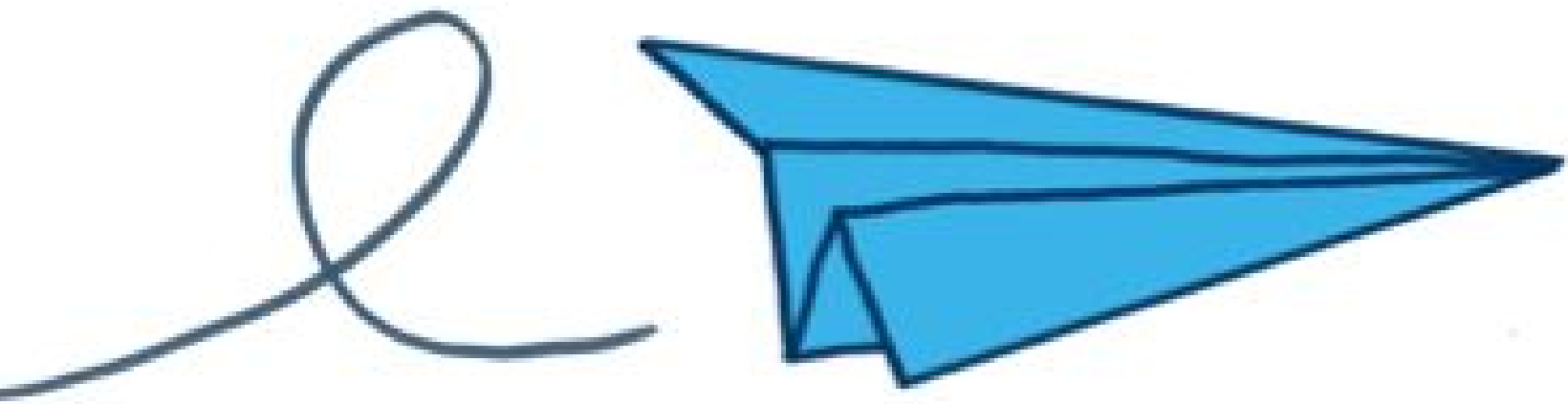
@sebineubauer
https://github.com/sebastianneubauer
sebastian.neubauer@blue-yonder.com

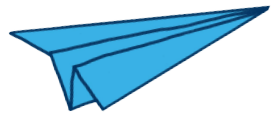


# A Pythonic Approach to Continuous Delivery

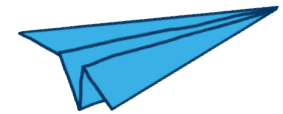
Sebastian Neubauer

Config Management Camp 2016, Gent

blue yonder



# How does a **delivery pipeline** look like?



# I have working python code, how do I start now?

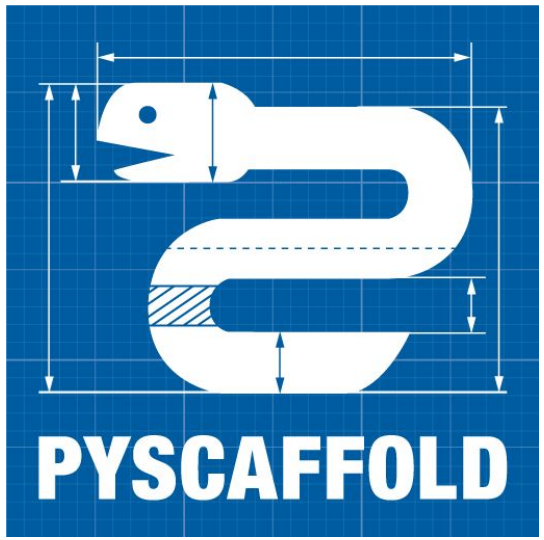
```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```

# A Proper Deployment Artifact

- This means put up everything for a proper deployable artifact:
  - python package
  - debian package
  - fancy docker
- It should be uniquely versioned
- It should manage dependencies

# A Proper Deployment Artifact

- This means put up everything for a proper deployable artifact:
  - python package
  - debian package
  - fancy docker
- It should be uniquely versioned
- It should manage dependencies
- Hint: https://github.com/blue-yonder/pyscaffold




```
>pip install pyscaffold
>putup my_app
```

# Continuous Integration

- All automated tests are executed each time someone commits to master

