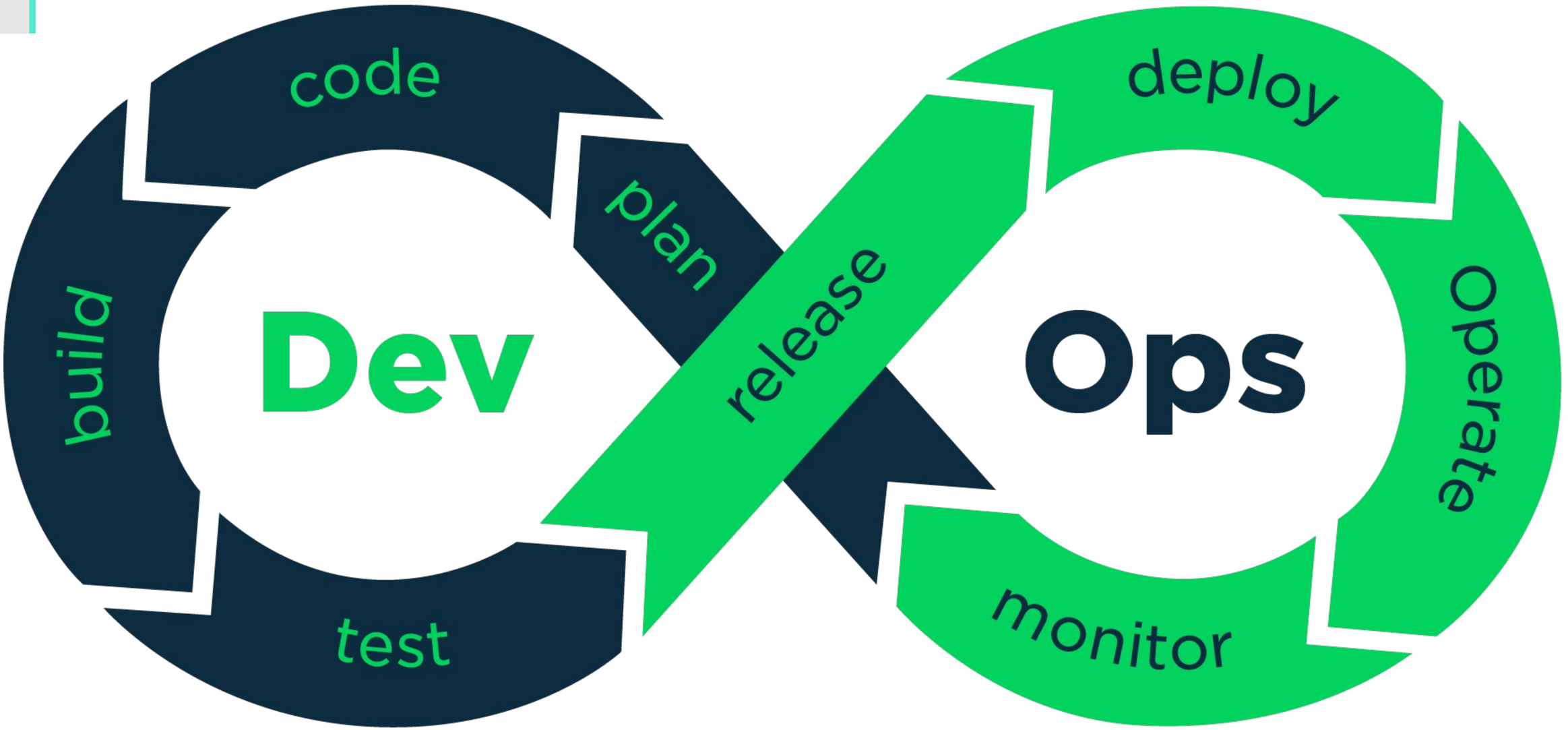
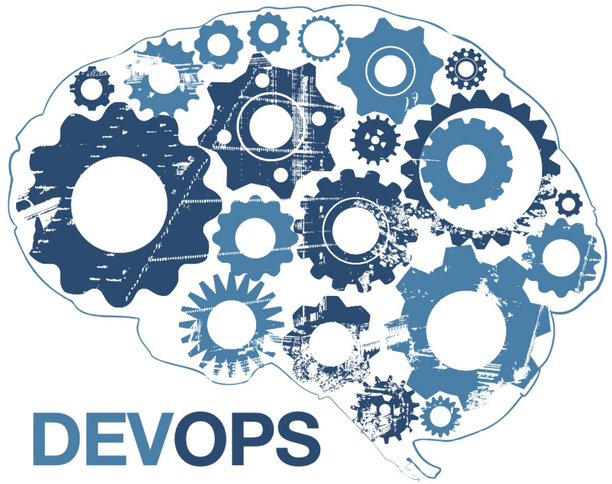

code
plan
build
Dev
test
release
deploy
Ops
operate
monitor



INTERMEDIO

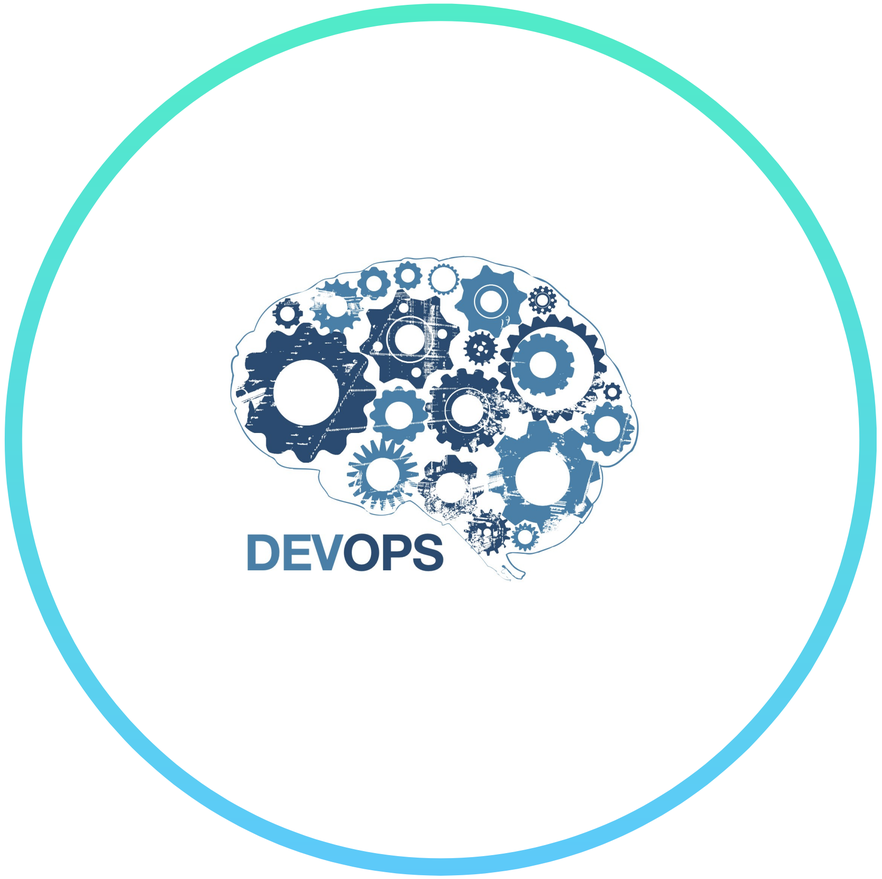
Unidad 01

Introducción

CURSOS DE DESARROLLO

Objetivos

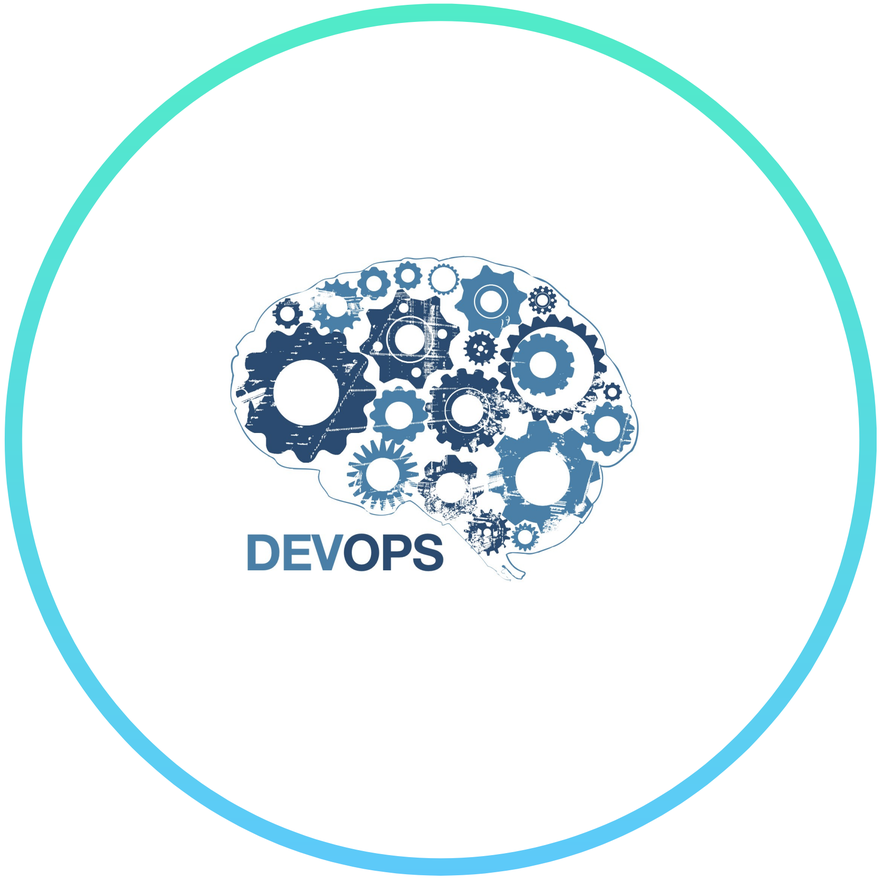
¿Qué voy a a aprender?



- Conocer los procesos y herramientas que conforman la Integración Continua (CI)
- Saber identificar las tareas y perfiles que intervienen en CI
- Saber definir los objetivos y fases en CI
- Conocer los principios aplicables a la calidad del software
- Saber aplicar políticas de seguridad en entornos CI

Contenidos

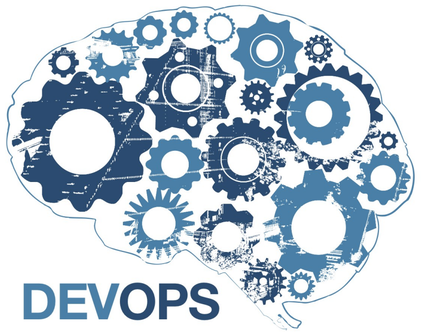
¿Cómo voy a aprenderlo?



1. Introducción
2. Conceptos
3. Objetivos
4. Principios
5. Herramientas
6. Versionado
7. Calidad del Software
8. Conclusiones
9. Referencias

Introducción

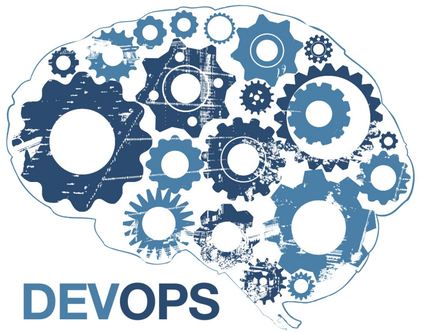
¿Por dónde empezamos?



En esta unidad veremos los fundamentos de la Integración continua, la definición de los conceptos, los objetivos que pretendemos cumplir, las herramientas a utilizar y los criterios de calidad de software que queremos cubrir

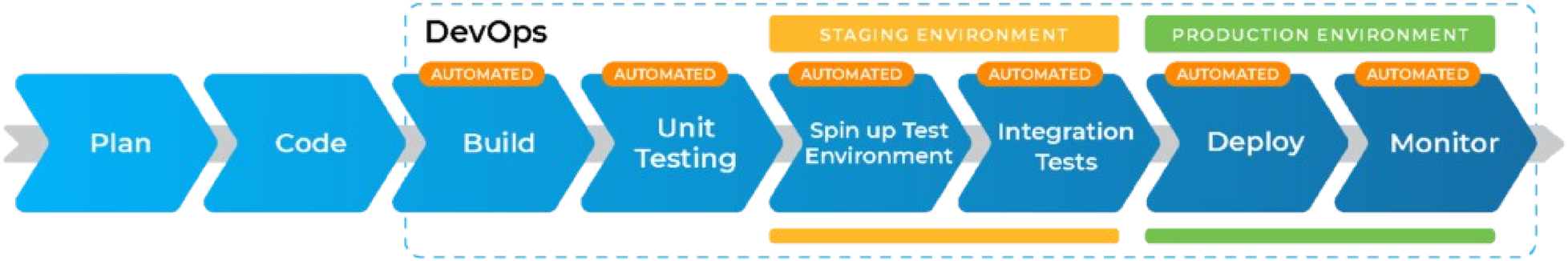
Introducción

¿Por dónde empezamos?



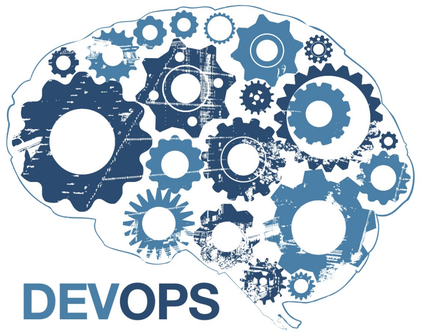
DevOps es una cultura que se enfoca a la colaboración y comunicación entre los desarrolladores y el resto de profesionales TIC, mientras que automatizan las entregas del software y los cambios en la infraestructura

Introducción



Introducción

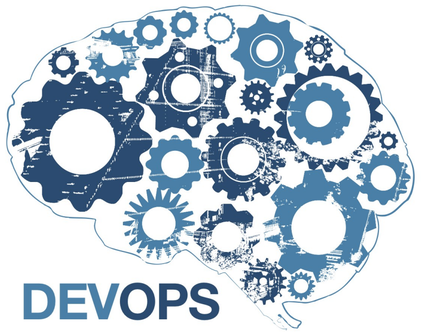
¿Por dónde empezamos?



Últimamente a este tipo especial de ingeniero que conoce estas tecnologías se le denomina Site Realiability Engineering (SRE)

Introducción

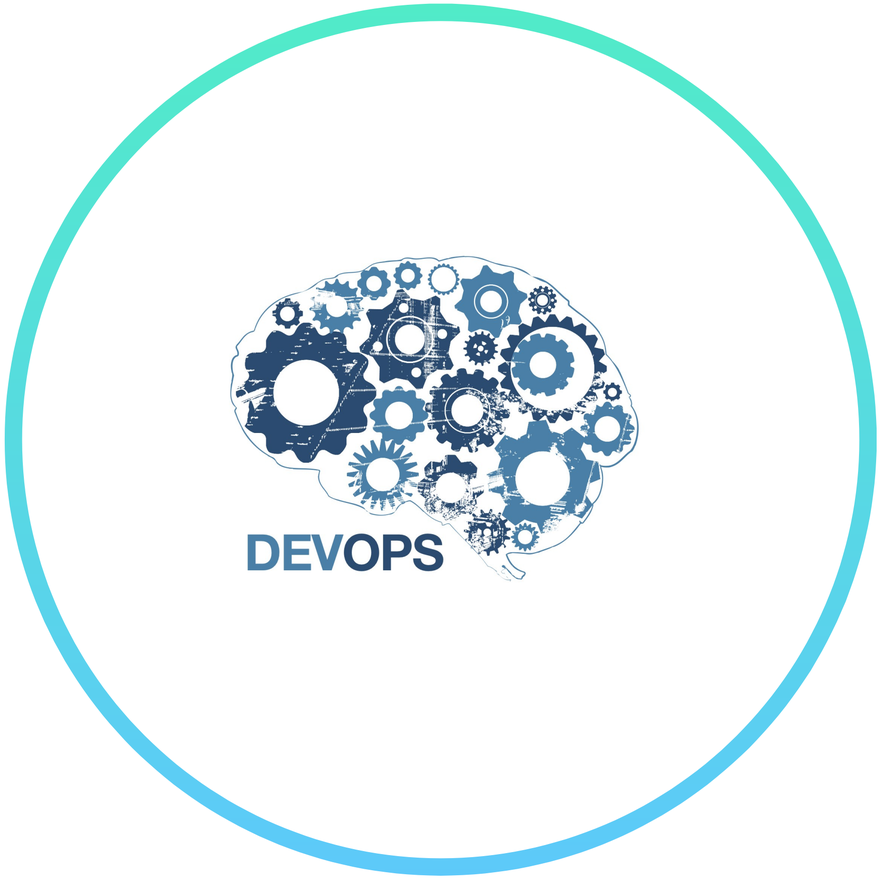
¿Por dónde empezamos?



El Ingeniero de Confiabilidad de Sitios (SRE) sería un ingeniero de software que aplica sus conocimientos a resolver problemas de infraestructura y operaciones

Introducción

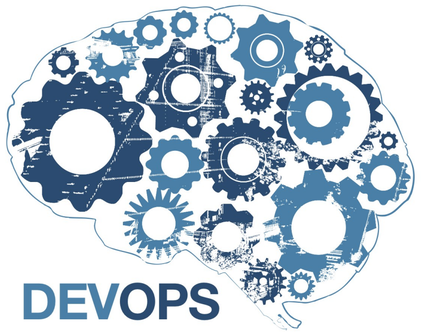
¿Por dónde empezamos?



