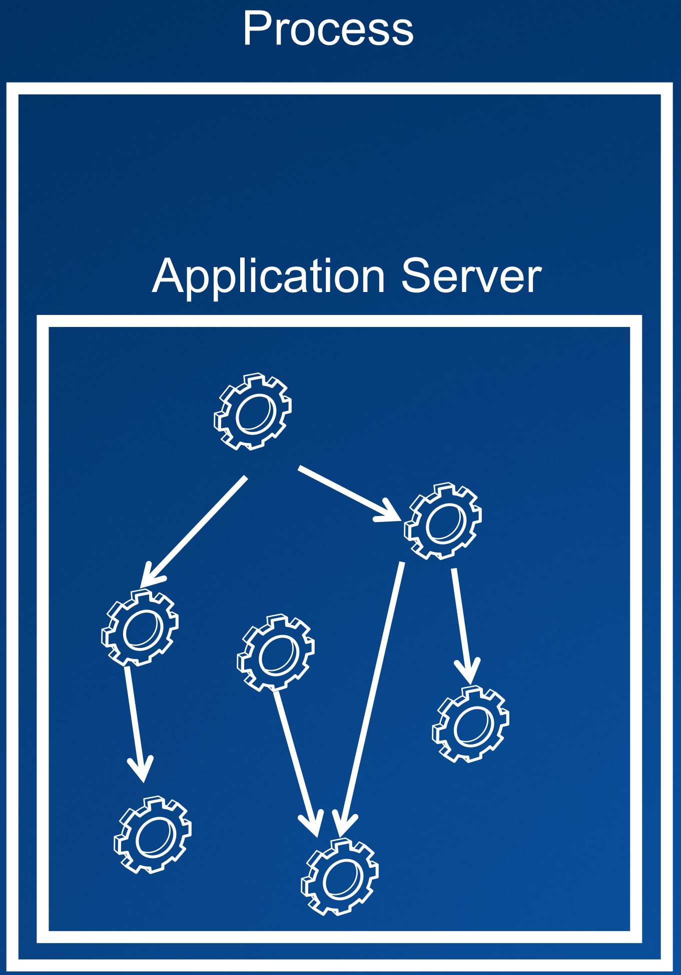
Introduction to YaaS Services

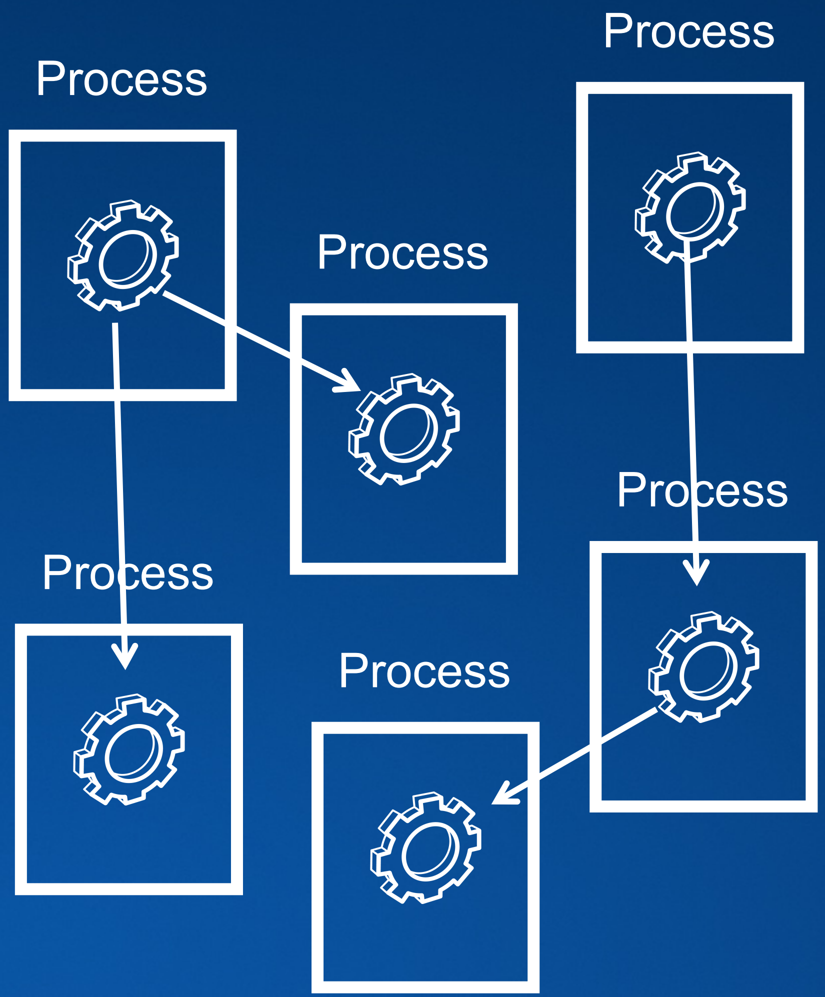
Sam Schneider
Sr. Director, YaaS GTM

"In short, the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API"

http://martinfowler.com/articles/microservices.html

Monolithic Architecture vs. Microservices





One size fits all optimization (CPU, Memory, IO) at App Server / Process level

Resource optimization per service / process

Innovate or Die?

http://blogs.wsj.com/cio/2015/10/05/innovate-or-die-the-rise-of-microservices/

Four principle benefits

- **Agility** – partial updates and deployments of a system
- **Efficiency** – efficient use of code and infrastructure
- **Resiliency** – no single point of failure
- **Revenue** – faster iteration and less downtime can translate to higher revenue

Can we Stand on the Shoulders of Giants?

