# Moving Data In S3 And Redshift With Python

AWS offers a nice solution to data warehousing with their columnar database, [Redshift](http://docs.aws.amazon.com/redshift/latest/mgmt/welcome.html), and an object storage, [S3](http://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html). Python and AWS SDK make it easy for us to move data in the ecosystem.

In this post, I will present code examples for the scenarios below:

* Uploading data from S3 to Redshift
* Unloading data from Redshift to S3
* Uploading data to S3 from a server or local computer

The best way to load data to Redshift is to go via S3 by calling a copy command because of its ease and speed. You can upload data into Redshift from both flat files and json files.

You can also unload data from Redshift to S3 by calling an unload command. [Boto3](http://boto3.readthedocs.io/en/latest/) (AWS SDK for Python) enables you to upload file into S3 from a server or local computer.

**Preparation**

I usually encourage people to use Python 3. When it comes to AWS, I highly recommend to use Python 2.7. It will make your life much easier. For example, if you want to deploy a Python script in an EC2 instance or EMR through [Data Pipeline](http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/what-is-datapipeline.html) to leverage their serverless archtechture, it is faster and easier to run code in 2.7. The code examples are all written 2.7, but they all work with 3.x, too.

You need to install boto3 and psycopg2 (which enables you to connect to Redshift).

[cc lang="python" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

pip install boto3

pip install psycopg2

[/cc]

Finally, you need to install the AWS Command Line Interface (see [Installing the AWS Command Line Interface](http://docs.aws.amazon.com/cli/latest/userguide/installing.html)) and configure it (see [Configuring the AWS CLI](http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html)) in the server you are running your program or the local machine. This is not necessary if you are running the code through Data Pipeline. This is pre-installed in the EC2 instance. Boto3 leverages the credentials stored in AWS CLI. Once AWS CLI is configured, you do not need to enter any AWS credentials in the code to move data to and from S3.

Let’s get down to the business!

**Code Examples**

**Example 1: Upload a file into Redshift from S3**

There are many options you can specify. In this case, the data is a pipe separated flat file. You can upload json, csv and so on. For further reference on Redshift copy command, you can start from [here](http://docs.aws.amazon.com/redshift/latest/dg/r_COPY.html).

[cc lang="python" tab\_size="4" lines="-1" theme="GeSHi" line\_numbers="false"

[/cc]

**Example 2: Unload data from Redshift into S3**

In this example, the data is unloaded as gzip format with manifest file. This is the recommended file format for unloading according to AWS. Unloading also has many options and you can create a different file formats according to your requirements. For further information, you can start from [here](http://docs.aws.amazon.com/redshift/latest/dg/t_Unloading_tables.html).

[cc lang="python" tab\_size="4" lines="-1" theme="GeSHi" line\_numbers="false"

[/cc]

**Example 3: Upload files into S3 with Boto3**

You need to have AWS CLI configured to make this code work. Whatever the credentials you configure is the environment for the file to be uploaded.

[cc lang="python" tab\_size="4" lines="-1" theme="GeSHi" line\_numbers="false"

[/cc]

**Example 4: Upload files into S3 by calling AWS command in Python**

All files in the specified local directory will be recursively copied to S3 by using aws cli.

[cc lang="python" tab\_size="4" lines="-1" theme="GeSHi" line\_numbers="false"

[/cc]

In the next few posts, I will show you the examples of ingesting API data into S3 and Redshift. Stay tuned!