

Testing & Monitoring ML Model Deployments: All Section Notes

Contents

| | |
|--|-----------|
| 2 - Setting the Scene & ML System Lifecycle | 6 |
| Additional Background Material | 6 |
| Setup Lectures (Python, git, jupyter, data) | 7 |
| Course Repo (you'll want to bookmark this) | 7 |
| Data Download From Kaggle | 7 |
| Test if your operating system is 32 or 64 bit | 7 |
| Git Install & Setup | 8 |
| Why use a Virtual Environment? | 8 |
| Why does 32-bit vs. 64-bit Python even matter? | 8 |
| Introduction to the Dataset Lecture | 8 |
| 3 - Testing Concepts | 9 |
| Additional Background Material | 9 |
| First Part of this Course Focus Areas | 10 |
| Hands-On Assignments | 10 |
| 4 - Unit Testing ML Systems | 12 |
| Section Additional Background Material | 12 |
| Code Conventions | 12 |
| Pytest Introduction | 12 |
| Setup Kaggle Data | 13 |
| Commands | 13 |

| | |
|--|-----------|
| Tox Introduction | 13 |
| Commands | 13 |
| Codebase Overview | 14 |
| Key Project Dependencies | 14 |
| Potentially Unfamiliar Files/Libraries/Tools in the Repo | 15 |
| Preprocessing & Feature Engineering Testing | 17 |
| Commands | 17 |
| Relevant Resources | 17 |
| Config Testing | 17 |
| Commands | 17 |
| Relevant Resources | 18 |
| Input Testing | 18 |
| Commands | 18 |
| Relevant Resources | 18 |
| Model Quality Testing | 19 |
| Commands | 19 |
| Relevant Resources | 19 |
| Tooling Lecture | 19 |
| Commands | 19 |
| Relevant Resources | 20 |
| Troubleshooting | 20 |
| 5 - Docker & Docker Compose | 24 |
| Section Additional Background Material | 24 |
| Docker & Docker Compose Installation | 25 |
| Important - Operating System Gotchas | 25 |
| Commands | 26 |
| Windows Specific Docker Issue | 26 |
| Relevant Resources | 26 |
| Commands (Windows only) | 27 |
| To undo the script | 27 |
| Docker Compose Exercise | 27 |

| | |
|---|-----------|
| Relevant Resources | 27 |
| Commands (Windows only) | 27 |
| Docker Cheatsheet | 28 |
| Advanced Commands (return to this later if you are new to docker-compose) | 29 |
| Troubleshooting This Section | 30 |
| 6 -Integration Testing ML Systems | 31 |
| Section Additional Background Material | 31 |
| Code Overview | 31 |
| Relevant Resources | 31 |
| Commands | 31 |
| Using our Open API Spec | 32 |
| Commands | 32 |
| Integration Testing Theory | 32 |
| Relevant Resources | 32 |
| 32-bit Operating System Workaround | 32 |
| Commands | 33 |
| Integration Tests - Code | 33 |
| Relevant Resources | 33 |
| Commands | 33 |
| Benchmark Tests | 34 |
| Relevant Resources | 34 |
| Troubleshooting This Section | 34 |
| 7 - Differential Testing | 37 |
| Differential Testing Implementation | 37 |
| Relevant Resources | 37 |
| Commands | 37 |
| 8 - Shadow Mode | 38 |
| Section Additional Background Material | 38 |
| Tests in Shadow Deployments | 38 |
| Relevant Resources | 38 |
| Shadow Mode Code Overview - DB Setup | 38 |

| | |
|---|-----------|
| Relevant Resources | 39 |
| Commands | 39 |
| Setup Tests for Shadow Mode | 39 |
| Relevant Resources | 39 |
| Commands | 39 |
| Shadow Mode - Asynchronous Implementation | 41 |
| Relevant Resources | 41 |
| Commands | 41 |
| Shadow Mode - Populate DB Script | 41 |
| Commands | 41 |
| Assess Shadow Model | 42 |
| Commands | 42 |
| Troubleshooting This Section | 42 |
| 9 - Monitoring Metrics with Prometheus | 45 |
| Section Additional Background Material | 45 |
| Metrics for ML Systems | 46 |
| Relevant Resources | 46 |
| Prometheus & Grafana Overview | 46 |
| Relevant Resources | 46 |
| Basic Prometheus Setup | 46 |
| Relevant Resources | 46 |
| Commands | 47 |
| Adding Metrics | 47 |
| Relevant Resources | 47 |
| Commands | 48 |
| Adding Grafana | 48 |
| Relevant Resources | 48 |
| Commands | 48 |
| Infrastructure Metrics | 49 |
| Relevant Resources | 49 |
| Commands | 49 |

| | |
|---|-----------|
| Adding Metrics Monitoring to Our Example Project | 49 |
| Relevant Resources | 49 |
| Commands | 49 |
| Creating an ML System Grafana Dashboard | 50 |
| Relevant Resources | 50 |
| Commands | 50 |
| Troubleshooting This Section | 51 |
| Running Prometheus In Production - Further Reading | 55 |
| On Multiprocessing | 55 |
| Deploying a Prometheus Server & Managed Prometheus Options: | 55 |
| 10 - Monitoring Logs with Kibana | 55 |
| Additional Background Material | 55 |
| Elastic Stack Overview | 56 |
| Relevant Resources | 56 |
| Kibana Hands-On Exercise | 56 |
| Relevant Resources | 56 |
| Commands | 56 |
| Integrating Kibana into the Example Project | 58 |
| Relevant Resources | 58 |
| Commands | 58 |
| Relevant Resources | 59 |
| Commands | 59 |
| Troubleshooting This Section | 61 |
| Running The Elastic Stack In Production - Further Reading | 61 |
| Deploying The Elastic Stack & Managed Prometheus Options: | 61 |

2 - Setting the Scene & ML System Lifecycle



Additional Background Material

- On “**Reproducibility**” see this conference talk Sole and I gave:
<https://www.datasciencefestival.com/video/dsf-mainstage-day-2019-soledad-and-chris-samiullah-train-in-data/>
- “Hidden technical debt in Machine learning systems” Sculley et al. (2015)
<https://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.pdf>
- “The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction” Breck et al. (2017)
<https://research.google/pubs/pub46555/>
- Continuous Delivery For Machine Learning (CD4ML)
<https://martinfowler.com/articles/cd4ml.html>
- “Rules of Machine Learning” by Martin Zinkevich
https://developers.google.com/machine-learning/guides/rules-of-ml#top_of_page

Pandas & Scikit-Learn

Pandas and scikit-learn are used heavily throughout this course, and a basic understanding of both libraries is a prerequisite. For those who are feeling a little rusty, I would recommend these articles, in this order:

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

<https://towardsdatascience.com/pandas-for-people-in-a-hurry-59d966630ae0>

https://scikit-learn.org/stable/getting_started.html

<https://www.machinelearningplus.com/python/101-pandas-exercises-python/>

And Wes McKinney (the creator of Pandas) has also written the book *Python for Data Analysis*:

<https://wesmckinney.com/pages/book.html>

Setup Lectures (Python, git, jupyter, data)

Course Repo (you'll want to bookmark this)

<https://github.com/trainindata/testing-and-monitoring-ml-deployments>

Data Download From Kaggle

<https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data>

Test if your operating system is 32 or 64 bit

Windows 7 / 8 / 10

<https://support.microsoft.com/en-gb/help/15056/windows-32-64-bit-faq>

MacOS

<https://apple.stackexchange.com/questions/140938/how-do-i-find-out-if-my-os-x-10-9-4-is-32-bit-or-64-bit>

Linux

<https://www.fastwebhost.in/blog/how-to-find-if-linux-is-running-on-32-bit-or-64-bit/>

Git Install & Setup

The Atlassian guide is a good resource:

<https://www.atlassian.com/git/tutorials/install-git>

Why use a Virtual Environment?

This is a solid overview:

<https://realpython.com/python-virtual-environments-a-primer/>

Why does 32-bit vs. 64-bit Python even matter?

As per the scikit-learn docs:

“Since a model internal representation may be different on two different architectures, dumping a model on one architecture and loading it on another architecture is not supported.”

See: https://scikit-learn.org/stable/modules/model_persistence.html

Since the `gradient_boosting_model` package we work with in this course is trained on a 64-bit architecture (i.e. my laptop), then it won't work on a 32-bit laptop. In industry this is typically handled in one of two ways:

1. By using an agreed interchange format, such as onnx (<https://onnx.ai/>)
2. By dockerizing everything

Our work around in this course, for those students with 32 bit operating systems, is the latter.

