Thadomal Shahani Engineering College

Bandra (W.), Mumbai - 400 050.

☙ CERTIFICATE ❧

Certify that Mr./Miss Soumil Salvi of IT Department, Semester VI with Roll No. 104 has completed a course of the necessary experiments in the subject Dev Ops under my supervision in the **Thadomal Shahani Engineering College** Laboratory in the year 2023 - 2024

Teacher In- Charge

Head of the Department

Date ____________

Principal

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

CONTENTS

ASSIGNMENT -1

AIM : To Understand DevOps : principles, practices & DevOps engineer role & responsibilities.

LO Mapped : LO1

THEORY :

Introduction to DevOps:

DevOps is a software development and IT operations approach that aims to bridge the gap between development and operations teams, enabling faster and more reliable software delivery. It emphasizes collaboration, automation, continuous integration, continuous delivery, and continuous monitoring throughout the software development lifecycle.

Principles of DevOps:

1. Collaboration: DevOps promotes strong collaboration between development, operations, and other stakeholders, breaking down silos and fostering communication.

2. Automation: Automation is a core tenet of DevOps, automating repetitive tasks such as code deployment, testing, and infrastructure provisioning to enhance efficiency and reduce human errors.

3. Continuous Integration (CI): CI involves frequently integrating code changes into a shared repository. This practice ensures that code is tested early and often, reducing integration challenges and identifying issues sooner.

4. Continuous Delivery (CD): CD extends CI by automating the process of deploying code changes to production or staging environments, enabling rapid and reliable releases.

5. Infrastructure as Code (IaC): IaC treats infrastructure provisioning as code, allowing developers to define and manage infrastructure using version-controlled scripts. This leads to consistent and reproducible environments.

6. Monitoring and Feedback: DevOps emphasizes continuous monitoring of applications and infrastructure, providing valuable feedback for improvements and quick issue resolution.

Practices in DevOps:

1. Version Control: Using version control systems like Git to manage and track code changes, ensuring collaboration and easy rollbacks.

2. Automated Testing: Employing automated testing frameworks to validate code changes and prevent defects from entering the production environment.

3. Continuous Integration and Continuous Delivery (CI/CD): Setting up pipelines that automatically build, test, and deploy code changes, facilitating rapid and consistent releases.

4. Containerization: Utilizing technologies like Docker to package applications and their dependencies into lightweight containers, ensuring consistency across different environments.

5. Infrastructure as Code (IaC): Defining infrastructure using code, allowing for automated provisioning and easy replication of environments.

6. Monitoring and Logging: Implementing monitoring tools to track application and infrastructure performance, and collecting logs for debugging and analysis.

DevOps Engineer: Role and Responsibilities:

A DevOps engineer plays a crucial role in implementing DevOps practices and principles within an organization. Their responsibilities include:

1. Automation: Developing and maintaining automation scripts for deployment, configuration management, and infrastructure provisioning.

2. CI/CD Pipeline: Designing and managing CI/CD pipelines to enable smooth and continuous software delivery.

3. Infrastructure Management: Creating and managing infrastructure using IaC tools, ensuring scalability, availability, and security.

4. Monitoring and Alerting: Setting up monitoring tools to track application and infrastructure performance, and establishing alerting mechanisms for issue detection.

5. Collaboration: Facilitating collaboration between development and operations teams, promoting a culture of shared responsibility.

6. Problem Solving: Troubleshooting and resolving issues in production environments promptly to minimize downtime and ensure application reliability.

7. Security: Integrating security practices into the DevOps process, ensuring the safety of applications and data.

8. Continuous Improvement: Continuously evaluating and optimizing DevOps practices to enhance efficiency, reliability, and delivery speed.

CONCLUSION :In this experiment we had understood principles , practices & DevOps engineer role & responsibilities

ASSIGNMENT - 2

AIM : To Understand Version Control System, Git Installation & GitHub account

LO Mapped : LO1 ,LO2

THEORY :

Introduction to Version Control System:

A Version Control System (VCS) is a software tool that helps track changes to files and directories over time, enabling multiple contributors to collaborate on projects seamlessly. It provides a history of changes, allows branching and merging, and ensures the integrity of project files.

Key Benefits of Using a VCS:

1. Collaboration: VCS enables multiple developers to work on the same project concurrently without conflicts.

2. History Tracking: It maintains a record of changes, making it easy to revert to previous versions if needed.

3. Branching and Merging: VCS allows developers to work on separate branches, which can be merged back into the main codebase.

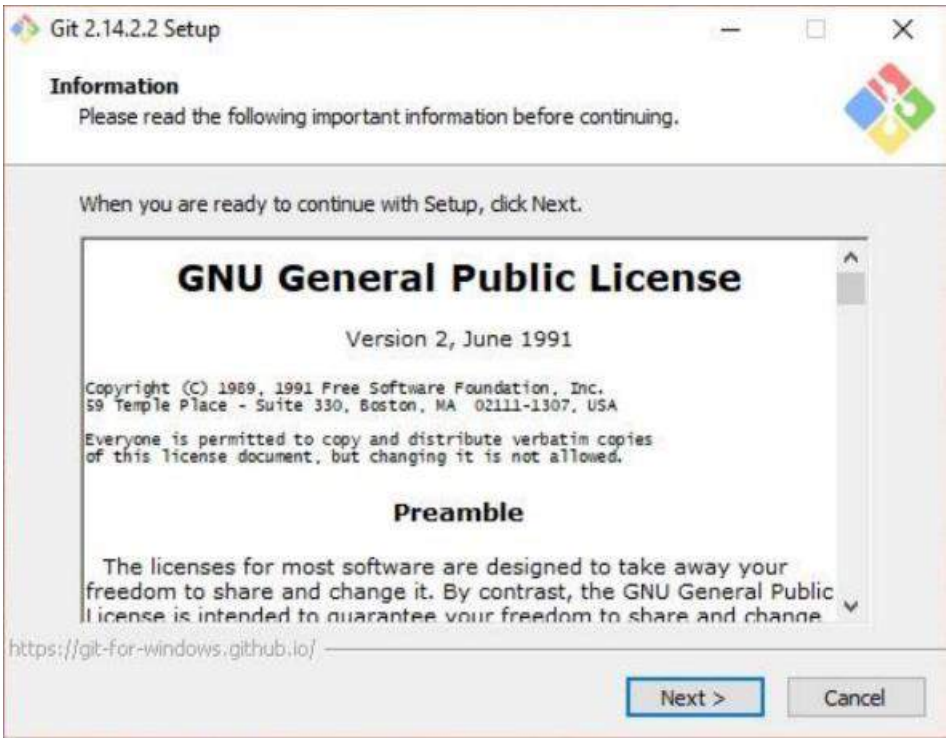
4. Conflict Resolution: When changes overlap, a VCS assists in resolving conflicts efficiently.

Git Installation:

Git is a widely used distributed version control system. Here's how to install Git:

1. Windows:

- Download the installer from the official Git website.
- Run the installer, following the on-screen instructions.
- Choose recommended settings and install Git Bash, a command-line interface.



Git 2.14.2.2 Setup

Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

GNU General Public License

Version 2, June 1991

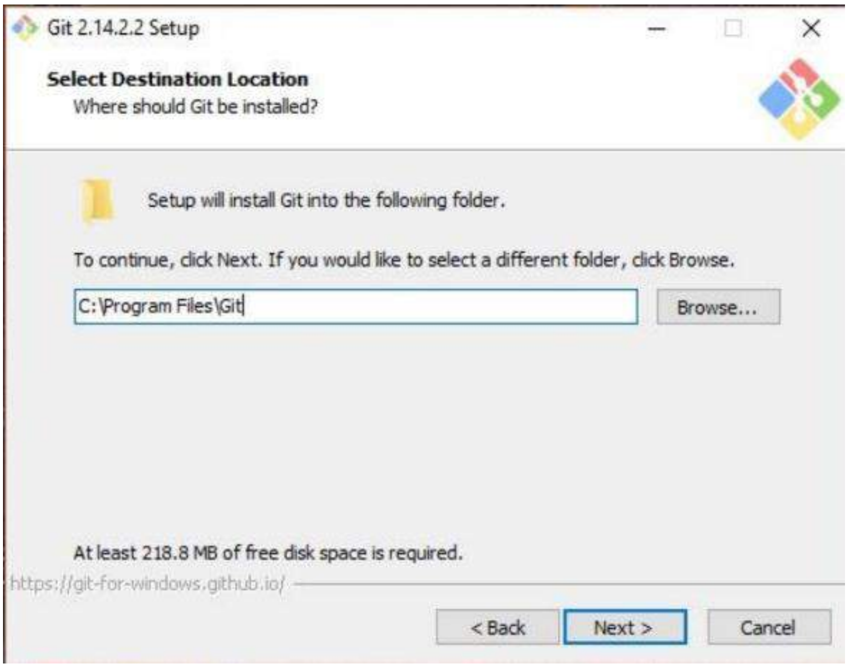
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change

https://git-for-windows.github.io/
Next >
Cancel



Git 2.14.2.2 Setup

Select Destination Location

Where should Git be installed?

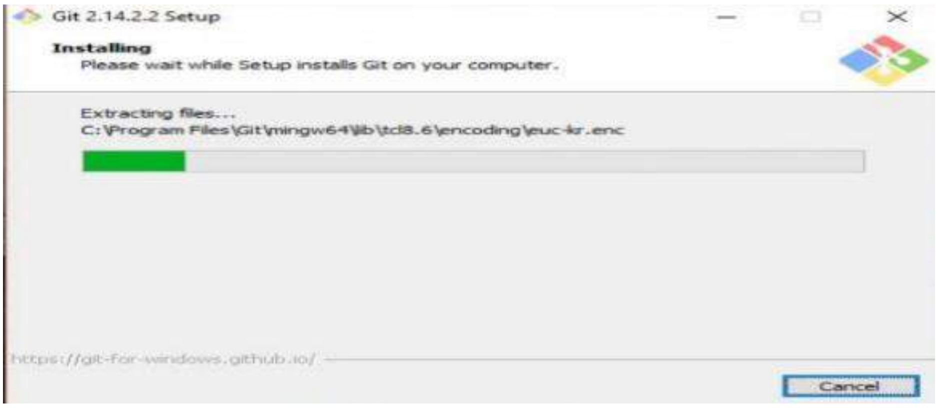
Setup will install Git into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Git

At least 218.8 MB of free disk space is required.

Browse...
https://git-for-windows.github.io/
< Back
Next >
Cancel



Git 2.14.2.2 Setup

Installing

Please wait while Setup installs Git on your computer.

Extracting files...
C:\Program Files\Git\mingw64\lib\tcl8.6\encoding\euc-kr.enc

Setting Up a GitHub Account:

GitHub is a popular platform for hosting Git repositories and collaborating on projects. Here's how to set up a GitHub account:

1. Create an Account:

   - Visit the GitHub website (github.com) and click "Sign Up".

   - Provide your details, including username, email, and password.

2. Verify Email:

   - GitHub will send a verification email. Click the link to verify your email address.

3. Set Up Profile:

   - After verification, log in to GitHub.

   - Customize your profile by adding a profile picture, bio, and other information.

4. Create a Repository:

- Click the "+" icon in the top right corner and select "New Repository".

- Give your repository a name, description, and choose visibility (public or private).

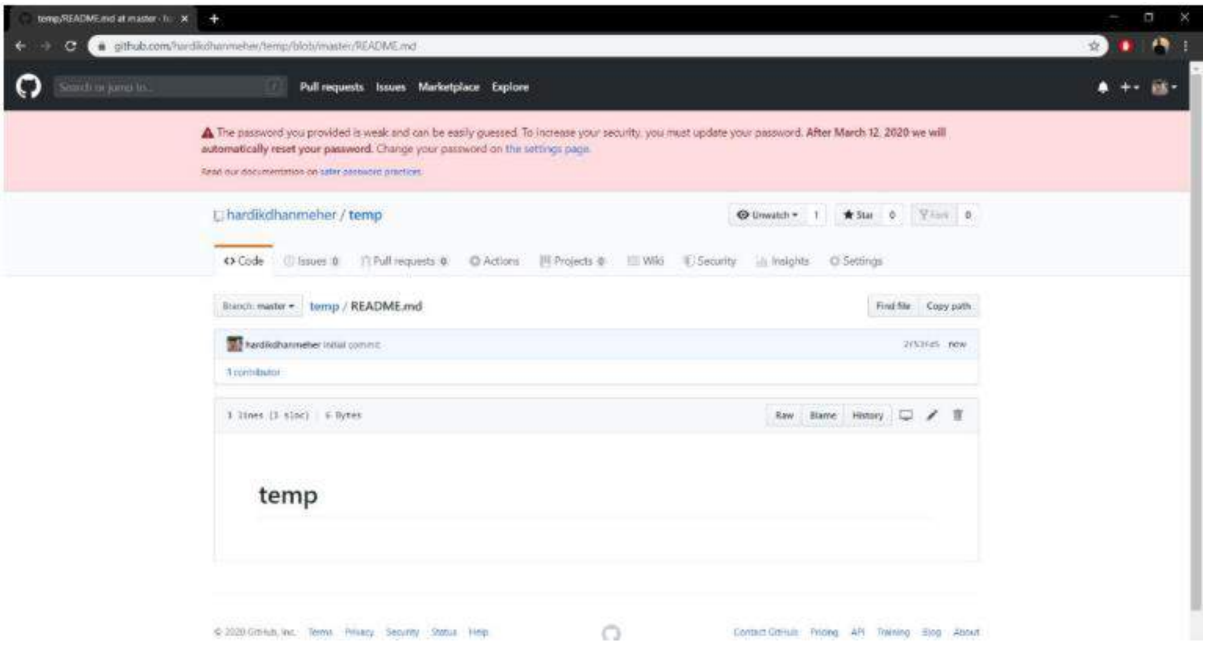
5. Clone Repository:

- Once your repository is created, you can clone it to your local machine using `git clone`.

6. Pushing Changes:

- Make changes to files in the repository, then use `git add` to stage changes and `git commit` to save changes.

- Use `git push` to upload changes to the GitHub repository.



The password you provided is weak and can be easily guessed. To increase your security, you must update your password. **After March 12, 2020 we will automatically reset your password.** Change your password on the settings page.

Read our documentation on safer password practices.

hardikdhanmeher / temp

temp / README.md

hardikdhanmeher Initial commit 2f53fd5 now

1 contributor

1 lines (1 sloc) 6 Bytes

temp

CONCLUSION :

Version Control Systems like Git and platforms like GitHub are essential tools for modern software development. We understood about Version Control System and also installed Git and created a GitHub account.

ASSIGNMENT - 3

AIM : To perform various Git operations

LO MAPPED : LO1, LO2

THEORY :

1) git config

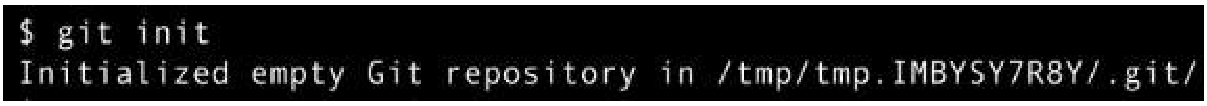
Utility : To set your user name and email in the main configuration file.

How to : To check your name and email type in git config --global user.name and git config -- global user.email.

2) git init

Utility : To initialise a git repository for a new or existing project.

How to : git init in the root of your project directory.



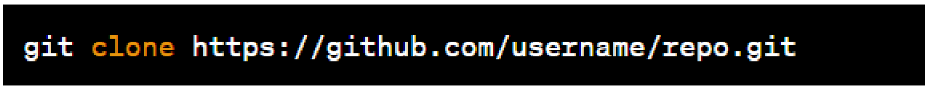
```
$ git init
Initialized empty Git repository in /tmp/tmp.IMBYSY7R8Y/.git/
```

3) git clone

Utility : To copy a git repository from remote source, also sets the remote to original source so

that you can pull again.

How to : git clone <:clone git url:>



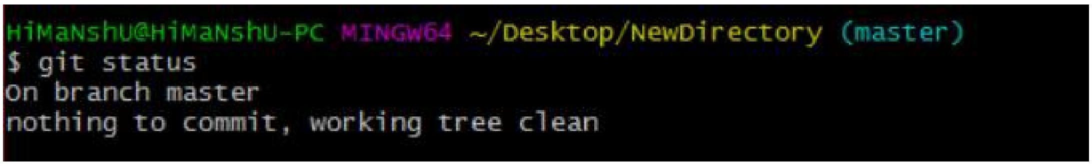
```
git clone https://github.com/username/repo.git
```

4) git status

Utility : To check the status of files you've changed in your working directory, i.e, what all has

changed since your last commit.

How to : git status in your working directory. lists out all the files that have been changed.

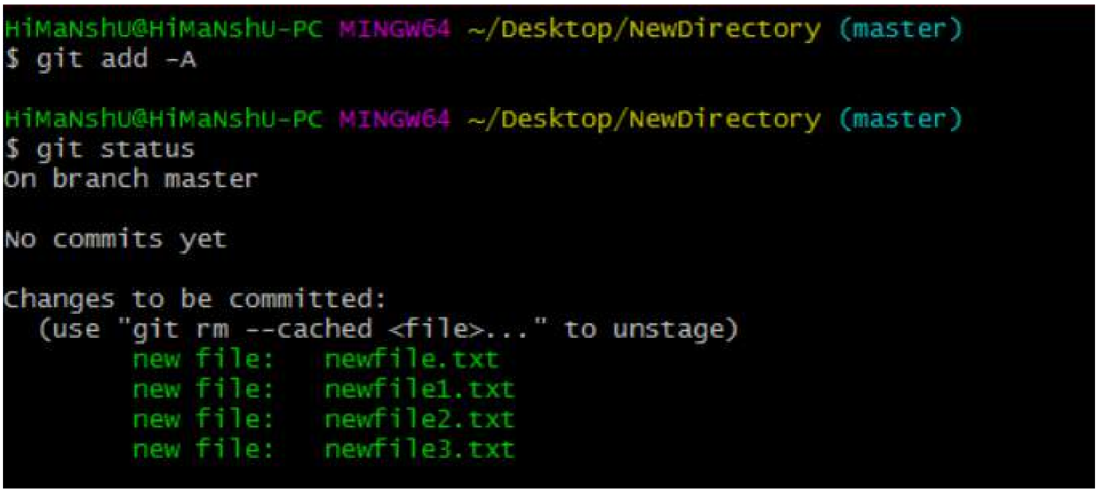


```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/NewDirectory (master)
$ git status
On branch master
nothing to commit, working tree clean
```

5) git add

Utility : adds changes to stage/index in your working directory.

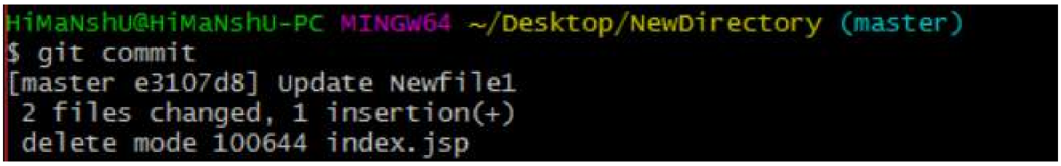
How to : git add .



```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/NewDirectory (master)
$ git add -A

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/NewDirectory (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   newfile.txt
        new file:   newfile1.txt
        new file:   newfile2.txt
        new file:   newfile3.txt
```

6) git commit

Utility : commits your changes and sets it to new commit object for your remote.



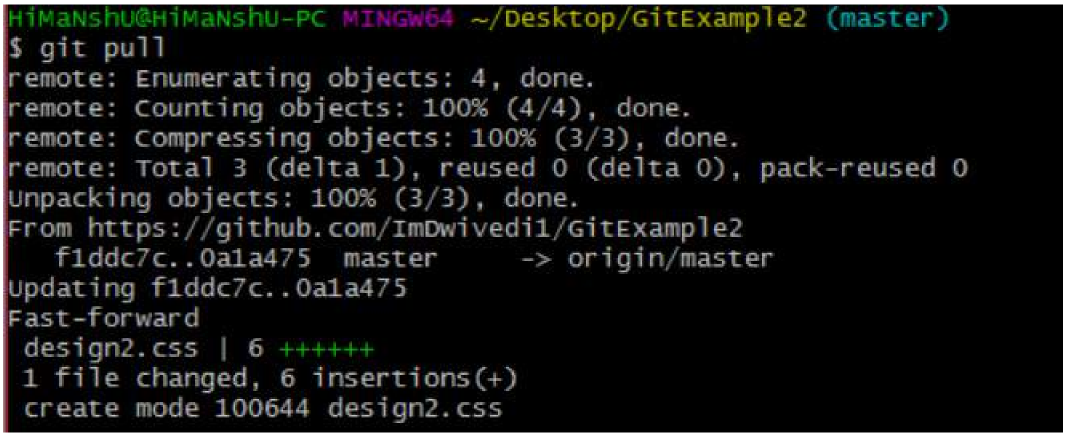
```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/NewDirectory (master)
$ git commit
[master e3107d8] Update Newfile1
 2 files changed, 1 insertion(+)
 delete mode 100644 index.jsp
```

7) git push/git pull

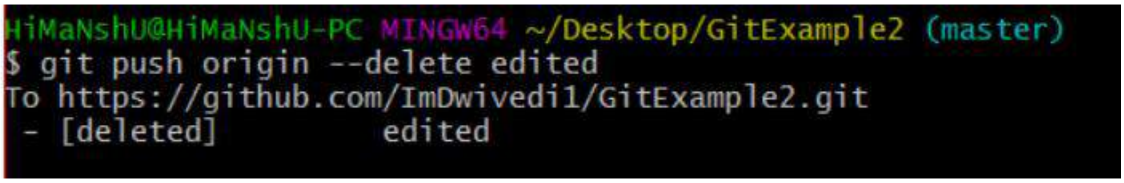
Utility : Push or Pull your changes to remote. If you have added and committed your changes

and you want to push them. Or if your remote has updated and you want those latest changes.

How to : git pull <:remote:> <:branch:> and git push <:remote:> <:branch:>



```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/ImDwivedi1/GitExample2
   f1ddc7c..0a1a475  master     -> origin/master
Updating f1ddc7c..0a1a475
Fast-forward
 design2.css | 6 ++++++
 1 file changed, 6 insertions(+)
 create mode 100644 design2.css
```

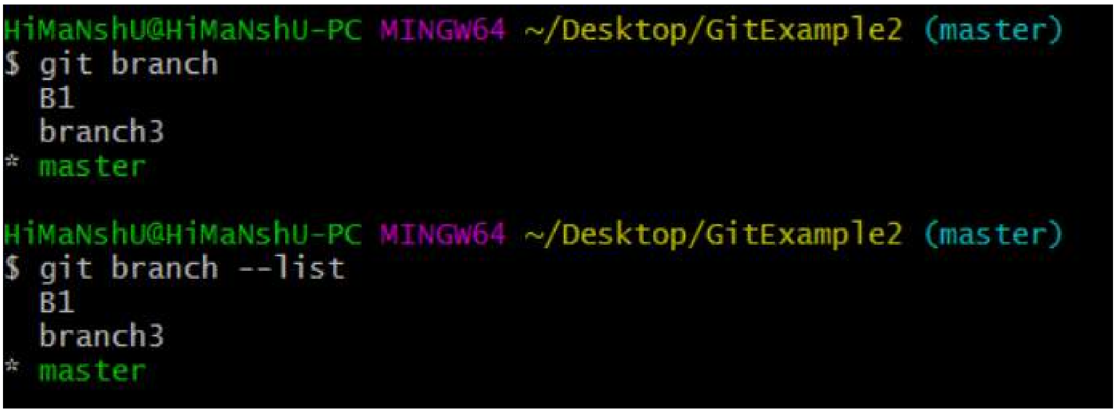


```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git push origin --delete edited
To https://github.com/ImDwivedi1/GitExample2.git
 - [deleted]         edited
```

8) git branch

Utility : Lists out all the branches.

How to : git branch or git branch -a to list all the remote branches as well.



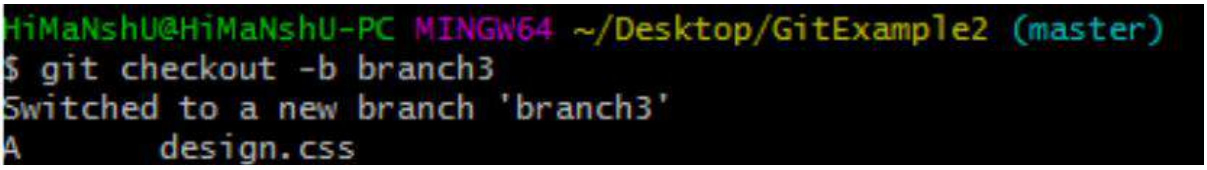
```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git branch
  B1
  branch3
* master

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git branch --list
  B1
  branch3
* master
```

9) git checkout

Utility : Switch to different branches

How to : git checkout <:branch:> or **_git checkout -b <:branch:> if you want to create and

switch to a new branch.



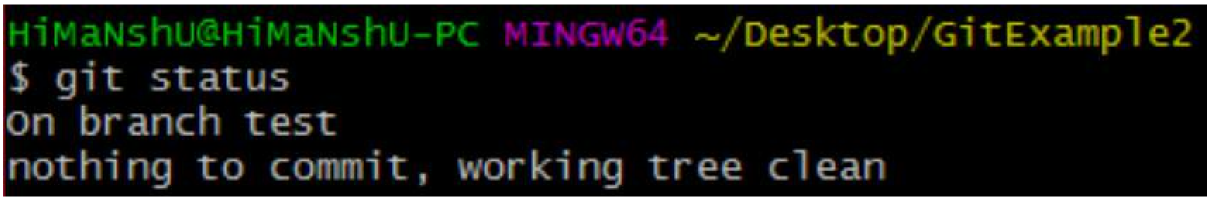
```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git checkout -b branch3
Switched to a new branch 'branch3'
A       design.css
```

10) git stash

Utility : Save changes that you don’t want to commit immediately.

How to : git stash in your working directory. git stash apply if you want to bring your saved

changes back.



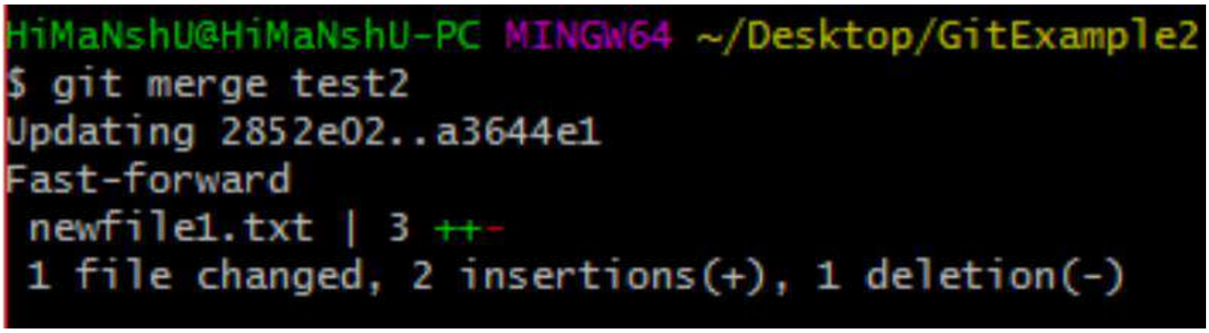
```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2
$ git status
On branch test
nothing to commit, working tree clean
```

11) git merge

Utility : Merge two branches you were working on.

How to : Switch to branch you want to merge everything in. git

merge <:branch_you_want_to_merge:>



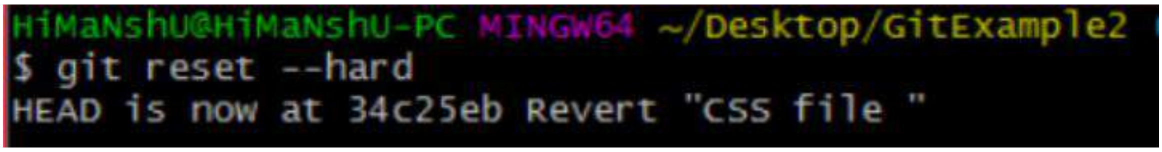
```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2
$ git merge test2
Updating 2852e02..a3644e1
Fast-forward
 newfile1.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

12) git reset

Utility : You know when you commit changes that are not complete, this sets your index to the

latest commit that you want to work on with.

How to : git reset <:mode:> <:COMMIT:>

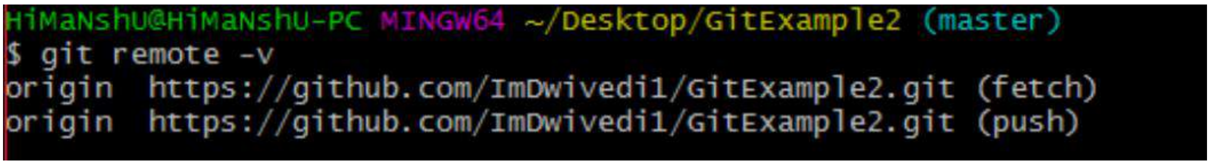


```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2
$ git reset --hard
HEAD is now at 34c25eb Revert "CSS file "
```

13) git remote

Utility : To check what remote/source you have or add a new remote.

How to : git remote to check and list. And git remote add <:remote_url:>



```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git remote -v
origin  https://github.com/ImDwivedi1/GitExample2.git (fetch)
origin  https://github.com/ImDwivedi1/GitExample2.git (push)
```

CONCLUSION :

In this Experiment we had studied different command that are used to perform operation on local Git .

ASSIGNMENT 4

AIM : To understand Continuous integration , Jenkins installation

LO MAPPED : LO1,LO3

THEORY :

Jenkins is an open source automation server. With Jenkins, organizations can accelerate the software development process by automating it. Jenkins manages and controls software delivery processes throughout the entire lifecycle, including build, document, test, package, stage, deployment, static code analysis and much more. Here's a basic outline of how to install Jenkins:

Prerequisites:

Make sure you have Java installed on your system since Jenkins is a Java-based application.

1. Download Jenkins:

"Download Jenkins" page.

Choose the appropriate package based on your operating system (usually a WAR file for manual installation or an installer for Windows).

2. Install Jenkins:

If you're using a WAR file, you'll need to run Jenkins as a Java application. Open a command prompt or terminal and navigate to the directory where you downloaded the Jenkins WAR file.

3. Run the command: java -jar jenkins.war

Jenkins will start initializing. Once it's done, you will see logs indicating that Jenkins is running. Keep this terminal open while Jenkins is running.

4. Access Jenkins Web UI:

Open a web browser and go this is the default URL for accessing the Jenkins web interface.

You will be prompted to enter an initial administrator password. To obtain this password, go back to the terminal where you started Jenkins. You'll find the initial password there. Copy and paste it into the web interface.

5. Customize Plugins (Optional):

Jenkins will then ask you to install plugins. You can either choose the recommended plugins or select specific plugins based on your needs.

6. Create an Admin User:

After the plugins are installed, you will need to set up an admin user account.

7. Configure Jenkins URL:

Jenkins will suggest a URL, but you can change it if needed.

8. Finish Installation:

Once the configuration is complete, Jenkins will display a confirmation.

9. Start Using Jenkins:

You can now log in with the admin credentials you created and start using Jenkins.

Remember, this is a general outline of the installation process. The actual steps might vary based on your operating system and the version of Jenkins you are installing. Always refer to the official Jenkins documentation or any specific instructions provided by your instructor or organization.

Additionally, it's a good practice to install Jenkins in a controlled environment, such as a virtual machine, to avoid conflicts with other software on your computer.

Understanding the CI Process in Jenkins:

1. Code Commit: Developers commit their code changes to the version control repository.
2. Triggering Build: Jenkins monitors the repository for code changes. When changes are detected, Jenkins triggers a build process.
3. Build Stage: During the build stage, Jenkins pulls the latest code, compiles it, resolves dependencies, and generates artifacts.
4. Testing Stage: After the build, Jenkins runs a series of automated tests, including unit tests and integration tests. This ensures that the code changes do not introduce new bugs or regressions.
5. Reporting: Jenkins generates reports based on the test results, highlighting any failures or issues.
6. Artifact Archival: Successful builds produce artifacts, which Jenkins archives for future deployment.
7. Deployment: While CI primarily focuses on integration and testing, Jenkins can also be configured to trigger deployments to staging environments for further testing or even to production environments for Continuous Delivery (CD).

Benefits of CI in Jenkins:

1. Early Issue Detection: CI identifies integration problems and bugs early in the development cycle, reducing the cost and effort required to fix them.

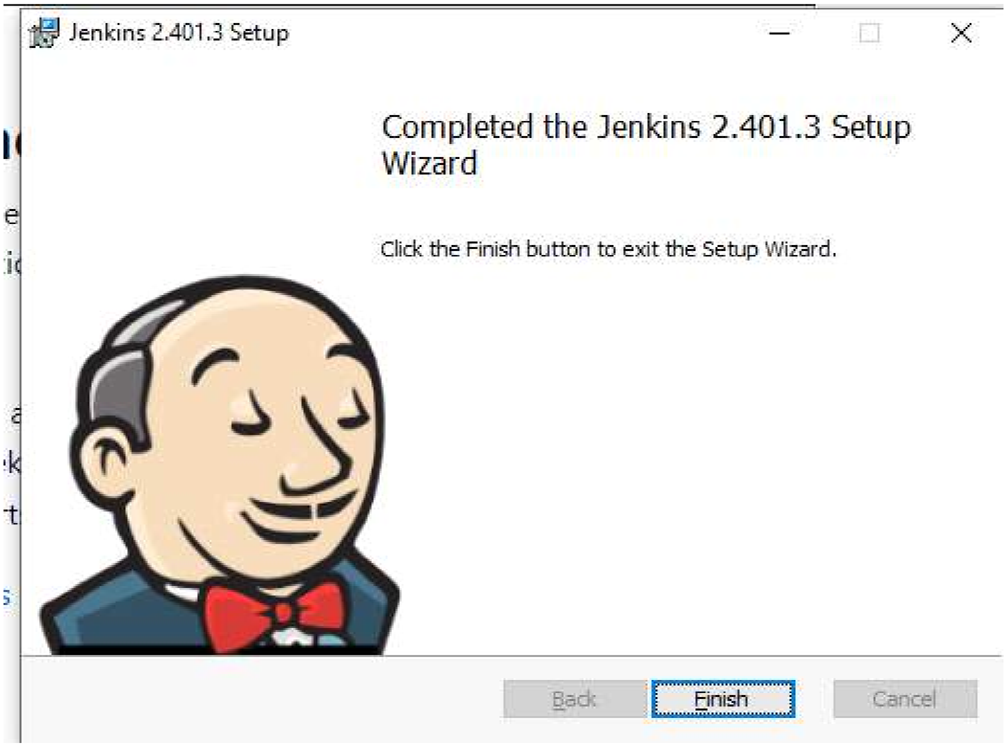
2. Consistent Builds: Jenkins ensures consistent and repeatable builds, reducing discrepancies between development and production environments.

3. Automated Testing: Automated testing in Jenkins enhances code quality by catching defects before they reach users.

4. Faster Feedback: Developers receive quick feedback on their code changes, encouraging them to make smaller, more frequent commits.

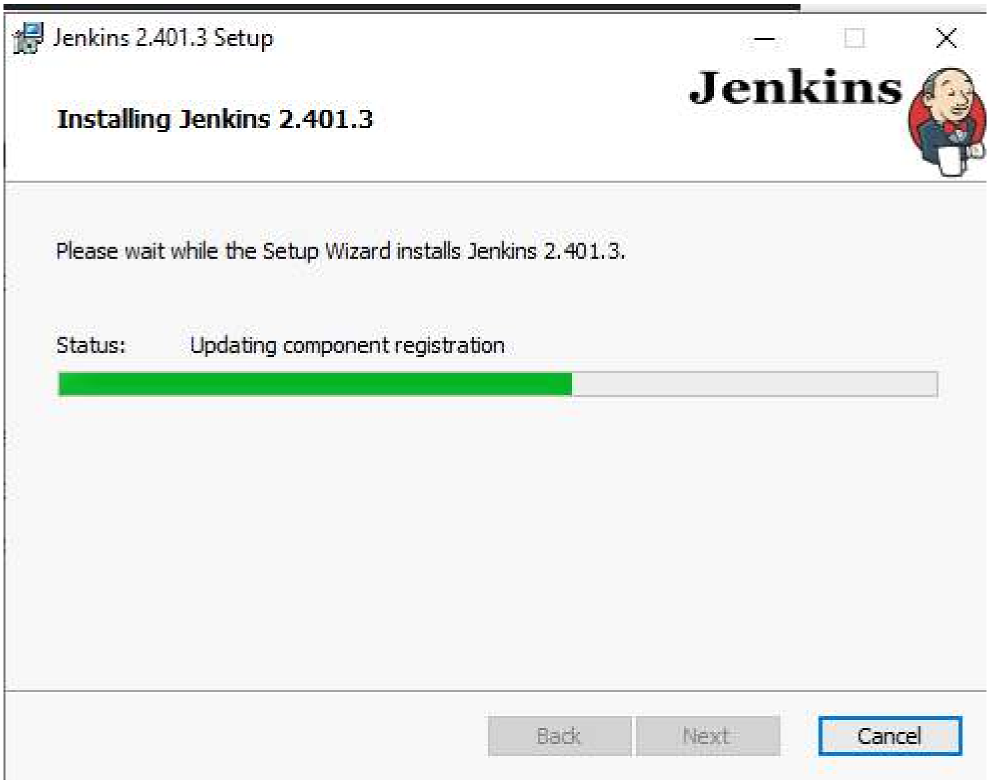
5. Efficient Collaboration: CI promotes collaboration among development teams by providing a central platform for code integration and testing.

6. Streamlined Deployment: When integrated with CD, Jenkins enables automated and reliable deployments, ensuring a smooth software delivery process.



Completed the Jenkins 2.401.3 Setup Wizard

Click the Finish button to exit the Setup Wizard.

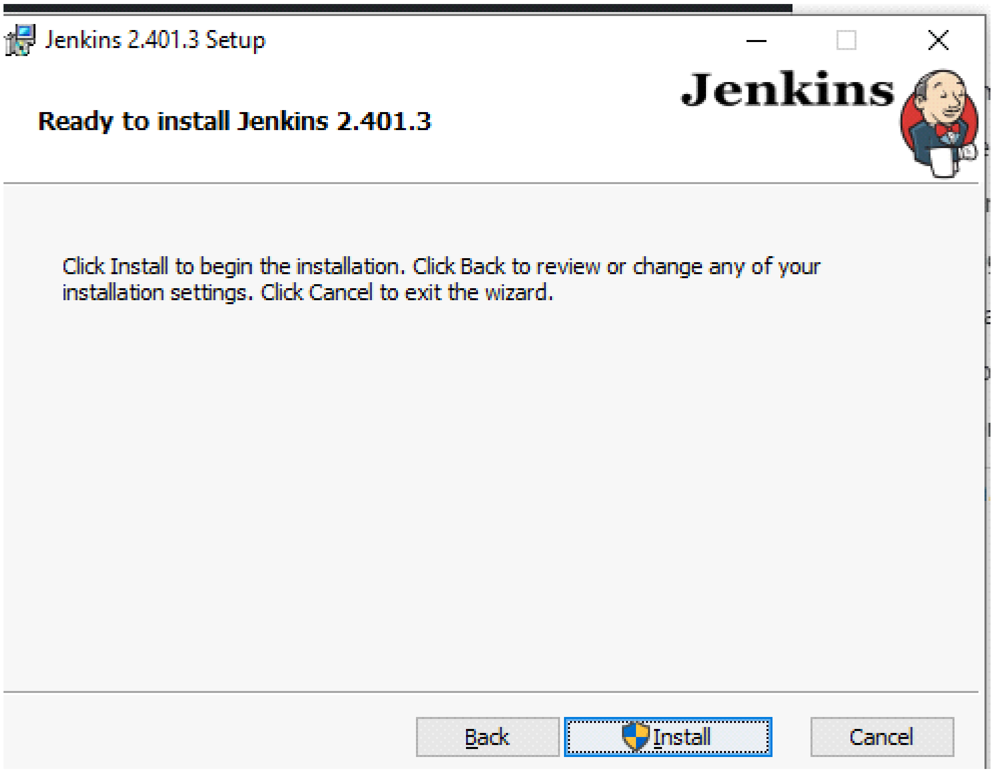


Jenkins 2.401.3 Setup

Installing Jenkins 2.401.3

Please wait while the Setup Wizard installs Jenkins 2.401.3.

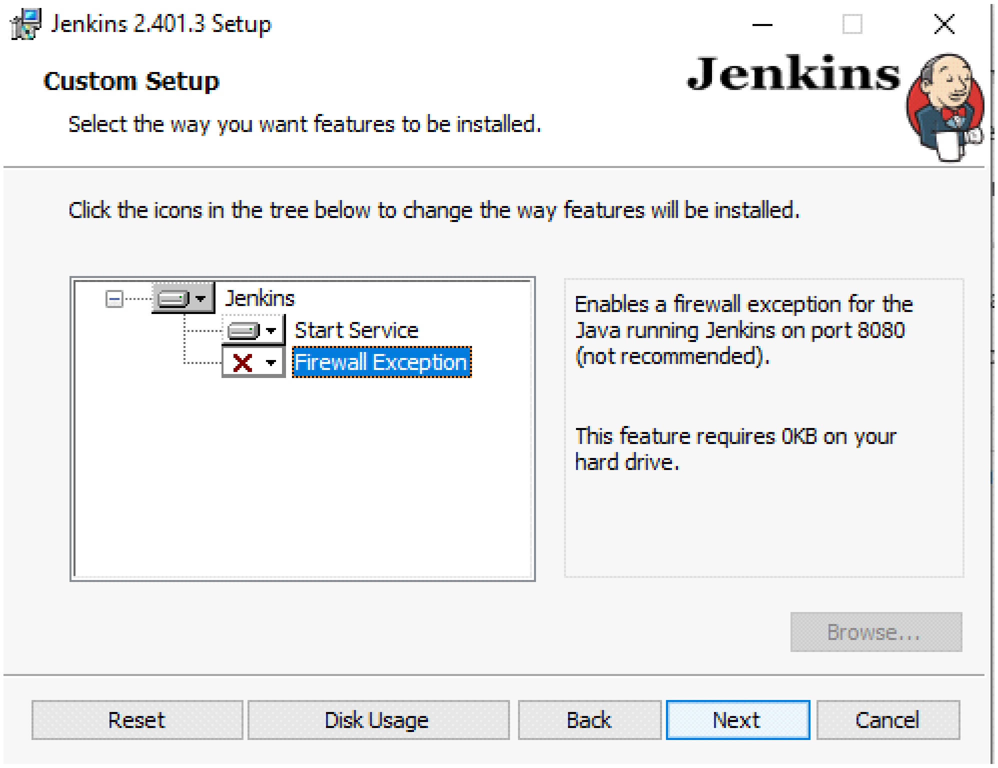
Status: Updating component registration



Jenkins 2.401.3 Setup

Ready to install Jenkins 2.401.3

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.



Jenkins 2.401.3 Setup

Custom Setup

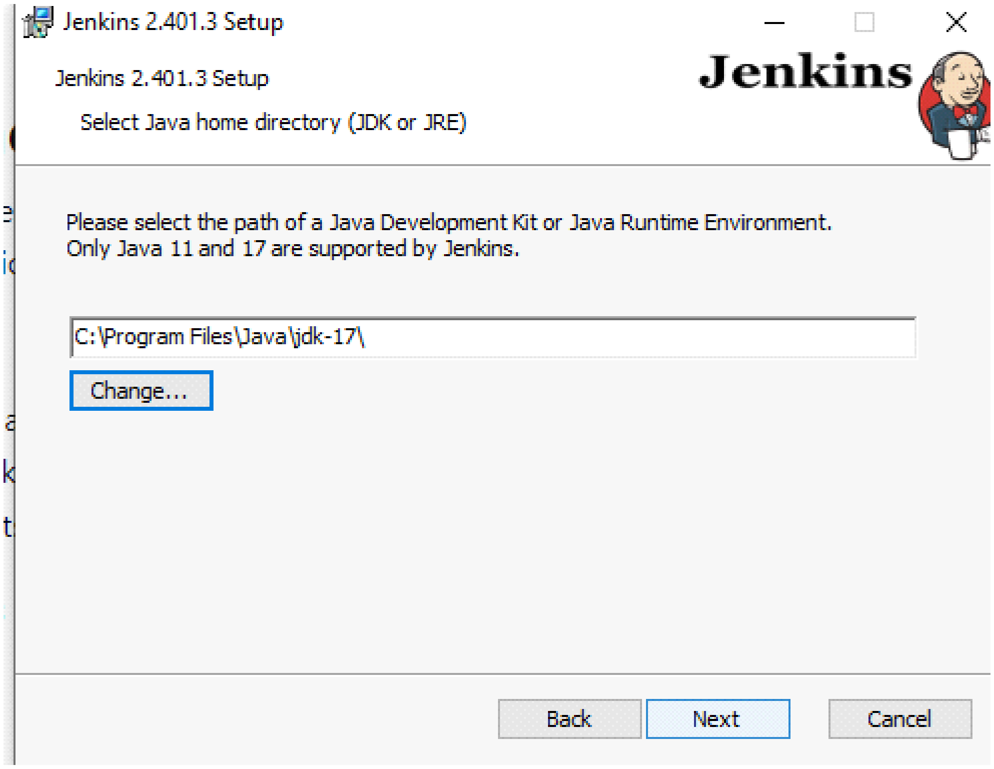
Select the way you want features to be installed.

Click the icons in the tree below to change the way features will be installed.

- Jenkins
  - Start Service
  - ☒ Firewall Exception

Enables a firewall exception for the Java running Jenkins on port 8080 (not recommended).

This feature requires 0KB on your hard drive.



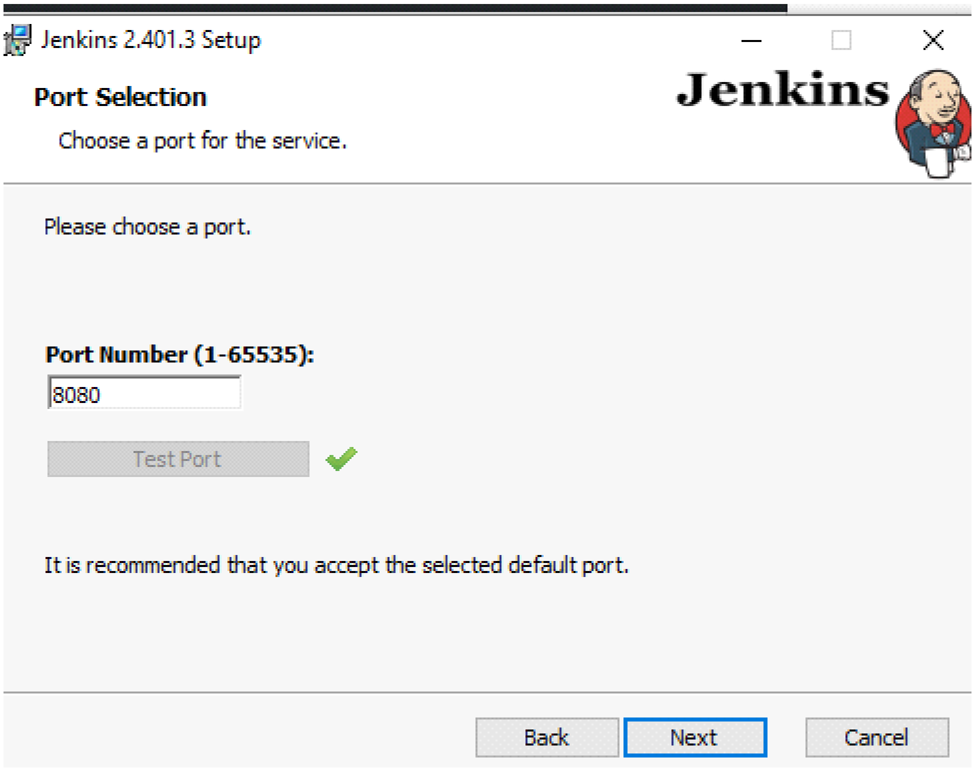
Jenkins 2.401.3 Setup

Jenkins 2.401.3 Setup

Select Java home directory (JDK or JRE)

Please select the path of a Java Development Kit or Java Runtime Environment.
Only Java 11 and 17 are supported by Jenkins.

C:\Program Files\Java\jdk-17\



Jenkins 2.401.3 Setup

Port Selection

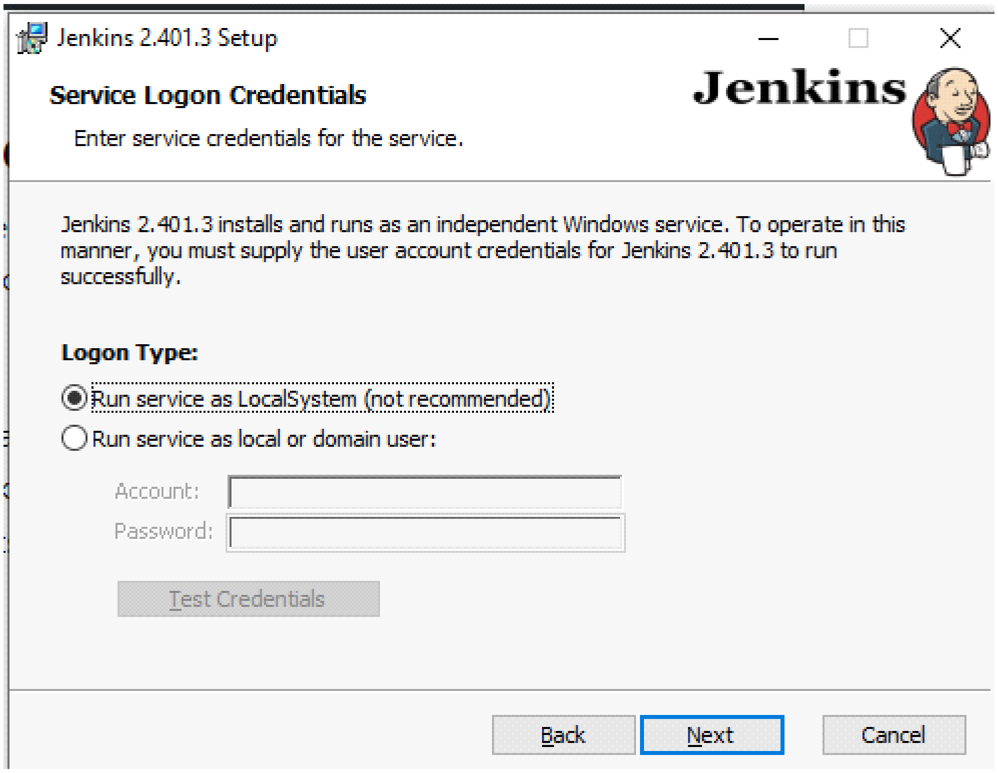
Choose a port for the service.

Please choose a port.

Port Number (1-65535):

8080

It is recommended that you accept the selected default port.



Jenkins 2.401.3 Setup

Service Logon Credentials

Enter service credentials for the service.

Jenkins 2.401.3 installs and runs as an independent Windows service. To operate in this manner, you must supply the user account credentials for Jenkins 2.401.3 to run successfully.

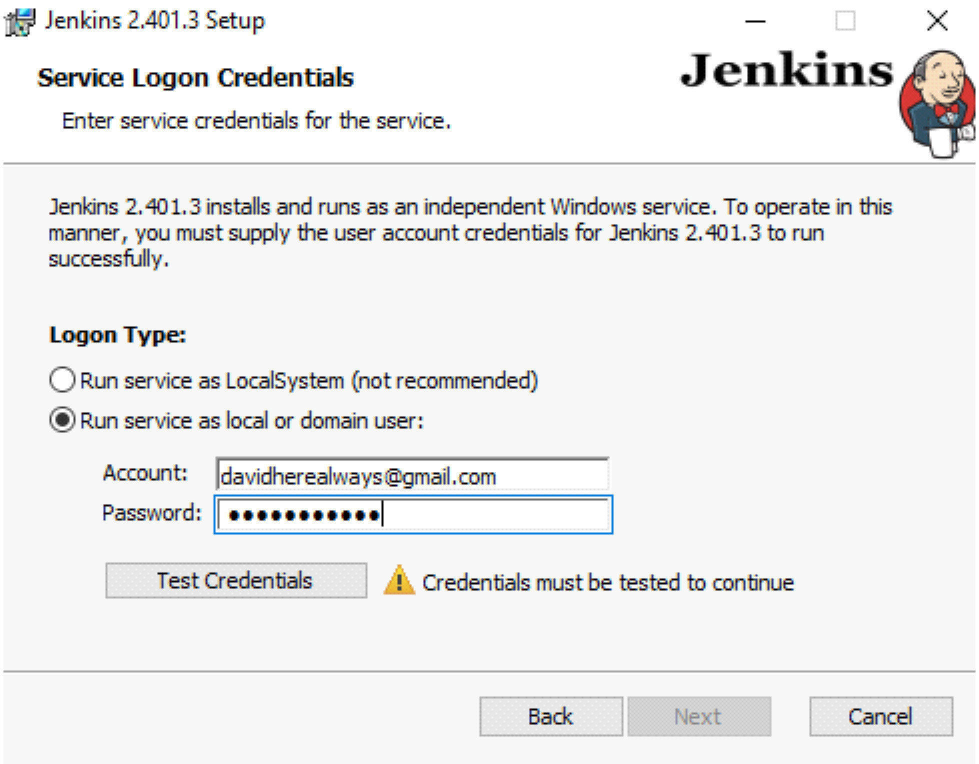
Logon Type:

● Run service as LocalSystem (not recommended)

○ Run service as local or domain user:

Account:

Password:



Service Logon Credentials

Enter service credentials for the service.

Jenkins 2.401.3 installs and runs as an independent Windows service. To operate in this manner, you must supply the user account credentials for Jenkins 2.401.3 to run successfully.

Logon Type:

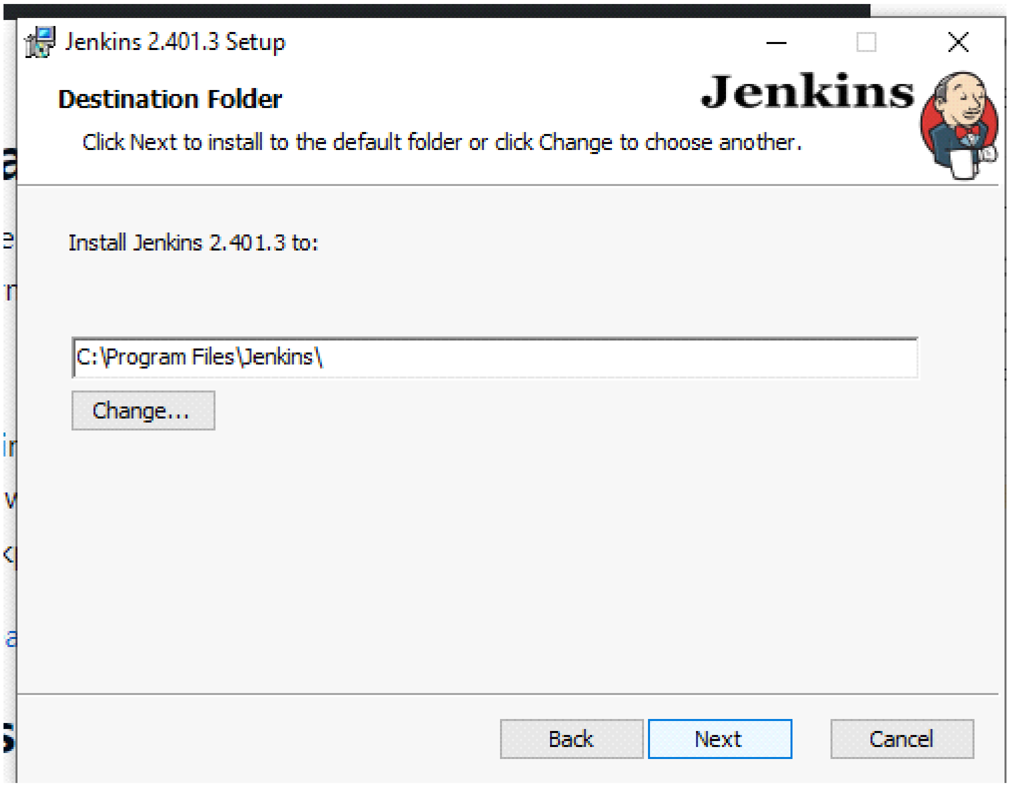
○ Run service as LocalSystem (not recommended)

● Run service as local or domain user:

Account: davidherealways@gmail.com

Password: ●●●●●●●●●●●●

⚠ Credentials must be tested to continue

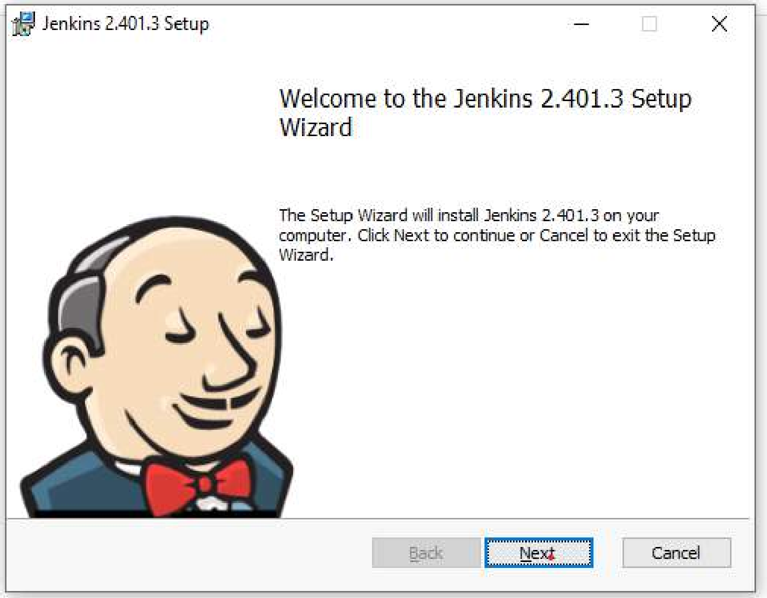


Destination Folder

Click Next to install to the default folder or click Change to choose another.

Install Jenkins 2.401.3 to:

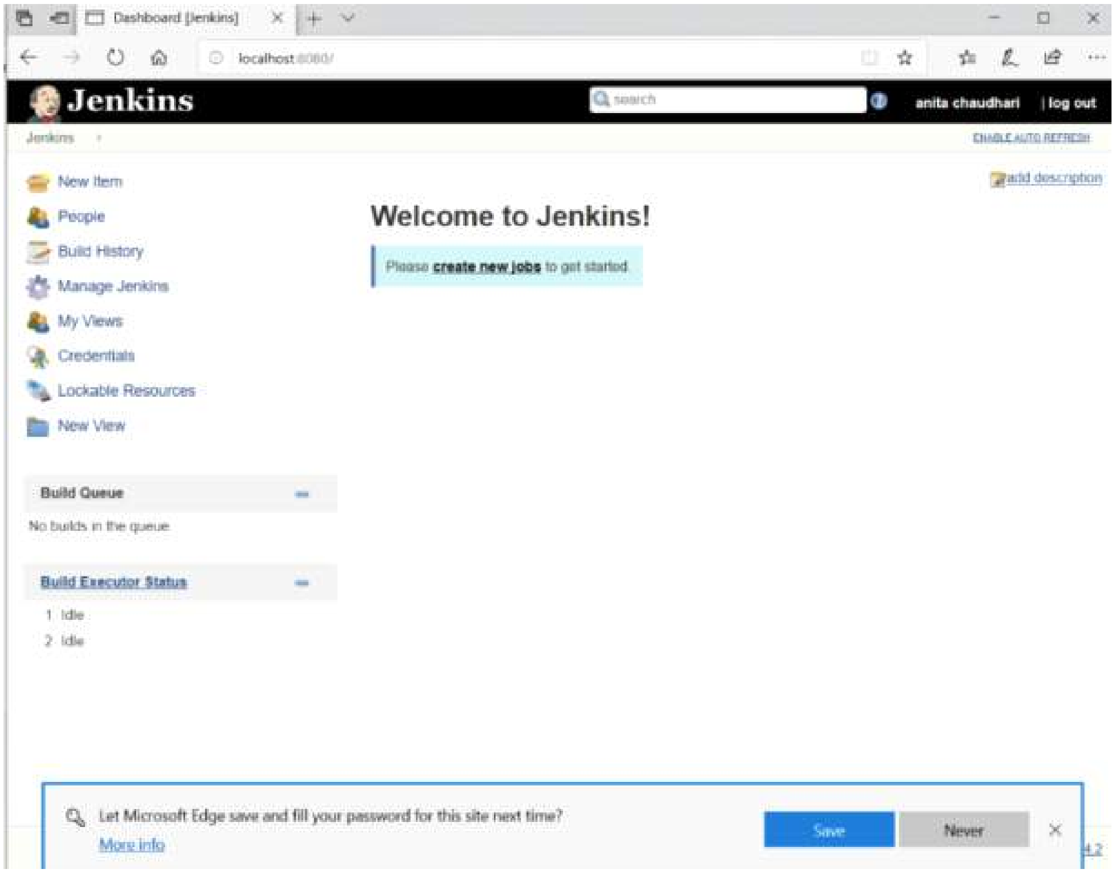
C:\Program Files\Jenkins\



Jenkins 2.401.3 Setup

Welcome to the Jenkins 2.401.3 Setup Wizard

The Setup Wizard will install Jenkins 2.401.3 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.



Welcome to Jenkins!

Please **create new jobs** to get started.

Build Queue

No builds in the queue

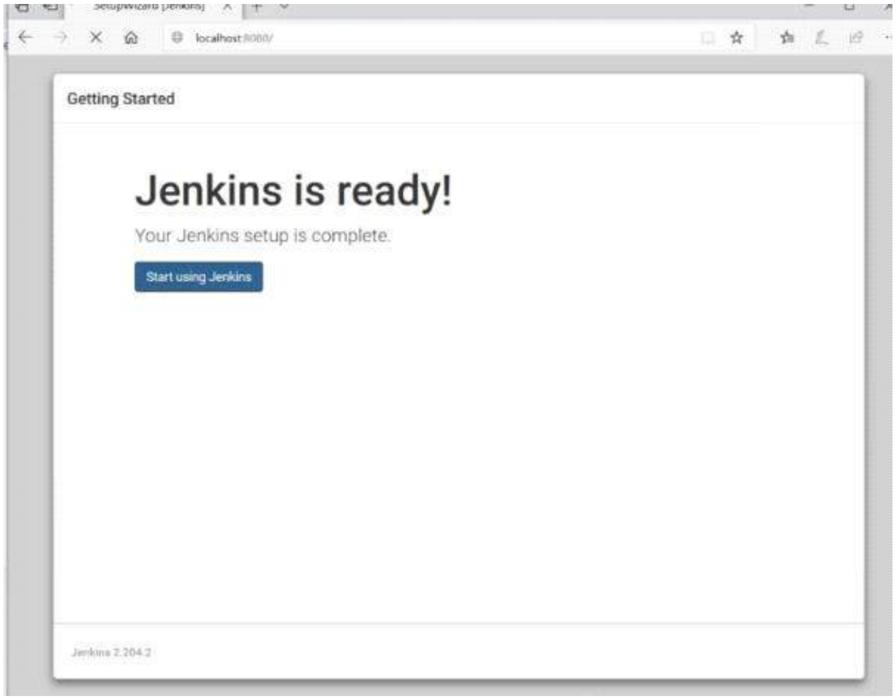
Build Executor Status

1 Idle

2 Idle

Let Microsoft Edge save and fill your password for this site next time?

More info

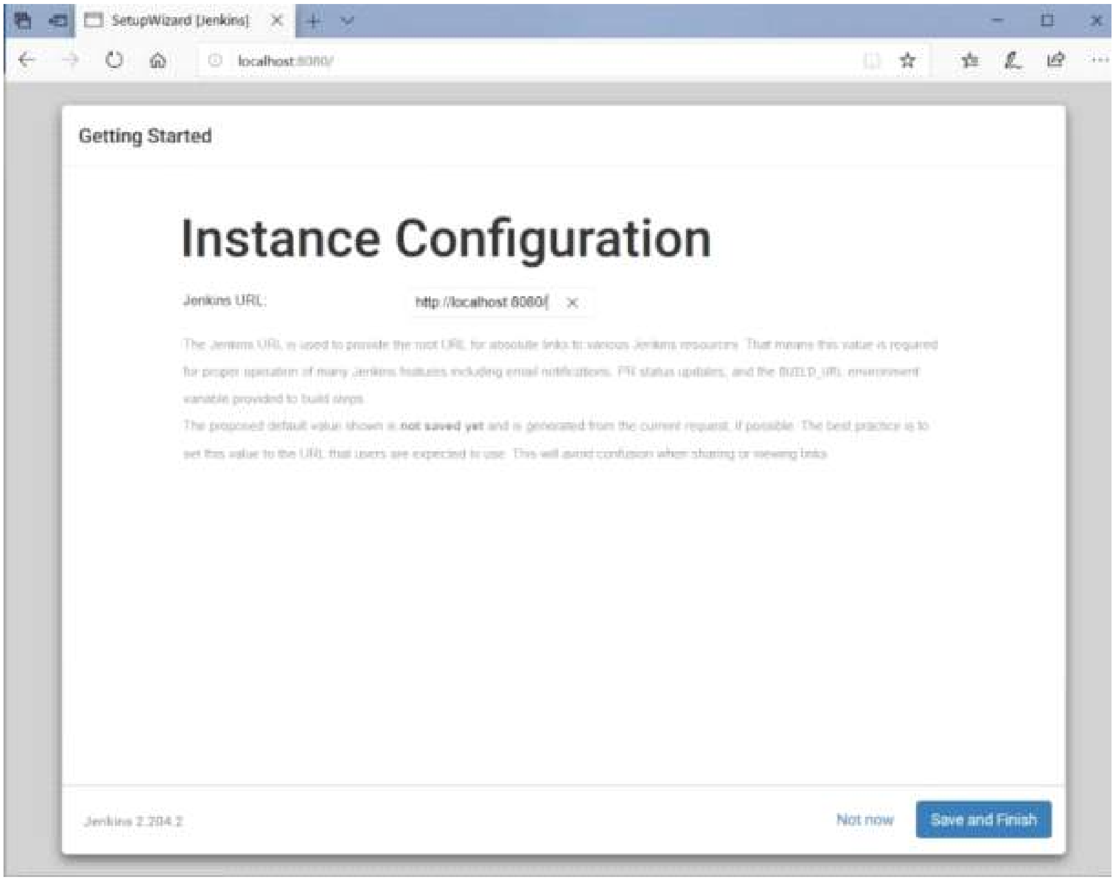


Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

Jenkins 2.204.2



Getting Started

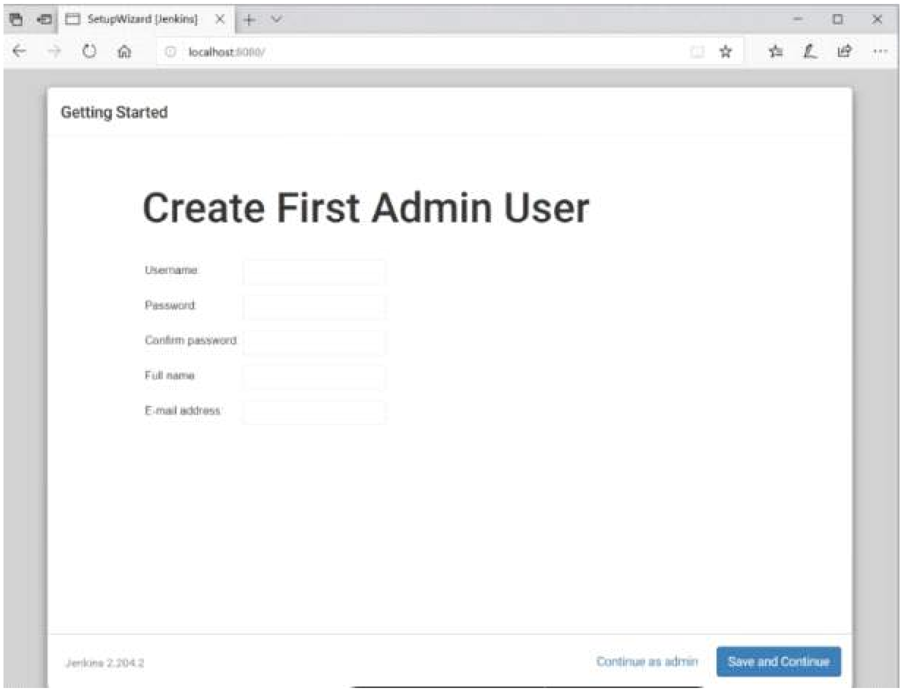
Instance Configuration

Jenkins URL: http://localhost:8080/

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.204.2



Getting Started

Create First Admin User

Username

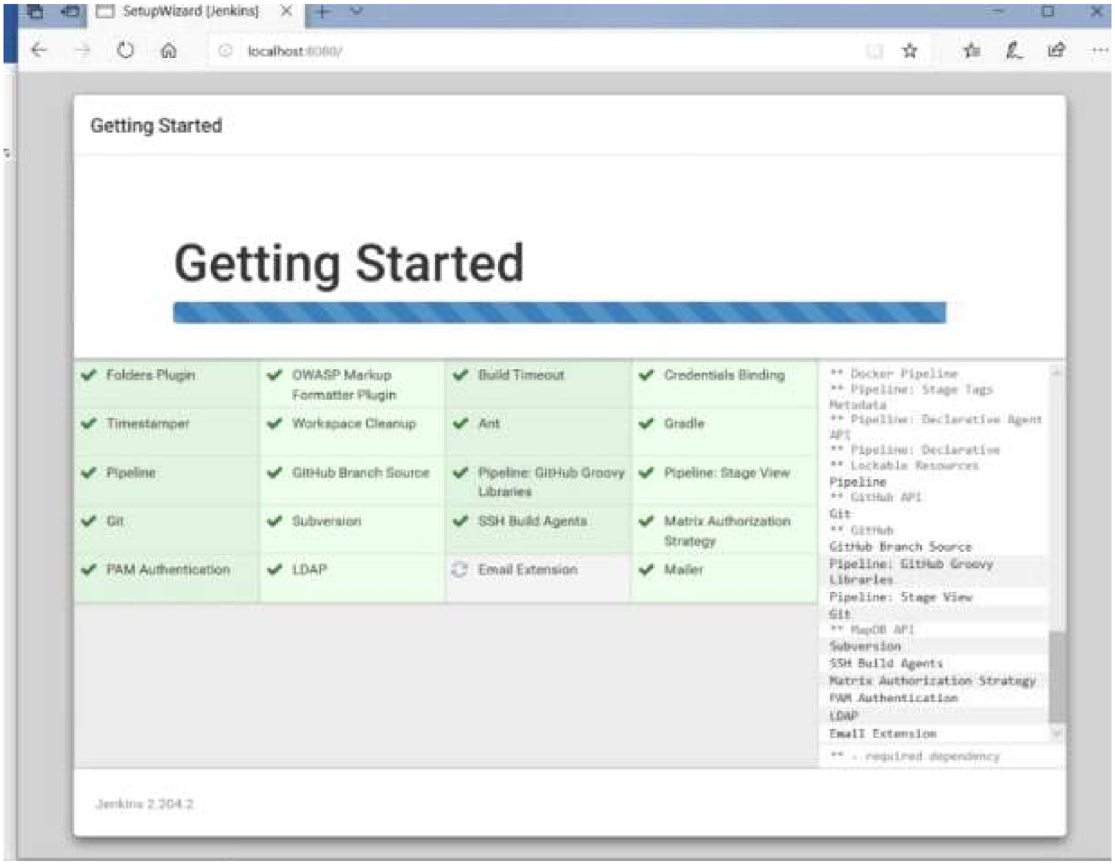
Password

Confirm password

Full name

E-mail address

Jenkins 2.204.2



Getting Started

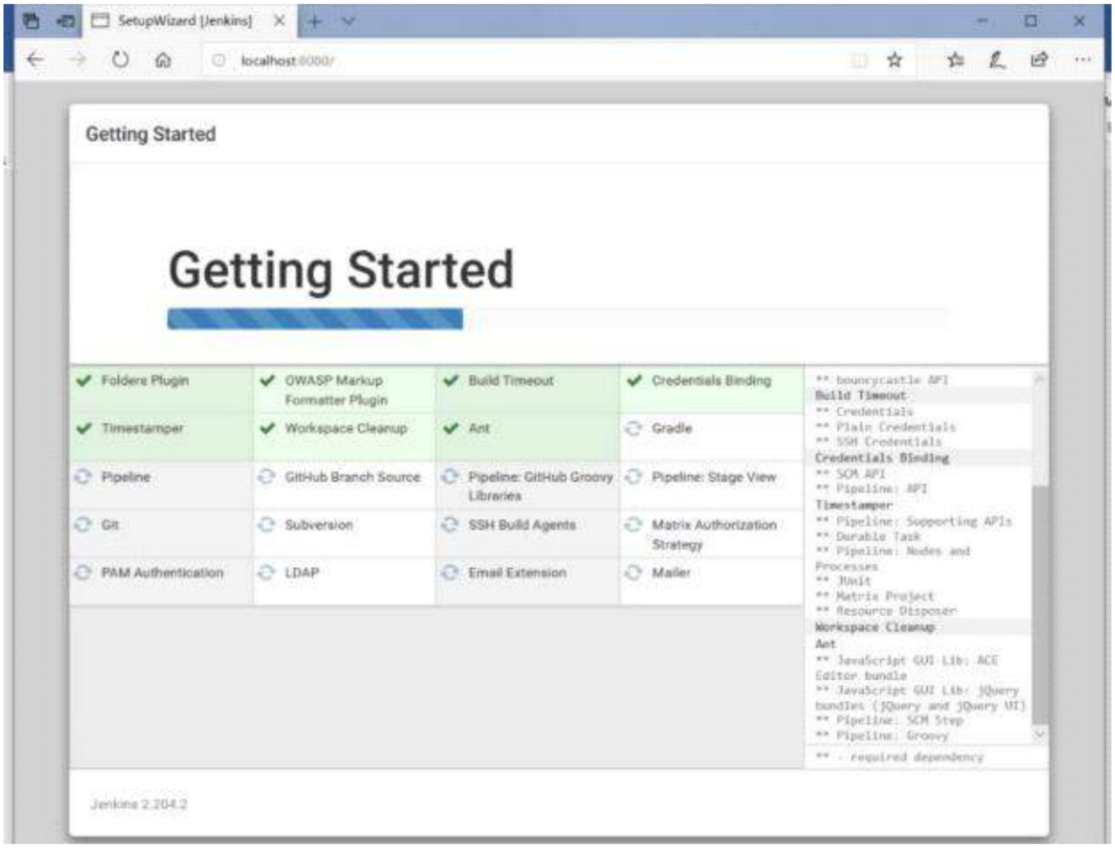
Getting Started

✓ Folders Plugin	✓ OWASP Markup Formatter Plugin	✓ Build Timeout	✓ Credentials Binding
✓ Timestamper	✓ Workspace Cleanup	✓ Ant	✓ Gradle
✓ Pipeline	✓ GitHub Branch Source	✓ Pipeline: GitHub Groovy Libraries	✓ Pipeline: Stage View
✓ Git	✓ Subversion	✓ SSH Build Agents	✓ Matrix Authorization Strategy
✓ PAM Authentication	✓ LDAP	Email Extension	✓ Mailer

```
** Docker Pipeline
** Pipeline: Stage Tags
Metadata
** Pipeline: Declarative Agent
API
** Pipeline: Declarative
** Lockable Resources
Pipeline
** GitHub API
Git
** GitHub
GitHub Branch Source
Pipeline: GitHub Groovy
Libraries
Pipeline: Stage View
Git
** MapDB API
Subversion
SSH Build Agents
Matrix Authorization Strategy
PAM Authentication
LDAP
Email Extension
```

** - required dependency

Jenkins 2.204.2



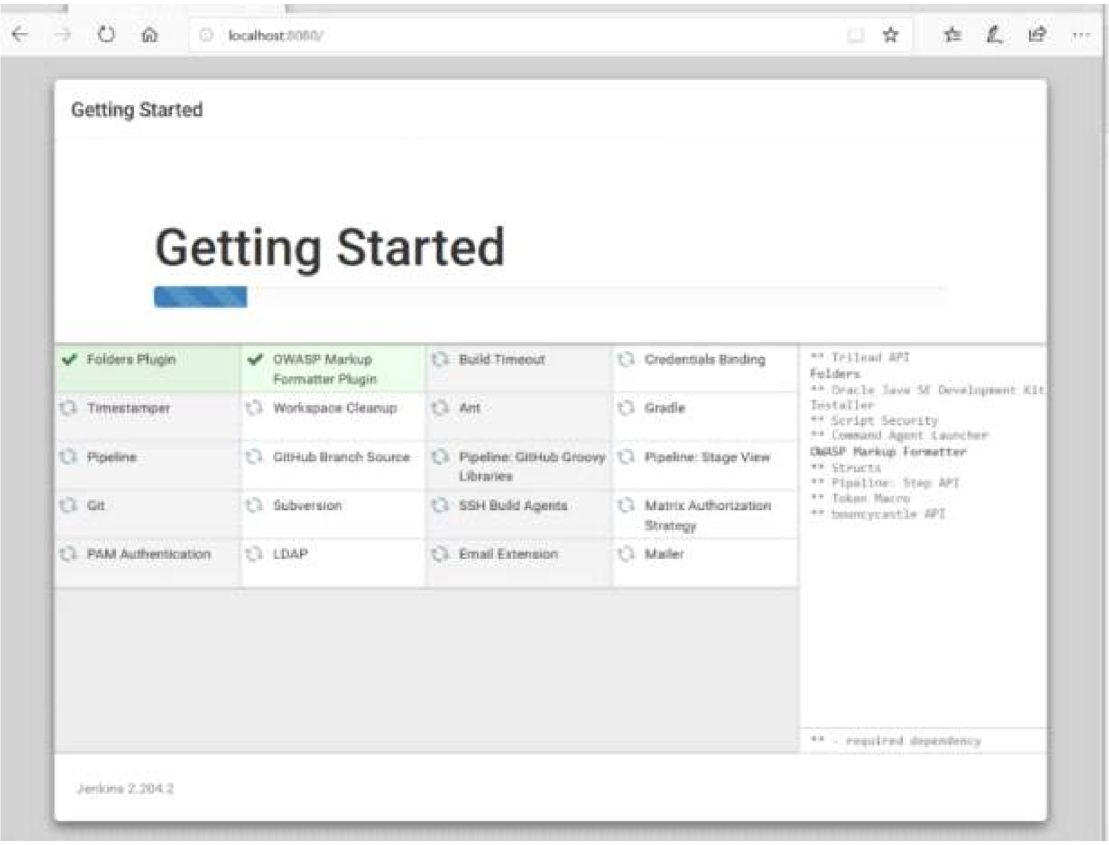
Getting Started

Getting Started

✓ Folders Plugin	✓ OWASP Markup Formatter Plugin	✓ Build Timeout	✓ Credentials Binding
✓ Timestamper	✓ Workspace Cleanup	✓ Ant	Gradle
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View
Git	Subversion	SSH Build Agents	Matrix Authorization Strategy
PAM Authentication	LDAP	Email Extension	Mailer

```
** bouncycastle API
Build Timeout
** Credentials
** Plain Credentials
** SSH Credentials
Credentials Binding
** SCM API
** Pipeline: API
Timestamper
** Pipeline: Supporting APIs
** Durable Task
** Pipeline: Nodes and
Processes
** JUnit
** Matrix Project
** Resource Disposer
Workspace Cleanup
Ant
** JavaScript GUI Lib: ACE
Editor bundle
** JavaScript GUI Lib: jQuery
bundles (jQuery and jQuery UI)
** Pipeline: SCM Step
** Pipeline: Groovy
** - required dependency
```

Jenkins 2.204.2



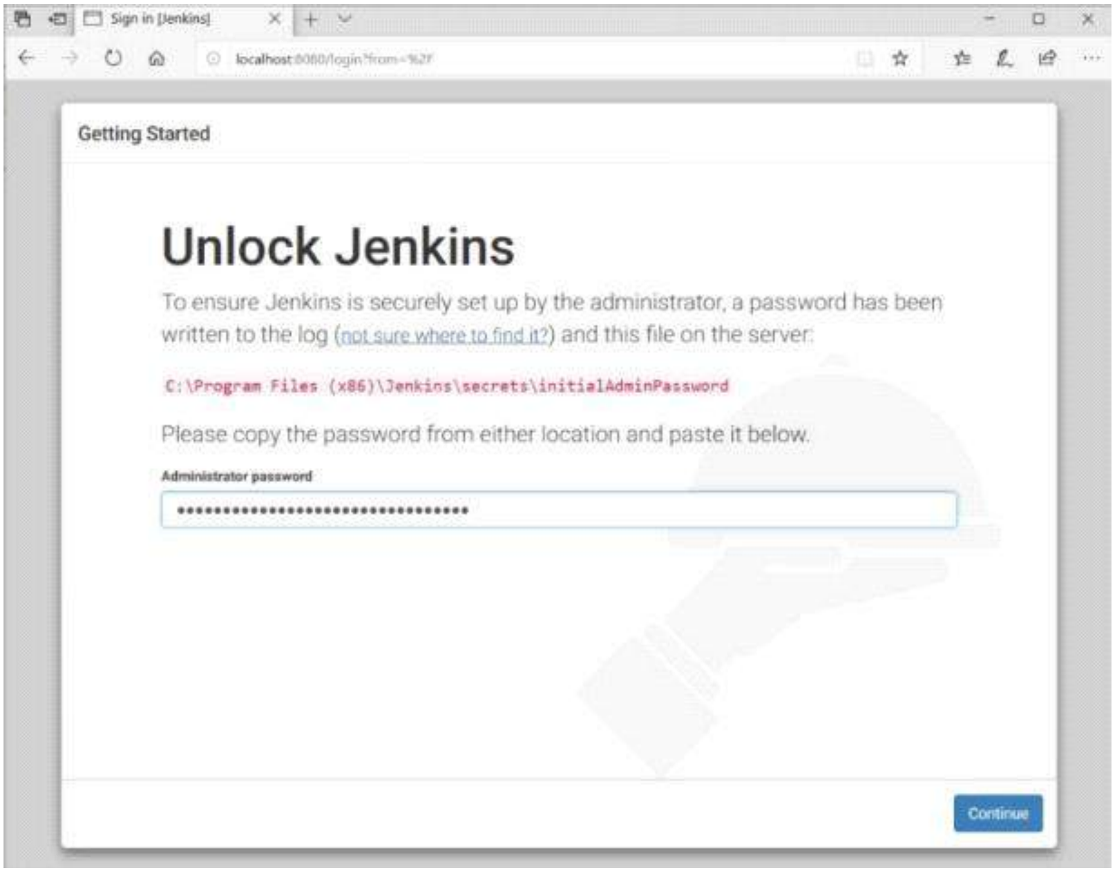
Getting Started

Getting Started

✓ Folders Plugin	✓ OWASP Markup Formatter Plugin	Build Timeout	Credentials Binding
Timestamper	Workspace Cleanup	Ant	Gradle
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View
Git	Subversion	SSH Build Agents	Matrix Authorization Strategy
PAM Authentication	LDAP	Email Extension	Mailer

```
** Trilead API
Folders
** Oracle Java SE Development Kit
Installer
** Script Security
** Command Agent Launcher
OWASP Markup Formatter
** Structs
** Pipeline: Step API
** Token Macro
** bouncycastle API
** - required dependency
```

Jenkins 2.204.2



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:

```
C:\Program Files (x86)\Jenkins\secrets\initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

••••••••••••••••••••••••••••••••

CONCLUSION : In this experiment we had understood Jenkins tool and installation of it.

ASSIGNMENT 5

AIM : To build Java Program using Jenkins

LO MAPPED : LO1 , LO3

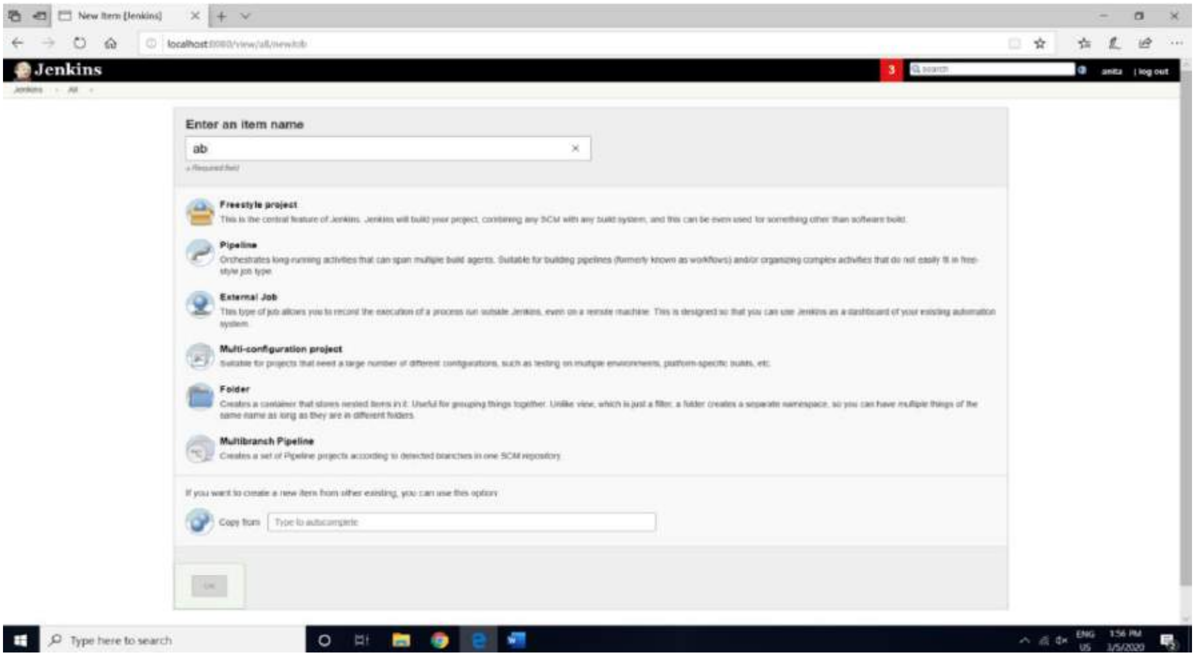
THEORY :

1. Create java Program.

```
public class hello{
public static void main(String[] args)
{
    for(int i=1;i<=15;i++)
        {
            System.out.println("hi sjcem"+i);
         }
}
 }
```

2. Create Jenkins job to run the program

In this select new item,



Enter an item name

ab

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

External Job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

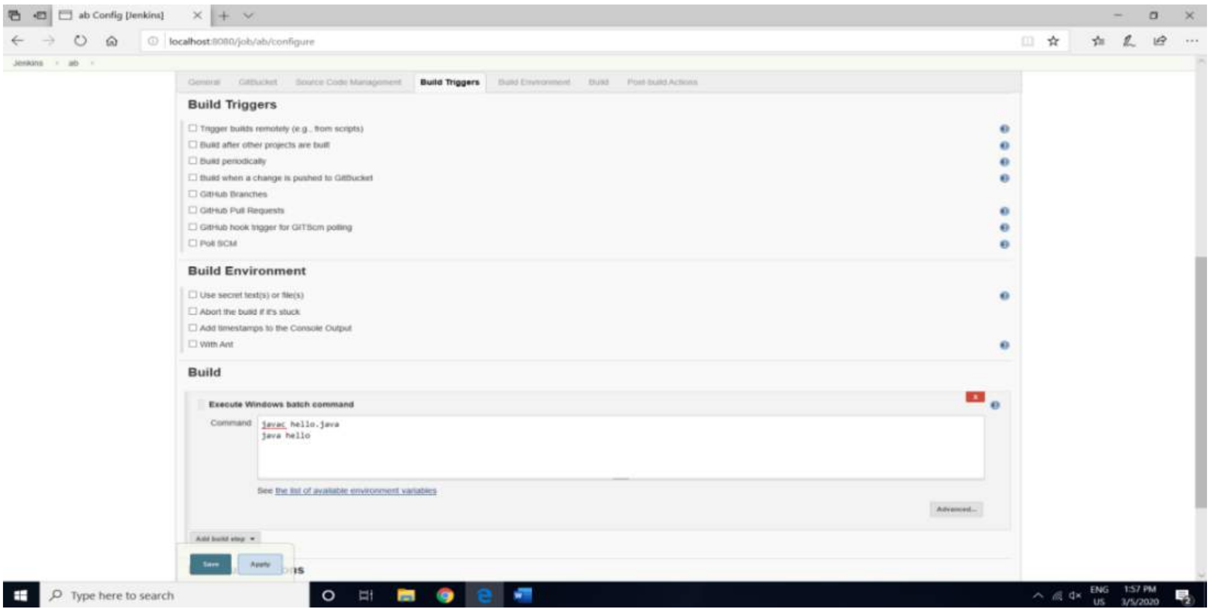
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

If you want to create a new item from other existing, you can use this option:

Copy from: Type to autocomplete

Execute windows batch command



localhost:8080/job/ab/configure

Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ Build when a change is pushed to GitBucket
- ☐ GitHub Branches
- ☐ GitHub Pull Requests
- ☐ GitHub hook trigger for GITScm polling
- ☐ Poll SCM

Build Environment

- ☐ Use secret text(s) or file(s)
- ☐ Abort the build if it's stuck
- ☐ Add timestamps to the Console Output
- ☐ With Ant

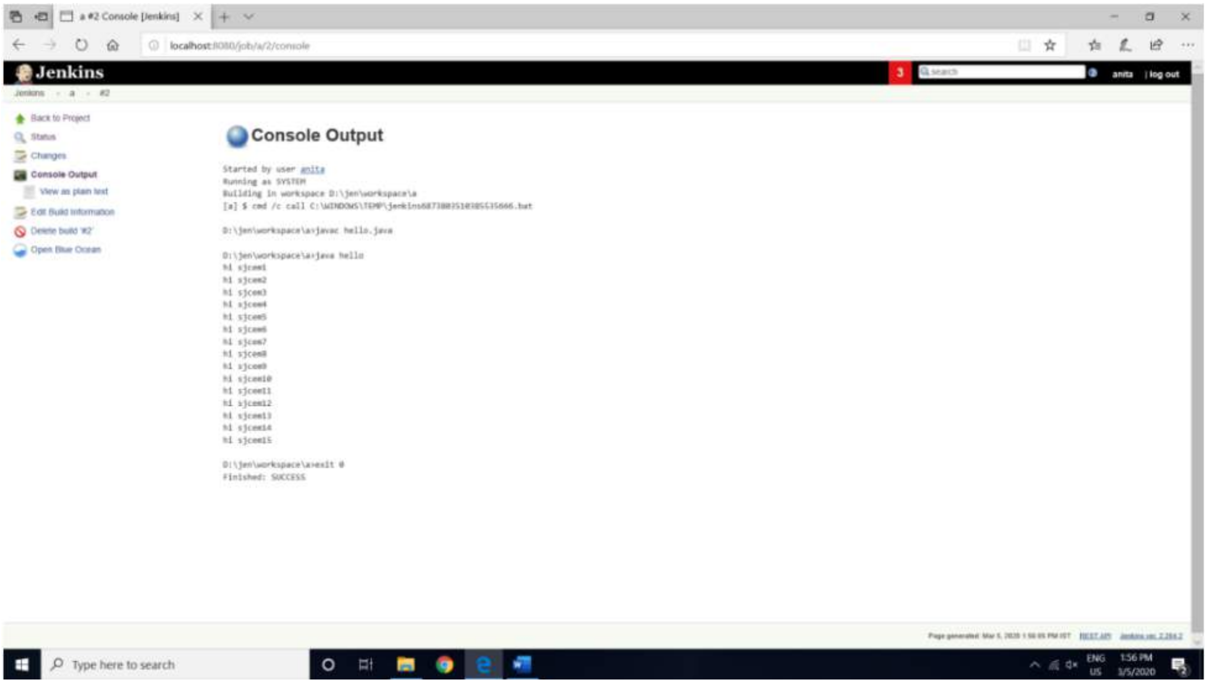
Build

Execute Windows batch command

Command

```
javac hello.java
java hello
```

See the list of available environment variables



localhost:8080/job/a/2/console

Console Output

```
Started by user anita
Running as SYSTEM
Building in workspace D:\jen\workspace\a
[a] $ cmd /c call C:\WINDOWS\TEMP\jenkins6873803510385535666.bat

D:\jen\workspace\a>javac hello.java

D:\jen\workspace\a>java hello
hi sjcem1
hi sjcem2
hi sjcem3
hi sjcem4
hi sjcem5
hi sjcem6
hi sjcem7
hi sjcem8
hi sjcem9
hi sjcem10
hi sjcem11
hi sjcem12
hi sjcem13
hi sjcem14
hi sjcem15

D:\jen\workspace\a>exit 0
Finished: SUCCESS
```

Conclusion: In this experiment we have executed Java project on Jenkins.

ASSIGNMENT 6

AIM : To build pipeline using Jenkins.

LO MAPPED : LO1 , LO3

THEORY :

Definition :-

Jenkins Pipeline (or simply "Pipeline" with a capital "P") is a suite of plugins which supports implementing and integrating continuous delivery pipelines into Jenkins. A continuous delivery (CD) pipeline is an automated expression of your process for getting software from version control right through to your users and customers. Every change to your software (committed in source control) goes through a complex process on its way to being released. This process involves building the software in a reliable and repeatable manner, as well as progressing the built software (called a "build") through multiple stages of testing and deployment. Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code" via the Pipeline domain-specific language (DSL) syntax. The definition of a Jenkins Pipeline is written into a text file (called a Jenkins file) which in turn can be committed to a project's source control repository. This is the foundation of "Pipeline-as-code"; treating the CD pipeline a part of the application to be versioned and reviewed like any other code.

Creating a Jenkins file and committing it to source control provides a number of immediate benefits:

- Automatically creates a Pipeline build process for all branches and pull requests.
- Code review/iteration on the Pipeline (along with the remaining source code).
- Audit trail for the Pipeline.
- Single source of truth [3] for the Pipeline, which can be viewed and edited by multiple members of the project.

While the syntax for defining a Pipeline, either in the web UI or with a Jenkins file is the same, it is generally considered best practice to define the Pipeline in a Jenkinsfile and check that in to source control.

Declarative versus Scripted Pipeline syntax

A Jenkinsfile can be written using two types of syntax - Declarative and Scripted

Declarative and Scripted Pipelines are constructed fundamentally differently. Declarative Pipeline is a more recent feature of Jenkins Pipeline which:

- provides richer syntactical features over Scripted Pipeline syntax, and is designed to make writing and reading Pipeline code easier

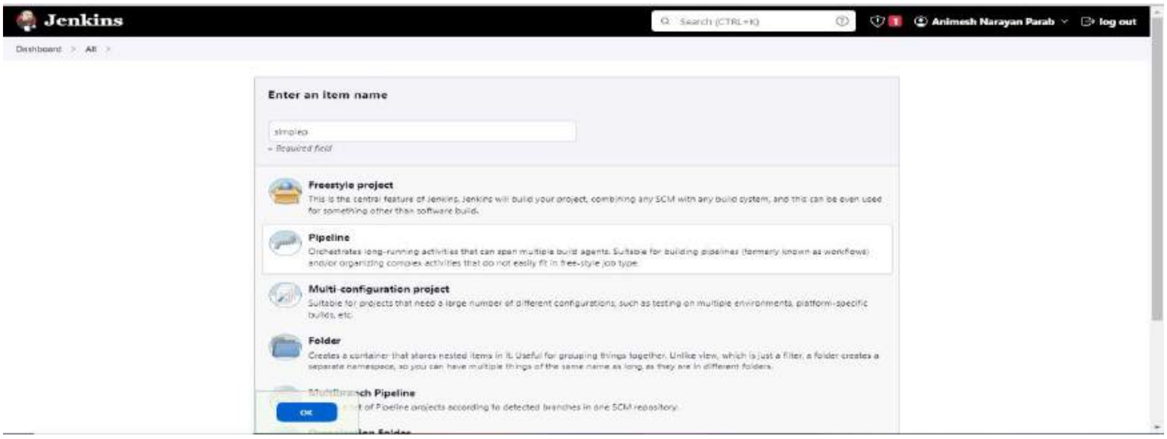
Many of the individual syntactical components (or "steps") written into a Jenkinsfile, however, are common to both Declarative and Scripted Pipeline. Read more about how these two types of syntax differ in Pipeline concepts and Pipeline syntax overview below **Why Pipeline?**

Jenkins is, fundamentally, an automation engine which supports a number of automation patterns. Pipeline adds a powerful set of automation tools onto Jenkins, supporting use cases that span from simple continuous integration to comprehensive CD pipelines. By modeling a series of related tasks, users can take advantage of the many features of Pipeline:

- Code: Pipelines are implemented in code and typically checked into source control, giving teams the ability to edit, review, and iterate upon their delivery pipeline.
- Durable: Pipelines can survive both planned and unplanned restarts of the Jenkins master. • Pausable: Pipelines can optionally stop and wait for human input or approval before continuing the Pipeline run.
- Versatile: Pipelines support complex real-world CD requirements, including the ability to fork/join, loop, and perform work in parallel.
- Extensible: The Pipeline plugin supports custom extensions to its DSL and multiple options for integration with other plugins.

While Jenkins has always allowed rudimentary forms of chaining Freestyle Jobs together to perform sequential tasks, Pipeline makes this concept a first-class citizen in Jenkins. Building on the core Jenkins value of extensibility, Pipeline is also extensible both by users with Pipeline Shared Libraries and by plugin developers.

Output :-



Enter an item name

simplep

» Required field

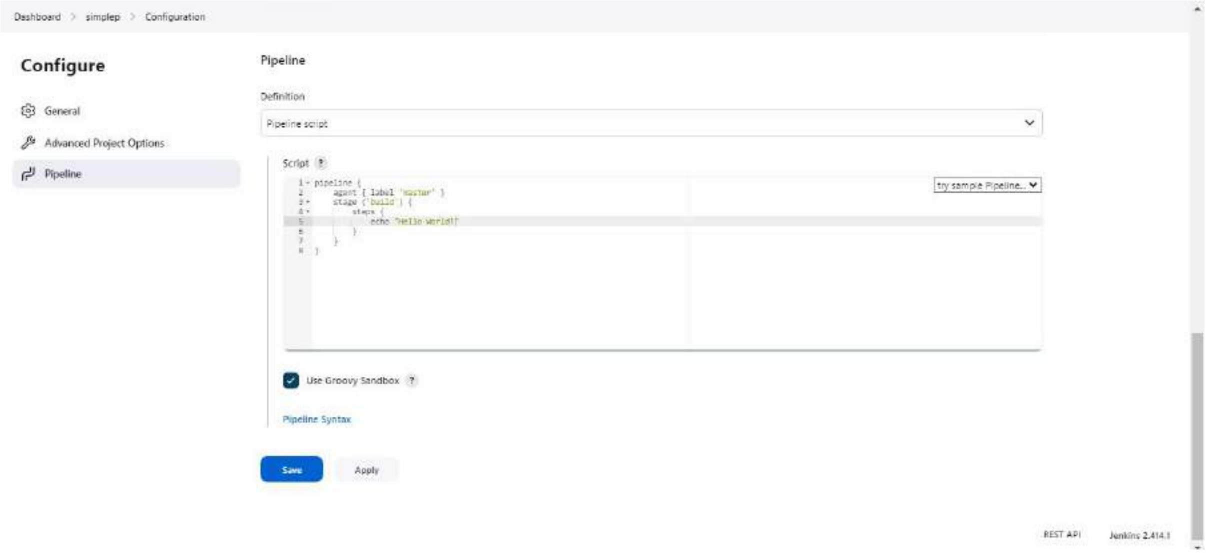
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
t of Pipeline projects according to detected branches in one SCM repository.



Dashboard > simplep > Configuration

Configure

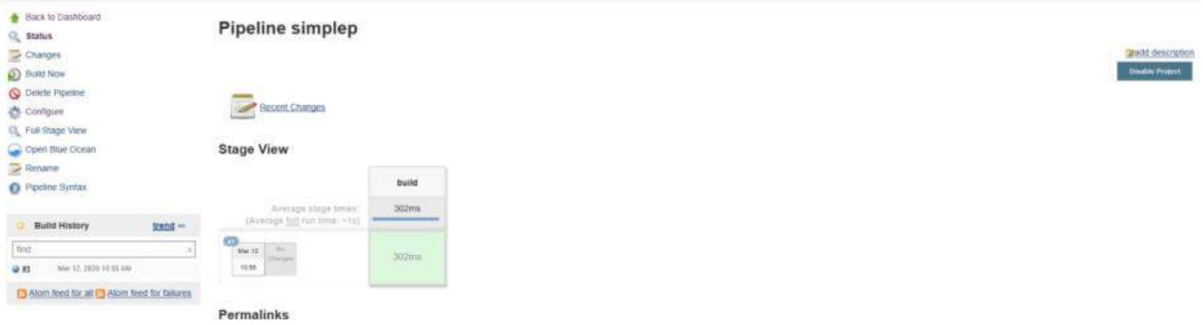
Pipeline

Definition: Pipeline script

Script

```
pipeline {
    agent { label 'master' }
    stage ('build') {
        steps {
            echo "Hello World!"
        }
    }
}
```

☑ Use Groovy Sandbox



Pipeline simplep

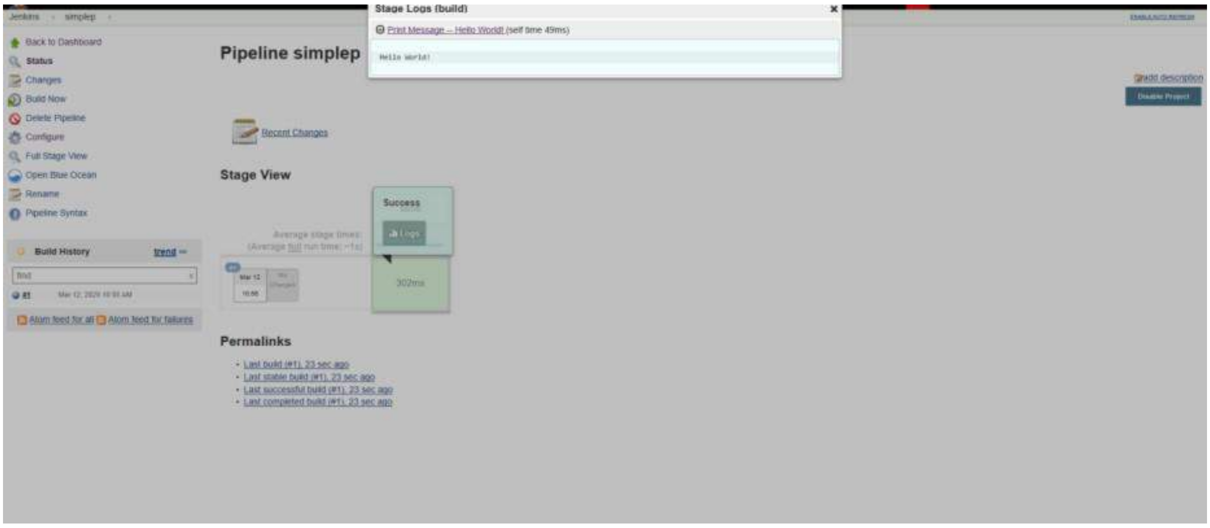
#1 Mar 12, 2020 10:55 AM

Stage View

Average stage times:
(Average full run time: ~1s)

	build
	302ms
#1 Mar 12 10:56 No Changes	302ms

Permalinks



Jenkins > simplep >

Stage Logs (build)

Print Message -- Hello World! (self time 49ms)

```
Hello World!
```

Pipeline simplep

Stage View

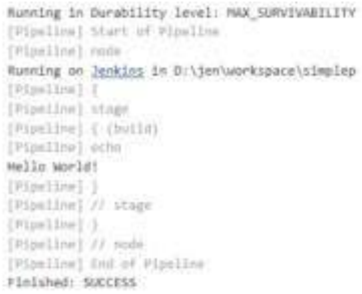
Success

Average stage times:
(Average full run time: ~1s)

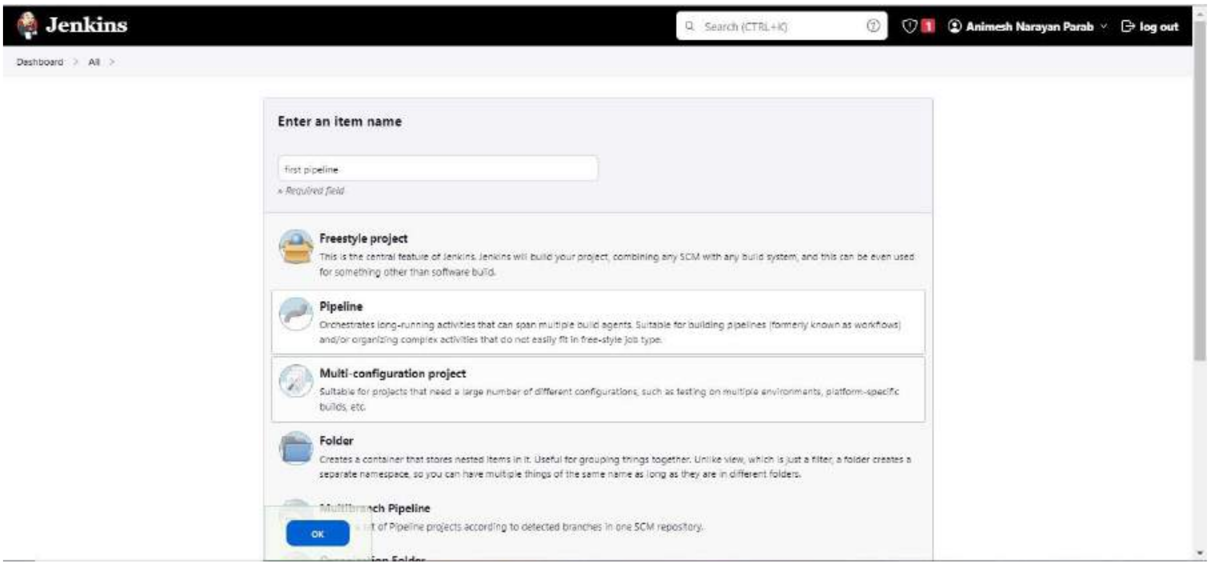
#1 Mar 12 10:56 No Changes 302ms

Permalinks

- Last build (#1), 23 sec ago
- Last stable build (#1), 23 sec ago
- Last successful build (#1), 23 sec ago
- Last completed build (#1), 23 sec ago



```
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in D:\jen\workspace\simplep
[Pipeline] {
[Pipeline] stage
[Pipeline] { (build)
[Pipeline] echo
Hello World!
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```



Enter an item name

first pipeline

» Required field

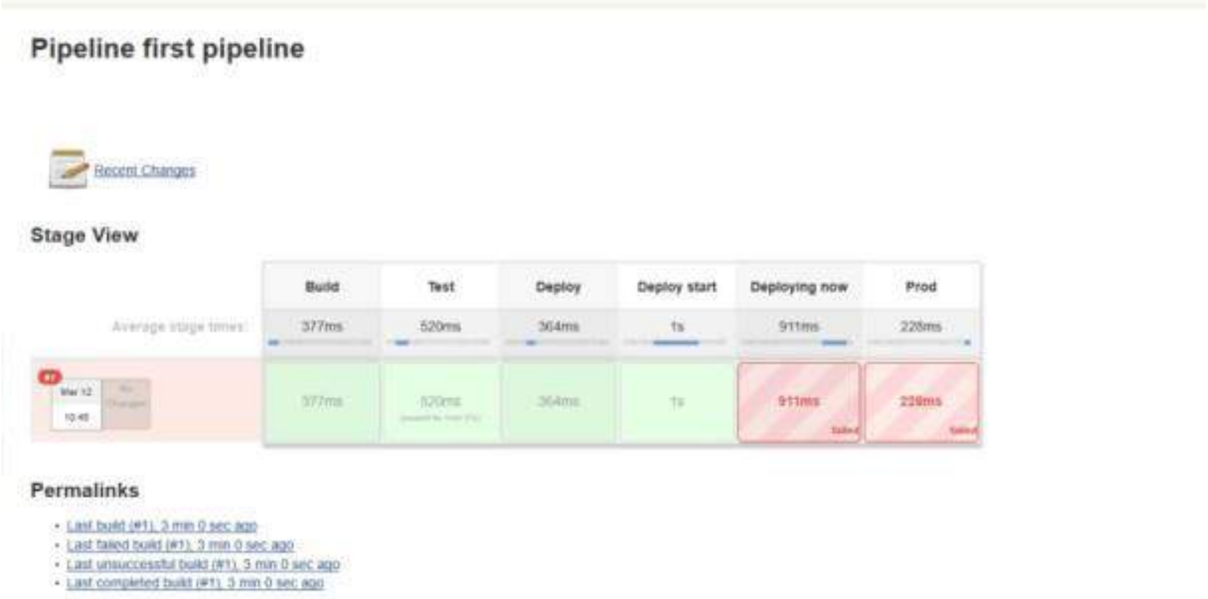
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
t of Pipeline projects according to detected branches in one SCM repository.



Pipeline first pipeline

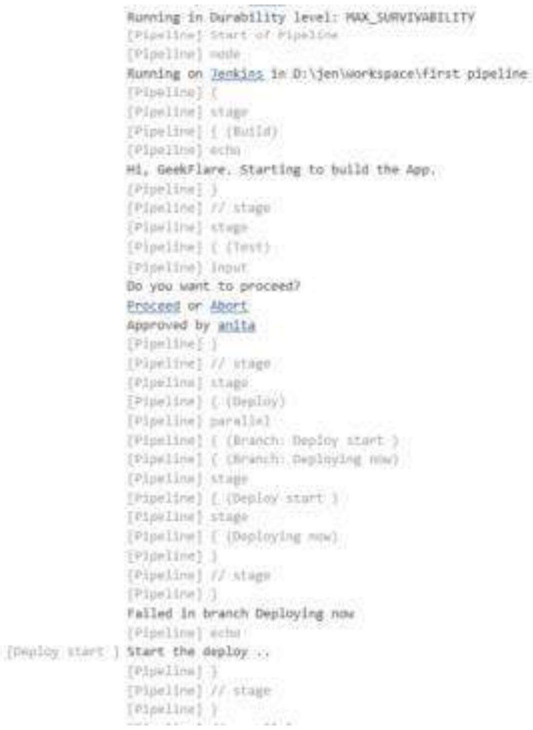
Recent Changes

Stage View

	Build	Test	Deploy	Deploy start	Deploying now	Prod
Average stage times:	377ms	520ms	364ms	1s	911ms	228ms
#1 Mar 12 10:48 No Changes	377ms	520ms	364ms	1s	911ms failed	228ms failed

Permalinks

- Last build (#1), 3 min 0 sec ago
- Last failed build (#1), 3 min 0 sec ago
- Last unsuccessful build (#1), 3 min 0 sec ago
- Last completed build (#1), 3 min 0 sec ago



```
                 Running in Durability level: MAX_SURVIVABILITY
                 [Pipeline] Start of Pipeline
                 [Pipeline] node
                 Running on Jenkins in D:\jen\workspace\first pipeline
                 [Pipeline] {
                 [Pipeline] stage
                 [Pipeline] { (Build)
                 [Pipeline] echo
                 Hi, GeekFlare. Starting to build the App.
                 [Pipeline] }
                 [Pipeline] // stage
                 [Pipeline] stage
                 [Pipeline] { (Test)
                 [Pipeline] input
                 Do you want to proceed?
                 Proceed or Abort
                 Approved by anita
                 [Pipeline] }
                 [Pipeline] // stage
                 [Pipeline] stage
                 [Pipeline] { (Deploy)
                 [Pipeline] parallel
                 [Pipeline] { (Branch: Deploy start )
                 [Pipeline] { (Branch: Deploying now)
                 [Pipeline] stage
                 [Pipeline] { (Deploy start )
                 [Pipeline] stage
                 [Pipeline] { (Deploying now)
                 [Pipeline] }
                 [Pipeline] // stage
                 [Pipeline] }
                 Failed in branch Deploying now
                 [Pipeline] echo
[Deploy start ]  Start the deploy ..
                 [Pipeline] }
                 [Pipeline] // stage
                 [Pipeline] }
```

CONCLUSION : In this experiment we had created pipeline using scripting language .

NAME : SOUMIL SALVI

ROLL NO : 104

ASSIGNMENT 7

AIM : To understand Docker architecture and container life cycle, install docker deploy container in Docker.

LO MAPPED : LO1,LO5

THEORY :
Docker is a software platform for building applications based on containers — small and lightweight execution environments that make shared use of the operating system kernel but otherwise run in isolation from one another. While containers as a concept have been around for some time, Docker, an open source project launched in 2013, helped popularize the technology, and has helped drive the trend towards containerization and microservices in software development that has come to be known as cloud-native development.

What are containers?

One of the goals of modern software development is to keep applications on the same host or cluster isolated from one another so they don't unduly interfere with each other's operation or maintenance. This can be difficult, thanks to the packages, libraries, and other software components required for them to run. One solution to this problem has been virtualmachines, which keep applications on the same hardware entirely separate, and reduce conflicts among software components and competition for hardware resources to a minimum. But virtual machines are bulky—each requires its own OS, so is typically gigabytes in size—and difficult to maintain and upgrade.

Containers, by contrast, isolate applications' execution environments from one another, but share the underlying OS kernel. They're typically measured in megabytes, use far fewer resources than VMs, and start up almost immediately. They can be packed far more densely on the same hardware and spun up and down en masse with far less effort and overhead. Containers provide a highly efficient and highly granular mechanism for combining software components into the kinds of application and service stacks needed in a modern enterprise, and for keeping those software components updated and maintained.

Creation of docker Container:

The structure is docker run <options> <image-name>.

```
$ docker search ubuntu
NAME DESCRIPTION STARS OFFICIAL
AUTOMATED
ubuntu Ubuntu is a Debian-based Linux operating sys... 10610
[OK] dorowu/ubuntu-desktop-lxde-vnc Docker image to provide HTML5 VNC
interface ... 404 [OK] rastasheep/ubuntu-sshd Dockerized SSH service, built on top of offi...
243 [OK] consol/ubuntu-xfce-vnc Ubuntu container with "headless" VNC session...
211 [OK] ubuntu-upstart Upstart is an event-based replacement for th... 106
[OK] ansible/ubuntu14.04-ansible Ubuntu 14.04 LTS with ansible 98
[OK] neurodebian NeuroDebian provides neuroscience research s... 67
[OK]

1and1internet/ubuntu-16-nginx-php-phpmyadmin-mysql-5 ubuntu-16-nginx-php- phpmyadmin-mysql-5 50 [OK]

ubuntu-debootstrap debootstrap --variant=minbase --components=m...
43 [OK] nuagebec/ubuntu Simple always updated Ubuntu docker images w...
24 [OK]
i386/ubuntu Ubuntu is a Debian-based Linux operating sys... 19
1and1internet/ubuntu-16-apache-php-5.6 ubuntu-16-apache-php-5.6 14 [OK]
1and1internet/ubuntu-16-apache-php-7.0 ubuntu-16-apache-php-7.0
13 [OK]
```

```
ppc64le/ubuntu Ubuntu is a Debian-based Linux operating sys... 13

1and1internet/ubuntu-16-nginx-php-phpmyadmin-mariadb-10 ubuntu-16-nginx-php- phpmyadmin-mariadb-10 11 [OK]

1and1internet/ubuntu-16-nginx-php-5.6-wordpress-4 ubuntu-16-nginx-php-5.6-wordpress-4 7 [OK]
1and1internet/ubuntu-16-apache-php-7.1 ubuntu-16-apache-php-7.1 6
[OK] darksheer/ubuntu Base Ubuntu Image -- Updated hourly 5
[OK]
1and1internet/ubuntu-16-nginx-php-7.0 ubuntu-16-nginx-php-7.0 4
[OK] pivotaldata/ubuntu A quick freshening-up of the base Ubuntu doc... 4
pivotaldata/ubuntu16.04-build Ubuntu 16.04 image for GPDB compilation 2
1and1internet/ubuntu-16-php-7.1 ubuntu-16-php-7.1 1
[OK]
1and1internet/ubuntu-16-sshd ubuntu-16-sshd 1
[OK] smartentry/ubuntu ubuntu with smartentry 1
[OK]
pivotaldata/ubuntu-gpdb-dev Ubuntu images for GPDB development 1
$ docker run -d ubuntu
f97b467f2b3153ef6cf89435391a465f2d98a6ac23dc1b8179ee8cae1eca07f5
$ docker run -d redis
f43d427f2174cb4105bddd638f67613dc890e8d5568b3a76a9796a9d77c5cf84
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
f43d427f2174 redis "docker-entrypoint.s..." 4 minutes ago Up 4 minutes
```

```
6379/tcp dazzling_payne
```

Running ubuntu and redis images

The launched container is running in the background, the docker ps command lists all running containers, the image used to start the container and uptime.

```
$ docker run -d redis

f43d427f2174cb4105bddd638f67613dc890e8d5568b3a76a9796a9d77c5cf84

$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

f43d427f2174 redis "docker-entrypoint.s..." 4 minutes ago Up 4 minutes

6379/tcp dazzling_payne $ docker run -d ubuntu
fb0431e37c815aeab64327bc98b17efd70f18defbf705055324f8f5c620e3994

$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS

PORTS NAMES

fb0431e37c81 ubuntu "/bin/bash" 4 seconds ago Up 3 seconds
focused_albattani f43d427f2174 redis "docker-entrypoint.s..." 4 minutes ago
Up 4 minutes
```

6379/tcp dazzling_payne docker inspect <friendly-name|container-id> provides more details about a running container, such as IP address.

Shows ubuntu image details based on container id

```
$ docker inspect fb0431e37c81

[

{

"Id":
"fb0431e37c815aeab64327bc98b17efd70f18defbf705055324f8f5c620e3994",

"Created": "2020-03-12T08:29:05.05550992Z",
"Path": "/bin/bash",
"Args": [],
"State": {
"Status": "exited",
"Running": false,
"Paused": false,
"Restarting": false,
"OOMKilled": false,
"Dead": false,
"Pid": 0,
"ExitCode": 0,
"Error": "",
"StartedAt": "2020-03-12T08:29:05.477881863Z",
"FinishedAt": "2020-03-12T08:29:05.463162864Z"
},
"Image":
"sha256:f975c50357489439eb9145dbfa16bb7cd06c02c31aa4df45c77de4d2ba
a4e232",
"ResolvConfPath":
"/var/lib/docker/containers/fb0431e37c815aeab64327bc98b17efd70f18defbf7
05055324f8f5c6
20 e3994/resolv.conf", "HostnamePath":
"/var/lib/docker/containers/fb0431e37c815aeab64327bc98b17efd70f18defbf7
05055324f8f5c6
20 e3994/hostname", "HostsPath":
"/var/lib/docker/containers/fb0431e37c815aeab64327bc98b17efd70f18defbf7
05055324f8f5c6
20 e3994/hosts",
"LogPath":
"/var/lib/docker/containers/fb0431e37c815aeab64327bc98b17efd70f18defbf7
05055324f8f5c6
20
e3994/fb0431e37c815aeab64327bc98b17efd70f18defbf705055324f8f5c620e3
994-json.log", "Name": "/focused_albattani",
"RestartCount": 0,
"Driver": "overlay",
"Platform": "linux",
"MountLabel": "",
"ProcessLabel": "",
"AppArmorProfile": "",
"ExecIDs": null,
"HostConfig": {
"Binds": null,
"ContainerIDFile": "",
"LogConfig": {
"Type": "json-file",
"Config": {}
},
"NetworkMode": "default",
"PortBindings": {},
"RestartPolicy": {
"Name": "no",
"MaximumRetryCount": 0
},
"AutoRemove": false,
"VolumeDriver": "",
"VolumesFrom": null,
"CapAdd": null,
"CapDrop": null,
"Dns": [],
"DnsOptions": [],
"DnsSearch": [],
"ExtraHosts": null,
"GroupAdd": null,
"IpcMode": "shareable",
"Cgroup": "",
"Links": null,
"OomScoreAdj": 0,
"PidMode": "",
"Privileged": false,
"PublishAllPorts": false,
"ReadonlyRootfs": false,
"SecurityOpt": null,
"UTSMode": "",
"UsernsMode": "",
"ShmSize": 67108864,
"Runtime": "runc",
"ConsoleSize": [
0,
0
],
"Isolation": "",
"CpuShares": 0,
"Memory": 0,
"NanoCpus": 0,
"CgroupParent": "",
"BlkioWeight": 0,
"BlkioWeightDevice": [],
"BlkioDeviceReadBps": null,
"BlkioDeviceWriteBps": null,
"BlkioDeviceReadIOps": null,
"BlkioDeviceWriteIOps": null,
"CpuPeriod": 0,
"CpuQuota": 0,
"CpuRealtimePeriod": 0,
"CpuRealtimeRuntime": 0, "CpusetCpus": "",
"CpusetMems": "",
"Devices": [],
"DeviceCgroupRules": null,
"DiskQuota": 0,
"KernelMemory": 0,
"MemoryReservation": 0,
"MemorySwap": 0,
"MemorySwappiness": null,
"OomKillDisable": false,
"PidsLimit": 0,
"Ulimits": null,
"CpuCount": 0,

"CpuPercent": 0,

"IOMaximumIOps": 0,

"IOMaximumBandwidth": 0

},

"GraphDriver": {

"Data": {

"LowerDir":

"/var/lib/docker/overlay/200efe94270e561b0044f3e61a46acccfbba0c73dd98b
262b34bef6214 06 66d3/root",

"MergedDir":

"/var/lib/docker/overlay/3a9eca851902b1f1945e59d9e574aae2bbb4f951f5c2
d5acd95dccea45 2f2 623/merged", "UpperDir":
"/var/lib/docker/overlay/3a9eca851902b1f1945e59d9e574aae2bbb4f951f5c2
d5acd95dccea45 2f2 623/upper",

"WorkDir":

"/var/lib/docker/overlay/3a9eca851902b1f1945e59d9e574aae2bbb4f951f5c2
d5acd95dccea45 2f2

0.0.0.0:32768

$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS

PORTS NAMES

4b2d84fc668d redis:latest "docker-entrypoint.s..." 9 seconds ago Up 8 seconds

0.0.0.0:32768->6379/tcp redisDynamic

51b3ef84a67d redis:latest "docker-entrypoint.s..." 18 seconds ago Up 17
seconds

0.0.0.0:6379->6379/tcp redisHostPort

1b5b5c679391 redis "docker-entrypoint.s..." 27 seconds ago Up 25 seconds
6379/tcp boring_clarke
```

Redis image stores logs and data into a /data directory. Any data which needs to be saved on the

Docker Host, and not inside containers, should be stored in /opt/docker/data/redis.

The complete command to solve the task is

```
$ docker run -d --name redisMapped -v /opt/docker/data/redis:/data redis

ab4358aa8869edc1811197c536141c70e662edc97ce03365dc487b7ab436c387

$ docker run ubuntu ps

PID TTY TIME CMD

1 ? 00:00:00 ps

$ docker run -it ubuntu bash

root@5879c2ebc254:/#

root@5879c2ebc254:/# ls bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
```

We'll make a change by echoing some text into the container's /tmp directory, then use cat to verify that it was successfully saved.

```
root@5879c2ebc254:/# echo "Example1" > /tmp/Example1.txt
root@5879c2ebc254:/# cat /tmp/Example1.txt

Example1
```

Exit from current container root@5879c2ebc254:/# exit

```
exit

$
```

CONCLUSION : In this experiment we Understood Docker concepts and Created first docker container

ASSIGNMENT NO 8

Aim: Understand and Install Nagios , Monitor host using Nagios

Objectives

1.Understand Nagios

2.Understand installation process for Nagios on Ubuntu

3.Monitor Localhost using different parameters.

Theory :

What is Nagios?

Nagios 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

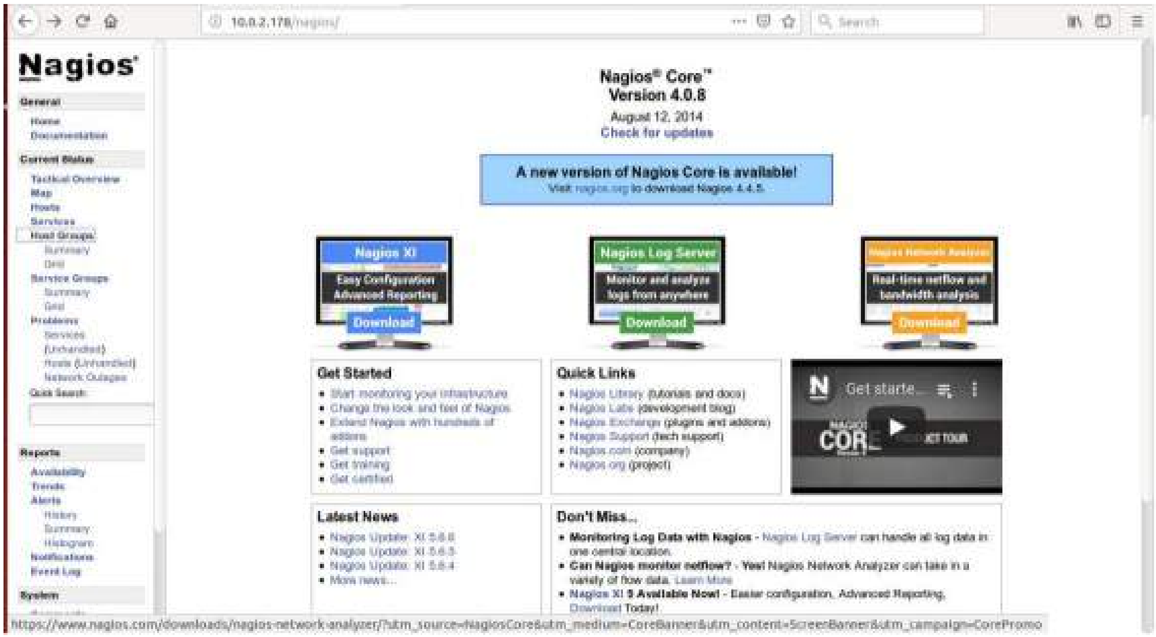
• Helps you to find the root cause of the problem which allows you to get the permanent

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

web interface



Nagios® Core™
Version 4.0.8
August 12, 2014
Check for updates

A new version of Nagios Core is available!
Visit nagios.org to download Nagios 4.4.5.

Get Started

- Start monitoring your infrastructure
- Change the look and feel of Nagios
- Extend Nagios with hundreds of addons
- Get support
- Get training
- Get certified

Quick Links

- Nagios Library (tutorials and docs)
- Nagios Labs (development blog)
- Nagios Exchange (plugins and addons)
- Nagios Support (tech support)
- Nagios.com (company)
- Nagios.org (project)

Latest News

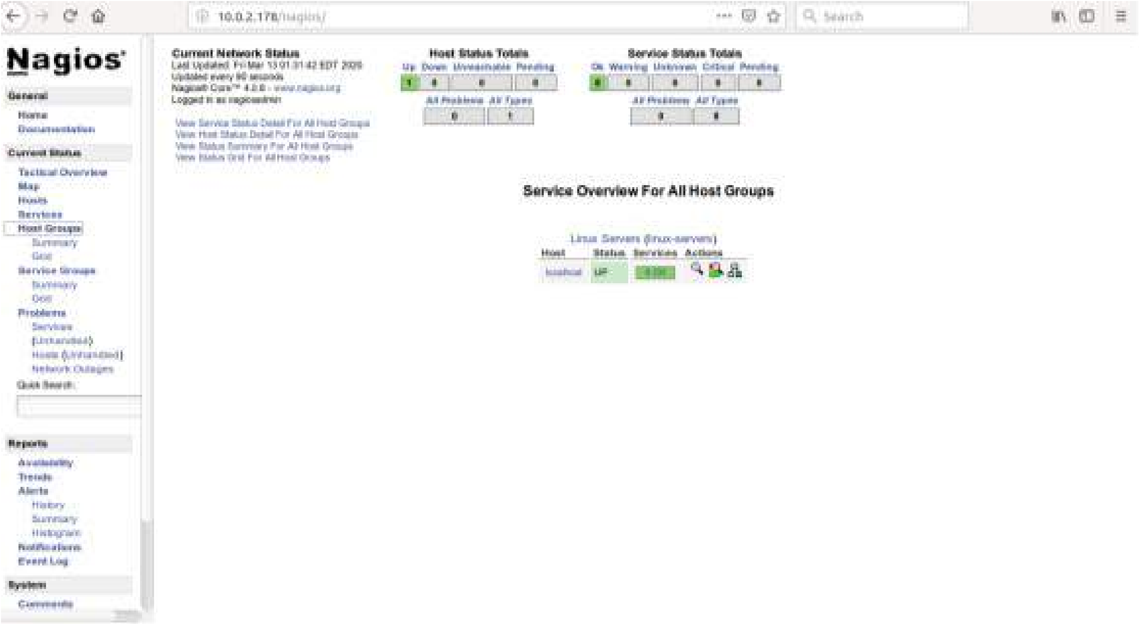
- Nagios Update: XI 5.6.6
- Nagios Update: XI 5.6.5
- Nagios Update: XI 5.6.4
- More news...

Don't Miss...

- **Monitoring Log Data with Nagios** - Nagios Log Server can handle all log data in one central location.
- **Can Nagios monitor netflow?** - **Yes!** Nagios Network Analyzer can take in a variety of flow data. Learn More
- Nagios XI **5 Available Now!** - Easier configuration, Advanced Reporting. Download Today!

https://www.nagios.com/downloads/nagios-network-analyzer/?utm_source=NagiosCore&utm_medium=CoreBanner&utm_content=ScreenBanner&utm_campaign=CorePromo

Click on Host Groups



Current Network Status
Last Updated: Fri Mar 13 01:31:42 EDT 2020
Updated every 90 seconds
Nagios® Core™ 4.0.8 - www.nagios.org
Logged in as nagiosadmin

View Service Status Detail For All Host Groups
View Host Status Detail For All Host Groups
View Status Summary For All Host Groups
View Status Grid For All Host Groups

Host Status Totals

Up	Down	Unreachable	Pending
1	0	0	0

All Problems	All Types
0	1

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
8	0	0	0	0

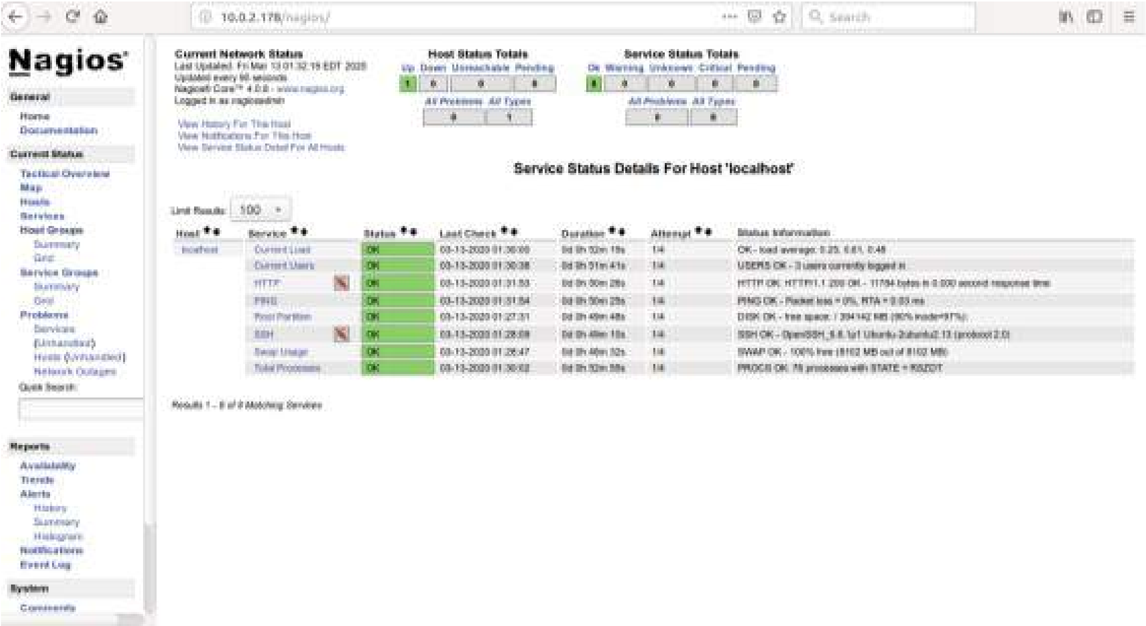
All Problems	All Types
0	8

Service Overview For All Host Groups

Linux Servers (linux-servers)

Host	Status	Services	Actions
localhost	UP	8 OK	

Click on localhost



Current Network Status
Last Updated: Fri Mar 13 01:32:19 EDT 2020
Updated every 90 seconds
Nagios® Core™ 4.0.8 - www.nagios.org
Logged in as nagiosadmin

View History For This Host
View Notifications For This Host
View Service Status Detail For All Hosts

Host Status Totals

Up	Down	Unreachable	Pending
1	0	0	0

All Problems	All Types
0	1

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
8	0	0	0	0

All Problems	All Types
0	8

Service Status Details For Host 'localhost'

Limit Results: 100

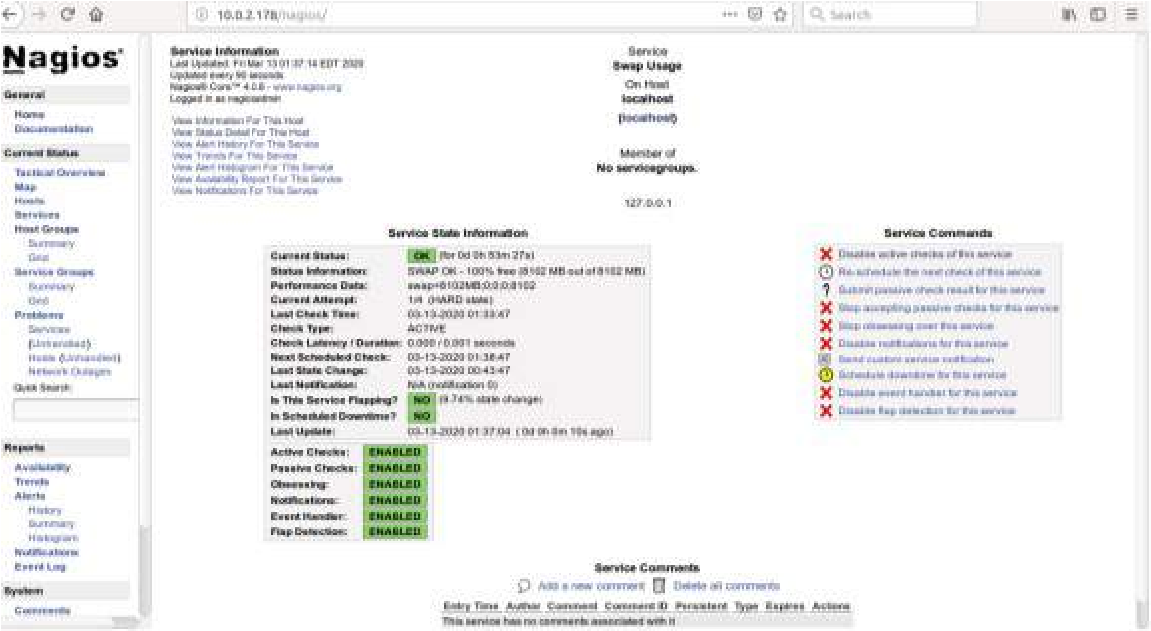
Host	Service	Status	Last Check	Duration	Attempt	Status Information
localhost	Current Load	OK	03-13-2020 01:30:00	0d 0h 52m 19s	1/4	OK - load average: 0.25, 0.61, 0.48
	Current Users	OK	03-13-2020 01:30:38	0d 0h 51m 41s	1/4	USERS OK - 3 users currently logged in
	HTTP	OK	03-13-2020 01:31:53	0d 0h 50m 26s	1/4	HTTP OK: HTTP/1.1 200 OK - 11784 bytes in 0.000 second response time
	PING	OK	03-13-2020 01:31:54	0d 0h 50m 25s	1/4	PING OK - Packet loss = 0%, RTA = 0.03 ms
	Root Partition	OK	03-13-2020 01:27:31	0d 0h 49m 48s	1/4	DISK OK - free space: / 394142 MB (90% inode=97%):
	SSH	OK	03-13-2020 01:28:09	0d 0h 48m 10s	1/4	SSH OK - OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.13 (protocol 2.0)
	Swap Usage	OK	03-13-2020 01:28:47	0d 0h 46m 32s	1/4	SWAP OK - 100% free (8102 MB out of 8102 MB)
	Total Processes	OK	03-13-2020 01:30:02	0d 0h 52m 58s	1/4	PROCS OK: 76 processes with STATE = RSZDT

Results 1 - 8 of 8 Matching Services

you can click on a particular entry to view more details about it. For example, here is the “Swap

Usage” information page for the current node. You can also issue a number of service commands

from the right, such as disabling the check.



Service Information
Last Updated: Fri Mar 13 01:37:14 EDT 2020
Updated every 90 seconds
Nagios® Core™ 4.0.8 - www.nagios.org
Logged in as nagiosadmin

View Information For This Host
View Status Detail For This Host
View Alert History For This Service
View Trends For This Service
View Alert Histogram For This Service
View Availability Report For This Service
View Notifications For This Service

Service
Swap Usage
On Host
localhost
(localhost)

Member of
No servicegroups.

127.0.0.1

Service State Information

Current Status:	OK (for 0d 0h 53m 27s)
Status Information:	SWAP OK - 100% free (8102 MB out of 8102 MB)
Performance Data:	swap=8102MB;0;0;0;8102
Current Attempt:	1/4 (HARD state)
Last Check Time:	03-13-2020 01:33:47
Check Type:	ACTIVE
Check Latency / Duration:	0.000 / 0.001 seconds
Next Scheduled Check:	03-13-2020 01:38:47
Last State Change:	03-13-2020 00:43:47
Last Notification:	N/A (notification 0)
Is This Service Flapping?	NO (9.74% state change)
In Scheduled Downtime?	NO
Last Update:	03-13-2020 01:37:04 (0d 0h 0m 10s ago)

Active Checks:	ENABLED
Passive Checks:	ENABLED
Obsessing:	ENABLED
Notifications:	ENABLED
Event Handler:	ENABLED
Flap Detection:	ENABLED

Service Commands

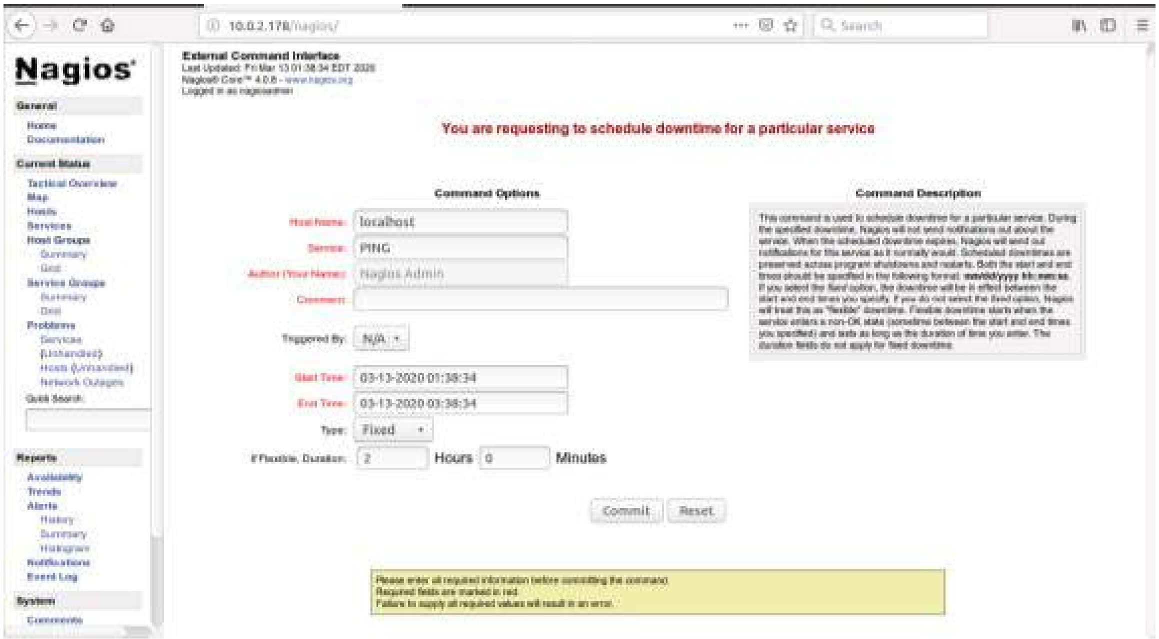
- Disable active checks of this service
- Re-schedule the next check of this service
- Submit passive check result for this service
- Stop accepting passive checks for this service
- Stop obsessing over this service
- Disable notifications for this service
- Send custom service notification
- Schedule downtime for this service
- Disable event handler for this service
- Disable flap detection for this service

Service Comments

Add a new comment   Delete all comments

Entry Time	Author	Comment	Comment ID	Persistent	Type	Expires	Actions
This service has no comments associated with it							

Reschedule downtime of ping service:



External Command Interface
Last Updated: Fri Mar 13 01:38:34 EDT 2020
Nagios® Core™ 4.0.8 - www.nagios.org
Logged in as nagiosadmin

You are requesting to schedule downtime for a particular service

Command Options

Host Name: localhost
Service: PING
Author (Your Name): Nagios Admin
Comment:
Triggered By: N/A
Start Time: 03-13-2020 01:38:34
End Time: 03-13-2020 03:38:34
Type: Fixed
If Flexible, Duration: 2 Hours 0 Minutes

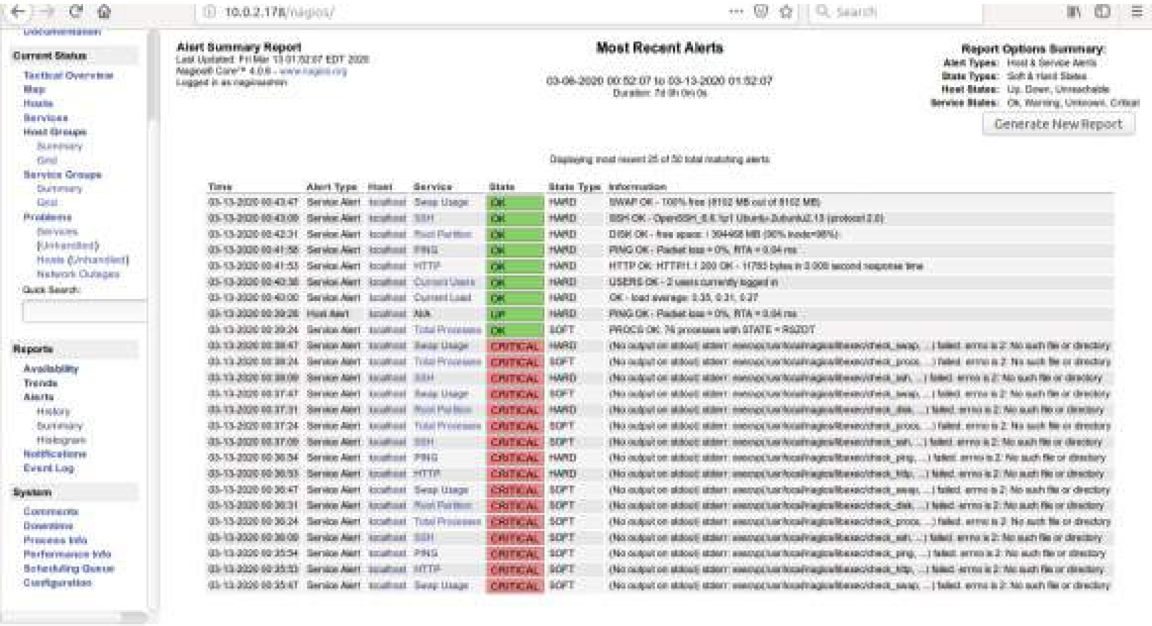
Commit Reset

Command Description

This command is used to schedule downtime for a particular service. During the specified downtime, Nagios will not send notifications out about the service. When the scheduled downtime expires, Nagios will send out notifications for this service as it normally would. Scheduled downtimes are preserved across program shutdowns and restarts. Both the start and end times should be specified in the following format: **mm/dd/yyyy hh:mm:ss**. If you select the fixed option, the downtime will be in effect between the start and end times you specify. If you do not select the fixed option, Nagios will treat this as "flexible" downtime. Flexible downtime starts when the service enters a non-OK state (sometime between the start and end times you specified) and lasts as long as the duration of time you enter. The duration fields do not apply for fixed downtime.

Please enter all required information before committing the command.
Required fields are marked in red.
Failure to supply all required values will result in an error.

Summary report of localhost



Alert Summary Report
Last Updated: Fri Mar 13 01:52:07 EDT 2020
Nagios® Core™ 4.0.8 - www.nagios.org
Logged in as nagiosadmin

Most Recent Alerts

03-06-2020 00:52:07 to 03-13-2020 01:52:07
Duration: 7d 0h 0m 0s

Report Options Summary:
Alert Types: Host & Service Alerts
State Types: Soft & Hard States
Host States: Up, Down, Unreachable
Service States: Ok, Warning, Unknown, Critical

Displaying most recent 25 of 50 total matching alerts

Time	Alert Type	Host	Service	State	State Type	Information
03-13-2020 00:43:47	Service Alert	localhost	Swap Usage	OK	HARD	SWAP OK - 100% free (8102 MB out of 8102 MB)
03-13-2020 00:43:09	Service Alert	localhost	SSH	OK	HARD	SSH OK - OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.13 (protocol 2.0)
03-13-2020 00:42:31	Service Alert	localhost	Root Partition	OK	HARD	DISK OK - free space: / 304468 MB (90% inode=98%);
03-13-2020 00:41:58	Service Alert	localhost	PING	OK	HARD	PING OK - Packet loss = 0%, RTA = 0.04 ms
03-13-2020 00:41:53	Service Alert	localhost	HTTP	OK	HARD	HTTP OK: HTTP/1.1 200 OK - 11783 bytes in 0.008 second response time
03-13-2020 00:40:38	Service Alert	localhost	Current Users	OK	HARD	USERS OK - 2 users currently logged in
03-13-2020 00:40:00	Service Alert	localhost	Current Load	OK	HARD	OK - load average: 0.35, 0.31, 0.27
03-13-2020 00:39:28	Host Alert	localhost	N/A	UP	HARD	PING OK - Packet loss = 0%, RTA = 0.04 ms
03-13-2020 00:39:24	Service Alert	localhost	Total Processes	OK	SOFT	PROCS OK: 74 processes with STATE = RSZDT
03-13-2020 00:38:47	Service Alert	localhost	Swap Usage	CRITICAL	HARD	(No output on stdout) stderr: execvp(/usr/local/nagios/libexec/check_swap, ...) failed. errno is 2: No such file or directory
03-13-2020 00:38:24	Service Alert	localhost	Total Processes	CRITICAL	SOFT	(No output on stdout) stderr: execvp(/usr/local/nagios/libexec/check_procs, ...) failed. errno is 2: No such file or directory
03-13-2020 00:38:09	Service Alert	localhost	SSH	CRITICAL	HARD	(No output on stdout) stderr: execvp(/usr/local/nagios/libexec/check_ssh, ...) failed. errno is 2: No such file or directory
03-13-2020 00:37:47	Service Alert	localhost	Swap Usage	CRITICAL	SOFT	(No output on stdout) stderr: execvp(/usr/local/nagios/libexec/check_swap, ...) failed. errno is 2: No such file or directory
03-13-2020 00:37:31	Service Alert	localhost	Root Partition	CRITICAL	HARD	(No output on stdout) stderr: execvp(/usr/local/nagios/libexec/check_disk, ...) failed. errno is 2: No such file or directory
03-13-2020 00:37:24	Service Alert	localhost	Total Processes	CRITICAL	SOFT	(No output on stdout) stderr: execvp(/usr/local/nagios/libexec/check_procs, ...) failed. errno is 2: No such file or directory
03-13-2020 00:37:09	Service Alert	localhost	SSH	CRITICAL	SOFT	(No output on stdout) stderr: execvp(/usr/local/nagios/libexec/check_ssh, ...) failed. errno is 2: No such file or directory
03-13-2020 00:36:54	Service Alert	localhost	PING	CRITICAL	HARD	(No output on stdout) stderr: execvp(/usr/local/nagios/libexec/check_ping, ...) failed. errno is 2: No such file or directory
03-13-2020 00:36:53	Service Alert	localhost	HTTP	CRITICAL	HARD	(No output on stdout) stderr: execvp(/usr/local/nagios/libexec/check_http, ...) failed. errno is 2: No such file or directory
03-13-2020 00:36:47	Service Alert	localhost	Swap Usage	CRITICAL	SOFT	(No output on stdout) stderr: execvp(/usr/local/nagios/libexec/check_swap, ...) failed. errno is 2: No such file or directory
03-13-2020 00:36:31	Service Alert	localhost	Root Partition	CRITICAL	SOFT	(No output on stdout) stderr: execvp(/usr/local/nagios/libexec/check_disk, ...) failed. errno is 2: No such file or directory
03-13-2020 00:36:24	Service Alert	localhost	Total Processes	CRITICAL	SOFT	(No output on stdout) stderr: execvp(/usr/local/nagios/libexec/check_procs, ...) failed. errno is 2: No such file or directory
03-13-2020 00:36:09	Service Alert	localhost	SSH	CRITICAL	SOFT	(No output on stdout) stderr: execvp(/usr/local/nagios/libexec/check_ssh, ...) failed. errno is 2: No such file or directory
03-13-2020 00:35:54	Service Alert	localhost	PING	CRITICAL	SOFT	(No output on stdout) stderr: execvp(/usr/local/nagios/libexec/check_ping, ...) failed. errno is 2: No such file or directory
03-13-2020 00:35:53	Service Alert	localhost	HTTP	CRITICAL	SOFT	(No output on stdout) stderr: execvp(/usr/local/nagios/libexec/check_http, ...) failed. errno is 2: No such file or directory
03-13-2020 00:35:47	Service Alert	localhost	Swap Usage	CRITICAL	SOFT	(No output on stdout) stderr: execvp(/usr/local/nagios/libexec/check_swap, ...) failed. errno is 2: No such file or directory

CONCLUSION: In this assignment we studied about NagiOs and how to study its different features.

Assignment 9 Roll No 104 Batch T22

Aim: To use Nagios XI Demo and understand features it offers.

Lab Outcomes: LO1 and LO5.

Theory:

Nagios is an open-source monitoring and alerting system that helps organizations track the status and performance of their IT infrastructure, including servers, network devices, applications, and services. It is widely used in IT environments to ensure the availability and reliability of critical systems and services.

Features and components of Nagios:

Host and Service Monitoring: Nagios can monitor the availability and health of network hosts (e.g., servers, switches, routers) and services (e.g., web servers, email servers) by periodically sending check requests to them.

Flexible Plugin Architecture: Nagios uses plugins to perform various checks and collect data. Users can create custom plugins or use existing ones to monitor specific aspects of their infrastructure.

Alerting and Notification: When a problem is detected, Nagios can send alerts and notifications to administrators via email, SMS, or other methods. It supports escalation mechanisms to ensure that critical issues are addressed promptly.

Historical Data Retention: Nagios can store historical performance data, allowing users to analyze trends and generate reports over time. This is useful for capacity planning and troubleshooting.

Web Interface: Nagios provides a web-based dashboard for configuring and managing monitoring settings, as well as viewing the status of hosts and services in real-time.

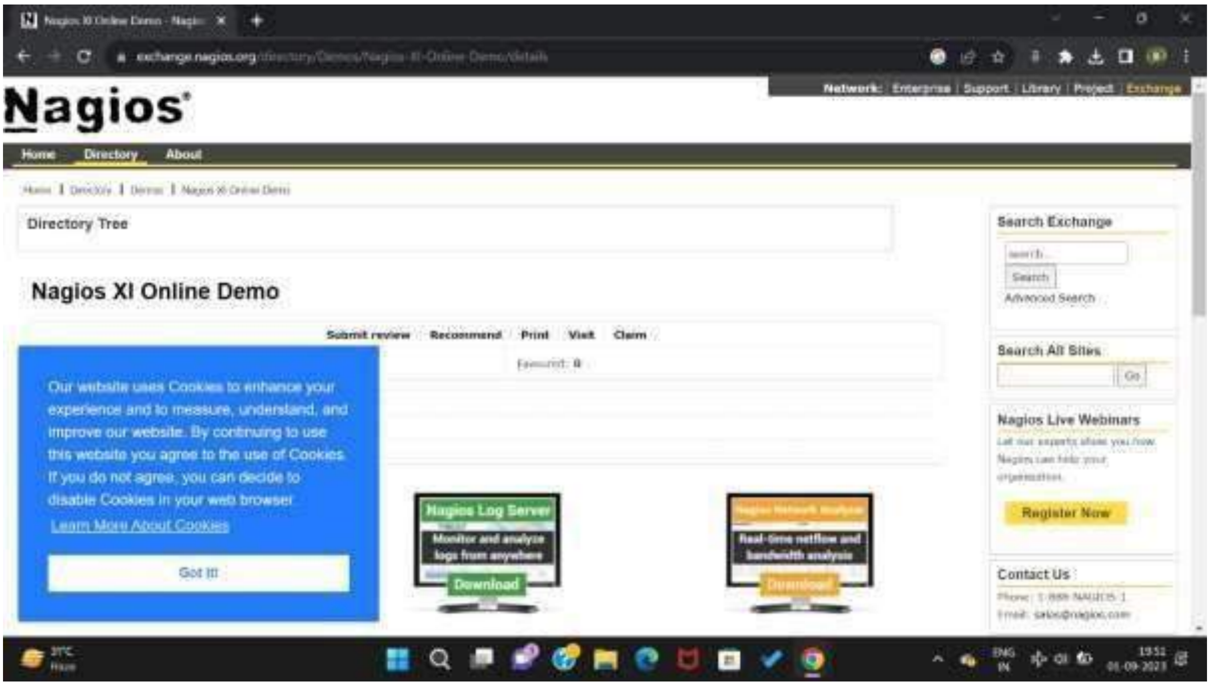
Security: Access control mechanisms are in place to restrict who can make changes to the Nagios configuration and view monitoring data.

Community and Add-ons: Nagios has a vibrant community that has developed numerous add-ons, extensions, and integrations to enhance its functionality. Some popular Nagios forks and extensions include Icinga, Naemon, and Nagios XI (a commercial version with additional features and support).

Scalability: Nagios can be scaled to monitor large and complex infrastructures by distributing monitoring tasks across multiple servers and using load balancing.

Active and Passive Checks: Nagios supports both active checks (where it actively queries hosts and services) and passive checks (where hosts and services submit status information to Nagios).

Event Handling: Nagios can be configured to automatically respond to certain events or issues by executing predefined scripts or actions



exchange.nagios.org/directory/Demos/Nagios-XI-Online-Demo/details

Network: Enterprise | Support | Library | Project | Exchange

Home | Directory | Demos | Nagios XI Online Demo

Directory Tree

Nagios XI Online Demo

Submit review Recommend Print Visit Claim

Favoured: 0

Our website uses Cookies to enhance your experience and to measure, understand, and improve our website. By continuing to use this website you agree to the use of Cookies. If you do not agree, you can decide to disable Cookies in your web browser.

Learn More About Cookies

Got It!

Search Exchange

Advanced Search

Search All Sites

Nagios Live Webinars

Let our experts show you how Nagios can help your organization.

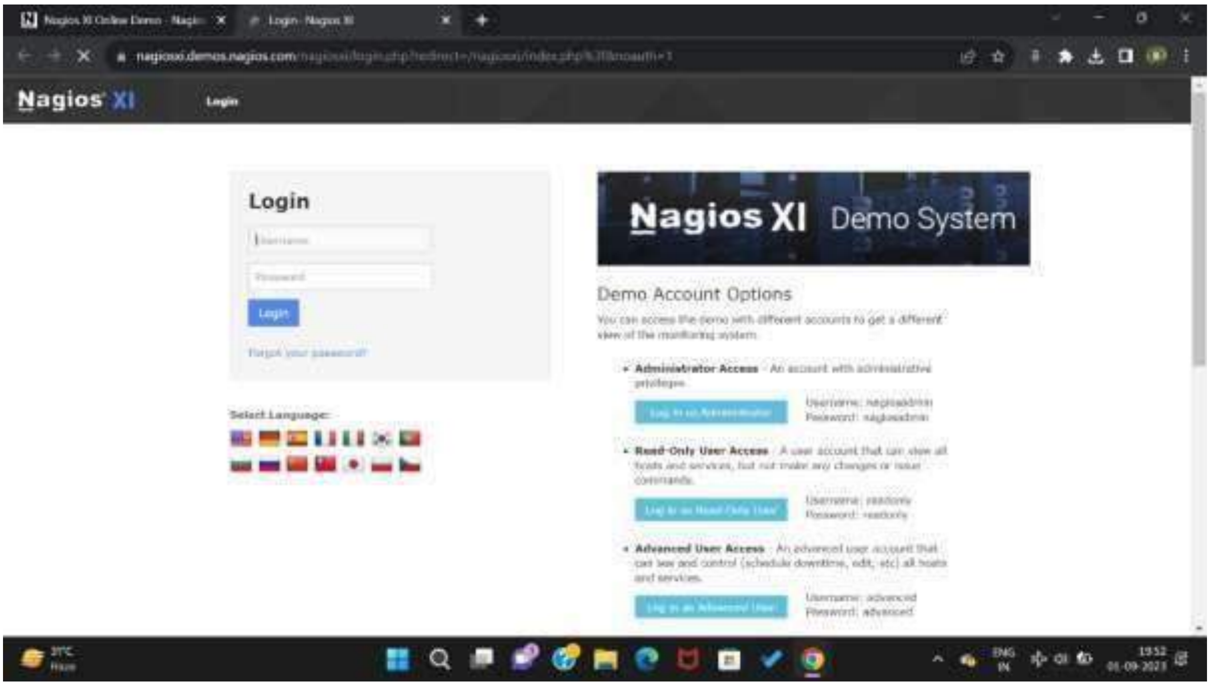
Register Now

Contact Us

Phone: 1-888-NAGIOS-1

Email: sales@nagios.com

Nagios Log Server
Monitor and analyze logs from anywhere
Download
Real-time netflow and bandwidth analysis
Download



nagiosxi.demos.nagios.com/nagiosxi/login.php?redirect=/nagiosxi/index.php%3f&noauth=1

Login

Login

Forgot your password?

Select Language:

Nagios XI Demo System

Demo Account Options

You can access the demo with different accounts to get a different view of the monitoring system.

- **Administrator Access** - An account with administrative privileges.

  Log In as Administrator

  Username: nagiosadmin
  Password: nagiosadmin

- **Read-Only User Access** - A user account that can view all hosts and services, but not make any changes or issue commands.

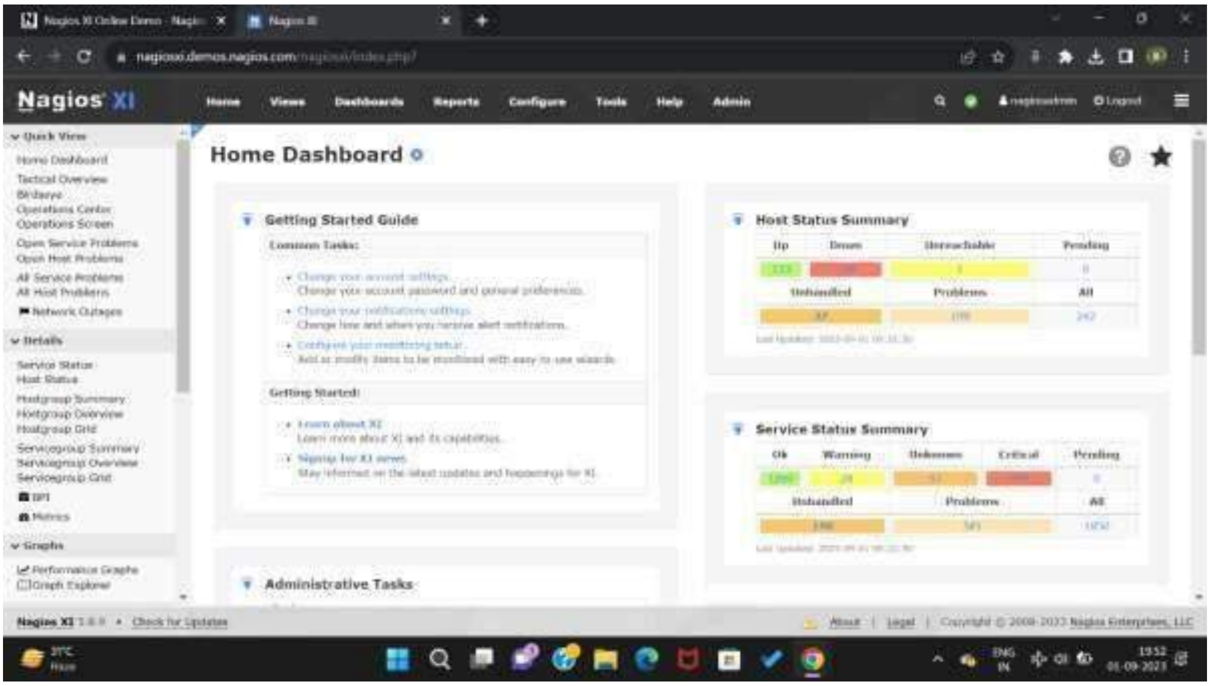
  Log In as Read-Only User

  Username: readonly
  Password: readonly

- **Advanced User Access** - An advanced user account that can see and control (schedule downtime, edit, etc) all hosts and services.

  Log In as Advanced User

  Username: advanced
  Password: advanced



Home Dashboard

Getting Started Guide

Common Tasks:

- Change your account settings
  Change your account password and general preferences.
- Change your notification settings
  Change how and when you receive alert notifications.
- Configure your monitoring setup
  Add or modify items to be monitored with easy to use wizards.

Getting Started:

- Learn about XI
  Learn more about XI and its capabilities.
- Signup for XI news
  Stay informed on the latest updates and happenings for XI.

Administrative Tasks

Host Status Summary

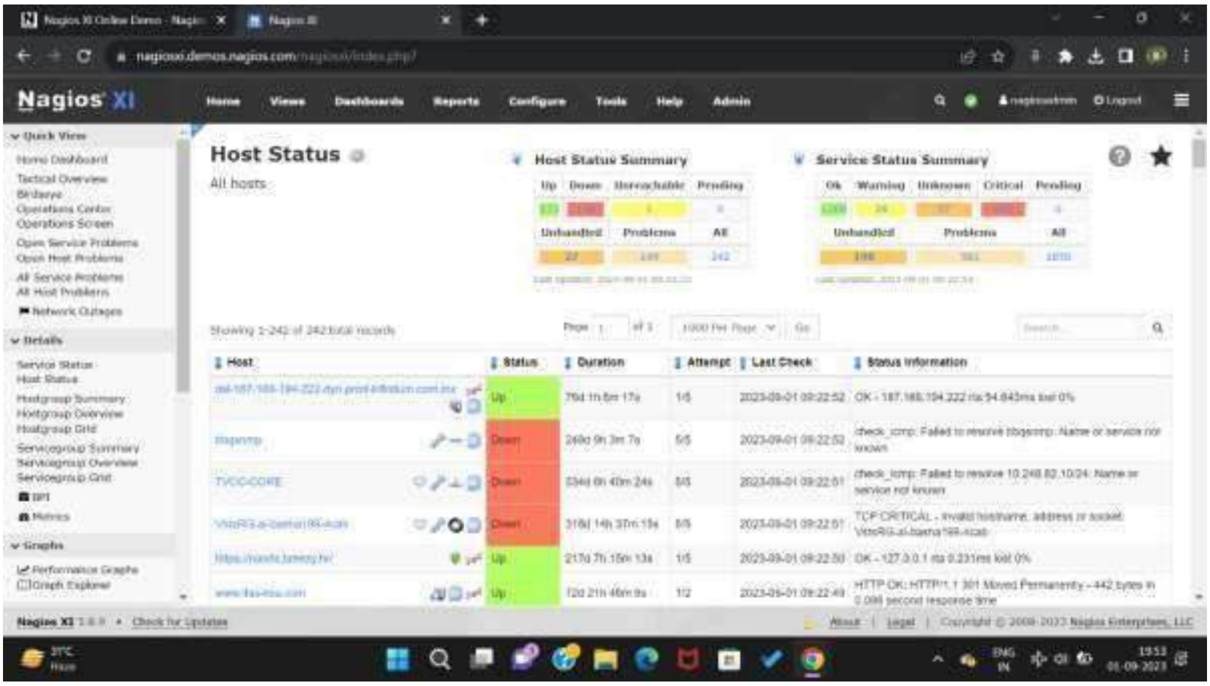
Up	Down	Unreachable	Pending
133	[illegible]	1	0
Unhandled	Problems	All	
37	109	242	

Last Updated: 2023-09-01 09:22:30

Service Status Summary

Ok	Warning	Unknown	Critical	Pending
1269	24	57	[illegible]	0
Unhandled	Problems	All		
198	581	1850		

Last Updated: 2023-09-01 09:22:30



Host Status

All hosts

Host Status Summary

Up	Down	Unreachable	Pending
133	[illegible]	1	0
Unhandled	Problems	All	
37	109	242	

Last Updated: 2023-09-01 09:22:33

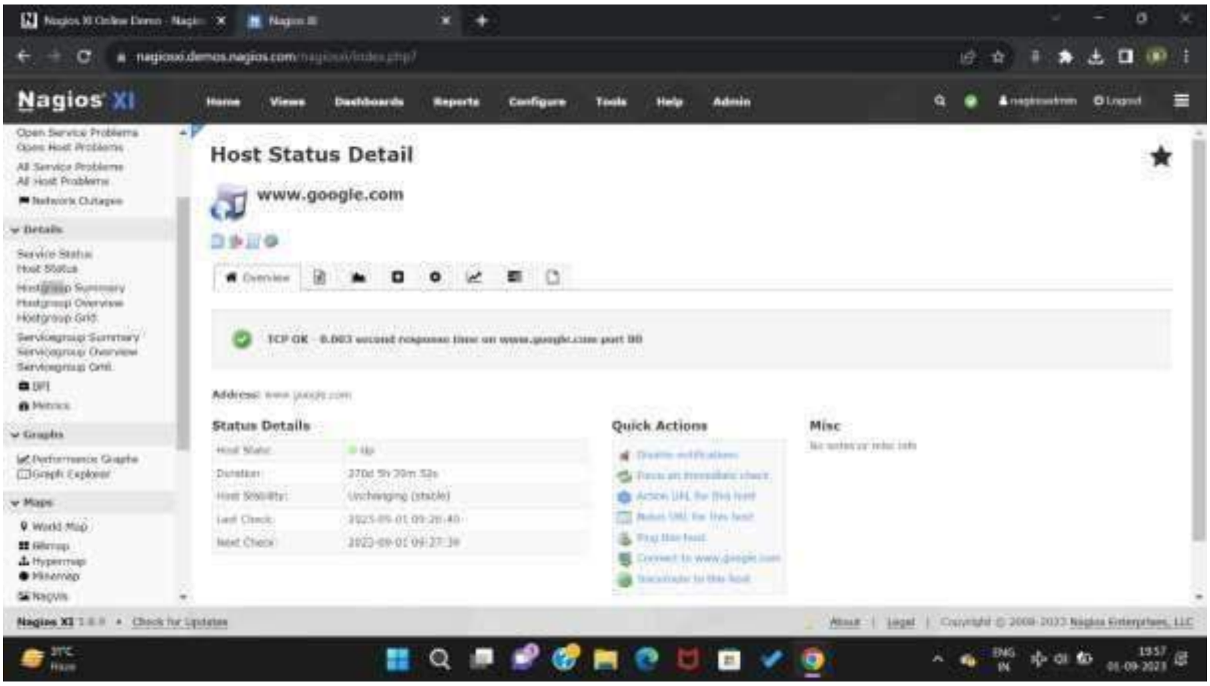
Service Status Summary

Ok	Warning	Unknown	Critical	Pending
1269	24	57	[illegible]	0
Unhandled	Problems	All		
198	581	1850		

Last Updated: 2023-09-01 09:22:34

Showing 1-242 of 242 total records

Host	Status	Duration	Attempt	Last Check	Status Information
dsl-187-188-194-222-dyn.prod-infinitum.com.mx	Up	79d 1h 8m 17s	1/5	2023-09-01 09:22:52	OK - 187.188.194.222 rta 54.843ms lost 0%
bbgsnmp	Down	248d 9h 3m 7s	5/5	2023-09-01 09:22:52	check_icmp: Failed to resolve bbgsnmp: Name or service not known
TVCC-CORE	Down	534d 6h 40m 24s	5/5	2023-09-01 09:22:51	check_icmp: Failed to resolve 10.248.82.10/24: Name or service not known
VitoRG-al-baena199-Acab	Down	318d 14h 37m 15s	5/5	2023-09-01 09:22:51	TCP CRITICAL - Invalid hostname, address or socket: VitoRG-al-baena199-Acab
https://nassfe.fareedy.hr/	Up	217d 7h 15m 13s	1/5	2023-09-01 09:22:50	OK - 127.0.0.1 rta 0.231ms lost 0%
www.ilas-mss.com	Up	120 21h 46m 9s	1/2	2023-09-01 09:22:49	HTTP OK: HTTP/1.1 301 Moved Permanently - 442 bytes in 0.098 second response time



Host Status Detail

www.google.com

TCP OK - 0.003 second response time on www.google.com port 80

Address: www.google.com

Status Details

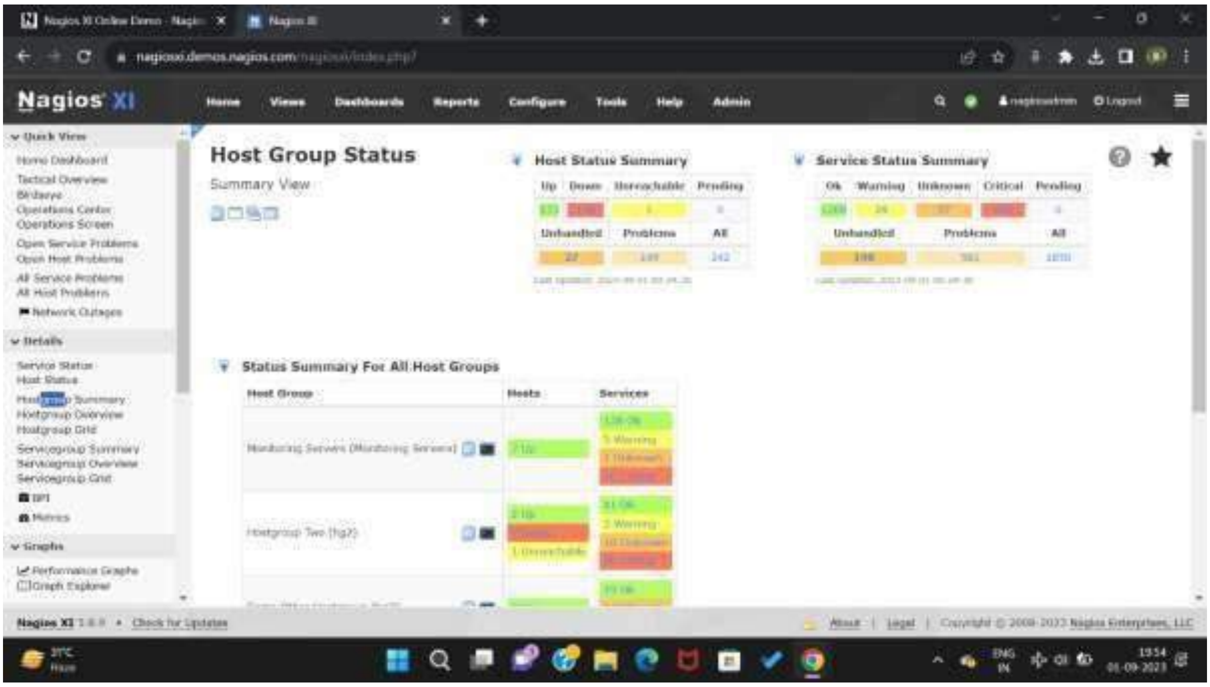
Host State:	Up
Duration:	270d 5h 20m 52s
Host Stability:	Unchanging (stable)
Last Check:	2023-09-01 09:20:40
Next Check:	2023-09-01 09:27:39

Quick Actions

- Disable notifications
- Force an immediate check
- Action URL for this host
- Notes URL for this host
- Ping this host
- Connect to www.google.com
- Traceroute to this host

Misc

No notes or misc info



Host Group Status

Summary View

Host Status Summary

Up	Down	Unreachable	Pending
[illegible]	[illegible]	1	0

Unhandled	Problems	All
27	[illegible]	242

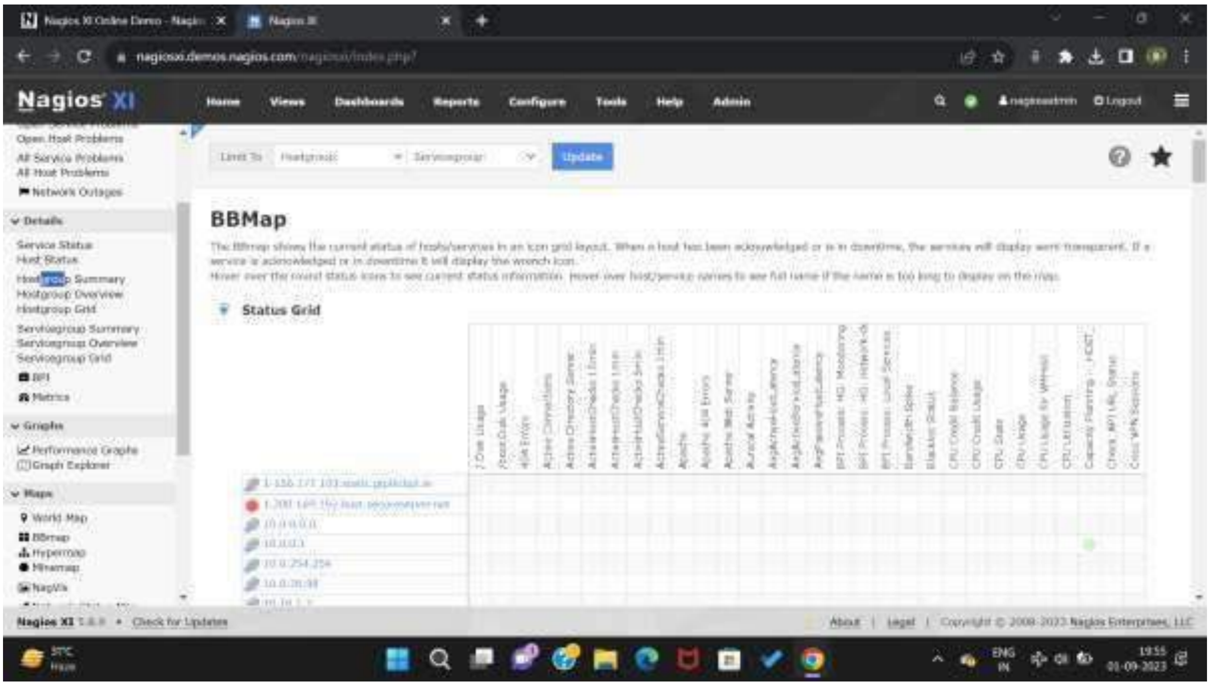
Service Status Summary

Ok	Warning	Unknown	Critical	Pending
[illegible]	24	[illegible]	[illegible]	0

Unhandled	Problems	All
198	[illegible]	[illegible]

Status Summary For All Host Groups

Host Group	Hosts	Services
Monitoring Servers (Monitoring Servers)	7 Up	126 Ok 5 Warning 1 Unknown [illegible]
Hostgroup Two (hg2)	2 Up [illegible] 1 Unreachable	31 Ok 2 Warning 10 Unknown [illegible]



Nagios XI Online Demo - Nagios | Nagios XI

nagiosxi.demos.nagios.com/nagiosxi/index.php?

Nagios XI — Home · Views · Dashboards · Reports · Configure · Tools · Help · Admin — nagiosadmin · Logout

Open Host Problems
All Service Problems
All Host Problems
Network Outages

Details
Service Status
Host Status
Hostgroup Summary
Hostgroup Overview
Hostgroup Grid
Servicegroup Summary
Servicegroup Overview
Servicegroup Grid
BPI
Metrics

Graphs
Performance Graphs
Graph Explorer

Maps
World Map
BBmap
Hypermap
Minemap
NagVis

Limit To: Hostgroup · Servicegroup · Update

BBMap

The BBmap shows the current status of hosts/services in an icon grid layout. When a host has been acknowledged or is in downtime, the services will display semi transparent. If a service is acknowledged or in downtime it will display the wrench icon.
Hover over the round status icons to see current status information. Hover over host/service names to see full name if the name is too long to display on the map.

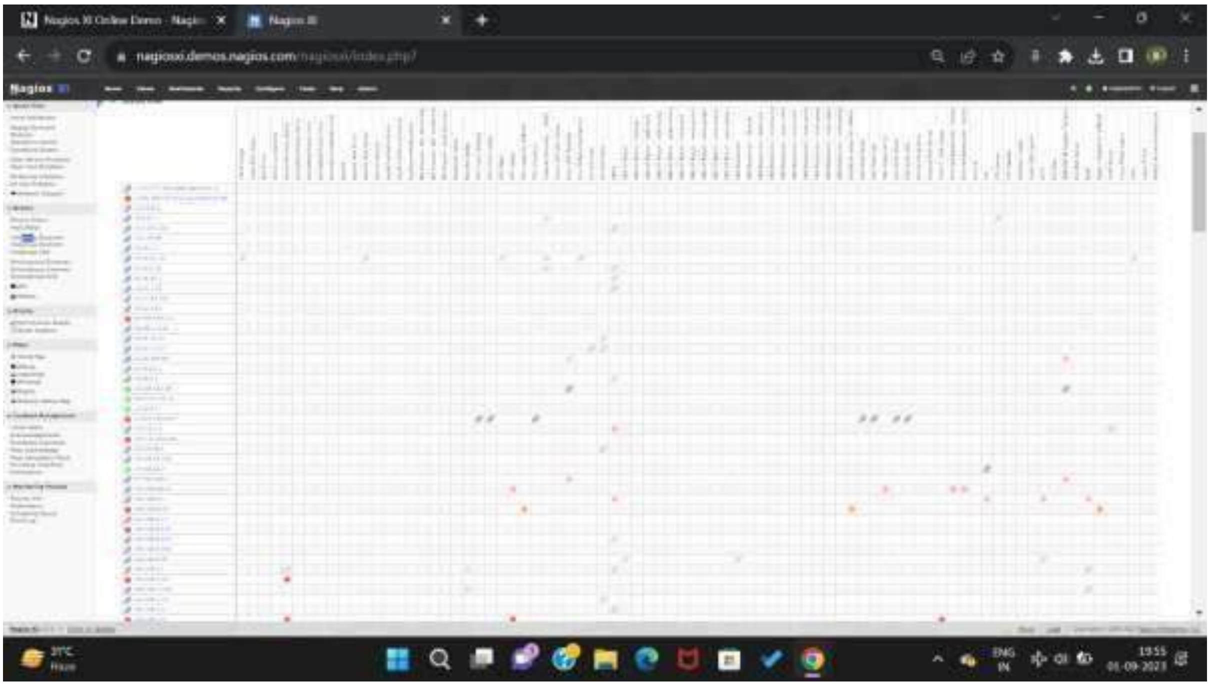
Status Grid

/ Disk Usage · /boot Disk Usage · 404 Errors · Active Connections · Active Directory Server · ActiveHostChecks 1 Error · ActiveHostChecks 1min · ActiveHostChecks 5min · ActiveServiceChecks 1min · Apache · Apache 404 Errors · Apache Web Server · Aurora Activity · AvgActiveHostLatency · AvgActiveServiceLatency · AvgPassiveHostLatency · BPI Process: HG: Monitoring · BPI Process: HG: Network-G · BPI Process: Local Services · Bandwidth Spike · Blacklist Status · CPU Credit Balance · CPU Credit Usage · CPU Stats · CPU Usage · CPU Usage for VMHost · CPU Utilization · Capacity Planning - HOST_ · Check: MPI URL Status · Cisco VPN Sessions

1-156-171-103.static.giplelink.in
1.200.149.150.host.secureserver.net
10.0.0.0.0
10.0.0.1
10.0.254.254
10.0.26.98
10.10.1.1

Nagios XI 5.8.8 • Check for Updates — About | Legal | Copyright © 2008-2023 Nagios Enterprises, LLC

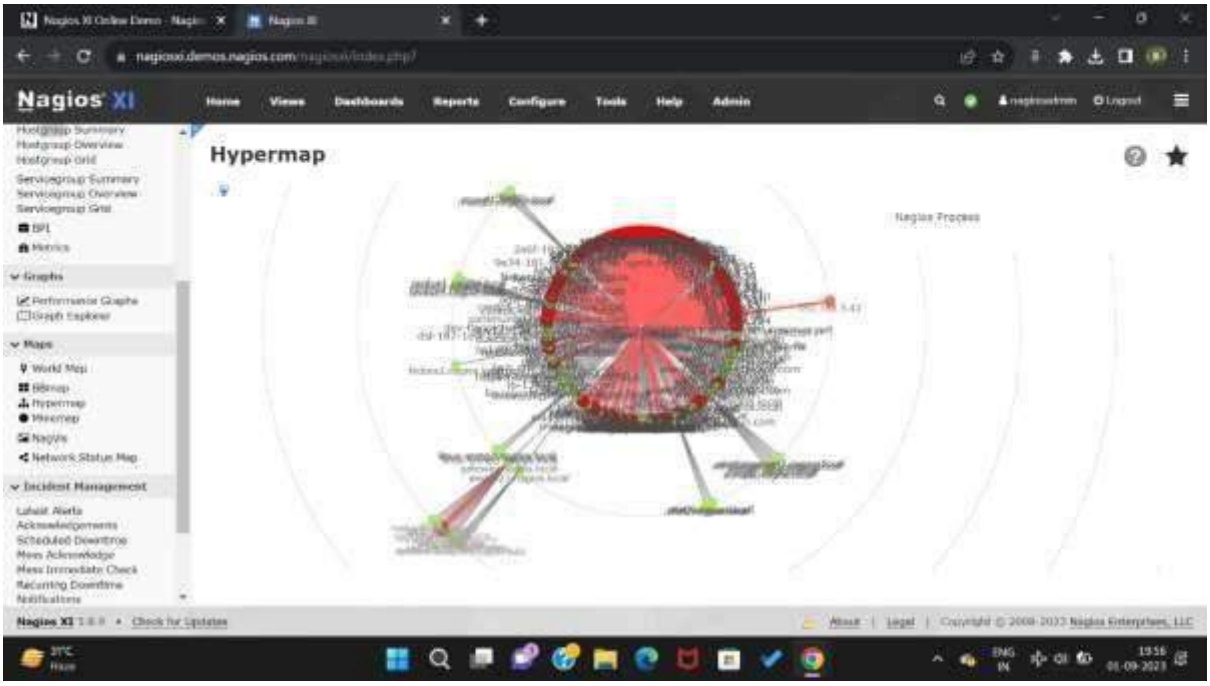
31°C Haze — ENG IN — 19:55 01-09-2023



Nagios XI Online Demo - Nagios | Nagios XI

nagiosxi.demos.nagios.com/nagiosxi/index.php?

31°C Haze — ENG IN — 19:55 01-09-2023



Conclusion:

LO1 and **LO5** were achieved.

Used Nagios XI demo.

Nagios is an open-source monitoring and alerting system widely used in IT environments to track the availability and performance of network hosts, services, and applications. Its flexible plugin architecture, alerting capabilities, historical data retention, and scalability make it a valuable tool for maintaining the reliability and health of critical infrastructure components.

Assignment 10

Roll No: 104   Batch T22

Aim : To study puppet tools

LO Mapped : LO1 , LO6

Theory :

Q) What is Puppet ?

Puppet is an efficient system management tool for centralizing and automating the configuration management process.

It can be used as a software deployment tool , It can also be utilized as open-source configuration management for server configuration, management, deployment, and orchestration. Puppet is specially designed to manage the configuration of Linux and Windows systems. It is written in Ruby and uses its unique Domain Specific Language (DSL) to describe system configuration.

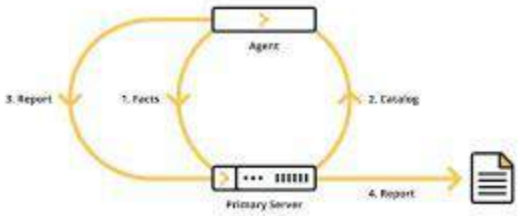
Configuration management is maintaining software and computer systems (servers, storage, networks) in a known, desired, and consistent state. It also allows access to an accurate historical record of the system state for project management and audit purposes.

There are two main versions of Puppet available in the market:

Open Source Puppet: This is a basic version of the Puppet configuration management tool. It is licensed under the Apache 2.0 system and can be downloaded from Puppet's website.

Puppet Enterprise: is a paid version that allows enterprises to manage nodes effectively with features like compliance reporting, orchestration, role-based access control, GUI, API, and command-line tools.

The diagram below shows how the server-agent architecture of a Puppet run works.



Q) What puppet can do?

Some of the key things that Puppet can do:

Configuration Management: Puppet allows you to define and manage the desired state of your infrastructure components (e.g., servers, applications, services) in a declarative manner. You specify how you want your systems to be configured, and Puppet ensures that they stay in that desired state.

Automation: Puppet automates the process of configuring and maintaining systems, reducing manual intervention and human error. It can handle routine tasks like software installation, configuration file management, and service provisioning.

Infrastructure as Code (IaC): Puppet enables the practice of treating infrastructure as code, meaning you define and manage your infrastructure using code files. This approach makes it easier to version, test, and collaborate on infrastructure changes.

Multi-Platform Support: Puppet supports various operating systems and can be used to manage both Unix/Linux and Windows environments. This makes it versatile for heterogeneous infrastructure.

Node Classification: Puppet allows you to classify nodes (servers or devices) based on their roles and responsibilities. You can define different configurations for different types of nodes, making it easy to scale and maintain large and diverse infrastructure.

Version Control: Puppet code can be stored in version control systems like Git, enabling collaboration, tracking changes, and ensuring a history of configuration changes.

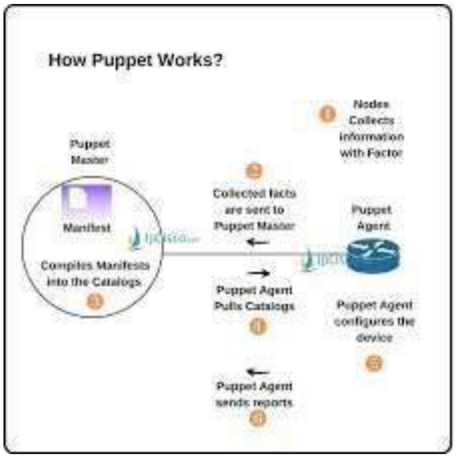
Reporting and Monitoring: Puppet provides reporting and monitoring capabilities to help you track the status of your infrastructure and configuration changes over time. You can identify issues, compliance violations, and performance bottlenecks.

Integration: Puppet can integrate with other DevOps tools and systems, such as continuous integration/continuous deployment (CI/CD) pipelines, monitoring tools, and orchestration platforms, to create a comprehensive automation and management ecosystem.

Q) How puppet works?

Puppet has a primary-secondary node architecture.

Puppet Architecture


How Puppet Works?
Nodes Collects information with Factor
Puppet Master
Manifest
Compiles Manifests into the Catalogs
Collected facts are sent to Puppet Master
Puppet Agent
Puppet Agent Pulls Catalogs
Puppet Agent configures the device
Puppet Agent sends reports

The clients are distributed across the network and communicate with the primary-secondary environment where Puppet modules are present. The client agent sends a certificate with its ID to the server; the server then signs that certificate and sends it back to the client. This authentication allows for secure and verifiable communication between the client and the master.

The factor then collects the state of the clients and sends it to the master. Based on the fact sent, the master compiles the manifests into the catalogs, which are sent to the clients, and an agent executes the manifests on its machine. A report is generated by the client that describes any changes made and is sent to the master.

This process is repeated at regular intervals, ensuring all client systems are up to date. In the next section, let us find out about the various companies adopting Puppet as a part of our learning about what is Puppet.

Q) Puppet Blocks- Puppet Resources, Puppet Classes, Puppet Manifest, Puppet Modules.

In Puppet, various concepts and constructs are used to manage and configure systems and infrastructure. The four terms mentioned—**Puppet Resources, Puppet Classes, Puppet Manifests, and Puppet Modules**—are fundamental components of Puppet's configuration management system.

Puppet Resources:

Puppet resources represent individual components or objects that you want to manage on your systems, such as files, packages, services, users, and groups.

Resources are defined in Puppet code using a specific syntax. For example, you can define a file resource to ensure the existence of a file or a package resource to ensure that a particular package is installed.

Puppet resources are the building blocks used to declare the desired state of your infrastructure.

Puppet Classes:

Puppet classes are a way to organize and group related resources and configurations together. Classes are defined in Puppet code and provide a modular and reusable way to organize your infrastructure configurations.
For example, you might create a "webserver" class that includes resources like a web server package, configuration files, and service management.

Puppet Manifests:
Puppet manifests are files that contain Puppet code. They are used to define the configuration and desired state of your systems.
Manifests typically have a .pp file extension and contain a collection of resource declarations, class includes, and other Puppet constructs.
Puppet manifests are the entry point for Puppet's configuration management, and they specify how resources and classes should be applied to nodes.

Puppet Modules:
Puppet modules are a way to package and distribute Puppet code and configurations in a structured and reusable manner.
A module is a directory structure that contains Puppet manifests, templates, files, and other assets needed to manage a specific piece of infrastructure or application.
Modules can be shared and reused across different Puppet environments and projects, making it easier to maintain and collaborate on Puppet code.
Common modules are often available on the Puppet Forge, a public repository of Puppet modules.

Q) What are benefits of Puppet tool?

Using a configuration management tool like Puppet in DevOps has many benefits.

Puppet's leveraging of Infrastructure as Code (IAC) features allow for efficient and flexible resolution of issues with version control, continuous delivery, peer review, and automated testing and deployment.

The other great benefit of using Puppet is that its **Master-Agent communication** using catalogs helps to identify the root causes of downtime and allows for a quick automated fix. This leads to overall downtime and faster Time To Recovery (TTR).

Puppet can also rely on the support of a **large open-source developer community** that assists developers in seeking solutions for various issues.

Puppet also **removes developers' need to adapt to a separate scripting language** to fulfill their tasks. It works seamlessly on platforms like Microsoft Windows, BSD, and Debian for Puppet.

Q) What are Puppet Manifest Files?

In Puppet, manifest files are used to define the desired configuration and state of your systems and infrastructure. Manifests are written in Puppet's domain-specific language (DSL), and they contain declarations of resources, classes, and other Puppet constructs. Manifests serve as the foundation for Puppet's configuration management process. Here are some key points about Puppet manifest files:

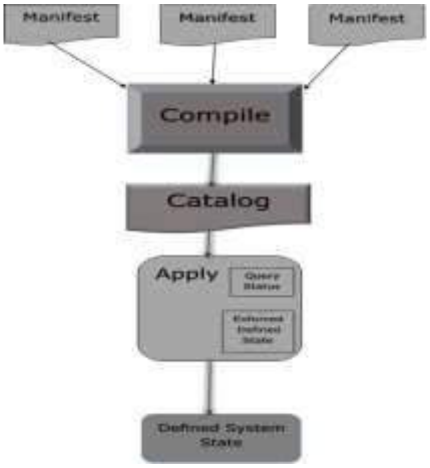
File Extension: Puppet manifest files typically have a .pp extension, which stands for Puppet Programming.

Resource Declarations: The primary purpose of a Puppet manifest is to declare the desired state of resources on a target node. Resources can represent various system components, such as files, packages, services, users, and groups. Each resource declaration specifies the resource type, a title (name or identifier for the resource), and attributes that define the desired state. For example, a file resource declaration might ensure that a specific file exists, has certain permissions, and is owned by a particular user.

Class Includes: Manifests can include or "include" Puppet classes. Classes are defined separately in modules (typically within a .pp file inside a module directory), and you can include them in your manifests to apply a predefined set of configurations to a node.
Variables and Facts: You can use variables and facts in Puppet manifests to make your configurations more dynamic and flexible. Facts provide information about the target node, and you can use them in your manifest logic.

Comments: Like any programming language, Puppet manifests can include comments to document your code and make it more understandable for others.

Modular Structure: Puppet manifests are often organized into modules for better code management and reuse. Modules encapsulate related Puppet code, including manifests, templates, and files, into a directory structure.



Q) Syntax of a Manifest File? why do we need Puppet Manifest Files? Writing a basic Manifest File with Examples.

A Puppet manifest file is written in Puppet's **domain-specific language (DSL)** and is used to declare the desired configuration and state of resources on target nodes. These manifest files are the heart of Puppet's configuration management system.

Syntax of a Puppet Manifest File:

A Puppet manifest file **consists of resource declarations, class includes, and other Puppet constructs.**
Resource Declaration: A resource declaration defines a specific resource (e.g., file, package, service) and its desired state. It follows this general syntax:

```
resource_type { 'resource_title':
attribute1 => value1,   attribute2
=> value2,
  # Additional attributes as needed
}
```

resource_type: Specifies the type of resource (e.g., file, package, service).
'resource_title': Provides a name or identifier for the resource.
attribute1, attribute2: Represents attributes specific to the resource type (e.g., ensure, owner, content).
Class Include: You can include Puppet classes within your manifest using the include keyword: include class_name **class_name:** The name of the Puppet class to include.
Conditional Statements: Puppet manifests can include conditional statements to apply configurations based on conditions: **if condition {**

```
  # Configuration for when the condition is true }
else {
  # Configuration for when the condition is false
}
```

Variables and Facts: You can use variables and facts to make your configurations dynamic:

```
$variable_name = value
```

$variable_name: Represents a variable that stores a value. value:
The value assigned to the variable.

Why Do We Need Puppet Manifest Files:

Declarative Configuration: Manifests allow you to declare the desired state of resources, making it clear what your infrastructure should look like.
Automation: Manifests automate the configuration of systems, reducing manual intervention and the potential for human error.
Consistency: Puppet ensures that configurations remain consistent across all nodes, even as they evolve or change.

Scalability: Puppet allows you to manage configurations for a large number of nodes simultaneously.
Version Control: Manifests can be version-controlled, enabling collaboration, change tracking, and rollback capabilities.

Example of a Basic Puppet Manifest File:

Simple example of a Puppet manifest that ensures the installation of the Nginx web server and starts the Nginx service:

```
# This manifest ensures that Nginx is installed and running.
class nginx {
package { 'nginx':
   ensure => 'installed',
  }

  service { 'nginx':    ensure  =>
'running',      enable   => true,
require => Package['nginx'],   }
}
```

In this example:
We define a class named "nginx" using the class keyword.
Inside the class, we declare a package resource to ensure that the Nginx package is installed.

We declare a service resource to ensure that the Nginx service is running and enabled.

The require attribute in the service resource specifies that it depends on the Nginx package being installed first.

When you apply this manifest to a node, Puppet will ensure that Nginx is installed and running according to the defined specifications.

Conclusion :

Learnt about basics of puppet tools in devops , their usage , benefits , basic syntax related to it and also implemented several examples to demonstrate above discussed things and explored about manifest files as well.

WRITTEN Assignment 1

Student Name: Soumil Salvi

Student Roll No: 104

Question 1: Test Automation

1.1 **Definition**:

Test automation, often referred to as automation testing, is the systematic use of software tools and scripts to automate the execution of test cases and the evaluation of test results. It aims to enhance the efficiency, effectiveness, and reliability of the testing process in software development. Automation testing involves creating test scripts that simulate user interactions with an application, allowing for repeatable and consistent testing procedures.

1.2 **Advantages of Automation Testing**:

Automation testing offers a plethora of benefits:

- **Repeatability**: Automated tests can be executed as many times as necessary without variations in test steps or data, which is ideal for regression testing.
- **Efficiency**: Automation significantly reduces the time and effort required for repetitive and labor-intensive testing tasks. It allows for faster feedback and quicker bug detection during the development process.
- **Coverage**: Automation can handle a broader range of test scenarios, including load testing, stress testing, and performance testing, ensuring comprehensive test coverage.
- **Consistency**: Automation eliminates the potential for human errors and ensures tests are executed in a uniform and precise manner.

- **Cost Savings**: While there are upfront costs associated with test automation, it ultimately reduces costs by minimizing the need for manual testing and accelerating time-to-market.
- **Parallel Testing**: Automation tools enable the simultaneous execution of tests across multiple environments or configurations, further expediting the testing process.
- **Continuous Integration**: Automation is integral to continuous integration and continuous delivery (CI/CD) pipelines, allowing for frequent and automated testing of code changes.

Question 2: XPath

2.1 **Definition**:

XPath is a language used for navigating and querying XML (eXtensible Markup Language) documents. It serves as a crucial component in web development and web testing by providing a standardized way to locate and select elements within XML documents, HTML pages, and other structured data formats. XPath expressions are used to pinpoint specific elements or nodes in a document, facilitating interactions, extraction, or validation of data.

2.2 **Single Slash vs. Double Slash**:

In XPath, the single slash ("/") and double slash ("//") have distinct functions:

- **Single Slash ("/")**: A single slash indicates an absolute path. It instructs XPath to begin the search from the root node of the document and navigate down the hierarchy to locate the desired element. For instance, the XPath `/html/body/div` denotes an absolute path to a `<div>` element within the `<body>` element of an HTML document.
- **Double Slash ("//")**: The double slash is used for specifying a relative path. It allows you to search for elements anywhere within the

document without starting from the root. For example, the XPath `//div` locates all `<div>` elements in the document, regardless of their position within the hierarchy.

The choice between a single and double slash depends on whether you require an absolute or relative path when searching for elements within a document. The double slash is particularly useful when the element's location may change within the document's structure, making it a versatile and powerful tool for web scraping and web testing.

Written Assignment 2

Roll No: 104 Batch T22

Q1. What is Selenium? What are the Selenium suite components?

Ans. :

Selenium is an open-source software suite for automating web browsers. It provides a framework for automating web application testing and interacts directly with a browser, simulating user interactions. Selenium is widely used for web testing and is known for its flexibility, extensibility, and support for multiple programming languages. The Selenium suite is composed of several components, each with a specific role:

1. Selenium WebDriver:

WebDriver is the core component of Selenium and provides a programming interface to interact with web browsers. It allows you to write test scripts in various programming languages to control web browsers and perform automated testing.

Key Features:

Supports multiple programming languages like Java, Python, C#, etc.

Provides a rich set of methods to interact with web elements (e.g., clicking buttons, filling forms, etc.).

Supports various web browsers, including Chrome, Firefox, Safari, Edge, and more.

Can run tests on different platforms (Windows, macOS, Linux) and in parallel.

Integrates with testing frameworks like JUnit and TestNG for test management.

2. Selenium IDE (Integrated Development Environment):

Selenium IDE is a browser extension that allows users to record, edit, and play back test scripts. It's primarily used for rapid prototyping and creating simple automation scripts.

Key Features:

No need for extensive programming knowledge; tests can be recorded through a point-and-click interface.

Offers a script editor for enhancing and customizing recorded scripts.

Supports exporting scripts to various programming languages for further refinement in WebDriver.

3. Selenium Grid:

Selenium Grid is used for parallel test execution on multiple machines or virtual environments. It facilitates distributed test execution, which is essential for large-scale testing.

Key Features:

Allows tests to be run in parallel on multiple browsers and platforms.

Provides centralized control and monitoring of test execution.

Optimizes test execution times, making it more efficient for test suites.

4. Selenium RC (Remote Control):
Selenium RC is an older component and is mostly deprecated in favor of WebDriver. It allows tests to interact with web browsers by acting as an HTTP proxy between the browser and the test script. While it is not recommended for new projects, some legacy systems still use it.
Key Features:
Supports multiple programming languages.
Provides a server that acts as a bridge between the test script and the browser.

5. Selenium Grid with Docker:
Selenium Grid with Docker is an extension of Selenium Grid that leverages Docker containers to create scalable and isolated test environments. It simplifies the setup and teardown of test execution environments.
Key Features:
Uses Docker containers to package browsers and WebDriver instances, allowing for efficient parallel test execution.
Easy management of test environments, including setup, scaling, and teardown. In-depth Explanation:

Selenium WebDriver is the primary component for interacting with web browsers. It provides a rich set of features to automate browser interactions and supports multiple programming languages.

Selenium IDE is a simple-to-use tool for recording and playing back test scripts. While it's great for quick scripting, it's often not suitable for complex testing scenarios.

Selenium Grid facilitates parallel test execution across multiple environments. It's essential for large-scale test suites and distributed testing.

Selenium RC is an older component and is generally considered outdated in favor of WebDriver. However, it may still be used in legacy projects.

Selenium Grid with Docker takes advantage of Docker containers to streamline the setup and management of test environments.

Selenium's flexibility, extensive browser support, and the ability to integrate with various testing frameworks make it a popular choice for web application testing and automation. The choice of Selenium component depends on your specific testing requirements and the scale of your project.

Q.2. What makes Selenium such a widely used testing tool? Give reasons. Why is it advised to select Selenium as a testing tool for web applications or systems?
Ans:

Selenium is widely used as a testing tool for web applications or systems for several compelling reasons:

1. Open Source: Selenium is an open-source tool, which means it's freely available to the testing community. This significantly reduces testing costs, making it an attractive option for organizations of all sizes.

2. Browser Compatibility: Selenium supports multiple web browsers, including Chrome, Firefox, Safari, Edge, and more. This cross-browser compatibility is crucial for ensuring your web application works consistently across different browsers.

3. Multi-Language Support: Selenium offers support for multiple programming languages, such as Java, Python, C#, Ruby, and others. This allows testing teams to choose a language they are familiar with and comfortable using.

4. Wide Adoption: Selenium has been widely adopted by the testing community, resulting in extensive documentation, tutorials, and a large user base. This means that there are plenty of resources and community support available for users.

5. Flexibility: Selenium provides flexibility in terms of test design and implementation. It allows for a wide range of testing scenarios, from simple UI interactions to complex test scripts and custom test frameworks.

6. Cross-Platform Testing: Selenium enables cross-platform testing, making it possible to test web applications on different operating systems, including Windows, macOS, and Linux.

7. Parallel Test Execution: Selenium Grid and related technologies allow for parallel test execution on multiple machines or browsers. This accelerates test execution and is vital for large-scale testing and continuous integration pipelines.

8. Integration with Testing Frameworks: Selenium seamlessly integrates with popular testing frameworks like JUnit and TestNG, which provides robust test management, reporting, and datadriven testing capabilities.

9. Extensibility: Selenium is highly extensible through the use of custom plugins and third-party libraries, allowing you to enhance and customize your test automation solutions.

10. Continuous Integration/Continuous Delivery (CI/CD) Support: Selenium can be easily integrated into CI/CD pipelines, providing automated testing as part of the software development and delivery process.

11. Record and Playback: Selenium IDE allows users to record and playback test scripts using a simple point-and-click interface, making it accessible to testers with varying technical backgrounds.

12. Regression Testing: Selenium is particularly valuable for regression testing, as it can quickly verify that new changes or features haven't introduced defects in existing functionality.

13. Large Community: The Selenium community is extensive, and this leads to the development of various plugins, tools, and extensions that enhance Selenium's capabilities.

14. Cross-Domain Testing: Selenium can test web applications across different domains and websites, making it suitable for complex test scenarios.

15. Automation of Repetitive Tasks: Selenium is highly effective at automating repetitive, timeconsuming manual testing tasks, reducing the potential for human errors and increasing efficiency.

For all these reasons, Selenium is often recommended as a testing tool for web applications or systems. It offers a powerful and versatile framework for automating web testing, supporting various web technologies, and providing a cost-effective and reliable solution for ensuring the quality of web applications