Design Docs and Devops

Michael Hilton Claire Le Goues

October 10, 2019

Administrativia

- Midterm, Oct 17th
- Review in recitation

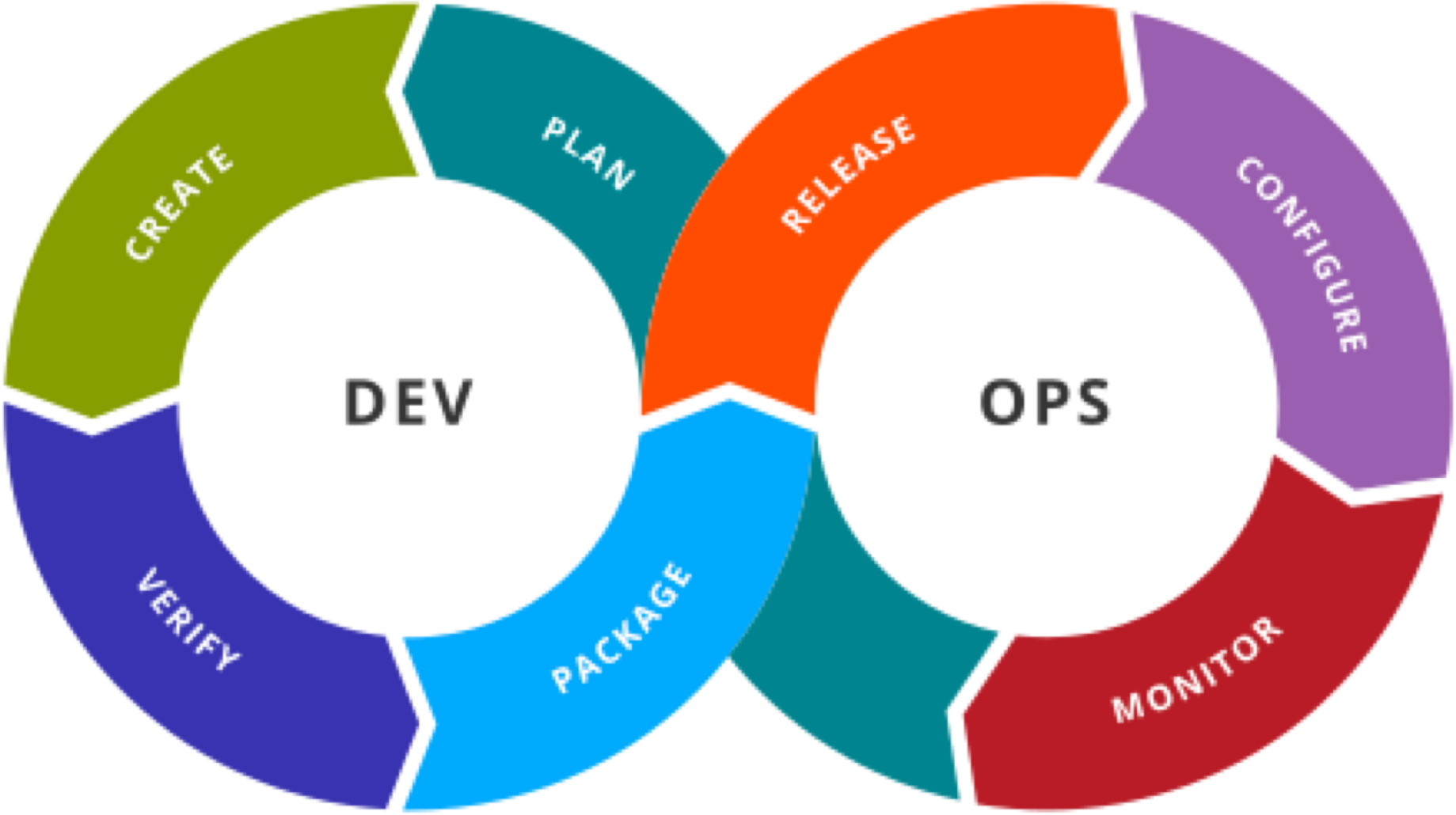
Learning Goals

- Understand history of Microservices
- Reason about tradeoffs of Microservices architectures.
- Practice Building Design Docs

