Data Engineering with Dagster — Week 3

# Project 3

## Project submission

*** Github link**

Enter your github project link

Woohoo! Heads up, there's no survey for Week 3 :)

THIS ANSWER WILL NOT BE PUBLISHED ANYWHERE

How are you feeling about your project and confidence with this week's material?

- ◯ Amazing! Feeling confident.
- ◯ Amazing! Feeling confident + willing to help others!
  Note: if you select this option we'll let people know that you are willing to help.
- ◯ Not amazing, not bad. Feeling medium confidence.
- ◯ Not great. I got it done but really need some review.

Submit project →

Required fields are empty: **Github link**

## Week 3 Directory

```
week_3
├── dagster.yaml
├── data
│   ├── stock_1.csv
│   ├── stock_10.csv
│   ├── stock_2.csv
│   ├── stock_3.csv
│   ├── stock_4.csv
│   ├── stock_5.csv
│   ├── stock_6.csv
│   ├── stock_7.csv
│   ├── stock_8.csv
│   └── stock_9.csv
├── local_stack.sh
├── tests
│   ├── __init__.py
│   ├── test_answer.py
│   └── test_resources.py
├── workspace.yaml
└── workspaces
    ├── __init__.py
    ├── challenge
    │   ├── __init__.py
    │   ├── repo.py
    │   └── week_3_challenge.py
    ├── content
    │   ├── __init__.py
    │   ├── etl.py
    │   ├── io_retry.py
    │   └── repo.py
    ├── project
    │   ├── __init__.py
    │   ├── repo.py
    │   ├── sensors.py
    │   └── week_3.py
    ├── resources.py
    └── types.py
```

The directory for this week contains the following:

### Directories

- `workspaces`: Contains the code for our Dagster workspace this include separate directories for the `content`, `project` and `challenge`.
- `data`: Any data files we use for the project
- `tests`: The tests for the `project` that we will use to determine if things are working as expected

### Files

- `local_stack.sh`: Contains configuration settings for our localstack AWS instance.
- `workspace.yaml` and `dagster.yaml`: Dagster specific configuration settings for our Dagster project

## Makefile Commands

This week, you can run the following commands in the command line using the `Makefile` to help:

- `week_3_tests`: Run the tests associated with week three.
- `week_3_start`: Start the Docker compose project detached in the background.
- `week_3_down`: Stop all the containers for the Docker compose project.
- `restart_content`: Restart the Docker container running the content workspace. This can be helpful when your Docker compose is running, and you've made a change to the content workspace that you would like to be applied.
- `restart_project`: Restart the Docker container running the project workspace.

- `restart_challenge`: Restart the Docker container running the project workspace.

## Dagster Features

The pipeline from week two was a great step forward! We already have something that can pull data from an external source, process the data, and push it to a cache. The goal of this week is to improve our pipeline to make it easier to maintain. We will incorporate a few of the features we talked about this week.

### Requirements

**week_3/project/week_3.py (sensors)**

You will need to create a sensor for the `docker_week_3_pipeline`. This sensor should check the "dagster" bucket with the prefix of "prefix" and run if there are any new files. A helper function `get_s3_keys()` is provided to determine if any new S3 files are present. We will configure this function to the bucket and prefix described above, as well as the endpoint URL (since we are using a Dockerized version of S3 you will want to connect to endpoint it is running on which is `http://localstack:4566`).

This will give you a list of S3 files to work with. Remember back to our earlier discussion about sensors: You have to anticipate two situations, whether there are or are not files. If no new files are present, we want to give a `SkipReason` (The unit tests are set to check the `skip_message` of "No new s3 files found in bucket") and do nothing else. However, if there are new files present, we  want to iterate through those files and yield a `RunRequest` for each new file. That `RunRequest` will need a run key, which should be the S3 key and a run config. The config will be similar to the run config used elsewhere for `docker`. (It will need the entire config, both ops and resources.) The only difference will be the `op` configuration section. Here, we will want to pass in the S3 key from `get_s3_keys()`.

**TESTING SENSORS (OPTIONAL)**

As well as the unit tests, if you want to try out your sensor you can start the docker instance and enable your sensor within the Dagit UI.

*If this is your first time enabling the sensor it *should* trigger 10 executions. This is because the localstack bucket our sensor is pointed at has 10 files in it so the first sensor run will pick up all of these as new files to be processed. If you were to add an additional file to the bucket (assuming it has a different key name) it would also be picked up assuming the sensor is still enabled.

### week_3/project/week_3.py (retries)

This Redis client this week has been tweaked to be less reliable. It is not guaranteed that pushing data will always be successful. You need to configure your pipeline **at the job** level to handle retries. The `docker_week_3_pipeline` should be configured to handle 10 retries with a delay of 1 second in between each retry attempt. You will need to add this configuration to the provided `docker_week_3_pipeline`.

### week_3/project/week_3.py (partitions)

One last change to our setup! Our source data has changed. Before, we were receiving a single CSV file that contained all our information, but now the files will be divided by month. The file `prefix/stock_1.csv` will contain all the data for January.

We need to change the `docker_week_3_pipeline` to be partitioned by this number. We will want a set number of partitions 1 through 10 to represent all the files that now exist. You will need to change the `docker_config` to be partitioned and supply that number into the configuration. You can look at the `docker` dictionary in the file, but that partition information should dictate which "s3_key" is run.

### week_3/project/week_3.py (schedules)

Let's add two schedules, one for each of our job configurations.

- `local_week_3_schedule`

This schedule should run the `local_week_3_pipeline` every 15 minutes.

- `docker_week_3_schedule`

This schedule should run the `docker_week_3_pipeline` at the beginning of every hour. Remember that this schedule will be slightly different because the job has a partition associated with it.

Remember, you can use https://crontab.guru/ if you need help figuring out your cron syntax

**TESTING SCHEDULES (OPTIONAL)**

As well as the unit tests, if you want to try out your schedule you can start the docker instance and enable your schedules within the Dagit UI.

*If you want to see what this looks like, you may want to tweak your schedule to be every minute or so.

## Running your pipeline

There are several ways to interact with your pipeline and check to see if it's working as expected.

### Testing

The quickest way to ensure your pipeline is running as expected is to run the unit tests. To run the tests, from the root of the repo run the following command:

```
make week_3_tests
```

This will use the `pytest` framework to execute all the tests associated with the week one run project. Not all the tests will pass at first, and your goal is to see each individual test pass.

### Docker

WEEK 3 JOBS

If you try launching Docker before your tests pass, it may fail to load the workspace. That is because Dagster cannot load a workspace with errors. It is recommended you get your pipeline to pass tests before trying to run it in Docker.

However, if you want to launch Docker before you have finished, simply remove the error-causing code from `repo.py`.

We talked about starting the Docker project in the setup guide. As a reminder, the `Makefile` provides all the commands you should need to launch the service. Assuming your tests pass and your workspace loads successfully, you can try executing your pipeline. It is also a good idea to try `docker ps` to make sure you are not still running any of our containers from last week before spinning up the containers for this week.

**FINISH**

You will know that you have completed the project when all the tests pass. Feel free to try running your pipeline in different modes (local and docker).