Lab Guide

Hands-on-Lab

DataStage DevOps with MettleCI

Table of Contents

Introduction

This Hands-On-Lab (HOL) session will cover on how to use DataStage and MettleCI to

1. **Check DataStage job compliance against out-of-box pre-defined job development rules,**

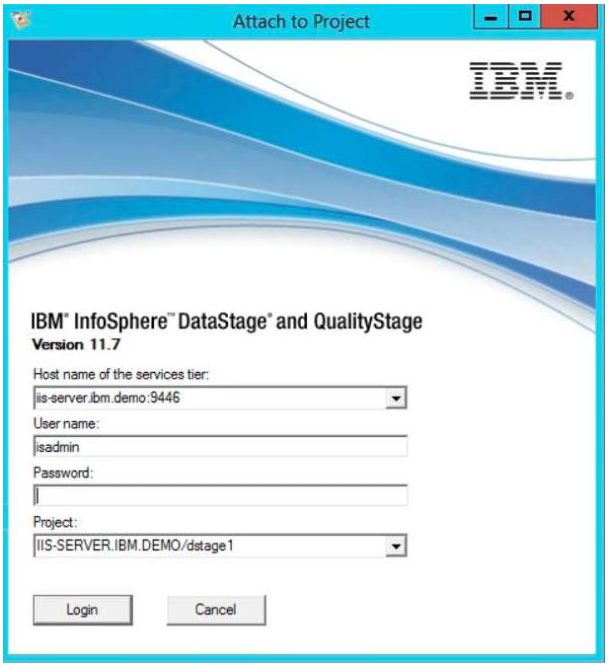
2. **Unit test DataStage jobs, and**

3. **Check in jobs into GitHub.**

Access the HOL VM environment

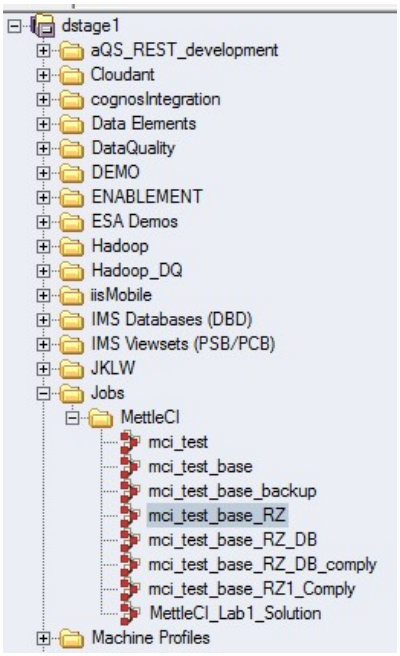
Please refer to the Access Hands on Lab document from your instructor

Click **DataStage Designer Client icon** on your desktop.
Login with user ID/password **"isadmin/inf0Xerver".**

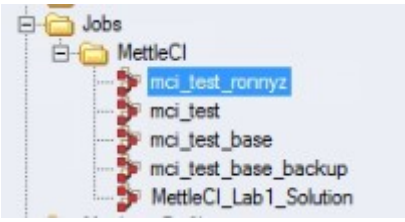


Lab-01
Check DataStage job compliance against out-of-box pre-defined job development rules

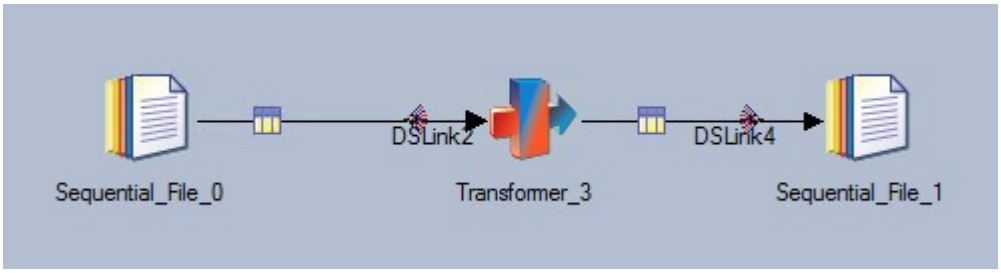
- With Designer's Repository view, open folder **"Jobs→ MettleCI".**



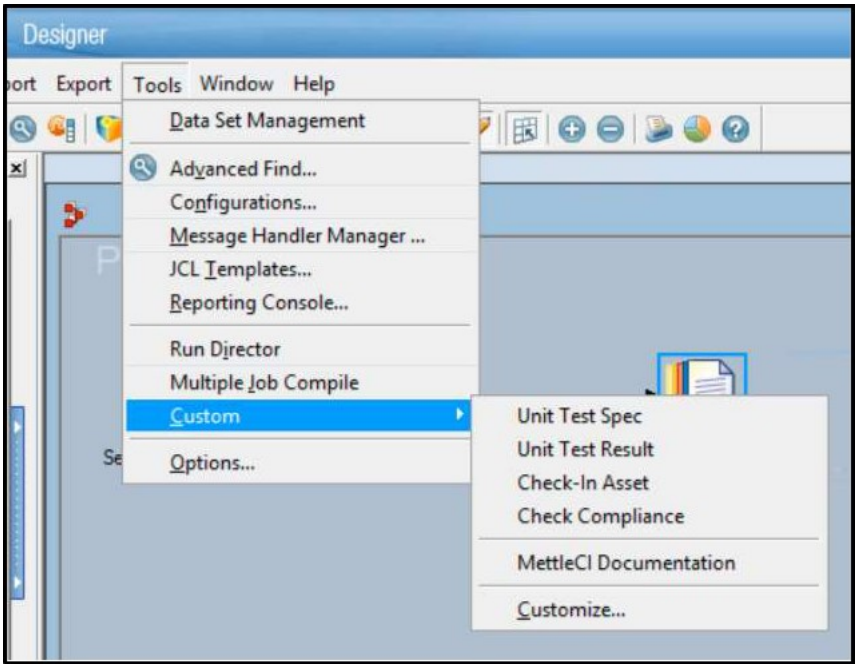
- **Make a copy** of job **"mci_test_base_RZ"** and rename to **"mci_test_<your name>** e.g. **"mci_test_ronnyz".**



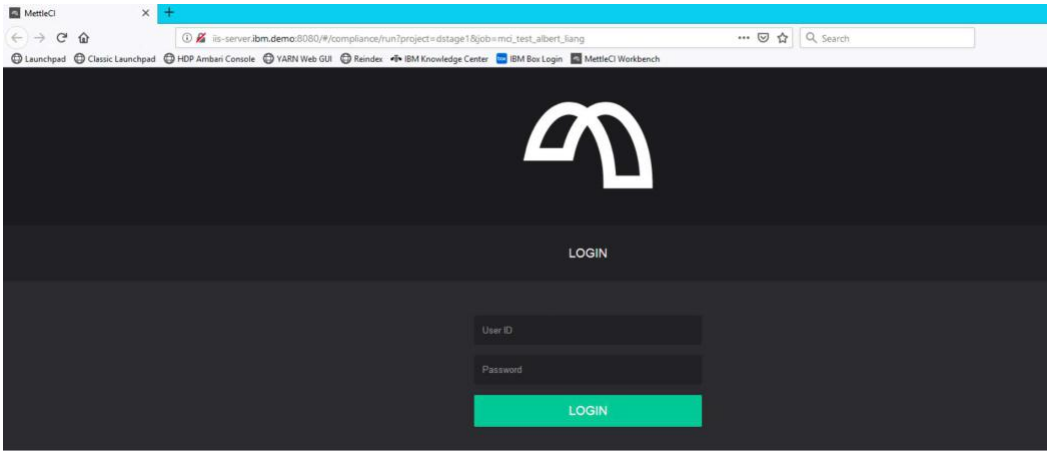
- **Double click** job **““mci_test_<your name>”** to open the job.
  The job has a very simple design, reading from a file using Sequential File Stage, use transformer and writing into a file using a Sequential File Stage.



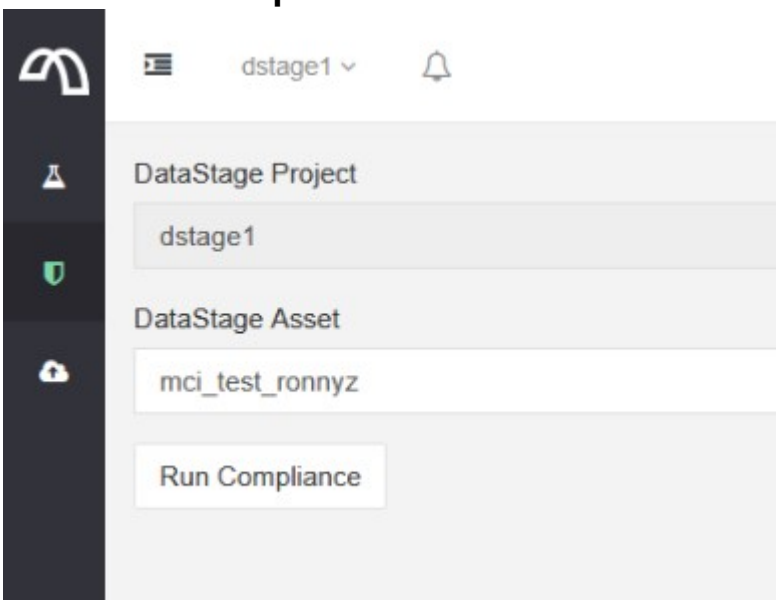
- **Click “Tools → Custom → Check Compliance”.**



- Your browser based **MettleCI workbench** home page will pop up. **Log in** with User ID/password **“isadmin/inf0Xerver”.**



- You will be presented with the page to run compliance against job **“mci_test_<your name>”.**
  Click **“Run compliance”.**



- **You will see two exceptions in the compliance report.**

1. Default Naming

PxSequentialFile 'Sequential_File_0' uses DataStage's default naming. Please provide a meaningful name meeting naming standards.

CTransformerStage 'Transformer_3' uses DataStage's default naming. Please provide a meaningful name meeting naming standards.

PxSequentialFile 'Sequential_File_1' uses DataStage's default naming. Please provide a meaningful name meeting naming standards. Link

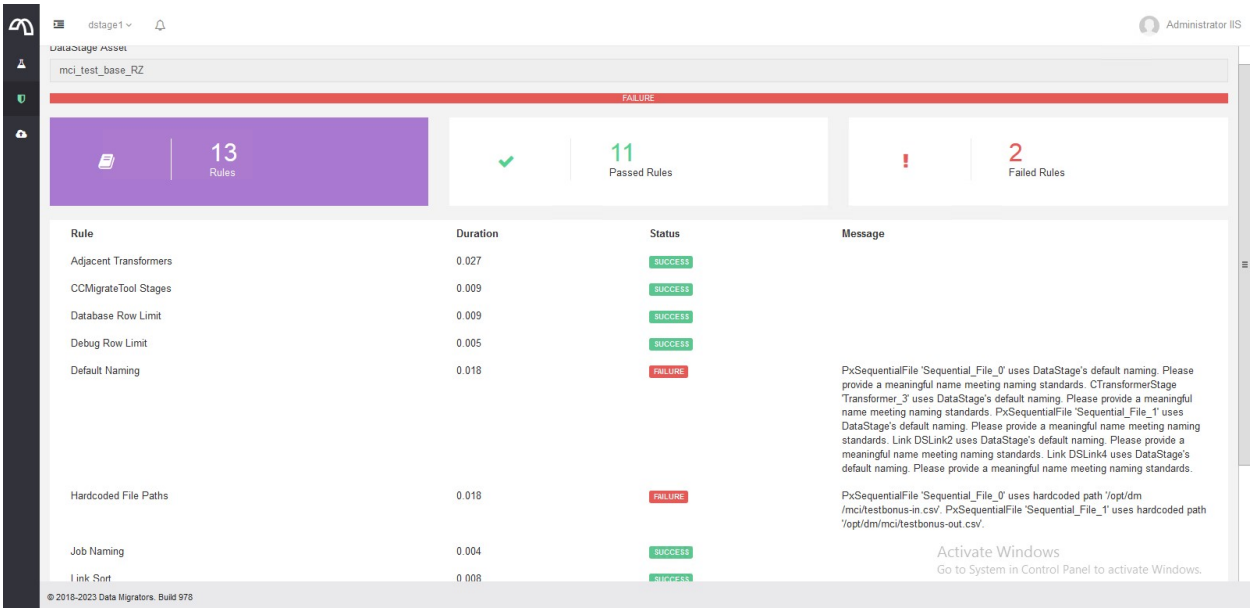
DSLink2 uses DataStage's default naming. Please provide a meaningful name meeting naming standards.

Link DSLink4 uses DataStage's default naming. Please provide a meaningful name meeting naming standards.

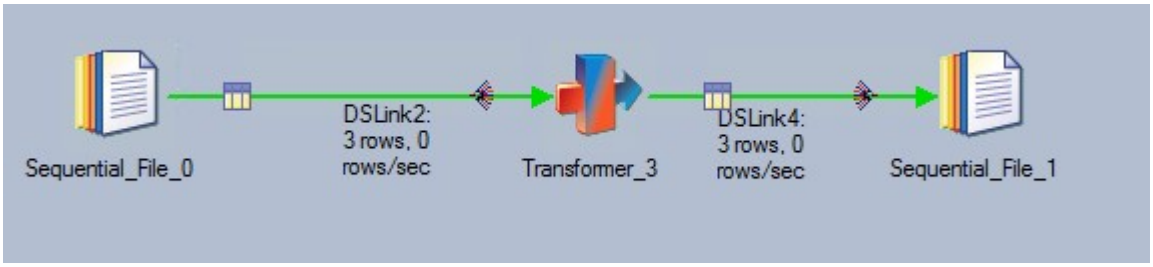
2. Hardcoded File Paths

PxSequentialFile 'Sequential_File_0' uses hardcoded path '/opt/dm/mci/testbonus-in.csv'.

PxSequentialFile 'Sequential_File_1' uses hardcoded path '/opt/dm/mci/testbonus-out.csv'..

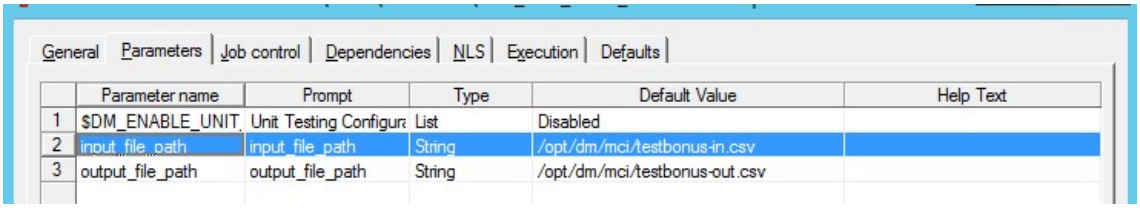


- You noticed that you can still compile and run the job successfully, but the job is not following the compliance rules.



