

Modern Continuous Delivery

*“ deploy to production
from commit #1*

Peter Bittner

Developer
of people, companies and code

@peterbittner, django@bittner.it



behave-django painless/tox PythonTurtle ansible-role-software
django-bootstrap-static codeship-yaml .djangocms-maps django-probes
django-apptemplates django-organice pyclean

Continuous Delivery

*“ a set of practices and principles in software engineering aimed at **building, testing, and releasing** software **safely, faster, more frequently, and in a sustainable** way.*

Continuous Delivery

*“ a set of practices and principles in software engineering aimed at **building, testing, and releasing** software **safely, faster, more frequently, and in a sustainable** way.*

*“ the goal is to put the **release schedule** in the **hands of the business**, not in the hands of IT.*

Continuous
integration ??

Continuous Delivery

*“ a set of practices and principles in software engineering aimed at **building, testing, and releasing** software **safely, faster, more frequently, and in a sustainable** way.*

*“ the goal is to put the **release schedule** in the **hands of the business**, not in the hands of IT.*

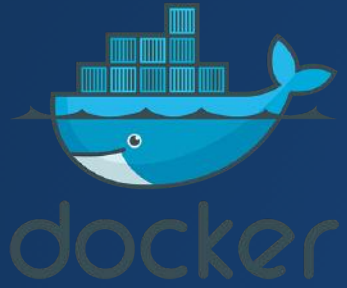
Continuous
integration ??

Continuous
deployment ??

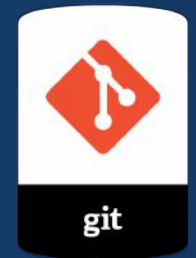
Continuous Delivery

“ a set of practices and principles in software engineering aimed at **building, testing, and releasing** software **safely, faster, more frequently, and in a sustainable** way.

“ the goal is to put the **release schedule** in the **hands of the business**, not in the hands of IT.

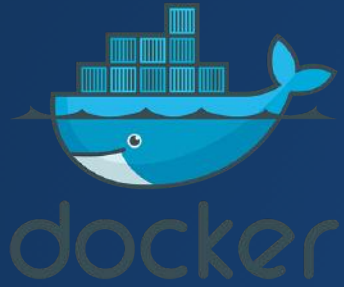


Modern?



CLOUD NATIVE
COMPUTING FOUNDATION





Modern?

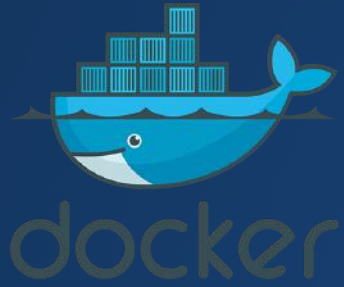


Immutable infrastructure



CLOUD NATIVE
COMPUTING FOUNDATION



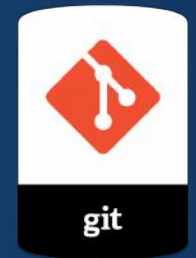


Modern?



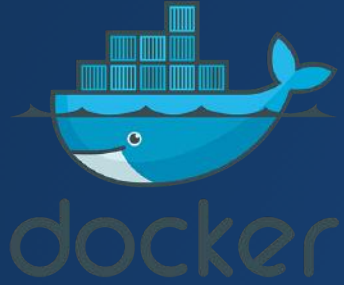
Immutable infrastructure

Container orchestration



CLOUD NATIVE
COMPUTING FOUNDATION



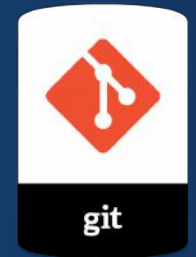


Modern?

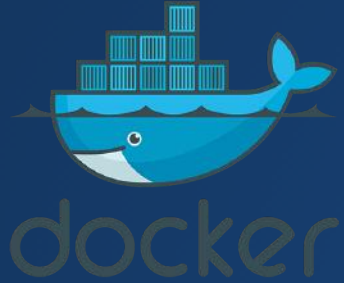
Immutable infrastructure

Container orchestration

Version control + automation



CLOUD NATIVE
COMPUTING FOUNDATION



Modern?

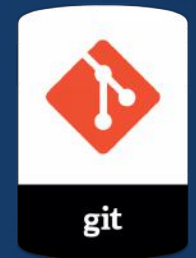


Immutable infrastructure

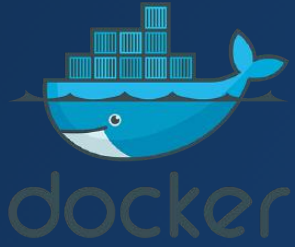
Container orchestration

Version control + automation

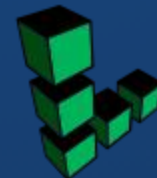
Cloud-native applications



CLOUD NATIVE
COMPUTING FOUNDATION



Choice or Lock-in?





There must be a better way!

