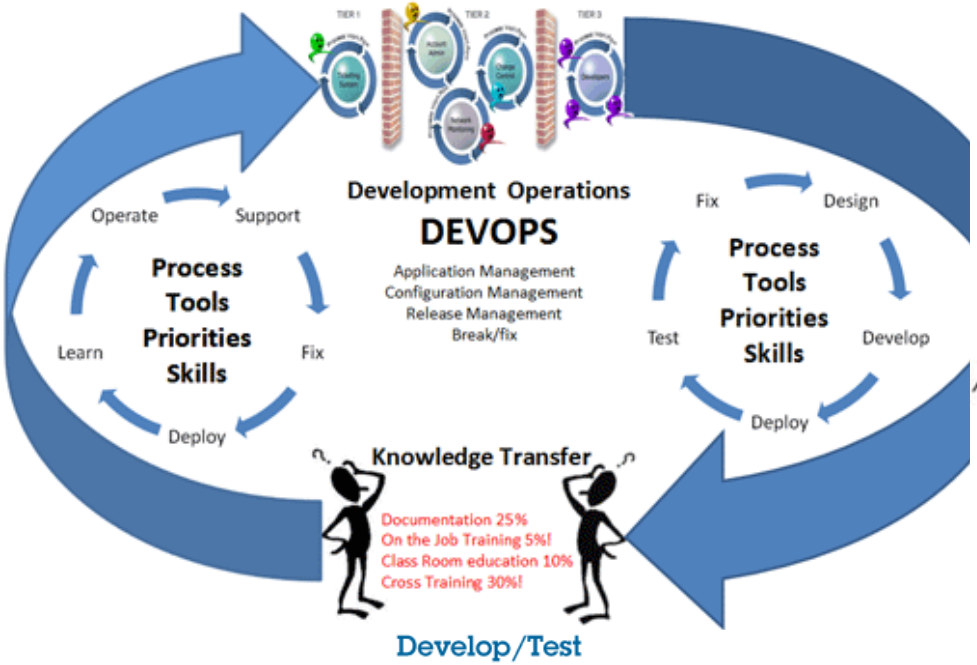
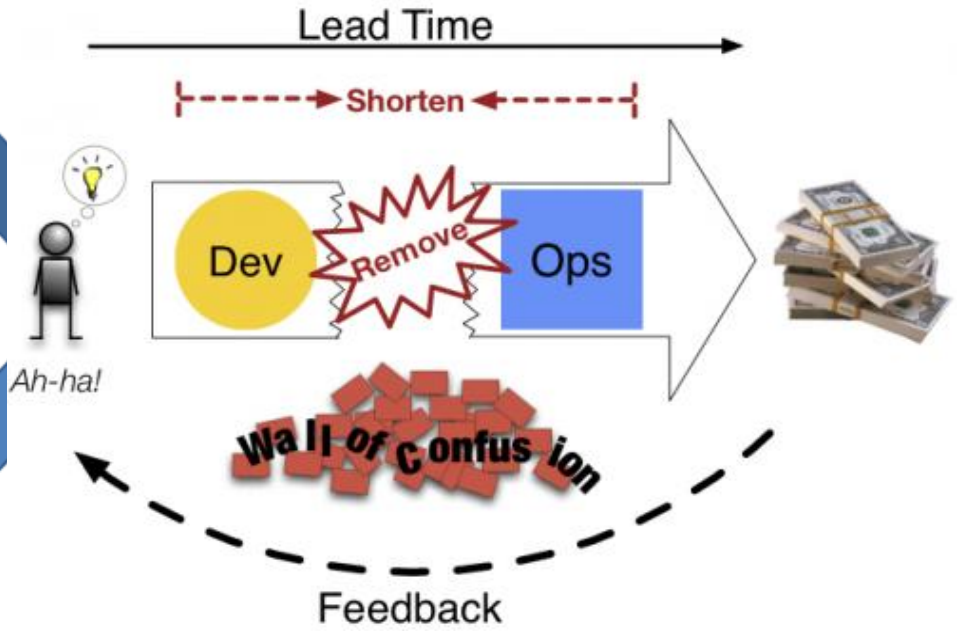
DevOps

