Lab 8
ICS423 - Internet of Things

Jayant Kolapkar - 2021BCS0132

Question

Task 1: Assuming that the IoT data is available in csv format or json format (atleast 100 rows of values), develop a frontend using reactjs to display the output (refresh the data periodically).

Task 2: Design a CI/CD pipeline using Jenkins to continuously update the codebase of Arduino sketches.

Task 1

Algorithm and Code

1. Create a node server which generates the IOT (temperature data) periodically and

stores the data in a CSV file.

Code: server.js

```
const express = require("express");
const fs = require("fs");
const cors = require("cors");
const fastCsv = require("fast-csv");
const app = express();
const PORT = 5000;
const FILE_PATH = "iot_data.csv";
```

```
// Enable CORS
app.use(cors());

// Function to generate random IoT data
const generateData = () => {
    const timestamp = new Date().toISOString();
```

```
    const temperature = (Math.random() * 15 + 20).toFixed(2); // Random
temp between 20-35°C
    return `${timestamp},${temperature}\n`;
};

// Function to write data to CSV file every 2 seconds
const appendData = () => {
    const header = "timestamp,temperature\n";
    if (!fs.existsSync(FILE_PATH)) {
        fs.writeFileSync(FILE_PATH, header);
    }
    setInterval(() => {
    fs.appendFileSync(FILE_PATH, generateData());
    }, 2000);
};

// API endpoint to get the last 100 rows
app.get("/data", (req, res) => {
    const rows = [];
    fs.createReadStream(FILE_PATH)
        .pipe(fastCsv.parse({ headers: true }))
        .on("data", (row) => rows.push(row))
        .on("end", () => {
            res.json(rows.slice(-100)); // Send last 100 rows
        });
});

// Start the server and begin data generation
app.listen(PORT, () => {
    console.log(`Server running on http://localhost:${PORT}`);
    appendData();
});
```

2. Start the server. You will observe that the data will get stored in the CSV file continuously.

3. Create React dashboard by creating the react app and installing the necessary dependencies.

4. Fetch the data from the CSV file on the server and fetch the data using the GET api by integrating the react frontend with node backend.

Code: App.js

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import "./App.css"; // Import external CSS

const API_URL = "http://localhost:5000/data";

const App = () => {
  const [data, setData] = useState([]);
  const [lastUpdated, setLastUpdated] = useState(null);
  const [rowCount, setRowCount] = useState(0);

  useEffect(() => {
    const fetchData = () => {
      axios.get(API_URL)
        .then((response) => {
          setData(response.data);
          setLastUpdated(new Date().toLocaleTimeString()); // Set last
update time
          setRowCount(response.data.length); // Set row count
        })
        .catch((error) => console.error("Error fetching data:", error));
    };

    fetchData(); // Initial fetch
    const interval = setInterval(fetchData, 5000); // Fetch every 5
seconds

    return () => clearInterval(interval); // Cleanup on unmount
  }, []);

  return (
    <div className="container">
```

```
      <h2 className="title">IoT Sensor Data</h2>
      <p className="info">
        <strong>Last Updated:</strong> {lastUpdated || "Fetching..."} |
        <strong> Rows Fetched:</strong> {rowCount}
      </p>
      <div className="table-container">
        <table className="styled-table">
          <thead>
            <tr>
              <th>Timestamp</th>
              <th>Temperature (°C)</th>
            </tr>
          </thead>
          <tbody>
            {data.map((row, index) => (
              <tr key={index}>
                <td>{row.timestamp}</td>
                <td>{row.temperature}</td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    </div>
  );
};

