WeCloudData

Programmatic Access with S3

Data Engineering Diploma

Content developed by: WeCloudData Academy

1 - Working with S3 via AWS CLI

AWS supports several APIs to work with different services. If we want to work with the object store S3, we will either use the `AWS CLI` (command line) API or the `boto3` python API.

1.1 - Set up AWS Credential

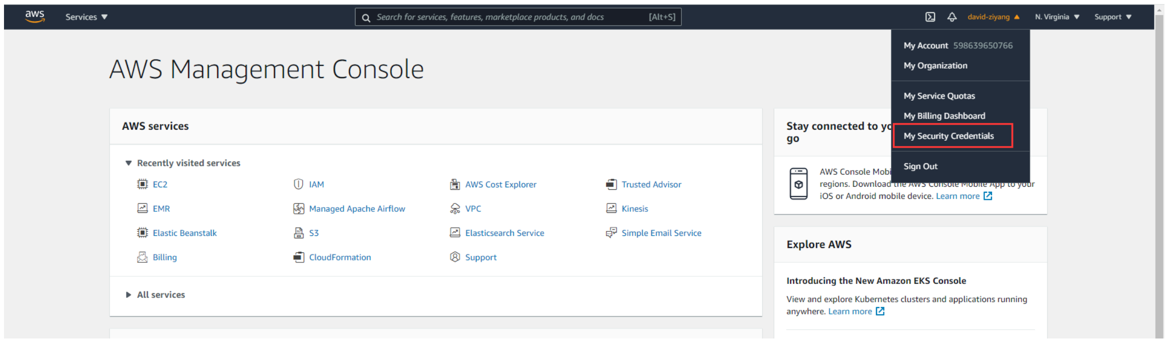
There are two ways to generate the AWS credential

- Create the IAM user to get the credential
- Generate the credential for root user directly (not recommended)

Create the IAM user to get the credential

We can create an IAM user to get the credential.

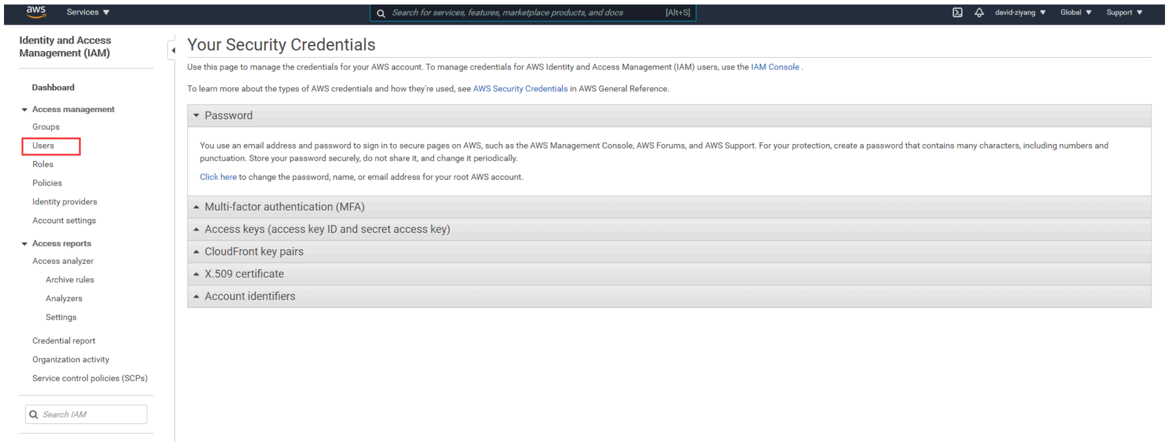
Click on the **My Security Credential** .



AWS Console Mobi… regions. Download the AWS Console Mobile App to your iOS or Android mobile device. Learn more

View and explore Kubernetes clusters and applications running anywhere. Learn more

Click on **Users**



Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials for AWS Identity and Access Management (IAM) users, use the IAM Console .

To learn more about the types of AWS credentials and how they're used, see AWS Security Credentials in AWS General Reference.

Password

You use an email address and password to sign in to secure pages on AWS, such as the AWS Management Console, AWS Forums, and AWS Support. For your protection, create a password that contains many characters, including numbers and punctuation. Store your password securely, do not share it, and change it periodically.

Click here to change the password, name, or email address for your root AWS account.

Multi-factor authentication (MFA)

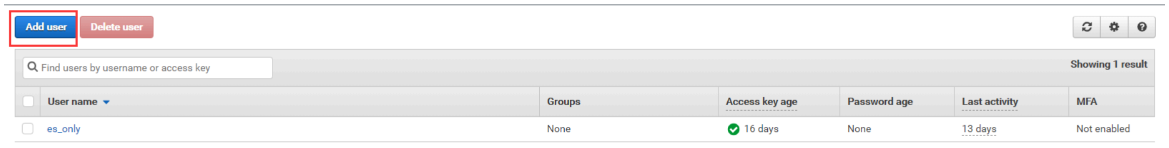
Access keys (access key ID and secret access key)

CloudFront key pairs

X.509 certificate

Account identifiers

Click on **Add user**

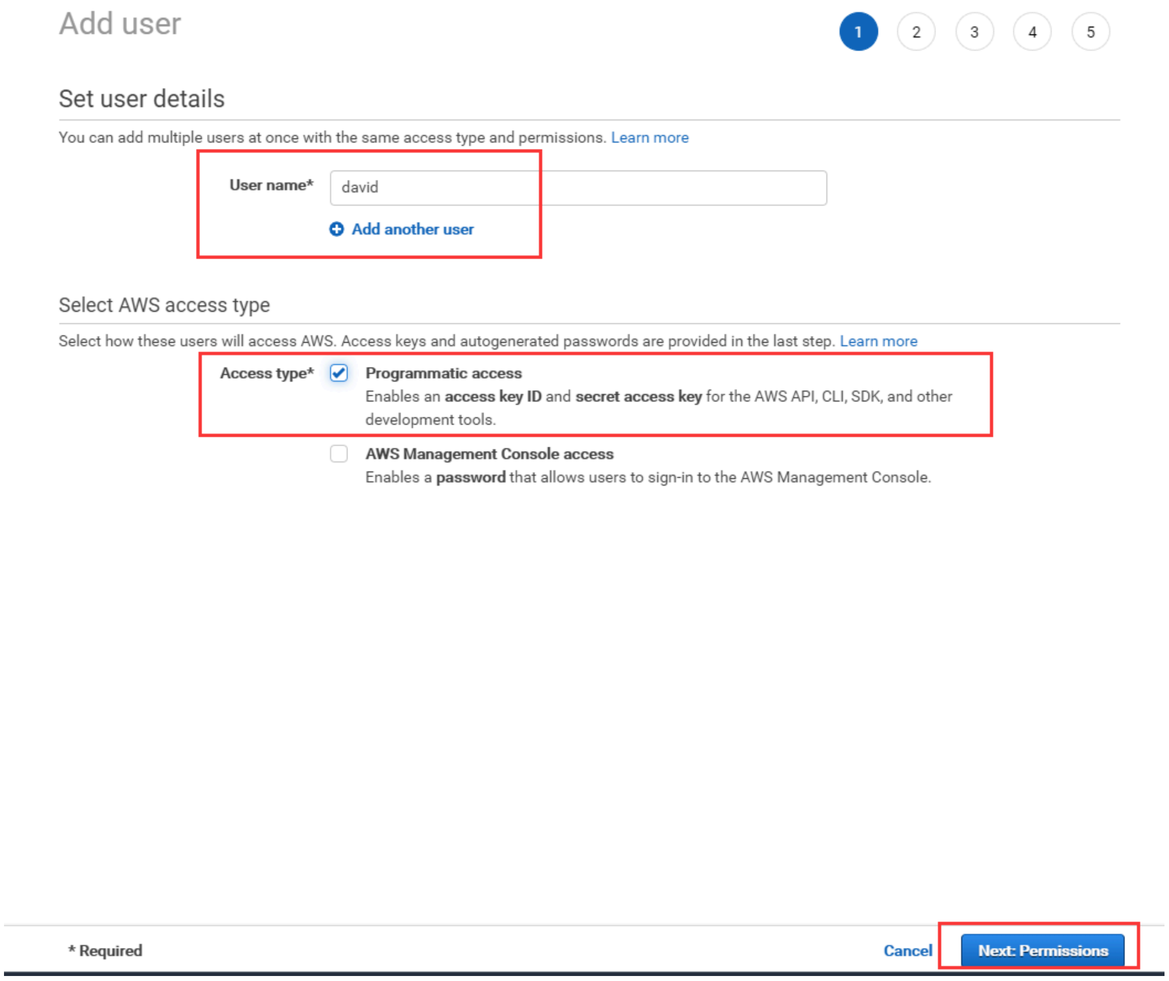


	User name	Groups	Access key age	Password age	Last activity	MFA
	es_only	None	16 days	None	13 days	Not enabled

For the **User name**, you can give it name whatever you like for example here I use `david`

For the **Access type**, we choose `Programmatic access` Here

Then click on **Next**



Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* david

Add another user

Select AWS access type

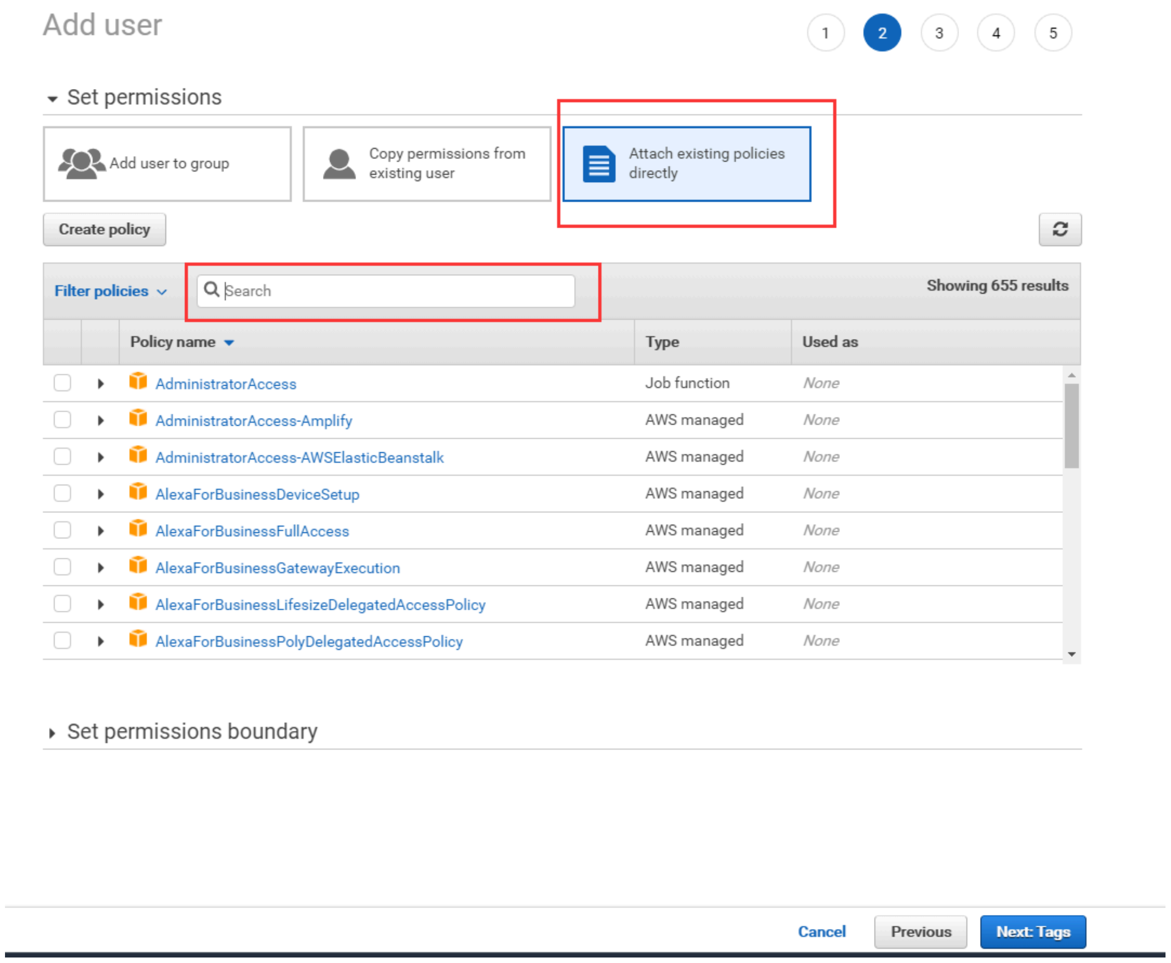
Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. Learn more

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.

For the permission here, we need to choose **Attach existing policies directly**

Then we need to search the policy we need.



Add user

1 2 3 4 5

Set permissions

Add user to group | Copy permissions from existing user | Attach existing policies directly

Create policy

Filter policies | Search | Showing 655 results

		Policy name	Type	Used as
☐	▸	AdministratorAccess	Job function	None
☐	▸	AdministratorAccess-Amplify	AWS managed	None
☐	▸	AdministratorAccess-AWSElasticBeanstalk	AWS managed	None
☐	▸	AlexaForBusinessDeviceSetup	AWS managed	None
☐	▸	AlexaForBusinessFullAccess	AWS managed	None
☐	▸	AlexaForBusinessGatewayExecution	AWS managed	None
☐	▸	AlexaForBusinessLifesizeDelegatedAccessPolicy	AWS managed	None
☐	▸	AlexaForBusinessPolyDelegatedAccessPolicy	AWS managed	None

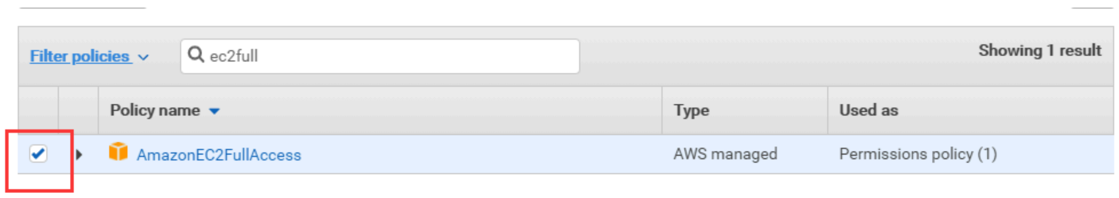
Set permissions boundary

Cancel | Previous | Next: Tags

Here we need the `AmazonEC2FullAccess` and `AmazonS3FullAccess`

So we can search the `ec2full` and in the result list, check the check box next to the `AmazonEC2FullAccess`

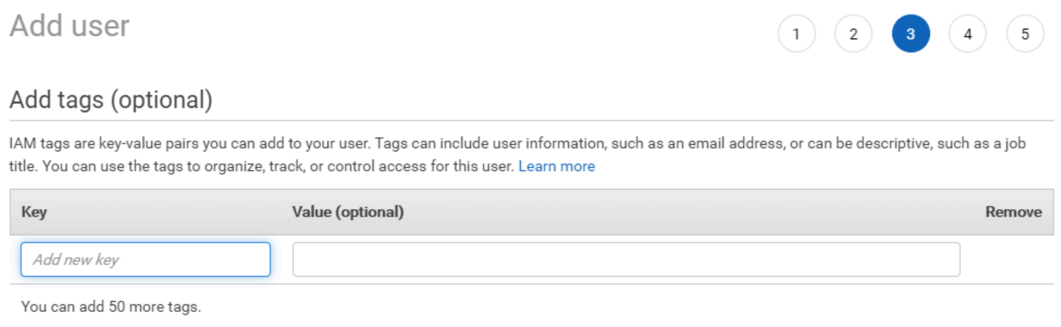
We need to search `s3full` and select the `AmazonS3FullAccess` as well.



Filter policies | ec2full | Showing 1 result

		Policy name	Type	Used as
✓	▸	AmazonEC2FullAccess	AWS managed	Permissions policy (1)

For the tag page, you can leave it empty.



Add user

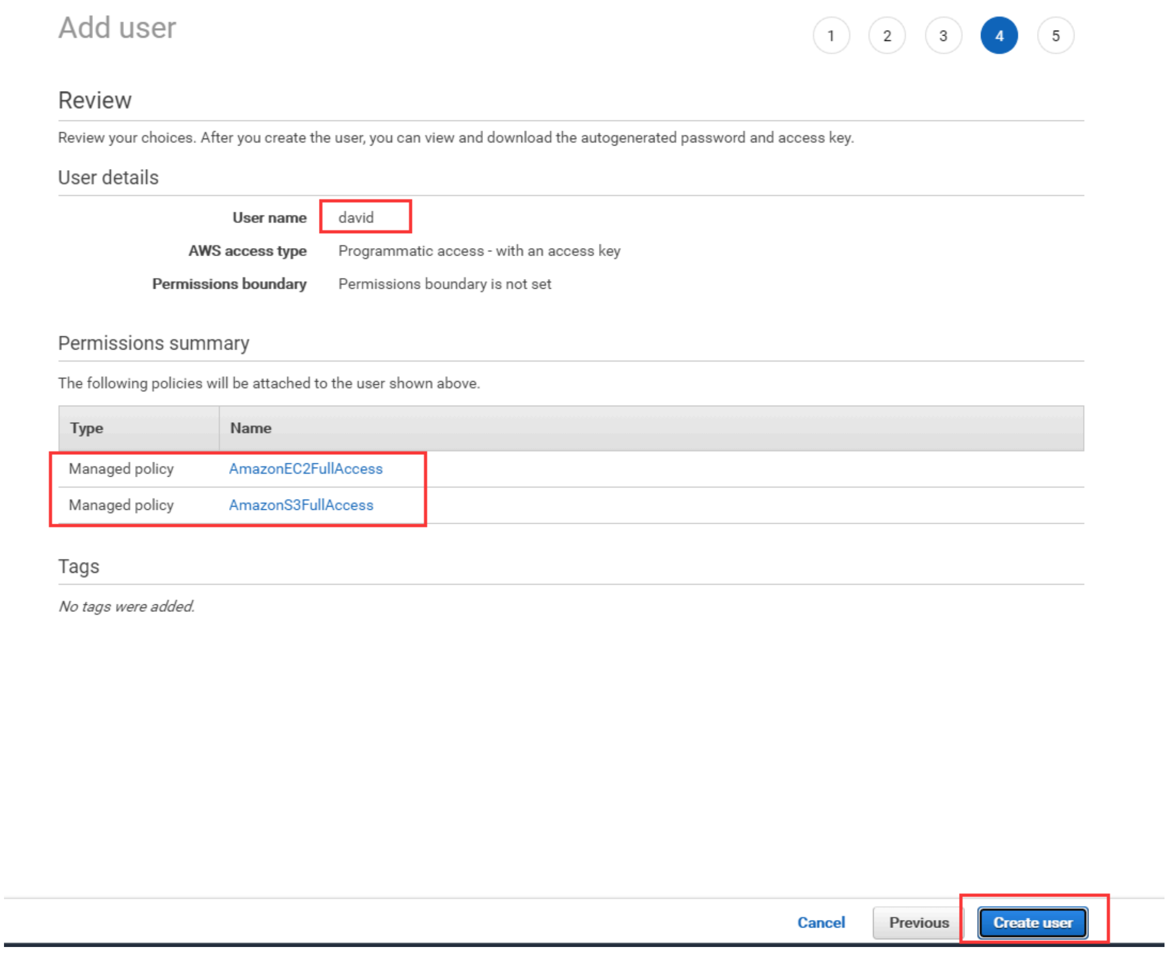
Add tags (optional)

IAM tags are key-value pairs you can add to your user. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this user. Learn more

Key	Value (optional)	Remove
Add new key		

You can add 50 more tags.

Then we need to review the permissions, make sure we have the **AmazonEC2FullAccess** and **AmazonS3FullAccess**



Add user

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

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

Permissions summary

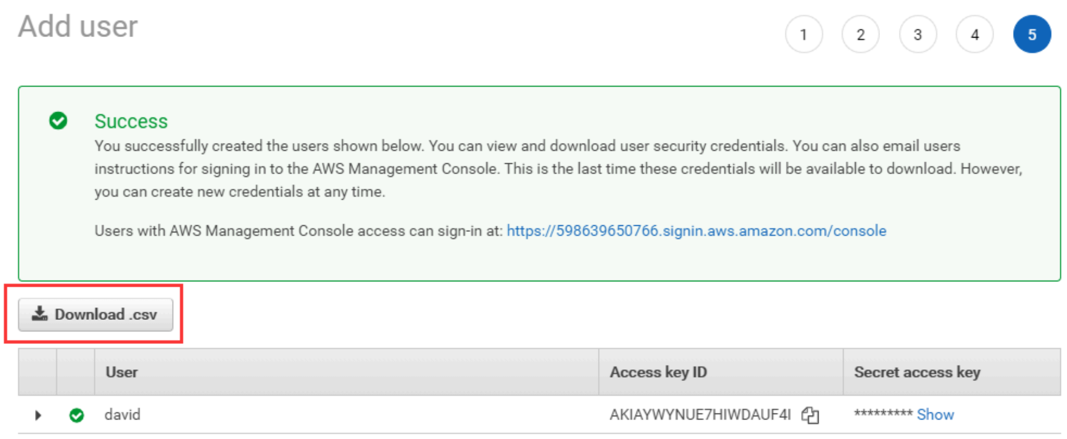
The following policies will be attached to the user shown above.

Type	Name
Managed policy	AmazonEC2FullAccess
Managed policy	AmazonS3FullAccess

Tags

No tags were added.

Finally, we need to **download** the credential. This is the **only** chance for you to download the credentials.



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://598639650766.signin.aws.amazon.com/console

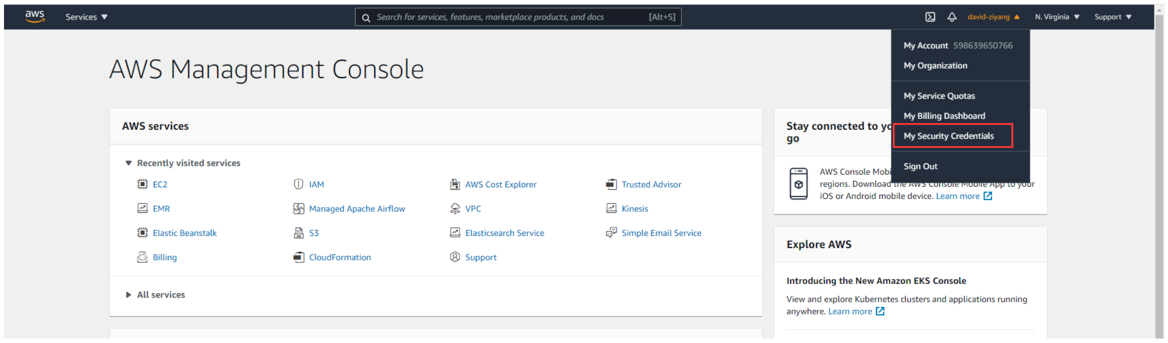
Download .csv

		User	Access key ID	Secret access key
▸	✓	david	AKIAYWYNUE7HIWDAUF4I	********* Show

Generate the credential for the root user (not recommended)

We can also generate the credential for the root user directly.

Click on the **My Security Credential** .



AWS Management Console

AWS services

Recently visited services

EC2, IAM, AWS Cost Explorer, Trusted Advisor, EMR, Managed Apache Airflow, VPC, Kinesis, Elastic Beanstalk, S3, Elasticsearch Service, Simple Email Service, Billing, CloudFormation, Support

All services

Stay connected to yo[...] go

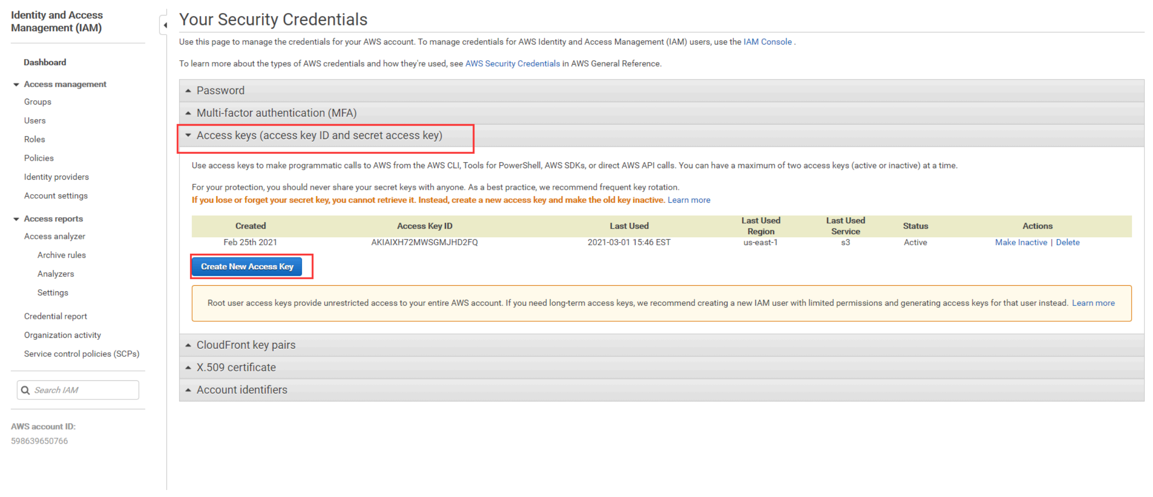
AWS Console Mobi[...] regions. Download the AWS Console Mobile App to your iOS or Android mobile device. Learn more

Explore AWS

Introducing the New Amazon EKS Console

View and explore Kubernetes clusters and applications running anywhere. Learn more

Click on the **Access keys** and then click the **Create New Access Key**



AWS account ID: 598639650766

Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials for AWS Identity and Access Management (IAM) users, use the IAM Console .

To learn more about the types of AWS credentials and how they're used, see AWS Security Credentials in AWS General Reference.

Password

Multi-factor authentication (MFA)

Access keys (access key ID and secret access key)

Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time.

For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation.
If you lose or forget your secret key, you cannot retrieve it. Instead, create a new access key and make the old key inactive. Learn more

Created	Access Key ID	Last Used	Last Used Region	Last Used Service	Status	Actions
Feb 25th 2021	AKIAIXH72MWSGMJHD2FQ	2021-03-01 15:46 EST	us-east-1	s3	Active	Make Inactive | Delete

Create New Access Key

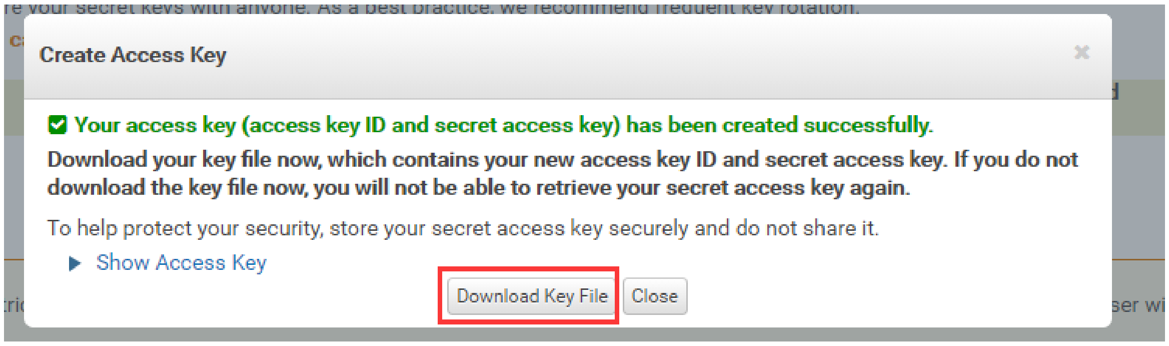
Root user access keys provide unrestricted access to your entire AWS account. If you need long-term access keys, we recommend creating a new IAM user with limited permissions and generating access keys for that user instead. Learn more

CloudFront key pairs

X.509 certificate

Account identifiers

Download your key file


Create Access Key

Your access key (access key ID and secret access key) has been created successfully.

Download your key file now, which contains your new access key ID and secret access key. If you do not download the key file now, you will not be able to retrieve your secret access key again.

To help protect your security, store your secret access key securely and do not share it.

Install AWS CLI on EC2 instance (or your laptop locally)

Make sure you're SSH connected to the EC2 instance

```
sudo apt-get update
sudo apt install python3-pip
sudo pip3 install awscli
sudo pip3 install boto3
```

Configure AWS Credential on EC2 (or your laptop locally)

To have programmatical access to AWS services, you will need to have credentials, a set of keys/secret. You can find the **Access Key** and **Secret Access Key** In the **csv** file you download before.

You can configure the credentials from the command line.

```
aws configure
```

Credentials for this lab (update accordingly):

- **AWS Access Key ID**: `<your access key>`
- **AWS Secret Access Key**: `<your secret access key>`
- **Default region name**: `us-east-1`
- **Default output format**: `json`

You're all set.

1.2 - Working with S3 via AWS CLI

The following s3 exercises can be done from the EC2 instance or your local laptop as long as the credentials are set up properly.

The demo below uses the EC2 instance. Make sure you are connected to the instance via ssh.

Make a bucket

You need to change the `weclouddata-demo` to your own bucket name

```
aws s3 mb s3://weclouddata-demo
```

List buckets

```
aws s3 ls s3://weclouddata/
aws s3 ls s3://weclouddata-demo/
```

Copy file

Upload a file from s3 to ec2 or local

```
touch test.csv
aws s3 cp test.csv s3://weclouddata-demo/
aws s3 ls s3://weclouddata-demo/
```

Download a file from local/ec2 to s3

```
aws s3 cp s3://weclouddata-demo/test.csv ./test1.csv
ll
```

Sync folder

```
mkdir test
cd test && touch test2.csv
cd ..
aws s3 sync test s3://weclouddata-demo/test
aws s3 ls s3://weclouddata-demo/test
```

Delete object

```
aws s3 rm s3://weclouddata-demo/test
```

---

2 - Working with S3 via Boto3 SDK (Optional)

Reference

- [Boto3 S3 API Doc]()

SSH to EC2 instance and run Python3 from the command line

```
python3
```

In python prompt

```
import boto3
# import os
```

2.1 - Get a `boto3` session

- Option 1: run `aws configure`
- Option 2: set ACCESS_KEY and SECRET_KEY in the OS environment variable
- Option 3: hard code (not suggested)

Option 1: create boto3 custom session

```
import boto3
session = boto3.Session()
```

Option 2: environment variable

```
import os
import boto3

ACCESS_KEY = os.getenv('ACCESS_KEY')
SECRET_KEY = os.getenv('SECRET_KEY')

session = boto3.Session(
    aws_access_key_id=ACCESS_KEY,
    aws_secret_access_key=SECRET_KEY
)
```

2.2 - Create s3 resource and client

```
s3 = session.resource('s3')
s3_client = session.client('s3')
```

2.3 - Working with s3 client

- client.list_buckets()
- client.list_objects()
- client.create_bucket()
- client.delete_bucket()
- client.put_object()
- client.copy()
- client.copy_object()
- client.delete_object()
- client.delete_objects()
- client.download_file()
- client.download_fileobj()
- upload_file()
- upload_fileobj()

create a new bucket ()

You need to change the `weclouddata-demo-bucket` to your own name.

If you remove the `CreateBucketConfiguration` line, it will use the "us-east-1" as default.

```
response = s3_client.create_bucket(
    Bucket='weclouddata-demo-bucket',
    CreateBucketConfiguration={'LocationConstraint': 'us-east-2'}
)

print(response)
```

list all buckets

```
for key in s3_client.list_buckets()['Buckets']:
    print(key['Name'])
```

list all objects (including folders) in a bucket/folder

```
objects = s3_client.list_objects(
    Bucket='weclouddata',
    Prefix='datasets/social'
)
```

```
for key in objects['Contents']:
    print(key['Key'])
```

Create a folder object in a bucket

```
response = s3_client.put_object(
    Bucket='weclouddata-demo-bucket',
    Key=('tmp/demo.csv'))

print(response)
```

List the folder

```
objects = s3_client.list_objects(
    Bucket='weclouddata-demo-bucket',
    Prefix='tmp'
)

for key in objects['Contents']:
    print(key['Key'])
```

Copy an object from one S3 location to another.

```
copy_source = {
    'Bucket': 'weclouddata-demo-bucket',
    'Key': 'tmp/demo.csv'
}
s3_client.copy(copy_source, 'weclouddata-demo-bucket', 'tmp1/demo1.csv')

objects = s3_client.list_objects(
    Bucket='weclouddata-demo-bucket',
    Prefix='tmp1'
)

for key in objects['Contents']:
    print(key['Key'])
```

Upload a file from local to s3

```
s3_client.upload_file('test.csv', 'weclouddata-demo-bucket', 'tmp/demo_local.csv')
```

Download from s3

```
s3_client.download_file('weclouddata-demo-bucket', 'tmp1/demo1.csv', 'demo1_s3.csv')
```

2.4 - Working with s3 session

- Bucket.copy()
- Bucket.create()
- Bucket.delete()
- Bucket.delete_objects()
- Bucket.download_file()
- Bucket.load()
- Bucket.put_object()
- Bucket.upload_file()
- Bucket.upload_fileobj()

list all objects in a bucket using resource

```
bucket = s3.Bucket('weclouddata-demo-bucket')
for obj in bucket.objects.all():
    print(obj.key)
```

upload a local file to s3 bucket

```
s3.Bucket('weclouddata-demo-bucket').Object('tmp/demo2.csv').upload_file('demo1_s3.csv')

for obj in bucket.objects.all():
    print(obj.key)
```

delete the bucket (need to delete all objects first)

```
bucket = s3.Bucket('weclouddata-demo-bucket')
bucket.delete()
```

delete all objects in a bucket

```
bucket = s3.Bucket('weclouddata-demo-bucket')
bucket.objects.all().delete()
bucket.delete()
```

3 - Working with EC2 via Boto3 (Optional)

Launch an ec2 instance

Required arguments:

- AMI ID
- Security Group Id
- VPC Subnet Id

```
import boto3
session = boto3.Session()
client = session.client('ec2', region_name='us-east-1')

response = client.run_instances(
    BlockDeviceMappings=[
        {
            'DeviceName': '/dev/xvda',
            'Ebs': {
                'DeleteOnTermination': True,
                'VolumeSize': 8,
                'VolumeType': 'gp2'
            },
        },
    ],
    ImageId='ami-0cd31be676780afa7',
    InstanceType='t2.micro',
    MaxCount=1,
    MinCount=1,
    Monitoring={
        'Enabled': False
    },
    SecurityGroupIds=[
        'sg-04331223e4cedde94',
    ],
    SubnetId='subnet-ed0db89b'
)
```

[Lecture Video] AWS Monday
Tuesday - Lambda
[Lecture Material] Lambda
[Lab] Mini Project: Lambda
Wednesday - Docker Basics
[Lecture Material] Docker Intro
[Quiz] Docker Commands Quiz
[Lab] Software Installation: Docker
[Lab] Exercise: Basic Docker Commands
[Lab] Workshop: Install Zepplin with Docker
Thursday - Docker Compose and Dockerfile
[Lecture Material] Docker Compose and dockerfile
[Lab] Mini Project: Dockerfile

[Lab] Workshop S3