ECE444: Software Engineering

DevOps

Shurui Zhou

The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

Administrivia

- About CoFM

  Updated: 7.5 pt

  (5) Group presentation

  (1.5) Individual Peer evaluation

  (1) Presentation Feedback

  Milestone 4.5: (2pt) 2-week report (Individual)

  Milestone 5: (35.5pt) [see rubrics in Files/Project1/Milestone5]

  Group report 8pt

  (4) final delivery

  (4) Architecture report [rubric is coming soon]

  Individual 27.5pt

  (2.5) Peer evaluation

  (8) Architecture report (individual)

  (7) Reflection report

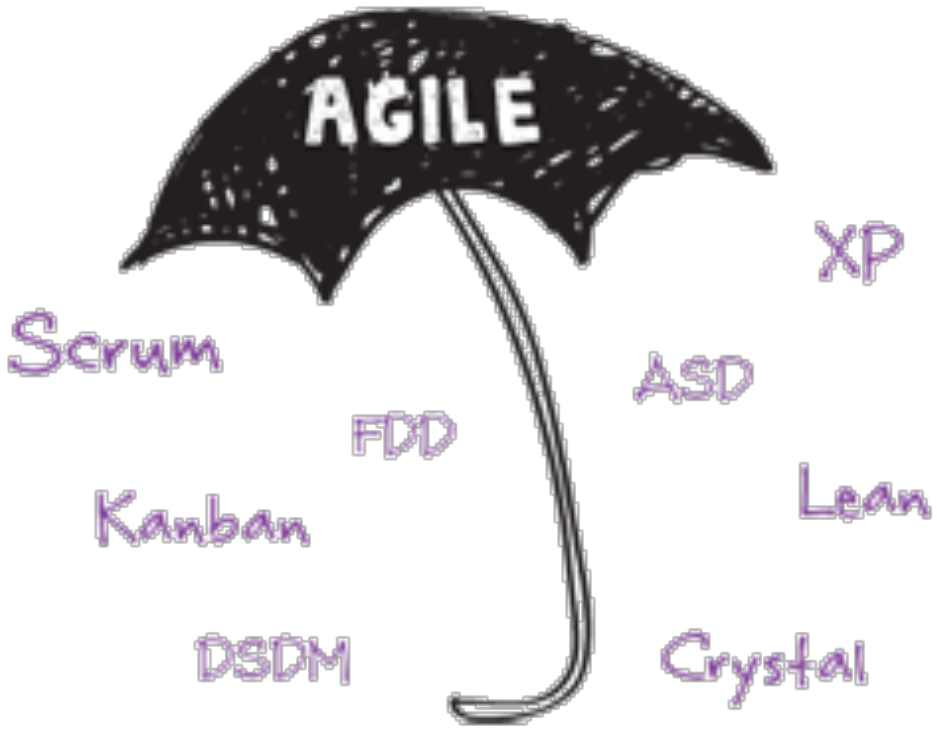
  (10) Code Quality, process, practices

Administrivia

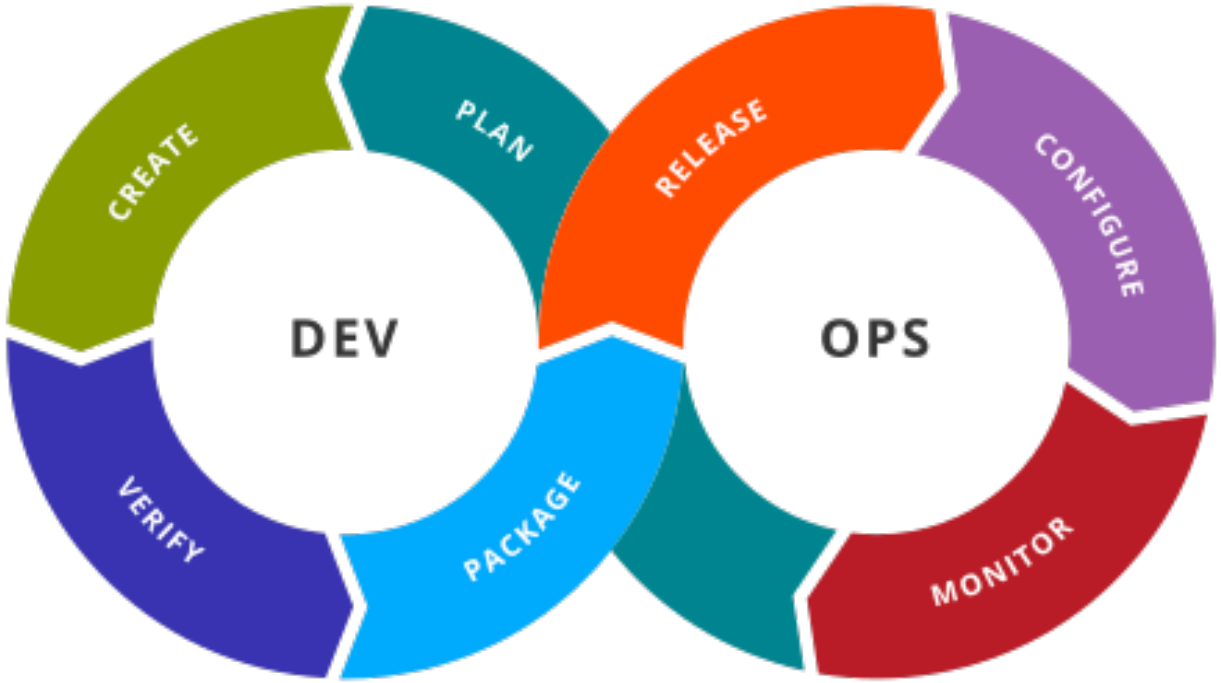
- Lab this week
  - Deploy your Chef Co-Pilot app on Heroku (follow up on Lab 6&7)
- Deadline for the Milestone 5
  - to adhere to the CoFM guidelines by having a final assessment for ECE444, we move the final delivery deadline to the exam period assigned (Dec 19)
  - No need to keep updating your project after **11/18** 11:59pm EST. We don't evaluate your code/report changed after 11/18

Learning Goals

- Understand DevOps
- Understand CI/CD
- Integrate DevOps into your web application



Developers + Operators = DevOps



What Are the Challenges DevOps Solves?

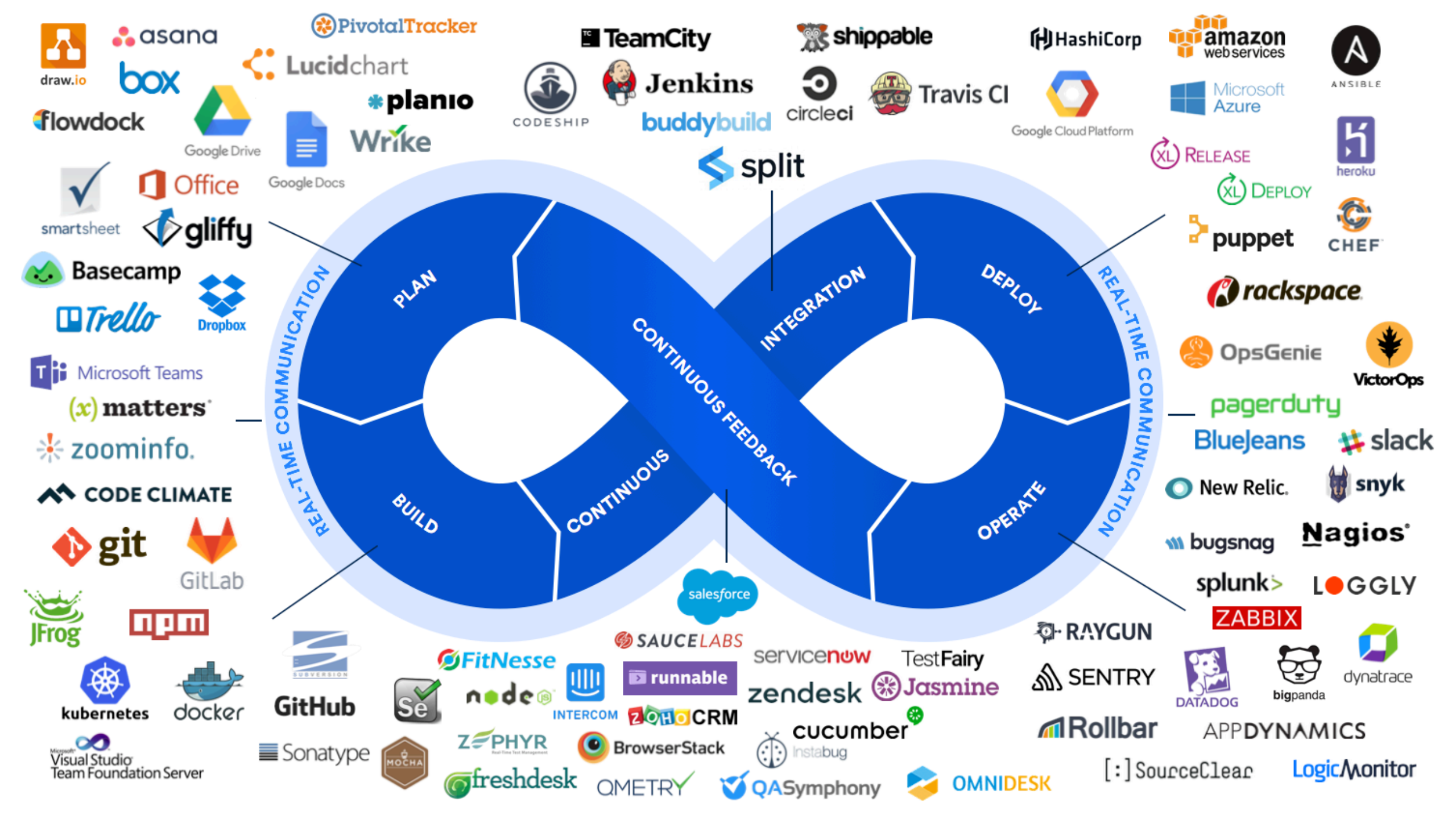
- Dev is often unaware of QA and Ops roadblocks that prevent the program from working as anticipated.
- QA and Ops are typically working across many features and have little context of the business purpose and value of the software.
- Each group has opposing goals that can lead to inefficiency and finger pointing when something goes wrong.

