

Set up a CI/CD Pipeline to Automate Application Deployment to Kubernetes Using Jenkins

Table of Contents

1. [Introduction](#)
 2. [Problem Statement](#)
 3. [Prerequisites](#)
 - [Software Requirements](#)
 - [Hardware Requirements](#)
 4. [Lab Guide: Setting Up CI/CD Pipeline with Jenkins and Kubernetes](#)
 - [Step 1: Download and Install Jenkins](#)
 - [Step 2: Create a Kubernetes Service Account for Jenkins](#)
 - [Step 3: Set Up Jenkins Kubernetes Plugin](#)
 - [Step 4: Create and Run a Jenkins Pipeline](#)
 5. [References](#)
-

Introduction

CI/CD pipelines automate the process of integrating code changes, testing, and deploying applications. In this guide, we'll set up **Jenkins** on a Windows machine and configure it to deploy applications to a **Kubernetes** cluster using the **Kubernetes Plugin**. Jenkins will run the deployments in Kubernetes pods, which will allow seamless integration and testing directly within the cluster environment.

Problem Statement

Manually deploying applications to Kubernetes can be time-consuming and error-prone. By setting up a CI/CD pipeline using Jenkins and Kubernetes, we can automate the deployment process, ensuring quick, reliable, and continuous integration of code changes to our Kubernetes environment.

Prerequisites

Completion of all previous lab guides (up to Lab Guide-09) is required before proceeding with Lab Guide-10.

- A **Kubernetes** cluster running on **Minikube** or another environment.
- **kubectl** configured to interact with the Kubernetes cluster.
- A Windows machine to install Jenkins.

Software Requirements

- **Jenkins** (download as MSI)
- **Java Development Kit (JDK)** 8 or later (required for Jenkins)
- **Minikube** (if using a local Kubernetes cluster)
- **kubectl** CLI tool for managing Kubernetes

Hardware Requirements

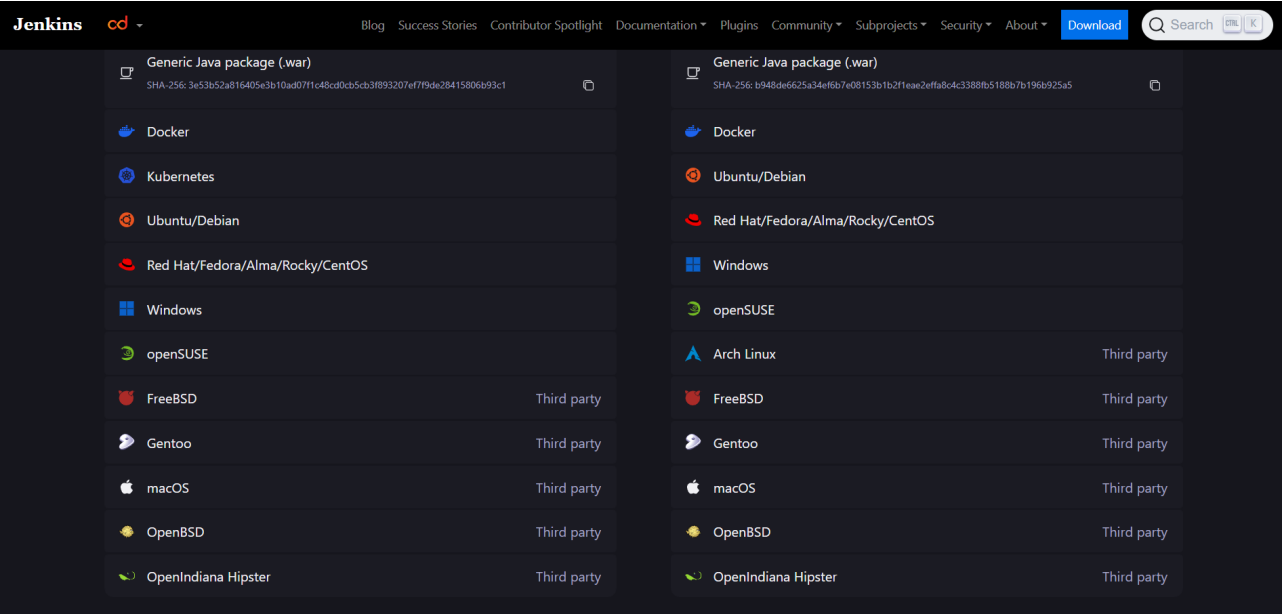
- Minimum 2 CPU cores
- 4GB RAM for Minikube and Jenkins

