COMS E6998: **Microservices and Cloud Applications**

Lecture 1: Introduction

Dr. Donald F. Ferguson
dff9@columbia.edu

A Little About Me

The image part with relationship ID rId2 was not found in the file.

The image part with relationship ID rId2 was not found in the file.

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

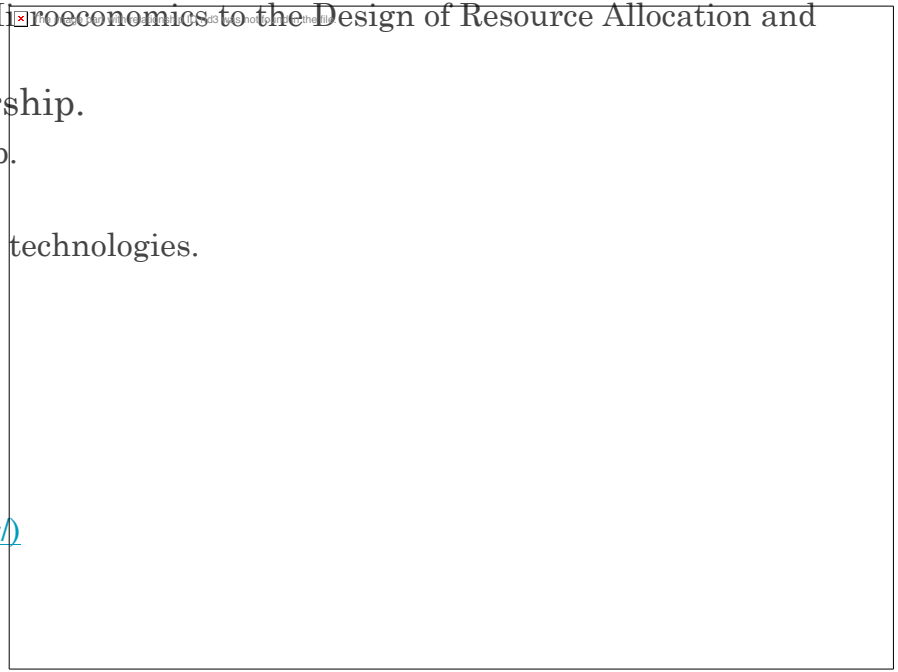
A Little About Me

- 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

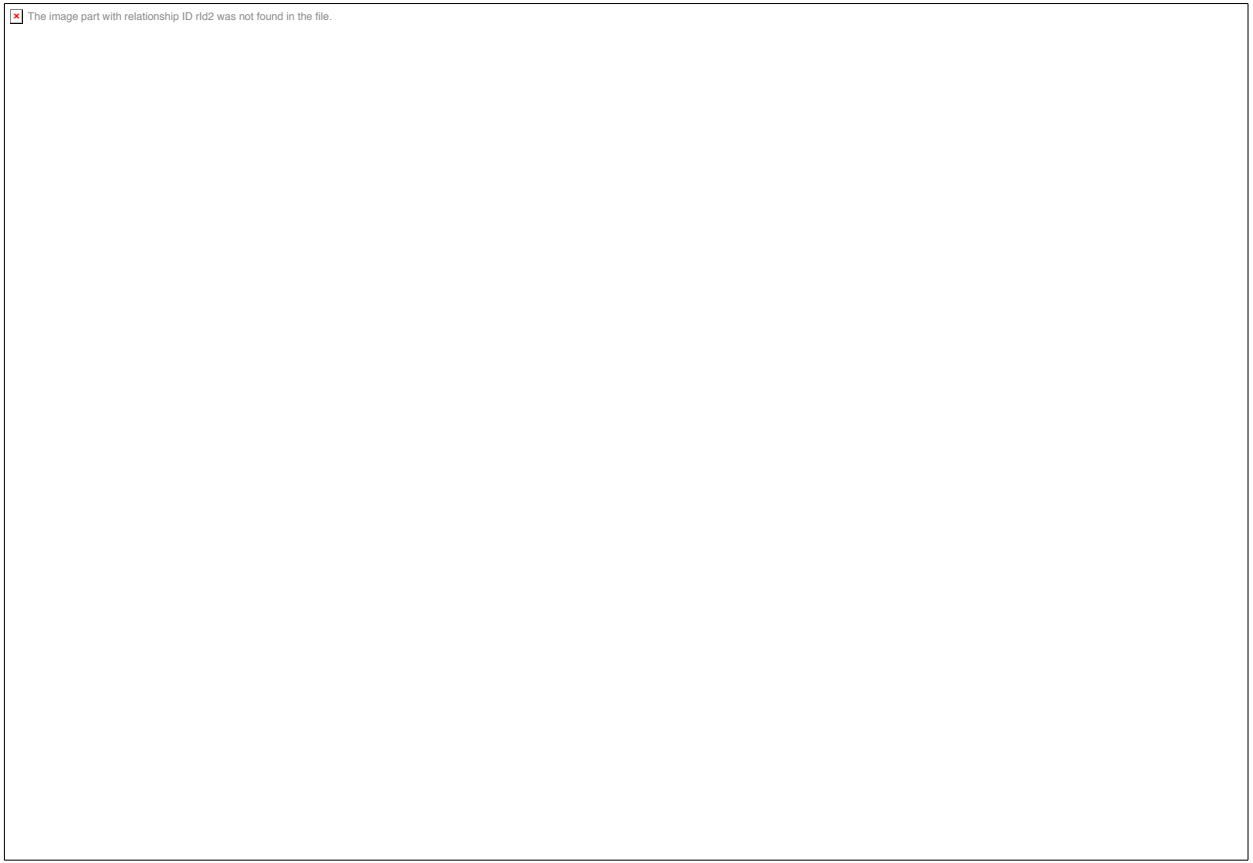
The image part with relationship ID rId3 was not found in the file.

A Little About Me

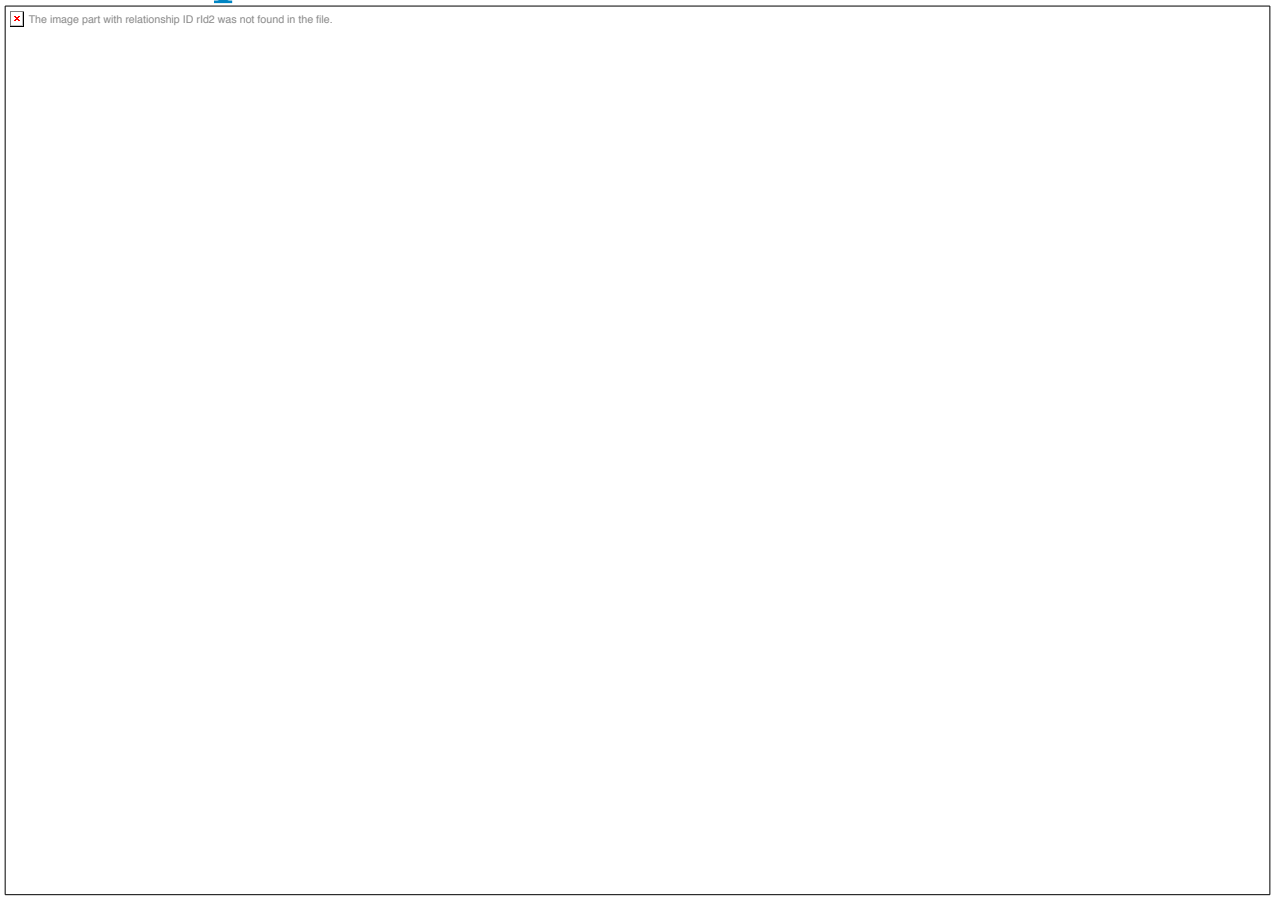
- Columbia through and through
  - BA (Columbia College) '82, M.S. '84 (Engineering), M.Phil. '84 (GSAS).
  - Ph.D. '87 (GSAS) – Thesis: "The Application of Microeconomics to the Design of Resource Allocation and Control Algorithms."
- 32 years of technology contribution and leadership.
  - IBM Fellow: IBM Research, IBM Software Group.
  - Microsoft Technical Fellow.
  - Distinguished Engineer, CTO, Executive VP, CA technologies.
  - Sr. Fellow, CTO, Dell Software Group.
  - ~~Co-Found, CTO Seeka TV.~~
- 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



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.

