Security in Multi-tier Systems (part 02)

- **Best practices**
- **Least-privilege**
- **Authentication**
- **Authorization**

- [CS 308 / 350 / 354]

Security

- **Security is a large, complex field**
  - *Software*
  - *Hardware*
  - *Networking*
  - *Encryption*
  - *Authentication*
  - *Authorization*
  - *Best practices*
  - *…*

apply as many of these as possible ("many lines of defense")

Best practices

- **The most common:**
  - *Don't store passwords in clear text*
    - Input into program variable, immediately use, clear variable
    - Store hashed version in database, not actual password ("hash" => one-way encryption, see project 04)
  - *Don't store credentials / keys on the client*
    - And if possible, eliminate use of keys altogether
  - *Encrypt communication with web server, RDS, S3, etc.*
    - Web server with https; RDS by opening connection with SSL/TLS
  - *Encrypt data at rest (e.g. assets in S3)*
  - *Principle of least-privilege*

Principle of least-privilege

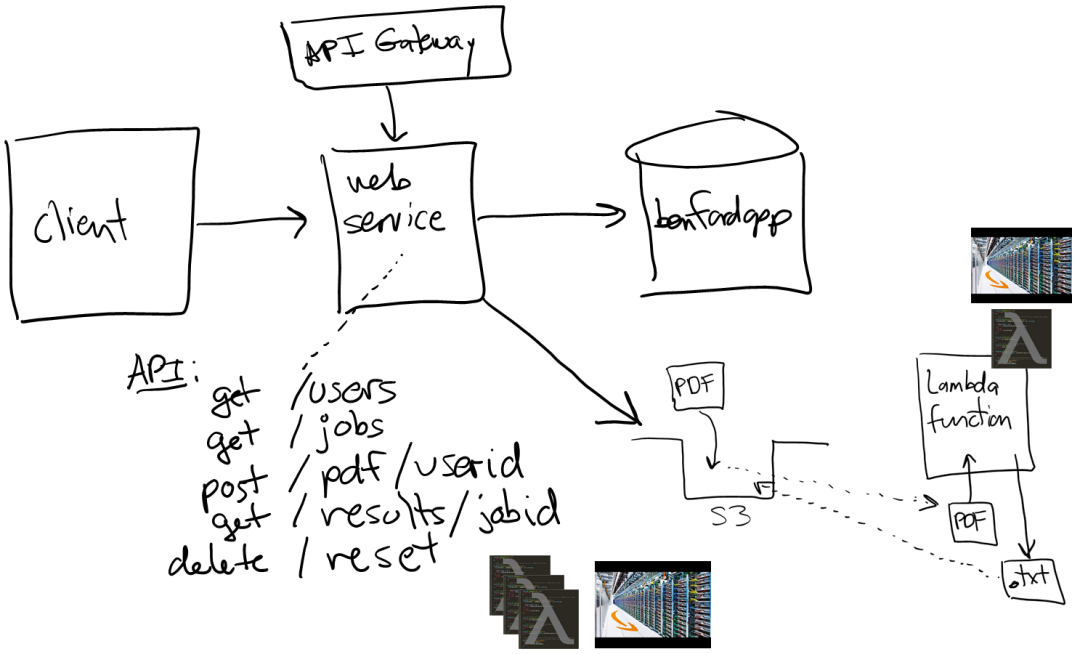
- "**Provide just enough permissions, no more**"

- **Example:**
  - *Lambda function needs to read data from MySQL*
  - *Is the following "config.ini" good or bad?*