Lab Guide: Setting Up CI/CD Pipeline with Jenkins and Kubernetes

Step 1: Download and Install Jenkins

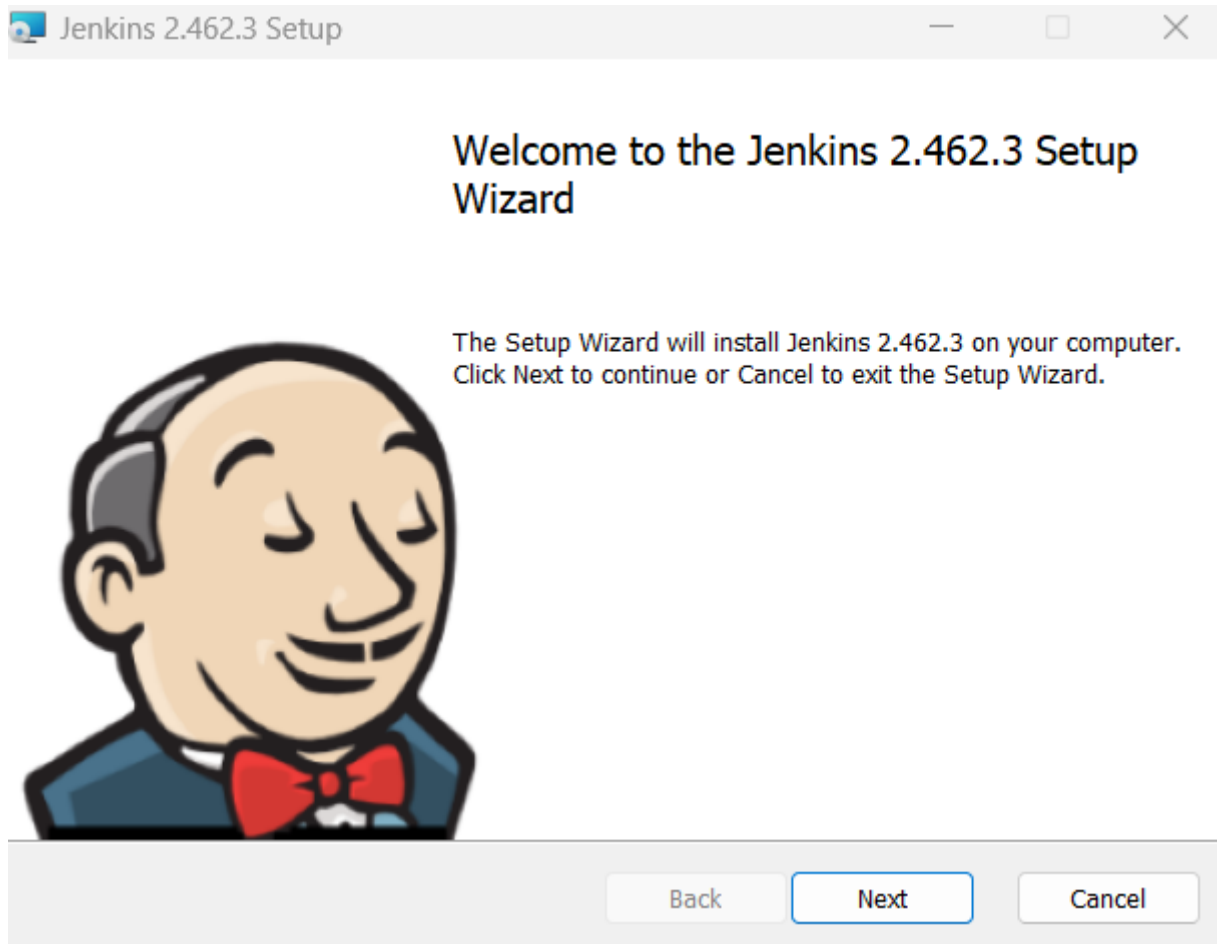
1. Download Jenkins MSI

Go to the Jenkins website and download the Windows installer [here](#).