export default App;
```

5. Open the localhost and view the updated data on the dashboard.

IoT Sensor Data

Last Updated: 10:10:20 pm | **Rows Fetched:** 100

Timestamp	Temperature (°C)
2025-03-03T16:37:00.111Z	25.83
2025-03-03T16:37:02.115Z	32.79
2025-03-03T16:37:04.121Z	23.91
2025-03-03T16:37:06.125Z	28.60
2025-03-03T16:37:08.130Z	27.01
2025-03-03T16:37:10.138Z	34.09
2025-03-03T16:37:12.146Z	20.39
2025-03-03T16:37:14.161Z	26.70
2025-03-03T16:37:16.162Z	21.89
2025-03-03T16:37:18.169Z	21.13
2025-03-03T16:37:20.173Z	26.54
2025-03-03T16:37:22.181Z	28.86
2025-03-03T16:37:24.189Z	31.66

IoT Sensor Data

Last Updated: 10:10:35 pm | **Rows Fetched:** 100

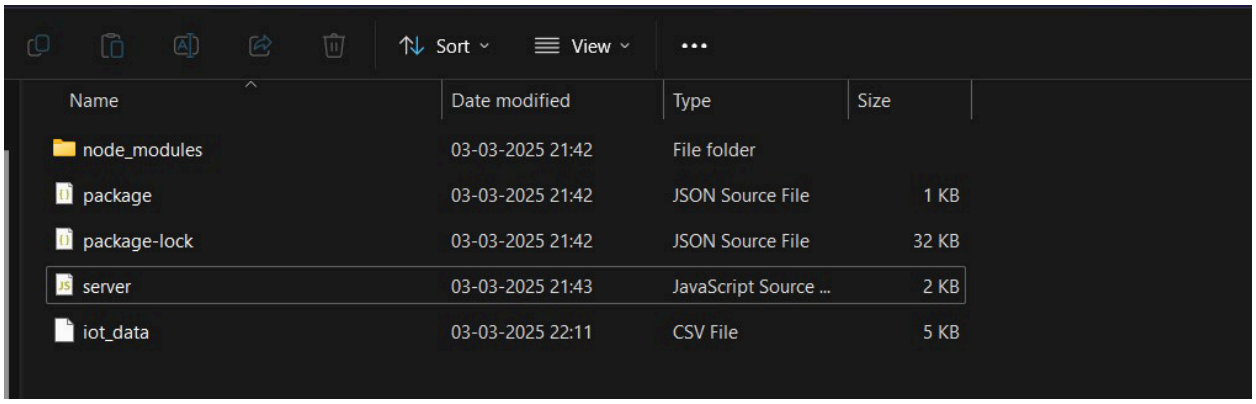
Timestamp	Temperature (°C)
2025-03-03T16:37:16.162Z	21.89
2025-03-03T16:37:18.169Z	21.13
2025-03-03T16:37:20.173Z	26.54
2025-03-03T16:37:22.181Z	28.86
2025-03-03T16:37:24.189Z	31.66
2025-03-03T16:37:26.189Z	28.12
2025-03-03T16:37:28.197Z	20.26
2025-03-03T16:37:30.201Z	23.31
2025-03-03T16:37:32.209Z	33.66
2025-03-03T16:37:34.218Z	34.76
2025-03-03T16:37:36.222Z	20.85
2025-03-03T16:37:38.229Z	32.67
2025-03-03T16:37:40.239Z	20.56

IoT Sensor Data

Last Updated: 10:10:56 pm | **Rows Fetched:** 100

Timestamp	Temperature (°C)
2025-03-03T16:37:36.222Z	20.85
2025-03-03T16:37:38.229Z	32.67
2025-03-03T16:37:40.239Z	20.56
2025-03-03T16:37:42.240Z	30.03
2025-03-03T16:37:44.241Z	21.59
2025-03-03T16:37:46.244Z	32.51
2025-03-03T16:37:48.251Z	21.99
2025-03-03T16:37:50.251Z	28.78
2025-03-03T16:37:52.254Z	23.69
2025-03-03T16:37:54.269Z	30.00
2025-03-03T16:37:56.274Z	33.65
2025-03-03T16:37:58.283Z	29.83
2025-03-03T16:38:00.288Z	24.92

6. View and verify the Data getting stored in CSV file.



	A	B	C	D
1	timestamp	temperature		
2	2025-03-03	30.38		
3	2025-03-03	25.83		
4	2025-03-03	32.79		
5	2025-03-03	23.91		
6	2025-03-03	28.6		
7	2025-03-03	27.01		
8	2025-03-03	34.09		
9	2025-03-03	20.39		
10	2025-03-03	26.7		
11	2025-03-03	21.89		
12	2025-03-03	21.13		
13	2025-03-03	26.54		
14	2025-03-03	28.86		
15	2025-03-03	31.66		
16	2025-03-03	28.12		
17	2025-03-03	20.26		
18	2025-03-03	23.31		
19	2025-03-03	33.66		
20	2025-03-03	34.76		
21	2025-03-03	20.85		
22	2025-03-03	32.67		
23	2025-03-03	20.56		
24	2025-03-03	30.03		
25	2025-03-03	21.59		
26	2025-03-03	32.51		
27	2025-03-03	21.99		
28	2025-03-03	28.78		
29	2025-03-03	23.69		

	A	B	C	D
120	2025-03-03	22.53		
121	2025-03-03	28.86		
122	2025-03-03	32.57		
123	2025-03-03	23.9		
124	2025-03-03	23.58		
125	2025-03-03	35		
126	2025-03-03	21.02		
127	2025-03-03	32.49		
128	2025-03-03	24.45		
129	2025-03-03	26.8		
130	2025-03-03	33.56		
131	2025-03-03	27.57		
132	2025-03-03	21.69		
133	2025-03-03	21.51		
134	2025-03-03	34.26		
135	2025-03-03	26.45		
136	2025-03-03	20.76		
137	2025-03-03	24.94		
138	2025-03-03	27.45		
139	2025-03-03	22.09		
140	2025-03-03	21.28		
141	2025-03-03	27.2		
142	2025-03-03	21.65		
143	2025-03-03	30.92		
144	2025-03-03	27.35		
145	2025-03-03	28.51		
146	2025-03-03	27.01		
147	2025-03-03	20.19		
148	2025-03-03	27.33		

iot_data

Task 2

Algorithm:

1. Install Java version 17 or 21. (For our case we will proceed with Java version 17)

