accenture technology

μServicios y Machine Learning

commit

MAD · NOV 23-24 · 2018

Rafa Hidalgo @oscuroweb
Julio Palma @restalion

Sobre nosotros

Julio Palma Vázquez

Desarrollador Java desde hace más de 17 años, muy interesado en IoT, microservicios y arquitectura de aplicaciones. El trabajo me ha llevado a buena parte de España pero también a Dinamarca, Alemania, Irlanda y EEUU. Speaker en eventos locales e internacionales. Implicado con la comunidad de desarrolladores. Gran fan de la ciencia ficción y los juegos de rol, ciclista de montaña y padre orgulloso. En mis ratos libres trabajo en Accenture Technology

@restalion **http://github.com/restalion**

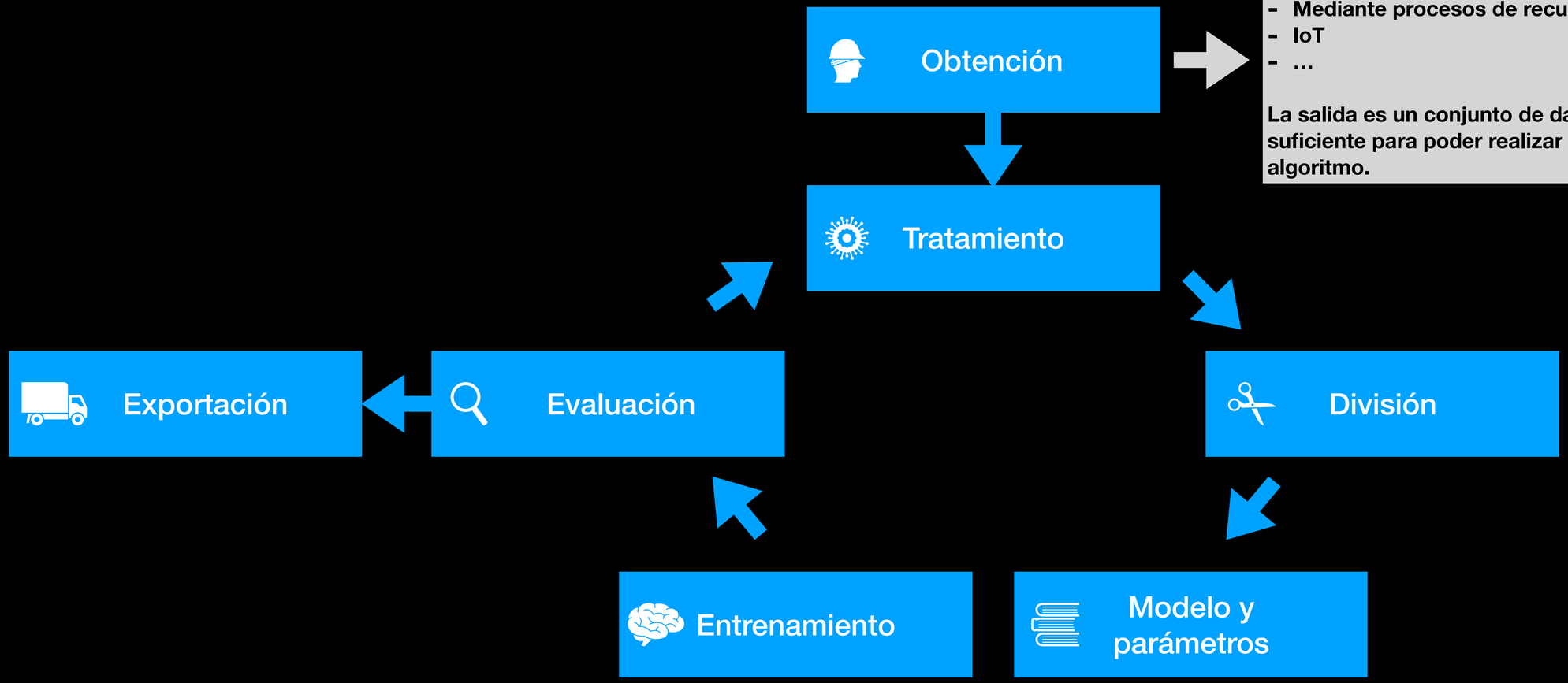
Rafael Hidalgo Calero

Echo la tarde programando desde hace unos 6 años, especialmente en Java. Ultimamente muy interesado en el mundo Big Data y el Machine Learning. Como speaker he participado en la edición del Lambda World, Cádiz en 2017 y en la Opensouthcode, Málaga en 2018. Trabajando en Accenture Technology desde 2012

@oscuroweb **http://github.com/oscuroweb**

Machine Learning (ML)

Machine Learning (ML)

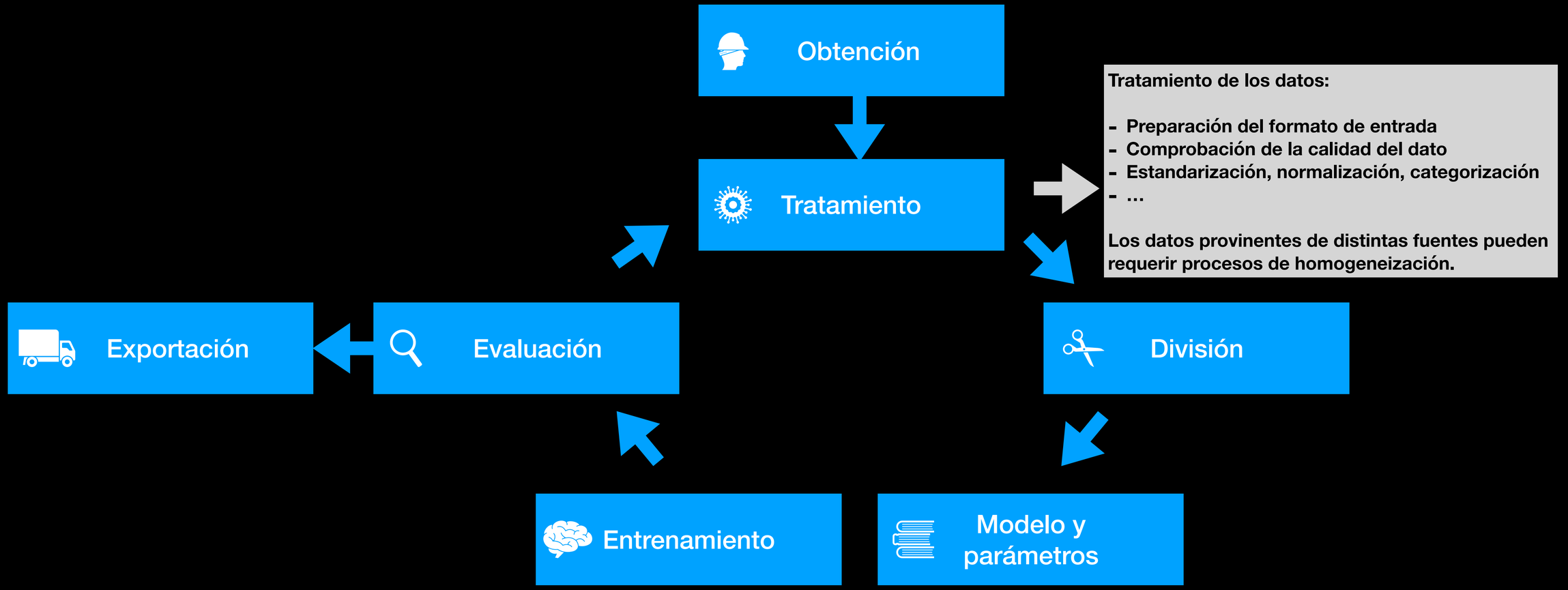


Recuperación de los datos:

- **A partir de datos del cliente.**
- **Mediante procesos de recuperación.**
- **IoT**
- **...**

La salida es un conjunto de datos con volumen suficiente para poder realizar el entrenamiento del algoritmo.

Machine Learning (ML)



Machine Learning (ML)

Machine Learning (ML)

Seleccionar y parametrizar el modelo a utilizar para el aprendizaje:

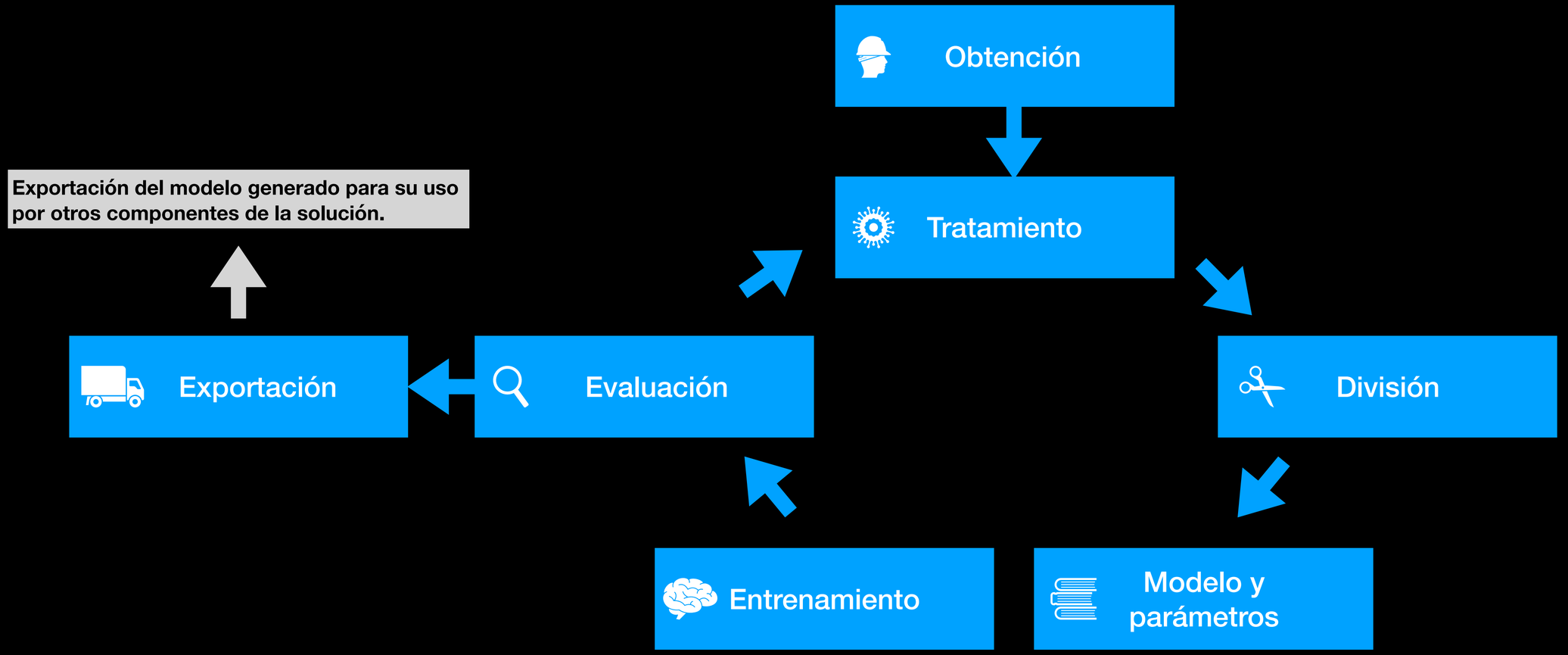
- **Random Forest**
- **Neural Net**
- **SVM**

Machine Learning (ML)

Ejecución del entrenamiento basado en los datos escogidos y el modelo seleccionado.

Machine Learning (ML)

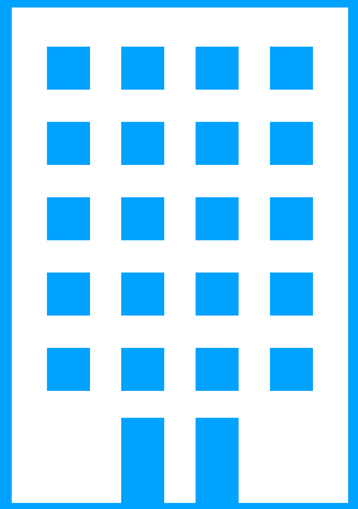
Machine Learning (ML)