For those wishing to learn more about the multitude of challenges the pickle (and therefore joblib) library opens up, see this great talk: <https://www.youtube.com/watch?v=7KnfGDajDQw>

Introduction to the Dataset Lecture

Repo location:

- You will also find a `requirements.txt` file in this directory which contains the notebook dependencies

- You should manually place the kaggle dataset in this directory in your cloned, local version.

https://github.com/trainindata/testing-and-monitoring-ml-deployments/tree/master/research_phase

3 - Testing Concepts



Additional Background Material

- Google's *Site Reliability Engineering* is one of the best references out there, it's available for free here: <https://landing.google.com/sre/sre-book/toc/index.html>
- Martin Fowler's testing guide is pretty comprehensive. If you are new to testing this may be overwhelming: <https://www.martinfowler.com/testing/>
- *Obey the Testing Goat* by Harry Percival is a good applied introduction to Test Driven Development (TDD): <https://www.obeythetestinggoat.com/>
- Advanced, narrow vs. broad integration tests: <https://martinfowler.com/bliki/IntegrationTest.html>
- On test coverage: <https://www.martinfowler.com/bliki/TestCoverage.html>

- Dependency Injection Theory: <https://www.martinfowler.com/articles/injection.html>
- Dependency Injection Practical example: <https://christophergs.github.io/python/2018/09/25/elegant-flask-apis-pt-1/>

First Part of this Course Focus Areas

| Types of ML System Testing and their Timing | | | | |
|---|-----------------------|--|--------------------|---|
| Unit Testing | Integration Testing | System Testing | Production Testing | Monitoring |
| Inputs Schema | Reproducible Training | Reproducible Predictions | Shadow Deployments | Monitor System Serving Performance (latency etc.) |
| Model Specification | Pipeline Testing | Model-Infra Compatibility | Canary Releasing | Monitor Model Quality (accuracy, skew, staleness) |
| Input Feature Code | | Validate Model Quality (differential tests) | | |
| Model Quality (algorithmic) | | Validate Computational Performance (benchmark tests) | | |
| Model Quality (benchmarking) | | Validate System Performance (load tests) | | |

Hands-On Assignments

Dataset

https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html

Scikit-Learn Recap

<https://www.datacamp.com/community/blog/scikit-learn-cheat-sheet>

Python Unittest Docs

<https://docs.python.org/3/library/unittest.html>

Python Class Inheritance Recap

<https://www.digitalocean.com/community/tutorials/understanding-class-inheritance-in-python-3>

Assignment 1 Notebook

https://github.com/trainindata/testing-and-monitoring-ml-deployments/blob/master/exercise_notebooks/unit_testing_exercise/unit_testing_input_data.ipynb

Assignment 2 Notebook

https://github.com/trainindata/testing-and-monitoring-ml-deployments/blob/master/exercise_notebooks/unit_testing_exercise/unit_testing_data_engineering.ipynb

Assignment 3 Notebook

https://github.com/trainindata/testing-and-monitoring-ml-deployments/blob/master/exercise_notebooks/unit_testing_exercise/unit_testing_model_predictions_quality.ipynb

Assignment 4 Notebook

https://github.com/trainindata/testing-and-monitoring-ml-deployments/blob/master/exercise_notebooks/unit_testing_exercise/unit_testing_model_configuration.ipynb

4 - Unit Testing ML Systems



Section Additional Background Material

- Getting started with testing in Python: <https://realpython.com/python-testing/>
- strictyaml design justifications: <https://hitchdev.com/strictyaml/#design-justifications>

Code Conventions

- Python style guide (PEP8): <https://www.python.org/dev/peps/pep-0008/>
- Type Hints (PEP 484): <https://www.python.org/dev/peps/pep-0484/>
- Literal String Interpolation (PEP 498): <https://www.python.org/dev/peps/pep-0498/>

Pytest Introduction

Pytest fixture documentation

<https://docs.pytest.org/en/latest/fixture.html>

Example refactoring tests to use pytest fixtures

<https://semaphoreci.com/community/tutorials/testing-python-applications-with-pytest>

Pytest raising exception documentation

<http://doc.pytest.org/en/latest/assert.html#assertions-about-expected-exceptions>

Setup Kaggle Data

Commands

```
git checkout {Unit Testing Production ML Code - Code Overview commit hash}
```

Kaggle url for house prices:

<https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data>

The data should go in this directory:

https://github.com/trainindata/testing-and-monitoring-ml-deployments/tree/master/packages/gradient_boosting_model/gradient_boosting_model/datasets

-> Rename train.csv to houseprice.csv

Tox Introduction

We use **tox** extensively in this course, which is a tool for standardizing testing in Python. Because of the functionality offered by tox, we don't have to worry about setting up virtual environments, setting PYTHONPATH environment variables, and many other headaches. Tox is cross platform compatible (OS X, Linux, Windows). There is a bit of syntax and config to pickup, but it is much less hassle than alternatives such as Makefiles and bash scripts.

Commands

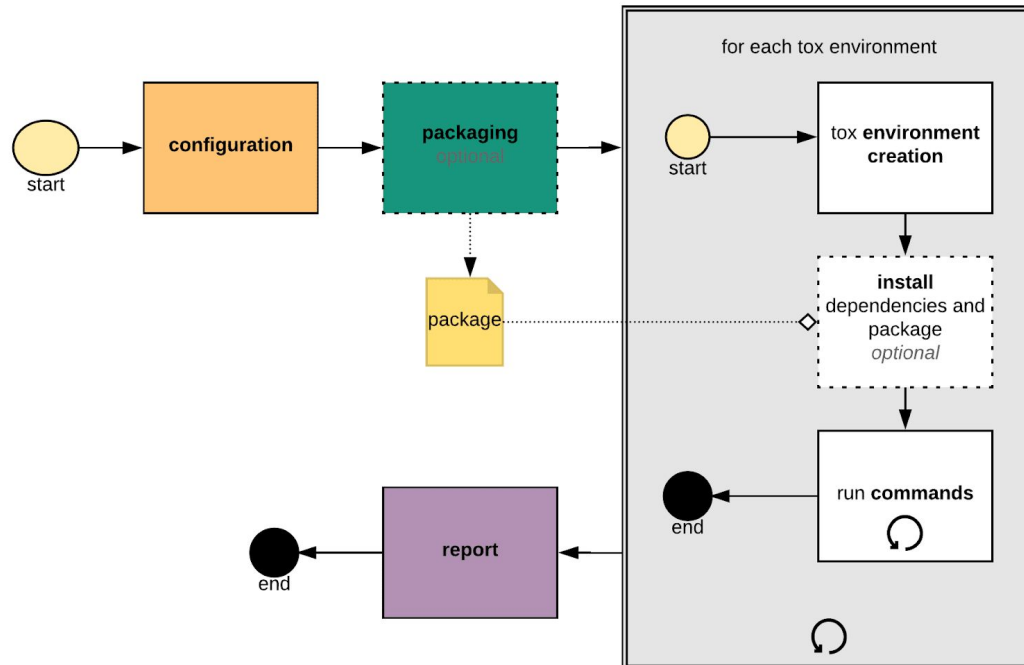
You need to have tox installed. Install with pip:

```
pip install tox
```

Tox documentation:

<https://tox.readthedocs.io/en/latest/>

*Tox Workflow Diagram (source:
<https://tox.readthedocs.io/en/latest/index.html/>)*



Codebase Overview

- Sublime Text: <https://www.sublimetext.com/>
- Monorepo primer: <https://www.atlassian.com/git/tutorials/monorepos>
- Python packaging user guide: <https://packaging.python.org/>
- Understanding model publishing & Python packages: <https://martinfowler.com/articles/cd4ml.html> (see the “Model Serving” section).
- Scikit-Learn Pipelines: <https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html>

Key Project Dependencies

- Feature Engine: https://github.com/solegalli/feature_engine
- Joblib: <https://joblib.readthedocs.io/en/latest/> (Note that joblib used to be included within scikit-learn but this usage is now deprecated and it is treated as a separate library).
- strictyaml: <https://github.com/crdoconnor/strictyaml>

- Pydantic: <https://pydantic-docs.helpmanual.io/>
- Marshmallow: <https://marshmallow.readthedocs.io/en/stable/>

Potentially Unfamiliar Files/Libraries/Tools in the Repo

`.gitignore` file instructs git which files not to track in version control.

More details here: <https://guide.freecodecamp.org/git/gitignore/>

`MANIFEST.in` file is used when creating Python packages for distribution, it provides instructions on which files in the repo should be tracked as part of the package. More details here: <https://packaging.python.org/guides/using-manifest-in/>

`VERSION` file is one approach used in Python packages for tracking the version. This is largely a matter of personal preference, as there are quite a few different ways you can version your Python packages. See here for a discussion on the topic:

<https://stackoverflow.com/questions/458550/standard-way-to-embed-version-into-python-package>

And here is the a deferred PEP (Python Enhancement Proposal) which provides some useful context on the topic:

<https://www.python.org/dev/peps/pep-0396/>

`__init__.py` files are used by Python to identify package directories. Removing an `__init__.py` file from a directory means that Python will no longer search within the submodules of that directory for other Python files, which is likely to cause imports to fail in your project.

See discussion here for more: <https://stackoverflow.com/questions/448271/what-is-init-py-for>

`config.yml` is a YAML file where the app's config is specified. YAML is a human-readable data-serialization language. See here for docs <https://yaml.org/> which is growing in popularity as it is both approachable to humans and can be processed by tools. The format is sensitive to whitespace though, so make sure to copy examples exactly and use spaces rather than tabs.