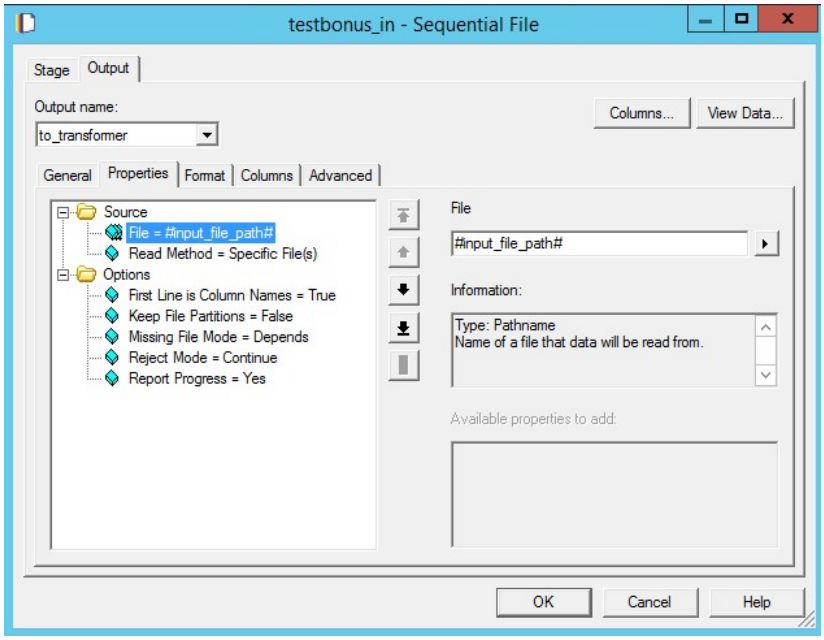
- To fix the Default Naming exception,
  replace the stage names for the input and output Sequential File stage with some more meaningful names.
  For example:
  change "**Sequential_File_0**" to "**testbonus_in**",
  and change "**Sequential_File_1**" to "**testbonus_out**".
  Replace the link named "**DSLink2**" with some meaningful name, e.g. "**to_transformer**".
  Replace the link named "**DSLink4**" with some meaningful name, e.g. "**to_output**".

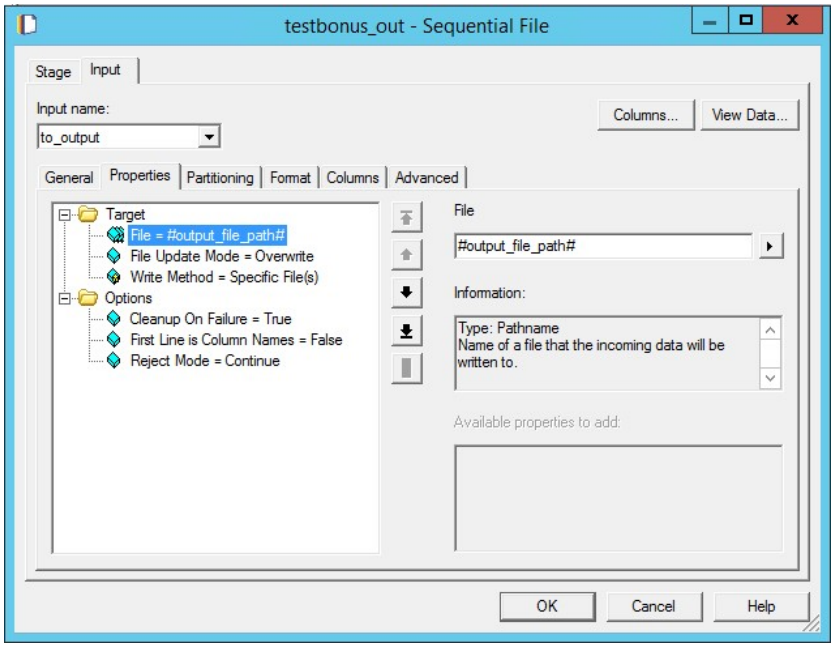
- To fix the Hardcoded File Paths exception, you need to define parameters for
  Input_file_path "/opt/dm/mci/testbonus-in.csv"
  output_file_path "/opt/dm/mci/testbonus-out.csv"



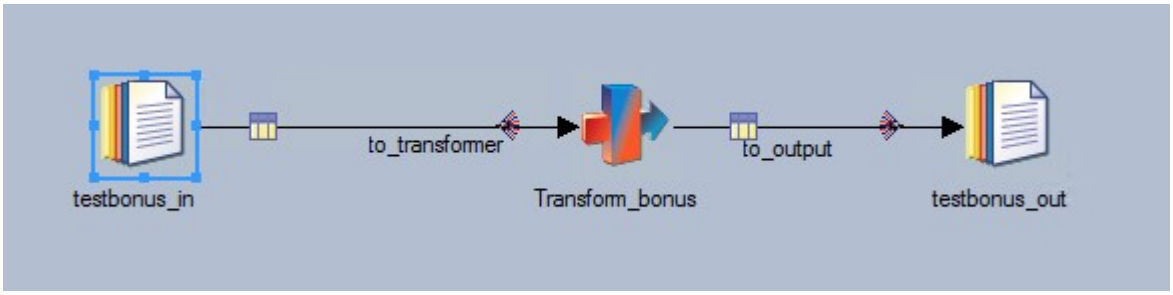
And change sequential file stage to use parameter: like below

- The stages that have the parameters defined should look like below.

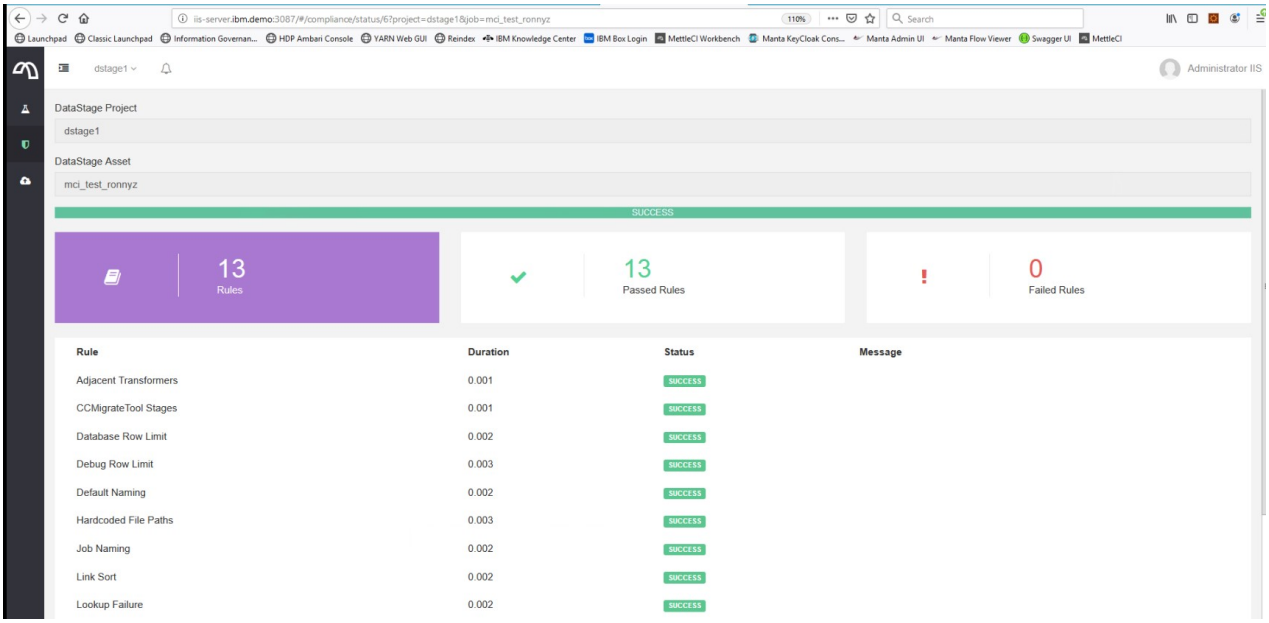




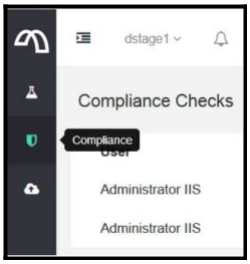
- Your modified Job **“mci_test_<your name>”** should look like below.

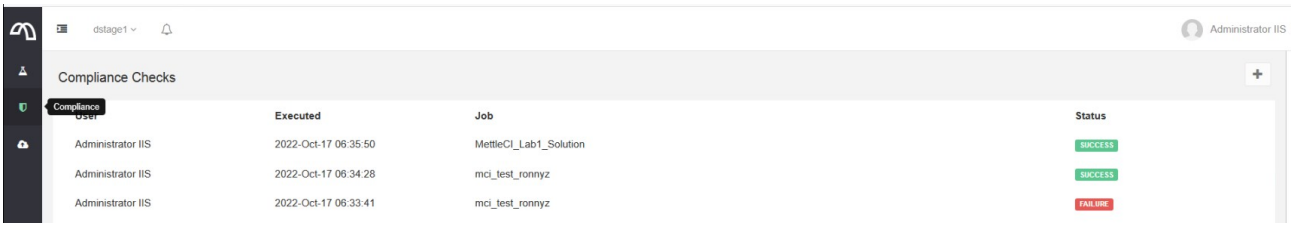


- **To check the compliance against “mci_test_<your name>”,** Open job **“mci_test_<your name>”.**
  Click **“Tools → Custom → Check compliance”.**
  If you modified your job properly, MettleCI workbench should show all 13 rules are satisfied by the compliance check and no exception reported



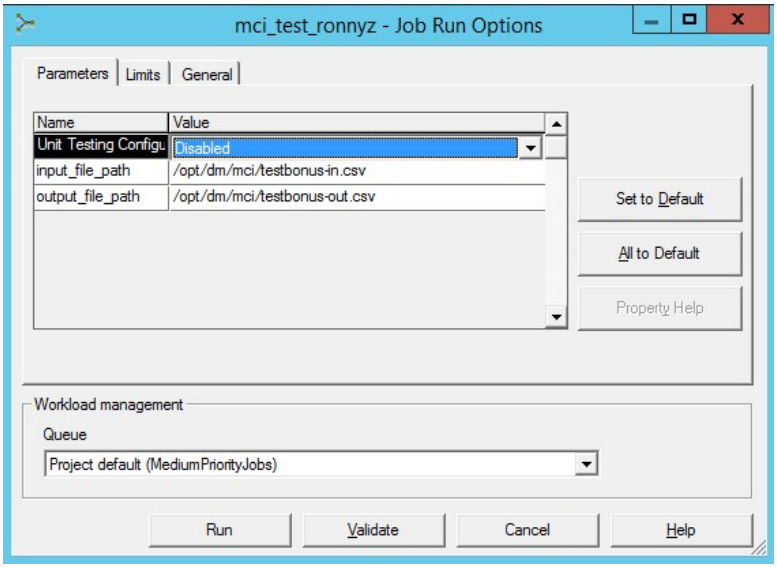
- You can review all the compliance checks you have done by clicking on the **“Compliance”** icon on the left pane.

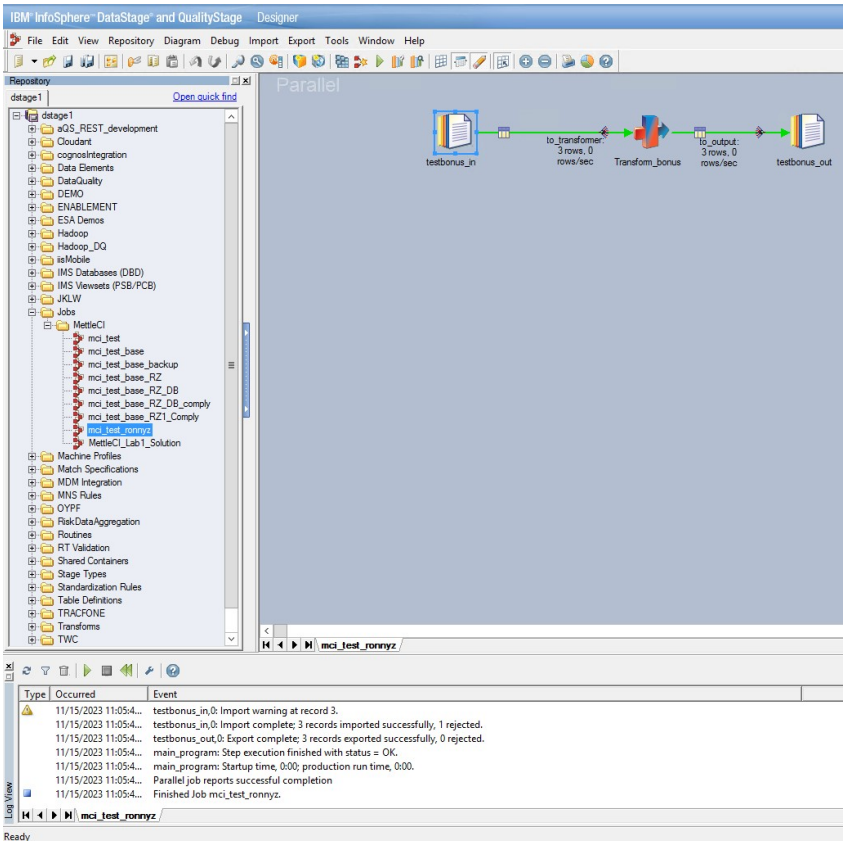




- After the compliance check passed successfully, you can compile and run the job again to verify the job can run successfully.

  When you **run job "mci_test_<your name>"**, use the default values for the parameters (See below). The job should run successfully.

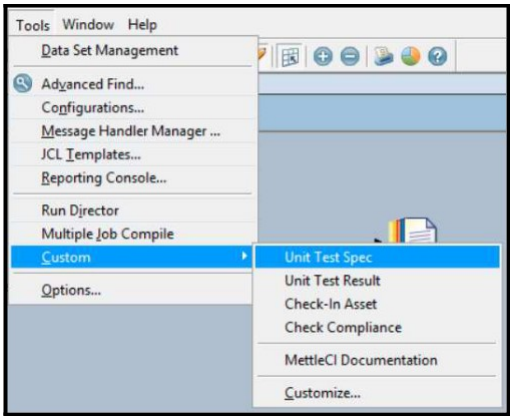




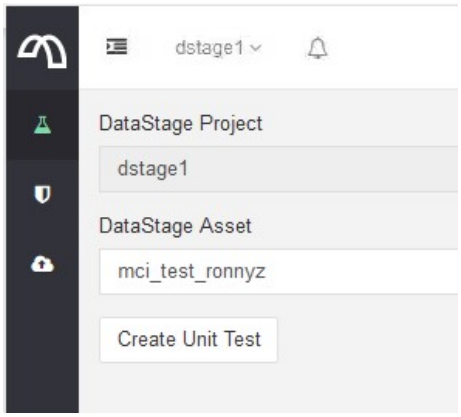
This concludes Lab 1

Lab-02
Use MettleCI to perform Unit Test DataStage jobs

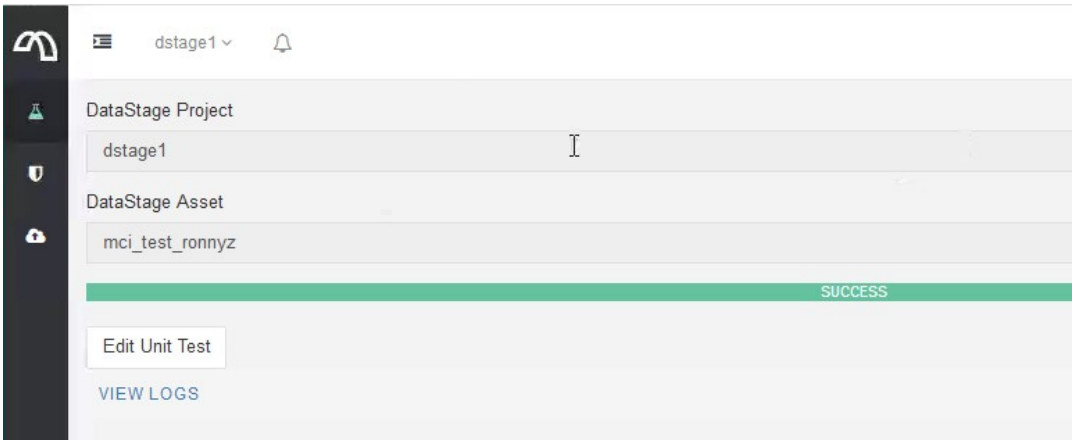
- Open job **“mci_test_<your_name>”**
- Click **“Tools → Custom → Unit Test Spec”** to generate the Unit Test Specification.



