W4156

Architecture:
Microservice Style

Architecture Recap

“The Architecture” (shared understanding / artifact)	The “important” set of decisions throughout a software project that enable 1. Ensure/derisk the project/iteration is a success / meets key Q.A 2. Captured in some artifact which effectively communicates to stakeholders [1. Architecture, Architecture vs Design & Architecturally Significant 2. Views, Styles & Patterns*]
“Architecting” (the process)	The *process* of taking a project from idea/requirements through to architecture, construction and managing evolution of the architecture in response to evolving requirements (Requirements, [Quality Attributes], [Architecturally Significant Requirements], [Process], [Tradeoffs & Decisions], [Evolutionary Architecture], [Galls Law])
“Architect” (the responsibility)	Community organizer / engineering leader who leads this process, elevates the maturity of the team and may be responsible for trade-offs including challenging/balancing requirements.

[] everything in brackets is a key topic we will cover

* Our dear friends patterns re-appear at a higher level of abstraction for system elements and how they can relate to each other. This can be considered an *input* to the architecture process

Agenda

- Understand the microservices architectural pattern
- Factors that drove its creation
- Properties/Pros/Cons
- When to Use

“Why”

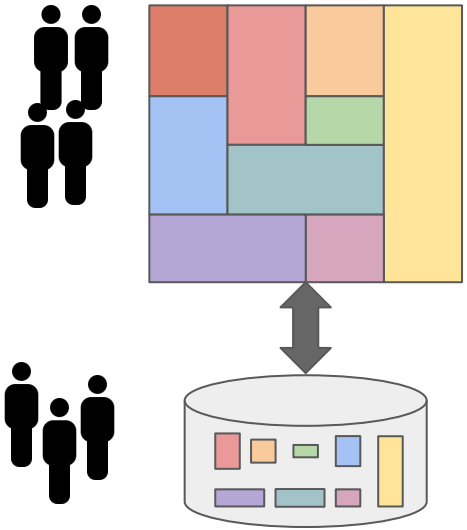
Monolithic

mon·o·lith
ˈmänəˌliTH/
noun
noun: **monolith**; plural noun: **monoliths**

1. A large single upright block of stone
2. A large impersonal political, corporate, or social structure regarded as intractably indivisible and uniform
3. A software application which is designed without modularity, is self contained and independent from other computing applications

Business function

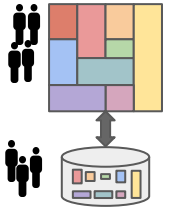
Monolithic



- Entire application a single process
- All of the functions share 1 database

Note - you will see a conflict in ‘monolith’ - often described as a single tier.
In the terms of microservices 3-tier applications described as monoliths or monolithic as an adjective (in comparison)

Monolithic



Pros and cons of this style at different scale/complexity?

Aspect	Smaller Scale	Larger Scale
Tight Coupling		
Encapsulation	<high/low and consequence?>	
Scalability		
Database Flexibility		
Organization		
Release		
Time to Market		

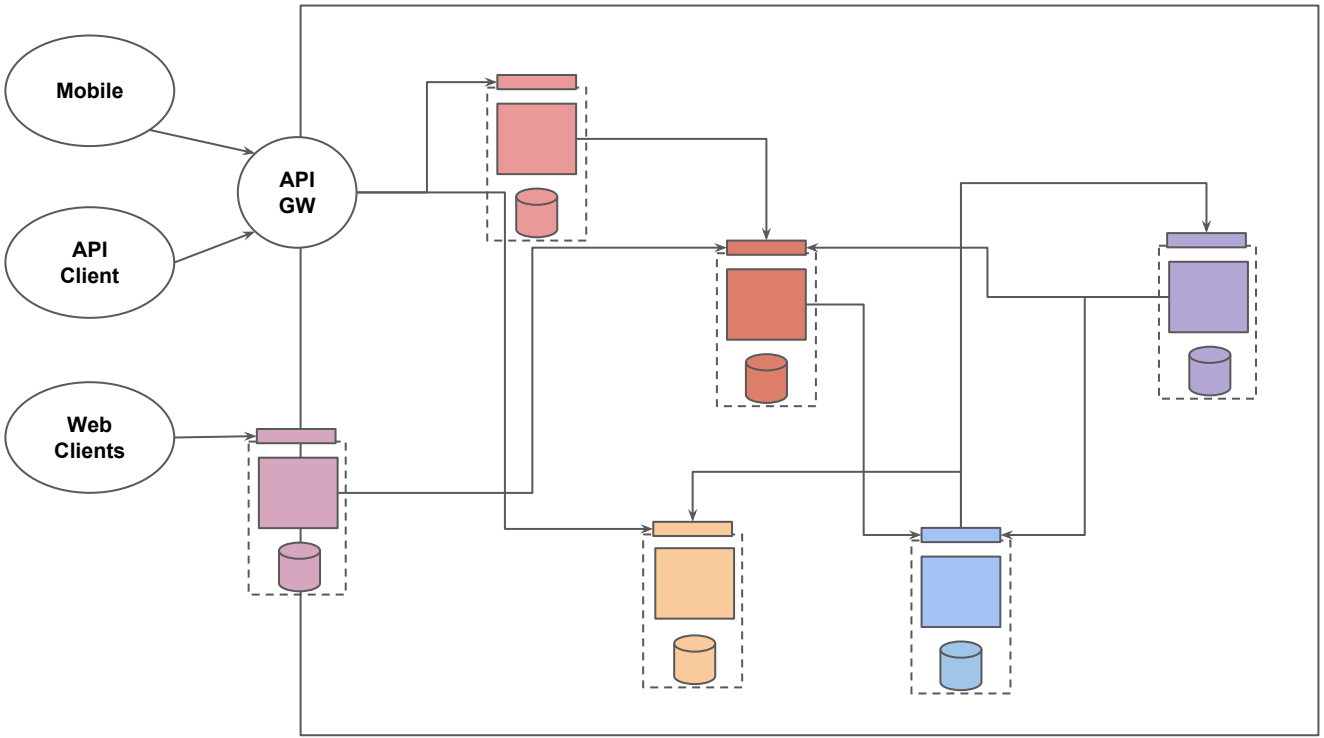
Note - you will see a conflict in ‘monolith’ - often described as a single tier.
In the terms of microservices 3-tier applications described as monoliths or monolithic as an adjective (in comparison)

“Enter Microservices”

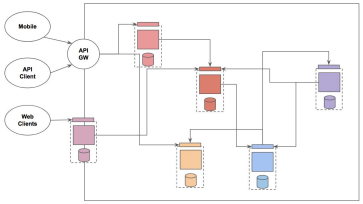
Microservices Architectural Style

“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. These **services are built around business capabilities** and **independently deployable** by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies”
[Fowler]

Microservice Architecture



In English

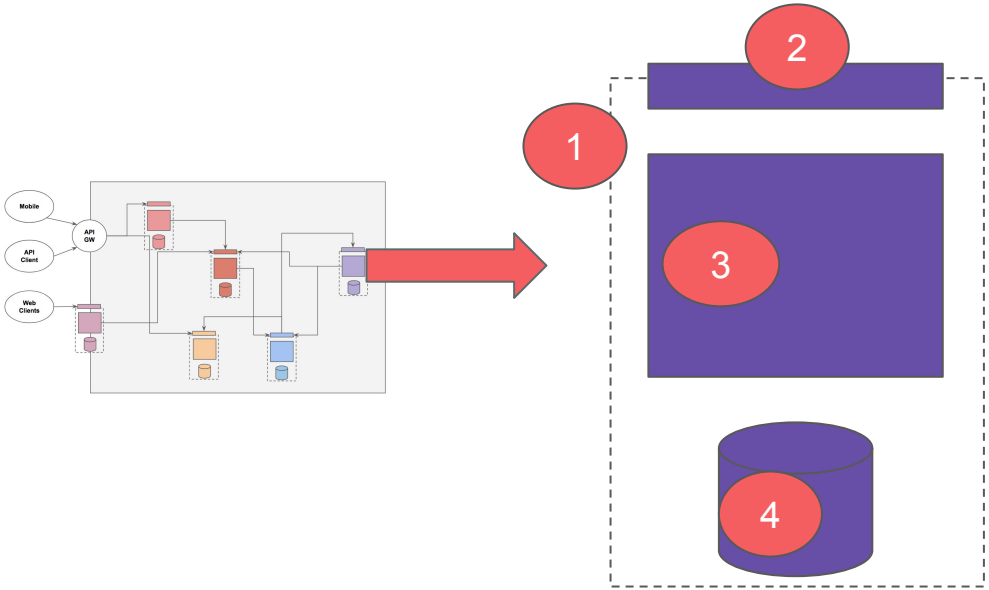


An approach to developing applications as several small services.
Where each service:

- Solves a particular *business capability*
- Is accessible (only) via a *well defined API* (often but not always RESTful)
- Owns its own data
- ‘Share nothing’ architecture between microservices
- Teams structures organized around services
- Features can be delivered independently (limitations apply)
- Scale cube model of scalability

(Connecting the dots: remember styles had elements, relations and constraints! What are the elements, relations and constraints!)

Individual Microservice

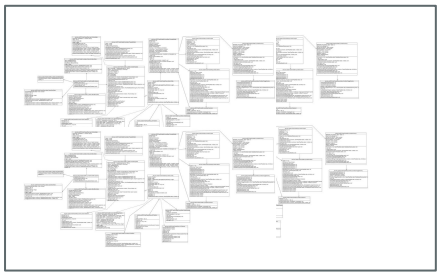


#	Description
1	Scope defined around a *business function / business capability*
2	Well defined API: only way of accessing *functionality* and *data*
3	Logic for business capability *encapsulated* within the microservice
4	Microservice owns its own data (and can choose its own store)

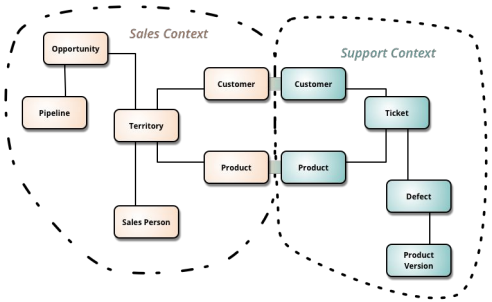
“How do I identify them?”

