devops:

A3: Demonstrate the process of integration github repository with Jenkins to automate the project execution in CI/CD pipeline.

Go to: Manage Jenkins → Plugins → Available
Search and install:

1. Git plugin
2. Pipeline plugin
3. GitHub Integration plugin

PART 2 — GitHub Repo Setup

Create Folder Structure in GitHub like:

If using Java only

```
/ (root)

  └──── Main.java
```

If using Python only

```
/ (root)

  └──── test.py
```

If using both

```
/ (root)

  ├──── Main.java

  ├──── test.py
```

Code Examples to push in GitHub

Main.java

```
public class Main {

    public static void main(String[] args) {
        System.out.println("Java Build Success in Jenkins 🚀");
    }
}
```

test.py

```
print("Python Build Success in Jenkins 🐍🚀")
```

Upload both files to GitHub → push to main branch.

PART 3 — Configure Jenkins Job

Step 1: Create a new job

Jenkins Dashboard → New Item → Freestyle Project → name: first → OK

Step 2: Add GitHub repository

Inside Job → Configure

Source Code Management → Git

Repository URL: https://github.com/YourUserName/RepoName.git

Branch: main

Step 3: Add Build Steps

Scroll down → Build → Add build step → Execute Windows batch command

To build both Java + Python

Paste this into the command script:

```
javac Main.java

java Main

python test.py
```

⚠ If Python is not recognized, use full path:

"C:\Users\Admin\AppData\Local\Programs\Python\Python312\python.exe " test.py

Step 4: Save & Build

C2:Create a maven projects with all dependencies required for the application in CI/CD pipeline.

PART 1 — Create Maven Project in Eclipse

Step 1: Open Eclipse

Go to:

File → New → Other → Maven → Maven Project → Next

Step 2: Select option

Check:

Create a simple project (skip archetype selection)

→ Click **Next**

Step 3: Fill details

Field	Value
Group Id	com.froxcy
Artifact Id	JSONReaderDemo
Version	0.0.1-SNAPSHOT
Packaging	jar

Click **Finish**

PART 2 — Correct Maven Folder Structure

After project creation, you will see:

```
JSONReaderDemo
 ├── src
 │   ├── main
 │   │   └── java     (Java source should be here)
 │   └── test
 │       └── java
 ├── pom.xml
```

If src/main/java folder doesn’t exist

Right click project → New → Source Folder → name:

src/main/java

PART 3 — Create JSON folder & file

Right click **JSONReaderDemo** project → New → Folder
Name folder:

JSON

Inside JSON folder create file:

Right click JSON → New → File → name:

student.json

Write JSON example:

```
{
  "firstname": "Harshith",
  "lastname": "Gowda"
}
```

Save the file.

PART 4 — Add JSON dependency to pom.xml

Open pom.xml → Switch to **pom.xml** tab → Paste this inside:

```
<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.froxcy</groupId>
  <artifactId>JSONReaderDemo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>JSONReaderDemo</name>

  <dependencies>
    <dependency>
      <groupId>com.googlecode.json-simple</groupId>
      <artifactId>json-simple</artifactId>
      <version>1.1.1</version>
    </dependency>
  </dependencies>

</project>
```

Update Maven:

Right click project → Maven → Update Project (Alt + F5)
→ OK

PART 5 — Create Java Class

Go to:
src/main/java → Right click → New → Package → name:

com.froxcy.json

Create Java class:

Right click package → New → Class → ReadJSON

Paste this code:

```
package com.froxcy.json;


import java.io.FileReader;
import java.io.IOException;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;


public class ReadJSON {
   public static void main(String[] args) throws IOException, ParseException
{


      JSONParser jsonparser = new JSONParser();
      FileReader reader = new FileReader(".\\JSON\\student.json");
```

```
        Object obj = jsonparser.parse(reader);

        JSONObject studentobj = (JSONObject) obj;


        String fname = (String) studentobj.get("firstname");

        String lname = (String) studentobj.get("lastname");


        System.out.println("Firstname: " + fname);

        System.out.println("Lastname: " + lname);

    }

}
```

PART 6 — Run the program

Right click ReadJSON.java →

Run As → Java Application

C2:Integrate communication channel with Jenkins for status of project and also enable email notification for a build.