DevOps: valores, principios, beneficios y metas

Definicion: DevOps is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality

Valores:

- **Feedback from stakeholders is essential:** La retroalimentación continua de las partes interesadas garantiza mejoras alineadas con las necesidades del negocio
- **Improve beyond the limits of today's processes:** La mejora continua, la automatización y el uso de herramientas adecuadas garantizan la satisfacción de los clientes y stakeholders
- **No new silos to break down silos:** Crecer como equipo implica evitar divisiones y fomentar la cooperación, en lugar de crear nuevos problemas o aislar grupos.
- **Knowing your customer means cross-organization collaboration:** Comprender a los clientes requiere colaboración entre equipos para satisfacer sus necesidades y fortalecer el negocio
- **Inspire adoption through enthusiasm:** Promover una cultura de aprendizaje y comunicación activa impulsa la adopción de nuevas tecnologías y metodologías.

Principios:

- **Collaboration**
- **Automation**
- **Continuous Improvement**

Adicionales:
• **Customer-centric action:** Desarrollar soluciones con enfoque en las necesidades y expectativas del cliente.
• **Create with the end in mind:** Diseñar procesos y soluciones considerando su impacto y sostenibilidad a largo plazo.

Beneficios y metas

- **Accelerating time to market:** Reducir el tiempo de entrega de software y actualizaciones para responder rápidamente a las necesidades del mercado
- **Adapting to market dynamics and competition:** Fomentar un desarrollo ágil para ajustarse rápidamente a cambios y presiones competitivas**.**
- **Sustaining stability and reliability in operations:** Promover la colaboración entre equipos para minimizar interrupciones y mejorar la estabilidad operativa
- **Minimizing the mean time to recovery (MTTR):** Automatizar y monitorear continuamente para detectar y resolver problemas más rápido, reduciendo el impacto de fallos.

The Three Ways:

- **The First Way: Flow/Systems Thinking ->** Optimizar el flujo de trabajo de desarrollo a producción, priorizando la entrega rápida y eficiente sin generar cuellos de botella ni transmitir defectos (optimizations of the system, but global, not local)

- **The Second Way: Amplify Feedback Loops ->** Recoger y responder rápidamente a la retroalimentación en cada etapa del proceso para mejorar continuamente la calidad y la experiencia del cliente.

- **The Third Way: Culture of Continual experimentation and Learning ->** Fomentar la experimentación, aceptar el fracaso como aprendizaje y reforzar la resiliencia del sistema a través de pruebas y simulaciones constantes.

DevOps: formas de coordinación

- **Direct:** los individuos que coordinan se conocen entre sí (por ejemplo, miembros de un equipo).
- **Indirect:** el mecanismo de coordinación está dirigido a una audiencia conocida solo por su caracterización (por ejemplo, administradores de sistemas).
- **Persistent:** los artefactos de coordinación están disponibles después del momento de la coordinación (por ejemplo, documentos, correos electrónicos, tablones de anuncios).
- **Ephemeral:** la coordinación, en sí misma, no produce artefactos (por ejemplo, reuniones cara a cara, conversaciones, conferencias telefónicas o por video). La coordinación efímera puede volverse persistente mediante el uso de grabadores humanos o mecánicos.
- **Synchronous:** los individuos se coordinan en tiempo real (por ejemplo, cara a cara).
- **Asynchronous:** los individuos no se coordinan en tiempo real (por ejemplo, documentos, correos electrónicos).

Devops roles

- **Team lead:** Facilita el trabajo del equipo, gestiona recursos y lo protege de obstáculos, enfocándose en habilidades blandas de gestión sin intervenir en la planificación técnica
- **Team member**: Responsable de la creación y entrega del sistema, incluyendo modelado, programación, pruebas y despliegue
- **Service owner**: Responsable de la coordinación externa, recopilación de requisitos y priorización de tareas. Mantiene y comunica la visión del servicio, asegurando la alineación con clientes y otros servicios. Su rol implica entender la arquitectura del sistema y facilitar la comunicación con stakeholders y el equipo
- **Reliability Engineer**: Monitorea el servicio después del despliegue, detecta problemas y garantiza su disponibilidad. Es el punto de contacto en incidentes críticos y realiza análisis a corto plazo para diagnosticar y mitigar fallos. También investiga la causa raíz de los problemas usando técnicas como los "5 Whys". Además, debe ser un buen desarrollador para automatizar tareas de diagnóstico y reparación.
- **Gatekeeper**: Es el responsable de decidir si una versión o parte de un servicio puede avanzar en la tubería de despliegue. Puede basarse en pruebas automatizadas, checklists y consultas con otros, pero la decisión final recae en él. En algunas industrias reguladas, un gatekeeper humano es obligatorio.
- **DevOps Engineer**: Es responsable de gestionar y optimizar las herramientas de la cadena DevOps, incluyendo pruebas de código, integración continua, despliegue y

post-despliegue. También se encarga de la gestión de configuración, asegurando el control de cambios y la correcta implementación de estrategias de ramificación. Su rol es clave en la automatización del desarrollo y despliegue, y puede operar a nivel individual, de equipo u organizacional.

Types of DeVops Team configuration

- **Centralized DevOps team:** Se encargan de todo, no hay roles definidos
- **Embedded DevOps Team:** Dentro del equipo de desarrollo pero encargado de los aspectos devops
- **Distributed DevOps Team:** En organizaciones grandes para no duplicar de trabajos de herramientas y procesos devops
- **Specialized DevOps team:** Enfocado en un área específica del proceso de entrega, como pruebas o gestión de infraestructura.
- **Standalone DevOps Teams:** Equipos independientes que promueven y estandarizan las prácticas DevOps en toda la organización.
- **External DevOps Teams or Consultants:** Expertos externos que asesoran y capacitan a empresas sin experiencia interna en DevOps.
- **Site Reliability Engineering (SRE) Teams:** Especialistas en la confiabilidad y rendimiento del sistema, automatizando operaciones y asegurando estabilidad

Prácticas en DevOps desde diferentes referentes

Key practices by Atlassian: (1) Atlassian enfatiza la integración continua y la entrega continua (CI/CD), lo que permite a los equipos automatizar la compilación, prueba y despliegue del software para reducir errores y acelerar las entregas. (2) También promueve la infraestructura como código (IaC), facilitando la gestión y provisión de entornos de desarrollo mediante código, lo que mejora la consistencia y escalabilidad. (3) La automatización de flujos de trabajo es clave, asegurando que procesos repetitivos sean manejados por herramientas para minimizar errores humanos y mejorar la eficiencia. (4) Además, Atlassian destaca la observabilidad y monitoreo, proporcionando visibilidad en tiempo real del rendimiento del software y ayudando a la detección temprana de problemas.

Practices by Microsoft: (1) Microsoft impulsa la implementación de pipelines de CI/CD altamente automatizados para garantizar despliegues rápidos y seguros en la nube. (2) La gestión de la configuración y seguridad mediante IaC es otra práctica esencial, permitiendo que los equipos definan entornos de manera programática con controles de seguridad integrados. (3) También promueve la adopción de DevSecOps, integrando la seguridad en cada fase del desarrollo para identificar vulnerabilidades antes de la producción. (4) Microsoft enfatiza el uso de métricas y telemetría para mejorar el rendimiento del software y la experiencia del usuario mediante la recopilación y análisis continuo de datos operativos.

Practices by dev.to: (1) En la comunidad de dev.to, una práctica clave es la cultura de colaboración y comunicación entre equipos de desarrollo y operaciones, fomentando la transparencia y el trabajo conjunto para mejorar la eficiencia del ciclo de vida del software. (2) Se enfatiza el uso de pruebas automatizadas en todas las etapas del desarrollo, asegurando que el código sea fiable antes de su implementación en producción. (3) Además, se impulsa la adopción de contenedores y orquestación, facilitando la portabilidad y escalabilidad de las aplicaciones. (4) Finalmente, se destaca la mejora continua basada en retroalimentación constante, lo que permite a los equipos iterar rápidamente y optimizar procesos con base en datos y experiencias previas.

Ciclo de vida DevOps

Continuous Development -> Continuous Integration -> Continuous Testing -> Continuous Deployment -> Continuous Monitoring -> Continuous Feedback -> Continuous Operations

Adopción de DevOps: negocio, personas, proceso y tecnología

Negocio: Para adoptar DevOps, es crucial alinear a todos con los mismos objetivos empresariales, asegurando que los incentivos sean compartidos y no generen conflictos entre equipos.

People:

- **DevOps Culture:** DevOps es una transformación cultural que fomenta la colaboración entre equipos, eliminando barreras y promoviendo la responsabilidad compartida en la entrega de software. Se basa en la confianza, la visibilidad y el

aprendizaje continuo para optimizar la entrega de valor al negocio. (no se puede medir la calidad facilmente)

- **Devops team:** La clave de un equipo DevOps es facilitar la colaboración y la excelencia sin convertirse en una barrera burocrática o en un único responsable de los problemas DevOps. Puede integrarse como un equipo especializado o fusionarse con desarrollo y operaciones, según la cultura de la organización.

Process:

- **Change management process**:: Es el conjunto de actividades que controlan y rastrean cambios en el desarrollo, asegurando que todas las modificaciones sean gestionadas de manera eficiente
- **Devops techniques**: Continuous improvement » Release planning » Continuous integration » Continuous delivery » Continuous testing » Continuous monitoring and feedback

Tecnology:

- **Infrastructure as Code**: Permite gestionar y aprovisionar infraestructuras mediante código, asegurando rapidez y consistencia en los entornos de desarrollo. Se basa en herramientas especializadas, de despliegue y genéricas para automatizar configuraciones y facilitar la entrega continua
- **Delivery Pipeline**: Es el conjunto de etapas que atraviesa una aplicación desde el desarrollo hasta la producción, con distintos niveles de automatización (Development environment > Build stage > Package repository > Test environment > Stage and production environments)
- **Deployment Automation & Release Management**: La automatización de despliegue gestiona la implementación coordinada de software, middleware y configuraciones en cada etapa del pipeline. La gestión de releases coordina los lanzamientos entre equipos, asegurando trazabilidad y colaboración.

Cloud: caracterización, modelos de servicio, características y evolución

Caracterizacion:

- **On-demand self-service**: Permite a los usuarios aprovisionar recursos computacionales automáticamente sin intervención humana.
- **Broad network access**: Los servicios están disponibles en la red y accesibles desde distintos dispositivos mediante protocolos estándar.
- **Resource pooling**: Recursos compartidos entre múltiples usuarios bajo un modelo multi-tenant, con asignación dinámica según demanda.
- **Rapid elasticity**: Escalabilidad automática de recursos según la necesidad, aparentando disponibilidad ilimitada.
- **Measured service**: Monitoreo y optimización automática del uso de recursos, proporcionando transparencia en el consumo.

Modelos de servicio

- **Software as a Service (SaaS):** Proporciona aplicaciones listas para usar a través de la nube, accesibles desde navegadores o interfaces. El usuario no gestiona la infraestructura ni el software, solo ciertas configuraciones.
- **Platform as a Service (PaaS):** Permite desplegar aplicaciones en una infraestructura cloud sin preocuparse por servidores, redes o almacenamiento, pero con control sobre las aplicaciones y su configuración.
- **Infrastructure as a Service (IaaS):** Brinda acceso a recursos computacionales básicos como servidores, almacenamiento y redes, permitiendo instalar sistemas operativos y software, con control parcial sobre la infraestructura.

Características:

- **Virtualization**: Permite crear y cargar máquinas virtuales o contenedores para optimizar el uso de recursos.
- **IP y DNS**: Facilitan la identificación y acceso a servicios en la nube mediante direcciones IP y nombres de dominio.
- **Distributed environment**: Infraestructura distribuida que mejora escalabilidad, disponibilidad y tolerancia a fallos.
- **Time and failures**: Gestiona sincronización y recuperación ante fallos para garantizar continuidad del servicio.
- **Consistency**: Asegura que los datos sean coherentes en múltiples nodos dentro de sistemas distribuidos.

Evolucion:
https://www.icesi.edu.co/moodle/pluginfile.php/1125856/mod_resource/content/2/Dev-Ops%20–%20cloud%20ed.pdf

Bash advantages: Automation • Portability • Flexibility • Accessibility • Integration • Debugging

Bash

- **Ventajas**: Sencillo, flexible y ampliamente compatible.
- **Usos**: Automatización de tareas, gestión de sistemas y scripting en servidores.
- **Características**: Interpreta comandos, maneja variables y permite programación estructurada.

Jenkins

- **Ventajas**: Extensible, automatiza CI/CD y tiene una gran comunidad.
- **Usos**: Integración y entrega continua, ejecución de pruebas y despliegues automatizados.
- **Características**: Basado en Java, usa pipelines y permite integración con múltiples herramientas.