```
>python setup.py test
```

- Any CI system will do the job: buildbot, travis...

Jenkins

# Continuous Integration

- The result of CI is a fixed artifact with a unique version

```
>python setup.py sdist
```

- If you use pyscaffold, a PEP440 compatible version is generated from the git commit and tag:

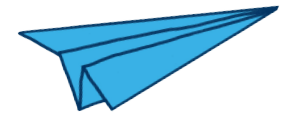
```
0.0.1.post0.dev15+g172635
```

Each commit which passes the tests is a new package

# Fill up the Artifact Repository

- Ah, yes you need one, let's use the: http://doc.devpi.net/
- Devpi: secure, on-premise, open source, pypi compatible artifact repository (short: index)

```
>devpi upload
```

"That was the fun part! Now comes **pain, tears and configuration**"

Personal feeling:

Alarming lack of interest and knowledge gaps in the python universe!

# Automated Deploy

- For automated acceptance test, we need a fully functional running instance, deployed in a testing stage / test environment (each commit!)
- We use ansible, because it's: python, simple, lightweight, declarative,...(highest acceptance in the python community...)

# Example Ansible Playbook

```
---
- hosts: webservers
  tasks:
  - name: ensure my app is installed
    pip:
      name=my_app
      virtualenv=/my_app_home/venv
      extra_args='-i https://our_devpi/simple --pre -U'
      state=present

  - name: start the app
    shell: /my_app_home/venv/bin/my_app_started
```

# Last step to Production

- You might want to have some additional non-functional tests:
  - performance
  - security
  - explorative
- You might want to have some manual approval (e.g. feature flags in django)

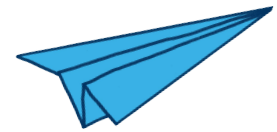
# What could possibly go **wrong**?

Example from official pip docs:

“As it is now, pip doesn't have true dependency resolution, but instead simply uses the first specification it finds for a project.”

# Traps, Tips & Tricks, Dangers…

- Stay pythonic: Keep it simple stupid!
- Evangelize that config management/deployment is important!
- Maintain and refactor your deployment



# What should the **future** bring?

# The not so perfect parts...

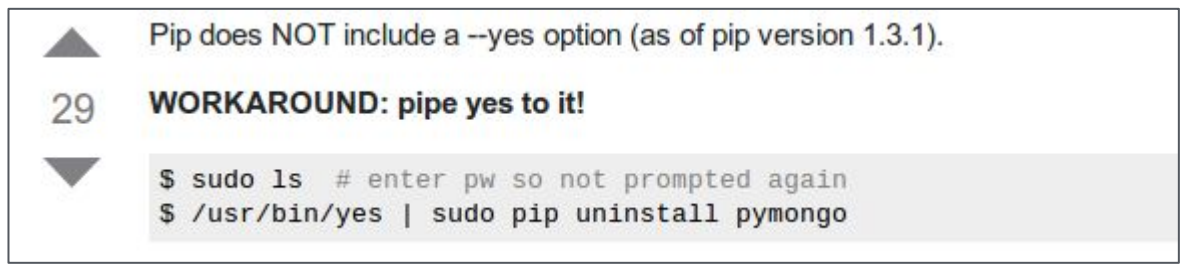
- The two worlds should unite: OS package managers vs. pip

# The not so perfect parts...

- The two worlds should unite: OS package managers vs. pip
- A pythonic continuous delivery tool is still missing, jenkins is not sufficient:
  - what configuration is deployed where
  - access management
  - awareness of the delivery pipeline

# The not so perfect parts...

- The two worlds should unite: OS package managers vs. pip
- A pythonic continuous delivery tool is still missing, jenkins is not sufficient:
  - what configuration is deployed where
  - access management
  - awareness of the delivery pipeline
- Many tools are still optimized for a manual workflow



# Summary

- You can build your own CD pipeline in a python universe, just start today!
- Example building blocks are:
  - pyscaffold for python packages
  - devpi as artifact repository
  - jenkins for CI and steering
  - python unittest for tests
  - ansible for automated deploys

# Summary

- You can build your own CD pipeline in a python universe, just start today!
- Example building blocks are:
  - pyscaffold for python packages
  - devpi as artifact repository
  - jenkins for CI and steering
  - python unittest for tests
  - ansible for automated deploys
  - need to improve awareness

Thank **you**!