Computer Infrastructure

Computer Infrastructure runs from September to December 2024 at ATU.

 A futuristic lecture.

Assessment

The deadline for all assessment elements is Friday, 20 December 2024.

Submission instructions are given below.

This assessment brief is available from the beginning of the semester. If anything is unclear, make sure to ask questions well before the deadline.

Purpose

The purpose of the assessment is for you to demonstrate ability in the following.

1. Use, configure, and script in a command line interface environment.
2. Manipulate and move data and code using the command line.
3. Compare commonly available software infrastructures and architectures.
4. Select appropriate infrastructure for a given computational task.

The assessment consists of three overlapping parts: a GitHub repository containing all your work (20%), a series of tasks (40%), and a small project (40%).

Feedback

Guidance on your expected progress will be provided during lectures. At certain points during the semester, submitted repositories will be reviewed. Feedback may be given individually, to the whole class, or both, depending on need. To get the most benefit from the feedback, ensure you regularly commit and push your work to GitHub.

If you have any questions, ask them well in advance of the deadline. Pay close attention to the marking scheme and the advice below. They are based on feedback given to previous students in this and other modules. You should treat it as a form of feed-forward to help you improve.

Repository (20%)

Start by creating a new repository on GitHub. Include in it only work that is part of your submission. Make sure your final submission is in the `main` branch, though you can use other branches for development. Your grade will be based on the last commit made on or before the deadline.

Submit Your Repository Info

Immediately submit your GitHub username and repository name using this form. The form is only accessible through your ATU student account.
You can find your username and repository name in the browser's address bar when viewing your repository on GitHub. In the browser's location bar your will see something like
`github.com/<username>/<repository_name>/`

Organize Your Repository

Your work should be easy to showcase during a technical interview. An interviewer should be able to understand and interact with your work without any assistance. This will have a significant impact on your mark for this and other components.

To this end, make sure your repository is well-structured and includes:

- A clear README.md.
- A relevant `.gitignore` file.
- No unnecessary files or folders.
- Lowercase file and folder names, except for files like `README.md`.
- No spaces or special characters in filenames. Underscores, hyphens, and full stops are okay.

Private Repositories

You can make your repository private. If you do, add `ianmcloughlin` as a collaborator.

Choosing a GitHub Username

Your GitHub username will be visible to potential employers. Choose wisely. You can use your real name, but a pseudonym is also fine if it's reasonable. Avoid using your student number. You can change your username by following the instructions on GitHub.

Tasks (40%)

Complete all tasks in your repository. Remember to commit and push your work regularly.

In Task 8, you will write a report summarizing your work on Tasks 1 to 7, so keep notes as you go. You may find it helpful to draft parts of Task 8 while working through the other tasks.

Task 1: Create Directory Structure

Using the command line, create a directory (that is, a folder) named `data` at the root of your repository. Inside `data`, create two subdirectories: `timestamps` and `weather`.

Task 2: Timestamps

Navigate to the `data/timestamps` directory. Use the `date` command to output the current date and time, appending the output to a file named `now.txt`. Make sure to use the `>>` operator to append (not overwrite) the file. Repeat this step ten times, then use the `more` command to verify that `now.txt` has the expected content.

Task 3: Formatting Timestamps

Run the `date` command again, but this time format the output using `YYYYmmdd_HHMMSS` (e.g., `20261114_130003` for 1:00:03 PM on November 14, 2026). Refer to the `date` man page (using `man date`) for more formatting options. (Press `q` to exit the man page). Append the formatted output to a file named `formatted.txt`.

Task 4: Create Timestamped Files

Use the `touch` command to create an empty file with a name in the `YYYYmmdd_HHMMSS.txt` format. You can achieve this by embedding your `date` command in backticks `` ` `` into the touch command. You should no longer use redirection (`>>`) in this step.

Task 5: Download Today's Weather Data

Change to the `data/weather` directory. Download the latest weather data for the Athenry weather station from Met Eireann using `wget`. Use the `-O <filename>` option to save the file as `weather.json`. The data can be found at this URL:
`https://prodapi.metweb.ie/observations/athenry/today`.

Task 6: Timestamp the Data

Modify the command from Task 5 to save the downloaded file with a timestamped name in the format `YYYYmmdd_HHMMSS.json`.

Task 7: Write the Script

Write a bash script called `weather.sh` in the root of your repository. This script should automate the process from Task 6, saving the weather data to the `data/weather` directory. Make the script executable and test it by running it.

Task 8: Notebook

Create a notebook called `weather.ipynb` at the root of your repository. In this notebook, write a brief report explaining how you completed Tasks 1 to 7. Provide short descriptions of the commands used in each task and explain their role in completing the tasks.

Task 9: pandas

In your `weather.ipynb` notebook, use the `pandas` function `read_json()` to load in any one of the weather data files you have downloaded with your script. Examine and summarize the data. Use the information provided data.gov.ie to write a short explanation of what the data set contains.

Project (40%)

In this project, you will automate your `weather.sh` script to run daily and push the new data to your repository. The following steps will create the necessary GitHub Actions workflow.

1. **Create a GitHub Actions Workflow:** In your repository, create a folder called `.github/workflows/` (if it doesn't already exist). Inside this folder, create a file called `weather-data.yml`. This file will define the GitHub Actions workflow.

2. **Run Daily at 10am:** Use the `schedule` event with `cron` to set the script to run once a day at 10am. Include also the `workflow_dispatch` event so you can test the workflow.
3. **Use a Linux Virtual Machine** In the workflow file, specify that a Ubuntu virtual machine should be used to run the action.
4. **Clone the Repository** Have the workflow clone your repository.
5. **Execute the `weather.sh` Script** Add a step that runs your `weather.sh` script.
6. **Commit and Push Changes Back to the Repository** Finally, configure the workflow to commit the new weather data and push those changes back to your repository.
7. **Test the Workflow** Commit and push the workflow to your repository. Check the logs in GitHub to ensure that the `weather.sh` script runs correctly, that new data is being committed.

Marking Scheme

Each component will be graded based on the following four categories. Each category carries equal weight.

Remember, your repository is what will be evaluated. It should demonstrate evidence of the criteria outlined for each category. That said, the examiners' overall impression of the submission may affect marks in each category.

You are expected to make steady progress on the assessment throughout the semester. This should be reflected in your commit history. Huge commits, especially late in the semester, will not be accepted. At any stage you may be asked to discuss the work to date in your repository.

If you encounter issues with your repository, seek help well before the deadline. Do not delete any files or commits without first consulting the lecturer.

Research

- Evidence of research on relevant topics.
- Appropriate referencing.
- Building upon the literature and documentation.
- Comparisons to similar work.

Development

- Clear, concise, and correct code.
- Appropriate tests.
- Knowledge of different approaches and algorithms.
- Clean architecture.

Documentation

- Clear explanations of concepts.
- Concise comments in code and elsewhere.
- Appropriate README for repository.

Consistency

- Tens of commits, each representing a reasonable amount of work.
- Literature, documentation, and code evidencing work on the assessment.
- Evidence of reviewing and refactoring.

Advice

The freedom provided in open-style assessments such as this one can feel challenging. You need to decide how to start, what content to include, how much to cover, and how to make the work your own.

The assessment is designed to provide you with opportunities for independent thinking and decision-making. Employers value graduates who can take initiative, work independently, and make design decisions with minimal guidance. We expect you to have basic programming knowledge and be able to find information on your own.

You should have a clear plan before you start coding. Your submission should include both your plan and an explanation of any design choices you made.

Make sure to follow ATU's policies and regulations which are on the Student Hub. Pay particular attention to any policies on plagiarism and the Student Code. If you're unsure about anything, ask the lecturer.