Si el ingeniero fullstack es el que es capaz de hacer tanto las tareas de backend y frontend, el SRE es el que es capaz de hacer lo propio en desarrollo y operaciones

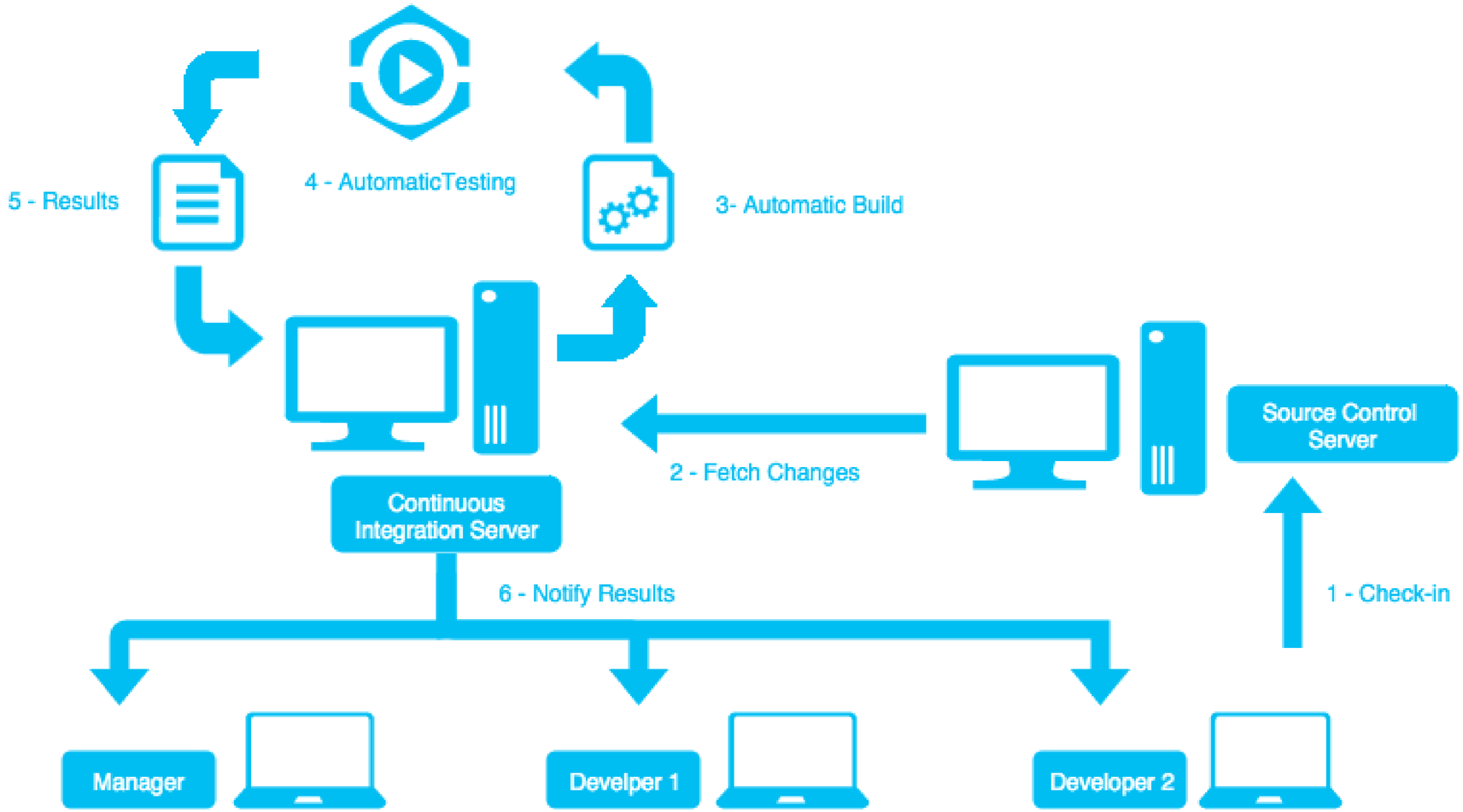
Conceptos

Integración Continua



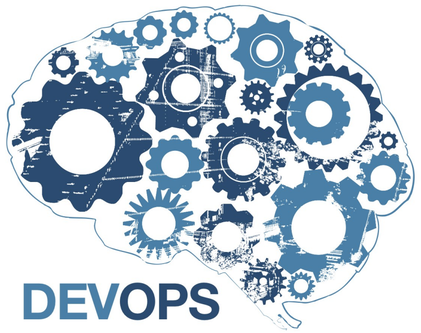
Martin Fowler fue el primero en proponer este concepto como un medio para conseguir detectar los fallos lo antes posible de un programa

Conceptos


5 - Results
4 - AutomaticTesting
3- Automatic Build
2 - Fetch Changes
Source Control Server
Continuous Integration Server
1 - Check-in
6 - Notify Results
Manager
Develper 1
Developer 2

Conceptos

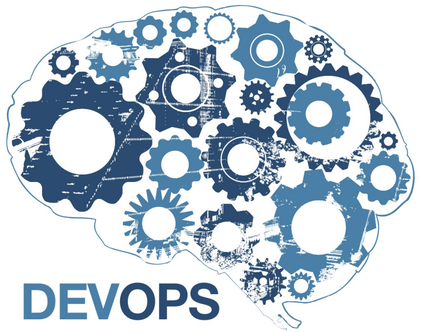
Integración Continua



La manera de detectarlos es enfocando la compilación y a la ejecución de pruebas sobre el proyecto

Conceptos

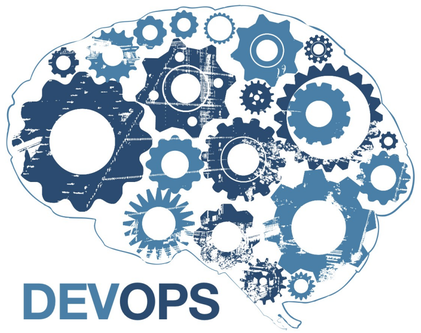
Integración Continua



La compilación del proyecto se realizará en base a un repositorio de código con una determinada periodicidad

Conceptos

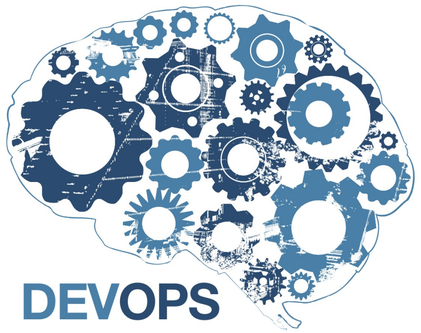
Integración Continua



Debe de influir en la calidad del código, para que siga los criterios definidos por la ingeniería del software

Conceptos

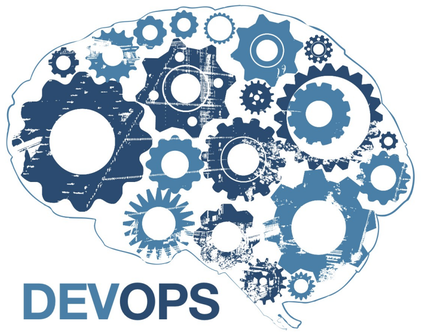
Integración Continua



Debería reducir las tareas repetitivas y manuales para mejorar la estructura de costes del proyecto, por lo que se deberían automatizar todas aquellas tareas que se pueda

Conceptos

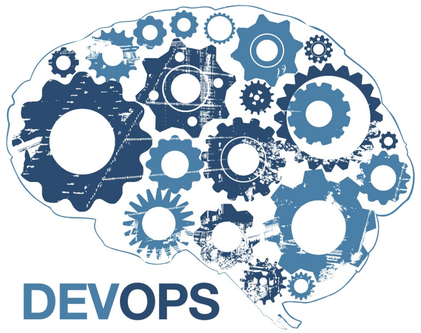
Integración Continua



Permite realizar el lanzamiento de las pruebas del proyecto desde todas las perspectivas posibles: desarrollo, calidad, seguridad, sistemas

Conceptos

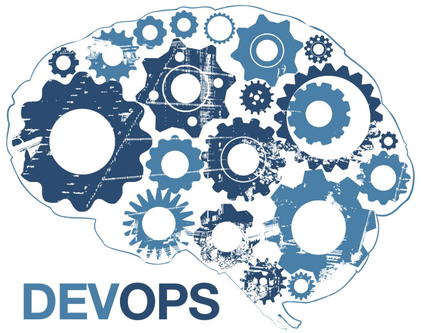
Integración Continua



Permite visibilizar mejor el desarrollo del proyecto, al emitir informes y gráficas

Conceptos

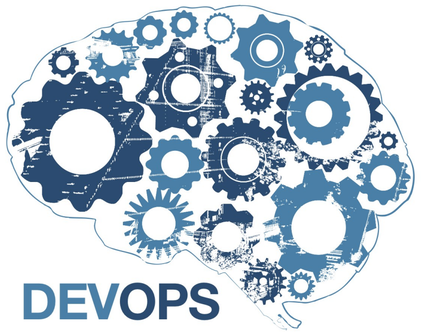
Integración Continua



Debería mejorar la confianza y seguridad del equipo de trabajo, a la hora de fijar que los cambios que se realizan no cambian funcionalidad previa que lo hacía de manera correcta

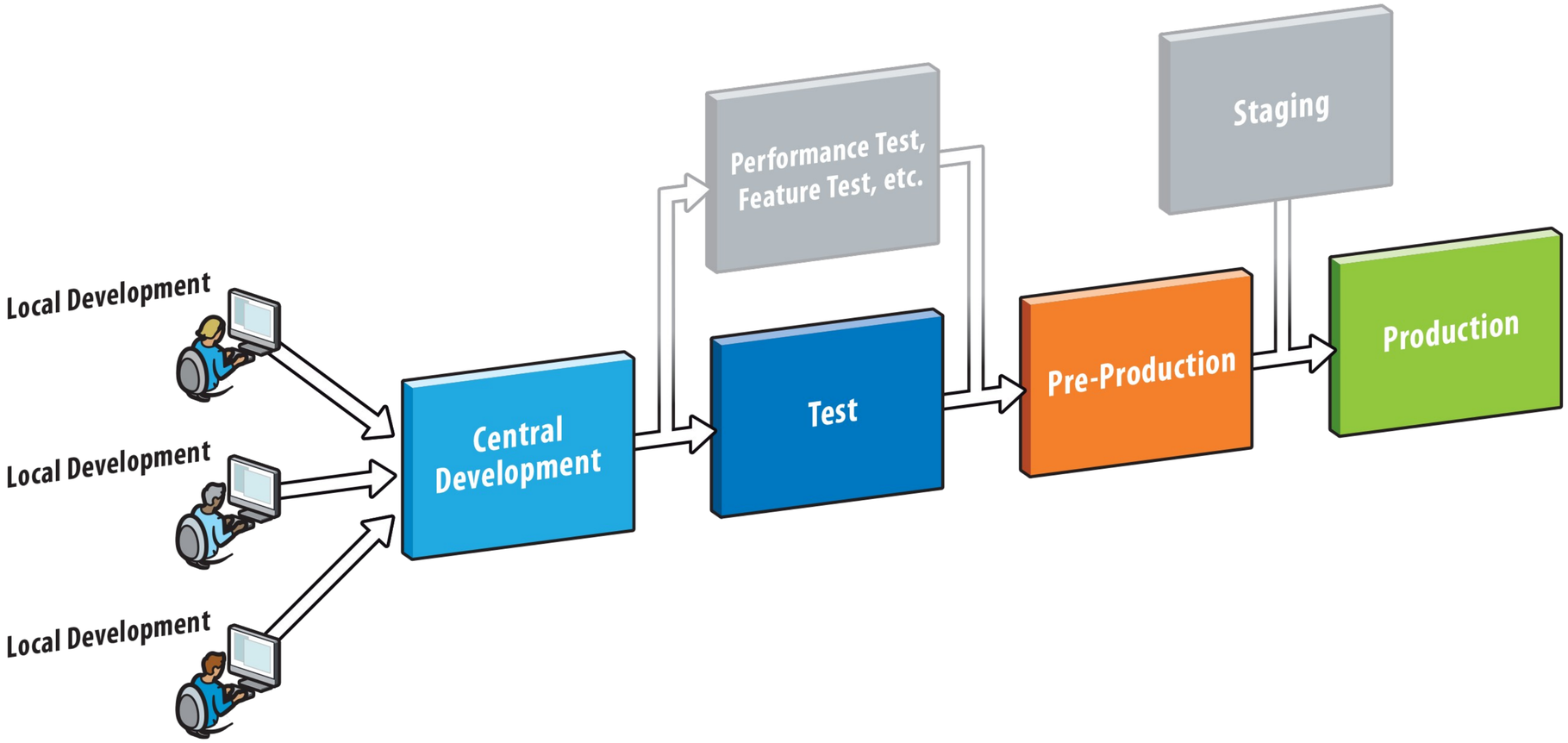
Conceptos

Entornos



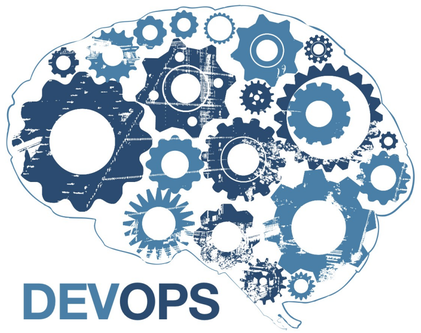
Los entornos son aquellas fases por las que pasa un proyecto hasta su puesta en producción

Conceptos


Local Development
Local Development
Local Development
Central Development
Performance Test, Feature Test, etc.
Test
Staging
Pre-Production
Production

Conceptos

Entornos



Si bien depende de cada empresa, el tipo de proyecto y de cliente se pueden establecer una serie de fases que puede tener un proyecto, coinciden con las indicadas en la gestión de proyectos

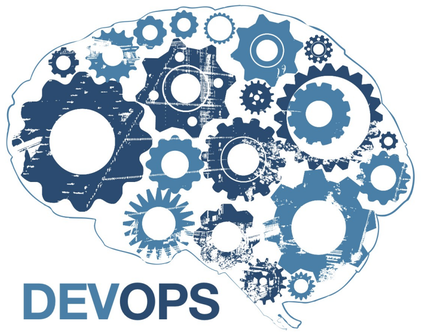
Conceptos

Entornos

- Definición de Requisitos
- Análisis
- Desarrollo
- Pruebas
- Entrega
- Mantenimiento

Conceptos

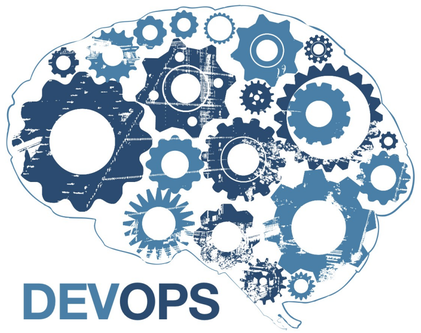
Entornos



Estas fases suelen ser gestionadas de manera separada por distintas herramientas y equipos en empresas grandes y por una sola persona en proyectos más pequeños

Conceptos

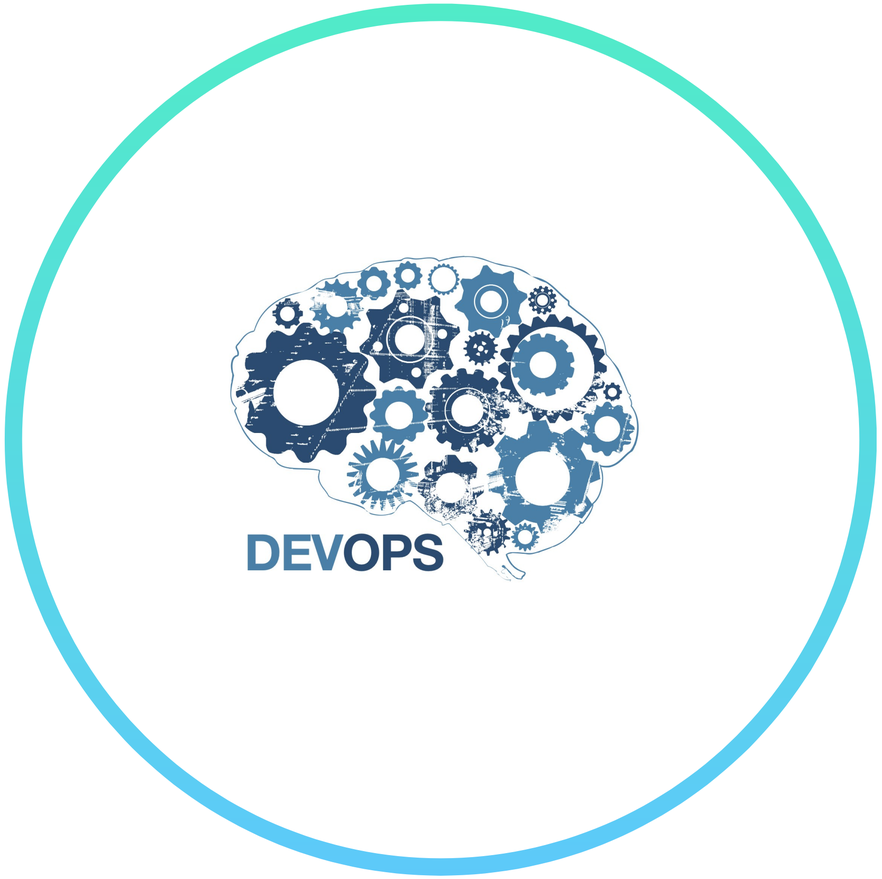
Entornos



Pero también se pueden definir una serie de entornos que permiten la gestión del proyecto desde su desarrollo

Conceptos

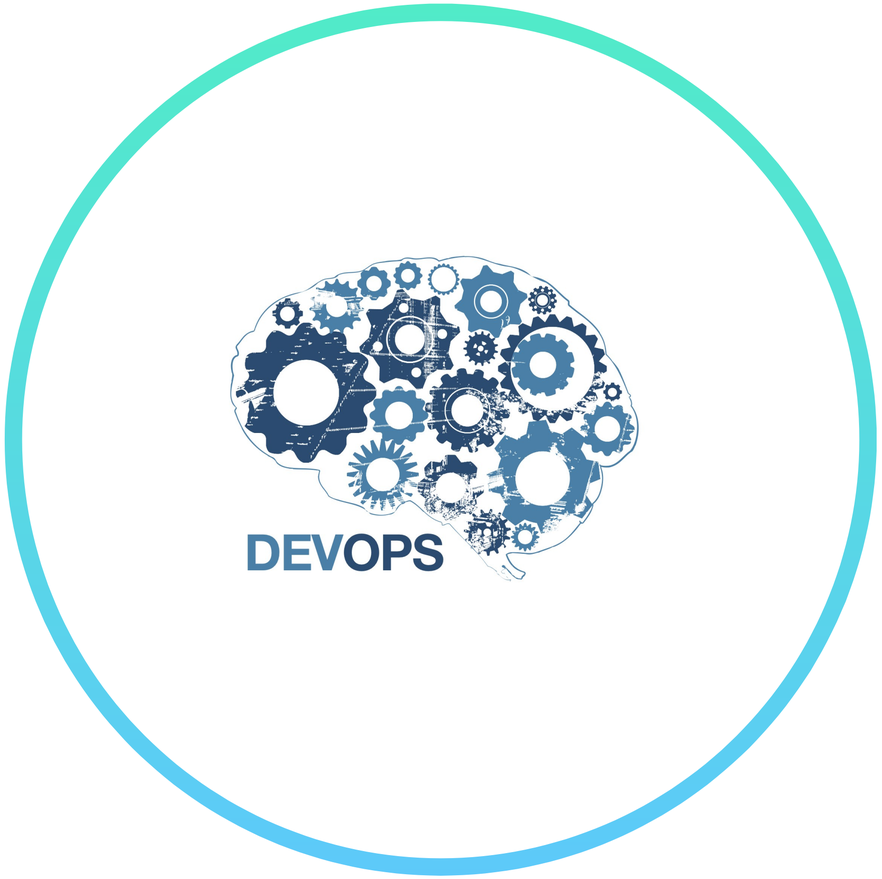
Entornos



- Desarrollo: realizado por cada desarrollador individual
- Integración: Unificación del trabajo de todos los desarrolladores
- Pruebas: Entorno en el que se realizan todas las pruebas necesarias
- Preproducción: Entorno lo más similar a producción, donde se simula su puesta en marcha
- Producción: Entorno final de puesta en marcha de la solución