Claire: Design Doc exercise

Thanks to Chris Parnin for some content in these slides

DEVOPS


CREATE
PLAN
RELEASE
CONFIGURE
DEV
OPS
VERIFY
PACKAGE
MONITOR

Continuous Integration /Deployment

- Release several times per day
- Incremental rollout, quick rollback

Travis CI

Quality Goals

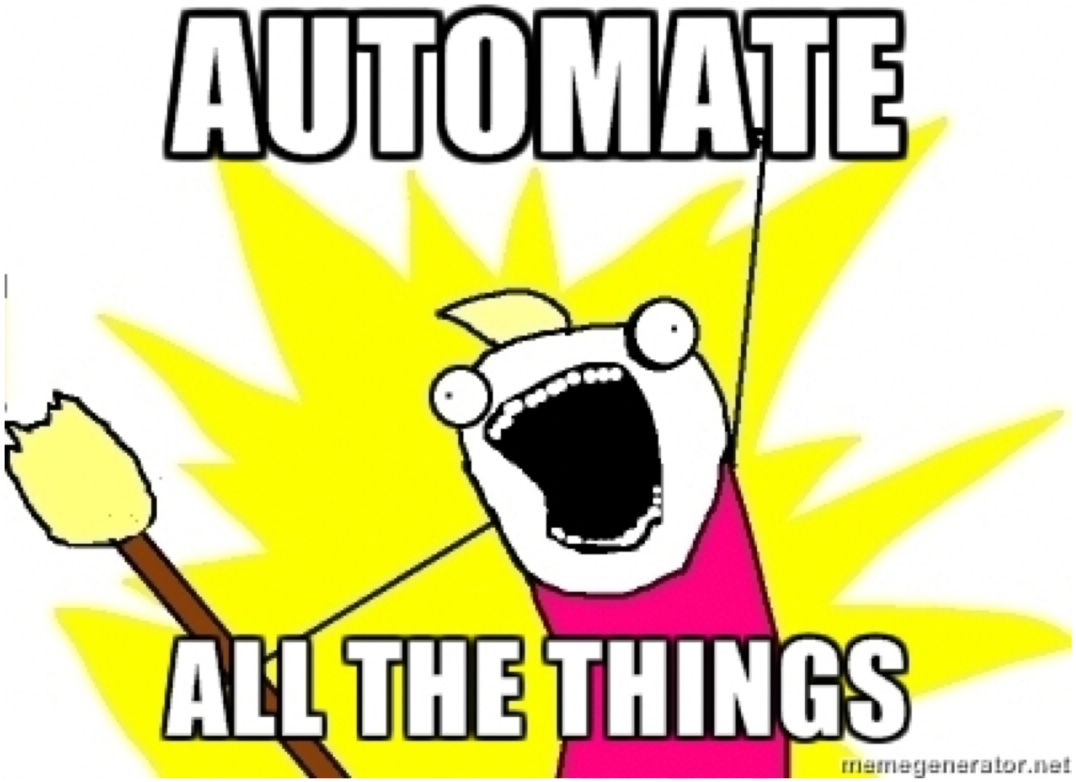
- Rapid releases and feedback (despite large code base)
- Quick onboarding

Infrastructure/Configuration as Code

- Manage configuration files in version control system
- Consistent infrastructure setup for testing, development, and deployment
- Configuration includes ports, target servers and routing, …

- Lightweight virtualization
- Sub-second boot time
- Sharable virtual images with full setup incl. configuration settings
- Used in development and deployment
- Separate docker images for separate services (web server, business logic, database, …)