There are many ways to write config in Python, and different trade offs to consider. Here is a thought provoking article on the pitfalls of using Python code for configuration:

<https://hitchdev.com/strictyaml/why-not/turing-complete-code/>

Pathlib was introduced in Python 3.4, and is the modern way to work with directories in Python (as opposed to using `os.path`). Docs are here: <https://docs.python.org/3/library/pathlib.html>

A reasonable question to ask is: Why do we need `marshmallow` *and* `Pydantic`. The answer is that `Pydantic` is primarily a parsing library, not a validation library. Validation is a means to an end: building a model which conforms to the types and constraints provided.

See this section of the `Pydantic` documentation which discusses this distinction:

<https://pydantic-docs.helpmanual.io/usage/models/>

This github issue also clarifies the distinction:

<https://github.com/samuelcolvin/pydantic/issues/578>

Throughout this course we use version ranges for our dependencies. For example:

`numpy>=1.17.3,<1.18.0`

Meaning that pip will search for a version of `numpy` greater than or equal to 1.17.3, but less than version 1.18.0. So for example, if `numpy` version 1.17.23 came out, that would be installed. But `numpy` version 1.18.1 would not. This is extremely useful as it means that incremental small fixes in later patch versions (see semantic versioning: <https://semver.org/>) will be installed. However, larger fixes in minor/major versions will not be installed, since those might contain breaking changes.

See discussion here:

<https://stackoverflow.com/questions/8795617/how-to-pip-install-a-package-with-min-and-max-version-range>

And PEP explaining distribution versioning here:

<https://www.python.org/dev/peps/pep-0440/>

Logging in the Repo

<https://docs.python.org/3/howto/logging.html#configuring-logging-for-a-library>

Preprocessing & Feature Engineering Testing

Commands

```
git checkout {Unit Testing Production ML Code - Test Preprocessing commit hash}
```

Make sure you are in the

testing-and-monitoring-ml-deployments/packages/gradient_boosting_model/

directory, then run (the first time this will take some time to install dependencies):

```
tox
```

You should see 4 tests passed.

Relevant Resources

- Understanding Pytest fixture scope:
<https://docs.pytest.org/en/latest/fixture.html#order-higher-scoped-fixtures-are-instantiated-first>
- Scikit-Learn Pipeline `_fit` method:
<https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/pipeline.py#L262>
- Tox config specification: <https://tox.readthedocs.io/en/latest/config.html>

Config Testing

Commands

```
git checkout {Unit Testing Production ML Code - Test Config commit hash}
```

Make sure you are in the

testing-and-monitoring-ml-deployments/packages/gradient_boosting_model/

directory, then run (this will take some time to install dependencies as we use the -r flag):

```
tox -r
```

You should see 7 tests passed.

Relevant Resources

- Pytest temporary directories: <https://docs.pytest.org/en/latest/tmpdir.html>
- Tox rebuild flag: <https://tox.readthedocs.io/en/latest/config.html#cmdoption-tox-r>
- The chapter on Release Engineering discusses some of the challenges of service config: <https://landing.google.com/sre/sre-book/chapters/release-engineering/>
- Pydantic validation: <https://pydantic-docs.helpmanual.io/usage/validators/>

Input Testing

Commands

```
git checkout {Unit Testing Production ML Code - Test Input Validation commit hash}
```

Make sure you are in the

```
testing-and-monitoring-ml-deployments/packages/gradient_boosting_model/
```

directory, then run (this will take some time to install dependencies as we use the -r flag):

```
tox
```

You should see 10 tests passed.

Relevant Resources

- Marshmallow schemas:
<https://marshmallow.readthedocs.io/en/stable/quickstart.html#declaring-schemas>
- Simple video on marshmallow validation:
<https://www.youtube.com/watch?v=YOD3HarcVXg>

Model Quality Testing

Commands

```
git checkout {Unit Testing Production ML Code - Test Model Quality commit hash}
```

Make sure you are in the

testing-and-monitoring-ml-deployments/packages/gradient_boosting_model/

directory, then run (this will take some time to install dependencies as we use the -r flag):

```
tox -r
```

You should see 12 tests passed.

Relevant Resources

- Link the regression model package we install during this lecture:
<https://github.com/trainindata/deploying-machine-learning-models>
- Standard library round function (note the ndigits explanation):
<https://docs.python.org/3/library/functions.html#round>
- Why Mean Squared Error (MSE):
<https://towardsdatascience.com/https-medium-com-chayankathuria-regression-why-mean-square-error-a8cad2a1c96f>
- Reproducibility in Keras:
<https://keras.io/getting-started/faq/#how-can-i-obtain-reproducible-results-using-keras-during-development>

Tooling Lecture

Commands

```
git checkout {Unit Testing Production ML Code - Add Tooling commit hash}
```

Make sure you are in the

testing-and-monitoring-ml-deployments/packages/gradient_boosting_model/

directory, then run (this will take some time to install dependencies if we use the -r flag & have added additional test environments for stylechecks/typechecks):

```
tox -e unit_tests
```

Run the mypy typechecks

```
tox -e typechecks
```

Run the flake8 stylechecks

```
tox -e stylechecks
```

Run all checks at once

```
tox -r
```

You should see 12 tests passed + stylechecks + typechecks passing

Relevant Resources

- **Black Python code formatter:** <https://github.com/psf/black>
- ***mypy optional static typing***
Docs: <https://mypy-lang.org/>
Config file (mypy.ini): https://mypy.readthedocs.io/en/latest/config_file.html
- ***Flake8 linting***
Docs: <https://flake8.pycqa.org/en/latest/>
Config file (setup.cfg): [Configuring Flake8](#)

Troubleshooting

Error

```
E   FileNotFoundError: [Errno 2] File
b'C:\\path\\testing-and-monitoring-ml-deployments\\packages\\gra
dient_boosting_model\\gradient_boosting_model\\datasets\\housepri
ce.csv' does not exist:
b'C:\\Users\\path\\testing-and-monitoring-ml-deployments\\packag
es\\gradient_boosting_model\\gradient_boosting_model\\datasets\\h
ouseprice.csv'
```

Solution

You have not downloaded the data from kaggle and/or you have not placed it in the correct directory and/or you have not renamed it.

For Anaconda users:

If you are running tox for the first time, open up the Anaconda prompt and follow these instructions:

1. Check the list of environments you already have:

```
conda info -e
```

2. Create a new environment called tmmlm where you will run these tests:

```
conda create -n tmmlm python=3.7
```

3. Switch into the new environment:

```
activate tmmlm
```

4. Install tox in the new environment:

```
pip install tox
```

5. Change into the unit tests directory:

```
cd path/to/cloned/repo/packages/gradient_boosting_model
```

6. Optional, check that tox.ini is in the folder:

```
ls
```

7. Run the tests:

```
tox -r
```

Error

When you run a tox command, if you see the warning:

```
C:\Users\chris\repos\testing-and-monitoring-ml-deployments>tox
-e unit_tests
ERROR: tox config file (either pyproject.toml, tox.ini,
setup.cfg) not found
```

Solution

You are not in the correct directory. Navigate to where the tox.ini file is located and run the command again.

Error

When you run a tox command, if you see the warning:

```
ERROR: could not install deps [-rrequirements.txt]; v =
InvocationError("'C:\\path\\testing-and-monitoring-ml-deployment
s\\packages\\gradient_boosting_model\\.tox\\unit_tests\\Scripts\\
\\pip.EXE' install -rrequirements.txt", 1)
```

Solution

You need to either (1) Add your Python install location to your system PATH or (2) You are an Anaconda user, and you should run the above command from the Anaconda Prompt.

Error (Windows)

When you install tox, if you see the warning:

```
WARNING: The scripts tox-quickstart.exe and tox.exe are
installed in 'C:\some\directory\on\your\computer' which is not
on PATH.
```

Consider adding this directory to PATH or, if you prefer to suppress this warning, use `--no-warn-script-location`.

Then when you try and run the tox command you will probably see:

```
'tox' is not recognized as an internal or external command,
operable program or batch file.
```

Solution

This is because your Python installation is in a non-standard directory. You need to ensure that the directory shown in the tox installation warning above is added to your system PATH environment variable. See this answer on setting environment variables:

<https://superuser.com/questions/284342/what-are-path-and-other-environment-variables-and-how-can-i-set-or-use-them>

Error (MacOS)

```
OSError: [WinError 123] The filename, directory name, or volume label syntax is incorrect
```

Solution

It is likely that you have an entry in your PYTHONPATH which conflicts with the course repo (this may be the case if you have taken our other course on model deployments). Delete the other entries in your PYTHONPATH to resolve this error.

Problem

```
`ERROR: No pyproject.toml or setup.py file found. The expected locations are:`.
```

Solution

Run tox with `skipsdist=True` set in the tox.ini file (ensure your tox.ini files matches the course repo).

5 - Docker & Docker Compose

Section Additional Background Material

- Docker key concepts: <https://docs.docker.com/get-started/>
- How is Docker different from a Virtual Machine?
<https://stackoverflow.com/questions/16047306/how-is-docker-different-from-a-virtual-machine>
- Dockerfile reference: <https://docs.docker.com/engine/reference/builder/>
- Dockerhub Introduction: <https://docs.docker.com/docker-hub/>
- Official Python Docker images on Dockerhub: https://hub.docker.com/_/python
- Flask docs: <https://flask.palletsprojects.com/en/1.1.x/>
- The Flask Mega-Tutorial:
<https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world>
- Docker Compose Introduction: <https://docs.docker.com/compose/>
- Practical Introduction to Docker Compose:
<https://hackernoon.com/practical-introduction-to-docker-compose-d34e79c4c2b6>
- Docker-Compose config file reference: <https://docs.docker.com/compose/compose-file/>
- Volumes in Docker:
<https://blog.container-solutions.com/understanding-volumes-docker>
- *Further reading: Introduction to Kubernetes (beyond the scope of this course, but worth knowing about):*
[What Is Kubernetes? An Introduction to the Container Orchestration Platform](https://www.kubernetes.io/docs/tutorials/kubernetes-basics/)
- What is virtualbox? (used by Docker toolbox under the hood)
<https://www.techrepublic.com/article/virtualbox-everything-the-pros-need-to-know/>

Docker in Practice by Ian Miell and Aidan Hobson Sayers

<https://www.manning.com/books/docker-in-practice-second-edition>

Docker Deep Dive by Nigel Poulton

Docker & Docker Compose Installation

Important - Operating System Gotchas

You need to install different Docker versions depending on your operating system. Check Docker system requirements carefully, to decide which Docker installation is the right one for you.

Install guide for Docker Desktop for Windows (be sure to check the requirements)

<https://docs.docker.com/docker-for-windows/install/>

Docker toolbox (for older operating systems, including some versions of Windows 10)

<https://docs.docker.com/toolbox/overview/>

Install guide for Docker Desktop for Mac (be sure to check the requirements)

<https://docs.docker.com/docker-for-mac/install/>

Note 1: Docker toolbox will also install Oracle VM and Git. If you already have these programmes installed in your computer, **uncheck** them in the docker installation wizard.

Note 2: If installing Docker from the toolbox, or Docker Desktop for Windows or Mac, you will install docker compose as well. So you won't need to install Docker-compose separately. More details below.

- 1) If necessary, install docker-compose

The docker-compose install docs are the best place to start

<https://docs.docker.com/compose/install/>

Install Compose on Windows desktop systems

Docker Desktop for Windows and **Docker Toolbox** already include Compose along with other Docker apps, so most Windows users do not need to install Compose separately. Docker install instructions for these are here:

- [Get Docker Desktop for Windows](#)
- [Get Docker Toolbox](#) (for older systems)

If you are running the Docker daemon and client directly on Microsoft Windows Server, follow the instructions in the Windows Server tab.

Docker Compose relies on Docker Engine for any meaningful work, so make sure you have Docker Engine installed either locally or remote, depending on your setup.

- On desktop systems like Docker Desktop for Mac and Windows, Docker Compose is included as part of those desktop installs.
- On Linux systems, first install the Docker for your OS as described on the [Get Docker](#) page, then come back here for instructions on installing Compose on Linux systems.
- To run Compose as a non-root user, see [Manage Docker as a non-root user](#).

Commands

Start your Docker Desktop application if you haven't already (e.g. Docker Desktop for Mac, Docker Toolbox etc.)

If you are able to run this command:

```
docker-compose --help
```

Then your installation is complete

Windows Specific Docker Issue

Relevant Resources

- **Challenges of accessing containers running on windows via localhost:**

<https://docs.docker.com/docker-for-windows/troubleshoot/#limitations-of-windows-containers-for-localhost-and-published-ports>

- **Find IP address with Docker Toolbox**
<https://devilbox.readthedocs.io/en/latest/howto/docker-toolbox/find-docker-toolbox-ip-address.html>
- **What is a TCP port?**
<https://www.bullguard.com/bullguard-security-center/pc-security/computer-security-resources/tcp-ip-ports>

Commands (Windows only)

```
git checkout {Create notebooks commit hash}
```

Make sure you are in the
testing-and-monitoring-ml-deployments/exercise_notebooks/utility
_scripts
directory

Start the command prompt **as an Administrator**

Run the script in this directory:

```
MapPortsForDocker.cmd 5000
```

The script should take about 10 seconds to run.

To undo the script

```
use netsh interface portproxy reset
```

And manually clear hosts file. See:

<https://www.howtogeek.com/howto/27350/beginner-geek-how-to-edit-your-hosts-file/>

Docker Compose Exercise

Relevant Resources

- Redis introduction: <https://redis.io/topics/introduction>
- Redis Python client: <https://github.com/andymccurdy/redis-py>
- Base image used in the exercise Dockerfile (python:3.7-alpine):
<https://github.com/docker-library/python/blob/5f1e3cbcb02c508a5357b5637f3b7a51937e4b5d/3.7/alpine3.10/Dockerfile>

Commands (Windows only)

```
git checkout {Create notebooks commit hash}
```

Navigate to the docker_exercise directory which should be reachable in the following paths:
testing-and-monitoring-ml-deployments\exercise_notebooks\docker_exercise

Make sure you are in the exercise directory, then start the flask and redis containers by running this command:

```
docker-compose up
```

then CTRL+C to stop

You should see the container being built, which may take a couple of minutes. Once this is complete, you should be able to see the web application running by navigating to <http://localhost:5000>

Docker Cheatsheet

Docker:

<https://www.docker.com/sites/default/files/d8/2019-09/docker-cheat-sheet.pdf>

Docker Compose:

<https://devhints.io/docker-compose>

Debugging Docker with logs

<https://hackernoon.com/to-boldly-log-debug-docker-apps-effectively-using-logs-options-tail-and-grep-53d2e655abcb>

Warning: Docker eats disk space like Cookie Monster eats cookies



Everytime you create a new docker image, hundreds of megabytes of disk space is consumed. Manage this disk usage by:

1. Removing all containers (running or not) - this is more to prevent containers fighting over the same TCP ports rather than saving disk space.

Windows (powershell)

```
docker ps -aq | foreach {docker rm $_}
```

OS X / Linux

```
docker rm $(docker ps -a -q)
```

2. Removing docker images - this will free up loads of disk space and is the command you want to use the most, just be aware that docker containers now have to be built from scratch and will take *much* longer to spin up as a result.

View images (and their IDs for use below)

```
docker images
```

Remove one specific image

```
docker rmi {IMAGE_ID}
```

Remove all images (careful, you may want to keep the base images for faster image building)

```
docker rmi $(docker images -q)
```

Advanced Commands (return to this later if you are new to docker-compose)

3. Clean up all unused images, containers, volumes, networks that are dangling (not associated with a container)

```
docker system prune --volumes
```

4. (OS X specific) If you notice docker consuming disk space even after you have removed images, you may need to remove your docker data. See this github issue:
<https://github.com/docker/for-mac/issues/371>

How to remove docker images & volumes:

<https://www.digitalocean.com/community/tutorials/how-to-remove-docker-images-containers-and-volumes>

Troubleshooting This Section

Problem

When double clicking the Docker Quickstart terminal you get an error message saying that docker can't locate the bash.exe file: that is probably because you already had git installed, and need to specify where the git bash is located.

Solution

Check this thread for the solution:

<https://stackoverflow.com/questions/34724712/running-docker-shell-on-windows>

Problem

When executing docker-compose up from docker quickstart terminal the container hangs up and never builds.

Solution

- If you are using legacy Docker Toolbox (for older systems) make sure you first start the Docker daemon with `docker-machine start default`
- Try restarting your machine.
- Check your windows firewall
(<https://stackoverflow.com/questions/42203488/settings-to-windows-firewall-to-allow-docker-for-windows-to-share-drive/43904051>)
- If this does not resolve the issue, work from windows powershell instead.

Problem

Docker toolbox issues:

<https://software.intel.com/en-us/intel-system-studio-docker-install-windows-troubleshoot-docker-toolbox>

Solution

As shown in the video, be sure to check “Install VirtualBox with NDIS5 driver” during Docker Toolbox installation. Restart your machine after installation. You should **not** need to update your host system BIOS if you are using VirtualBox.

6 -Integration Testing ML Systems

Section Additional Background Material

- Open API Specification: <https://swagger.io/specification/>
- Benefits of spec-driven API development:
<https://swagger.io/blog/api-strategy/benefits-of-openapi-api-development/>
- Comprehensive Flask Tutorial:
<https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world>
- Decent Flask/Connexion tutorial:
<https://realpython.com/flask-connexion-rest-api/>
- *Logging*
<https://docs.python.org/3/library/logging.html>
https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html

<https://www.scalyr.com/blog/getting-started-quickly-with-flask-logging/>

Code Overview

Relevant Resources

-
- Configuring Flask: <https://flask.palletsprojects.com/en/1.1.x/config/>
- Connexion docs: <https://github.com/zalando/connexion>
- Makefiles: <https://opensource.com/article/18/8/what-how-makefile>
- Using apt-get: <https://itsfoss.com/apt-get-linux-guide/>

Commands

```
git checkout {Integration Testing Production ML Code - Initial Setup commit hash}
```

Using our Open API Spec

Commands

```
git checkout {Integration Testing Production ML Code - Initial Setup commit hash}
```

Navigate to the ml_api directory:

```
testing-and-monitoring-ml-deployments/packages/ml_api/
```

MacOS / Linux:

```
docker-compose -f docker/docker-compose.yml -d --build
```

Windows (note the backslash):

```
docker-compose -f docker\docker-compose.yml -d --build
```

check the container is up and running with:

```
docker ps
```

Then navigate to <http://localhost:5000/ui> and click on the “Try it out” button.

