Introduction to Cloud Applications (I)

Lecture 1: Introduction, Core Concepts

Introduction and Overview

Introduction

- The first 10 lectures are part I and foundational. The lectures:
  - Are a prerequisite to part II and III that cover deeper concepts.
  - Some may find the material very basic. These students will find the primary value to be how to realize concepts on clouds, initially Amazon Web Services.
- There are many, many perspectives on this course's material.
  - The lectures represent my perspective and suggestions based on experience.
  - There are many, many other perspectives and approaches.
  - One goal is to help you form your (or your teams') perspectives.
- We are going to build a simple, cloud-native application.
  - The application will be broad to allow you to play with many technologies and concepts.
  - Some of the work may seem odd because the benefit surface in more complex applications and scenarios.

Introduction to Core Concepts

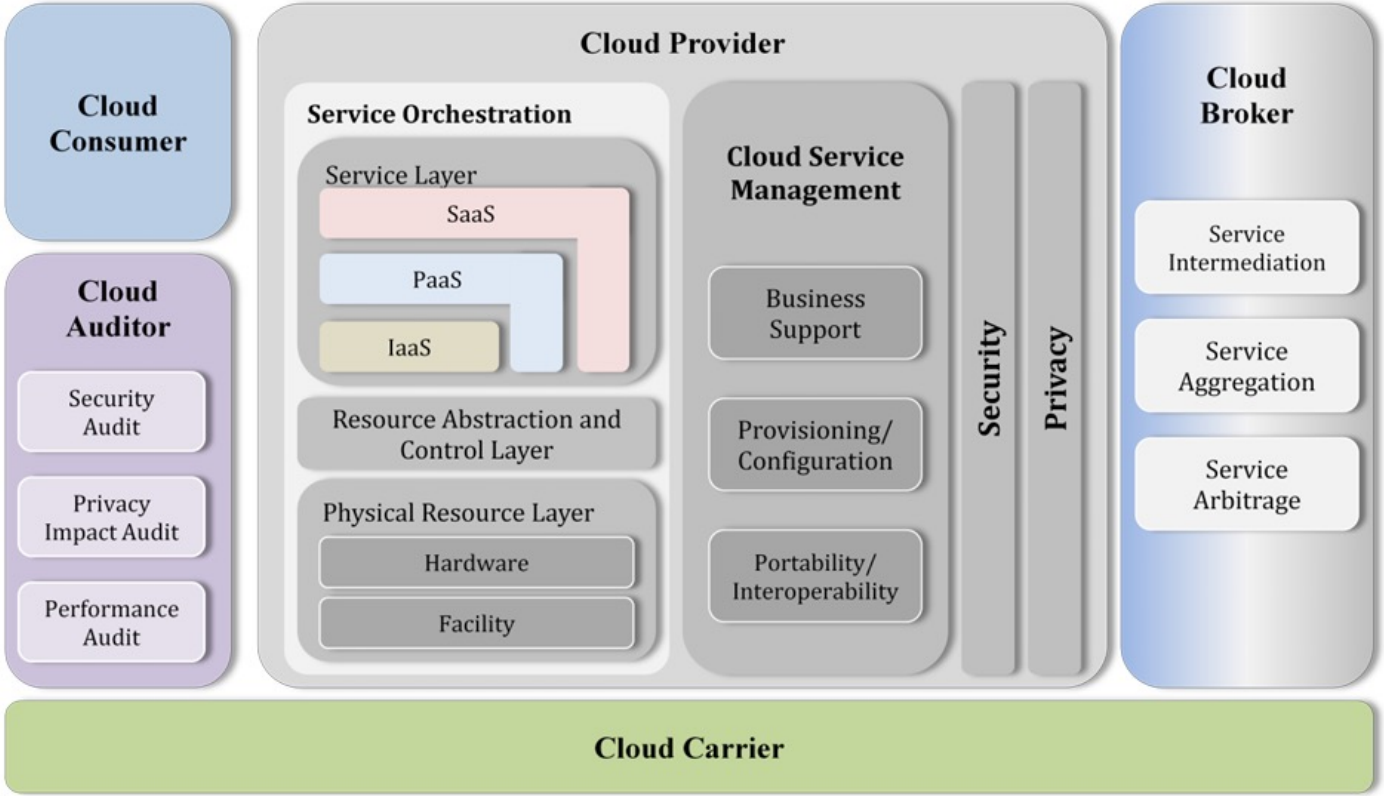
Cloud Concepts – One Perspective

Categorizing and Comparing the Cloud Landscape

http://www.theenterprisearchitect.eu/blog/2013/10/12/the-cloud-landscape-described-categorized-and-compared/

6	**SaaS**	Applications			*End-users*
5	**App Services**	App Services	Communication and Social Services	Data-as-a-Service	*Citizen Developers*
4	**Model-Driven PaaS**	Model-Driven aPaaS, bpmPaaS	Model-Driven iPaaS	Data Analytics, baPaaS	*Rapid Developers*
3	**PaaS**	aPaaS	iPaaS	dbPaaS	*Developers / Coders*
2	**Foundational PaaS**	Application Containers	Routing, Messaging, Orchestration	Object Storage	*DevOps*
1	**Software-Defined Datacenter**	Virtual Machines	Software-Defined Networking (SDN), NFV	Software-Defined Storage (SDS), Block Storage	*Infrastructure Engineers*
0	**Hardware**	Servers	Switches, Routers	Storage	
		Compute	Communicate	Store	

Cloud Concepts – Another Perspective



NIST Cloud Computing Reference Architecture

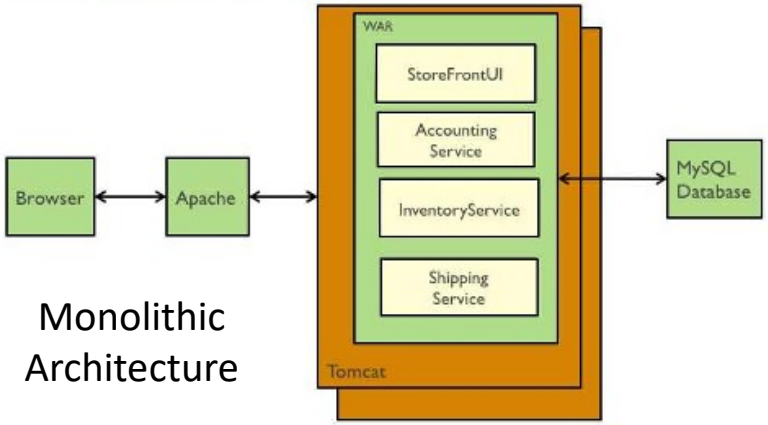
Microservices (https://microservices.io/index.html)

What are microservices?

Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities
- Owned by a small team

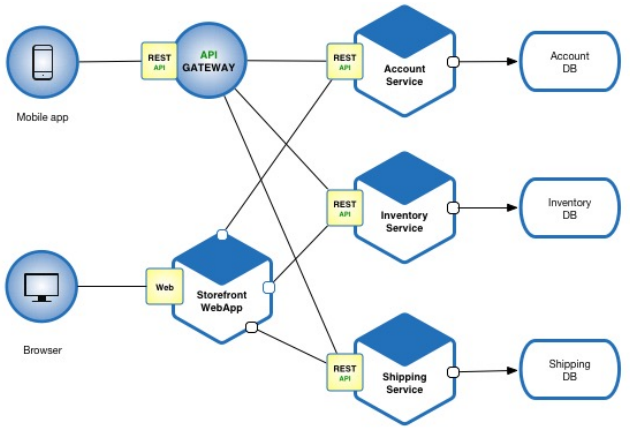
The microservice architecture enables the rapid, frequent and reliable delivery of large, complex applications. It also enables an organization to evolve its technology stack.



Monolithic Architecture

- The literature discussed microservices in the context of "web applications."
- The approach does apply to building CAE products.
- We think of our products as a logical set of subsystems/components. Microservices
  - Replace "language runtime linking"
  - With a more flexible approach to
  - Product development and assembly.

Microservices Architecture



Classification, Definitions, Pros/Cons

What is true of any:

- Mapping, classification, taxonomy, ...?
- Technology definition, ...?
- Enumeration of technology pros and cons ... ...?

- When you transform monolithic systems to
- Containers, microservices, cloud, ...
- You initially have some pretty “Macro” microservices and containers.
- APIs, SaaS and the cloud allow you to “hide a lot of ugly by putting ‘lipstick on the pig.’”
- Transformation occurs through continuous improvement.

Answers:

- “... ... it is ... ...
  More honor'd in the breach than the observance”
- If you have three subject matter experts, you have 7 conflicting opinions.
- ”The most dangerous thing in the world is a 2nd LT with a map.”
  Like any map, it is both a way of orientating yourself and a way of having a spectacular disaster because you are studying the map instead of thinking.

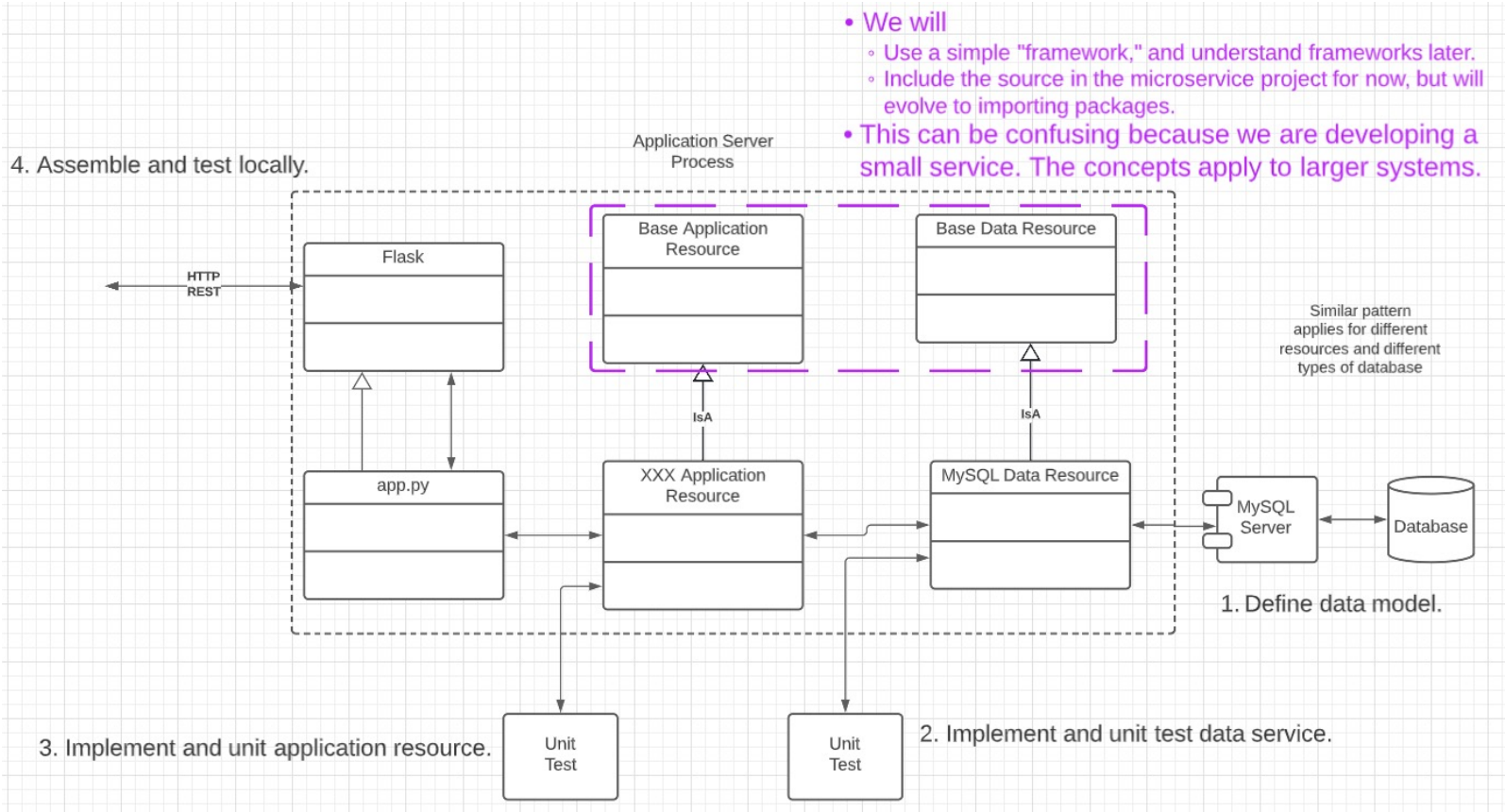
First Application and Microservice

First Microservice

There are four distinct phases. The workflow becomes more sophisticated later:

1. Develop and test SW: In this example:
    1. Develop and unit test:
        1. Database
        2. Data service
        3. Application “resource,” which will be foundation for REST
    2. Develop and test SW system by assembling components and application application container to form microservice.
2. Define and configure Infrastructure-as-a-Service:
    1. Virtual machine (server, storage, networking).
    2. Install and test supporting infrastructure SW.
3. Deploy and configure microservice, and test on IaaS.

Software Development Overview

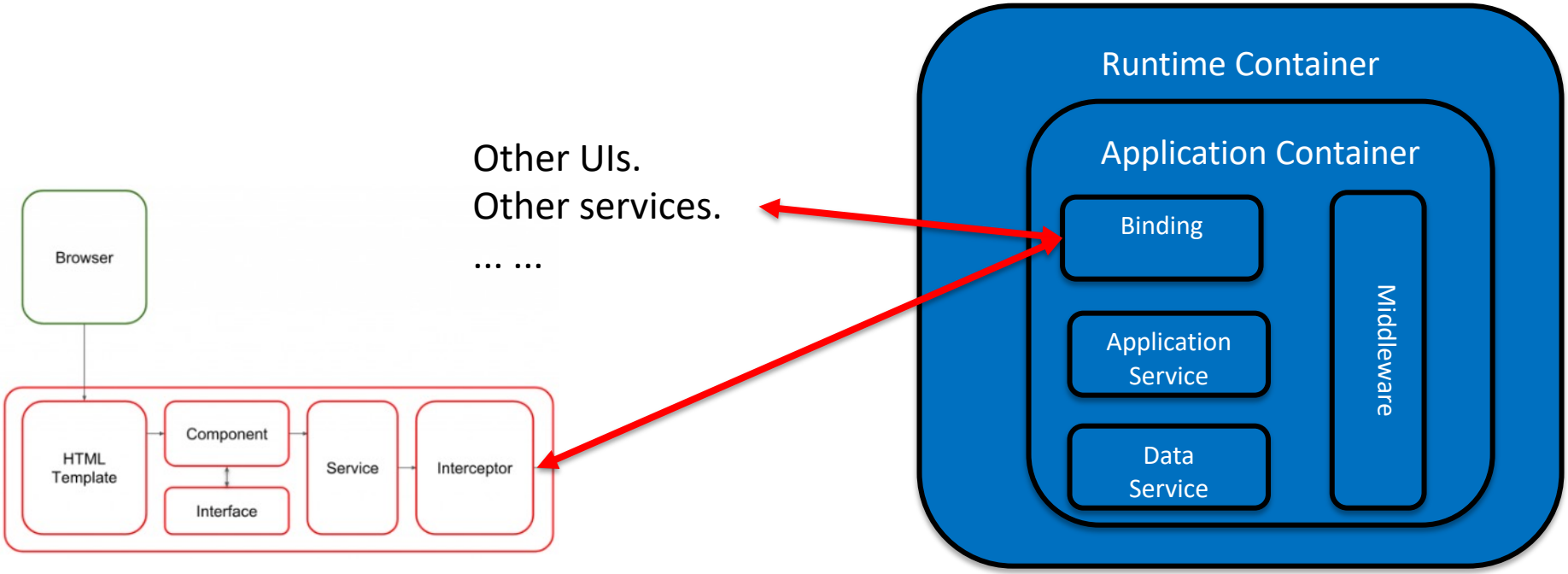


Environment

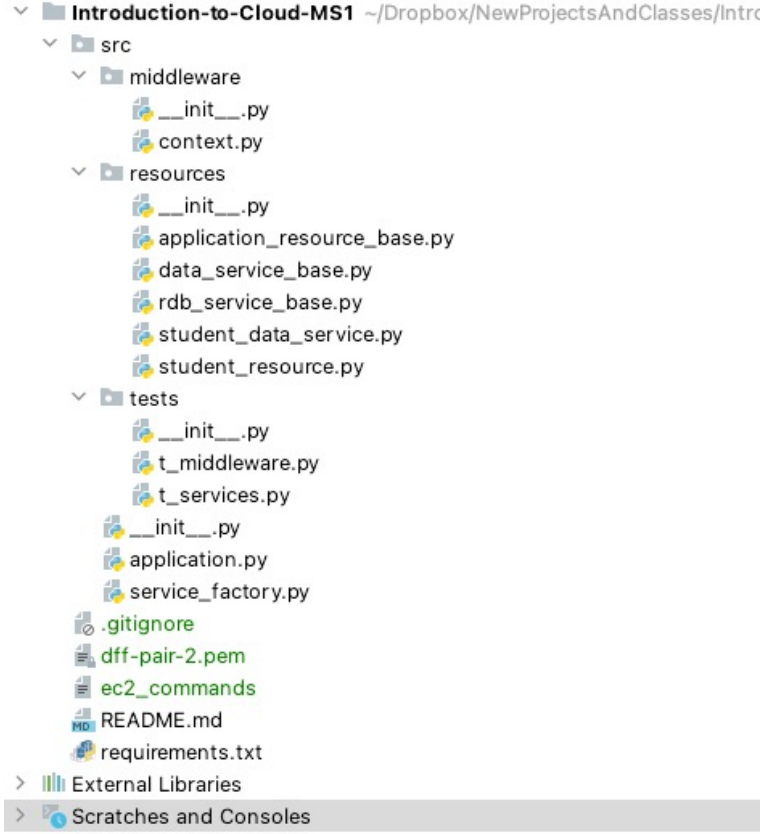
- We are working on getting students credits to use Amazon Web Services.
  - The first lecture and example starts with developing on your computer.
  - I will show the basics of the first deployment to AWS. Hopefully, we will receive credits within a few days.
- Computer environment:
  - Integrated Development Environment (IDE):
    - "Real programmers don't need no stinkin' IDE."
    - I use the JetBrains products.
      - I have a professional license because I am a professor.
      - Ansys has some licenses but I do not know the details.
      - You should be fine using the free community editions or VS Code.
  - I will use MySQL for my local RDBMs.
    - We will switch over to cloud databases (DB-as-a-Service).
    - You will be OK with SQLite for the first examples.

Overall Structure

- API First: All services have a well-defined API.
- ”The UI” is one of many applications that use the API.



Simple Project Walkthroug (https://github.com/donald-f-ferguson/Introduction-to-Cloud-MS1.git)



- This is not a great project structure. Ansys will evolve to a set of project templates.
- Some comments:
  - The “base” stuffs’ source should not be in the project source.
  - I wrote this quickly but partially followed some patterns.
    - This is not a course on “good programming.”
    - I am an **architect**. I do not sully my mind with such mundane tasks.
  - There are environment variables.
  - The project is a really, really bad place for the key file, even if it is in .gitignore.

IaaS and Deployment

- We will do many (most) of these tasks manually with commands or console.
  - Manual and UI helps visualize what is happening.
  - Automation and “infrastructure as code” will come later.
- Create instance (EC2, Ubuntu) in VPC, security group.
- Connect using ssh and key pair (get info from console)
  - Scroll through some_commands.md, .secret_stuff, app_env.sh
  - apt install
    - Mysql (and configure) (https://linuxbeast.com/tutorials/aws/how-to-install-mysql-on-amazon-ec2-ubuntu-18-04/)
    - Python
    - Git
  - Modify mysql bind rules in .cnf to enable remote access.
  - Deploy code from Git and set up environment.
- Test.

Summary

Summary

- That as was fun. **Let's not do that again.**
- All of that was pretty tedious. In future lectures, we will learn:
  - Infrastructure-as-Code (https://en.wikipedia.org/wiki/Infrastructure_as_code) to enable automation and reuse.
  - Pipelines to automate tasks when development events occur, e.g. commit.
  - Higher layers in the cloud stack that make this MUCH simpler.
    - Containers and Container-as-a-Service
    - Platform-as-a-Service
- We will also start using some more advanced concepts:
  - Security, secret management.
  - API management.
  - Service composition.
  - … …