

DevOps en Base de Datos

Estrella Palacios, Katherine Lizbeth (2016056193)), Gonzales Cave, Angel Gabriel (xxxxxxxxxx)), Huichi Contreras, Franklin Carlos (2016054948)), Huillca Umpiri, Willian Arturo (xxxxxxxxxx))

Escuela Profesional de Ingeniería de Sistemas

Universidad Privada de Tacna

Tacna, Perú

Abstract

Devops is described as a cooperative and productive relationship between development teams and operating teams. That is why DevOps manages to establish an effective relationship of efficiency, collaboration and development strategies among employees. DevOps also adopts new development methodologies such as agile development, operations, quality control and implementation in order to create integration and collaboration between the two departments. In conclusion, communication and collaboration between the departments of developers and operations make this methodology the best option to resort to an orderly and well-structured work model.

1. Resumen

Devops es descrito como una relación cooperativa y productiva entre los equipos de desarrollo y los equipos de operación. Es por ello que DevOps logra establecer una relación eficaz, de eficiencia, colaboración y estrategias de desarrollo entre los colaboradores. DevOps también adopta nuevas metodologías de desarrollo como el desarrollo ágil, operaciones, control de calidad e implementación con el fin de crear integración y colaboración entre los dos departamentos. En conclusión, la comunicación y colaboración entre los departamentos de desarrolladores y operaciones hacen de esta metodología la mejor opción para recurrir a un modelo de trabajo ordenado y bien estructurado.

2. Introducción

Con la llegada de las metodologías ágiles de desarrollo y las necesidades de realizar integración y entrega continua (CI, continuous integration y CD, continuous delivery) aparece una nueva corriente organizativa llamada DevOps, que, en resumidas cuentas pretende aunar en un único equipo a perfiles muy separados en organizaciones más tradicionales como puedan ser los desarrolladores y los equipos de operaciones, todo ello con el objetivo final de realizar despliegues en entornos productivos de forma más regular. Haciendo nuevas entregas del software de una forma regular (semanalmente, diariamente o, incluso, varias veces al día) se consigue dotar al proceso del paso a producción de más seguridad o estabilidad y más eficiencia. Cuanto más regularmente se haga una tarea, en este caso un despliegue en producción, menos «doloroso» será. Para ello se requiere un nivel muy alto en la automatización de los procesos de compilación, empaquetado, pruebas, despliegues, smoke tests, etc.

3. Marco Teórico

3.1. *DevOps*

3.1.1. *Definición*

DevOps fue acuñado en 2009 por Patrick Debois, que se ha convertido desde entonces en uno de los gurús dentro de la comunidad. El término se conforma de combinar las palabras "desarrollo" y "operaciones, del inglés "Development & Operations", y puede servir como punto de partida para entender qué significa exactamente el término DevOps. Esta nueva cultura no es un proceso, una tecnología concreta o un estándar, sino un conjunto de técnicas, pensamientos, y modelos de trabajo. También se utiliza el término "movimiento DevOps" cuando se habla de temas acerca de la adopción de nuevos ratios e indicadores y tendencias de futuro y "entorno DevOps" para referirse a la estrategia organizativa que sugiere la cultura DevOps. [8]

DevOps representa un cambio en la cultura de TI, centrándose en la entrega rápida de servicios de TI a través de la adopción de prácticas ágiles y esbeltas en el contexto de un enfoque orientado al sistema. DevOps hace hincapié en las personas (y la cultura) y busca mejorar la colaboración entre las operaciones y los equipos de desarrollo. Las implementaciones de DevOps utilizan tecnología, especialmente herramientas de automatización que pueden aprovechar una infraestructura cada vez más programable y dinámica desde una perspectiva del ciclo de vida [1]

3.1.2. *Beneficios*

En una encuesta reciente, realizada a 1.770 altos ejecutivos de TI a nivel mundial, por el instituto Technologies y el Coleman Parks Research, han llegado a la conclusión de que existen al menos 8 beneficios que se pueden obtener de la implementación de las DevOps en el ámbito empresarial.

Los resultados a los que estas dos empresas llegaron luego de su estudio, son los siguientes: [7]

- Más plazas de trabajo
- La buena experiencia del cliente
- Un cliente feliz es una empresa exitosa
- Empleados más productivos
- Calidad de la aplicación
- Eficiencia de los procesos
- El crecimiento de nuevos negocios
- Reducción de costos en las TI

3.1.3. *Principios*

DevOps maneja principios que son parte de la estructura colaborativa y son utilizados en toda la etapa de desarrollo y despliegue de aplicaciones.

Los principios en los que se desenvuelve DevOps son los siguientes: [3] [2]

- **Integración Continua** La Integración Continua es la forma en la que el equipo de desarrollo de software integra su trabajo parcial o total, en un determinado tiempo establecido por el equipo de trabajo. Requiere de herramientas de automatización que son únicas para todo el equipo de desarrolladores. Estas herramientas ayudan a integrar en forma continua partes de código que son validados por pruebas automáticas, lo cual vuelve más eficiente el trabajo del equipo de desarrollo, ya que permite detectar fallos en etapas tempranas del ciclo de desarrollo. La Integración continua de cada uno de los integrantes del equipo va a favorecer en los tiempos y calidad del producto. La automatización de la integración continua ayuda a descubrir errores, debido a las continuas pruebas de unidad automatizadas.

