An OCIO Digital Literacy Course

DevOps For Product Owners

Part 1: The Big Questions

Stephen Curran, Cloud Compass Computing, Inc.

DevOps For Product Owners

Part 1: The Big Questions

1. Introductions
2. What is DevOps?
3. What is the Cloud?
4. What is the Bustiness Impact?

graze

Approach

The course will be completed over two sessions covering a mixture of presentation, lab work and discussion. Please, feel free to jump in at a time with questions, comments, suggestions, snorts, etc. The goal is the material is presented in *your* context.

There will be a couple of labs that allow you to say - *I done DevOps*

Lots of opportunity for you to drive the course direction.

Logistics...

- Any constraints on time?
- Washrooms
- Food and beverages

Introductions

Who are you?

- Project
- Role
- Experience with Digital Services?

Who Am I?

Stephen Curran, *Cloud Compass Computing, Inc., Quartech Systems*

- Tightrope guy - business and technology
- All about the delivery
- Agile Development Methodology
- DevOps since before it was DevOps
- BC Government Projects ICM, JAG and MOTI - *School Bus* and *Hired Equipment*

What is DevOps?

DevOps (a clipped compound of "software DEVelopment" and "information technology OPerationS") is a term used to refer to a set of practices that emphasize the collaboration and communication of both software developers and information technology (IT) professionals while automating the process of software delivery and infrastructure changes. It aims at establishing a culture and environment where building, testing, and releasing software can happen rapidly, frequently, and more reliably.*

Well, that doesn't help...

* Wikipedia

Why is DevOps?

Roots - merging Developers and Operations work

- Developers - make the code
  - User Interface (UI/UX)
  - Business Logic/Rules
  - Integrations
  - Database (usually)
- Ops - runs the code
  - Servers
  - Networks
  - Databases

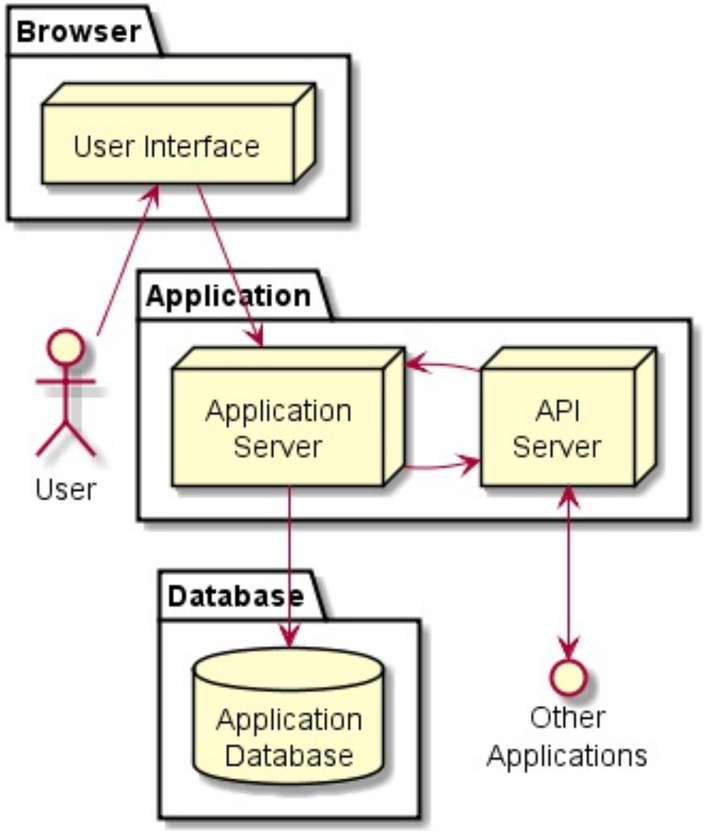
Backup a bit - what's an app?

Examples

- .NET + front end + database
- Java + front end + database
- MEAN (Mongo Express Angular Node)
- Django (Python + front end + database

- Front End: Bootstrap, React, Backbone, Angular, etc.
- Database: Postgres, SQL Server, Oracle, Mongo

User Stories, usability, logic, rules...

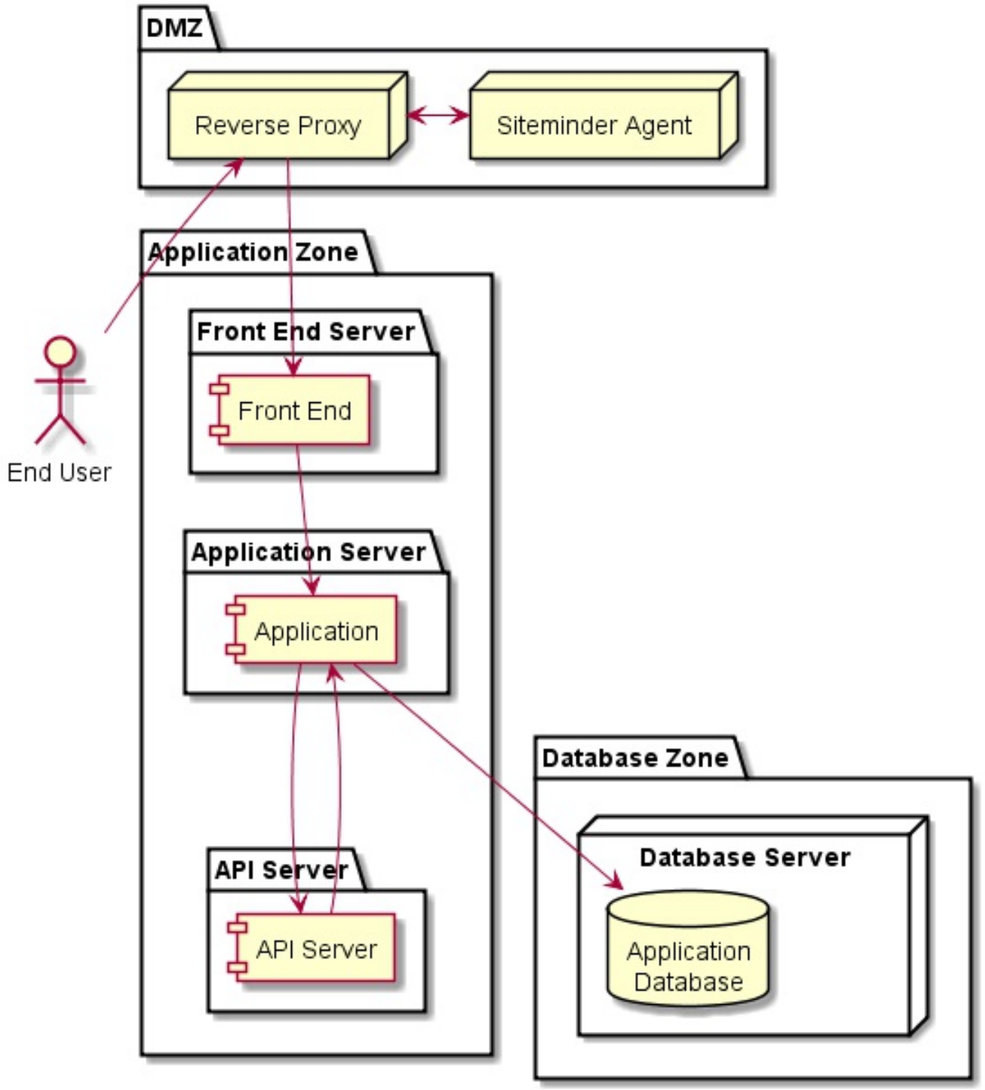


Backup a bit - where does an app run?

Ops View

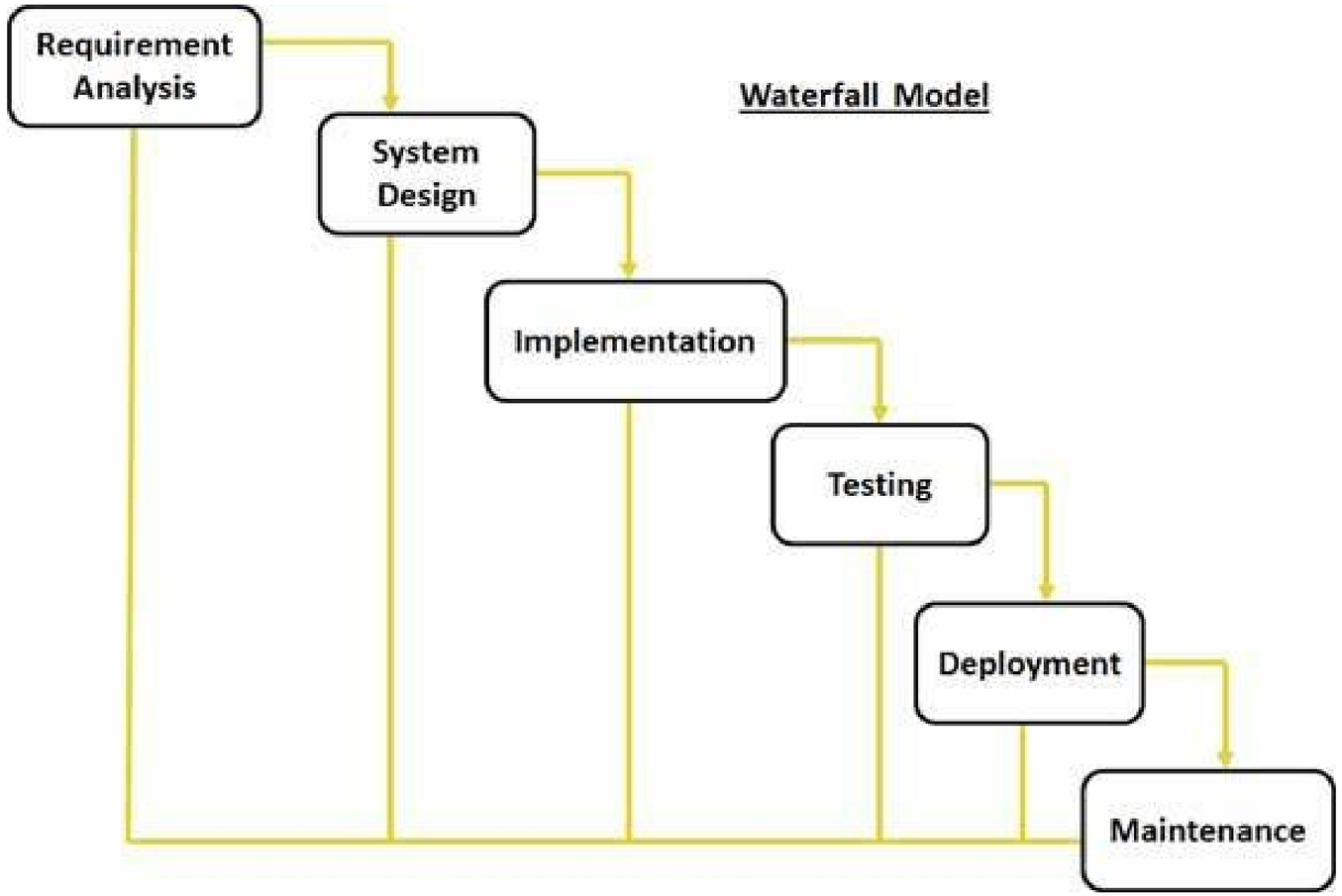
- Networking zones
- URLs - *https://myapp.gov.bc.ca*
- Authentication - siteminder
- Encryption - SSL
- Firewalls
- Servers
- Storage

Times 3: Dev/Test/Prod



graze

Making it Work - Theory

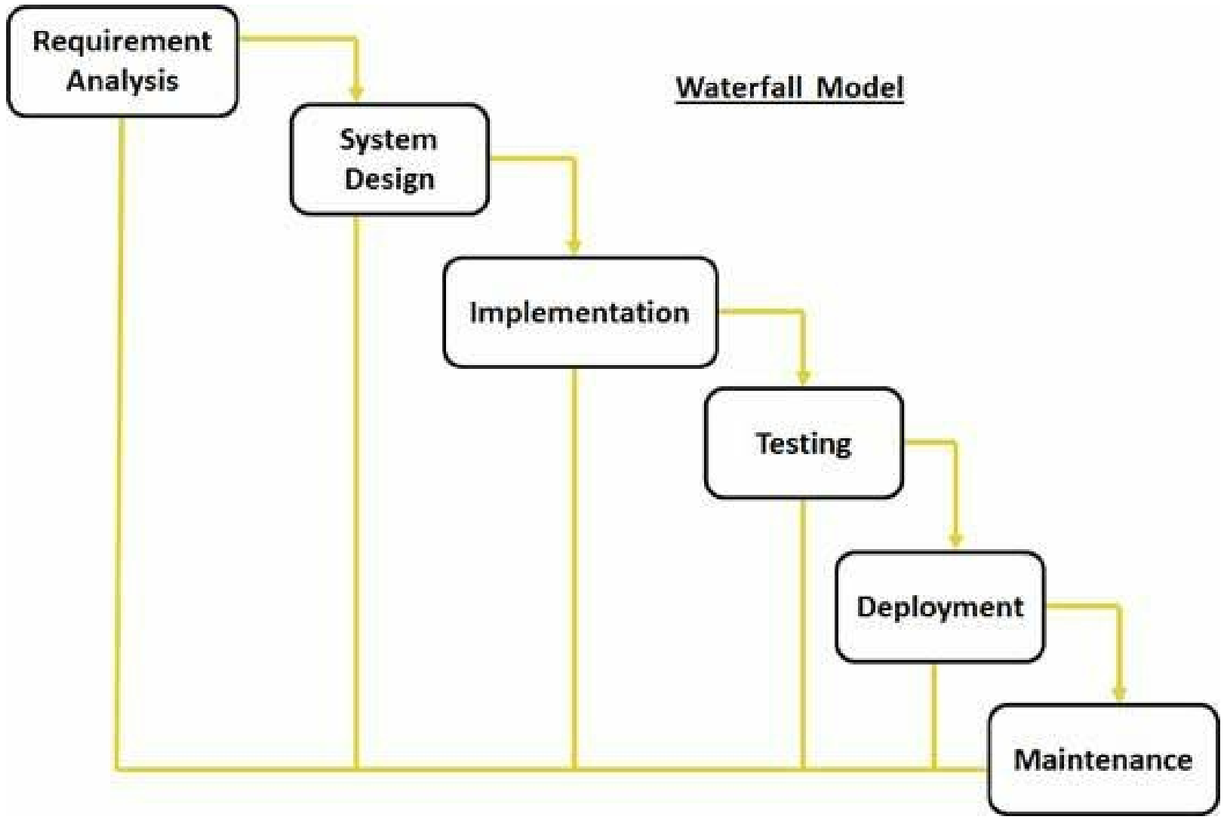


Meetings, documents, agreements and requests

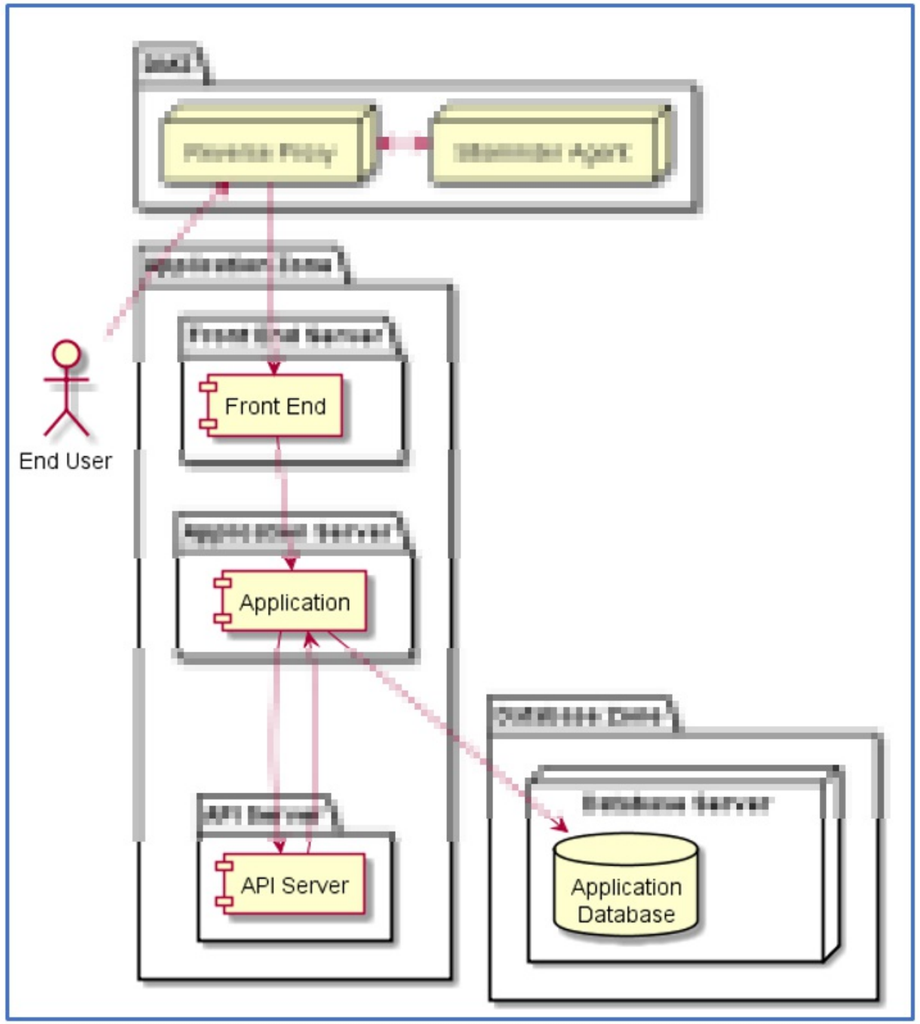
Reality

- Requirements: *change*
- Implementation: *Takes too long*
  - Devs: Code Development
  - Ops: Acquisition/Configuration of Servers
- Testing: *Skipped*
- Deployment is...

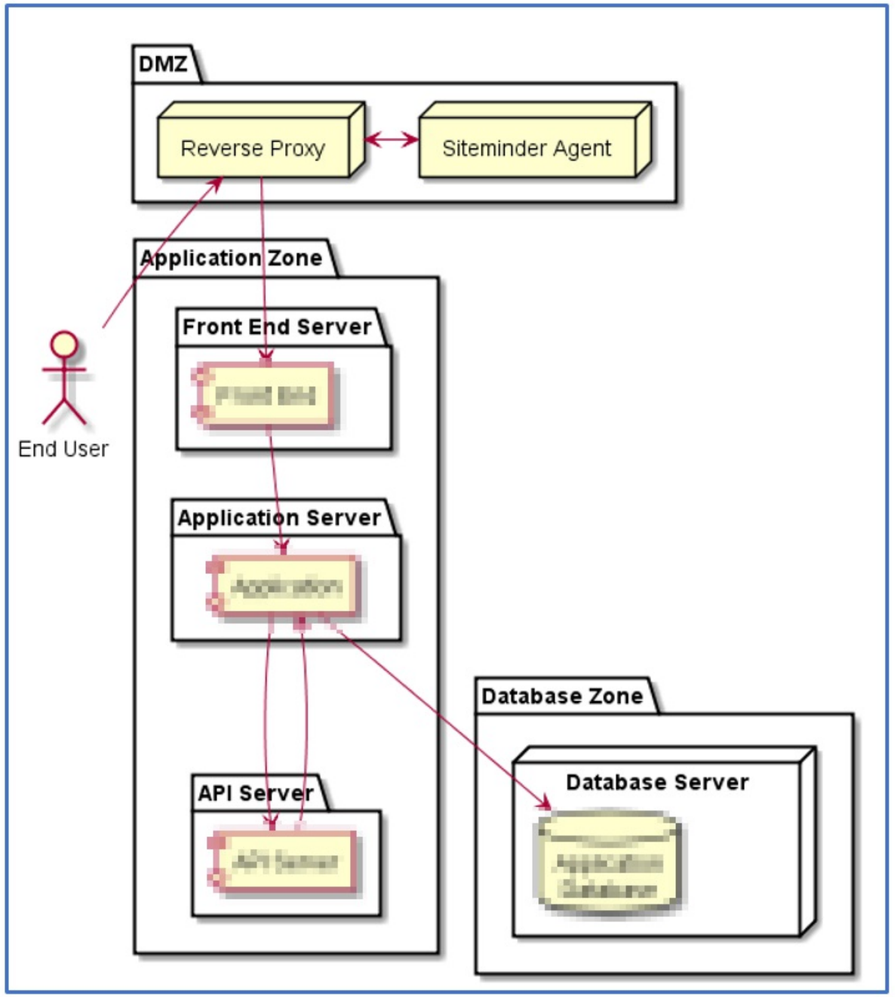
...Dreaded



What Ginger Hears...



What Developers See

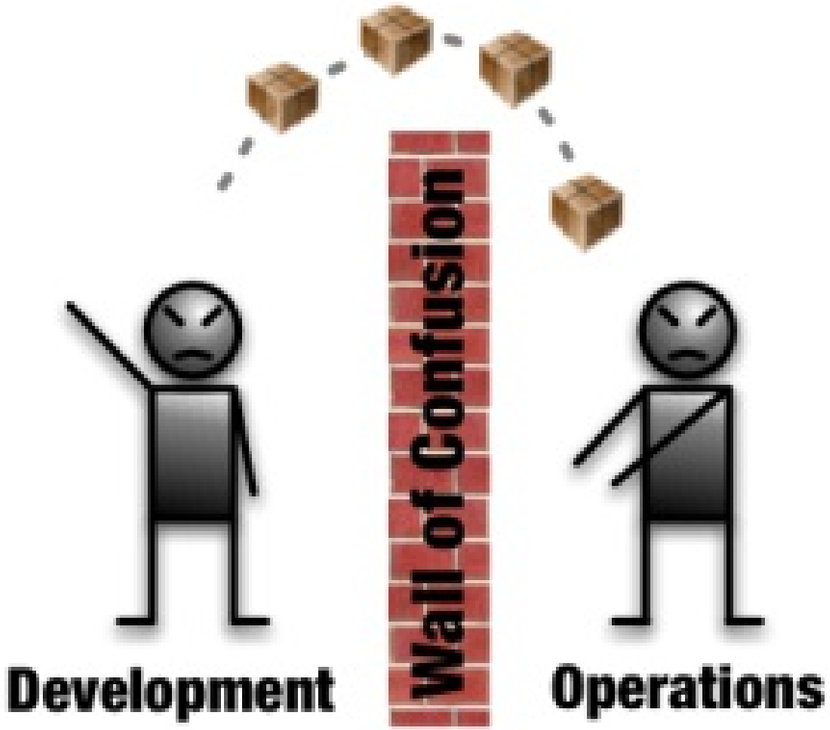


What Ops See

Deployment

The rubber hits the road and...

...so does *The Wall of Confusion*



What goes wrong?

Inconsistent Environments

- Developers build in their world, deliver to a different one
  - Each Dev creates their own development/test capability - best efforts
    - Execution environment doesn't match reality, either does test data
  - Periodically delivers code - usually at a milestone - e.g. UAT
    - Agile methodology SHOULD address this
  - Test data doesn't match production
    - *Non-Functional Requirements*

Impact:

- It works on my machine!

graze

What goes wrong?

Ineffective Communications

- Communication is via Word documents - the dreaded *Release Guide*
  - Premise: To deploy this app, do this...
  - Assumption: The writer knows the readers world...impossible

Impact:

- Steps are performed manually
- On-the-fly adjustments are made...further invalidating the assumption
  - On Dev, Test and Prod

What goes wrong?

Unnecessary Dependencies

- The *iStore* optimization
  - iStores/funding force optimizations on time and cost
  - Method: Few servers, shared resources

Impact:

- Unwanted dependencies between apps
  - Coordination of multple apps because of shared dependencies
  - Outages of an app because of an upgrade to another
- The Release Party

All of which leads to...

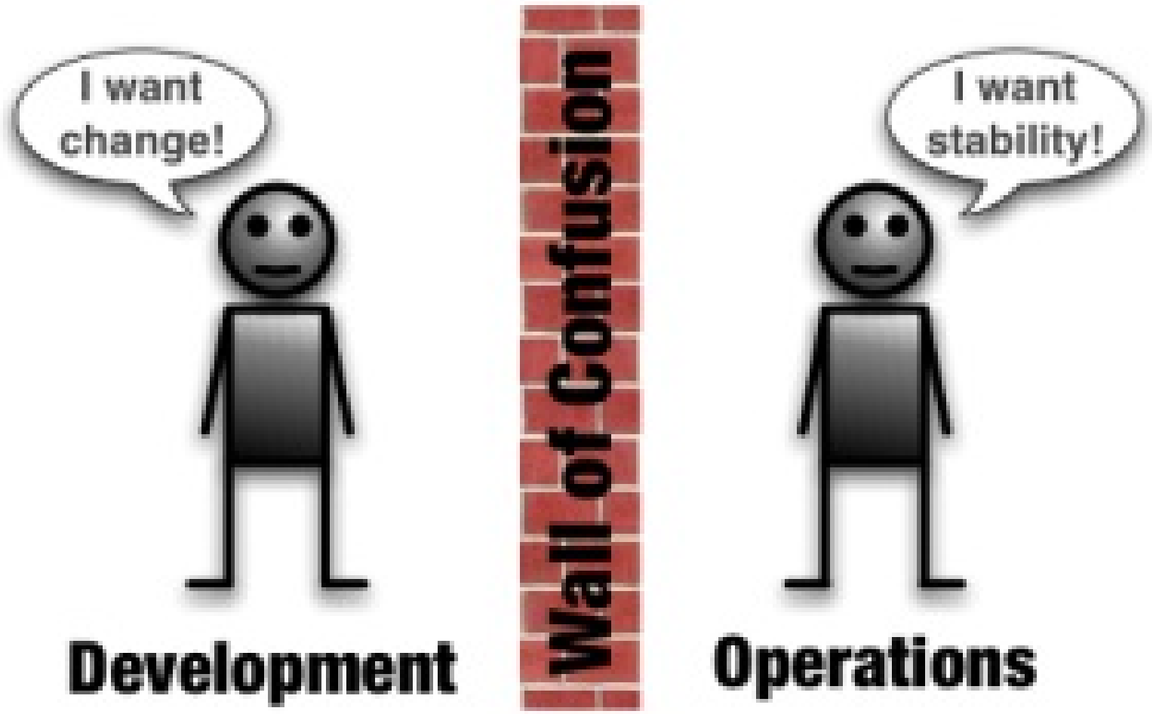
The Day After

Twentieth Century Fox

graze

The Reflex Response

- We are doing it right, we just need to do it *better* next time
- Test more - take longer, check *EVERYTHING*
- Except - the users still want more fixes/capabilities



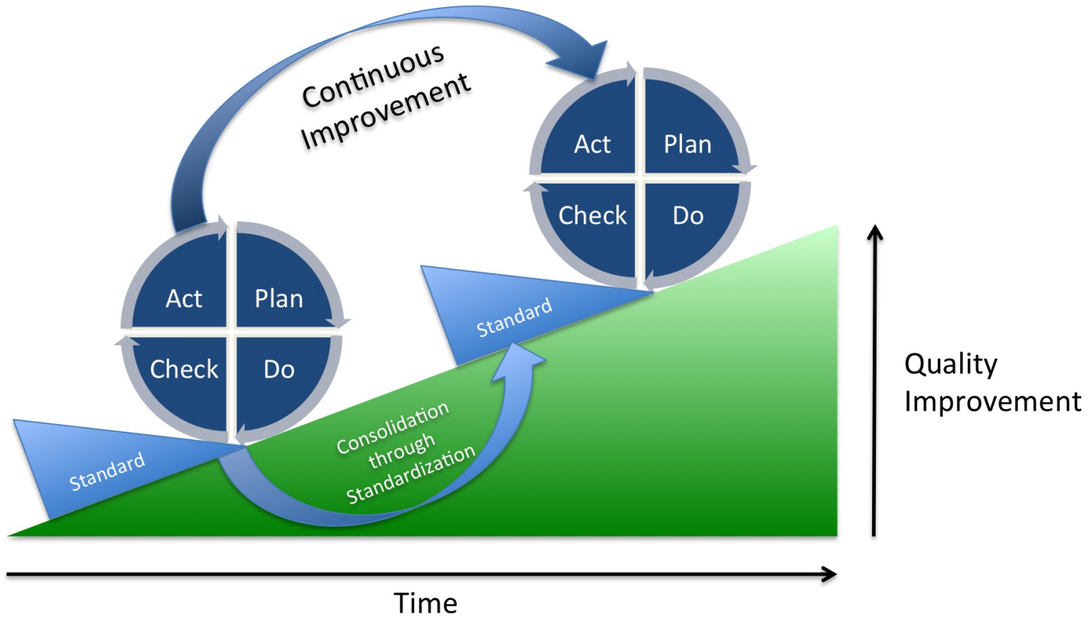
It's a little worse in Government

- Each application is a project - an event
  - Not a product with a lifecycle
  - Focus is on the application, not long term
- Contracted teams
  - Each starts with own approach & tools
  - Highly variable contact with Ops
  - Improvements are local (team) not system-wide
- Limited access to data
  - Production type data
  - Production volumes of data

So...What is DevOps?

The application of Lean principles to the end to end systems:

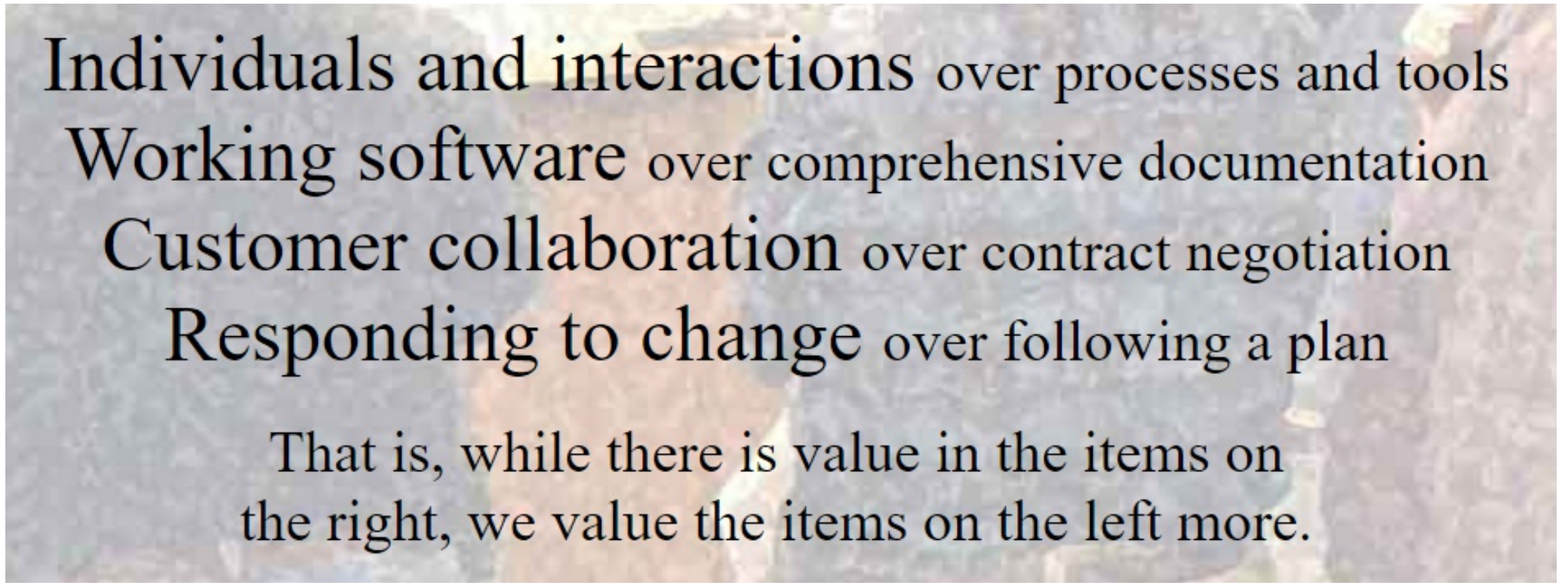
Maximize value; minimize waste



...using some really powerful tools

Analogous to Agile

And to some extent driven by agile...

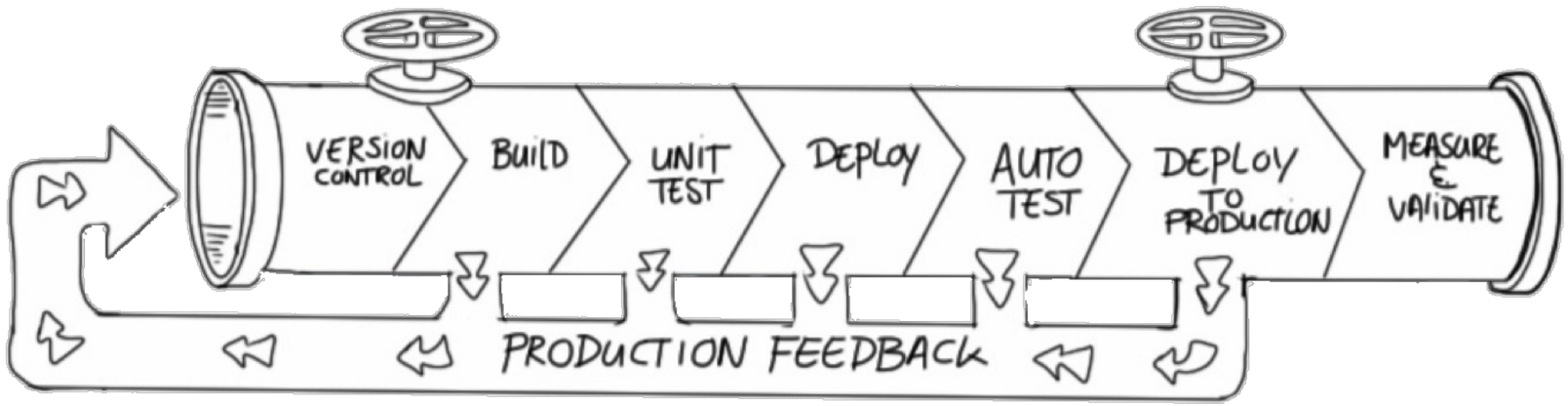


graze

Problem: The Release Guide

- Old: Write It in Word - every step
  - Compile Code
  - Build Code - for each component
  - Test the Build
  - Install pre-requisites
  - Install code
  - Restart components
  - Verify components
- Better: Write it as a repeatable script
  - Not easy - done incrementally - by lazy programmers
  - ...in isolation
- Even Better: Create tools to improve each step

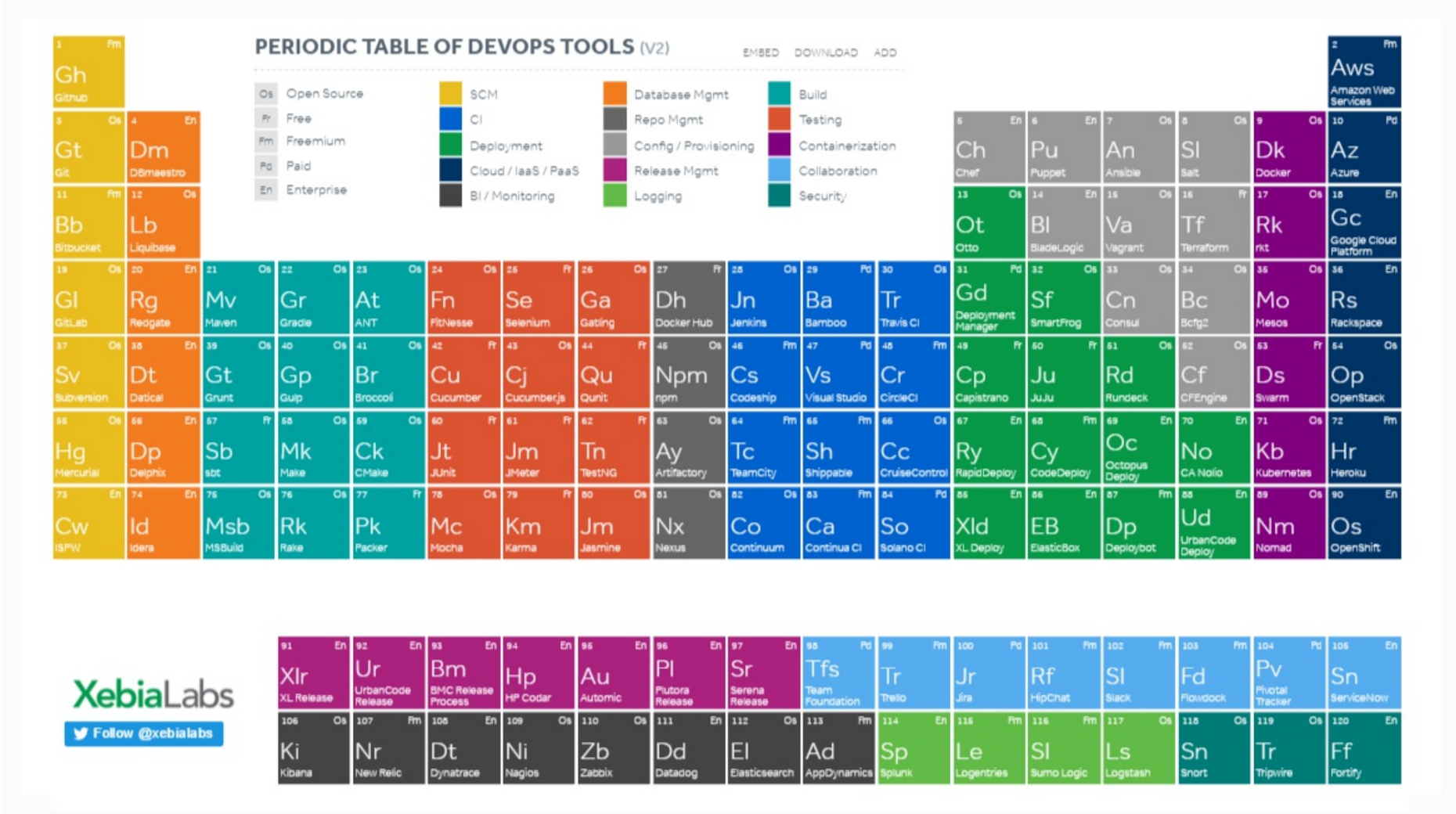
Solution: The Deployment Pipeline



- Subversion, git, github - manage code
- Maven, grunt - build tools
- xUnit - unit test tools
- Selenium, Jmeter - integration test tools
- Migrations, Datical, E-F - database upgrades
- Jenkins - Continuous Integration, job runner

- Connected via triggers

So Many Tools...

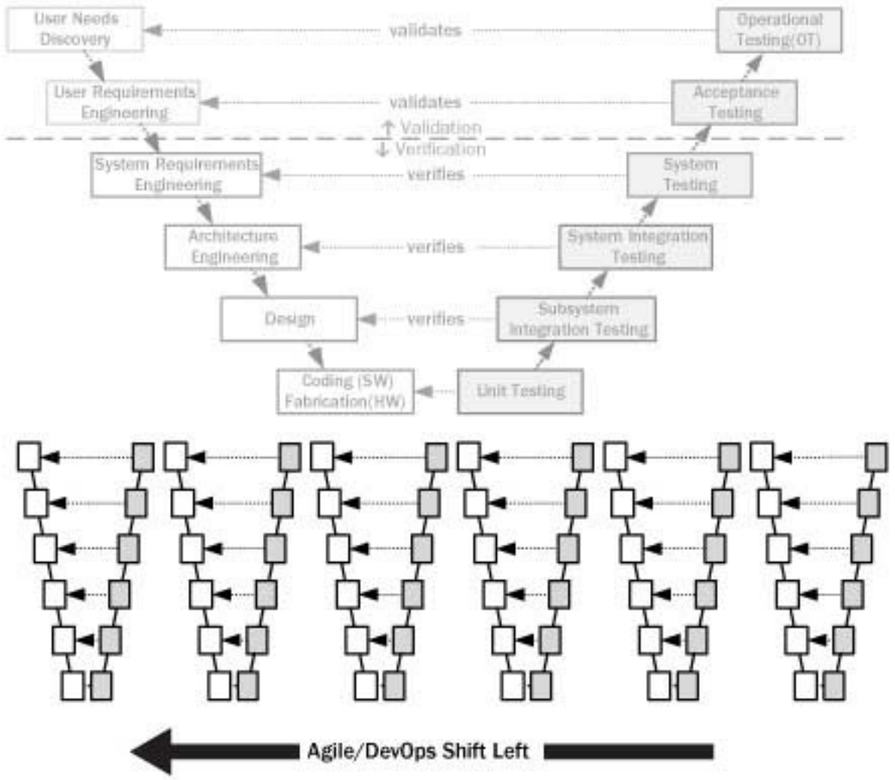


Credit - Xebia Labs - https://xebialabs.com/periodic-table-of-devops-tools/

graze

Problem: The Day After

Solution: Release Early and Often - "*Shift Left*"



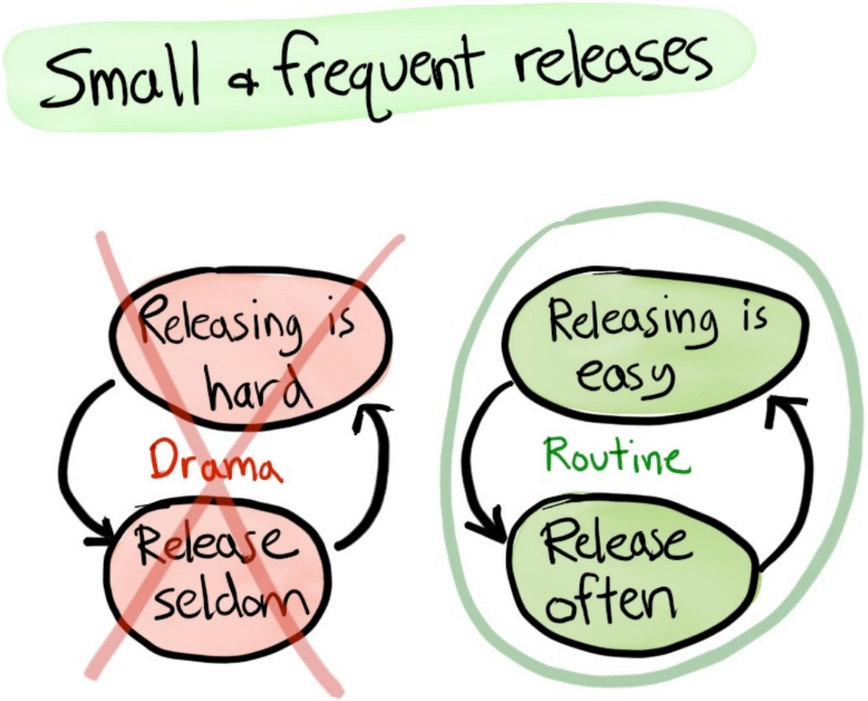
Problem: The Day After

Solution: Really Fast Releases

Done *properly* - aka "Roll-forward"

1. Issue found
2. Issue documented - e.g. JIRA entered
3. Issue investigated
4. Issue fixed, checked in
5. Build / Deploy
6. Verify fix
7. Deploy to Test
8. Verify
9. Deploy To Production...phewwww!!!

Problem: Change is Bad



Big Release - Big Risk - many things to break - hard to fix

Small Release - Small Risk - only a few things to break - easy to fix

Credit: Spotify Engineering Culture - https://labs.spotify.com/2014/03/27/spotify-engineering-culture-part-1/

Problem: Works on my System!

Solution: Consistent Environments

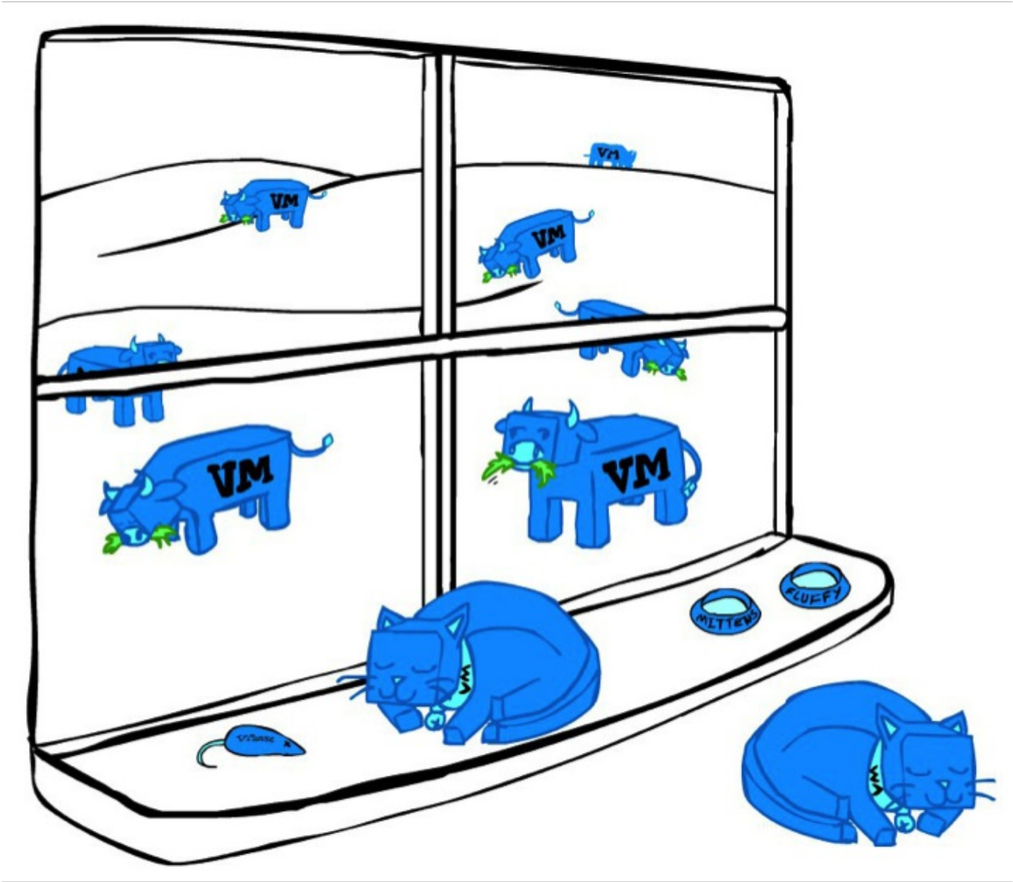
Tools to enable consistency

- Ansible, Puppet, Chef - server setup tools
- Subversion, git, github - configuration as code

Tools so Dev = Production

- Vagrant - VMs
- Docker - Containers

NOTE: Open source licensing *REALLY* helps.



Problem: Unnecessary Dependencies

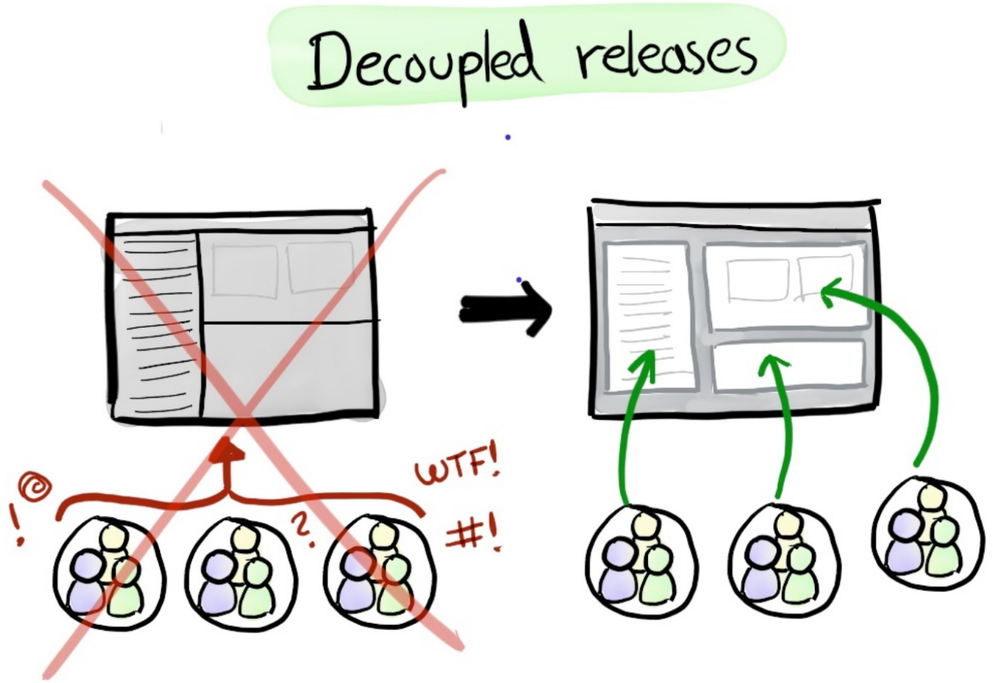
Solution: Stop it!!

Fake Dependencies

- Enterprise Release Scheduling - don't!!
- Eliminate artificial deadlines

Architectural Dependencies

- Isolate apps / parts of apps
  - Different servers ($$$)
  - Docker, etc.
- Don't share databases
  - But don't duplicate data
  - Use APIs

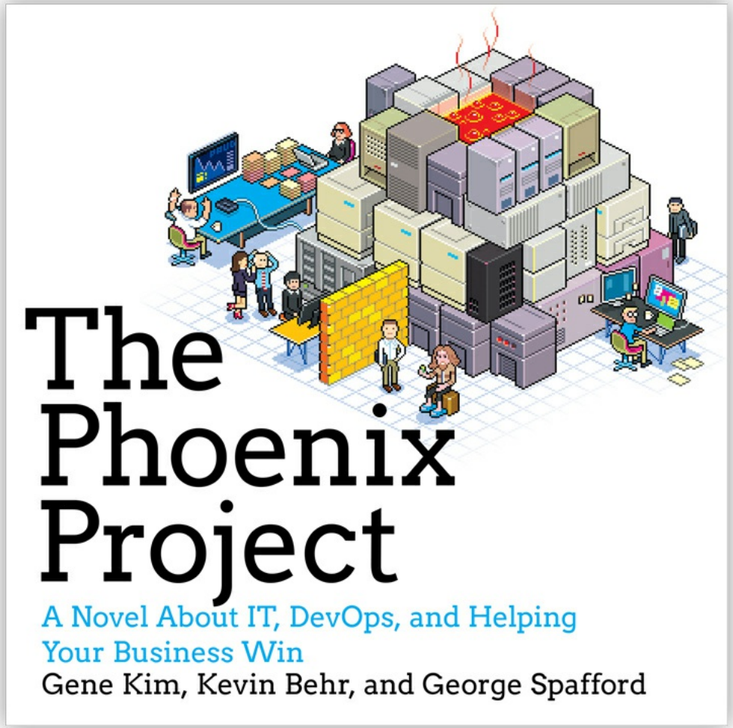


So...what is DevOps?

- A culture of continuous improvement as it relates to the *end to end* delivery of systems
- ...supported by a growing (and standardizing) set of automation tools

The Three Ways

- Systems Thinking
  - Focus on impacts to the *entire* system
- Create Feedback Loops
  - Verify your assumptions/theories
- Continual Experimentation and Learning



Are we doing DevOps??

Anti-Patterns

- No version control
- No discussion of Dev environments
- Cross Project Release Schedules
- Text-based Release Guide
- Post-deployment Fixes without Releases
- Multi-app Release Party Email Chains
- Server Names - pets (vs. *Services*)
- Test Date = Start UAT Date - 1 Day
- Go to Test, go to Production dates
- Production Date = Go Live Date - 1 Day

- Day After syndrome - it hurts!

Patterns

- Version Control for everything
- Devs world ~= Production
- Many, many, many deployments
  - Support for Agile Methodology
  - Automatic deployment to Dev
  - Triggered deployment to Test, Prod
  - No manual steps - *database*

- Early deployments to test, prod
- Automated tests
- Automatic feedback and notifications
- App deployments are independent
- Business drives production releases

- Day After is just like the Day Before

graze