Que es y como se come y pa que sirve, si quieren, al final de la presentacion esta toda la paja.

Escenario donde devops es usado.

LEGACY IT

CLOUD INNOVATION

DIGITAL BUSINESS

Devops y agile no son lo mismo.

- Agile and DevOps are not the same thing.
- **Treating them as the same thing can cause departments to abandon good and safe practices in the pursuit of something undesirable**.

- Agile software development is a methodology for developing software. Once the software is developed and released, the agile team doesn't formally care what happens to it. They're on to the next sprint and the next revision of the user story.

- DevOps, on the other hand, is all about taking software which is ready for release and deploying it in the safest, most reliable manner possible. DevOps doesn't depend on the software being developed by the agile discipline. It's entirely possible to have waterfall, kanban, scrum, agile, scrumban, development feeding DevOps.

Que es devops.

- Devops es la integracion de los equipos de desarrollo y operación con el objetivo de incrementar la productividad y colaboracion.
- Bajo ese esquema un solo grupo de ingenieros (developers, system admins, QA's. Testers etc) Tiene la responsabilida de inicio a fin de reunir los requieromientos de la aplicacion, servicio, plataforma o proyecto, para efectura el desarollo, testeo, deployment de infraestructura y finalmente el monitoreo y obtencion del feedback de los usuarios finales, para proceder de nuevo a repetir el ciclo de manera continua.