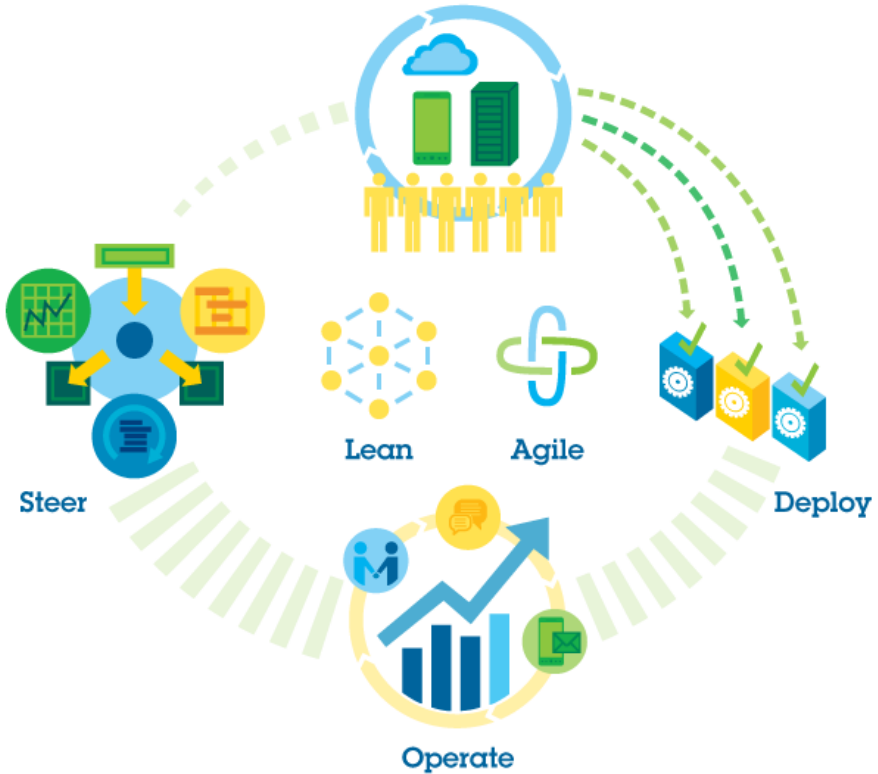
Questions I Will Answer Today

- What Is DevOps?
- What Problems Will DevOps Help Me Solve?
- How Do I Get Started?
- What Mistakes Can I Avoid?

Who Am I?


Development Operations
DEVOPS
Application Management
Configuration Management
Release Management
Break/fix
Operate
Support
Process
Tools
Priorities
Skills
Learn
Fix
Deploy
Fix
Design
Process
Tools
Priorities
Skills
Test
Develop
Deploy
Knowledge Transfer
Documentation 25%
On the Job Training 5%!
Class Room education 10%
Cross Training 30%!
Develop/Test


Lead Time
Shorten
Dev
Remove
Ops
Ah-ha!
Wall of Confusion
Feedback


Steer
Lean
Agile
Deploy
Operate


WSO2 Stratos
Project and Team Management
Software development workflow
WSO2 App Factory
Governance and Compliance
Development Dashboards
Develop Code
Source Control
Issue Tracking
Continuous Build
Continuous Integration
Test Automation
Continuous Deployment
puppet labs
JUnit
Se
sonar
EMMA
Bamboo
Maven
WSO2 App Factory

The Problem In A Nutshell

- Everything needs software.
- Software runs on a server to become a service.
- Delivering a service from inception to its users is too slow and error-prone.
- There are internal friction points that make this the case.
- This loses you money. (Delay = loss)
- Therefore IT is frequently the bottleneck in the transition of “concept to cash.”

Symptoms

- Defects released into production, causing outage
- Inability to diagnose production issues quickly
- Problems appear in some environments only
- Blame shifting/finger pointing
- Long delays while dev, QA, or another team waits on resource or response from other teams
- “Manual error” is a commonly cited root cause
- Releases slip/fail
- Quality of life issues in IT

Why Does This Problem Exist?

