MGM's

Jawaharlal Nehru Engineering College Aurangabad

ISO 9001:2015, 140001:2015 Certified, AICTE Approved

Department of Computer Science & Engineering

LAB MANUAL

Programme(UG/PG)	: UG
Year	: Third Year
Semester	: VI
Course Code	: 20UCS610L
Course Title	: DevOps Tools & Techniques Lab

Prepared By
Mr. S. N. Jaiswal
Associate Professor
Department of Computer Science & Engineering

FOREWORD

It is my great pleasure to present this laboratory manual for third year engineering students for the subject of DevOps Tools & Techniques Lab.

As a student, many of you may be wondering with some of the questions in your mind regarding the subject and exactly what has been tried is to answer through this manual.

As you may be aware that MGM has already been awarded with ISO 9001:2015,140001:2015 certification and it is our endure to technically equip our students taking the advantage of the procedural aspects of ISO Certification.

Faculty members are also advised that covering these aspects in initial stage itself, will greatly relived them in future as much of the load will be taken care by the enthusiasm energies of the students once they are conceptually clear.

Dr. H. H. Shinde
Principal

LABORATORY MANUAL CONTENTS

This manual is intended for the Third year students of Computer Science & Engineering in the subject of Introduction to DevOps Tools & Techniques Lab. This manual typically contains practical/Lab Sessions related to Introduction to DevOps Tools & Techniques Lab covering various aspects related the subject to enhanced understanding.

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market Students are advised to thoroughly go through this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Good Luck for your Enjoyable Laboratory Sessions

Mr.S.N.Jaiswal
Subject Teacher

Dr. Deepa Deshpande
HOD

LIST OF EXPERIMENTS

Course Code: 20UCS610L

Course Title: DevOps Tools & Techniques Lab

S. No.	Name of the Experiment
1.	Introduction to different Webservers
2.	Study of Virtualization Software
3.	Version control : Git Installation
4.	Branching and Merging, Stashing, Rebasing, Reverting and Resetting
5.	Study and implementation of various git commands to push and pull a repository, from GitHub
6.	Creating simple Maven project and perform unit test and resolve dependencies
7.	Installing and Configuring Jenkins
8.	Creating a build using Jenkins
9.	Building Images using Docker File
10.	Creating multi-containers using Docker Compose
11.	Installation of Kubernete
12.	Study and use of Kubernetes services
13.	Installation and Configuration of Ansible
14	Continuous Monitoring using Nagios

DOs and DON'Ts in Laboratory:

1. Make entry in the Log Book as soon as you enter the Laboratory.
2. All the students should sit according to their roll numbers starting from their left to right.
3. All the students are supposed to enter the terminal number in the log book.
4. Do not change the terminal on which you are working.
5. All the students are expected to get at least the algorithm of the program/concept to be implemented.
6. Strictly observe the instructions given by the teacher/Lab Instructor.
7. Do not disturb machine Hardware / Software Setup.

Instruction for Laboratory Teachers:

1. Submission related to whatever lab work has been completed should be done during the next lab session along with signing the index.
2. The promptness of submission should be encouraged by way of marking and evaluation patterns that will benefit the sincere students.
3. Continuous assessment in the prescribed format must be followed.

Jawaharlal Nehru Engineering College, Aurangabad

Department of Computer Science and Engineering

Vision of CSE Department

To develop computer engineers with necessary analytical ability and human values who can creatively design, implement a wide spectrum of computer systems for welfare of the society.

Mission of the CSE Department:

~~1.~~ Preparing graduates to work on multidisciplinary platforms associated with their professional position both independently and in a team environment.

~~2.~~ Preparing graduates for higher education and research in computer science and engineering enabling them to develop systems for society development.

Programme Educational Objectives

Graduates will be able to

- **I.** To analyze, design and provide optimal solution for Computer Science & Engineering and multidisciplinary problems.
- **II.** To pursue higher studies and research by applying knowledge of mathematics and fundamentals of computer science.
- **III.** To exhibit professionalism, communication skills and adapt to current trends by engaging in lifelong learning.

Programme Outcomes (POs):

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems anddesign system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms ofthe engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage independent and life-long learning in the broadest context of technological change.

LABORATORY OUTCOMES

The practical/exercises in this section are psychomotor domain Learning Outcomes (i.e. subcomponents of the COs), to be developed and assessed to lead to the attainment of the competency. At the end of this course student will be able to -

LO-1: Install, configure and use Git on different platforms

LO-2: Create Continuous integration using Jenkins

LO-3: Install, configure and use Docker as container

LO-4: Install and use different services of Kuberenetes

LO-5: Write infrastructure as a code scripts using Ansible

LO-6: Install and use Nagios for continuous monitoring

1. Lab Exercise

Exercise No 1: (2 Hours) – 1 Practical

Aim: - Introduction to Different Web Servers (Prerequisite)

Objectives:

1. Student will able to install web servers
2. Students should be able to configure the web server.

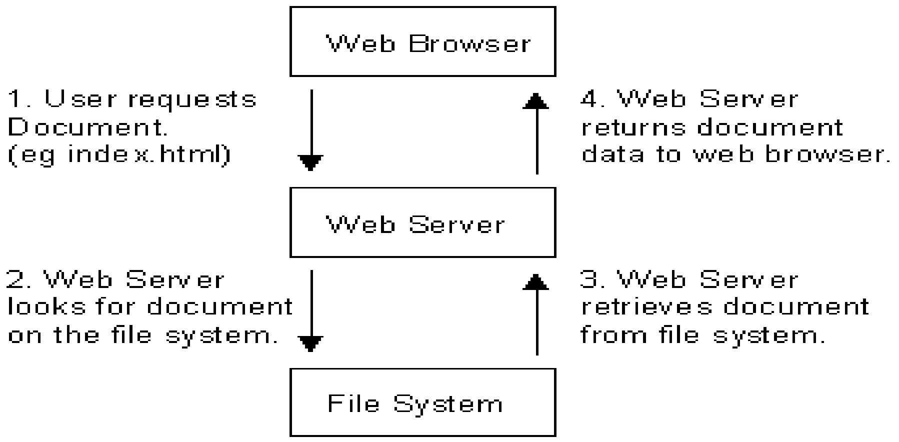
THEORY:

Introduction:

Web server is a program which processes the network requests of the users and serves them with files that create web pages. This exchange takes place using Hypertext Transfer Protocol (HTTP). Basically, web servers are computers used to store HTTP files which make a website and when a client requests a certain website, it delivers the requested website to the client. For example, you want to open Facebook on your laptop and enter the URL in the search bar of google. Now, the laptop will send an HTTP request to view the facebook webpage to another computer known as the web server. This computer (web server) contains all the files (usually in HTTP format) which make up the website like text, images, gif files, etc. After processing the request, the web server will send the requested website-related files to your computer and then you can reach the website.

Different websites can be stored on the same or different web servers but that doesn't affect the actual website that you are seeing in your computer. The web server can be any software or hardware but is usually a software running on a computer. One web server can handle multiple users at any given time which is a necessity otherwise there had to be a web server for each user and considering the current world population, is nearly close to impossible. A web server is never disconnected from the internet because if it was, then it won't be able to receive any requests, and therefore cannot process them.

Working of Web Server:


Web Browser
1. User requests Document. (eg index.html)
4. Web Server returns document data to web browser.
Web Server
2. Web Server looks for document on the file system.
3. Web Server retrieves document from file system.
File System

The step-by-step process of what happens whenever a web browser approaches the web server and requests a web file or file. Follow the below steps:

1. First, any web user is required to **type the URL of the web page in the address bar** of your web browser.
2. With the help of the URL, your **web browser will fetch the IP address of your domain** name either by converting the URL via DNS (Domain Name System) or by looking for the IP in cache memory. The IP address will direct your browser to the web server.
3. After making the connection, the **web browser will request for the web page from the web server** with the help of an HTTP request.
4. As soon as the web server receives this request, it immediately **responds by sending back the requested page** or file to the web browser HTTP.
5. If the web page requested by the **browser does not exist or if there occurs some error in the process**, the web server will return an error message.
6. If there occurs no error, the browser will successfully display the webpage.

. **Common and top web server software in the market:**

There are a number of common web servers available, some including:

- **Apache HTTP Server.** Developed by Apache Software Foundation, it is a free and open source web server for Windows, Mac OS X, Unix, Linux, Solaris and other operating systems; it needs the Apache license.
- **Microsoft Internet Information Services (IIS).** Developed by Microsoft for Microsoft platforms; it is not open sourced, but widely used.
- **Nginx.** A popular open source web server for administrators because of its light resource utilization and scalability. It can handle many concurrent sessions due to its event-driven architecture. Nginx also can be used as a proxy server and load balancer.
- **Lighttpd.** A free web server that comes with the FreeBSD operating system. It is seen as fast and secure, while consuming less CPU power.
- **Sun Java System Web Server.** A free web server from Sun Microsystems that can run on Windows, Linux and Unix. It is well-equipped to handle medium to large websites.

Web Server Configuration:

The way a web server is configured greatly influences its reliability and security. To ensure the best performance, server administrators carry out several tasks:

- **Settings configuration**. Administrators adjust parameters like cache size, request limits, and connection timeouts to align with the hosted website's specific requirements. By fine-tuning these settings, the server can efficiently manage incoming requests and quickly deliver content.
- **Security measures**. To safeguard sensitive data and mitigate potential threats, administrators implement robust security measures. They include setting up firewalls, detection systems, and encryption protocols. Regular security patches are also crucial to keep hackers at bay.
- **Performance optimization**. Server configuration lets administrators optimize resource allocation, implement effective load balancing, and utilize caching mechanisms. These

measures enhance the server's ability to handle traffic spikes, resulting in reduced **latency** and faster response times.

Install Web server on Windows:
System Requirements

Before downloading and installing the following requirements are essential.

- Operating System Version - Microsoft Windows (32-bit or 64-bit).
- Random Access Memory (RAM) - Minimum 4 GB RAM recommended.
- Free Disk Space - Minimum 25 GB free space recommended.
- Good Internet Connection

You can download Apache with OpenSSL from https://www.apachehaus.com/cgi-bin/download.plx.
To install, copy the Apache24 folder to the root of your C drive, then run *C:\Apache24\bin\httpd -k install* at the command prompt.
Once installed, you can start the Apache service and edit the httpd.conf file for your website

Again, click on the "Next" button.

Select the amount of space for your virtual machine and click the "Create" button. (This will be used for your operating system which is going to be installed, so give as much space as possible).

Outcome:

To learn the installation process of different web servers and configure them.

CONCLUSIONS:
Students will be able to install a web server on the laptop or desktop. Students understand the directory structure of the web server and configure the system.

Task to conduct in Laboratory :

- **Install Apache Tomcat & IIS server**
- **Configure the web server to run on specified port**
- **Change the password of user**
- **Configure the database**
- **Run the server from any machine**
- **Run Host Application from any machine and upload war from any machine**
- **Configure the server for clustering**

Journal Write-up

- Introduction
- What is web server?
- History of Web Server

- HTTP Protocol
- Working of Web Server
- Steps to install Apache Tomcat
- Directory Structure of Apache Tomcat
- Configuration of Apache Tomcat
- Conclusion

2. Lab Exercise

Exercise No 2: (2 Hours) – 1 Practical

Aim: - Study of Virtualization Software

Objectives:

1. Learn the need of virtualization software.
2. Study the different virtualization software in the market
3. Compare virtualization with containerization

THEORY:
Virtualization is technology that you can use to create virtual representations of servers, storage, networks, and other physical machines. Virtual software mimics the functions of physical hardware to run multiple virtual machines simultaneously on a single physical machine. Businesses use virtualization to use their hardware resources efficiently and get greater returns from their investment. It also powers cloud computing services that help organizations manage infrastructure more efficiently.

Virtual machines and hypervisors are two important concepts in virtualization.

Virtual machine:- A *virtual machine* is a software-defined computer that runs on a physical computer with a separate operating system and computing resources. The physical computer is called the *host machine* and virtual machines are *guest machines*. Multiple virtual machines can run on a single physical machine. Virtual machines are abstracted from the computer hardware by a hypervisor.

Hypervisor:- The *hypervisor* is a software component that manages multiple virtual machines in a computer. It ensures that each virtual machine gets the allocated resources and does not interfere with the operation of other virtual machines. There are two types of hypervisors.
What are the benefits of virtualization?
Virtualization provides several benefits to any organization:
Efficient resource use:- Virtualization improves hardware resources used in your data center. For example, instead of running one server on one computer system, you can create a virtual

server pool on the same computer system by using and returning servers to the pool as required. Having fewer underlying physical servers frees up space in your data center and saves money on electricity, generators, and cooling appliances.

Automated IT management:- Now that physical computers are virtual, you can manage them by using software tools. Administrators create deployment and configuration programs to define virtual machine templates. You can duplicate your infrastructure repeatedly and consistently and avoid error-prone manual configurations.

Faster disaster recovery:- When events such as natural disasters or cyberattacks negatively affect business operations, regaining access to IT infrastructure and replacing or fixing a physical server can take hours or even days. By contrast, the process takes minutes with virtualized environments. This prompt response significantly improves resiliency and facilitates business continuity so that operations can continue as scheduled.

What are the different types of virtualization?

You can use virtualization technology to get the functions of many different types of physical infrastructure and all the benefits of a virtualized environment. You can go beyond virtual machines to create a collection of virtual resources in your virtual environment.

Server virtualization

Server virtualization is a process that partitions a physical server into multiple virtual servers. It is an efficient and cost-effective way to use server resources and deploy IT services in an organization. Without server virtualization, physical servers use only a small amount of their processing capacities, which leave devices idle.

Storage virtualization

Storage virtualization combines the functions of physical storage devices such as network attached storage (NAS) and storage area network (SAN). You can pool the storage hardware in your data center, even if it is from different vendors or of different types. Storage virtualization uses all your physical data storage and creates a large unit of virtual storage that you can assign and control by using management software. IT administrators can streamline storage activities, such as archiving, backup, and recovery, because they can combine multiple network storage devices virtually into a single storage device.

Network virtualization

Any computer network has hardware elements such as switches, routers, and firewalls. An organization with offices in multiple geographic locations can have several different network technologies working together to create its enterprise network. Network virtualization is a process that combines all of these network resources to centralize administrative tasks. Administrators can adjust and control these elements virtually without touching the physical components, which greatly simplifies network management.

CONCLUSIONS:

Students will learn about the virtualization and tools available for virtualization

3. Lab Exercise

Exercise No 1: (2 Hours) – 1 Practical

Aim: - Version Control: Git Installation

Objectives:

1. Students should able to understand the version control system
2. Students should able to understand the Git for VCS and install it

Pre-requisites: Basic knowledge of Windows & Linux commands

THEORY:

What is Git?

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

It is designed to handle minor to major projects with high speed and efficiency. It is developed to co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

Git is a version control system. Git helps you keep track of code changes. Git is used to collaborate on code.

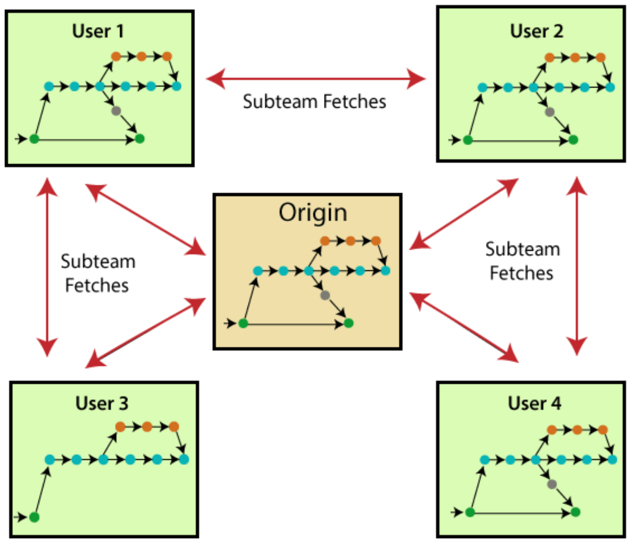
Git is foundation of many services like GitHub and GitLab, but we can use Git without using any other Git services. Git can be used privately and publicly.

Git was created by Linus Torvalds in 2005 to develop Linux Kernel. It is also used as an important distributed version-control tool for the DevOps. Git is easy to learn, and has fast performance. It is superior to other SCM tools like Subversion, CVS, Perforce, and ClearCase.

Features of Git

- **Open Source**
  Git is an **open-source tool**. It is released under the **GPL** (General Public License) license.
- **Scalable**
  Git is **scalable**, which means when the number of users increases, the Git can easily handle such situations.
- **Distributed**
  One of Git's great features is that it is **distributed**. Distributed means that instead of switching the project to another machine, we can create a "clone" of the entire repository. Also, instead of just having one central repository that you send changes to, every user has their own repository that contains the entire commit history of the project. We do not need to

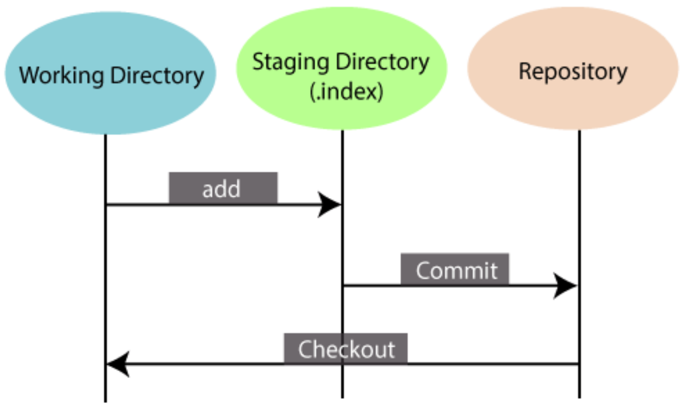
connect to the remote repository; the change is just stored on our local repository. If necessary, we can push these changes to a remote repository.



- **Security**
  Git is secure. It uses the **SHA1 (Secure Hash Function)** to name and identify objects within its repository. Files and commits are checked and retrieved by its checksum at the time of checkout. It stores its history in such a way that the ID of particular commits depends upon the complete development history leading up to that commit. Once it is published, one cannot make changes to its old version.
- **Speed**
  Git is very **fast**, so it can complete all the tasks in a while. Most of the git operations are done on the local repository, so it provides a **huge speed**. Also, a centralized version control system continually communicates with a server somewhere.
  Performance tests conducted by Mozilla showed that it was **extremely fast compared to other VCSs**. Fetching version history from a locally stored repository is much faster than fetching it from the remote server. The **core part of Git** is **written in C**, which **ignores** runtime overheads associated with other high-level languages.
  Git was developed to work on the Linux kernel; therefore, it is **capable** enough to **handle large repositories** effectively. From the beginning, **speed** and **performance** have been Git's primary goals.
- **Supports non-linear development**
  Git supports **seamless branching and merging**, which helps in visualizing and navigating a non-linear development. A branch in Git represents a single commit. We can construct the full branch structure with the help of its parental commit.
- **Branching and Merging**
  Branching and merging are the **great feature**s of Git, which makes it different from the other SCM tools. Git allows the **creation of multiple branches** without affecting each other. We can perform tasks like **creation**, **deletion**, and **merging** on branches, and these tasks take a few seconds only. Below are some features that can be achieved by branching:

- We can **create a separate branch** for a new module of the project, commit and delete it whenever we want.
- We can have a **production branch**, which always has what goes into production and can be merged for testing in the test branch.
- We can create a **demo branch** for the experiment and check if it is working. We can also remove it if needed.
- The core benefit of branching is if we want to push something to a remote repository, we do not have to push all of our branches. We can select a few of our branches, or all of them together.

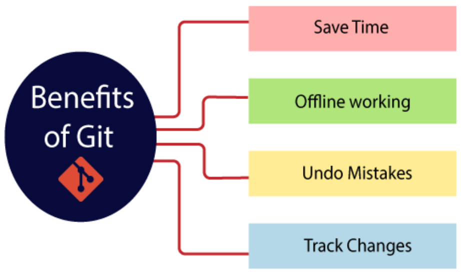
- **Data Assurance**
  The Git data model ensures the **cryptographic integrity** of every unit of our project. It provides a **unique commit ID** to every commit through a **SHA algorithm**. We can **retrieve** and **update** the commit by commit ID. Most of the centralized version control systems do not provide such integrity by default.
- **Staging Area**
  The **Staging area** is also a **unique functionality** of Git. It can be considered as a **preview of our next commit**, moreover, an **intermediate area** where commits can be formatted and reviewed before completion. When you make a commit, Git takes changes that are in the staging area and make them as a new commit. We are allowed to add and remove changes from the staging area. The staging area can be considered as a place where Git stores the changes.
  Although, Git doesn't have a dedicated staging directory where it can store some objects representing file changes (blobs). Instead of this, it uses a file called index.



  Another feature of Git that makes it apart from other SCM tools is that **it is possible to quickly stage some of our files and commit them without committing other modified files in our working directory.**
- **Maintain the clean history**
  Git facilitates with Git Rebase; It is one of the most helpful features of Git. It fetches the latest commits from the master branch and puts our code on top of that. Thus, it maintains a clean history of the project.

Benefits of GIT

A version control application allows us to keep track of all the changes that we make in the files of our project. Every time we make changes in files of an existing project, we can push those changes to a repository. Other developers are allowed to pull your changes from the repository and continue to work with the updates that you added to the project files.
Some significant benefits of using Git are as follows:



Git Installation:
Installing on Windows
There are also a few ways to install Git on Windows. The most official build is available for download on the Git website. Just go to https://git-scm.com/download/win and the download will start automatically. Note that this is a project called Git for Windows, which is separate from Git itself; for more information on it, go to https://gitforwindows.org.
To get an automated installation you can use the Git Chocolatey package. Note that the Chocolatey package is community maintained.
GitBash:
Git Bash is an application for the Windows environment. It is used as Git command line for windows. Git Bash provides an emulation layer for a Git command-line experience. Bash is an abbreviation of Bourne Again Shell. Git package installer contains Bash, bash utilities, and Git on a Windows operating system.

Basic Git Commands:
1) Git Config command:
This command configures the user. The Git config command is the first and necessary command used on the Git command line. This command sets the author name and email address to be used with your commits. Git config is also used in other scenarios.
Syntax

```
$ git config --global user.name "username1"
$ git config --global user.email “username11@gmail.com"
```

2) Git Init command

This command is used to create a local repository.

Syntax

```
$ git init Demo
```

3) Git clone command

This command is used to make a copy of a repository from an existing URL. If I want a local copy of my repository from GitHub, this command allows creating a local copy of that repository on your local directory from the repository URL.

Syntax

```
$ git clone URL
```

4) Git add command

This command is used to add one or more files to staging (Index) area.

Syntax

To add one file

```
$ git add Filename
```

To add more than one file

```
$ git add*
```

5) Git commit command

Commit command is used in two scenarios. They are as follows.

Git commit -m

This command changes the head. It records or snapshots the file permanently in the version history with a message.

Syntax

```
$ git commit -m " Commit Message"
```

Git commit -a

This command commits any files added in the repository with git add and also commits any files you've changed since then.

Syntax

```
$ git commit -a
```

6) Git status command

The status command is used to display the state of the working directory and the staging area. It allows you to see which changes have been staged, which haven't, and which files aren't being tracked by Git. It does not show you any information about the committed project history. For this, you need to use the git log. It also lists the files that you've changed and those you still need to add or commit.