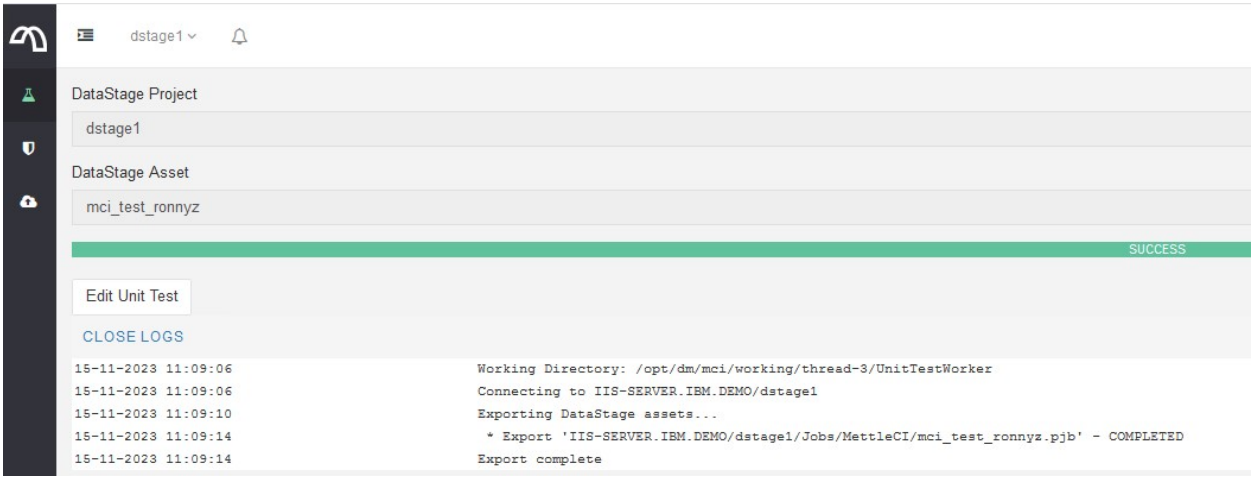
- If the Unit Test is not created, you should see below **“Create Unit Test”** screen on MettleCI Workbench popup window. Click **“Create Unit Test”** to create the unit test specification**.**



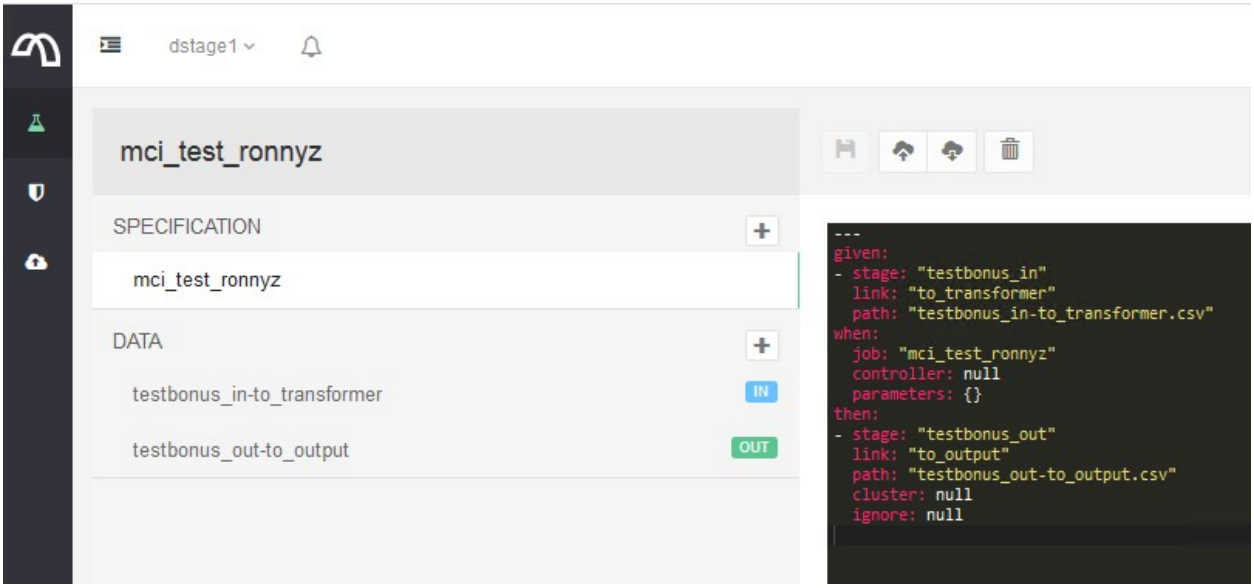
- You should see Unit Test Specification was created successfully.



Click **"VIEW LOGS"** to view the **"Create Unit Test" results.** Click **"CLOSE LOGS"** to close the log.



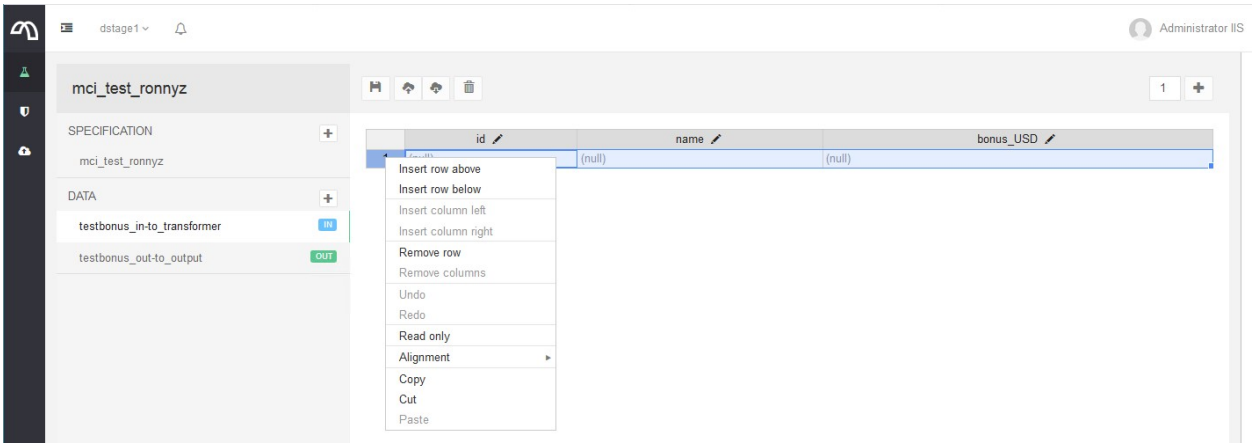
Click **“Edit Unit Test”** to view the unit test SPECIFICATION (on right pane below with black background) and unit test DATA.



You can click on a specific data file
(e.g. **testbonus_in-to_transformer** and **testbonus_out-to_output)**
To review the contents of the file.
To revise the contents, you can click a specific cell and add/revise the cell contents**.**

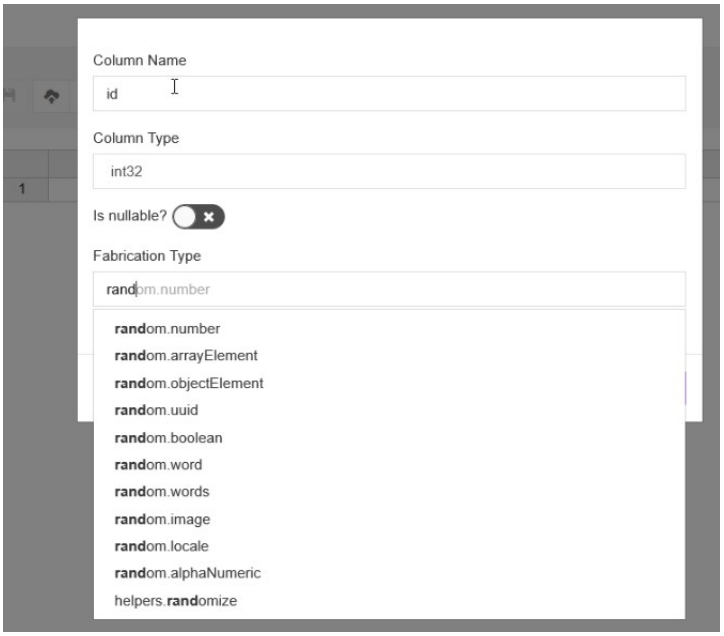
You can manipulate the row(s) of the data files by right clicking on the row number (e.g. row 1 below)

A popup window will display the functions you can use,
e.g. insert/remove rows, copy/cut/paste rows.

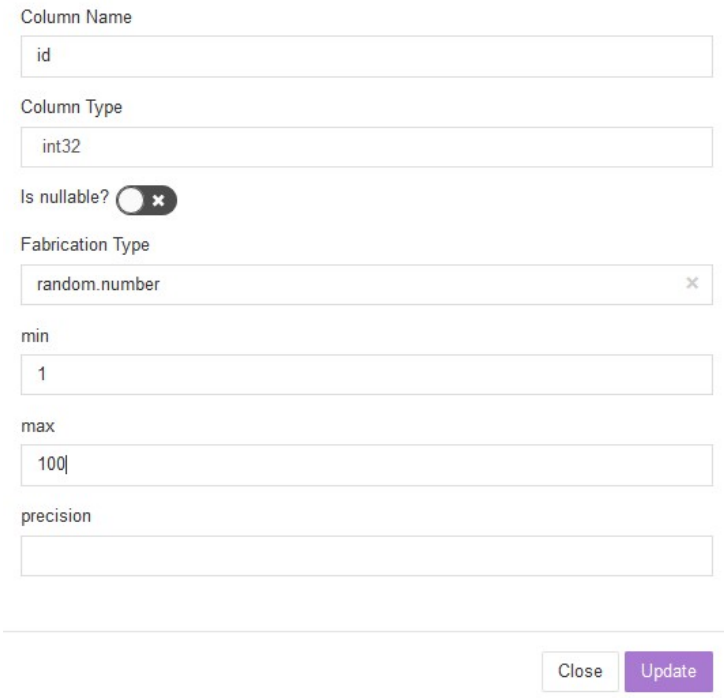


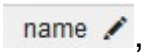
Now we will provide input for the input test file
click **testbonus_in-to_transformer** on the left pane under DATA section

MettleCI provide 200+ fabrication function for unit test data generation, you can click the pencil icon in the column header

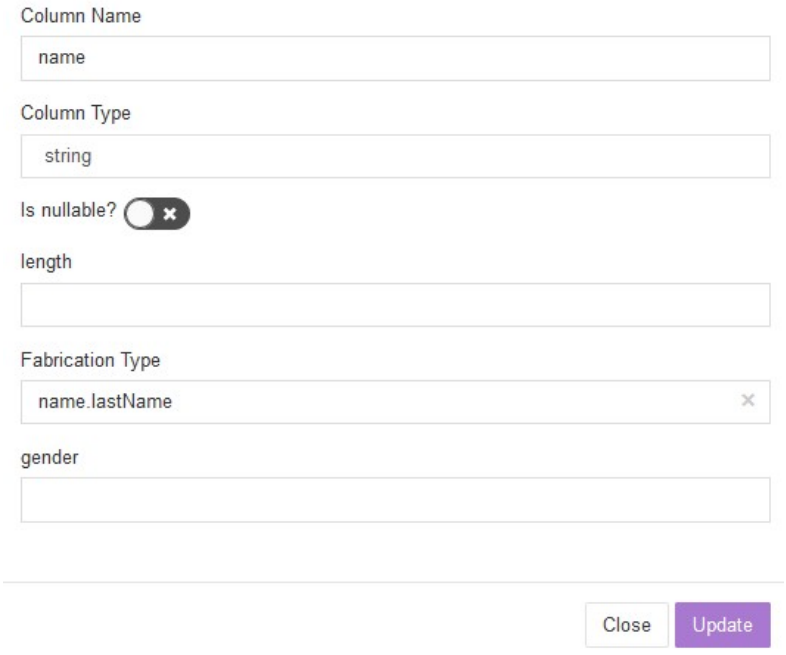


Click the pencil icon beside **id** column  ,
then choose *random.number* Fabrication Type
with min 1 and max 100
then click **update**



Now click pencil icon beside **name** column 

, choose *name.lastName* fabrication type, then click **update**



Then go to top right, fill in 2 and **click** the plus sign,

notice that the **id** and **name** column, automatically populated based

on the fabrication type we choosed above

	id	name	bonus_USD
1	46	Brakus	(null)
2	28	Kerluke	(null)
3	22	Bartell	(null)

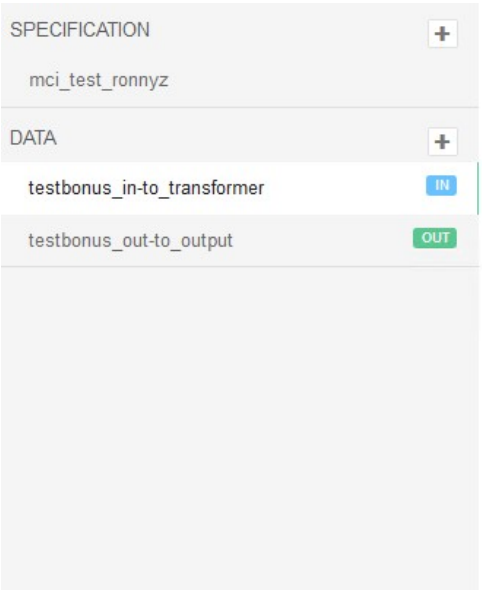
now type in the **bonus_USD** column: 100, 200 and 300 like below

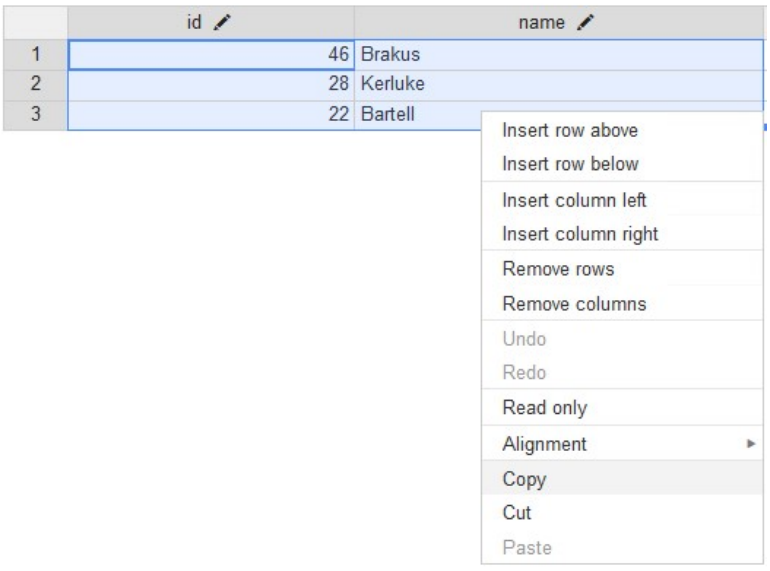
	id	name	bonus_USD
1	46	Brakus	100
2	28	Kerluke	200
3	22	Bartell	300

Then click **“Save”** icon

You can also **“Upload”** your csv file**, “Download”** the data you specified or **“Delete”** a specific data file, but we will not use it in this lab

highlight the content of **id** and **name** column, right click → **copy**



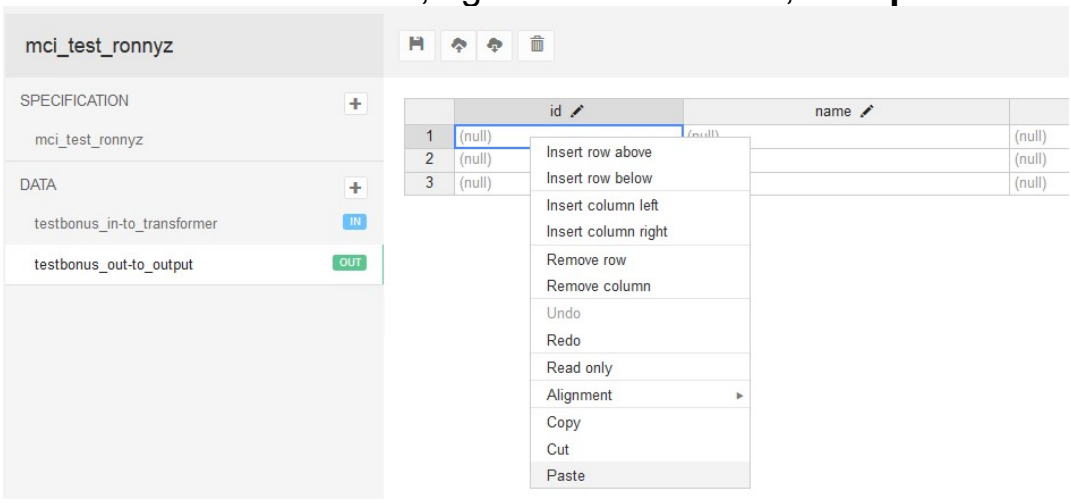


Now we will specify the expected test output file
click **testbonus_out-to_output** on the left pane as below.

