Form2Flow Setup Instructions

MB & GI

December 19, 2024

Setting up the environment requires access to a HPC cluster and a Google account. The following instructions are up to date as of Summer 2024.

- **Rclone**
  1. *On Yellowstone:* Load the Rclone module: `module load apps/rclone/1.61.0`
  2. Configure Rclone: `rclone config`. Select Google Drive and follow the basic installation (ignore the advanced configuration options). Authenticate your Google Drive account by running `rclone authorize "drive"` in an environment that can launch a browser. Once you sign into the Drive account you want to use for the class, check the terminal for a token, which you will use for the `config_token` option in the setup. Do **not** configure a Team Drive.
  3. Create the folder for the class on your Google Drive (say, `me123-fall2024`, and the local mountpoint (say, `me123-mountpoint`). See the HPCC Intranet Guide for more details: `https://hpcc-intranet.stanford.edu/user-guides/rclone-googledrive/`. To mount, use:

     `rclone mount me123:/me123-fall2024 ./me123-mountpoint --vfs-cache-mode full &`
  4. To test that the mount works as expected, move a file into the Google Drive folder and verify that it appears in the mount directory on Yellowstone.
  5. *Note:* Do not modify any of the files directly in the mount point; the files are stored on Google Drive. To circumvent this, we will create a local copy of the mount point on Yellowstone, called `./local-me123-mountpoint`.
  6. *Note:* If you are unable to view the contents of the mount point, this means that it has been disconnected. Rerun the mount command and follow any instructions that are printed.
  7. *Note:* For a persistent mount, make sure that Fusermount is installed (or load the appropriate module).
- **Google Drive**
  1. *On Google Drive:* In the class folder (`me123-fall2024`), create a folder for each student in the class, using their SUNetID; for e.g.: `jops`.
  2. Give each student read permissions to his or her own directory, so that they will be able to see the files for each assignment as soon as we make them available in the mount point on Yellowstone. For the project, give each person access to all teammates' folders as well.
- **Google Forms, Google Slides, Google Apps Scripts**
  1. *On Google Drive:* Create a Google Form for the assignment.
  2. Under the Responses tab, link it to a Google Sheet.
  3. *In the Google Sheet:* Under the Extensions Tab, click "Apps Script".
  4. Below is a sample automation script, which stores the student's email address and response to the question "Choose mesh refinement" in a file titled `responses.txt` in the path

     `me123-fall2024/studentID/assignment-t1/submission-xx/`. The submission number is automatically updated to allow a maximum of 100 submissions before overwriting.

```
function onFormSubmit(e) {
  var parentFolderName = "me123-fall2024";
  var parentFolders = DriveApp.getFoldersByName(parentFolderName);

  if (!parentFolders.hasNext()) return;
  var classFolder = parentFolders.next();

  var responses = e.namedValues;
  var userEmail = responses['Email Address'][0].split('@')[0];
  var res1 = responses['Choose mesh refinement'][0];
  var userFolder = getOrCreateFolder(classFolder, userEmail);
  // create or get the 'assignment-x' folder within the userFolder
  var assignmentFolder = getOrCreateFolder(userFolder, 'assignment-t1');
  var subNo = getNextSubmissionNumber(assignmentFolder);
  var fileName = 'responses.txt';
  var fileContent = res1;

  var subFold = assignmentFolder.createFolder('submission-' + ('0' + subNo).slice(-2));
  subFold.createFile(fileName, fileContent);
}

function getOrCreateFolder(parentFolder, folderName) {
  var folders = parentFolder.getFoldersByName(folderName);
  if (folders.hasNext()) {
    return folders.next();
  } else {
    var newFolder = parentFolder.createFolder(folderName);
    Logger.log("Created folder: " + folderName);
    return newFolder;
  }
}

function getNextSubmissionNumber(folder) {
  var folders = folder.getFolders();
  var highestNumber = 0;

  while (folders.hasNext()) {
    var match = folders.next().getName().match(/submission-(\d+)/);
    if (match) {
      var currentNumber = parseInt(match[1], 10);
      if (currentNumber > highestNumber) highestNumber = currentNumber;
    }
  }
  // Increment to allow up to 99 submissions
  return highestNumber + 1;
}

function extractFileId(url) {
  var match = url.match(/[-\w]{25,}/);
  return match ? match[0] : null;
}
```

5. The above script must be saved under the Editor tab.

6. Under the Triggers tab, add a new trigger with the following options: `Choose which function to run: onFormSubmit Choose which deployment should run: Head Select event source: From spreadsheet Select event type: On form submit`
7. Now, whenever a student makes a submission using the Form, the relevant `responses.txt` (or any other files you write in your Apps Script) will automatically appear in the mount point on Yellowstone, with a mirrored folder structure. A set of bash scripts on Yellowstone will run on a timing job to process these submissions as they arrive.

