Keeping objects secure

INTRODUCTION TO AWS BOTO IN PYTHON

Maksim Pecherskiy
Data engineer

datacamp

Why care about permissions?

```
df = pd.read_csv('https://gid-staging.s3.amazonaws.com/potholes.csv')
```

```
/usr/local/Cellar/python/3.7.1/Frameworks/Python.framework/Versions/3.7/lib/pyth
    647 class HTTPDefaultErrorHandler(BaseHandler):
    648     def http_error_default(self, req, fp, code, msg, hdrs):
--> 649         raise HTTPError(req.full_url, code, msg, hdrs, fp)
    650
    651 class HTTPRedirectHandler(BaseHandler):

HTTPError: HTTP Error 403: Forbidden
```

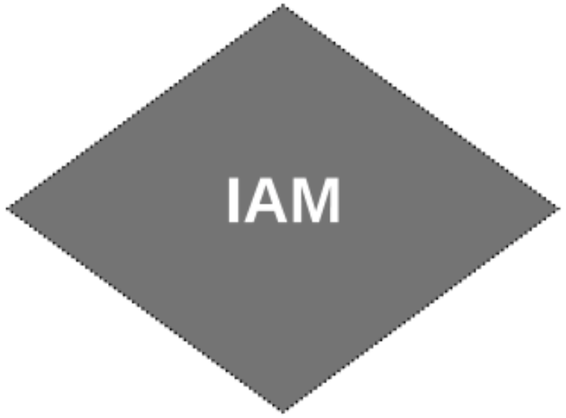
Why care about permissions?

Permission Allowed!

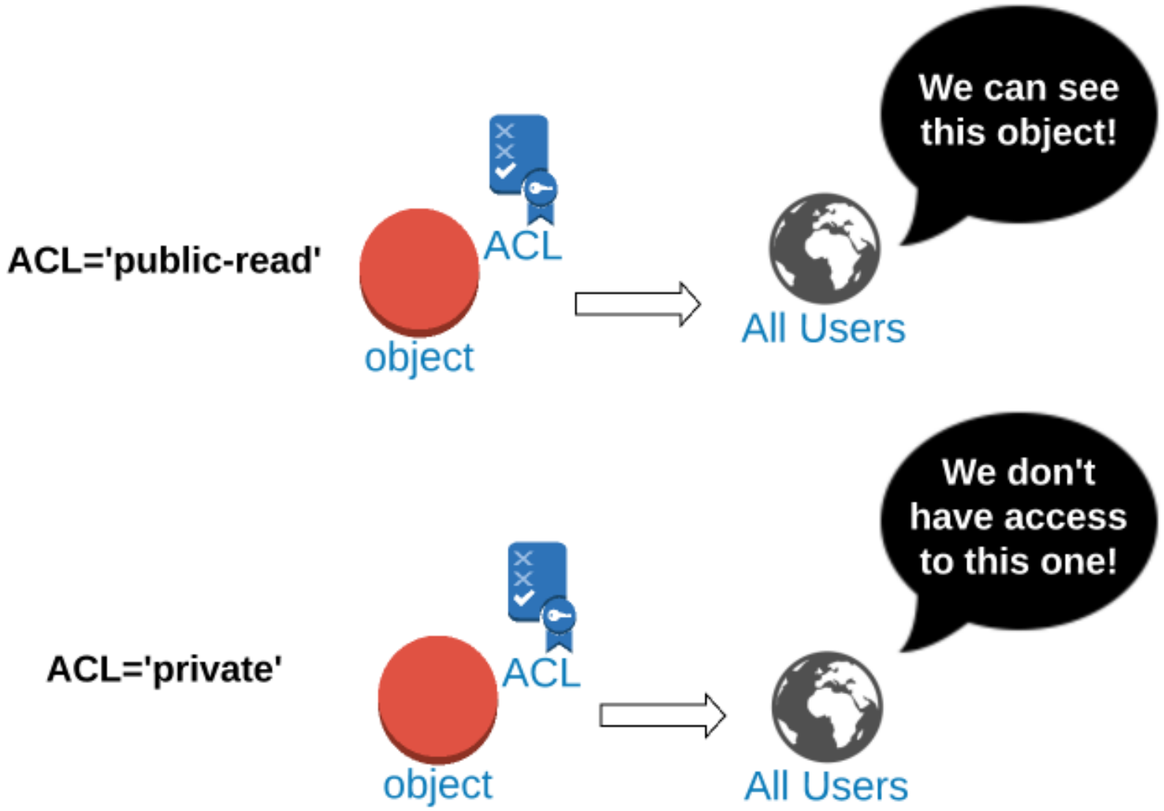
```
# Generate the boto3 client for interacting with S3
s3 = boto3.client('s3', region_name='us-east-1',
                        aws_access_key_id=AWS_KEY_ID,
                        aws_secret_access_key=AWS_SECRET)
# Use client to download a file
s3.download_file(
  Filename='potholes.csv',
  Bucket='gid-requests',
  Key='potholes.csv')
```

