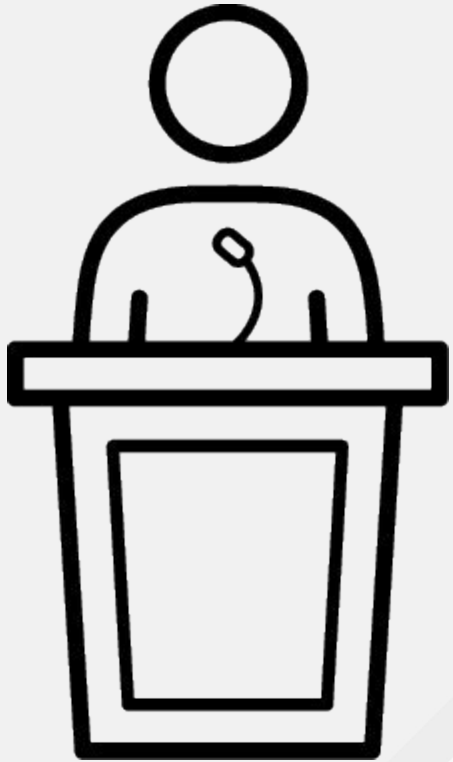
Our Path from Dev and Ops to DevOps

Stanislav Ochotnicky
Business System Analyst, DevOps Automation, Products and Technologies
January 2017

About me

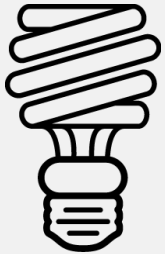
- 7 years at Red Hat
- Worked on packaging for 4 years
- Started Fedora Java SIG
- In love with Git
- Currently working as Business Analyst in DevOps Automation



https://plus.google.com/+StanislavOchotnicky

sochotnicky@redhat.com

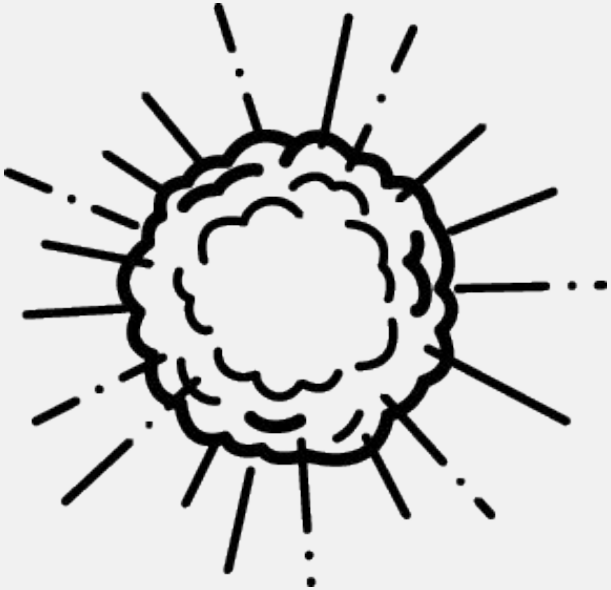
The way we were



Combination of all of the following and more:

- Team going from 100 to 200 people within 3 years
- Old infrastructure supporting critical product releases
- Different provisioning of Dev/QA vs Stage/Prod
- Inconsistent monitoring of services
- Globally distributed teams without common manager
- Unclear escalation paths
- Environment where Dev cannot directly help Ops and vice-versa

NOT DevOps



Examples

- Devs had no access to change deployment or test changes on their own
- Release Engineering was a separate organization
- Services with a single developer working from Australia
- Some critical services had no monitoring
- 50 % of our team was in different timezones from our customers
- Release tooling with no QA and no code review

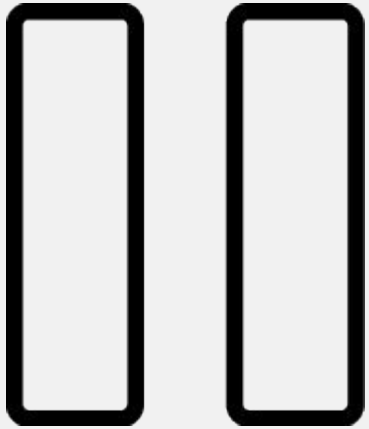
What changed?

Clear mission

Supercharge Red Hat product engineering

Fewer things, but done better

- Stopped doing Big Bang approach
- Started doing Minimum Viable Products
- Avoiding “shadow ops”

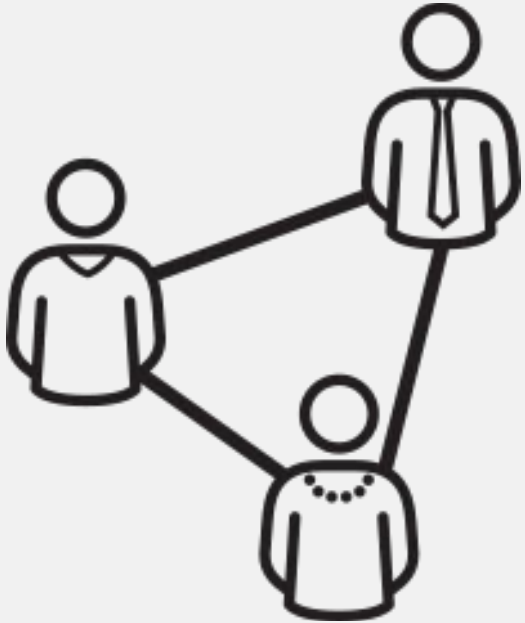


Sunsetting

- We paused some projects
- We merged some projects
- Unwritten rule of 3 devs per project
- Less tooling without reviews

