

Hands-on Cloud Computing Services

Lezione 4

Gabriele Russo Russo
University of Rome Tor Vergata, Italy

A.A. 2022/23



TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

AWS Storage Services

AWS offers various **storage** services, including:

- ▶ S3: Simple Storage Service
- ▶ EBS: Elastic Block Storage
- ▶ EFS: Elastic File System

Amazon S3

- ▶ Pricing (discussed during prev. lecture):
<https://aws.amazon.com/it/s3/pricing/>
- ▶ **Buckets and objects**
- ▶ Let's create a bucket using S3 console
- ▶ Bucket name must be unique across all AWS regions and accounts
- ▶ We can choose who can access objects and buckets:
https://docs.aws.amazon.com/it_it/AmazonS3/latest/dev/example-bucket-policies.html
- ▶ For Photogallery, we want everyone to read objects

We can reference an object like this:

<https://BUCKETNAME.s3.amazonaws.com/FILENAME>

Using S3 through the CLI

```
$ aws s3 ls
```

```
$ aws s3 ls s3://mybucket
```

```
$ aws s3 cp prova.txt s3://mybucket/
```

```
$ aws s3 ls s3://mybucket
```

```
$ aws s3 rm s3://mybucket/prova.txt
```

Third-party clients also available: e.g., *s3cmd*

Hosting Static Web Content on S3

- ▶ Objects in a public bucket can be accessed through HTTP
- ▶ You can use S3 to host static web content
 - ▶ a static website
 - ▶ the frontend of a web application
- ▶ To enable web hosting on a bucket: <https://docs.aws.amazon.com/AmazonS3/latest/userguide/EnableWebsiteHosting.html>

Boto: Python API for AWS

- ▶ *Boto is the AWS SDK for Python. It enables Python developers to create, configure, and manage AWS services, such as EC2 and S3. Boto provides an easy to use, object-oriented API, as well as low-level access to AWS services.*
- ▶ Similar APIs available for other languages as well
- ▶ We'll use **boto3**: `https://boto3.amazonaws.com/v1/documentation/api/latest/index.html`

Configuring boto3

- ▶ Boto configuration is similar to AWS CLI configuration (default region, ...)
- ▶ Important issue: providing **credentials** to Boto
 - ▶ Especially important when deploying applications on remote machines
- ▶ Several ways to provide credentials (and configs):
<https://boto3.amazonaws.com/v1/documentation/api/latest/guide/credentials.html>
- ▶ If AWS CLI has been configured on your PC, boto3 works out-of-the-box

Examples

1. List objects in our bucket: `s3list.py`
2. List EC2 instances: `ec2list.py`

Cloud Automation

We have introduced a few tools for automation:

- ▶ Ansible (with dynamic inventories)
- ▶ AWS CLI
- ▶ AWS SDK (e.g., boto3)

Enough for [infrastructure management](#)?

Infrastructure-as-Code (IaC)

- ▶ Define and manage the infrastructure by means of a set of **text files**, instead of a user interface (CLI, Web, ...)
- ▶ Use simple text files to describe your **resources** (e.g., VMs, security groups, networks)
- ▶ Update the files to update the infrastructure
- ▶ Benefits:
 - ▶ Reduced costs
 - ▶ Reduced risks
 - ▶ Faster operations
 - ▶ Important to enable DevOps practices

Terraform

- ▶ Free tool for IaC
- ▶ Available on Linux, macOS, Windows (www.terraform.io)
- ▶ Can be used with several platforms (AWS, Azure, VMWare, CloudFlare, ...)
- ▶ Infrastructure defined using the *HashiCorp Configuration Language* (HCL)
- ▶ Key concepts: Providers + Resources

Terraform + AWS

Requirements:

- ▶ AWS CLI installed and configured
- ▶ Terraform installed (I am using Terraform 1.3.x)

We create `example.tf` and run:

- ▶ `$ terraform init`
- ▶ `$ terraform validate # check syntax`
- ▶ `$ terraform apply`
- ▶ `$ terraform show`
- ▶ `$ terraform apply # nothing to do`

We now **update** `example.tf` adding a tag to the instance:

- ▶ Edit `example.tf` adding a tag to the instance
- ▶ `$ terraform apply`
- ▶ Edit `example.tf` changing the instance type
- ▶ `$ terraform apply`
- ▶ Let's destroy all the created resources: `terraform destroy`

Terraform: beyond this example

- ▶ Resource definitions not limited to EC2 instances!
- ▶ Remote storage for `tf.state`
- ▶ Versioning Terraform code (e.g., git repo)
- ▶ Variables to make code more reusable

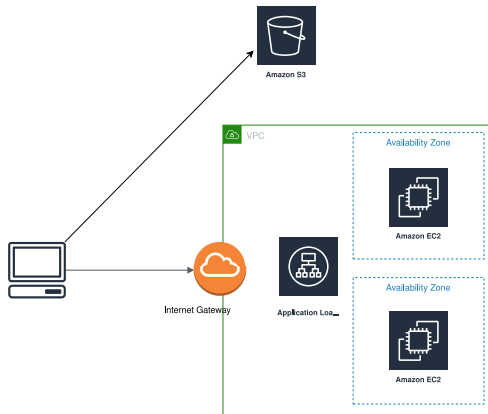
- ▶ IaC code solution by AWS
- ▶ Stack + Template (YAML/JSON) + Resources
- ▶ <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/GettingStarted.Walkthrough.html>

AWS CloudFormation: Example

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "A sample template",
  "Resources" : {
    "MyEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : "ami-0ff8a91507f77f867",
        "InstanceType" : "t2.micro",
        "KeyName" : "testkey",
        "BlockDeviceMappings" : [
          {
            "DeviceName" : "/dev/sdm",
            "Ebs" : {
              "VolumeType" : "io1",
              ...
            }
          }
        ]
      }
    }
  }
}
```


Example: Photogallery

- ▶ Extend PhotoGallery with the following features:
 - ▶ display pictures stored in a S3 bucket, along with their upload time
 - ▶ users can upload pictures



Solution

- ▶ Source code: `photogallery_v2`
- ▶ How to provide credentials to `boto3` to access the bucket?
 - ▶ Create a [IAM Role](#) for EC2
 - ▶ Attach the pre-defined `S3FullAccess` policy
 - ▶ Associate the EC2 instance(s) with the new role
- ▶ (Check alternative methods in the previous slides)

CloudFront

- ▶ CDN provided by AWS
- ▶ Easy to integrate with S3 buckets and ELBs

How to use it in Photogallery:

- ▶ Create a **distribution** for our S3 bucket
- ▶ Replace picture URLs as follows:

`http://bucketname.s3.amazonaws.com/imagename.jpg`

`http://distributionname.cloudfront.net/imagename.jpg`

Note: to delete a distribution, you need to Disable it first (and wait a couple of minutes)

Route 53

- ▶ DNS service by AWS
- ▶ Register a domain name
- ▶ Fine-grained control over your DNS zone
- ▶ Extra features (e.g., latency-based query handling)

AWS: Database Services

AWS provides several database-oriented services. Among them:

- ▶ DynamoDB (Key-Value NoSQL tables)
- ▶ Aurora (relational DBMS)
- ▶ ElastiCache (in-memory databases: Memcached, Redis)
- ▶ Neptune (graph database)
- ▶ Timestream (for time series)
- ▶ RDS (Relational Database Service): easily deploy MariaDB, Aurora, PostgreSQL, ...

AWS: Database Services

AWS provides several database-oriented services. Among them:

- ▶ DynamoDB (Key-Value NoSQL tables)
- ▶ Aurora (relational DBMS)
- ▶ ElastiCache (in-memory databases: Memcached, Redis)
- ▶ Neptune (graph database)
- ▶ Timestream (for time series)
- ▶ RDS (Relational Database Service): easily deploy MariaDB, Aurora, PostgreSQL, ...

We'll use DynamoDB to store picture metadata in PhotoGallery

DynamoDB

- ▶ Schemaless
- ▶ Tables, Items, Attributes
- ▶ Primary Key + (optional) Sorting Key
- ▶ 2 pricing models:
 - ▶ provisioned capacity (default)
 - ▶ on-demand
- ▶ 2 consistency models:
 - ▶ eventual
 - ▶ strong

Example: `dynamodb_example/`

Photogallery + DynamoDB

► Exercise: Use DynamoDB to store image tags

