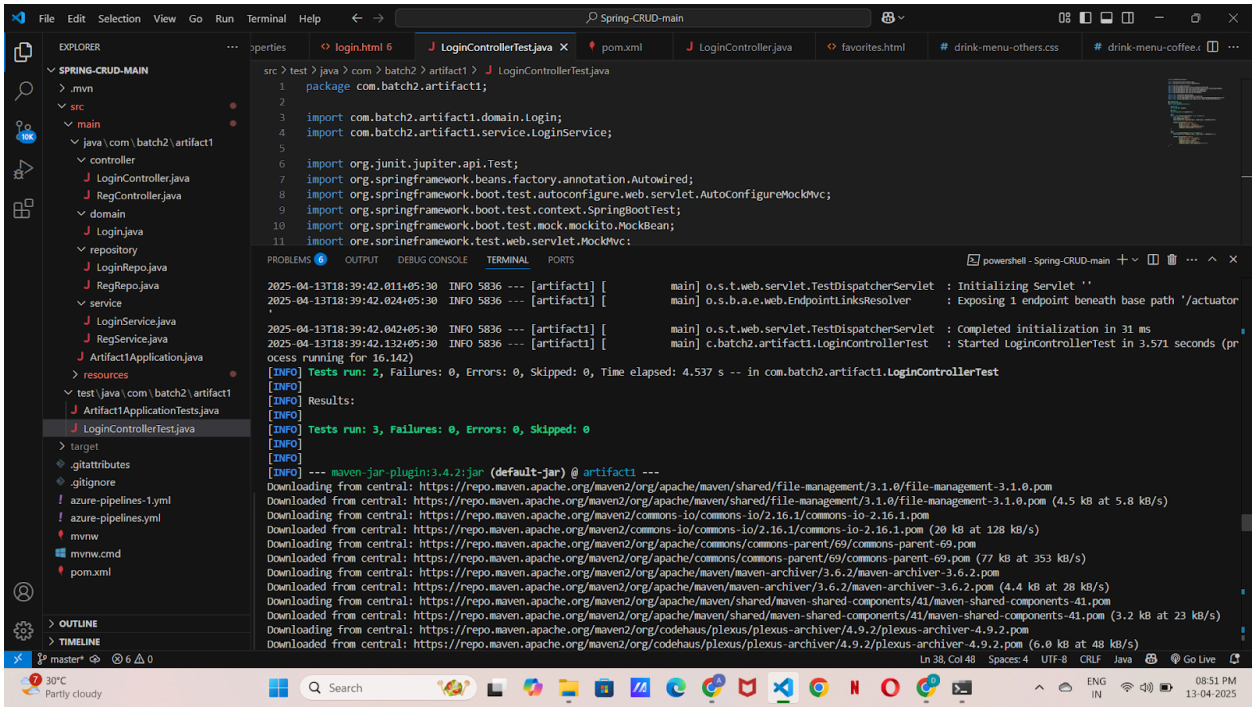
Ex No 7 RUN REGRESSION TEST USING MAVEN BUILD PIPELINE IN AZURE

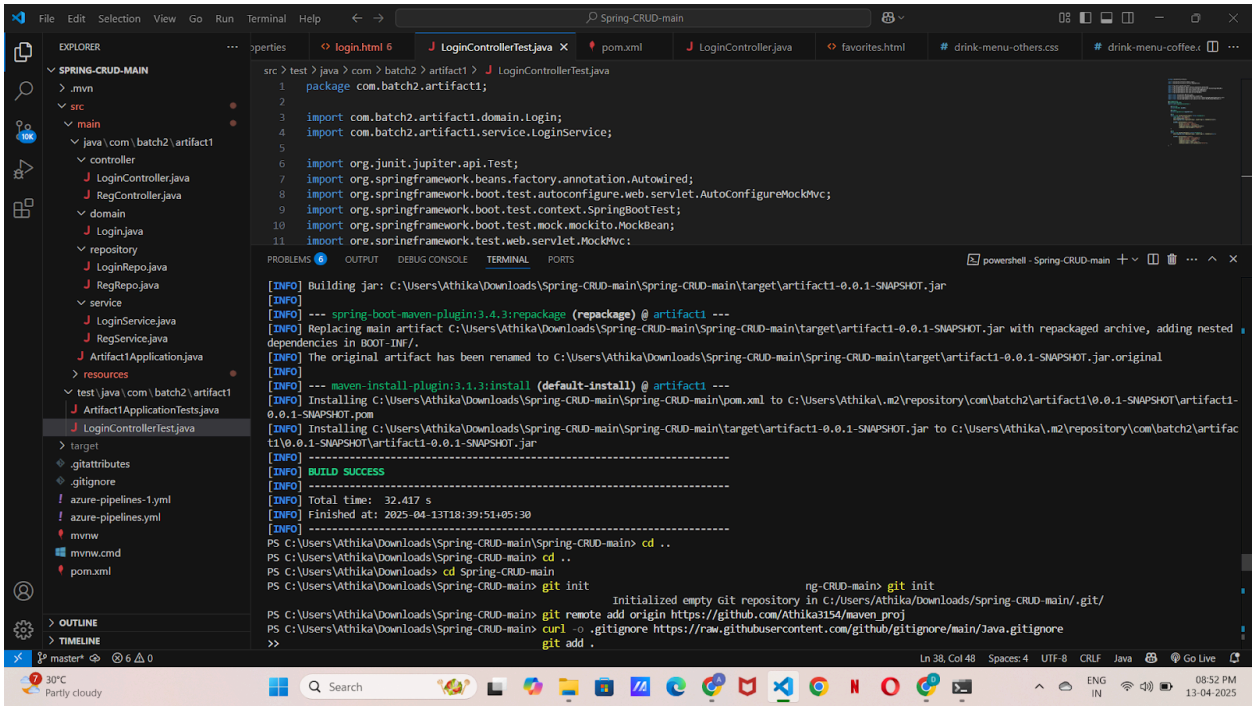
AIM:

To run a regression test using maven build pipeline in azure.

PROCEDURE:

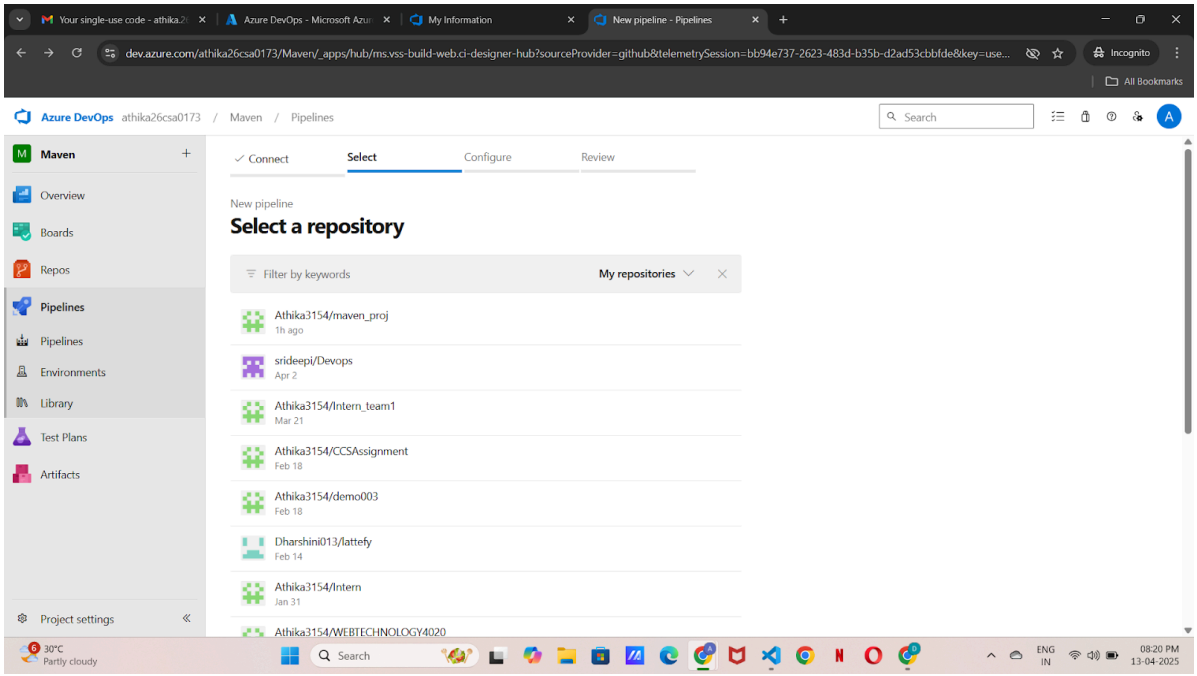
Step 1: Implement the test code and run the command “mvn test”