Conceptos

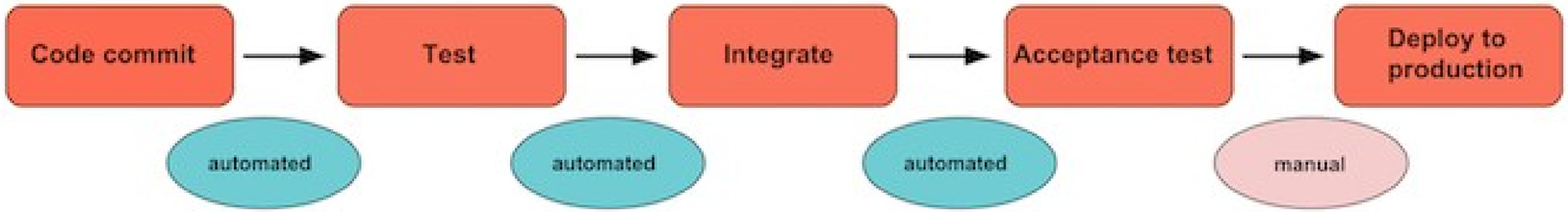
Entrega Continua



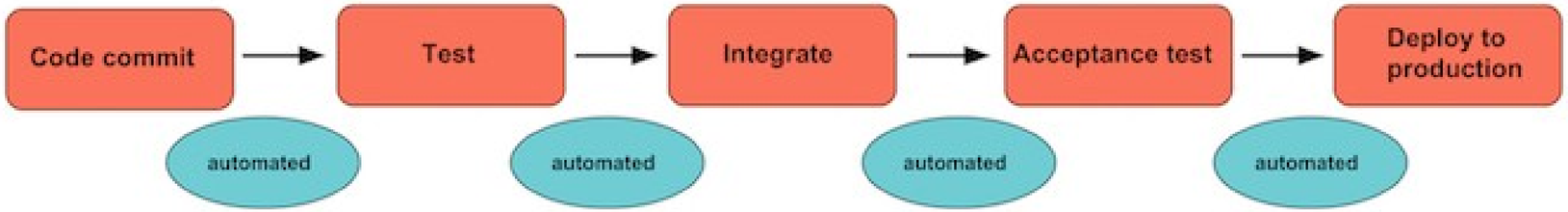
La entrega continua es la manera de afrontar un proyecto con ciclos cortos, de la manera más confiable posible

Conceptos

Continuous delivery

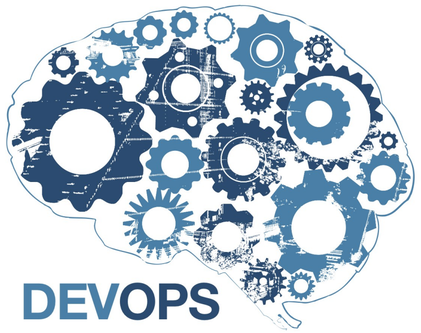


Continuous deployment



Conceptos

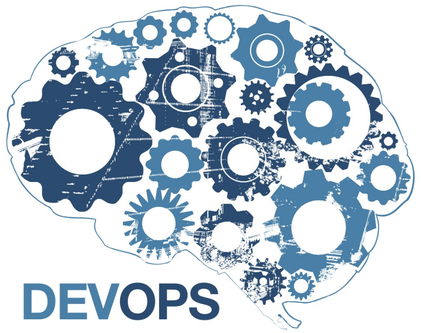
Entrega Continua



Este concepto incluye lo definido en la integración continua, pero incluye otras cuestiones importante que tengan que ver con el despliegue

Conceptos

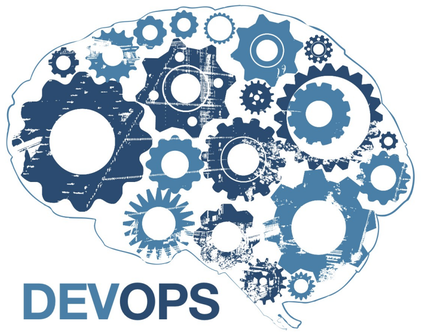
Entrega Continua



La idea principal es la de realizar un despliegue de la manera más rápida, replicable y confiable posible

Conceptos

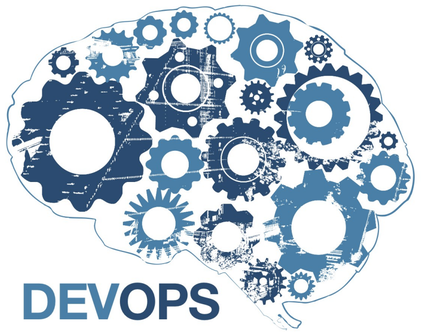
Entrega Continua



Para realizar esta fase en necesario la automatización de las tareas de puesta en producción de la aplicación

Conceptos

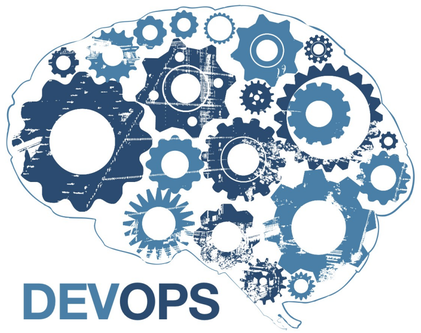
Entrega Continua



La conversión de los manuales de instalación a herramientas de creación de instalaciones automatizadas ha permitido el desarrollo de este concepto

Conceptos

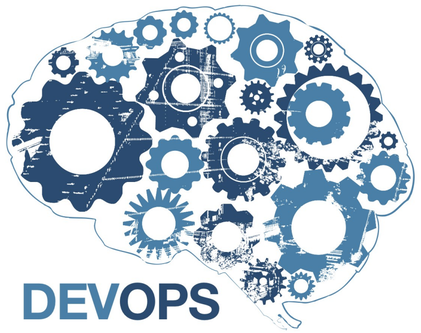
Entrega Continua



Estas herramientas han avanzado desde los scripts de instalación, a las herramientas independientes del sistema operativo o distribución, a la creación de contenedores

Conceptos

Entrega Continua

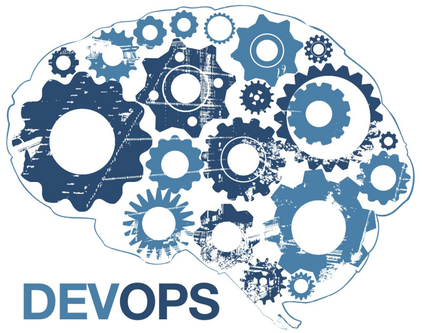


Las fases básicas son:

- Integración continua
- Paso a preproducción
- Pruebas en preproducción
- Paso a producción
- Pruebas en Producción

Conceptos

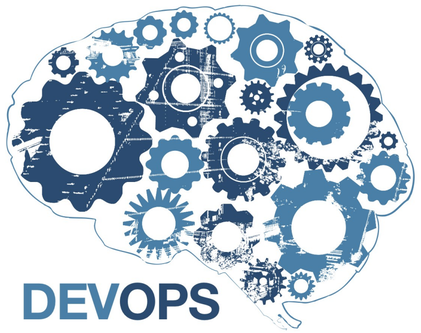
Despliegue Continuo



Este concepto es la automatización completa de entrega continua

Conceptos

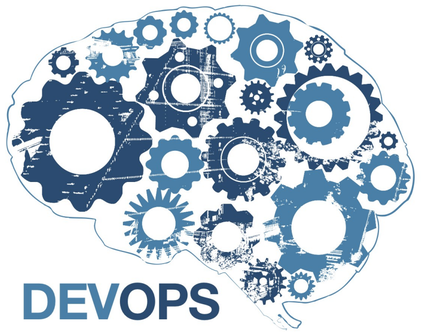
Despliegue Continuo



En este caso la subida a producción y preproducción se realizará de manera automática una vez que se han pasado las pruebas de una manera satisfactoria

Conceptos

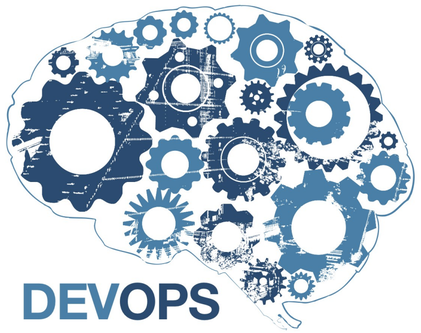
Despliegue Continuo



El salto de confianza de la entrega continua al despliegue continuo es muy alto y hay organizaciones que siguen prefiriendo validar manualmente el paso a producción y preproducción

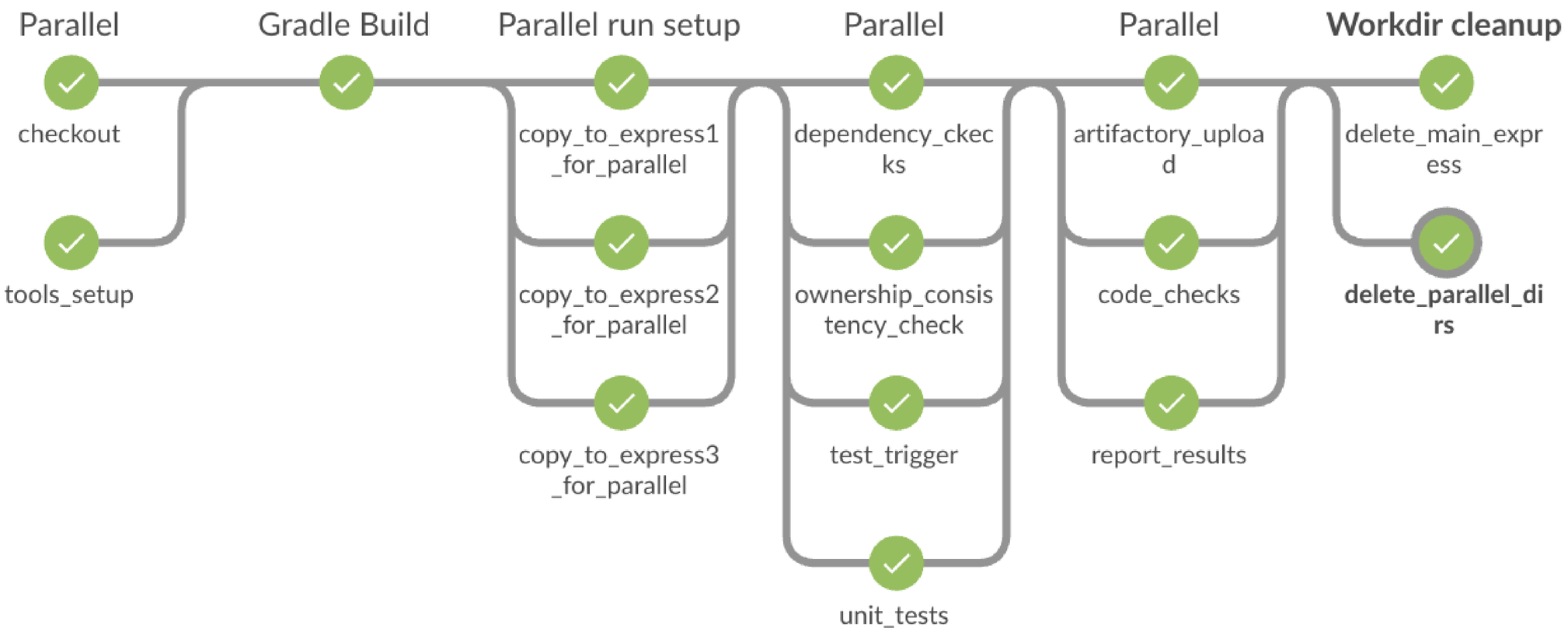
Conceptos

Pipelines



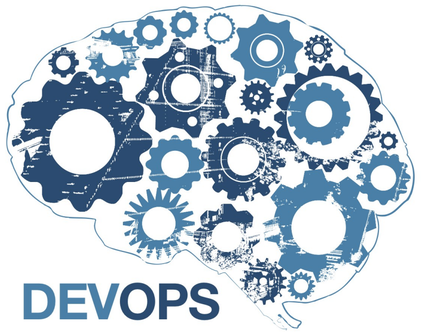
Las pipeline son los flujos de tareas y responsables asociados a un proceso de integración, entrega o despliegue continuos

Conceptos


Parallel
Gradle Build
Parallel run setup
Parallel
Parallel
Workdir cleanup
checkout
tools_setup
copy_to_express1_for_parallel
copy_to_express2_for_parallel
copy_to_express3_for_parallel
dependency_ckecks
ownership_consistency_check
test_trigger
unit_tests
artifactory_upload
code_checks
report_results
delete_main_express
delete_parallel_dirs

Conceptos

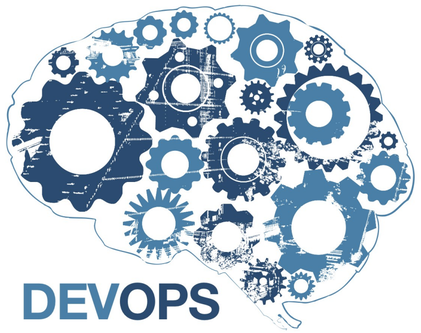
Pipelines



En estas pipelines se definen las fases que deberían incluirse en el proceso controlado, así como todas las tareas y la manera de verificar cada una de ellas

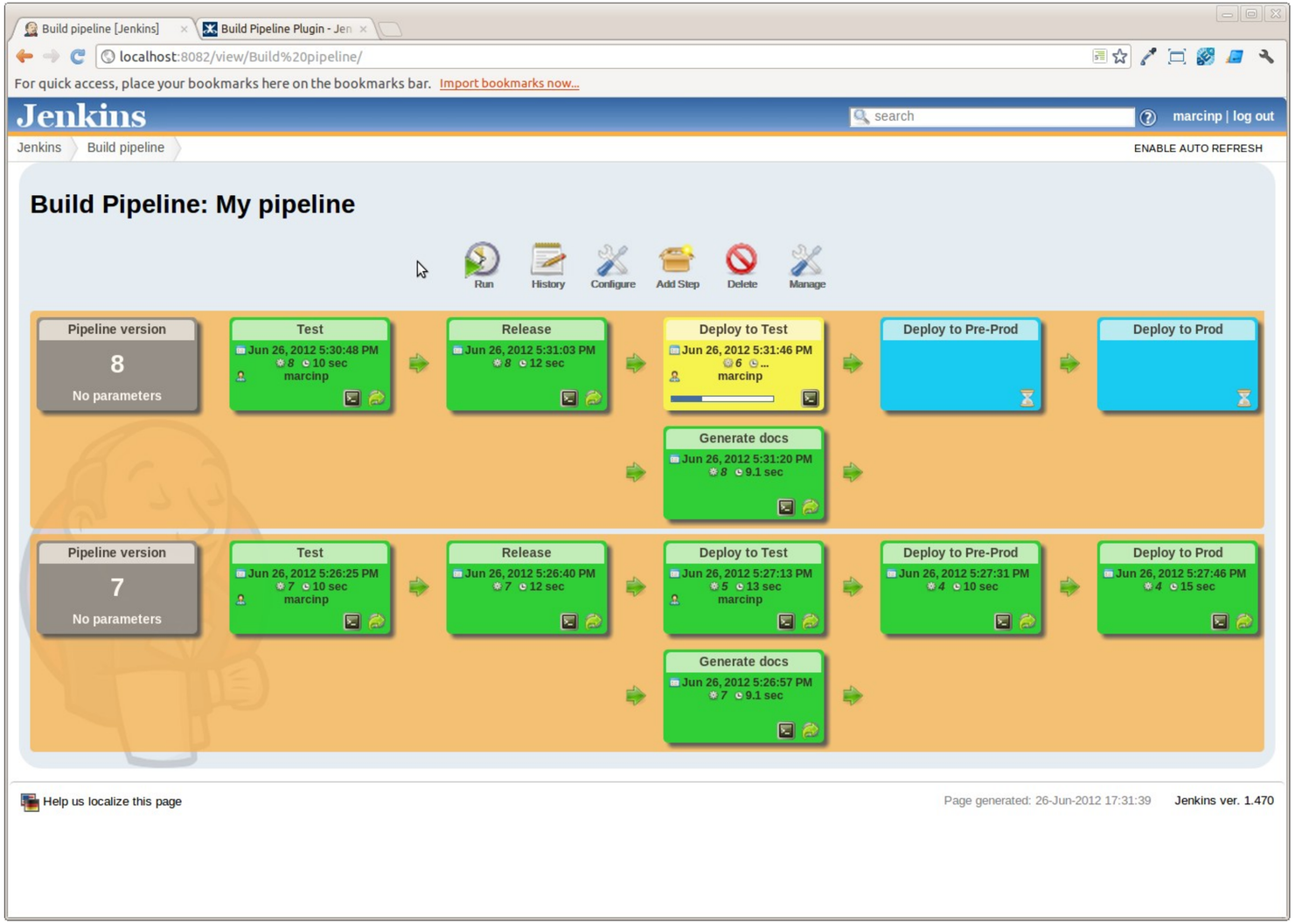
Conceptos

Pipelines



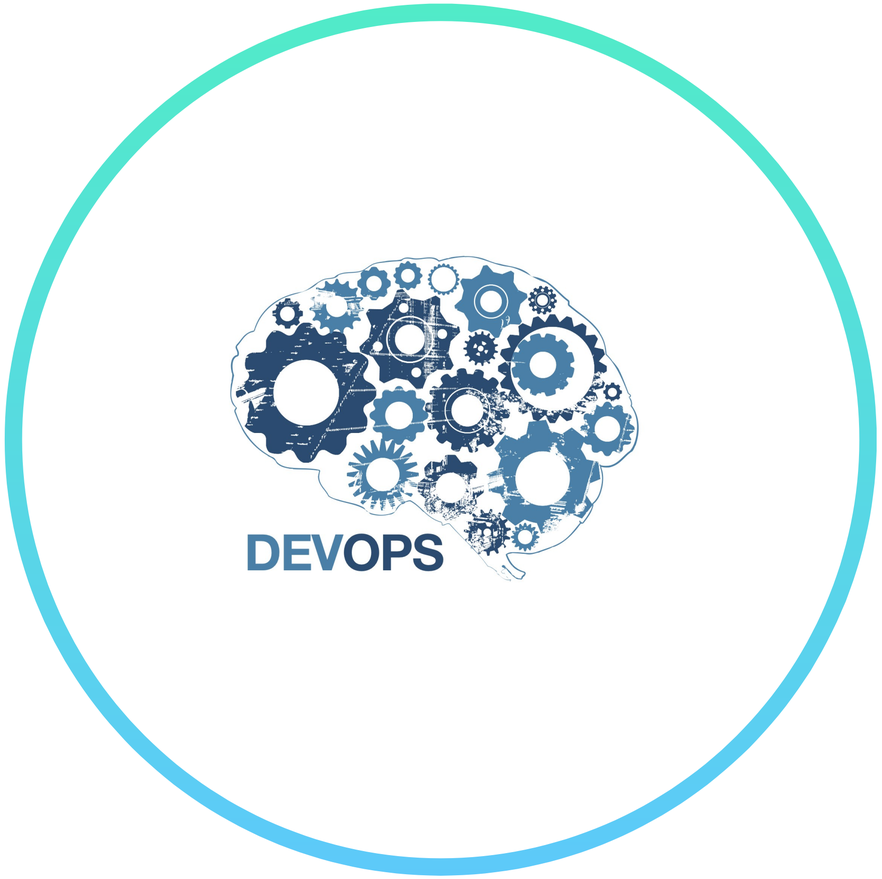
Por supuesto cada una de estas fases pueden tener una o varias personas responsables que puedan tomar la decisión, de manera automatizada o manual, a la hora de pasar de fase

Conceptos


Build pipeline [Jenkins]
Build Pipeline Plugin - Jen
localhost:8082/view/Build%20pipeline/
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...
Jenkins
search
marcinp | log out
Jenkins
Build pipeline
ENABLE AUTO REFRESH
Build Pipeline: My pipeline
Run
History
Configure
Add Step
Delete
Manage
Pipeline version
8
No parameters
Test
Jun 26, 2012 5:30:48 PM
8 10 sec
marcinp
Release
Jun 26, 2012 5:31:03 PM
8 12 sec
Deploy to Test
Jun 26, 2012 5:31:46 PM
6 ...
marcinp
Deploy to Pre-Prod
Deploy to Prod
Generate docs
Jun 26, 2012 5:31:20 PM
8 9.1 sec
Pipeline version
7
No parameters
Test
Jun 26, 2012 5:26:25 PM
7 10 sec
marcinp
Release
Jun 26, 2012 5:26:40 PM
7 12 sec
Deploy to Test
Jun 26, 2012 5:27:13 PM
5 13 sec
marcinp
Deploy to Pre-Prod
Jun 26, 2012 5:27:31 PM
4 10 sec
Deploy to Prod
Jun 26, 2012 5:27:46 PM
4 15 sec
Generate docs
Jun 26, 2012 5:26:57 PM
7 9.1 sec
Help us localize this page
Page generated: 26-Jun-2012 17:31:39
Jenkins ver. 1.470

Conceptos

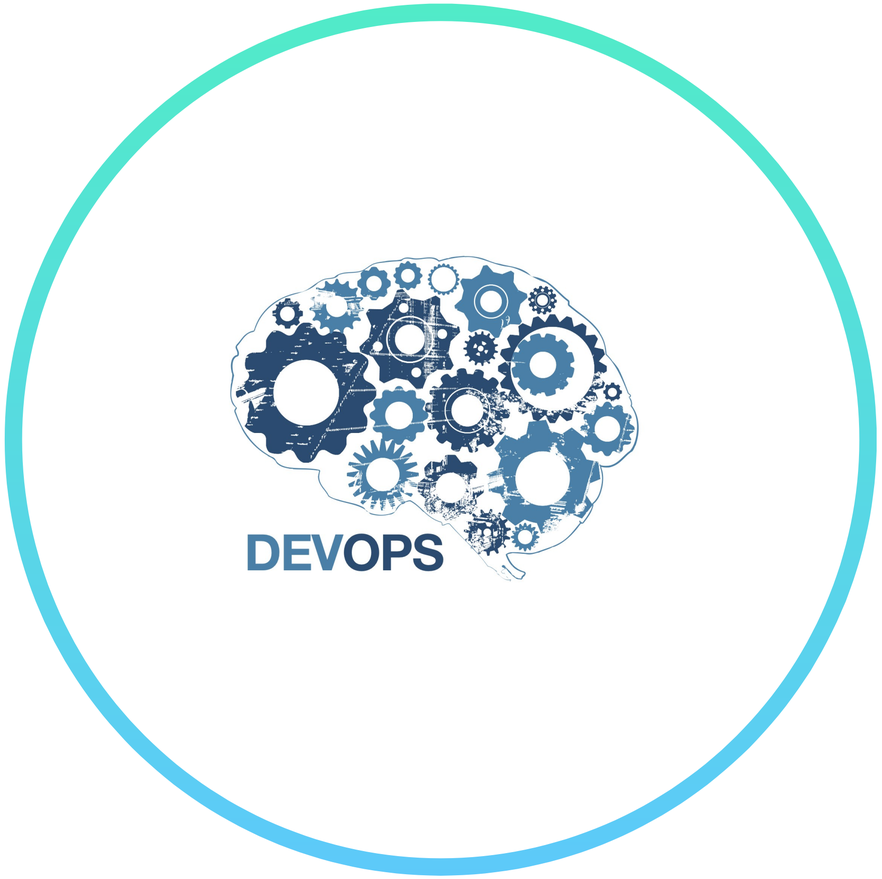
Repositorio



El repositorio es el sistema de gestión de versiones de código que se aplica a una parte del proyecto

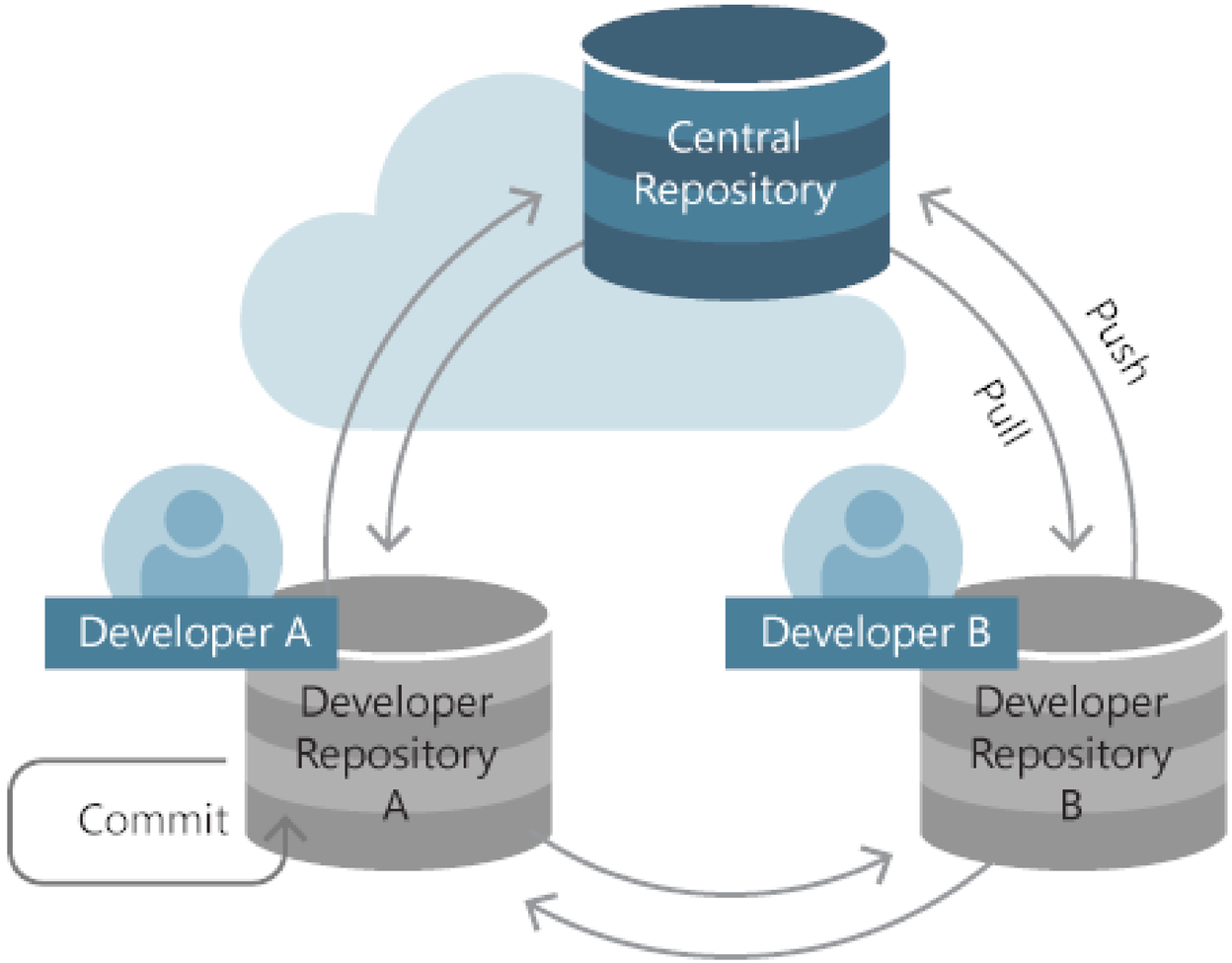
Conceptos

Repositorio



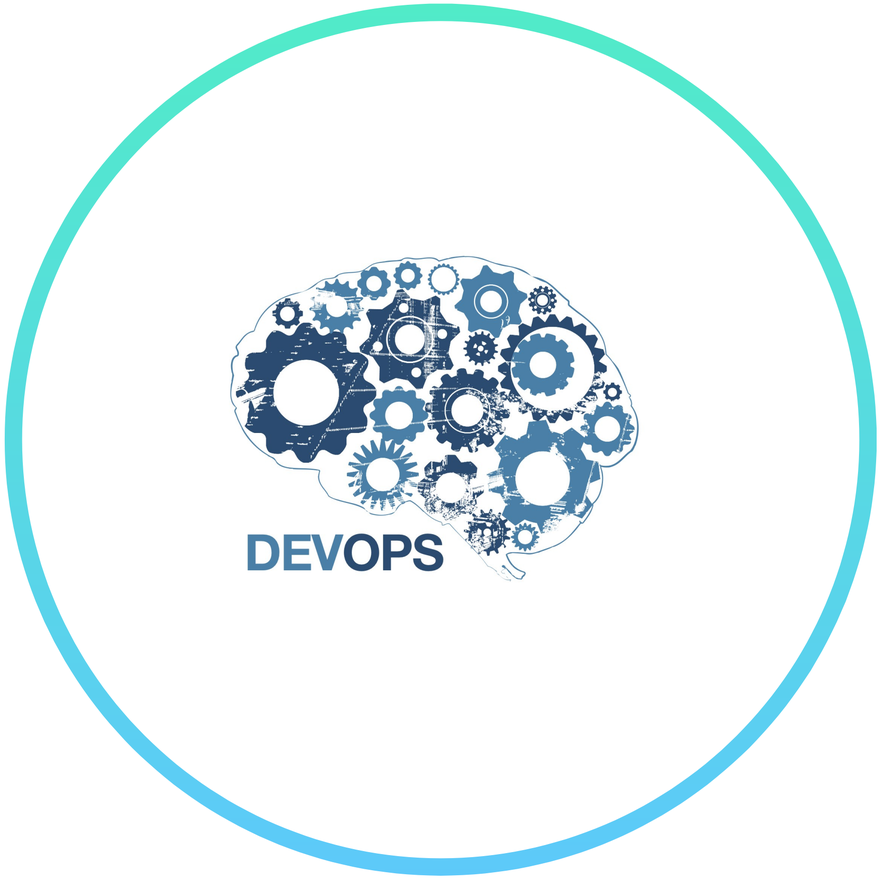
Estos repositorios serán necesarios para que se guarde el histórico de cambios de código del proyecto

Conceptos


Central
Repository
Push
Pull
Developer A
Developer
Repository
A
Commit
Developer B
Developer
Repository
B

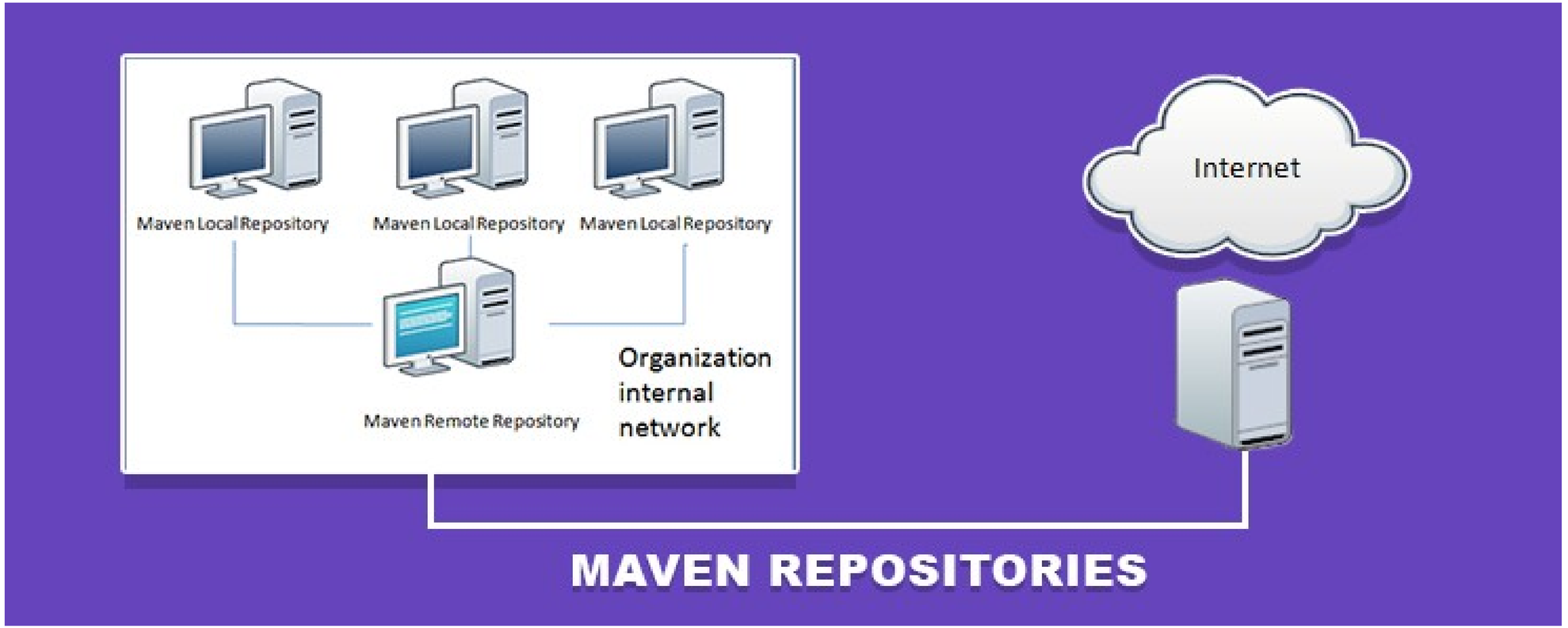
Conceptos

Gestor de dependencias



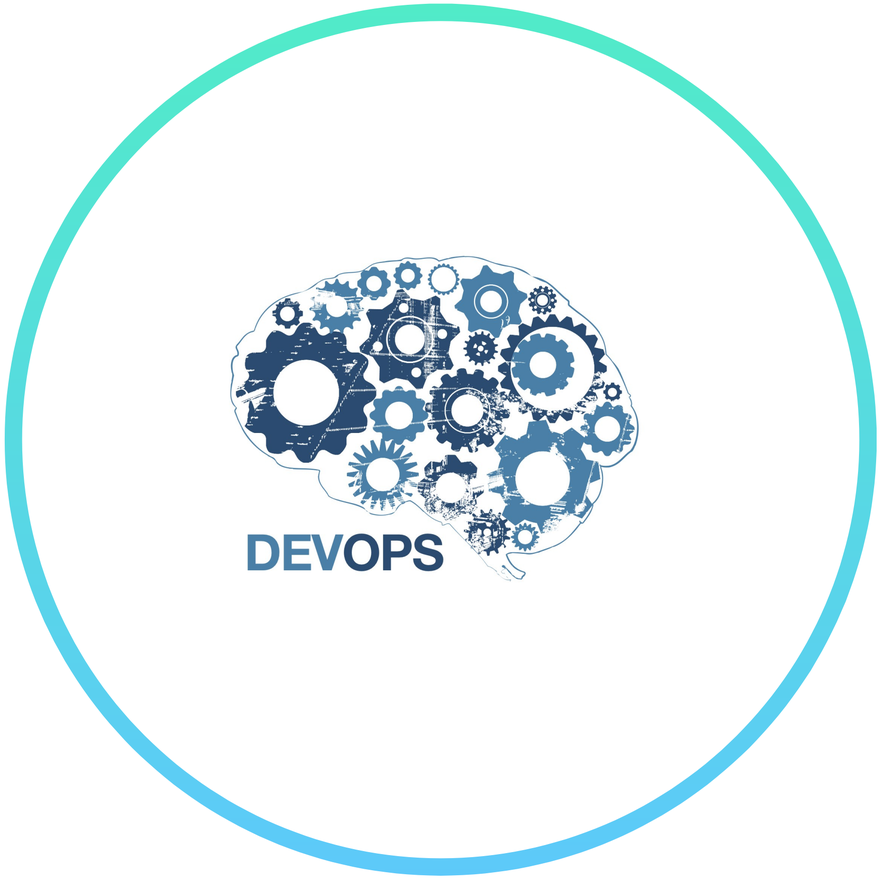
Software que permite la gestión de las dependencias software de un proyecto

Conceptos


Maven Local Repository
Maven Local Repository
Maven Local Repository
Maven Remote Repository
Organization internal network
Internet
MAVEN REPOSITORIES

Conceptos

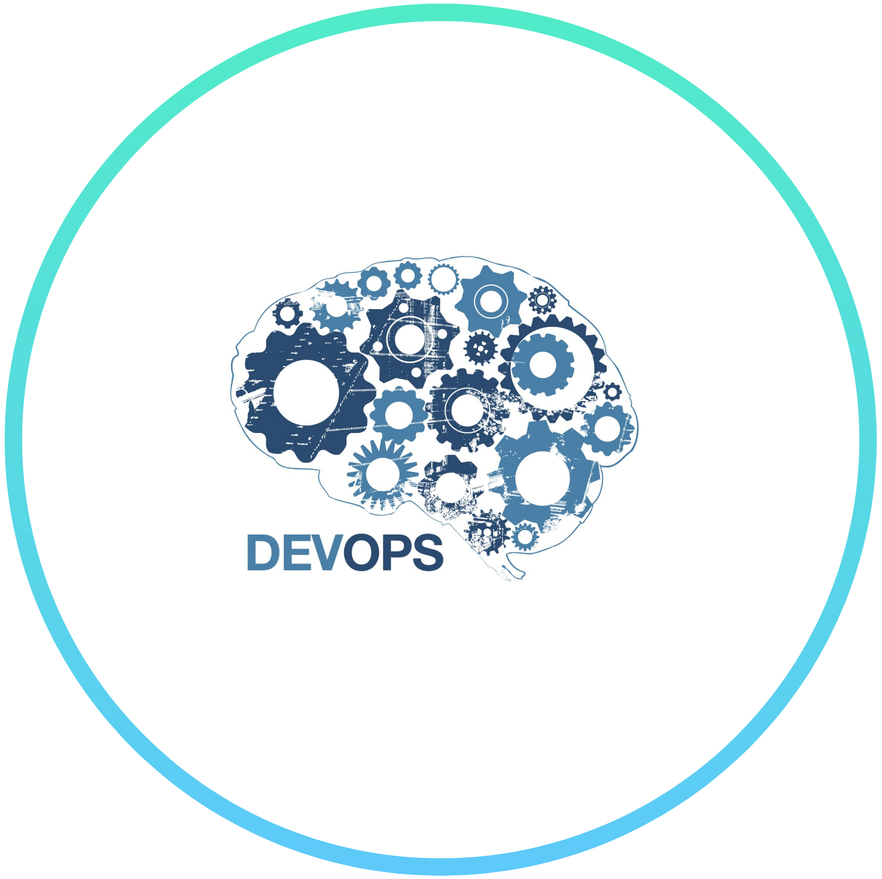
Gestor de dependencias



Normalmente guarda la versiones del software del que dependemos y facilita su descarga y compilación del proyecto

Conceptos

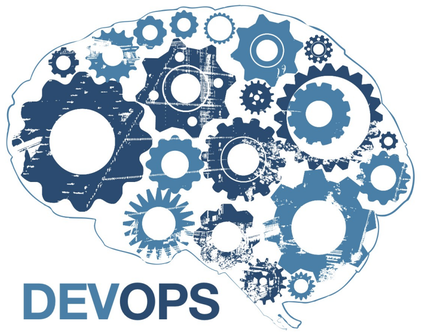
Gestor de dependencias



Estos sistemas habitualmente también permite la ejecución de tareas al rededor del proyecto: empaquetado, lanzamiento de pruebas, despliegue, etc...

Objetivos

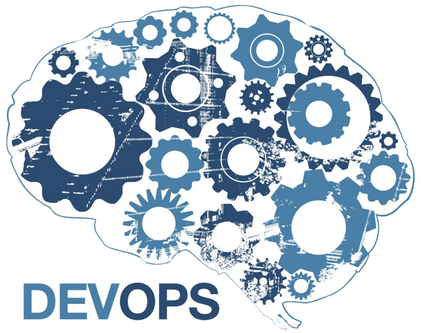
Principales



A continuación detallaremos los objetivos principales y secundarios de este tipo de técnicas y cultura de desarrollo

Objetivos

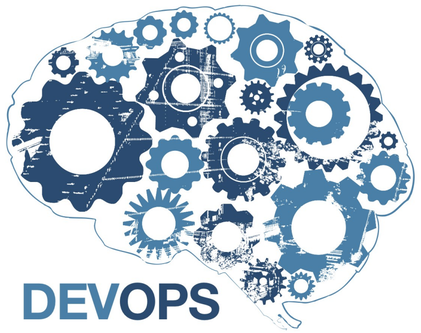
Principales



- Aumentar la experimentación
- Aumentar la Capacidad de tomar riesgos
- Reducir los fallos en las nuevas entregas
- Mejorar la frecuencia de entregas
- Reducir el tiempo para una recuperación
- Reducir el tiempo de una funcionalidad puesta en producción desde la detección de la necesidad
- Automatizar la Recogida del Feedback

Objetivos

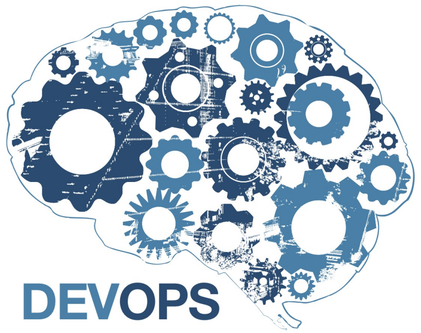
Código



- Introducir la revisión de código
- Evaluar la calidad de código
- Automatizar la compilación y empaquetado
- Automatizar las Pruebas
- Automatizar la Gestión de fallos

Objetivos

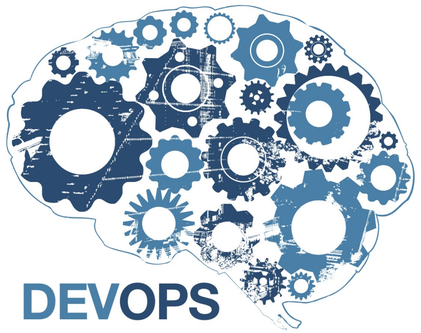
Sistemas



- Automatizar el Despliegue
- Automatizar la Monitorización
- Reducir los tiempos de puesta en marcha
- Reducir los tiempos de intervención ante incidencias
- Reducir las incidencias
- Reducir la intervención humana ante fallos

Principios

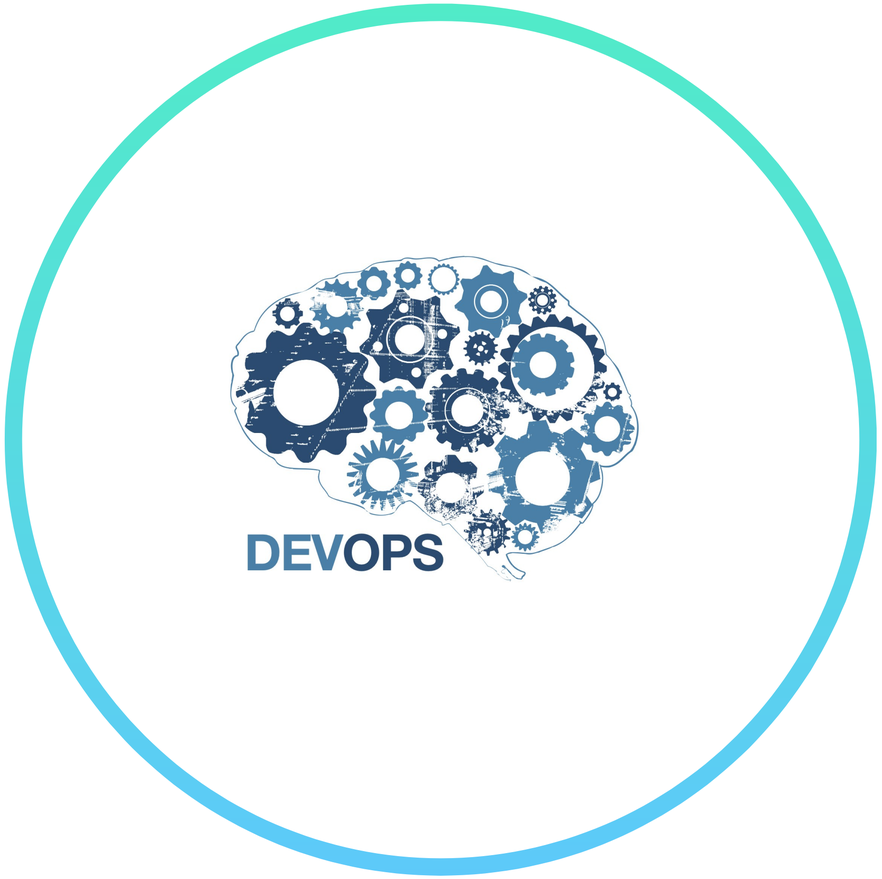
Buenas Prácticas



- Proceso confiable replicable
- Automatiza Todo
- Controla las Versiones de Todo
- Traete el Dolor Antes
- Todos son Responsables
- Construye con Calidad Desde el Principio
- Hecho significa liberado

Principios

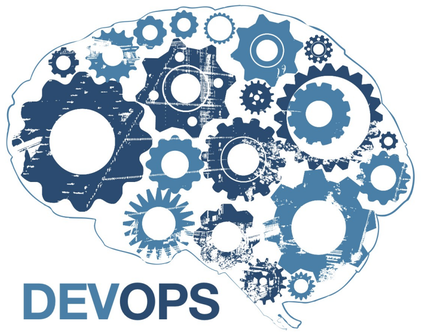
Proceso Confiable Replicable



- Utiliza el mismo método en todas partes
- Mantén la lógica fuera de las herramientas de CI/CD
- Utiliza scripts que sean llamados desde las herramientas de compilación

Principios

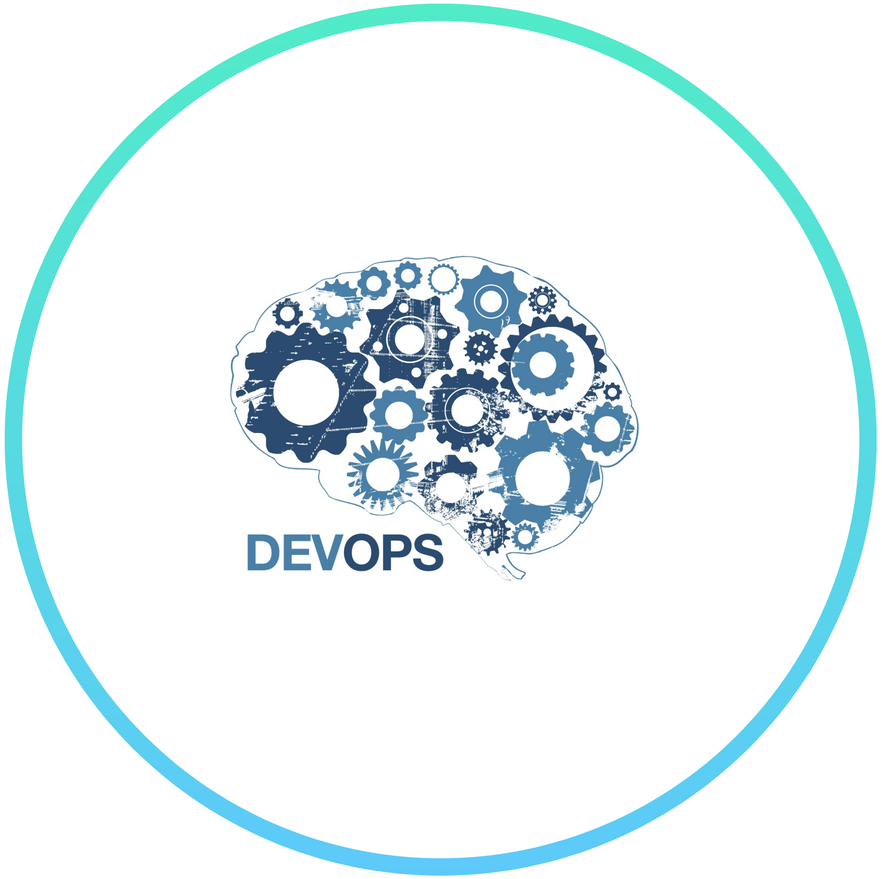
Automatiza Todo



- Test de regresión (aceptación)
- Aprovisionamiento
- Despliegues

Principios

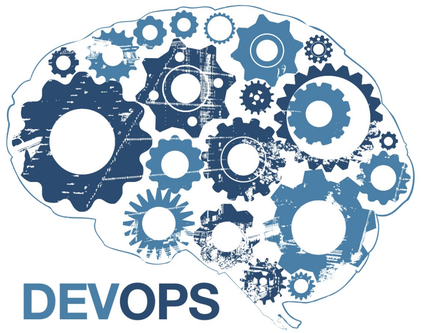
Controla las Versiones de Todo



- Código de la aplicación
- Configuraciones
- Scripts de Base de Datos
- Documentación

Principios

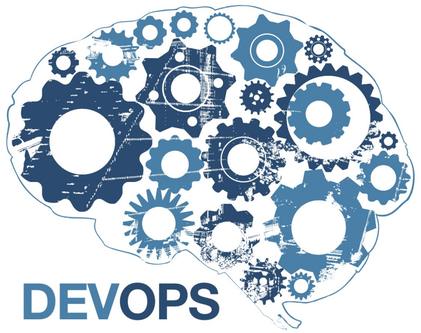
Trae el Dolor Antes



- Haz las cosas complicadas al principio
- Liberar más a menudo reduce el riesgo de fallos

Principios

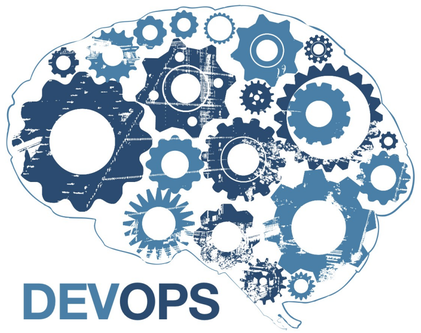
Todos son Responsables



- El “En mi máquina funciona” no vale como excusa
- El cambio cultural es el más complicado
- El soporte de gestión es vital
- Ten cuidado con las métricas

Principios

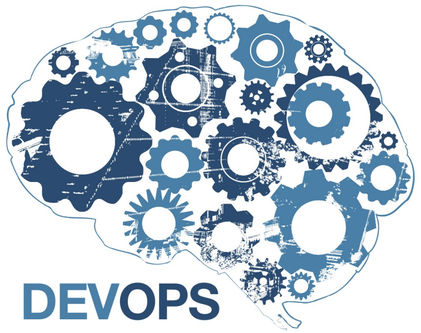
Calidad Desde el Principio



- La calidad no sólo está en si algo es funcional
- El código puede medirse de muchas maneras
- Las pruebas son fundamentales
- La seguridad es importante en el proceso de desarrollo
- El rendimiento se puede medir desde el primer día

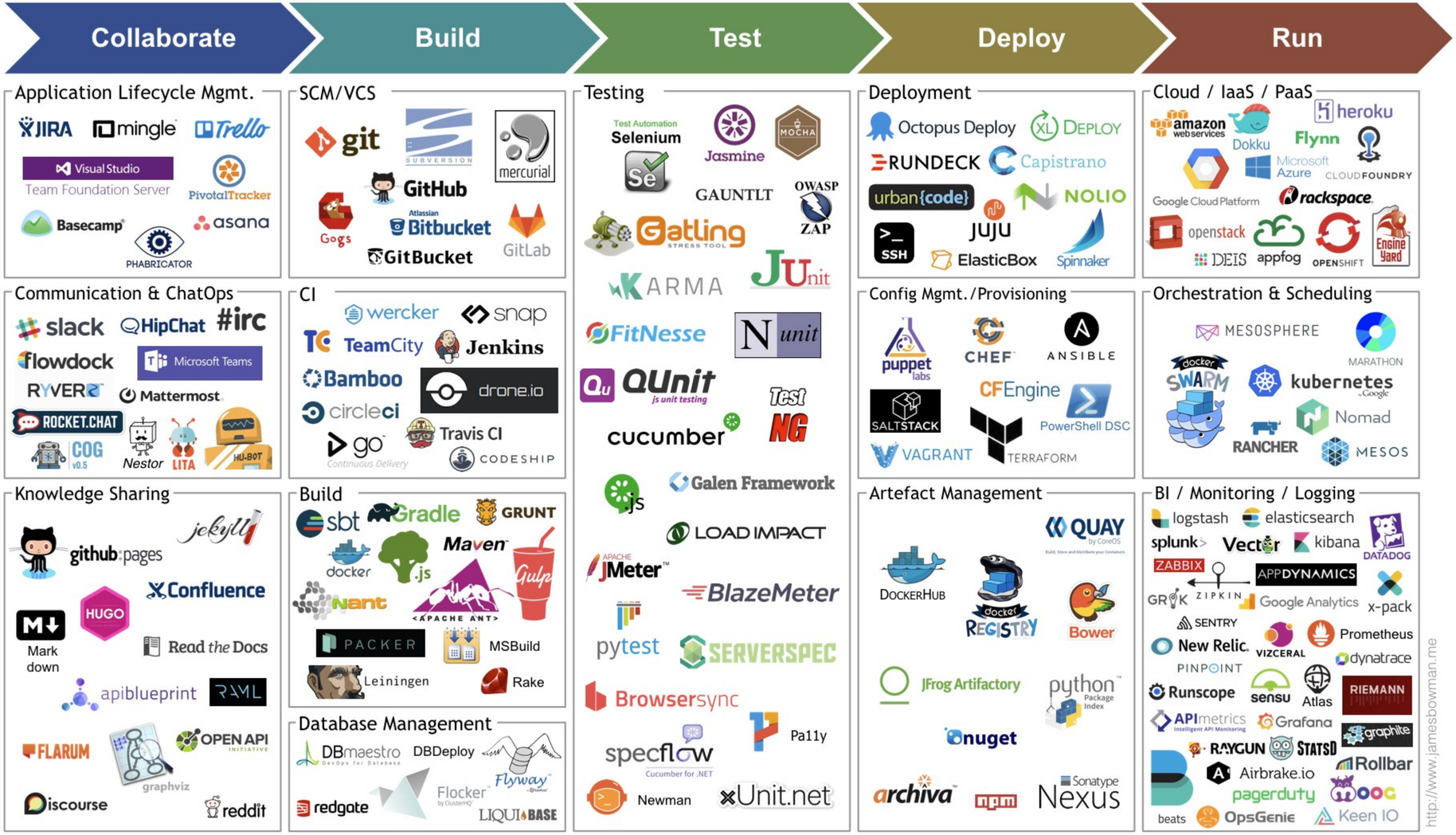
Principios

Hecho significa liberado



- No está hecho hasta que está subido
- No confundas liberado son desplegado
- Considera métodos como “features toggles” (cambiar el comportamiento por configuración)
- Las liberaciones son una decisión de negocio

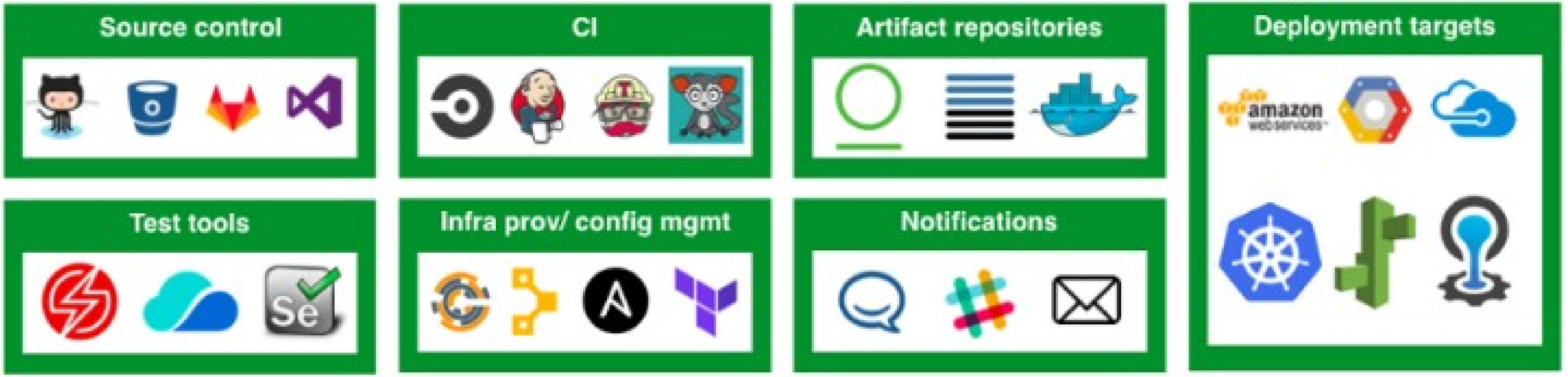
Herramientas



http://www.jamesbowman.me

http://www.jamesbowman.me/post/continuous-delivery-tool-landscape/

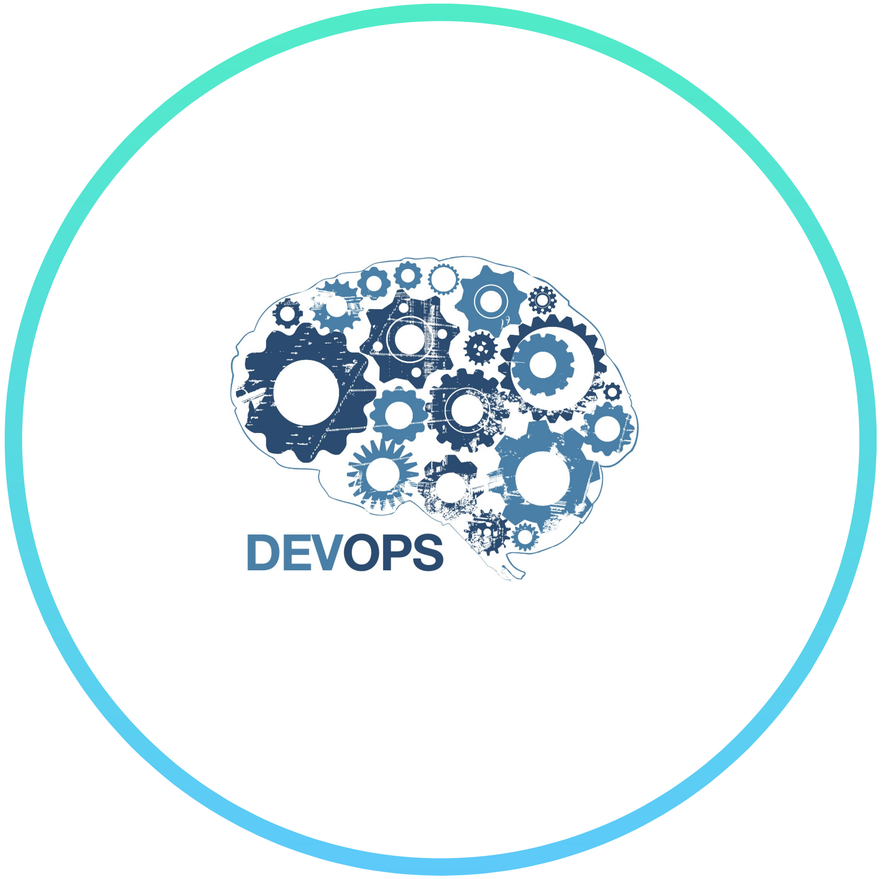
Herramientas



DevOps Islands Of Automation

Herramientas

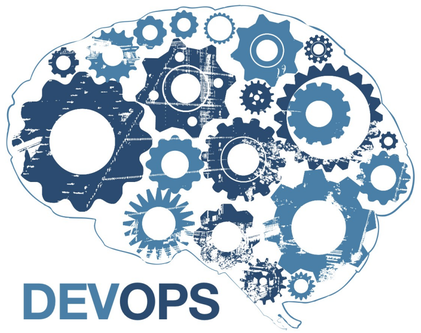
Funcional



Requisitos
Documentación
Análisis
Historias de Usuario
Gestión y Reparto de Tareas
Comunicación

Herramientas

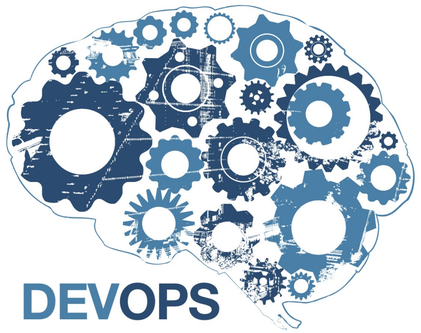
Desarrollo



Repositorio de Código
Repositorio de Artefactos
Gestión y Reparto de Tareas
Pruebas de Unidad e Integración
Compilación
Empaquetado
Comunicación
Documentación

Herramientas

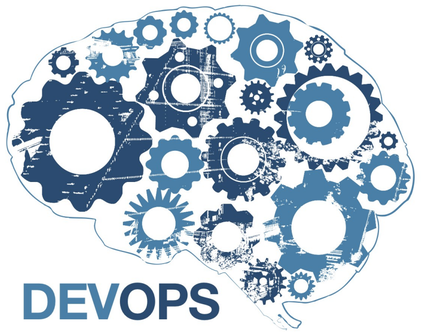
Testing



Historias de Usuario
Gestión y Reparto de Tareas
Comunicación
Repositorio de Pruebas
Integración Continua
Pruebas de Aceptación
Pruebas de Rendimiento
Seguimiento de Resultados
Documentación

Herramientas

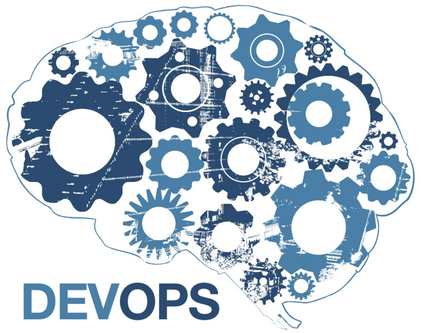
Seguridad



Repositorio de Pruebas
Pipeline de Seguridad
Pruebas de Ataques
Informes de seguridad
Documentación

Herramientas

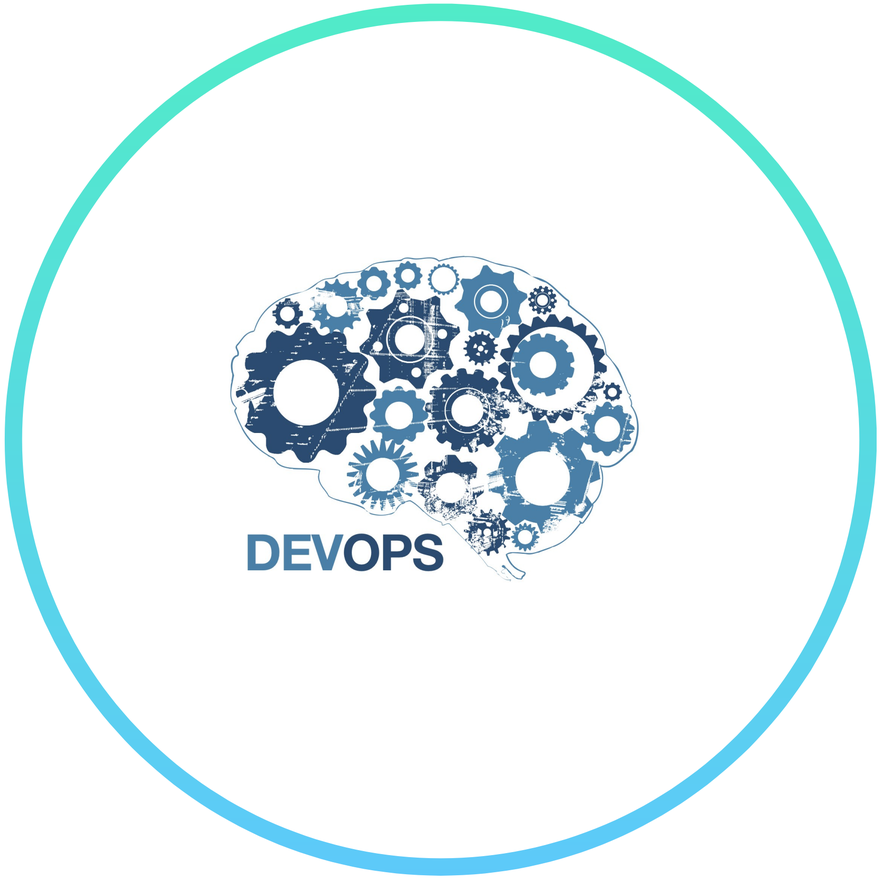
Operaciones



Aprovisionamiento
Repositorio de Configuraciones
Gestión de Incidencias
Monitorización
Empaquetado de Aplicaciones
Empaquetado de Despliegues
Infraestructura
Orquestación
Documentación

Versionado

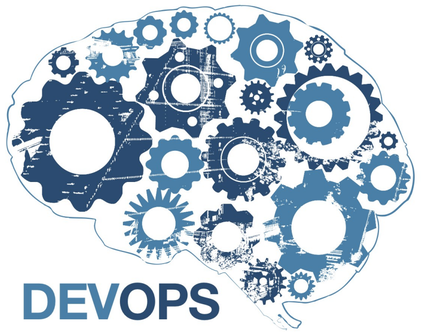
Código



El versionado se ha convertido en una parte crucial a la hora de manejar no sólo el código de las aplicaciones sino toda la información del proyecto

Versionado

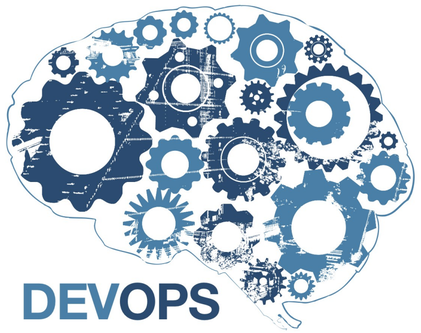
Código



Para ello los repositorios de código y de documentación se han convertido ya es parte imprescindible de la gestión del proyecto

Versionado

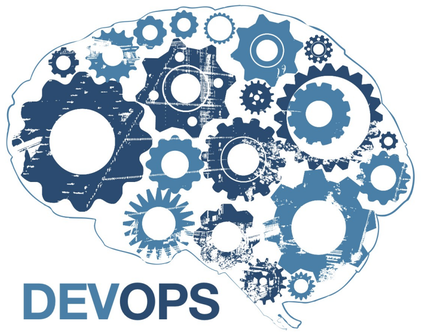
Código



Existen distinto software que permite realizar esta gestión de versiones, empecemos por el código

Versionado

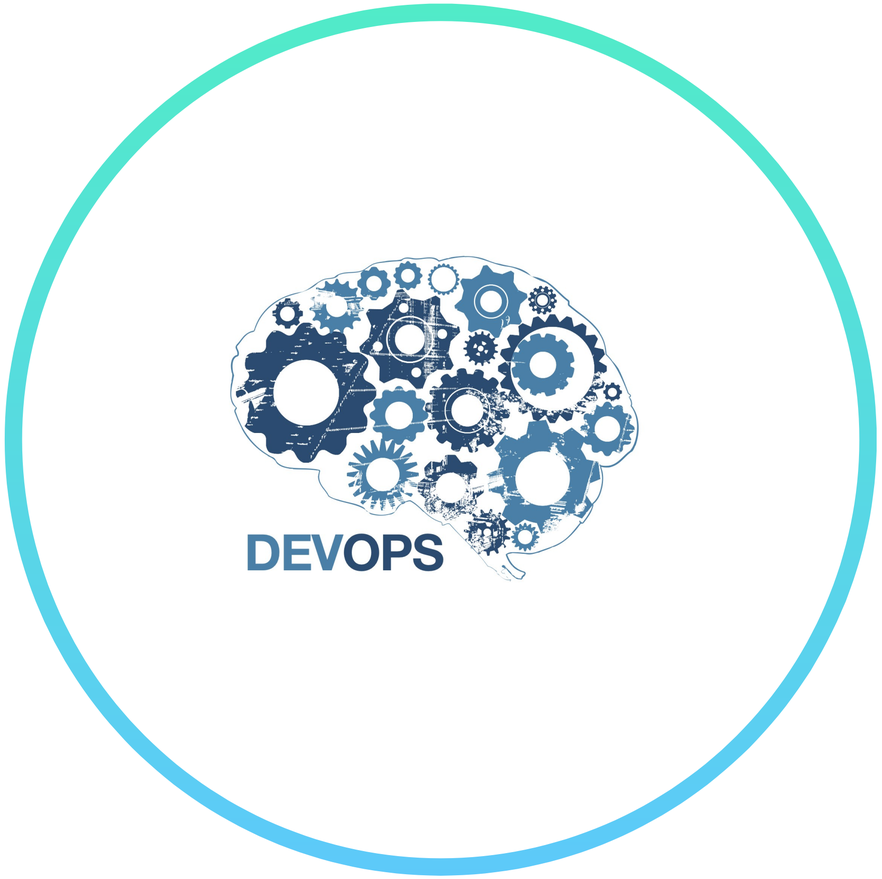
Código



Git, es la herramienta principal de gestión de código en la actualidad
Se ha convertido en el substituto de SVN y por supuesto de CVS

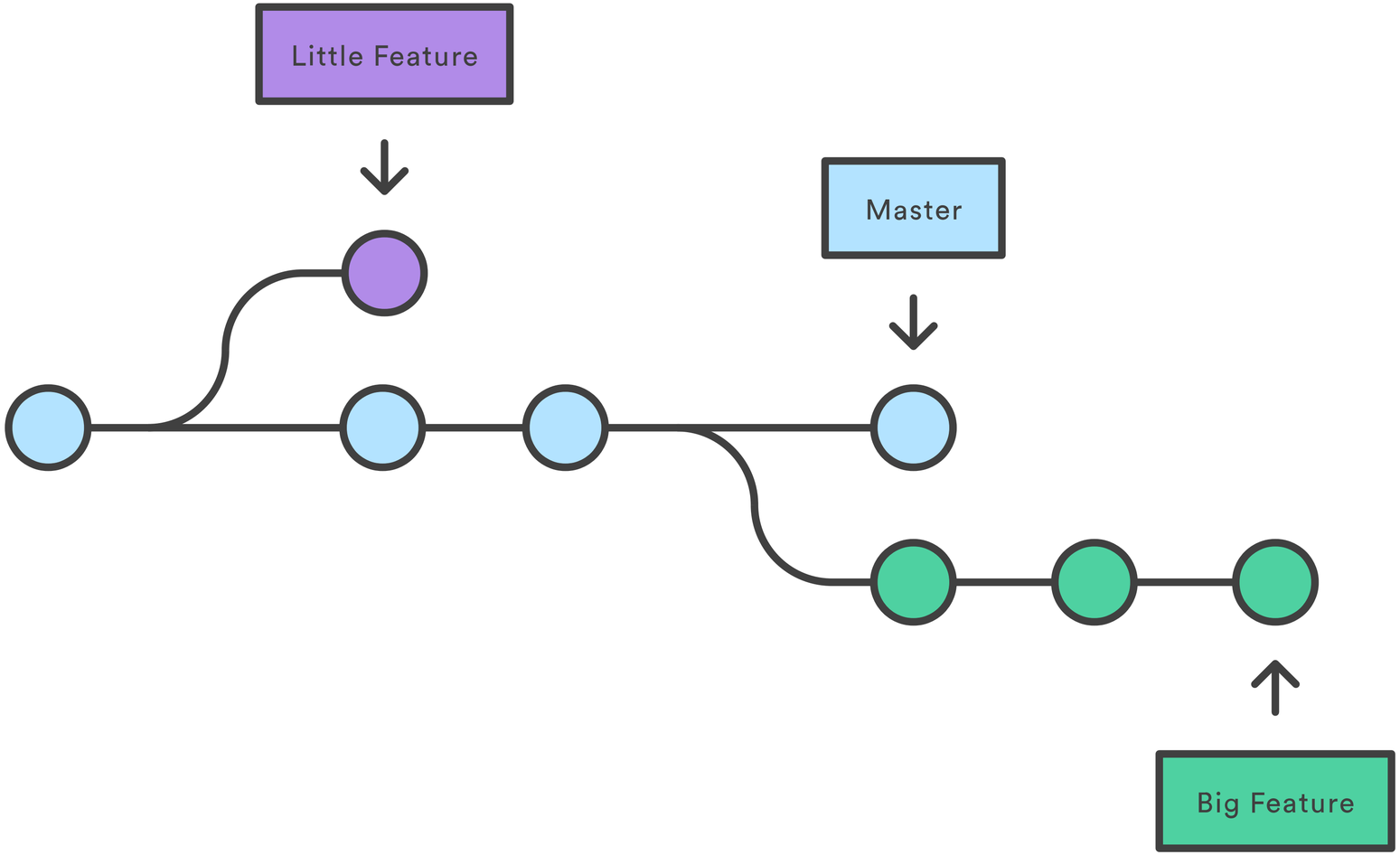
Versionado

Código



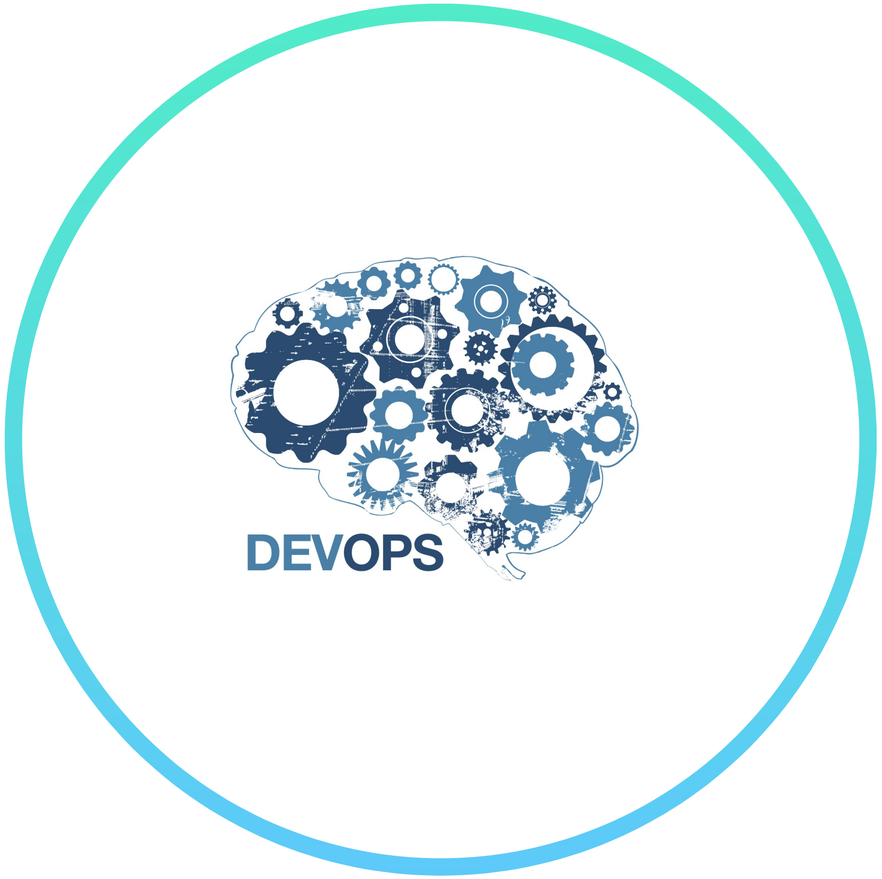
Git es un sistema distribuido de versiones por lo que tiene una serie de ventajas sobre el resto de sistemas

Versionado



Versionado

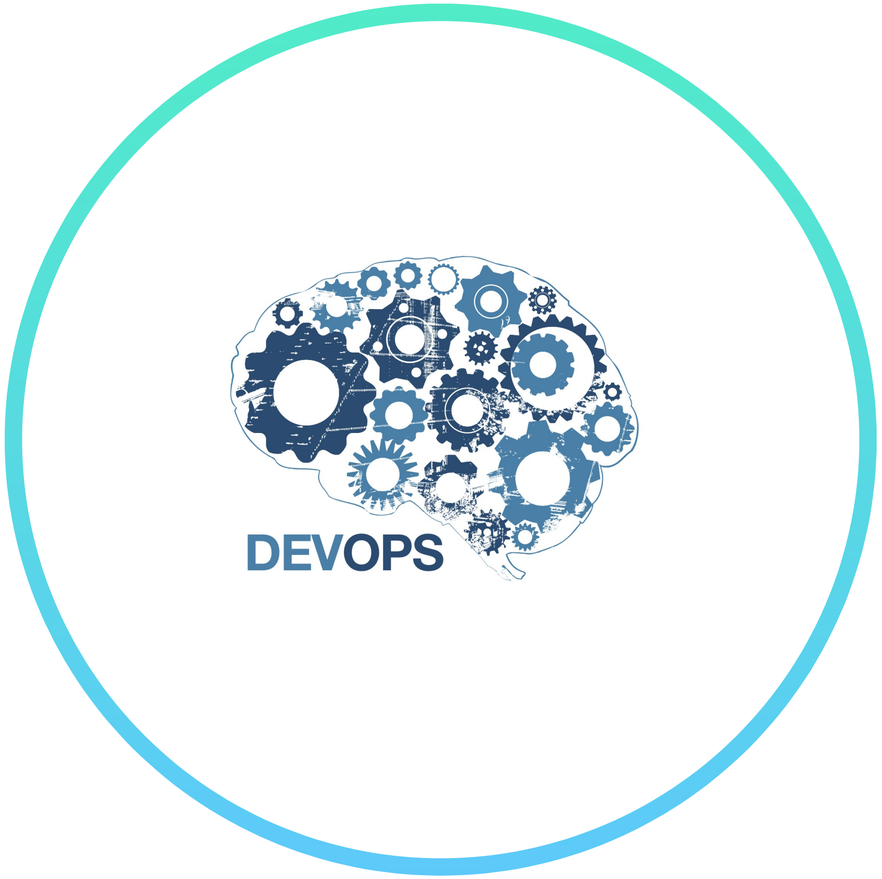
Código



Los commit son en local
El desarrollador puede ir consolidando cambios en local antes de enviar al servidor los cambios de una funcionalidad nueva

Versionado

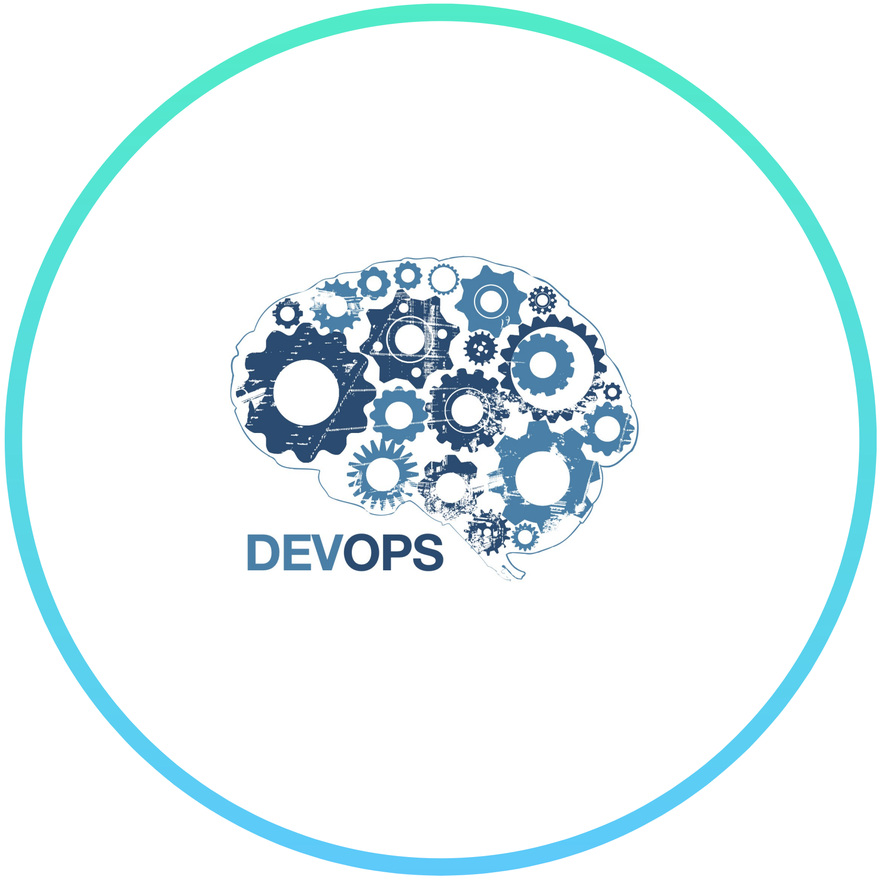
Código



La gestión de ramas de código es muy ágil
Cada una de las ramas que se abren en un proyecto pueden sincronizarse con otras de una manera mucho más rápida y sencilla

Versionado

Código



El uso que se hace de esas ramas depende de muchos factores pero normalmente se realizan para las siguientes tareas:

Liberar versiones

Resolver funcionalidades

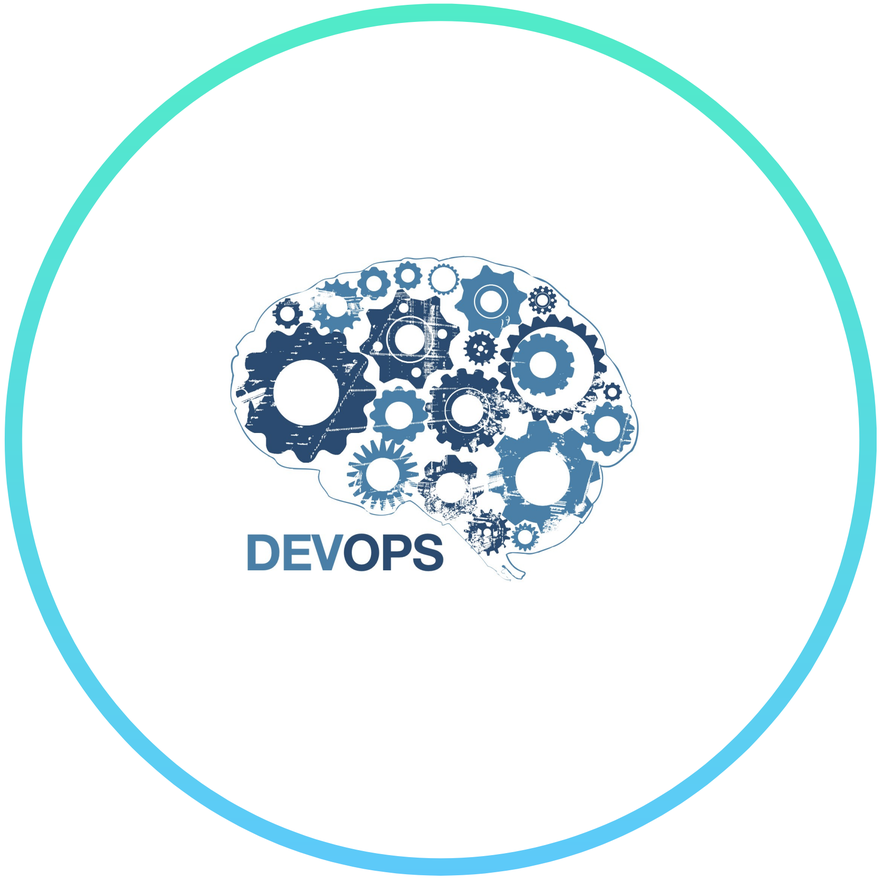
Coordinar código de integración

Experimientos

Dar soporte a distintas versiones

Versionado

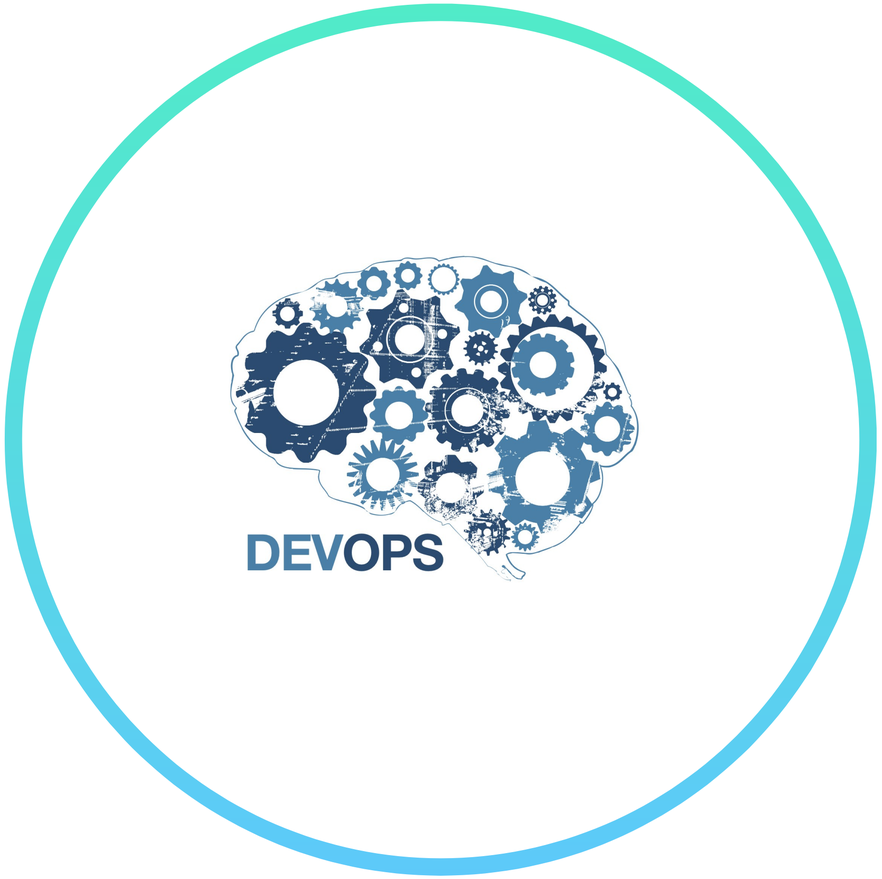
Código



Para que funcionen correctamente estos repositorios es necesario disponer de un sistema en un servidor

Versionado

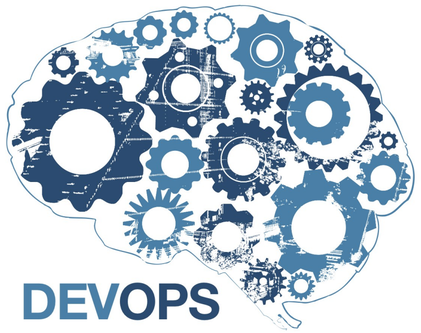
Código



Github
Gitlab
BitBucket
Son varias opciones de instalación de este servidor

Versionado

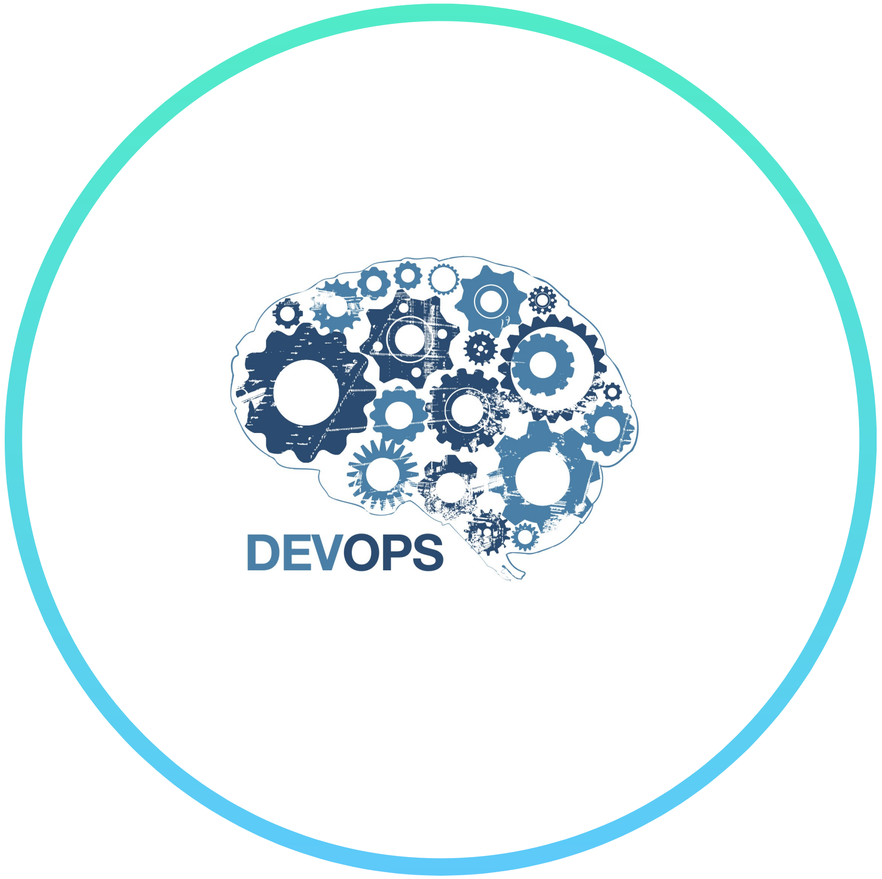
Código



La mayor parte de los servidores de código tienen la posibilidad de integrarse con otras herramientas del proceso de desarrollo

Versionado

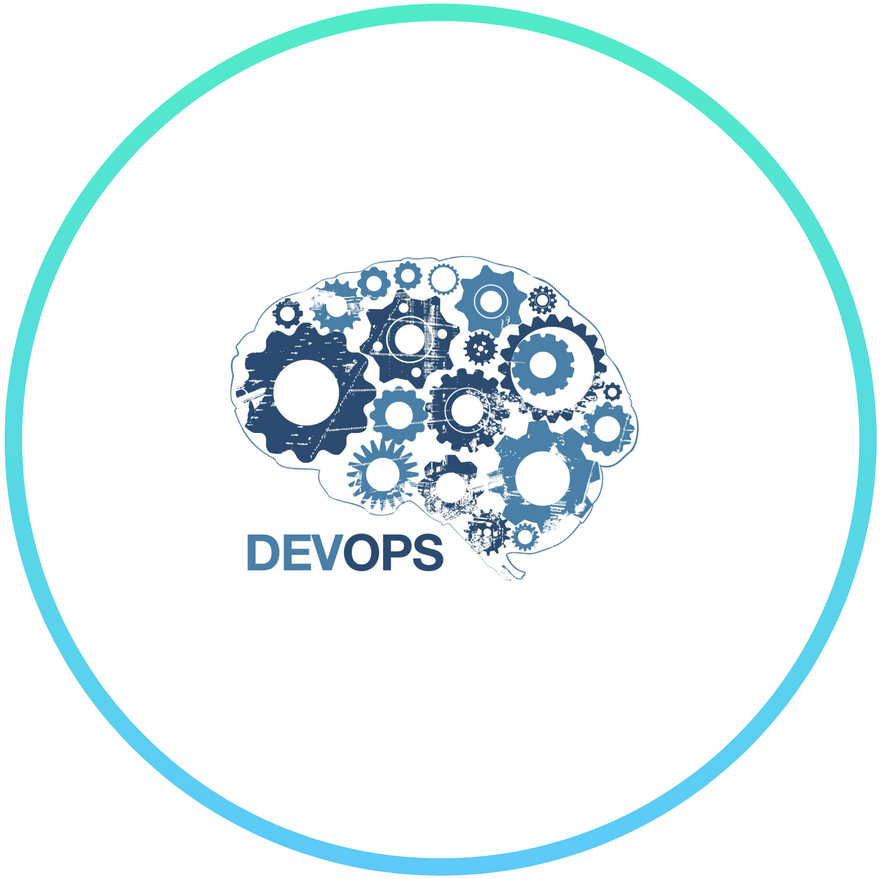
Documentación



En el mercado disponemos de multitud de herramientas de versionado de documentación

Versionado

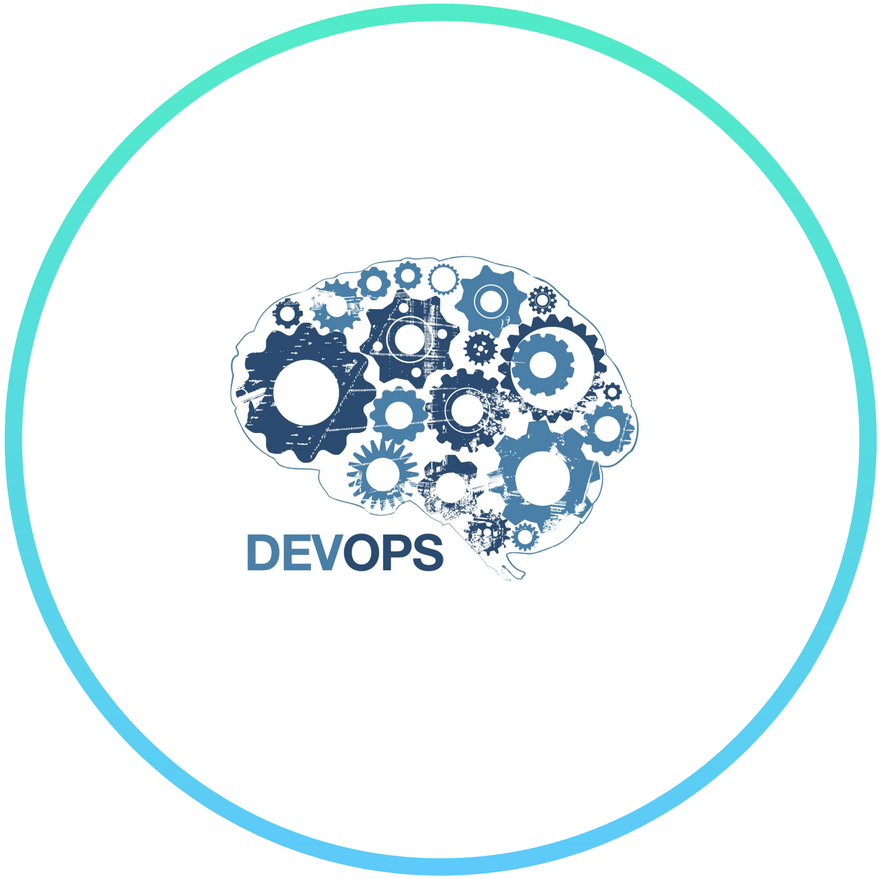
Documentación



Un simple Wiki permite guardar el histórico de cambios de un documento web

Versionado

Documentación



Soluciones más avanzadas de Gestor Documental (ECM) como Athento (Nuxeo), OpenKM o Alfresco permiten realizar una gestión más cómoda, ya que la edición puede realizarse con las típicas suites de Ofimática

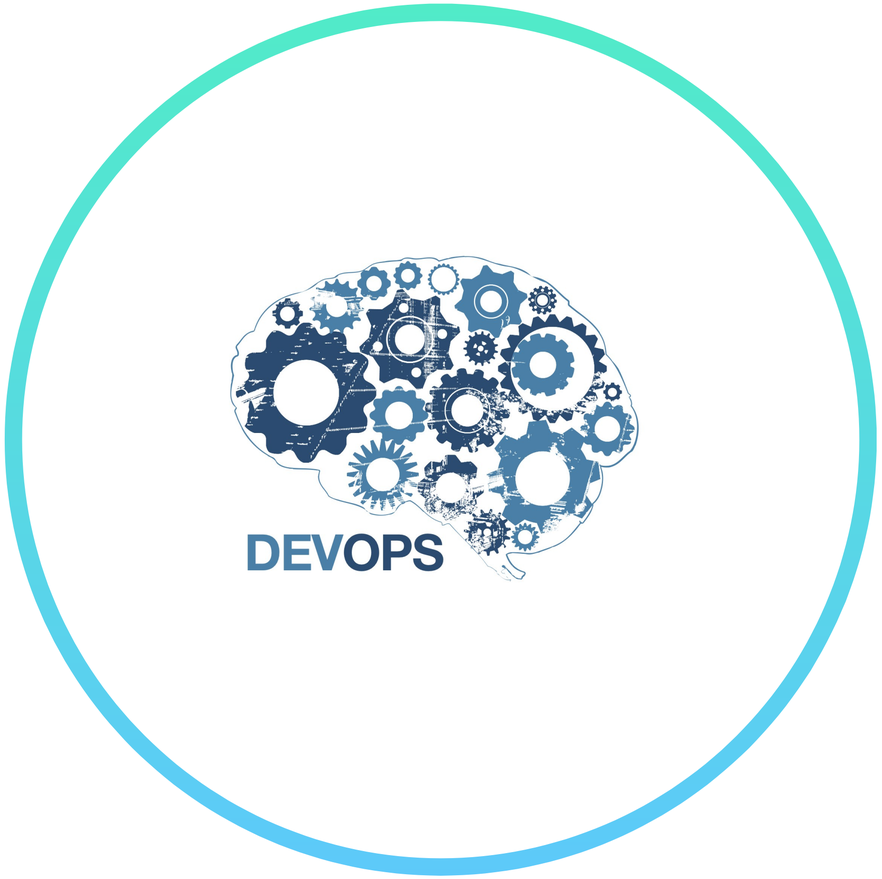
Versionado

Testing

Para las pruebas suele disponerse de un repositorio específico (o carpeta en el de desarrollo) donde se agrupan las pruebas automatizadas

Versionado

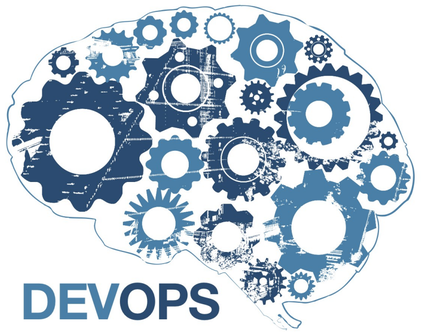
Testing



A parte de los informes en documento suele utilizarse algún tipo de herramienta de seguimiento de informes de resultados de las pruebas, que permita hacer su seguimiento

