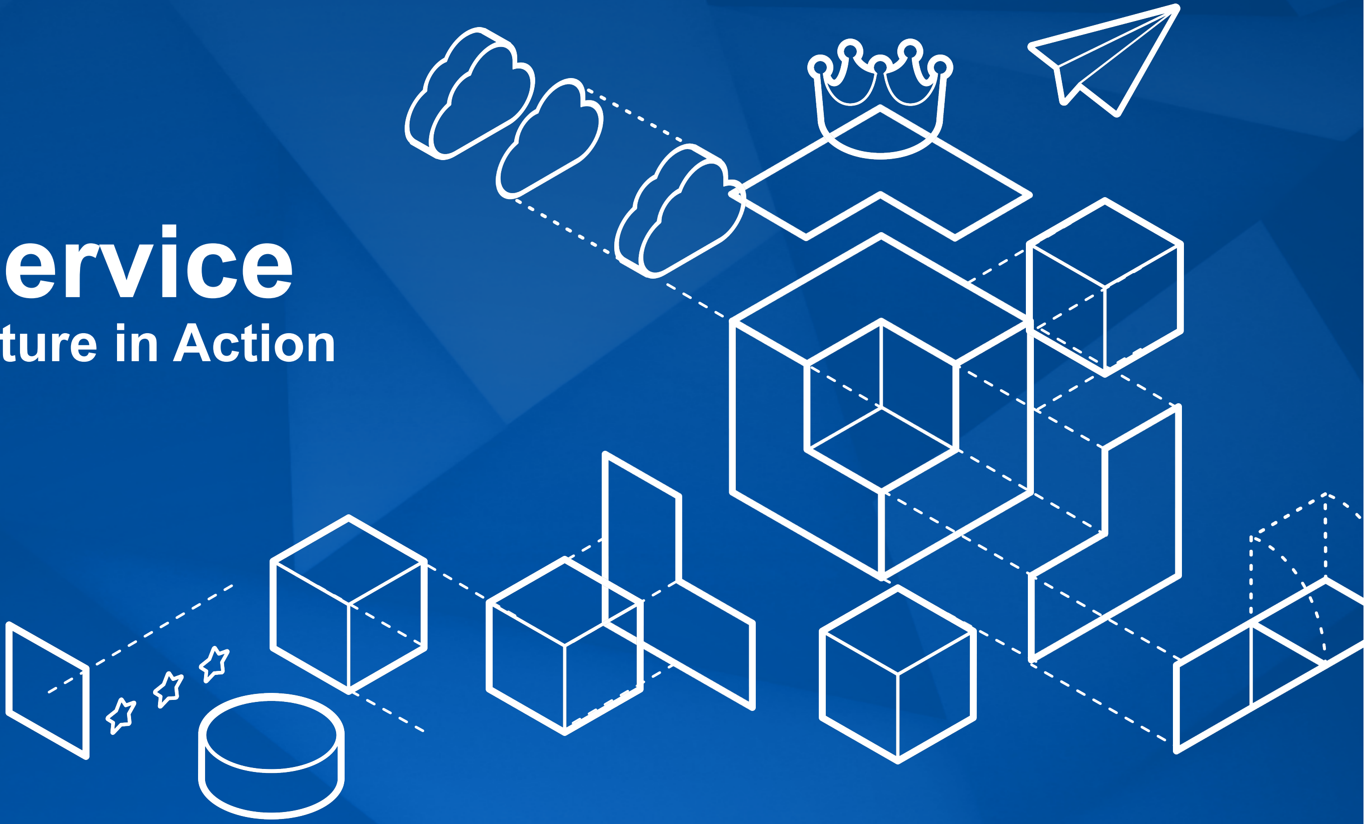
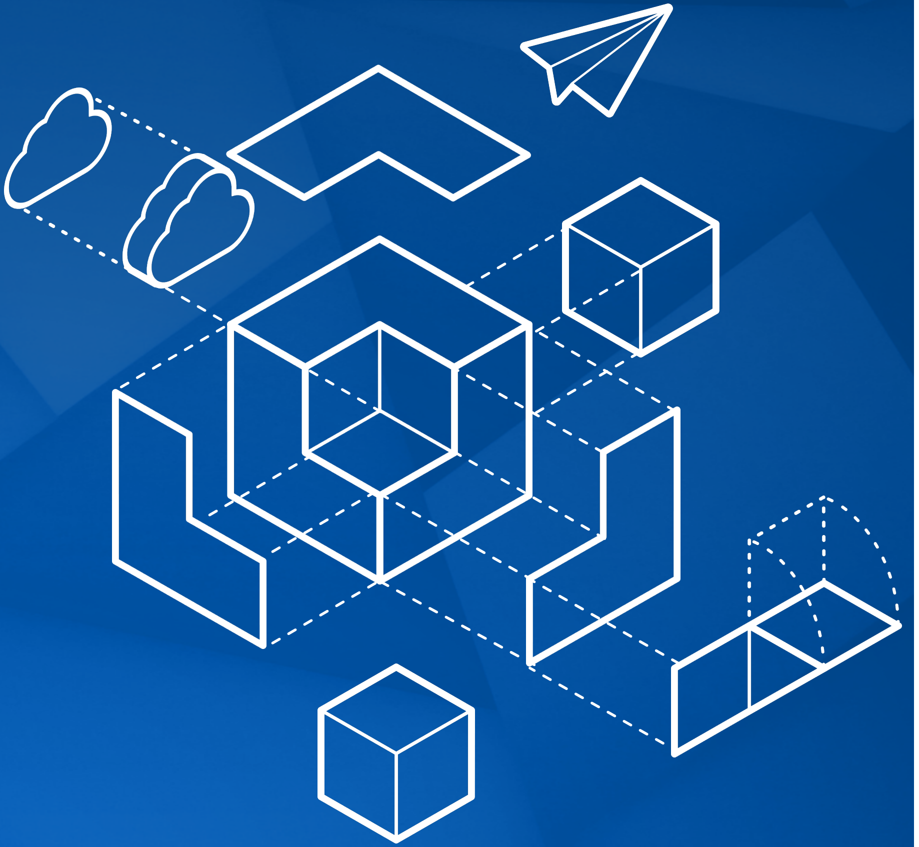
hybris-as-a-service

A Microservices Architecture in Action

Andrea Stubbe
Product Manager at hybris



The Vision



Why Microservices?