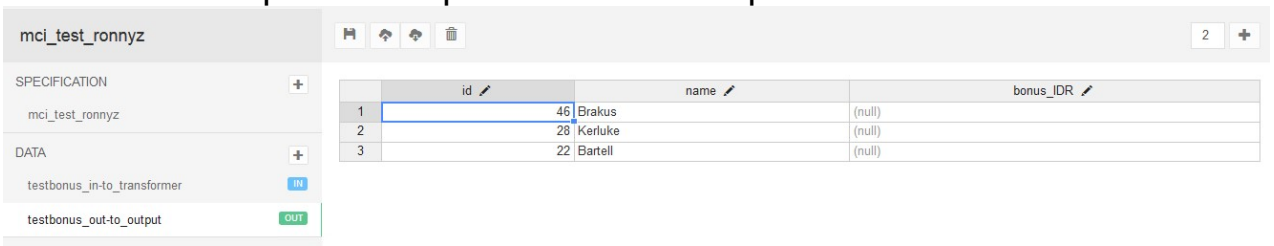
Type in 2 on the top right, then click plus sign 

,

2 new rows will be added, right click on cell id 1, then **paste**



The value from previous input test file will be pasted like below

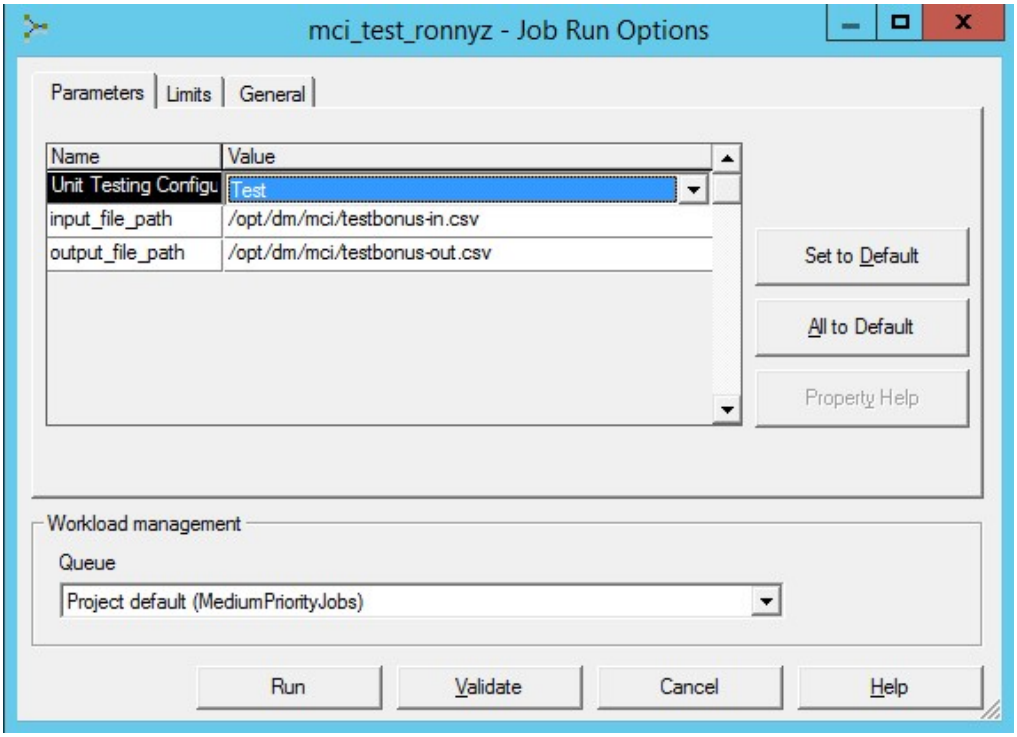


Now fill in the **bonus_IDR** column with:
1500000 ; 3000000 ; 4500000 , the expected test output file become

	id	name	bonus_IDR
1	46	Brakus	1500000
2	28	Kerluke	3000000
3	22	Bartell	4500000

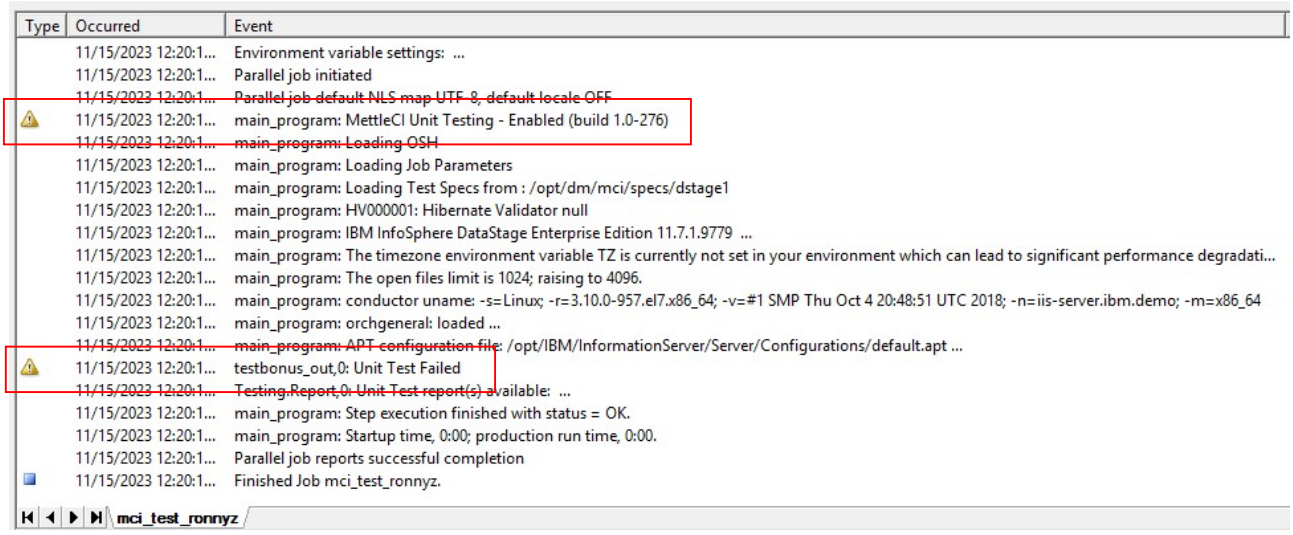
after you finished input 3 rows above, then click save icon 

Go back to DS Designer, click **“Compile”,** then **“Run”** to run your job with parameter **“Unit Test Configuration”** set to **“Test”.**

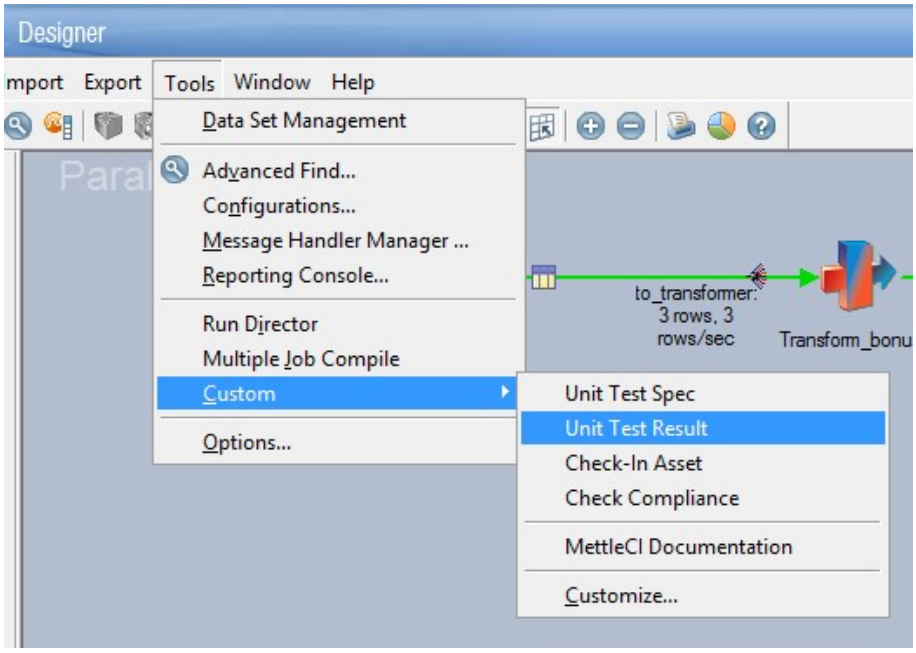


You should see the log displayed as below with below key messages.

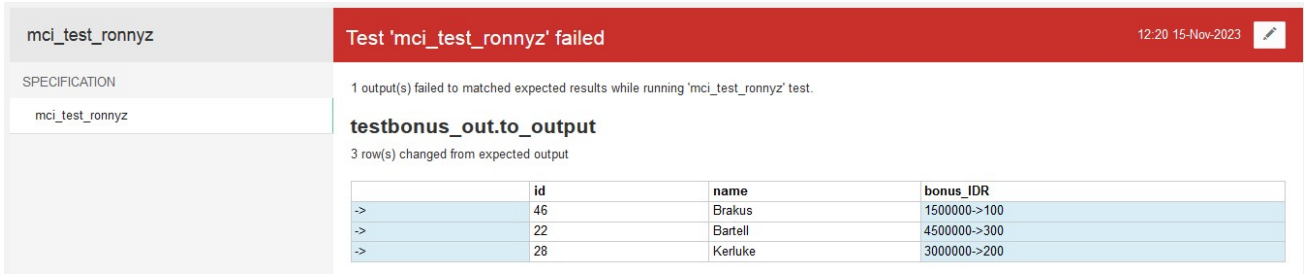
main_program: MettleCI Unit Testing - Enabled (build 1.0-276)
testbonus_out,0: Unit Test Failed
Testing.Report,0: Unit Test report(s) available:



On **DS Designer,** click **Tools → Custom → Unit Test Result"**

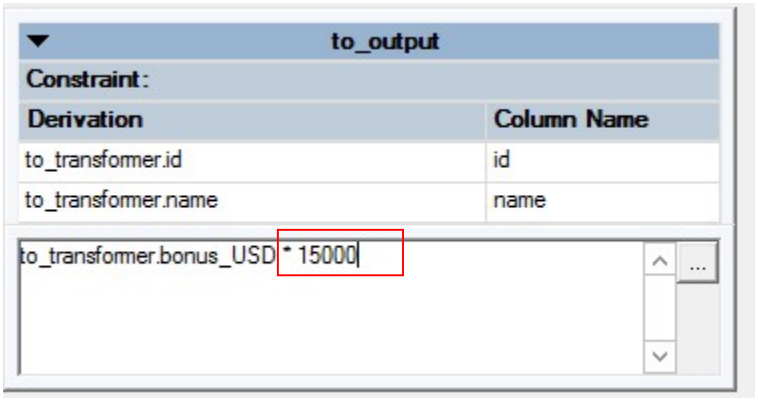


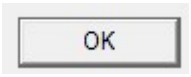
You will see the message Test **'mci_test_<your name>'** failed with **red** background. The reason for the unit test failure is indicated as
The job didn't produce the correct expected output value as specified in the **testbonus_out.to_output**, it expect **'1500000'** while it received **'100'** value
it expect **'3000000'** while it received **'200'** value,
it expect **'4500000'** while it received **'300'** value
in real world case this can be wrong calculation in the job logic, can also be a wrong lookup logic etc.



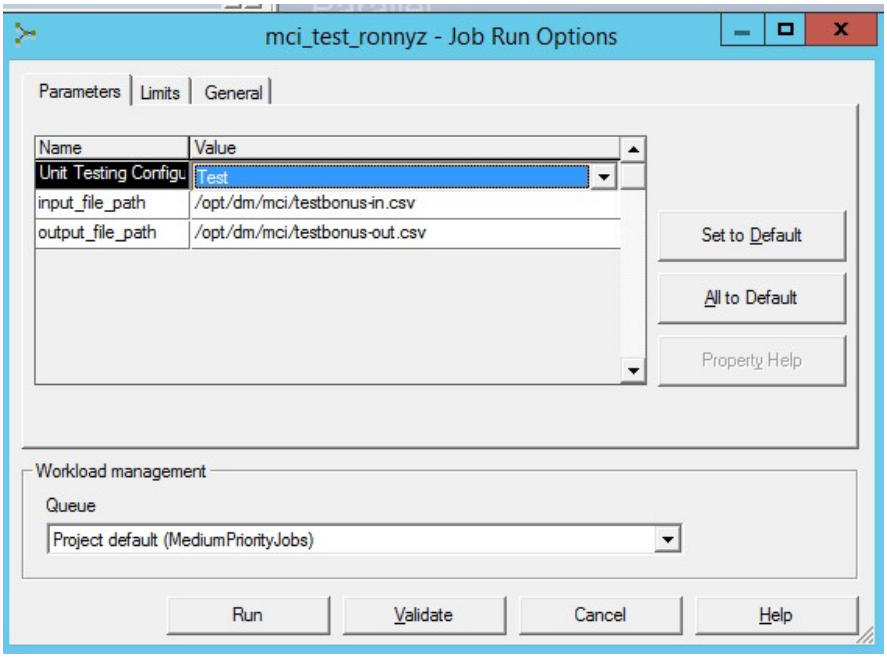
Now go back to DS Designer, open your job, and double click the transformer stage to modify the transformation logic

edit the bonus_IDR logic, add * 15000
(this represents conversion rate from USD to IDR)



