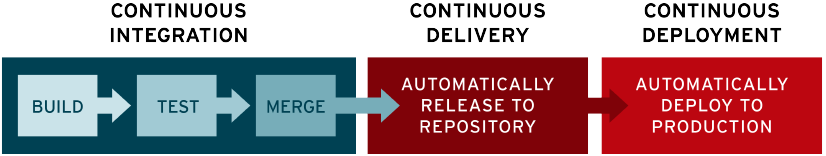
La CI/CD es un método para distribuir las aplicaciones a los clientes, mediante el uso de la automatización en las etapas del Desarrollo de aplicaciones. Se trata de una solución para los problemas que puede generar la [integración](https://www.redhat.com/es/topics/integration) del código nuevo para los equipos de desarrollo y de operaciones.

En concreto esto incorpora la automatizacion y la supervicion constante en todo el ciclo de vida de las aplicaciones, desde las etapas de integracion y prueba hasta las de distribución e implementación.



**CI** = integración continua ejecutadas a la hora de ejecutar un pull request (PR) son tareas de control y mantenimiento de la calidad del código.

**CD** = Delivery continuo: Son tareas que se centran en el deploy o despliegue de una aplicación una vez que un PR ha sido mezclado (mergeado// MEZCLADO) o se hizo un push.

**Pull request (PR):** También conocida como solicitud de fusión, es un evento que tiene lugar en el desarrollo de software cuando un colaborador/desarrollador está listo para comenzar el proceso de fusionar nuevos cambios de código con el repositorio principal del proyecto.

**TCP/IP vs OSI**

**OSI =** modelo de referencia (Aplicación, presentación, sesión, transporte, red, enlace, Fisica)

El modelo de Referencia **OSI** es una herramienta conceptual que se encarga de la **conexión entre sistemas** abiertos a la comunicación con otros sistemas. Los principios en los que basó su creación son, una mayor definición de las funciones de cada capa, evitar agrupar funciones diferentes en la misma capa y una mayor simplificación en el funcionamiento del modelo en general. **OSI no es un protocolo sino un modelo de referencia utilizado para comprender y diseñar la arquitectura del sistema.**

|  |  |
| --- | --- |
| Aplicación | Transferencia de archivos, intercambio de Mensajes. |
| Presentación | Representación y Formato de los datos. |
| Sesión | Organización y Sincronización del Intercambio de Datos. |
| Transporte | Canal de transferencia de mensaje de una aplicación a otra. |
| Red | Direcciones y mejor ruta. |
| Enlace | Acceso al medio. Detección de errores. Retransmisión. |
| Física | Transmisión binaria. Definiciones eléctrica y mecánicas del sistema físico |

**TCP/IP:**

Es más simple y está formado por menos capas. Este modelo es el implantado actualmente a nivel mundial. Su nombre deriva de los dos principales protocolos que lo conforman: TCP en la Capa de transporte e IP en la Capa de red. Se compone de 4 capas o 5 dependiendo de a quien le preguntes. Las capas de TCP/IP son: Aplicación (Reúne Aplicación, presentación y sesión del OSI), Transporte, Red, Enlace (estas 2 ultimas se pueden unir y ser llamada como una sola capa de Internet) y Física.

**TCP**: (Protocolo de Control de Transmisión) es un [protocolo](https://developer.mozilla.org/es/docs/Glossary/Protocol) de red importante que permite que dos anfitriones se conecten e intercambien flujos de datos. TCP garantiza la entrega de datos y [paquetes](https://developer.mozilla.org/en-US/docs/Glossary/Packet) en el mismo orden en que se enviaron.

**UDP**: es un protocolo **que permite la transmisión sin conexión de datagramas** en redes basadas en IP. Para obtener los servicios deseados en los hosts de destino, se basa en los puertos que están listados como uno de los campos principales en la cabecera UDP. Mientras que la transmisión en el protocolo TCP tiene lugar una vez se ha producido el enlace obligatorio de 3 vías (con acuse de recibo mutuo entre el emisor y el receptor, incluida la sesión de comunicación), el protocolo UDP no utiliza este procedimiento con el fin de mantener el tiempo de transmisión lo más bajo posible.

**Encabezados de capa** = PDU (Protocolo de Unidad de Datos)

**Protocolos de correo (en aplicación**): smtp-pop-imap

**Protocolos de Aplicación**: http, ftp, dns, telnet

**Protocolos de transporte**: UDP, TCP

**Protocolos de red**: IP

**HTTP (Protocolo de transferencia de hipertexto):** protocolo el cual nos permite realizar una petición de datos y recursos, como pueden ser documentos [HTML](https://developer.mozilla.org/es/docs/Glossary/HTML). Es la base de cualquier intercambio de datos en la Web, y un protocolo de estructura cliente-servidor, esto quiere decir que una petición de datos es iniciado por el elemento que recibirá los datos (el cliente), normalmente a un navegador Web.

Clientes y servidores se comunican intercambiando mensajes individuales. Los mensajes que envía el cliente, normalmente a un navegador Web, se llaman ***peticiones***, y los mensajes enviados por el servidor se llaman ***respuestas***.

Es un protocolo para transmitir información/ texto entre el host y el usuario de forma ordenada. No es el más seguro y confiable.

**DNS (Sistema de nombres de dominio):** Es un protocolo que se encarga de vincular los nombres de los sitios web que visitamos con su correspondiente dirección IP. Básicamente lo que hace es traducir lo que ponemos. Por ejemplo si entramos en RedesZone, va a traducir el nombre de dominio redeszone.net por la dirección IP que corresponda y nos mostrará el contenido en el navegador.

Esto permite que los usuarios no tengamos que memorizar números sin mucho sentido. Simplemente hay que saber cuál es el nombre del sitio y automáticamente los **servidores DNS** van a vincularlos a la dirección IP y poder navegar por ellos.

**Métodos POST y GET**: Hace peticiones a través de la URL y header, le informa del estado a través de los códigos de estados (200, 502, 404, etc.).

**POST**: El método POST introduce los parámetros **en la solicitud HTTP**para el servidor. Por ello, no quedan visibles para el usuario.

**GET**: solicita una representación del recurso especificado. Las solicitudes usando GET solo deben usarse para solicitar datos.

**Github Actions:** es una herramienta que permite reducir la cadena de acciones necesaria para la ejecución de código, mediante la creación de un de flujo de trabajo encargado del Pipeline. Siendo configurable para que GitHub reaccione a ciertos eventos de forma automática según nuestras preferencias.

Por lo tanto, GitHub Actions permite crear workflows que se puedan utilizar para compilar, testear y desplegar código. Además, da la posibilidad de crear flujos de integración y despliegue continuo dentro de nuestro repositorio.

**Workflow:** Es un procedimiento automatizado compuesto por uno o varios trabajos que se añade a un repositorio y puede ser activado por un evento. Se definen mediante archivos YAML y con él se puede hacer el build, test, package, reléase o deploy de un proyecto.