Automate all the things



INSTALL.SH

```
#!/bin/bash

pip install "$1" &
easy_install "$1" &
brew install "$1" &
npm install "$1" &
yum install "$1" & dnf install "$1" &
docker run "$1" &
pkg install "$1" &
apt-get install "$1" &
sudo apt-get install "$1" &
steamcmd +app_update "$1" validate &
git clone https://github.com/"$1"/"$1" &
cd "$1";./configure;make;make install &
curl "$1" | bash &
```

Two sides to DevOps

- **Operation-centric:**
  - Manage inventory of servers automatically
  - Provisioned, configured automatically Monitoring, analysis, automation of operations
- **Developer centric:**
  - Continuous deployment
  - Push code to production through pipeline

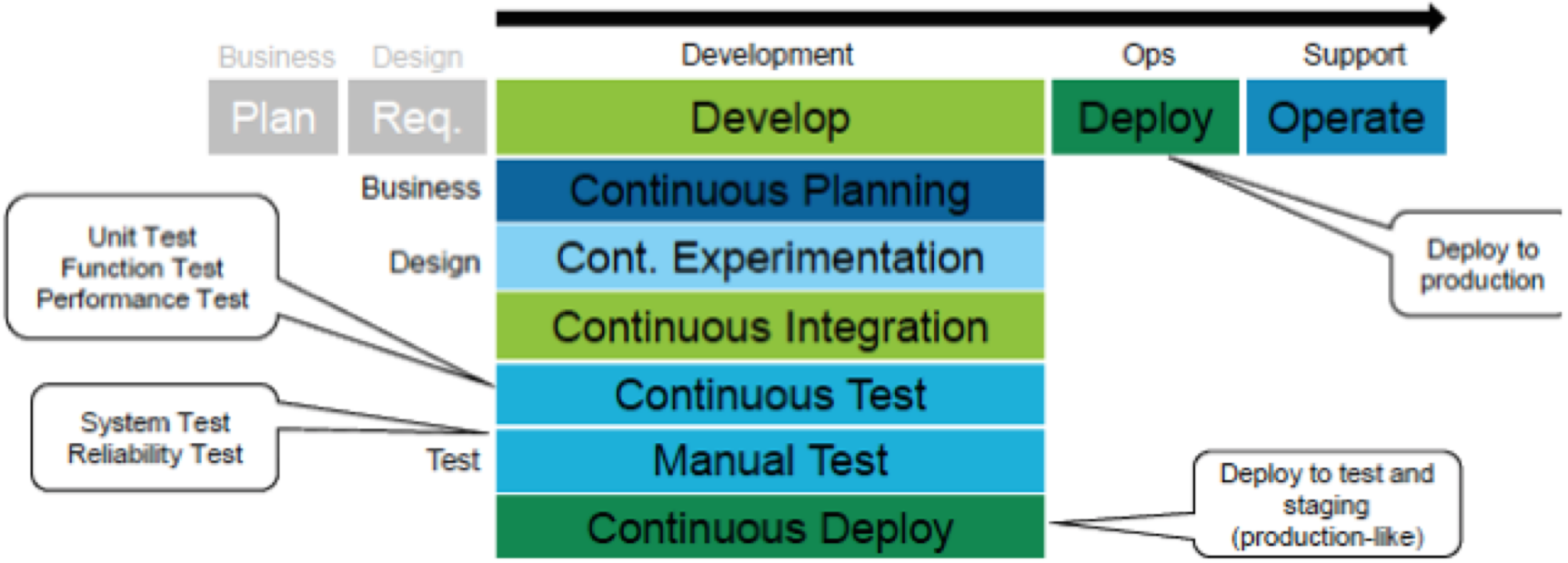
PRINCIPLES, WITH A LITTLE BIT OF HISTORY…

