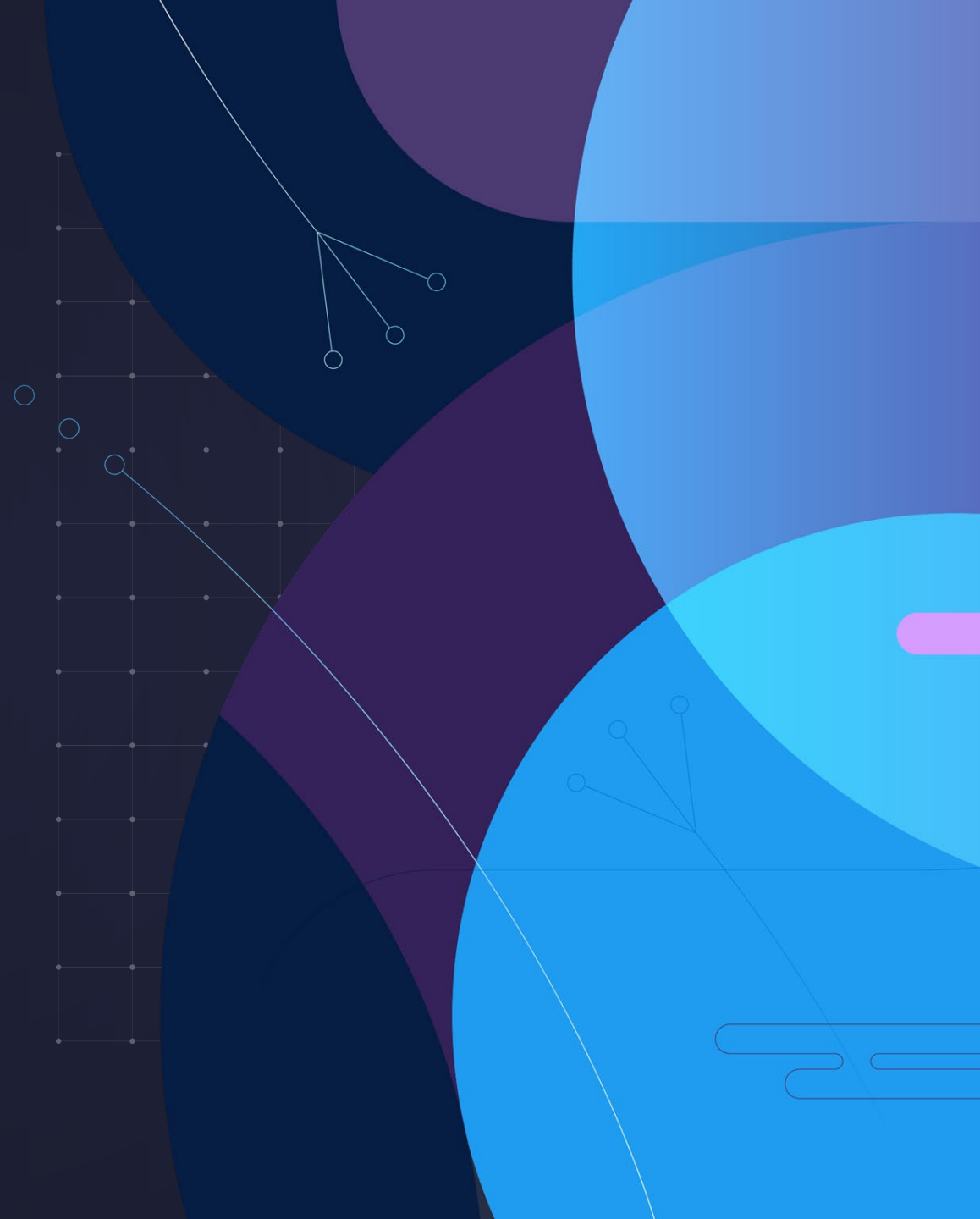# Accelerate the delivery - reduce your dependencies

Mohamed Megahed (mohamed.megahed@microsoft.com)

Dennis van de Laar (devandel@microsoft.com)

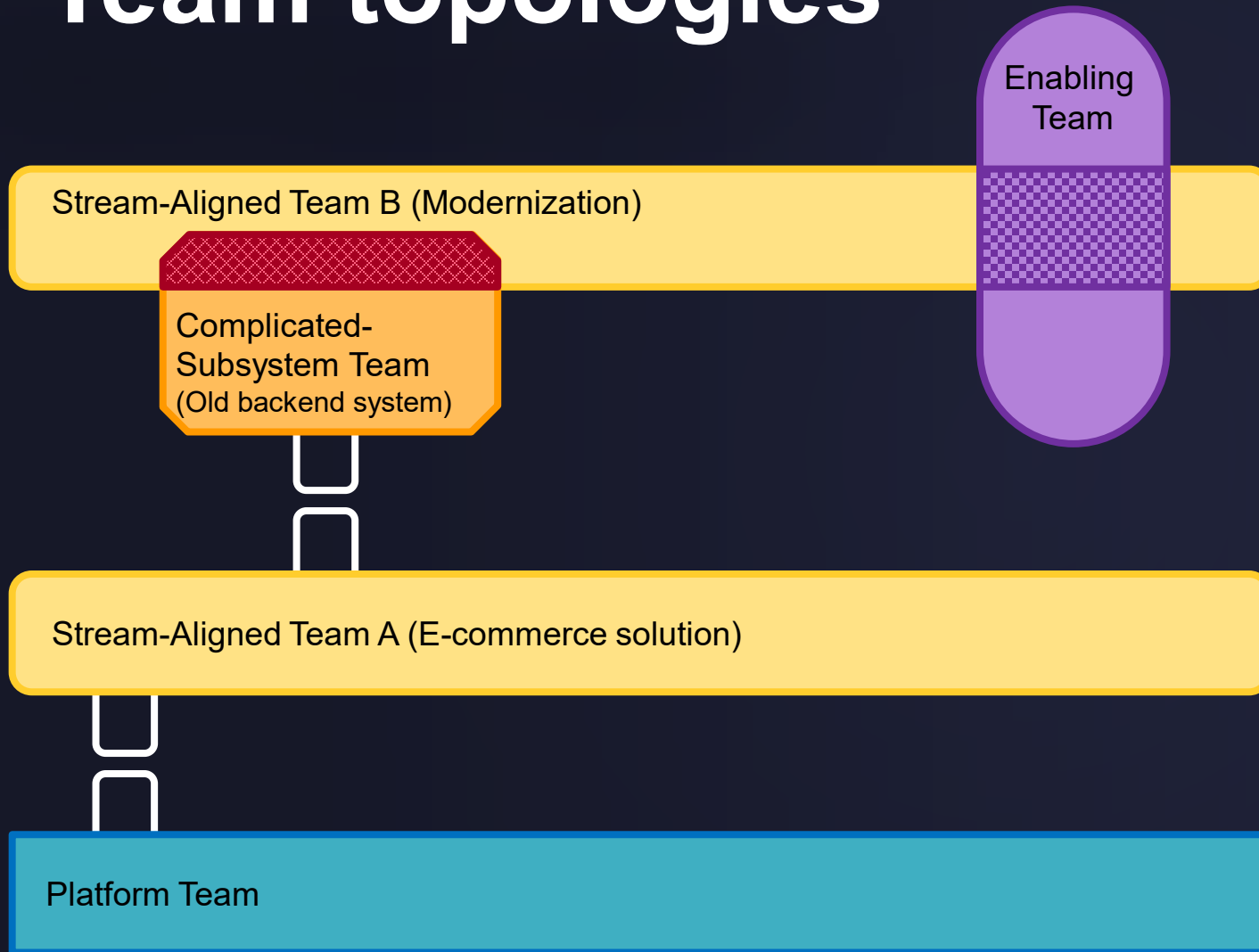# Agenda

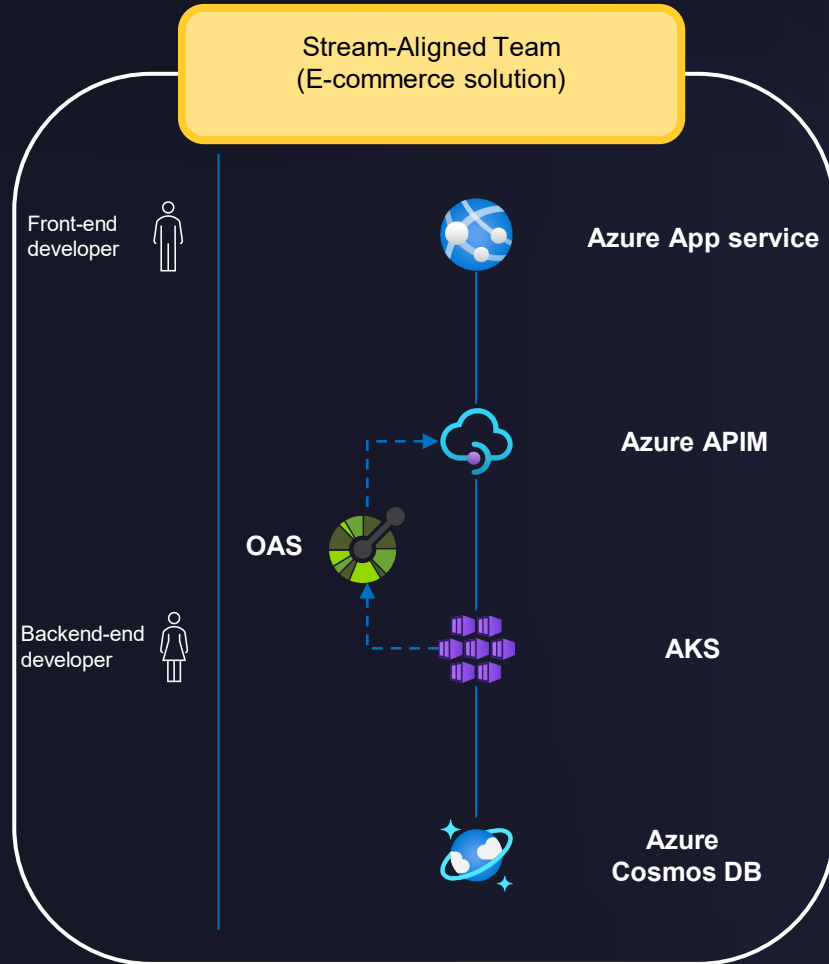Team topologies

Build API's at scale


Enabling consuming parties

Team topologies

Enabling Team

Stream-Aligned Team B (Modernization)

Complicated-Subsystem Team
(Old backend system)

Stream-Aligned Team A (E-commerce solution)

Platform Team

API usage within a stream aligned team

Stream-Aligned Team
(E-commerce solution)

Front-end developer

Azure App service

Azure APIM

OAS

Backend-end developer

AKS

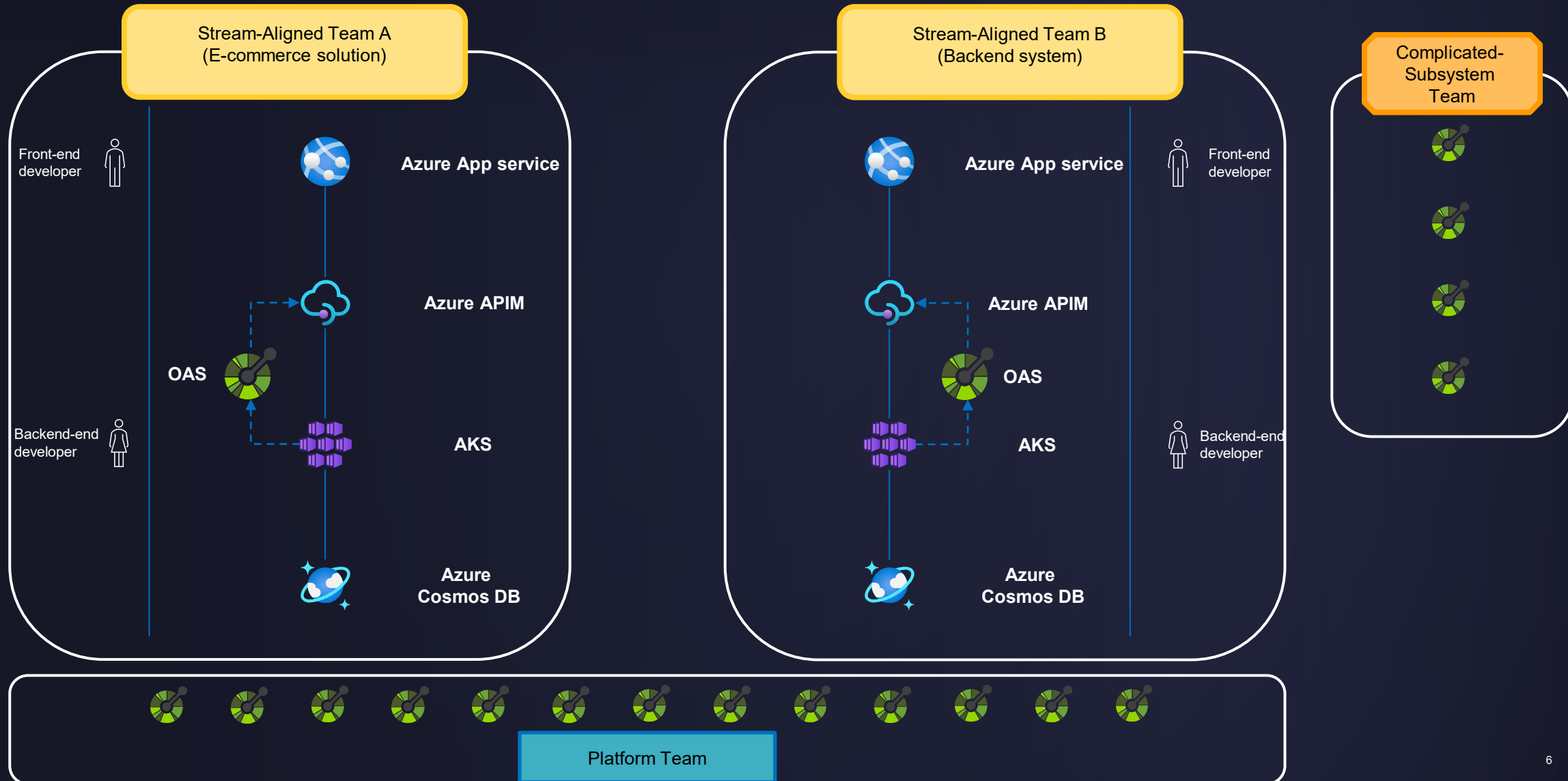Azure Cosmos DB

Two approaches are being used:
- Code-First
- API-First

API usage across multiple stream-aligned teams
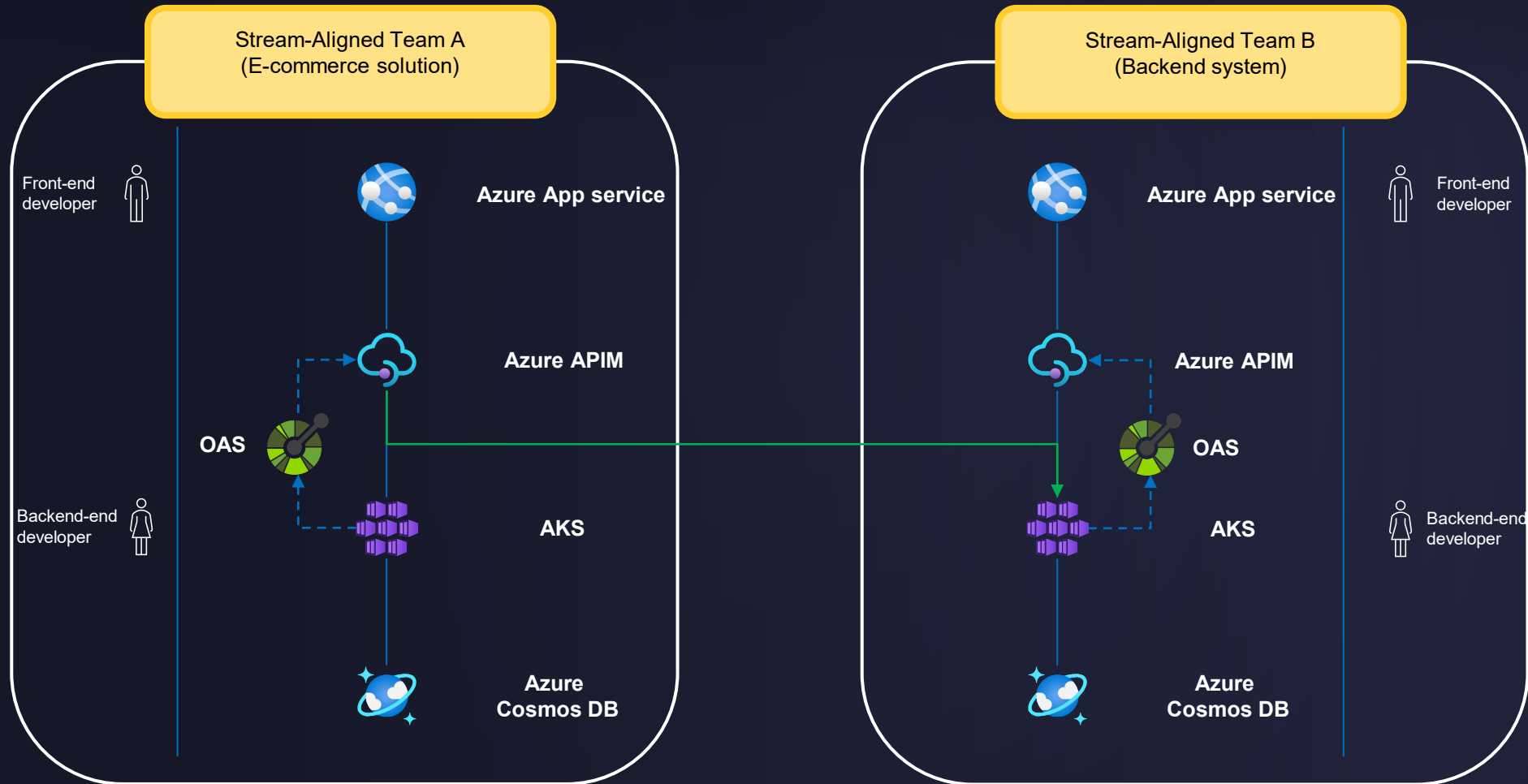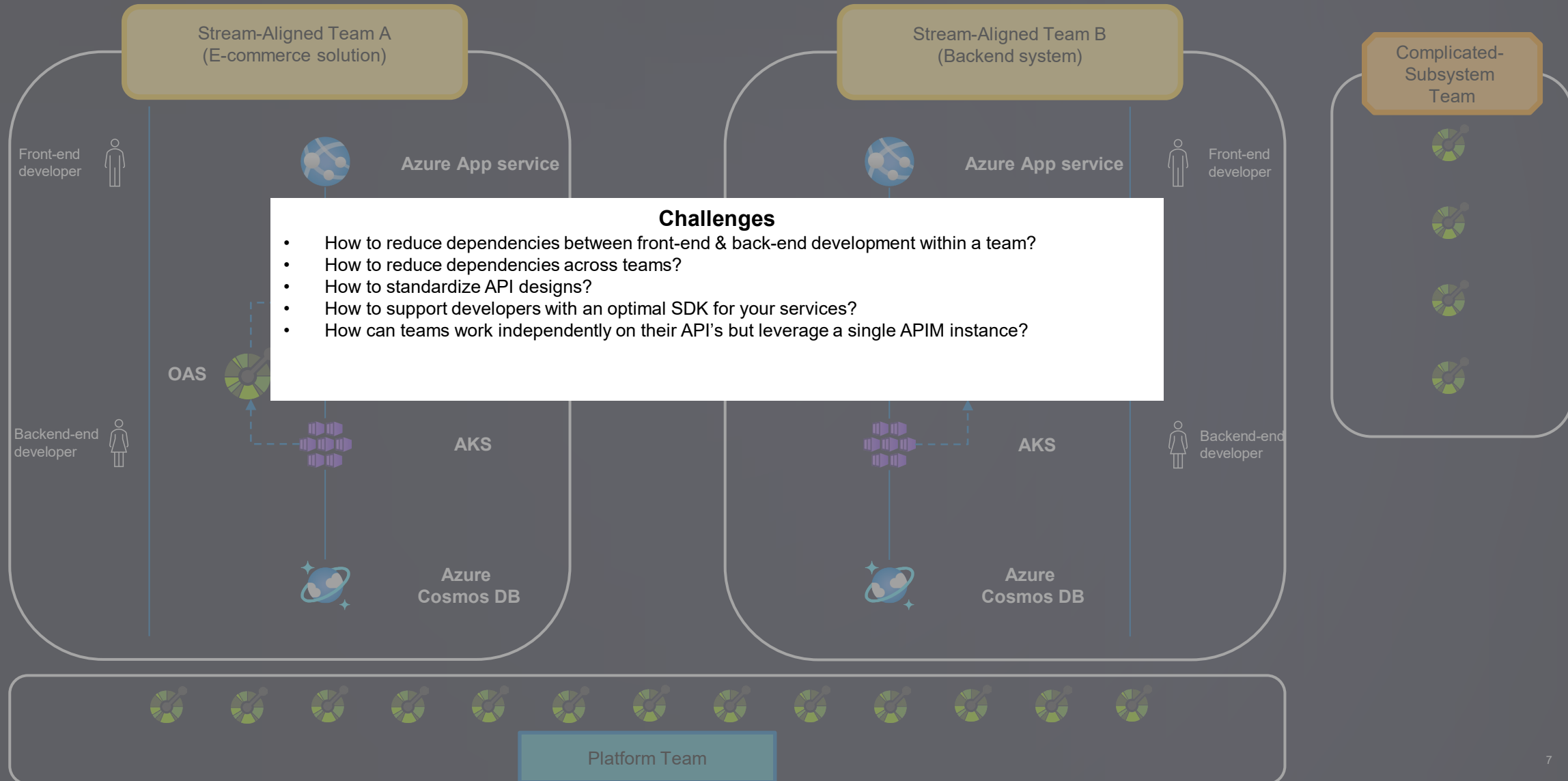

One APIM instance will be used for multiple stream-aligned teams

API usage across the organization

API usage across the organization

Stream-Aligned Team A
(E-commerce solution)

Stream-Aligned Team B
(Backend system)

Complicated-
Subsystem
Team

Front-end
developer

Azure App service

Azure App service

Front-end
developer

Challenges
- How to reduce dependencies between front-end & back-end development within a team?
- How to reduce dependencies across teams?
- How to standardize API designs?
- How to support developers with an optimal SDK for your services?
- How can teams work independently on their API's but leverage a single APIM instance?

OAS

Backend-end
developer

AKS

AKS

Backend-end
developer

Azure
Cosmos DB

Azure
Cosmos DB

Platform Team

How to reduce dependencies between front-end & back-end development within a team?
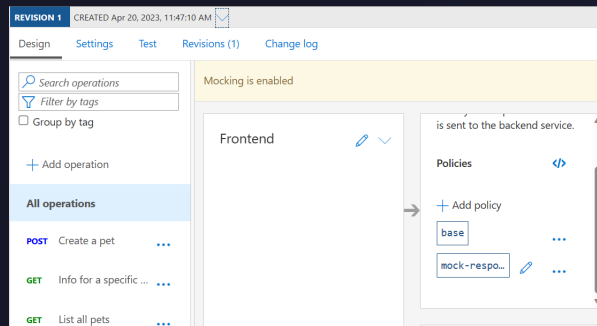
1. Apply versioning to your API's
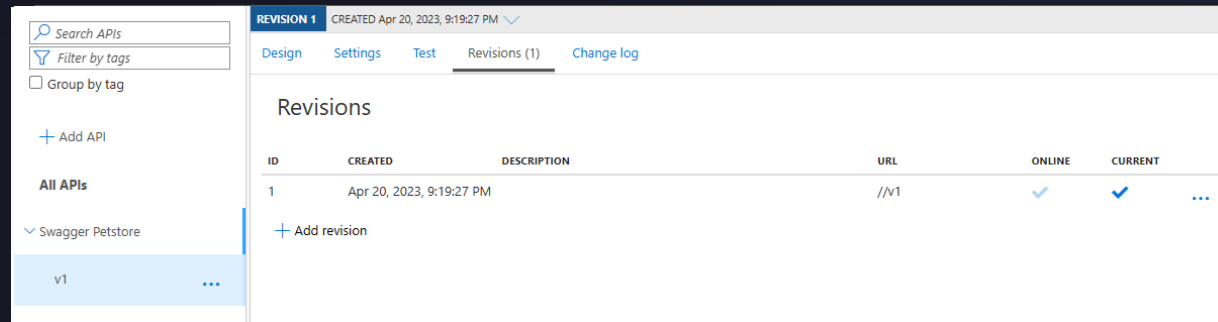
2. Start with API-first → Import your API definition
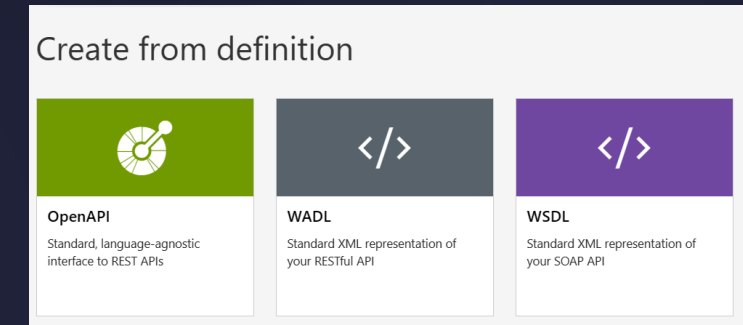
3. Enable mocking

How to standardize API designs?

Educate

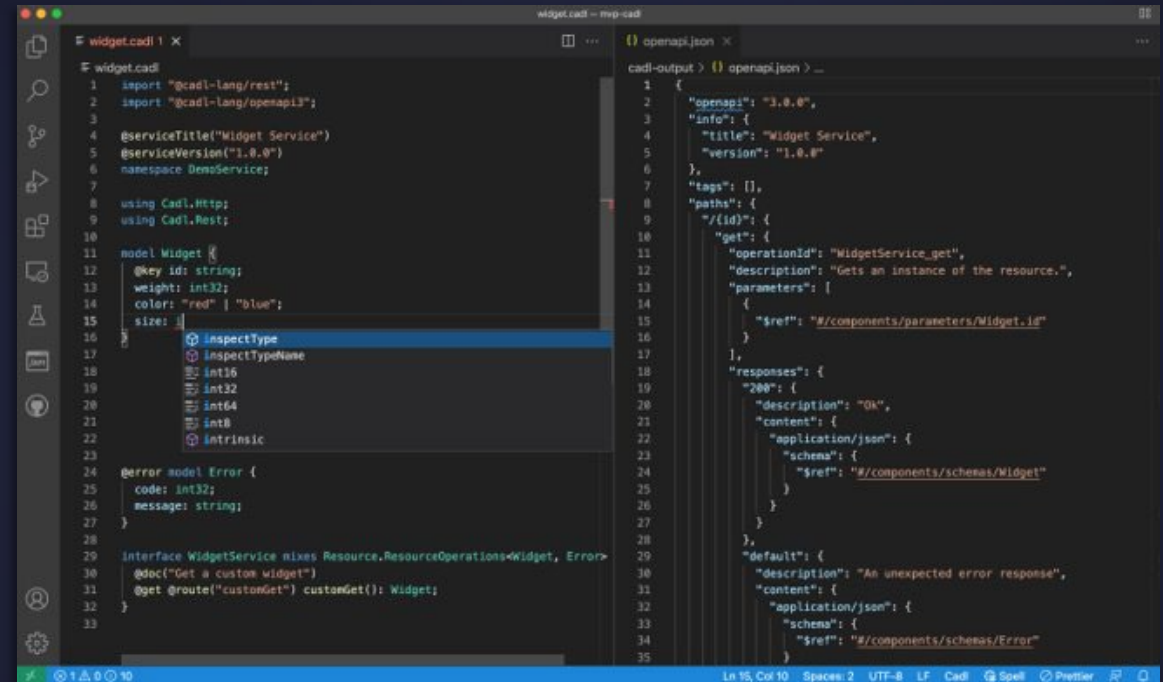https://aka.ms/azapi/specs

Enable

Empower

How to standardize API designs?

- A new language for *designing* APIs

- Supports libraries that *implement* our guidelines

- Promotes *reuse* through code sharing

- IntelliSense *tooling* support

- *Simplified* authoring experience for Open API

- Produces Open API *2.x & 3.x*

- Enables *version-aware* API development

microsoft/typespec (github.com)

How to support developers with an optimal SDK for your services?

Kiota is a command line tool for generating an API client to call any OpenAPI described API you are interested in. Supported languages are:

- C#
- Go
- Java
- Php
- Python
- Ruby
- Shell
- Swift
- Typescript / Javascript

How can teams work independently on their API's but leverage a single APIM instance?
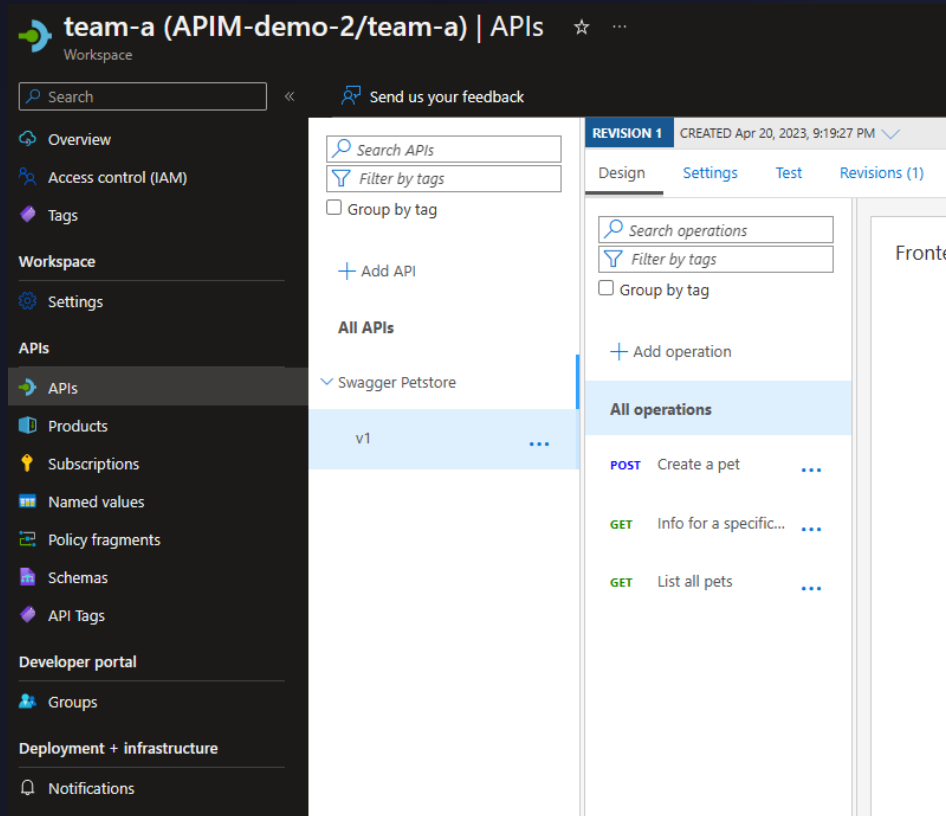
In API Management, *workspaces* allow decentralized API development teams to manage and productize their own APIs, while a central API platform team maintains the API Management infrastructure. Each workspace contains APIs, products, subscriptions, and related entities that are accessible only to the workspace collaborators. Access is controlled through Azure role-based access control (RBAC).

Demo

1. Create an OAS with TypeSpec

2. Upload OAS in workspace and enable mocking

3. Generate SDK with Kiota