



Spring Boot Application Deployment in Kubernetes with Jenkins CI/CD Pipeline

Last Updated : 25 Jul, 2024

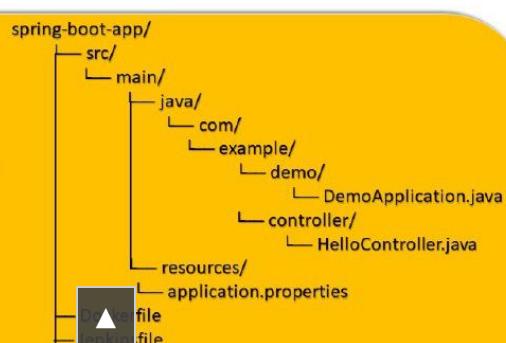


As we know in the modern world we use Continuous Integration and Continuous Deployment for fast and efficient delivery of our application to the market. Here in this Article, we are deep-diving into how we can actually deploy Spring boot applications on Kubernetes with the help of Jenkins CI/CD.

Project WorkFlow

- **.java files:** These files contain the Java source code for the application.
- **application.properties:** This file contains configuration properties for the application.
- **Dockerfile:** This file contains instructions for building a Docker image of the application.
- **Jenkinsfile:** This file contains instructions for building and deploying the application using Jenkins, a continuous integration and continuous delivery (CI/CD) tool.
- **deployment.yaml** and **service.yaml:** These files are YAML files that define how the application will be deployed in a Kubernetes cluster.
- **mvnw** and **mvnw.cmd:** These files are wrappers for the Maven commands that can be used to build, test, and package the application.
- **pom.xml:** This file is the project object model (POM) for the application. It contains information about the project, such as its dependencies and build configuration.
- **README.md:** This file is a markdown file that contains information about the project, such as how to set it up and how to use it.

Structure
of Spring
boot



Primary Terminologies

- **Spring boot:** It is free and open-source java framework which is built on the top of the spring framework. Their main benefits are Faster development, Improve the productivity and suitable for stand-alone application, production ready features and production ready features.
- **Node.js:** This is a free, open-source cross-platform and JavaScript runtime environment that executes JavaScript code outside the browser.
- **Jenkins:** It's also free, Open-source CI/CD tools that automate software development processes like Build, Test, and deployment.
- **Docker:** Docker is a platform that allows you to put all applications and their dependencies in standardized units called containers. These containers are lightweight and portable, allowing for continuity across environments (operating systems and hardware).
- **Kubernetes:** It is open-source containerized orchestration tool that basically uses it for ease of deployment, auto-scaling, auto-healing, and monitoring purposes.

GitHub: It is a free and open-source version control system which is developed by Microsoft. That provides,

- Storage for our codes
- Continuously track our code.
- the capability of sharing with another
- project collaboration support

Step-by-Step Guide For Spring boot application deployment in Kubernetes with Jenkins CI/CD pipeline

Step 1: Update System packages

```
sudo apt-get update
```

The reason for updating is that sometimes some packages of Docker are not working well, so for that reason we update them.



APT stands for Advanced Packaging Tool.

```
root@LAPTOP-GV2S1119:~$ sudo apt-get update
[sudo] password for root:
Get:1 http://archive.ubuntu.com/ubuntu jammy InRelease [129 kB]
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1517 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [259 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [1933 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [329 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [166 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-security/universe amd64 Metadata [16.8 kB]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [37.2 kB]
Get:13 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [7588 kB]
Get:14 http://security.ubuntu.com/ubuntu jammy-security/multiverse Translation-en [7588 kB]
Get:15 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [268 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:17 http://archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:18 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:19 http://archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [111 kB]
Get:20 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [6377 kB]
Get:21 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [10 kB]
Get:22 http://archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [10 kB]
Get:23 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [1996 kB]
Get:24 http://archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [338 kB]
Get:25 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1086 kB]
Get:26 http://archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [251 kB]
Get:27 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [22.1 kB]
Get:28 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [43.0 kB]
Get:29 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [10.7 kB]
Get:30 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [472 kB]
Get:31 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [167.1 kB]
Get:32 http://archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [167.1 kB]
Get:33 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 c-n-f Metadata [388 kB]
Get:34 http://archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 c-n-f Metadata [116 kB]
Get:35 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [27.2 kB]
Get:36 http://archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [16.3 kB]
Get:37 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [644 kB]
Get:38 http://archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 c-n-f Metadata [116 kB]
Fetched 31.8 MB in 13s (2373 kB/s)
Reading package lists... Done
root@LAPTOP-GV2S1119:~$
```

Step 2: creating spring-boot-application

Here First we are creating a directory that names spring-boot-app.



As per the structure of the spring boot application we are creating files and folder.

```
sudo mkdir spring-boot-app
```

after creation of spring-boot-app directory we are moving to inside it using below command.

```
cd spring-boot-app
```

```
romil@LAPTOP-GV2S1110:/home$ sudo mkdir spring-boot-app
romil@LAPTOP-GV2S1110:/home$ ls
romil@LAPTOP-GV2S1110:~/spring-boot-app$ three-tier-app
romil@LAPTOP-GV2S1110:/home/spring-boot-app$ ls
romil@LAPTOP-GV2S1110:/home/spring-boot-app$ git init
/home/spring-boot-app/.git: Permission denied
romil@LAPTOP-GV2S1110:/home/spring-boot-app$ sudo git init
hint: Using 'master' as the default branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:   git config --global init.defaultBranch <name>
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:   git branch -m <name>
Initialized empty Git repository in /home/spring-boot-app/.git/
romil@LAPTOP-GV2S1110:/home/spring-boot-app$
```

Then we are creating another directory as per the structure of the spring boot application diagram.

```
mkdir -p src/main/java/com/example/demo/controller
mkdir -p src/main/resources
```

after completion of the directory we check through "ls" command.

```
romil@LAPTOP-GV2S1110:/home/spring-boot-app$ sudo mkdir -p src/main/java/com/example/demo/controller
src
romil@LAPTOP-GV2S1110:/home/spring-boot-app$ ls
src
romil@LAPTOP-GV2S1110:/home/spring-boot-app$ sudo mkdir -p src/main/resources
src
romil@LAPTOP-GV2S1110:/home/spring-boot-app$ ls
src
romil@LAPTOP-GV2S1110:/home/spring-boot-app$ |
```

Create **DemoApplication.java** file inside the **demo** directory.



```
nano DemoApplication.java
```

we use "**sudo**" for super user permission. and "**nano**" for file creation instead of it you can also use "**vim**".

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```

After writing above scripts then Save and Exit nano:

- Press Ctrl + O to write out (save) the file.



- Press Enter to confirm the filename.
- Press Ctrl + X to exit nano.

Create **HelloController.java** file inside the **controller** directory.

```
nano HelloController.java
```

we use "**sudo**" for super user permission. and "**nano**" for file creation instead of it you can also use "**vim**".

```
package com.example.demo.controller;

import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {

    @GetMapping("/hello")
    public String sayHello() {
        return "Hello, World!";
    }
}
```

After writing above scripts then Save and Exit nano:

- Press Ctrl + O to write out (save) the file.
- Press Enter to confirm the filename.
- Press Ctrl + X to exit nano.

```
romil@LAPTOP-GV2S1119:/home/spring-boot-app$ sudo mkdir -p src/main/java/com/example/demo/controller
src
romil@LAPTOP-GV2S1119:/home/spring-boot-app$ ls
src
romil@LAPTOP-GV2S1119:/home/spring-boot-app$ sudo mkdir -p src/main/resources
src
romil@LAPTOP-GV2S1119:/home/spring-boot-app$ ls
src
romil@LAPTOP-GV2S1119:/home/spring-boot-app$ cd src/main/java/com/example/demo
romil@LAPTOP-GV2S1119:/home/spring-boot-app/src/main/java/com/example/demo$ sudo nano DemoApplication.java
romil@LAPTOP-GV2S1119:/home/spring-boot-app/src/main/java/com/example/demo$ sudo nano DemoApplication.java
romil@LAPTOP-GV2S1119:/home/spring-boot-app/src/main/java/com/example/demo$ cd controller
romil@LAPTOP-GV2S1119:/home/spring-boot-app/src/main/java/com/example/demo/controllers$ nano HelloController.java
romil@LAPTOP-GV2S1119:/home/spring-boot-app/src/main/java/com/example/demo/controllers$ sudo nano HelloController.java
romil@LAPTOP-GV2S1119:/home/spring-boot-app/src/main/java/com/example/demo/controllers$ |
```

