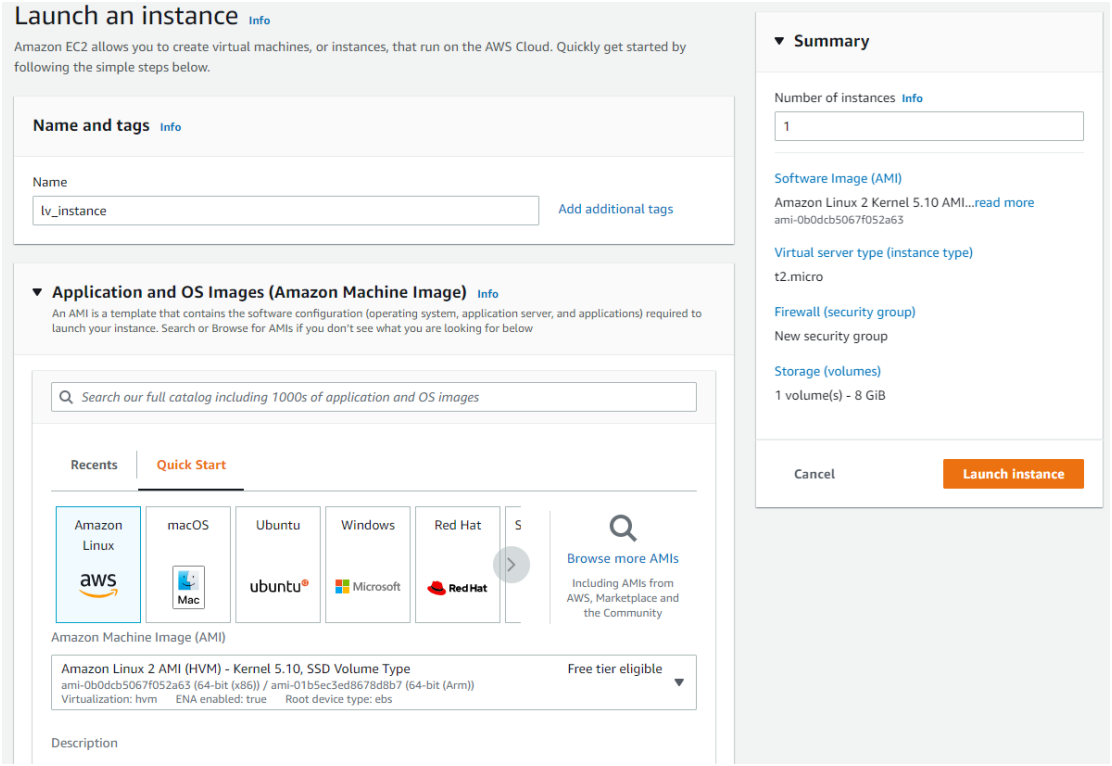
EPAM University Programs

Cloud&DevOps Fundamentals Autumn 2022

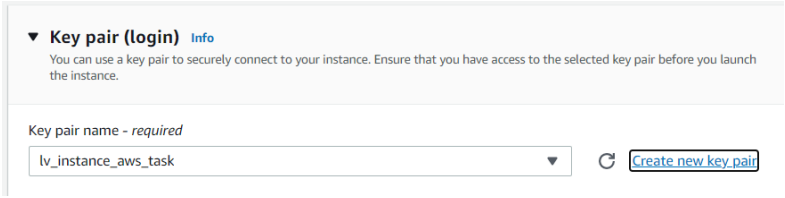
AWS Cloud Basic

1) Review Getting Started with Amazon EC2. Log Into Your AWS Account, Launch, Configure, Connect and Terminate your Instance.

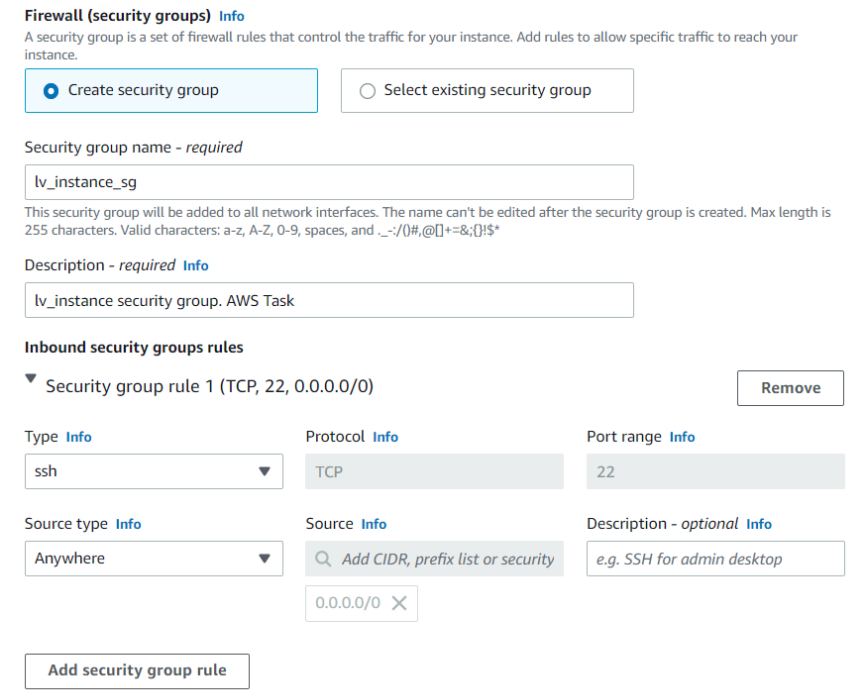
I create an instance (I choose Amazon Linux AMI):



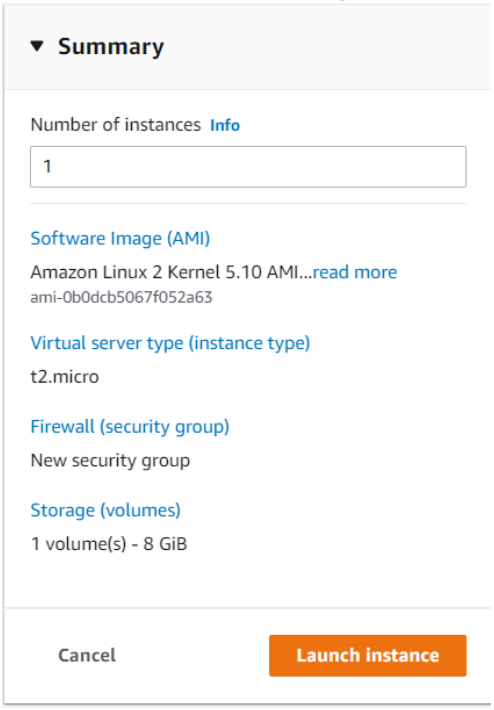
I create a key pair to connect to the instance via SSH:



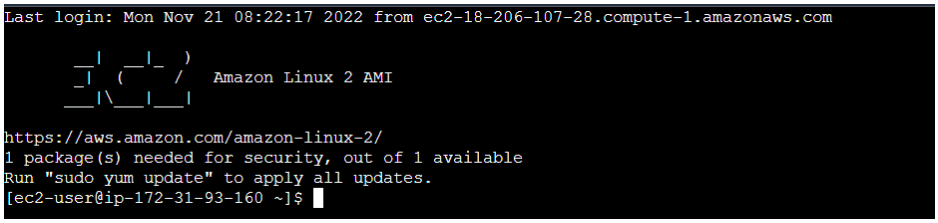
I configure a security group:



I leave the other settings as default and launch the instance:



I connect to the instance via SSH:

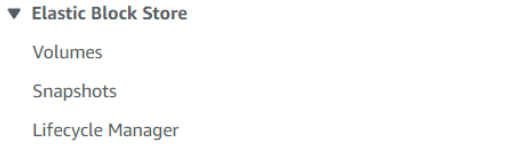


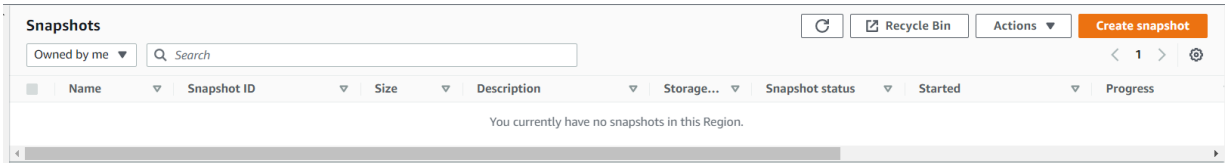
I terminate the instance:

2) **Create a snapshot of your instance to keep as a backup.**

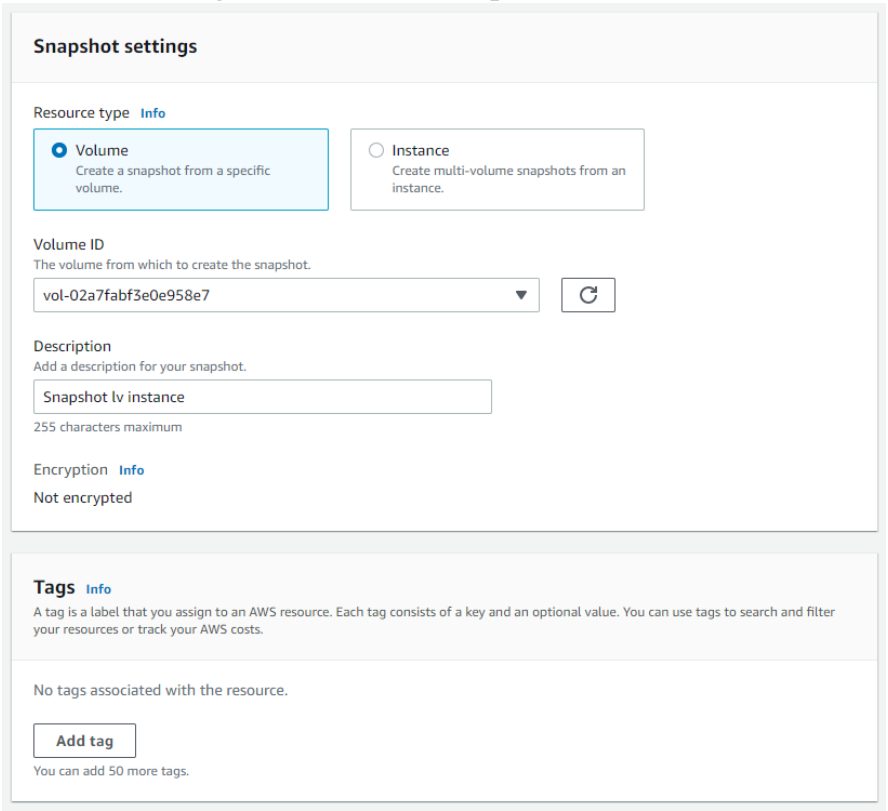
I launch a new instance with similar settings:

I choose «Snapshots» in the navigation pane:



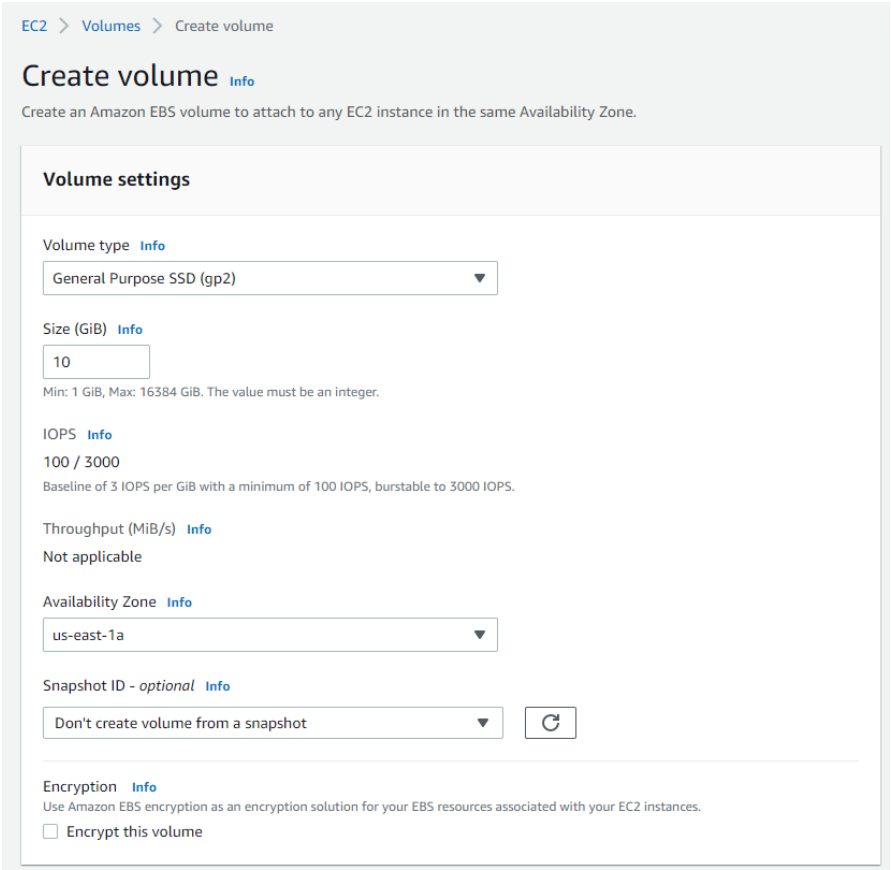


I set the settings and create a snapshot:

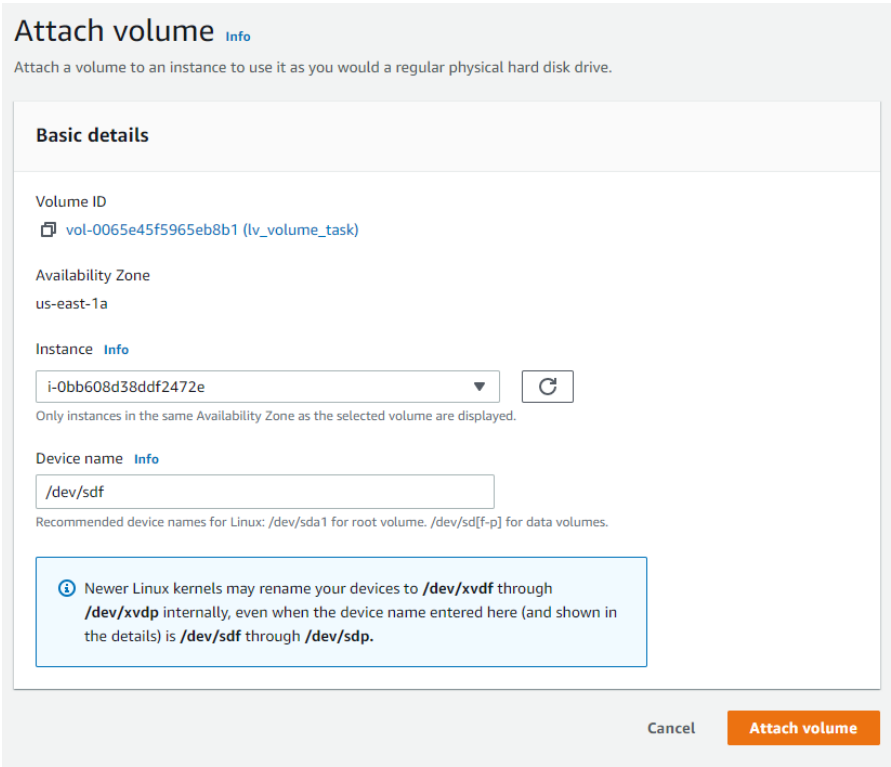


3) **Create and attach a Disk_D (EBS) to your instance to add more storage space. Create and save some file on Disk_D.**

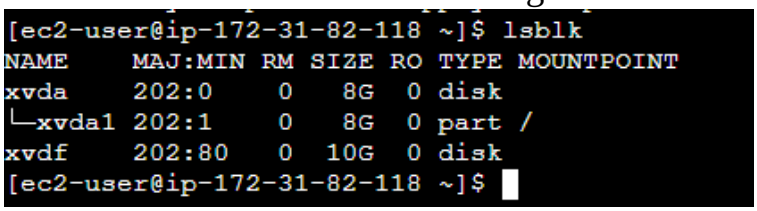
I create a new EBS volume:



I attach this volume to the created instance:



I check if the drive is attached using the terminal:



The volume is successfully attached (xvdf).

I format the volume with an ext4 file system:

```
[ec2-user@ip-172-31-82-118 ~]$ sudo mkfs.ext4 /dev/sdf
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
655360 inodes, 2621440 blocks
131072 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2151677952
80 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

I create a mount point:

```
[ec2-user@ip-172-31-82-118 ~]$ sudo mkdir /mnt/new_volume
[ec2-user@ip-172-31-82-118 ~]$
```

I configure fstab file:

```
#
UUID=1377e573-627c-46ee-b7ca-9b86138b39db     /           xfs    defaults,noatime  1   1
/dev/sdf        /mnt/new_volume         ext4    defaults        0       2
~
```

I mount the volume and check if it was mounted:

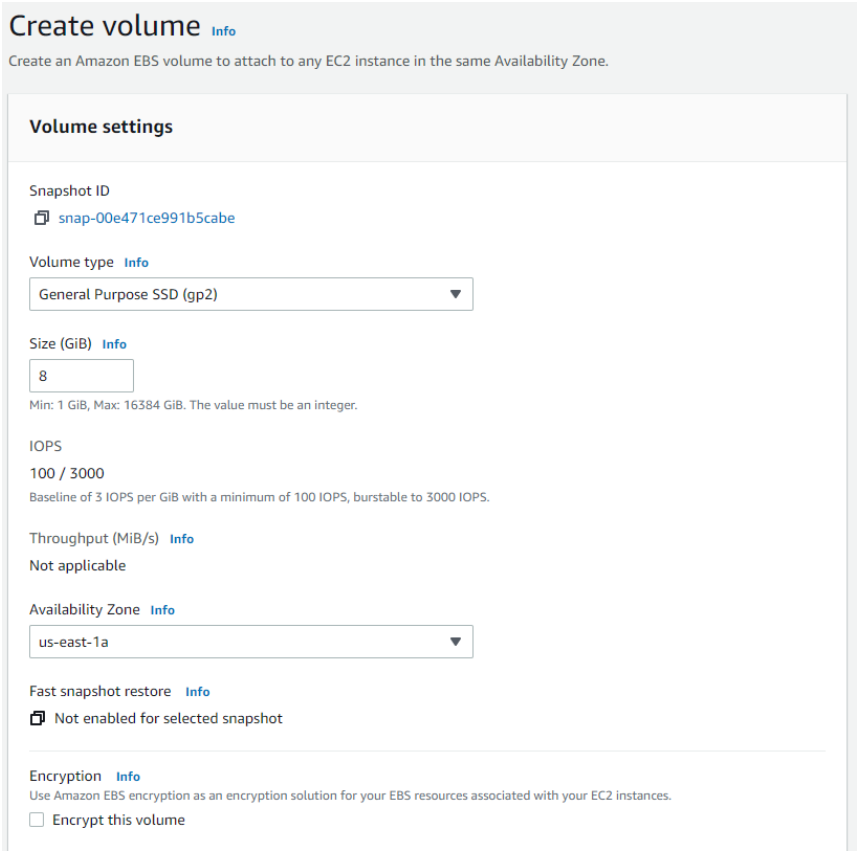
```
[ec2-user@ip-172-31-82-118 ~]$ sudo mount -a
[ec2-user@ip-172-31-82-118 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        474M     0  474M   0% /dev
tmpfs           483M     0  483M   0% /dev/shm
tmpfs           483M  468K  483M   1% /run
tmpfs           483M     0  483M   0% /sys/fs/cgroup
/dev/xvda1      8.0G  1.6G  6.5G  20% /
tmpfs            97M     0   97M   0% /run/user/0
tmpfs            97M     0   97M   0% /run/user/1000
/dev/xvdf       9.7G   24K  9.2G   1% /mnt/new_volume
```

I create some files on the drive:

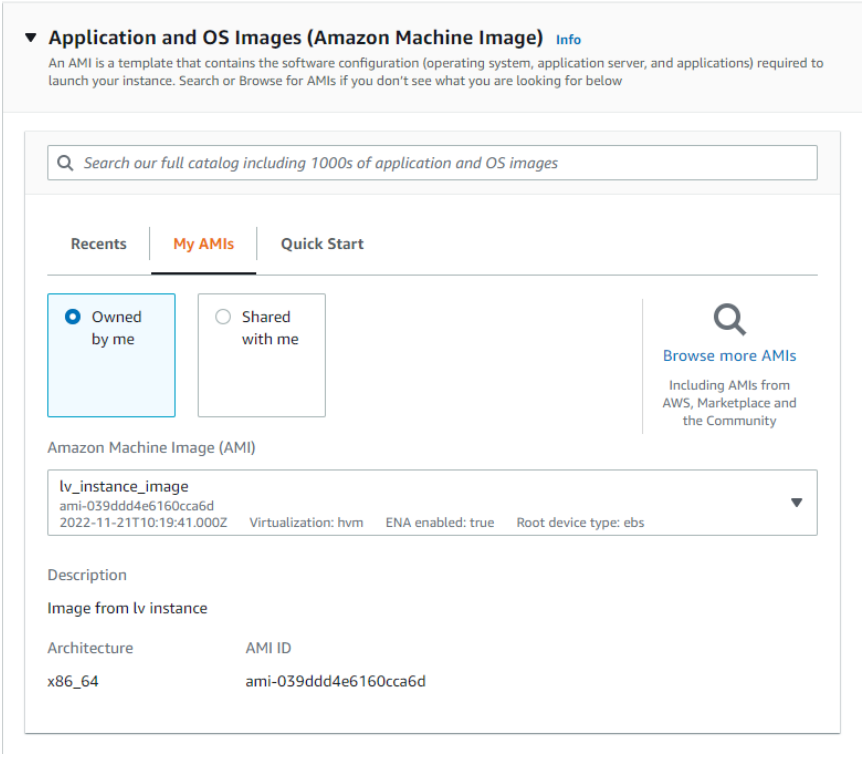
```
[ec2-user@ip-172-31-82-118 ~]$ sudo touch /mnt/new_volume/file{1..10}
[ec2-user@ip-172-31-82-118 ~]$ ls /mnt/new_volume/
file1  file10  file2  file3  file4  file5  file6  file7  file8  file9  lost+found
[ec2-user@ip-172-31-82-118 ~]$
```

4) Launch the second instance from backup

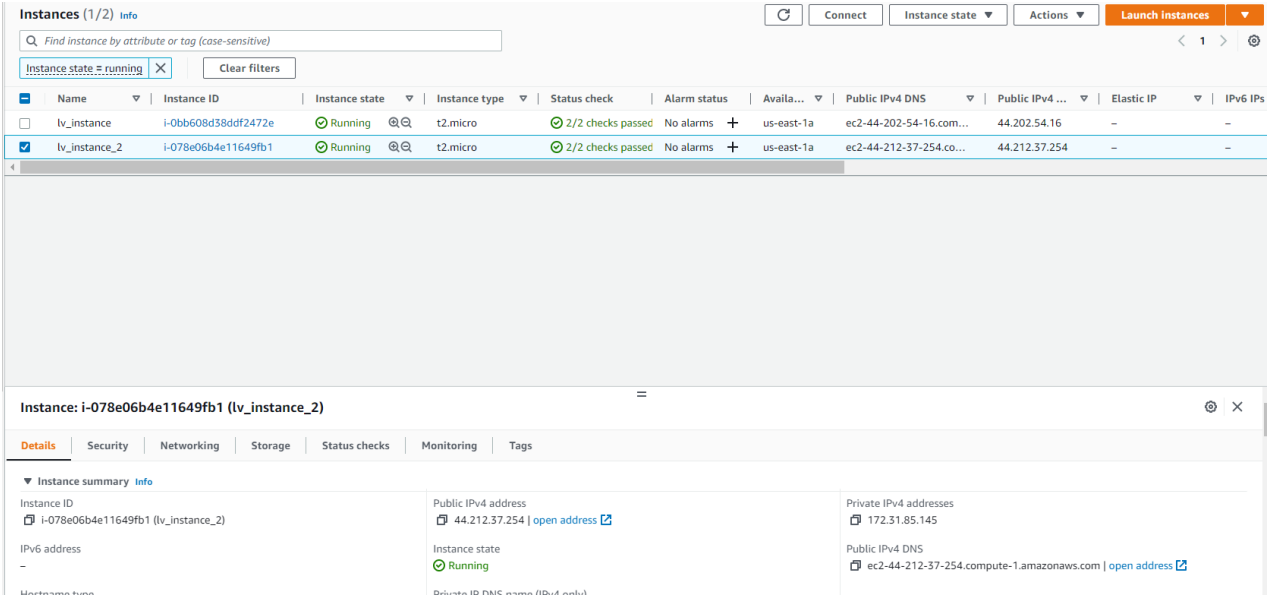
I create a volume from the snapshot:

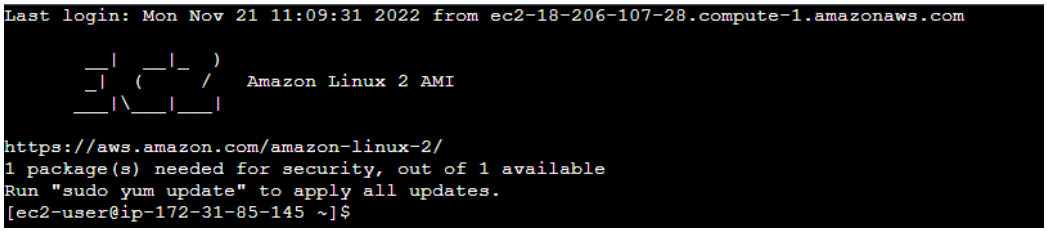


I create the instance as before, but I use the created AMI:



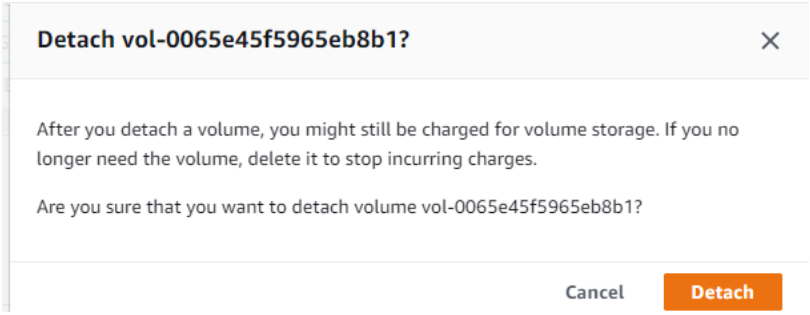
I check the instance via terminal:



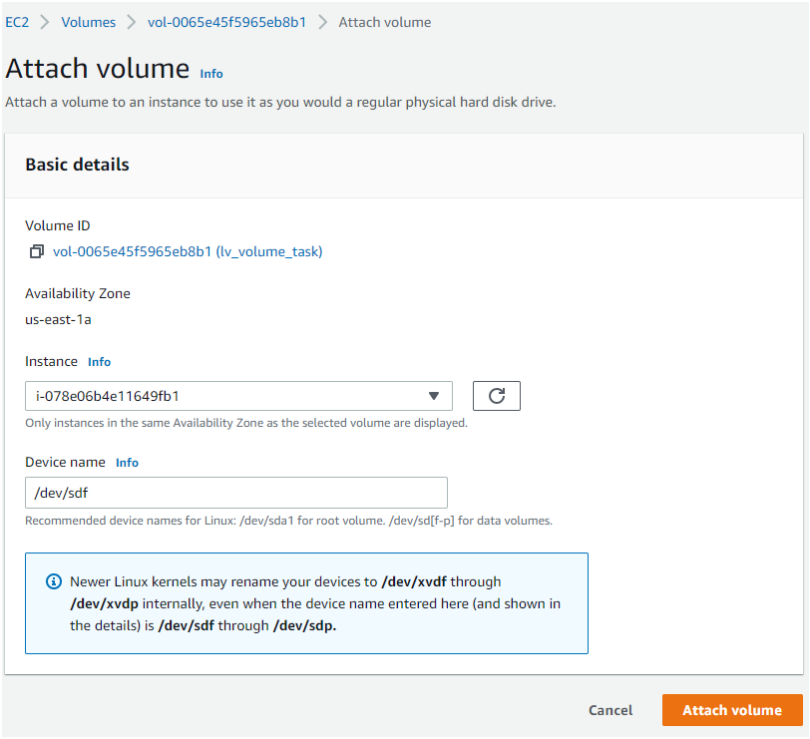


5) Detach volume from the 1st instance and attach volume to the new instance.

I detach the volume from the first instance:



I attach the volume to the second instance:

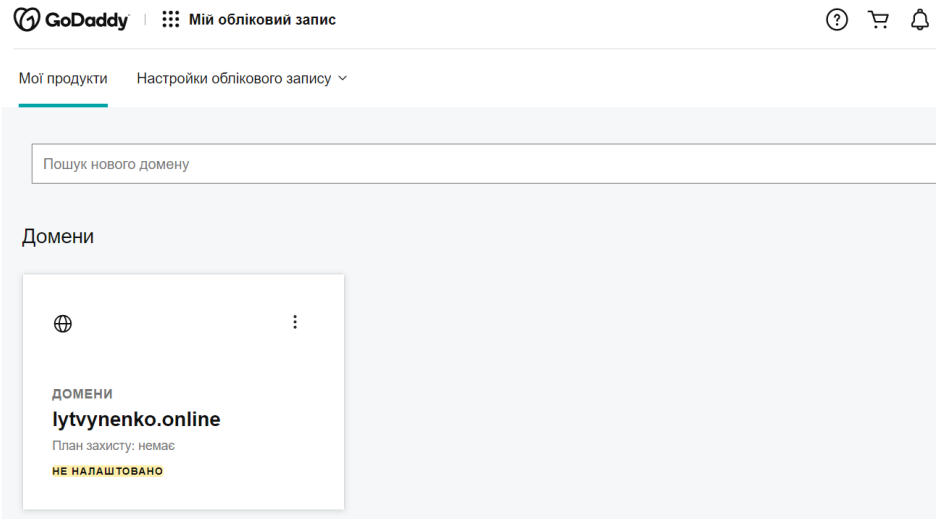


I create a directory for mounting and mount the drive there. I check the presence of created files in the previous instance.

```
[ec2-user@ip-172-31-85-145 ~]$ sudo su
[root@ip-172-31-85-145 ec2-user]# lsblk
NAME    MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0   8G  0 disk
└─xvda1 202:1    0   8G  0 part /
xvdf    202:80   0  10G  0 disk
[root@ip-172-31-85-145 ec2-user]# mkdir /mnt/new_mount_dir
[root@ip-172-31-85-145 ec2-user]# mount /dev/sdf /mnt/new_mount_dir/
[root@ip-172-31-85-145 ec2-user]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        474M     0  474M   0% /dev
tmpfs           483M     0  483M   0% /dev/shm
tmpfs           483M  464K  483M   1% /run
tmpfs           483M     0  483M   0% /sys/fs/cgroup
/dev/xvda1      8.0G  1.6G  6.5G  20% /
tmpfs            97M     0   97M   0% /run/user/0
tmpfs            97M     0   97M   0% /run/user/1000
/dev/xvdf       9.7G   24K  9.2G   1% /mnt/new_mount_dir
[root@ip-172-31-85-145 ec2-user]# ls /mnt/new_mount_dir/
file1  file10  file2  file3  file4  file5  file6  file7  file8  file9  lost+found
[root@ip-172-31-85-145 ec2-user]#
```

6) Review the 10-minute example. Explore the possibilities of creating your own domain and domain name for your site.

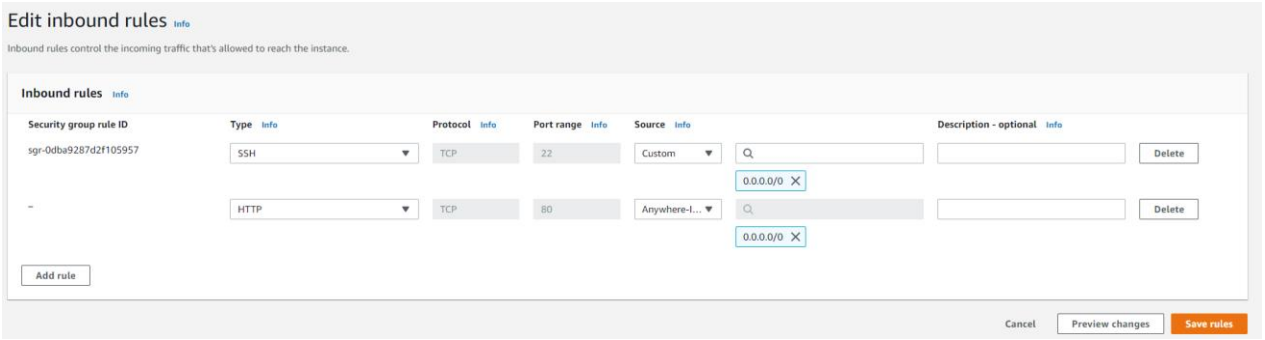
I'm going to use an already created domain that I usually use for training purposes:



I install a web server to check the correctness of the settings in the future (on the first machine created):

