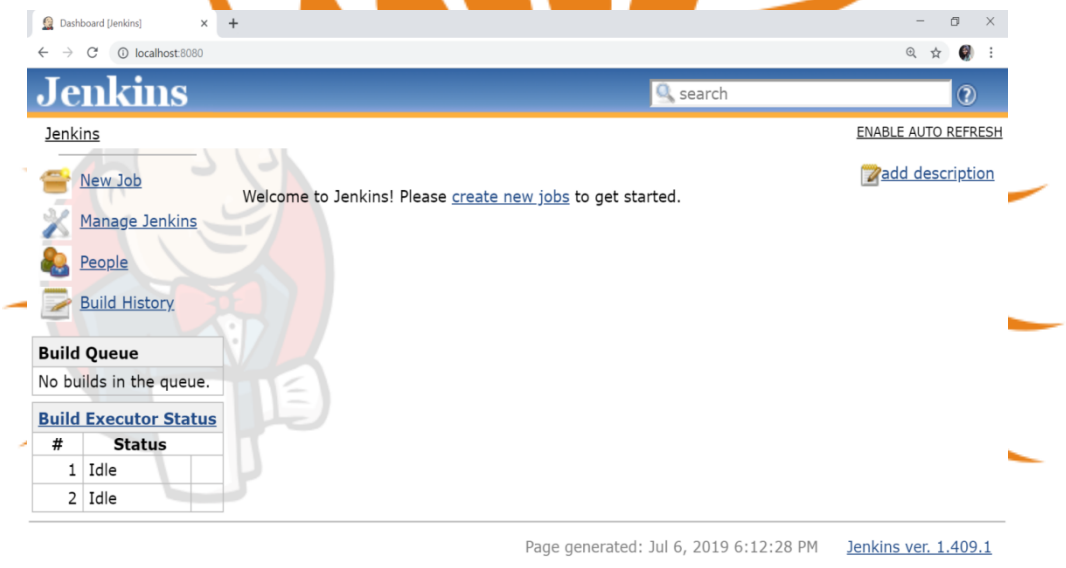
Once Jenkins is up and running, one can access Jenkins from the link −**http://localhost:8080**

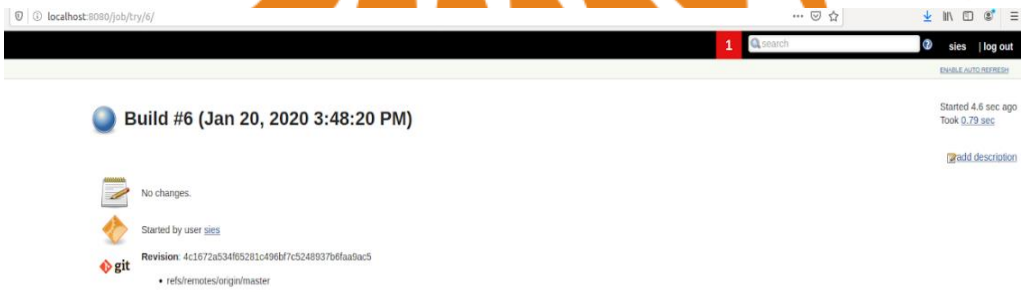
$sudo cat/var/lib/jenkins/secrets/initialAdminPassword


Getting Started
Customize Jenkins
Plugins extend Jenkins with additional features to support many different needs.
Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.
Jenkins 2.85


Getting Started
Unlock Jenkins
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:
/var/lib/jenkins/secrets/initialAdminPassword
Please copy the password from either location and paste it below.
Administrator password
Continue


Getting Started
Create First Admin User
Username:
Jenkins-sample
Password:
Confirm password:
Full name:
Jenkins
E-mail address:
Jenkins-sample@myorga
Jenkins 2.85
Continue as admin
Save and Finish


Dashboard [Jenkins]
localhost:8080
Jenkins
search
Jenkins
ENABLE AUTO REFRESH
New Job
add description
Welcome to Jenkins! Please create new jobs to get started.
Manage Jenkins
People
Build History
Build Queue
No builds in the queue.
Build Executor Status
Status
1 Idle
2 Idle
Page generated: Jul 6, 2019 6:12:28 PM
Jenkins ver. 1.409.1


localhost:8080/job/try/6/
1
search
sies
log out
ENABLE AUTO REFRESH
Build #6 (Jan 20, 2020 3:48:20 PM)
Started 4.6 sec ago
Took 0.79 sec
add description
No changes.
Started by user sies
git
Revision: 4c1672a534f65281c496bf7c5248937b6faa9ac5
refs/remotes/origin/master

Conclusion:

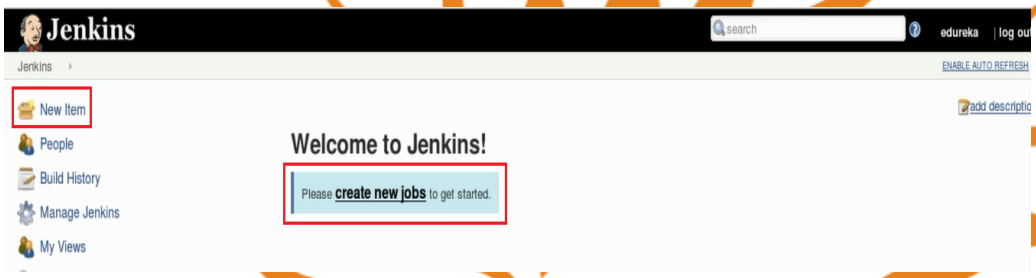
Experiment No:4

AIM: To perform testing and Deployment using Jenkins

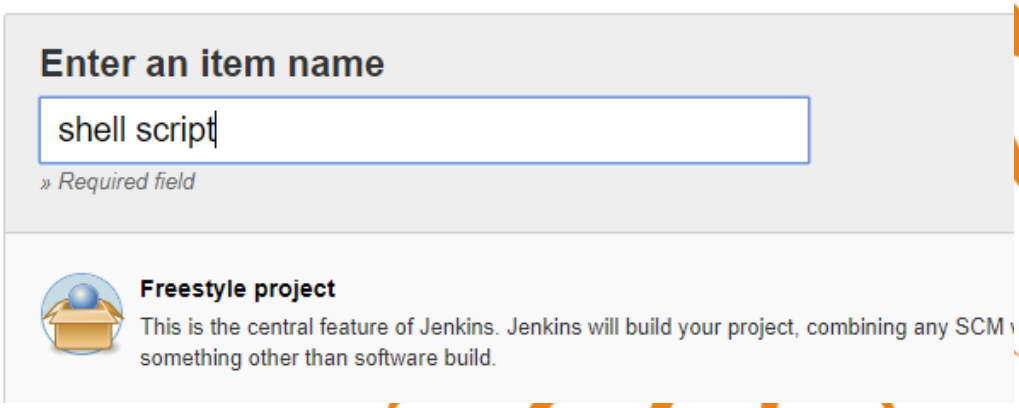
THEORY:

Creating a Build Using Jenkins

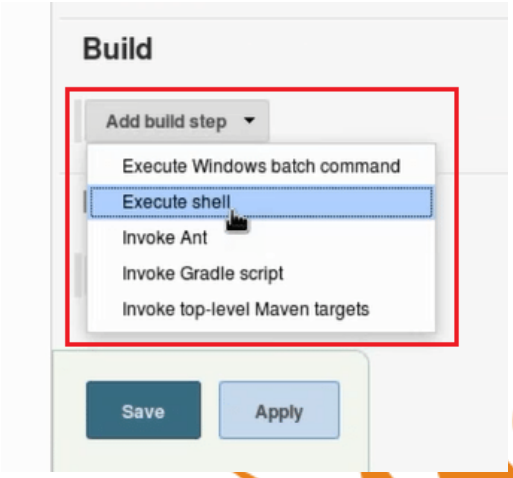
Step 1: From the Jenkins interface home, select **New Item.**



Step 2: Enter a name and select **Freestyle project**.



Step 3: This next page is where you specify the job configuration. As you'll quickly observe, there are a number of settings available when you create a new project. On this configuration page, you also have the option to **Add build step** to perform extra actions like running scripts. I will execute a shell script.



Integration of github repository with Jenkins.

Configuring GitHub

Step 1: go to your GitHub repository and click on 'Settings'

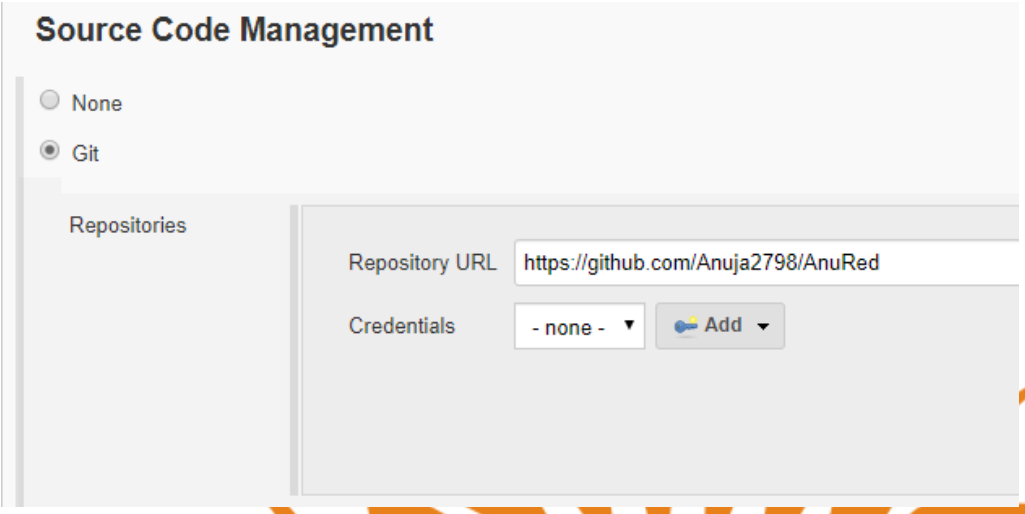
Step 2: Click on Webhooks and then click on 'Add webhook'.

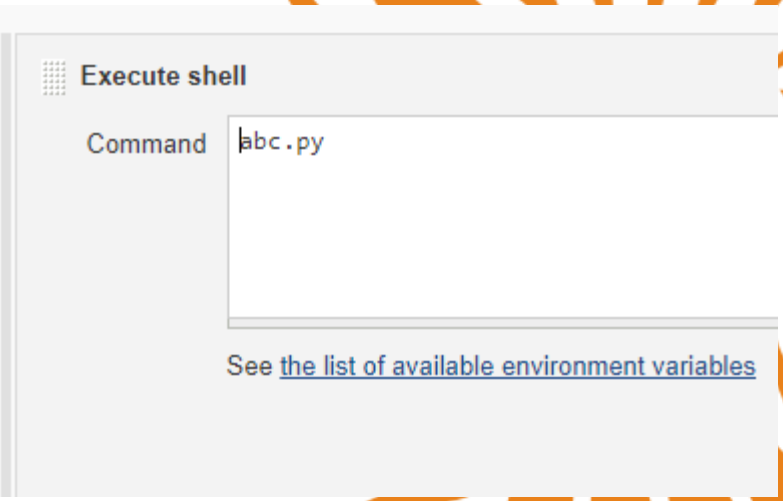
Step 3: in the 'Payload URL' field, paste your Jenkins environment URL. At the end of this URL add /github-webhook/. In the 'Content type' select 'application/json' and leave the 'Secret' field empty.

Step 4: in the 'Which events would you like to trigger this webhook?' choose 'Let me select individual events.' Then, check 'Pull Requests' and 'Pushes'. At the end of this option, make sure that 'Active' is checked and click on 'Add webhook'.

Step 5: In Jenkins, click on 'New Item' to create a new project.
Step 6: Give your project a name, them choose 'Freestyle project' and finally click on 'OK'.
Step 7: Click on the 'Source Code Management' tab.

Step 8: Click on Git and paste your GitHub repository URL in the 'Repository URL' field.
Step 9: Click on the 'Build Triggers' tab and then on the 'GitHub hook trigger for GITScm polling'. Or, choose the trigger of your choice.
That's it! Your GitHub repository is integrated with your Jenkins project.
Step 10: Click on the 'Build' tab, then click on 'Add build step' and choose 'Execute shell'.


Source Code Management
None
Git
Repositories
Repository URL
https://github.com/Anuja2798/AnuRed
Credentials
- none -
Add


Execute shell
Command
abc.py
See the list of available environment variables

Anuja2798 / AnuRed

Code | Issues 0 | Pull requests 0 | Actions | Projects 0 | Wiki

No description, website, or topics provided.

Manage topics

2 commits | 1 branch | 0 packages

Branch: master | New pull request

Anuja2798 Express commit

abc.py	Express commit
index.html	Express commit

Help people interested in this repository understand your project by adding a README.

