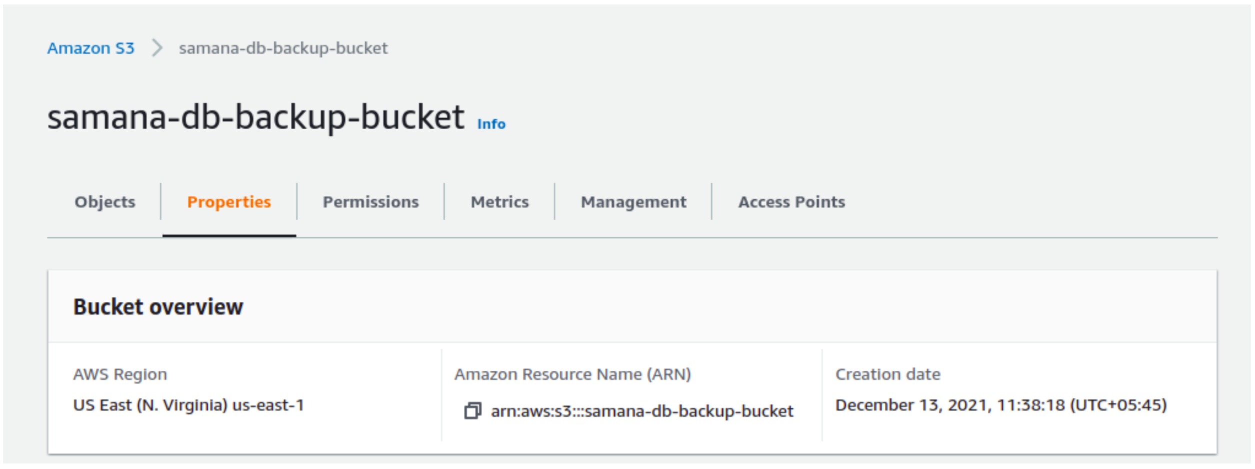
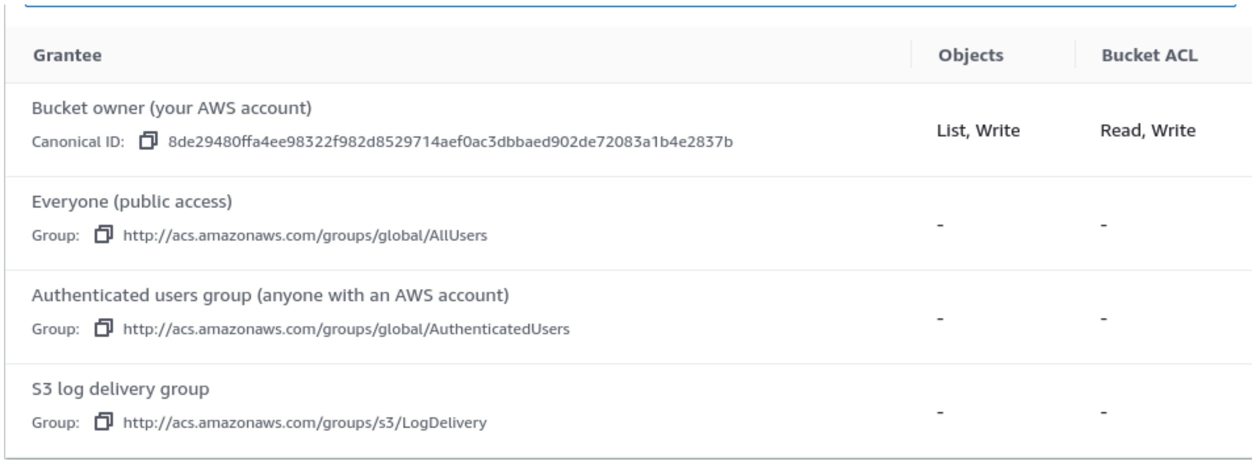
Write a script that backs up an SQL dump and uploads it to an S3 Bucket. The contents of the S3 bucket should not be accessible via public.

First of all we create a s3 bucket with no public access.


Amazon S3 > samana-db-backup-bucket
samana-db-backup-bucket Info
Objects
Properties
Permissions
Metrics
Management
Access Points