Many companies have made the transition to microservices from monolithic architectures

GILT GROUPE

SOUNDCLOUD

NETFLIX

amazon

comcast®

UBER

Most have done so for greater agility, resiliency and scaling potential

Werner Vogels (Amazon CTO) on "Microservices"…

before they were cool (2006)

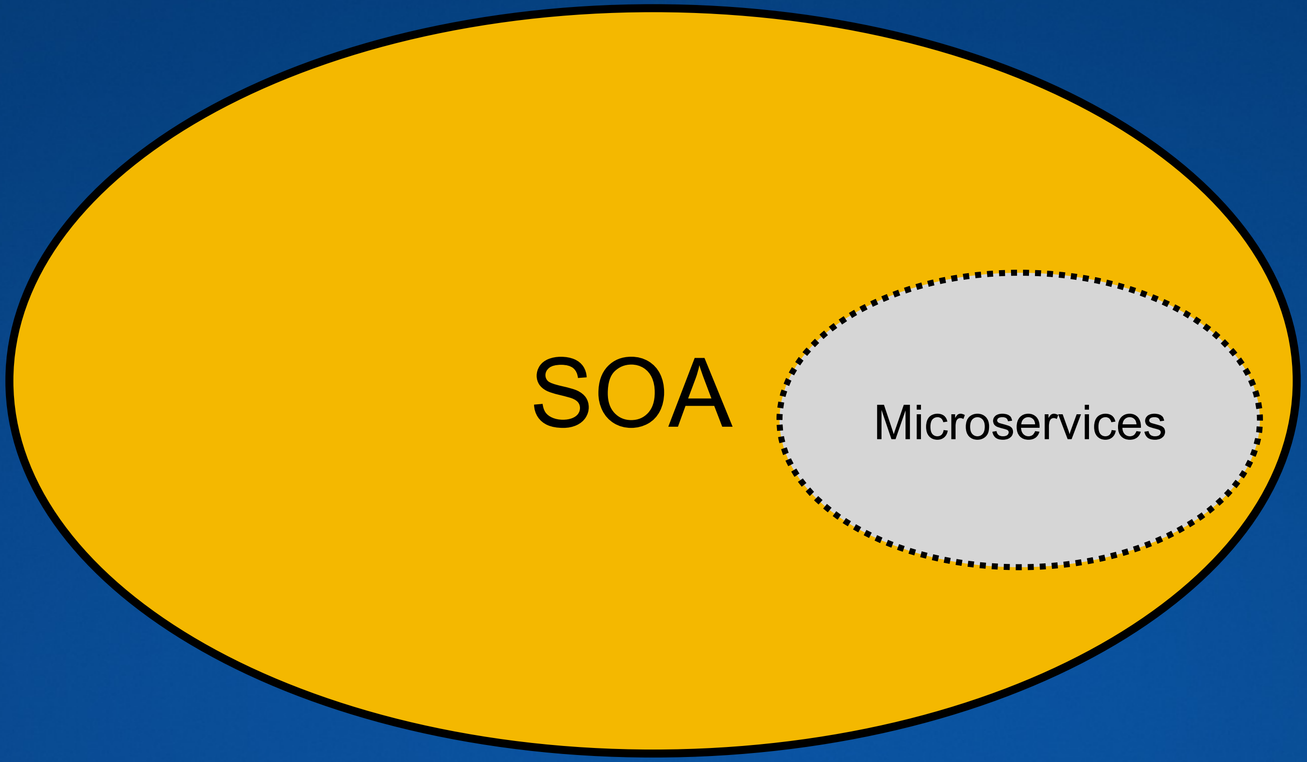
"We went through a period of serious introspection and concluded that a service-oriented architecture would give us the level of isolation that would allow us to build many software components rapidly and independently. By the way, this was way before service-oriented was a buzzword. For us service orientation means encapsulating the data with the business logic that operates on the data, with the only access through a published service interface. No direct database access is allowed from outside the service, and there's no data sharing among the services."

"If you hit the Amazon.com gateway page, the application calls more than 100 services to collect data and construct the page for you."

http://queue.acm.org/detail.cfm?id=1142065

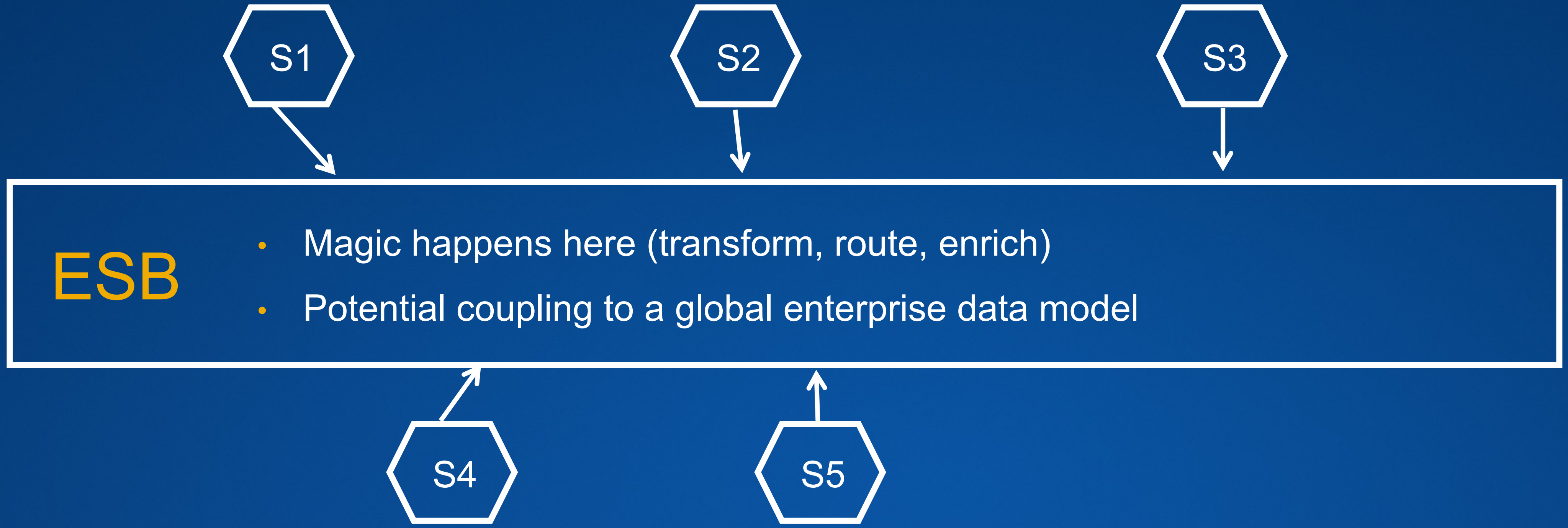
Microservices: A New Idea?

- Microservices can be thought of as a well-defined subset of SOA, ergo not new



What Key Parts of SOA Does Microservices Remove?

- One key interpretation of SOA involves an **ESB** (Enterprise Service Bus) mediating communication



- Magic happens here (transform, route, enrich)
- Potential coupling to a global enterprise data model

- Communications in microservices are a mix of point-to-point and asynchronous messaging – typically via simple queues

(x)Factors – Principles for the Cloud

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

MONITOR EVERYTHING

Specify everything, monitor and alert

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, …

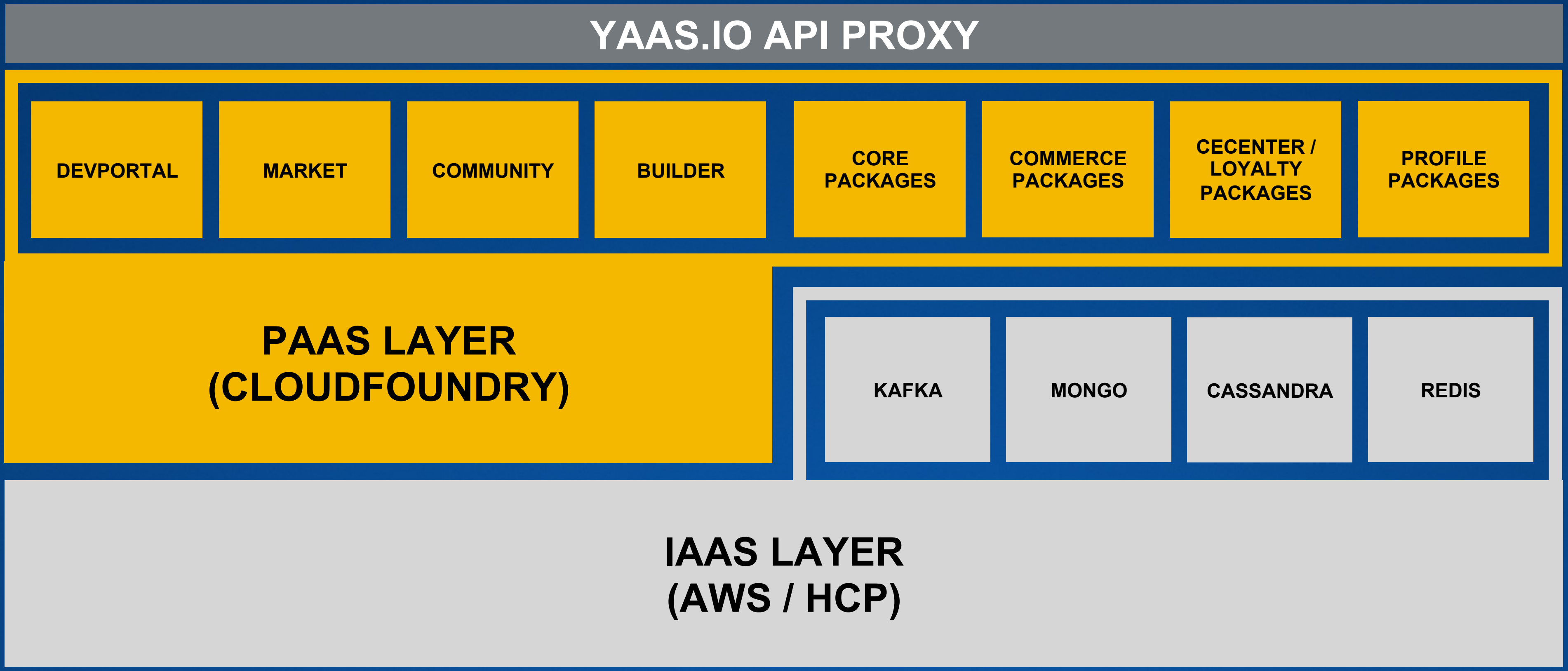
Unabashedly Borrowed from *The 12 Factor App*

The methodology outlined by the 12 Factor App outlines the experiences with development, operating, and scaling services in the cloud https://12factor.net

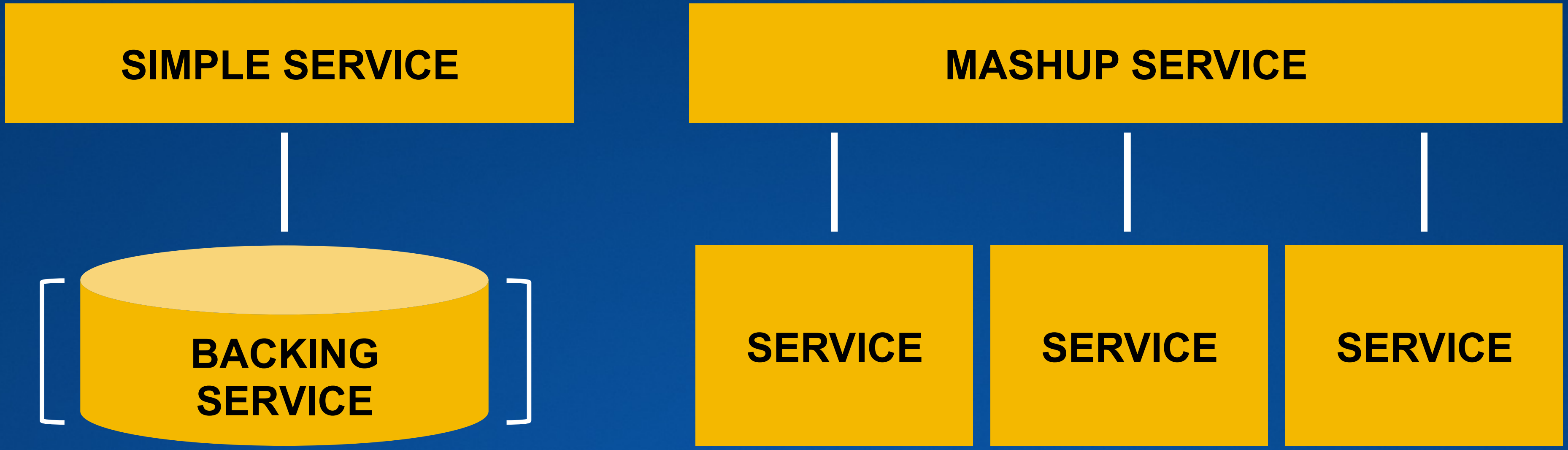
12 Factors

Codebase	Backing Services	Port Binding	Dev/Prod Parity
Dependencies	Build, Release, Run	Concurrency	Logs
Config	Processes	Disposability	Admin Processes

YaaS High-Level



Types of YaaS Microservices



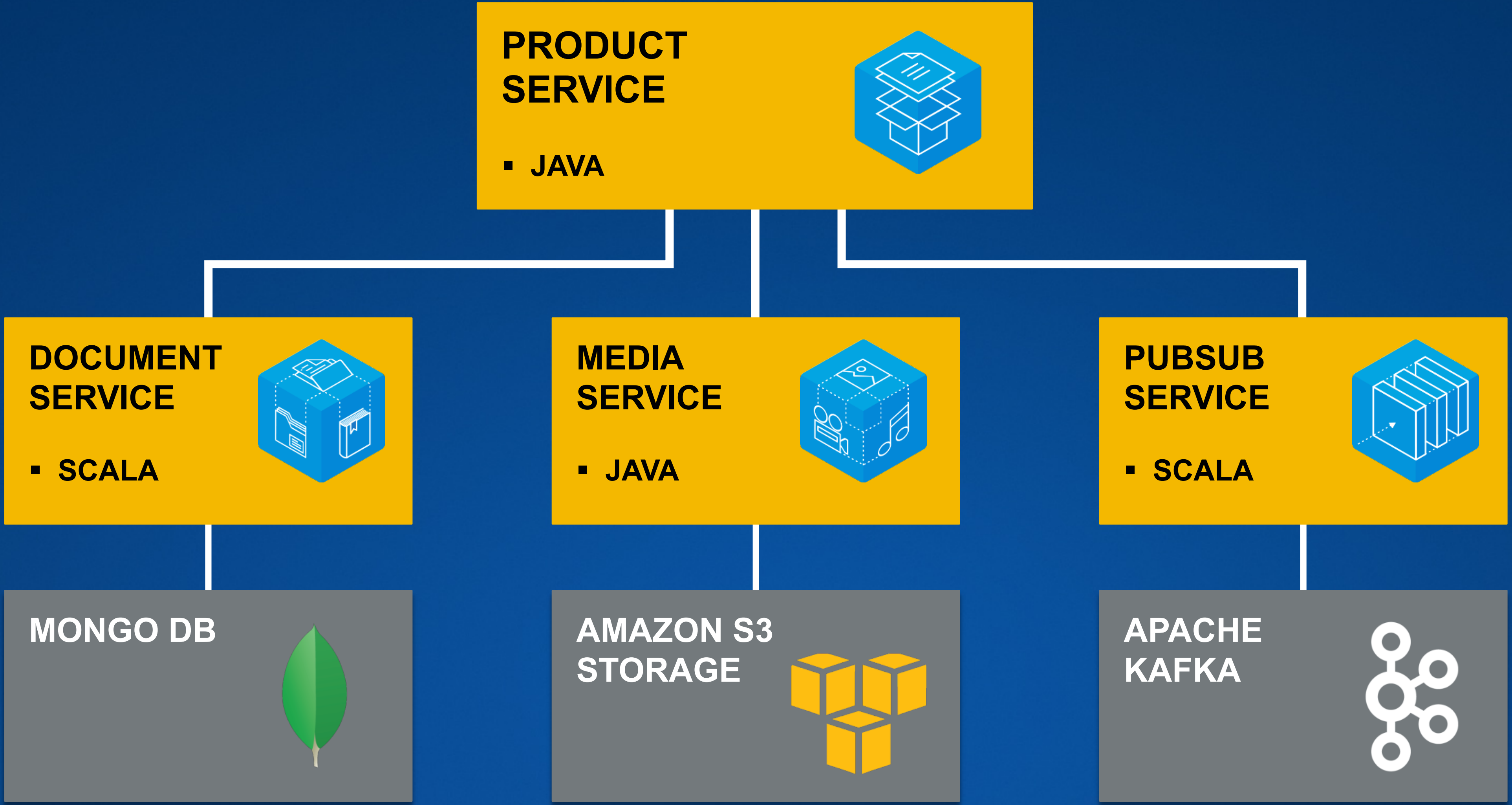
YaaS Product Service

Master-of-record repository for structured product content

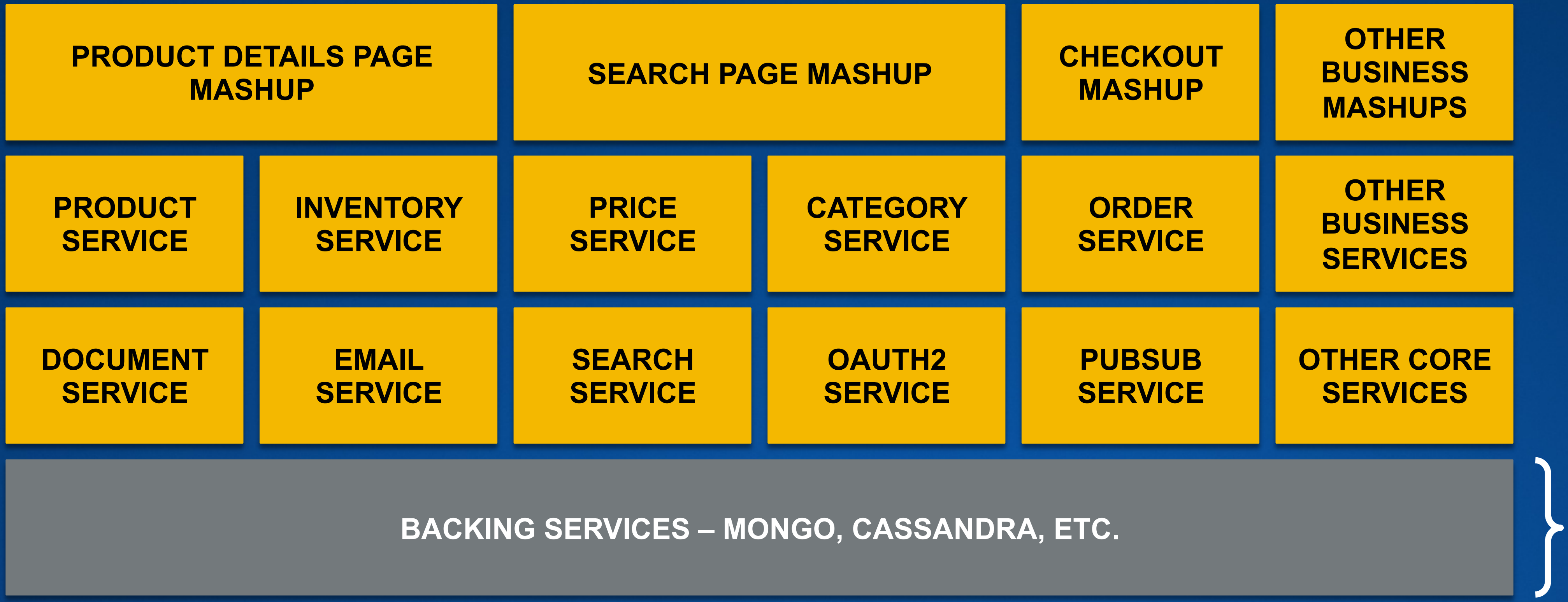
Provides scalable, fast, and easy-to-use back-end for

- Storefronts (web and native apps) serving product content
- Back-office PCM and data editing tools for products
- Systems integration scenarios

Internals of the YaaS Product Service

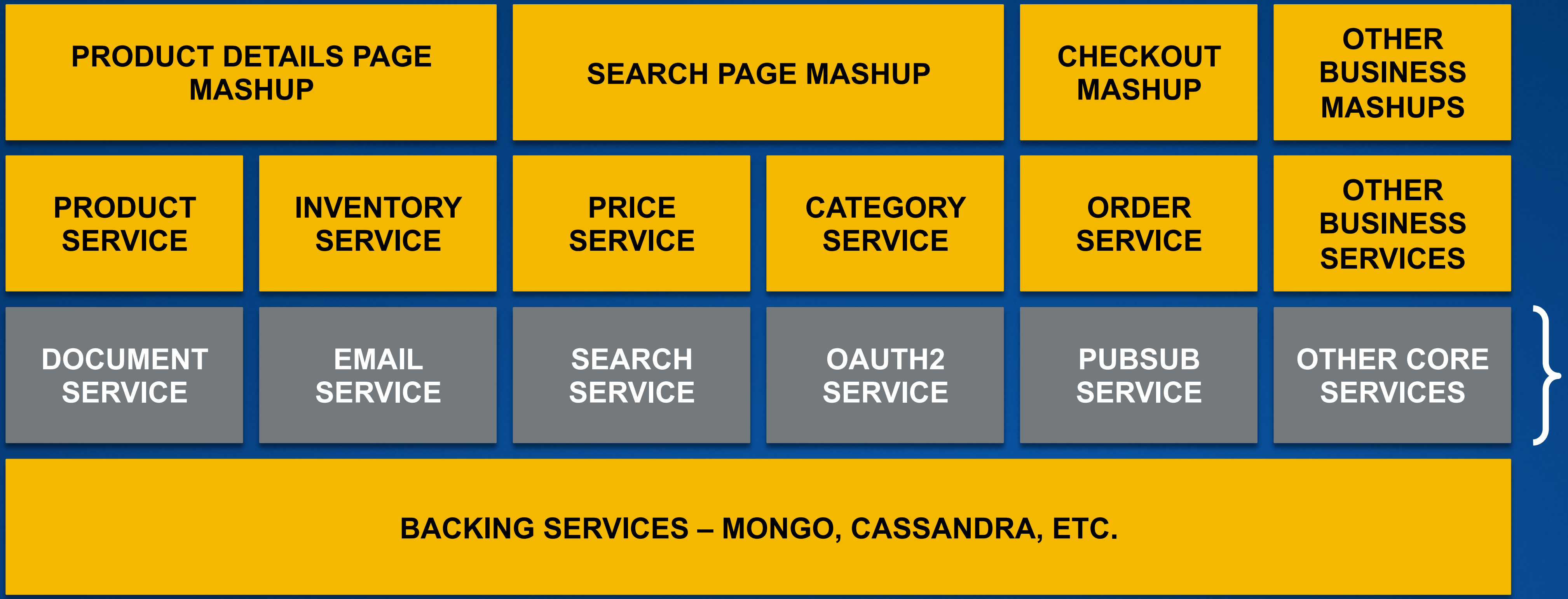


How is the YaaS architecture layered



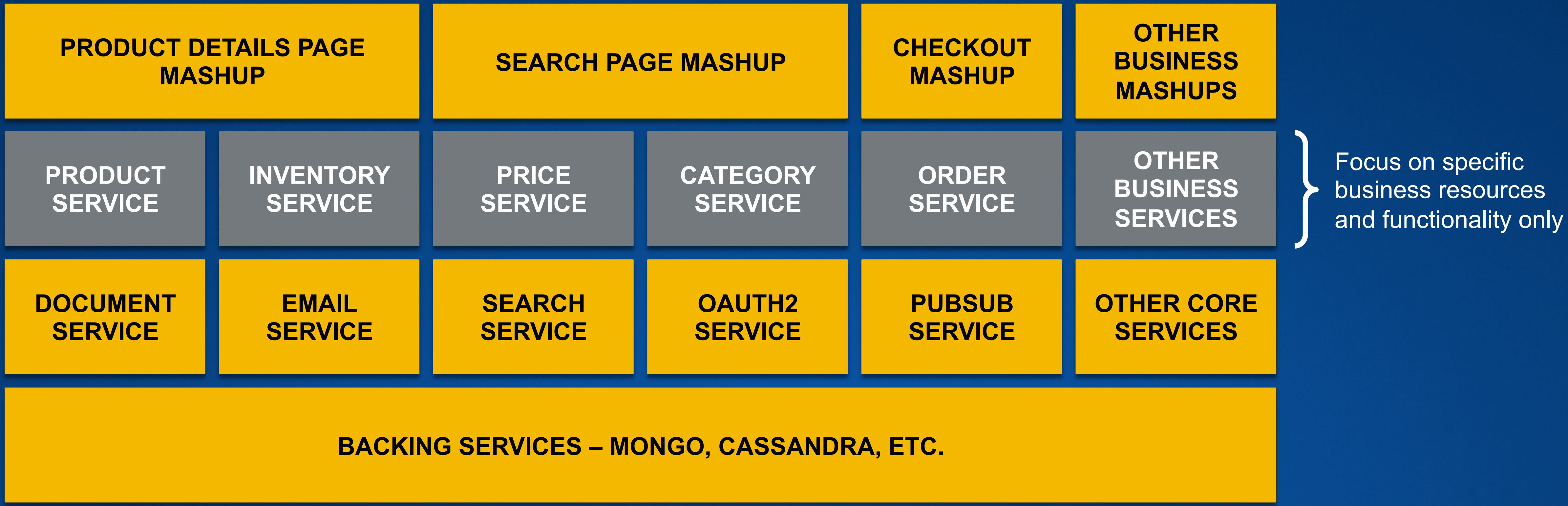
Native applications accessible only by owning teams

How is the YaaS architecture layered

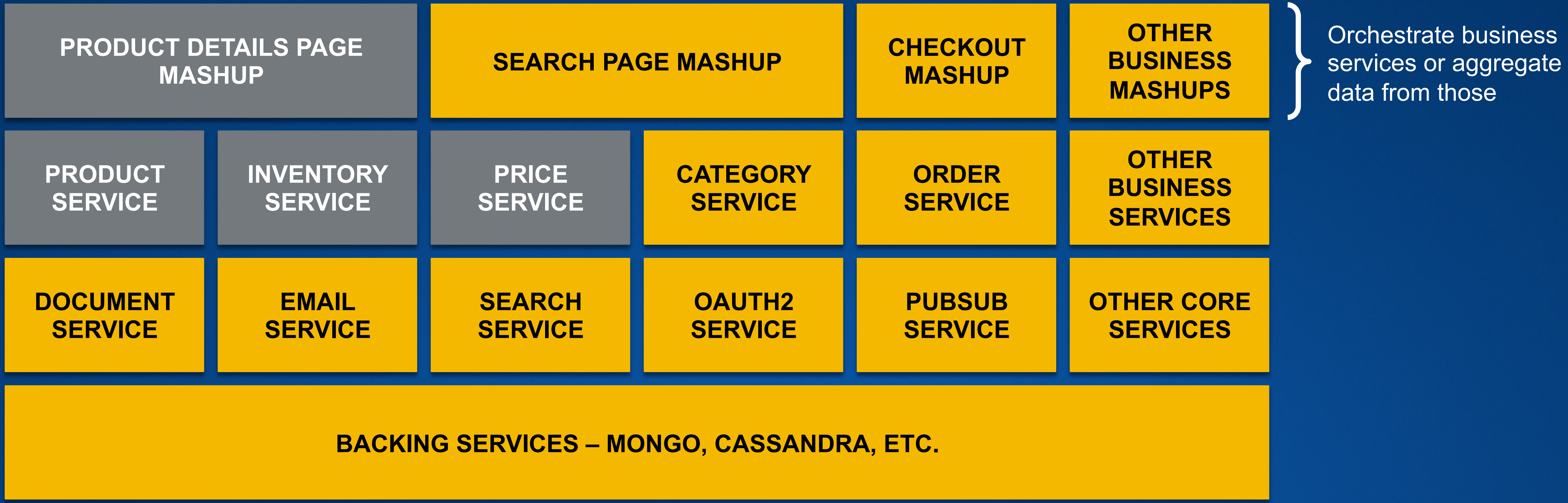


Focus on core functionality. Re-use enables teams to deliver business value rapidly

