Production-ready applications with Python

Christian Barra

PyCon Estonia 2020

@christianbarra

A bit about myself

- Based in Berlin
- Tech Lead @ Infarm
- PSF Fellow
- Run pybootcamp.com (training & consulting on Python and DevOps)

Agenda

- What production-ready means
- Production-ready pillars
    - Developer experience
    - Automation
    - Documentation
    - Observability
    - Stability
- Conclusions

What does ***production-ready*** mean?

[…] is capable of delivering business value, in a reliable way, consistently and without compromising on quality.

What is a **production-ready** application?

A production-ready application

1. Develop and deploy reliably (ideally by anyone at anytime)
2. Onboard new developers easily
3. Introspect the application at runtime
4. Dependencies are well known, software and infrastructure
5. Other systems can rely on it (APIs and availability)

A production-ready application

1. DevEx and automation
2. Documentation and DevEx
3. Observability
4. Stability, DevEx and automation
5. Stability

@christianbarra

A production-ready application

1. Developer Experience
2. Automation
3. Documentation
4. Observability
5. Stability

Developer experience

Developer Experience

- How do I run PostgreSQL on my machine?
- How do install project dependencies?
- How do I run tests?
- How do I run this project?

@christianbarra

Developer Experience

- Docker compose
- How-to-start inside your README
- Makefile
- Pre-commit
- Poetry

Automation

@christianbarra

Automation

- You make manual releases
- You don’t know when a release is broken
- You don’t know what has been released
- You can’t easily roll back
- You don’t trust your release process

@christianbarra

Automation

- Manual processes are banned
- Healthchecks
- Build & linters & tests merge flow
- 1 (or 2) click release process
- Each commit is a docker image
- Infrastructure as code

Documentation

Documentation

- Who are the users of this application?
- Production is broken, how do I debug it?
- What are the steps to run the application locally?
- Which systems are relying on this application?
- How do I create a release?
- What happens if the application goes down?

Documentation

- Proper README.md
- **Always onboard new members through your doc**
- Some effort 💪
- How to operate your application: doc/runbook.md
- How to debug things: doc/playbook.md
- How to do a release: doc/release.md

Observability

Observability

- **Your users know before you when the application is not working like expected**
- Logs are not giving you enough information (or context)
- You can’t answer simple metric related questions about your application (throughput, req/s, average latency, …)

@christianbarra

Observability

- **Structured logging**
- Alerts on top of your metrics and logs
- Metrics, start with something easy
- Tracing (app & apps)

Stability

Stability

- You’re paged too often
- People complain about you breaking APIs
- Uptime is terrible
- Doing a release is scary

Stability

- Linters (isort, flake8, black, mypy)
- Follow a code review process
- Standardized release process
- Orchestrator for deployments (k8s, nomad, …)
- Schema first (GraphQL, OpenAPI, protobuf)
- Improve test coverage

Conclusions

Start with low-hanging fruit and then evolve from there.

The payoff of any of these improvements can be small today but will be exponential in the long term.

Create a production-ready checklist inside your team or organization.

Paying technical debt
=
protecting revenue streams

Thanks, time for Q&A!

- @christianbarra
- Checklist: https://bit.ly/36PVmyu
- For more about this: pybootcamp.com