AWS Permissions Systems

AWS Permissions Systems



ACLs


ACL='public-read'
ACL
object
We can see this object!
All Users
ACL='private'
ACL
object
We don't have access to this one!
All Users

ACLs

Upload File

```
s3.upload_file(
  Filename='potholes.csv', Bucket='gid-requests', Key='potholes.csv')
```

Set ACL to `'public-read'`

```
s3.put_object_acl(
  Bucket='gid-requests', Key='potholes.csv', ACL='public-read')
```

Setting ACLs on upload

Upload file with `'public-read'` ACL

```
s3.upload_file(
  Bucket='gid-requests',
  Filename='potholes.csv',
  Key='potholes.csv',
  ExtraArgs={'ACL':'public-read'})
```

Accessing public objects

S3 Object URL Template

```
https://{bucket}.s3.amazonaws.com/{key}
```

URL for Key= `'2019/potholes.csv'`

```
https://gid-requests.s3.amazonaws.com/2019/potholes.csv
```

Generating public object URL

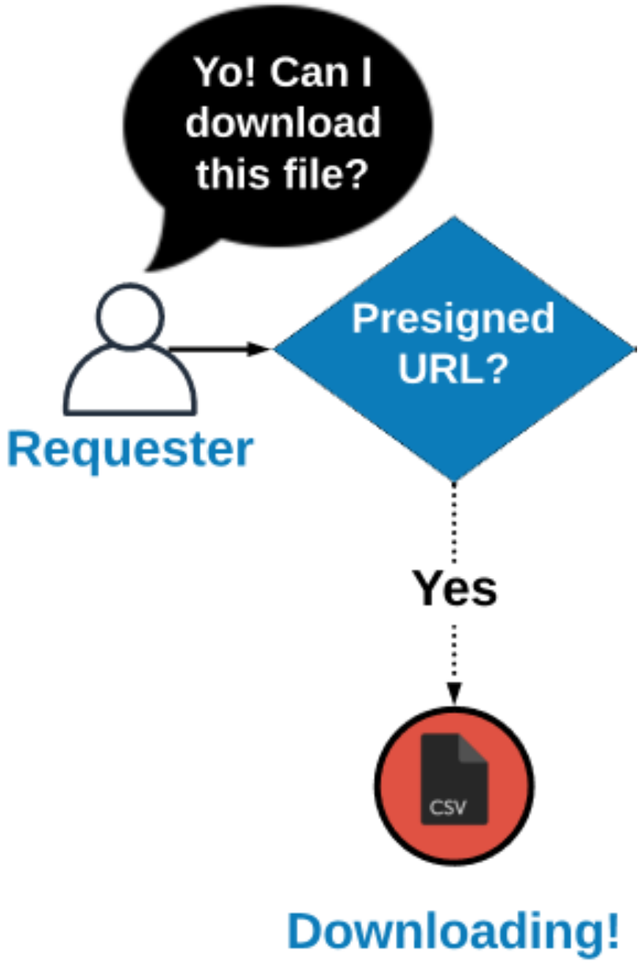
Generate Object URL String

```
url = "https://{}.s3.amazonaws.com/{}".format(
  "gid-requests",
  "2019/potholes.csv")
```

