

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.

A script is made to backup our mysql database:

### **Nano backup.sh**

```
GNU nano 4.8 backup.sh
NOW=$(date +%Y-%m-%d)
NOW_TIME=$(date +%Y-%m-%d %T %p)
NOW_MONTH=$(date +%Y-%m)

MYSQL_HOST="localhost"
MYSQL_PORT="3306"
MYSQL_DATABASE="zabbixdb"
MYSQL_USER="zabbix"
MYSQL_PASSWORD="password"

BACKUP_DIR="/home/bj/aws/$NOW_MONTH"
BACKUP_FULL_PATH="$BACKUP_DIR/$MYSQL_DATABASE-$NOW.sql.gz"
AMAZON_S3_BUCKET="s3://Intern-bijaykandel37/backup/$NOW_MONTH/"

mkdir -p ${BACKUP_DIR}

backup_mysql(){
    mysqldump -h ${MYSQL_HOST} \
        -P ${MYSQL_PORT} \
        -u ${MYSQL_USER} \
        -p${MYSQL_PASSWORD} ${MYSQL_DATABASE} --no-tablespaces | gzip > ${BACKUP_FULL_PATH}
}

upload_s3(){
    aws s3 cp ${BACKUP_FULL_PATH} ${AMAZON_S3_BUCKET}
}

backup_mysql
upload_s3
```



To schedule this script 3 times a day, ie. every 8 hours,

**‘Crontab -e’** command is used and this part is appended at the end of the file:

```
* * * /8 * * /path/to/script
```

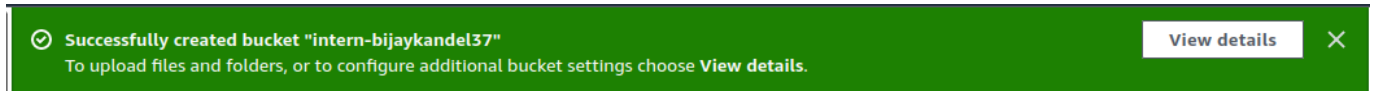
```
# m h dom mon dow    command
* * * /8 * * /home/bj/aws/backup.sh > /dev/null 2>&1
```

It can be verified by viewing our s3 bucket.

<input type="checkbox"/>	Name ▲	Type ▼	Last modified ▼	Size ▼	Storage class ▼
<input type="checkbox"/>	 zabbixdb-2021-12-11.sql.gz	gz	December 11, 2021, 18:43:29 (UTC+05:45)	3.9 MB	Standard
<input type="checkbox"/>	 zabbixdb-2021-12-12.sql.gz	gz	December 12, 2021, 14:59:51 (UTC+05:45)	3.9 MB	Standard

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.

To create a lambda function, we must already have a s3 bucket,  
I created a bucket with name intern-bijaykandel37

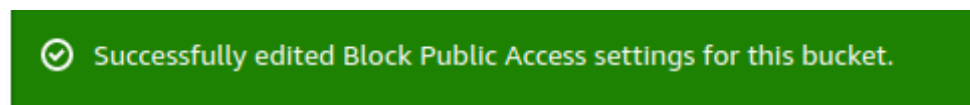


And created a lambda function,

The screenshot shows the "Create function" page in the AWS Lambda console. At the top, there's a breadcrumb trail: "Lambda > Functions > Create function". The main heading is "Create function" with an "Info" link. Below it, a prompt says "Choose one of the following options to create your function." There are three options: "Author from scratch" (selected with a blue radio button), "Use a blueprint", and "Container image". The "Author from scratch" option has a subtext: "Start with a simple Hello World example." Below these options is a section titled "Basic information". It contains three fields: "Function name" with the value "intern-bijaykandel37", "Runtime" set to "Python 3.9", and "Architecture" set to "x86\_64".

And access of the s3 bucket was edited:

The **Block all Public Access** was unchecked.



Under Edit Object ownership, ACLs are enabled:

Amazon S3

Intern-bijaykandel37

Edit Object Ownership

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)

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.

