



Trabajo Fin de Máster

Servicio de análisis de cambios en repositorios de código para la identificación de fallos potenciales

DIEGO FERMÍN SANZ ALONSO


Directores: Pablo Bermejo López
Luis de la Ossa Jiménez

15/07/2019

Índice de contenidos

1. **Introducción**
2. Estado del arte
3. Servicio web
 1. Selección de variables
 2. Creación de modelo de predicción
4. Implementación
5. Metodología
6. Conclusiones y trabajo futuro

Introducción

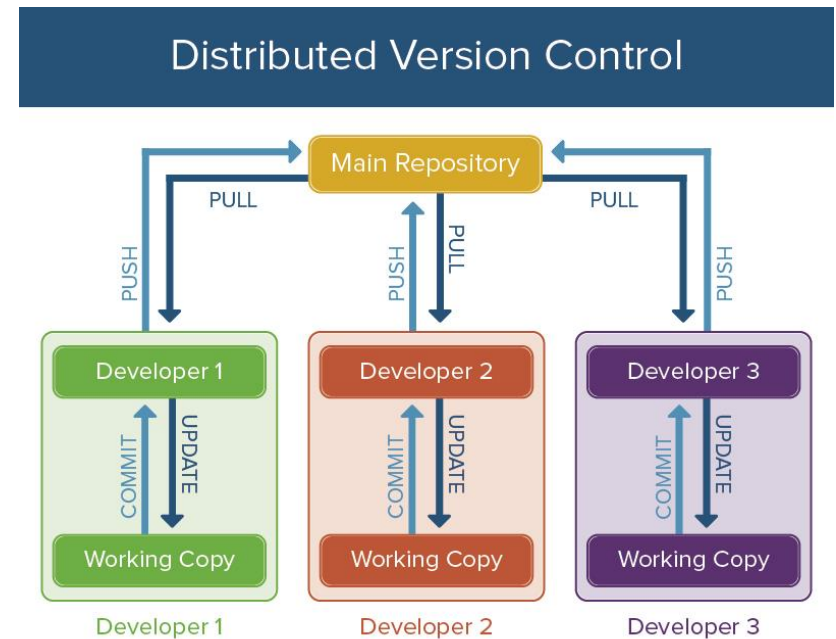
- **Objetivo:** creación de un servicio desplegado en la nube que ayude al programador a identificar fallos potenciales al modificar un repositorio de código antes de subir estos cambios. 
- **Metodología:** uso de una metodología ágil y del cumplimiento de la Primera y la Segunda Vía de DevOps.
 - Entregar valor a los clientes lo antes posible
 - Reducir circuitos de feedback

Índice de contenidos

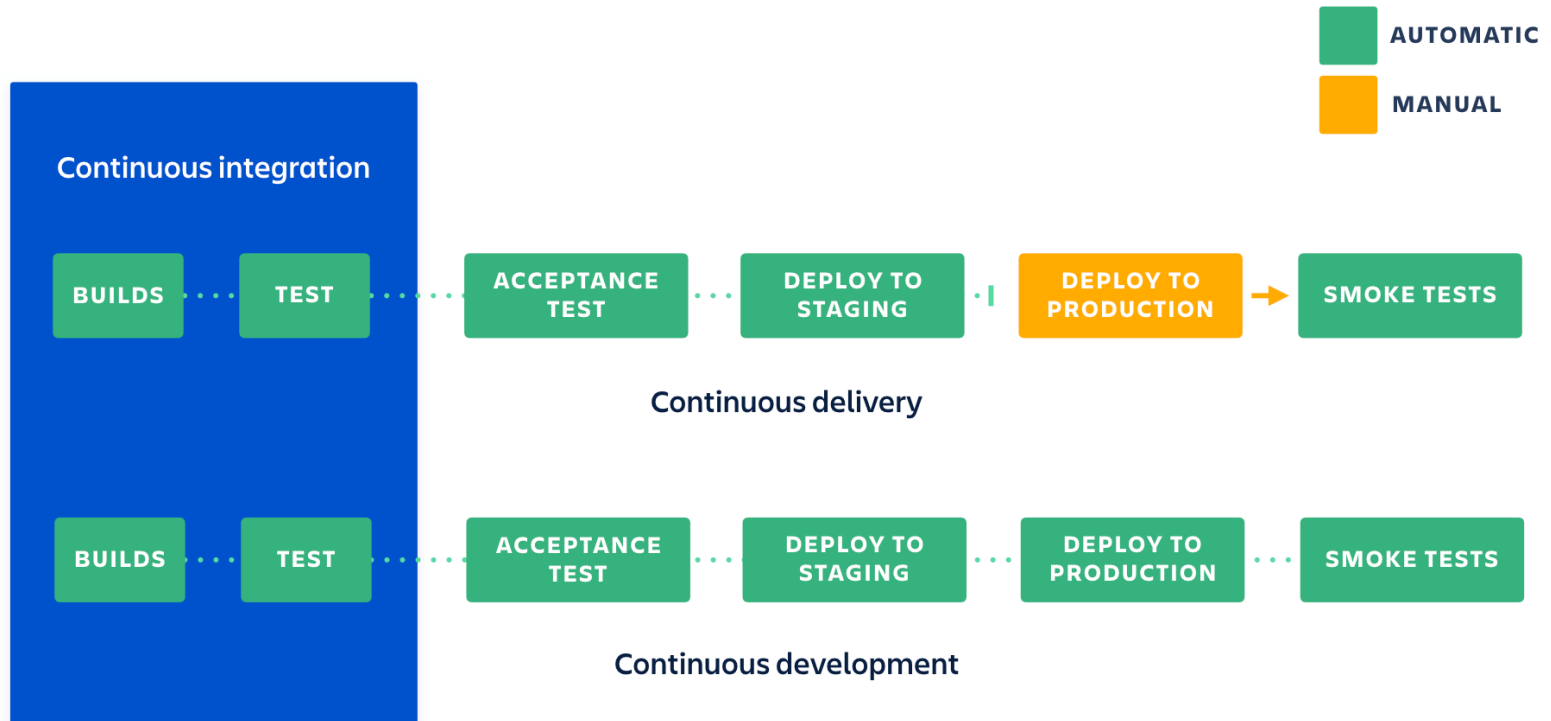
1. Introducción
- 2. Estado del arte**
3. Servicio web
 1. Selección de variables
 2. Creación de modelo de predicción
4. Implementación
5. Metodología
6. Conclusiones y trabajo futuro

Estado del arte

- Sistemas de control de versiones
- Ventajas:
 - Históricos de cada archivo
 - Permite diversificar el desarrollo: Ramificación + Fusión
 - Trazabilidad
- Inconvenientes:
 - Conflictos
 - Calidad del código



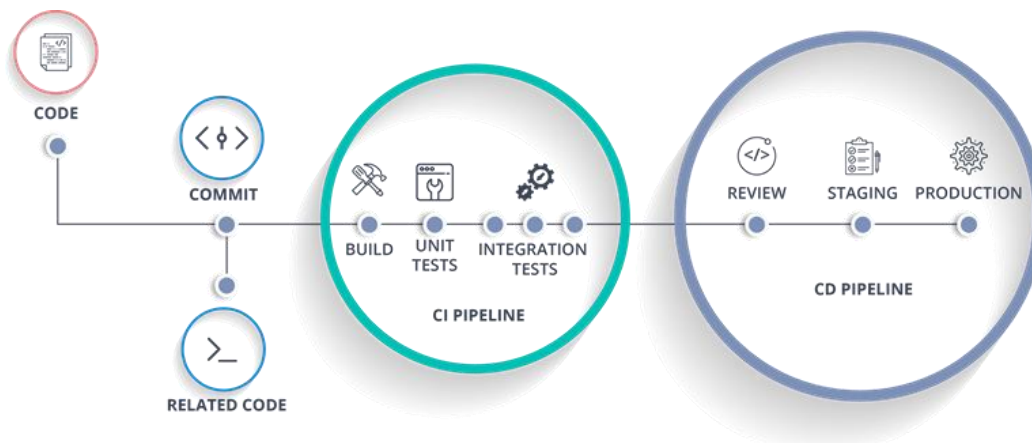
Estado del arte



Estado del arte



- Incremento de código → Proceso automático de CI
- Si las pruebas pasan correctamente → Los cambios se mantienen
- Si las pruebas no pasan correctamente → Pipeline se ha roto
 - Todo el equipo debe parar su trabajo hasta que se arregle el fallo
- **Objetivo** → Identificar fallos potenciales antes de iniciar proceso de CI

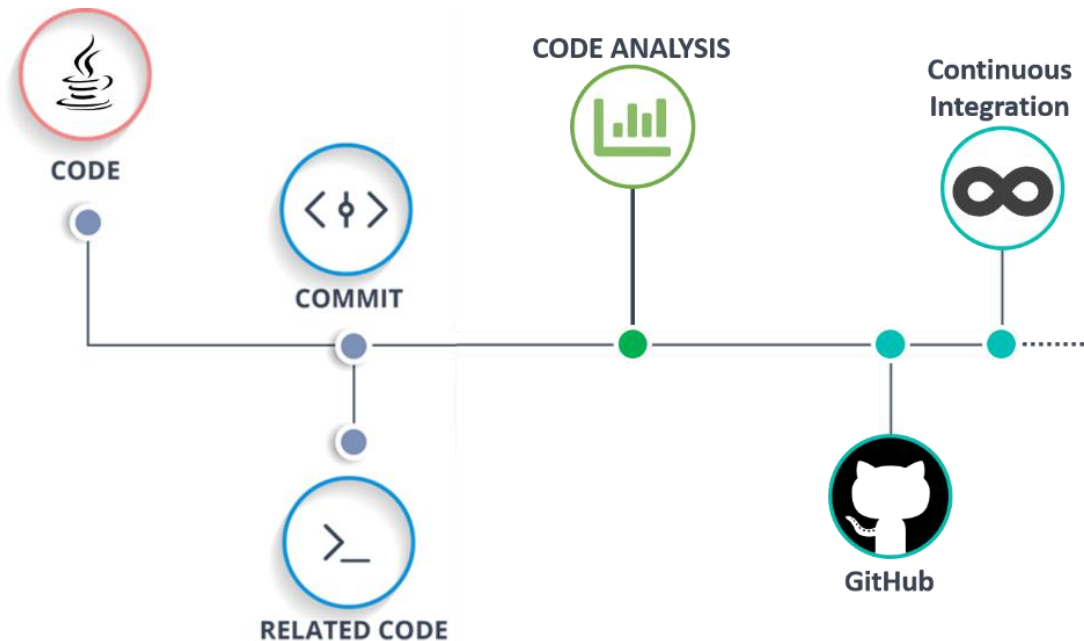


Índice de contenidos

1. Introducción
2. Estado del arte
3. **Servicio web**
 1. Selección de variables
 2. Creación de modelo de predicción
4. Implementación
5. Metodología
6. Conclusiones y trabajo futuro

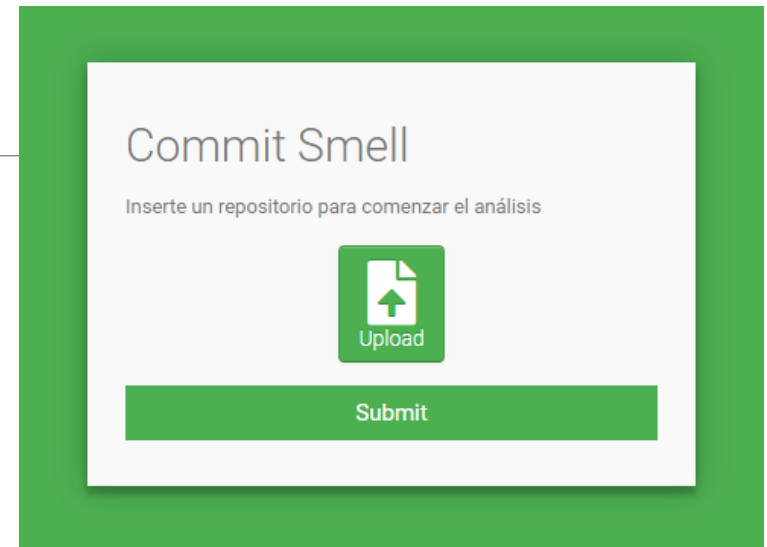
Servicio web

- Objetivos:
 - Mostrar métricas más importantes para la identificación de fallos potenciales.
 - Creación de modelo de predicción que analice automáticamente si pueden haber fallos potenciales.



Servicio web

1. Usuario sube repositorio local
2. Detección de archivos modificados
3. Análisis estático de código
4. Mostrar resultados al usuario



Commit Smell

Inserte un repositorio para comenzar el análisis

Upload

Submit

Class	CBO	NLE	RFC	CXMR	WMC	DMR	CPMR	TNLA	WI	SMR
Hammer	0	1	5	0	6	13	0	2	13	0
User	0	1	9	0	15	19	0	5	19	0
AppTest	0	2	1	0	3	5	0	0	5	0

Información sobre las métricas:

- **CBO:** *Coupling Between Object classes*, número de usos de otras clases. Un valor muy alto de este valor indica que es muy dependiente de otros módulos, y por tanto más difícil de testear y utilizar, además de muy sensible a cambios. Si tiene un valor alto quizás debería revisar sus cambios.
- **NLE:** *Nesting Level Else-If*, grado de anidamiento máximo de cada clase (bloques de tipo if-else-if cuentan como 1 nivel)
- **RFC:** *Response set For Class*: combinación de número de métodos locales y métodos llamados de otras clases.

Índice de contenidos

1. Introducción
2. Estado del arte
3. Servicio web
 1. Selección de variables
 2. Creación de modelo de predicción
4. Implementación
5. Metodología
6. Conclusiones y trabajo futuro

Servicio web: Selección de variables

- Base de datos de Bugs
 - 15 proyectos GitHub
 - Análisis estático de cada clase
 - Número de bugs de cada clase
- Preprocesamiento de las variables
- Selección de variables
 - 10 más representativas para la identificación de fallos potenciales



Class	CBO	NLE	RFC	CXMR	WMC	DMR	CPMR	TNLA	WI	SMR
Hammer	0	1	5	0	6	13	0	2	13	0
User	0	1	9	0	15	19	0	5	19	0
AppTest	0	2	1	0	3	5	0	0	5	0

Información sobre las métricas:

- **CBO:** *Coupling Between Object classes*, número de usos de otras clases. Un valor muy alto de este valor indica que es muy dependiente de otros módulos, y por tanto más difícil de testear y utilizar, además de muy sensible a cambios. Si tiene un valor alto quizás debería revisar sus cambios.
- **NLE:** *Nesting Level Else-If*, grado de anidamiento máximo de cada clase (bloques de tipo if-else-if cuentan como 1 nivel)
- **RFC:** *Response set For Class*: combinación de número de métodos locales y métodos llamados de otras clases.
- **CXMR:** *Complexity Metric Rules*, violaciones en las buenas prácticas relativas a métricas de complejidad. Si es distinto de 0, quizás deba revisar sus cambios.
- **WMC:** *Weighted Methods per Class*, número de caminos independientes de una clase. Se calcula como la suma de la complejidad ciclomática de los métodos locales y bloques de inicialización.
- **DMR:** *Documentation Metric Rules*, violaciones de buenas prácticas relativas a la cantidad de comentarios y documentación.
- **CPMR:** *Coupling Metric Rules*, violaciones en las buenas prácticas relativas al acoplamiento de las clases. Si es distinto de 0, quizás deba revisar sus cambios.
- **TNLA:** *Total Number of Local Attributes*, número de atributos locales de cada clase.
- **WI:** *Warning Info*, advertencias de tipo `WarningInfo` en cada clase
- **SMR:** *Size Metric Rules*, violaciones en las buenas prácticas relativas al tamaño de las clases. Si es distinto de 0, quizás deba revisar sus cambios.



Servicio web: Creación de modelo de predicción

- Entrenamiento de modelos con la base de datos disponible
 - Árboles de decisión
 - Regresión lineal 
 - Random Forest
 - XGBoost
 - ...
- Optimización de parámetros
- Preprocesamiento del conjunto de entrenamiento 

Ver predicción de bugs (BETA)

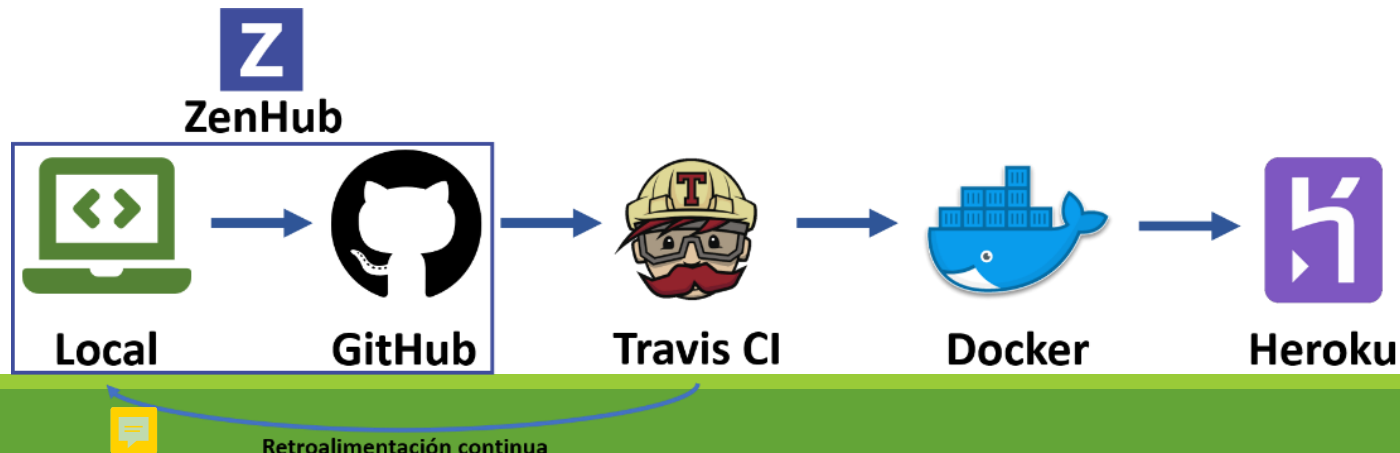
Class ConfigureForDosVisitor probably hasn't bugs
Class ConfigureForUnixVisitor probably hasn't bugs
Class ConfigureForDosVisitorTest probably hasn't bugs
Class ConfigureForUnixVisitorTest probably hasn't bugs

Índice de contenidos

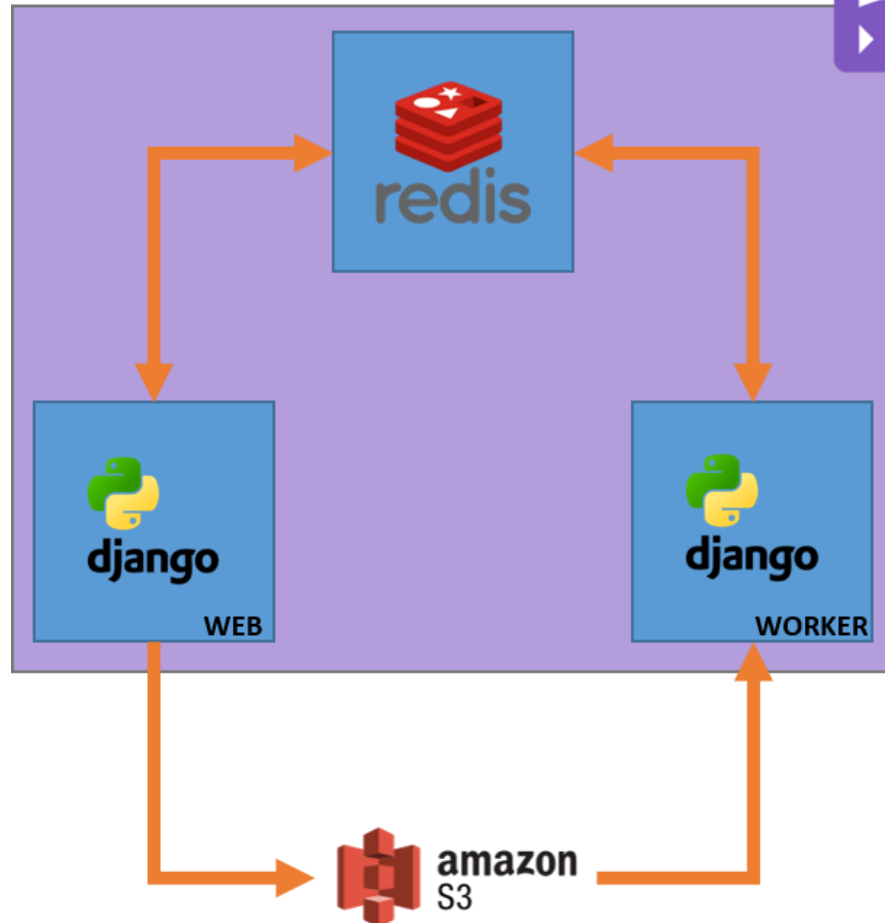
1. Introducción
2. Estado del arte
3. Servicio web
 1. Selección de variables
 2. Creación de modelo de predicción
- 4. Implementación**
5. Metodología
6. Conclusiones y trabajo futuro

Implementación

- **GitHub + ZenHub**: Gestión del proyecto.
- **Python + Django**: Código de la aplicación web.
- **TravisCI**: Integración y Despliegue Continuo.
- **Docker**: orquestación de contenedores.
- **Heroku**: despliegue en la nube.
- **Amazon S3**: base de datos.




Implementación

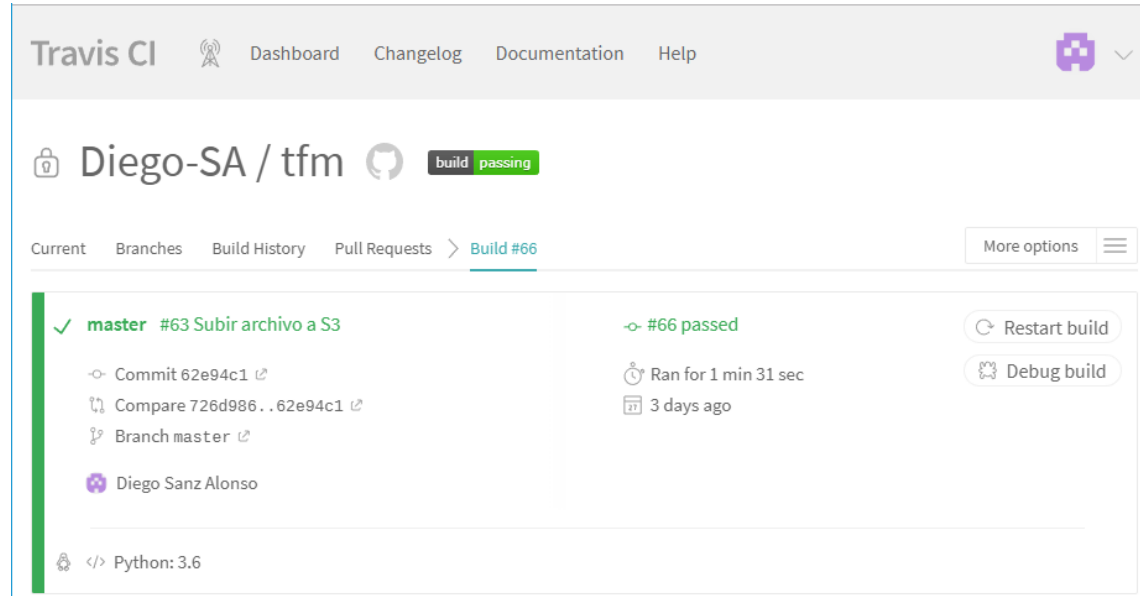


Índice de contenidos

1. Introducción
2. Estado del arte
3. Servicio web
 1. Selección de variables
 2. Creación de modelo de predicción
4. Implementación
5. **Metodología**
6. Conclusiones y trabajo futuro

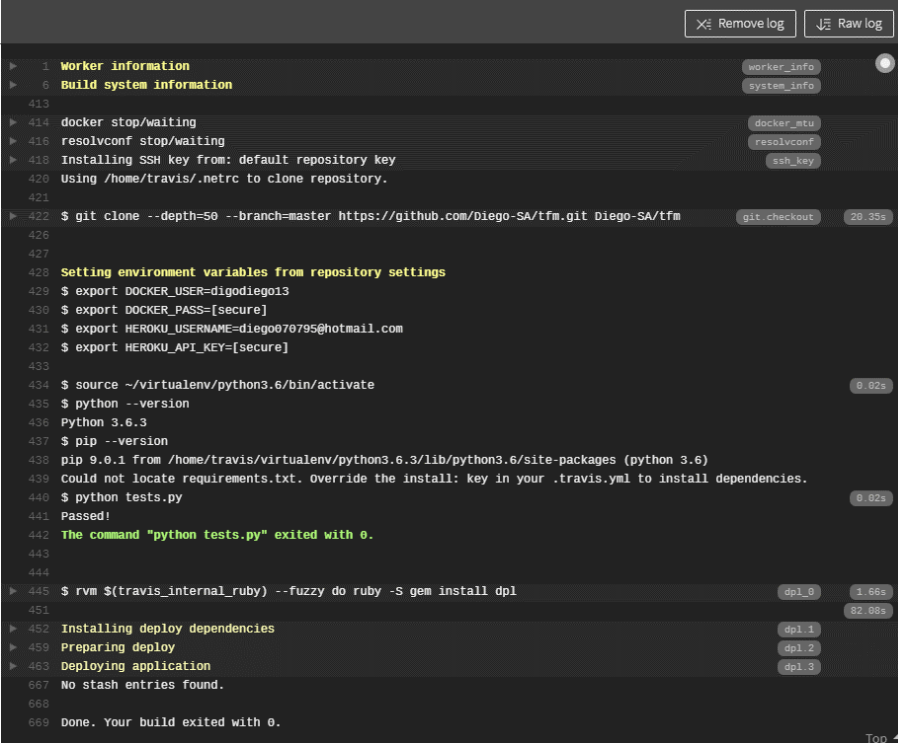
Metodología

- Scrum adaptado a proyectos de ciencia de datos. 
- Filosofía DevOps: cumplir con las dos primeras Vías:
 1. Aportar valor al cliente lo antes posible → Flujo de CI/CD
 2. Reducir los circuitos de Feedback → Notificación ante fallos en CI



Flujo de CI/CD

1. Desarrollador sube un nuevo incremento a GitHub.
2. TravisCI compila el proyecto y ejecuta las pruebas automáticas.
3. Creación de contenedores mediante Docker Compose.
4. Despliegue de esos contenedores en Heroku.



The screenshot displays a TravisCI build log for a project named 'Diego-SA/tfm'. The log is organized into sections with expandable/collapsible icons on the right. The steps shown are:

- Worker information** (worker_info)
- Build system information** (system_info)
- docker stop/waiting** (docker_mtu)
- resolvconf stop/waiting** (resolvconf)
- Installing SSH key from: default repository key** (ssh_key)
- Using /home/travis/.netrc to clone repository.**
- \$ git clone --depth=50 --branch=master https://github.com/Diego-SA/tfm.git Diego-SA/tfm** (git_checkout, 20.35s)
- Setting environment variables from repository settings**
- \$ export DOCKER_USER=digodiego13**
- \$ export DOCKER_PASS=[secure]**
- \$ export HEROKU_USERNAME=diego070795@hotmail.com**
- \$ export HEROKU_API_KEY=[secure]**
- \$ source ~/virtualenv/python3.6/bin/activate** (0.02s)
- \$ python --version**
- Python 3.6.3**
- \$ pip --version**
- pip 9.0.1 from /home/travis/virtualenv/python3.6.3/lib/python3.6/site-packages (python 3.6)**
- could not locate requirements.txt. Override the install: key in your .travis.yml to install dependencies.**
- \$ python tests.py** (0.02s)
- Passed!**
- The command "python tests.py" exited with 0.**
- \$ rvm \$(travis_internal_ruby) --fuzzy do ruby -S gem install dpl** (dpl_0, 1.66s)
- Installing deploy dependencies** (dpl_1, 52.08s)
- Preparing deploy** (dpl_2)
- Deploying application** (dpl_3)
- No stash entries found.**
- Done. Your build exited with 0.**

At the bottom right, there is a 'Top' link with an upward arrow.


Índice de contenidos

1. Introducción
2. Estado del arte
3. Servicio web
 1. Selección de variables
 2. Creación de modelo de predicción
4. Implementación
5. Metodología
6. Conclusiones y trabajo futuro

Conclusiones y trabajo futuro

- Servicio desplegado en la nube para la identificación de fallos potenciales.
- Metodología ágil + DevOps
- Resultados del flujo de CI/CD
 - Cada vez que se haga un nuevo incremento de código la aplicación se desplegará y llegará al usuario de manera transparente y muy rápida.
- Trabajo futuro
 - Permitir proyectos con otros lenguajes de programación.
 - Mejorar resultados de modelos de predicción.
 - Búsqueda de otras bases de datos.
 - Nuevas características: variables a nivel de fichero, histórico de cambios, etc.

Conclusiones y trabajo futuro

- Valoración personal
 - Uso de varias asignaturas del Máster
 - Desarrollo de Sistemas Inteligentes
 - Planificación y Gestión de Infraestructuras TIC
 - Gestión, Certificación y Evaluación de Sistemas y Servicios 
- Además del desarrollo del proyecto, se ha conseguido adquirir una base en cuanto a metodologías ágiles y técnicas de CI y CD, lo que nos proporciona conocimiento que se puede usar en muchos contextos:
 - Tecnologías
 - Lenguajes de programación
 - Áreas de trabajo
- Capacidad de desarrollar un producto completo gracias a los conocimientos multidisciplinares adquiridos.

Muchas gracias.
¿Alguna pregunta?