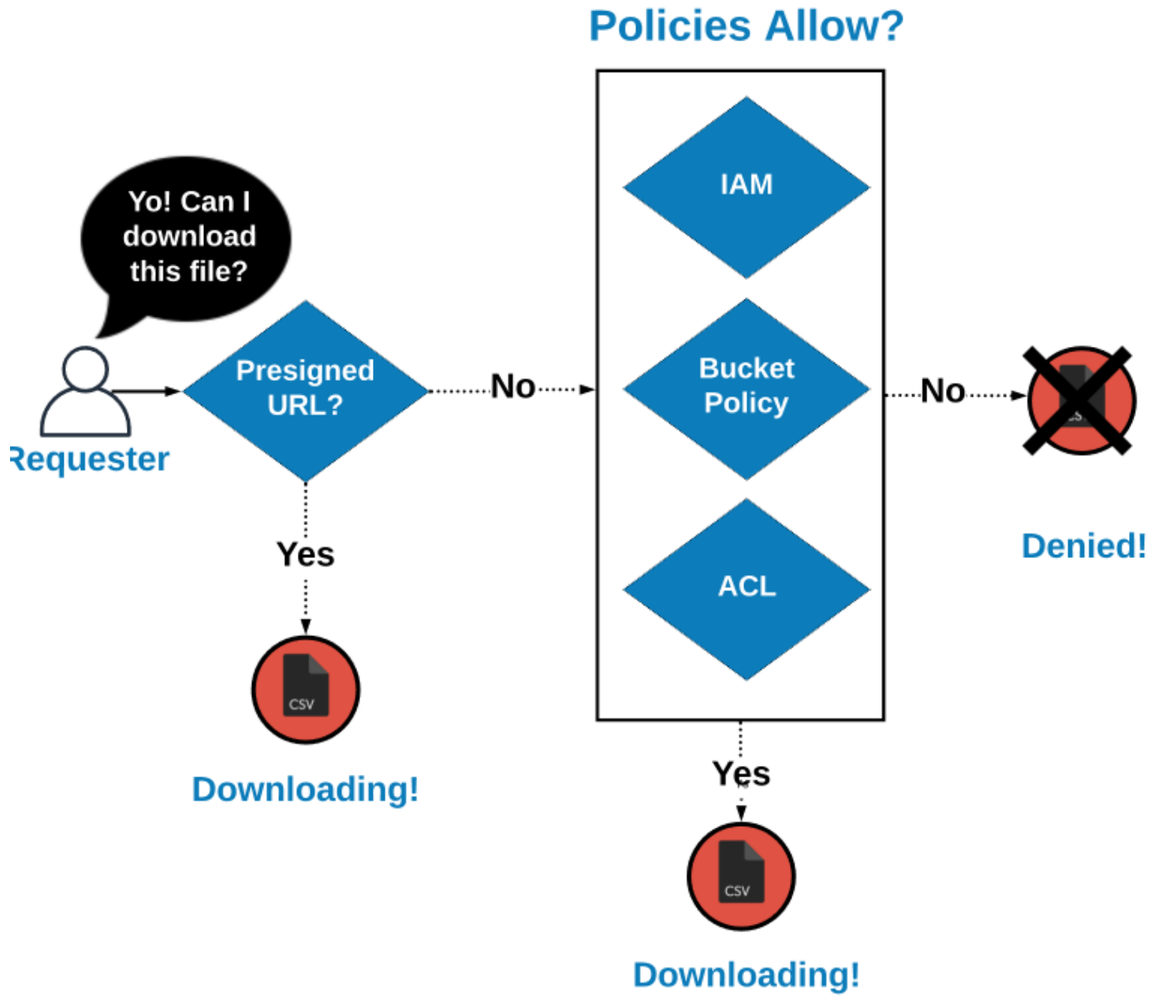
```
'https://gid-requests.s3.amazonaws.com/2019/potholes.csv'
```

```
# Read the URL into Pandas
df = pd.read_csv(url)
```

How access is decided



How access is decided



Review

Review

Set ACL to `'public-read'`

```
s3.put_object_acl(
  Bucket='gid-requests', Key='potholes.csv', ACL='public-read')
```

Set ACL to `'private'`

```
s3.put_object_acl(
  Bucket='gid-requests', Key='potholes.csv', ACL='private')
```

Review

Upload file with `'public-read'` ACL

```
s3.upload_file(
  Bucket='gid-requests',
  Filename='potholes.csv',
  Key='potholes2.csv',
  ExtraArgs={'ACL':'public-read'})
```

Review

Generate Object URL String

```
url = "https://{}.s3.amazonaws.com/{}".format(
  "gid-requests",
  "2019/potholes.csv")
```

```
'https://gid-requests.s3.amazonaws.com/2019/potholes.csv'
```

```
# Read the URL into Pandas
df = pd.read_csv(url)
```

Let's practice!

INTRODUCTION TO AWS BOTO IN PYTHON

Accessing private objects in S3

INTRODUCTION TO AWS BOTO IN PYTHON

Maksim Pecherskiy
Data Engineer

datacamp

Downloading a private file

```
df = pd.read_csv('https://gid-staging.s3.amazonaws.com/potholes.csv')
```

```
/usr/local/Cellar/python/3.7.1/Frameworks/Python.framework/Versions/3.7/lib/pyth
    647 class HTTPDefaultErrorHandler(BaseHandler):
    648     def http_error_default(self, req, fp, code, msg, hdrs):
--> 649         raise HTTPError(req.full_url, code, msg, hdrs, fp)
    650
    651 class HTTPRedirectHandler(BaseHandler):

HTTPError: HTTP Error 403: Forbidden
```

Downloading private files

Download File

```
s3.download_file(
  Filename='potholes_local.csv',
  Bucket='gid-staging',
  Key='2019/potholes_private.csv')
```

Read From Disk

```
pd.read_csv('./potholes_local.csv')
```

Accessing private files

Use `'.get_object()'`

```
obj = s3.get_object(Bucket='gid-requests', Key='2019/potholes.csv')
print(obj)
```

Accessing private files

```
{'ResponseMetadata': {'RequestId': '5B9B1FA6E703AB51',
  'HTTPStatusCode': 200,
  'RetryAttempts': 0},
 'AcceptRanges': 'bytes',
 'LastModified': datetime.datetime(2019, 5, 11, 23, 55, 43, tzinfo=tzutc()),
 'ContentLength': 2012850,
 'ETag': '"16966a303c7893b43bc7c04e76b020f9"',
 'ContentType': 'binary/octet-stream',
 'Metadata': {},
 'Body': <botocore.response.StreamingBody at 0x11f392b38>}
```

Accessing private Files

Get the object

```
obj = s3.get_object(
  Bucket='gid-requests',
  Key='2019/potholes.csv')
```

Read `StreamingBody` into Pandas

```
pd.read_csv(obj['Body'])
```

Pre-signed URLs

- Expire after a certain timeframe
- Great for temporary access

Example

```
https://s3.amazonaws.com/?AWSAccessKeyId=12345&Signature=rBmnrwutb6VkJ9hE8Uub%2BBYA9mY%3D&Expires=1557624801
```

Pre-signed URLs

Upload a file

```
s3.upload_file(
  Filename='./potholes.csv',
  Key='potholes.csv',
  Bucket='gid-requests')
```

Pre-signed URLs

Generate Presigned URL

```
share_url = s3.generate_presigned_url(
  ClientMethod='get_object',
  ExpiresIn=3600,
  Params={'Bucket': 'gid-requests','Key': 'potholes.csv'}
)
```

Open in Pandas

```
pd.read_csv(share_url)
```

Load multiple files into one DataFrame

```
# Create list to hold our DataFrames
df_list = []

# Request the list of csv's from S3 with prefix; Get contents
response = s3.list_objects(
  Bucket='gid-requests',
  Prefix='2019/')

# Get response contents
request_files = response['Contents']
```

Load multiple files into one DataFrame

```
# Iterate over each object
for file in request_files:
    obj = s3.get_object(Bucket='gid-requests', Key=file['Key'])

    # Read it as DataFrame
    obj_df = pd.read_csv(obj['Body'])

    # Append DataFrame to list
    df_list.append(obj_df)
```

Load multiple files into one DataFrame

```
# Concatenate all the DataFrames in the list
df = pd.concat(df_list)

# Preview the DataFrame
df.head()
```

	service_request_id	service_request_parent_id	sap_notification_number	requested_datetime	case_age_days	service_name	case_record_type	updated_datetime	status	l
0	2553572	NaN	NaN	2019-04-03T08:58:00	0.0	72 Hour Violation	Parking	NaN	New	32.8308
1	2553573	NaN	NaN	2019-04-03T08:58:00	0.0	Graffiti Removal	TSW	2019-04-03T00:00:00	Closed	32.7550
2	2553570	NaN	NaN	2019-04-03T08:55:00	0.0	Missed Collection	ESD Complaint/Report	NaN	In Process	32.7789
3	2553568	2538156.0	NaN	2019-04-03T08:54:00	0.0	Street Light Out	TSW	NaN	In Process	32.8212
4	2553565	NaN	NaN	2019-04-03T08:53:00	0.0	Graffiti Removal	TSW	NaN	In Process	32.7147

Review Accessing private objects in S3

Download then open

```
s3.download_file()
```

Open directly

```
s3.get_object()
```

Generate presigned URL

```
s3.generate_presigned_url()
```

Review - Sharing URLs

PUBLIC FILES: PUBLIC OBJECT URL

Generate using `.format()`

```
'https://{bucket}.s3.amazonaws.com/{key}'
```

PRIVATE FILES: PRESIGNED URL

Generate using `.get_presigned_url()`

```
'https://s3.amazonaws.com/?AWSAccessKeyId=12345&Signature=rBmnrwutb6VkJ9hE8Uub%2BBYA
```

Let's practice!

