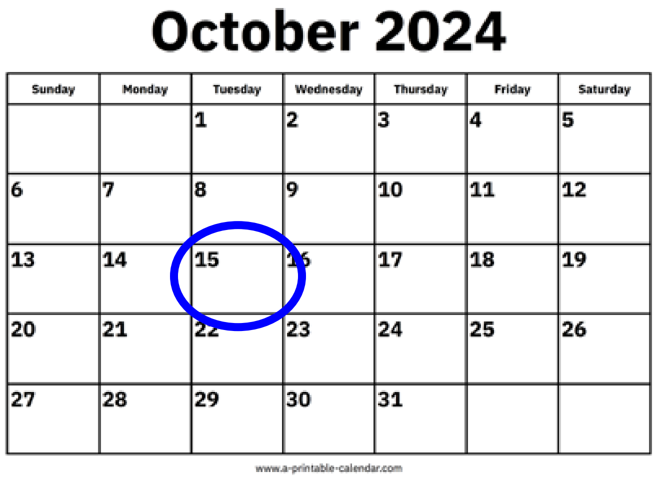
CS 310 : Scalable Software Architectures

Class session on Tuesday, October 15th



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

Northwestern University

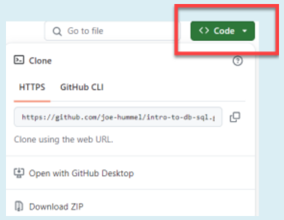
Notes:

- ***Focus this week***:
  - *Designing and building web services*
- ***No class session on Thursday!***
- ***Today's session is being recorded…***
- ***Project 02*** *was released*
  - *Part 01: web service for photoapp, rewrite client*
  - *Part 02: deploy to AWS EC2*
  - *Due Friday October 25th*

Getting the necessary software

1. **Make sure Docker Desktop is running**

2. **Download files you need for today**
   - https://github.com/joe-hummel/web-service-async-demo



3. **Update the repo's .ini file. Two options:**

   a) *Start your database server, then copy over your photoapp-config.ini file from project 01 to the repo*

   b) *Open your project 01 photoapp-config.ini file, copy your bucket name and s3readwrite info, and paste into .ini file in the repo*

```
[s3]
bucket_name = YOUR_BUCKET_NAME
```

```
[s3readwrite]
region_name = us-east-2
aws_access_key_id = YOUR_READWRITE_S3_ACCESS_KEY_ID
aws_secret_access_key = YOUR_READWRITE_S3_SECRET_ACCESS_KEY
```

Build and run docker

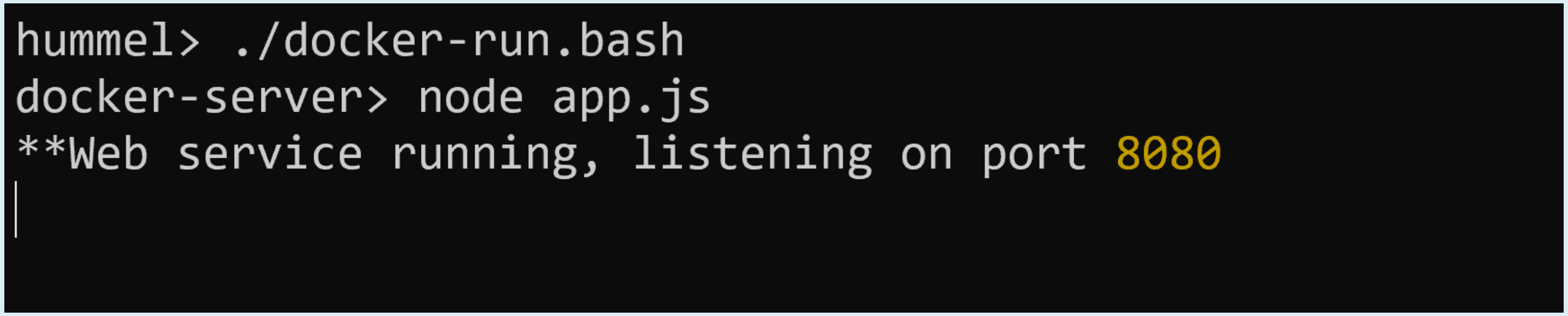
4. Open a terminal window, navigate to repo:

Linux/Mac/Windows WSL:

1) Open terminal, navigate to repo folder
2) `chmod 755 *.bash`
3) `./docker-build.bash`
4) `./docker-run.bash`

Windows:

1) Open Powershell, navigate to repo folder
2) `.\docker-build.bat`
3) `.\docker-run.bat`



```
hummel> ./docker-run.bash
docker-server> node app.js
**Web service running, listening on port 8080
```

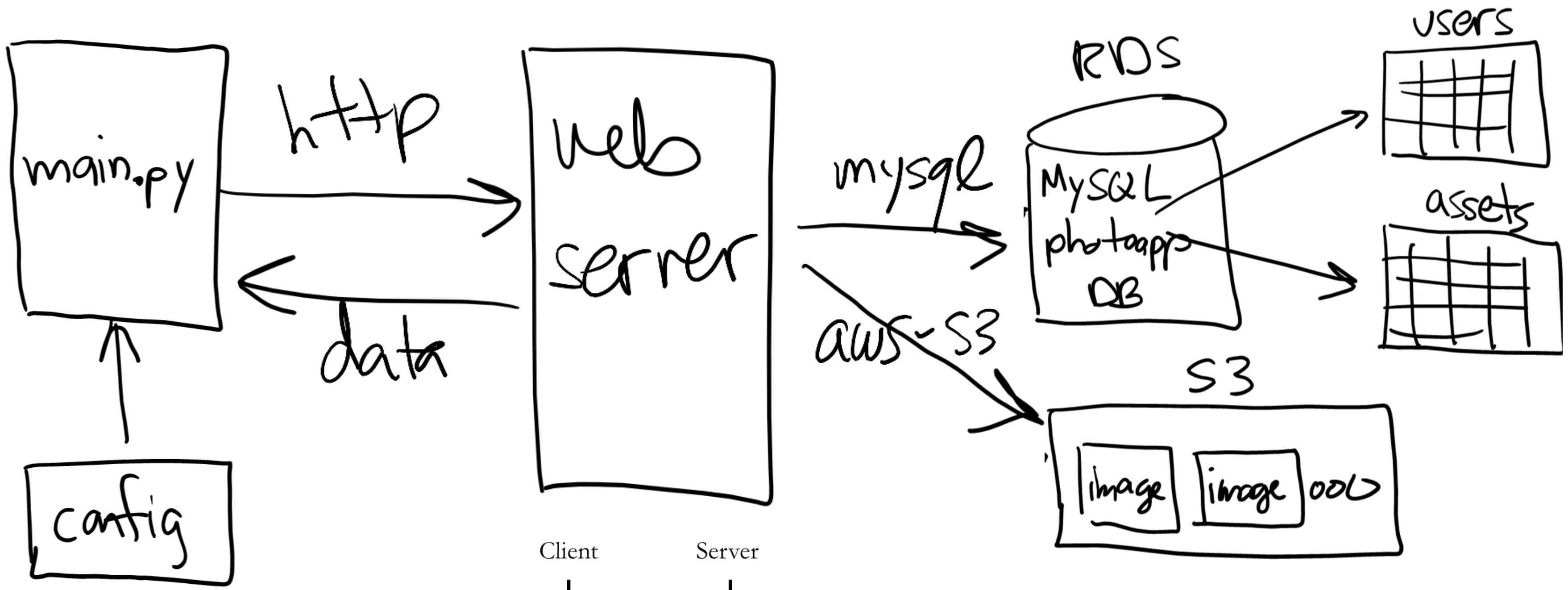
Common docker errors

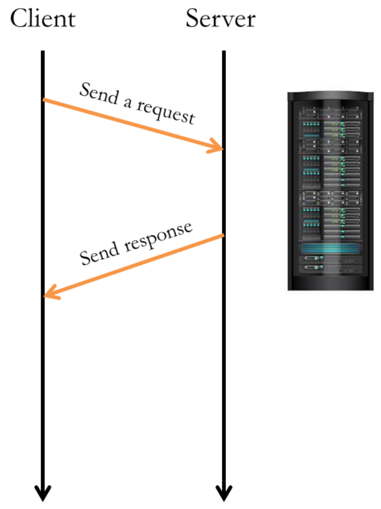
1. **"docker" command not found**
   - *Uninstall and reinstall Docker Desktop*

2. **When you try to build, you are not authorized**
   - *`docker login -u docker-username`*

3. **When you try to run, you get errors like "bash: $\r: command not found"**
   1. *If you see the* ***docker****> prompt, type* ***exit***
   2. `((Get-Content .bashrc) -join "`n") + "`n" | Set-Content -NoNewLine .bashrc`