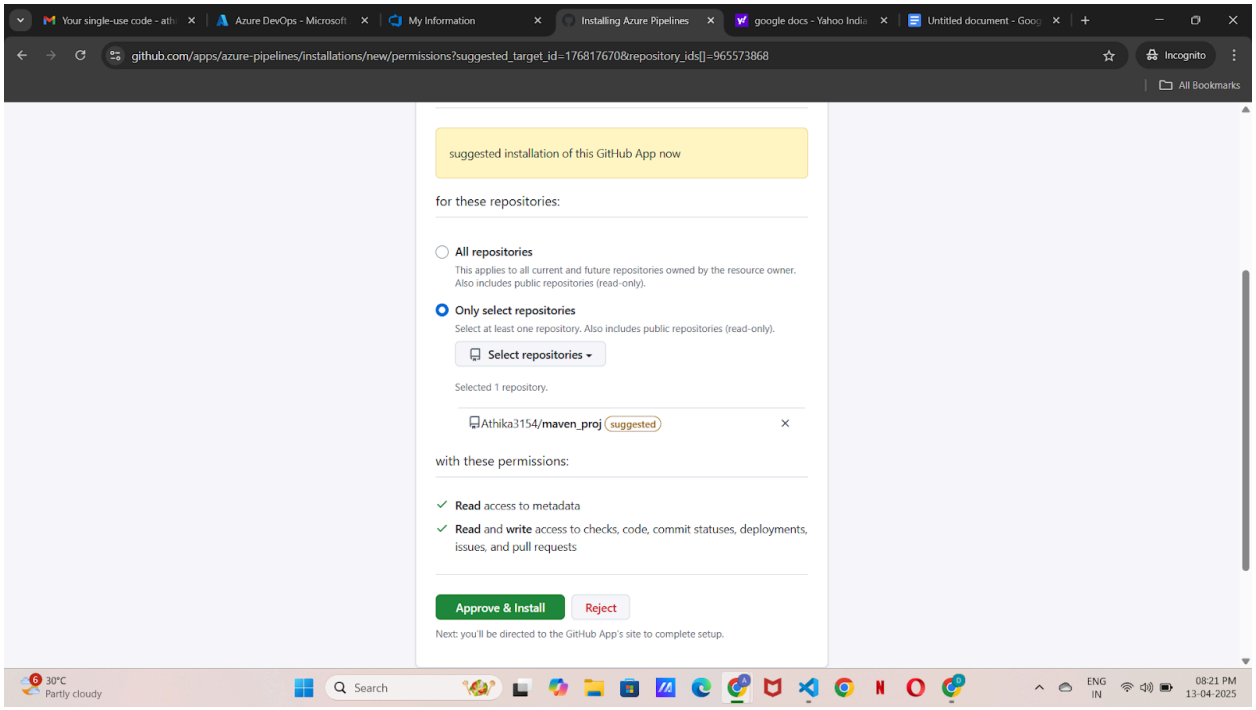


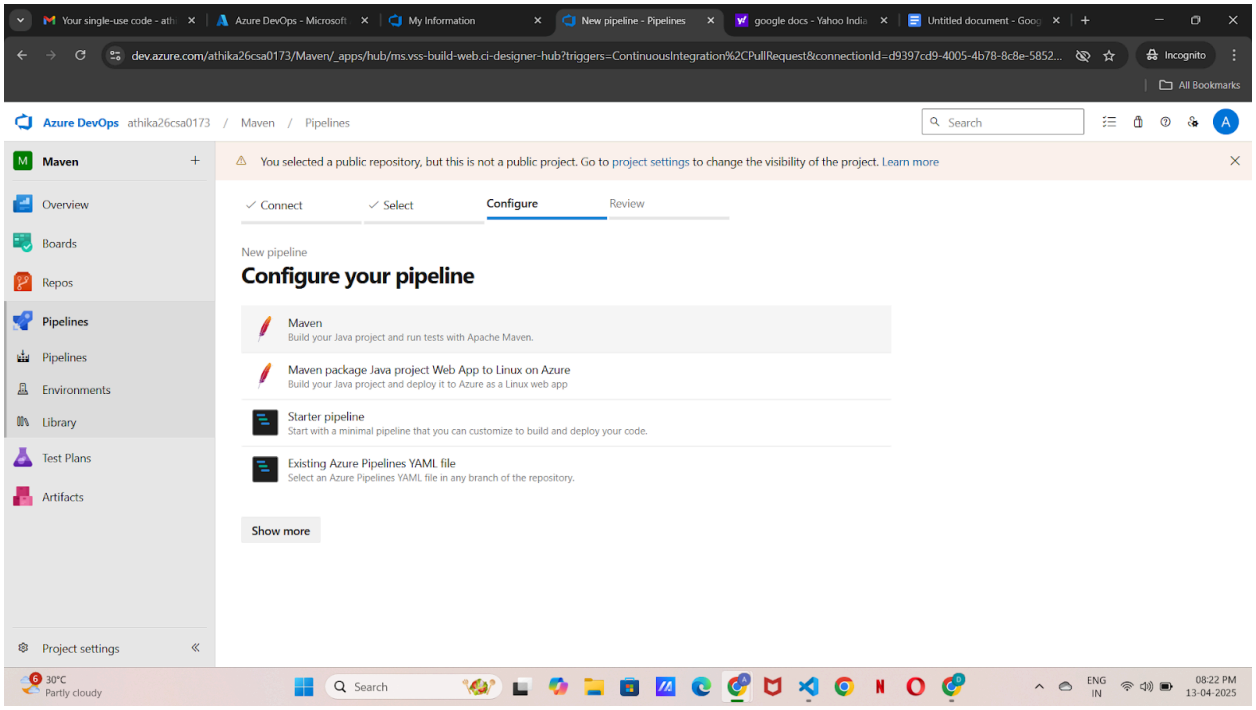


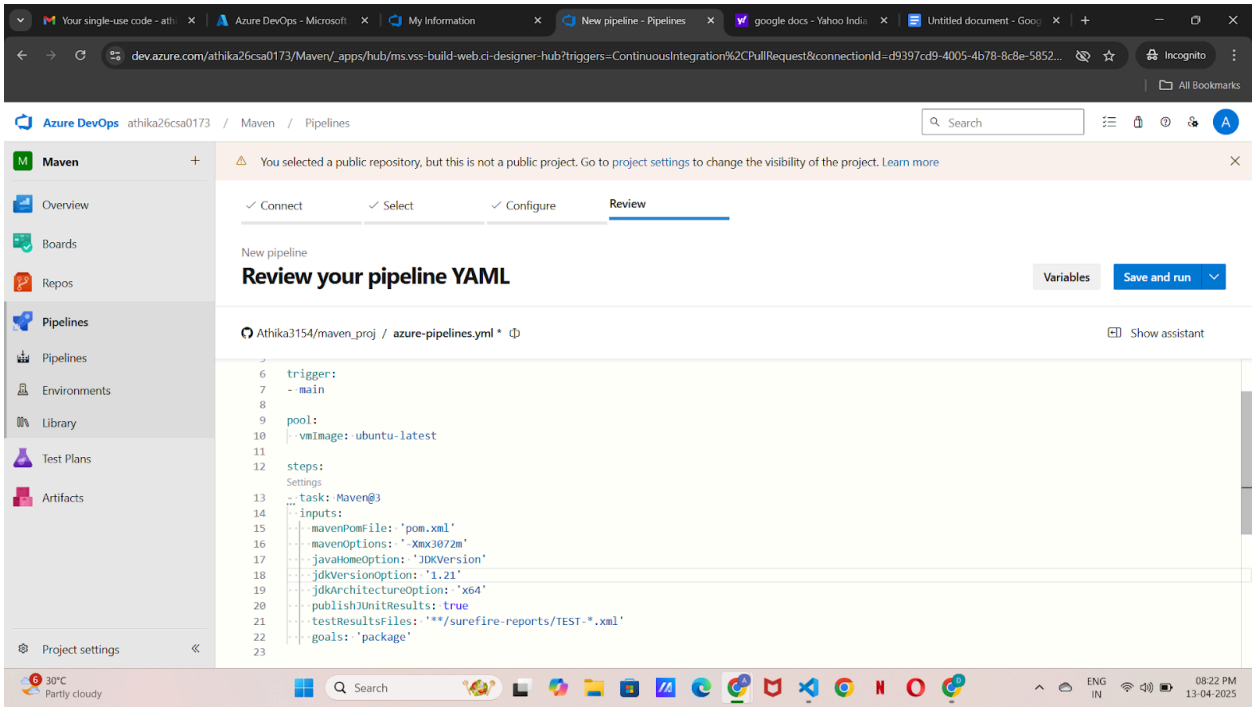
Step 2:Sign In to azure devops organization and create a project

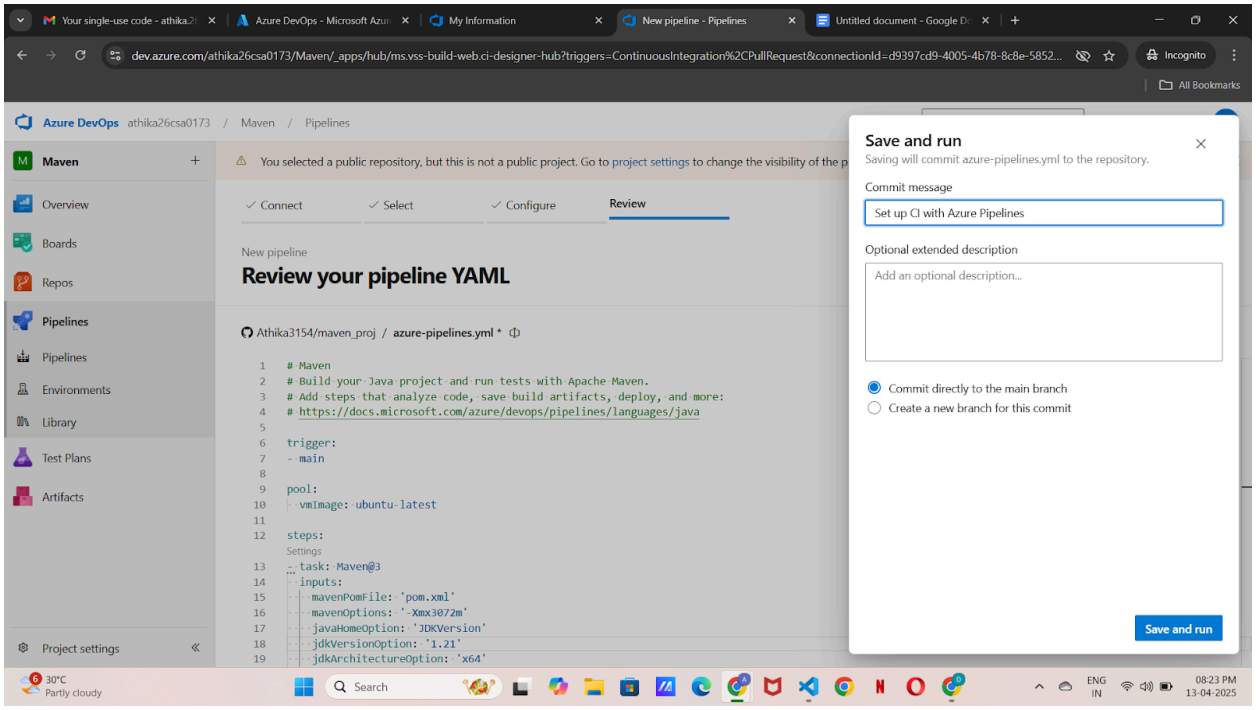
1. Create New Pipeline
2. Connect with GitHub
3. Select your project repository
4. Configure and Review Pipeline
5. Define the Maven goals and options, such as clean, compile, package, test, etc., based on
6. your project's needs.
7. Ensure you have the necessary plugins and dependencies configured in your pom.xml file.


Azure DevOps athika26csa0173 / Maven / Pipelines
Search
Maven
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Library
Test Plans
Artifacts
Project settings
Connect
Select
Configure
Review
New pipeline
Select a repository
Filter by keywords
My repositories
Athika3154/maven_proj
1h ago
srideepi/Devops
Apr 2
Athika3154/Intern_team1
Mar 21
Athika3154/CCSAssignment
Feb 18
Athika3154/demo003
Feb 18
Dharshini013/lattefy
Feb 14
Athika3154/Intern
Jan 31
Athika3154/WEBTECHNOLOGY4020


suggested installation of this GitHub App now
for these repositories:
All repositories
This applies to all current and future repositories owned by the resource owner. Also includes public repositories (read-only).
Only select repositories
Select at least one repository. Also includes public repositories (read-only).
Select repositories
Selected 1 repository.
Athika3154/maven_proj suggested
with these permissions:
Read access to metadata
Read and write access to checks, code, commit statuses, deployments, issues, and pull requests
Approve & Install
Reject
Next: you'll be directed to the GitHub App's site to complete setup.