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```



```
File Name to Write: HelloController.java
GNU nano 6.2
package com.example.demo.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {
    @GetMapping("/hello")
    public String sayHello() {
        return "Hello, World!";
    }
}
```



```
File Name to Write: HelloController.java
GNU nano 6.2
package com.example.demo.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {
    @GetMapping("/hello")
    public String sayHello() {
        return "Hello, World!";
    }
}
```

Create **application.properties** file inside the **resources** directory.

```
nano application.properties
```

we use "**sudo**" for super user permission. and "**nano**" for file creation instead of it you can also use "**vim**".

```
# Server configuration
server.port=8080

# Spring application name
spring.application.name=spring-boot-app

# Logging configuration
logging.level.org.springframework=INFO
logging.level.com.example.demo=DEBUG

# Custom properties (if any)
# custom.property.name=value
```

After writing above scripts then Save and Exit nano:

- Press Ctrl + O to write out (save) the file.
- Press Enter to confirm the filename.
- Press Ctrl + X to exit nano.

```
roamil@LAPTOP-GV2S1110:~/home/spring-boot-app/src/main/java/com/example/demo/controller$ cd ..
roamil@LAPTOP-GV2S1110:~/home/spring-boot-app/src/main/java/com/example/demo$ cd ..
roamil@LAPTOP-GV2S1110:~/home/spring-boot-app/src/main/java/com/example$ cd ..
roamil@LAPTOP-GV2S1110:~/home/spring-boot-app/src/main/java/com$ cd ..
roamil@LAPTOP-GV2S1110:~/home/spring-boot-app/src/main/java$ cd ..
roamil@LAPTOP-GV2S1110:~/home/spring-boot-app/src/main$ ls
java resources
roamil@LAPTOP-GV2S1110:~/home/spring-boot-app/src/main$ cd resources
roamil@LAPTOP-GV2S1110:~/home/spring-boot-app/src/main/resources$ sudo nano application.properties
roamil@LAPTOP-GV2S1110:~/home/spring-boot-app/src/main/resources$ |
```

```
GNU nano 6.2                                     application.properties *
# Server configuration
server.port=8080

# Spring application name
spring.application.name=spring-boot-app

# Logging configuration
logging.level.org.springframework=INFO
logging.level.com.example.demo=DEBUG

# Custom properties (if any)
# custom.property.name=value
```

File Name to Write: application.properties

^G Help ^D DOS Format ^A Append ^B Backup File
^C Cancel ^M Mac Format ^P Prepend ^T Browse

Step 3: Dockerfile Creation

Create Dockerfile inside the spring-boot-app directory.

```
sudo nano Dockerfile
```

we use "**sudo**" for super user permission. and "**nano**" for file creation instead of it you can also use "**vim**".

```
roamil@LAPTOP-GV2S1110:~/home/spring-boot-app$ ls
src
roamil@LAPTOP-GV2S1110:~/home/spring-boot-app$ sudo nano Dockerfile
```

Add configuration scripts in the **Dockerfile**.

```
# Use a base image with Java and a minimal Linux distribution
FROM openjdk:11-jdk-slim

# Set the working directory inside the container
WORKDIR /app

# Copy the Maven wrapper and source code into the container
COPY .mvn/ .mvn
COPY mvnw .
COPY pom.xml .

# Copy the application source code into the container
COPY src/ src/

# Expose the port your Spring Boot app is running on (default
# is 8080)
EXPOSE 8080

# Build the application
RUN ./mvnw clean package -DskipTests

# Copy the JAR file into the container
COPY target/spring-boot-app-0.0.1-SNAPSHOT.jar app.jar

# Define the command to run your Spring Boot application
CMD ["java", "-jar", "app.jar"]After writing above scripts
then Save and Exit nano:
```

- Press Ctrl + O to write out (save) the file.
- Press Enter to confirm the filename.
- Press Ctrl + X to exit nano.

```
GNU nano 6.2
# Use a base image with Java and a minimal Linux distribution
FROM openjdk:11-jdk-slim

# Set the working directory inside the container
WORKDIR /app

# Copy the Maven wrapper and source code into the container
COPY .mvn/ .mvn
```



```
COPY mvnw .  
COPY pom.xml .  
  
# Copy the application source code into the container  
COPY src/ src/  
  
# Expose the port your Spring Boot app is running on (default is 8080)  
EXPOSE 8080  
  
# Build the application  
RUN ./mvnw clean package -DskipTests  
  
# Copy the JAR file into the container  
COPY target/spring-boot-app-0.0.1-SNAPSHOT.jar app.jar  
  
# Define the command to run your Spring Boot application  
CMD ["java", "-jar", "app.jar"]
```



Step 4: Jenkinsfile Creation

Create Jenkinsfile inside the spring-boot-app directory.

```
sudo nano Jenkinsfile
```

we use "**sudo**" for super user permission. and "**nano**" for file creation instead of it you can also use "**vim**".

```
romil@LAPTOP-GV251119:/home/spring-boot-app$ ls  
src  
romil@LAPTOP-GV251119:/home/spring-boot-app$ sudo nano Dockerfile  
[sudo] password for romil:  
romil@LAPTOP-GV251119:/home/spring-boot-app$ ls  
Dockerfile src  
romil@LAPTOP-GV251119:/home/spring-boot-app$ sudo nano Jenkinsfile  
[sudo] password for romil:
```

Add pipeline configuration scripts in the Jenkinsfile.

```
pipeline {  
    agent any  
  
    environment {  
        DOCKER_IMAGE = "romilbhai/spring-boot-app"  
        K8S_NAMESPACE = "springboot-demo"  
        DOCKER_CREDENTIALS_ID = 'dockerhub-credentials-id' //  
        Define Docker Hub credentials ID  
        KUBECONFIG_CREDENTIALS_ID = 'kubeconfig-springboot'  
        // Define Kubernetes config credentials ID
```

```
}

stages {
    stage('Checkout') {
        steps {
            script {
                checkout scm
            }
        }
    }

    stage('Build and Test') {
        steps {
            script {
                sh "mvn clean package"
            }
        }
    }

    stage('Build Docker Image') {
        steps {
            script {
                sh "docker build -t romilbhai/spring-boot-app:${env.BUILD_ID} ."
                sh "docker tag romilbhai/spring-boot-app:${env.BUILD_ID} romilbhai/spring-boot-app:latest"
            }
        }
    }

    stage('Push to dockerhub') {
        steps {
            script {
                withCredentials([string(credentialsId: 'dockerhub-password', variable: 'DOCKER_PASSWORD')]) {
                    sh """
                        echo \$DOCKER_PASSWORD | docker login -u romilbhai --password-stdin
                    """
                    sh "docker push romilbhai/spring-boot-app:${env.BUILD_ID}"
                    sh "docker push romilbhai/spring-boot-app:latest"
                }
            }
        }
    }
}
```

```

        }
    }

}

stage('Deploy to Kubernetes') {
    steps {
        script {

            kubeconfig(credentialsId: 'kubeconfig')

{

                sh 'kubectl apply -f k8s/
deployment.yaml -n ${env.K8S_NAMESPACE}'
                sh 'kubectl apply -f k8s/service.yaml
-n ${env.K8S_NAMESPACE}'
            }

        }
    }
}

post {
    always {
        cleanWs()
    }
    success {
        echo 'Build and deployment successful!'
    }
    failure {
        echo 'Build or deployment failed.'
    }
}
}

```

After writing above scripts then Save and Exit nano:

- Press Ctrl + O to write out (save) the file.
- Press Enter to confirm the filename.
- Press Ctrl + X to exit nano.



```

environment {
    DOCKER_IMAGE = "romilbhai/spring-boot-app"
    K8S_NAMESPACE = "springboot-demo"
    DOCKER_CREDENTIALS_ID = 'dockerhub-credentials-id' // Define Docker Hub credentials ID
    KUBECONFIG_CREDENTIALS_ID = 'kubecfg-springboot' // Define Kubernetes config credentials ID
}

stages {
    stage('Checkout') {
        steps {
            script {
                checkout scm
            }
        }
    }
    stage('Build and Test') {
        steps {
            script {
                sh "mvn clean package"
            }
        }
    }
    stage('Build Docker Image') {
        steps {
            script {
                sh "docker build -t romilbhai/spring-boot-app:${env.BUILD_ID} ."
                sh "docker tag romilbhai/spring-boot-app:${env.BUILD_ID} romilbhai/spring-boot-app:latest"
            }
        }
    }
    stage('Push to dockerhub') {
        steps {
            script {
                withCredentials([string(credentialsId: 'dockerhub-password', variable: 'DOCKER_PASSWORD')]) {
                    sh """
                        echo \$DOCKER_PASSWORD | docker login -u romilbhai --password-stdin
                    """
                }
            }
        }
    }
}

```

File Name to Write: Jenkinsfile
 ⌘ Help ⌘ Cancel ⌘ DOS Format ⌘ Mac Format ⌘ Append ⌘ Prepend ⌘ Backup File ⌘ Browse

Step 5: Kubernetes files creation

Then we are creating another directory as per the structure of the spring boot application diagram.

```
sudo mkdir k8s
```

after creation of "k8s" directory we are moving to inside it using below command.

```
cd k8s
```

```

romil@LAPTOP-GV25110:~/home/spring-boot-app$ ls
src
romil@LAPTOP-GV25110:~/home/spring-boot-app$ sudo nano Dockerfile
[sudo] password for romil:
romil@LAPTOP-GV25110:~/home/spring-boot-app$ ls
Dockerfile src
romil@LAPTOP-GV25110:~/home/spring-boot-app$ sudo nano Jenkinsfile
[sudo] password for romil:
romil@LAPTOP-GV25110:~/home/spring-boot-app$ ls
Dockerfile Jenkinsfile src
romil@LAPTOP-GV25110:~/home/spring-boot-app$ sudo mkdir k8s
romil@LAPTOP-GV25110:~/home/spring-boot-app$ cd k8s
romil@LAPTOP-GV25110:~/home/spring-boot-app/k8s$ 

```

Create deployment file inside the k8s directory.

```
sudo nano deployment.yaml
```

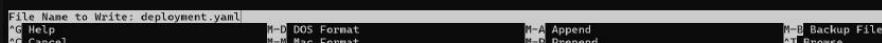
we use "**sudo**" for super user permission. and "**nano**" for file creation instead of it you can also use "**vim**".



Add deployment configuration scripts in the "deployment.yaml" file.

```
GNU nano 6.2                                         deployment.yaml *
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-boot-app
  namespace: springboot-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: spring-boot-app
  template:
    metadata:
      labels:
        app: spring-boot-app
    spec:
      containers:
        - name: spring-boot-app
          image: romilbhai/springboot-demo:latest
          ports:
            - containerPort: 8080
```



After writing above scripts then Save and Exit nano:

- Press Ctrl + O to write out (save) the file.
- Press Enter to confirm the filename.
- Press Ctrl + X to exit nano.

Create service file inside the k8s directory.

```
sudo nano service.yaml
```

```
romil@LAPTOP-GV251119:/home/spring-boot-app$ ls
src
romil@LAPTOP-GV251119:/home/spring-boot-app$ sudo nano Dockerfile
[sudo] password for romil:
romil@LAPTOP-GV251119:/home/spring-boot-app$ ls
Dockerfile src
romil@LAPTOP-GV251119:/home/spring-boot-app$ sudo nano Jenkinsfile
[sudo] password for romil:
romil@LAPTOP-GV251119:/home/spring-boot-app$ ls
Dockerfile Jenkinsfile src
romil@LAPTOP-GV251119:/home/spring-boot-app$ sudo mkdir k8s
romil@LAPTOP-GV251119:/home/spring-boot-app$ cd k8s
romil@LAPTOP-GV251119:/home/spring-boot-app/k8s$ sudo nano deployment.yaml
romil@LAPTOP-GV251119:/home/spring-boot-app/k8s$ sudo nano service.yaml
```

we use "**sudo**" for super user permission. and "**nano**" for file creation instead of it you can also use "**vim**".

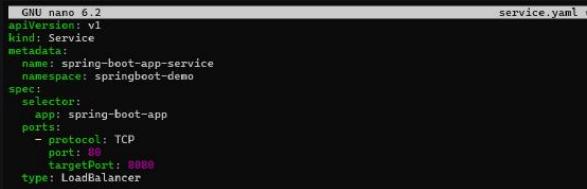
Add service configuration scripts in the "**service.yaml**" file.

```
apiVersion: v1
kind: Service
```

```

metadata:
  name: spring-boot-app-service
  namespace: springboot-demo
spec:
  selector:
    app: spring-boot-app
ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
type: LoadBalancer

```



```

  GNU nano 6.2
  apiVersion: v1
  kind: Service
  metadata:
    name: spring-boot-app-service
    namespace: springboot-demo
  spec:
    selector:
      app: spring-boot-app
    ports:
      - protocol: TCP
        port: 80
        targetPort: 8080
  type: LoadBalancer

```

After writing above scripts then Save and Exit nano:

- Press Ctrl + O to write out (save) the file.
- Press Enter to confirm the filename.
- Press Ctrl + X to exit nano.

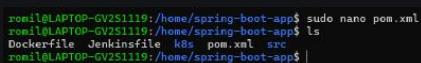
Step 6: POM.xml file creation

Create pom.xml file inside the spring-boot-app directory.

```
sudo nano pom.xml
```

we use "**sudo**" for super user permission. and "**nano**" for file creation instead of it you can also use "**vim**".

after completion of the directory we check through "**ls**" command.



```

root@LAPTOP-GV251110:/home/spring-boot-app$ sudo nano pom.xml
root@LAPTOP-GV251110:/home/spring-boot-app$ ls
Dockerfile Jenkinsfile k8s pom.xml src
root@LAPTOP-GV251110:/home/spring-boot-app$ |

```

Add xml code in the POM.xml file.

```
<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
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>spring-boot-app</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>spring-boot-app</name>
  <description>Demo project for Spring Boot</description>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.5.4</version>
    <relativePath/> <!-- lookup parent from repository -->
  >
  </parent>

  <properties>
    <java.version>11</java.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
```

```

        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</
artifactId>
            </plugin>
        </plugins>
    </build>

</project>

```

```

GNU nano 6.2                               pom.xml. *
<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 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>spring-boot-app</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>
    <name>spring-boot-app</name>
    <description>Demo project for Spring Boot</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.5.4</version>
        <relativePath> <!-- lookup parent from repository -->
    </parent>

    <properties>
        <java.version>11</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

```

File Name to Write: pom.xml
 ⌘ Help M-D DOS Format M-A Append
 ⌘ C Cancel M-N Mac Format M-P Prepend
 ⌘ B Backup File M-B Browse

After writing above scripts then Save and Exit nano:

- Press Ctrl + O to write out (save) the file.
- Press Enter to confirm the filename.
- Press Ctrl + X to exit nano.

Step 7: Building and running Docker image.

```

docker build -t spring-boot-app:latest .
docker run -d -it spring-boot-app:latest

```

Here, ('-t') stands for tag, which basically allows you to give a name or tag with a Docker image. In this scenario, spring-boot-app:latest is the tag or name that we want to build. and

(' -i ') stands for interactive and basically allows us to interact with command-

line. It keeps STDIN (standard input) open even if not attached.

If we combine ('-it'), that means that it allows you to run a tagged or named container with an interactive command interface.

Once you write the build command, the image will be created, and through this image, a Docker container will run.

You can check your container working status through 'Docker -ps' command.

