Let Developers and Operators Focus on What They Know Best

Rafal Gajdulewicz, Software Engineer

Julie Zhuying Li, Interaction Designer
PRODUCT DEVELOPMENT

Core tasks

- developers
  - writing code
  - testing code
  - observability
PRODUCT DEVELOPMENT

Core tasks

developers
- writing code
- testing code
- observability

operators
- test application
- monitoring
- upgrades, updates
General development and release workflow

code → commit → request deployment to test
General development and release workflow

- code
- commit
- request deployment to test
- what to deploy, where
  - build and publish test
  - deployment done, let devs know
  - deploy artifact to test env
General development and release workflow

1. code
2. commit
3. request deployment to test
   - what to deploy, where
   - build and publish test
4. request deployment to prod
5. verify deployment and correctness
   - deployment done, let devs know
   - deploy artifact to test env

*devs* *ops*
General development and release workflow

1. **Code**
2. **Commit**
3. **Request deployment to test**
4. What to deploy, where
5. Build and publish test
6. Verify deployment and correctness
7. Deployment done, let devs know
8. Deploy artifact to test env
9. What to deploy, where
10. Build and publish prod
11. Deploy artifact to prod env
12. Shipped!
Can we automate some of these steps so that developers and operators can focus on what they know best?
Let’s use some tools that are familiar

version control

CI/CD

cluster registry

GitHub

GitLab

Concourse

CircleCI

Bamboo

Jenkins

codefresh

AND MORE ...
Where CI/CD tools can help us

1. Code
2. Commit
3. Request deployment to test
   - What to deploy, where
4. Build and publish test
5. Verify deployment and correctness
6. Deployment done, let devs know
7. Deploy artifact to test env
8. What to deploy, where
9. Build and publish prod
10. Deploy artifact to prod env
11. Shipped!
Where cluster registry can come in

- code
- commit
- request deployment to test
- what to deploy, where
- build and publish test
- request deployment to prod
- verify deployment and correctness
- deployment done, let devs know
- deploy artifact to test env
- what to deploy, where
- build and publish prod
- deploy artifact to prod env
- shipped!
And you should probably avoid doing this

- code
- commit
- request deployment to test
- what to deploy, where
- build and publish test
- request deployment to prod
- verify deployment and correctness
- deployment done, let devs know
- deploy artifact to test env
- what to deploy, where
- build and publish prod
- deploy artifact to prod env
- shipped!
After these jobs automated, your pipeline gets much shorter

- code → commit
- request deployment to test
- what to deploy, where
- build and publish test
- git, CI, cluster registry deploy to test
- deploy artifact to test env
- deployment done, let devs know
- deploy the artifact to canary, observe metrics / probers, decide if rollout should be continued
- shipped!
- request deployment to prod
- verify deployment and correctness
After optimization

code → commit → git, CI, CR deploy to test → verify deployment and correctness → request deployment to prod → deploy the artifact to prod env

devs

ops

shipped!
Demo
Before

devs = 5 steps
ops = 7 steps

After

devs = 4 steps
ops = 1 steps
Thank you