# CA1 – Lab Portfolio

## Overview

The purpose of this assignment is to become familiar and comfortable with using different tools of my choice that are used in DevOps. I have used tools that I am in some way familiar and other tools that I have never used before. Please find my more detailed information within.

Contents

[CA1 – Lab Portfolio 1](#_Toc327604282)

[Overview 1](#_Toc1414548581)

[Lab Instructions: 2](#_Toc1072706251)

[Lab 1 2](#_Toc1676637646)

[Lab Details: 3](#_Toc345957836)

[Conclusion 3](#_Toc354155767)

[Lab 2 4](#_Toc453575714)

[Conclusion 4](#_Toc1456583635)

[4](#_Toc1579260555)

[Lab 3: 4](#_Toc1498256958)

[Conclusion 4](#_Toc549804781)

[4](#_Toc1730783520)

[Submission and Deadlines 4](#_Toc239148794)

[Formatting 4](#_Toc2072860508)

[Submission and Grading Details 4](#_Toc817766710)

[Plagiarism 4](#_Toc265951154)

[Appendencies 4](#_Toc1288199933)

[Fig 1.1 GitCloneCi 900010600 is my UserName 4](#_Toc864753575)

[Fig 1.2 Intellij after Clone from GitBash 5](#_Toc330842449)

[Fig 1.3 New Feature Aidan 5](#_Toc1227115893)

[Fig 1.3.1 Commit 6](#_Toc140461447)

[Fig 1.4 New Feature(BitBucket) 6](#_Toc294558532)

[Fig 1.4.1 No update to this Folder(Course Work) 6](#_Toc322863693)

[Fig 1.4.2 Course Work Other Updated 7](#_Toc1934721568)

[Fig 1.5 Feature Updates in BitBucket 7](#_Toc2006619653)

[Fig 1.5.1 Side by Side 7](#_Toc191694413)

[Fig 1.6 CLI Commit 8](#_Toc871859536)

[Fig 1.7 Pull Request 8](#_Toc91700769)

[Fig 1.8 CircleCI 9](#_Toc62922158)

## Lab Instructions:

You must choose the tools that you want to work with yourself. Choosing new tools may take some time and effort, so allow for time to be spent on identifying and learning new tools/technologies.

**Lab 1**  
This lab must describe the use of a **source code (version control) management tool** as part of a DevOps pipeline. Note: **GitHub** and any **One** other sample should be mentioned in the conclusions.

Your lab should show a project that is under version control, has branches and commits. You should demonstrate changing code in a code editor as well as through a terminal window, with the results visible in the main/master repo.

The lab must focus on higher level aspirations rather than the detailed step by step. The conclusions section must be 2-4 pages. The page count does not include references or images which must be included in the appendices.

**You need:** A java (or other language) **project**, that is stored in GitHub (or other **version control tool**) that I can clone and run on my machine. (You need to make me a collaborator in your GitHub project). You need to show that you can **pull** and **push** changes from your local machine to your GitHub repo.

For this lab I started by following the learnings in each lecture, Using GITHub initially I got as far as installing Jenkins and that is as far as I got, I attempted and researched several different ways by also attempting to use VMWare to then install it as per the other supplied user guide. This also led me to an issue that I tried to overcome but due to time constraints I also didn’t get far with that. Plan B was put into place as I wanted to progress. Please note all of this is on my work laptop so I am unable to give access to the source control in this case BitBucket.

## Lab Details:

In my work Example I cloned an existing working Java Project that was stored in BitBucket by using the Git Clone command in GITBASH [Fig 1.1 Git CloneCi](#_Fig_1.1_GitCloneCi) from the GITRepo folder, I use Intellij as an editor to open the cloned project [Fig 1.2 Intellij](#_Fig_1.2_Intellij). I then proceeded to create 2 new branches with the intention of ensuring I don’t conflict any changes within the Master(I need to try to avoid the work Project being compromised by my created project and pipeline). The branches are called “Course” and “Course Other”. I navigated to the Test resources and created a new Feature call [Aidan.feature](#_Fig_1.3_New). I then committed the feature and observed the feature is now available within BitBucket in the correct Branch [Fig 1.4](#_Fig_1.4_New). I then updated the feature by removing some functionality i.e. some additional tests. I committed these changes in Intellij [Fig 1.3.1 Commit](#_Fig_1.3.1_Commit). I then confirmed within Bitbucket that the History of the file has been correctly updated and BitBucket also allowed me to see the difference with a green(inserted)/red(deleted) or side by side Fig 1.5

Using GITBash I was able to update the feature “Aidan” with some comments [Fig 1.6 CLI Commit](#_Fig_1.6_CLI) I then created a Pull Request [Fig 1.7](#_Fig_1.7) with 2 Approvers from the CourseWork Other to CourseWork(difference being the new “Aidan.feature” file

## Conclusion

Within work we use BitBucket so it was interesting to learn GITHUB during the course. I believe they both go back to a git server(work wise anyway). Over the past 6 months as a manager I have appreciated the importance of a source code control tool more from an overseeing point of view rather than writing and updating/merging of code, it allowed me to track what specific people had been working on and the history of the code from inception. It allows the dev team and management to see the full story from creation to updates through different versions and branches. From a version source control point of view and maybe I am swayed by my current position where there is an option to use GITHUB yet everyone uses BITBucket but I would have to say I find them equal. From a UI point of view there isn’t much difference either and as Bitbucket is connected to JIRA it makes sense to use this in a corporate setup where JIRA is our bread and butter. If I was going it alone and again maybe because I have no spent more time with GitHub I think I may chose it. It also appears to me as a student as the impression is that the public repositories would be a fantastic resource as learning increases. It is slightly easier to navigate around, I like the marketplace with the apps and the categories for Add Ons, the actions I am not yet sure of. It may also be that I associate BitBucket with work and it feels more locked down due to the lack of or non-existent of public repositories. Collaboration or more like HELP that I will need will certainly sway me towards GitHub but maybe when I become familiar it may feel like there is a lot of unnecessary noise associated with it and thus when I grow up I will seek the comfort of BitBucket.

GO BACK TO and include additional GITHUB details

**Lab 2**   
This lab must describe the use of **Test** **Frameworks** and supporting software as part of a DevOps pipeline. Note: Any 2 test frameworks should be discussed/contrasted in the conclusions

The Postman or a. n. other tool should be used to assist with automated testing

The lab must focus on higher level aspirations rather than the detailed step by step. The conclusions section must be 2-4 pages. The page count does not include references or images which must be included in the appendices.

You need to choose a Test Tool that you wish to use, get it up and running, you can you your GitHub project from the first lab as your subject, or use a open source project)

**Note:** Tests do not have to actually test the project, they can simple be outputting a message to say that a test has been run. We are not interested in testing the code in full, but we want to experiment and utilise the tools that can be used for automated testing.

Maven

Within Intellij Maven test tooling is available and can be run as and when needed, I ran the clean Fig 2.0 Maven

Conclusion

As a Tester the majority or my career I would consider myself very familiar with Testing and the tools that go with it, having said that I went the managerial role instead of automation so my exposure to testing tools is a bit limited. I am very familiar with Postman with a strong focus on API testing. Postman allows testing of structured data agreed using something like swagger which enables testing from Point A to B without a front end that may only reside in a customer's external system thus allowing processing of data from and to the external systems through specific APIs. Postman allows for easy importing of project/files that may include what is known as collections of files that focus on functionality i.e. POST ing or sending info to another system internally or externally using specified inputs and reply/responses, the other 2 most popular are PUT(Updating) GET(which is retrieving) and DEL(usually cancelling rather than deleting. The tool allows communication between different calls and response and the ability to configure what happens when a call is made. Collections (groupings of calls) can be saved and run in a specified order as needed or as part of an automated test.

Selenium is a tool that I would be very interested in learning more about as it appears to be the leading tool when it comes to webpage testing, on the surface after running the described test in one of the labs it does seem appealing and relatively straightforward. The tool allows for specific browsers to be tested and the IDE allows record and play back of users actions. Tests that are written in Gherkin using cucumber flows are interchangeable and assist in building suites of automated tests

Summary

Postman is the tool of choice if testing APIs whereas if testing Webpages I would have to invest in the time to use something like Selenium but either should work in conjunction with a Test Framework that calls out what will be tested and what will be used to be tested and how it will be tested. Automated Testing built from such a structure should be more robust and provide a better overall solution.

**Lab 3:**  
This lab must describe the use of a **continuous integration (CI) tool with automated** **Pipelines** and **supporting** **software** as part of a DevOps pipeline.  
Note: The practical implementation must include steps for build, test, package, deploy.

The tool must perform an automated test on code pushed to the repository

Any 2 tools (e.g. GitHub Actions, Jenkins, Circle CI) should be contrasted in the conclusions section.

Tools should be utilized in practice for the purposes of experimentation. (not just described or reported)

The lab must focus on higher level aspirations rather than the detailed step by step. The conclusions section must be 2-4 pages. The page count does not include references or images which must be included in the appendices.

You should have: A java (or other language) **project**, with **unit tests**, built with maven (or other **build** **tool**) that is stored in GitHub (or other **version control tool**), this is being **automatically** build but a **CI tool** (Jenkins, Git Hub actions or other) and will **run your test suite** when a change is committed to the code. (You need to make me a collaborator in your GitHub project).

Conclusion

I installed Circle CI but didn’t spend a lot of time using it but my initial thoughts were it looked great and appeared to be very user friendly(at least initially), once it asked for credit card details it put me off, As a tester the testing functionality looked impressive. It does look more like a publishing tool than a coding CI tool but that might be why I like it, it allows someone relatively new to get up and running with ease. I was able to easily create a CICD pipeline and login with GITHUB to allow seamless integration. I may add credit card details and see what it is really like. I loved the Status details and the apparent ease of creating the components very quickly [Fig 1.8](#_Fig_1.8_CircleCI). I played with the connecting of code and setting up a project linked to github with ease. The User guide was also impressive stressing the ease of adding any VCS but calling out Bitbucket and GitHub. The more I use it the more I am warming to it. I get the impression that the easiness of it all may annoy the more experienced or those who like to conifgure the tool themselves in a specific way.

Jenkins

I have worked with Jenkins for a few years now and it has been part of each company's strategy, plans or future. From a testing point of view, we use it to mostly run automated scripts within the tool as needed or scheduled within Jenkins e.g. Nightly Regression Test Scripts, Jenkins is reliable and once you have the correct role it can be easily setup. We have a lot of tests that run for different countries on different environments, Jenkins allows cloning/copying of jobs that can be configured to run in a different env/country by only updating some parts of the cloned Jobs instead of starting from scratch. It allows for periodic Builds or adhoc builds to be run prior to or post a build or based on a successful build and only run the next build if the tests complete and are passed. The Post step test steps are also very useful to confirm a build was successful prior to any new round of testing commencing. It helps identify Regression Issues related to the latest Build. Configuring Jenkins jobs in parts is initially tricky to grasp but once you become familiar with the expressions, they can be very useful and very specific. The notification functionality is very useful and configurable as it can alert certain people without spamming everyone. It is important not to spam anyone i.e. only send an email when a Failure occurs. The ability to only run builds or test if a previous build or test is passed allows for quick responses and avoids wasted time. The Outpus from the job e.g. Reports also allows for details to be included from the logs identifying where exactly the Build/Tests failed or when the expected result differed. These parameters are easily added to a Jenkins job as and when needed. Plugins are plentiful and as it is most likely top of the CI tools its contributors help it improve. Most important of all has to be that it is reliable and FREE.

I installed GITLab and I also found it user friendly

## Submission and Deadlines

* All the work on the lab will be submitted through Blackboard
* The labs must be submitted in **one** document at the end of the module.
  + Final Date: **Sunday 20th November 2022 @ 23:59pm**
* NOTE: Lab 1 may be submitted any time before Sunday 23rd October, for formative feedback only.

## Formatting

Please use following formatting throughout your paper.

* Word Doc **ONLY**
* 12pt font
* Justified text (no ragged edges)
* Headers have a maximum size of 16pt font
* Tables & Figures must be numbered starting at 1 going through the whole document.
* Tables and figures must have their caption below the table/figure
* **References should use the Harvard UL Referencing system.**

**See:** [**https://library.lyit.ie/wp-content/uploads/2020/05/Harvard-Reference-Quick-Guide.pdf**](https://library.lyit.ie/wp-content/uploads/2020/05/Harvard-Reference-Quick-Guide.pdf)

## Submission and Grading Details

This is the first part of your CA and the 3 labs together are worth 60% of your overall mark.

* Lab 1 (20%)
* Lab 2 (20%)
* Lab 3 (20%)

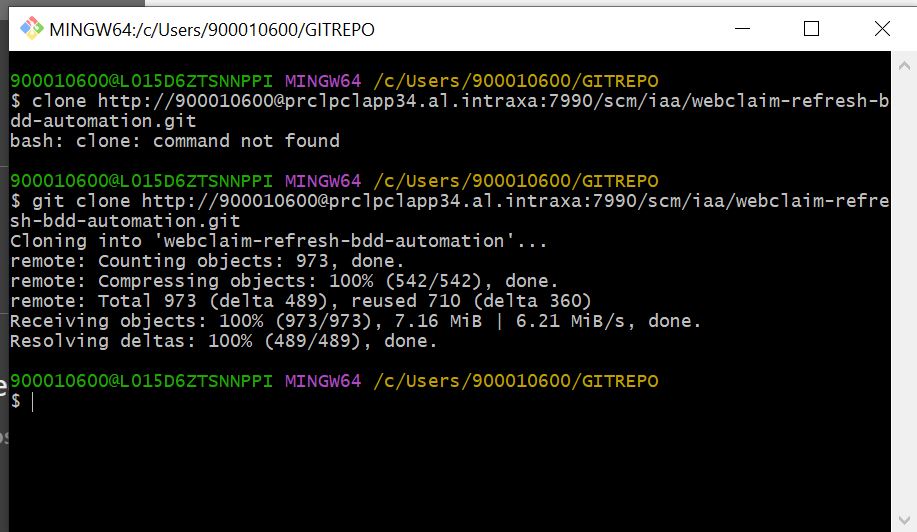
Each Lab will be marked as follows:

|  |  |
| --- | --- |
| **Section** | **Marks** |
| Aims/Description | 10 |
| Method | 15 |
| Practical Implementation | 20 |
| Results | 20 |
| Conclusion | 35 |
| **Total** | **100** |

