



sunscrapers

U mnie (nie)działa

Przemek Lewandowski

PyStok - 21.11.2019

About me



sunscrapers



The Story



1. The problem when the team grows
2. Real life examples
3. Root causes
4. Changes in how we work
5. Key takeaways

The Problem



~_(\ツ)_/~

It works on my machine

The Problem



(- ° □ °) - _ _ _ _

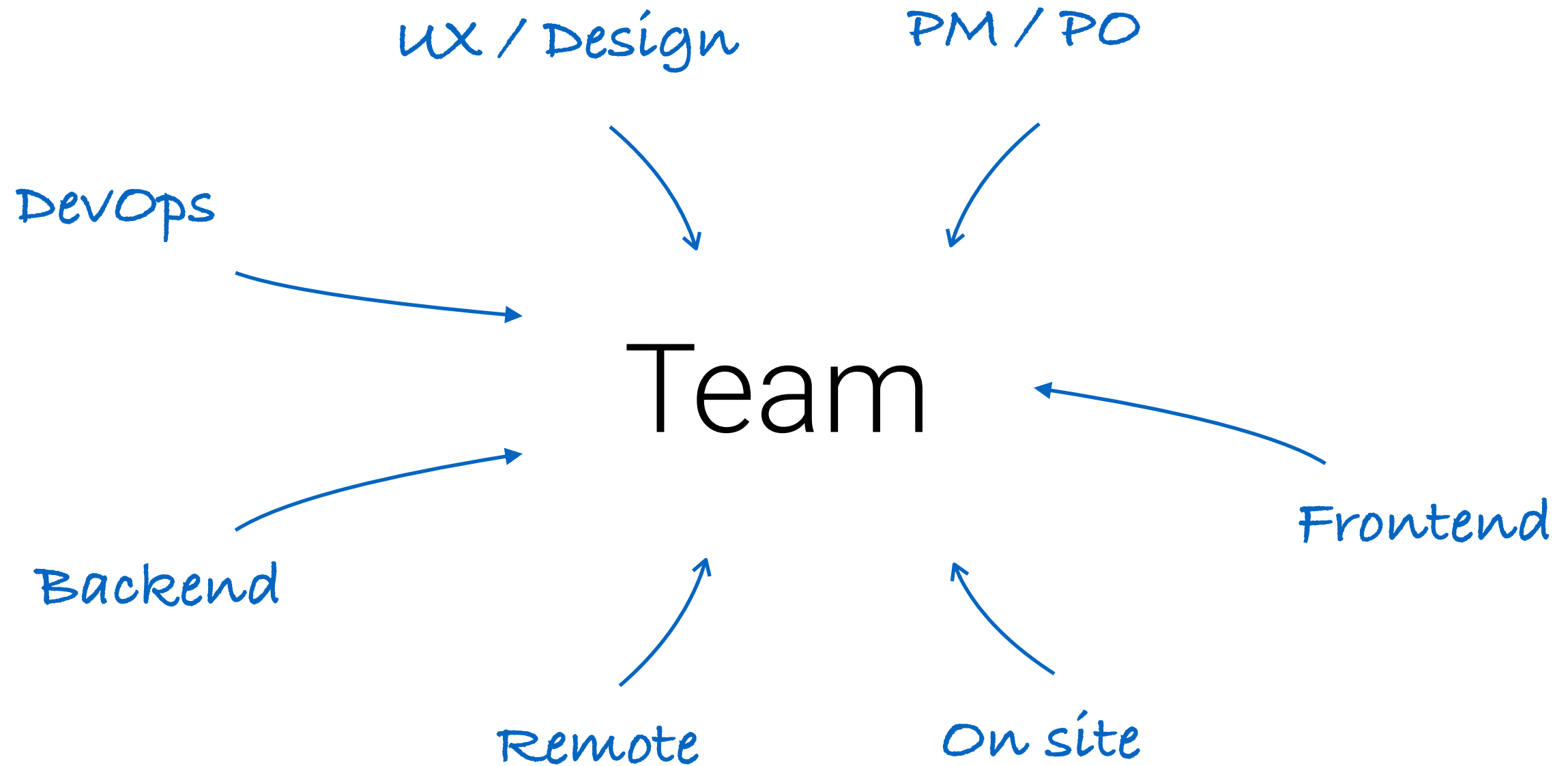
It does **not** work on my machine

A photograph of a paved sidewalk made of interlocking bricks in shades of grey, red, and green. A tree trunk is on the right side of the sidewalk. To the left of the sidewalk is a dirt area with some dry leaves and a wooden pallet. The text '*Backend' is overlaid on the dirt area.