Azure DevOps athika26csa0173 / Maven / Pipelines
Search
Maven
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Library
Test Plans
Artifacts
Project settings
You selected a public repository, but this is not a public project. Go to project settings to change the visibility of the project. Learn more
Connect
Select
Configure
Review
New pipeline
Configure your pipeline
Maven
Build your Java project and run tests with Apache Maven.
Maven package Java project Web App to Linux on Azure
Build your Java project and deploy it to Azure as a Linux web app
Starter pipeline
Start with a minimal pipeline that you can customize to build and deploy your code.
Existing Azure Pipelines YAML file
Select an Azure Pipelines YAML file in any branch of the repository.
Show more


Azure DevOps athika26csa0173 / Maven / Pipelines
Search
Maven
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Library
Test Plans
Artifacts
Project settings
You selected a public repository, but this is not a public project. Go to project settings to change the visibility of the project. Learn more
Connect
Select
Configure
Review
New pipeline
Review your pipeline YAML
Variables
Save and run
Athika3154/maven_proj / azure-pipelines.yml *
Show assistant
trigger:
- main
pool:
vmImage: ubuntu-latest
steps:
Settings
- task: Maven@3
inputs:
mavenPomFile: 'pom.xml'
mavenOptions: '-Xmx3072m'
javaHomeOption: 'JDKVersion'
jdkVersionOption: '1.21'
jdkArchitectureOption: 'x64'
publishJUnitResults: true
testResultsFiles: '**/surefire-reports/TEST-*.xml'
goals: 'package'

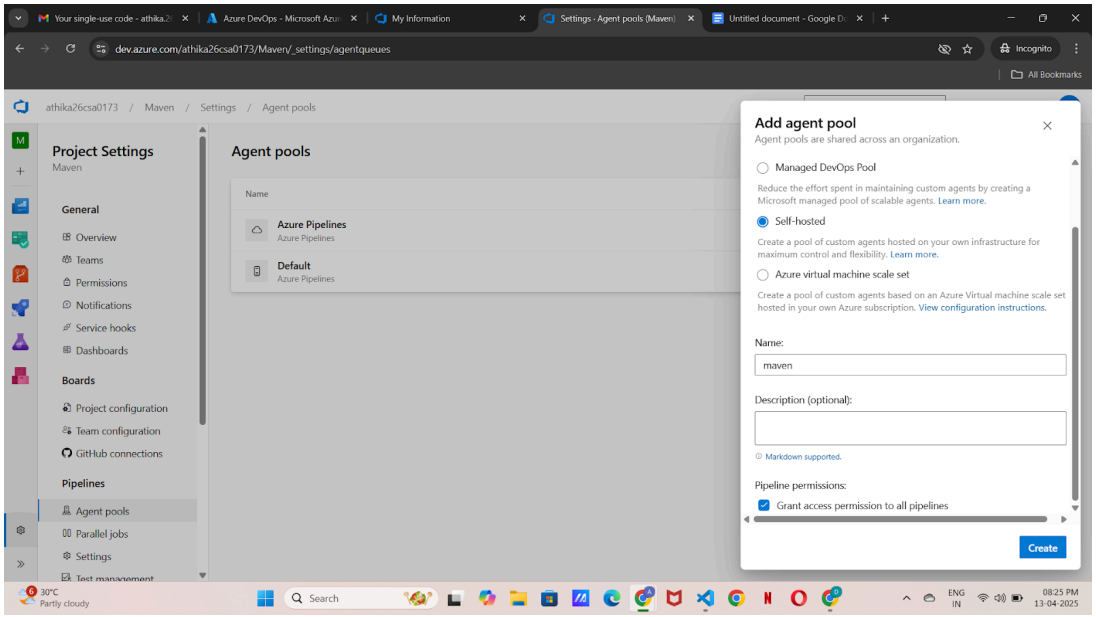


Step 3: Create agent pool and create agent name

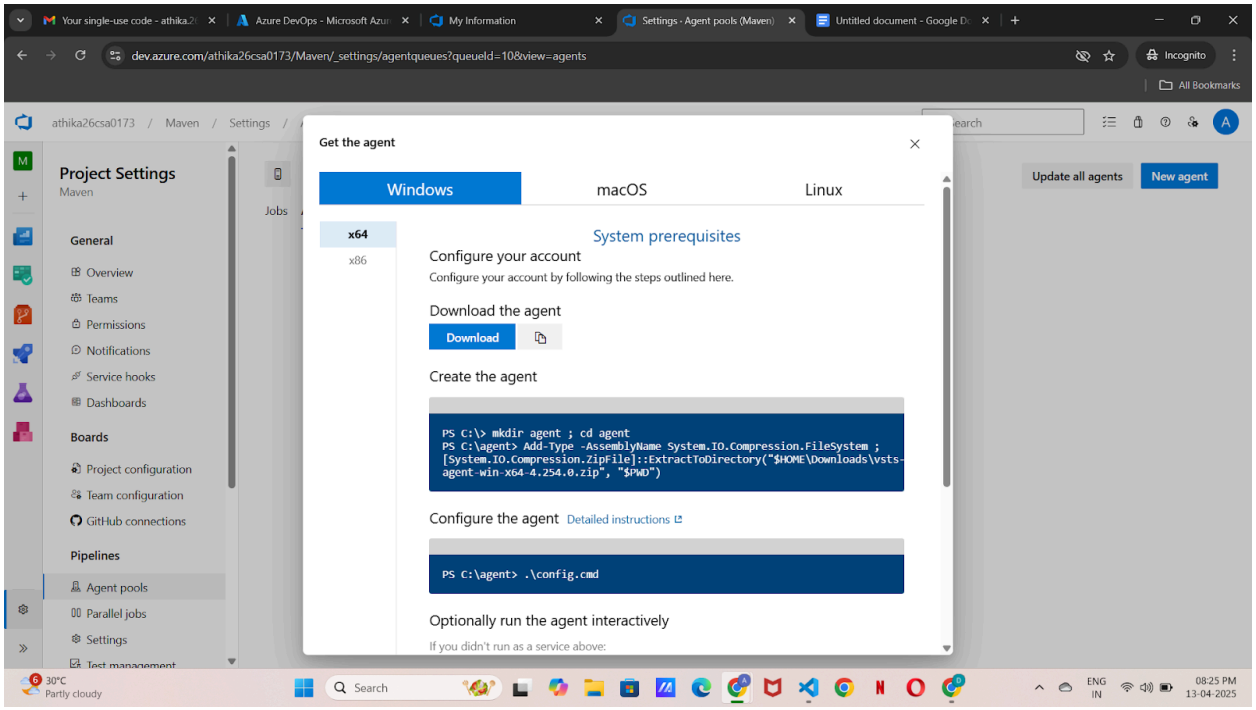
1.Open your web browser and log in to your Azure DevOps account.

2. Navigate to your Azure DevOps project and click on Project settings on the left side of the page.

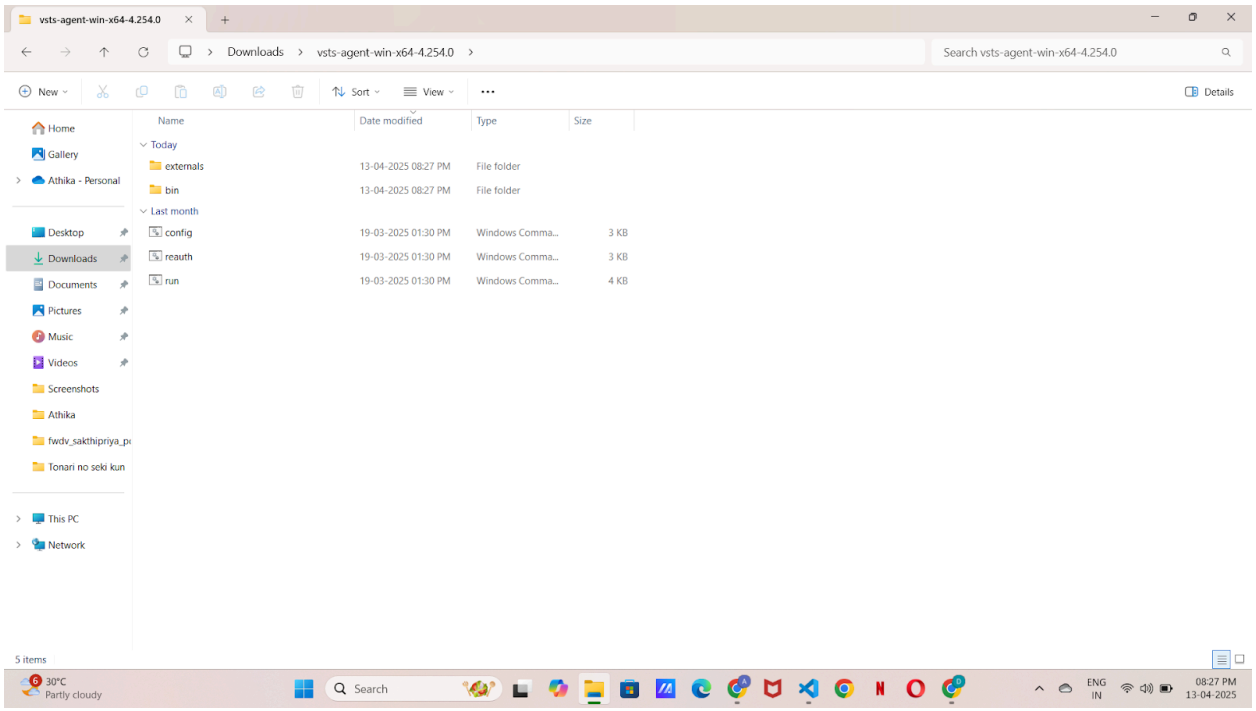
3. Click on Agent Pools under Pipelines and click on Add pool.



Step 4: Click on the created agent → Click on New agent. It will show the below screen. Just follow the steps to create the agent. Click on the Download button to download the agent.

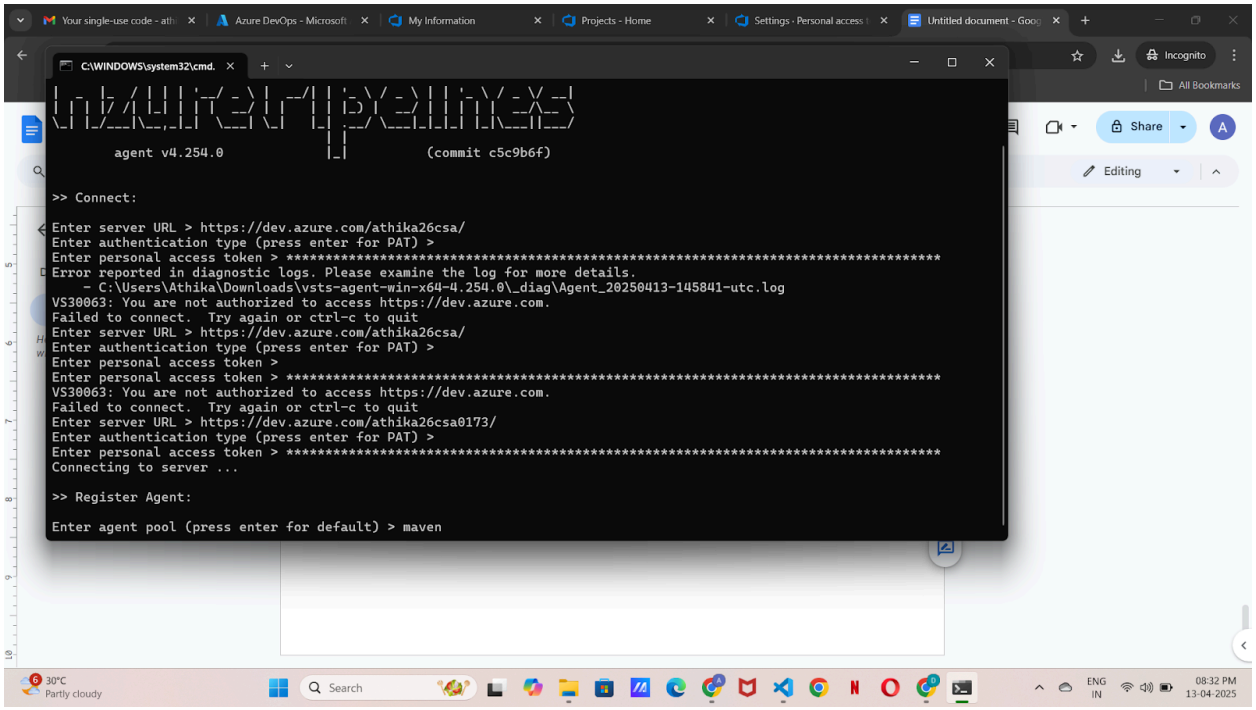


Step 5:Extract the downloaded agent package to a directory on your system.



