CS 310 : Scalable Software Architectures

Class session on Tuesday, October 22nd

October 2024

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

www.a-printable-calendar.com

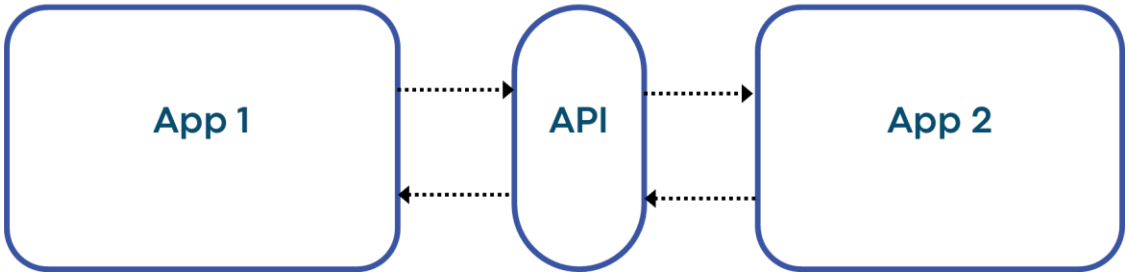
Notes:

- ***Focus this week***:
  - *Scalability, performance, serverless*
- ***Project 02*** *due Friday*
  - *Part 01: web service for photoapp, rewrite client*
  - *Part 02: deploy to AWS EC2*
  - *Due Friday with late submissions Sunday*
- ***Exam*** *next Tuesday*
  - *Practice problems and review in class on Thursday*

Northwestern University

RESTful API

- **Most web service APIs are said to be "RESTful"**
- **This is an architectural style with design constraints:**

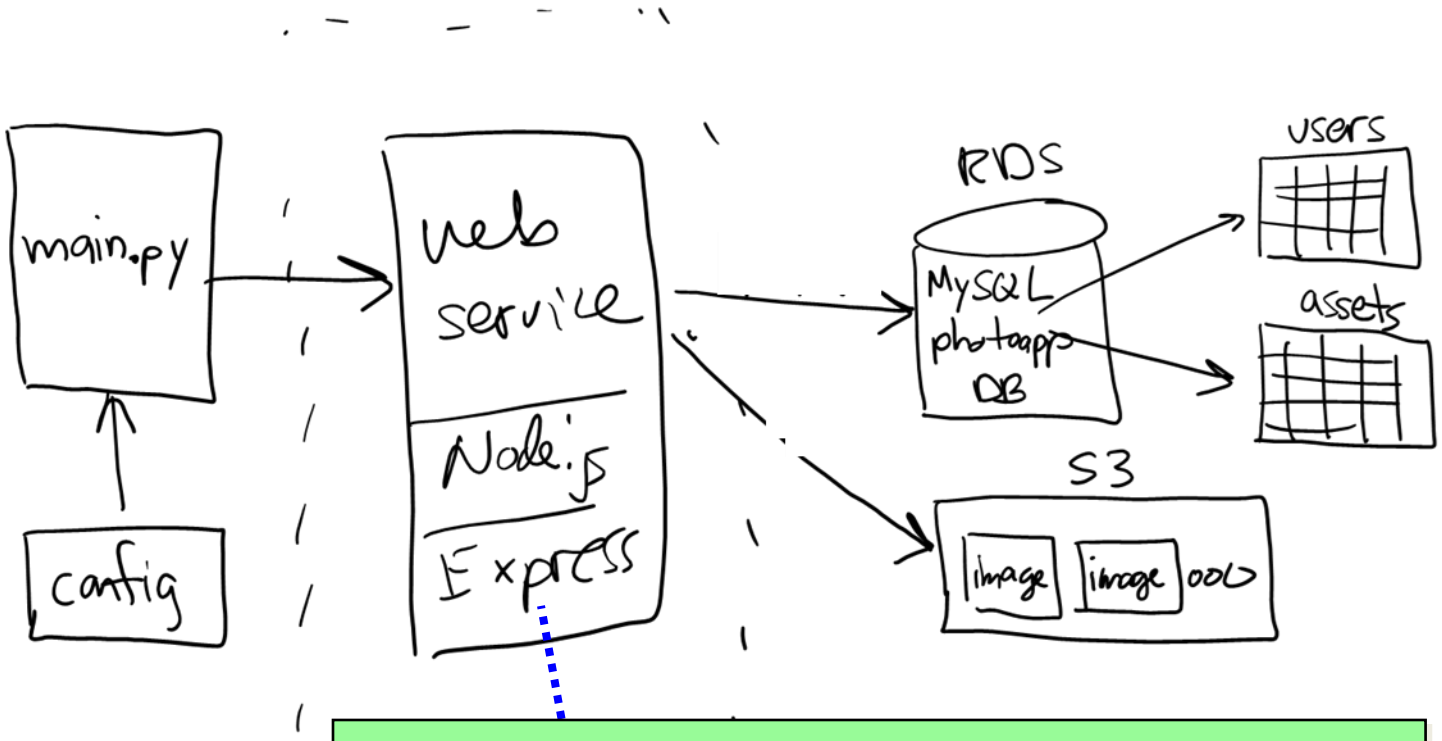


Constraints:

- Uniform Interface
- Stateless Operations
- Layered System
- Client-server based
- Cacheable
- Code on demand (optional)

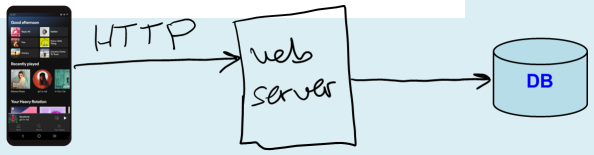
that adheres to web server standards

Project 02 --- photoapp 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) => {…});
```

/user



- **Design API to insert / update a user**
  - *Option #1: HTTP **GET**, sending data via parameters*

```
// data in URL /user?email=...&lastname=...&firstname=...
app.get('/user', (req, res) => {…});
```

```
// data in URL /user/:email/:lastname/:firstname
app.get('/user/:email/:last/:first', (req, res) => {…});
```

  - *Option #2: HTTP **PUT**, sending data in body of request*

```
// put a new user or update an existing user
app.put('/user', (req, res) => {…});
```

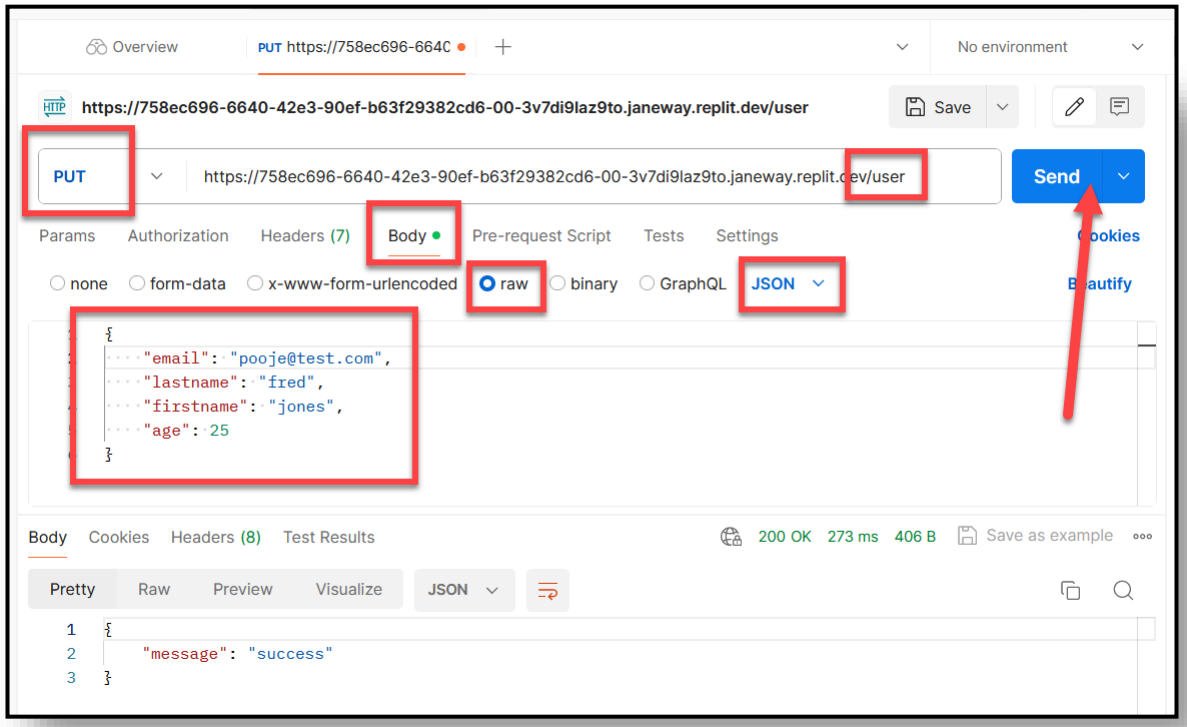
***PUT** makes more sense because we are inserting a user, not getting one. And we use PUT, not POST, because if repeated we want server to update not insert again.*

/user -- server-side

```
//
// Insert/modify user in database:
//
app.use(express.json({strict: false})); // deserialize incoming JSON

app.put('/user', (req, res) => {

   console.log("**Call to /user...");

   var data = req.body;  // JSON data is deserialized => JS object

   let email = data.email;

   .
   . // lookup email, if exists modify else insert
   .

   res.json(…);  // send a response when done

});
```

Testing with postman

- **Browsers only GET**
- **Postman can GET / PUT / POST**
  - *https://postman.com*

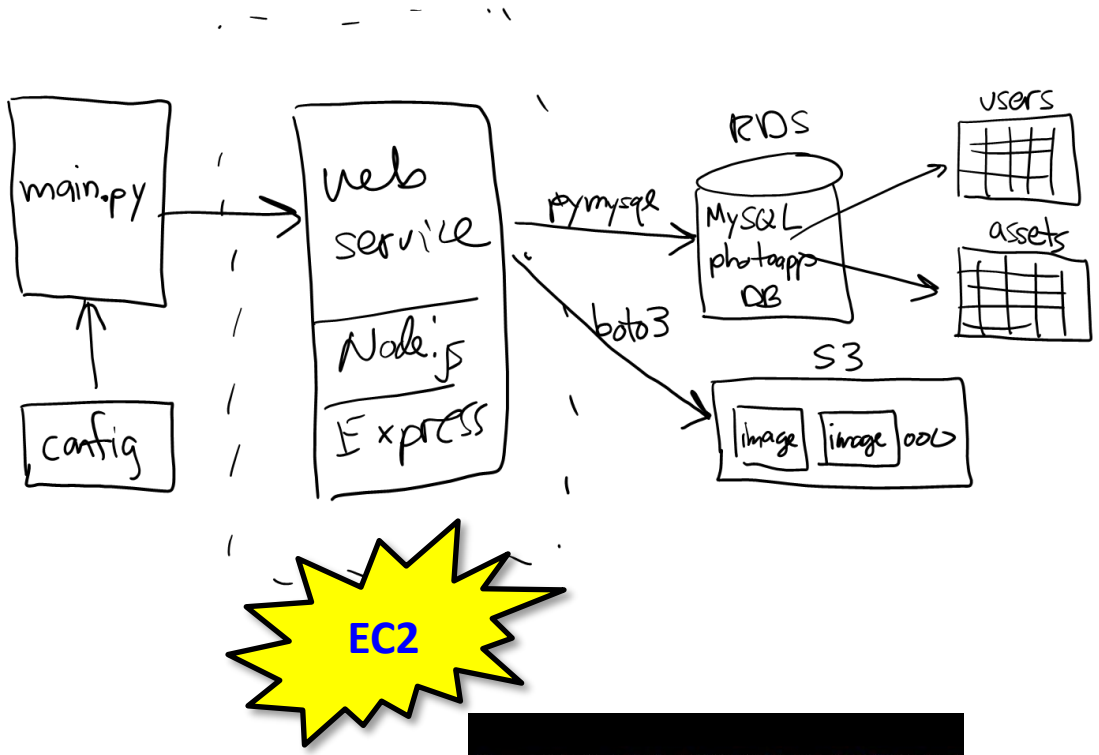


/user -- client-side

- The client PUTs user's data to the web service…


```
import requests

url = 'http://photoapp-web-service.com/user'

data = {
  "email":        "pooja@piazza.com",
  "lastname":     "sarkar",
  "firstname":    "pooja",
  "buckerfolder": "..."
}

response = requests.put(url, json=data)  # post data in JSON format

print("status code:", response.status_code)
```

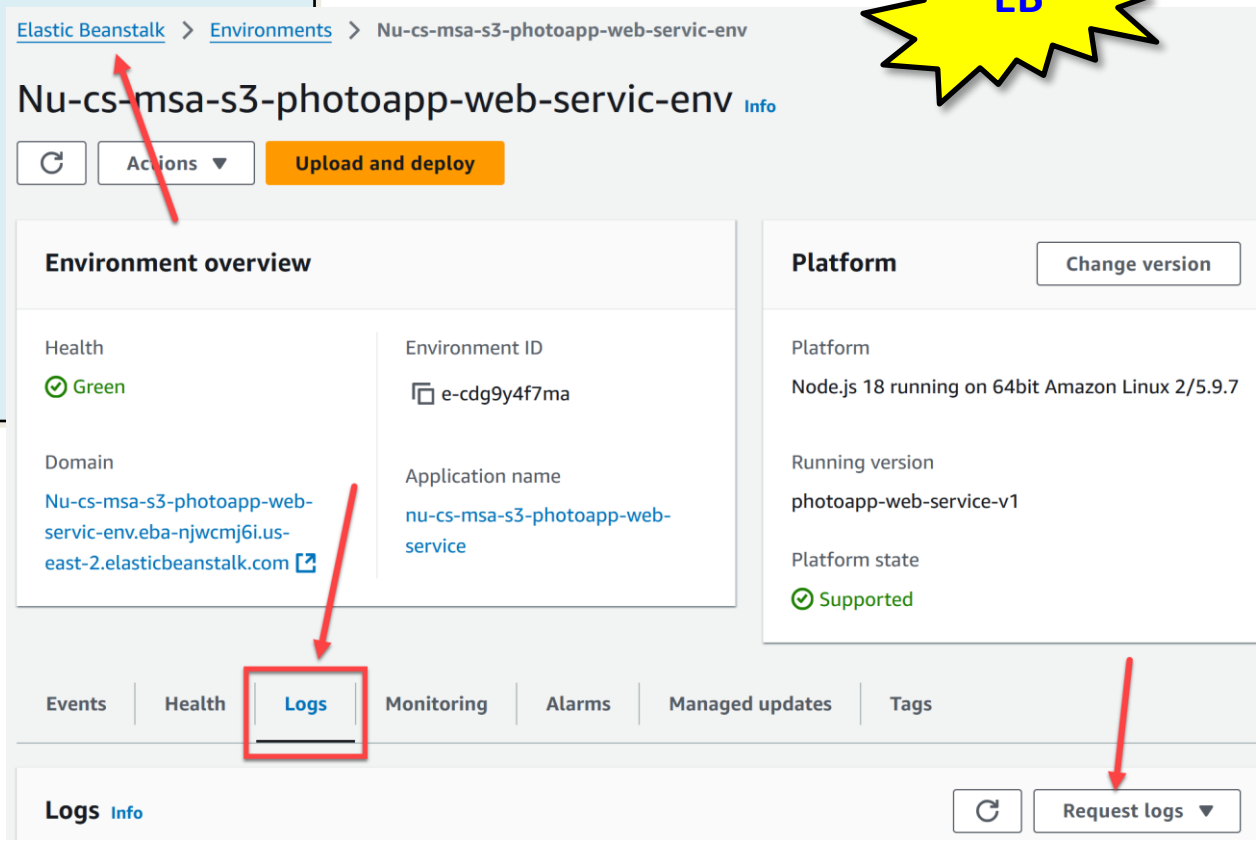
Debugging in the cloud



Print debugging

- **Print debugging is still very common --- view logs in AWS**

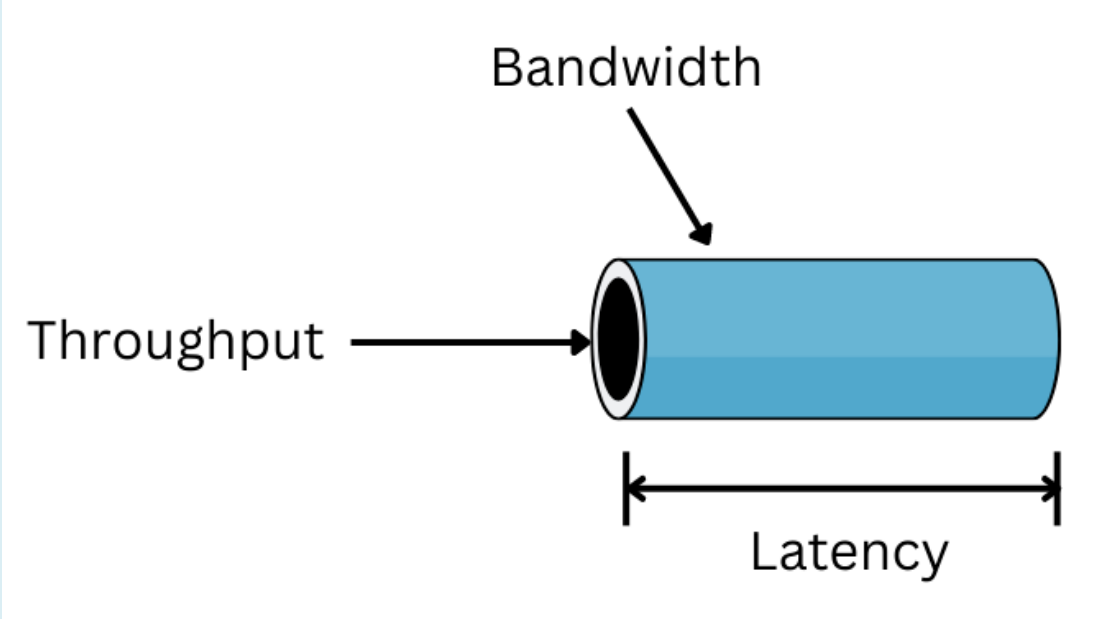
```
app.put('/user', (req, res) => {

    console.log("**Call to /user...");

    var data = req.body;

    console.log(data);

    .
    .
    .
```



```
----------------------------------------
/var/log/web.stdout.log
----------------------------------------
Oct 22 05:25:50 ip-172-30-0-130 web: /stats: calling S3...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats: calling RDS to get # of users...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats: calling RDS to get # of assets...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats done, sending response...
Oct 22 05:25:50 ip-172-30-0-130 web: **Call to get /stats...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats: calling S3...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats: calling RDS to get # of users...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats: calling RDS to get # of assets...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats done, sending response...
Oct 22 05:25:50 ip-172-30-0-130 web: **Call to get /stats...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats: calling S3...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats: calling RDS to get # of users...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats: calling RDS to get # of assets...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats done, sending response...
Oct 22 05:25:50 ip-172-30-0-130 web: **Call to get /stats...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats done, sending response...
Oct 22 05:25:50 ip-172-30-0-130 web: **Call to get /stats...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats: calling S3...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats: calling RDS to get # of users...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats: calling RDS to get # of assets...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats done, sending response...
Oct 22 05:25:50 ip-172-30-0-130 web: **Call to get /stats...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats: calling S3...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats: calling RDS to get # of users...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats: calling RDS to get # of assets...
Oct 22 05:25:50 ip-172-30-0-130 web: /stats done, sending response...
```

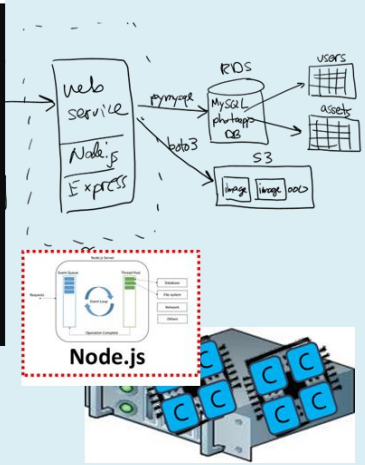
Performance of cloud-based apps

- **Latency**: how long does it take? (speed = response time)
- **Bandwidth**: how much data can be transmitted? (volume)
- **Throughput**: how much processing per time unit? (# of users)



ab == apache benchmark

```
hummel> ab -c 1 -n 10 http://photoapp-nu-web-service-env-2.eba-htg5fauw.us-east-2.elasticbeanstalk.com/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking photoapp-nu-web-service-env-2.eba-htg5fauw.us-east-2.elasticbeanstalk.com (be patient).....done


Server Software:        nginx
Server Hostname:        photoapp-nu-web-service-env-2.eba-htg5fauw.us-east-2.elasticbeanstalk.com
Server Port:            80

Document Path:          /
Document Length:        74 bytes

Concurrency Level:      1
Time taken for tests:   0.617 seconds
Complete requests:      10
```



- **Reference:** https://httpd.apache.org/docs/2.4/programs/ab.html
  - *Linux: sudo apt-get install apache2-tools*
  - *Mac: already installed (terminal window)*
  - *Windows: download apache (https://www.apachelounge.com/download/), extract /bin/ab.exe*

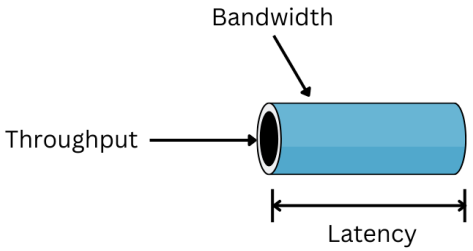
- **Usage: ab -k -c 10 -n 1000 URL**

  -k => keep-alive the TCP connection (cold-start vs. warm-start)

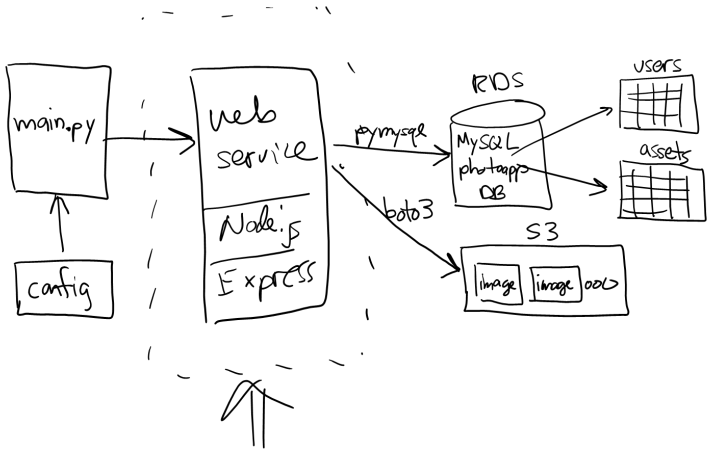
  -c => concurrency level (use this to simulate concurrent users / load on the server)

  -n => # of requests (use this to get a more accurate "average" response time)

Question

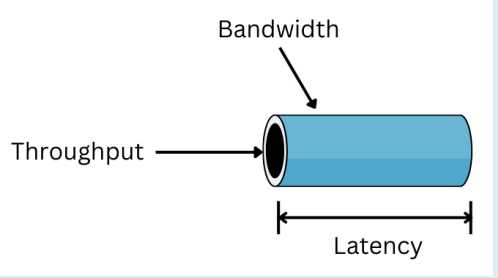


- **Baseline latency is _____ ms**
- **Speed of RDS _____ ms**
- **Speed of S3 _____ ms**

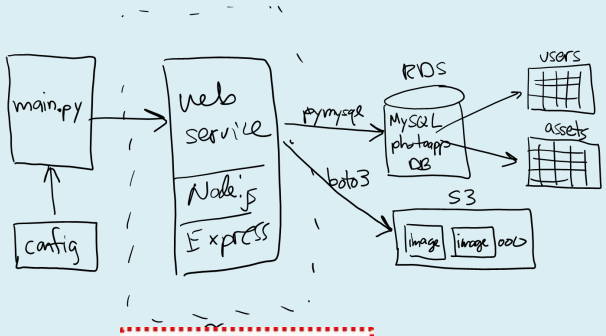


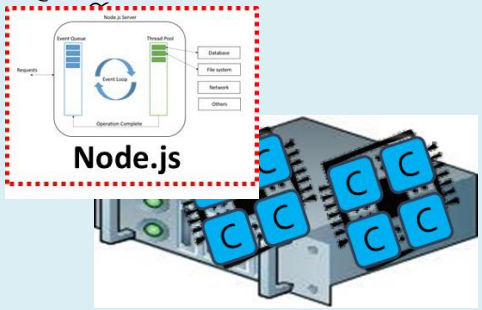

