DevOps

Produced by:

Dr. Siobhán Drohan (sdrohan@wit.ie)

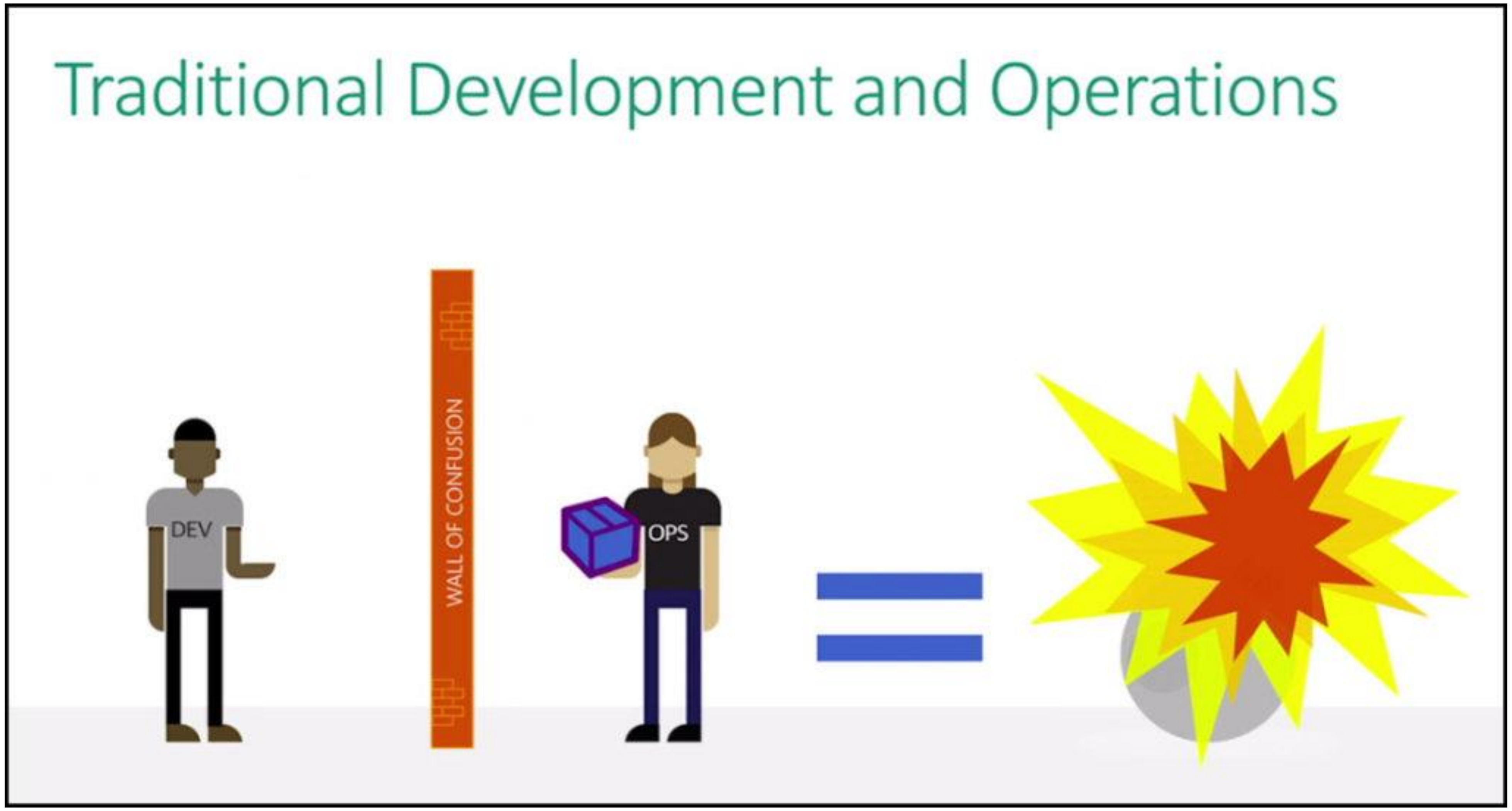
Eamonn de Leastar (edeleastar@wit.ie)

Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

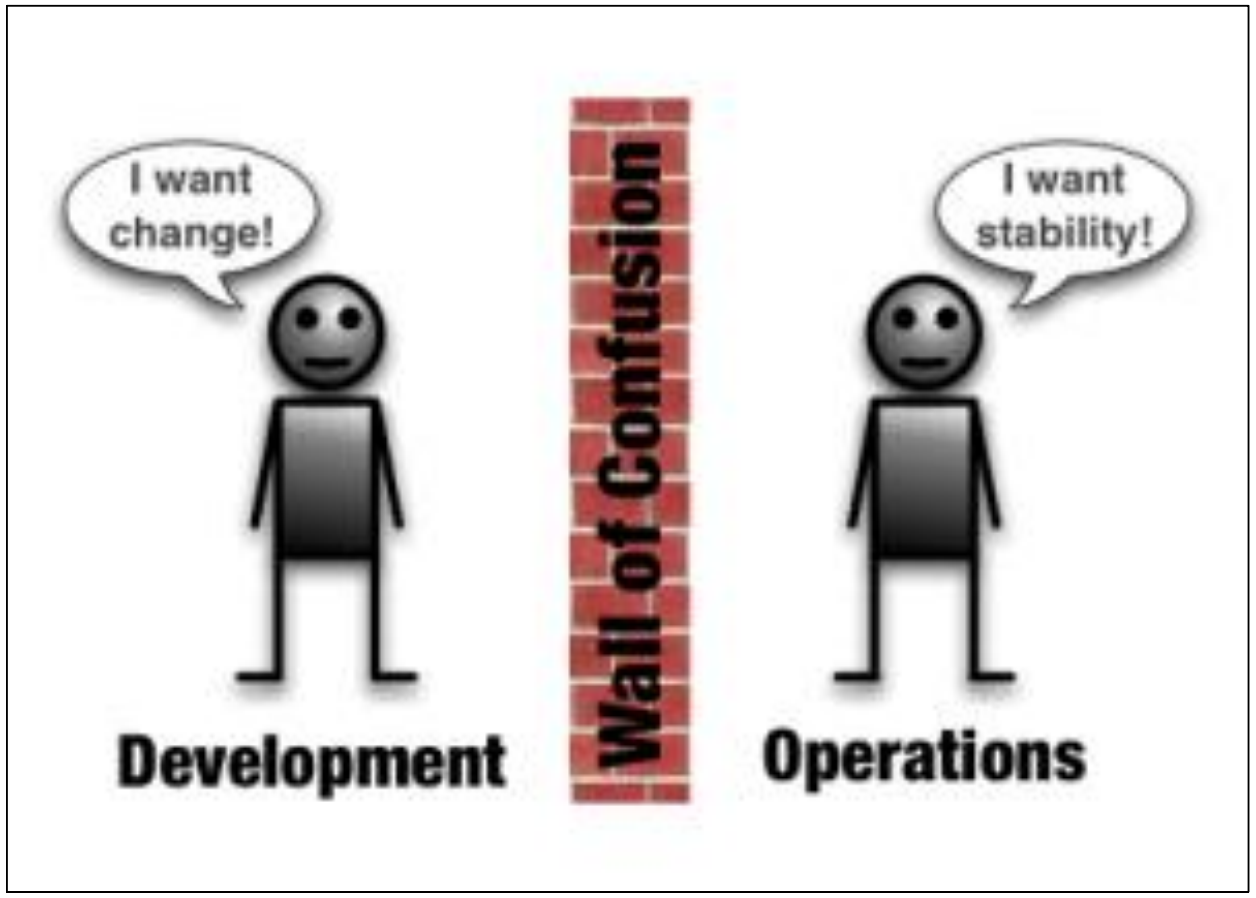
Department of Computing and Mathematics
http://www.wit.ie/

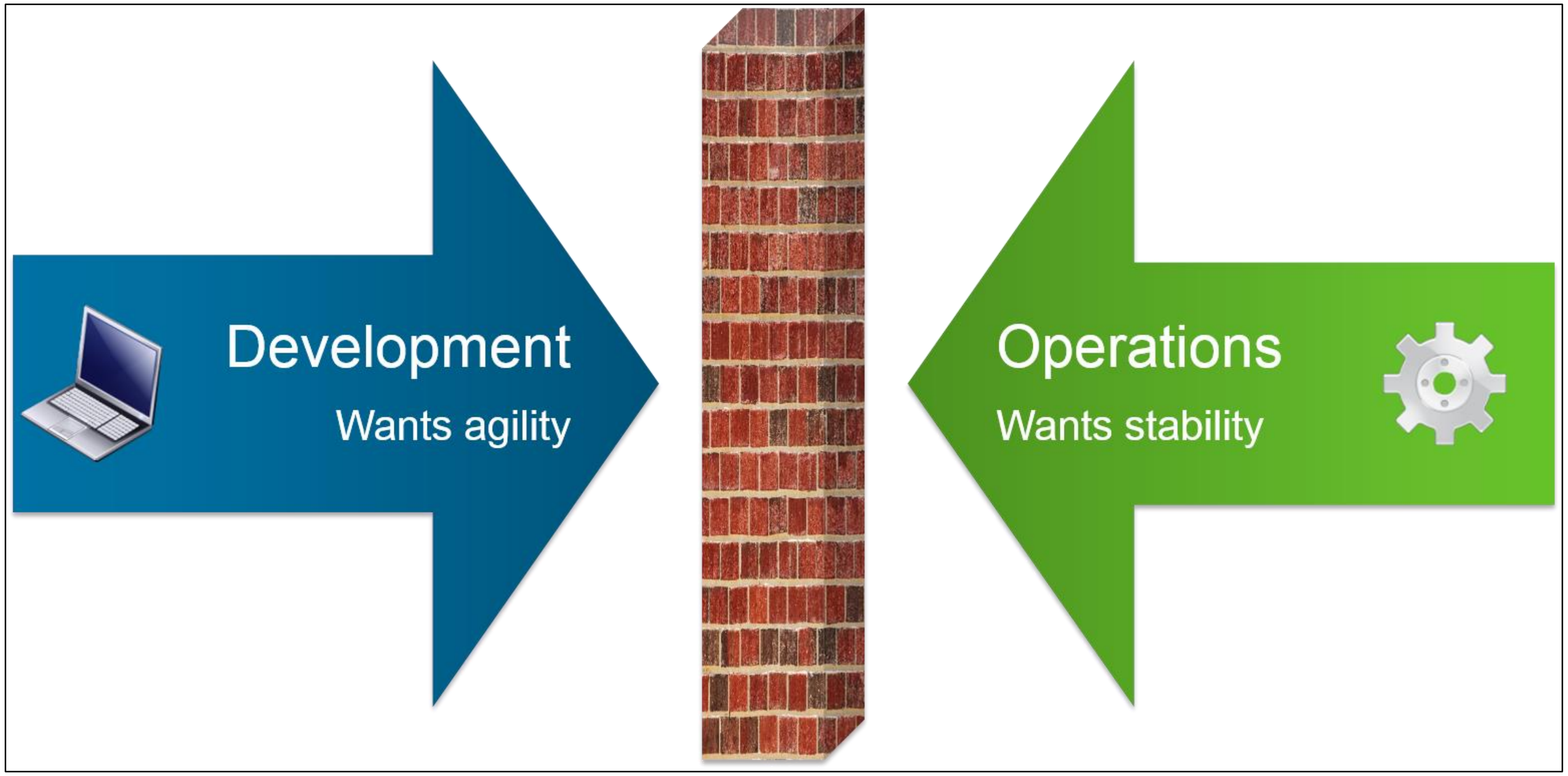
1. Dev team created a solution for production.

2. When it was finished they handed it over to the ops team.



3. Ops job is to implement the project in production by manually changing configuration files and other data in order to comply for deployment.


