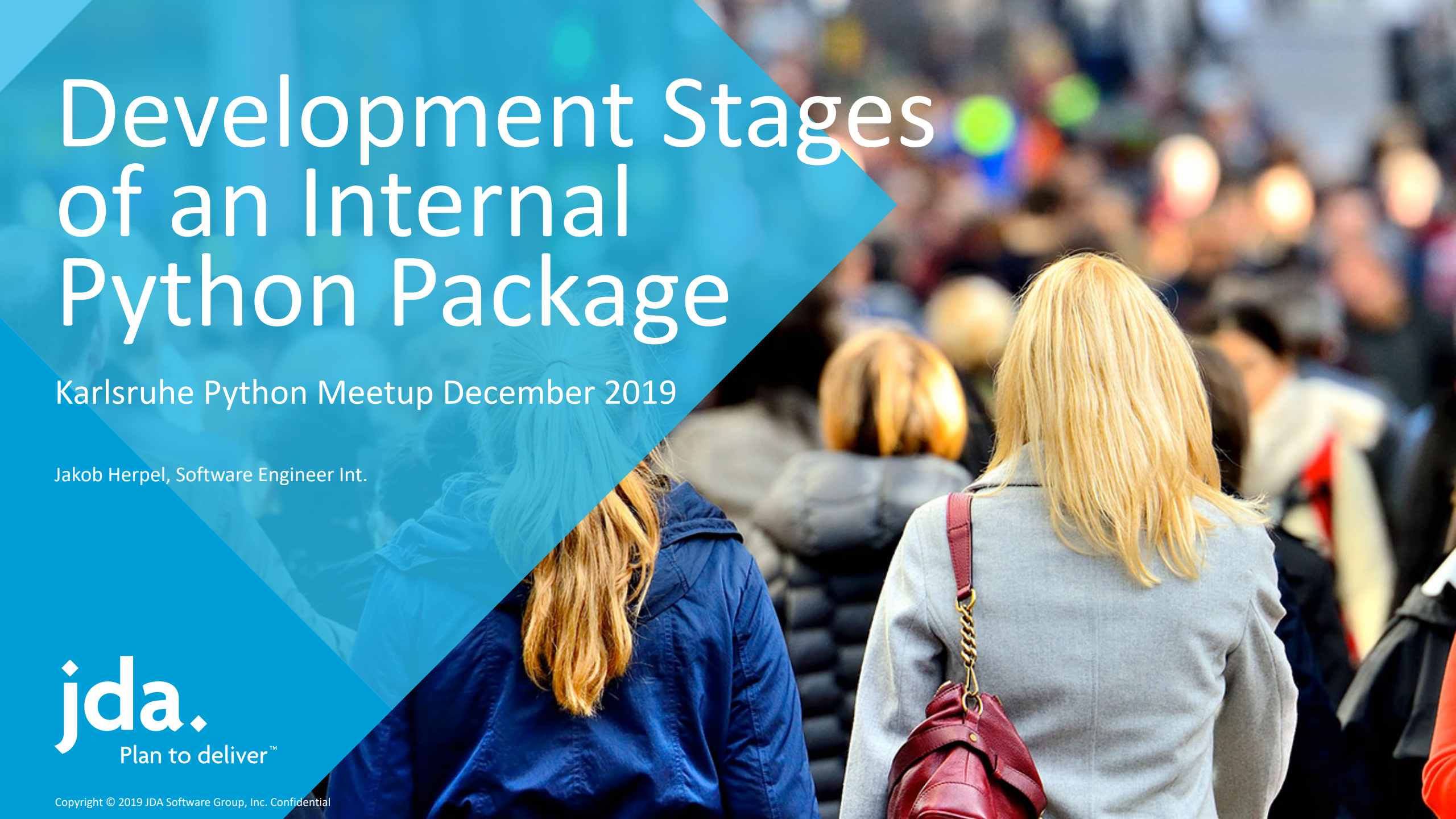
Development Stages of an Internal Python Package

Karlsruhe Python Meetup December 2019

Jakob Herpel, Software Engineer Int.



jda.
Plan to deliver™

Stage -1: The Problem

- Professional software engineers are very good at producing a lot of additional complexity
- Services and packages tend to diverge over time
- The solution:
  1. Make artefacts as uniform as possible
  2. Make deleting things your favorite hobby

Stage 0: Do we need this?