Once you're done, don't forget to remove your container (see docker section notes)

Integration Testing Theory

Relevant Resources

- Narrow vs. broad integration tests: <https://martinfowler.com/bliki/IntegrationTest.html>

32-bit Operating System Workaround

Test if your operating system is 32 or 64 bit

Windows 7 / 8 / 10

<https://support.microsoft.com/en-gb/help/15056/windows-32-64-bit-faq>

MacOS

<https://apple.stackexchange.com/questions/140938/how-do-i-find-out-of-my-os-x-10-9-4-is-32-bit-or-64-bit>

Linux

<https://www.fastwebhost.in/blog/how-to-find-if-linux-is-running-on-32-bit-or-64-bit/>

Commands

If you do have a 32-bit OS, then you will need to use docker-compose to run the tox commands in this course (from this point onwards in the lecture, where we are installing the `gradient_boosting_model` package).

Make sure you have this commit hash checked out:

Integration Testing Production ML Code - Create Integration Tests

Navigate to

https://github.com/trainindata/testing-and-monitoring-ml-deployments/tree/master/packages/ml_api

Then run:

```
docker-compose -f
docker\workaround_32_os\docker-compose-workaround.yml up --build
```

You should see the tests run in the container. In future lectures, update the tox commands in the docker-compose-workaround.yml accordingly. Note that you will also need to update the Dockerfile.workaround to match the main Dockerfile (e.g. for apt-get updates).

Integration Tests - Code

Relevant Resources

- Mypy config files: https://mypy.readthedocs.io/en/stable/config_file.html
- The Flask app context: <https://flask.palletsprojects.com/en/1.0.x/appcontext/>
- The Flask built-in client: <https://flask.palletsprojects.com/en/1.1.x/testing/>
- Yield keyword recap: <https://realpython.com/introduction-to-python-generators/>
- Pytest custom mark: <https://docs.pytest.org/en/latest/mark.html>
- Pytest config in tox.ini: <http://doc.pytest.org/en/latest/customize.html>

Commands

```
git checkout {Integration Testing Production ML Code - Create Integration Tests commit hash}
```

Navigate to the ml_api directory then run:

```
tox -r
```

or

```
tox -e integration_tests (for just the integration tests)
```

6 Tests Should Pass

Benchmark Tests

Relevant Resources

Important Data Slices:

<https://developers.google.com/machine-learning/testing-debugging/metrics/metrics#check-metrics-for-important-data-slices>

Troubleshooting This Section

Problem

If you see this error when you run the docker commands:

```
Error response from daemon: Get
https://registry-1.docker.io/v2/: net/http: request canceled
while waiting for connection (Client.Timeout exceeded while
awaiting headers)
```

Then you are likely experiencing this issue:

<https://stackoverflow.com/questions/41262622/error-connecting-to-docker-hub>

Solution

And you should restart the docker machine(Docker Toolbox):

```
docker-machine restart default
```

Problem

When you run a tox command, if you see the warning:

```
ERROR: tox config file (either pyproject.toml, tox.ini,
setup.cfg) not found
```

Solution

You are not in the correct directory. Navigate to where the tox.ini file is located and run the command again.

Problem

When running the tox command, it is stuck at the installing requirements for > 5 mins

Solution

Give it *one* gentle CTRL+C tap.

Problem

If you tried running the docker container and got:

```
E: Release file for
http://security.debian.org/debian-security/dists/buster/updates/
InRelease is not valid yet (invalid for another 11h 15min 28s).
Updates for this repository will not be applied.
E: Release file for
http://deb.debian.org/debian/dists/buster-updates/InRelease is
not valid yet (invalid for another 10h 59min 39s). Updates for
this repository will not be applied.
ERROR: Service 'ml_api' failed to build: The command '/bin/sh -c
apt-get update -y && apt-get install -y make && apt-get
install -y libffi-dev gcc && apt-get install -y curl'
returned a non-zero code: 100
```

Solution

Restart your computer (Yeah I know, but it actually works).

7 - Differential Testing

Differential Testing Implementation

Relevant Resources

- *.dockerignore file docs*
<https://docs.docker.com/engine/reference/builder/#dockerignore-file>
- *Dockerignore primer*
<https://codefresh.io/docker-tutorial/not-ignore-dockerignore/>
- *Math isclose*
<https://docs.python.org/3/library/math.html#math.isclose>

Commands

```
git checkout {Advanced Testing Production ML Code - Create Differential Tests commit hash}
```

From the `ml_api` directory (rebuilds the environment, so will take a few mins to run):

```
tox -e differential_tests -r
```

You should see **1 test pass**.

8 - Shadow Mode

Section Additional Background Material

- Overview of Shadow Deployments
<https://christophergs.github.io/machine%20learning/2019/03/30/deploying-machine-learning-applications-in-shadow-mode/>
- Cindy Sridharan's excellent "Testing In Production" blog series (highly recommended)
<https://medium.com/@copyconstruct/testing-in-production-the-safe-way-18ca102d0ef1>
- On Feature Flags: <https://martinfowler.com/articles/feature-toggles.html>

Tests in Shadow Deployments

Relevant Resources

- On the Kolmogorow Smirnov Test:
<https://towardsdatascience.com/kolmogorov-smirnov-test-84c92fb4158d>
- The Kruskal-Wallis Test:
<http://www.biostathandbook.com/kruskalwallis.html>
- Chi-squared Test:
<http://www.biostathandbook.com/chigof.html>

Shadow Mode Code Overview - DB Setup

Relevant Resources

- SQLAlchemy docs: <https://docs.sqlalchemy.org/en/13/>
- SQLAlchemy session: <https://docs.sqlalchemy.org/en/13/orm/session.html>
- Introduction to SQLAlchemy in Python:
<https://towardsdatascience.com/sql-and-etl-an-introduction-to-sqlalchemy-in-python-fc66e8be1cd4>
- SQL vs ORM discussion on Hacker News:
<https://news.ycombinator.com/item?id=16809620>
- Postgres JSON types: <https://www.postgresql.org/docs/12/datatype-json.html>
- Volumes in Docker:
<https://blog.container-solutions.com/understanding-volumes-docker>
- Alembic docs: <https://alembic.sqlalchemy.org/en/latest/>
- Introduction to using Alembic:
<https://www.compose.com/articles/schema-migrations-with-alembic-python-and-postgresql/>
- Autogenerating migrations:
<https://alembic.sqlalchemy.org/en/latest/tutorial.html#create-a-migration-script>

e.g. `PYTHONPATH=. alembic revision --autogenerate -m "create prediction tables"`

Note the autogenerate flag

Commands

```
git checkout {Shadow Mode ML Code - Initial Setup commit hash}
```

Setup Tests for Shadow Mode

Relevant Resources

- Python mock docs: <https://docs.python.org/3/library/unittest.mock.html>
- Understanding mocking: <https://realpython.com/python-mock-library/>
- Mocks vs. MagicMock:
<https://medium.com/ryans-dev-notes/python-mock-and-magicmock-b3295c2cc7eb>

Commands

```
git checkout {Shadow Mode ML Code - Implementation and Tests commit hash}
```

Start your Docker Desktop application if you haven't already (e.g. Docker Desktop for Mac, Docker Toolbox etc.)

From the `ml_api` directory run:

Windows

Make sure you have run the port mapping script. Navigate to this directory:

```
testing-and-monitoring-ml-deployments\exercise_notebooks\utility_scripts>
```

Then run **as an Administrator**:

```
MapPortsForDocker.cmd 6608
```

Then

```
MapPortsForDocker.cmd 6609
```

Then navigate back to the `ml_api` directory and run (note the `docker-compose.test.yml`):

```
docker-compose -f docker\docker-compose.test.yml up -d --build  
test_database
```

OS X / Linux

```
docker-compose -f docker/docker-compose.test.yml up -d --build  
test_database
```

Check the DB container is running with `docker ps`

With your test docker database container running (but not the API), then run:

```
tox -r
```

You should see all differential, integration, unit and style/type checks pass.