```
C:\Users\kanak>java -version
java version "17.0.12" 2024-07-16 LTS
Java(TM) SE Runtime Environment (build 17.0.12+8-LTS-286)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.12+8-LTS-286, mixed mode, sharing)
```

2. Install Jenkins.

3. Start jenkins using CMD (as administrator)

```
C:\Windows\System32>C:\Users\kanak>net start jenkins
```

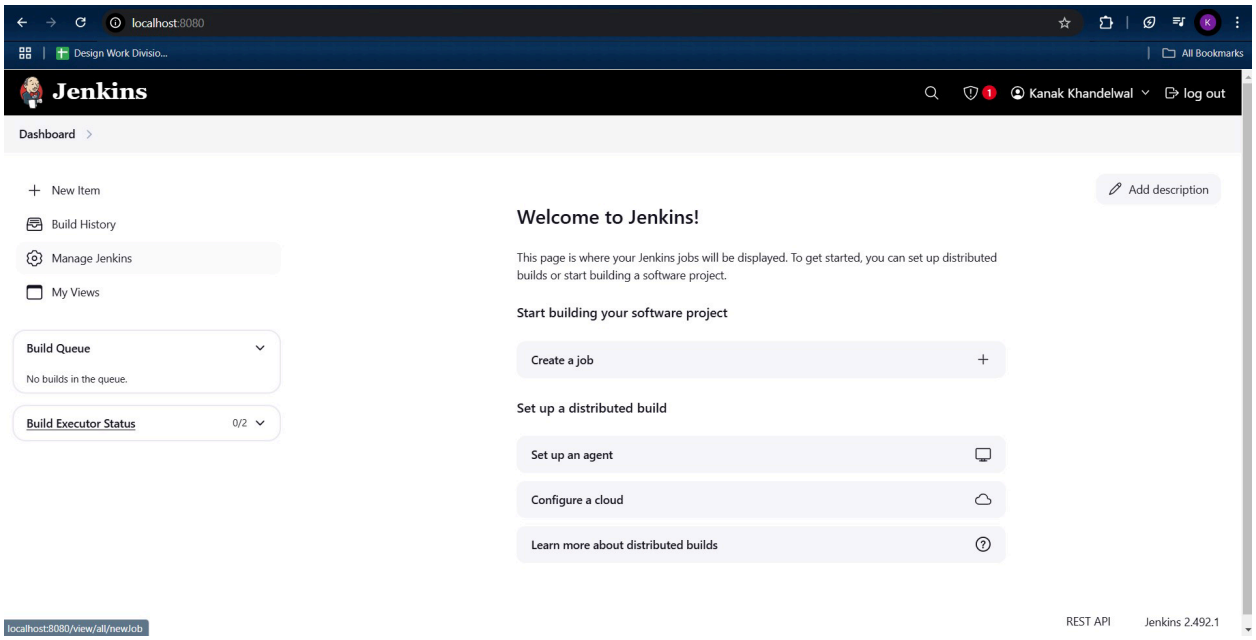
```
C:\Windows\System32>net start jenkins
The requested service has already been started.
```

4. Start your localhost (port 8080 which was added during installation).

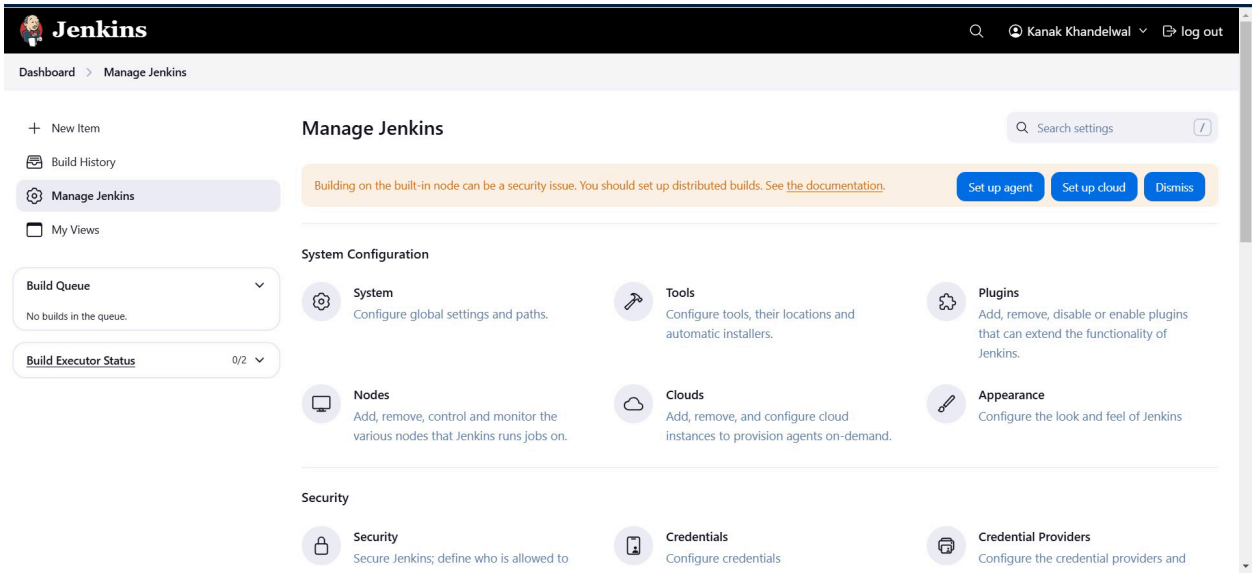
http://localhost:8080

5. Install necessary plugins and setup admin.

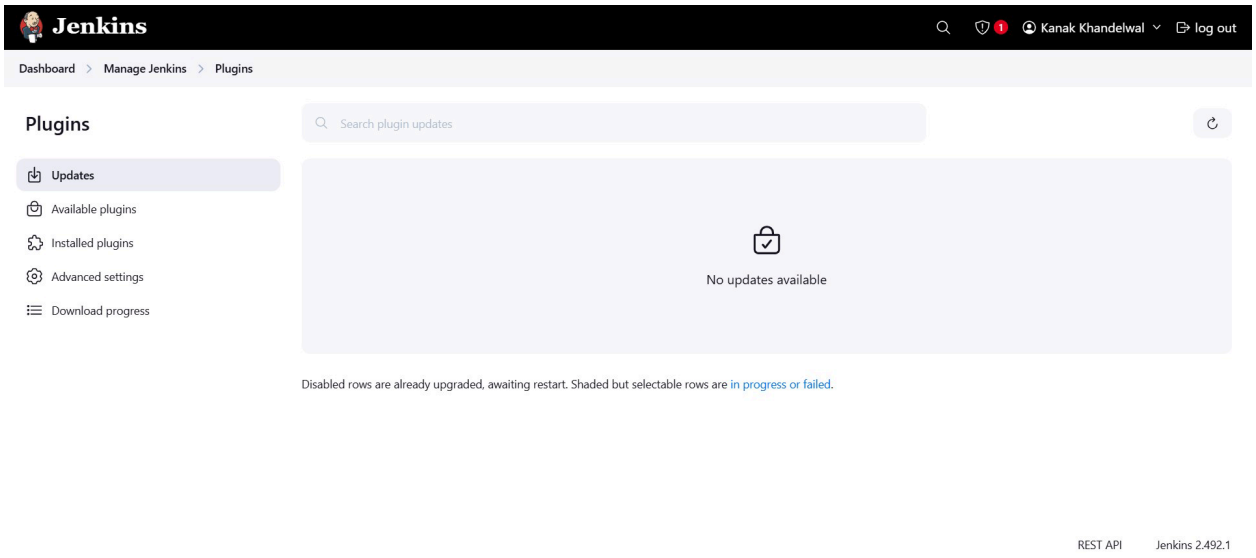
