PARTNERS & SPONSORS

MADISON PUBLIC LIBRARY

VERVE
A ROCKWELL AUTOMATION COMPANY

David Webber, PhD
MadPy Meetup
Sept 12, 2024

Outline

- Introduce myself
- What is DevOps?
- A simple get/set module
- Webify
- Containerize
- Deploy
- Monitor
- Summary

Follow Along on GitHub

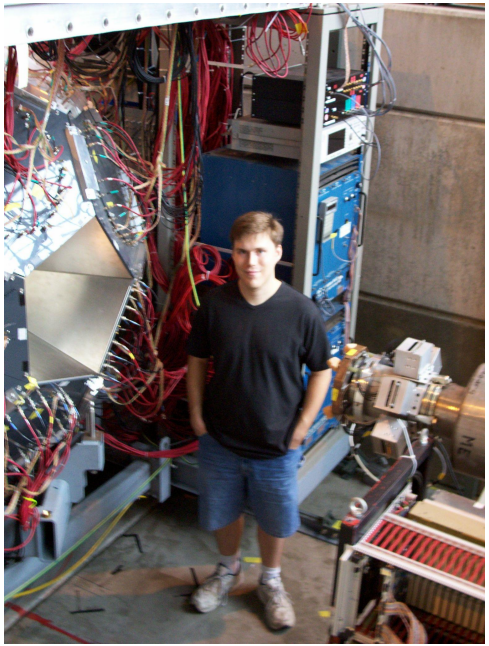
https://github.com/davidwebber/devops_for_python

Physicist turned Entrepreneur

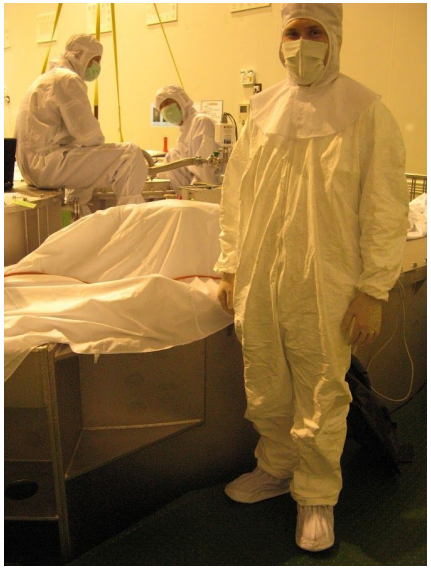
Mini CV

- Muon lifetime
- Reactor neutrino θ_{13}
- IoT Startup

MuLan



DayaBay

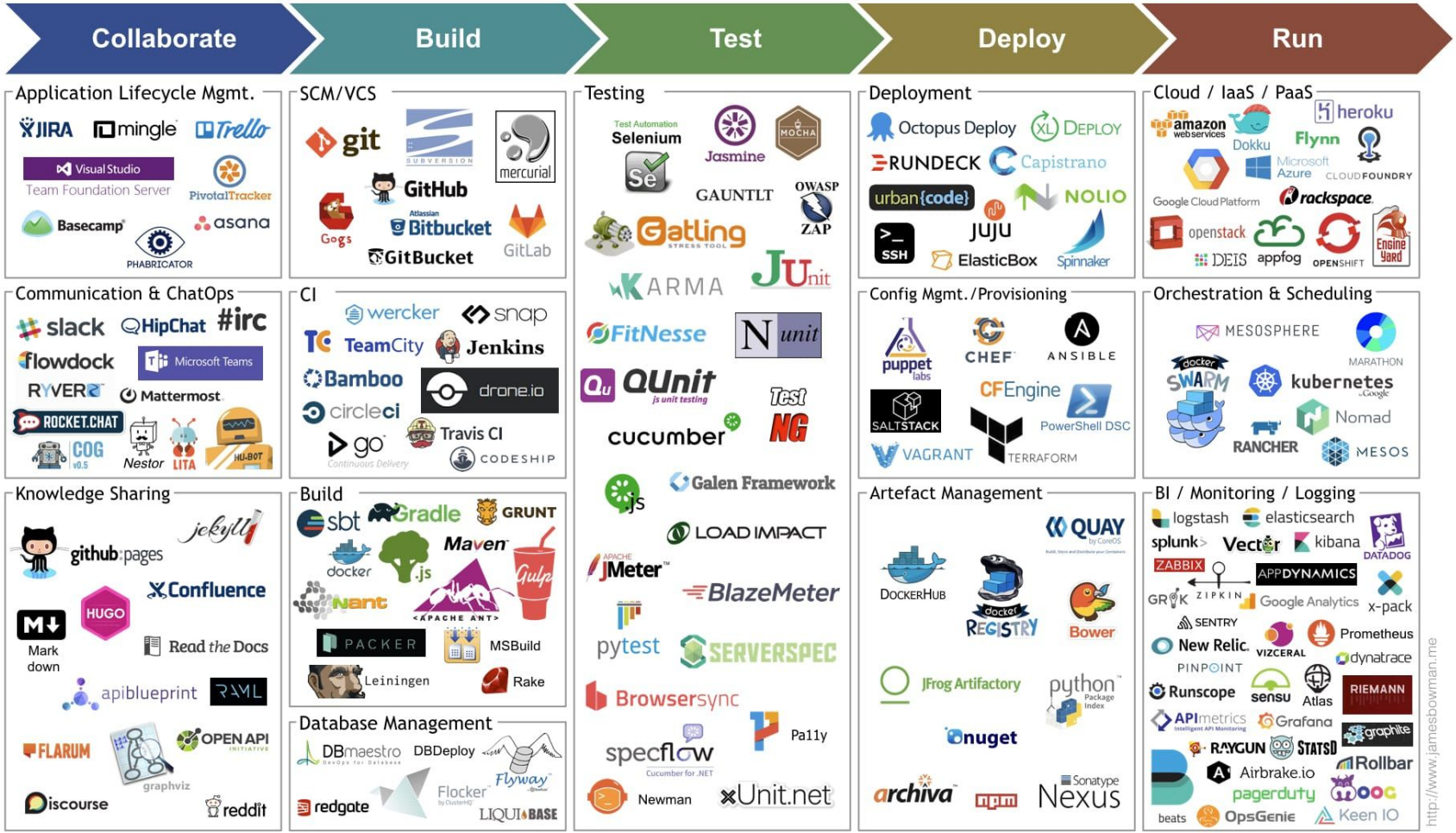


Scanalytics Inc.

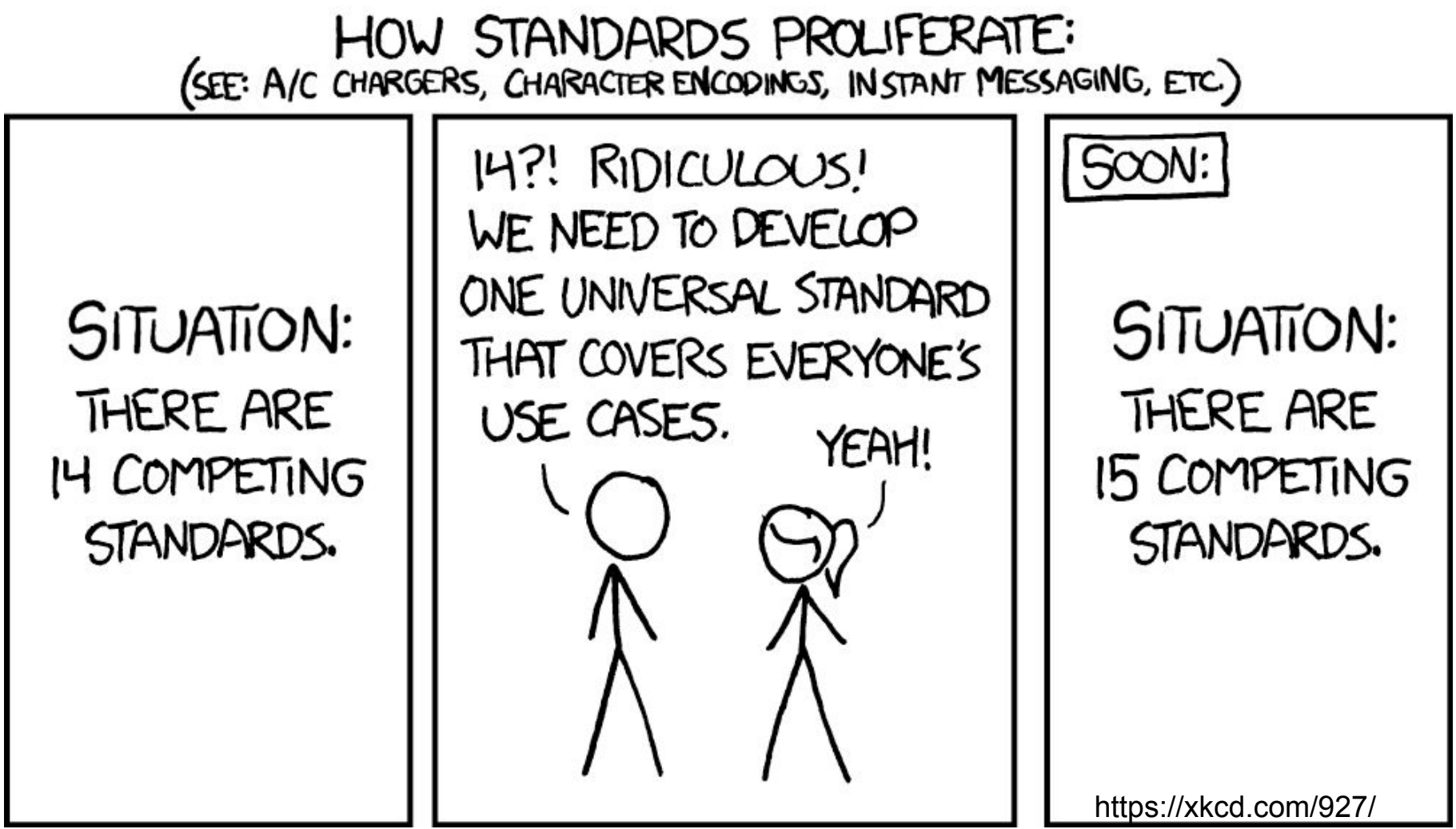
What is DevOps?

- A portmanteau of Development and Operations
- A philosophy/process/culture of software development as an iterative process involving frequent delivery. Technology/tools is not a replacement.
  - See also [agile manifesto]
- Why DevOps?
  - You may not want what you think you want.
  - Long-lived code needs maintenance. IRS Master File, 1962-present
- A number of tools and frameworks exist to support a DevOps workflow

https://profreynolds.com/2018/05/28/the-journey-into-devops-as-a-mindset-not-a-project/



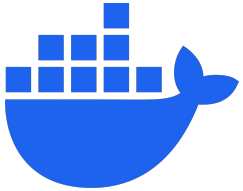
https://www.jamesbowman.me/post/continuous-delivery-tool-landscape/



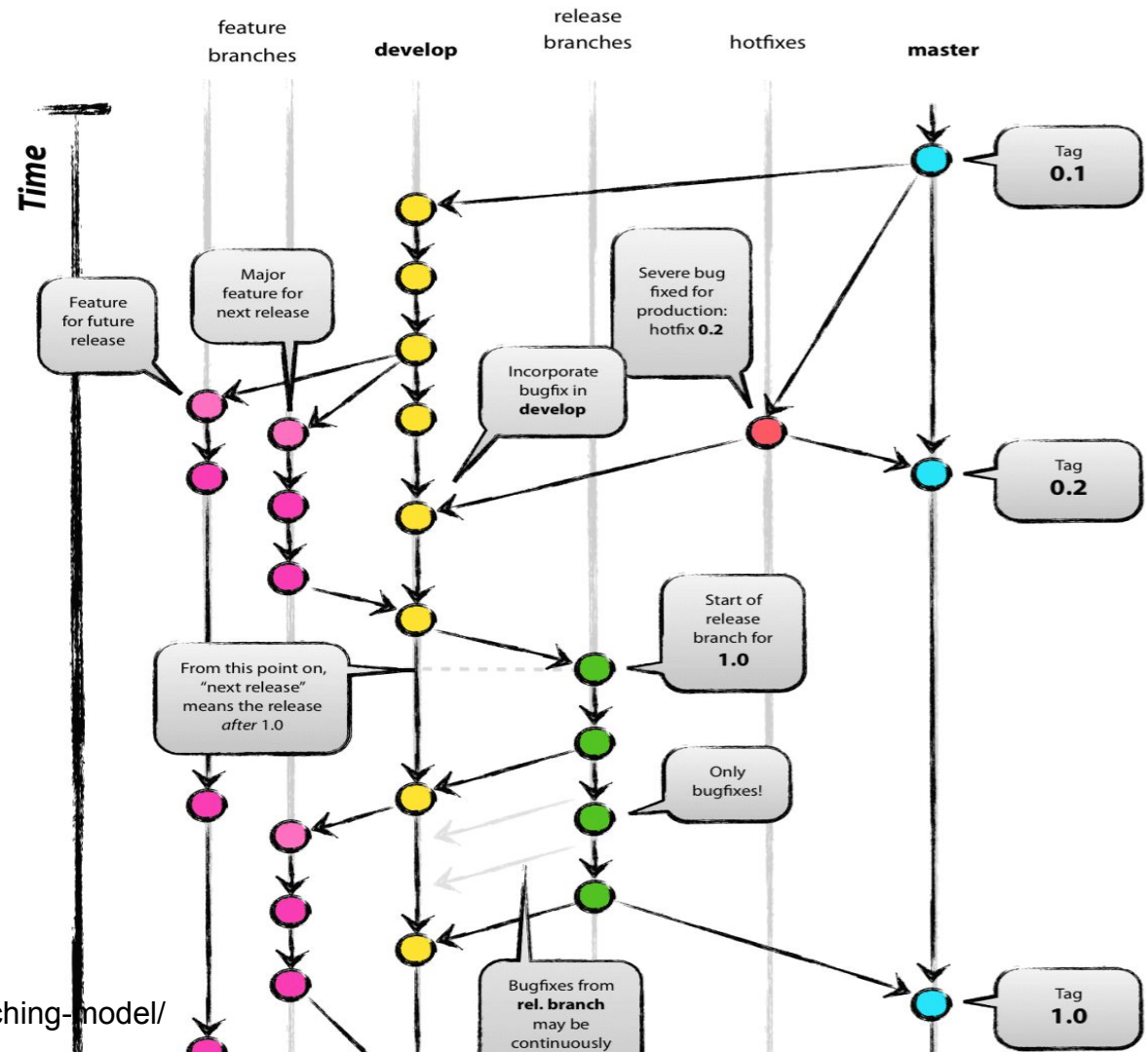
https://xkcd.com/927/

For now, focus on a self-hosted / home-lab setup

- Git
- A simple get/set module
- Webify — Flask
- Containerize — Docker
- Deploy — Google Cloud Compute
- Monitor — Prometheus.io



A Git workflow



https://nvie.com/posts/a-successful-git-branching-model/

A simple set/get module: dict_cache

```python
cache = {}

def set_value(key, value):
    cache[key] = value
    success = True
    return value, success

def get_value(key):
    if key in cache.keys():
        value = cache[key]
        success = True
    else:
        value = None
        success = False
    return value, success
```

```
dict_cache/
├── Dockerfile
├── README.md
├── dict_cache
│   └── __init__.py
└── setup.py
```

```python
from setuptools import setup

setup(
    name='dict_cache',
    version='0.1',
    description='A simple dictionary-based cache',
    author='David Webber',
    author_email='david.webber@gmail.com',
    packages=['dict_cache'],  #same as name
    install_requires=[], #external packages as dependencies
    scripts=[]
)
```

Choose a framework

Flask

https://wiki.python.org/moin/WebFrameworks

Webify the set/get module

```
flask_app
├── Dockerfile
├── README.md
├── flask_app
│   ├── __init__.py
│   └── templates
│       └── index.html
├── requirements.txt
└── setup.py
```

```python
from flask import Flask, jsonify, request, render_template
import dict_cache

app = Flask(__name__)

@app.route("/")
def hello_world():
    return render_template('index.html')

@app.route('/v1/key/')
@app.route('/v1/key/<key>', methods=['GET', 'POST'])
def key(key = None):
    if request.method == 'GET':
        value, success = dict_cache.get_value(key)
        if success:
            msg = "it worked!"
        else:
            msg = f"key '{key}' not found"
    return {'value': value, 'success': success, 'msg': msg}
```

Aside: Google Cloud Run and Google App Engine

```
$ gcloud run deploy
```

https://cloud.google.com/run/docs/quickstarts/build-and-deploy/deploy-python-service

```
$ gcloud app deploy
```

https://cloud.google.com/appengine/docs/standard/python3/building-app/deploying-web-service

Containerize the app with Docker

https://rosecoloredgaming.com/products/game-cartridge-display-stand-sega-genesis

Dockerfile contains the image build instructions

```
# dict_cache/Dockerfile

from python:3

COPY dict_cache/ /staging/dict_cache

COPY setup.py /staging

RUN pip install /staging/
```

```
# flask_app/Dockerfile

from dict_cache

WORKDIR /working/flask_app

# copy the current directory, ignoring files in .dockerignore

COPY . ./

RUN pip install .

EXPOSE 5000

CMD ["flask", "--app", "flask_app", "run", "--host=0.0.0.0"]
```

docker-compose.yaml

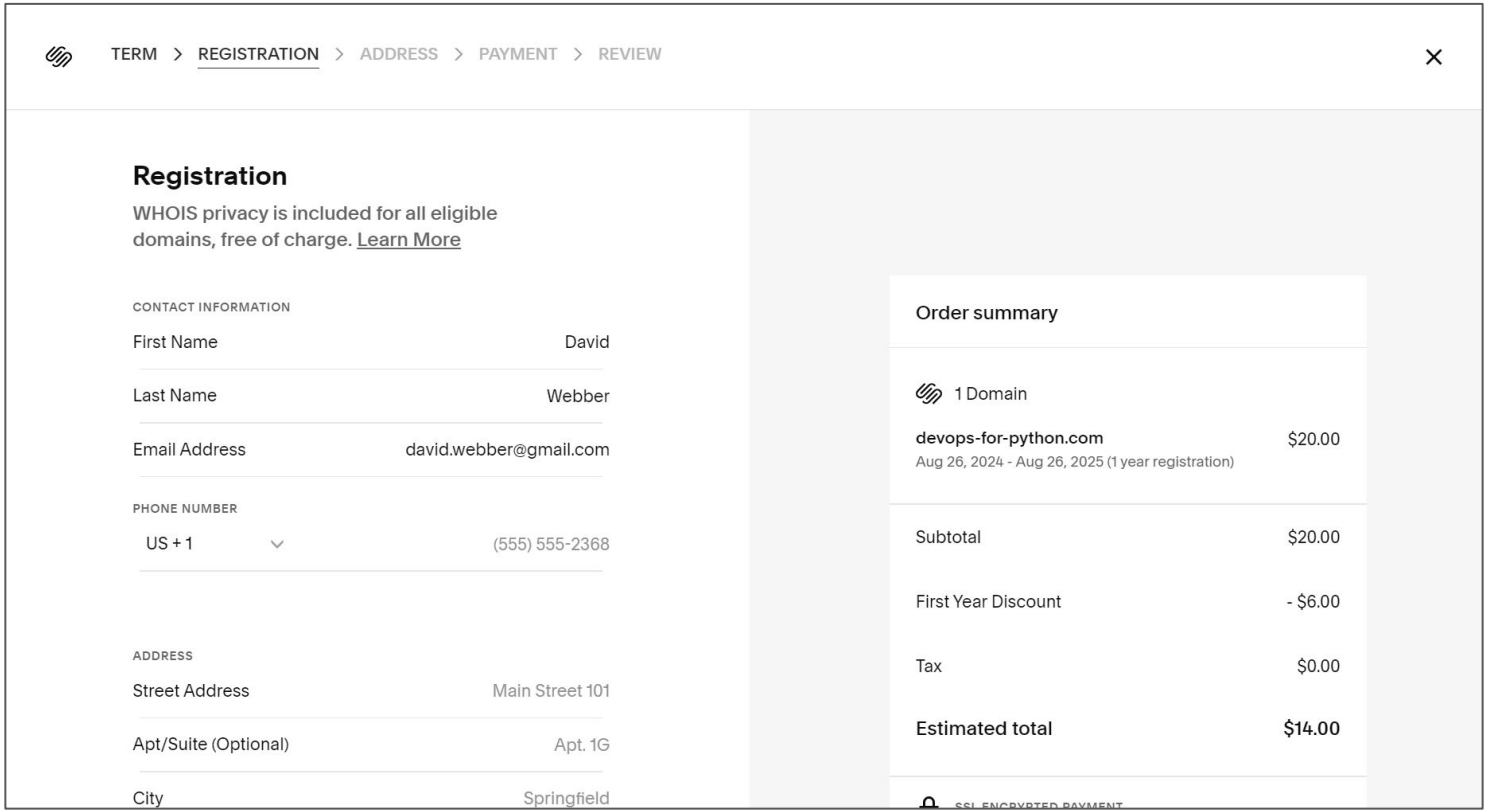
```yaml
services:
    demo:
        restart: unless-stopped
        image: us-central1-docker.pkg.dev/devops-for-python/images/demo:latest
        network_mode: host
        build:
            context: ./flask_app
            dockerfile: Dockerfile
```

Deploy

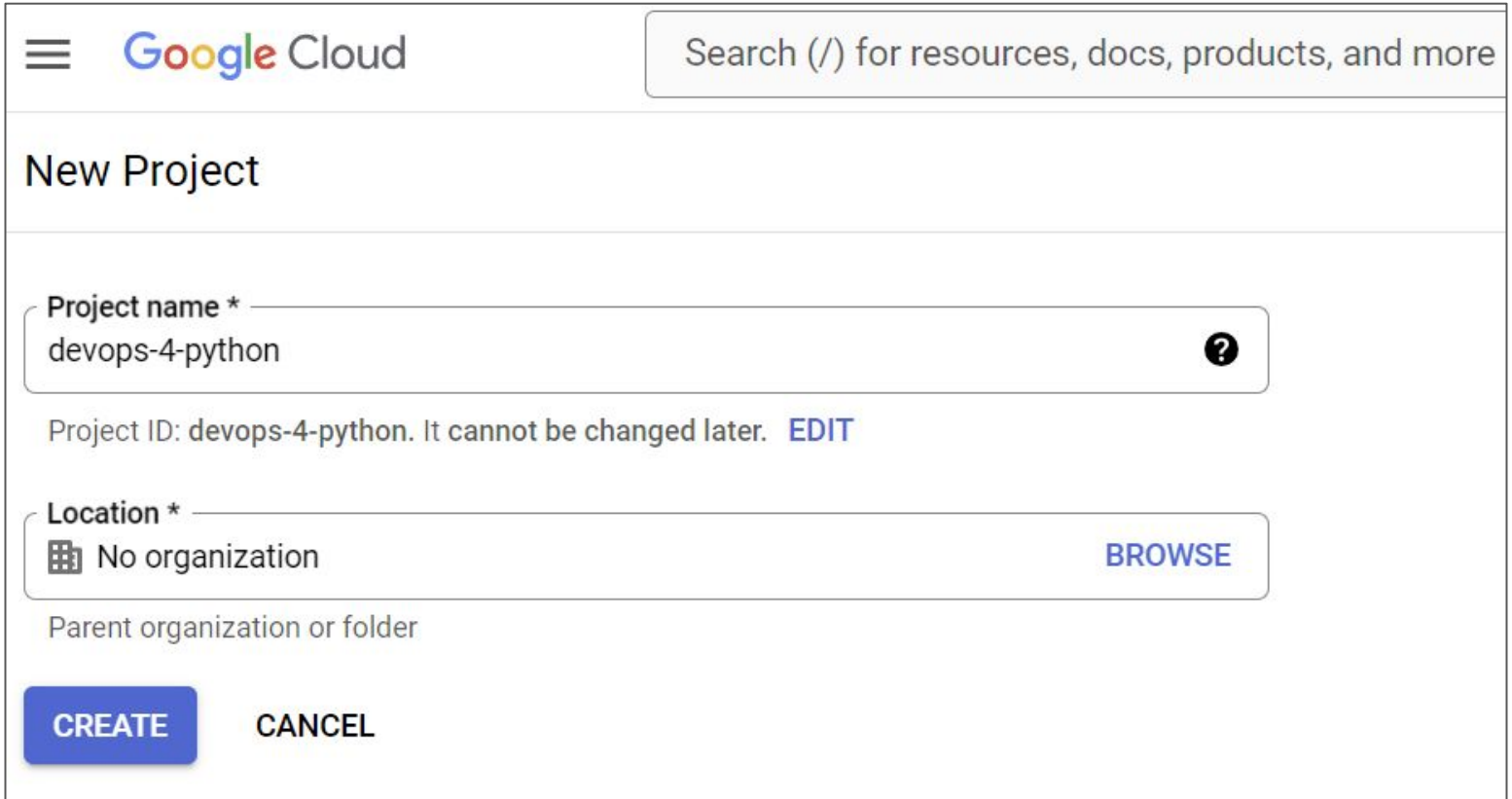
Deploy

- Register a domain (eg squarespace.com)
- Set up a project on console.cloud.google.com
  - Enable compute engine (aka rented server)
    - Create a server
    - Point domain to IP address
  - Enable artifact storage (to hold images)
- Local setup
  - sudo snap install google-cloud-cli --classic
  - gcloud init
  - gcloud auth configure-docker us-central1-docker.pkg.dev
- Server setup
  - sudo apt install docker-compose-v2
  - gcloud auth configure-docker us-central1-docker.pkg.dev

Register a Domain on SquareSpace.com



Create a project on console.cloud.google.com



Monitor

Monitoring and Alerting

- Naive monitoring:
  - If something is wrong: send a slack/text/email/page
- Prometheus.io monitoring
  - If something is wrong
    - AND it has been wrong for over 5 minutes (configurable)
    - THEN send an alert to the alertmanager
  - Alertmanager
    - If there's an alert
    - AND there is no override silence in effect
    - AND it has been longer than alert_interval (eg 3 hours)
    - Then send a message to a particular team on a particular channel
  - Reduces alert "fatigue"
  - Similar to what Google uses internally

Prometheus.io capabilities

- Metrics sources are (mostly) simple http web pages with “key” “value”, one per line
- Prometheus uses a time-series database to store metrics
- Simple analysis functions can do sliding window aggregation and linear projection
- Prometheus allows predictive triggers:
  - Disk will fill up in 24 hours
  - SSL certificate will expire in 14 days
  - Average web or database traffic (connections or data rate) has exceeded expectation
  - 90% quantile latency exceeds threshold

Document your System

Documentation

- Documentation is tech credit, and pays dividends
- README files
  - Getting a development environment set up
- Wiki
  - Overall structure of app, database, frontend, message queue
  - History and Future plans
  - Troubleshooting: Can link directly to wiki pages in alert system messages
- Document what you're going to do
- Get input from stakeholders
  - People who will be administering it, interacting with it
- Build it
  - Frequent feedback
  - Agile cycle
  - Agile manifesto
- Document what you built
  - Where it's running
  - How it's deployed
  - Where the knobs are
  - Where the secrets and sacred crystals live

Summary

- What is DevOps?
- A simple get/set module
- Webify with Flask
- Containerize with Docker
- Deploy with Docker compose and GCP
- Monitor with Prometheus

Follow Along on GitHub

Thanks for attending!

(PS: Board of Visitors)

Physics Board of Visitors

I sit on the UW-Madison Physics Board of Visitors.

My goal is to exist as a node between academic and industry networks.

If your company provides internships for physics undergraduates, please ping me.

Addendum

Organizations create software like themselves

- Tightly knit teams create tightly knit software
- Geographically separate teams create geographically separate modules
- Everyone on the team needs to speak the same language
  - Eg Rust, Python, node

Backup

DevOops

- Croudstrike

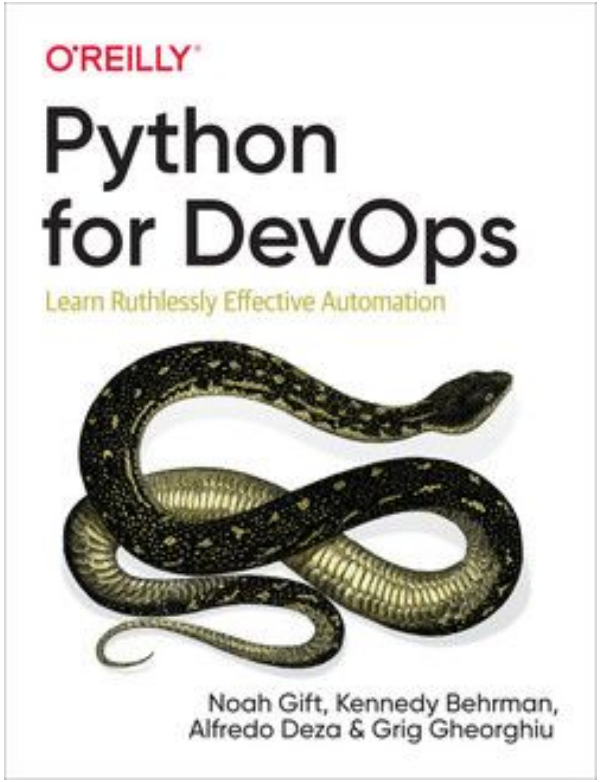
Configuration

- Command-line options
- Environment variables
- Configuration file
- Sensible defaults

Get a Value, Set a Value

- Can be used as a basis for a lot of stuff
  - Simple object storage
  - Voting system
  - Seat reservation for concert or airline
- Version 1
  - Dictionary
- Version 2
  - Lock, with timeout
- Version 3
  - Robust backend

Not in this DevOps for Python talk



- Ansible
- Jupyter Notebooks