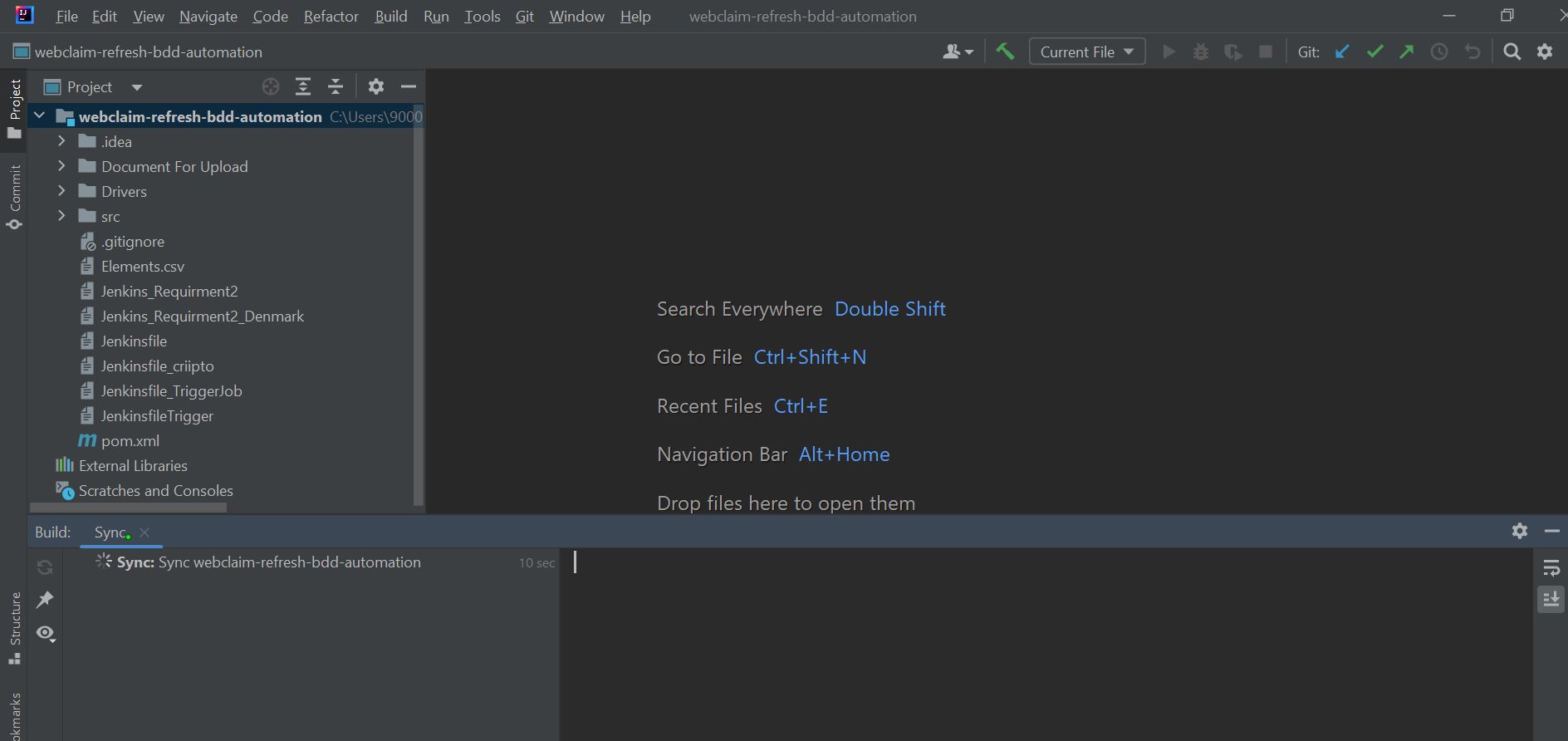
## **Plagiarism**

Stealing someone else's writing will not be tolerated. Do not copy text from other sources. For example, do not simply copy the project website's description into your description. **You must reference your work.**