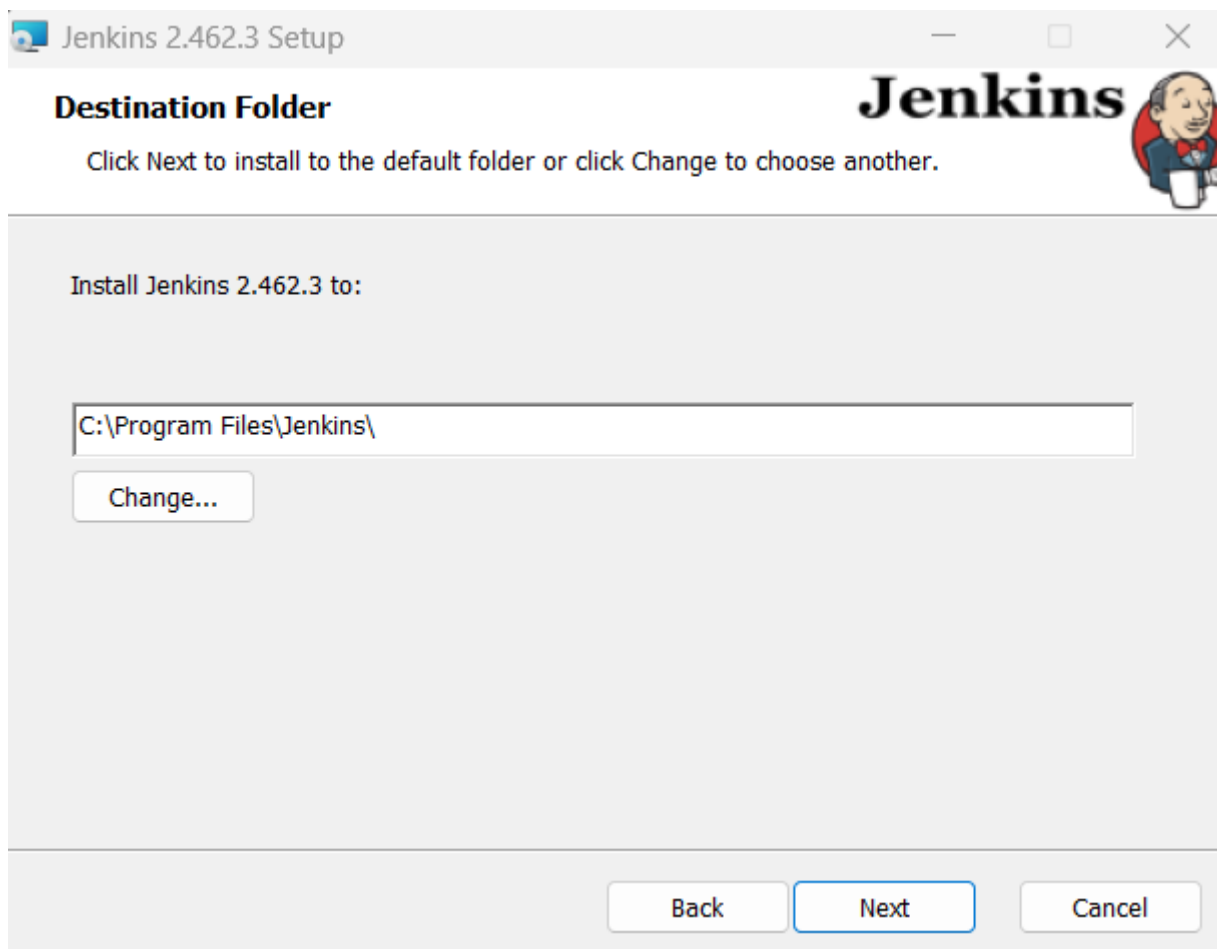


2. Install Jenkins

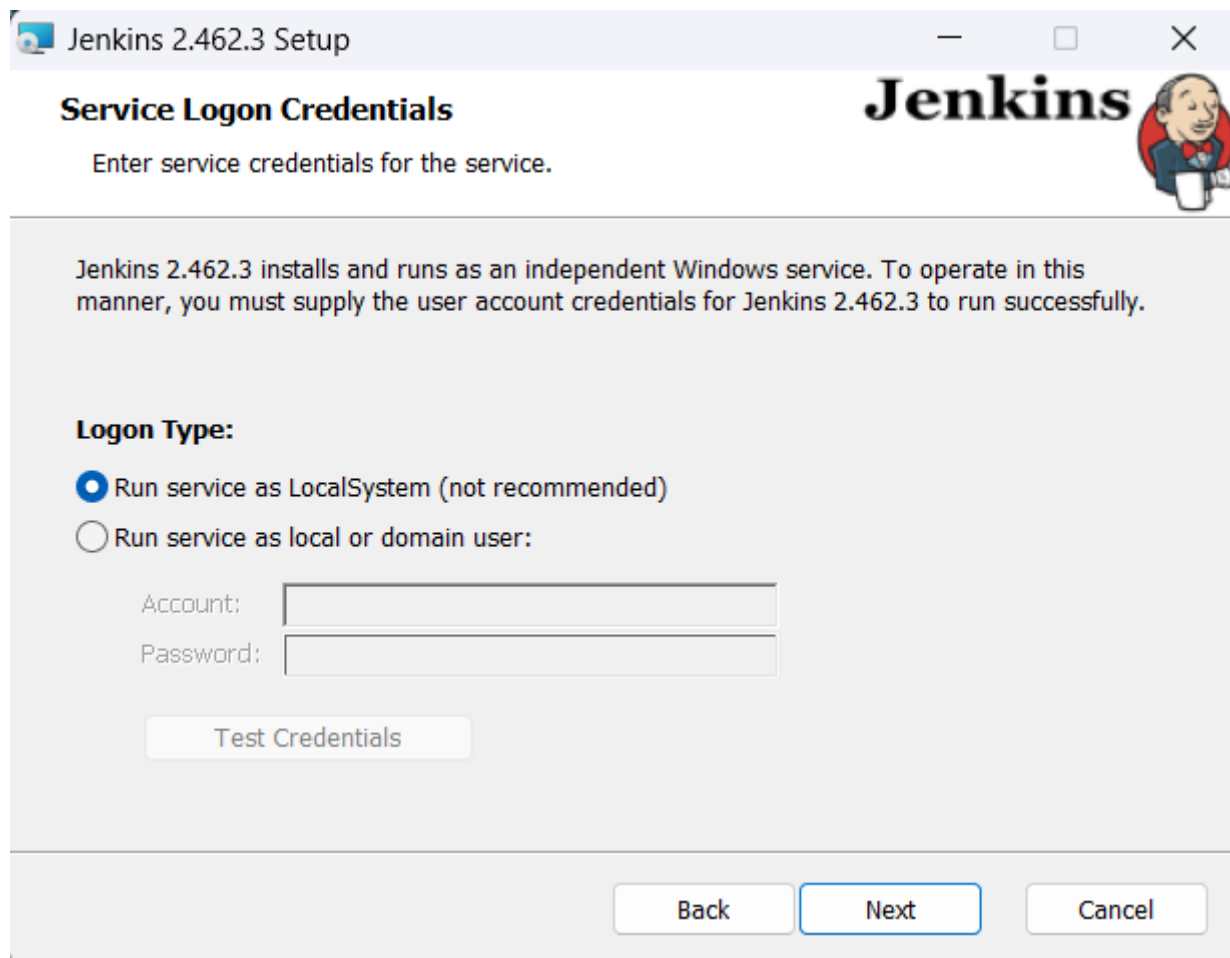
- Double-click the MSI file and follow the instructions to install Jenkins.



