

COMS E6998: Microservices and Cloud Applications

Lecture 1: Introduction

Dr. Donald F. Ferguson
dff9@columbia.edu

© Donald F. Ferguson, 2017. All rights reserved.

Contents

Contents

- Introduction
 - A bit about your instructor.
 - Coursework and grading.
 - Logistics.
- Course Overview and Contents
 - Schedule and topics.
 - Cloud
 - Microservices
 - Cloud applications.
- An example “microservice, cloud application”
 - Demo.
 - Overall structure.
- 1st Assignment

Introduction

Your Instructor

A Little About Me

- Columbia through and through
 - BA (Columbia College) '82, M.S. '84 (Engineering), M.Phil. '84 (GSAS).
 - Ph.D. '87 (GSAS) – Thesis: “The Application of Microeconomics to the Design of Resource Allocation and Control Algorithms.”
 - Adjunct Professor at Columbia Univ: 2006, 2009, 2014-2017. 7 “Topics in CS” courses.
- 32 years of technology contribution and leadership.
 - VP, IBM Fellow: IBM Research, IBM Software Group.
 - Microsoft Technical Fellow.
 - Distinguished Engineer, CTO, Executive VP, CA technologies.
 - VP, Sr. Fellow, CTO, Dell Software Group.
 - Co-Founder, CTO Seeka TV (www.seeka.tv).
- Personal
 - Two fantastic daughters; high school and one is a senior at Barnard.
 - Black belt in Kenpo Karate (<http://www.tracyskarateworldwide.com/about.html>, www.elitedefensivetactics.com)
 - 2LT in New York Guard (<http://dmna.ny.gov/nyg/>)
 - Conversational Spanish, learning Arabic
 - Hobbies: Bicycling, Astronomy



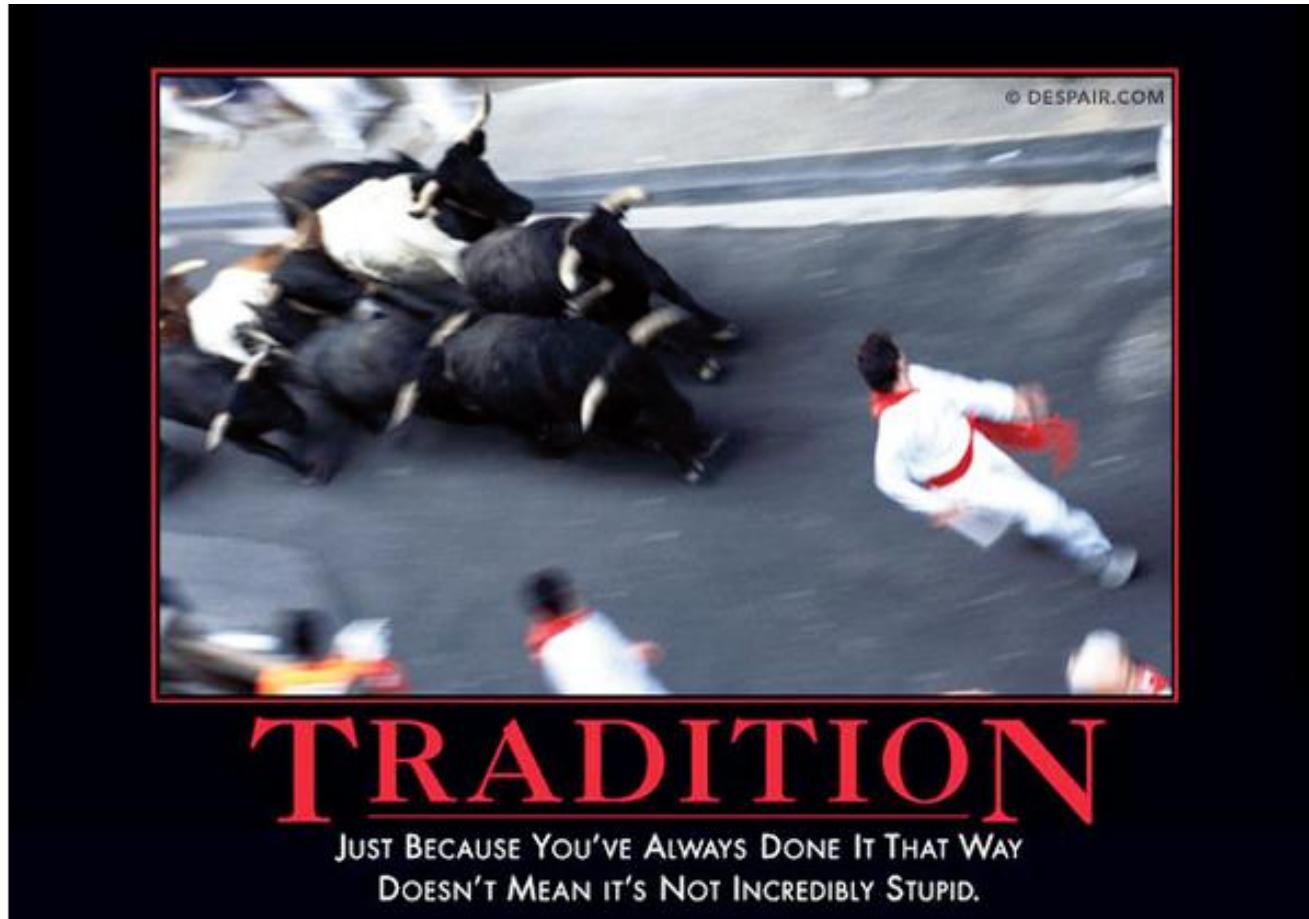
— Co-Founder, CTO SeeKA IV.

- Personal
 - Two daughters
 - Black belt in Kenpo Karate
 - 2LT in New York Guard (<http://dmna.ny.gov/nyg/>)
 - Conversation Spanish, learning Arabic
 - Hobbies: Bicycling, Astronomy

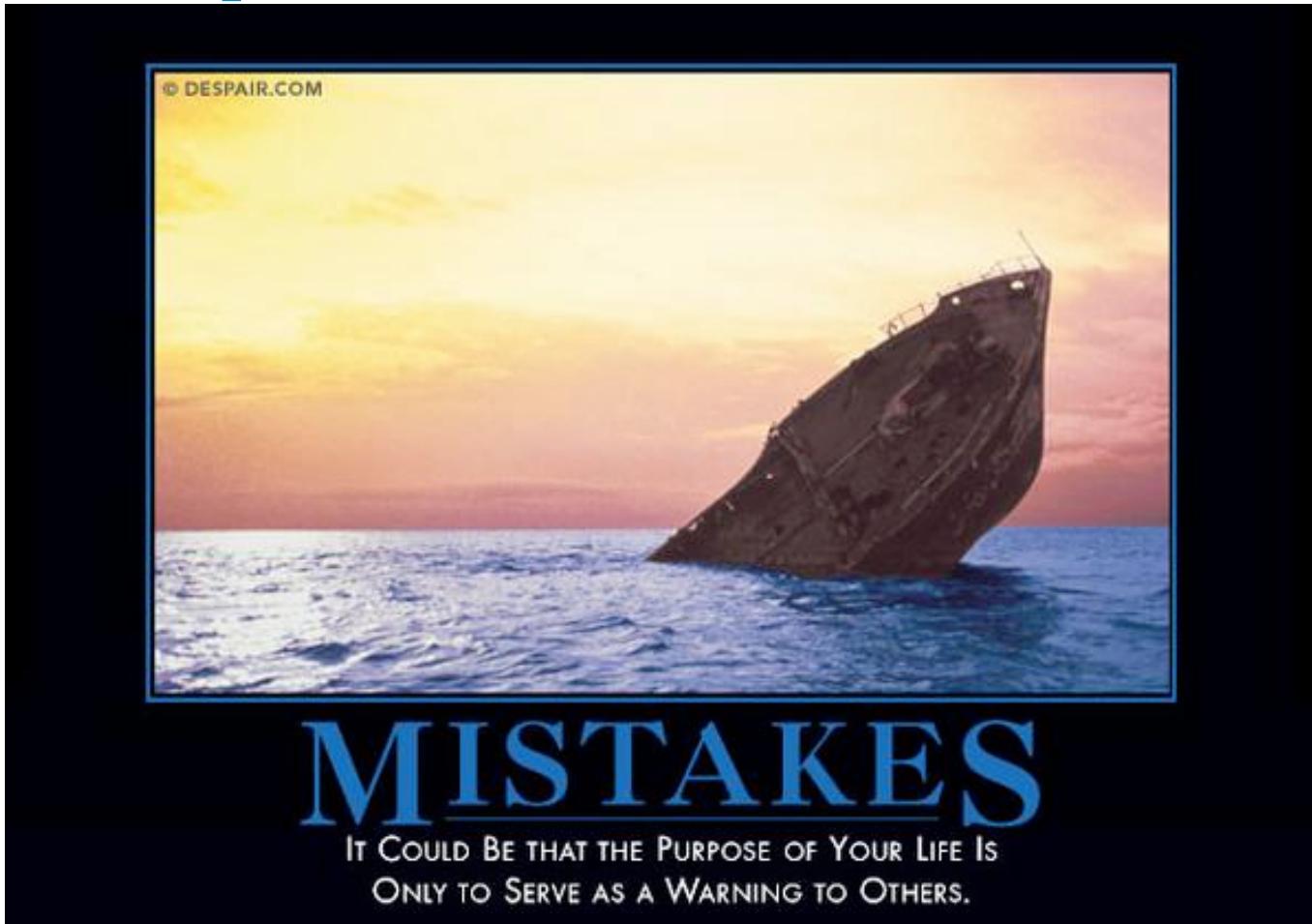
g), M.Phil. '84 (GSAS).



A Lot of Experience



A Lot of Experience



A Lot of Experience

I may not be able to teach you the right
way to do something,

but

I can show you plenty of
wrong ways to avoid.

Coursework Grading

Coursework and Grading

- No midterm, final or exams.
- Five 20 point projects, each of which extends previous projects.
 - Approximately every two weeks, with two weeks to complete.
 - Teams
 - Approximately 5 team members.
 - Each team has a designated *primary contact* and *team name*.
 - Each project has the following sub-deliverables, packaged in a zip file with a readme.
 - A set of user stories.
 - A short, simple architecture/design description with diagrams.
 - Demo
 - Application artefacts, e.g. code, DB schema/data, HTML, URL for testing, ...
 - There will be a simulated “project review,” which is a meeting with presentation, demo, Q&A, etc.
- In addition to the technology, I am trying to *mentor* you to learn how to be a successful technical professional. You will experience very basic concepts in
 - Product management.
 - Presentation and documentation skills.
 - Transforming vague requirements into something clear and implementable.
 - Q&A and discussion of alternatives.
 - Teamwork and team dynamics.

This will drive you crazy in the beginning.

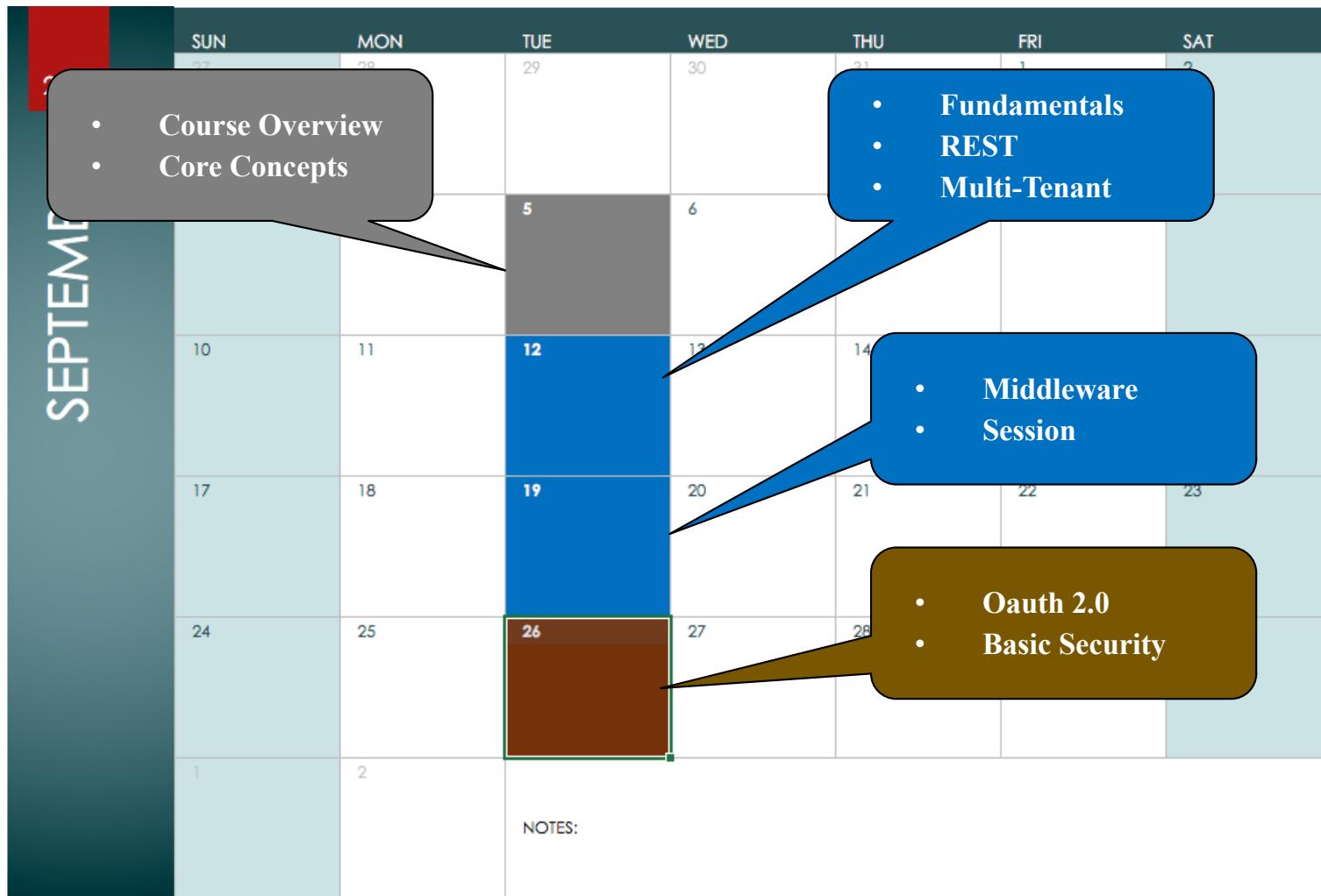
Logistics

Logistics

- Course material
 - *No textbooks.* Technology changes too quickly, and we will cover concepts and technologies from several disciplines.
 - Lecture and lecture notes will provide links to online information.
 - Web search is your first and best resource.
- CAs:
 - We have two excellent CAs, with whom I have previous worked.
 - Will introduce themselves via email, Piazza and other collaboration environments.
- Office hours
 - Professor: Tu/Th 10:15 to 12:00, or by appt (typically Google Hangout).
 - CAs: TBA
- Collaboration environments:
 - CourseWorks for submitting projects.
 - Piazza: Slides, discussion (piazza.com/columbia/fall2017/comse6998_014_2017_3topicsincomputerscience)
 - Slack: Good way to have short discussions, get ahold of me, instant messages, etc.
 - Team: (https://join.slack.com/t/dff-columbia/shared_invite/MjM1MjA5NjAwOTk2LTE1MDQyODExOTQtNzEzM2FjZmMzMw).
 - Join channel #e6998-2017
- We will use Amazon Web Services, but the concepts are general.
- **Questions and discussion are expected, appreciated and essential.**

Course Overview and Content

Course Overview



OCTOBER

2017

SUN	MON	TUE	WED	THU	FRI	SAT
1	2	3	4	5		
8	9	10	11	12		
15	16	17	18	19		
22	23	24	25	26		
29	30	31	1			
5	6		NOTES:			