Syntax

```
$ git status
```

7) Git push Command

It is used to upload local repository content to a remote repository. Pushing is an act of transfer commits from your local repository to a remote repo. It's the complement to git fetch, but whereas fetching imports commits to local branches on comparatively pushing exports commits to remote branches. Remote branches are configured by using the git remote command. Pushing is capable of overwriting changes, and caution should be taken when pushing.

Git push command can be used as follows.

Git push origin master

This command sends the changes made on the master branch, to your remote repository.

Syntax

```
$ git push [variable name] master
```

Git push -all

This command pushes all the branches to the server repository.

Syntax

```
$ git push --all
```

8) Git pull command

Pull command is used to receive data from GitHub. It fetches and merges changes on the remote server to your working directory.

Syntax

```
$ git pull URL
```

9) Git Branch Command

This command lists all the branches available in the repository.

Syntax

```
$ git branch
```

10) Git Merge Command

This command is used to merge the specified branch?s history into the current branch.

Syntax

```
$ git merge BranchName
```

11) Git log command

This command is used to check the commit history.

Syntax

```
$ git log
```

By default, if no argument passed, Git log shows the most recent commits first. We can limit the number of log entries displayed by passing a number as an option, such as -3 to show only the last three entries.

```
$ git log -3
```

12) Git remote command

Git Remote command is used to connect your local repository to the remote server. This command allows you to create, view, and delete connections to other repositories. These connections are more like bookmarks rather than direct links into other repositories. This command doesn't provide real-time access to repositories.

Git Init

The git init command is the first command that you will run on Git. The git init command is used to create a new blank repository. It is used to make an existing project as a Git project. Several Git commands run inside the repository, but init command can be run outside of the repository.

The git init command creates a .git subdirectory in the current working directory. This newly created subdirectory contains all of the necessary metadata. These metadata can be categorized into objects, refs, and temp files. It also initializes a HEAD pointer for the master branch of the repository.

Creating the first repository

Git version control system allows you to share projects among developers. For learning Git, it is essential to understand that how can we create a project on Git. A repository is a directory that contains all the project-related data. There can also be more than one project on a single repository.

We can create a repository for blank and existing projects. Let's understand how to create a repository.

Create a Repository for a Blank (New) Project:

To create a blank repository, open command line on your desired directory and run the init command as follows:

```
$ git init
```

The above command will create an empty .git repository. Suppose we want to make a git repository on our desktop. To do so, open Git Bash on the desktop and run the above command.

Outcome: Install, configure and use Git on different platforms

CONCLUSIONS:

Students will be able to install and configure Git as VCS

Experiment 1:

Introduction

Version Control System

Types of version control system

Version Control System tools

History of Git

What is Git?

Properties of Git

Installation of Git on Windows & Linux

Different states of Git

Git Setup and configuration

Terminology of Git

Simple Git commands

Conclusion

<u>4. Lab Exercise</u>

<u>Exercise No 2: (2 Hours) – 1 Practical</u>

Aim: - Branching and Merging, Stashing, Rebasing, Reverting and Resetting.

Objectives:

1. Students should able to understand the concept of Branching and Merging, Stashing, Rebasing, Reverting and Resetting with Git

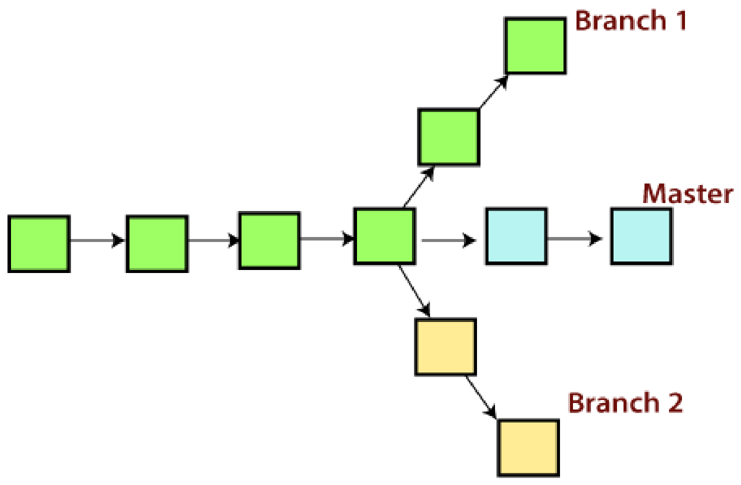
Pre-requisites: Knowledge of version control system

THEORY:

1. Branching & Merging:

Git Branch

A branch is a version of the repository that diverges from the main working project. It is a feature available in most modern version control systems. A Git project can have more than one branch. These branches are a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug, you spawn a new branch to summarize your changes. So, it is complex to merge the unstable code with the main code base and also facilitates you to clean up your future history before merging with the main branch.



Git Master Branch

The master branch is a default branch in Git. It is instantiated when first commit made on the project. When you make the first commit, you're given a master branch to the starting commit point. When you start making a commit, then master branch pointer automatically moves forward. A repository can have only one master branch.

Master branch is the branch in which all the changes eventually get merged back. It can be called as an official working version of your project.

Operations on Branches

We can perform various operations on Git branches. The git branch command allows you to create, list, rename and delete branches. Many operations on branches are applied by git checkout and git merge command. So, the git branch is tightly integrated with the git checkout and git merge commands.

The Operations that can be performed on a branch:

Create Branch

You can create a new branch with the help of the git branch command. This command will be used as:

Syntax:

```
$ git branch  <branch name>
```

This command will create the branch B1 locally in Git directory.

List Branch

You can List all of the available branches in your repository by using the following command. Either we can use git branch - list or git branch command to list the available branches in the repository.

Syntax:

```
$ git branch --list
```

or

```
$ git branch
```

Here, both commands are listing the available branches in the repository. The symbol * is representing currently active branch.

Delete Branch

You can delete the specified branch. It is a safe operation. In this command, Git prevents you from deleting the branch if it has unmerged changes. Below is the command to do this.

```
$ git branch -d<branch name>
```

The git branch d command can be used in two formats. Another format of this command is git branch D. The 'git branch D' command is used to delete the specified branch.

```
$ git branch -D <branch name>
```

Delete a Remote Branch

You can delete a remote branch from Git desktop application. Below command is used to delete a remote branch:

Syntax:

```
$ git push origin -delete <branch name>
```

Switch Branch

Git allows you to switch between the branches without making a commit. You can switch between two branches with the git checkout command. To switch between the branches, below command is used:

```
$ git checkout<branch name>
```

Switch from master Branch

You can switch from master to any other branch available on your repository without making any commit.

Syntax:

```
$ git checkout <branch name>
```

Switch to master branch

You can switch to the master branch from any other branch with the help of below command.

Syntax:

```
$ git branch -m master
```

Rename Branch

We can rename the branch with the help of the git branch command. To rename a branch, use the below command:

Syntax:

```
$ git branch -m <old branch name><new branch name>
```

Merge Branch

Git allows you to merge the other branch with the currently active branch. You can merge two branches with the help of git merge command. Below command is used to merge the branches:

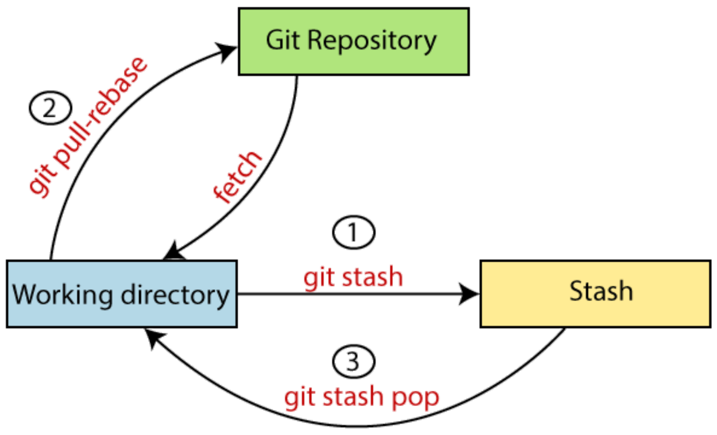
Syntax:

```
$ git merge <branch name>
```

2. **Stashing**

Sometimes you want to switch the branches, but you are working on an incomplete part of your current project. You don't want to make a commit of half-done work. Git stashing allows you to do so. The git stash command enables you to switch branches without committing the current branch.

The below figure demonstrates the properties and role of stashing concerning repository and working directory.



Git Stash

Generally, the stash's meaning is "store something safely in a hidden place." The sense in Git is also the same for stash; Git temporarily saves your data safely without committing.

Stashing takes the messy state of your working directory, and temporarily save it for further use.

Many options are available with git stash. Some useful options are given below:

Git stash

Git stash save

Git stash list

Git stash apply

Git stash changes

Git stash pop

Git stash drop

Git stash clear

Git stash branch

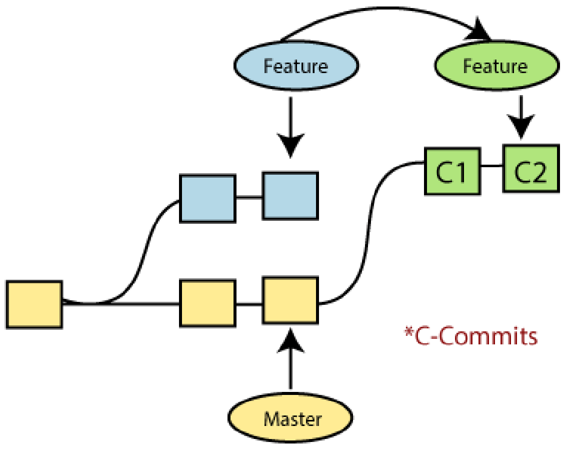
3. Rebasing,

Git Rebase

Rebasing is a process to reapply commits on top of another base trip. It is used to apply a sequence of commits from distinct branches into a final commit. It is an alternative of git merge command. It is a linear process of merging.

In Git, the term rebase is referred to as the process of moving or combining a sequence of commits to a new base commit. Rebasing is very beneficial and it visualized the process in the environment of a feature branching workflow.

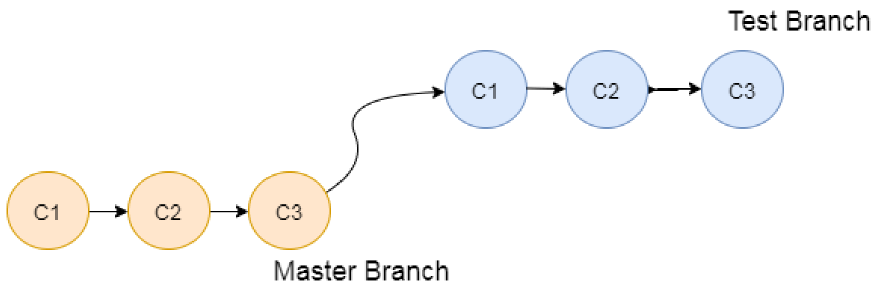
It is good to rebase your branch before merging it.



Git Rebase

Generally, it is an alternative of git merge command. Merge is always a forward changing record. Comparatively, rebase is a compelling history rewriting tool in git. It merges the different commits one by one.

Suppose you have made three commits in your master branch and three in your other branch named test. If you merge this, then it will merge all commits in a time. But if you rebase it, then it will be merged in a linear manner. Consider the below image:



Git Rebase

The above image describes how git rebase works. The three commits of the master branch are merged linearly with the commits of the test branch.

Merging is the most straightforward way to integrate the branches. It performs a three-way merge between the two latest branch commits.

How to Rebase

When you made some commits on a feature branch (test branch) and some in the master branch. You can rebase any of these branches. Use the git log command to track the changes (commit history). Checkout to the desired branch you want to rebase. Now perform the rebase command as follows:

Syntax:

```
$git rebase <branch name>
```

4. Reverting and Resetting.

In Git, the term revert is used to revert some changes. The git revert command is used to apply revert operation. It is an undo type command. However, it is not a traditional undo alternative. It does not delete any data in this process; instead, it will create a new change with the opposite effect and thereby undo the specified commit. Generally, git revert is a commit.

It can be useful for tracking bugs in the project. If you want to remove something from history then git revert is a wrong choice.

Moreover, we can say that git revert records some new changes that are just opposite to previously made commits. To undo the changes, run the below command:

Syntax:

```
$ git revert
```

5. Reset

Git Reset

The term reset stands for undoing changes. The git reset command is used to reset the changes. The git reset command has three core forms of invocation.

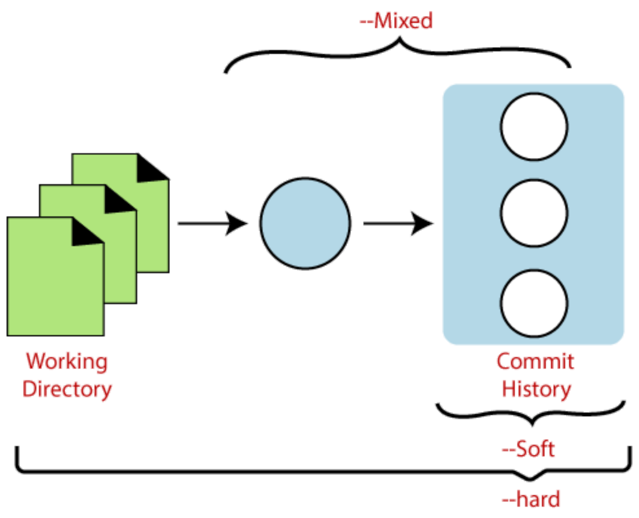
Soft

Mixed

Hard

If we say in terms of Git, then Git is a tool that resets the current state of HEAD to a specified state. It is a sophisticated and versatile tool for undoing changes. It acts as a time machine for Git. You can jump up and forth between the various commits. Each of these reset variations affects specific trees that git uses to handle your file in its content.

Additionally, git reset can operate on whole commits objects or at an individual file level. Each of these reset variations affects specific trees that git uses to handle your file and its contents.



Git Reset

Git uses an index (staging area), HEAD, and working directory for creating and reverting commits. If you have no idea about what is Head, trees, index, then do visit here Git Index and Git Head.

The working directory lets you change the file, and you can stage into the index. The staging area enables you to select what you want to put into your next commit. A commit object is a cryptographically hashed version of the content. It has some Metadata and points which are used to switch on the previous commits.

OUTCOME: Use git platform for Branching and Merging, Stashing, Rebasing, Reverting and Resetting

CONCLUSIONS:

Students will be able to perform different git operations

Task to conduct in Lab :

- **Install git**
- **Create a repository with your name**
- **Create minimum four java files or copy from your earlier work**
- **Perform add & commit operations**
- **Observe the output after add & commit**
- **Use all the options of *status* & *log* command**
- **Perform stashing operations**
- **Create minimum two branches & add some files or features in it**

- **Merge the branch with master**
- **Demonstrate the merge conflict**
- **Demonstrate the rebasing**
-

Journal Writeup:

Introduction

Git Branching

Operations on branches

Branch Merging

Merge Confict and its resolution

Git Rebase

Rebase working

Interactive Rebasing

Different options with interactive rebasing

Git Merge vs Rebase

Git Squashing

Challenges with Squashing

Conclusion

5. Lab Exercise

Exercise No 3: (2 Hours) – 1 Practical

Aim: Study and implementation of various git commands to push and pull a repository from GitHub

Objective:

1. To Create a free GitHub account and a GitHub repository.
2. To study PUSH & PULL git commands

THEORY:
GitHub is an online software development platform. It's used for storing, tracking, and collaborating on software projects. It makes it easy for developers to share code files and collaborate with fellow developers on open-source projects. GitHub also serves as a social networking site where developers can openly network, collaborate, and pitch their work.

How does GitHub work?
GitHub users create accounts, upload files, and create coding projects. But the real work of GitHub happens when users begin to collaborate.
While anyone can code independently, teams of people build most development projects. Sometimes these teams are all in one place at once time, but more often they work asynchronously. There are many challenges to creating collaborative projects with distributed teams. GitHub makes this process much simpler in a few different ways.

Create a free GitHub account

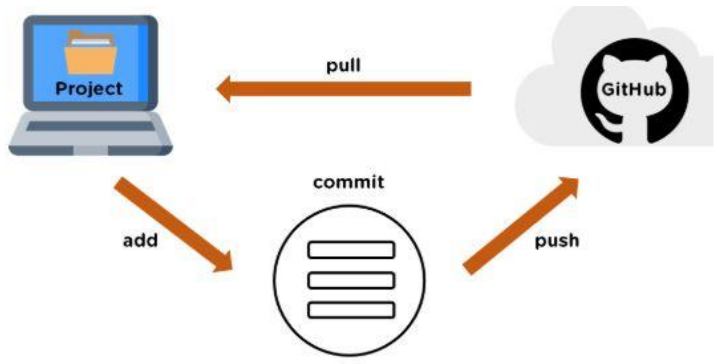
1. Go to GitHub.com and sign up for a free GitHub account.
2. Enter your email address and press the **Sign up for GitHub** button. If you have an account, press the **Sign in** button and login.

Create a GitHub repository

- If you just created your GitHub account, create your first repository by pressing the **Create repository** button, then go to step **3**.
- If you already have a GitHub account, sign into GitHub and on your account page, press the **New** repository button.
- Name your repository **Git_Demo** and give it a good description like: *This repository contains the devops files* and make sure the **Public** option is selected.
- Scroll down on that page and select **Add a README file** and then press the **Create repository** button to create the new repository.

Git Push Command
The git push command uploads content from a local repository to a remote repository. Pushing refers to the process of moving commits from one repository to another. Pushing is the equivalent of git fetch, except that instead of importing commits to a local branch, it exports commits to an external branch.

The Git push command is used to push the local repository content to a remote repository. After a local repository has been modified, a push is executed to share the modifications with remote team members. Pushing is the way commits are transferred from the local repository to the remote repository.



PUSH

1. Open Git Bash in your Desktop/system and configuring it with a user name and email ID.

```
$ Git config --global user.name "jneccse"
$ Git config --global user.email jnec.cse@gmail.com
$ Git config –list
```

2. Check current working directory

```
$ pwd
```

To change the path use cd command ($ cd path_name)

3. To create a repository in the working directory, use the following commands:

```
$ mkdir git_demo
$ cd git_demo
$pwd
```

We will now initialize a repository to our folder.

```
$ git init
```

Something called the "master" appears on the screen. Whenever a Git repository is created for the first time, it creates a branch, and the name of the branch is master. Navigate to the folder; you can find a hidden ".git" folder.

If you check the folder, you can see several directories and configurations. Make sure you don't make any changes to any of the directories.

.git hidden folder is created when a repository is initialized.

4. Create text files in folder git_demo with commands

```
$ touch abc.txt
$ notepad abc.txt
```

Type “Hello DevOps Lab” inside file with the notepad. Save it & Close.

5. Check status of file

```
$ git status
```

Output shows that there isn't a file committed yet, and there are untracked files. The untracked files can be seen in red.

For Git to track that file, add command is used. If you know the exact name of the file, you can specify it and simply type the following command:

```
$ git add .
```

6. Commit the file

```
$ git commit -m "abc"
```
Lets check the status of file again
```
$ git status
```
You'll notice that there are no more commits to be made, as there was a single notepad and that was committed in the previous step.
Next, check all the information regarding the commits that were made.
```
$ git log
```
This displays the commit ID, author's name, and email ID used. You can also find the date and commit message on the screen.

7. Add the URL copied, which is your remote repository to where your local content from your repository is pushed

Now, let's push the notepad file on GitHub. Open your GitHub account, and create a new repository. The name of the repository will be "Git_Demo."

Copy the repository URL from GitHub account and Paste the copied URL onto the Git Bash. The HTTPS or URL is copied from the given GitHub account, which is the place of the remote repository.

```
$ git remote add origin 'your_url_name'
```

```
$ git remote -v
```

8. Push the code in your local repository to GitHub

Let’s PUSH the content to remote repository of GitHub
```
$ git push -u origin master
```

In the code, the origin is your default remote repository name and '-u' flag is upstream, which is equivalent to '-set-upstream.' and the master is the branch, name.upstream is the repository that we have cloned the project.

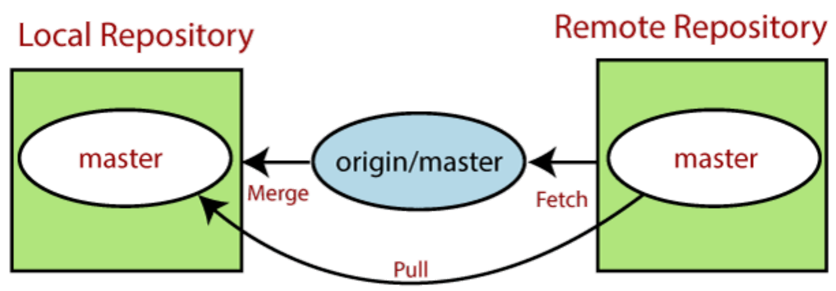
The repository is created on the server, and the content is pushed into that repository. It links the master branch on the local repository to the master branch on the server.

Fill in your GitHub username and password.

9. View your files in your repository hosted on GitHub

You can finally see the file hosted on GitHub.

Git PULL Command
The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content. Merging remote upstream changes into your local repository is a common task in Git-based collaboration work flows. The git pull command is actually a combination of two other commands, git fetch followed by git merge. In the first stage of operation git pull will execute a git fetch scoped to the local branch that HEAD is pointed at. Once the content is downloaded, git pull will enter a merge workflow. A new merge commit will be-created and HEAD updated to point at the new commit.



The pull command is used to access the changes (commits)from a remote repository to the local repository. It updates the local branches with the remote-tracking branches. Remote tracking branches are branches that have been set up to push and pull from the remote repository. Generally, it is a collection of the fetch and merges command. First, it fetches the changes from remote and combined them with the local repository.

Syntax:

```
$ git pull <remote branch URL>
$ git pull <option> [<repository URL><refspec>...]
```

In which:

<option>: Options are the commands; these commands are used as an additional option in a particular command. Options can be -q (quiet), -v (verbose), -e(edit) and more.

<repository URL>: Repository URL is your remote repository's URL where you have stored your original repositories like GitHub or any other git service. This URL looks like:

https://github.com/jneccse/Git_Demo.git

To access this URL, go to your account on GitHub and select the repository you want to clone. After that, click on the clone or download option from the repository menu. A new pop up window will open, select clone with https option from available options.

Copy the highlighted URL. This URL is used to Clone the repository.

<Refspec>: A ref is referred to commit, for example, head (branches), tags, and remote branches. You can check head, tags, and remote repository in .git/ref directory on your local repository. Refspec specifies and updates the refs.

OUTCOME: Use PUSH and PULL commands with git

CONCLUSIONS:

Learned the basics of the push & pull command and followed a hands-on demo of the Git Push & pull command using Git Bash. In the demo, we saw how files from the local repository could be pushed to the remote repository. The process makes it possible for the team to stay updated on different people performing different tasks in the same program.

Laboratory Task :

- **Create account on GitHub**
- **Create repository on GitHub**
- **Create local repository**
-

- **Clone your last year mini project from Github**
- **Create two branches for your mini project repository**
- **List all repositories**
- **Add new features to your project in both branches**
- **Merge the branches with master**
- **Push the local repository to GitHub**
- **Install GitLab, Bitbucket & Bazar**
- **Use different features of above tools**
-

Journal Writeup:
Introduction
GitHub
Remote Repository
Create Remote Repository
Push Local Repository to Remote
Push a branch to GitHub
Pull from Remote to Local Repository
Git pull vs git fetch
Git remote repository management commands
Conclusion

6. Lab Exercise

Exercise No 4: (2 Hours) – 1 Practical

Aim: - Creating simple Maven project and perform unit test and resolve dependencies
Objective:
1. Understanding Maven Project

THEORY:
Maven tutorial provides basic and advanced concepts of apache maven technology. Our maven tutorial is developed for beginners and professionals.
Maven is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation.
It simplifies the build process like ANT. But it is too much advanced than ANT.
Current version of Maven is 3.

Understanding the problem without Maven
There are many problems that we face during the project development. They are discussed below:
1) Adding set of Jars in each project: In case of struts, spring, hibernate frameworks, we need to add set of jar files in each project. It must include all the dependencies of jars also.

2) Creating the right project structure: We must create the right project structure in servlet, struts etc, otherwise it will not be executed.
3) Building and Deploying the project: We must have to build and deploy the project so that it may work.

What it does?
Maven simplifies the above mentioned problems. It does mainly following tasks.

It makes a project easy to build
It provides uniform build process (maven project can be shared by all the maven projects)
It provides project information (log document, cross referenced sources, mailing list, dependency list, unit test reports etc.)
It is easy to migrate for new features of Maven
Apache Maven helps to manage
Builds
Documentation
Reporing
SCMs
Releases
Distribution

What is Build Tool
A build tool takes care of everything for building a process. It does following:
Generates source code (if auto-generated code is used)
Generates documentation from source code
Compiles source code
Packages compiled code into JAR of ZIP file
Installs the packaged code in local repository, server repository, or central repository
How to install Maven on windows
To install maven on windows, you need to perform following steps:
Download maven and extract it
Add JAVA_HOME and MAVEN_HOME in environment variable
Add maven path in environment variable
Verify Maven
1) Download Maven
To install maven on windows, you need to download apache maven first.
Download Maven latest Maven software from Download latest version of Maven
For example: apache-maven-3.1.1-bin.zip
Extract it.

2) Add MAVEN_HOME in environment variable
Right click on MyComputer -> properties -> Advanced System Settings -> Environment variables -> click new button

Now add MAVEN_HOME in variable name and path of maven in variable value. It must be the home directory of maven i.e. outer directory of bin. For example: E:\apache-maven-3.1.1
Now click on OK button.

3) Add Maven Path in environment variable
Click on new tab if path is not set, then set the path of maven. If it is set, edit the path and append the path of maven.
Here, we have installed JDK and its path is set by default, so we are going to append the path of maven.
The path of maven should be %maven home%/bin. For example, E:\apache-maven-3.1.1\bin .
4) Verify maven
To verify whether maven is installed or not, open the command prompt and write:

```
mvn −version
```

Now it will display the version of maven and jdk including the maven home and java home.
Set up the project
First you'll need to setup a Java project for Maven to build. To keep the focus on Maven, make the project as simple as possible for now. Create this structure in a project folder of your choosing.

Create the directory structure
In a project directory of your choosing, create the following subdirectory structure; for example, with mkdir -p src/main/java/hello on systems:

Within the src/main/java/hello directory, you can create any Java classes you want.
Create `HelloWorld.java` and `Greeter.java`.

src/main/java/hello/HelloWorld.java

```
package hello;

public class HelloWorld {
  public static void main(String[] args) {
    Greeter greeter = new Greeter();
    System.out.println(greeter.sayHello());
  }
}
```

src/main/java/hello/Greeter.java

```
package hello;

public class Greeter {
  public String sayHello() {
    return "Hello world!";
  }
}
```
Now that you have a project that is ready to be built with Maven, the next step is to install Maven.
We have already installed Maven

Define a simple Maven build
Now that Maven is installed, you need to create a Maven project definition. Maven projects are defined with an XML file named pom.xml. Among other things, this file gives the project's name, version, and dependencies that it has on external libraries.
Create a file named pom.xml at the root of the project (i.e. put it next to the src folder) and give it the following contents:

pom.xml
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>

   <groupId>org.springframework</groupId>
   <artifactId>gs-maven</artifactId>
   <packaging>jar</packaging>
   <version>0.1.0</version>

   <properties>
      <maven.compiler.source>1.8</maven.compiler.source>
      <maven.compiler.target>1.8</maven.compiler.target>
   </properties>

   <build>
      <plugins>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-shade-plugin</artifactId>
            <version>3.2.4</version>
            <executions>
               <execution>
                  <phase>package</phase>
                  <goals>
                     <goal>shade</goal>
                  </goals>
```

```
                    <configuration>
                        <transformers>
                            <transformer

implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
                                <mainClass>hello.HelloWorld</mainClass>
                            </transformer>
                        </transformers>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
  </build>
</project>
```
With the exception of the optional <packaging> element, this is the simplest possible pom.xml file necessary to build a Java project.

Build Java code
Maven is now ready to build the project. You can execute several build lifecycle goals with Maven now, including goals to compile the project's code, create a library package (such as a JAR file), and install the library in the local Maven dependency repository.

To try out the build, issue the following at the command line:

```
mvn compile
```
This will run Maven, telling it to execute the compile goal. When it's finished, you should find the compiled .class files in the target/classes directory.
Since it's unlikely that you'll want to distribute or work with .class files directly, you'll probably want to run the package goal instead:

```
mvn package
```
The package goal will compile your Java code, run any tests, and finish by packaging the code up in a JAR file within the target directory. The name of the JAR file will be based on the project's <artifactId> and <version>. For example, given the minimal pom.xml file from before, the JAR file will be named gs-maven-0.1.0.jar.

To execute the JAR file run:

```
java -jar target/gs-maven-0.1.0.jar
```

Maven also maintains a repository of dependencies on your local machine (usually in a .m2/repository directory in your home directory) for quick access to project dependencies. If you'd like to install your project's JAR file to that local repository, then you should invoke the install goal:

```
mvn install
```

The install goal will compile, test, and package your project's code and then copy it into the local dependency repository, ready for another project to reference it as a dependency.

Declare Dependencies

The simple Hello World sample is completely self-contained and does not depend on any additional libraries. Most applications, however, depend on external libraries to handle common and complex functionality.

For example, suppose that in addition to saying "Hello World!", you want the application to print the current date and time. While you could use the date and time facilities in the native Java libraries, you can make things more interesting by using the Joda Time libraries.

First, change HelloWorld.java to look like this:

src/main/java/hello/HelloWorld.java

```
package hello;

import org.joda.time.LocalTime;

public class HelloWorld {
  public static void main(String[] args) {
    LocalTime currentTime = new LocalTime();
    System.out.println("The current local time is: " + currentTime);
    Greeter greeter = new Greeter();
    System.out.println(greeter.sayHello());
  }
}
```

Here HelloWorld uses Joda Time's LocalTime class to get and print the current time.

If you were to run mvn compile to build the project now, the build would fail because you've not declared Joda Time as a compile dependency in the build. You can fix that by adding the following lines to pom.xml (within the <project> element):

```
<dependencies>
		<dependency>
			<groupId>joda-time</groupId>
			<artifactId>joda-time</artifactId>
			<version>2.9.2</version>
		</dependency>
```

```
</dependencies>
```
This block of XML declares a list of dependencies for the project. Specifically, it declares a single dependency for the Joda Time library. Within the <dependency> element, the dependency coordinates are defined by three sub-elements:

<groupId> - The group or organization that the dependency belongs to.

<artifactId> - The library that is required.

<version> - The specific version of the library that is required.

By default, all dependencies are scoped as compile dependencies. That is, they should be available at compile-time (and if you were building a WAR file, including in the /WEB-INF/libs folder of the WAR). Additionally, you may specify a <scope> element to specify one of the following scopes:

provided - Dependencies that are required for compiling the project code, but that will be provided at runtime by a container running the code (e.g., the Java Servlet API).

test - Dependencies that are used for compiling and running tests, but not required for building or running the project's runtime code.

Now if you run mvn compile or mvn package, Maven should resolve the Joda Time dependency from the Maven Central repository and the build will be successful.

Write a Test
First add JUnit as a dependency to your pom.xml, in the test scope:

```
<dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
</dependency>
```
Then create a test case like this:

src/test/java/hello/GreeterTest.java

```
package hello;

import static org.hamcrest.CoreMatchers.containsString;
import static org.junit.Assert.*;
import org.junit.Test;

public class GreeterTest {

  private Greeter greeter = new Greeter();

  @Test
  public void greeterSaysHello() {
    assertThat(greeter.sayHello(), containsString("Hello"));
  }

}
```
Maven uses a plugin called "surefire" to run unit tests. The default configuration of this plugin compiles and runs all classes in src/test/java with a name matching *Test. You can run the tests on the command line like this

```
mvn test
```
or just use mvn install step as we already showed above (there is a lifecycle definition where "test" is included as a stage in "install").

Here’s the completed pom.xml file:

pom.xml
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>

	<groupId>org.springframework</groupId>
	<artifactId>gs-maven</artifactId>
	<packaging>jar</packaging>
	<version>0.1.0</version>

	<properties>
		<maven.compiler.source>1.8</maven.compiler.source>
		<maven.compiler.target>1.8</maven.compiler.target>
	</properties>

	<dependencies>
		<!-- tag::joda[] -->
		<dependency>
```

```
                <groupId>joda-time</groupId>
                <artifactId>joda-time</artifactId>
                <version>2.9.2</version>
            </dependency>
            <!-- end::joda[] -->
            <!-- tag::junit[] -->
            <dependency>
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>4.12</version>
                <scope>test</scope>
            </dependency>
            <!-- end::junit[] -->
        </dependencies>

        <build>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-shade-plugin</artifactId>
                    <version>3.2.4</version>
                    <executions>
                        <execution>
                            <phase>package</phase>
                            <goals>
                                <goal>shade</goal>
                            </goals>
                            <configuration>
                                <transformers>
                                    <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">

<mainClass>hello.HelloWorld</mainClass>
                                    </transformer>
                                </transformers>
                            </configuration>
                        </execution>
                    </executions>
                </plugin>
            </plugins>
        </build>

</project>
```

CONCLUSIONS:
Students will be able to build maven project with unit testing for resolving dependencies

Experiment 4: Apache Maven Project

Introduction

What is Apache Maven ?

Feaures of Maven

Working of Maven

Installation of Apache Maven

Maven Repository

Build life cycle

Maven CLI commands

steps to create Maven Project

Details of POM.xml

Maven Plugins

Maven - Project Templates

Testing web application using Maven

Conclusion

7. Lab Exercise

Exercise No 5: (2 Hours) – 1 Practical

Aim: - Installing and Configuring Jenkins

Objective:
1. Understand the role of Jenkins
2. Install and Configure Jenkins

THEORY:
Jenkins is an open source continuous integration/continuous delivery and deployment (CI/CD) automation software DevOps tool written in the Java programming language. It is used to implement CI/CD workflows, called pipelines.

Pipelines automate testing and reporting on isolated changes in a larger code base in real time and facilitates the integration of disparate branches of the code into a main branch. They also rapidly detect defects in a code base, build the software, automate testing of their builds, prepare the code base for deployment (delivery), and ultimately deploy code to containers and virtual machines, as well as bare metal and cloud servers. There are several commercial versions of Jenkins. This definition only describes the upstream Open Source project.

History

Jenkins is a fork of a project called Hudson, which was trademarked by Oracle. Hudson was eventually donated to the Eclipse Foundation and is no longer under development. Jenkins development is now managed as an open source project under the governance of the CD Foundation, an organization within the Linux Foundation.

Jenkins and CI/CD

Over time, continuous delivery and deployment features have been added to Jenkins. Continuous delivery is the process of automating the building and packaging of code for eventual deployment to test, production staging, and production environments. Continuous deployment automates the final step of deploying the code to its final destination.

In both cases, automation reduces the number of errors that occur because the correct steps and best practices are encoded into Jenkins. Jenkins describes a desired state and the automation server ensures that that state is achieved. In addition, the velocity of releases can be increased since deployments are no longer bounded by personnel limitations, such as operator availability. Finally, Jenkins reduces stress on the development and operations team, by removing the need for middle of the night and weekend rollouts.

How Jenkins works

Jenkins runs as a server on a variety of platforms including Windows, MacOS, Unix variants and especially, Linux. It requires a Java 8 VM and above and can be run on the Oracle JRE or OpenJDK. Usually, Jenkins runs as a Java servlet within a Jetty application server. It can be run on other Java application servers such as Apache Tomcat. More recently, Jenkins has been adapted to run in a Docker container. There are read-only Jenkins images available in the Docker Hub online repository.

To operate Jenkins, pipelines are created. A pipeline is a series of steps the Jenkins server will take to perform the required tasks of the CI/CD process. These are stored in a plain text

Jenkinsfile. The Jenkinsfile uses a curly bracket syntax that looks similar to JSON. Steps in the pipeline are declared as commands with parameters and encapsulated in curly brackets. The Jenkins server then reads the Jenkinsfile and executes its commands, pushing the code down the pipeline from committed source code to production runtime. A Jenkinsfile can be created through a GUI or by writing code directly.

Plugins

A plugin is an enhancement to the Jenkins system. They help extend Jenkins capabilities and integrated Jenkins with other software. Plugins can be downloaded from the online Jenkins Plugin repository and loaded using the Jenkins Web UI or CLI. Currently, the Jenkins community claims over 1500 plugins available for a wide range of uses.

Plugins help to integrate other developer tools into the Jenkins environment, add new user interface elements to the Jenkins Web UI, help with administration of Jenkins, and enhance Jenkins for build and source code management. One of the more common uses of plugins is to provide integration points for CI/CD sources and destinations. These include software version control systems (SVCs) such as Git and Atlassian BitBucket, container runtime systems -- especially Docker, virtual machine hypervisors such as VMware vSphere, public cloud instances including Google Cloud Platform and Amazon AWS, and private cloud systems such as OpenStack. There are also plugins that assist in communicating with operating systems over FTP, CIFS, and SSH.

Steps to Install Jenkins on Windows

1. Install Java Development Kit (JDK)

Download JDK 8 and choose windows 32-bit or 64-bit according to your system configuration. Click on "accept the license agreement."

2. Set the Path for the Environmental Variable for JDK

Go to System Properties. Under the "Advanced" tab, select "Environment Variables."

Under system variables, select "new." Then copy the path of the JDK folder and paste it in the corresponding value field. Similarly, do this for JRE.

Under system variables, set up a bin folder for JDK in PATH variables.

Go to command prompt and type the following to check if Java has been successfully installed:

```
C:\Users\Simplilearn>java -version
```

3. Download and Install Jenkins

Download Jenkins. Under LTS, click on windows.

After the file is downloaded, unzip it. Click on the folder and install it. Select "finish" once done.

4. Run Jenkins on Localhost 8080

Once Jenkins is installed, explore it. Open the web browser and type "localhost:8080".

Enter the credentials and log in. If you install Jenkins for the first time, the dashboard will ask you to install the recommended plugins. Install all the recommended plugins.

5. Jenkins Server Interface

New Item allows you to create a new project.

Build History shows the status of your builds.

Manage System deals with the various configurations of the system.

6. Build and Run a Job on Jenkins

Select a new item (Name - Jenkins_demo). Choose a freestyle project and click Ok.

Under the General tab, give a description like "This is my first Jenkins job." Under the "Build Triggers" tab, select add built step and then click on the "Execute Windows" batch command.

In the command box, type the following: echo "Hello... This is my first Jenkins Demo: %date%: %time% ". Click on apply and then save.

Select build now. You can see a building history has been created. Click on that. In the console output, you can see the output of the first Jenkins job with time and date.

Congratulations, you've just installed Jenkins on your Windows system!

OUTCOME: Create Continuous integration using Jenkins

CONCLUSIONS:

Students will be able to install and configure jenkins

Experiment 5: Jenkins Installation and Configuration
Introduction
What is Jenkins
History of Jenkins
Jenkins Architecture
Installation of Jenkins on windows and linux
Plugin installtion
Jenkins Configuration
Steps to set JENKINS_HOME

Jenkins URL
Email Notification
Create/Add Users in Jenkins
Role Management in Jenkins
System Information
System log
Conclusion

8. Lab Exercise

Exercise No 6: (2 Hours) – 1 Practical

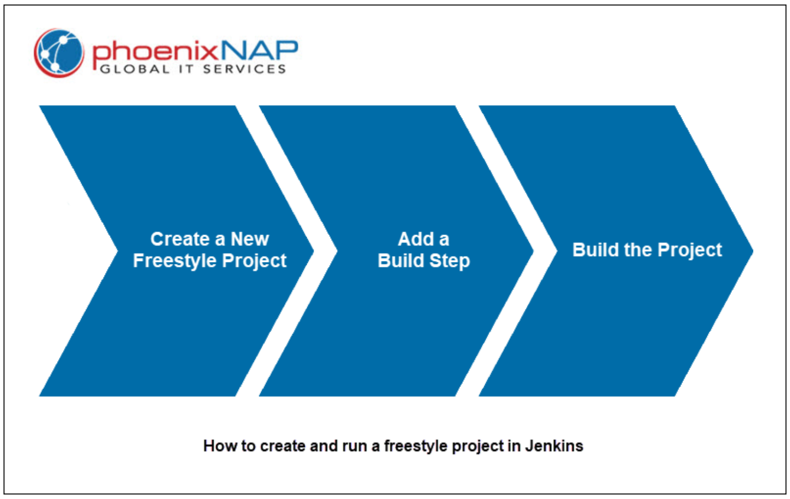
Aim: - Creating a build using Jenkins

Pre-requisites: A copy of Jenkins installed and ready to use
Access to a web browser

Objective
1. Understand build job in Jenkins

THEORY:

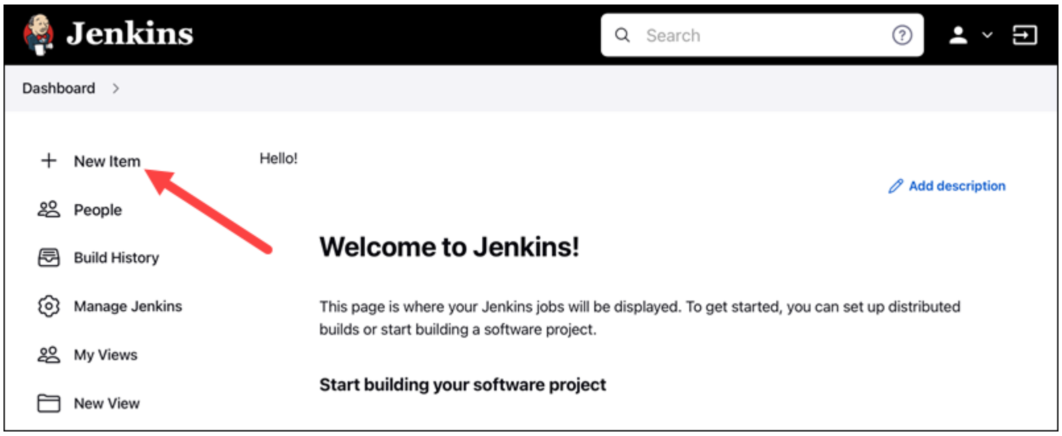
Jenkins is an open-source server used for automating software development and deployment. It allows developers to create repeatable jobs that contain all the code necessary to build an application.
What Is a Build Job?
A Jenkins build job contains the configuration for automating a specific task or step in the application building process. These tasks include gathering dependencies, compiling, archiving, or transforming code, and testing and deploying code in different environments.
Jenkins supports several types of build jobs, such as freestyle projects, pipelines, multi-configuration projects, folders, multibranch pipelines, and organization folders.
What is a Jenkins Freestyle Project?


phoenixNAP
GLOBAL IT SERVICES
Create a New Freestyle Project
Add a Build Step
Build the Project
How to create and run a freestyle project in Jenkins

Jenkins freestyle projects allow users to automate simple jobs, such as running tests, creating and packaging applications, producing reports, or executing commands. Freestyle projects are repeatable and contain both build steps and post-build actions.
Even though freestyle jobs are highly flexible, they support a limited number of general build and post-build actions. Any specialized or non-typical action a user wants to add to a freestyle project requires additional plugins.

How to Set up a Build Job in Jenkins
Follow the steps outlined below to set up and run a new Jenkins freestyle project.

Step 1: Create a New Freestyle Project
1. Click the New Item link on the left-hand side of the Jenkins dashboard

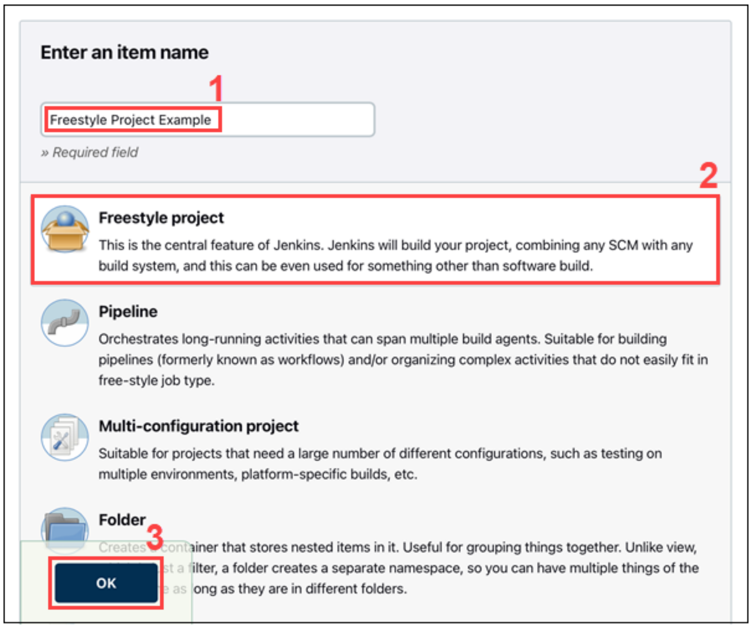


Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

2. Enter the new project's name in the **Enter an item name** field and select the **Freestyle project** type. Click **OK** to continue.


Enter an item name

Freestyle Project Example

» Required field

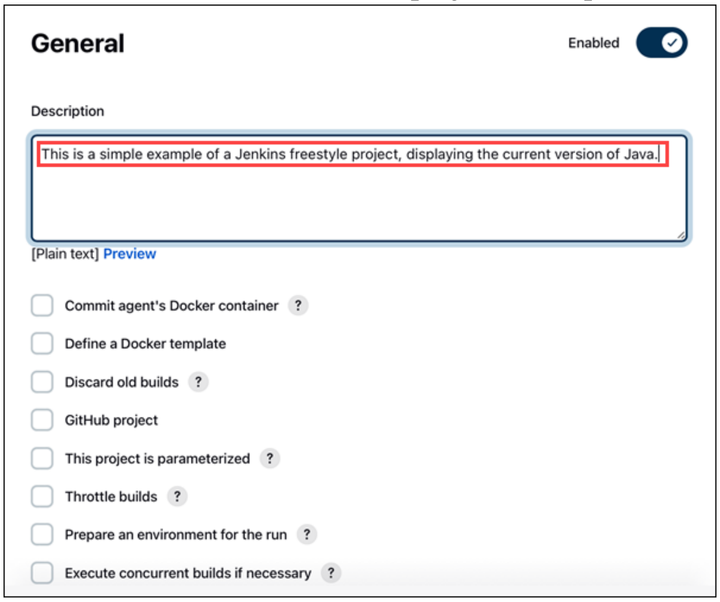
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

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.

3. Under the *General* tab, add a project description in the **Description** field.


General

Enabled

Description

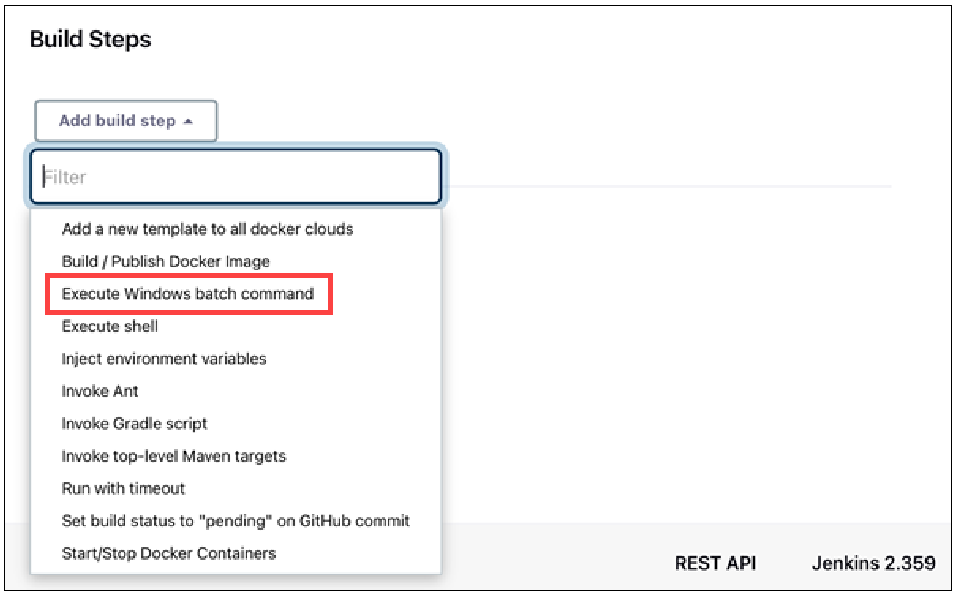
This is a simple example of a Jenkins freestyle project, displaying the current version of Java.

[Plain text] Preview

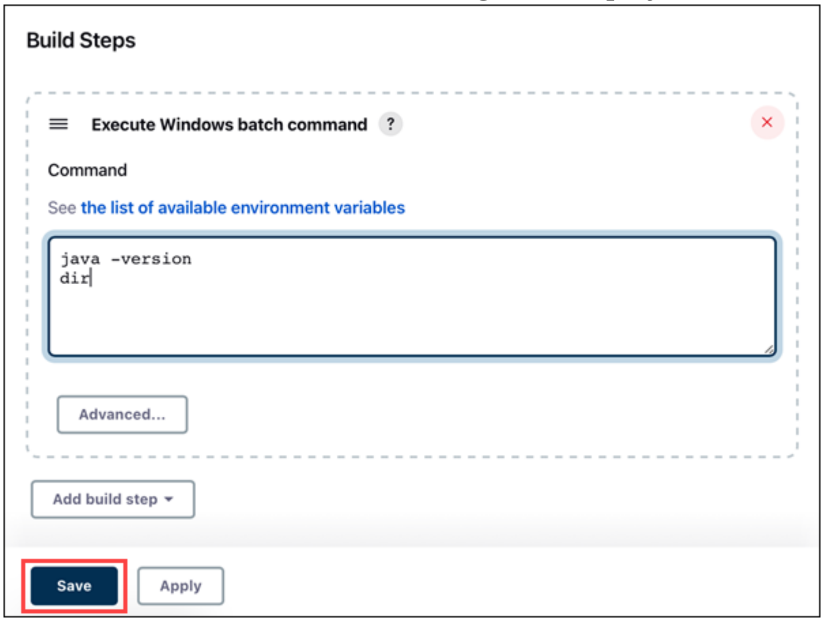
- [ ] Commit agent's Docker container
- [ ] Define a Docker template
- [ ] Discard old builds
- [ ] GitHub project
- [ ] This project is parameterized
- [ ] Throttle builds
- [ ] Prepare an environment for the run
- [ ] Execute concurrent builds if necessary

Step 2: Add a Build Step

1. Scroll down to the *Build* section.
2. Open the **Add build step** drop-down menu and select **Execute Windows batch command**.


Build Steps
Add build step
Filter
Add a new template to all docker clouds
Build / Publish Docker Image
Execute Windows batch command
Execute shell
Inject environment variables
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Run with timeout
Set build status to "pending" on GitHub commit
Start/Stop Docker Containers
REST API
Jenkins 2.359

3. Enter the commands you want to execute in the **Command** field. For this tutorial, we are using a simple set of commands that display the current version of Java and Jenkins working directory:
java -version
dir
4. Click the **Save** button to save changes to the project.

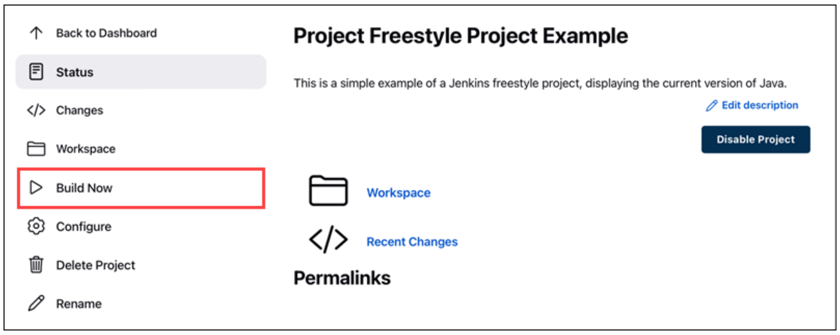

Build Steps
Execute Windows batch command
Command
See the list of available environment variables
Advanced...
Add build step
Save
Apply

```
java -version
dir
```

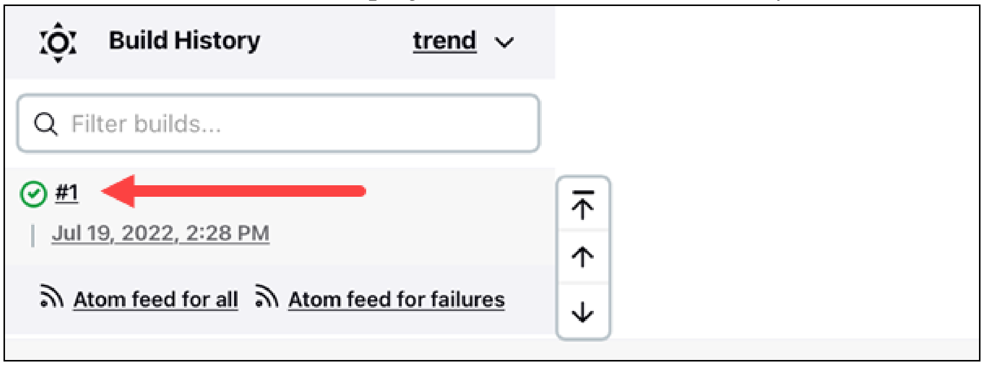
Note: Need a cheap sandboxing environment with automated OS-deployment options? See how easy it is to deploy a development sandbox for as little as $0.10/hour.

Step 3: Build the Project

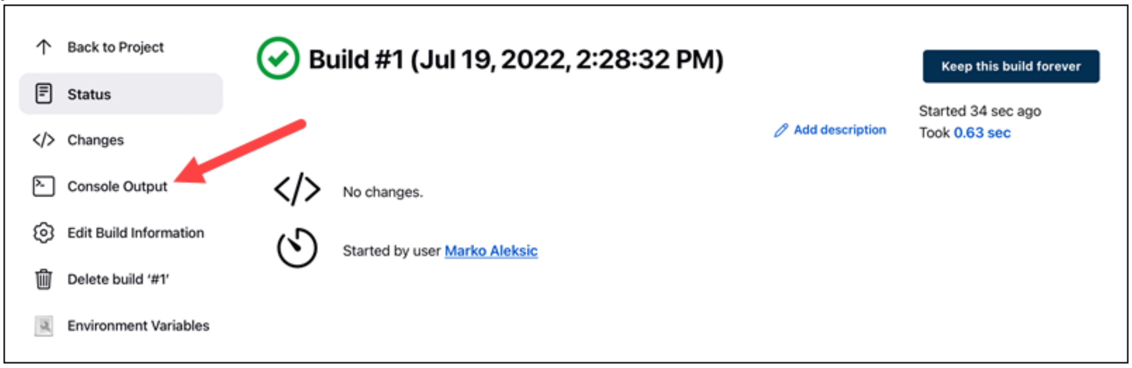
1. Click the **Build Now** link on the left-hand side of the new project page.



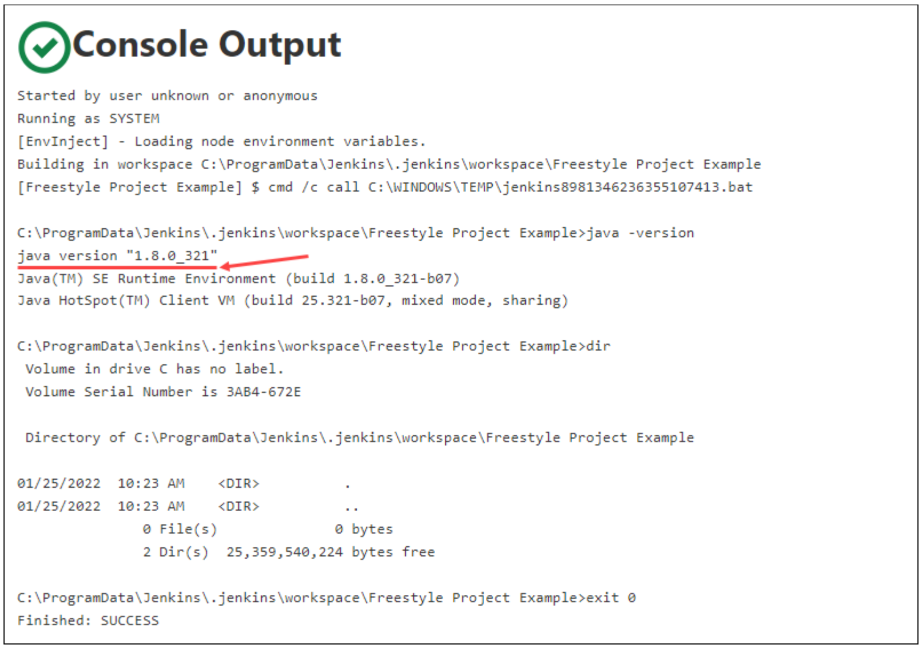
2. Click the link to the latest project build in the *Build History* section.



3. Click the **Console Output** link on the left-hand side to display the output for the commands you entered.



4. The console output indicates that Jenkins is successfully executing the commands, displaying the current version of Java and Jenkins working directory.


Console Output

```
Started by user unknown or anonymous
Running as SYSTEM
[EnvInject] - Loading node environment variables.
Building in workspace C:\ProgramData\Jenkins\.jenkins\workspace\Freestyle Project Example
[Freestyle Project Example] $ cmd /c call C:\WINDOWS\TEMP\jenkins8981346236355107413.bat

C:\ProgramData\Jenkins\.jenkins\workspace\Freestyle Project Example>java -version
java version "1.8.0_321"
Java(TM) SE Runtime Environment (build 1.8.0_321-b07)
Java HotSpot(TM) Client VM (build 25.321-b07, mixed mode, sharing)

C:\ProgramData\Jenkins\.jenkins\workspace\Freestyle Project Example>dir
 Volume in drive C has no label.
 Volume Serial Number is 3AB4-672E

 Directory of C:\ProgramData\Jenkins\.jenkins\workspace\Freestyle Project Example

01/25/2022  10:23 AM    <DIR>          .
01/25/2022  10:23 AM    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)  25,359,540,224 bytes free

C:\ProgramData\Jenkins\.jenkins\workspace\Freestyle Project Example>exit 0
Finished: SUCCESS
```

CONCLUSION:

Students should be able to create and run your first Jenkins freestyle project.

Experiment 6: Creating Build Using Jenkins
Introduction
What is Build Tools?
CI/CD Pipeline
Jenkins Pipeline
Jenkinsfile Details
Pipelines Syntax
Declarative versus Scripted Pipeline syntax
Steps to create and build pipepline using jenkinsfile
Steps to run java project
Testing in Jenkins
Conclusion

9. Lab Exercise

Exercise No 7: (2 Hours) – 1 Practical

Aim: - Building Images using DockerFile

Objective:
1. Understand the Docker Image and Dockerfile

THEORY:

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code, you can significantly reduce the delay between writing code and running it in production.

The Docker platform

Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security lets you run many containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application, so you don't need to rely on what's installed on the host. You can share containers while you work, and be sure that everyone you share with gets the same container that works in the same way.

Docker provides tooling and a platform to manage the lifecycle of your containers:

Develop your application and its supporting components using containers.

The container becomes the unit for distributing and testing your application.

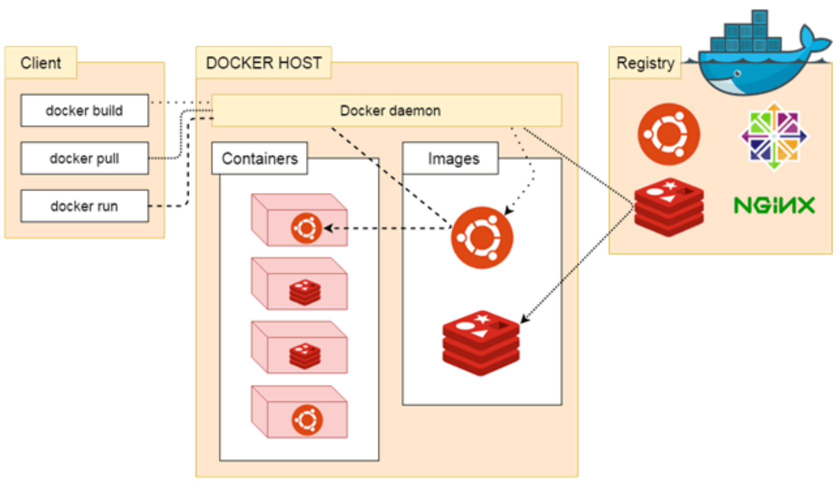
When you're ready, deploy your application into your production environment, as a container or an orchestrated service. This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two.

What is Docker daemon?

Docker daemon runs on the host operating system. It is responsible for running containers to manage docker services. Docker daemon communicates with other daemons. It offers various Docker objects such as images, containers, networking, and storage.

Docker architecture

Docker uses a client-server architecture. The Docker client talks to the Docker daemon,which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface. Another Docker client is Docker Compose, that lets you work with applications consisting of a set of containers.



How to install docker on Windows

We can install docker on any operating system like Windows, Linux, or Mac. Here, we are going to install docker-engine on Windows. The main advantage of using Docker on Windows is that it provides an ability to run natively on Windows without any kind of virtualization. To install docker on windows, we need to download and install the Docker Toolbox.

Follow the below steps to install docker on windows -

Step 1: Click on the below link to download DockerToolbox.exe. https://download.docker.com/win/stable/DockerToolbox.exe

Step 2: Once the DockerToolbox.exe file is downloaded, double click on that file. The following window appears on the screen, in which click on the Next.

Step 3: Browse the location where you want to install the Docker Toolbox and click on the Next.

Step 4: Select the components according to your requirement and click on the Next.

Step 5: Select Additional Tasks and click on the Next.

Step 6: The Docker Toolbox is ready to install. Click on Install.

Step 7: Once the installation is completed, the Wizard appears on the screen, in which click on the Finish.

Step 8: After the successful installation, three icons will appear on the screen that are: Docker Quickstart Terminal, Kitematic (Alpha), and OracleVM VirtualBox. Double click on the Docker Quickstart Terminal.

Step 9: A Docker Quickstart Terminal window appears on the screen.

To verify that the docker is successfully installed, type the below command and press enter key.

```
docker run hello-world
```

The output will be visible on the screen, otherwise not.

You can check the Docker version using the following command.

```
docker -version
```

Docker Container and Image

Docker container is a running instance of an image. You can use Command Line Interface (CLI) commands to run, start, stop, move, or delete a container. You can also provide configuration for the network and environment variables. Docker container is an isolated and secure application platform, but it can share and access to resources running in a different host or container.

An image is a read-only template with instructions for creating a Docker container. A docker image is described in text file called a Dockerfile, which has a simple, well-defined syntax. An image does not have states and never changes. Docker Engine provides the core Docker technology that enables images and containers.

You can understand container and image with the help of the following command.

```
$ docker run hello-world
```

The above command docker run hello-world has three parts.

1) docker: It is docker engine and used to run docker program. It tells to the operating system that you are running docker program.

2) run: This subcommand is used to create and run a docker container.

3) hello-world: It is a name of an image. You need to specify the name of an image which is to load into the container.

OUTCOME: Install, configure and use Docker as container

CONCLUSION:
Students will be able to build images using DockerFile

Experiment 7: Building Images using Docker File
Introduction
What is Docker?
Docker History
Docker Architecture
Docker Terminolgy
Windows Subsytem for Linux(WSL)
Docker INstallation on windows & Linux
Docker Commands
Docker file & its contents
conclusion

10. Lab Exercise

Exercise No 8: (2 Hours) – 1 Practical

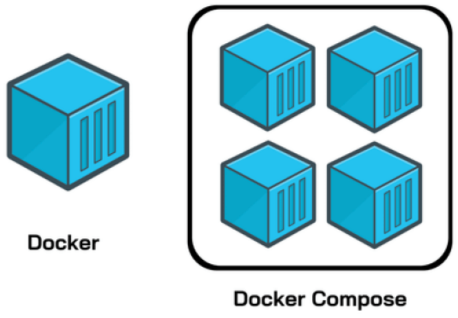
Aim: - Creating multi-containers using Docker Compose

Objective:
1. Understand the Docker and containers

Pre-requisite: know how to build images with Dockerfile

THEORY:
A container is something that packages your code along with any other dependencies so that it can be deployed across multiple platforms reliably.
A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. Containers are built to include everything needed to run an application: code, runtime, system tools, system libraries and settings.
This means a container is fully isolated from another even if they are for the same application with similar dependencies.
These containers can be run locally on your Windows, Mac, and Linux. And major cloud systems like AWS or Azure support them out of the box. You can also use Docker on any hosting space where it can be installed and run.
Docker Compose provides a way to orchestrate multiple containers that work together. Examples include a service that processes requests and a front-end web site, or a service that uses a

supporting function such as a Redis cache. If you are using the microservices model for your app development, you can use Docker Compose to factor the app code into several independently running services that communicate using web requests.
Docker Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services and start them with a single command. The underlying network for your service containers is configured automatically by Docker Compose.



How Does Docker Compose Work?
docker compose is a yaml file in which we can configure different types of services. Then with a single command all containers will be built and fired up.

There are 3 main steps involved in using compose:
Generate a Dockerfile for each project.
Setup services in the docker-compose.yml file.
Fire up the containers.

Installation

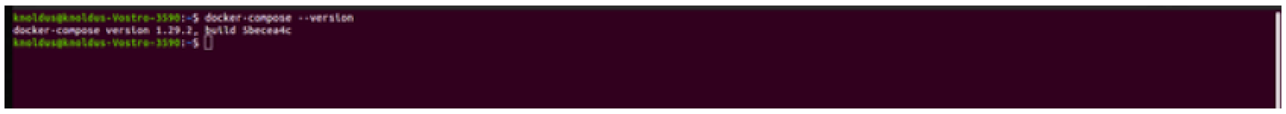
Docker Compose relies on Docker Engine. So before installing it make sure you have Docker Engine installed on your system.
On desktop systems like Docker Desktop for Mac and Windows, Docker Compose is included as part of those desktop installs. You don't need to install it manually.
On Linux systems, you'll need to

1. Install Docker Engine
2. Run the following command to download the current stable release of Docker Compose

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname
-s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

3. Apply executable permissions to the binary:
```
sudo chmod +x /usr/local/bin/docker-compose
```

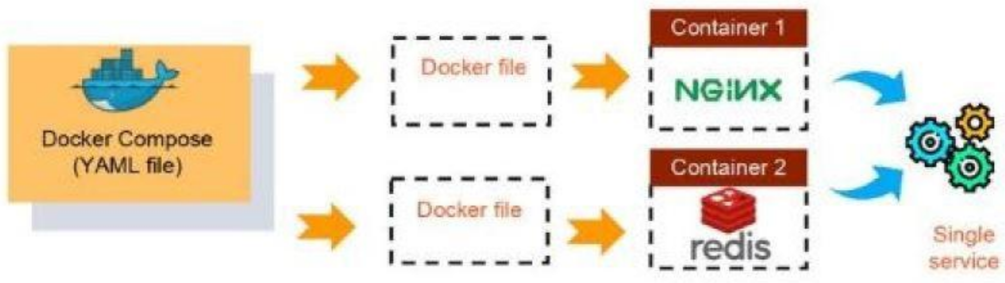
Now test the installation using the following command
```
docker-compose --version
```



```
knoldus@knoldus-Vostro-3590:~$ docker-compose --version
docker-compose version 1.29.2, build 5becea4c
knoldus@knoldus-Vostro-3590:~$ 
```

Creating a docker-compose.yml file to Run a Multi-container Docker Application

Let’s say you have an application that requires an NGINX web server and Redis database, you can create a **docker-compose.yml** file that can run both the containers as a service without the need to start each of them separately.



Now, let’s look at the **docker-compose.yml** file for this purpose:

```
version: '3'

services:

  web:

    image: nginx

    ports:

    - 8080:80



  database:

    image: redis
```

Let’s understand the meaning of each of the keywords used here:

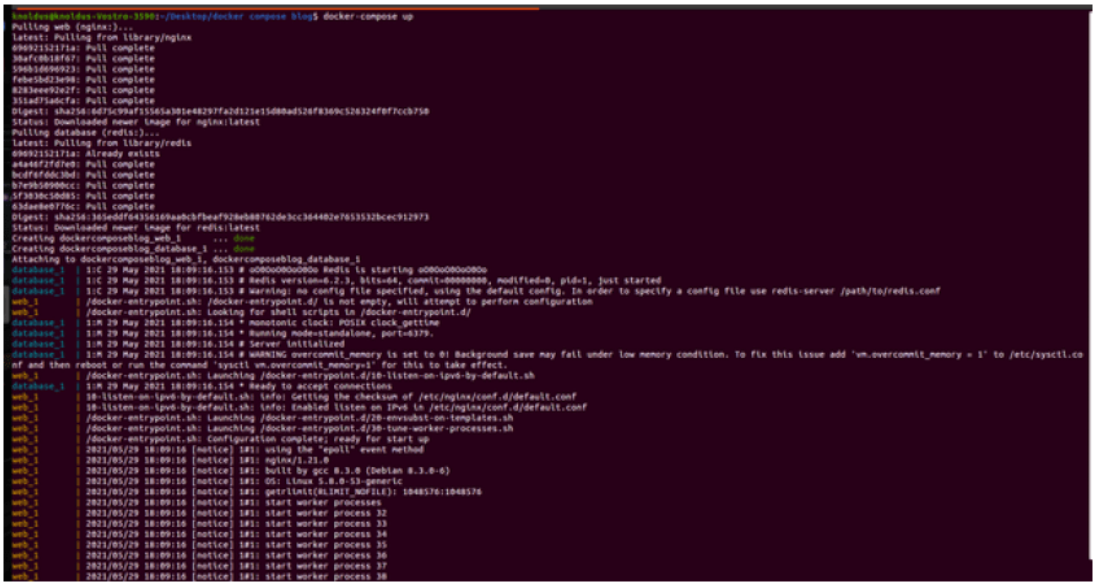
- **version ‘3’:** This denotes that we are using version 3 of the Compose file.
- **services:** This section defines all the different containers we will create. In the above example, we have two services, web and database.
- **web and database:** These are the names of our services. Containers will be created with the names we provide.
- **ports:** This is used to map the container’s ports to the host machine.
- **image:** To run a service using a pre-built image, we specify the image location using the image clause.

Building our Application

Now we have our **docker-compose.yml** file constructed, so it's time to build our application. As this is a very simple application, it will basically deploy two containers — the web server and the database.

So now, to build the stack, go back to your terminal window and run the following command:

docker-compose up



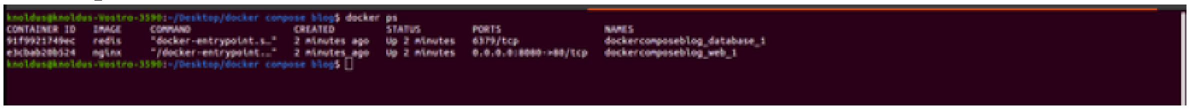
```
knoldus@knoldus-Vostro-3590:~/Desktop/docker compose blog$ docker-compose up
Pulling web (nginx:)...
latest: Pulling from library/nginx
69692152171a: Pull complete
30afc0b18f67: Pull complete
596b1d696923: Pull complete
febe5bd23e98: Pull complete
8283eee92e2f: Pull complete
351ad75a6cfa: Pull complete
Digest: sha256:6d75c99af15565a301e48297fa2d121e15d80ad526f8369c526324f0f7ccb750
Status: Downloaded newer image for nginx:latest
Pulling database (redis:)...
latest: Pulling from library/redis
69692152171a: Already exists
a4a46f2fd7e0: Pull complete
bcdf6fddc3bd: Pull complete
b7e9b50900cc: Pull complete
5f3030c50d85: Pull complete
63dae8e0776c: Pull complete
Digest: sha256:365eddf64356169aa0cbfbeaf928eb88762de3cc364402e7653532bcec912973
Status: Downloaded newer image for redis:latest
Creating dockercomposeblog_web_1      ... done
Creating dockercomposeblog_database_1 ... done
Attaching to dockercomposeblog_web_1, dockercomposeblog_database_1
database_1  | 1:C 29 May 2021 18:09:16.153 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
database_1  | 1:C 29 May 2021 18:09:16.153 # Redis version=6.2.3, bits=64, commit=00000000, modified=0, pid=1, just started
database_1  | 1:C 29 May 2021 18:09:16.153 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
web_1       | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
web_1       | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
database_1  | 1:M 29 May 2021 18:09:16.154 * monotonic clock: POSIX clock_gettime
database_1  | 1:M 29 May 2021 18:09:16.154 * Running mode=standalone, port=6379.
database_1  | 1:M 29 May 2021 18:09:16.154 # Server initialized
database_1  | 1:M 29 May 2021 18:09:16.154 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.co
nf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
web_1       | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
database_1  | 1:M 29 May 2021 18:09:16.154 * Ready to accept connections
web_1       | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
web_1       | 10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
web_1       | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
web_1       | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
web_1       | /docker-entrypoint.sh: Configuration complete; ready for start up
web_1       | 2021/05/29 18:09:16 [notice] 1#1: using the "epoll" event method
web_1       | 2021/05/29 18:09:16 [notice] 1#1: nginx/1.21.0
web_1       | 2021/05/29 18:09:16 [notice] 1#1: built by gcc 8.3.0 (Debian 8.3.0-6)
web_1       | 2021/05/29 18:09:16 [notice] 1#1: OS: Linux 5.8.0-53-generic
web_1       | 2021/05/29 18:09:16 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
web_1       | 2021/05/29 18:09:16 [notice] 1#1: start worker processes
web_1       | 2021/05/29 18:09:16 [notice] 1#1: start worker process 32
web_1       | 2021/05/29 18:09:16 [notice] 1#1: start worker process 33
web_1       | 2021/05/29 18:09:16 [notice] 1#1: start worker process 34
web_1       | 2021/05/29 18:09:16 [notice] 1#1: start worker process 35
web_1       | 2021/05/29 18:09:16 [notice] 1#1: start worker process 36
web_1       | 2021/05/29 18:09:16 [notice] 1#1: start worker process 37
web_1       | 2021/05/29 18:09:16 [notice] 1#1: start worker process 38
```

The above command will deploy both the web and database containers in attached mode, so you won't get your bash prompt returned. If you want to run them in detached mode, use the below command:

docker-compose up -d

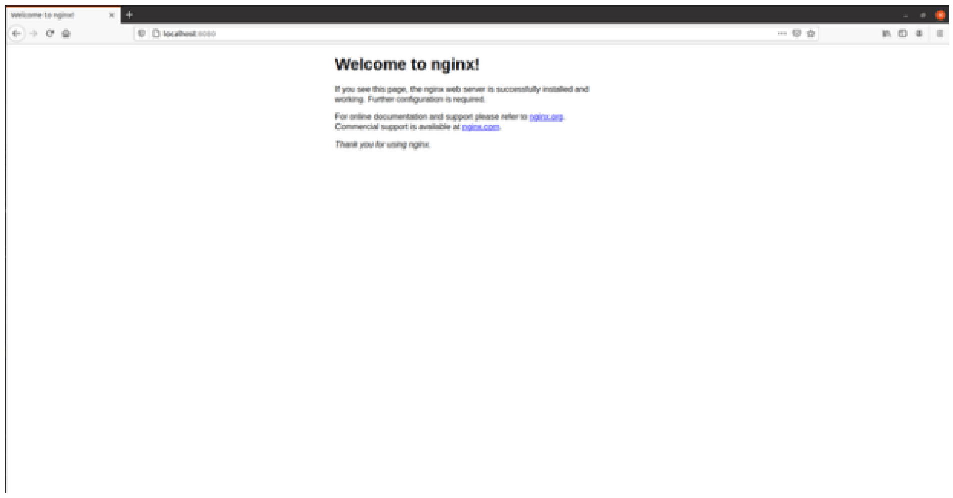
To make sure the containers are running, use the command:

docker ps



```
knoldus@knoldus-Vostro-3590:~/Desktop/docker compose blog$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                  NAMES
91f992174 9ec  redis     "docker-entrypoint.s…"   2 minutes ago   Up 2 minutes   6379/tcp               dockercomposeblog_database_1
e3cbab20b524   nginx     "/docker-entrypoint.…"   2 minutes ago   Up 2 minutes   0.0.0.0:8080->80/tcp   dockercomposeblog_web_1
knoldus@knoldus-Vostro-3590:~/Desktop/docker compose blog$ 
```

Now to see your application up and running, go to your browser and type the following url http://localhost:8080



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

To stop all the running containers, use the following command:
docker-compose down

OUTCOME: Configure Dockerfile for Docker compose

CONCLUSION:
Successfully Deployed container stack using Docker compose

Experiment 8: Creating multi-containers using Docker Compose

Introduction

Virtual Machine vs Container

Docker Hub

Docker Registry

What is Docker Compose?

The benefits of Docker Compose

Steps to deploy multi-container application with Docker Compose

Conclusion

<u>11. Lab Exercise</u>

<u>Exercise No 9: (2 Hours) – 1 Practical</u>

Aim: - Installation of Kubernetes

Objective:

1. Understand the basic concepts of Kubernetes

Pre-requisites: A machine running Windows 10 or 11

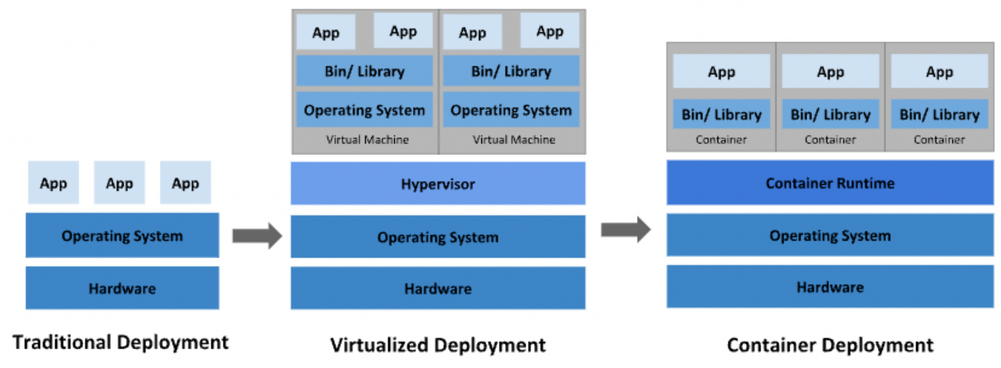
A user account with administrator privileges
A network connection

THEORY:

Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

The name Kubernetes originates from Greek, meaning helmsman or pilot. K8s as an abbreviation results from counting the eight letters between the "K" and the "s". Google open-sourced the Kubernetes project in 2014. Kubernetes combines over 15 years of Google's experience running production workloads at scale with best-of-breed ideas and practices from the community.

Deployment Model



Why you need Kubernetes and what it can do

Containers are a good way to bundle and run your applications. In a production environment, you need to manage the containers that run the applications and ensure that there is no downtime. For example, if a container goes down, another container needs to start. Wouldn't it be easier if this behavior was handled by a system?

That's how Kubernetes comes to the rescue! Kubernetes provides you with a framework to run distributed systems resiliently. It takes care of scaling and failover for your application, provides deployment patterns, and more. For example: Kubernetes can easily manage a canary deployment for your system.

Kubernetes Basics

Cluster

It is a collection of hosts(servers) that helps you to aggregate their available resources. That includes ram, CPU, ram, disk, and their devices into a usable pool.

Master

The master is a collection of components which make up the control panel of Kubernetes. These components are used for all cluster decisions. It includes both scheduling and responding to cluster events.

Node

It is a single host which is capable of running on a physical or virtual machine. A node should run both kube-proxy, minikube, and kubelet which are considered as a part of the cluster.

Namespace

It is a logical cluster or environment. It is a widely used method which is used for scoping access or dividing a cluster.

Kubernetes provides you with:

Service discovery and load balancing Kubernetes can expose a container using the DNS name or using their own IP address. If traffic to a container is high, Kubernetes is able to load balance and distribute the network traffic so that the deployment is stable.

Storage orchestration Kubernetes allows you to automatically mount a storage system of your choice, such as local storages, public cloud providers, and more.

Automated rollouts and rollbacks You can describe the desired state for your deployed containers using Kubernetes, and it can change the actual state to the desired state at a controlled rate. For example, you can automate Kubernetes to create new containers for your deployment, remove existing containers and adopt all their resources to the new container.

Automatic bin packing You provide Kubernetes with a cluster of nodes that it can use to run containerized tasks. You tell Kubernetes how much CPU and memory (RAM) each container needs. Kubernetes can fit containers onto your nodes to make the best use of your resources.

Self-healing Kubernetes restarts containers that fail, replaces containers, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.

Secret and configuration management Kubernetes lets you store and manage sensitive information, such as passwords, OAuth tokens, and SSH keys. You can deploy and update secrets and application configuration without rebuilding your container images, and without exposing secrets in your stack configuration.

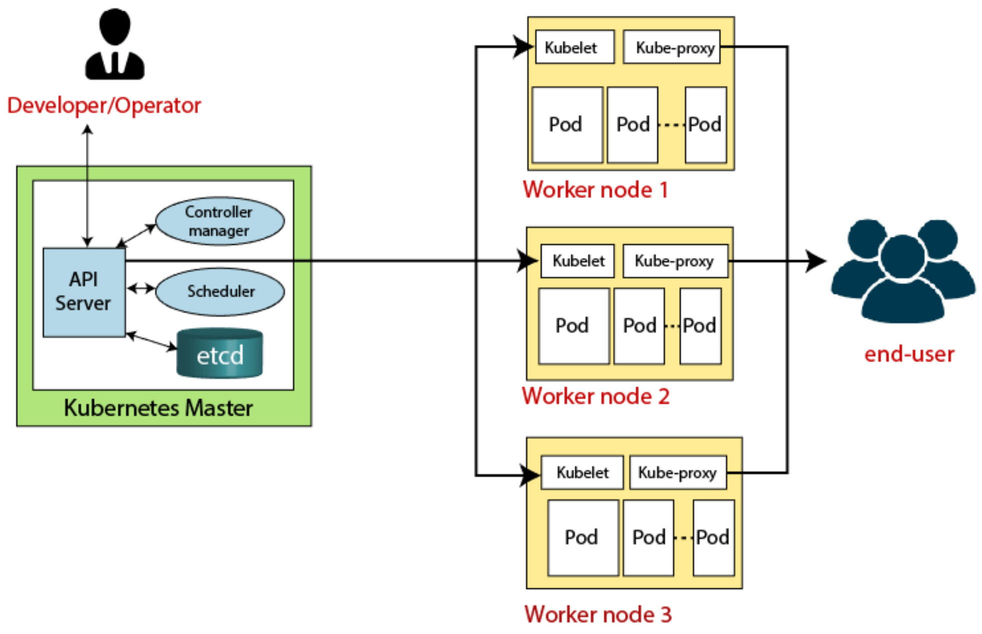
Batch execution In addition to services, Kubernetes can manage your batch and CI workloads, replacing containers that fail, if desired.

Horizontal scaling Scale your application up and down with a simple command, with a UI, or automatically based on CPU usage.

IPv4/IPv6 dual-stack Allocation of IPv4 and IPv6 addresses to Pods and Services

Designed for extensibility Add features to your Kubernetes cluster without changing upstream source code.

Kubernetes Architecture



Kubernets Architecture

Installation of Kubernetes on Linux

Since Kubernetes is made for Linux, the only way to run it on Windows is in a virtual machine. Follow the steps below to set up a virtual environment for running Kubernetes.

Step 1: Enable Hyper-V

Hyper-V is Microsoft's hardware virtualization product that allows the creation of virtual machines in their own isolated space. Hyper-V manages VMs via the default GUI or the CLI.

(Note: The requirements for Hyper-V are Windows 10 (Enterprise, Pro, or Education) with at least 4GB of RAM and CPU Virtualization support. If Hyper-V isn't available, check your BIOS settings and ensure virtualization is enabled.)

Follow the steps below to **enable Hyper-V on Windows**:

1. Press the **Windows key** and search for "*Turn Windows features on or off*". Select the first result to load the Windows Features window.
2. Check the boxes for Hyper-V and Windows Hypervisor Platform in the feature list.
3. Click **OK** and wait for the feature installation to finish. When prompted, click **Restart now** to restart the PC and finish setting up Hyper-V.

When the PC boots back up, the Hyper-V feature is enabled.

4. Check if Hyper-V is correctly installed. Open Windows PowerShell as an administrator and run the following command:

```
Get-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V
```

If Hyper-V is correctly installed, the *State* section shows as *Enabled*.

Step 2: Install Docker for Windows

Docker and Kubernetes complement each other. Kubernetes is built on top of Docker and automates tasks such as container creation, deployment, configuration, and resource access.

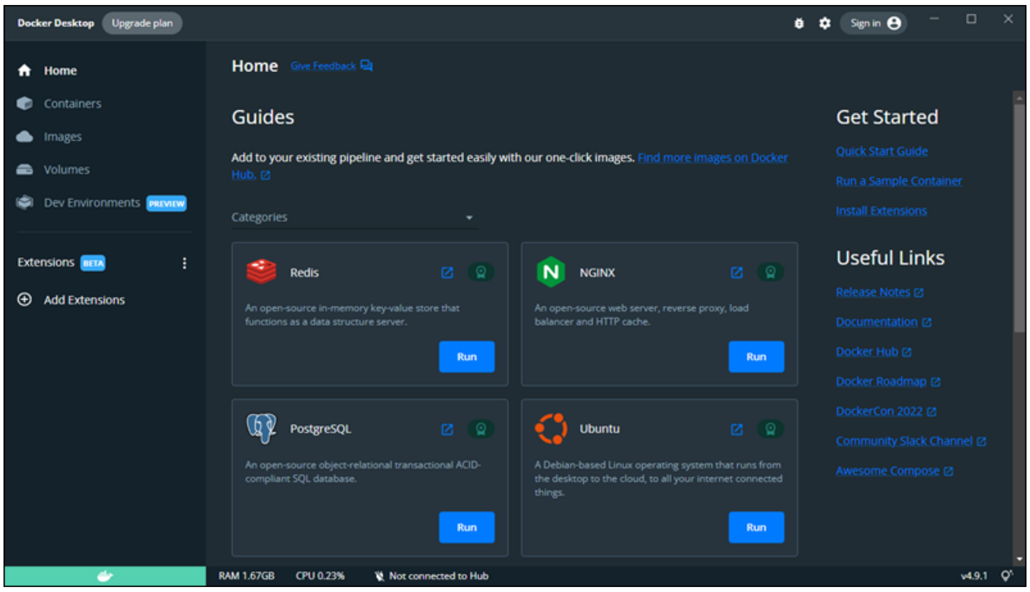
Follow the steps below to install Docker for Windows:

1. Using a web browser, navigate to the Docker Desktop download page and locate the *Download Docker Desktop for Windows* section. Click the download link to download the installation package.
2. Launch the downloaded file to start the Docker installation wizard. The installer prompts you to choose whether to use Hyper-V or WSL 2 and if you want to add a Desktop shortcut. Leave both boxes checked if you have WSL 2 installed and click **Ok**.

Note: Follow our instructions to install WSL 2 on Windows. WSL 2 runs on top of Hyper-V, offering the best performance. It features superior memory management and deeply integrates with the rest of the Windows host.

3. Press the **Close and log out** button to complete the installation.
4. Log back into your user account, review the *Service Agreement*, check the **I accept the terms** box, and click **Accept** to complete the Docker installation.

After accepting the agreement, the Docker GUI tool starts.



Add to your existing pipeline and get started easily with our one-click images. Find more images on Docker Hub.

An open-source in-memory key-value store that functions as a data structure server.

An open-source web server, reverse proxy, load balancer and HTTP cache.

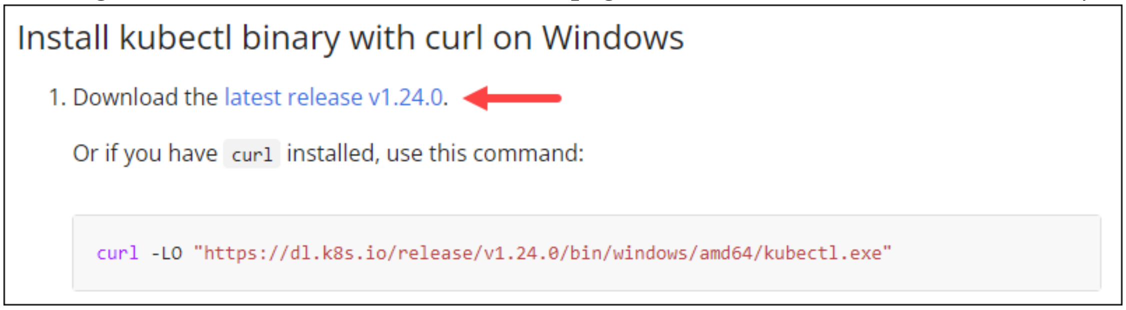
An open-source object-relational transactional ACID-compliant SQL database.

A Debian-based Linux operating system that runs from the desktop to the cloud, to all your internet connected things.

Step 3: Install Kubernetes
Docker comes with a GUI tool that allows users to configure Docker settings and install and enable Kubernetes.
There are **several methods for installing Kubernetes**. This article will cover installing Kubernetes via the Docker settings, Minikube, and Kind. Depending on your machine's specifications, choose the method that suits your system:

- **Minikube** requires at least 2GB of RAM and 2 CPUs.
- **Kind** requires 8GB of RAM to deliver good performance.
- Installing Kubernetes via **Docker settings** takes up to 8 GB of RAM.

Before installing Kubernetes, install kubectl, the Kubernetes CLI tool. This utility lets you run commands against Kubernetes clusters.
Follow these steps to install kubectl:
1. Navigate to the official kubectl download page and locate the *Install kubectl binary* section:


Install kubectl binary with curl on Windows

1. Download the latest release v1.24.0.

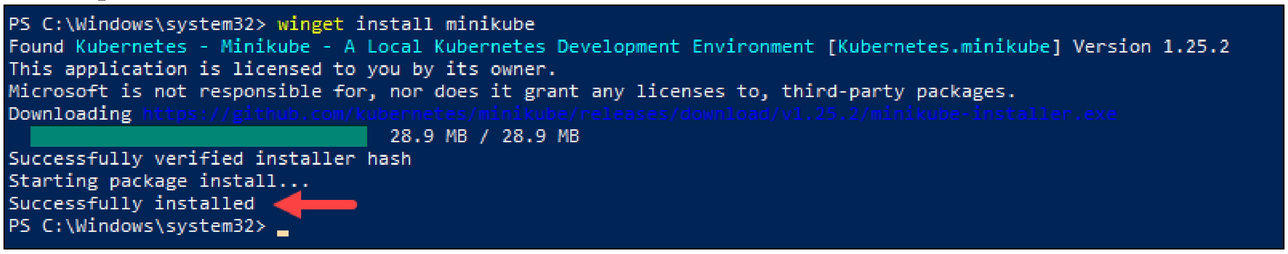
   Or if you have `curl` installed, use this command:

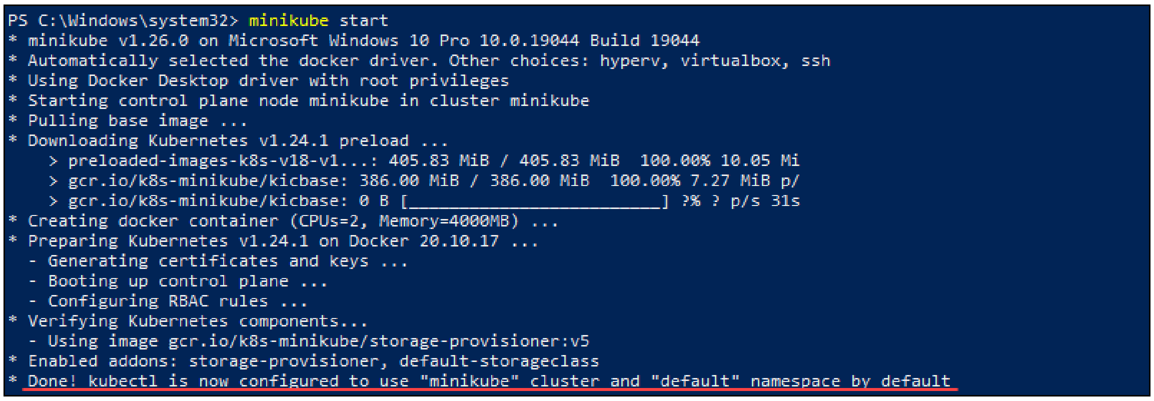
```
curl -LO "https://dl.k8s.io/release/v1.24.0/bin/windows/amd64/kubectl.exe"
```

2. Click the download link for the latest release. At the time of writing this article, the latest release was 1.24.0. Save the file to a directory such as *C:\kubectl*.
3. Press the **Windows** button and search for *Environment variables*. Select **Edit the system environment variables**.
4. In the System Properties window, click **Environment Variables…**

5. Under the *System Variables* section, click the **Path** environment variable and select **Edit** to add the kubectl system variable:
6. Click **New** and add the path to the downloaded kubectl binary file. Select **OK** in all windows to confirm the changes.
7. Check if everything is set up correctly by running **kubectl** in Windows PowerShell:

Via Docker GUI
The easiest way to install Kubernetes is by enabling it in Docker settings. Follow the steps below to do so:
1. In the system tray, right-click the **Docker icon**. Select **Settings** from the menu.

Important: If the Docker icon is missing from the system tray, reboot your system. If the problem persists, check the official troubleshooting guide.
2. In Docker settings, select the *Kubernetes* tab. Check the **Enable Kubernetes** box and click **Apply & Restart**.
3. When prompted, click **Install** to proceed.
4. The tool downloads the necessary cluster components and creates another VM in the background. When the installation finishes, both the Docker and Kubernetes icons are green, which means they are up and running:
Via Minikube
Minikube is an open-source tool for running Kubernetes. It works with Linux, Mac, and Windows by running a single-node cluster inside a virtual machine on the local machine. Follow the steps below to install Kubernetes via Minikube:
Install Using winget:
1. If you are using winget, the Windows package manager, install Minikube by running:
winget install minikube
The output shows when the installation finishes.

```
PS C:\Windows\system32> winget install minikube
Found Kubernetes - Minikube - A Local Kubernetes Development Environment [Kubernetes.minikube] Version 1.25.2
This application is licensed to you by its owner.
Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.
Downloading https://github.com/kubernetes/minikube/releases/download/v1.25.2/minikube-installer.exe
                              28.9 MB / 28.9 MB
Successfully verified installer hash
Starting package install...
Successfully installed
PS C:\Windows\system32>
```
Install Using the Installer Wizard
1. Download the latest Minikube release and start the executable installer.
2. On the Welcome screen, click **Next** to proceed.
3. Read the License Agreement and click **I Agree** to proceed to the next step.
4. Choose a custom install location for Minikube or leave the default one and click **Install**.
5. After the installation completes, click **Next** and **Finish** to exit the installer.
After installation, start the Minikube cluster by running the following command:
minikube start
The output shows **Done!** once the installation finishes.



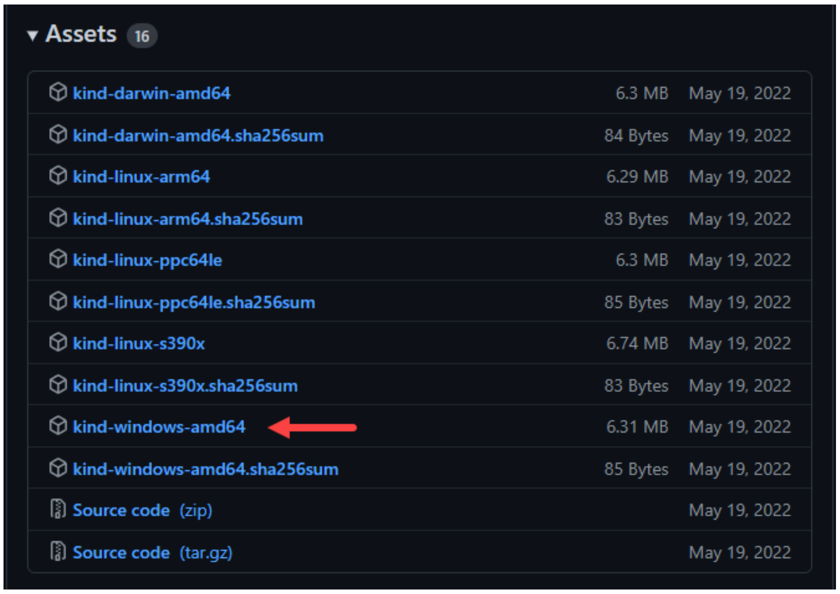
```
PS C:\Windows\system32> minikube start
* minikube v1.26.0 on Microsoft Windows 10 Pro 10.0.19044 Build 19044
* Automatically selected the docker driver. Other choices: hyperv, virtualbox, ssh
* Using Docker Desktop driver with root privileges
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Downloading Kubernetes v1.24.1 preload ...
    > preloaded-images-k8s-v18-v1...: 405.83 MiB / 405.83 MiB  100.00% 10.05 Mi
    > gcr.io/k8s-minikube/kicbase: 386.00 MiB / 386.00 MiB  100.00% 7.27 MiB p/
    > gcr.io/k8s-minikube/kicbase: 0 B [_________________________] ?% ? p/s 31s
* Creating docker container (CPUs=2, Memory=4000MB) ...
* Preparing Kubernetes v1.24.1 on Docker 20.10.17 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Via Kind (Kubernetes in Docker)

Kind is short for Kubernetes in Docker, which means having Docker installed on your machine is a prerequisite for Kind. Kind works by running Kubernetes as a group of Docker containers without creating a VM. Thus, Kind has a faster startup time compared to Minikube.

Follow the steps below to install Kind:

1. Using a web browser, navigate to the official Kind releases page.
2. Scroll down to the *Assets* section and click the download link for the Windows version.

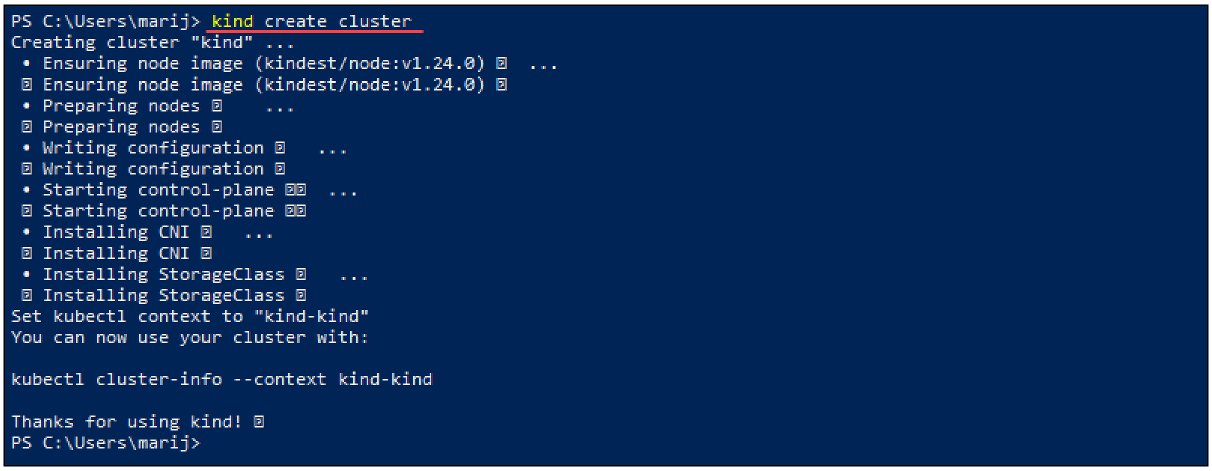


▾ Assets 16

kind-darwin-amd64	6.3 MB	May 19, 2022
kind-darwin-amd64.sha256sum	84 Bytes	May 19, 2022
kind-linux-arm64	6.29 MB	May 19, 2022
kind-linux-arm64.sha256sum	83 Bytes	May 19, 2022
kind-linux-ppc64le	6.3 MB	May 19, 2022
kind-linux-ppc64le.sha256sum	85 Bytes	May 19, 2022
kind-linux-s390x	6.74 MB	May 19, 2022
kind-linux-s390x.sha256sum	83 Bytes	May 19, 2022
kind-windows-amd64	6.31 MB	May 19, 2022
kind-windows-amd64.sha256sum	85 Bytes	May 19, 2022
Source code (zip)		May 19, 2022
Source code (tar.gz)		May 19, 2022

3. Rename the downloaded file to *kind.exe*.
4. Move the downloaded file to a directory such as *C:\kind*, and add the path to the system environment variables. Follow the same steps listed above for adding the *kubectl* environment variable path.
4. Create a cluster with Kind by running the following command in Windows PowerShell:

kind create cluster



```
PS C:\Users\marij> kind create cluster
Creating cluster "kind" ...
 • Ensuring node image (kindest/node:v1.24.0) ⍰  ...
 ⍰ Ensuring node image (kindest/node:v1.24.0) ⍰
 • Preparing nodes ⍰    ...
 ⍰ Preparing nodes ⍰
 • Writing configuration ⍰   ...
 ⍰ Writing configuration ⍰
 • Starting control-plane ⍰⍰  ...
 ⍰ Starting control-plane ⍰⍰
 • Installing CNI ⍰   ...
 ⍰ Installing CNI ⍰
 • Installing StorageClass ⍰   ...
 ⍰ Installing StorageClass ⍰
Set kubectl context to "kind-kind"
You can now use your cluster with:

kubectl cluster-info --context kind-kind

Thanks for using kind! ⍰
PS C:\Users\marij>
```

Step 4: Install Kubernetes Dashboard

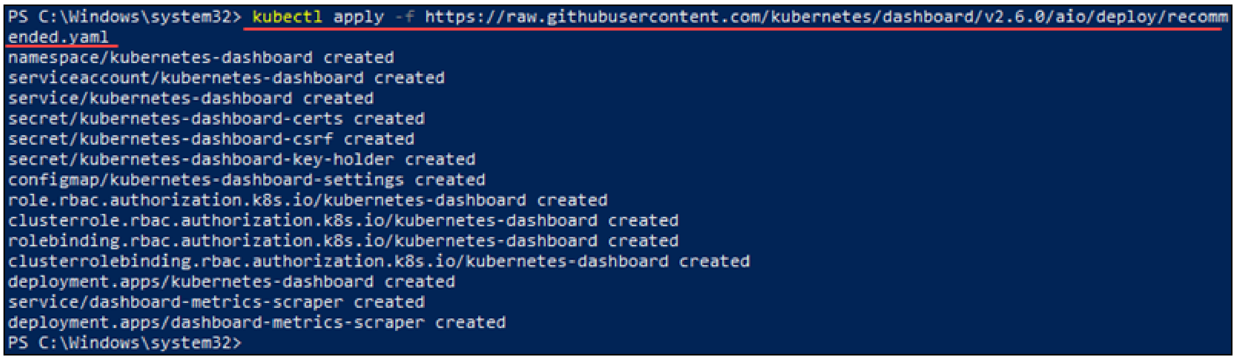
The Kubernetes Dashboard is the official web-based UI for managing Kubernetes resources, and it is an alternative to the kubectl CLI tool. However, it needs to be set up manually as it doesn't automatically deploy.

Use the Dashboard to deploy containerized applications to a Kubernetes cluster or manage the cluster resources. You can view a summary of applications running on the cluster and use the options to create or modify individual Kubernetes resources.

Follow the steps below to **install the Kubernetes Dashboard**:

1. Open Windows PowerShell as an administrator and run the following command to deploy the Kubernetes Dashboard:

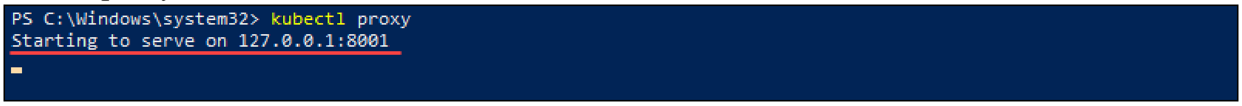
kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.6.0/aio/deploy/recommended.yaml



```
PS C:\Windows\system32> kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.6.0/aio/deploy/recomm
ended.yaml
namespace/kubernetes-dashboard created
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created
PS C:\Windows\system32>
```

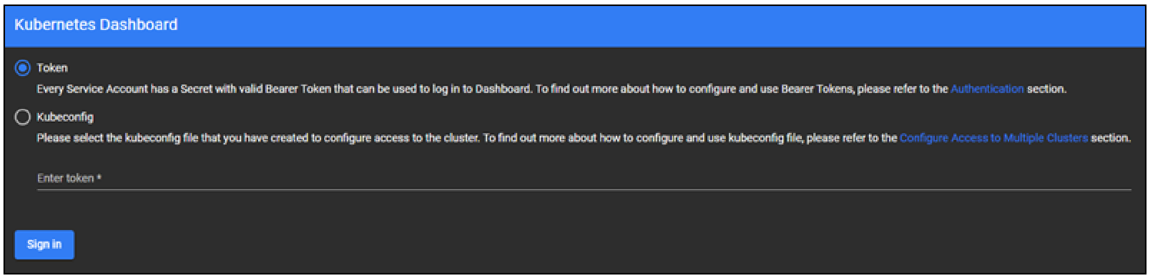
2. Create a secure channel for accessing the Dashboard by running:

kubectl proxy



```
PS C:\Windows\system32> kubectl proxy
Starting to serve on 127.0.0.1:8001
```

3. Access the Dashboard Login page at **http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/**



Kubernetes Dashboard

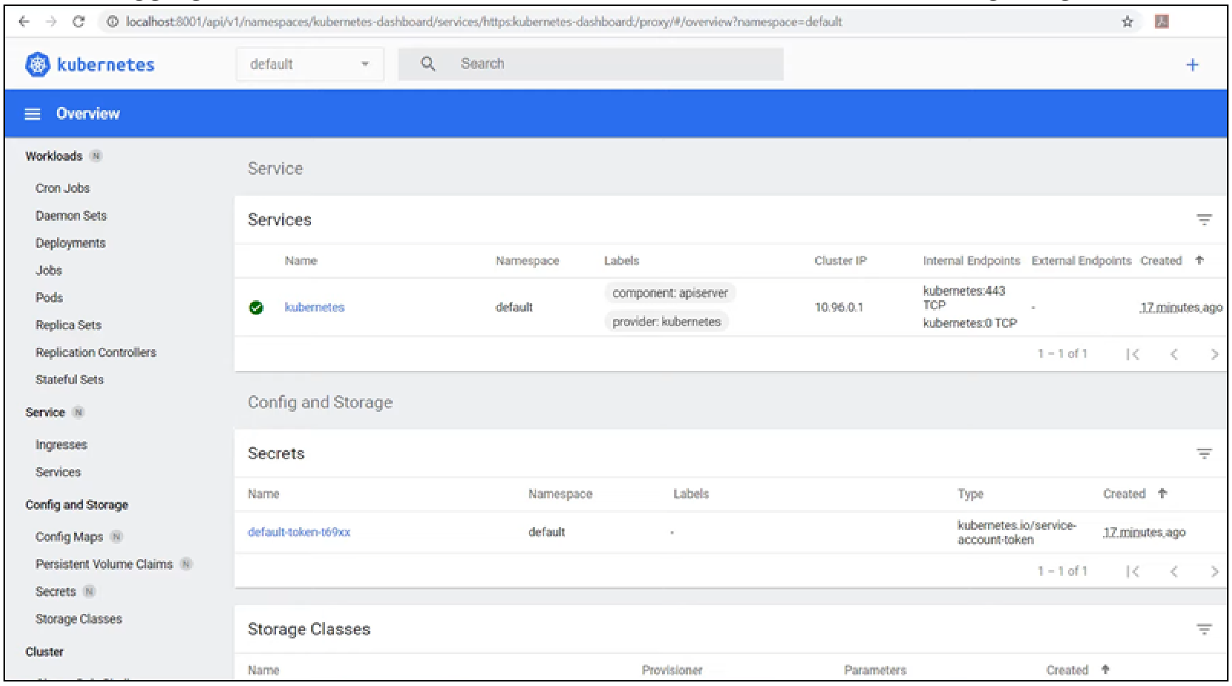
[✓ Token]
Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the Authentication section.

Kubeconfig
Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the Configure Access to Multiple Clusters section.

Enter token *

Sign in

Note: To be able to log in, you need to create a user and have a token. Follow the official guide for creating a sample user and bearer token.
4. After logging in, the Dashboard shows an overview similar to the following image:



localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/#/overview?namespace=default

kubernetes — default — Search

Overview

Workloads: Cron Jobs, Daemon Sets, Deployments, Jobs, Pods, Replica Sets, Replication Controllers, Stateful Sets
Service: Ingresses, Services
Config and Storage: Config Maps, Persistent Volume Claims, Secrets, Storage Classes
Cluster

Service

Services

Name	Namespace	Labels	Cluster IP	Internal Endpoints	External Endpoints	Created
kubernetes	default	component: apiserver provider: kubernetes	10.96.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	17 minutes ago

1 – 1 of 1

Config and Storage

Secrets

Name	Namespace	Labels	Type	Created
default-token-t69xx	default	-	kubernetes.io/service-account-token	17 minutes ago

1 – 1 of 1

Storage Classes

Name	Provisioner	Parameters	Created

Installation of Kubernetes on Linux

The installation of Kubernetes on Linux is a straight forward process. Follow the below steps to install the Kubernetes. In the installation of Kubernetes, each step is mandatory.

Step 1: In this step, we have to update the necessary dependencies of a system using two commands.

The first command is used to get all the updates. Execute the following command in the terminal; it will ask to enter the system's password.

```
sudo apt-get update
```

When the first command is successfully executed, type the following second command, which is used to make the repositories.

```
sudo apt-get install -y apt-transport-https
```

Step 2: After the above steps are successfully executed, we have to install the dependencies of docker in this step.

Type the following command to install the docker. In the installation process, we have to choose Y for confirmation of the installation.

```
sudo apt install docker.io
```

After installing the docker, we have to type the different two commands for starting and enabling the docker. Type the following first command, which starts the docker:

```
sudo systemctl start docker
```

Now, type the following second command, which enables the docker:

```
sudo systemctl enable docker
```

Now, we can check the version of docker by typing the following command:

```
Docker -version
```

Step 3: After the successful execution of all the commands of the second step, we have to install the curl command. The curl is used to send the data using URL syntax.

Now, install the curl by using the following command. In the installation, we have to type Y.

```
sudo apt-get install curl
```

Now, we have to download the add package key for Kubernetes by the following command:

```
sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add
```

If you get an error from the above command, then it means your curl command is not successfully installed, so first install the curl command, and again run the above command.

Now, we have to add the Kubernetes repositories by the following command:

```
sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
```

After the successful execution of the above command, we have to check any updates by executing the following command:

```
sudo apt-get update
```

Step 4: After the execution of the above commands in the above steps, we have to install the components of Kubernetes by executing the following command:

```
sudo apt-get install -y kubelet kubeadm kubectl kubernetes-cni
```

Step 5: After the above installation is done, we have to initialize the kubeadm by executing the following command. The following command disables the swapping on other devices:

```
sudo swapoff -a
```

Now, we have to initialize the kubeadm by executing the following command:

```
sudo kubeadm init
```

Step 6: After the above command is successfully executed, we have to run the following commands, which are given in the initialization of kubeadm. These commands are shown in the above screenshot. The following commands are used to start a cluster:

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Step 7: In this step, we have to deploy the paths using the following command:

```
sudo kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

Step 8: After the execution of the above command, we have to run the following command to verify the installation:

```
sudo kubectl get pods --all-namespaces
```

If the output is displayed as shown in the above screenshot. It means that the Kubernetes is successfully installed on our system.

OUTCOME: Install Kuberenetes

CONCLUSION:

Students will be able to install Kubernetes

Experiment 9: Installation of Kubernetes
Introduction
What is Kubernetes?
Orchestration & its benefits
Kubernetes Components
    Diagram
    Nodes
    Pods
    Control Plane
    kube-apiserver
    etcd
    kube-scheduler
Installation of Kubernetes
What is Kubetctl?
Kubetctl commands
Conclusion

12. Lab Exercise

Exercise No 10: (2 Hours) – 1 Practical

Aim: - Study and use of Kubernetes Services

Objective:

1. Understand the basics of Kubernetes services

Pre-requisites: understating of how the Docker works, how the Docker images are created

THEORY:

Kubernetes is a container management technology developed in Google lab to manage containerized applications in different kind of environments such as physical, virtual, and cloud infrastructure. It is an open source system which helps in creating and managing containerization of application. This tutorial provides an overview of different kind of features and functionalities of Kubernetes and teaches how to manage the containerized infrastructure and application deployment.

Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

The name Kubernetes originates from Greek, meaning helmsman or pilot. K8s as an abbreviation results from counting the eight letters between the "K" and the "s". Google open-sourced the Kubernetes project in 2014. Kubernetes combines over 15 years of Google's experience running production workloads at scale with best-of-breed ideas and practices from the community.

What can Kubernetes do for you?

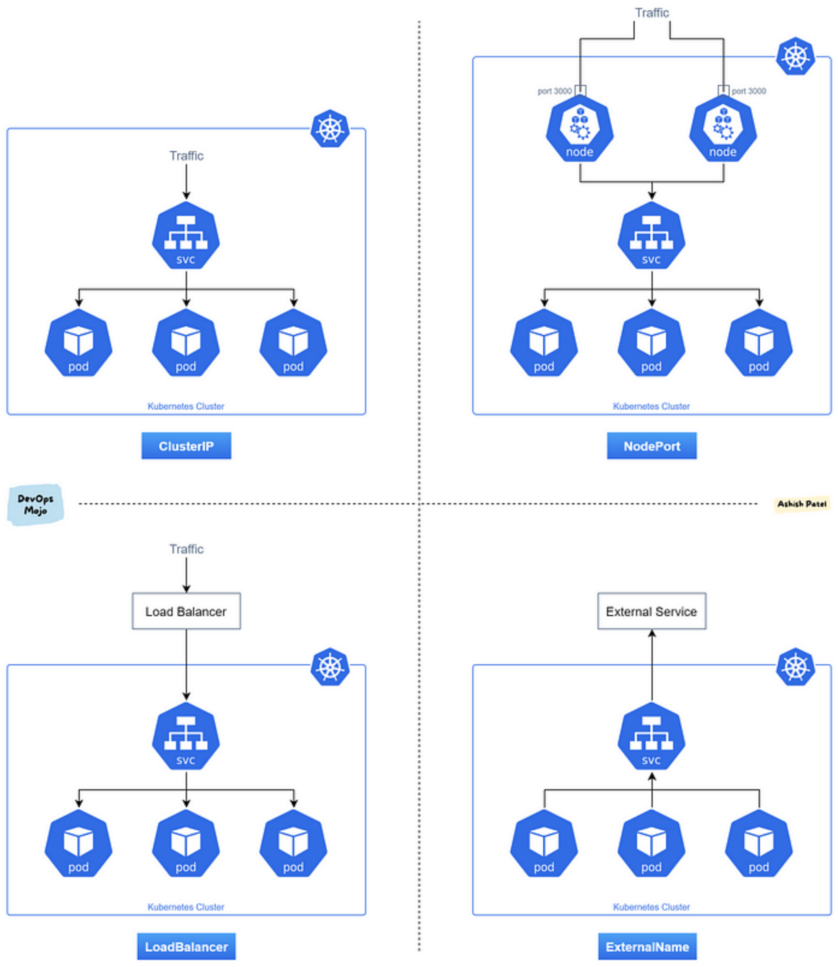
With modern web services, users expect applications to be available 24/7, and developers expect to deploy new versions of those applications several times a day. Containerization helps package software to serve these goals, enabling applications to be released and updated without downtime. Kubernetes helps you make sure those containerized applications run where and when you want, and helps them find the resources and tools they need to work. Kubernetes is a production-ready, open source platform designed with Google's accumulated experience in container orchestration, combined with best-of-breed ideas from the community.

Features of Kubernetes

Following are some of the important features of Kubernetes.
Continues development, integration and deployment
Containerized infrastructure
Application-centric management
Auto-scalable infrastructure
Environment consistency across development testing and production
Loosely coupled infrastructure, where each component can act as a separate unit
Higher density of resource utilization
Predictable infrastructure which is going to be created

Types of Kubernetes services

There are four types of Kubernetes services:



ClusterIP: ClusterIP is the default service that enables the communication of multiple pods within the cluster. By default, your service will be exposed on a ClusterIP if you don't manually define it. ClusterIP can't be accessed from the outside world. But, a Kubernetes proxy can be used to access your services. This service type is used for internal networking between your workloads, while debugging your services, displaying internal dashboards, etc.

NodePort: A NodePort is the simplest networking type of all. It requires no configuration, and it simply routes traffic on a random port on the host to a random port on the container. This is suitable for most cases, but it does have some disadvantages:
You may need to use a reverse proxy (like Nginx) to ensure that web requests are routed correctly.
You can only expose one single service per port.
Container IPs will be different each time the pod starts, making DNS resolution impossible.
The container cannot access localhost from outside of the pod, as there is no IP configured.
Nevertheless, you can use NodePort during experimentation and for temporary use cases, such as demos, POCs, and internal training to show how traffic routing works. It is recommended **not** to use NodePort in production to expose services.

LoadBalancer: LoadBalancer is the most commonly used service type for Kubernetes networking. It is a standard load balancer service that runs on each pod and establishes a connection to the outside world, either to networks like the Internet, or within your datacenter.
The LoadBalancer will keep connections open to pods that are up, and close connections to those that are down. This is similar to what you have on AWS with ELBs, or Azure with Application Gateway. Upstreams provide Layer 4 routing for HTTP(S) traffic, whereas Downstreams provide Layer 7 routing for HTTP(S) traffic.
You can route traffic on destination port number, protocol and hostname, or use application labels. You can send almost any kind of traffic to this service type, such as HTTP, TCP, UDP, Grpc, and more. Use this approach to expose your services directly.

Use a Service to Access an Application in a Cluster

This page shows how to create a Kubernetes Service object that external clients can use to access an application running in a cluster. The Service provides load balancing for an application that has two running instances.

Before you begin

You need to have a Kubernetes cluster, and the kubectl command-line tool must be configured to communicate with your cluster. It is recommended to run this tutorial on a cluster with at least two nodes that are not acting as control plane hosts. If you do not already have a cluster, you can create one by using minikube or you can use one of these Kubernetes playgrounds:

Killercoda

Play with Kubernetes

Creating a service for an application running in two pods

Here is the configuration file for the application Deployment:

service/access/hello-application.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-world
spec:
  selector:
    matchLabels:
      run: load-balancer-example
  replicas: 2
```

```
 template:
   metadata:
     labels:
       run: load-balancer-example
   spec:
     containers:
       - name: hello-world
         image: gcr.io/google-samples/node-hello:1.0
         ports:
           - containerPort: 8080
             protocol: TCP
```

Run a Hello World application in your cluster: Create the application Deployment using the file above:

```
kubectl apply -f https://k8s.io/examples/service/access/hello-application.yaml
```

The preceding command creates a Deployment and an associated ReplicaSet. The ReplicaSet has two Pods each of which runs the Hello World application.

Display information about the Deployment:

```
kubectl get deployments hello-world
kubectl describe deployments hello-world
```

Display information about your ReplicaSet objects:

```
kubectl get replicasets
kubectl describe replicasets
```

Create a Service object that exposes the deployment:

```
kubectl expose deployment hello-world --type=NodePort --name=example-service
```

Display information about the Service:

```
kubectl describe services example-service
```

The output is similar to this:

```
Name:               example-service
Namespace:          default
Labels:             run=load-balancer-example
Annotations:        <none>
Selector:           run=load-balancer-example
Type:               NodePort
IP:                 10.32.0.16
Port:               <unset> 8080/TCP
TargetPort:         8080/TCP
NodePort:           <unset> 31496/TCP
Endpoints:          10.200.1.4:8080,10.200.2.5:8080
Session Affinity:   None
Events:             <none>
```

Make a note of the NodePort value for the service. For example, in the preceding output, the NodePort value is 31496.

List the pods that are running the Hello World application:

```
kubectl get pods --selector="run=load-balancer-example" --output=wide
```

The output is similar to this:

```
NAME                           READY   STATUS    ...  IP           NODE
hello-world-2895499144-bsbk5   1/1     Running   ...  10.200.1.4   worker1
hello-world-2895499144-m1pwt   1/1     Running   ...  10.200.2.5   worker2
```

Get the public IP address of one of your nodes that is running a Hello World pod. How you get this address depends on how you set up your cluster. For example, if you are using Minikube, you can see the node address by running kubectl cluster-info. If you are using Google Compute Engine instances, you can use the gcloud compute instances list command to see the public addresses of your nodes.

On your chosen node, create a firewall rule that allows TCP traffic on your node port. For example, if your Service has a NodePort value of 31568, create a firewall rule that allows TCP traffic on port 31568. Different cloud providers offer different ways of configuring firewall rules.

Use the node address and node port to access the Hello World application:

```
curl http://<public-node-ip>:<node-port>
```

where <public-node-ip> is the public IP address of your node, and <node-port> is the NodePort value for your service. The response to a successful request is a hello message:

```
Hello Kubernetes!
```

Cleaning up

To delete the Service, enter this command:

```
kubectl delete services example-service
```

To delete the Deployment, the ReplicaSet, and the Pods that are running the Hello World application, enter this command:

```
kubectl delete deployment hello-world
```

OUTCOME: Use different service of Kuberenetes

CONCLUSION:

Learned different services of Kubernetes and used the service

Experiment 10: Study and use of Kubernetes services
Introduction
Kubernetes Dashboard
YAML
What are Kubernetes Services?
Components of a Kubernetes services
Types of Kubernetes services
Defining
Selectors & Labels in Kubernetes
Conclusion

13. Lab Exercise

Exercise No 11: (2 Hours) – 1 Practical

Aim: - Installation and Configuration of Ansible

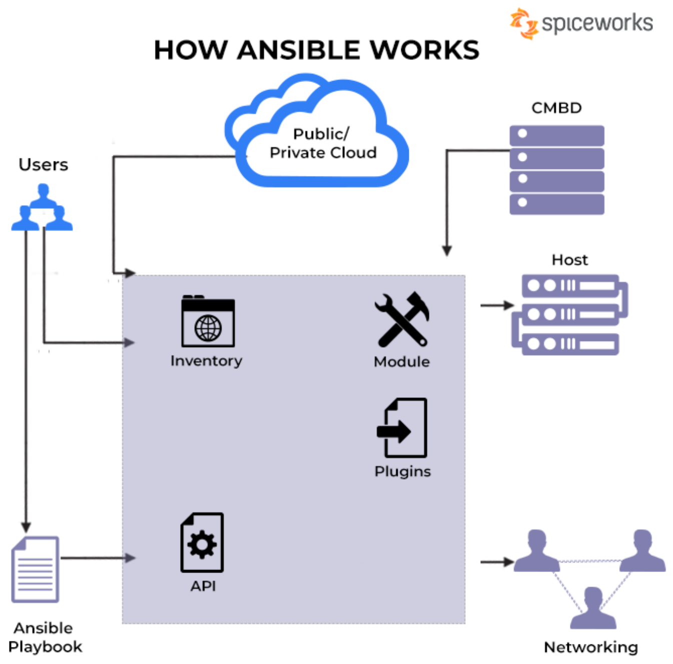
Objective:

1. Understand the installation and configuration process of Ansible

Pre-requisites:

THEORY:

Ansible is an open-source, cross-platform tool for resource provisioning automation that DevOps professionals popularly use for continuous delivery of software code by taking advantage of an "infrastructure as code" approach.



Ansible® is an open source IT automation engine that automates provisioning, configuration management, application deployment, orchestration, and many other IT processes.

Ansible can be used to install software, automate daily tasks, provision infrastructure and network components, improve security and compliance, patch systems, and orchestrate complex workflows.

How does Ansible work?

Modules

Ansible works by connecting to nodes (or hosts) and pushing out small programs—called modules—to these nodes. Nodes are the target endpoints—servers, network devices, or any computer—that you aim to manage with Ansible. Modules are used to accomplish automation tasks in Ansible. These programs are written to be resource models of the desired state of the system. Ansible then executes these modules and removes them when finished.

Without modules, you'd have to rely on ad-hoc commands and scripting to accomplish tasks. Ansible contains built-in modules that you can use to automate tasks, or you can write new ones on your own. Ansible modules can be written in any language that can return JSON, such as Ruby, Python, or bash. Windows automation modules can even be written in Powershell.

Agentless automation

Ansible is agentless, which means the nodes it manages do not require any software to be installed on them. Ansible reads information about which machines you want to manage from your inventory. Ansible has a default inventory file, but you can create your own and define which servers you want to be managed.

Ansible uses SSH protocol to connect to servers and run tasks. By default, Ansible uses SSH keys with ssh-agent and connects to remote machines using your current user name. Root logins are not required. You can log in as any user, and then use su or sudo commands as any user.

Once it has connected, Ansible transfers the modules required by your command or Ansible Playbook to the remote machine(s) for execution. Ansible uses human-readable YAML templates so users can program repetitive tasks to happen automatically without having to learn an advanced programming language.

Using Ansible for ad-hoc commands

You can also use Ansible to run ad-hoc commands, which automate a single task on one or more managed nodes. To do this, you will need to run a command or call a module directly from the command line. No playbook is used, and ad-hoc commands are not reusable. This is fine for a one-time task, but anything more frequent or complex will require the use of an Ansible Playbook.

Ansible Playbooks

Ansible Playbooks are used to orchestrate IT processes. A playbook is a YAML file—which uses a .yml or .yaml extension—containing 1 or more plays, and is used to define the desired state of a system. This differs from an Ansible module, which is a standalone script that can be used inside an Ansible Playbook.

Install Ansible on Windows

There are three ways to run Ansible on Windows 10:

- Cygwin
- Linux Virtual Machine
- Enabling Ubuntu on Windows 10

We shall explain all three methods of installing Ansible on Windows.

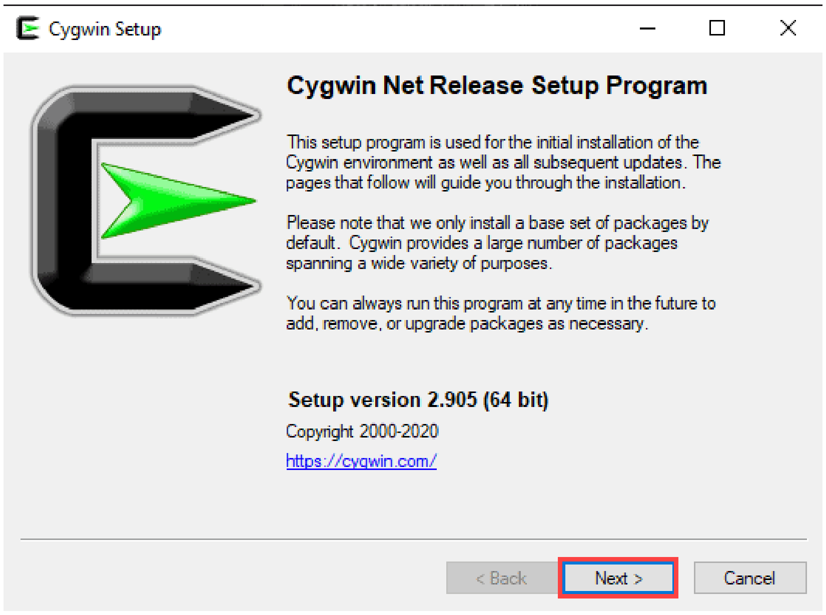
Method 1: Using Cygwin

Cygwin is a POSIX-compatible environment that lets you run tools and code designed for Unix-like operating systems on Microsoft Windows.

Even though the default Cygwin installation contains hundreds of tools for Unix-based systems, Ansible is not one of them. You must manually add Ansible during the installation process.

To install Ansible on Windows using Cygwin, follow these steps:

1. Download the Cygwin installation file. This file is compatible with both the 32-bit and 64-bit versions of Windows 10. It automatically installs the right version for your system.
2. Run the Cygwin installation file. On the starting screen of the installation wizard, click **Next** to continue.



Cygwin Net Release Setup Program

This setup program is used for the initial installation of the Cygwin environment as well as all subsequent updates. The pages that follow will guide you through the installation.

Please note that we only install a base set of packages by default. Cygwin provides a large number of packages spanning a wide variety of purposes.

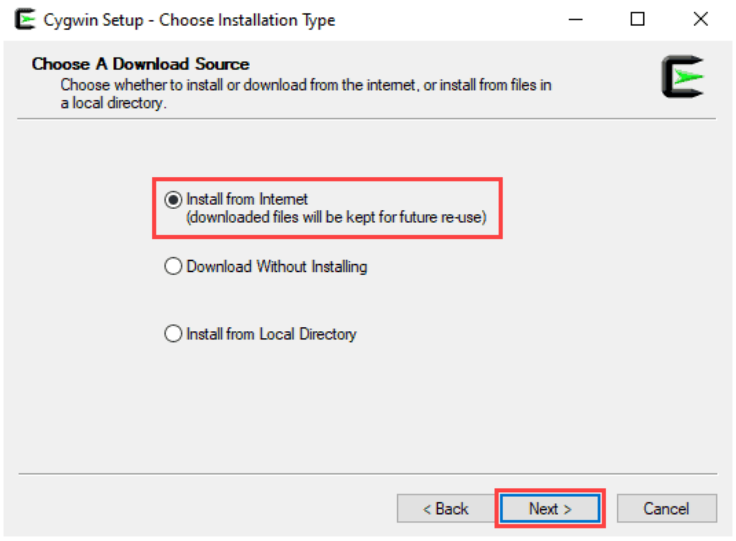
You can always run this program at any time in the future to add, remove, or upgrade packages as necessary.

Setup version 2.905 (64 bit)

Copyright 2000-2020

https://cygwin.com/

3. Select **Install from Internet** as the download source and click **Next**.



Choose A Download Source

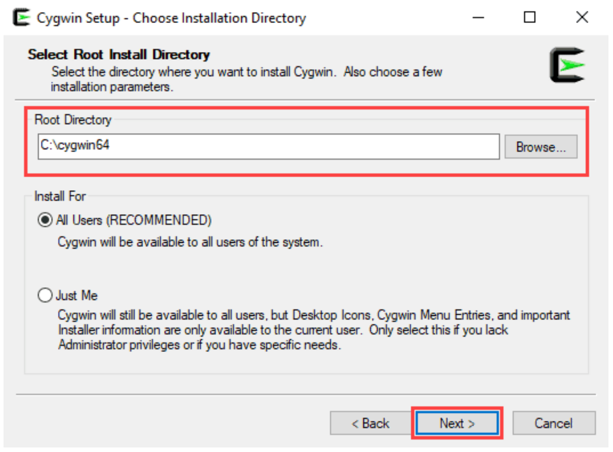
Choose whether to install or download from the internet, or install from files in a local directory.

[✓ Install from Internet (downloaded files will be kept for future re-use)]

Download Without Installing

Install from Local Directory

4. In the *Root Directory* field, specify where you want the application installed, then click **Next**.


Cygwin Setup - Choose Installation Directory

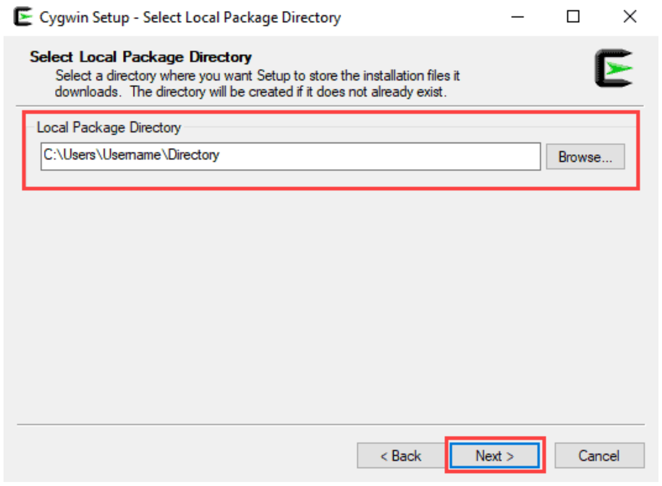
Select Root Install Directory
Select the directory where you want to install Cygwin. Also choose a few installation parameters.

Root Directory
C:\cygwin64

Install For
(●) All Users (RECOMMENDED)
Cygwin will be available to all users of the system.

( ) Just Me
Cygwin will still be available to all users, but Desktop Icons, Cygwin Menu Entries, and important Installer information are only available to the current user. Only select this if you lack Administrator privileges or if you have specific needs.

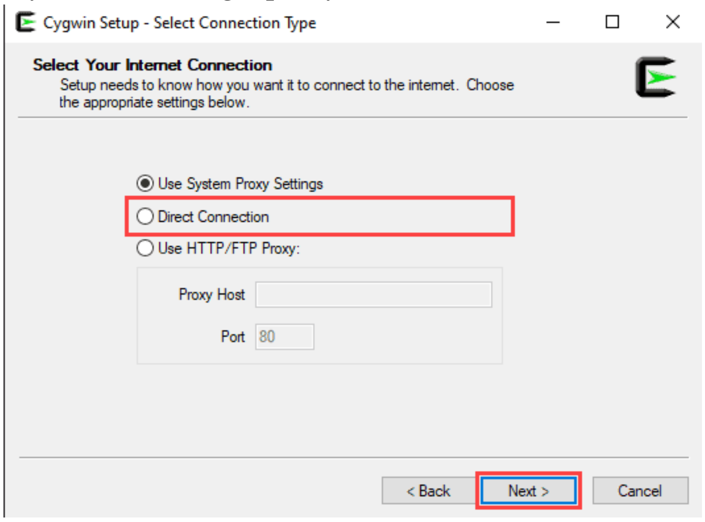
5. In the *Local Package Directory* field, select where you want to install your Cygwin packages, then click **Next**.


Cygwin Setup - Select Local Package Directory

Select Local Package Directory
Select a directory where you want Setup to store the installation files it downloads. The directory will be created if it does not already exist.

Local Package Directory
C:\Users\Username\Directory

6. Choose the appropriate Internet connection option.
If you aren't using a proxy, select **Direct Connection**.

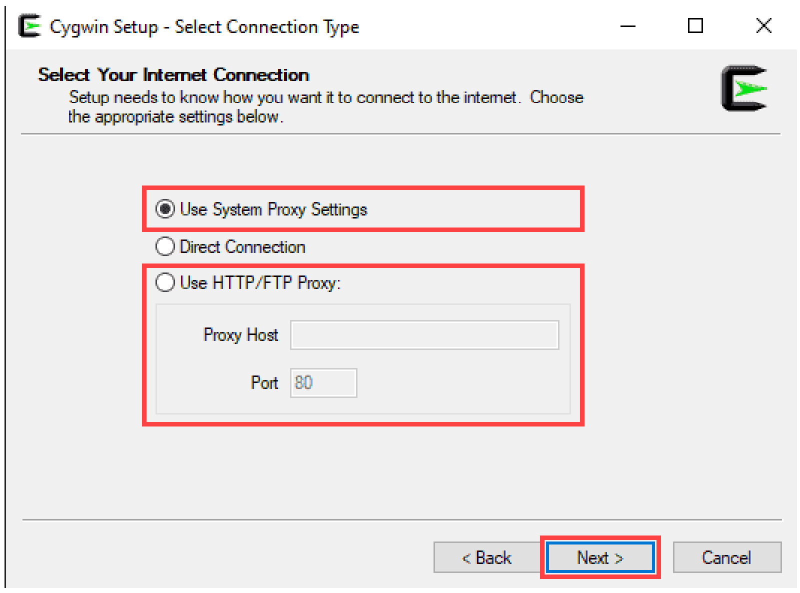

Cygwin Setup - Select Connection Type

Select Your Internet Connection
Setup needs to know how you want it to connect to the internet. Choose the appropriate settings below.

(●) Use System Proxy Settings
( ) Direct Connection
( ) Use HTTP/FTP Proxy:

Proxy Host
Port 80

If you are using a proxy, select **Use System Proxy Settings** or enter the proxy settings manually with the **Use HTTP/FTP Proxy.**


Cygwin Setup - Select Connection Type

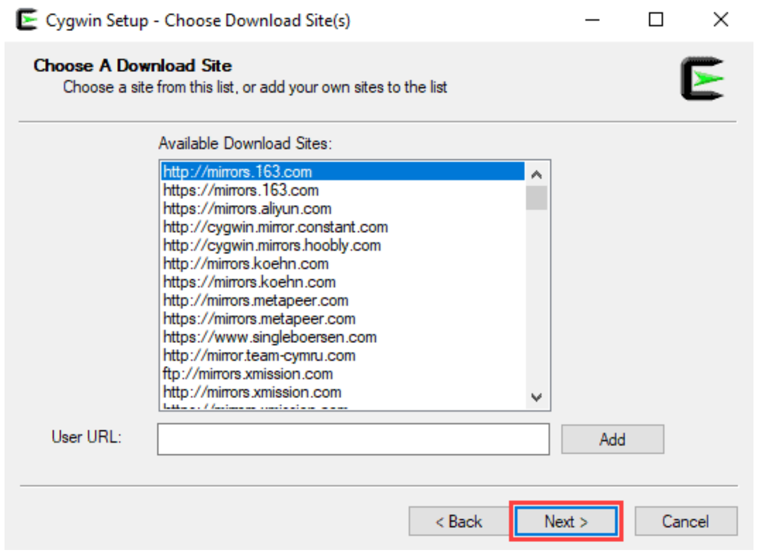
Select Your Internet Connection
Setup needs to know how you want it to connect to the internet. Choose the appropriate settings below.

[● Use System Proxy Settings]
○ Direct Connection
○ Use HTTP/FTP Proxy:
Proxy Host
Port 80

< Back | Next > | Cancel

Click **Next** to continue.

7. Choose one of the available mirrors to download the installation files, then click **Next**.


Cygwin Setup - Choose Download Site(s)

Choose A Download Site
Choose a site from this list, or add your own sites to the list

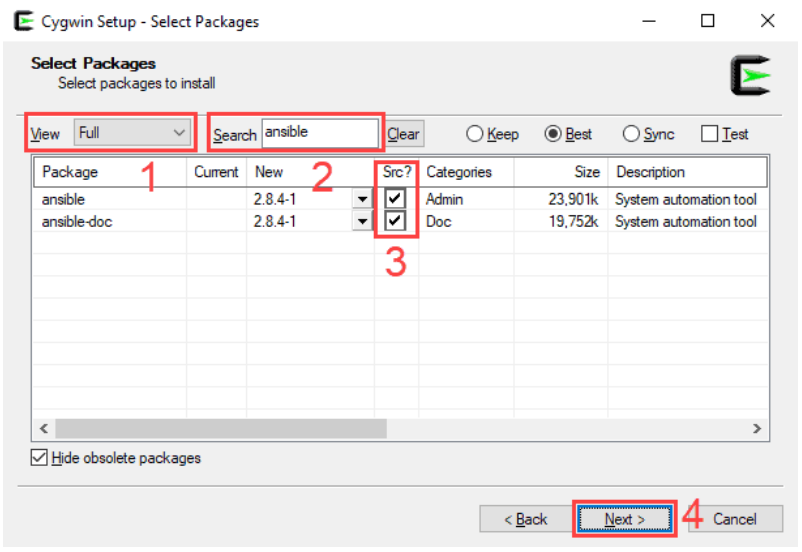
Available Download Sites:

```
http://mirrors.163.com
https://mirrors.163.com
https://mirrors.aliyun.com
http://cygwin.mirror.constant.com
http://cygwin.mirrors.hoobly.com
http://mirrors.koehn.com
https://mirrors.koehn.com
http://mirrors.metapeer.com
https://mirrors.metapeer.com
https://www.singleboersen.com
http://mirror.team-cymru.com
ftp://mirrors.xmission.com
http://mirrors.xmission.com
```

User URL: | Add

< Back | Next > | Cancel

8. On the *Select Packages* screen, change the **View** option to **Full** and type ‘ansible’ in the search bar.
Select both **Ansible** and **Ansible Doc** by checking the boxes under *Src?* and click **Next** to continue.


Cygwin Setup - Select Packages

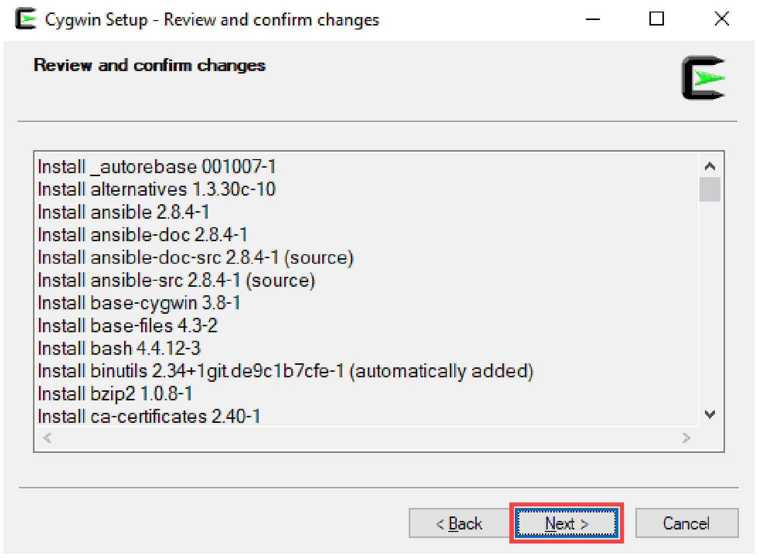
Select Packages
Select packages to install

View: Full | Search: ansible | Clear | Keep [✓ Best] Sync | Test

Package	Current	New	Src?	Categories	Size	Description
ansible		2.8.4-1	✓	Admin	23,901k	System automation tool
ansible-doc		2.8.4-1	✓	Doc	19,752k	System automation tool

✓ Hide obsolete packages

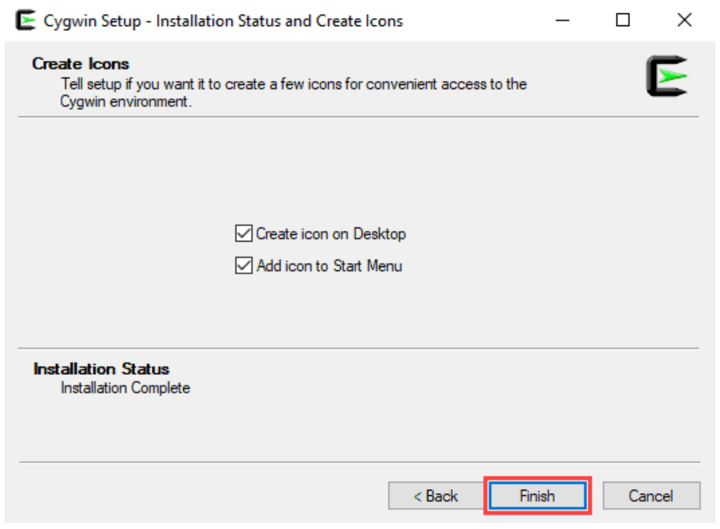
9. This screen lets you review the installation settings. To confirm and begin the install process, click on **Next**.


Cygwin Setup - Review and confirm changes

Review and confirm changes

```
Install _autorebase 001007-1
Install alternatives 1.3.30c-10
Install ansible 2.8.4-1
Install ansible-doc 2.8.4-1
Install ansible-doc-src 2.8.4-1 (source)
Install ansible-src 2.8.4-1 (source)
Install base-cygwin 3.8-1
Install base-files 4.3-2
Install bash 4.4.12-3
Install binutils 2.34+1git.de9c1b7cfe-1 (automatically added)
Install bzip2 1.0.8-1
Install ca-certificates 2.40-1
```

10. The install wizard will download and install all the selected packages, including Ansible.
11. Once the installation is complete, select whether you want to add a Cygwin desktop and Start Menu icon, then click on **Finish** to close the wizard.


Cygwin Setup - Installation Status and Create Icons

Create Icons
Tell setup if you want it to create a few icons for convenient access to the Cygwin environment.

☑ Create icon on Desktop
☑ Add icon to Start Menu

Installation Status
Installation Complete

OUTCOME: Install Ansible

CONCLUSION:

Students will be able to install ansible

Experiment 11: Installation and Configuration of Ansible

Introduction

Configuration Management

Configuration Management Elements

Configuration Management user roles

Ansible ?

Ansible Design Principles

Ansible History

Ansible Working

Installation of Ansible

Ansible Commands

Conclusion

14. Lab Exercise

Exercise No 12: (2 Hours) – 1 Practical

Aim: - Continuous Monitoring using Nagios

Objective:

1. Understand the use of Nagios for continuous monitoring

Pre-requisites:

THEORY:

What is Continuous Monitoring

Continuous monitoring starts when the deployment is done on the production servers. From then on, this stage is responsible to monitor everything happening. This stage is very crucial for the business productivity.

There are several benefits of using Continuous monitoring −

It detects all the server and network problems.

It finds the root cause of the failure.

It helps in reducing the maintenance cost.

It helps in troubleshooting the performance issues.

It helps in updating infrastructure before it gets outdated.

It can fix problems automatically when detected.

It makes sure the servers, services, applications, network is always up and running.

It monitors complete infrastructure every second.

What is Nagios

Nagios is an open-source app for monitoring systems, networks, and IT infrastructure. The tool allows users to track the state and performance of:

- Hardware (routers, switches, firewalls, dedicated servers, workstations, printers, etc.).
- Networks.
- Apps.
- Services.
- Business processes.
- Operating systems (Windows, Linux, Unix, and OSX).

Nagios runs periodic checks on critical thresholds and metrics to monitor for system changes and potential problems. If the software runs into an issue, the tool notifies admins and can also run automatic scripts to contain and remedy the situation.

You can use Nagios to monitor:

- Memory and disk usage.
- CPU loads.
- The number of running processes.
- Log files.

- System availability.
- Response times.
- URL and content monitoring metrics.
- Services and network protocols (SMTP, POP3, HTTP, etc.).

The tool is available in two main variants:

- **Nagios Core:** The free version of the software that allows users to track all essential metrics.
- **Nagios XI:** A paid, extended version of Core that provides advanced components and tools for monitoring.

This software is a common tool of choice in DevOps circles due to the solution's scalability, efficiency, and flexibility.

Why Nagios

Nagios offers the following features making it usable by a large group of user community −

It can monitor Database servers such as SQL Server, Oracle, Mysql, Postgres

It gives application level information (Apache, Postfix, LDAP, Citrix etc.).

Provides active development.

Has excellent support form huge active community.

Nagios runs on any operating system.

It can ping to see if host is reachable.

Benefits of Nagios

Nagios offers the following benefits for the users −

It helps in getting rid of periodic testing.

It detects split-second failures when the wrist strap is still in the “intermittent” stage.

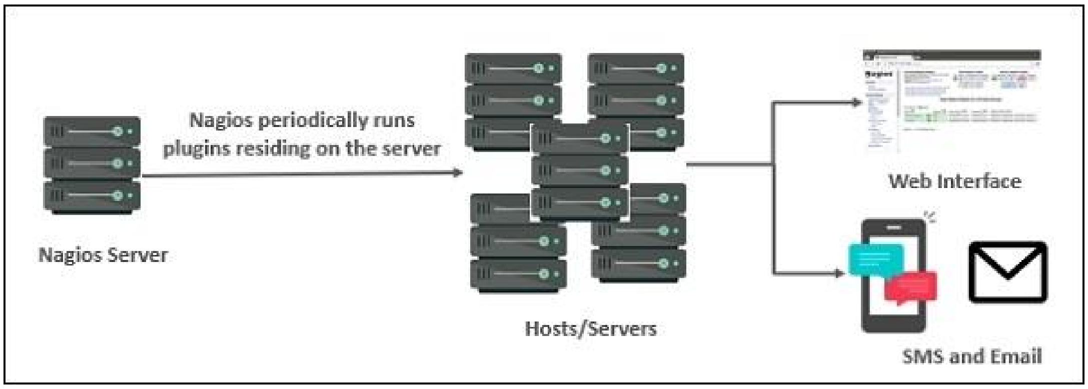
It reduces maintenance cost without sacrificing performance.

It provides timely notification to the management of control and breakdown.

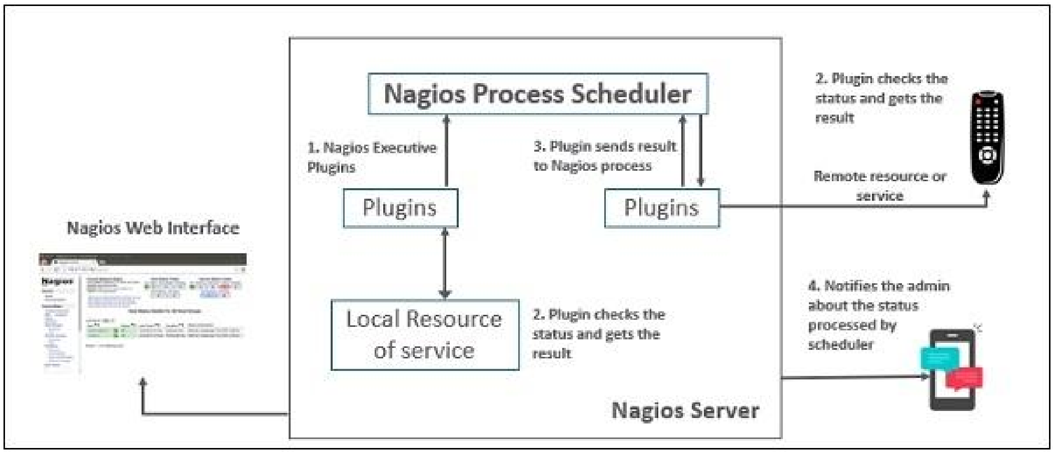
Nagios Architecture

The following points are worth notable about Nagios architecture −

- Nagios has server-agent architecture.
- Nagios server is installed on the host and plugins are installed on the remote hosts/servers which are to be monitored.
- Nagios sends a signal through a process scheduler to run the plugins on the local/remote hosts/servers.
- Plugins collect the data (CPU usage, memory usage etc.) and sends it back to the scheduler.
- Then the process schedules send the notifications to the admin/s and updates Nagios GUI.



The following figure shows Nagios Server Agent Architecture in detail −


Nagios Process Scheduler
1. Nagios Executive Plugins
3. Plugin sends result to Nagios process
Plugins
Plugins
2. Plugin checks the status and gets the result
Remote resource or service
Nagios Web Interface
Local Resource of service
2. Plugin checks the status and gets the result
4. Notifies the admin about the status processed by scheduler
Nagios Server

Nagios Core Installation

The step-by-step guide below shows how to install Nagios Core on Ubuntu 20.04.

Step 1: Check for System Updates

1. You should first check the web for the latest available packages:

$ sudo apt update

2. Next, upgrade the system packages to the latest versions:

$ sudo apt upgrade

Step 2: Install Prerequisite Packages

After updating the system, you need to install the packages required to run Core. Enter the following code into the command line:

$ sudo apt install wget unzip vim curl gcc openssl build-essential libgd-dev libssl-dev libapache2-mod-php php-gd php apache2

Step 3: Download Nagios Core

Download Nagios Core. To do so, browse to the official Git repository and select the latest release. Alternatively, you can download the tool from the official website. At the time of writing this article, the latest version of Nagios is 4.4.6, so we use the following command to extract the tool:

```
$ export VER="4.4.6"
```

Now, use the curl command:

```
$ curl -SL https://github.com/NagiosEnterprises/nagioscore/releases/download/nagios-$VER/nagios-$VER.tar.gz | tar -xzf -
```

This command downloads a directory called nagios-4.4.6 and adds it to your current working directory.

Step 4: Install Nagios

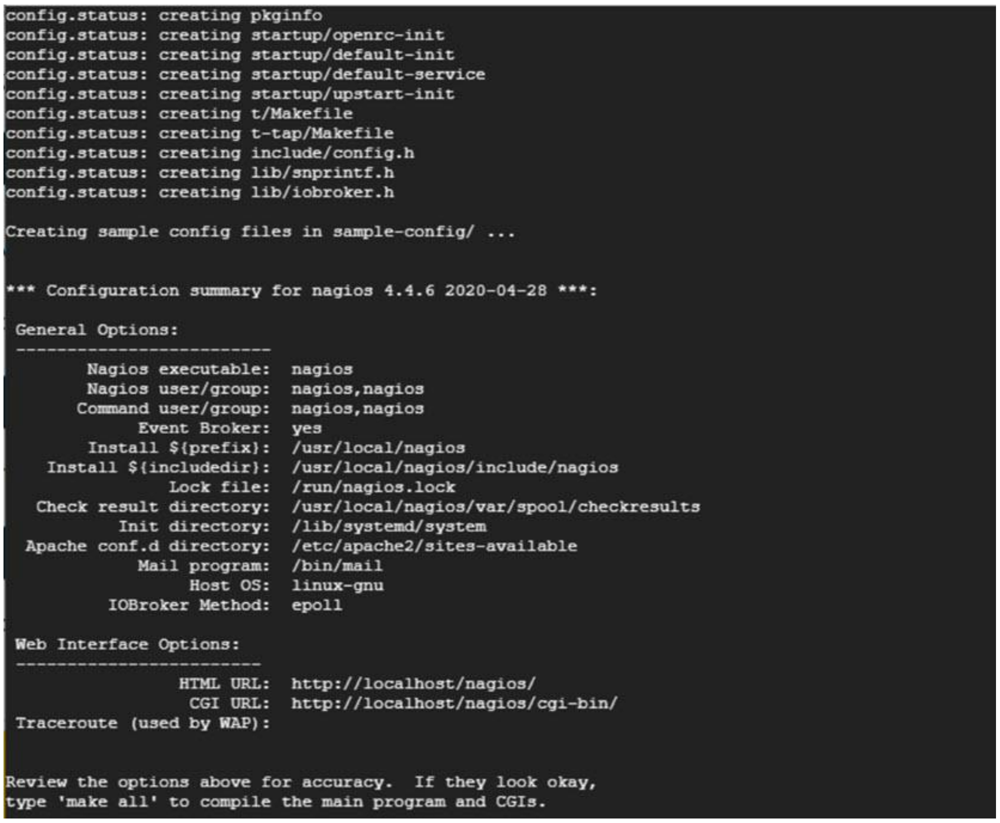
We now need to install Core by compiling from the source.

1. Navigate into the Nagios directory:

```
$ cd nagios-4.4.6
```

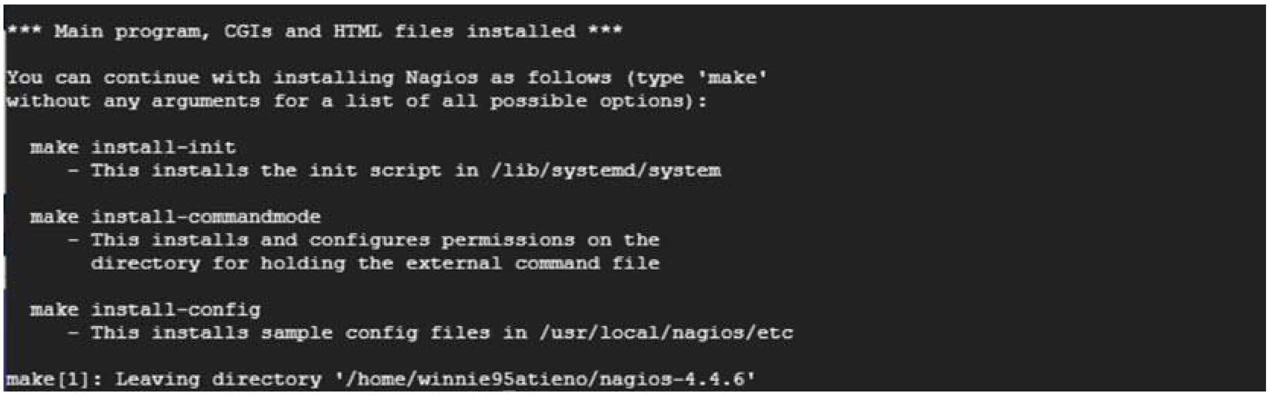
2. Run the configure script:

```
$ ./configure
```



```
config.status: creating pkginfo
config.status: creating startup/openrc-init
config.status: creating startup/default-init
config.status: creating startup/default-service
config.status: creating startup/upstart-init
config.status: creating t/Makefile
config.status: creating t-tap/Makefile
config.status: creating include/config.h
config.status: creating lib/snprintf.h
config.status: creating lib/iobroker.h

Creating sample config files in sample-config/ ...


*** Configuration summary for nagios 4.4.6 2020-04-28 ***:

 General Options:
 -------------------------
        Nagios executable:  nagios
        Nagios user/group:  nagios,nagios
       Command user/group:  nagios,nagios
             Event Broker:  yes
        Install ${prefix}:  /usr/local/nagios
    Install ${includedir}:  /usr/local/nagios/include/nagios
                Lock file:  /run/nagios.lock
   Check result directory:  /usr/local/nagios/var/spool/checkresults
           Init directory:  /lib/systemd/system
  Apache conf.d directory:  /etc/apache2/sites-available
             Mail program:  /bin/mail
                  Host OS:  linux-gnu
          IOBroker Method:  epoll

 Web Interface Options:
 ------------------------
                 HTML URL:  http://localhost/nagios/
                  CGI URL:  http://localhost/nagios/cgi-bin/
 Traceroute (used by WAP):


Review the options above for accuracy.  If they look okay,
type 'make all' to compile the main program and CGIs.
```

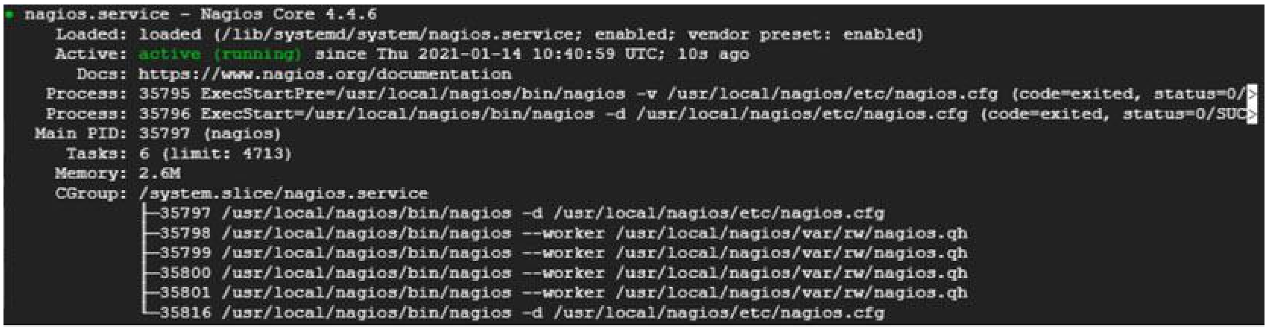
3. Run the make all command to compile the program alongside the CGIs:
$ sudo make all
4. Next, we need to create group users:
$ sudo make install-groups-users
$ sudo usermod -a -G nagios www-data
5. Now install Nagios Core on your Ubuntu system:
$ sudo make install



```
*** Main program, CGIs and HTML files installed ***

You can continue with installing Nagios as follows (type 'make'
without any arguments for a list of all possible options):

  make install-init
     - This installs the init script in /lib/systemd/system

  make install-commandmode
     - This installs and configures permissions on the
       directory for holding the external command file

  make install-config
     - This installs sample config files in /usr/local/nagios/etc

make[1]: Leaving directory '/home/winnie95atieno/nagios-4.4.6'
```

6. As you can see, some additional instructions appear on the screen. Run the following command to install the init script in the /lib/systemd/system path:
$ sudo make install-init
7. Next, install and configure permissions on the directory:
$ sudo make install-commandmode
8. Finally, install sample config files in /usr/local/nagios/etc/:

```
$ sudo make install-config
```
Step 5: Set up Apache and Nagios UI
1. You need to enable the Apache module required for the Nagios web interface, so run the following command:
```
$ sudo make install-webconf
$ sudo a2enmod rewrite cgi
$ sudo systemctl restart apache2
```
2. Type in the following command for the classic Nagios monitoring theme:
```
$ sudo make install-classicui
```
Step 6: Create the First Nagios User
We now need to create a user that can log in to Nagios. The following command creates a user called nagadmin:
```
$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagadmin
```
You need to provide a password for the user and confirm it (by default, passwords are stored in /usr/local/nagios/etc/htpasswd.users).
Step 7: Install Nagios Plugins
Look at the latest available plugins at the official repository (at the time of writing this article, the newest released version is 2.3.3).
1. To download plugins, type the following command:
```
$ VER="2.3.3"
$ curl -SL
https://github.com/nagios-plugins/nagios-plugins/releases/download/release-$VER/na
gios-plugins-$VER.tar.gz | tar -xzf -
```
2. This command creates a new directory (nagios-plugins-2.3.3) in your current working directory. To install plugins, you first need to navigate to the new directory:
```
$ cd nagios-plugins-2.3.3
```
3. Now compile the plugins from source:
```
$ ./configure --with-nagios-user=nagios --with-nagios-group=nagios
$ sudo make install
```
4. To make sure all configurations are in order, run the following command:
```
$ sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```
Step 8: Start the Nagios Daemon
1. The last step is to start the Nagios service, which we achieve with the following command:
```
$ sudo systemctl enable --now nagios
```
2. To make sure the tool is running, use the following command:
```
$ sudo systemctl status nagios
```



```
● nagios.service - Nagios Core 4.4.6
     Loaded: loaded (/lib/systemd/system/nagios.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2021-01-14 10:40:59 UTC; 10s ago
       Docs: https://www.nagios.org/documentation
    Process: 35795 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/>
    Process: 35796 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUC>
   Main PID: 35797 (nagios)
      Tasks: 6 (limit: 4713)
     Memory: 2.6M
     CGroup: /system.slice/nagios.service
             ├─35797 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─35798 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─35799 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─35800 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─35801 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─35816 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
```

3. You can now access the tool by opening your browser and navigating to the http://server-IP/nagios URL.

4. Once prompted, type in the credentials defined in step 6 to sign in and you are ready to start monitoring.

OUTCOME: Install and use Nagios for continuous monitoring

CONCLUSION:

Successfully installed Nagios for continuous monitoring