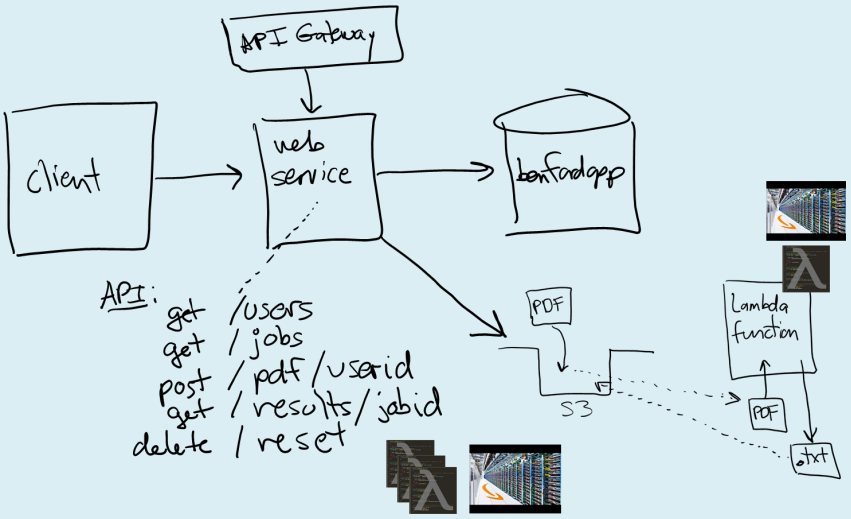
```
[s3]
bucket_name = photoapp-nu-cs310-p01test

[rds]
endpoint = mysql-nu-cs310.fe1xaky39aq8.us-east-2.rds.amazonaws.com
port_number = 3306
region_name = us-east-2
user_name = admin
user_pwd = password123
db_name = photoapp

[s3readonly]
region_name = us-east-2
aws_access_key_id = AK...
aws_secret_access_key = rc...

[s3readwrite]
region_name = us-east-2
aws_access_key_id = AK...
aws_secret_access_key = pj...
```

users

- **Software executes under some identity ("user")**
- **That identity dictates what the software can do…**



Example: "users" in project 03




```
43      config_file = 'config.ini'
44      os.environ['AWS_SHARED_CREDENTIALS_FILE'] = config_file
45
46      configur = ConfigParser()
47      configur.read(config_file)
48
49      #
50      # configure for S3 access:
51      #
52      s3_profile = 's3readwrite'
53      boto3.setup_default_session(profile_name=s3_profile)
```

benfordapp-read-write

s3readonly

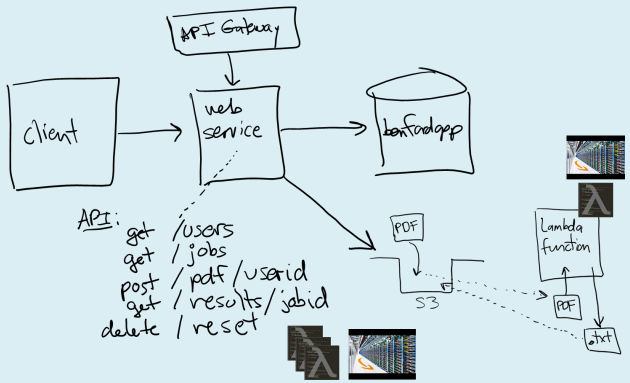
s3readwrite

```
1  [s3]
2  bucket_name = photoapp-nu-cs310-p01test
3
4  [rds]
5  endpoint = mysql-nu-cs310-p01test.cb1x[redacted].us-east-2.rds.amazonaws.com
6  port_number = 3306
7  region_name = us-east-2
8  user_name = benfordapp-read-write
9  user_pwd = [redacted]
10 db_name = benfordapp
11
12 [s3readonly]
13 region_name = us-east-2
14 aws_access_key_id = AKIAWI[redacted]
15 aws_secret_access_key = rcgcy[redacted]
16
17 [s3readwrite]
18 region_name = us-east-2
19 aws_access_key_id = AKIAW[redacted]
20 aws_secret_access_key = pjqQg[redacted]
```

authentication vs. authorization

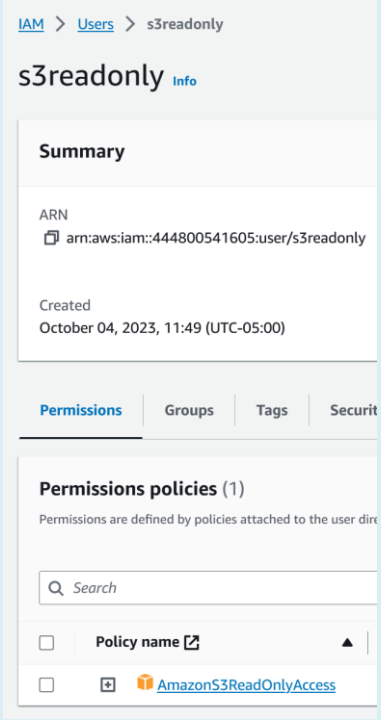
- **Authentication: who are you?**
- **Authorization: what are you allowed to do?**

