java -jar jenkins.war --httpPort=9090

username : laxman

password: Laxman1436@

==========================

What is Build & Deployment

===========================

1) Take source code from git repo

2) Compile & Package that code

3) Perform Code Review

4) Upload Build Artifact to Nexus

5) Create Docker Image

6) Create Container

=========================

Application Environments

=========================

1) DEV

2) SIT

3) UAT

4) PILOT

5) PROD

=> Build and Deployment process in all these environments is difficult and time taking process.

=> To avoid the challenges involved in Manual Build and Deployment process we are going for JENKINS.

========

Jenkins

========

=> Jenkins is used to automate build and deployment process

=> Jenkins is a CI CD software

=> CI CD means continuous integration & Continuous deployment

=> Jenkins Software developed by using Java language (To run jenkins java is mandatory).

=> Jenkins Server Runs on Port : 8080

===============

Jenkins Setup

===============

1) Create Ubuntu VM in AWS Cloud

2) Connect to Ubuntu VM using MobaXterm

```
$ curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

```
$ echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install fontconfig openjdk-11-jre
```

```
$ sudo apt-get install jenkins
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install jenkins
```

Note: Enable 8080 port number in security group

=> Access Jenkins Server using below URL

URL : http://public-ip:8080/

==

========

Logging

========

-> The process of storing application execution details to a file is called as Logging.

-> With log messages we can understand execution flow of the application.

-> We can understand exceptions occuring in the project by seeing log messages.

Logging Frameworks

----

1) Log4J

2) Log4J2

3) LogBack

4) LogStash

Log Monitoring Tools

----

1) Putty

2) WinScp

3) ELK

4) Splunk (Licensed)

Logging Architecture

----

1) Logger : This class providing methods to generate log messages

2) Layout : It represents log message structure (format of log msg)

3) Appender : It is used to write log message to destination

4) Destiation : It can be console/file/database

Note: We will use files to store our log messages.

===============

Logging Levels

================

1) TRACE

2) DEBUG

3) INFO   (it is default log level in boot application)

4) WARN

5) ERROR

6) FATAL

=====================

Log Level Hierarchy

=====================

TRACE > DEBUG > INFO > WARN > ERROR > FATAL

=> When we set one Log level, application will print log message from that level to all higher level messages.

=> In Spring Boot by default it will use level as INFO

-> In Spring Boot by default it will use ConsoleAppender

-> To generate log msgs in log file we have set below property in application.properties file

```
---------------------------------------------------------------
logging.file.name=app.log
---------------------------------------------------------------
```

```
package in.ashokit.rest;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
```

```
public class MessageController {

        private Logger logger = LoggerFactory.getLogger(MessageController.class);

        @GetMapping("/welcome")
        public String welcomeMsg() {
                logger.debug("this is debug msg from welcome.....");
                logger.info("welcomeMsg() execution started.....");

                String msg = "Welcome To Ashok IT...";

                try {
                        int i = 10 / 0;
                } catch (Exception e) {
                        logger.error("Exception Occured" + e.getMessage());
                }

                logger.warn("This is warning from welcome method...");

                logger.info("welcomeMsg() execution ended...");
                return msg;
        }

        @GetMapping("/greet")
        public String greetMsg() {

                logger.debug("this is debug msg from greet.....");
                logger.info("greetMsg() execution started...");
                String msg = "Good Morning...";
                logger.warn("This is warning from greet method...");


                logger.info("greetMsg() execution ended...");

                return msg;

        }


}
```

==================

Rolling Appenders

==================

1) Size Based Rolling

2) Time Based Rolling

=> We can customize springboot application log configuration by creating logback.xml file under src/main/resources folder.

1) What is Logging ?

2) What is Log Monitoring?

3) Logging Architecture

- Logger
- Layout
- Appender

4) Log Levels

5) Log Level Hierarchy

6) How to set Log Level (log.level.root = DEBUG)

7) How to implement Logging in java class

8) What is Rolling in Logging ?