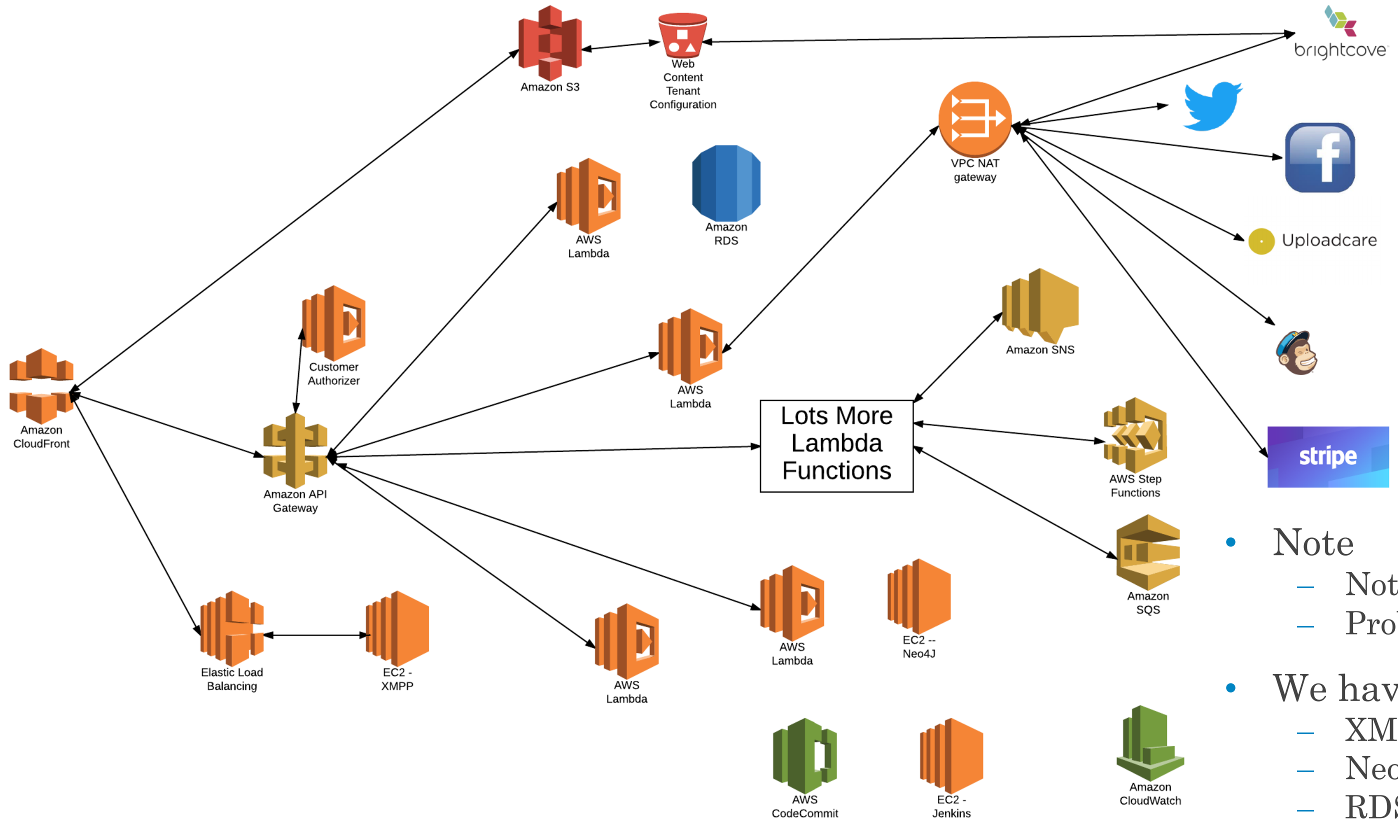
Experience and Lessons from Building and Teaching a Serverless Solution

Second International Workshop on Serverless Computing (WoSC) 2017, ACM/IFIP/USENIX Middleware 2017

Donald F. Ferguson
Adjunct Professor, Dept. of Computer Science, Columbia University
Co-founder and CTO, Sparq TV
dff@cs.columbia.edu, donald.ferguson@seeka.tv

Seeka TV Architecture



- Note
  - Not all connections shown
  - Probably forgot stuff
- We have a couple of servers
  - XMPP
  - Neo4J (Graphene)
  - RDS

Seeka TV Architecture

Lambda implementing microservices for

- Registration, authentication
- User and profile management
- Catalog and digital asset management
- Watch parties
- Commenting, tagging, …
- Social media integration
- Placement (business videos)
- Tipping, crowd funding
- Multi-tenant management
- Other stuff I forgot

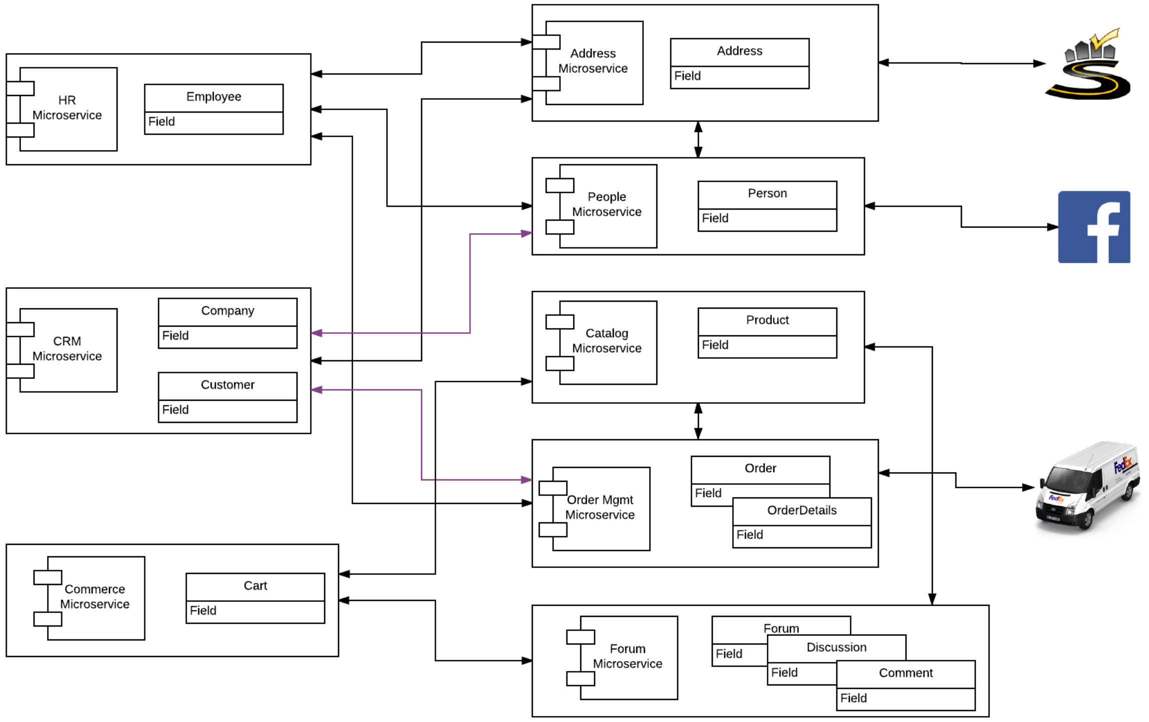
ections shown
got stuff

ple of servers

aphene)

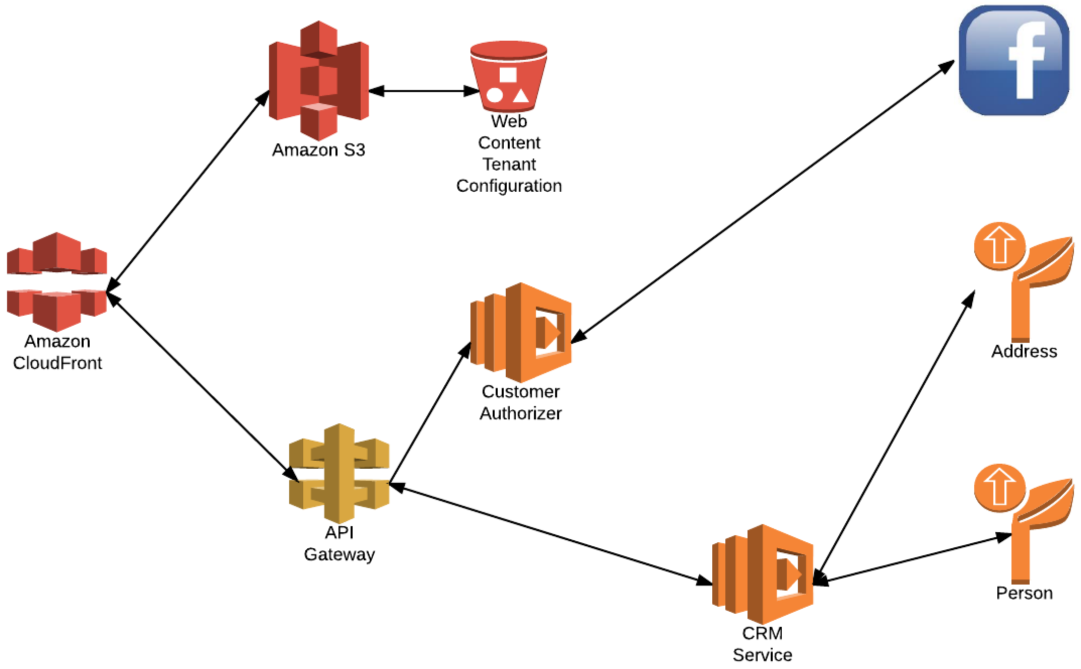
E6998 – Microservice and Cloud Applications

Microservices Model

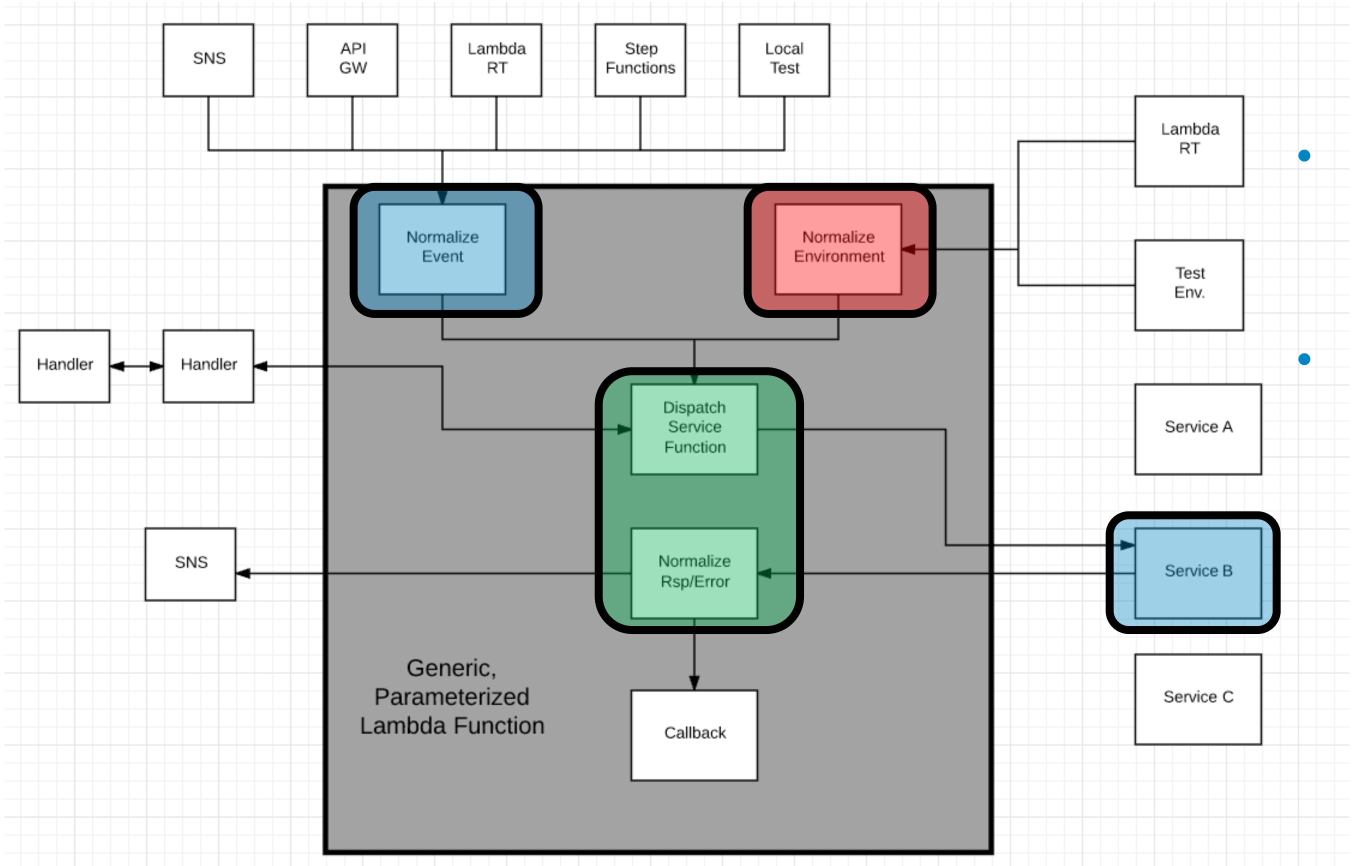


- We only accomplished a fraction
  - Address
  - Person
  - OAuth2
  - Some composite microservice functions

E6998 – Microservice and Cloud Applications
Component Model



Design Pattern – Generic Lambda Function



- Options
  - In zip file
  - Separate Lambda
- Config info
  - Added to Lambda
  - Env Variable
  - From S3

Lessons Learned and Research Directions

- Lessons learned
  - Serverless is much more than Lambda functions/function.
    - Think of the environment the way I drew it. A bunch of icons.
    - If you can configure and program with a web browser, and you do not manage hardware, SW, upgrade, etc. → It is serverless.
    - The environment is like a massive programmable wiki of /URLs
  - Productivity
    - There is significant productivity, especially initially, by eliminating all HW and SW server configuration and management.
    - The stateless model becomes incredibly productive but requires evolving from a more traditional microservice/service/application model to a event-function-event model.
    - There are a lot of subtle configuration settings and interactions between elements, and this is within a single environment. Azure-IBM-Google-AWS-… terrifies me.
- Research opportunities
  - Service composition, even with SWF and Step Functions, is too tedious.
  - Application dependency mapping and end-to-end unit of work monitoring.
  - Declarative quality of service and middleware injection