***Backend**

***Frontend**

The Problem



The Problem



From many, one

Real life examples

Case #1



- P2P betting app
- First MVP release in private beta
- 20 users to be invited
- Last sprint to integrate PayPal
- A few ways of doing so
- Two different ways implemented

Case #2



- Health care app for Australian market
- REST API and SPA on K8S in AWS
- Backend was “faster” then Frontend
- Progress measured in User Stories
- Can't mark User Story as done until front is done
- Later unexpected changes in API

Case #3



- Upcoming project for a financial company
- UI based on the interface already known to users
- Preps to kick off the development phase
- Backend and frontend estimates
- Final estimates for two different architectures

Case #4



- Team extension for an enterprise client
- Two teams - Data Engineering and Data Science
- Shared repo with a common code
- Packaging, distributing and dependency management
- DS team escapes to different repo

Case #5



- AdTech app for aggregating ads traffic
- Legacy project in Django
- New DB and API service introduced just for aggregates
- New API was created in Node.js
- Current team members struggled to contribute

The Process

The Process



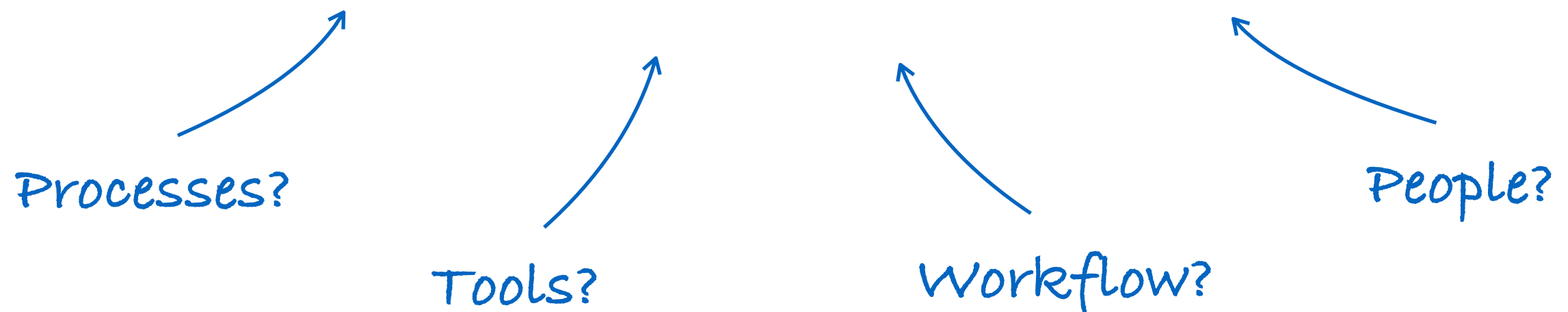
Communication

- Daily catch-ups
- Retrospective sessions
- Planning sessions
- Intensive Slack usage
- Blameless post-mortems

Workflow

- Branch per feature
- Poly repo
- Code reviews
- CI / CD
- Error and uptime tracking

So, what's wrong?



Causes

Causes



- Poor visibility
 - code, changes, architecture, errors
- Building silos
- End-to-end tests away from the app
- Documentation away from the code
- Guidelines different then reality
- Difficult company wide changes
- Everyone as fullstack is an utopia

Poly Repo

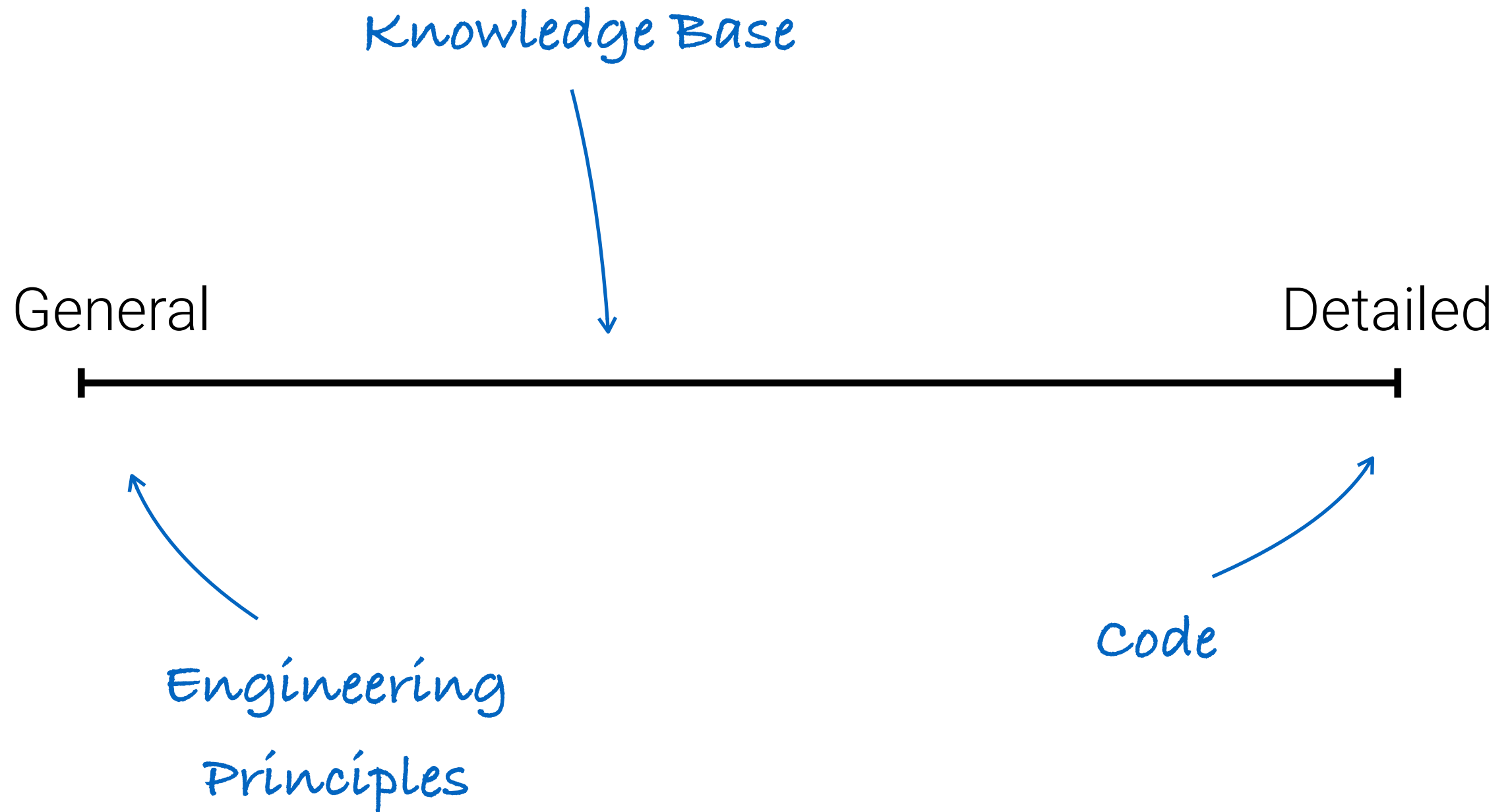
Custom local setup

Knowledge Base

Starting from scratch

REST APIs

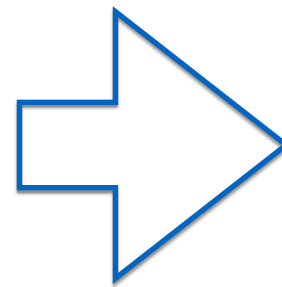
Level of details



Changes



- Poly Repo
- Custom local setup
- Knowledge Base
- Starting from scratch
- REST



- Mono Repo
- Enforcing Docker
- High Level Principles
- Project Boilerplates
- GraphQL

Mono Repo



History

1. mono repo per project
2. poly repo per project
3. mono repo reinvented

Default structure

- backend
- frontend
- infrastructure
- e2e tests
- docs

Mono Repo



Pros

- No boundaries for a team
- Features in one MR
- e2e tests in the same repo
- docs close to the code
- Promoting responsibility

Cons

- Potentially huge MRs
- Long CR sessions
- e2e tests may kill your CI
- Frequent rebase

Disclaimer: Biased

Local dev with Docker



- Not really enforcing, but the only supported
- Containers are awesome
- Not everyone is happy

