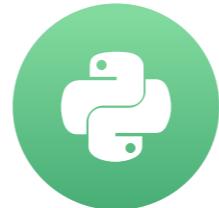


# Intro to AWS and Boto3

INTRODUCTION TO AWS BOTO IN PYTHON



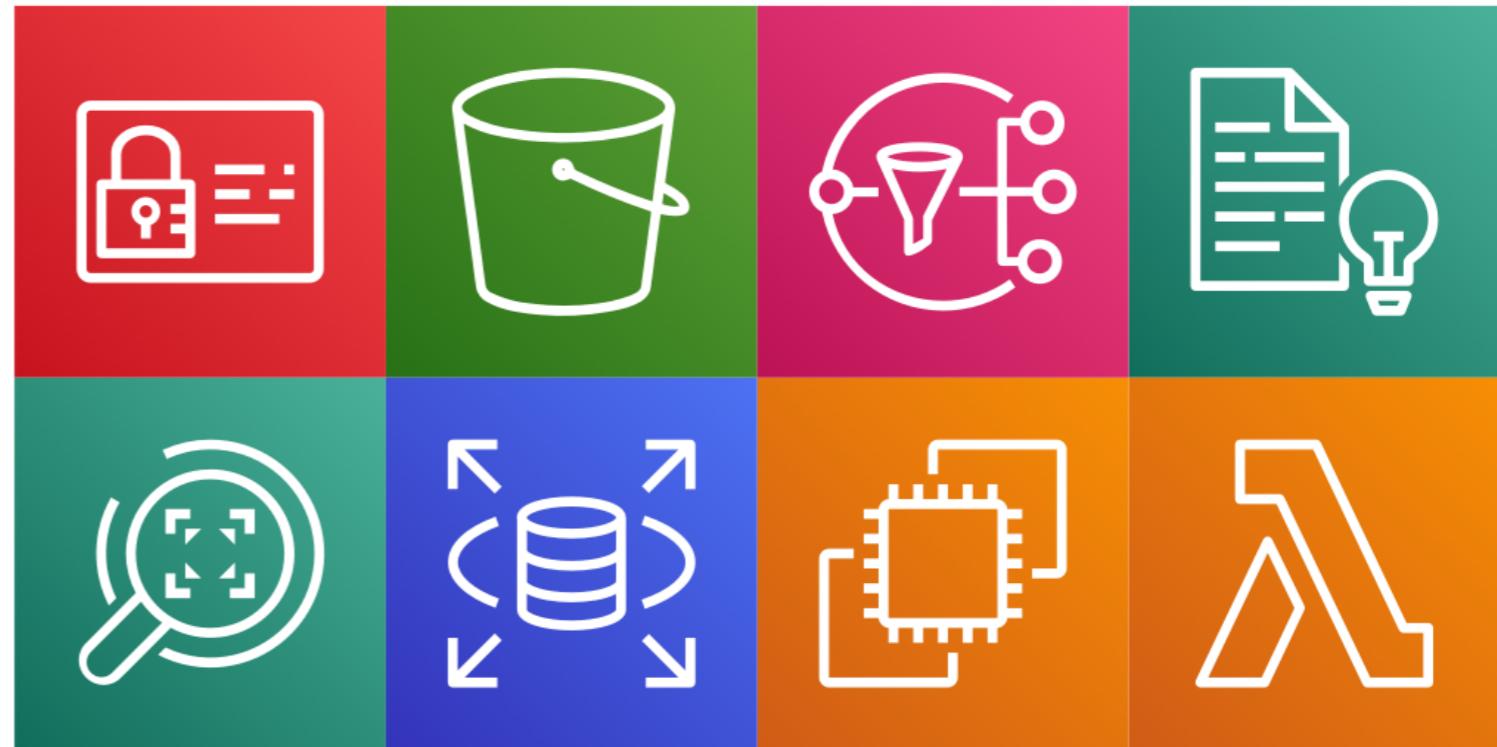
Maksim Pecherskiy  
Data Engineer

# What is Amazon Web Services?

Our Home



Our Data Project



# What is Boto3?

```
import boto3
```

```
s3 = boto3.client('s3',  
                  region_name='us-east-1',  
                  aws_access_key_id=AWS_KEY_ID,  
                  aws_secret_access_key=AWS_SECRET)
```

```
response = s3.list_buckets()
```

# AWS console

AWS Management Console

**AWS services**

**Find Services**  
You can enter names, keywords or acronyms.

Example: Relational Database Service, database, RDS

▼ Recently visited services

AWS Cost Explorer    Systems Manager    AWS Cloud Map

Billing    EC2

► All services

**Build a solution**  
Get started with simple wizards and automated workflows.

**Launch a virtual machine**  
With EC2

**Build a web app**  
With Elastic Beanstalk

**Build using virtual servers**  
With Lightsail

**Access resources on the go**

Access the Management Console using the AWS Console Mobile App. [Learn more](#)

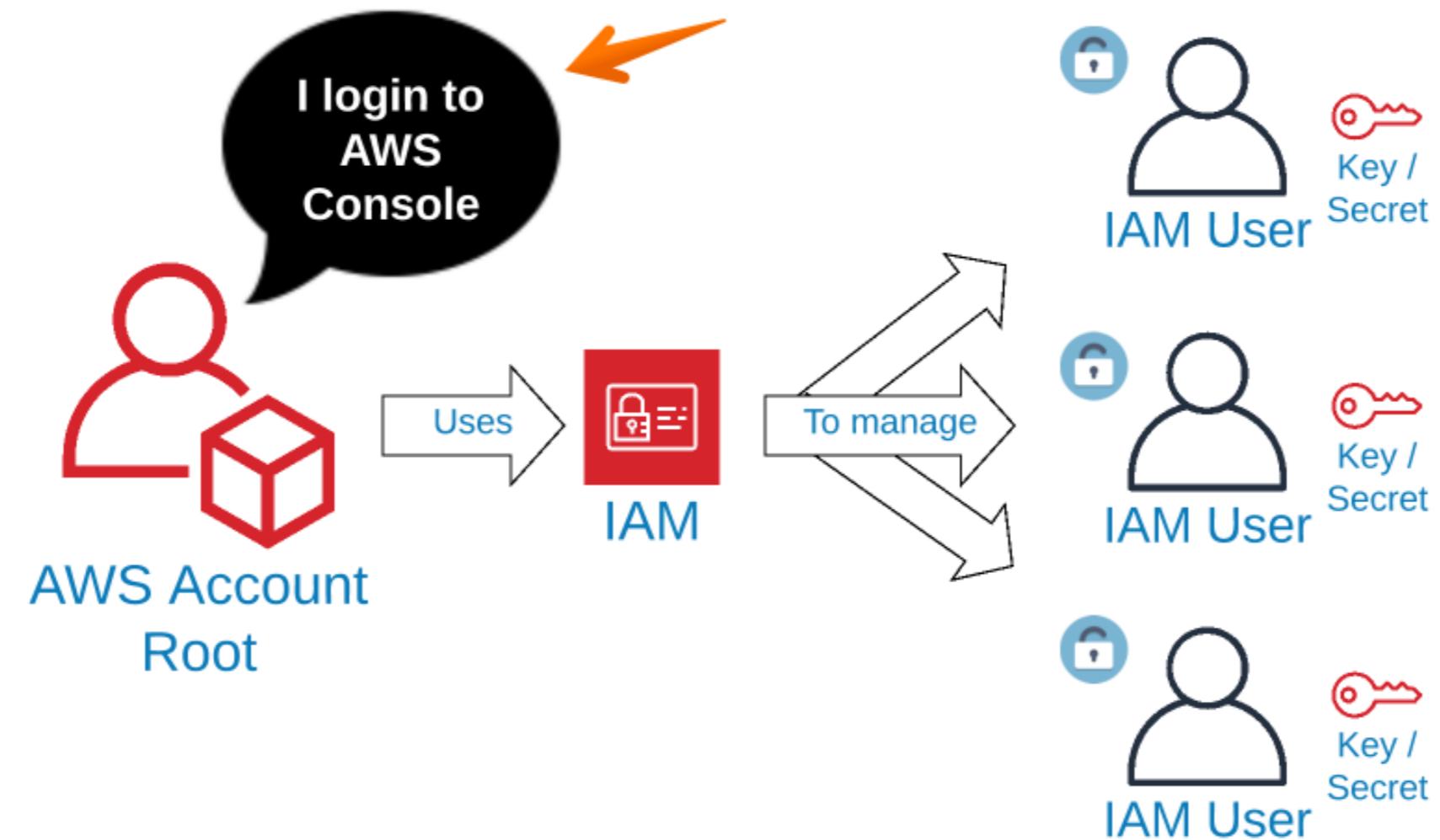
**Explore AWS**

**Visit AWS around the world at a Summit**  
AWS Global Summits bring the cloud computing community together to connect, collaborate, and learn about AWS. [Learn more](#)

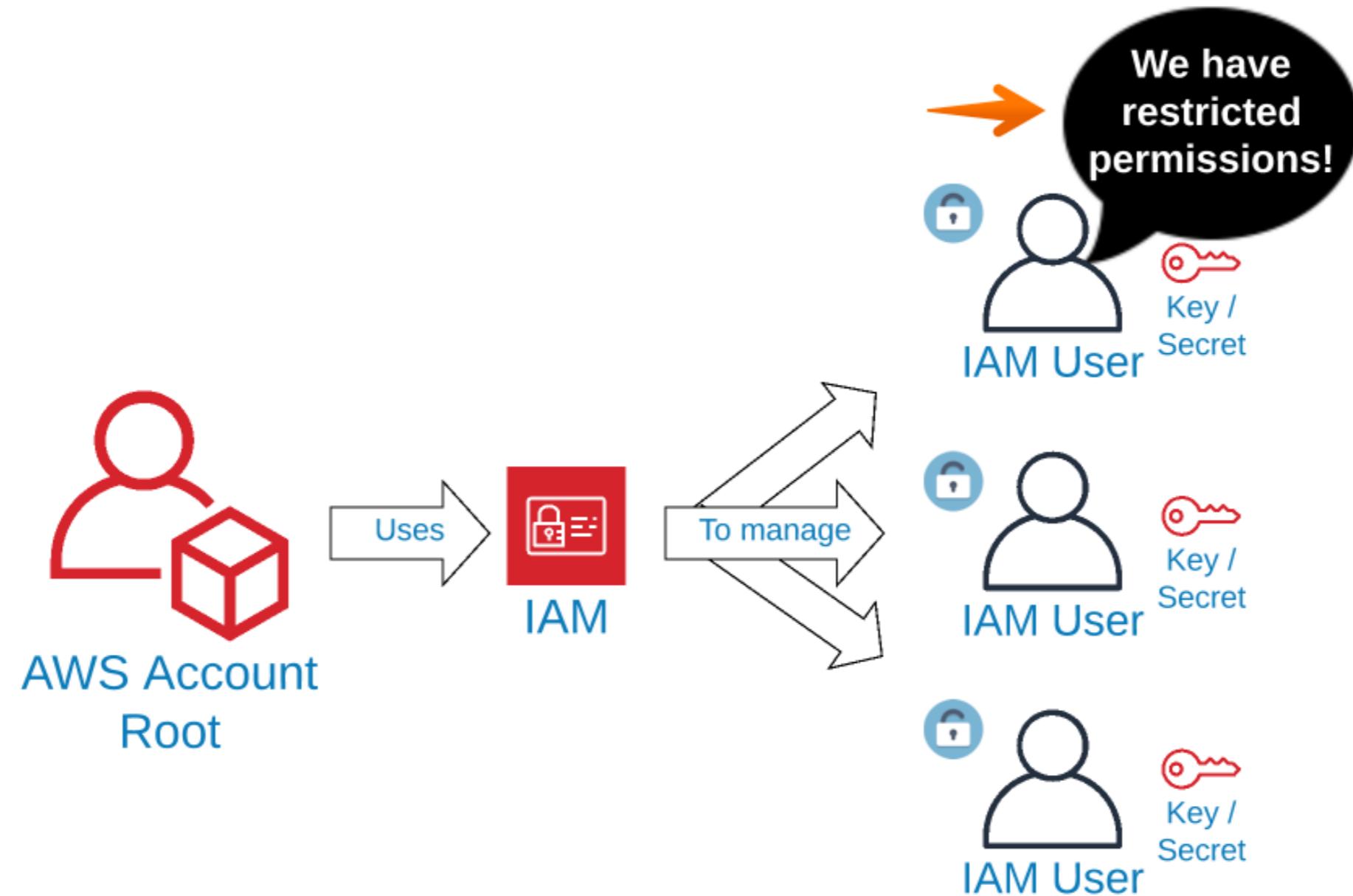
**Amazon SageMaker**  
Machine learning for every developer and data scientist. [Learn more](#)

**Amazon RDS**  
Setup, manage, and scale your relational databases

# Creating keys with IAM.



# Creating keys with IAM



# AWS Management Console

## AWS services

### Find Services

You can enter names, keywords or acronyms.

 Example: Relational Database Service, database, RDS

### ▼ Recently visited services



IAM

### ► All services

## Build a solution

Get started with simple wizards and automated workflows.

### [Launch a virtual machine](#)

With EC2

2-3 minutes

### [Build a web app](#)

With Elastic Beanstalk

6 minutes

### [Build using virtual servers](#)

With Lightsail

1-2 minutes

## Access resources on the go



Access the Management Console using the AWS Console Mobile App. [Learn more](#)

## Explore AWS

### Amazon RDS

Set up, operate, and scale your relational database in the cloud. [Learn more](#)

### Run Serverless Containers with AWS Fargate

AWS Fargate runs and scales your containers without having to manage servers or clusters. [Learn more](#)

### Amazon SageMaker

Machine learning for every developer and data scientist. [Learn more](#)



Services ▾

Resource Groups ▾



Search IAM

Dashboard

Groups

Users

Roles

Policies

Identity providers

Account settings

Credential report

Encryption keys

## Welcome to Identity and Access Management

IAM users sign-in link:

<https://320333787981.signin.aws.amazon.com/console>

| Customize

### IAM Resources

Users: 2

Roles: 2

Groups: 0

Identity Providers: 0

Customer Managed Policies: 0

### Security Status

2 out of 5 complete.

Delete your root access keys

Activate MFA on your root account

Create individual IAM users

Use groups to assign permissions

Apply an IAM password policy



Services ▾

Resource Groups ▾



Search IAM

Add user

Delete user

Dashboard

Groups

Users

Roles

Policies

Identity providers

Account settings

Credential report

Find users by username or access key

User name ▾

 datacampDemoUser

Groups

None

 datacampUser1

None

# Add user

1

2

3

4

5

## Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name\*

This field is required.

 [Add another user](#)

## Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

Access type\*

**Programmatic access**

Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

**AWS Management Console access**

Enables a **password** that allows users to sign-in to the AWS Management Console.

# Add user

1

2

3

4

5

## ▼ Set permissions



Add user to group



Copy permissions from  
existing user



Attach existing policies  
directly 



### Get started with groups

You haven't created any groups yet. Using groups is a best-practice way to manage users' permissions by job functions, AWS service access, or your custom permissions. Get started by creating a group. [Learn more](#)

Create group

## ▶ Set permissions boundary

## User details

User name	datacampDemoUser2
AWS access type	Programmatic access - with an access key
Permissions boundary	Permissions boundary is not set

## Permissions summary

The following policies will be attached to the user shown above.

Type	Name
Managed policy	<a href="#">AmazonS3FullAccess</a>
Managed policy	<a href="#">AmazonSNSFullAccess</a>
Managed policy	<a href="#">AmazonRekognitionFullAccess</a>
Managed policy	<a href="#">ComprehendFullAccess</a>

# Add user

- 1
- 2
- 3
- 4
- 5



## Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://320333787981.signin.aws.amazon.com/console>

IAM User		Key	Secret
User		Access key ID	Secret access key
▶  datacampDemoUser2		AKIAUVFLBWNGYT2JQ7MQ	***** Show

# AWS services



# AWS services



# AWS services



# AWS services



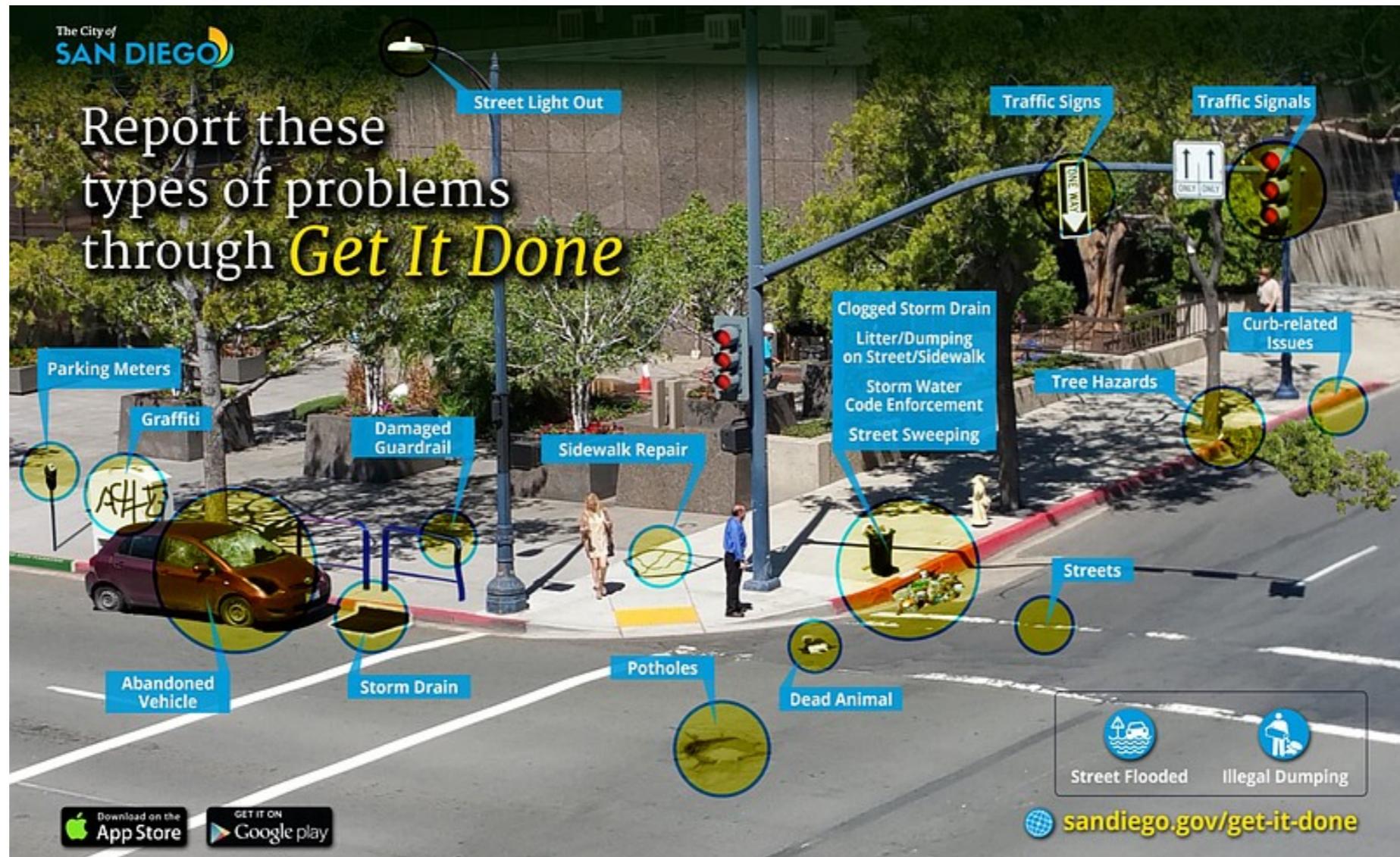
# AWS services



# Sam



# GetItDone



<sup>1</sup> [https://data.san diego.gov/datasets/get\\_it\\_done](https://data.san diego.gov/datasets/get_it_done) <sup>2</sup> [311](#)

# Summary

- AWS Services = Home Utilities
- IAM, S3, SNS, Comprehend and Rekognition
- AWS Key / Secret
- Connecting to S3 Using Boto

```
import boto3
s3 = boto3.client('s3',
                    region_name='us-east-1',
                    aws_access_key_id=AWS_KEY_ID,
                    aws_secret_access_key=AWS_SECRET)
```

```
response = s3.list_buckets()
```

# Let's harness the cloud!

## INTRODUCTION TO AWS BOTO IN PYTHON

# Diving into buckets

INTRODUCTION TO AWS BOTO IN PYTHON



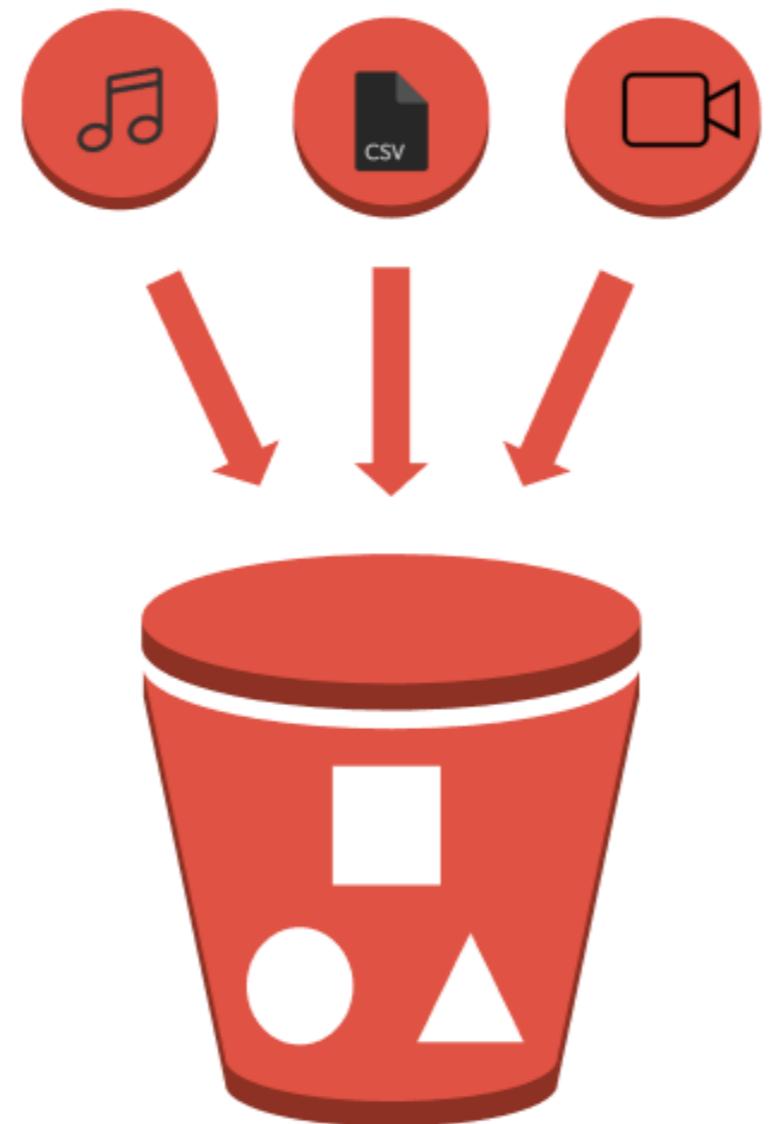
Maksim Pecherskiy  
Data Engineer

# S3 Components - Buckets

- Desktop folders
- Own permission policy
- Website storage
- Generate logs



# S3 Components - Objects



# What can we do with buckets?

- Create Bucket
- List Buckets
- Delete Bucket



# Creating a Bucket

## Create boto3 client

```
import boto3  
s3 = boto3.client('s3', region_name='us-east-1',  
                  aws_access_key_id=AWS_KEY_ID,  
                  aws_secret_access_key=AWS_SECRET)
```

## Create bucket!

```
bucket = s3.create_bucket(Bucket='gid-requests')
```

# Bang!



# Our bucket in the console

S3 buckets

 Discover the console

<input type="text"/> Search for buckets				All access types			
 Create bucket		Edit public access settings	Empty	Delete	6 Buckets	1 Regions	
<input type="checkbox"/>	Bucket name ▾	Access  ▾	Region ▾	Date created ▾			
<input type="checkbox"/>	 datacamp-gid-images	Objects can be public	US East (N. Virginia)	Feb 18, 2019 4:08:14 PM GMT-0800			
<input type="checkbox"/>	 datacamp-gid-images-positive-match	Objects can be public	US East (N. Virginia)	Feb 18, 2019 4:19:51 PM GMT-0800			
<input type="checkbox"/>	 datacamp-incoming	Objects can be public	US East (N. Virginia)	Feb 18, 2019 12:33:44 PM GMT-0800			
<input type="checkbox"/>	 datacamp-public	Public	US East (N. Virginia)	Feb 18, 2019 12:33:19 PM GMT-0800			
<input type="checkbox"/>	 dc-incoming	Objects can be public	US East (N. Virginia)	Feb 18, 2019 12:31:15 PM GMT-0800			
<input type="checkbox"/>	 gid-requests	Objects can be public	US East (N. Virginia)	Apr 9, 2019 10:05:15 PM GMT-0700			

# Listing buckets

## Create boto3 client

```
import boto3  
s3 = boto3.client('s3', region_name='us-east-1',  
                  aws_access_key_id=AWS_KEY_ID,  
                  aws_secret_access_key=AWS_SECRET)
```

## List Buckets

```
bucket_response = s3.list_buckets()
```

# Listing Buckets

## Get Buckets Dictionary

```
buckets = bucket_response[ 'Buckets' ]  
print(buckets)
```

# Listing Buckets

```
[{'Name': 'dc-incoming',
 'CreationDate': datetime.datetime(2019, 2, 18, 20, 31, 15, tzinfo=tzutc())},
 {'Name': 'gid-requests',
 'CreationDate': datetime.datetime(2019, 4, 10, 5, 5, 15, tzinfo=tzutc())}]
```

# Deleting buckets

## Create boto3 client

```
import boto3  
  
s3 = boto3.client('s3', region_name='us-east-1',  
                  aws_access_key_id=AWS_KEY_ID,  
                  aws_secret_access_key=AWS_SECRET)
```

## Delete Bucket

```
response = s3.delete_bucket('gid-requests')
```

# Bye Bye Bucket



# Bye Bye Bucket

S3 buckets

 [Discover the console](#)

<input type="text"/> Search for buckets				All access types
 <a href="#">Create bucket</a>	<a href="#">Edit public access settings</a>	<a href="#">Empty</a>	<a href="#">Delete</a>	5 Buckets 1 Regions 
<input type="checkbox"/> Bucket name ▾	Access  ▾	Region ▾	Date created ▾	
<input type="checkbox"/>  datacamp-gid-images	Objects can be public	US East (N. Virginia)	Feb 18, 2019 4:08:14 PM GMT-0800	
<input type="checkbox"/>  datacamp-gid-images-positive-match	Objects can be public	US East (N. Virginia)	Feb 18, 2019 4:19:51 PM GMT-0800	
<input type="checkbox"/>  datacamp-incoming	Objects can be public	US East (N. Virginia)	Feb 18, 2019 12:33:44 PM GMT-0800	
<input type="checkbox"/>  datacamp-public	Public	US East (N. Virginia)	Feb 18, 2019 12:33:19 PM GMT-0800	
<input type="checkbox"/>  dc-incoming	Objects can be public	US East (N. Virginia)	Feb 18, 2019 12:31:15 PM GMT-0800	

# Other operations

## Client

---

### `class S3.Client`

A low-level client representing Amazon Simple Storage Service (S3):

```
import boto3

client = boto3.client('s3')
```

These are the available methods:

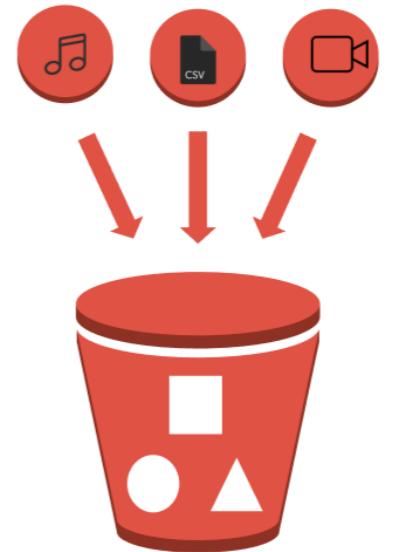
- `abort_multipart_upload()`
- `can_paginate()`
- `complete_multipart_upload()`

# Summary

```
s3.create_bucket(Bucket='buck')
```

```
s3.list_buckets()
```

```
s3.delete_bucket(Bucket='buck')
```

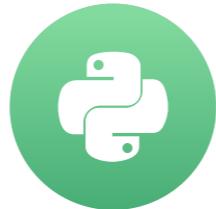


# Let's practice!

INTRODUCTION TO AWS BOTO IN PYTHON

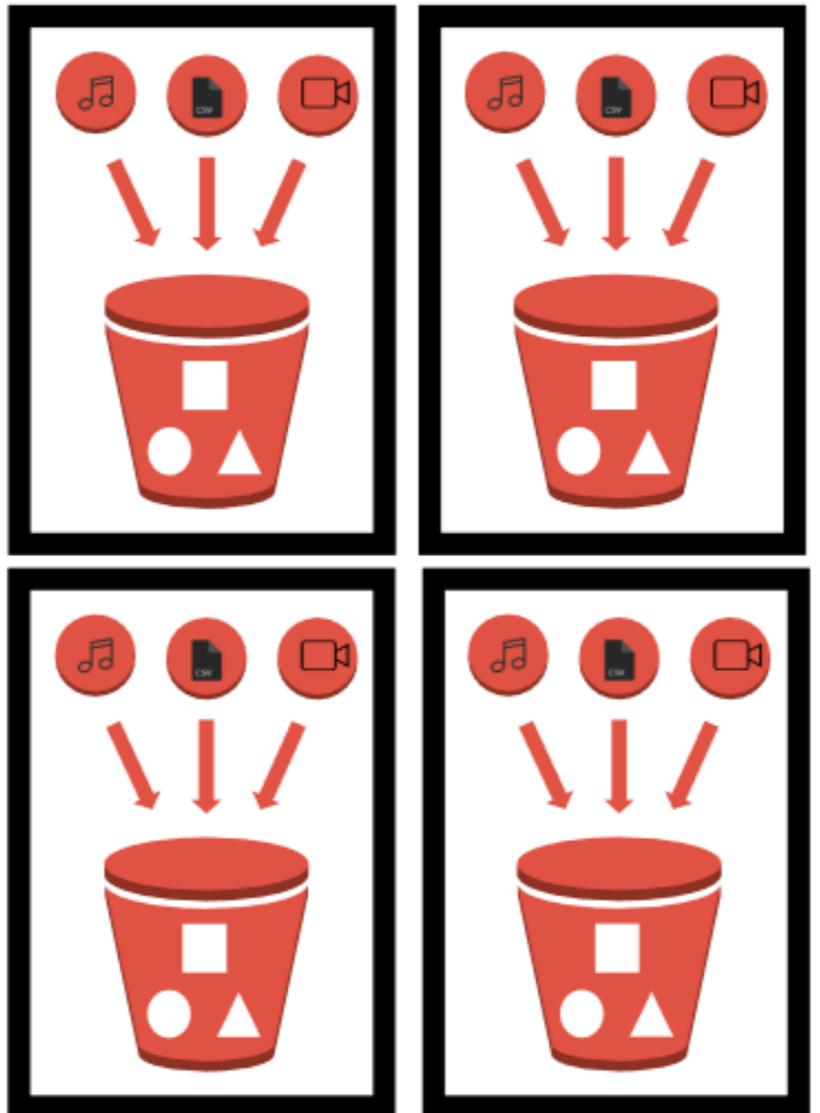
# Uploading and retrieving files

INTRODUCTION TO AWS BOTO IN PYTHON



Maksim Pecherskiy  
Data engineer

# Buckets and objects



# A Bucket



- A bucket has a **name**
- **Name** is a string
- **Unique** name in all of S3.
- Contains **many** objects

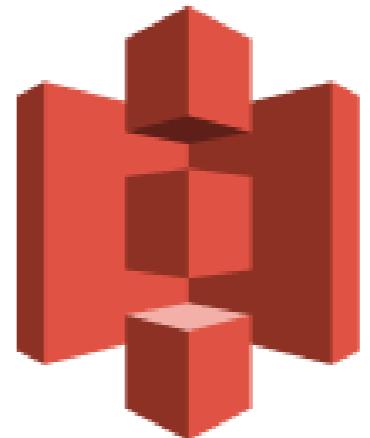
# An Object



- An object has a **key**
- **Name** is full path from bucket root
- **Unique** key in the bucket
- Can only be in **one** parent bucket

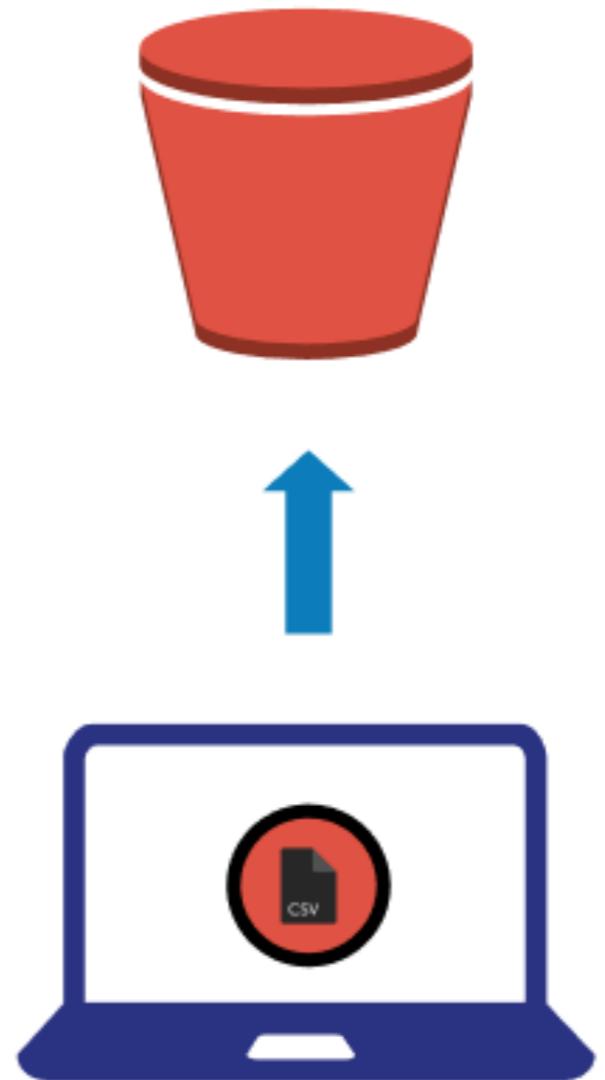
# Creating the client

```
s3 = boto3.client(  
    's3',  
    region_name='us-east-1',  
    aws_access_key_id=AWS_KEY_ID,  
    aws_secret_access_key=AWS_SECRET  
)
```



# Uploading files

```
s3.upload_file(  
    Filename='gid_requests_2019_01_01.csv',  
    Bucket='gid-requests',  
    Key='gid_requests_2019_01_01.csv')
```

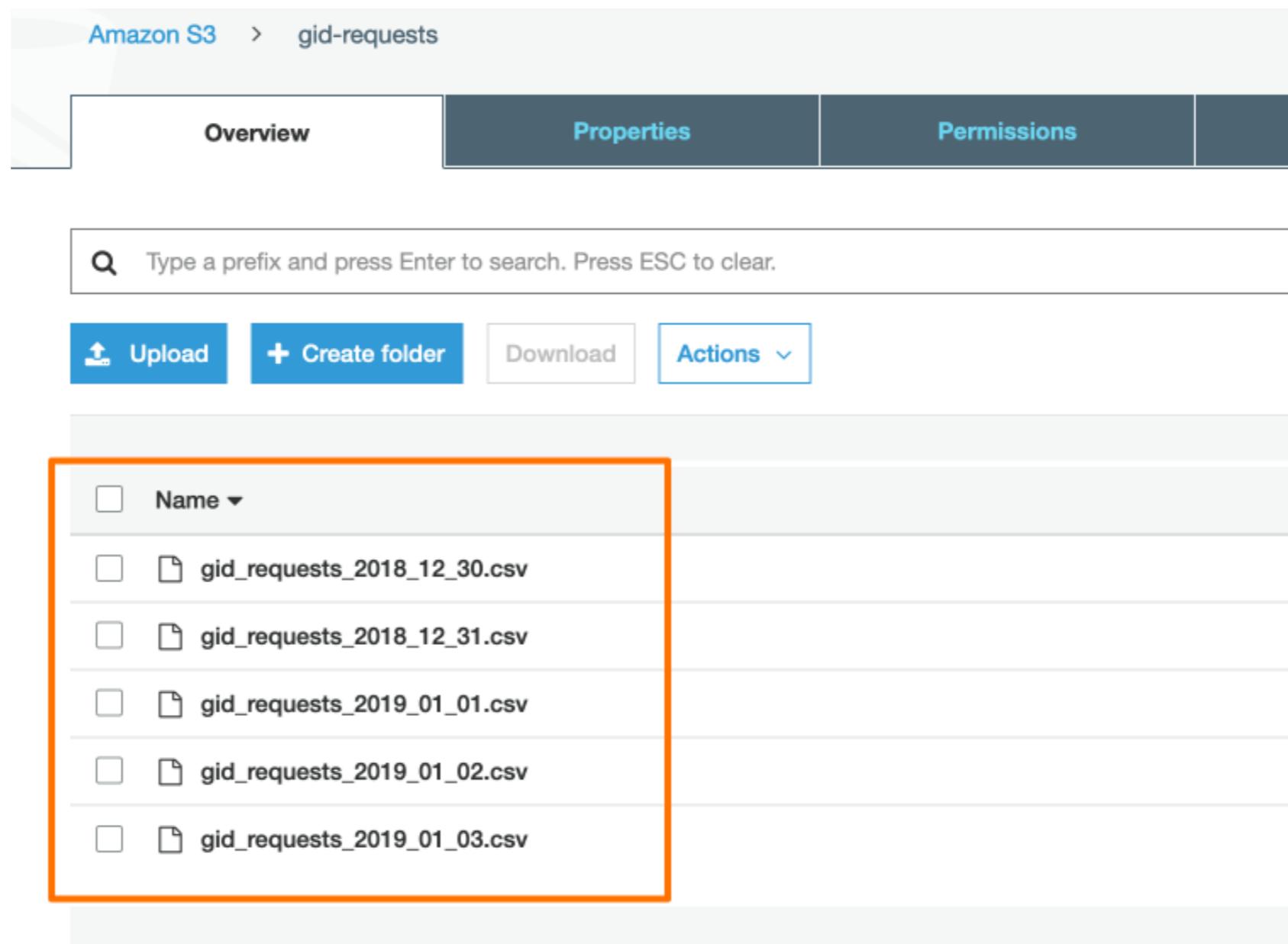


# Uploading files

The screenshot shows the AWS S3 console interface for the 'gid-requests' bucket. The top navigation bar includes the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, and a star icon. Below the navigation is the breadcrumb path 'Amazon S3 > gid-requests'. A horizontal tab bar at the top of the main content area includes 'Overview' (white background), 'Properties' (dark blue background, currently selected), 'Permissions' (dark blue background), and 'Management' (dark blue background). Below the tabs is a search bar with a magnifying glass icon and the placeholder text 'Type a prefix and press Enter to search. Press ESC to clear.' Underneath the search bar are four buttons: 'Upload' (blue background), 'Create folder' (blue background), 'Download' (light gray background), and 'Actions' (blue background with a dropdown arrow). The main content area displays a table of files. The columns are 'Name' (with a sorting arrow), 'Last modified' (with a sorting arrow), and two empty checkboxes. One file, 'gid\_requests\_2019\_01\_01.csv', is highlighted with an orange rectangular box around its entire row.

Name	Last modified		
gid_requests_2019_01_01.csv	Apr 18, 2019 1:17:24 PM GMT-0700	<input type="checkbox"/>	<input type="checkbox"/>

# Uploading more objects



Amazon S3 > gid-requests

Overview Properties Permissions

Type a prefix and press Enter to search. Press ESC to clear.

Upload Create folder Download Actions

<input type="checkbox"/> Name ▾
<input type="checkbox"/> <a href="#">gid_requests_2018_12_30.csv</a>
<input type="checkbox"/> <a href="#">gid_requests_2018_12_31.csv</a>
<input type="checkbox"/> <a href="#">gid_requests_2019_01_01.csv</a>
<input type="checkbox"/> <a href="#">gid_requests_2019_01_02.csv</a>
<input type="checkbox"/> <a href="#">gid_requests_2019_01_03.csv</a>

# Listing objects in a bucket

```
response = s3.list_objects(  
    Bucket='gid-requests',  
    MaxKeys=2,  
    Prefix='gid_requests_2019_')  
  
print(response)
```



# Listing objects in a bucket

```
'Contents': [{'Key': 'gid_requests_2018_12_30.csv',     'LastModified': datetime.datetime(2019, 4, 18, 21, 38, 30, tzinfo=tzutc()),  
    'ETag': '"2ffc551dccadb18aba921c2d88501325"',  
    'Size': 57137,  
    'StorageClass': 'STANDARD',  
    'Owner': {'DisplayName': 'maksim+aws-demos',  
              'ID': '12346cf1b2f0e923b64d624ce166bb570c6dae4a2a905b419916bd365ea5a596'}},  
    {''Key': 'gid_requests_2018_12_31.csv',  
     'LastModified': datetime.datetime(2019, 4, 18, 21, 38, 27, tzinfo=tzutc()),  
     'ETag': '"2ffc551dccadb18aba921c2d88501325"'},  
    ]
```

# Listing objects in a bucket

```
'Contents': [{'Key': 'gid_requests_2018_12_30.csv',  
    'LastModified': datetime.datetime(2019, 4, 18, 21, 38, 30, tzinfo=tzutc()),  
    'ETag': '"2ffc551dccadb18aba921c2d88501325"',  
    'Size': 57137,  
    'StorageClass': 'STANDARD',  
    'Owner': {'DisplayName        'ID    {'Key    'LastModified': datetime.datetime(2019, 4, 18, 21, 38, 27, tzinfo=tzutc()),  
    'ETag': '"2ffc551dccadb18aba921c2d88501325"'},
```

# Listing objects in a bucket

```
'Contents': [{'Key': 'gid_requests_2018_12_30.csv',  
    'LastModified    'ETag    'Size': 57137,  
    'StorageClass': 'STANDARD',  
    'OwnerDisplayName        'ID    {''Key': 'gid_requests_2018_12_31.csv',  
    'LastModified    'ETag
```

# Getting object metadata

```
response = s3.head_object(  
    Bucket='gid-requests',  
    Key='gid_requests_2018_12_30.csv')  
  
print(response)
```

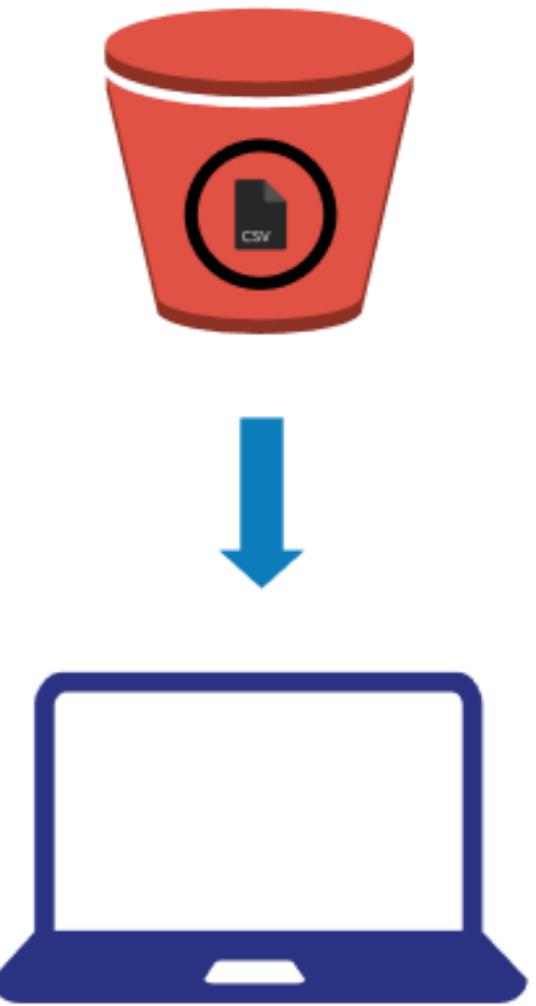


# Getting object metadata

```
{'ResponseMetadata': {'RequestId': '27FB1088203DD28E',  
                      'HostId': '',  
                      'HTTPStatusCode': 200,  
                      'RetryAttempts': 0},  
     'AcceptRanges': 'bytes',  
     'LastModified': datetime.datetime(2019, 4, 18, 21, 38, 30, tzinfo=tzutc()),  
     'ContentLength': 57137,  
     'ETag': '"2ffc551dccadb18aba921c2d88501325"',  
     'ContentType': 'binary/octet-stream',  
     'Metadata': {}}
```

# Downloading files

```
s3.download_file(  
    Filename='gid_requests_downed.csv',  
    Bucket='gid-requests',  
    Key='gid_requests_2018_12_30.csv')
```



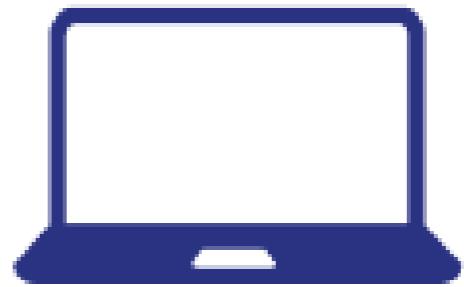
# Deleting objects

```
s3.delete_object(  
    Bucket='gid-requests',  
    Key='gid_requests_2018_12_30.csv')
```



# Summary

- Buckets are like folders
- Objects are like files
- `boto3.client()`
- `s3.upload_file()`
- `s3.list_objects()`
- `s3.head_object()`
- `s3.download_file()`
- `s3.delete_object()`



# Let's make some objects!

INTRODUCTION TO AWS BOTO IN PYTHON