```
[ec2-user@ip-172-31-82-118 ~]$ sudo yum install -y httpd
```

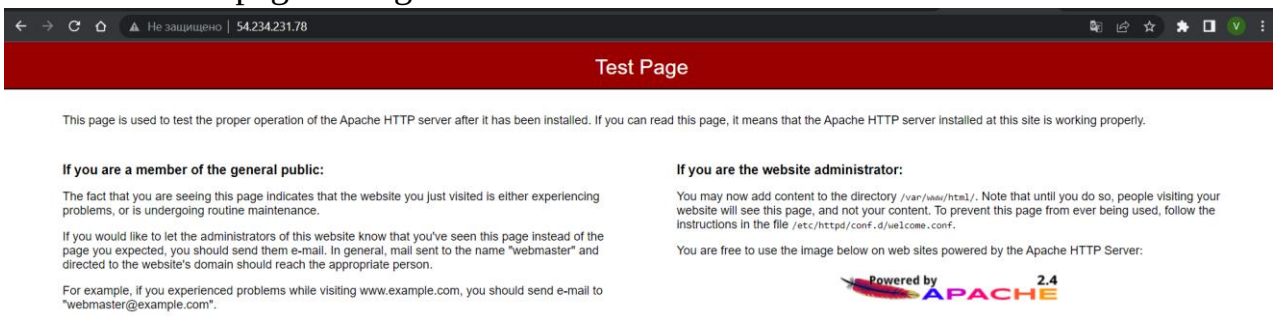
I configure the security group to allow traffic on port 80:



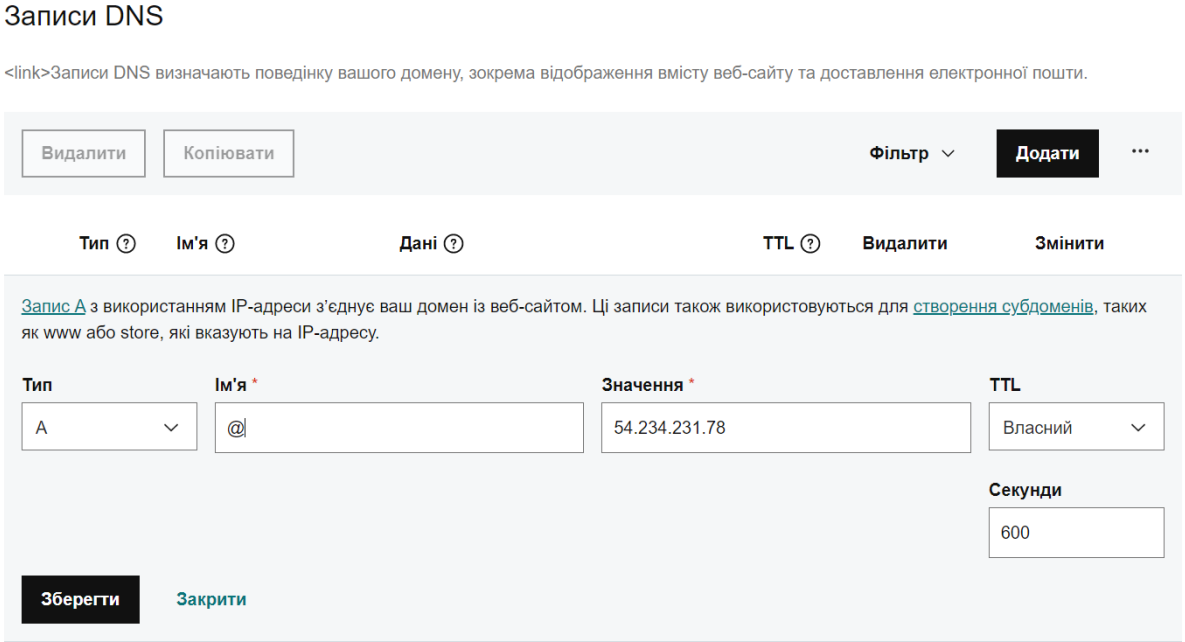
I run a web server:

```
[ec2-user@ip-172-31-82-118 ~]$ sudo systemctl start httpd
```

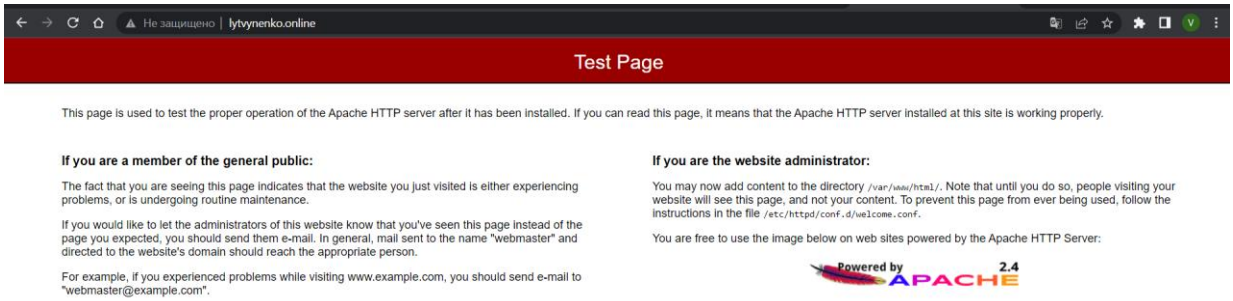
Check the web page through the IP address:



On GoDaddy, I configure the domain records to point to the EC2 instance:

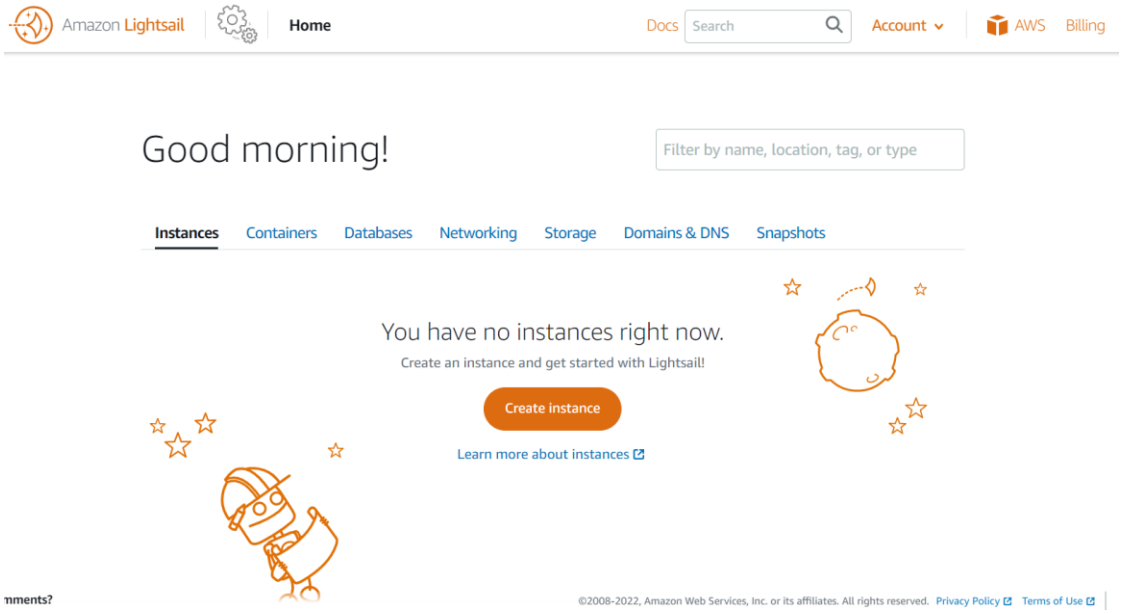


Everything works fine. The website can be found by its domain name

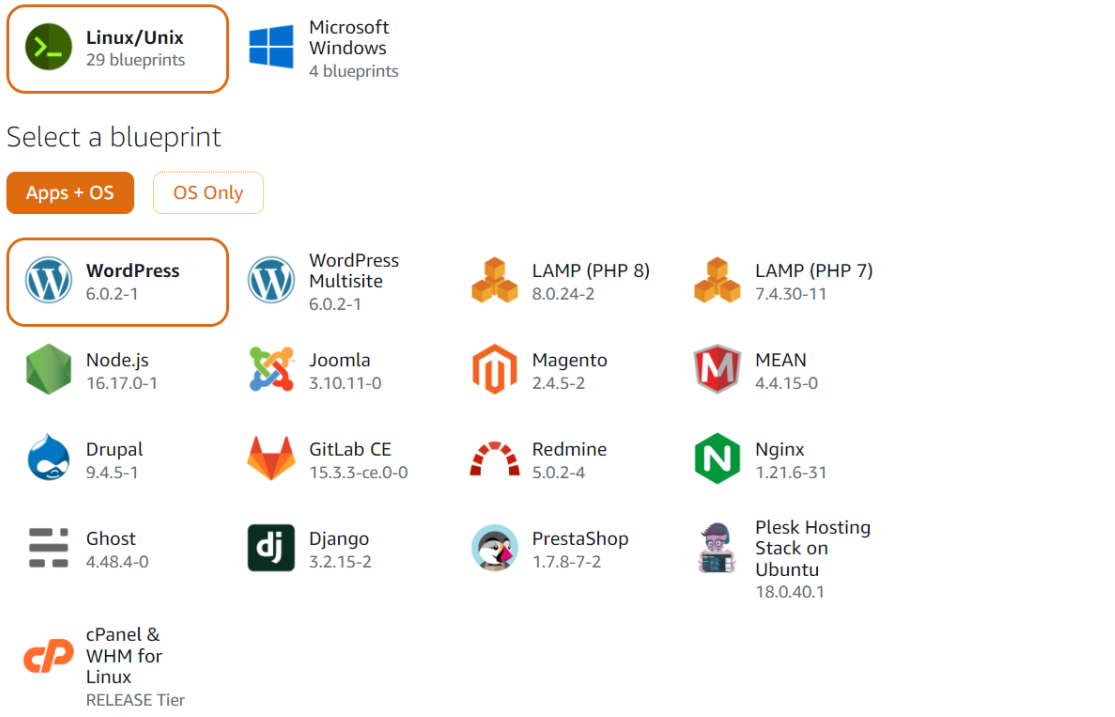


7) Launch and configure a WordPress instancewith Amazon Lightsail

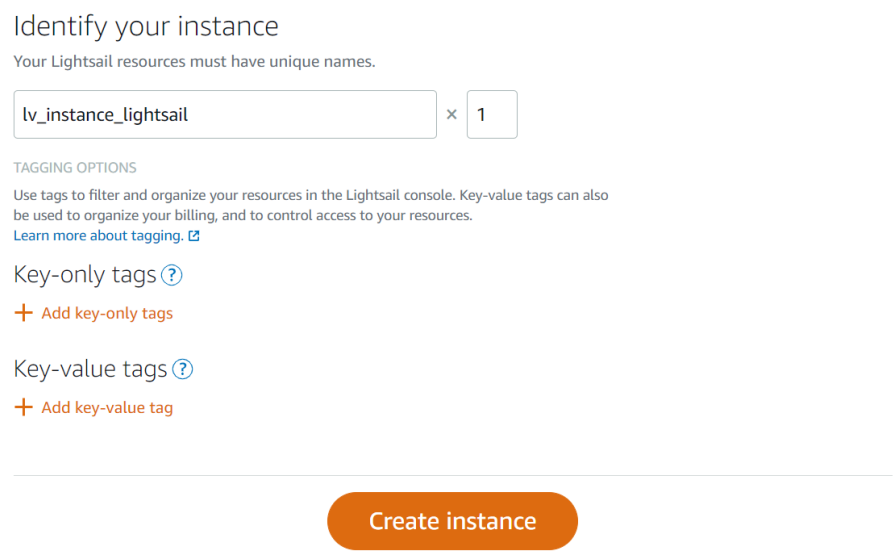
I create an instance with Lightsail:



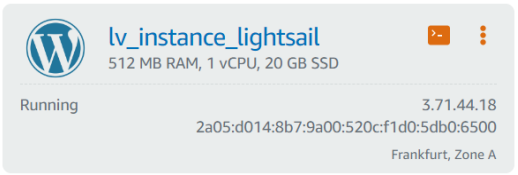
I choose Linux/Unix as the platform and WordPress as the blueprint:



I enter a name for my instance and choose «Create instance»:



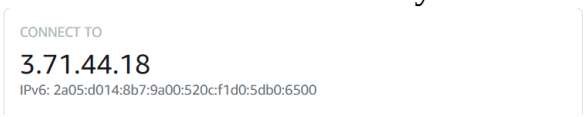
On the Instances tab of the Lightsail home page, I choose the SSH quick-connect icon:



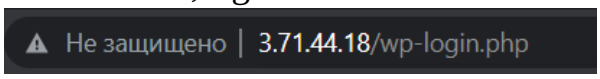
I enter the following command to retrieve the default application password:

```
bitnami@ip-172-26-12-224:~$ cat $HOME/bitnami_application_password
00C7nlq3tFmM
```

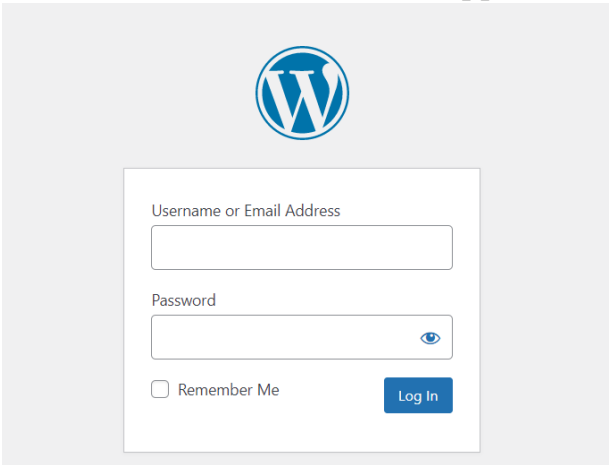
I define the IP address of my instance:



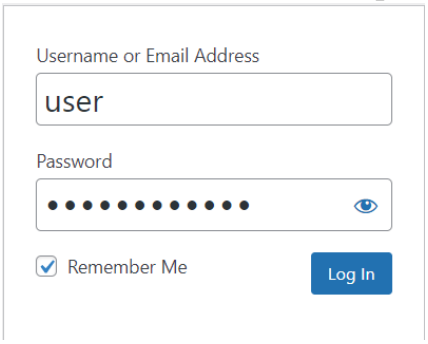
In a browser, I go to:



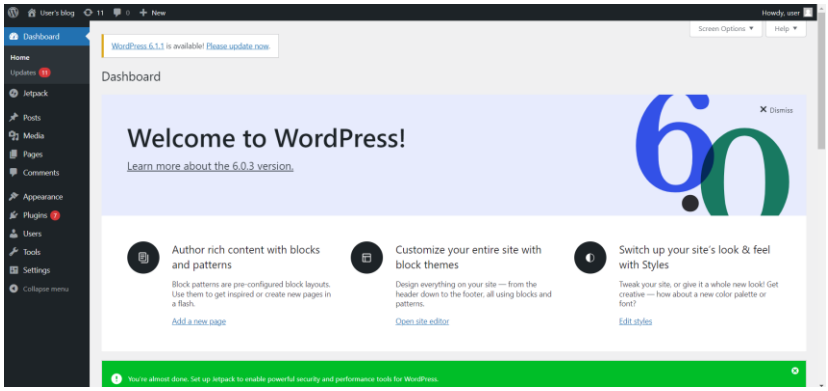
An administration dashboard appears:



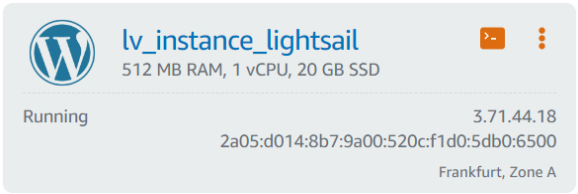
I enter a username and a password:



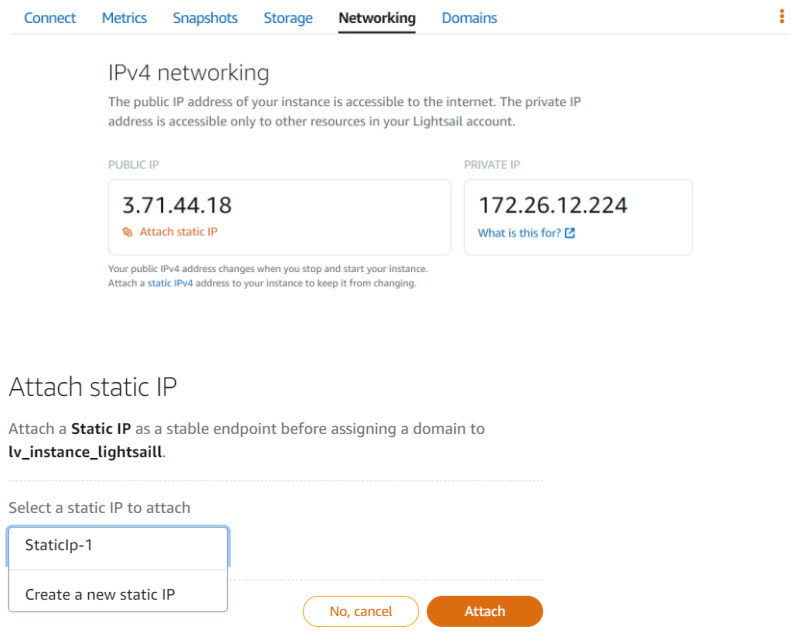
The result of the above actions:



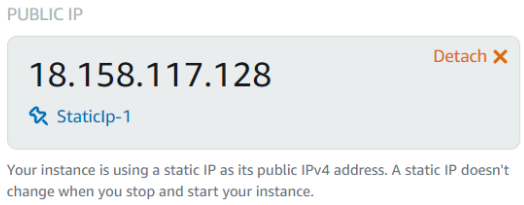
On the Instances tab of the Lightsail home page, I choose my running WordPress instance:



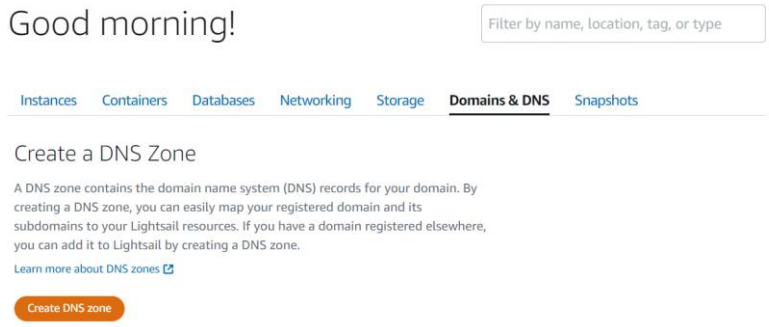
I open the «Networking» tab, then choose «Attach static IP»:



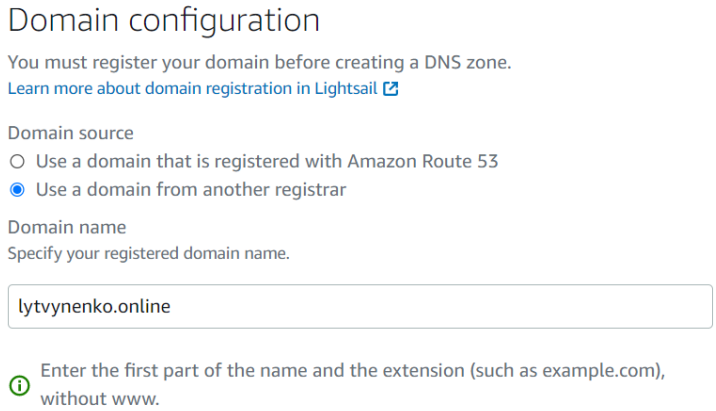
A static IP is attached:



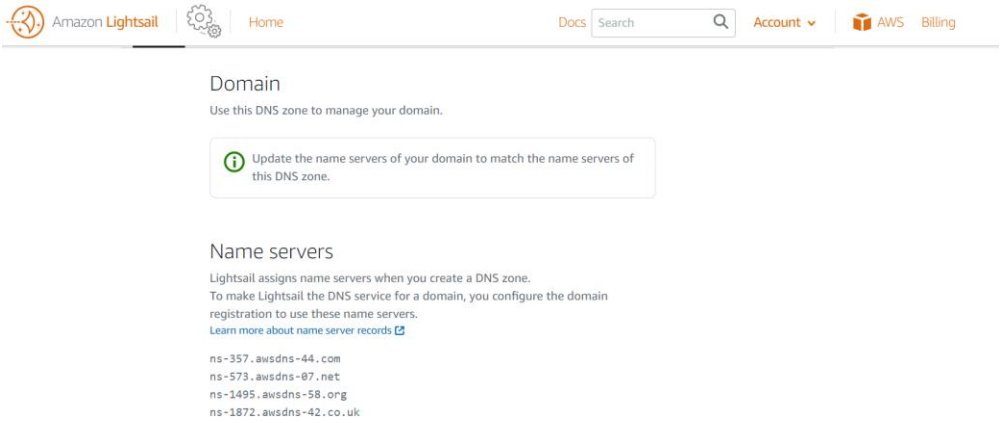
On the «Domains & DNS» tab of the home page, I choose «Create DNS zone»:

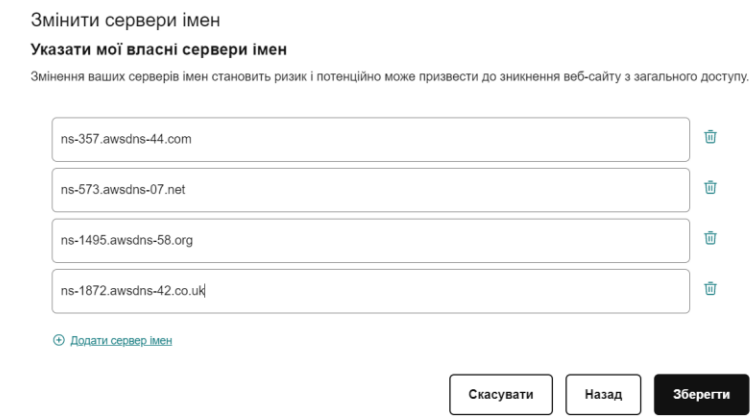


I enter domain configuration and create a domain zone:

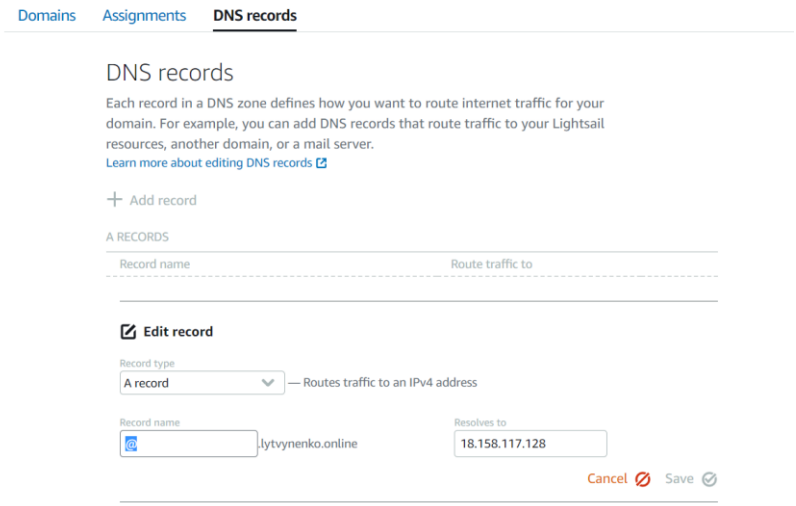


I add the received name server addresses to my domain registrar:

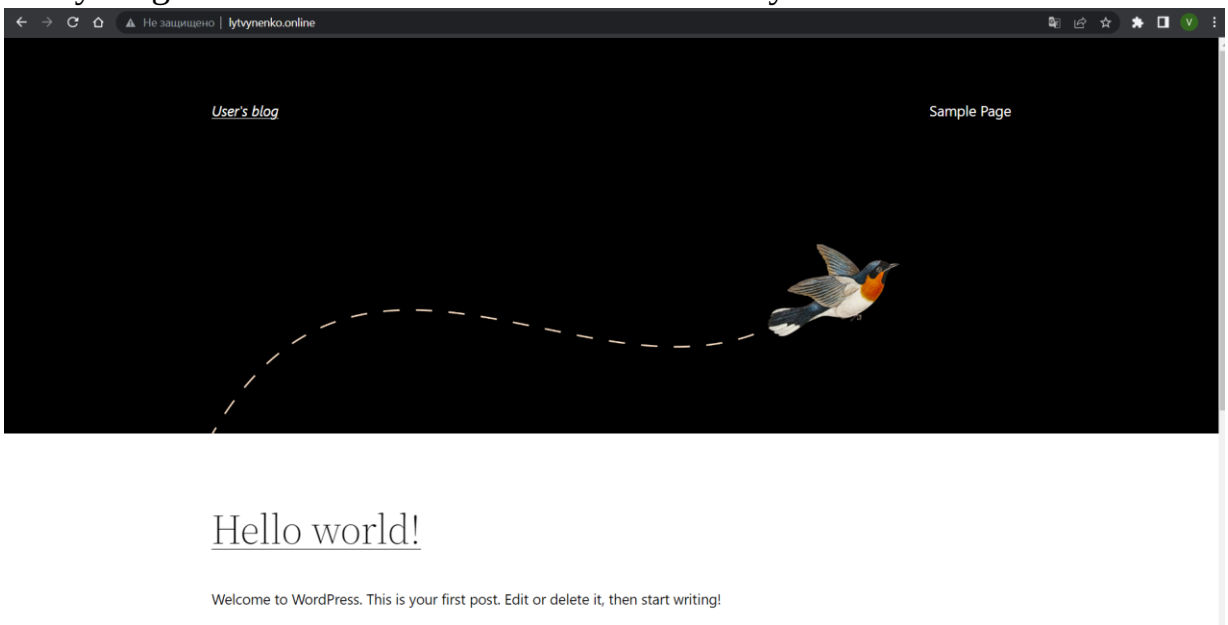




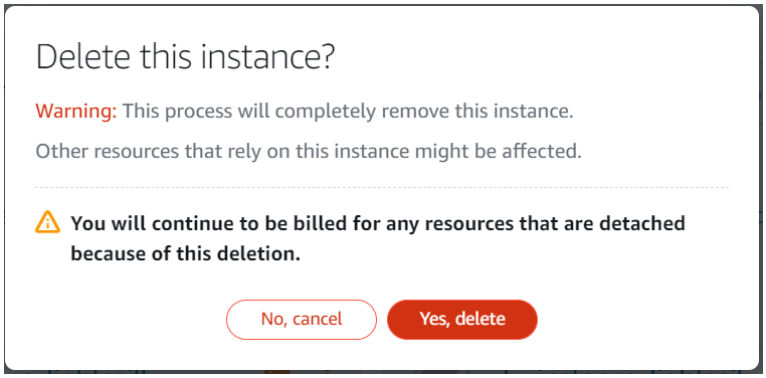
I add new DNS record to Lightsail:



Everything works fine. The website can be found by its domain name:

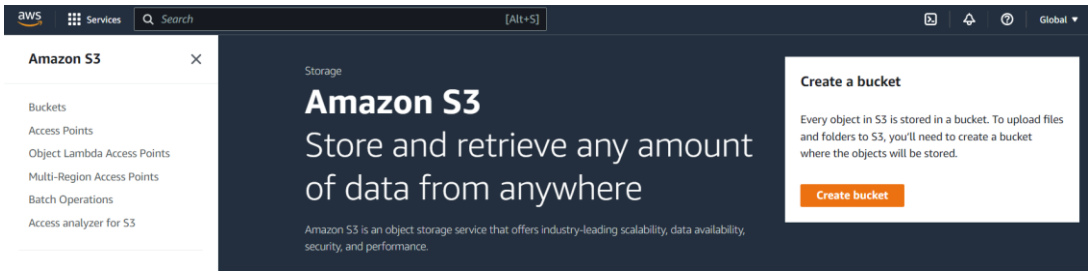


I delete the instance:

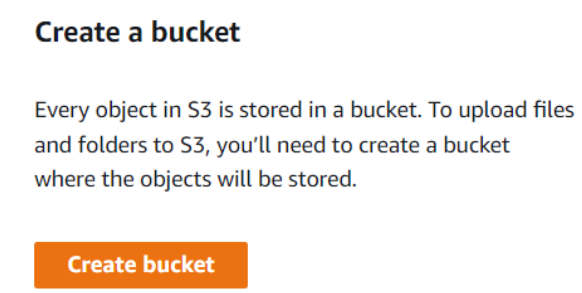


8) Review the 10-minute Store and Retrieve a File. Repeat, creating your own repository.

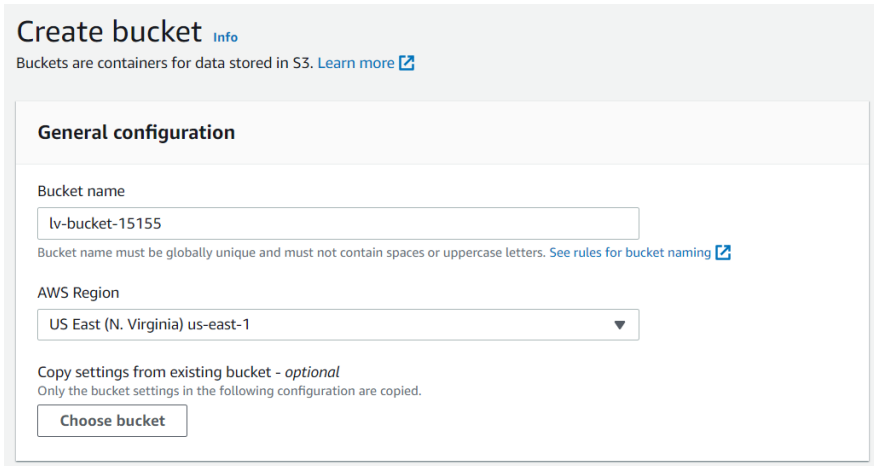
I enter the Amazon S3 console:



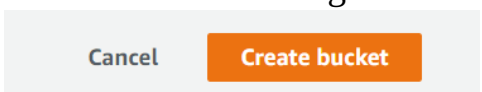
I create a new bucket:



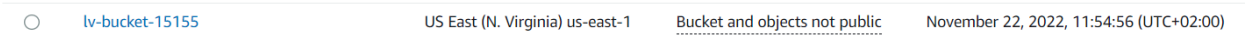
I enter a bucket name:



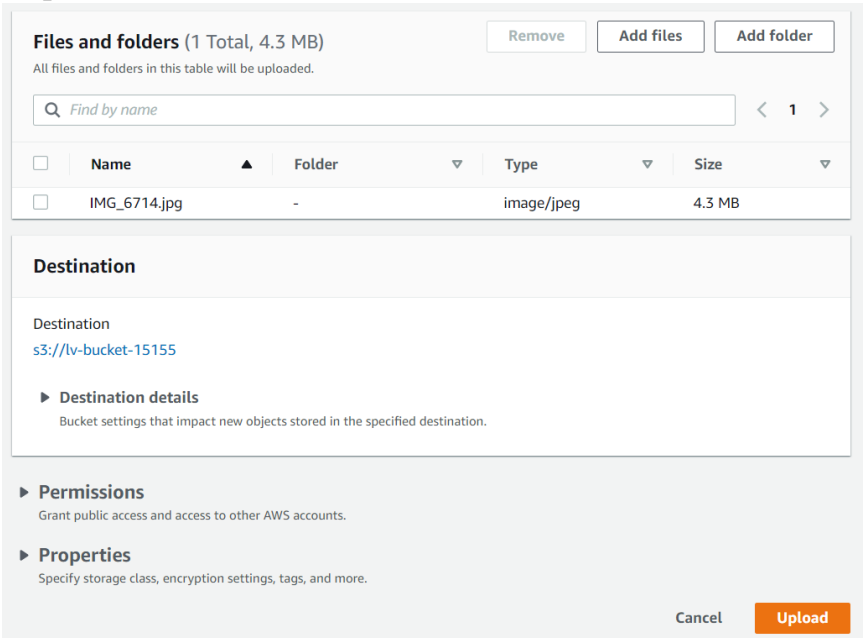
I leave all other settings as default and create a bucket:



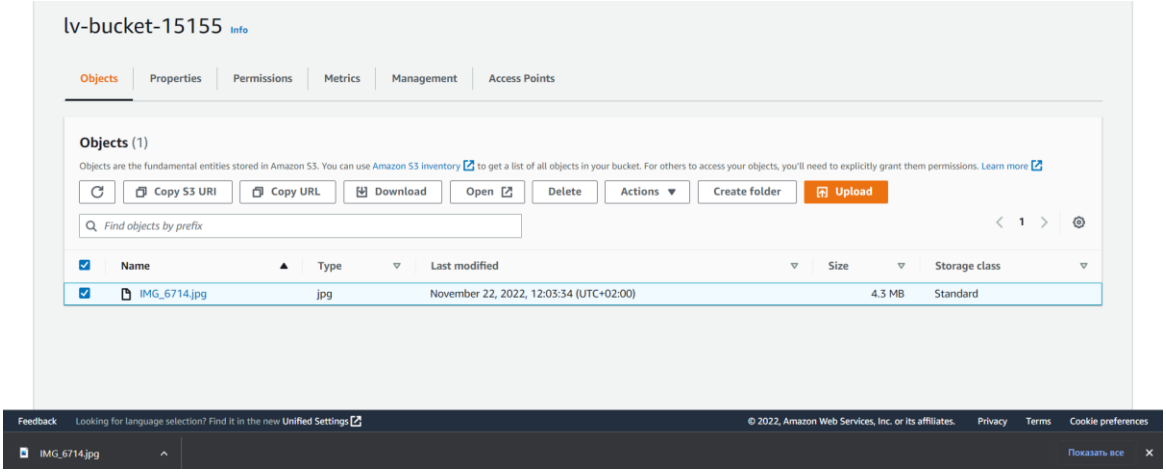
I navigate to the created bucket:



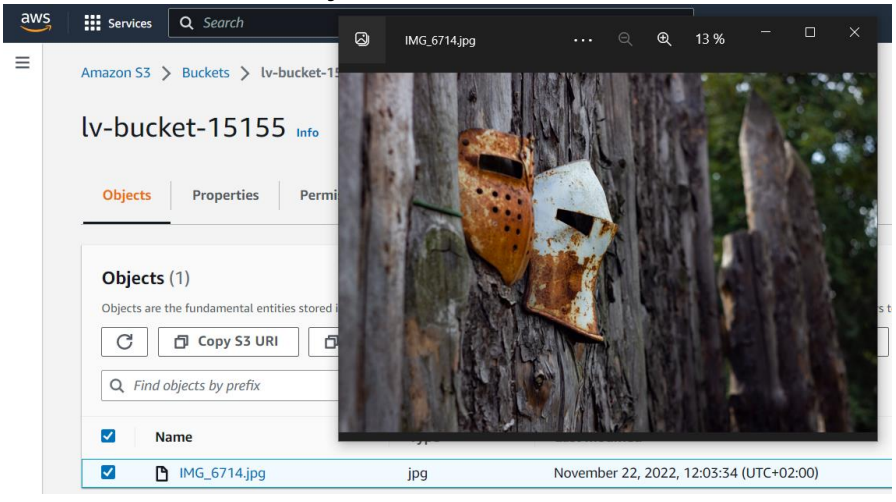
I upload a file to the bucket:



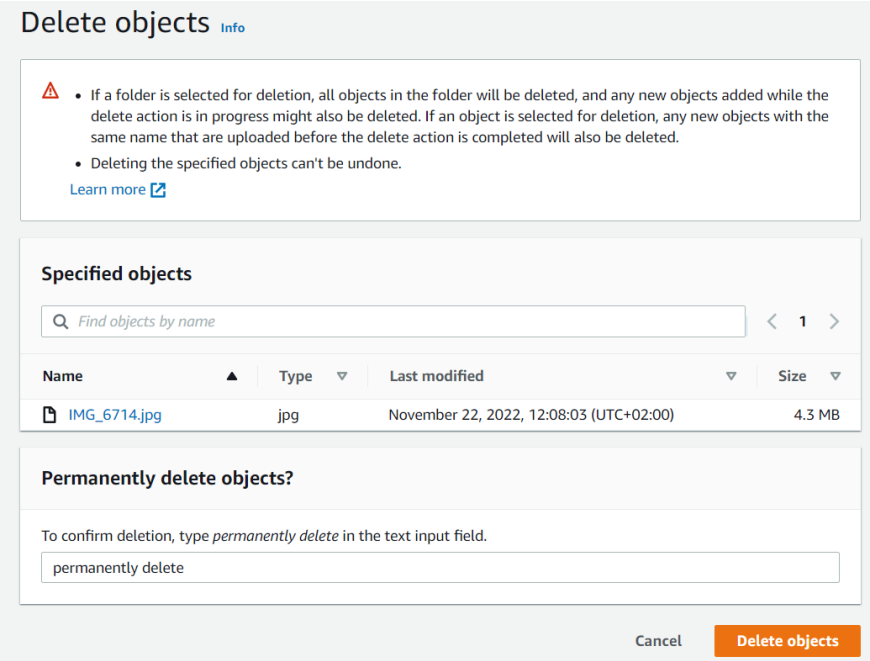
In the bucket, I select a file that I want to download, click «Download» button:



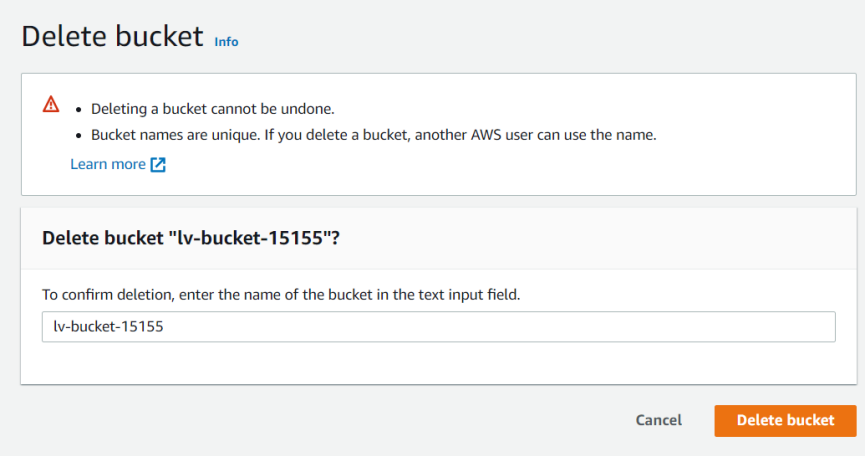
The file is successfully downloaded:



I delete the file from the bucket:

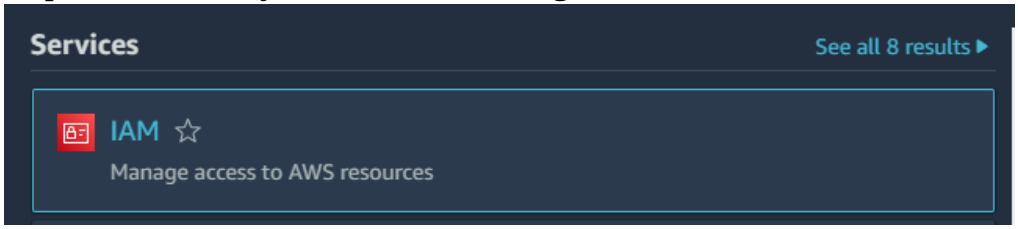


I delete the bucket:

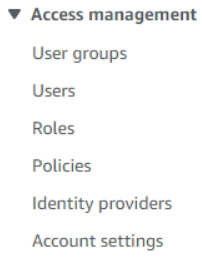


9) Review the 10-minute example Batch upload files to the cloud to Amazon S3 using the AWS CLI. Create a user AWS IAM, configure CLI AWS and upload any files to S3.

I open the Identity and Access Management dashboard:

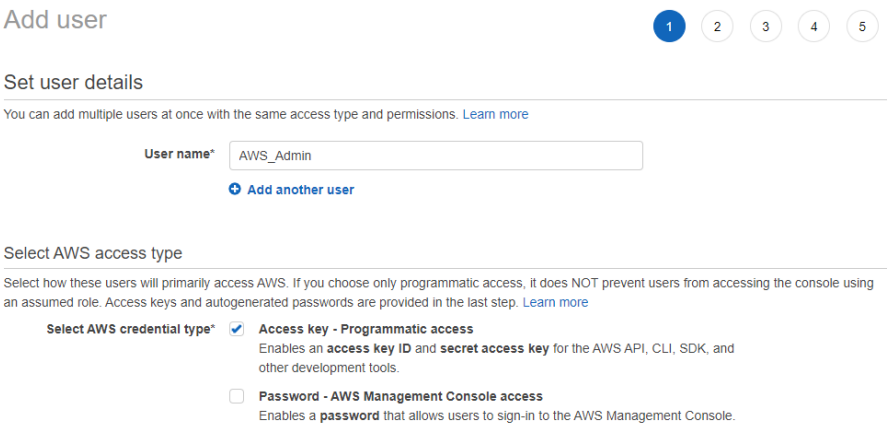


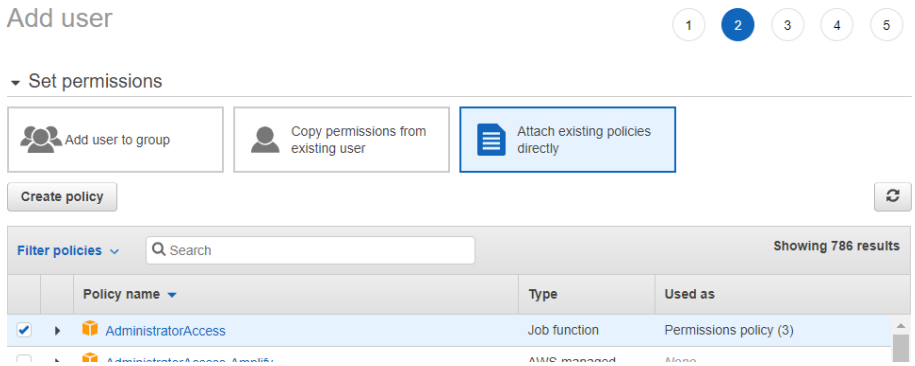
I select «Users» tab:



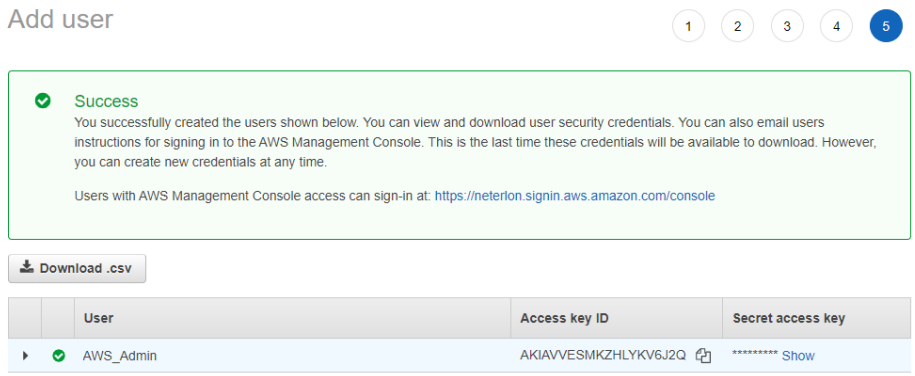
I add a new user:

I configure a user:





I leave the other settings as default and download user security credentials:



I type «aws configure» in console and press enter:

```
C:\Users\vlady>aws configure
AWS Access Key ID [****************LEKP]:
```

I enter user credentials:

```
C:\Users\vlady>aws configure
AWS Access Key ID [****************LEKP]: AKIAVVESMKZHLYKV6J2Q
AWS Secret Access Key [****************LIXa]: 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
Default region name [us-east-1]: us-east-1
Default output format [json]: json
```

I create a new bucket through a console**:**

```
C:\Users\vlady>aws s3 mb s3://my-backup-bucket-454545
make_bucket: my-backup-bucket-454545
```

I upload a file located in my local directory to the S3 bucket:

```
C:\Users\vlady>aws s3 cp "C:\Users\vlady\Desktop\IMG_6650.jpg" s3://my-backup-bucket-454545
upload: Desktop\IMG_6650.jpg to s3://my-backup-bucket-454545/IMG_6650.jpg
```

I do a reverse command and download the file:

```
C:\Users\vlady>aws s3 cp s3://my-backup-bucket-454545/IMG_6650.jpg C:\Users\vlady\Desktop\
download: s3://my-backup-bucket-454545/IMG_6650.jpg to Desktop\IMG_6650.jpg
```

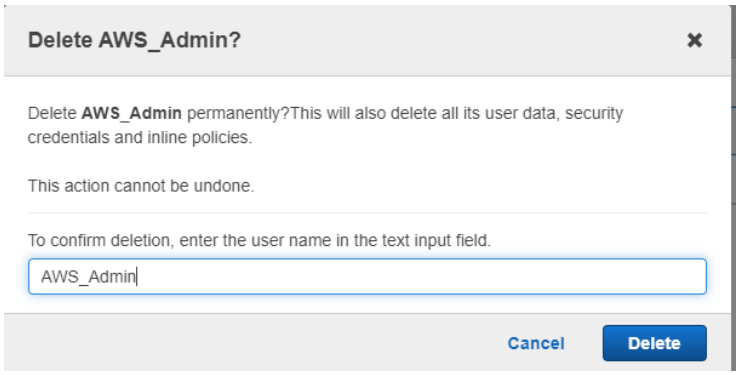
I delete the file from the bucket:

```
C:\Users\vlady>aws s3 rm s3://my-backup-bucket-454545/IMG_6650.jpg
delete: s3://my-backup-bucket-454545/IMG_6650.jpg
```

I delete the bucket:

```
C:\Users\vlady>aws s3api delete-bucket --bucket my-backup-bucket-454545
```

I delete the user:

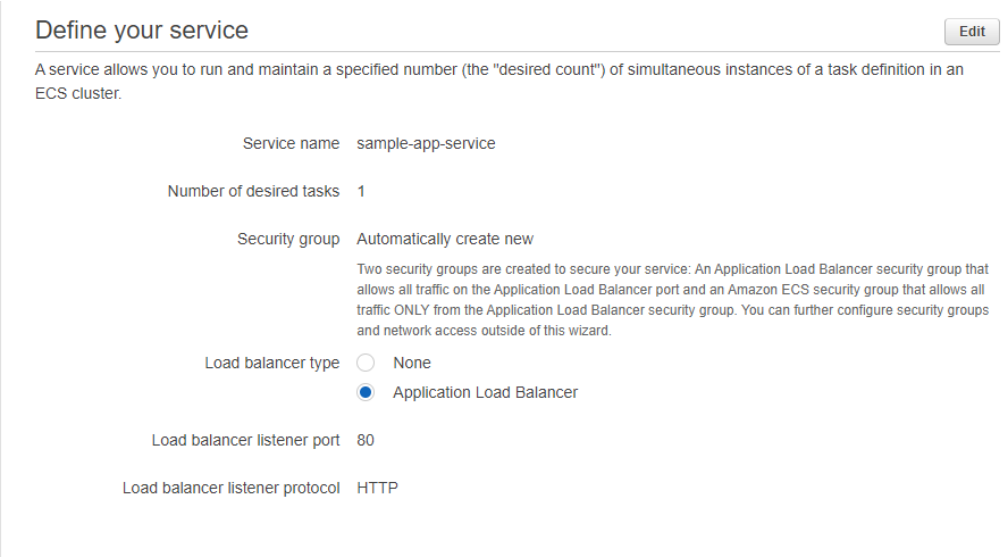


10) Review the «10-minute example Deploy Docker Containers on Amazon Elastic Container Service (Amazon ECS)».

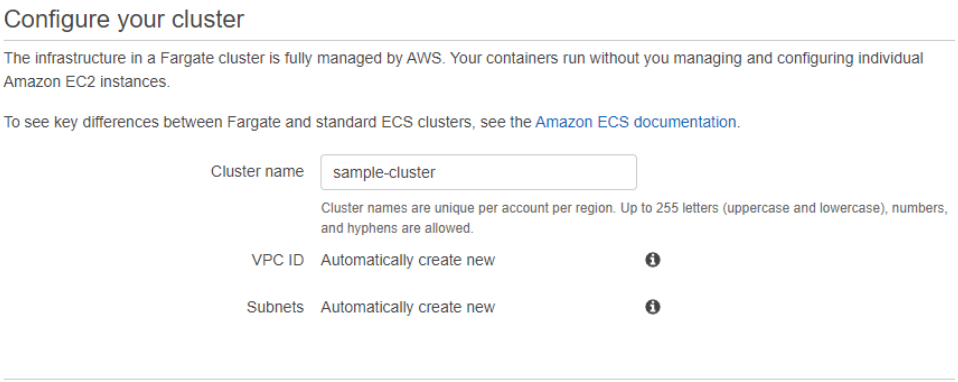
To launch the Amazon ECS first-run wizard, I choose the «Get started» button.

In the Container definition field, I select «sample-app»:

I enable Application Load Balancer:

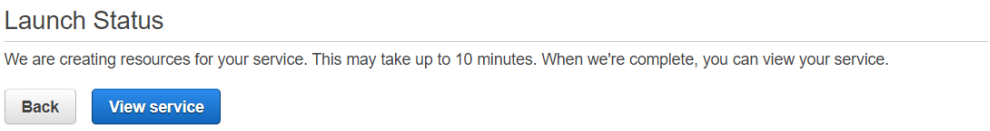


I enter a name for a cluster:

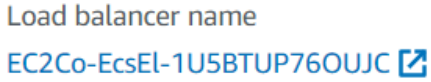


I launch my resources:

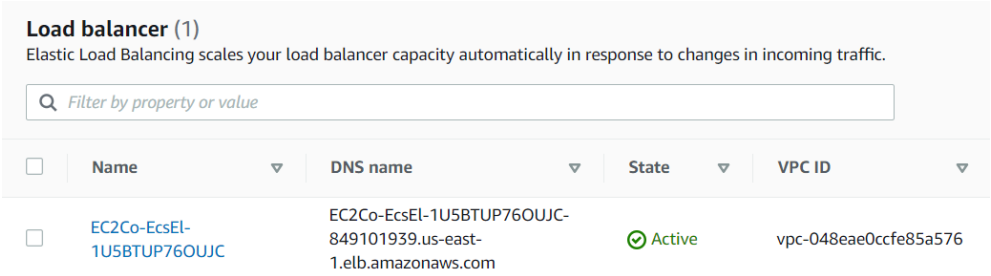
After the launch is complete, I choose «View service»:



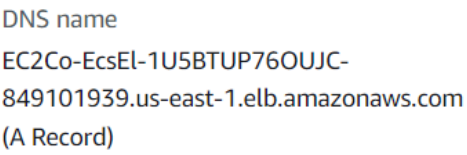
I select the Load balancer name:



I select the target group name:

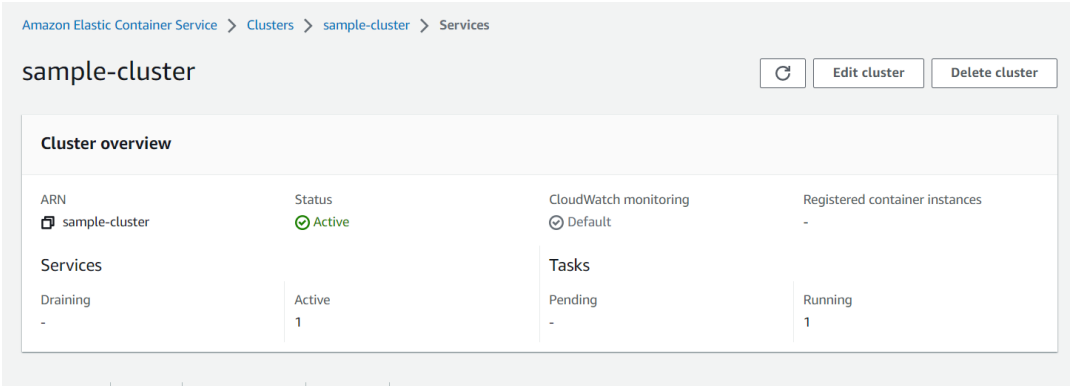


I copy the DNS name and paste it into a new browser window:

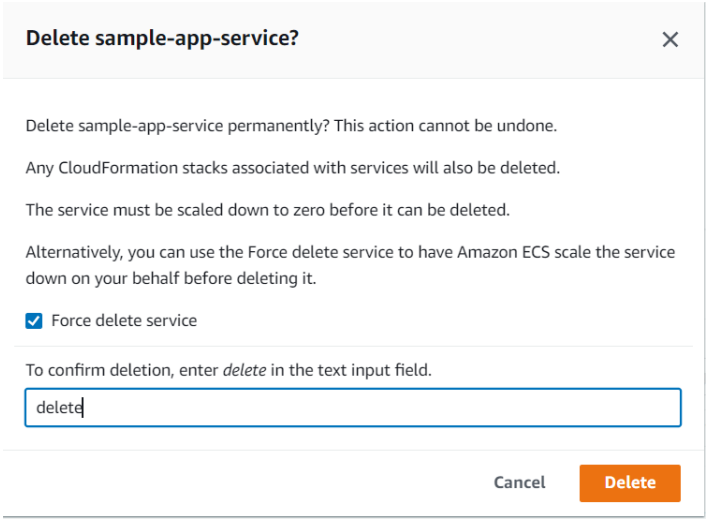


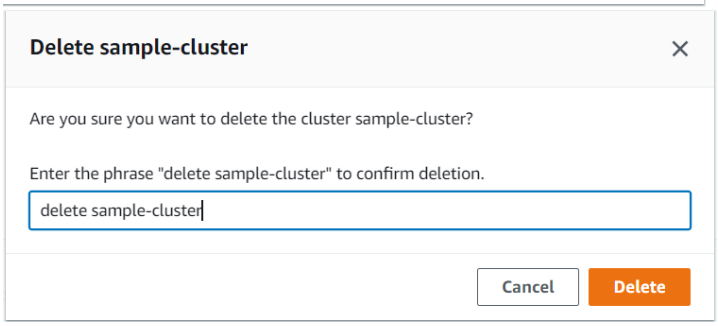
Everything works fine:

I navigate back to the Amazon ECS console page and select the cluster name:



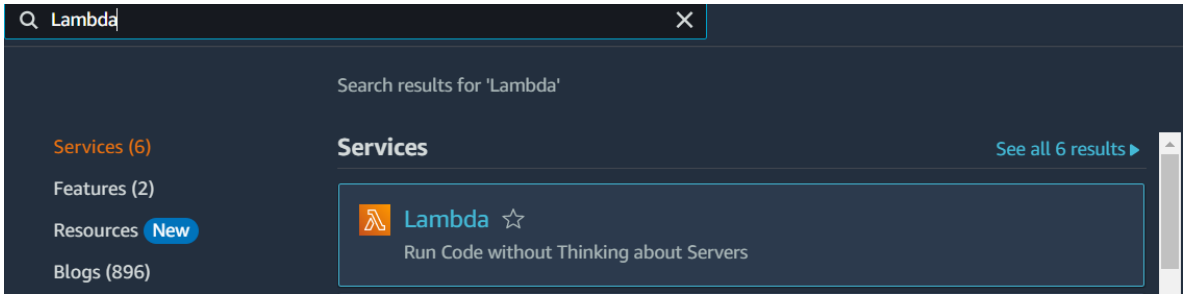
I delete the service, stop the task and then delete the cluster:





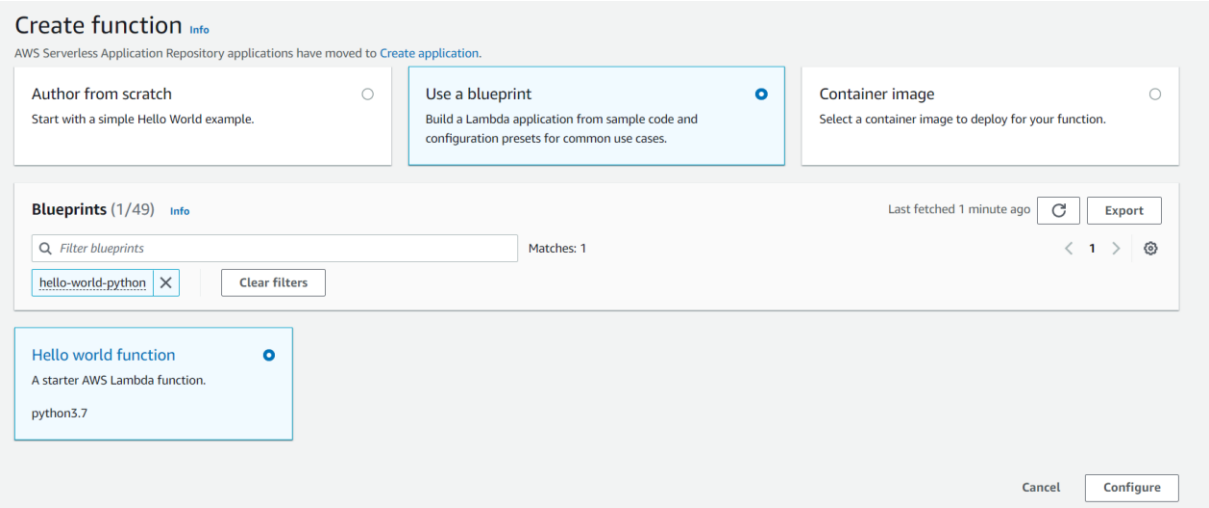
11) Run a Serverless "Hello, World!" with AWS Lambda.

I open the AWS Lambda Console:

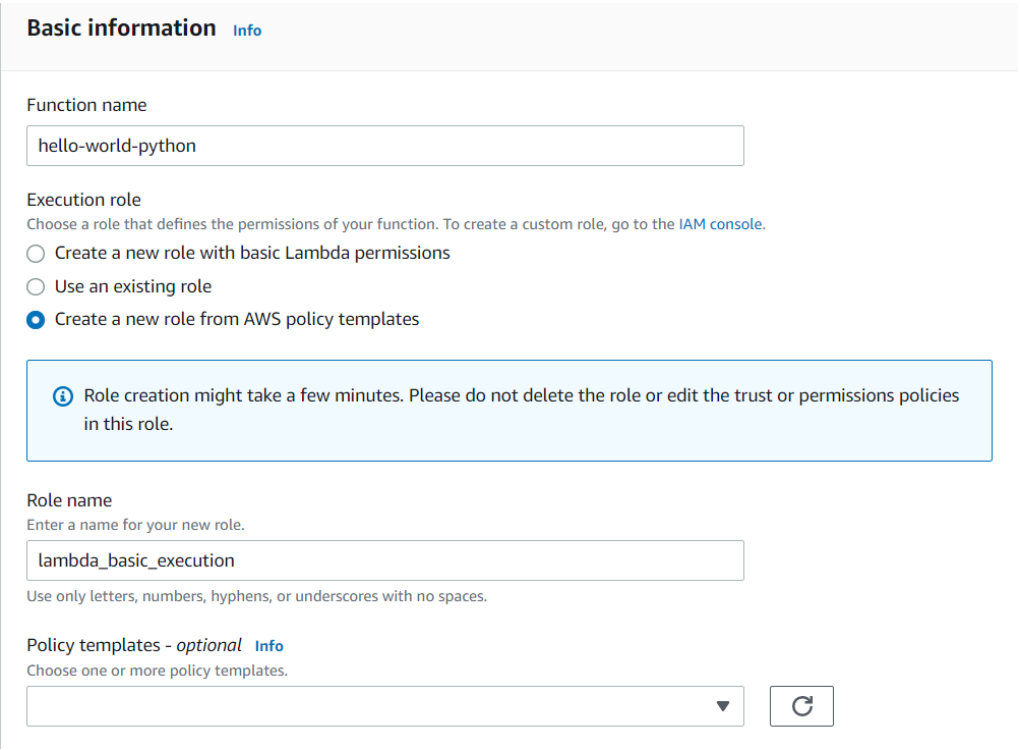


In the AWS Lambda console, I choose «Create function»:

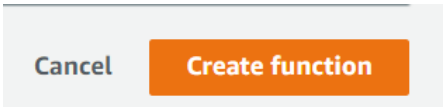
I select «use a blueprint». In the filter box, I enter «hello-world-python» and select the blueprint, then I choose «Configure»:



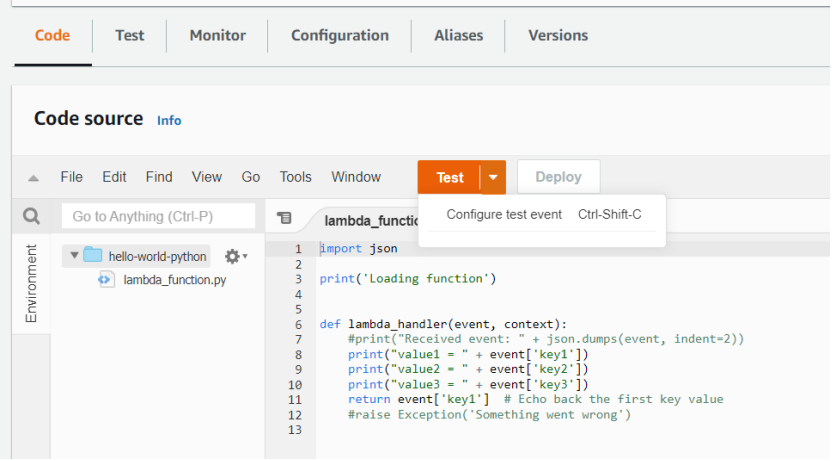
I enter some basic information about the Lambda function:



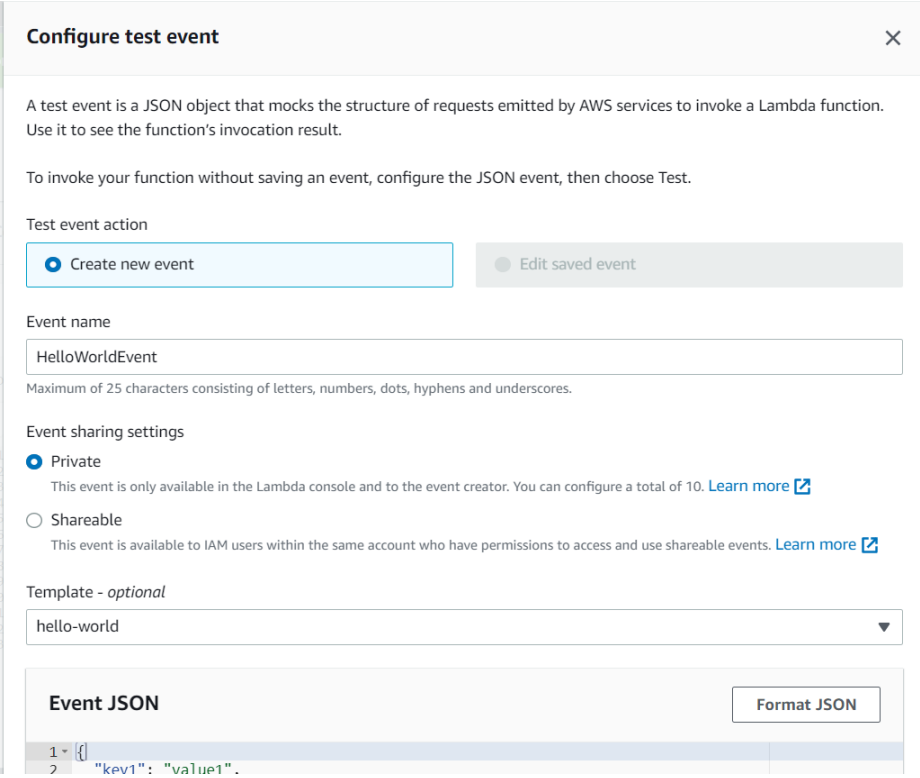
I select «Create function»:



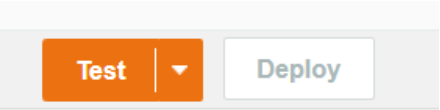
I select «Configure Test Event» from the drop-down menu called «Test» to test the function:



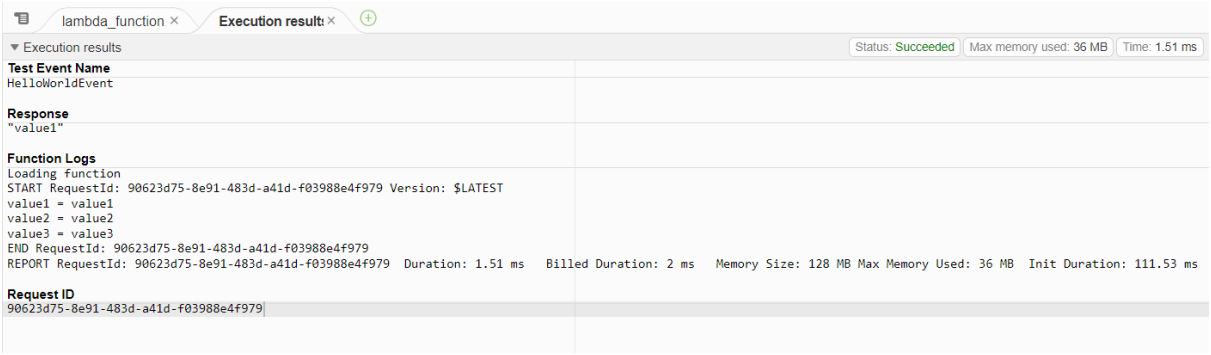
I create a new event:



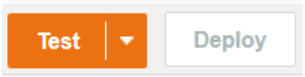
I choose «Test»:



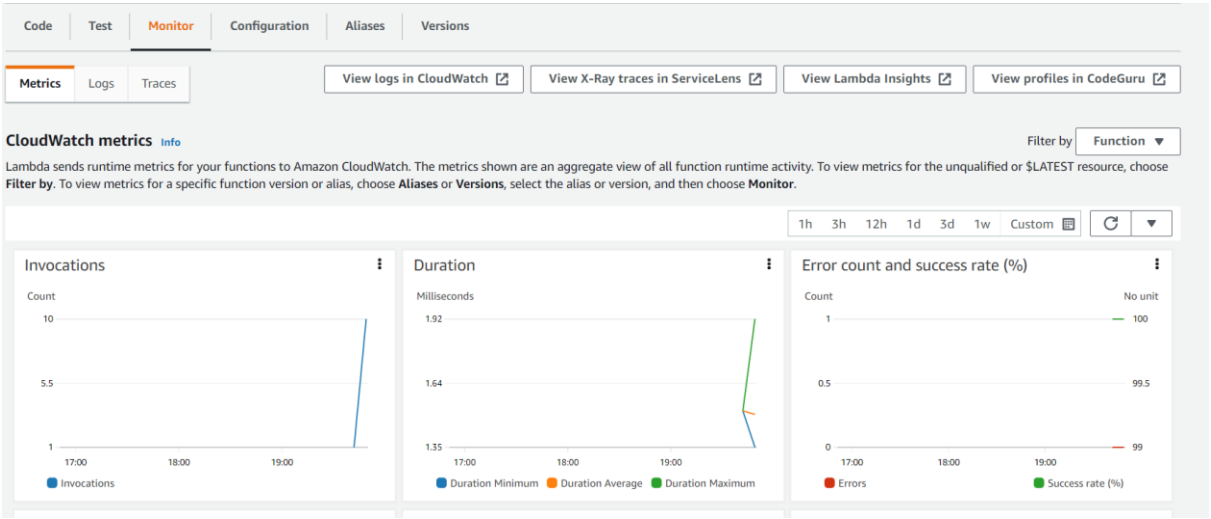
Upon successful execution, I view the results in the console:



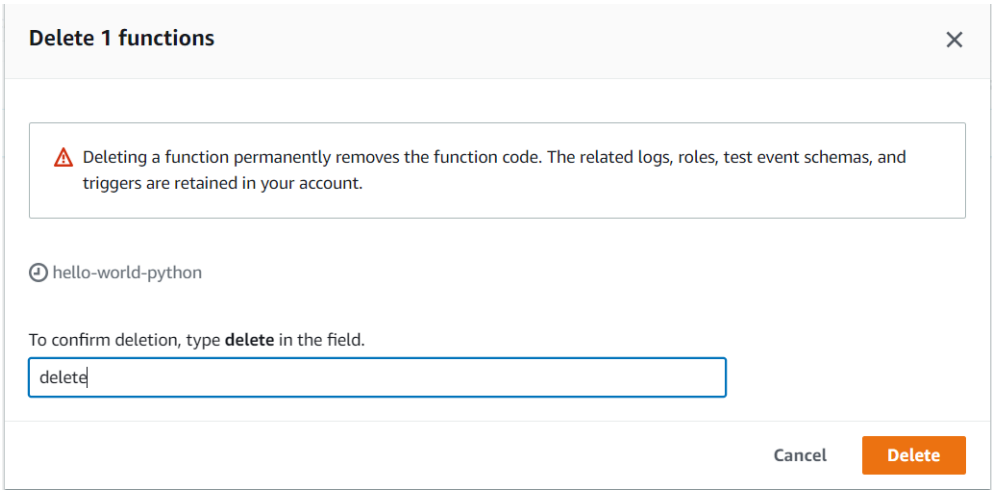
I invoke the Lambda function a few more times by repeatedly choosing the «Test» button to generate metrics:



I select the «Monitor» tab to view the results:

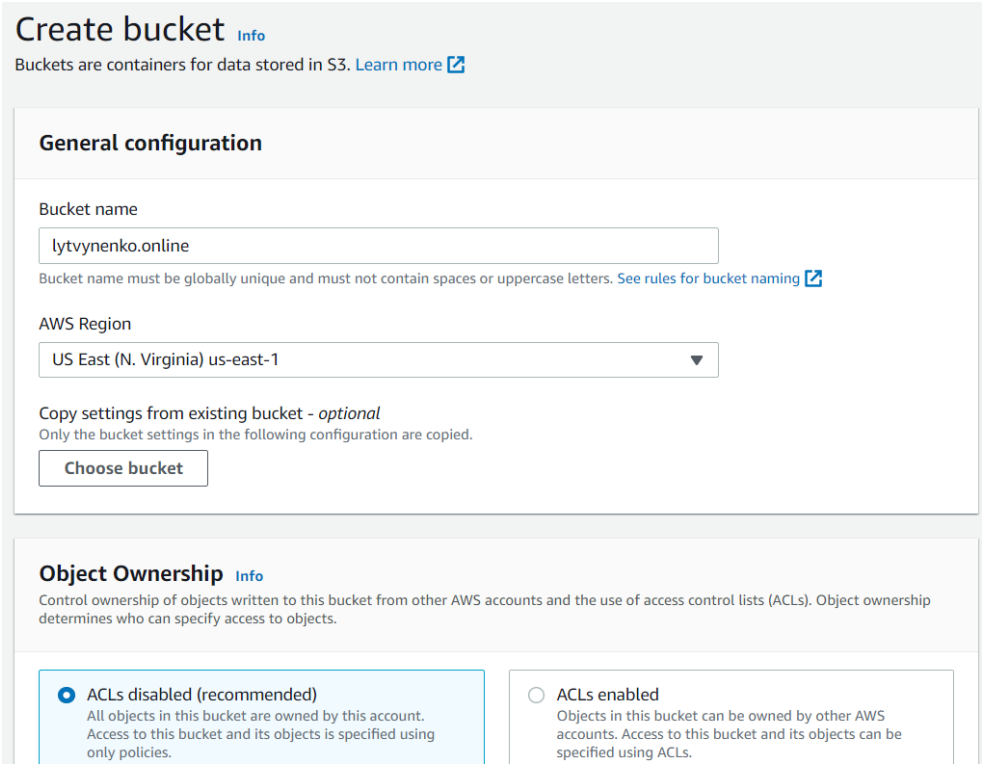


I delete the Lambda function:

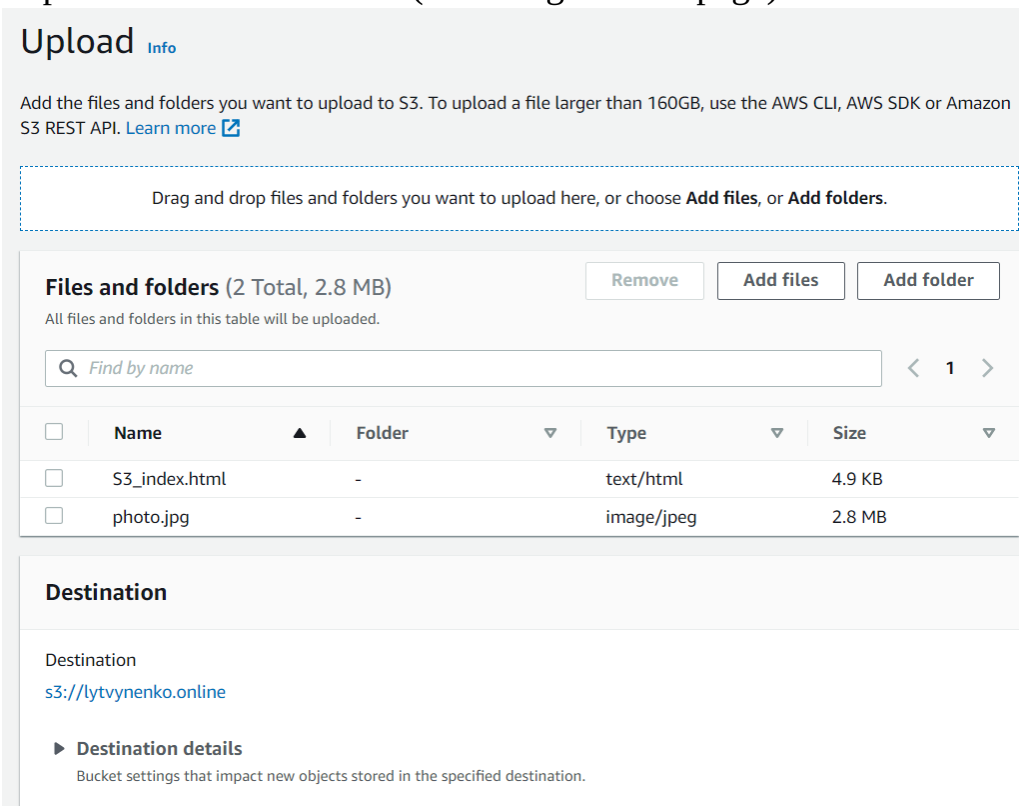


12) Create a static website on Amazon S3, publicly available (using a custom domain). Post on the page your own photo, the name of the educational program, the list of AWS services with which the student worked within the educational program or earlier and the full list with links of completed labs.

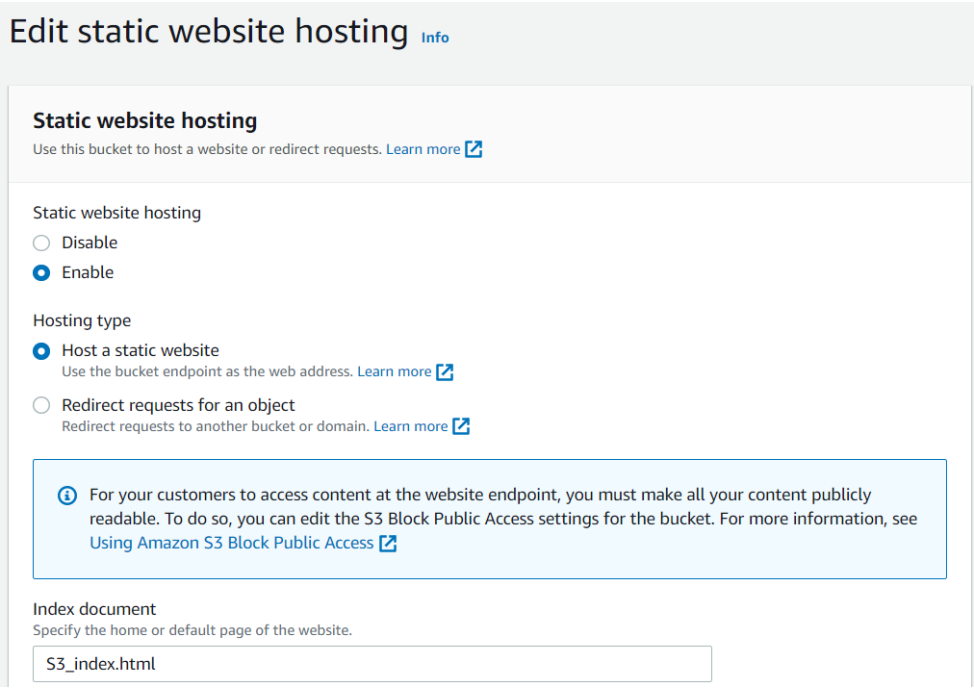
I create a bucket:



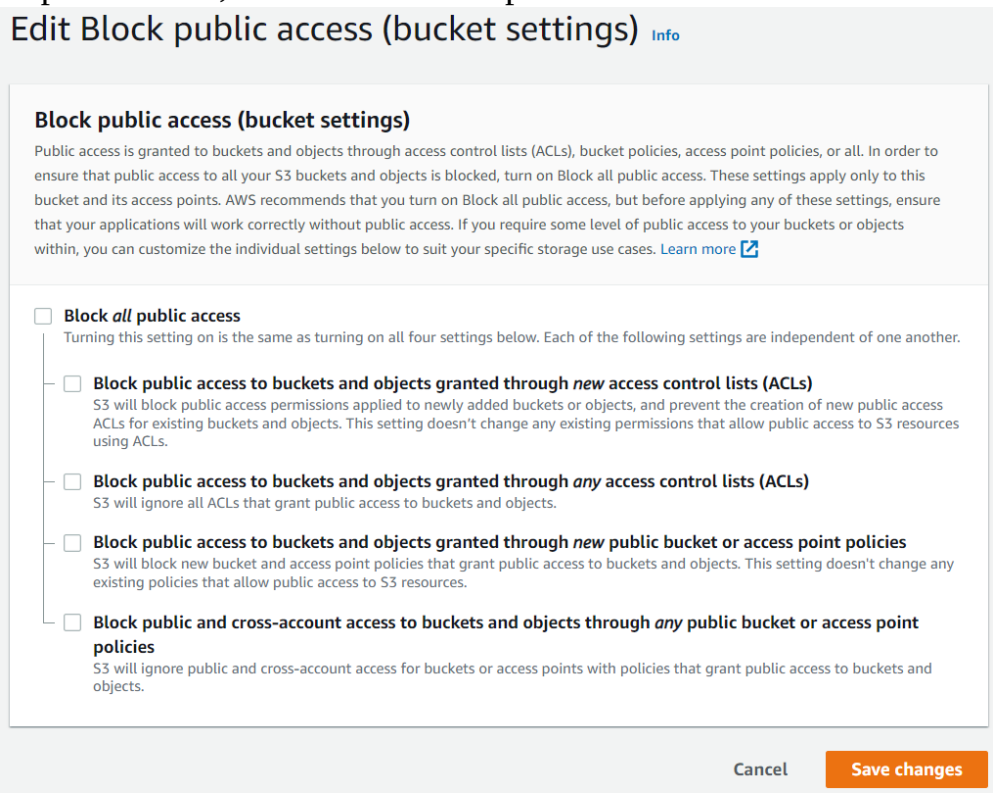
I upload files to the bucket (including an html page):



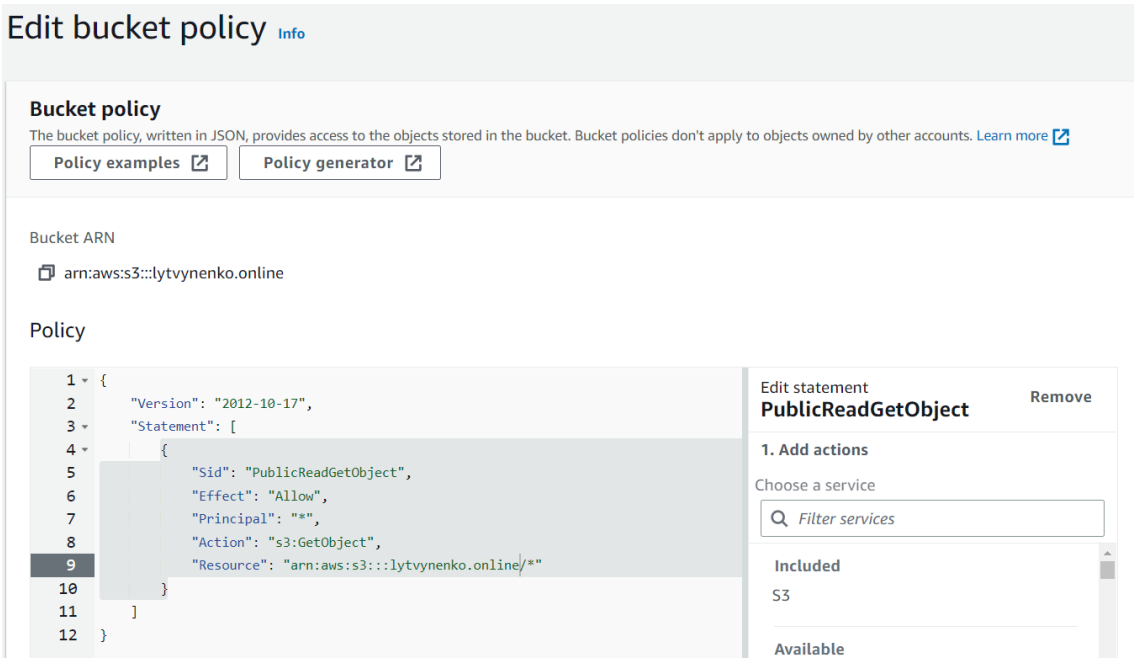
In properties, I enable static website hosting:



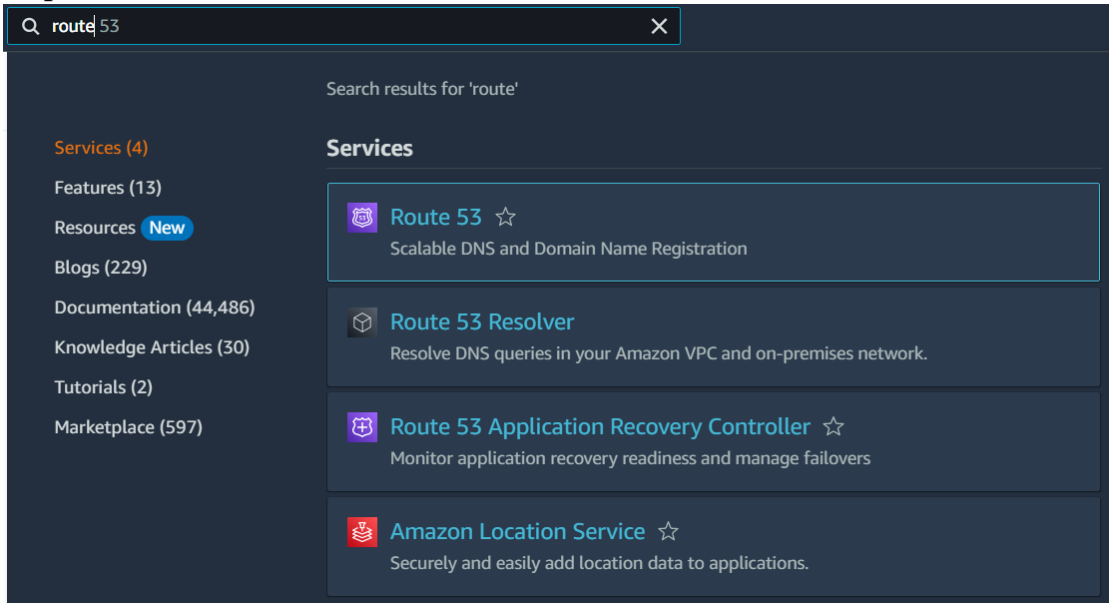
In permissions, I clear «Block all public access»:



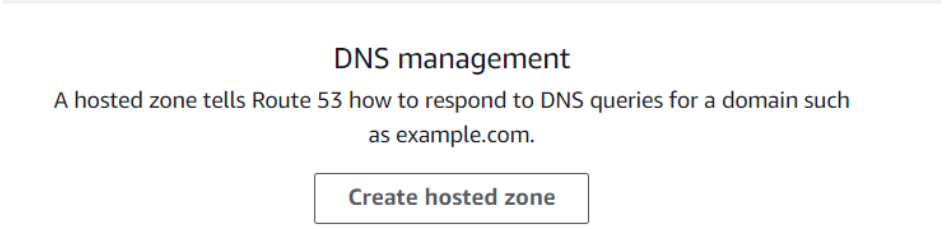
I grant public read access for the website (Bucket policy editor):



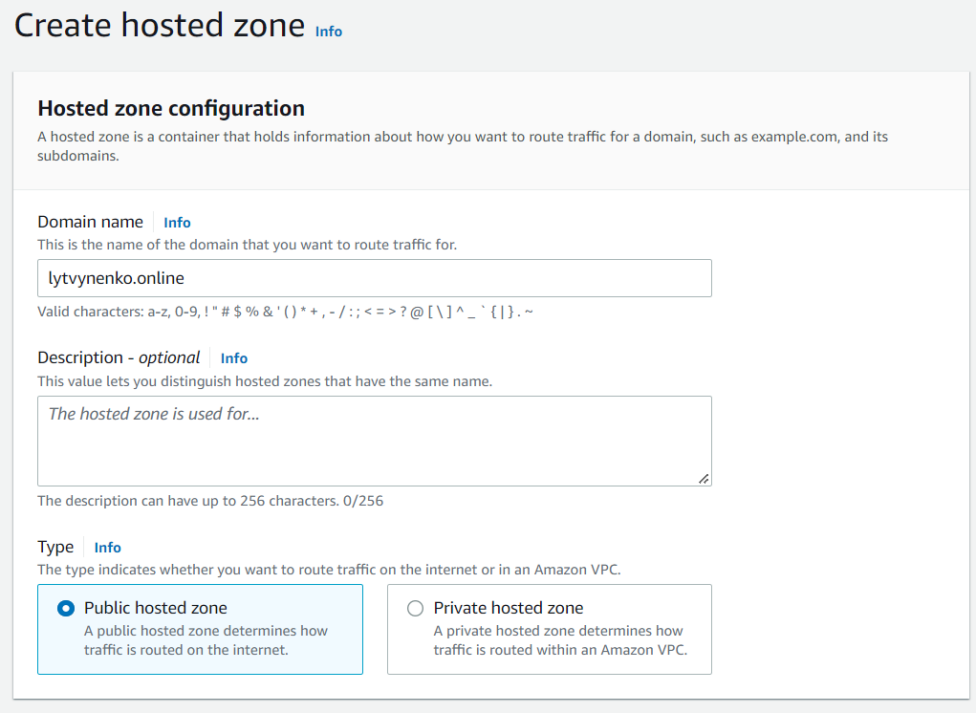