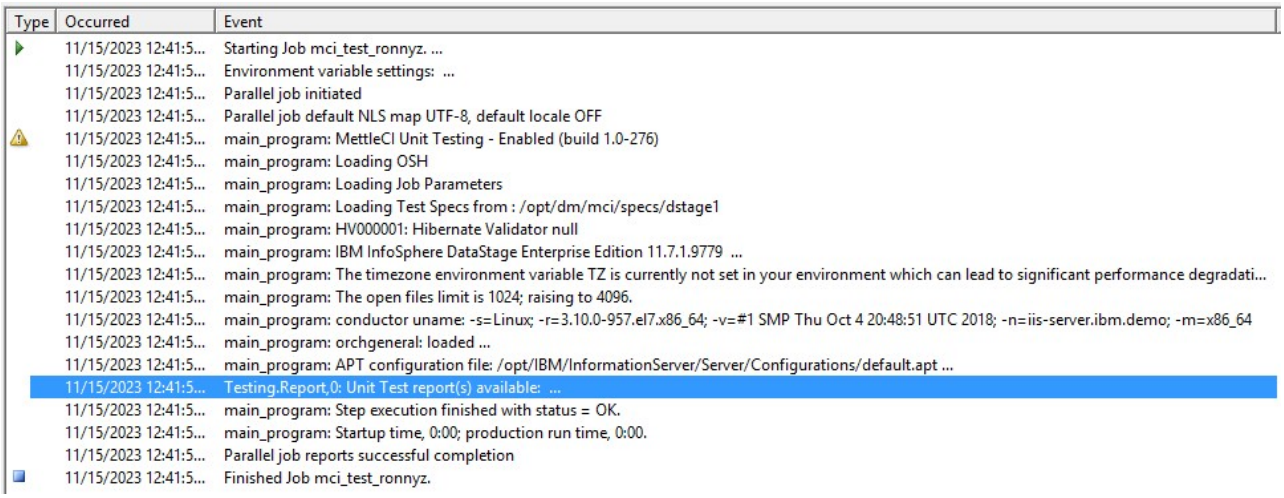
Then click ok 

click **“Compile”**, then **“Run”** to run the job with parameter **“Unit Test Configuration”** set to **“Test”.**



You should see the log displayed as below with below key messages.

main_program: MettleCI Unit Testing - Enabled (build 1.0-276)
Testing.Report,0: Unit Test report(s) available:

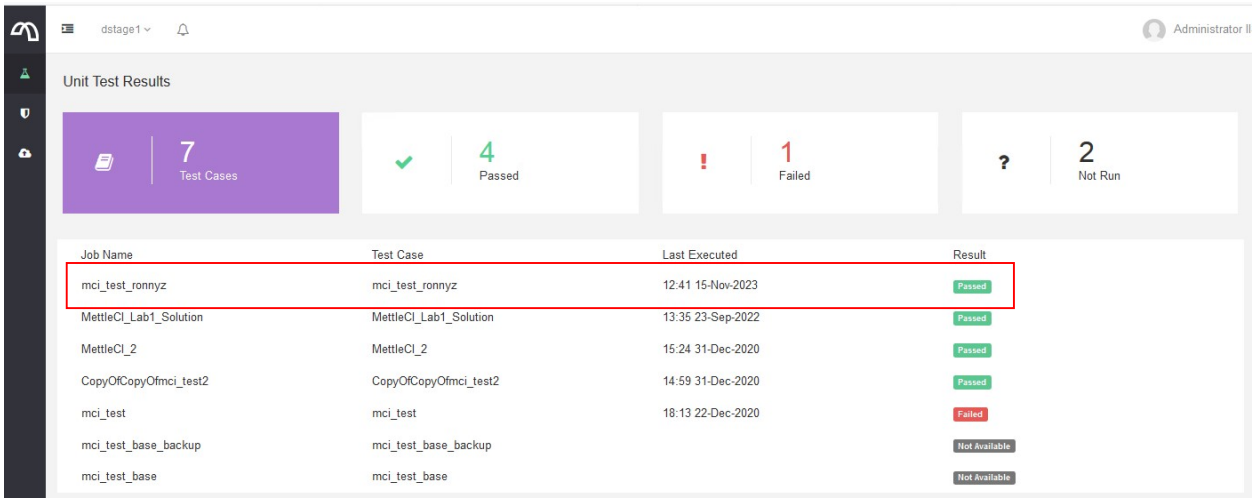


Type	Occurred	Event
	11/15/2023 12:41:5...	Starting Job mci_test_ronnyz. ...
	11/15/2023 12:41:5...	Environment variable settings: ...
	11/15/2023 12:41:5...	Parallel job initiated
	11/15/2023 12:41:5...	Parallel job default NLS map UTF-8, default locale OFF
	11/15/2023 12:41:5...	main_program: MettleCI Unit Testing - Enabled (build 1.0-276)
	11/15/2023 12:41:5...	main_program: Loading OSH
	11/15/2023 12:41:5...	main_program: Loading Job Parameters
	11/15/2023 12:41:5...	main_program: Loading Test Specs from : /opt/dm/mci/specs/dstage1
	11/15/2023 12:41:5...	main_program: HV000001: Hibernate Validator null
	11/15/2023 12:41:5...	main_program: IBM InfoSphere DataStage Enterprise Edition 11.7.1.9779 ...
	11/15/2023 12:41:5...	main_program: The timezone environment variable TZ is currently not set in your environment which can lead to significant performance degradati...
	11/15/2023 12:41:5...	main_program: The open files limit is 1024; raising to 4096.
	11/15/2023 12:41:5...	main_program: conductor uname: -s=Linux; -r=3.10.0-957.el7.x86_64; -v=#1 SMP Thu Oct 4 20:48:51 UTC 2018; -n=iis-server.ibm.demo; -m=x86_64
	11/15/2023 12:41:5...	main_program: orchgeneral: loaded ...
	11/15/2023 12:41:5...	main_program: APT configuration file: /opt/IBM/InformationServer/Server/Configurations/default.apt ...
	11/15/2023 12:41:5...	Testing.Report,0: Unit Test report(s) available: ...
	11/15/2023 12:41:5...	main_program: Step execution finished with status = OK.
	11/15/2023 12:41:5...	main_program: Startup time, 0:00; production run time, 0:00.
	11/15/2023 12:41:5...	Parallel job reports successful completion
	11/15/2023 12:41:5...	Finished Job mci_test_ronnyz.

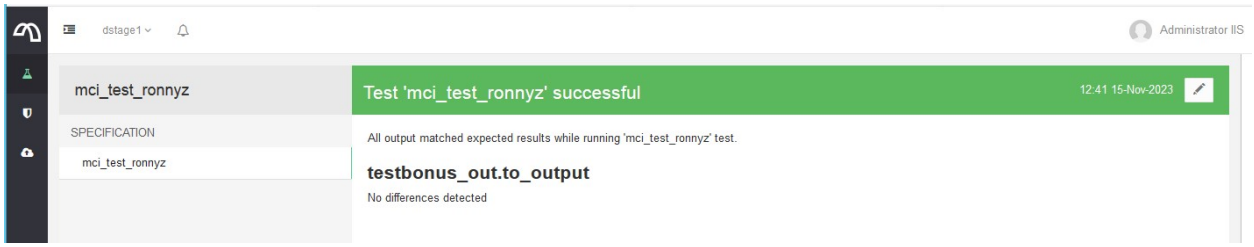
Failed Message has disappeared.

On **MettleCI workbench,** click **“Unit Testing → Test Reports”** You will see a list of test reports for various jobs.
The status for job **“mci_test_<your name>”** is **“Passed”.**

Click **“Passed”** icon for job **“mci_test_<your name>”.**



You will see the message **Test 'mci_test_<your name>' successful** with **green** background,
showing No difference detected

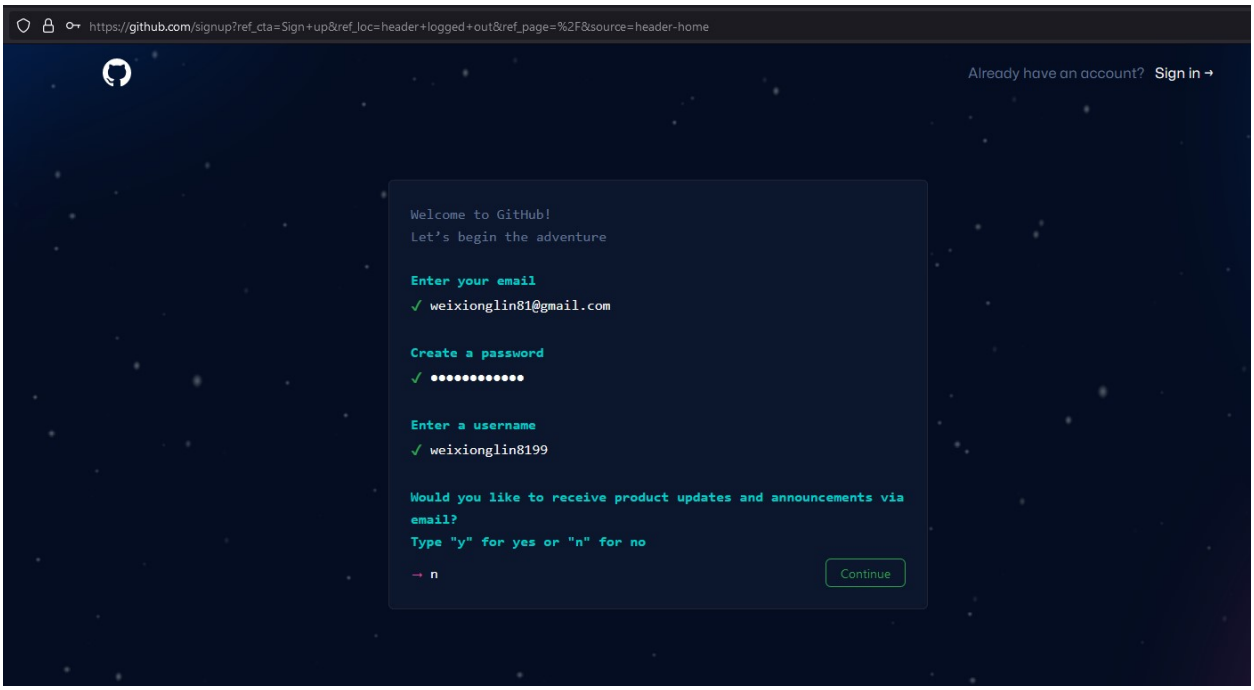


This concludes Lab 2

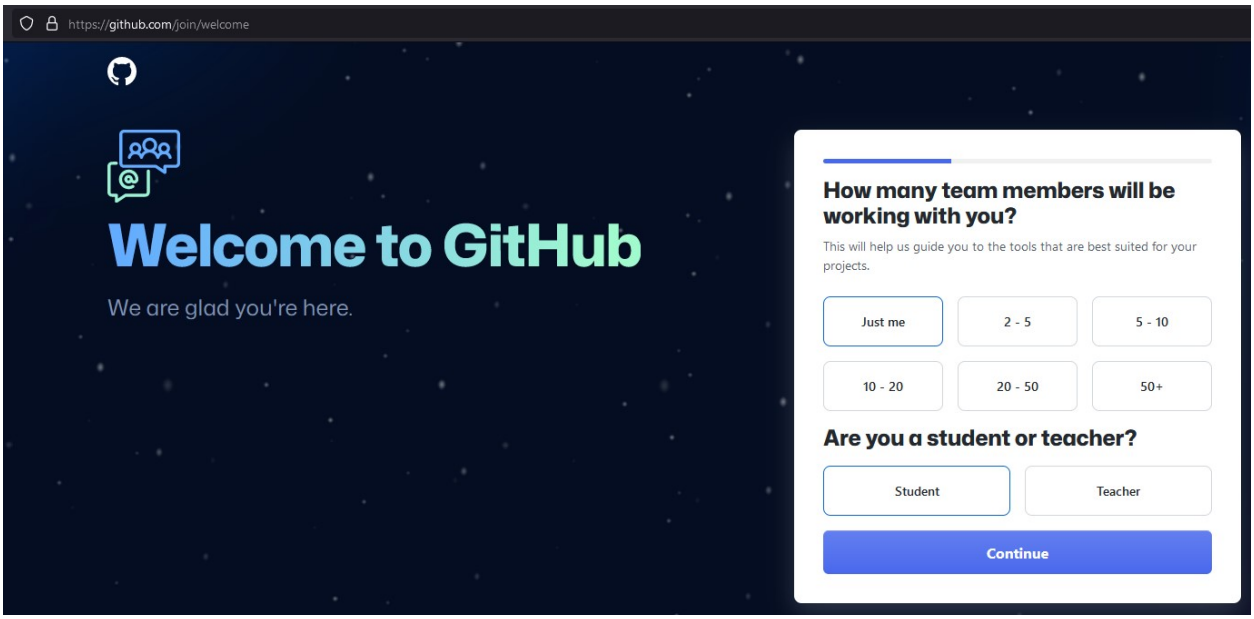
Lab-03
Check in DataStage job into GitHub

This lab will require you to have a Github account and Github repository

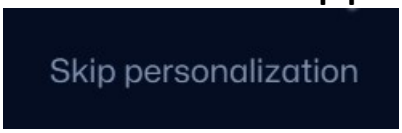
If you don’t have one, **Sign up** a Github account from github.com
Use your email address to sign up and follow the steps from your computer



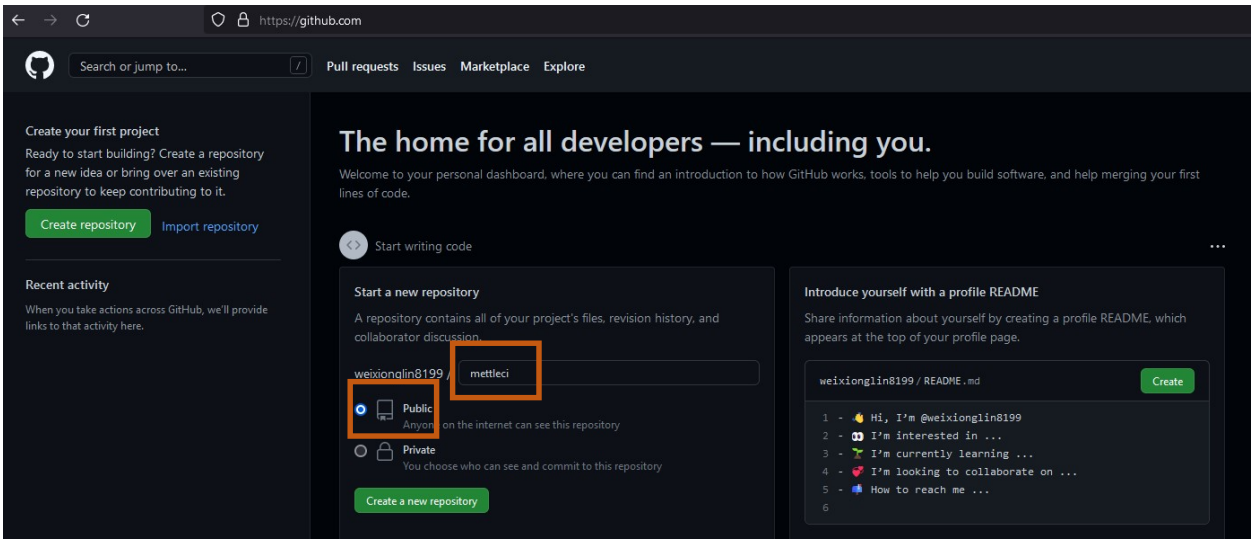
Then follow to start the puzzle and key in the code sent to your email