Nightly Build

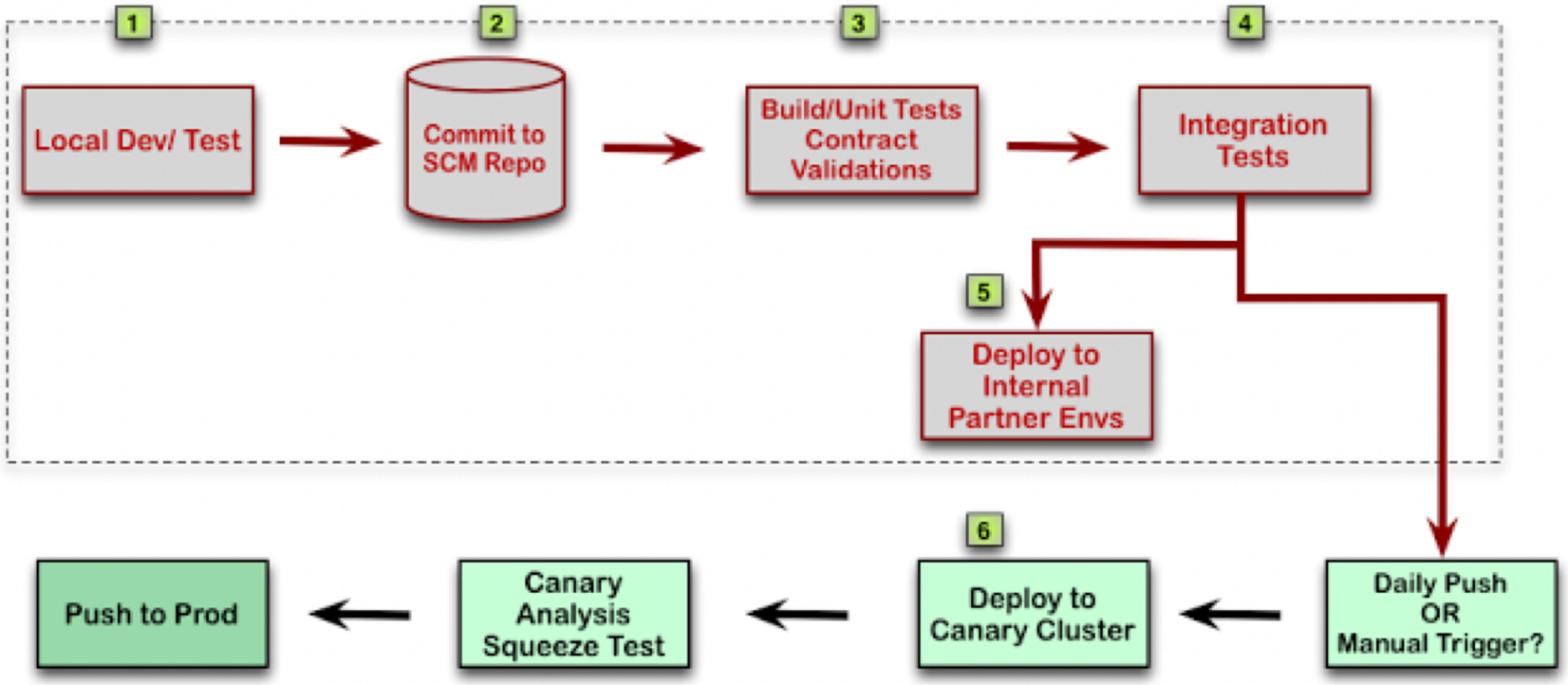
- Build code and run smoke test (Microsoft 1995)

- Benefits
  - it minimizes integration risk.
  - It reduces the risk of low quality
  - It supports easier defect diagnosis It improves morale

Continuous * (Perpetual Development)



