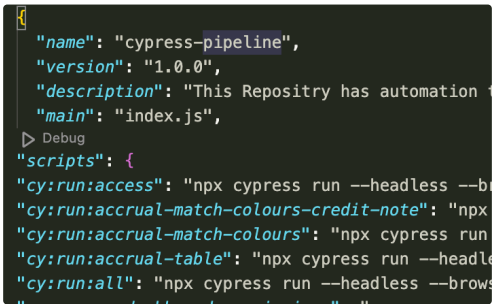
Using QA Automation

The Repo is housed at SAML single sign-on for Raft · GitLab

Adding a test to the run

Two file will need editing
`.gitlab-ci.yml` and `package.json`

In the `package.json` file, you will need to add the name and the directory of the test, add the line to the `scripts` block at the top of the file (try to remain alphabetical)




```
"cy:run:<TEST NAME>": "npx cypress run --headless --browser chrome --spec <ROUTE TO THE TEST",
```

eg `"cy:run:my-new-test": "npx cypress run --headless --browser chrome --spec cypress/e2e/ap/accrual-table/my-new-test.spec.js",`

In the `.gitlab-ci.yaml` we will need to add the test block (try to remain alphabetical)

```
<TEST NAME>:
  extends: .cypress_test
  script:
    - npm run cy:run:<TEST NAME>
  rules:
    - if: '$ALL == "Y" && $TESTS_TO_RUN == "" && $CODE == "" && ($TITLE == null || $TITLE == "") && ($TAG == null || $TAG == "") && ($FILE == null || $FILE == "")'
  needs:
    - <PREVIOUS TEST IN FILE>
  allow_failure: true
```

eg

```
multishipment-accrual-match:
  extends: .cypress_test
  script:
    - npm run cy:run:multishipment-accrual-match
  rules:
    - if: '$ALL == "Y" && $TESTS_TO_RUN == "" && $CODE == "" && ($TITLE == null || $TITLE == "") && ($TAG == null || $TAG == "") && ($FILE == null || $FILE == "")'
  needs:
    - ml-extraction
```

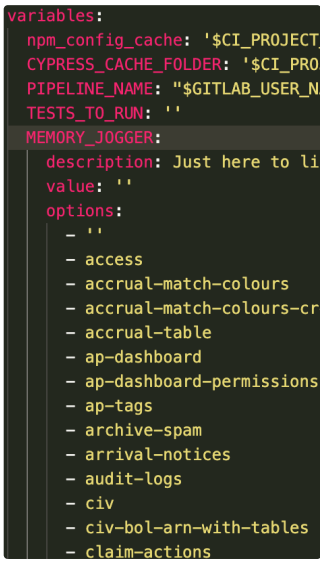
```
  allow_failure: true

my-new-test:
  extends: .cypress_test
  script:
    - npm run cy:run:my-new-test
  rules:
    - if: '$ALL == "Y" && $TESTS_TO_RUN == "" && $CODE == "" && ($TITLE == null || $TITLE == "") && ($TAG == null || $TAG == "") && ($FILE == null || $FILE == "")'
  needs:
    - multishipment-accrual-match
  allow_failure: true

netchb-hawb:
  extends: .cypress_test
```

We also need to add the test to the memory jogger var. This is null and only acts to help the user, please again place it alphabetical.

