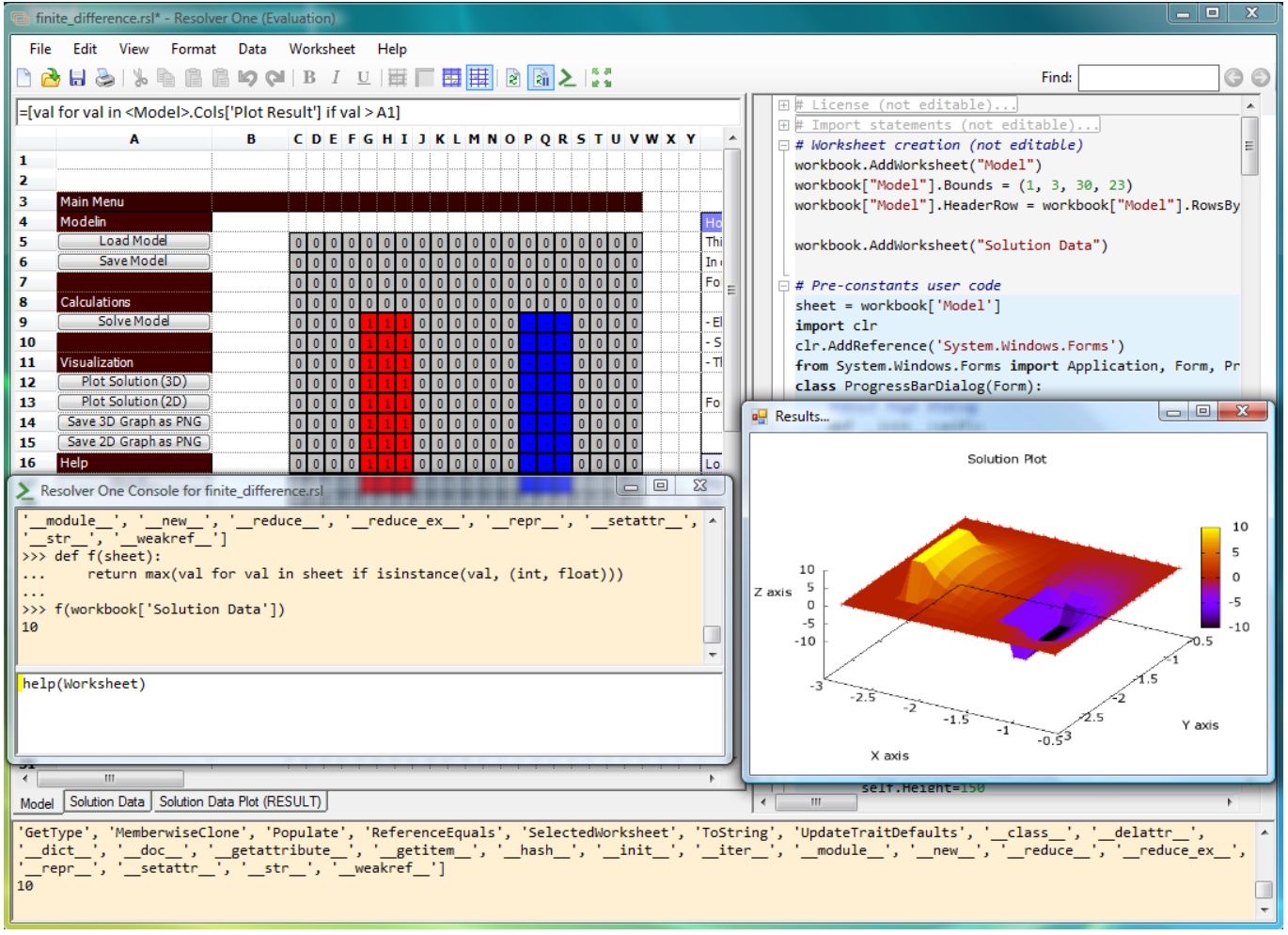
Michael Foord
https://agileabstractions.com/

- Python trainer & contractor
- Python developer since 2002
- Core Python Developer
- Author of IronPython in Action
- Creator of “unittest.mock”
- Twitter: @voidspace

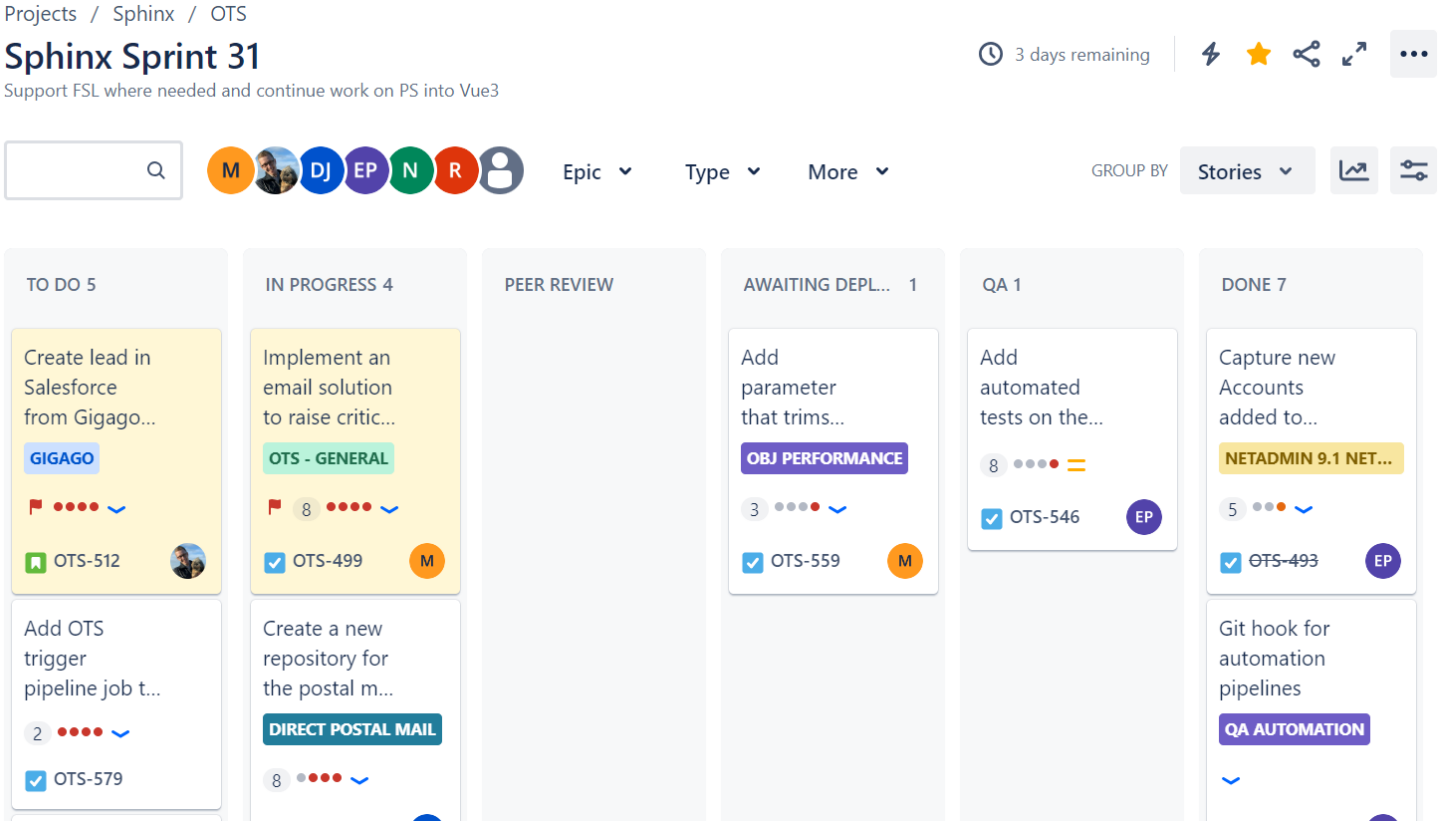


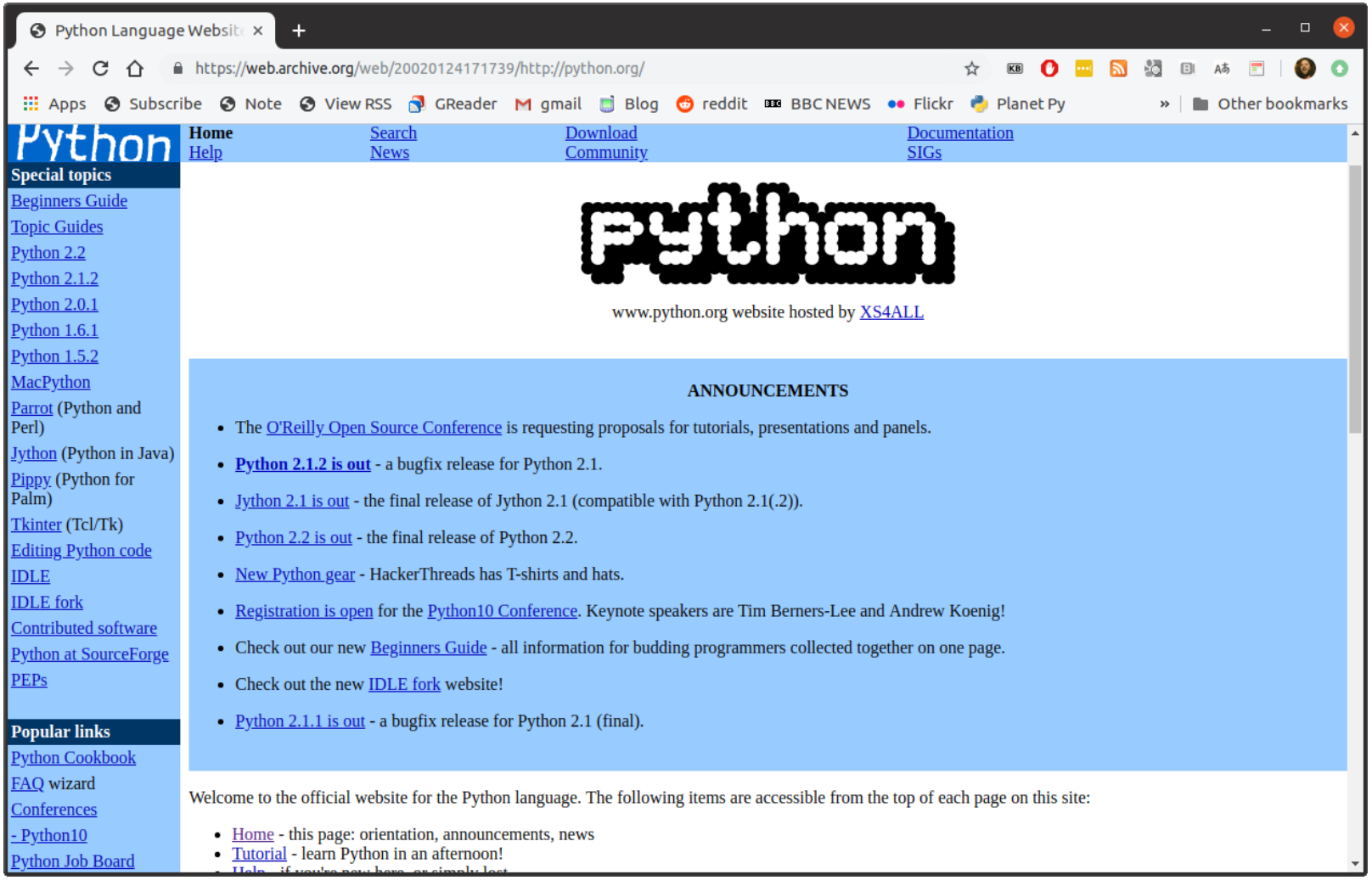
Agile, DevOps and Scrum, along with the tooling and technologies of modern engineering enable us to tackle the difficult problems of building and running robust systems with confidence. – Michael Foord