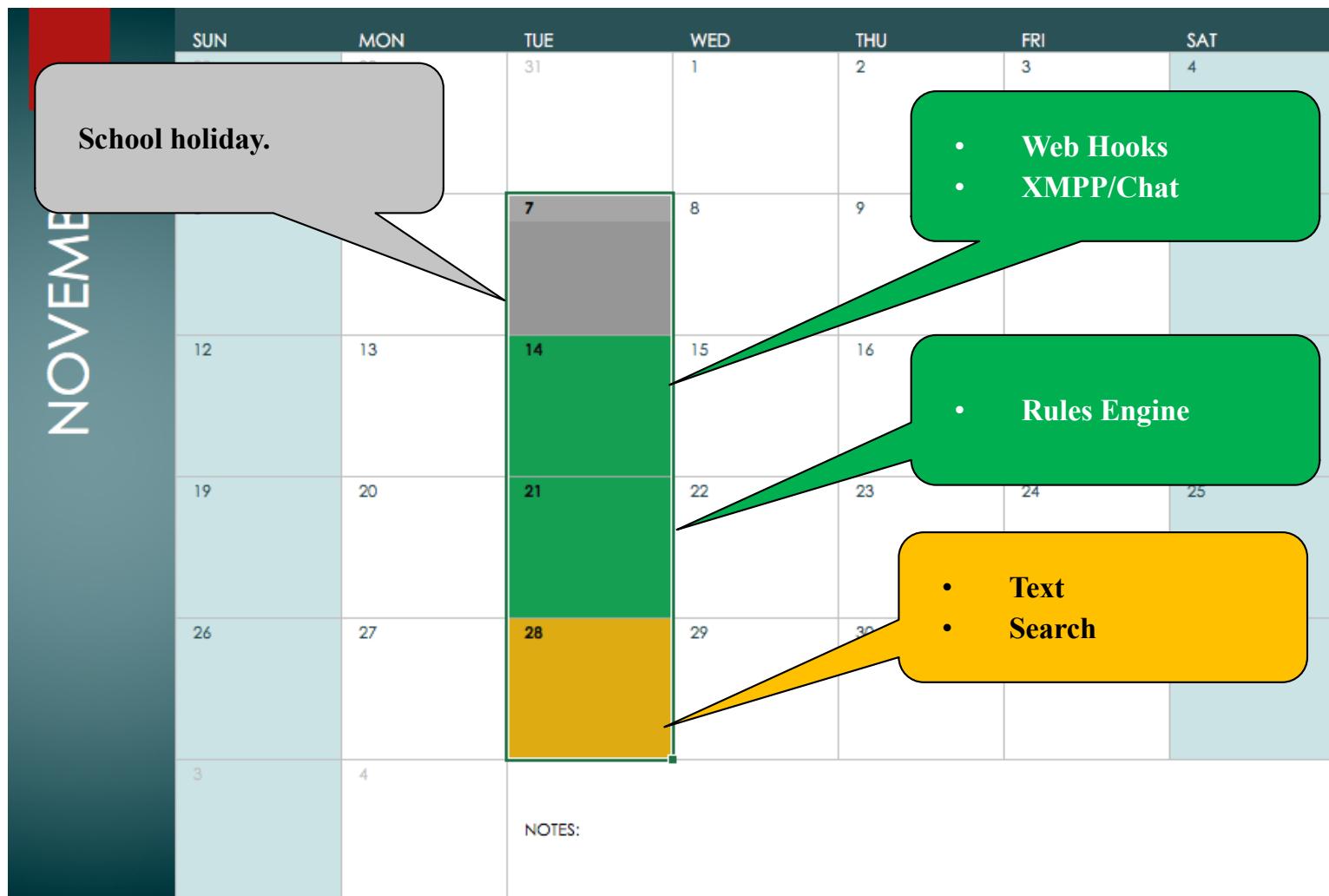
- Calling Cloud APIs
- Integration Patterns

- Pub/Sub
- Message Queues

- Serverless
- API Gateway

- Dynamo DB
- Neo4J
- Redis

- Data Pipeline
- Step Functions



2017		SUN	MON	TUE	WED	THU	FRI	SAT
		26	27	28	29	30	1	2
		3	4	5	6	7	8	9
		10	11	12	13	14	15	16
		17	18	19	20	21	22	23
		24	25	26	27	28		
		31	1					
NOTES:								

- Overflow
- Discussion

Comments

- This course is about building applications
 - That apply best practices and design patterns,
 - Using a broad set of technologies,
 - Each of which is several lectures or a complete course unto itself.We will cover the essentials but not the depth of the individual technologies.
- One place where we will "go deep" is on the basic structure of a well-designed application/set of microservices. Some example topics are
 - Model-View-Controller.
 - Application Tier-Data Tier.
 - Encapsulation/Abstraction.
 - REST API

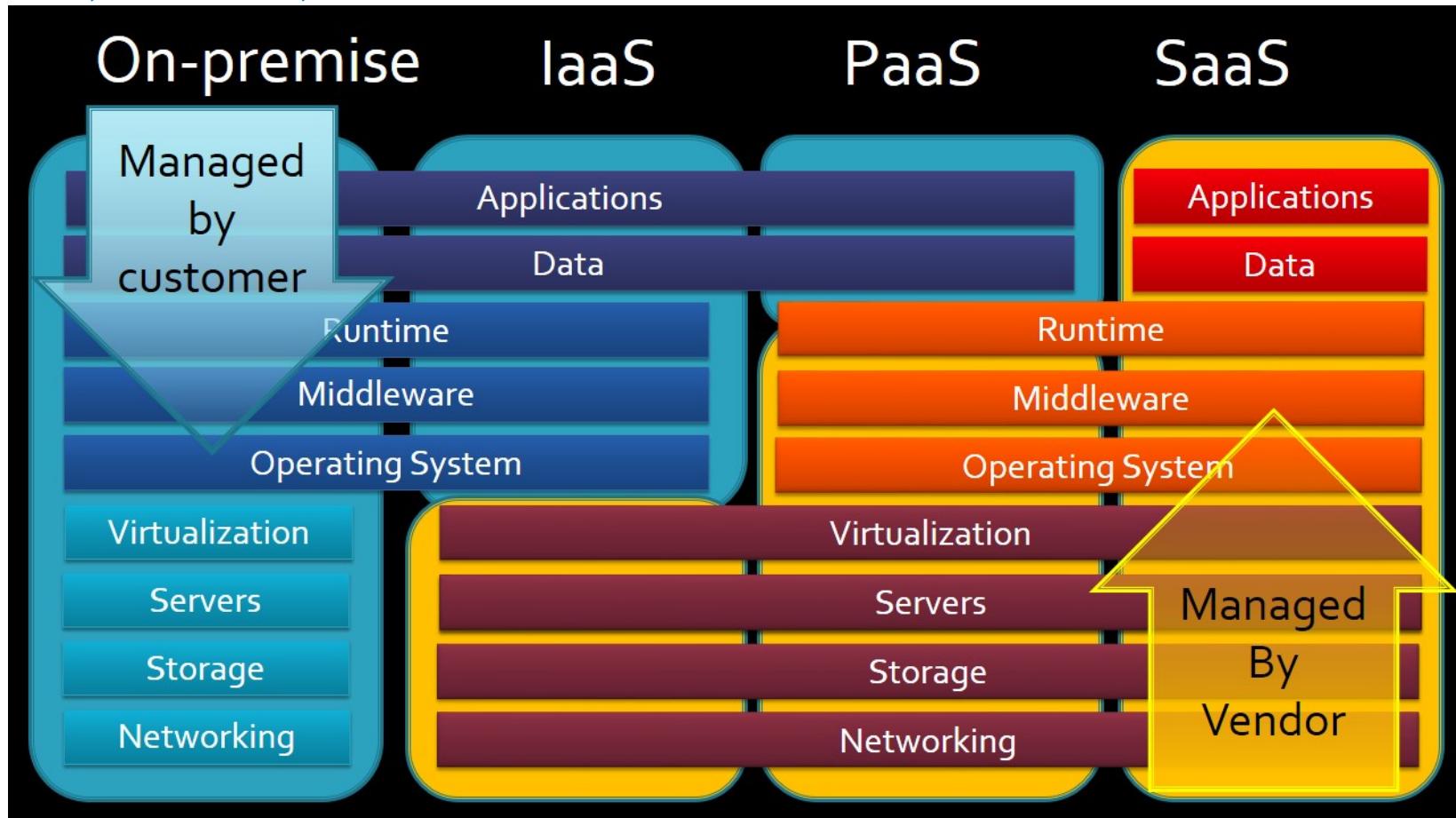
My experience from prior courses has been that I have gone through these topics too quickly, causing students to do advanced topics with a poor design.

Course Content:

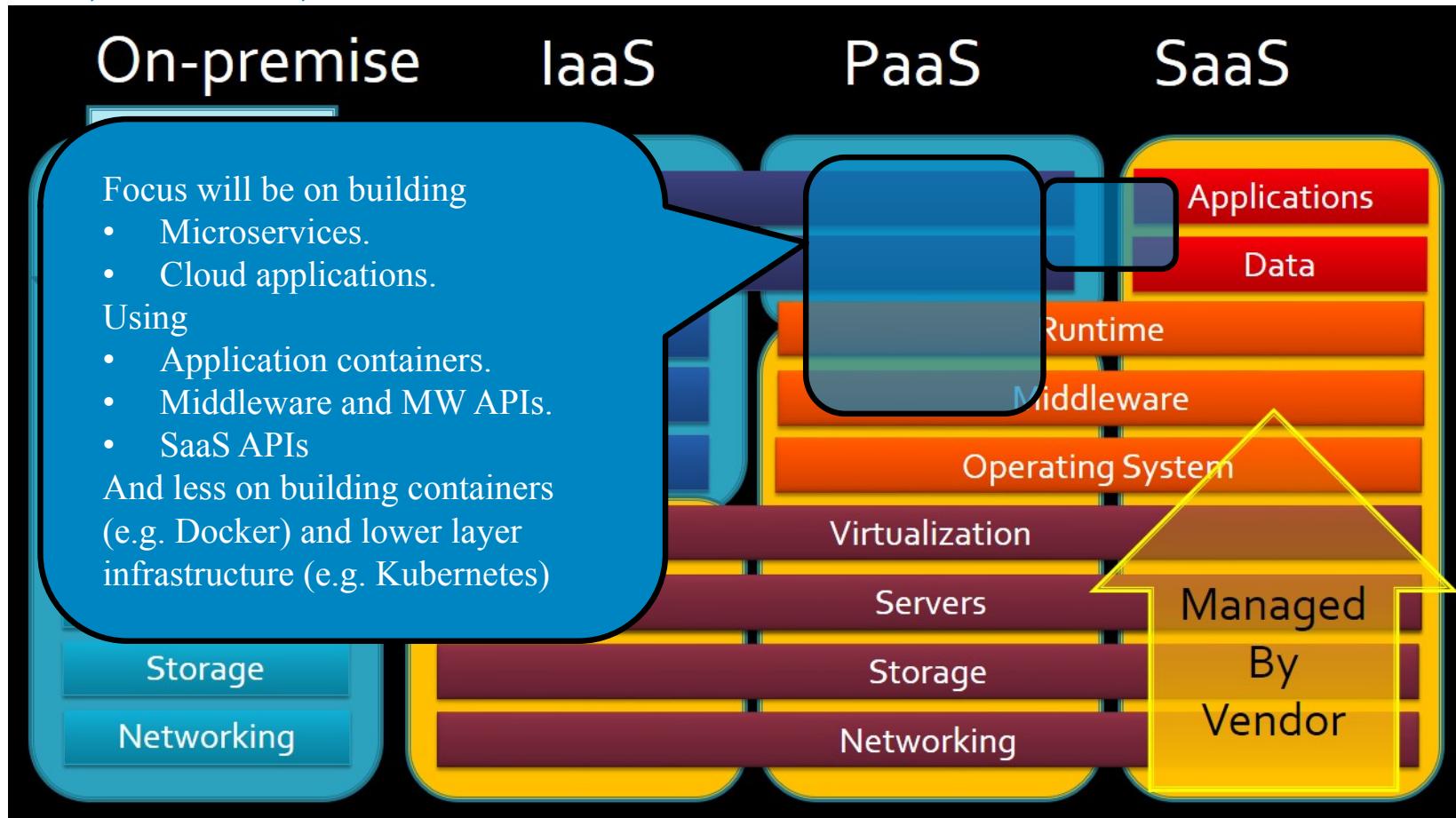
Cloud
Microservices
Applications

Cloud

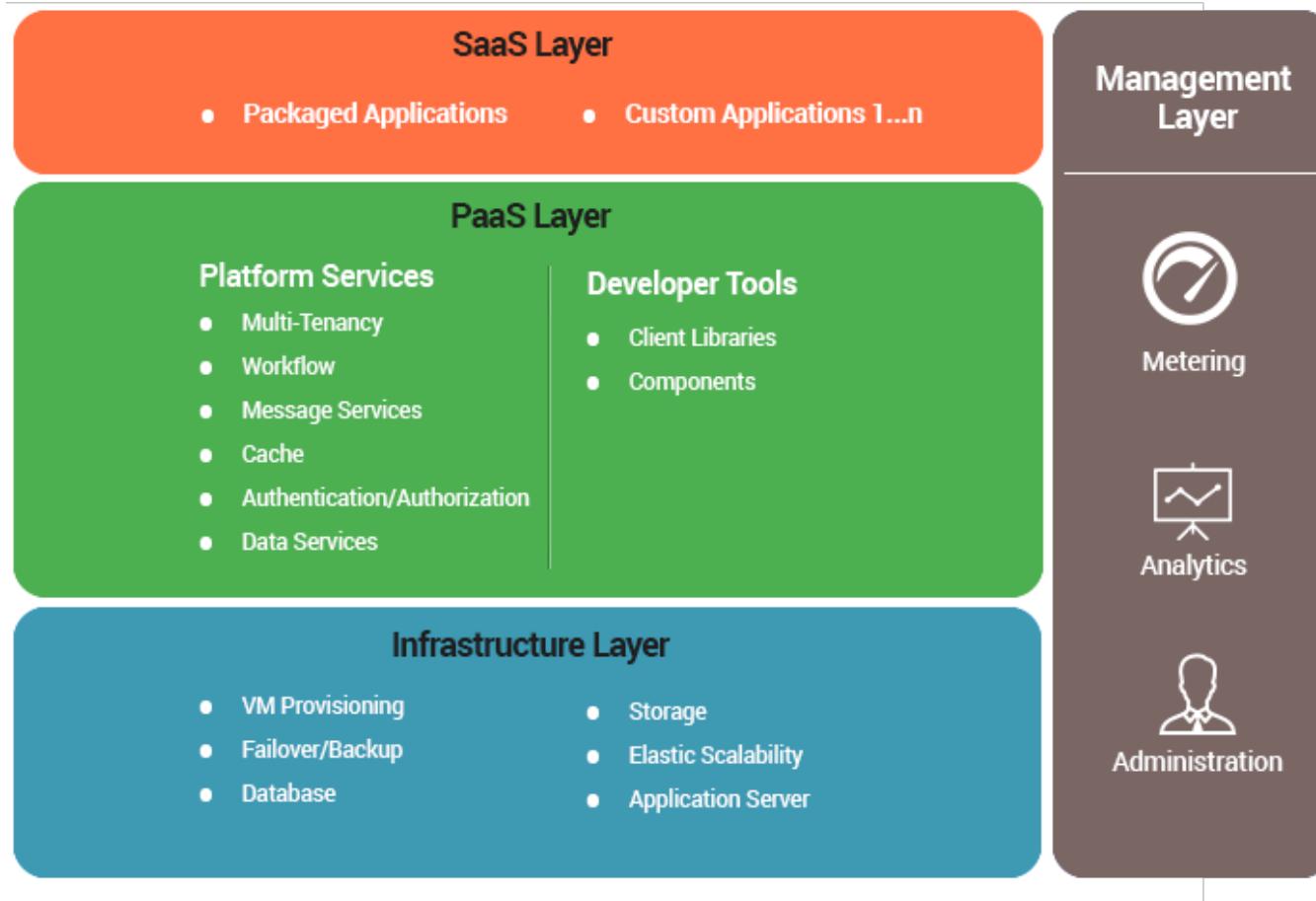
IaaS, PaaS, SaaS



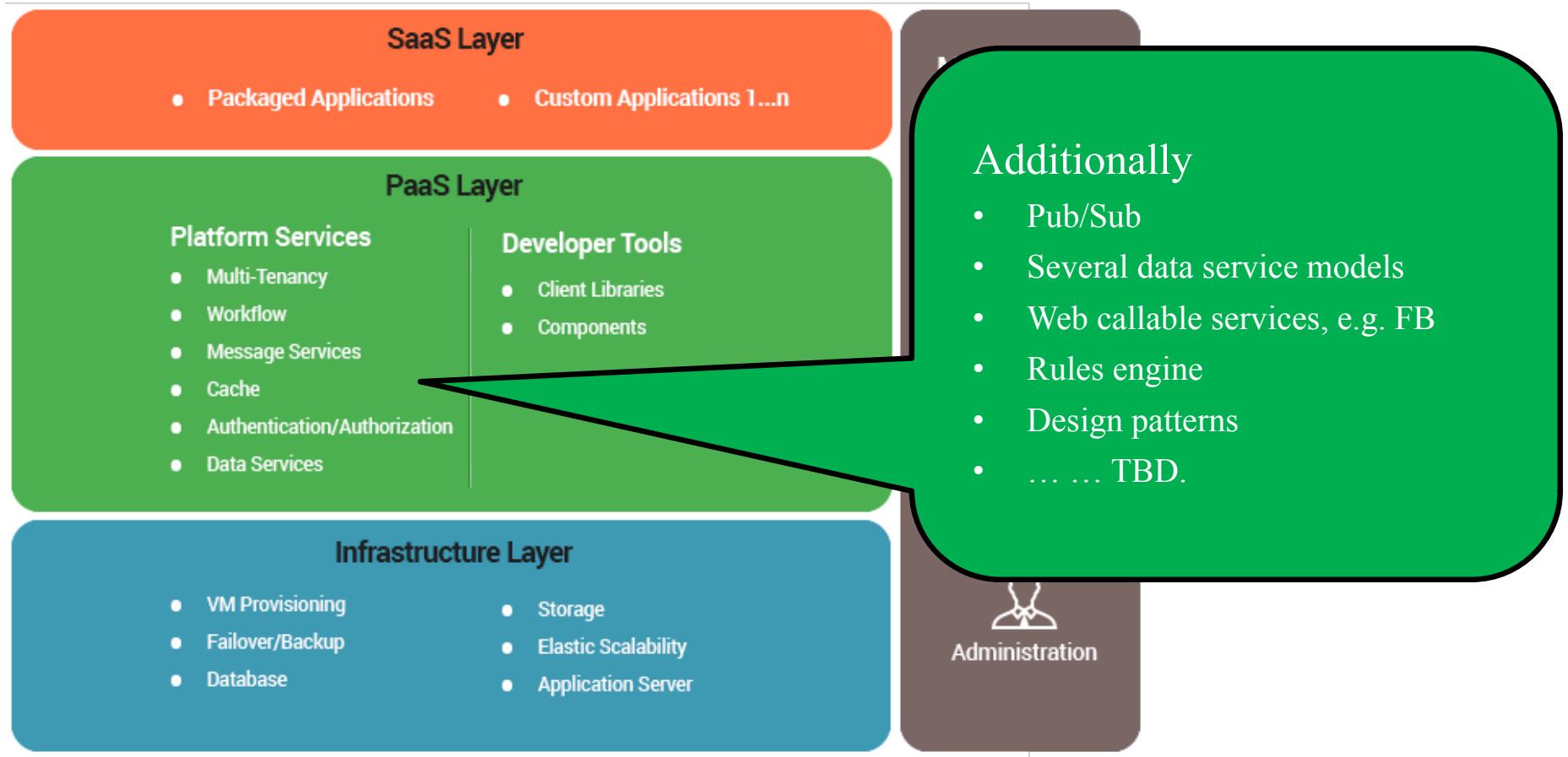
IaaS, PaaS, SaaS



IaaS, PaaS, SaaS – A Little More about PaaS



IaaS, PaaS, SaaS – A Little More about PaaS



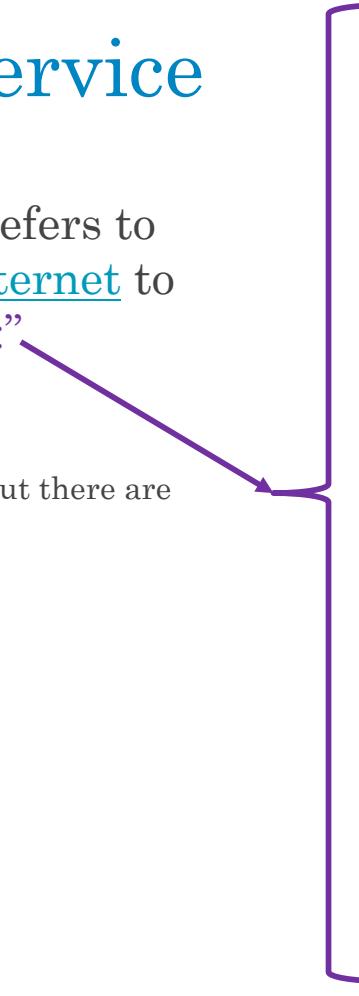
Cloud is about *- as-a-Service

(https://en.wikipedia.org/wiki/As_a_service)

“*aaS is an acronym for **as a service**, and refers to something being made available over the Internet to a customer as a service.^[1] Examples include:”

Like any list, this is an overview and

- Incomplete, e.g.
 - Business-APIs-as-a-Service: Payments is an example, but there are others, e.g.
 - Order Management, Inventory, Shipping, ...
 - Calendar and events. Contact management.
 - Human Resources.
 - Information-as-a-Service
 - Demographics.
 - Credit history and information.
 - Market data, prices, etc.
- Overly simplistic, e.g.
 - IaaS is many types of compute, storage, networking, ...
 - There are many, many types of PaaS
 - Integration-Platform-as-a-Service
 - More classical app/container, e.g. Cloud Foundry
 - Force.com and high-level component assembly.

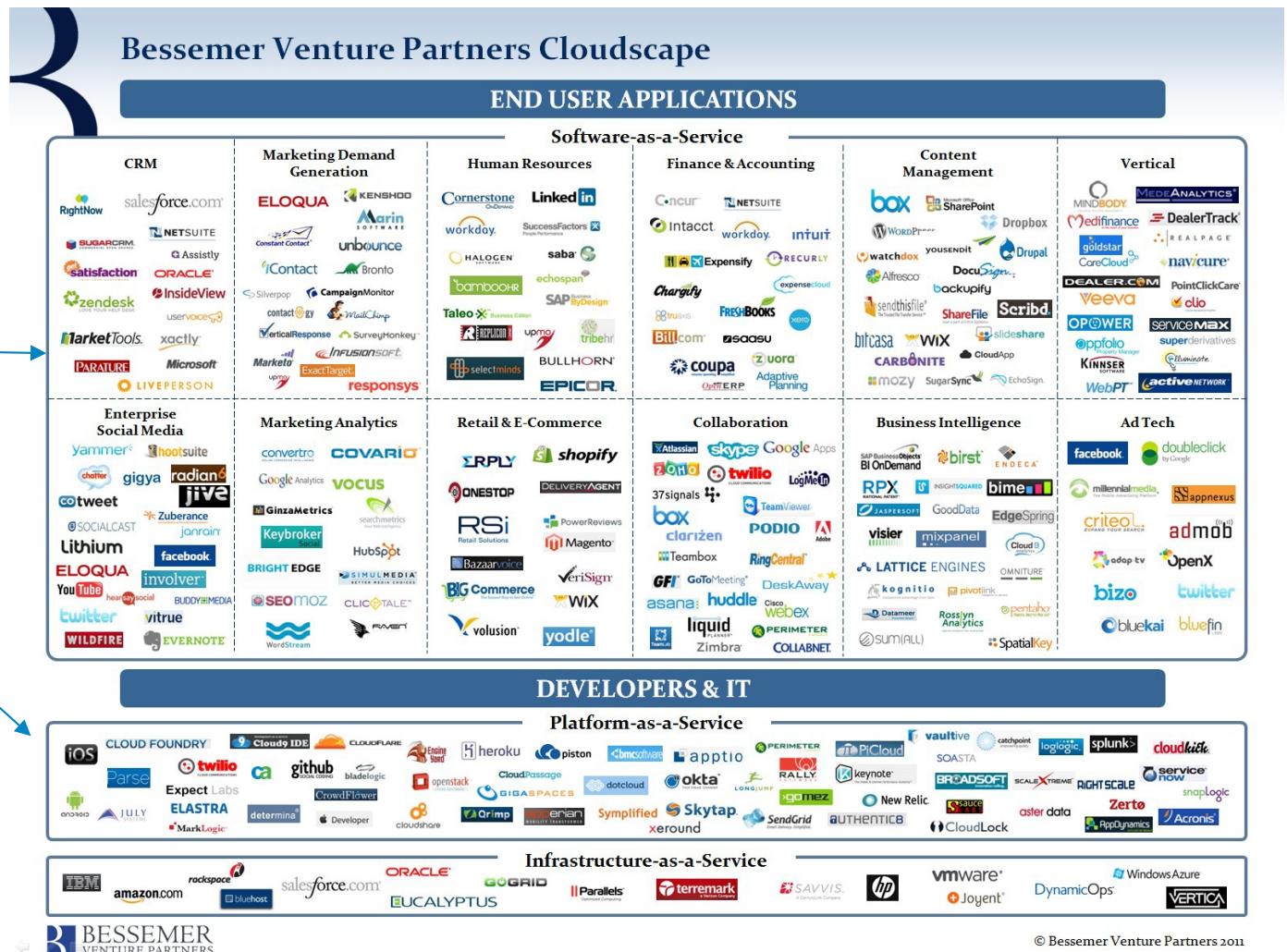


Service	Abbr.
Data as a service	DaaS
Desktop as a Service	
Energy Storage as a Service	ESaaS
Function as a Service	
Identity as a service	IDaaS
Infrastructure as a service	IaaS
IT as a service	
Logging as a service	LaaS
Mobile backend as a service	
Network as a service	
Payments as a service	
Platform as a service	PaaS
Recovery as a service	
Robot as a service	
Search as a service	
Security as a service	
Software as a service	SaaS
Storage as a service	

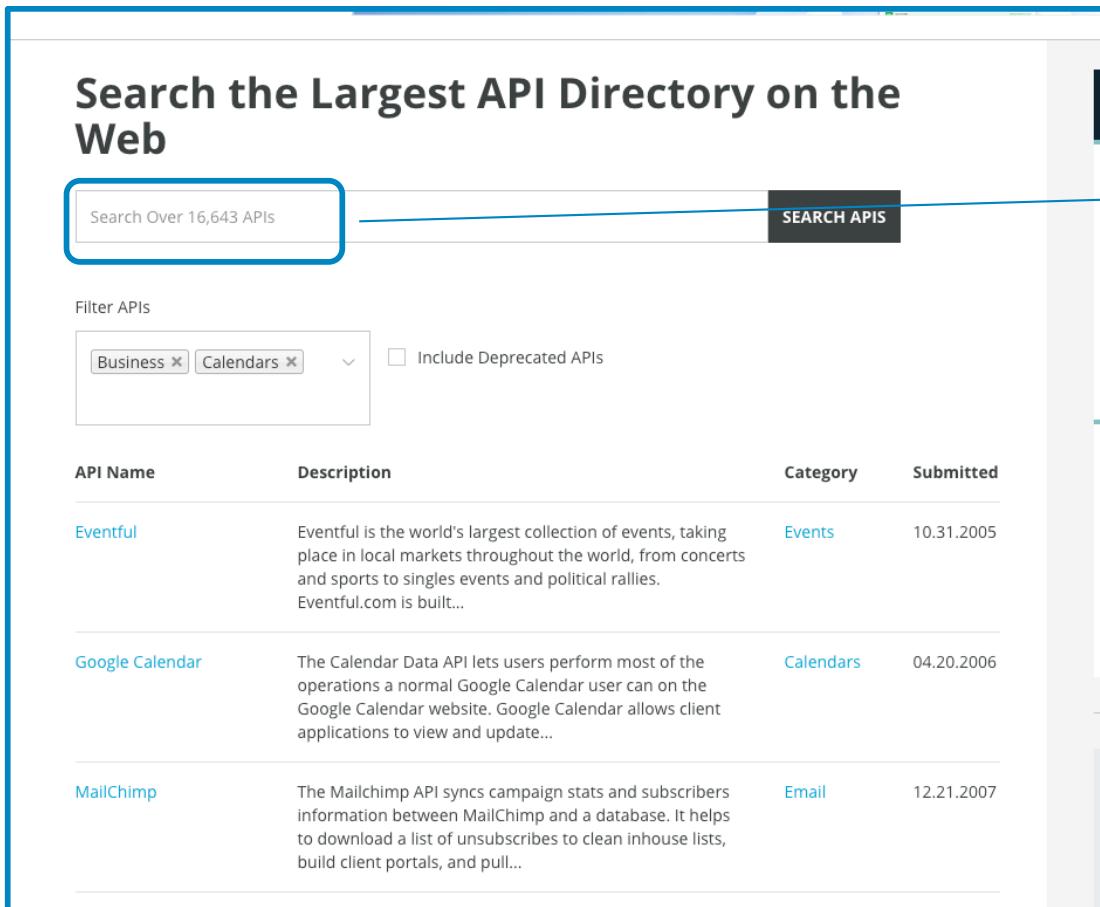
Cloudscape

Our applications will

- Call APIs
- Use and run in PaaS



www.programmableweb.com (Example)



Search the Largest API Directory on the Web

Search Over 16,643 APIs **SEARCH APIs**

Filter APIs

Business Calendars Include Deprecated APIs

API Name	Description	Category	Submitted
Eventful	Eventful is the world's largest collection of events, taking place in local markets throughout the world, from concerts and sports to singles events and political rallies. Eventful.com is built...	Events	10.31.2005
Google Calendar	The Calendar Data API lets users perform most of the operations a normal Google Calendar user can on the Google Calendar website. Google Calendar allows client applications to view and update...	Calendars	04.20.2006
MailChimp	The Mailchimp API syncs campaign stats and subscribers information between MailChimp and a database. It helps to download a list of unsubscribes to clean inhouse lists, build client portals, and pull...	Email	12.21.2007

Over 16,000 APIs

Microservices

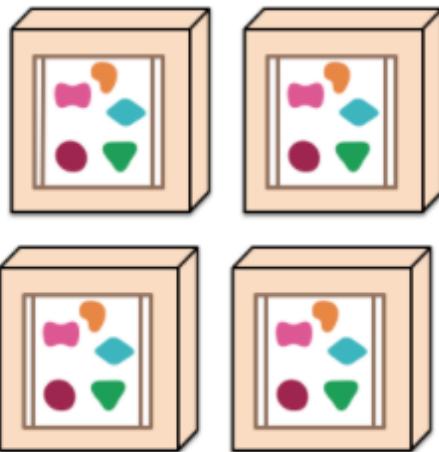
Microservices

(<https://martinfowler.com/articles/microservices.html>)

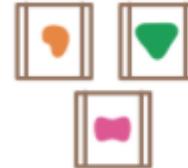
A monolithic application puts all its functionality into a single process...



... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...



... and scales by distributing these services across servers, replicating as needed.

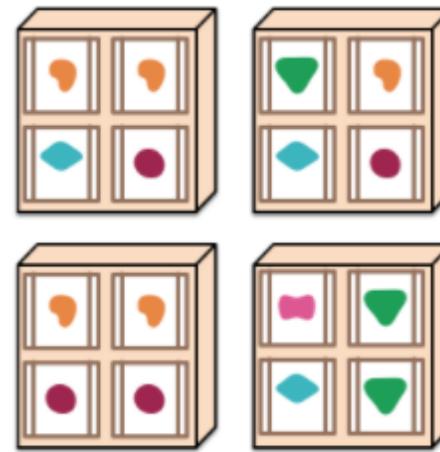


Figure 1: Monoliths and Microservices

Web Application Basic Concepts

1. User performs an action that requires data from a database to be displayed.



2. A request is formed and sent from the client to the web server.



3. The request is processed and the database is queried.



6. Information is displayed to the user.

5. An appropriate response is generated and sent back.

4. Data is retrieved.

Application User

Web Client
(Presentation Tier)

Web Server
(Application Tier)

Database
(Data Tier)

“Traditional” Web Application Architecture

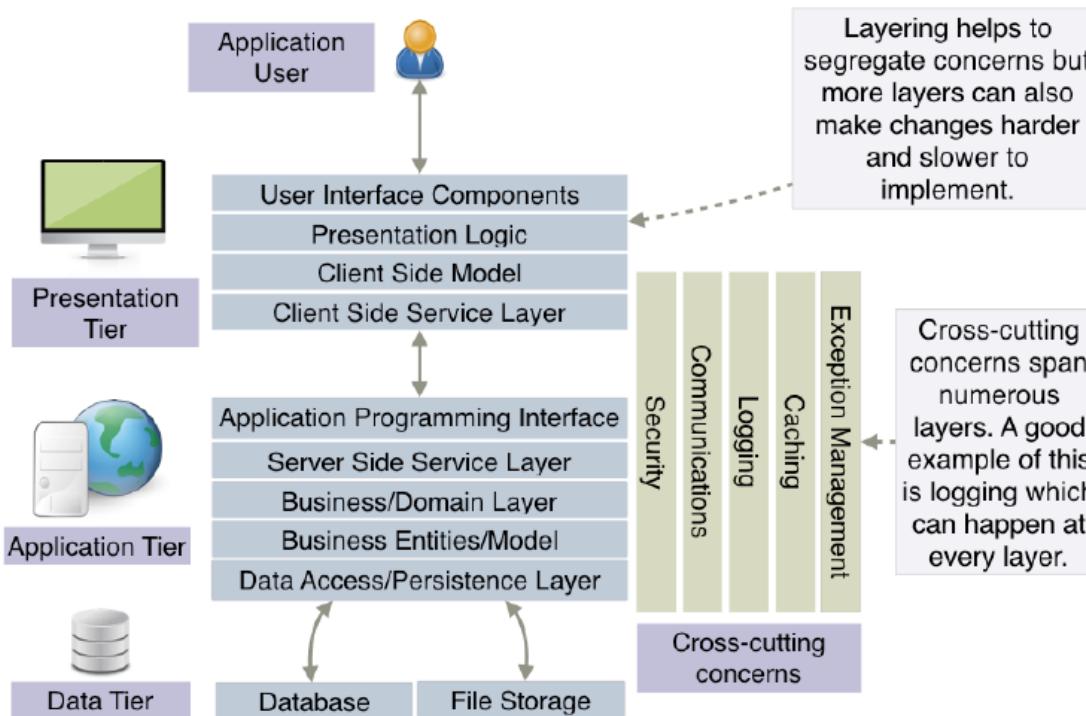
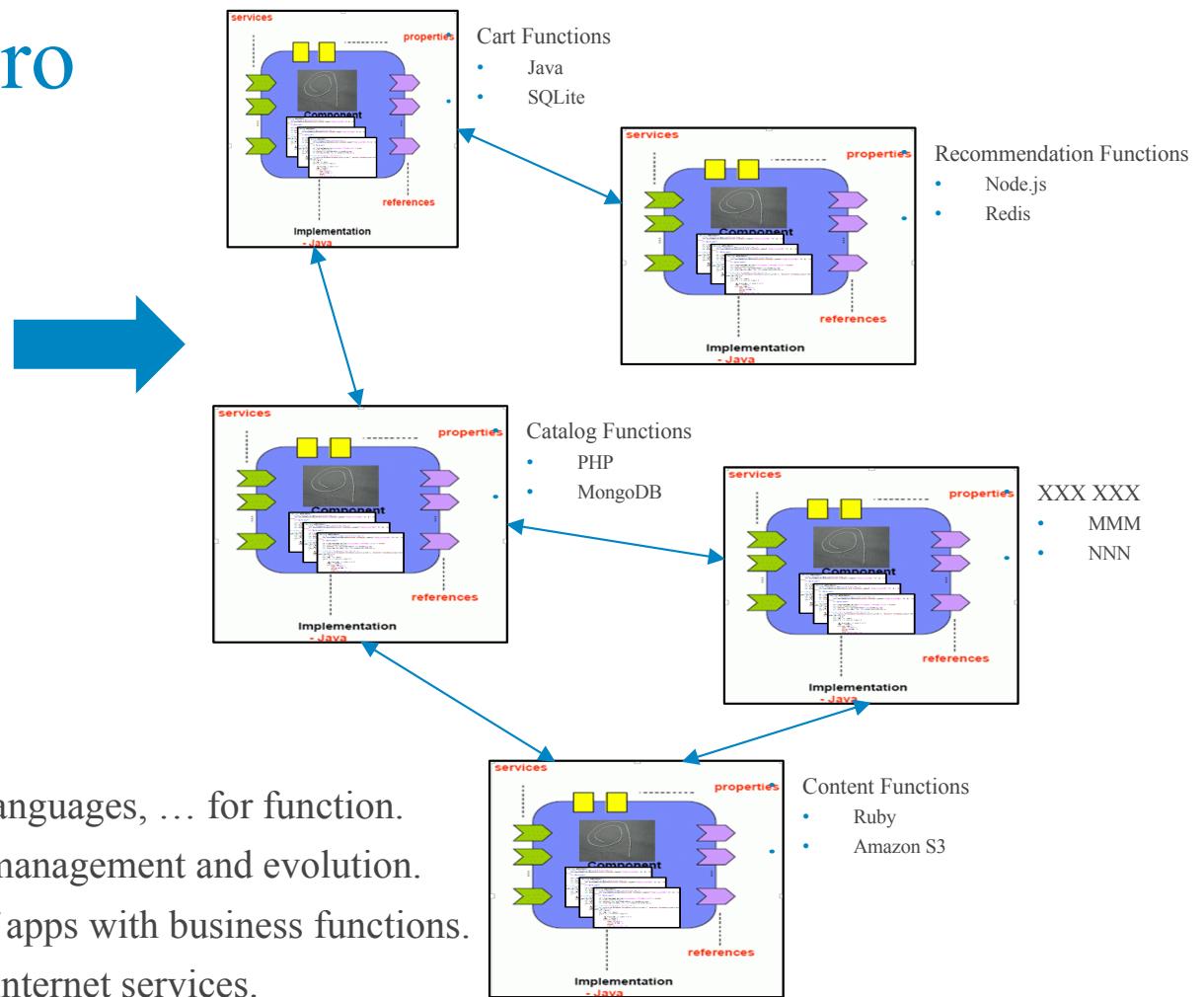
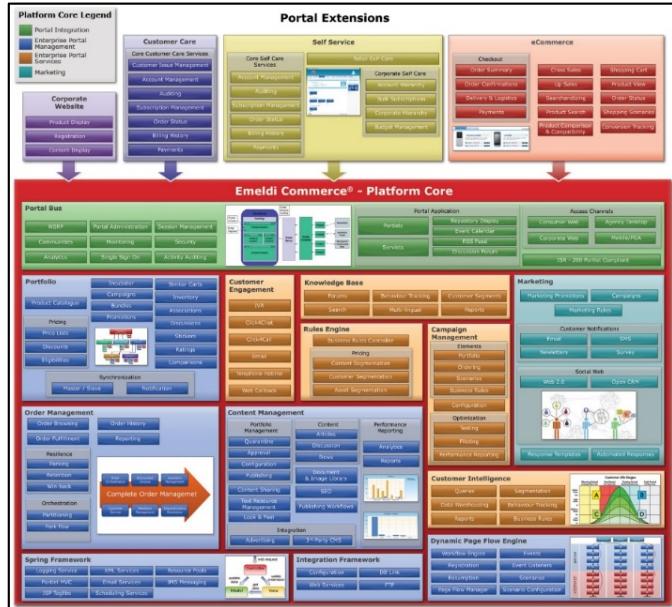


Figure 1.2. A typical 3-tier application is usually made up of a presentation, application, and data tiers. In a tier there may be multiple layers that have specific responsibilities. A developer can choose how layers will interact with one another. This can be strictly top-down or in a loose way where layers can bypass their immediate neighbors to talk to other layers.

Monolithic to Micro



Motivations

- Enable best tools, languages, ... for function.
 - Simplifies change management and evolution.
 - Better alignment of apps with business functions.
 - Reuse of code and internet services.

Micro-services Characteristics

- Componentization via Services
- Organized around Business Capabilities
- Products not Projects
- Smart endpoints and dumb pipes
- Decentralized Governance
- Decentralized Data Management
- Infrastructure Automation
- Design for failure
- Evolutionary Design

SOA VS Microservices

	Traditional SOA	Microservices
Messaging type	Smart, but dependency-laden ESB	Dumb, fast messaging (as with Apache Kafka)
Programming style	Imperative model	Reactive actor programming model that echoes agent-based systems
Lines of code per service	Hundreds or thousands of lines of code	100 or fewer lines of code
State	Stateful	Stateless
Messaging type	Synchronous: wait to connect	Asynchronous: publish and subscribe
Databases	Large relational databases	NoSQL or micro-SQL databases blended with conventional databases
Code type	Procedural	Functional
Means of evolution	Each big service evolves	Each small service is immutable and can be abandoned or ignored
Means of systemic change	Modify the monolith	Create a new service
Means of scaling	Optimize the monolith	Add more powerful services and cluster by activity
System-level awareness	Less aware and event driven	More aware and event driven

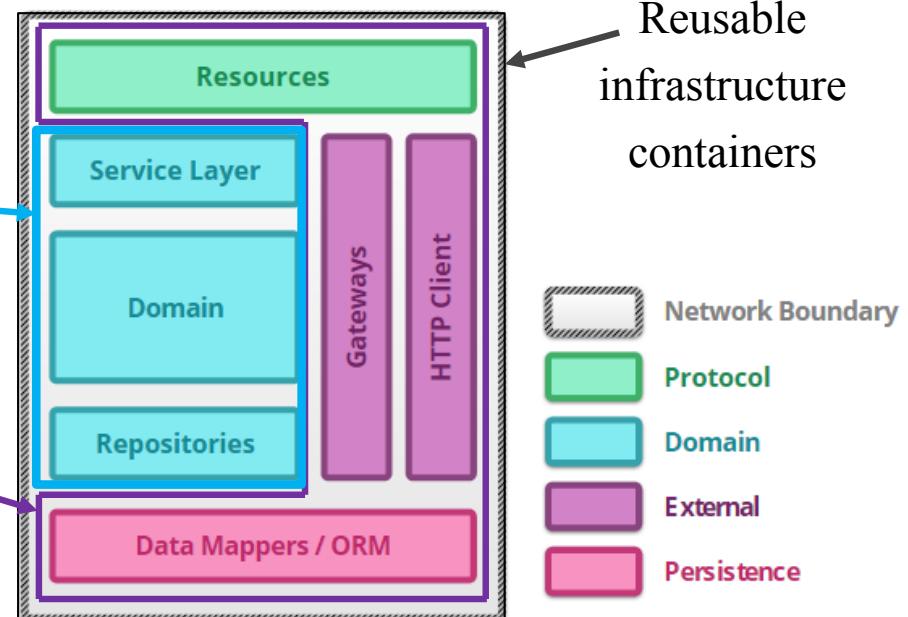
<http://usblogs.pwc.com/emerging-technology/agile-coding-in-enterprise-it-code-small-and-local/>

Microservices can usually be split into similar kinds of modules

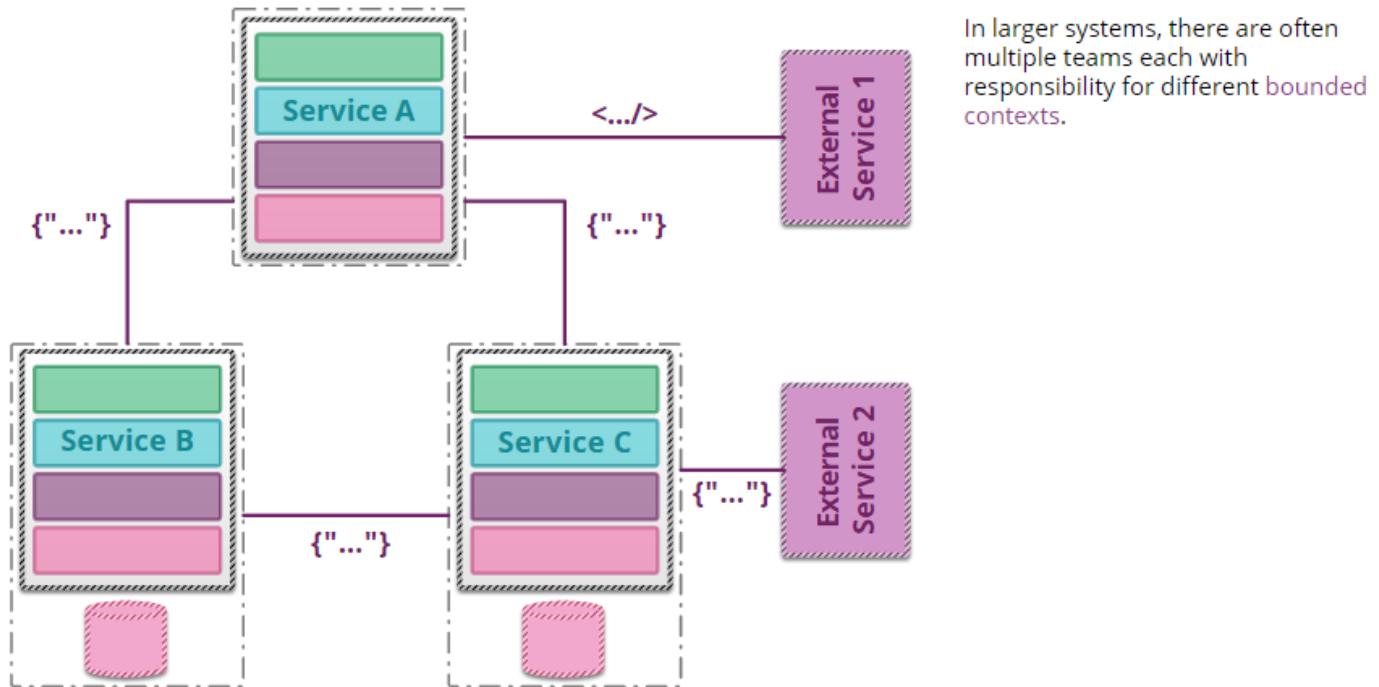
Often, microservices display similar internal structure consisting of some or all of the displayed layers.

