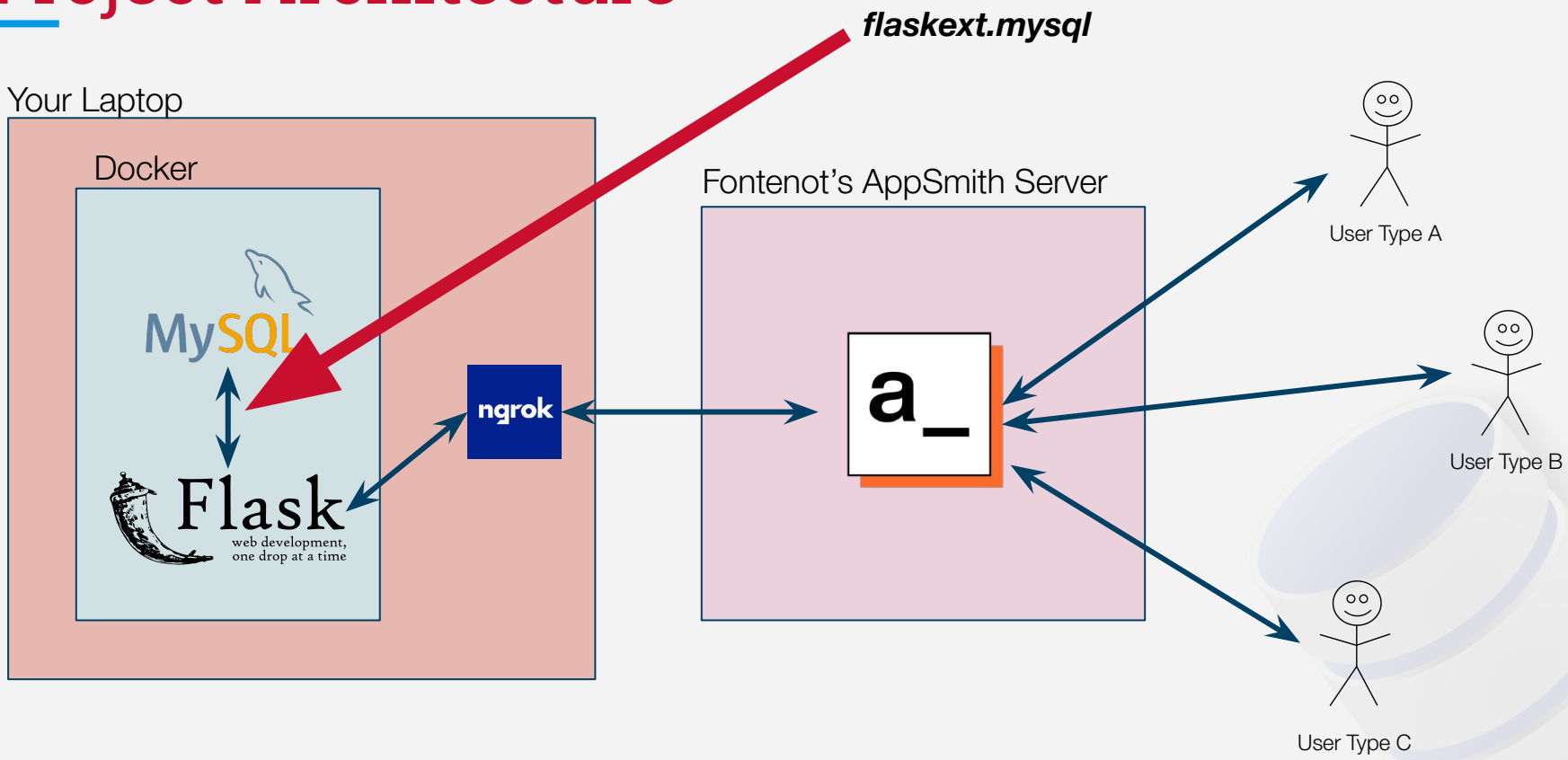
Project: MySQL and Flask

Mark Fontenot, PhD
Northeastern University

Taking a Step Back

- Where is all this stuff coming from?
- I'm giving you bite-sized chunks of integrated material
  - Python
  - Flask
  - MySQL
  - Flask <-> MySQL
  - Docker
- How do you learn more?

Project Architecture



Starter Code for Today

On the Webpage, you can d/l the starter project:

- Flask + MySQL
  - In Class Starter Project for Flask + My SQL

Or click > here < to download.

Step 1: Update the Flask App Dockerfile

- In order for the Flask app to connect to MySQL securely, the server it runs on will need some additional libraries.
- Add the line in the red box below to **the_app/Dockerfile** and save.

```
FROM python:3.9-alpine

# make a folder inside the container named /usr/src/app
#RUN mkdir /usr/src/app/

# set the new folder we just created as the working directory
WORKDIR /usr/src/app/

# needed so python crpytography stuff can be installed
RUN apk add gcc musl-dev python3-dev libffi-dev openssl-dev

# copy everything from the current folder (the one where this
# Dockerfile lives) into the folder we created above
```

Step 2: Update requirements.txt

- We need 2 additional Python libraries to connect to MySQL: **flask-mysql** and **cryptography**.
- Open **the_app/requirements.txt** and add the two additional lines shown below.

```
1  flask
2  flask-mysql
3  cryptography
```

Step 3: Create a Database Bootstrap File

- Create a new folder named **db_bootstrap** at the same level as my_app.
- Create a new file inside that new folder called **create_db.sql**
- Add the SQL on the right to that file and save it.

```sql
CREATE DATABASE cool_db;
CREATE USER 'webapp'@'%' IDENTIFIED BY 'abc123';
GRANT ALL PRIVILEGES ON cool_db.* TO 'webapp'@'%';
FLUSH PRIVILEGES;

-- Move into the database we just created.
-- TODO: If you changed the name of the
-- database above, you need to change it here too.
USE cool_db;

-- Put your DDL
CREATE TABLE test_table (
  name VARCHAR(20),
  color VARCHAR(10)
);

-- Add sample data.
INSERT INTO test_table
  (name, color)
VALUES
  ('dev', 'blue'),
  ('pro', 'yellow'),
  ('junior', 'red');
```

Step 4: Update the docker-compose.yml

- Add a volume command line in **db** service of **docker-compose.yml** to map the **db_bootstrap** folder into the container as read-only.
- The MySQL Docker container is setup to automatically execute any SQL files that are mapped to or copied into **/docker-entrypoint-initdb.d**

```yaml
db:
    # basing it on mysql v.8.0.x
    image: mysql:8
    # mapping container port 3306 to host port 3306
    # (note: 3306 is the default mysql port)
    ports:
      - 3306:3306

    # anything in (or mounted in) /docker-entrypoint-initdb.d in the container
    # will automatically be executed when the container is created
    volumes:
    - ./db_bootstrap:/docker-entrypoint-initdb.d:ro

    # Provide an environtment variable containing
    # whatever we want the root password to be
    # NOTE: This is for demo purposese only.
    # abc123 is a TERRIBLE password and you wouldn't ever
    # want to store the root password in a shared public file
    environment:
      - MYSQL_ROOT_PASSWORD=abc123
```

Pause: Check that things are working

- Stop any running containers
- **`docker compose down`**
- **`docker compose build`**
- **`docker compose up`**

- *Do you see any errors?*
    - *Check both containers*

- From DataGrip, connect to **cool_db** and select all the data from the table within.

Pause: New Docker Command

You can spin up only one service from a docker-compose.yml file by putting the name of the service at the end of docker compose up

Service names in the docker-compose.yml

```
docker compose up db
docker compose up my-api-service
```

```
services:
  my-api-service:
  # creating a new service
  db:
```

Step 5: Setting up a connection in Python

In **the_app/app.py**:

```
from flask import Flask, jsonify
from flaskext.mysql import MySQL

# create a flask object
app = Flask(__name__)

# add db config variables to the app object
app.config['MYSQL_DATABASE_HOST'] = 'db'
app.config['MYSQL_DATABASE_PORT'] = 3306
app.config['MYSQL_DATABASE_USER'] = 'webapp'
app.config['MYSQL_DATABASE_PASSWORD'] = 'abc123'
app.config['MYSQL_DATABASE_DB'] = 'cool_db'

# create the MySQL object and connect it to the
# Flask app object
db_connection = MySQL()
db_connection.init_app(app)
```

If you had to change the port in DataGrip to connect to MySQL, change it here, too.

Step 6: Add a Route to Retrieve Data

```
@app.route('/db_test')
def db_testing():
    cur = db_connection.get_db().cursor()
    cur.execute('select * from test_table')
    row_headers = [x[0] for x in cur.description]
    json_data = []
    theData = cur.fetchall()
    for row in theData:
        json_data.append(dict(zip(row_headers, row)))
    return jsonify(json_data)
```

Step 7: Test the Route in your Browser.

1. Stop the containers.
2. **docker compose down**
3. **docker compose build**
4. **docker compose up**
5. In browser, go to **127.0.0.1:9000/db_test**

127.0.0.1:9000/db_test


```
[
  - {
        color: "blue",
        name: "dev"
    },
  - {
        color: "yellow",
        name: "pro"
    },
  - {
        color: "red",
        name: "junior"
    }
]
```