Shadow Mode - Asynchronous Implementation

Relevant Resources

- Threading in Python <https://docs.python.org/3/library/threading.html>
- *Note that here we show a relatively simple asynchronous implementation within flask. For more complex asynchronous jobs, you would setup a distributed task queue such as celery (which also supports scheduling): <http://www.celeryproject.org/>*

Commands

```
git checkout {Shadow Mode ML Code - Asynchronous Implementation commit hash}
```

From the ml_api directory run (note the docker-compose.test.yml):

```
docker-compose -f docker\docker-compose.test.yml up -d --build  
test_database
```

OS X / Linux

```
docker-compose -f docker/docker-compose.test.yml up -d --build  
test_database
```

Check the DB container is running with `docker ps`

With your test docker database container running (but not the API), then run:

```
tox
```

Shadow Mode - Populate DB Script

Commands

```
git checkout {Shadow Mode ML Code - Populate DB Script commit hash}
```

Windows ONLY

Make sure you have run the port mapping script. Navigate to this directory:

```
testing-and-monitoring-ml-deployments\exercise_notebooks\utility_scripts>
```

Then run as an **Administrator**:

```
MapPortsForDocker.cmd 6608
```

Then

```
MapPortsForDocker.cmd 6609
```

From the ml_api directory run:

```
docker-compose -f docker\docker-compose.yml up -d --build
```

Then run:

```
tox -e generate_predictions
```

Open the jupyter notebook here (make sure you keep your DB container running):

[Testing and monitoring ML deployments from Train in Data Github](#)

Assess Shadow Model

Commands

```
git checkout {Shadow Mode ML Code - Analyse Results commit hash}
```

Make sure you've completed all the commands from the previous lecture, then navigate to:
testing-and-monitoring-ml-deployments\exercise_notebooks\shadow_mode_exercise

Make sure you have installed the requirements in the requirements.txt in this directory. Then start the jupyter notebook and open: assessing_model_results.ipynb

Troubleshooting This Section

Problem

You see this error when you run the docker commands:

```
Error: pg_config executable not found.
```

```
pg_config is required to build psycopg2 from source. Please  
add the directory
```

containing pg_config to the \$PATH or specify the full executable path with the option:

```
python setup.py build_ext --pg-config /path/to/pg_config
build ...
```

or with the pg_config option in 'setup.cfg'.

If you prefer to avoid building psycopg2 from source, please install the PyPI 'psycopg2-binary' package instead.

For further information please check the 'doc/src/install.rst' file (also at <http://initd.org/psycopg/docs/install.html>).

Solution

You ran the docker command without specifying just the test_database container.

- *Remove any running docker containers*
- *Run the following docker-compose command to start the test database only*

Windows

```
docker-compose -f docker\docker-compose.test.yml up --build -d
test_database
```

OS X/Linux

```
docker-compose -f docker/docker-compose.test.yml up --build -d
test_database
```

Problem

Tox hangs

Solution

If it's stuck longer than 5 mins (it's normal for tox to take a few mins to create its virtualenv when you pass the -r flag or run a new command for the first time), then tap **CTRL+C ONCE**

Other Gotchas

<https://stackoverflow.com/questions/56657683/postgres-docker-image-is-not-creating-database-with-custom-name>

Problem

```
ERROR: for docker_test_database_1  Cannot start service
test_database: network
0818221c820e61df155ec2b11986ecdac144945464dede589ec2c5f18150adf5
not found
```

ERROR: Encountered errors while bringing up the project.

Solution

Add the force recreate flag:

<https://stackoverflow.com/questions/53347951/docker-network-not-found>

If your API container is unable to connect to the DB, there may be an image mismatch:

- 1) Stop and remove all containers
- 2) Remove all images
- 3) Rerun the commands to start the API and DB containers

Problem

```
raise ConnectionError(err, request=request)

requests.exceptions.ConnectionError: ('Connection aborted.',
ConnectionResetError(10054, 'An existing connection was forcibly
closed by the remote host', None, 10054, None))
```

Solution

Remove and restart docker containers

Problem

Only the DB container starts

Solution

Rerun the docker command (and check the docker-compose -f referenced file is correct).

9 - Monitoring Metrics with Prometheus

Section Additional Background Material

- *Prometheus Up and Running* by Brian Brazil:
<https://www.oreilly.com/library/view/prometheus-up/9781492034131/>
- Brian Brazil's blog is also a great source of information on Prometheus:
<https://www.robustperception.io/blog>
- *Distributed Systems Observability* by Cindy Sridharan:
<https://www.oreilly.com/library/view/distributed-systems-observability/9781492033431/>

- “Monitoring and Observability”, by Cindy Sridharan:
<https://medium.com/@copyconstruct/monitoring-and-observability-8417d1952e1c>
- “Metrics vs. Logs” by Cindy Sridharan:
<https://medium.com/@copyconstruct/logs-and-metrics-6d34d3026e38>
- *Machine Learning Logistics* by Ted Dunning & Ellen Friedman:
<https://mapr.com/ebook/machine-learning-logistics/assets/machine-learning-logistics.pdf>
- RED metrics:
<https://www.weave.works/blog/the-red-method-key-metrics-for-microservices-architecture/>

Metrics for ML Systems

Relevant Resources

- *On Cardinality:*
<https://logz.io/blog/cardinality-challenge-in-monitoring/>
<https://www.robustperception.io/cardinality-is-key>

Prometheus & Grafana Overview

Relevant Resources

- Prometheus docs: <https://prometheus.io/docs/introduction/overview/>
- Grafana docs: <https://grafana.com/docs/grafana/latest/>
- PromQL docs: <https://prometheus.io/docs/prometheus/latest/querying/basics/>
- PromQL Introduction:
<https://medium.com/@valyala/promql-tutorial-for-beginners-9ab455142085>
- Alerting in Prometheus: <https://prometheus.io/docs/alerting/overview/>

Basic Prometheus Setup

Relevant Resources

- Unicorn docs: <https://docs.gunicorn.org/en/stable/>
- WSGI servers: <https://www.fullstackpython.com/wsgi-servers.html>

- Flask dispatcher middleware: <https://flask.palletsprojects.com/en/1.1.x/patterns/appdispatch/#combining-applications>
- Prometheus config docs: <https://prometheus.io/docs/prometheus/latest/configuration/configuration/>
- Prometheus Python client: https://github.com/prometheus/client_python

Commands

```
git checkout {Monitoring with Prometheus - Basic Setup commit hash}
```

Windows ONLY

Start your Docker Desktop application if you haven't already (Docker Toolbox etc.)

Make sure you have run the port mapping script. Navigate to this directory:

```
testing-and-monitoring-ml-deployments\exercise_notebooks\utility_scripts>
```

Then run **as an Administrator**:

```
MapPortsForDocker.cmd 9090
```

Then

```
MapPortsForDocker.cmd 3000
```

Then

```
MapPortsForDocker.cmd 5601
```

Navigate to:

```
testing-and-monitoring-ml-deployments/exercise_notebooks/prometheus_exercise/
```

Run

```
docker-compose up -d
```

When you run `docker ps` you should see both the prometheus and webapp containers running.

Open your browser at: <http://localhost:9090>

You should see the Prometheus expression browser.

Adding Metrics

Relevant Resources

- Prometheus Metric Types: https://prometheus.io/docs/concepts/metric_types/
- Metric naming and labelling: <https://prometheus.io/docs/practices/naming/>

- Rate docs: <https://prometheus.io/docs/prometheus/latest/querying/functions/#rate>
- Rate gotchas: <https://www.robustperception.io/rate-then-sum-never-sum-then-rate>

Commands

```
git checkout {Monitoring with Prometheus - Add Simple Metrics commit hash}
```

Navigate to:

testing-and-monitoring-ml-deployments/exercise_notebooks/prometheus_exercise/

Start your Docker Desktop application if you haven't already (e.g. Docker Desktop for Mac, Docker Toolbox etc.)

Run

```
docker-compose up -d --build
```

Adding Grafana

Relevant Resources

- Grafana docs: <https://grafana.com/docs/grafana/latest/>
- Grafana support for Prometheus: <https://prometheus.io/docs/visualization/grafana/>

Commands

```
git checkout {Monitoring with Prometheus - Setup Grafana commit hash}
```

Navigate to:

testing-and-monitoring-ml-deployments/exercise_notebooks/prometheus_exercise/

Run

```
docker-compose up -d --build
```

When you run `docker ps` you should see 3 containers:

- webapp
- Prometheus
- Grafana

Check you can navigate to the Grafana UI by navigating to <http://localhost:3000>

Username: admin

Password: foobar

Infrastructure Metrics

Relevant Resources

- cadvisor docs: <https://github.com/google/cadvisor/tree/master/docs>
- Prometheus exporters: <https://prometheus.io/docs/instrumenting/exporters/>

Commands

```
git checkout {Monitoring with Prometheus - Basic Infrastructure Metrics commit hash}
```

Navigate to:

```
testing-and-monitoring-ml-deployments/exercise_notebooks/prometheus_exercise/
```

Run

```
docker-compose up -d --build
```

You should see 4 containers running when you check with `docker ps`

Adding Metrics Monitoring to Our Example Project

Relevant Resources

- cadvisor docs: <https://github.com/google/cadvisor/tree/master/docs>

Commands

```
git checkout {Monitoring with Prometheus - Instrument Project API commit hash}
```

Navigate to:

```
testing-and-monitoring-ml-deployments/packages/ml_api
```

Remove any running/stopped containers:

Windows (powershell)

```
docker ps -aq | foreach {docker rm $_ -f}
```

MacOS / Linux

```
docker rm $(docker ps -a -q) -f
```

Then run:

```
docker-compose -f docker/docker-compose.yml up -d --build
```

You should see 5 containers running.