> It could be used in the future for input but not at this testing stage



Running a Test

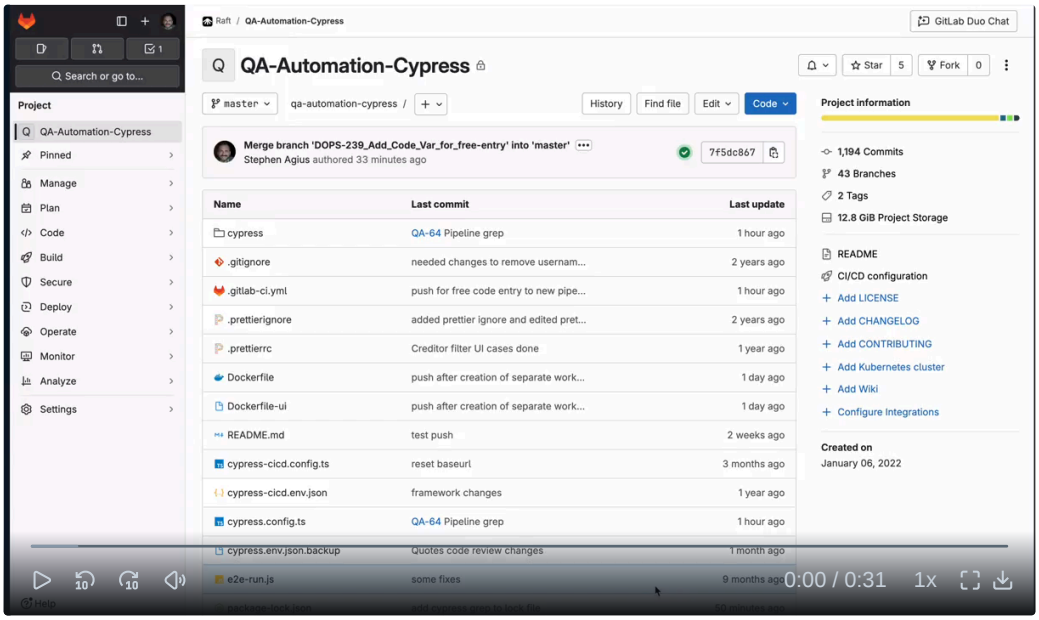
List Of Vars	
VAR	VALUES / NOTES / OPTIONS
MEMORY JOGGER	USED TO PROVIDE A LIST OF TESTS THAT CAN BE USED AND REFERENCED IN THE `TESTS_TO_RUN` VAR
PIPELINE_NAME	THE NAME YOU WANT TO USE FOR THE PIPELINE
ALL	IF `Y`, ALL TESTS WILL RUN
TESTS_TO_RUN	A COMMA SEPARATED LIST OF TESTS. THESE CAN BE FOUND IN THE `MEMORY JOGGER` VAR

CODE	THIS IS FREE ENTRY CODE INPUT AS IF RUNNING LOCALLY
FILE	A DIRECTORY LOCATION FOR A SPEC FILE
TAG	A TAG TO GREP FOR
TITLE	A TEST TITLE TO GREP FOR
BRANCH	TO ALLOW A BRANCH URL TO REPLACE THE DEFAULT

Here is a short video guiding through the run of all tests
Firstly find the Build - Pipelines link
We then can create a new pipeline run by selecting the Run Pipeline button
We then can select the name of the pipeline by providing a naming VAR `PIPELINE_NAME`

ALL

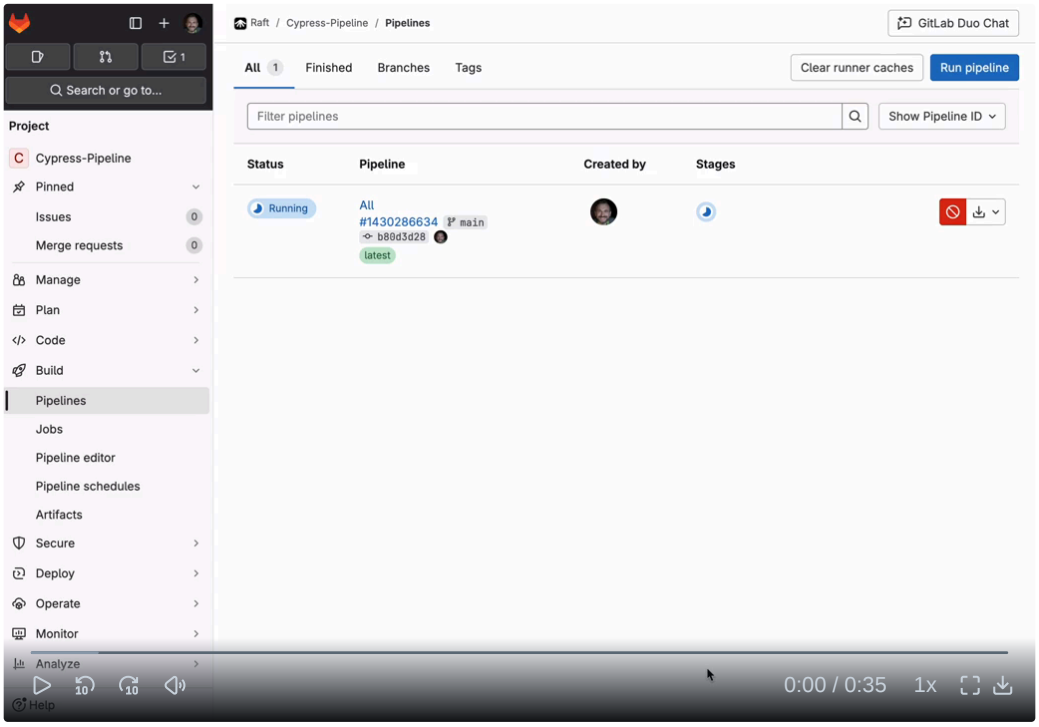
To run all tests, provide the VAR `ALL` a VALUE of `Y` .



If we need to run a selected test. There is a memory jogger provided.
You can select as many tests as required comma separated using the VAR `TESTS_TO_RUN`

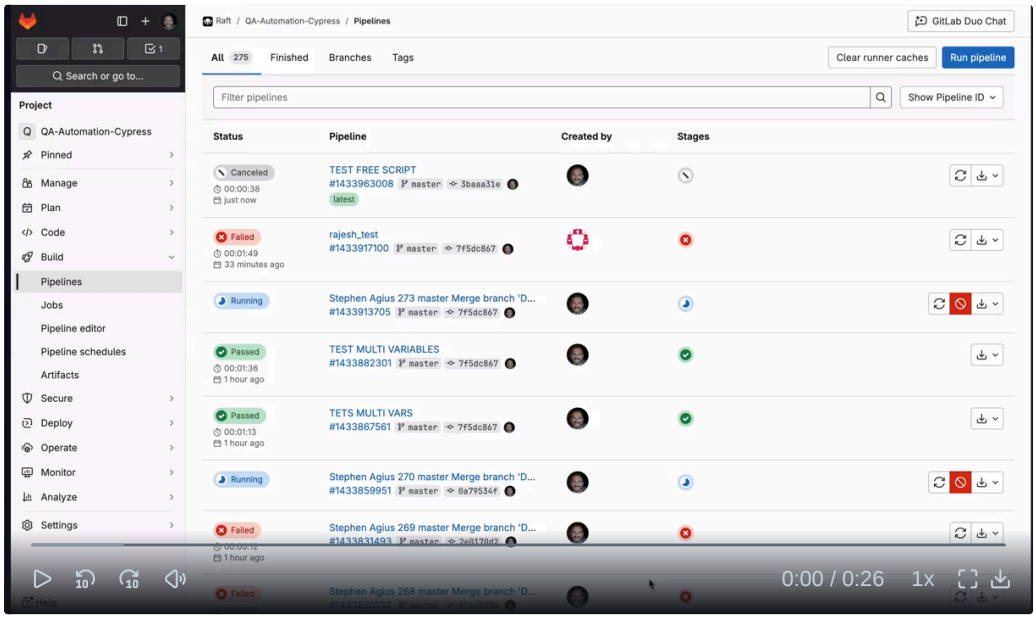
TESTS_TO_RUN
Select tests to run

PIPELINE_NAME
Give pipeline a name



If we wish to enter free commands as if running locally. Enter the VAR `CODE` with the VALUE eg `npx cypress run --headless chrome --spec  cypress/e2e/teams/access.spec.js`