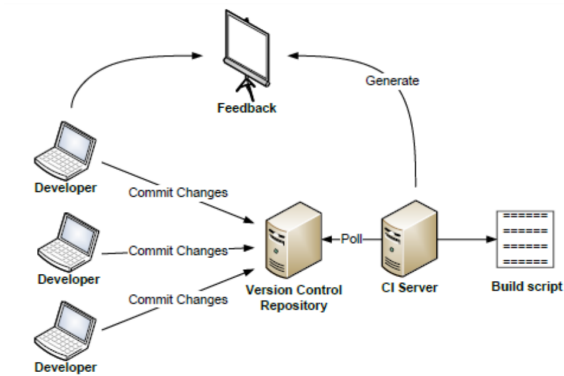


Figura 1: Proceso de Integración Continua

- **Entrega Continua** Una vez lograda la integración, se debe continuar con el ciclo para realizar el testing que, si es satisfactorio, permita que la aplicación sea desplegada. En esta etapa se implementan todos los cambios en el código y se somete a un proceso de pruebas estandarizado con el fin de generar un producto que pueda ser pasado a producción.

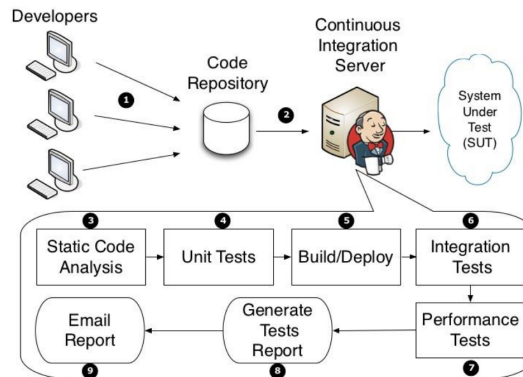


Figura 2: Proceso de Entrega Continua

- **Despliegue Continuo** El despliegue continuo es una práctica que permite llevar los resultados de un proceso de desarrollo a un entorno similar al de producción donde las pruebas funcionales puedan darse a escala completa. El objetivo es detectar problemas en producción lo más rápido posible.

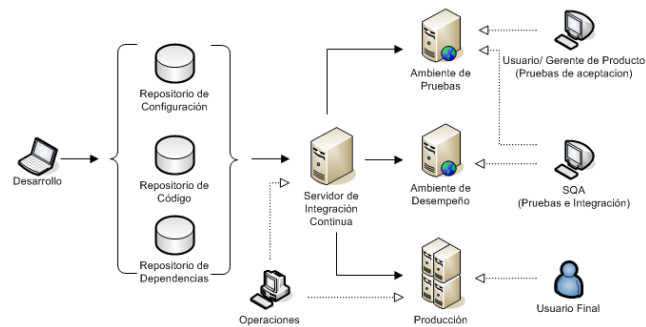


Figura 3: Proceso de Despliegue Continuo

3.1.4. *Ámbito de aplicación de DevOps*

La mayoría de estas empresas organizan la estructura en dos grandes ámbitos, el de desarrollo de software y el de operaciones. A su vez, estos dos ámbitos tienen sus propios departamentos, los cuales pueden variar en función de las necesidades de cada empresa. Algunos de ellos pueden aparecer fusionados, o en ocasiones prescindir de ellos dado que el volumen de negocio no lo requiere. Un ejemplo de los distintos departamentos que puede haber en una empresa es que se muestra en la siguiente figura: [4]

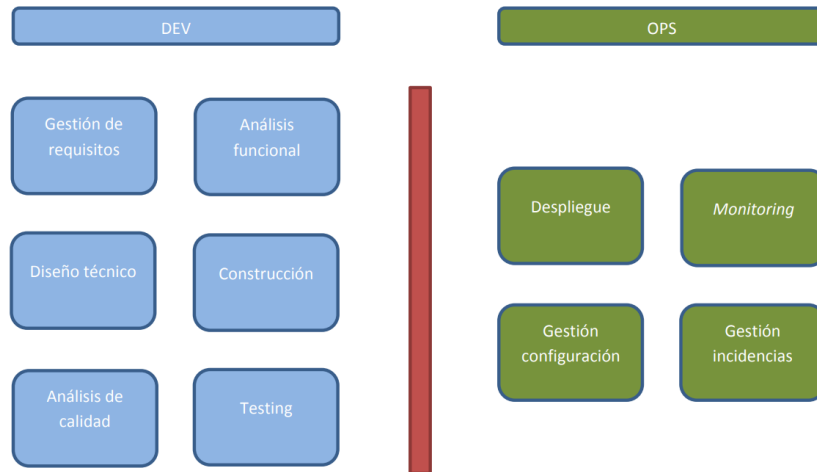


Figura 4: Ciclo de Vida del DevOps

3.1.5. *Ciclo de Vida y Herramientas de DevOps*

Una cadena de herramientas es un conjunto de herramientas de programación que se pueden utilizar para llevar a cabo una tarea compleja de desarrollo de software o para crear un producto de software, que comúnmente es algún otro programa informático o un conjunto de programas relacionados. En el campo DevOps, son recursos útiles en la entrega, desarrollo y administración de aplicaciones durante el ciclo de vida de desarrollo de software, según lo coordinado mediante el uso de una organización que utiliza las prácticas DevOps.[6] El ciclo de vida de DevOps consta de las siguientes etapas: Planificar, Crear, Verificar, empaquetar, liberar, configurar y monitorear. [5]

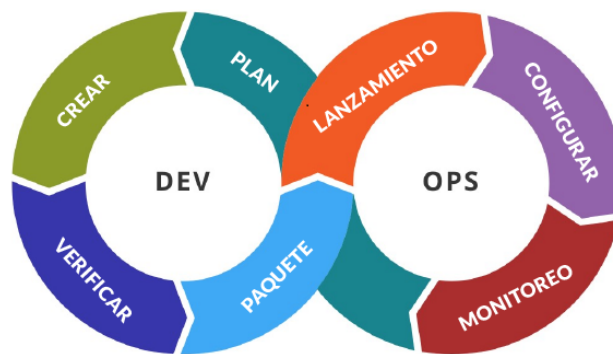


Figura 5: Ciclo de Vida de DevOps

- **Plan** Sin importar la metodología que uses como Waterfall o Agile, en esta primera etapa definimos las labores del equipo, los requerimientos necesarios a implementar en la plataforma o producto. Podemos hacer usos de herramientas como issues para hacer un monitoreo de nuestro progreso, así como boards como Trello o Asana.
- **Create** Empezamos a escribir el código que necesitamos para resolver los problemas que planteamos en el paso anterior. Todo este código puede estar almacenado en un solo lugar como un repositorio de Github o Gitlab para hacer uso de herramientas proporcionadas por estas páginas y por git en general como branches, tags y mucho más.
- **Verify** Como somos desarrolladores increíbles, hemos escritos diferentes pruebas de software para disminuir la cantidad de bugs que podemos

agregar a nuestro producto una vez estemos en producción, también es una manera de descubrir errores. Estas pruebas son definidas con antelación. Se pueden usar varias herramientas de Continuous Integration como TravisCI, CircleCI o Jenkins

- **Package** Empaquetamos nuestro código para correr en una infraestructura determinada. Esto puede hacerse incluso desde la etapa de creación donde escribimos nuestro código de una forma empaquetada y lista para que sea llevada a producción para no tener complicaciones y dolores de cabeza en el futuro. Es muy común realizarlo en contenedores de Docker.
- **Release** Automatizamos el proceso de enviar el código a producción, liberar una nueva versión de nuestro producto cada que construyamos un feature nuevo o resolvamos un bug mientras haya pasado por las etapas anteriores. No es más que una nueva versión de nuestro código disponible para hacerlo de manera manual o automática.
- **Monitor** Necesitamos revisar las métricas necesarias de cómo nuestro código está funcionando, qué tipo de performance ocurre en los dispositivos de nuestros clientes para tratar de optimizar siempre que sea posible y mejorar nuestro producto. Al ser un modelo iterativo es importante estar siempre en este constante proceso, no podemos dar nunca por terminado nuestro producto, esa es la manera en la que las empresas mueren.

3.2. *DevOps en Base de Datos*

3.2.1. *B.1*

EDITAR

4. **Análisis**

EDITAR

5. **Conclusiones**

- Conclusion 1 :
A

- Conclusion 2 :
B
- Conclusion 3 :
C

Referencias

- [1] Gartner (ne). It glossary. Recuperado de <https://www.gartner.com/it-glossary/devops>. Accedido el 28-08-2019.
- [2] Hüttermann, M. (2012). *DevOps for Developers*. Apress.
- [3] Humble, J. y Farley, D. (2011). *Continuous Delivery*. Editorial Addison Wesley.
- [4] Jiménez, G. (2016). Devops, la nueva tendencia en el desarrollo de sistemas ti, un caso práctico en el análisis de incidencias de software. Master's thesis, Universitat Politècnica de Catalunya, España. Accedido el 29-08-2019.
- [5] Platzi (2019). Ciclo de vida del devops. Recuperado de <https://platzi.com/blog/ciclo-de-vida-del-devops/>. Accedido el 28-08-2019.
- [6] Pérez, L. (2018). Devops: It development in the era of digitalization. Master's thesis, Universidad de Valladolid, España. Accedido el 29-08-2019.
- [7] TicNews (2017). 8 beneficios de adoptar las devops en tu empresa. Recuperado de <https://ticnews.net/8-beneficios-de-adoptar-las-devops-en-tu-empresa/>. Accedido el 28-08-2019.
- [8] Walls, M. (2013). *Building a DevOps Culture*. Sebastopol: O'Reilly.