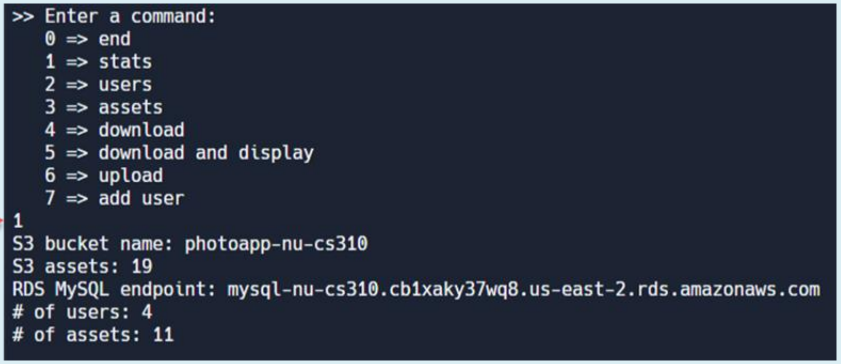
Project 02


main.py
http
data
config
web server
mysql
aws-S3
RDS
MySQL photoapp DB
users
assets
S3
image
image
ooo

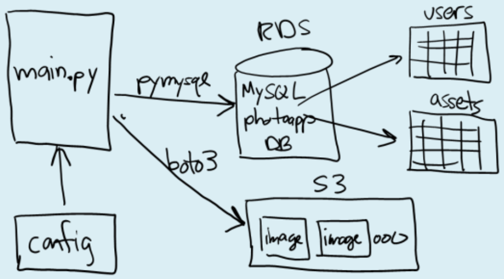

Client
Server
Send a request
Send response

Accessing S3

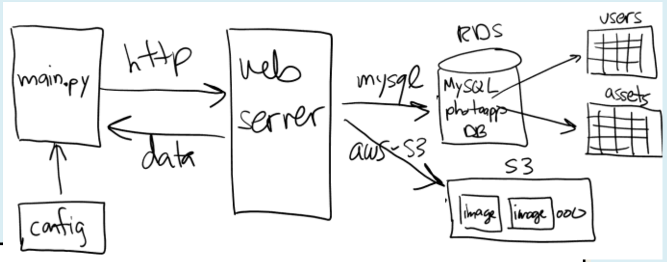
- **Recall the "stats" command from project 01...**

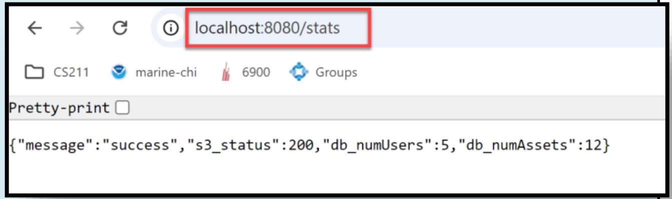


```
>> Enter a command:
   0 => end
   1 => stats
   2 => users
   3 => assets
   4 => download
   5 => download and display
   6 => upload
   7 => add user
1
S3 bucket name: photoapp-nu-cs310
S3 assets: 19
RDS MySQL endpoint: mysql-nu-cs310.cb1xaky37wq8.us-east-2.rds.amazonaws.com
# of users: 4
# of assets: 11
```



- **Project 02 has a similar command...**





```
{"message":"success","s3_status":200,"db_numUsers":5,"db_numAssets":12}
```

```
app.get('/stats', (req, res) => {

  (1) call S3, get status of bucket
  (2) call MySQL, get # of users in the users table
  (3) call MySQL, get # of assets in the assets table
  (4) res.json({ "message": ...,
                 "s3_status": ...,
                 "db_numUsers": ...,
                 "db_numAssets": ... });
});
```

Attempt #1

```
app.get('/stats', (req, res) => {

  console.log("**Call to get /stats...");


  let input = {
    Bucket: s3_bucket_name
  };

  let command = new HeadBucketCommand(input);
  let s3_response = photoapp_s3.send(command);


  res.json({ "message": "success",
             "s3_status": s3_response["$metadata"]["httpStatusCode"],
             "db_numUsers": -1,
             "db_numAssets": -1 });
});
```

S3 call is asynchronous, you have to wait for response…

Promises

- **The modern way to wait…**
- **A promise is an object that eventually resolves to a value**
  - *When you need the value, you "await" for it*
  - *Example: s3.send(…)*

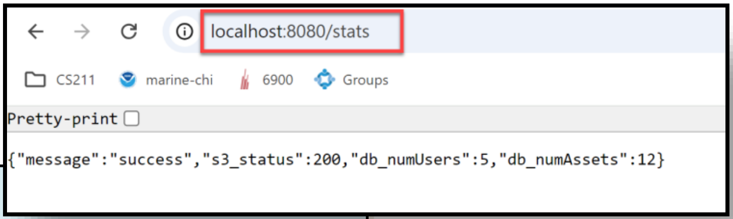
```
app.get('/path', async (req, res) => {
  try {
    let response = F(params);  // F returns a promise

    let result = await response;

    res.json(result);
  }
  catch(err) { res.status(500).json(…); }
});
```

Solution

```
app.get('/stats', async (req, res) => {

  console.log("**Call to get /stats...");

  let input = {
    Bucket: s3_bucket_name
  };

  let command = new HeadBucketCommand(input);
  let s3_response = photoapp_s3.send(command);

  let s3_result = await s3_response;

  res.json({ "message": "success",
             "s3_status": s3_result["$metadata"]["httpStatusCode"],
             "db_numUsers": -1,
             "db_numAssets": -1 });
});
```

Accessing MySQL

- The /stats function is also supposed to get the # of users and # of assets in the database…


localhost:8080/stats

Pretty-print

```
{"message":"success","s3_status":200,"db_numUsers":5,"db_numAssets":12}
```

```
app.get('/stats', (req, res) => {

  call S3, get status code of bucket

  call MySQL to get # of users in the users table

  call MySQL to get # of assets in the assets table

  res.json({ "message": ...,
             "s3_status": ...,
             "db_numUsers": ...,
             "db_numAssets": ... });
});
```

Callbacks

- **MySQL library is based on callbacks, not promises…**
- **In this case, the result is ONLY available inside the callback**
  - *Example: db.query(…)*

```
app.get('/path', (req, res) => {
  try {

    db.query(sql, (err, result, …) => {
        try {
          if (err)
            res.status(500).json(err.message);
          else
            res.json(result);
        }
        catch(err) {…}
    });

  }
  catch(err) {…}
});
```

Solution

```
app.get('/stats', async (req, res) => {

  console.log("**Call to get /stats...");

  let input = {
    Bucket: s3_bucket_name
  };

  let command = new HeadBucketCommand(input);
  let s3_response = photoapp_s3.send(command);

  let sql = "select count(*) as NumUsers from users;";

  photoapp_db.query(sql, async (err, db_result, _) => {

    if (err) {
      res.status(500).json({ … });
    }
    else {
      let row = db_result[0];  // we got one row back, extract it
      let s3_result = await s3_response;

      res.json({ "message": "success",
                 "s3_status": s3_result["$metadata"]["httpStatusCode"],
                 "db_numUsers": row["NumUsers"],
                 "db_numAssets": -1 });
    }//else

  });
});
```

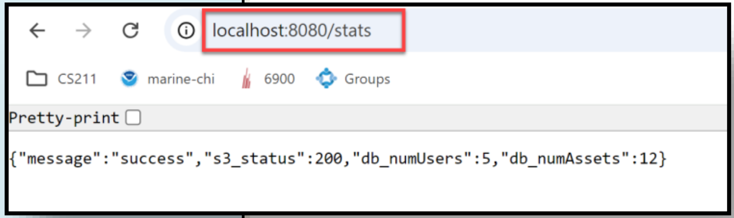
*We **await** for S3 inside the callback so it runs concurrently with MySQL…*

*We have to move the **res.json( )** into the callback as well because this is where the results are*

What about the # of assets?

- We also have to count the # of assets in the assets table…
- This implies nesting ANOTHER callback and moving res.json( )…

```
app.get('/stats', async (req, res) => {
  .
  .
  .
  let sql = "select count(*) as NumUsers from users;";

  photoapp_db.query(sql, async (err, db_result, _) => {

    if (err) {
      res.status(500).json({ … });
    }
    else {
      let user_row = db_result[0];

      let sql = "select count(*) as NumAssets from assets;";

      photoapp_db.query(sql, async (err, db_result, _) => {

        if (err) {
          res.status(500).json({ … });
        }
        else {
          let asset_row = db_result[0];
          let s3_result = await s3_response;

          res.json({ "message": "success",
             "s3_status": s3_result["$metadata"]["httpStatusCode"],
             "db_numUsers": user_row["NumUsers"],
             "db_numAssets": asset_row["NumAssets"] });
        }
      });
    }
  });
});
```



```
{"message":"success","s3_status":200,"db_numUsers":5,"db_numAssets":12}
```

THERE'S GOT TO BE A
BETTER WAY

Goal

```
app.get('/stats', (req, res) => {

  (1) call S3, get status of bucket
  (2) call MySQL, get # of users in the users table
  (3) call MySQL, get # of assets in the assets table
  (4) res.json({ "message": ...,
                 "s3_status": ...,
                 "db_numUsers": ...,
                 "db_numAssets": ... });
});
```

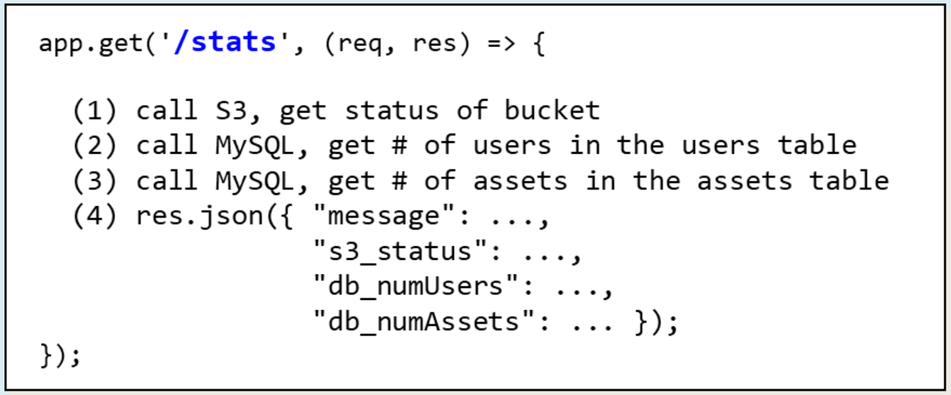
Step #1

- **Turn the callback into a promise, then await!**

```
let db_result = new Promise( (resolve, reject) => {
     let sql = "select count(*) as NumUsers from users;";
     photoapp_db.query(sql, (err, rows, …) => {
       try {
         if (err)
           reject(err);
         else
           resolve(rows[0]);  // we want the first row
       }
       catch(err) { reject(err); }
     });
});

let user_row = await db_result;
```

Step #2

- Turn both callbacks into promises…
- Get it to work synchronously with await



```
app.get('/stats', (req, res) => {

  (1) call S3, get status of bucket
  (2) call MySQL, get # of users in the users table
  (3) call MySQL, get # of assets in the assets table
  (4) res.json({ "message": ...,
                 "s3_status": ...,
                 "db_numUsers": ...,
                 "db_numAssets": ... });
});
```

Step #3

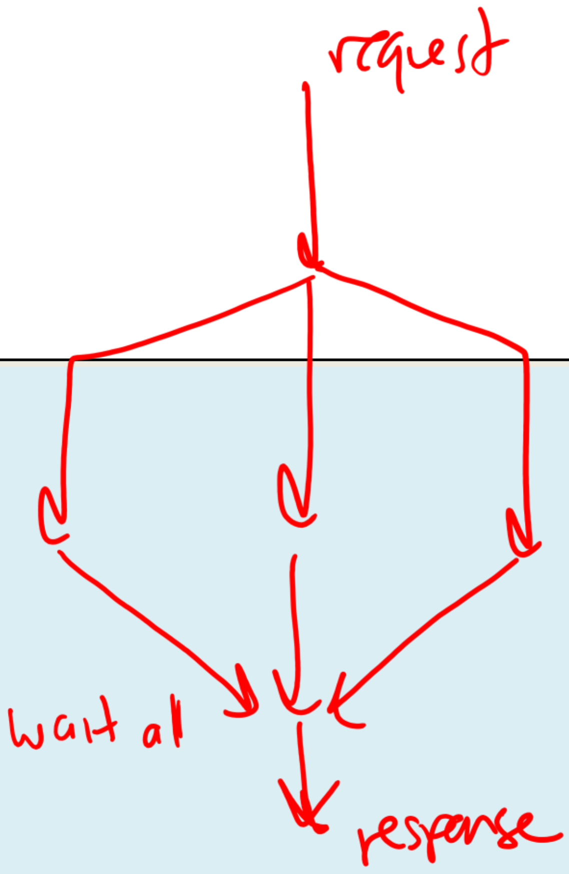
- Now let's maximize concurrency with Promise.all

```
let results = await Promise.all([s3_result, db_result, db_result2]);

s3_result  = results[0];  // first result
users_row  = results[1];  // second result
assets_row = results[2];  // third result

res.json({ ... });
```

Step #3

- Now let's maximize concurrency with Promise.all

```
let results = await Promise.all([s3_result, db_result, db_result2]);

s3_result  = results[0];  // first result
users_row  = results[1];  // second result
assets_row = results[2];  // third result

res.json({ ... });
```



That's it, thank you!