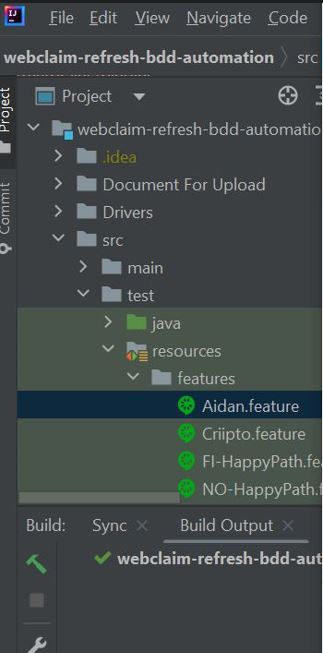
### Appendencies



### Fig 1.1 GitCloneCi 900010600 is my UserName



### Fig 1.2 Intellij after Clone from GitBash



### **Fig 1.3 New Feature Aidan**

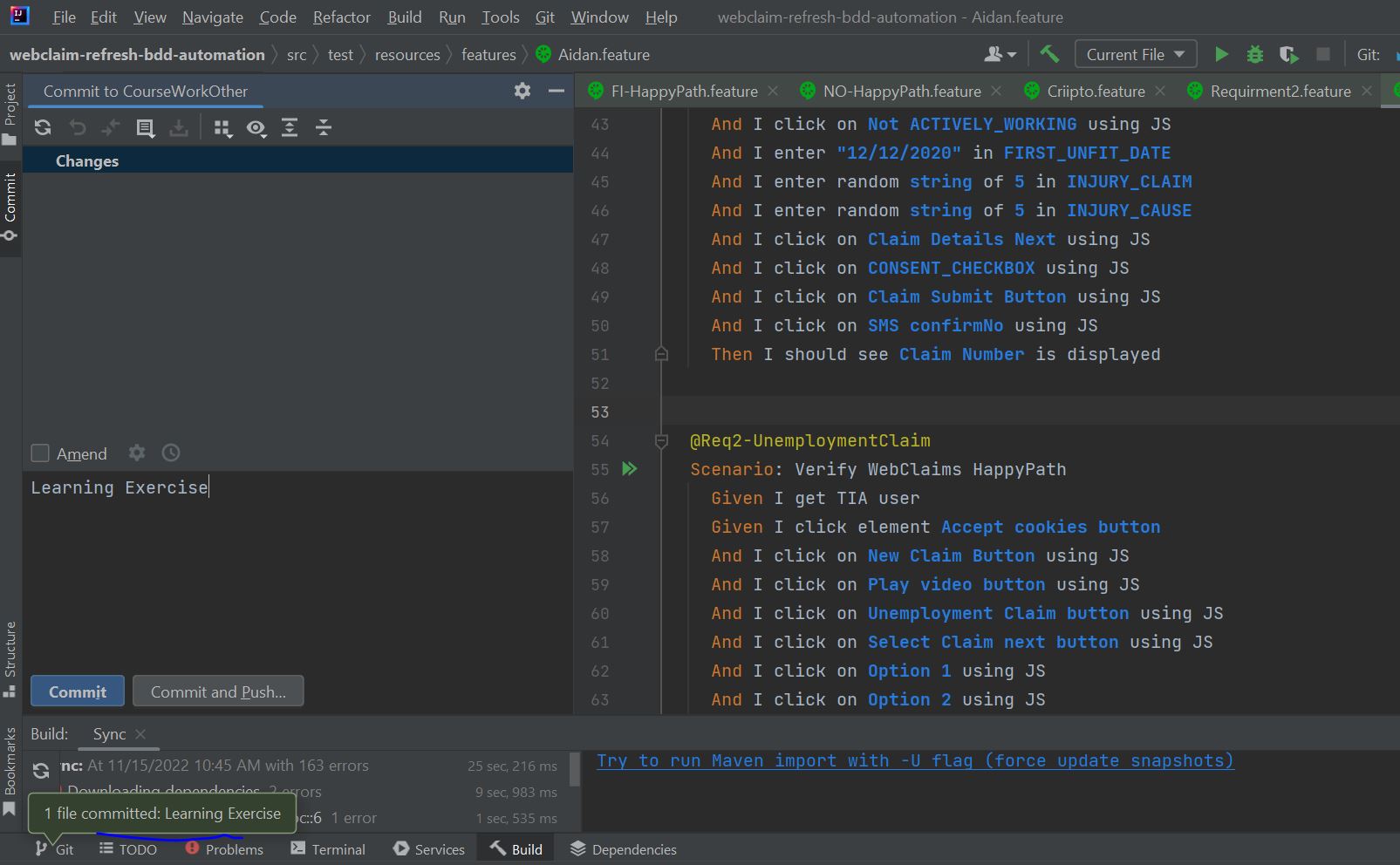


Fig 1.3.1 Commit

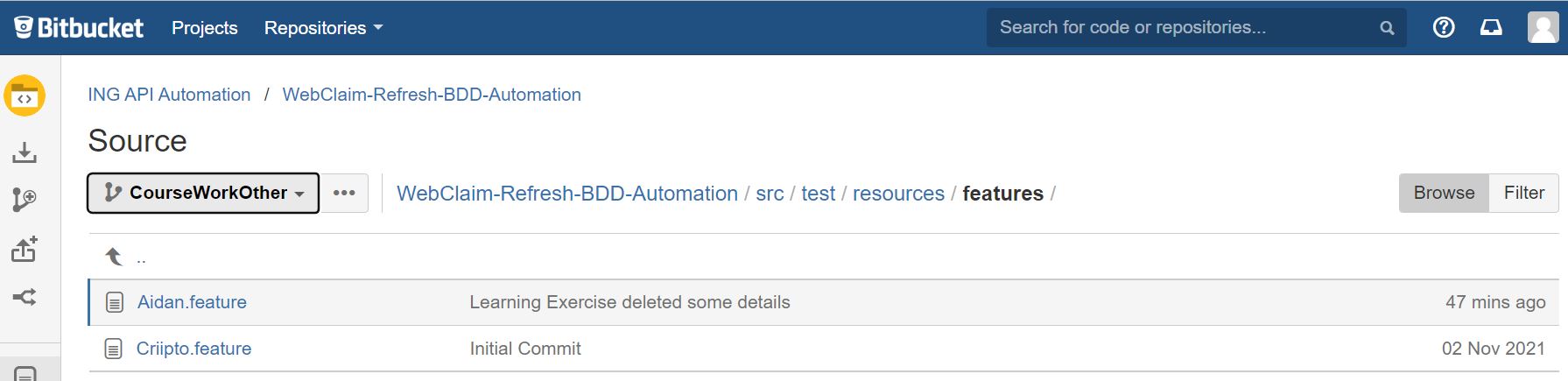


Fig 1.4 New Feature(BitBucket)

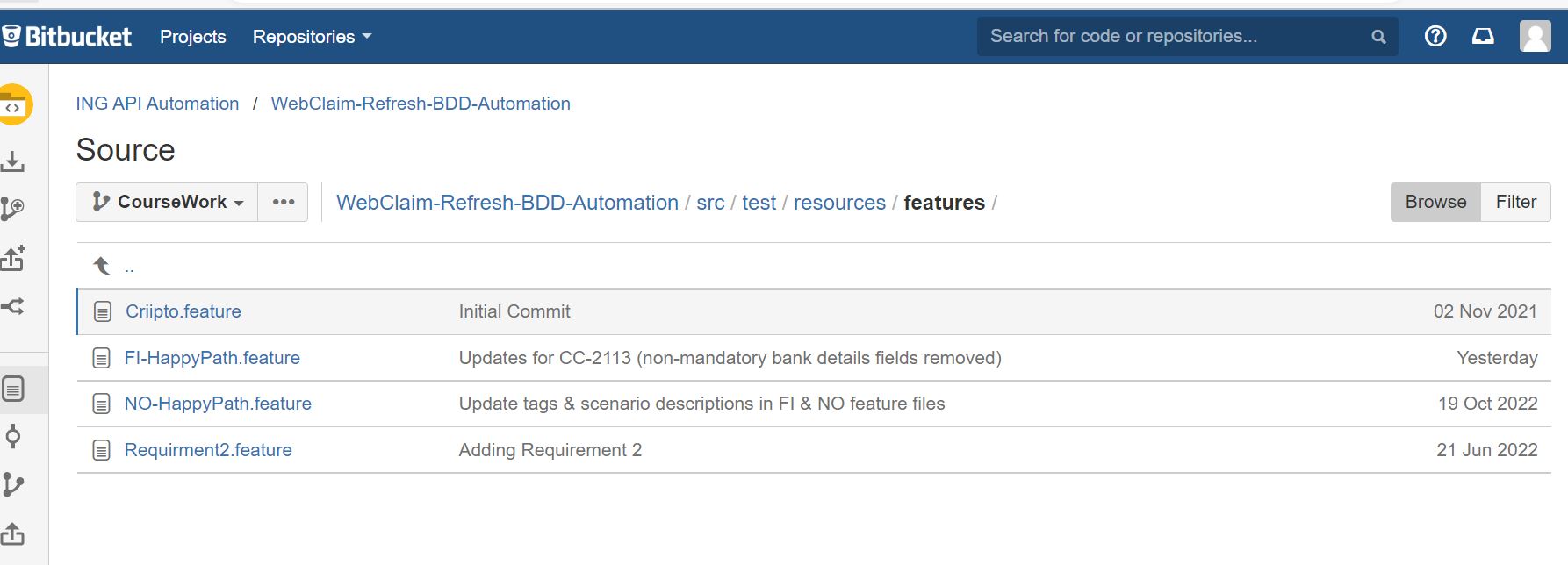


Fig 1.4.1 No update to this Folder(Course Work)