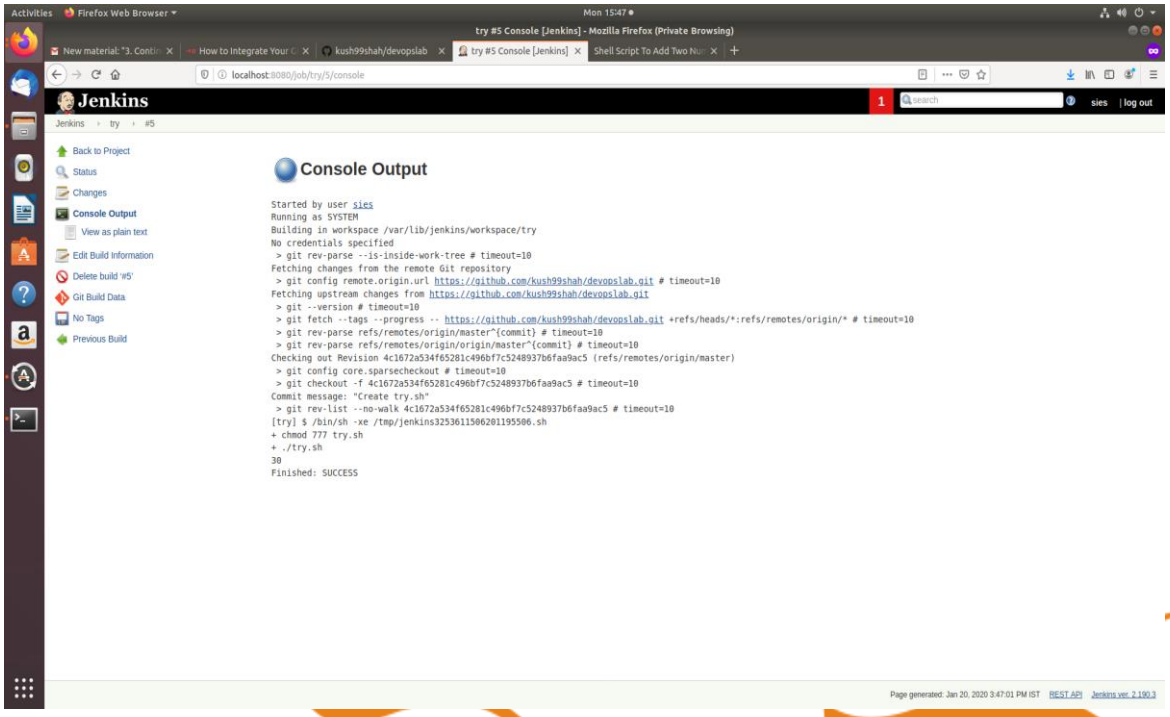
Anuja2798 Express commit

1 contributor

5 lines (4 sloc) | 43 Bytes

Code navigation is available for this reposi

```
a=10
b=20
sum = a + b.
print("sum:", sum)
```


Activities
Firefox Web Browser
Mon 15:47
try #5 Console [Jenkins] - Mozilla Firefox (Private Browsing)
New material: "3. Contin
How to Integrate Your G
kush99shah/devopslab
try #5 Console [Jenkins]
Shell Script To Add Two Nu
localhost:8080/job/try/5/console
Jenkins
search
sies
| log out
Jenkins › try › #5
Back to Project
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#5'
Git Build Data
No Tags
Previous Build
Console Output
Started by user sies
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/try
No credentials specified
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/kush99shah/devopslab.git # timeout=10
Fetching upstream changes from https://github.com/kush99shah/devopslab.git
> git --version # timeout=10
> git fetch --tags --progress -- https://github.com/kush99shah/devopslab.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 4c1672a534f65281c496bf7c5248937b6faa9ac5 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 4c1672a534f65281c496bf7c5248937b6faa9ac5 # timeout=10
Commit message: "Create try.sh"
> git rev-list --no-walk 4c1672a534f65281c496bf7c5248937b6faa9ac5 # timeout=10
[try] $ /bin/sh -xe /tmp/jenkins3253611506201195506.sh
+ chmod 777 try.sh
+ ./try.sh
30
Finished: SUCCESS
Page generated: Jan 20, 2020 3:47:01 PM IST
REST API
Jenkins ver. 2.190.3

Conclusion:-

Experiment No. 5

Aim: To study and Install Docker for contanerization.

Theory:

- Docker is a software containerization platform, meaning you can build your application, package them along with their dependencies into a container and then these containers can be easily shipped to run on other machines.
- These containers use Containerization which can be considered as an evolved version of Virtualization. The same task can also be achieved using Virtual Machines;however it is not very efficient. Docker is gaining popularity and its usage is spreading like wildfire. The reason for Docker's growing popularity is the extent to which it can be used in an IT organization.
- Virtualization is the technique of importing a Guest operating system on top of a Host operating system. Virtualization also has some shortcomings. Running multiple Virtual Machines in the same host operating system leads to performance degradation.
- Another problem with Virtual Machines which uses virtualization is that it takes almost a minute to boot-up. This is very critical in case of real-time applications.
- Containerization is the technique of bringing virtualization to the operating system level. While Virtualization brings abstraction to the hardware, Containerization brings abstraction to the operating system.
- Do note that Containerization is also a type of Virtualization. Containerization is however more efficient because there is no guest OS here and utilizes a host's operating system, share relevant libraries & resources as and when needed unlike virtual machines. Application specific binaries and libraries of containers run on the host kernel, which makes processing and execution very fast. Even booting-up a container takes only a fraction of a second.
- Because all the containers share, host operating system and holds only the application related binaries & libraries. They are lightweight and faster than Virtual Machines.
- On Docker, each application will run on a separate container and will have its own set of libraries and dependencies. This also ensures that there is process level isolation, meaning each application is independent of other applications, giving developers surety that they can build applications that will not interfere with one another.

Installation of Docker:

1. Installing required packages: Curl and setting up docker packages

```
 Version:          1.0.0-rc8+dev
 GitCommit:        3e425f80a8c931f88e6d94a8c831b9d5aa481657
docker-init:
 Version:          0.18.0
 GitCommit:        fec3683
If you would like to use Docker as a non-root user, you should now consider
adding your user to the "docker" group with something like:

 sudo usermod -aG docker your-user

Remember that you will have to log out and back in for this to take effect!

WARNING: Adding a user to the "docker" group will grant the ability to run
         containers which can be used to obtain root privileges on the
         docker host.
         Refer to https://docs.docker.com/engine/security/security/#docker-daemon-attack-surface
         for more information.
```

```
root@it74-OptiPlex-3050: /home/it74/Desktop
File  Edit  View  Search  Terminal  Help
root@it74-OptiPlex-3050:/home/it74/Desktop# curl -fsSL https://get.docker.com -o get-docker.sh
root@it74-OptiPlex-3050:/home/it74/Desktop# sudo sh get-docker.sh
# Executing docker install script, commit: f45d7c11389849ff46a6b4d94e0dd1ffebca32c1
Warning: the "docker" command appears to already exist on this system.

If you already have Docker installed, this script can cause trouble, which is
why we're displaying this warning and provide the opportunity to cancel the
installation.

If you installed the current Docker package using this script and are using it
again to update Docker, you can safely ignore this message.

You may press Ctrl+C now to abort this script.
+ sleep 20
+ sh -c apt-get update -qq >/dev/null
+ sh -c DEBIAN_FRONTEND=noninteractive apt-get install -y -qq apt-transport-https ca-certificates cur
l >/dev/null
+ sh -c curl -fsSL "https://download.docker.com/linux/ubuntu/gpg" | apt-key add -qq - >/dev/null
Warning: apt-key output should not be parsed (stdout is not a terminal)
+ sh -c echo "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable" > /etc/apt/sou
rces.list.d/docker.list
+ sh -c apt-get update -qq >/dev/null
+ [ -n  ]
+ sh -c apt-get install -y -qq --no-install-recommends docker-ce >/dev/null
+ sh -c docker version
Client: Docker Engine - Community
 Version:           19.03.5
 API version:       1.40
 Go version:        go1.12.12
 Git commit:        633a0ea838
 Built:             Wed Nov 13 07:29:52 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.5
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.12
  Git commit:       633a0ea838
  Built:            Wed Nov 13 07:28:22 2019
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.2.10
  GitCommit:        b34a5c8af56e510852c35414db4c1f4fa6172339
 runc:
  Version:          1.0.0-rc8+dev
  GitCommit:        3e425f80a8c931f88e6d94a8c831b9d5aa481657
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
If you would like to use Docker as a non-root user, you should now consider
adding your user to the "docker" group with something like:

  sudo usermod -aG docker your-user
```

2. Checking the version of installed Docker

```
root@it74-OptiPlex-3050:/home/it74/Desktop# docker version
Client: Docker Engine - Community
 Version:           19.03.5
 API version:       1.40
 Go version:        go1.12.12
 Git commit:        633a0ea838
 Built:             Wed Nov 13 07:29:52 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.5
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.12
  Git commit:       633a0ea838
  Built:            Wed Nov 13 07:28:22 2019
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.2.10
  GitCommit:        b34a5c8af56e510852c35414db4c1f4fa6172339
 runc:
  Version:          1.0.0-rc8+dev
  GitCommit:        3e425f80a8c931f88e6d94a8c831b9d5aa481657
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
```

3. Running an image after pulling it from Docker Hub

```
root@it74-OptiPlex-3050:/home/it74/Desktop# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
5c939e3a4d10: Pull complete
c63719cdbe7a: Pull complete
19a861ea6baf: Pull complete
651c9d2d6c4f: Pull complete
Digest: sha256:8d31dad0c58f552e890d68bbfb735588b6b820a46e459672d96e585871acc110
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
root@it74-OptiPlex-3050:/home/it74# docker run -it -d ubuntu
1016d3b59140f8b14fee74e341c98c6d04e41a74e5e97355e5ac450b6419f5e1
root@it74-OptiPlex-3050:/home/it74#
```

4. Image Example

```
root@it74-OptiPlex-3050:/home/it74/Desktop# docker run docker/whalesay cowsay hello_mahendra
 _________________
< hello_mahendra >
 -----------------
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \    \        __/
          \____\______/
root@it74-OptiPlex-3050:/home/it74/Desktop#
```

5. Docker MySQL Image

```
root@it74-OptiPlex-3050:/home/it74/Desktop# docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
804555ee0376: Pull complete
c53bab458734: Pull complete
ca9d72777f90: Pull complete
2d7aad6cb96e: Pull complete
8d6ca35c7908: Pull complete
6ddae009e760: Pull complete
327ae67bbe7b: Pull complete
e039ec97e847: Pull complete
fbda57f041c1: Pull complete
f70ca9049b77: Pull complete
30ecde006e59: Pull complete
7def2a4ede73: Pull complete
Digest: sha256:f1df505c4c6e8eae599a0482e3bde3e761cd700c00cbc371a8161648a26817c0
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest
root@it74-OptiPlex-3050:/home/it74/Desktop#
```

6. Setting up Docker Services

```
root@it74-OptiPlex-3050:/home/it74/Desktop# sudo nano /etc/resolv.conf
root@it74-OptiPlex-3050:/home/it74/Desktop#
```

```
root@it74-OptiPlex-3050: /home/it74/Desktop
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3                    /etc/resolv.conf

# This file is managed by man:systemd-resolved(8). Do not edit.
#
# This is a dynamic resolv.conf file for connecting local clients to the
# internal DNS stub resolver of systemd-resolved. This file lists all
# configured search domains.
#
# Run "systemd-resolve --status" to see details about the uplink DNS servers
# currently in use.
#
# Third party programs must not access this file directly, but only through the
# symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a different way,
# replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported modes of
# operation for /etc/resolv.conf.
nameserver 8.8.8.8
nameserver 8.8.4.4
nameserver 127.0.0.53
options edns0
                              [ Read 20 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

```
root@it74-OptiPlex-3050:/home/it74/Desktop# sudo systemctl restart docker
root@it74-OptiPlex-3050:/home/it74/Desktop# sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: e
   Active: active (running) since Thu 2020-01-30 09:37:20 IST; 12s ago
     Docs: https://docs.docker.com
 Main PID: 22126 (dockerd)
    Tasks: 13
   CGroup: /system.slice/docker.service
           └─22126 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/contai

Jan 30 09:37:18 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:18.807
Jan 30 09:37:18 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:18.807
Jan 30 09:37:18 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:18.807
Jan 30 09:37:18 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:18.807
Jan 30 09:37:19 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:19.390
Jan 30 09:37:19 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:19.585
Jan 30 09:37:20 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:20.456
Jan 30 09:37:20 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:20.512
Jan 30 09:37:20 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:20.785
Jan 30 09:37:20 it74-OptiPlex-3050 systemd[1]: Started Docker Application Contai
lines 1-19/19 (END)...skipping...
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2020-01-30 09:37:20 IST; 12s ago
     Docs: https://docs.docker.com
 Main PID: 22126 (dockerd)
    Tasks: 13
   CGroup: /system.slice/docker.service
           └─22126 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jan 30 09:37:18 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:18.807267334+05:30" level=w
arning msg="Your kernel does not support cgroup rt runtime"
Jan 30 09:37:18 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:18.807271587+05:30" level=w
arning msg="Your kernel does not support cgroup blkio weight"
Jan 30 09:37:18 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:18.807275757+05:30" level=w
arning msg="Your kernel does not support cgroup blkio weight_device"
Jan 30 09:37:18 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:18.807388304+05:30" level=i
nfo msg="Loading containers: start."
Jan 30 09:37:19 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:19.390931353+05:30" level=i
nfo msg="Default bridge (docker0) is assigned with an IP address 172.17.0.0/16. Daemon option --bip c
an
Jan 30 09:37:19 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:19.585647196+05:30" level=i
nfo msg="Loading containers: done."
Jan 30 09:37:20 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:20.456246613+05:30" level=i
nfo msg="Docker daemon" commit=633a0ea838 graphdriver(s)=overlay2 version=19.03.5
Jan 30 09:37:20 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:20.512240921+05:30" level=i
nfo msg="Daemon has completed initialization"
Jan 30 09:37:20 it74-OptiPlex-3050 dockerd[22126]: time="2020-01-30T09:37:20.785862073+05:30" level=i
nfo msg="API listen on /var/run/docker.sock"
Jan 30 09:37:20 it74-OptiPlex-3050 systemd[1]: Started Docker Application Container Engine.
~
~
~
~
```

7. Configuring Proxy for Docker

```
root@it74-OptiPlex-3050:/# cd /etc/systemd/system/docker.service.d
root@it74-OptiPlex-3050:/etc/systemd/system/docker.service.d# nano http-proxy.conf
root@it74-OptiPlex-3050:/etc/systemd/system/docker.service.d# systemctl daemon-reload
root@it74-OptiPlex-3050:/etc/systemd/system/docker.service.d# systemctl restart docker
root@it74-OptiPlex-3050:/etc/systemd/system/docker.service.d#
```

```
root@it74-OptiPlex-3050: /etc/systemd/system/docker.service.d
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3                    http-proxy.conf

[Service]
Environment=&quot;HTTP_PROXY=http://USER:PASSWD@SERVER:PORT/&quot;
Environment=&quot;HTTPS_PROXY=http://USER:PASSWD@SERVER:PORT/&quot;
```

8. Displaying all containers in the system

```
root@it74-OptiPlex-3050:/home/it74# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
1016d3b59140        ubuntu              "/bin/bash"         About a minute ago  Up About a minute
                    friendly_newton
root@it74-OptiPlex-3050:/home/it74#

root@it74-OptiPlex-3050:/home/it74# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
             PORTS               NAMES
1016d3b59140        ubuntu              "/bin/bash"              About a minute ago  Up About a minu
te                              friendly_newton
05b6d9550ebe        docker/whalesay     "cowsay hello_mahend…"   32 minutes ago      Exited (0) 32 m
inutes ago                      magical_diffie
1eb0a18ef923        docker/whalesay     "cowsay hello_mahend…"   2 days ago          Exited (0) 2 da
ys ago                          funny_boyd
root@it74-OptiPlex-3050:/home/it74#
```

9. Container Operations: Executing, Running and Killing a Container