('-'d') stands for daemon that run process in the background.

```
ronil@LAPTOP-GV251110:~/home/spring-boot-app$ sudo nano pom.xml
ronil@LAPTOP-GV251110:~/home/spring-boot-app$ ls
Dockerfile Jenkinsfile pom.xml src
ronil@LAPTOP-GV251110:~/home/spring-boot-app$ docker build -t spring-boot-app:latest .
failed to fetch metadata: /var/empty/nut/local/lib/docker/cli-plugins/docker-buildx: no such file or directory
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/
Sending build context to Docker daemon 61.95kB
Step 1/9 : FROM maven:3.8.4-openjdk-11 AS builder
--> 6b42b260678c
Step 2/9 : COPY . /app
--> Using cache
--> db607b522e9e
Step 3/9 : COPY pom.xml .
--> 3c4f3dd1a8d6
Step 4/9 : COPY src ./src
--> cdd6adaleaf
Step 5/9 : RUN mvn clean package -DskipTests
--> Running in 34166a4960bc0
[INFO] Scanning for projects...
Downloading from central: https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-starter-parent/2.5.4/spring-boot-starter-parent-2.5.4.pom
m
Downloaded from central: https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-starter-parent/2.5.4/spring-boot-starter-parent-2.5.4.pom (8.6 kB at 21 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-dependencies/2.5.4/spring-boot-dependencies-2.5.4.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-dependencies/2.5.4/spring-boot-dependencies-2.5.4.pom (18.9 kB at 1.0 MB/s)
Downloading from central: https://repo.maven.apache.org/maven2/com/databricks/java-driver-bom/4.11.3/java-driver-bom-4.11.3.pom
Downloaded from central: https://repo.maven.apache.org/maven2/com/databricks/java-driver-bom/4.11.3/java-driver-bom-4.11.3.pom (4.1 kB at 99 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/io/dropwizard/metrics/metrics-bom/4.1.25/metrics-bom-4.1.25.pom
Downloaded from central: https://repo.maven.apache.org/maven2/io/dropwizard/metrics/metrics-bom/4.1.25/metrics-bom-4.1.25.pom (5.3 kB at 106 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/io/dropwizard/metrics/metrics-parent/4.1.25/metrics-parent-4.1.25.pom
Downloaded from central: https://repo.maven.apache.org/maven2/io/dropwizard/metrics/metrics-parent/4.1.25/metrics-parent-4.1.25.pom (17 kB at 458 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/groovy/groovy-bom/3.0.8/groovy-bom-3.0.8.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/groovy/groovy-bom/3.0.8/groovy-bom-3.0.8.pom (26 kB at 550 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/infinispan/infinispan-bom/12.1.7.Final/infinispan-bom-12.1.7.Final.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/infinispan/infinispan-build-configuration-parent/12.1.7.Final/infinispan-build-configuration-parent-12.1.7.Final.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/infinispan/infinispan-build-configuration-parent/12.1.7.Final/infinispan-build-configuration-parent-12.1.7.Final.pom (14 kB at 329 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/jboss/jboss-parent/36/jboss-parent-36.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/jboss/jboss-parent/36/jboss-parent-36.pom (66 kB at 1.4 MB/s)
```

Step 8: Setup Kubernetes

installing latest minikube stable x86-64 Linux variant, Installation as per your requirement.

Follow Minikube for more detailed guidance.

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube &&
rm minikube-linux-amd64
```

```
minikube start
```

```
ronil@LAPTOP-GV251110:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64
% Total % Received % Xferd Average Speed Time Time Current
          Dload Upload Total Spent Left Speed
100 91.1M 100 91.1M 0      0  3530k 0  0:00:26 0:00:26 ---:--- 3666k
[sudo] password for ronil:
ronil@LAPTOP-GV251110:~$ minikube status
✖ Profile "minikube" not found. Run "minikube profile list" to view all profiles.
👉 To start a cluster, run: "minikube start"
ronil@LAPTOP-GV251110:~$ minikube start
(minikube) Starting on Ubuntu 22.04 (amd64)
Automatic selection of Docker driver. Other choices: none, ssh
👉 Using Docker driver with root privileges.
For an improved experience it's recommended to use Docker Engine instead of Docker Desktop.
Docker Engine installation instructions: https://docs.docker.com/engine/install/#server
👉 Starting "minikube" primary control-plane node in "minikube" cluster
🔍 Pulling base image v0.44 ...
💡 Downloading Kubernetes v1.38.0 preload ...
> preloaded-images-k8s-v18-v1...: 342.90 MiB / 342.90 MiB 100.00% 2.19 Mi
> gcr.io/k8s-minikube/kicbase...: 481.58 MiB / 481.58 MiB 100.00% 2.15 Mi
Creating docker container (CPUs=2, Memory=2208MiB) ...
💡 Preparing kubelet v1.38.0 on Docker 26.1.1 ...
👉 Generating certificates and keys ...
👉 Booting up control plane ...
👉 Configuring RBAC rules
🔗 Configuring bridge CNI (Container Networking Interface) ...
```

```
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
root@LAPTOP-GV2S1119:~$
```

Step 9: Creating namespace

```
kubectl create namespace springboot-demo
```

Here we are creating a namespace called springboot-demo because otherwise namespace is default.

```
Restarting existing docker container for "minikube" ...
Preparing Kubernetes v1.38.0 on Docker 26.1.1 ...
Verifying Kubernetes components...
  * Using image docker.io/kubernetesui/dashboard:v2.7.0
  * Using image docker.io/kubernetesui/metrics-scraper:v1.6.8
  * Using image gcr.io/k8s-minikube/storage-provisioner:v5
  * Some dashboard features require the metrics-server addon. To enable all features please run:
    minikube addons enable metrics-server

  * Enabled addons: default-storageclass, storage-provisioner, dashboard
  * Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
root@LAPTOP-GV2S1119:~$ kubectl status
error: unknown command "status" for "kubectl"
root@LAPTOP-GV2S1119:~$ minikube status
minikube
  type: Control Plane
  host: Running
  kubelet: Running
  apiserver: Running
  kubeconfig: Configured

root@LAPTOP-GV2S1119:~$ kubectl apply -f deployment.yaml
error: the path "deployment.yaml" does not exist
root@LAPTOP-GV2S1119:~$ cd ..
root@LAPTOP-GV2S1119:~/home$ cd spring-boot-app
root@LAPTOP-GV2S1119:~/home/spring-boot-app$ ls
Dockerfile Jenkinsfile k8s pom.xml src
root@LAPTOP-GV2S1119:~/home/spring-boot-app$ kubectl get pods
root@LAPTOP-GV2S1119:~/home/spring-boot-app$ ls
deployment.yaml service.yaml
root@LAPTOP-GV2S1119:~/home/spring-boot-app$ kubectl apply -f deployment.yaml
Error from server (NotFound): error when creating "deployment.yaml": namespaces "springboot-demo" not found
root@LAPTOP-GV2S1119:~/home/spring-boot-app$ kubectl get namespaces
NAME          STATUS   AGE
default       Active   10d
kube-node-lease Active   10d
kube-public    Active   10d
kube-system    Active   10d
kubernetes-dashboard Active   10d
root@LAPTOP-GV2S1119:~/home/spring-boot-app$ kubectl create namespace
error: exactly one NAME is required; got 0
See 'kubectl create namespace -h' for help and examples
root@LAPTOP-GV2S1119:~/home/spring-boot-app$ kubectl create namespace springboot-demo
namespace/springboot-demo created
root@LAPTOP-GV2S1119:~/home/spring-boot-app$
```

Step 9: Checking deployment in local

```
kubectl apply -f deployment.yaml -n springboot-demo
kubectl apply -f service.yaml -n springboot-demo
```

Write above command that are tailored for deployment with specific namespace.

