TERCERA ENTREGA CURSO .NET CORE

YEISER SMITH MERCADO CORDERO

Tutor

FELIPE SALGADO

CETIA – EXPERTMEDICA

CURSO NET CORE Y ANGULAR

MEDELLIN – COLOMBIA

2024

1. Como implentaria CI/CD en su dia a dia

Respuesta:

La implementación de CI/CD es fundamental en mi trabajo diario, ya que me permite automatizar procesos de desarrollo, pruebas y despliegue, lo que garantiza entregas rápidas, fiables y con calidad. Soy responsable de diseñar y mantener una infraestructura que soporte estas prácticas, asegurando que los desarrolladores puedan integrar su código de manera fluida y que las entregas de software sean consistentes y rápidas.

A continuación describo las tareas que creo debería realizar para implementar CI/CD en mi trabajo.

- Diseño de la infraestructura de CI/CD: Diseñar la infraestructura de CI/CD que soporte las necesidades del proyecto, teniendo en cuenta las siguientes decisiones:

  Elección de las herramientas: Herramientas como Jenkins, GitLab CI, Travis CI, CircleCI, GitHub Actions para integración continua.
  Y Herramientas como Docker, Kubernetes, Terraform, Ansible, AWS CodePipeline, Azure DevOps para entrega y despliegue continuo.

  Definir el pipeline: Un pipeline básico de CI/CD generalmente incluye:

  Compilación: Construcción del código y generación de artefactos.

  Pruebas unitarias y de integración: Ejecución de pruebas automáticas para validar que los cambios no introduzcan errores.

  Generación de artefactos: Empaquetado del código (Docker, ZIP, etc.).

  Despliegue a entornos de prueba: Despliegue automático en un entorno de testing o staging para validaciones adicionales.

  Despliegue a producción: Si se adopta CD completo, el despliegue en producción es automático; de lo contrario, se realiza manualmente.

- Automatización de las pruebas: Definir cómo se integrarán las pruebas automatizadas en el proceso de CI. En un entorno bien diseñado, el pipeline de CI debe incluir:

  Pruebas unitarias: Cada vez que los desarrolladores hacen un commit, las pruebas unitarias se ejecutan automáticamente.

  Pruebas de integración: Garantizan que las diferentes partes del sistema funcionen bien juntas.

Pruebas de aceptación: Validan si las funcionalidades cumplen con los requisitos establecidos.

Pruebas de regresión: Aseguran que nuevas funcionalidades no rompan las existentes.

- Integración de control de versiones (Git): Definir las políticas de control de versiones (Git) para asegurar que el flujo de trabajo sea compatible con CI/CD:

  Uso de ramas (branching): Git Flow: Ramas para desarrollo (develop), features y hotfixes.

  Trunk-based development: Un enfoque más moderno donde el desarrollo ocurre en una rama principal, con integración frecuente para evitar divergencias.

  Configurar el pipeline de CI para ejecutarse automáticamente en cada push o merge a ramas específicas, como main o develop.

- Gestión de la entrega continua: La CD permite que los cambios pasen automáticamente de desarrollo a producción de manera controlada. Se define cómo y cuándo se realizarán estos despliegues:

  Despliegues en entornos de prueba (staging): Automáticamente tras el paso exitoso de las pruebas en CI.

  Despliegue en producción: Se despliega automáticamente a producción si todas las pruebas pasan.

- Contenerización y orquestación: Aprovechar herramientas de contenerización como Docker para garantizar que los entornos de desarrollo, prueba y producción sean consistentes. Utilizando Docker en combinación con Kubernetes (para la orquestación), el arquitecto puede implementar una solución escalable y eficiente para el despliegue continuo.

2. Como ayudaria Devops a su productividad

DevOps es una práctica que combina el desarrollo de software (Dev) y las operaciones de TI (Ops) con el objetivo de mejorar la colaboración, la automatización y la eficiencia en el ciclo de vida del software. Implementar un enfoque DevOps en un equipo de desarrollo puede tener un impacto significativo en la productividad, ya que reduce tiempos de entrega, mejora la calidad del software y promueve una cultura de colaboración.