Workloads in the cloud

- Types of workloads
- Packaging options
- Execution options

Northwestern University

New architectural concepts…

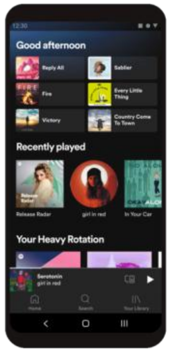
- I want to motivate two design concepts…

    1. Lambda functions

    2. Serverless computing

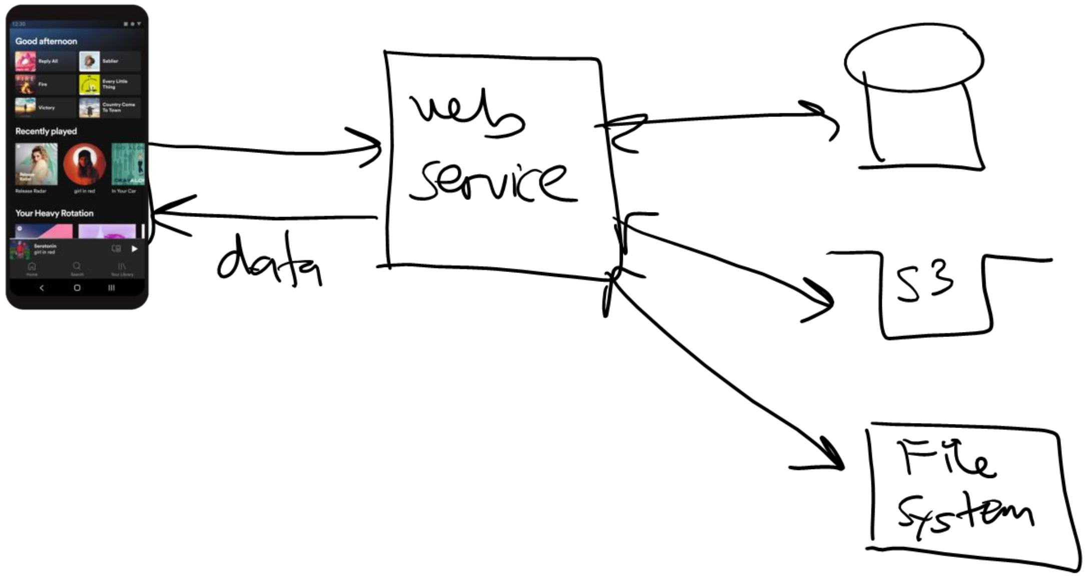
Multi-tier, data-driven apps

- **Our examples (so far) have all been data-driven**



I/O bound

- **We call this kind of workload "I/O bound"**
  - *Server is spending most of its time waiting for requests / data, i.e. input/output*
  - *This is typically handled via async programming*



Lambda

- Lambda functions
- Intro to serverless computing

Execution continuum

↔

EC2, EKS, ECS, Fargate	Elastic Beanstalk	API Gateway + Lambda	Lambda
• *Run any software you want for as long as you want* • *Complete control over HW and SW* • *Hardest to config*	• *Upload .zip file* • *Limited software choices* • *Some control over HW and SW*	• *Function based* • *Near-zero config* • *Web service + functions (15-min limit)*	• *Function based* • *Near-zero config* • *Short execution (< 15 mins)*

AWS lambda

- **By far the simplest, least expensive way to compute**

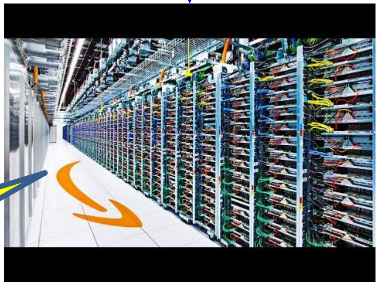
EC2, EKS, ECS, Fargate

Elastic Beanstalk

API Gateway + Lambda

Lambda

function



Scalable! Each call executes independently on different HW => horizontal scaling

AWS lambda / Azure functions / Google functions

- **Standalone functions executed on demand**
  - *Can be written in JavaScript, Python, Java, C++, etc.*
  - *Execution time is limited (AWS => 15 minutes)*

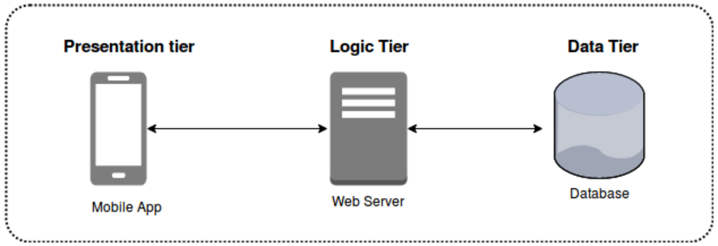
- **Callable in a variety of ways:**
  - *Like a traditional function( ) using AWS library*
  - *Based on **events** that occur (e.g. uploading an item into S3)*
  - *Via **function URL** through AWS-managed web server*
  - *Via **API Gateway** offering a more customizable AWS-managed web server (e.g. test vs. production, more authentication options, …)*

Serverless

- Serverless computing
- API Gateway + lambda

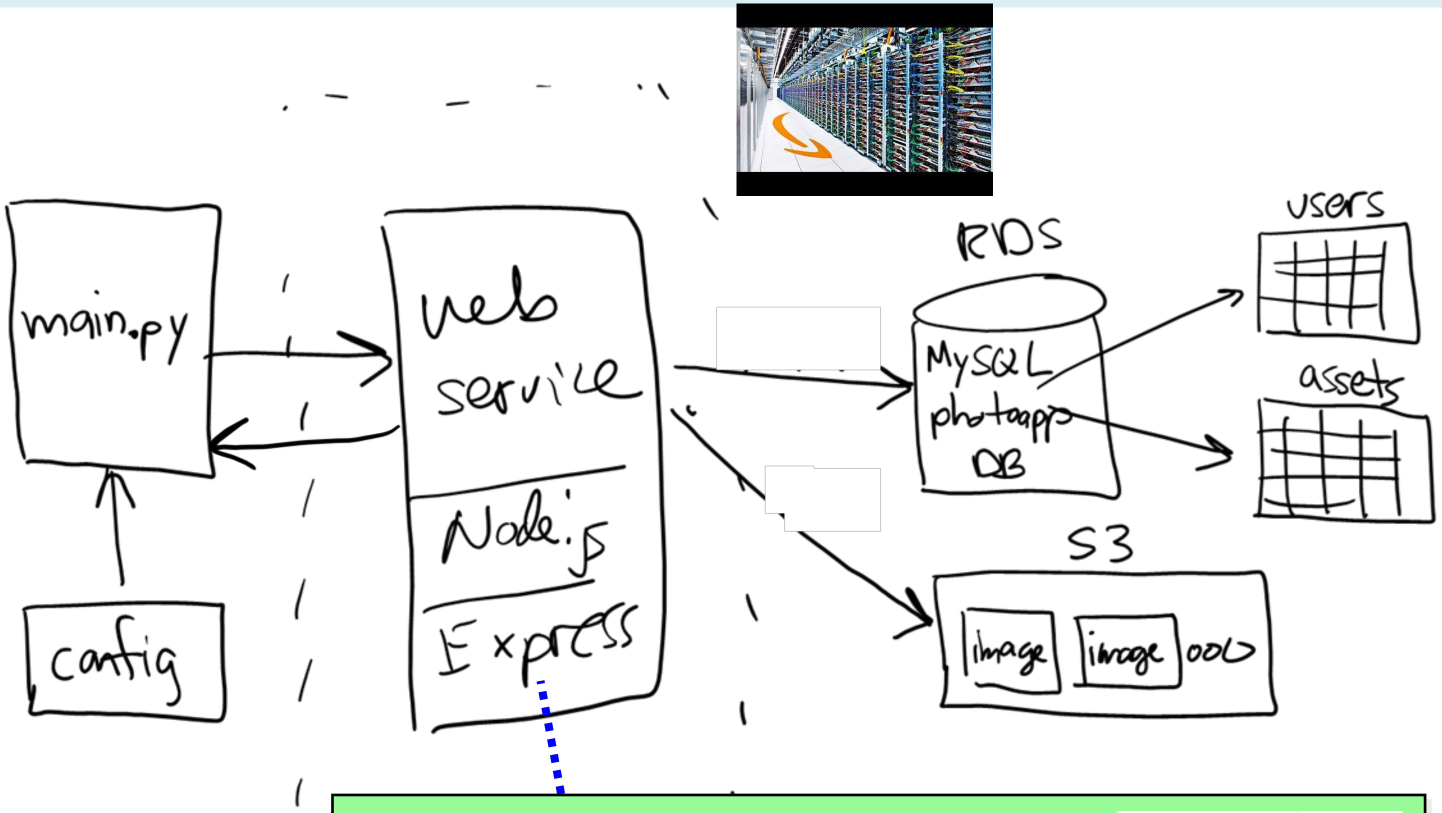
Northwestern University

Monolithic multi-tier



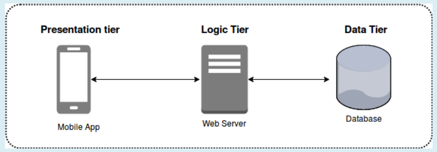
- **Traditional software design for the cloud**
- **Monolithic approach --- one large code base on server**
  - *Safe, conservative engineering*
  - *No one gets fired for building systems this way :-)*

Project 02 --- monolithic web service



```
//
// PhotoApp web service
//
app.get('/stats', (req, res) => {…});
app.put('/user', (req, res) => {…});
app.get('/users', (req, res) => {…});
app.get('/assets', (req, res) => {…});
app.get('/bucket', (req, res) => {…});
app.get('/image/:assetid', (req, res) => {…});
app.post('/image/:userid', (req, res) => {…});
```

Alternative designs?



1. **Microservices**
   - *Break monolithic system apart --- easier to develop, update, release, but more moving parts to manage*
   - *Example: **Netflix** was one of the first to do this*

2. **Event-driven**
   - *Design based on events that occur / application states*
   - *Example: food delivery => menu, order, purchase, prepare, deliver*

3. **Serverless computing…**