- Add Destination folder

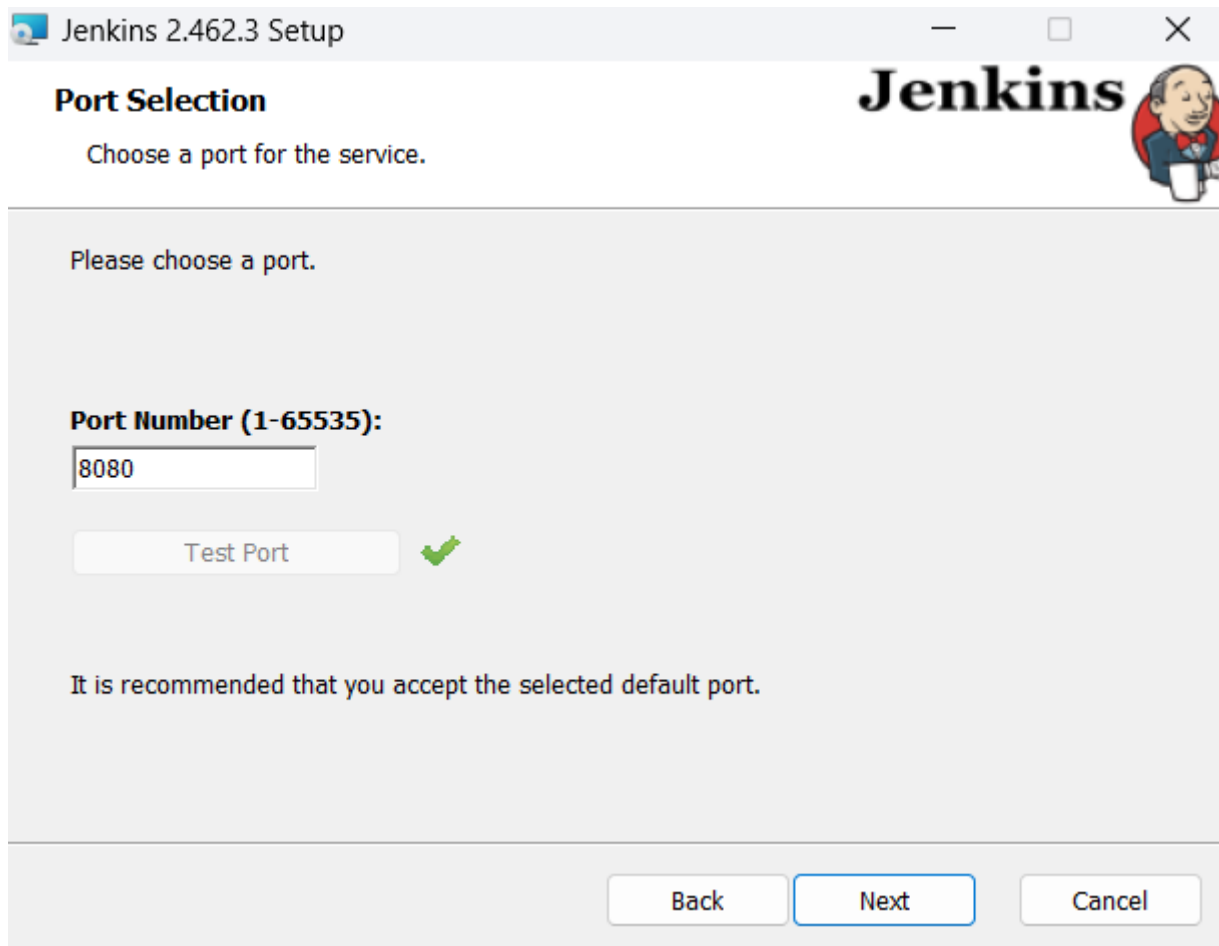


- Run Service on Local System

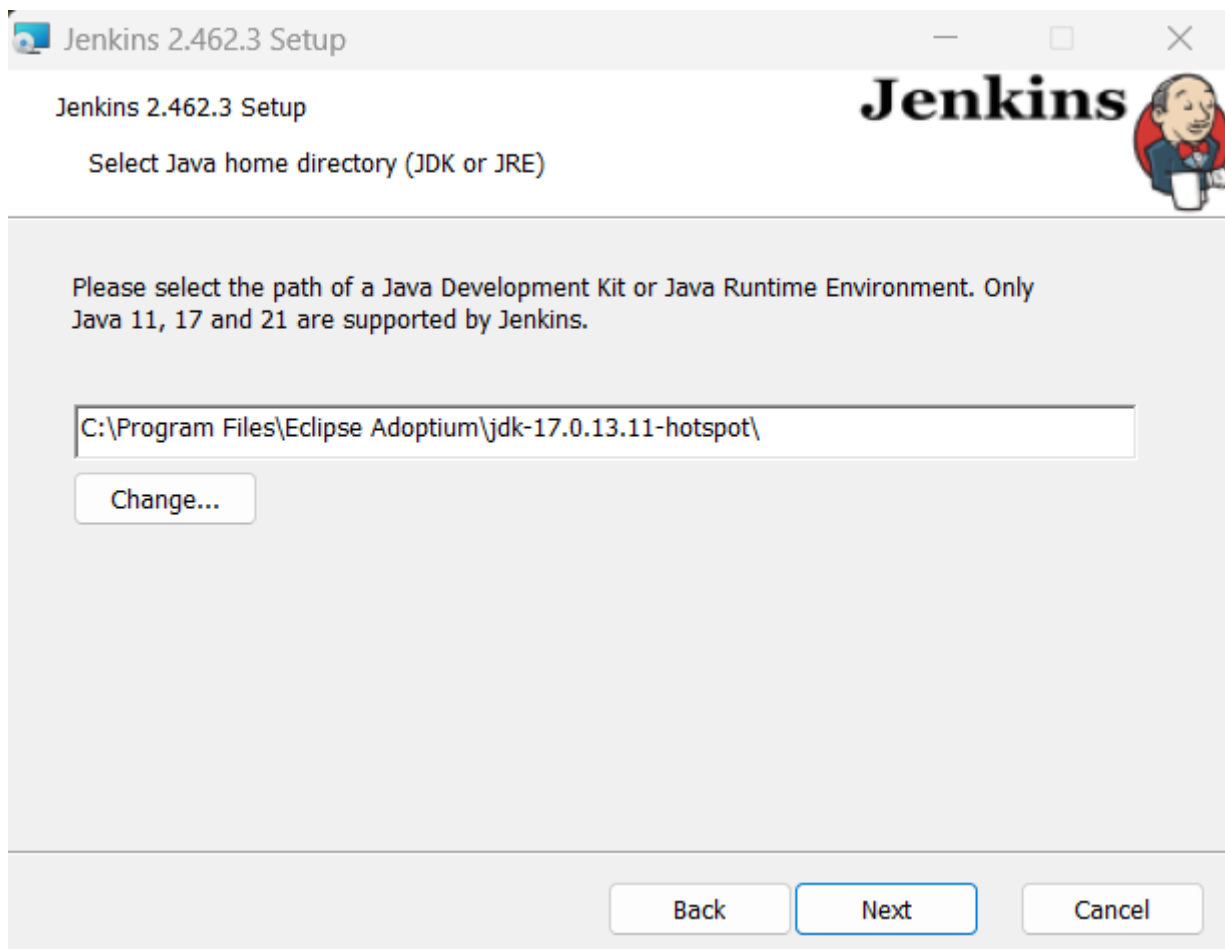


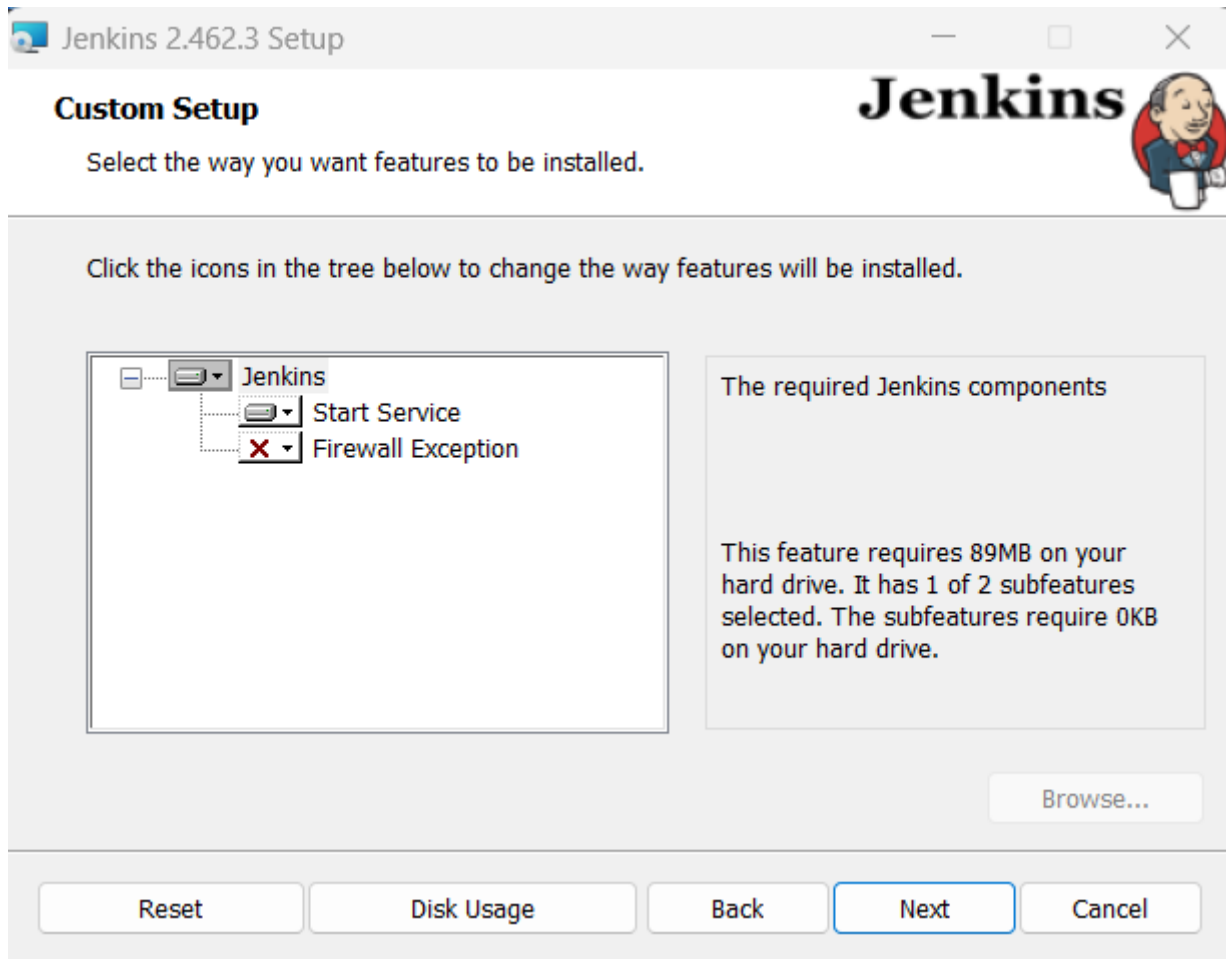
The screenshot shows the 'Jenkins 2.462.3 Setup' window. The title bar says 'Jenkins 2.462.3 Setup'. The main heading is 'Service Logon Credentials' with the Jenkins logo to the right. Below the heading is the instruction 'Enter service credentials for the service.' A large text box contains the following text: 'Jenkins 2.462.3 installs and runs as an independent Windows service. To operate in this manner, you must supply the user account credentials for Jenkins 2.462.3 to run successfully.' Under the heading 'Logon Type:', there are two radio button options: 'Run service as LocalSystem (not recommended)' (which is selected) and 'Run service as local or domain user:'. Below these are two text input fields labeled 'Account:' and 'Password:'. A 'Test Credentials' button is located below the password field. At the bottom right, there are three buttons: 'Back', 'Next' (which is highlighted with a blue border), and 'Cancel'.