You can also click **Skip personalization** at the bottom

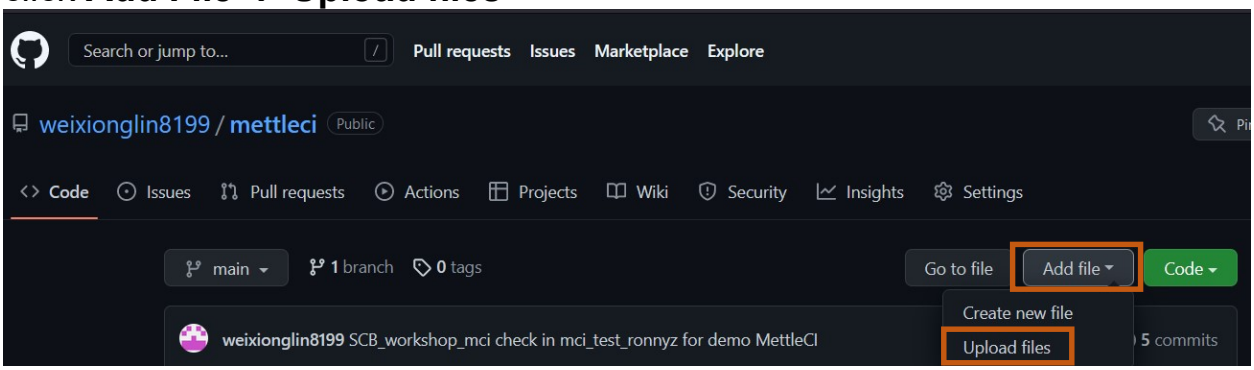


After you created your github account, you need to **Create a public repository**, give it a name **mettleci**

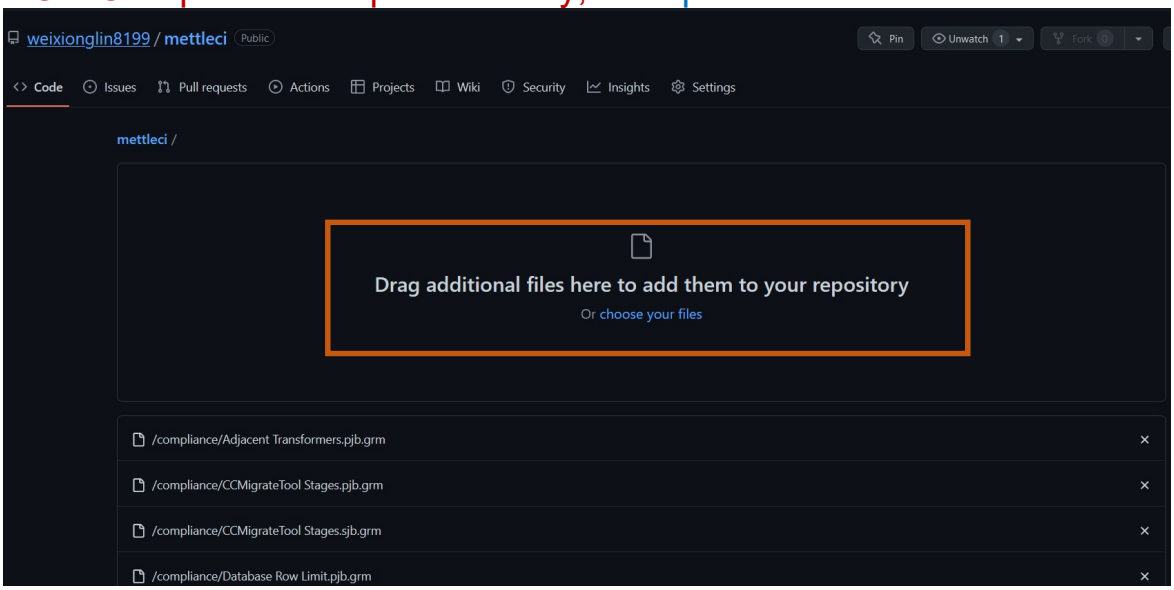


- Now from your Techzone Image computer, go to: https://ibm.ent.box.com/folder/234744591722
  download the zip repo files **proj-techzone-datastage-mettleci.zip**

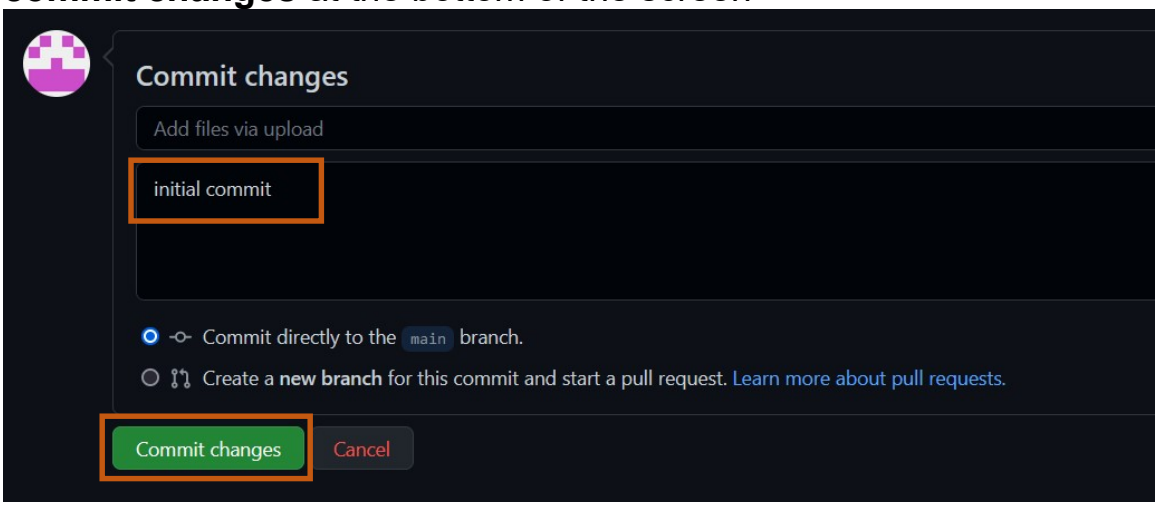
- Then on your mettleci repository you created above make a commit and push to origin, this can be done by uploading the content of zip folder like below (make sure you extract the zip file first and upload the content, DO NOT upload the zip file directly
  click **Add File → Upload files**



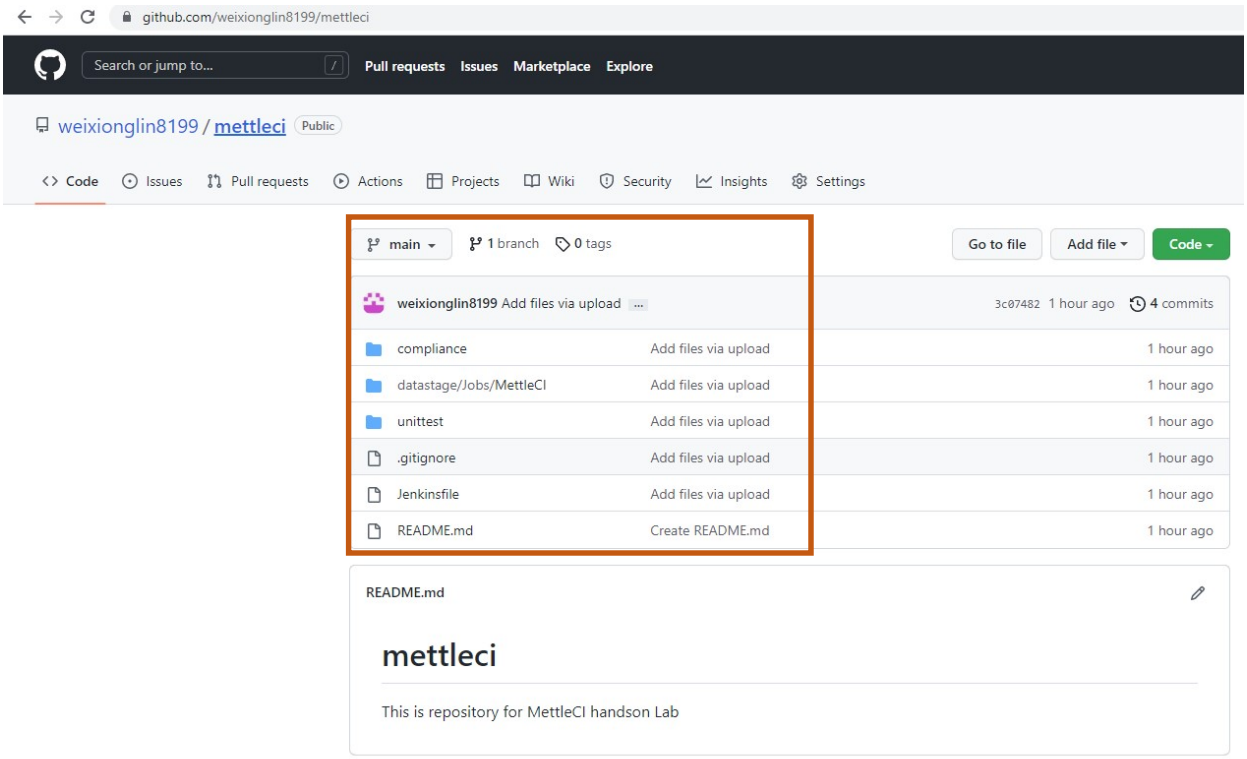
Drag you're the content of zip folder to upload it
DO NOT upload the zip file directly, but upload the extracted content



After the file is uploaded, then add **comment 'initial commit'** click **commit changes** at the bottom of the screen



After the Upload your mettleci repository should be like below structure



If you already have github account,

- Login to GitHub page at **https://github.com/**
- And create your own **MettleCI repository** example:
  (Please note that the link will be different, depends on your git username)
  https://github.com/weixionglin8199/mettleci
- Then perform the upload file step in page 30 above to push to origin

Set up the GitHub SSH key between your GitHub account and MettleCI:

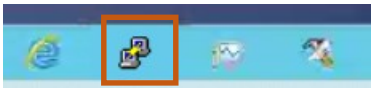
This step required because the workbench.key in the iis-server environment is created using rsa and Github has dropped support for ***rsa*** keytype.

The next version of MettleCI workbench will be enhanced to create ***ecdsa*** keytype for the workbench.key which will be created during the setup wizard.

- We will now generate a key pair on iis-server.

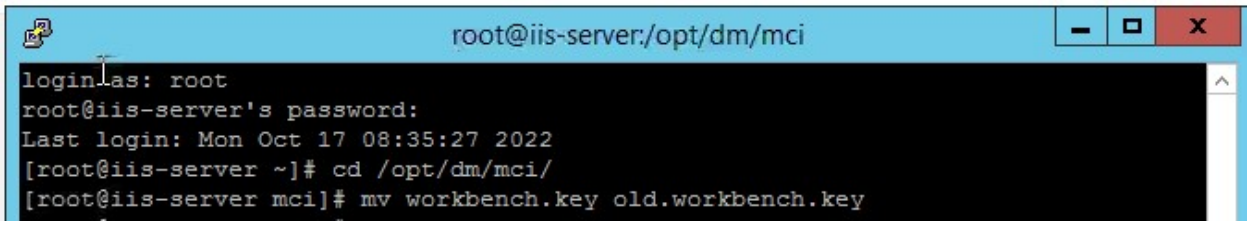
The public key will be copied into GitHub, the private key will stay on iis-server.

- Open **putty** from iis-client windows taskbar

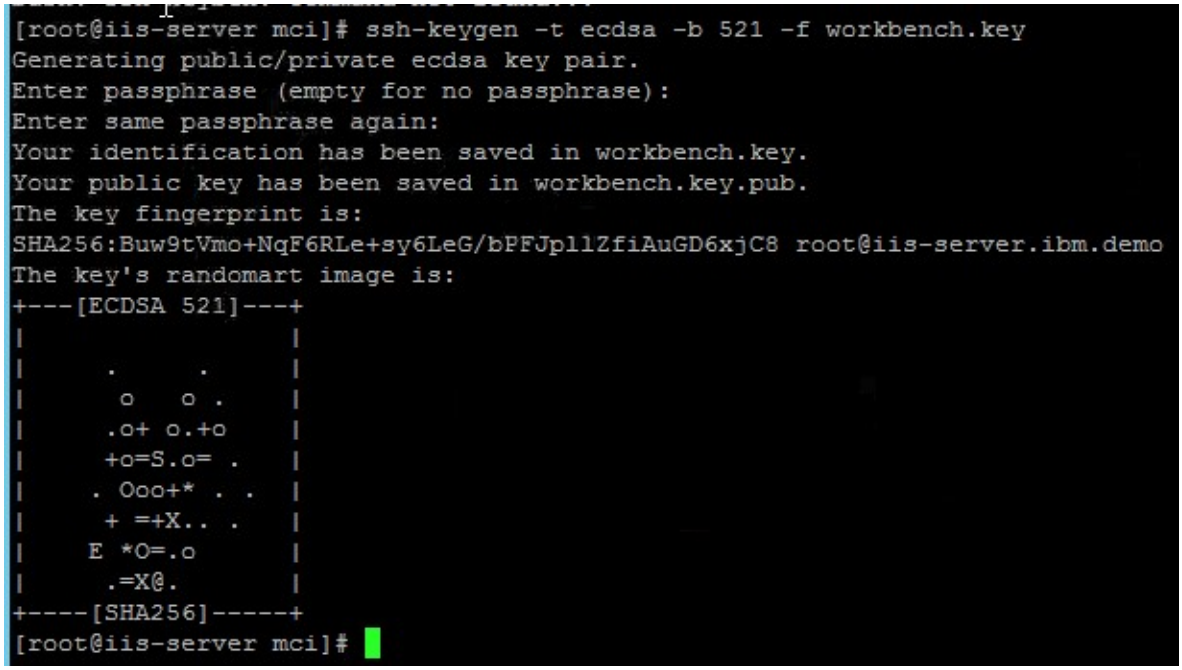


Log into iis-server with User: **root** Pass: **inf0Xerver**
and type the following commands:

cd /opt/dm/mci
mv workbench.key old.workbench.key



ssh-keygen -t ecdsa -b 521 -f workbench.key
Press ENTER/RETURN when prompted for a passphrase.



Then type the following commands to change the owner of the file:
chown mciworkb workbench.key

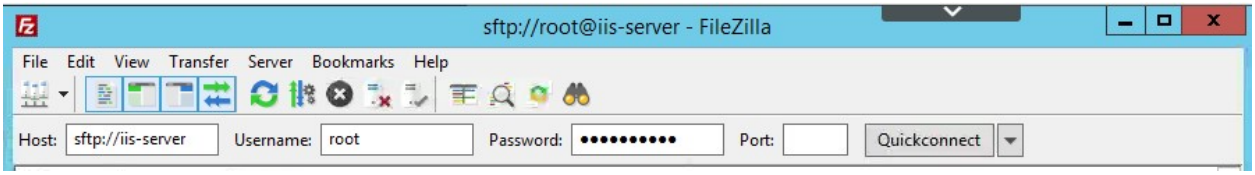
Then restart the mettleci workbench service by executing 2 commands below
sudo service dm-mettleci-workbench stop
sudo service dm-mettleci-workbench start

```
[root@iis-server mci]# chown mciworkb workbench.key
[root@iis-server mci]# sudo service dm-mettleci-workbench stop
Stopping MettleCI Workbench ...
MettleCI Workbench has been stopped
[root@iis-server mci]# sudo service dm-mettleci-workbench start
MettleCI Workbench is not running
Starting MettleCI Workbench ...
MettleCI Workbench: Java Executable location /usr/local/sbin/jdk8u275-b01-jre/bi
n/java
MettleCI Workbench: Java Vendor is AdoptOpenJDK
MettleCI Workbench: Java Version is 1.8
MettleCI Workbench has been started
```

- The private key is set up, we now need to get your public key "**workbench.key.pub**" copied into your GitHub profile.

