Writing and Running Tests in Docker

Or How to Test your Web Application Seamlessly...

Alexandre Figura & Steffen Neubauer @ SysEleven GmbH
Agenda

1. **Setting-up** our development environment.
2. **Writing** tests with **Pytest**.
3. **Automating** tests with **Tox**.
4. **Running** our application in **Docker Compose**.
5. **Managing** our workflow with **Invoke**.
Useful Links

1. **Pytest** documentation: [https://docs.pytest.org/](https://docs.pytest.org/)
2. **Tox** documentation: [https://tox.readthedocs.io/](https://tox.readthedocs.io/)
3. **Docker** documentation: [https://docs.docker.com/engine/reference/builder/](https://docs.docker.com/engine/reference/builder/)
4. **Docker Compose** documentation: [https://docs.docker.com/compose/](https://docs.docker.com/compose/)
Setting-Up Environment

1. **Clone the demo application** (blog + web API):
   - `git clone https://github.com/arugifa/ep2018-workshop`
   - `git reset --hard setup`

2. **Install requirements:**
   - Python 3.6
   - Tox (+ Pip + Virtualenv)
   - Docker & Docker Compose
   - Google Chrome

3. **Create a temporary virtualenv:**
   - `virtualenv -p python3.6 venv` **&** `source venv/bin/activate`
Writing Tests

1. Install Pytest + extensions:
   pip install requirements-test.txt

2. Have a look to existing fixtures:
   vim tests/conftest.py

3. Write tests:
   - Acceptance tests with Pytest-BDD for the blog,
   - End-to-end tests with Webtest for the web API.
Automating Tests

- **Write a Tox file** *(Tox.ini)* **with:**
  - one **Testing** environment,
  - two **Development** environments:
    - One to be used **locally** *(to get auto-completion in your IDE)*,
    - Another one to be used **in Docker** later on.
  - one **Linting** environment,
  - one environment to check **Security Issues** in dependencies.
Running Tests

• Write two Dockerfiles:
  1. One for Production:
     • Based on **Python 3.6 Alpine**,  
     • With **manage.py** as entrypoint,  
     • We should be able to configure the database connection with an environment variable,
  2. Another one for Testing:
     • Based on the PROD image,  
     • With test requirements and **Tox** installed inside the container.
• Write a Docker Compose file (docker-compose.yml):
  • With two services:
    • One for the web application,  
    • Another one for **PostgreSQL**.
  • Share your local source code with the container.
Managing Workflow

- Write Invoke tasks (tasks.py):
  - Two tasks:
    - One to run the demo server,
    - Another one to run tests with Tox.
  - Both tasks should run in Docker Compose,
  - Provide a debug mode so we can:
    - manually execute manage.py or Pytest inside Docker,
    - and use PDB for troubleshooting bugs.
We're hiring! Interested? Just say: "Hello!"

Mail: jobs@syseleven.de
WhatsApp, SMS, Threema: +49 171 89 34 073