Certificación avanzada en DevOps

Build & Package Manager Tools

UNC FCEFyN mundosE PEOPLE & BUSINESS SCHOOL

Retomemos lo trabajado en el encuentro anterior

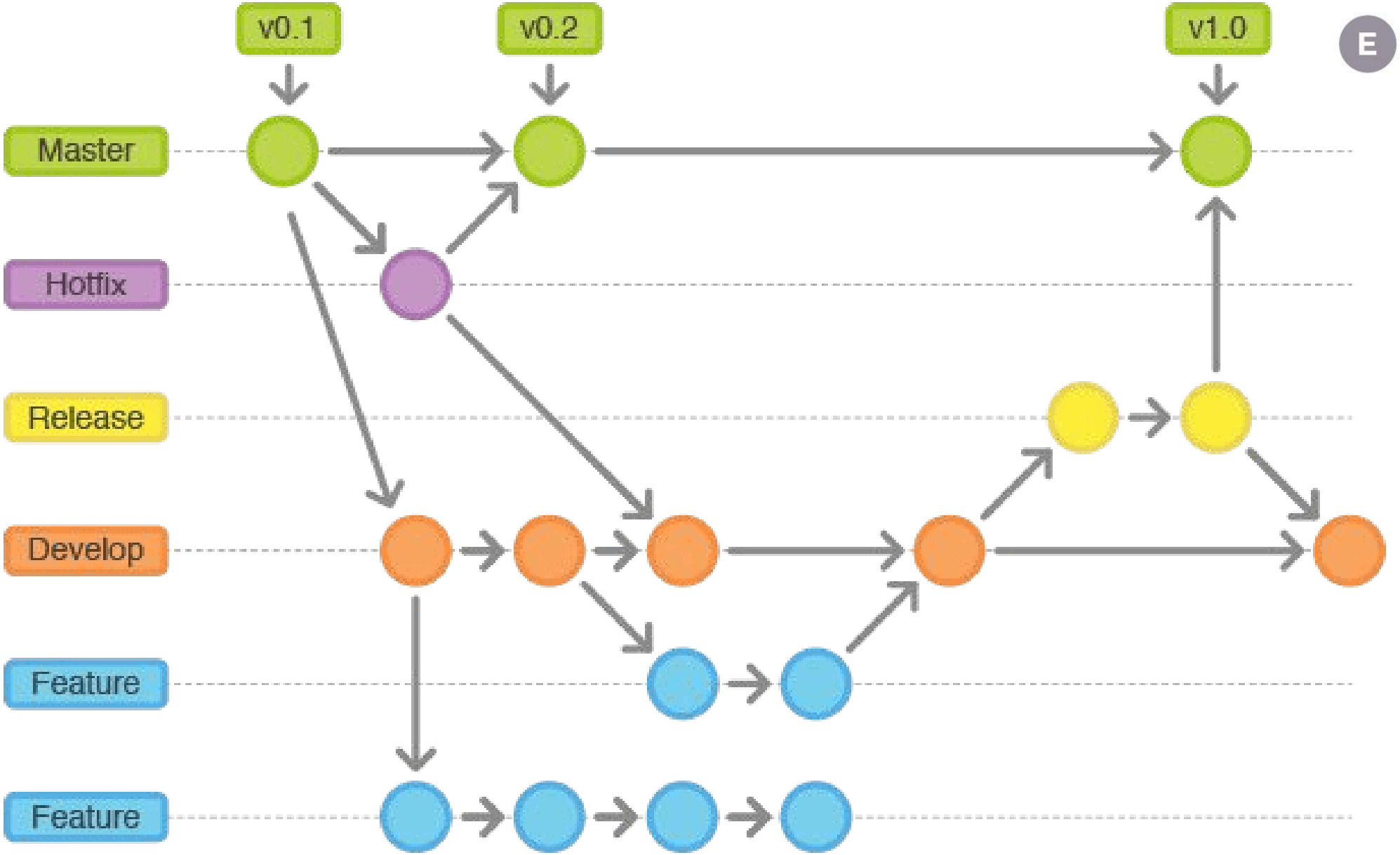
Git - Git hub

¿Alguna duda?

