DZone

THE GUIDE TO

CONTINUOUS DELIVERY

RESEARCH PARTNER SPOTLIGHT

SELECTIONS FROM **THE GUIDE TO CONTINUOUS DELIVERY, VOL. II**

PRESENTED BY

SAUCE LABS

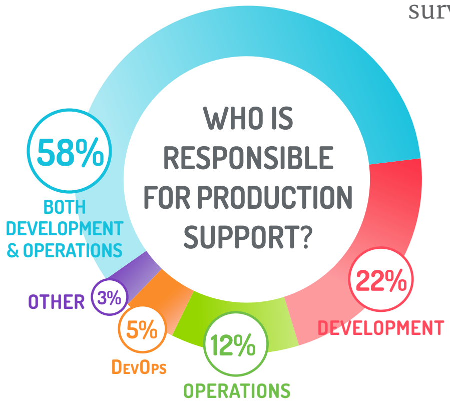
KEY RESEARCH FINDINGS

More than 900 IT professionals responded to DZone's 2015 Continuous Delivery Survey. Here are the demographics for this survey:

- Developers and Engineers made up 65% of the total respondents.
- 62% of respondents come from large organizations (100 or more employees) and 38% come from small organizations (under 100 employees).
- The majority of respondents are headquartered in Europe (48%) or the US (28%).

DIVISION BETWEEN DEV AND OPS IS CLOSING

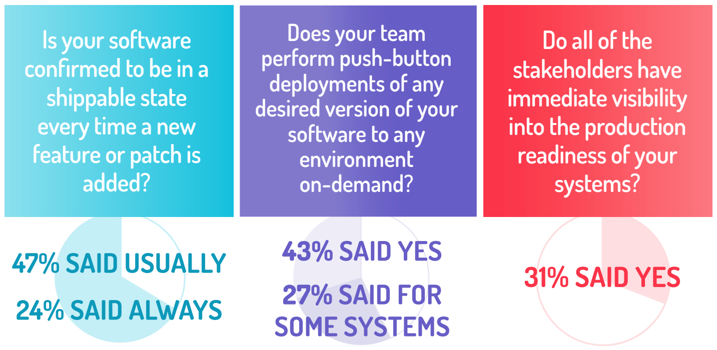
Continuous Delivery has always been positioned as being part of the larger DevOps philosophy. To implement this philosophy, many companies choose to create a team that is focused on cross-compatible skills for multiple disciplines between development and operations. 35% of the survey respondents have an officially designated DevOps team (up 5% from last year). For organizations without DevOps teams, the division of labor between development and operations teams can become blurry.



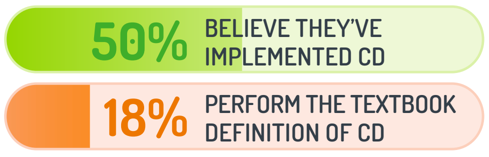
For example, development teams were only slightly more likely to perform code deployments to production (45%) than operations teams (32%). 58% of respondents that said both development and operations were both responsible for production support.

MORE PROFESSIONALS ARE ACHIEVING CONTINUOUS DELIVERY

The authors of the Continuous Delivery methodology defined three key traits to determine when an organization has fully implemented its practices [1]. The panels below shows how many survey respondents have these traits in their systems:



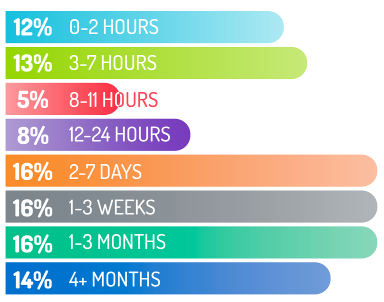
Most organizations are somewhere in the process of having adopted Continuous Delivery practices, but not having fully achieved the primary principles. 36% of those surveyed believe they have achieved CD for some of their projects, and 14% believe they are doing it in all of their projects. The combined statistic is that 50% of respondents have implemented Continuous Delivery for some or all of their projects, which represents a 9% growth in implementation over the last year. To determine the amount of respondents performing a "textbook" implementation, respondents were filtered by the three key traits of Continuous Delivery from the section above; 18% said "yes" or "always" for all of the questions. While this is smaller than the half of respondents who believe they've implemented Continuous Delivery, this number still represents a 10% growth compared to the number of respondents who had achieved "textbook" Continuous Delivery last year (8%).