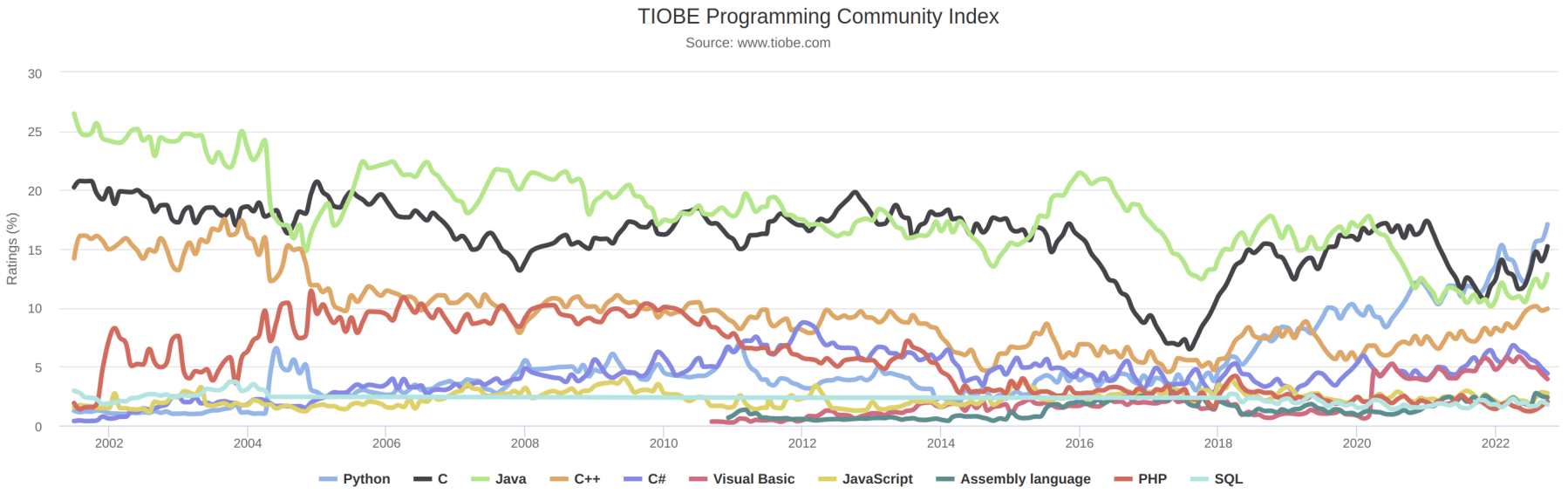
Some Core Values from Agile

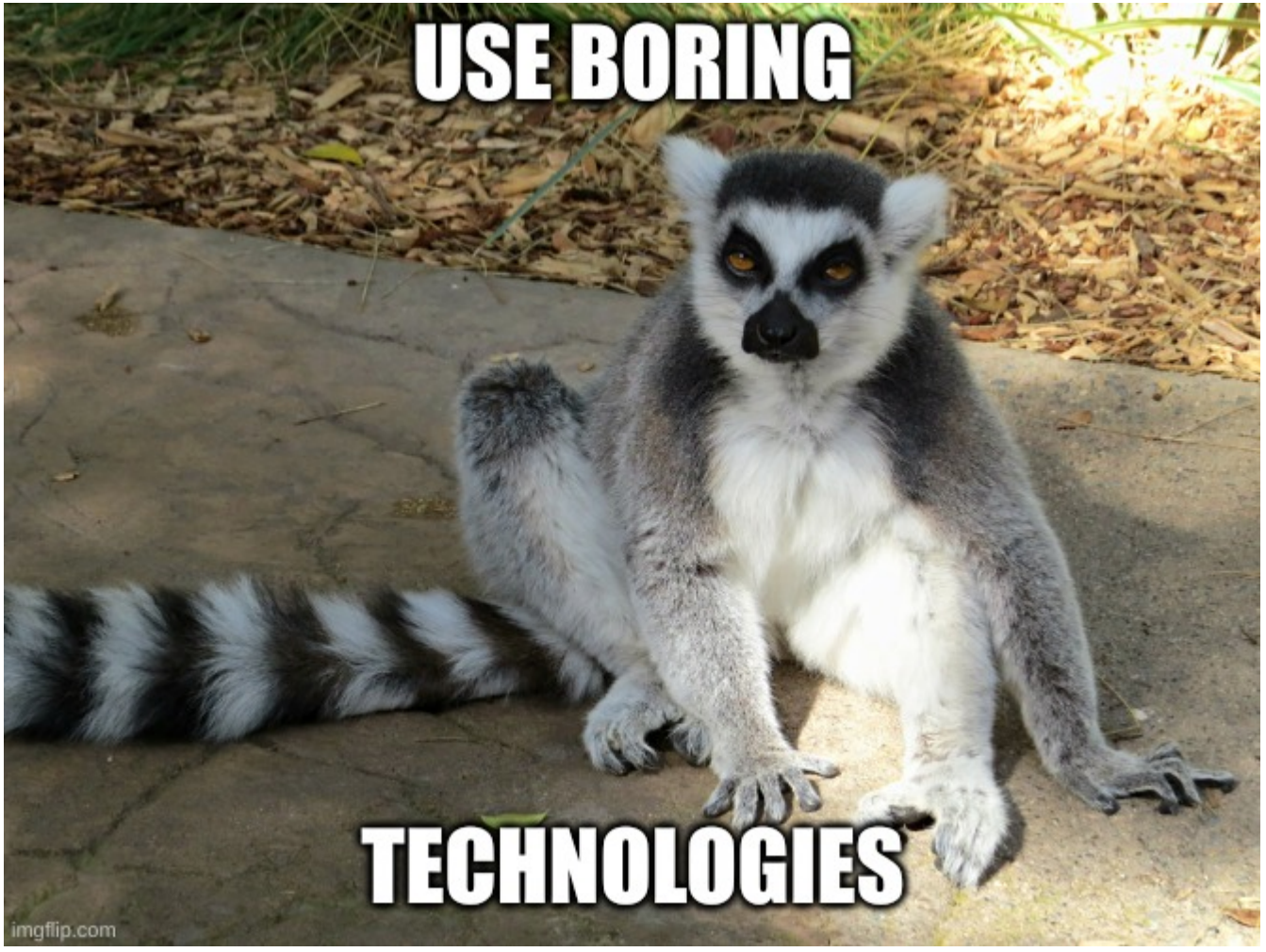
- Testing: quality and confidence
- MVP, do the simplest thing, grow organically and refactor
- Tooling: version control, Kanban, issue tracking
- Iterations: able to change direction (agile)
- Estimation and prioritisation