```
//
// PhotoApp web service
//
app.get('/stats', (req, res) => {…});
app.get('/users', (req, res) => {…});
app.get('/assets', (req, res) => {…});
app.get('/bucket', (req, res) => {…});
app.get('/image/:assetid', (req, res) => {…});
```

Question



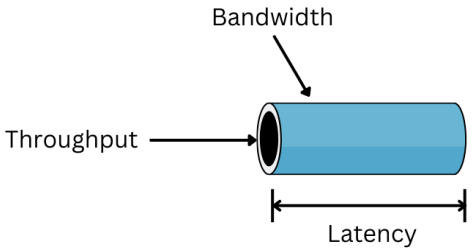
- **Throughput…**
- **Node.js is single-threaded and runs on one core. How many users can it support before latency suffers?**



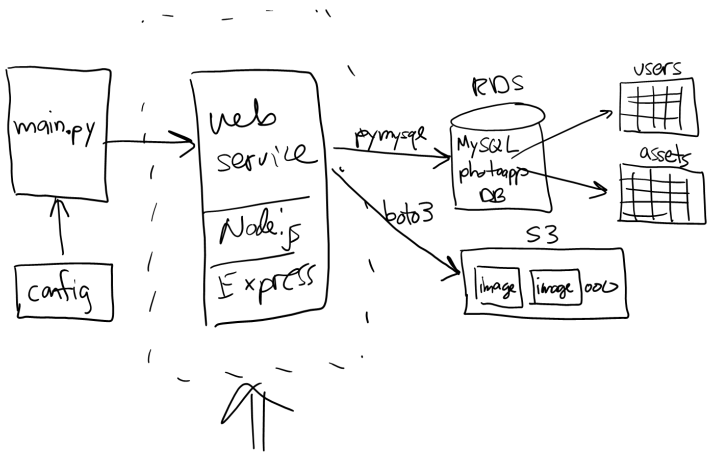



```
//
// PhotoApp web service
//
app.get('/stats', (req, res) => {…});
app.get('/users', (req, res) => {…});
app.get('/assets', (req, res) => {…});
app.get('/bucket', (req, res) => {…});
app.get('/image/:assetid', (req, res) => {…});
```

Question



- **How efficient is the network (bandwidth)?**
- **Asset 1004 is 43K, asset 1014 is 1,964K --- 46x bigger**
  - *Time to download image # 1004 is _____ ms*
  - *How much longer does it take to download image # 1014? (_____ ms)*




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

That's it, thank you!