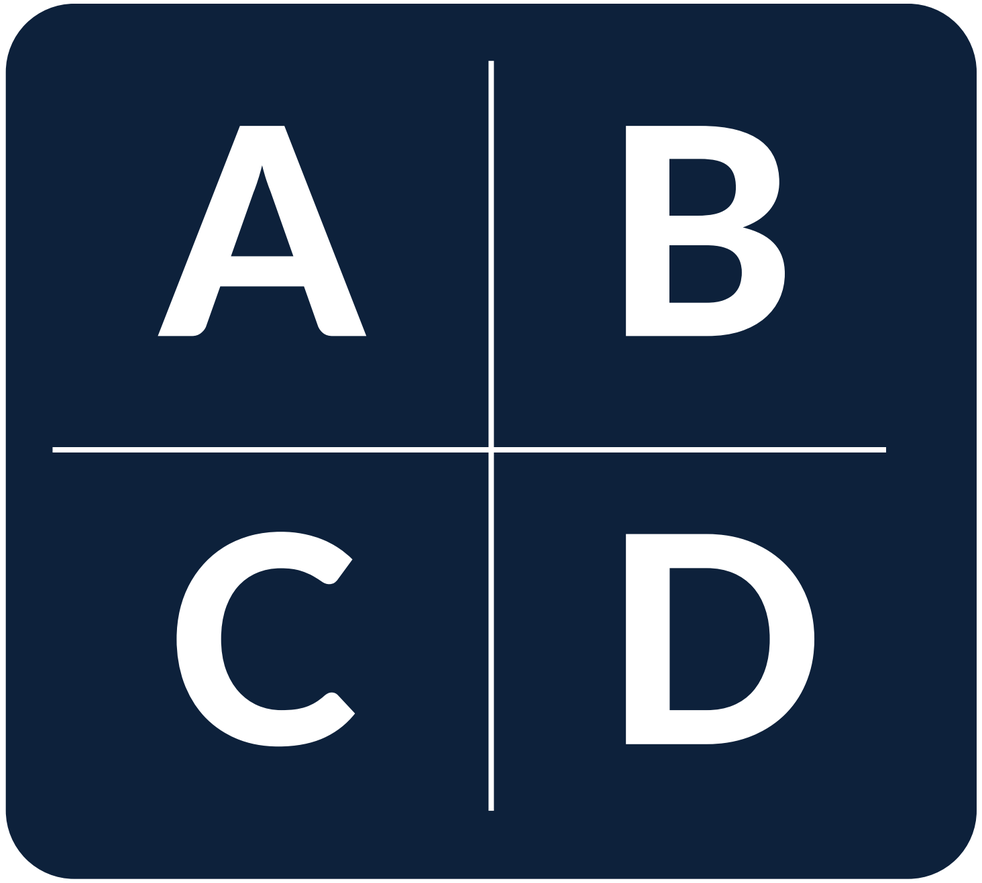
{dev/talles}

MICROSERVICIOS

UNA COMPARATIVA NEUTRAL

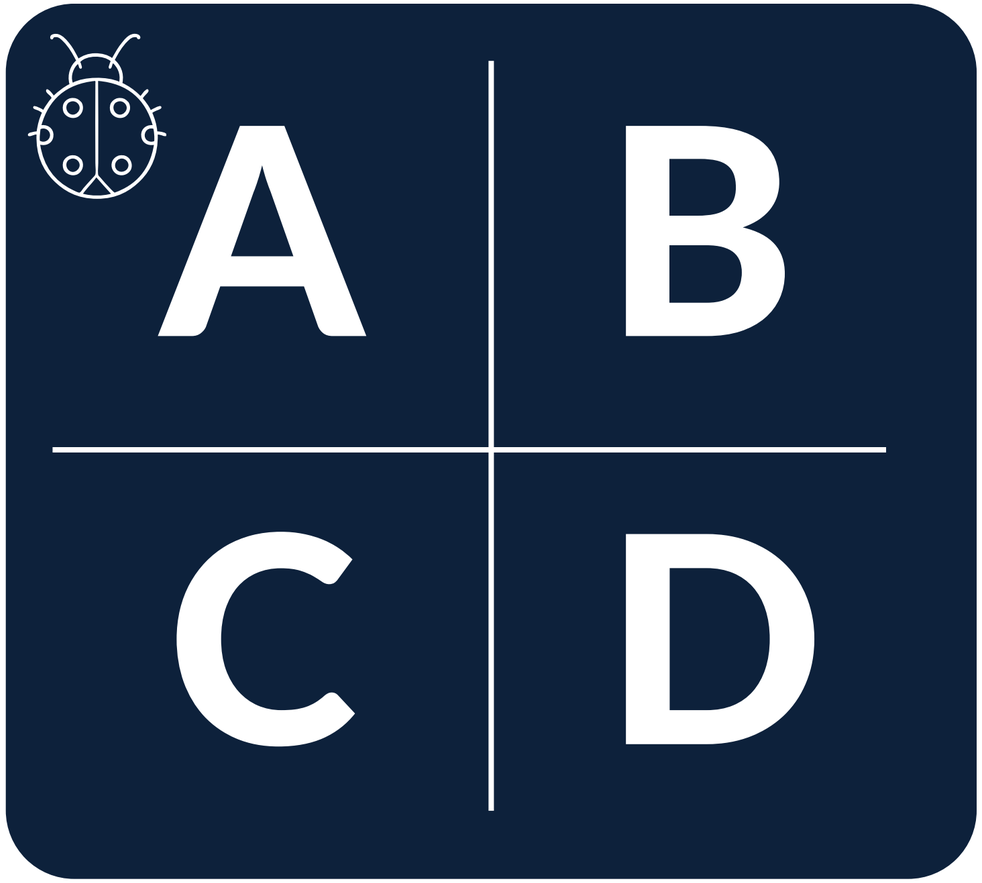
TABLA DE CONTENIDO

MONOLITICO
TRADICIONAL



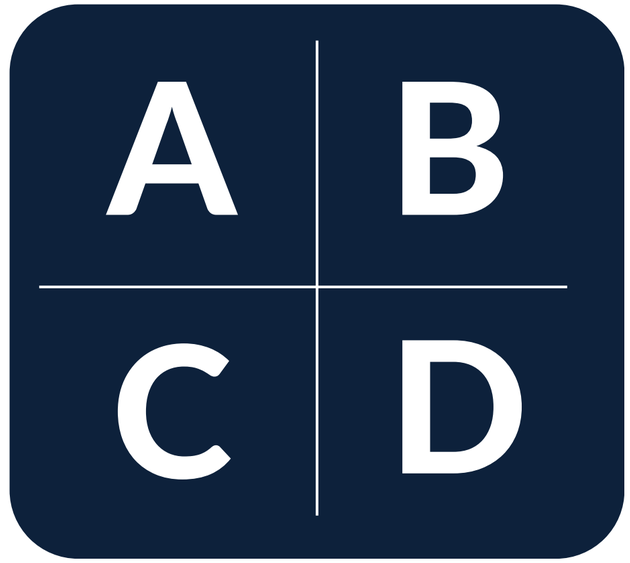
CONS

MONOLITICO
TRADICIONAL



CONS

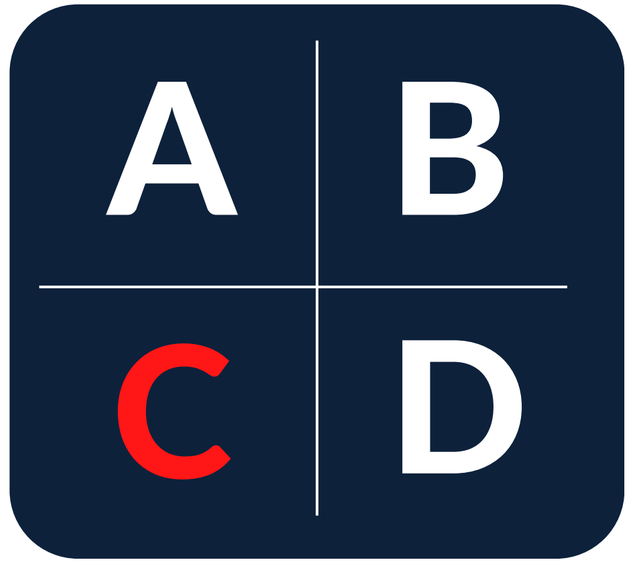
PUNTOS A CONSIDERAR



MONOLITICO

CONS

PUNTOS A CONSIDERAR



Mucho estrés

No podemos escalar C

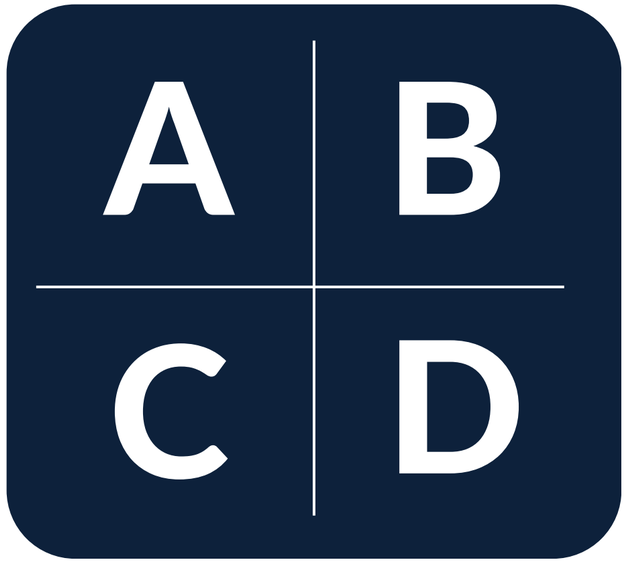
Sin escalar todo el monolítico

MONOLITICO

ESCALAMIENTO VERTICAL

CONS

PUNTOS A CONSIDERAR



D V2.0

Nueva versión

MONOLITICO

CONS

PUNTOS A CONSIDERAR

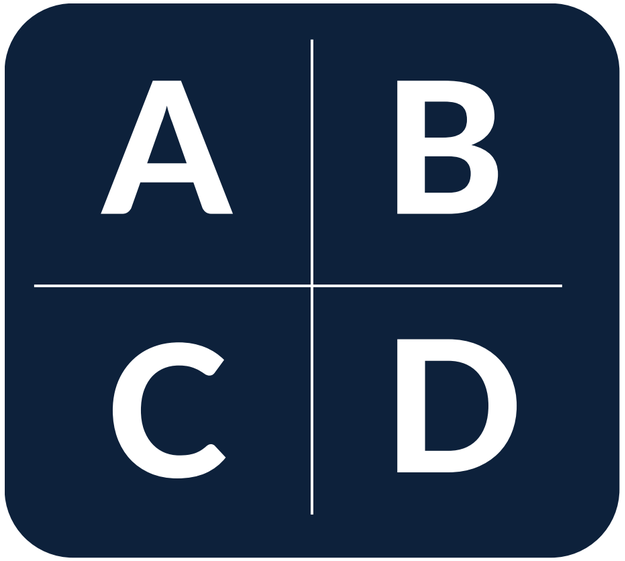
No se puede actualizar D

Sin redesplegar A,B y C también

MONOLITICO

CONS

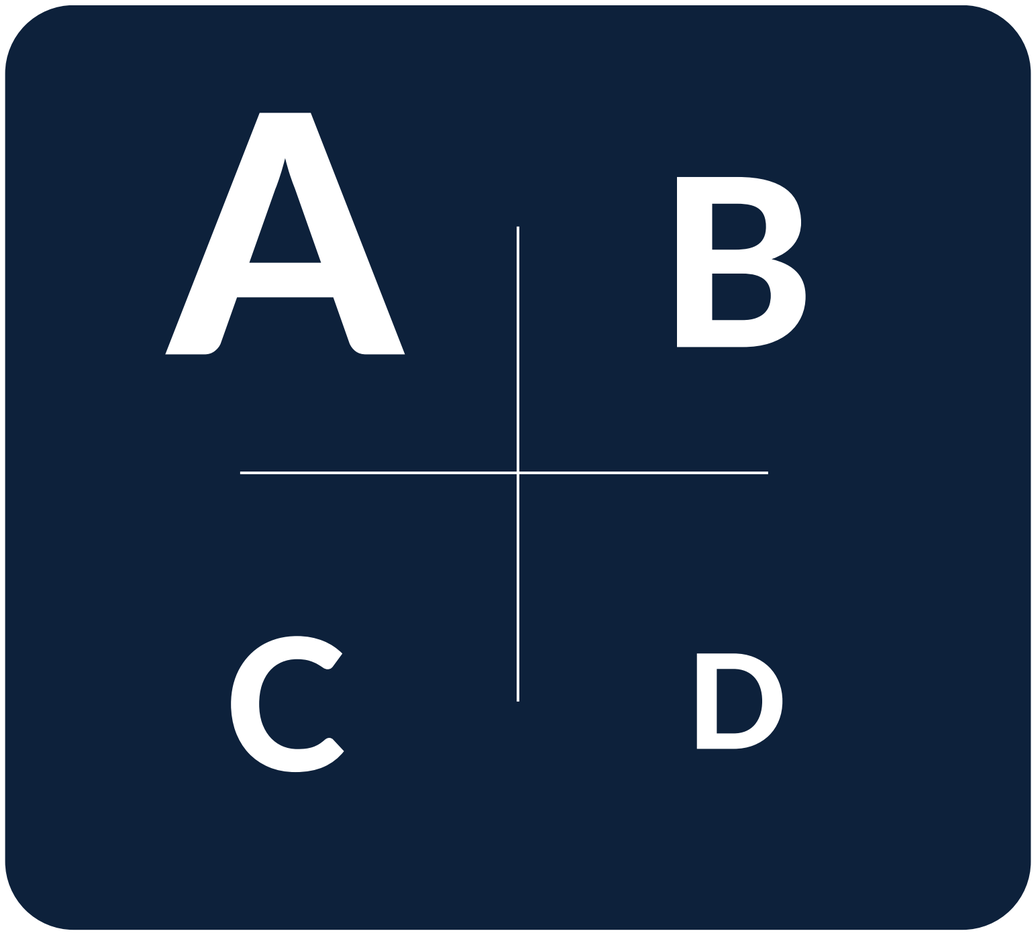
PUNTOS A CONSIDERAR



MONOLITICO

CONS

PUNTOS A CONSIDERAR

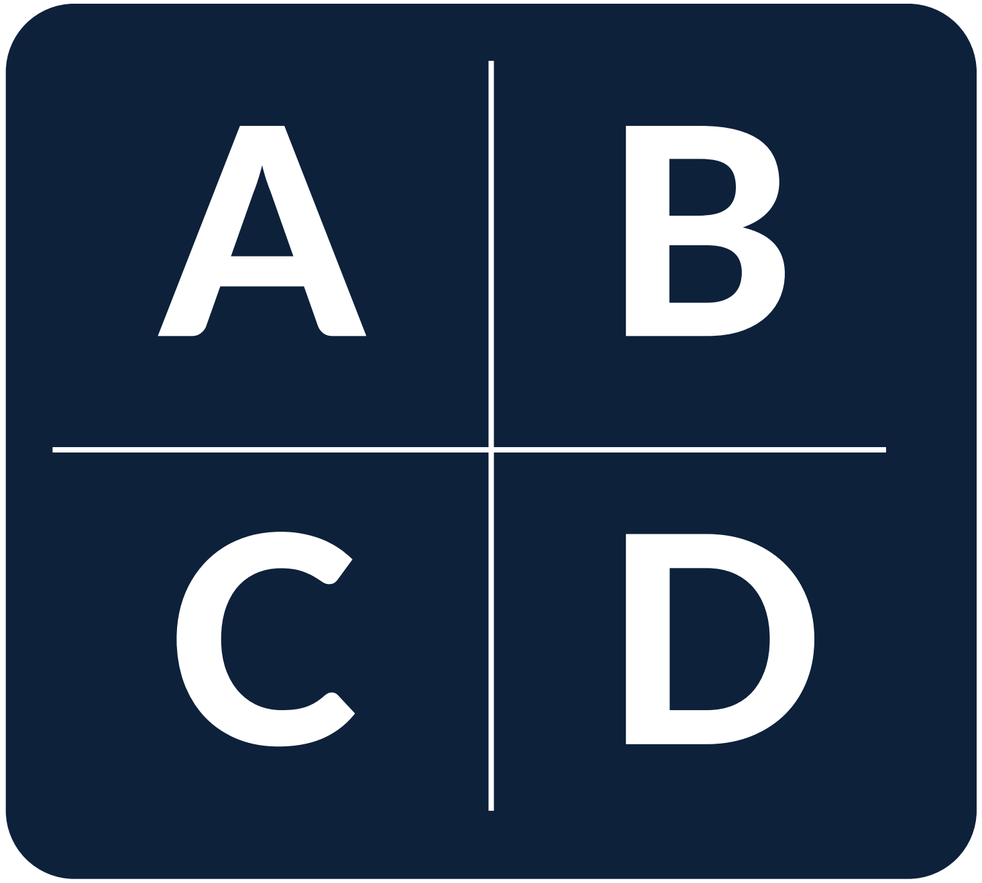


- Código base va creciendo.
- Difícil de ejecutar localmente.