I want change!
Development
Wall of Confusion
I want stability!
Operations


Development
Wants agility
Operations
Wants stability

“The idea of shipping code faster has been a priority since the practice of software development began”

https://dzone.com/whitepapers/devops-is-crippled-without-continuous-integration

“DevOps is about

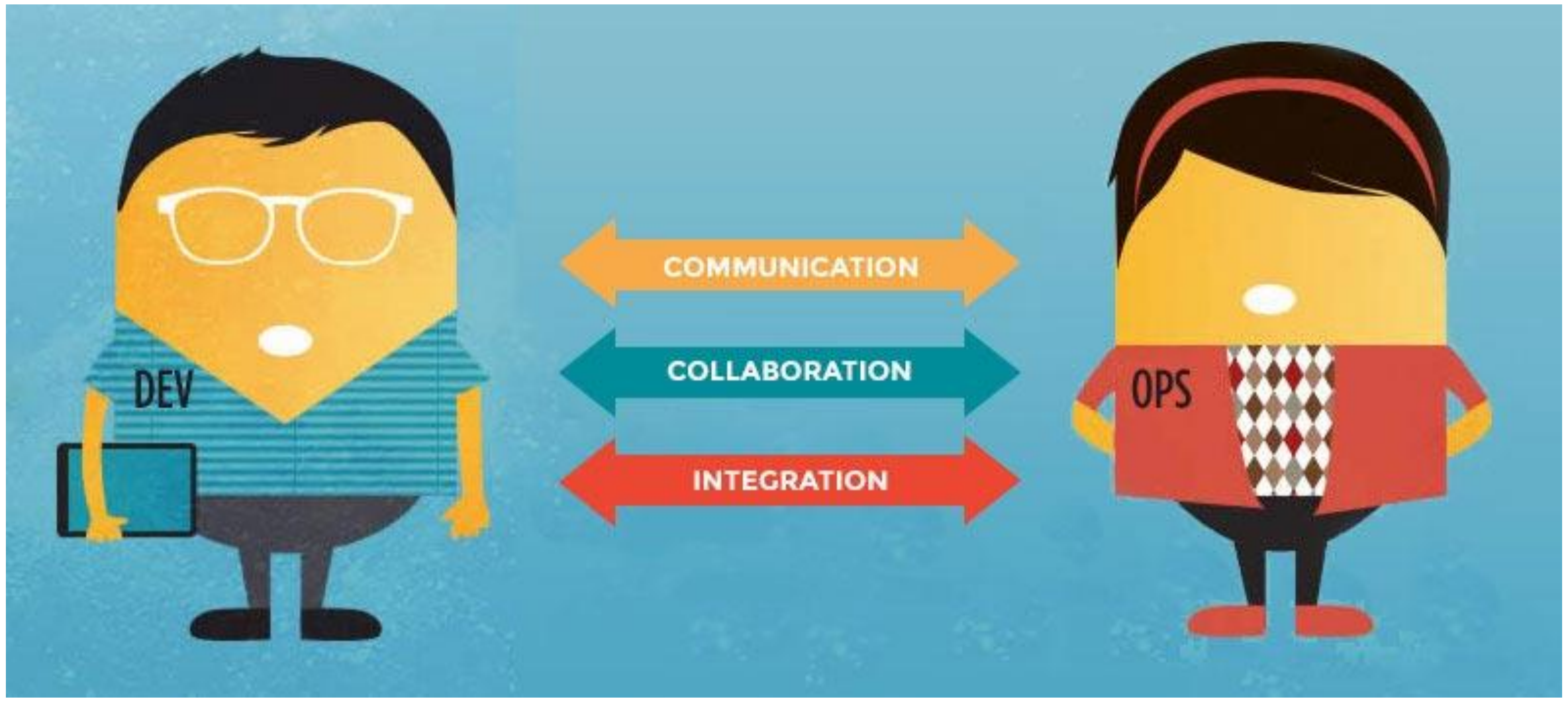
more frequent,

higher quality releases.”

https://dzone.com/whitepapers/devops-is-crippled-without-continuous-integration

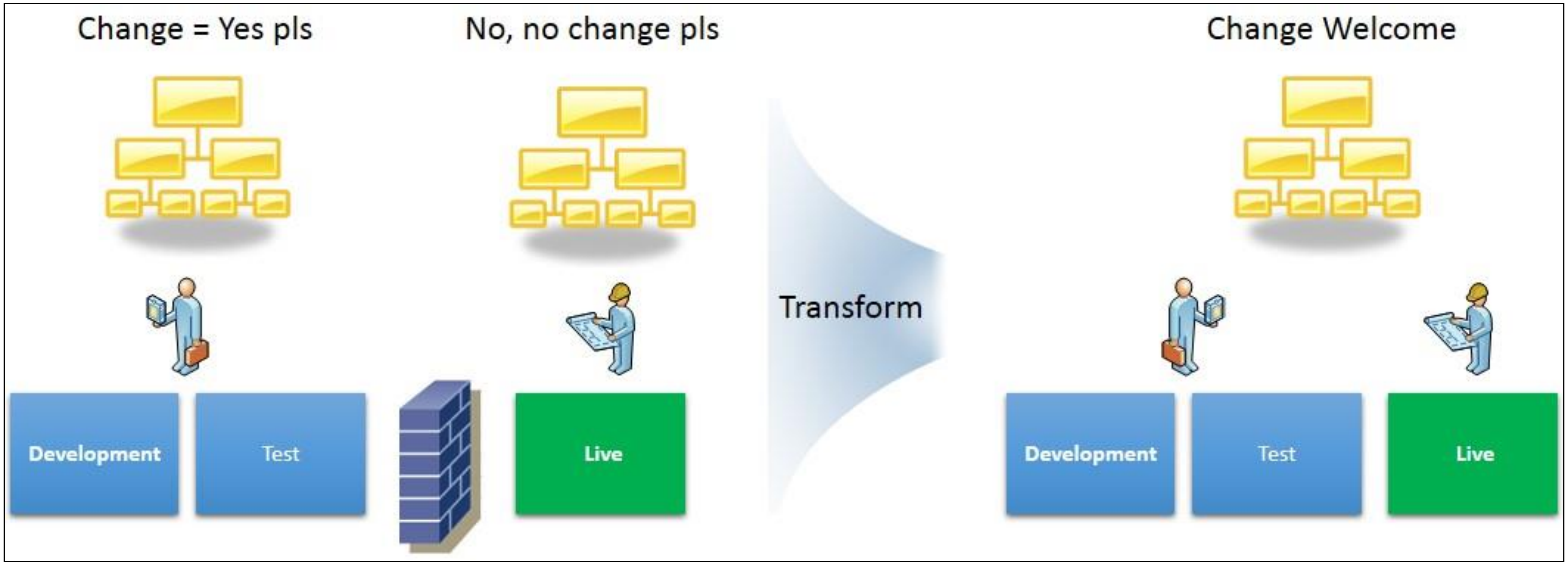
What is DevOps?

- DevOps is a **software development approach** that stresses:
  - Communication
  - Collaboration
  - Integration
  - Trust



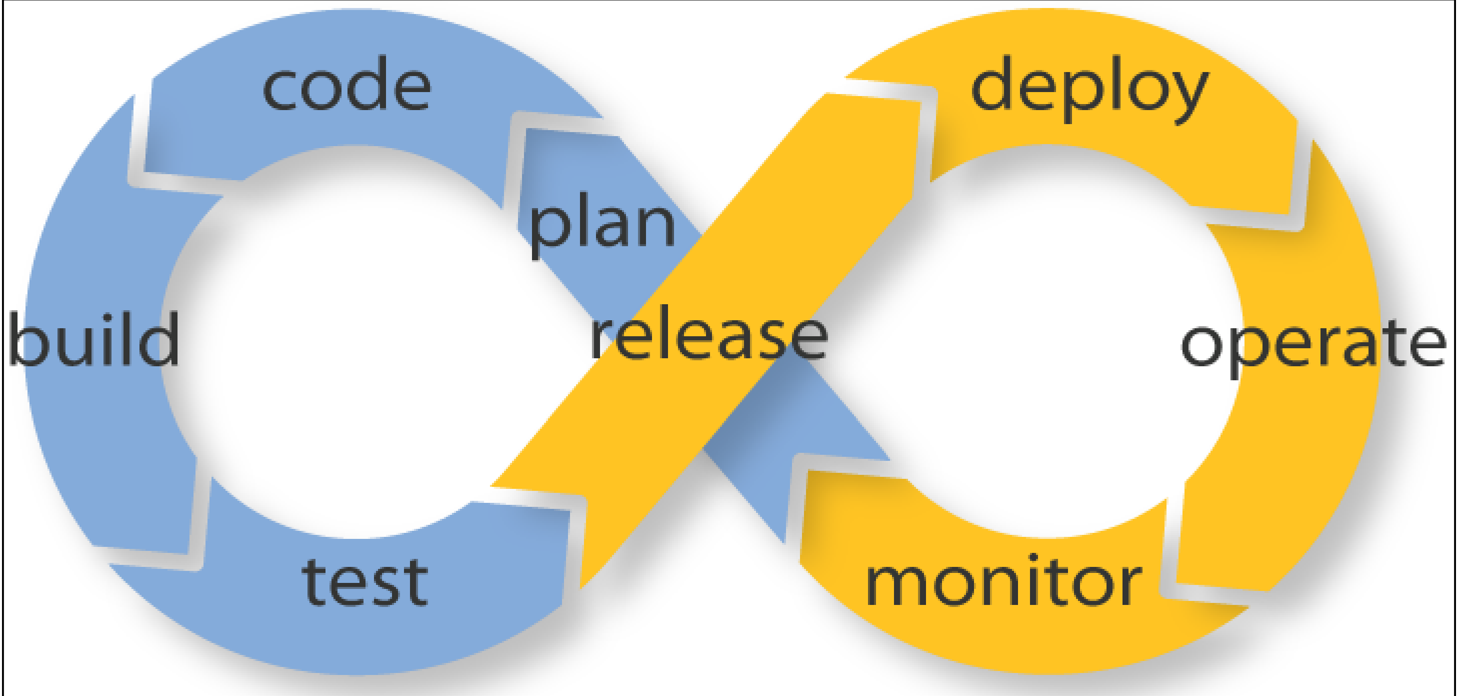
- between software developers and operations i.e. the merging of two different disciplines → DevOps!

With DevOps, change is welcome



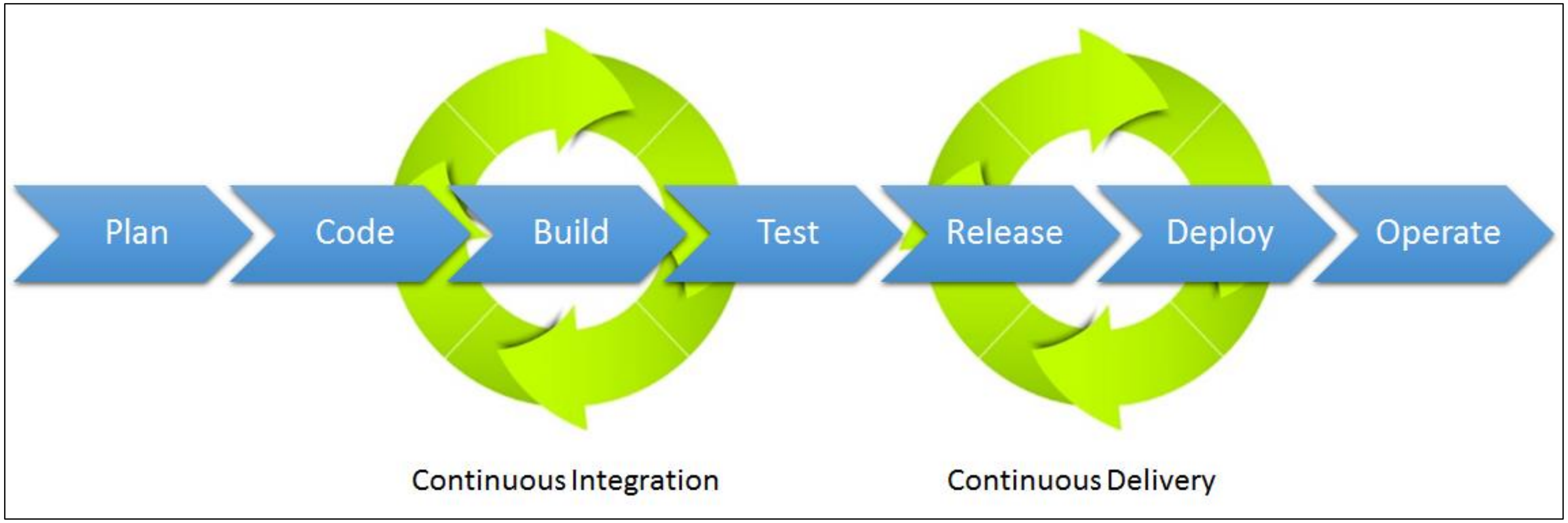
What is DevOps?