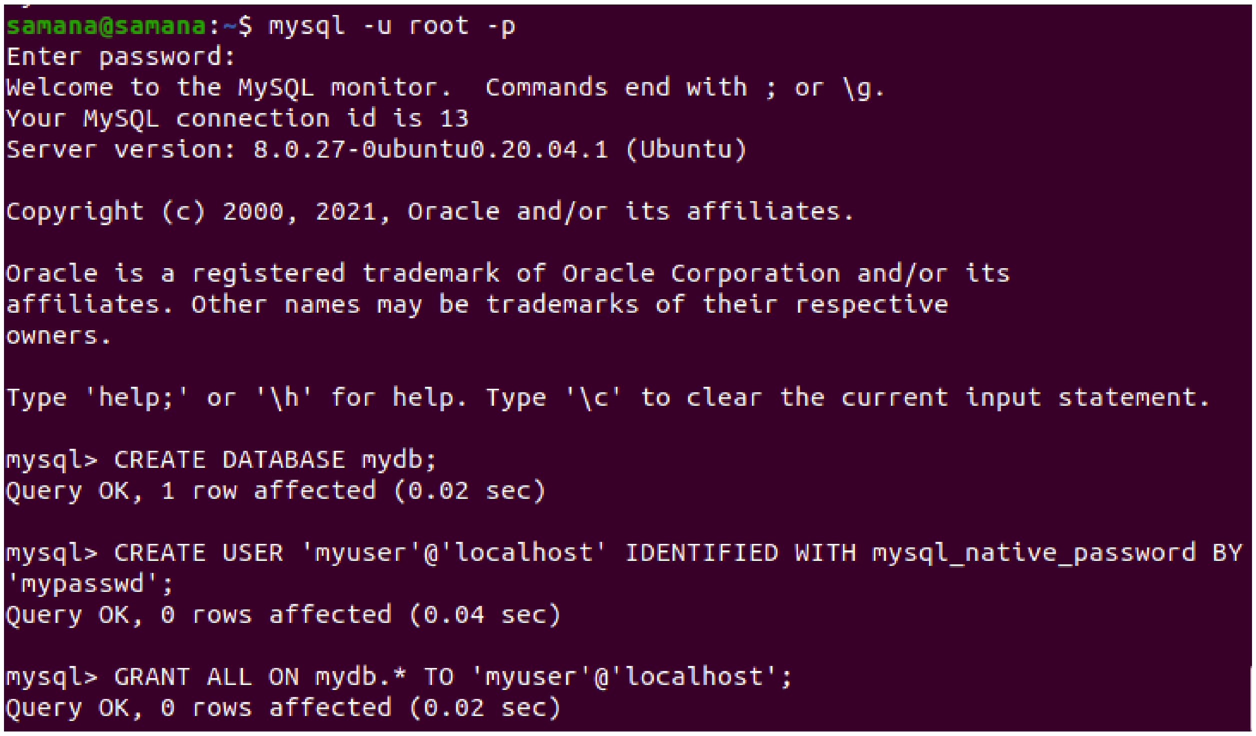
Bucket overview

AWS Region	Amazon Resource Name (ARN)	Creation date
US East (N. Virginia) us-east-1	arn:aws:s3:::samana-db-backup-bucket	December 13, 2021, 11:38:18 (UTC+05:45)



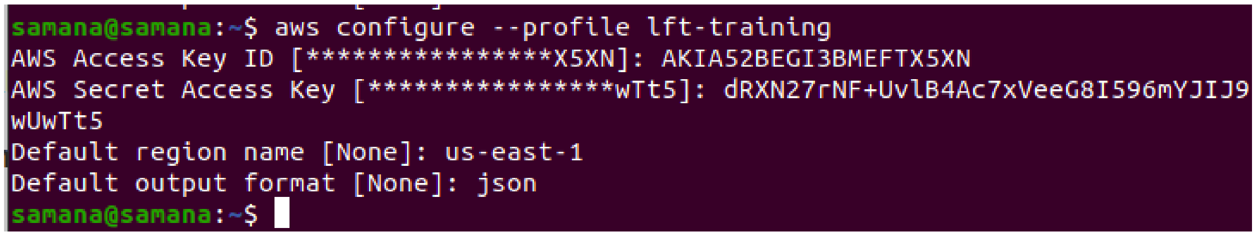
Grantee	Objects	Bucket ACL
Bucket owner (your AWS account) Canonical ID: 8de29480ffa4ee98322f982d8529714aef0ac3dbbaed902de72083a1b4e2837b	List, Write	Read, Write
Everyone (public access) Group: http://acs.amazonaws.com/groups/global/AllUsers	-	-
Authenticated users group (anyone with an AWS account) Group: http://acs.amazonaws.com/groups/global/AuthenticatedUsers	-	-
S3 log delivery group Group: http://acs.amazonaws.com/groups/s3/LogDelivery	-	-

Now, we create a database and user for that database.



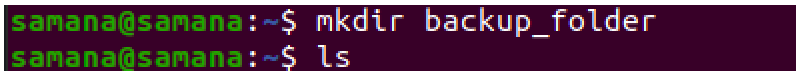
```
samana@samana:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.27-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE mydb;
Query OK, 1 row affected (0.02 sec)

mysql> CREATE USER 'myuser'@'localhost' IDENTIFIED WITH mysql_native_password BY
'mypasswd';
Query OK, 0 rows affected (0.04 sec)

mysql> GRANT ALL ON mydb.* TO 'myuser'@'localhost';
Query OK, 0 rows affected (0.02 sec)
```

Now we configure the aws CLI.



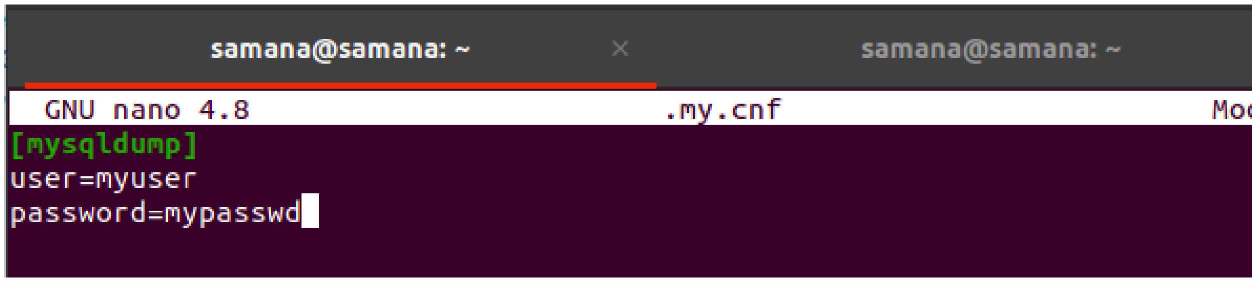
```
samana@samana:~$ aws configure --profile lft-training
AWS Access Key ID [****************X5XN]: AKIA52BEGI3BMEFTX5XN
AWS Secret Access Key [****************wTt5]: dRXN27rNF+UvlB4Ac7xVeeG8I596mYJIJ9
wUwTt5
Default region name [None]: us-east-1
Default output format [None]: json
samana@samana:~$
```

Now, we create a folder for backing up the mysql dumps.



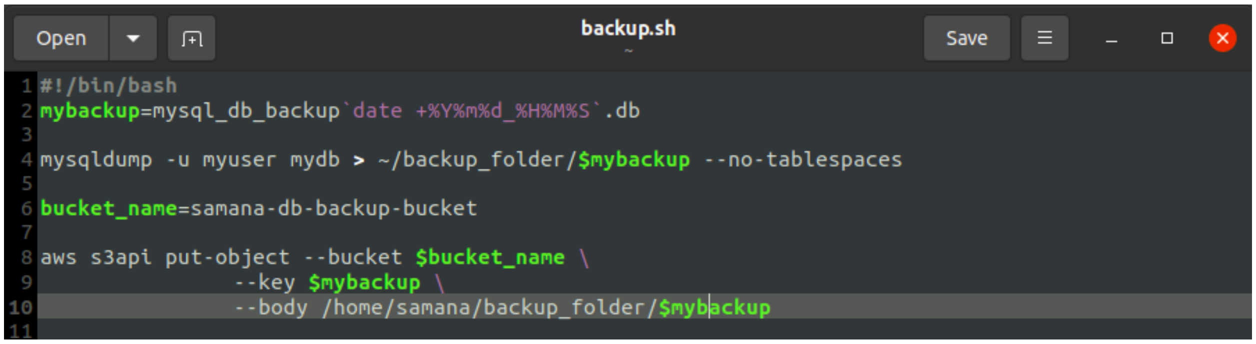
```
samana@samana:~$ mkdir backup_folder
samana@samana:~$ ls
```

We store the database username and password for mysqldump in .my.cnf



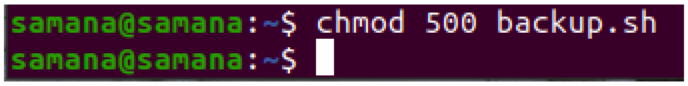
```
  GNU nano 4.8                        .my.cnf                            Mo
[mysqldump]
user=myuser
password=mypasswd
```

In order to upload the sql dumps in s3 bucket, we write a script as follows:



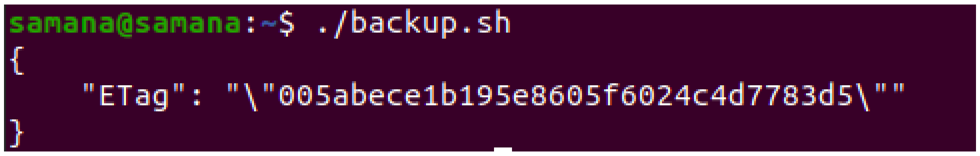
```
#!/bin/bash
mybackup=mysql_db_backup`date +%Y%m%d_%H%M%S`.db

mysqldump -u myuser mydb > ~/backup_folder/$mybackup --no-tablespaces

bucket_name=samana-db-backup-bucket

aws s3api put-object --bucket $bucket_name \
                --key $mybackup \
                --body /home/samana/backup_folder/$mybackup
```

To make the script executable we give read and execute permission to the script.



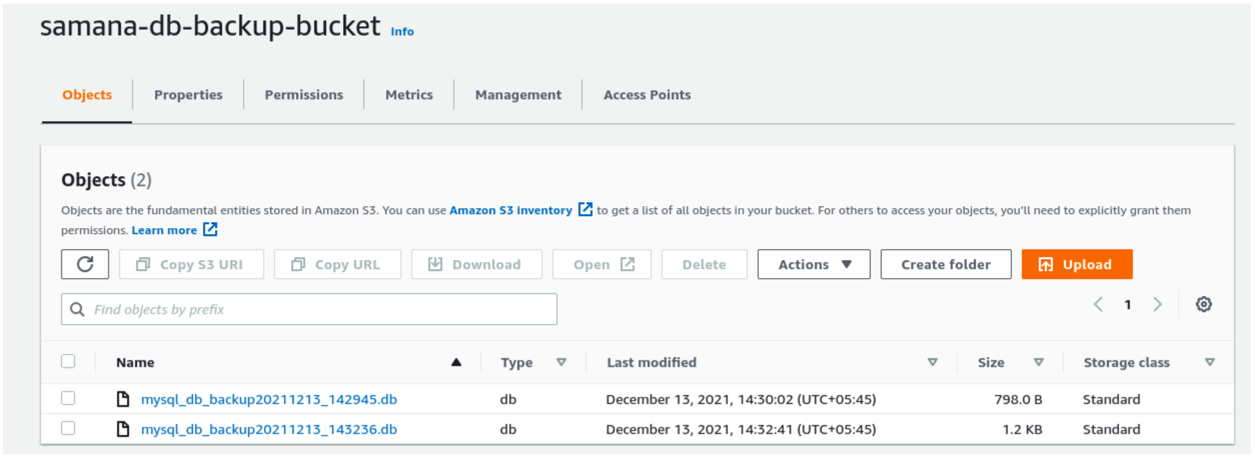
```
samana@samana:~$ chmod 500 backup.sh
samana@samana:~$
```

We run the script as follows:



```
samana@samana:~$ ./backup.sh
{
    "ETag": "\"005abece1b195e8605f6024c4d7783d5\""
}
```

We can see that in response to our script execution, objects are uploaded in our s3 bucket



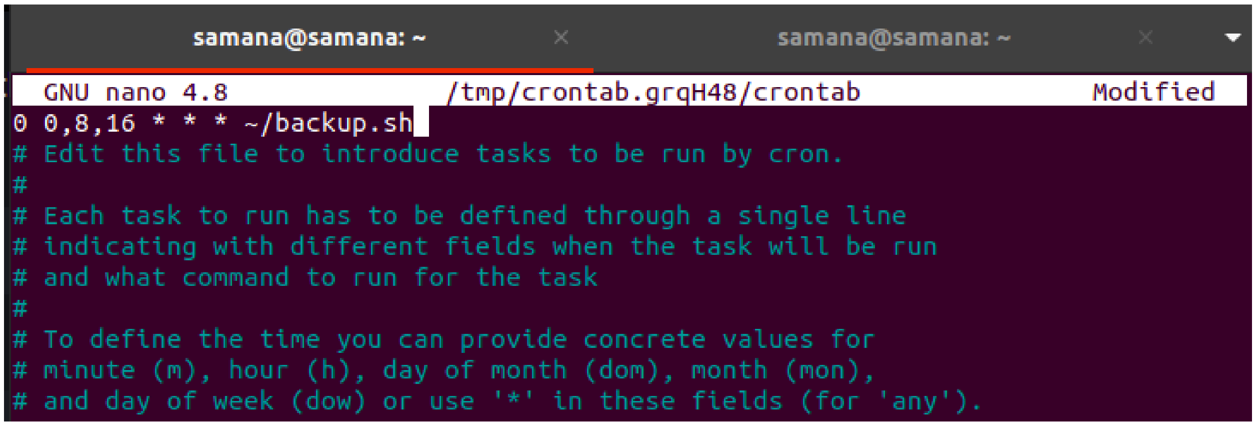
samana-db-backup-bucket Info

Objects (2)

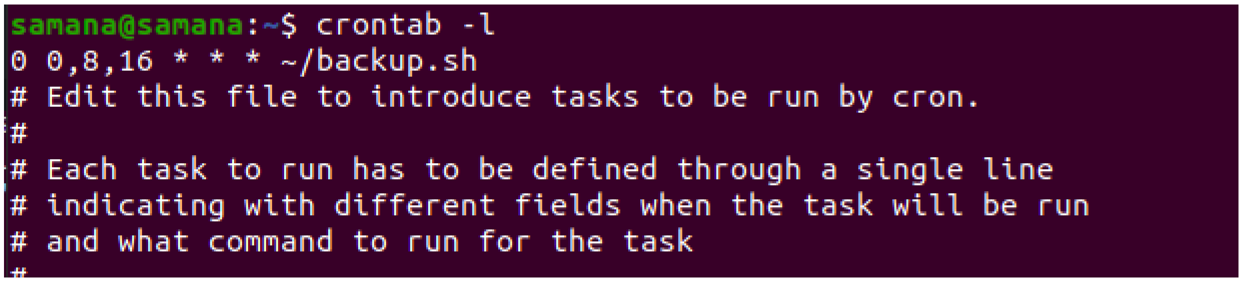
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
mysql_db_backup20211213_142945.db	db	December 13, 2021, 14:30:02 (UTC+05:45)	798.0 B	Standard
mysql_db_backup20211213_143236.db	db	December 13, 2021, 14:32:41 (UTC+05:45)	1.2 KB	Standard

Now in order to automate the backup and upload process we write a cronjob as follows:
Where 0 stands for 0th minute, 0,8 and 16 mean the script would run at 12 am 8 am and 4 pm everyday.



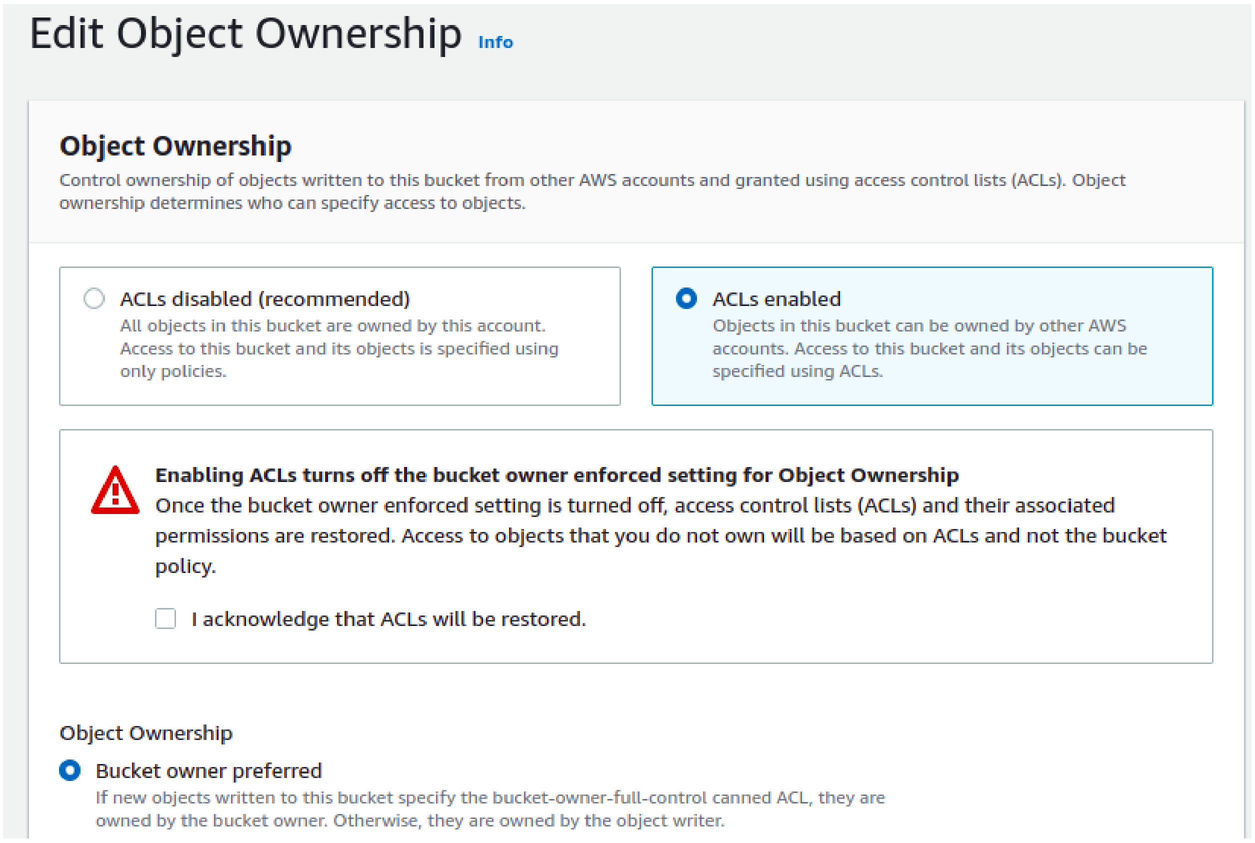
```
  GNU nano 4.8              /tmp/crontab.grqH48/crontab                Modified
0 0,8,16 * * * ~/backup.sh
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
```



```
samana@samana:~$ crontab -l
0 0,8,16 * * * ~/backup.sh
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
```

Create a Lambda function that is triggered by an object being uploaded to an S3 bucket. If the object's name starts with make_public, ensure that the object is publicly accessible.

First of all we edit the object ownership to enable ACLs



Edit Object Ownership Info

Object Ownership

Control ownership of objects written to this bucket from other AWS accounts and granted using access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (recommended) [✓ ACLs enabled]

ACLs disabled (recommended): All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled: Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

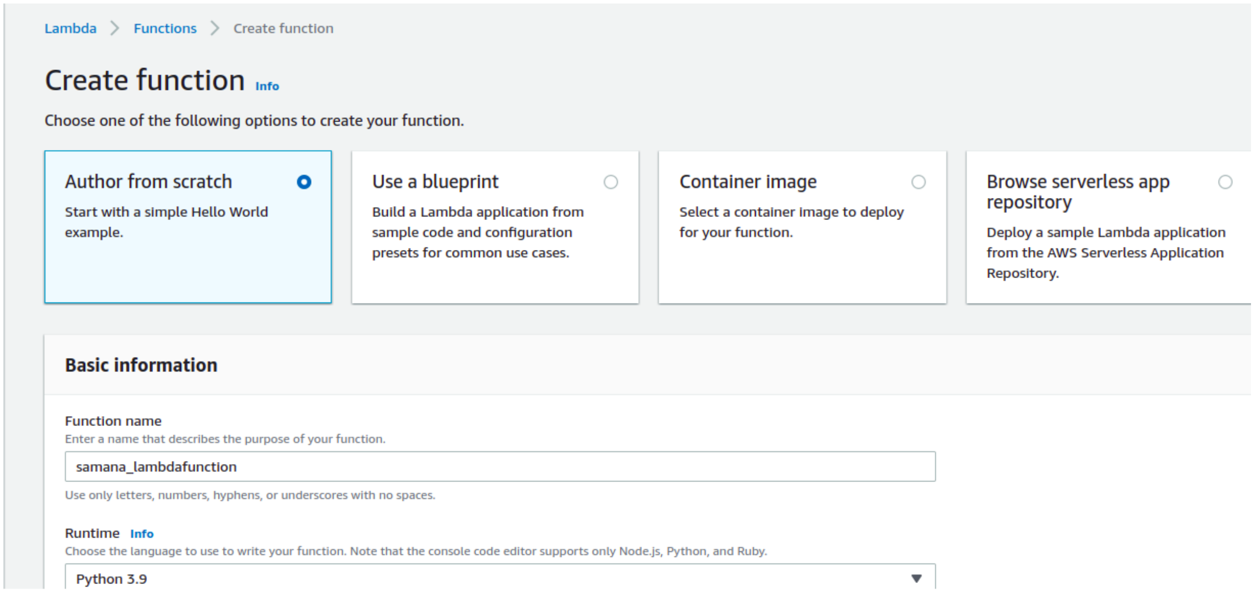
Enabling ACLs turns off the bucket owner enforced setting for Object Ownership
Once the bucket owner enforced setting is turned off, access control lists (ACLs) and their associated permissions are restored. Access to objects that you do not own will be based on ACLs and not the bucket policy.

☐ I acknowledge that ACLs will be restored.

Object Ownership

[✓ Bucket owner preferred]
If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.

Now, we create a lambda function as follows:



Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

[✓ Author from scratch] Use a blueprint | Container image | Browse serverless app repository

Author from scratch: Start with a simple Hello World example.

Use a blueprint: Build a Lambda application from sample code and configuration presets for common use cases.

Container image: Select a container image to deploy for your function.

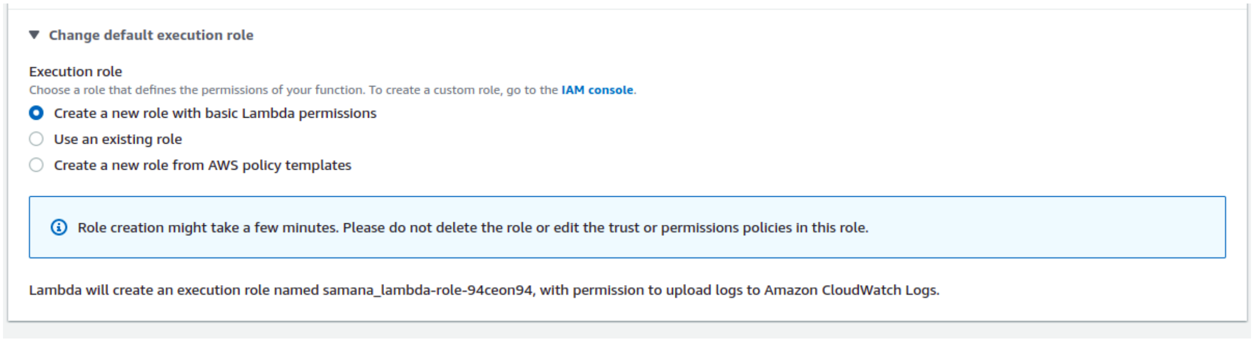
Browse serverless app repository: Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

Function name
Enter a name that describes the purpose of your function.
samana_lambdafunction
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.9

We create a new role with basic lambda permissions



▼ Change default execution role

Execution role

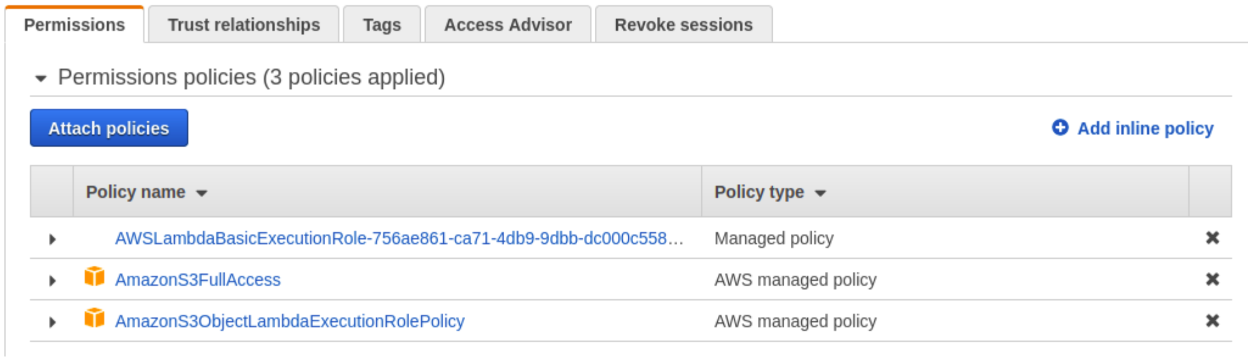
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

- [✓ Create a new role with basic Lambda permissions]
- Use an existing role
- Create a new role from AWS policy templates

ⓘ Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Lambda will create an execution role named samana_lambda-role-94ceon94, with permission to upload logs to Amazon CloudWatch Logs.

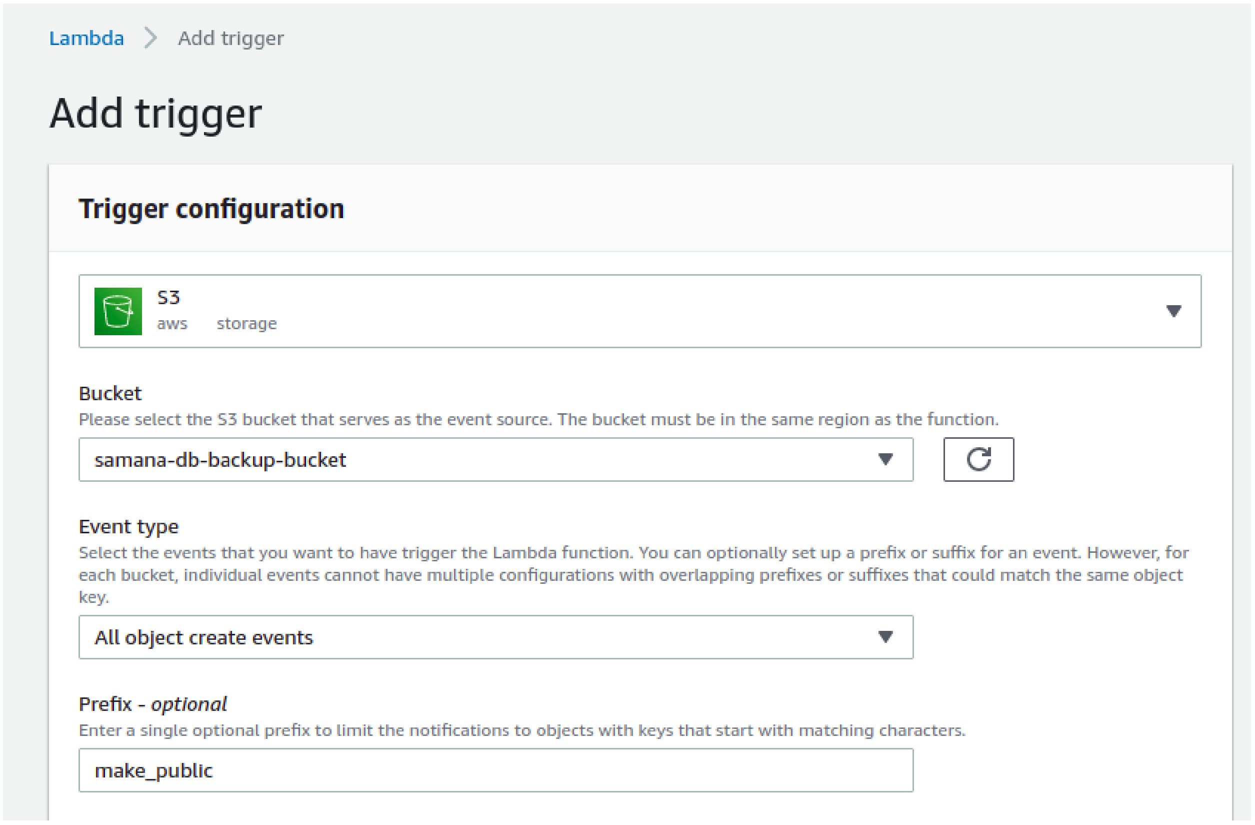
We then add other permissions as follows:



Permissions policies (3 policies applied)

Policy name	Policy type
AWSLambdaBasicExecutionRole-756ae861-ca71-4db9-9dbb-dc000c558…	Managed policy
AmazonS3FullAccess	AWS managed policy
AmazonS3ObjectLambdaExecutionRolePolicy	AWS managed policy

In the lambda function we create a trigger for our s3 bucket with prefix make_Public



Add trigger

Trigger configuration

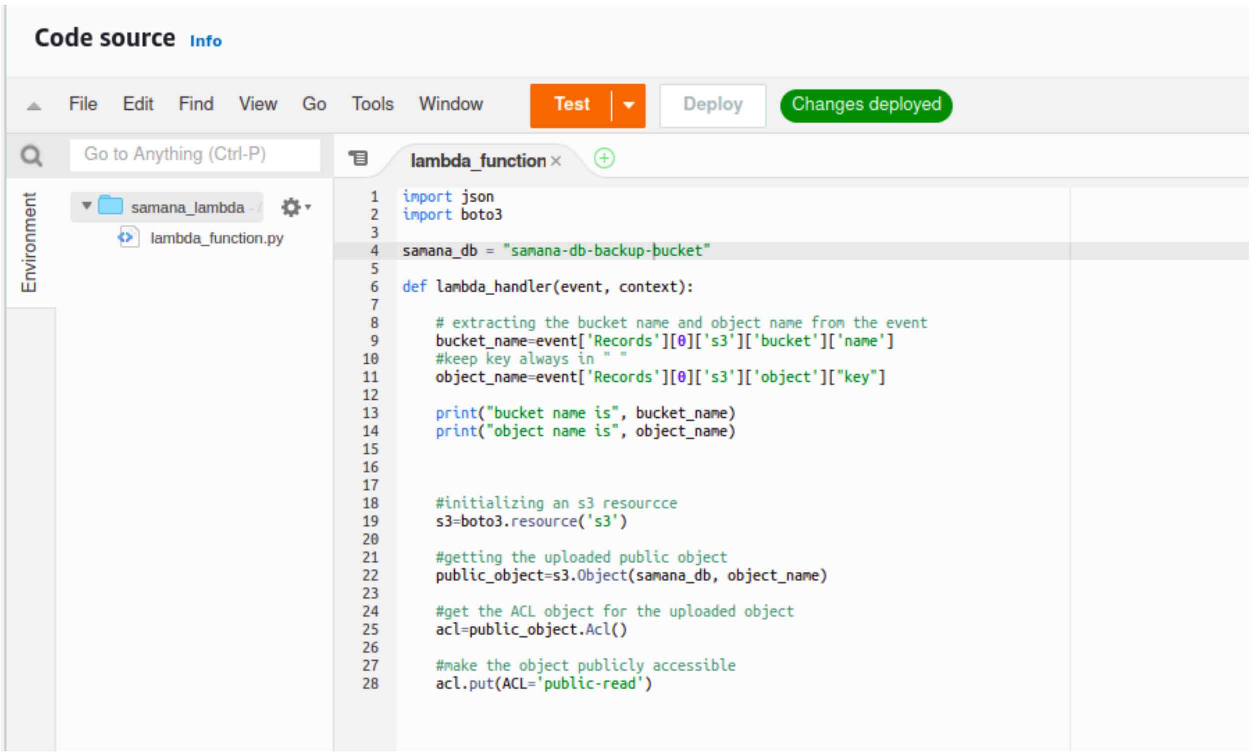
S3
aws storage

Bucket
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.
samana-db-backup-bucket

Event type
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.
All object create events

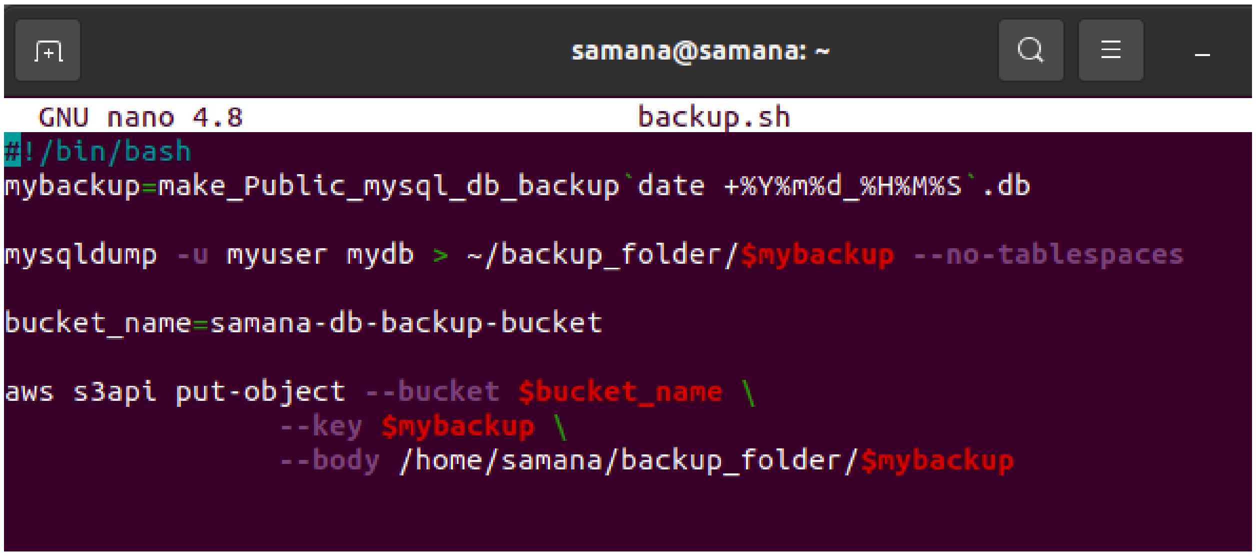
Prefix - *optional*
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.
make_public

We write the lambda function as follows.



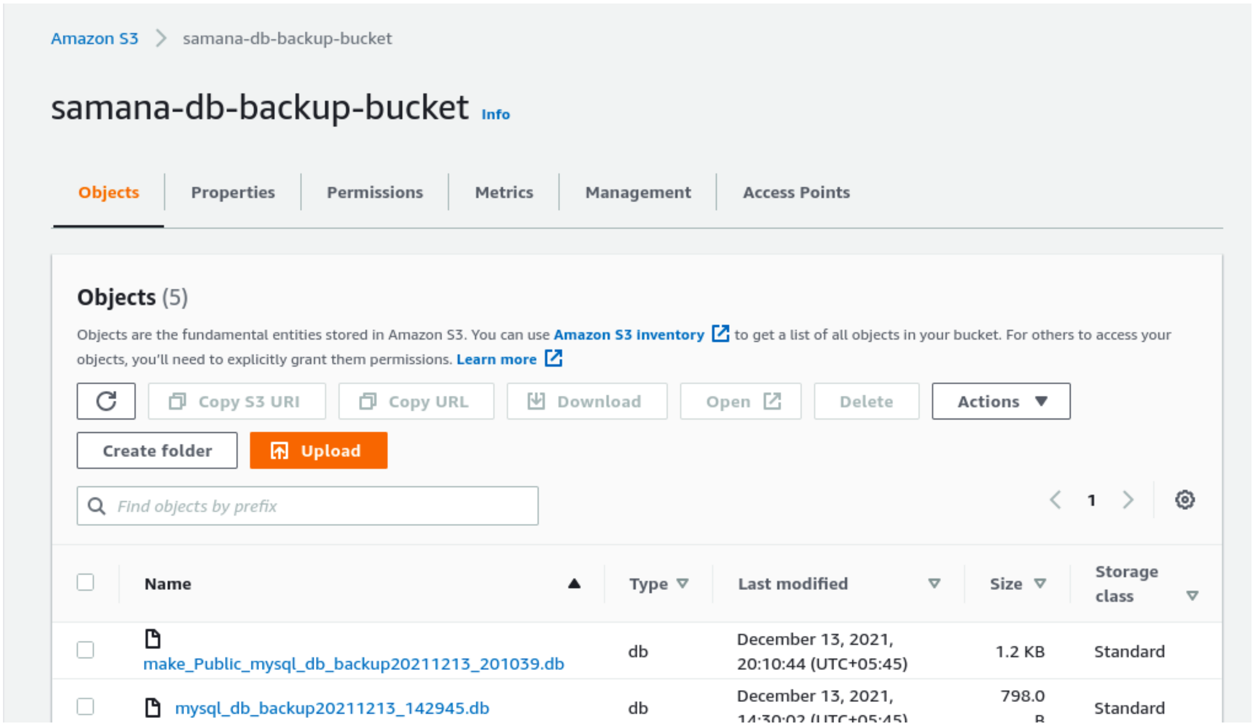
```
import json
import boto3

samana_db = "samana-db-backup-bucket"

def lambda_handler(event, context):

    # extracting the bucket name and object name from the event
    bucket_name=event['Records'][0]['s3']['bucket']['name']
    #keep key always in " "
    object_name=event['Records'][0]['s3']['object']["key"]

    print("bucket name is", bucket_name)
    print("object name is", object_name)



    #initializing an s3 resourcce
    s3=boto3.resource('s3')

    #getting the uploaded public object
    public_object=s3.Object(samana_db, object_name)

    #get the ACL object for the uploaded object
    acl=public_object.Acl()

    #make the object publicly accessible
    acl.put(ACL='public-read')
```

Now we edit our previously created script with prefix make_Public to trigger the lambda function



```
#!/bin/bash
mybackup=make_Public_mysql_db_backup`date +%Y%m%d_%H%M%S`.db

mysqldump -u myuser mydb > ~/backup_folder/$mybackup --no-tablespaces

bucket_name=samana-db-backup-bucket

aws s3api put-object --bucket $bucket_name \
                --key $mybackup \
                --body /home/samana/backup_folder/$mybackup
```

After executing the above script, we can see object with make_Public prefix as follows:

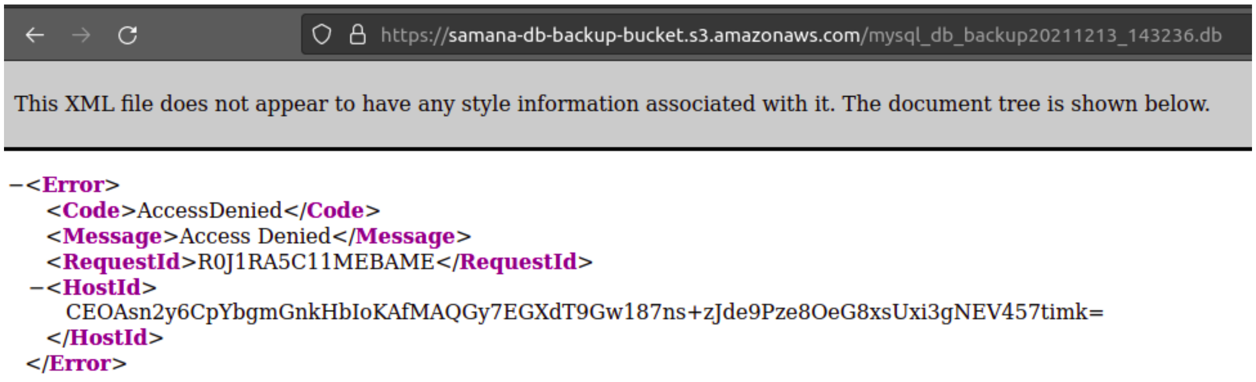


Objects (5)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
make_Public_mysql_db_backup20211213_201039.db	db	December 13, 2021, 20:10:44 (UTC+05:45)	1.2 KB	Standard
mysql_db_backup20211213_142945.db	db	December 13, 2021, 14:30:02 (UTC+05:45)	798.0 B	Standard

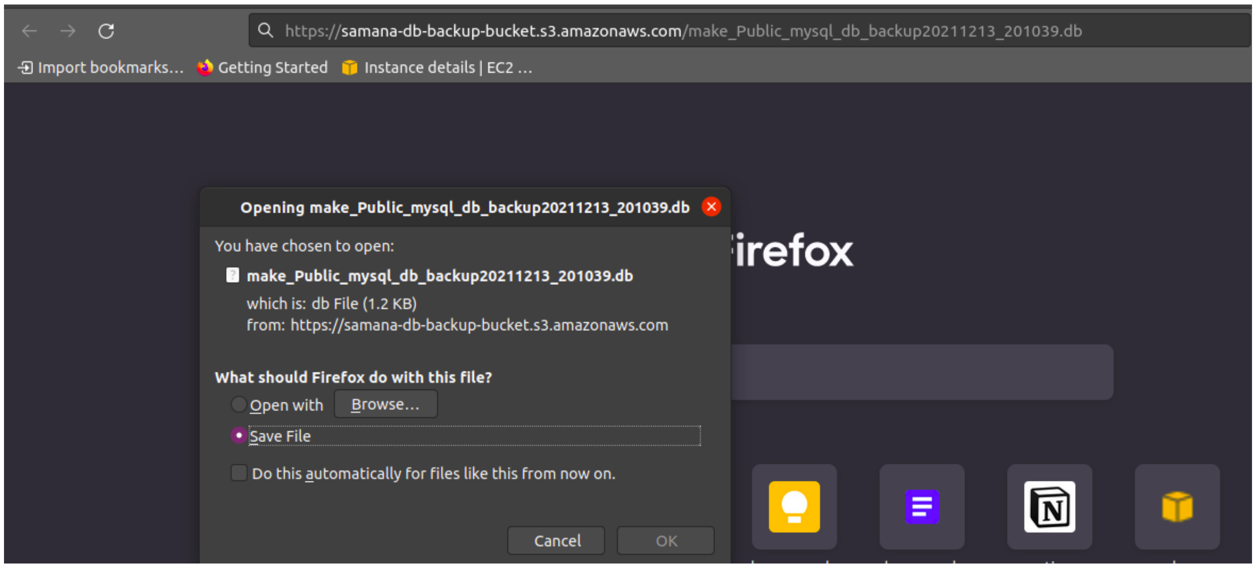
Previously our object was not publicly accessible


https://samana-db-backup-bucket.s3.amazonaws.com/mysql_db_backup20211213_143236.db

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
–<Error>
   <Code>AccessDenied</Code>
   <Message>Access Denied</Message>
   <RequestId>R0J1RA5C11MEBAME</RequestId>
 –<HostId>
    CEOAsn2y6CpYbgmGnkHbIoKAfMAQGy7EGXdT9Gw187ns+zJde9Pze8OeG8xsUxi3gNEV457timk=
   </HostId>
 </Error>
```

The object with make_Public prefix was found to be publicly accessible


https://samana-db-backup-bucket.s3.amazonaws.com/make_Public_mysql_db_backup20211213_201039.db

Opening make_Public_mysql_db_backup20211213_201039.db

You have chosen to open:

make_Public_mysql_db_backup20211213_201039.db

which is: db File (1.2 KB)
from: https://samana-db-backup-bucket.s3.amazonaws.com

What should Firefox do with this file?

Open with [Browse...]

[✓ Save File]

Do this automatically for files like this from now on.