Estrategias de Branching

Existen muchas estrategias de branching. Una de la más utilizada es la que establece GitFlow (modelo de branching creado por Vincent Driessen):

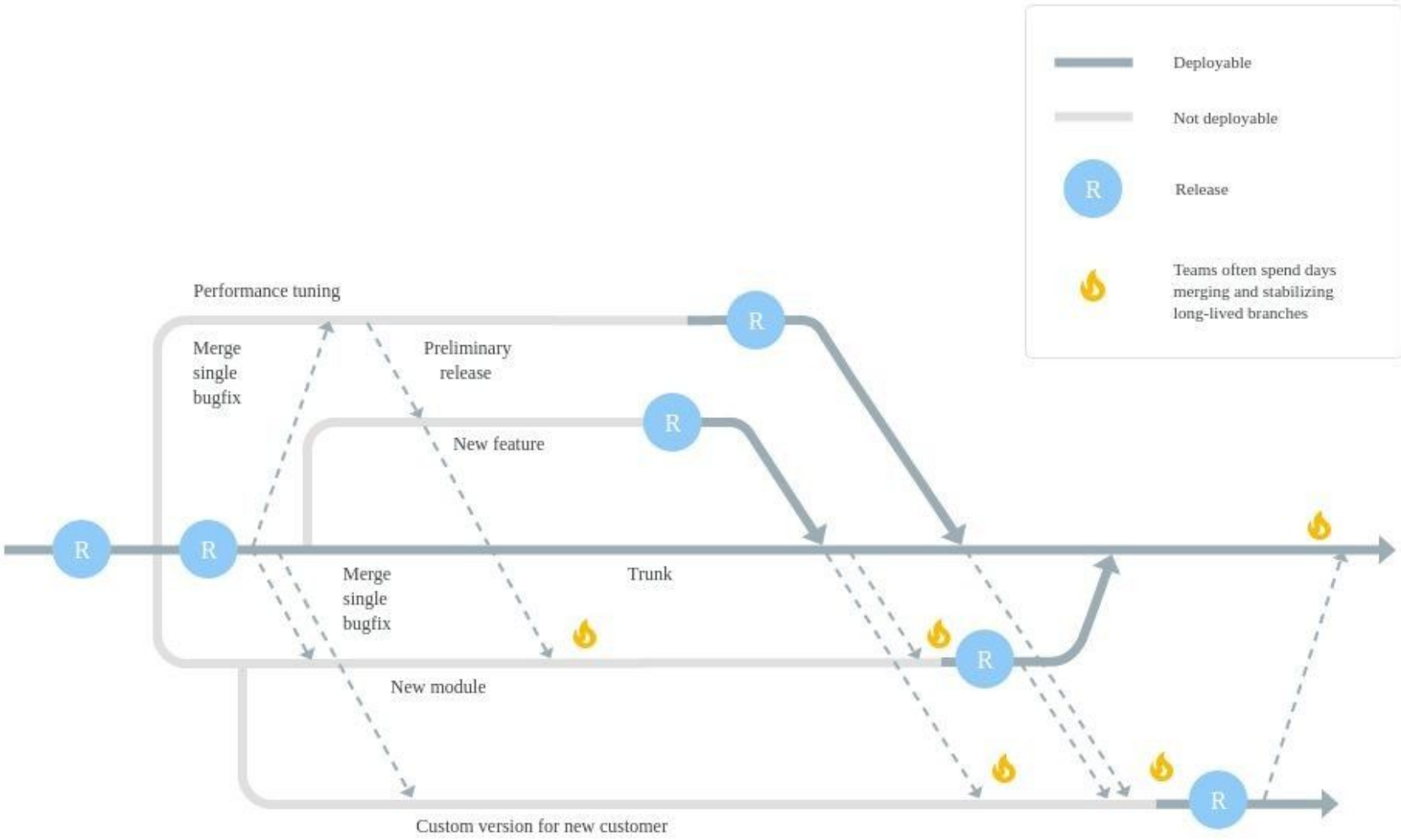
- *El trabajo se divide en dos branches principales:* ***Develop*** *y* ***Master****.*
- *Las nuevas funcionalidades se van implementando en los branches* ***Feature*** *que derivan del branch* ***Develop****.*
- *Cuando se quiere liberar código se utilizan los branches* ***Release****.*
- *Si existe un bug grave en producción se utilizan branches* ***Hotfix****.*


