Course:CT5014 DevSecOps **Assignment 20 Marks**
Date:13/11/2025 **submission date:2/12/2025**
CICD Pipelines Assignment

Lab Instructions/Assignment

1.Create repository with your name+CT5014 by logging at GitHub
2.create file index.js
And Add the following contents

```
const http = require('http');
const PORT = process.env.PORT || 3000;

const server = http.createServer((req, res) => {
   res.statusCode = 200;
   res.setHeader('Content-Type', 'text/plain');
   res.end('Hello World\n');
});

// CRITICAL FIX: The server is only told to listen if this file is the main module,
// ensuring tests can manually control startup.
if (require.main === module) {
   server.listen(PORT, () => console.log(`Server listening on http://localhost:${PORT}`));
}

// Crucial: Export the server instance so the test file can start and stop it
module.exports = server;
```

3. Create file index.test.js and the following contents

```
const request = require('http'); // Use built-in 'http' module for requests
const server = require('./index'); // Import the raw server instance

// We will use a variable to store the dynamically assigned port
let testPort;
let testURL;

// Describe the test suite for the server
describe('Node.js Hello World Server', () => {

   // 🛠️ FIX 1: Use port 0 to let the OS assign an available port, then store it.
   beforeAll((done) => {

      // Add error listener for startup failures
      server.on('error', (err) => {
```

```
        console.error('Server failed to start:', err.message);
        done(err);
    });

    // Start the server on port 0
    server.listen(0, () => {
        // Get the dynamically assigned port and update the URL
        testPort = server.address().port;
        testURL = `http://localhost:${testPort}`;

        console.log(`Test server successfully started on dynamic port ${testPort}`);
        done(); // Signal Jest that the asynchronous setup is complete
    });
});

// Test case: Check if the server returns "Hello World\n" and status 200
test('should return "Hello World\\n" and status 200', (done) => {

    // Use the dynamically assigned URL for the request
    request.get(testURL, (res) => {
        let data = '';

        // Collect the response data chunks
        res.on('data', (chunk) => {
            data += chunk;
        });

        // Once the entire response is received
        res.on('end', () => {
            try {
                expect(res.statusCode).toBe(200);
                expect(data).toBe('Hello World\n');
                done(); // Signal to Jest that the async test is complete
            } catch (error) {
                done(error); // Signal failure
            }
        });
    }).on("error", (err) => {
        // This catches connection errors (like ECONNREFUSED)
        console.error('Connection error during test:', err.message);
        done(err); // Fail the test if there's an error
    });
});
```

```
    // CRITICAL FIX: Shut down the server after all tests are done
    afterAll((done) => {
        // Remove the error listener to prevent resource leaks/unexpected behavior
        server.removeAllListeners('error');

        server.close((err) => {
            console.log("Test server stopped.");
            done(err); // Ensure the teardown completes
        });
    });
});
```

4..Add Package-lock.json file with data
Use the file share in the whatapps group

5..Add Package.json file with data

```
{
  "name": "nodejs-hello-world",
  "version": "1.0.0",
  "description": "A simple Node.js project for GitHub Actions",
  "main": "index.js",
  "scripts": {
    "start": "node index.js",
    "test": "jest"
  },
  "devDependencies": {
    "jest": "^29.7.0"
  },
  "author": "",
  "license": "ISC"
}
```

6.using GitHub actions
6.1 design CICD Pipelines
Using following information
6.2 trigger pipelines automatically by push branch and manually
6.3 build with require actions
6.4,scan repository with synk or perform SAST

6.5 setup secrets with GitHub repository for your dockerhub account
6.6 push tag image to docker hub
Provide evidence or snapshot of your lab and its successful run
1.workflow scripts
2.Workflow logs
3.DockerHub Repository snapshot

7.Pre-Requistics
1.Github Account
2.Synk account
3.Docker Hub account