How often different companies deploy to the release environment

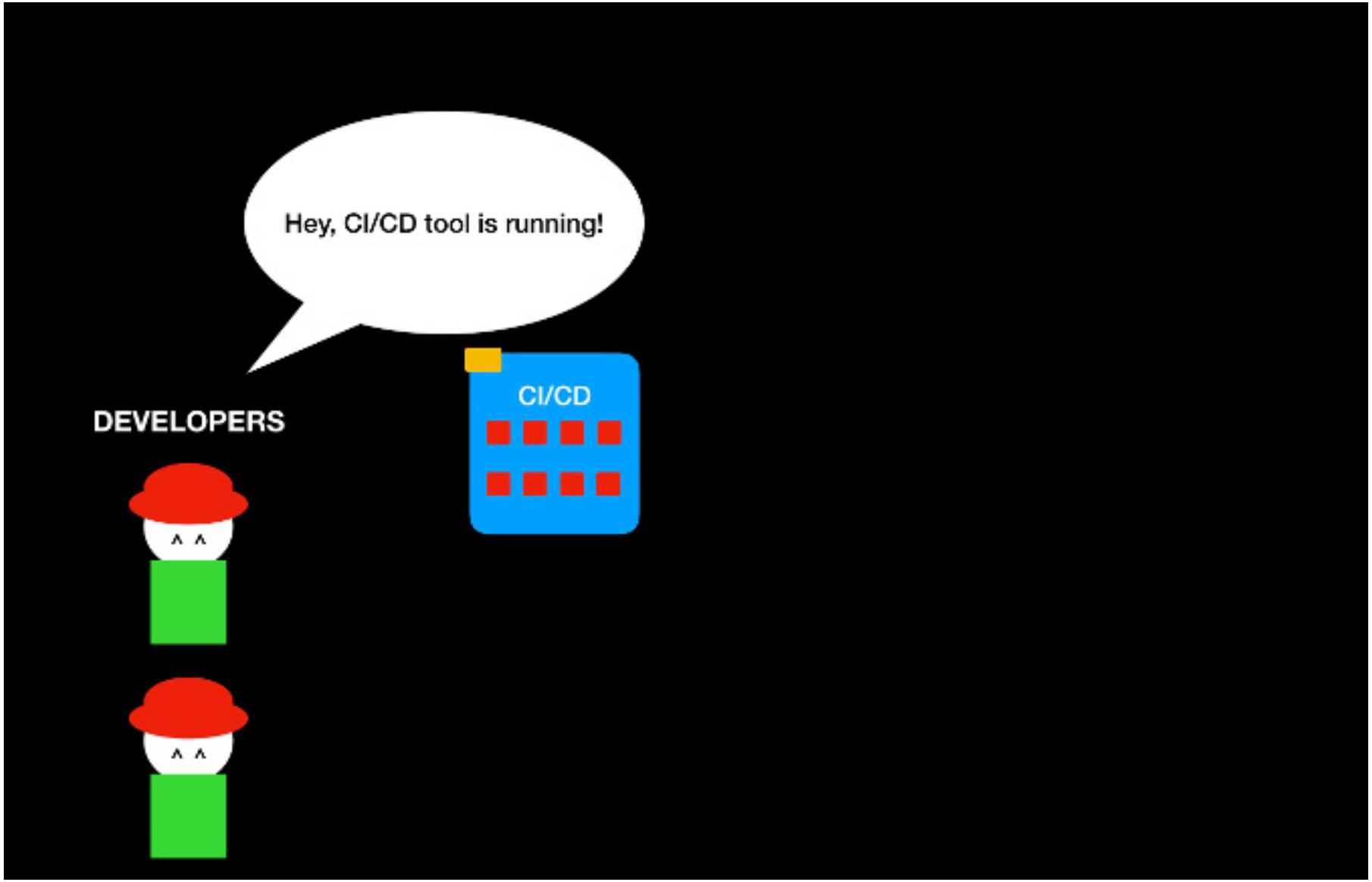
Company	Deployment Frequency
Amazon	23,000 per day
Google	5,500 per day
Netflix	500 per day
Facebook	1 per day
Twitter	3 per week
Typical enterprise	1 every 9 months

Goal of DevOps

- Improve deployment frequency
- Achieve faster time to market
- Lower failure rate of new releases
- Shorten lead time between fixes
- Improve mean time to recovery


PLAN
CONTINUOUS
BUILD
CONTINUOUS FEEDBACK
INTEGRATION
DEPLOY
OPERATE
REAL-TIME COMMUNICATION
REAL-TIME COMMUNICATION
draw.io
asana
PivotalTracker
Lucidchart
box
planio
flowdock
Google Drive
Google Docs
Wrike
Office
smartsheet
gliffy
Basecamp
Trello
Dropbox
Microsoft Teams
(x) matters
zoominfo.
CODE CLIMATE
git
GitLab
JFrog
npm
kubernetes
docker
Visual Studio Team Foundation Server
GitHub
Sonatype
CODESHIP
TeamCity
Jenkins
buddybuild
shippable
circleci
Travis CI
split
salesforce
SAUCELABS
FitNesse
node
runnable
INTERCOM
zohoCRM
ZEPHYR
BrowserStack
freshdesk
QMETRY
servicenow
TestFairy
zendesk
Jasmine
cucumber
Instabug
QASymphony
OMNIDESK
HashiCorp
amazon web services
Microsoft Azure
Google Cloud Platform
ANSIBLE
RELEASE
heroku
DEPLOY
puppet
CHEF
rackspace
OpsGenie
VictorOps
pagerduty
BlueJeans
slack
New Relic.
snyk
bugsnag
Nagios
splunk>
LOGGLY
ZABBIX
RAYGUN
SENTRY
DATADOG
bigpanda
dynatrace
Rollbar
APPDYNAMICS
[:] SourceClear
LogicMonitor

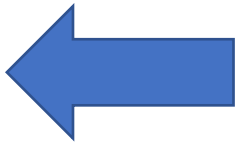
CI/CD



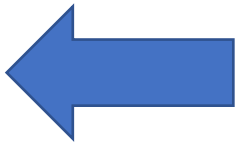
https://opensource.com/article/19/4/devops-pipeline

Continuous Integration

- Merging in small code changes frequently



Continuous Delivery



- Add additional automation and testing, get the code nearly ready to deploy with almost no human intervention

Continuous Deployment

- Deploying all the way into production without any human intervention.

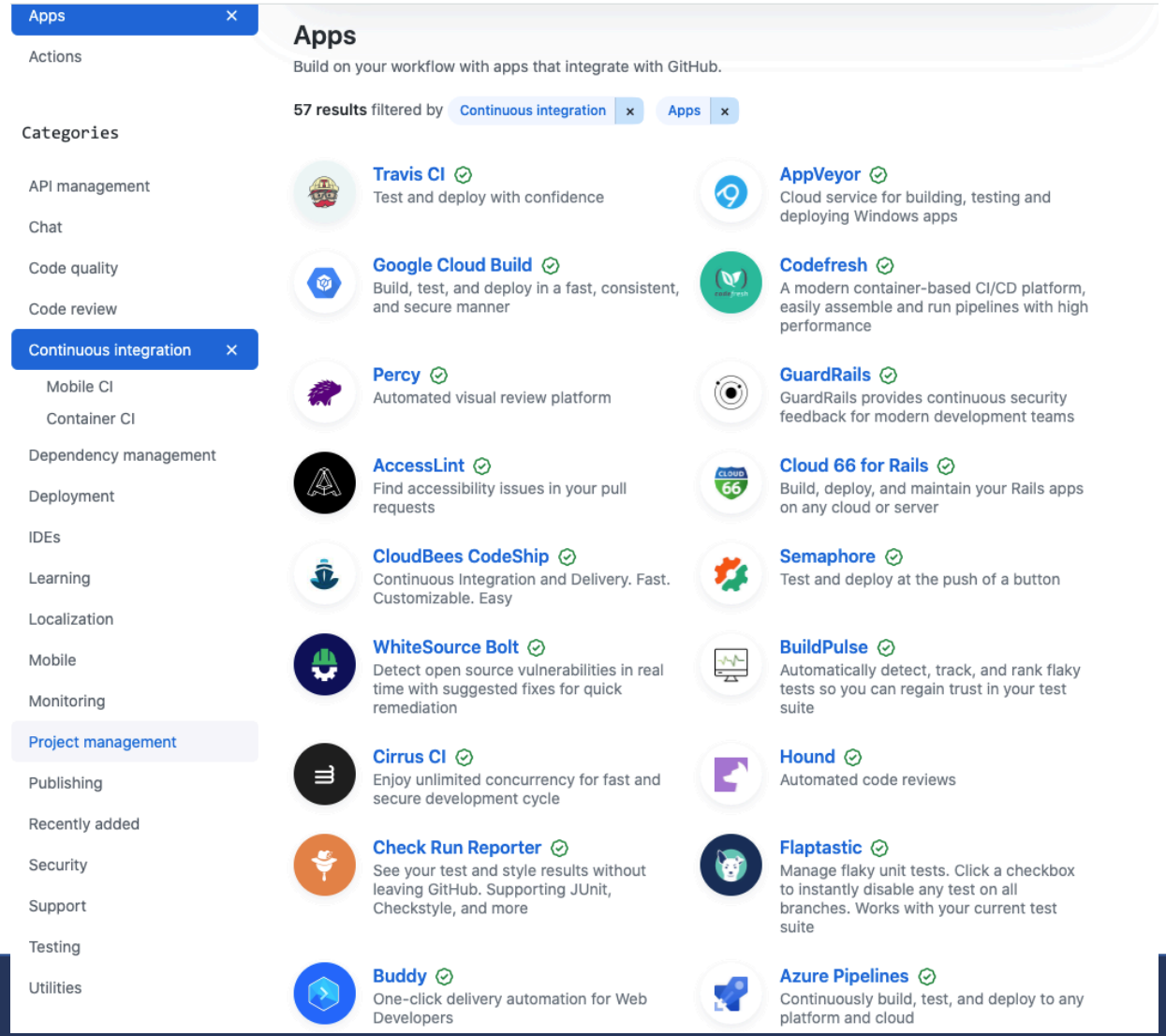
Continuous Integration

- Quickly integrating newly developed code with the main body of code that is to be released

