

# *E6156 – Topics in SW Engineering (F23)*

## *Cloud Computing*

### *Lecture 3: REST, PaaS, FaaS, DBaaS, ... ..*



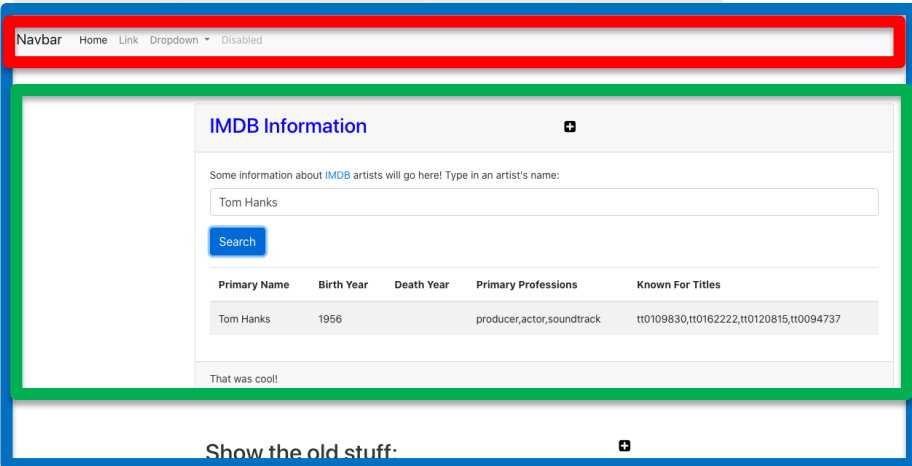
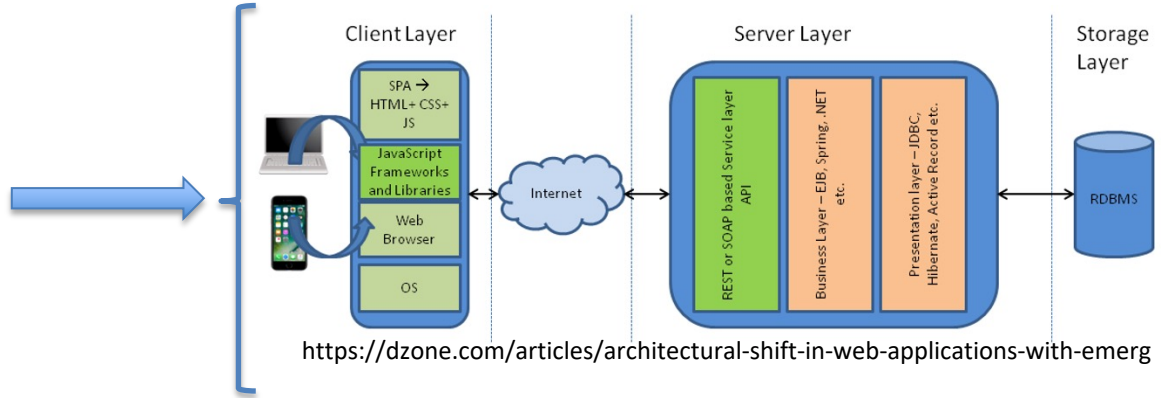
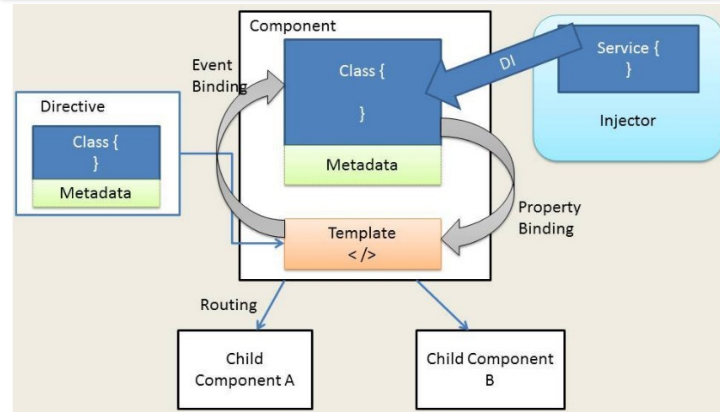
# *Contents*