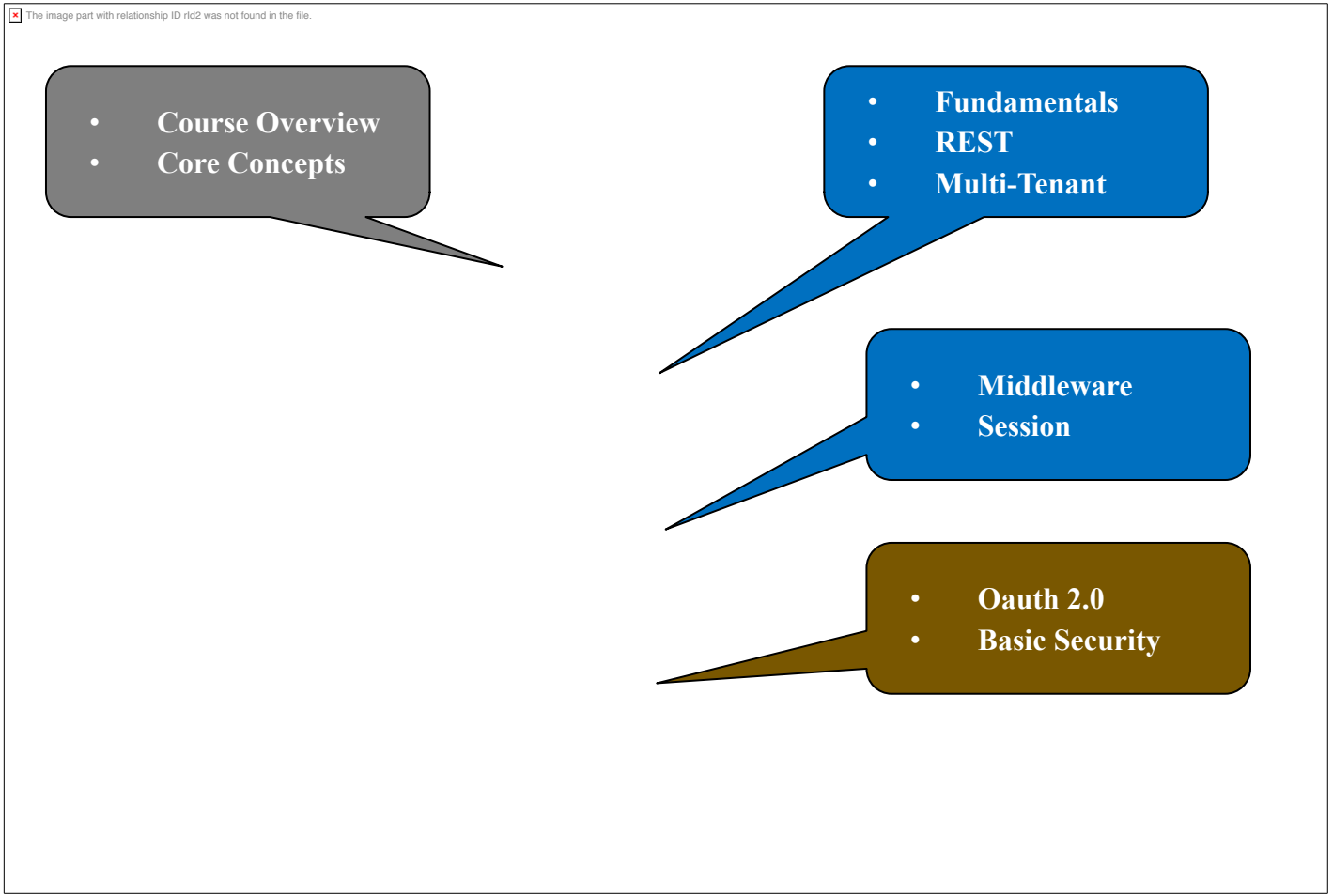
Logistics

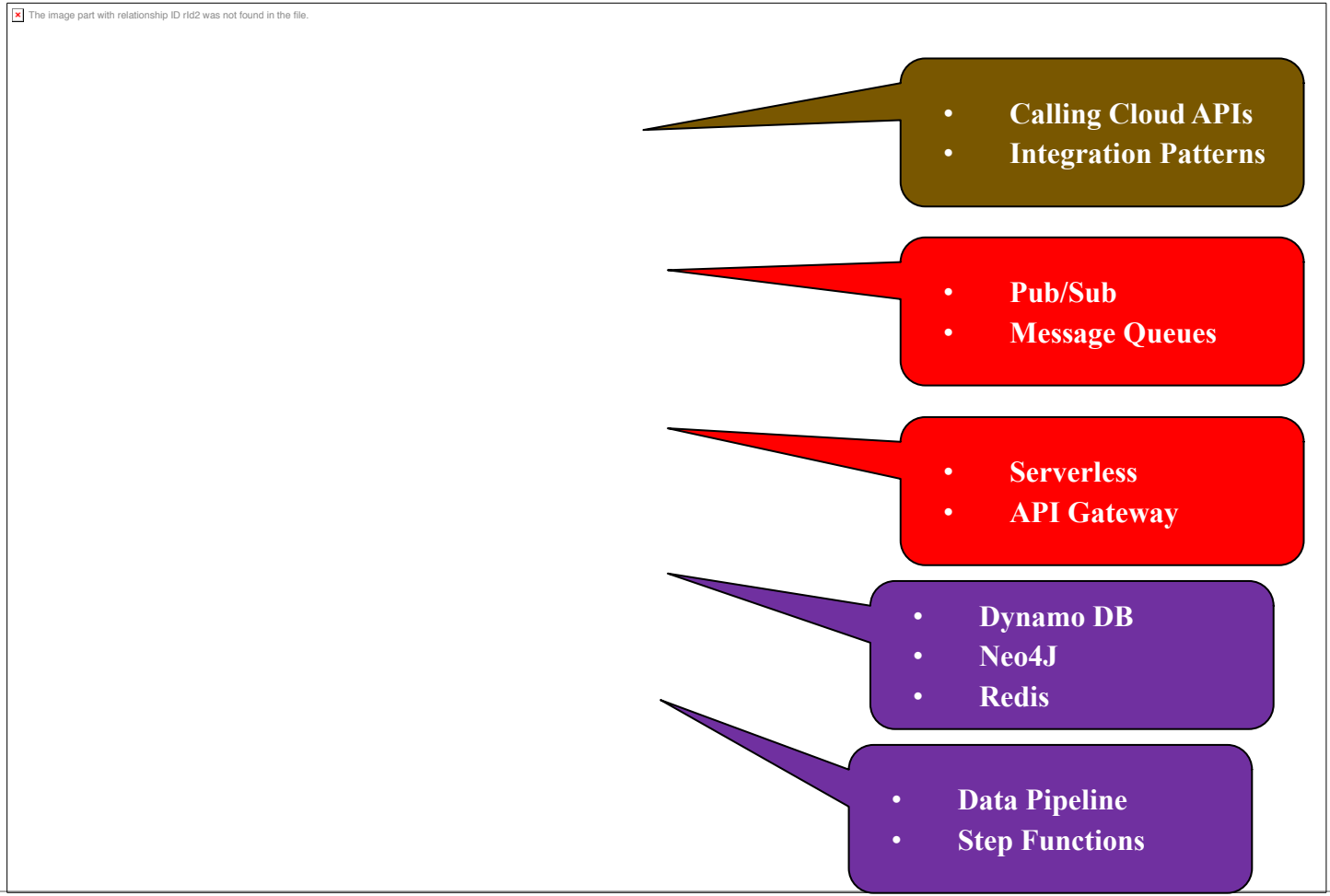
Logistics

- 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.**

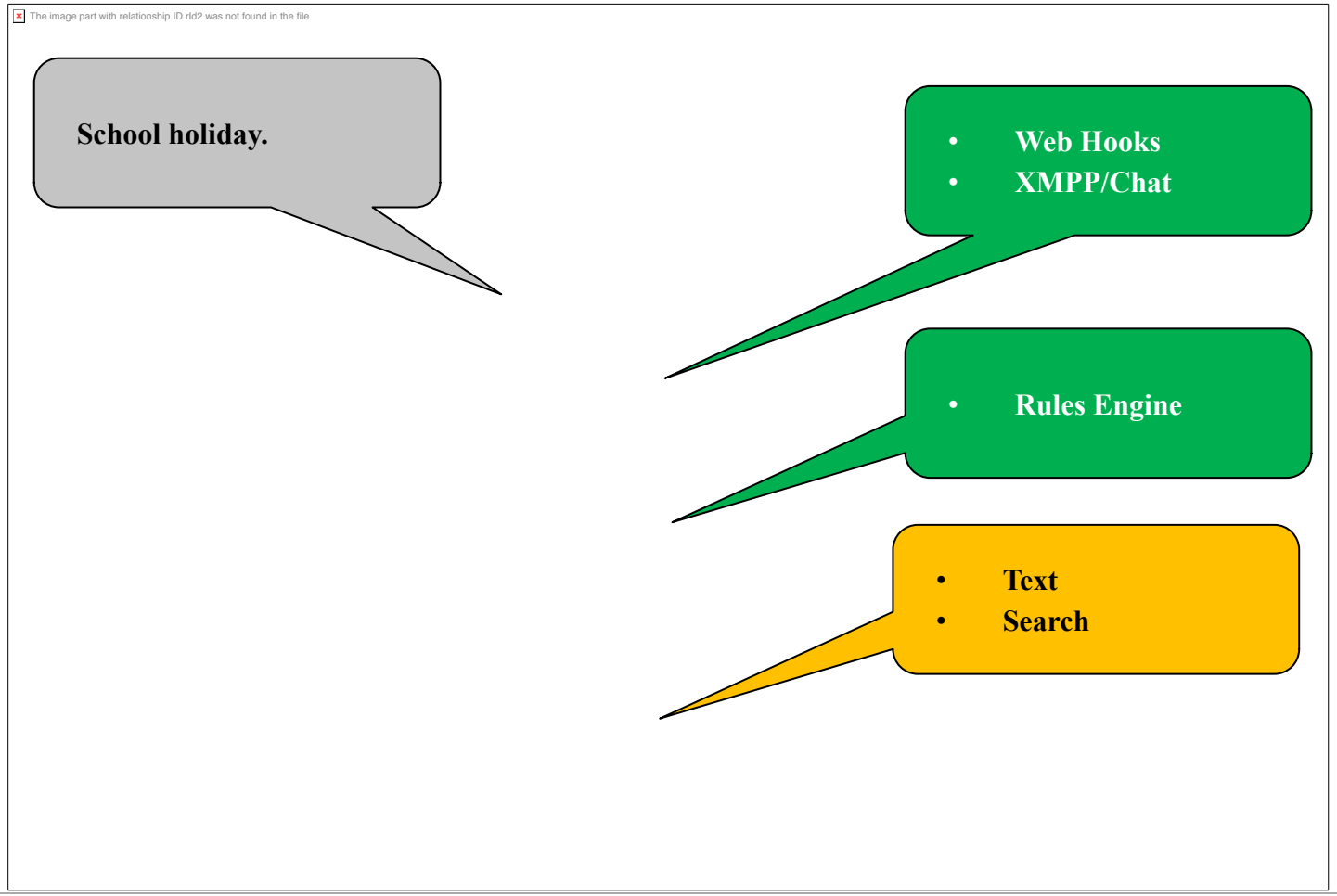
Course Overview and Content

Course Overview

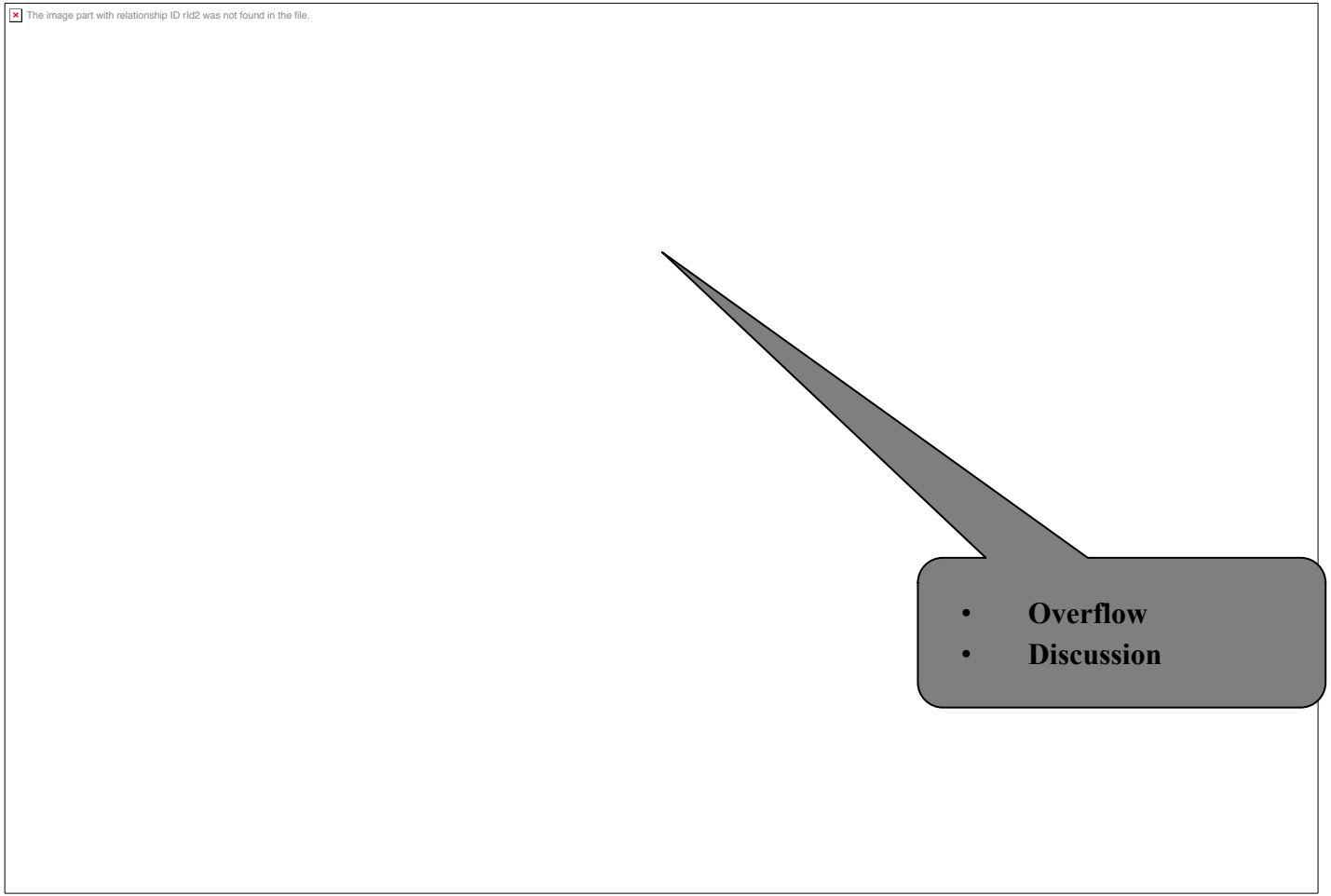




The image part with relationship ID rId2 was not found in the file.



The image part with relationship ID rId2 was not found in the file.



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

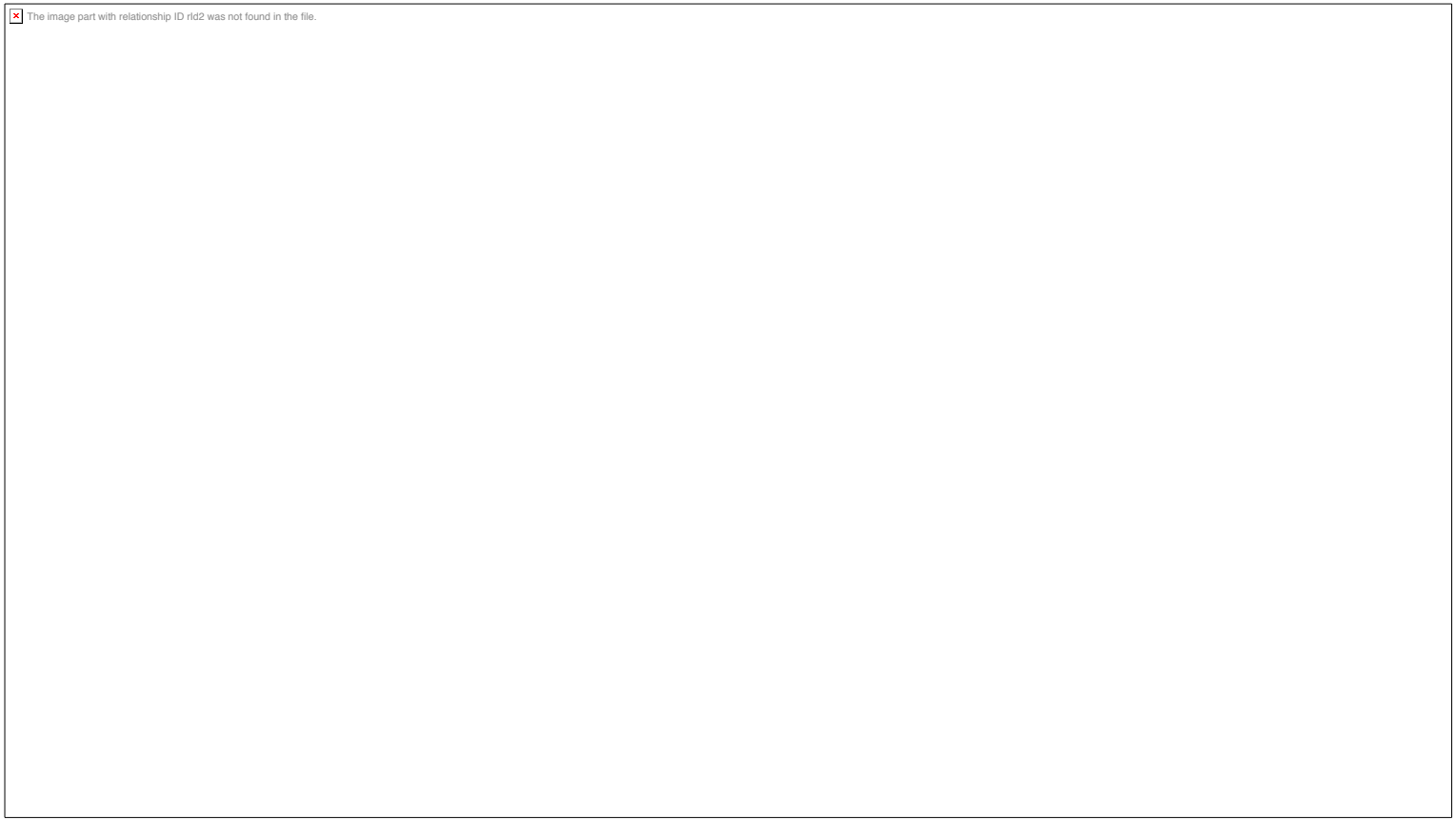
  My experience from prior courses has been that I have gone through these topics too quickly, causing students to do advanced topics with a poor design.

