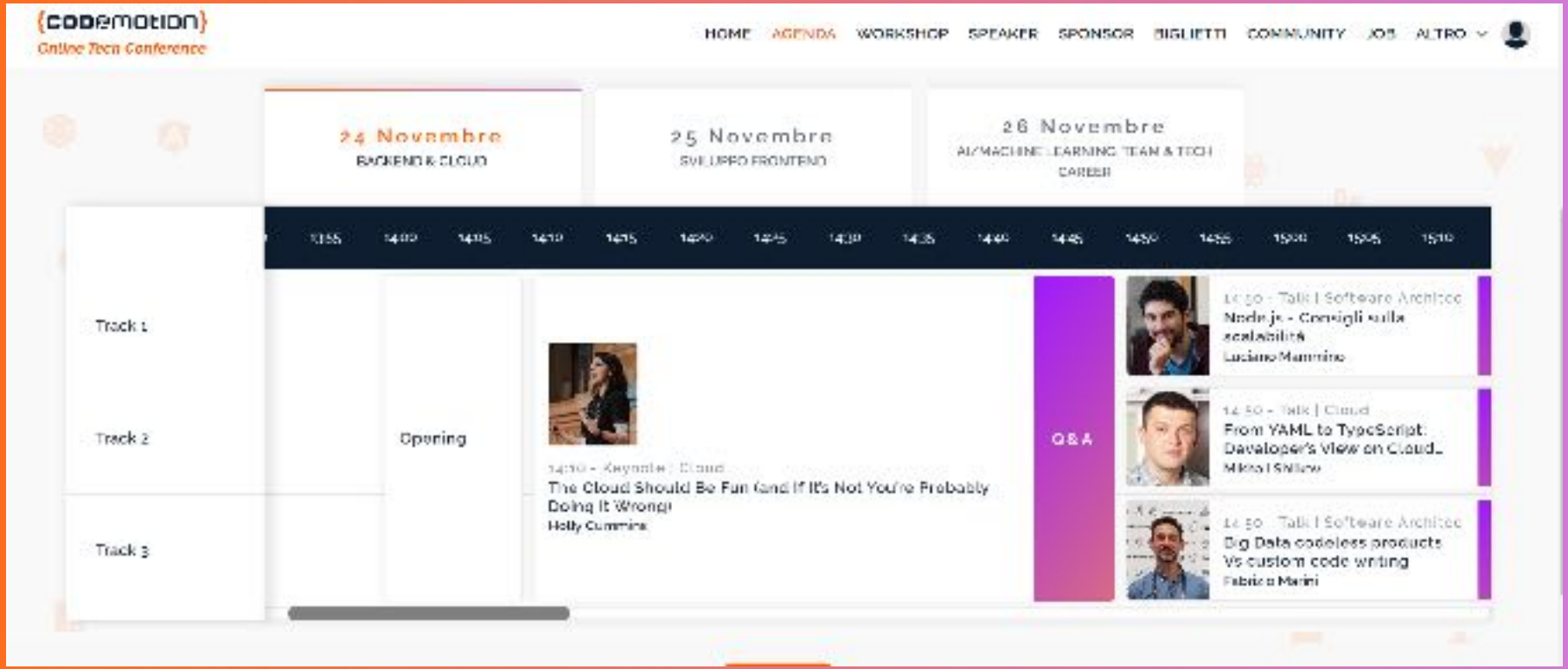

{codemotion}
Online Tech Conference
- Italian edition -
The Multitrack Tech Conference
made by Developers for Developers
November 24-25-26, 2020



Link all' agenda:

https://events.codemotion.com/conferences/online/2020/online-tech-conference-italian-edition/agenda/


WORKSHOP 1
PREMIUM TICKET
Meet our sponsors
Your 2 cents
Join Q&A and chat
Join chat with sponsors and tech communities on
DISCORD

Code of Conduct

Codemotion is committed to holding a Conference that reflects the diversity of its community and provides a harassment-free experience for everyone.

GET MORE INFO ON  DISCORD

Going SAGA!

The Distributed Saga pattern in a serverless environment

Gabriele Provinciali

Today

- Serverless
- Fn Architecture
- Oracle Functions Architecture
- Use Cases
- Fn Flow
- Saga
- Application
- Q&A

What is a function?

EVENT
DRIVEN

SHORT
DURATION

STATELESS
BEHAVIOR

fn

The Fn Project

- Independent open-source serverless compute platform built by Iron.io team that led Docker-centric serverless
- Can be deployed to any cloud and on-premise
- Containers are primitives
- Active w/ large core team, 3500+ commits, 75+ contributors
- Simple by design, enterprise built
- Native CloudEvents support
- FDKs
- Language-based Workflow with Fn Flow

7 open source platforms to get started with serverless computing

Serverless computing is transforming traditional software development. These open source platforms will help you get started.

15 Nov 2018 | Daniel Oh (Red Hat) | 28 | 1 comment

Quick Start

The Fn Project

Fn Platform

FDK's

Fn Flow

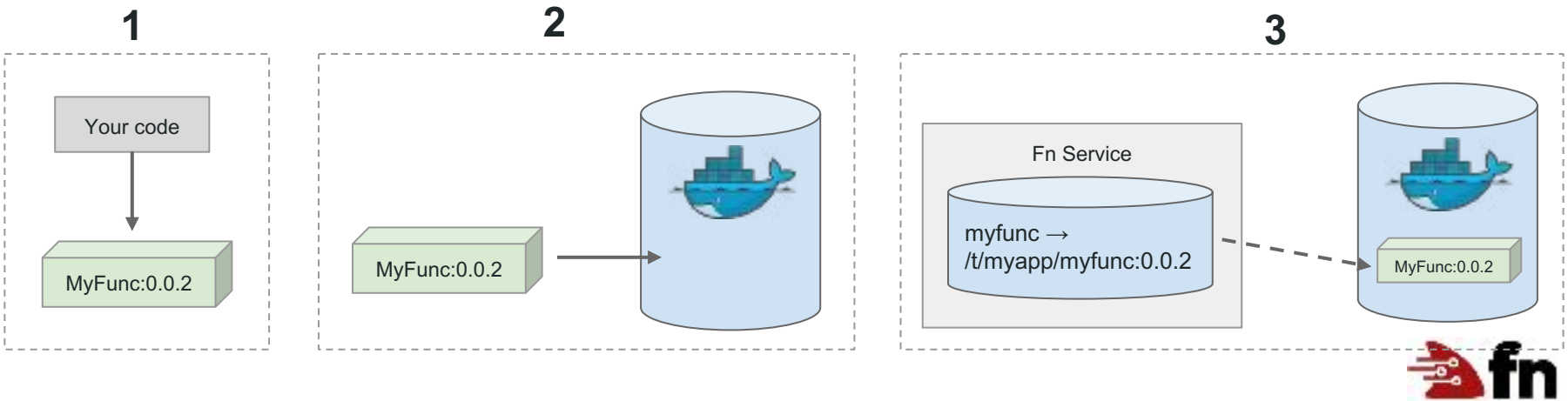
An Fn Function

- Small chunk of code wrapped into a container image
- Gets input via FDK http-stream and environment
- Produces output to http-stream
- Logs to STDERR / syslog

The Fn server handles everything else, like the API gateway, piping things around, storing logs, etc.

function deploy details

1. Builds container (multi-stage) + bumps version
2. Pushes container to registry
3. Creates/updates function & triggers



fn context

- `fn list contexts`
- `fn ls ctx`
- `fn use context <context-name>`
- `fn create context oci --api-url <ip> --registry myreg`

fn CLI

- `fn create app nodeapp`

App creation

- `fn init --runtime node nodefunc`

Function initialization and creation of boilerplate

- `fn invoke nodeapp nodefunc`

Invoke a function

- `fn --verbose deploy --app nodeapp –local`

Deploy a function

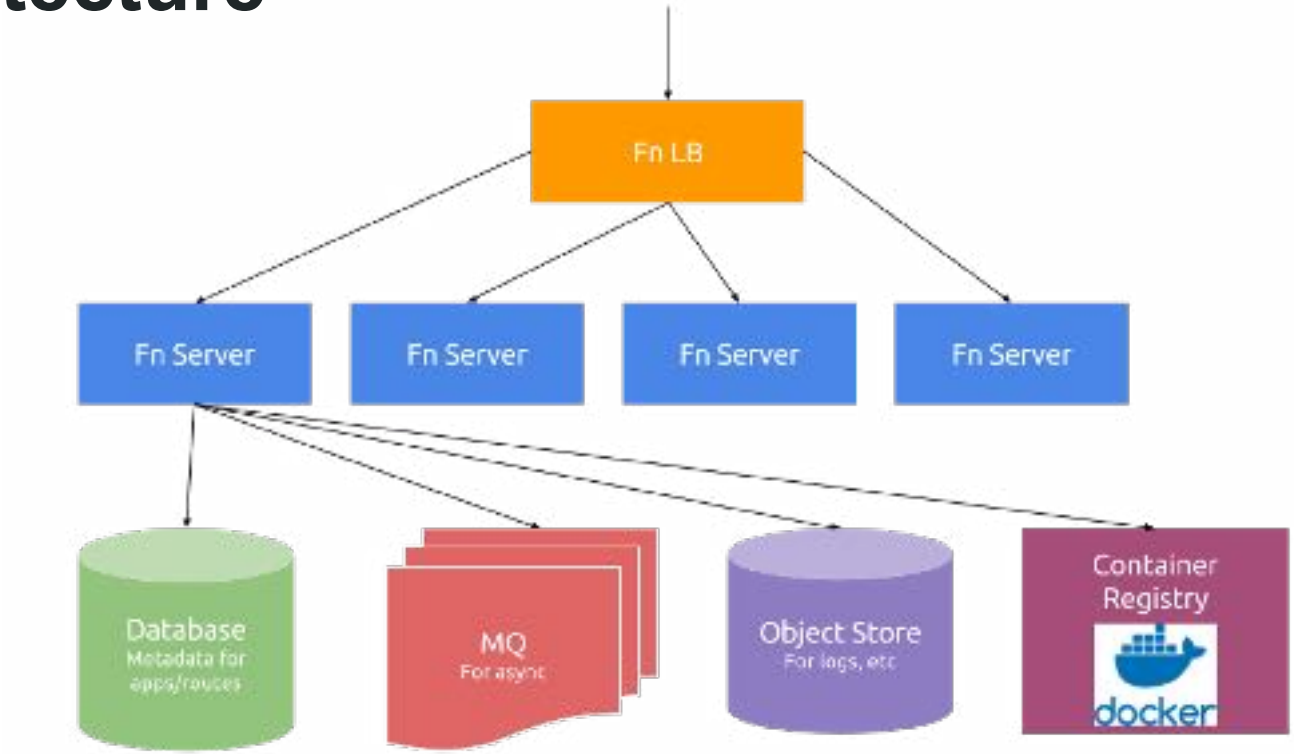
func.yaml - example

```
schema_version: 20180708
name: nodefunc
version: 0.0.2
runtime: node
entrypoint: node func.js
format: http-stream
memory: 128
cpus: 100m
idle_timeout: 600
```

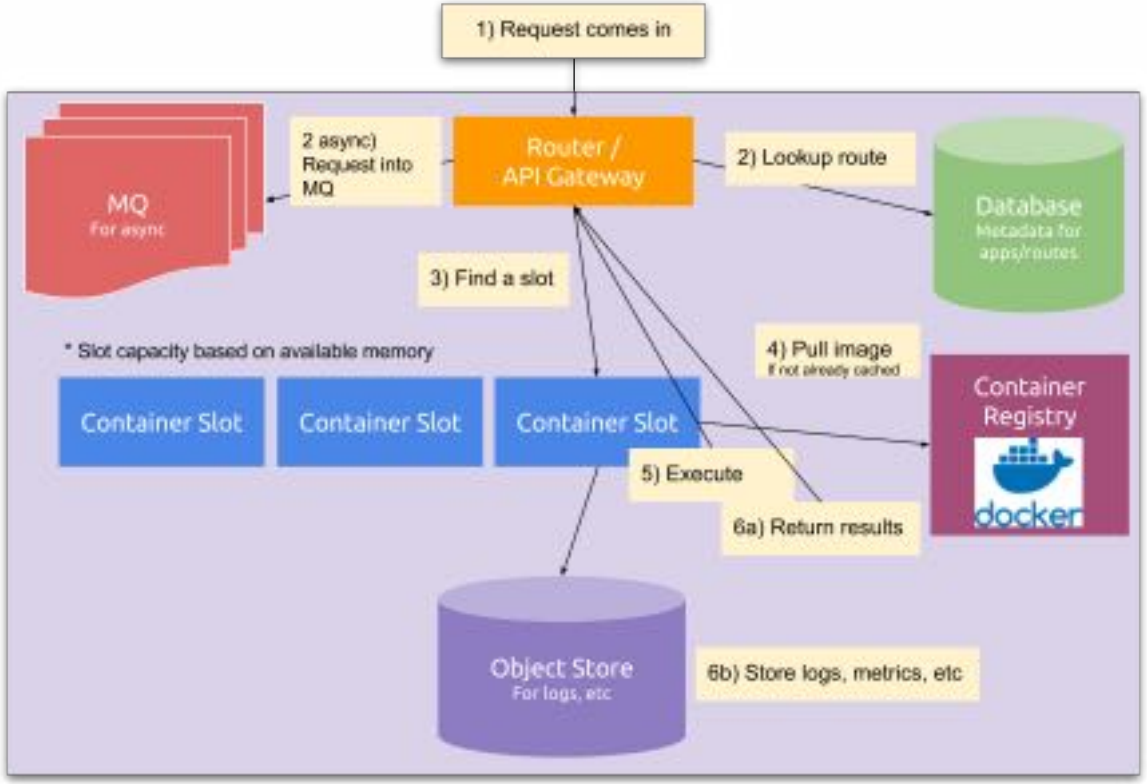
Function Development Kits (FDKs)

- Used to help with parsing input and writing output
- Familiar syntax for Lambda developers
- Simply write a `handler` function that adheres to the FDK's interface and it will parse http-stream and provide the input data to your function and deal with writing the proper output format.
- Makes it a lot easier to write hot functions


Architecture

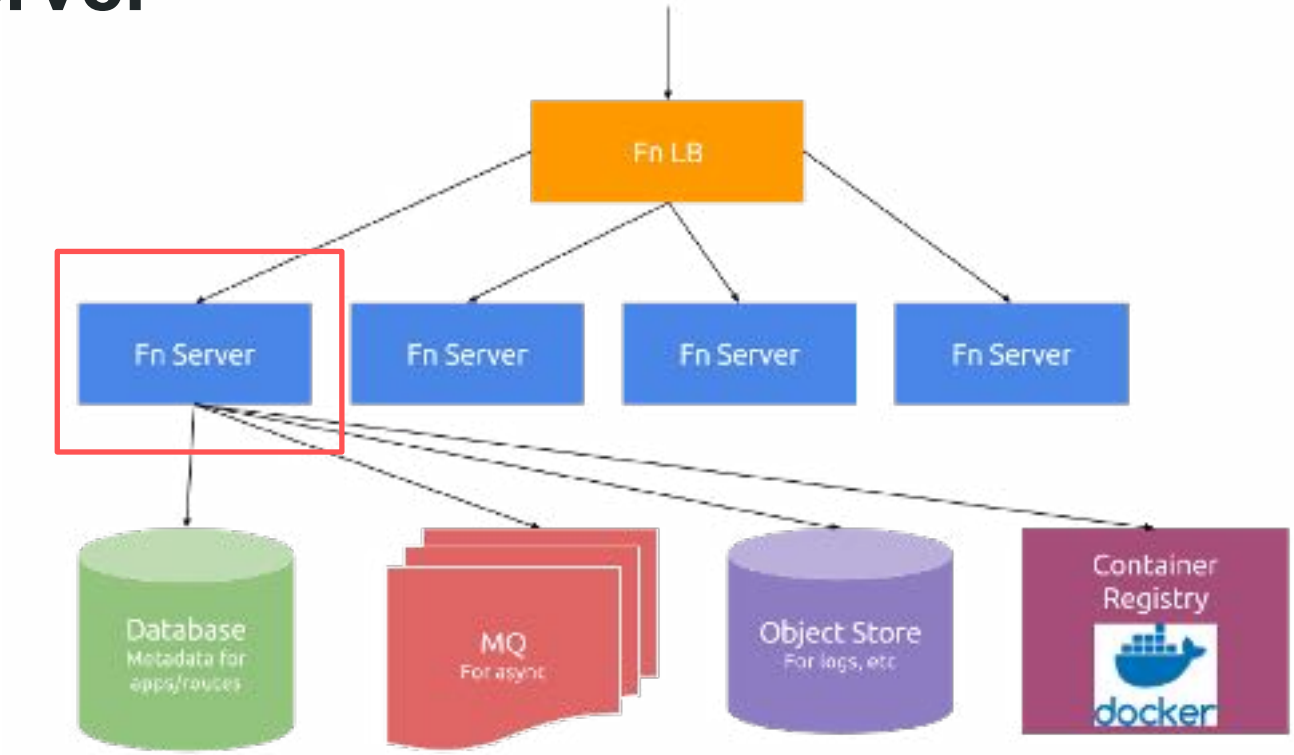


Request Flow



fn

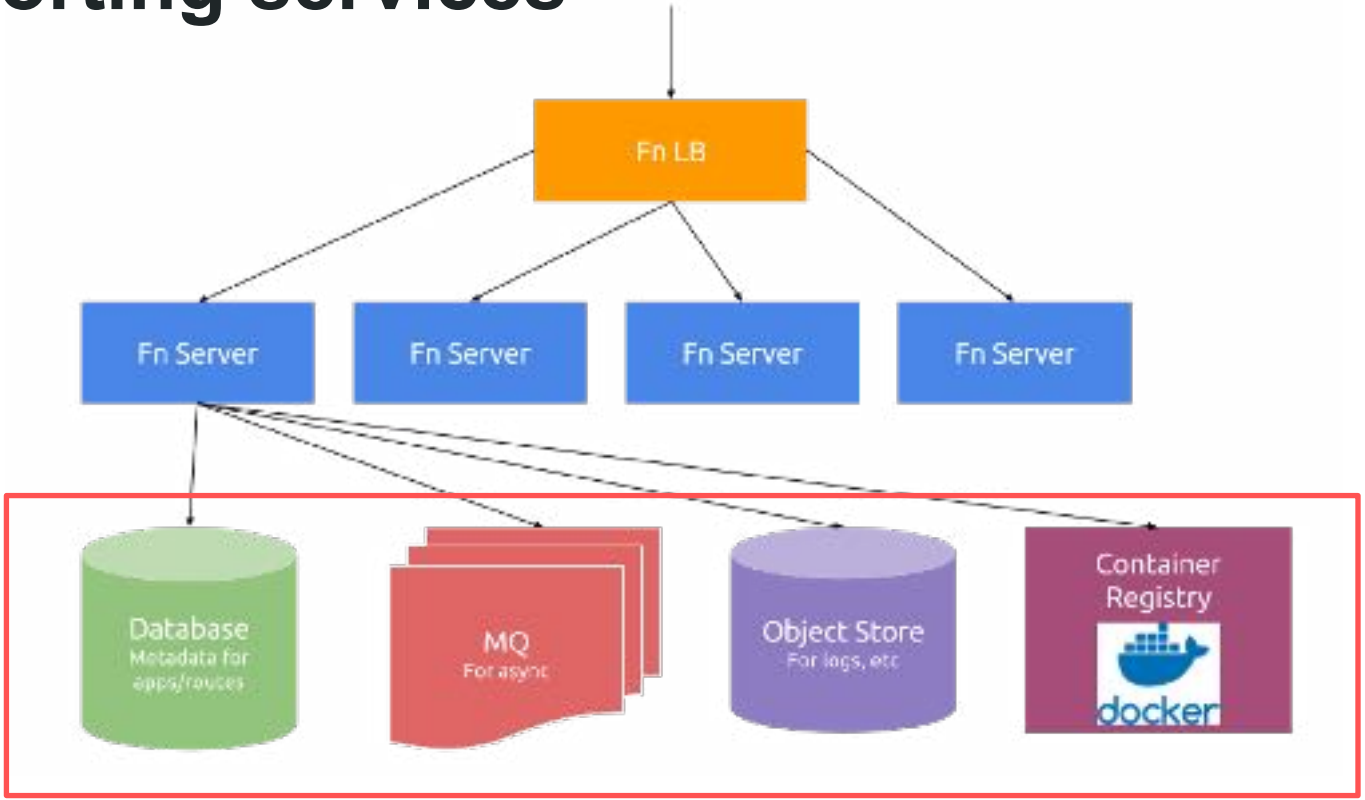
Fn server


Fn LB
Fn Server
Fn Server
Fn Server
Fn Server
Database
Metadata for
apps/routes
MQ
For async
Object Store
For logs, etc
Container
Registry
docker

Fn server

- Handles CRUD operations for setting up triggers and functions
- Executes sync functions, returning responses to clients immediately
- Queues async function calls
- Executes async functions when capacity is available
- Written in Go, easy to extend via plugin module system
- …Metrics!

Supporting services



Supporting Services

- DB, MQ, blob store are all pluggable modules that are thin wrappers around their respective drivers.
  - DB: MySQL, sqlite3, Postgres
  - Queue: Redis, Kafka
  - Registry: Any Docker v2-compliant, even private
- Metrics/Monitoring
  - OpenTracing API for metrics
  - Prometheus support, pluggable backends
  - Logging via syslog

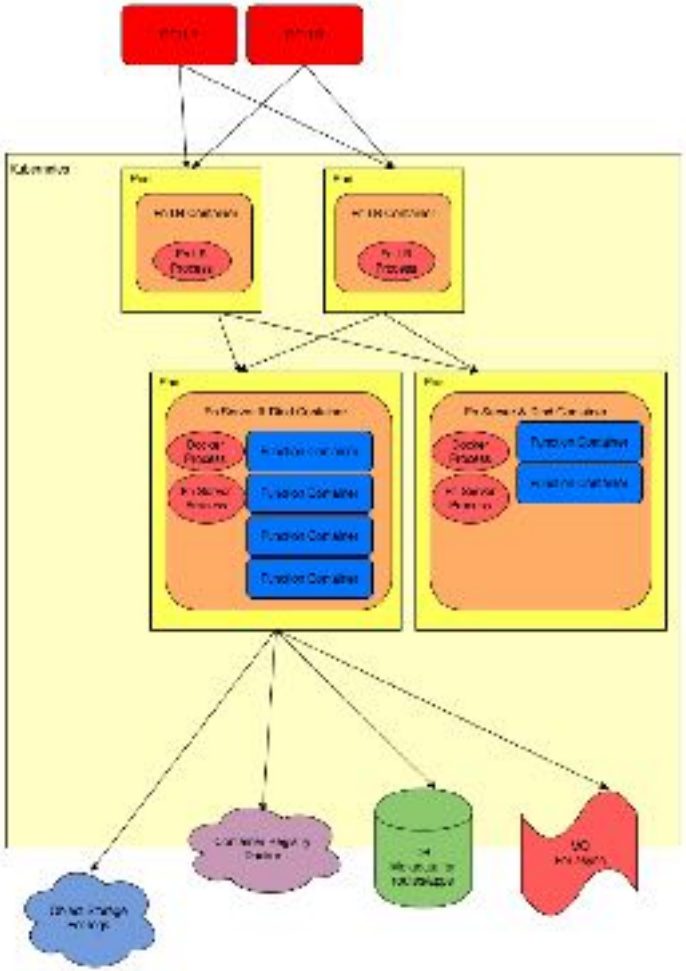
Prerequisites

- Docker 17.10.0-ce or greater
- DockerHub Account - or, your own container registry ;)
- docker login
- Mac: `brew install fn`
- Linux: `curl -LSs \ https://raw.githubusercontent.com/fnproject/cli/master/install | sh`
- Windows: `https://github.com/fnproject/cli/releases`

Kubernetes

- Fn is scheduler agnostic but lots of optimization/management work in process to optimize on Kubernetes
- Helm chart available at https://github.com/fnproject/fn-helm
- Thinking about deeper Kubernetes integrations including CRD’s to model functions

Kubernetes Deployment



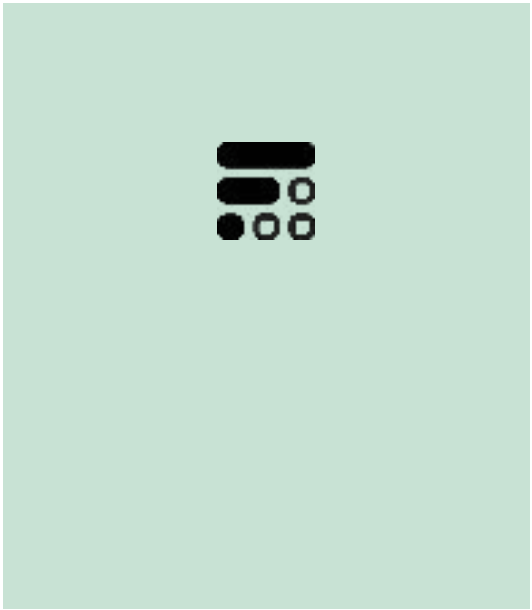
Why not K8s scheduling?

1. Speed
    a. Pod launch time is too slow for sync requests
    b. Coordinating all resource alloc to one k8s master is slow
    c. Yes we can preload + hot pod like we do with current scheduling but…

2. Scale
    a. Runs out of addressable network space quickly
    b. Functions easily scale to the hundreds of thousands / millions

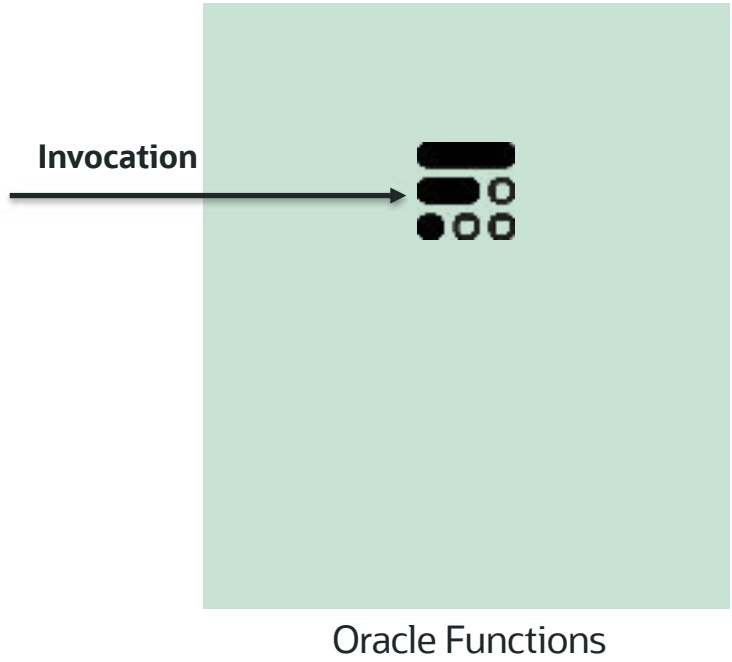
Oracle Functions

Functions – High Level Architecture

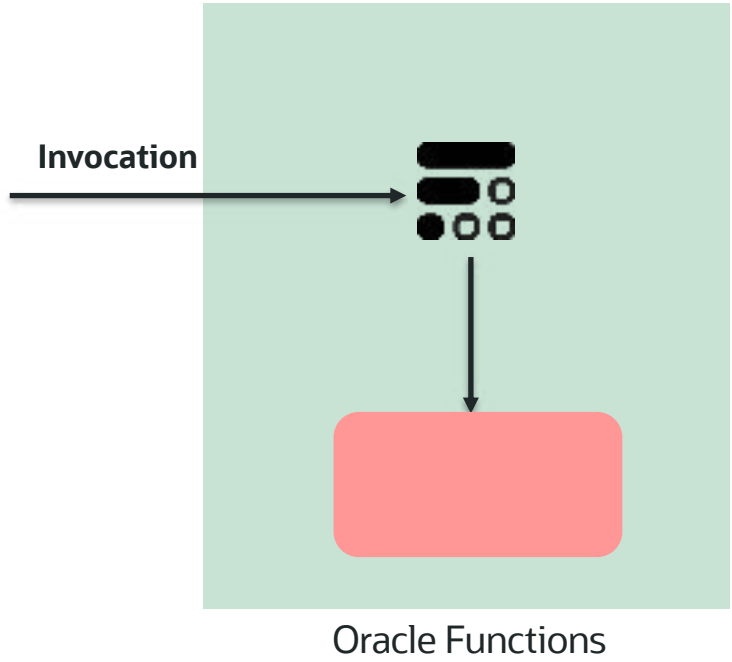


Oracle Functions

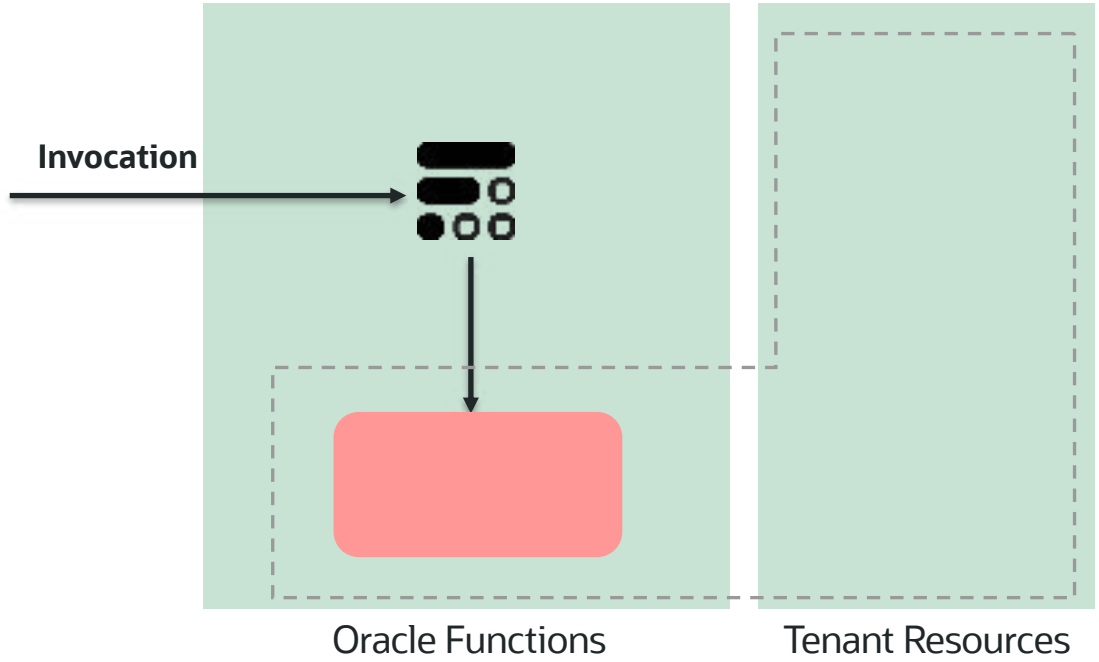
Functions – High Level Architecture



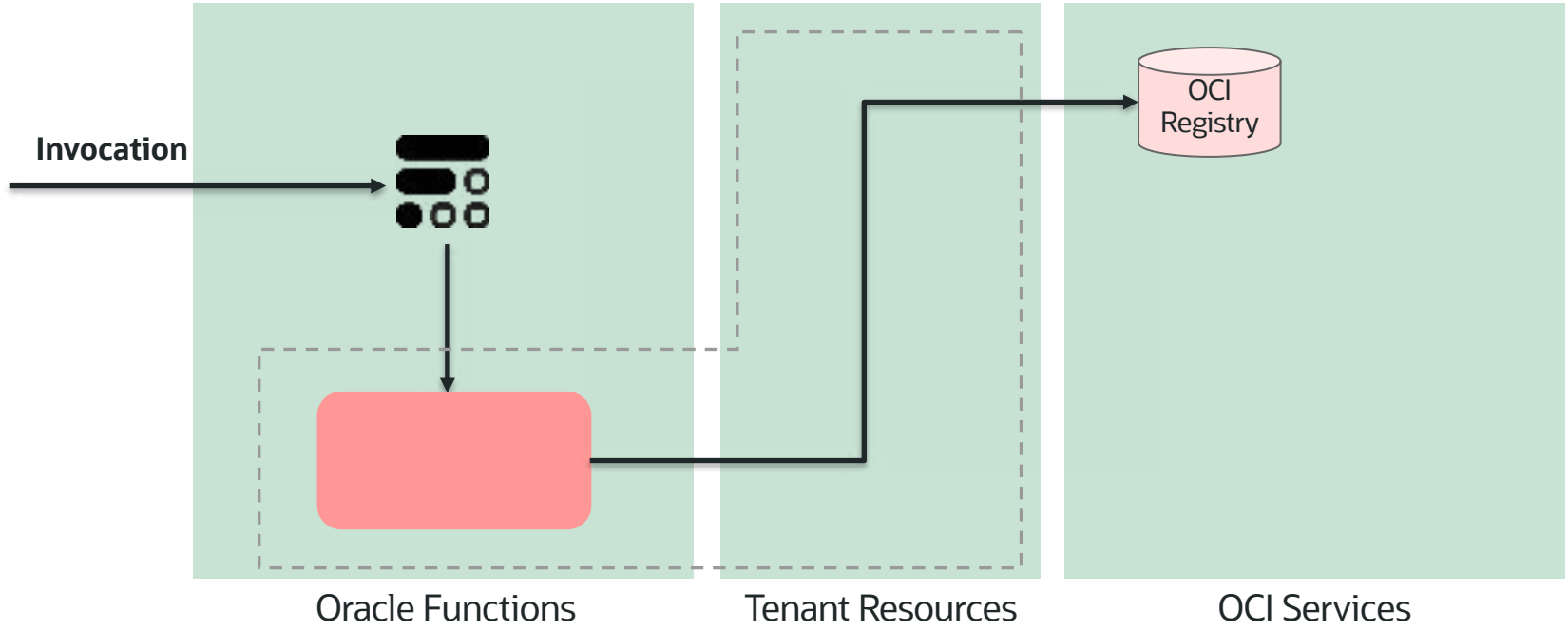
Functions – High Level Architecture



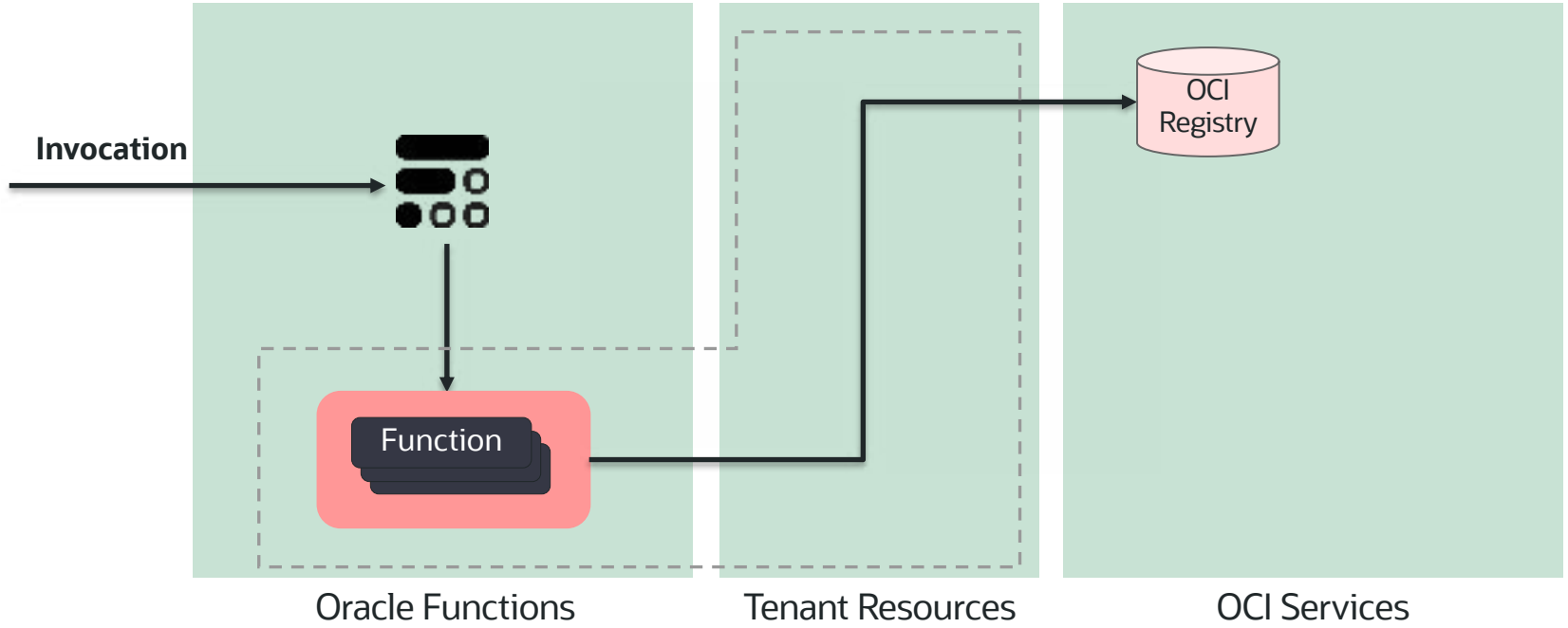
Functions – High Level Architecture



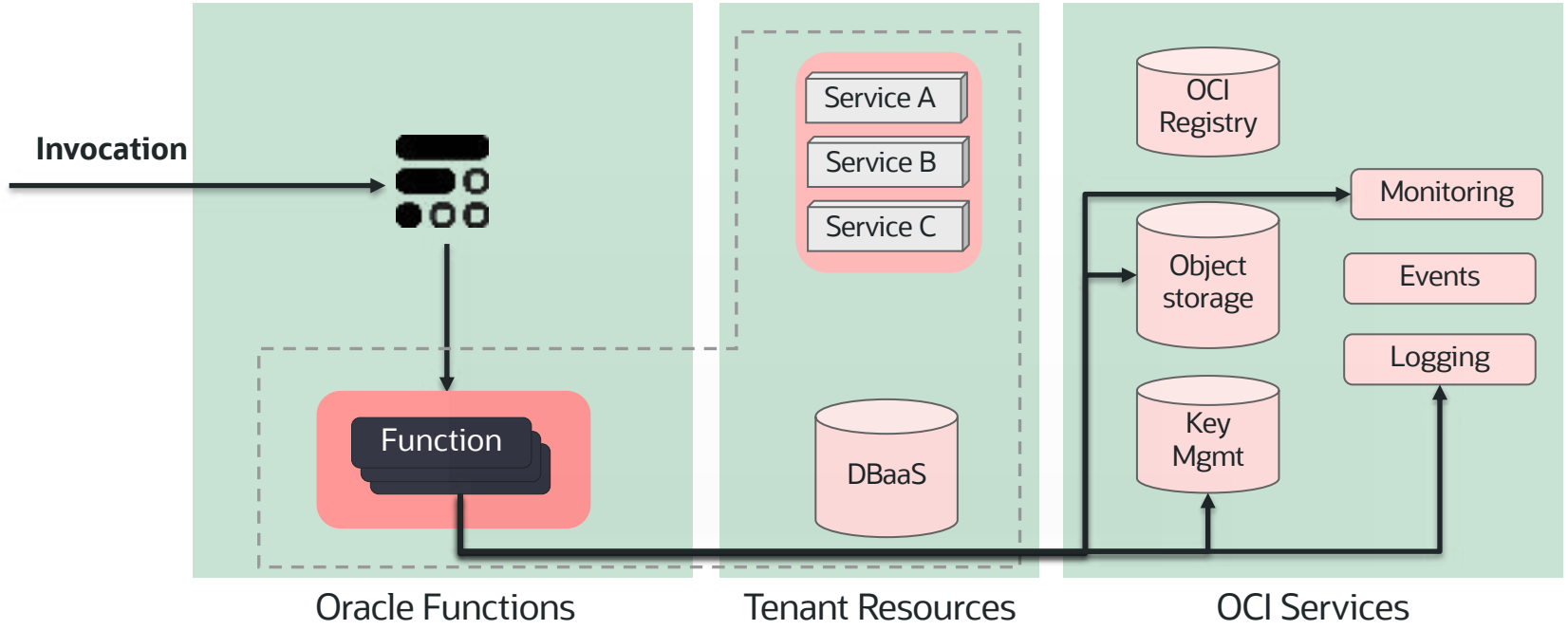
Functions – High Level Architecture



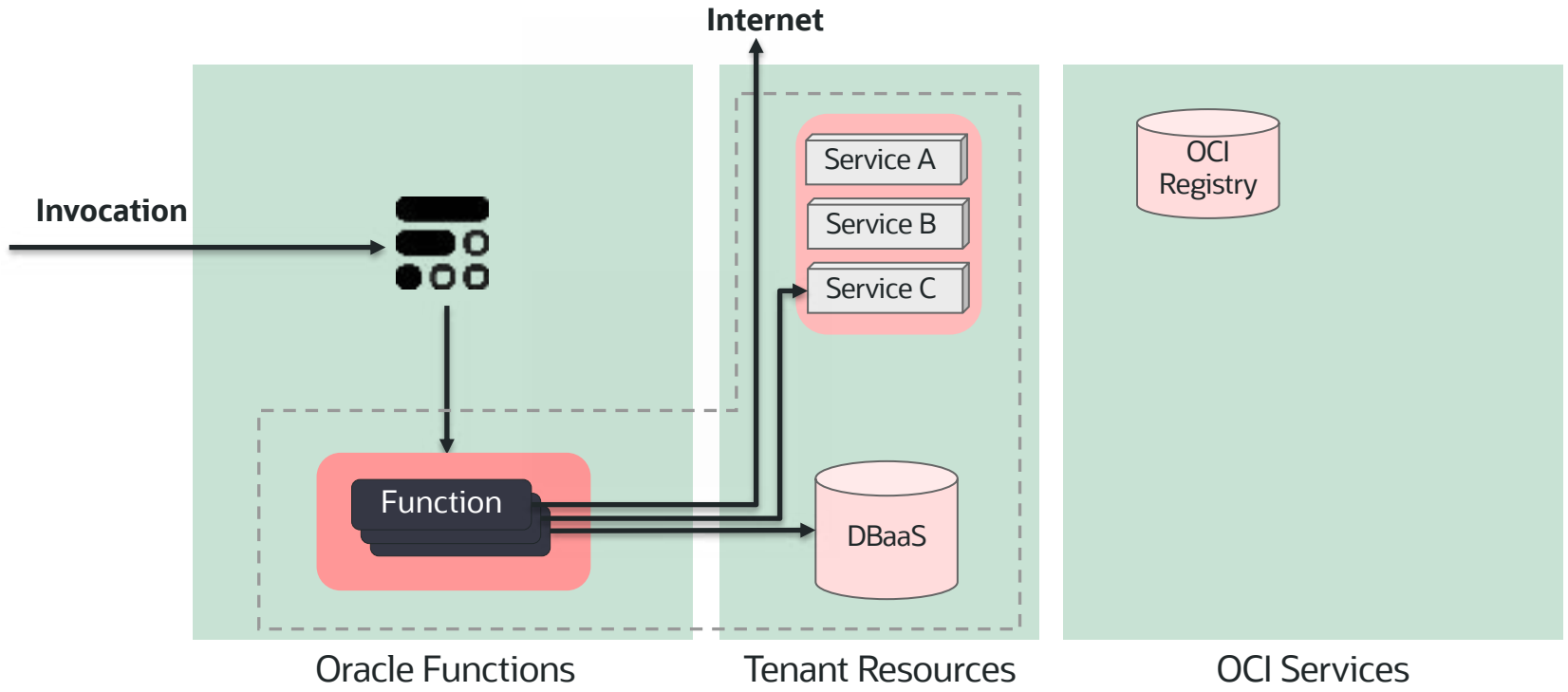
Functions – High Level Architecture



Functions – High Level Architecture

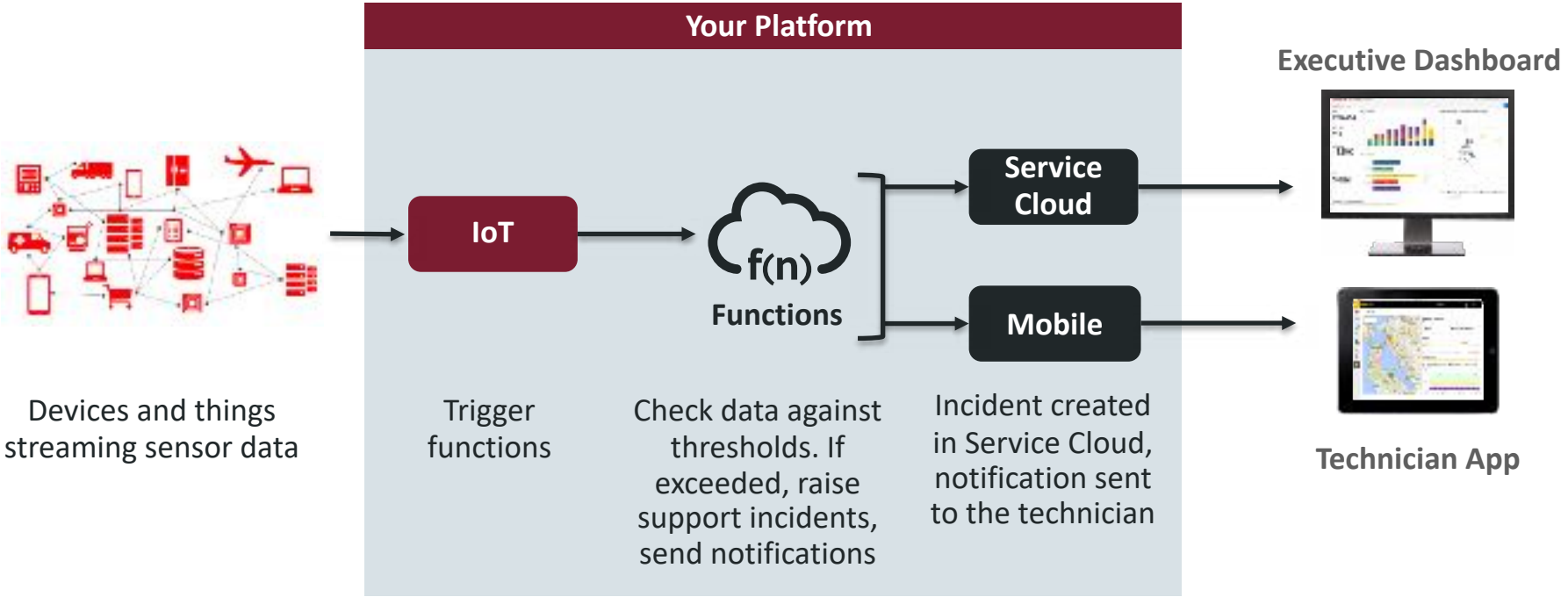


Functions – High Level Architecture

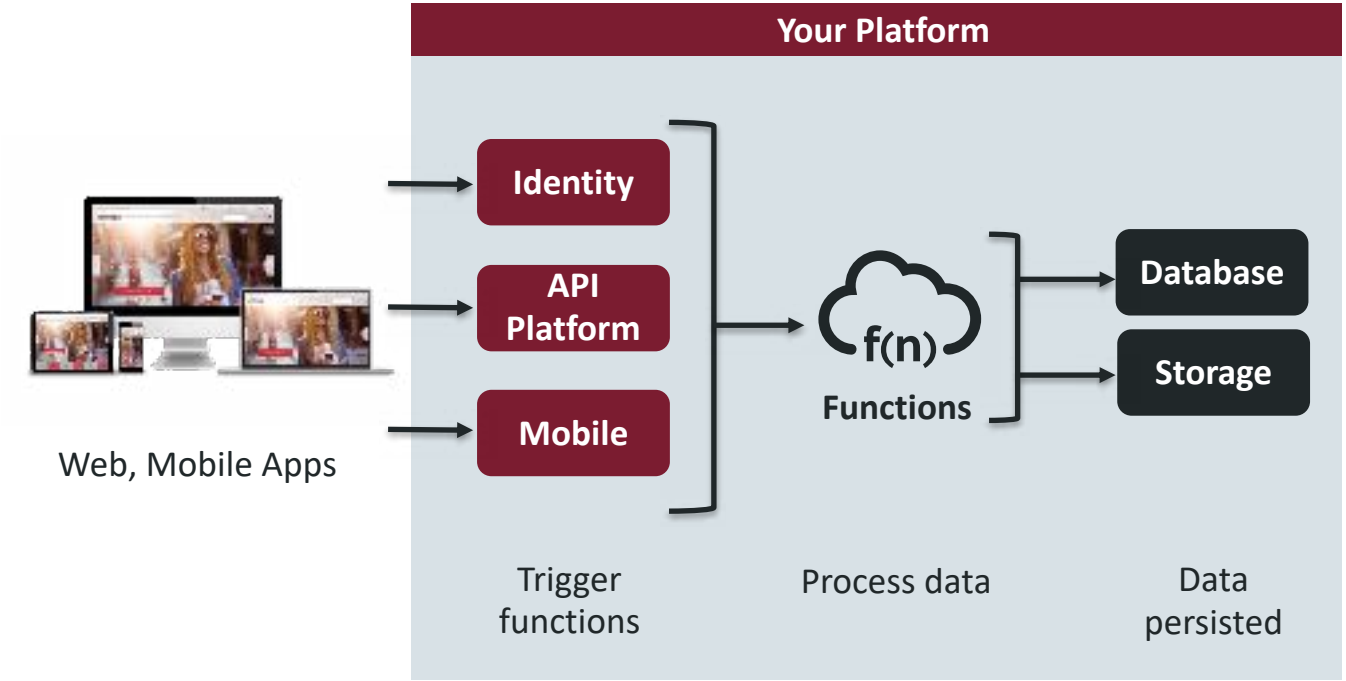


Use Cases

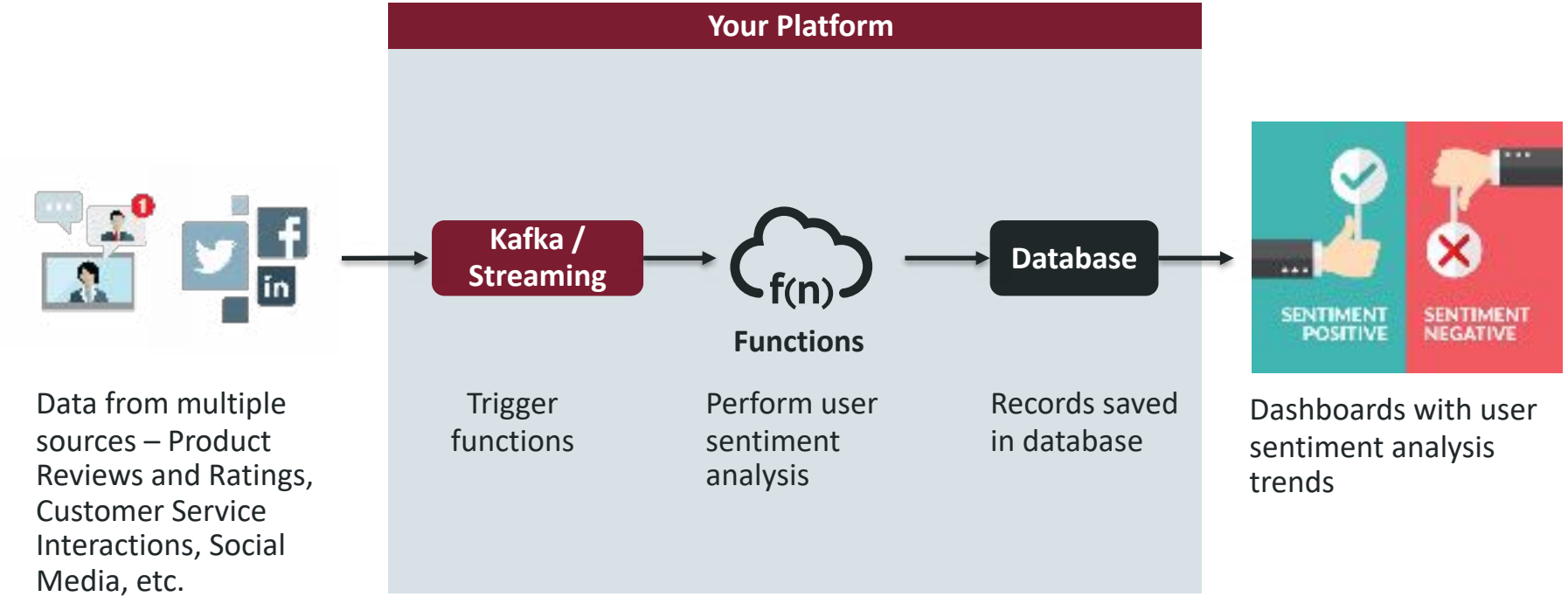
Internet of Things



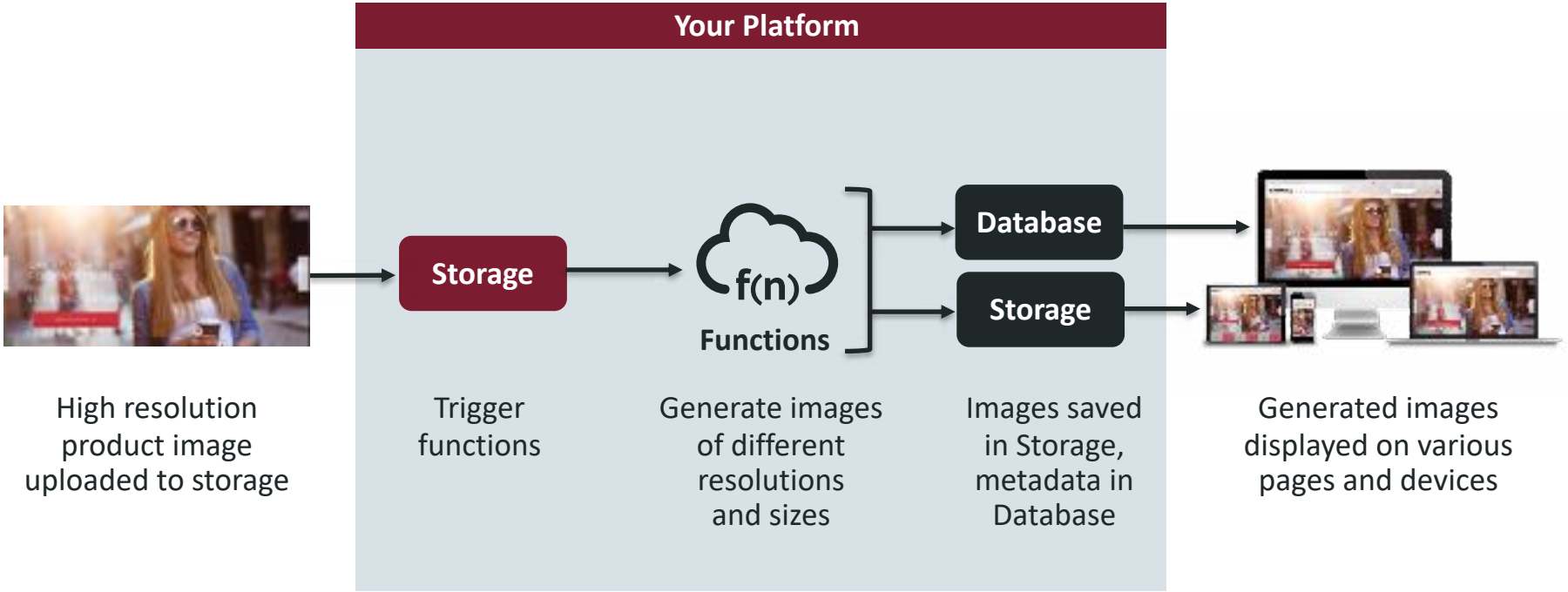
Mobile


Your Platform
Identity
API Platform
Mobile
Web, Mobile Apps
f(n)
Functions
Database
Storage
Trigger functions
Process data
Data persisted

Stream Processing



File Processing



Batch

Credit card transactions

Utility consumption

Your Platform

Database

Transaction details

Scheduled batch job

Functions

Calculate bonus points

Database

Bonus points updated

Congratulations!

Loyalty bonus received

Database

Consumption details

Scheduled batch job

Functions

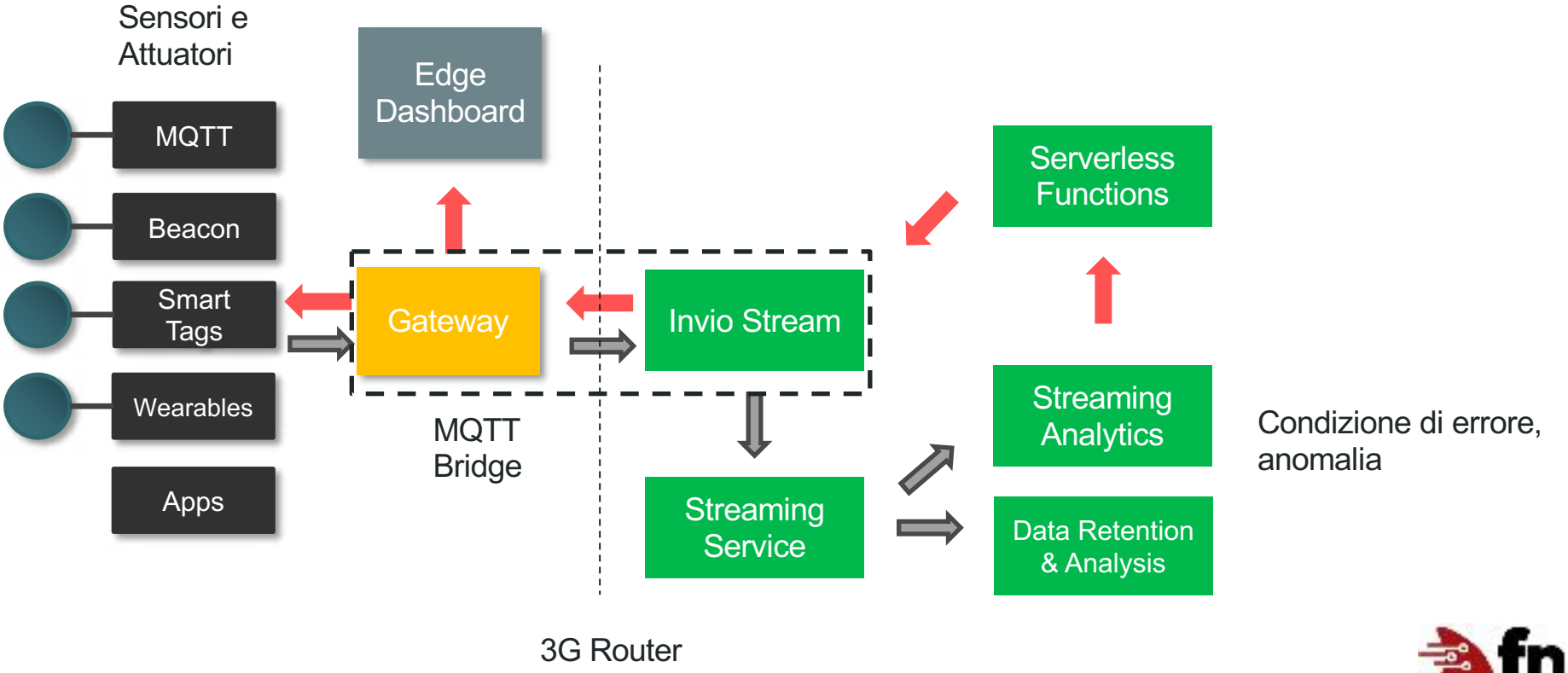
Generate utility bill PDF file

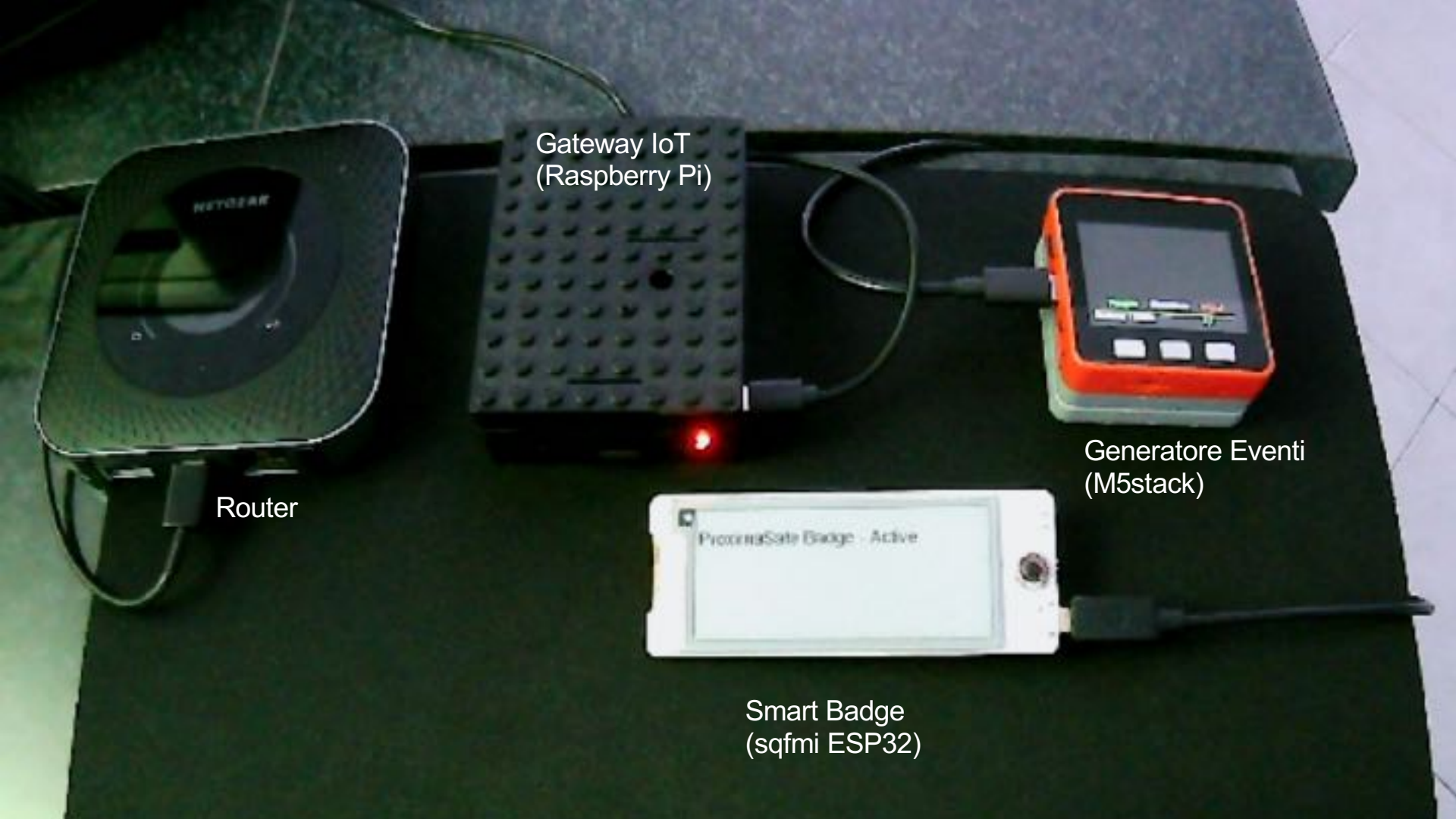
Storage

PDF files saved in Storage

Utility bill PDF file

IoT – in real world




Gateway IoT
(Raspberry Pi)
Generatore Eventi
(M5stack)
Router
Smart Badge
(sqfmi ESP32)

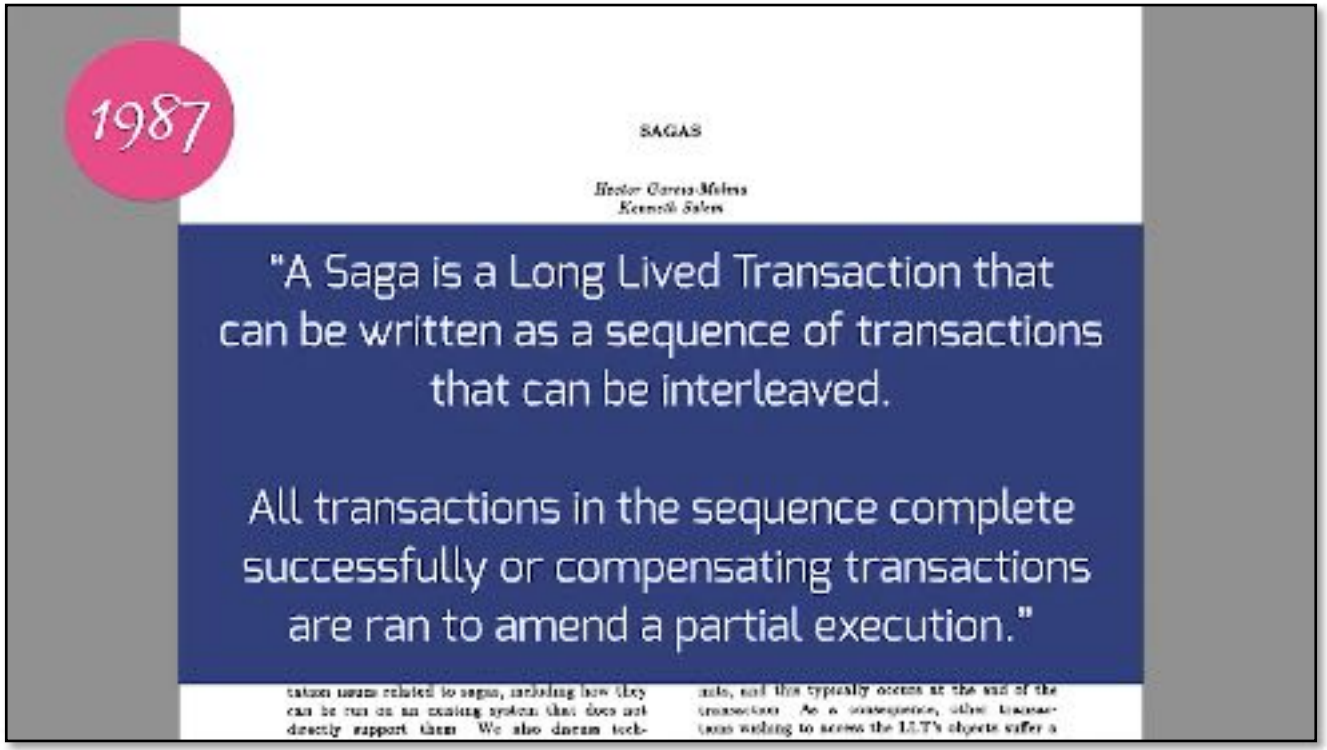
Fn Flow

Fn Flow

- Build long-running, reliable, scalable functions with rich sets of language-specific primitives including fork-join, chaining, delays and error handling
- Supports complex parallel processes that are readable and testable (including unit tests) with standard programming tools
- Java support using CompletableFuture API from Java 8 with JS, Python, Go language support on the way!

Saga

Saga


1987
SAGAS
"A Saga is a Long Lived Transaction that can be written as a sequence of transactions that can be interleaved.
All transactions in the sequence complete successfully or compensating transactions are ran to amend a partial execution."

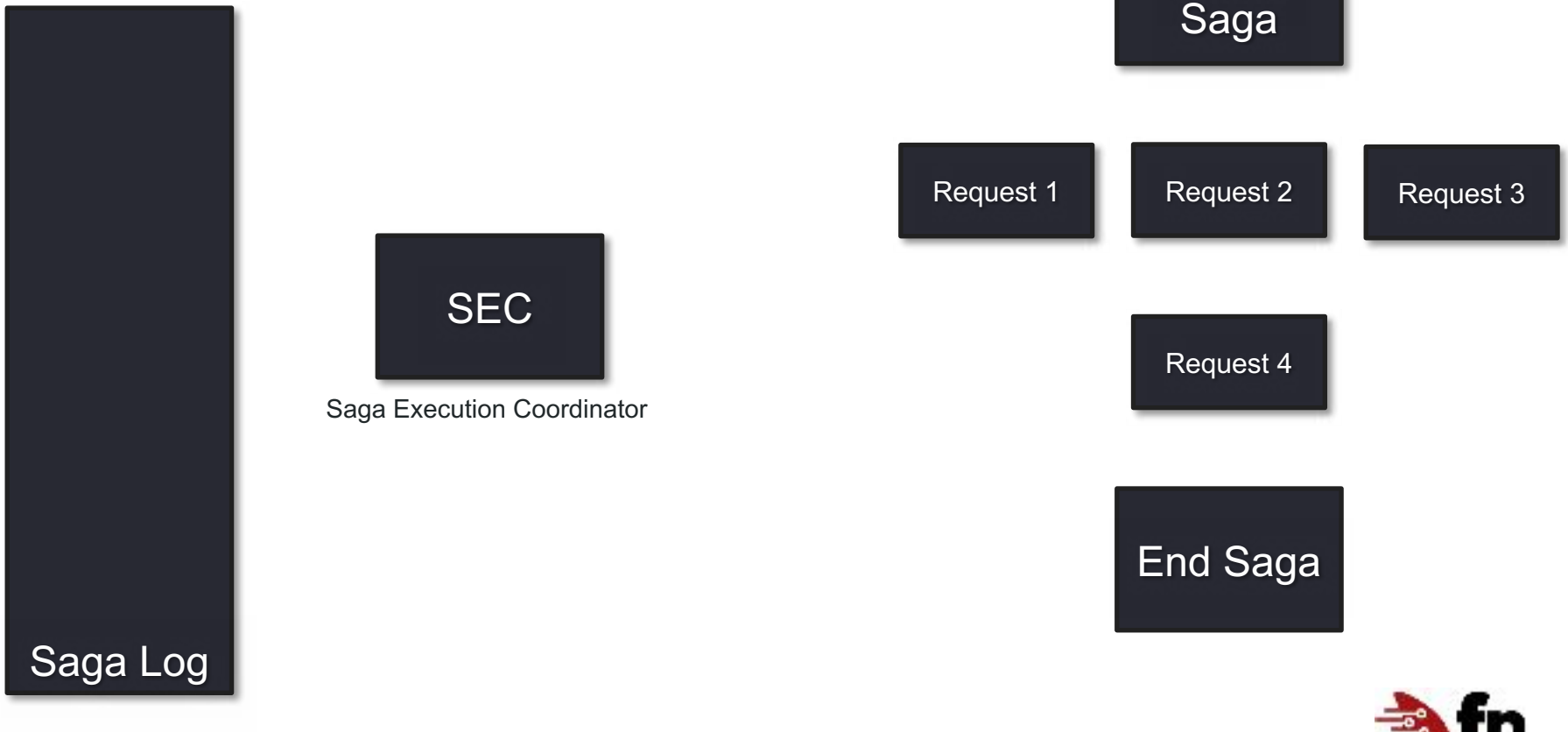
Distributed Saga

- A collection of **Requests** and **Compensating Requests** representing a single business operation

- **Saga Requests**
  - Can fail
  - Must be idempotent

- **Saga Compensating Requests**
  - Cannot Fail
  - Must be idempotent
  - Have commutative properties (2 + 7 = 7 + 2)

Architecture

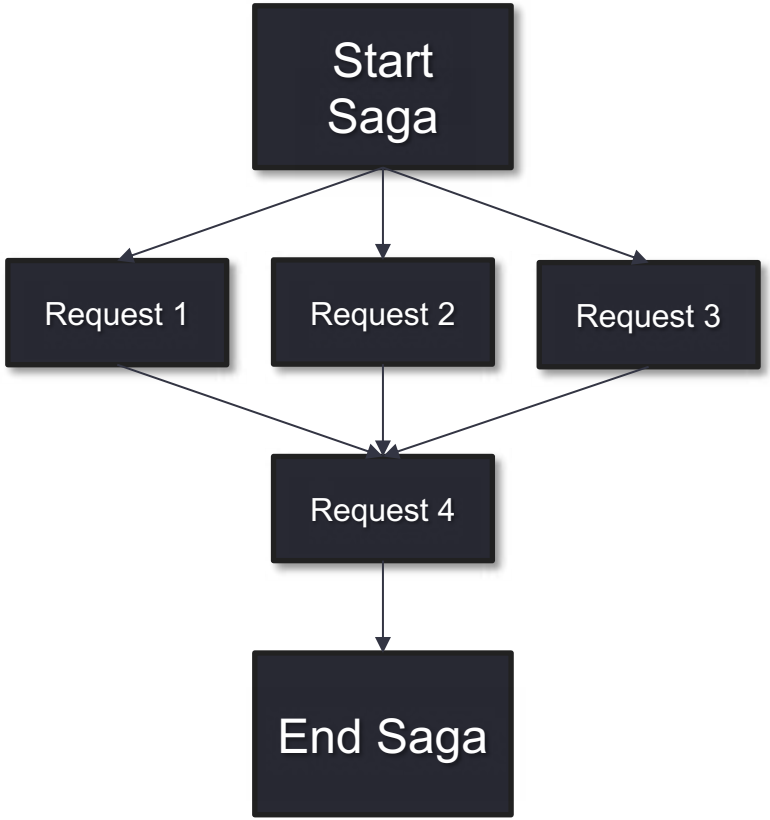


Architecture

Saga Log

SEC

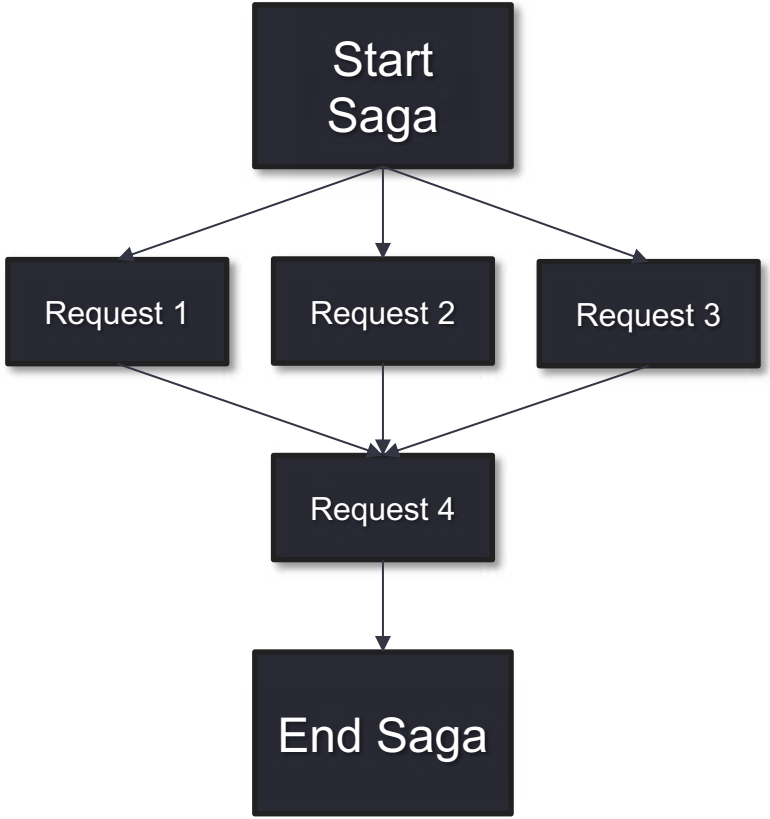
Saga Execution Coordinator



fn

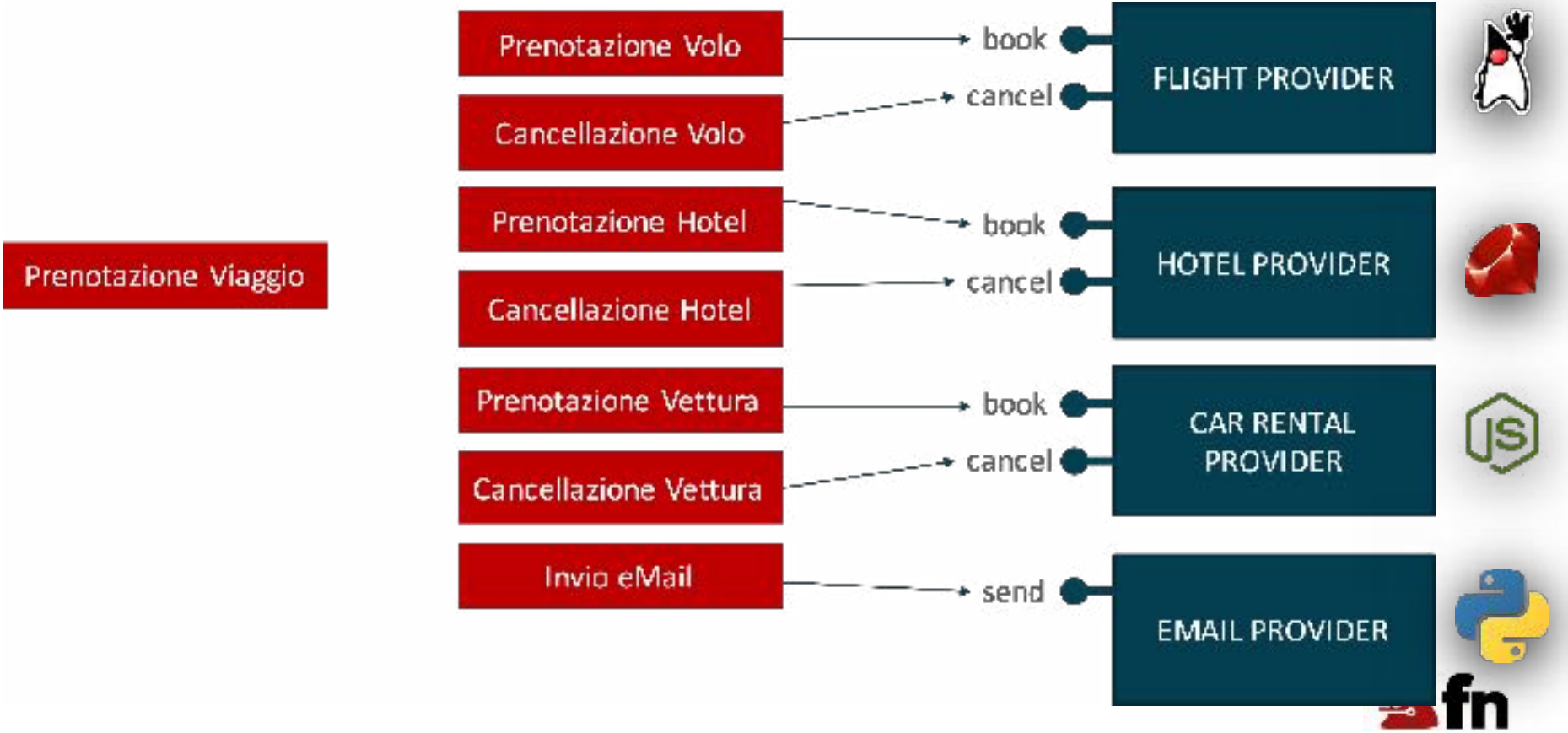
Architecture

Saga Execution Coordinator



fn

The Example


Prenotazione Viaggio
Prenotazione Volo
Cancellazione Volo
Prenotazione Hotel
Cancellazione Hotel
Prenotazione Vettura
Cancellazione Vettura
Invio eMail
book
cancel
book
cancel
book
cancel
send
FLIGHT PROVIDER
HOTEL PROVIDER
CAR RENTAL PROVIDER
EMAIL PROVIDER
fn

Future

- **future** – Construct used in concurrent programming
- Sort of placeholder waiting for a value returned by an asynchronous operation
- Define operations to be performed on this object when the async operation will be completed

sample-payload.json

```
{
  "flight": {
    "departureTime": "2018-04-14",
    "flightCode": "AZ286"
  },
  "hotel": {
    "city": "Roma",
    "hotel": "Marriot, Fiumicino Airport"
  },
  "carRental": {
    "model": "BMW X3"
  }
}
```

TripFunction.java

```
public void book2(TripReq input) {
        Flow f = Flows.currentFlow();

        FlowFuture<BookingRes> flightFuture =
            f.invokeFunction("./flight/book", input.flight, BookingRes.class);

        FlowFuture<BookingRes> hotelFuture =
            f.invokeFunction("./hotel/book", input.hotel, BookingRes.class);

        FlowFuture<BookingRes> carFuture =
            f.invokeFunction("./car/book", input.carRental, BookingRes.class);

        flightFuture.thenCompose(
            (flightRes) -> hotelFuture.thenCompose(
                (hotelRes) -> carFuture.whenComplete(
                    (carRes, e) -> EmailReq.sendSuccessMail(flightRes, hotelRes, carRes)
                )
                .exceptionallyCompose( (e) -> cancel("./car/cancel", input.carRental, e) )
            )
            .exceptionallyCompose( (e) -> cancel("./hotel/cancel", input.hotel, e) )
        )
        .exceptionallyCompose( (e) -> cancel("./flight/cancel", input.flight, e) )
        .exceptionally( (err) -> {EmailReq.sendFailEmail(); return null;} );
    }
```

Accesso al Flow Object

```
public void book2(TripReq input) {
    Flow f = Flows.currentFlow();

    FlowFuture<BookingRes> flightFuture =
        f.invokeFunction("./flight/book", input.flight, BookingRes.class);

    FlowFuture<BookingRes> hotelFuture =
        f.invokeFunction("./hotel/book", input.hotel, BookingRes.class);

    FlowFuture<BookingRes> carFuture =
        f.invokeFunction("./car/book", input.carRental, BookingRes.class);

    flightFuture.thenCompose(
        (flightRes) -> hotelFuture.thenCompose(
            (hotelRes) -> carFuture.whenComplete(
                (carRes, e) -> EmailReq.sendSuccessMail(flightRes, hotelRes, carRes)
            )
            .exceptionallyCompose( (e) -> cancel("./car/cancel", input.carRental, e) )
        )
        .exceptionallyCompose( (e) -> cancel("./hotel/cancel", input.hotel, e) )
    )
    .exceptionallyCompose( (e) -> cancel("./flight/cancel", input.flight, e) )
    .exceptionally( (err) -> {EmailReq.sendFailEmail(); return null;} );
}
```

Ritorno al Futuro

```
public void book2(TripReq input) {
        Flow f = Flows.currentFlow();

        FlowFuture<BookingRes> flightFuture =
            f.invokeFunction("./flight/book", input.flight, BookingRes.class);

        FlowFuture<BookingRes> hotelFuture =
            f.invokeFunction("./hotel/book", input.hotel, BookingRes.class);

        FlowFuture<BookingRes> carFuture =
            f.invokeFunction("./car/book", input.carRental, BookingRes.class);

        flightFuture.thenCompose(
            (flightRes) -> hotelFuture.thenCompose(
                (hotelRes) -> carFuture.whenComplete(
                    (carRes, e) -> EmailReq.sendSuccessMail(flightRes, hotelRes, carRes)
                )
                .exceptionallyCompose( (e) -> cancel("./car/cancel", input.carRental, e) )
            )
            .exceptionallyCompose( (e) -> cancel("./hotel/cancel", input.hotel, e) )
        )
        .exceptionallyCompose( (e) -> cancel("./flight/cancel", input.flight, e) )
        .exceptionally( (err) -> {EmailReq.sendFailEmail(); return null;} );
    }
```

Concatenazione

```
public void book2(TripReq input) {
        Flow f = Flows.currentFlow();

        FlowFuture<BookingRes> flightFuture =
            f.invokeFunction("./flight/book", input.flight, BookingRes.class);

        FlowFuture<BookingRes> hotelFuture =
            f.invokeFunction("./hotel/book", input.hotel, BookingRes.class);

        FlowFuture<BookingRes> carFuture =
            f.invokeFunction("./car/book", input.carRental, BookingRes.class);

        flightFuture.thenCompose(
            (flightRes) -> hotelFuture.thenCompose(
                (hotelRes) -> carFuture.whenComplete(
                    (carRes, e) -> EmailReq.sendSuccessMail(flightRes, hotelRes, carRes)
                )
                .exceptionallyCompose( (e) -> cancel("./car/cancel", input.carRental, e) )
            )
            .exceptionallyCompose( (e) -> cancel("./hotel/cancel", input.hotel, e) )
        )
        .exceptionallyCompose( (e) -> cancel("./flight/cancel", input.flight, e) )
        .exceptionally( (err) -> {EmailReq.sendFailEmail(); return null;} );
    }
```

#CodemotionConf_IT