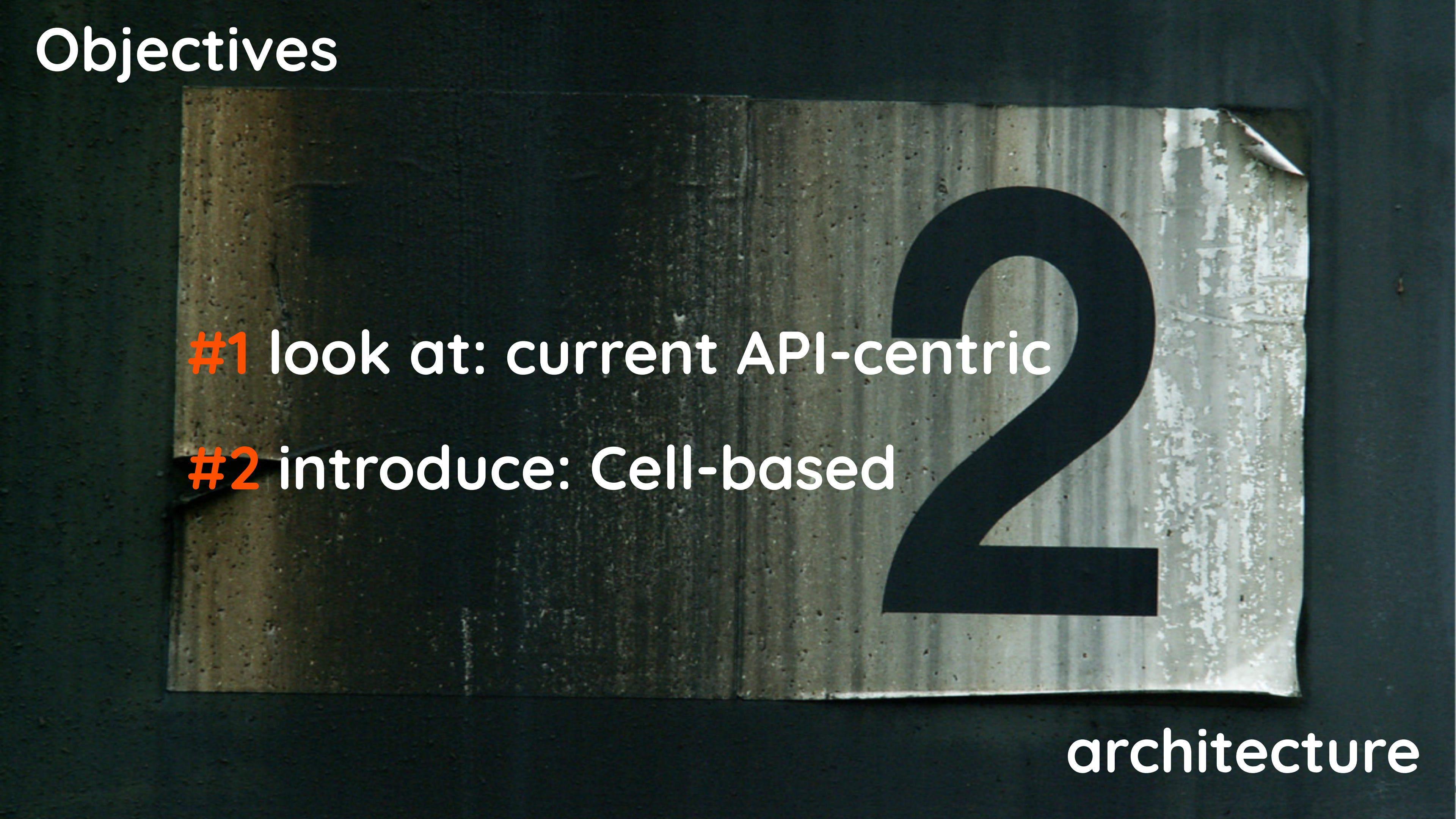
Cell-based Architecture

An Emerging Architecture Pattern for Agile Integration

Asanka Abeysinghe
Deputy CTO & VP, Architecture - CTO Office
WSO2 Inc.

Objectives



#1 look at: current API-centric

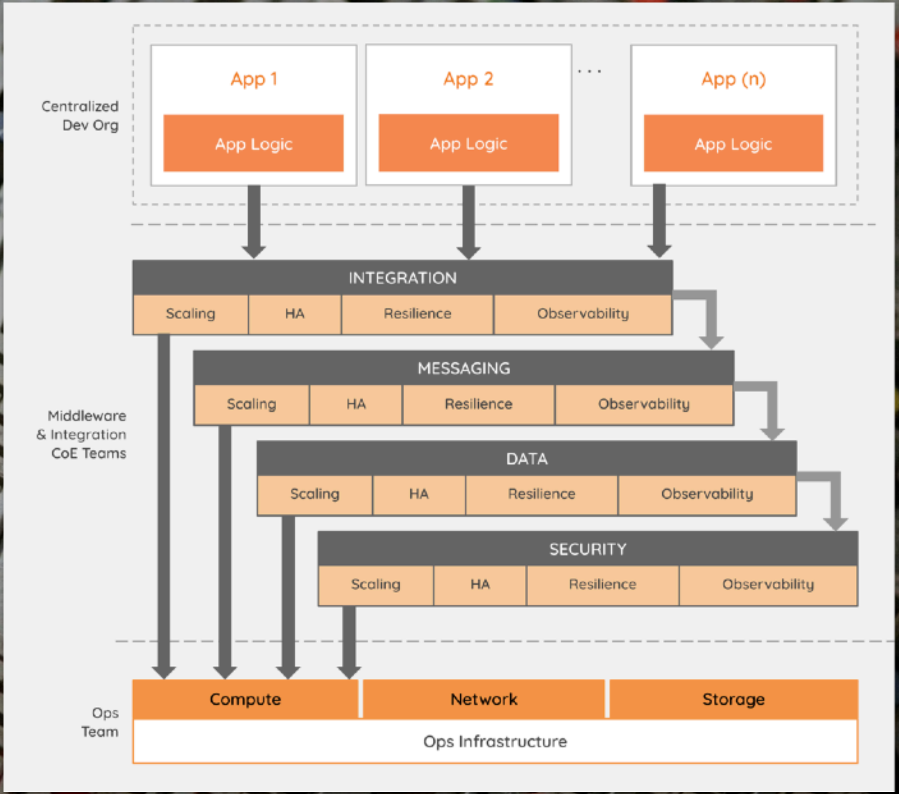
#2 introduce: Cell-based

architecture

picture credit: https://www.flickr.com/

Motivation

Centralized & Layered



picture credit: https://www.flickr.com/photos/kaktuslampa/

Powerpoint Architects

picture credit: https://www.flickr.com/photos/69929929@N06/

Reality of the Enterprise

picture credit: https://www.flickr.com/photos/johnerlandsen/

Brownfield > Greenfield

picture credit: https://www.apstra.com/home/landscape-of-meadow-field-with-the-changing-environment-2/

Reference Implementations

picture credit: https://www.marinersmuseum.org/

Underutilization of the Technology

picture credit: http://unlocked.footlocker.com/

Gap: architecture | development | deployment

picture credit: https://www.flickr.com/photos/28994379@N02/

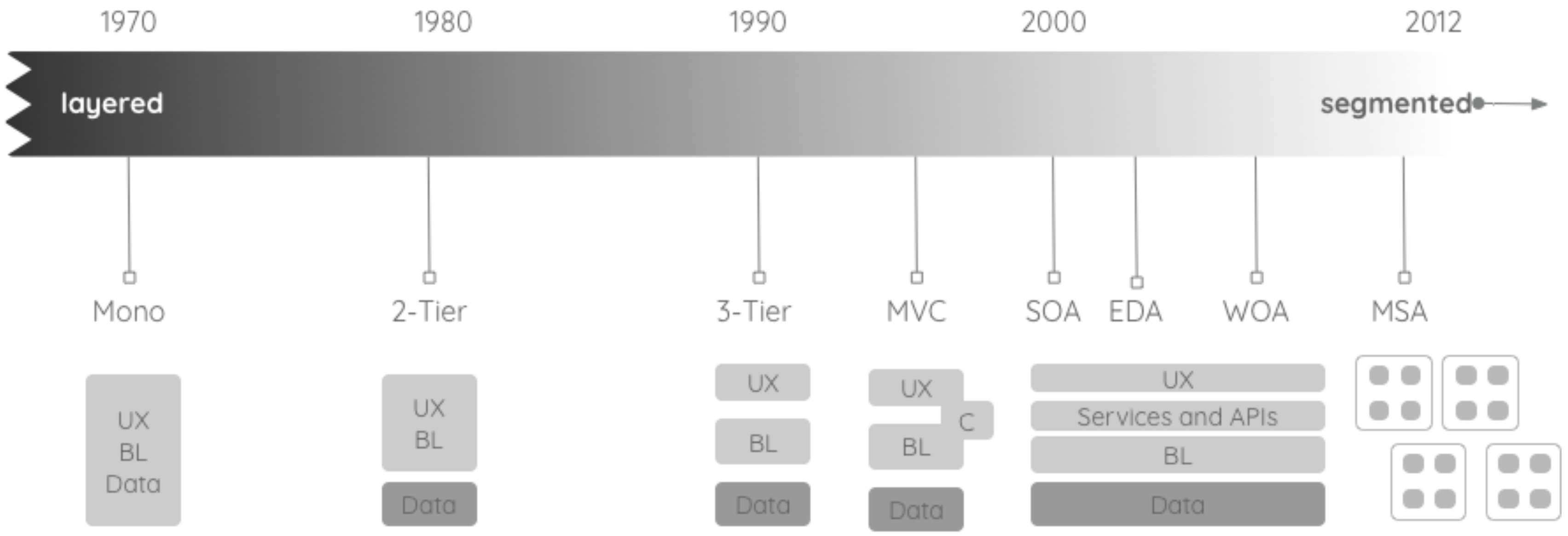
Dependency management

picture credit: https://www.solarpowerworldonline.com/2017/07/georgia-contractor-sells-farmers-solar-tough-market/

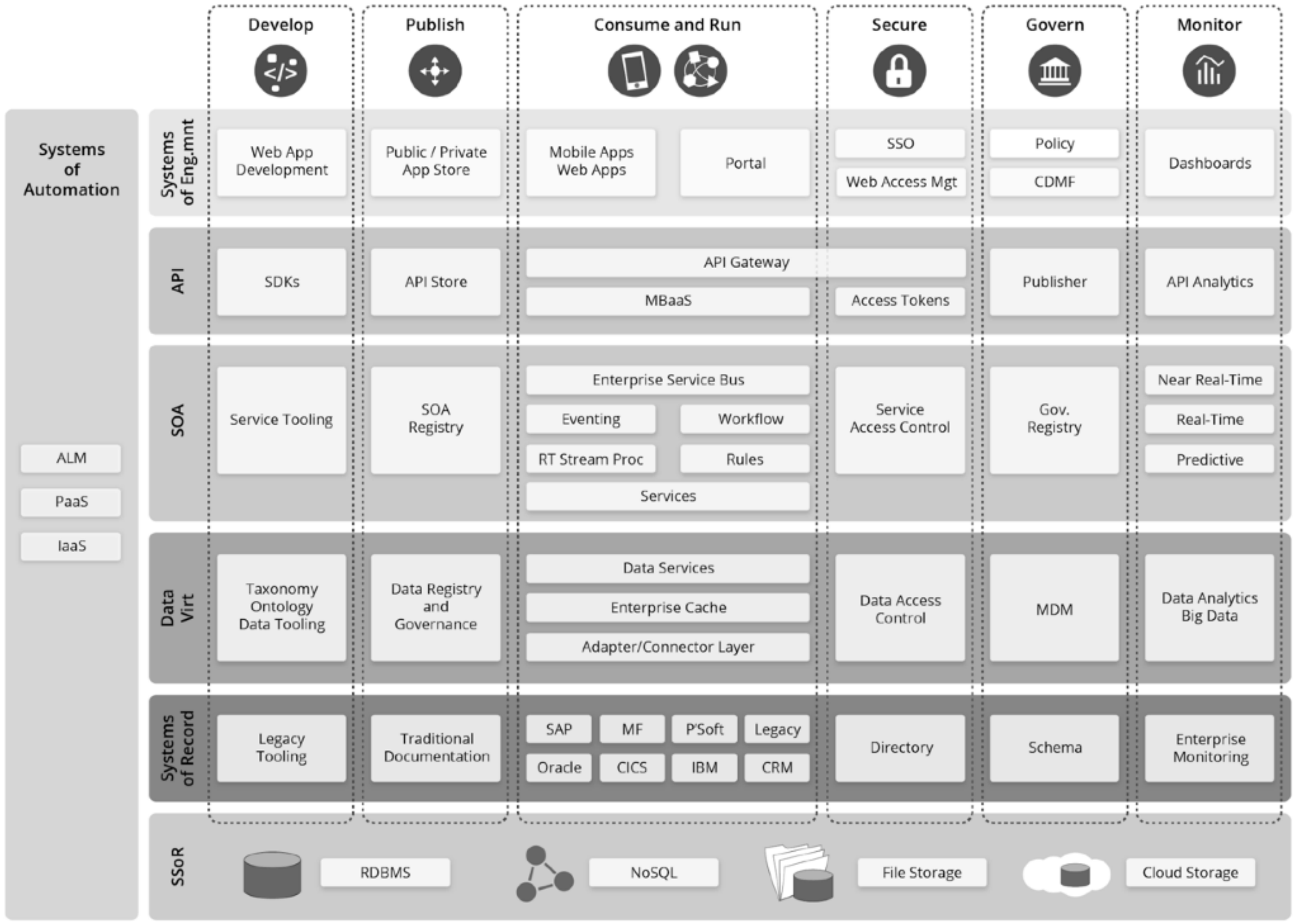
Architecture Patterns

picture credit: https://www.flickr.com/photos/tiocfaidh_ar_la_1916/

Timeline


1970
1980
1990
2000
2012
layered
segmented
Mono
2-Tier
3-Tier
MVC
SOA
EDA
WOA
MSA
UX
BL
Data
UX
BL
Data
UX
BL
Data
UX
C
BL
Data
UX
Services and APIs
BL
Data

Background: Layered Architecture



A platform with an agile team

100 APIs, 60 message flows, 80 services, n DBs

Multi-tenanted, 3 active tenants

First release after 3 years

picture credit: https://www.flickr.com/photos/23119666@N03/

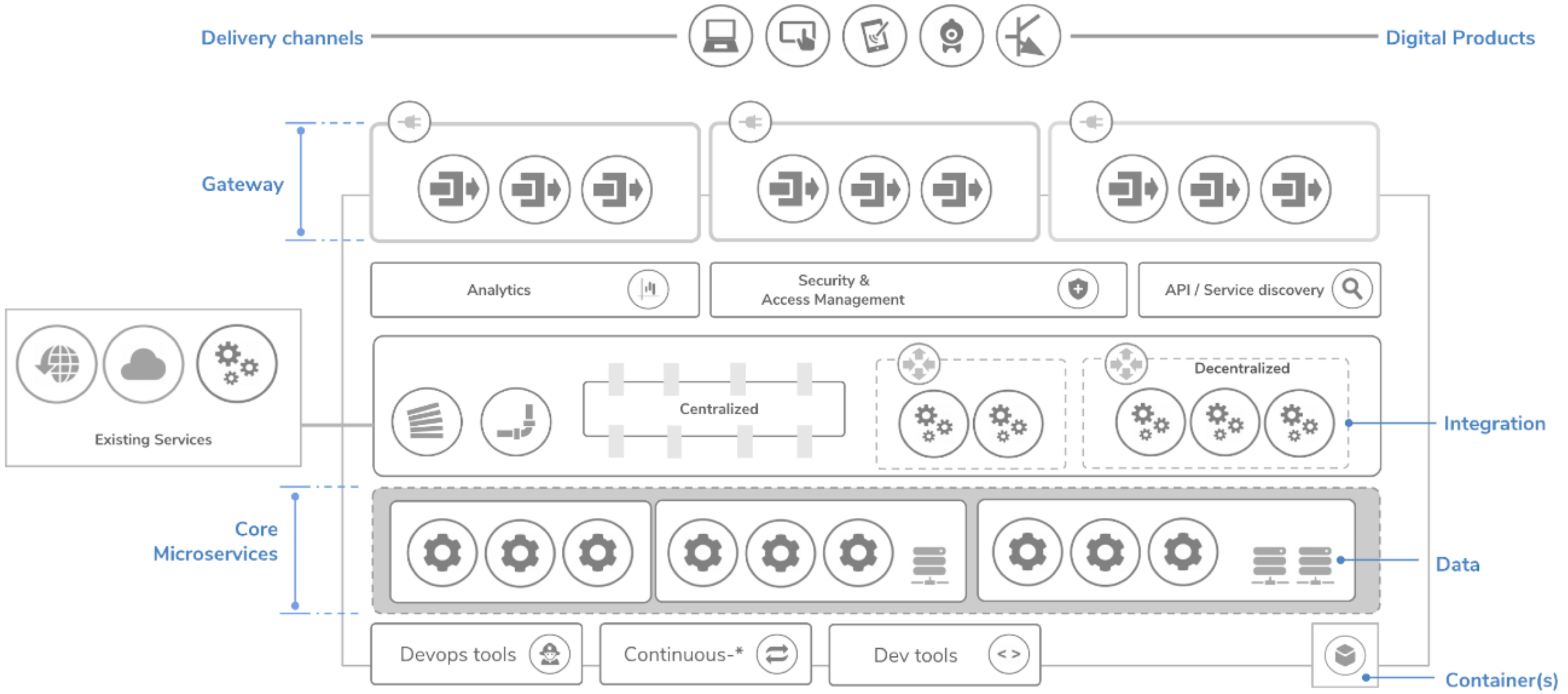
Rise of Microservices

picture credit: https://www.flickr.com/photos/imcountingufoz/

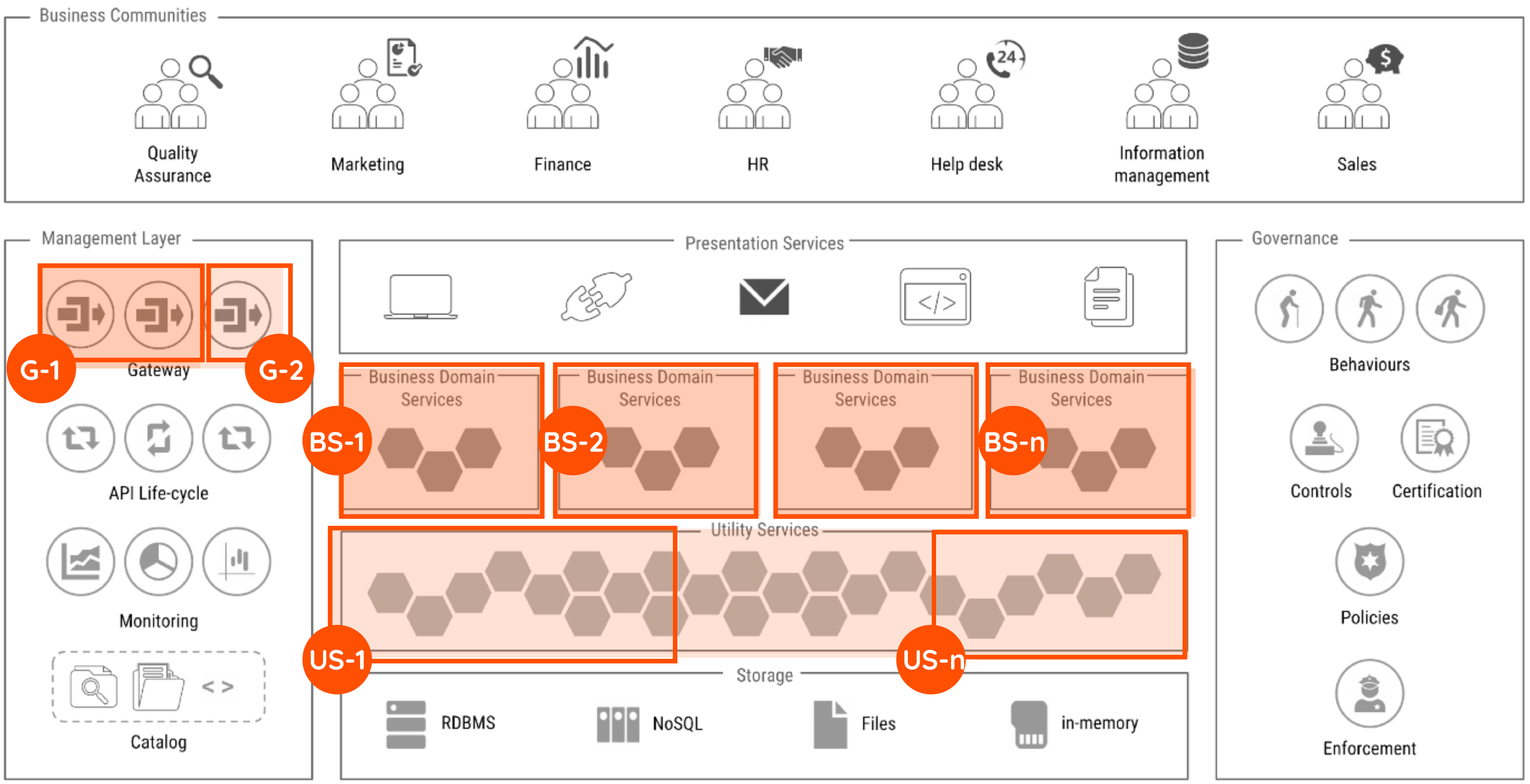
Pragmatic Microservices

- Netflix: APIs
- Uber: Edge Gateway
- eBay: API Facade
- Gartner: Mini Services

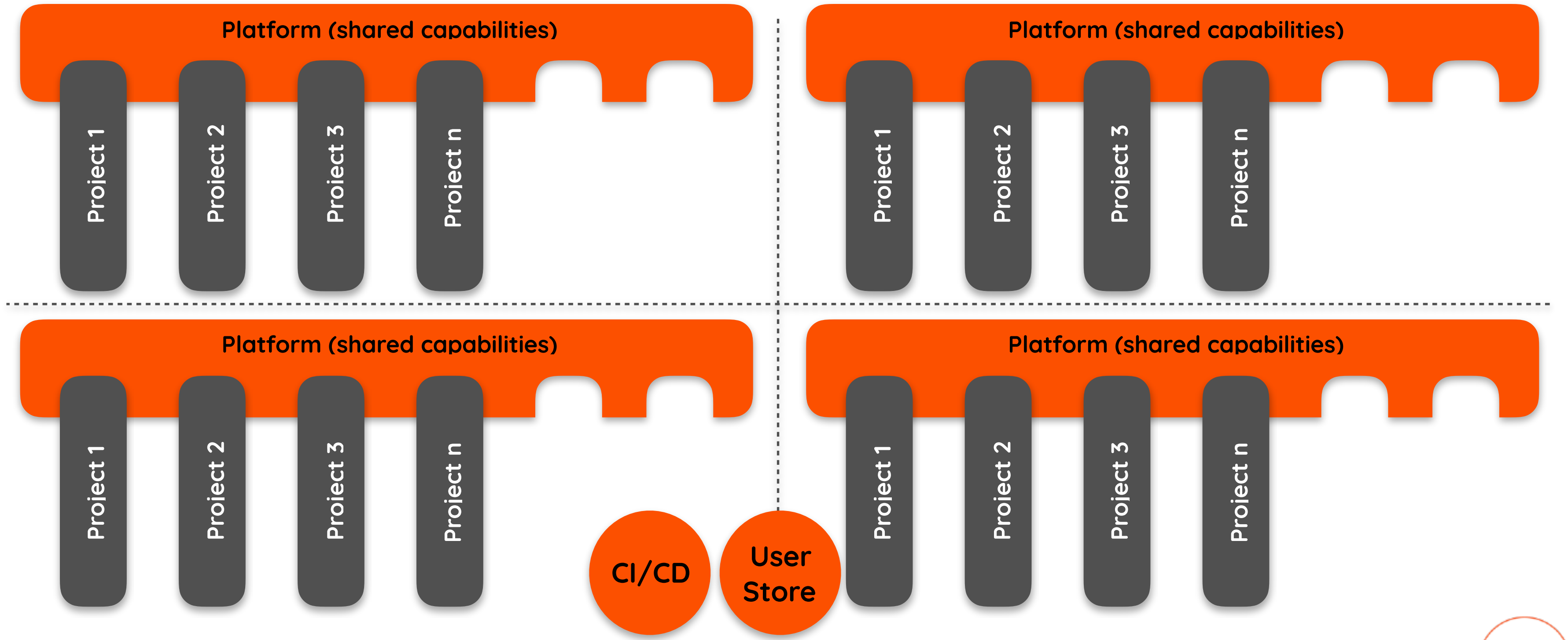
Background: Layered Architecture with MSA



Background: Segmented Architecture



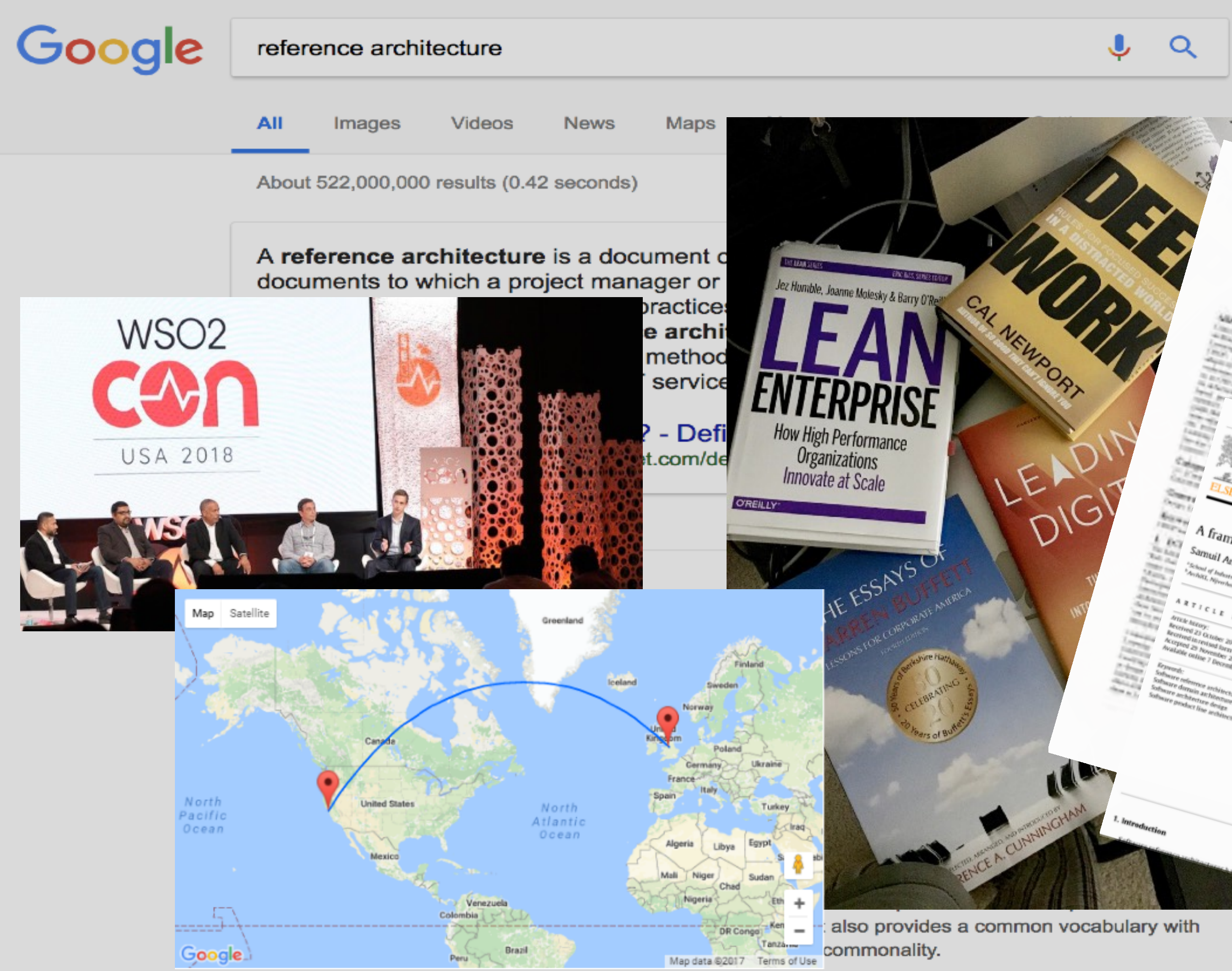
Platform of Platforms



picture credit: https://www.flickr.com/photos/ableman/

Making of......

picture credit: https://commons.wikimedia.org/wiki/User:Frank_Schulenburg CC BY 3.0



reference architecture

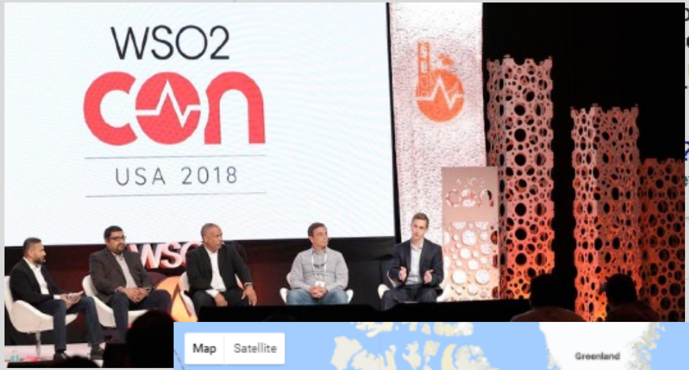
All Images Videos News Maps

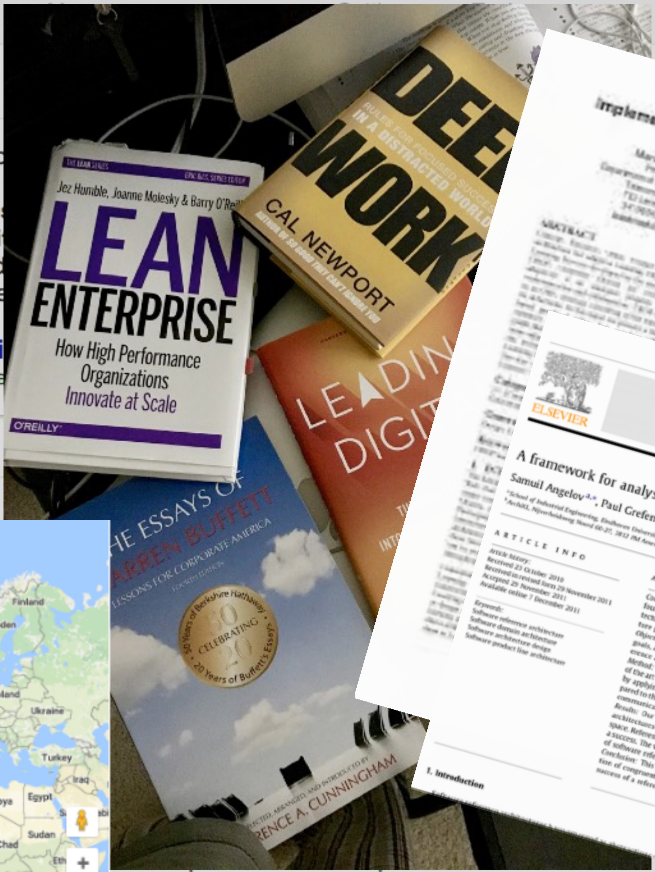
About 522,000,000 results (0.42 seconds)

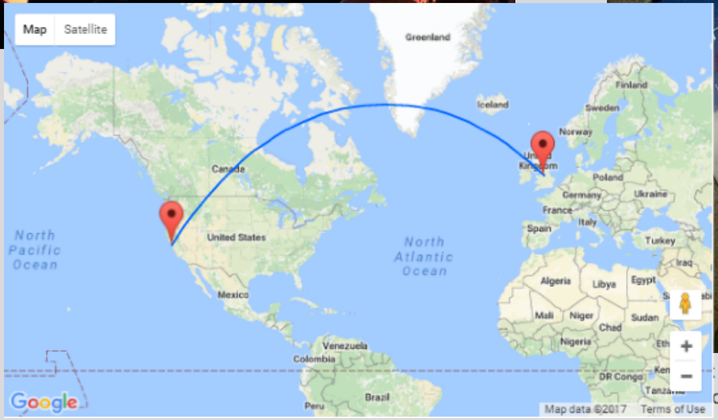
A **reference architecture** is a document o
documents to which a project manager or
practice
e archi
method
service

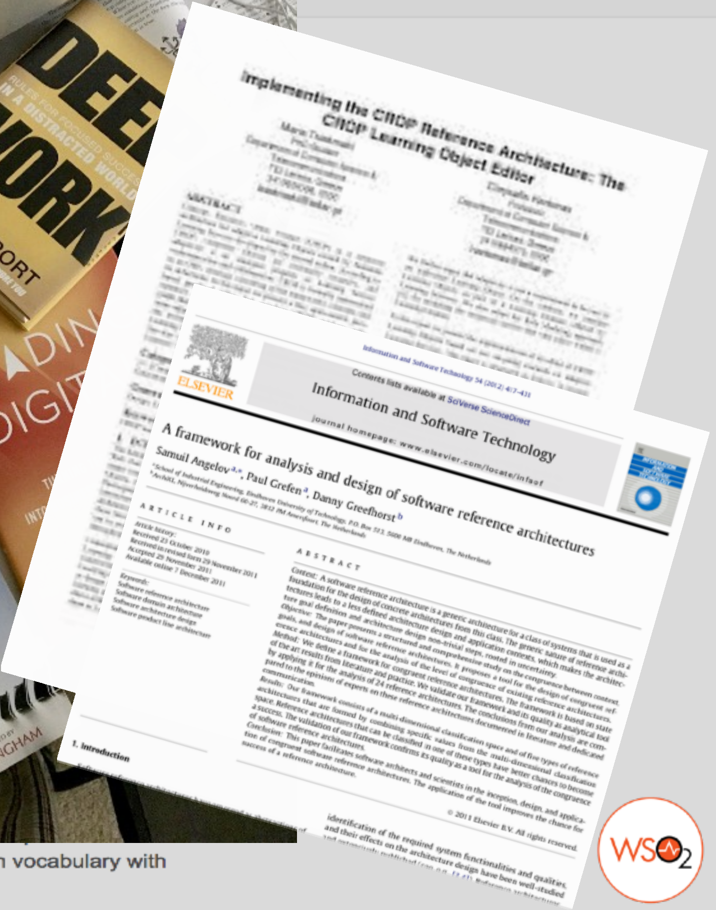
? - Defi
t.com/de

also provides a common vocabulary with
commonality.




DEEP WORK
RULES FOR FOCUSED SUCCESS IN A DISTRACTED WORLD
CAL NEWPORT
Jez Humble, Joanne Molesky & Barry O'Reilly
LEAN ENTERPRISE
How High Performance Organizations Innovate at Scale
O'REILLY
LEADING DIGITAL
THE ESSAYS OF WARREN BUFFETT
LESSONS FOR CORPORATE AMERICA
CELEBRATING 20 Years of Buffett's Essays
LAWRENCE A. CUNNINGHAM





Implementing the CROP Reference Architecture: The CROP Learning Object Editor

Information and Software Technology 54 (2012) 417–431

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

A framework for analysis and design of software reference architectures

Samuil Angelov[a,*], Paul Grefen[a], Danny Greefhorst[b]

[a] School of Industrial Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[b] ArchiXL, Nijverheidsweg Noord 60-27, 3812 PM Amersfoort, The Netherlands

ARTICLE INFO

Article history:
Received 23 October 2010
Received in revised form 29 November 2011
Accepted 29 November 2011
Available online 7 December 2011

Keywords:
Software reference architecture
Software domain architecture
Software architecture design
Software product line architecture

ABSTRACT

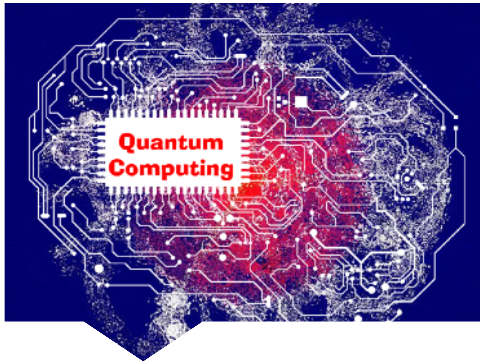
Context: A software reference architecture is a generic architecture for a class of systems that is used as a foundation for the design of concrete architectures from this class. The generic nature of reference architectures leads to a less defined architecture design and application contexts, which makes the architecture goal definition and architecture design non-trivial steps, rooted in uncertainty.
Objective: The paper presents a structured and comprehensive study on the congruence between context, goals, and design of software reference architectures. It proposes a tool for the design of congruent reference architectures and for the analysis of the level of congruence of existing reference architectures.
Method: We define a framework for congruent reference architectures. The framework is based on state of the art results from literature and practice. We validate our framework and its quality as analytical tool by applying it for the analysis of 24 reference architectures. The conclusions from our analysis are compared to the opinions of experts on these reference architectures documented in literature and dedicated communication.
Results: Our framework consists of a multi-dimensional classification space and of five types of reference architectures that are formed by combining specific values from the multi-dimensional classification space. Reference architectures that can be classified in one of these types have better chances to become a success. The validation of our framework confirms its quality as a tool for the analysis of the congruence of software reference architectures.
Conclusion: This paper facilitates software architects and scientists in the inception, design, and application of congruent software reference architectures. The application of the tool improves the chance for success of a reference architecture.

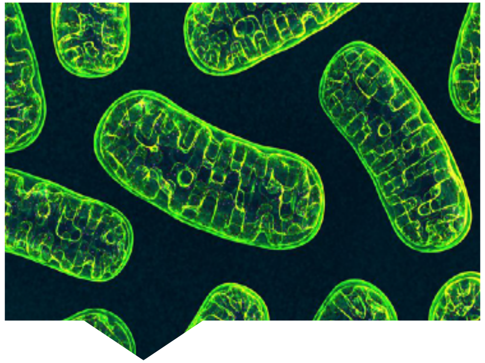
© 2011 Elsevier B.V. All rights reserved.

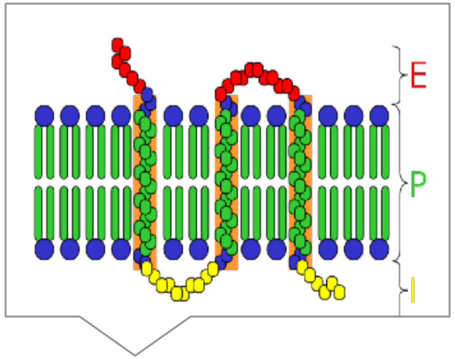
1. Introduction

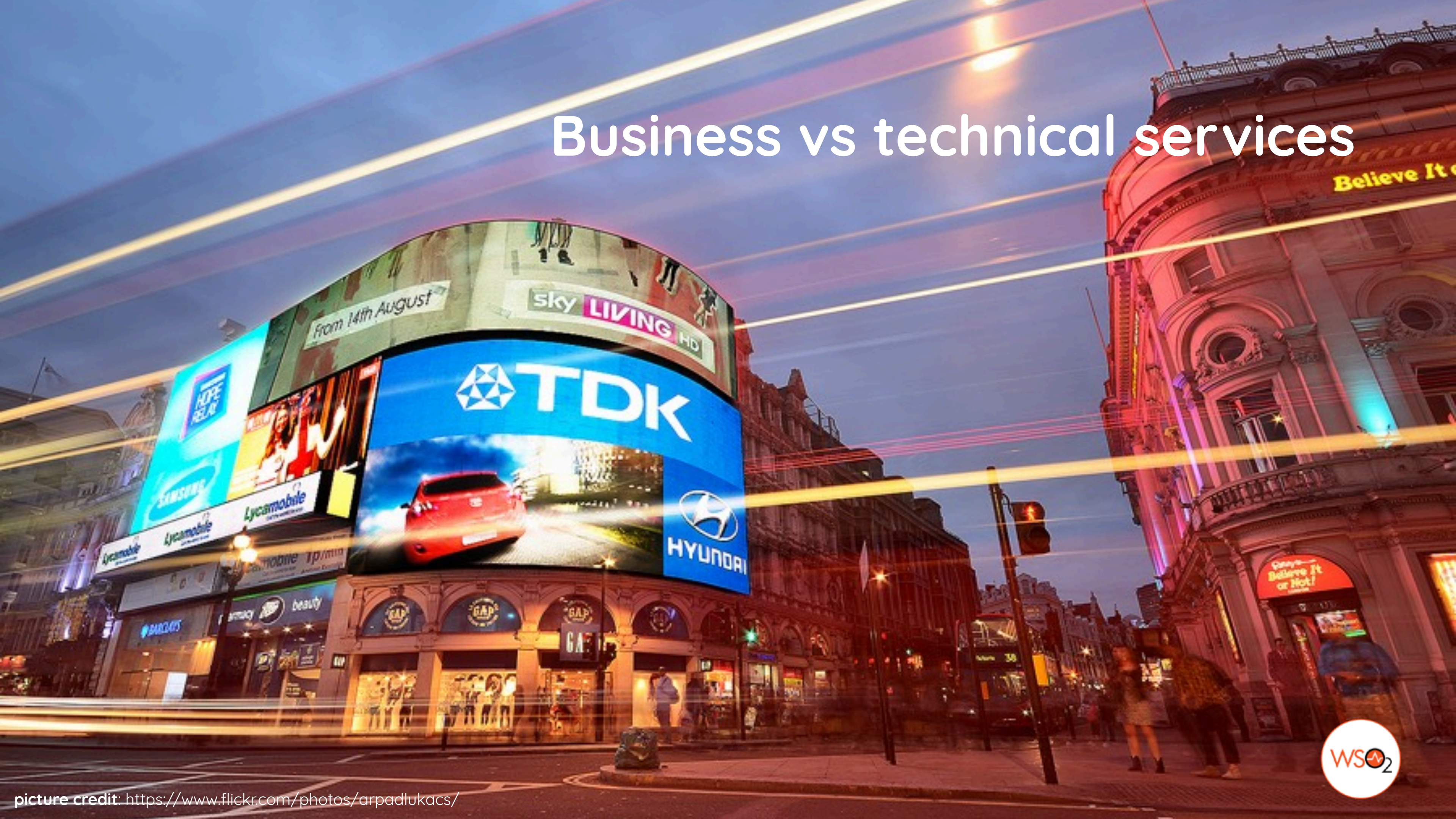
identification of the required system functionalities and qualities, and their effects on the architecture design have been well-studied

Building the Concept







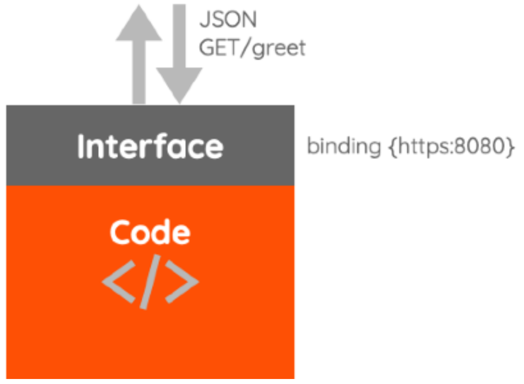


Business vs technical services

picture credit: https://www.flickr.com/photos/arpadlukacs/

Service: Technical definition

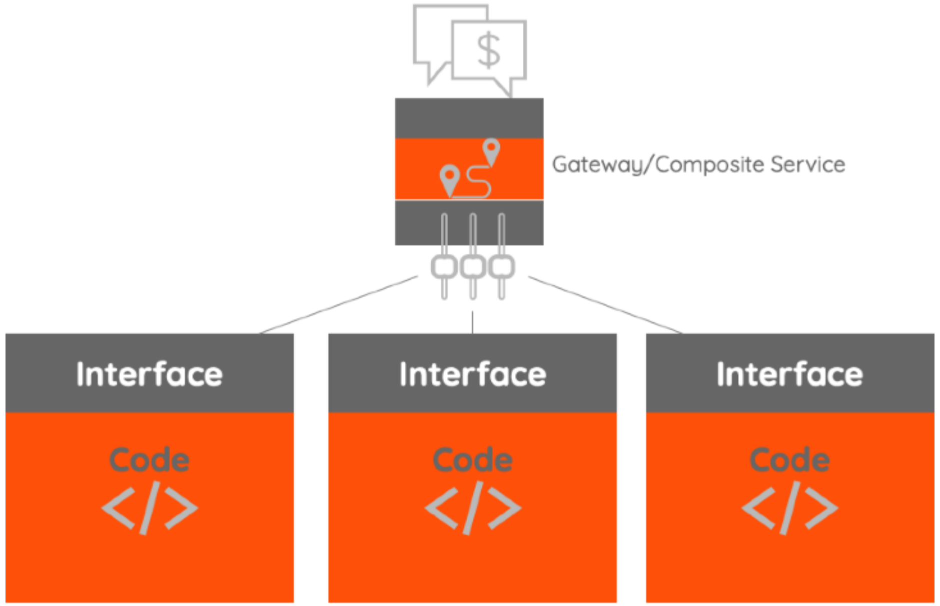
A **code** exposes through an **interface** that describes a collection of operations that are **network accessible** using a standardized messaging protocol.



```
public class PersonApi {

    @Path("/greet")
    @GET
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
     public JSONObject sayHello() {
       try {
          return new JSONObject().put("greeting", "Hello worl
       } catch (JSONException e) {
          return null;
       }

     }

}
```

Service: Business definition

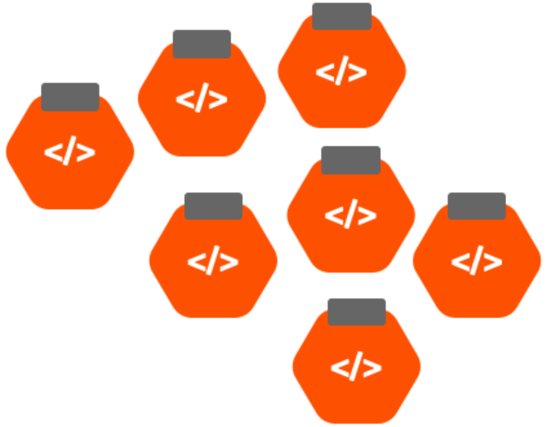
Software components that can be spontaneously discovered, **combined**, and **recombined** to provide a solution to a **business problem**.

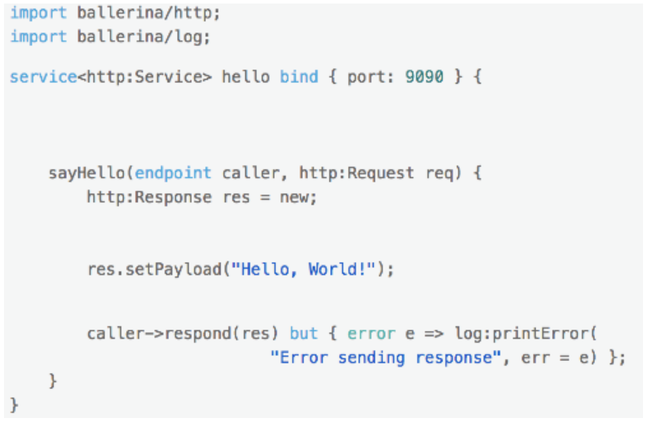


Microservice: Technical definition

A microservice must have a **single purpose** and be loosely coupled in design and deployed independently of other microservices.

"Micro" is a concept of **scope** rather than **size**.



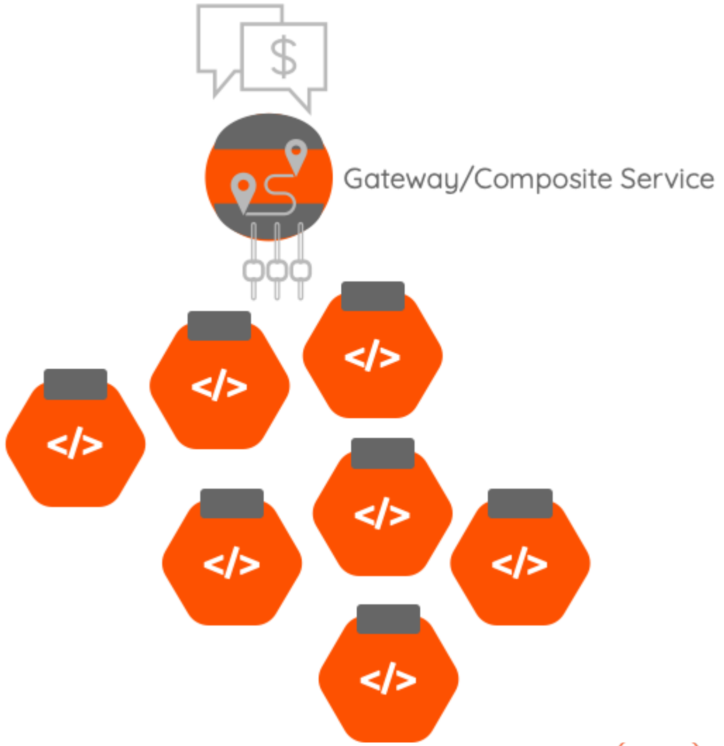


```
import ballerina/http;
import ballerina/log;

service<http:Service> hello bind { port: 9090 } {


    sayHello(endpoint caller, http:Request req) {
        http:Response res = new;


        res.setPayload("Hello, World!");


        caller->respond(res) but { error e => log:printError(
                                 "Error sending response", err = e) };
    }
}
```

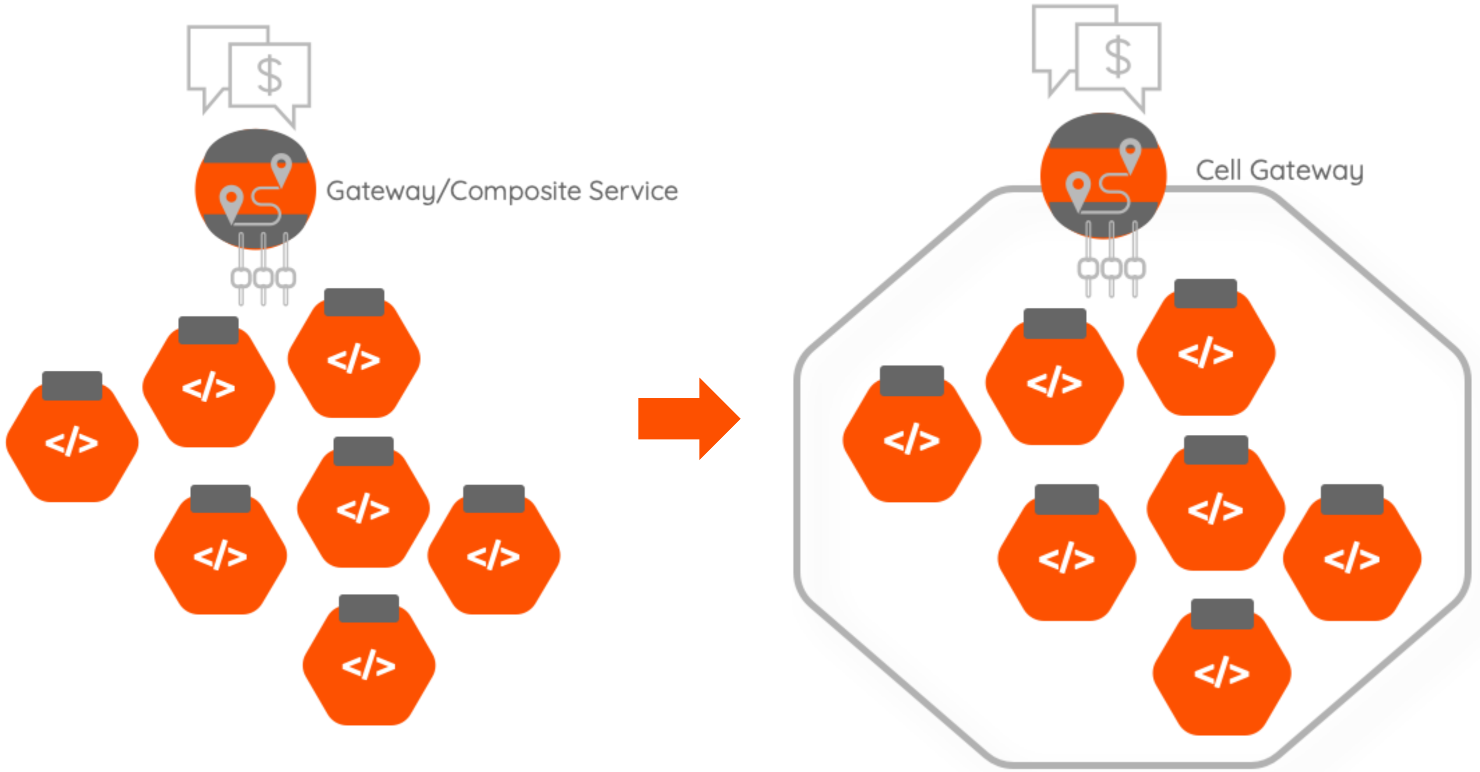
Microservice: Business definition

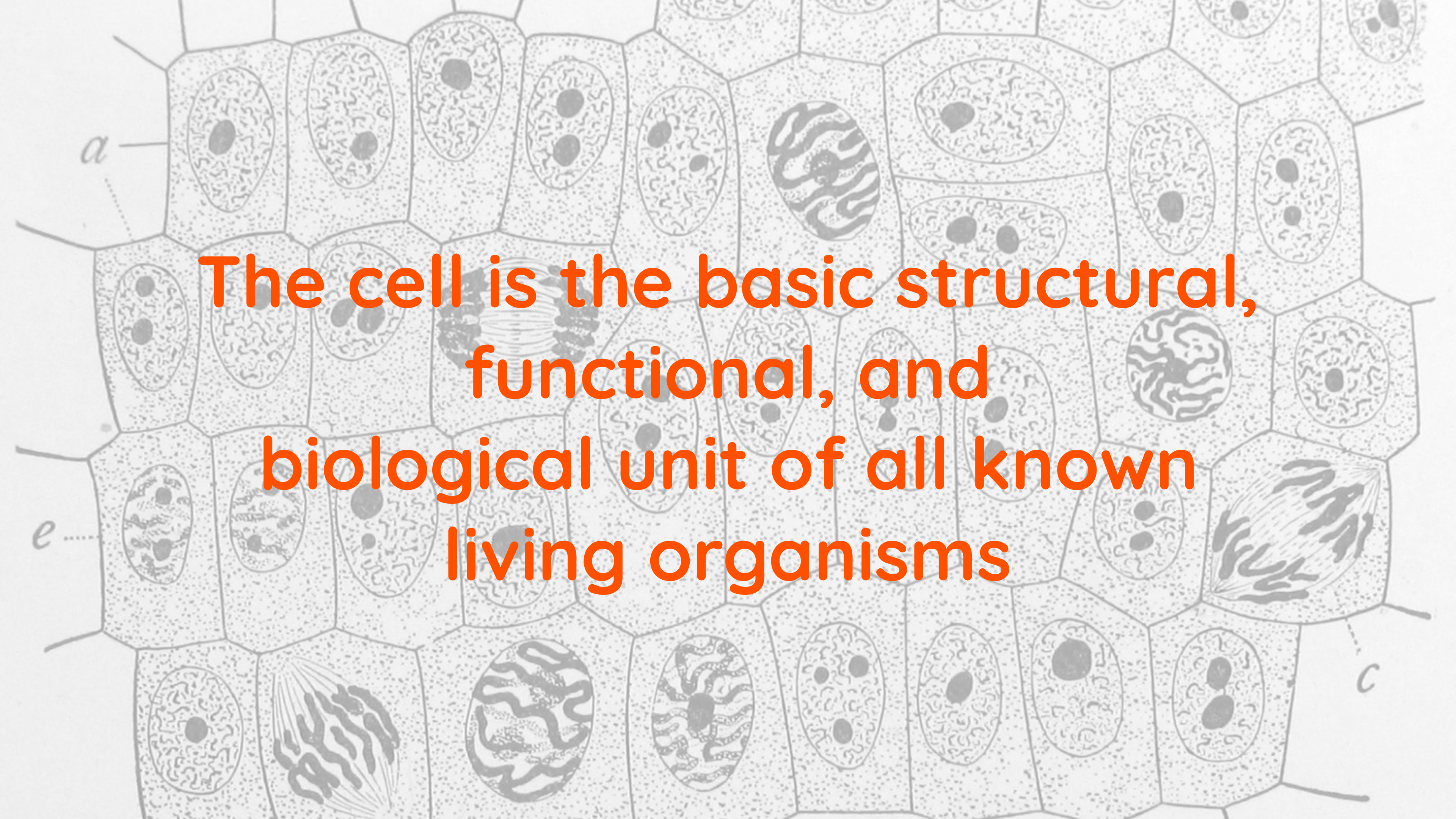
Microservices is an approach to application development in which a large application is built as a suite of **modular components** or services.

These services are built around **business capabilities**.



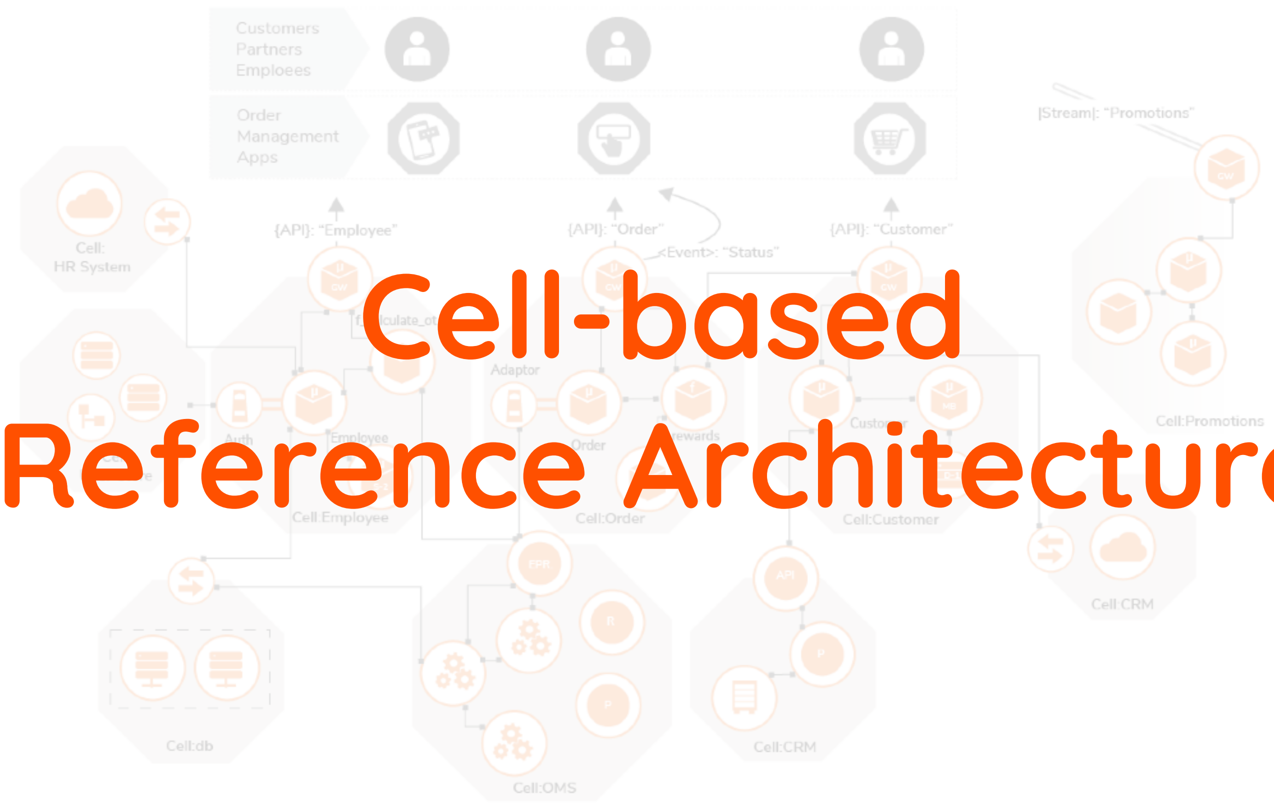
Group of (Micro)services





The cell is the basic structural, functional, and biological unit of all known living organisms

Cell-based Reference Architecture

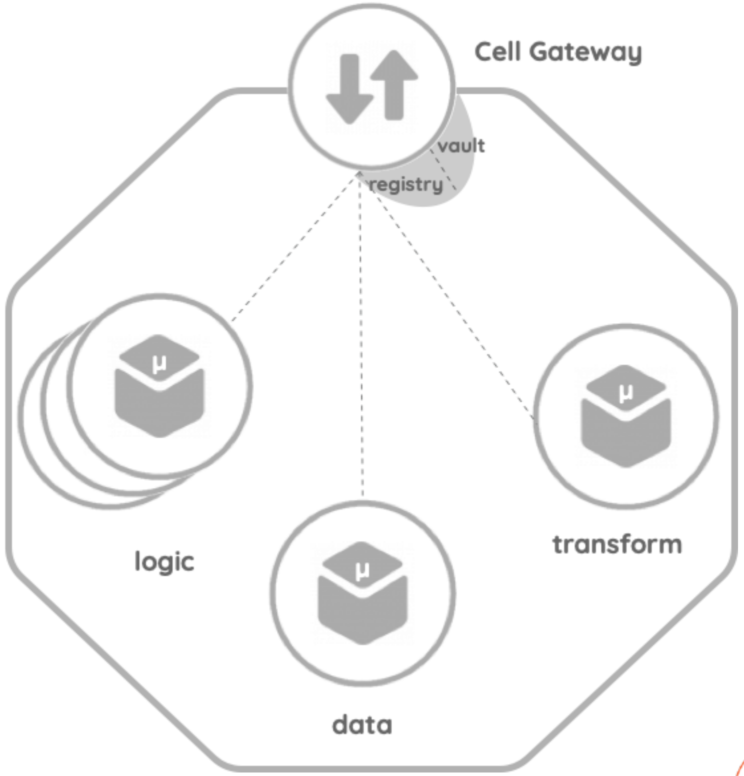


Component: Atomic Units

A **component** represents a process or business logic running in a container, serverless environment, or an existing runtime. A component is designed based on a specific scope, which can be independently run and reused at the runtime.

Cell: Units of Enterprise Architecture

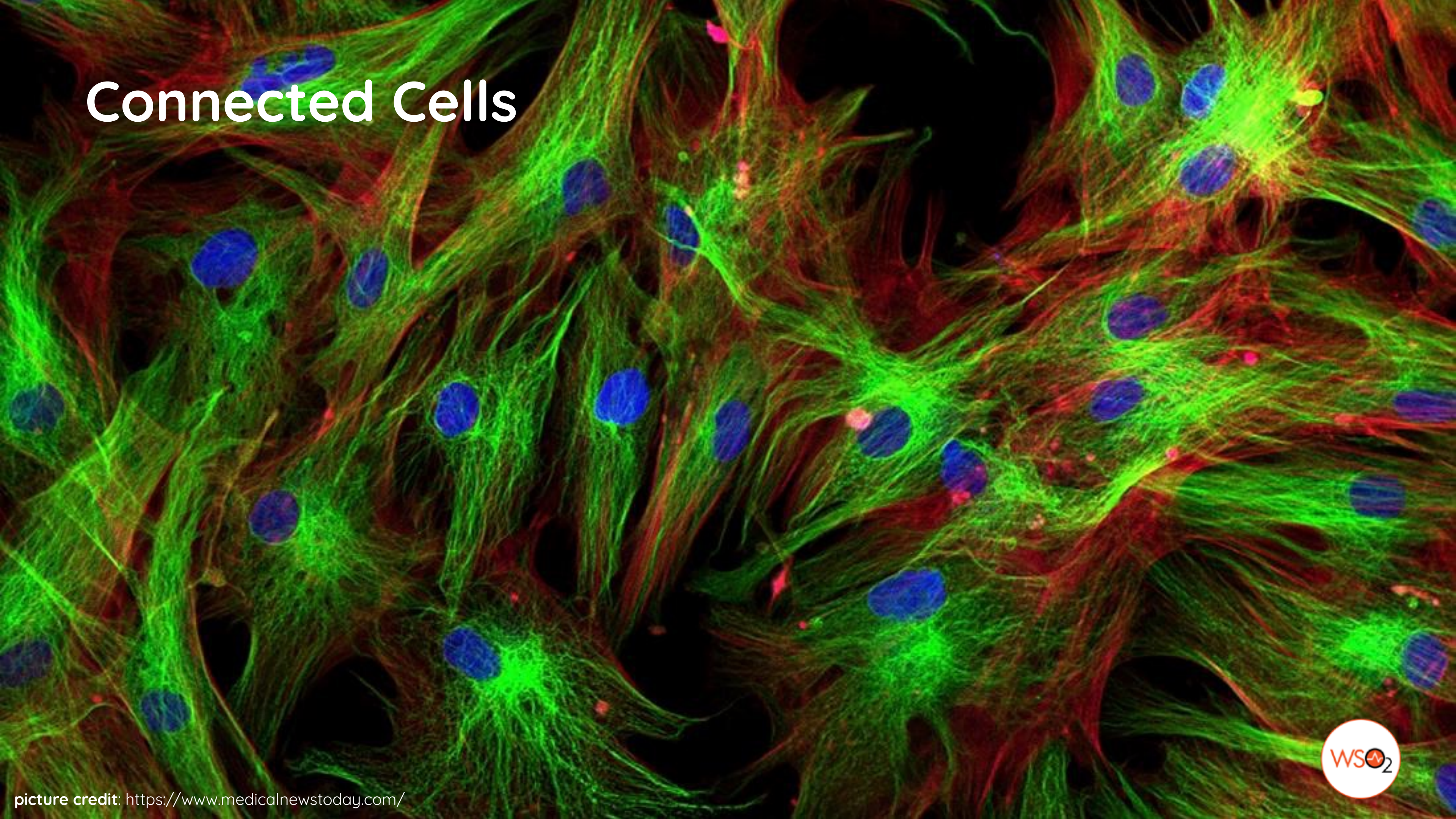
A **cell** is a collection of components, grouped from design and implementation into deployment. A cell is independently deployable, manageable, and observable.



Cell:Component

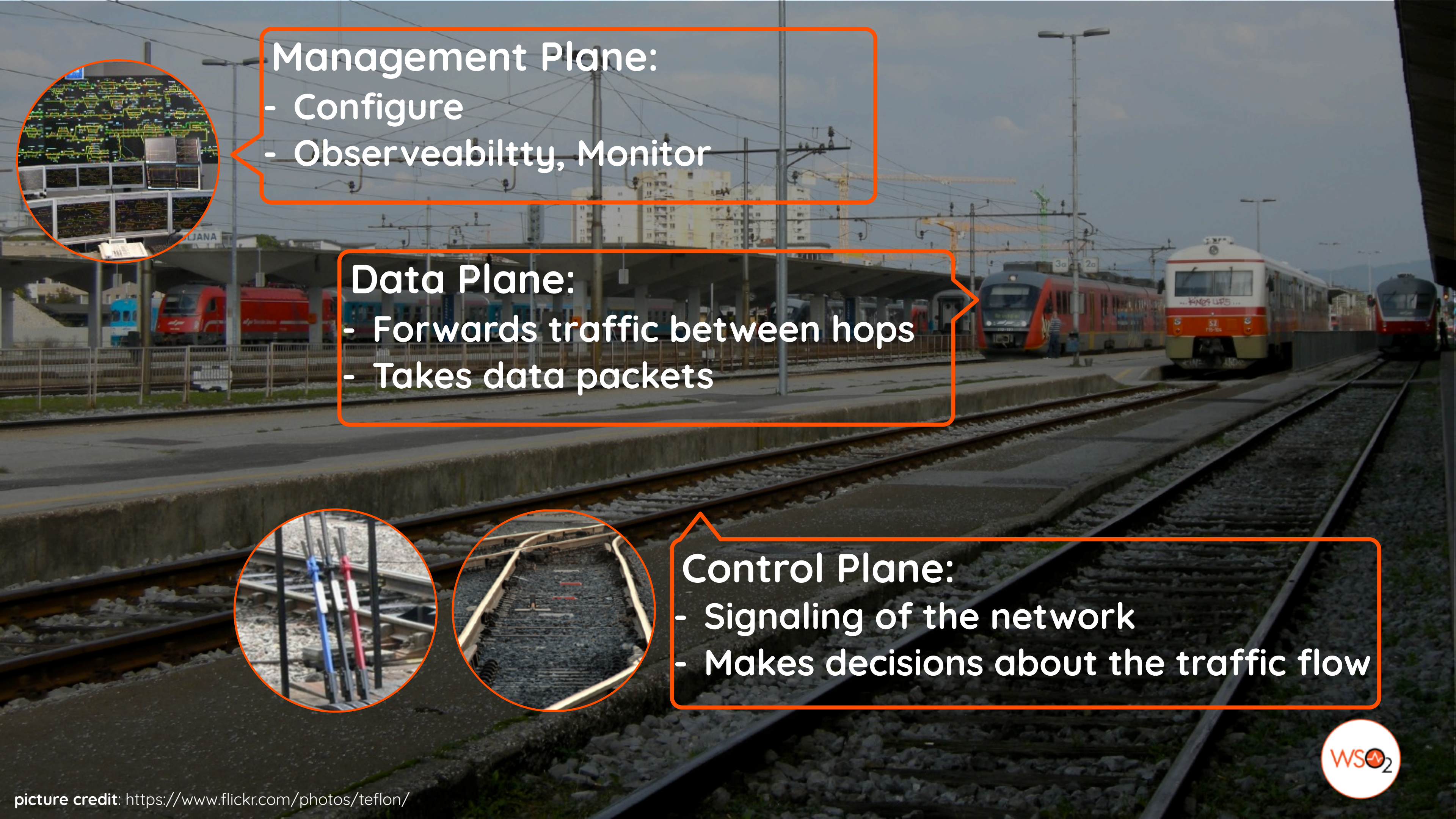
1:M

1:1



Connected Cells

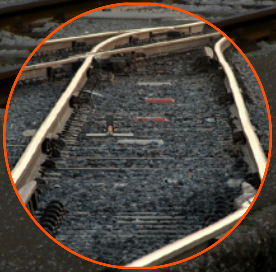
picture credit: https://www.medicalnewstoday.com/



Management Plane:
- Configure
- Observeabiltty, Monitor

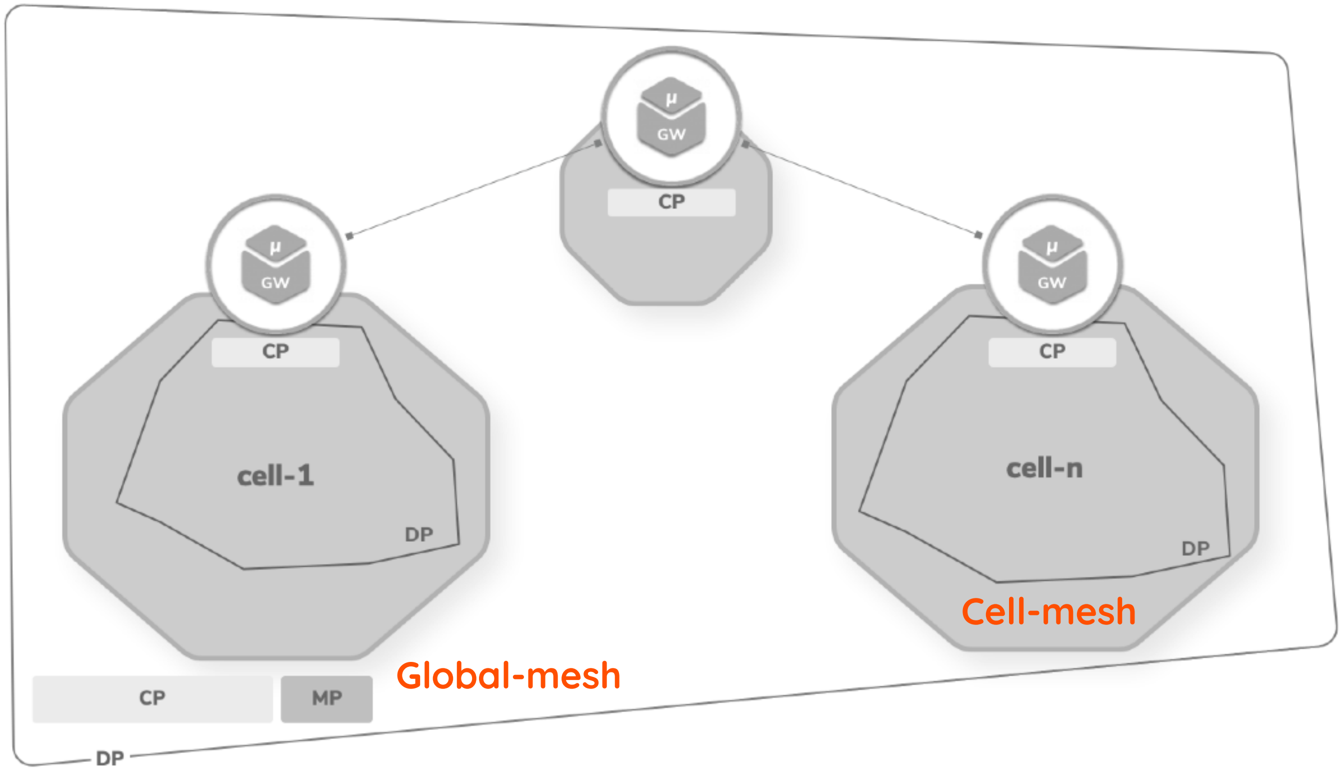
Data Plane:
- Forwards traffic between hops
- Takes data packets

Control Plane:
- Signaling of the network
- Makes decisions about the traffic flow



picture credit: https://www.flickr.com/photos/teflon/

Inter and Intra Cell communication



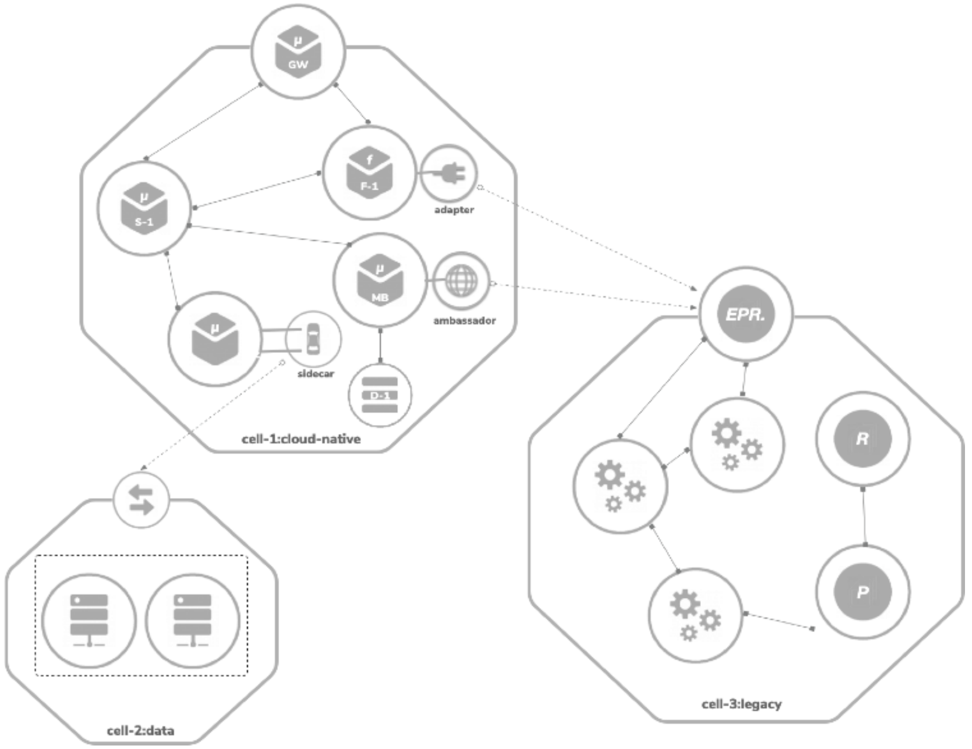
Connected Cells

Cell gateway (ingress)

Sidecar (egress)

Adaptor (egress)

Ambassador (egress)



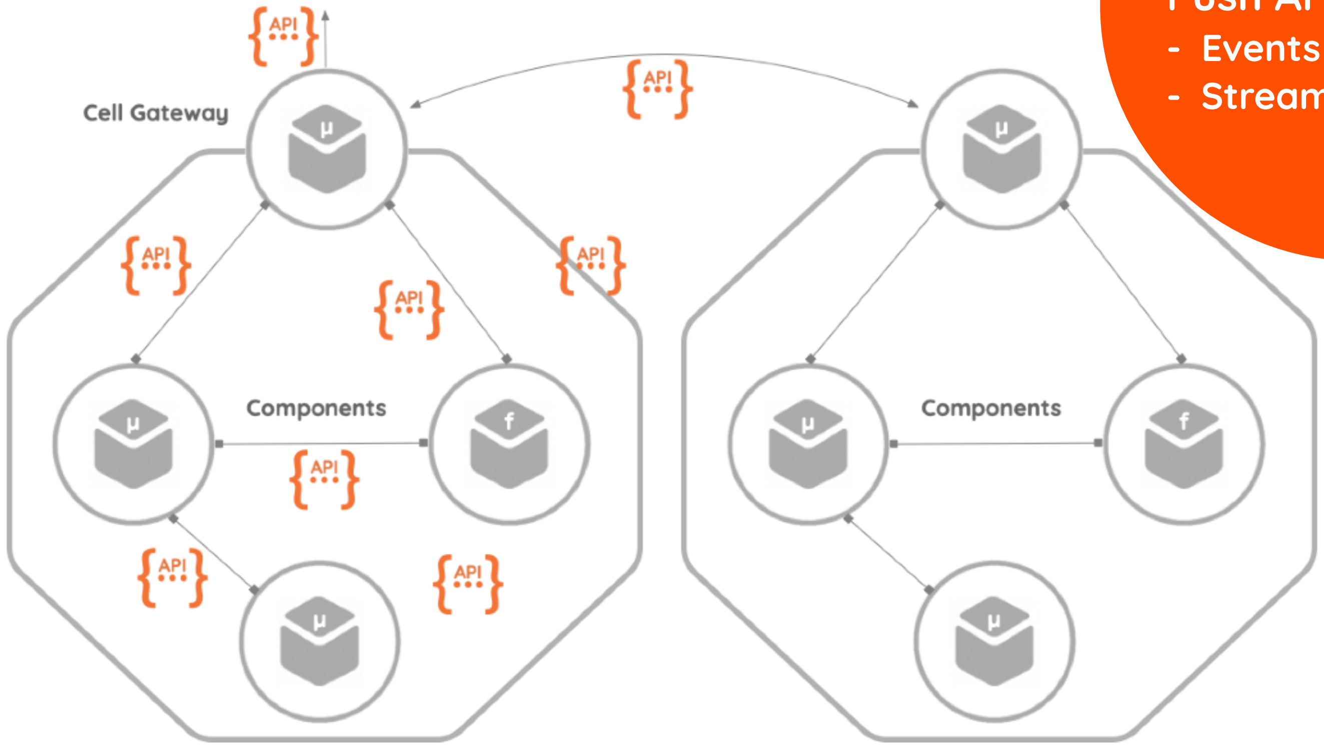
API-centric Architecture

Pull APIs

- RESTful HTTP, gRPC

Push APIs

- Events JMS, AMQP, SMTP
- Streams Kafka, MQTT



Gateway Pattern

picture credit: https://www.flickr.com/photos/ell-r-brown/

Automated Governance (Re)-enables Flow

Automated governance is made of three things:

A source of truth:

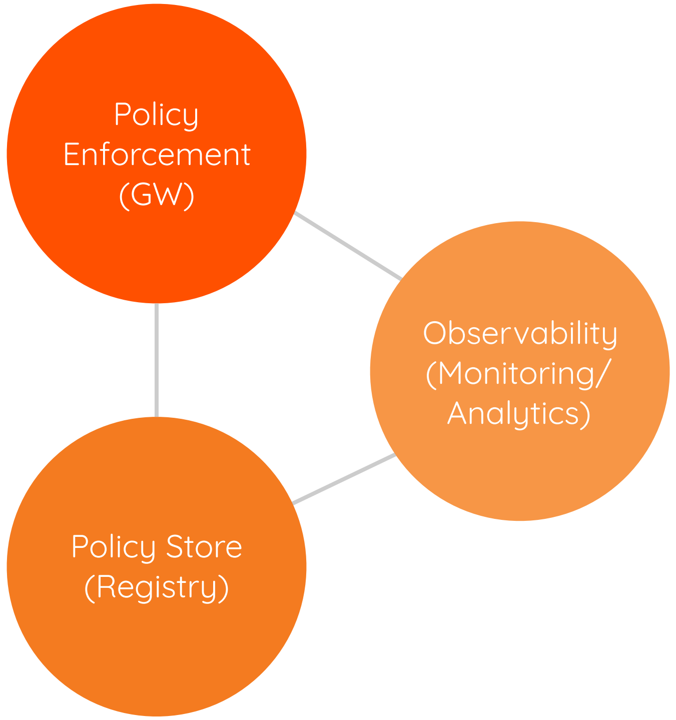
Policy store/registry

Enforcement of the policy

Gateway or plugin attempting to keep the desired state

Observability

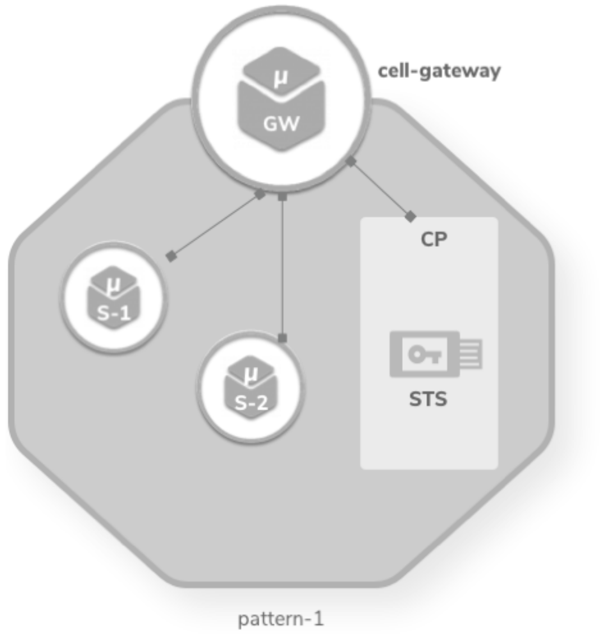
How close to the desired state are we?

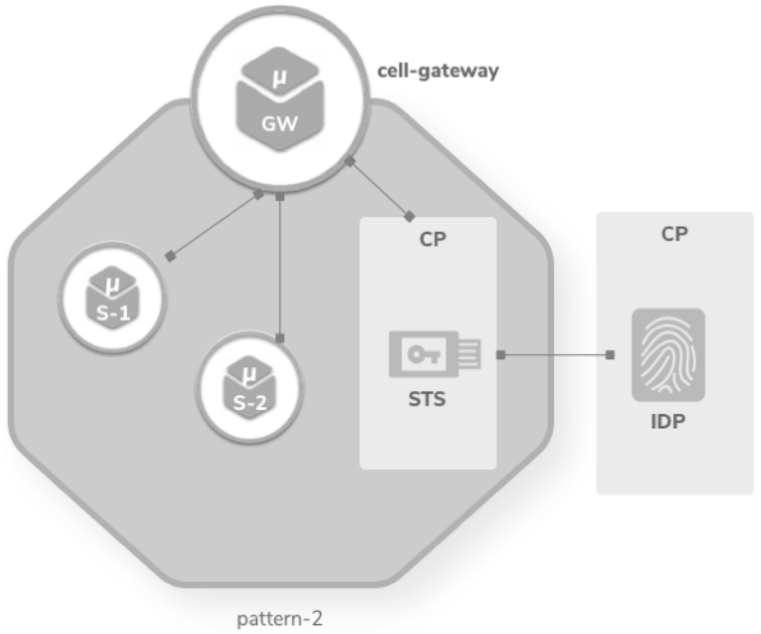


Security of Cells

picture credit: https://www.flickr.com/photos/laurelfan/ https://www.flickr.com/photos/sahdblunders/

Security of Cells





Developer Experience (DX) of a Cell

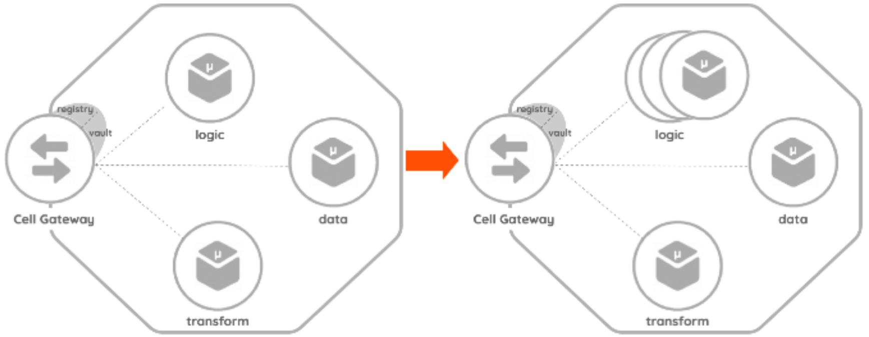
Creating Cells

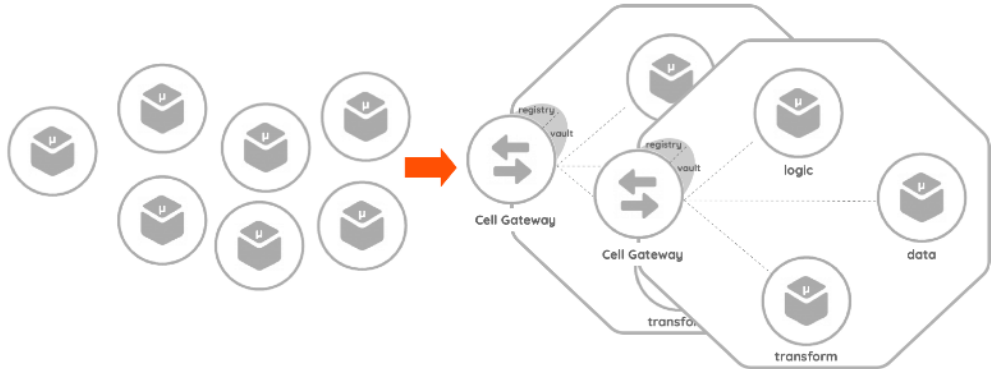
Brand new Cell

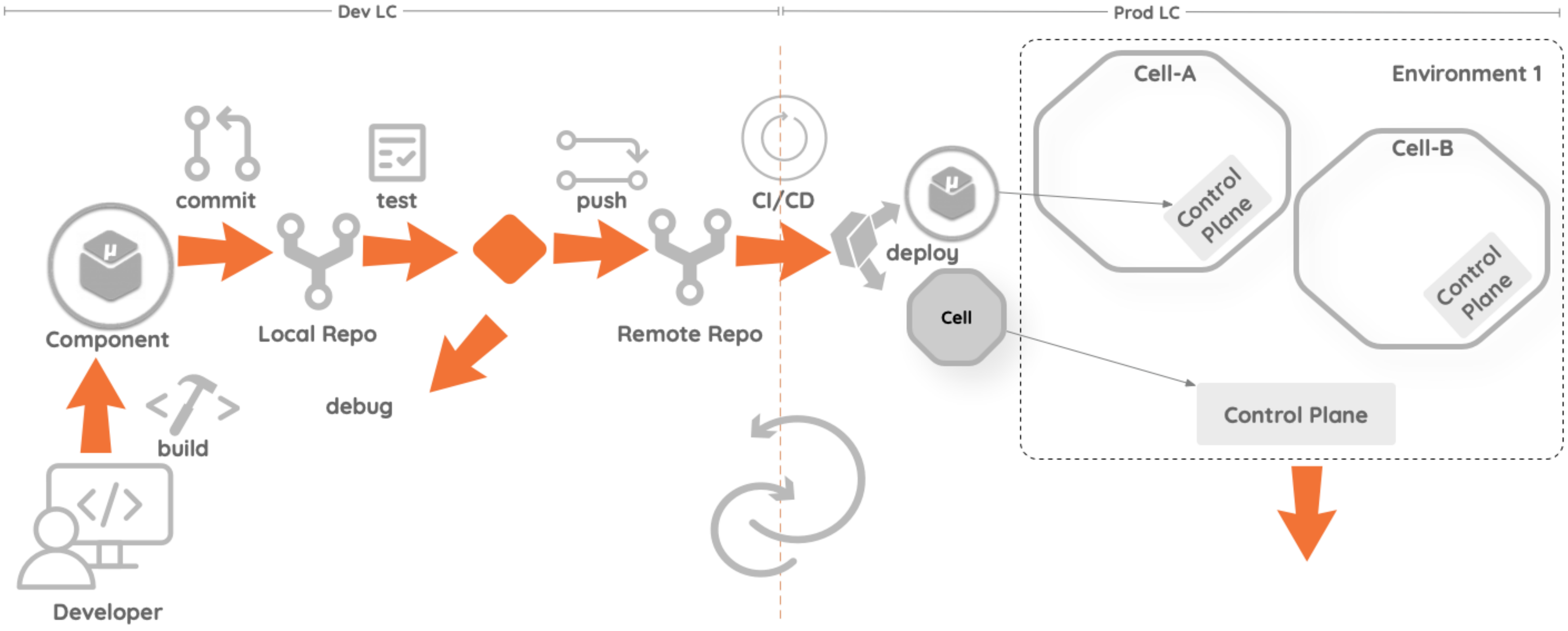
Existing (micro)services

Update an existing Cell

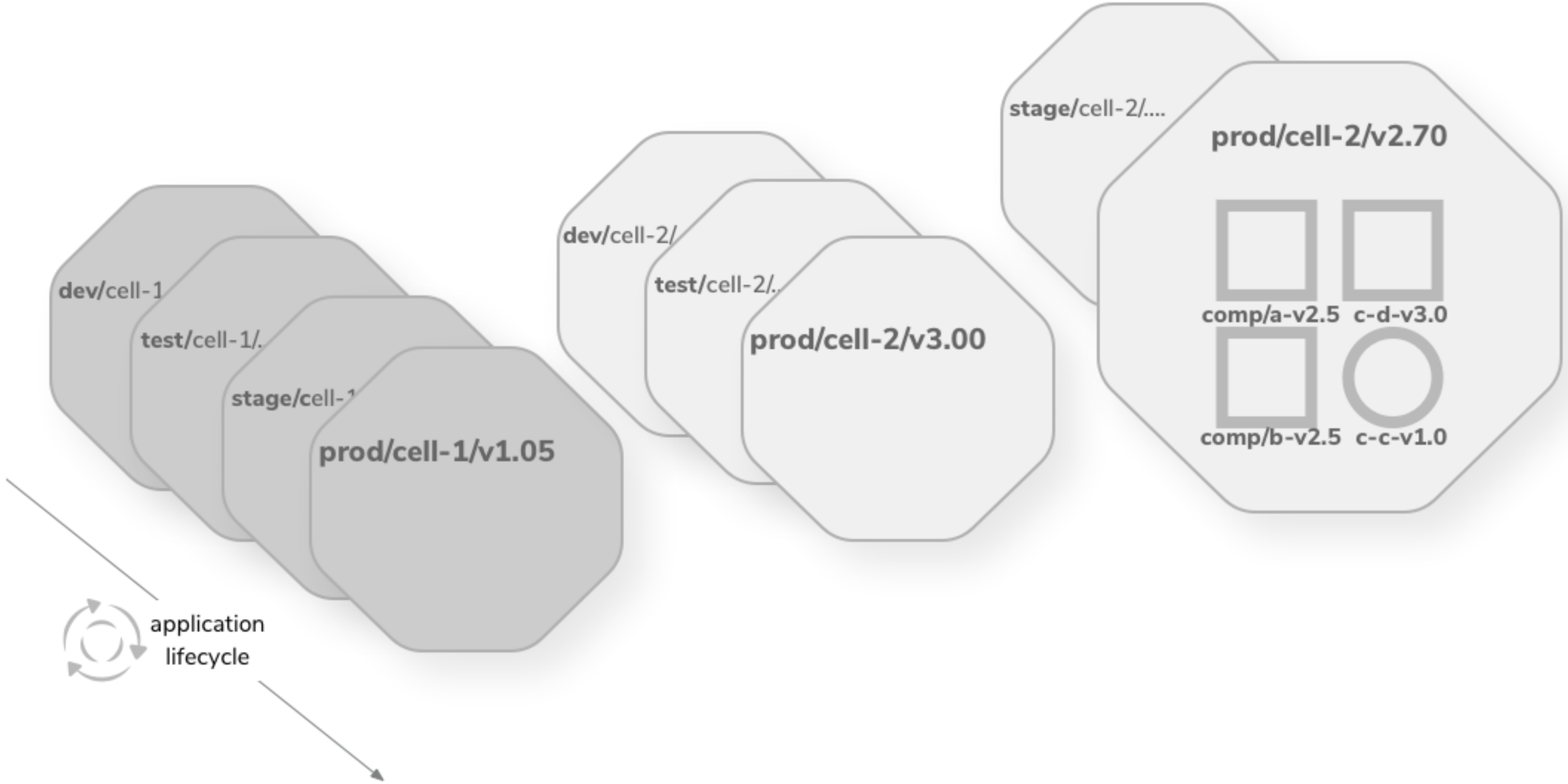
Create a new version






Dev LC
Prod LC
commit
test
push
CI/CD
Component
Local Repo
Remote Repo
debug
build
Developer
deploy
Cell
Cell-A
Environment 1
Control Plane
Cell-B
Control Plane
Control Plane

Lifecycle of a Cell


dev/cell-1
test/cell-1/.
stage/cell-1
prod/cell-1/v1.05
application
lifecycle
dev/cell-2/
test/cell-2/..
prod/cell-2/v3.00
stage/cell-2/....
prod/cell-2/v2.70
comp/a-v2.5
c-d-v3.0
comp/b-v2.5
c-c-v1.0

Structured Agility

Versioned Components

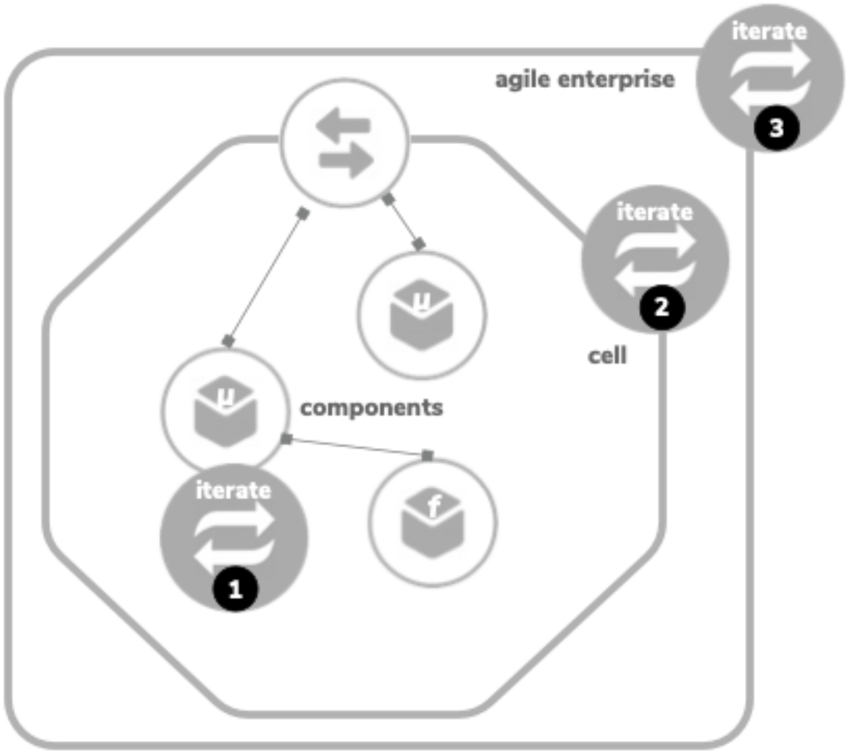
Versioned Cells

Dependency managed

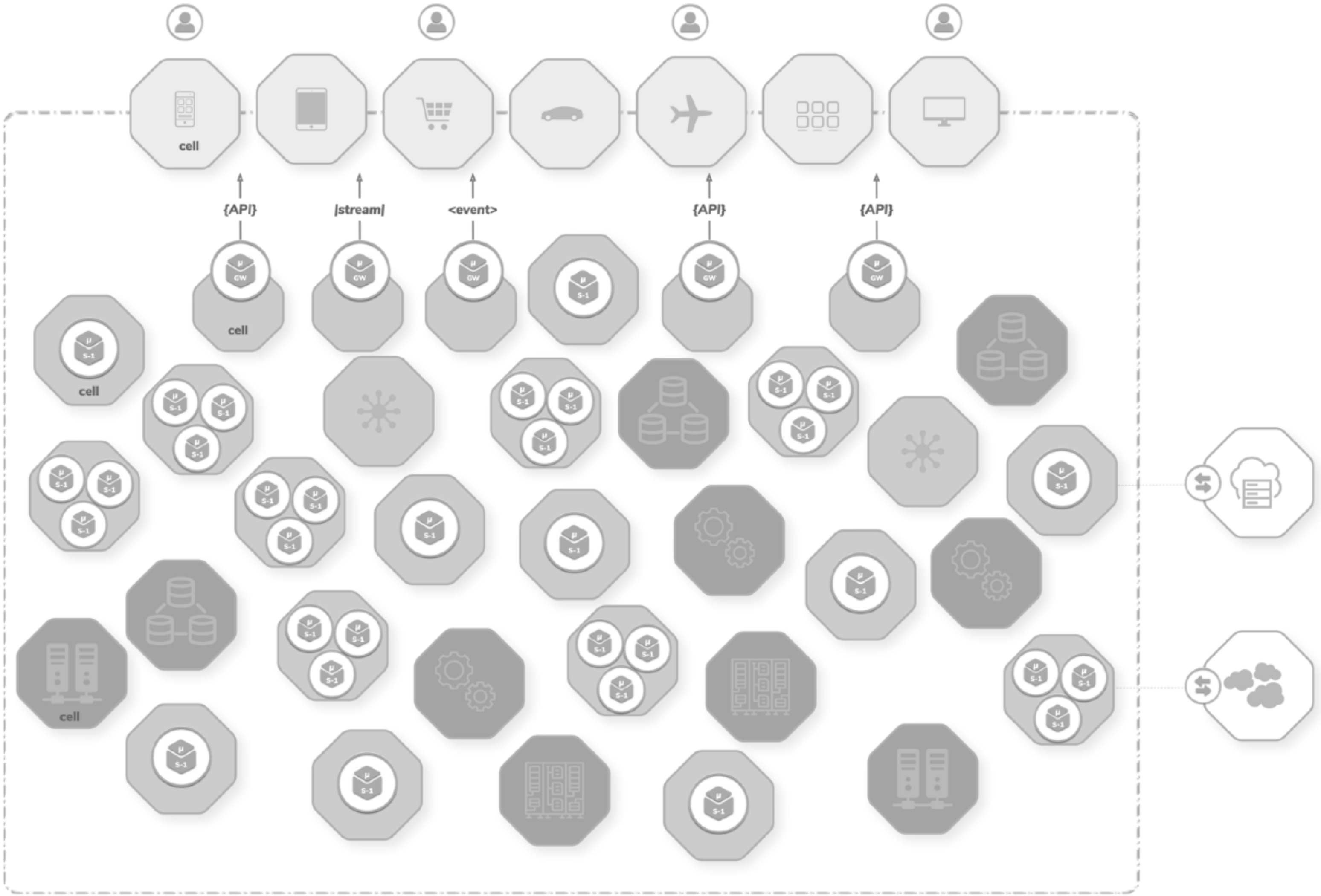
Autowired

Reusable

Enhanced MSA & CNA



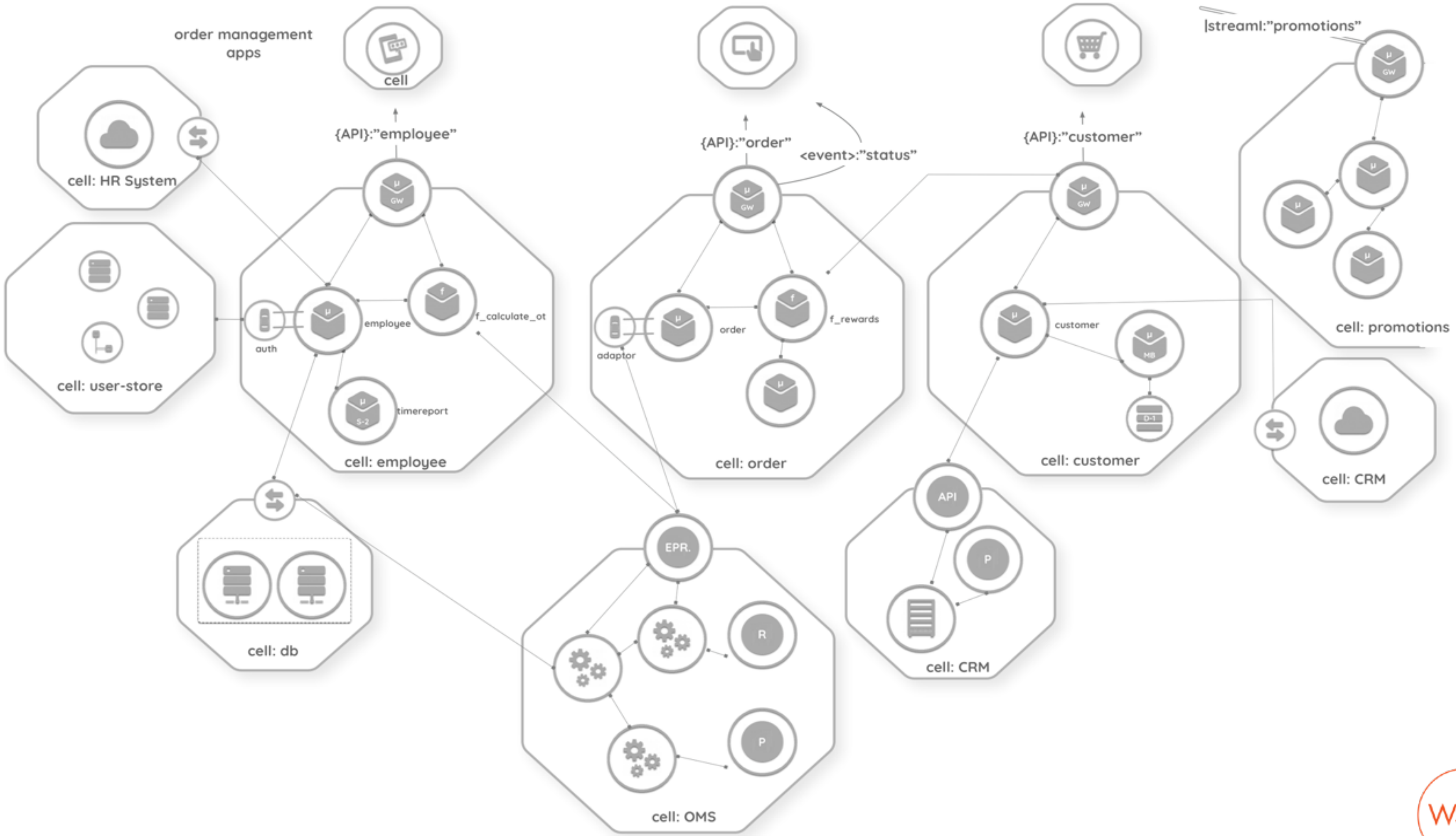
Cell-based Enterprise Architecture


cell
{API}
/stream/
<event>
{API}
{API}
GW
cell
cell
S-1
cell

Cell Types

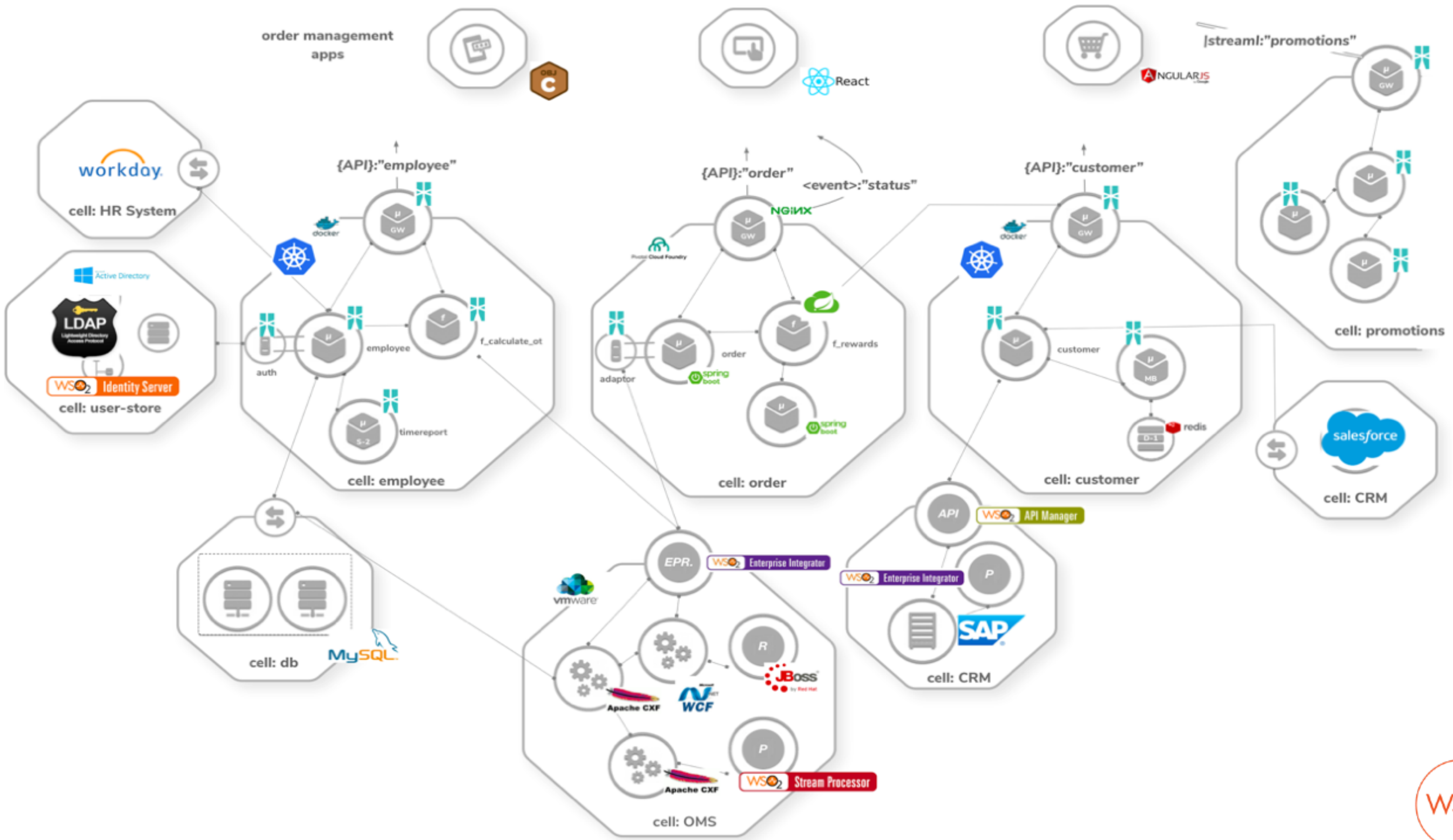
Cell Type	Components
Logic	Microservices, Functions, MicroGateways, lightweight storages
Integration	MicroESB or other integration microservices, lightweight storage and/or cache
Legacy	Existing systems, legacy services
External	SaaS and partner systems
Data	RDBMS, NoSQL, File, Message Broker*
Identity	IDP, user stores
Channel	Web Apps, IoT, mobile apps

Reference Implementation L0

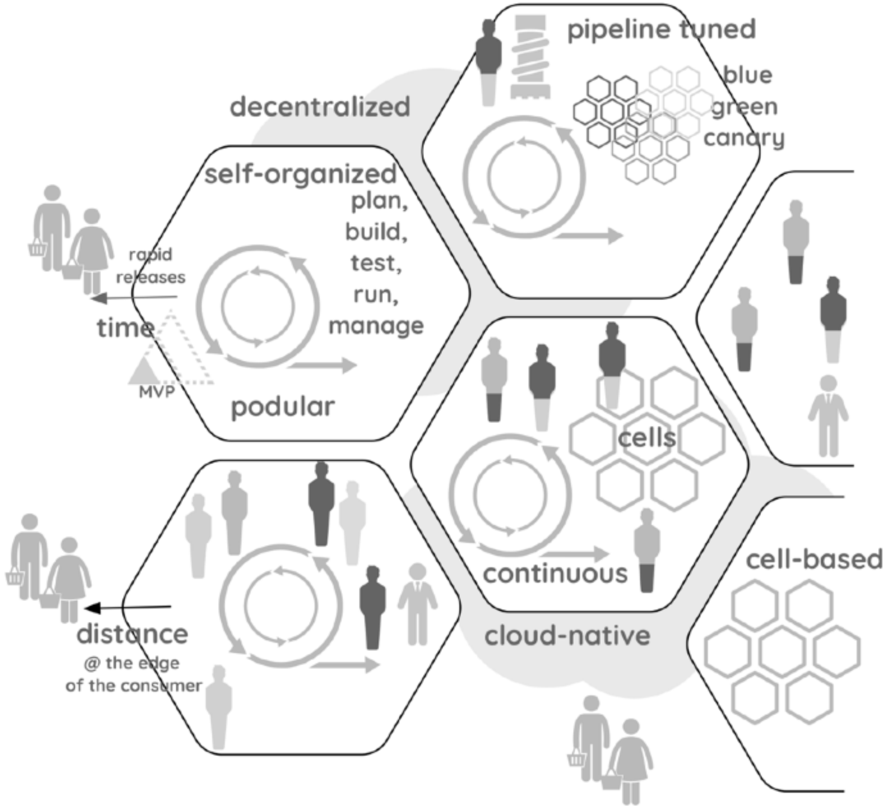


order management apps
cell
{API}:"employee"
{API}:"order"
<event>:"status"
{API}:"customer"
|stream|:"promotions"
cell: HR System
cell: user-store
auth
employee
f_calculate_ot
timereport
cell: employee
adaptor
order
f_rewards
cell: order
customer
cell: customer
cell: promotions
cell: CRM
API
P
cell: CRM
cell: db
EPR.
R
P
cell: OMS

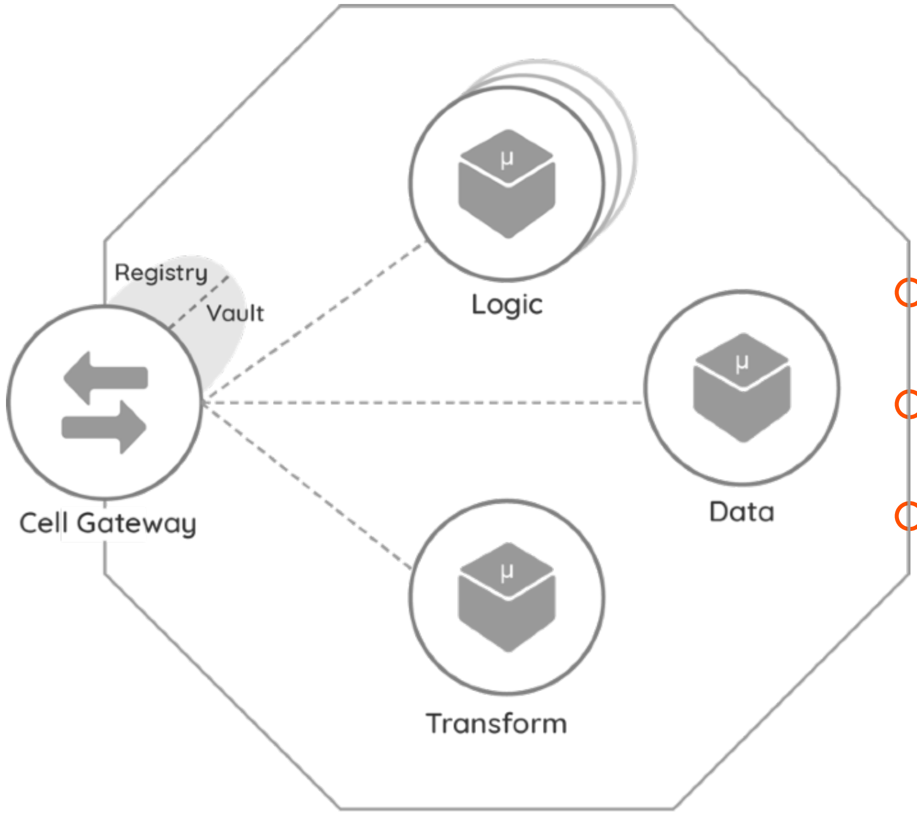
Reference Implementation L1


order management apps
React
ANGULARJS
|stream|:"promotions"
workday.
cell: HR System
{API}:"employee"
{API}:"order"
<event>:"status"
{API}:"customer"
NGINX
Pivotal Cloud Foundry
docker
GW
Active Directory
LDAP
Lightweight Directory Access Protocol
WSO2 Identity Server
cell: user-store
auth
employee
f_calculate_ot
timereport
cell: employee
adaptor
order
f_rewards
spring boot
cell: order
customer
MB
redis
cell: customer
cell: promotions
salesforce
cell: CRM
API
WSO2 API Manager
P
WSO2 Enterprise Integrator
SAP
cell: CRM
cell: db
MySQL
EPR.
WSO2 Enterprise Integrator
vmware
R
JBoss
Apache CXF
WCF
P
WSO2 Stream Processor
cell: OMS

Cells and Podular Organizations



Summary: Cell-based Architecture



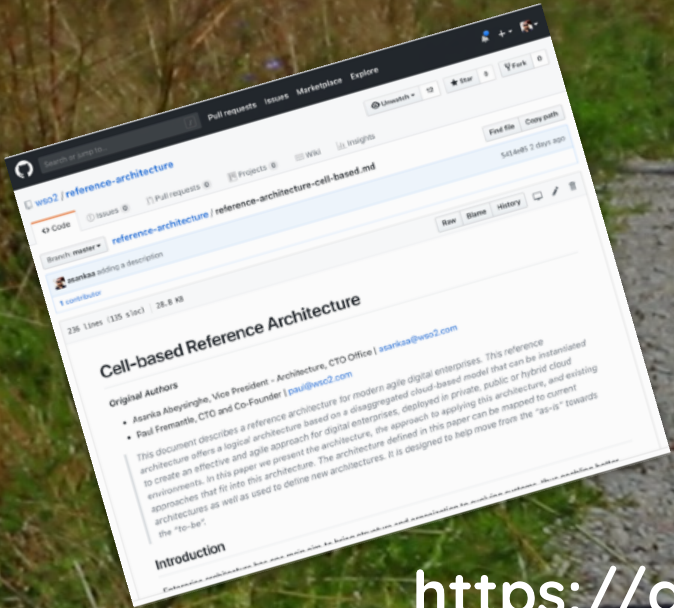
Self-contained

Deployable as a unit

Independently elastic

Data, control & management plane

Just a (steady) start



wso2 / reference-architecture

Branch: master reference-architecture / reference-architecture-cell-based.md

asankaa adding a description

541de05 2 days ago

1 contributor

236 lines (135 sloc) | 28.8 KB

Cell-based Reference Architecture

Original Authors

- Asanka Abeysinghe, Vice President - Architecture, CTO Office | asankaa@wso2.com
- Paul Fremantle, CTO and Co-Founder | paul@wso2.com

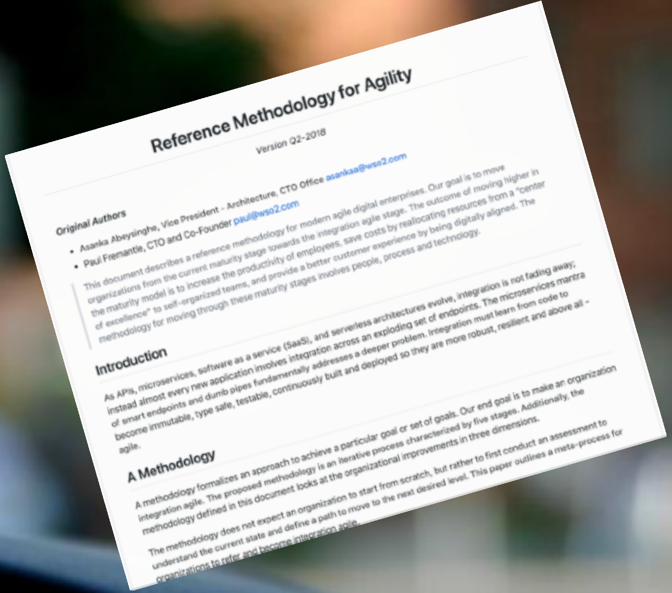
> This document describes a reference architecture for modern agile digital enterprises. This reference architecture offers a logical architecture based on a disaggregated cloud-based model that can be instantiated to create an effective and agile approach for digital enterprises, deployed in private, public or hybrid cloud environments. In this paper we present the architecture, the approach to applying this architecture, and existing approaches that fit into this architecture. The architecture defined in this paper can be mapped to current architectures as well as used to define new architectures. It is designed to help move from the "as-is" towards the "to-be".

Introduction

https://github.com/wso2/reference-architecture

picture credit: https://www.flickr.com/photos/josephmorris/

Reference Methodology



Reference Methodology for Agility

Version Q2-2018

Original Authors

- Asanka Abeysinghe, Vice President - Architecture, CTO Office asankaa@wso2.com
- Paul Fremantle, CTO and Co-Founder paul@wso2.com

> This document describes a reference methodology for modern agile digital enterprises. Our goal is to move organizations from the current maturity stage towards the integration agile stage. The outcome of moving higher in the maturity model is to increase the productivity of employees, save costs by reallocating resources from a "center of excellence" to self-organized teams, and provide a better customer experience by being digitally aligned. The methodology for moving through these maturity stages involves people, process and technology.

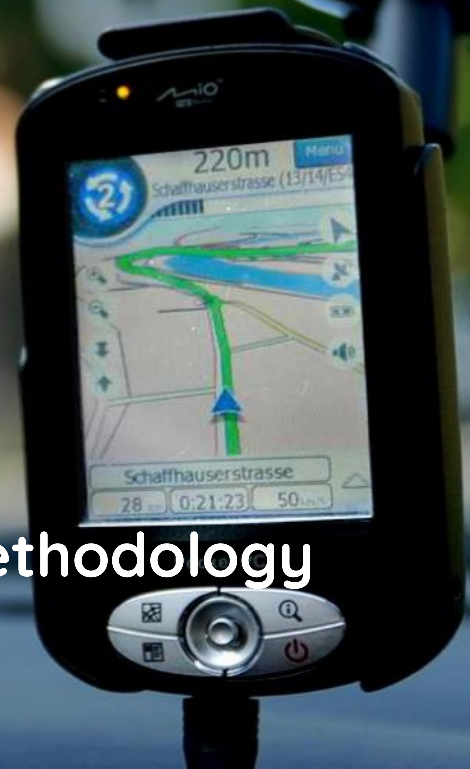
Introduction

As APIs, microservices, software as a service (SaaS), and serverless architectures evolve, integration is not fading away; instead almost every new application involves integration across an exploding set of endpoints. The microservices mantra of smart endpoints and dumb pipes fundamentally addresses a deeper problem. Integration must learn from code to become immutable, type safe, testable, continuously built and deployed so they are more robust, resilient and above all - agile.

A Methodology

A methodology formalizes an approach to achieve a particular goal or set of goals. Our end goal is to make an organization integration agile. The proposed methodology is an iterative process characterized by five stages. Additionally, the methodology defined in this document looks at the organizational improvements in three dimensions.

The methodology does not expect an organization to start from scratch, but rather to first conduct an assessment to understand the current state and define a path to move to the next desired level. This paper outlines a meta-process for organizations to refer and become integration agile.

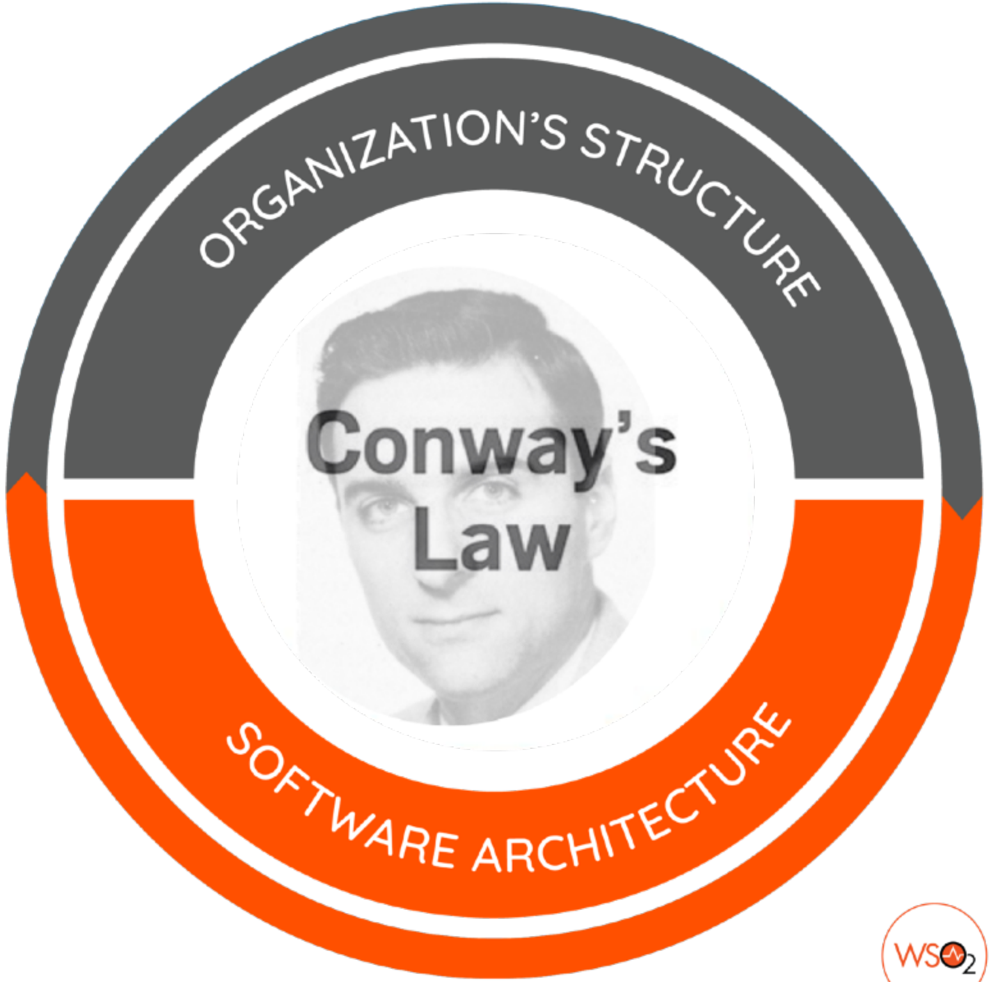


https://github.com/wso2/reference-methodology

picture credit: https://www.flickr.com/photos/vasile23/

“Organizations which design systems … are constrained to produce designs which are copies of the <u>communication structures</u> of these organizations.”

- M. Conway



THANK YOU

@asankama

https://www.linkedin.com/in/asankaabeysinghe/