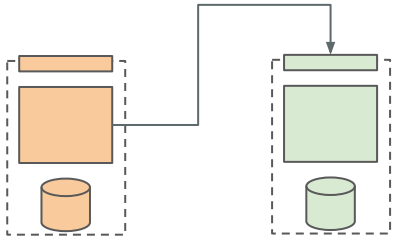
Domain Driven Design in One Slide

(very crudely OOAD for services)



DDD





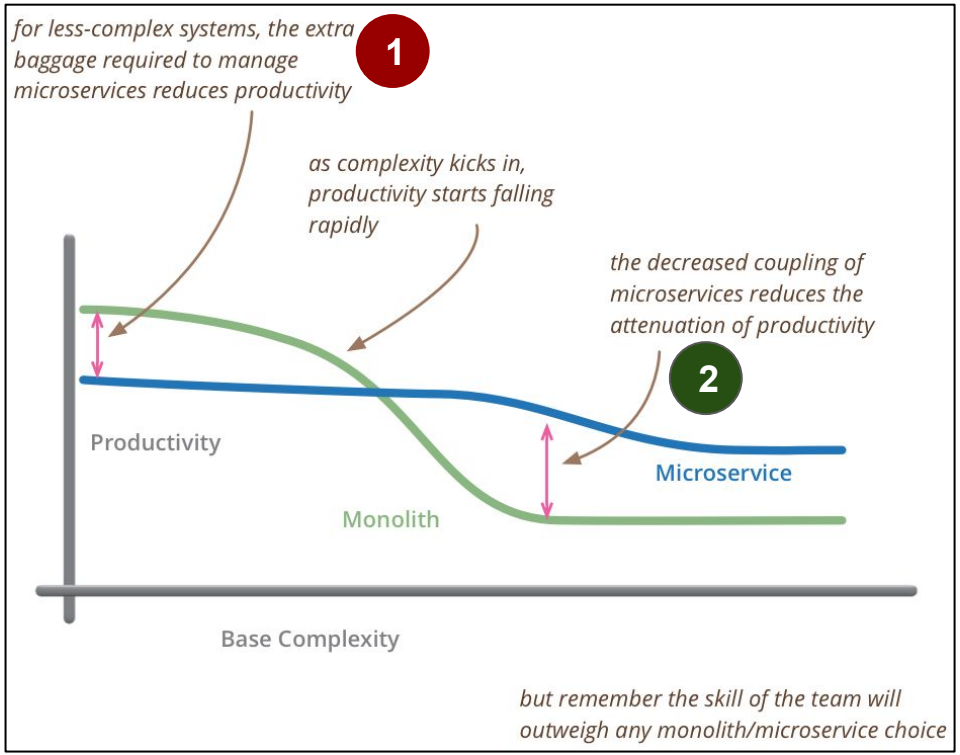
Complex system would have 1000s of ***classes***. Hard to keep coherent @ large complexity

Decompose/Encapsulate complexity within “bounded contexts”

Build services for each context

“When are microservices appropriate?”

Microservice Penalty



1 Hassle

- Decomposing into REST services
- Service discovery
- Eventual consistency
- Deployment management
- Debugging

2 Value

- Decoupling
- Independent evolution
- Flexible Scalability

https://martinfowler.com/microservices/#what

When to Use

“So my primary guideline would be don’t even consider microservices unless you have a system that’s too complex to manage as a monolith. The majority of software system should be build as a monolithic application. Do pay attention to good modularity within that monolith, but don’t try to separate into separate services” [Fowler]

https://martinfowler.com/bliki/MicroservicePremium.html

When to Use

Ok …. so what do I do if my system starts simple and grows more complex?

[Be wary of microservice envy ….]

Summary

- Microservices are an architectural style
- Characterized by small, independent services, providing a business capability, owning their own data, accessible via an API, owning their own data
- As a style they have both pros and cons and are not appropriate for low complexity (advantages not worth pain)
- As complexity grows an application may be *refactored* into microservices

Reading

Type	Link	Chapters
Required	https://martinfowler.com/microservices/#what	
Required	https://martinfowler.com/articles/microservices.html	
Required	http://microservices.io/patterns/microservices.html	
Highly Recommended	Implementing DDD	First few (for concept)
Recommended	Building Microservices	
Optional	Production Ready Microservices (Deployment pipeline)	
Optional	Introduction to Microservices	
Optional	https://en.wikipedia.org/wiki/Microservices	
FYI	http://www.mattstine.com/microservices/	