Introducción a los Repositorios de Código Distribuido

Integración continua

Diego Madariaga

Contenidos de la clase

1. Integración continua
2. GitHub Actions

Proyecto final

- Inician la próxima semana
- Grupos de 3 personas
  - Elegidos por ustedes
  - El resto de personas, al azar
- Tema del proyecto: Programación en Python de un bot de Telegram

Bots de Telegram

- A cada grupo le asignaremos un bot de Telegram con un conjunto de funcionalidades
- Cada grupo deberá organizar el desarrollo de su proyecto, contemplando todo lo visto durante el curso

Bots de Telegram

- Evaluación del proyecto:
  - Repositorio (commits, mensajes de commits, creación de ramas, pull requests, issues, integración continua, etc)
  - Presentación grupal (Wiki del proyecto y demo)

Bots de Telegram

- Les entregaremos una guía con los pasos a seguir (con avances estimados para cada fecha)
- Estaremos en Discord durante los horarios de clases
- La próxima semana de clases entregaremos instrucciones y pauta en forma detallada

1.
Integración continua

Integración continua

- Práctica de desarrollo de software
  - Añadir nuevo código a un repositorio central
  - Ejecución de pruebas automáticas

- Objetivos:
  - Encontrar errores de forma temprana
  - Arreglar errores oportunamente
  - Mejorar la calidad del software
  - Reducir tiempos de validación y publicación

Integración continua

- Sin este enfoque, los desarrolladores trabajan de forma aislada
- Resulta más difícil la validación de nuevas versiones que agrupan esfuerzos de múltiples trabajadores
- De forma individual se pueden automatizar ciertas tareas (Git Hooks)
  - Dependen de cada máquina específica (no son compartidos dentro del repositorio)

Integración continua

- Pasos a seguir:
  - Desarrolladores envían cambios de forma periódica al repositorio compartido (Git)
  - Un servicio de Integración continua compila y ejecuta de forma automática tests unitarios para los nuevos cambios

Integración continua

- Existen variados servicios de integración continua que pueden incorporarse a un repositorio Git en GitHub

- En 2018, GitHub incluyó entre sus servicios la posibilidad de automatizar los pasos de compilación y testeo de proyectos
  - GitHub Actions

2.
GitHub Actions

GitHub Actions

- Workflow:
  - Procedimiento automatizado añadido a un proyecto
  - Puede configurarse para que responda a ciertos eventos (ej: “push en rama main”)
  - Puede configurarse para que se ejecute de forma periódica o ante eventos externos

GitHub Actions

- Runner:
  - Es el lugar donde se ejecuta un workflow
  - Puede estar hosteado en un servidor propio
  - Puede estar hosteado por GitHub

GitHub Actions

- Un *workflow* puede ejecutar una serie de *Jobs*, que está compuesto por uno o más *Steps*.
- Un *Action*, es un comando de ejecución del proceso, ejecutado en un *Step* para crear un *Job*.

GitHub Actions

- Cada workflow creado tiene que ser un archivo de tipo .yml (sintaxis de YAML)

- Se deben agregar al directorio `.git/workflows/`

GitHub Actions

```yaml
name: Build and test of Java Project

on: [push]

jobs:
 build:
   runs-on: ubuntu-latest

   steps:
     - uses: actions/checkout@v2
     - name: Set up JDK 1.8
       uses: actions/setup-java@v1
       with:
         java-version: 1.8
     - name: Build with Maven
       run: mvn -B package --file pom.xml
```

```yaml
on: [push]
    Branches: [master]
```

Introducción a los Repositorios de Código Distribuido

Integración continua

Diego Madariaga