Kanban with Jira









Python is Boring Technology

Version Control

- A backup
- Always know which is the latest version
- Be able to go back to earlier versions
- Merge changes from several sources

hynek.micro.blog

Hynek's Brief Thoughts

About Python Package Metadata

Feb 14, 2020

Since this topic keeps coming up, I'd like to briefly share my thoughts on Python package metadata because it's – as always – more complex than it seems.

When I say metadata I mean mostly the version so I will talk abou it interchangeably. But the description, the license, or the project URL are also part of the game.

The overarching problem is that we have two places where we may need that metadata:

1. In the packaging mechanism. *setuptools*, *pip*, *flit*, *poetry*, et all

A quick-and-dirty guide on how to install packages for Python

BRETT CANNON
21 JAN 2020 • 5 MIN READ

When people start learning Python, they often will come across a package they want to try and it will usually start with "just `pip install` it!" The problem with that advice is it's a very simplistic view of how to manage packages and can actually lead to problems down the road. And while there is a tutorial on installing packages at packaging.python.org, it might be a bit intimidating for some if they are just looking to quickly get up and going.

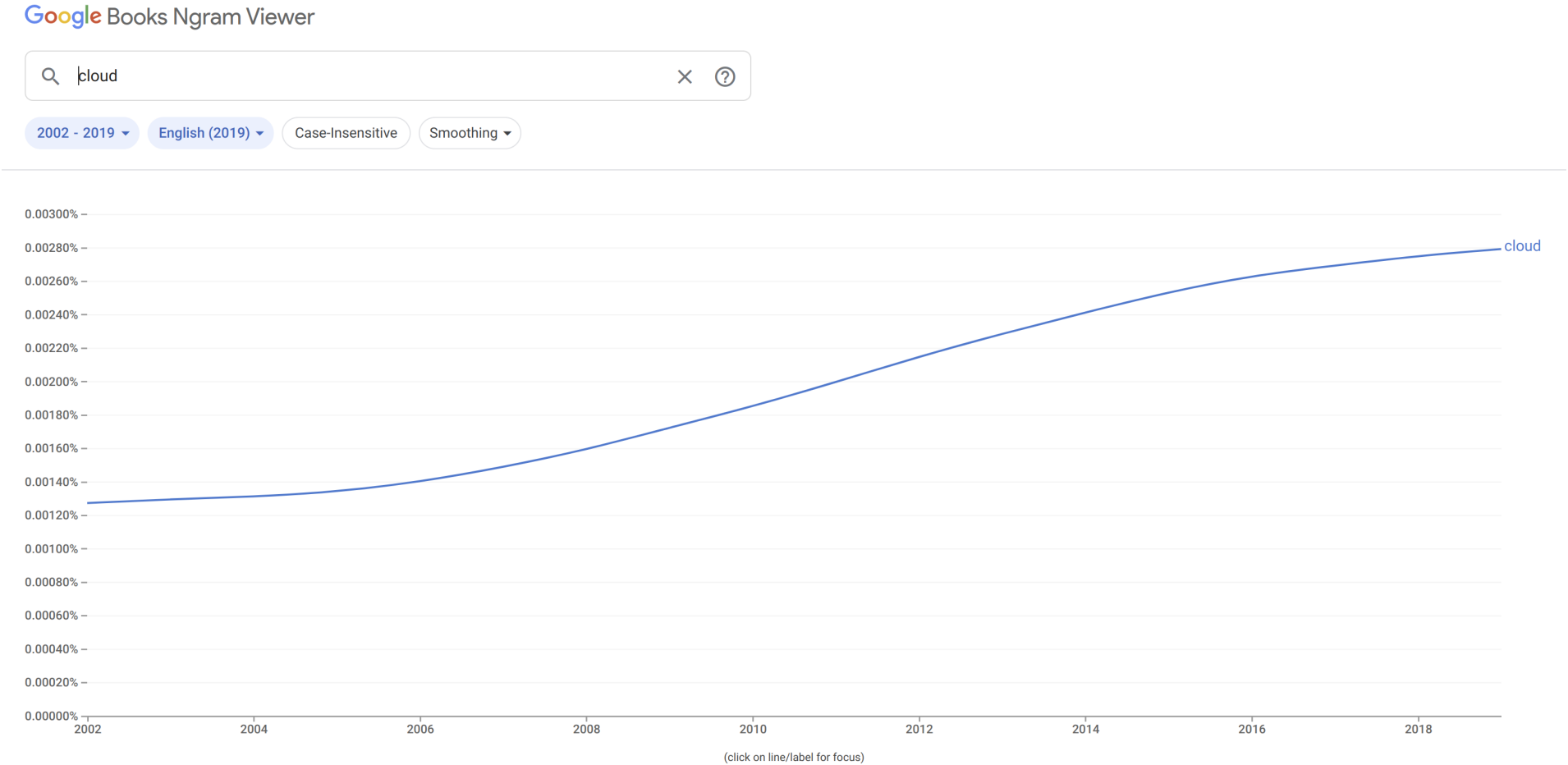
If you just want to start poking at Python and want to avoid the pitfalls to installing packages globally, it only takes 3 steps to do the right thing.