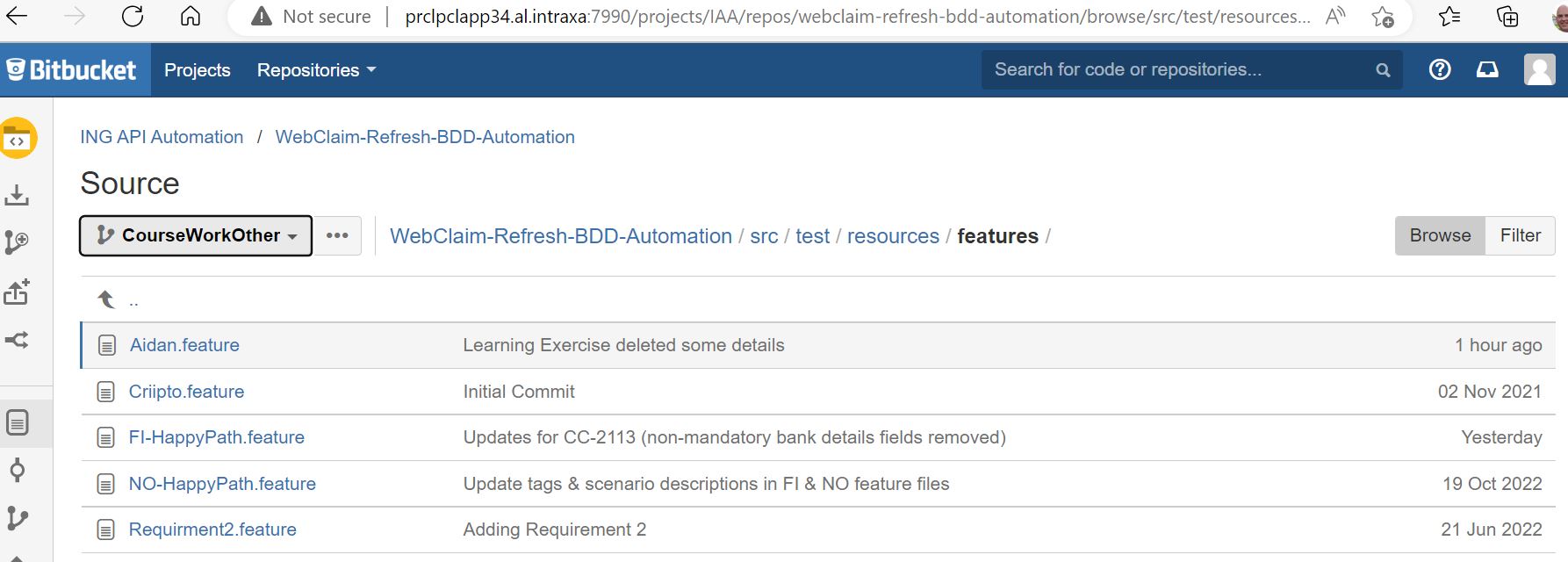
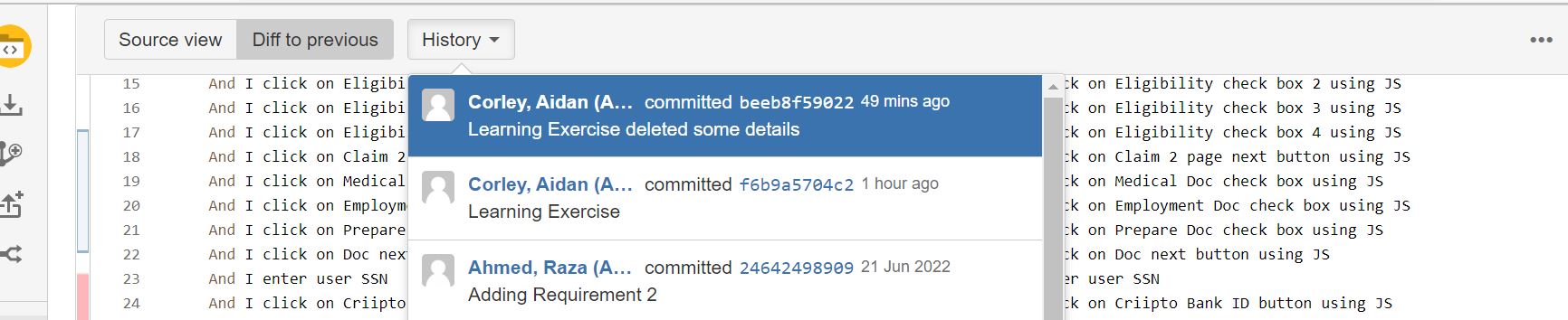