Then run:

```
tox -e generate_predictions
```

Login to Grafana at <http://localhost:3000>

Username: Admin

Password: foobar

Creating an ML System Grafana Dashboard

Relevant Resources

- Prometheus aggregation over time:
https://prometheus.io/docs/prometheus/latest/querying/functions/#aggregation_over_time
- Tox arguments:
<https://tox.readthedocs.io/en/latest/example/general.html#interactively-passing-positional-arguments>
- ML Dashboard table panel query and explanation:

```
count by(live_model, live_version, shadow_model, shadow_version, version)(model_version_details_info* on (instance, job) group_left(version) python_info)
```

by specifies the labels to keep
group_left lets you specify that there can be multiple matching samples in the group of the left-hand operand
- Standard error of the mean (SEM):
<https://www.khanacademy.org/math/ap-statistics/sampling-distribution-ap/sampling-distribution-on-mean/v/standard-error-of-the-mean>
- Standard Score: https://en.wikipedia.org/wiki/Standard_score
- Gitlab anomaly detection with Prometheus:
<https://about.gitlab.com/blog/2019/07/23/anomaly-detection-using-prometheus/>
- Prometheus alerting overview: <https://prometheus.io/docs/alerting/overview/>
- Setting up the alert manager: <https://daenney.github.io/2018/04/21/setting-up-alertmanager>

Commands

```
git checkout {Monitoring with Prometheus - Instrument Project API commit hash}
```

Navigate to:

```
testing-and-monitoring-ml-deployments/packages/ml_api
```

Remove any running/stopped containers:

Windows (powershell)

```
docker ps -aq | foreach {docker rm $_ -f}
```

MacOS / Linux

```
docker rm $(docker ps -a -q) -f
```

Then run (switch forward slash to backslash for Windows):

```
docker-compose -f docker/docker-compose.yml up -d --build
```

You should see 5 containers running.

Then run:

```
tox -e generate_predictions
```

Login to Grafana at <http://localhost:3000>

Username: Admin

Password: foobar

To generate an anomaly, run:

```
tox -e generate_predictions -- --anomaly True
```

Troubleshooting This Section

Problem

When starting the prometheus container you see:

```
ERROR: for prometheus  Cannot create container for service
prometheus: Conflict. The container name "/prometheus" is
already in use by container
"f6e27c1f23f6656808c30613f233d8cd0d56a2a6f9cd5b633d2224ca8874a8
b". You have to remove (or rename) that container to be able to
reuse that name.
ERROR: Encountered errors while bringing up the project.
```

Solution

Remove all your docker containers, there is a name conflict (see docker cheatsheet in this guide)

Problem

No data is sent to Grafana, running the “up” command in the Prometheus expression browser yields no results.

Solution

- Remove all containers
- Rebuild and restart
- Wait 2-3 mins
- If that fails, delete the images and then repeat the above steps

Problem

Grafana cannot find data

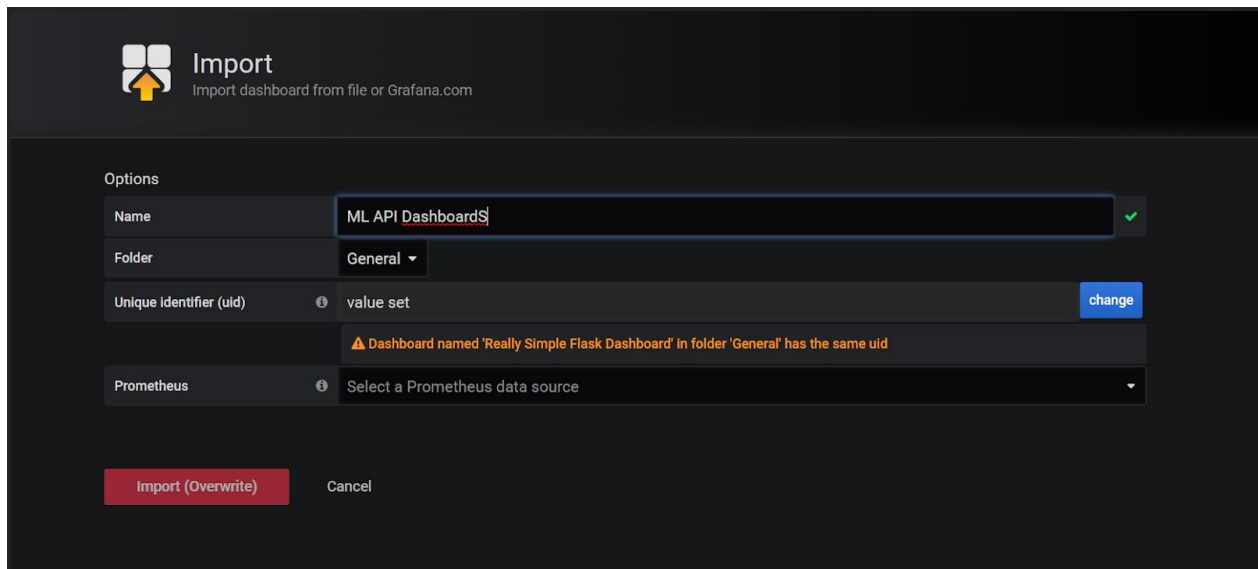
Solution

<https://community.grafana.com/t/grafana-could-not-find-data-source-for-prometheus/2881/3>

Problem

When you got import the ML API dashboard you see:

Dashboard named 'Really Simple Flask Dashboard' in folder 'General' has the same uid



The image shows the Grafana 'Import' interface. At the top, there's a logo and the text 'Import dashboard from file or Grafana.com'. Below this, there's a section titled 'Options' with several input fields: 'Name' (containing 'ML API DashboardS' with a green checkmark), 'Folder' (set to 'General'), 'Unique identifier (uid)' (set to 'value set' with a 'change' button), and 'Prometheus' (a dropdown menu showing 'Select a Prometheus data source'). A warning message states: 'Dashboard named 'Really Simple Flask Dashboard' in folder 'General' has the same uid'. At the bottom, there are two buttons: 'Import (Overwrite)' and 'Cancel'.

Select Prometheus from the “Select a Prometheus data source” bar, then click “Import (overwrite)”

Problem

Docker just hangs when you run a command:

```

Select Command Prompt - docker-compose -f docker\docker-compose.yml up -d --build
Microsoft Windows [Version 10.0.18362.657]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\chris>cd repos
C:\Users\chris\repos>cd testing-and-monitoring-ml-deployments
C:\Users\chris\repos\testing-and-monitoring-ml-deployments>cd packages
C:\Users\chris\repos\testing-and-monitoring-ml-deployments\packages>cd ml_api
C:\Users\chris\repos\testing-and-monitoring-ml-deployments\packages\ml_api>docker-compose -f docker\docker-compose.yml up -d --build

```

Solution

run `docker-machine start default`

Problem

When you run docker-machine start default you get:
(default) Waiting for an IP...
And it just hangs there

Solution

If when Docker Toolbox starts (running as administrator) you get “waiting for IP”, it’s worth giving in a couple of minutes as per:
<https://github.com/docker/toolbox/issues/457>

Problem

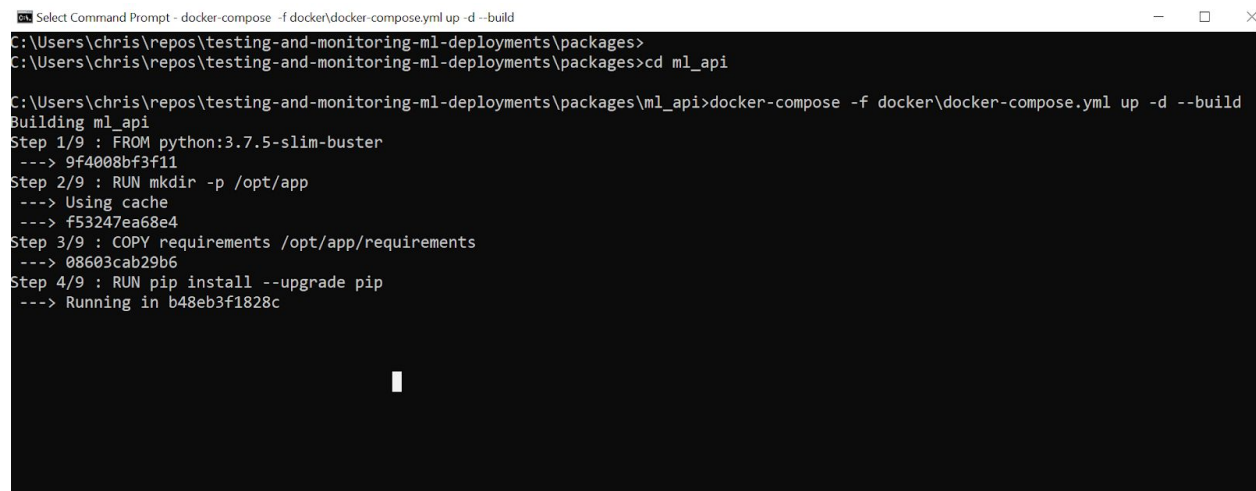
When starting Docker Toolbox:

```
(default) Check network to re-create if needed...
(default) Windows might ask for the permission to configure a dhcp
server. Sometimes, such confirmation window is minimized in the
taskbar.
Error setting up host only network on machine start: C:\Program
Files\Oracle\VirtualBox\VBoxManage.exe modifyvm default --nic2
hostonly --nictype2 82540EM --nicpromisc2 deny --hostonlyadapter2
VirtualBox Host-Only Ethernet Adapter #4 --cableconnected2 on failed:
VBoxManage.exe: error: Code E_FAIL (0x80004005) - Unspecified error
(extended info not available)
VBoxManage.exe: error: Context: "LockMachine(a->session,
LockType_Write)" at line 529 of file VBoxManageModifyVM.cpp
Looks like something went wrong in step 'Checking status on
default'... Press any key to continue...
```