Example Deployment Pipeline



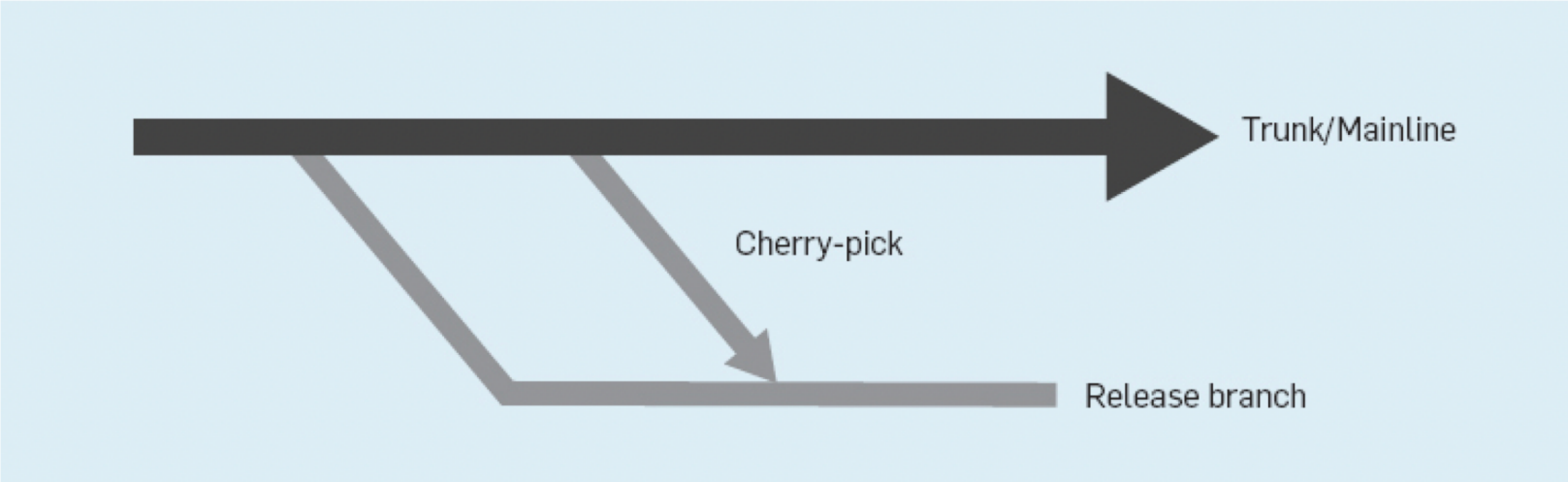
Principle: Fast to Deploy, Slow to Release

