Open source and architecture

a bit of background understanding

18F AcqStack

1/ Open source

Open source is software code that is made publicly available. Anyone can see it, use it, and modify it.

Crowd power! Since everyone can see it, you can get lots of different eyes on the problem. Different backgrounds and perspectives can yield better solutions.

Open source enables sharing and reuse.

Open source can help prevent vendor lock-in.

In 2008, the Standish Group reported that open source models were saving consumers about $60 billion every year.

Isn’t it risky to put my code out in the public for everyone to see?

Probably not!

Code is usually not sensitive, only in a handful of rare cases

- things like secret algorithms
- separate the sensitive code from the rest, publish the non-sensitive parts

Configuration and data are sensitive

- database connection information
- locations of other servers
- file paths, etc.

Separate your configuration from your code, then publish that code!

- Adopt an approach such as 12-Factor App

2/ Architecture

18F

Agile development favors emergent architecture over big design up-front.

Let the best architecture emerge as development progresses. Start with a monolithic application and see what happens.

As complexities arise, abstract them. These abstractions serve as natural places to create modules/functions/services

Software is completely malleable.
Use that to your advantage.

Don’t spend too much time trying to define your architecture up front.

Automated tests can give you confidence that your changing architecture isn’t breaking things that already work.

Service-oriented architecture

Distinct pieces of functionality are broken out into services and put on the network.

Service-oriented architecture

Any application can use the services. The services typically hide some complex resource making it easier for the applications to use.

Service-oriented architecture

This is just a new variant of classical modular development. The main difference is the “modules” are accessed over the network.

Microservices

Service-oriented architecture for applications.

Microservices

Application functions are split into services, and then just that application uses them.

Microservices

Microservices allow functions to make architecture choices that best fit the needs of the function without being restricted by application-level decisions.

Microservices

Applications talk directly to their microservices. They do not go through an intermediary, like an enterprise service bus.

Microservices

Microservices can be deployed independently, so one can be changed or fixed without having to redeploy a whole application.

SOA and Microservices

Basically the same thing, just at different scales.