20TH ANNIVERSARY EDITION
WSO2CON ASIA
PLATFORMLESS MODERNIZATION

Mastering Cell-based Architecture for Modern Enterprises

Chintana Wilamuna
VP, Solutions Architecture
WSO2

Agenda

The challenge: Limits of monolithic architectures

- Slow delivery
  - Tightly coupled components make change risky and slow
- Reduced resilience
  - A failure in one part can bring down the entire system
- Difficult to scale
  - Must scale the entire application, even if only one feature is under heavy load
- High cognitive load
  - Teams struggle to understand the complex, tangled codebase

The shift: A new architectural approach

- Faster software delivery requires a new architectural approach with,

- **Decentralization:** Distributing control and data
- **Loose coupling:** Ensure components can be changed without breaking others
- **Scalability:** Allowing independent growth of different system parts

Introducing Cell-based Architecture (CBA)

- CBA is a transformative approach that structures an organization's capabilities into a network of independent, self-contained "cells"
- Similar to building with advanced, autonomous LEGO bricks instead of a single block of clay

Reference:

https://github.com/wso2/reference-architecture/blob/master/reference-architecture-cell-based.md

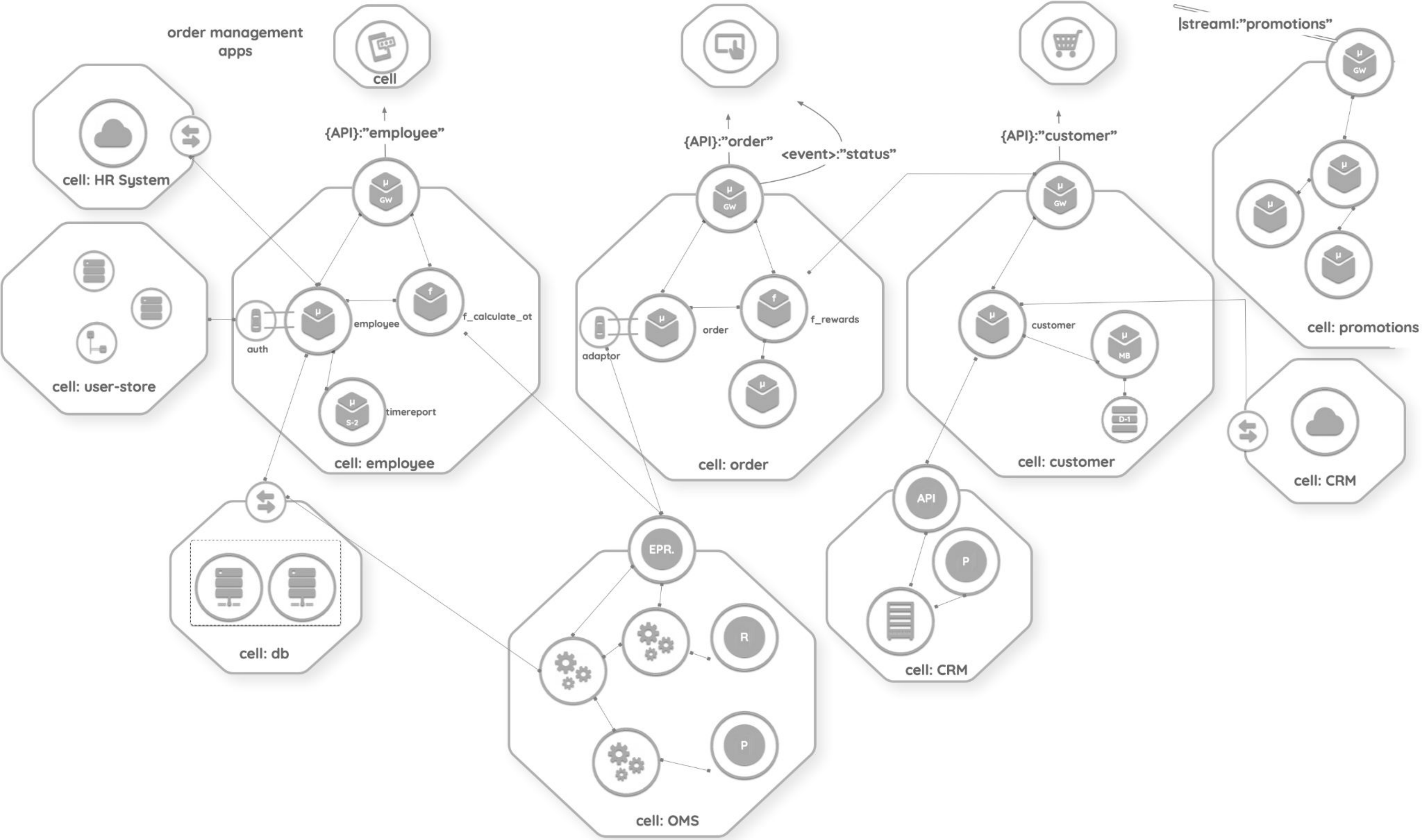
Who’s using cell-based architecture today?

okta

DOORDASH

Booking.com

Okta - https://www.okta.com/resources/whitepaper/how-okta-builds-and-runs-scalable-infrastructure/
Booking.com - https://www.youtube.com/watch?v=z8KLVZBHK-E
DoorDash - https://www.infoq.com/news/2024/01/doordash-service-mesh/

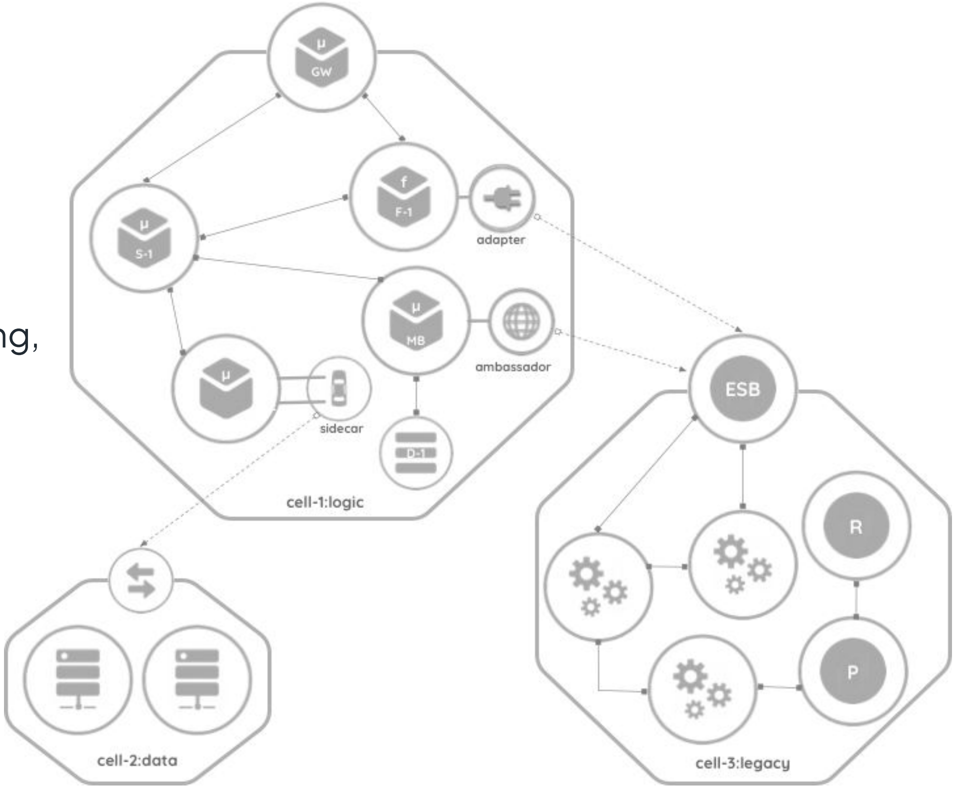

order management apps
cell
{API}:"employee"
{API}:"order"
<event>:"status"
{API}:"customer"
|streaml:"promotions"
cell: HR System
cell: user-store
auth
employee
f_calculate_ot
timereport
S-2
cell: employee
adaptor
order
f_rewards
cell: order
customer
MB
D-1
cell: customer
cell: promotions
cell: CRM
API
P
cell: CRM
cell: db
EPR.
R
P
cell: OMS
GW
f

Core principles of CBA

- Decentralization
  - No central point of failure or control. Each cell is a first class citizen in the network
- Loose coupling
  - Cells communicate only through their secure, managed gateways using well-defined APIs and events
- Independent deployability
  - A team can update and deploy their cell at any time without coordinating a “big bang” release
- High observability
  - Each cell is responsible for its own monitoring and logging, can be aggregated for a system-wide view
- Team ownership
  - A single, empowered team owns the full lifecycle of a cell (build, deploy, operate, maintain)

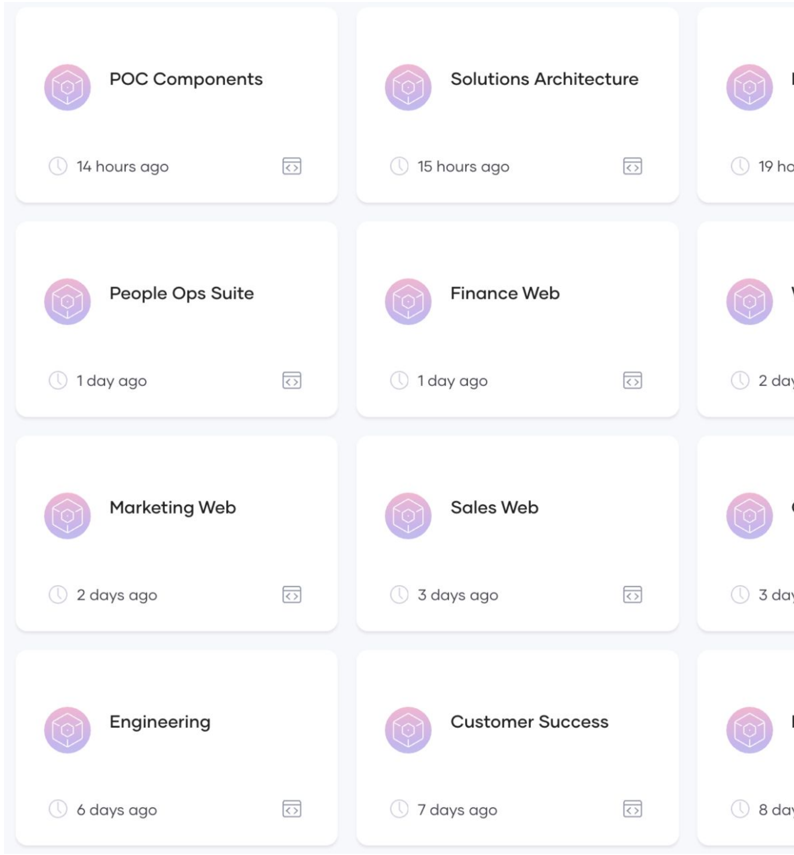
Anatomy of a cell

- A cell is more than a microservice, it's a complete, vertical slice of functionality
- API Gateway
  - The single, secure entry point. Manages routing, security, and policies
- Components
  - The actual business logic (microservices, functions and applications)
- Private data
  - The cell's own database or storage. It's never directly accessed by other cells
- Policies
  - Governance rules for security, rate-limiting, and access control are embedded within the cell



Designing with cells – A practical guide

- Identify business domains
  - Use techniques like Domain-Driven Design (DDD) to find the logical boundaries in your organization (e.g. “Shipping”, “Marketing”, “Finance”)
- Define cell boundaries
  - Map these domains to cells. A cell should have a clear purpose and a high level internal cohesion
- Design the gateway contract
  - Define the public API for each cell. What capabilities does it expose? What events does it publish?
- Establish communication
  - Decide how cells will interact, through synchronous API calls for immediate needs or asynchronous events for decoupled workflows



POC Components
14 hours ago

Solutions Architecture
15 hours ago

19 ho

People Ops Suite
1 day ago

Finance Web
1 day ago

2 day

Marketing Web
2 days ago

Sales Web
3 days ago

3 day

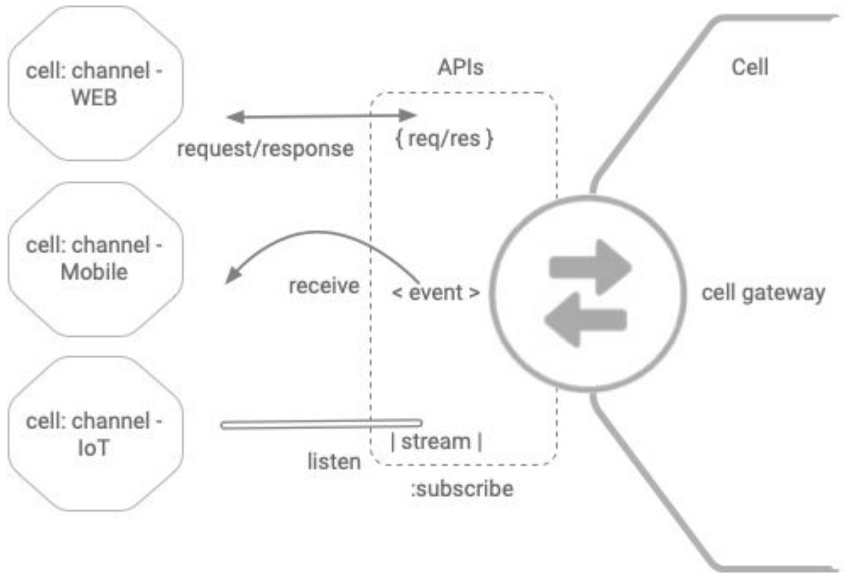
Engineering
6 days ago

Customer Success
7 days ago

8 day

Cell communication patterns

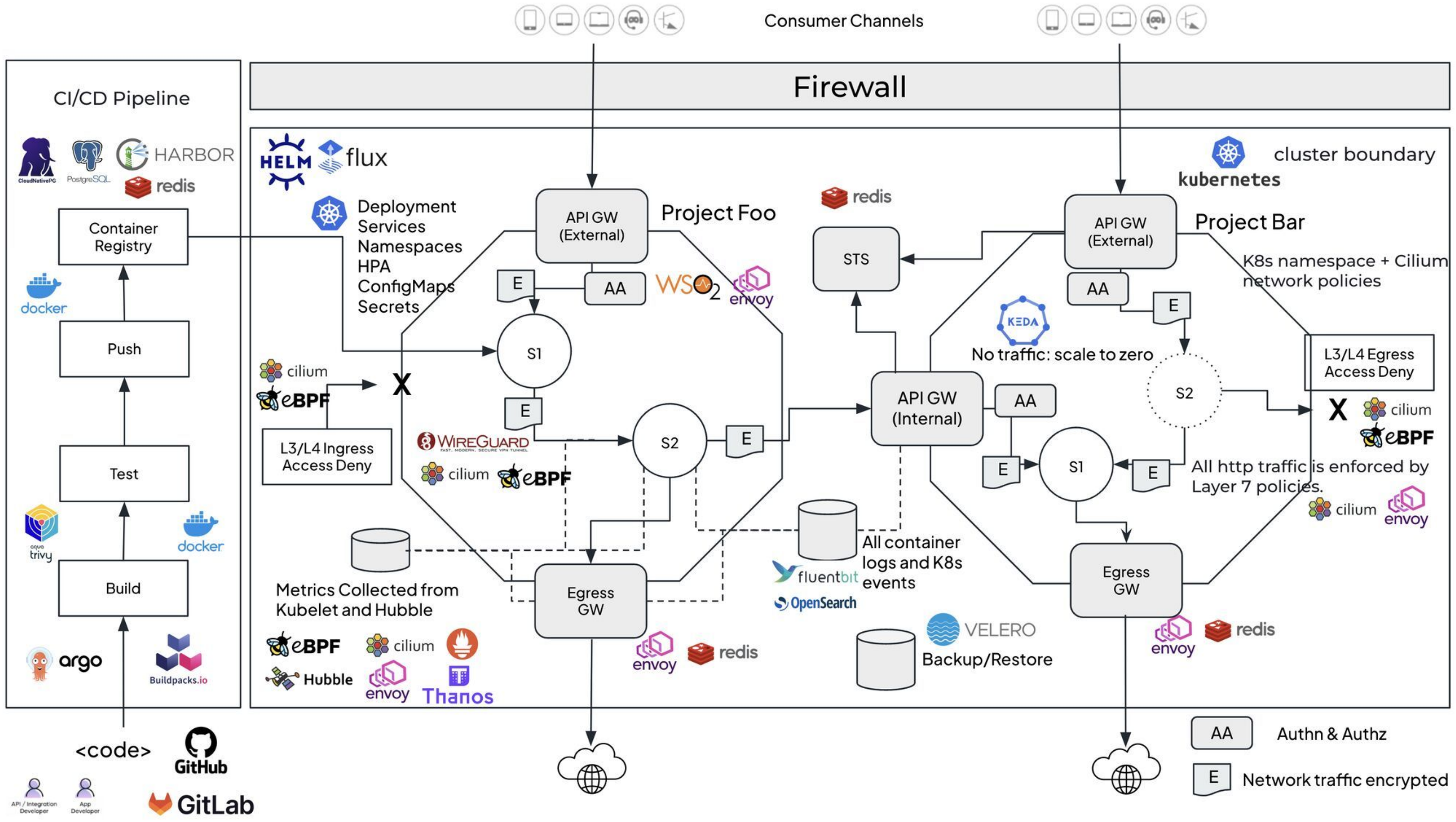
- Cells communicate through their gateways, never directly with internal components
- Synchronous request/response
  - Cell A makes a direct API call to Cell B's gateway
  - Used for queries or actions that need an immediate response
- Asynchronous event-based
  - Cell A publishes an event (e.g. OrderPlaced) to a message bus
  - Other cells (like Billing and Shipping) subscribe to that event and react independently
  - This is the preferred pattern for loose coupling



Benefits and considerations

- Benefits
  - Accelerated development: Teams can innovate and deploy independently
  - Improved system resilience: Failures are contained within a cell, not catastrophic
  - Targeted scalability: Scale only the cells that need it, saving costs
  - Clear ownership and accountability: Cultivate a culture of ownership
- Considerations
  - Distributed systems complexity: Requires robust observability and debugging tools
  - Eventual consistency: Managing data consistency across cells is a design challenge
  - Organizational shift: Requires moving from project teams to long-lived product teams

Implementing Cell-based architecture



Implementing and adopting CBA

DIY	Choreo	OpenChoreo
Choose components you want and build the platform yourself	SaaS https://choreo.dev Sign up and start using a cell-based implementation from the browser today	Free and open source https://github.com/openchoreo/openchoreo Setup the infrastructure in your data center/cloud provider

Key takeaways

- Cell-based architecture is a powerful model for building agile, scalable and resilient enterprise systems
- It is built on the core principles of decentralization, loose coupling, and clear team ownership
- A cell is a self-contained, independently deployable unit with its own gateway, logic and data
- Adopting CBA is as much an organizational and cultural shift as it is a technical one

Question Time!

Thank you!

20TH ANNIVERSARY EDITION
WSO2CON ASIA
PLATFORMLESS MODERNIZATION