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 Impact on Business?

graze

Approach

The course is a mixture of presentation, labs and discussion. Please, feel free to jump in at a time with questions, comments, suggestions, snores, 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

Introductions

Who are you?

- Project
- Role
- Experience with Digital Services?

Me??

Stephen Curran, *Cloud Compass Computing, Inc.*

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

graze

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

graze

Why is DevOps?

Roots - merging the Work of Developers and Operations Teams

- Developers - make the code
  - User Interface (UI), Business Logic/Rules, Integrations
  - Database (sometimes)
    - Or a Database Group does the design
    - Developers implement and populate the database
- Ops - runs the code
  - Servers
  - Networks
  - Databases

graze

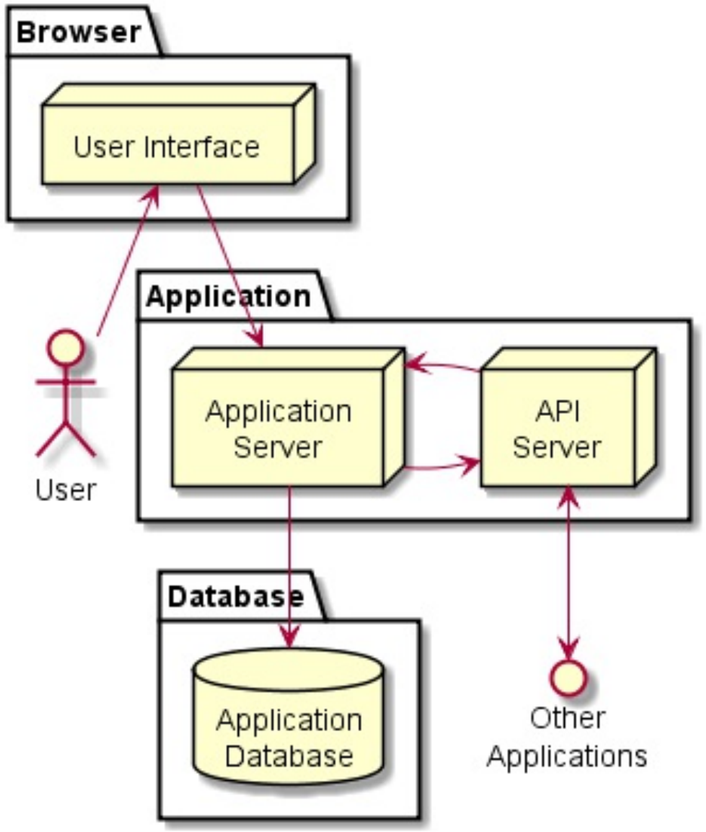
Backup a bit - what's an app?

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...

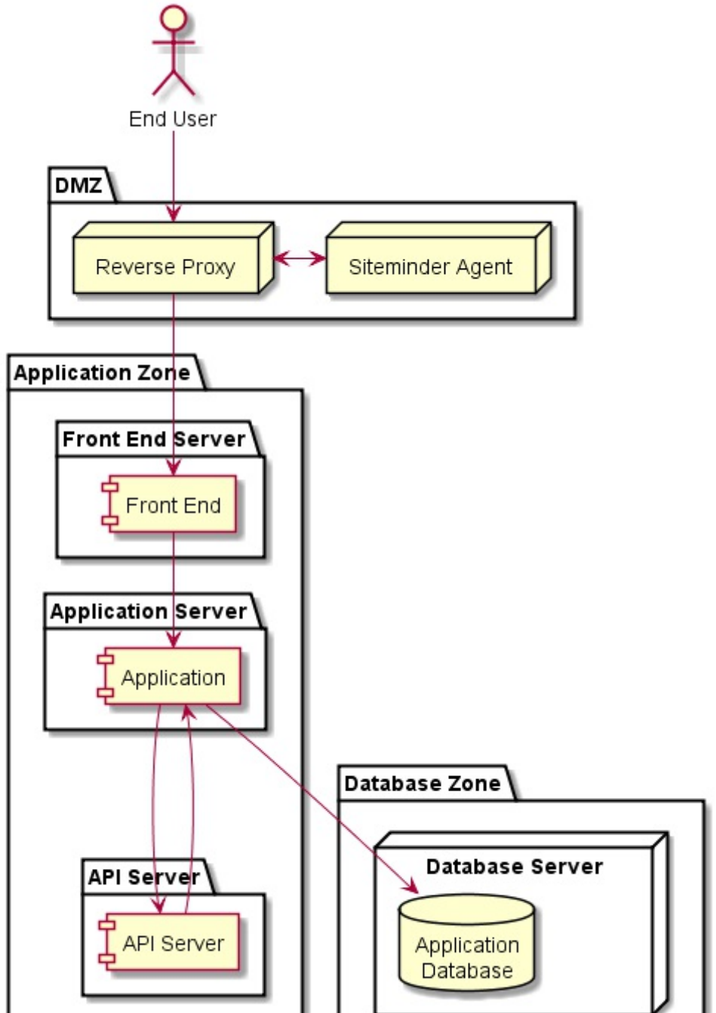


Backup a bit - what is an application?

Ops View

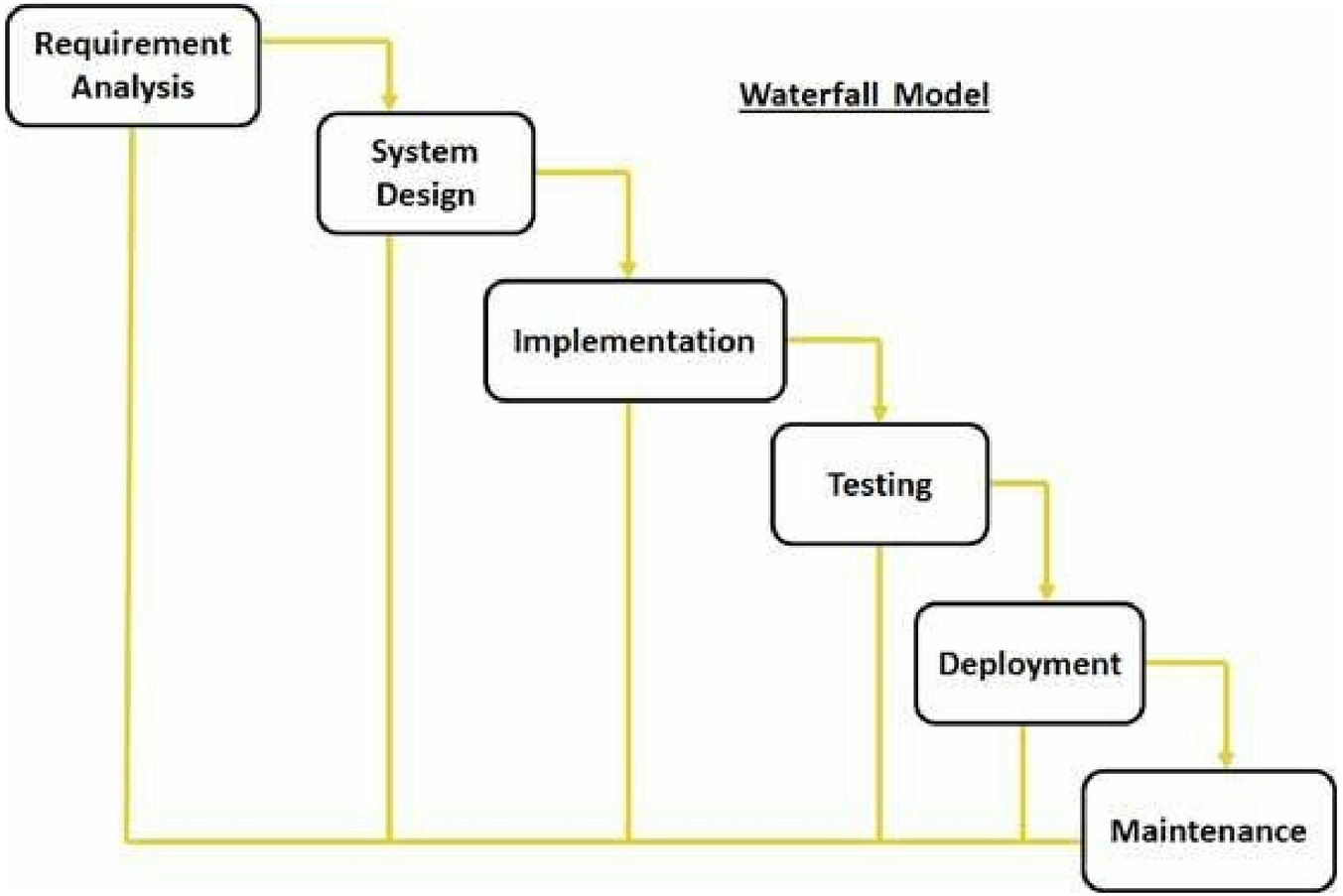
- Networking zones
- URLs - *https://myapp.gov.bc.ca*
- Authentication - siteminder
- Encryption - SSL
- Firewalls
- Servers
- Storage

Times three: Dev/Test/Prod



graze

Making it Work - Theory

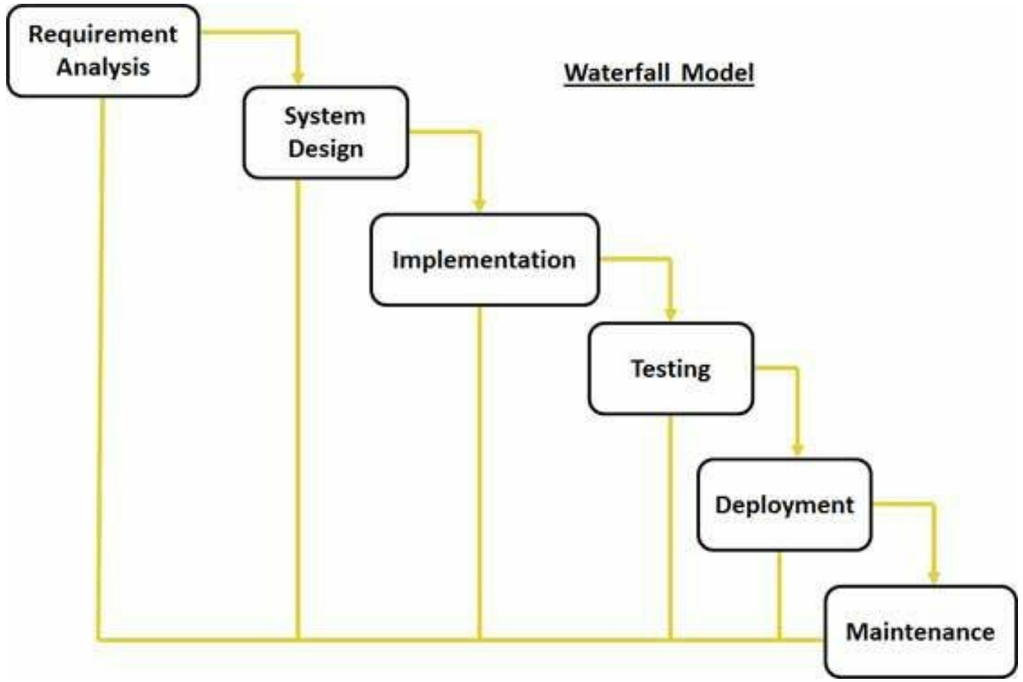


Meetings, documents, agreements and requests

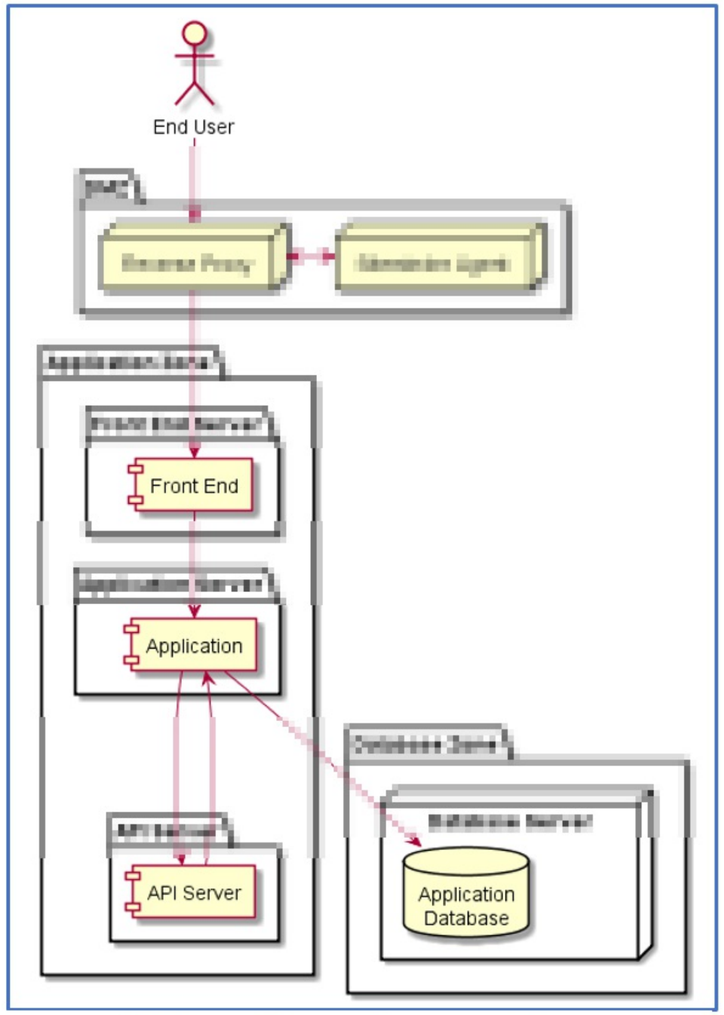
Reality

- Requirements: *change*
- Implementation: *Takes too long*
- Testing: *Skipped*
- Deployment is...

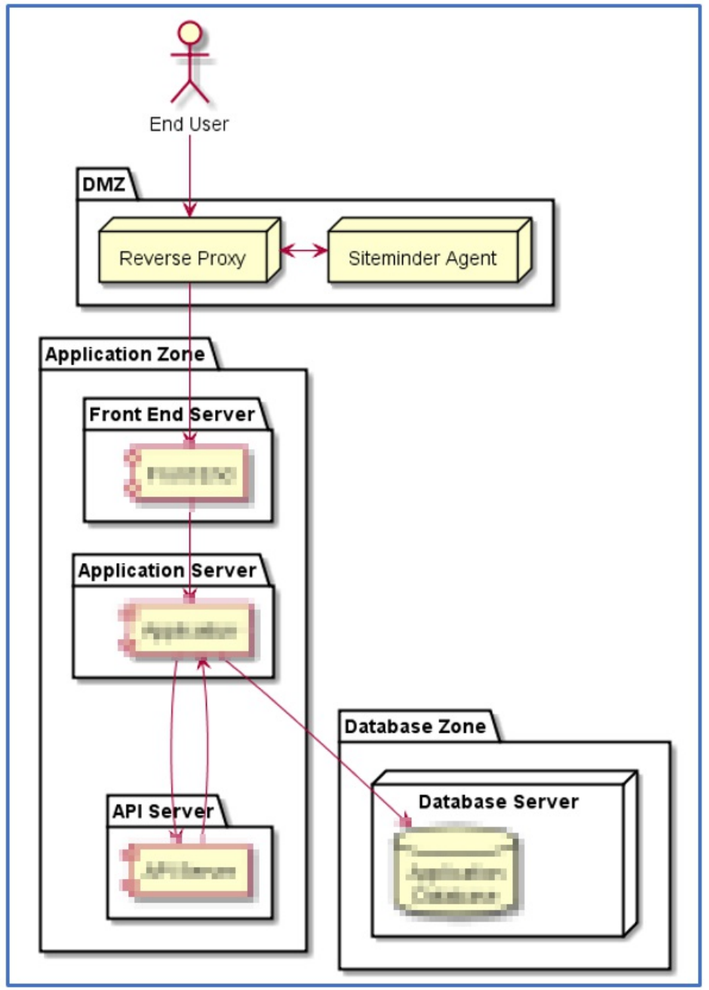
...Dreaded



What Ginger Hears...



What Developers See

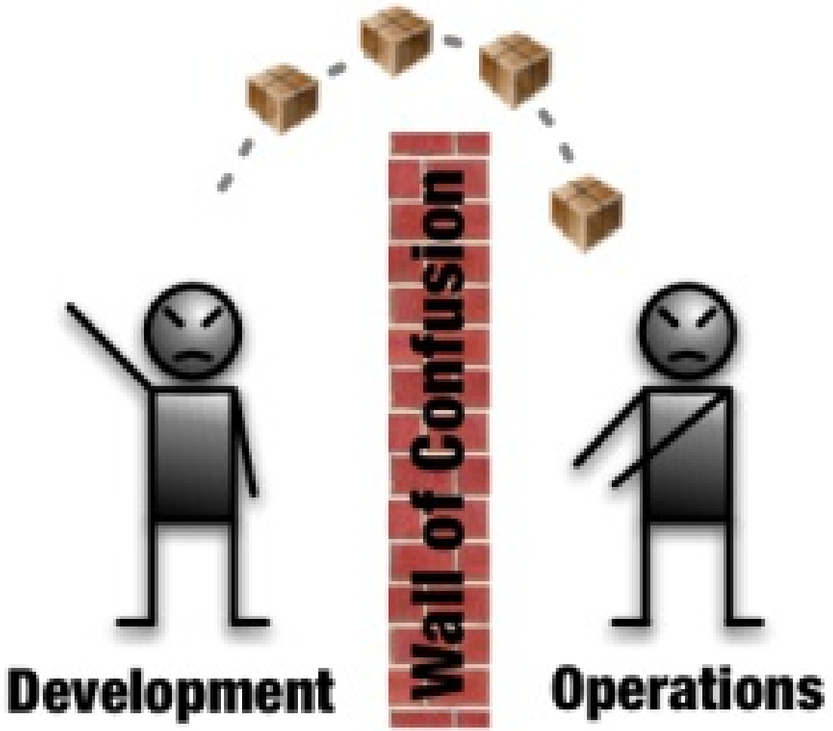


What Ops See

Deployment

The rubber hits the road and...

...so does *The Wall of Confusion*



graze

What goes wrong?

- 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

What goes wrong?

- 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?

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

Which leads to...

- The *iStore* waste
  - To eliminate unwanted dependencies - over-provision - take longer, waste funds
  - NOTE: Particularly tough in the BC Government domain - physical vs. virtual machines

graze

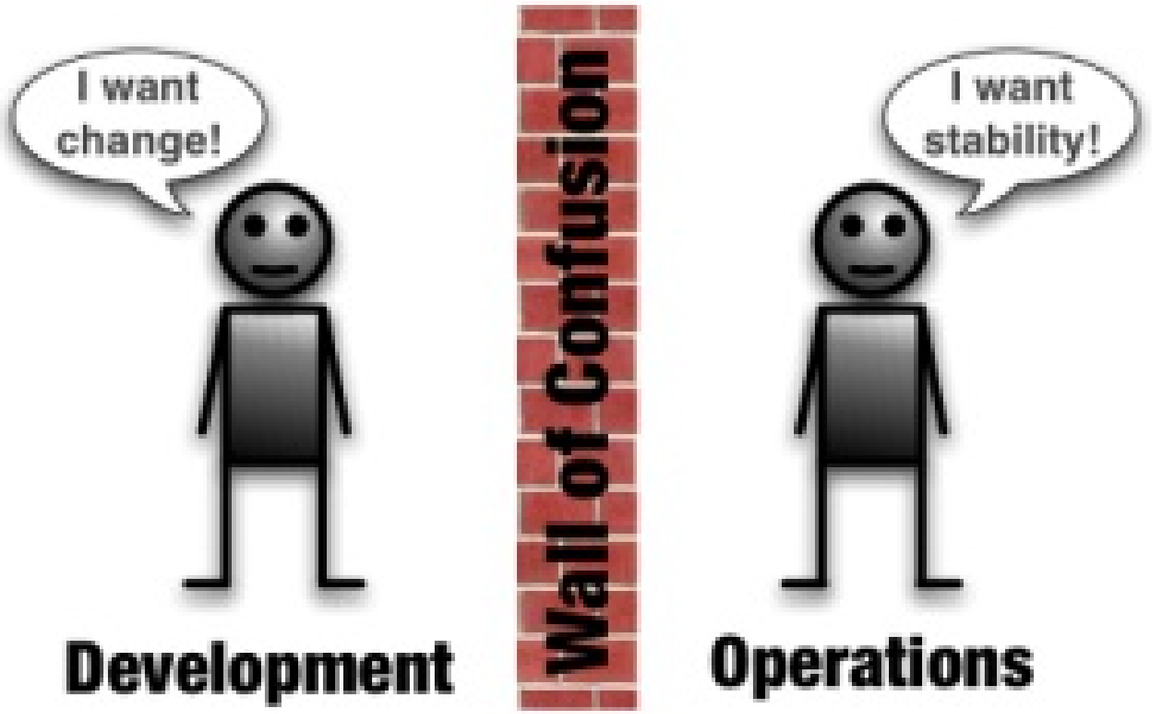
Which leads to...

The release party night, and...

Ops favourite...the Day After

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



graze

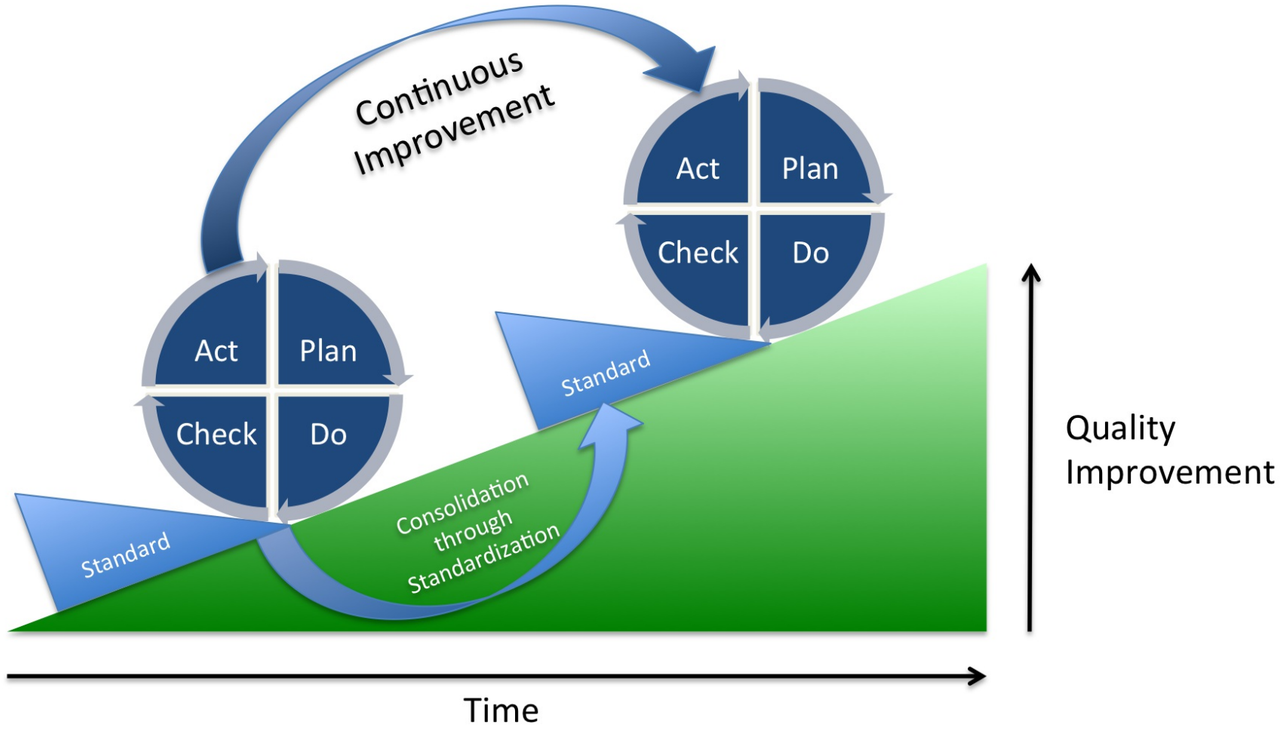
It's a little worse in Government

- Each application is a project - an event
  - Not a product with a lifecycle
  - Focus is on the application, not the delivery/maintenance of the app
- Contracted teams
  - Each starts with own dev approach, tools
  - Highly variable contact with Ops - especially the first time
- Limited access to data
  - Production type data
  - Production volumes of data

graze

So...What is DevOps?

Applying Lean principles to deployment: Maximize value; minimize waste

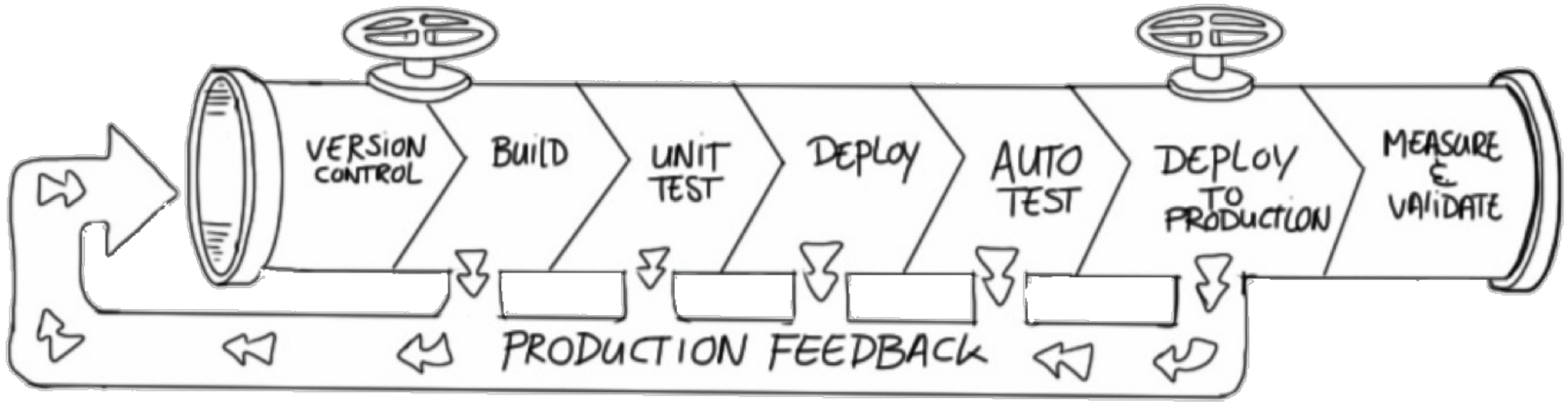


...using some really powerful tools

Problem: The Release Guide

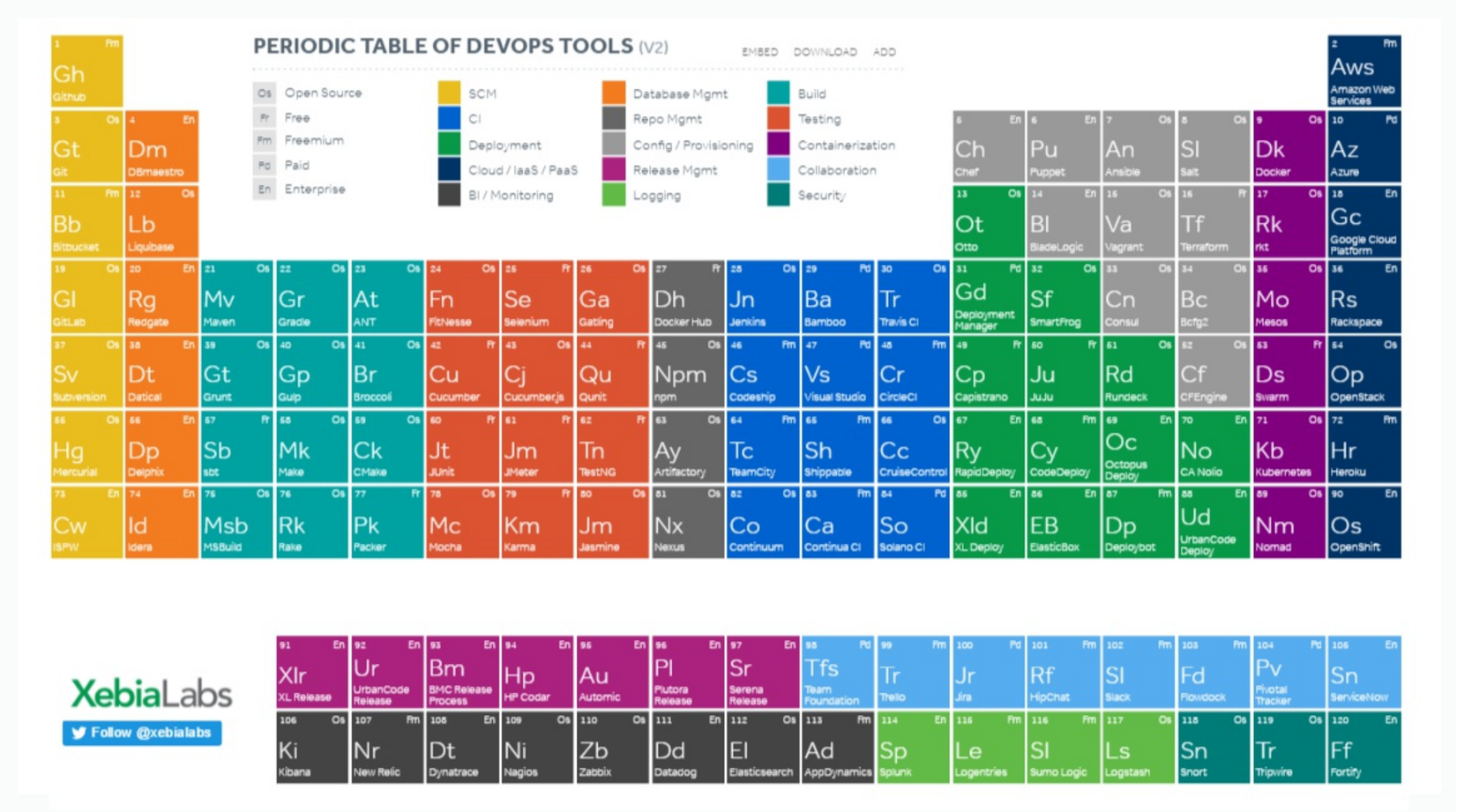
- Old: Write It in Word - every step
  - Compile Code
  - Build Code - for each component
  - Test the Build
  - Install pre-requisites
  - Install code
  - Restart components
  - Verify components
- Better: Write it as a repeatable script
  - Not easy - done incrementally - by lazy programmers
- Even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

graze



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

graze

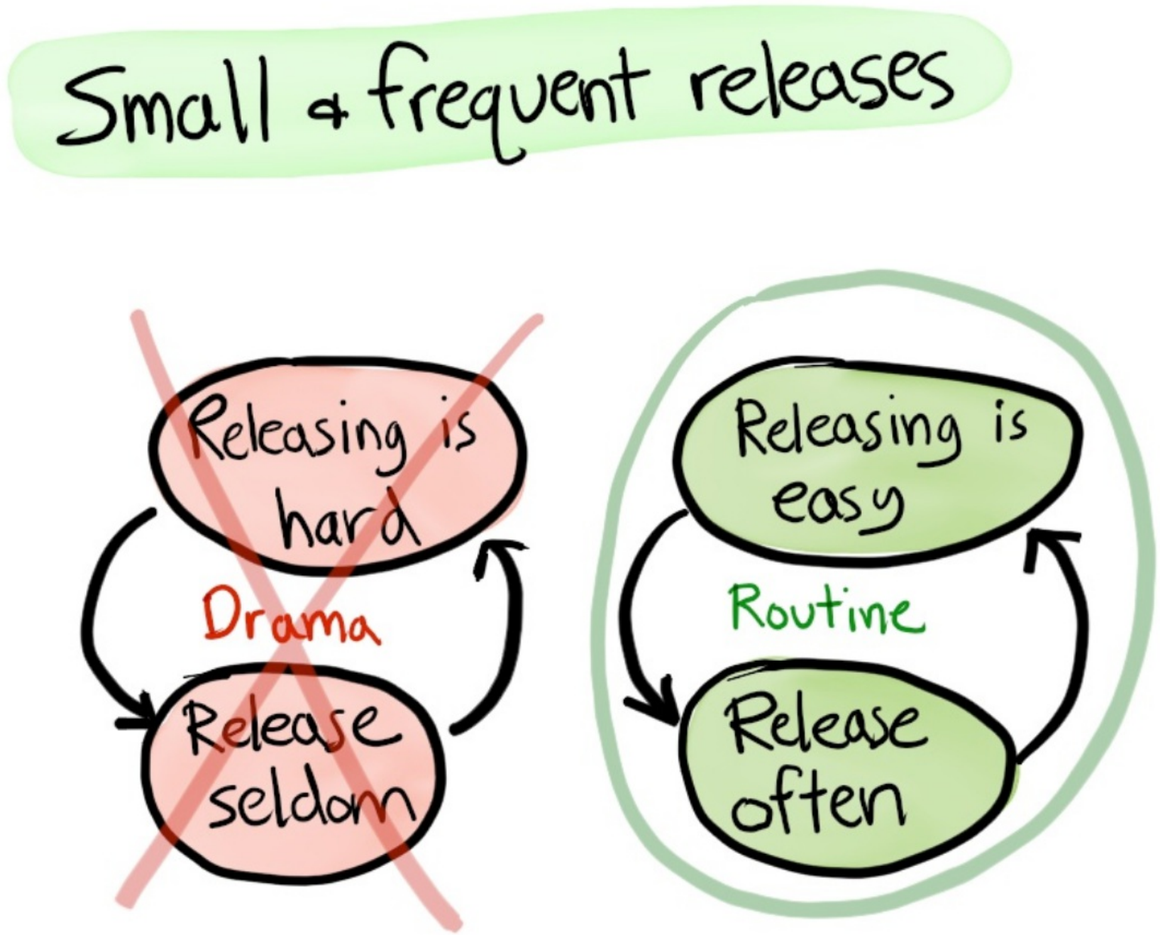
Problem: The Day After

Solution: Really Fast Releases

- Done *properly* - aka "Roll-forward"
  - Issue found
  - Issue documented - e.g. JIRA entered
  - Issue investigated
  - Issue fixed, checked in
  - Build/Deploy
  - Verify fix
  - Deploy to Test
  - Verify
  - Deploy To Production...phewwww!!!
- or Rollback (rare)

graze

Problem: Change is Bad



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

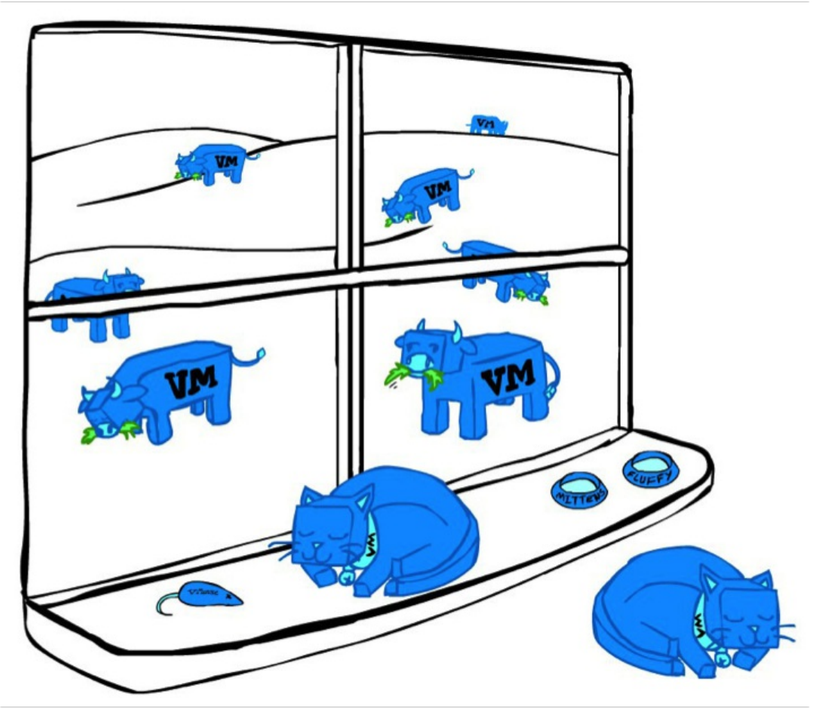
graze

Problem: Works on my System!

Solution: Consistent Environments

- Ansible, Puppet, Chef - server setup tools
- Subversion, git, github - configuration as code
- Vagrant, Docker - VMs (containers) for Developers
- Kubernetes, Docker Compose - Server orchestration

NOTE: Open source licensing *REALLY* helps.

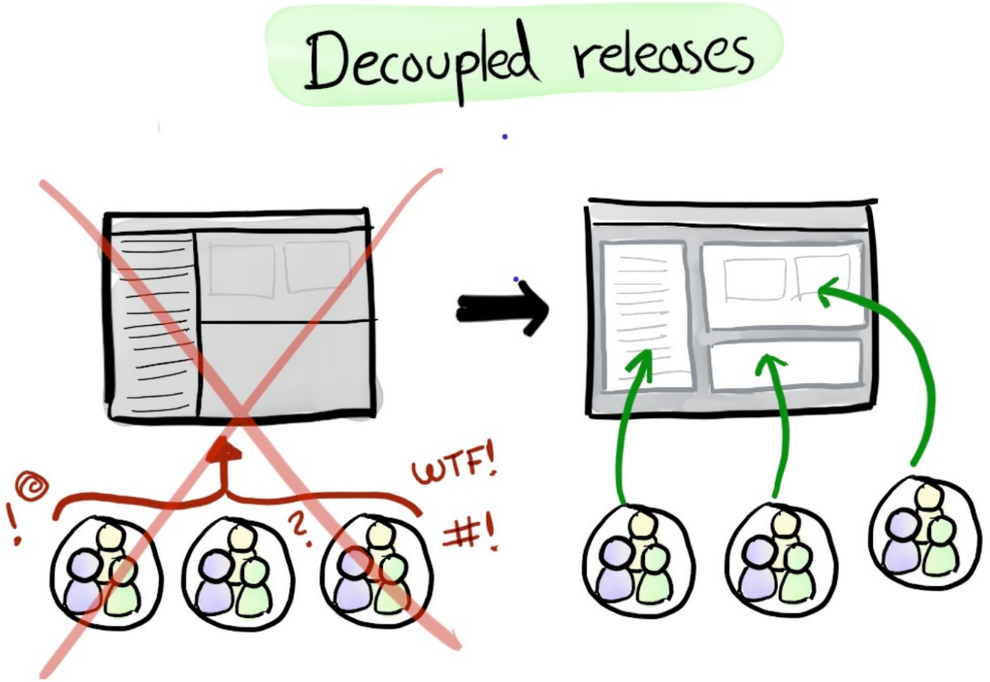


graze

Problem: Dependencies

Solution: Stop it!!

- Enterprise Release Scheduling - don't!!
- Eliminate artificial deadlines
- Don't share resources (servers, etc.) etc.
  - Architectural changes
    - Isolate apps on the same server
      - Docker, etc.
    - Don't share databases
      - Use APIs



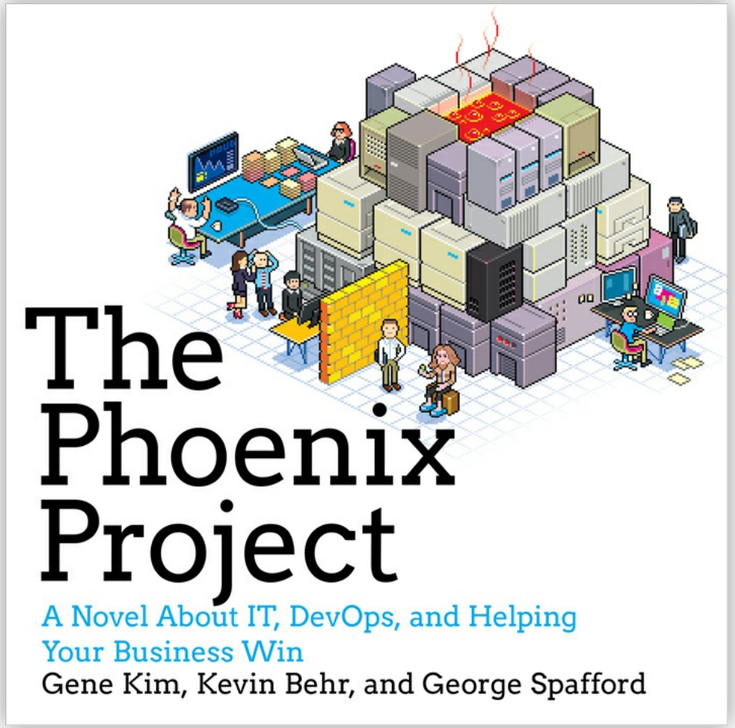
graze

So...what is DevOps?

- A culture of continuous improvement as it relates to the 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



graze

DevOps Indicators

- Automatic Deployment
- Developers environments
-

- four
- five
- six

graze