- Add port number 8080 and Test Port

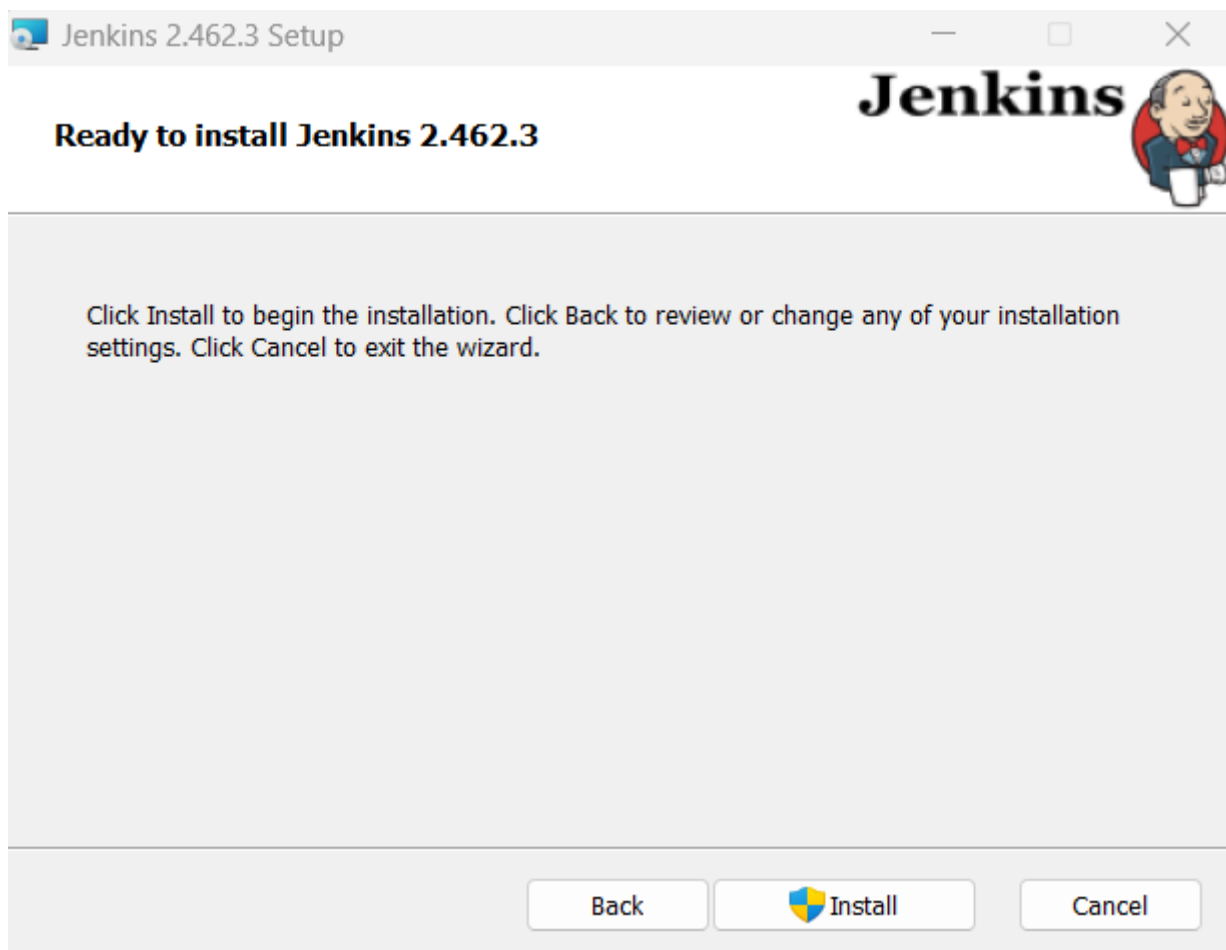


- Select java home directory





- Click on Install



- During the installation, Jenkins will ask for the installation path, Java path, and a port number (default is 8080).

3. Start Jenkins

Once the installation is complete, start Jenkins by visiting <http://localhost:8080> in your browser.