```
root@it74-OptiPlex-3050:/home/it74# docker exec -it 1016d3b59140 bash
root@1016d3b59140:/# exit
exit
root@it74-OptiPlex-3050:/home/it74# docker stop 1016d3b59140
1016d3b59140
root@it74-OptiPlex-3050:/home/it74# docker kill 1016d3b59140
Error response from daemon: Cannot kill container: 1016d3b59140: Container 1016d3b59140f8b14fee74e341
c98c6d04e41a74e5e97355e5ac450b6419f5e1 is not running
root@it74-OptiPlex-3050:/home/it74# docker commit 1016d3b59140 hshar/ubuntunew
sha256:c2ca18b6a692524f17a6dd4b1ab5204f53e2cd22cca267b54e8346023037848c
root@it74-OptiPlex-3050:/home/it74#
```

10. Logging into Docker Hub

```
root@it74-OptiPlex-3050:/home/it74# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, hea
d over to https://hub.docker.com to create one.
Username: msiddhesh
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@it74-OptiPlex-3050:/home/it74#
```

11. Displaying all docker images in the system

```
root@it74-OptiPlex-3050:/home/it74# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hshar/ubuntunew     latest              c2ca18b6a692        8 minutes ago       64.2MB
ubuntu              latest              ccc6e87d482b        2 weeks ago         64.2MB
mysql               latest              3a5e53f63281        2 weeks ago         465MB
hello-world         latest              fce289e99eb9        13 months ago       1.84kB
docker/whalesay     latest              6b362a9f73eb        4 years ago         247MB
root@it74-OptiPlex-3050:/home/it74# docker rmi c2ca18b6a692
Untagged: hshar/ubuntunew:latest
Deleted: sha256:c2ca18b6a692524f17a6dd4b1ab5204f53e2cd22cca267b54e8346023037848c
Deleted: sha256:4edde1b6ff8987cb899b66ea3b9a31c3682f119f765c5e3d4615fe414c9ad209
root@it74-OptiPlex-3050:/home/it74# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest              ccc6e87d482b        2 weeks ago         64.2MB
mysql               latest              3a5e53f63281        2 weeks ago         465MB
hello-world         latest              fce289e99eb9        13 months ago       1.84kB
docker/whalesay     latest              6b362a9f73eb        4 years ago         247MB
root@it74-OptiPlex-3050:/home/it74# docker-push hshar/ubuntunew
docker-push: command not found
root@it74-OptiPlex-3050:/home/it74# docker push hshar/ubuntunew
The push refers to repository [docker.io/hshar/ubuntunew]
An image does not exist locally with the tag: hshar/ubuntunew
root@it74-OptiPlex-3050:/home/it74# docker push ubuntu
The push refers to repository [docker.io/library/ubuntu]
f55aa0bd26b8: Layer already exists
1d0dfb259f6a: Layer already exists
21ec61b65b20: Layer already exists
43c67172d1d1: Layer already exists
[DEPRECATION NOTICE] registry v2 schema1 support will be removed in an upcoming release. Please conta
ct admins of the docker.io registry NOW to avoid future disruption. More information at https://docs.
docker.com/registry/spec/deprecated-schema-v1/
errors:
denied: requested access to the resource is denied
unauthorized: authentication required
```

12. Running an image named “Hello-World”

```
root@it74-OptiPlex-3050:/home/it74# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:9572f7cdcee8591948c2963463447a53466950b3fc15a247fcad1917ca215a2f
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Conclusion:

Experiment No. 6

Aim: To build and deploy web application on Docker.

Theory:

- Docker is an open source platform which can be used to run/test applications in an isolated environment on the host system. Docker could be defined as an implementation of container using the concept of images.
- Unlike VM which is a complete OS installed on host system managed by hypervisors and needs dedicated resources from the host machine, docker uses Linux Containers which directly interacts with the Linux kernel like any other process on Linux to request resources.
- The Apache HTTP Server, colloquially called Apache, is a Web server application notable for playing a key role in the initial growth of the World Wide Web. Originally based on the NCSA HTTPd server, development of Apache began in early 1995 after work on the NCSA code stalled. Apache quickly overtook NCSA HTTPd as the dominant HTTP server and has remained the most popular HTTP server in use since April 1996.

Apache Web Server Container:

1. Setup Docker and install all configurations
2. Pulling Postgres Image from Docker Hub

```
root@it75-OptiPlex-3050:/home/it75# docker pull postgres
Using default tag: latest
latest: Pulling from library/postgres
Digest: sha256:5181eccc7c903e4f1beffa89a735cb7ed72e0c81d6c34c471552c3fa8bff0858
Status: Image is up to date for postgres:latest
docker.io/library/postgres:latest
root@it75-OptiPlex-3050:/home/it75# docker images
REPOSITORY                TAG                 IMAGE ID            CREATED
     SIZE
postgres                  latest              cf879a45faaa        3 days ago
     394MB
ubuntu                    latest              ccc6e87d482b        3 weeks ago
     64.2MB
stuti220882/devopsttp1    latest              149a6fbe456a        7 weeks ago
     189MB
bushsk/ubuntu_1           latest              845c3411773d        7 weeks ago
     189MB
stuti220882/devopsttp     latest              a08d8adab7b6        7 weeks ago
     189MB
ubuntu                    <none>              775349758637        3 months ago
     64.2MB
docker/whalesay           latest              6b362a9f73eb        4 years ago
     247MB
root@it75-OptiPlex-3050:/home/it75#
```

3. Checking all pulled images

```
root@it75-OptiPlex-3050:/home/it75# docker images
REPOSITORY                TAG                 IMAGE ID            CREATED
     SIZE
postgres                  latest              cf879a45faaa        3 days ago
     394MB
ubuntu                    latest              ccc6e87d482b        3 weeks ago
     64.2MB
stuti220882/devopsttp1    latest              149a6fbe456a        7 weeks ago
     189MB
bushsk/ubuntu_1           latest              845c3411773d        7 weeks ago
     189MB
stuti220882/devopsttp     latest              a08d8adab7b6        7 weeks ago
     189MB
ubuntu                    <none>              775349758637        3 months ago
     64.2MB
docker/whalesay           latest              6b362a9f73eb        4 years ago
     247MB
```

4. Checking all existing containers

```
root@it75-OptiPlex-3050:/home/it75# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              P
ORTS                  NAMES
75cffad5daba        ubuntu:latest       "bash"              9 minutes ago       Up 9 minutes
                      admiring_heisenberg
740f24ef97da        test/test1          "bash"              21 minutes ago      Up 21 minutes       0
.0.0.0:8888->80/tcp   ubuntu
root@it75-OptiPlex-3050:/home/it75# docker exec -it 75cffad5daba service apache2 restart
 * Restarting Apache httpd web server apache2                                        AH00558: apache2: Cou
ld not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'Server
Name' directive globally to suppress this message
                                                                                       [ OK ]
root@it75-OptiPlex-3050:/home/it75#
```

5. Pull and run Ubuntu Image from Docker Hub
6. Create an apache server inside the container

```
root@it75-OptiPlex-3050:/home/it75# apt-get install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
apache2 is already the newest version (2.4.29-1ubuntu4.11).
You might want to run 'apt --fix-broken install' to correct these.
The following packages have unmet dependencies:
 owncloud-client : Depends: ocqt5125-libsqlite0 but it is not going to be installed
E: Unmet dependencies. Try 'apt --fix-broken install' with no packages (or specify a solution).
root@it75-OptiPlex-3050:/home/it75# cd /var/www/html
root@it75-OptiPlex-3050:/var/www/html# mv index.html index.backup
root@it75-OptiPlex-3050:/var/www/html# ls
index.backup  index.nginx-debian.html
root@it75-OptiPlex-3050:/var/www/html#

root@12db9f1eb108:/var/www/html# service apache2 restart
 * Restarting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17
.0.2. Set the 'ServerName' directive globally to suppress this message
                                                                                       [ OK ]
root@12db9f1eb108:/var/www/html# service apache2 status
 * apache2 is running
root@12db9f1eb108:/var/www/html#
```

7. In another terminal, check all running docker containers

```
root@it75-OptiPlex-3050: /home/it75
File  Edit  View  Search  Terminal  Help
it75@it75-OptiPlex-3050:~$ sudo su
root@it75-OptiPlex-3050:/home/it75# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
root@it75-OptiPlex-3050:/home/it75# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
7772f086726e        ubuntu:latest       "bash"              10 seconds ago
Up 3 seconds                            youthful_bassi
root@it75-OptiPlex-3050:/home/it75#
```

8. Commit an image using the previous image ID

```
root@it75-OptiPlex-3050:/home/it75# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
7772f086726e        ubuntu:latest       "bash"              About a minute ago  Up 53 seconds
                    youthful_bassi
root@it75-OptiPlex-3050:/home/it75# docker commit 7772f086726e test/test1
sha256:b8df2435d4969b4f524559d3867fb2a8887606438704ca0142fce3b4e7c667b3
root@it75-OptiPlex-3050:/home/it75#
root@it75-OptiPlex-3050:/home/it75#
root@it75-OptiPlex-3050:/home/it75# docker images
REPOSITORY                TAG                 IMAGE ID            CREATED             SIZE
test/test1                latest              b8df2435d496        15 seconds ago      64.2MB
postgres                  latest              cf879a45faaa        3 days ago          394MB
ubuntu                    latest              ccc6e87d482b        3 weeks ago         64.2MB
stuti220882/devopsttp1    latest              149a6fbe456a        7 weeks ago         189MB
bushsk/ubuntu_1           latest              845c3411773d        7 weeks ago         189MB
stuti220882/devopsttp     latest              a08d8adab7b6        7 weeks ago         189MB
ubuntu                    <none>              775349758637        3 months ago        64.2MB
docker/whalesay           latest              6b362a9f73eb        4 years ago         247MB
```

9. Execute new created image

```
root@it77-OptiPlex-3050:/home/it77# docker exec -it d2490e6e4b91 bash
root@d2490e6e4b91:/#  cd var/www/html
root@d2490e6e4b91:/var/www/html# ls
index.backup  index.html
root@d2490e6e4b91:/var/www/html# nano index.html
```

10. Run image at the required port

```
root@it77-OptiPlex-3050:/home/it77# docker run -itd -p 8888:80 --name d24906e4b9
1 bash
69e5ed1caecd6c56def7d83844ac494aa39a43f2ac8636d13ae6440dd62e823e
```

11. Create Firewall Exceptions for required port

```
root@it77-OptiPlex-3050:/home/it77# ufw allow 8888
Skipping adding existing rule
Skipping adding existing rule (v6)
```

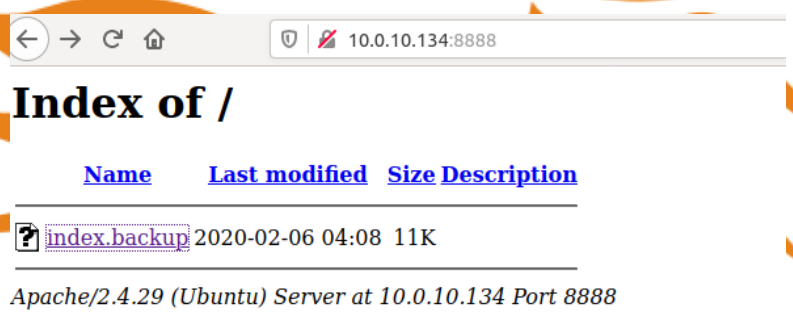
12. Execute the image on the apache server

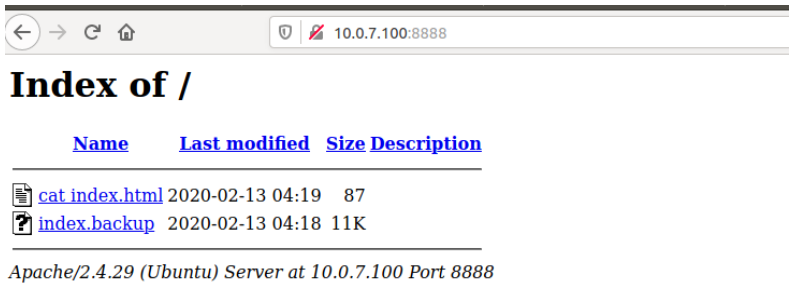
```
root@it77-OptiPlex-3050:/home/it77# docker exec -it d24906e4b91 service apache2
restart
 * Restarting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified doma
in name, using 172.17.0.3. Set the 'ServerName' directive globally to suppress t
his message
                                                                          [ OK ]
```

13. Run ifconfig to find neighbouring PC's IP address

```
root@it75-OptiPlex-3050:/home/it75# ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:5fff:fe7f:f605  prefixlen 64  scopeid 0x20<link>
        ether 02:42:5f:7f:f6:05  txqueuelen 0  (Ethernet)
        RX packets 37751  bytes 1531870 (1.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 51834  bytes 80385643 (80.3 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.9.174  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::d03f:49a4:f309:15ef  prefixlen 64  scopeid 0x20<link>
        ether 50:9a:4c:28:cb:dc  txqueuelen 1000  (Ethernet)
        RX packets 353843  bytes 149335166 (149.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 74419  bytes 7635103 (7.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

14. Enter neighbour's PC's IP address and port number on the browser





15. Login to Docker and push image to neighbour's PC

```
root@it77-OptiPlex-3050:/home/it77# docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@it77-OptiPlex-3050:/home/it77# docker tag bash bhavin1202/bash
root@it77-OptiPlex-3050:/home/it77# docker push bhavin1202/bash
The push refers to repository [docker.io/bhavin1202/bash]
9ce5ad584b57: Pushed
f55aa0bd26b8: Pushed
1d0dfb259f6a: Pushed
21ec61b65b20: Pushed
43c67172d1d1: Pushed
latest: digest: sha256:5b3c6961cdc4f27e81652ed3c67cfda15334356d5c7b45c9b36f7cb0e
70d606d size: 1364
```

16. Create folder on neighbour'ssystem and execute image in that system