- Chuck Rossi at Facebook: *"Get your s*** in, fix it in production"*

Dark Launches at Instagram

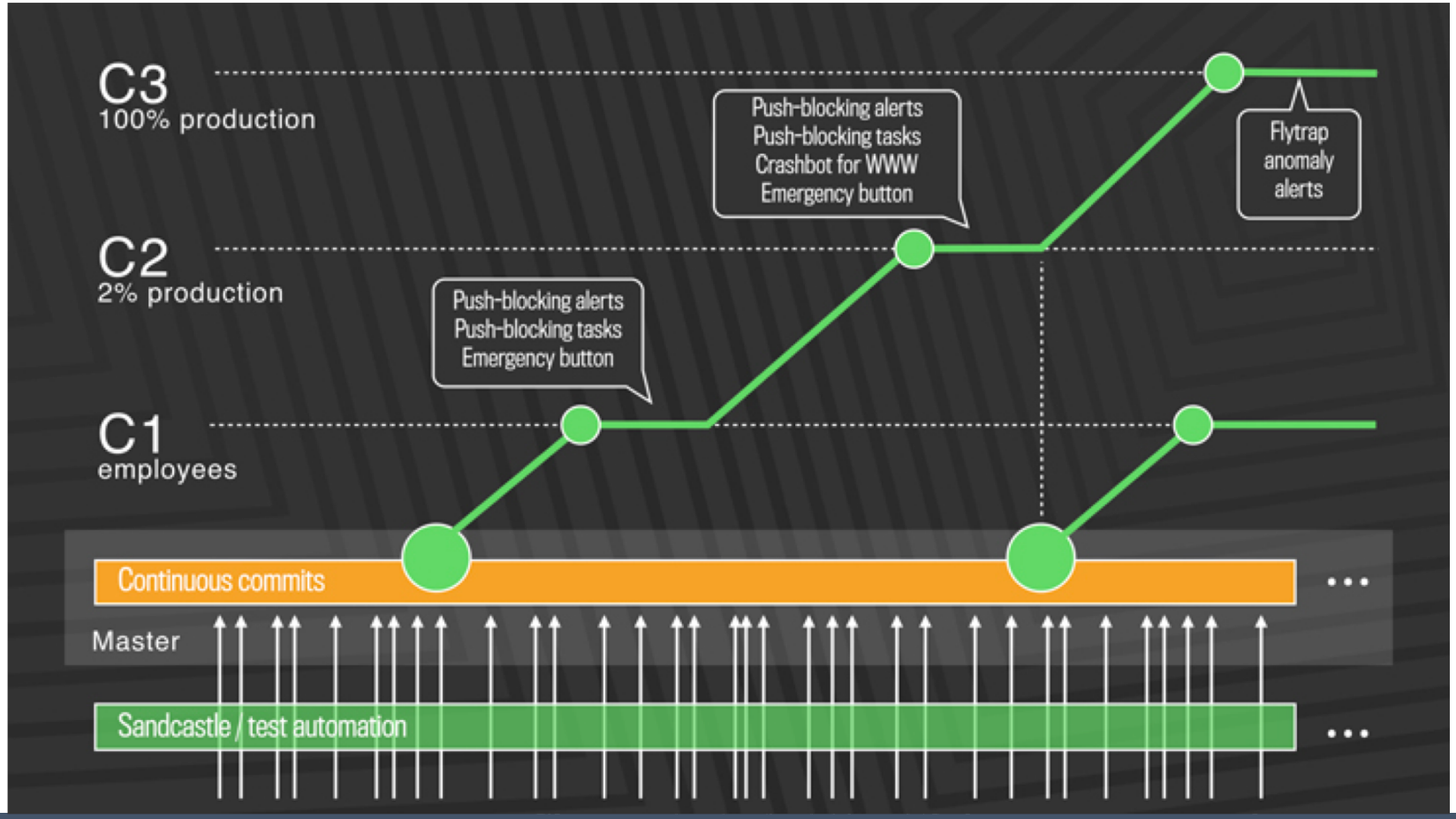
- **Early**: Integrate as soon as possible. Find bugs early. Code can run in production about 6 months before being publicly announced.
- **Often**: Reduce friction. Try things out. See what works. Push small changes just to gather metrics, feasibility testing. Large changes just slow down the team. Do dark launches, to see what performance is in production, can scale up and down. *"Shadow infrastructure" is too expensive, just do in production.*
- **Incremental**: Deploy in increments. Contain risk. Pinpoint issues.

Facebook process (until 2016)



- Release is cut Sunday 6pm
- Stabilize until Tuesday, canaries, release. Tuesday push is 12,000 diffs.
- Cherry pick: Push 3 times a day (Wed-Fri) 300-700 cherry picks / day.

Facebook quasi-continuous release



Rapid Release/Mozilla

If deployment requires on-prem deployment, say a web browser

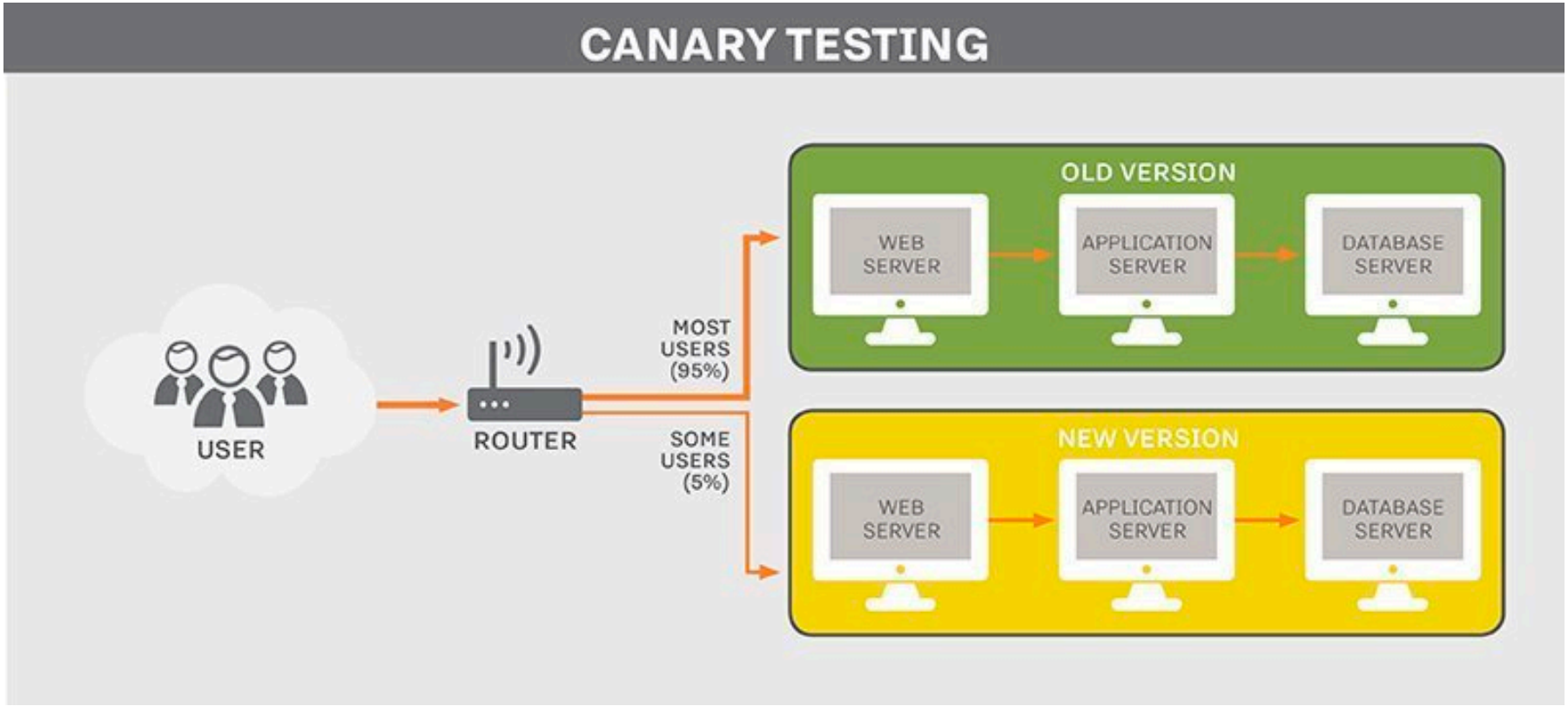
- There are three channels: Alpha, Beta, Release Candidate
- Code flows every 2 weeks to next channel, unless fast tracked by release engineer.
- Involve corporate customer specific testing in testing (Practice also used by IBM, Redhat)