Local dev with Docker



Pros

- versioning dependencies
- reproducible
- build once, run many times
- spreading devops culture

Cons

- potentially slower development
- Work arounds
- learning curve for juniors

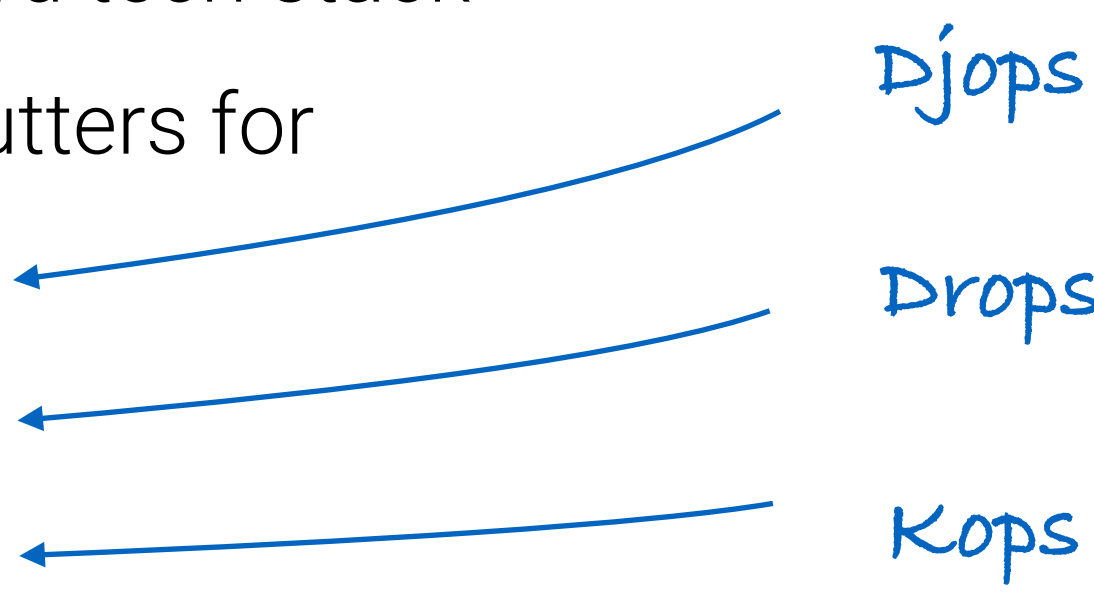
Oh, Hi containers!



Disclaimer: Biased

Project Boilerplates



- Cookiecutter FTW
 - Reasonable defaults
 - Code as a standard tech stack
 - Different cookiecutters for
 - backend
 - frontend
 - infra
 - Recently mono repo for cookiecutters itself
- 
- A hand-drawn diagram in blue ink. On the right side, the words "Djops", "Drops", and "Kops" are written vertically. Three blue arrows originate from the right side of these words and point left towards the sub-items of the fourth list item: "backend", "frontend", and "infra".

Djops



- Kind of light weight django-cookiecutter
- Docker images built and pushed to registry
- Staging and production images based on alpine
- django, celery, redis, postgres, drf, black

Open sourced soon

👉 <https://github.com/sunscrapers/djops>

Kops



- Kubernetes setup with kops tool (the one from Google) on AWS
- Traefik, Prometheus, Loki and Graphana
- Helm and helm secrets
- Helm Chart for djops based app

Engineering Principles



- The software must
 - accomplish its stated goals
 - be maintainable, thus readable
- Right tools to specific problems
- Do not reinvent the wheel
- Chesterton's fence
- KISS, DRY, SOLID, etc
- Kent Beck's design rules

Key Takeaways



- Build shared consciousness of a team
- Retrospect, implement, increment
- Code over guidelines

Resources



- https://github.com/joelparkerhenderson/monorepo_vs_polyrepo#monorepo-scaling-problem
- <https://github.com/cookiecutter/cookiecutter>
- <https://github.com/pydanny/cookiecutter-django>
- <https://pythonspeed.com/articles/base-image-python-docker-images/>
- The book titled "Team of Teams" by General Stanley McChrystal
- [Kent Beck's design rules](#)
- https://en.wikipedia.org/wiki/Wikipedia:Chesterton%27s_fence



sunscrapers

Join the team!

<https://sunscrapers.com/careers/>



sunscrapers

Thank you!

Questions?