Inject application implementation into reusable SW containers

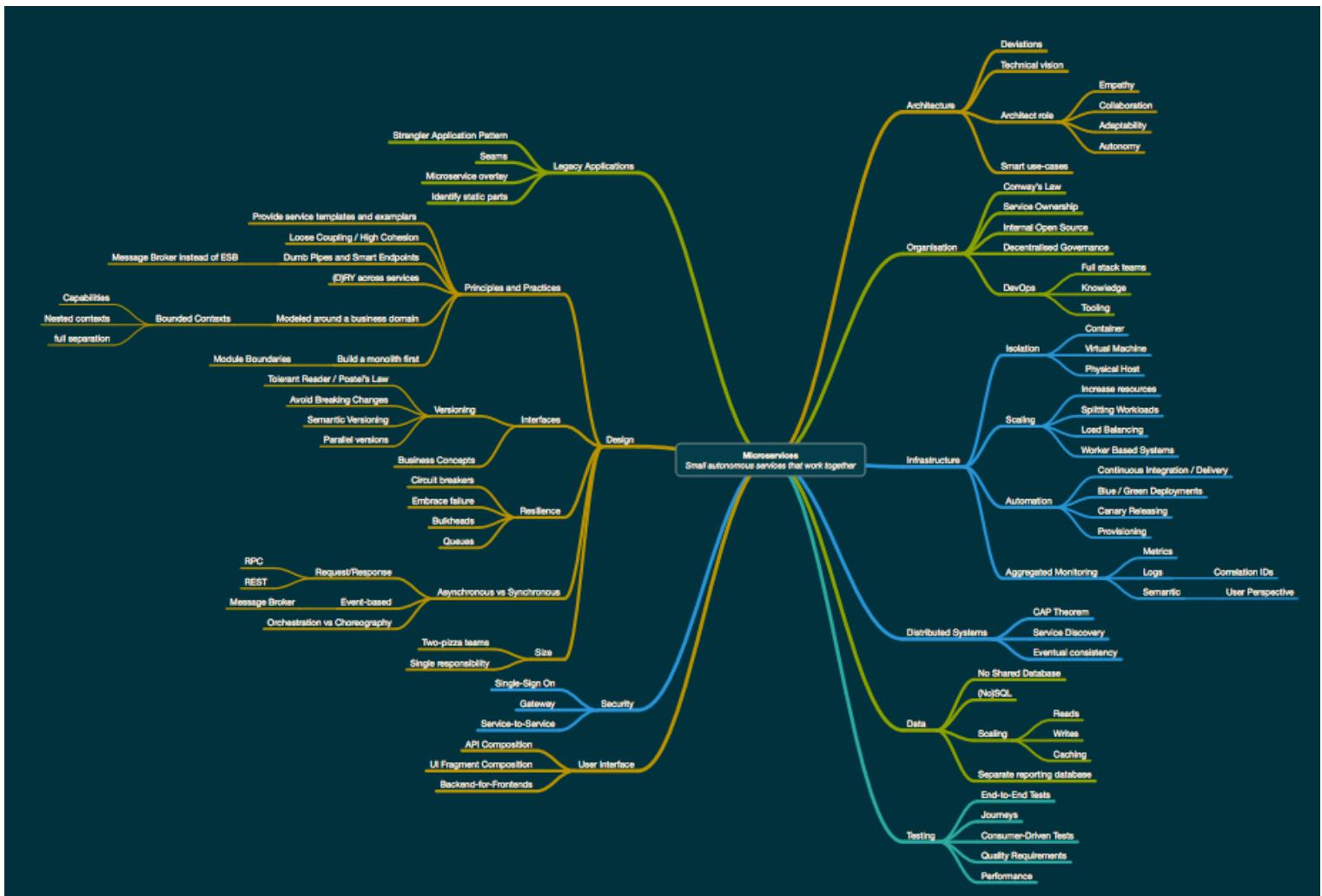
Reusable SW containers but with core technology and frameworks



*Multiple services work together as a system...
...to provide business valuable features*



Patterns



Patterns

Patterns and practices are essential

- PaaS and advanced infrastructure eliminate the need to worry about a lot of things, but ...
- No infrastructure is smart enough to make a poorly designed application flexible, scalable, available, ...



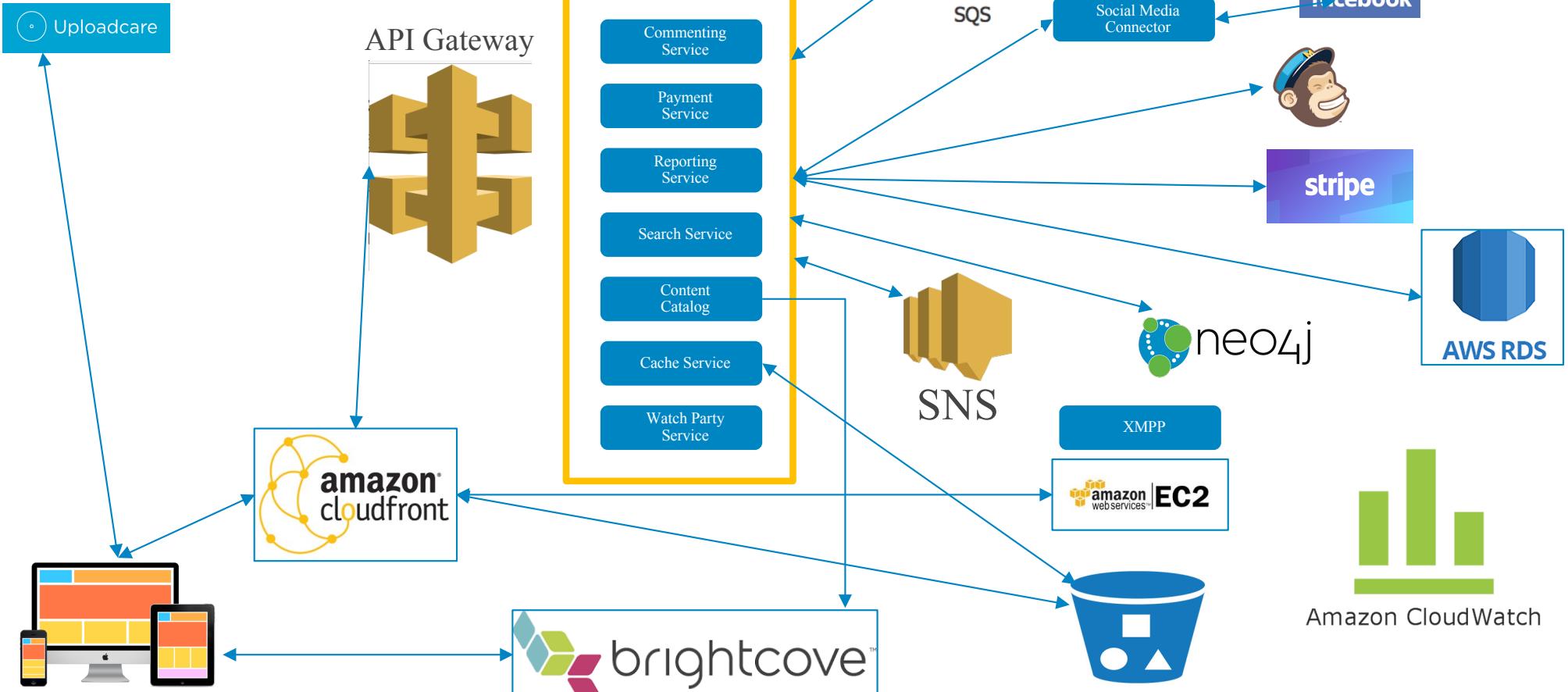
Weinberg's Second Law

“If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization.”

An Example

Seeka Demo

(Some) Elements



0th Project

0th Project

- Teams
 - Form your teams (approx. 5 people)
 - Identify contact focal point.
 - Give your team a “cool” name.
- Signup (reuse) and Amazon Web Service Account
 - Free Tier should be fine.
 - Provide access to team members.
- Create an Elastic Beanstalk instance/application.
 - Use one of the sample application.
 - Will have to make more sophisticated starting next week;
I will use Node JS with Express for Elastic Beanstalk examples.