Travis CI

Continuous Integration

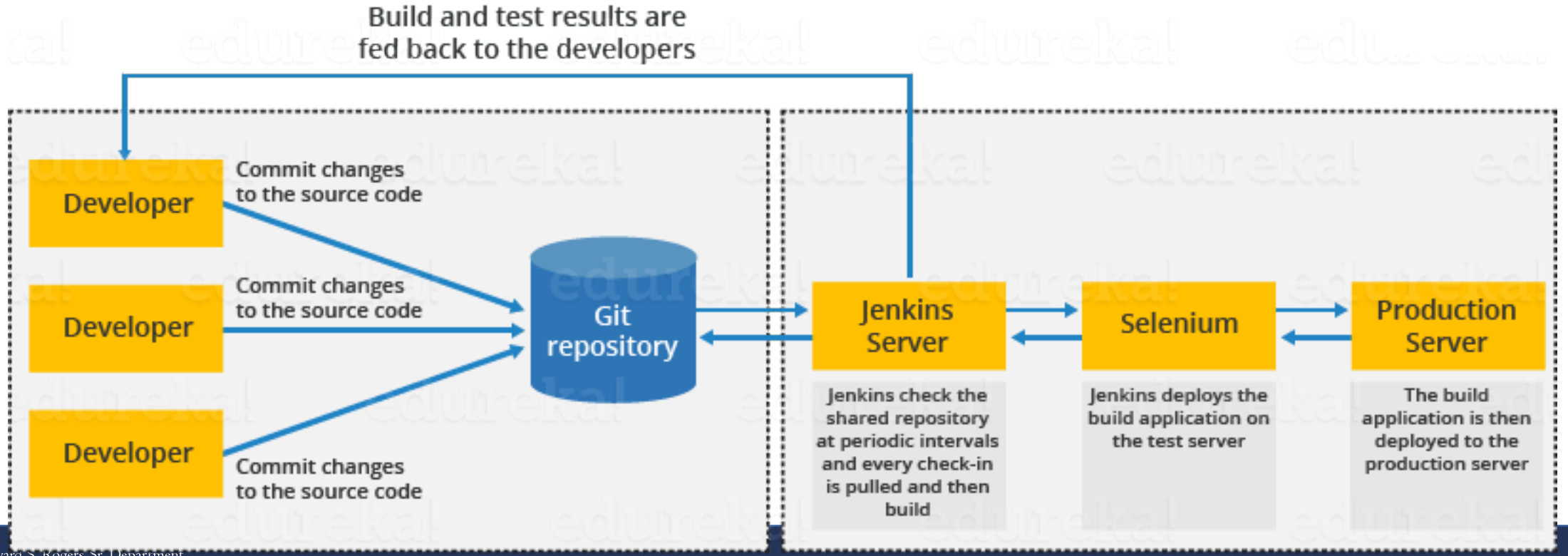
https://martinfowler.com/articles/continuousIntegration.html



Tools - Continuous Integration

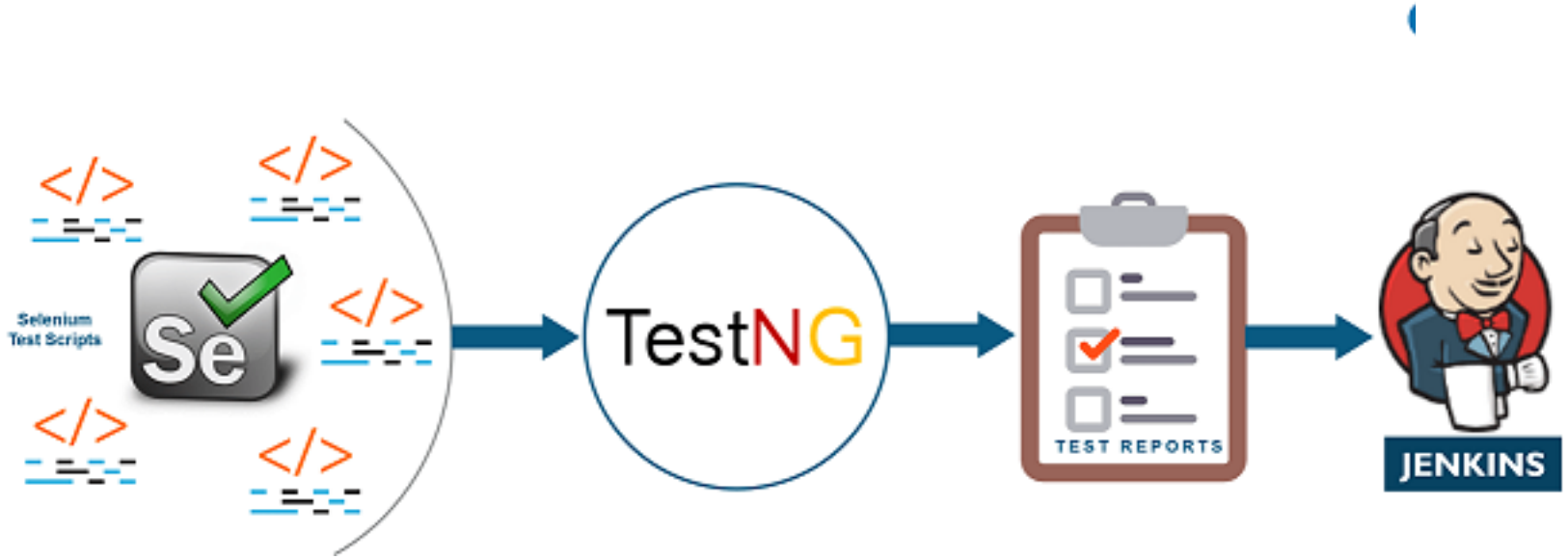
Hudson Jenkins

- Quickly integrating newly developed code with the main body of code that is to be released