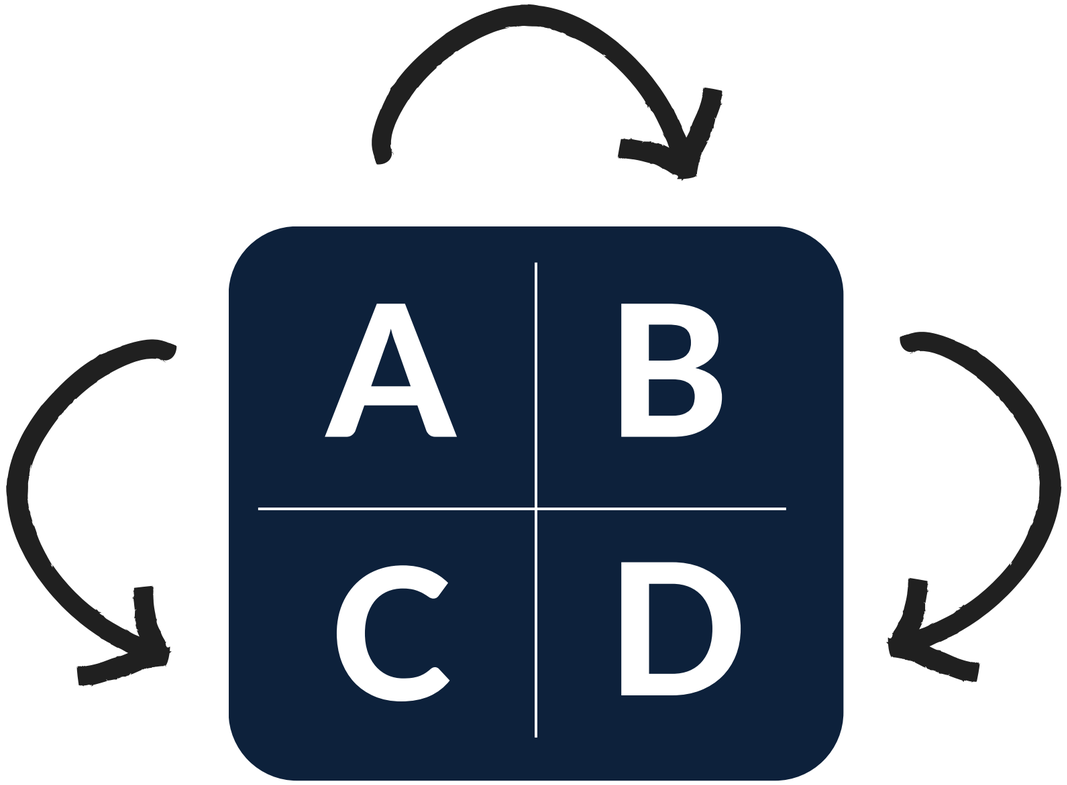
MONOLITICO

PROS

MONOLITICO
TRADICIONAL



PROS

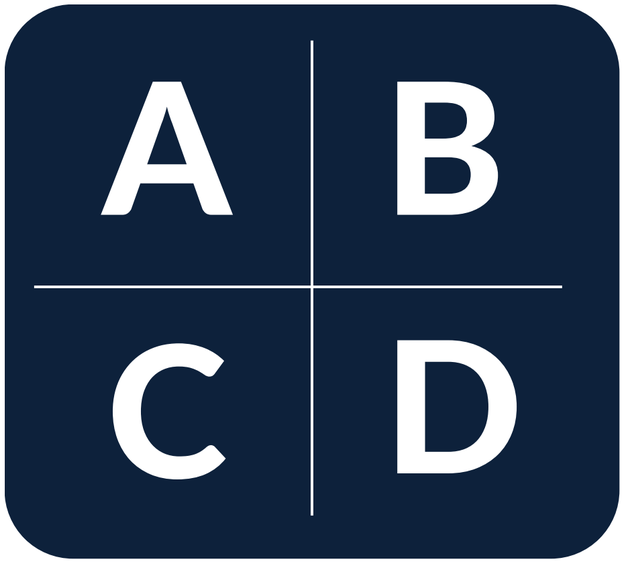


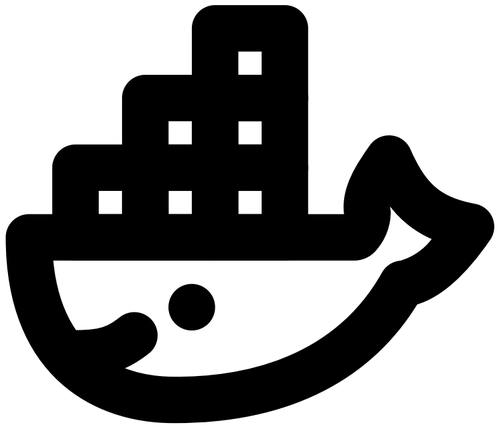
Cero latencia

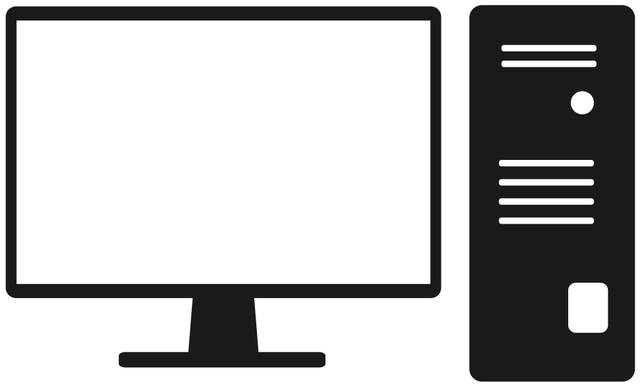
COMUNICACIÓN INSTANTÁNEA ENTRE LAS PARTES

MONOLITICO

PROS





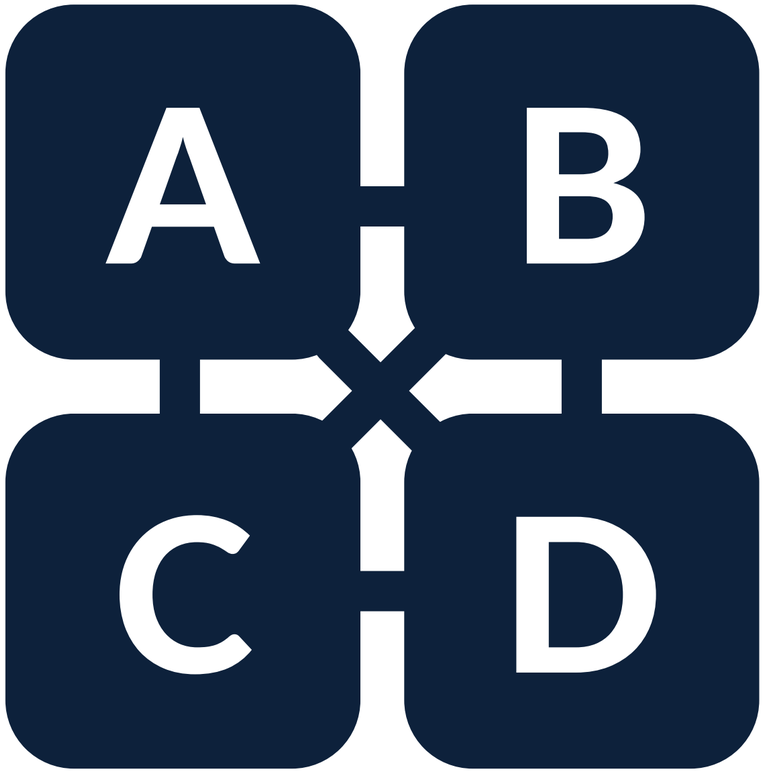


MONOLITICO

Montar todo en una máquina o contenedor

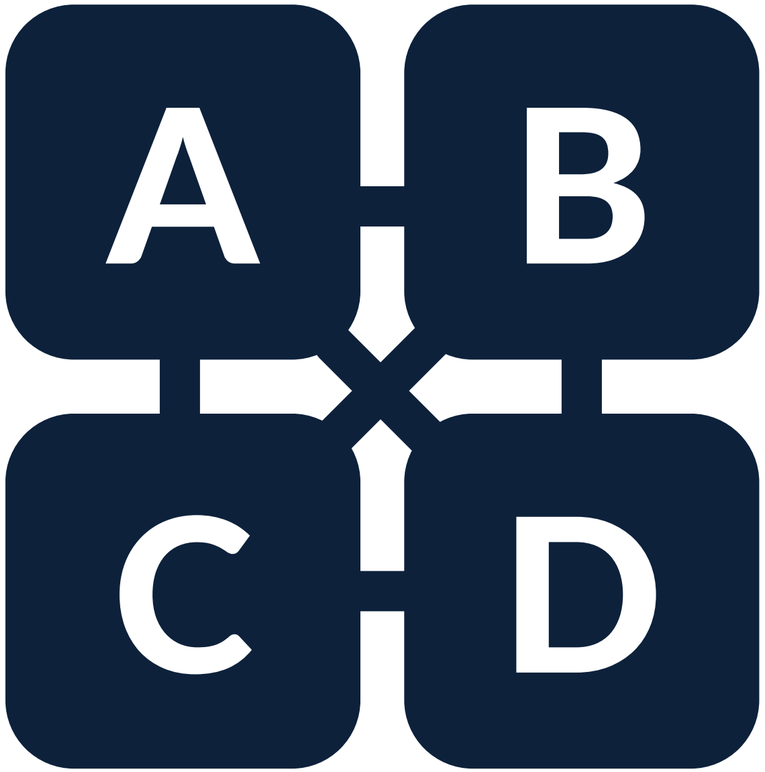
MICROSERVICIOS

PROS Y CONS



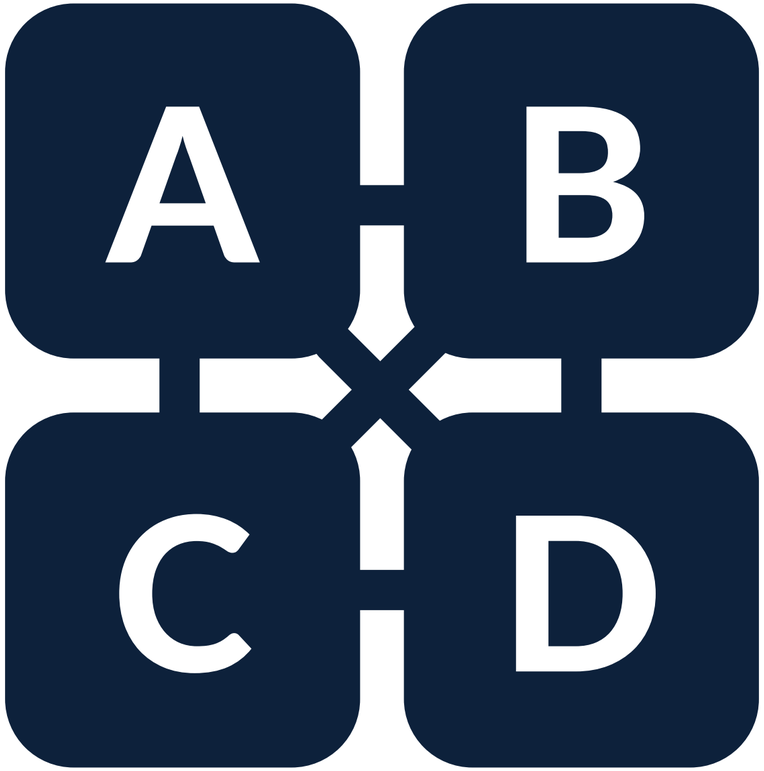
MICROSERVICIOS

NO ES UNA BALA DE PLATA



Los microservicios resuelven algunos problemas pero incorporan otros

PROS



PROS

A

B

TODOS LOS MICROSERVICIOS SON INDEPENDIENTES, ESCALABLES Y DISTRIBUÍDOS

C

D

PROS

A

B

C

D

PROS

PROS

A

B

C

D

PROS

A

B

C

D

PROS

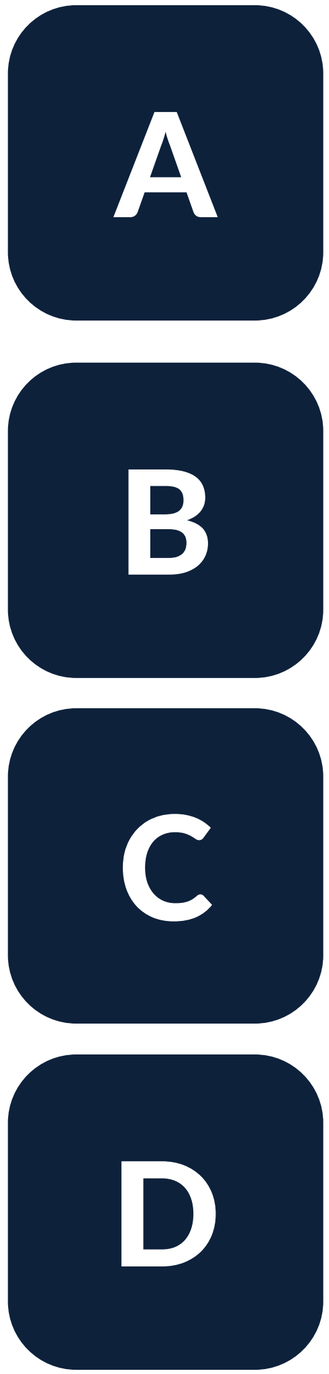
A

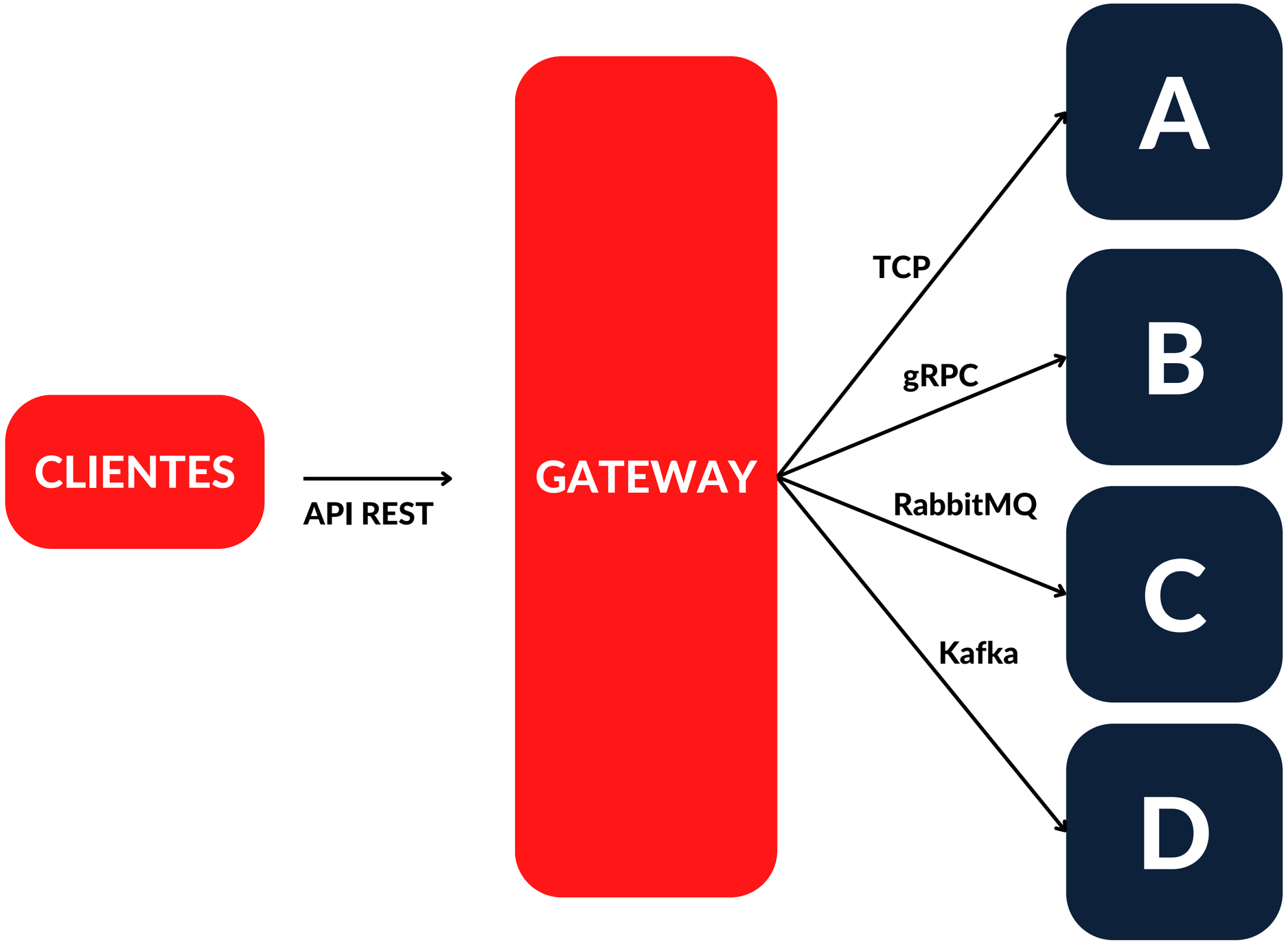
B

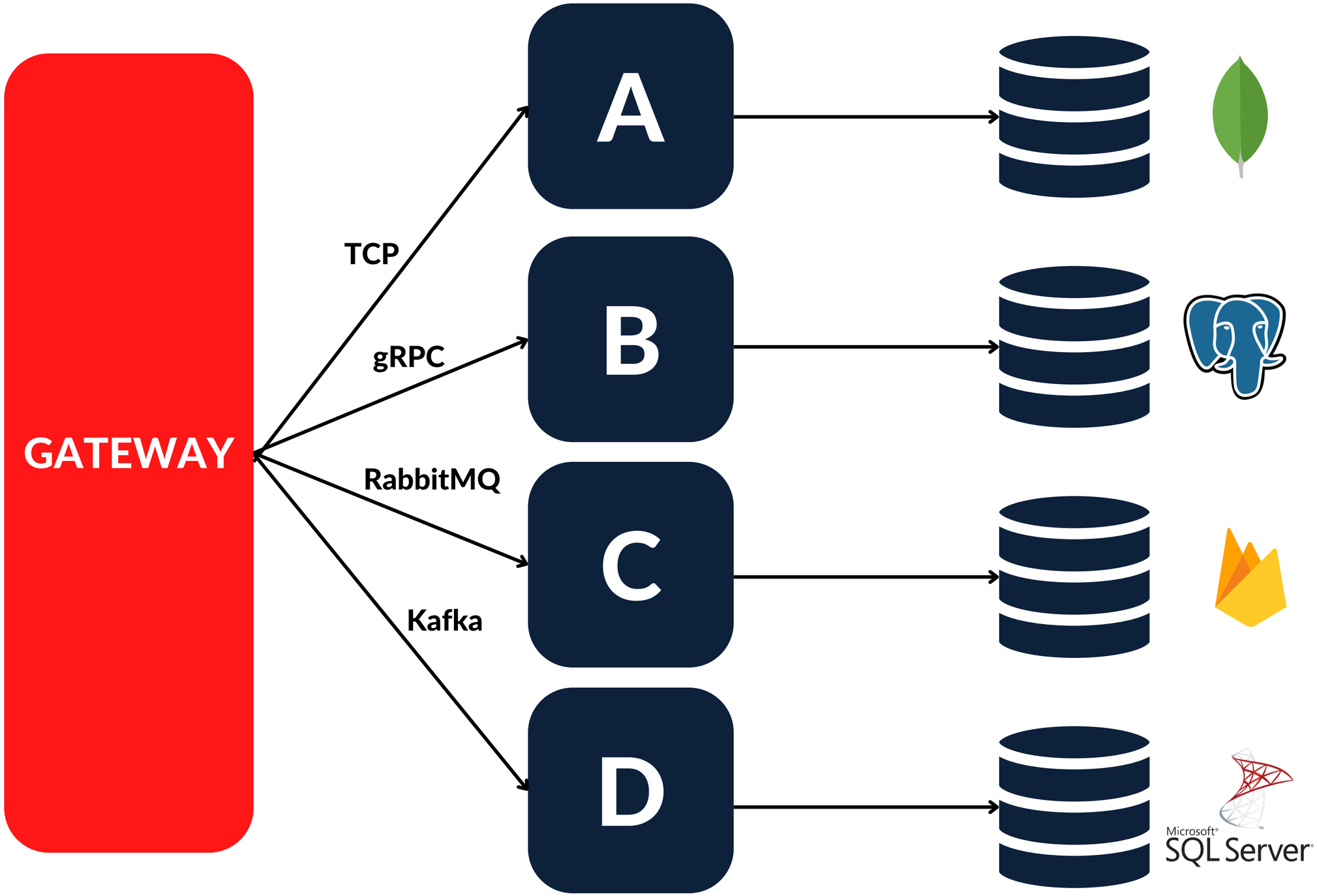
PUEDEN ESTAR AISLADOS Y COMUNICARSE MEDIANTE REST,
SER HÍBRIDOS O USAR UN CANAL DE COMUNICACIÓN ESPECÍFICO

C

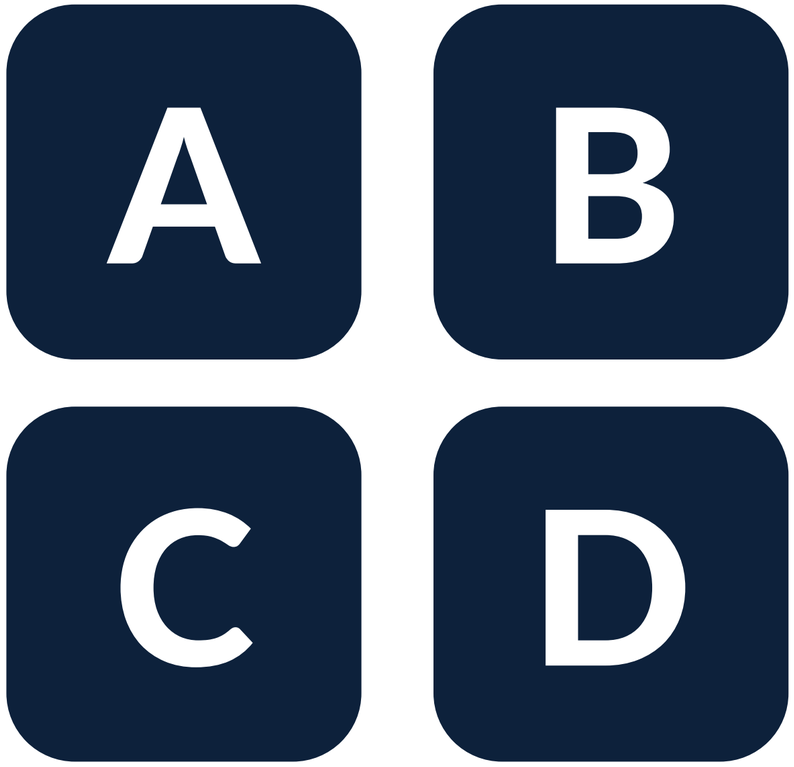
D


A
B
C
D


CLIENTES
API REST
GATEWAY
TCP
gRPC
RabbitMQ
Kafka
A
B
C
D


GATEWAY
TCP
gRPC
RabbitMQ
Kafka
A
B
C
D
Microsoft
SQL Server

CONS



F

A

B

CONS

C

J

G

K

H

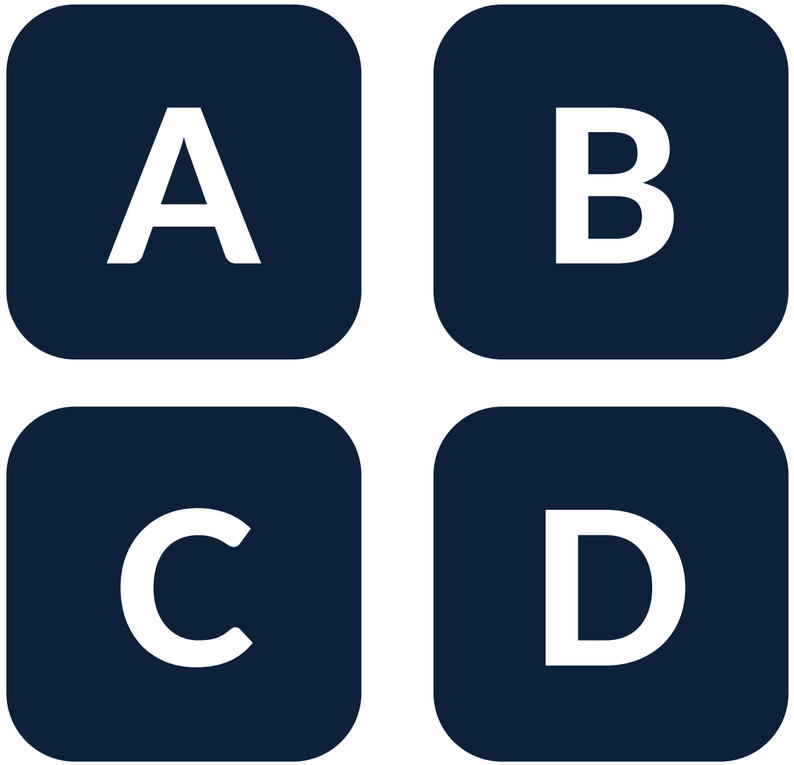
I

D

E

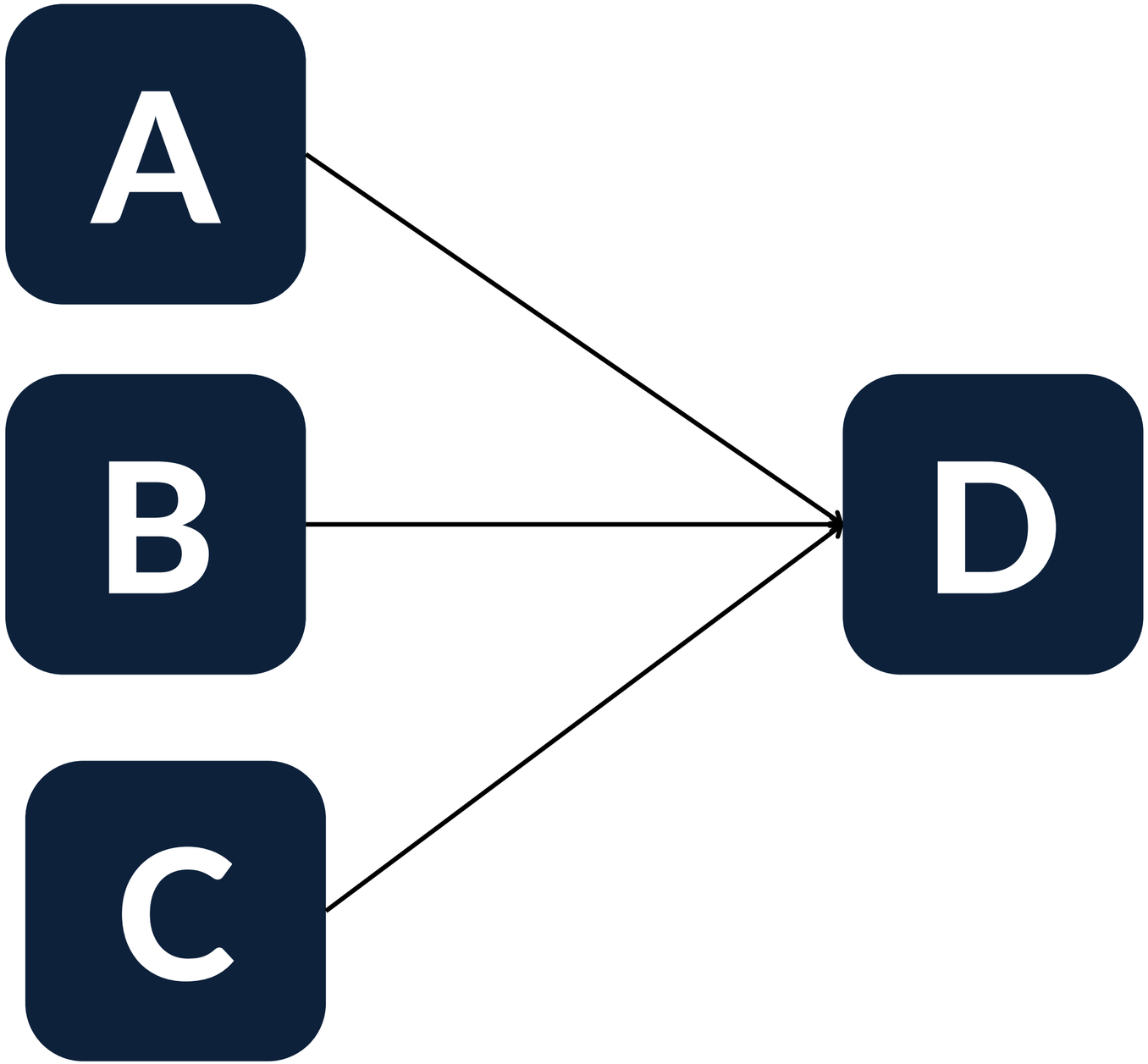
ENCONTRARLOS

CONS

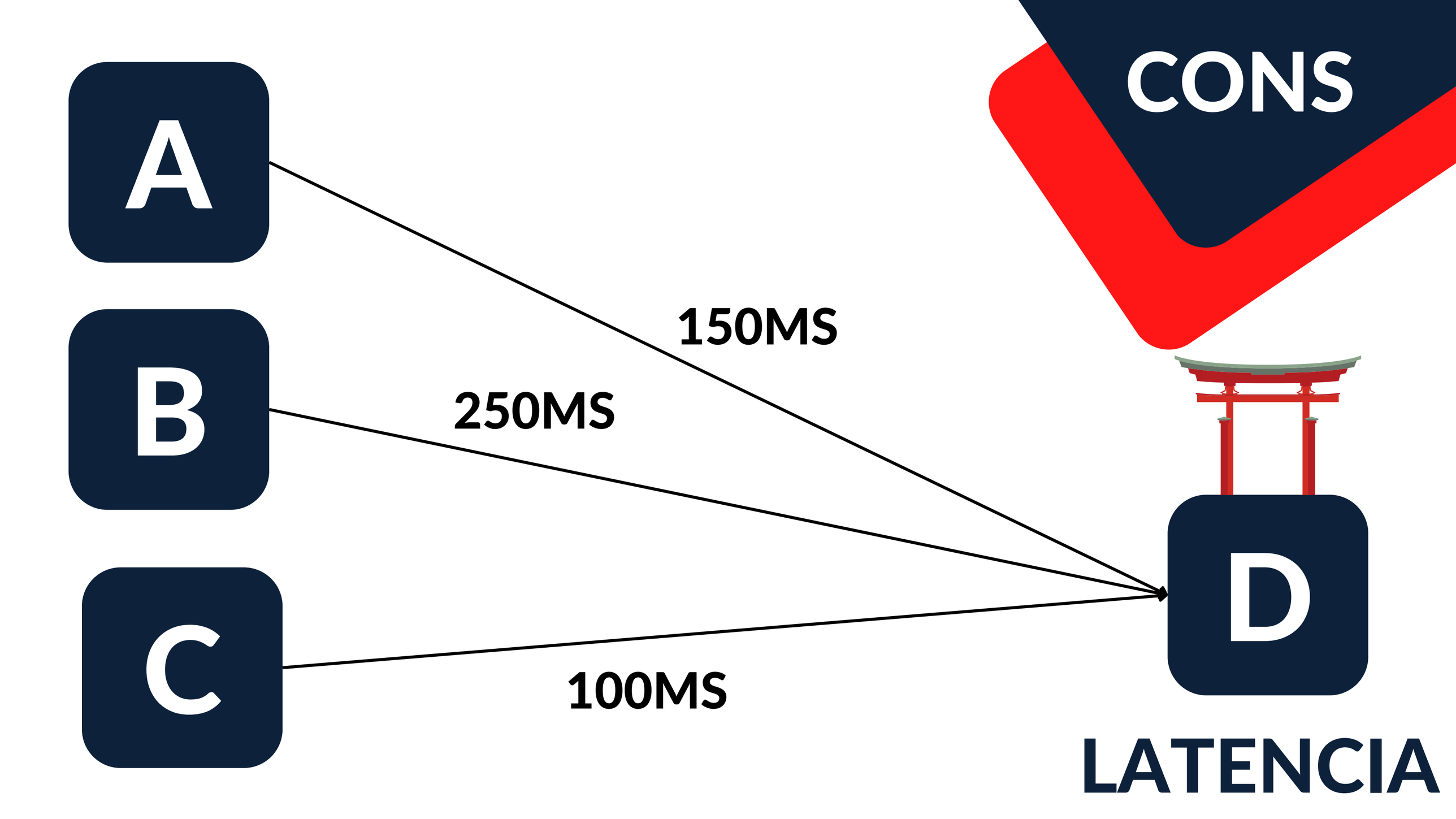


LATENCIA

CONS



LATENCIA


CONS
A
B
C
D
150MS
250MS
100MS
LATENCIA

SI SE REQUIERE ALGO DE
LOS SERVICIOS A,B,C

CONS

A

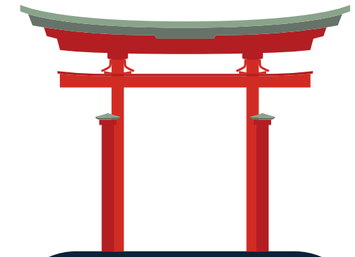
B

C

150MS

250MS

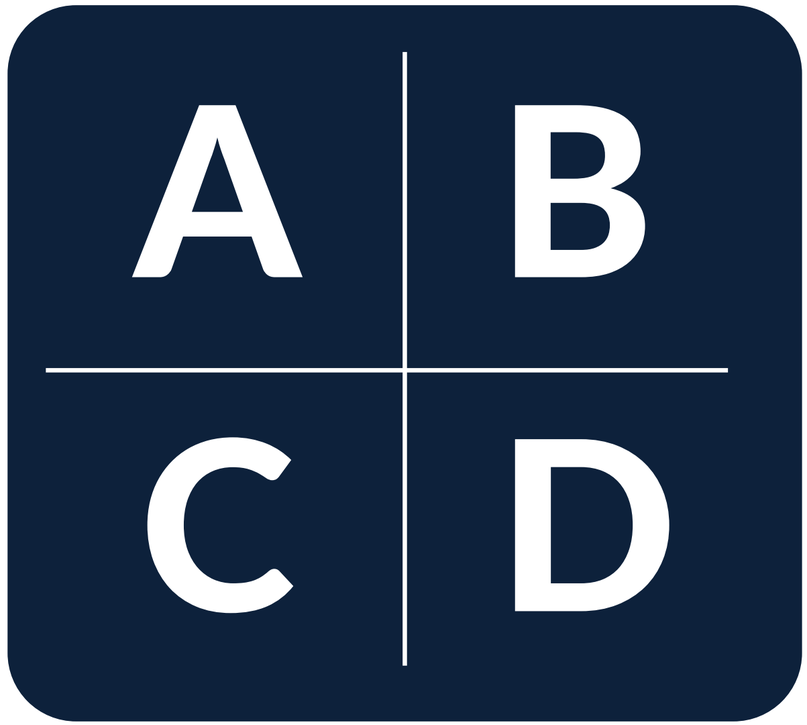
100MS

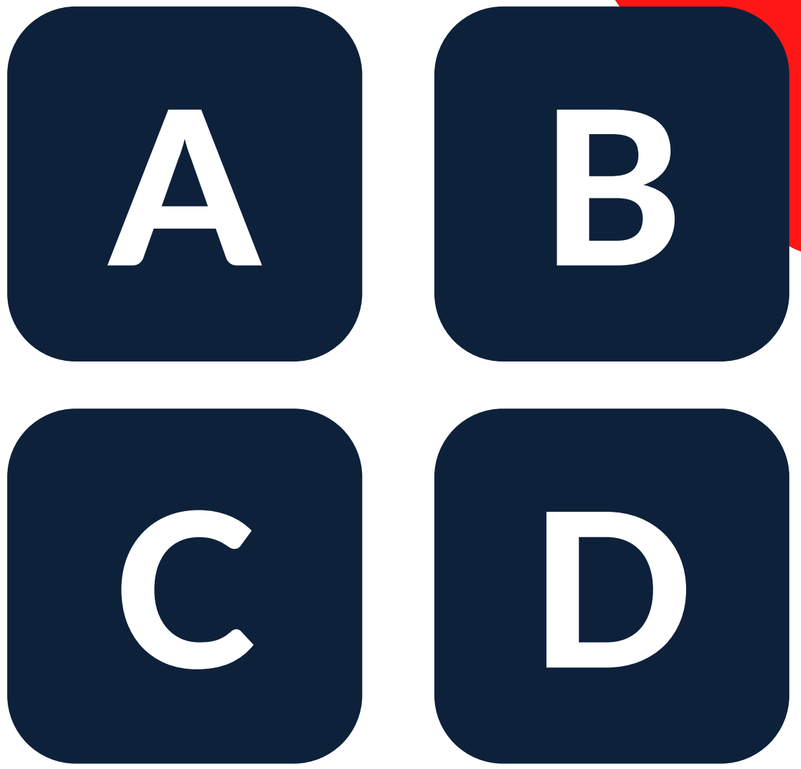


D

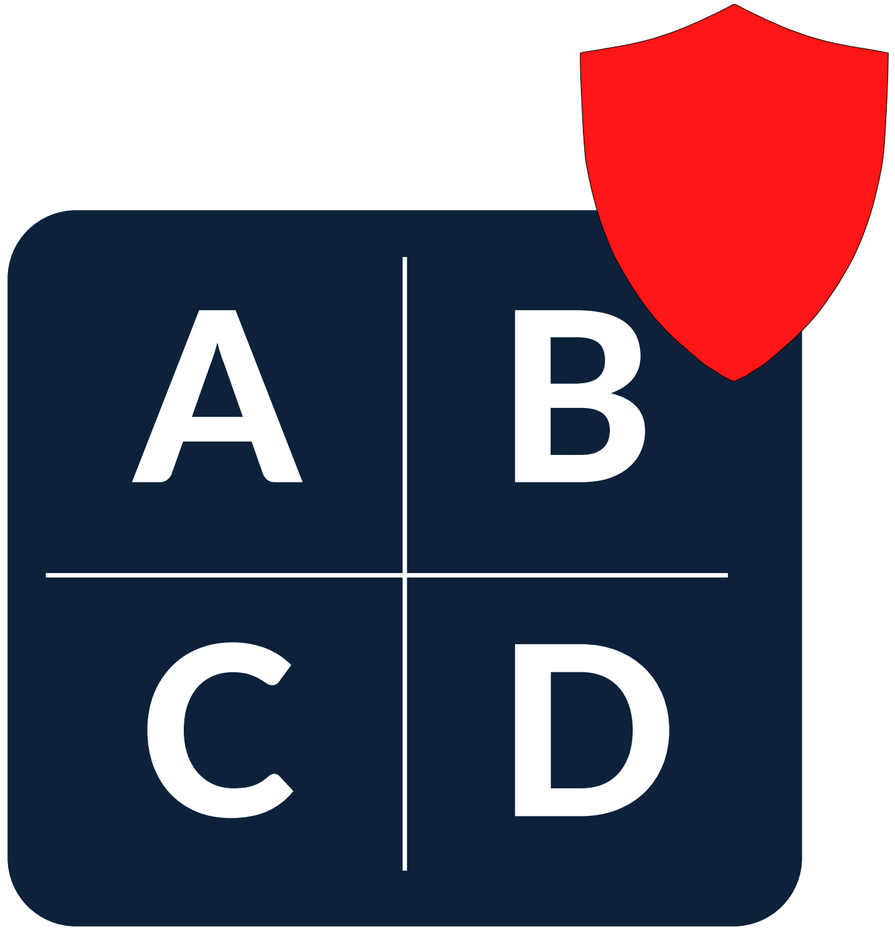
LATENCIA

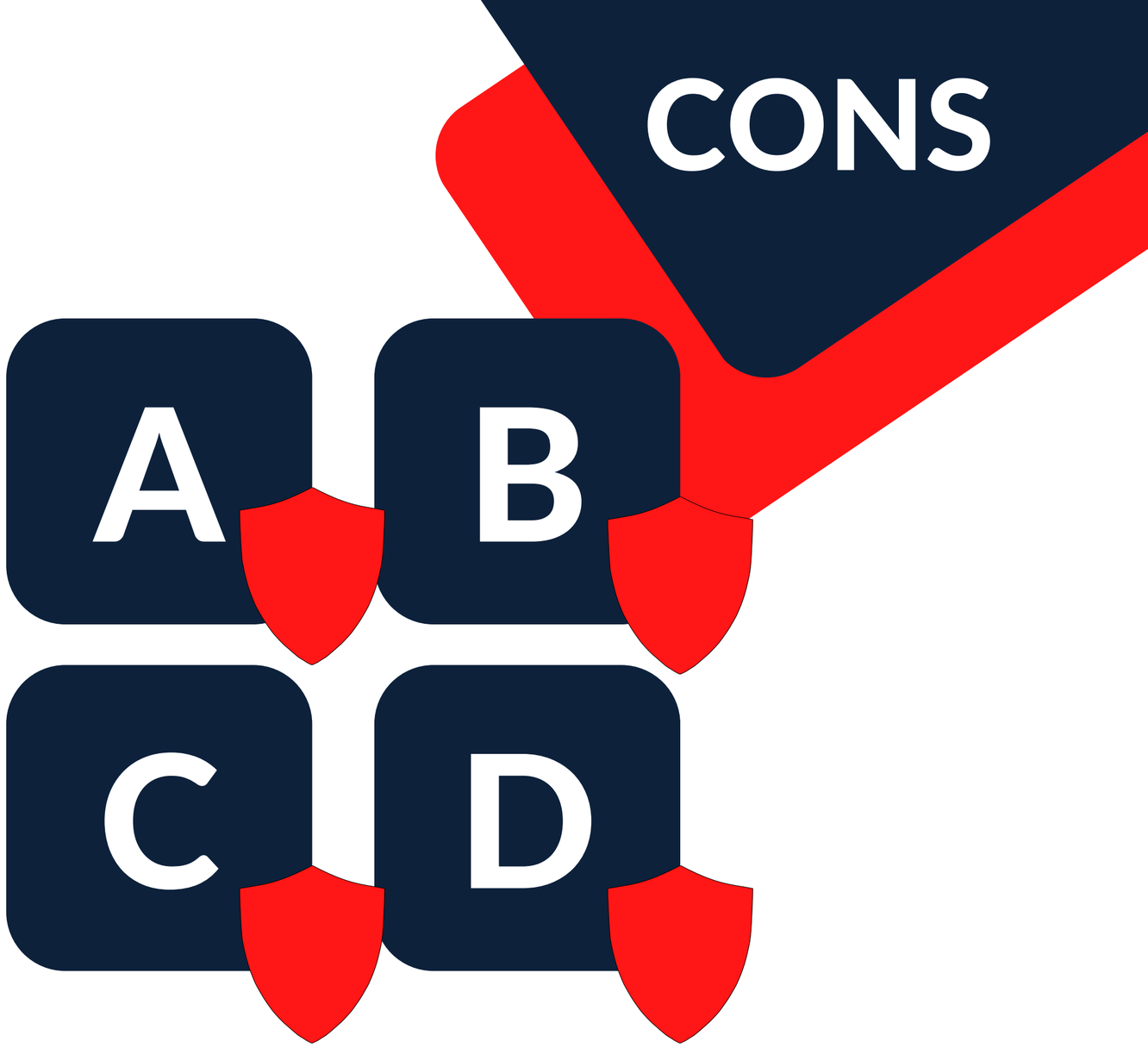
CONS





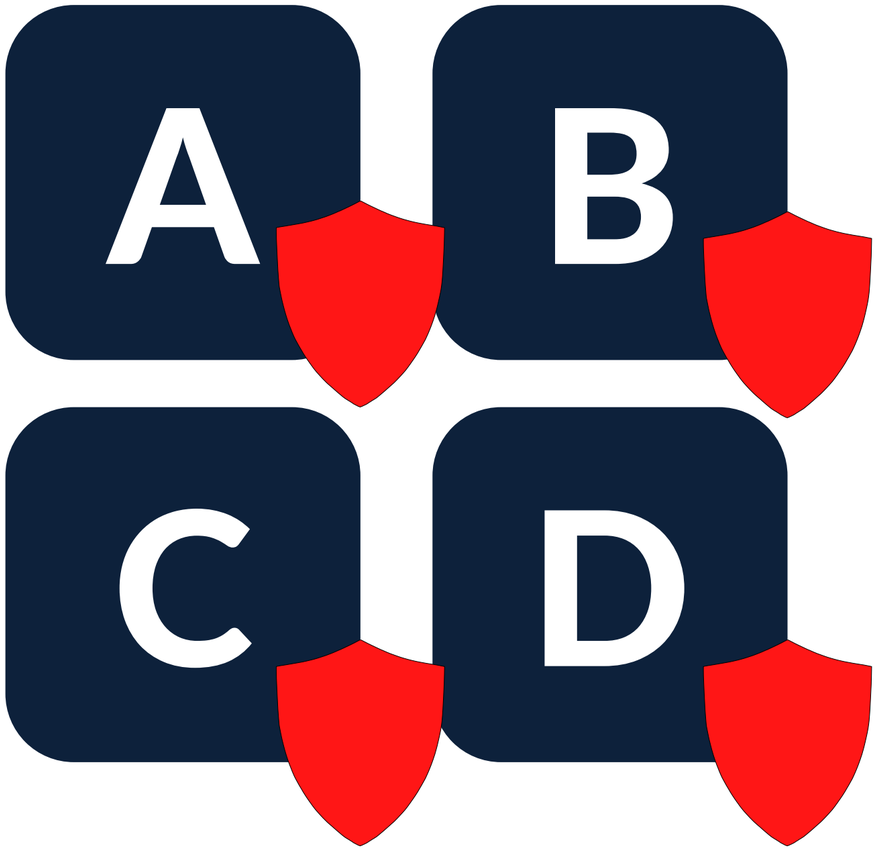
SEGURIDAD





SEGURIDAD

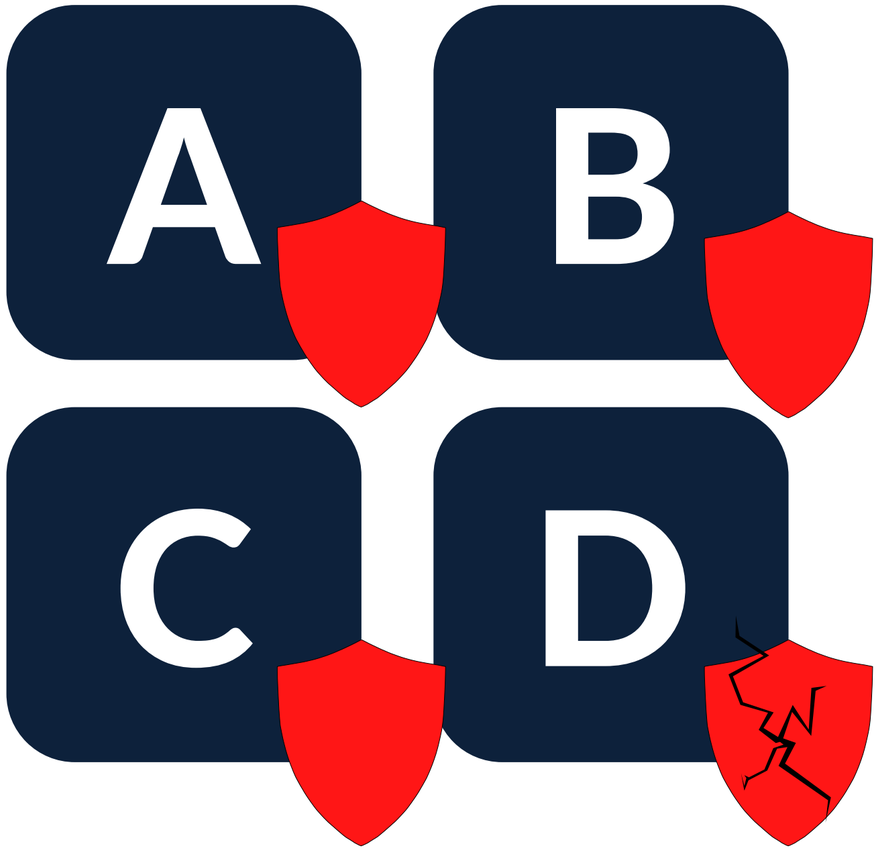
CONS



SEGURIDAD

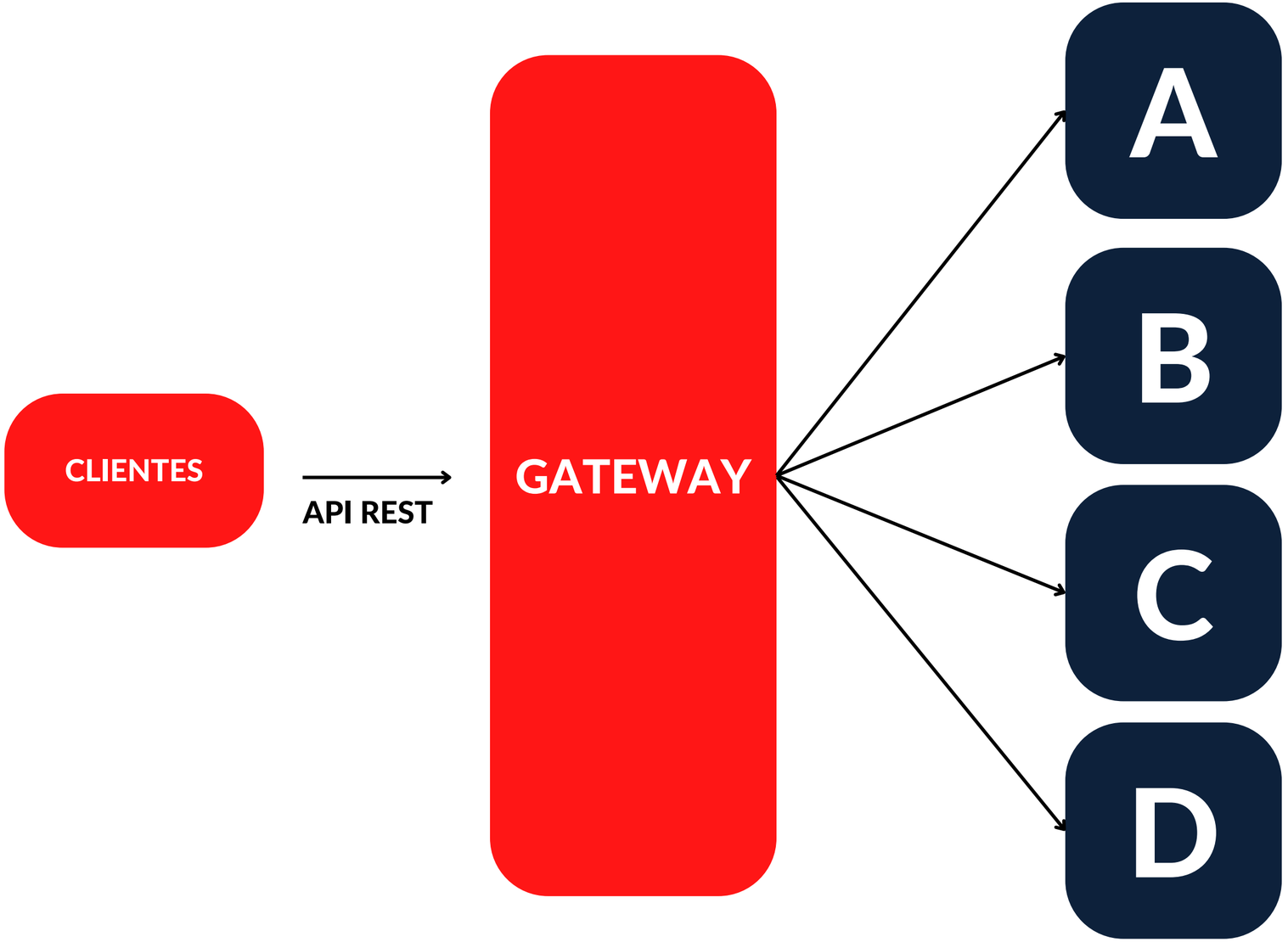
Mayores puntos de quiebre.

CONS



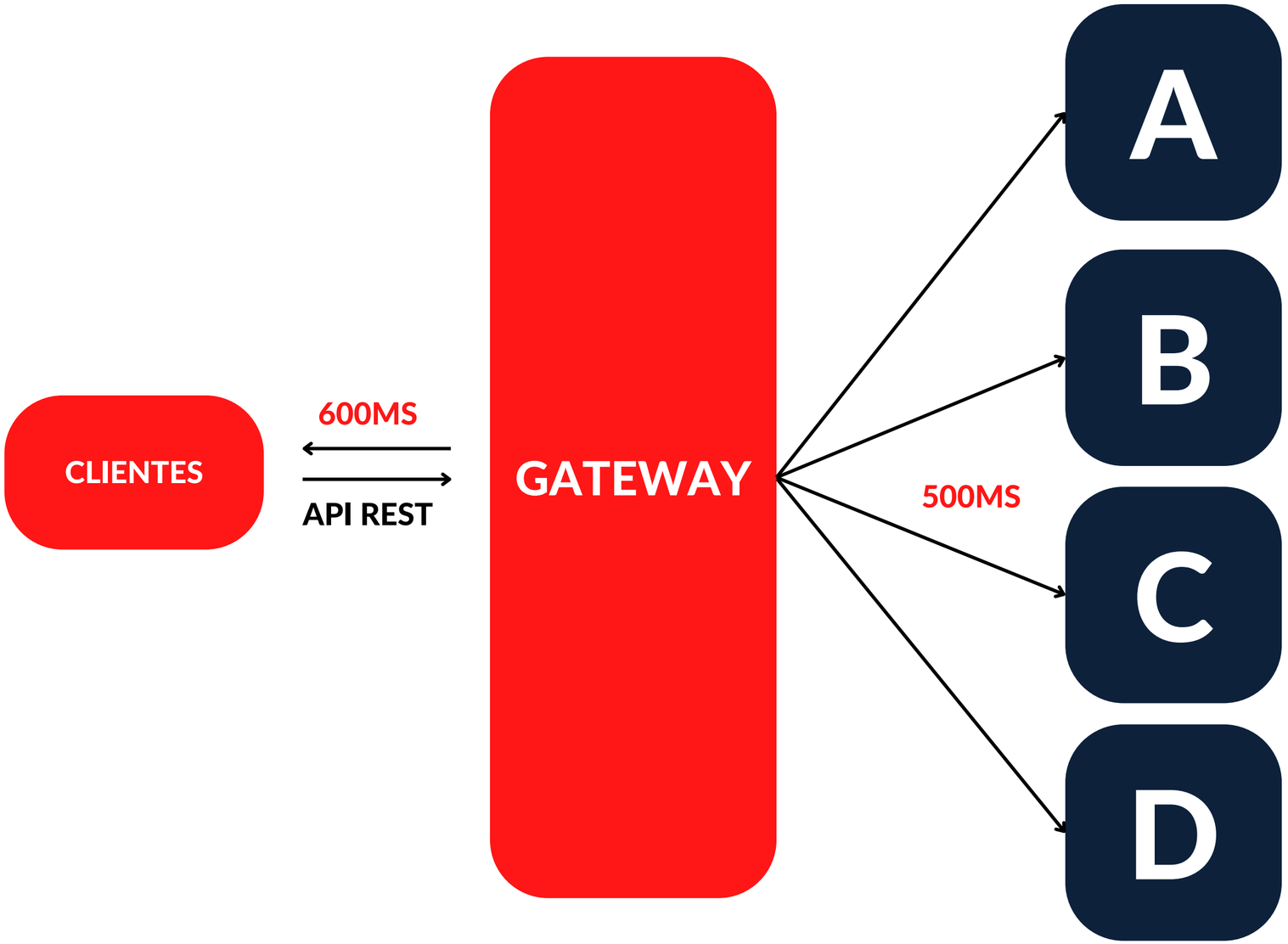
SEGURIDAD

CONS



MANEJO DE ERRORES Y PROBLEMAS

CONS



MANEJO DE ERRORES Y PROBLEMAS

TIPOS DE TRANSPORTE
MICROSERVICIOS

Al final del día, todos tienen por objetivo enviar y/o recibir información entre ellos o entes externos.

- **HTTP/HTTPS:** Restful
- **gRPC:** Protocolo de buffers creado por Google
- **Queues o colas:** Mensajería tipo oficina postal, ejemplos como RabbitMQ, Apache Kafka y Amazon SQS.
- **Flujo de eventos (Streams):** Eventos pueden ser consumidos por multiples microservicios interesados. Ejemplos como Apache Kafka, Apache Pulsar y AWS Kinesis.
- **TCP/UDP:** Eficiente comunicación entre equipos en el cuarta capa del modelo OSI, confiable y ordenada.

BUENAS PRACTICAS MICROSERVICIOS

- **Descomposición adecuada**
- **Responsabilidad única**
- **Tamaño adecuado**

- **Independencia**
- **Autonomía**
- **Comunicación asíncrona**

Puede llevar a duplicidad, pero

- **Escalamiento independiente**
- **Despliegues independientes**

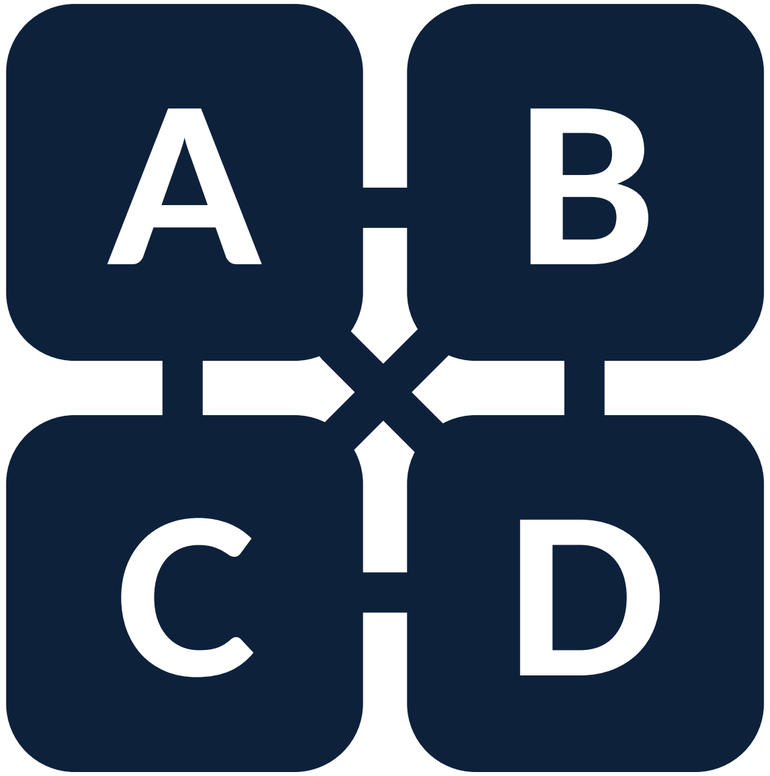
Duplicidad o problemas de integridad referencial, pero

- **Tolerancia a errores**
- **Recuperación automática**

- **Seguridad independiente**
- **Autenticación, autorización, cifrado**

PERO *

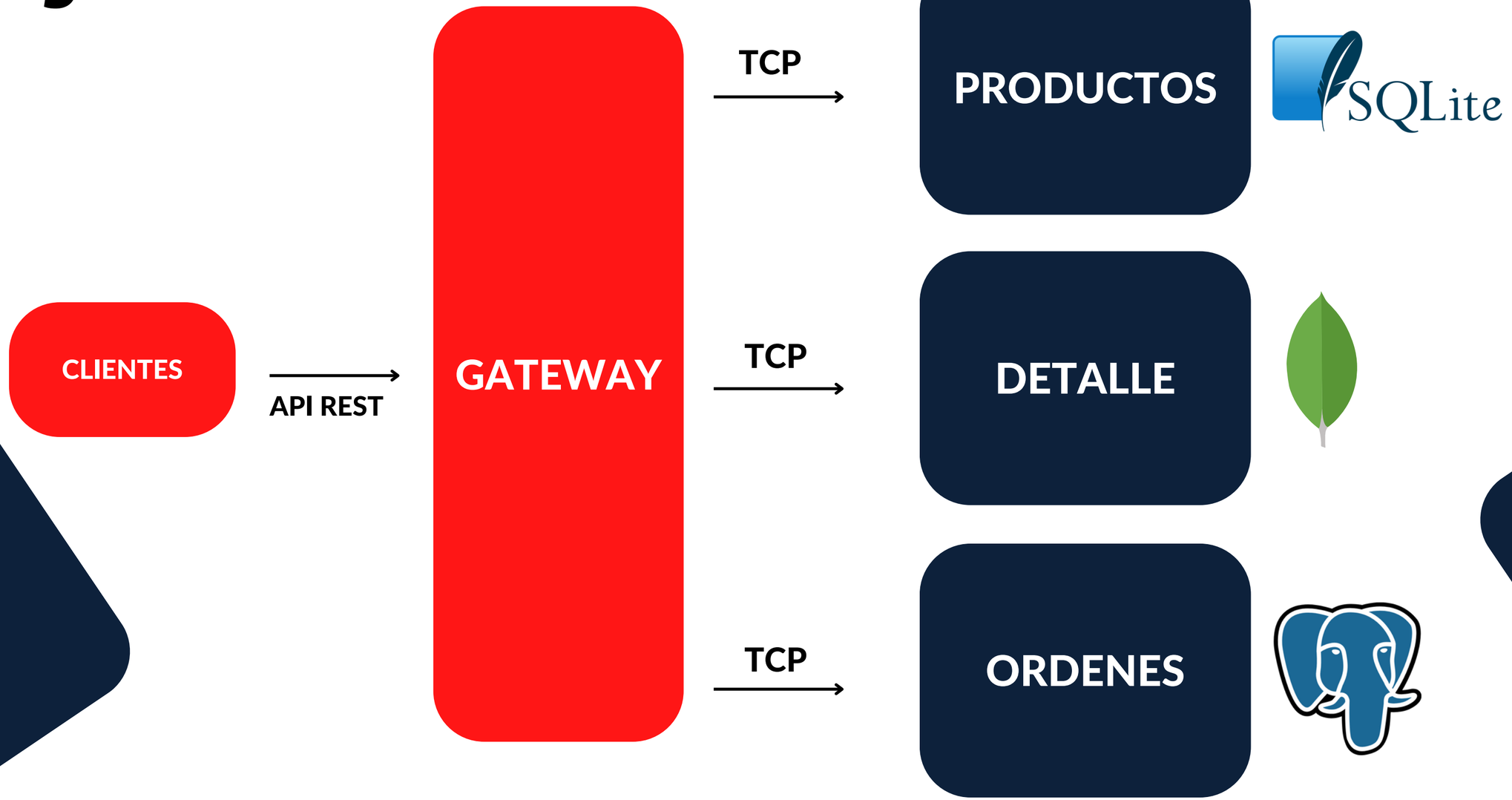
- **Evitar el acoplamiento excesivo.**
- **Evitar la dependencia entre ellos.**



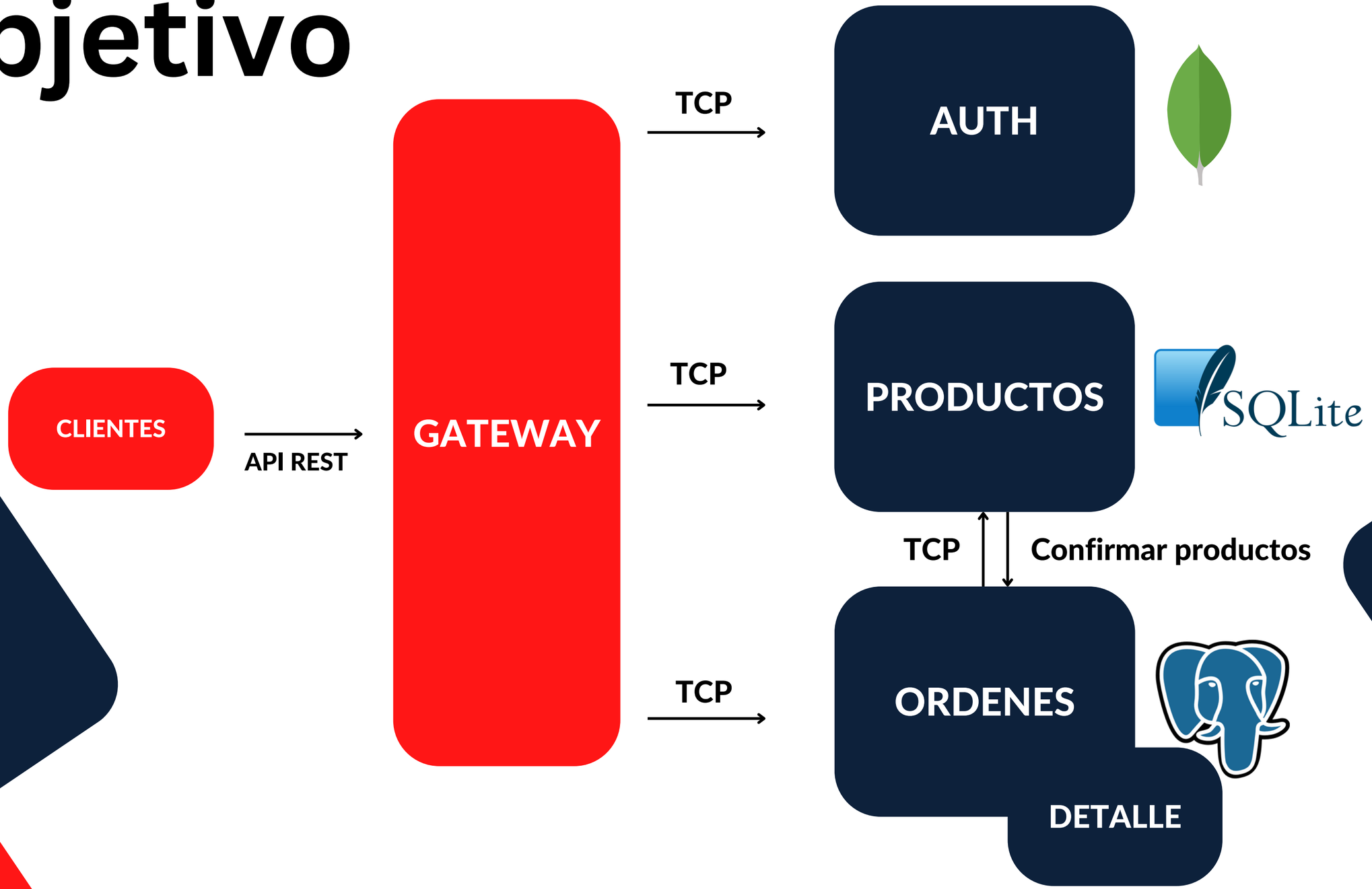
SUFICIENTE TEORÍA

EMPEZEMOS LA PRÁCTICA

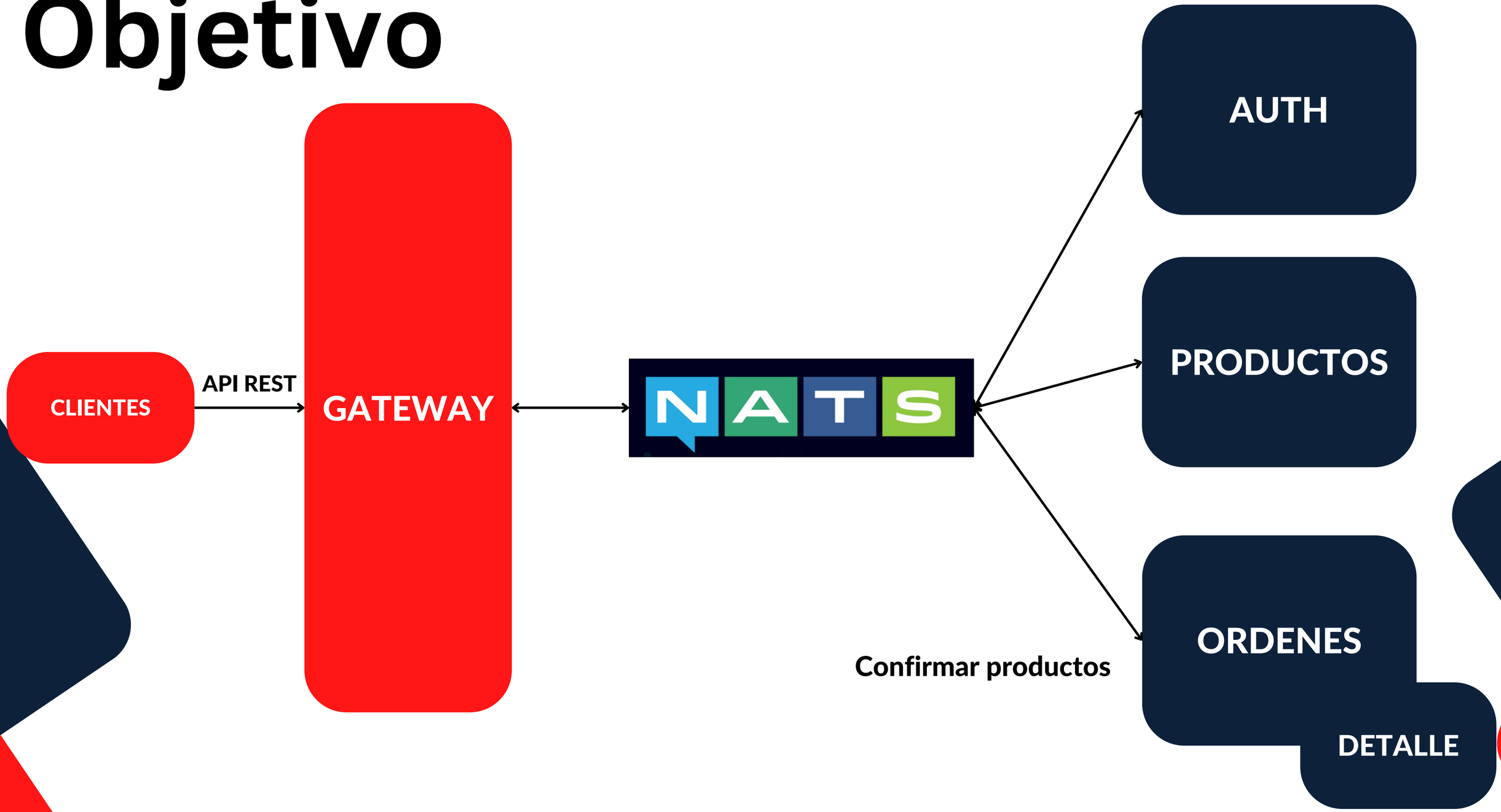
Objetivo


CLIENTES
API REST
GATEWAY
TCP
TCP
TCP
PRODUCTOS
DETALLE
ORDENES
SQLite

Objetivo


CLIENTES
API REST
GATEWAY
TCP
TCP
TCP
AUTH
PRODUCTOS
SQLite
TCP
Confirmar productos
ORDENES
DETALLE

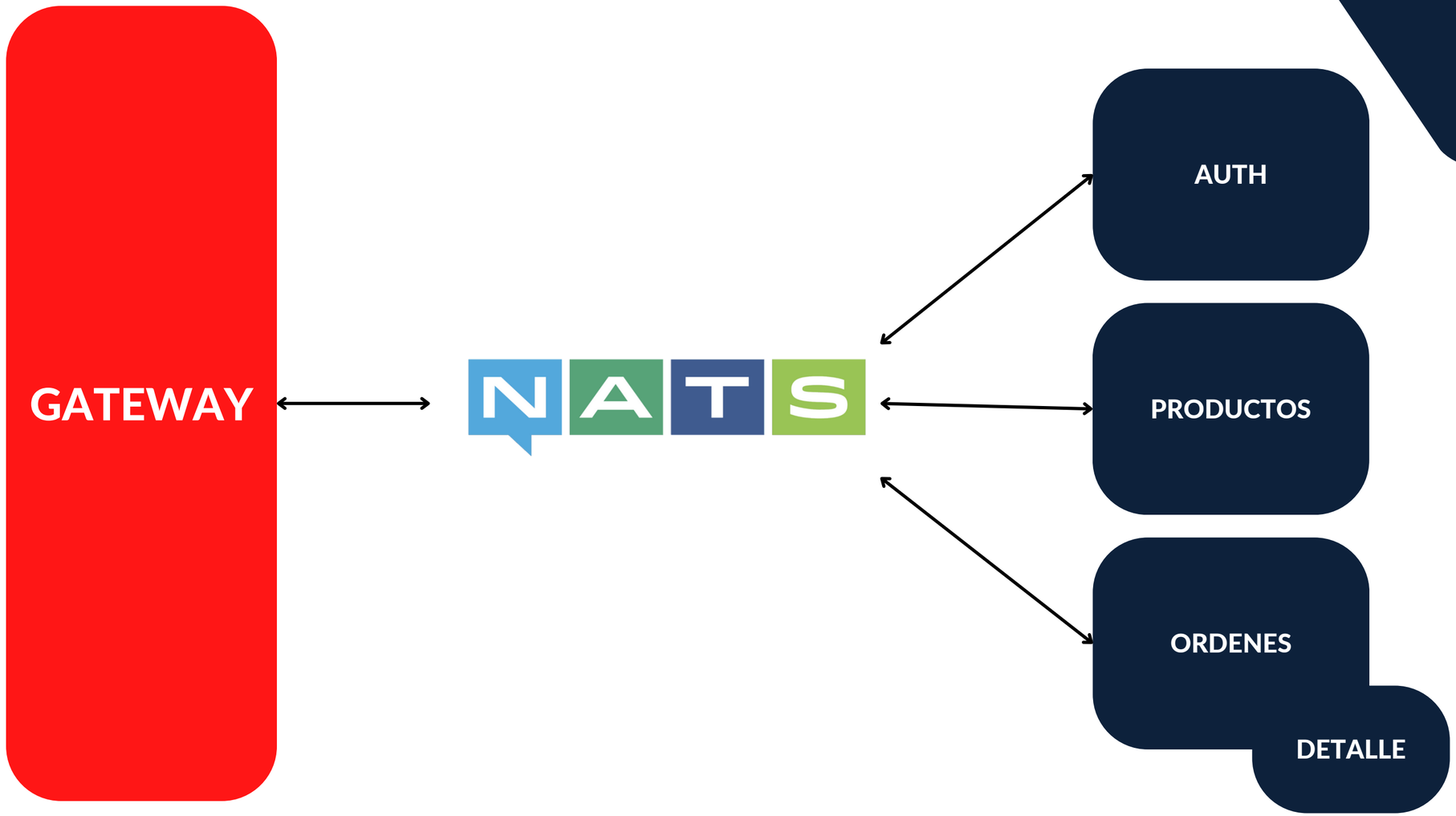
Objetivo


CLIENTES
API REST
GATEWAY
NATS
AUTH
PRODUCTOS
ORDENES
DETALLE
Confirmar productos

NATS
BROKER

Open source

Middleman

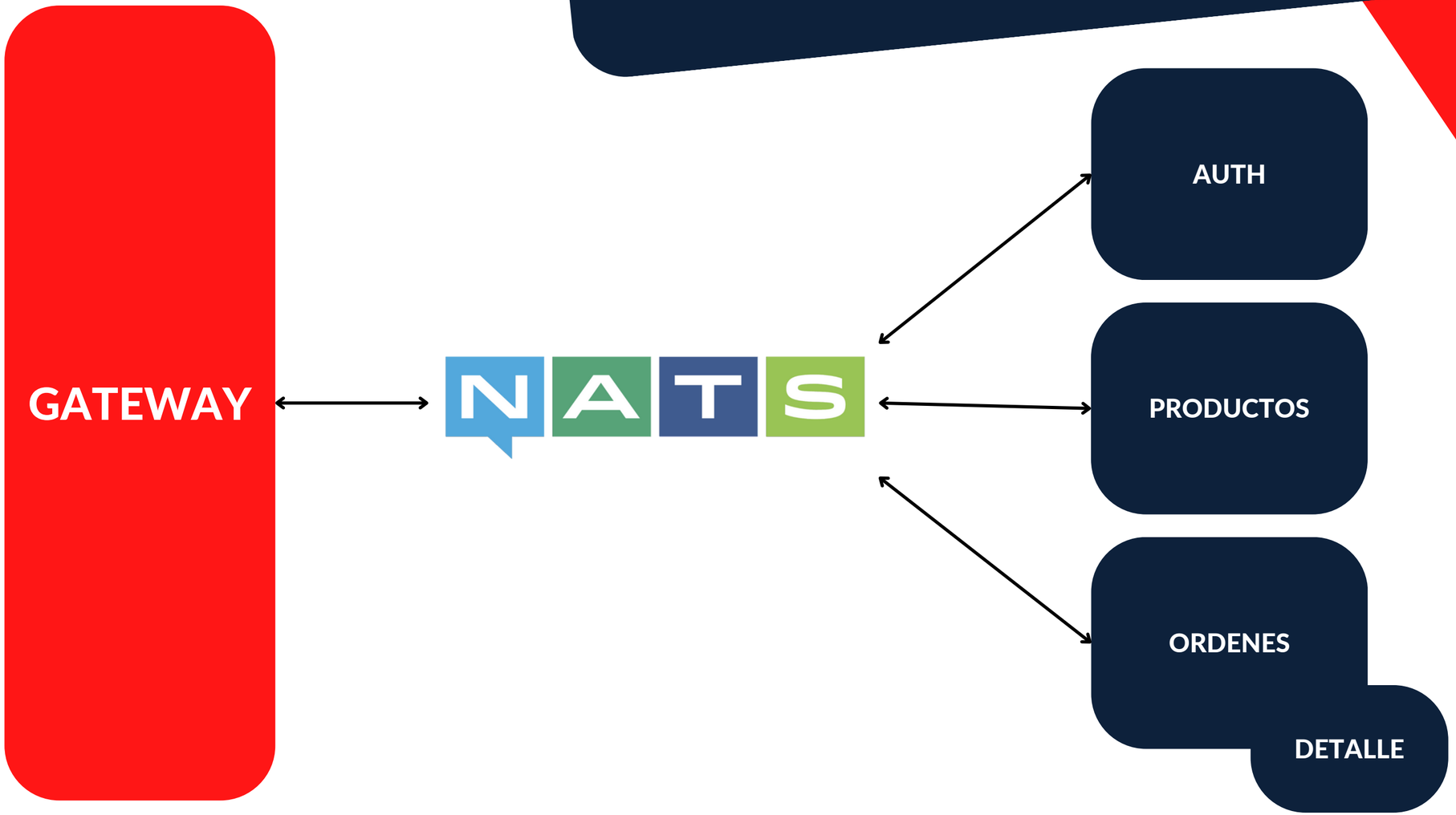


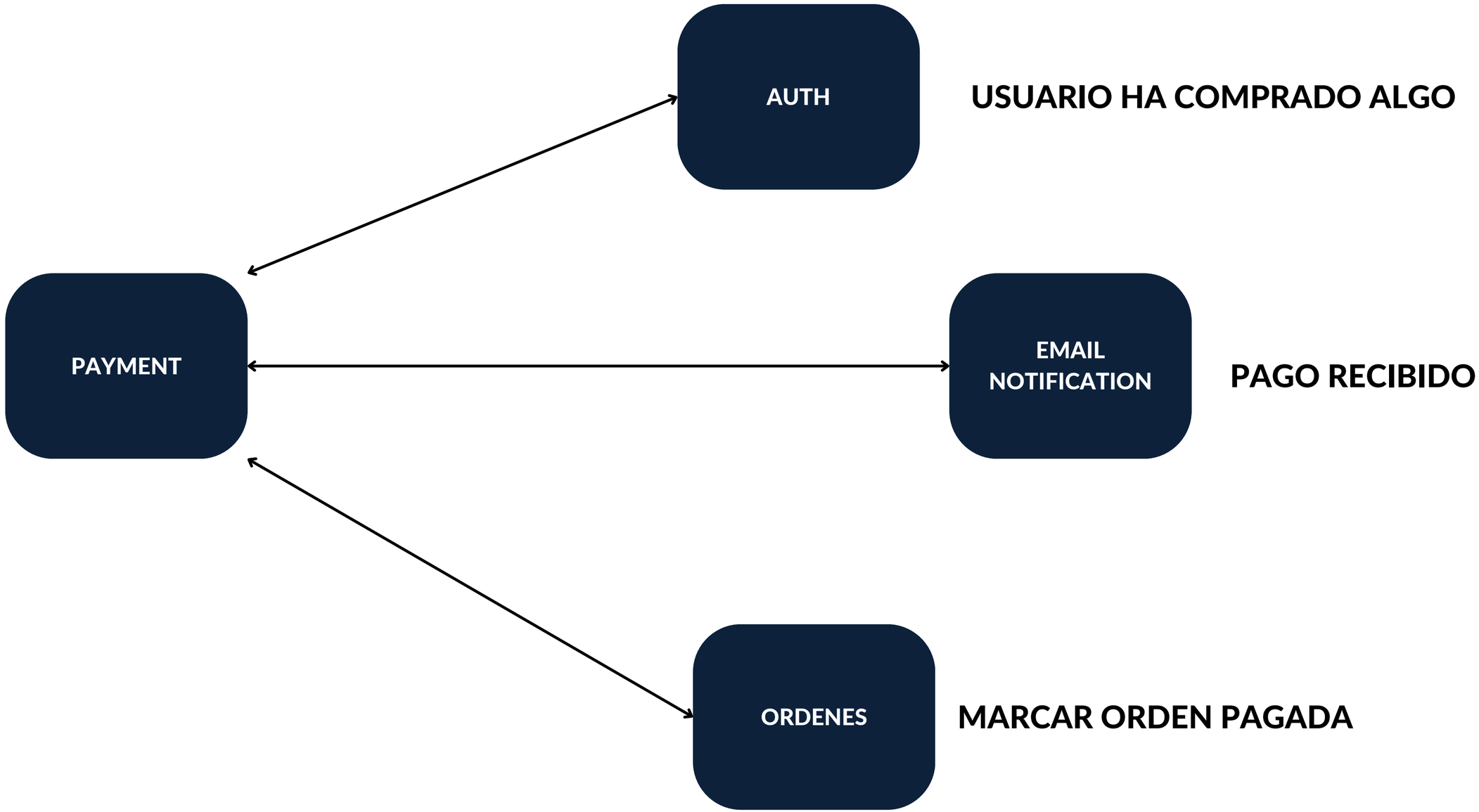
UNA TELA QUE UNE TODOS NUESTROS SISTEMAS DISTRIBUÍDOS

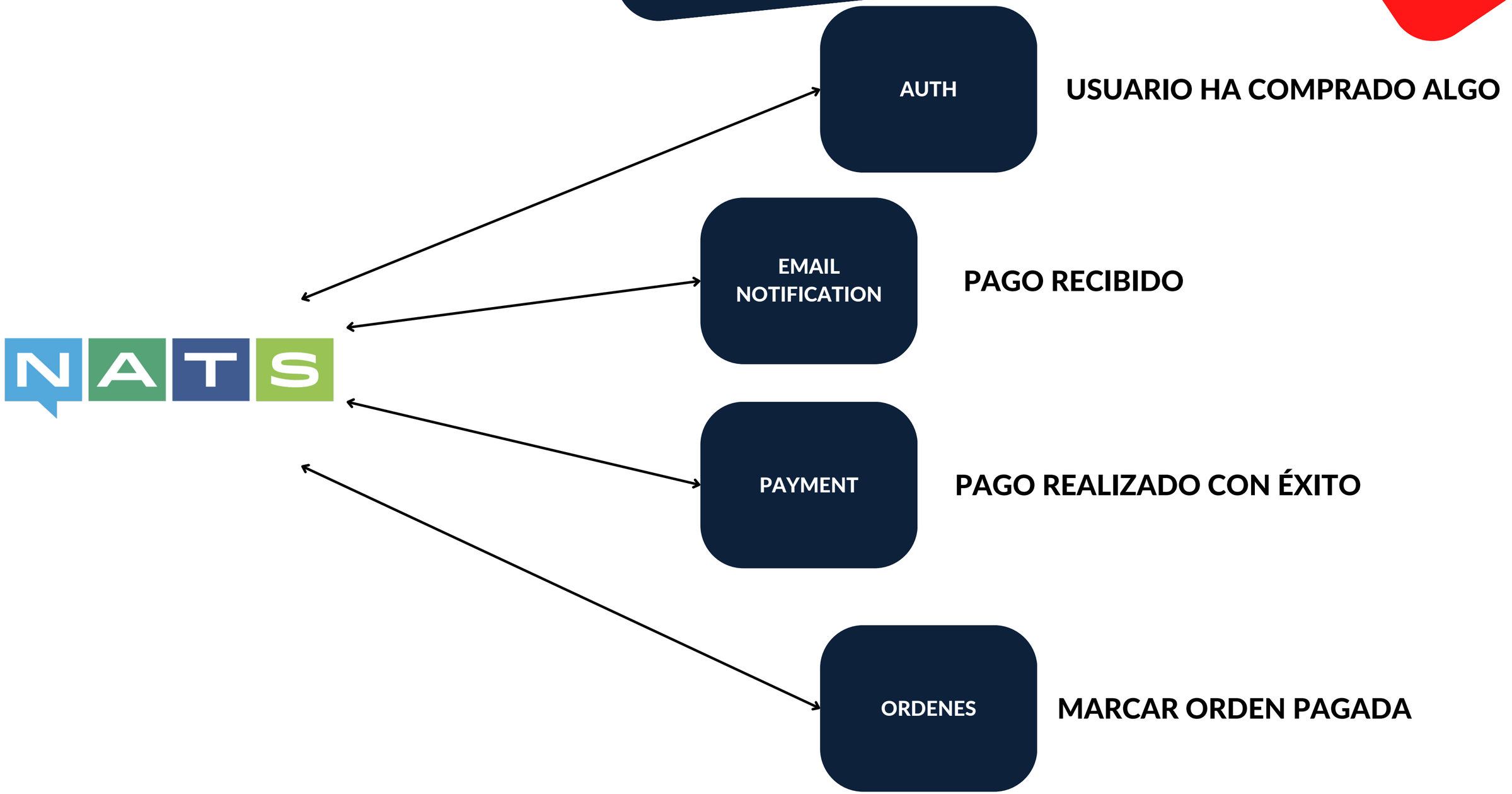
- Trabaja con mensajería tipo publicar y suscribir.
- Hay temas (topics/subjects) a los cuales se escucha.
- Puede tener múltiples escuchas (listeners) al mismo topic.
- Pensado en para escalamiento horizontal.
- Seguridad, balanceo de carga incluído.
- Payload agnóstico.
- Rápido y eficiente.

Conectar microservicios

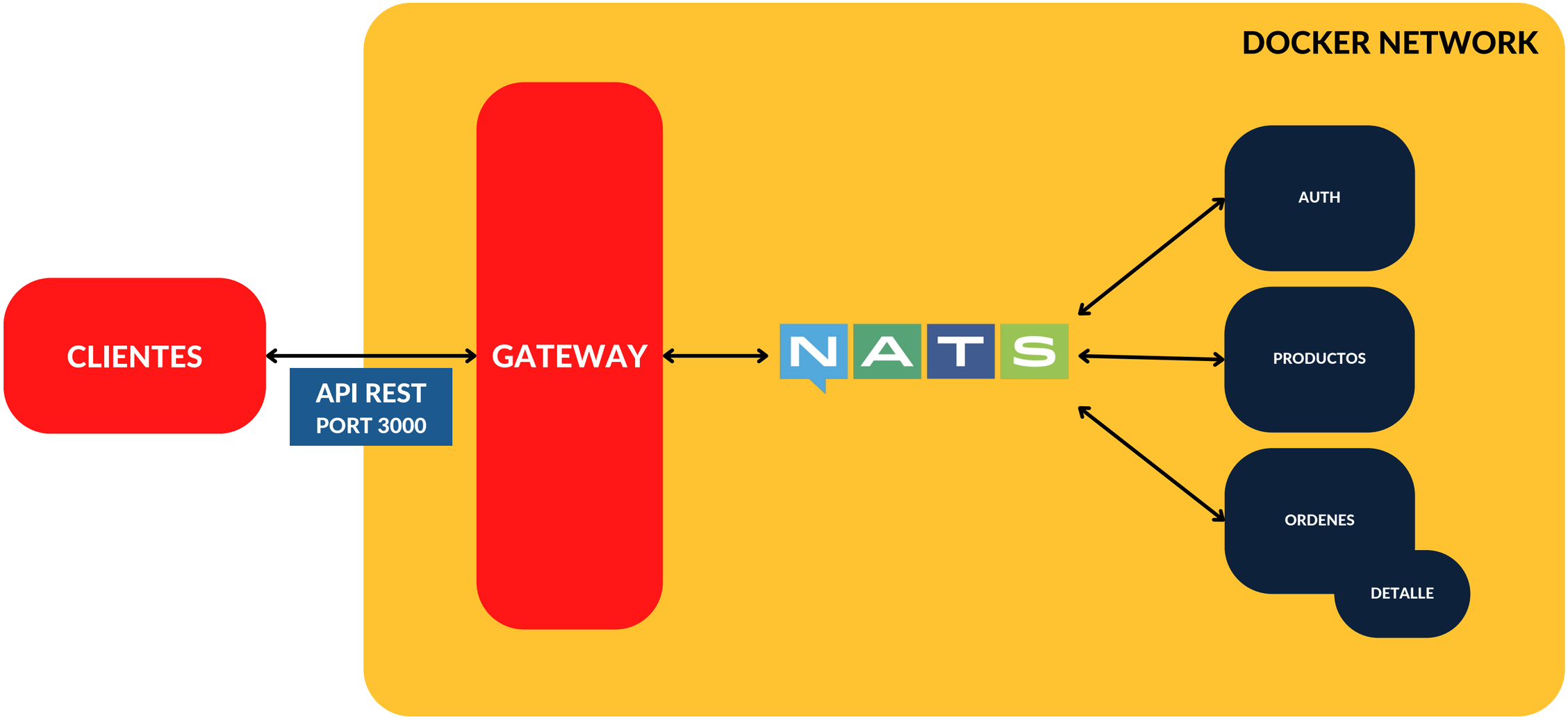
DE FORMA EFICIENTE Y RÁPIDA




AUTH
USUARIO HA COMPRADO ALGO
PAYMENT
EMAIL
NOTIFICATION
PAGO RECIBIDO
ORDENES
MARCAR ORDEN PAGADA


AUTH
USUARIO HA COMPRADO ALGO
EMAIL
NOTIFICATION
PAGO RECIBIDO
NATS
PAYMENT
PAGO REALIZADO CON ÉXITO
ORDENES
MARCAR ORDEN PAGADA

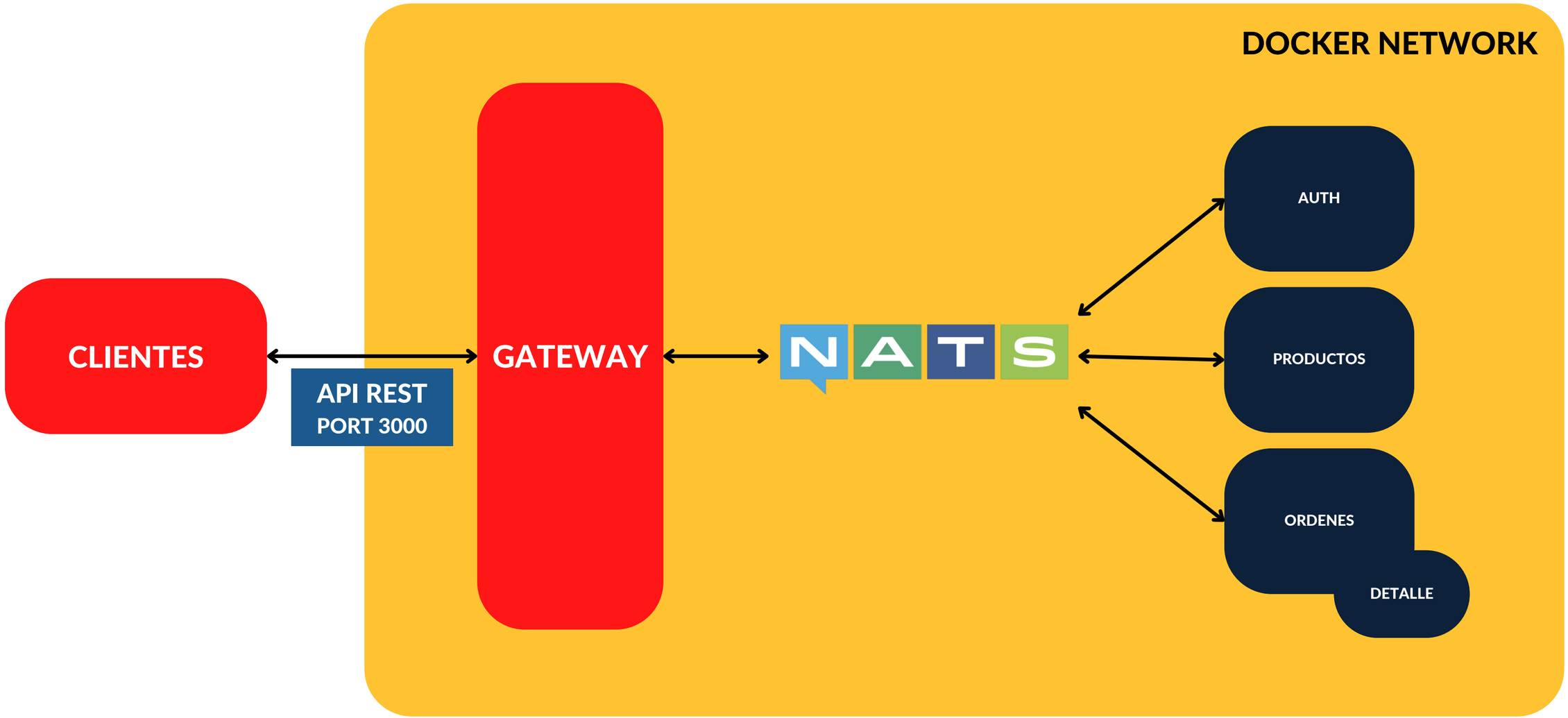
Docker Network



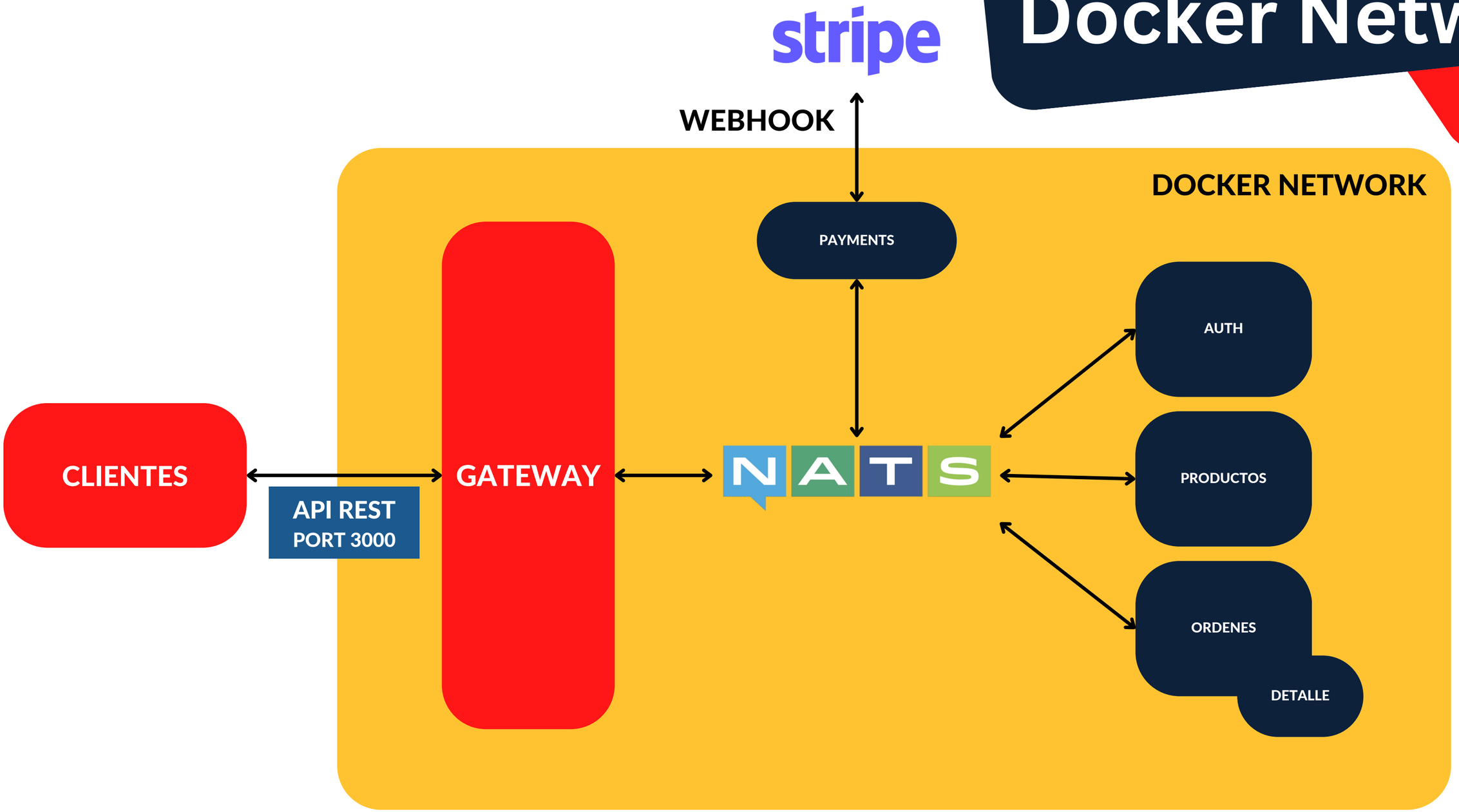
Docker Network

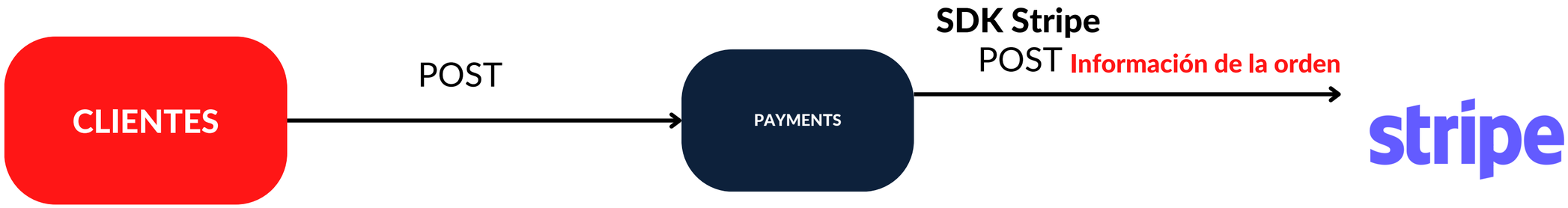
PAYMENTS

- POST: Crear sesión de pago
- Webhook: Escuchar pago desde Stripe



Docker Network




CLIENTES
POST
PAYMENTS
SDK Stripe
POST Información de la orden
stripe

stripe

CLIENTES

POST

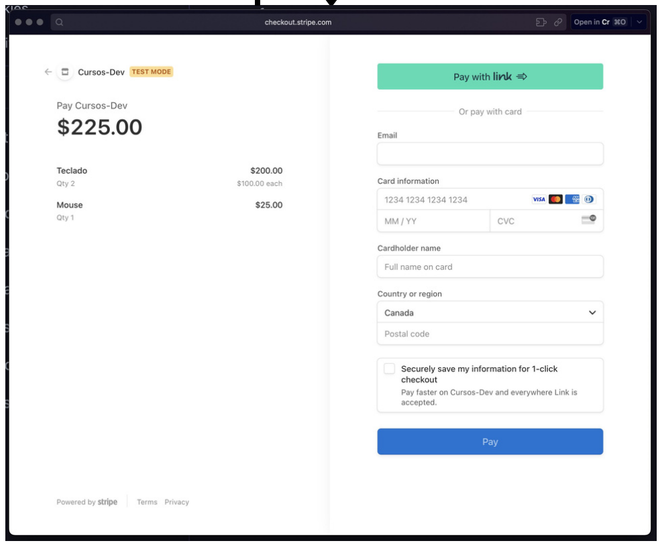
PAYMENTS

SDK Stripe

POST **Información de la orden**

stripe

URL + Tokens



stripe

CLIENTES

POST

PAYMENTS

SDK Stripe

POST **Información de la orden**

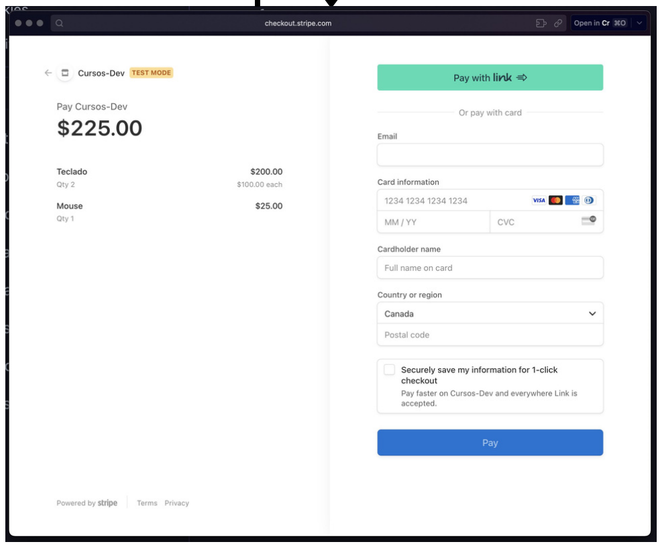
stripe

Webhook

Pago realizado

orderId=XYZ

URL + Tokens



stripe

CLIENTES

POST

PAYMENTS

SDK Stripe

POST **Información de la orden**

stripe

Webhook

Pago realizado

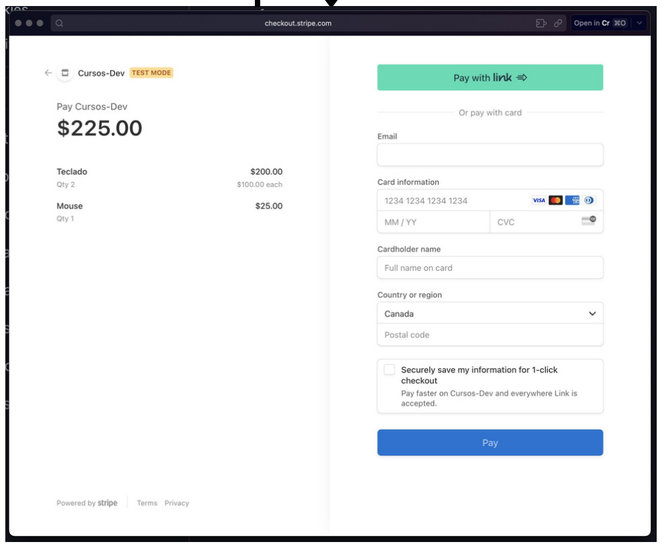
orderId=XYZ

URL + Tokens

Marcar como pagada

orden XYZ

En el microservicio respectivo



KUBERNETES
INTRODUCCIÓN

Permite a los desarrolladores desplegar, actualizar y administrar aplicaciones de manera eficiente, proporcionando herramientas para la gestión de recursos, el balanceo de carga, la auto curación y la escalabilidad horizontal

KUBERNETES

KUBERNETES


docker
docker
docker
docker
docker
docker
docker
docker
docker

KUBERNETES


docker
docker
docker
docker
docker
docker
docker
docker
docker

KUBERNETES

REPLICAS

CLUSTER

MASTER NODE

API SERVER
Administración

CONTROLLER MANAGER
Sabe lo que pasa

SCHEDULER
Asegura el cambio de PODs

ETCD
KeyValue storage

Controla toda la infraestructura

- Es el más importante
- Puede tener réplicas

Accedemos a él

- CLI
- UI
- API

CLUSTER

MASTER NODE

API

C-M

SCHED

ETCD

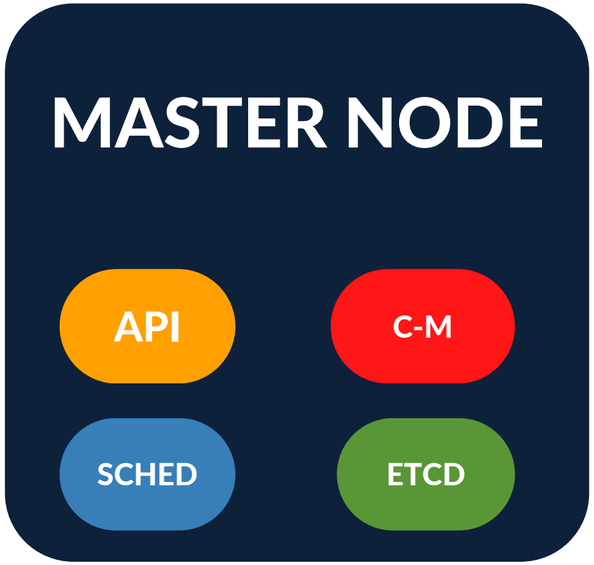
Controla toda la infraestructura

- **Es el más importante**
- **Puede tener réplicas**

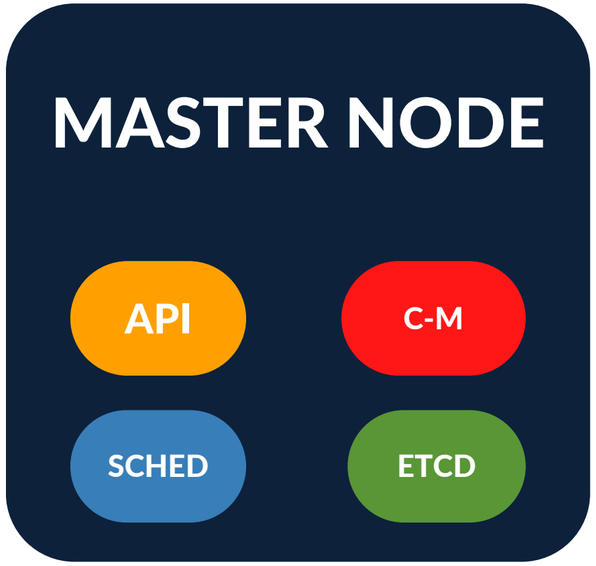
Accedemos a él

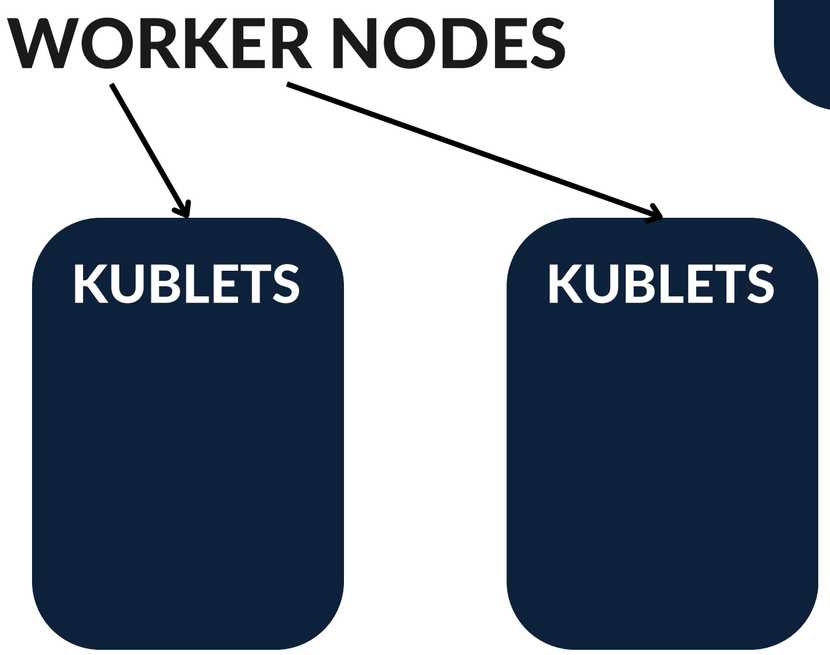
- **CLI**
- **UI**
- **API**

CLUSTER



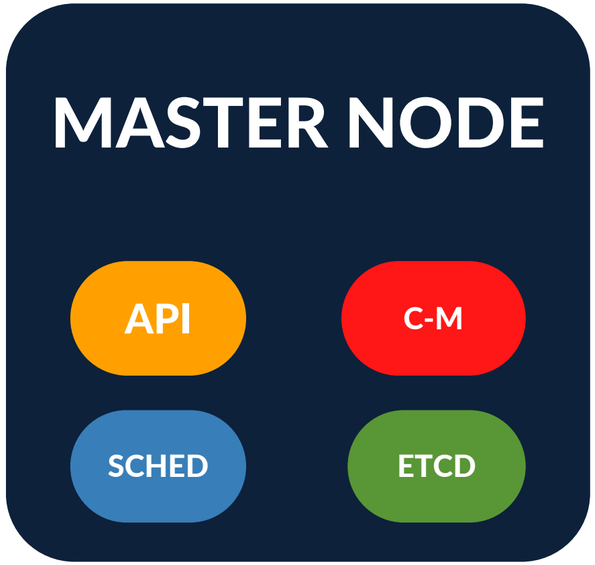
CLUSTER

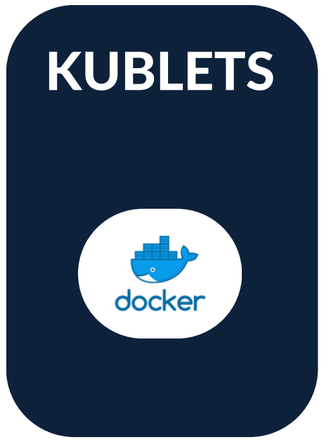


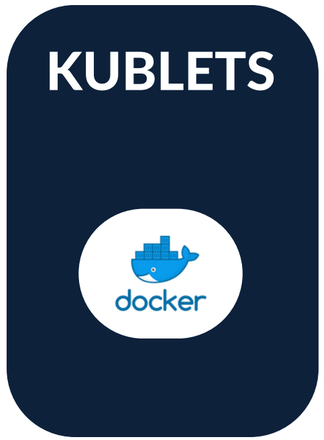


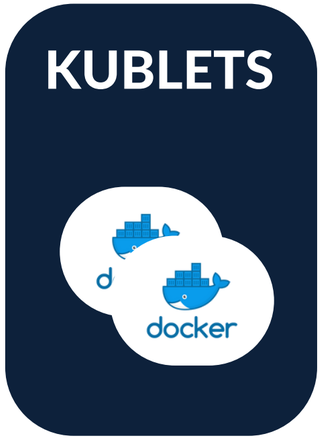
Kublet: Proceso corriendo que permite la comunicación entre el cluster

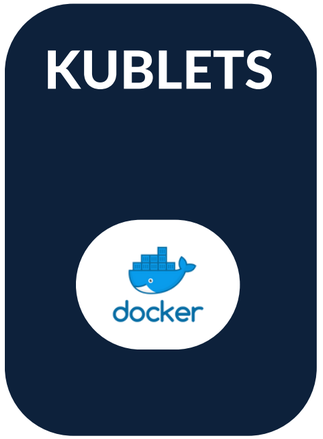
CLUSTER

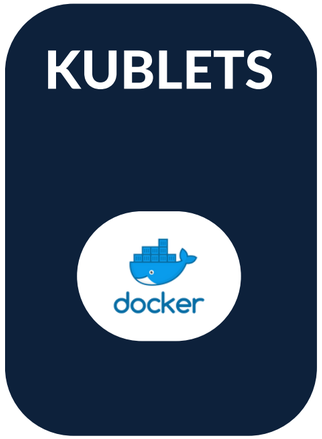

MASTER NODE
API
C-M
SCHED
ETCD


KUBLETS
docker

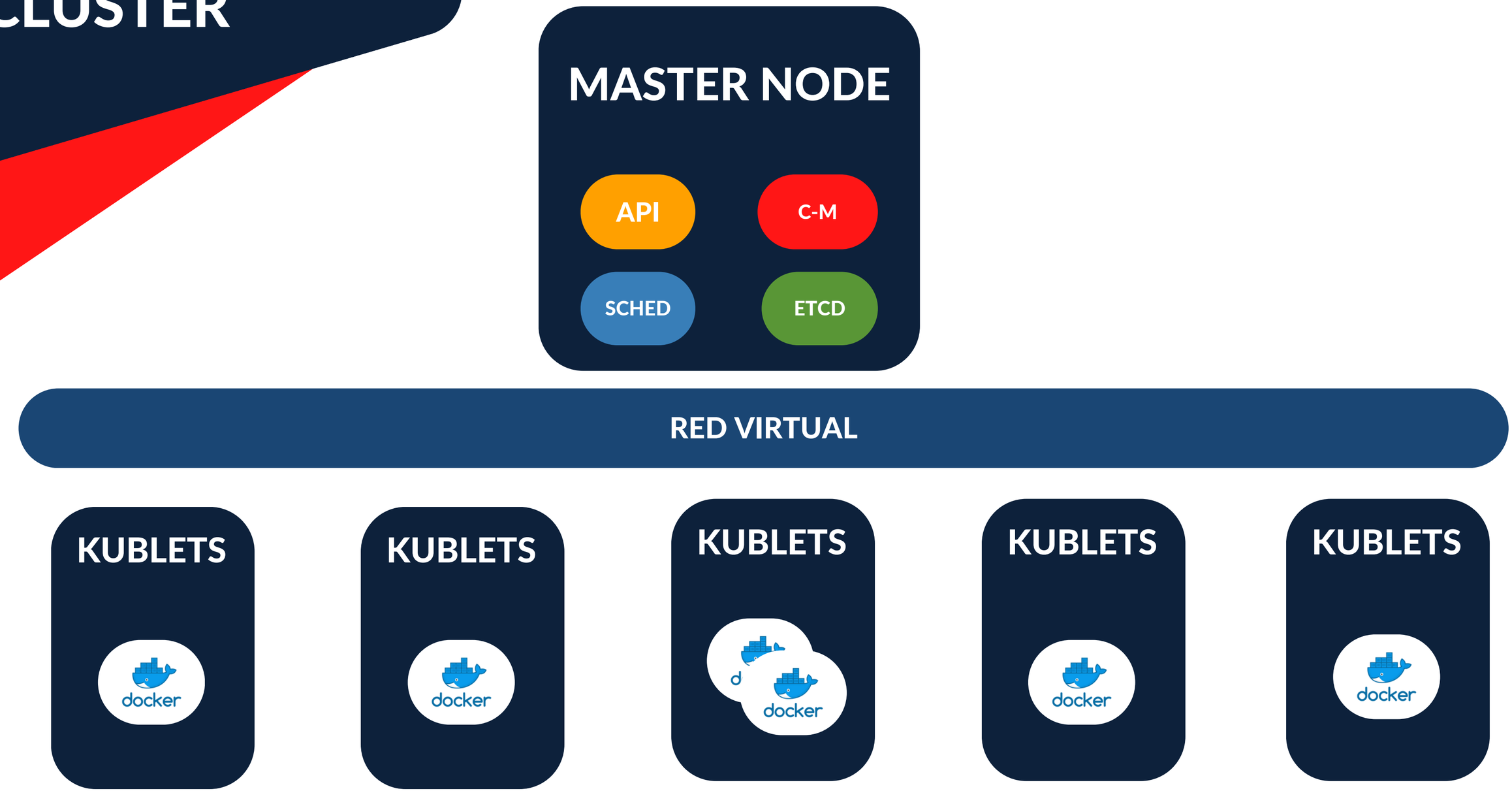

KUBLETS
docker


KUBLETS
docker


KUBLETS
docker

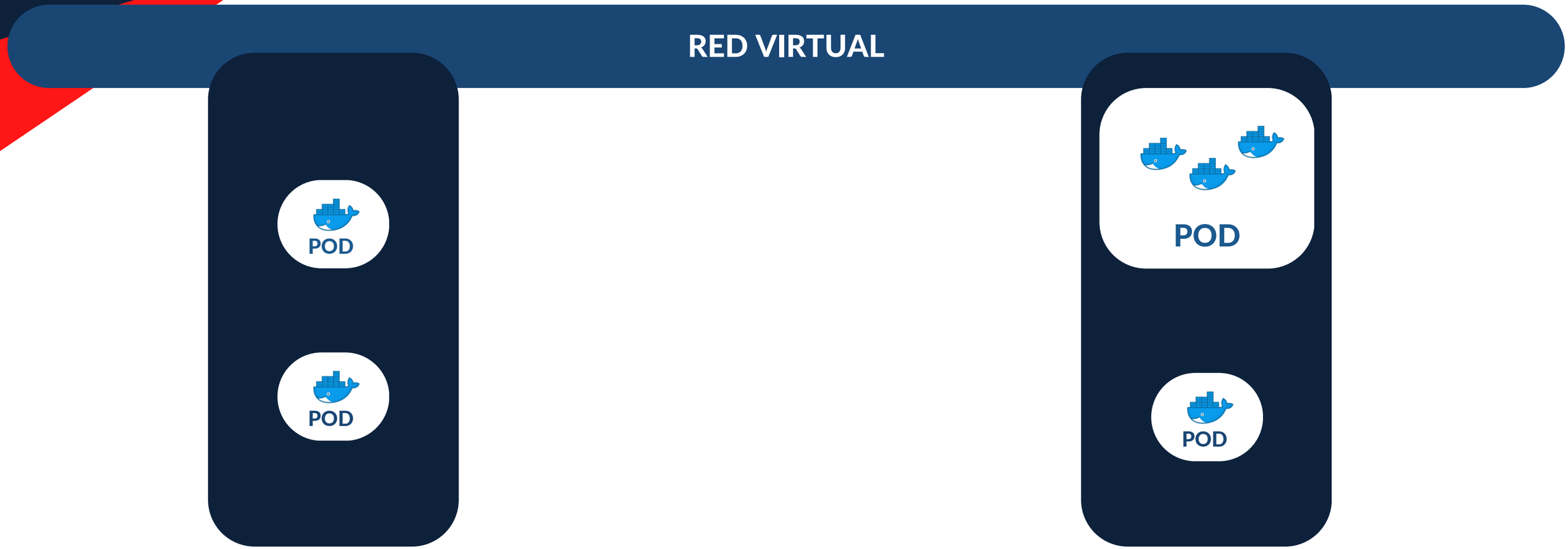

KUBLETS
docker

CLUSTER



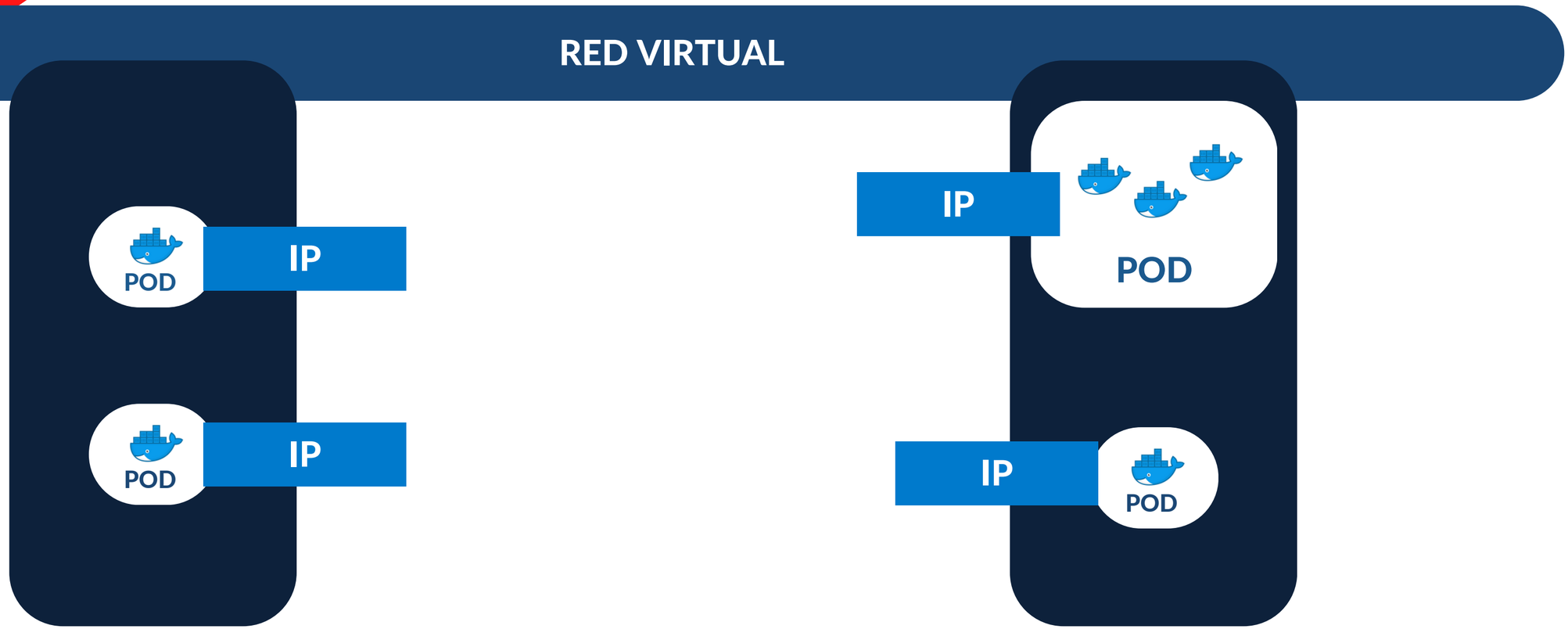
Virtual Network: Convierte nuestro cluster en una única máquina

PODs



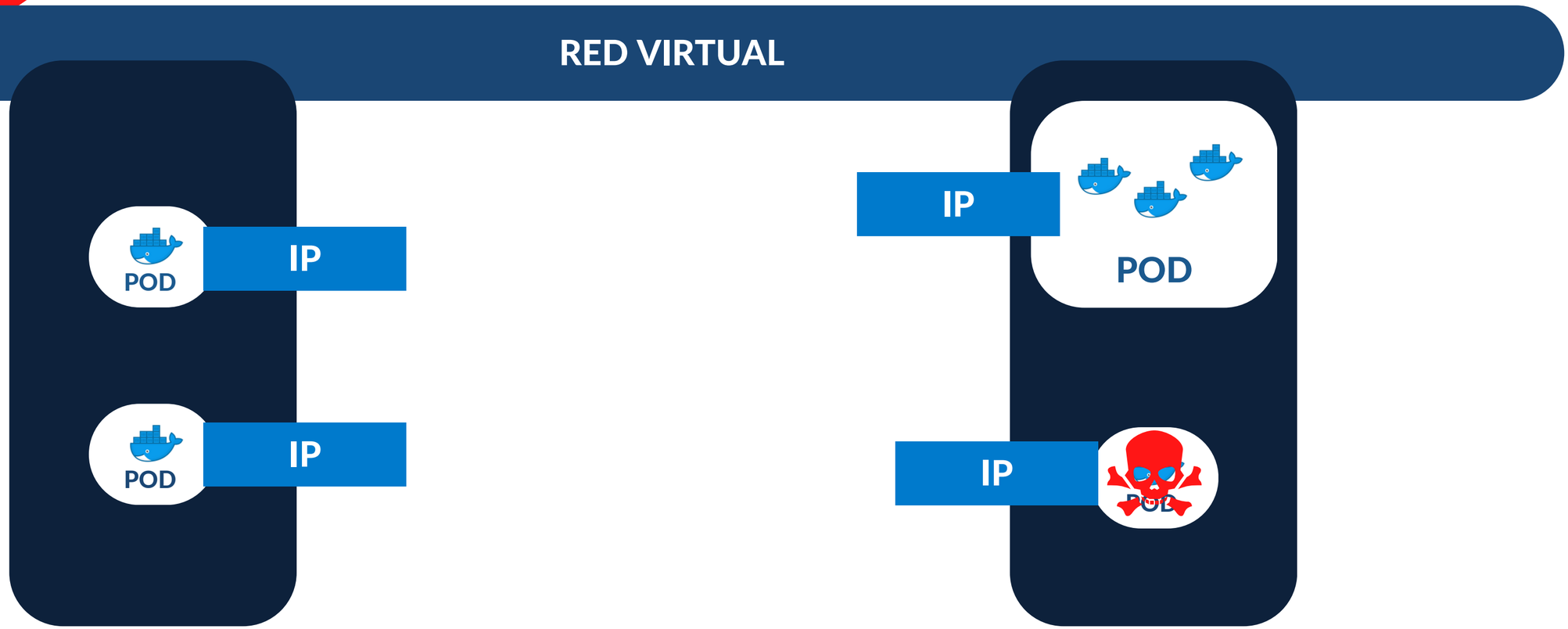
POD: Unidad más pequeña, es efímera, con su propia IP estática e ID único

PODs



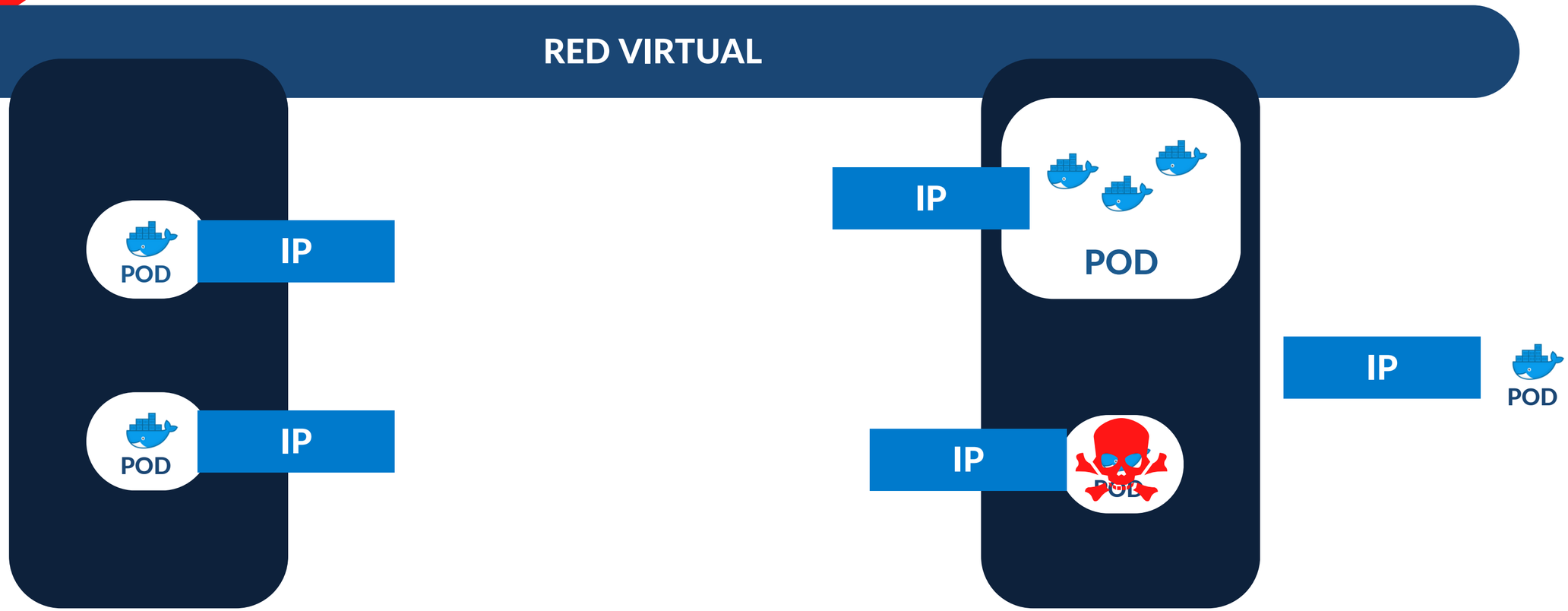
POD: Unidad más pequeña, es efímera, con su propia IP estática e ID único

PODs



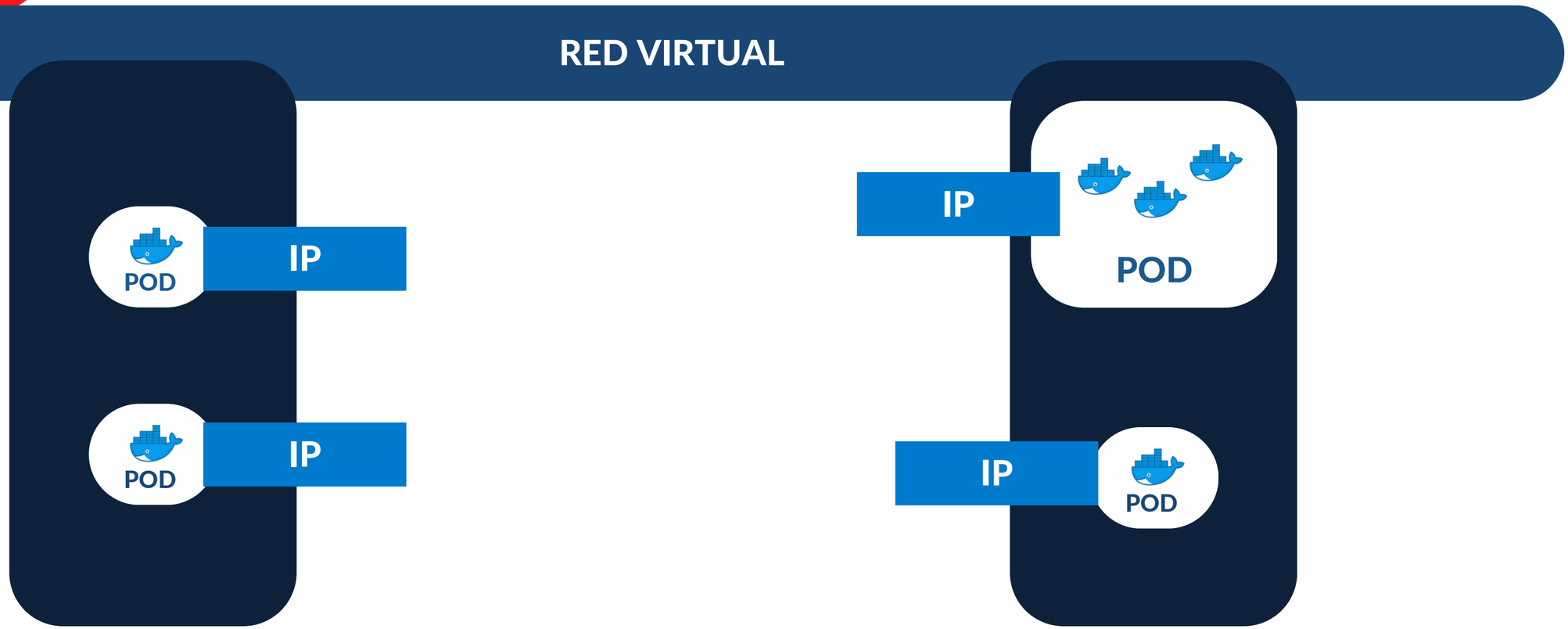
POD: Unidad más pequeña, es efímera, con su propia IP estática e ID único

PODs



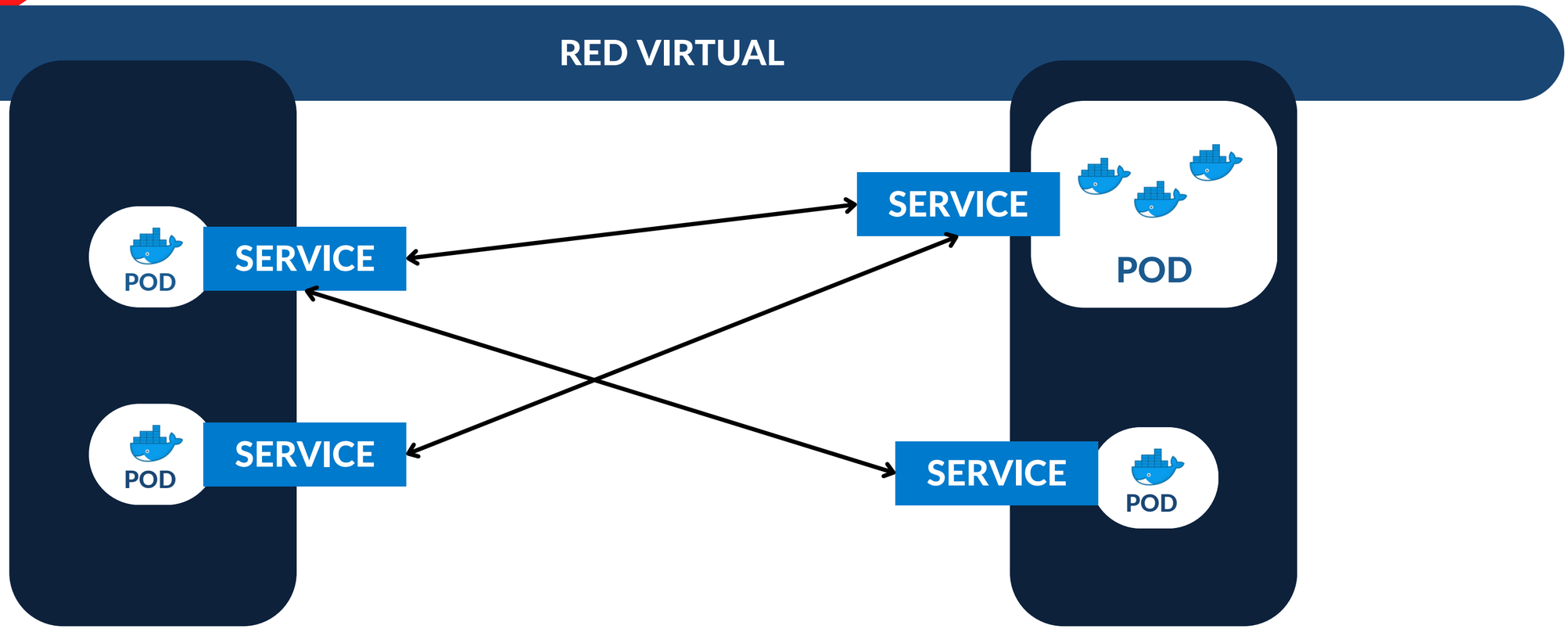
POD: Unidad más pequeña, es efímera, con su propia IP estática e ID único

PODs



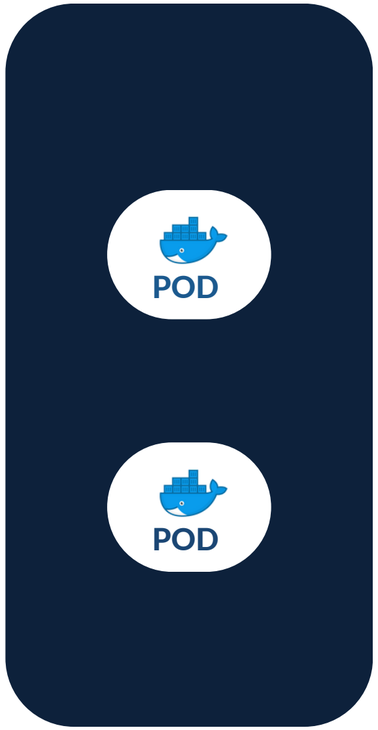
POD: Unidad más pequeña, es efímera, con su propia IP estática e ID único

Services



Servicios: En lugar de la IP, usamos el servicio, se coloca antes del POD, y tiene su IP estática que no cambia

Secrets



Database-Secrets

- **Key=value**
- **mongo_uri=mongoXXXX**
- **postgre_uri=postgreXXX**
- **db_user=fernando**
- **db_pass=XXXXXX**

Auth-Secrets

- **jwt_secret=MiSuperSecretoJwtSeed**

Secrets: Son valores base64 que se configuran para no tener expuesto sus valores en archivos o repositorios.