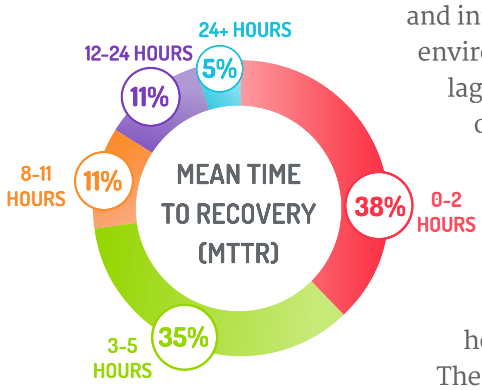


QUICK TO RECOVER AND SLOW TO FAIL

Among other deployment and support metrics, we checked into the Mean Time to Recover (MTTR) and Mean Time Between Failure (MTBF) for our respondents' support and operations teams. The survey indicated that recovery time after a failure (MTTR) averaged close to 6 hours. The respondents also reported that the average time between failures (MTBF) is just over 4 hours, with 13% reporting.

MEAN TIME BETWEEN FAILURES (MTBF)

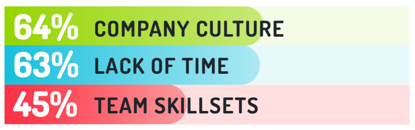




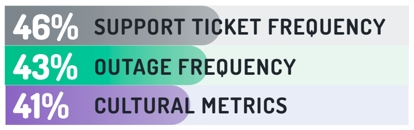
CULTURE IS BOTH OBSTACLE AND MEASURE OF SUCCESS

The three biggest barriers to Continuous Delivery adoption are company culture (64%), a lack of time (63%), and team skillsets (45%). This is the second time that company culture and a lack of time have topped the list of reasons why IT professionals are having problems adopting CD practices. The healthy growth of certain practices within a company culture has always been a major focus within DevOps, so it seems natural that negative culture would be a barrier to implementation. It's not surprising then that the responsiveness of DevOps culture can also be used to measure the success of Continuous Delivery. Cultural metrics (41%) are the third most important measure of success after support ticket frequency (46%) and outage frequency (43%).

MAIN BARRIERS TO ADOPTING CD

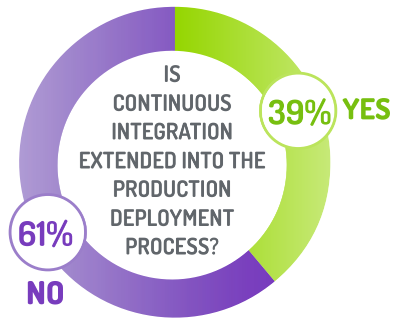


MEASURES OF DEVOPS EXCELLENCE



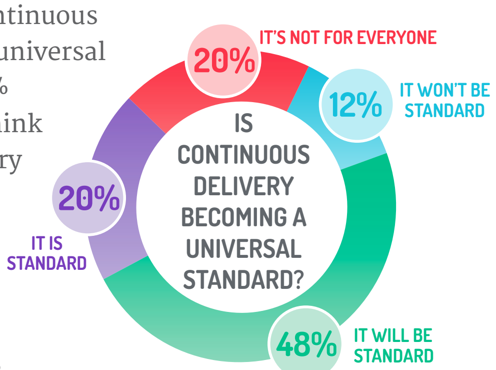
CONTINUOUS DELIVERY IS SPREADING TO OTHER ENVIRONMENTS

Continuous Delivery isn't a hard sell for developers. 61% say that they have already implemented it for their application build environments, and only 6% have no desire to do so. Database and infrastructure environments are still lagging behind, but they've seen decent growth over the last year. 30% of respondents have implemented it for database environments and 22% have implemented it for infrastructure, which represents a combined 13% growth in these environments. Another positive sign is that over half of all respondents hope to implement CD for those environments. The survey results do show some promising stats for organizations taking the first steps toward complete CD. 39% of respondents say they have extended their CI practices to production deployments.