```
kubectl get pods -n springboot-demo
```

once you done deployment after check then you will get output like,

Ready



0/1
0/1
0/1
0/1

Here 0/1 means 0 pods are running out of 1. and it will show 4 times due to replication for disaster recovery.

Then write below command,

```
eval $(minikube docker-env)
```

And check again through below command

```
kubectl get pods -n springboot-demo
```

now you will get 1/1 ready status like,

Ready
1/1
1/1
1/1
1/1

```
spec:
  containers:
  - name: spring-boot-app
    image: romilbhai/springboot-demo:latest
    imagePullPolicy: IfPresent
    ports:
      - containerPort: 8080
romil@LAPTOP-GV251119:/home/spring-boot-app/k8s$ sudo nano deployment.yaml
[sudo] password for romil:
romil@LAPTOP-GV251119:/home/spring-boot-app/k8s$ kubectl apply -f deployment.yaml -n springboot-demo
The Deployment "spring-boot-app" is invalid: spec.template.spec.containers[0].imagePullPolicy: Unsupported value: "IfPresent"; supported values: "Always", "IfNotPresent", "Never"
romil@LAPTOP-GV251119:/home/spring-boot-app/k8s$ sudo nano deployment.yaml
romil@LAPTOP-GV251119:/home/spring-boot-app/k8s$ kubectl apply -f deployment.yaml -n springboot-demo
The Deployment "spring-boot-app" is invalid: spec.template.spec.containers[0].imagePullPolicy: Unsupported value: "Always"; supported values: "Always", "IfNotPresent", "Never"
romil@LAPTOP-GV251119:/home/spring-boot-app/k8s$ sudo nano deployment.yaml
romil@LAPTOP-GV251119:/home/spring-boot-app/k8s$ kubectl apply -f deployment.yaml -n springboot-demo
deployment.apps/spring-boot-app configured
romil@LAPTOP-GV251119:/home/spring-boot-app/k8s$ kubectl apply -f service.yaml -n springboot-demo
service/spring-boot-app-service unchanged
romil@LAPTOP-GV251119:/home/spring-boot-app/k8s$ kubectl get pods -n springboot-demo
NAME          READY   STATUS    RESTARTS   AGE
spring-boot-app-57958497fb-qswq5  0/1   ErrImageNeverPull  0   32m
spring-boot-app-68c56fd7c9-jxrhx  0/1   ImagePullBackOff  0   4d
spring-boot-app-6cd096949-qmgbk  0/1   ImagePullBackOff  0   1d
spring-boot-app-748484948-qjw4t  0/1   ImagePullBackOff  0   46m
romil@LAPTOP-GV251119:/home/spring-boot-app/k8s$ sudo nano deployment.yaml
romil@LAPTOP-GV251119:/home/spring-boot-app/k8s$ kubectl apply -f deployment.yaml -n springboot-demo
```

```

deployment:app$spring-boot-app configured
romil@LAPTOP-GV2S1110:~/home/spring-boot-app$ kubectl get pods -n springboot-demo
NAME          READY   STATUS    RESTARTS   AGE
spring-boot-app-57958497fb-qswq5  0/1   ErrImageNeverPull  0        33m
spring-boot-app-68c56fd7c9-jrxh  0/1   ImagePullBackOff  0        45m
spring-boot-app-6c88969d4u-gmqlb  0/1   InvalidImageName  0        99s
spring-boot-app-6fd9695845-xfvf6  0/1   ContainerCreating  0        35s
romil@LAPTOP-GV2S1110:~/home/spring-boot-app$ kubectl eval $(minikube docker-env)
romil@LAPTOP-GV2S1110:~/home/spring-boot-app$ kubectl get pods -n springboot-demo
NAME          READY   STATUS    RESTARTS   AGE
spring-boot-app-6fd9695845-xfvf6  1/1   Running  0        2m1s
spring-boot-app-6fd9695845-xwmd7  1/1   Running  0        72s
spring-boot-app-6fd9695845-zlwpq  1/1   Running  0        60s
romil@LAPTOP-GV2S1110:~/home/spring-boot-app$ sudo nano deployment.yaml
romil@LAPTOP-GV2S1110:~/home/spring-boot-app$ 

```

then push code in your remote repository or (GitHub repository).

```

* master
romil@LAPTOP-GV2S1110:~/home/spring-boot-app$ sudo git -d checkout
unknown option: -d
usage: git [<--version] [<--help>] [<C <path>] [<c <name>=<value>]
           [<commit>|<tag>|<branch>] [<--html-path>] [<--man-path>] [<--info-path>
           [<p>|<annotate> | <p>|<replace>|<replace-objects>|<bare>
           [<git-dir>|<path>] [<work-tree>|<path>] [<namespace>=<name>]
           [<super-prefix>|<path>] [<config-env>=<name>=<envvar>
           [<command> [<args>]
romil@LAPTOP-GV2S1110:~/home/spring-boot-app$ sudo git branch -d checkout
Deleted branch checkout (was 5465f39).
romil@LAPTOP-GV2S1110:~/home/spring-boot-app$ sudo git branch
* master
romil@LAPTOP-GV2S1110:~/home/spring-boot-app$ git remote add origin https://github.com/RomilMovaliya/spring-boot-app.git
fatal: detected dubious ownership in repository at '/home/spring-boot-app'
To add an exception for this directory, call:
git config --global --add safe.directory /home/spring-boot-app
romil@LAPTOP-GV2S1110:~/home/spring-boot-app$ sudo git config --global --add safe.directory /home/spring-boot-app
romil@LAPTOP-GV2S1110:~/home/spring-boot-app$ sudo git push -u origin main
Username for 'https://github.com': RomilMovaliya
Password for 'https://RomilMovaliya@github.com':
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 8 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (19/19), 2.80 KiB | 1.40 MiB/s, done.
Total 19 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/RomilMovaliya/spring-boot-app.git
 * [new branch]  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
romil@LAPTOP-GV2S1110:~/home/spring-boot-app$ sudo git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
romil@LAPTOP-GV2S1110:~/home/spring-boot-app$ sudo git remote -v
origin  https://github.com/RomilMovaliya/spring-boot-app.git (fetch)
origin  https://github.com/RomilMovaliya/spring-boot-app.git (push)
romil@LAPTOP-GV2S1110:~/home/spring-boot-app$ 

```

Step 10: Jenkins installation

```

sudo apt update
sudo apt upgrade

```

update: It is basically used for refreshing list of available software packages and versions but it not download or install anything new by itself.

upgrade: It is installing newer version of software packages that are available on your system.

```

romil@LAPTOP-GV2S1110:~/nodejs-app$ cd ..
romil@LAPTOP-GV2S1110:~$ sudo apt-get update
[sudo] password for romil:
Hit:1 http://archive.ubuntu.com/ubuntu jammy-security InRelease
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
romil@LAPTOP-GV2S1110:~$ sudo apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages will be installed:
  libcurl4-openssl-dev
The following packages have been kept back:
  python3-update-manager update-manager-core
The following packages will be upgraded:
  apt apt-utils base-files bash bind9-dnsutils bind9-host bind9-libs binutils binutils-common binutils-x86_64-linux-gnu bsdxtrautils bsduutils coreutils
  curl curl-distro-info-data dpgk ejct git git-man iptables irqbalance less libaptg6.8 libbinutis libblkid libc-bin libc6
  libcryptsetup1 liblbcf0 libcurl3-gnutls libcurl4 libexpat1 libglib2.0-0 libglib2.0-0-bin libglib2.0-data libgnutls38 libip4tc2 libip6tc2
  libidn2-2.5.0 libltdl-common libmount libhttpng2-14 libsys-system libpam-modules libpam-modules-bin libpam-runtime libpam-system libpam0g libperl5.34
  libpython3.10 libpython3.10-minimal libpython3.10-stdeb libsmartsconf libsqlite3-0 libssh-4 libssl3 libsystemd libudev libuv libxml2
  libxkbcommon locales msttcorefonts libxkbcommon0 libxkbcommon0d libxkbcommon-dictionaries python3-software-properties python3.10 python3.10-minimal snapd software-properties-common
  systemd-systemd libsystemd libsystemd-sysv libsystemd-timesyncd tar libpomp tdata ubuntu-advantage-tools ubuntu-pro-client lib0n ubuntu-release-upgrader-core
  udev util-linux uild-runtime vim vim-common vim-runtime vim-tiny xxd
193 upgraded, 1 newly installed, 0 to remove and 2 not upgraded.
58 standard LTS security updates
Need to get 97.9 MB of archives.
After this operation, 1765 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 motd-news-config all 12ubuntu4.6 [4352 B]
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libc 2.35-0ubuntu3.1 [238 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libcurl3 7.61.1-0ubuntu1.4 [178 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 bash 5.1-1ubuntu1.1 [769 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 bsdtar 5.1.2-37.2-4ubuntu3.4 [88.9 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 coreutils 8.32-4.1ubuntu1.2 [1437 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 liblens-systemd amd64 249.11-0ubuntu3.12 [133 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libsystemd amd64 249.11-0ubuntu3.12 [319 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 systemd-timesyncd amd64 249.11-0ubuntu3.12 [31.2 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 systemd-sysv amd64 249.11-0ubuntu3.12 [10.5 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libpam-systemd amd64 249.11-0ubuntu3.12 [283 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 systemd amd64 249.11-0ubuntu3.12 [4581 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 udev amd64 249.11-0ubuntu3.12 [1587 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libudev1 amd64 249.11-0ubuntu3.12 [78.2 kB]

```