Ring Deployment: Microsoft

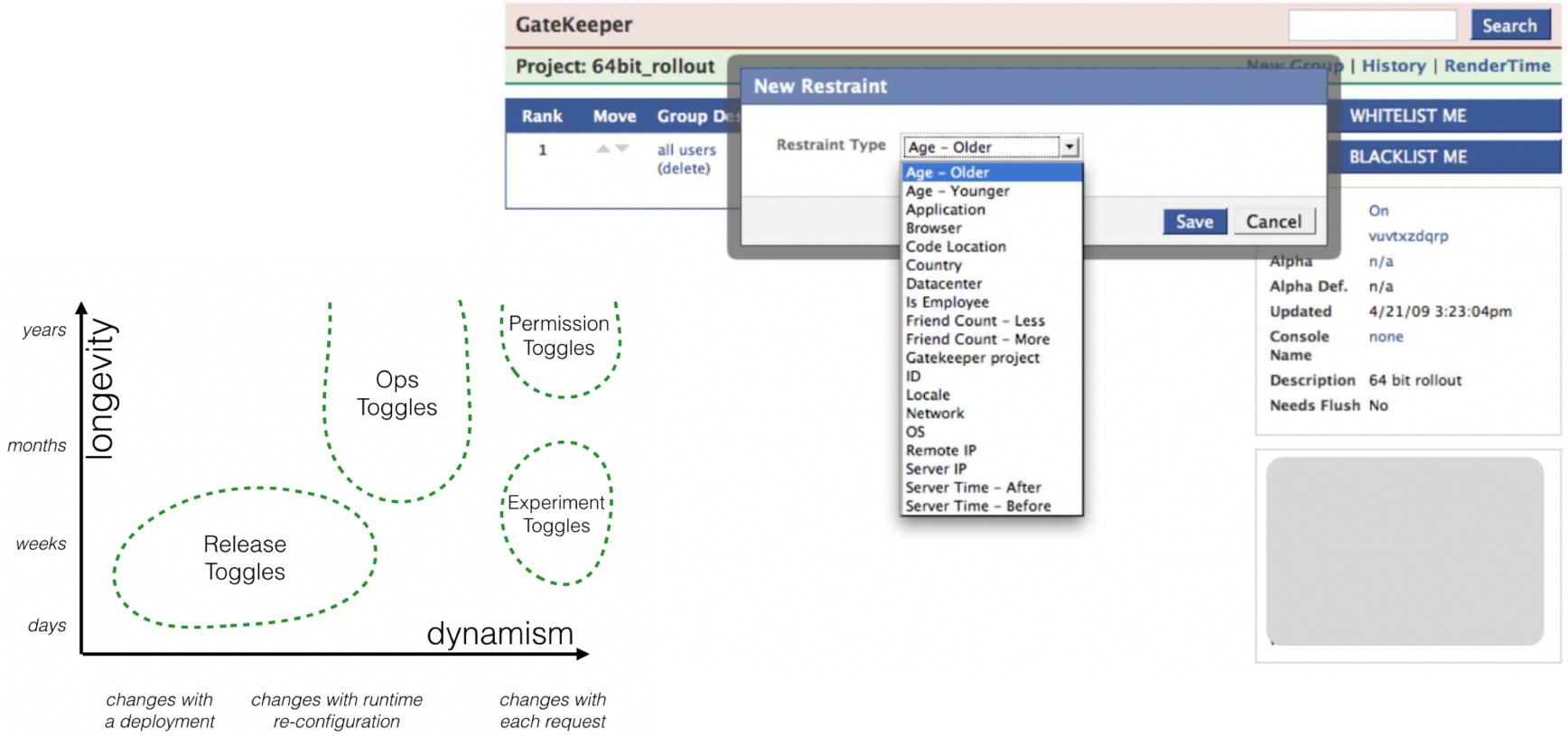
- Commits flow out to rings, deflight if issue.
- For example:
  - Ring 0 => Team
  - Ring 1 => Dogfood
  - Ring 2 => Beta
  - Ring 3 => Many
  - Ring 4 => All

PRINCIPLE: EVERY FEATURE IS AN EXPERIMENT

Canary testing



Controlling feature flags



Netflix

- 60,000 configuration changes a day. 4000 commits a day.
- Every commit creates an Amazon Machine Imagine (AMI).
- AMI is automated deployed to a new RED/BLACK cluster.
- Have automated canary analysis, if okay, switch to new version, if not, **rollback** commit.

Who Does Operations?