1. Clean code
2. Deploy to production from commit #1



Demo

```
00:22 $  
✓ ~/Development/scratch/euopython-demo [master L|✓]  
00:22 $ tree  
├── application  
│   ├── __init__.py  
│   ├── settings.py  
│   ├── urls.py  
│   └── wsgi.py  
├── deployment  
│   ├── application  
│   │   ├── Dockerfile  
│   │   ├── entrypoint.sh  
│   │   └── uwsgi.ini  
│   ├── application-secrets.yaml  
│   ├── application.yaml  
│   ├── envs  
│   │   ├── development  
│   │   ├── integration  
│   │   └── production  
│   ├── postgres-secrets.yaml  
│   ├── postgres.yaml  
│   ├── README.rst  
│   └── webservice  
│       └── nginx.conf  
├── docker-compose.yml  
├── LICENSE  
├── manage.py  
├── README.rst  
├── requirements  
│   ├── base.in  
│   ├── development.in  
│   ├── production.in  
│   └── production.txt  
├── requirements.txt  
├── tests  
│   ├── acceptance  
│   │   ├── environment.py  
│   │   ├── features  
│   │   │   └── login-logout.feature  
│   │   └── steps  
│   │       ├── fixtures.py  
│   │       ├── given.py  
│   │       ├── then.py  
│   │       └── when.py  
│   ├── README.rst  
│   ├── unit  
│   │   └── test_application.py  
└── tox.ini  
  
11 directories, 34 files  
✓ ~/Development/scratch/euopython-demo [master L|✓]  
00:22 $
```



Responsibility Layers



Responsibility Layers

Application



Responsibility Layers

Application
Development



Responsibility Layers

Application
Development
Deployment



Responsibility Layers

Application
Development
Deployment
Automation



Application

One environment!

12-factor app.

Build with features.

Compose in environments.

```
✓ ~/Development/scratch/europython-demo [master | +26]
01:04 $ tree
.
├── application
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── LICENSE
├── manage.py
├── README.rst
└── requirements.txt

1 directory, 8 files
✓ ~/Development/scratch/europython-demo [master | +26]
01:04 $ █
[0] 0:bash*Z
```

Development

Make it easy!

Standard practices.

No comprehensive instructions.

Simple & user-friendly!

```
✓ ~/Development/scratch/euopython-demo [master | +22]
01:14 $ tree
.
├── application
├── docker-compose.yml
├── manage.py
├── requirements.txt
├── tests
│   ├── acceptance
│   │   ├── environment.py
│   │   ├── features
│   │   │   └── login-logout.feature
│   │   └── steps
│   │       ├── fixtures.py
│   │       ├── given.py
│   │       ├── then.py
│   │       └── when.py
│   ├── README.rst
│   └── unit
│       └── test_application.py
└── tox.ini

6 directories, 12 files
✓ ~/Development/scratch/euopython-demo [master | +22]
01:14 $ █
[0] 0:bash*Z
```

Deployment

Make it beautiful!

Easy to explain.

*Generate + seal your secrets,
or seal + commit your secrets.*

```
✓ ~/Development/scratch/euopython-demo [master | +23]
01:41 $ tree
.
├── application
├── deployment
│   ├── application
│   │   ├── Dockerfile
│   │   ├── entrypoint.sh
│   │   └── uwsgi.ini
│   ├── application-secrets.yaml
│   ├── application.yaml
│   ├── envs
│   │   ├── development
│   │   ├── integration
│   │   └── production
│   ├── postgres-secrets.yaml
│   ├── postgres.yaml
│   └── webserver
│       └── nginx.conf
5 directories, 11 files
✓ ~/Development/scratch/euopython-demo [master | +23]
01:41 $ █
[0] 0:bash*Z
```

Automation

Keep it simple!

What you would do manually.

Tell a story!

ASAP

```
✓ ~/Development/scratch/euopython-demo [master | +39...6]
01:50 $ tree -a
.
├── deployment
│   └── application
│       └── Dockerfile
├── .dockerignore
├── .gitlab-ci.yml
├── tests
│   ├── acceptance
│   └── unit
└── tox.ini

5 directories, 4 files
✓ ~/Development/scratch/euopython-demo [master | +39...6]
01:50 $ █
```

ASAP!

as simple as possible

Deploy to production!

often + from commit #1

Iterate!
... and improve

Agile, please.

test-driven, pair-programming

Free your software

no secrets, no security holes

*“ The only way to go fast
is to go well.*

--- Robert C. Martin

Thank you!
for your precious time



Painless Software
Less pain, more fun.



Pythonistas Oath

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough
to break the rules.
Although practicality beats purity.

Pythonistas Oath

Errors should never pass silently.
Unless explicitly silenced.

In the face of ambiguity, refuse
the temptation to guess.

There should be one-- only one --obvious way
to do it.

Although that way may not be obvious
at first sight.

Pythonistas Oath

Now is better than never.
Although never is often better
than **right** now.
If the implementation is hard to explain,
it's a bad idea.
If the implementation is easy to explain,
it may be a good idea.

Pythonistas Oath

Continuous delivery is
a honking great idea.
If you deploy to production
from commit #1.

Let's do it! -- I start today.

Python