v0.1
v0.2
v1.0
Master
Hotfix
Release
Develop
Feature
Feature

Trunk Based Development:

Es una estrategia de Git

- Éxiste un branch principal (usualmente llamado **Master/Main**) en el cual todo el equipo colabora e integra directamente
- No existen branches de larga duración.
- No se le da mantenimiento a ningún branch.
- Se debe hacer **Commit** al menos una vez al día
- Todo lo que se le haga **Commit** es código funcional, esto implica que se cumpla la "definición de trabajo terminado" (**Definition Of Done**), se hayan creado las pruebas necesarias y todo lo que sea requerido para asegurarse que el código no está introduciendo un bug.
- El equipo debe de tener un grado de madurez y responsabilidad alto al momento de entregar el código.


Deployable
Not deployable
R
Release
Teams often spend days merging and stabilizing long-lived branches
Performance tuning
Merge single bugfix
Preliminary release
New feature
Merge single bugfix
Trunk
New module
Custom version for new customer

Calidad de Código

Mantener un cierto nivel de calidad y legibilidad de código es muy importante para lograr el éxito en ambientes de desarrollo dinámico, en el que múltiples equipos trabajan en un mismo código. Para que no se produzcan inconvenientes, es necesario que se sigan ciertos lineamientos que garanticen la calidad del producto final.

Existen tres procesos que deberían estar presentes en todos los procesos de desarrollo ágil:

- Análisis de la calidad del código
- Profiling de aplicaciones para detectar cuellos de botella y problemas de utilización de recursos
- Ejecución de tests de carga que ayudan a analizar la escalabilidad que presentan las aplicaciones, los servicios y la infraestructura

Calidad de Código / Lineamientos

- No tener líneas de código duplicado (“Duplicated Code”).
- Respetar los estándares de codificación y las mejores prácticas establecidas (“Coding Standards”).
- No tener una cobertura baja (o nula) de pruebas unitarias, especialmente en partes complejas del programa (“Unit Tests”).
- Evitar componentes complejos y/o una mala distribución de la complejidad entre los componentes (“Complex Code”).
- Analizar fallos potenciales (“Potential Bugs”).
- Comentar el código fuente, especialmente las APIs públicas (“Comments”).
- Evitar el temido diseño spaghetti, con multitud de dependencias cíclicas (“Design and Architecture”).

Calidad de Código / Comprobación

- **Code Coverage:** La cobertura código es la cantidad de código (medida porcentualmente) que está siendo cubierto por las pruebas unitarias (Unit Tests). Esto se logra recorriendo todas las pruebas y verificando que partes del código están propiamente cubiertas. Si el código consta de 100 líneas y solo 50 líneas están siendo ejecutadas al correr las pruebas, entonces la cobertura es del 50%.

Calidad de Código / Comprobación

- **Análisis Estático:** Es una proceso automático mediante el cual se revisa el código sin necesidad de ejecutarlo. En la mayoría de los casos implica revisar un conjunto de reglas en el código fuente.

