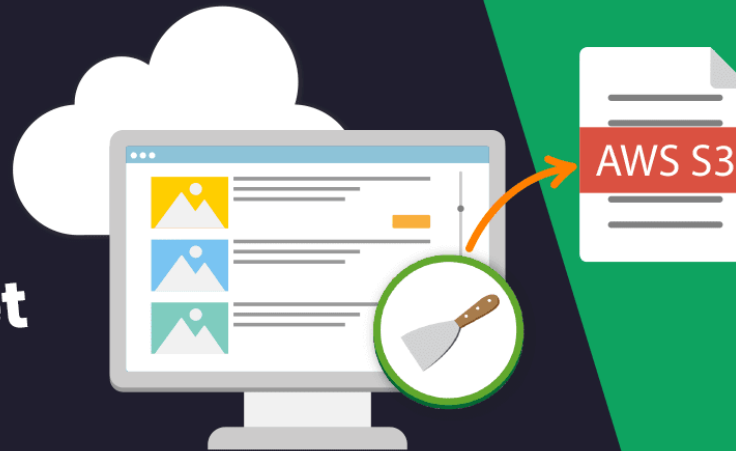


THE SCRAPY PLAYBOOK

Scrapy

Saving Data to Amazon AWS S3 Bucket



Saving Scraped Data To Amazon AWS S3 Bucket With Scrapy

Storing CSV and JSON files of scraped data on your local computer is fine for small projects, however, a better option is to store it on a file storage system like in a Amazon AWS S3 bucket.

In this guide, we show you how to store your scraped data in CSV and JSON files and automatically sync it with AWS S3 bucket.

- [What Are Scrapy Feed Exporters?](#)
- [Getting Setup With Amazon AWS S3](#)
- [Saving CSV Files To AWS S3 Bucket](#)
- [Saving JSON Files To AWS S3 Bucket](#)

First, let's go over what are **Scrapy Feed Exporters**.

Need help scraping the web?

Then check out [ScrapeOps](#), the complete toolkit for web scraping.



Proxy Manager



Scraper Monitoring



Job Scheduling

What Are Scrapy Feed Exporters?

The need to save scraped data to a file is a very common requirement for developers, so to make our lives easier the developers behind Scrapy have implemented [Feed Exporters](#).

Feed Exporters are a ready made toolbox of methods we can use to easily save/export our scraped data into:

- JSON file format
- CVS file format
- XML file format
- Python's pickle format

And save them to:

- The local machine Scrapy is running on
- A remote machine using FTP (file transfer protocol)
- Amazon S3 Storage
- Google Cloud Storage
- Standard output

In this guide, we will walk you through how you can save your data to CSV and JSON files and store those files in a AWS S3 bucket using Scrapy.

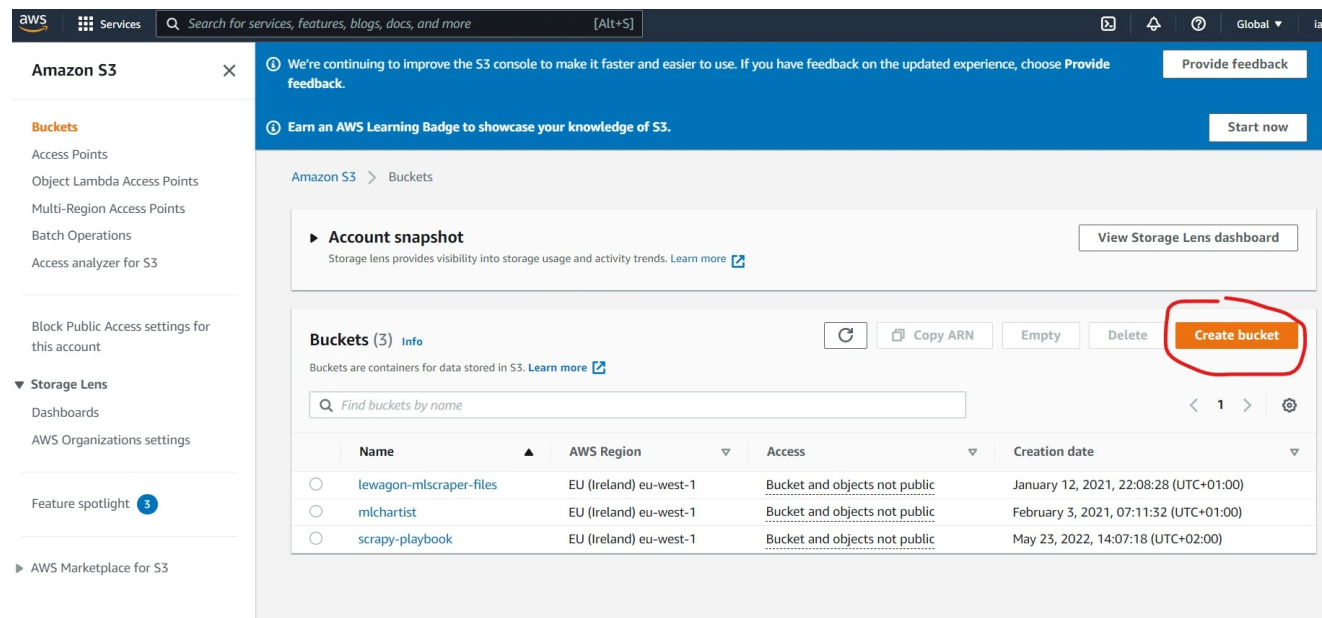
Getting Setup With Amazon AWS S3

The first step is we need to setup a AWS S3 bucket, get the appropriate access keys and install the `botocore` library if you haven't done so already.

1. Create AWS S3 Bucket

First we need to signup for a [AWS account here](#). Amazon gives you 5GB of free S3 storage for 12 months on their free tier.

Once logged in navigate to S3 by entering **"S3"** into the search bar, and once there click on the **"Create Bucket"** button.



Then give your bucket a name, and you can leave all the default settings as they are.

aws Services Search for services, features, blogs, docs, and more [Alt+S]

Amazon S3 > Buckets > Create bucket

Create bucket [Info](#)

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

Bucket name

scrapy-playbook

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

AWS Region

EU (Ireland) eu-west-1

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.

Choose bucket

Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership

2. Generate S3 Access Credentials

Next, we need to enter **IAM** into the search bar and navigate to the **IAM service**.

Once there, click on **Users**.

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

User groups

Users

Roles

Policies

Identity providers

Account settings

Access reports

Access analyzer

Archive rules

Analyzers

Settings

Credential report

Organization activity

Service control policies (SCPs)

Introducing the new IAM dashboard experience

We've redesigned the IAM dashboard experience to make it easier to use. [Let us know what you think.](#)

IAM dashboard

Security recommendations 1

Add MFA for root user

Add MFA for root user - Enable multi-factor authentication (MFA) for the root user to improve security for this account.

Add MFA

Root user has no active access keys

Using access keys attached to an IAM user instead of the root user improves security.

IAM resources

User groups

2

Users

2

Roles

8

Policies

6

Identity providers

0

What's new

Updates for features in IAM

Right-size permissions for more roles in your account using IAM Access Analyzer to generate 50 fine-grained IAM policies per day. 6 months ago

View all

AWS Account

Account ID

647479052395

Account Alias

647479052395

Create

Sign-in URL for IAM users in this account

[https://647479052395.signin.aws.a](https://647479052395.signin.aws.amazon.com/console)
[mazon.com/console](https://647479052395.signin.aws.a)

Quick Links

My security credentials

Manage your access keys, multi-factor authentication (MFA) and other credentials.

Tools

[Policy simulator](#)

Then on **Add Users**.

aws

Services

Search for services, features, blogs, docs, and more

[Alt+S]

Global

iam

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

User groups

Users

Roles

Policies

Identity providers

Account settings

Access reports

Access analyzer

Archive rules

Analyzers

Settings

Credential report

Organization activity

Service control policies (SCPs)

Introducing the new Users list experience

We've redesigned the Users list experience to make it easier to use. [Let us know what you think.](#)

IAM > Users

Users (2) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

Find users by username or access key

Refresh

Delete

Add users

	User name	Groups	Last activity	MFA	Password age	Active key age
<input type="checkbox"/>	s3-full-access	lewagon-full-access	425 days ago	None	None	464 days ago
<input type="checkbox"/>	scrapy-demo	scrapy-playbook	26 minutes ago	None	None	2 hours ago

Here give your user a user name, and select **Access key - Programmatic access** in the **Select AWS access type**

section.

The screenshot shows the 'Add user' console in the AWS IAM service. The top navigation bar includes the AWS logo, 'Services' link, a search bar, and a '[Alt+S]' shortcut. The main heading is 'Add user' with a progress indicator showing five steps, with step 1 being the active one. Below the heading is the section 'Set user details' with a subtext: 'You can add multiple users at once with the same access type and permissions. [Learn more](#)'. There is a text input field for 'User name*' containing the text 'scrapy'. Below the input field is a blue button with a plus icon and the text 'Add another user'. Further down is the section 'Select AWS access type' with a subtext: 'Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)'. There are two radio button options: 'Access key - Programmatic access' (which is selected) and 'Password - AWS Management Console access'. A red arrow points to the 'Access key - Programmatic access' radio button. The description for the selected option reads: 'Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.'

Then click on **Attach existing policies directly**, and search for "S3" in the search bar and select **AmazonS3FullAccess**.

The screenshot shows the 'Add user' console in the AWS IAM service, specifically the 'Set permissions' step. The top navigation bar is the same as the previous screenshot. The main heading is 'Add user' with a progress indicator showing five steps, with step 2 being the active one. Below the heading is the section 'Set permissions'. There are three buttons: 'Add user to group', 'Copy permissions from existing user', and 'Attach existing policies directly' (which is highlighted with a blue border). Below these buttons is a 'Create policy' button. Below that is a search bar with the text 's3' and a 'Filter policies' dropdown. To the right of the search bar is the text 'Showing 9 results'. Below the search bar is a table with the following columns: 'Policy name', 'Type', and 'Used as'. The table contains the following rows:

	Policy name	Type	Used as
<input type="checkbox"/>	AmazonDMSRedshiftS3Role	AWS managed	None
<input checked="" type="checkbox"/>	AmazonS3FullAccess	AWS managed	Permissions policy (3)
<input type="checkbox"/>	AmazonS3ObjectLambdaExecutionRolePolicy	AWS managed	None
<input type="checkbox"/>	AmazonS3OutpostsFullAccess	AWS managed	None
<input type="checkbox"/>	AmazonS3OutpostsReadOnlyAccess	AWS managed	None
<input type="checkbox"/>	AmazonS3ReadOnlyAccess	AWS managed	None
<input type="checkbox"/>	AWSBackupServiceRolePolicyForS3Backup	AWS managed	None

A red arrow points to the 'AmazonS3FullAccess' row, which is highlighted in blue.

Finally, skip the **Tags** section and click **Create User**. Here you will be shown the following screen showing your

username, **Access key ID**, and **secret access key**. Copy these down.

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://647479052395.signin.aws.amazon.com/console>

Download .csv

	User	Access key ID	Secret access key
▶	✓ scrapy	AKIAZNQGCJBV2RCOXJDU	***** Show

3. Install boto3

Finally, the last step to getting setup is to install `boto3` on our machine which is as simple as:

```
pip install boto3
```

Saving CSV Files To AWS S3 Bucket

Configuring Scrapy to save our S3 bucket is very simple. We just need to update the `settings.py` with the following:

```
FEEDS = {
```

```
"s3://scrapy-playbook/%(name)s/%(name)s_%(time)s.csv": {  
    "format": "csv",  
}
```

```
AWS_ACCESS_KEY_ID = 'YOUR_AWS_ACCESS_KEY_ID'  
AWS_SECRET_ACCESS_KEY = 'YOUR_AWS_SECRET_ACCESS_KEY'
```

Here, I'm saving my CSV files into the `scrapy-playbook` S3 bucket, and inside that bucket it will create/use the folder using the name of my spider and call the file **SpiderName_TimeCreated.csv**.

Now, everytime I run my spider the CSV files will be saved to my S3 bucket.

Saving JSON Files To AWS S3 Bucket

Configuring Scrapy to save our CSV files to our S3 bucket is very simple. We just need to update the `settings.py` with the following:

```
FEEDS = {  
    "s3://scrapy-playbook/%(name)s/%(name)s_%(time)s.jsonl": {  
        "format": "jsonlines",  
    }  
}  
  
AWS_ACCESS_KEY_ID = 'YOUR_AWS_ACCESS_KEY_ID'  
AWS_SECRET_ACCESS_KEY = 'YOUR_AWS_SECRET_ACCESS_KEY'
```

Here, I'm saving my JSON files into the `scrapy-playbook` S3 bucket, and inside that bucket it will create/use the folder using the name of my spider and call the file **SpiderName_TimeCreated.jsonl**.

Now, everytime I run my spider the JSON files will be saved to my S3 bucket.

More Scrapy Tutorials

For more information on saving your scraped data to different formats then be sure to check these guides:

- [Saving Data to CSV](#)
- [Saving Data to JSON](#)
- [Saving Data to SQLite Database](#)
- [Saving Data to MySQL Database](#)
- [Saving Data to Postgres Database](#)

If you would like to learn more about Scrapy in general, then be sure to check out [The Scrapy Playbook](#).