```
root@it77-OptiPlex-3050:/home/it77# mkdir dockertut
root@it77-OptiPlex-3050:/home/it77# cd dockertut
root@it77-OptiPlex-3050:/home/it77/dockertut# pwd
/home/it77/dockertut
root@it77-OptiPlex-3050:/home/it77/dockertut# nano index.html
root@it77-OptiPlex-3050:/home/it77/dockertut# ls
index.html
root@it77-OptiPlex-3050:/home/it77/dockertut# nano Dockerfile
root@it77-OptiPlex-3050:/home/it77/dockertut# nano Dockerfile
root@it77-OptiPlex-3050:/home/it77# docker exec -it d2490e6e4b91 bash
root@d2490e6e4b91:/#  cd var/www/html
root@d2490e6e4b91:/var/www/html# ls
index.backup  index.html
root@d2490e6e4b91:/var/www/html# nano index.html
```

Conclusion:

Experiment 7

AIM : Puppet - Basic Installation and Setup on Ubuntu 18.04

THEORY:

Puppet is a configuration management tool that simplifies system administration. Puppet uses a client/server model in which your managed nodes, running a process called the Puppet *agent*, talk to and pull down configuration profiles from a Puppet *master*.

Puppet deployments can range from small groups of servers up to enterprise-level operations. This guide will demonstrate how to install Puppet 6.1 on three servers:

- A Puppet master running Ubuntu 18.04
- A managed Puppet Agent node running Ubuntu 18.04

Prerequisites

To follow this tutorial, you will need three Ubuntu 16.04 servers, each with a non-root user with sudo privileges. You can learn more about how to set up a user with sudo privileges in our Initial Server Setup with Ubuntu 16.04 guide.

One Puppet master

One server will be the Puppet master. The Puppet master will run Puppet Server, which is resource intensive and requires:

at least 4GB of memory

at least 2 CPU cores

To manage larger infrastructures, the Puppet master will require more resources.

Two Puppet agents

The other two servers will be Puppet agent nodes, managed by the Puppet master. We'll call them db1 and web1.

When these three servers are in place, you're ready to begin.

Step 1 — Configuring /etc/hosts

Puppet master servers and the nodes they manage need to be able to communicate with each other. In most situations, this will be accomplished using DNS, either configured on an externally hosted service or on self-hosted DNS servers maintained as part of the infrastructure.

On every machine

On each machine, edit the /etc/hosts file. At the end of the file, specify the Puppet master server as follows, substituting the IP address for your Puppet master:

sudo nano /etc/hosts

/etc/hosts

. . .

puppet_ip_address puppet

. . .

When you're done, save and exit.

Note: By default, Puppet agents will look for the Puppet master at puppet to make it easier to get Puppet set up. This means we must use the name puppet in /etc/hosts. If puppet does not resolve to the Puppet master, the agents will not be able to make contact without configuring the server value in the agent's puppet.conf.

Step 2 — Installing Puppet Server

Puppet Server is the software that pushes configuration from the Puppet master to the other servers. It runs only on the Puppet master; the other hosts will run the Puppet Agent.

Note: The Ubuntu package manager does contain packages for Puppet, but many administrators need to manage multiple operating systems and versions. In this case, working with the official Puppet Labs repositories can simplify administration by allowing you to maintain the same Puppet version on all systems.

We'll enable the official Puppet Labs collection repository with these commands:

curl -O https://apt.puppetlabs.com/puppetlabs-release-pc1-xenial.deb

sudo dpkg -i puppetlabs-release-pc1-xenial.deb

sudo apt-get update

When apt-get update is complete, ensuring that we'll be pulling from the Puppet Labs repository, we'll install the puppetserver package:

sudo apt-get install puppetserver

Press Y to proceed. Once installation is complete, and before we start the server, we'll take a moment to configure the memory.

Configure memory allocation

By default, Puppet Server is configured to use 2 GB of RAM. You can customize this setting based on how much free memory the master server has and how many agent nodes it will manage.

To customize it, open /etc/default/puppetserver:

sudo nano /etc/default/puppetserver

Then find the JAVA_ARGS line, and use the -Xms and -Xmx parameters to set the memory allocation. We'll increase ours to 3 gigabytes:

/etc/default/puppetserver

JAVA_ARGS="-Xms3g -Xmx3g -XX:MaxPermSize=256m"

Save and exit when you're done.

Open the firewall

When we start Puppet Server, it will use port 8140 to communicate, so we'll ensure it's open:

sudo ufw allow 8140

Next, we'll start Puppet server.

Start Puppet server

We'll use systemctl to start Puppet server:

sudo systemctl start puppetserver

This will take some time to complete.

Once we're returned to the command prompt, we'll verify we've succeeded since systemctl doesn't display the outcome of all service management commands:

sudo systemctl status puppetserver

We should see a line that says "active (running)" and the last line should look something like:

Output

Dec 07 16:27:33 puppet systemd[1]: Started puppetserver Service.

Now that we've ensured the server is running, we'll configure it to start at boot:

sudo systemctl enable puppetserver

With the server running, now we're ready to set up Puppet Agent on our two agent machines, db1 and web1.

Step 3 — Installing the Puppet Agent

The Puppet agent software must be installed on any server that the Puppet master will manage. In most cases, this will include every server in your infrastructure.

Note: The Puppet agent can run on all major Linux distributions, some UNIX platforms, and Windows. Installation instructions vary on each OS. Directions to install the Puppet agent on CentOS are available here, and you can find directions for the complete set of installation targets in the Puppet Reference Manual.

Enable the official Puppet Labs repository

First we'll enable the official Puppet Labs collection repository with these commands:

wget https://apt.puppetlabs.com/puppetlabs-release-pc1-xenial.deb

sudo dpkg -i puppetlabs-release-pc1-xenial.deb

sudo apt-get update

Install the Puppet agent package

Then, we'll install the puppet-agent package:

sudo apt-get install puppet-agent

We'll start the agent and enable it to start on boot:

sudo systemctl start puppet

sudo systemctl enable puppet

Finally, we'll repeat these steps on web1:

wget https://apt.puppetlabs.com/puppetlabs-release-pc1-xenial.deb

sudo dpkg -i puppetlabs-release-pc1-xenial.deb

sudo apt-get update

sudo apt-get install puppet-agent

sudo systemctl enable puppet

sudo systemctl start puppet

Now that both agent nodes are running the Puppet agent software, we will sign the certificates on the Puppet master.

Step 4 — Signing Certificates on Puppet Master

The first time Puppet runs on an agent node, it sends a certificate signing request to the Puppet master. Before Puppet Server will be able to communicate with and control the agent node, it must sign that particular agent node's certificate.

List current certificate requests

To list all unsigned certificate requests, run the following command on the Puppet master:

sudo /opt/puppetlabs/bin/puppet cert list

There should be one request for each host you set up, that looks something like the following:

Output:

"db1.localdomain" (SHA256) 46:19:79:3F:70:19:0A:FB:DA:3D:C8:74:47:EF:C8:B0:05:8A:06:50:2B:40:B3:B9:26:35:F6:96:17:85:5E:7C

"web1.localdomain" (SHA256) 9D:49:DE:46:1C:0F:40:19:9B:55:FC:97:69:E9:2B:C4:93:D8:A6:3C:B8:AB:CB:DD:E6:F5:A0:9C:37:C8:66:A0

A + in front of a certificate indicates it has been signed. The absence of a plus sign indicates our new certificate has not been signed yet.

Sign requests

To sign a single certificate request, use the puppet cert sign command, with the hostname of the certificate as it is displayed in the certificate request.

For example, to sign db1's certificate, you would use the following command:

sudo /opt/puppetlabs/bin/puppet cert sign db1.localdomain

Output similar to the example below indicates that the certificate request has been signed:

Output:

Notice: Signed certificate request for db.localdomain

Notice: Removing file Puppet::SSL::CertificateRequest db1.localdomain at '/etc/puppetlabs/puppet/ssl/ca/requests/db1.localdomain.pem'

The Puppet master can now communicate and control the node that the signed certificate belongs to. You can also sign all current requests at once.

We'll use the --all option to sign the remaining certificate:

sudo /opt/puppetlabs/bin/puppet cert sign --all

Now that all of the certificates are signed, Puppet can manage the infrastructure. You can learn more about managing certificates in the How to Manage Puppet 4 Certificates cheat sheet.

Step 5 — Verifying the Installation

Puppet uses a domain-specific language to describe system configurations, and these descriptions are saved to files called "manifests", which have a .pp file extension. You can learn more about these in the Getting Started with Puppet Code: Manifests and Modules guide, but for now we'll create a brief directive to verify that the Puppet Server can manage the Agents as expected.

We'll begin by creating the default manifest, site.pp, in the default location:

sudo nano /etc/puppetlabs/code/environments/production/manifests/site.pp

Conclusion :

Experiment 8

AIM : To configure and perform Software Configuration Management and provisioning using Puppet.

Add Modules to Configure Agent Nodes

The Puppet master and agent nodes are now functional, but they are not secure. Based on concepts from the How to Secure your Server guide, a limited user and a firewall should be configured. This can be done on all nodes through the creation of basic Puppet modules, shown below.

Note
This is not meant to provide a basis for a fully-hardened server, and is intended only as a starting point. Alter and add firewall rules and other configuration options, depending on your specific needs.

Puppet modules are Puppet's prescribed way of organizing configuration code to serve specific purposes, like installing and configuration an application. You can create custom modules, or you can download and use modules published on Puppet Forge.
Add a Limited User

To create a new limited user on your nodes, you will create and apply a new module called accounts. This module will employ the user resource.

1. From the **Puppet master**, navigate to the /etc/puppetlabs/code/environments/production/modules directory. When a managed node requests its configuration from the master, the Puppet server process will look in this location for your modules:
2. cd /etc/puppetlabs/code/environments/production/modules/
3. Create the directory for a new accounts module:
4. mkdir accounts
5. cd accounts
6. Create the following directories inside the accounts module:
7. mkdir {examples,files,manifests,templates}

Directory	Description
manifests	The Puppet code which powers the module

Directory	Description
files	Static files to be copied to managed nodes
templates	Template files to be copied to managed nodes that can e customized with variables
examples	Example code which shows how to use the module

8. **Note**
9. Review Puppet's Module fundamentals article for more information on how a module is structured.
10. Navigate to the manifests directory:
11. cd manifests
12. Any file which contains Puppet code is called a *manifest*, and each manifest file ends in .pp. When located inside a module, a manifest should only define one class. If a module's manifests directory has an init.pp file, the class definition it contains is considered the main class for the module. The class definition inside init.pp should have the same name as the module.
Create an init.pp file with the contents of the following snippet. Replace all instances of username with a username of your choosing:

accounts/manifests/init.pp

```
class accounts {

  user { 'username':
    ensure      => present,
    home        => '/home/username',
    shell       => '/bin/bash',
    managehome  => true,
    gid         => 'username',
  }

}
```

Option	Description

ensure	Ensures that the user exists if set to present, or does not exist if set to absent
home	The path for the user's home directory
managehome	Controls whether a home directory should be created when creating the user
shell	The path to the shell for the user
gid	The user's primary group

13. Although the class declares what the user's primary group should be, it will not create the group itself. Create a new file called groups.pp inside the manifests directory with the following contents. Replace username with your chosen username:

accounts/manifests/groups.pp

```
class accounts::groups {

  group { 'username':
    ensure  => present,
  }

}
```

14. Your accounts class can declare your new accounts::groups class for use within the accounts class scope. Open your init.pp in your editor and enter a new include declaration at the beginning of the class:

accounts/manifests/init.pp

```
class accounts {

  include accounts::groups

  # ...

}
```

15. The new user should have administrative privileges. Because we have agent nodes on both Debian- and Red Hat-based systems, the new user needs to be in the sudo group on Debian systems, and the wheel group on Red Hat systems.

This value can be set dynamically through the use of Puppet *facts*. The facts system collects system information about your nodes and makes it available in your manifests.

Add a selector statement to the top of your accounts class:

accounts/manifests/init.pp

```
class accounts {

  $rootgroup = $osfamily ? {
    'Debian'  => 'sudo',
    'RedHat'  => 'wheel',
    default   => warning('This distribution is not supported by the Accounts module'),
  }

  include accounts::groups

  # ...

}
```

This code defines the value for the $rootgroup variable by checking the value of $osfamily, which is one of Puppet's core facts. If the value for $osfamily does not match Debian or Red Hat, the default value will output a warning that the distribution selected is not supported by this module.

Note

The Puppet Configuration Language executes code from top to bottom. Because the user resource declaration will reference the $rootgroup variable, you must define $rootgroup *before* the user declaration.

16. Update the user resource to include the groups option as follows:

accounts/manifests/init.pp

```
# ...

user { 'username':
  ensure      => present,
  home        => '/home/username',
  shell       => '/bin/bash',
  managehome  => true,
  gid         => 'username',
  groups      => "$rootgroup",
}

# ...
```

The value "$rootgroup" is enclosed in double quotes " " instead of single quotes ' ' because it is a variable which needs to be interpolated in your code.

17. The final value that needs to be added is the user's password. Since we do not want to use plain text, the password should be supplied to Puppet as a SHA1 digest, which is supported by default. Generate a digest with the openssl command:

18. openssl passwd -1

You will be prompted to enter your password. A hashed password will be output. Copy this value to your clipboard.

19. Update the user resource to include the password option as follows; insert your copied password hash as the value for the option:

accounts/manifests/init.pp

```
# ...

user { 'username':
  ensure      => present,
  home        => '/home/username',
  shell       => '/bin/bash',
  managehome  => true,
  gid         => 'username',
  groups      => "$rootgroup",
  password    => 'your_password_hash',
}

# ...
```

Caution

The hashed password **must** be included in single quotes ' '.

20. After saving your changes, use the Puppet parser to ensure that the code is correct:

21. /opt/puppetlabs/bin/puppet parser validate init.pp

Any errors that need to be addressed will be logged to standard output. If nothing is returned, your code is valid.

22. Navigate to the examples directory and create another init.pp file:

23. cd ../examples

accounts/examples/init.pp

```
include accounts
```

24. While still in the examples directory, test the module:

25. /opt/puppetlabs/bin/puppet apply --noop init.pp

Note

The --noop parameter prevents Puppet from actually applying the module to your system and making any changes.

It should return:

```
Notice: Compiled catalog for puppet.example.com in environment production in 0.26 seconds
Notice: /Stage[main]/Accounts::Groups/Group[username]/ensure: current_value absent, should be present (noop)
Notice: Class[Accounts::Groups]: Would have triggered 'refresh' from 1 events
Notice: /Stage[main]/Accounts/User[username]/ensure: current_value absent, should be present (noop)
Notice: Class[Accounts]: Would have triggered 'refresh' from 1 events
Notice: Stage[main]: Would have triggered 'refresh' from 2 events
Notice: Finished catalog run in 0.02 seconds
```

26. Again from the examples directory, run puppet apply to make these changes to the Puppet master server:

27. /opt/puppetlabs/bin/puppet apply init.pp

Puppet will create your limited Linux user on your master.

28. Log out as root and log in to the Puppet master as your new user.

Edit SSH Settings

Although a new limited user has successfully been added to the Puppet master, it is still possible to login to the system as root. To properly secure your system, root access should be disabled.

Note

Because you are now logged in to the Puppet master as a limited user, you will need to execute commands and edit files with the user's sudo privileges.

1. Navigate to the files directory within the accounts module:
2. cd /etc/puppetlabs/code/environments/production/modules/accounts/files
3. Copy your system's existing sshd_config file to this directory:
4. sudo cp /etc/ssh/sshd_config .
5. Open the file in your editor (making sure that you open it with sudo privileges) and set the PermitRootLogin value to no:

accounts/files/sshd_config