Course Content:

Cloud
Microservices
Applications

Cloud

IaaS, PaaS, SaaS



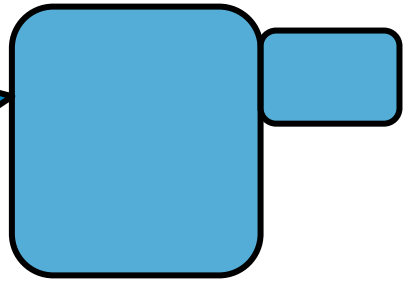
IaaS, PaaS, SaaS

The image part with relationship ID rId2 was not found in the file.

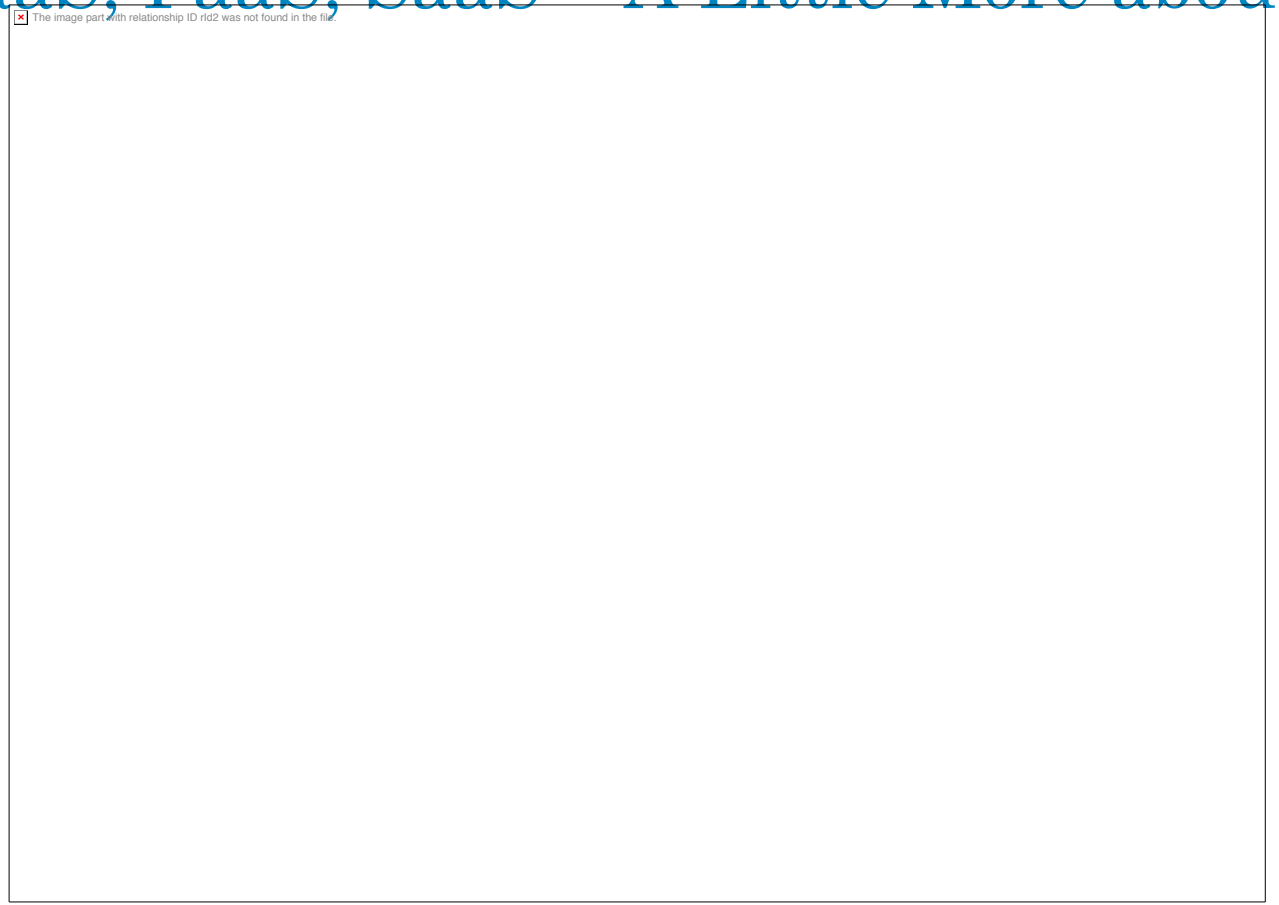
Focus will be on building
- Microservices.
- Cloud applications.

Using
- Application containers.
- Middleware and MW APIs.
- SaaS APIs

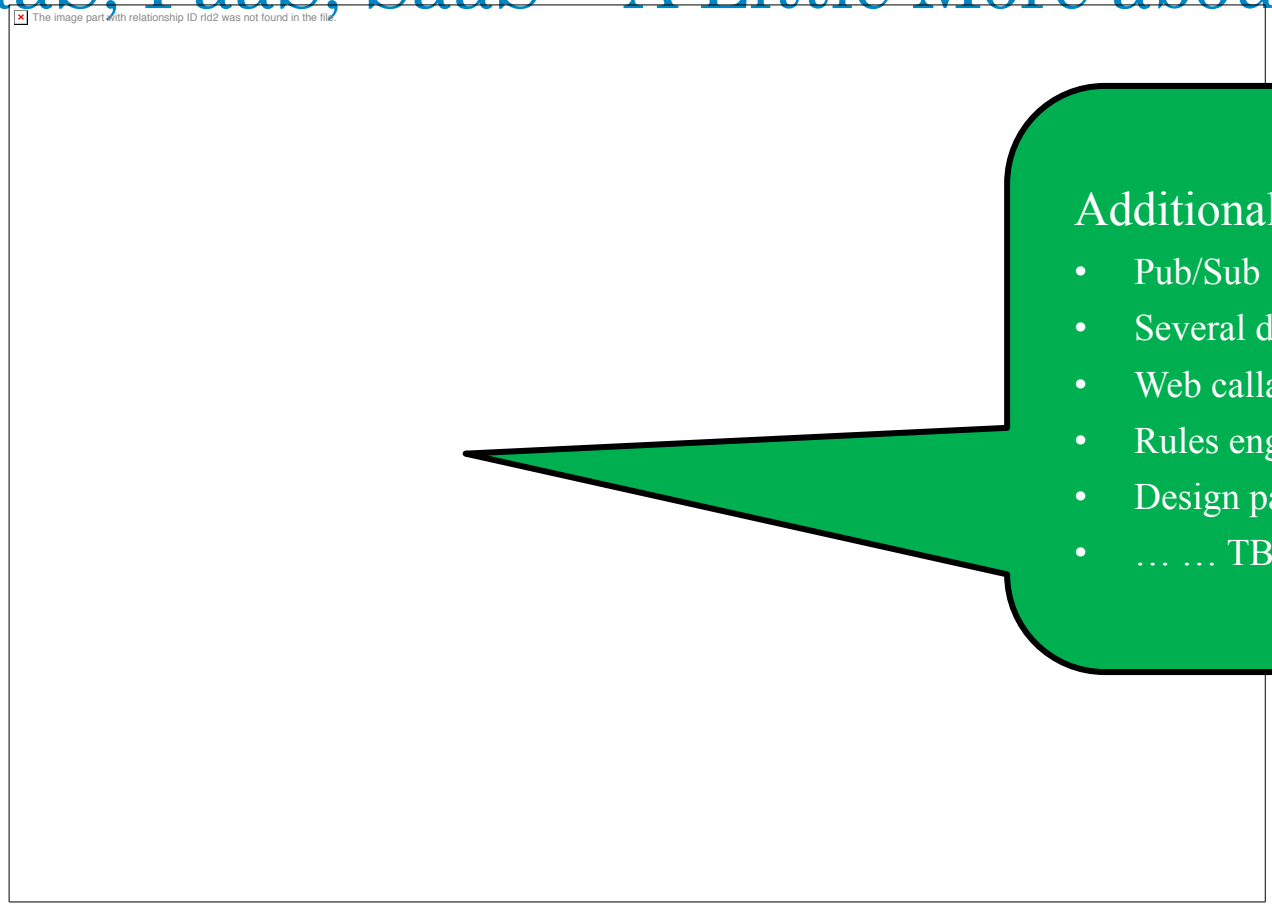
And less on building containers (e.g. Docker) and lower layer infrastructure (e.g. Kubernetes)



IaaS, PaaS, SaaS – A Little More about PaaS



IaaS, PaaS, SaaS – A Little More about PaaS



Additionally

- Pub/Sub
- Several data service models
- Web callable services, e.g. FB
- Rules engine
- Design patterns
- … … TBD.

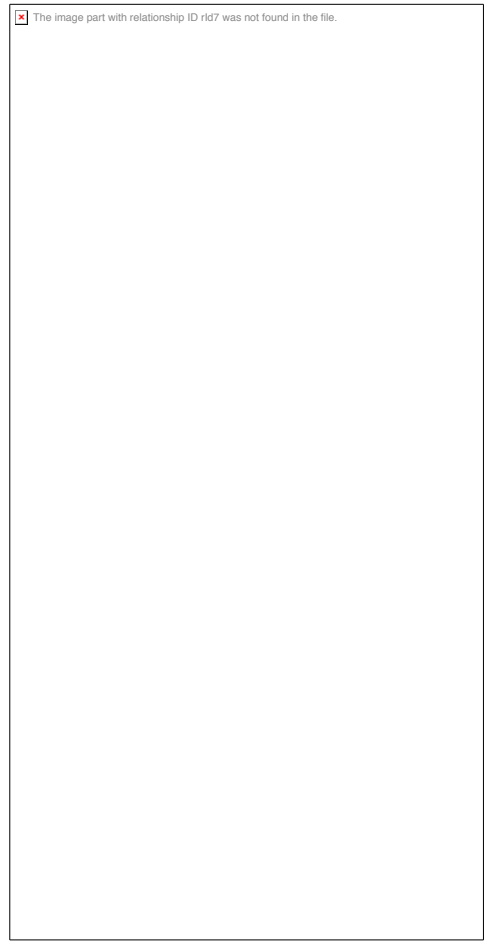
Cloud is about *- as-a-Service

(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

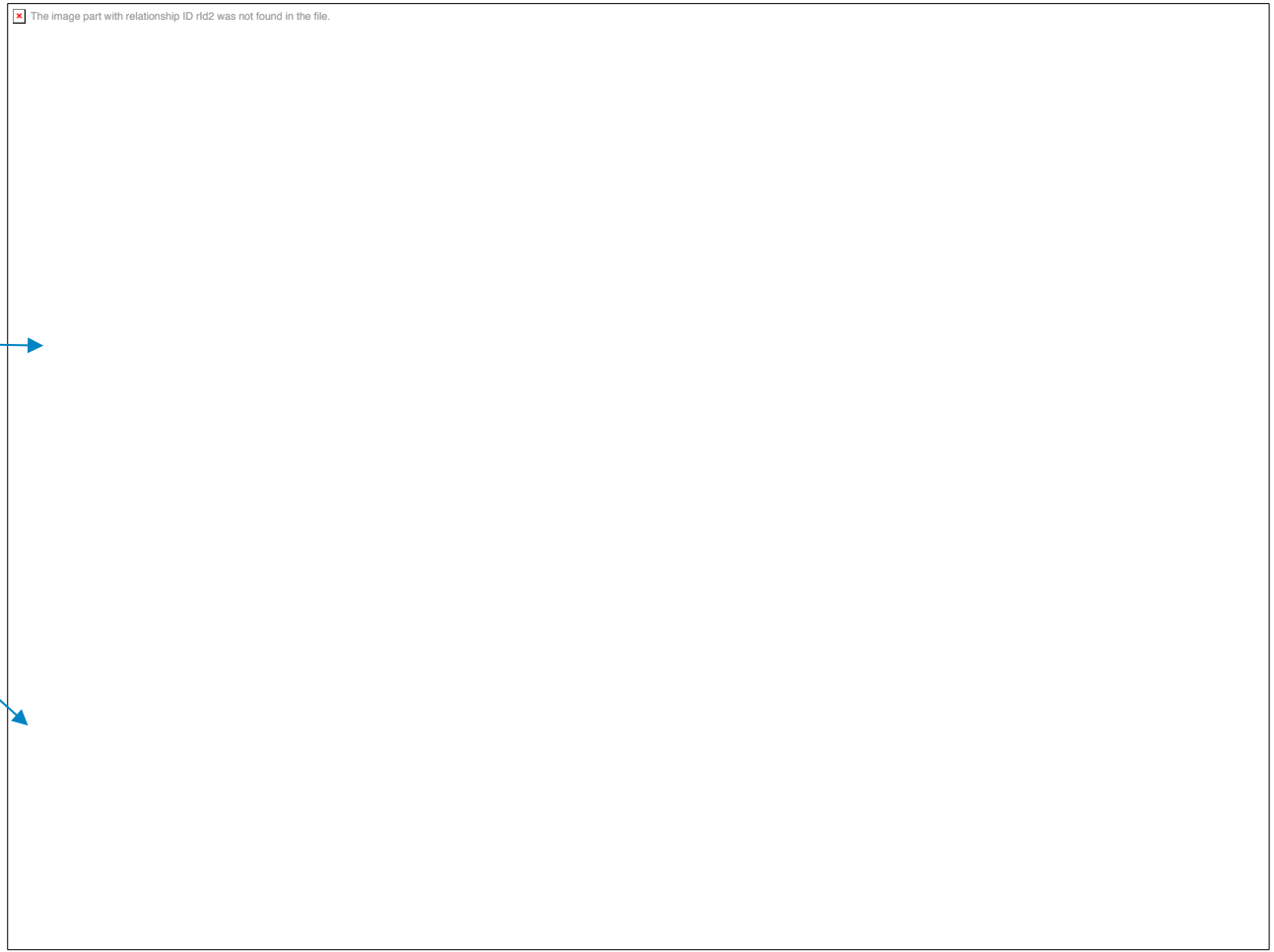
- Incomplete, e.g.
  - Business-APIs-as-a-Service: Payments is an example, but there are others, e.g.
    - Order Management, Inventory, Shipping, …
    - Calendar and events. Contact management.
    - Human Resources.
  - Information-as-a-Service
    - Demographics.
    - Credit history and information.
    - 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.



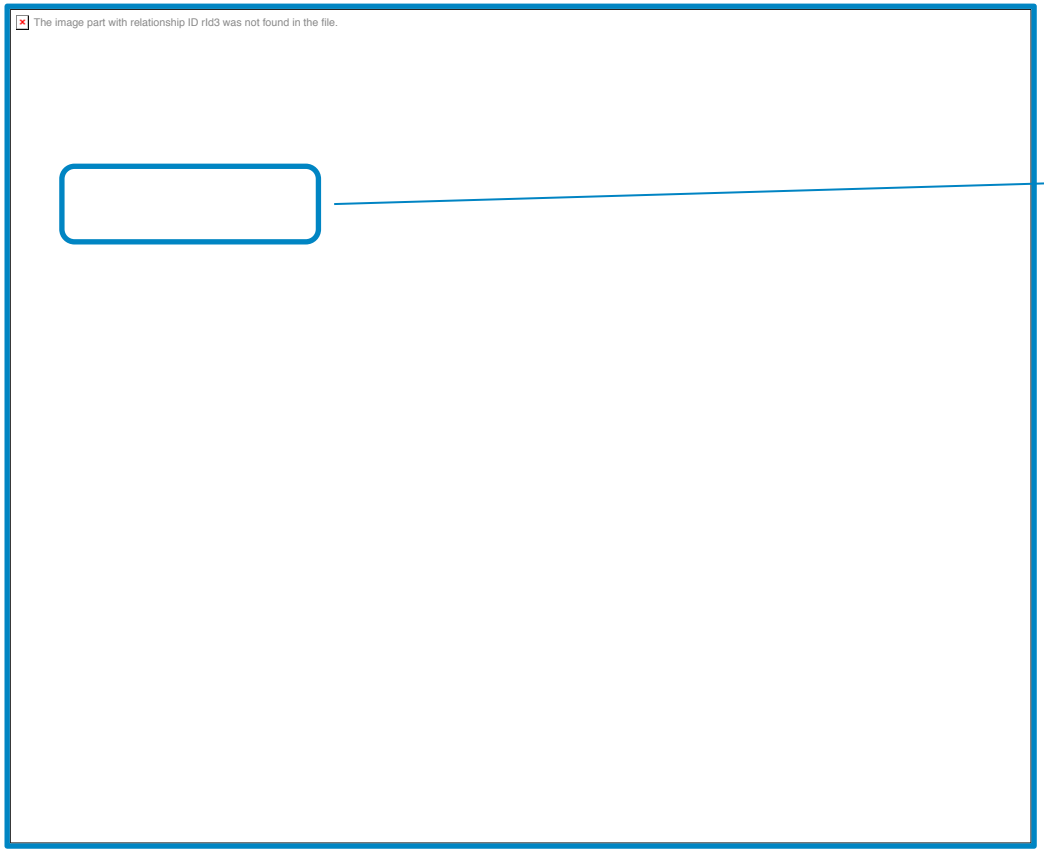
Cloudscape

Our applications will

- Call APIs
- Use and run in PaaS



www.programmableweb.com (Example)



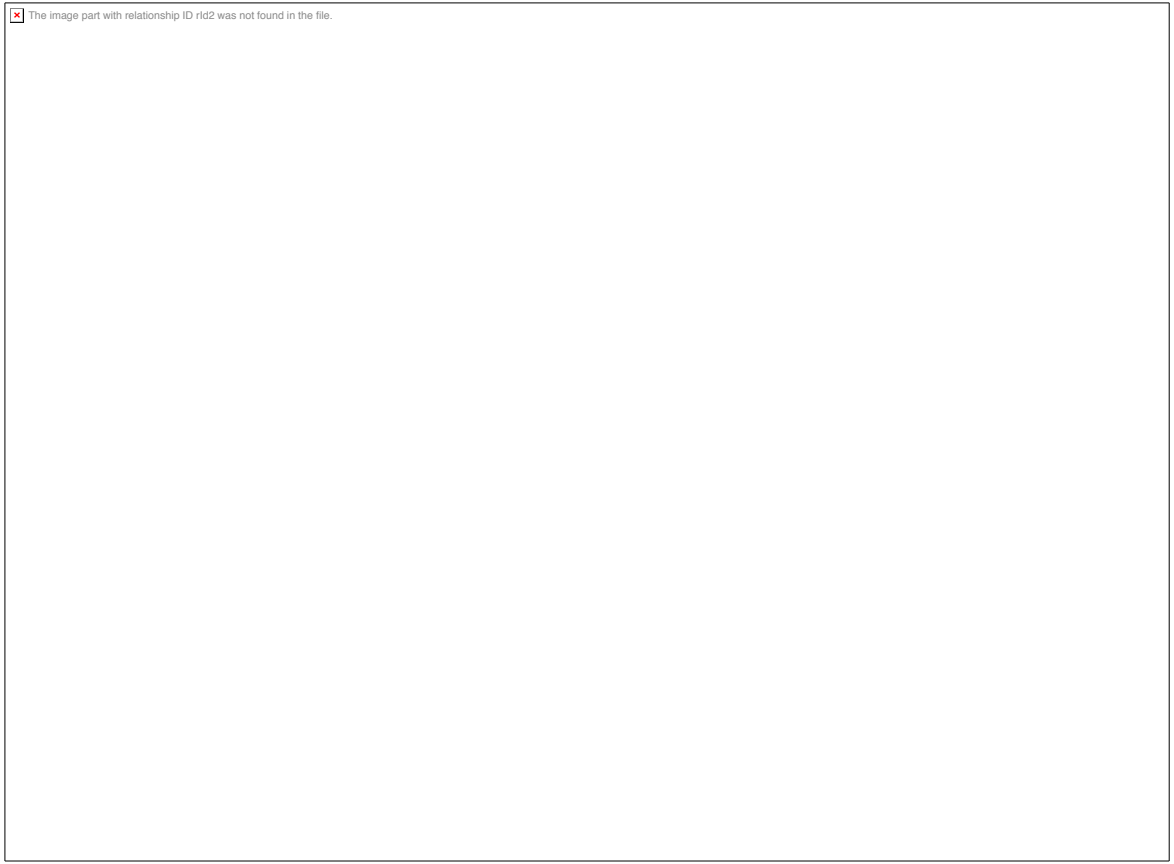
Over 16,000 APIs

Microservices

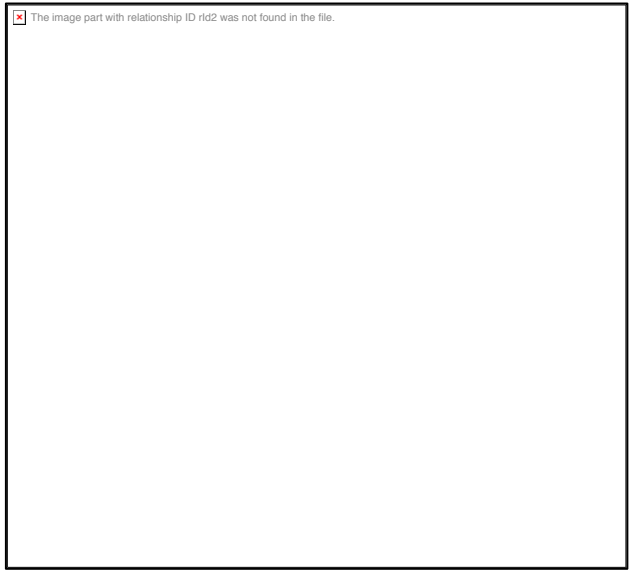
Microservices (https://martinfowler.com/articles/microservices.html)

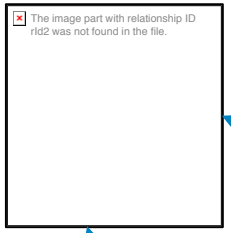
Web Application Basic Concepts

“Traditional” Web Application Architecture

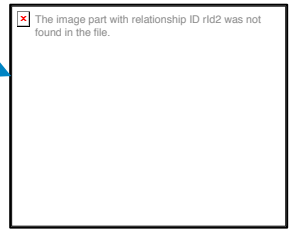


Monolithic to Micro

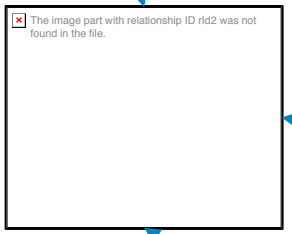




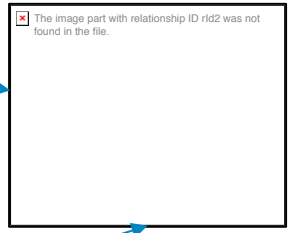
Cart Functions
- Java
- SQLite



Recommendation Functions
- Node.js
- Redis



Catalog Functions
- PHP
- MongoDB



XXX XXX
- MMM
- NNN

Content Functions
- Ruby
- Amazon S3

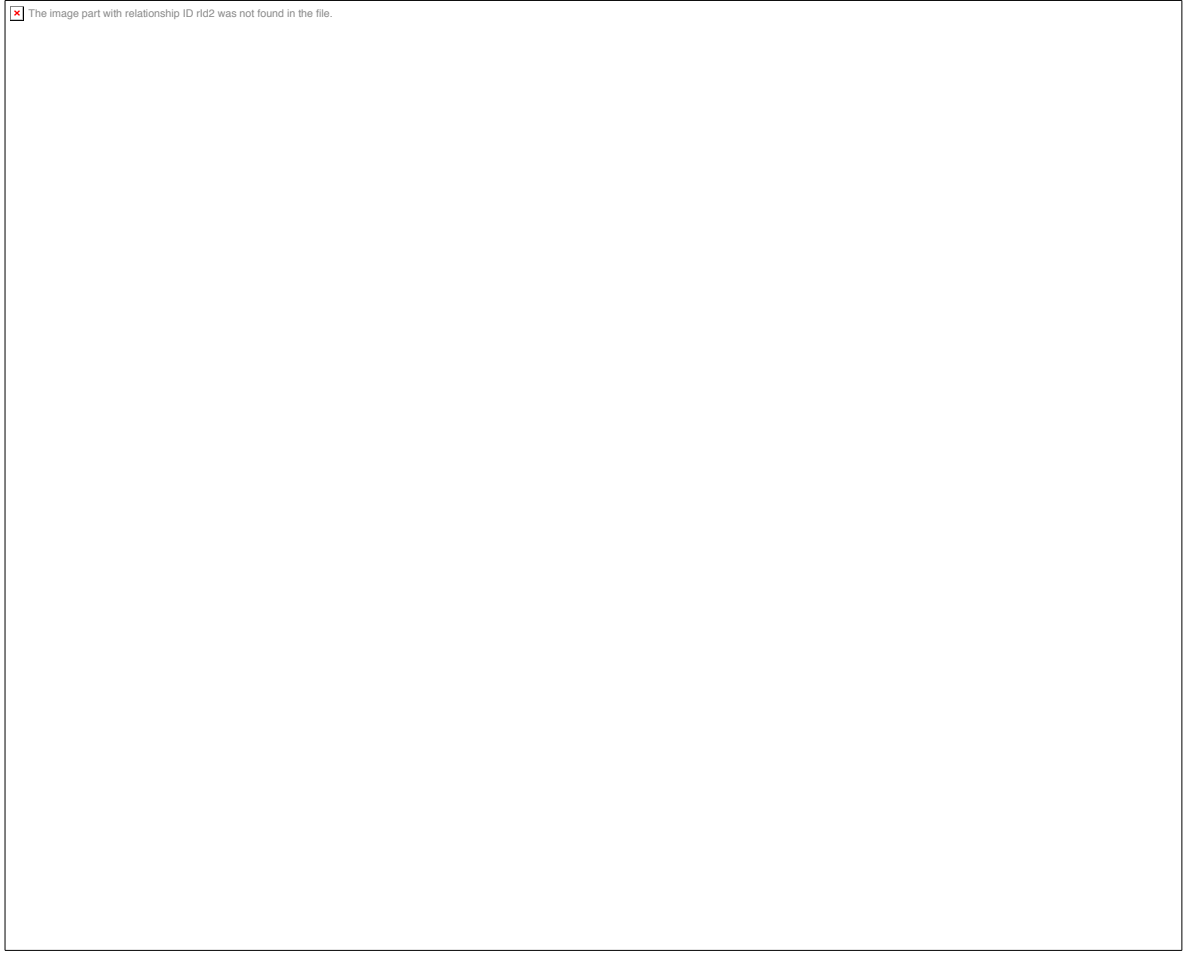
Motivations

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

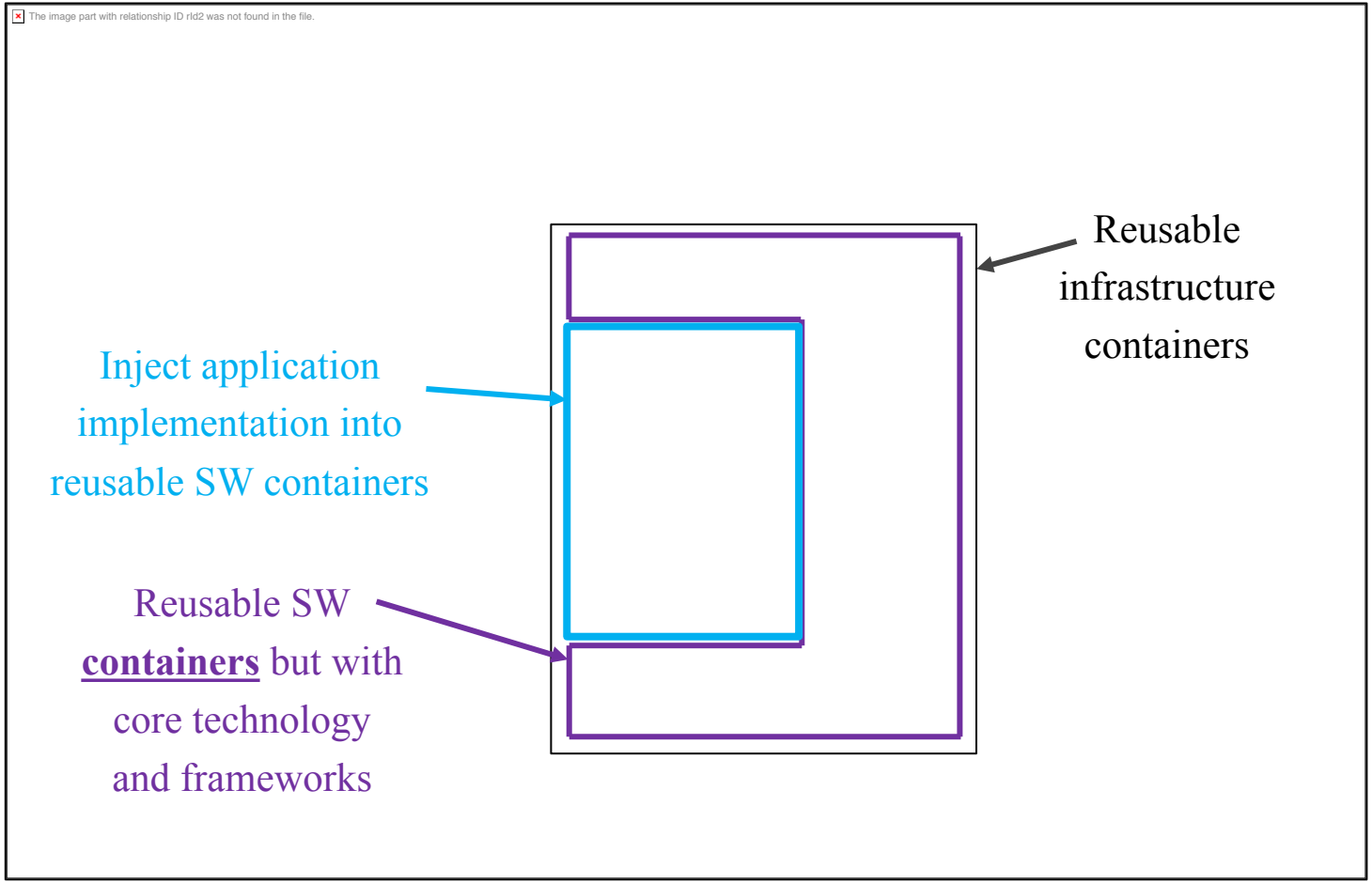
Micro-services Characteristics

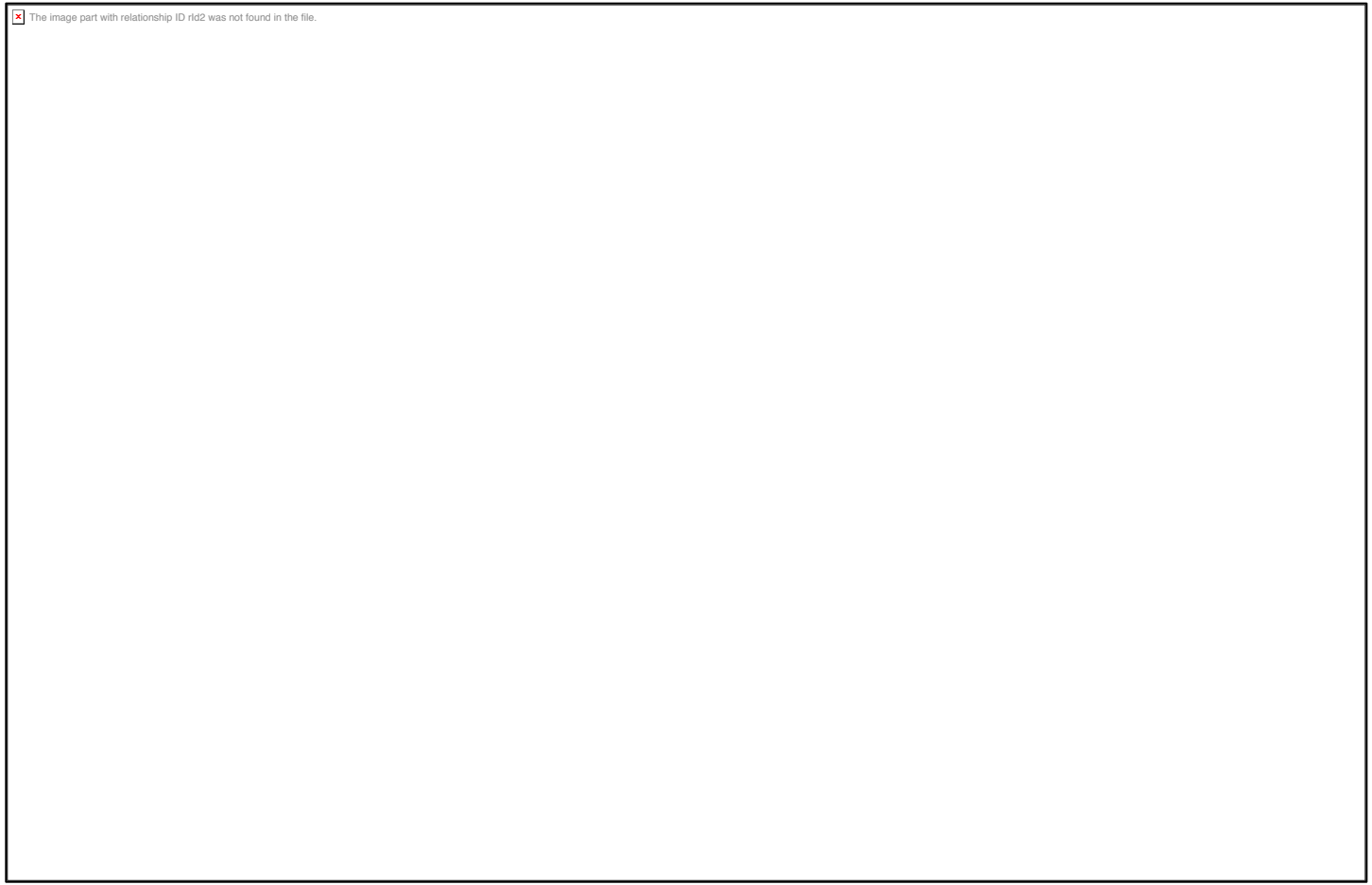
- Componentization via Services
- Organized around Business Capabilities
- Products not Projects
- Smart endpoints and dumb pipes
- Decentralized Governance
- Decentralized Data Management
- Infrastructure Automation
- Design for failure
- Evolutionary Design

SOA vs Microservices



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





Patterns

Patterns

The image part with relationship ID rId2 was not found in the file.

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, …

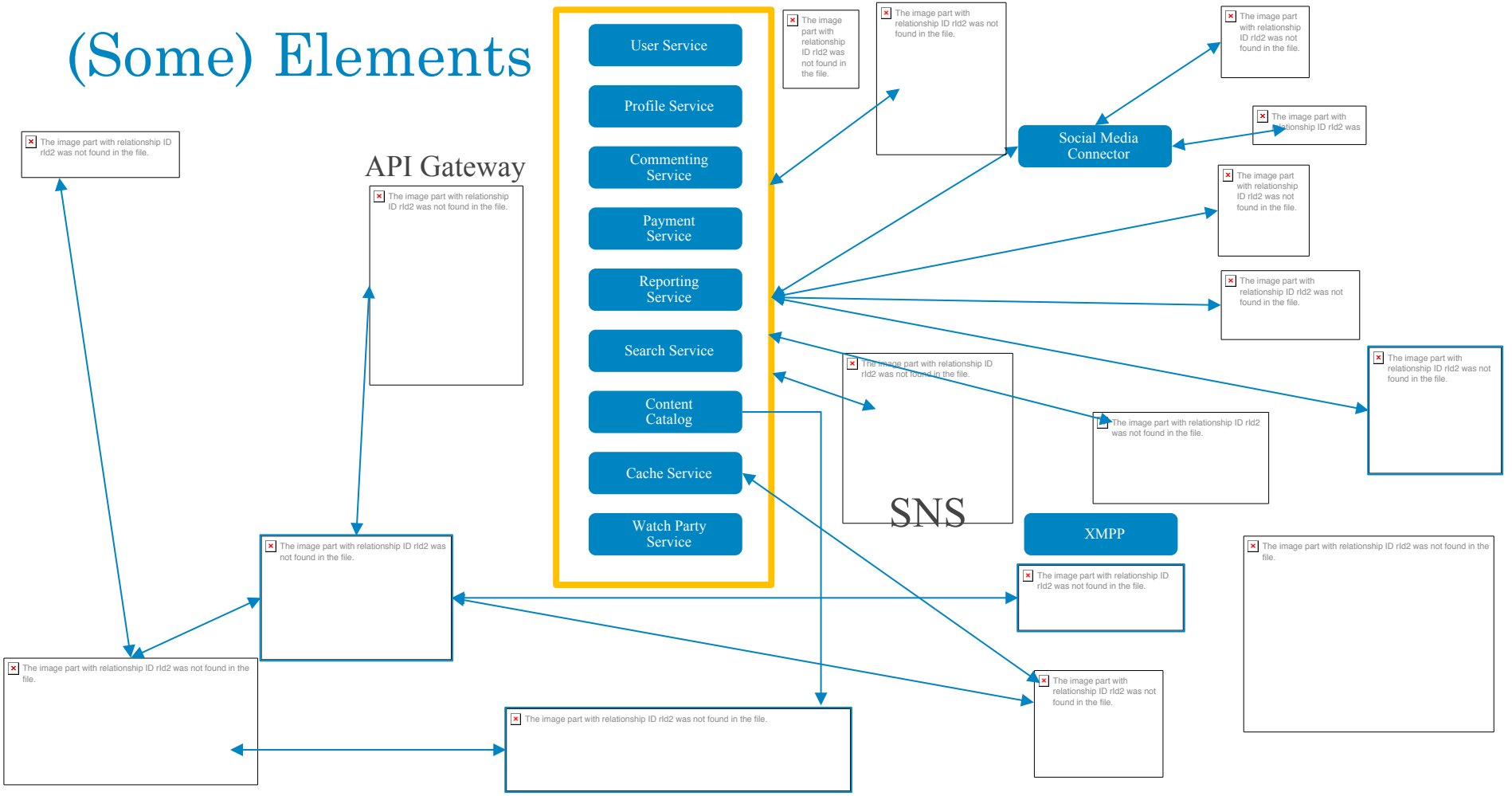
Weinberg's Second Law

"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

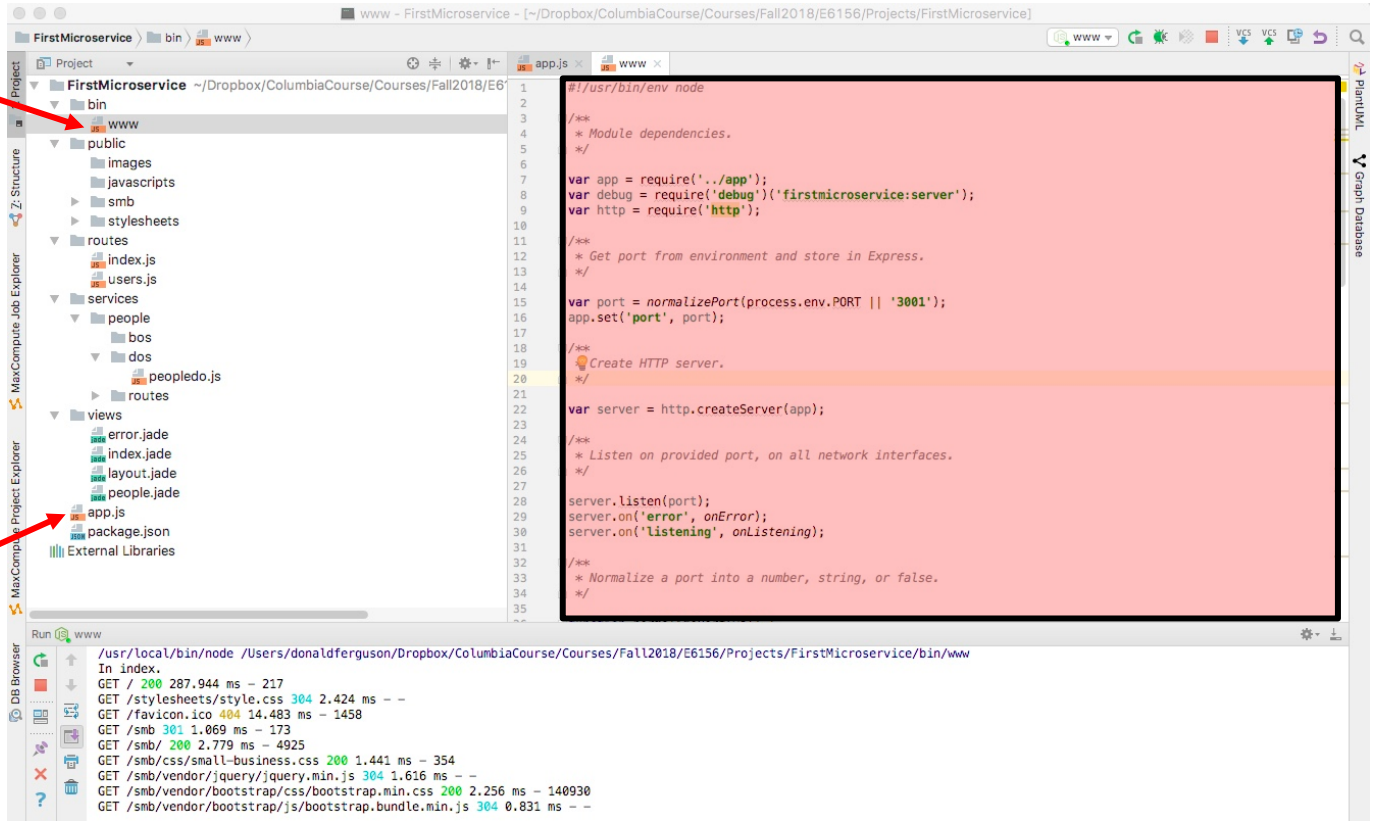
0th Project

- 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.

Project Structure

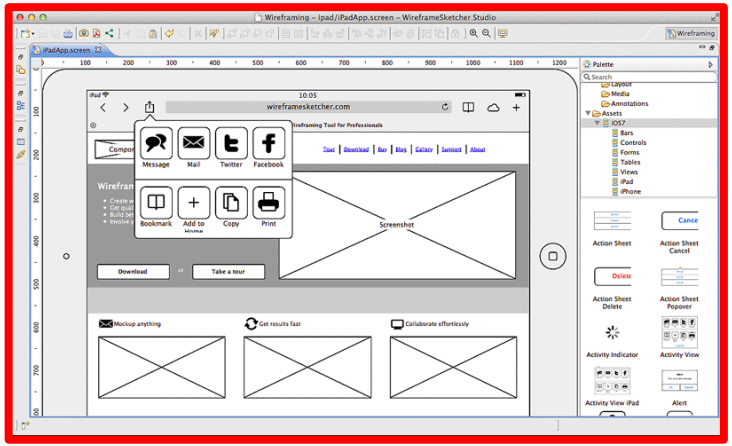
Server setup

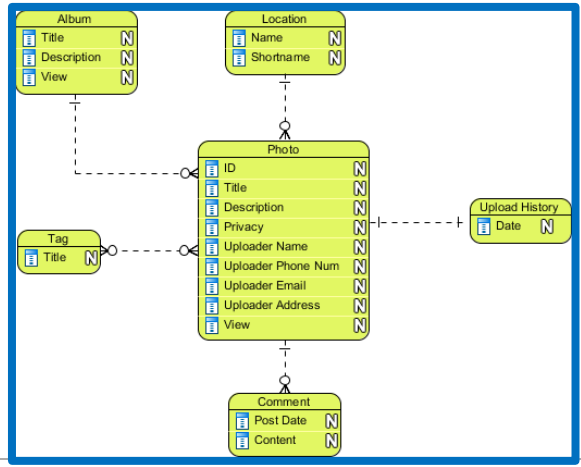
Application setup



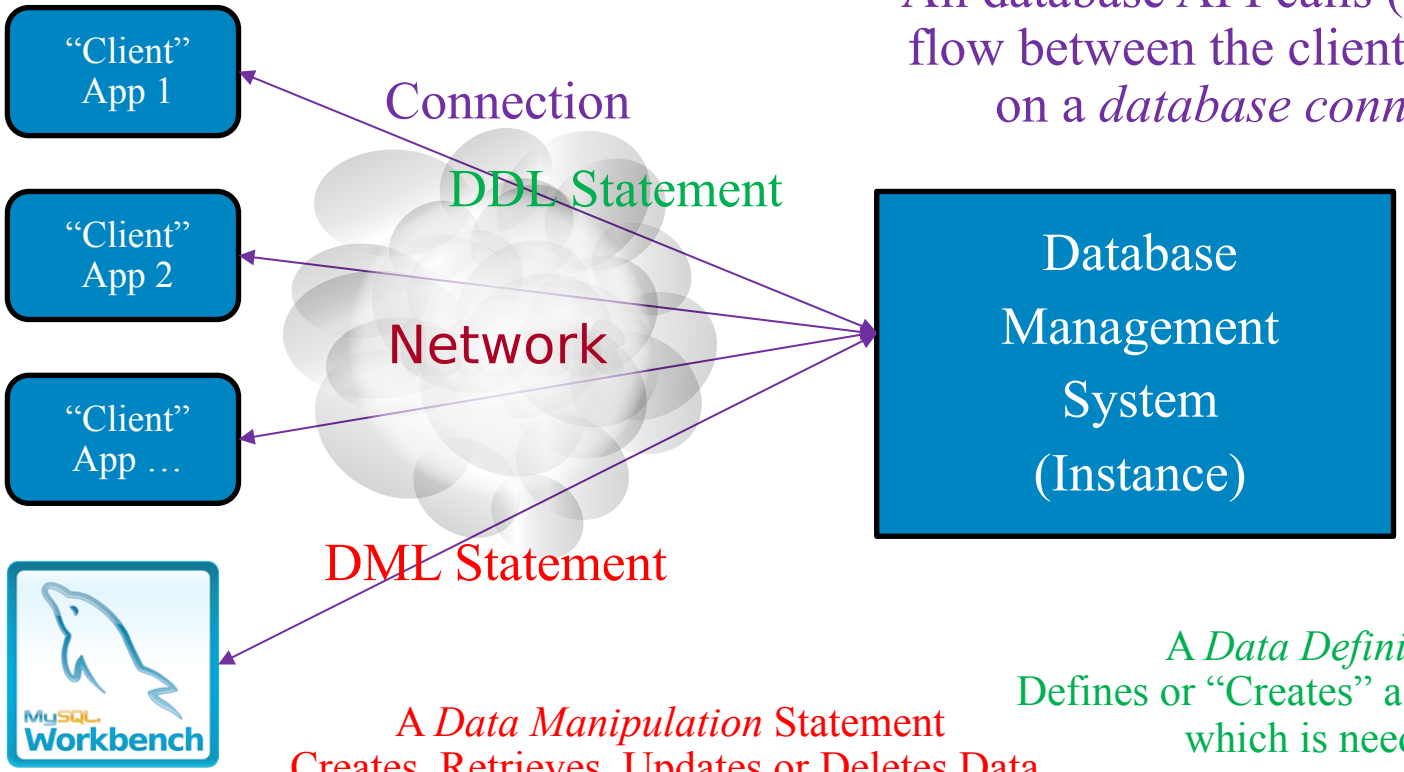
Application Design Methodologies
(Simplistic, Somewhat Dated).
(But, you get the idea).

- Start with User Interface
  - Roles and personas.
  - Wireframes for pages and interactions.
  - Page flow/transition diagrams.
  - Implement a model layer supporting UI
    - Functions
    - Data
- Start with Data Model
  - Entity types, properties.
  - Relationships/associations.
  - Constraints, integrity rules
  - Define reusable core operations.
  - Design role/task UIs that use core operations





Database Connection

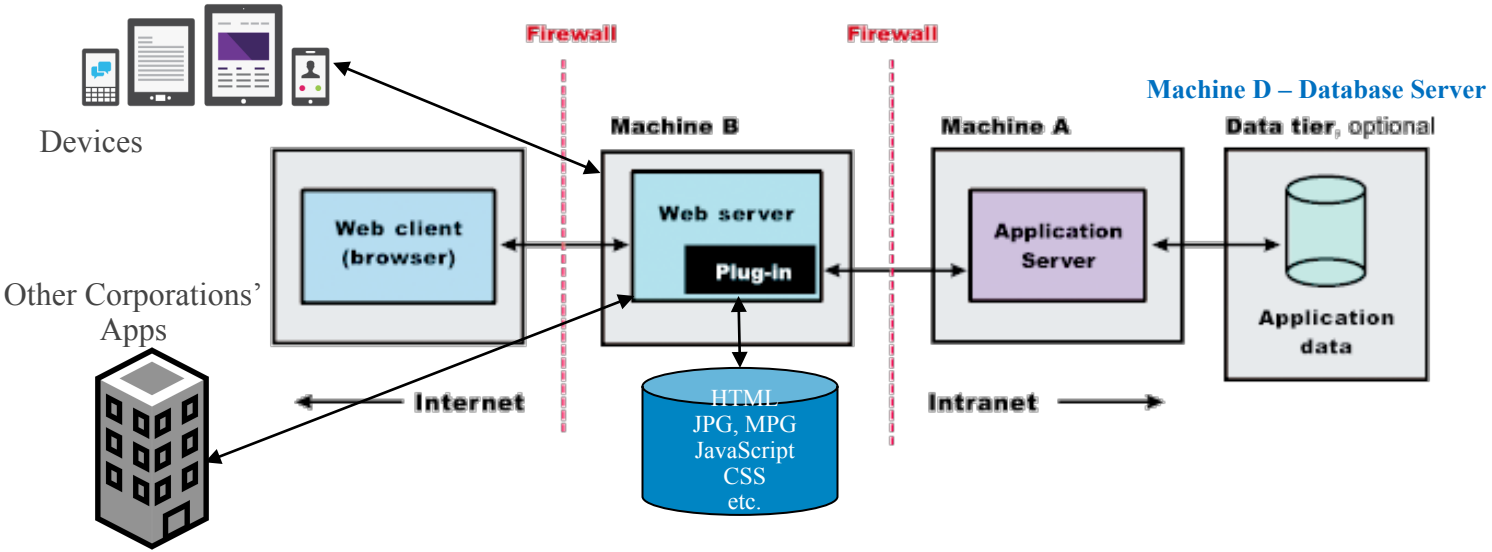


All database API calls (statements) flow between the client and server on a *database connection*.

A *Data Manipulation* Statement
Creates, Retrieves, Updates or Deletes Data.

A *Data Definition* Statement
Defines or "Creates" a definition/type of data, which is needed for DML.

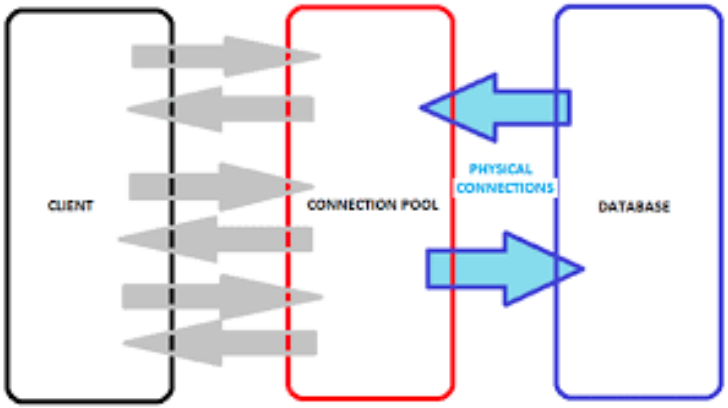
Web/Internet Application Structure

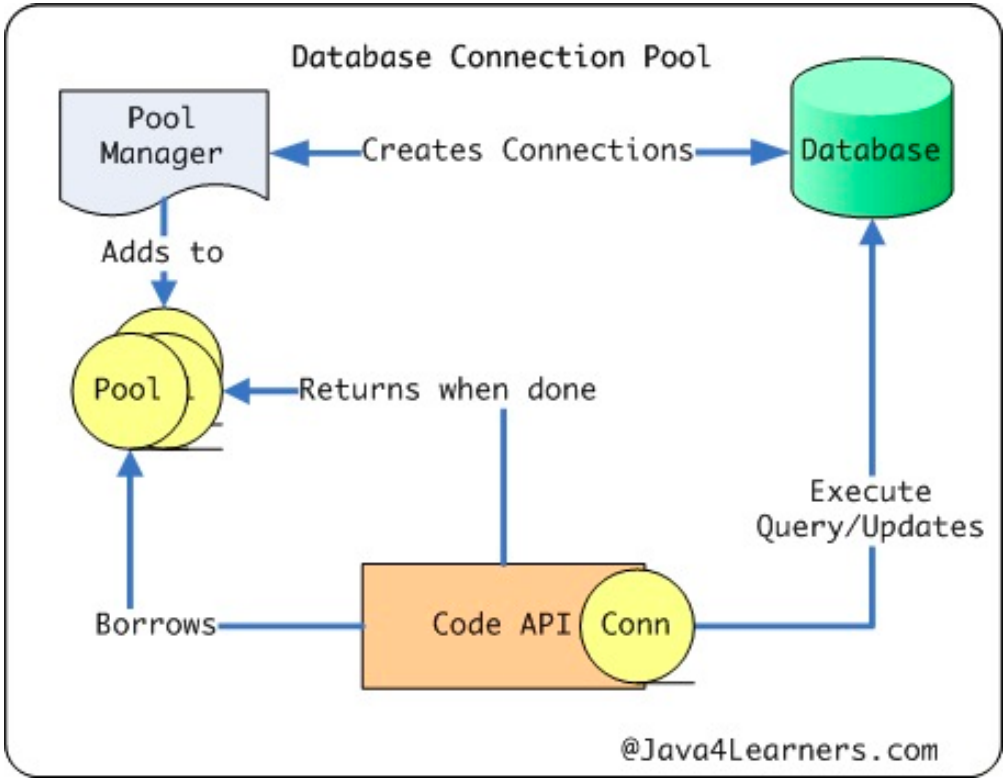


- Accounts
- Products
- User profiles
- Registrations
- Course schedule
- etc.

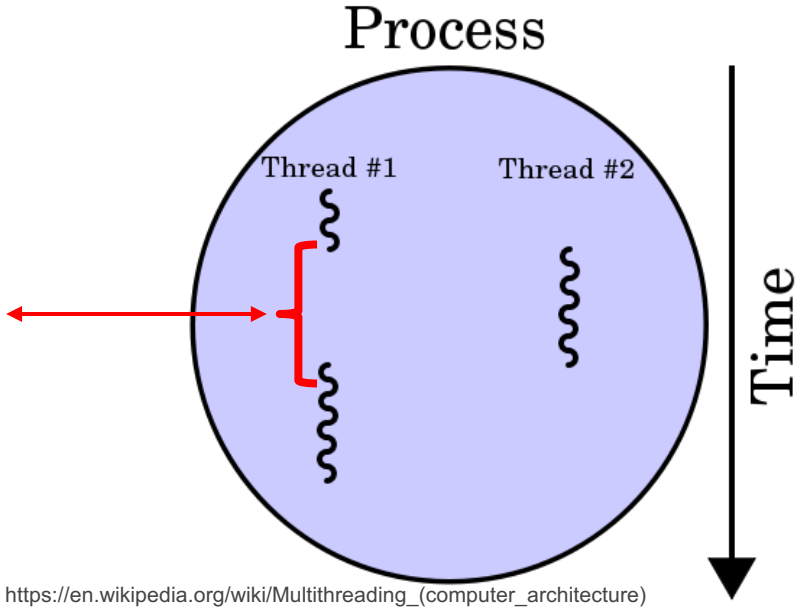
- There are 100s, 1000s, millions of simultaneous user sessions.
- There may be 100s, 1000s of servers at each tier to achieve
  - Scalability (throughput, response time)
  - Availability, fault tolerance.
- Servers may be spread over multiple data centers, public/private clouds, etc.

Connection Pool



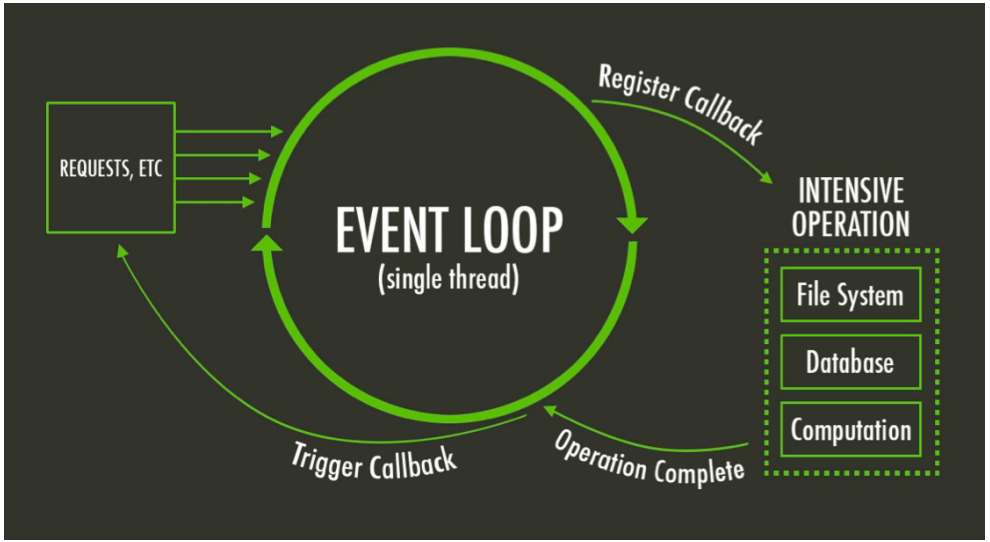


Multithreading and Callbacks



https://en.wikipedia.org/wiki/Multithreading_(computer_architecture)

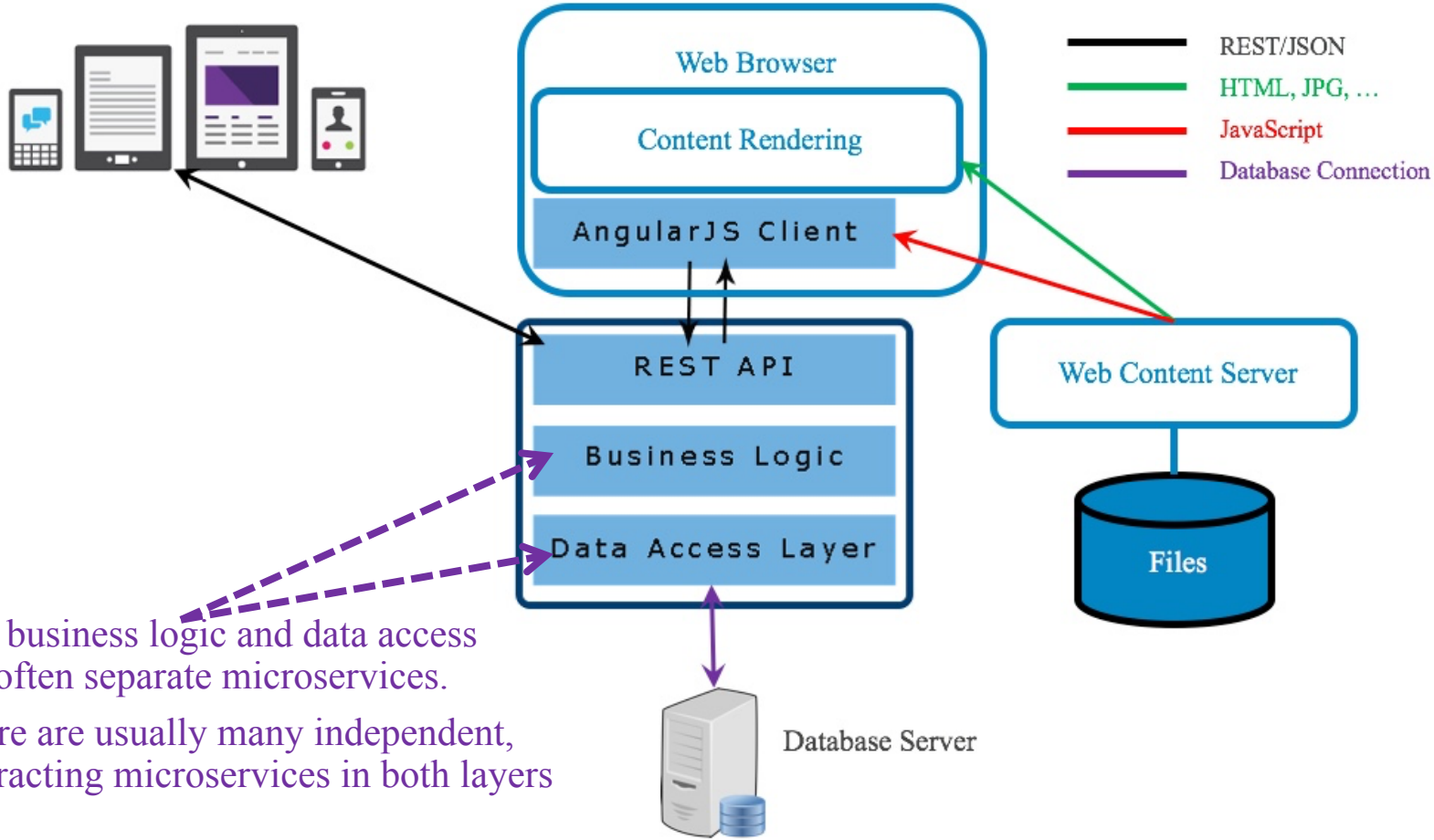
- Multithreading
  - Thread #1 “blocks” for an IO or API call.
  - Process switches to “ready” thread #2



- Callbacks

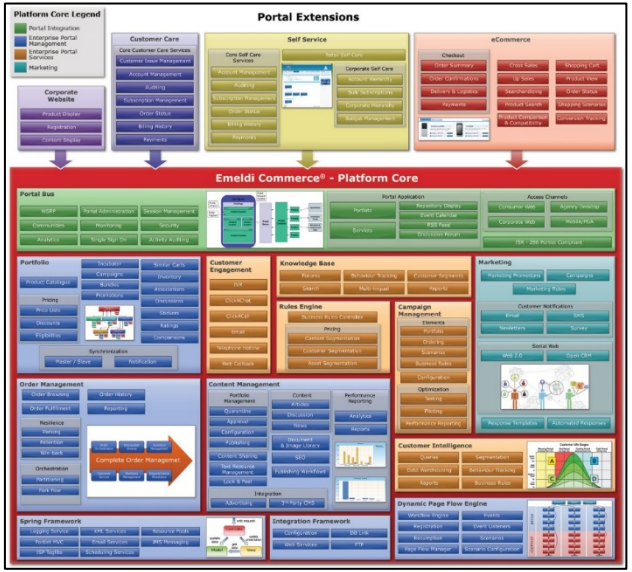
https://medium.com/@amragaey/shortly-how-node-js-works-on-a-single-thread-763fda99f012

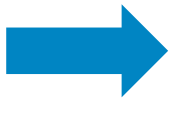
  - Incoming requests map to an event.
  - There is a single thread running application code associated with an event..
  - Long running calls get assigned to a separate worker thread, and register a callback.
  - Completion of an operation results in an event, which resumes the application via a callback.

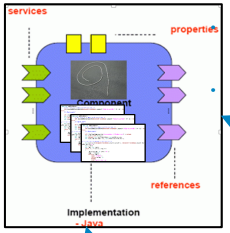


- The business logic and data access are often separate microservices.
- There are usually many independent, interacting microservices in both layers

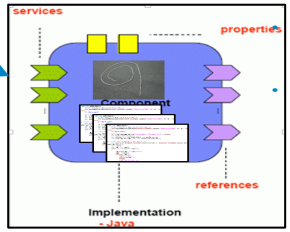
Monolithic to Micro



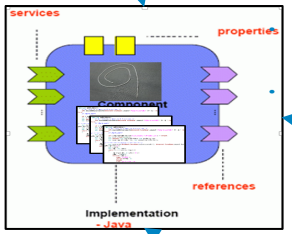




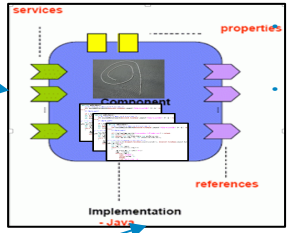
Cart Functions
- Java
- SQLite



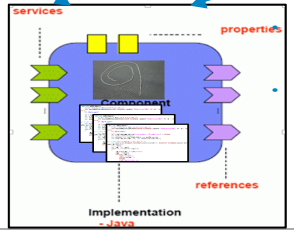
Recommendation Functions
- Node.js
- Redis



Catalog Functions
- PHP
- MongoDB



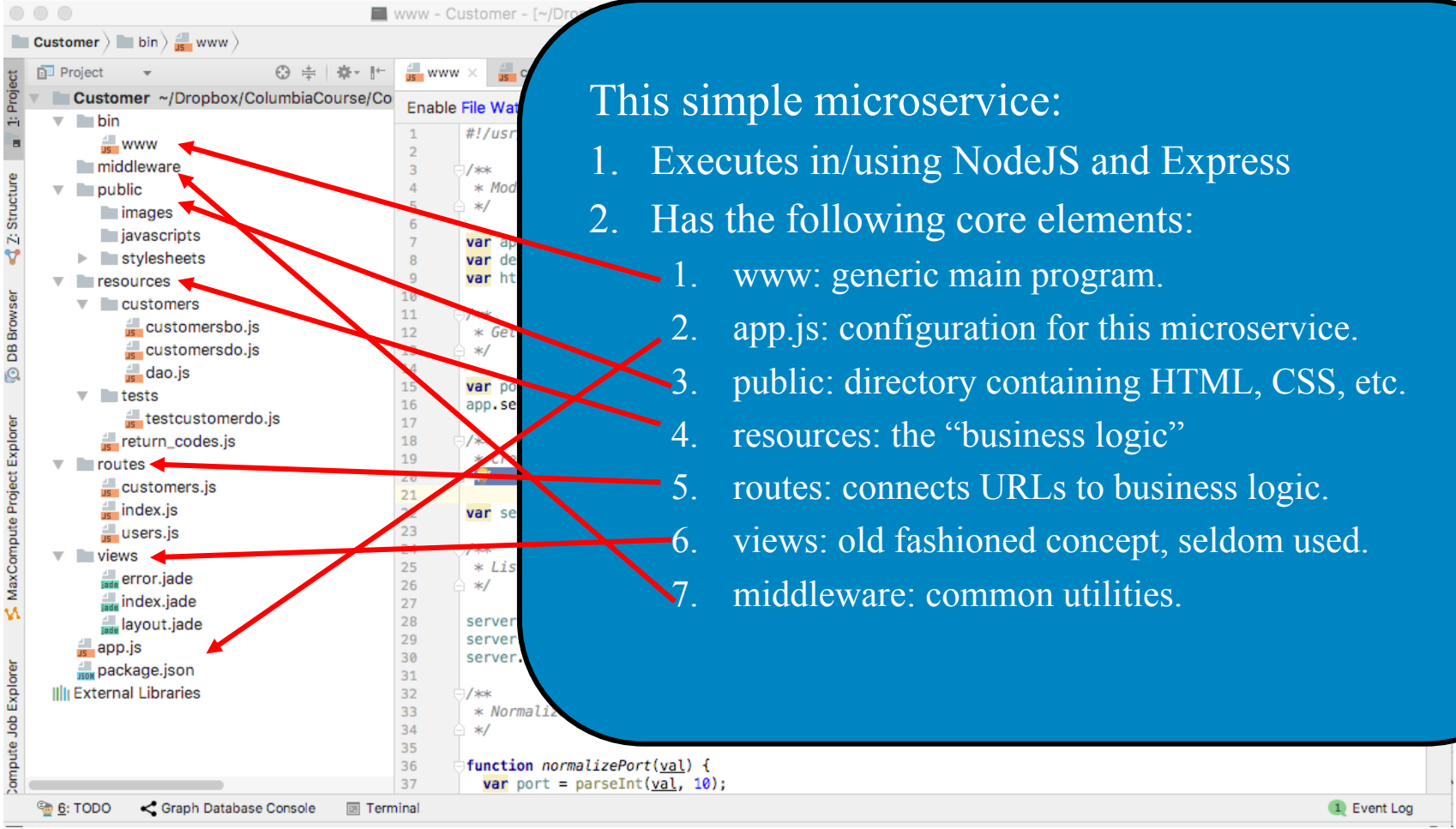
XXX XXX
- MMM
- NNN



Content Functions
- Ruby
- Amazon S3

Motivations

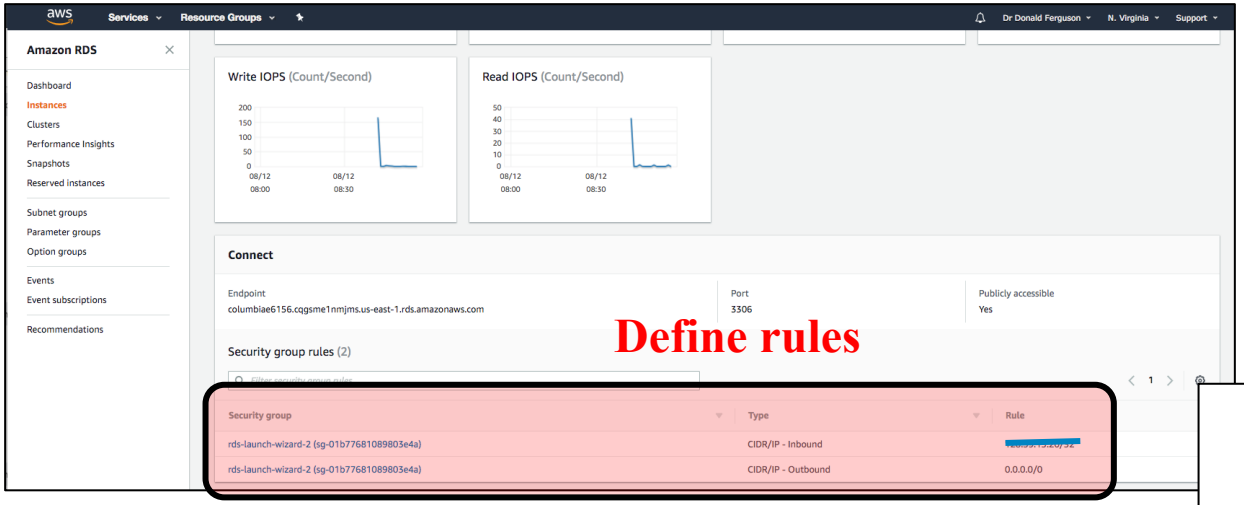
- Enable best tools, languages, … for module.
  - Polyglot persistence
  - Polyglot programming
- Simplifies change management and evolution.
- Better alignment of "apps" with business functions.
- Reuse of code and internet services.



