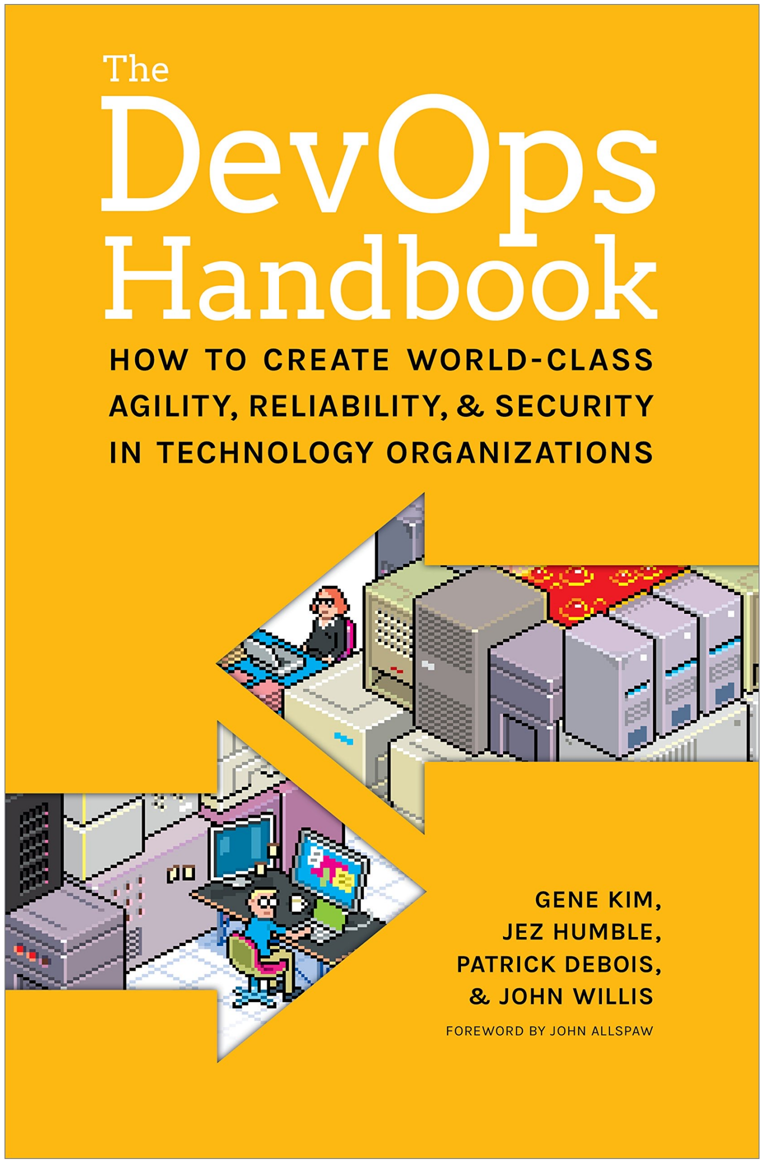
CSCI 5409 Cloud Computing – Fall, 2023
Week 9 – Lecture (Oct 30, 2023)

# DevOps and Release Management

Dr. Lu Yang
Faculty of Computer Science
Dalhousie University
luyang@dal.ca

## Housekeeping and Feedback

- Start recording


The DevOps Handbook
HOW TO CREATE WORLD-CLASS AGILITY, RELIABILITY, & SECURITY IN TECHNOLOGY ORGANIZATIONS
GENE KIM, JEZ HUMBLE, PATRICK DEBOIS, & JOHN WILLIS
FOREWORD BY JOHN ALLSPAW

# DevOps & Managing Releases

[Image Cover] Kim, G., Debois, P., Willis, J., Humble, J., & Allspaw, J. The DevOps handbook.

## Lecture Summary

- Learning outcomes
  - Understand the value add of DevOps practices in cloud computing software development
  - Understanding the complexities of cloud computing development, and how DevOps practices protect you from catastrophic failure
  - Exploring common release management practices

- Plan:
  1. Agile refresher
  2. DevOps and the first way
  3. Kahoot fun!
  4. Break
  5. The other two ways
  6. Release management strategies
  7. Kahoot fun!
  8. Advice for introducing DevOps practices to your future workplace

## Average release process for web/cloud apps

1. **Release planning** – The development team and business decide what new features to implement (or existing bugs to fix) in the sprint.
2. **Sprint begins** – Developers write code / develop tools to achieve release goals.
3. **Testing** – The app is deployed to a test server; QA looks for bugs.
4. **Staging** – The app is staged on a production-like environment, further testing occurs.
5. **Release** – The app is deployed to the production environment. QA performs regression testing and signs off on the release.

This is just the average release process, there are many variants and additional processes that can add complexity:

- Strict compliance regulations (for example credit cards – PCI, data privacy, corporate reporting requirements, etc.)
- Information Security checks

# Agile Refresher

- **DevOps is Agile++**
- 12 Agile principles (my emphasis added where I feel organizations often fail):

1. Early and continuous delivery of **valuable** software.
2. **Welcome** changing requirements, even late in development.
3. Deliver **working** software frequently.
4. Business people and developers must **work together** daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and **trust them** to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.
7. **Working** software is the primary measure of progress.
8. Sustainable development, ability to **maintain a constant pace**.
9. **Continuous** attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from **self-organizing** teams.
12. At **regular intervals**, the team reflects on how to become more effective, then tunes **and adjusts** its behavior accordingly.

These principles are **EQUALLY** important to agility.

## The Downward Spiral…

Most organizations ignore, or don't value equally, all agile principles except "early and continuous delivery of software"

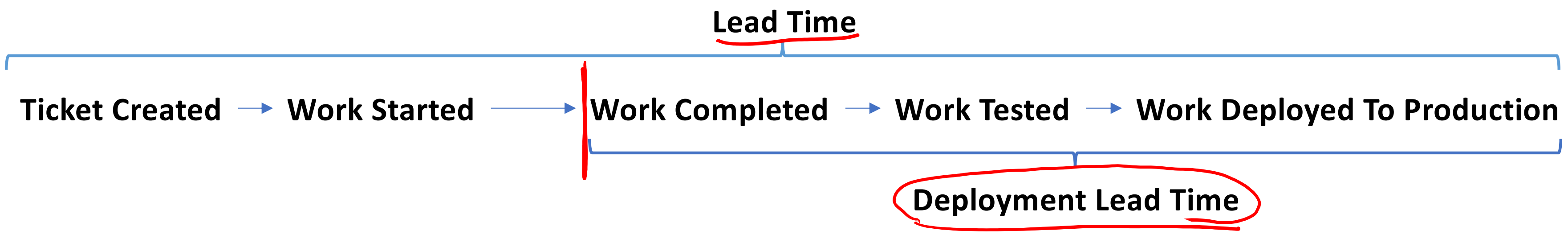
**Act 1:** Your organization rushes to build something that delivers value to customers, **in your rush you accrue technical debt**

**Act 2:** Someone commits the tech team to deliver upon a new urgent promise, the tech team cuts corners to meet the deadline and **more debt is added to the pile**

**Act 3 – Final Act:** Everything becomes a little more difficult, bit by bit. Our code is now tightly coupled, small changes cause big failures. We become fearful and less tolerant of changes. Approvals and meetings pile up. **Our organization loses agility and is out competed. Customers leave. Our software dies.**

Paraphrased from: Kim, G., Debois, P., Willis, J., Humble, J., & Allspaw, J. The DevOps handbook. Page xxvi.

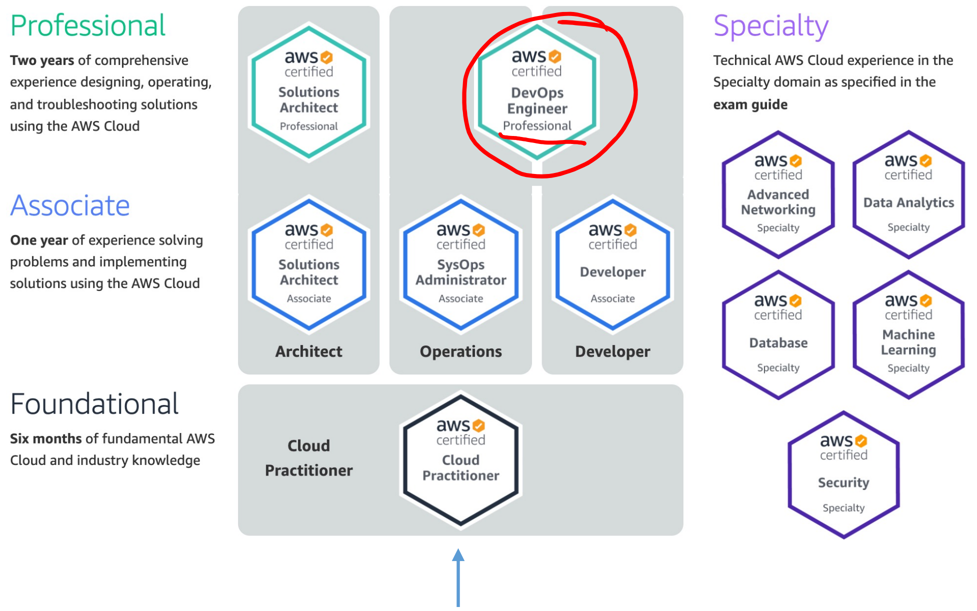
# The Technology Value Stream



- **Lead Time** = Time from customer request to customer receipt, the shorter this is the more competitive your organization becomes
- **Deployment Lead Time** = Time from developer committing work to version control to work running in production
- DevOps is concerned about reducing **Deployment Lead Time**
- Our DevOps Ideal: ***Deployment Lead Time of MINUTES***

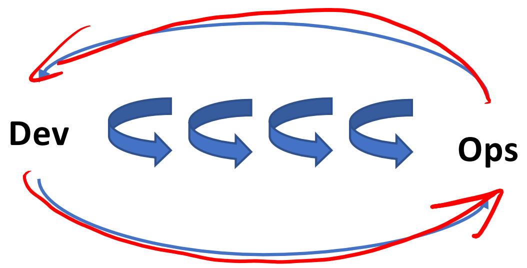
# Why is DevOps crucial to Cloud Computing?

- As we learned in our Deployment & Delivery Model lecture, provisioning complex architectures in the cloud adds complexity to our projects, especially in areas that affect deployment lead time
- If we do not plan for this, and adopt DevOps practices, our deployment lead time grows and grows until our organization is no longer competitive
- Once our organization is no longer competitive our software dies and we're out of work!


Professional
Two years of comprehensive experience designing, operating, and troubleshooting solutions using the AWS Cloud
Associate
One year of experience solving problems and implementing solutions using the AWS Cloud
Foundational
Six months of fundamental AWS Cloud and industry knowledge
aws certified Solutions Architect Professional
aws certified DevOps Engineer Professional
aws certified Solutions Architect Associate
aws certified SysOps Administrator Associate
aws certified Developer Associate
Architect
Operations
Developer
Cloud Practitioner
aws certified Cloud Practitioner
Specialty
Technical AWS Cloud experience in the Specialty domain as specified in the exam guide
aws certified Advanced Networking Specialty
aws certified Data Analytics Specialty
aws certified Database Specialty
aws certified Machine Learning Specialty
aws certified Security Specialty

DevOps is so critical to cloud computing that AWS has their own certification path for it. At the end of this course, you will be very close to the requisite knowledge to take the Certified Cloud Practitioner exam!

## The Three Ways [1]

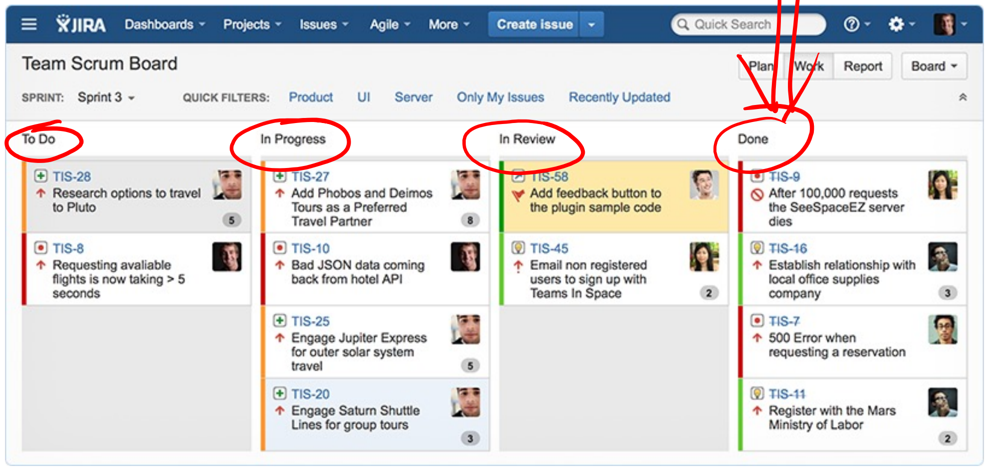


- **The Principles of Flow:** Enabling fast left-to-right flow of work from Development to Operations to the customer.
- **The Principles of Feedback:** Enabling fast and constant flow of feedback from right-to-left at all stages of the value stream.
- **The Principles of Continuous Learning and Experimentation:** Enabling the creation of a generative, high-trust culture that supports a dynamic, disciplined, and scientific approach to experimentation and risk-taking, facilitating the creation of organizational learning, both from our success and failures.

[1] Kim, G., Debois, P., Willis, J., Humble, J., & Allspaw, J. The DevOps handbook. Pages 11 – 13.

# The First Way: The Principles of Flow

- **Make our work visible:**
  - "Unlike physical processes, in the technology value stream we cannot easily see where flow is being impeded or when work is piling up in front of constrained work centers."[1]
  - We must use tools to track our work:
    - Kanban boards
    - Sprint planning boards
  - Benefits:
    - Identify and solve bottlenecks
    - Measure lead time
    - Visibility allows all stakeholders to more easily prioritize work in the context of global goals



Team Scrum Board

To Do
- TIS-28 Research options to travel to Pluto (5)
- TIS-8 Requesting avaliable flights is now taking > 5 seconds

In Progress
- TIS-27 Add Phobos and Deimos Tours as a Preferred Travel Partner (8)
- TIS-10 Bad JSON data coming back from hotel API
- TIS-25 Engage Jupiter Express for outer solar system travel (5)
- TIS-20 Engage Saturn Shuttle Lines for group tours (3)

In Review
- TIS-58 Add feedback button to the plugin sample code
- TIS-45 Email non registered users to sign up with Teams In Space (2)

Done
- TIS-9 After 100,000 requests the SeeSpaceEZ server dies
- TIS-16 Establish relationship with local office supplies company (3)
- TIS-7 500 Error when requesting a reservation
- TIS-11 Register with the Mars Ministry of Labor (2)

An example scrum board, tickets move left to right.

Image - https://medium.com/tiket-com/scrum-board-physical-or-virtual-9a18c014c575

[1] Kim, G., Debois, P., Willis, J., Humble, J., & Allspaw, J. The DevOps handbook. Pages 15 - 25

## Principles of Flow (Continued)

- **Limit work in progress (WIP):**
  - Studies have shown that context switching is **terrible** for developers and increases risk of errors [1]
  - Limiting WIP makes it easier to see problems, when delayed it is better to solve the delay than to start something new. Solving the delay provides future lasting efficiency benefits. [2]
  - "Stop starting. Start Finishing." [3]
- **Reduce batch sizes:**
  - "Small batch sizes result in less WIP, faster lead times, faster detection of errors, and less rework." [2]
  - The bigger the change you release, the more difficult it is to diagnose and fix errors
- **Reduce the number of handoffs** (e.g. Dev -> QA -> Info Sec -> Ops):
  - Knowledge is inevitably lost with each handoff
  - Solve by automating as much as possible **so team members can deliver value to the customer themselves**, instead of having to constantly depend on others [2]

[1] https://www.apa.org/research/action/multitask
[2] Kim, G., Debois, P., Willis, J., Humble, J., & Allspaw, J. The DevOps handbook. Pages 18 - 21.
[3] David J. Anderson, Kanban: Successful Evolutionary Change for Your Technology Business

## Principles of Flow (Continued)

- **Continually identify and elevate our constraints:**
  - Bottlenecks cause work to pile up, or workers to be starved waiting
  - Typical bottlenecks:
    - **Environment creation:** Automate this with infrastructure as code!
    - **Code deployment:** Automate deployments, goal of 100% automation
    - **Testing:** Automate tests, hire QA developers not button pushers. Parallelize so they run fast.
    - **Overly tight architecture:** Decouple, create loosely coupled systems so that changes can be made safely and with more autonomy
  - "When there are no constraints, we are limited only by the number of good business hypotheses we create and our ability to develop the code necessary to test these hypotheses with real customers."[1]
  - **This is what most businesses do not want to do;** it is difficult for business deciders to understand the benefit of this work because it is not customer facing. **TOUGH!**

[1] Kim, G., Debois, P., Willis, J., Humble, J., & Allspaw, J. The DevOps handbook. Page 23.

## Principles of Flow (Continued)

- **Eliminate hardships and waste in the value stream:**
  - Burnout:
    - Manual repetitive processes
    - Practices that lead to excessive bugs/debugging
    - Constantly being in a state of putting out fires
    - Constantly needing "heroics" to release value to customers
  - Burnout makes good employees leave; this destroys your organization over time **(Dead sea effect)**
  - Swarm these issues and get rid of them! [1]
    - Partially done work → Becomes obsolete and loses value over time
    - Extra processes → Rarely add value and add effort increasing lead times
    - Extra features ("gold plating") → Add effort to testing and managing functionality
    - Task switching
    - Waiting
    - Defects → The longer the time between defect creation and detection, the more difficult it is to resolve the defect and the more time it has to cause damage
    - Nonstandard or manual work → Automate everything! Become obsessed with this.

[1] Kim, G., Debois, P., Willis, J., Humble, J., & Allspaw, J. The DevOps handbook. Pages 24 - 25

## The Strangler Application Pattern

- Coined by Martin Fowler, author of Refactoring: Improving the Design of Existing Code, from observation of strangler vines in Australia
- When your architecture is too tightly coupled or "monolithic" it becomes difficult to change, fear and uncertainty grows, cementing you in place
- Place existing functionality behind an API with an interface, unchanged
- Implement new functionality using desired architecture, making calls to the old system where necessary, implementing new versions of the API when possible
- Eventually it will be replaced, and you can deprecate / decommission