An OCIO Digital Literacy Course
DevOps For Product Owners

Part 1: The Big Questions

Stephen Curran, Cloud Compass Computing, Inc.

This work is licensed to the public under a Creative Commons Attribution 4.0 license. To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/.

DevOps For Product Owners

Part 1: The Big Questions

1. Introductions
2. What is DevOps?
3. What is the Cloud?
4. Why does it matter to the Business?

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 will allow you to say - *I done DevOps*

Logistics...

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

Introductions

Who are you?

- Project
- Role
- Digital Services experience?

Who Am I?

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

- I walk the line - business and technology
- All about delivery
- Agile Development Leader
- 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 much...

Wikipedia

Why is DevOps?

Roots - merging Developers and Operations work

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

NOTE: DevOps is Development Methodology agnostic, but Agile requires DevOps.

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...

Where does an app run?

Ops View

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

3x: Dev/Test/Prod

Making it Work - Theory

Meetings, documents, agreements and requests

Reality

- Requirements: ***change***
- Implementation: ***takes too long***
  - Devs: Code Development
  - Ops: Procure/Setup Servers
- Testing: ***get skipped***
- As a result, deployment is...

... ***Dreaded***

What Ginger Hears...

What Developers See

What Ops See

Deployment

The rubber hits the road and...

...up pops *The Wall of Confusion*

What goes wrong?

Inconsistent Environments

- Developers develop in their world, deliver to a different one
  - Each Dev creates their own development/test capability
    - Execution environment doesn't match production
    - Minimal test data
  - Periodically delivers code - usually at a milestone - e.g. UAT
    - Agile *should* fix that
  - Test data doesn't match production

Impact:

- It works on my machine!

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 compensations 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: multiple businesses invovled in one release

All of which leads to...

The Day After

redit: Twentieth Century Fox - *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*
- For now,though... **DON'T CHANGE ANYTHING!!**

- 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 app, not the long term

- Contracted teams
  - Each starts with own approach & tools
  - Highly variable contact with Ops
  - Improvements are local (team)

- Ops employees
  - Work with many teams...they come and go

- Limited access to data
  - Production-type data
  - Production data volumes

So...What is DevOps?

A culture of applying 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...

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

The Agile Manifesto - http://agilemanifesto.org/

Problem: The Release Guide

- Old/Miserable: Write It in Word - every step (example: ICM)

Table of Contents

1.0 Release Summary 2
1.1 Release Package Content 3
1.2 Change Control / Release Information 4
2.0 Release Migration 5
2.1 Dependent Software 5
2.2 Migration Steps 5
WebMethods Migration 5
Version Control 5
Resources Required 5
Migration Order 5
2.3 PL/SQL Migration 6
2.3.1 PL/SQL Migration Steps 6
3.0 Database 8
4.0 Configuration 9
5.0 Other 10
6.0 Rollback Plan 11

- Better: Write it as a repeatable script
  - Not easy - done incrementally - by lazy programmers
  - ...in isolation

- Even Better: Create and share 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...

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

Problem: The Day After

Traditional: Testing V

Problem: The Day After

Solution: Test (and deploy) 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

redit: 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
- Chaos Monkey - "no pets" verification

Tools so Dev = Production

- Vagrant - VMs
- Docker - Containers

Open source licensing *REALLY* helps here

Problem: Unnecessary Dependencies

Solution: Stop it!!

Fake Dependencies

- Enterprise Release Scheduling - don't!!
- Eliminate artificial deadlines
- Dependencies on people
- Dependencies on products/licenses

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
- Devs environments not like Prod
- "Agile" but no DevOps
- Cross Project Release Schedules
- Text-based Release Guide
- Post-deployment fixes without Deploys
- Multi-app Release Party Email Chains
- Server Names - pets (vs. *Services*)
- Test Date = Start UAT Date - 1 Day
- Production Date = Go Live Date - 1 Day

- Day After Syndrome - it hurts!

Patterns

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

- Early and often to test, prod
- Automatic feedback and notifications
- App independence

- Day After is just like the Day Before

What's Important For the Business

Raising the "anti-pattern" questions

- How often are we releasing?
- How soon can we deploy to Test? to Prod?
  - What is the risk?
  - Is there a way to mitigate that risk?

- How long does a full release take?

Lowering the risk to YOUR project

Lab - Deploying an App

Scenario To Do Web App

- Version control - github
- Architecture - typical Web App

Task Deploy Application

What we learned

- Automated deployment is possible
- Non-Technical people can do it
- There are lots of moving pieces
  - Unthinkable to do all the steps manually

What we glossed over...

- The code
- Server setup
- Database setup
- Network setup
- Security setup
- Testing
- Integrations
- Documentation

Summary

- Architecture
- Developing/Managing Code
  - Version Control
  - Github
  - Open Source
  - Issue Tracking
- Environments
  - Servers
  - Networks
  - Storage
  - Security
- Continuous Integration / Continuous Delivery (CI/CD)
  - Build
  - Test
  - Deploy
  - Verify
  - Monitor
- Visualizing it all

Side Note: The impact of Open Source on DevOps

What is the Cloud?

- The "Cloud" is the promise of the future
  - Everybody says so
  - What does that mean?
- Often discussed at the same time as DevOps
  - What's the connection?

Let's talk Cloud in the same context

* as a Service

Once you aren't using your computer for computer purposes - you are using the Cloud

- Dropbox / Google Drive - File Storage Service
- Office 365 / Google Docs - Office Editing Service
- Netflix / CraveTV - Video Entertainment Service

Since all gov't apps aren't on our computers - they are all "in the Cloud"

So let's talk * as a Service - IaaS, PaaS, SaaS

Pizza as a Service

On-Premise, IaaS, PaaS, SaaS

HPAS Services / BC Gov't Data Centres

- On-premise-ish, with iStore orders
- Physical and virtual machines (VMs)
- Standard images - Windows, Linux
- Standard software - e.g. Oracle, .NET
- No containers (Docker)
- Extra services - patching, monitoring
- Backups and restores
- New! IaaS-ish Offering

IaaS * AWS (Amazon), Azure

- Now in Canada
- Procurement: credit card, website
- By use billing - minute and gigabyte
- Spin up servers, configure storage/network
- Resilency - multiple regions (data centres)
- API brings the power (more later...)

And a lot more services...

PaaS: HPAS OpenShift, AWS, Azure

- Dynamic setup and configuration
  - Spin up app components
  - Network them
  - True cattle - that die (and get replaced)
  - The cattle are less stable - but that's Ok

- New technologies
  - Containers (Docker and others)
  - Orchestration (Kubernetes and others)

Digression! Physicals, VMs, Containers

Goals: Resource optimization, consist execution - same everywhere

Container Benefits

- Easy isolation - *feels like a whole computer*
- Really lightweight - high density deployment - many on one host
- Create once, run everywhere - the "Shipping Container" analogy
  - Run containers on dev machine and on Production - same thing!

But...

- New and evolving quickly
- Complex to manage at scale - which is where they are most useful
- True use...as building blocks

Digression! What is Orchestration?

- What if *Front End 1* node crashes?
- What if the load goes up? down?
- What if the main database crashes?
- What if we want to deploy an update?

Orchestration

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
name: rss-site
spec:
replicas: 2
template:
metadata:
labels:
app: web
spec:
containers:
– name: front-end
image: nginx
ports:
– containerPort: 80
– name: rss-reader
image: nickchase/rss-php-nginx:v1
ports:
– containerPort: 88
```

PaaS: HPAS OpenShift, AWS, Azure

- Dynamic setup and configuration
  - Spin up app components
  - Network them
  - True cattle - that die (and get replaced)
  - The cattle are less stable - but that's Ok

- New technologies
  - Containers (Docker and others)
  - Orchestration (Kubernetes and others)

Devs dream

- Create code
- Declare configuration, evolve it easily
- Declare requirements
- Deploy easily

Challenges

- New technology
- New techniques
- For now: Open Source only

SaaS: Office 365, Salesforce

- End users login, use the services
- Some integration with the Enterprise - Single-Sign On (SSO)

Salesforce - also a PaaS

- Roots as SaaS - Sales automation system
- Coming to Canada - running on AWS Canada instance
- Has evolved into a PaaS - **Force.com**
  - Heavy on the "configure" model vs. code - especially the backend
  - Option to build custom frontend talking to Force.com

BC Gov't Salesforce deployments (MTICS, MSDSI, JAG) but have been challenging

- Data Centres in the US - so no personal information
  - Informational apps - no stored data
  - Data Residency handling for personal information

* as a Service Options for Business

Government hosted services are (sort of) easy

- Traditional - with iStores and a DIY-attitude - you are on your way
- IaaS - with a DIY-attitude - you are on your way
- PaaS - with an I-want-to-learn-attitude - you are on your way

Cloud Options are becoming possible:

- Azure, AWS IaaS are in Canada
- Azure and AWS - many other services
- Cloud BC option is coming
- Salesforce will soon be in Canada - on AWS
- Concern is governance - where are all the apps, where is the data?

Cloud Summary and Directions

Four major Cloud models - On-Premise and I, P and S aaS

- HPAS supports (mostly) On-Premise, but also IaaS and PaaS (via Red Hat OpenShift)
- New public cloud options are coming - but not easily obtained
  - Cloud BC could enable those options
  - Potential: Private/Public Cloud - capabilities extend to use Azure/AWS resources
- Direction is towards PaaS - user expectations make requirements too complex for any other approach

Review

Why is DevOps?

- We looked at:
  - Complexity in deploying applications using documentation and manual processes
  - Application Architecture
  - Why traditional approaches are hard for Devs and Ops
  - Why traditional approaches are prevalent in Government - project-focus

What is DevOps?

- We learned:
  - A culture of applying Lean principles to the end to end systems
  - Reduce risk by addressing it as early as possible
  - Backed by lots of powerful tools - largely developed and shared in the open
  - Lab - deploying a complex Web Application - easily

Review

What is the Cloud?

- We discovered:
  - All the Services (and perhaps some digressions...)
  - Options for delivering applications on platforms
  - In BC - HPAS is the main choice, but more options are coming
  - Regardless, the underpinnings are DevOps
    - Declare what you need - let the platform figure it out

How does DevOps and the Cloud Impact Business?

- Summary:
  - Style of developing (agile), delivering apps (DevOps) - iterative, small chunks
  - Control over when to deliver
  - The potential for shorter cycles - idea to business value
  - Options for where to deploy apps

End of Part 1