INTRODUCTION TO AWS BOTO IN PYTHON

Sharing files through a website

INTRODUCTION TO AWS BOTO IN PYTHON

Maksim Pecherskiy
Data Engineer

datacamp

Serving HTML Pages

Link	LastModified	Size
http://gid-reports.s3-website.us-east-1.amazonaws.com/2019/jan/final_chart.html	2019-05-13 01:11:56+00:00	6759
http://gid-reports.s3-website.us-east-1.amazonaws.com/2019/jan/final_report.csv	2019-05-13 01:11:55+00:00	138
http://gid-reports.s3-website.us-east-1.amazonaws.com/2019/jan/final_report.html	2019-05-13 01:11:55+00:00	536

HTML table in Pandas

Convert DataFrame to html

```
df.to_html('table_agg.html')
```

	service_name	request_count	info_link	description
0	72 Hour Violation	8	https://www.sandiego.gov/get-it-done	A Description placeholder
1	Graffiti Removal	2	https://www.sandiego.gov/get-it-done	A Description placeholder
2	Missed Collection	12	https://www.sandiego.gov/get-it-done	A Description placeholder
3	Street Light Out	21	https://www.sandiego.gov/get-it-done	A Description placeholder
4	Pothole	33	https://www.sandiego.gov/get-it-done	A Description placeholder
5	Parking Zone Violation	44	https://www.sandiego.gov/get-it-done	A Description placeholder
6	Oversized Vehicle Complaints	2	https://www.sandiego.gov/get-it-done	A Description placeholder
7	Sidewalk Repair Issue	1	https://www.sandiego.gov/get-it-done	A Description placeholder

HTML Table in Pandas with links

Convert DataFrame to html

```
df.to_html('table_agg.html', render_links=True)
```

	service_name	request_count	info_link	description
0	72 Hour Violation	8	https://www.sandiego.gov/get-it-done	A Description placeholder
1	Graffiti Removal	2	https://www.sandiego.gov/get-it-done	A Description placeholder
2	Missed Collection	12	https://www.sandiego.gov/get-it-done	A Description placeholder
3	Street Light Out	21	https://www.sandiego.gov/get-it-done	A Description placeholder
4	Pothole	33	https://www.sandiego.gov/get-it-done	A Description placeholder
5	Parking Zone Violation	44	https://www.sandiego.gov/get-it-done	A Description placeholder
6	Oversized Vehicle Complaints	2	https://www.sandiego.gov/get-it-done	A Description placeholder
7	Sidewalk Repair Issue	1	https://www.sandiego.gov/get-it-done	A Description placeholder

Certain columns to HTML

Convert DataFrame to html

```
df.to_html('table_agg.html',
           render_links=True,
           columns['service_name', 'request_count', 'info_link'])
```

	service_name	request_count	info_link
0	72 Hour Violation	8	https://www.sandiego.gov/get-it-done
1	Graffiti Removal	2	https://www.sandiego.gov/get-it-done
2	Missed Collection	12	https://www.sandiego.gov/get-it-done
3	Street Light Out	21	https://www.sandiego.gov/get-it-done
4	Pothole	33	https://www.sandiego.gov/get-it-done
5	Parking Zone Violation	44	https://www.sandiego.gov/get-it-done
6	Oversized Vehicle Complaints	2	https://www.sandiego.gov/get-it-done
7	Sidewalk Repair Issue	1	https://www.sandiego.gov/get-it-done

Borders

Convert DataFrame to html

```
df.to_html('table_agg.html',
           render_links=True,
           columns['service_name', 'request_count', 'info_link'],
           border=0)
```

	service_name	request_count	info_link
0	72 Hour Violation	8	https://www.sandiego.gov/get-it-done
1	Graffiti Removal	2	https://www.sandiego.gov/get-it-done
2	Missed Collection	12	https://www.sandiego.gov/get-it-done
3	Street Light Out	21	https://www.sandiego.gov/get-it-done
4	Pothole	33	https://www.sandiego.gov/get-it-done
5	Parking Zone Violation	44	https://www.sandiego.gov/get-it-done
6	Oversized Vehicle Complaints	2	https://www.sandiego.gov/get-it-done
7	Sidewalk Repair Issue	1	https://www.sandiego.gov/get-it-done

Uploading an HTML file to S3

Upload an HTML file to S3

```
s3.upload_file(
  Filename='./table_agg.html',
  Bucket='datacamp-website',
  Key='table.html',
  ExtraArgs = {
    'ContentType': 'text/html',
    'ACL': 'public-read'}
)
```

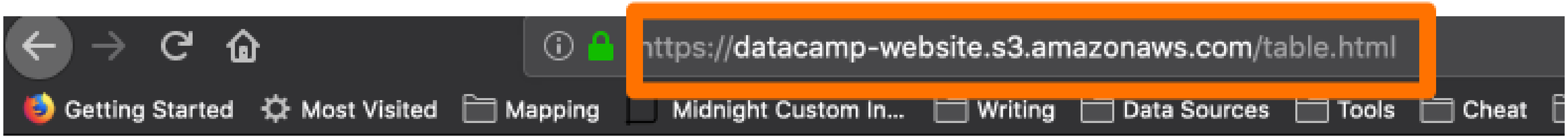
Accessing HTML file

S3 Object URL Template

```
https://{bucket}.s3.amazonaws.com/{key}
```

```
https://datacamp-website.s3.amazonaws.com/table.html
```

HTML Page



	service_name	request_count	info_link
0	72 Hour Violation	8	https://www.sandiego.gov/get-it-done
1	Graffiti Removal	2	https://www.sandiego.gov/get-it-done
2	Missed Collection	12	https://www.sandiego.gov/get-it-done
3	Street Light Out	21	https://www.sandiego.gov/get-it-done
4	Pothole	33	https://www.sandiego.gov/get-it-done
5	Parking Zone Violation	44	https://www.sandiego.gov/get-it-done
6	Oversized Vehicle Complaints	2	https://www.sandiego.gov/get-it-done
7	Sidewalk Repair Issue	1	https://www.sandiego.gov/get-it-done

Uploading other types of content

Upload an image file to S3

```
s3.upload_file(
  Filename='./plot_image.png',
  Bucket='datacamp-website',
  Key='plot_image.png',
  ExtraArgs = {
    'ContentType': 'image/png',
    'ACL': 'public-read'}
)
```

IANA Media Types

- JSON : `application/json`
- PNG : `image/png`
- PDF : `application/pdf`
- CSV : `text/csv`

[1] http://www.iana.org/assignments/media [2] types/media [3] types.xhtml

Generating an index page

```
# List the gid-reports bucket objects starting with 2019/
r = s3.list_objects(Bucket='gid-reports', Prefix='2019/')

# Convert the response contents to DataFrame
objects_df = pd.DataFrame(r['Contents'])
```

ETag	Key	LastModified	Owner	Size	Storage
"9a682c7e6fd151d18912c319b3fac8dc"	2019/jan/final_chart.html	2019-05-13 01:11:56+00:00	{'DisplayName': 'maksim+aws-demos', 'ID': '12346cf1b2f0e923b64d624ce166bb570c6dae4a2a905b419916bd365ea5a596'}	6759	STAN
"bddc4af094a7cdad783b8829479058d6"	2019/jan/final_report.csv	2019-05-13 01:11:55+00:00	{'DisplayName': 'maksim+aws-demos', 'ID': '12346cf1b2f0e923b64d624ce166bb570c6dae4a2a905b419916bd365ea5a596'}	138	STAN
"03baa6b325d75dff02ef83af39a8205f"	2019/jan/final_report.html	2019-05-13 01:11:55+00:00	{'DisplayName': 'maksim+aws-demos', 'ID': '12346cf1b2f0e923b64d624ce166bb570c6dae4a2a905b419916bd365ea5a596'}	536	STAN

Generating an index page

```
# Create a column "Link" that contains website url + key
base_url = "http://datacamp-website.s3.amazonaws.com/"
objects_df['Link'] = base_url + objects_df['Key']
```

```
# Write DataFrame to html
objects_df.to_html('report_listing.html',
                   columns=['Link', 'LastModified', 'Size'],
                   render_links=True)
```

	Link	LastModified	Size
0	http://datacamp-website.s3.amazonaws.com/index.html	2019-05-12 16:41:57+00:00	906
1	http://datacamp-website.s3.amazonaws.com/table.html	2019-05-19 20:07:07+00:00	1910
2	http://datacamp-website.s3.amazonaws.com/table_col_limit.html	2019-05-19 20:12:09+00:00	1883
3	http://datacamp-website.s3.amazonaws.com/table_no_border.html	2019-05-19 20:12:09+00:00	1883

Uploading index page

Upload an HTML file to S3

```
s3.upload_file(
  Filename='./report_listing.html',
  Bucket='datacamp-website',
  Key='index.html',
  ExtraArgs = {
    'ContentType': 'text/html',
    'ACL': 'public-read'}
)
```

https://datacamp-website.s3.amazonaws.com/index.html

Review

- HTML Table in Pandas (`df.to_html('table.html')`)
- Upload HTML file (`ContentType: text/html`)
- Upload Image file (`ContentType: image/png`)
- Share the URL for our html page!

Let's practice!

INTRODUCTION TO AWS BOTO IN PYTHON

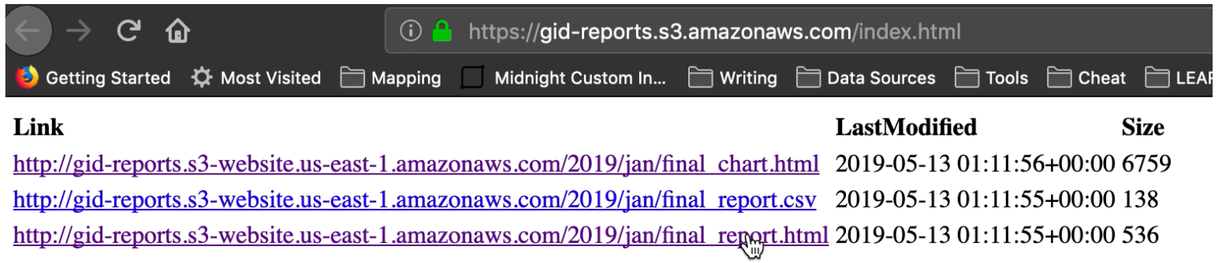
Case Study: Generating a Report Repository

INTRODUCTION TO AWS BOTO IN PYTHON

Maksim Pecherskiy
Data Engineer

datacamp

Final product



The steps

Prepare the data

- Download files for the month from the raw data bucket
- Concatenate them into one csv
- Create an aggregated DataFrame

The steps

Create the report

- Write the DataFrame to CSV and HTML
- Generate a Bokeh plot, save as HTML

The steps

Upload report to shareable website

- Create `gid-reports` bucket
- Upload all the three files for the month to S3
- Generate an index.html file that lists all the files
- Get the website URL!

Raw data bucket

- Private files
- Daily CSVs of requests from the App
- Raw data

Name
2019_jan_10_gid_requests
2019_jan_11_gid_requests
2019_jan_12_gid_requests
2019_jan_13_gid_requests
2019_jan_14_gid_requests

Read raw data files

```
# Create list to hold our DataFrames
df_list = []

# Request the list of csv's from S3 with prefix; Get contents
response = s3.list_objects(
  Bucket='gid-requests',
  Prefix='2019_jan')

# Get response contents
request_files = response['Contents']
```

Read raw data files

```
# Iterate over each object
for file in request_files:
    obj = s3.get_object(Bucket='gid-requests', Key=file['Key'])

    # Read it as DataFrame
    obj_df = pd.read_csv(obj['Body'])

    # Append DataFrame to list
    df_list.append(obj_df)
```

Read raw data files

```
# Concatenate all the DataFrames in the list
df = pd.concat(df_list)

# Preview the DataFrame
df.head()
```

	service_request_id	service_request_parent_id	sap_notification_number	requested_datetime	case_age_days	service_name	case_record_type	updated_datetime	status	l
0	2553572	NaN	NaN	2019-04-03T08:58:00	0.0	72 Hour Violation	Parking	NaN	New	32.8308
1	2553573	NaN	NaN	2019-04-03T08:58:00	0.0	Graffiti Removal	TSW	2019-04-03T00:00:00	Closed	32.7550
2	2553570	NaN	NaN	2019-04-03T08:55:00	0.0	Missed Collection	ESD Complaint/Report	NaN	In Process	32.7789
3	2553568	2538156.0	NaN	2019-04-03T08:54:00	0.0	Street Light Out	TSW	NaN	In Process	32.8212
4	2553565	NaN	NaN	2019-04-03T08:53:00	0.0	Graffiti Removal	TSW	NaN	In Process	32.7147

Create aggregated reports

- Perform some aggregation
- `df.to_csv('jan_final_report.csv')`
- `df.to_html('jan_final_report.html')`
- `jan_final_chart.html`

	case_record_type	count
0	ESD Complaint/Report	4770
1	Parking	3240
2	Storm Water Code Enforcement	210
3	TSW	6690
4	Traffic Engineering	60

Report bucket

- Bucket website
- Publicly Accessible
- Aggregated data and HTML reports

Upload Aggregated CSV

```
# Upload Aggregated CSV to S3
s3.upload_file(Filename='./jan_final_report.csv',
                        Key='2019/jan/final_report.csv',
                        Bucket='gid-reports',
                        ExtraArgs = {'ACL': 'public-read'})
```