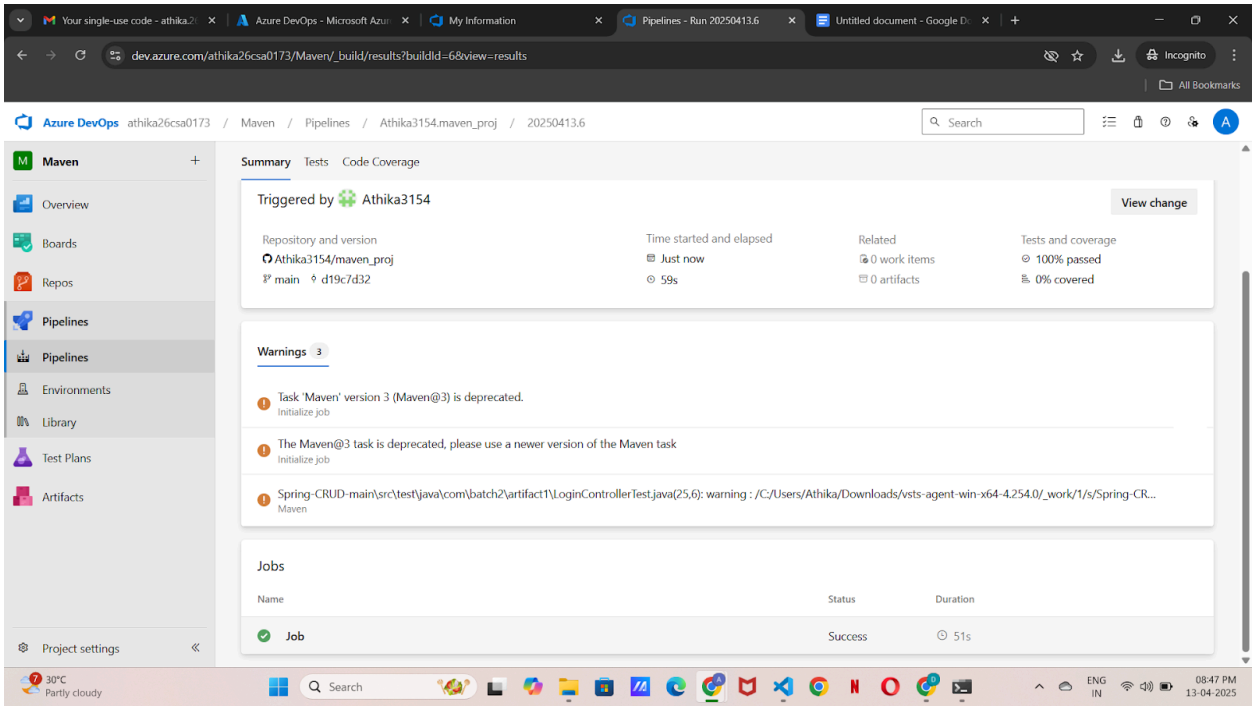
Step 6: Run the configuration script. The configuration script will prompt for your Azure Organization account URL and a personal access token (PAT).

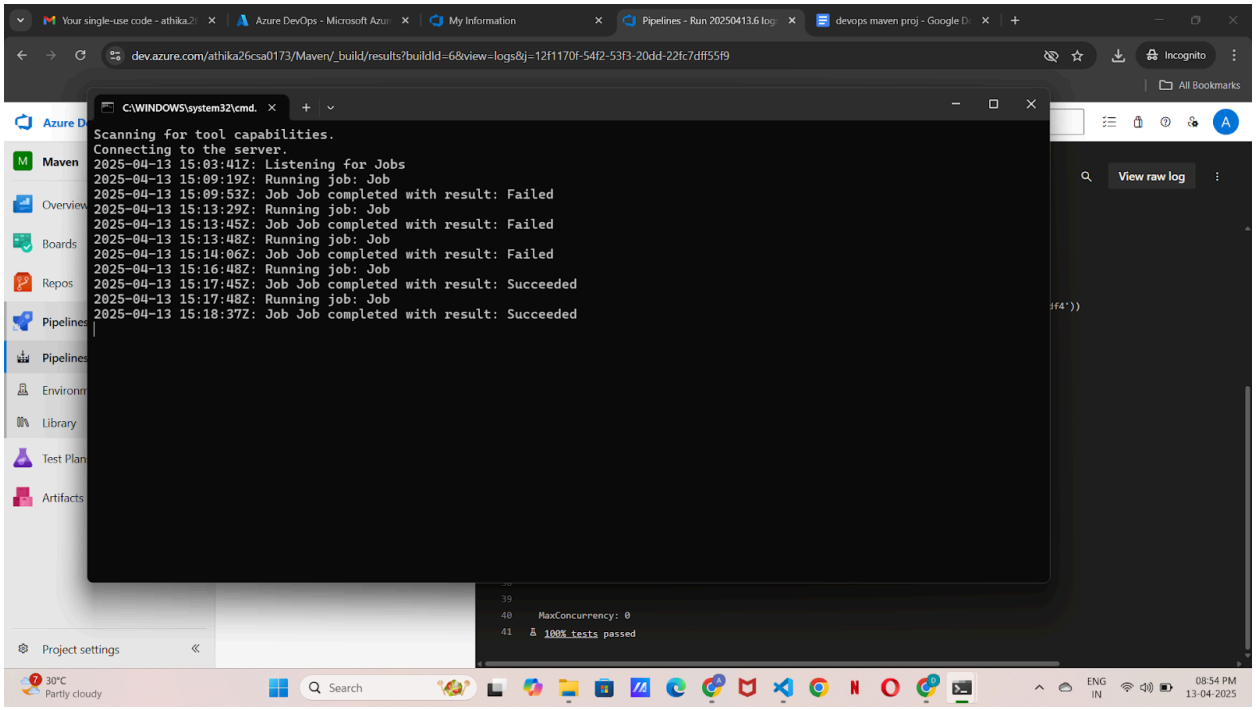
1. To generate a PAT, go to Azure DevOps account, click on the small icon on the left side of the profile picture in the top-right corner, and select Personal access tokens from the dropdown menu.