Fig 1.4.2 Course Work Other Updated



### Fig 1.5 Feature Updates in BitBucket

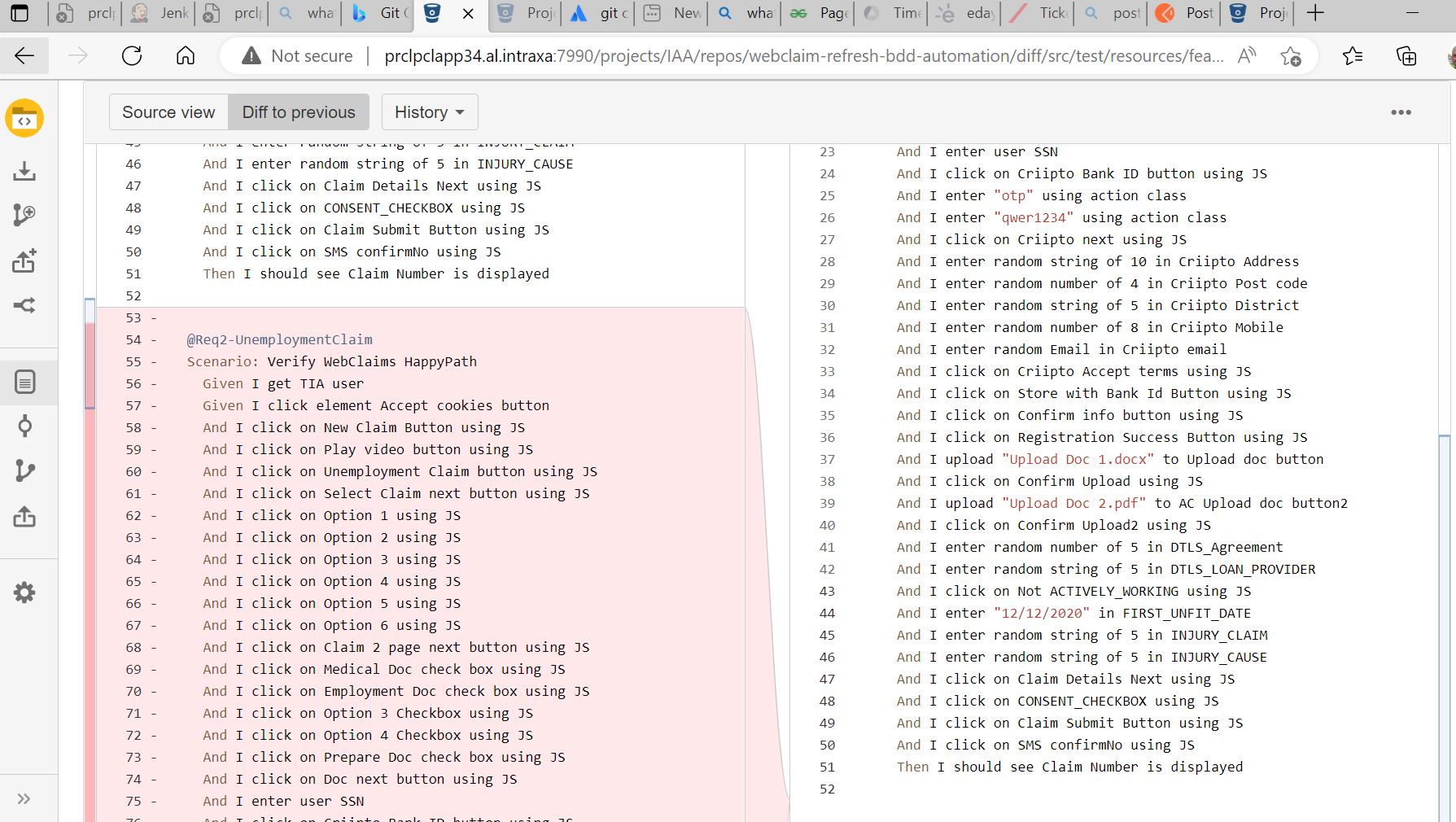
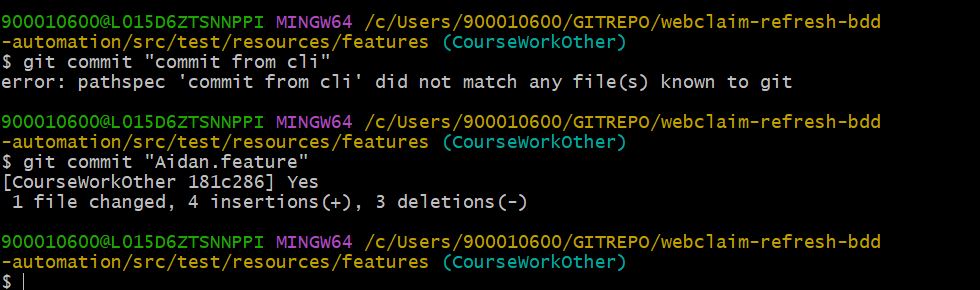
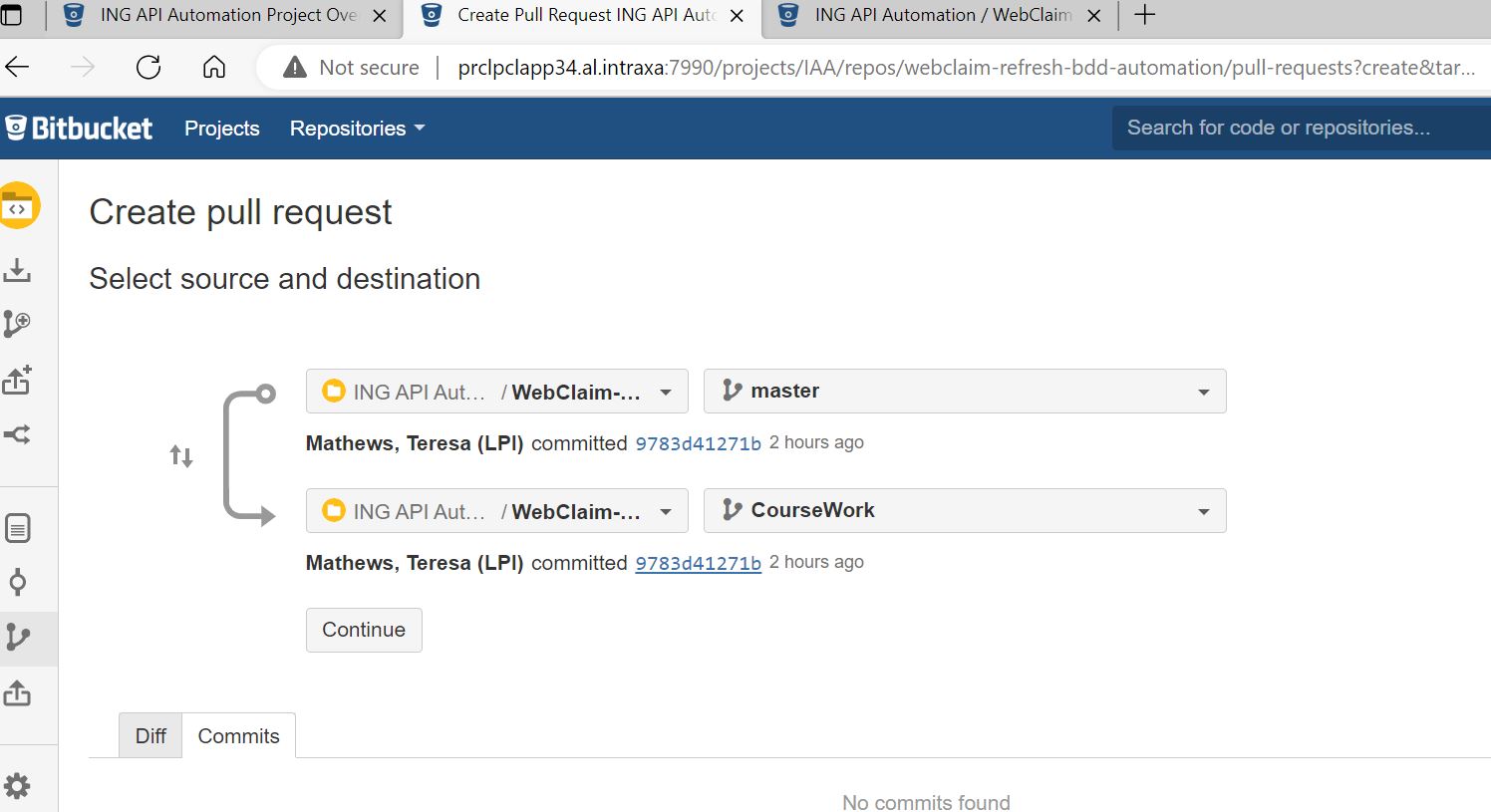
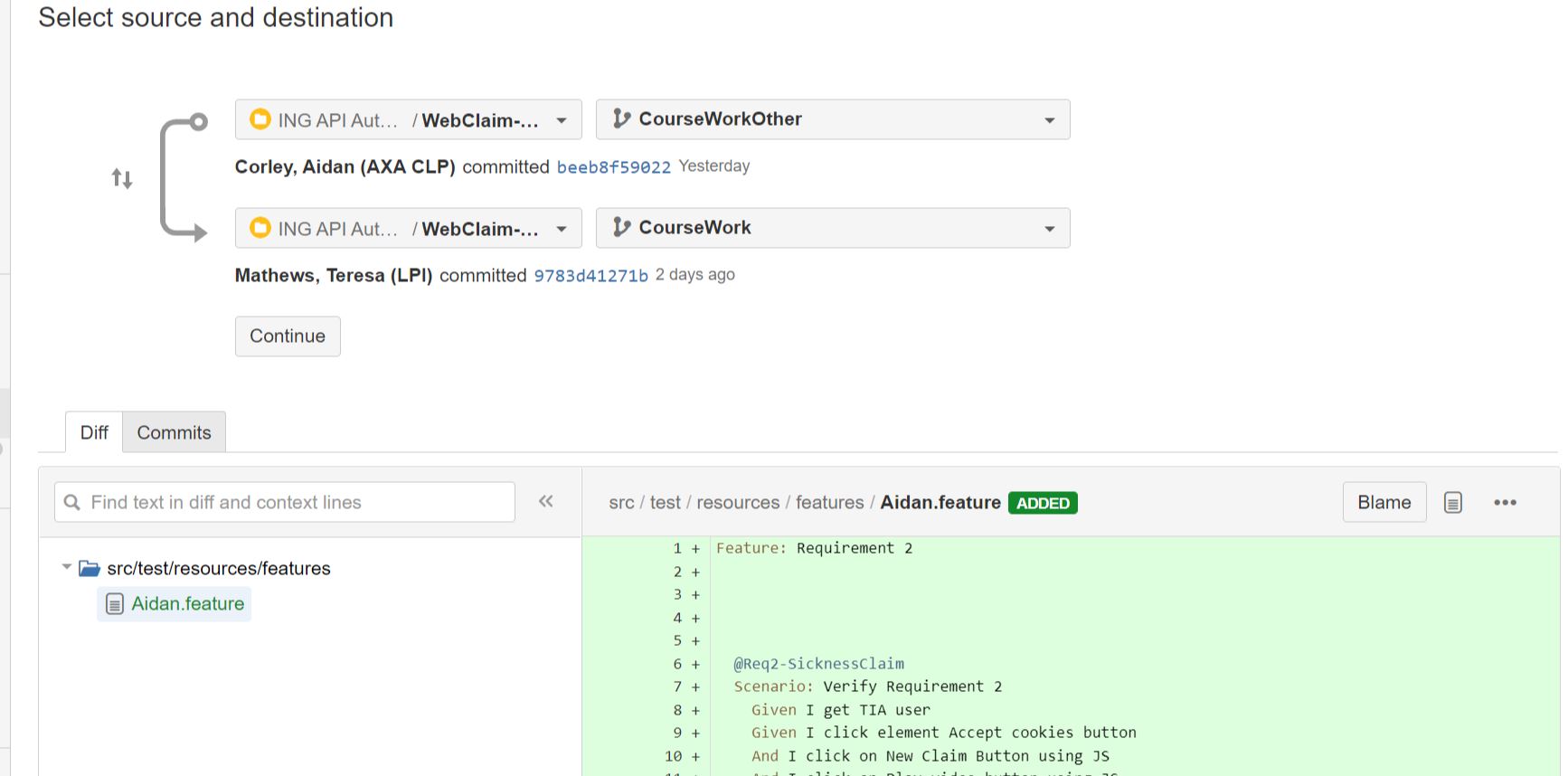


