DevOps assessment for executives

<http://devopschecklist.com/>

This checklist is comprised of 48 items you can use to gauge the maturity of your software delivery competency, and form a baseline to measure your future improvements. It's not meant to say "you're failing DevOps" or deprive you of a badge **(make yourself a badge just for reading this!)**, but surface areas of potential improvement.

The content here is based on many case studies, talks and content on DevOps, collected over the past 5 years - you can see some of the highlights down in the ['References' section](http://devopschecklist.com/#references). We're also continually adding more context and content - so tweet [@SteveElsewhere](http://twitter.com/steveElsewhere) or [email us](mailto:checklist@myplanet.com?subject=DevOps%20Checklist%20Feedback) if you have suggestions, disagree, have a question or want more info!

**Alignment**

**Unifying group and individual direction and goals around the singular vision of the organization.**

* We prioritize according to business objectives. DevOps is all about alignment. It was born out of frustration and trouble between the disparate priorities and incentives of Development and Operations teams. Alignment comes from unifying towards a common goal.Read more
  + Culture
* We volunteer for tasks rather than having them assigned. Autonomy is a critical facet to motivation and job satisfaction. By allowing team members to self-select tasks, you'll see them work on what they're best suited to complete.Read more
  + Culture
* Our team has clear objectives that correspond with our company vision.
  + Culture
  + Measurement
* Our product team is focused on sustainable velocity rather than current speed.
  + Culture
  + Automation
  + Measurement
* We focus on time to repair rather than time between issues. A great deal of software delivery theory actually stems from manufacturing. The major difference is that testing failure is faster, cheaper and non-destructive. Thus, we focus on early failure detection through automated testing, as far upstream as possible to catch issues before they become costly. We also focus on learning from simulating production failure by staging ['Game Days'](https://stripe.com/blog/game-day-exercises-at-stripe), ['Failure Fridays'](http://www.pagerduty.com/blog/failure-friday-at-pagerduty/) and optimizing reaction time and corrective process.Read more
  + Culture
  + Measurement
* DevOps is not isolated to a specific role in our organization. Hiring a 'DevOps engineer' is an antipattern because it charges an individual with understanding and operating in two worlds with their respective opposing incentives, while remaining independent of either, as their own silo. Even more confusing is reconciling that position with a larger organizational DevOps pursuit. Hiring an Automation, Systems, Operations or Development engineer specifically, or just simply an engineer would be more humane and doesn't require you to write a manifesto for your job descriptions. For more info on hiring for DevOps, see [O'Reilly's free book.](http://www.oreilly.com/webops-perf/free/devops-hiring.csp)Read more
  + Culture
* DevOps is not isolated to a specific team in our organization. DevOps works best as a cross-functional team focus. Large organizations have often seen success with an exploratory task force team or a greenfield project team, but in order to get optimal results every team should be able to build and deliver without relying on another department. Vote no on silos!Read more
  + Culture
* Our operational functions are seen as a source of competitive advantage. A large part of why DevOps exists is in reaction to common perceptions of IT as a cost to an organization, yet we now know it can just as easily be a source of innovation.Read more
  + Culture

**Context**

**Making relevant information and contact available to those who need it, when they need it.**

* Representation from our operations team is involved in development sprint planning.
  + Culture
  + Sharing
* We make potential changes visible to all members of our product team.
  + Culture
  + Sharing
* We have an automated system for running tasks and receiving notifications with our team chat.
  + Culture
  + Sharing
* We consult with auditors and regulators regularly and seek guidance when designing systems.
  + Culture
  + Sharing
* Our team is encouraged to question tasks and priorities.
  + Culture
  + Sharing
* We have a centralized instant message system including all members of our product team.
  + Sharing
* All members of our product team have access to environment status, metrics and history.
  + Sharing
  + Measurement
* All members of our product team have access to code status, metrics and history.
  + Sharing
  + Measurement

**Learning**

**Empowering personal growth and fostering understanding through continuous improvement.**

* We cultivate an environment of continuous learning.
  + Culture
* We regularly celebrate our product team's learnings and success internally.
  + Culture
* We regularly share our product team's learnings and success with the rest of our organization.
  + Culture
  + Sharing
* We openly discuss failures in order to share learning.
  + Culture
  + Sharing
  + Measurement
* We identify skills needed to improve or address future objectives.
  + Culture
  + Measurement
* We strive to examine how we complete our work, and how effectively we complete it.
  + Culture
  + Measurement
* We estimate based on measurement and past experience.
  + Measurement