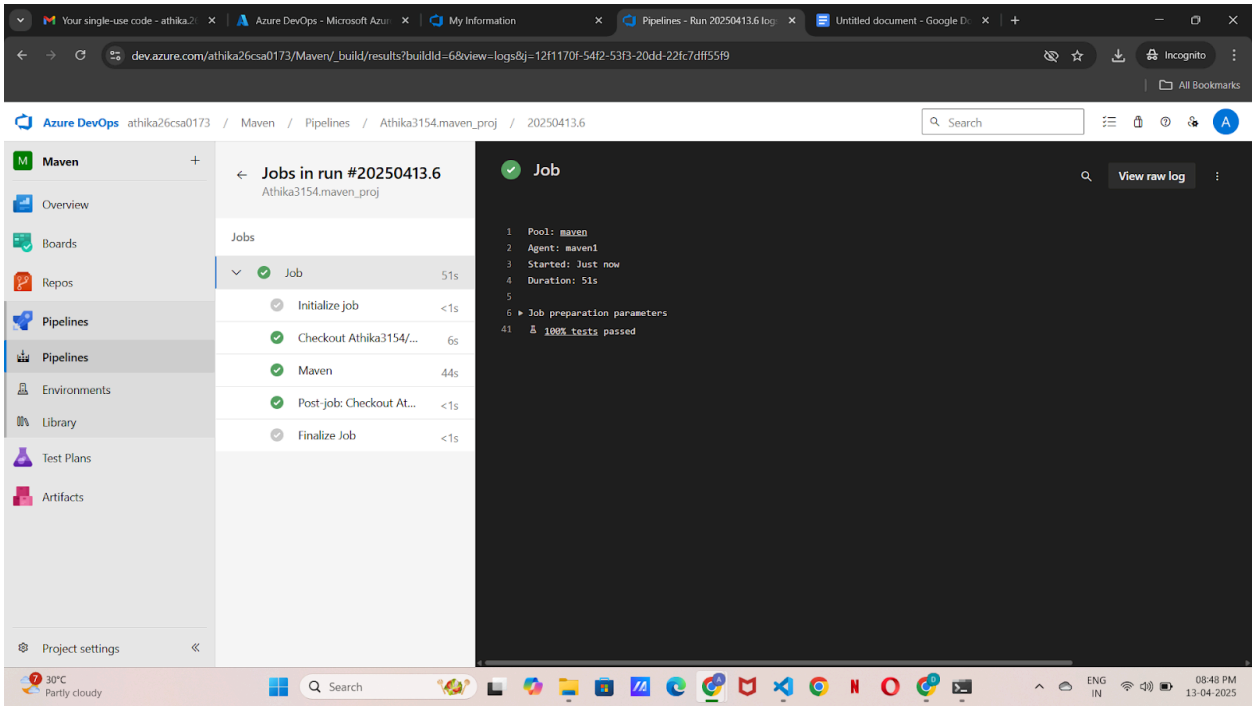


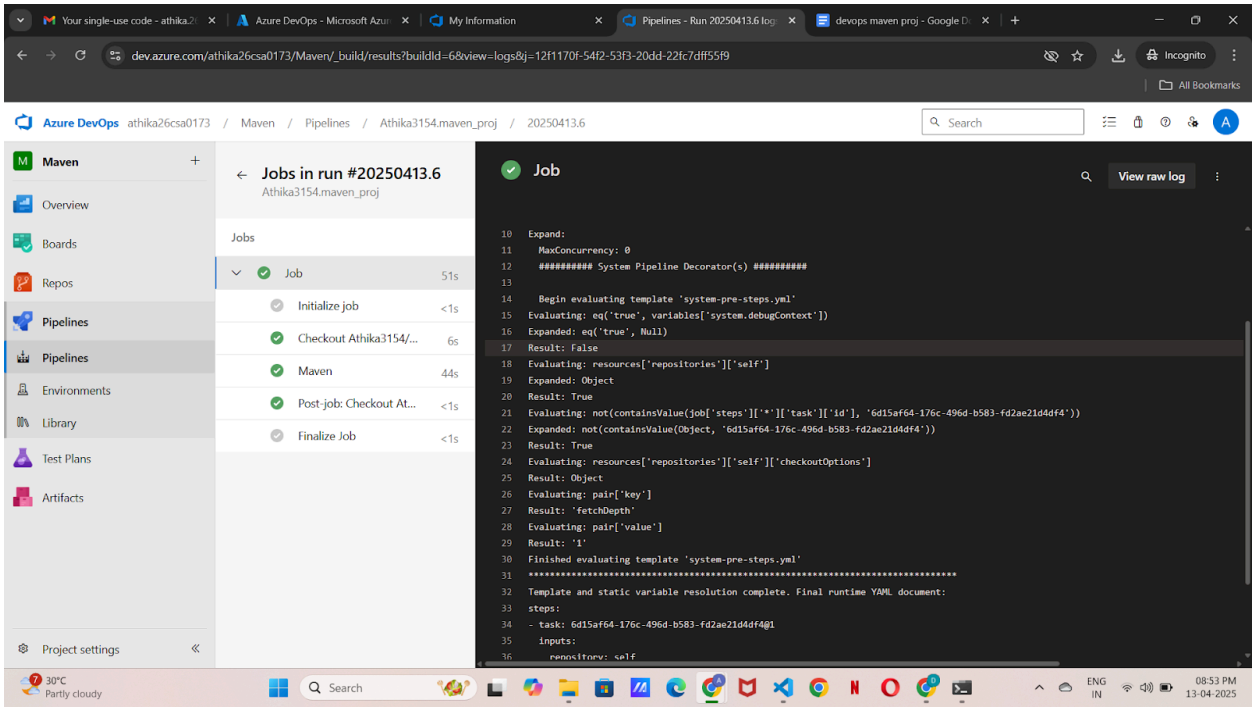
Step 7: Open the already created pipeline and edit the pipeline.

1. Change the pool name
2. Check the path of the pom.xml file
3. Again validate and run the pipeline
4. If Job runs with error check for the maven path setup
5. If job runs with a success, Test succeed


Azure DevOps athika26csa0173 / Maven / Pipelines / Athika3154.maven_proj / 20250413.6
Maven
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Library
Test Plans
Artifacts
Project settings
Summary
Tests
Code Coverage
Triggered by Athika3154
View change
Repository and version
Athika3154/maven_proj
main d19c7d32
Time started and elapsed
Just now
59s
Related
0 work items
0 artifacts
Tests and coverage
100% passed
0% covered
Warnings 3
Task 'Maven' version 3 (Maven@3) is deprecated.
Initialize job
The Maven@3 task is deprecated, please use a newer version of the Maven task
Initialize job
Spring-CRUD-main\src\test\java\com\batch2\artifact1\LoginControllerTest.java(25,6): warning : /C:/Users/Athika/Downloads/vsts-agent-win-x64-4.254.0/_work/1/s/Spring-CR...
Maven
Jobs
Name
Status
Duration
Job
Success
51s


C:\WINDOWS\system32\cmd.
Scanning for tool capabilities.
Connecting to the server.
2025-04-13 15:03:41Z: Listening for Jobs
2025-04-13 15:09:19Z: Running job: Job
2025-04-13 15:09:53Z: Job Job completed with result: Failed
2025-04-13 15:13:29Z: Running job: Job
2025-04-13 15:13:45Z: Job Job completed with result: Failed
2025-04-13 15:13:48Z: Running job: Job
2025-04-13 15:14:06Z: Job Job completed with result: Failed
2025-04-13 15:16:48Z: Running job: Job
2025-04-13 15:17:45Z: Job Job completed with result: Succeeded
2025-04-13 15:17:48Z: Running job: Job
2025-04-13 15:18:37Z: Job Job completed with result: Succeeded
View raw log
MaxConcurrency: 0
100% tests passed


Azure DevOps athika26csa0173 / Maven / Pipelines / Athika3154.maven_proj / 20250413.6
Search
Maven
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Library
Test Plans
Artifacts
Project settings
Jobs in run #20250413.6
Athika3154.maven_proj
Jobs
Job 51s
Initialize job <1s
Checkout Athika3154/... 6s
Maven 44s
Post-job: Checkout At... <1s
Finalize Job <1s
Job
View raw log
1 Pool: maven
2 Agent: maven1
3 Started: Just now
4 Duration: 51s
6 Job preparation parameters
41 100% tests passed


Azure DevOps athika26csa0173 / Maven / Pipelines / Athika3154.maven_proj / 20250413.6
Search
Maven
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Library
Test Plans
Artifacts
Project settings
Jobs in run #20250413.6
Athika3154.maven_proj
Jobs
Job 51s
Initialize job <1s
Checkout Athika3154/... 6s
Maven 44s
Post-job: Checkout At... <1s
Finalize Job <1s
Job
View raw log
10 Expand:
11 MaxConcurrency: 0
12 ######### System Pipeline Decorator(s) #########
14 Begin evaluating template 'system-pre-steps.yml'
15 Evaluating: eq('true', variables['system.debugContext'])
16 Expanded: eq('true', Null)
17 Result: False
18 Evaluating: resources['repositories']['self']
19 Expanded: Object
20 Result: True
21 Evaluating: not(containsValue(job['steps']['*']['task']['id'], '6d15af64-176c-496d-b583-fd2ae21d4df4'))
22 Expanded: not(containsValue(Object, '6d15af64-176c-496d-b583-fd2ae21d4df4'))
23 Result: True
24 Evaluating: resources['repositories']['self']['checkoutOptions']
25 Result: Object
26 Evaluating: pair['key']
27 Result: 'fetchDepth'
28 Evaluating: pair['value']
29 Result: '1'
30 Finished evaluating template 'system-pre-steps.yml'
31 **
32 Template and static variable resolution complete. Final runtime YAML document:
33 steps:
34 - task: 6d15af64-176c-496d-b583-fd2ae21d4df4@1
35 inputs:
36 repository: self