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

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

Administrator password

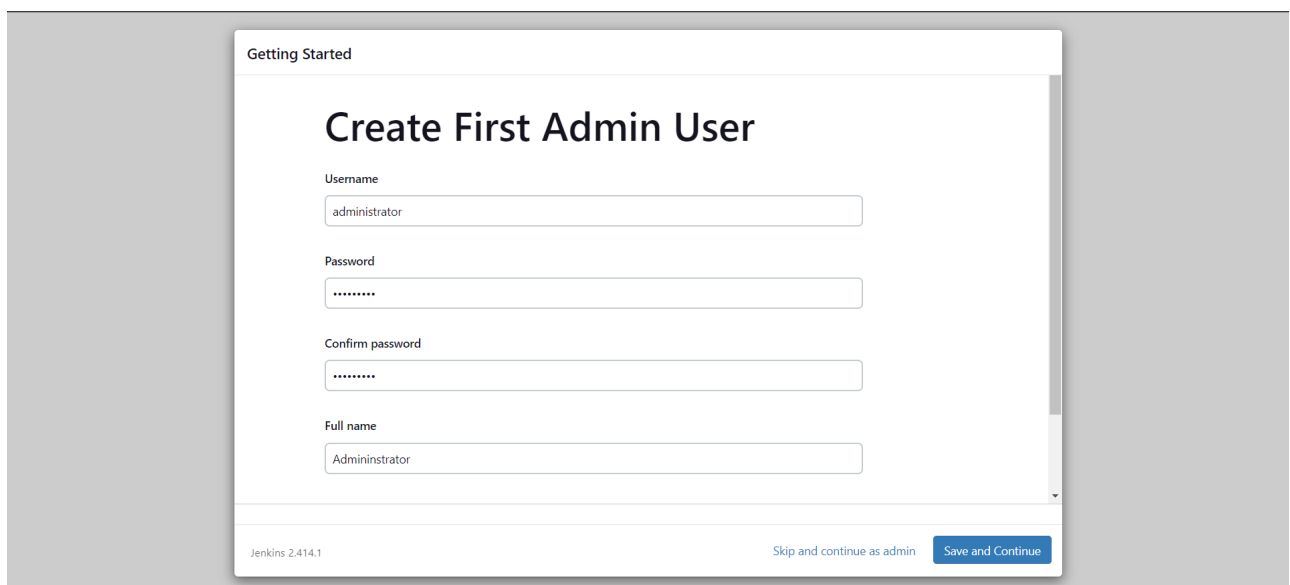
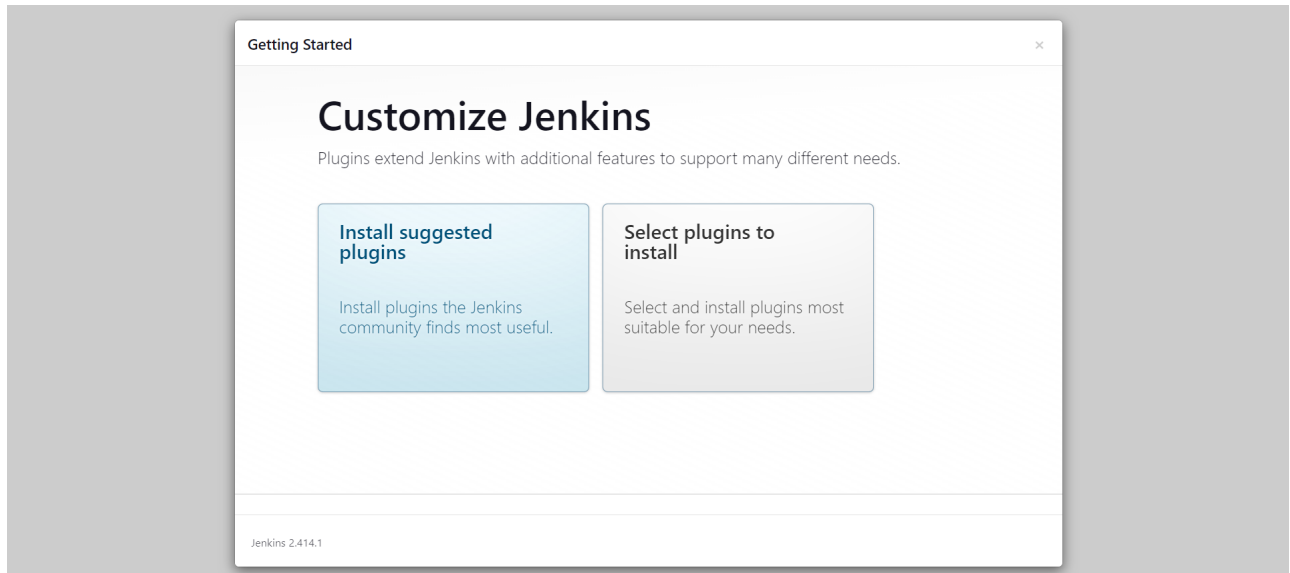
4. Unlock Jenkins

During the initial setup, Jenkins will ask for an admin password. Find the password in the following file:

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

5. Install Suggested Plugins

Jenkins will prompt you to install the suggested plugins. Complete this step and create an admin user when prompted.



Step 2: Create a Kubernetes Service Account for Jenkins

1. Create the Jenkins Namespace

Run the following command to create a namespace for Jenkins:

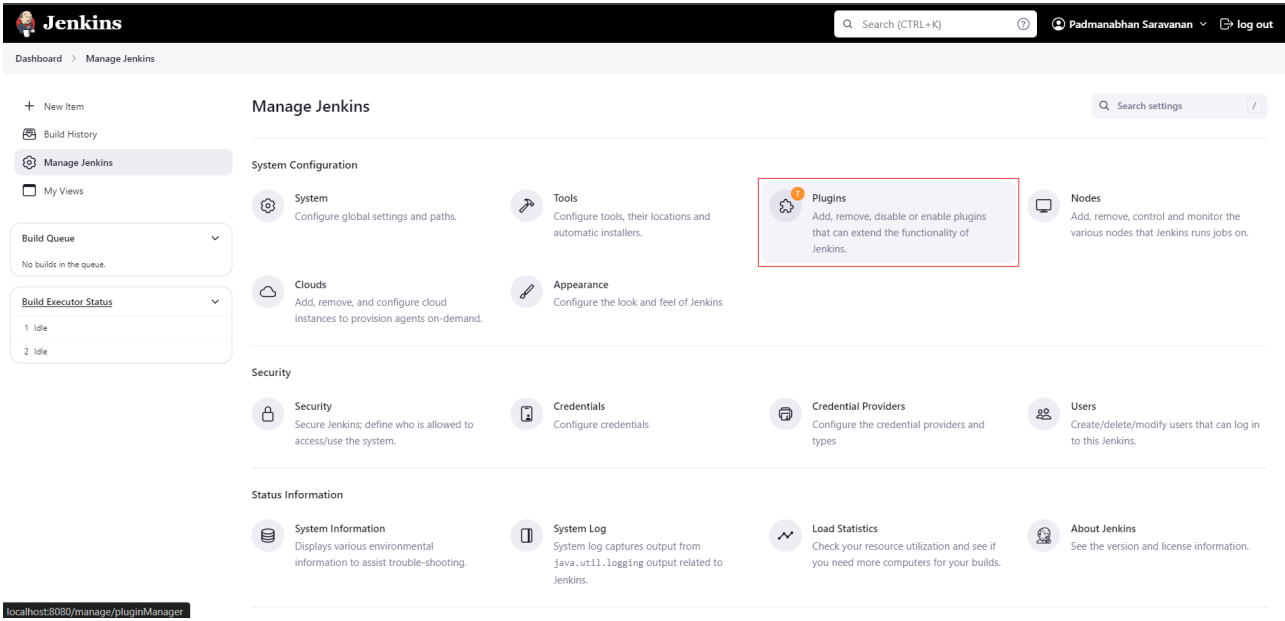
```
kubectl create namespace jenkins
```

```
C:\Windows\System32>kubectl create namespace jenkins
namespace/jenkins created
```