CONTINUOUS DELIVERY IS BECOMING THE UNIVERSAL STANDARD

As popular as Continuous Delivery has become, it would be a stretch to say it's currently a universal standard for software delivery, though it's not that hard to say that it's well on its way. Respondents largely praised Continuous Delivery as a near universal standard, with 20% saying that they think Continuous Delivery is currently a universal standard, and 48% saying it will soon become the standard; that's a combined 68% of respondents. Even the negative responses were relatively tame—20% of respondents just don't think CD practices are appropriate for every environment. 12% said they just don't think it'll be a universal standard.



[1] http://martinfowler.com/bliki/ContinuousDelivery.html

DZone PRESENTS

CONTINUOUS DELIVERY visualized

Continuous Delivery advocates the creation of maximally automated deployment pipelines.

This visualization of an optimally (but not entirely) automated deployment pipeline shows how Continuous Delivery works.

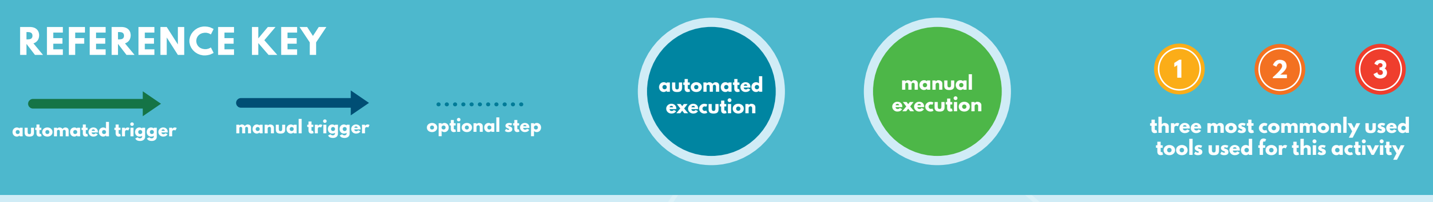
Each stage (big circle) is composed of multiple activities (little circles). Each activity can be automated or otherwise facilitated by various (mostly open-source) tools. We surveyed our audience to see which tools they used for which deployment-related activities. The three most commonly used tools are listed next to each activity.