```

Setting previously unselected package net-tools.
(Reading database ... 27426 files and directories currently installed.)
Preparing to unpack .../net-tools_1.60+git20181103.0eebece~ubuntu5_amd64.deb ...
Unpacking net-tools (1.60+git20181103.0eebece~ubuntu5) ...
Selecting previously unselected package jenkins.
Preparing to unpack .../archives/jenkins_2.462_all.deb ...
Unpacking jenkins (2.462) ...
Setting up net-tools (1.60+git20181103.0eebece~ubuntu5) ...
Setting up jenkins (2.462) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /lib/systemd/system/jenkins.service.
Processing triggers for man-db (2.18.2-1) ...
root@BLAPTOP-GV2S1119:~$ 

The following additional packages will be installed:
  net-tools
This package contains NOC packages will be installed:
  jenkins net-tools
0 upgraded, 2 newly installed, 0 to remove and 2 not upgraded.
Need to get 91.4 MB of additional disk space will be used.
After this operation, 94.1 MB of additional disk space will be used.
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 net-tools amd64 1.60+git20181103.0eebece~ubuntu5 [204 kB]
Get:1 https://pkg.jenkins.io/debian binary/ jenkins 2.462 [91.2 MB]
Fetched 91.4 MB in 36s (2541 kB/s)
Selecting previously unselected package net-tools.
(Reading database ... 27426 files and directories currently installed.)
Preparing to unpack .../net-tools_1.60+git20181103.0eebece~ubuntu5_amd64.deb ...
Unpacking net-tools (1.60+git20181103.0eebece~ubuntu5) ...
Selecting previously unselected package jenkins.
Preparing to unpack .../archives/jenkins_2.462_all.deb ...
Unpacking jenkins (2.462) ...
Setting up net-tools (1.60+git20181103.0eebece~ubuntu5) ...
Setting up jenkins (2.462) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /lib/systemd/system/jenkins.service.
Processing triggers for man-db (2.18.2-1) ...
root@BLAPTOP-GV2S1119:~$ sudo systemctl start jenkins
root@BLAPTOP-GV2S1119:~$ sudo systemctl enable jenkins
Syncing state of jenkins.service with current configuration...
Executing: /lib/systemd/systemctl enable jenkins
root@BLAPTOP-GV2S1119:~$ sudo systemctl status jenkins
● Jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2024-06-15 18:51:43 IST; 3min 21s ago
     Main PID: 9542 (java)
       Tasks: 53 (limit: 4564)
      Memory: 769.1M
        CGroup: /system.slice/jenkins.service
           └─ 9542 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