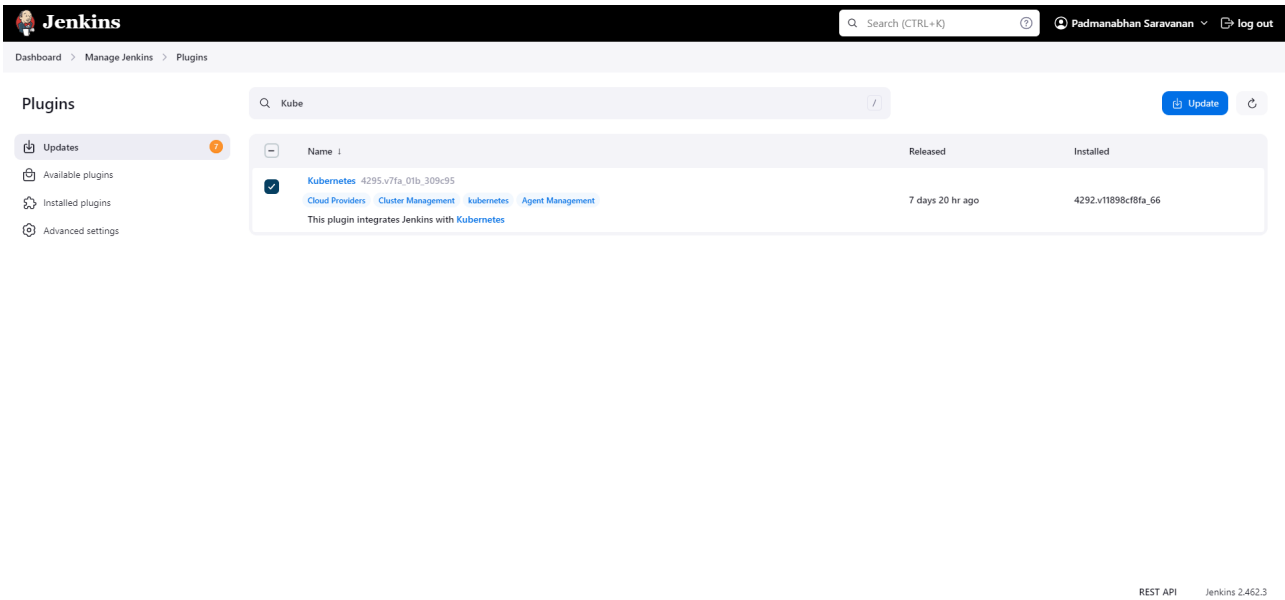
2. Create a Service Account for Jenkins

In the Jenkins namespace, create a service account for Jenkins to access the Kubernetes cluster:

```
kubectl create sa jenkins -n jenkins
```

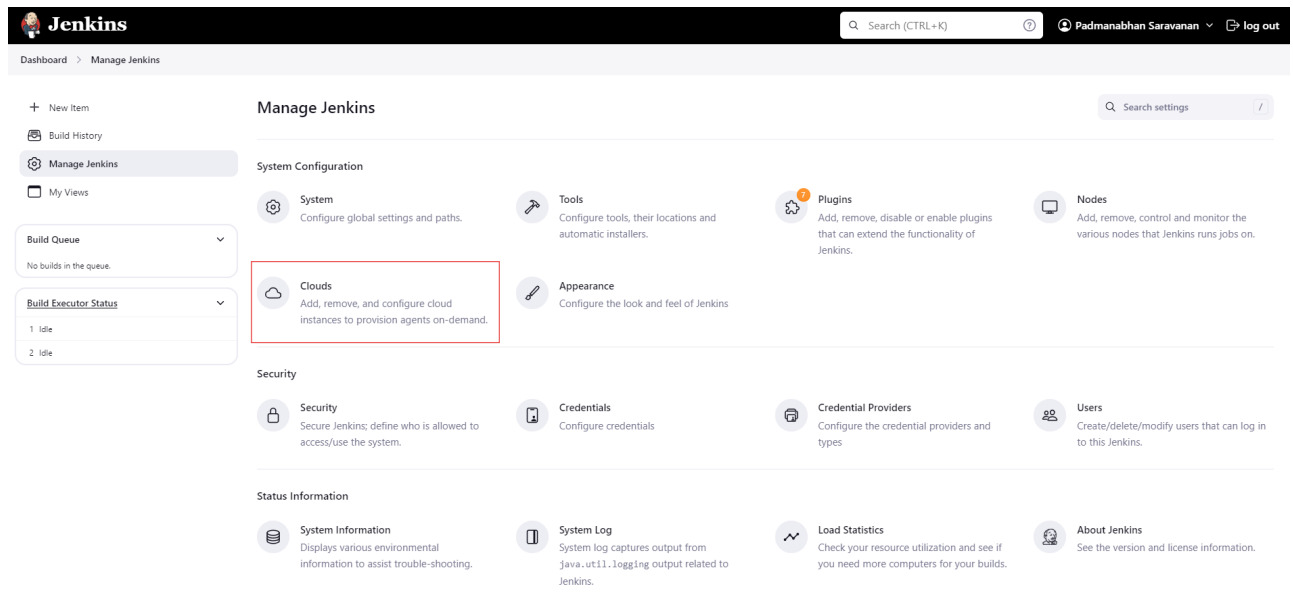



- In the **Available** tab, search for "Kubernetes" and install the **Kubernetes Plugin**.



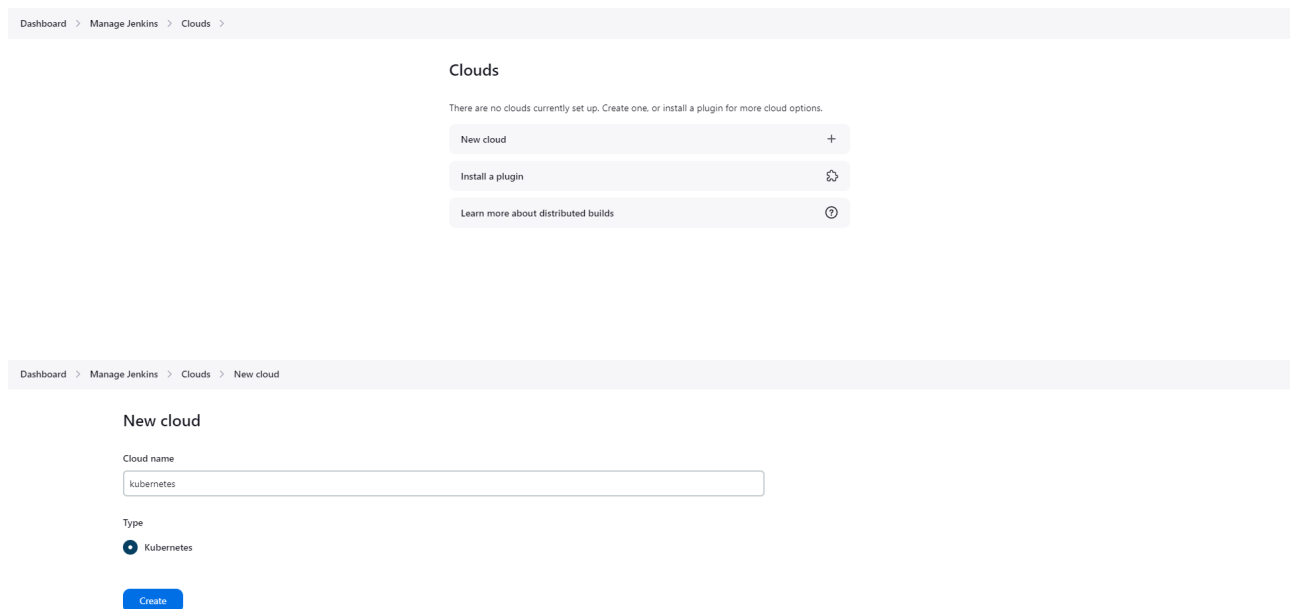
3. **Configure Jenkins to Connect to Kubernetes**

- Go to **Manage Jenkins > Cloud**.



The image shows the Jenkins 'Manage Jenkins' dashboard. The left sidebar contains navigation links: 'New Item', 'Build History', 'Manage Jenkins' (selected), 'My Views', 'Build Queue' (showing 'No builds in the queue'), and 'Build Executor Status' (showing two 'Idle' executors). The main content area is titled 'Manage Jenkins' and features a search bar. It is organized into three sections: 'System Configuration' (with links for System, Tools, Plugins, Nodes, Clouds, and Appearance), 'Security' (with links for Security, Credentials, Credential Providers, and Users), and 'Status Information' (with links for System Information, System Log, Load Statistics, and About Jenkins). The 'Clouds' link in the System Configuration section is highlighted with a red rectangle.

- click **Add a new cloud**. Select **Kubernetes**.



The image displays two screenshots from the Jenkins interface. The top screenshot shows the 'Clouds' page, which indicates that no clouds are currently set up and provides three options: 'New cloud', 'Install a plugin', and 'Learn more about distributed builds'. The bottom screenshot shows the 'New cloud' configuration page. It has a 'Cloud name' text field containing 'kubernetes' and a 'Type' dropdown menu with 'Kubernetes' selected. A blue 'Create' button is located at the bottom of the form.

- Configure the following settings:
 - **Kubernetes URL:** Obtain the URL by running `kubectl config view` and look for the `cluster.server` field.

```
C:\Windows\System32>kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://kubernetes.docker.internal:6443
    name: docker-desktop
- cluster:
    certificate-authority: C:\Users\vijay\.minikube\ca.crt
    extensions:
    - extension:
        last-update: Fri, 25 Oct 2024 09:12:05 IST
        provider: minikube.sigs.k8s.io
        version: v1.34.0
        name: cluster_info
    server: https://127.0.0.1:65024
    name: minikube
contexts:
- context:
    cluster: docker-desktop
    user: docker-desktop
    name: docker-desktop
- context:
    cluster: minikube
    extensions:
    - extension:
        last-update: Fri, 25 Oct 2024 09:12:05 IST
        provider: minikube.sigs.k8s.io
        version: v1.34.0
        name: context_info
    namespace: default
    user: minikube
    name: minikube
current-context: minikube
kind: Config
preferences: {}
users:
- name: docker-desktop
  user:
    client-certificate-data: DATA+OMITTED
    client-key-data: DATA+OMITTED
- name: minikube
  user:
    client-certificate: C:\Users\vijay\.minikube\profiles\minikube\client.crt
    client-key: C:\Users\vijay\.minikube\profiles\minikube\client.key
```

■ Kubernetes Namespace: Enter `jenkins`.

Dashboard > Manage Jenkins > Clouds > New cloud

New cloud

Name [?]

kubernetes

Kubernetes URL [?]

https://127.0.0.1:65024

☐ Use Jenkins Proxy [?]

Kubernetes server certificate key [?]

☒ Disable https certificate check [?]

Kubernetes Namespace

jenkins

- **Jenkins URL:** Add your Jenkins URL (<http://localhost:8080>) and ensure **WebSocket** is enabled.

Dashboard > Manage Jenkins > Clouds > New cloud

☒ **WebSocket** ?
☐ Direct Connection ?

Jenkins URL ?

Jenkins tunnel ?

Connection Timeout ?

Read Timeout ?

Concurrency Limit ?

Pod Labels ?

- **Disable HTTPS Certificate Check:** Ensure this is checked for local environments.

- In the **Credentials** section, use the token created earlier by selecting **Add > Jenkins** and selecting **Secret Text**. Paste the Kubernetes token from the earlier step.

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Kind

Scope ?

Secret

ID ?

Description ?

4. Test the Connection

- Click **Test Connection** to verify Jenkins can connect to Kubernetes. If the connection is successful, click **Save**.

Dashboard > Manage Jenkins > Clouds > New cloud

Kubernetes Namespace

Agent Docker Registry ?

☐ Inject restricted PSS security context in agent container definition ?

Credentials

Connected to Kubernetes v1.31.0

Step 4: Create and Run a Jenkins Pipeline

1. Create a New Pipeline Job

- In Jenkins, click **New Item**, select **Pipeline**, and give it a name.







Dashboard > All > New Item

New Item

Enter an item name

Nginx-Deployment-Pipeline

Select an item type

-  **Freestyle project**
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
-  **Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
-  **Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
-  **Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
-  **Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.
-  **Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

OK

2. Define the Pipeline Script

In the pipeline script section, use the following code to create a pipeline that deploys and runs an Nginx container in a Kubernetes pod:

```
pipeline {
  agent {
    kubernetes {
      yaml '''
        apiVersion: v1
        kind: Pod
        spec:
          containers:
            - name: nginx
              image: nginx:alpine
              command:
                - cat
              tty: true
      '''
    }
  }
  stages {
    stage('Run Nginx') {
      steps {
        container('nginx') {
          // Check the Nginx version
          sh 'nginx -v'
        }
      }
    }
  }
}
```

```
}
}
```

Dashboard > Nginx-Deployment-Pipeline > Configuration

Configure

General
Advanced Project Options
Pipeline

Pipeline

Definition
Pipeline script

Script ?

```

1 pipeline {
2   agent {
3     kubernetes {
4       jenkins {
5         apiVersion: v1
6         kind: Pod
7         spec {
8           containers:
9             - name: nginx
10              image: nginx:alpine
11              command:
12                - cat
13              tty: true
14            ...
15          }
16        }
17      }
18      stages {
19        stage('Run Nginx') {
20          steps {
21            container('nginx') {
22              // Check the Nginx version
23              sh 'nginx -v'
24            }
25          }
26        }
27      }
28    }
29  }

```

try sample Pipeline...

☒ Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

REST API Jenkins 2.462.3

Careerflow Extension

3. Run the Pipeline

- Click **Build Now** to run the pipeline.
- Jenkins will create a pod in your Kubernetes cluster and run the Nginx container inside it.

4. Verify Deployment

To check if the pod was created successfully, run the following command in your terminal:

```
kubectl get pods -n jenkins
```

```
C:\Users\Administrator>kubectl get pods -n jenkins
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-pipeline-3-qf14m-lqzqg-1jvjd  2/2     Running   0          11s
```

You should see a pod with the Nginx container running.

References

- [Jenkins Official Website](#)
- [Kubernetes Official Documentation](#)
- [Minikube Documentation](#)