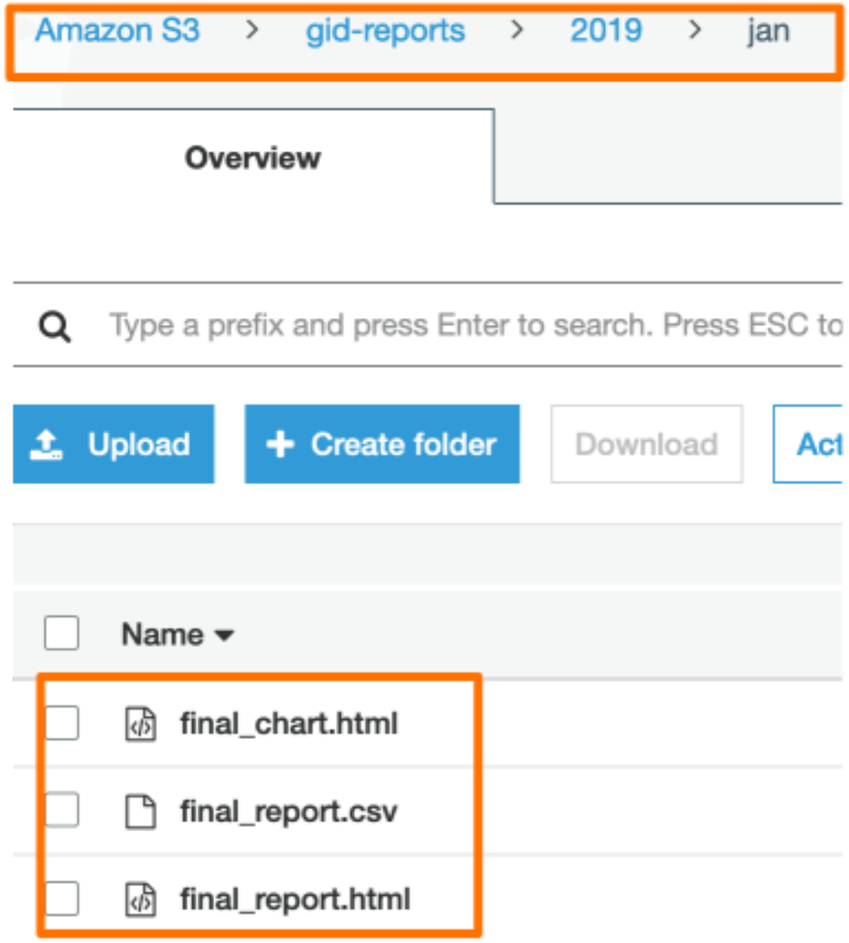
Upload HTML Table

```
# Upload HTML table to S3
s3.upload_file(Filename='./jan_final_report.html',
                        Key='2019/jan/final_report.html',
                        Bucket='gid-reports',
                        ExtraArgs = {
                          'ContentType': 'text/html',
                          'ACL': 'public-read'})
```

Upload HTML Chart

```
# Upload Aggregated Chart to S3
s3.upload_file(Filename='./jan_final_chart.html',
                        Key='2019/jan/final_chart.html',
                        Bucket='gid-reports',
                        ExtraArgs = {
                          'ContentType': 'text/html',
                          'ACL': 'public-read'})
```

Uploaded reports



Create index.html

```
# List the gid-reports bucket objects starting with 2019/
r = s3.list_objects(Bucket='gid-reports', Prefix='2019/')

# Convert the response contents to DataFrame
objects_df = pd.DataFrame(r['Contents'])

# Create a column "Link" that contains website url + key
base_url = "https://gid-reports.s3.amazonaws.com/"
objects_df['Link'] = base_url + objects_df['Key']
```

Create index.html

```
# Write DataFrame to html
objects_df.to_html('report_listing.html',
                   columns=['Link', 'LastModified', 'Size'],
                   render_links=True)
```

Link	LastModified	Size
http://gid-reports.s3-website.us-east-1.amazonaws.com/2019/jan/final_chart.html	2019-05-13 01:11:56+00:00	6759
http://gid-reports.s3-website.us-east-1.amazonaws.com/2019/jan/final_report.csv	2019-05-13 01:11:55+00:00	138
http://gid-reports.s3-website.us-east-1.amazonaws.com/2019/jan/final_report.html	2019-05-13 01:11:55+00:00	536

Upload index.html

```
# Upload the file to gid-reports bucket root.
s3.upload_file(
  Filename='./report_listing.html',
  Key='index.html',
  Bucket='gid-reports',
  ExtraArgs = {
    'ContentType': 'text/html',
    'ACL': 'public-read'
  })
```

Get the URL of the index!

Bucket website URL *

```
"http://gid-reports.s3.amazonaws.com/index.html"
```

Let's tweak!

INTRODUCTION TO AWS BOTO IN PYTHON