Login to jenkins. Go to dashboard



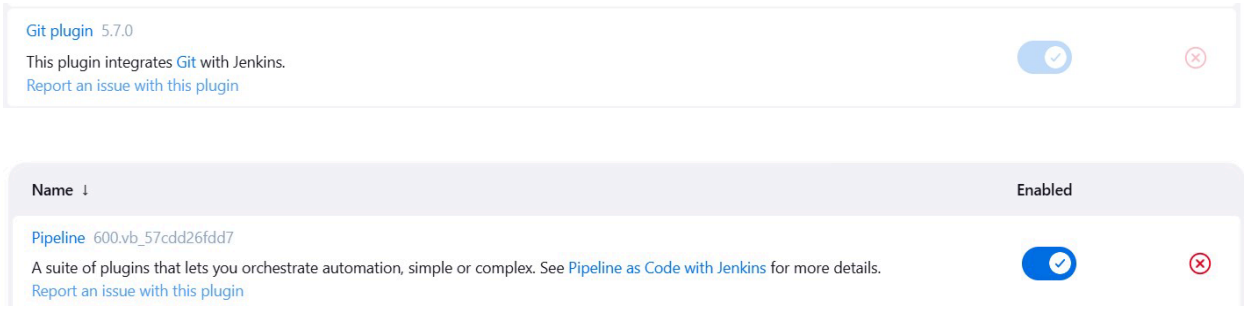
7. Go to manage jenkins from the left panel.



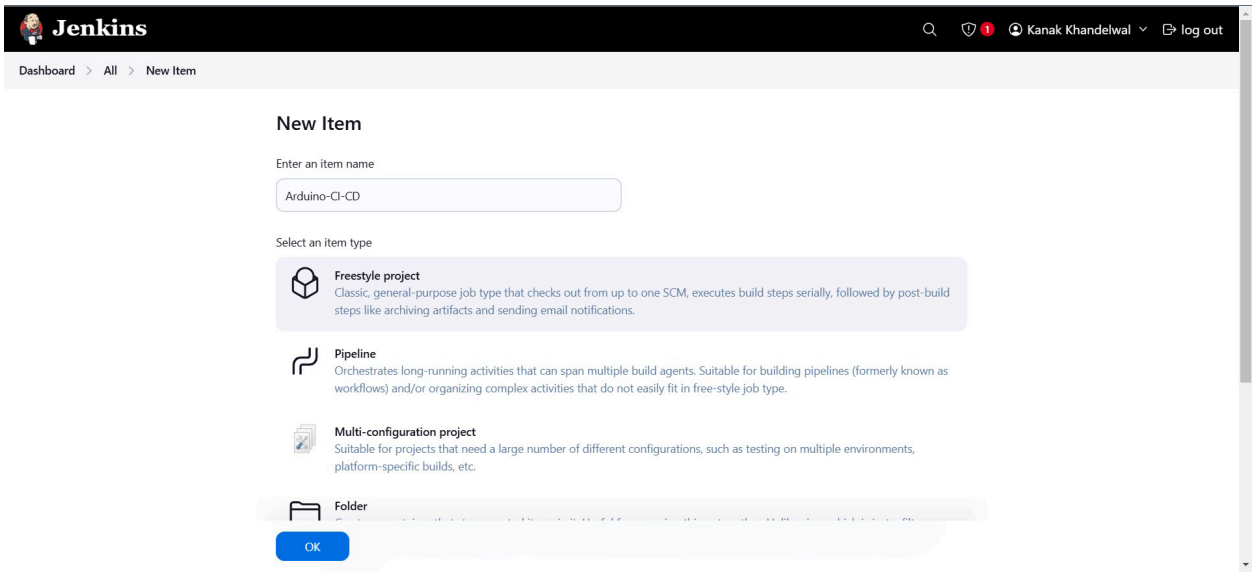
8. Go to plugins



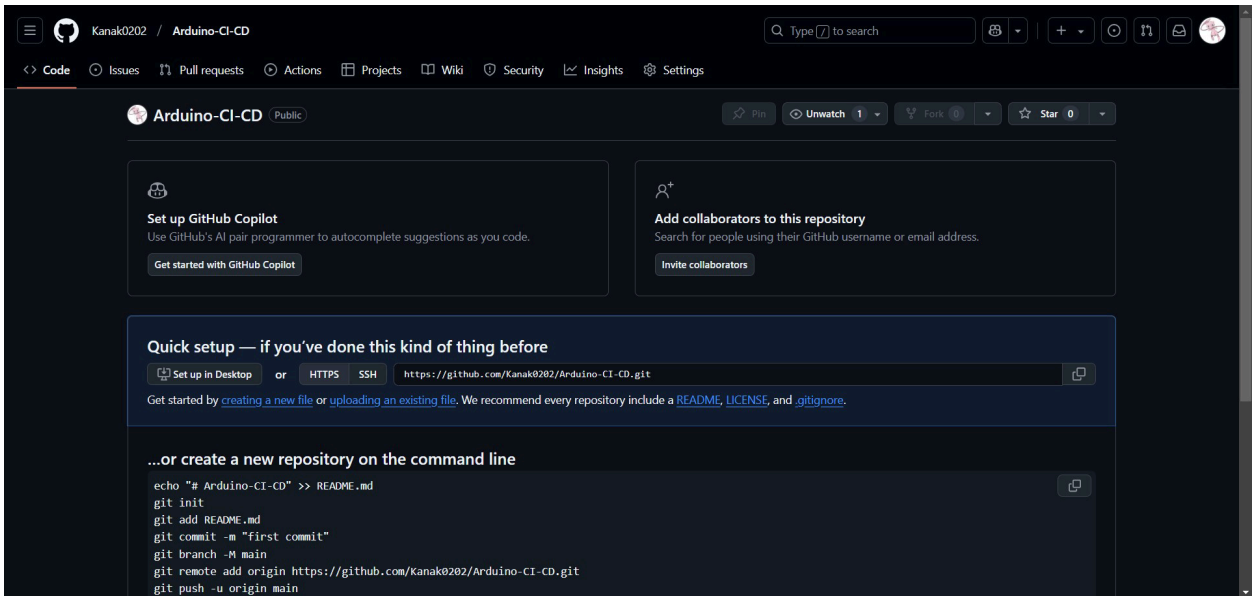
9. Install and enable Git Plugin and Pipeline Plugin.



10. Create a new jenkins job, enter desired name and select pipeline.



11. Create git repository for arduino.

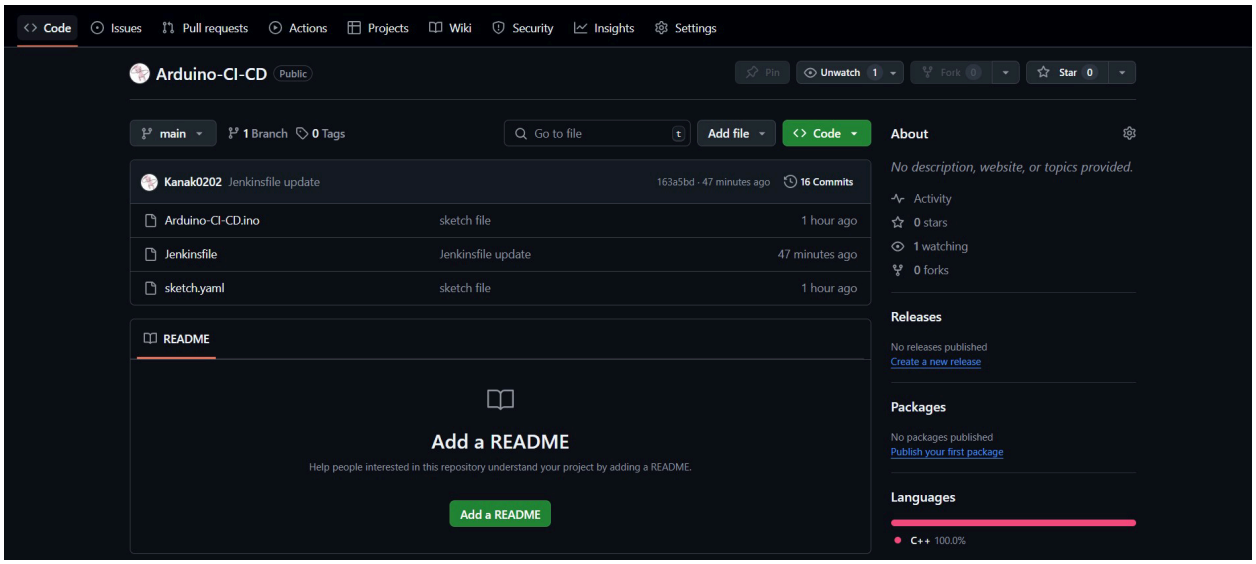


12. Setup project on local.

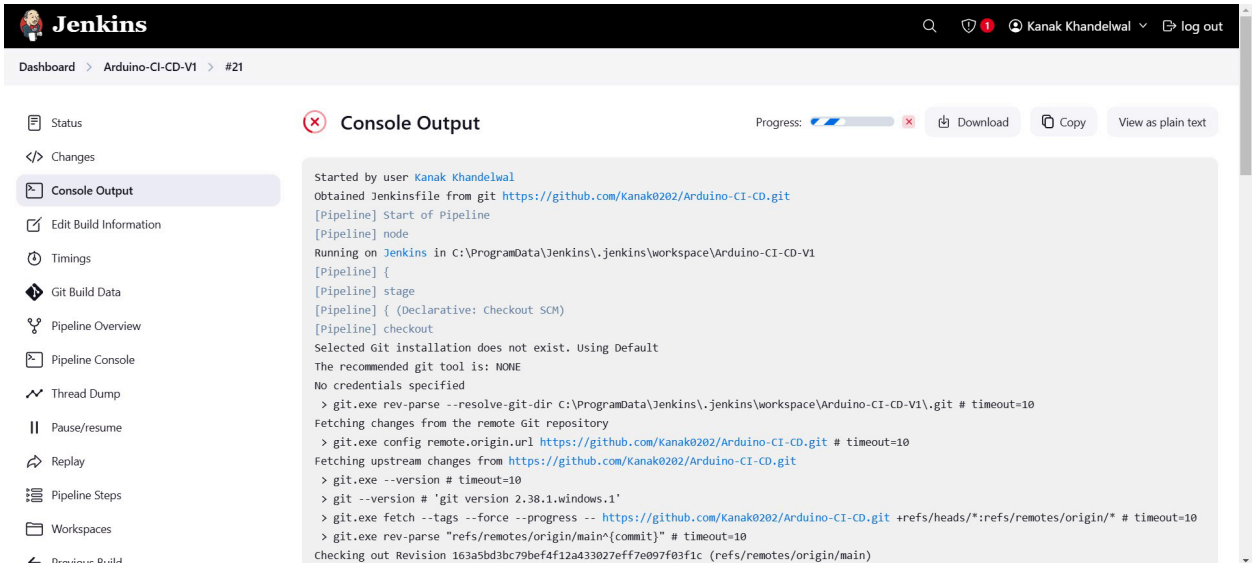
```
PS D:\KANAK\SEM-VIII\IOT\Arduino-CI-CD> git init
Initialized empty Git repository in D:/KANAK/SEM-VIII/IOT/Arduino-CI-CD/.git/
PS D:\KANAK\SEM-VIII\IOT\Arduino-CI-CD> git remote add origin https://github.com/Kanak0202/Arduino-CI-CD.