Summary

1. Create a virtual environment, e.g. `python3.8 -m venv .venv` (substitute `py -3.8` for `python3.8` if necessary)
2. Activate the virtual environment, e.g. `source .venv/bin/activate.fish` (assuming you are using the fish shell)
3. Install the packages you want, e.g. `python -m pip install --upgrade pip`

Dependency Management

```
git clone https://github.com/voidspace/…
cd repository
python -m venv .env
.env\Scripts\activate
pip install -r requirements.txt
```

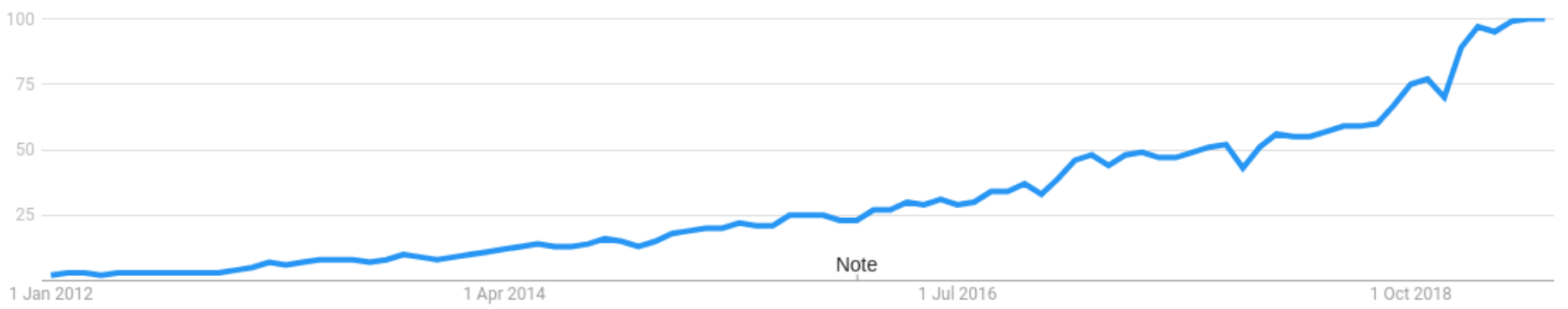


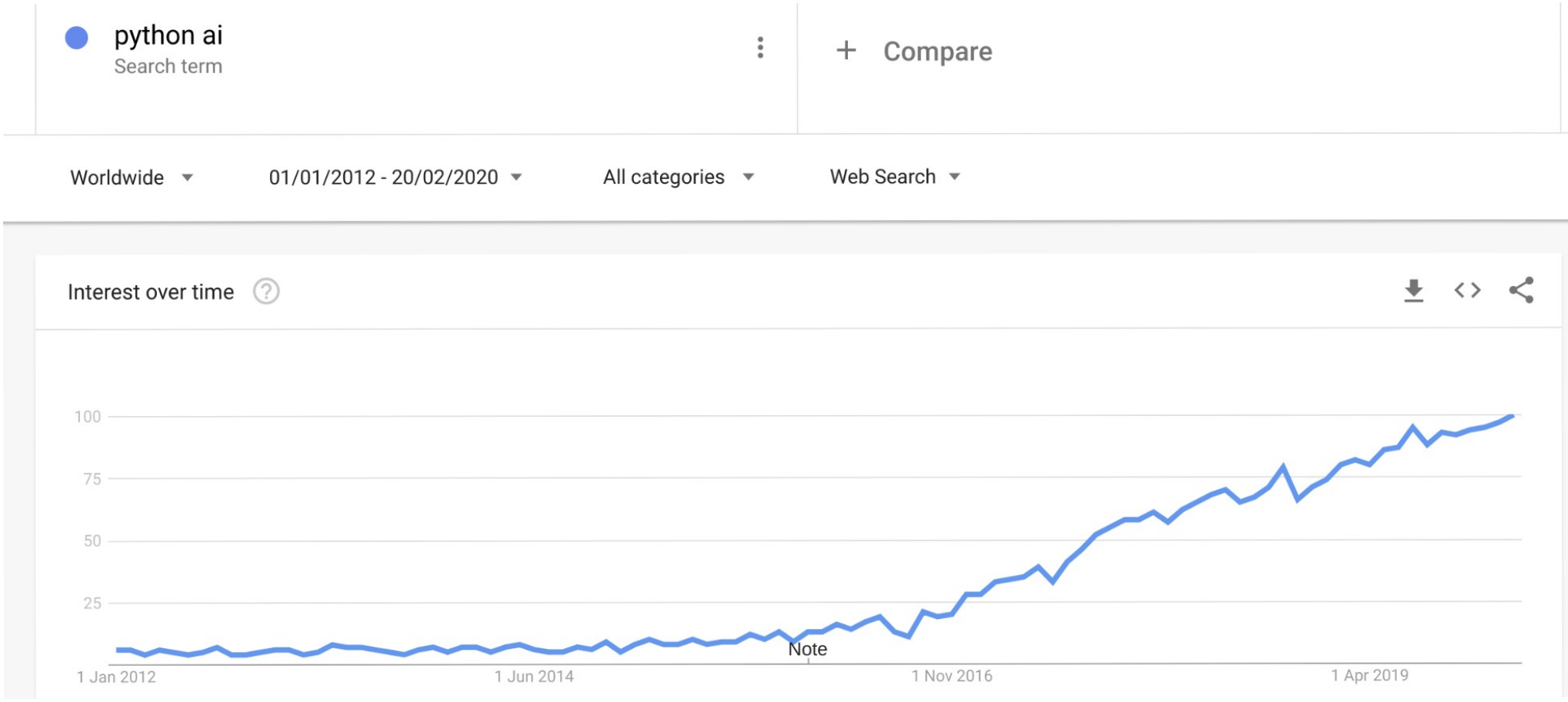
devops
Search term

+ Compare

Worldwide | 01/01/2012 - 21/06/2019 | All categories | Web Search

Interest over time

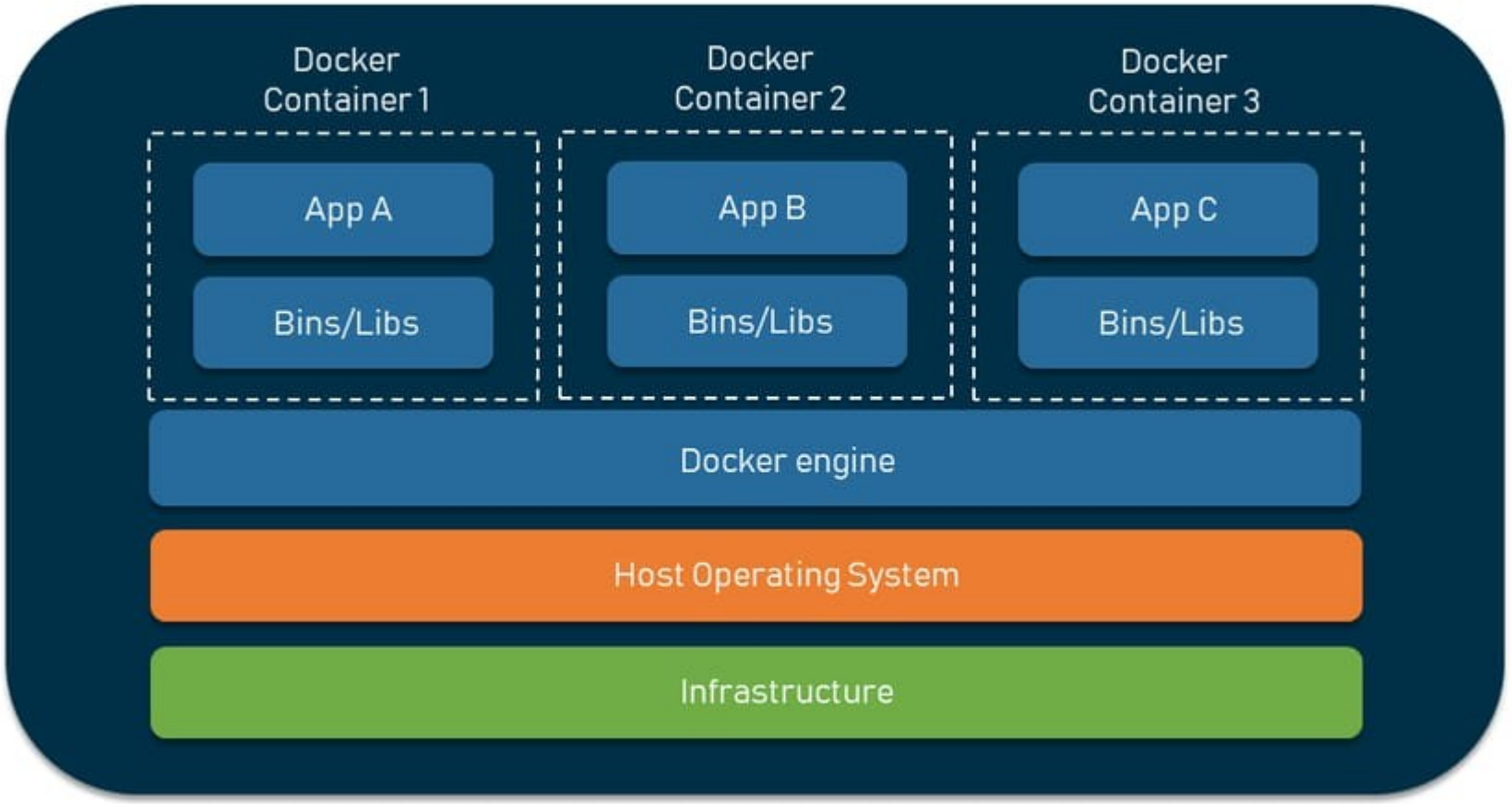




The Cloud and DevOps

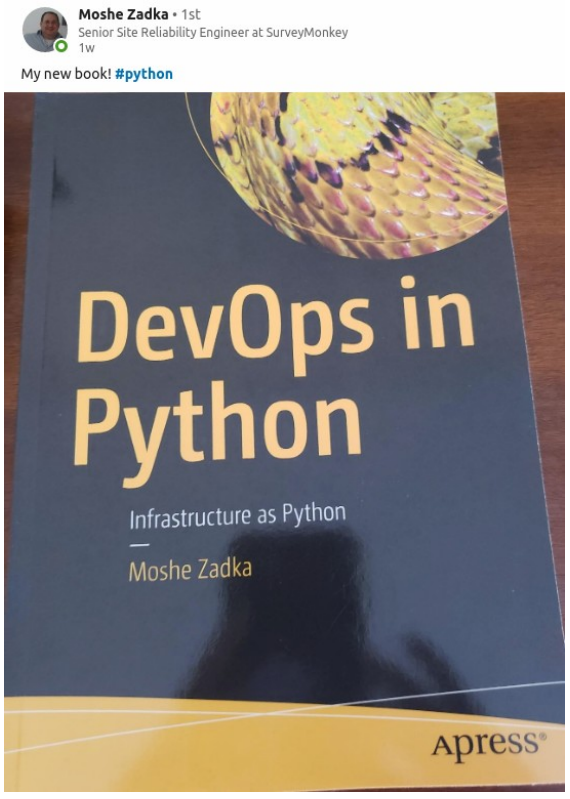
- The cloud enabled devops
- Containers as the unit of deployment
- Repeatable automated builds and deployment
- Dev build artefact similar to prod environment
- Docker and Docker images! (and then kubernetes)

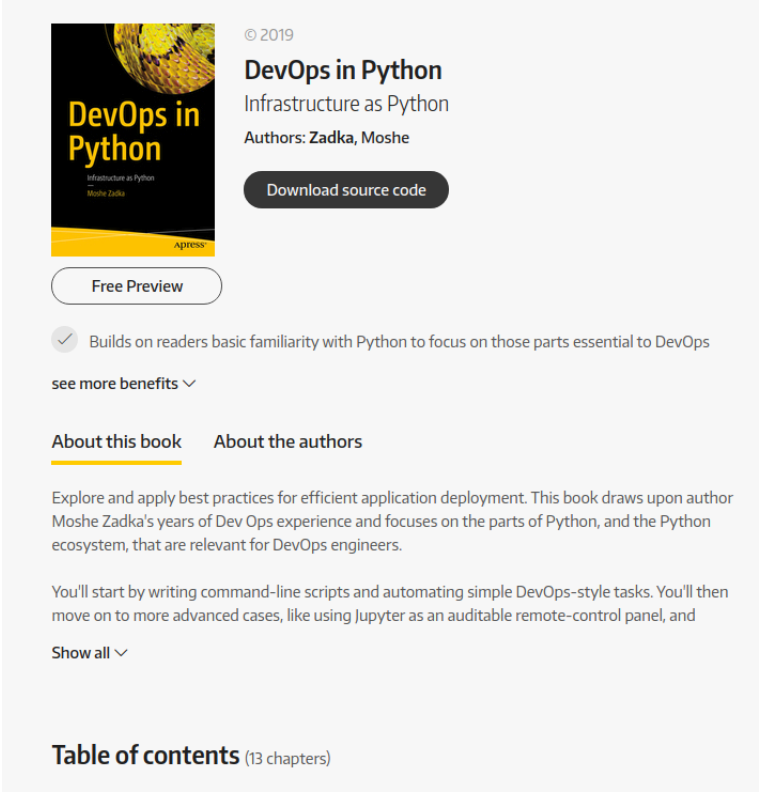
DOCKER CONTAINERS



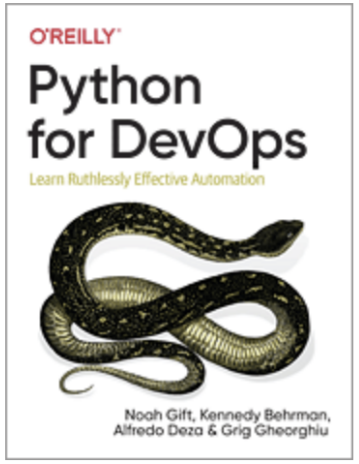
DevOps

- Running services as well as building them
- Automated workflows
  - Automated testing: deploy with confidence
  - Push button deploy and rollback
  - Continuous Integration/Continuous Deployment
- Caring for the system for the whole lifecycle
- Tooling like docker, kubernetes, ansible, infrastructure as code





Disposable virtual environments



Python for DevOps

Learn Ruthlessly Effective Automation

By Grig Gheorghiu, Noah Gift, Kennedy Behrman, Alfredo Deza

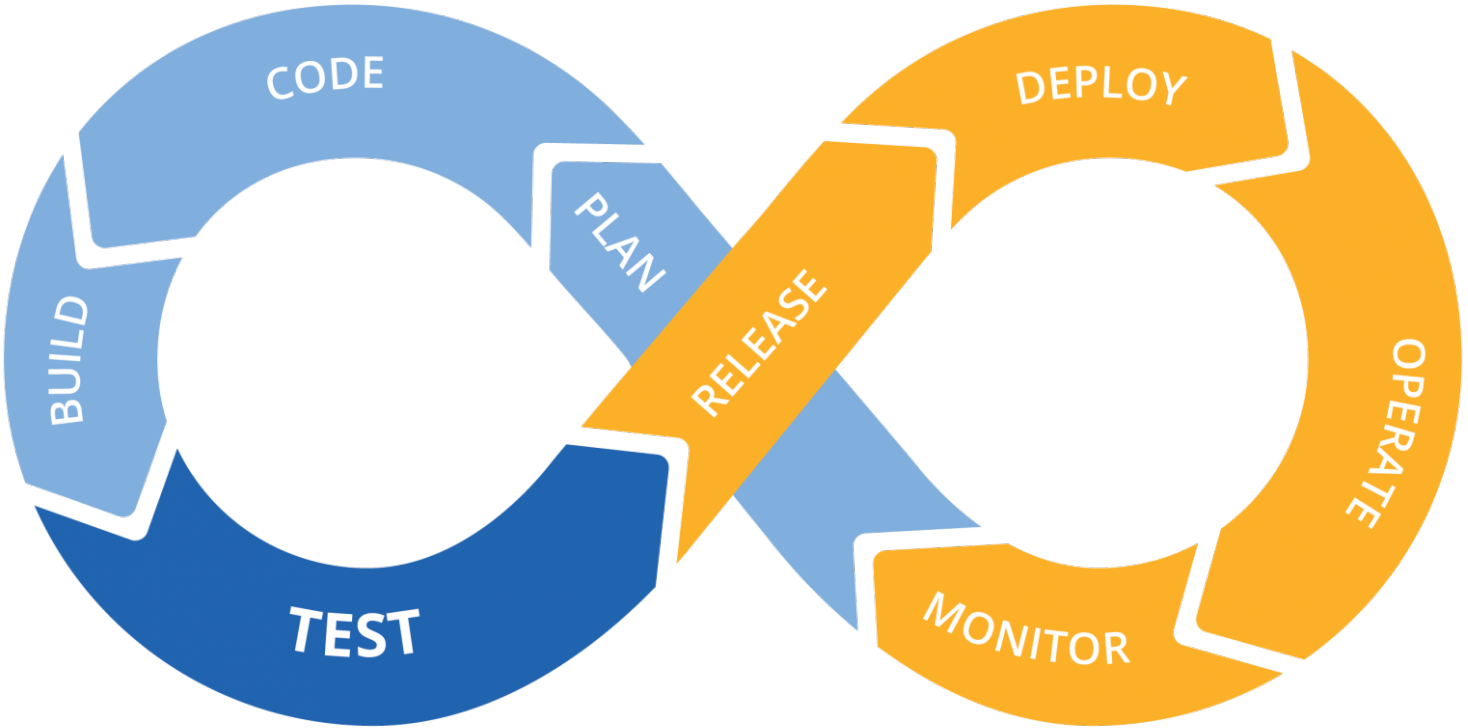
Publisher: O'Reilly Media
Release Date: December 2019
Pages: 506

Much has changed in technology over the past decade. Data is hot, the cloud is ubiquitous, and many organizations need some form of automation. Throughout these transformations, Python has become one of the most popular languages in the world. This practical resource shows you how to use Python for everyday Linux systems administration tasks with today's most useful DevOps tools, including Docker, Kubernetes, and Terraform.

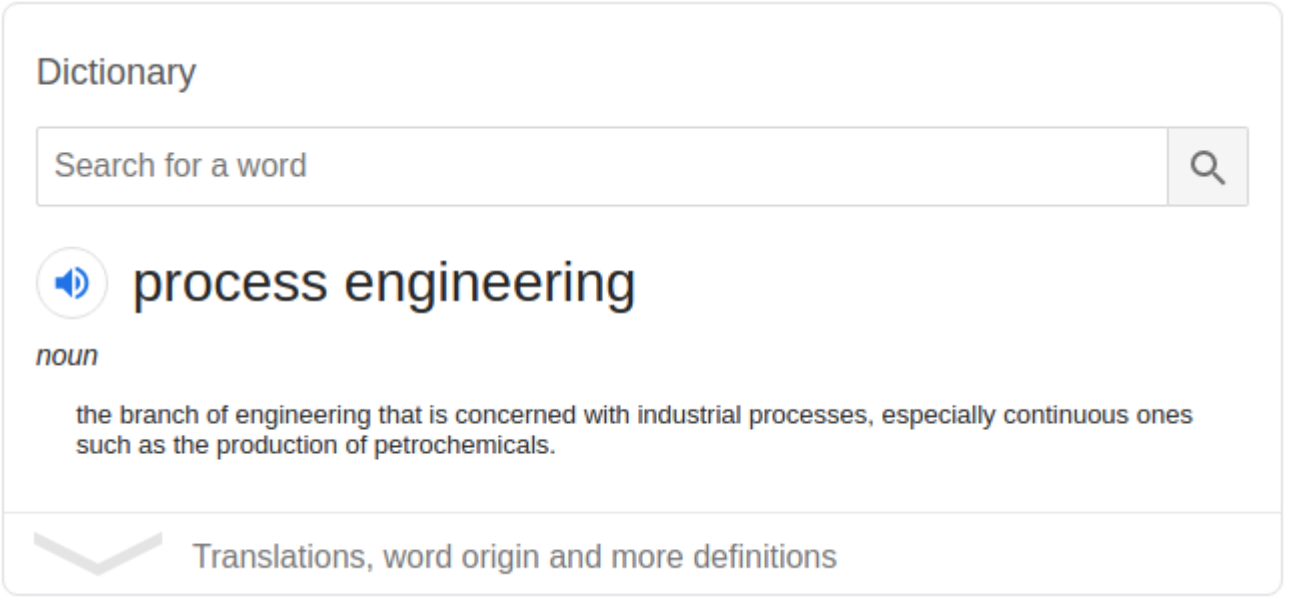
Learning how to interact and automate with Linux is essential for millions of professionals. Python makes it much easier. With this book, you'll learn how to develop software and solve problems using containers, as well as how to monitor, instrument, load-test, and operationalize your software. Looking for effective ways to "get stuff done" in Python? This is your guide.

Great interviews with prominent Python developers

CI/CD



Government Digital Services, Singapore



The deliberate design of processes and workflows, supported by tooling, to enable the continuous and sustained delivery and maintenance of a quality system.

The modern Python ecosystem and infrastructure, along with conventions and understanding from the wider programming community, means that the “how” of software engineering is often a solved problem. What and why still remain to be solved...

Agile, DevOps and Scrum, along with the tooling and technologies of modern engineering enable us to tackle the difficult problems of building and running robust systems with confidence –
Michael Foord

sys.exit(0)