- "Business-IT Alignment?"
- The business has demanded the wrong things out of IT
  - Cost sensitive
  - Risk averse
- IT has metastasized over time into a form to give the business what it's said it wants
  - Centralized and monolithic
  - Slow and penny wise, pound foolish

But Then We Demanded Innovation

DevOps Defined

- **DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support.**
- **DevOps is also characterized by operations staff making use many of the same techniques as developers for their systems work.**

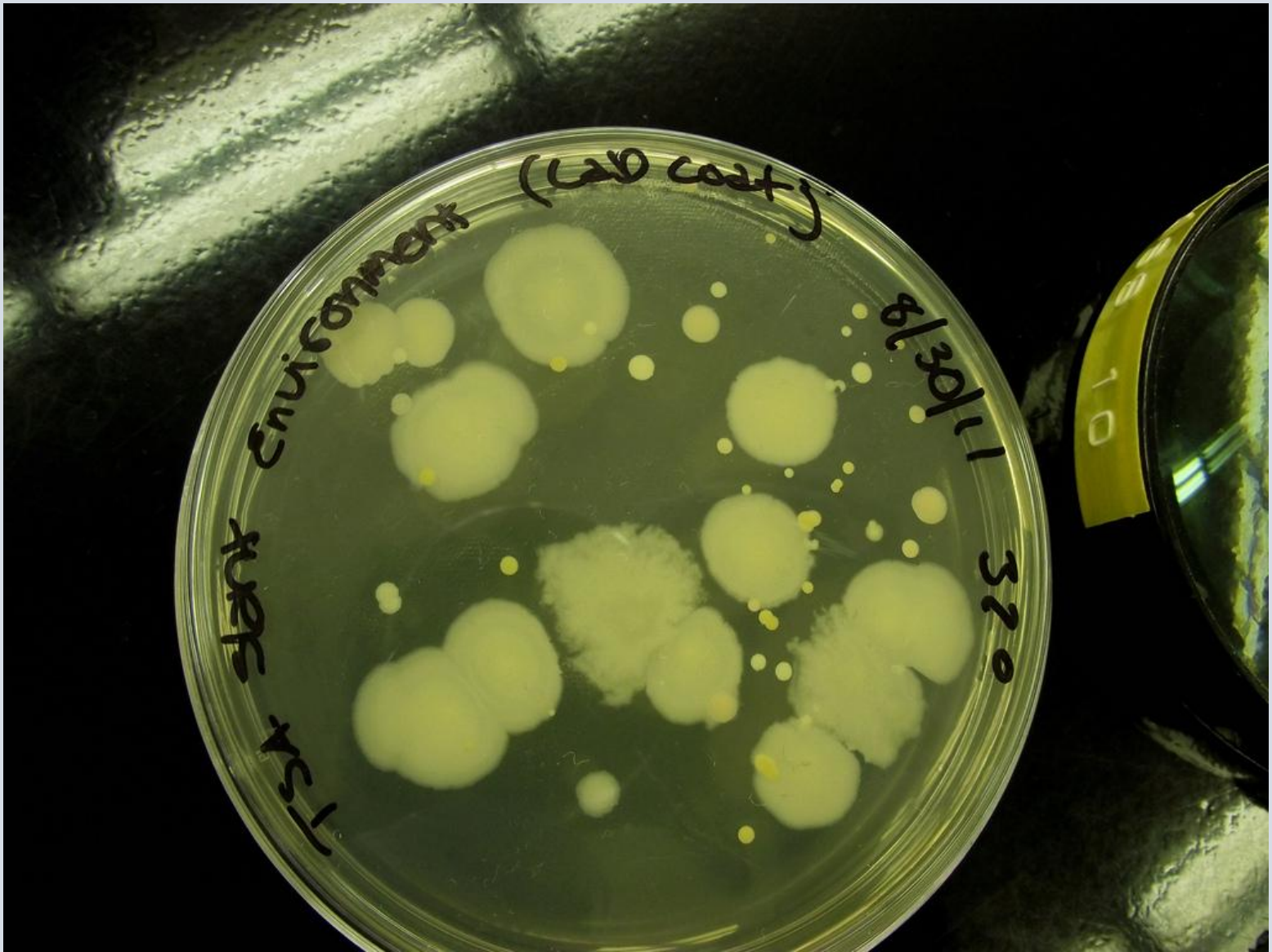
DevOps Defined

- DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support.
- **DevOps is also characterized by operations staff making use many of the same techniques as developers for their systems work.**

DevOps History In 60 Seconds

- ITIL, ITSM, ESM, etc. underdeliver in IT from 1989 on
- Agile comes to the developer world in 2001
- Lean comes to the developer world in 2003 (more slowly)
- O'Reilly Radar "Operations: The New Secret Sauce" in 2006
- Agile Infrastructure discussions start in Europe circa 2007
- Patrick Debois and Andrew Schafer meet up at Agile 2008
- O'Reilly Velocity Conference starts 2008
- Velocity 2009, seminal John Allspaw "10+ Deploys Per Day: Dev and Ops Cooperation" presentation
- Patrick Debois and Kris Buytaert put together first DevOpsDays in Ghent in 2009. Many more follow
- Lean influences enter DevOps via startup culture
- Large companies start branding DevOps "solutions"

Where Do I Start?