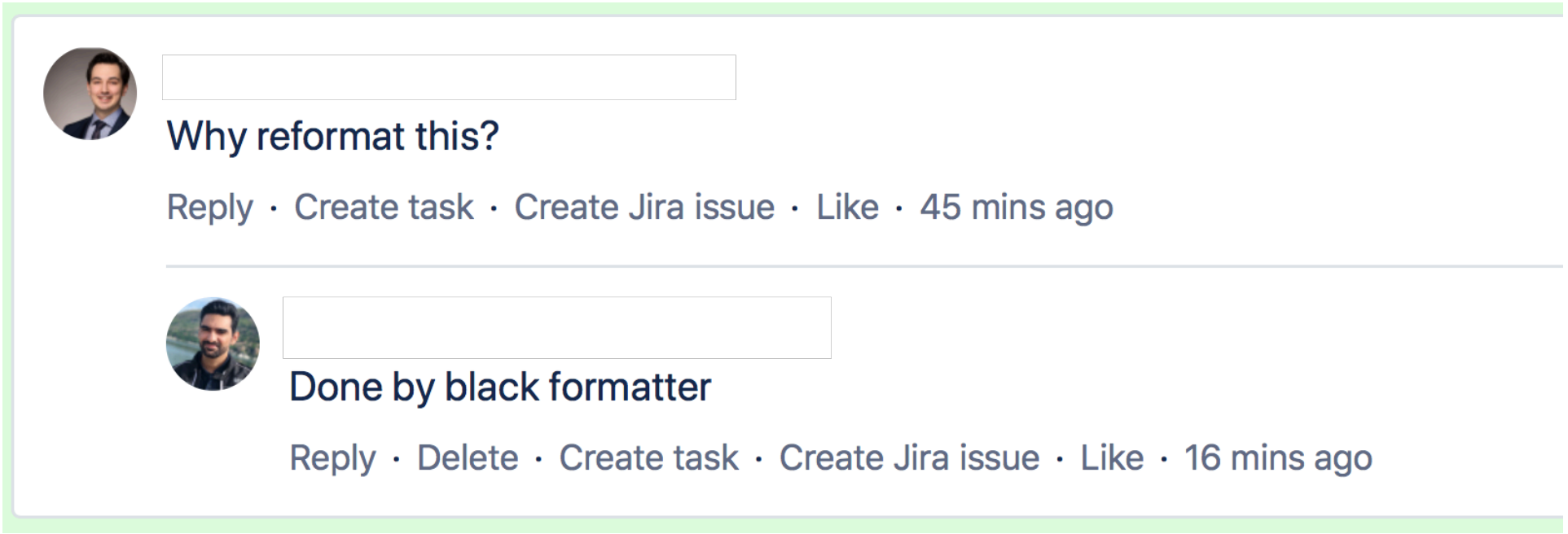
- Can we use existing open source software?
- Can we use a cloud service?
- Have other teams solved a similar problem?
- Have **we** solved a similar problem in the past?
- Will this solve the problem?
- Work with your team process (scrum, Kanban, ...)

Stage 1: Reliable and Modular Tests

jda.

- Write tests!
- `pytest` is way cooler than `unittest`
- Use pytest fixtures for composable test setups

Stage 2: formatting – just use `black`



Stage 2: formatting and linting

jda.

- Black: No more discussions about code format
- Pyflakes / flake8: additional linting

Stage 3: managing requirements

- Often overlooked source of bugs and incidents!
- Pin your requirements with `pip-compile`
- Actually check changelogs of new versions

```
$ pip-compile requirements.in
#
# This file is autogenerated by pip-compile
# To update, run:
#
#    pip-compile requirements.in
#

certifi==2019.11.28       # via requests
chardet==3.0.4            # via requests
idna==2.8                 # via requests
requests==2.22.0
urllib3==1.25.7           # via requests
```

Stage 4: ossaudit

jda.

- Automatically scan your requirements for public security vulnerabilities
- Have a process for ignoring vulnerabilities

Stage 4: ossaudit

```
$ ossaudit -f requirements.txt --column name --column version --column title --column id

+-----------+---------+---------------------------+---------------------------+
|    name   | version |           title           |             id            |
+===========+=========+===========================+===========================+
| sqlalchemy | 1.2.0   | [CVE-2019-7164]  Improper | 69cff794-78c9-4eeb-93e9-3 |
|           |         | Neutralization of Special | b88b9817ed5               |
|           |         | Elements used in an SQL   |                           |
|           |         | Command (SQL Injection)   |                           |
+-----------+---------+---------------------------+---------------------------+
Found 1 vulnerabilities in 67 packages
```

Stage 5: The CI pipeline

jda.

- Share continuous integration pipelines across packages
- Run `black`, `pytest`, `ossaudit`, `mypy`, etc. automatically
- Run for every new commit + once per day
- Alert for failing builds

Stage N: Going too far

- Automation is worth additional complexity!
- Different problems require different solutions
- Different teams have different needs
- Often you cannot change everything at once, but you need to start somewhere

Links

- pytest.org
- https://github.com/psf/black
- PyCon2019 talk about black by Łukasz Langa: https://www.youtube.com/watch?v=esZLCuWs_2Y
- https://jenkins.io
- https://ossindex.sonatype.org
- https://github.com/illikainen/ossaudit