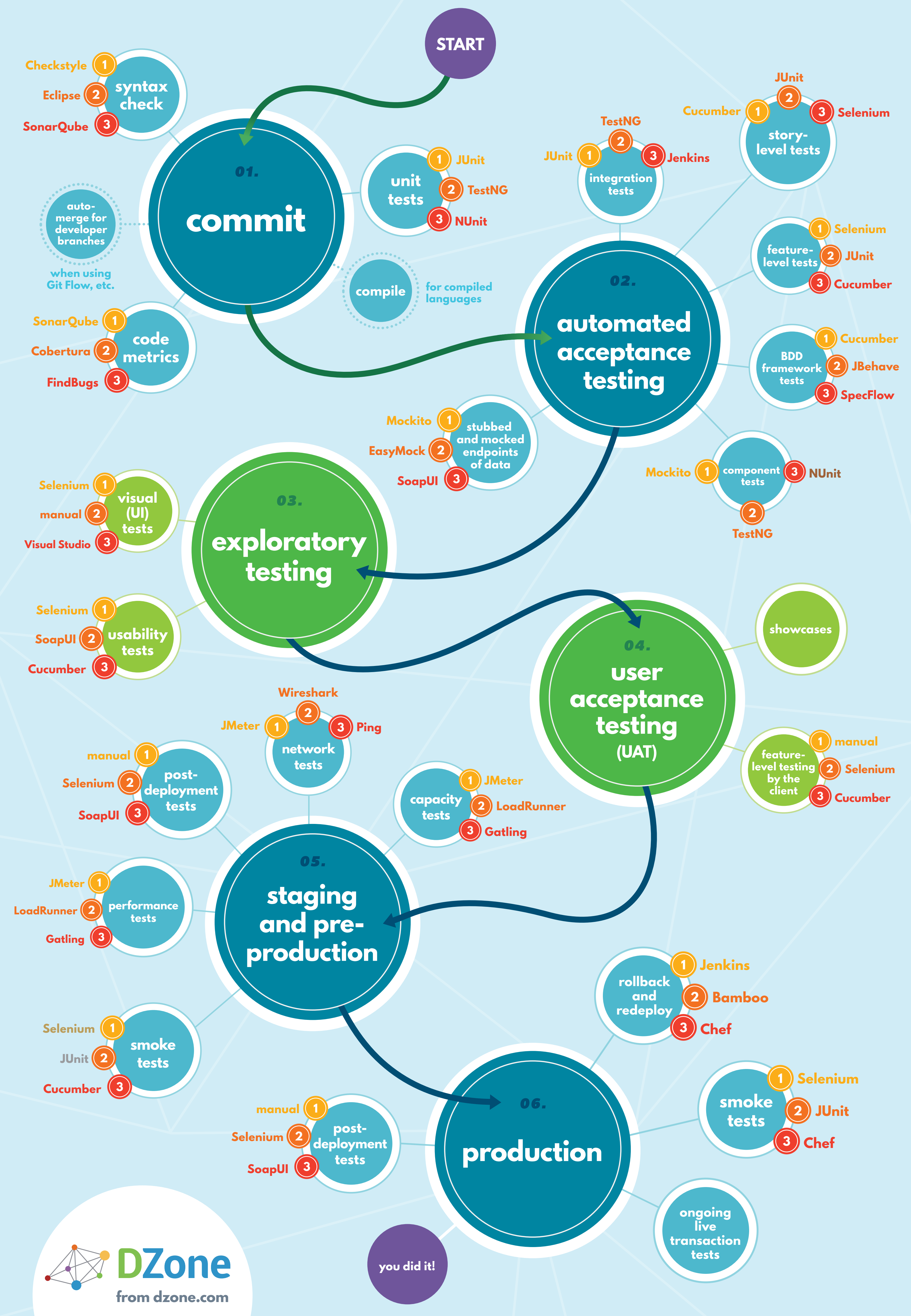
- Any long-running step, such as UAT, Pre-Production testing, or Exploratory Testing, can happen even after the change has already been deployed to Production.
- If significant issues are found in any long-running step, and the change has not been deployed to Production, the team should manually halt the pipeline.
- If significant issues are found in any long-running step, and the change has already been deployed to Production, the team should rollback Production to the last working release.

Diagrams are based on Jez Humble's diagrams from the Continuous Delivery blog (http://continuousdelivery.com/2010/09/deployment-pipeline-anti-patterns)

Special thanks to Matthew Skelton for helping build these diagrams.

REFERENCE KEY





Continuous Delivery Maturity Checklist

Check the boxes next to the practices you currently perform to see your maturity in each area of Continuous Delivery. Add up your score at the end based on the highest levels you checked.

Source Control

BASELINE
- ☐ Early branching
- ☐ Branches tend to remain apart

NOVICE
- ☐ Branches are used for isolating work
- ☐ Merges are common

INTERMEDIATE
- ☐ Pre-tested commits
- ☐ Integration branch is pristine

ADVANCED
- ☐ All commits are tied to tasks
- ☐ History used to rewrite features before pushing to central repository
- ☐ Version control DB schema changes

EXPERT
- ☐ Traceability analysis and release notes auto-generated
- ☐ Commits are clean enough for the master branch/trunk

Testing & QA

BASELINE
- ☐ Automatic unit testing with every build
- ☐ Code coverage is measured

NOVICE
- ☐ Peer-reviews
- ☐ Mockups & proxies used

INTERMEDIATE
- ☐ Periodic static code analysis
- ☐ Automated functional testing

ADVANCED
- ☐ Integrated management and maintenance of the test data
- ☐ Automated performance and security tests in target environments

EXPERT
- ☐ Automated acceptance testing

Build Process

BASELINE
- ☐ Official builds are not performed on developers' machines
- ☐ Self-service build or nightly build

NOVICE
- ☐ System polls source control and builds on commit
- ☐ Build artifacts are managed, some manual scripts still used

INTERMEDIATE
- ☐ Build artifacts are managed by purpose-built tools, no manual scripts
- ☐ Dependencies are managed in a repository

ADVANCED
- ☐ Distributed builds on build cluster, can be done in sequence
- ☐ Source control tells system when to build, no polling

EXPERT
- ☐ Build environments based on VMs
- ☐ Streams are never "broken"
- ☐ Gated commits

Visibility

BASELINE
- ☐ Build status notification is sent to committer

NOVICE
- ☐ Latest build status is available to all team members

INTERMEDIATE
- ☐ Trend reports are automatically generated from build server events
- ☐ People outside the team can subscribe to build statuses

ADVANCED
- ☐ Stakeholders have dashboards with real-time product and dependency stats

EXPERT
- ☐ Cross-team data mining and analysis

Deployment

BASELINE
- ☐ Fully scripted deployments

NOVICE
- ☐ Push-button deployments to test environments

INTERMEDIATE
- ☐ Auto deploy to first test environment
- ☐ Standard deployments across all environments
- ☐ Push-button deployments to production

ADVANCED
- ☐ Automated deployments after tests pass
- ☐ Database deployments
- ☐ Multi-tier deployments

EXPERT
- ☐ Ability to implement continuous deployment

Overall Maturity Scorecard

For each check mark add the assigned number of points and total them for each section:

POINT KEY:

BASELINE 0 PTS	**INTERMEDIATE** 2 PTS
NOVICE 1 PT	**ADVANCED** 3 PTS
	EXPERT 4 PTS

TALLY YOUR SCORES:

Source Control	
Build Process	
Testing & QA	
Deployment	
Visibility	
TOTAL	

0-7 ADEQUATE **8-11** AVERAGE **12-14** SKILLED **15-17** ADEPT **18-20** MASTER

INSPIRED BY CHRIS SHAYAN AND ERIC MINICK

SOURCES: CHRISSHAYAN.ATLASSIAN.NET/WIKI/DISPLAY/MY/2013/07/23/CONTINUOUS+DELIVERY+MATRIX
IBMDW.NET/URBANCODE/DOCS/CONTINUOUS-DELIVERY-MATURITY-MODEL/

ELIMINATING ROADBLOCKS ON THE PATH TO CONTINUOUS DELIVERY

AUTOMATED TESTING PLAYS A KEY ROLE IN SUCCESSFULLY implementing Continuous Delivery. We've witnessed enterprises increasingly adopting fully automated delivery pipelines, successfully accelerating release cycles, achieving consistently high quality, and allowing their development teams to focus on writing software rather than on the mechanics of delivering it.

An example of an idealized, modern software delivery pipeline might look like the following:

- Plan user stories and manage issues with a project management tool like JIRA.
- Collaborate on code via GitHub pull requests or a code review tool.
- Kick off a build in a CI system like Jenkins or Bamboo.
- Automatically run unit and functional tests with open source testing tools like xUnit and other testing frameworks, and automation tools like Selenium and Appium.
- Deploy with an IT automation tool like Puppet or Chef, or using a PaaS.
- Monitor performance and impact on business metrics with systems like New Relic and Mixpanel.

Different organizations make different tool choices, of course,and there are usually a few pieces handled in a custom way due to the need to work with legacy systems or specialized processes. Whatever the challenge, software development teams have a thriving ecosystem of tools and services available to support a CD workflow. Indeed, it's this abundance of tool choices that is changing the equation and making Continuous Delivery possible for more and more teams.

Automated testing itself has come a long way as a part of this trend. Starting from early "test automation" tools designed to make QA teams more efficient, automated testing is now a critical part of automated delivery pipelines that are expected to run through complete test suites many times a day, with little tolerance for manual intervention, false failures, or infrastructural reliability problems.

Errors or bottlenecks introduced by automated testing infrastructure can break your build and block your deploy pipeline, creating expensive delays for software developers. Running automated tests rapidly and reliably is therefore critically important to a successful Continuous Delivery process.

By providing a high-reliability, scalable automated testing platform, we've been able to help enterprises sweep aside the time-consuming and error-prone maintenance of virtual machines and mobile devices, and allow them to instantly provision additional testing resources on demand. And we've prioritized fitting into the ecosystem of popular testing frameworks, CI systems, and surrounding tools and services, so that you can leverage existing investments and focus on optimizing your CD pipeline.

WRITTEN BY
Steve Hazel, *Cofounder, Chief Product Officer, Sauce Labs*

Automated Testing Platform by Sauce Labs

FUNCTIONAL AND UNIT TESTING

sauce LABS

Sauce Labs provides a complete testing platform for native, hybrid, and web apps, allowing users to run Selenium, Appium, JS unit, and manual tests in any language on over 450 browser, OS, and platform combinations.

PRICING	OPEN SOURCE
By number of VMs and test run minutes	Yes

CASE STUDY HomeAway is a family of 50 websites and hundreds of applications that provide the largest collection of vacation rental listings in the world. One of the challenges they face is supporting the diverse set of devices and browsers on which people view their apps. They also feel pressure to speed up delivery time. HomeAway leverages Sauce Labs for production monitoring using an internal tool named Green Screen. Developers built Selenium scripts to execute primary customer user flows through Sauce Labs against their family of vacation sites. The objective is for these tests is to always be green; however, if a test fails, they receive an alert from Sauce Labs and the company responds. As a result, HomeAway has found the biggest value gained from using a combination of Sauce Labs infrastructure, real-user monitoring, and running their Selenium and Appium frameworks continuously so that quality isn't compromised.

CUSTOMERS

- Yahoo!
- Capital One
- Twitter
- Bank of America
- Mozilla
- Zendesk
- Salesforce
- Puppet Labs

CI TOOL SUPPORT

- Jenkins
- Travis CI
- CircleCI
- Bamboo
- TeamCity

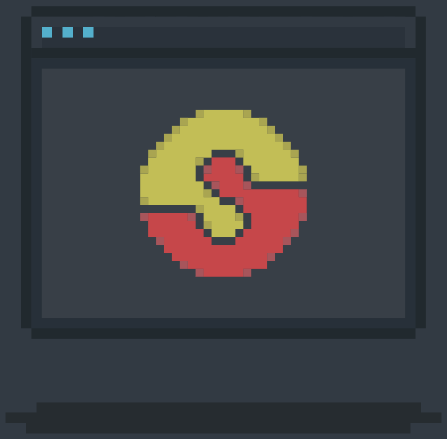
TESTING SUPPORT

- Cross-browser testing
- Mobile testing
- Selenium
- Appium
- JavaScript testing
- Manual testing

LANGUAGE SUPPORT

C# Java JavaScript Node.js Perl Ruby PHP Python

BLOG sauceio.com | TWITTER @saucelabs | WEBSITE saucelabs.com



MARTIAL ARTS
HAS BRUCE LEE.

AUTOMATED TESTING
HAS SAUCE LABS.

Maybe you can't do a one-fingered push-up, but you can master speed and scale with Sauce Labs. Optimized for the continuous integration and delivery workflows of today and tomorrow, our reliable, secure cloud enables you to run your builds in parallel, so you can get to market faster without sacrificing coverage.

Try it for free at saucelabs.com and see why these companies trust Sauce Labs.

Dropbox salesforce YAHOO! PayPal Bank of America VISA

sauce LABS