CLOUD FIRST

Scale different parts of he application independently

AUTONOMY

Independent teams, freedom to choose technologies

RETAIN SPEED

Ship new features as soon as they are done, independently

COMMUNITY

Share knowledge, ideas and extensions

Microservices sound like a good fit

A cloud platform that allows everyone to easily develop, extend and sell services and applications.

DESIGNED TO SCALE

Core services for storage, messaging, search and more are built with technologies which are known to scale

READY TO USE

Persistence, messaging, API security layer - all is there. In the cloud.

MULTI-TENANT

Infrastructure and core services are shared between all tenants

A cloud platform that allows everyone to easily develop, extend and sell services and applications.

NO SECRETS

SDK, Core APIs and guidelines are visible to everyone

NO SALES CONTACT

Just sign up and start

COMMUNITY HUB

Contribute your knowledge, and offer your own services to partners and companies

A cloud platform that allows everyone to **easily develop**, extend and sell services and applications.

OPEN

Use your favorite languages and technologies

LOW LEARNING CURVE

Tools and an active community help getting you started in minutes

SUPPORTIVE

Core APIs and SDKs are there to help you, not to restrict you

The YaaS Universe

hybris TEAMS

Offer key core services

DEVELOPERS

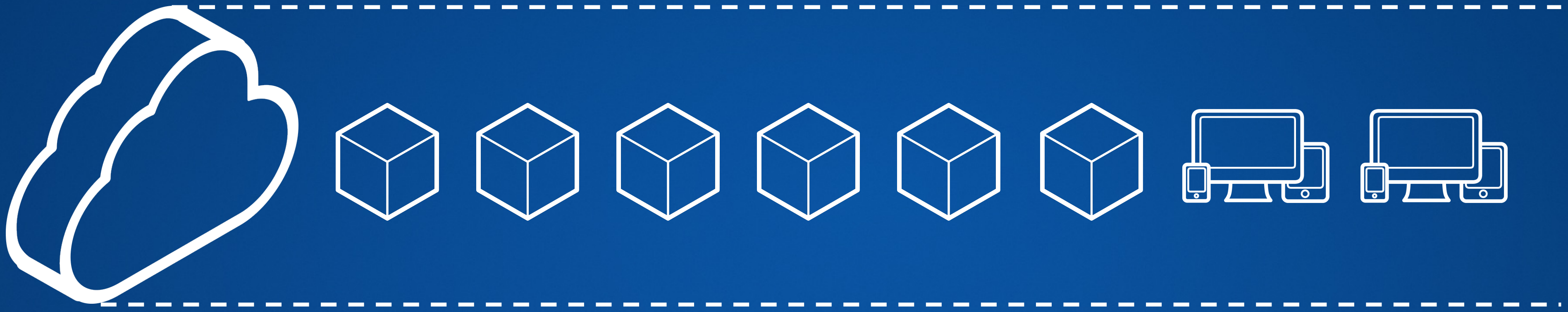
Offer services and applications and use other services

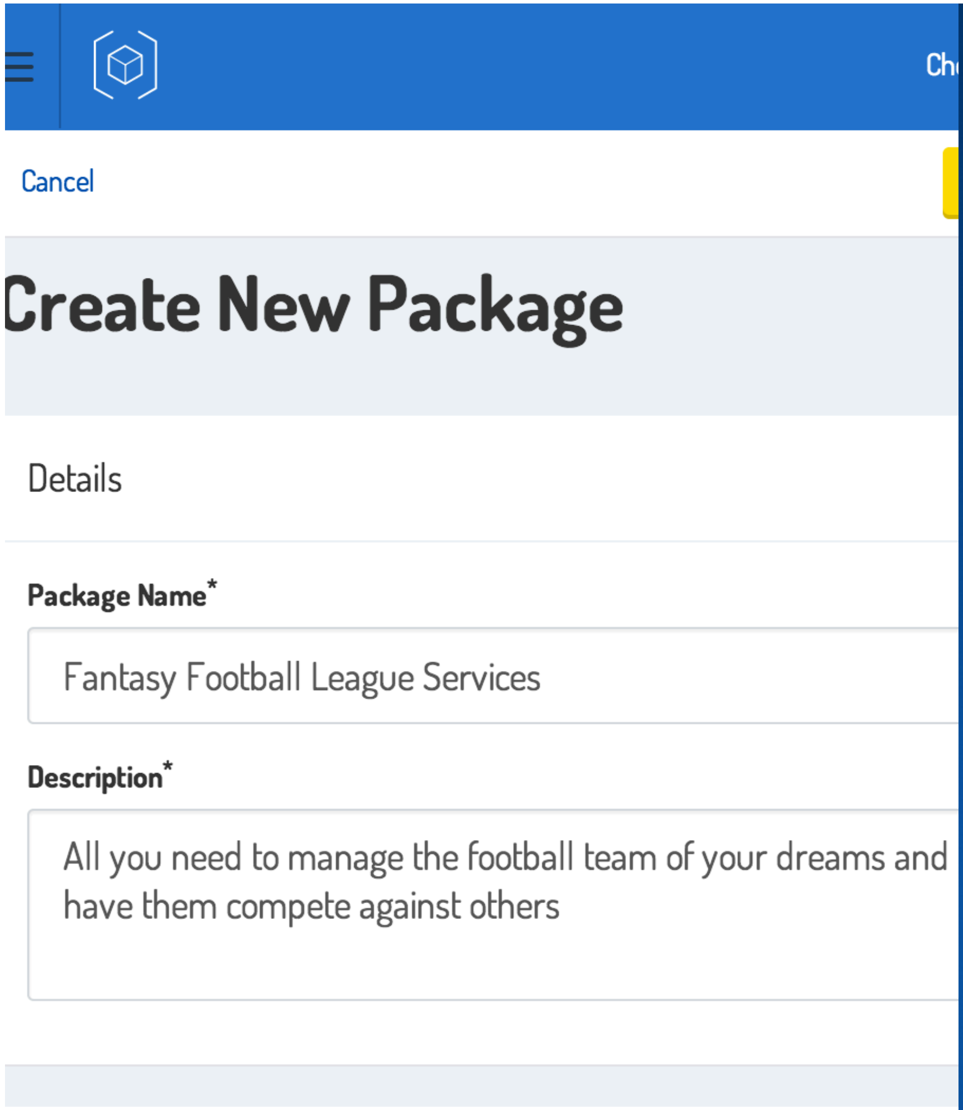
BUSINESSES

Use applications and services to engage with consumers

CONSUMERS

Use applications to interact with businesses





Create New Package

Details

Package Name*

Fantasy Football League Services

Description*

All you need to manage the football team of your dreams and have them compete against others

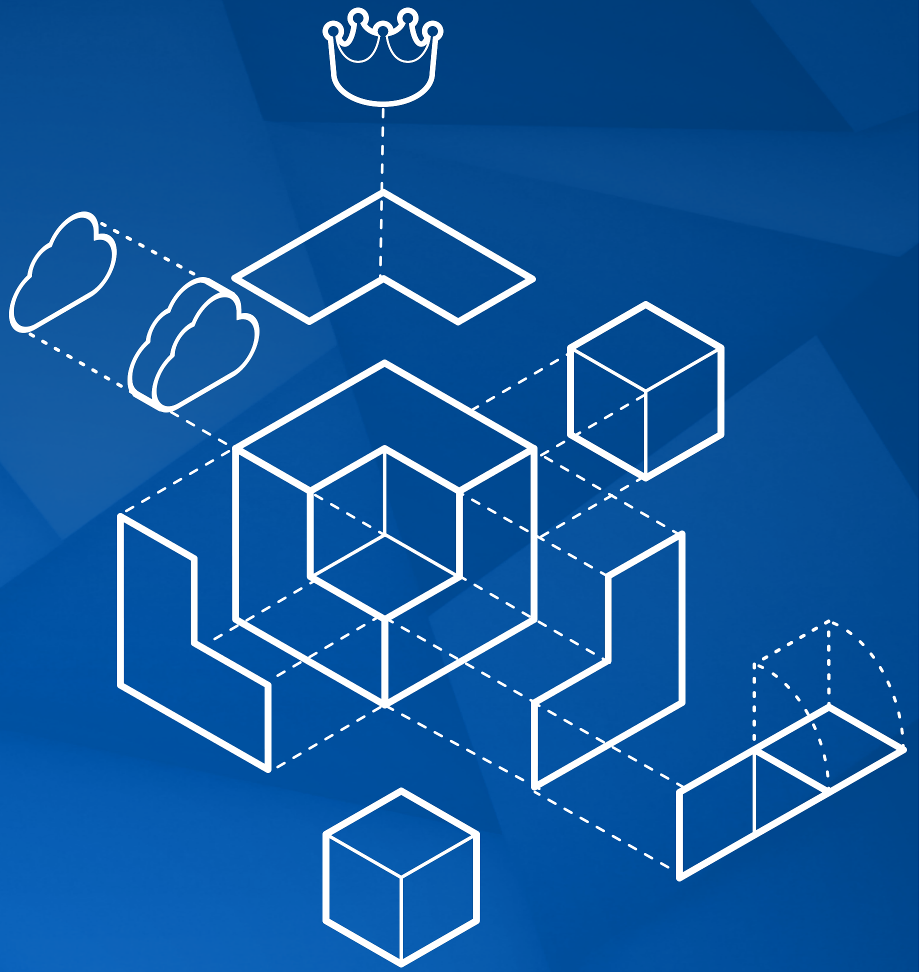
DEVELOPERS

Explore

Develop

Sell

The (γ)-Factors



The (x) Factors

OPEN TECHNOLOGY LANDSCAPE

Freedom to pick the right tool for the job

SCALABILITY OF TECHNOLOGY

Linear horizontal scalability: lower costs, less limits on maximal scalability

DON’T SURPRISE YOUR COSTUMERS

Use pre-defined patterns and best practices to ensure a consistent API and UI. Use technologies your customers know.

SMALL, INDEPENDENT SERVICES

The perfect service has zero dependencies, functionality limited to one domain. Keep the design simple.

DESIGN FOR FAILURE

If it can be down, it will be down. Design for failure and recovery.

API FIRST

Focus on developing rich APIs and develop the functionality later.

Design the API for your customers

SELF SUFFICIENT TEAMS

Teams can take a product from the concept to production with limited dependencies outside of the team

RELEASE EARLY, RELEASE OFTEN

Establish a deployment pipeline that allows to deliver without fear of breaking things

RESPONSIBILITY

You build it, you run it. And release it, scale it, maintain it, support it, improve it, …

The (ɣ) Factors - Balance

OPEN TECHNOLOGY LANDSCAPE
Freedom to pick the right tool for the job

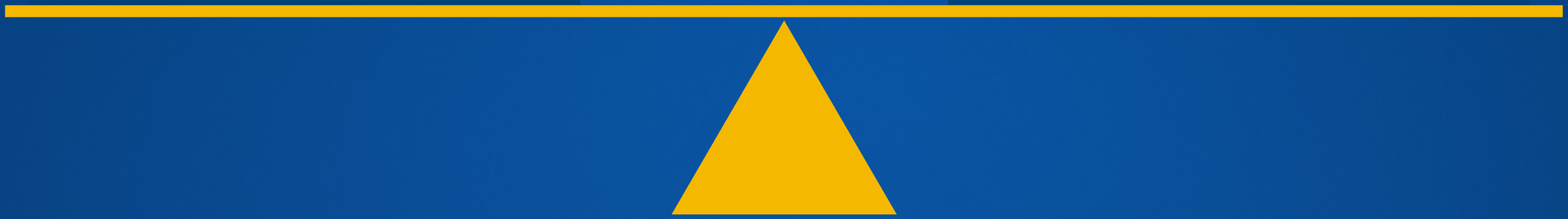
The Factors - Balance

OPEN TECHNOLOGY LANDSCAPE

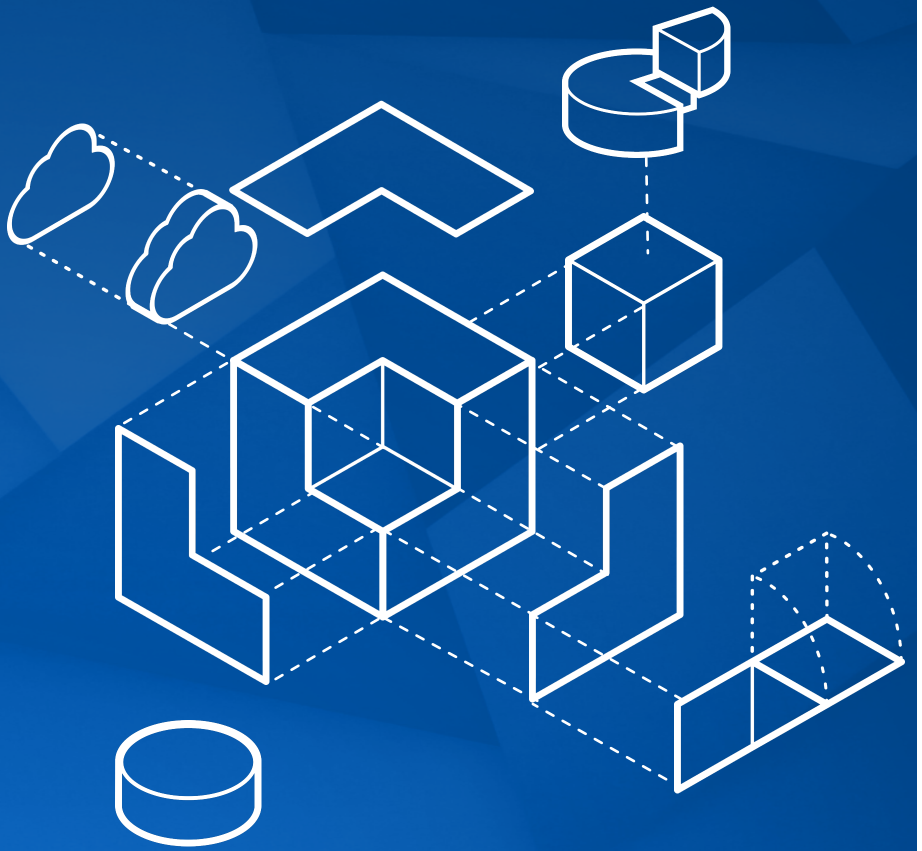
Freedom to pick the right tool for the job

RESPONSIBILITY

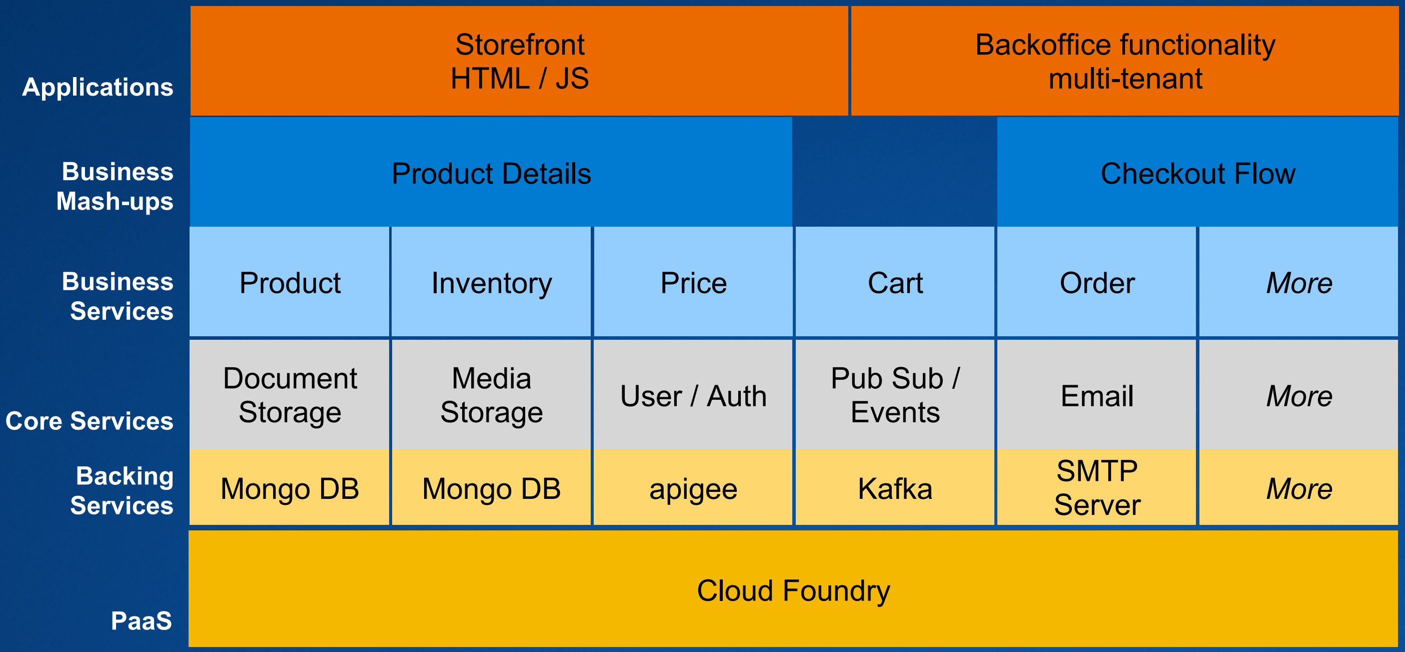
You build it, you run it. And release it, scale it, maintain it, support it, improve it, …



Architecture

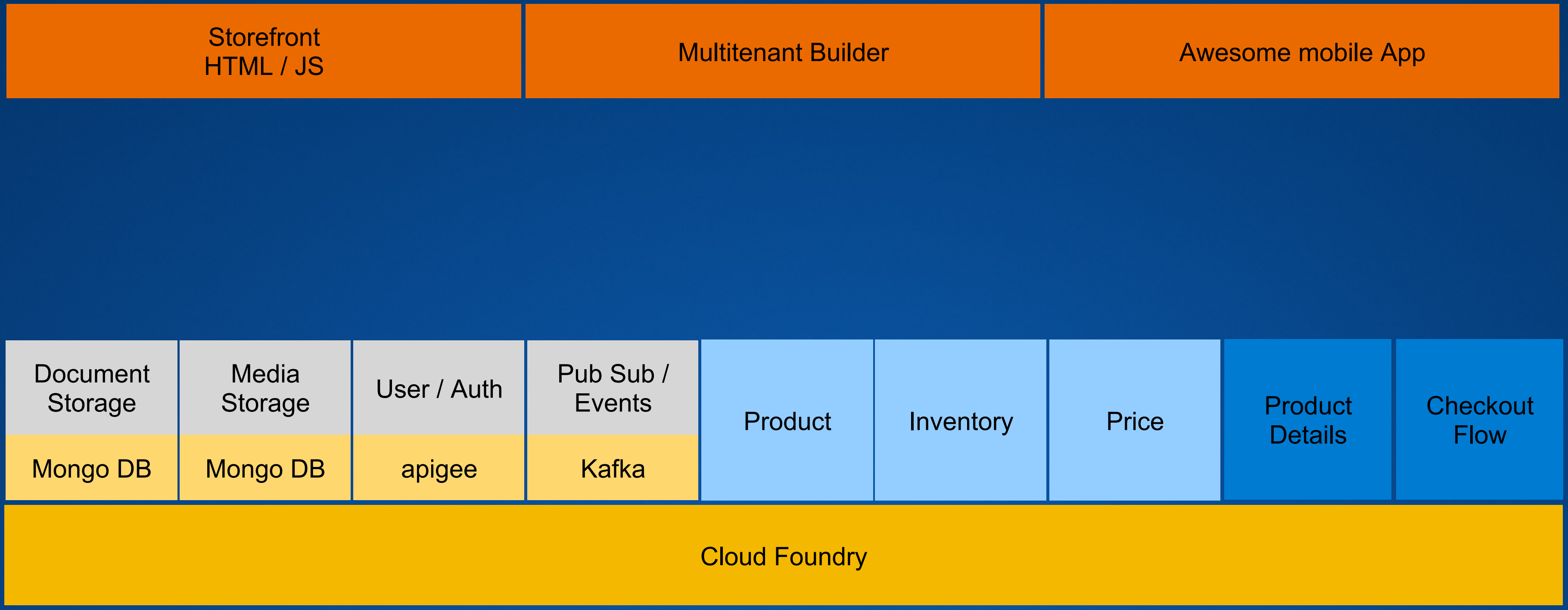


Layers…

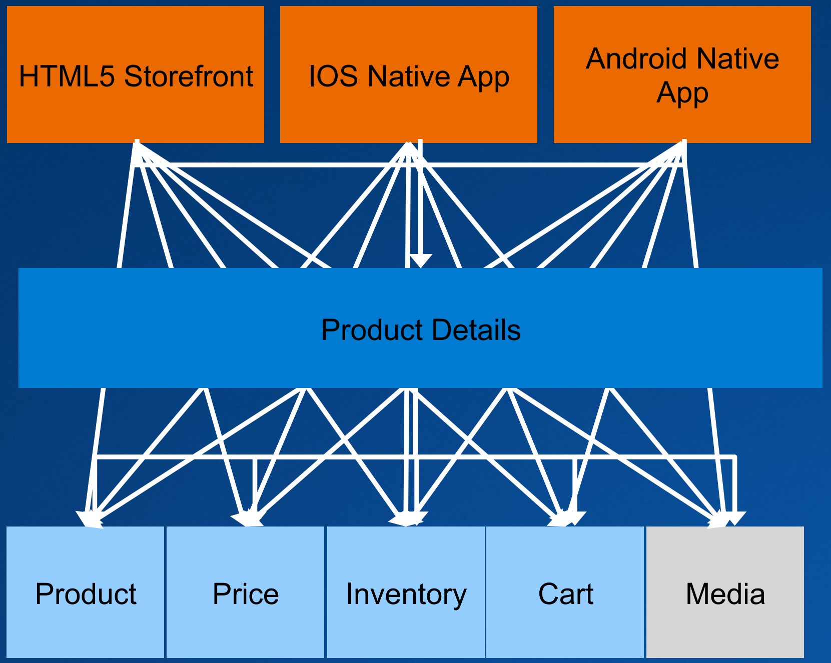


Applications	Storefront HTML / JS			Backoffice functionality multi-tenant		
Business Mash-ups	Product Details				Checkout Flow	
Business Services	Product	Inventory	Price	Cart	Order	*More*
Core Services	Document Storage	Media Storage	User / Auth	Pub Sub / Events	Email	*More*
Backing Services	Mongo DB	Mongo DB	apigee	Kafka	SMTP Server	*More*
PaaS	Cloud Foundry					

… or just a set of APIs



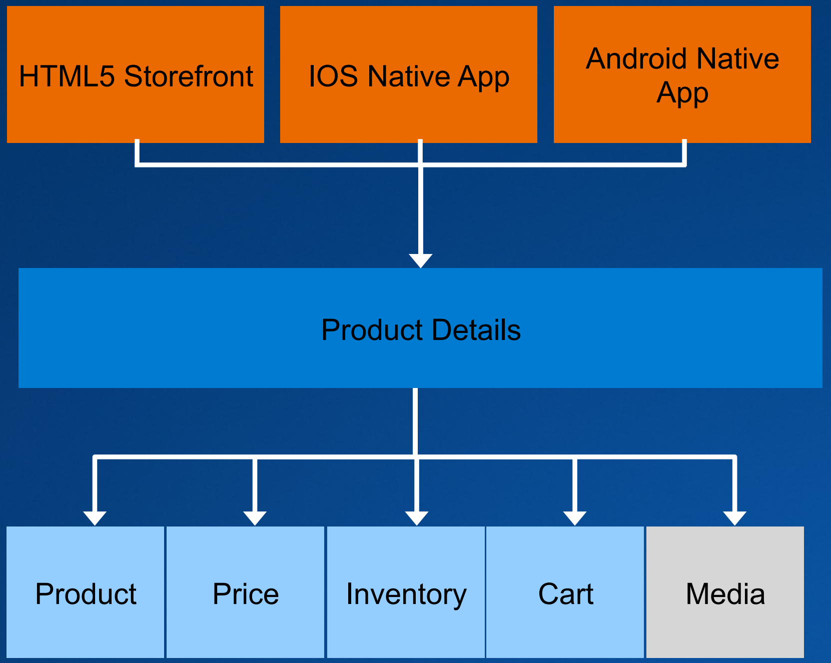
The Role of Mash-ups



If clients would use microservices directly, it…

- ★ moves a lot of business logic & error handling logic to the clients
- ★ requires multiple requests for standard flows

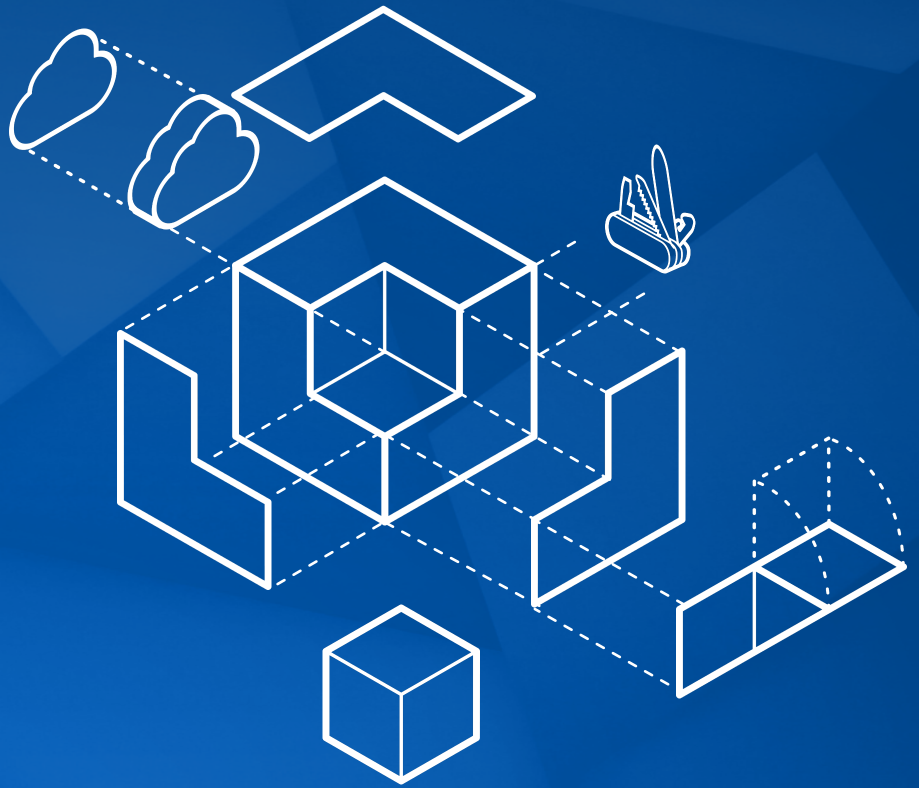
The Role of Mash-ups



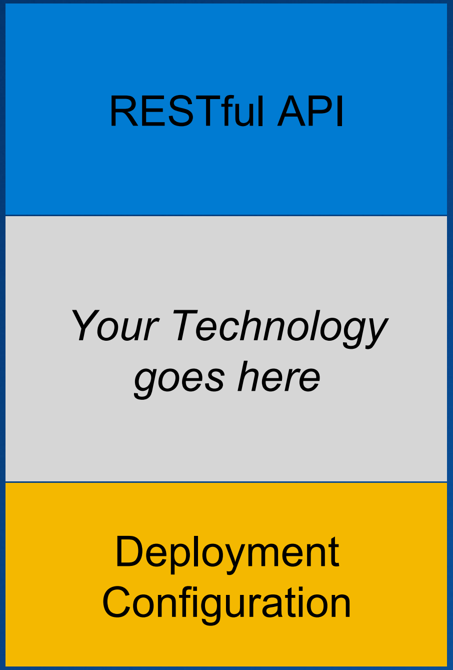
Mash-ups can be used to aggregate service calls or to compose service flows

- ★ higher performance
- ★ optimized APIs for applications
- ★ More consistent behavior of applications
- ★ promotes isolation of functionality into microservices (as it moves most dependencies into mash-up layer)

Writing Microservices



The Anatomy of a Service



- ★ Services are consumed over RESTful APIs
- ★ Deployment Configuration matching your containers / infrastructure
- ★ Everything in between is up to you!

The Anatomy of a hybris Service

RESTful API

Your Technology goes here

Deployment Configuration

Modeled in RAML
JSON for payloads
Traits and schemas

RX Java
Hystrix
Groovy, Scala,
Go, Node, Ruby, …

Environment variables
Build packs
Cloud Foundry

DIY – A Service in 3 Simple Steps

1. Develop your service, API first

2. Deploy it to any platform you like

3. Offer it on the App Exchange

Use our Microservices Development Kit

DEFINE THE API

Using RAML, a simple, open language to model RESTful APIs with YAML and JSON

USE THE TEMPLATE

Maven based archetype for Java projects

Basic Java project
API implementation stub
API documentation

We use RAML to define APIs

```
/products:
  type: collection
  get:
    is: [paged]
    description: Gets all products
  post:
    description: Creates a new product

  /{productId}:
    type: element
    get:
      description: Gets a product

    put:
      description: Updates a product

    delete:
      description: Deletes a product
```

- ★ The RESTful API Modeling Language is an open spec, built on standards such as YAML and JSON
- ★ It encourages reuse through pattern-sharing (schemas, traits, types)
- ★ Broad tool support to design and test APIs, and to generate server and client code

Common traits ensure consistency

```
traits:
- !include http://api.yaas.io/patterns/v1/trait-paged.yaml

/products:
   get:
     is: [ paged ]
```

```
http://api.yaas.io/products?pageNumber=2&pageSize=10
```

Share schemas for input and output

```
schemas:
- error: !include http://api.yaas.io/patterns/v1/schema-error.json

 ...400:
      body:
        application/json:
            schema: error
```

```
{
    "status": 400,
    ”info": "https://developer.yaas.io/errors/missing.header",
    "message": "Missing header"
}
```

Generate a service stub

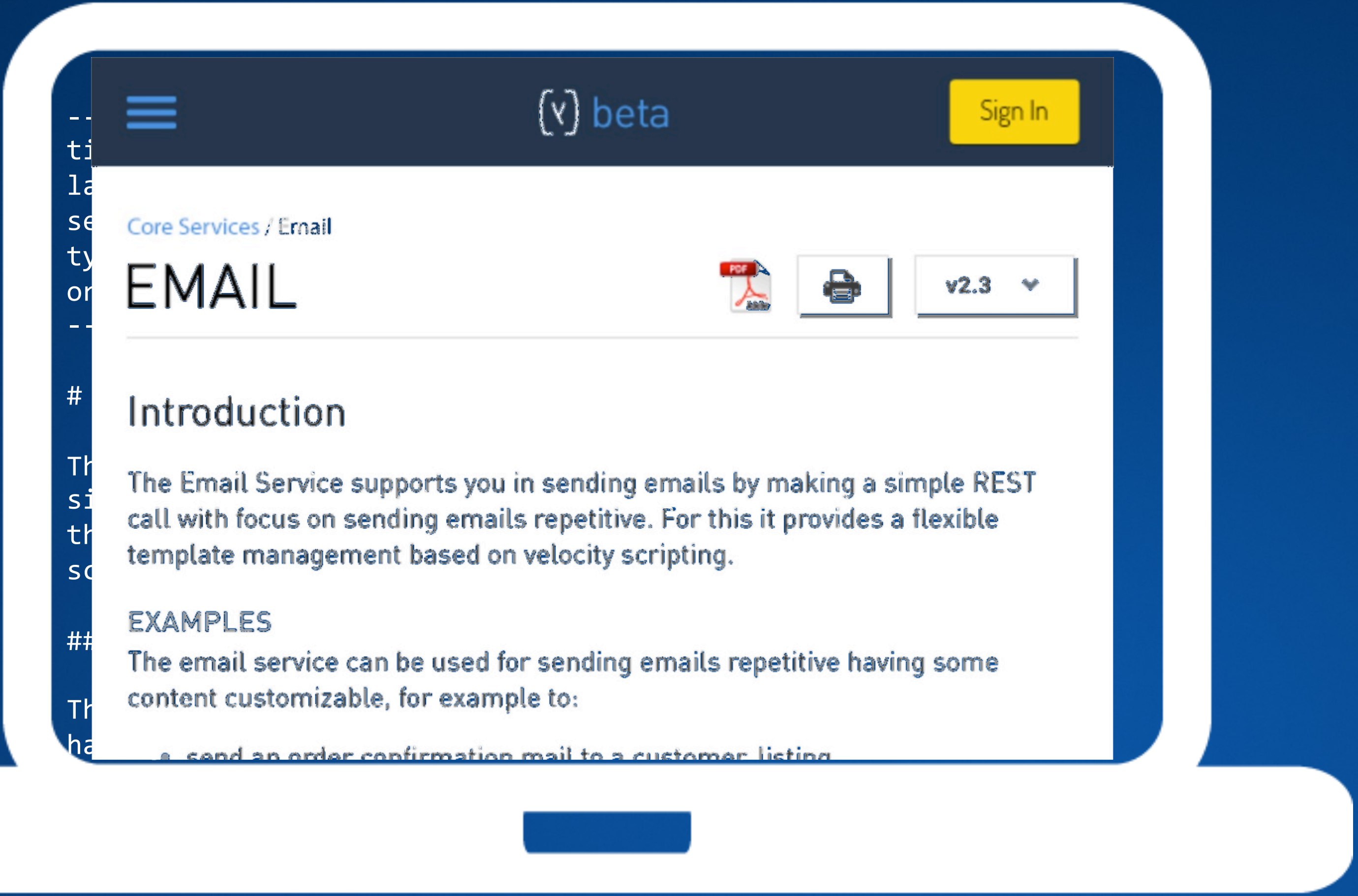
```
# Three simple commands
mvn archetype:generate [group, artifact, version]
mvn clean install
mvn jetty:run


# Play with the API in the API Console
http://localhost:8080
```

Use existing microservices *

Secured with OAuth 2.0
One access token for all APIs

* Data and media storage, events, mail, configuration, authorization, authentication, customer, product, order, cart, category, coupons, price, tax, shipping costs

Documentation as part of the codebase



```
--
ti
la
se
ty
or
--

#

Th
si
th
sc

##

Th
ha
```

Core Services / Email

EMAIL

Introduction

The Email Service supports you in sending emails by making a simple REST call with focus on sending emails repetitive. For this it provides a flexible template management based on velocity scripting.

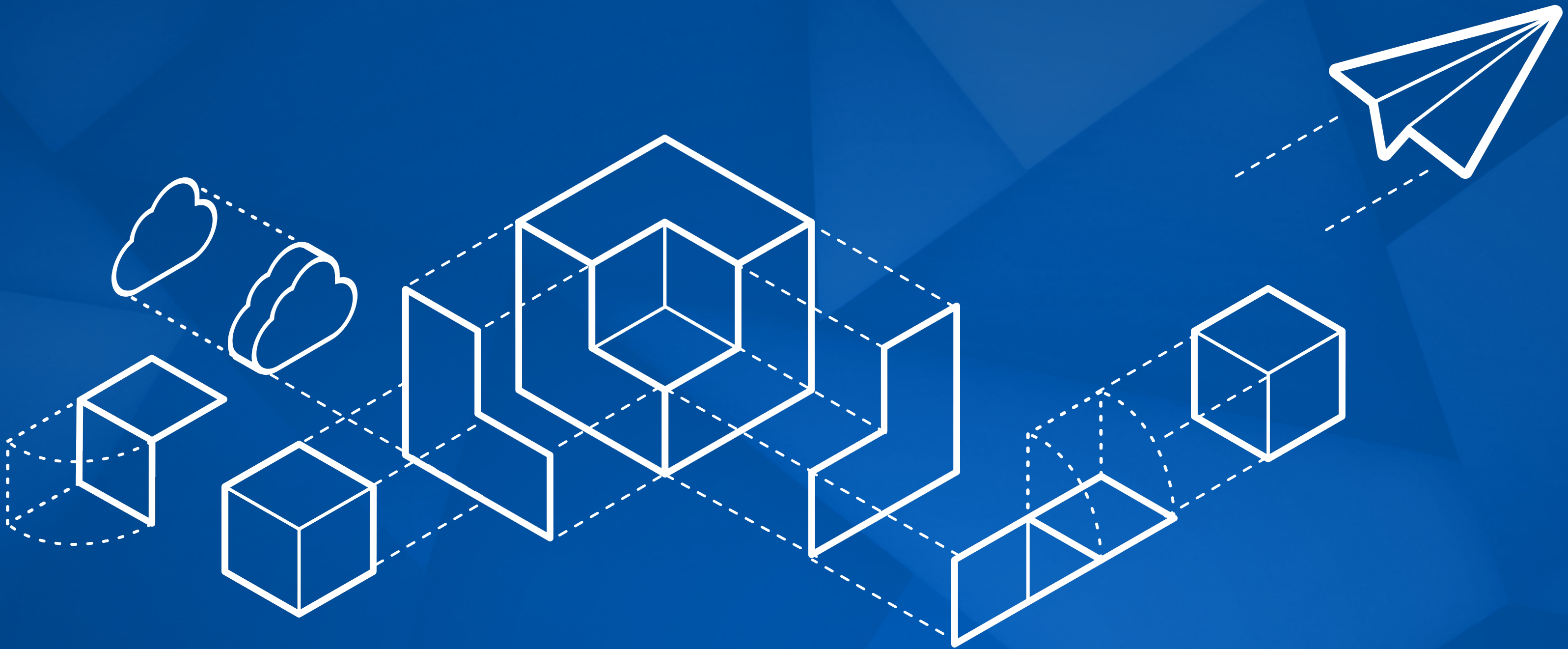
EXAMPLES

The email service can be used for sending emails repetitive having some content customizable, for example to:

- send an order confirmation mail to a customer listing

DEMO

WRITE A SERVICE. REALLY FAST.



What you just saw

SDKs to develop microservices, API first

Builder to manage your services and packages

Secured with OAuth2, https, and an API gateway

Offer your services on the **App Exchange**

Services for general functionality

REGISTER ON YAAS.IO

Join our community to build and exchange cloud-based enterprise services, and innovate faster.

THANK YOU