DevOps Handbook

Part I

* CH 1 - Agile, continuous delivery, and the three ways
* CH 2 - The first way: the principles of flow
* CH 3 - The second way: the principles of feedback
* CH 4 - The third way: the principles of continual learning and experimentation

Part II

* CH 5 - Selecting which value stream to start with pg. 51
  + Greenfield vs. brownfield services pg. 54
  + Consider both **systems of record** (ERP, HR, Finance) and **systems of engagement** (employee or customer facing) pg. 56
  + Start with the most sympathetic and innovative groups
    - Technology adoption curve Fig 9. Pg. 58
  + Expanding DevOps across our organization pg. 58
    - PH 1. Find innovators and early adopters
    - PH 2. Build critical mass and silent majority
    - PH 3. Identify the holdouts
  + “Little fish learn to be big fish in little ponds.” - Peter Drucker
* CH 6 - Understanding the work in our value stream, making it visible, and expanding it across the organization pg. 61
  + Identify the teams supporting our value stream pg. 63
  + Create a value stream map to see the work pg. 63
  + Creating a dedicating transformation team pg. 66
  + Agree on a shared goal pg. 68
  + Keep our improvement planning horizons short pg.68
  + Reserve 20% of cycles for **non-functional requirements** and reducing technical debt pg. 69
    - Positive user-invisible value Figure 11 pg. 70
  + Increase the visibility of work pg. 73
  + Use tools to reinforce desired behavior pg. 73
* CH 7 - How to design our organization and architecture with Conway’s Law in mind pg. 77
  + ORMs pg. 79
  + Organizational archetypes pg. 80
    - Functional, Matrix, Market
  + Problems often caused by overly functional orientation (“optimized for cost”) pg. 81
    - PSC, IT, HR, shared services
  + Enable market-oriented teams (“optimized for speed”) pg. 82
  + Making functional orientation work pg. 83
  + Testing, operations, and security as everyone’s job, every day pg. 84
  + Enable every team member to be a generalist pg. 85
    - Specialists vs. generalists vs. “E-shaped” Table 2 pg. 86
      * QE transition last yr
    - Dr. Carol Dweck fixed vs. growth mindset pg.87
  + Fund not projects, but services and products pg. 87
  + Design team boundaries in accordance with Conway’s Law pg. 88
  + Create loosely-coupled architectures to enable developer productivity and safety pg. 89
  + Keep team sizes small (the “two-pizza team” rule) pg. 90
* CH 8 - How to get great outcomes by integrating operations into the daily work of development pg. 95
  + Create shared services to increase developer productivity pg. 97
  + Embed ops engineers into our service teams pg. 99
  + Assign an ops engineer to each service team pg. 100
  + Integrate ops into dev rituls pg. 101
    - Invite ops to our dev standups pg. 102
    - Invite ops to our dev retrospectives pg. 102
    - Make relevant ops work visible on shared kanban boards pg. 104

Part III

* CH 9 - Create the Foundations of Our Deployment Pipeline
  + Enable on-demand creation of dev, test, and production environments pg. 113
    - Use automation for any or all:
      * Copying a virtualized environment (eg. VMware image, etc.)
      * Building an automated environment creation process that starts from “bare metal” (eg, PXE install from a baseline image)
      * Using “infrastructure as code” config management tools (eg Puppet, Chef, Ansible, Salt, CFEngine, etc.)
      * Using automated automated operating system configuration tools (eg Solaris Jumpstart, Red Hat Kickstart, Debian preseed)
      * Assembling an environment from a set of virtual images or containers (eg. Vagrant, Docker)
      * AWS, Azure, other public cloud
  + Create our single repository of truth for the entire system pg. 115
    - All application code and dependencies
    - Any script used to create db schemas, application reference data
    - All env create tools
    - Any file used to create containers
    - All supporting automated tests and any manual test scripts
    - Any script that supports code packaging
    - Cloud config files
    - ***\*\*\*Use of version control by operations is the highest predictor of both IT performance and organizational performance\*\*\**** pg 117
  + ***Make infrastructure easier to rebuild than to repair*** pg. 118
    - Pets vs. Cattle pg 118 - Bill Baker (Microsoft) quote
    - Make changes to one config, then automatically deploy everywhere via Puppet or Ansible, etc.
    - *Immutable infrastructure*
  + Modify our definition of “done” to include running in production-like environments
* CH 10 - Enable fast and reliable automated testing
  + *Imposter syndrome pg. 124*
  + *Fix-it days - improvement blitzes pg 125*
  + *Containers pg. 128*
  + *Test pyramid pg. 133*
  + *Test Driven Development pg. 134*
* CH 11 - Enable and practice continuous integration
  + *Version control branches pg. 143*
  + *Trunk-based development pg. 145*
    - *Optimize for indiv productivity*
    - *Optimize for team productivity*
  + *BLUF don’t use feature branching pg. 147 AKA small code batch sizes*
* CH 12 - Automate and enable low-risk releases
  + *Fully document the steps in the deployment process with an eye towards automating as many as possible. Pg. 155*
  + *Mean Time To Repair (MTTR) pg. 158*
  + *Decouple deployments from releases pg. 164*
  + *Environment-based release patterns pg. 165*
    - *Blue-green deployment pattern pg. 166*
    - *Database changes pg. 167*
  + *Application-based release patterns pg. 165*
    - *Dark launches*
* CH 13 - Architect for low-risk releases
  + *Architect for low-risk releases*
    - *Strangler application pattern pg. 180*
    - *Martin Fowler*

Part IV

* CH 14 - Create telemetry to enable seeing and solving problems
  + *Create telemetry to enable seeing and solving problems*
  + *Create within our applications and environments*
    - *Ganglia, Graphite*
  + *Mean Time To Repair (MTTR) in minutes not days*
    - *Graph on pg. 197*
  + *Art of Monitoring by James Turnball*
  + *Example tools footnote pg. 199*
  + *Get logs, transform them into metrics using the event router*
  + *“Monitoring is so important that our monitoring systems need to be more available and scalable than the systems being monitored.” - Adrian Cockcroft*
  + *logging of critical features*
  + *INFO – wrong login for FPY*
  + *WARN – DB not writing or taking too long to connect*
  + *WARN – Internet connectivity*
  + *inability to create institutional knowledge*
  + *80% of all outages are caused by change and 80% of MTTR is spent trying to determine what changed*
  + *questions on pg. 204*
  + *Enable creation of Production metrics as part of daily work*
  + *StatsD from Etsy*
  + *Create self-service access to telemetry and* ***information radiators***
  + ***IDEA*** *– create a 320F new product launch dashboard*
  + *When metrics aren’t actionable, they are likely vanity metrics that provide little useful information*
  + *infrastructure metrics*
  + *cost of delayed features*
  + *instead of measuring downtime, measure real business consequences of downtime. How much revenue should we have attained but didn’t.*
* CH 15 - Analyze telemetry to better anticipate problems and achieve goals
  + *Analyze telemetry to better anticipate problems and achieve goals*
  + *outlier detection*
    - *Normally distributed data*
      * *compute mean and std for normal*
      * *what to alert on*
        + *analyze severe outages in last 30 days*
        + *create a list of telemetry that could have enabled earlier and faster detection and diagnosis*
  + *anomaly detection*
    - *chi-squared distribution*
    - *smoothing - moving averages (rolling avgs)*
    - *Fast Fourier Transform test*
    - *Kolmogorov Smirnov and other "non-parametric"*
    - *R*
    - *Oculus*
    - *Opsweekly*
    - *Skyline*
  + *Application logs*
    - *INFO - wrong FPY tool login*
    - *WARN - Db writes not working or taking too long*
    - *WARN - internet connectivity or uptime*
* CH 16 - Enable feedback so development and operations can safely deploy code
  + *Pg 228 Progression*
  + *Optimize for MTTR instead of MTBF*
  + *Fix forward or roll back*
  + *Contextual inquiry pg 232*
    - *UX observation*
  + *Service handback mechanism*
    - *Fig. 38 Pg 237*
  + *Launch Readiness Review & Handoff Readiness Review*
* CH 17 - Integrate hypothesis-driven development and A/B testing into our daily work
  + Balsamiq
  + A/B testing
* CH 18 - Create review and coordination processes to increase quality of our current work
  + Peer review
  + Pull request
  + Gitflow pg 250
  + “The Knight Capital” failure pg 251
  + Counterfactual
  + Code review
  + “Ask a programmer to review ten lines of code, he’ll find ten issues. Ask him to do five hundred lines, and he’ll say it looks good.” pg 256
  + Pair programming pg 256
    - Pair pattern: driver/ observer
    - Pair pattern: one write tests/ other implement
  + Extreme programming pg 259

Part V

* CH 19 - Enable and inject learning into daily work
  + *Dr. Steven Spears quote pg. 271*
  + *NETFLIX “Cloud Native” architecture pg. 272*
    - *Chaos Monkey*
  + *Establish a just, learning culture*
  + *Dr. Sidney Dekker*
    - *Bad Apple Theory*
    - *“Human error is not the cause of troubles; instead, human error is a consequence of the design of the tools that we gave them”*
    - *Instead of “naming, blaming, and shaming or goal should always be to maximize opportunities for organizational learning.*
  + *Blameless postmortems pg. 274-275*
  + *Controlled introduction of failures into production pg. 274*
  + *John Allspaw quote pg. 274*
  + *System as imagined vs system that actually exists*
  + *Publish post mortems as widely as possible*
    - *Search: “Chef” “postmortem”*
* CH 20 - Convert local discoveries into global improvements
  + *Chatbots and chat rooms pg. 287*
  + *Automate standardized processes in software for re-use pg. 289*
  + *Create a single, shared source code repository for our entire organization pg. 290*
  + *Spread knowledge by using automated tests as documentation and communities of practice pg. 293*
  + *Design for operations through codified nonfunctional requirements pg. 293*
  + *Build reusable operations user stories into development pg. 294*
    - *Ops checklist pg. 295*
  + *Ensure technology choices help achieve organizational goals pg. 295*
* CH 21 - Reserve time to create organizational learning and improvement
  + Institutionalize rituals to pay down technical debt pg. 300
  + Enable everyone to teach and learn pg. 303
  + Create internal consulting and coaches to spread practices pg. 306

Part VI

* CH 22 - Information security as everyone’s job, every day
  + *James Wickett creator of Gauntlet security tool*
  + *Integrate security into development iteration demonstrations*
  + *Possible metrics: development velocity, failed customer interactions*
  + *Integrate security into defect tracking and post-mortems*
  + *Integrate preventive security controls into shared source code repositories and shared services*
  + *Integrate security into our deployment pipeline*
  + *Ensure security of the application*
* CH 23 - Protecting the deployment pipeline
  + *Integrate security and compliance into change approval processes*
  + *Re-categorize the majority of our lower risk changes as standard changes*
  + *What to do when changes are categorized as normal changes*
  + *Reduce reliance on separation of duty*
  + *Ensure documentation and proof for auditors and compliance officers*