How is the YaaS architecture layered



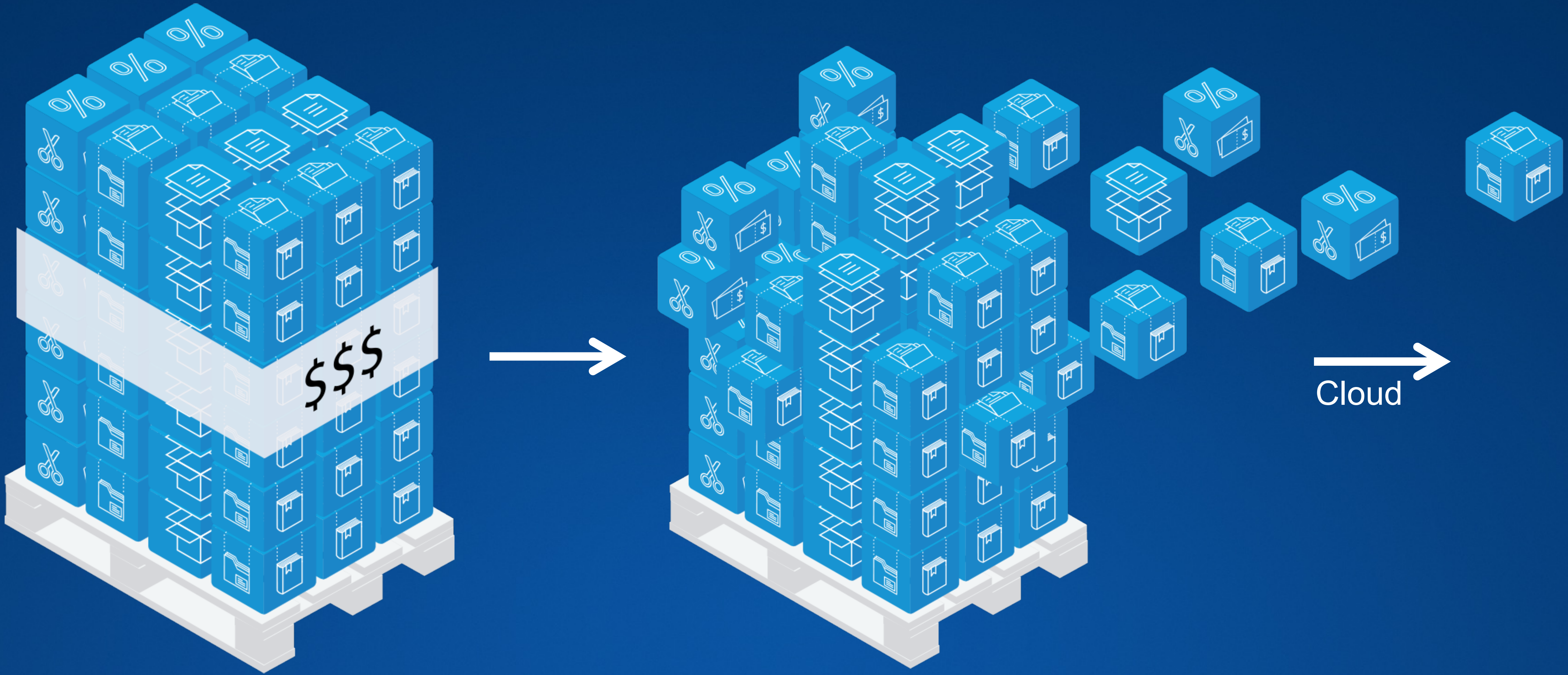
How is the YaaS architecture layered



Constructing with Microservices

- Extending a monolithic application with cloud-based services is nothing new
- Many application functions do not belong in the monolith and it is only force of habit vs. good design which drives a function/feature to be implemented in the monolith
- With the YaaS Market, many services are currently available and the number will continue to grow

Decomposition of monolithic applications



Connected, monolithic, hosted in 1 DC, application

Smaller, distributed into the cloud self-contained digital services

Getting from the Monolith the the Hybrid Model

- **There are many features that are orthogonal to the core domain that can be factored out (…and executed / deployed elsewhere)**
- **We can turn to *Domain Driven Design*, which advocates creating Bounded Contexts around business concepts**
  - Develop a simple shared model (think: API) that communicates the intent and interface of these contexts
  - Internal details can differ from the core platform (models, data store, programming language)
- **Refactor functionality into a package and separate that package into a service**
- **Alternatively, the package can simply by the glue between disparate domains (think: YaaS Services)**

No Free Lunch

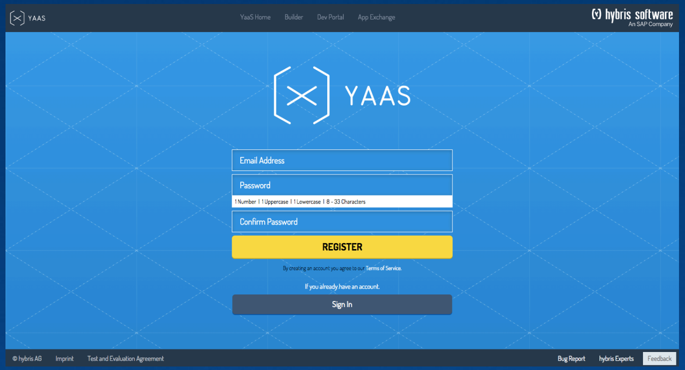
Microservices provide benefits…	…but come with costs
Strong Module Boundaries: Microservices reinforce modular structure, which is particularly important for larger teams.	**Distribution**: Distributed systems are harder to program, since remote calls are slow and are always at risk of failure.
Independent Deployment: Simple services are easier to deploy, and since they are autonomous, are less likely to cause system failures when they go wrong.	**Eventual Consistency**: Maintaining strong consistency is extremely difficult for a distributed system, which means everyone has to manage eventual consistency.
Technology Diversity: With microservices you can mix multiple languages, development frameworks and data-storage technologies.	**Operational Complexity**: You need a mature operations team to manage lots of services, which are being redeployed regularly.

http://martinfowler.com/articles/microservice-trade-offs.html July, 2015

Get Started With YaaS

Build Something Amazing

Register



Email Address

Password

1 Number | 1 Uppercase | 1 Lowercase | 8 - 33 Characters

Confirm Password

REGISTER

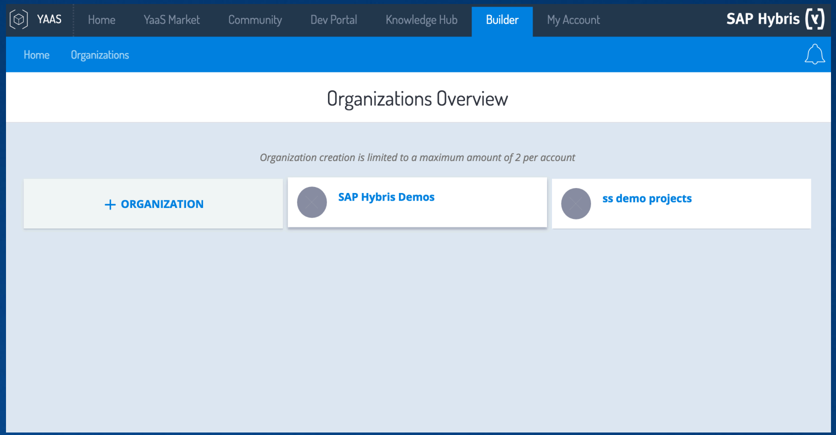
By creating an account you agree to our Terms of Service.

If you already have an account.

Sign In

© hybris AG   Imprint   Test and Evaluation Agreement   Bug Report   hybris Experts   Feedback

Create Organization

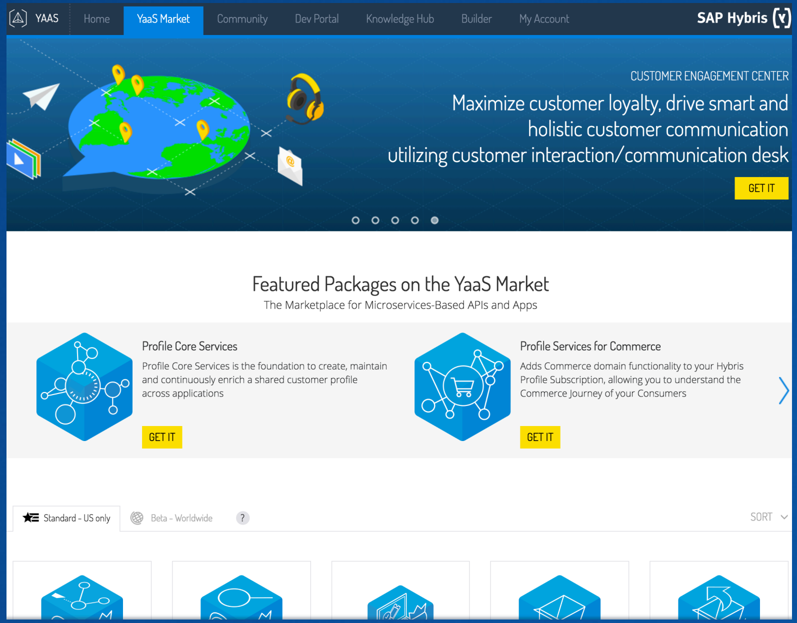


Organizations Overview

Organization creation is limited to a maximum amount of 2 per account

+ ORGANIZATION | SAP Hybris Demos | ss demo projects

Subscribe to Packages



CUSTOMER ENGAGEMENT CENTER

Maximize customer loyalty, drive smart and holistic customer communication utilizing customer interaction/communication desk

GET IT

Featured Packages on the YaaS Market

The Marketplace for Microservices-Based APIs and Apps

Profile Core Services

Profile Core Services is the foundation to create, maintain and continuously enrich a shared customer profile across applications

GET IT

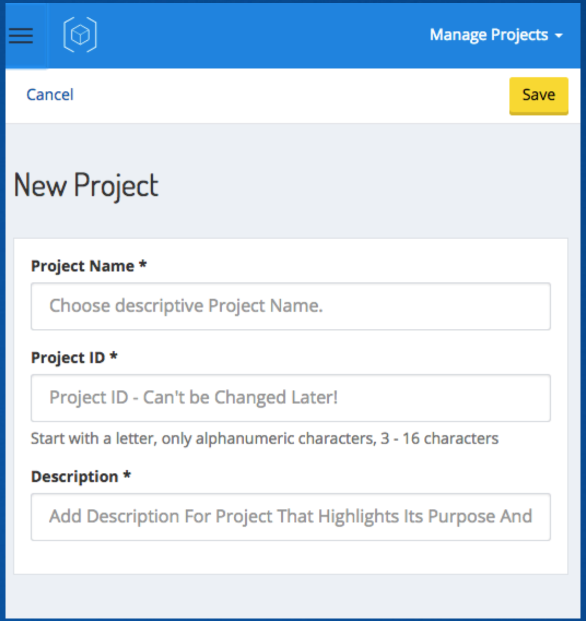
Profile Services for Commerce

Adds Commerce domain functionality to your Hybris Profile Subscription, allowing you to understand the Commerce Journey of your Consumers

GET IT

Standard - US only   Beta - Worldwide   ?   SORT

Create Project



New Project

Project Name *

Choose descriptive Project Name.

Project ID *

Project ID - Can't be Changed Later!

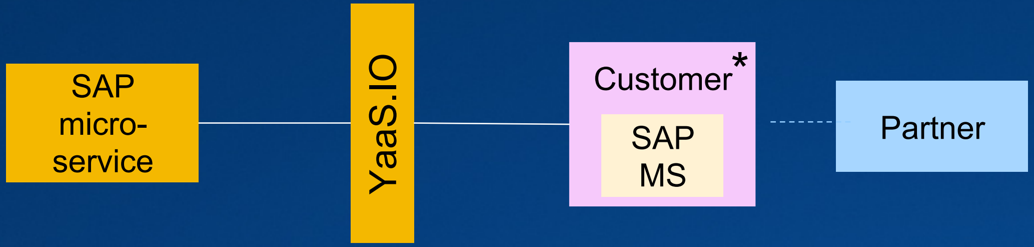
Start with a letter, only alphanumeric characters, 3 - 16 characters

Description *

Add Description For Project That Highlights Its Purpose And

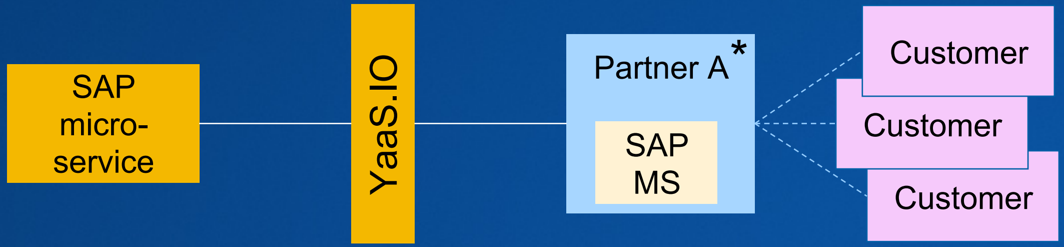
Consumption Scenarios (available in US)

Partner Business Opportunities



- Customer subscribes
- Partner can work on Customer subscription once Customer creates Authorized User for Partner
- ***Implementation, Integration & Migration Services***

- ***Transformation & Advisory Services***



- Partner builds own Cloud offering including SAP packaged services and microservices subscriptions
- Partner contracts with Customer for their Cloud offering
- Customer can work on Partner subscription (no contractual relationship with SAP)
- ***IP monetization through Cloud offering***
- ***Reselling***

* Ideal Cloud PaaS & development environment:
Hana Cloud Platform, separate contract

YaaS Resources

- https://www.yaas.io/
- https://knowledge.yaas.io/
- https://devportal.yaas.io/gettingstarted/
- (Clojurescript sample client) https://github.com/samcschneider/yaas-essentials

Example Business Case

- The customer has some static product information, but no ability to manage it except through a CMS as content
- The customer wishes to have maintainable product content management
- Phase two involves commerce and the solution implemented should allow for a purchase process
- The rollout should be phased and evolve their existing web property vs. a big bang replacement

START INNOVATING TODAY!

Sign up at:

www.yaas.io