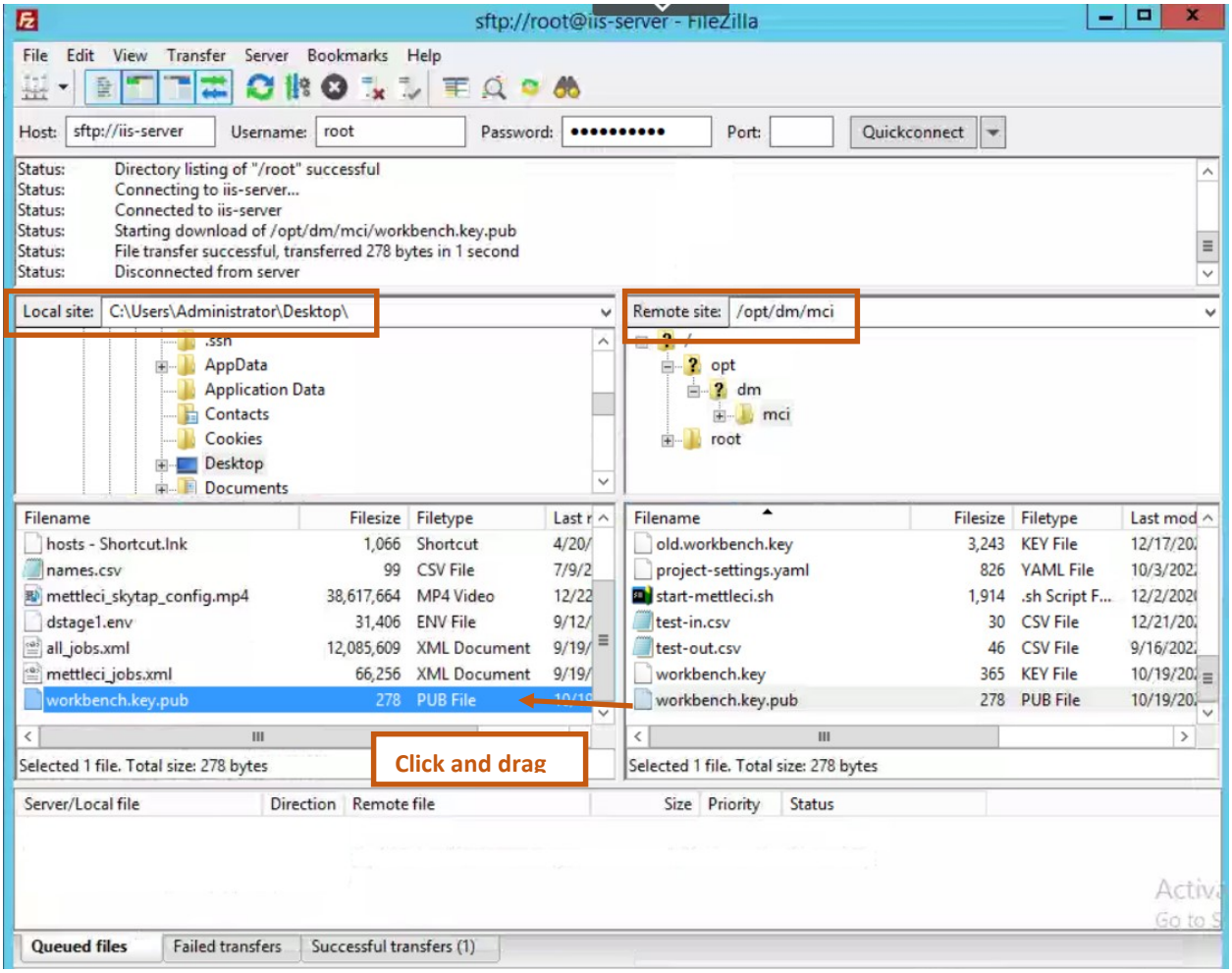
- Log into iis-client and open FileZilla Client from the Desktop icon

and enter the following credentials
Host: **sftp://iis-server**
Username: **root**
Password: **inf0Xerver**
and click **"Quickconnect"**.

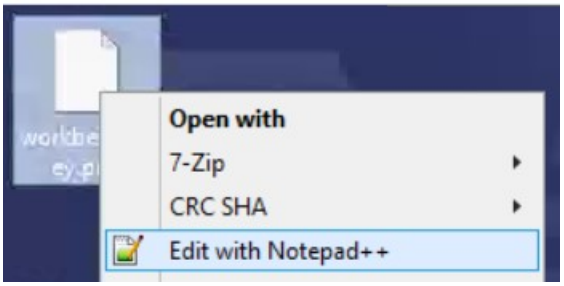


- Now, type in the "Remote site" textbox: "**/opt/dm/mci**"
and press ENTER/RETURN.
On the left side, change the "Local site" to your Desktop, or type "**C:\Users\Administrator\Desktop**" Like the screenshot below.

- **Click and drag** the file **“workbench.key.pub”** from the right side of the screen over to the left side, which will transfer the file to your Desktop.

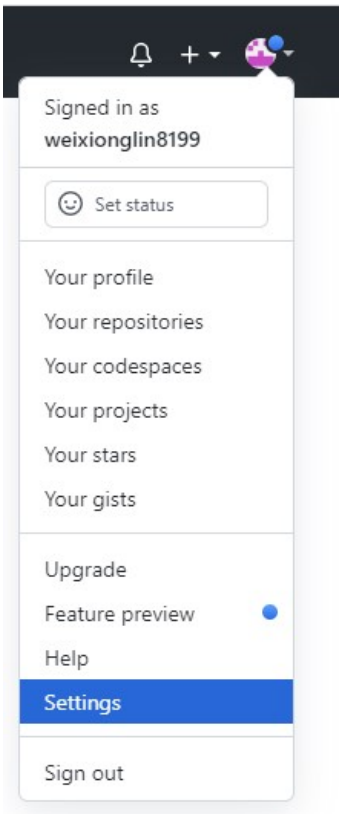


- Open **“workbench.key.pub”** using **notepad++ text editor**

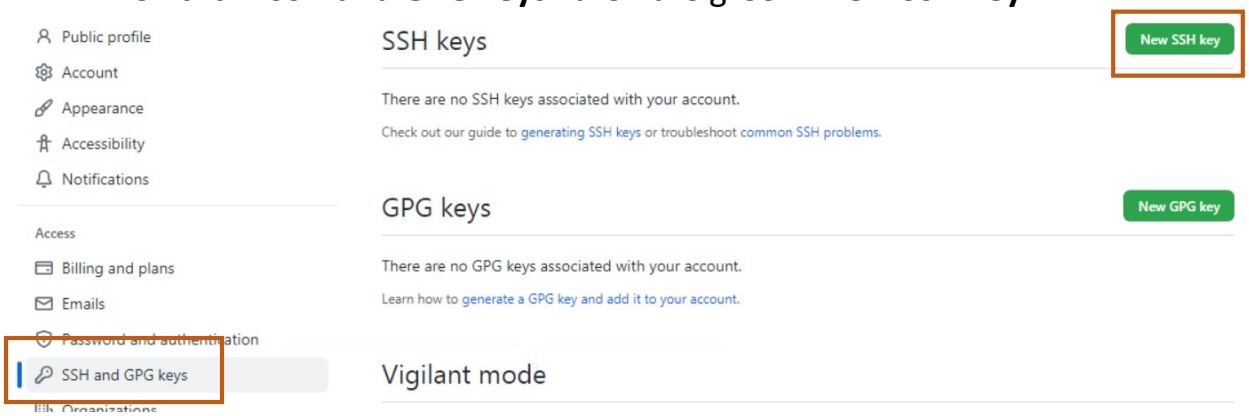


- Then copy the entire text.

- Open the GitHub page from iis-client **Chrome browser**, click your profile picture on the top right and click **“Settings**”.

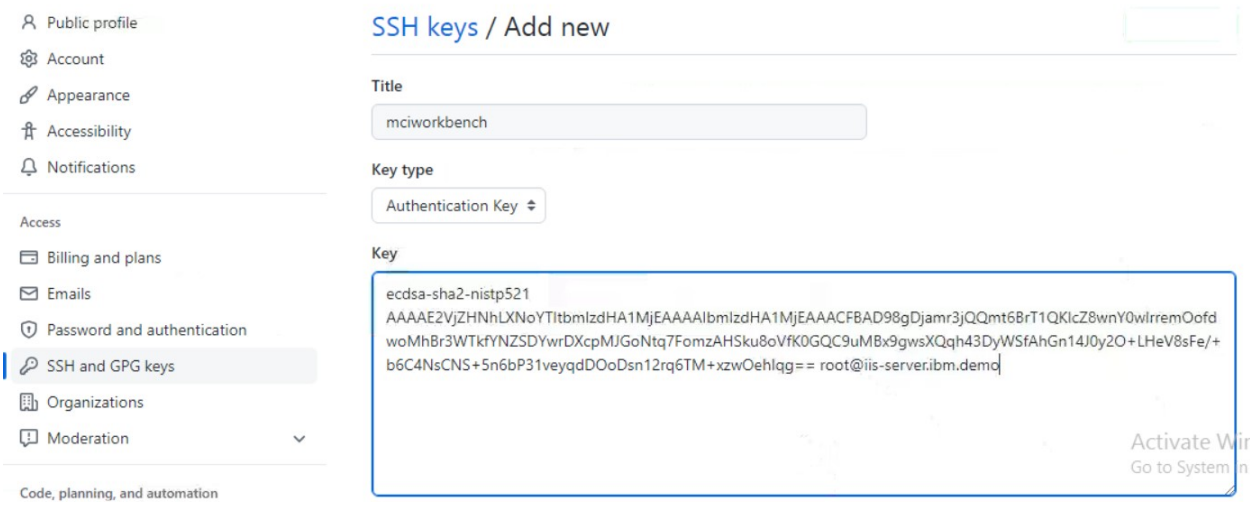


- Then click “**SSH and GPG keys**” then the green “**New SSH key**”.



- Paste the text into the "**Key**" section on the github page open.

Give any title, example '**mciworkbench**' and click "**Add SSH key**"

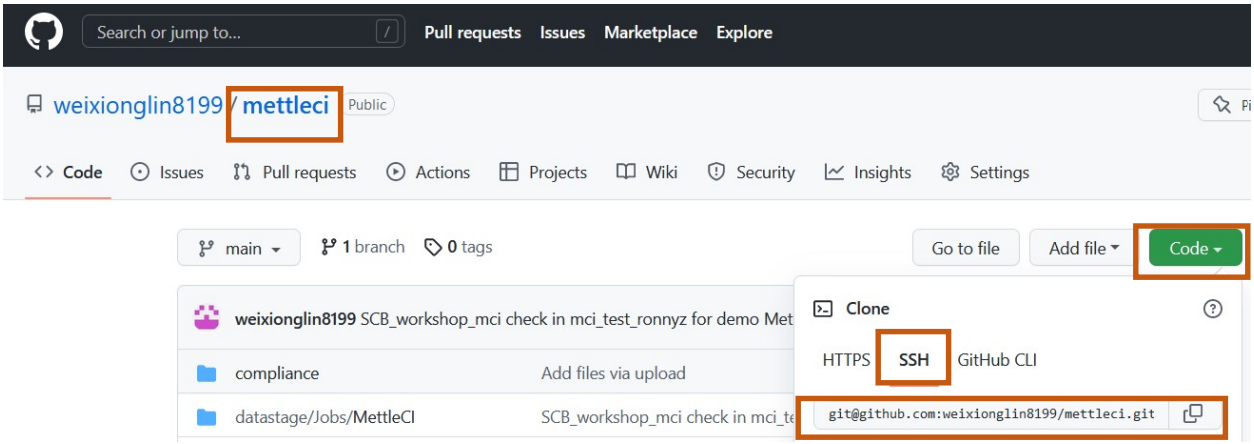


- Now go to your Github **mettleci repository,**

Click **Code**, then **SSH**

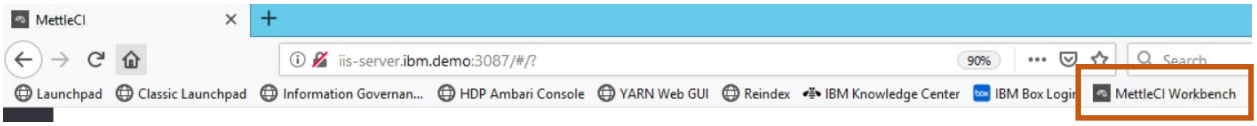
Copy the **SSH URL**

e.g: *git@github.com:weixionglin8199/mettleci.git*

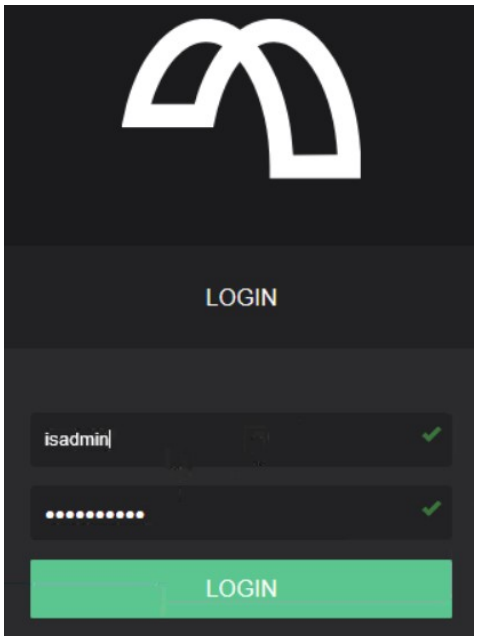


- From your Windows Remote Desktop, open mozilla firefox browser,

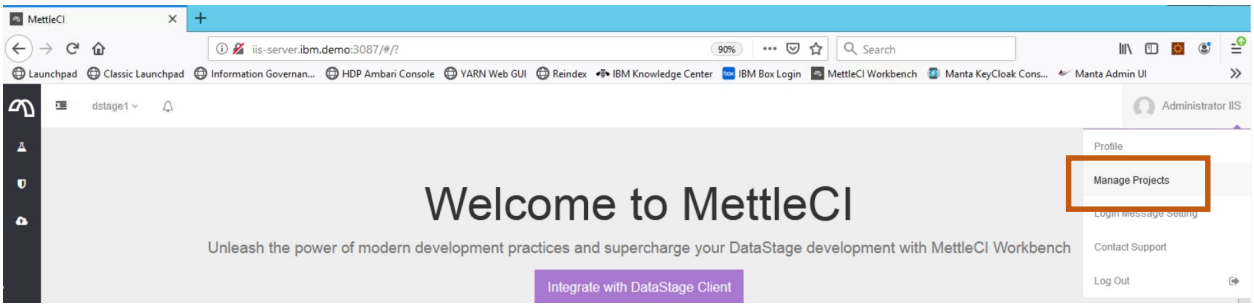
click the **MettleCI Workbench** bookmark toolbar