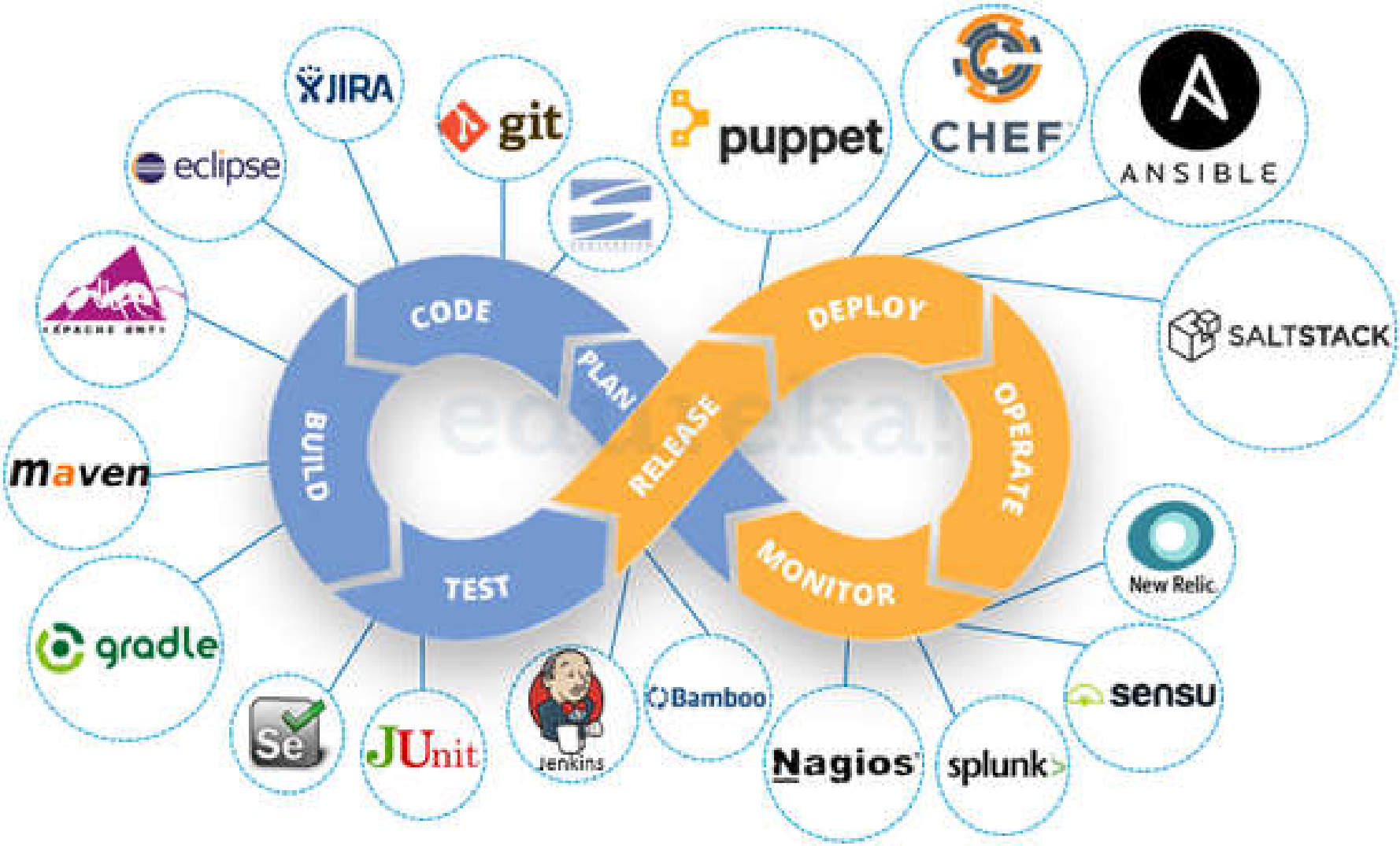
Como se logra

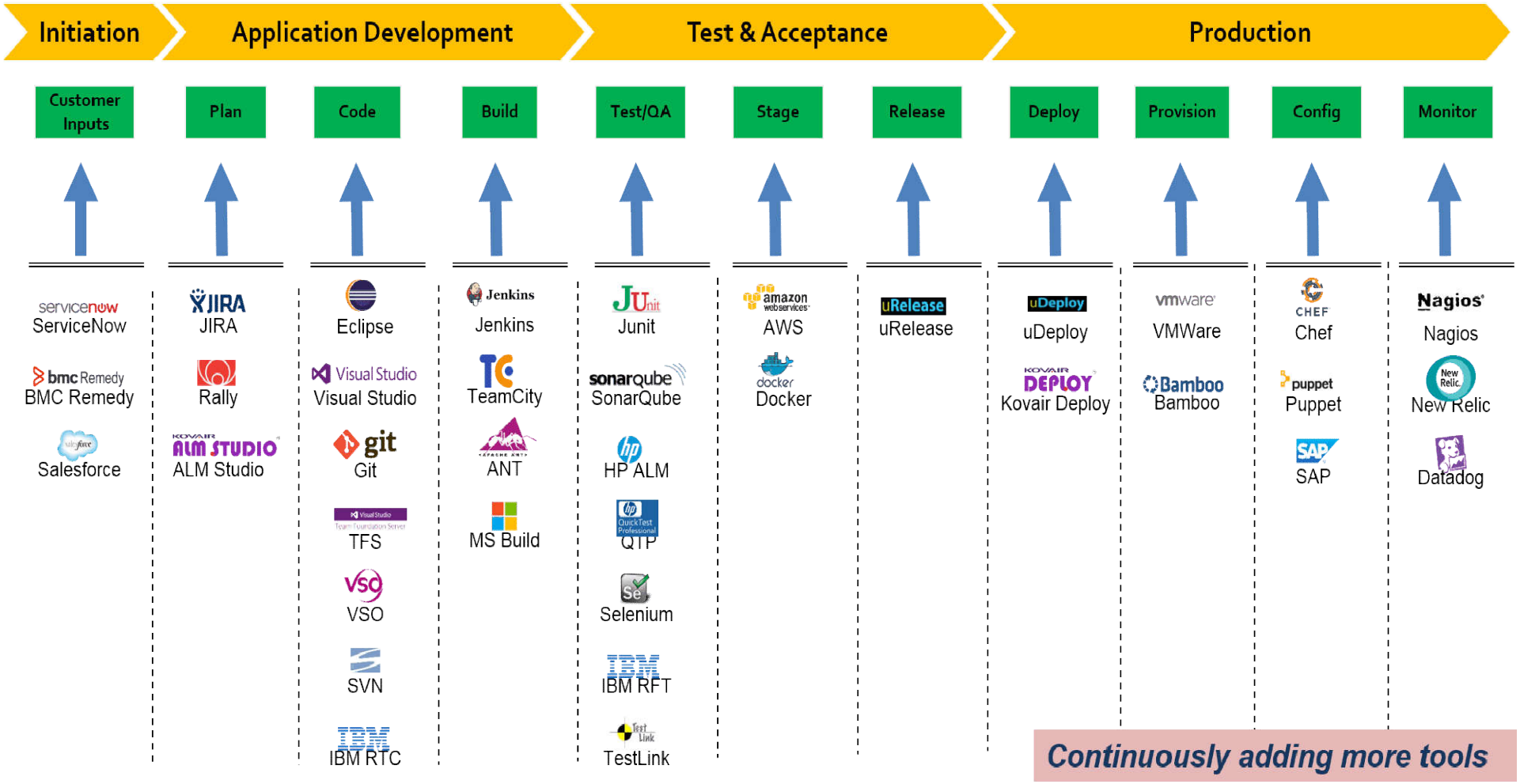
- Para hacer posible lo explicado en la presentacion anterior se necesita un cambio de mentalidad, dificil, pero en el area tecnica para actualizar a devops y podre hacer el proceso continuo se necesitan herramientas como: Puppet, Jenkins, GIT, Chef, Docker, Selenium, Aws, Azure y gcp,etc. El objetivo es logra la automatizacion en los procesos de Continuous Development, Continuous Integration, Continuous Testing, Continuous Deployment, Continuous Monitoring, reduciendo los errores humanos y obteniendo un mejor codigo, un mayor ahorro de recursos y una mejor administracion, y al hacer el ciclo mas rapido se obtiene una mejor experiencia del usuario.

Y como vamos a jalar con eso

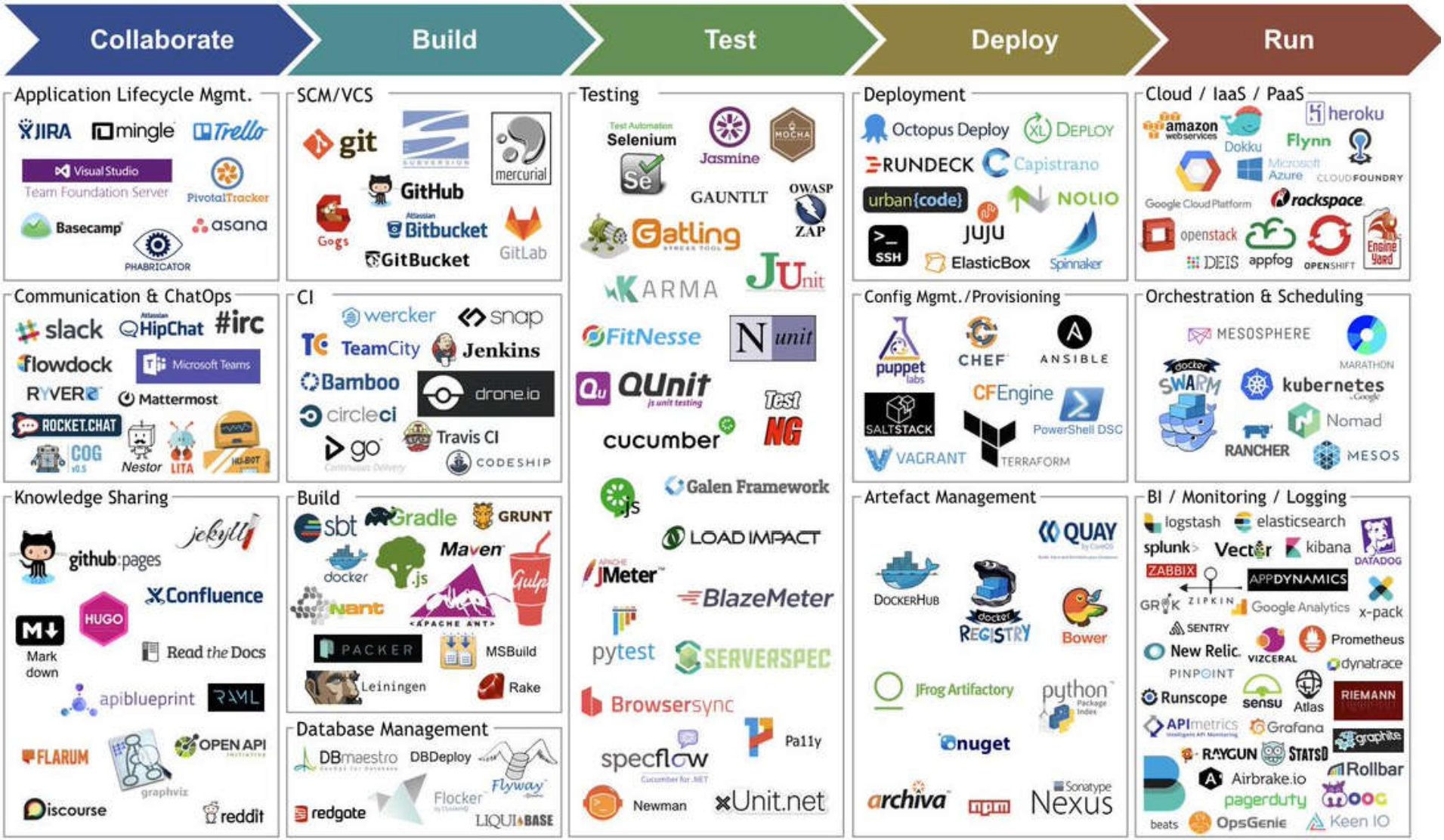
- Van a usar git, para hacer descargar y hacer commits de los archivos de notas, como si fueran versiones de codigo.
- https://github.com/jesusoctavioas/virtualizacion

- Van a producir maquinas virtuales, van a anotar las direcciones ip asignadas y las van a destruir.
- Van a usar los servicios de las nubes para hacer maquinas virtuales anotar sus ips y accesarlas asi como sus caracteristicas, las van a destruir y volver a crear.
- Se va a documentar las mejoras que puedan encontrar para hacer mas rapido el proceso.

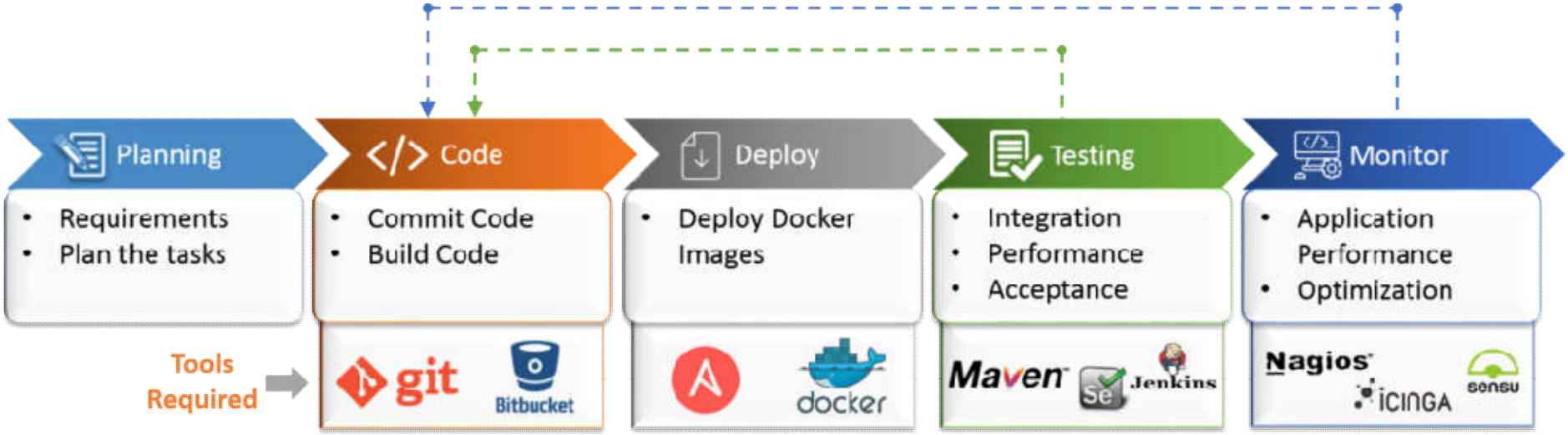

JIRA
git
eclipse
puppet
CHEF
ANSIBLE
SALTSTACK
CODE
PLAN
BUILD
TEST
RELEASE
DEPLOY
OPERATE
MONITOR
Maven
gradle
New Relic
sensu
Bamboo
JUnit
jenkins
Nagios
splunk


Initiation
Application Development
Test & Acceptance
Production
Customer Inputs
Plan
Code
Build
Test/QA
Stage
Release
Deploy
Provision
Config
Monitor
ServiceNow
BMC Remedy
Salesforce
JIRA
Rally
ALM Studio
Eclipse
Visual Studio
Git
TFS
VSO
SVN
IBM RTC
Jenkins
TeamCity
ANT
MS Build
Junit
SonarQube
HP ALM
QTP
Selenium
IBM RFT
TestLink
AWS
Docker
uRelease
uDeploy
Kovair Deploy
VMWare
Bamboo
Chef
Puppet
SAP
Nagios
New Relic
Datadog
Continuously adding more tools


DevOps
Nagios
VAGRANT
GRUNT
gradle
Clients
Xamarin
CORDOVA
Management
CHEF
puppet labs
ANSIBLE
SALTSTACK
mist.io
libcloud
SCALR
Applications
Joomla!
Drupal
PaaS & DevOps
apprenda
Jelastic
Java
Ruby
IntelliJ IDEA
eclipse
App Frameworks & Tools
php
nodeJS
Databases & Middleware
hadoop
redis
cleardb
cloudera
MySQL
mongoDB
Couchbase
Infrastructure
SUSE
bitnami
ORACLE LINUX
FreeBSD
docker


Collaborate
Build
Test
Deploy
Run
Application Lifecycle Mgmt.
JIRA
mingle
Trello
Visual Studio
Team Foundation Server
PivotalTracker
Basecamp
asana
PHABRICATOR
SCM/VCS
git
SUBVERSION
mercurial
GitHub
Bitbucket
Gogs
GitLab
GitBucket
Testing
Test Automation
Selenium
Jasmine
MOCHA
GAUNTLT
OWASP
ZAP
Gatling
KARMA
JUnit
FitNesse
Nunit
QUnit
js unit testing
TestNG
cucumber
.js
Galen Framework
LOAD IMPACT
JMeter
BlazeMeter
pytest
SERVERSPEC
Browsersync
specflow
Pa11y
Newman
Unit.net
Deployment
Octopus Deploy
DEPLOY
RUNDECK
Capistrano
urban{code}
NOLIO
JUJU
SSH
ElasticBox
Spinnaker
Cloud / IaaS / PaaS
amazon web services
Dokku
heroku
Flynn
Microsoft Azure
CLOUDFOUNDRY
Google Cloud Platform
rackspace
openstack
DEIS
appfog
OPENSHIFT
Engine Yard
Communication & ChatOps
slack
HipChat
#irc
flowdock
Microsoft Teams
RYVER
Mattermost
ROCKET.CHAT
COG
Nestor
LITA
CI
wercker
snap
TeamCity
Jenkins
Bamboo
drone.io
circleci
Travis CI
go
CODESHIP
Config Mgmt./Provisioning
puppet labs
CHEF
ANSIBLE
SALTSTACK
CFEngine
PowerShell DSC
VAGRANT
TERRAFORM
Orchestration & Scheduling
MESOSPHERE
MARATHON
docker SWARM
kubernetes
Nomad
RANCHER
MESOS
Knowledge Sharing
github pages
jekyll
Confluence
HUGO
Mark down
Read the Docs
apiblueprint
RAML
FLARUM
OPEN API
graphviz
Discourse
reddit
Build
sbt
Gradle
GRUNT
docker
.js
Maven
Gulp
nant
<APACHE ANT>
PACKER
MSBuild
Leiningen
Rake
Artefact Management
QUAY
DockerHub
docker REGISTRY
Bower
JFrog Artifactory
python Package Index
nuget
archiva
npm
Sonatype Nexus
BI / Monitoring / Logging
logstash
elasticsearch
splunk>
Vector
kibana
DATADOG
ZABBIX
APPDYNAMICS
GROK
ZIPKIN
Google Analytics
x-pack
SENTRY
New Relic.
VIZCERAL
Prometheus
dynatrace
PINPOINT
Runscope
sensu
Atlas
RIEMANN
APImetrics
Grafana
graphite
RAYGUN
STATSD
Airbrake.io
Rollbar
pagerduty
beats
OpsGenie
Keen IO
Database Management
DBmaestro
DBDeploy
Flyway
redgate
Flocker
LIQUIBASE

Tools for successful DevOps Adoption


Planning
Code
Deploy
Testing
Monitor
Requirements
Plan the tasks
Commit Code
Build Code
Deploy Docker Images
Integration
Performance
Acceptance
Application Performance
Optimization
Tools Required
git
Bitbucket
docker
Maven
Jenkins
Nagios
ICINGA
sensu

De aquí en adelante esta la paja reglamentaria de blabla devops y buzzword.

- El movimiento DevOps empezó a fusionarse entre el 2007 y el 2008, cuando las comunidades de operaciones de TI y desarrollo de software manifestaron claramente lo que veían como un nivel fatal de disfunción del sector.
- Se alzaron contra el modelo tradicional de desarrollo de software, que exigía que los que escribían el código se mantuvieran al margen, en términos de organización y operación, de los que desplegaban y mantenían tal código.
- Los desarrolladores y profesionales de TI/Operaciones tenían objetivos distintos (y, a menudo, rivales), direcciones de departamento independientes, indicadores clave del rendimiento diferentes por los que se les evaluaba y, con frecuencia, trabajaban en plantas separadas o, incluso, en edificios separados. El resultado eran equipos aislados únicamente preocupados por sus propios dominios, largas jornadas, publicaciones chapuceras y clientes insatisfechos.

- Tiene que haber un método mejor, dijeron. Así pues, las dos comunidades se juntaron y se pusieron a hablar, con gente como Patrick Dubois, Gene Kim y John Willis al frente de la conversación.
- Lo que empezó en foros de internet y reuniones locales es ahora uno de los aspectos principales del ámbito del software actual, y seguramente lo que te ha traído hasta aquí. Tú y tu equipo estáis sufriendo el dolor causado por equipos encasillados y líneas de comunicación rotas dentro de la empresa.
- Estás utilizando metodologías ágiles en la planificación y el desarrollo, pero te sigue costando sacar ese código sin montar un drama. Has oído hablar de DevOps y el efecto aparentemente mágico que puede causar en los equipos, y te has dicho: “Yo quiero un poco de esa magia”.

¿Y tú qué ganas?

- Normalmente, los equipos que trabajan en grupos aislados no se adhieren al 'pensamiento sistémico' de DevOps. El 'pensamiento sistémico' consiste en ser consciente de que tus acciones no solo afectan a tu equipo, sino a todos los equipos involucrados en el proceso de publicación. La falta de visibilidad y de objetivos compartidos se traduce en falta de planificación de dependencias, prioridades mal organizadas, acusaciones personales y una actitud de 'no es problema mío', provocando una disminución de la velocidad y la calidad. DevOps es ese cambio de mentalidad con el que se tiene una visión holística del proceso de desarrollo y se rompe la barrera entre Dev y Ops.
- **Publicaciones más rápidas y una forma de trabajar más inteligente**
- **Acelerar el tiempo de resolución**
- **Mejor gestión del trabajo imprevisto**
- **Automatización**
- **Cultura**
- **Medición**
- **Compartir**

TOP 10 BARRIERS TO DevOps*

"...and best practices to overcome them for a successful rollout in 2017".

1 CULTURE

Focus on building a collaborative culture with shared goals. Elect champions.

2 TEST AUTOMATION

Don't neglect test automation while focusing on CI/CD deployments. Continuous Testing is hugely important. Security should not be an afterthought.

3 LEGACY

Include modeling for legacy infrastructure and applications. Brownfield deployments are always tougher

4 APP COMPLEXITY

Factor in application architecture changes based on on-prem, cloud and containers early on.

5 NO DEVOPS PLAN

Have a clear plan with milestones, owners and well-defined deliverables

6 MANAGING ENVIRONMENTS

Standardize and automate complex DevOps environments with cloud sandboxes

7 SKILLSET

Train your teams, Standardize processes and establish common operational procedures.

8 BUDGET

Don't assume open source means free. Factor in integration and operational complexity.

9 TOOLS

Avoid fragmented toolset adoption. Tool chain sprawl can impose operational overhead.

10 EXECUTIVE SUPPORT

Take time to educate executive chain to get resource and budget alignment

* Based on 2045 responses to Quali's annual DevOps survey in 2016.

Quali
www.quali.com