Solution

Restart your computer

Docker Compose up command hangs half way through



```
Select Command Prompt - docker-compose -f docker\docker-compose.yml up -d --build
C:\Users\chris\repos\testing-and-monitoring-ml-deployments\packages>
C:\Users\chris\repos\testing-and-monitoring-ml-deployments\packages>cd ml_api
C:\Users\chris\repos\testing-and-monitoring-ml-deployments\packages\ml_api>docker-compose -f docker\docker-compose.yml up -d --build
Building ml_api
Step 1/9 : FROM python:3.7.5-slim-buster
--> 9f4008bf3f11
Step 2/9 : RUN mkdir -p /opt/app
--> Using cache
--> f53247ea68e4
Step 3/9 : COPY requirements /opt/app/requirements
--> 08603cab29b6
Step 4/9 : RUN pip install --upgrade pip
--> Running in b48eb3f1828c
```

Solution

Solution tap CTRL+C **ONCE**

Running Prometheus In Production - Further Reading

On Multiprocessing

Prometheus client libraries presume a threaded model, where metrics are shared across workers. This doesn't work so well for languages such as Python where it's common to have processes rather than threads to handle large workloads, docs:

https://github.com/prometheus/client_python#multiprocess-mode-gunicorn

Deploying a Prometheus Server & Managed Prometheus Options:

<https://www.digitalocean.com/community/tutorials/how-to-install-prometheus-on-ubuntu-16-04>

<https://linuxacademy.com/blog/kubernetes/running-prometheus-on-kubernetes/>

Hosted options: <https://www.metricfire.com/>

10 - Monitoring Logs with Kibana

Additional Background Material

- *Learning Kibana 7: Build powerful Elastic dashboards with Kibana's data visualization capabilities, 2nd Edition* by [Anurag Srivastava](#) and [Bahaaldine Azarmi](#)
- What is the ELK stack? <https://www.elastic.co/what-is/elk-stack>
- Elastic Stack documentation: <https://www.elastic.co/guide/index.html>
- The complete guide to the Elastic Stack: <https://logz.io/learn/complete-guide-elk-stack/>

Elastic Stack Overview

Relevant Resources

- Logstash input plugins:
<https://www.elastic.co/guide/en/logstash/current/input-plugins.html>
- Logstash filter plugins:
<https://www.elastic.co/guide/en/logstash/current/filter-plugins.html>
- Logstash output plugins:
<https://www.elastic.co/guide/en/logstash/current/output-plugins.html>
- Elastic Search Index:
<https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started-index.html>
- Elastic Stack scalability:
<https://www.elastic.co/guide/en/elasticsearch/reference/current/scalability.html>
- Beats: <https://www.elastic.co/beats>
- Elastic Search query DSL:
<https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html>
- Querying in Kibana:
<https://www.elastic.co/guide/en/beats/packetbeat/current/kibana-queries-filters.html>
- Alerting: <https://www.elastic.co/what-is/elasticsearch-alerting>

Kibana Hands-On Exercise

Relevant Resources

- Unicorn configuration: <https://docs.gunicorn.org/en/stable/settings.html>
- Configuring logstash:
<https://www.elastic.co/guide/en/logstash/current/configuration.html>
- Python3 Logstash: <https://github.com/israel-fl/python3-logstash>
- Python JSON logger: <https://github.com/madzak/python-json-logger>
- Configuring logging in Python:
<https://docs.python.org/3/howto/logging.html#configuring-logging>
- Werkzeug docs: <https://werkzeug.palletsprojects.com/en/1.0.x/>

Commands

```
git checkout {Monitoring Logs with the Elastic Stack - Basic ELK Setup commit hash}
```

Navigate to:

testing-and-monitoring-ml-deployments/exercise_notebooks/elk_exercise

Start your Docker Desktop application if you haven't already (e.g. Docker Desktop for Mac, Docker Toolbox etc.)

Remove any running/stopped containers:

Windows (powershell)

```
docker ps -aq | foreach {docker rm $_ -f}
```

MacOS / Linux

```
docker rm $(docker ps -a -q) -f
```

Then run (this will take a few mins):

MacOS/Linux

```
ELK_VERSION=7.5.1 docker-compose up -d
```

Windows

```
set ELK_VERSION=7.5.1
```

Then

```
docker-compose up -d
```

You should see 4 containers running:

- elk_exercise_webapp
- logstash
- elastic search
- kibana

Navigate to <http://localhost:5000> and refresh the page a couple of times. Then visit kibana at <http://localhost:5601>

Username: elastic

Password: changeme

Integrating Kibana into the Example Project

Relevant Resources

- Logging fileConfig:
<https://docs.python.org/3/library/logging.config.html#logging.config.fileConfig>

Commands

```
git checkout {Monitoring Logs with the Elastic Stack - Integrate with API commit hash}
```

Navigate to:

testing-and-monitoring-ml-deployments/packages/ml_api

Start your Docker Desktop application if you haven't already (e.g. Docker Desktop for Mac, Docker Toolbox etc.)

Remove any running/stopped containers:

Windows (powershell)

```
docker ps -aq | foreach {docker rm $_ -f}
```

MacOS / Linux

```
docker rm $(docker ps -a -q) -f
```

Then run:

MacOS/Linux

```
ELK_VERSION=7.5.1 docker-compose -f  
docker/docker-compose-elk.yml up -d --build
```

Windows

```
set ELK_VERSION=7.5.1
```

Then

```
docker-compose -f docker\docker-compose-elk.yml up -d --build
```

You should see 5 containers running:

- docker_ml_api
- postgres
- logstash
- elastic search
- kibana

Navigate to <http://localhost:5000> and refresh the page a couple of times. Then visit kibana at <http://localhost:5601>

Username: elastic

Password: changeme

Setting Up a Kibana Dashboard for Model Inputs

Relevant Resources

- Kibana Saved Objects UI:
<https://www.elastic.co/guide/en/kibana/current/managing-saved-objects.html>
- Kibana define index guide:
<https://www.elastic.co/guide/en/kibana/current/tutorial-define-index.html>
- Visualizing your data in Kibana:
<https://www.elastic.co/guide/en/kibana/current/tutorial-visualizing.html>
- Alerting with Elastic Search: <https://www.elastic.co/what-is/elasticsearch-alerting>

Commands

```
git checkout {Monitoring Logs with the Elastic Stack - Create Kibana  
Dashboard for Model Inputs commit hash}
```

Navigate to:

```
testing-and-monitoring-ml-deployments/packages/ml_api
```

Start your Docker Desktop application if you haven't already (e.g. Docker Desktop for Mac, Docker Toolbox etc.)

Remove any running/stopped containers:

Windows (powershell)

```
docker ps -aq | foreach {docker rm $_ -f}
```

MacOS / Linux

```
docker rm $(docker ps -a -q) -f
```

Then run:

MacOS/Linux

```
ELK_VERSION=7.5.1 docker-compose -f  
docker/docker-compose-elk.yml up -d --build
```

Windows

```
set ELK_VERSION=7.5.1
```

Then

```
docker-compose -f docker\docker-compose-elk.yml up -d --build
```

You should see 5 containers running:

- docker_ml_api
- postgres
- logstash
- elastic search
- kibana

```
tox -e generate_predictions
```

Navigate to <http://localhost:5000> and refresh the page a couple of times. Then visit kibana at <http://localhost:5601>

Username: elastic

Password: changeme

Troubleshooting This Section

Problem

Can't open <http://localhost:5000> even though all the containers are running

Solution

Have you run the port mapping script on port 5601?

Problem

Kibana can't find my index

Solution

Once all the containers have been running for a minute or two, navigate to <http://localhost:5000> and refresh the page a few times to send some data to Kibana, then try searching for the index again.

Problem

'ELK_VERSION' is not recognized as an internal or external command,
operable program or batch file.

Solution

You need to set the ELK_VERSION:

Windows

```
set ELK_VERSION=7.5.1
```

Running The Elastic Stack In Production - Further Reading

Deploying The Elastic Stack & Managed Prometheus Options:

<https://www.elastic.co/guide/en/cloud/current/ec-create-deployment.html>

Hosted options: <https://logz.io/>