PS D:\KANAK\SEM-VIII\IOT\Arduino-CI-CD>
```

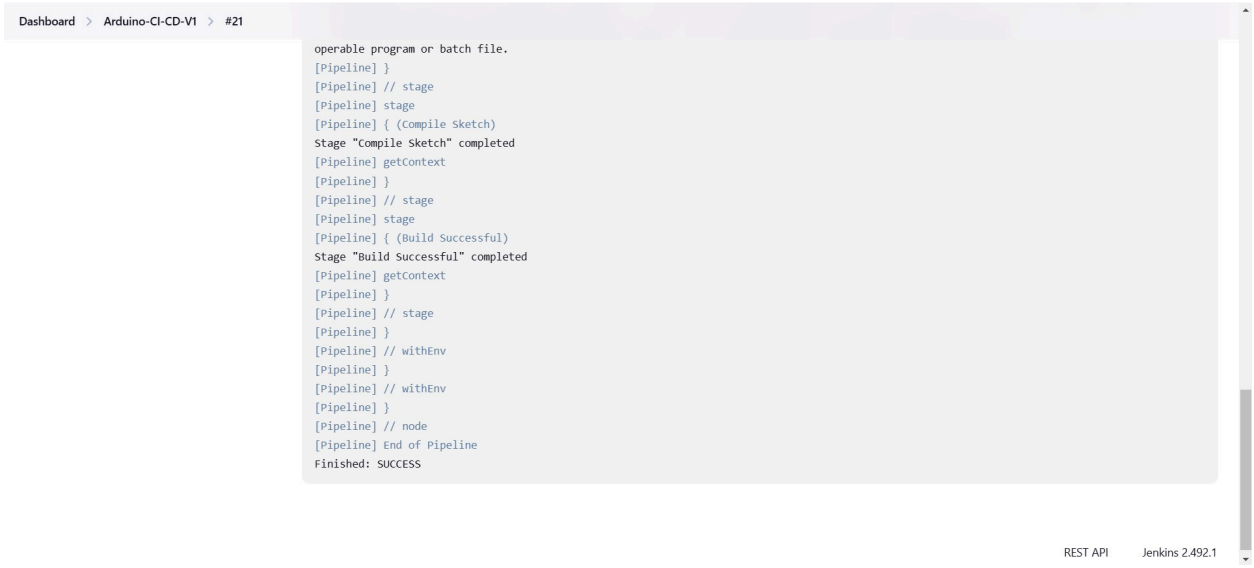
```
Administrator: C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.22631.4974]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>mkdir C:\arduino-cli-data
A subdirectory or file C:\arduino-cli-data already exists.