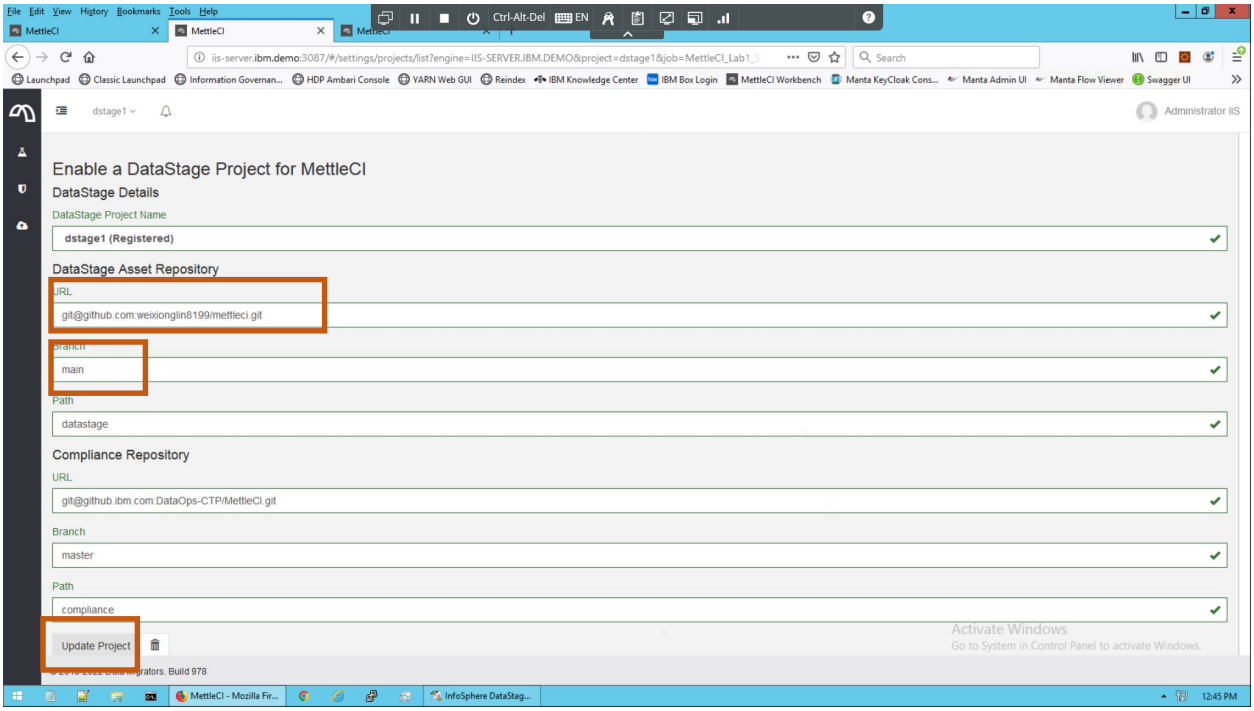
- Login using **isadmin/inf0Xerver**



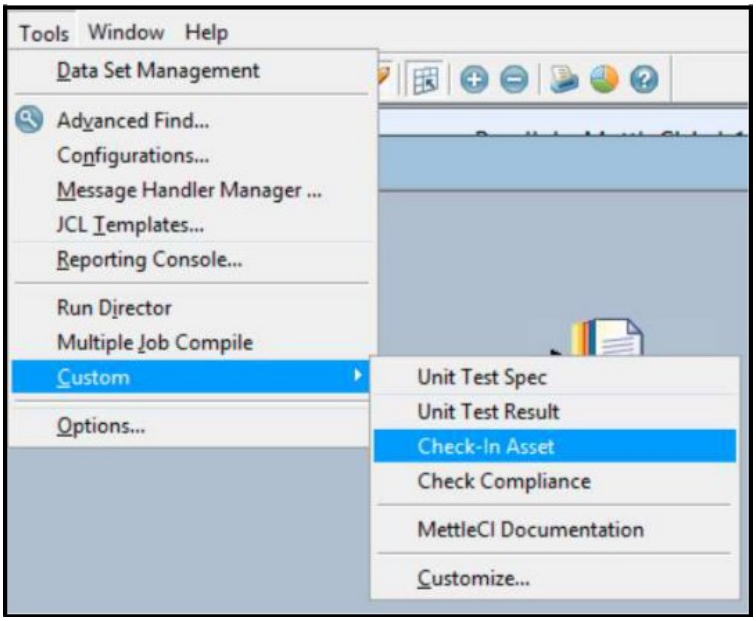
- Click on **Administrator IIS → Manage Projects** on the top right



- Change the **DataStage Asset Repository** with **your GitHub repository SSH URL** that you copy above e.g: *git@github.com:weixionglin8199/mettleci.git* and change the Branch value: **main**
- Then click **Update Project**



- With Designer, open the job with your name **"mci_test_<your name>" (e.g. mci_test_ronnyz").**
- Click **"Tools → Custom → Check-in Asset".**



- The check in screen will popup. You need to select/fill in below information.

    - **Include Unit Tests**
      Decide to include unit test assets or not

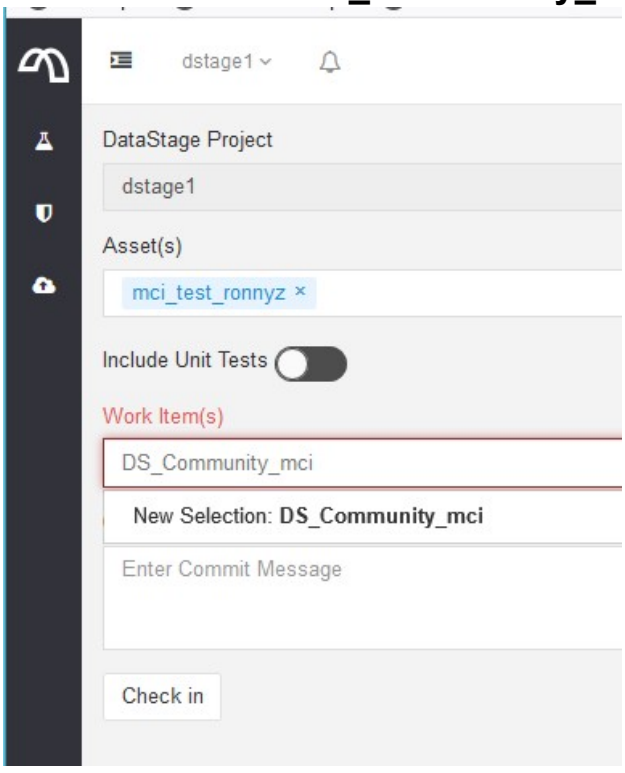
    - **Work items**
      Provide the work items name

    - **Commit Messages**
      Provide messages you want to associate with the check-in asset on GitHub

- Select/Fill in below values.

  Include Unit Tests - YES
  Work items
  – “DS_Community_mci”

  NOTE: You will get a red warning message, just click **“New Selection: DS_Community_mci”**



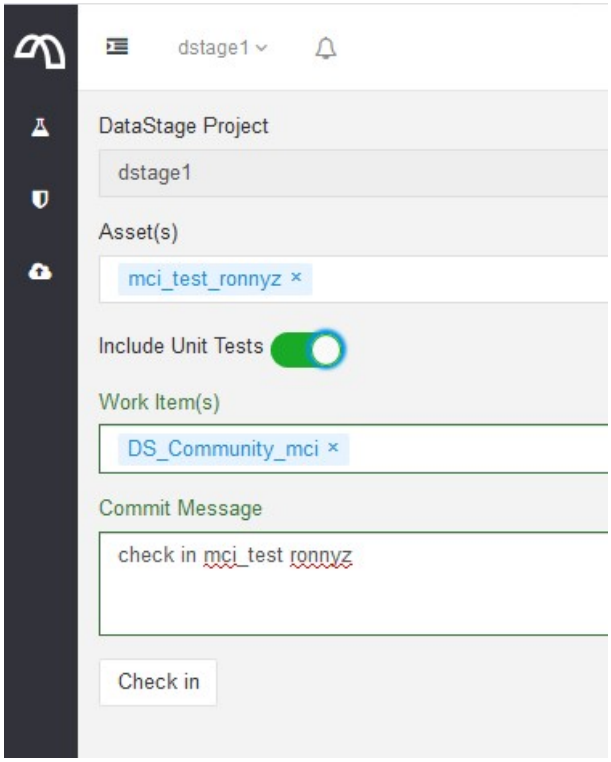
MettleCI can integrate with many work item solutions such as: Jira, Azure DevOps, ServiceNow, etc, but in this lab we don't' integrate with any work item solutions.

Commit Message

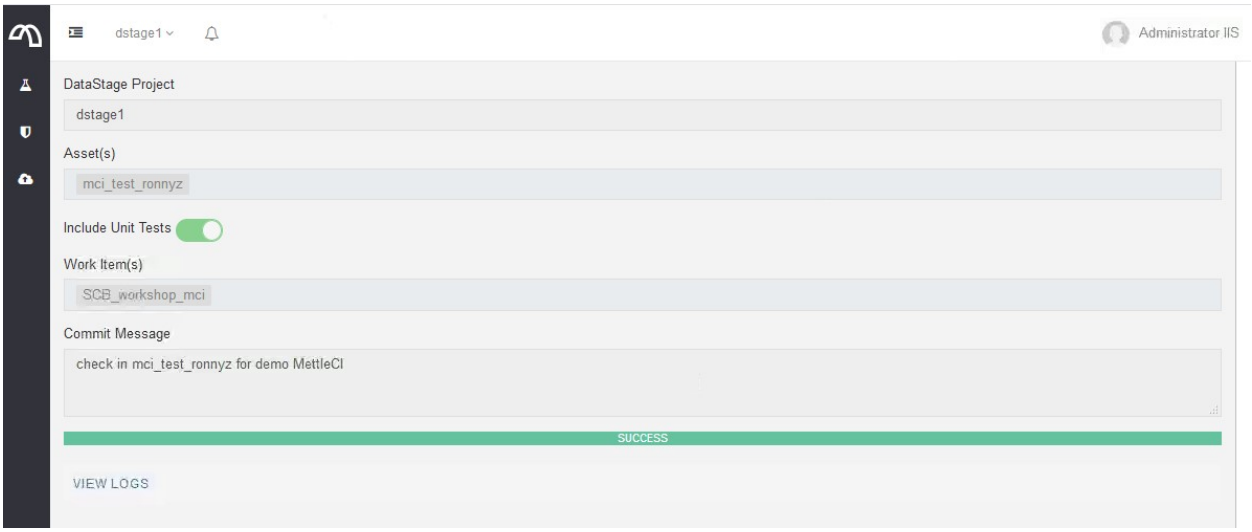
– "Check in mci_test_<your name>"

When ready, click "**Check In**",

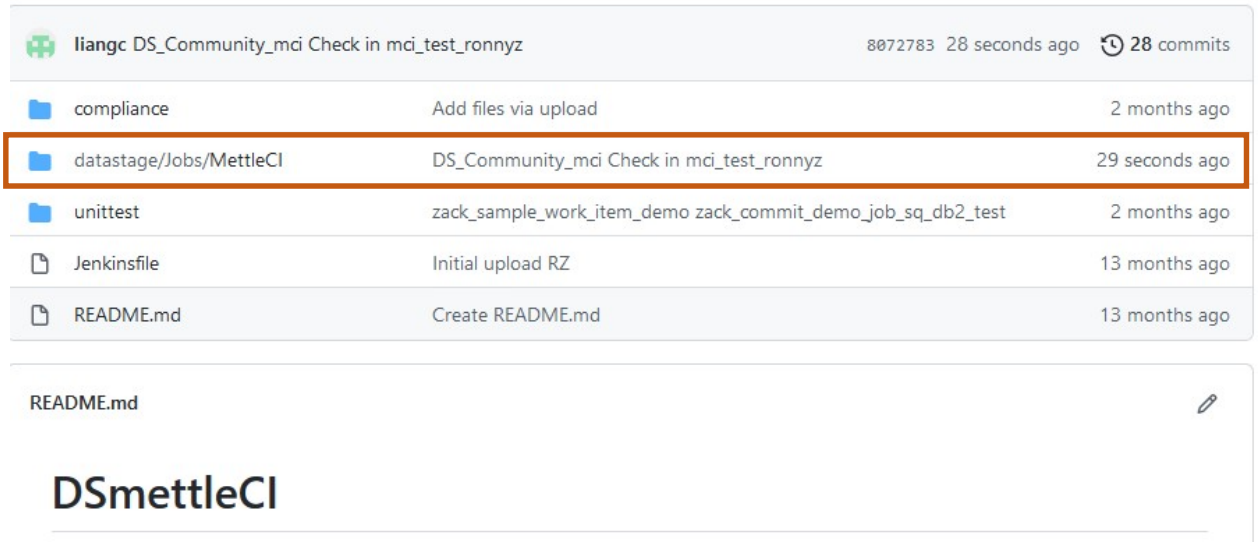
please note that check in for the 1st time will take a bit longer



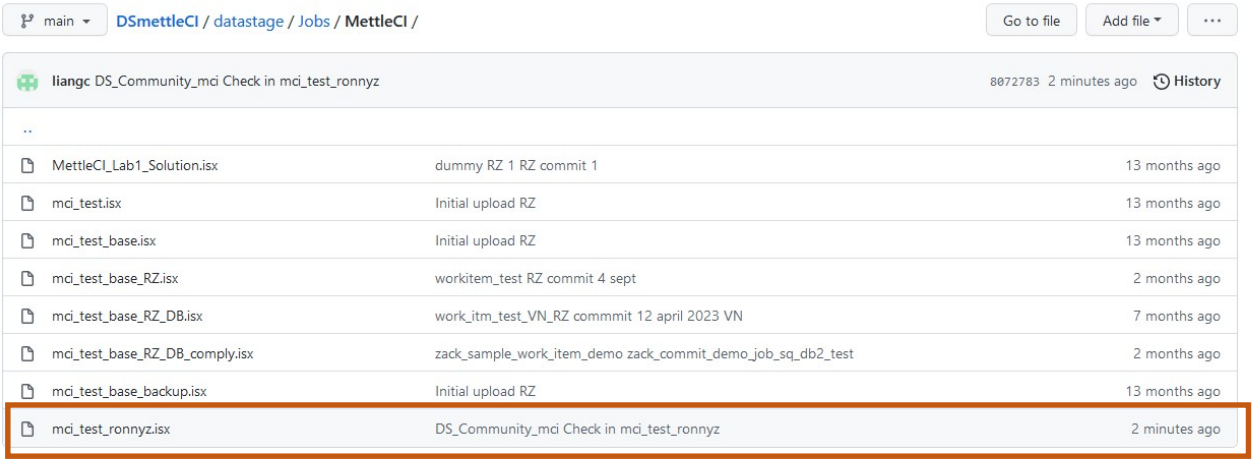
- You will see the check-in process running then with a SUCCESS message.



- **Refresh your GitHub MettleCI main page.** You will see the assets you just check in under **"datastage/Jobs/MettleCI" and "unitest/mci_test_<your name>".**

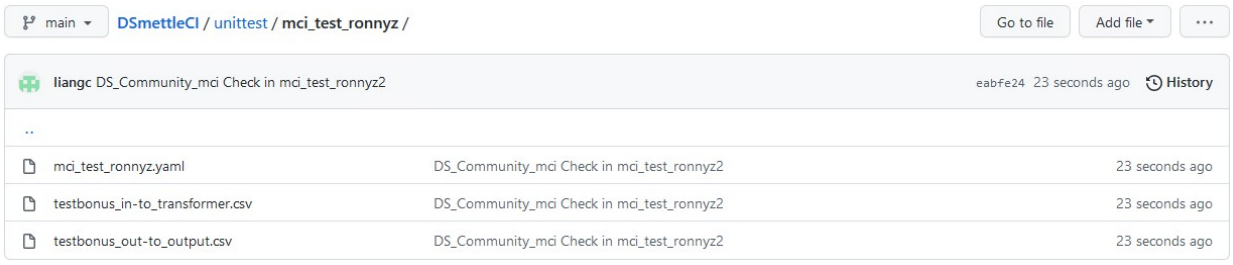


- Click **“datastage/Jobs/MettleCI”** link.
  You will see your job **“mci_test_<your name>”** is checked in to the repository with the isx file format of your job.



- Click **“<your Github username>/mettleci”** link on the top.
  Click **“unittest/mci_test_<your name>”** link to review the unit test assets you checked in. You will see below assets are checked in.

1. **Your unit test input file,**
2. **Your unit test output file,**
3. **Your unit test specification in yaml**



THIS IS THE END OF THE HANDS ON LAB