Estas reglas podrían aplicarse a un rango muy amplio de aspectos:

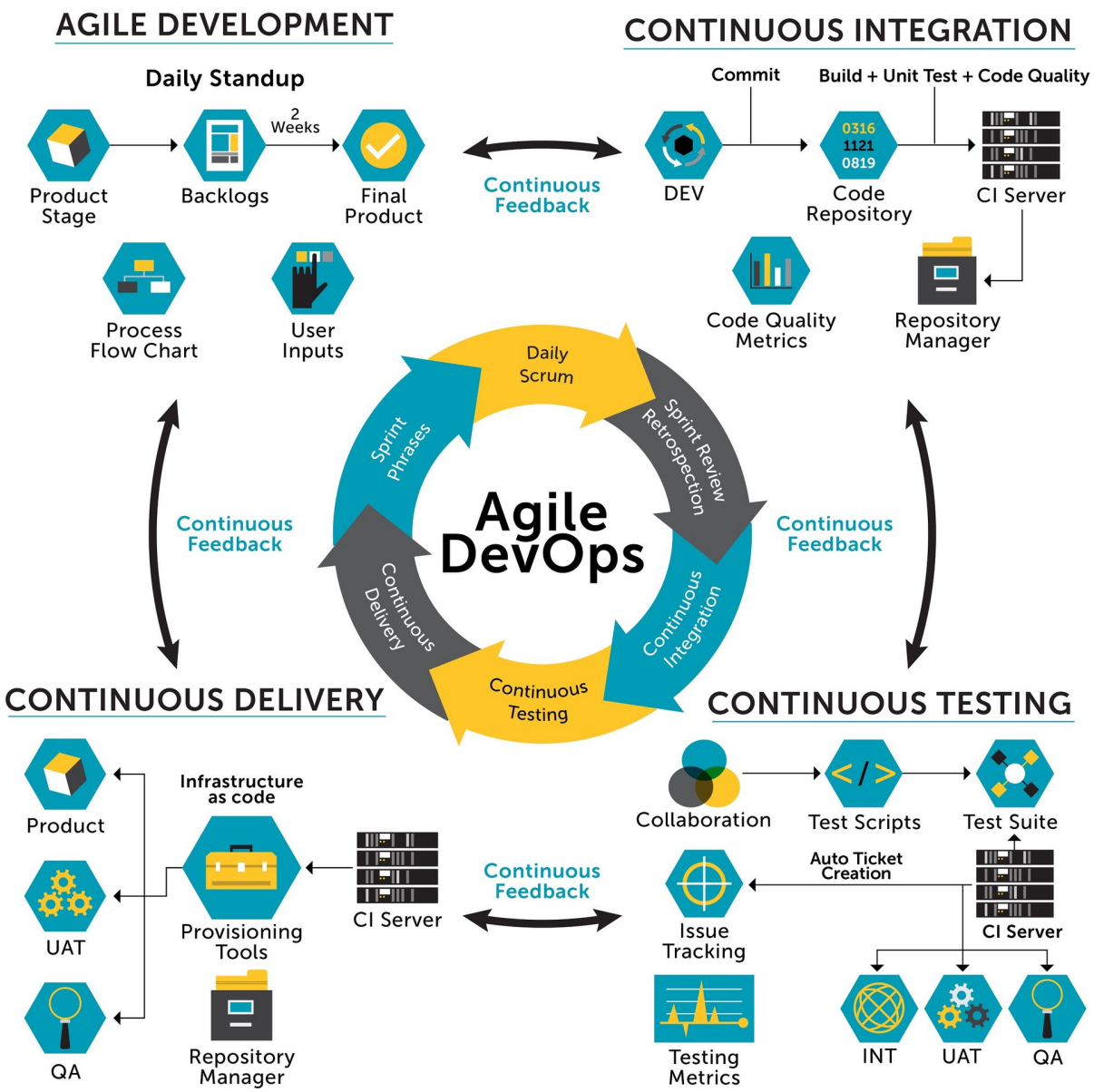
- Seguridad
- Errores de sintaxis
- Código poco intuitivo, difícil de mantener o sin documentación
- Uso de "malas prácticas"
- Estilos de código inconsistentes

Procesos de Build / ¿Que es?

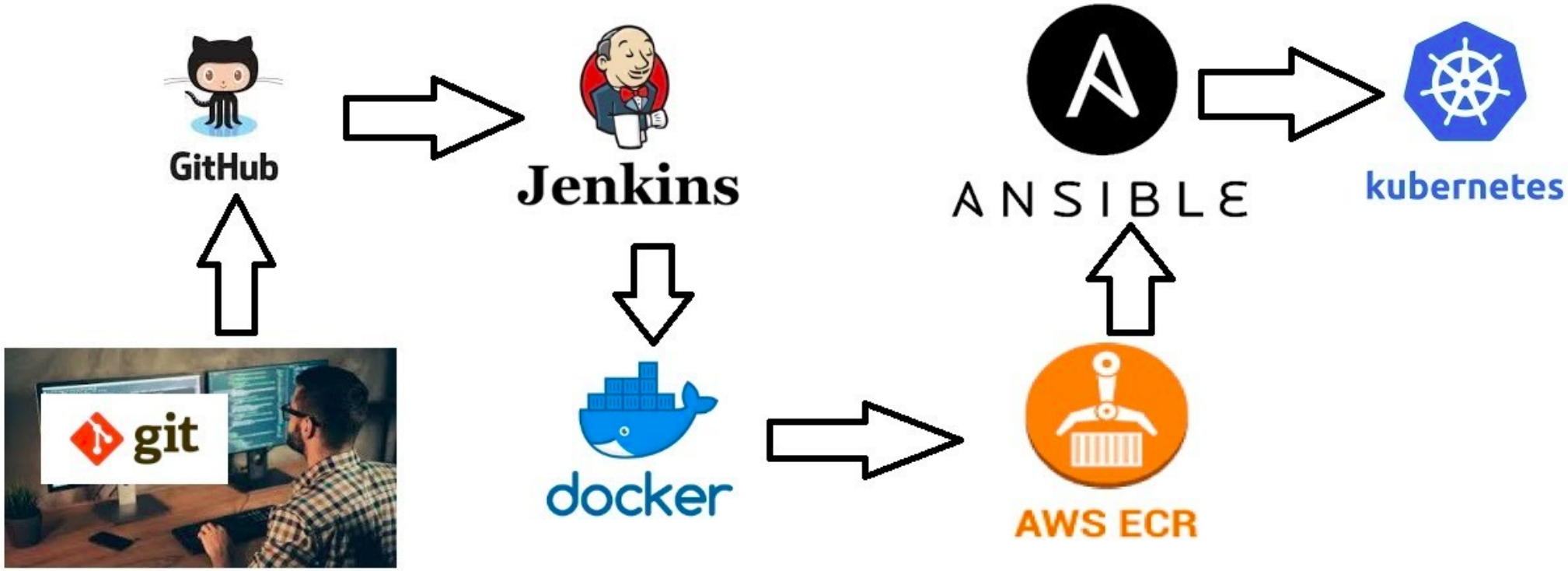
El proceso de Build es aquel por el cual de forma programable y automática se consolidan los cambios realizados en un proyecto, generando una nueva versión del producto. El resultado de este proceso es lo que se denomina “Deploy Unit”.

Procesos de Build / Objetivos

Las herramientas de Build tiene como objetivo automatizar de forma eficiente el proceso compilación, el cual generalmente consta de la generación y/o actualización de un conjunto de ficheros (Unidad de Deploy) que se construyen a partir de otros (Código Fuente).


AGILE DEVELOPMENT
Daily Standup
2 Weeks
Product Stage
Backlogs
Final Product
Process Flow Chart
User Inputs
Continuous Feedback
CONTINUOUS INTEGRATION
Commit
Build + Unit Test + Code Quality
DEV
0316
1121
0819
Code Repository
CI Server
Code Quality Metrics
Repository Manager
Daily Scrum
Sprint Review Retrospection
Continuous Integration
Continuous Testing
Continuous Delivery
Sprint Phrases
Agile DevOps
Continuous Feedback
Continuous Feedback
CONTINUOUS DELIVERY
Infrastructure as code
Product
UAT
QA
Provisioning Tools
Repository Manager
CI Server
Continuous Feedback
CONTINUOUS TESTING
Collaboration
Test Scripts
Test Suite
Auto Ticket Creation
Issue Tracking
Testing Metrics
CI Server
INT
UAT
QA