Fig 1.5.1 Side by Side

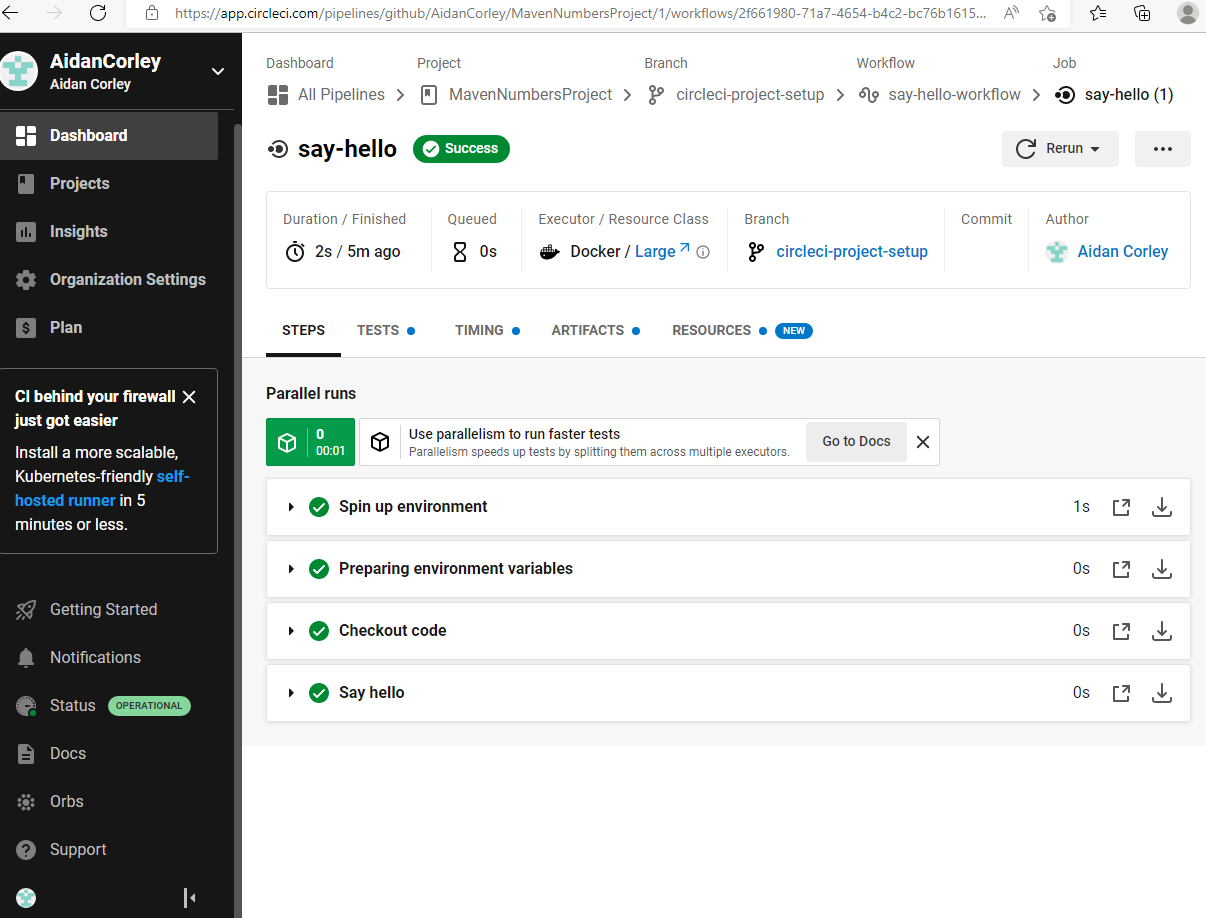


### **Fig 1.6 CLI Commit**

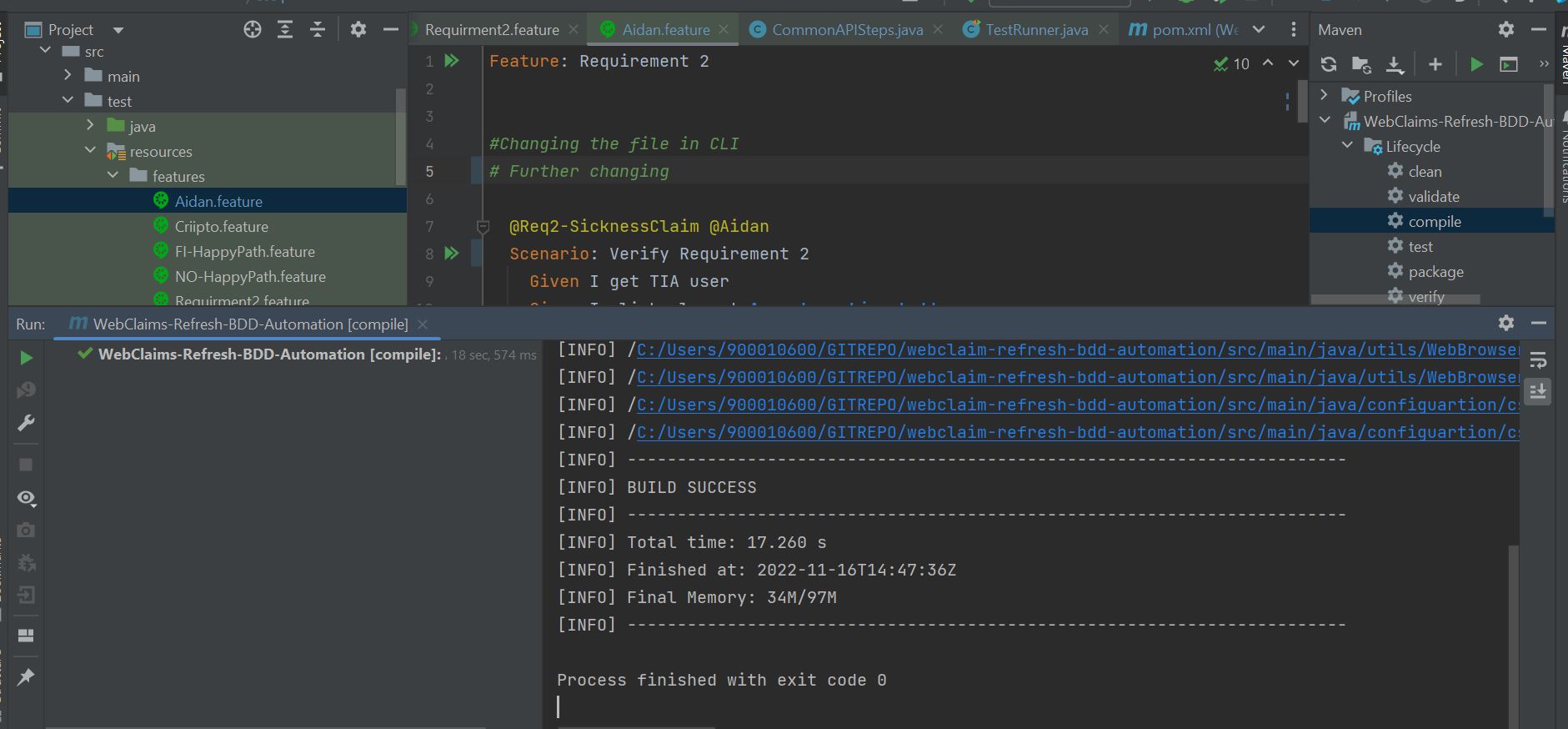
****

****

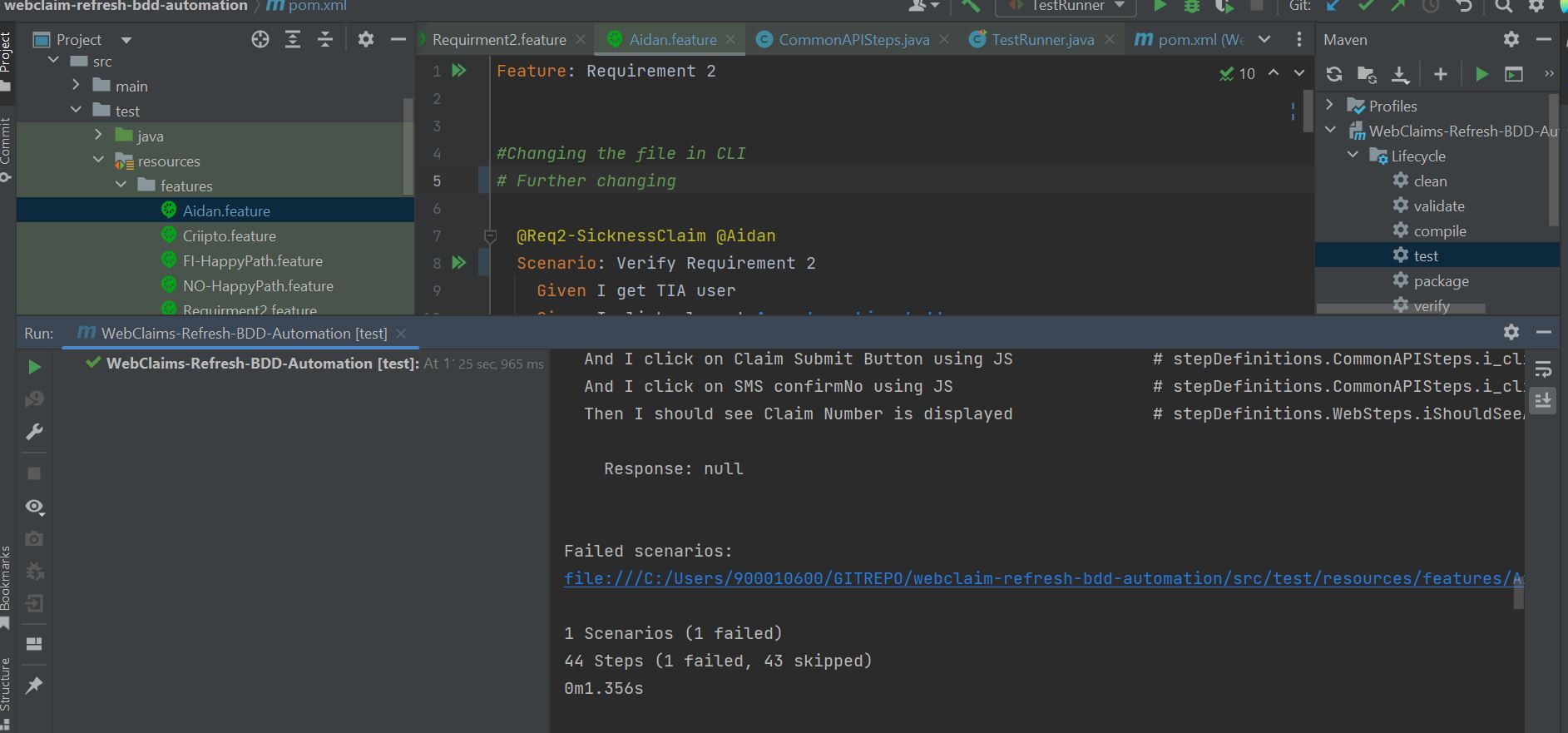