I open the Route 53 dashboard:



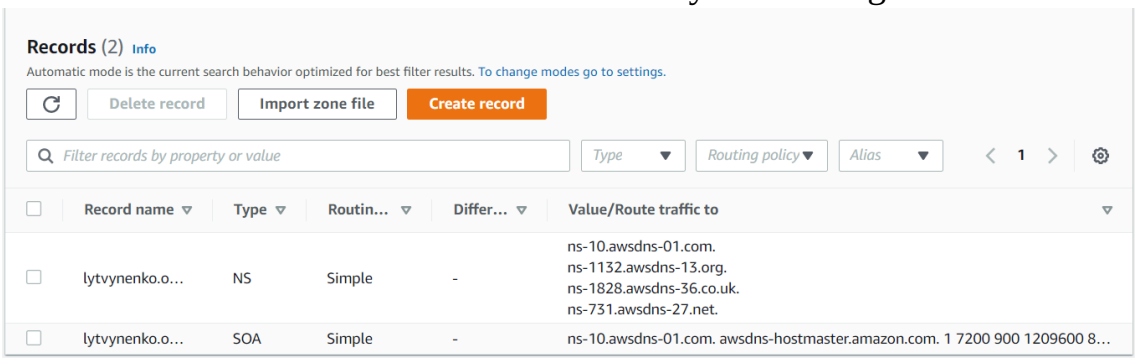
I create a new hosted zone:

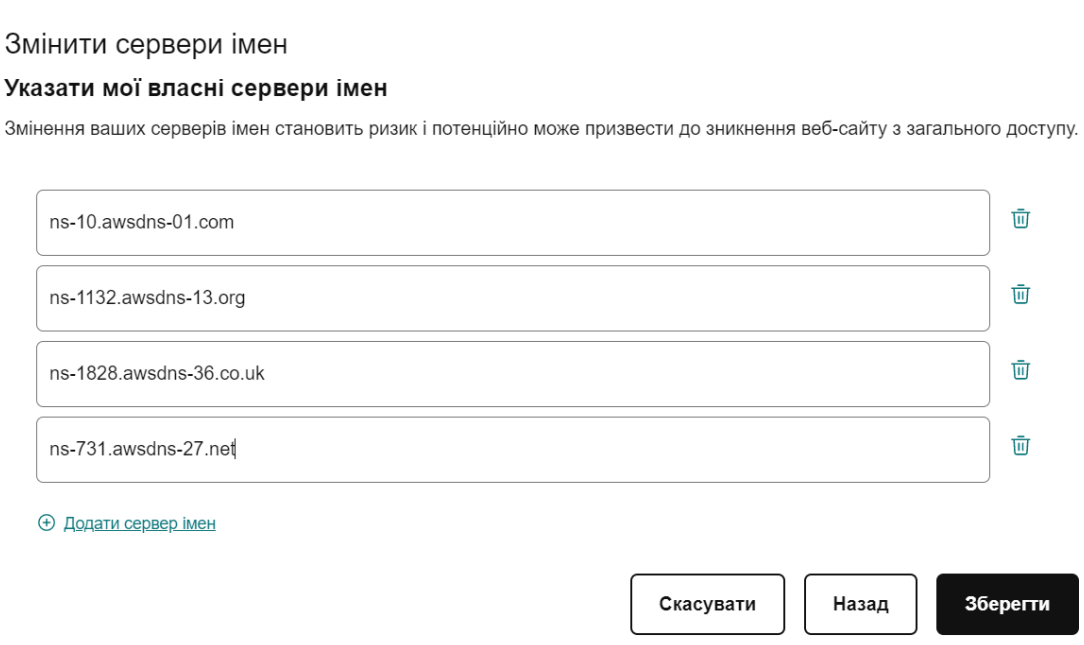


I enter a domain name and choose «Create hosted zone» button:

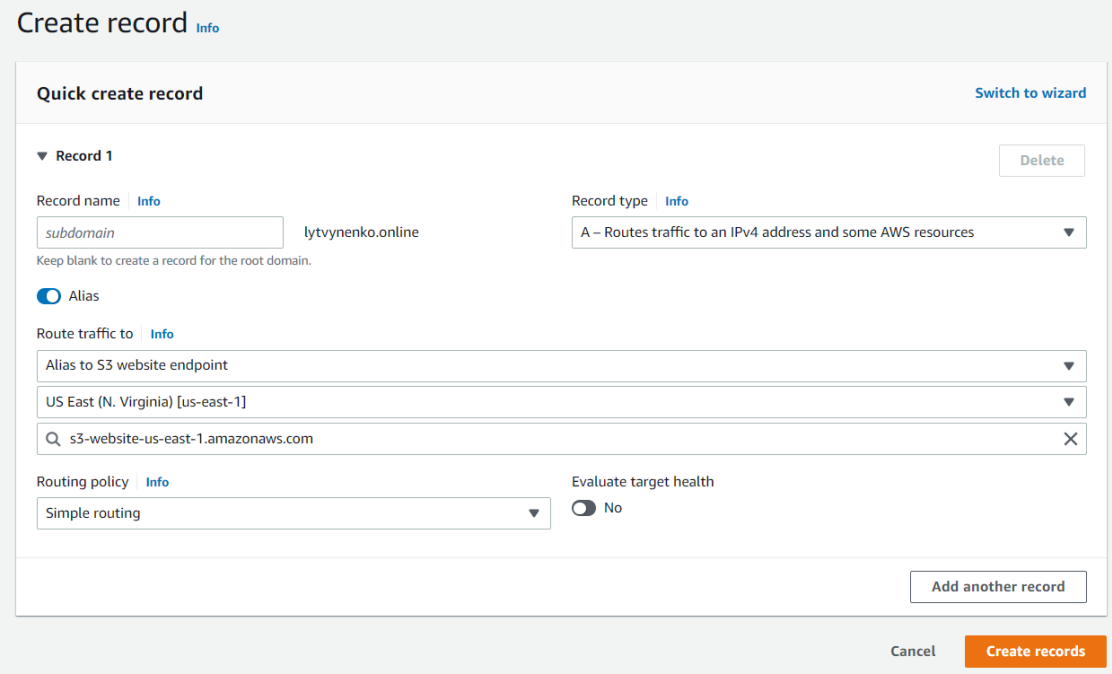


I add the received name server addresses to my domain registrar:

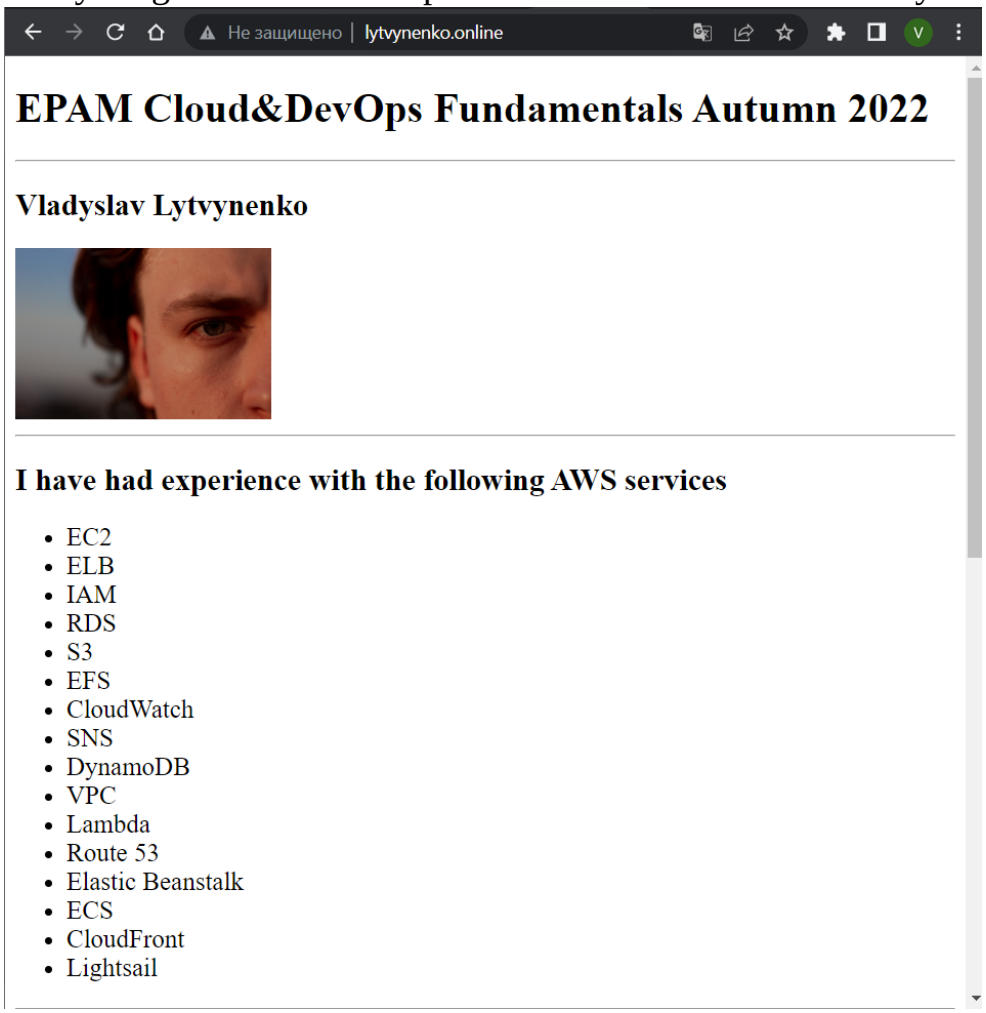




I create a new record in the DNS zone (pointing to the S3 bucket):



Everything works fine. It is possible to reach the S3 bucket by its domain name:



Completed AWS labs in this course

AWS Educate

- Introduction to Cloud 101
- Getting Started with Compute
- Getting Started with Storage

AWS hands-on tutorials

- Protect Data on Amazon S3 Against Accidental Deletion or Application Bugs Using S3 Versioning, S3 Object Lock, and S3 Replication
- Deploy a LAMP Stack Application to Amazon Lightsail
- Connecting a WordPress website to an Amazon Lightsail bucket and distribution
- Amazon EC2 Auto Scaling with EC2 Spot Instances
- Getting started using the Amazon S3 Glacier storage classes

AWS Skillbuilder

- AWS Cloud Practitioner Essentials: Core Services
- AWS Cloud Practitioner Essentials: Cloud Concepts

Amazon qwiklabs

- Security on AWS
- Introduction to Amazon DynamoDB
- Introduction to Amazon Elastic Block Store (Amazon EBS)
- Introduction to Amazon CloudFront
- Introduction to AWS Identity and Access Management (IAM)
- Introduction to Amazon Redshift
- Introduction to AWS Key Management Service
- Introduction to Amazon Simple Storage Service (S3)
- Troubleshooting connectivity using EC2 Serial Console

I delete the bucket and the DNS zone: