

# Ecosistema de Storage

IT BOARDING

**BOOTCAMP**



# Índice



**01** Servicios de  
datos

**02** Un caso  
particular

**03** Números

**04** Integraciones entre  
servicios

IT BOARDING

**BOOTCAMP**

# // 01

# Servicios de datos

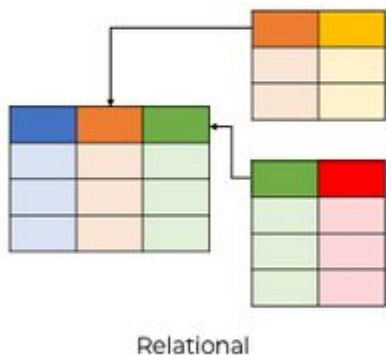
IT BOARDING

**BOOTCAMP**

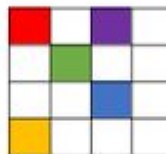


# Servicios de datos

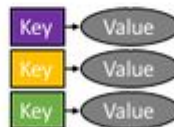
## SQL DATABASES



## NoSQL DATABASES



Column



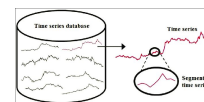
Key-Value



Graph



Document

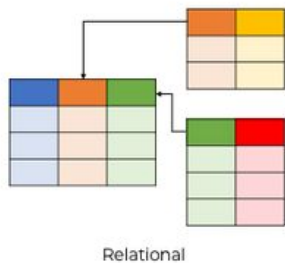


Timeseries

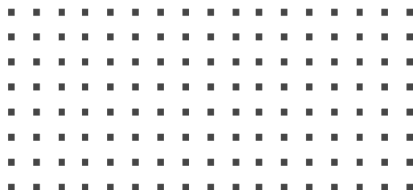
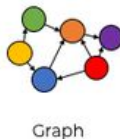
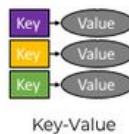


# Decidir

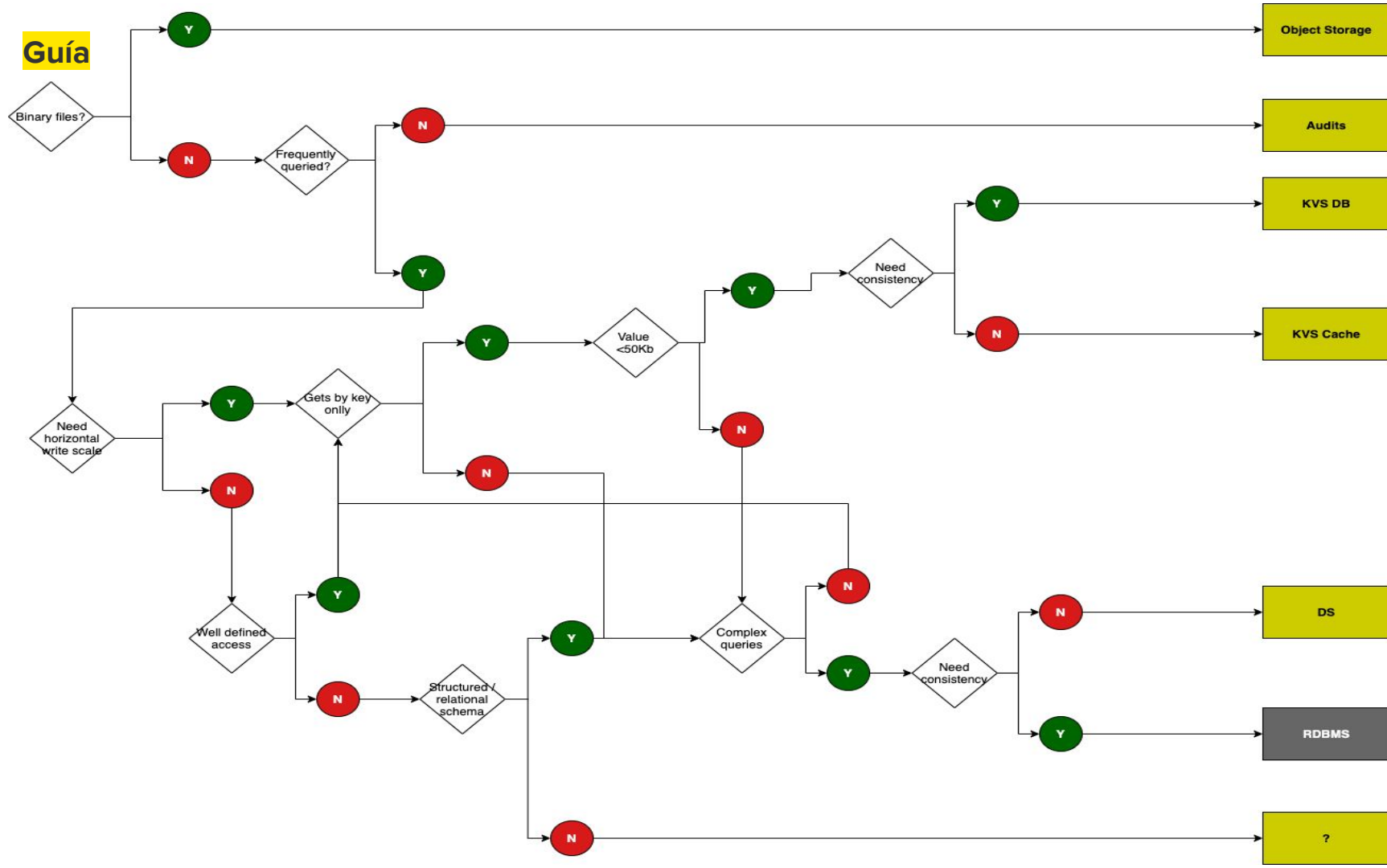
## SQL DATABASES



## NoSQL DATABASES



# Guía



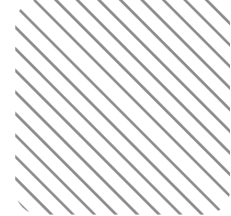
## // 02

# Un caso particular

IT BOARDING

**BOOTCAMP**

# Un caso particular

The background of the slide features a dark grey field with various geometric elements: vertical bars of different heights on the right, a grid of small squares in the bottom right, and a square with a pixelated pattern on the left. The word 'DECODED' appears in small white boxes within the layout.

DECODED

BACKEND TALK

# Storing Critical Transactions

When using the right tool is not enough

|||||

Mariano Labariñas -> Pelito

DECODED

m



# Elección difícil



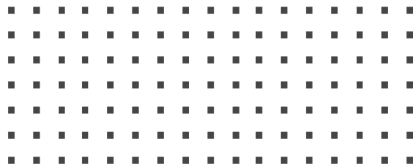
# Resumen





# Ejercicio

1. ¿Tengo que almacenar archivos binarios (recibos) para mi sistema de procesamiento de Billing, que debo utilizar?
2. ¿Tengo que guardar datos de los usuarios de MELI, que debo utilizar?



ECOSISTEMA DE STORAGE

// 03

# Números

IT BOARDING

**BOOTCAMP**

## Datos

**70% apps  
productivas**

**500.000.000  
RPM**

**60% YoY avg  
growth**

**10000+  
servicios de datos**

**8000+  
servicios NoSQL**

**6000  
terabytes**



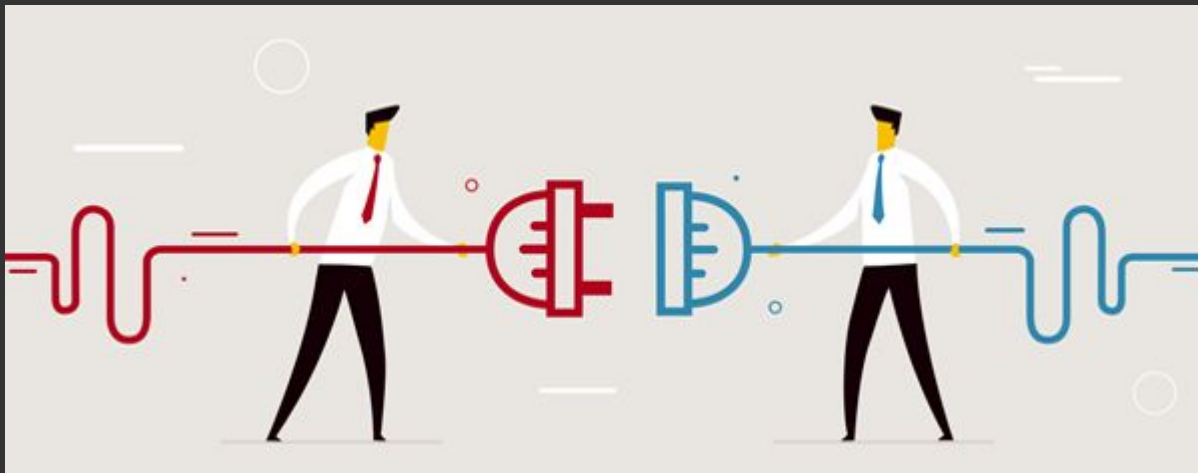
# // 04

# Ecosistema de servicios

IT BOARDING

**BOOTCAMP**

# Ecosistema





## Casos de uso

1. Tengo una base de datos relacional que utilizo para guardar información sobre el carrito de compras. Cuando alguien abandona el carrito, se inserta un registro en una tabla. Quiero que cuando eso pase, la API de Growth pueda enviar una notificación al usuario para que retome el carrito.
2. Tengo un KVS que utilizo para guardar mis items, a los cuales se accede habitualmente por clave, no obstante, ahora me surgió una necesidad de que en ciertos casos quiero poder buscar y agrupar por categoría.
3. Tengo un KVS y quiero poder obtener todos los datos almacenados para poder hacer operaciones en bulk sobre ellos. Cómo lo puedo resolver?
4. Tengo una KVS que utilizo para guardar información sobre el carrito de compras. Cuando alguien abandona el carrito, se inserta un registro en la tabla. Quiero que cuando eso pase, la API de Growth pueda enviar una notificación al usuario para que retome el carrito.
5. Etc

✖ ✖ ✖

✖ ✖ ✖

✖ ✖ ✖

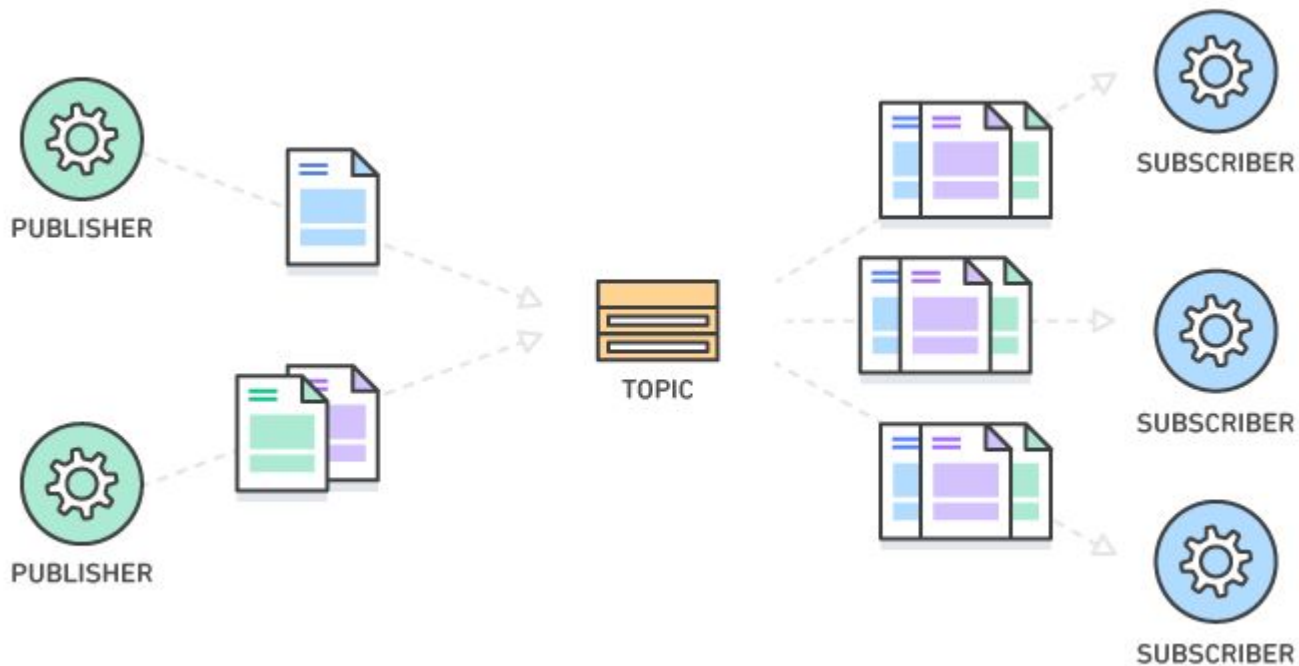




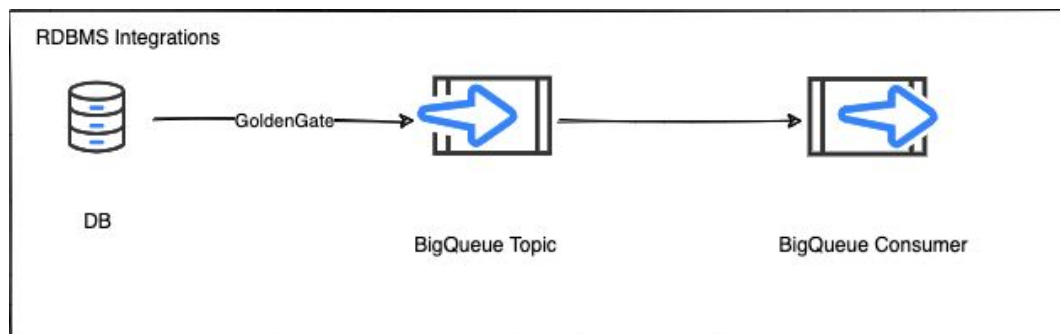
## CDC - Change Data Capture



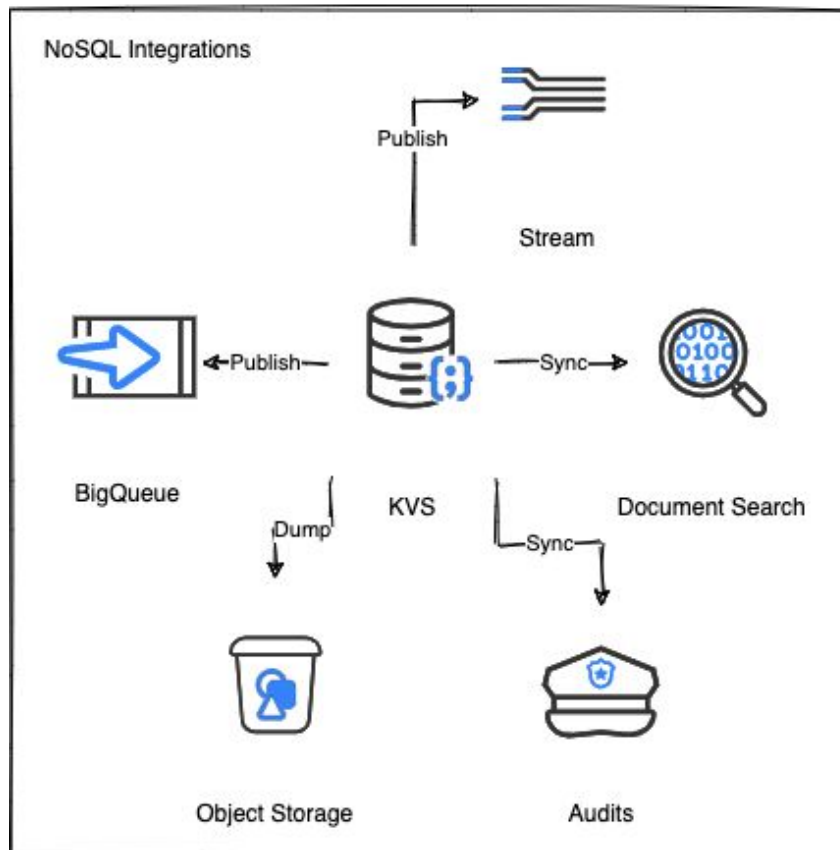
## ASYNCR Communication - Message queues / PubSub / Buses / Streams



# RDBMS integraciones



# NOSQL integraciones





## Do's and Don'ts



Usar las integraciones que sean necesarias



Usar las integraciones pre-definidas



Definir el mínimo nivel necesario de verbosidad



Tener claras las implicancias de las asincronías



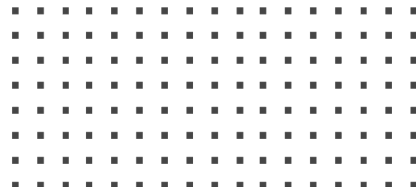
Prender integraciones innecesarias “por las dudas”



Olvidar apagar syncs que ya no sean necesarias o no se usen



No controlar el crecimiento de los servicios asociados



# // Hora de ejercicios!!

IT BOARDING

**BOOTCAMP**

## Ejercicios

1. Tengo que crear una aplicación que se encarga de manejar la gestión de dinero en cuenta de los usuarios de Mercado Pago. Cuando un usuario paga, luego de validar que tiene saldo, tengo que registrar el pago y decrementar su saldo. Cuando un usuario deposita dinero, tengo que incrementar su saldo. Asimismo, con cada pago aprobado, y usando los valores del pago y el usuario varias aplicaciones de Fraude calculan modelos de riesgo de los usuarios. Desde una perspectiva de base de datos, ¿que utilizarías? ¿Porque? ¿Se te ocurre más de una forma de hacerlo?
2. Tengo que crear una aplicación que guarda datos de envíos. Para cada envío tengo que guardar algunos datos clave como shippingId, comprador, vendedor, producto, costo total, transportista y ruta. Asimismo, asociado con eso, se debe guardar una foto de la orden de envío. Usualmente el equipo accede a esta data por el shippingId. Además de esto, necesitan obtener en ocasiones, informaciones sobre los envíos para un comprador o vendedor en particular y agrupar por ellos el costo total generado. Finalmente, por cuestiones regulatorias, deben saber hasta por 10 años por que ruta se envió un producto. Desde una perspectiva de base de datos, ¿que utilizarías? ¿Porque? ¿Se te ocurre más de una forma de hacerlo?

ECOSISTEMA DE STORAGE

# // Storage y Fury

IT BOARDING

**BOOTCAMP**



# Índice



**01** Cómo usar  
una DB en  
Fury?

**02** Cómo lo integro  
con mi código?

**03** Métricas Core / Backups  
& Mirrors - Secondaries

**04** Qué métricas me  
resultan relevantes y  
cómo las veo?

IT BOARDING

**BOOTCAMP**

// 01

# Usar una DB en Fury

IT BOARDING

**BOOTCAMP**



**Vamos a verlo directamente en Fury!**

## Cluster lppricingapi00

### general

**Virtual IP** : lppricingapi00.master.mlaws.com  
**Flavor** : small  
**Version** : 8.0.17  
**Owner** : fury  
**Created by** : app\_main  
**Owned by** : app\_main  
**Created at** : 2020-07-15  
**Traff encrypted** : yes  
**7x24** : **YES**

### schemas

**schema**    **conf** - **integ** - **avail**    **created at**    **created by**    **owned by**  
 fpPrngPro    LOW - LOW - HIGH    2020-07-15    app\_main    avernackt

### instances

instance	ip	role	datacenter
ip-10-54-9-13	10.54.9.13	MASTER	us-east-1d
ip-10-54-0-23	10.54.0.23	SLAVE	us-east-1a
mysql-lppricingapi00-1	10.86.0.211	SLAVE	us-east1-b

### slave status

instance	role	seconds lag	binlog file	binlog position	master binlog	master position	Channel Name
ip-10-54-0-23	SLAVE	0	mysql-bin.000366	854492078	mysql-bin.000366	854492078	
mysql-lppricingapi00-1	SLAVE	0	mysql-bin.000366	854492078	mysql-bin.000366	854492078	





## Create schema

Name	testaula
Description	base de prueba
Availability	High: Site application, productive 7x24
Integrity	High: Information that can't be altered (e.g. activity logs, account balance)
Confidentiality	High: Critical business information (e.g. credit cards, phones, passwords)
Test ⓘ	No
Core-metric 🗑 ⓘ	not-apply ✕
Core-metric-site 🗑 ⓘ	not-apply ✕

A new cluster will be created, please input your configuration:

Email	user@mercadolibre.com
Flavor	small

Perfect for: Credit card information, passwords, bank accounts

# Database

[Create schema](#)[Attach Existing](#)

## MySQL

schema	cluster	status	test	cloned
dbshield	dbaoperations00	approved		



## Metrics

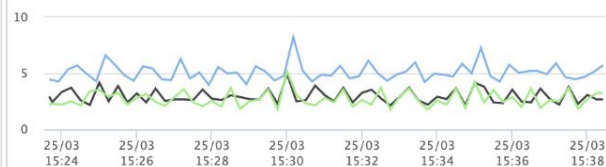
Database:

MySQL - dbshield ▾

The past 15 minutes ▾

☒ Hide legends

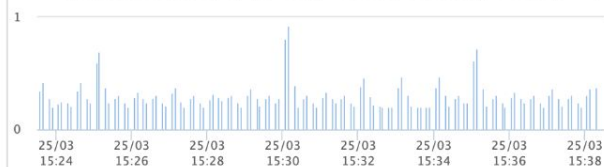
%CPU by Host



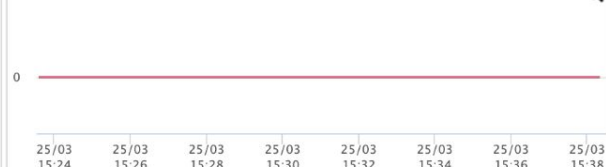
Connection limit % used by host



Operations (QTY/s) (Select / Insert / Update / Delete)



Lock Time





Slave Lag



Slow queries (QTY/s)



Fury


dba-operations-shield

Summary

^ Release

Pipelines

Versions

Deployments 

^ Ops

Service Map


**Infrastructure**

Doctor

Logs

^ Traffic Catalog

Summary

Dba-c

Last activities

SERVICE

updat

SERVICE

updat

DEPLOYMENT

scope

DEPLOYMENT

scope

DEPLOYMENT

fteijid

[View more](#)

Fury

dba-operations-shield

Summary

Release

Pipelines

Versions

Deployments

Ops

Service Map

Infrastructure

Doctor

Logs

Infrastructure

Search

prod

2

prod

Information updated at 15:40hs  
[View events in Doctor](#)

DATADOG

New Relic

Kibana

Current auto scaling group

Instances: 2 Min: 2 Max: 100 Desired: 2 ELB: 2 Boost

<input type="checkbox"/>	Instance	Status	Health ↓	Uptime	CPU	Load	Free Heap	Steal	Commands
<input type="checkbox"/>	i-0077a2ea148881684 - 10.50.131.180	InService		C	C	C	C	C	Terminal



# Aspectos claves



- Metadata
- Criticidad
- Métricas core
- PII
- Rate Limits

## // 02

# ¿Cómo lo integro con mi código?

IT BOARDING

**BOOTCAMP**



**Vamos a verlo directamente en Fury!**

## Aspectos claves



- Los snippets son guías
- La implementación final puede (y probablemente variará) particularmente para RDBMS, pero muy probablemente para NoSQL también
- Mantener las librerías actualizadas y darle seguimiento a los pedidos de cambios de los servicios es clave para mantener la seguridad y performance
- Siempre revisar la documentación y consultar con tu equipo

## // 03

# Métricas Core/ Backups & Mirrors

IT BOARDING

**BOOTCAMP**

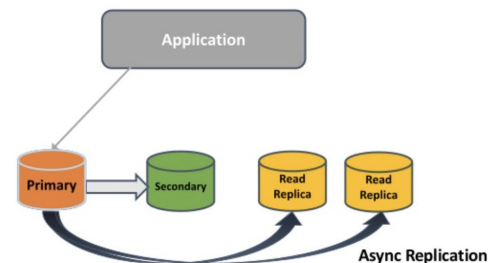
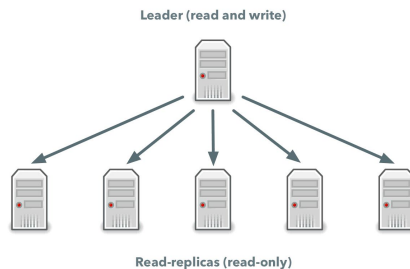
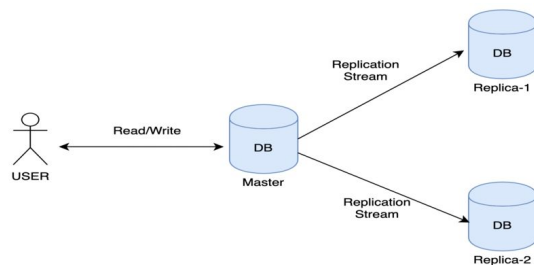


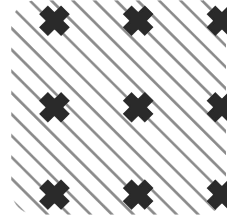
## **Agregar los tags apropiados es fundamental!**

- Las métricas core son aquellas métricas que miden la salud de los negocios principales de MELI y aplicaciones que generan afectación sobre estas métricas en caso de caída, tienen características de cuidado particulares
- Apps con métricas core tienen, dependiendo del servicio, mejores o mayores garantías en términos de respaldos, tiempos de restore, alta disponibilidad, tiempos de respuesta, etc. (Y entonces... porqué no para todos!?)
- Cuando hay una afectación, se priorizan aquellas apps que afectan métricas core
- Cuando hay un release se dejan para lo último las apps que afectan métricas core

## Mirrors / Slaves / Secondaries

- Algunos servicios tienen réplicas (habitualmente inconsistentes) con respecto a sus primarios
- Esto agrega una capa de redundancia y alta disponibilidad, mientras que permite segmentar el tráfico según tipo o prioridad
- Se puede redireccionar tráfico en caso de pérdida del master
- Hay varios modelos en función de diferentes categorizaciones
  - Activo / Activo
  - Activo / Pasivo
  - Escritura-Lectura / Lectura
  - Sync replica / Async replica
  - Write local - read global / Write local - read local
  - Etc





## Backup / Restore

- Si bien puedes hacer vos mismo backup en algunos servicios, en general se hacen automáticamente
- KVS y bases de datos soportan PITR. Esto puede tener alguna pérdida de datos de 1 o 2 minutos (CDC podría ayudar)
- Object Storage no hace backup, pero tiene histórico de version y cross region replication
- DS hace backups una vez por día y guarda el diff en BigQueue para re-hidratar en caso de ser necesario. También la data está replicada en los mirrors.
- Audits también tiene cross region replication

## Aspectos claves

- Es fundamental estar al tanto de los tiempos de restore. Si tu DB o servicio crece de forma no sana (demasiado o demasiado sobre una cierta partición) convendrá analizar si es mejor hacer un sharding porque las consultas y el restore serán lentos





# // 04

## Métricas resultan relevantes y cómo verlas?

IT BOARDING

**BOOTCAMP**

# Métricas

- ¿Qué métricas nos importan?
- ¿Porqué?
- ¿Son las mismas para una base SQL que para un KVS o un DS?





**Vamos a verlo directamente en Fury!**



**Vamos a verlo directamente en  
Datadog!**



**Vamos a verlo directamente en  
NewRelic!**

## Aspectos claves

- En caso de errores es fundamental que ENTIENDAS correctamente lo que significa y te quiere decir cada métrica. Para esto es fundamental que las analices apropiadamente
- Es interesante hacer pruebas de carga y drills de eventos de afectación de tu app para que todos estén familiarizados con las métricas y los logs de tu app.
- La mayoría de las veces el problema es del lado aplicativo
- Un timeout contra un servicio no necesariamente quiere decir que el servicio está mal, hay que analizar las instancias de la app que origina el tráfico
- Entender bien qué métricas hay para evitar duplicarlas (\$\$)
- Atender a las opportunities o services police

# // Hora de ejercicios!!

IT BOARDING

**BOOTCAMP**

# RDBMS

- No vamos a hacer ejercicios sobre bases relacionales porque eso lo verán profundamente durante los próximos días



## En sus apps

Asumiendo que tienen un KVS cuya variable de ambiente es:

```
"KEY_VALUE_STORE_MY_CONTAINER_CONTAINER_NAME"
```

1. Implementar en Java una clase "User" que tenga varios campos como id, nombre, apellido, género, fecha de nacimiento
2. Implementar una clase que permita hacer el CRUD de ese user en KVS
3. Asegurarse que el KVS tenga 2 reintentos
4. Asegurarse que si se pasa del rate limit, se loguee la información correspondiente




## En sus apps

Asumiendo que tienen aquel KVS tiene un DS asociado cuyo nombre es `my-container-ds`

1. Implementar una clase que permita hacer consultas contra ese DS para los users
2. Hacer un método que devuelva los usuarios de un país





Los invitamos a completar la  
siguiente encuesta sobre el  
módulo Storage

**¡Es muy muy importante para  
nosotros contar con su  
feedback!**

Solamente les tomará unos  
minutos completarla :)



[Link a la encuesta](#)





# Gracias.

IT BOARDING

**BOOTCAMP**