Permission to AmazonS3FullAccess is given to intern-bijaykandel37

Add permissions to intern-bijaykandel37-role-j7bkfa7g

Attach Permissions

Create policy

Filter policies

Q s3

Showing 11 results

	Policy name	Type	Used as
<input type="checkbox"/>	AmazonDMSRedshiftS3Role	AWS managed	None
<input checked="" type="checkbox"/>	AmazonS3FullAccess	AWS managed	Permissions policy (7)
<input type="checkbox"/>	AmazonS3ObjectLambdaExecutionRolePolicy	AWS managed	Permissions policy (2)
<input type="checkbox"/>	AmazonS3OutpostsFullAccess	AWS managed	None
<input type="checkbox"/>	AmazonS3OutpostsReadOnlyAccess	AWS managed	None
<input type="checkbox"/>	AmazonS3ReadOnlyAccess	AWS managed	Permissions policy (3)

Cancel

Attach policy

Attach policies

Add inline policy

Policy name	Policy type	
AWSLambdaBasicExecutionRole-0ac91402-9733-4066-9ad8-f5e9f07d952c	Managed policy	✕
AmazonS3FullAccess	AWS managed policy	✕

A simple test event is configured:

### Configure test event

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

☒ Create new test event

☐ Edit saved test events

Event template

s3-put

Event name

MyEventName

```
1 {
2   "Records": [
3     {
4       "eventVersion": "2.0",
5       "eventSource": "aws:s3",
6       "awsRegion": "us-east-1",
7       "eventTime": "1970-01-01T00:00:00.000Z",
8       "eventName": "ObjectCreated:Put",
```

And then add an s3 trigger by clicking on '+ 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.

intern-bijaykandel37

**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.

PUT


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

make\_public

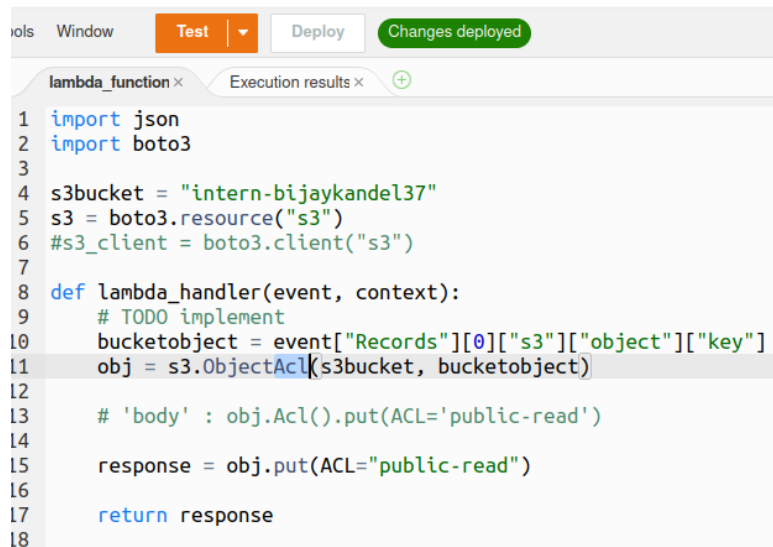
After the trigger is added, it may show like this:

### Triggers (1)

Find triggers

Trigger
<div> <b>S3: intern-bijaykandel37</b> arn:aws:s3:::intern-bijaykandel37</div> <div><p>▼ Details</p><p>Bucket: s3/intern-bijaykandel37</p><p>Event type: ObjectCreatedByPut</p><p>Notification name: 61d22e89-3f5f-459e-a4ec-de2a8f503c87</p><p>Prefix: make_public</p></div>

Now a lambda function is written:

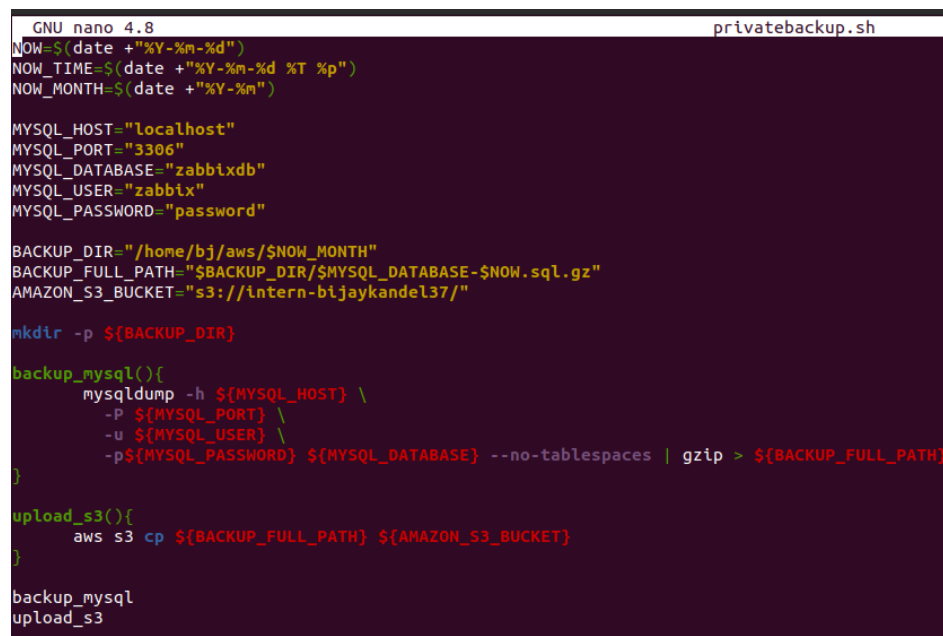


```
1 import json
2 import boto3
3
4 s3bucket = "intern-bijaykandel37"
5 s3 = boto3.resource("s3")
6 #s3_client = boto3.client("s3")
7
8 def lambda_handler(event, context):
9     # TODO implement
10    bucketobject = event["Records"][0]["s3"]["object"]["key"]
11    obj = s3.ObjectAcl(s3bucket, bucketobject)
12
13    # 'body' : obj.Acl().put(ACL='public-read')
14
15    response = obj.put(ACL="public-read")
16
17    return response
18
```

This lambda function is triggered when an object with make\_public prefix is uploaded to the s3 bucket which makes the uploaded content public.

Now, we created two backup scripts, each for private and public access.

This is script for private access:



```
GNU nano 4.8 privatebackup.sh
NOW=$(date +%Y-%m-%d)
NOW_TIME=$(date +%Y-%m-%d %T %p)
NOW_MONTH=$(date +%Y-%m)

MYSQL_HOST="localhost"
MYSQL_PORT="3306"
MYSQL_DATABASE="zabbixdb"
MYSQL_USER="zabbix"
MYSQL_PASSWORD="password"

BACKUP_DIR="/home/bj/aws/$NOW_MONTH"
BACKUP_FULL_PATH="$BACKUP_DIR/$MYSQL_DATABASE-$NOW.sql.gz"
AMAZON_S3_BUCKET="s3://intern-bijaykandel37/"

mkdir -p ${BACKUP_DIR}

backup_mysql(){
    mysqldump -h ${MYSQL_HOST} \
        -P ${MYSQL_PORT} \
        -u ${MYSQL_USER} \
        -p${MYSQL_PASSWORD} ${MYSQL_DATABASE} --no-tablespaces | gzip > ${BACKUP_FULL_PATH}
}

upload_s3(){
    aws s3 cp ${BACKUP_FULL_PATH} ${AMAZON_S3_BUCKET}
}

backup_mysql
upload_s3
```

And below is the script for public access, which can be seen in BACKUP\_FULL\_PATH with make\_public prefix.

```

GNU nano 4.8 publicbackup.sh
NOW=$(date +%Y-%m-%d)
NOW_TIME=$(date +%Y-%m-%d %T %p)
NOW_MONTH=$(date +%Y-%m)

MYSQL_HOST="localhost"
MYSQL_PORT="3306"
MYSQL_DATABASE="zabbixdb"
MYSQL_USER="zabbix"
MYSQL_PASSWORD="password"

BACKUP_DIR="/home/bj/aws/$NOW_MONTH"
BACKUP_FULL_PATH="$BACKUP_DIR/make_public_${MYSQL_DATABASE}-$NOW.sql.gz"
AMAZON_S3_BUCKET="s3://intern-bijaykandel37/"

mkdir -p ${BACKUP_DIR}

backup_mysql(){
  mysqldump -h ${MYSQL_HOST} \
    -P ${MYSQL_PORT} \
    -u ${MYSQL_USER} \
    -p${MYSQL_PASSWORD} ${MYSQL_DATABASE} --no-tablespaces | gzip > ${BACKUP_FULL_PATH}
}

upload_s3(){
  aws s3 cp ${BACKUP_FULL_PATH} ${AMAZON_S3_BUCKET}
}

backup_mysql
upload_s3

```

To run the scripts,

`./privatebackup.sh`

`./publicbackup.sh`

```

bj@batman:~/aws$ ./privatebackup.sh
mysqldump: [Warning] Using a password on the command line interface can be insecure.
upload: 2021-12/zabbixdb-2021-12-12.sql.gz to s3://intern-bijaykandel37/zabbixdb-2021-12-12.sql.gz
bj@batman:~/aws$ ./publicbackup.sh
mysqldump: [Warning] Using a password on the command line interface can be insecure.
upload: 2021-12/make_public_zabbixdb-2021-12-12.sql.gz to s3://intern-bijaykandel37/make_public_zabbixdb-2021-12-12.sql.gz
bj@batman:~/aws$

```

And in the s3 bucket, we can see the uploaded items:

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	backup/	Folder	-	-	-
<input type="checkbox"/>	make_public_zabbixdb-2021-12-12.sql.gz	gz	December 12, 2021, 22:44:20 (UTC+05:45)	3.9 MB	Standard
<input type="checkbox"/>	zabbixdb-2021-12-12.sql.gz	gz	December 12, 2021, 22:44:09 (UTC+05:45)	3.9 MB	Standard

One is with make\_public prefix which is uploaded from publicbackup.sh with Read permission to the public.

Amazon S3 > Intern-bijaykandel37 > make\_public\_zabbixdb-2021-12-12.sql.gz

### make\_public\_zabbixdb-2021-12-12.sql.gz [Info](#)

[Copy S3 URI](#) [Download](#) [Open](#) [Object actions](#)

Properties | **Permissions** | Versions

#### Access control list (ACL) [Edit](#)

Grant basic read/write permissions to AWS accounts. [Learn more](#)

Grantee	Object	Object ACL
Object owner (your AWS account) Canonical ID: <a href="#">8de29480ffa4ee98322f982d8529714aef0ac3dbbaed902de72083a1b4e2837b</a>	Read	Read, Write
Everyone (public access) Group: <a href="#">http://acs.amazonaws.com/groups/global/AllUsers</a>	Read	-
Authenticated users group (anyone with an AWS account)		

One is with no prefix which is uploaded from privatebackup.sh with no Read permission to the public.

Amazon S3 > Intern-bijaykandel37 > zabbixdb-2021-12-12.sql.gz

### zabbixdb-2021-12-12.sql.gz [Info](#)

[Copy S3 URI](#) [Download](#) [Open](#) [Object actions](#)

Properties | **Permissions** | Versions

#### Access control list (ACL) [Edit](#)

Grant basic read/write permissions to AWS accounts. [Learn more](#)

Grantee	Object	Object ACL
Object owner (your AWS account) Canonical ID: <a href="#">8de29480ffa4ee98322f982d8529714aef0ac3dbbaed902de72083a1b4e2837b</a>	Read	Read, Write
Everyone (public access) Group: <a href="#">http://acs.amazonaws.com/groups/global/AllUsers</a>	-	-
Authenticated users group (anyone with an AWS account)		