**Introduction to CI/CD**

* **What is CI/CD?**
  + Continuous Integration:
    - Build refers to preparing everything to run your code:
      * Dependencies, given the requirements, are met
      * Docker container is instantiated
      * …
    - Testing and automatically testing
    - As soon as you merge a new feature, automatically releasing that code to either production or a testing environment
* **Why Does This Matter**
  + Visibility across the company keeps log of which versions of which applications were released
* **Why CircleCI**
  + CircleCI is a CI/CD platform
    - Hosted platform: we don’t have to install and operate it in our own service

**Setting up CircleCI**

* Show circleci\config.yml
  + Jobs: Defines the jobs for CircleCI to run
    - The job that we defined is the test\_regression\_model
    - working\_directory: Is where CircleCI is going to run the commands
    - docker: which docker image we will run the commands
      * Meaning that python 3.7.2 will be available when it comes time for us to run our commands
    - steps: the series of steps to run as part of this job, given by *command*
      * We create a virtual env
      * We activate it
      * We upgrade pip
      * We install the requirements for the regression model
      * Chmod sets the permissions so that we’re allowed to run the following script
      * We then run the script
      * Then we run the regression model test
  + Workflows: Organizes the jobs, as some jobs may have dependencies on one another
    - For now, we only have one job
* Show scripts\fetch\_kaggle\_dataset.sh
  + This command uses the Kaggle command line interface, so that something we have to install (defined in requirements).
    - This command downloads and save it in a specific directory
    - We need to use this command to save our files into our version in the cloud
    - This command requires a Kaggle API key
    - We set the KAGGLE\_USERNAME and KAGGLE\_KEY env variables in CircleCI
* Show packages\regression\_model\requirements.txt
* Show packages\regression\_model\regression\_model\datasets
  + As the datasets that we have aren’t stored in the version control, CircleCI is not able to find them
* The jobs are going to be run every time we open a pull request and merge to master. Therefore we started to automate the process of testing

**Publishing the Model to Gemfury**

* Gemfury is a private package repository that helps you reuse your code without worrying about its hosting or deployment
* We have been using *pip -e* to install our package locally.
* Now we’re going to upload our package to an index server.
  + The default index server for Python packages is called PyPi (read Py P I). But this server is public
* We will be setting up a private package index
  + A package index is just a file server
* Show packages\ml\_api\requirements.txt
  + We don’t have the *pip -e* installation method anymore, now we will be pulling our package from GemFury
* Show circleci\config.yml. We added two new jobs:
  + Test\_ml\_api:
    - Is the same as the previous job, except that now we are running the ml\_api tests
  + Train\_and\_upload\_regression\_model:
    - Standardizes both the training and publishing of our model
    - Chmod prepares for the following script
    - Set the Python path to be able to run the train pipeline script (we are training the model)
    - After the model is trained, we are uploading it to GemFury
  + In our workflows, we require that for testing the ml api we first need to train our model. But we only need that the first time we execute it, because afterwards, we will already have our model uploaded
* Show scripts\publish\_model.sh
  + SETUP=”setup.py” specifies where the setup.py file is
    - This is connected to ./scripts/publish\_model.sh *./packages/regression\_model/*
  + Python “$SETUP” sdist bdist\_wheel which generates the distribution
  + For X in $(ls dist) loops over the dist directory
    - curl -F package=@”dist/$X” “$GEMFURY\_URL” uploads the files that we find in dist to GemFury

**Testing the CI Pipeline**

* “yml aliases and anchors”, set reusable sets
* -py-deps-… checks if the requirements have changed, if they have, then we won’t restore the cache
* In workflows
  + we need to have passed some tests before we publish anything
  + we only run train\_and\_upload… if we’re merging in the master branch
* run *git checkout -b update-regression-model*
* change regression\_model version to 0.1.1 (patch)
* run *git add –all*
* run *git commit -m “increment regression model version”*
* git push origin update-regression-model
* create the pull request
* notice the tests in git and circleci
* notice the regression model in gemfury