```
PermitRootLogin no
```

6. Navigate back to the manifests directory:
7. cd ../manifests

8. Create a new manifest called ssh.pp. Use the file resource to replace the default SSH configuration file with one managed by Puppet:

accounts/manifests/ssh.pp

```
class accounts::ssh {

  file { '/etc/ssh/sshd_config':
    ensure  => present,
    source  => 'puppet:///modules/accounts/sshd_config',
  }

}
```

Note

The files directory is omitted from the source line because the files folder is the default location of files within a module. For more information on the format used to access resources in a module, refer to the official Puppet module documentation.

9. Create a second resource to restart the SSH service and set it to run whenever sshd_config is changed. This will also require a selector statement because the SSH service is named ssh on Debian systems and sshd on Red Hat systems:

accounts/manifests/ssh.pp

```
class accounts::ssh {

  $sshname = $osfamily ? {
    'Debian'  => 'ssh',
    'RedHat'  => 'sshd',
    default   => warning('This distribution is not supported by the Accounts module'),
  }

  file { '/etc/ssh/sshd_config':
    ensure  => present,
    source  => 'puppet:///modules/accounts/sshd_config',
    notify  => Service["$sshname"],
  }

  service { "$sshname":
    hasrestart  => true,
  }

}
```

Note

notify is one of Puppet's relationship metaparameters.

10. Include the accounts::ssh class within the accounts class in init.pp:
11.

accounts/manifests/init.pp

```
class accounts {

  # ...

  include accounts::groups
  include accounts::ssh

  # ...

}
```

The complete init.pp

The contents of your init.pp should now look like the following snippet:

accounts/manifests/init.pp

```
class accounts {

  $rootgroup = $osfamily ? {
    'Debian' => 'sudo',
    'RedHat' => 'wheel',
    default => warning('This distro not supported by Accounts module'),
  }

  include accounts::groups
  include accounts::ssh

  user { 'example':
    ensure  => present,
    home    => '/home/username',
    shell   => '/bin/bash',
    managehome  => true,
    gid     => 'username',
    groups  => "$rootgroup",
    password => 'your_password_hash'
  }

}
```

12. Run the Puppet parser to test the syntax of the new class, then navigate to the examples directory to test and run the update to your accounts class:

```
13. sudo /opt/puppetlabs/bin/puppet parser validate ssh.pp
14. cd ../examples
15. sudo /opt/puppetlabs/bin/puppet apply --noop init.pp
16. sudo /opt/puppetlabs/bin/puppet apply init.pp
```

Note

You may see the following line in your output when validating:

```
Error: Removing mount "files": /etc/puppet/files does not exist or is not a directory
```

This refers to a Puppet configuration file, not the module resource you're trying to copy. If this is the only error in your output, the operation should still succeed.

17. To ensure that the ssh class is working properly, log out of the Puppet master and then try to log in as root. You should not be able to do so.

Add and Configure IPtables

To complete this guide's security settings, the firewall needs to be configure on your Puppet master and nodes. The iptables firewall software will be used.

1. By default, changes to your iptables rules will not persist across reboots. To avoid this, install the appropriate package on your Puppet master and nodes:
   Ubuntu/Debian:

```
sudo apt install iptables-persistent
```

   CentOS 7:
   CentOS 7 uses firewalld by default as a controller for iptables. Be sure firewalld is stopped and disabled before starting to work directly with iptables:

```
sudo systemctl stop firewalld && sudo systemctl disable firewalld
sudo yum install iptables-services
```

2. On your Puppet master, install Puppet Lab's firewall module from the Puppet Forge:

3. sudo /opt/puppetlabs/bin/puppet module install puppetlabs-firewall

   The module will be installed in your /etc/puppetlabs/code/environments/production/modules directory.

4. Navigate to the manifests directory inside the new firewall module:

5. cd /etc/puppetlabs/code/environments/production/modules/firewall/manifests/

6. Create a file titled pre.pp, which will contain all basic networking rules that should be run first:

firewall/manifests/pre.pp

```
class firewall::pre {

  Firewall {
    require => undef,
  }

   # Accept all loopback traffic
  firewall { '000 lo traffic':
    proto       => 'all',
    iniface     => 'lo',
    action      => 'accept',
  }->

   #Drop non-loopback traffic
  firewall { '001 reject non-lo':
    proto       => 'all',
    iniface     => '! lo',
    destination => '127.0.0.0/8',
    action      => 'reject',
  }->

   #Accept established inbound connections
```

```
  firewall { '002 accept established':
    proto       => 'all',
    state       => ['RELATED', 'ESTABLISHED'],
    action      => 'accept',
  }->

  #Allow all outbound traffic
  firewall { '003 allow outbound':
    chain       => 'OUTPUT',
    action      => 'accept',
  }->

  #Allow ICMP/ping
  firewall { '004 allow icmp':
    proto       => 'icmp',
    action      => 'accept',
  }

  #Allow SSH connections
  firewall { '005 Allow SSH':
    dport    => '22',
    proto   => 'tcp',
    action  => 'accept',
  }->

  #Allow HTTP/HTTPS connections
  firewall { '006 HTTP/HTTPS connections':
    dport    => ['80', '443'],
    proto   => 'tcp',
    action  => 'accept',
  }

}
```

7. In the same directory, create post.pp, which will run any firewall rules that need to be input last:

firewall/manifests/post.pp

```
class firewall::post {

  firewall { '999 drop all':
    proto  => 'all',
    action => 'drop',
    before => undef,
  }

}
```

These rules will direct the system to drop all inbound traffic that is not already permitted in the firewall.

8. Run the Puppet parser on both files to check their syntax for errors:

9. sudo /opt/puppetlabs/bin/puppet parser validate pre.pp
10. sudo /opt/puppetlabs/bin/puppet parser validate post.pp

11. Navigate to the main manifests directory:

12. cd /etc/puppetlabs/code/environments/production/manifests

13. Create a file named site.pp inside /etc/puppetlabs/code/environments/production/manifests. This file is the main manifest for the Puppet server service. It is used to map modules, classes, and resources to the nodes that they should be applied to.

site.pp

```
node default {

}

node 'puppet.example.com' {

  include accounts

  resources { 'firewall':
    purge => true,
  }

  Firewall {
    before        => Class['firewall::post'],
    require       => Class['firewall::pre'],
  }

  class { ['firewall::pre', 'firewall::post']: }

  firewall { '200 Allow Puppet Master':
    dport         => '8140',
    proto         => 'tcp',
    action        => 'accept',
  }

}
```

14. Run the site.pp file through the Puppet parser to check its syntax for errors. Then, test the file with the --noop option to see if it will run:

15. sudo /opt/puppetlabs/bin/puppet parser validate site.pp

16. sudo /opt/puppetlabs/bin/puppet apply --noop site.pp

If successful, run puppet apply without the --noop option:

sudo /opt/puppetlabs/bin/puppet apply site.pp

17. Once Puppet has finished applying the changes, check the Puppet master's iptables rules:

18. sudo iptables -L

It should return:

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             /* 000 lo traffic */
REJECT     all  --  anywhere             127.0.0.0/8          /* 001 reject non-lo */ reject-with icmp-port-
unreachable
ACCEPT     all  --  anywhere             anywhere             /* 002 accept established */ state
RELATED,ESTABLISHED
ACCEPT     icmp --  anywhere             anywhere             /* 004 allow icmp */
ACCEPT     tcp  --  anywhere             anywhere             multiport ports ssh /* 005 Allow SSH */
ACCEPT     tcp  --  anywhere             anywhere             multiport ports http,https /* 006
HTTP/HTTPS connections */
ACCEPT     tcp  --  anywhere             anywhere             multiport ports 8140 /* 200 Allow Puppet
Master */
DROP       all  --  anywhere             anywhere             /* 999 drop all */

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             /* 003 allow outbound */
```

Apply Modules to the Agent Nodes

Now that the accounts and firewall modules have been created, tested, and run on the Puppet master, it is time to apply them to your managed nodes.

1. On the **Puppet master**, navigate to /etc/puppetlabs/code/environments/production/manifests:
2. cd /etc/puppetlabs/code/environments/production/manifests
3. Update site.pp to declare the modules, classes, and resources that should be applied to each managed node:

site.pp

```
node default {

}

node 'puppet.example.com' {
  # ...
}

node 'puppet-agent-ubuntu.example.com' {

  include accounts

  resources { 'firewall':
    purge => true,
  }

  Firewall {
    before        => Class['firewall::post'],
    require       => Class['firewall::pre'],
  }

  class { ['firewall::pre', 'firewall::post']: }

}

node 'puppet-agent-centos.example.com' {

  include accounts

  resources { 'firewall':
    purge => true,
  }

  Firewall {
    before        => Class['firewall::post'],
    require       => Class['firewall::pre'],
  }

  class { ['firewall::pre', 'firewall::post']: }

}
```

4. By default, the Puppet agent service on your managed nodes will automatically check with the master once every 30 minutes and apply any new configurations from the master. You can also manually invoke the Puppet agent process in-between automatic agent runs.

   Log in to each managed node (as root) and run the Puppet agent:

```
/opt/puppetlabs/bin/puppet agent -t
```

5. To ensure the Puppet agent worked:
   - Log out from your root SSH session and log back in as the limited user that was created.
   - Check the node's firewall rules:

```
sudo iptables -L
```

Conclusion:

Aim: To configure and perform Software Configuration Management and provisioning using Ansible

Theory:

How to install Ansible on Ubuntu 18.04
Ansible is a free and open source automation engine that removes the workload associated with repetitive tasks. It allows you to comfortably simplify your workload by automating different tasks in your IT environment. It makes use of SSH protocol to retrieve information from remote machines and manage them. In this guide, we will show you how you can install Ansible on Ubuntu 18.04 server.

Prerequisites;
●Ubuntu 18.04 server node with 2 client Nodes for testing out. In this case, we are using ubuntu 18.04 for client node as well
Ubuntu 18.04 – Ansible Master Ubuntu 18.04 – client_1

●OpenSSH installed on all the nodes (Ansible Master and client_1)

To install and enable SSH on your Ubuntu system (Ansible Master and client_1) complete the following steps on both server and client machines:
Open your terminal either by using the Ctrl+Alt+T keyboard shortcut or by clicking on the terminal icon and install the openssh-server package by typing:

Enter the password when prompted and enter Y to continue with the installation.

Once the installation is completed, the SSH service will start automatically. To verify that the installation was successful and SSH service is running type the following command which will print the SSH server status:

You should see something like Active: active (running) :

Press q to get back to the command line prompt.

Ubuntu comes with a firewall configuration tool called UFW. If the firewall is enabled on your system, make sure to open the SSH port:

Now that SSH is installed and running on your Ubuntu system you can connect to it via SSH from any remote machine. Linux and macOS systems have SSH clients installed by default. If you want to connect from a Windows machine then you can use an SSH client such as PuTTY.

Connecting to SSH Over LAN

To connect to your Ubuntu machine over LAN you only need to enter the following command:

onmaster# ssh username@ip_address_of_client

Change the username with the actual user name and ip_address with the IP Address of the Ubuntu machine where you installed SSH.

If you don't know your IP address you can easily find it using the ip command:

$ip a

As you can see from the output, the system IP address is 192.168.121.111.

Once you've found the IP address, go back to the remote machine you're trying to log in with normal user and enter the following command:

onmaster# ssh normaluser_account_name_on_client@192.168.121.111

When you connect through SSH for the first time, you will see a message looking something like this:

The authenticity of host '192.168.121.111 (192.168.121.111)' can't be established.

ECDSA key fingerprint is SHA256:Vybt22mVXuNuB5unE++yowF7lgA/9/2bLSiO3qmYWBY.

Are you sure you want to continue connecting (yes/no)? yes

Type yes and you'll be prompted to enter your password.

Warning: Permanently added '192.168.121.111' (ECDSA) to the list of known hosts.

linuxize@192.168.121.111's password:

Once you enter the password you will be greeted with a message similar to the one below. Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-33-generic x86_64)

* Documentation: https://help.ubuntu.com

* Management: https://landscape.canonical.com

* Support: https://ubuntu.com/advantage

..

You are now logged in to your Ubuntu machine.

Allow SSH root login on Ubuntu 18.04 Bionic Beaver Linux- Ansible client systems (client_1)
Set Root Password

By default Ubuntu 18.04 Bionic Beaver installation comes with unset root password. To set root password open up terminal and execute the following linux command. When prompted enter your current user password and new root password:

Enable SSH root login

By default SSH root login is disabled. Any attempt to ssh as root user will result in the following error message:

Restart SSH server to apply changes:

client_1$ sudo service ssh restart

SSH login as root

Your server now allows SSH login as root user. Use the password you set in the first step:

Welcome to Ubuntu Bionic Beaver (GNU/Linux 4.13.0-25-generic x86_64)

Step 1: Installing Ansible
We are first going to update and upgrade our system repositories using the commands below

Now that our system repositories are up to date, we can proceed to install ansible. We are going to manually append the Ansible PPA to the server. To achieve this, run the following command

onmaster# apt-add-repository ppa:ansible/ansible

Sample output

Next, we are going to update the system repositories with the command

onmaster# apt-get update -y

Sample output

Now, we are going to install Ansible using the command

onmaster# apt-get install ansible -y

Sample output

Upon successful installation of Ansible, you can verify the version of ansible installed by running

onmaster# ansible --version

Sample output

Step 2: Configuring Ansible
Next, you will need to define your client system which you want to manage in Ansible hosts file. You can do this by editing /etc/ansible/hosts file:

Next, we need to define the client systems that we want to manage in Ansible hosts file located in /etc/ansible/hosts file.

Open the /etc/ansible/hosts file

onmaster# nano /etc/ansible/hosts

Add the following line so that we can manage our client systems

Step 3: Configuring SSH keys for client systems
Since Ansible uses SSH protocol to communicate with client systems, we are going to configure SSH passwordless authentication.

First, generate SSH keys using the command

onmaster# ssh-keygen
When prompted, simply hit Enter
Sample Output

Next, copy the key from the server to the client systems as shown Example: ansible-server$ ssh-copy-id -i ~/.ssh/id_rsa.pub root@host-IP
onmaster# ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.121.111

Sample Output

Similarly, do the same for any other client systems

Step 4: Using Ansible
At this point, we have successfully installed and configured Ansible to work with our one client. To ping our clients defined in /etc/ansible/hosts client using ansible, use the command
onmaster# ansible -m ping client_1

If you have multiple servers in your environment, you can run the command below to ensure that they are running

onmaster# ansible -m ping all

Step 5 — Running Ad-Hoc Commands (Optional)
After confirming that your Ansible control node is able to communicate with your hosts, you can start running ad-hoc commands and playbooks on your servers.

Any command that you would normally execute on a remote server over SSH can be run with Ansible on the servers specified in your inventory file. As an example, you can check disk usage on all servers with:

onmaster# ansible all -a "df -h" -u root

Output

server1 | CHANGED | rc=0 >>
Filesystem Size Used Avail Use% Mounted on udev 3.9G 0 3.9G 0% /dev
tmpfs 798M 624K 798M 1% /run
/dev/vda1 155G 2.3G 153G 2% /

tmpfs 3.9G 0 3.9G 0% /dev/shm
tmpfs 5.0M 0 5.0M 0% /run/lock
tmpfs 3.9G 0 3.9G 0% /sys/fs/cgroup
/dev/vda15 105M 3.6M 101M 4% /boot/efi tmpfs 798M 0 798M 0% /run/user/0

To check the status of SSH in our first Client system run

onmaster# ansible -m shell -a 'systemctl status sshd' client_1

Sample Output

To display hidden files in the root directory on our Client_1 system, run

onmaster# ansible -m shell -a 'ls -al /root' client_1

Sample Output

If you need to copy a file to multiple destinations rapidly, you can use the copy module in ansible which uses SCP. So the command and its output look like below:

onmaster# ansible -m copy -a "src=/etc/ansible/hosts dest=/tmp/testfile" client_1
Explanation:

1. Copy module defined
2. Module arguments, in this case, are source absolute path and destination absolute path.
3. Ansible command output reflecting the success of the copy command and other details like the sha1 or md5 checksums for file integrity check and metadata like owner, size, or permissions.

It is effortless to have a package installed on a bunch of servers. Ansible has several modules that interact with used installers, like yum, apt, dnf, etc.

For example, here's how we can use the apt module to install the latest version of curl on all the servers in your inventory:

```
onmaster# ansible all -m apt -a "name=curl state=latest" -u root
```

You can also target individual hosts, as well as groups and subgroups, when running Ansible commands.

This confirms that Ansible is working and we are able to manage our client systems from a central point.

Conclusion :

Experiment 10

Aim: To study and perform Infrastructure monitoring using NagiOS.

Theory:

What is Nagios?

Nagio is a free to use open source software tool for continuous monitoring. It helps you to monitor system, network, and infrastructure. It is used for continuous monitoring of systems, applications, service and business process in a DevOps culture.

Nagios runs plugins stored on the same server. It plugin's connects with a host or another server on your network or the Internet. Therefore, in the case of failure Nagios core can alert the technical staff about the issues. So that, your technical team performs the recovery process before outage in the business processes.

Why We Need Nagios?

Here, are Important reasons to use Nagios monitoring tool are:

- Detects all types of network or server issues
- Helps you to find the root cause of the problem which allows you to get the permanent solution to the problem
- Active monitoring of your entire infrastructure and business processes
- Allows you to monitors and troubleshoot server performance issues
- Helps you to plan for infrastructure upgrades before outdated systems create failures
- You can maintain the security and availability of the service
- Automatically fix problems in a panic situation

History of Nagios

1996-Ethan Galstad uses the ideas and architecture of his earlier work to begin building a new application which runs under Linux OS

1999-The plugins that were which were originally distributed as a part of the NetSaint distribution are soon as a separate Nagios Plugins project

2002- Ethan renames the project to "Nagios" because of trademark issues with the name "NetSaint."

2005- Nagios becomes SourceForge.net Project of the Month in June

2009-Nagios Enterprises releases its first commercial version, Nagios XI

2012-Nagios again renamed as Nagios Core

2016-Nagios core surpasses 7,500,000 downloads directly from SourceForge.net website

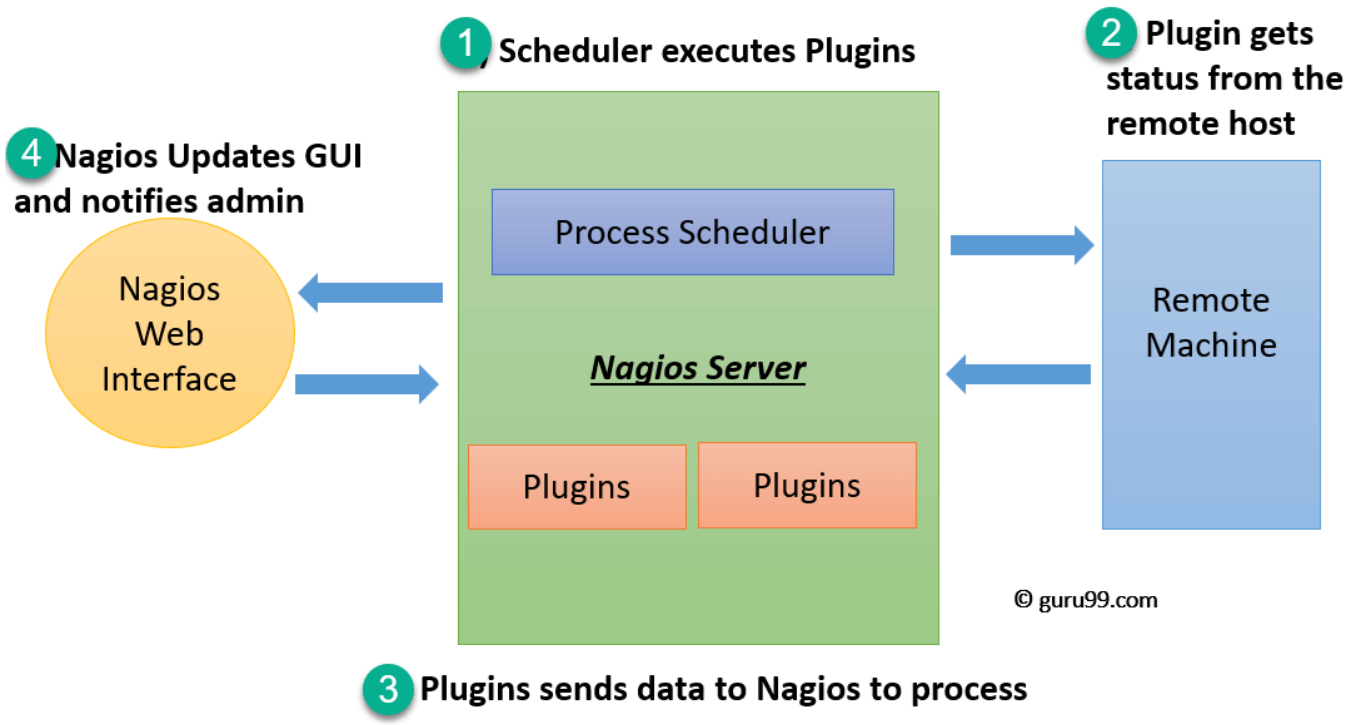
Features of Nagios

Following are the important features of Nagios:

- Relatively scalable, Manageable, and Secure
- Good log and database system
- Informative and attractive web interfaces
- Automatically send alerts if condition changes
- If the services are running fine, then there is no need to do check that host is an alive
- Helps you to detect network errors or server crashes
- You can troubleshoot the performance issues of the server.
- The issues, if any, can be fixed automatically as they are identified during the monitoring process
- You can monitor the entire business process and IT infrastructure with a single pass
- The product's architecture is easy writing new plugins in the language of your choice
- Nagios allows you to read its configuration from an entire directory which helps you to decide how to define individual files
- Utilizes topology to determine dependencies
- Monitor network services like HTTP, SMTP, HTTP, SNMP, FTP, SSH, POP, etc.
- Helps you to define network host hierarchy using parent hosts
- Ability to define event handlers which runs during service or host events for proactive problem resolution
- Support for implementing redundant monitoring hosts

Nagios Architecture

Nagios is a client-server architecture. Usually, on a network, a Nagios server is running on a host, and plugins are running on all the remote hosts which should be monitored.

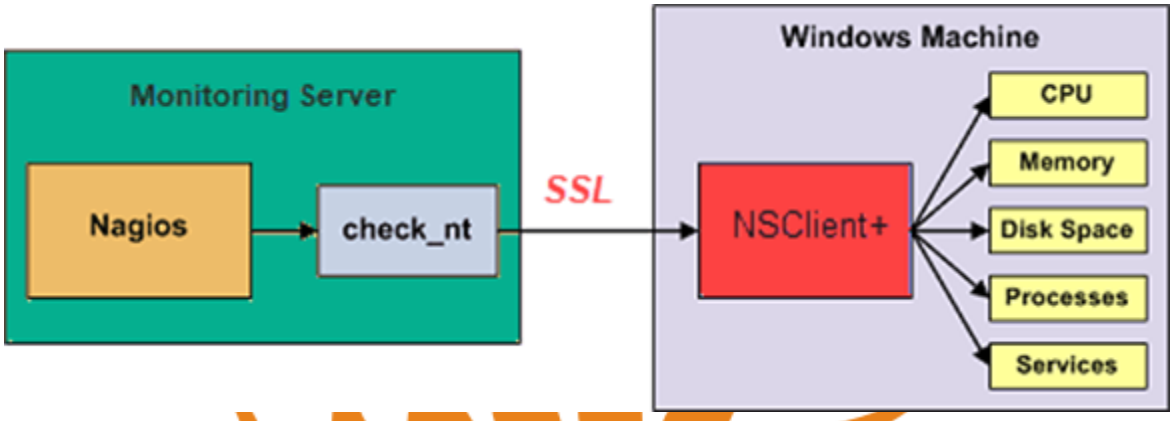


1. The scheduler is a component of server part of Nagios. It sends a signal to execute the plugins at the remote host.
2. The plugin gets the status from the remote host
3. The plugin sends the data to the process scheduler
4. The process scheduler updates the GUI and notifications are sent to admins

Plugins:

Nagios plugins provide low-level intelligence on how to monitor anything and everything with Nagios Core. Plugins operate acts as a standalone application, but they are designed to be executed by Nagios Core. It connects to Apache that is controlled by CGI to display the result. Moreover, a database connected to Nagios to keep a log file.

How do plugins work?



Consider the above example-

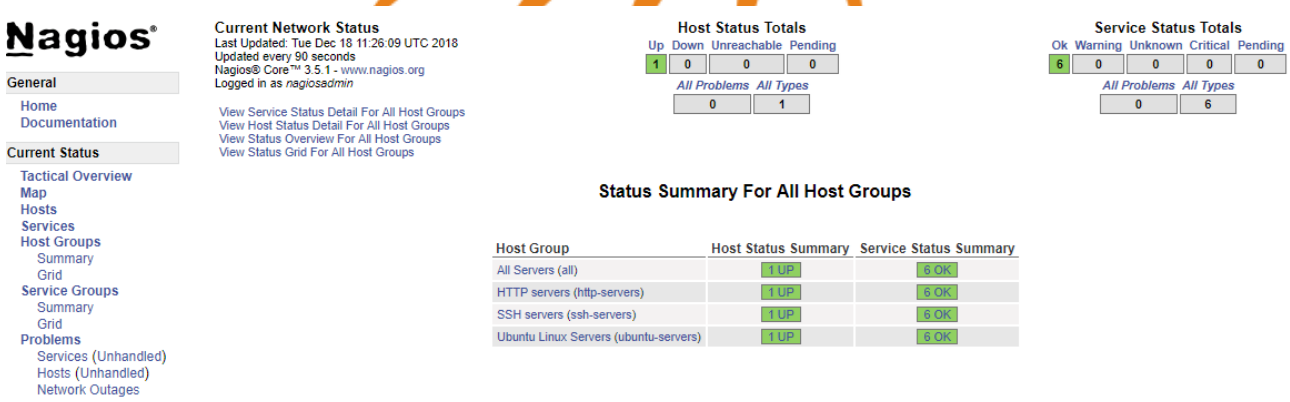
- Check_nt is a plugin to monitor a windows machine which is mostly available in the monitoring server
- NSClinet++ should be installed in every Windows machine that you wants to monitor
- There is an SSL connection between the server and the host which continuously exchange information with each other

Likewise, NRPE(Nagios Remote plug-in Executor) and NSCA plugins are used to monitor Linux and Mac OS X respectively.

GUI:

An interface of Nagios is used to display in web pages generated by CGI. It can be buttons to green or red, sound, graph, etc.

When the soft alert is raised many times, a hard alert is raised, then the Nagios server sends a notification to the administrator.

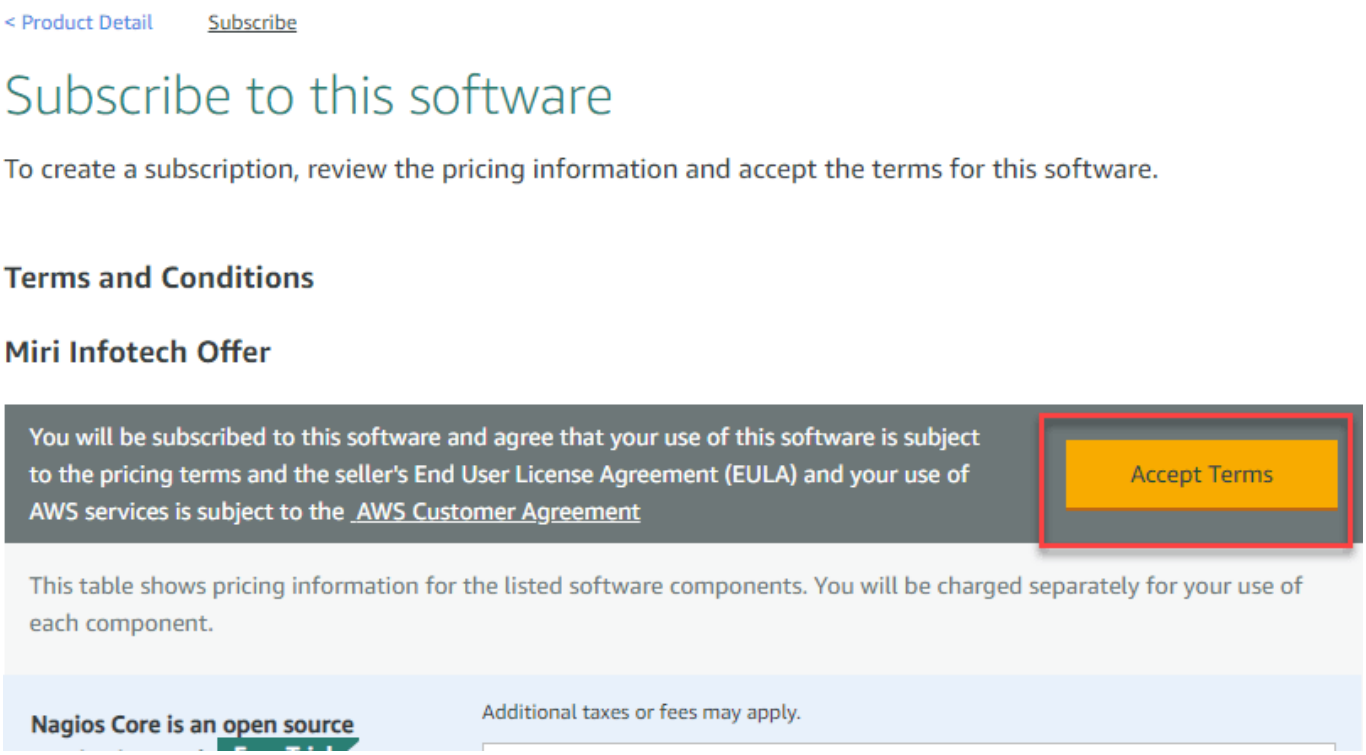


Nagios GUI

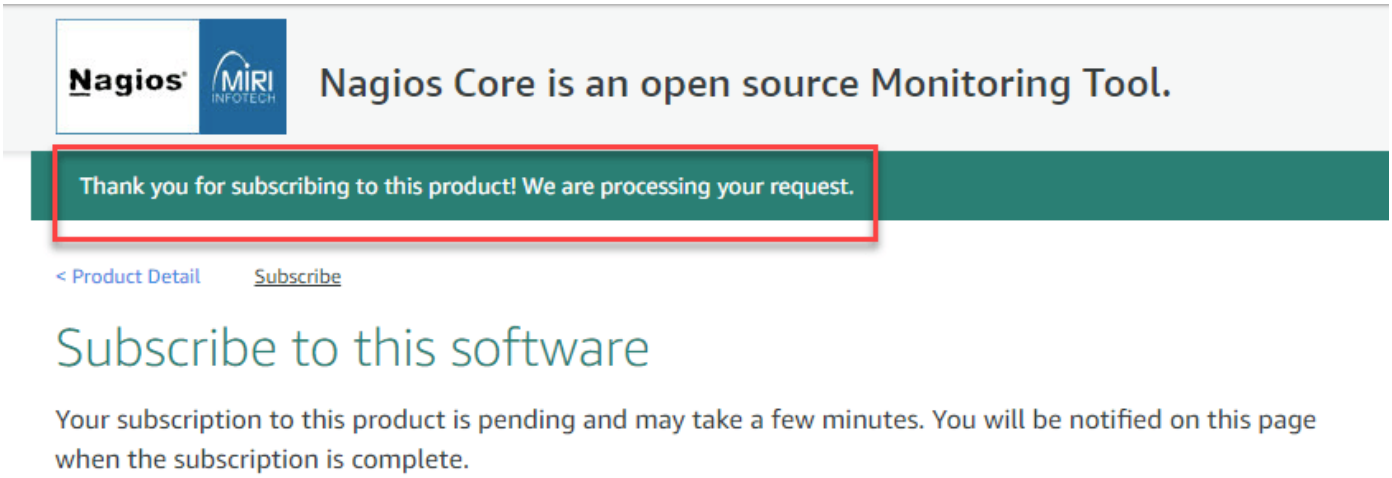
Install Nagios at AWS

Step 1) Got to https://aws.amazon.com/marketplace/pp/B0773T3529 and click Continue to Subscribe

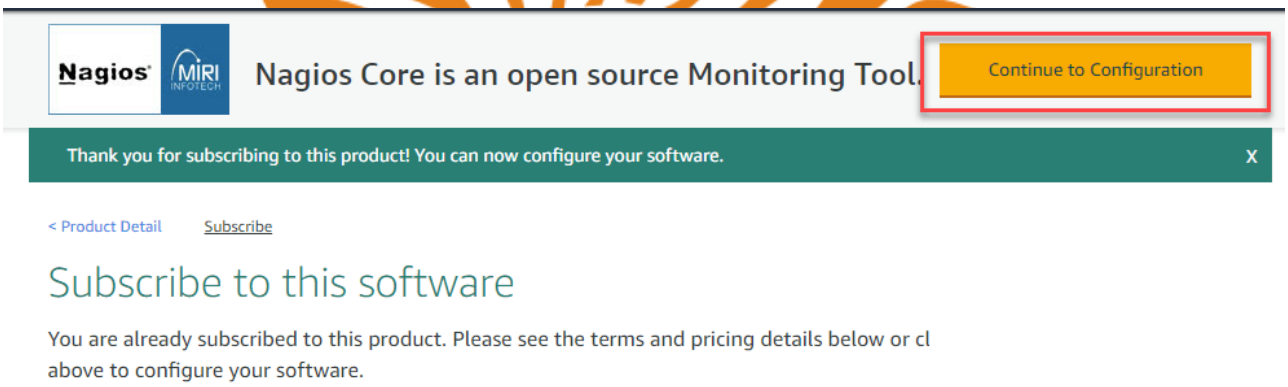
Step 2) Accept Terms



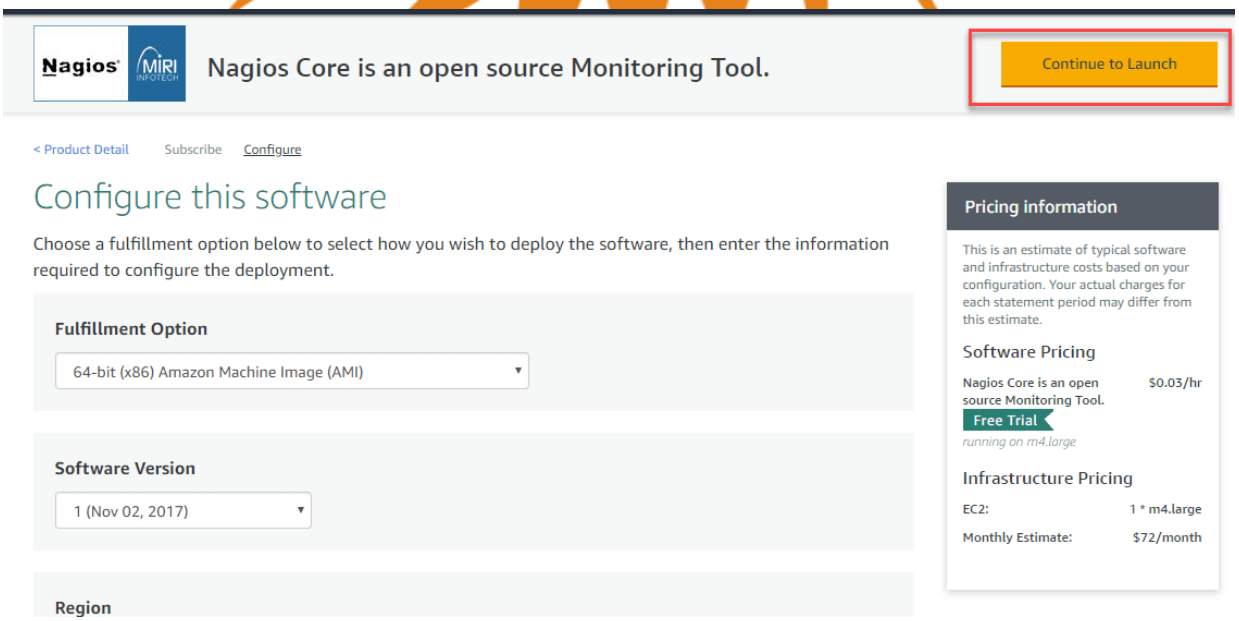
Step 3) You will see subscription pending message



Step 4) Refresh the same page after a few minutes and click "Continue to Configuration



Step 5) Keep the settings default and click Continue to Launch



Step 6) Review the settings. Create a new Key and click launch

Launch this software

Review your configuration and choose how you wish to launch the software.

Configuration Details

Fulfillment Option	64-bit (x86) Amazon Machine Image (AMI) Nagios Core is an open source Monitoring Tool. *running on m4.large*
Software Version	1
Region	US East (N. Virginia)

Usage Instructions

Choose Action

Launch from Website

Choose this action to launch from this website

EC2 Instance Type

m4.large

Memory: 8 GiB
CPU: 6.5 EC2 Compute Units (2 Virtual cores with 3.25 Units each)
Storage: EBS storage only
Network Performance: Moderate

VPC Settings

* indicates a default vpc

vpc-2b469d51

Create a VPC in EC2

Subnet Settings

subnet-d9fb29e7 (172.31.48.0/20)

Create a subnet in EC2
(Ensure you are in the selected VPC above)

Security Group Settings

A security group acts as a firewall that controls the traffic allowed to reach one or more instances. You can create a new security group based on seller-recommended settings or choose one of your existing groups. Learn more

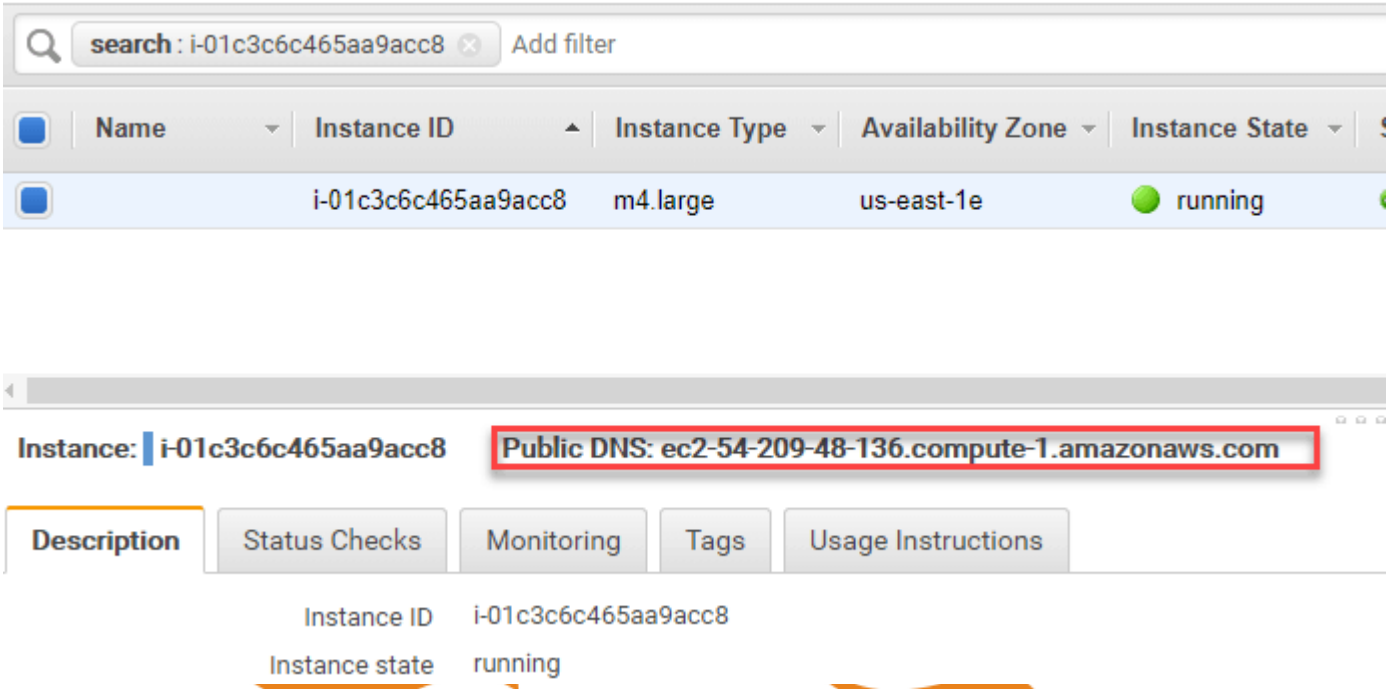
default

Create New Based On Seller Settings

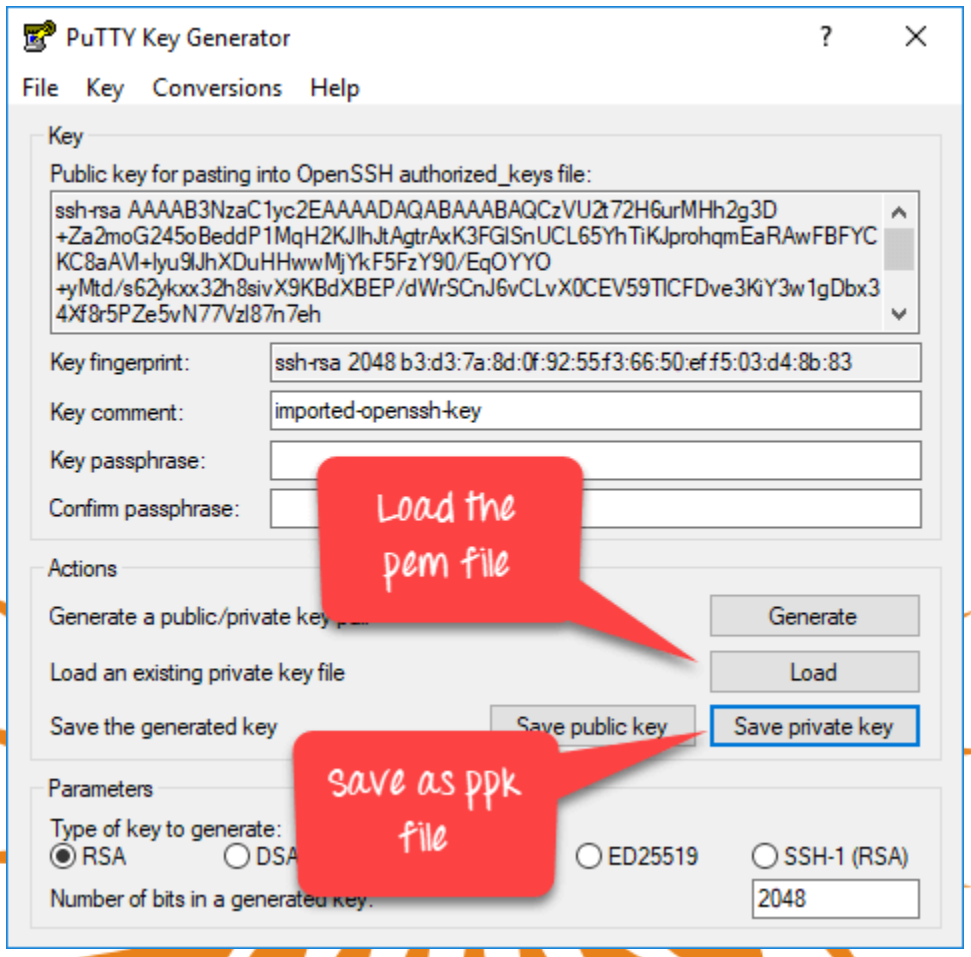
Key Pair Settings

To ensure that no other person has access to your software, the software installs on an EC2 instance with an EC2 key pair

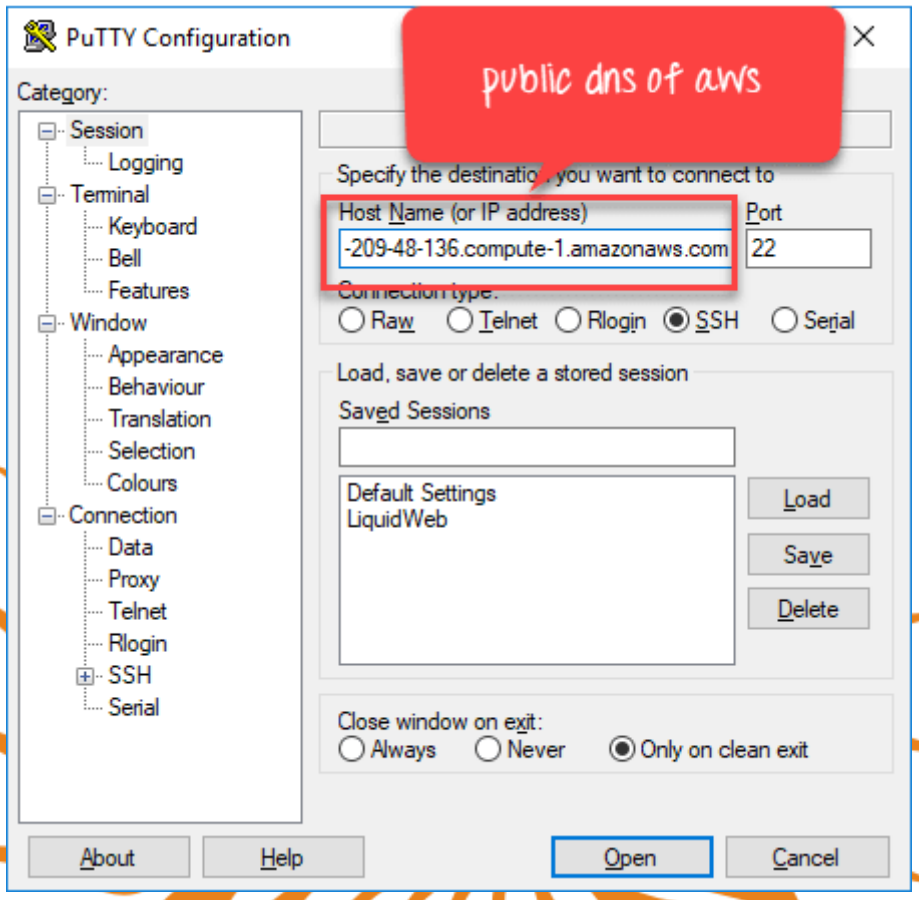
Step 7) Note the public DNS of your instance



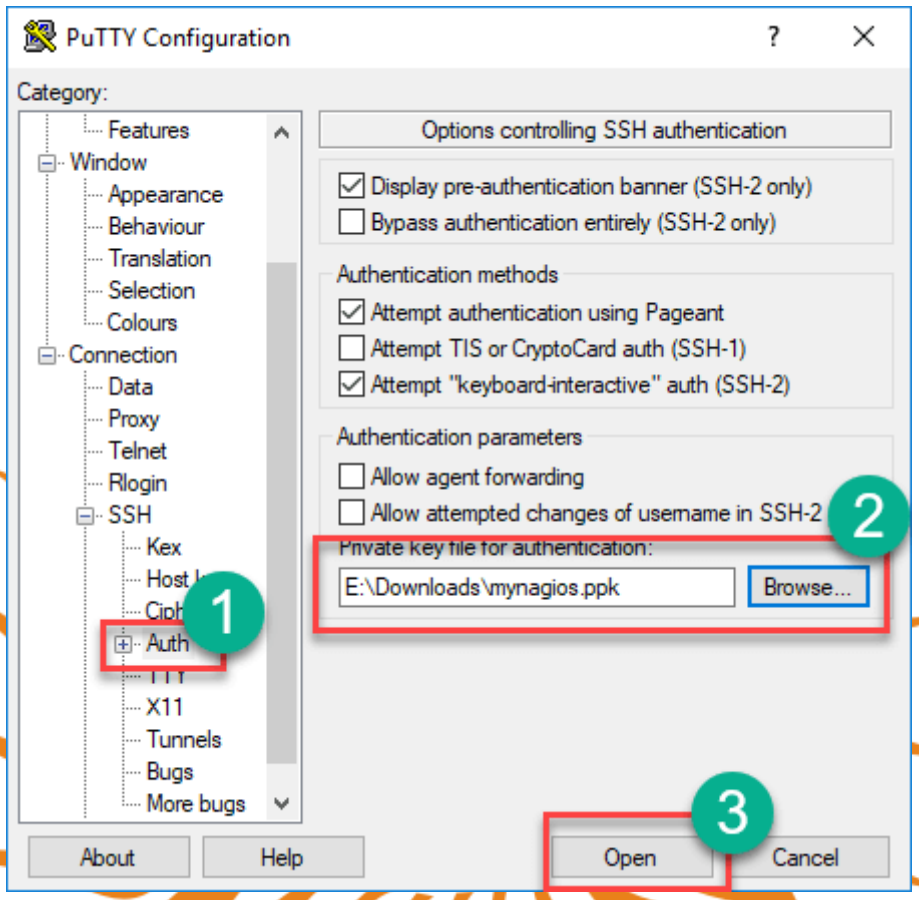
Step 8) In your windows machine, use the tool putty generator to convert pem file to ppk



Step 9) In putty, enter the public DNS



Step 10) In Auth section, enter the ppk key and click open



Step 11) In terminal,

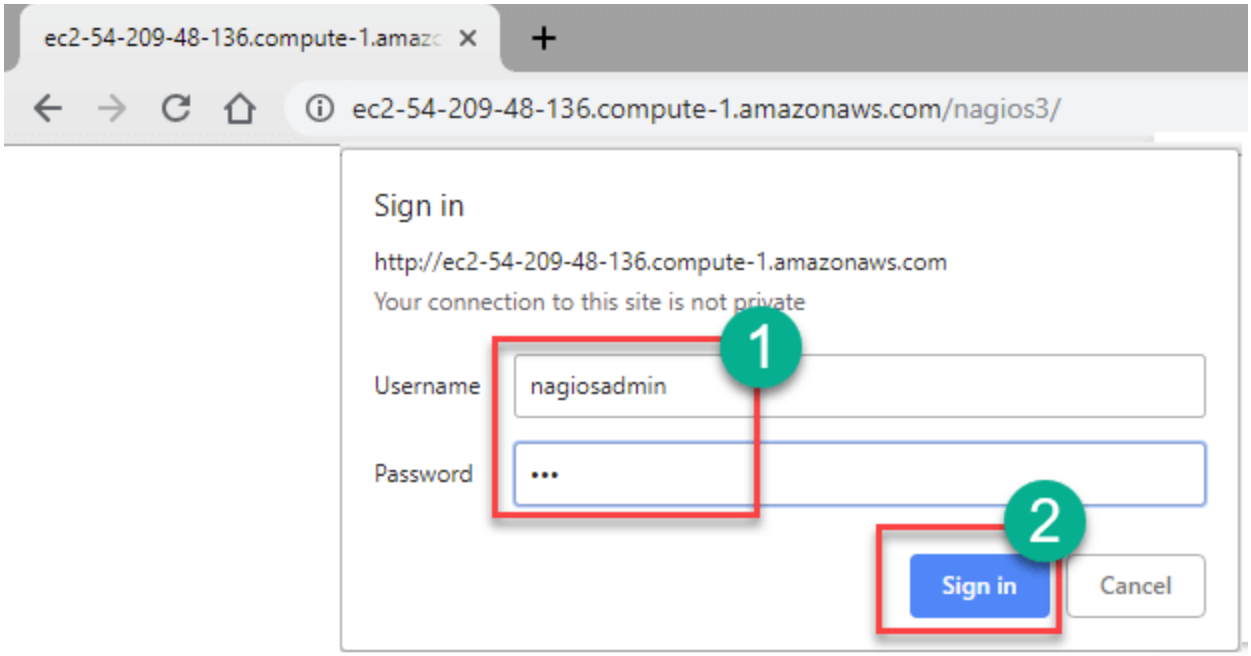
1. **Enter login name as ubuntu**
2. **Run this command** sudo htpasswd -c /etc/nagios3/htpasswd.users nagiosadmin
3. **Enter a new password of your choice**

```
ubuntu@ip-172-31-61-62: ~
login as: ubuntu
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-1035-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

239 packages can be updated.
133 updates are security updates.


*** System restart required ***
Last login: Fri Sep 22 09:07:22 2017 from 182.76.40.182
ubuntu@ip-172-31-61-62:~$ sudo htpasswd -c /etc/nagios3/htpasswd.users nagiosadm
in
New password:
Re-type new password:
Adding password for user nagiosadmin
ubuntu@ip-172-31-61-62:~$
```

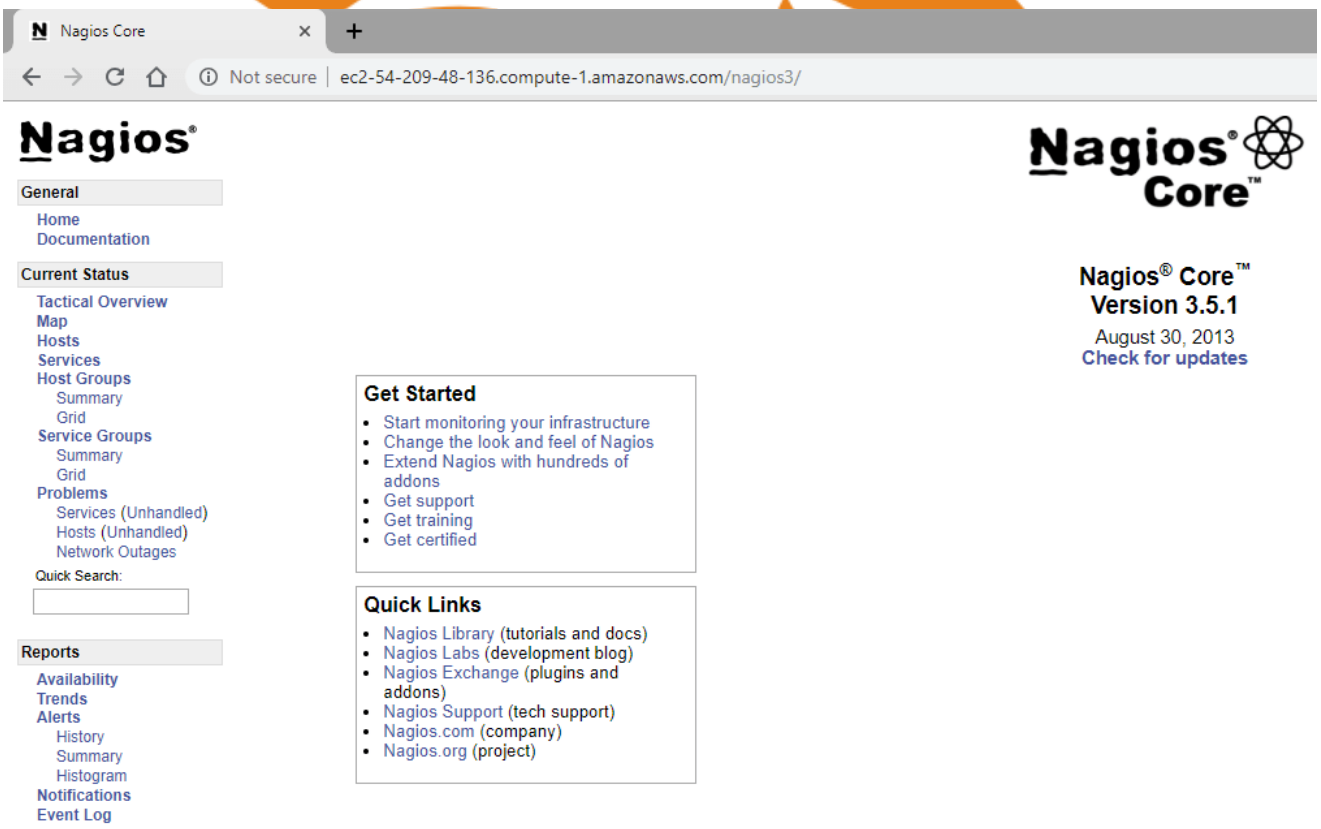
Step 12) In your browser, Go to location http://<Public DNS>/nagios3 in my case http://ec2-54-209-48-136.compute-1.amazonaws.com/nagios3/.

Enter Username: nagiosadmin

pass: set in the previous step



Step 13) Nagios Loads



Application of Nagios

Nagios is a health check & monitoring system for a typical Data Centre, comprises all type of equipment's such as:

- Server & Network Nodes
- Application monitoring from a single console
- Application Monitoring with transaction-level insights
- Monitor Middleware & Messaging Components
- Customizable Reports and Dashboards
- UPS Backup System
- Bio-Metric Identification System
- Temperature & Humidity Control System (Sensing Mechanism)
- CCTV/NVR System
- Storage Subsystem (NAS&SAN)

Disadvantages of Using Nagios

- Important feature like wizards or interactive dashboard are only available on Nagios XI, which is quite an expensive tool
- Nagios core has a confusing interface
- There're many configuration files which are very hard to configure for users
- Nagios can't monitor network throughput
- The tool not allows you to manage the network but only allows to monitor the network
- Nagios makes no difference between various devices like servers, routers, or switches as it treats every device as a host

Conclusion: