



# Connecting to AWS S3 with Python

Ben Gorman

2020-04-18 799 words 4 minutes

Connecting AWS S3 to Python is easy thanks to the [boto3 package](#). In this tutorial, we'll see how to

1. Set up credentials to connect Python to S3
2. Authenticate with boto3
3. Read and write data from/to S3

## # 1. Set Up Credentials To Connect Python To S3

1. If you haven't done so already, you'll need to [create an AWS account](#).
2. Sign in to the management console. Search for and pull up the [S3 homepage](#).
3. Next, create a bucket. Give it a unique name, choose a region close to you, and keep the other default settings in place (or change them as you see fit). I'll call my bucket *cheez-willikers*.

aws Services Resource Groups

Amazon S3 > Create bucket

### Create bucket

**General configuration**

Bucket name  
cheez-willikers  
Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

Region  
US East (Ohio) us-east-2

**Bucket settings for Block Public Access**

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

☒ **Block all public access**  
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- ☒ **Block public access to buckets and objects granted through new access control lists (ACLs)**  
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- ☒ **Block public access to buckets and objects granted through any access control lists (ACLs)**  
S3 will ignore all ACLs that grant public access to buckets and objects.
- ☒ **Block public access to buckets and objects granted through new public bucket or access point policies**  
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- ☒ **Block public and cross-account access to buckets and objects through any public bucket or access point policies**  
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

4. Now we need to create a special user with S3 read and write permissions. Navigate to [Identity and Access Management \(IAM\)](#) and click on *users*.





The screenshot shows the AWS IAM console for account ID 886712843916. The left sidebar contains navigation links for Dashboard, Access management (Groups, Users, Roles, Policies, Identity providers, Account settings), Access reports (Access analyzer, Archive rules, Analyzers, Settings), Credential report, Organization activity, and Service control policies (SCPs). The main content area shows IAM Resources with counts: Users: 0, Groups: 0, Roles: 2, Identity Providers: 0, and Customer Managed Policies: 0. A red arrow points to the 'Users' count. Below this is the Security Status section, which is 1 out of 5 complete. It lists five tasks: Delete your root access keys (checked), Activate MFA on your root account (warning), Create individual IAM users (warning), Use groups to assign permissions (warning), and Apply an IAM password policy (warning).

5. Give your user a name (like *rconnector*) and give the user programmatic access.

### 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\*

[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.

6. Now we need to give the user permission to read and write data to S3.

We have two options here. The easy option is to give the user full access to S3, meaning the user can read and write from/to all S3 buckets, and even create new buckets, delete buckets, and change permissions to buckets. To do this, select *Attach Existing Policies Directly* > search for S3 > check the box next to AmazonS3FullAccess.

### Add user

1 2 3 4 5

#### Set permissions

The screenshot shows the 'Set permissions' step. It has three tabs: 'Add user to group', 'Copy permissions from existing user', and 'Attach existing policies directly' (which is selected). Below the tabs is a 'Create policy' button. A table titled 'Filter policies' shows search results for 'S3'. The table has columns for 'Policy name', 'Type', and 'Used as'. It lists four policies: AmazonDMSRedshiftS3Role, AmazonS3FullAccess, AmazonS3ReadOnlyAccess, and QuickSightAccessForS3StorageManagementAnalyticsReadOnly. The 'AmazonS3FullAccess' policy is highlighted.

|                          | Policy name                                             | Type        | Used as |
|--------------------------|---------------------------------------------------------|-------------|---------|
| <input type="checkbox"/> | AmazonDMSRedshiftS3Role                                 | AWS managed | None    |
| <input type="checkbox"/> | AmazonS3FullAccess                                      | AWS managed | None    |
| <input type="checkbox"/> | AmazonS3ReadOnlyAccess                                  | AWS managed | None    |
| <input type="checkbox"/> | QuickSightAccessForS3StorageManagementAnalyticsReadOnly | AWS managed | None    |

The harder, but better approach is to give the user access to read and write files only for the bucket we just created. To do this, select *Attach Existing Policies Directly* and then work through the Visual Policy Editor.

- **Service:** S3
- **Actions:** Everything besides Permissions Management
- **Resources:** add the following Amazon Resource Names (ARNs):



- *job*: Any
- *object*: cheez-willikers (bucket) | Any (objects)
- Review the policy and give it a name.
- Attach the policy to the user.

## How To Make A Custom Policy In AWS With Permission To Read And Write To A Specific



7. Skip through the remaining steps to create the user until you get a Success message with user credentials. Download the user credentials and store them somewhere safe because **this is your only opportunity to see the Secret Access Key from the AWS console**. If you lose it, you can't recover it (but you can create a new key). These credentials are what we'll use to authenticate from Python, to get access to our S3 bucket.

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://886712843916.signin.aws.amazon.com/console>

Download .csv

|   | User         | Access key ID        | Secret access key |
|---|--------------|----------------------|-------------------|
| ▶ | ✓ reconector | AKIA445BUH2GBGUQKQNU | ***** Show        |

## # 2. Authenticate With boto3

Obviously, if you haven't done so already, you'll need to install the boto3 package.

### Python

```
pip install boto3
```

In order to connect to S3, you need to authenticate. You can do this in many ways using boto. Perhaps the easiest and most direct method is just to include



## ▼ Python



```
s3 = boto3.resource(
    service_name='s3',
    region_name='us-east-2',
    aws_access_key_id='mykey',
    aws_secret_access_key='mysecretkey'
)
```

Note that *region\_name* should be the region of your S3 bucket.

You can print a list of all the S3 buckets in your resource like this.

## ▼ Python



```
# Print out bucket names
for bucket in s3.buckets.all():
    print(bucket.name)

# cheez-willikers
```

Two errors I stumbled upon here were

1. *Forbidden (HTTP 403)* because I inserted the wrong credentials for my user
2. *Forbidden (HTTP 403)* because I incorrectly set up my user's permission to access S3

Alternatively, you can create environment variables with your credentials like so

## ▼ Python



```
import os
os.environ["AWS_DEFAULT_REGION"] = 'us-east-2'
os.environ["AWS_ACCESS_KEY_ID"] = 'mykey'
os.environ["AWS_SECRET_ACCESS_KEY"] = 'mysecretkey'
```

Or you can store your credentials inside a *credentials* file. (See [here](#) for details).

## # 3. Read And Write Data From/To S3

Let's start by uploading a couple CSV files to S3.

## ▼ Python



```
import pandas as pd

# Make dataframes
foo = pd.DataFrame({'x': [1, 2, 3], 'y': ['a', 'b', 'c']})
bar = pd.DataFrame({'x': [10, 20, 30], 'y': ['aa', 'bb', 'cc']})

# Save to csv
foo.to_csv('foo.csv')
bar.to_csv('bar.csv')
```

## ▼ Python



```
# Upload files to S3 bucket
s3.Bucket('cheez-willikers').upload_file(Filename='foo.csv', Key='foo.c
s3.Bucket('cheez-willikers').upload_file(Filename='bar.csv', Key='bar.c
```

(Here *Filename* is the name of the local file and *Key* is the filename you'll see in S3).





1. You've misspelled or inserted the wrong region name for the environment variable `AWS_DEFAULT_REGION` (if you're using environment vars)
2. You've misspelled or inserted the wrong region name for the `region_name` parameter of `boto3.resource()` (if you aren't using environment vars)
3. You've incorrectly set up your user's permissions

Now let's list all the objects in our bucket.

#### ▼ Python



```
for obj in s3.Bucket('cheez-willikers').objects.all():
    print(obj)

# s3.ObjectSummary(bucket_name='cheez-willikers', key='bar.csv')
# s3.ObjectSummary(bucket_name='cheez-willikers', key='foo.csv')
```

This returns a list of `s3_objects`. We can load one of these CSV files from S3 into python by fetching an object and then the object's `Body`, like so.

#### ▼ Python



```
# Load csv file directly into python
obj = s3.Bucket('cheez-willikers').Object('foo.csv').get()
foo = pd.read_csv(obj['Body'], index_col=0)
```

Alternatively, we could download a file from S3 and then read it from disc.

#### ▼ Python



```
# Download file and read from disc
s3.Bucket('cheez-willikers').download_file(Key='foo.csv', Filename='foo.csv')
pd.read_csv('foo2.csv', index_col=0)
```

Updated on 2020-04-18

[Read Markdown](#)



🔖 Python, tutorial

[Back](#) | [Home](#)

◀ [Connecting to AWS S3 with R](#)

[Sending Adobe Analytics Clickstream Data To AWS S3](#) ▶

Subscribe For Updates