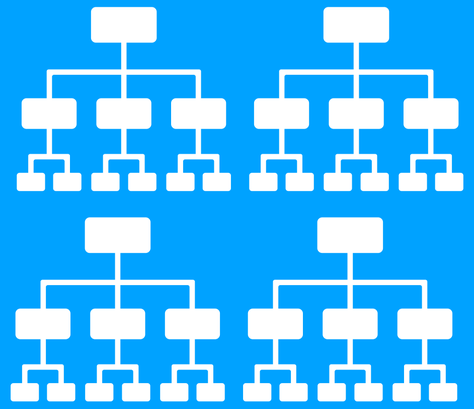
Modelos de arquitectura

Modelos de arquitectura

Monolito

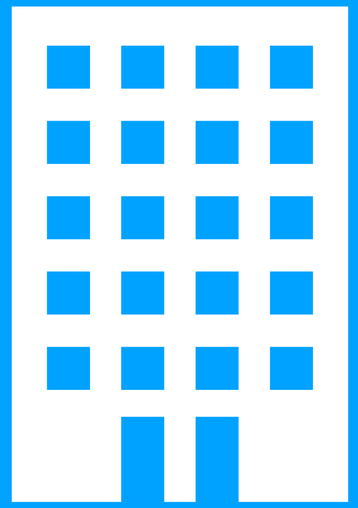


µServicios

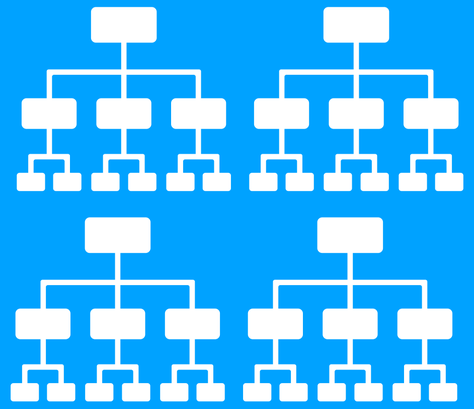


Modelos de arquitectura

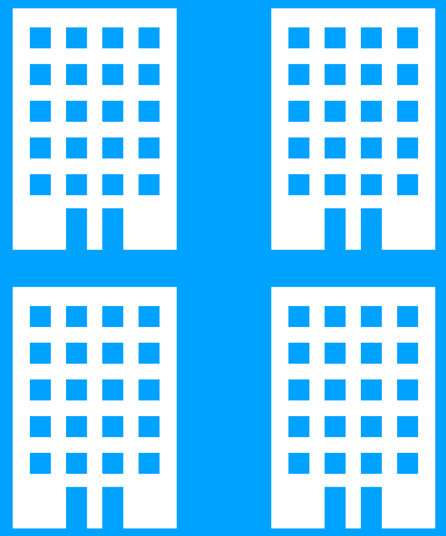
Monolito



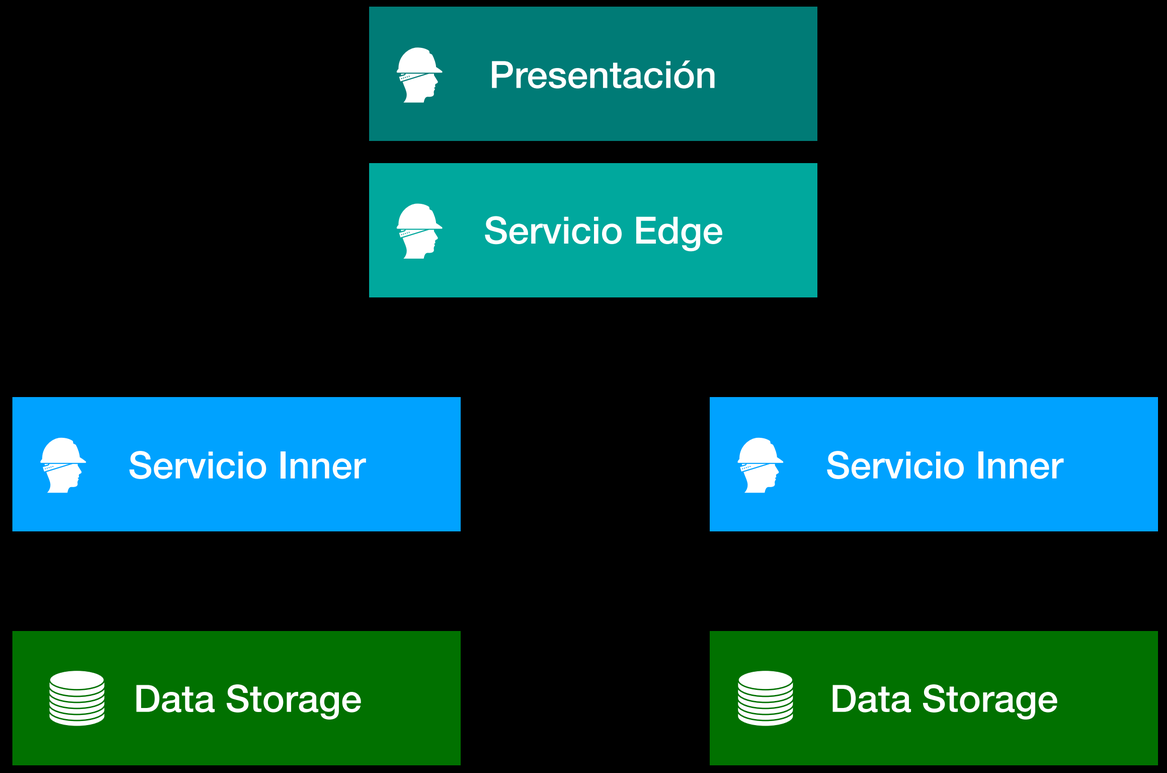
µServicios



Minilito

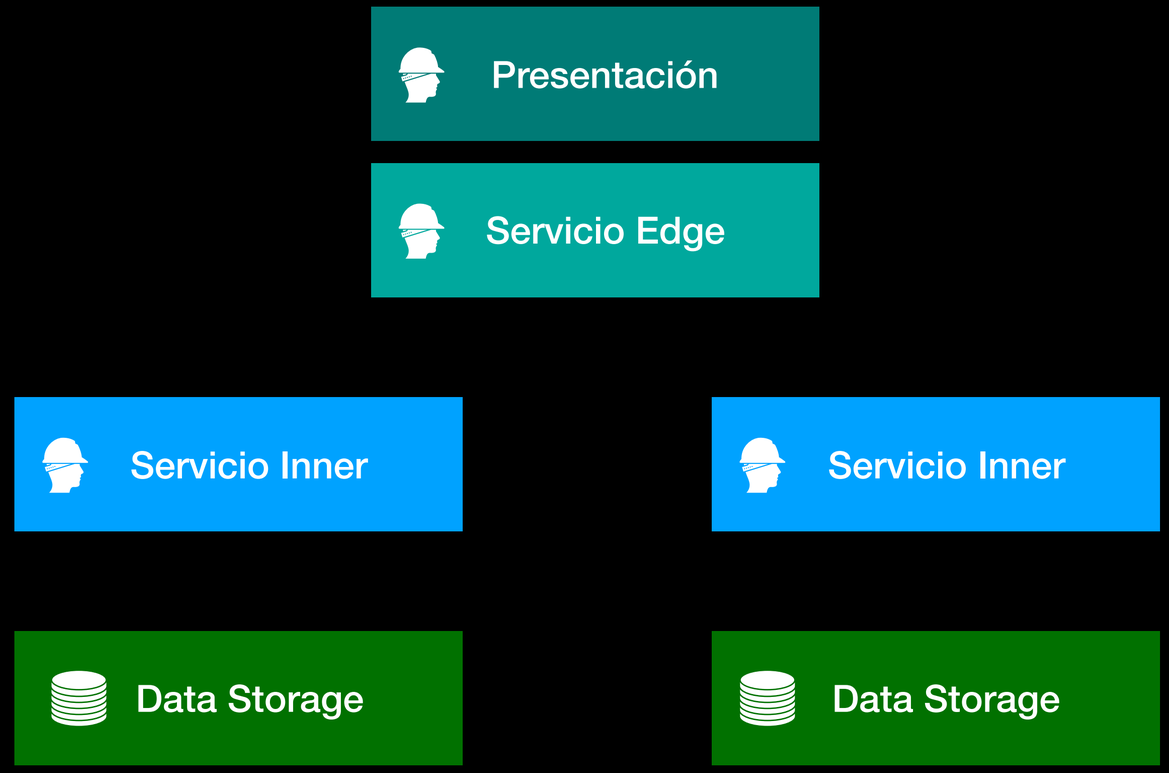


Arquitectura de μServicios



Arquitectura de μServicios

- Configuración Centralizada
- Seguridad
- Logs distribuidos
- Auditoría
- Documentación Interfaces

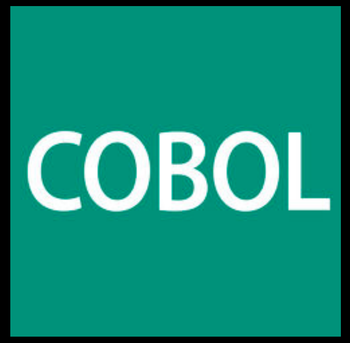


- Métricas
- Composición de la solución.
- Resistencia a errores
- Localización de servicios
- Gestión de la carga

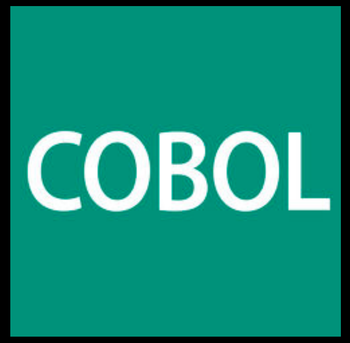
Lenguajes para ML

Lenguajes para ML

Lenguajes para aplicaciones empresariales



Lenguajes para aplicaciones empresariales



¿Frameworks para ML?

	Python	Scala	R	JS	Java	.NET	COBOL
H2O.ai	✓	✓	✓				
TensorFlow	✓	✓		✓	✓	✓	
scikit learn	✓						
Apache Spark	✓	✓			✓		

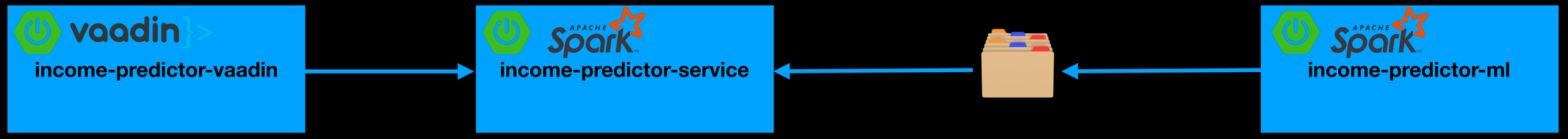
¿Frameworks para ML?

	Python	Scala	R	JS	Java	.NET	COBOL
H2O.ai	✓	✓	✓				
TensorFlow	✓	✓		✓	✓	✓	
scikit learn	✓						
Apache Spark	✓	✓			✓		

ML + µS

- Un problema habitual es trasladar los algoritmos de ML a sistemas en producción
- Búsqueda de mecanismos para la integración de los distintos componentes de la solución
- Necesitamos una arquitectura que nos permita aprovechar:
  - Ejecuciones de los algoritmos de ML (y toda su potencia)
  - Flexibilidad de las arquitecturas de µServicios
  - Todos los mecanismos de DevOps aplicables al resto de la solución

Arquitectura de Aplicación

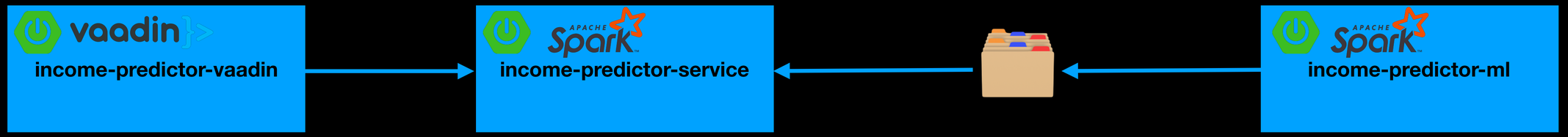


- Expone la UI
- Consume el servicio REST

- Expone el interfaz REST
- Lee el modelo desde el fichero
- Guarda el feedback de los usuarios para permitir Reentrenar el algoritmo.

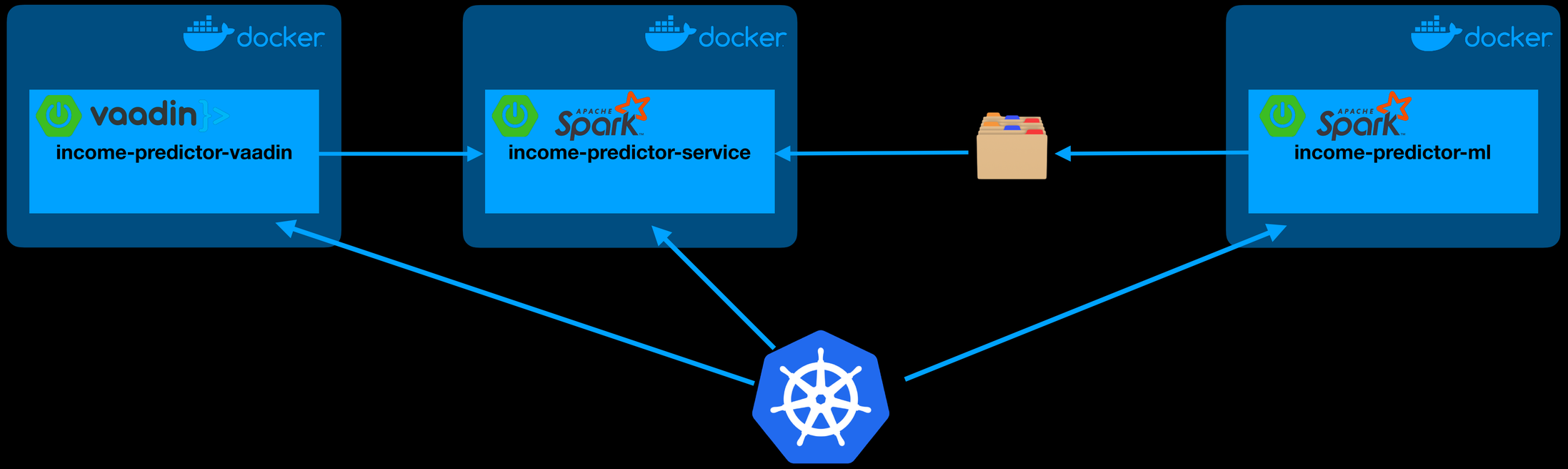
- Servicio planificado para reentrenar el modelo con Los datos originales y el feedback de los usuarios
- Genera un nuevo modelo y lo exporta para su uso

Arquitectura de Aplicación



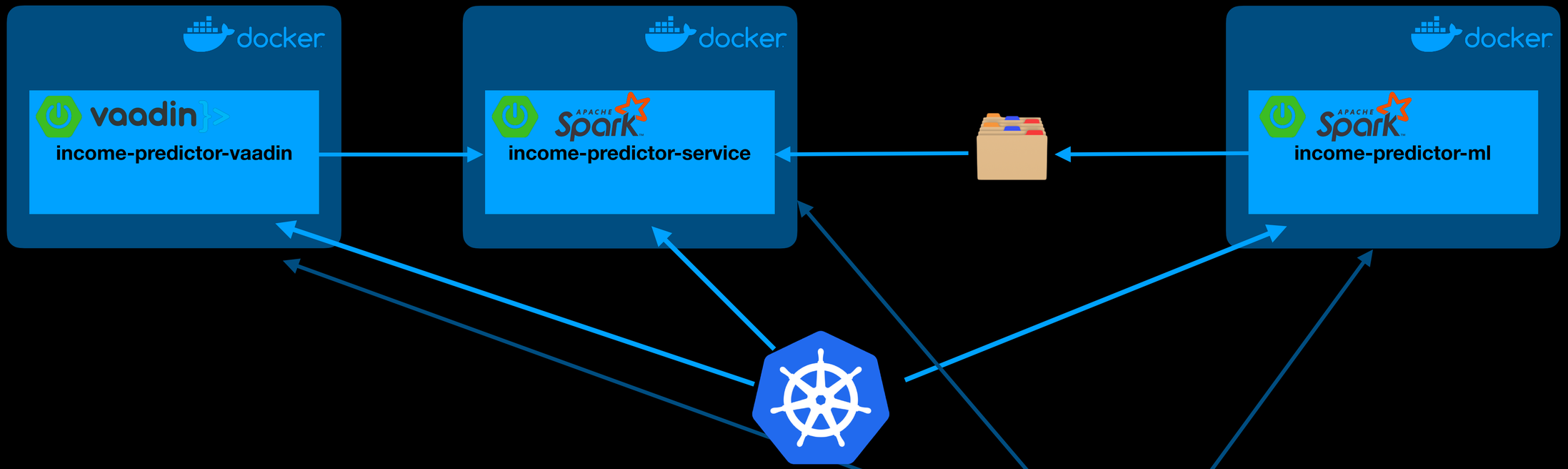
- Spring Boot como framework base, completamente integrado con el resto de la aplicación
- Apache Spark como gestor de ML
- Vaadin como framework de presentación
- Todos los elementos modelados como µServicios

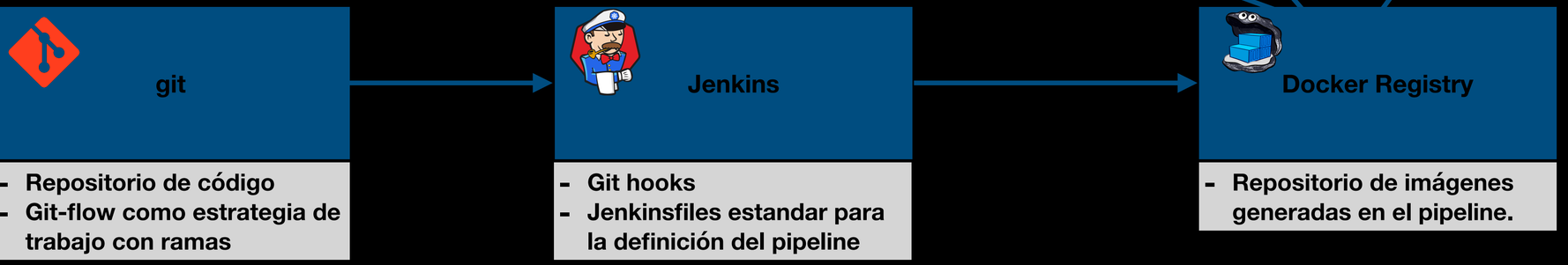
Arquitectura de Ejecución



- Docker para empaquetar en contenedores los servicios y distribuirlos
- Kubernetes como orquestador y gestor

CI y Pipelines



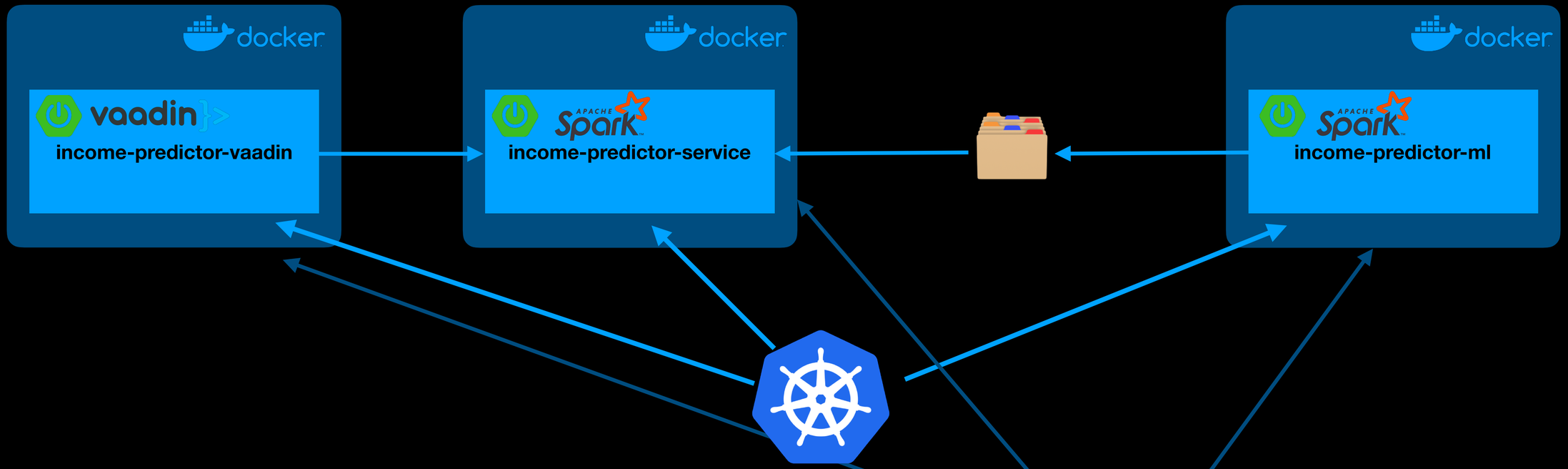


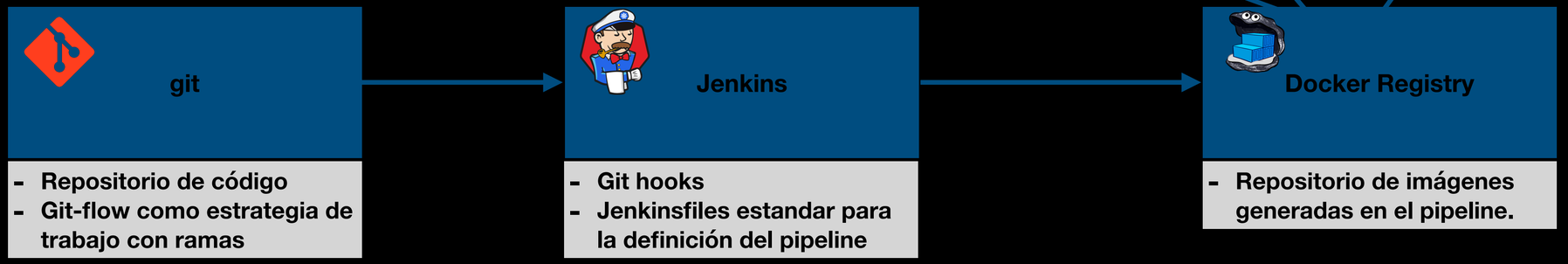
Demo Time

Estándar vs Libertad

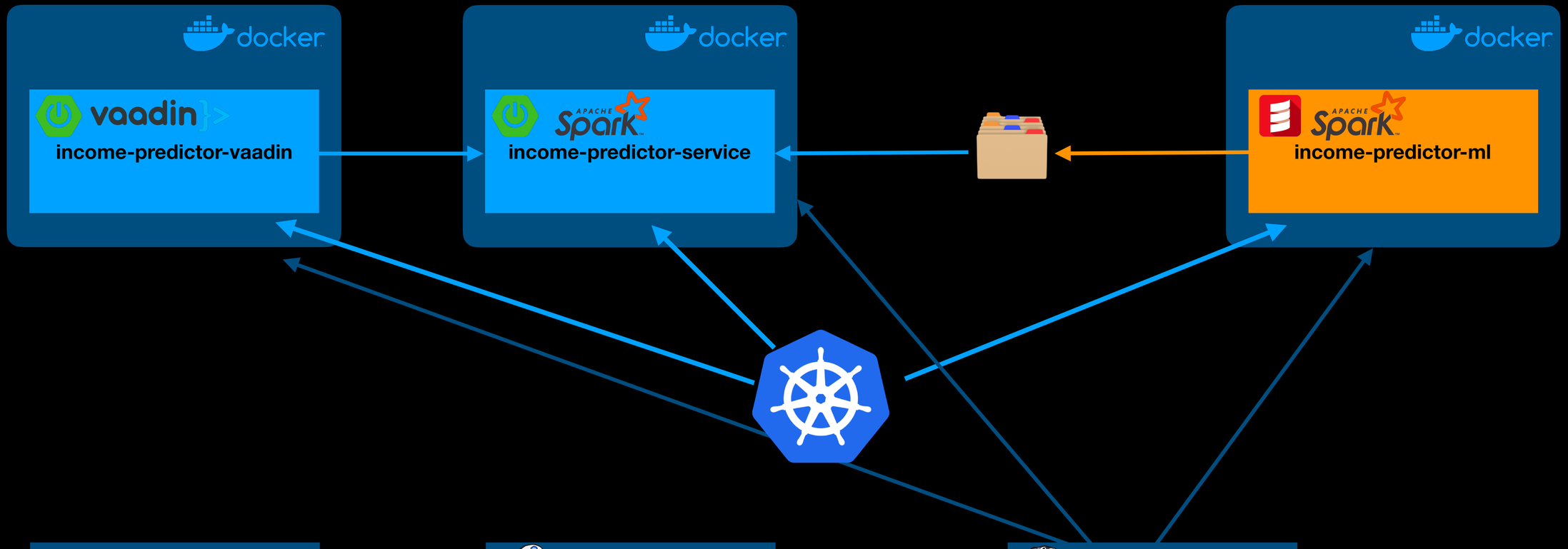
- En nuestra arquitectura hemos intentado estandarizar el desarrollo de forma que todos los equipos trabajen de la misma manera
- En el mundo real siempre hay equipos que requieren modificar estos estándares para poder cumplir con sus objetivos o para aprovechar sus conocimientos
- ¿Nos permite nuestro planteamiento ofrecer esa libertad?

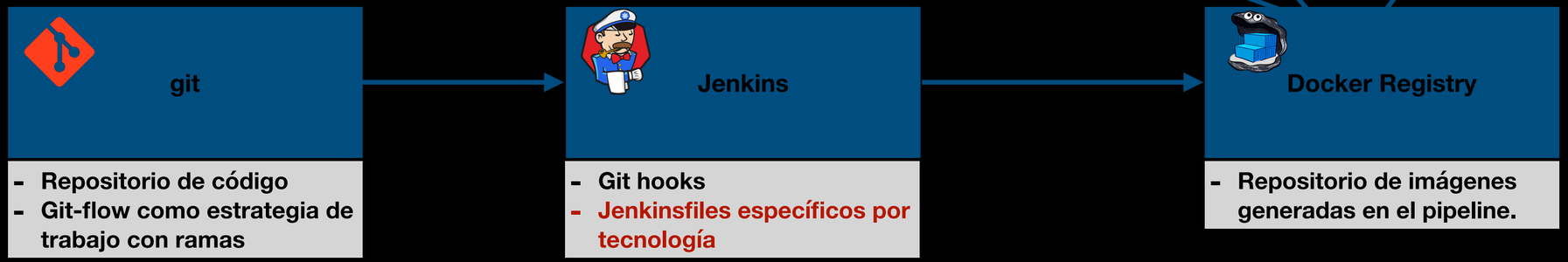
Soluciones Políglotas



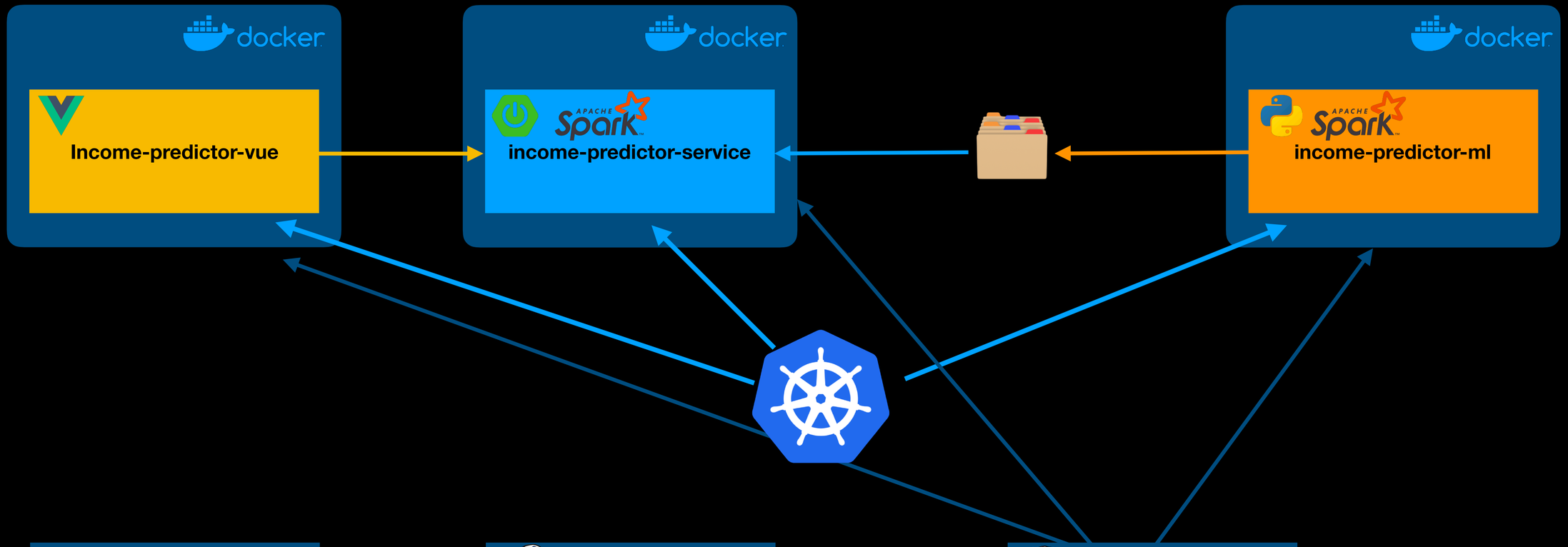


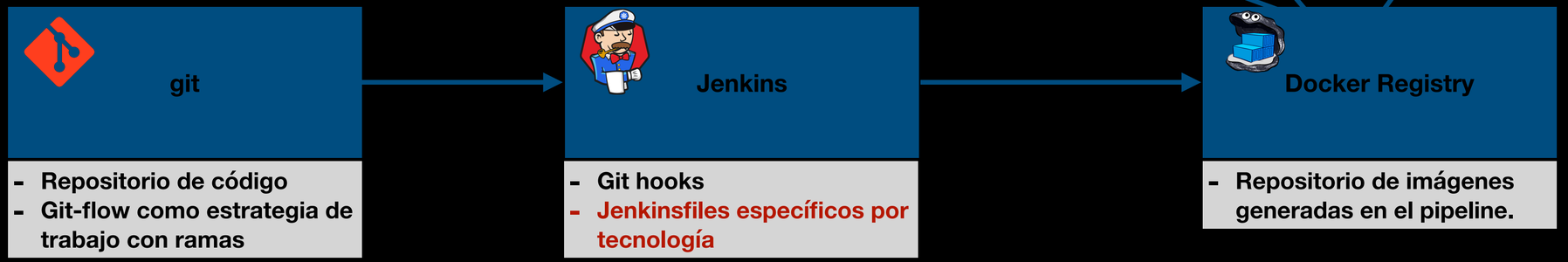
Soluciones Políglotas





Soluciones Políglotas



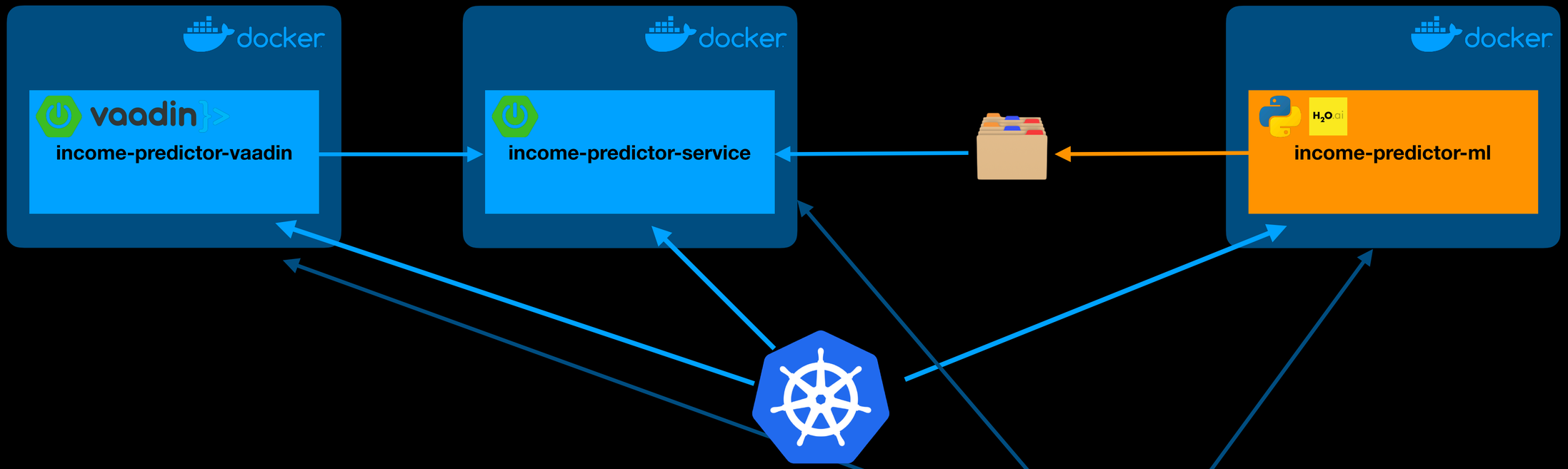


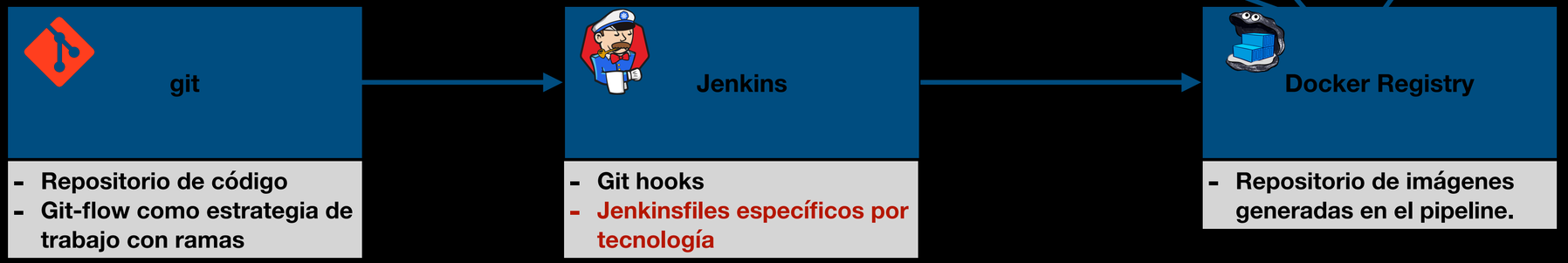
- Repositorio de código
- Git-flow como estrategia de trabajo con ramas

- Git hooks
- Jenkinsfiles específicos por tecnología

- Repositorio de imágenes generadas en el pipeline.

Soluciones Políglotas





Demo Time

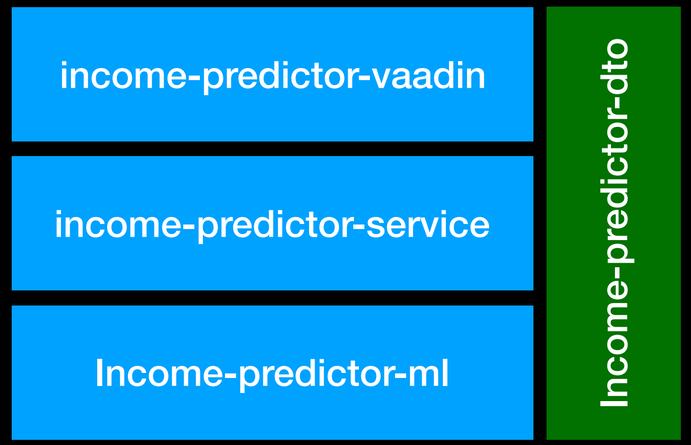
Ventajas de la arquitectura

- Uso de frameworks bien conocidos como base de la arquitectura
- Integración con los entornos de CI y gestión de la calidad
- Utilización de las herramientas de gestión de la calidad para todos los lenguajes a partir de Jenkinsfiles
- Definición de las imágenes Docker usando Dockerfiles
- Entorno políglota a través de interfaces REST o mensajería, podemos combinar la potencia de cada lenguaje

Bonus Track: GraalVM™

- ¿Y si pudiésemos usar la misma VM para distintas tecnologías?
- Soporta: JavaScript, Python, Ruby, R, Java, Scala, Kotlin, C, C++
- Utiliza para ejecutarse OpenJDK, Node.JS, Oracle, MySQL o ejecución Standalone.
- Sólo tendríamos que modificar los Dockerfiles de los proyectos en los que queramos usar GraalVM
- Permite tener un modelo común para debugado, monitorización, …

Proyectos de Ejemplo



- https://github.com/Oscuro-Restalion
  - income-predictor-dto
  - income-predictor-vaadin
  - income-predictor-service
  - income-predictor-ml
  - income-predictor-h2o
  - income-predictor-h2o-service

Referencias

- Apache Spark: https://spark.apache.org/
- Apache Hadoop: https://hadoop.apache.org/
- Spring Boot: http://spring.io/projects/spring-boot
- Kubernetes: https://kubernetes.io/
- GraalVM: https://www.graalvm.org/
- H2O AI: http://docs.h2o.ai/