CODE:

LoginController.java:

```
package com.batch2.artifact1.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestParam;

import com.batch2.artifact1.domain.Login;
import com.batch2.artifact1.service.LoginService;

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

@Controller
public class LoginController {

    @Autowired
    private LoginService service;

    // Show login page on root or "/login"
    @GetMapping({"/", "/login"})
    public String showLoginPage(@RequestParam(required = false) String error, Model model) {
        if (error != null) {
            model.addAttribute("error", "Invalid username or password");
        }
        return "login"; // login.html
    }

    // Home page after login
    @GetMapping("/home")
    public String homePage() {
        return "home"; // home.html (Lattefy dashboard)
    }

    // Your Orders page
    @GetMapping("/urorder")
    public String orderPage() {
```

```
    return "urorderrec"; // urorderrec.html
}

// Favorites page
@GetMapping("/favorites")
public String favoritesPage() {
    return "favorites"; // favorites.html
}

// Drink Menu (default landing on Coffee tab)
@GetMapping("/drink-menu-coffee")
public String drinkMenuCoffeePage() {
    return "drink-menu-coffee"; // drink-menu-coffee.html
}

// Drink Menu - Chocolate category
@GetMapping("/drink-menu-choco")
public String drinkMenuChocoPage() {
    return "drink-menu-choco"; // drink-menu-choco.html
}

// Drink Menu - Others category
@GetMapping("/drink-menu-others")
public String drinkMenuOthersPage() {
    return "drink-menu-others"; // drink-menu-others.html
}

@GetMapping("/urpastorder")
public String urPastOrderPage() {
    return "urpastorder";
}

// Handle login form POST
@PostMapping("/login")
public String processLogin(@RequestParam String username,
                           @RequestParam String password,
                           Model model) {

    Login user = service.log(username, password);
```

```
        if (user != null) {
            return "redirect:/home";
        } else {
            model.addAttribute("error", "Invalid username or password");
            return "login"; // login.html with error
        }
    }
}
```

Login.html:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <title>Login - Lattefy Café</title>
    <style>
      body {
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        background-color: #fdf6f0;
        color: #4e342e;
        display: flex;
        justify-content: center;
        align-items: center;
        height: 100vh;
        margin: 0;
      }

      form {
        background-color: #fff8f0;
        padding: 30px 40px;
        border-radius: 12px;
        box-shadow: 0 8px 16px rgba(0, 0, 0, 0.1);
        width: 320px;
      }

      h1 {
        text-align: center;
        margin-bottom: 20px;
        font-size: 28px;
        color: #6d4c41;
```

```
}

label {
  display: block;
  margin-top: 10px;
  font-weight: bold;
}

input[type="text"],
input[type="password"] {
  width: 93%;
  padding: 10px;
  margin-top: 5px;
  margin-bottom: 15px;
  border: 1px solid #c7a17a;
  border-radius: 8px;
  background-color: #fff;
}

input[type="submit"] {
  width: 100%;
  padding: 10px;
  background-color: #a1887f;
  color: white;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-weight: bold;
}

input[type="submit"]:hover {
  background-color: #8d6e63;
}

p[th\\:if] {
  color: red;
  font-size: 14px;
  margin-top: 10px;
  text-align: center;
}
```

```
    div[align="left"] {
      margin-top: 20px;
      text-align: center;
    }

    a {
      text-decoration: none;
      color: #6d4c41;
      font-weight: bold;
    }

    a:hover {
      color: #3e2723;
    }
  </style>
 </head>
 <body>
  <form th:action="@{/login}" method="post">
    <h1>Login to Lattefy Café</h1>

    <label for="username">Username</label>
    <input type="text" id="username" name="username" autofocus="autofocus" />

    <label for="password">Password</label>
    <input type="password" id="password" name="password" />

    <input type="submit" value="Login" />

    <p th:if="${error}" th:text="${error}" style="color: red;"></p>
    <div align="left">
      <h3><a th:href="@{/register}">Don't have an account? Register</a></h3>
    </div>
  </form>
 </body>
</html>
```

Login.java:

```
package com.batch2.artifact1.domain;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name = "login")
public class Login {
    @Id
    public String username;
    public String password;

    public Login() {
    }

    public Login(String username, String password) {
        this.username = username;
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

LoginRepo.java:

```
package com.batch2.artifact1.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.batch2.artifact1.domain.Login;

@Repository
public interface LoginRepo extends JpaRepository<Login, String> {
   Login findByUsernameAndPassword(String username, String password);
}
```

LoginControllerTest.java:

```
package com.batch2.artifact1;

import com.batch2.artifact1.domain.Login;
import com.batch2.artifact1.service.LoginService;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.test.web.servlet.MockMvc;

import static org.mockito.Mockito.when;
import static org.mockito.ArgumentMatchers.anyString;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;

@SpringBootTest
@AutoConfigureMockMvc
public class LoginControllerTest {

   @Autowired
```

```
    private MockMvc mockMvc;

    @MockBean
    private LoginService loginService;

    @Test
    public void testSuccessfulLogin() throws Exception {
        Login user = new Login();
        user.setUsername("admin");
        when(loginService.log(anyString(), anyString())).thenReturn(user);

        mockMvc.perform(post("/login")
                .param("username", "admin")
                .param("password", "password"))
                .andExpect(status().is3xxRedirection())
                .andExpect(redirectedUrl("/home"));
    }

    @Test
    public void testFailedLogin() throws Exception {
        when(loginService.log(anyString(), anyString())).thenReturn(null);

        mockMvc.perform(post("/login")
                .param("username", "invalidUser")
                .param("password", "wrongPass"))
                .andExpect(status().isOk())
                .andExpect(view().name("login"))
                .andExpect(model().attributeExists("error"));
    }
}
```

Application Properties:

```
spring.application.name=artifact1
spring.datasource.url=jdbc:mysql://localhost:3306/demo
spring.datasource.username=root
spring.datasource.password=12345
server.port=8081
```

Pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.4.3</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <groupId>com.batch2</groupId>
    <artifactId>artifact1</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>artifact1</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>17</java.version>
    </properties>

    <dependencies>
        <!-- Web + REST -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <!-- Thymeleaf template engine -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>

        <!-- JPA (Hibernate) -->
```

```
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <!-- MySQL JDBC -->
    <dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <scope>runtime</scope>
    </dependency>

    <!-- Lombok (boilerplate remover) -->
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.30</version>
        <scope>provided</scope>
    </dependency>

    <!-- Spring Boot Actuator (optional monitoring) -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>

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

<build>
    <plugins>
        <!-- Maven Compiler Plugin -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
```

```
            <version>3.11.0</version>
            <configuration>
               <source>${java.version}</source>
               <target>${java.version}</target>
               <annotationProcessorPaths>
                  <path>
                     <groupId>org.projectlombok</groupId>
                     <artifactId>lombok</artifactId>
                     <version>1.18.30</version>
                  </path>
               </annotationProcessorPaths>
            </configuration>
         </plugin>

         <!-- Spring Boot Maven Plugin -->
         <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <version>3.4.3</version>
         </plugin>
      </plugins>
   </build>
</project>
```

RESULT:

Thus to run a regression test using maven build pipeline in azure is successful.