Collaboration on deployments

- Ansible roles developed by Devs
- Guidelines and best practices
- Roles going through code review
- ansible-review to help with validation
- Dev/QE deployed same as Prod/Stage
- Ansible Tower making things better

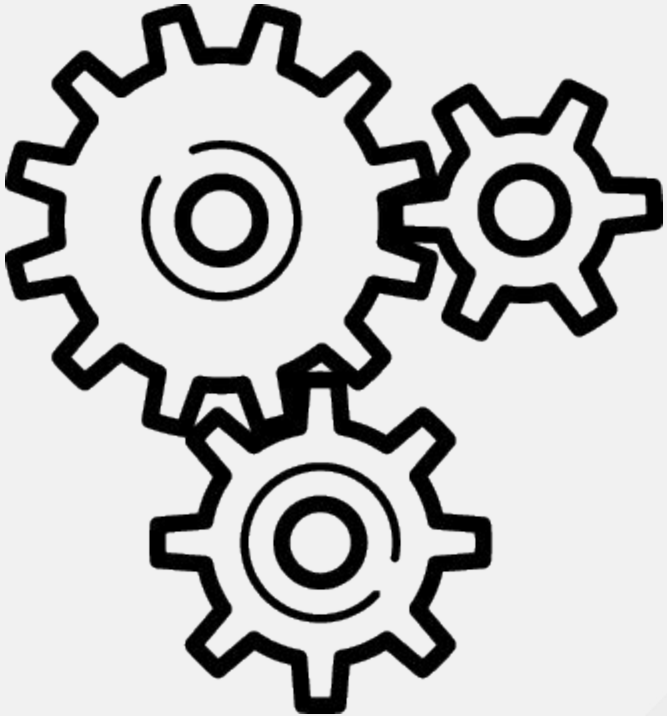


Release engineering joining the family

- Release engineering is now part of a bigger team
- Release tooling getting QE for their tools
- Escalation paths became clearer
- Easier to work together on the same projects

Culture changes

- Little things add up
- Need **champions** not heroes
- Give people access to data
- Run small scale pilots **across** departments

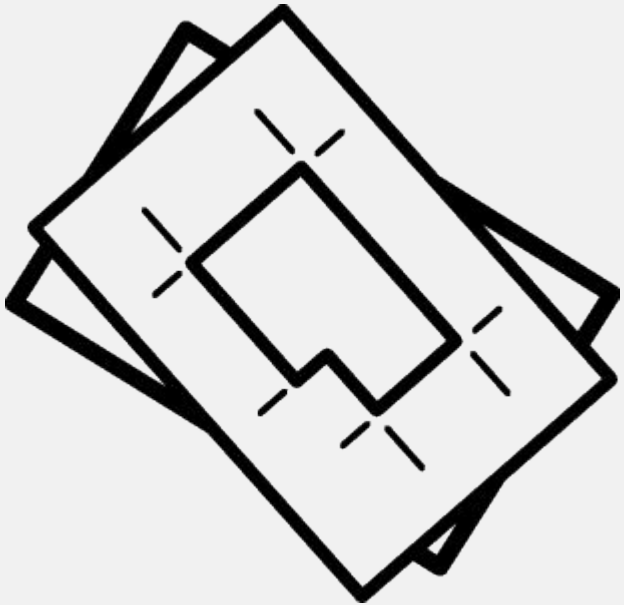


Monitoring, metrics, self-healing

- Services being monitored by Zabbix
- Logs sent to ELK
- Some errors get fixed by automation

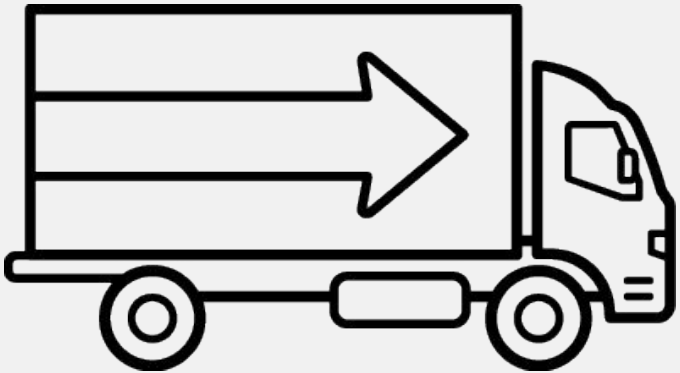
Lessons learned

- Focus
- Choose the right metrics
- Communication is key
- Acquire champions
- People not tools. Except...



What is next?

- We are moving towards container usage for production
- More use of FUSE message bus
- Breaking up bigger services
- Start with continuous delivery for some of the services
- More automation



Questions?

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos