DevOps

SE for Startups

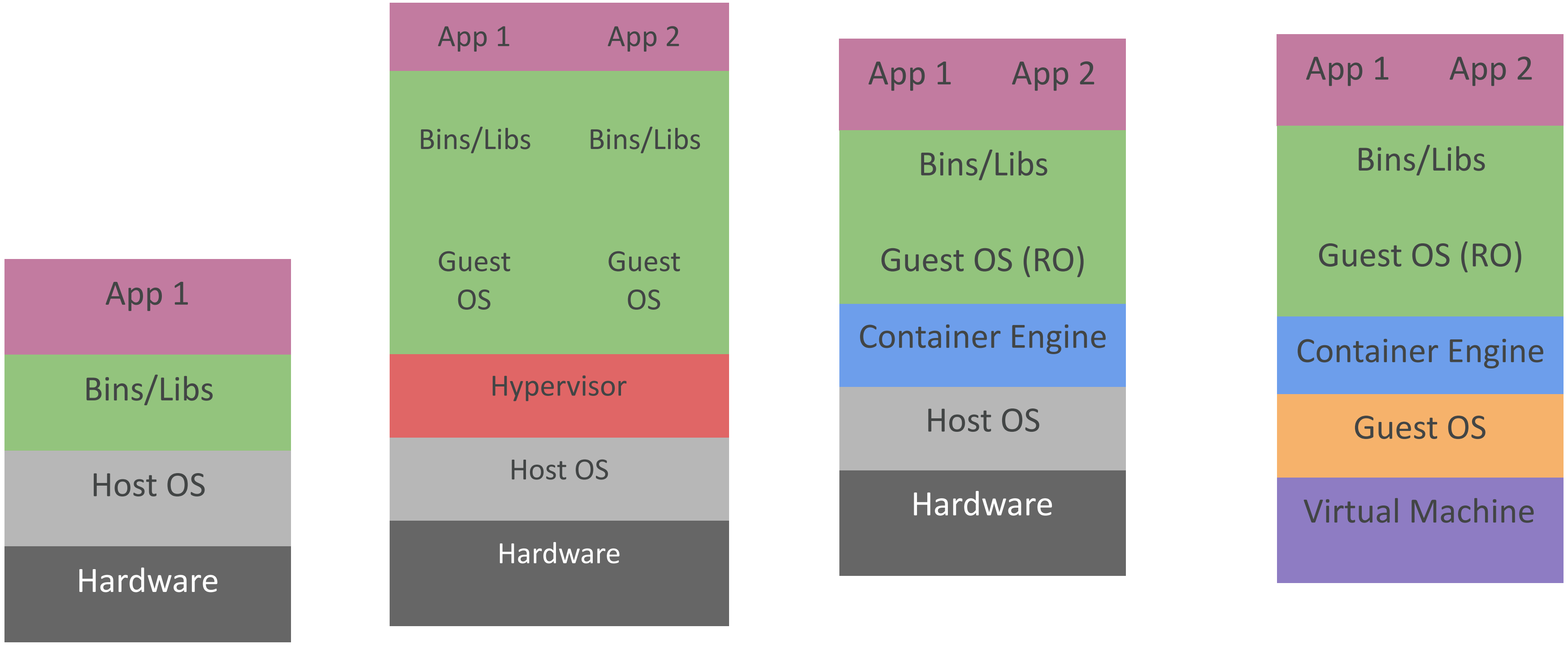
Much thanks to Chris Parnin, from whom slides were appropriated.

KEY IDEA:

There are lots of ways of implementing a microservice infrastructure.

Microservices in Practice - Instances

“Instances” are the fundamental unit of microservices:



Physical Machines (BM)

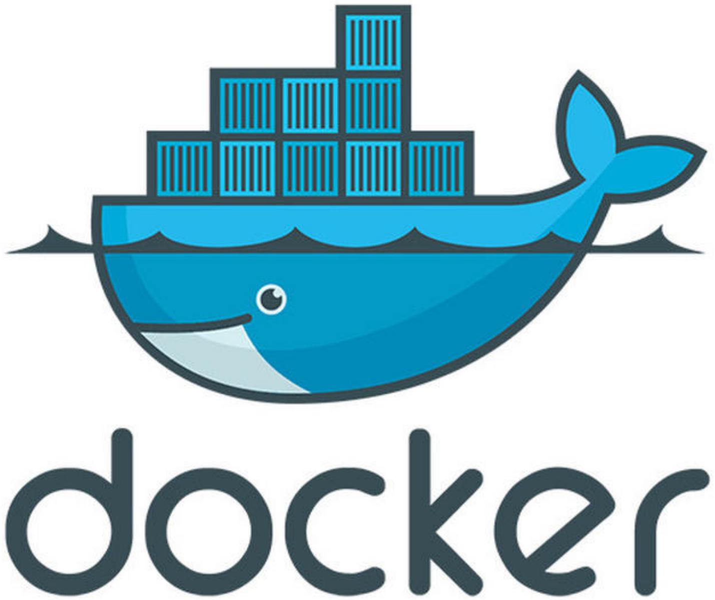
Virtual Machines (VM)

Containers (BM)

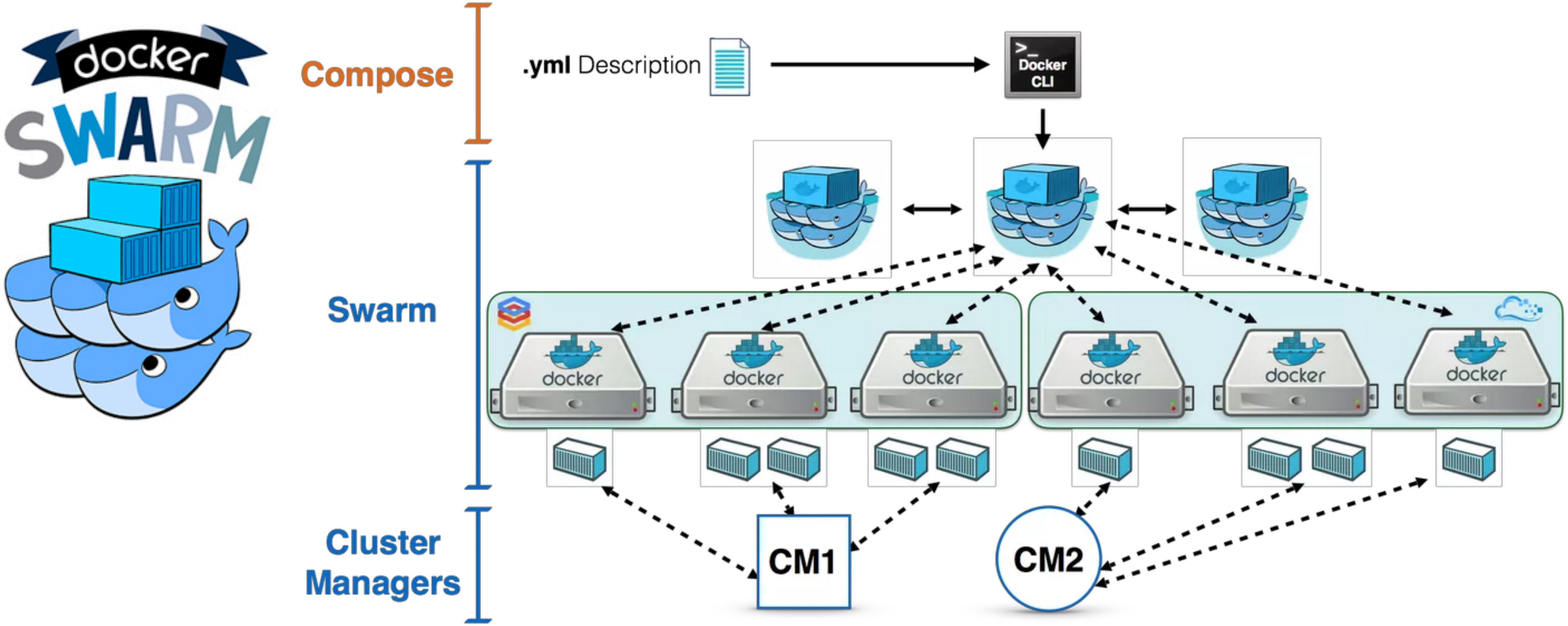
Containers (VM)

Microservices - Containers

- Containers are a more efficient sandbox than VMs.
- Microservices are often conflated with containers; Containers are one possible way to implement microservices, but not the only one.
- Docker is king.



Microservices - Containers



Other container orchestration tools:

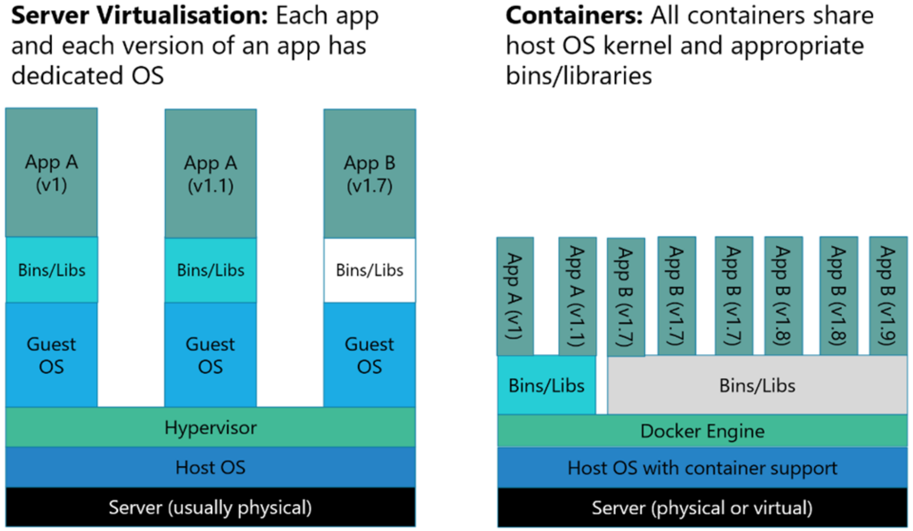
kubernetes

fleet

Microservices - Containers

Advantages:

- Provision once.
- Single copy of OS/Binaries/Libraries.
- OS is purpose-built / more stable.
- No emulation required.
- Good support for shared networking, mounting distributed file systems.
- Faster startup.



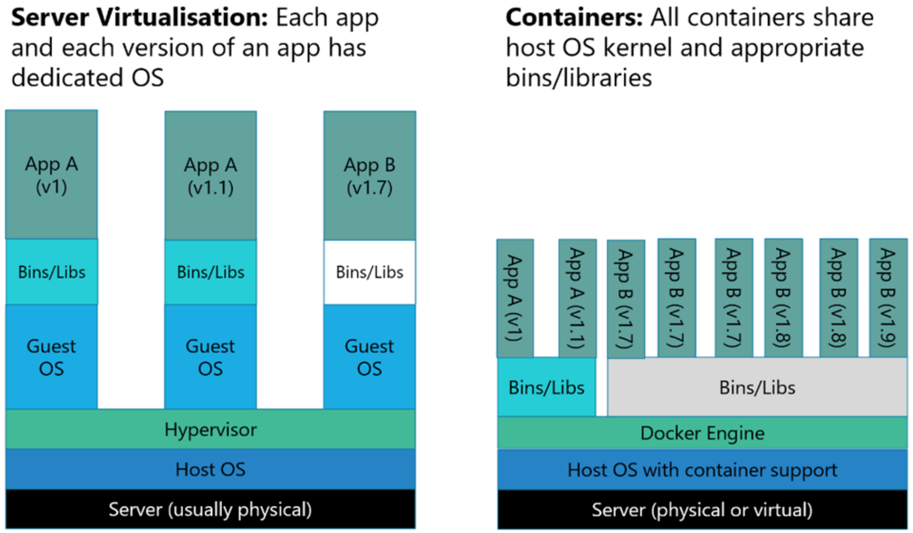
Server Virtualisation: Each app and each version of an app has dedicated OS

Containers: All containers share host OS kernel and appropriate bins/libraries

Microservices - Containers

Disadvantages:

- Not 100% as secure as separate operating systems.
- Do not have hardware-level access to many features like networking.
- Less control over data/process residency.
- Overhead as compared to bare metal server.



Server Virtualisation: Each app and each version of an app has dedicated OS

Containers: All containers share host OS kernel and appropriate bins/libraries

Microservices in Practice - Databases

Because microservices are stateless,
they require databases
to store data. We will cover this later.

Process at a large company…

1. Set feature freeze date.
2. Post feature freeze, two week bug-fixing/minor rework.
3. Pass to QA; QA tests, reports back.
4. Two week rework, to Gold Standard freeze.
5. Gold standard release printed to discs, shipped…

- Why is this good?
- Why is it bad?
    - …specifically in a startup context?

STARTUP ENGINEERING GOAL: BUILD STUFF TO CHANGE

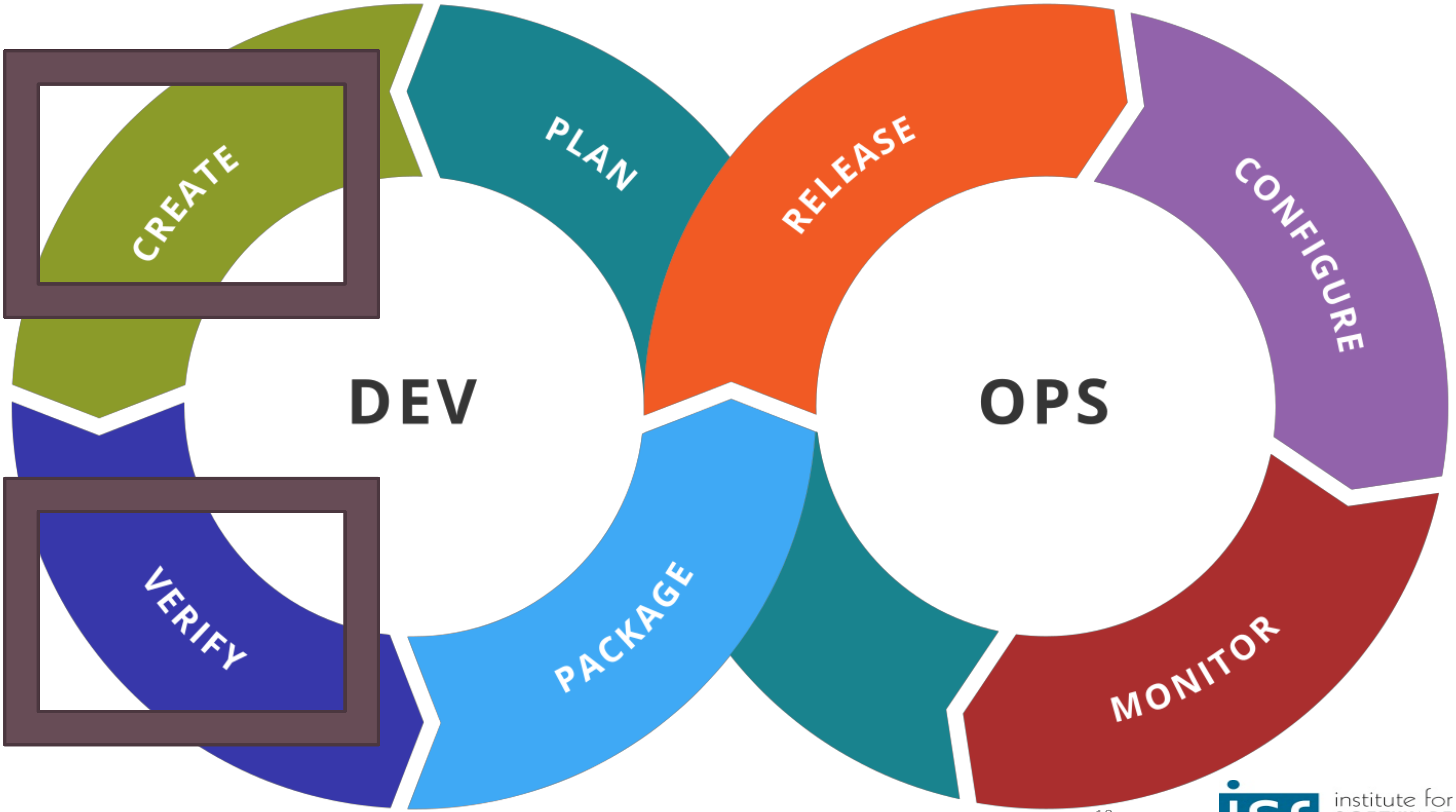
Buzzword of the hour: DEVOPS

- To quote Wikipedia: “a software engineering culture and practice that aims at unifying software development (Dev) and software operation (Ops).”
  - …I am not ashamed.
- Also: thank you to Chris Parnin, NCSU, from whom we appropriated some content/slides (including/especially memes).

DevOps

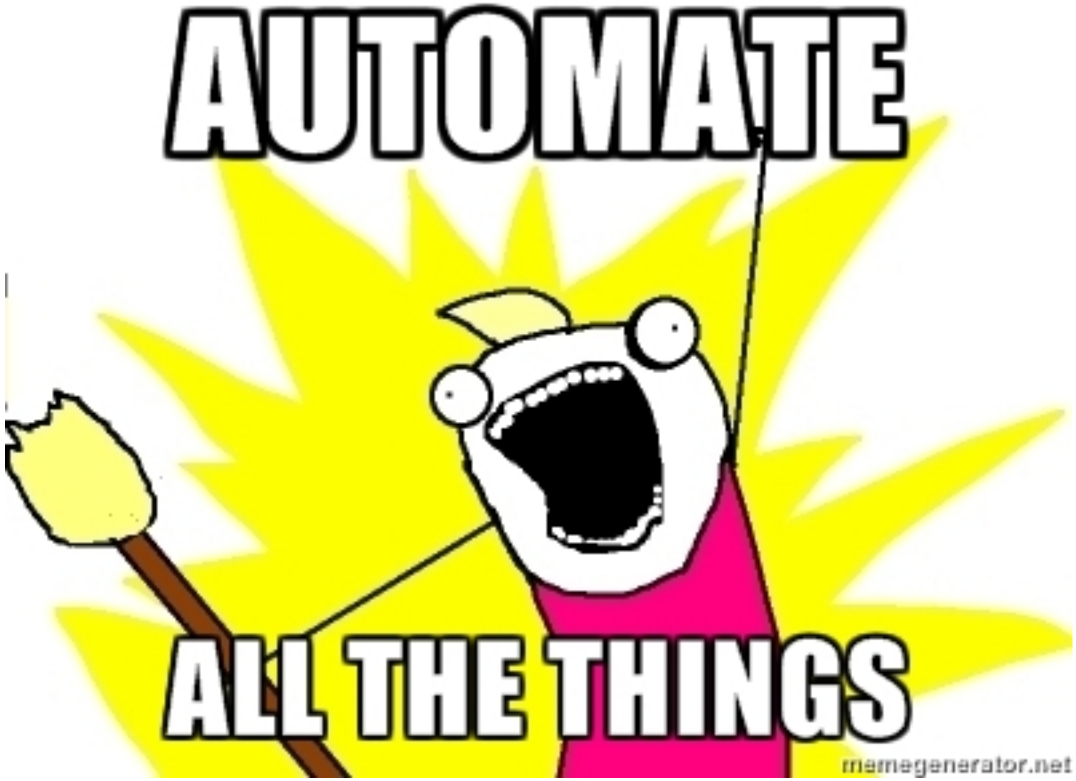
A short history of devops

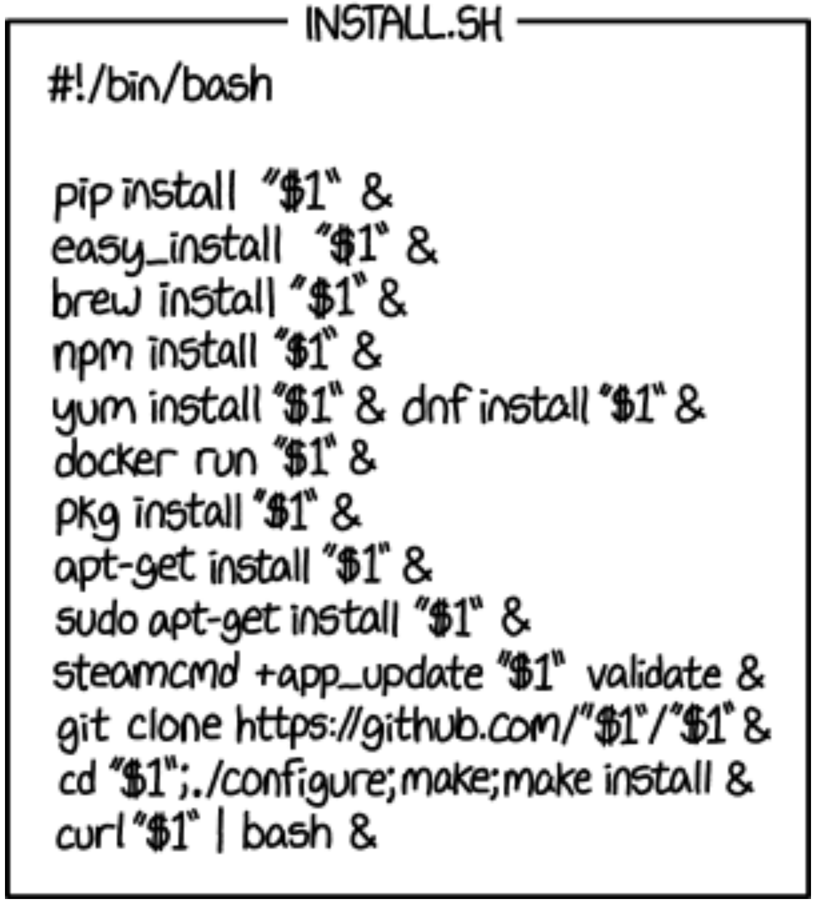
http://itrevolution.com/the-history-of-devops/



isr institute for SOFTWARE RESEARCH

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

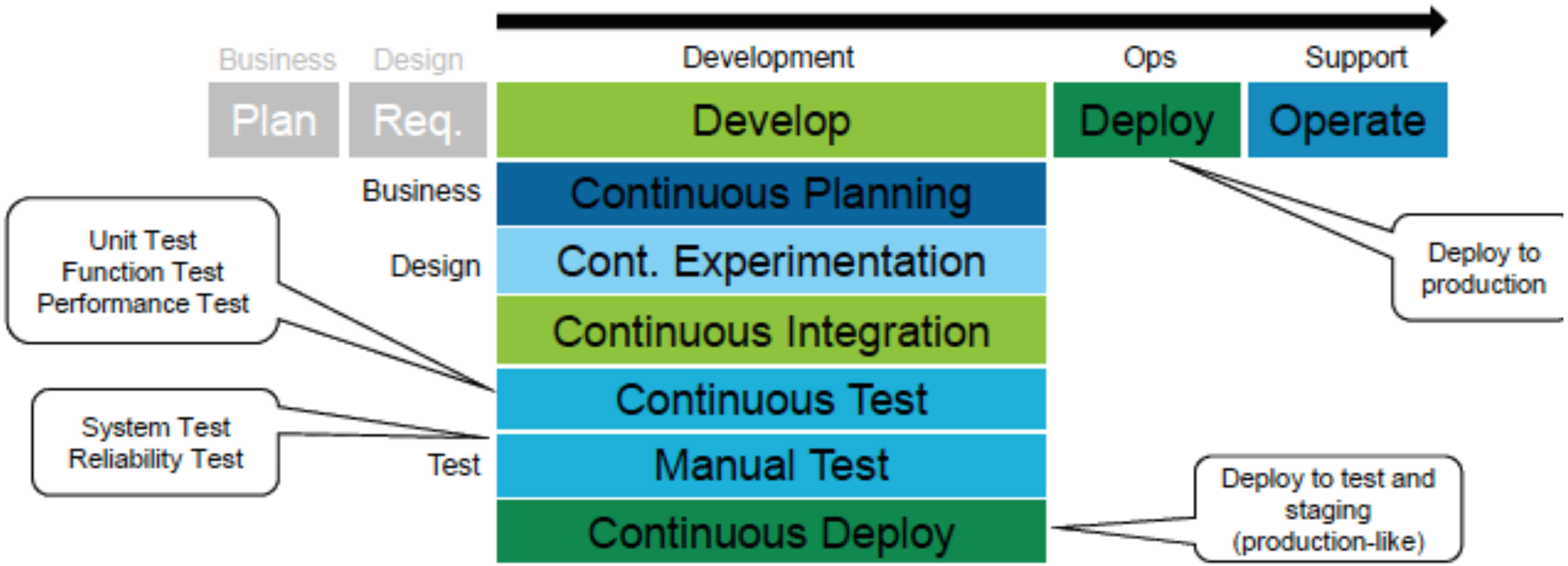
- It minimizes integration risk.
- It reduces the risk of low quality
- It supports easier defect diagnosis
- It improves morale

AGILE! (THE GIFT THAT KEEPS ON GIVING)

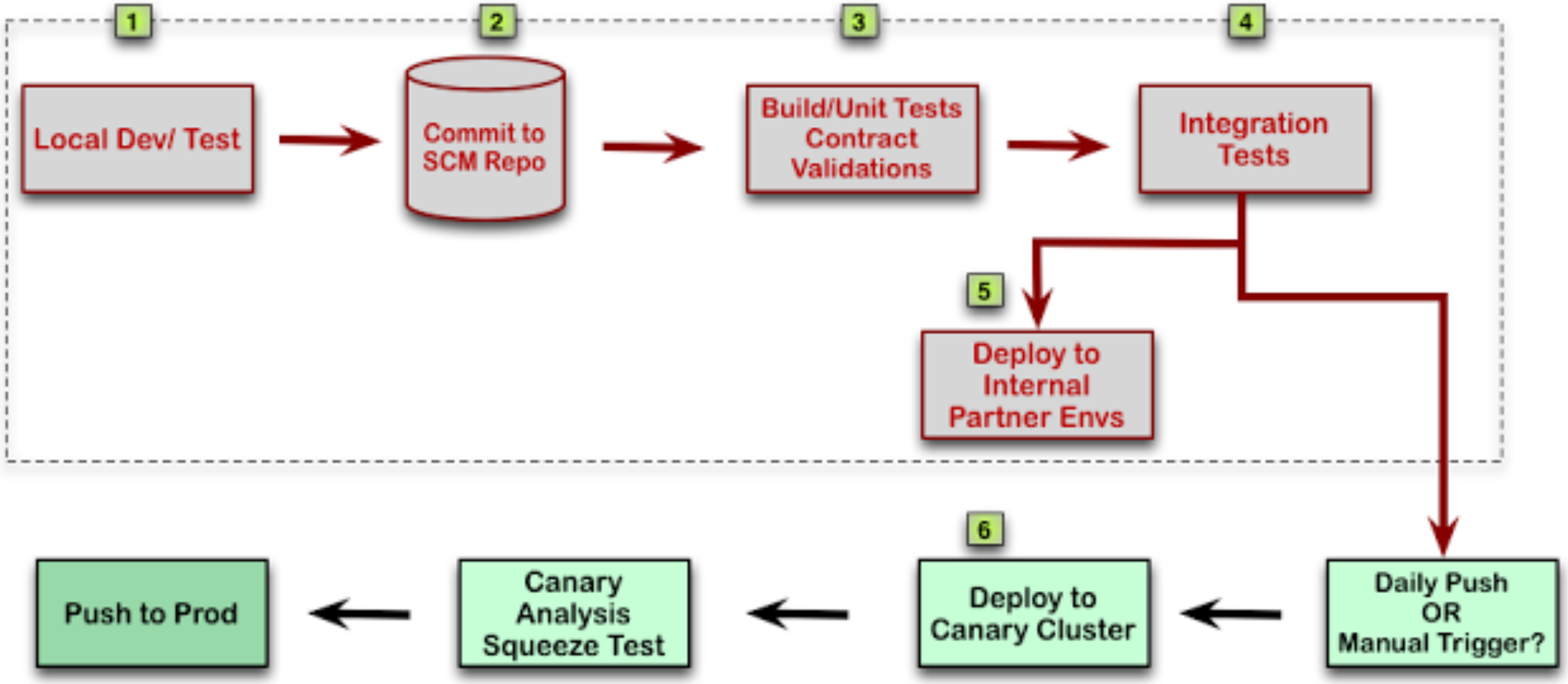
Continuous…

Integration: A practice where developers automatically build, test, and analyze a software change in response to every software change committed to the source repository.
Delivery: A practice that ensures that a software change can be delivered and ready for use by a customer by testing in production-like environments.
Deployment: A practice where incremental software changes are automatically tested, vetted, and deployed to production environments.

Continuous * (Perpetual Development)



Example Deployment Pipeline


1
2
3
4
Local Dev/ Test
Commit to SCM Repo
Build/Unit Tests Contract Validations
Integration Tests
5
Deploy to Internal Partner Envs
6
Push to Prod
Canary Analysis Squeeze Test
Deploy to Canary Cluster
Daily Push OR Manual Trigger?

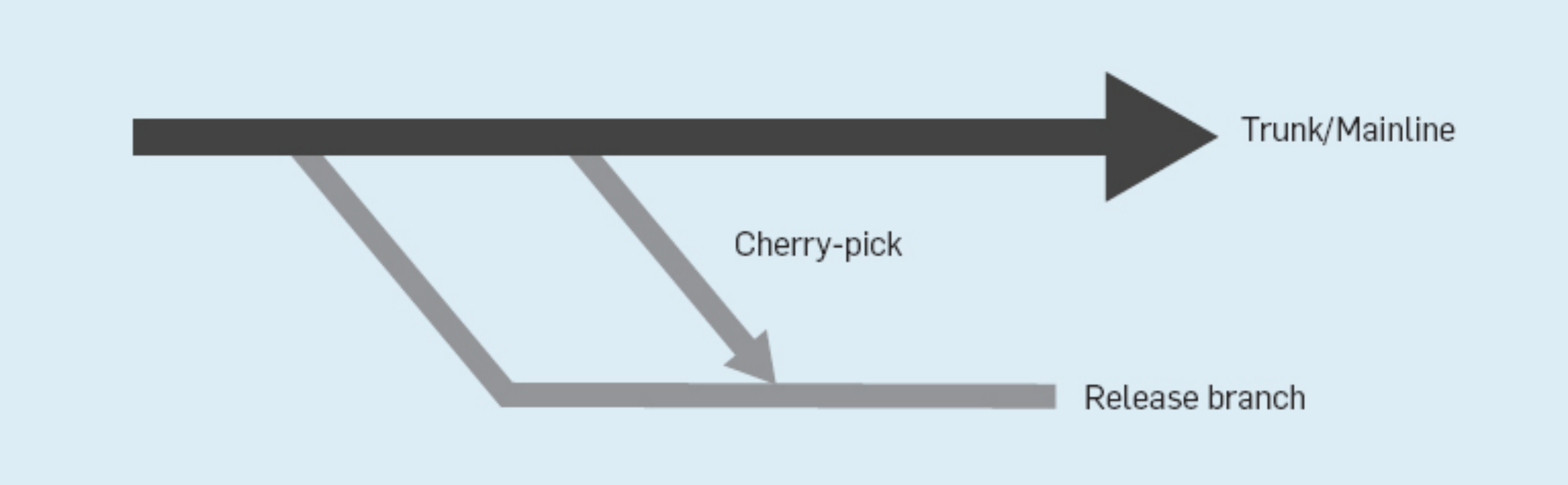
Principle: Fast to Deploy, Slow to Release

Chuck Rossi at Facebook: *“Get your shit 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)

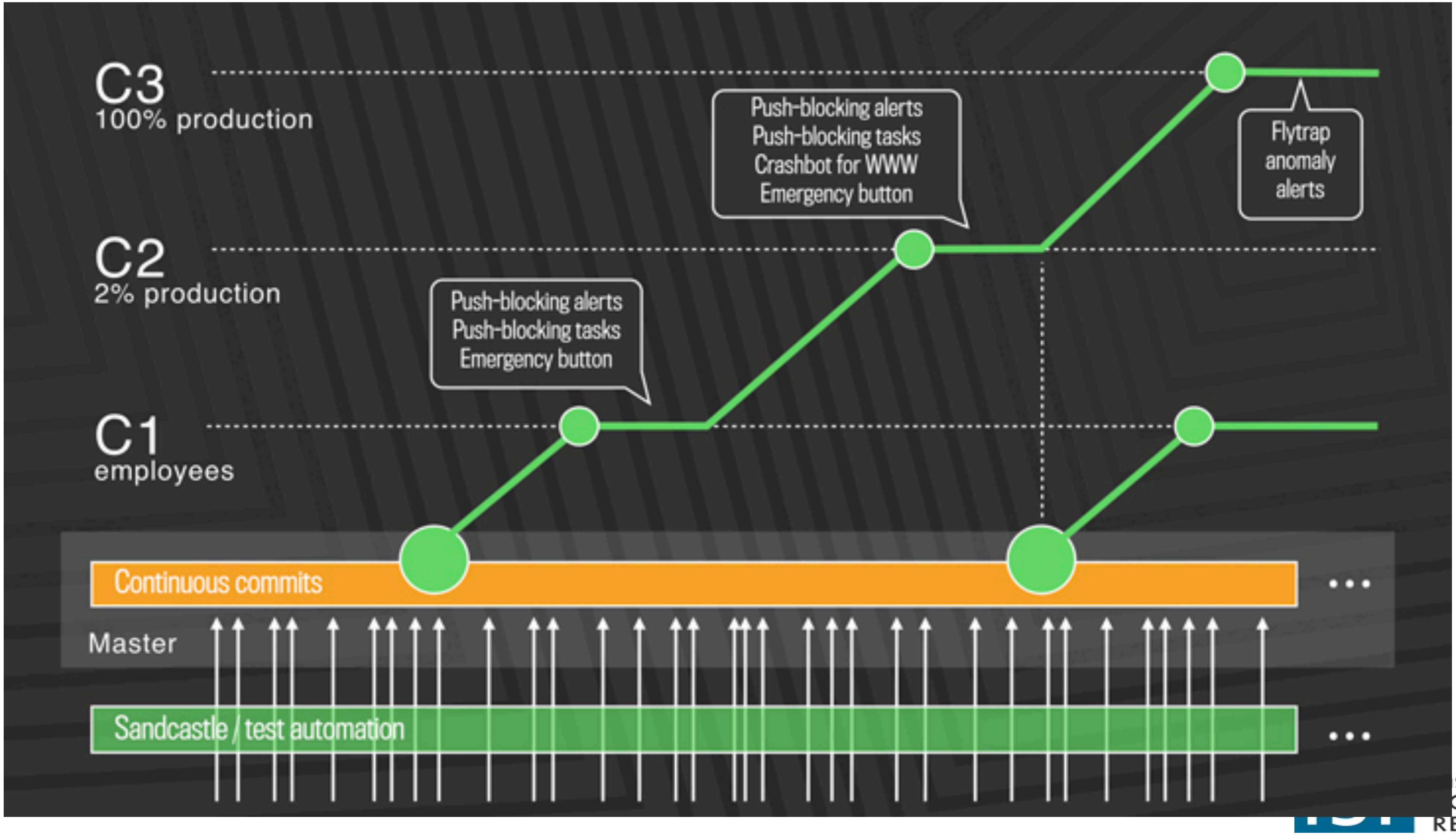


Release is cut Sunday 6pm

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 (PURELY FABRICATED) example, assume we want to apply to LexisNexus:
    - Ring 0 => LexisNexis Legal Department (2 people)
    - Ring 1 => UNC Law School (Free broken software for students)
    - Ring 2 => Beta
    - Ring 3 => Many
    - Ring 4 => All

Broadly: what’s release management?

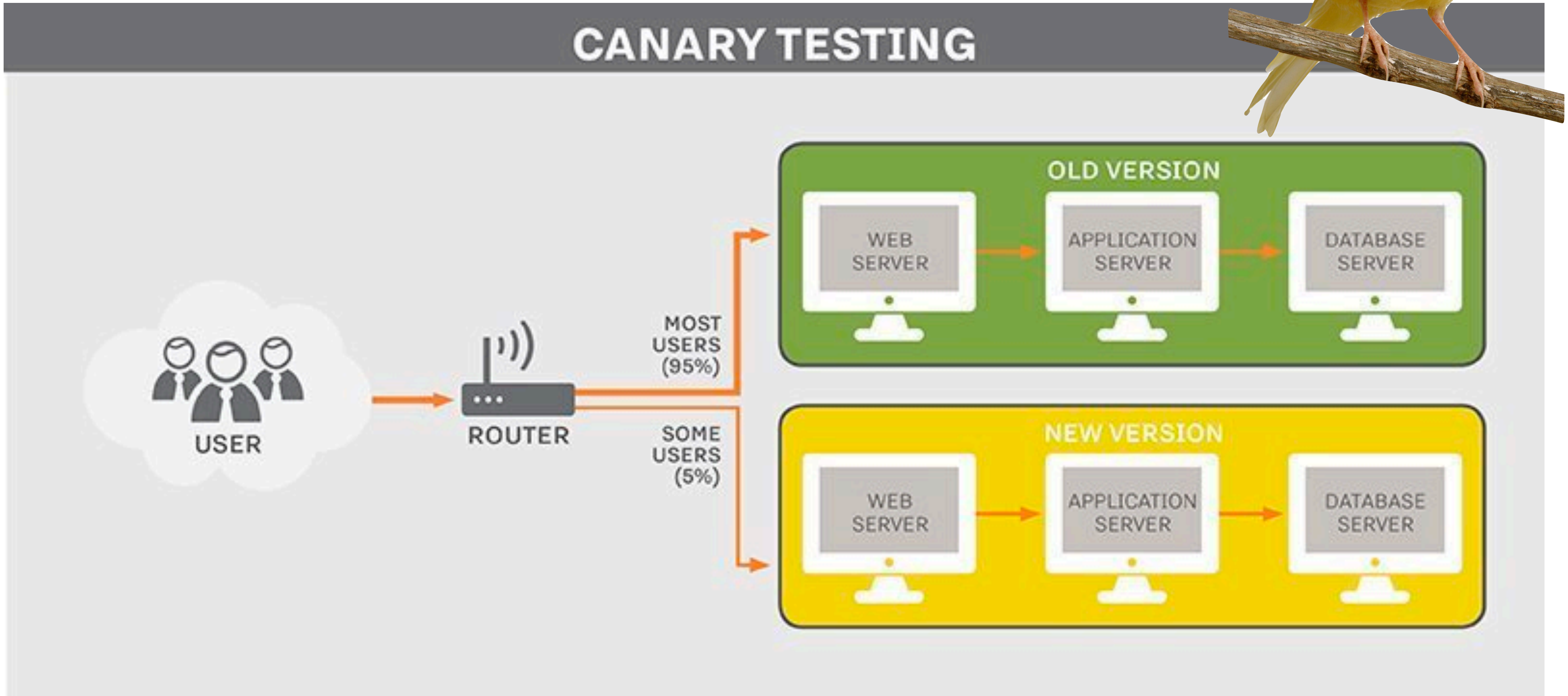
- And how to do it in a startup context?
- Do these principles generalize?

- If nothing else, Rule for Life: Never Ship on a Friday.

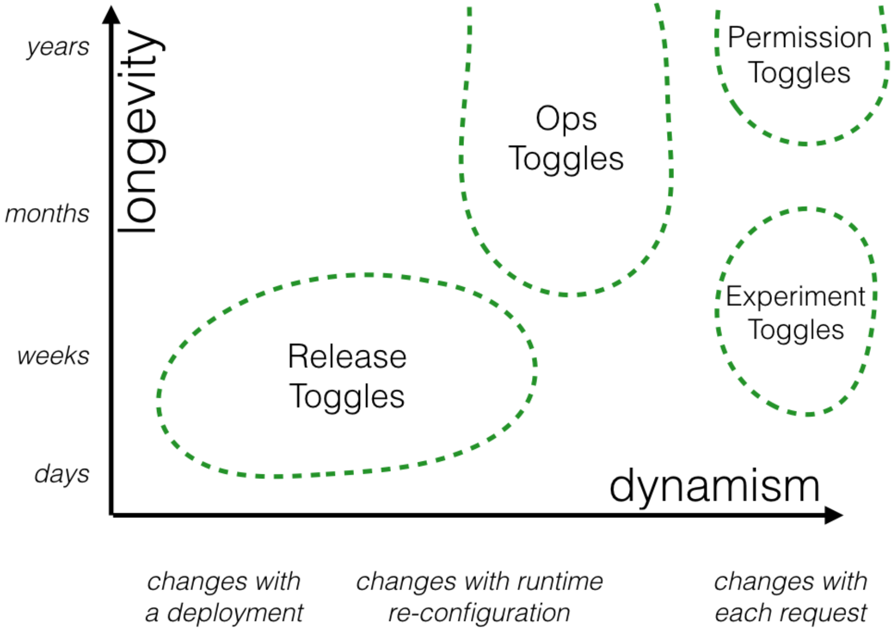
PRINCIPLE: EVERY FEATURE IS AN EXPERIMENT

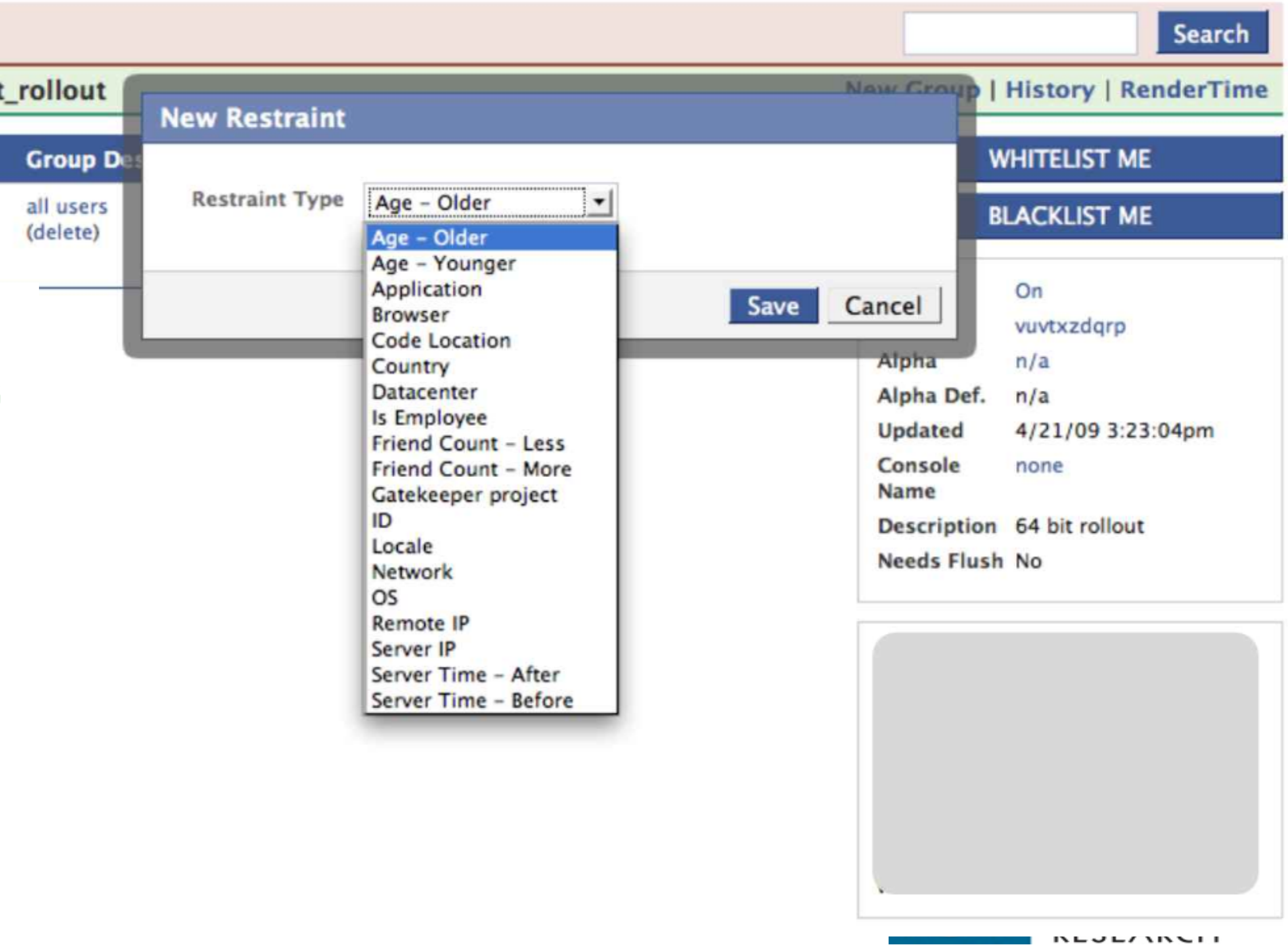
Feature testing:

Canary testing



Controlling feature flags





Rank	Move	Group Des
1		all users (delete)

Restraint Type: Age – Older

	On
	vuvtxzdqrp
Alpha	n/a
Alpha Def.	n/a
Updated	4/21/09 3:23:04pm
Console Name	none
Description	64 bit rollout
Needs Flush	No

Netflix

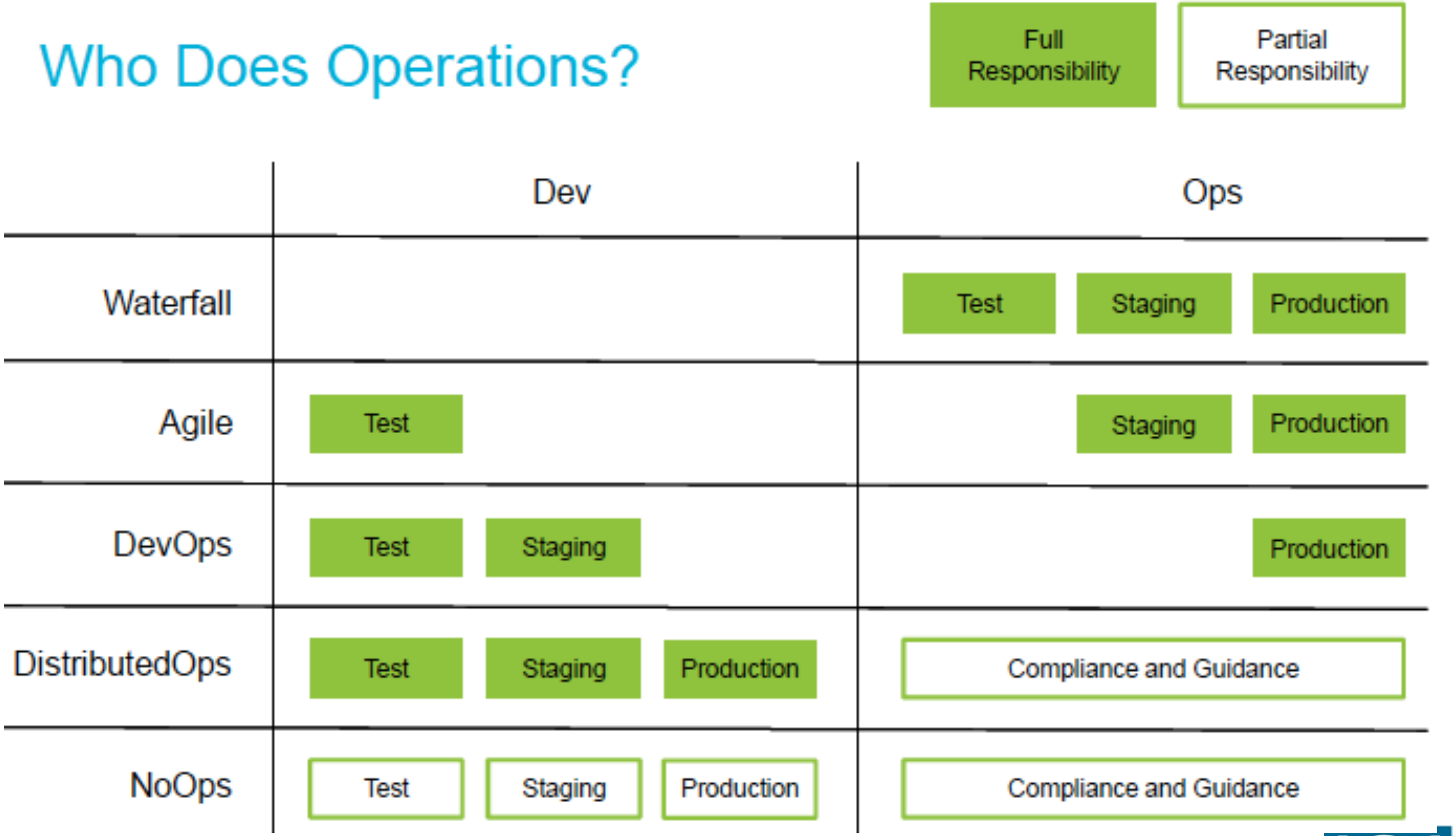
60,000 configuration changes a day. 4000 commits a day.

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.

So who's responsibility is all of this?



Who Does Operations?

🟩 Full Responsibility

□ Partial Responsibility

	Dev	Ops
Waterfall		🟩 Test 🟩 Staging 🟩 Production
Agile	🟩 Test	🟩 Staging 🟩 Production
DevOps	🟩 Test 🟩 Staging	🟩 Production
DistributedOps	🟩 Test 🟩 Staging 🟩 Production	□ Compliance and Guidance
NoOps	□ Test □ Staging □ Production	□ Compliance and Guidance

What is a candidate deployment plan for Dronuts?

Jenkins Job Builder

```yaml
scm:
 - git:
     url: https://github.com/openstack-infra/jenkins-job-builder.git
     credentials-id: "43ed1990-46e5-4ed0-bfda-8d83e5cdd65f"
     branches:
       - master
     clean: true
     shallow-clone: true
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<project>
 <scm class="hudson.plugins.git.GitSCM">
   <configVersion>2</configVersion>
   <userRemoteConfigs>
     <hudson.plugins.git.UserRemoteConfig>
       <name>origin</name>
       <refspec>+refs/heads/*:refs/remotes/origin/*</refspec>
       <url>https://github.com/openstack-infra/jenkins-job-builder.git</url>
       <credentialsId>43ed1990-46e5-4ed0-bfda-8d83e5cdd65f</credentialsId>
     </hudson.plugins.git.UserRemoteConfig>
   </userRemoteConfigs>
   <branches>
     <hudson.plugins.git.BranchSpec>
       <name>master</name>
     </hudson.plugins.git.BranchSpec>
   </branches>
   <disableSubmodules>false</disableSubmodules>
   <recursiveSubmodules>false</recursiveSubmodules>
   <doGenerateSubmoduleConfigurations>false</doGenerateSubmoduleConfigurations>
   <remotePoll>false</remotePoll>
   <gitTool>Default</gitTool>
   <submoduleCfg class="list"/>
   <reference/>
   <gitConfigName/>
   <gitConfigEmail/>
   <extensions>
     <hudson.plugins.git.extensions.impl.CleanCheckout/>
     <hudson.plugins.git.extensions.impl.CloneOption>
       <shallow>true</shallow>
     </hudson.plugins.git.extensions.impl.CloneOption>
     <hudson.plugins.git.extensions.impl.WipeWorkspace/>
   </extensions>
 </scm>
</project>
```

isr institute for SOFTWARE RESEARCH

Issues

- Project can take **hours** to build.
- **Bad state**
  - Sometimes build jobs leave side-effects which need to be manually cleared.
- **High memory usage** or memory leaks in unit testing code can exceed server memory and melt… needing frequent restarts.
- **High volume** of build requests…
  - Executors can help, which will run jobs on subordinate servers.

Maven

A tool for managing dependencies and build lifecycles.

Primarily configured via a **pom.xml** file.

Dependencies

```
<dependencies>
        <dependency>
                <groupId>org.apache.tomcat</groupId>
                <artifactId>tomcat</artifactId>
                <version>8.0.28</version>
                <type>pom</type>
        </dependency>
        <dependency>
                <groupId>org.seleniumhq.selenium</groupId>
                <artifactId>selenium-java</artifactId>
                <version>2.25.0</version>
        </dependency>
```