Designing microservices

Masashi Narumoto
Principal PM Lead
AzureCAT patterns&practices

Agenda

- What is microservices?
- Benefits / Challenges
- Designing microservices
- Migrating to microservices

What is microservices?

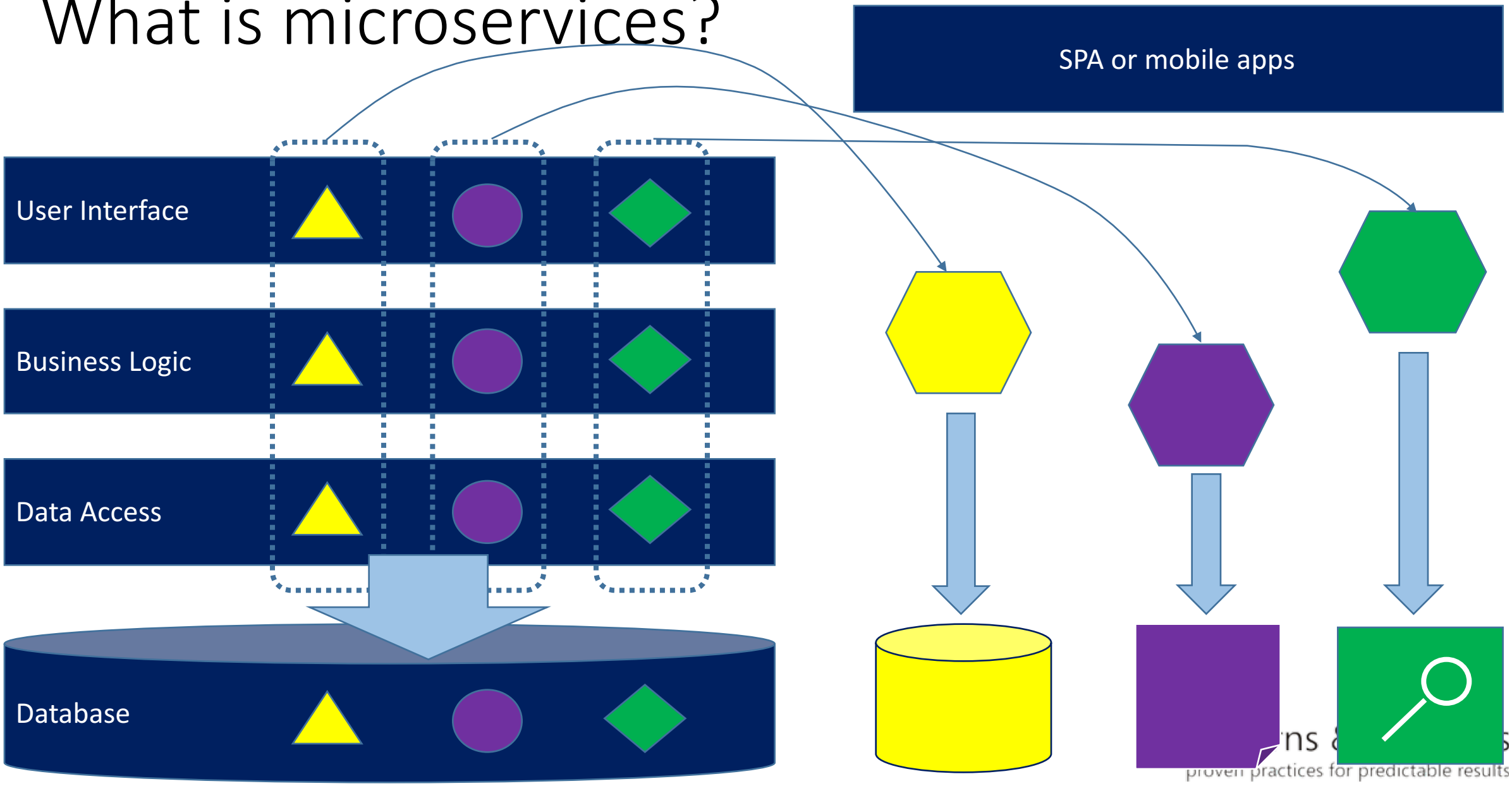
Loosely coupled services that work together, modeled around a business domain

- Sam Newman

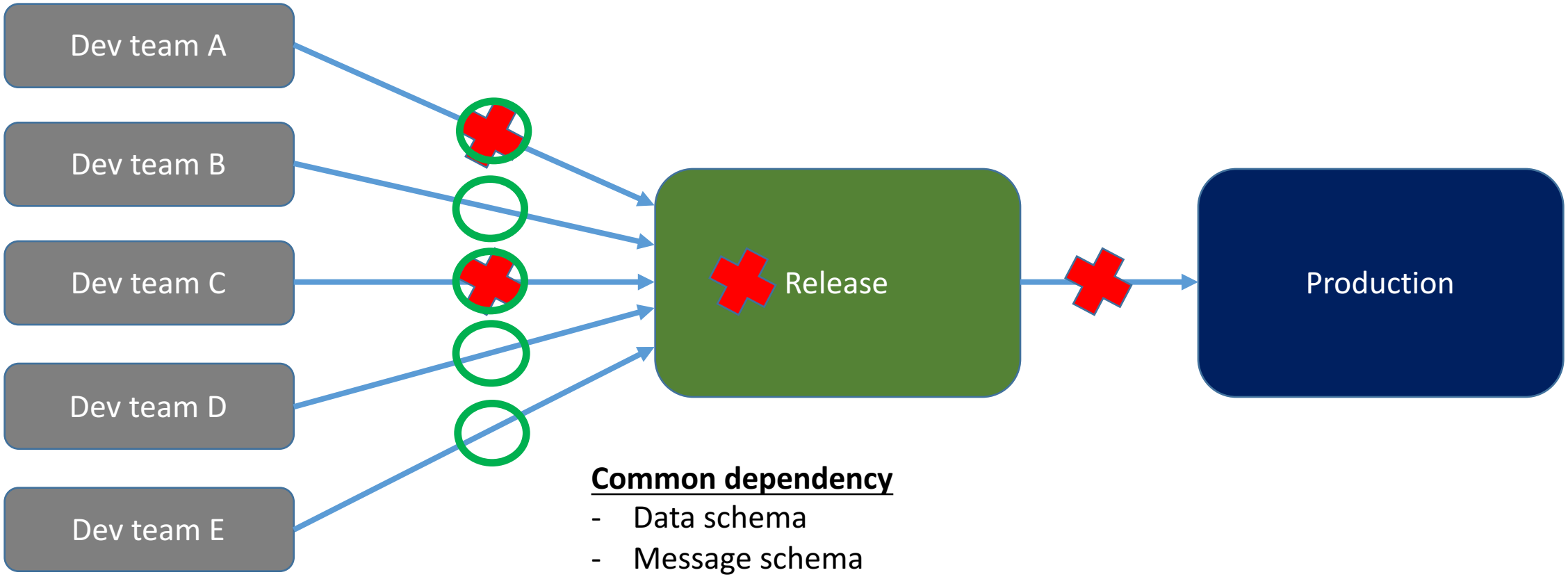
Small autonomous services that work together

- Adrian Cockcroft-

What is microservices?



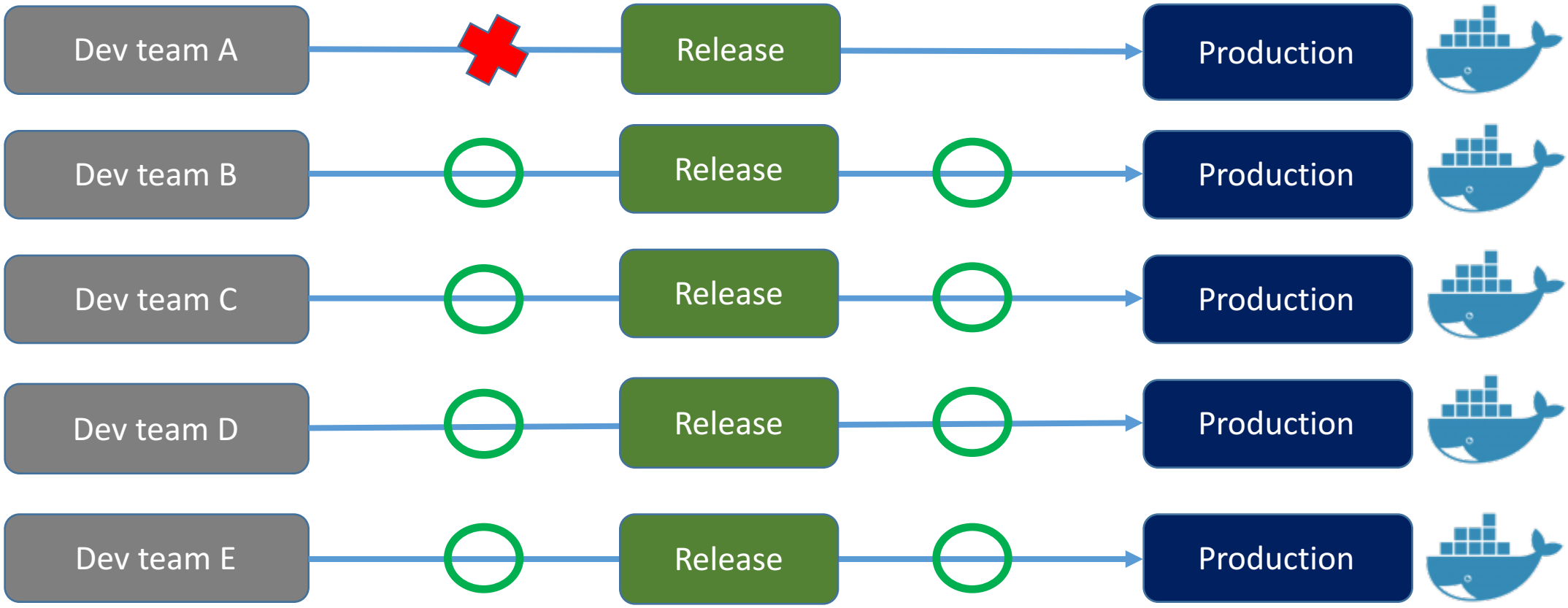
Why microservices? Why not monolith?



Common dependency

- Data schema
- Message schema
- Leaking service internals via API
- Framework/Library version
- Shared component

Why microservices? Why not monolith?

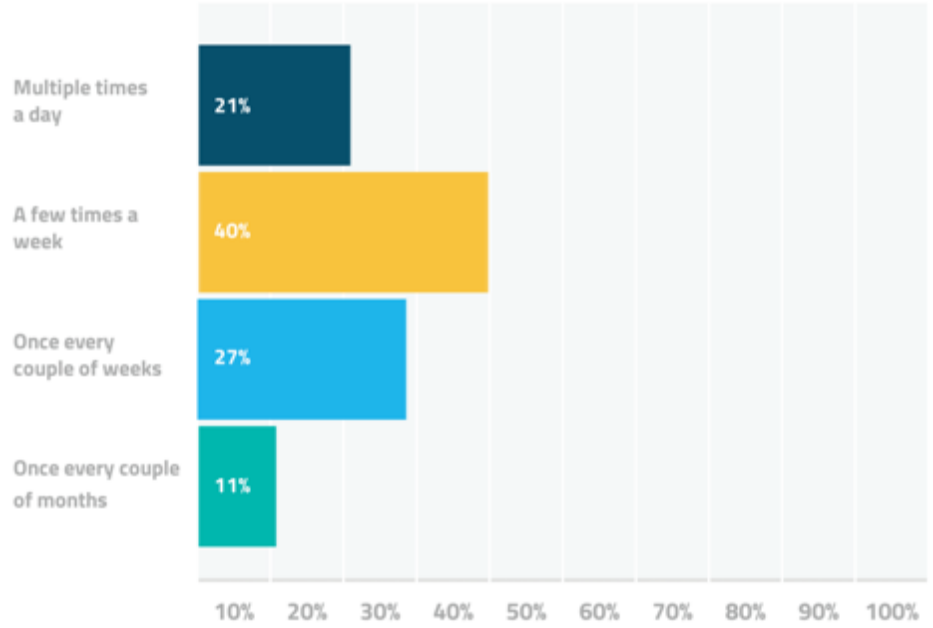


.
.
.

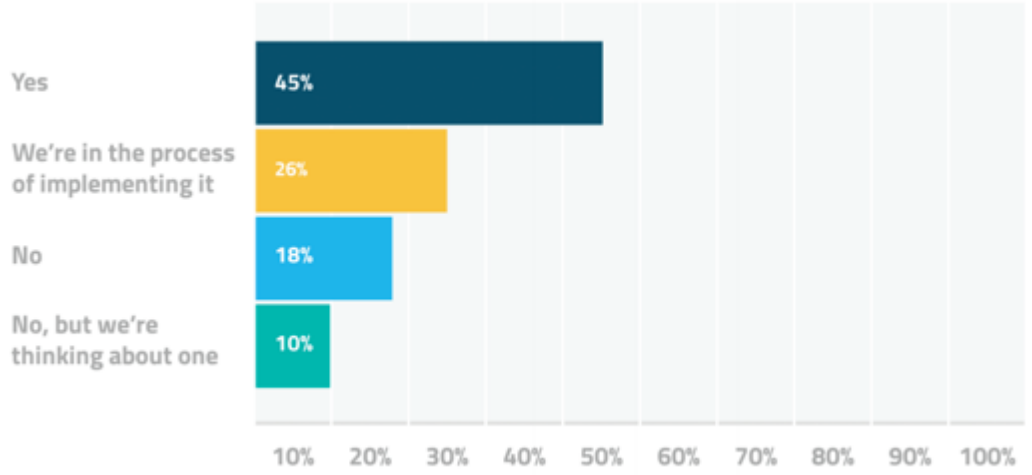
Integration, E2E test?

DevOps practices

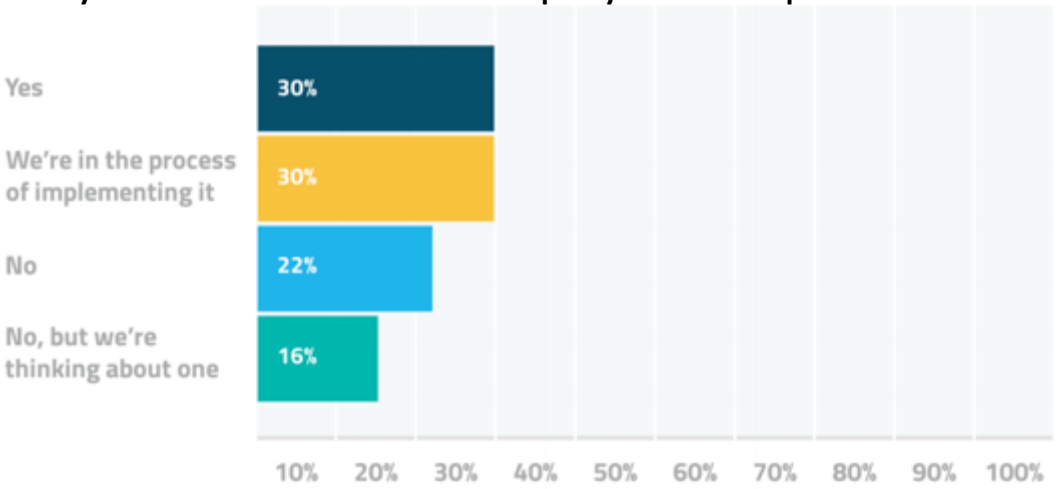
How frequently do you deploy code?



Do you have continuous integration in place?



Do you have continuous deployment in place?



Source: 2016 DevOps pulse

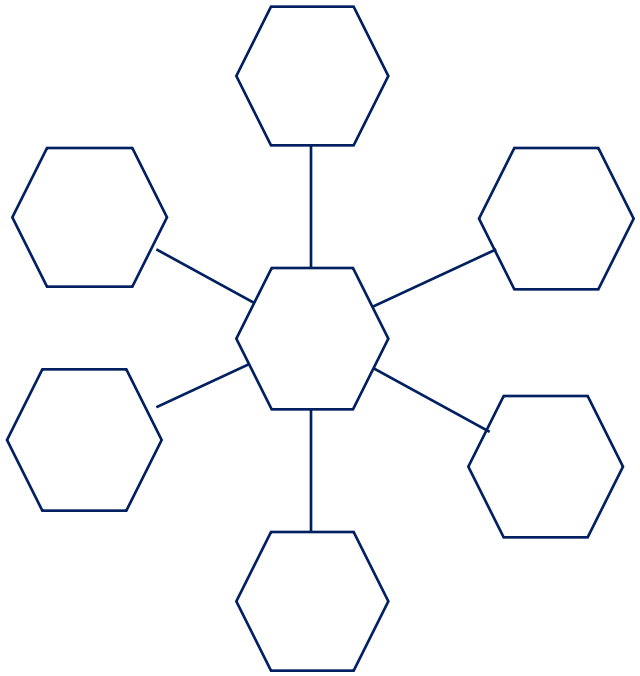
patterns & practices
proven practices for predictable results

microservices – case studies

patterns & practices
proven practices for predictable results

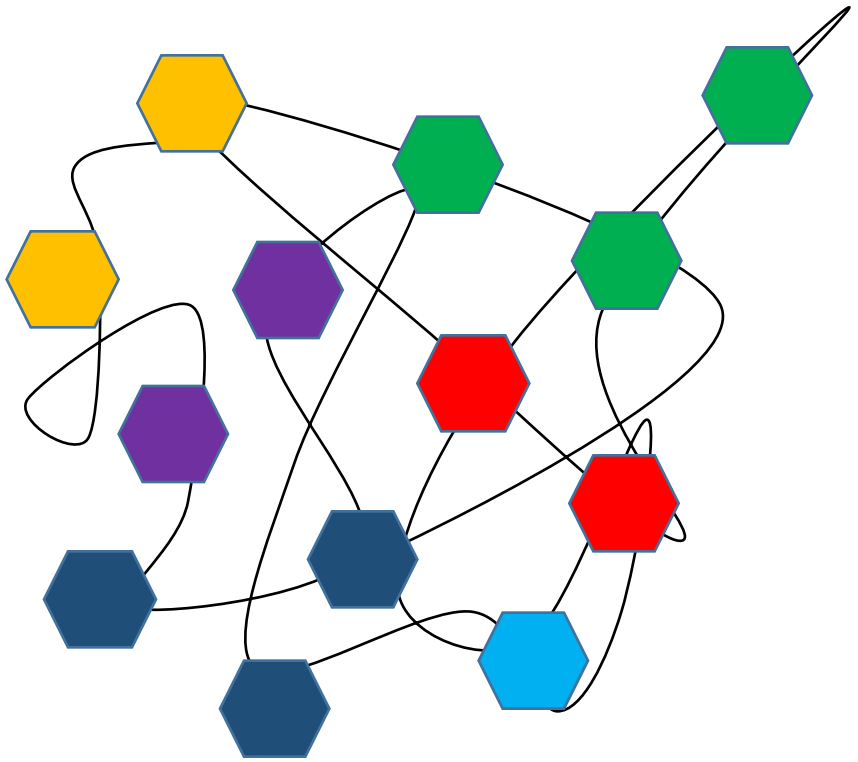
microservices - Benefits

- **Continuous innovation**
- Independent deployments
- Technology diversity
- Small focused team
- Separate scalability/availability
- Fault isolation

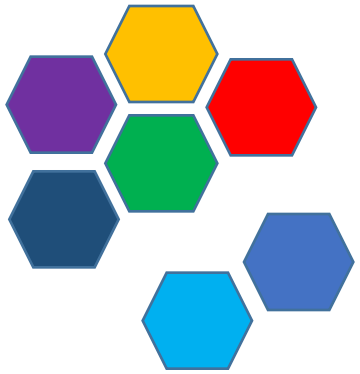


microservices - Challenges

- Complexity
- Network congestion
- Data integrity/consistency
- Integration and versioning
- Testing
- Reliability
- Service discovery and routing
- Monitoring and logging



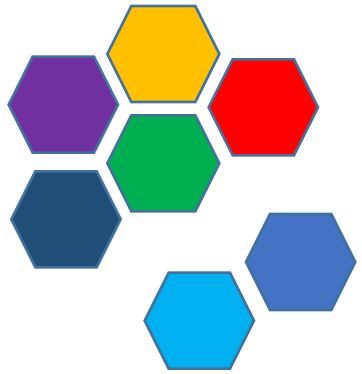
microservices - Principles



- Model services around a business domain
- Make each service independently deployable
- Decentralize all things
- Hide implementation details
- Data is private to its service
- Automate DevOps tasks
- Isolate failure

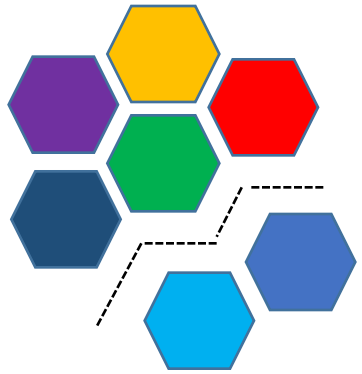
Less dependency more isolation!!

Designing microservices



- Service boundary
  - Granularity
- Gateway
  - Offloading, Aggregation, Routing
- Inter service communication
  - Sync/Async, Protocol/Serialization, Messaging
- Data management
  - Integrity/Consistency
- Distributed transactions
  - Dealing with partial failure
- Monitoring
  - Sidecar, Distributed tracing

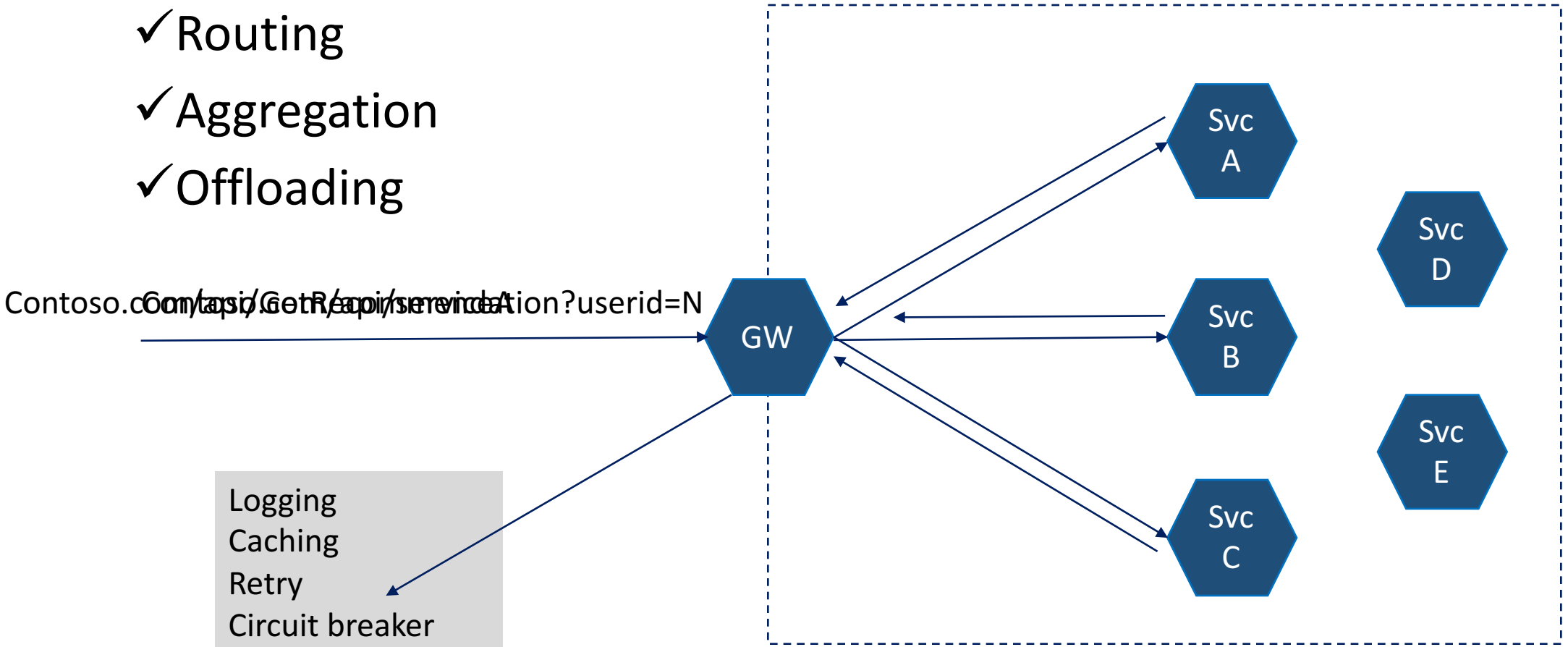
Finding service boundary



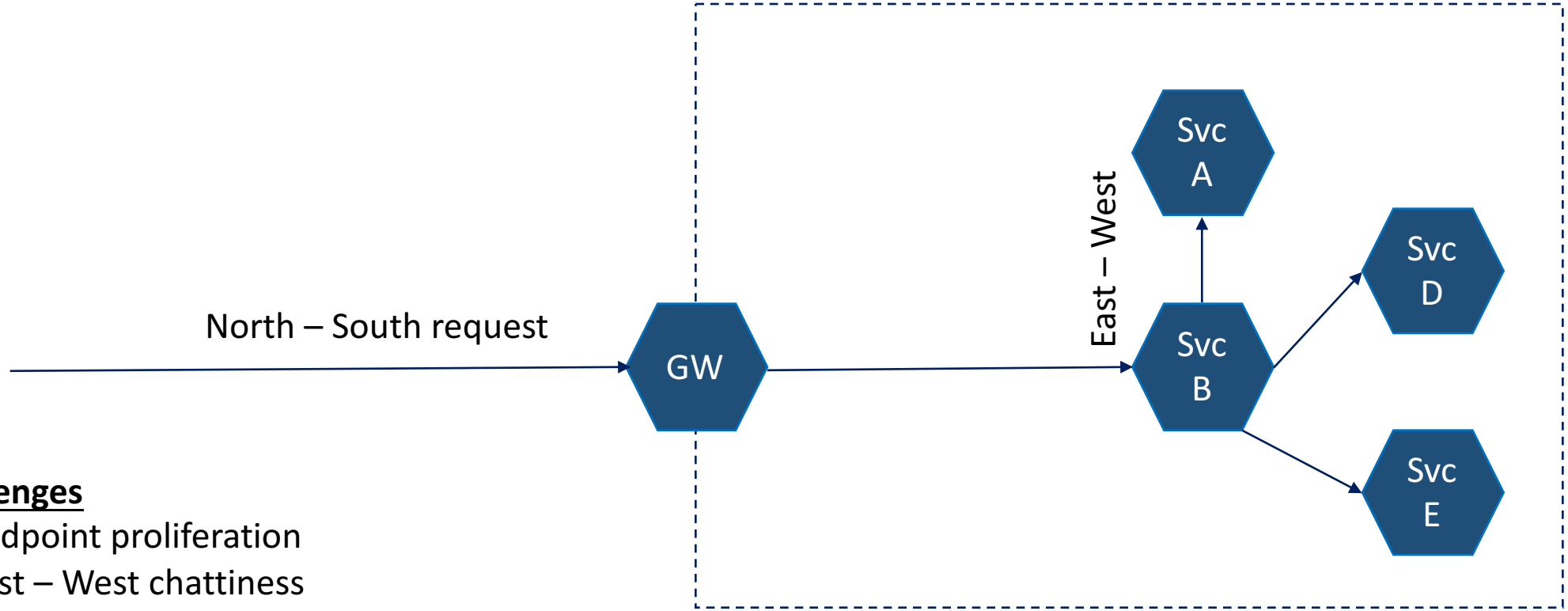
- Start with bounded context
- Further breakdown per non-functional requirements
- Vertical decomposition rather than horizontal (layers)
- Also consider
  - Rate of change
  - Technology used
  - Communication overhead
  - Splitting data is challenge due to consistency issues
- Refactoring across boundary is an extremely expensive operation

API gateway

- ✓Routing
- ✓Aggregation
- ✓Offloading



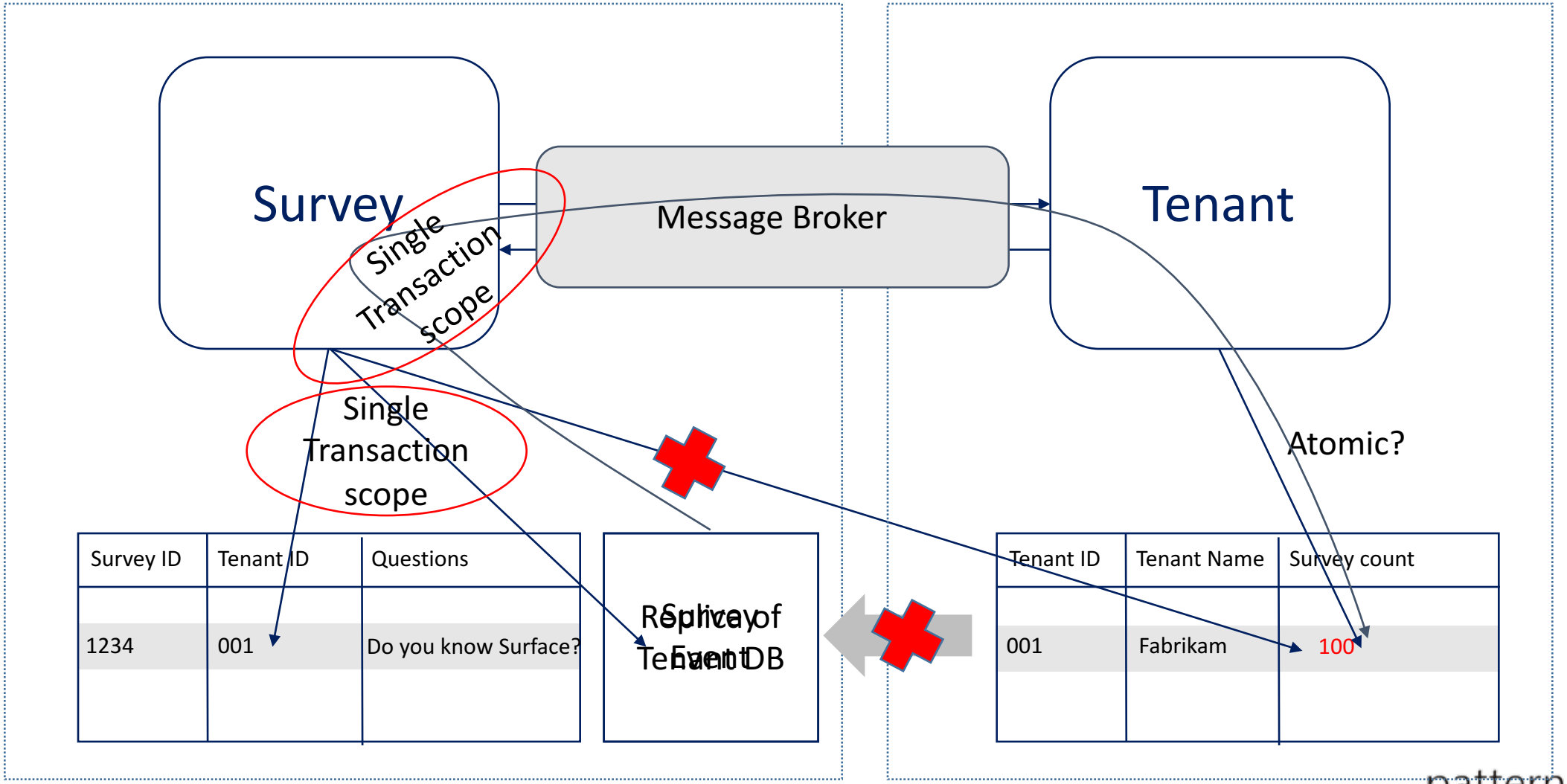
Inter service communication



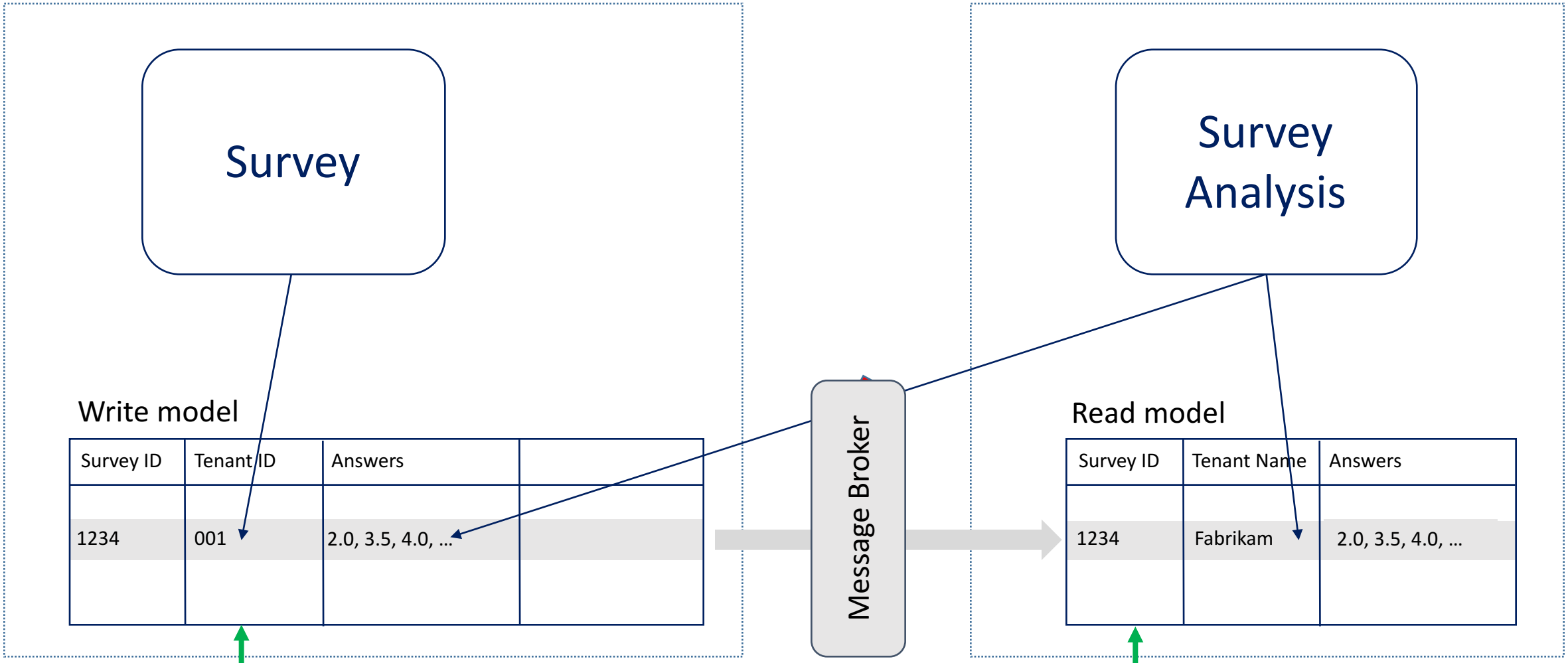
Challenges

- Endpoint proliferation
- East – West chattiness
- Overhead by serialization
- Different svc lifecycle requires decoupling
- Versioning
- IP masquerading

Data integrity/consistency

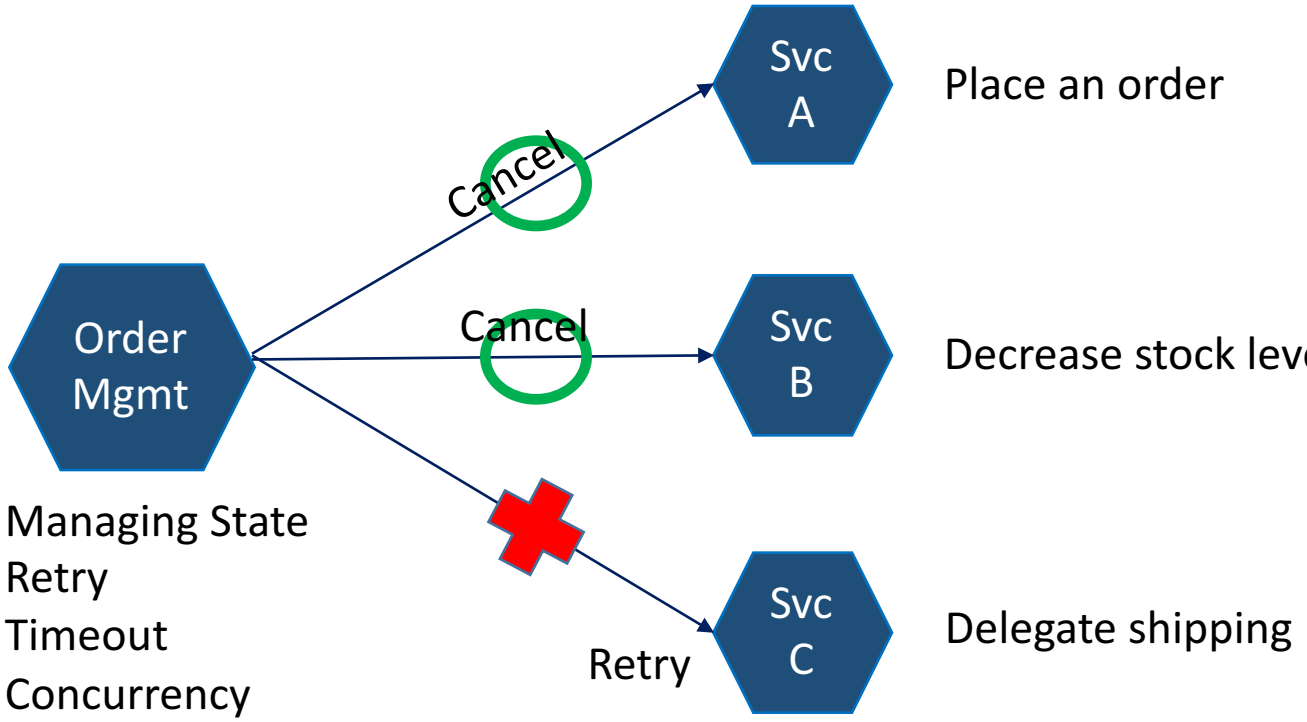


Decoupling data by CQRS



Reversible workflow by Sagas

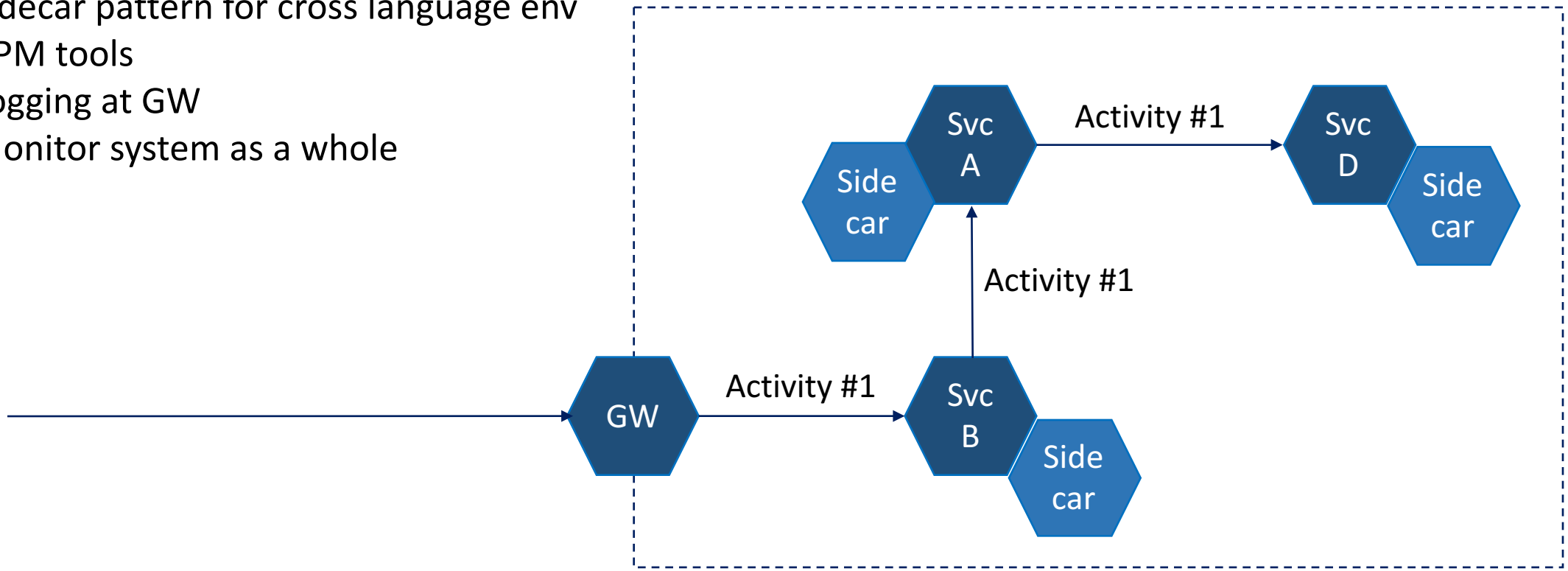
- Sagas is long running transactions that can be written as a sequence of transactions that can be interleaved with other transactions.
  - Hector Garcia Molina , et al. 1987-



scheduler agent supervisor pattern

Monitoring microservices

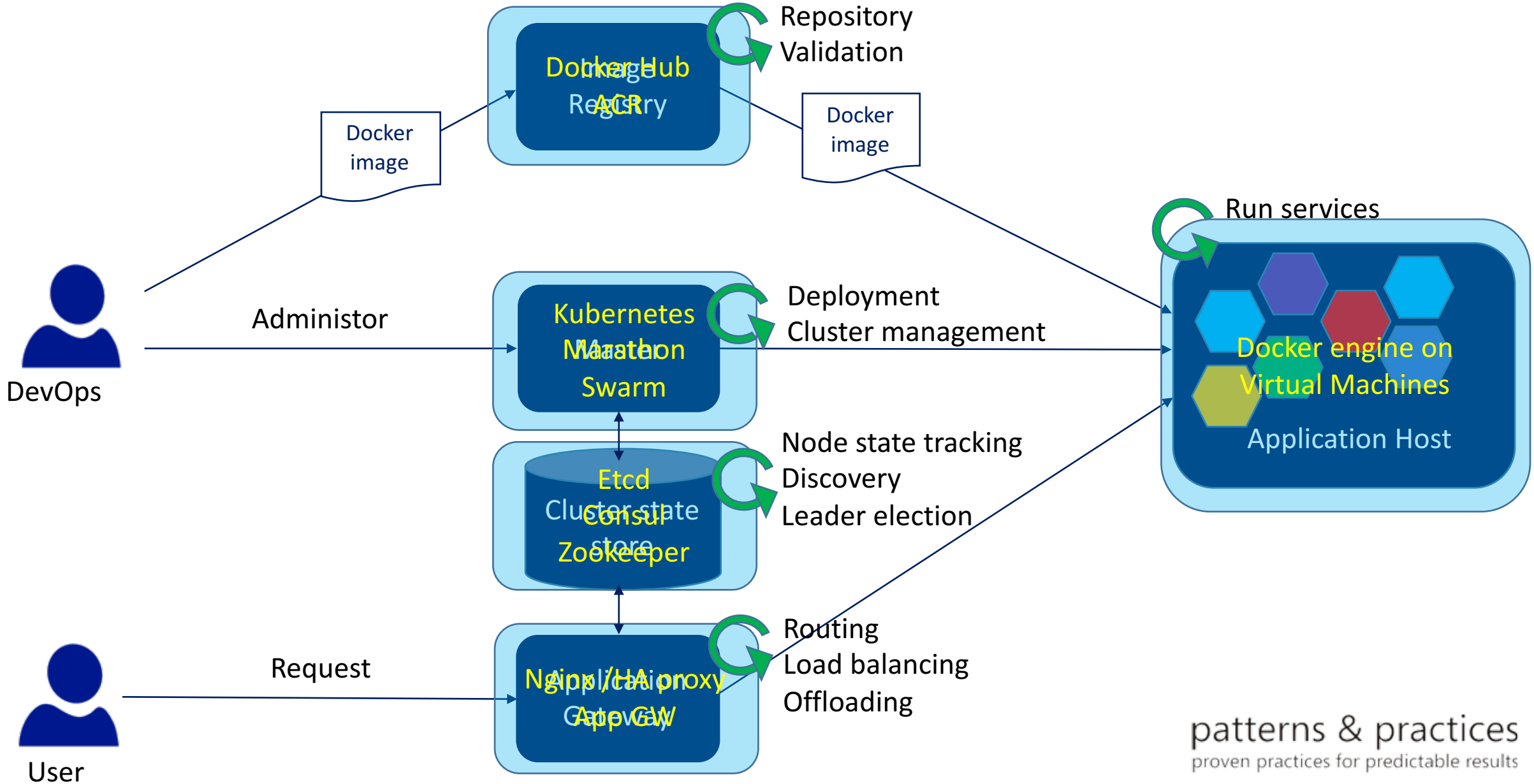
- Correlating distributed transactions
- Sidecar pattern for cross language env
- APM tools
- Logging at GW
- Monitor system as a whole



Options to implement microservices on Azure

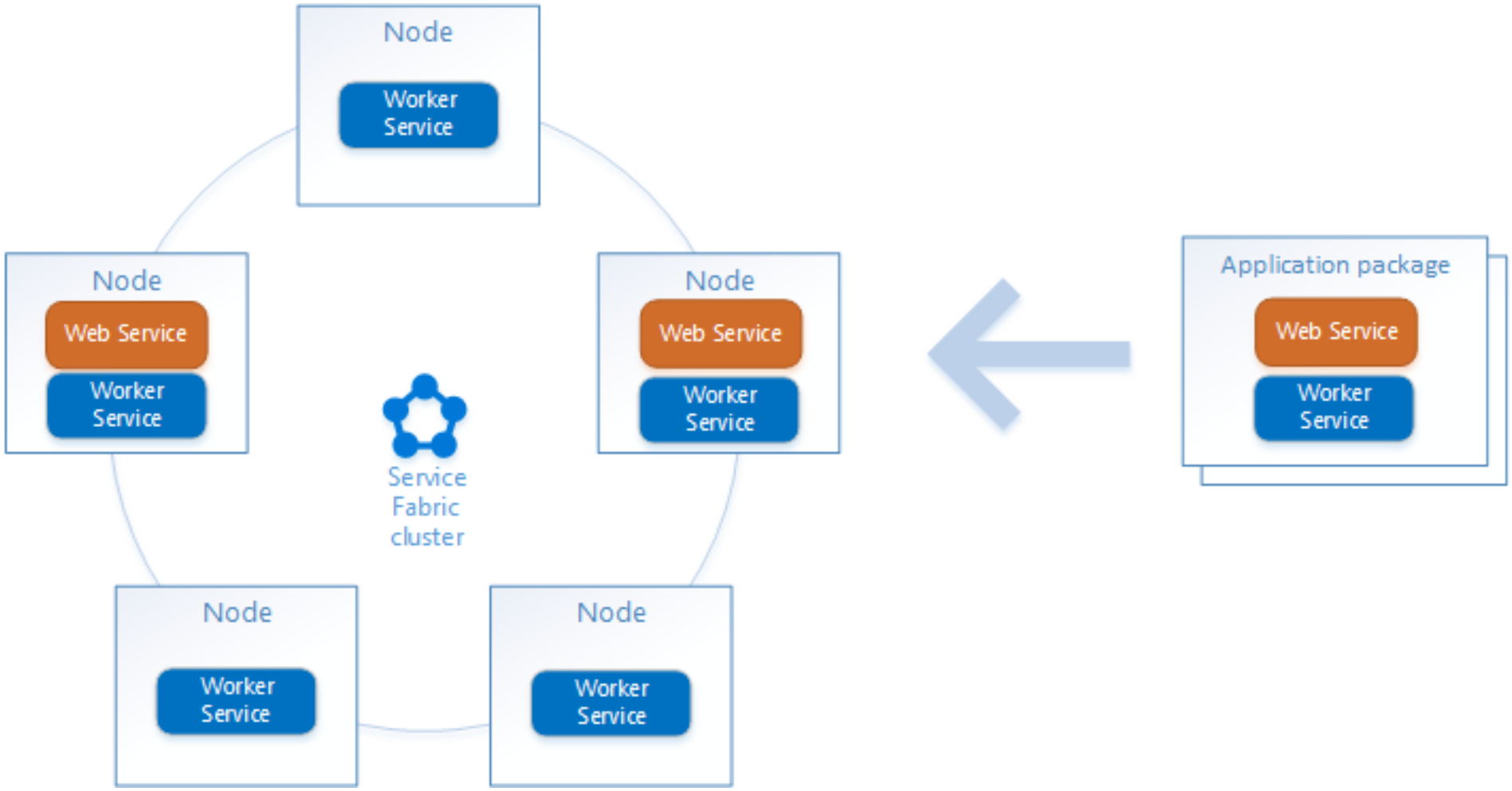
- Service Fabric
- Azure Container Service
- Azure functions
- Docker cloud (supports Azure)
- Docker on Virtual Machine
- App service

Container and orchestration



patterns & practices
proven practices for predictable results

Azure Service Fabric



Migrating monolith to microservices

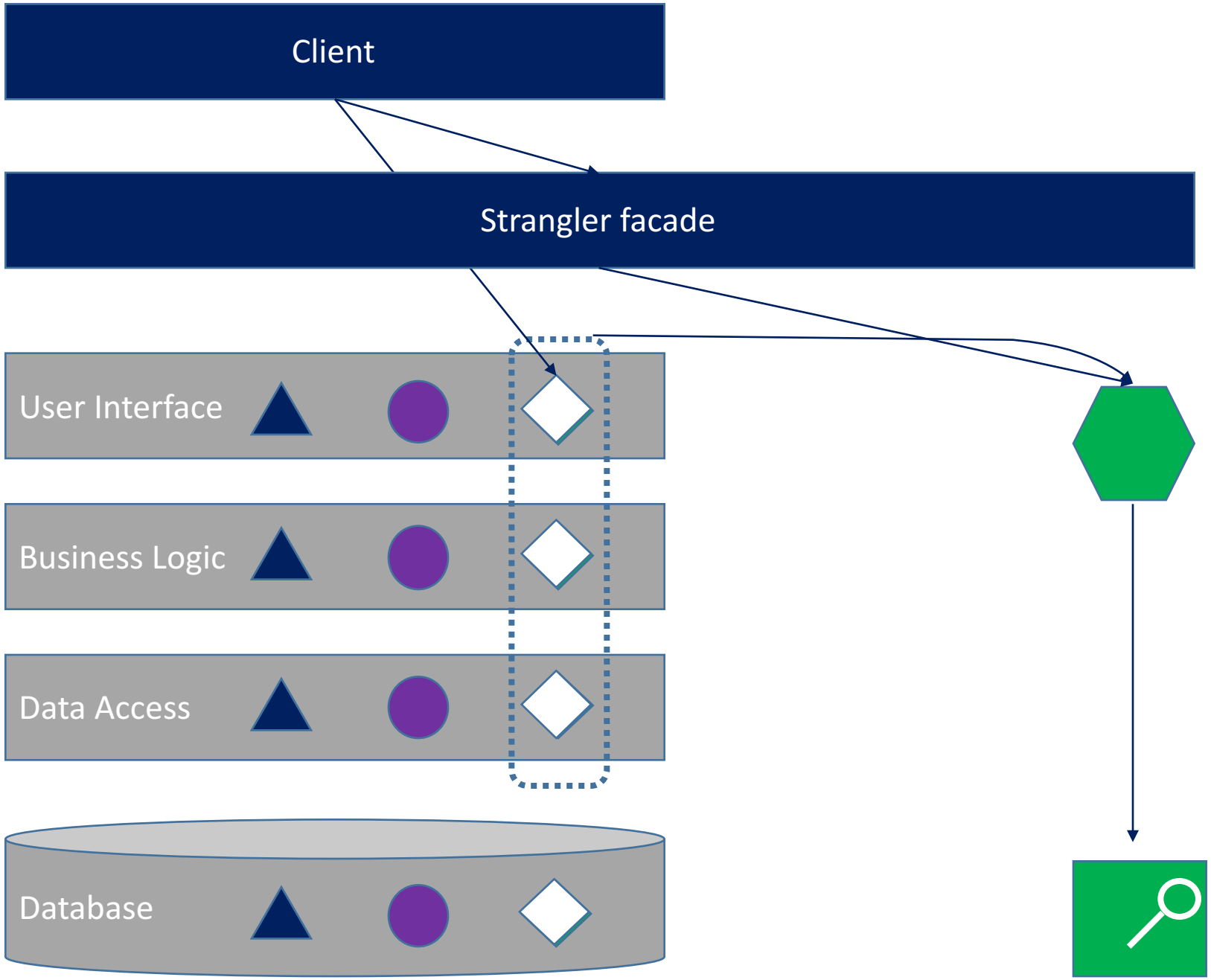
- Extract one service at a time
- Add glue code that takes care of dirty work
- Strangler / Anti-corruption layer in transition period

Monolith
(Big ball of mud)

Coarse grained
Somewhat decomposed

microservices

Strangler pattern


Client
Strangler facade
User Interface
Business Logic
Data Access
Database

Summary

- microservices is not something running on containers
- Choose microservices for continuous innovation
- Independent deployment is the key
- Incrementally migrate from monolith to MSA
- Use a few hosting options on Azure

Resources

- Microservices with Docker on Microsoft Azure (Trent Swanson, et al.)
- Building microservices (Sam Newman)
- Microservice architecture (Irakli Nadareishvili, et al.)
- https://www.nginx.com/blog/introduction-to-microservices/
- http://www.vinaysahni.com/best-practices-for-building-a-microservice-architecture
- http://www.grahamlea.com/2015/07/microservices-security-questions/
- Principles of Microservices by Sam Newman
- Adrian Cockcroft on InfoQ
- Service fabric training course on MVA