**Lifecycle**

**Focus on software as a product deserving of care, attention and reflection, within an ever-changing ecosystem.**

* Our software development cycle is 2 weeks or less.
  + Culture
* Our software development cycle is defined by releasing a working change into production. Iterative development and process optimization depends on repeatedly performing the critical functions of building and deploying software. The more it's done, the better it's understood, the easier it is to improve and automate.Read more
  + Culture
  + Measurement
* We stop development upon discovering a defect and prioritize its repair.
  + Culture
* Developers or product owners are able to deploy our product to production. The focus here is empowering the most informed to perform the task. Someone intimately familiar with a change is best positioned to evaluate its performance and correctness. Developers and POs should be empowered to measure and visualize the operation of the change as well, in order to give them a holistic view of the changed system.Read more
  + Culture
  + Automation
* We have automated testing prior to automated production deployment. Production deploys can still be manually initiated in cases where a signoff is required, yet the testing, deployment process and validation should be entirely automated to ensure consistency, accuracy and optimal speed. Automating deploys means no copy and paste, less security risk, more auditability, more confidence.Read more
  + Automation
  + Testing
* Our configuration of systems is automated. Friends don't let friends manually configure systems.Read more
  + Automation
* Our deployed system configuration is immutable. Immutability is an ideal state as it allows operations to focus on optimizing consistent and predictable systems.Read more
  + Automation
* Our release and deployment automation is environment agnostic. To avoid fragmentation or repetition, it's best to limit environmental differences to variable values only.Read more
  + Automation
  + Testing

**Organization**

**Providing structure for interaction and cohesion supporting collaboration and productivity.**

* Our subject matter expertise is not isolated to individuals. If you're lucky, you're surrounded by experts. Ideally, they're not just islands of knowledge and responsibility, but part of a peer group within the organization.Read more
  + Culture
  + Sharing
* We enable peer and cross-functional review for changes.
  + Culture
  + Sharing
  + Testing
* Our organization is configured around cross-functional teams. Cross-functional teams are more empowered to build and deploy their own software without being hindered by constraints. They also avoid an 'us vs them' effect of siloed functional teams, learn from, and foster awareness and empathy for eachothers priorities and interests.Read more
  + Culture
* Our teams are customer and product oriented. Customer focus helps align teams towards valuable goals, and keeps development and engineering zeroed in on delivery and processing feedback.Read more
  + Culture
* We review priorities on a regular basis.
  + Culture
  + Sharing
* Our developers have access to production-like systems to work and test on.
  + Automation
  + Testing
* Our developers have access to production-like data to work and test against.
  + Automation
  + Testing
* Our developers have access to dependencies required to build and test software.
  + Automation
  + Testing

**Process**

**Rituals crafted to foster consistency and confidence, providing a framework for continuous improvement.**

* Our organization follows agile development practices. To realize the full power of a focus on DevOps, feedback and regular iteration are critical. [DevOps is built on agile.](http://www.agileweboperations.com/lean-agile-devops-related)Read more
  + Culture
* We practice blameless postmortems. Focus on learning and continual improvement. This is the human equivalent of focusing on MTTR vs MTBF. Read: [John Allspaw's great article on Etsy's process](https://codeascraft.com/2012/05/22/blameless-postmortems/)Read more
  + Culture
  + Sharing
* We regularly examine constraints in our delivery process.
  + Culture
* Our system configuration is committed into version control.
  + Automation
  + Testing
* Our documentation is version controlled and shared.
  + Sharing
* We maintain a backlog of tasks, visible to all team members and available for comment.
* We practice test or behaviour driven development.
  + Testing
* We test our changes against a merge with our mainline code. Until you prove your code isn't a destructive regression, it shouldn't be deployed for someone else to discover. Automated testing and frequent merging should catch most of these issues, and make sure you don't find out about problems from someone else - or worse, Twitter.Read more
  + Testing

[Tweet this](https://twitter.com/share?text=We+test+our+changes+against+a+merge+with+our+mainline+code.&url=http%3A%2F%2Fdevopschecklist.com%2F&hashtags=devops&via=myplanetHQ)

* We test our changes against production-equivalent load and use patterns.
  + Testing