MDA DevOps for Data Science

MDA

2024-05-21

Table of contents

Preface

The MDA team at CORI will do a book club and review the book DevOps for Data Science written by **Alex K Gold** here.

- Book study repo
- Book study site
- Book study google drive

Planning:

1. Introduction + Environments as Code: Olivier
2. Data Project Architecture: John
3. Databases and Data APIs: Brittany
4. Logging and Monitoring: Dolley
5. Deployments and Code Promotion: Camden
6. Demystifying Docker: John
7. ...

Workflow:

1. Clone this repo
2. Create a specific branch:

```
git checkout -b ch-?/???
# (replace `?` with the chapter number and `???` with a name/title)
```

3. Install Quarto CLI
4. Install local dependencies:

```
npm install
```

6. Preview Quarto site:

```
npm run preview
```

7. Make changes and commit as needed
8. Push commits and create a pull request

This is a Quarto book.

To learn more about Quarto books visit https://quarto.org/docs/books.

Introduction

Learning objectives:

- introducing some definitions and a bit of history

Definitions

production :

> affecting decision in your orgs / world
>
> putting your work in front of someone else's eyes

We want:

- our works to be **reliable**
- in a safe **environment**
- our work to be **available**

DevOps

> DevOps is a set of **cultural norms**, **practices**, and **tooling** to help make developing and deploying software **smoother** and **lower risk**.

.. but is a squishy concept and a "vendor" associated name

It came in opposition to the *waterfall dev process* were you had a team doing Dev. and then one other doing Ops (Ops "make it works on everyone computer").

Process and people

- Are data scientist software developper?
- Are we in the red flags number 3?
- Do we need a workbench (ie using ec2) ?
- should we do the exercice with `penguins` or one of our dataset?

Part I

DevOps Lessons for Data Science

You are a software developer.

But:

- Writing code for data science is different than writing code:

  You're pointed at some data and asked to derive value from it without even knowing if that's possible.

- difference between architect and archaeologist

1 Environments as Code

1.1 Environments:

- stack of software and hardware below our code
- should be treated as “cattle not pet” / should be *stateless*
- Risk of it “only works on my machine”

Building a completly reproducible environement is a “fool’s errand” but first step should be easy.

(any trouble with renv and sf anyone?)

1.2 Environements have layers

Layer	Contents	Example
Packages	R packages	cori.db
System	R versions / GDAL / MacOS	14.4.1
hardware	Physical / Virtual hardware	Apple M3

Hardware and System should be in the hand of IT (see later chapter 7 and 14), packages layer should be the data scientist.

1.3 The package layer

Package can in 3 places:

- repository: CRAN / GH / “Supermarket”
- library: a folder on a drive / “pantry”
- loaded : “ready to cook”

Each **project** should have it's own "pantry"

Project was higlighted in text but I think it is important: if you do not have a project workflow it is way harder to do it.

A package environement shouldbe :

- isolated and cannot be disrupted (example updating a packge in an other project)
- can be "captured" and "transported"

In R: `{Renv}` ("light"/"not exactly the same" option also exist, Box, capsule)

Author does not like Conda (good to not being alone!)

1.4 Workflow

- Create a standalone directory with a virtual environment

(spend time exploring `renv/` and `.gitignore`)

- Document environment state (see `lockfile`)
- Collaborate / deploy: you can't share package because their binay can be OS or system specific, hence specific package need to be installed (could be a pain point).
- Use virtual env

1.5 Under the hood

- test `.libpaths()` in a specific project and in a "random" R session
- order of Paths matter

1.6 Key points

- being in production is what make a DS a software developper
- kill and create new environment fast is important

2 Data Project Architecture

2.1 Initial project setup

The last chapter, Environments as Code, introduced the example project that we will use throughout the book. You can either ~~clone a starter template for~~ fork the project repo from do4ds_project or create the project from scratch yourself using the following Quarto CLI commands (taken from the Quarto documentation):

```
quarto create project website do4ds_project
# Choose (don't open) when prompted

quarto preview do4ds_project
```

… if the quarto preview command loads a new website in your web browser, go back to the terminal and use `Ctrl+C` to terminate the preview server. Change to the project directory and setup a local python virtual environment (if needed, you can grab the `requirements.txt` file from here):