CODE
The is the option to run a free entry command



A requested option is for a select set of options

FILE

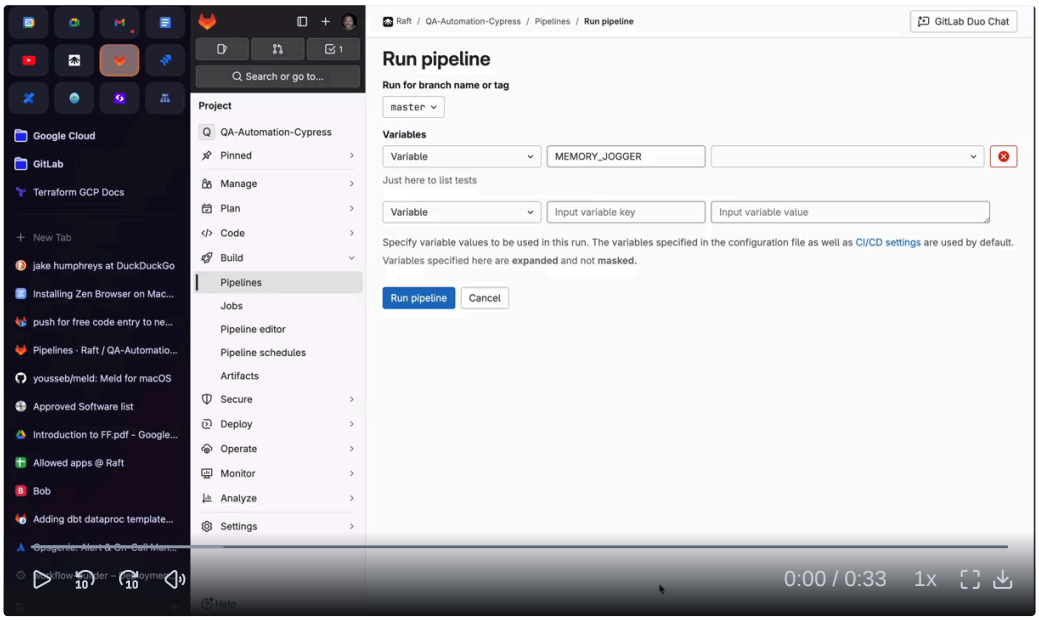
A spec file referenced via directory

TAG

A tag to grep for

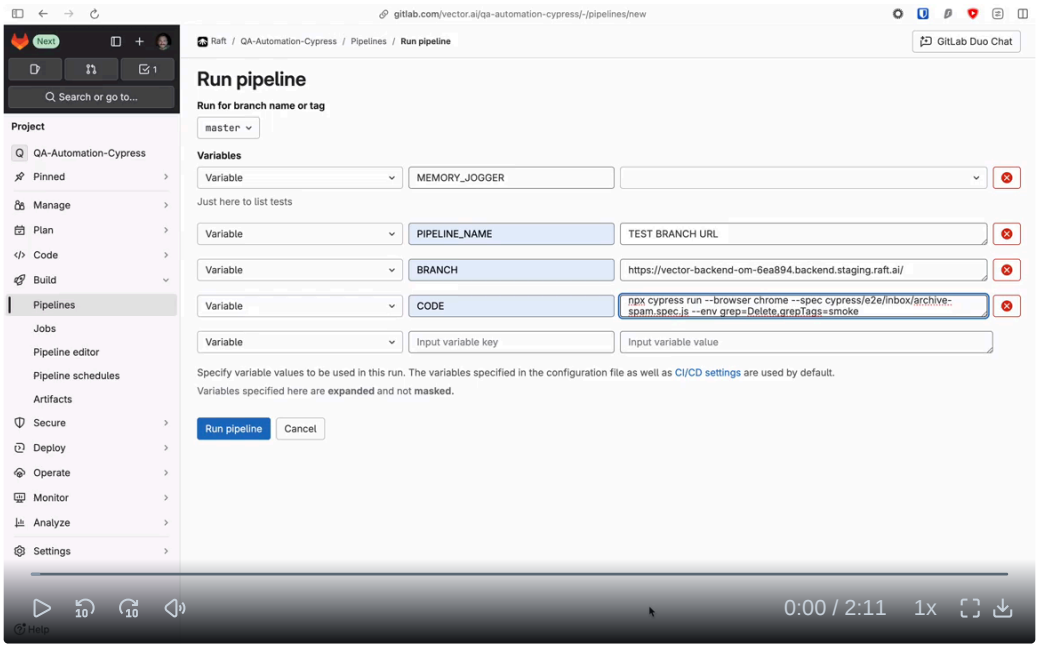
TITLE

A title to grep for



BRANCH

This var will allow the base url to that of a test branch

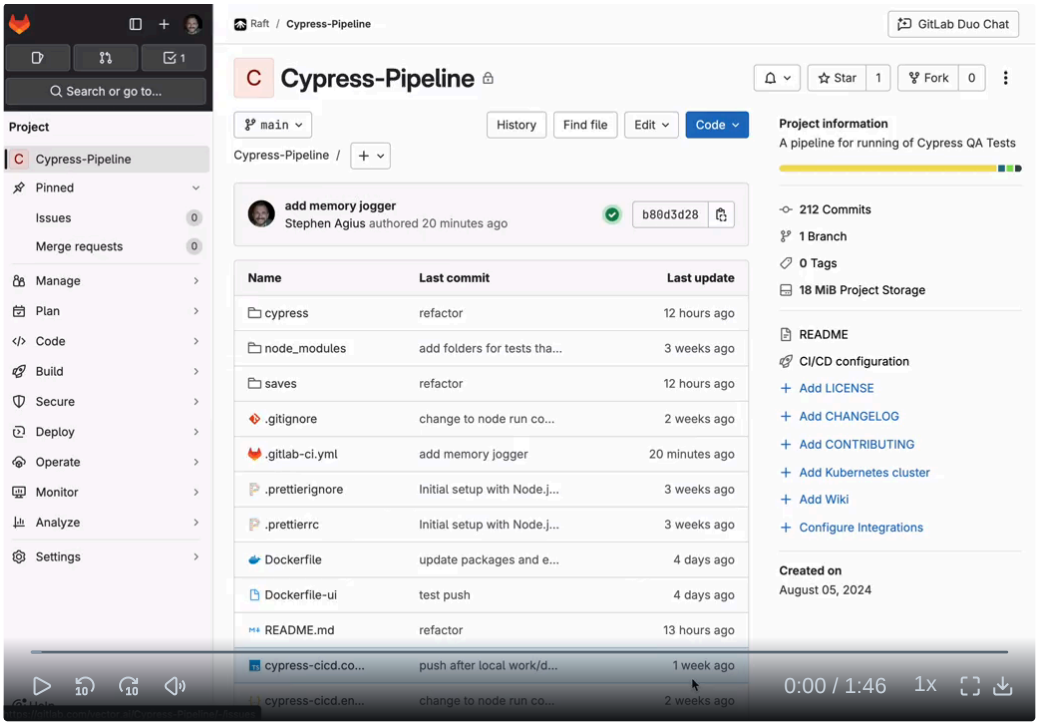


Finding Artifacts

During the run we save information, this is saved to the artifacts section of the pipeline.

To access the output, again find the pipeline you are interested in. You will be able to see the output from the run there. To see actual screen grabs and assets, you will see the artifacts buttons. Following this, you can find output that can be

viewed or saved.



> ⚠ This project is still in POC stage and is open to change

Pipeline Structure

Stage:

- test: The main stage where all test jobs are executed.

Variables:

- Defines caching directories for npm and Cypress.
- Sets up environment variables for pipeline naming and test execution.

Job Templates

.cypress_test:

- A template job that sets up common configurations for Cypress test jobs.
- Defines artifacts to be saved and sets a retry limit.

Test Execution Jobs

There are several types of test execution jobs:

1. **run_selected_tests:**
   - Runs specific tests defined in the **TESTS_TO_RUN** variable.
   - Tests are executed sequentially.
2. **run_selected_code:**
   - Executes custom code specified in the **CODE** variable.
3. **run_multivar_code:**
   - Allows running tests based on **TITLE, TAG,** or **FILE** specifications.
   - Constructs the Cypress command based on provided variables.
4. **Feature-specific test jobs:**
   - Multiple jobs are defined for different features (e.g., access, accrual-match-colours, ap-dashboard).
   - Each job runs a specific Cypress test suite.
   - Jobs are configured to run sequentially with dependencies.

Execution Flow

The pipeline is designed to:

1. Run selected tests if specified.
2. Execute custom code if provided.
3. Run all feature tests sequentially if **ALL** is set to "**Y**".

Key Features

- **Flexibility:** Allows running specific tests, custom code, or all tests.
- **Caching:** Implements caching for npm and Cypress to speed up execution.
- **Artifacts:** Saves test reports as artifacts for later analysis.
- **Retry Mechanism:** Includes a retry option for failed tests.
- **Sequential Execution:** Feature tests are run in a specific order with dependencies.