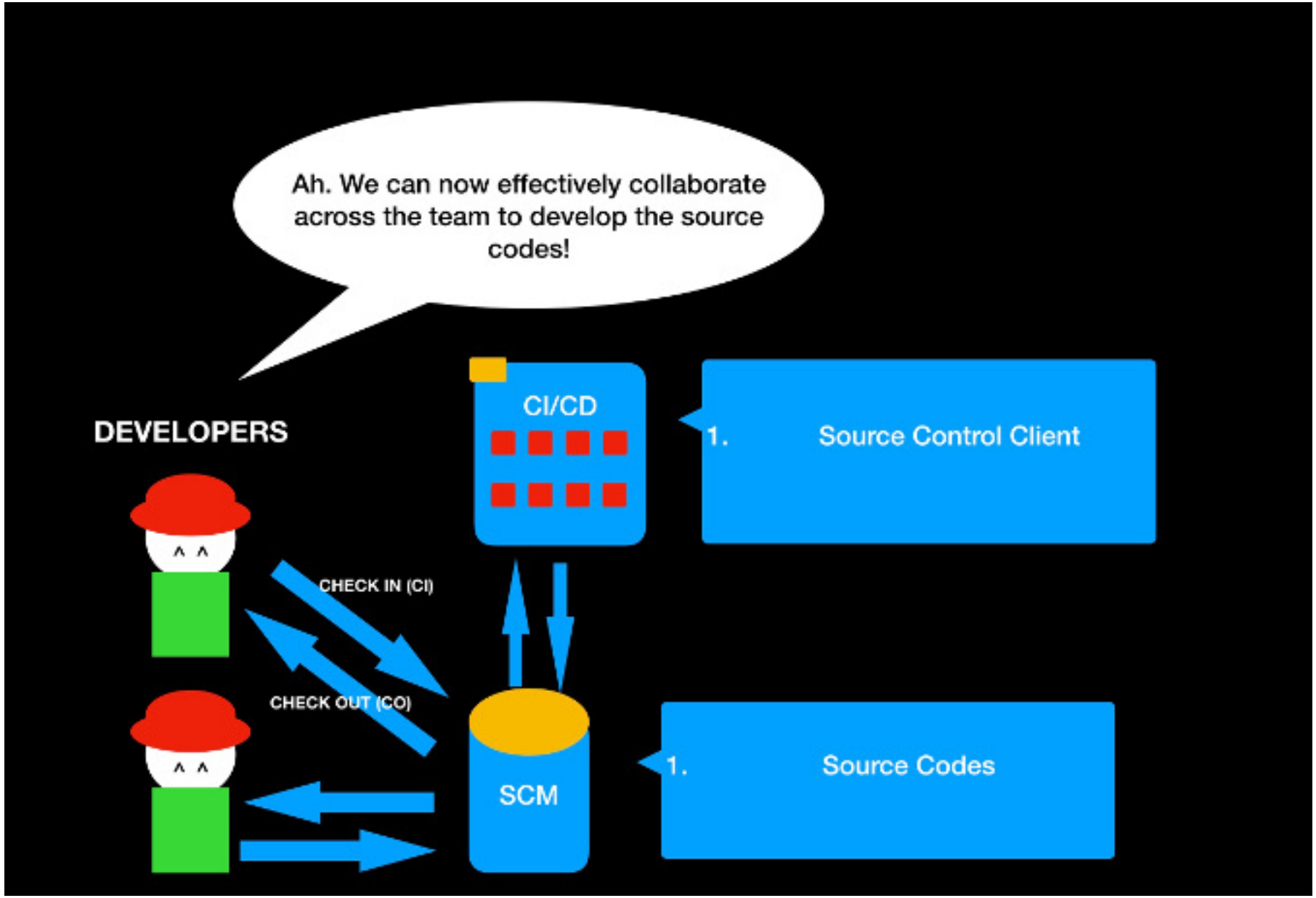


Continuous Testing

- **Selenium**

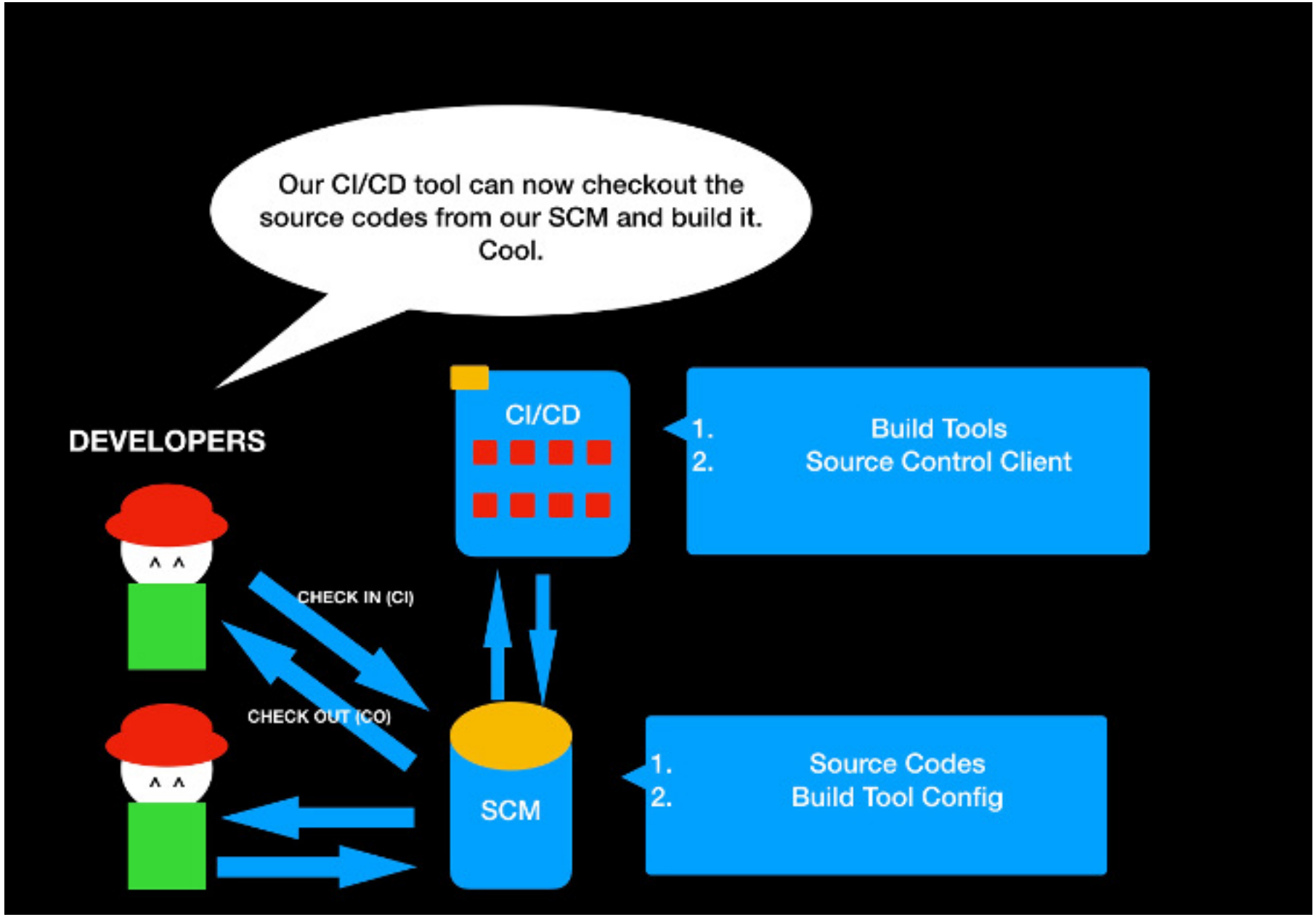


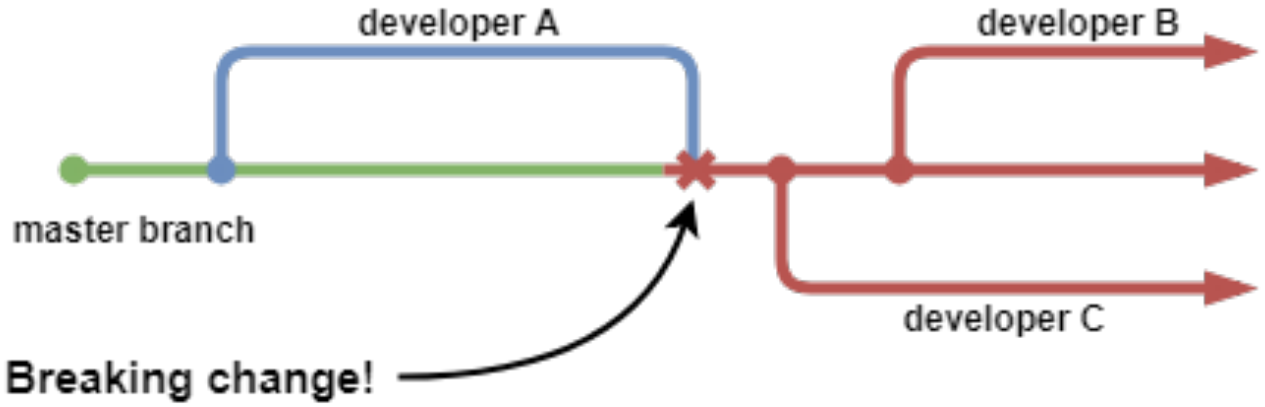
Version Control



SCM-Source Control Mgmt

Build


Our CI/CD tool can now checkout the source codes from our SCM and build it. Cool.
DEVELOPERS
CI/CD
1. Build Tools
2. Source Control Client
CHECK IN (CI)
CHECK OUT (CO)
SCM
1. Source Codes
2. Build Tool Config



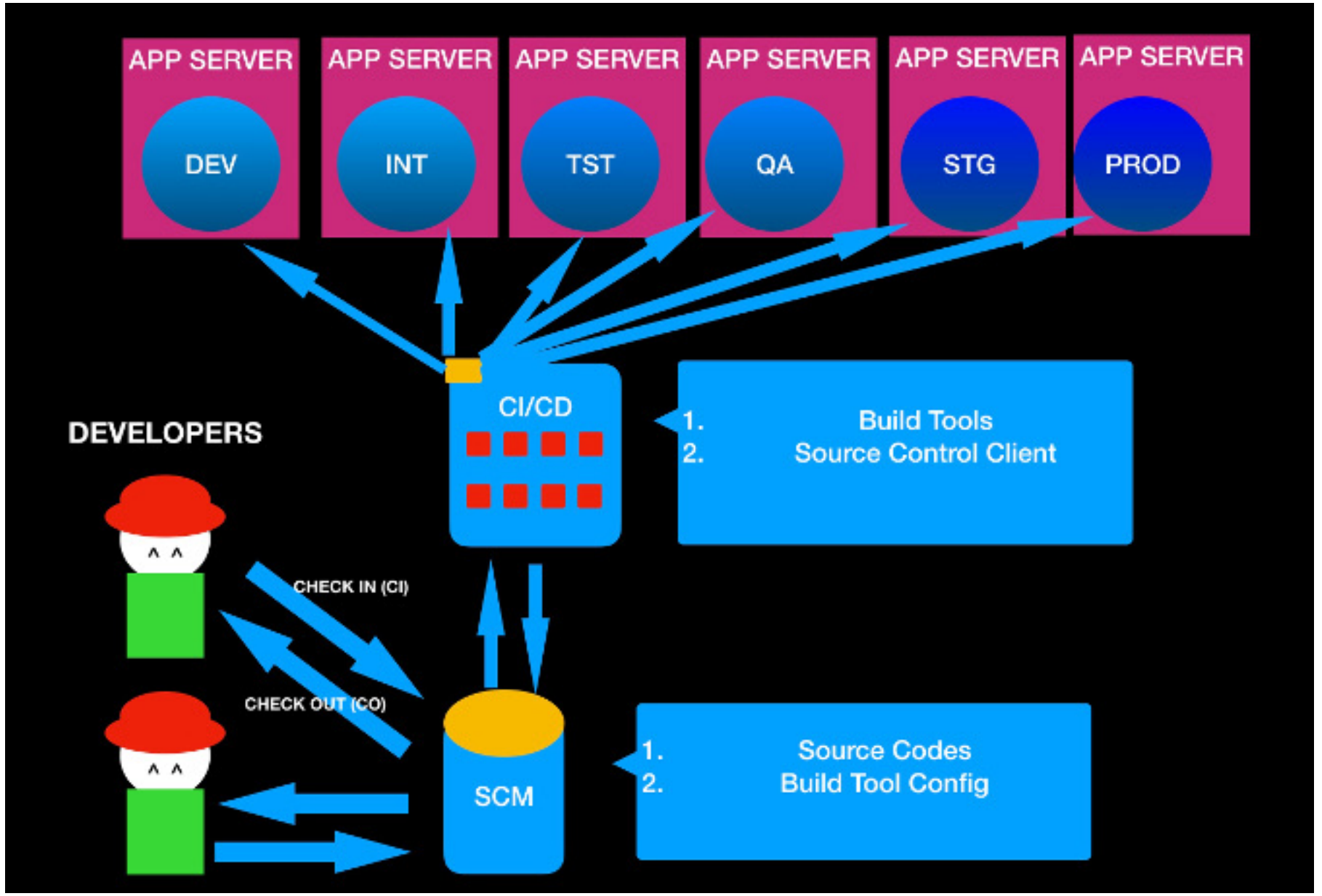
I will not break the build.
I will not break the build.
I will not break the build.
I will not break the build.
I will not break the build.
I will not break the build.
I will not break the build.
I will not break the build.
I will not break the build.

