Modern Continuous Delivery

“deploy to production from commit #1"
Peter Bittner
Developer of people, companies and code

@peterbittner, django@bittner.it

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

Source: painless.software/continuous-delivery
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.

Source: painless.software/continuous-delivery
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.

Source: painless.software/continuous-delivery
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."

Source: painless.software/continuous-delivery
Modern?
Modern?

Immutable infrastructure
Modern?

Immutable infrastructure

Container orchestration
Modern?

Immutable infrastructure
Container orchestration
Version control + automation
Modern?

Immutable infrastructure

Container orchestration

Version control + automation

Cloud-native applications
Choice or Lock-in?
There must be a better way!

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

Make it easy!
Standard practices.

No comprehensive instructions.
Simple & user-friendly!
Deployment

Make it beautiful!
Easy to explain.

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

Keep it simple!
What you would do manually.

Tell a story!
ASAP
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

Source: Technology and Friends, Episode 354, 2015
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.
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.
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*