DevOps Concepts

DevOps Principles

DevOps Practices

DevOps Tools

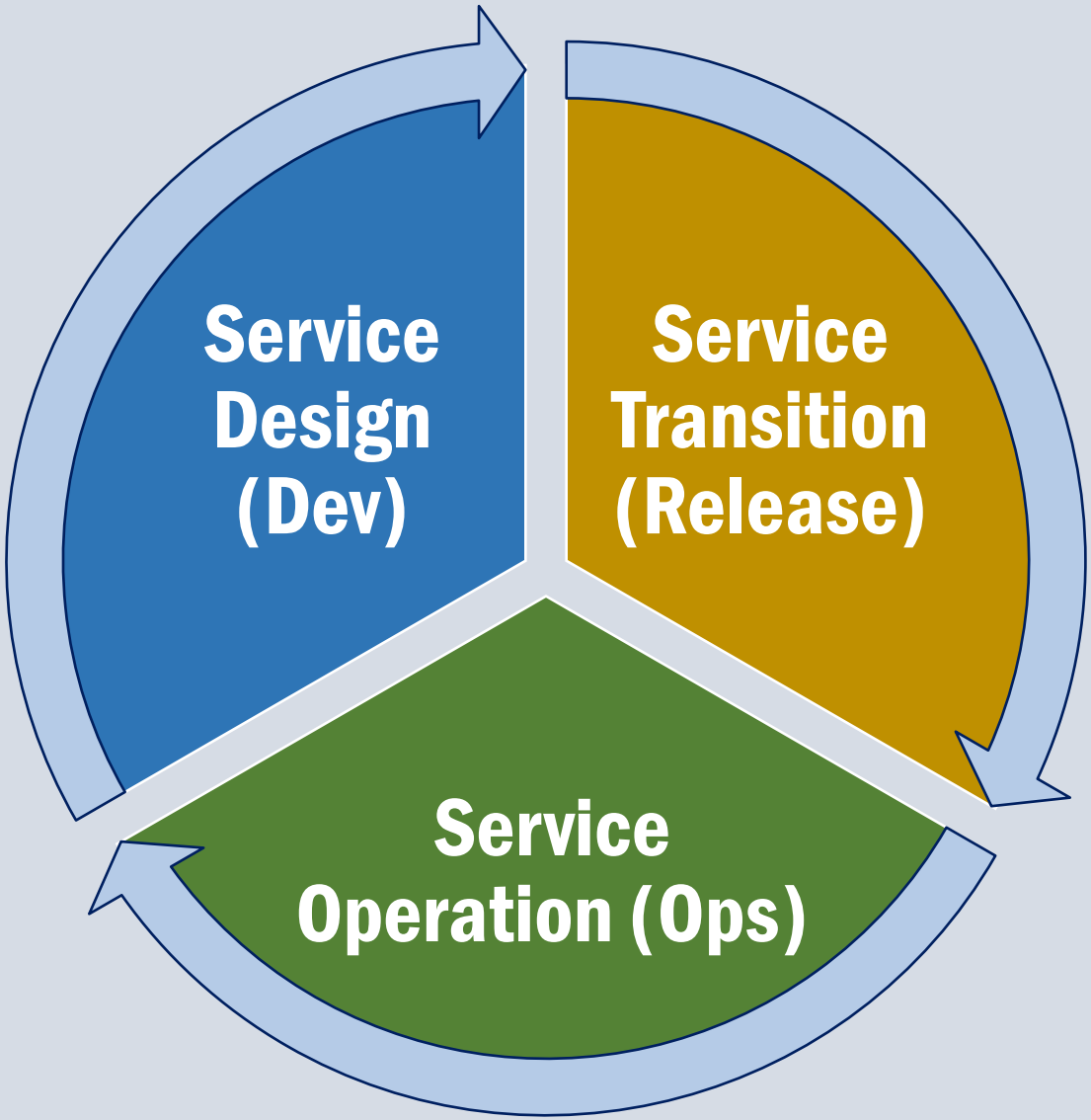
DevOps Principles

- The Three Ways
  - Systems Thinking
  - Amplify Feedback Loops
  - Culture of Continual Experimentation
- CAMS
  - Culture – People > Process > Tools
  - Automation – Infrastructure as Code
  - Measurement – Measure Everything
  - Sharing – Collaboration/Feedback
- Informed by the values in the Agile Manifesto and Lean Theory of Constraints

DevOps Practices

- Version Control For All
- Automated Testing
- Proactive Monitoring and Metrics
- Kanban/Scrum
- Visible Ops/Change Management
- Configuration Management
- Incident Command System
- Continuous Integration/Deployment/Delivery
- “Put Developers On Call”
- Virtualization/Cloud/Containers
- Toolchain Approach
- Transparent Uptime/Incident Retrospectives

An Implementation Model



Add Ops Into Dev

- Enhance Service Design With Operational Knowledge
  - Reliability
  - Performance
  - Security
  - Test Them
- Build Feedback Paths Back from Production
  - Monitoring and metrics
  - Postmortems
- Foster a Culture of Responsibility
  - Whether your code passes test, gets deployed, and stays up for users is your responsibility – not someone else's
- Make Development Better With Ops
  - Productionlike environments
  - Power tooling

Accelerate Flow To Production

- Reduce batch size
- Automated environments mean identical dev/test/prod environments
- Create safety through automation
  - Continuous Integration/Testing
  - Automated Regression Testing
  - Continuous Delivery
  - Continuous Deployment
  - Feature Flags (A/B testing)
  - Security Testing

Add Dev Into Ops

- Don’t do tasks for people. Build tools so they can do their own work.
- Monitoring/logging/metrics feeds back into dev (and the business)
- Blameless Incident Postmortems
- Developers Do Production Support/Empower Ops Acceptance

Grass Roots Checklist

- Find ways to **collaborate** – involve others early
- Find ways to **automate** and make self-service
- Become **metrics** driven
- Learn new things, **continually improve**
- Understand the larger **business goals**, metrics, and priorities you support
- **Communicate**
- Work in parallel with **small batches**
- Allow **refactoring**
- **Prove the business value to management**

Management Checklist

- **Experiment** – choose a test case as a pilot
- Then document and spread **best practices**
- Empower your teams, but guide their **values**
- **Metrics** are your friend – demand measurable outcomes
- **Don't accept excuses** when the old baseline isn't good enough
- Fail fast, **continually improve**
- **Build** on small successes to gain broad support for more substantive change.
- **Align** roles and responsibilities across groups – enable collaboration even if it seems "inefficient"

Things Not To Do

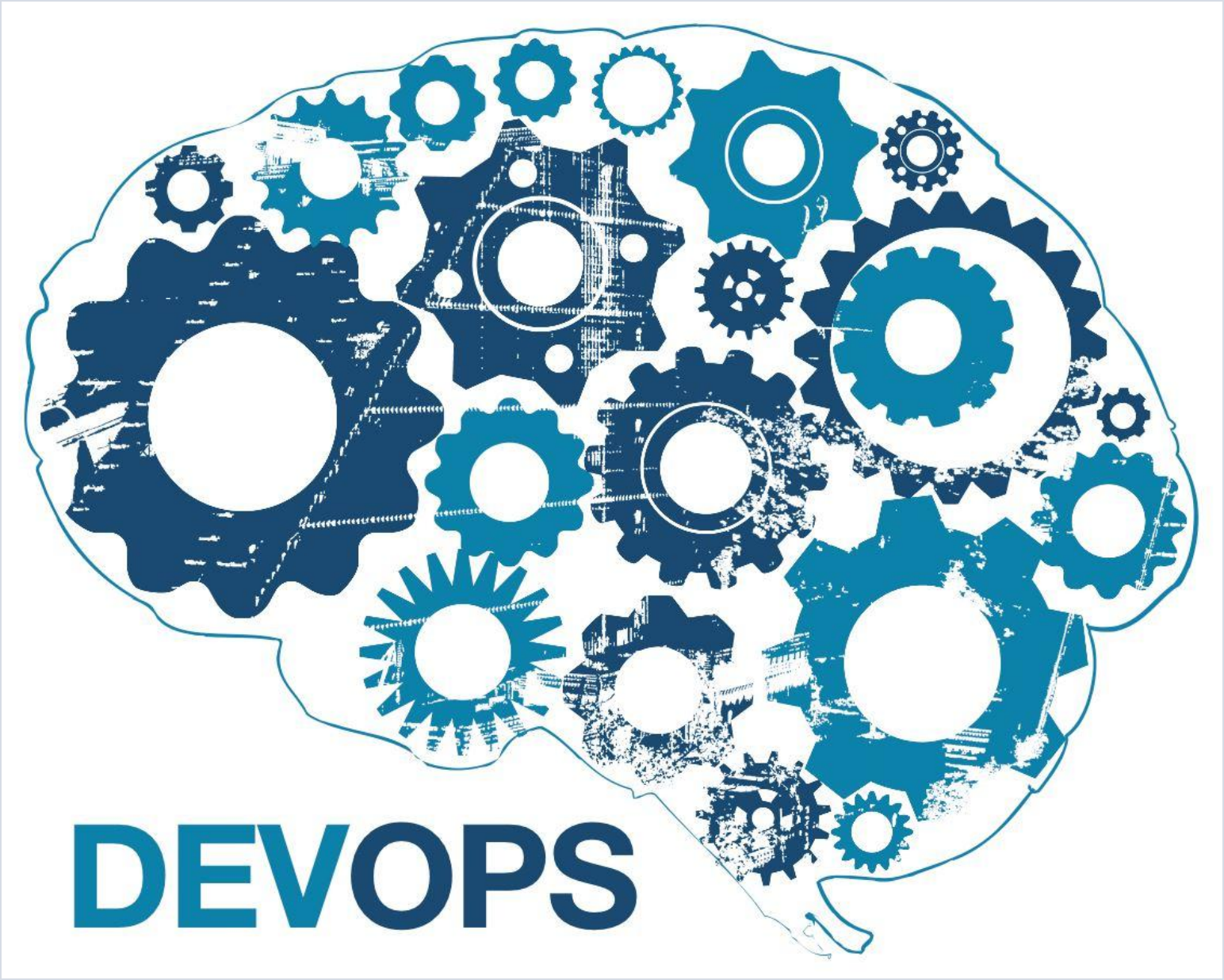
- Only Token Gestures
  - “Ops team, change your name to DevOps team!”
  - “Put DevOps in those job titles!”
- Only Implement Tools
  - Changing tools without changing tactics leaves the battlefield strewn with bodies
- Create More Silos
- Devalue Operations Or Development Knowledge
- Anything You’re Not Measuring The Impact Of

Does It Really Help?

- 2014 State of DevOps Report (9200 surveyed) measured correlation between high performing organizations and DevOps practice adoption
  - Lead time to changes down
  - MTTR up
  - No alteration in change fail rate

Core DevOps Research List

- Gene Kim's Visible Ops
- Tom Limoncelli's The Practice Of Cloud System Administration
- Gene Kim's The Phoenix Project (modeled on Goldratt's The Goal)
- Jez Humble's Continuous Delivery
- Michael Nygard's Release It!
- Gene Kim's The DevOps Cookbook (coming soon-ish)
- Various Mary and Tom Poppendieck Lean Software Development Books
- Velocity Conference (velocityconf.com)
- DevOpsDays Unconferences – There's one near you! (devopsdays.org)
- DevOps Weekly newsletter (devopsweekly.com)
- DevOps Café Podcast (devopscafe.com)
- The Twelve Factor App (12factor.net)
- The Agile Admin (theagileadmin.com)


DEVOPS


devops
automation
continuous
management
docker
cloud
delivery
deployment
chef
containers
dev
monitoring
tools
testing
development
integration
ops
configuration
enterprise
puppet
systems
everything
windows
scaling
deployments
pipeline
orchestration
sessions
aws
automated
practices
security
production
data
operations
culture
workflows
governance
infrastructure
app
code
metrics
process
analytics
saltstack
ansible
tooling
building
like
want
common
kubernetes
people
scalability
collaborative
organization
related
release
anything
qa
openstack
iaas
etc
source
python
technical
alerts
technology
mesos
success
elasticsearch
software
container
big
based
compliance
collaboration
real-time
level
world
overwhelmed
learning
best
moving
orchestrating
always
continous
provisioning
posted
pretty
ci
ci/cd
logging
service
yet
projects
new
environments
jenkins
real
network
improve
discovery
traditional
talks
things
architecture
general
open
change
topics
functional
time
lean
monitoring
get
flow
artifacts
interested
processes
business
quality
linux
experiences
concerns
list
end
using