Proceso simple de Build



Procesos de Build / Fallas

- El código contiene errores de sintaxis
- No se encuentran o no se pueden descargar las dependencias necesarias
- La versión del compilador no es compatible con la versión del runtime
- La cobertura de las pruebas no cubre el porcentaje establecido
- El proceso de análisis estático no termina de manera correcta
- Fallan los Unit Tests (o test unitarios)

Procesos de Build / Paquetes y repositorios

¿Que es un paquete? (package)

Un Paquete es un contenedor para la unidad de deploy para que la misma pueda ser distribuida en los sistemas o instancias en la que deba ser instalada. También suele denominarse como Artefacto. Cada sistema de paquetes dispone de sus propias características y mecanismos de gestión.

¿Que es un repositorio de software?

Un repositorio de software es donde se almacenan paquetes que después van a ser promovidos y/o instalados en entornos o en instancias.

Procesos de Build / Versionamiento Semántico

Comúnmente llamado SemVer, es un estándar utilizado para de definir la versión un aplicativo, librería o API, dependiendo de la naturaleza del cambio que se está introduciendo.

Se identifican tres tipos

- **PATCH**
- **MINOR**
- **MAJOR**

Procesos de Build / Versionamiento Semántico

- Patch: Cuando se arregla un bug siendo el cambio retrocompatible.
- Minor: Cambio que añade alguna característica nueva al software o modifica alguna ya existente, pero que sigue siendo compatible con código existente. También cuando algo se marca como obsoleto.
- Major: Cambio drástico en el software. No es compatible con código hecho en versiones anteriores.

TAKE A BREAK!

y en 15´ volvemos.

UNC FCEFyN

mundosE PEOPLE & BUSINESS SCHOOL

¿Preguntas?

Build Tools / Java

MAVEN

Maven es una herramienta open-source, que se creó en 2001 con el objetivo de simplificar los procesos de build (compilar y generar ejecutables a partir del código fuente). De todas formas, Maven es mucho más que una herramienta que hace builds del código ya que es capaz de gestionar un proyecto software completo, desde la etapa en la que se comprueba que el código es correcto, hasta que se despliega la aplicación, pasando por la ejecución de pruebas y generación de informes y documentación.

GRADLE

Gradle, es una herramienta que permite la automatización de compilación de código abierto, la cual se encuentra centrada en la flexibilidad y el rendimiento. Los scripts de compilación de Gradle se escriben utilizando Groovy o Kotlin DSL (Domain Specific Language) en lugar de XML como se hacen en MAVEN

Build Tools / Node JS

NPM

Es el sistema de gestión de paquetes por defecto para NodeJS. Con Node Package Manager se puede correr tareas de testing, compilar en código JavaScript, administrar módulos, distribuir paquetes y agregar dependencias.

YARN

Yarn es un gestor de paquetes y de dependencias para NodeJS lanzado por Facebook en colaboración con Google. Introduce cambios en esa gestión de dependencias, en la ejecución de tareas y algunas mejoras de rendimiento, también en el cambio de enfoque en la descarga e instalación de los paquetes. Es compatible con el registro del npm.

Problemas con dependencias

- Networking
- Proxies
- Certificados SSL no válidos
- Autenticación
- Dependencias ciclicas
- Enlace de librerías
- Paquetes con ejecutables binarios

Artifacts

Librerías:
- **npm** (NodeJS)
- **NuGet** (.NET)
- **PIP** (Python)
- **Jar, Jad, Jmod** (Java)
- **Gem** (Ruby)

Imágenes:
- **Docker** Images
- **VMDK, VDI** (Virtual Machine)

Aplicativos:
- **EXE** (Windows)
- **War, Jar, Ear** (Java)
- **ELF** (Linux)

Paquetes:
- **DEB, RPM** (Linux)

Packaging Tools

Librerías:

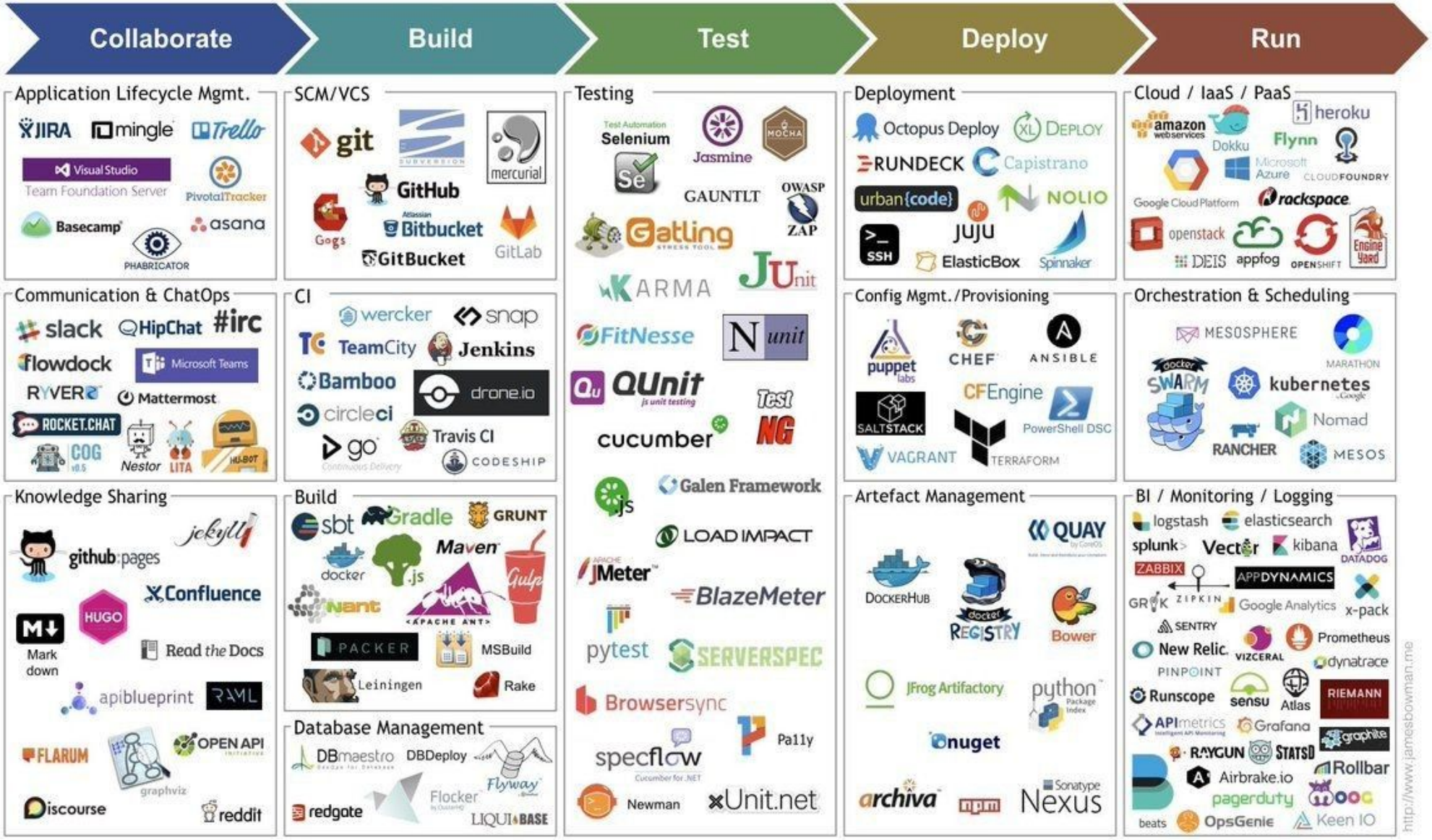
- **npm, yarn** (NodeJS)
- **Dotnet** (.NET)
- **Python**
- **Maven, Gradle** (Java)
- **Gem** (Ruby)

Imágenes:

- **Docker** CLI

Paquetes:

- **MAKE, DPKG** (Linux)



Dudas