- DevOps allows us to build, deploy, and change our software with accelerated delivery cycle times.
- DevOps integration targets product delivery, quality testing, feature development, and maintenance releases in order to improve reliability and security and faster development and deployment cycles.

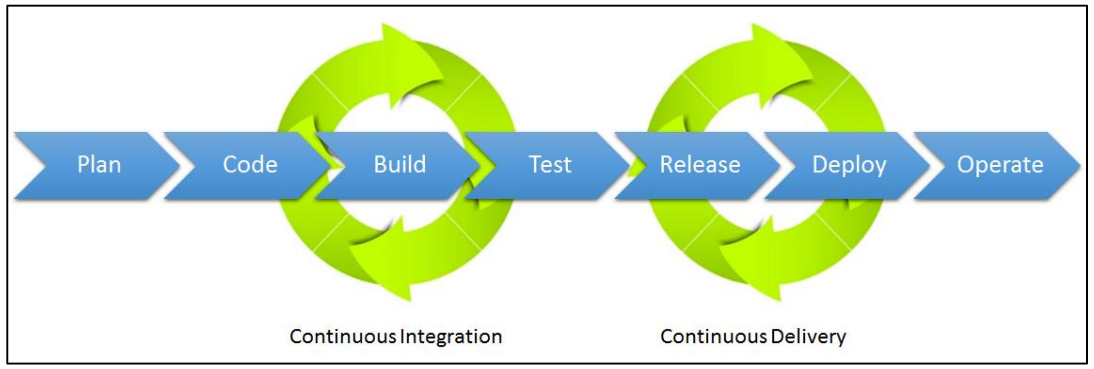


Endless Possibilities: DevOps can create an infinite loop of release and feedback for all your code and deployment targets.

DevOps enables the merging of Continuous Integration and Continuous Delivery



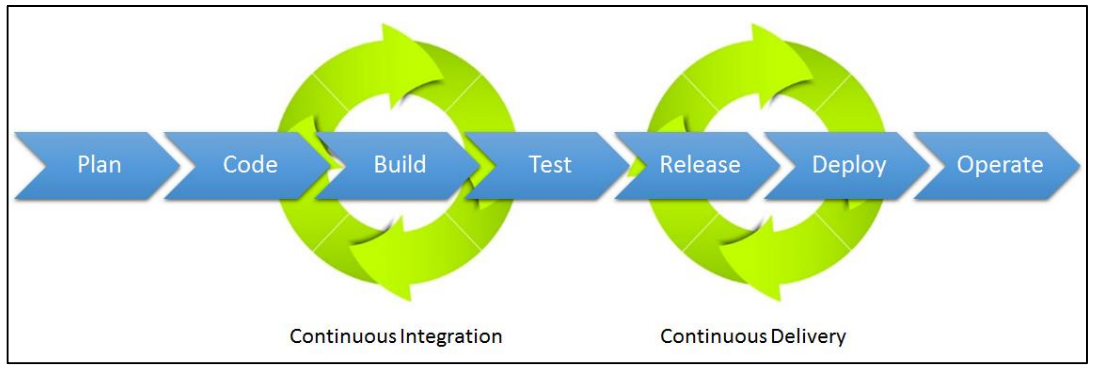
Continuous Integration



- The process of steadily adding new code commits to source code.
- Originally, a daily build was the standard for continuous integration.

http://searchsoftwarequality.techtarget.com/definition/Continuous-Software-Development

Continuous Integration



- The process of steadily adding new code commits to source code.

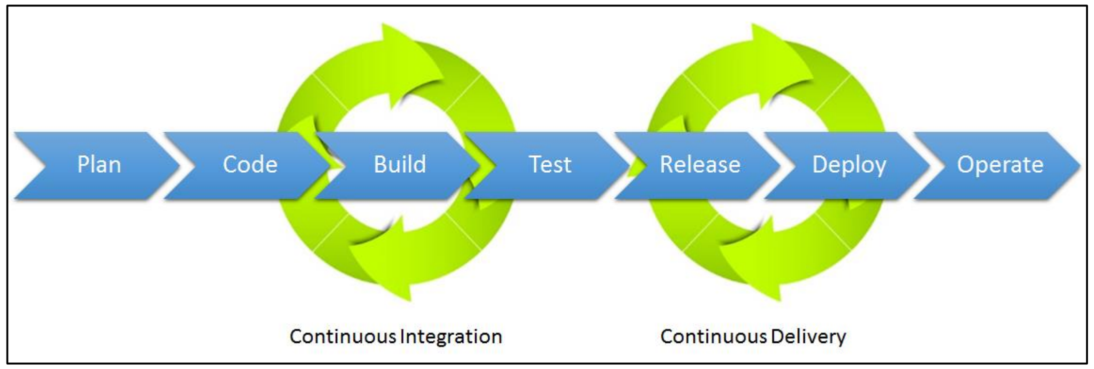
- Originally, a daily build was the standard for continuous integration.

- Today, team members submit work as soon as it is finished and a build would be conducted with each significant change.
  - Usually, a certain baseline of automated unit and integration testing is performed to ensure that new code does not break the build.
  - This way developers know as soon as they're done if their code will meet minimum standards and they can fix problems while the code is still fresh in their minds.

- An important advantage of continuous integration is that it provides developers with immediate feedback and status updates for the software they are working on.

http://searchsoftwarequality.techtarget.com/definition/Continuous-Software-Development

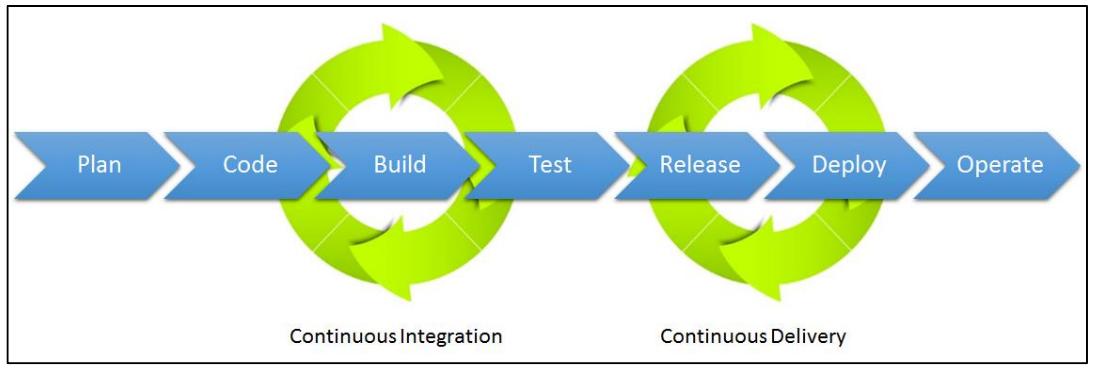
Continuous Delivery



"Continuous Delivery is the ability to get changes of all types — including new features, configuration changes, bug fixes and experiments — into production, or into the hands of users, safely and quickly in a sustainable way."

https://www.continuousdelivery.com

Continuous Delivery



"Continuous Delivery is the ability to get changes of all types — including new features, configuration changes, bug fixes and experiments — into production, or into the hands of users, safely and quickly in a sustainable way."

https://www.continuousdelivery.com

- Common goal of faster time to market for new services / releases.
- Approach whereby teams ensure that every change to the system can be released, and that any version can be released at the push of a button.

http://automic.com/blog/whats-the-difference-between-devops-and-continuous-delivery

Why Continuous Delivery?

Low-risk releases	Make software deployments painless, low-risk events that can be performed at any time, on demand

https://www.continuousdelivery.com/

Why Continuous Delivery?

Low-risk releases	Make software deployments painless, low-risk events that can be performed at any time, on demand
Faster time to market	Integration and test/fix phase of the traditional phased software delivery lifecycle to consume weeks or even months. When teams work together to automate the build and deployment, environment provisioning, and regression testing processes, developers can incorporate integration and regression testing into their daily work and we completely remove these phases.

https://www.continuousdelivery.com/

Why Continuous Delivery?

Low-risk releases	Make software deployments painless, low-risk events that can be performed at any time, on demand
Faster time to market	Integration and test/fix phase of the traditional phased software delivery lifecycle to consume weeks or even months. When teams work together to automate the build and deployment, environment provisioning, and regression testing processes, developers can incorporate integration and regression testing into their daily work and we completely remove these phases.
Higher quality	When developers have automated tools that discover regressions within minutes, teams are freed to focus their effort on user research and higher level testing activities such as exploratory testing, usability testing, and performance and security testing.

https://www.continuousdelivery.com/

Why Continuous Delivery?

Low-risk releases	Make software deployments painless, low-risk events that can be performed at any time, on demand
Faster time to market	Integration and test/fix phase of the traditional phased software delivery lifecycle to consume weeks or even months. When teams work together to automate the build and deployment, environment provisioning, and regression testing processes, developers can incorporate integration and regression testing into their daily work and we completely remove these phases.
Higher quality	When developers have automated tools that discover regressions within minutes, teams are freed to focus their effort on user research and higher level testing activities such as exploratory testing, usability testing, and performance and security testing.
Lower costs	Software changes. By investing in build, test, deployment and environment automation, we substantially reduce the cost of making and delivering incremental changes to software by eliminating many of the fixed costs associated with the release process.

https://www.continuousdelivery.com/

Why Continuous Delivery?

Low-risk releases	Make software deployments painless, low-risk events that can be performed at any time, on demand
Faster time to market	Integration and test/fix phase of the traditional phased software delivery lifecycle to consume weeks or even months. When teams work together to automate the build and deployment, environment provisioning, and regression testing processes, developers can incorporate integration and regression testing into their daily work and we completely remove these phases.
Higher quality	When developers have automated tools that discover regressions within minutes, teams are freed to focus their effort on user research and higher level testing activities such as exploratory testing, usability testing, and performance and security testing.
Lower costs	Software changes. By investing in build, test, deployment and environment automation, we substantially reduce the cost of making and delivering incremental changes to software by eliminating many of the fixed costs associated with the release process.
Better products	Continuous delivery makes it economic to work in small batches.