# Contents

- Full Stack Web Application:
  - Overview
  - Walkthrough
  - Mapping Layers to the Cloud
- S3, BLOB Store and “Static” Web Content
  - Overview
  - Deployment and configuration
- Platform-as-a-Service:
  - Concepts
  - Elastic beanstalk realization and details
  - Demo/Example

# *Full Stack Web Application*

# Web Application Architecture

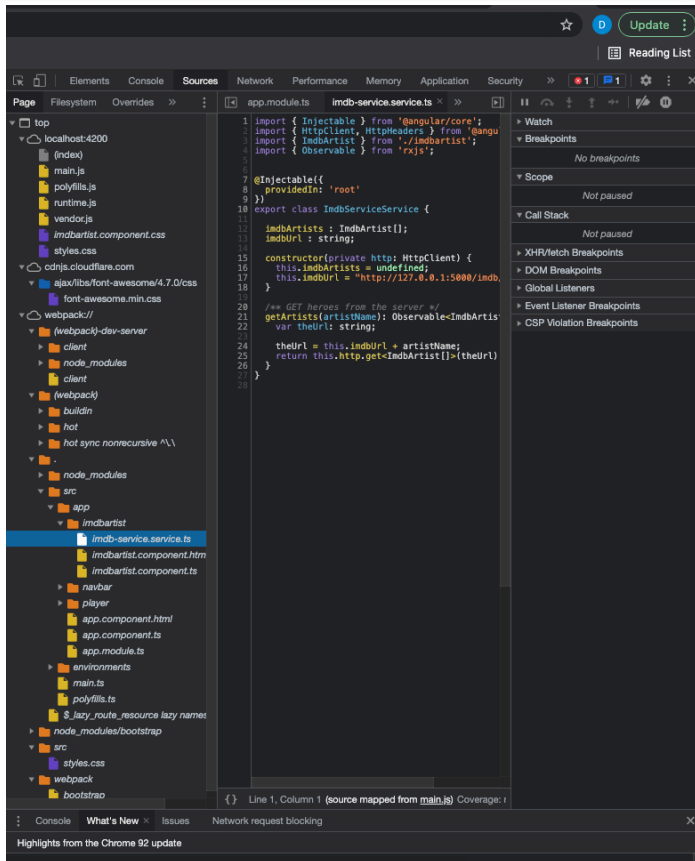


- app.component
- navbar.component
- imdbartist.component
  - Template (HTML)
  - CSS
  - Component implementation
  - Service

Show  
the  
Code



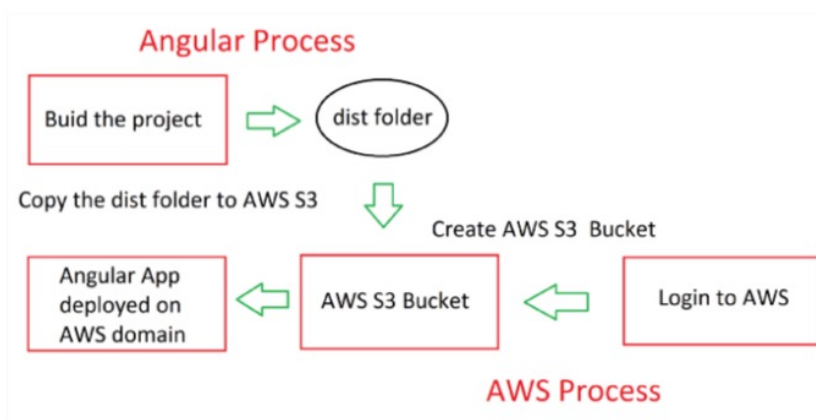
# What's in the Browser



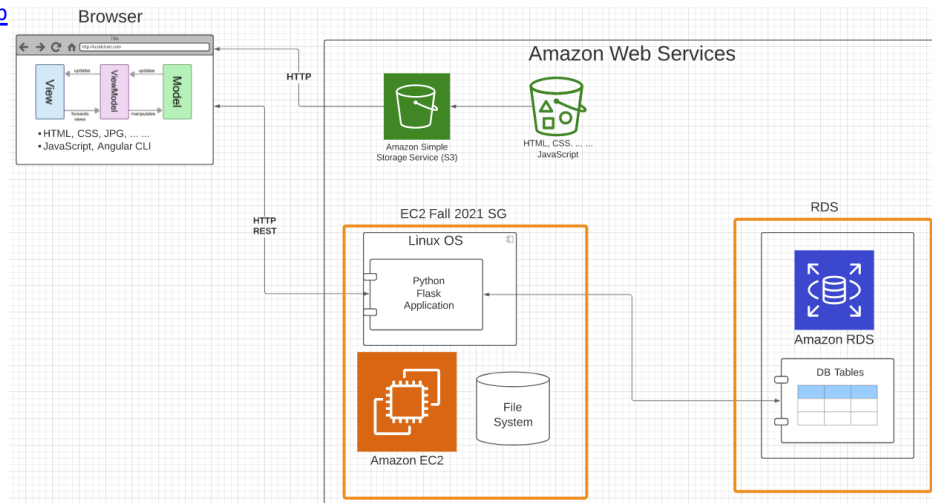
- The browser is an application that interprets the files:
  - HTML, CSS, etc. define the document and content.
  - JavaScript defines dynamic behavior.
- Browser loads index.html
  - index.html contains links to files.
  - The browser loads the linked files.
  - These also have links, which the browser loads.
  - etc.
- How did all of this get in the browser?
  - A *web server* delivers these files from “the file system.”
  - During development, Angular uses a simple, embedded web server.
  - “build” compiles the files into a [webpack](#) (see dist folder)
- Deploying the application to the cloud →
  - We have seen deploying the application logic to the cloud.
  - What about the content?
- **Notes:**
  - Show loading the demo-ui.
  - Show loading cnn.com.

# Deploy to S3

<https://baljindersingh013.medium.com/angular-app-deployment-with-aws-s3-42d9008734ab>



Block Diagram: Angular-AWS Deployment process



- Compile the browser application (ng build -- prod)
- You can test with ng serve -- prod
- Generates “bundles” that contain:
  - JavaScript application logic.
  - HTML, content, ... In the app logic.

- Create an S3 bucket.
- Upload files from /dist folder.
- Enable static web hosting.
- Set access permissions.
- Set index.html.
- We will learn how to automate later.

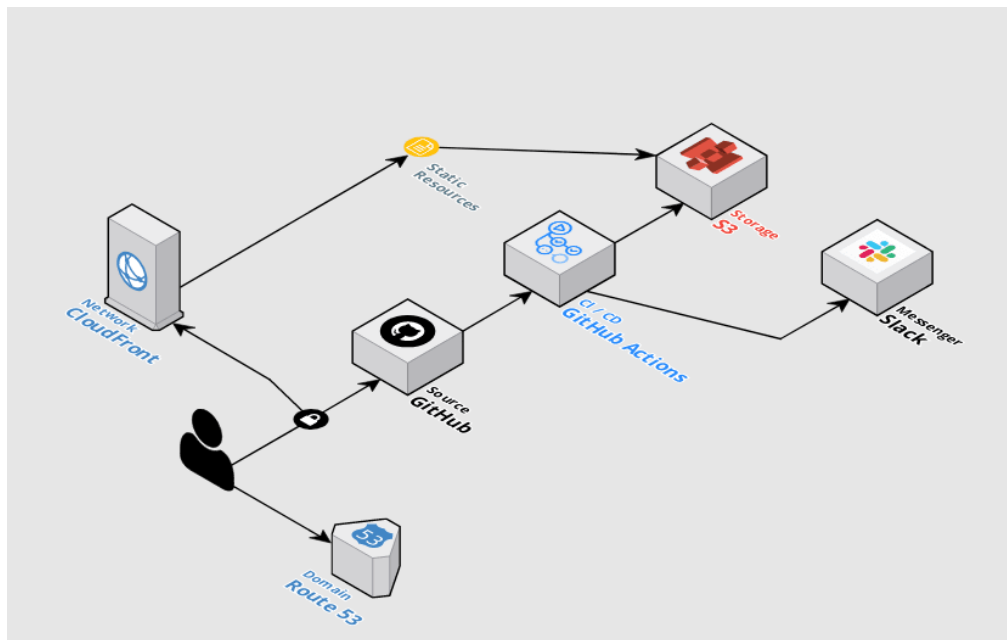
# Walkthrough

- Basics:
  - /Users/donaldferguson/Dropbox/000-NewProjects/W4111-FastAPI-IMDB-Solution
  - Local MySQL: GoT\_Basics, classicmodels.
  - /Users/donaldferguson/Dropbox/000-NewProjects/current-dashboard
- Cloud Deployment:
  - Angular project → S3.
  - IMDB-GoT microservice to one of our targets.
  - Relational data → RDS or something similar.
- Show various S3 APIs
- Show the setup of the bucket, uploading, ... ..
- Explain end-to-end, including CDN.



# Automating Deployment

- The previous slides showed manual steps to
  - Setup and configure.
  - “Deploy” the Angular application.
- There are several ways to automate the process.
  - Automation using CLI scripts, Python scripts, ... ..
  - Higher layer frameworks, e.g. CloudFormation, Terraform, ... ..
  - GitHub actions.
  - ... ..



<https://github.com/byangular/angular-github-actions-s3-deploy>

One of many tutorials and tools.

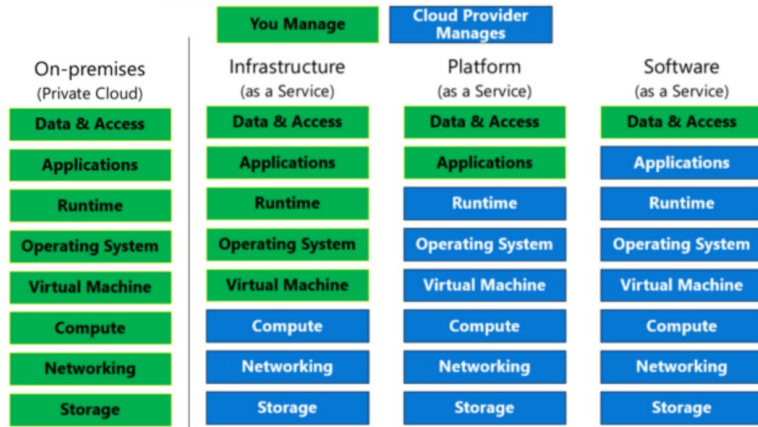
# *Introduction to PaaS*

## *Elastic Beanstalk*

# Platform-as-a-Service

- “ Platform as a service (PaaS) is an enabler for software development where a third-party service provider delivers a platform to customers so they can develop, run, and manage software applications without the need to build and maintain the underlying infrastructure themselves.”  
(<https://www.infoworld.com/article/3223434/what-is-paas-a-simpler-way-to-build-software-applications.html>)
- “Platform as a service (PaaS) is a complete development and deployment environment in the cloud, with resources that enable you to deliver everything from simple cloud-based apps to sophisticated, cloud-enabled enterprise applications.  
... ...  
PaaS allows you to avoid the expense and complexity of buying and managing software licenses, the underlying application infrastructure and middleware, container orchestrators such as Kubernetes, or the development tools and other resources. You manage the applications and services you develop, and the cloud service provider typically manages everything else.  
(<https://azure.microsoft.com/en-us/overview/what-is-paas/>)

# Platform-as-a-Service

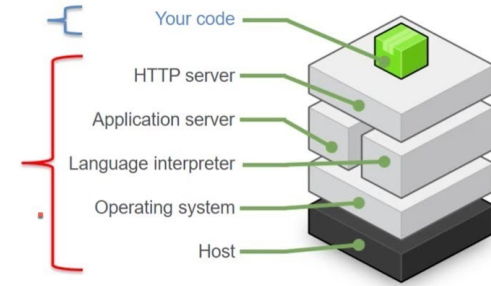


## Elastic Beanstalk

### On-instance configuration

Focus on building your application

Elastic Beanstalk configures each Amazon EC2 instance in your environment with the components necessary to run applications for the selected platform. No more worrying about logging into instances to install and configure your application stack.



Provided by you

Provided and managed by Elastic Beanstalk

- Simplistically, PaaS adds two additional layers on top of IaaS:
  - Runtime means the language environment, libraries, etc. your application needs. For example, in our case this will mean a predefined and configured Flask environment.
  - “Middleware is a type of computer software that provides services to software applications beyond those available from the operating system.” (<https://en.wikipedia.org/wiki/Middleware>)
- Basically, you do not have to build an application execution environment by installing libraries, services, ... Your application “goes into a premade environment.”
- Less flexible and customizable than IaaS but can be more productive to use for application scenarios.

# Clarifying PaaS

- This is a little hard to understand until you have done it a few times.
  - IaaS basically stops at the operating system layer. You are responsible for everything else your application code needs:
    - Database
    - Logging Service
    - ... ..
  - PaaS provides a prebuilt environment with “your code goes here.”
- The easiest way to see the difference is
  - You are using EC2.
  - We will use a PaaS (Google AppEngine because Beanstalk is terrible)  
And you will get a feel for the differences.
- *So, let's do this.*

# Examples

- [/Users/donaldferguson/Dropbox/00-Fall-2023/Google/google\\_1](/Users/donaldferguson/Dropbox/00-Fall-2023/Google/google_1)
- <https://cloud.google.com/appengine/docs/legacy/standard/python/how-to>
- Show google console.
  - Note: I have coupons for Google and will distribute.
  - Show application.
  - Use "free" for now.
- Quick walkthrough of AWS Elastic BeanStalk



# *Function-as-a-Service*

# Serverless and Function-as-a-Service

- **Serverless computing** is a [cloud computing execution model](#) in which the cloud provider runs the [server](#), and dynamically manages the allocation of machine resources. Pricing is based on the actual amount of resources consumed by an application, rather than on pre-purchased units of capacity.<sup>[1]</sup> It can be a form of [utility computing](#).



Serverless computing can simplify the process of [deploying code](#) into production. Scaling, capacity planning and maintenance operations may be hidden from the developer or operator. Serverless code can be used in conjunction with code deployed in traditional styles, such as [microservices](#). Alternatively, applications can be written to be purely serverless and use no provisioned servers at all.<sup>[2]</sup> This should not be confused with computing or networking models that do not require an actual server to function, such as [peer-to-peer](#) (P2P)."

- **Function as a service (FaaS)** is a category of [cloud computing services](#) that provides a [platform](#) allowing customers to develop, run, and manage application functionalities without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app.<sup>[1]</sup> Building an application following this model is one way of achieving a "[serverless](#)" architecture, and is typically used when building [microservices](#) applications."
- That was baffling:
  - IaaS, CaaS – You control the SW stack and the cloud provides (virtual) HW.
  - PaaS – The cloud hides the lower layer SW and provides an application container (e.g. Flask) with "Your code goes here." You are aware of container.
  - FaaS – The cloud provides the container, and you implement functions (corresponding to routes in REST).

# Serverless and Function-as-a-Service

Private Cloud	IaaS Infrastructure as a Service	PaaS Platform as a Service	FaaS Function as a Service	SaaS Software as a Service
Function	Function	Function	Function	Function
Application	Application	Application	Application	Application
Runtime	Runtime	Runtime	Runtime	Runtime
Operating System	Operating System	Operating System	Operating System	Operating System
Virtualization	Virtualization	Virtualization	Virtualization	Virtualization
Server	Server	Server	Server	Server
Storage	Storage	Storage	Storage	Storage
Networking	Networking	Networking	Networking	Networking

<https://medium.com/@tanmayct/serverless-architecture-function-as-a-service-19e127b8c990>

Managed by the customer   
Managed by the provider 

# What is Serverless Good/Not Good For ... ..

Serverless is **good** for  
*short-running*  
*stateless*  
*event-driven*



Microservices



Mobile Backends



Bots, ML Inferencing



IoT



Modest Stream Processing



Service integration

Serverless is **not good** for  
*long-running*  
*stateful*  
*number crunching*



Databases



Deep Learning Training



Heavy-Duty Stream Analytics



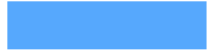
Numerical Simulation



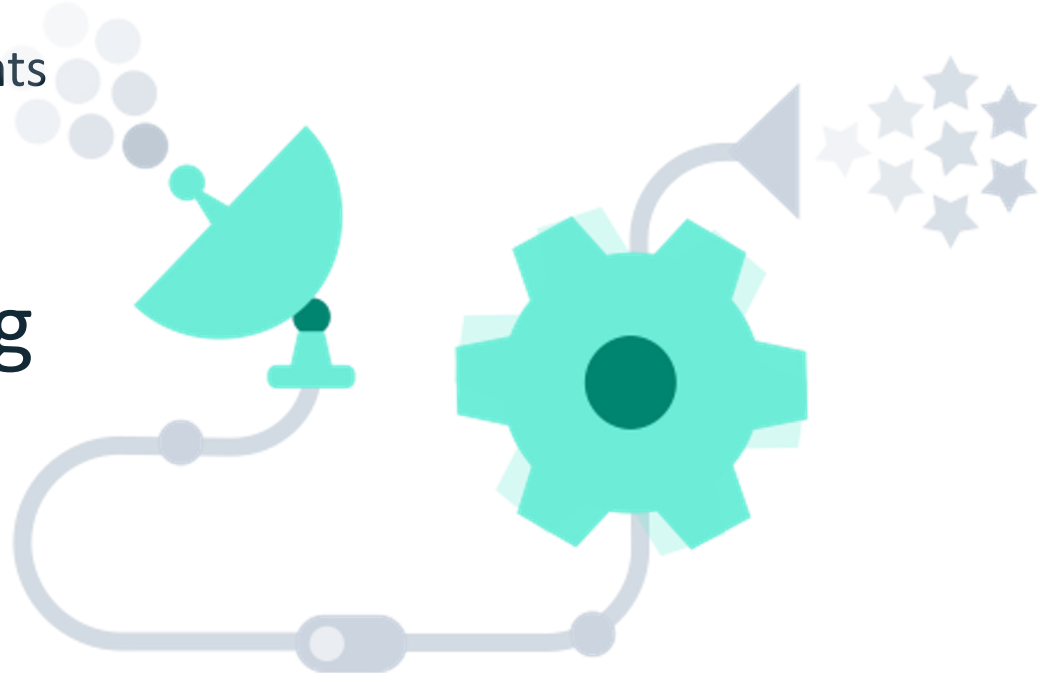
Video Streaming

# What triggers code execution?

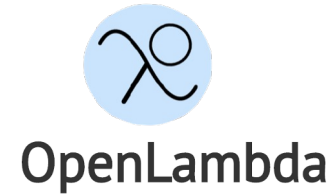
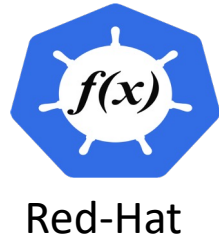
Runs code **in response** to events



## Event-programming model



# (Some) Current Platforms for Serverless





# Let's Build a Lambda Function

- We are going to do it manually to understand the process/steps.
  - Do through the console starting with a template.
  - Publish through API GW
  - Explain the concepts of events and how we will use.
- There are several more complete approaches:
  - Pipelines
  - Serverless Framework  
(<https://www.serverless.com/framework/docs/providers/aws/guide/intro>)
  - AWS Toolkit for PyCharm  
(<https://aws.amazon.com/pycharm/>)
- We will explore some of these ... ..

# *Data-as-a-Service*

# Cloud Concepts – One Perspective

## Categorizing and Comparing the Cloud Landscape

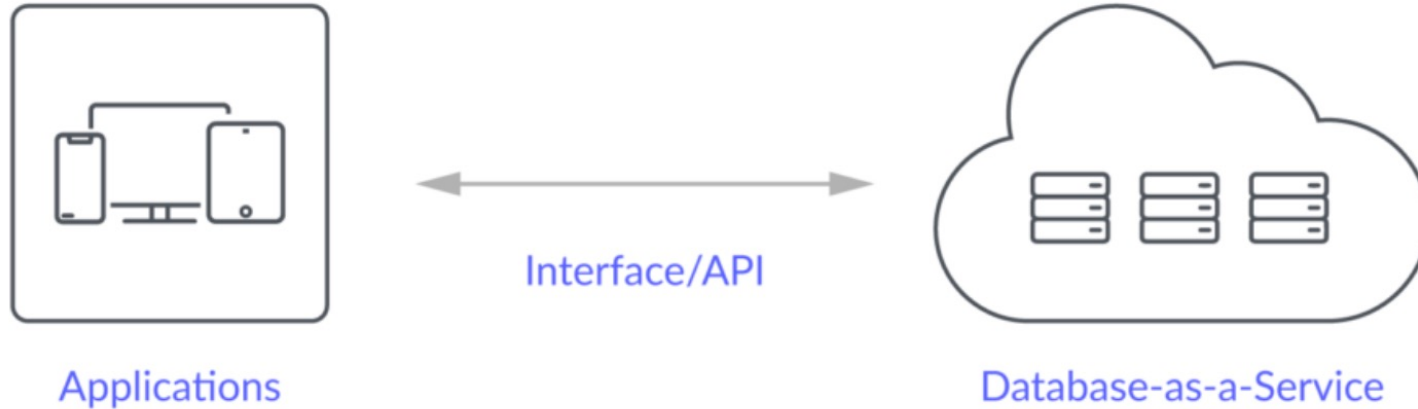
<http://www.theenterpriseearchitect.eu/blog/2013/10/12/the-cloud-landscape-described-categorized-and-compared/>

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	

# Database-as-a-Service

“A cloud database is a database that typically runs on a cloud computing platform and access to the database is provided as-a-service. There are two common deployment models: users can run databases on the cloud independently, using a virtual machine image, or they can purchase access to a database service, maintained by a cloud database provider. Of the databases available on the cloud, some are SQL-based and some use a NoSQL data model.

Database services take care of scalability and high availability of the database. Database services make the underlying software-stack transparent to the user.” (Wikipedia)



Let's take a look at 3: Relational Data Service, ~~Dynamo DB~~, ~~MongoDB (Compass)~~

# Walkthrough

- RDS
- Setup and configuration
- DataGrip
- Use from a program
- Google equivalent.

*Diagram of First Major Deliverable  
and  
What I am Doing this Weekend*