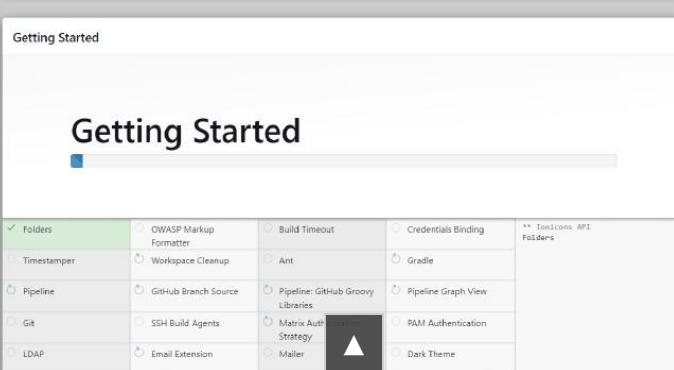
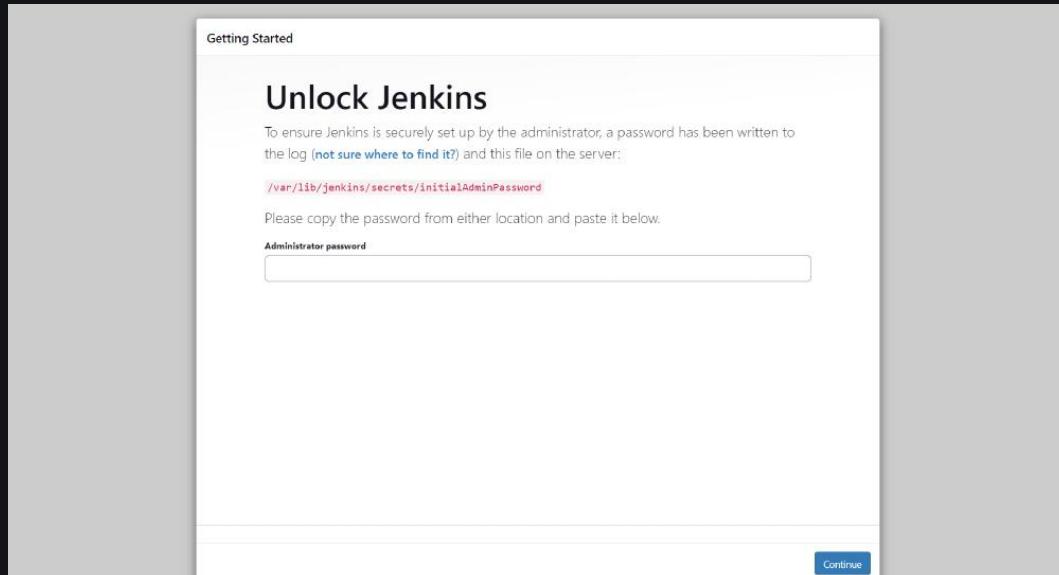
Jun 15 18:51:27 LAPTOP-GV2S1119 jenkins[9542]: c68bfef68f1e9b39115afe77b15ed5
Jun 15 18:51:37 LAPTOP-GV2S1119 jenkins[9542]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Jun 15 18:51:37 LAPTOP-GV2S1119 jenkins[9542]: ****
Jun 15 18:51:37 LAPTOP-GV2S1119 jenkins[9542]: ****
Jun 15 18:51:37 LAPTOP-GV2S1119 jenkins[9542]: ****
Jun 15 18:51:43 LAPTOP-GV2S1119 jenkins[9542]: 2024-06-15 13:21:43.035+0000 [id:31] INFO jenkins.InitReactorRunner$1#onAttained: Completed in
Jun 15 18:51:43 LAPTOP-GV2S1119 jenkins[9542]: 2024-06-15 13:21:43.069+0000 [id:24] INFO hudson.Lifecycle.LifecycleOnReady: Jenkins is fully
Jun 15 18:51:43 LAPTOP-GV2S1119 systemd[1]: Started Jenkins Continuous Integration Server.
Jun 15 18:51:43 LAPTOP-GV2S1119 jenkins[9542]: 2024-06-15 13:21:43.978+0000 [id:61] INFO h.m.DownloadService$Downloadable#load: Obtained the
Jun 15 18:51:43 LAPTOP-GV2S1119 jenkins[9542]: 2024-06-15 13:21:43.971+0000 [id:61] INFO hudson.util.Retrier#start: Performed the action chec
[Lines 1-19/19 (END)]
```

once you write above command then open the browser and write,

<https://localhost:8080>

Now Jenkins UI(First page) will display but it require the initialAdminPassword so that reason you need to write below command that will show the password.

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





Step 11: Configuring Jenkins

Once you done personal login or as a guest login, you will see the Jenkins Dashboard In that We need to install some necessary plugins: Go to Manage Jenkins > Manage Plugins. And installing following plugins:

- Docker Pipeline
- Kubernetes
- Git

The screenshot shows the Jenkins 'Manage Jenkins' page. On the left, there's a sidebar with options like 'New Item', 'Build History', 'Manage Jenkins' (which is selected), and 'My Views'. The main area has a heading 'Manage Jenkins' and a note about Java 11 end-of-life. It includes sections for 'System Configuration' (with links to 'System', 'Tools', 'Nodes', 'Clouds', and 'Appearance'), 'Security', 'Credentials', and 'Credential Providers'. At the bottom, there's a search bar for 'Plugins' and a list of available plugins including 'Git server', 'git', and 'Docker Pipeline'.

The screenshot shows the Jenkins Marketplace interface. At the top, there are tabs for Pipeline, DevOps, Deployment, and Docker. Below the tabs, a search bar contains the text "Kubernetes". A message says "Build and use Docker containers from pipelines." There is a checked checkbox next to "Kubernetes". Below the checkbox, it says "4246 v5a.12b.1fe120e" and "Cloud Providers Cluster Management Kubernetes Agent Management". It also states "This plugin integrates Jenkins with Kubernetes". To the right, it says "2 days ago". Below this, there are several other plugin entries: "Kubernetes Client API" (version 6.10.0-240.v57880ce8b.0b.2, 4 mo 20 days ago), "Kubernetes Credentials" (version 174.va.3Ge093562d9, 18 days ago), "Kubernetes CLI" (version 1.12.1, 9 mo 20 days ago), and "Kubernetes Credentials Provider" (version 1.262.v2670cf7ea.0c5, 9 mo 20 days ago).

Step 12: Create a New Pipeline Job

Create New Item:

Create a new job or new item from left-hand side menu on the Jenkins Dashboard.

Enter Job Details:

Name: Enter a name for your new job (e.g., springboot-app-Pipeline).

- Type: Select Pipeline.
- Click OK to proceed.

The screenshot shows the Jenkins "New Item" dialog. At the top, it says "Dashboard > All > New Item". The main title is "New Item". Below it, a field says "Enter an item name" with "springboot-app" typed in. A section titled "Select an item type" lists five options: "Freestyle project" (selected), "Pipeline" (disabled), "Multi-configuration project" (disabled), "Folder" (disabled), and "Multibranch Pipeline" (disabled). At the bottom right is a blue "OK" button.

Step 13: Configure the Job

General Configuration:

Description: Optionally, provide a description for the job to explain what this pipeline does.

The screenshot shows the Jenkins configuration page for the "springboot-app" job. At the top, it says "Dashboard > springboot-app > Configuration". The configuration tab is selected. Below it, there are sections for "Configure" and "General". In the "General" section, there is a large green "Enabled" switch with a blue circle. To the right of the switch is a small circular icon with a white triangle pointing up.

The screenshot shows the Jenkins Pipeline configuration page under the 'General' tab. It includes a description field with the text 'here i creating pipeline for spring-boot application deployment on jenkins.', a 'Plain text' link, and a 'Preview' button. Below are several checkboxes for pipeline behavior: 'Discard old builds', 'Do not allow concurrent builds', 'Do not allow the pipeline to resume if the controller restarts', 'GitHub project', 'Pipeline speed/durability override', 'Preserve stashes from completed builds', 'This project is parameterized', and 'Throttle builds'. At the bottom are 'Save' and 'Apply' buttons.

Pipeline Configuration:

- **Definition:** Choose Pipeline script from SCM.
- **SCM:** Select Git.
- **Repository URL:** Enter your Git repository URL (e.g., <https://github.com/your-username/your-repo.git>).

The screenshot shows the Jenkins Pipeline configuration page under the 'Pipeline' tab. In the 'Definition' section, 'Pipeline script from SCM' is selected. Under 'SCM', 'Git' is chosen. In the 'Repositories' section, the 'Repository URL' is set to <https://github.com/RomilMavalia/spring-boot-app.git>. The 'Credentials' dropdown is set to '- none -'. There is an 'Add' button and an 'Advanced' dropdown. At the bottom are 'Save' and 'Apply' buttons.

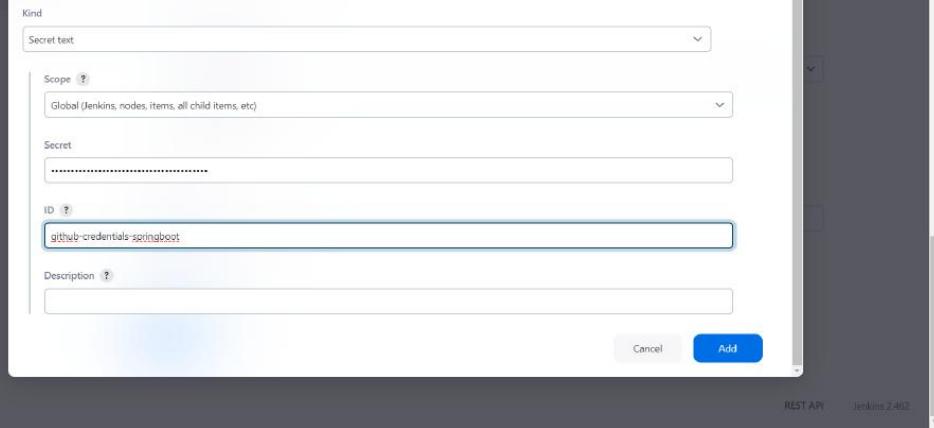
Credentials:

If your repository requires authentication, you need to add credentials.

- Credentials:** Click Add to add your credentials (e.g., GitHub username and password, or personal access token).
- Kind:** Choose the appropriate kind (e.g., Username with password or SSH Username with private key).
- Scope:** Set to Global.
- Username:** Enter your GitHub username.
- Password:** Enter your GitHub password or personal access token.

Click Add to save.

The screenshot shows the Jenkins Credentials Provider dialog. It displays the message 'Jenkins Credentials Provider: Jenkins'. Under the 'Domain' section, it says 'Global credentials (unrestricted)'. At the bottom right is a large blue 'Add' button.



Branch Specifier:

By default, this is set to ***/master**. If you are using a different branch, update it accordingly (e.g., ***/main**).

Script Path:

Specify the path to your **Jenkinsfile** if it's not in the root directory of your repository. By default, it is set to **Jenkinsfile**.

Adding Pipeline syntax:

click on pipeline syntax and setup the **kubeconfig**.

you need to add credentials for **kubeconfig**,

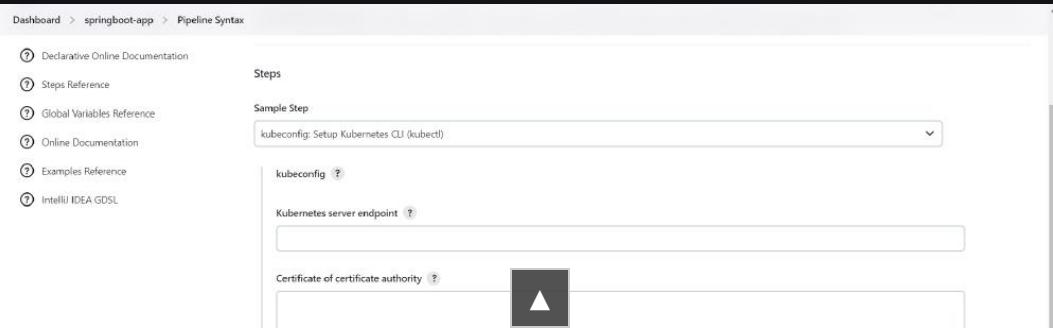
1. Credentials: Click Add to add your credentials.
2. Kind: Choose the appropriate kind (e.g., Secret text).
3. Scope: Set to Global.
4. Secret: Here you need to add the kubeconfig data which is present at .kube/config file in your local system.

if you want to check your config file's data then write below command,

```
ls -la
```

After that you can see many files from that you move to config file using below command,

```
cd .kube/config
```



Credentials

- none -

+ Add + Jenkins

Jenkins Credentials Provider: Jenkins

Domain: Global credentials (unrestricted)

Kind: Secret text

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret:

ID: kubeconfig-springboot

Description: Kubernetes configuration file for spring boot application

Add

Overview

This Snippet Generator will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click Generate Pipeline Script, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step: kubeconfig: Setup Kubernetes CLI (kubectl)

kubeconfig: ?

Kubernetes server endpoint: ?
https://127.0.0.1:32769

Certificate of certificate authority: ?

Credentials

Kubernetes configuration file for spring boot application

+ Add +

Dashboard > springboot-app > Pipeline Syntax

Sample Step: kubeconfig: Setup Kubernetes CLI (kubectl)

kubeconfig: ?

Kubernetes server endpoint: ?
https://127.0.0.1:32769

Certificate of certificate authority: ?

Credentials

Kubernetes configuration file for spring boot application

+ Add +

Generate Pipeline Script