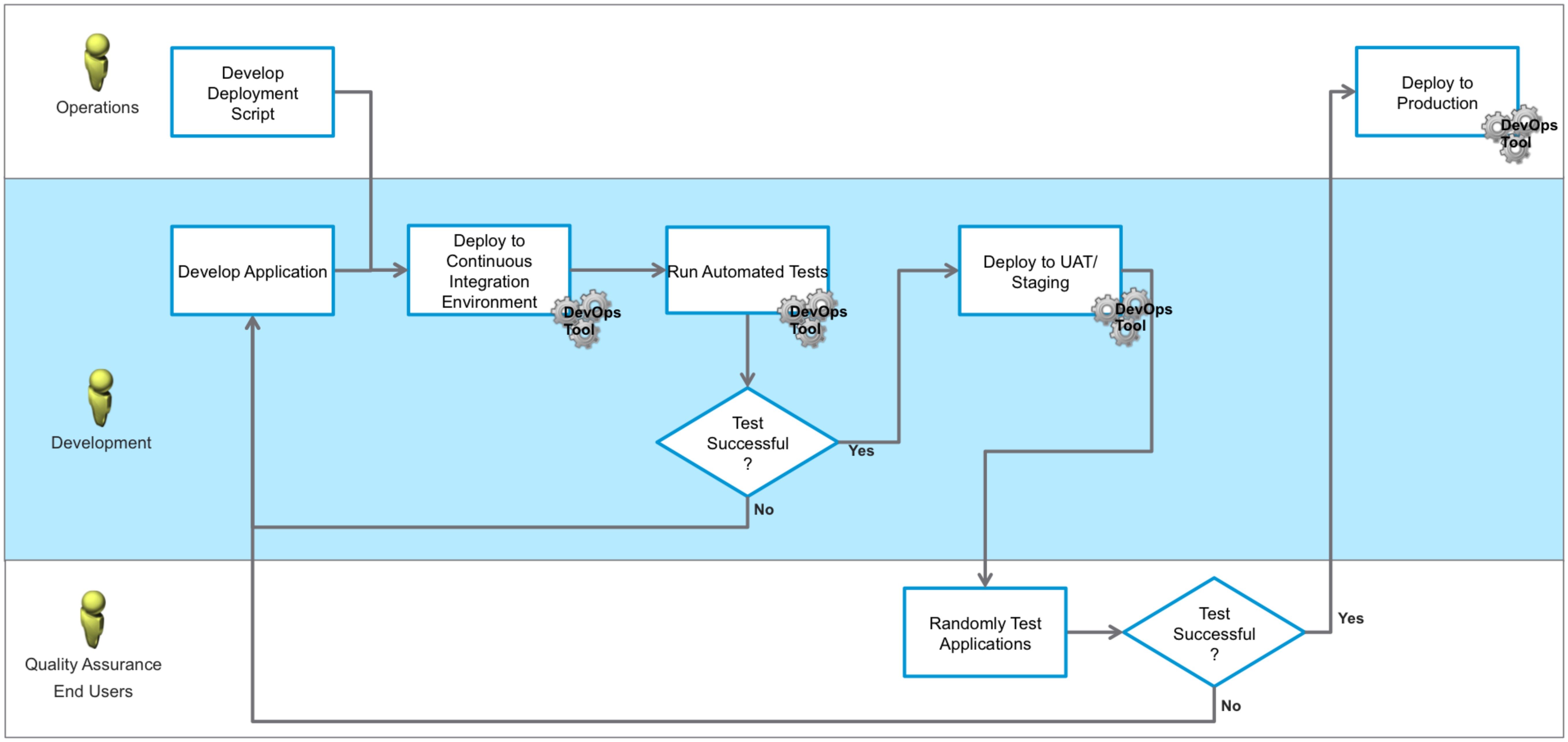
https://www.continuousdelivery.com/

Why Continuous Delivery?

Low-risk releases	Make software deployments painless, low-risk events that can be performed at any time, on demand
Faster time to market	Integration and test/fix phase of the traditional phased software delivery lifecycle to consume weeks or even months. When teams work together to automate the build and deployment, environment provisioning, and regression testing processes, developers can incorporate integration and regression testing into their daily work and we completely remove these phases.
Higher quality	When developers have automated tools that discover regressions within minutes, teams are freed to focus their effort on user research and higher level testing activities such as exploratory testing, usability testing, and performance and security testing.
Lower costs	Software changes. By investing in build, test, deployment and environment automation, we substantially reduce the cost of making and delivering incremental changes to software by eliminating many of the fixed costs associated with the release process.
Better products	Continuous delivery makes it economic to work in small batches.
Happier teams	Continuous delivery makes releases less painful and reduces team burnout. By removing the low-value painful activities associated with software delivery, we can focus on what we care about most—continuously delighting our users.

https://www.continuousdelivery.com/

Sample DevOps LifeCycle

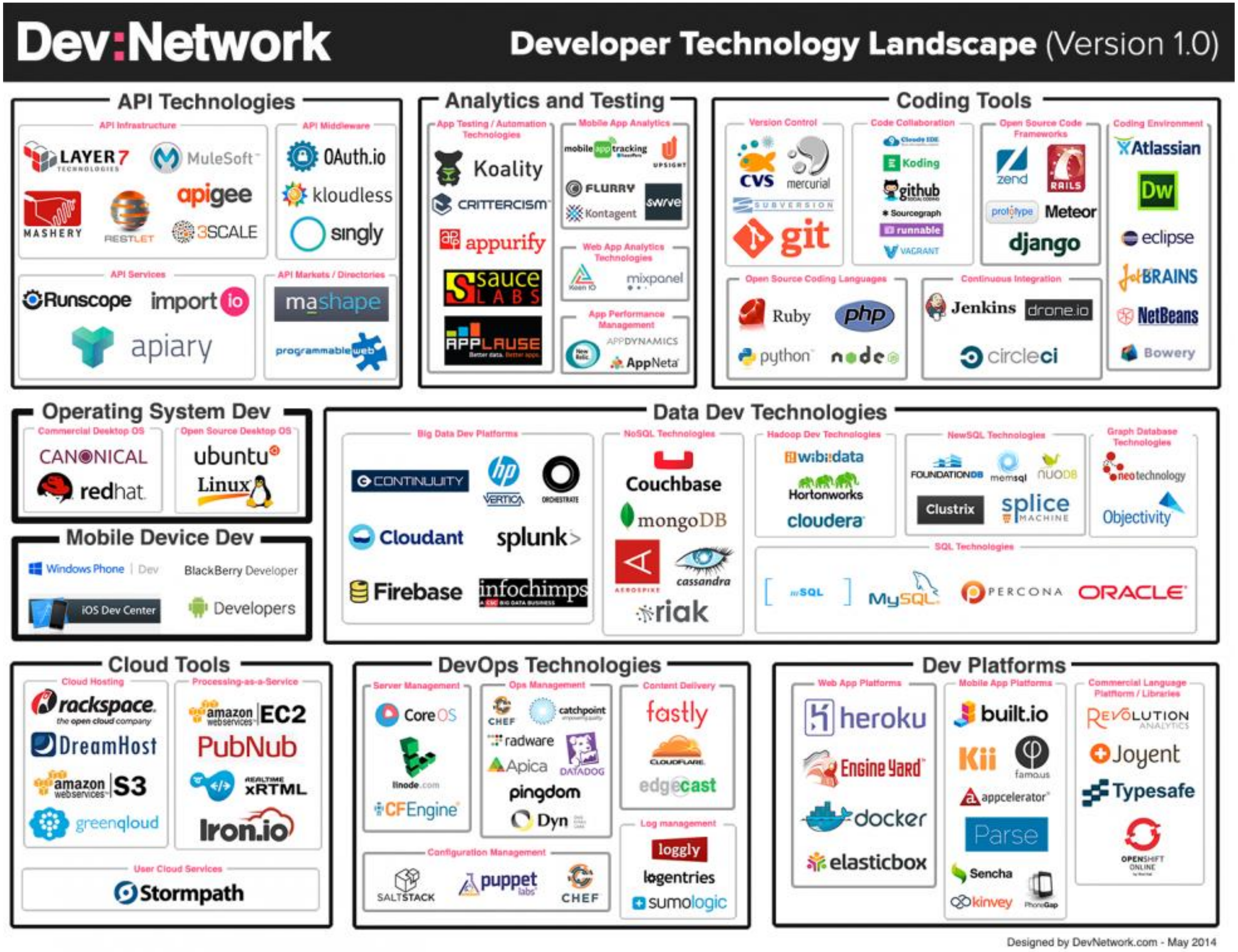


DevNetwork™

Developer Technology Landscape

2014

http://www.devnetwork.com/devnetwork-devtech-landscape/


Dev:Network
Developer Technology Landscape (Version 1.0)
API Technologies
API Infrastructure
LAYER7 TECHNOLOGIES
MuleSoft
MASHERY
RESTLET
apigee
3SCALE
API Middleware
OAuth.io
kloudless
singly
API Services
Runscope
import io
apiary
API Markets / Directories
mashape
programmableweb
Analytics and Testing
App Testing / Automation Technologies
Koality
CRITTERCISM
appurify
sauce LABS
APPLAUSE Better data. Better apps.
Mobile App Analytics
mobile app tracking
UPSIGHT
FLURRY
swrve
Kontagent
Web App Analytics Technologies
Keen IO
mixpanel
App Performance Management
New Relic
APPDYNAMICS
AppNeta
Coding Tools
Version Control
CVS
mercurial
SUBVERSION
git
Code Collaboration
Cloud9 IDE
Koding
github
Sourcegraph
runnable
VAGRANT
Open Source Code Frameworks
zend
RAILS
prototype
Meteor
django
Coding Environment
Atlassian
Dw
eclipse
JetBRAINS
NetBeans
Bowery
Open Source Coding Languages
Ruby
php
python
node
Continuous Integration
Jenkins
drone.io
circleci
Operating System Dev
Commercial Desktop OS
CANONICAL
redhat
Open Source Desktop OS
ubuntu
Linux
Mobile Device Dev
Windows Phone | Dev
BlackBerry Developer
iOS Dev Center
Developers
Data Dev Technologies
Big Data Dev Platforms
CONTINUUITY
hp VERTICA
ORCHESTRATE
Cloudant
splunk>
Firebase
infochimps A CSC BIG DATA BUSINESS
NoSQL Technologies
Couchbase
mongoDB
AEROSPIKE
cassandra
riak
Hadoop Dev Technologies
wibidata
Hortonworks
cloudera
NewSQL Technologies
FOUNDATIONDB
memsql
NUODB
Clustrix
splice MACHINE
Graph Database Technologies
neotechnology
Objectivity
SQL Technologies
mSQL
MySQL
PERCONA
ORACLE
Cloud Tools
Cloud Hosting
rackspace the open cloud company
DreamHost
amazon webservices S3
greenqloud
Processing-as-a-Service
amazon webservices EC2
PubNub
REALTIME xRTML
Iron.io
User Cloud Services
Stormpath
DevOps Technologies
Server Management
CoreOS
linode.com
CFEngine
Ops Management
CHEF
catchpoint
radware
Apica
DATADOG
pingdom
Dyn
Content Delivery
fastly
CLOUDFLARE
edgecast
Configuration Management
SALTSTACK
puppet labs
CHEF
Log management
loggly
logentries
sumologic
Dev Platforms
Web App Platforms
heroku
Engine Yard
docker
elasticbox
Mobile App Platforms
built.io
Kii
famo.us
appcelerator
Parse
Sencha
kinvey
PhoneGap
Commercial Language Platform / Libraries
REVOLUTION ANALYTICS
Joyent
Typesafe
OPENSHIFT ONLINE
Designed by DevNetwork.com - May 2014

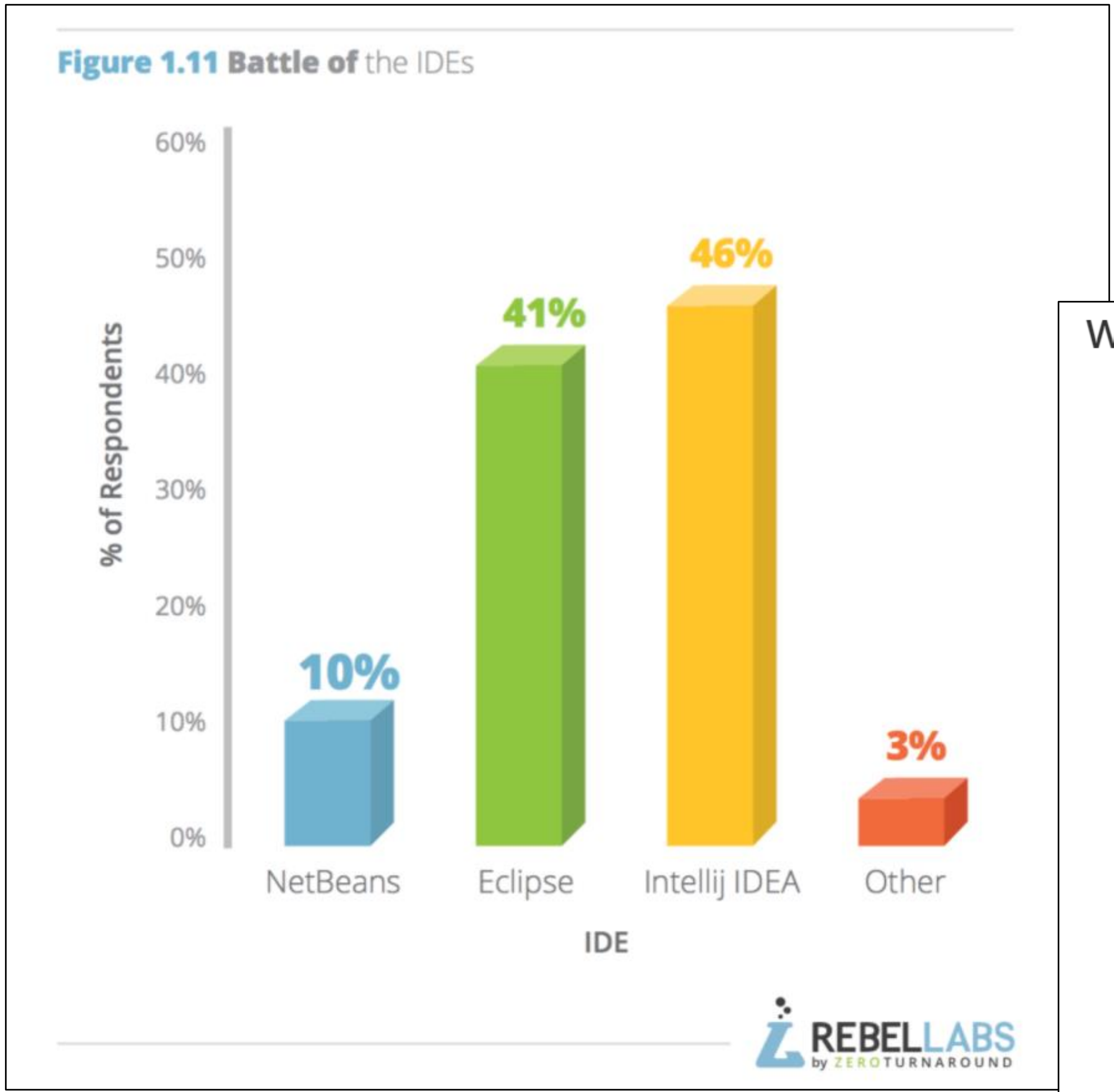
REBELLABS

Developer Tooling Landscape

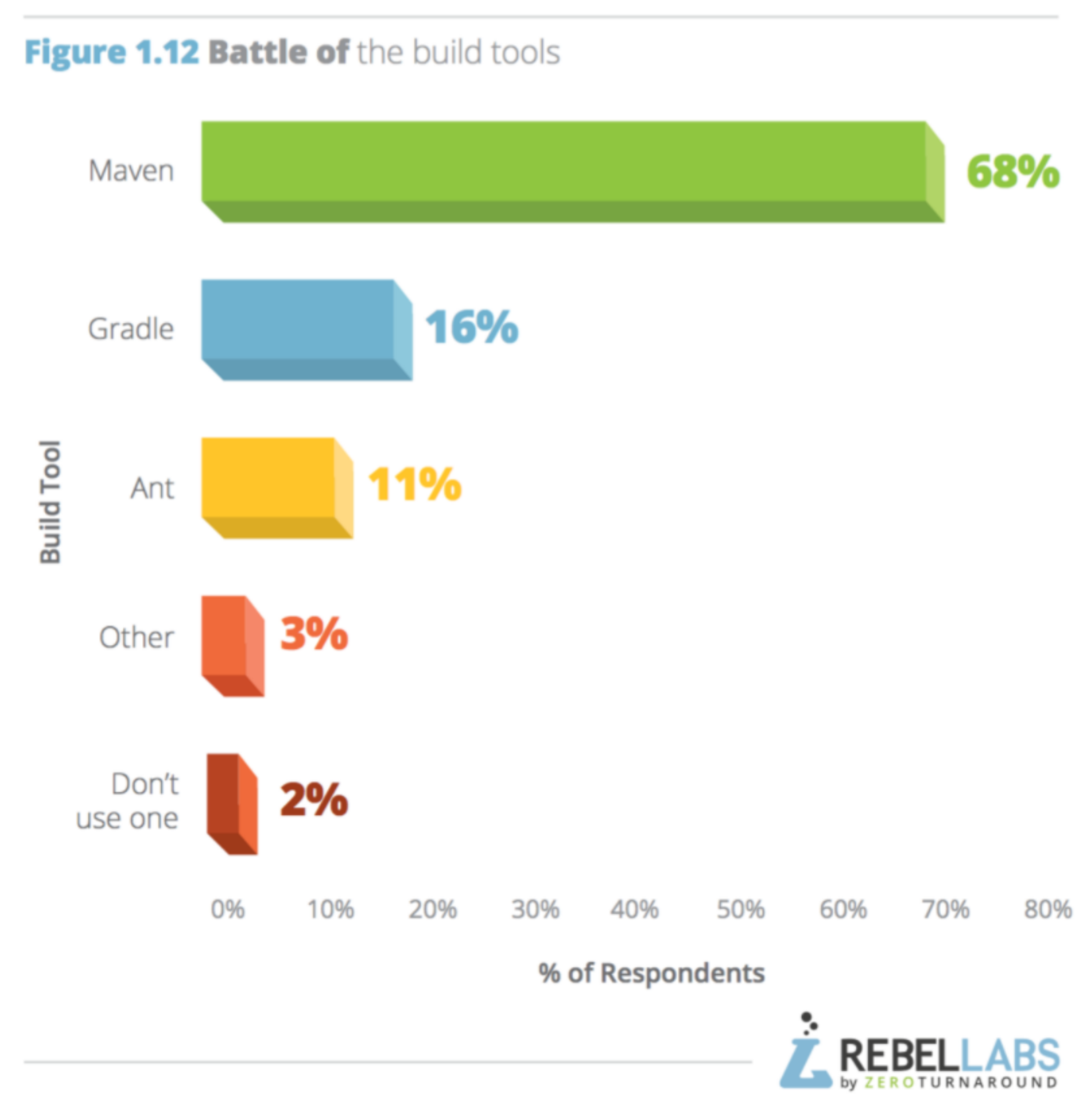
2016

https://zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-2016/

2016 Stats:



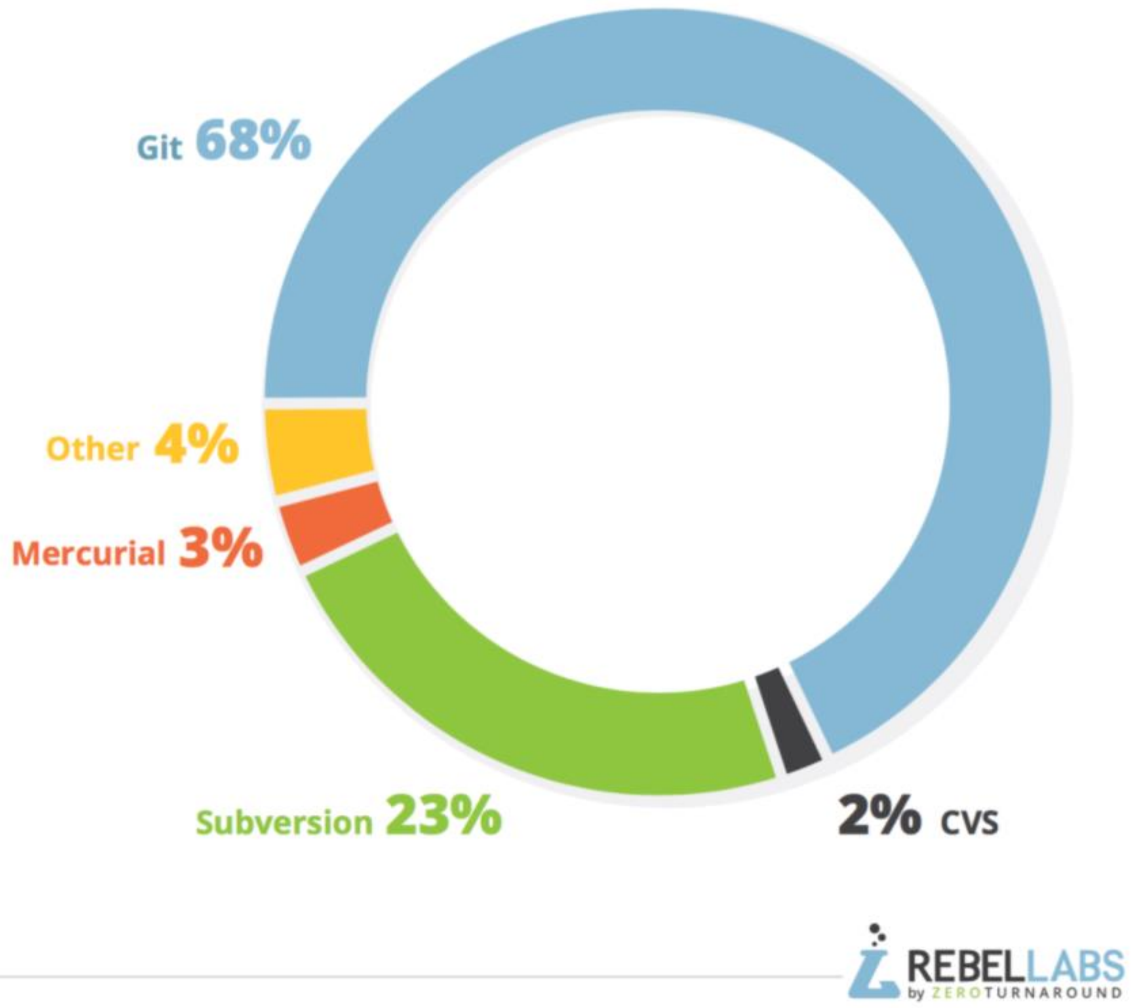
Which **build tool** do you use most often?



2016 Stats:

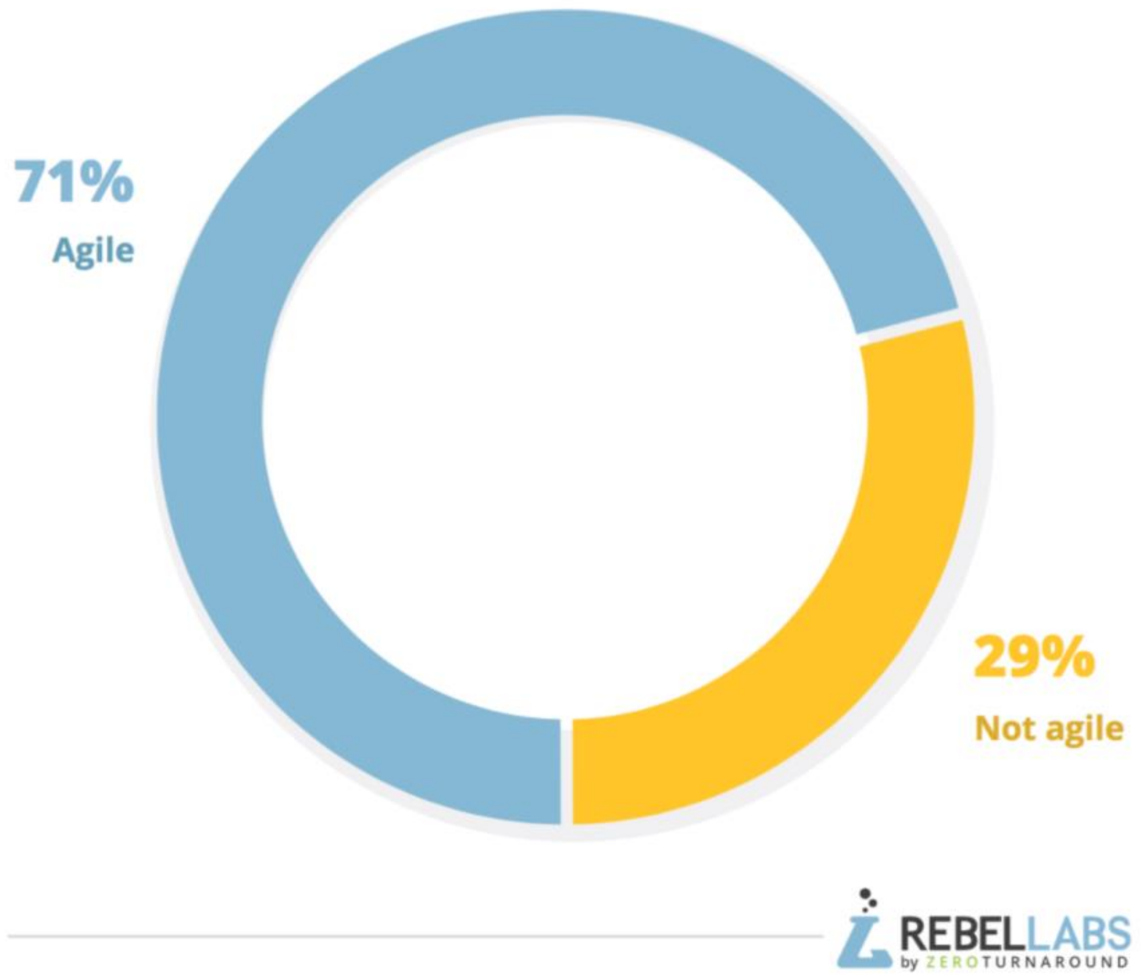
Which **VCS** do you use most often?

Figure 1.18 Most Commonly Used VCS



Is your team **agile**?

Figure 1.22 To Agile or Not To Agile?



Developer Ecosystem
2018

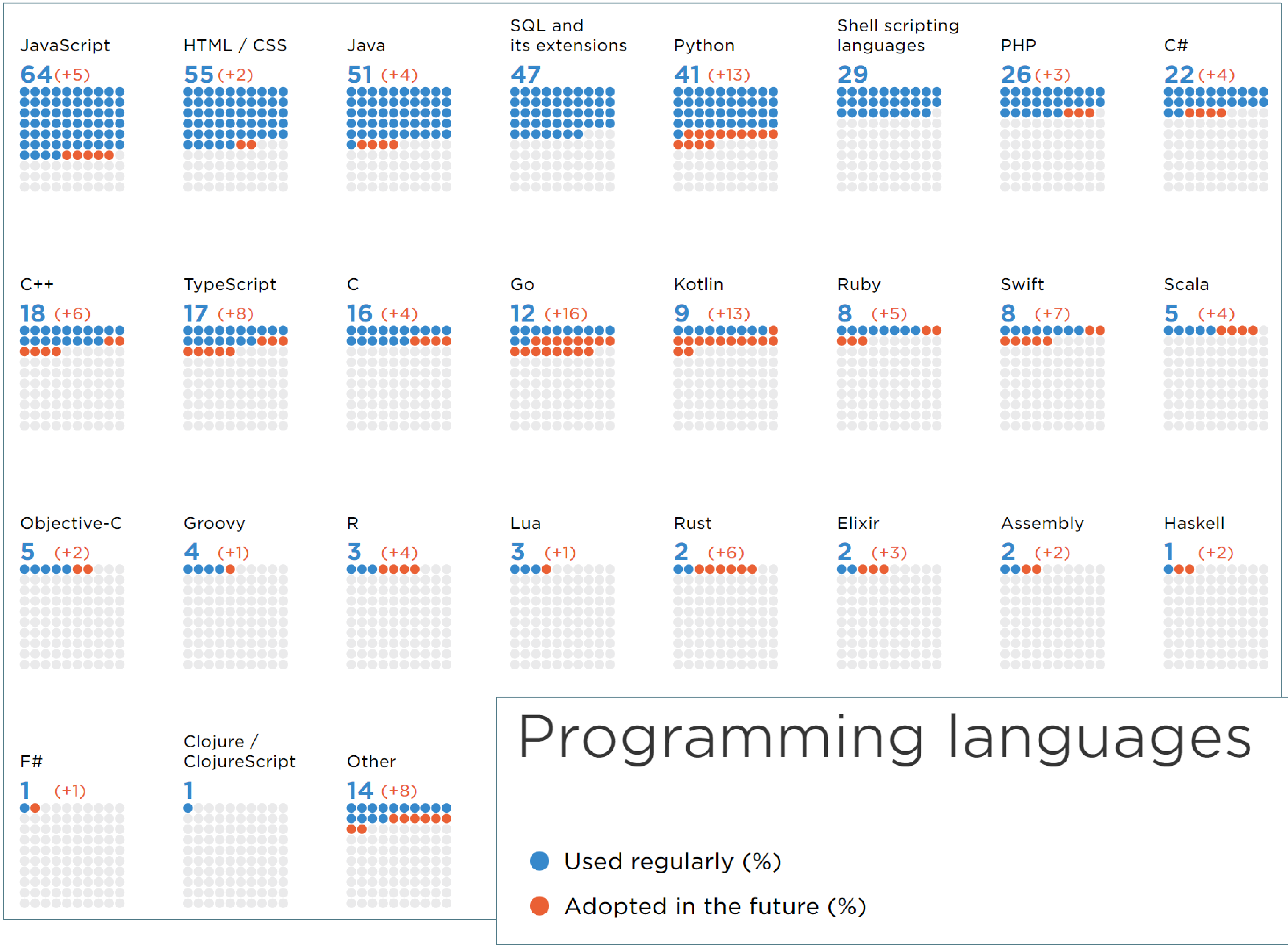
The State of Developer Ecosystem in 2018

In the beginning of 2018 we surveyed 6,000 developers to identify the State of Developer Ecosystem.

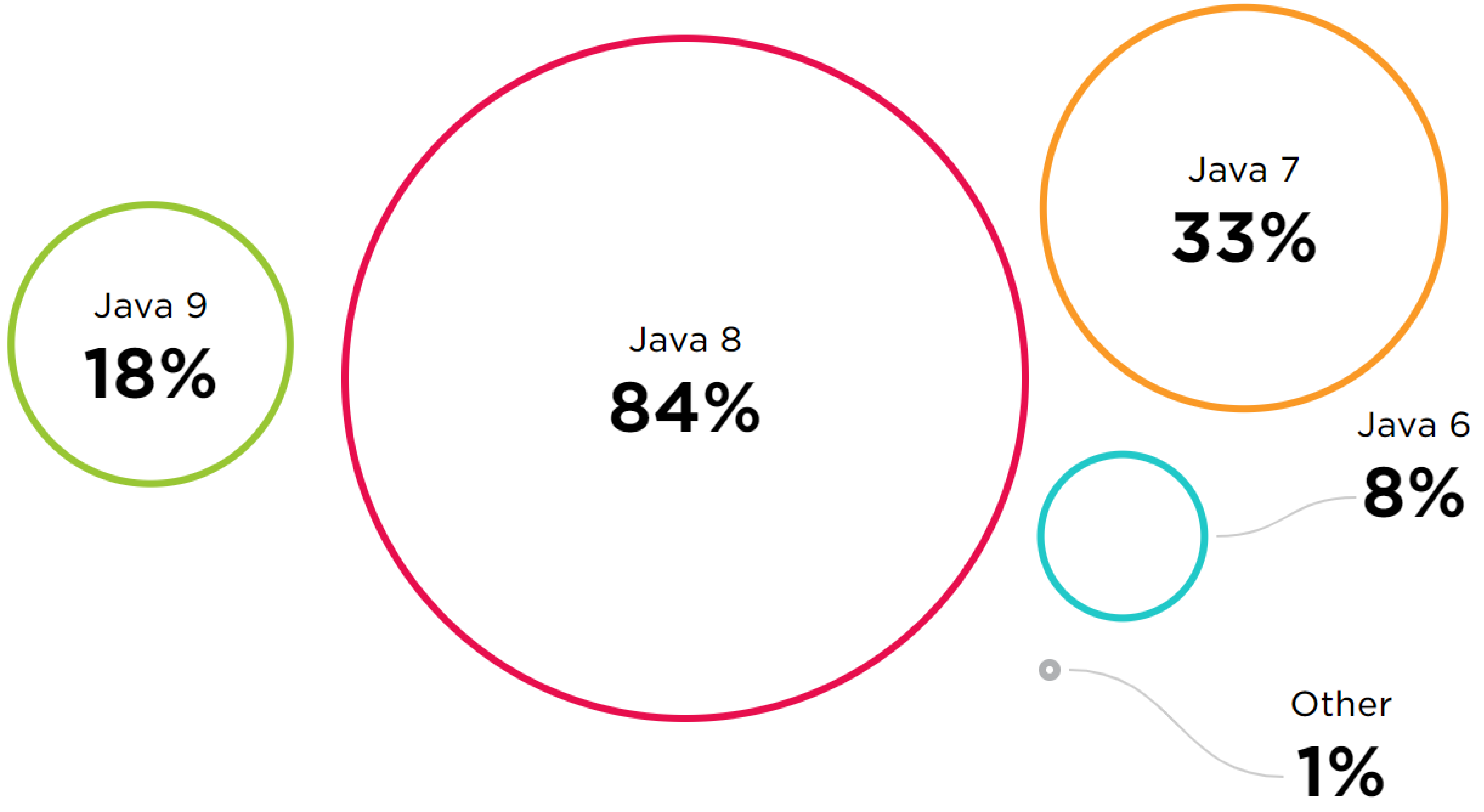
Here's what we learned.

https://www.jetbrains.com/research/devecosystem-2018/

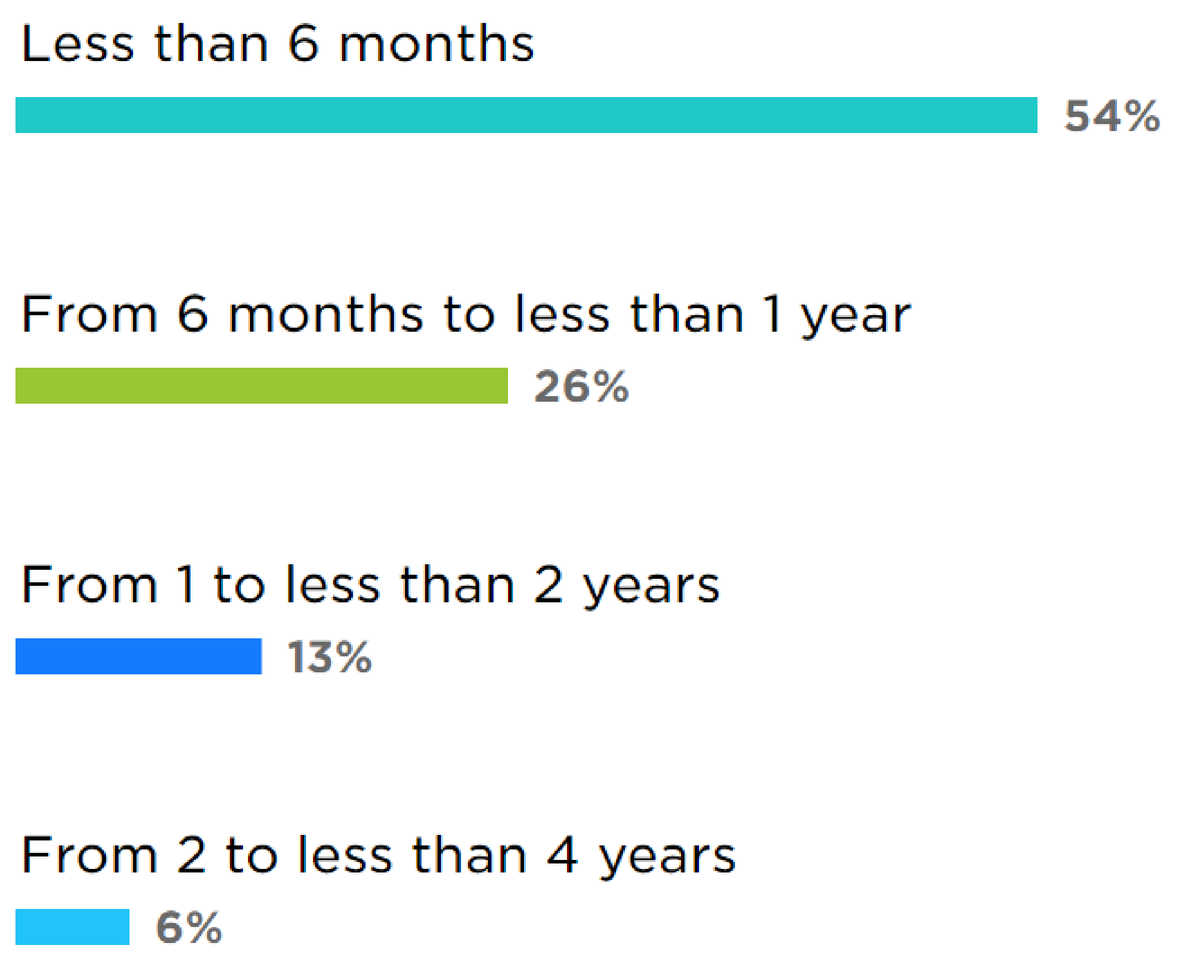
Programming languages



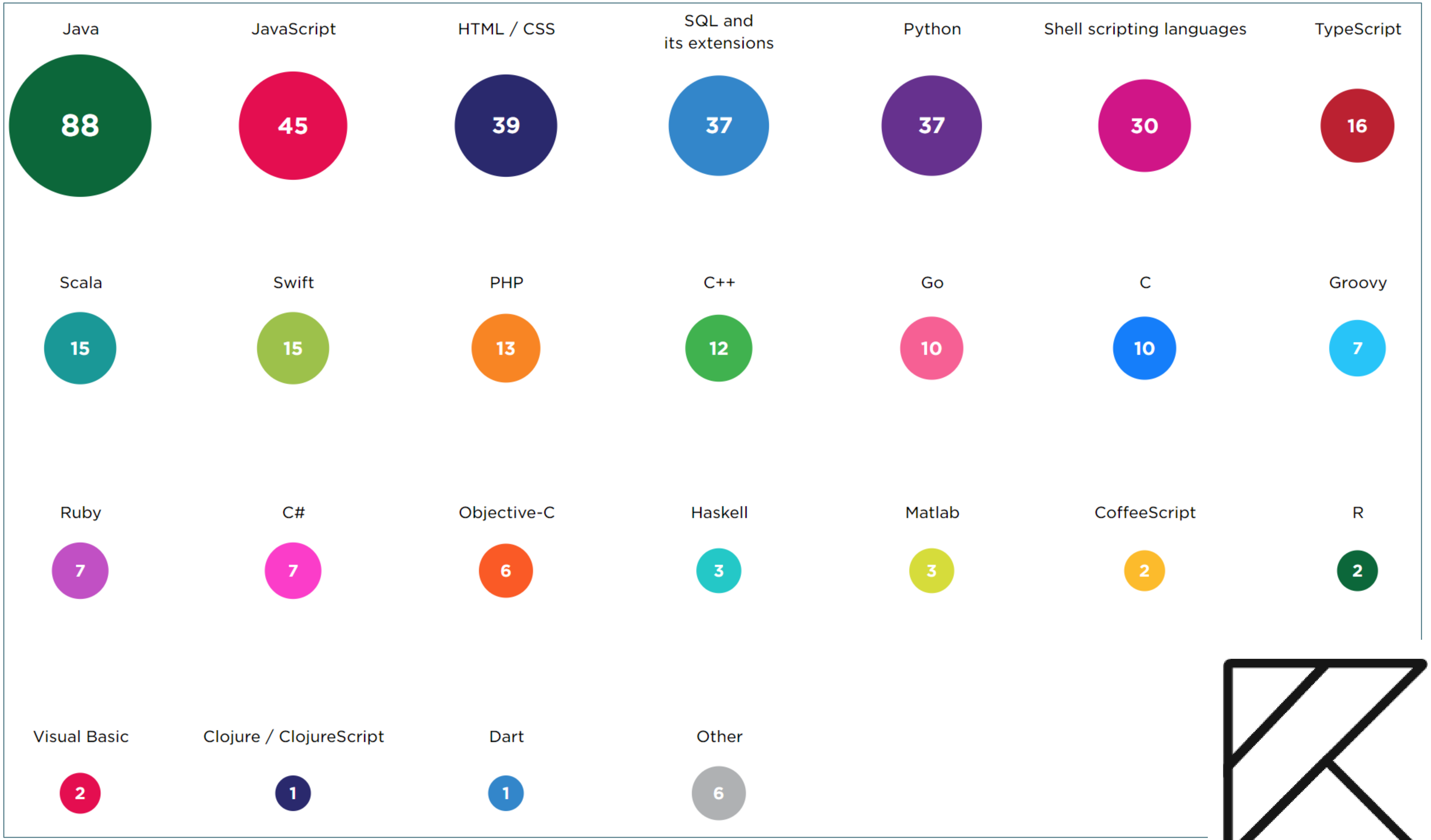
Which versions of Java do you regularly use?



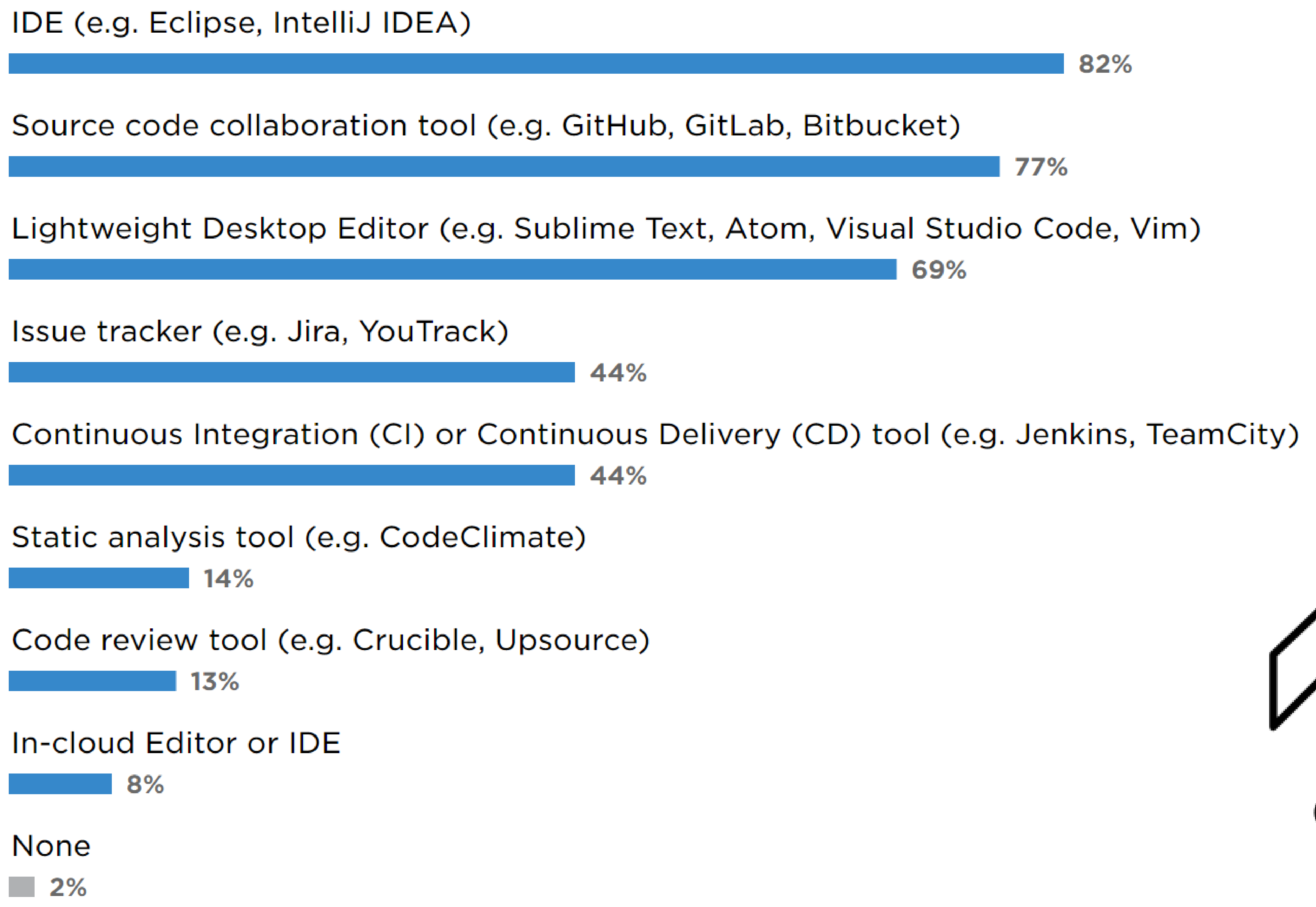
How long have you been using Kotlin?



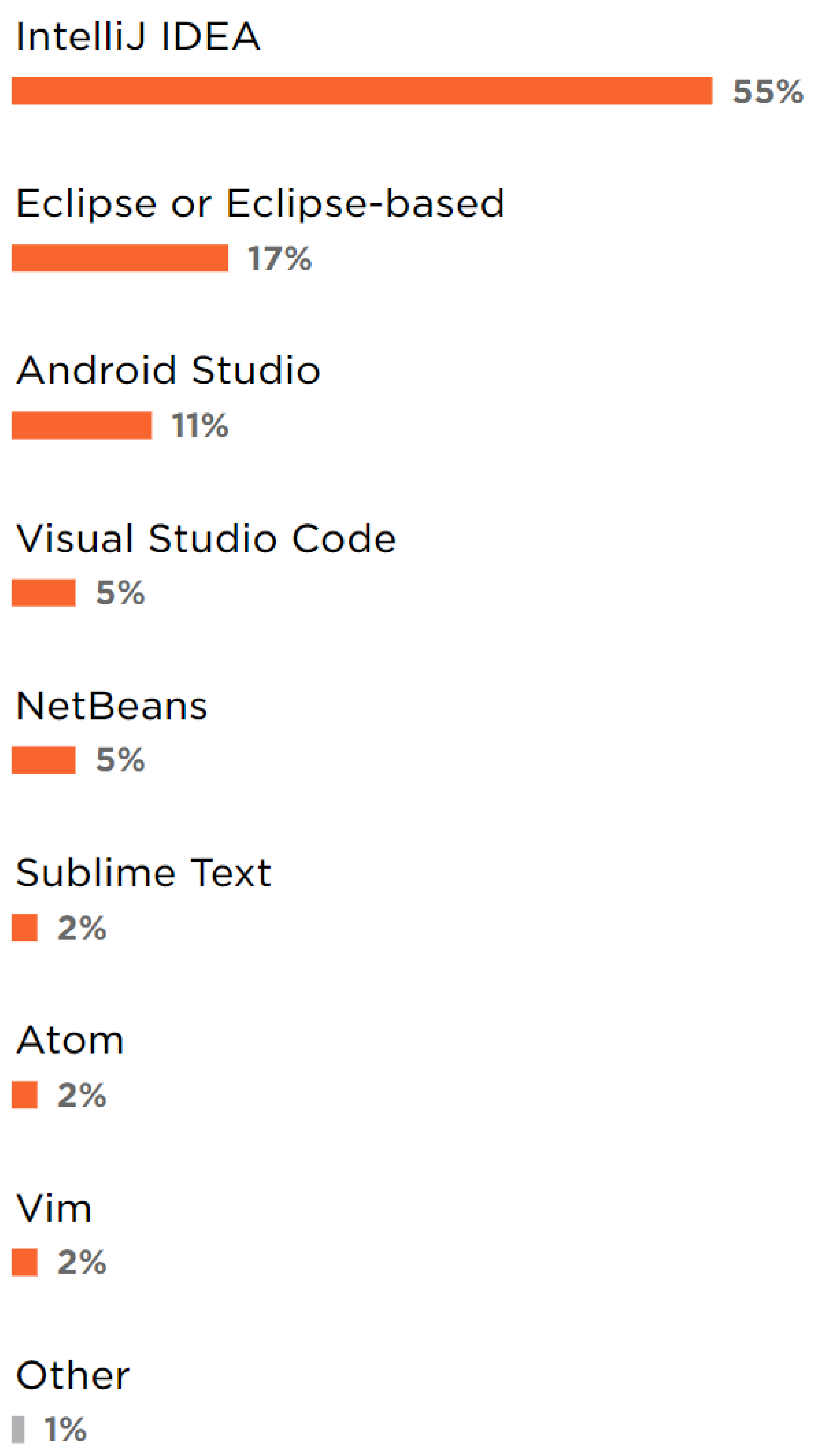
What other programming languages do Kotlin developers regularly use? (%)



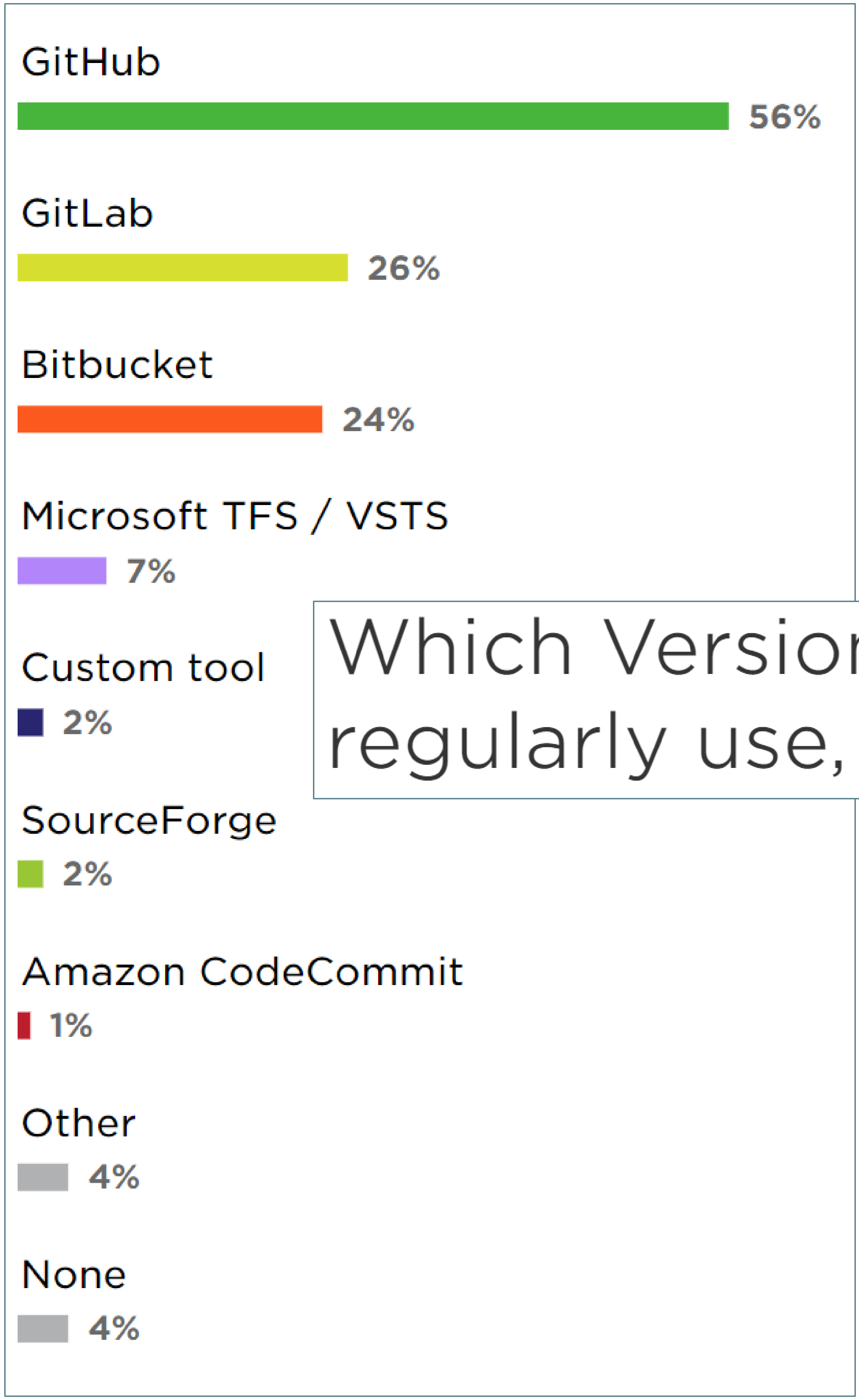
Which of the following tools do you regularly use?



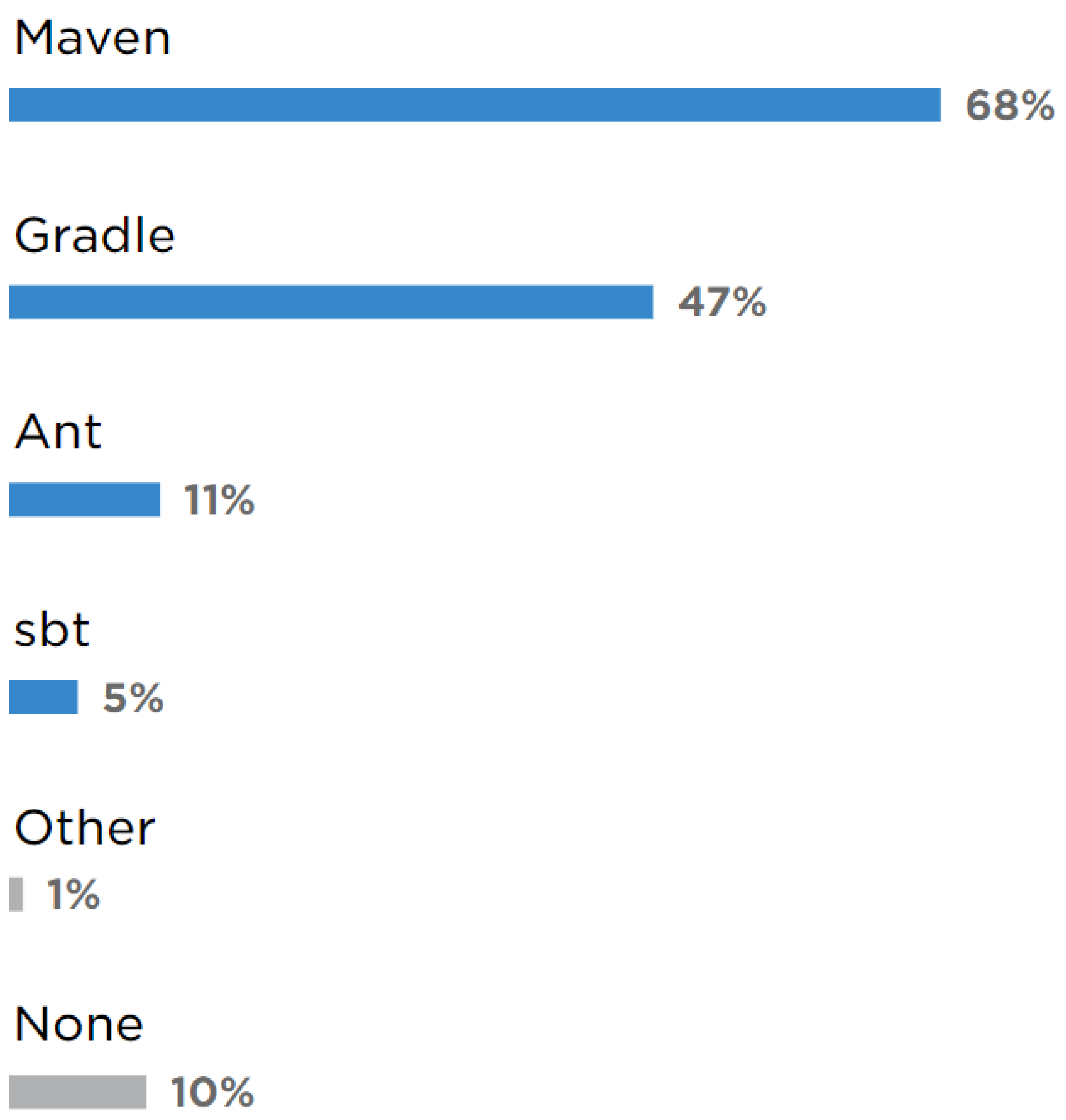
Which IDE / editor do you use the most?



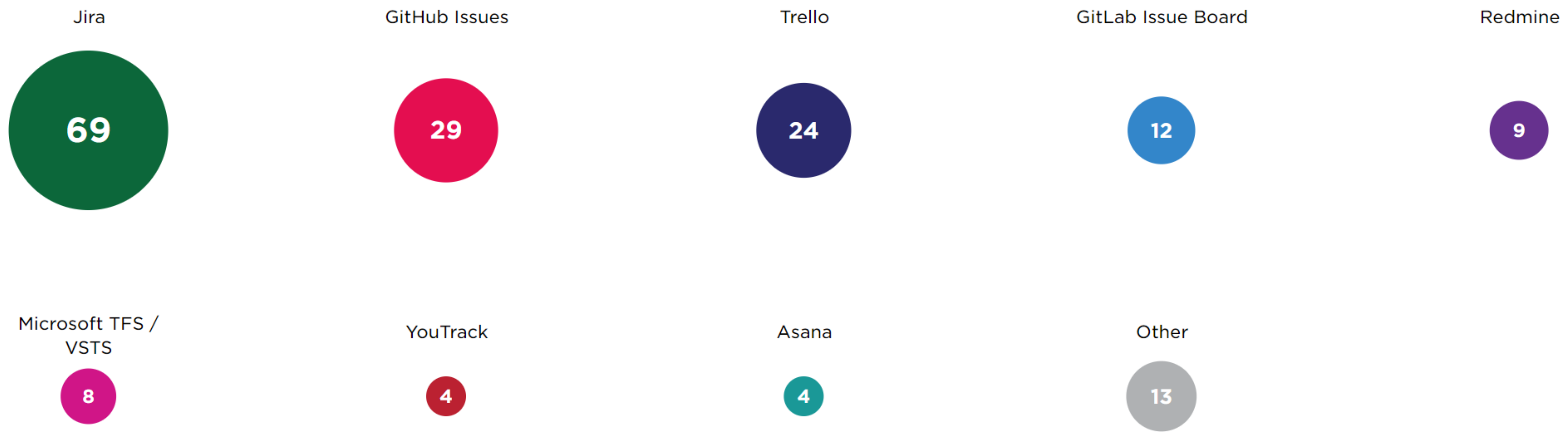
Which Version Control Services do you regularly use, if any?



Which build systems do you regularly use, if any?



Which Issue Tracking systems do you regularly use? (%)



Any questions?