Traditional Software Development Model
Traditional Software Development Model

"we should quickly put it in production"

"Do not confuse speed and haste. Production is serious!"
WORKED FINE IN DEV

OPS PROBLEM NOW

https://medium.com/point-nine-news/do-you-speak-devtools-73237550bf4f
What’s DevOps?

“A software engineering practice that aims at unifying software development (Dev) and software operation (Ops).”

DevOps has been initially coined by Patrick Debois in 2009
What’s DevOps?
Motivations

• Reduce the release cycle (time to market, lead time between fixes...)

• More fragmented approach (small increments vs. bigbang)

• Seamless updates

• Shared responsibilities (all in the same boat)

• Continuous improvement
Typical Stories

• Story 0: Dev and Ops collaborate to develop environment definitions
  — Value: Ensures that Dev understands and deals with production-like environments; avoids architectural miscommunications

• Story 1: Dev continuously delivers application changes to a realistic environment for testing
  — Value: Shared technology ensures testable environments and script reuse for repeatable delivery; Test org always has known good builds, properly deployed

• Story 2: Release Applications from Test/Staging to production
  — Value: Shared technology and automation ensures no gratuitous differences between dev/test and prod

• Story 3: Collaborative incident management
  — Value: ensures an integrated process for reproducing and resolving defects and issues between dev, test, and ops

• Story 4: Dev and Ops use the same analysis and instrumentation in dev, test, and ops
  — Value: Ensures a common understanding of quality and performance (and no fingerpointing)

• Story 5: Manage the entire delivery pipeline with end-to-end visibility and dashboards
  — Value: Enables end-to-end delivery metrics and visibility into bottlenecks

cf. wikipedia
Responsabilities

Pre-DevOps

• Developpers produce the source code
  – *Do not care about the impact on the overall system in production*

• **IT teams operate the system and ensure the quality of service**
  – *Do not care about the performance of the code*

Post-DevOps

• **Shared responsibilities with all stakeholders in the same boat**

• *“You build it, You run it.”* - Walter Vogels, Amazon CTO
Expected Benefits

• Faster time-to-market/delivery times that improves ROI

• Engaged, empowered cross-discipline teams

• Stable/reliable operating environments

• Early detection and faster correction of defects

• Improved quality
DevOps: 3 Basic Principles

- System thinking
- Amplify feedback loops
- Culture of continual experiment and learning

https://itrevolution.com/the-three-ways-principles-underpinning-devops by Gene Kim
System thinking

Create a smooth and fast flow from dev to ops

• Make your work visible (*visual board, lead time*)
• Limit Work In Progress ("Stop starting. Start finishing.")
• Reduce batch size and intervals of work
• Continually identify and elevate the system’s constraints
• Eliminate hardships and waste in the value stream

Enable a fast and constant feedback from ops to dev

- Working safely within complex systems
- See problems as they occur
- Swarm and solve problems to build new knowledge
- Keep pushing quality closer to the source
- Enable optimizing for downstream work centers

https://itrevolution.com/the-three-ways-principles-underpinning-devops by Gene Kim
Continual experiment and learning

Enable constant creation of individual knowledge, which is then turned into team and organizational knowledge

• Enabling organizational learning and a safety culture
• Institutionalize the improvement of daily work
• Transform local discoveries into global improvements
• Inject resilience patterns into our daily work
• Leaders reinforce a learning culture

https://itrevolution.com/the-three-ways-principles-underpinning-devops by Gene Kim
DevOps vs. Agile

• DevOps is especially complementary to the Agile software development process.
  – extends and completes the continuous integration and release process

• DevOps enables a far more continuous flow of work into IT Operations
  – Avoid situation where development delivers code every two weeks but it's deployed only every two months
Some DevOps Principles

- Observability
- Stateless architecture
- Reproducibility and replicability
- Accountability
- Software lifecycle automation
The CALMS Conceptual Framework

• **Culture**: There is nothing fluffy about culture.

• **Automation**: Automation is the idea that you should program everything.

• **Lean**: Running lean means keeping everything to a minimum.

• **Measurement**: If a team does not have visibility into everything, something will eventually go horribly wrong.

• **Sharing**: Sharing is not just reporting facts, it is regular exchanging of ideas across teams.

✓ Often used as a maturity model

✓ Proposed by Jez Humble
Common Attributes of Successful Cultures

• **Infrastructure As Code**
  – Full Stack Automation
  – Commodity Hardware and/or Cloud infra
  – Reliability in software stack
  – Datacenter or Cloud Infrastructure APIs
  – Core Infra Services

• **Application As Services**
  – Service Orientation
  – Lightweight Protocols
  – Versioned APIs
  – Software Resiliency (Design for Failure)
  – Database/Storage Abstraction

• **Dev/Ops/All As Teams**
  – Shared Metrics/Monitoring
  – Incident Management
  – Service Owners On-call
  – Tight integration
  – Continuous Integration
  – Continuous Deployment
  – GameDay
Some DevOps Practices

• Static analysis and test automation
• Continuous integration/deployment/delivery
• Release/configuration management and Infrastructure as Code (IaC)
• Monitoring (e.g., performance, availability…)

• Change management, hypothesis driven dev.
• Resilience engineering
Purpose of the Configuration, Release and Deployment Pipeline

- **Visibility**: All aspects of the delivery system are visible to all team members promoting collaboration.

- **Feedback**: Team members learn of problems as soon as they occur so that issues are fixed as soon as possible.

- **Continually Deploy**: Through a fully automated process, you can deploy and release any version of the software to any environment.
Performance Metrics

![Performance Metrics Diagram](https://services.google.com/fh/files/misc/state-of-devops-2019.pdf)
DevOps Tools
DevOps Tools
The adoption of DevOps is being driven by factors

- Use of **agile** and other development processes and methodologies

- Demand for an **increased rate of production releases** from application and business unit stakeholders

- Wide availability of **virtualized and cloud infrastructure** from internal and external providers

- Increased usage of data center **automation and configuration management tools**

- Increased focus on **test automation and continuous integration methods**;
Key indicators of performers

Comparing highest to lowest performers.

- 208 TIMES MORE frequent code deployments
- 106 TIMES FASTER lead time from commit to deploy
- 2,604 TIMES FASTER time to recover from incidents
- 7 TIMES LOWER change failure rate (changes are 1/7 as likely to fail)

Key indicators of performers

Comparing highest to lowest performers.

Antipatterns

- Management just saying we’re doing DevOps
- Just changing job titles to DevOps
- Just merging dev and ops teams or creating a separate DevOps team
- Committing is done
- My responsibility ends here
- Devs blaming Ops; Ops blaming Devs
- Ops not involved early
- DevOps means Developers Managing Production
- It’s not just automation or a tool (or set of tools)
Further Material

• Books

• Conferences
  – DevOpsCon
  – DevOpsDays
  – DevSecDays
  – KubeCon + CloudNativeCon

• A lot of high-quality posts (e.g., medium)

• Teaching initiative: https://teachdevops.github.io