```
kubeconfig(credentialsId: 'kubeconfig-springboot', serverUrl: 'https://127.0.0.1:32769') {
    // some block
}
```

Global Variables

Save Configuration:

Scroll down and click Save to save the job configuration.



You can check pipeline overview in your Jenkins Dashboard.

when you click on the console then you will get below result.

Then you can see it in your main dashboard.



Conclusion

Deploying Spring Boot applications on Kubernetes using Jenkins CI/CD ensures fast and efficient deployment. By using Docker for containerization and Kubernetes for orchestration, we achieve easy scalability and robust management of our applications. Jenkins builds, tests, and automates deployments, increasing productivity and reliability. GitHub simplifies version control and collaborative development, ensuring code integrity and traceability. This comprehensive approach accelerates development cycles, ensures implementation stability, and enables teams to consistently deliver high-quality software by effectively meeting dynamic market requirements.

Spring boot application deployment in Kubernetes with Jenkins CI/CD pipeline - FAQs

What are the main difference between Spring boot and other framework?

Here main 2 consideration that are given below

- 1. language preference:** If you are comfortable or familiar with java then spring boot is right choice.
- 2. Project requirements:** Spring boot is vary brilliant for micro service and backend development but If other framework may serve broader web development needs then you select as per your project requirement.

What is the main role of Autoconfiguration in spring boot?

Autoconfiguration is a powerful features that automates the configuration of your application that based on library included in your project.

Imagine, that you want to connect MYSQL database with your application. Let's see, How it simplifies the process.

- 1. Add dependencies:** Add the `spring-boot-starter-data-jpa` dependencies to your project's build configuration (e.g., `pom.xml`). This dependency provides libraries to integrate with the relational database.

2. Autoconfiguration tasks: Spring Boot checks for `spring-boot-starter-data-jpa` dependencies. It then triggers the appropriate autoconfiguration class, like `DataSourceAutoConfiguration`.

3. Automatic Configuration: If you do not explicitly configure the `DataSource` bean yourself, the `DataSourceAutoConfiguration` class will create one based on the default settings or properties you provide in your `application.yml` or `application.properties` file.

4. Ready to use: You can now interact with a MySQL database using Spring Data JPA features without writing any manual configuration code for the data source itself.

What is pods and nodes in the Kubernetes?

The nodes are the physical or virtual machine where pods are the smallest unit of Kubernetes.

Each pod has own IP address and only one application is run per pod. pods are ephemeral means pods can be die easily.

for example, Our DB pod is die due to docker crashes or application crashes or resource at that new pod will take place with new IP address. To overcome this issues Kubernetes has one feature called services.

services has fixed IP address that can't be change.

Why .kube file is not sometimes showing?

By default, `.kube` file resides our home directory and you are finding it in another directory.

```
ls -a ~/.kube
```

This command is basically used for reveal the hidden files and folders including `.kube` folder.

- Here "ls" command is used for showing list of content of the directory.
- "-a" command is used for show the hidden files and folder.
- "~/" command is used for expand your home directory path.

How can a basic Spring Boot application be created?



Write the necessary Java classes (e.g. HelloController.java, DemoApplication.java), create directories and files for your application, and put in place the configuration settings of your app.

Are you ready to unleash the power of **DevOps** to streamline your **Software Development and Deployment**? Learn about our [**DevOps Live Course**](#) at GeeksforGeeks, created for all professionals in practice with continuous integration, delivery, and deployment. Learn about leading tools, industry best practices, and techniques for **automation** through an interactive session with hands-on **live projects**. Whether you are new to DevOps or looking to improve your skills, this course equips you with everything needed to streamline workflows and deliver excellent quality software in the least amount of time. Learn to take your skills in DevOps to the next level now, and harness the power of streamlined software development!



Rromil... + Follow



⟨ Previous Article

How to Deploy Java Application in
Kubernetes?

Next Article ›

Similar Reads

Java Application Deployment In Kubernetes With Jenkins CI/CD Pipeline

In modern software development, deploying applications to Kubernetes clusters has become a common practice due to their scalability and reliability....

⌚ 5 min read

Python Application Deployment In Kubernetes With Jenkins CI/CD Pipeline

In modern software development, deploying applications to Kubernetes clusters has become a common practice due to their scalability and reliability....

⌚ 6 min read



Node.Js Application Deployment in Kubernetes with Jenkins CI/CD Pipeline

As we know in the modern world we use Continuous Integration and Continuous Deployment for fast and efficient delivery of our application to...

⌚ 9 min read

Deployment Of Spring Boot Application In Jenkins

In this article, we will be exploring how to automate the process of building Spring Boot applications by employing a polling trigger. We will also learn ho...

⌚ 5 min read

Build, Test and Deploy a Flask REST API Application from GitHub using...

Nowadays even for small web applications or microservices, we need an easier and faster way to deploy applications that is reliable and safe. These...

⌚ 4 min read

Kubernetes - Creating Deployment and Services using Helm in Kubernetes

Prerequisite: Kubernetes Helm is used for managing your Kubernetes Deployment. With helm, we can tear down and create a deployment with a...

⌚ 4 min read

How to Deploy Spring Boot Application in Kubernetes ?

The Spring Boot framework provides an assortment of pre-configured templates and tools to make the development of Java-based applications simpler. With...

⌚ 7 min read

Understanding Jenkins CI/CD Pipeline And Its Stages

Jenkins is an open-source automation server that enables developers to reliably build, test, and deploy applications. It supports continuous integration and...

⌚ 15 min read

How To Create Methods In Jenkins Declarative Pipeline?

Jenkins is an open-source automation tool used to automate the fetch, build, test, and deploy stages of a software application. Here in this guide, we will fir...

⌚ 4 min read

Setting Up A CI/CD Pipeline For PHP Applications Using Jenkins

In contemporary software development, Continuous Integration and Continuous Deployment (CI/CD) pipelines have arisen as key devices for smoothing out the...



Article Tags :[DevOps](#)[Kubernetes](#)[DevOps Projects](#)

- ⌚ Corporate & Communications Address:-
A-143, 9th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305) | Registered Address:- K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddha Nagar, Uttar Pradesh, 201305



Company	Languages	DSA	Data Science &	Web	Python
About Us	Python	Data Structures	ML	Technologies	Tutorial
Legal	Java	Algorithms	Data Science With	HTML	Python
In Media	C++	DSA for Beginners	Python	CSS	Programming
Contact Us	PHP	Basic DSA	Data Science For	JavaScript	Examples
Advertise with us	GoLang	Problems	Beginner	TypeScript	Python Projects
GFG Corporate	SQL	DSA Roadmap	Machine Learning	ReactJS	Python Tkinter
Solution	R Language	Top 100 DSA	ML Maths	NextJS	Web Scraping
Placement	Android Tutorial	Interview	Data Visualisation	Bootstrap	OpenCV Tutorial
Training Program	Tutorials Archive	Problems	Pandas	Web Design	Python Interview
GeeksforGeeks		DSA Roadmap by	NumPy		Question
Community		Sandeep Jain	NLP		Django
		All Cheat Sheets	Deep Learning		
Computer Science	DevOps	System Design	Interview	School Subjects	GeeksforGeeks Videos
Operating Systems	Linux	High Level Design	Preparation		
Computer Network	AWS	Low Level Design	Competitive	Mathematics	DSA
Database	Docker	UML Diagrams	Programming	Physics	Python
Management System	Kubernetes	Interview Guide	Top DS or Algo for	Chemistry	Java
Software Engineering	Azure	Design Patterns	CP	Biology	C++
	GCP	OOAD	Company-Wise	Social Science	Web Development
	DevOps Roadmap	System Design	Recruitment	English Grammar	Data Science
		Bootcamp	Process	Commerce	CS Subjects
		Interview	Company-Wise	World GK	
		Questions	Preparation		



Preparation

Digital Logic

Design

Engineering Maths

Software

Development

Software Testing

Aptitude

Preparation

Puzzles

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved

