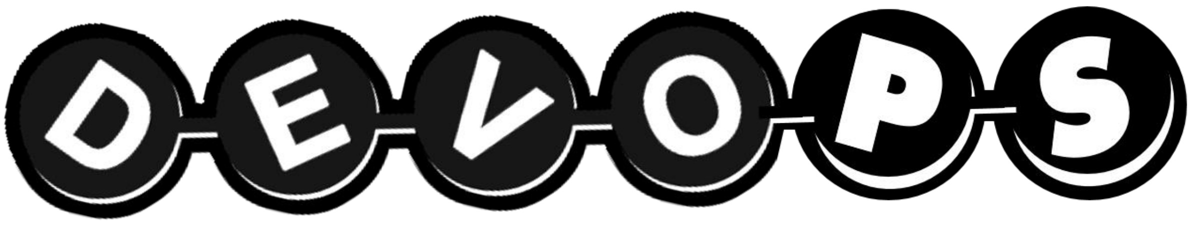
Everything You Wanted To Know About DevOps But Were Afraid To Ask
with Claire Moss @aclairefication

Duty Now for the Future: Recognizing DevOps

DevOps Buzzword Bingo



Development	Effective	Values	Operations	People	Self-Service
Delivery	Empower	Velocity	Organization	Process	System
Deployment	Escalate	Version Control	Objectives	Practice	Server
Dashboard	Enterprise	Virtualization	Optimize	Partnership	Software
Dark Launch	Evolution	Visualize	Orchestration	Pager	Support
Discipline	End User	Validate	Open Source	Push	Sanity

Bonus words!

Decisions
Dedicated
Docker
Emergent
Voice (of the customer)
Value Stream Mapping
(service) Oriented
Office
Principles
Production
Partnership
Pager
People
Process

Practice
Post-mortems
Push (build, code)
Protocol
Pipeline
Postman
Protractor
Pivotal Tracker
PaaS (Platform)
Provisioning
Ship
System
Support
Service

Security
Self-Service
Sanity
Scalable
Sandbox
SaaS
Server
Software
Splunk
Subversion
Slack
Selenium
ServiceNow
SonarQube

aclairefication

Everything You Wanted To Know About DevOps But Were Afraid To Ask
with Claire Moss @aclairefication

Duty Now for the Future: Recognizing DevOps

Fill in the blank Q&A

What is DevOps?

How do I know whether I'm already doing DevOps?

Why DevOps? What problem is DevOps solving?

Who is part of DevOps?

What are barriers to DevOps?

What are the benefits of DevOps?

How do we decide how to DevOps?

What makes DevOps possible? What choices support DevOps?

What interpersonal behaviors help with DevOps?

What release cadence do we have?

Other things *I* want to know about DevOps:

aclairefication

Everything You Wanted To Know About DevOps But Were Afraid To Ask
with Claire Moss @aclairefication

Duty Now for the Future: Recognizing DevOps

Name: Danielle Development	Quote:
Job/Role: “developers” or “engineers”	Motivations/Goals: ship competitive features
List of traits: (tagging style)	Biggest problem? long learning cycle causing rework
Likes about job: learning innovative technology	Dislikes about job: bug fixing

Job titles/roles in this space: software developer (a.k.a. back-end), client software developer (a.k.a. front-end), software architect, middleware developer, data architect, enterprise architect, software tester/QA, user experience (UX), graphic designer, product owner, product manager

Activities: test, code, & integrate applications/services; test automation/scripting, test data management, bug fixing, reducing friction/process automation

Metrics: respond to competitive landscape changes, time to market/deliver user features, time to start a project, lead time/throughput/cycle time/frequency of releases, quality, detect problems quickly, technical debt, independently develop & validate code, safe/routine/predictable/low stress deployments, reversible deployments, maximize developer productivity, cost

Barriers: tightly coupled architecture (i.e. which applications depend on which services), “cutting corners” (e.g. workarounds), coordination/handoffs (e.g. waiting for approvals), complexity, fragility, risk tolerance, poor documentation

Impacts: rework, one-off processes, deployment pain

Enablers: agile, retrospectives, small empowered teams (e.g. choosing tools), ownership of work, skills training/internal sharing/lifelong learning, peer reviews, demos, version control, design thinking, pipelines, architecting for low-risk releases, Continuous Integration (CI), Continuous Delivery (CD), Production-like environments, Lean, kanban, limit Work In Progress (WIP)/work in small batches, making flow of work visible (e.g. value stream mapping)

Newer activities: deploy, & run services in Production; “zero bugs” (i.e. fix problems as they are found), create/update specification during development, multidisciplinary/“full stack”, deploy any time of day, Continuous Deployment (CD), dark launch, feature toggles, alpha/beta releases for customer feedback, A/B testing, trunk-based development, hypothesis-driven development, understanding operational and environment requirements, usage reports and notifications (e.g. performance changes, crashes) for deployed instances, feedback from Operations, self-serve on-demand resources, repeatable processes, lightweight change approvals

aclairefication

Everything You Wanted To Know About DevOps But Were Afraid To Ask
with Claire Moss @aclairefication

Duty Now for the Future: Recognizing DevOps

Name: Owen Operations	Quote: “the platform is the product”
Job/Role: “operators” or “maintainers”	Motivations/Goals: prevent outages, chaos, & disruption
Feelings: burned out /exhausted, indifferent/ cynical	Biggest problem? Manual processes that are hard to repeat
Likes about job: solving real-world problems	Dislikes about job: being on call/work disrupting life

Job titles/roles in this space: Platform Engineer, Change Control, Site Reliability Engineer (SRE), Server Build Engineer, Network Engineer, System administrator/Sysadmin, Support, Database Administrator (DBA), Data Architect, Service Governance, see ITIL and ITSM for more

Activities: deploy Production changes, capacity planning, controlling access to the servers, hardware and infrastructure configuration, configuration management, monitoring, release management, process automation, incident detection & recovery, problem management, data center migration; service strategy, design, & support

Metrics: Service-Level Agreements (SLAs), Mean Time To Restore (MTTR), change fail percentage, stability, reliability, availability, shorter lead times/frequent deployments; safe, secure, & quick platforms

Barriers: viewed as tactical, manual work from ticketing system, cutting corners (e.g. workarounds, taking on technical debt), lack of trust/“wall of confusion” between silos

Impacts: “firefighting” (i.e. unplanned work), burnout, speed to market, scaling, compliance, budgeting

Enablers: DevOps cultural norms, blameless postmortems (i.e. safe to fail), small empowered teams (e.g. choosing tools), infrastructure-as-code, source control, Continuous Integration/ Continuous Delivery patterns, architecture aligned with value stream, Lean, kanban, agile, limit Work In Progress (WIP)/work in small batches, making flow of work visible (e.g. value stream mapping)

Newer activities: deployment automation, infrastructure-as-code, making infrastructure more self-service, telemetry, proactive notifications (e.g. threshold, rate-of-change), chaos engineering (e.g. Production fault and network latency injection)

aclairefication