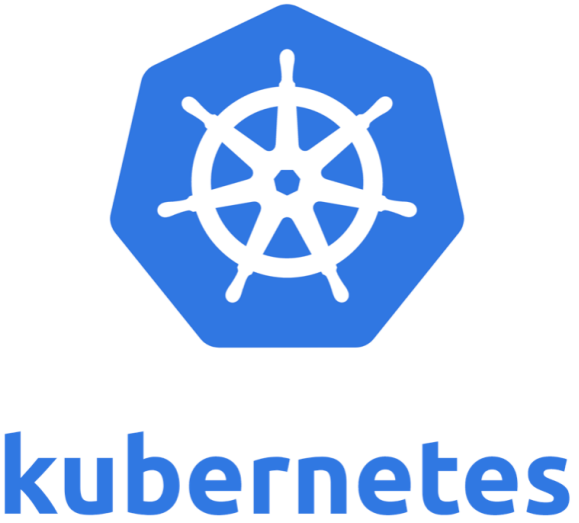
CODESMACK

Brian the Build Bunny

http://www.woodwardweb.com/gadgets/000434.html

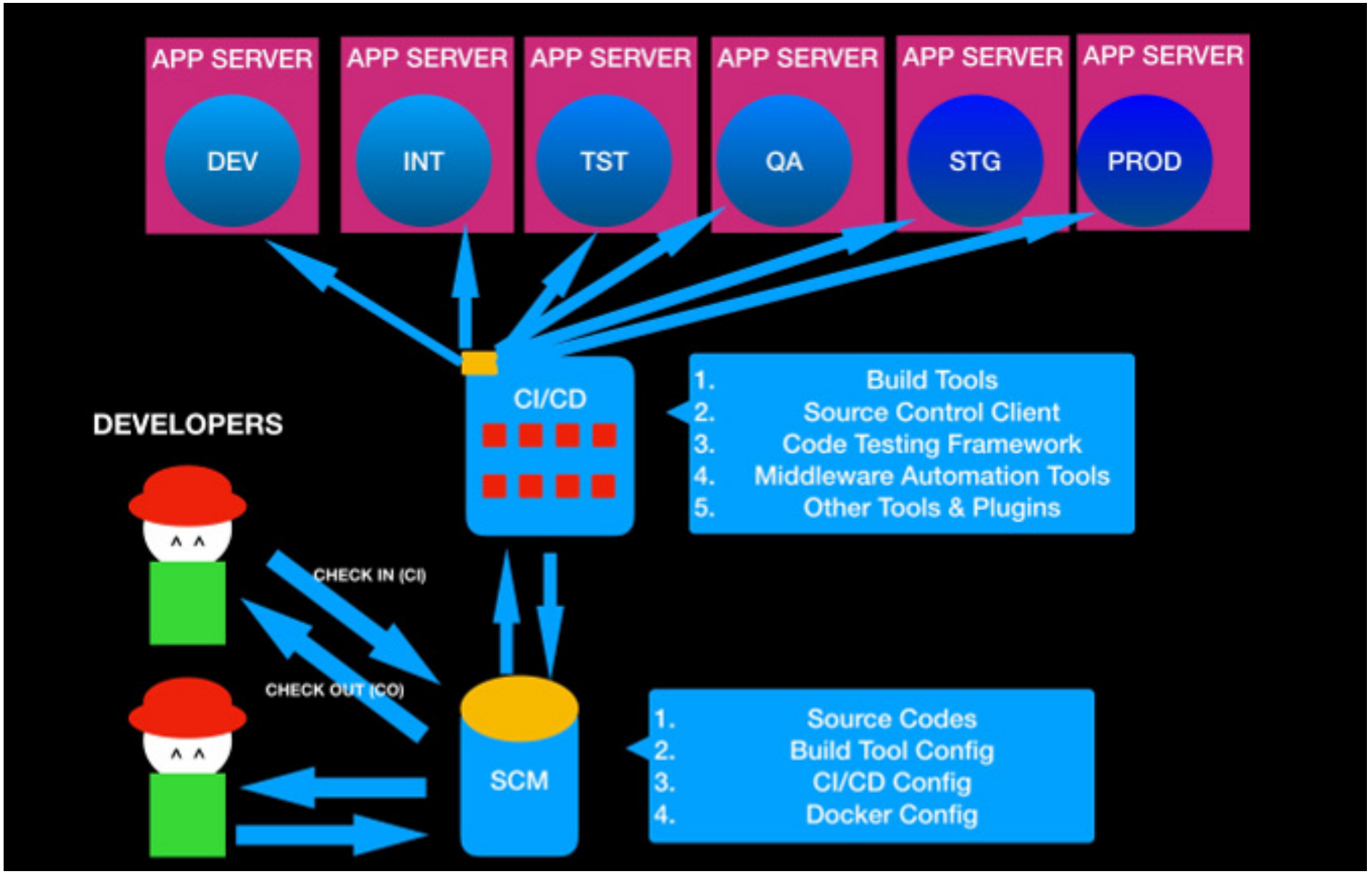
Web app server





- Lightweight virtualization
- Separate docker images for separate services (web server, business logic, database, …)

Automated Testing



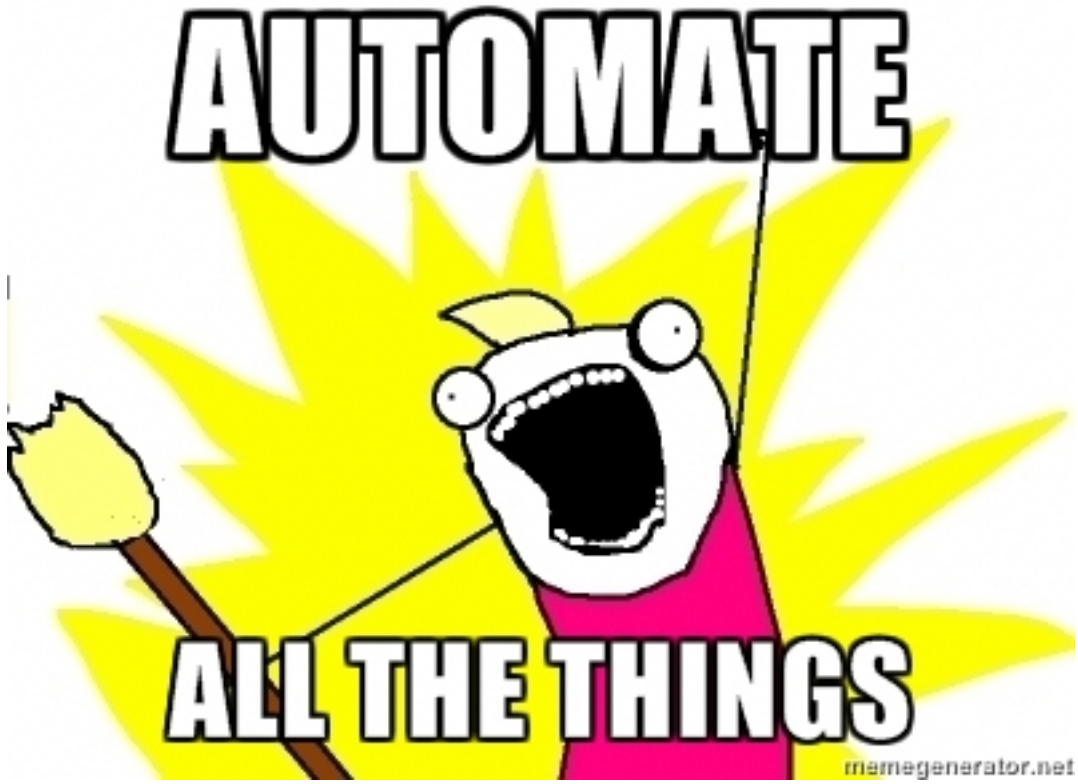
Infrastructure/Configuration as Code (IAC/CAC)

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

Jenkins Configuration as Code

The ‘as code’ paradigm is about being able to reproduce and/or restore a full environment within minutes based on recipes and automation, managed as code.