C:\Windows\System32>arduino-cli config init
Config file written to: C:\Users\kanak\AppData\Local\Arduino15\arduino-cli.yaml

C:\Windows\System32>arduino-cli config set directories.data C:\arduino-cli-data

C:\Windows\System32>arduino-cli config dump
board_manager:
    additional_urls: []
directories:
    data: C:\arduino-cli-data


C:\Windows\System32>
```

Push the code to github onto your repository.



Run your build.





Code

Arduino-CI-CD.ino

```
// Simple Blink Sketch

#define LED_PIN 13
void setup() {
    pinMode(LED_PIN, OUTPUT);
}
```

```
void loop() {
    digitalWrite(LED_PIN, HIGH); // Turn LED on
    delay(1000);
    digitalWrite(LED_PIN, LOW); // Turn LED off
    delay(1000);
}
```

Jenkinsfile

```
pipeline {
    agent any

    environment {
        ARDUINO_CLI =
"C:\\Users\\kanak\\AppData\\Local\\Arduino15\\arduino-cli.exe"
        ARDUINO_DATA_PATH = "C:\\arduino-cli-data"
    }

    stages {
        stage('Checkout Code') {
        steps {
            git branch: 'main', url:
'https://github.com/Kanak0202/Arduino-CI-CD.git'
        }
    }

        stage('Setup Environment') {
            steps {
                bat 'set ARDUINO_DATA_PATH=C:\\arduino-cli-data'
                bat '"%ARDUINO_CLI%" core update-index'
                bat '"%ARDUINO_CLI%" core install arduino:avr'
            }
        }

        stage('Compile Sketch') {
            steps {
                bat '"%ARDUINO_CLI%" compile --fqbn arduino:avr:uno .'
            }
```

```
        }

        stage('Build Successful') {
            steps {
                echo 'Build completed successfully!'
            }
        }
    }
}
```