Complex Software System Development

Modern Apps are Complex

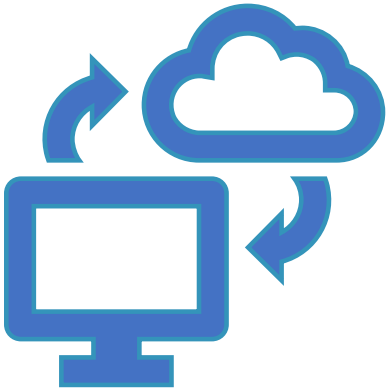
Highly complex systems characterised by:

Variety of Interfaces
- Mobile devices
- Tablets
- Laptops
- Desktops

Decentralised control
- Servers
- Sensors/Actuators
- Databases
- External APIs

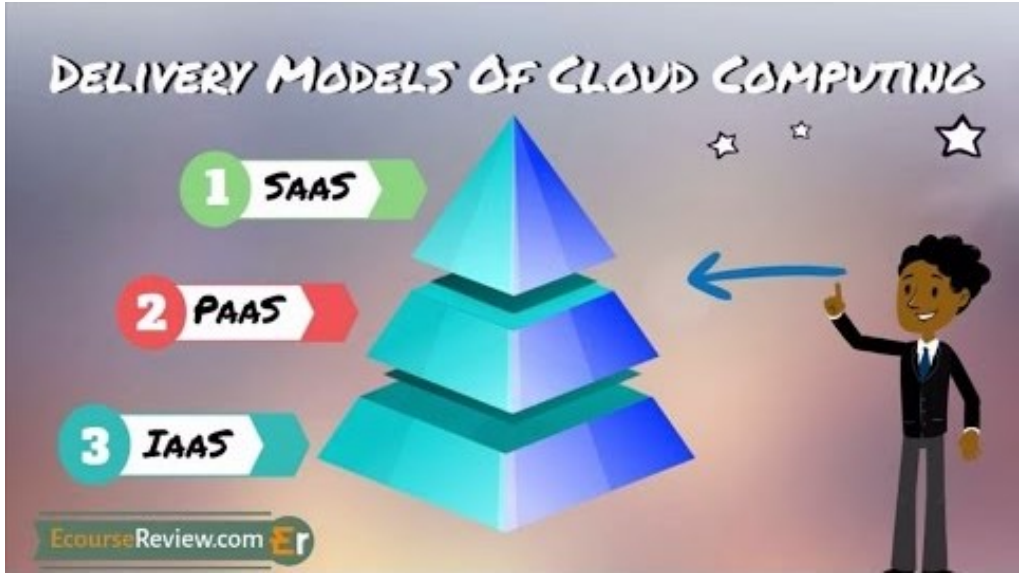
We will use Mobile/Web interchangeably

Speed of changes



- The technologies landscape is evolving rapidly and is extremely diverse
- Cloud computing is causing profound changes in both business and technology landscape
- New architectural models come in response to new technological developments
- Changes in architectures are very cyclical in nature
  - thin client -> fat client -> thin client
  - Centralised -> decentralised
- Open standards are becoming prevalent (e.g. SAP).

Cloud computing service models



https://www.youtube.com/watch?v=36zducUX16w 6 minutes

New trends

Cloud now supports

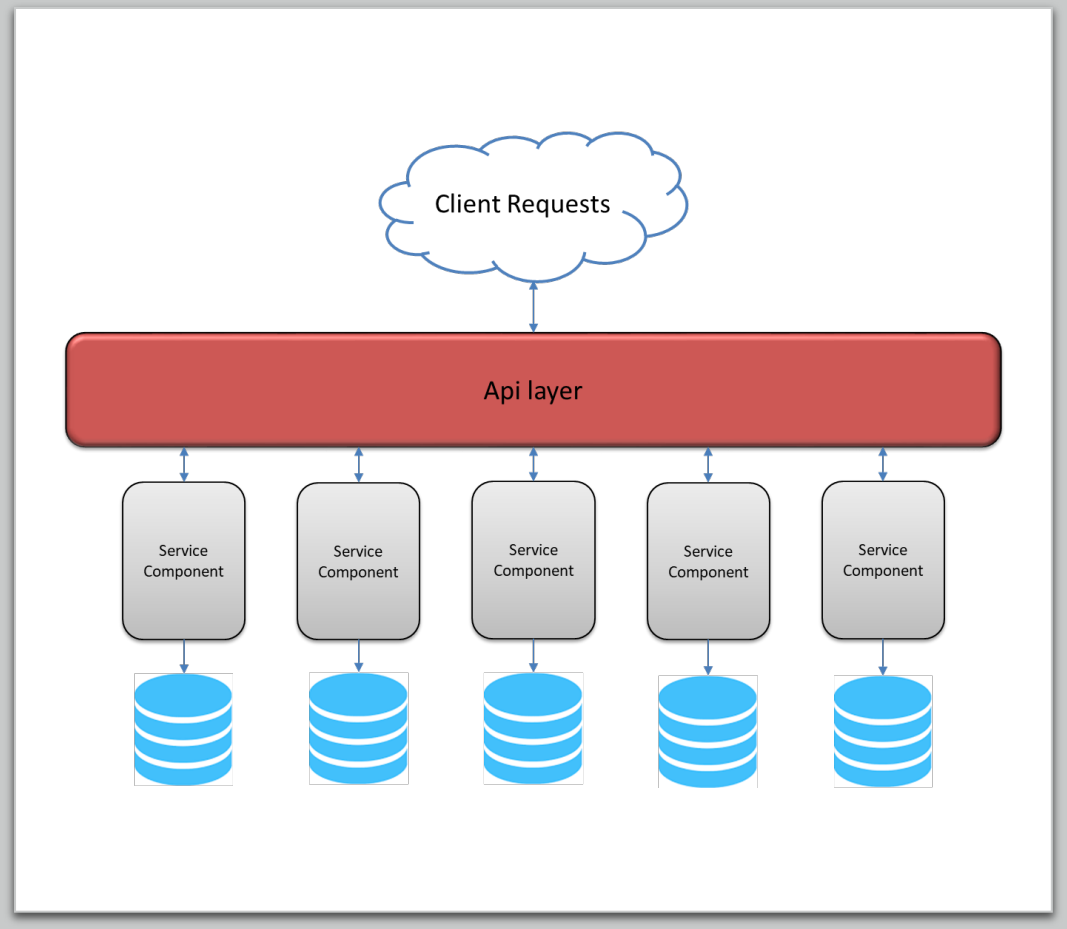
- Automated testing.
- Continuous Integration/Deployment.
- Serverless computing

Architectural trends

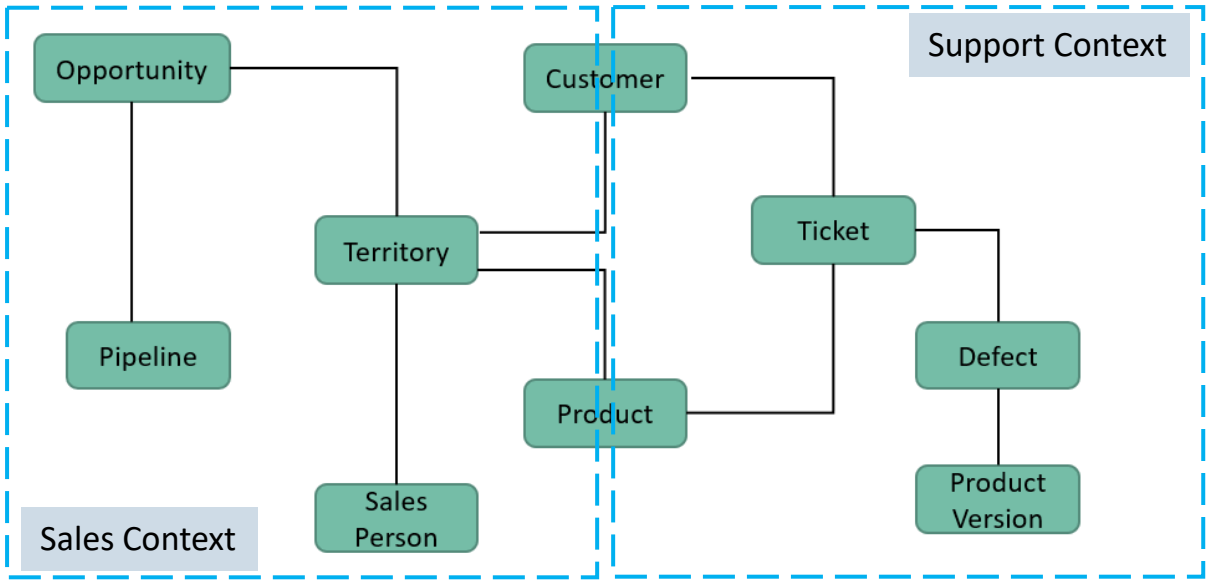
- No longer a monoculture approach
- Engineering for interoperability is key
- No one technology is the silver bullet
- Open source is winning
- Event sourcing is rising in popularity (Kafka)
- Move towards microservice architectures

Microservices Main Characteristics

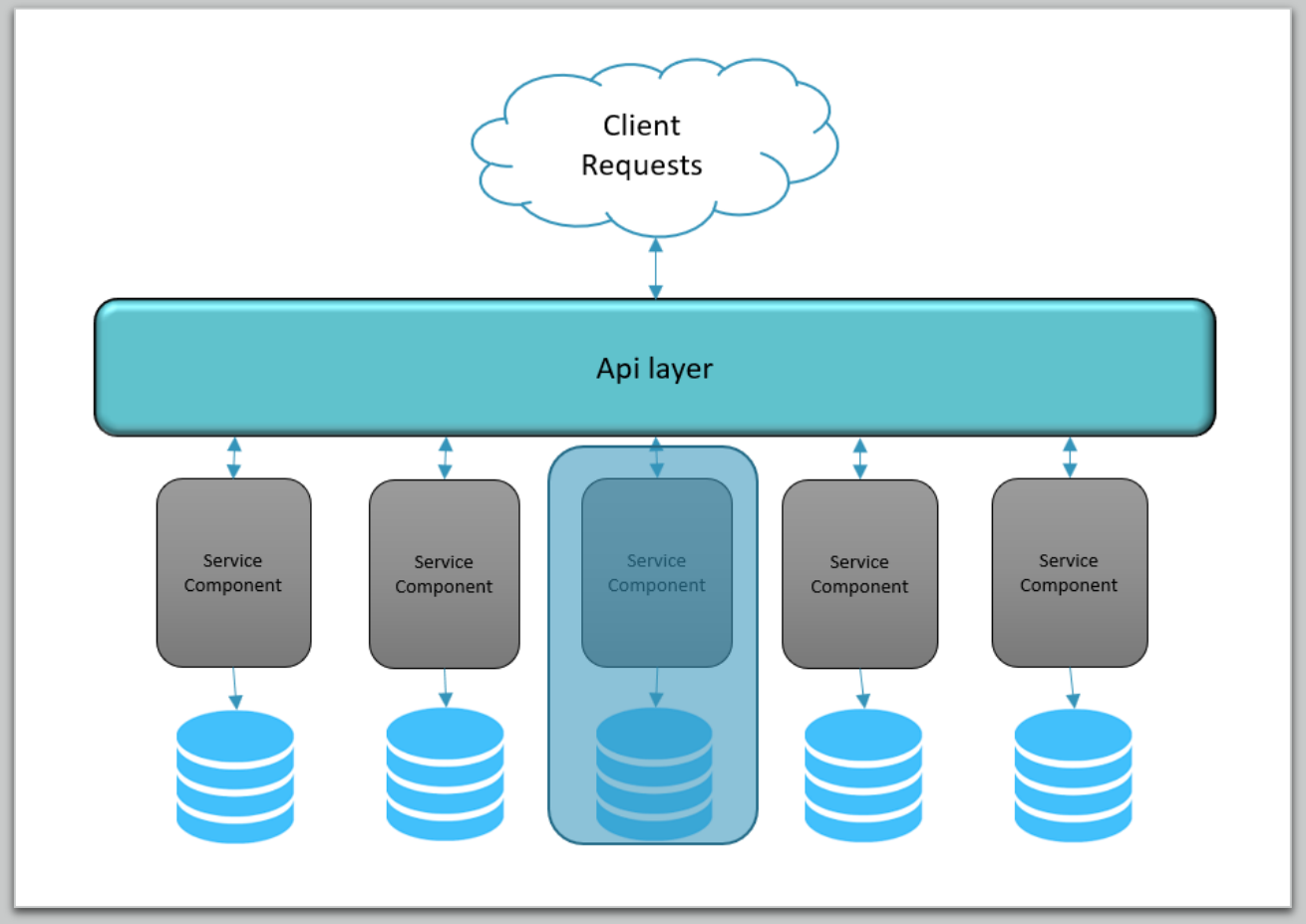
1. Distributed Architecture
2. Separately Deployed
3. Service Component
4. Bounded Context



Bounded Context

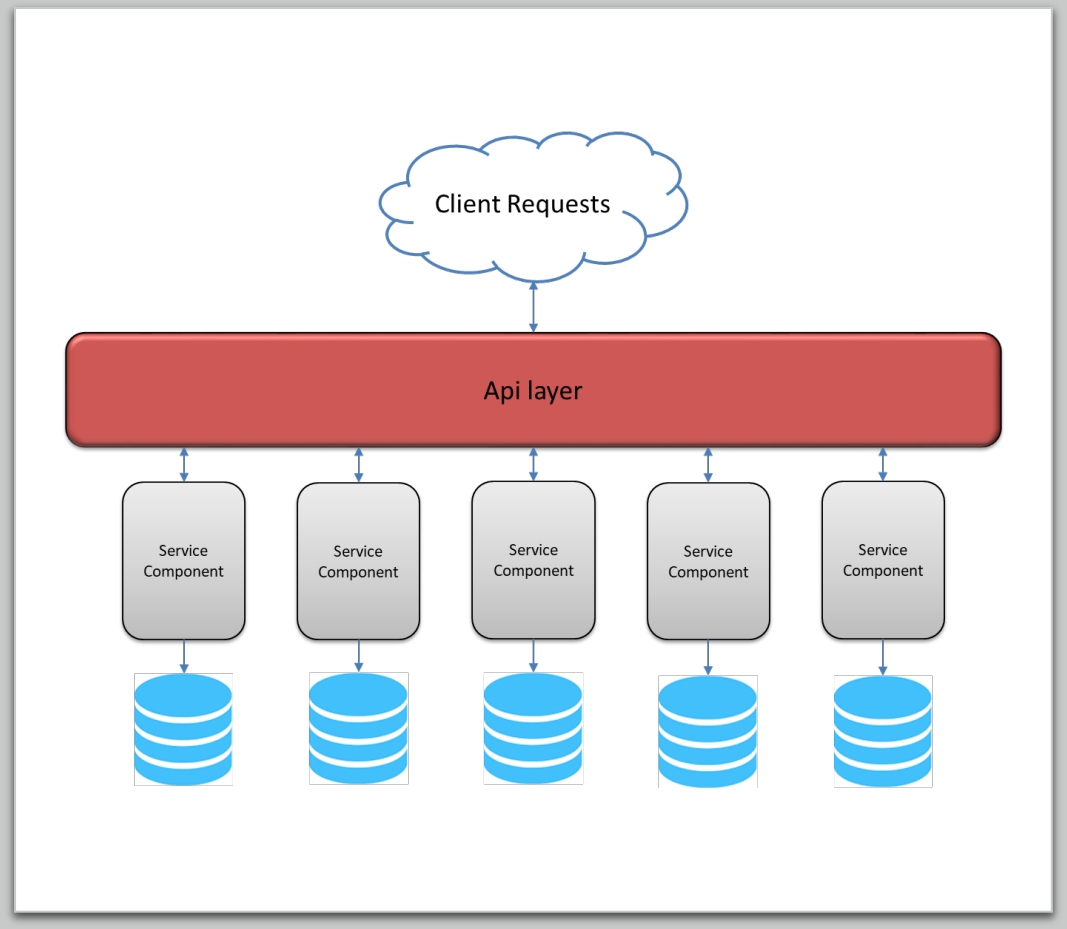


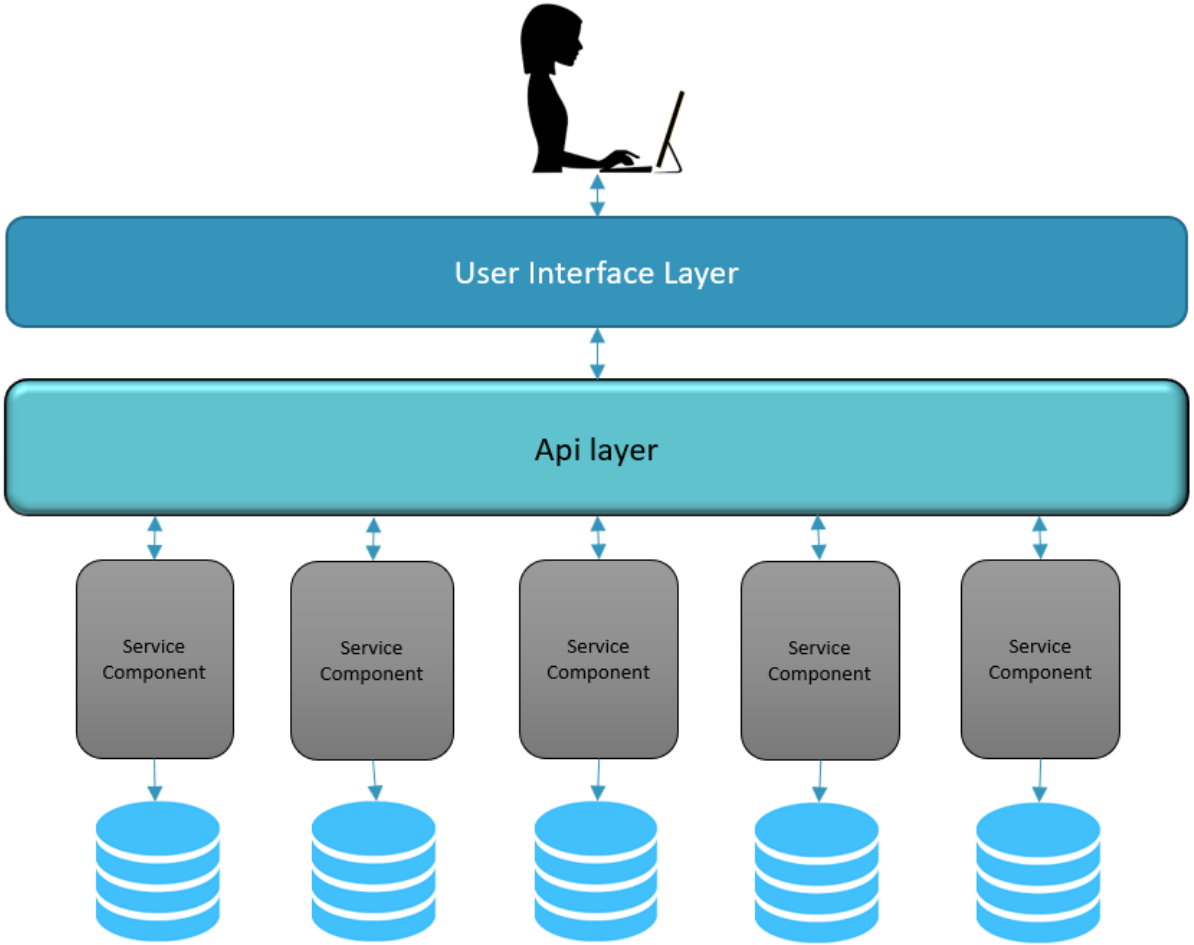
Bounded Context



Microservices Main Characteristics

1. Distributed Architecture
2. Separately Deployed
3. Service Component
4. Bounded Context
5. Share Nothing
6. Api Layer
7. Favor rewriting




User Interface Layer
Api layer
Service Component
Service Component
Service Component
Service Component
Service Component

Choosing a stack for prototyping (1)

Choosing between

- Design a mobile app or design a web site optimised for a smartphone
- Second one is easier for beginners

Front-end stack provides

- Package Management
- Templating
- Testing framework
- Common widgets
- Databinding
- Server communication

Choosing a Front-end stack

- HTML/CSS/Bootstrap recommended for beginners
- For advanced users, use Angular, React or Vue.js

Choosing a stack for prototyping (2)

Back	Back-end stack provides access to back-end services: APIs, data generation, external systems etc.
Choice	Choosing a back-end stack involves different considerations
Beginners	Can use Python back-end using the Flask web framework (which comes with the Jinja templating engine by default)
Database	If you need database, use SQLite, otherwise just simple JSON files
Advanced	Advanced users can adopt JavaScript for the back-end which requires the package management system for Node ‘npm’
See	See provided FAQ

Focus of this workshop

- Previous workshop was focusing on:
  - Designing user interfaces: User Experience
  - Front End Stack: Javascript and other tools
- This workshop is more on:
  - API design
  - Cloud deployment
  - Testing
  - Data integration
  - Using tools to manage development