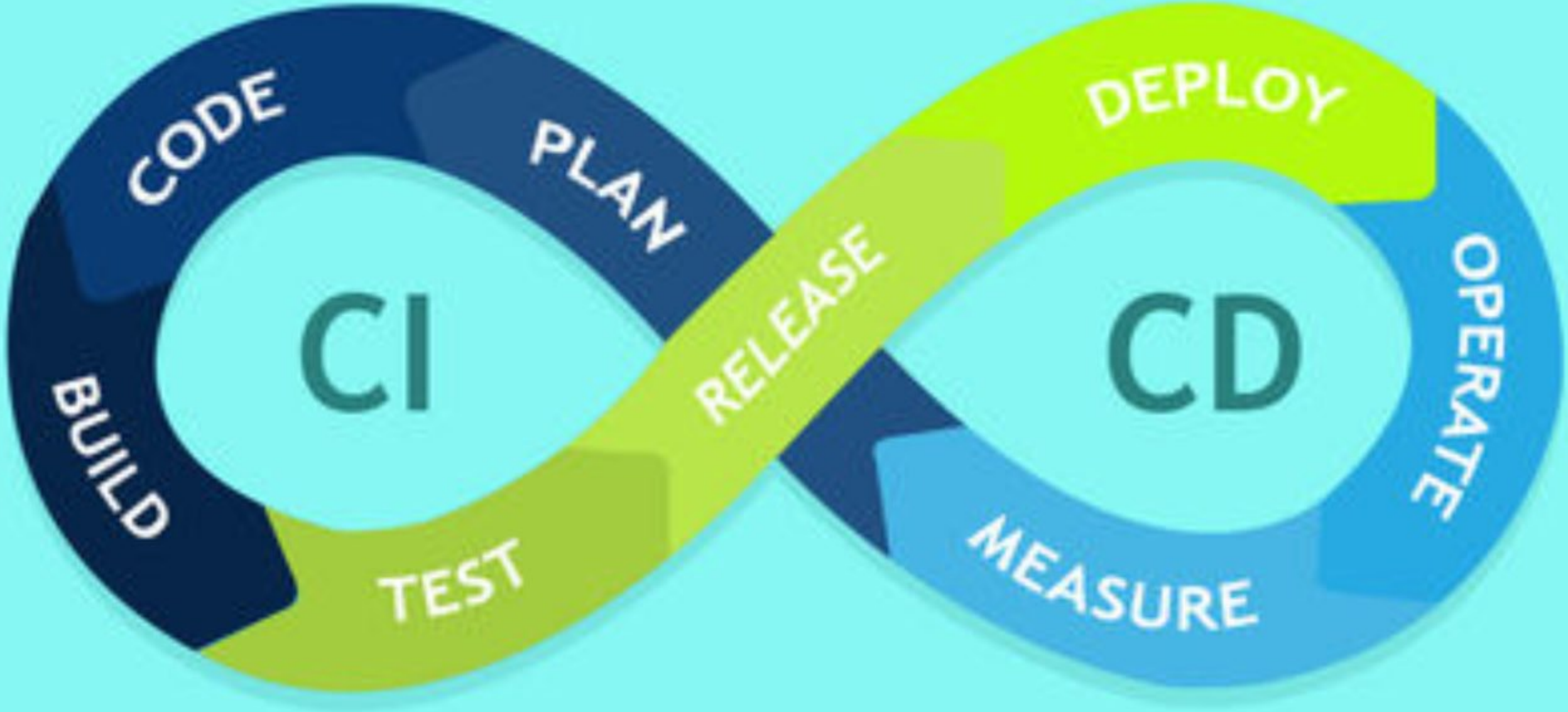
Automate all the things



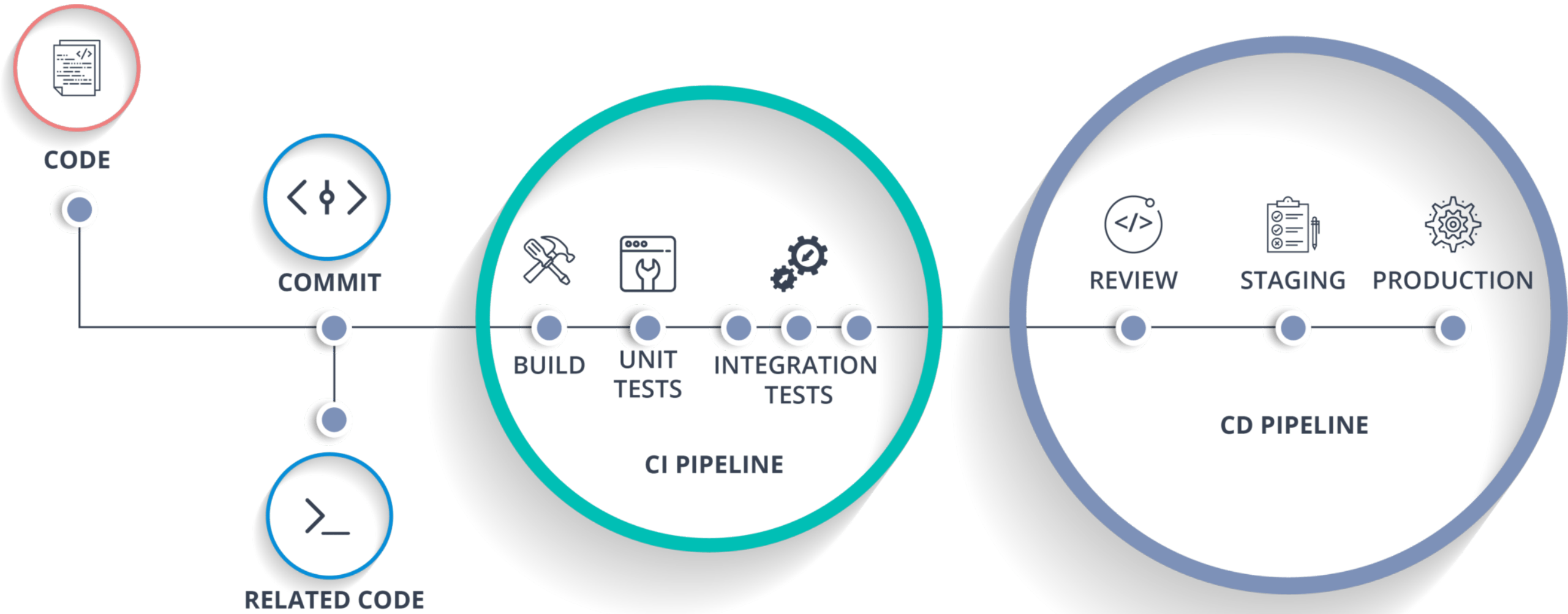
INSTALL.SH

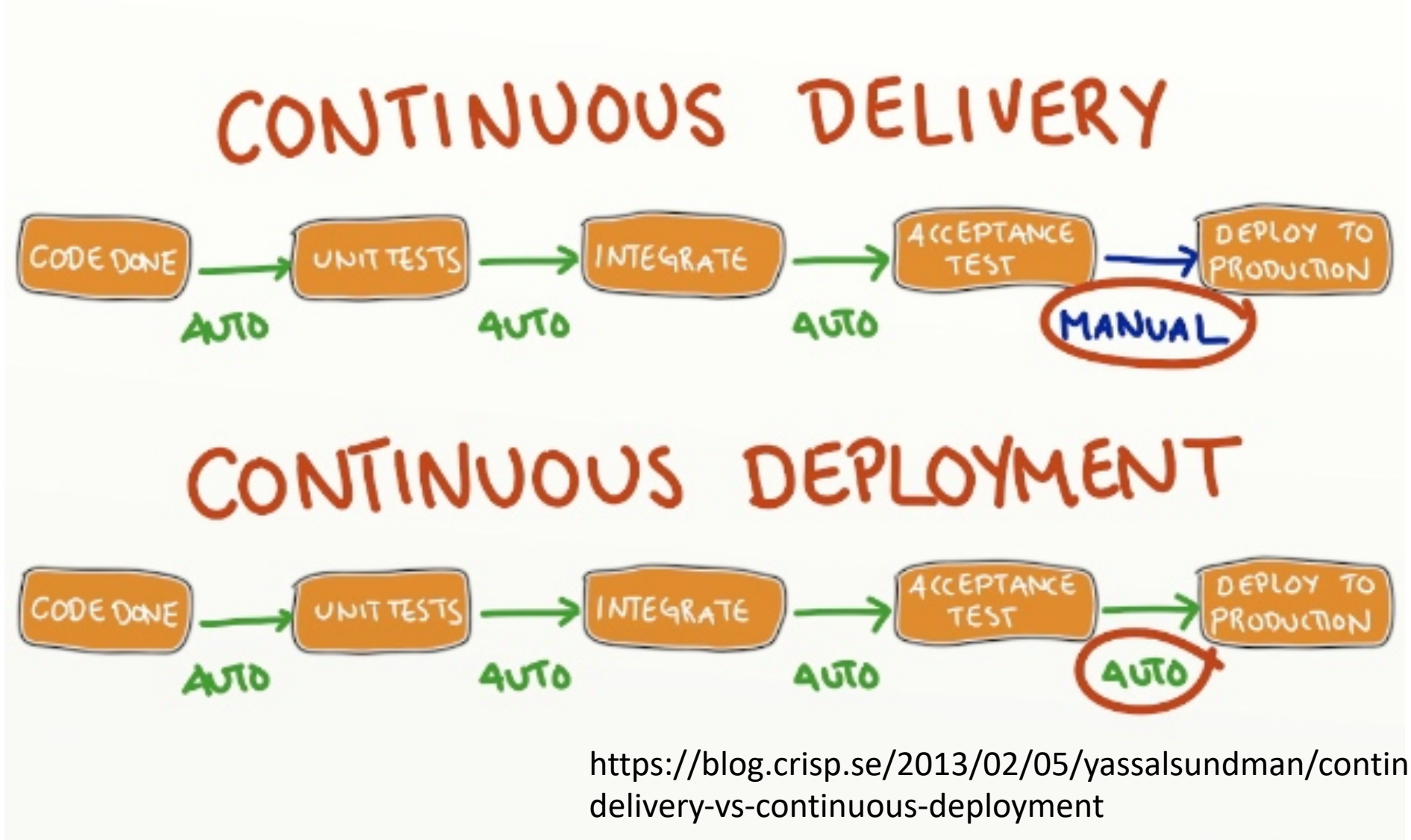
```
#!/bin/bash

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 &
```


CODE
PLAN
BUILD
TEST
CI
RELEASE
DEPLOY
OPERATE
MEASURE
CD

https://solidstudio.io/blog/ci-cd-pipelines.html





https://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment

Continuous Deployment

Two sides to DevOps

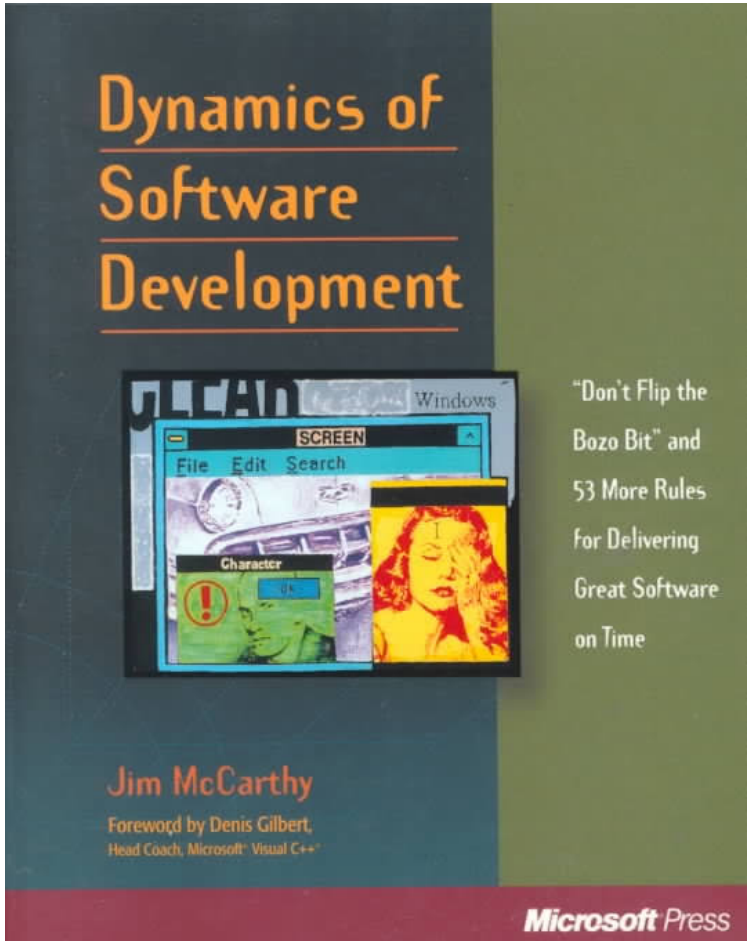
- **Operation-centric:**
  - Manage inventory of servers automatically
  - Provisioned, configured automatically Monitoring, analysis, automation of operations
- **Developer centric:**
  - Continuous deployment
  - Push code to production through pipeline

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



FSE'16

Continuous Deployment of Mobile Software at Facebook (Showcase)

Chuck Rossi
Facebook Inc.
1 Hacker Way
Menlo Park, CA USA 94025
chuckr@fb.com

Elisa Shibley
University of Michigan
2260 Hayward Street
Ann Arbor, MI USA 48109
eshibley@umich.edu

Shi Su
Carnegie Mellon University
PO Box 1
Moffett Field, CA USA 94035
shis@andrew.cmu.edu