### **Fig 1.7 Pull Request**



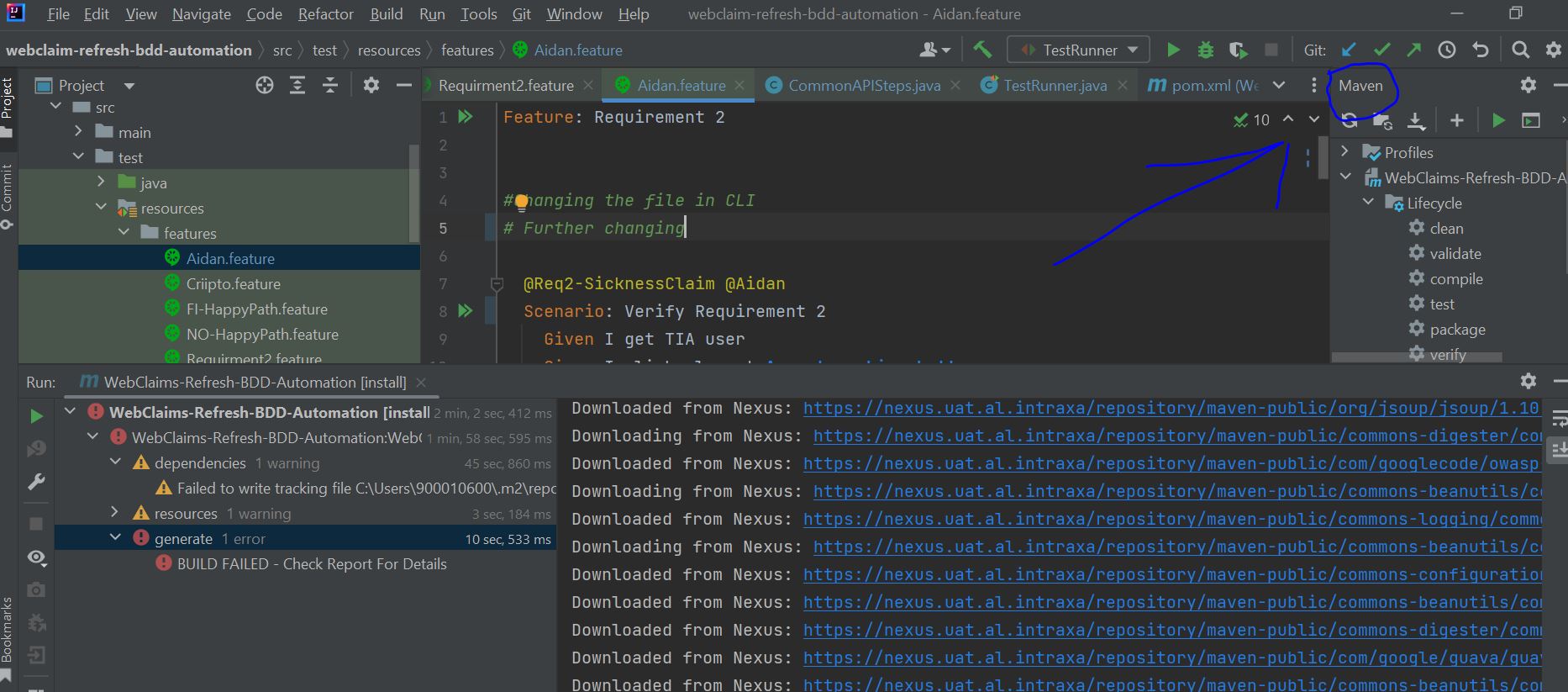
### Fig 1.8 CircleCI



### Fig 2.0 Maven Compile



### Fig 2.1 Maven Test



### Fig 2.1 Nexus within Intellij