Versionado

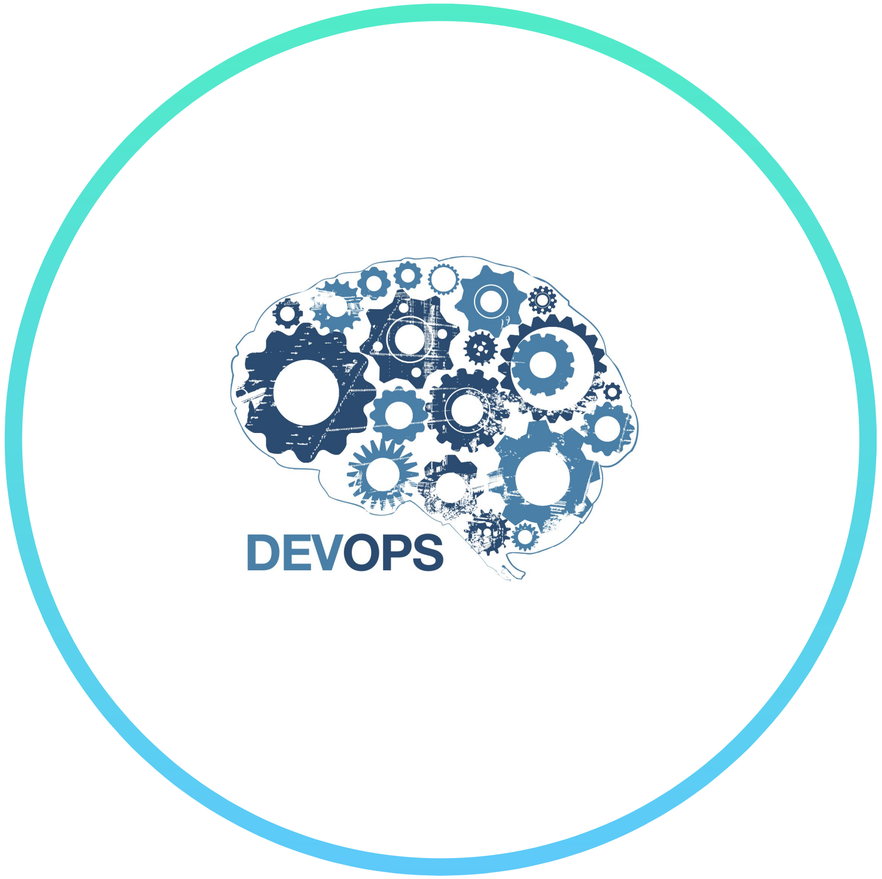
Operaciones



De cara a manejar las configuraciones el equipo de operaciones también suele tener un repositorio de código asociado al proyecto Scripts de Ansible, Chef o Puppet son algunos ejemplos de código de despliegle

Versionado

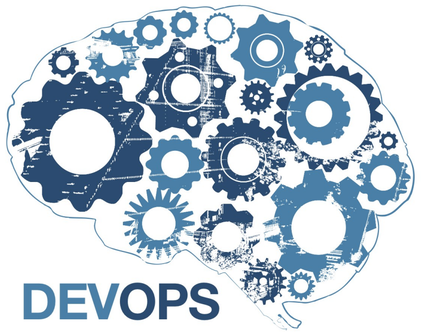
Operaciones



Por supuesto también tendremos que versionar los ficheros de configuración que son necesarios para lanzar el proyecto en pre y producción

Calidad De Software

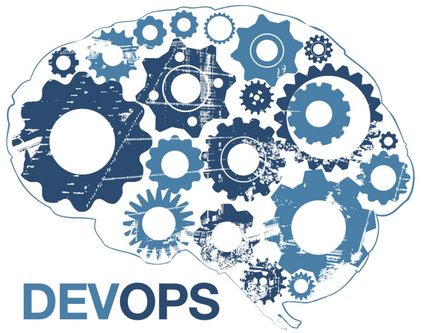
Código



Para poder especificar la calidad de un software será necesario aplicar una serie de actividades

Calidad De Software

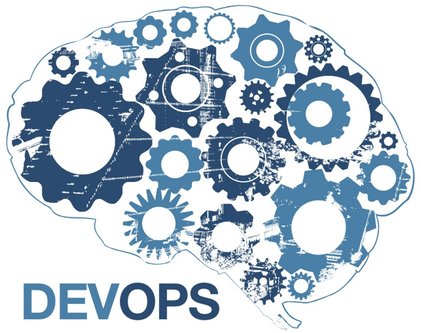
Código



Formulación: definición de las medidas y métricas apropiadas para el software que se está construyendo

Calidad De Software

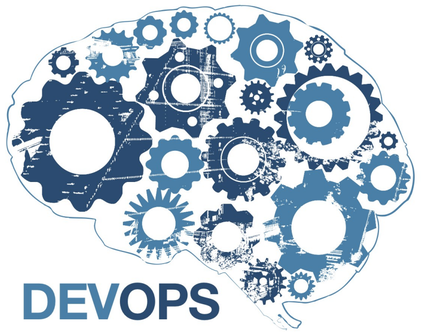
Código



Recolección: mecanismo de recopilación de datos basados en esas métricas

Calidad De Software

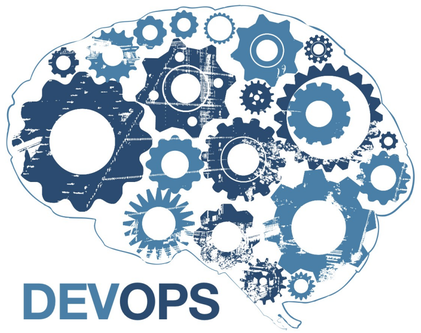
Código



Análisis: Cálculo de métricas y aplicación de herramientas

Calidad De Software

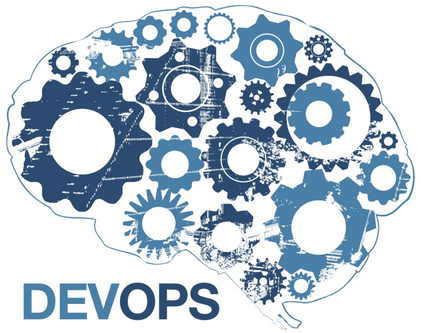
Código



Interpretación: Evaluación de las métricas

Calidad De Software

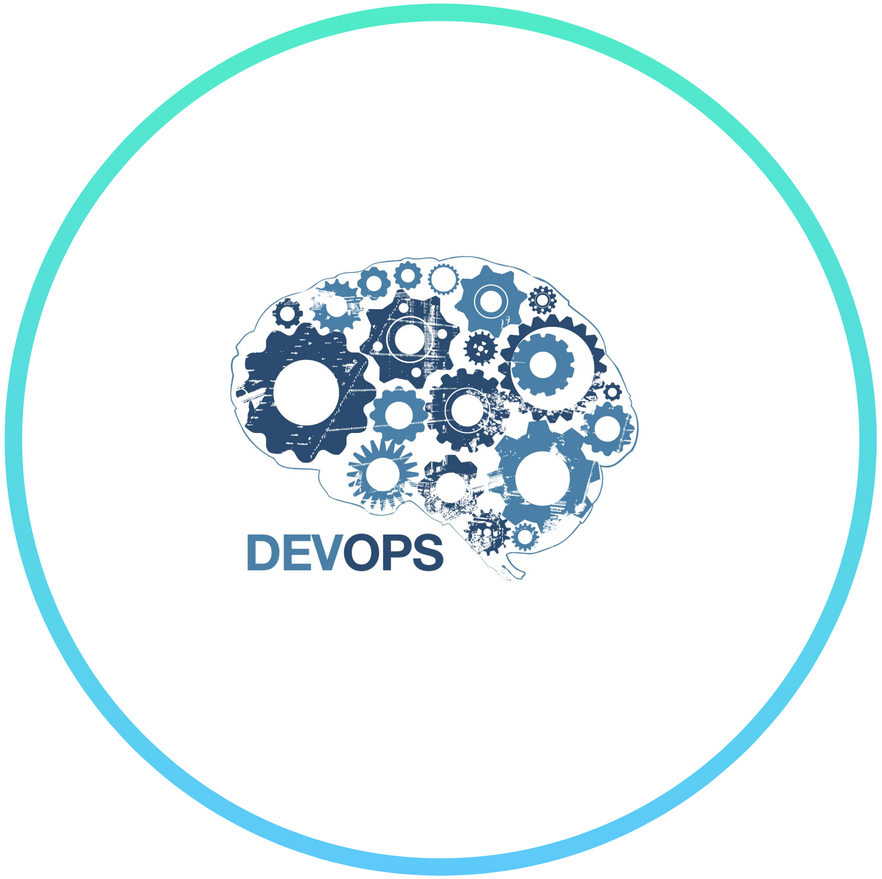
Código



Retroalimentación: Recomendaciones derivadas de la interpretación de las métricas

Calidad De Software

Código



Elementos a analizar de un código:

Confiabilidad

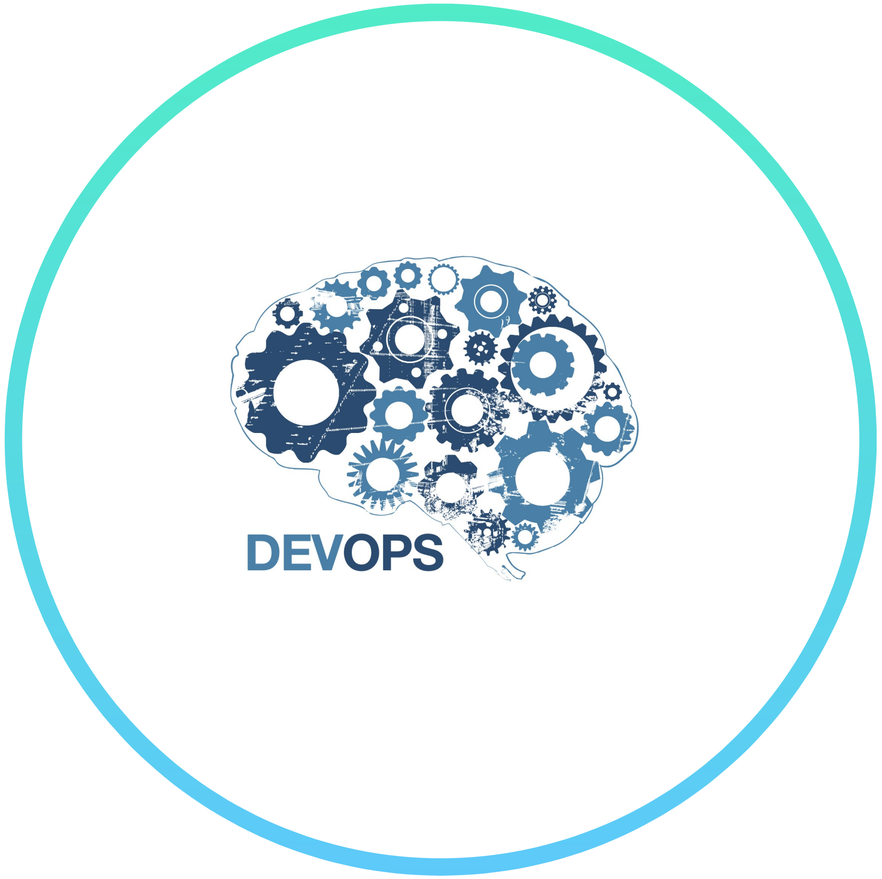
Rendimiento

Seguridad

Mantenibilidad

Calidad De Software

Código



Análisis Estático de código:

Posibles errores

Código Duplicado

Código muerto (no utilizado)

Estilo de codificación

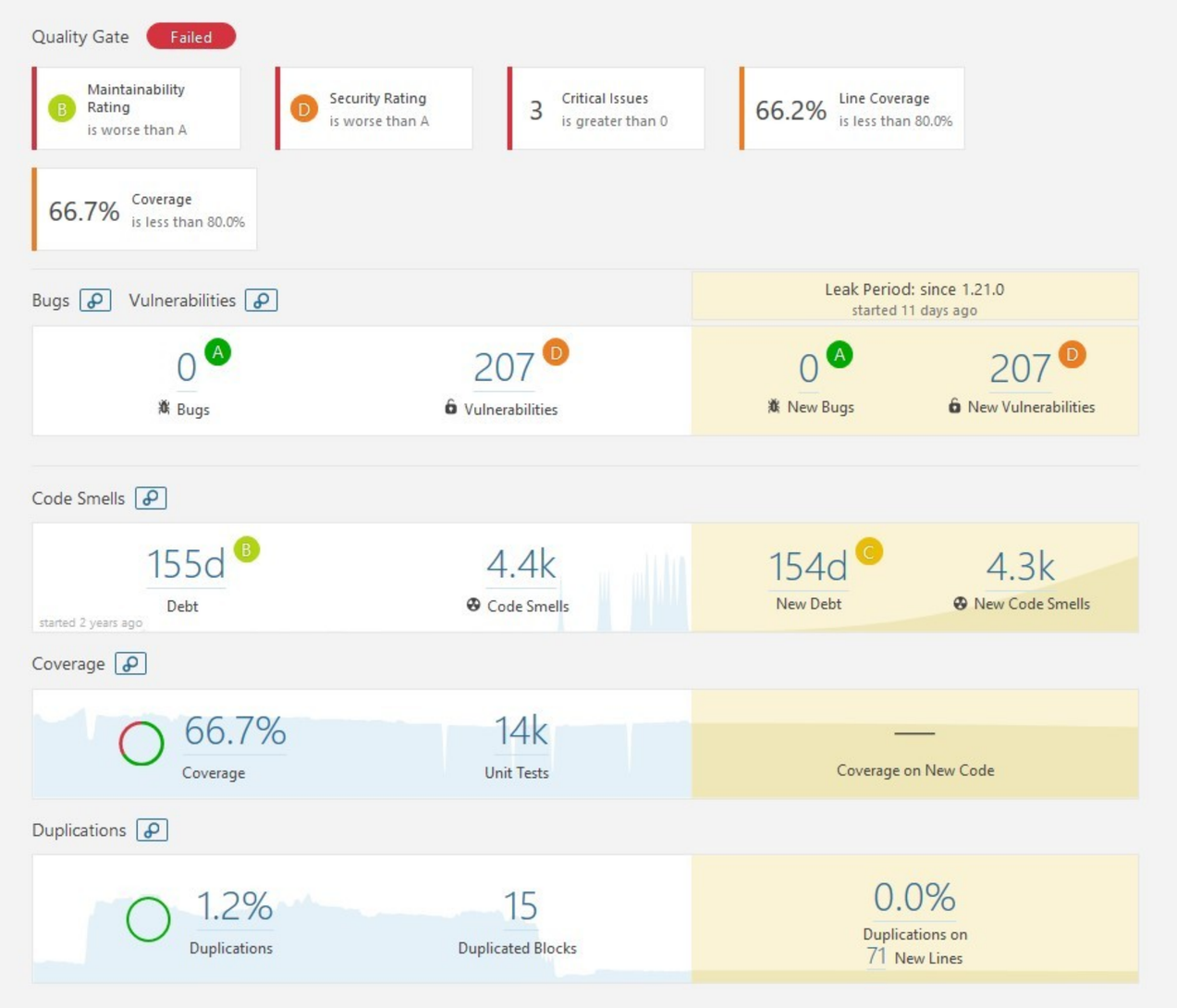
Comentarios

Expresiones sobrecomplicadas

Complejidad Ciclomática (caminos diferentes)

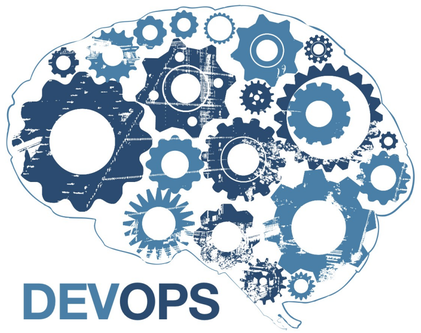
Cobertura de código (test)

Calidad de Software



Calidad De Software

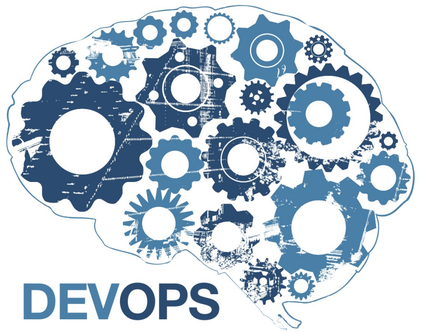
Código



El resto de mediciones pueden ir enfocadas al cumplimiento de los requisitos aplicables al proyecto mediante test

Conclusiones

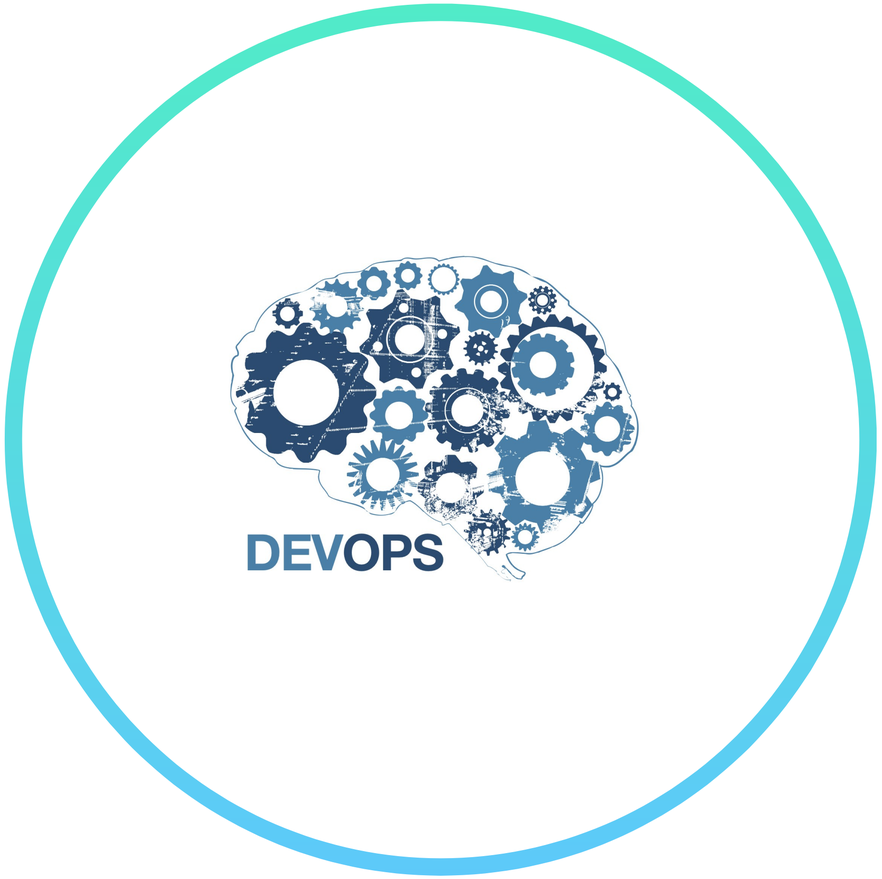
¿Qué podemos sacar en claro?



- Hemos definido los conceptos principales
- Hemos identificado las herramientas a utlizar

Referencias

¿Fuentes de información?



- Artículo sobre Integración Continua: https://www.martinfowler.com/articles/continuousIntegration.html
- Artículo sobre Feature Toggles: https://martinfowler.com/articles/feature-toggles.html

CURSOS DE DESARROLLO
DAVID VAQUERO
LICENCIA CC-BY-SA-NC 4.0

info@cursosdedesarrollo.com

http://cursosdedesarrollo.com