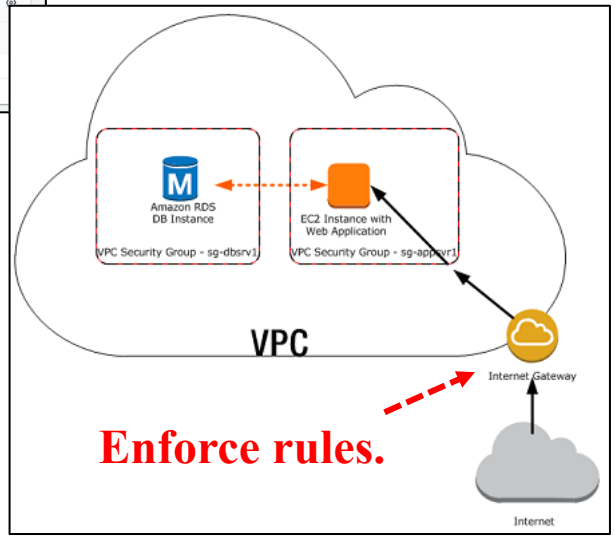
This simple microservice:

1. Executes in/using NodeJS and Express
2. Has the following core elements:
    1. www: generic main program.
    2. app.js: configuration for this microservice.
    3. public: directory containing HTML, CSS, etc.
    4. resources: the "business logic"
    5. routes: connects URLs to business logic.
    6. views: old fashioned concept, seldom used.
    7. middleware: common utilities.

RDS/EC2 Security Groups



Define rules



Enforce rules.

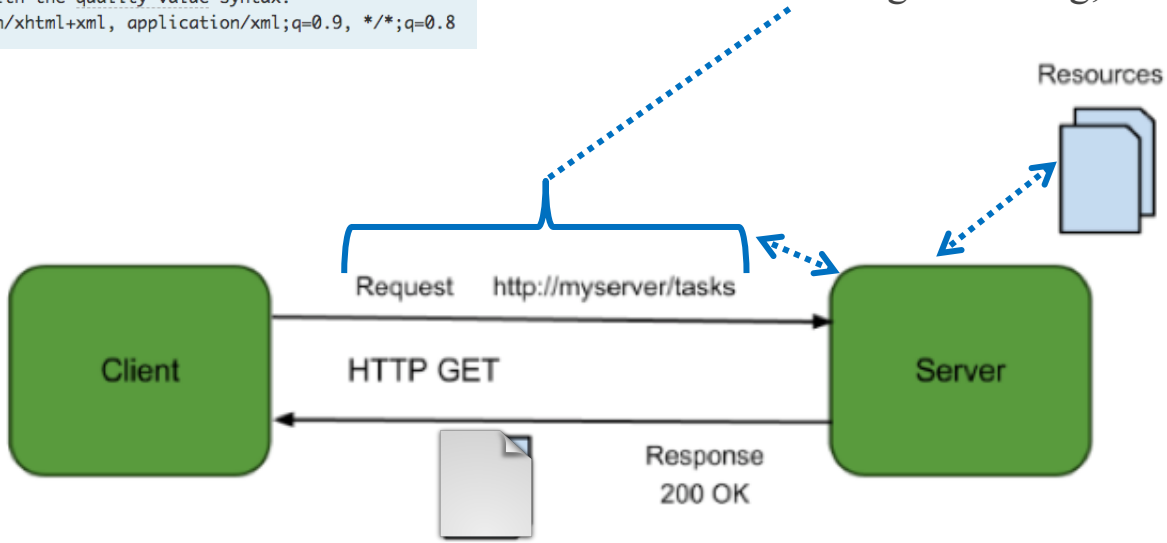
- RDS (and EC2) instances are in a security group.
- Security groups has several functions, including
  - Preventing network access to group resources.
  - Unless the source address is on a whitelist.
- A gateway enforces these rules, including access from the public Internet.

Accept type in headers.

```
Accept: <MIME_type>/<MIME_subtype>
Accept: <MIME_type>/*
Accept: */*

// Multiple types, weighted with the quality value syntax:
Accept: text/html, application/xhtml+xml, application/xml;q=0.9, */*;q=0.8
```

- Relative URL identifies "resource" on the server.
- Server implementation maps abstract resource to tangible "thing," file, DB row, …



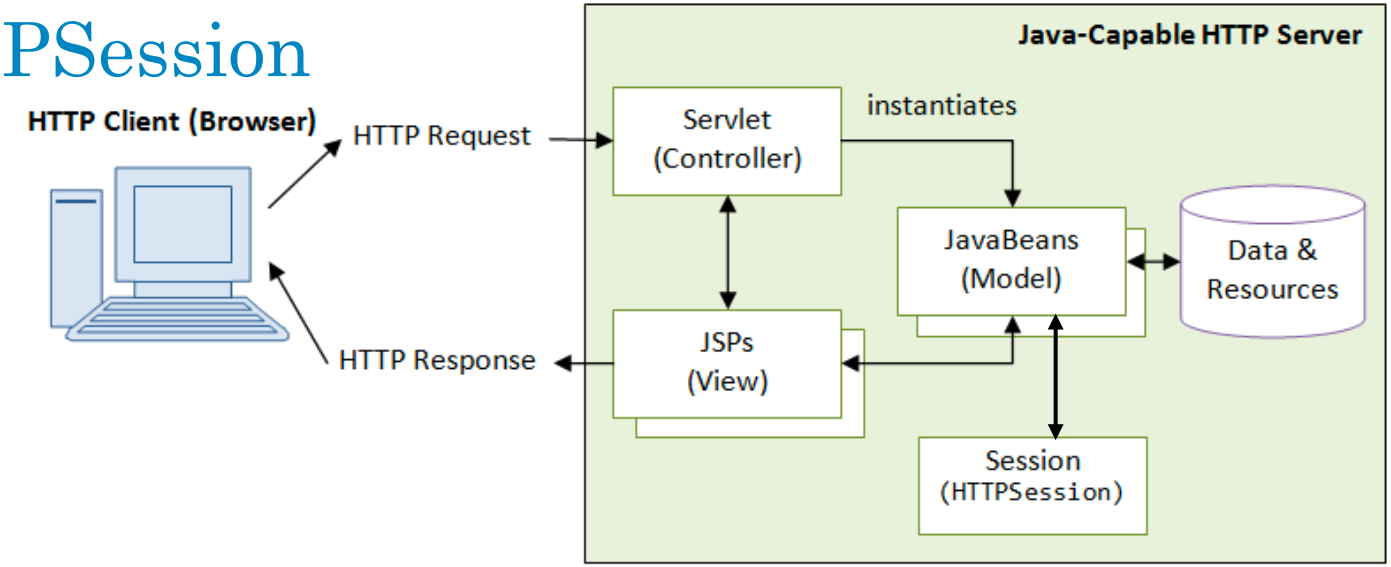
Client may be

- Browser
- Mobile device
- Microservce
- … …

Web media (content) type, e.g.

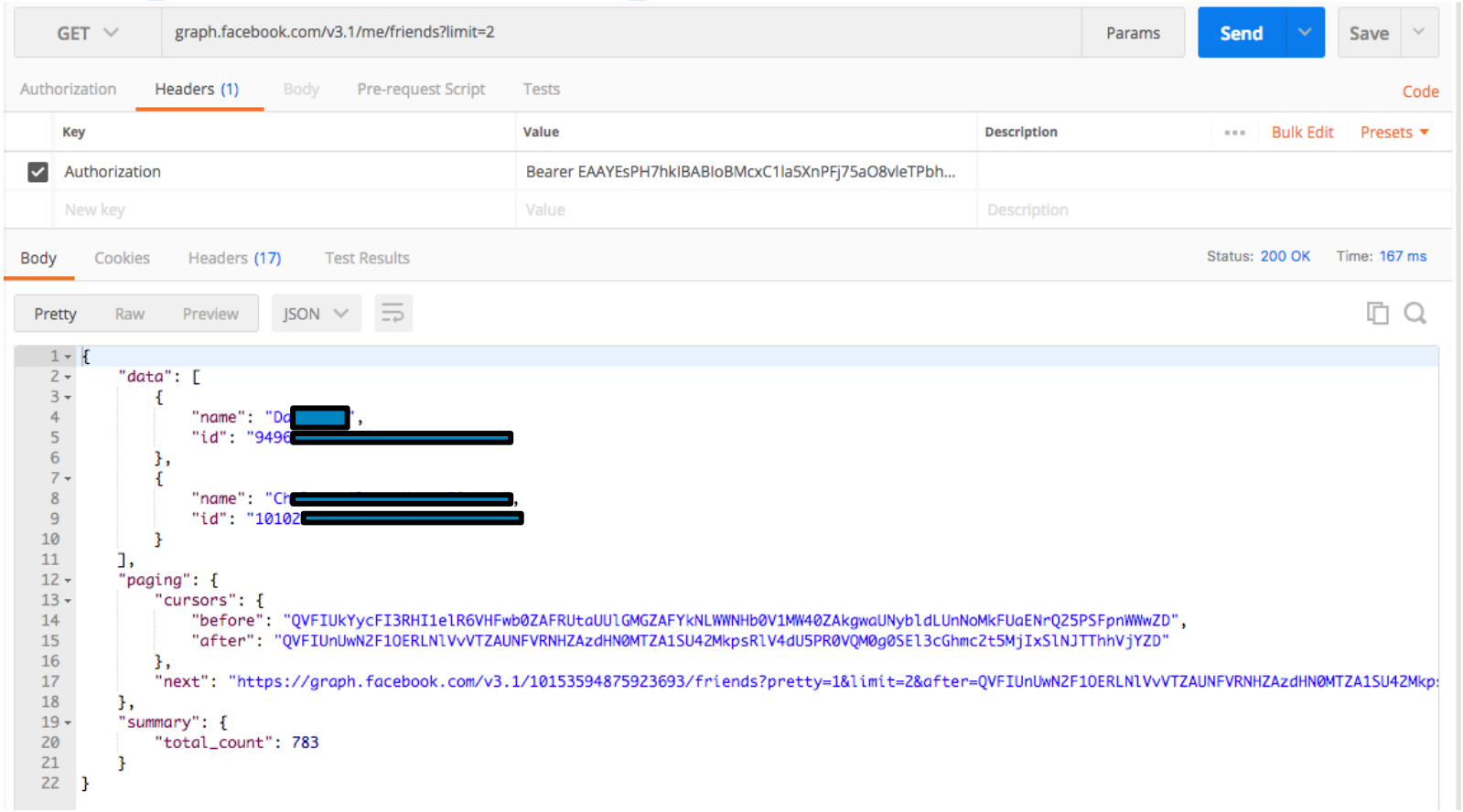
- text/html
- application/json

HTTPSession



- Server can start a session based on login, first request from IP address, specific request, …
- Session has an ID.
- Server can save data in the HTTPSession and access via ID.
- Server reads and updates HTTPSession on each request.
- The response to request N is a function of:
  - Request parameters.
  - Session information from previous requests.
  - Resource state

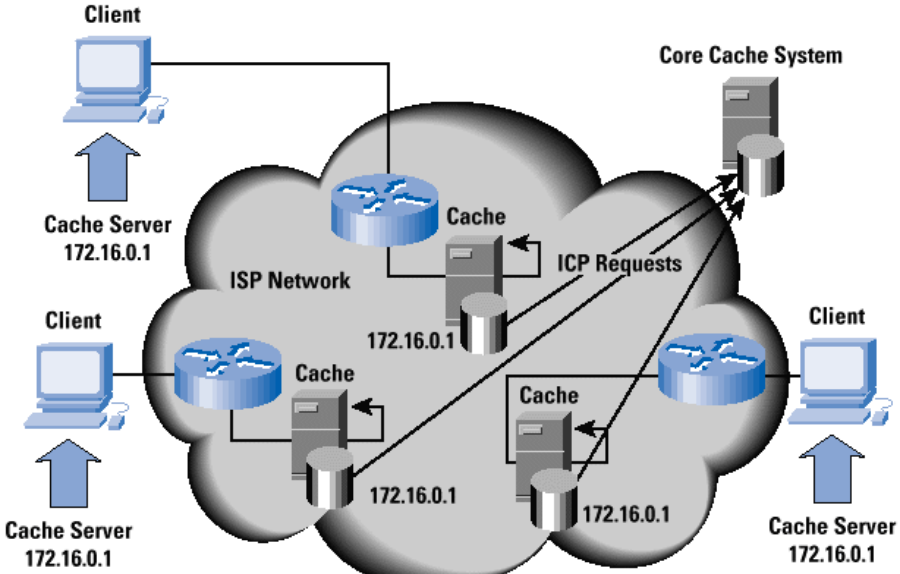
Sample Facebook Operation



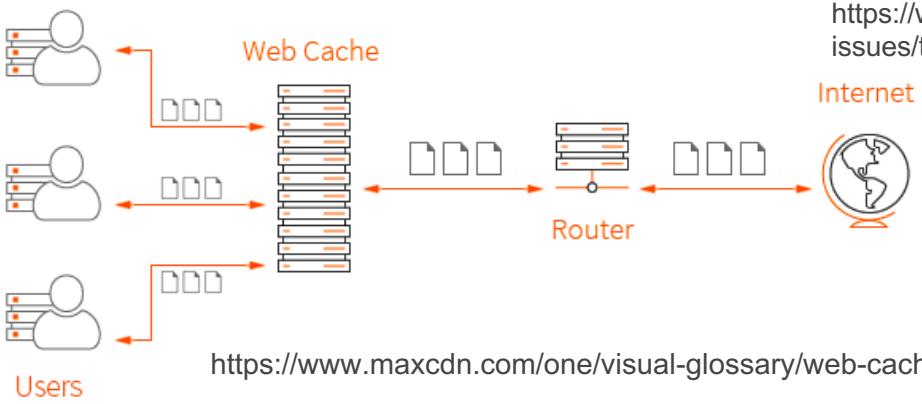
Cacheability

There are caches everywhere:

- Browser client.
- Enterprise perimeter.
- Content delivery network (CDB)
- ISP nodes.



https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-2/ipj-archive/article09186a00800c8903.html



https://www.maxcdn.com/one/visual-glossary/web-cache/