Kent Beck
Facebook Inc.
1 Hacker Way
Menlo Park, CA USA 94025
kbeck@fb.com

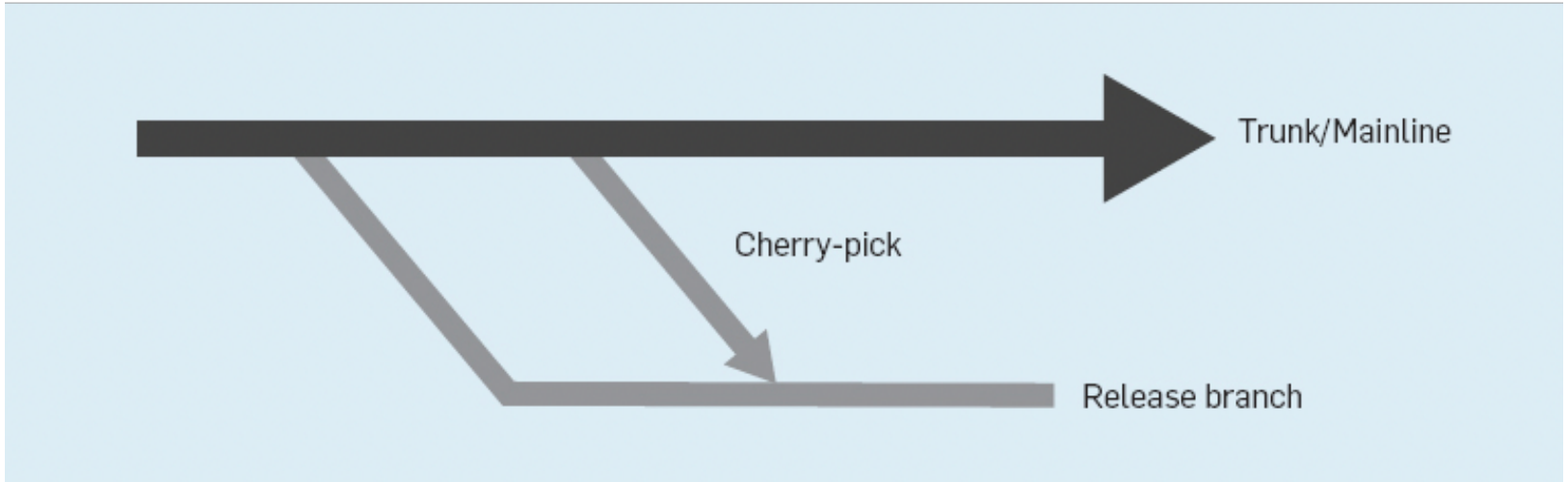
Tony Savor
Facebook Inc.
1 Hacker Way
Menlo Park, CA USA 94025
tsavor@fb.com

Michael Stumm
University of Toronto
10 Kings College Rd
Toronto, Canada M8X 2A6
stumm@eecg.toronto.edu

Release engineering and push karma: Chuck Rossi

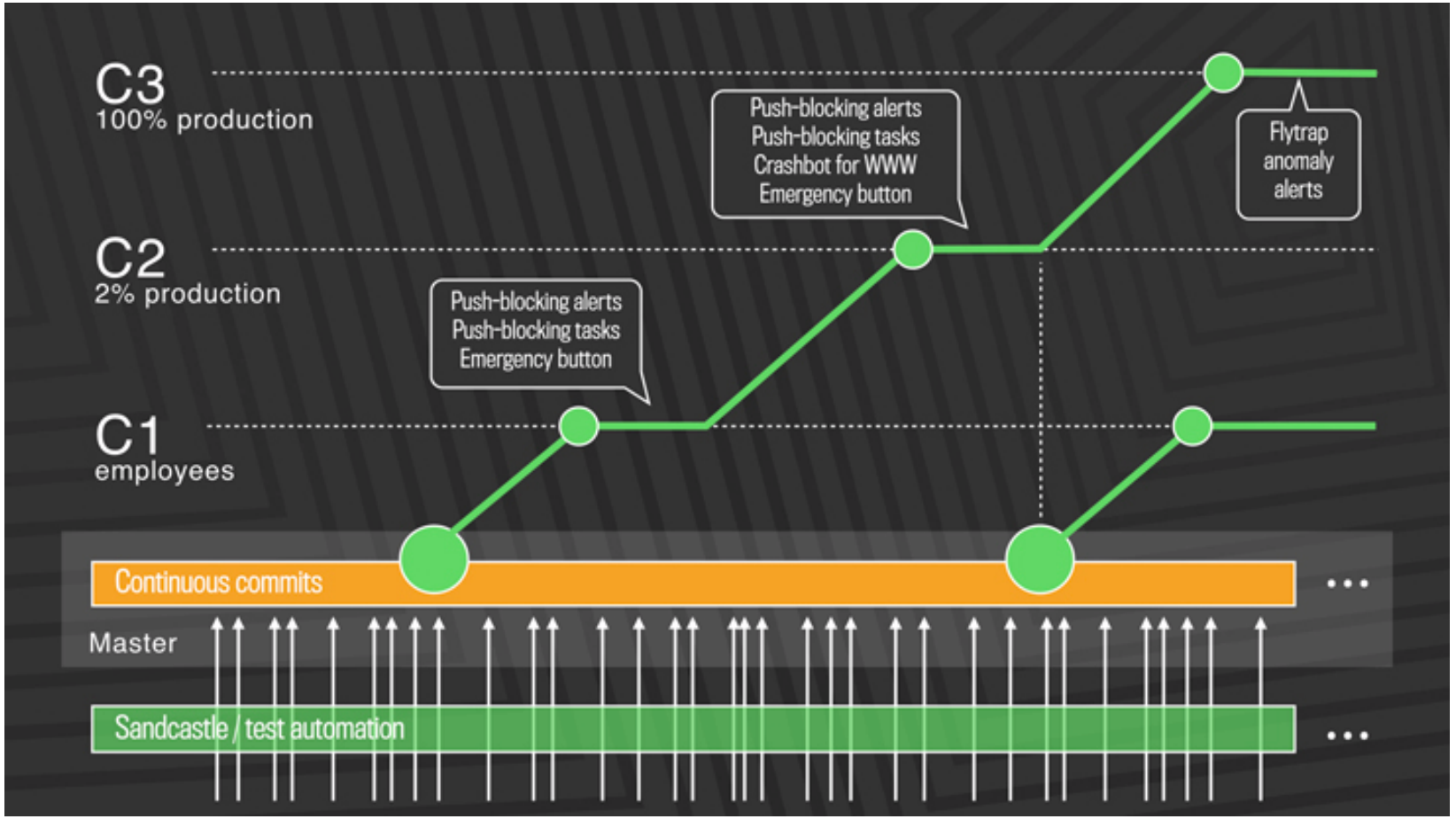
FB Facebook Engineering
April 5, 2012 · 5 min read ·

Facebook process (until 2016)



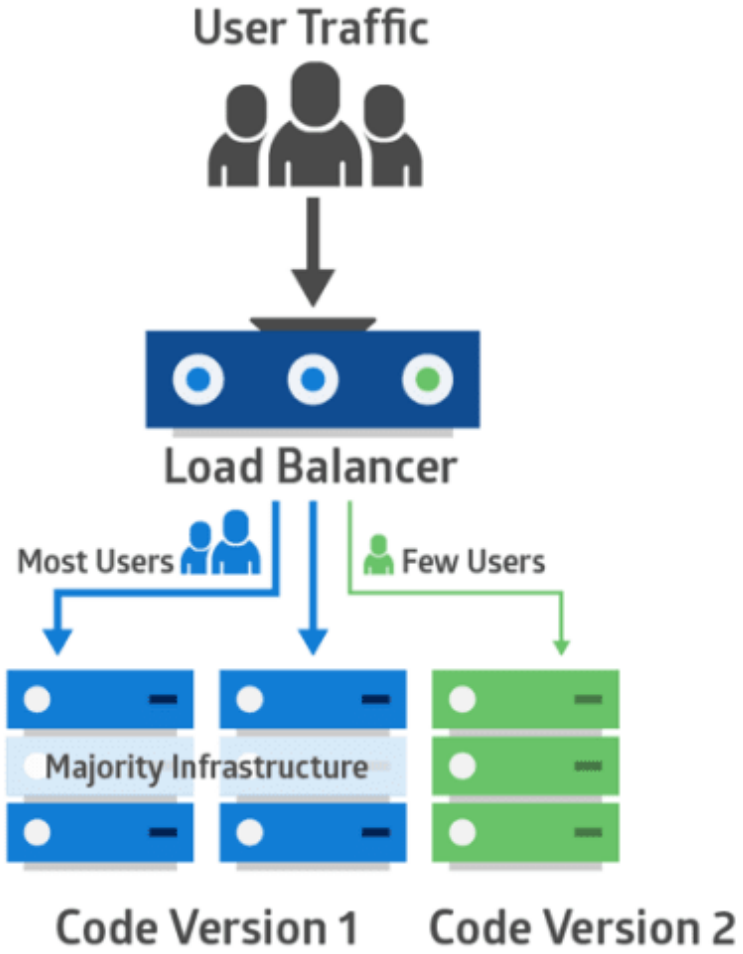
- Release is cut Sunday 6pm
- Stabilize until Tuesday, canaries, release. Tuesday push is 12,000 diffs.
- Cherry pick: Push 3 times a day (Wed-Fri) 300-700 cherry picks / day.