Bash

1. Below is an automation script for the test assignment `assignment-t1`, which when run, monitors the mountpoint for any new submissions. New submissions are identified by a tagging system that writes a tag for each student's submission to a directory, and cross-checks for any new ones. The new submissions are then processed: in this test case, processing involves copying over a file depending on the option selected in the student's `responses.txt` file. This portion of the script will need to be rewritten for subsequent assignments. The script also includes error handling for if the response file is not found. The Google Form used for this dummy assignment can be found here:

   `https://docs.google.com/forms/d/e/1FAIpQLSdJ0BpFgGf-0EgllkjnI3kmkFGlIRG_9KBHFwe9I8LaUlpCLA/viewform?usp=sf_link`

```
#!/bin/bash

MOUNTPOINT_DIR=/home/iaccarino/markben/me123-mountpoint
LOCAL_MOUNTPOINT_DIR=/home/iaccarino/markben/me123-files/local-me123-mountpoint
# HARDCODED
ASSIGNMENT_FILES_DIR=/home/iaccarino/markben/me123-files/assignment-files/assignment-t1
# HARDCODED
SCRIPTS_DIR=/home/iaccarino/markben/me123-files/scripts/assignment-t1
# HARDCODED
ASSIGNMENT=t1

# find all submission directories in the mountpoint
find "$MOUNTPOINT_DIR" -type d -name "submission-*" | while read SUBMISSION_DIR; do
    # extract student ID and submission number from the path
    STUDENT_ID=$(echo "$SUBMISSION_DIR" | awk -F '/' '{print $(NF-2)}')
    SUBMISSION_NO=$(echo "$SUBMISSION_DIR" | awk -F '/' '{print $NF}')

    # create a unique record file name
    RECORD_FILE="$SCRIPTS_DIR/${STUDENT_ID}-${SUBMISSION_NO}"
    # check if this submission has already been processed
    if [ -f "$RECORD_FILE" ]; then
        continue
    fi

    # mark submission as processed by creating the record file
    touch "$RECORD_FILE"

    # create the destination path in the local mountpoint, mirroring the submission structure
    LOCAL_SUBMISSION_DIR="$LOCAL_MOUNTPOINT_DIR/$STUDENT_ID/$ASSIGNMENT/$SUBMISSION_NO"
    mkdir -p "$LOCAL_SUBMISSION_DIR"

    # copy submission files to the local mountpoint
    cp -r "$SUBMISSION_DIR"/* "$LOCAL_SUBMISSION_DIR/"
```

```
    # check for responses.txt, with a 10-second retry if not found
    RESPONSE_FILE="$LOCAL_SUBMISSION_DIR/responses.txt"
    if [ ! -f "$RESPONSE_FILE" ]; then
        echo "Waiting for responses.txt to appear..."
        sleep 10
        if [ ! -f "$RESPONSE_FILE" ]; then
            echo "ERROR: responses.txt not found in $SUBMISSION_DIR"
            continue
        fi
    fi

    # read the content of responses.txt and perform actions based on the value
    RESPONSE=$(cat "$RESPONSE_FILE")
    if [ "$RESPONSE" == "Low" ]; then
        cp "$ASSIGNMENT_FILES_DIR/low.txt" "$SUBMISSION_DIR/"
    elif [ "$RESPONSE" == "Medium" ]; then
        cp "$ASSIGNMENT_FILES_DIR/medium.txt" "$SUBMISSION_DIR/"
    elif [ "$RESPONSE" == "High" ]; then
        cp "$ASSIGNMENT_FILES_DIR/high.txt" "$SUBMISSION_DIR/"
    else
        echo "ERROR: Unexpected response value in $RESPONSE_FILE"
    fi

done
```

2. To make the script run every minute, put the following line in your Cron file (`crontab -e`):
   `* * * * * /home/iaccarino/markben/me123-files/scripts/assignment-t1/run_me.sh >> /tmp/t1.log 2>&1`
3. While the Apps Script automates the folder creation for each student's submissions, the folder structure for the cluster is manually managed, since it involves only one set of files (not multiple folders per user); everything is run under the instructor's account.
4. *Note:* In my experience, the Rclone mount can become disconnected every so often (roughly once every two weeks in the Spring 2024 offering of the class), so if several students report issues with submissions, this is the likely cause. You will also need to be aware of any changes to the environment on the cluster, such as updated modules, or planned maintenance, so that you can anticipate and troubleshoot issues as needed.
5. *Note:* The directory structure of the files on the cluster is provided in the Github repository as a working reference; however, the names of the folders differ. The mountpoint is called `mountpoint`, the local mountpoint is called `local-mountpoint` and the Google Drive class folder is called `f2f-remote`.

Extensions

1. A telemetry script that collects data about student submissions (such as total submissions, which options were selected, etc) for diagnostics. Also useful to know total usage for the future when live simulations are run.
2. A system status applet hosted somewhere that is accessible by all students that reports any known outages on the cluster, whether the queue (for live simulations) is backed up, etc.
3. A report generation template script that takes images from simulations and embeds them in a LaTeXdocument that can be used to quickly start documenting the student's findings. The student should be able to toggle this in the Form.