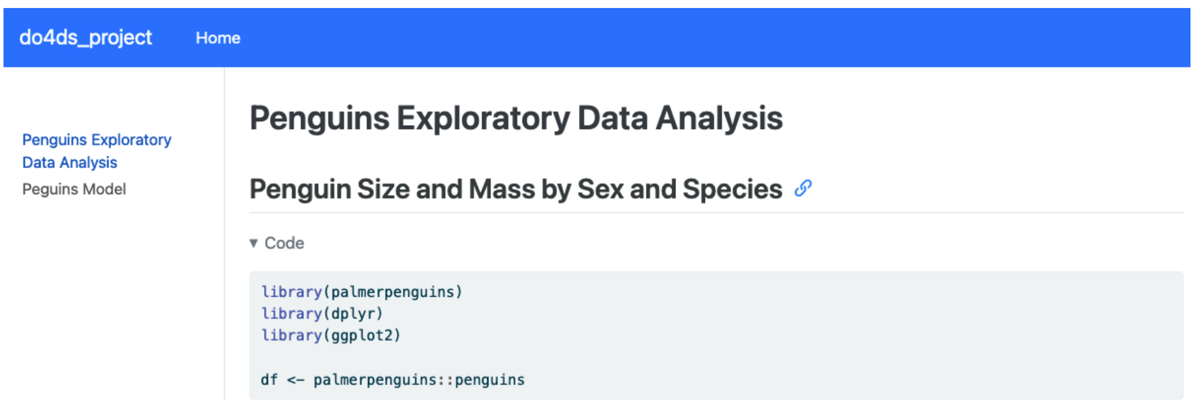
```
cd do4ds_project
# If using python, create and activate a local virtual environment
python -m venv ./venv
source venv/bin/activate
venv/bin/python -m pip install -r requirements.txt
```

Now that you are in the local project directory you can use the quarto `preview` command without arguments to continue seeing updates to the local project in your browser:

```
quarto preview

# Alternately, if you forked the project sample from Github, you can use npm...
npm run preview
```

If you did not fork the project sample, make sure to create the `eda.qmd` and `model.qmd` files from chapter 1 and add them to the sidebar section of `_quarto.yml`:

```
project:
  type: website

website:
  title: "do4ds_project"
  navbar:
    left:
      - href: index.qmd
        text: Home
  sidebar:
    style: "docked"
    search: true
    contents:
      - eda.qmd
      - model.qmd
```



2.2 Key Takeaways

This chapter gives an opiniated overview of good design and conceptual layout practices in regards to a data project. The areas of responsibility within the project our broken out into 1) *Presentation*, 2) *Processing*, and 3) *Data* layers. The categories that a given data project may fall into our further divided into 1) *jobs*, 2) *apps*, 3) *reports* and 4) *API's*. The rest of the chapter discusses how to break a project down into the previously mentioned layers, as well as considerations for optimizing the Processing and Data layers.

2.3 Lab

To complete part 1 of the lab, I had to modify the example code. First, I added a line that would generate a `vetiver` model and assign it to `v` and then I changed the path to the local folder where the model could be stored:

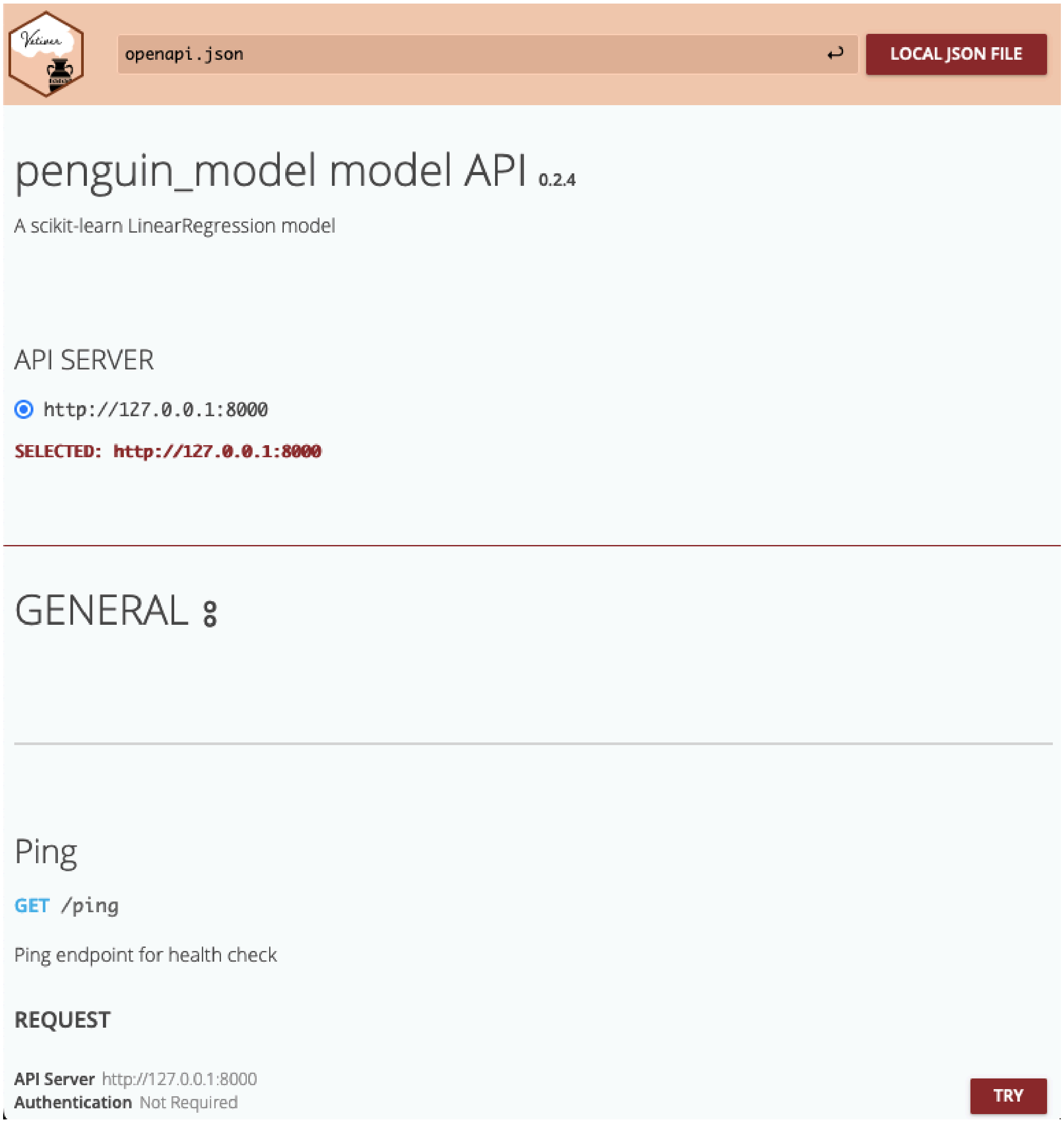
```
from pins import board_folder
from vetiver import vetiver_pin_write
from vetiver import VetiverModel

v = VetiverModel(model, model_name = "penguin_model")

model_board = board_folder(
  "data/model",
  allow_pickle_read = True
)
vetiver_pin_write(model_board, v)
```

In addition to these changes, I created a separate Python file with the code to run the `vetiver` API, called `api.py`, which also required updates to the `VetiverApi` call to ensure that the API server had the correct input params in order to process the prediction:

```
from palmerpenguins import penguins
from pandas import get_dummies
from sklearn.linear_model import LinearRegression
from pins import board_folder
from vetiver import VetiverModel
from vetiver import VetiverAPI

# This is how you would reload the model from disk...
b = board_folder('data/model', allow_pickle_read = True)
v = VetiverModel.from_pin(b, 'penguin_model')

# ... however VertiverAPI also uses the model inputs to define params from the prototype
df = penguins.load_penguins().dropna()
df.head(3)
X = get_dummies(df[['bill_length_mm', 'species', 'sex']], drop_first = True)
y = df['body_mass_g']

model = LinearRegression().fit(X, y)

v = VetiverModel(model, model_name = "penguin_model", prototype_data = X)
```

```python
app = VetiverAPI(v, check_prototype = True)
app.run(port = 8000)
```

… and then used `python api.py` to run the API. Once running, you can navigate to http://127.0.0.1:8000/**docs** in a web browser to see the autogenerated API documentation



3

4

5

6

Part II

IT/Admin for Data Science

7

8

9

10

11

12

13

14

Part III

Enterprise-grade Data Science

15

16

17

18

A

B

C

D