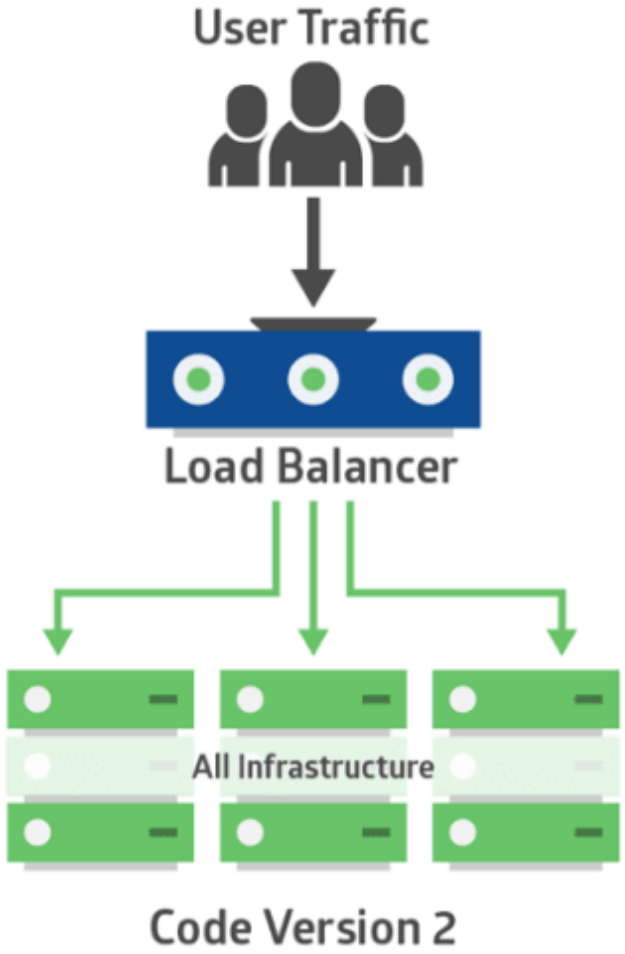
Facebook quasi-continuous release



https://engineering.fb.com/2017/08/31/web/rapid-release-at-massive-scale/

Canary Deployment

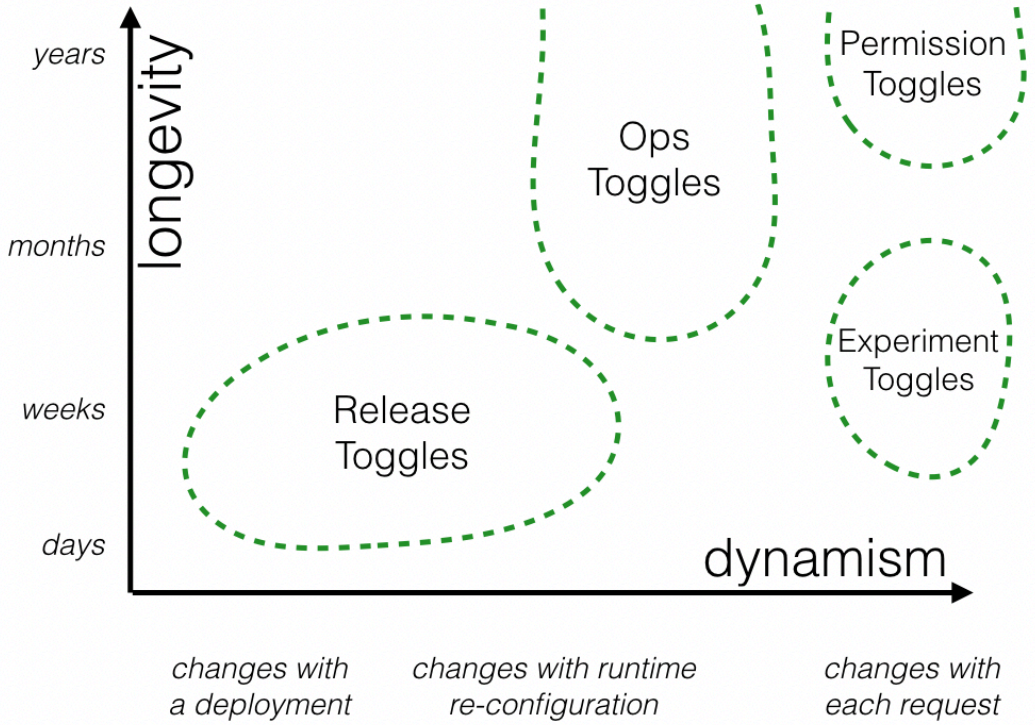


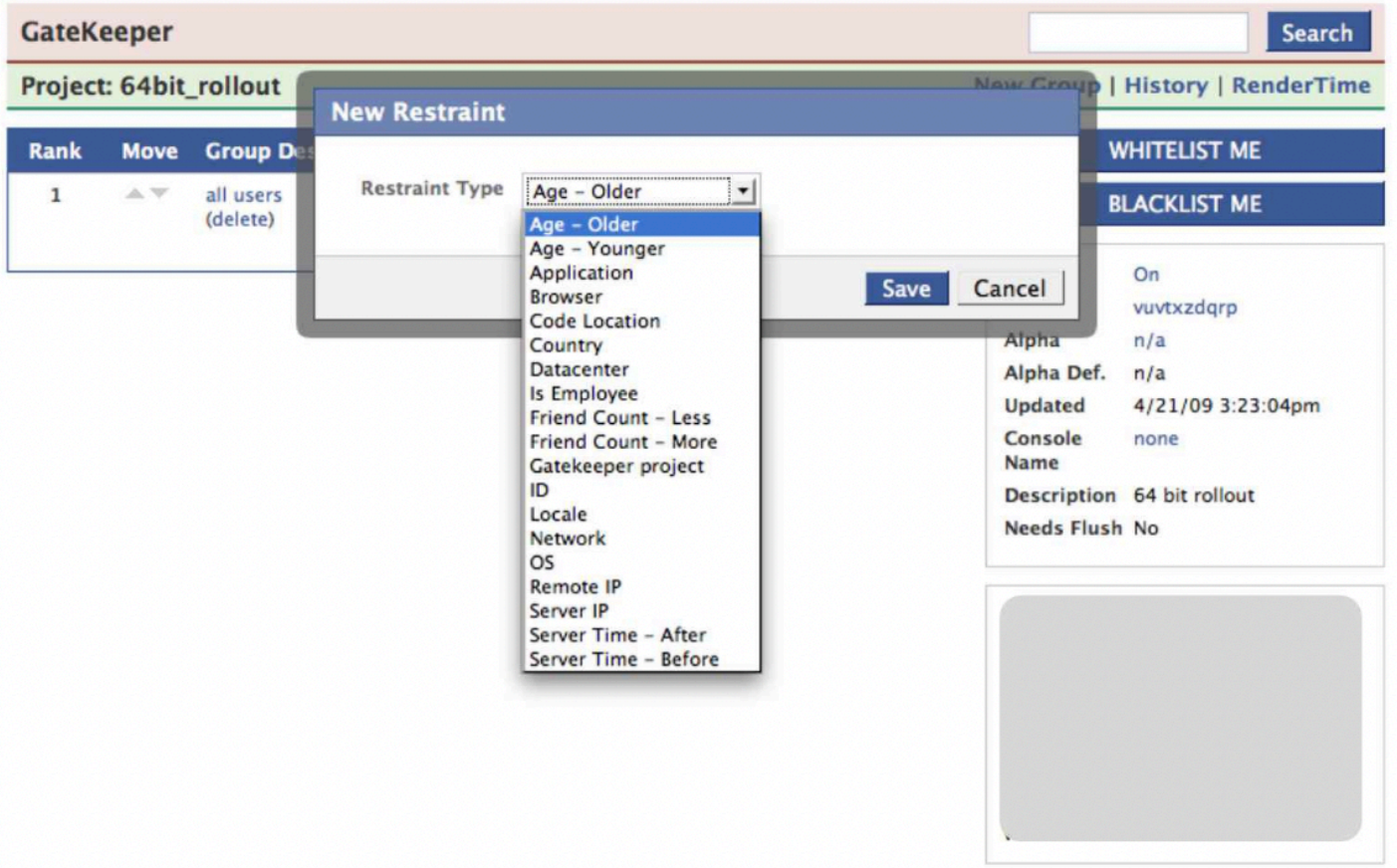


Dark Launches at Instagram

- **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.

Controlling feature flags





PRINCIPLE: EVERY FEATURE IS AN EXPERIMENT

NETFLIX

“Our journey to the cloud at Netflix began in August of 2008, when we experienced a major database corruption and for three days could not ship DVDs to our members. That is when we realised that we had to move away from vertically-scaled single points of failure, like relational databases in our datacenter, towards highly reliable, horizontally-scalable, distributed systems in the cloud.”

Yury Izrailevsky, VP, Cloud Computing and Platform Engineering, Netflix.

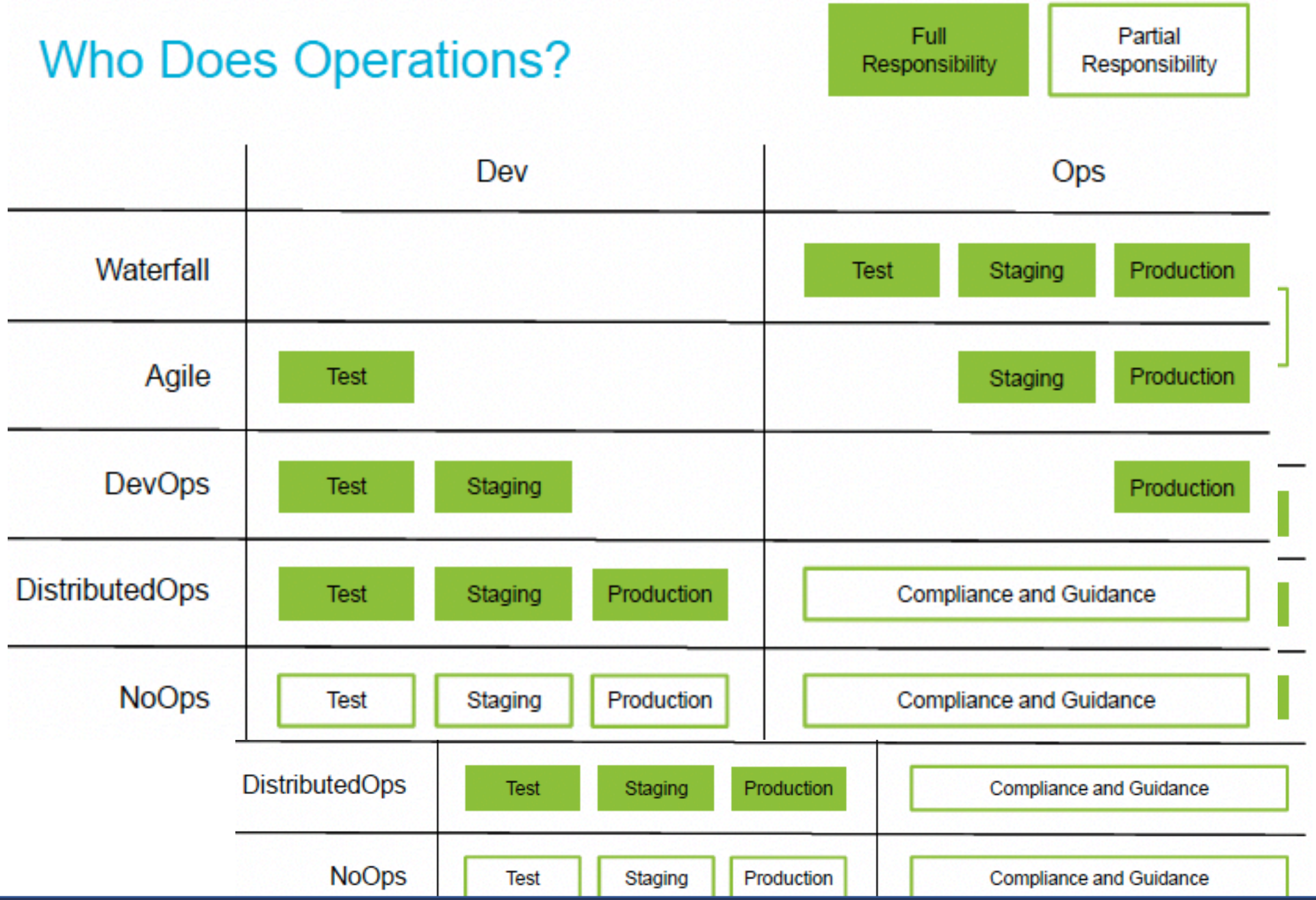
NETFLIX

- 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

https://thenewstack.io/netflix-devops-scale/

Who Does Operations?

So




Time for DevOps