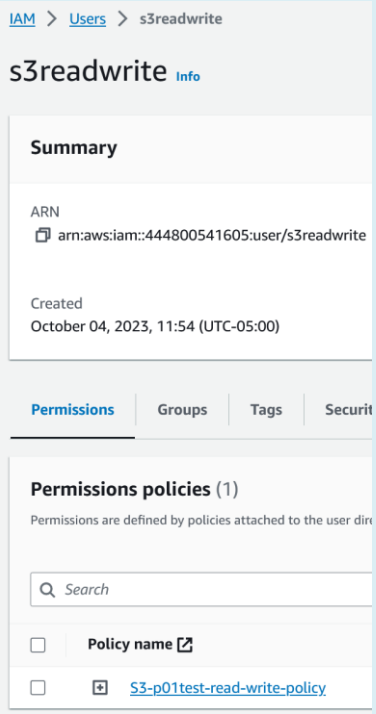
Example: project 03




```
--
-- SQL
--

USE benfordapp;

CREATE USER 'benfordapp-read-only' IDENTIFIED BY 'abc123!!';
CREATE USER 'benfordapp-read-write' IDENTIFIED BY 'def456!!';

GRANT SELECT, SHOW VIEW ON benfordapp.*
      TO 'benfordapp-read-only';
GRANT SELECT, SHOW VIEW, INSERT, UPDATE, DELETE, DROP, CREATE, ALTER ON benfordapp.*
      TO 'benfordapp-read-write';

```

```
[s3]
bucket_name = photoapp-nu-cs310-p01test

[rds]
endpoint = mysql-nu-cs310-p01test.cb1[redacted].us-east-2.rds.amazonaws.com
port_number = 3306
region_name = us-east-2
user_name = benfordapp-read-write
user_pwd = [redacted]
db_name = benfordapp

[s3readonly]
region_name = us-east-2
aws_access_key_id = AKIAW[redacted]
aws_secret_access_key = rcgcy[redacted]

[s3readwrite]
region_name = us-east-2
aws_access_key_id = AKIAW[redacted]
aws_secret_access_key = pjqQg[redacted]
```

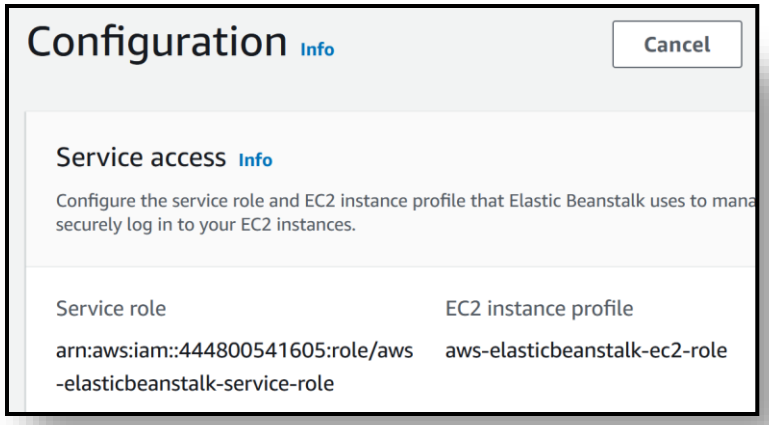


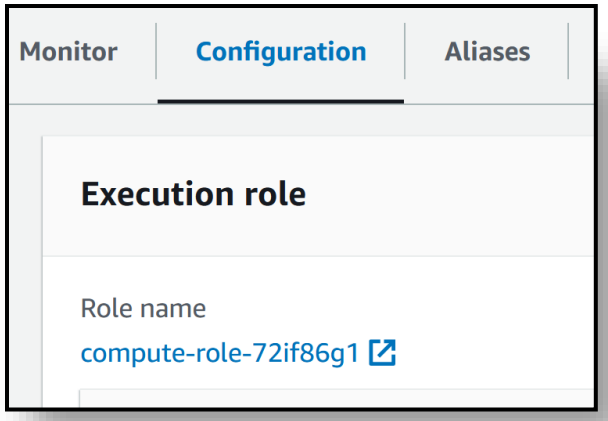


Moving away from access keys

- **Access keys are generally not a good idea**
  - *They get embedded into code and hard to change*
  - *Easy to leak out (e.g. upload to GitHub --- AWS will detect!)*

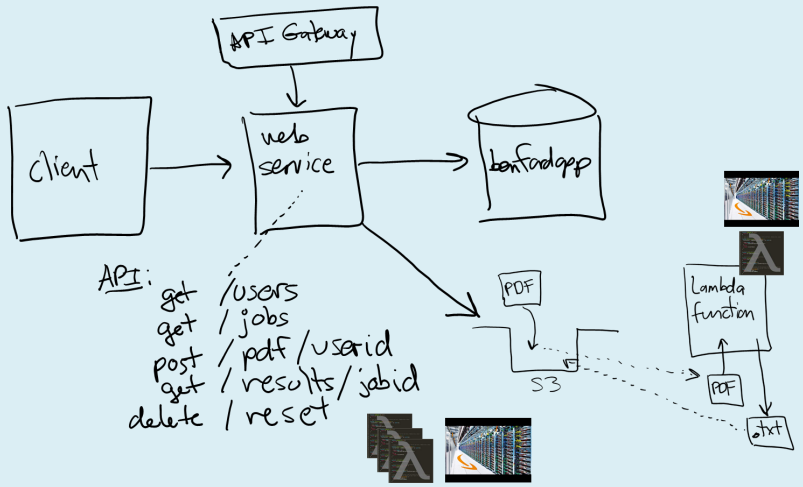
- **On server-side, assign permissions to users / roles**
  - *Example: assign RDS and S3 permissions to EB instance*
  - *Example: assign RDS and S3 permissions to lambda functions*





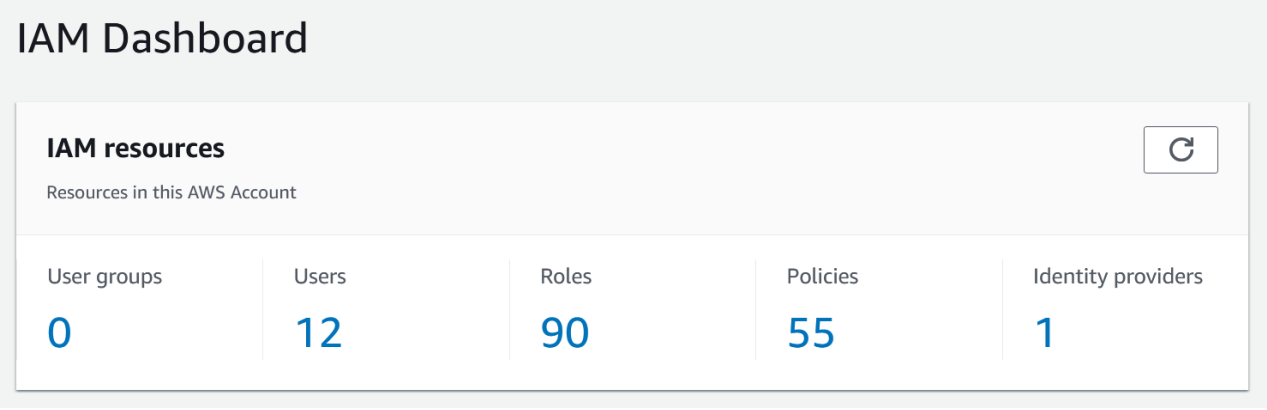
Demo: access keys => add policy to lambda role

- **Project 03**
- **Open proj03_upload lambda:**
  - *Comment out code that sets s3_profile, calls boto3.setup_default_session( )*
  - *Test upload --- it will fail*
  - *Configuration tab… Permissions…*
  - *Click on "arrow" next to Role name*
  - *Now looking at role in IAM*
  - *Add permission…*
  - *Attach policy…*
  - *Find S3-read-write-access policy*
  - *Select*
  - *Add*
  - *Run and test --- should work!*
  - *Can now delete S3-related keys…*



IAM

- **AWS Identity and Access Management service**
- **Critically important service for security**
  - *Create 1 user per client?*
  - *Create 1 user for ALL clients?*
  - *Use groups to categorize users?*
  - *Define common policies, attach to roles / users / groups?*



That's it, thank you!