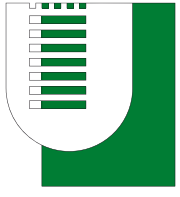
Hands-on Cloud Computing Services

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

PRATICA: creiamo bucket s3

- vado su aws, main page, ho lista bucket, faccio "crea bucket", mi chiede nome. nb: nome bucket è univoco a livello globale di AWS, non posso esserci altri nomi uguali. lascio le impostazioni default (enable/disable acl, impostazini di blocco dell'accesso pubblico per questo bucket == blocco qualcuno ! = me che può accedere a questi oggetti in R/W, non vuol dire che diventa pubblico, blocco solo gli accessi pubblici, con questa spunta non li autorizzo proprio. Noi la disattiviamo.
Poi creo bucket : se lo apro ho lista oggetti appartenenti, è vuota. posso caricare nuovi file, es caricare immagini, le carico. per ora è simile a farlo in gdrive. Ora ho lista oggetti aggiornata.

```
$ 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*

oltre alla console posso usare command line,
es: aws s3 ls vedo list
aws s3 ls s3://nomebucket lista dei file nel bucket
aws s3 cp prova.txt s3;//nomebucket/ carico prova.txt
con rm la tolgo

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

EXAMPLES pratica
s3 list stampa bucket che ho + contenuto.
importo libreria, mi collego a s3 (perchè scelgo io di usarla).
faccio for che itera su buckets.all, stampando il nome. c'è filtro sddc messo dal prof.

c'è anche ec2 list che stampa istanze ec2 attive.

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

Nel codice di Terraform: prime 10 righe standard, provider usato, sue impostazioni, profilo (account diversi, posso sceglierne uno predefinito).
ho poi my_sec_group con tipo e valore default: sto creando security group da usare, memorizzo id e ci metto valore di default.
my subnet ha id subnet.
sotto ho risorsa da configurare, "resource", tipo : aws istance, identificativo risorsa : example.
dentro ho info risorsa: ami, istance type, sec group (riusando my sec group), my subnet ed eventuali tag.

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`

Terraform + AWS

come lo uso?

nella cartella uso comando "terraform init", se specifico aws, scarica plugin aws.
poi comando "terraform apply", cioè lo applica. stampa piano con tutte le modifiche da fare per applicare la nostra config. alcune modifiche tipo istance type sono messe da noi, altre verrano perfezionate dopo le modifiche (es indirizzo ip). quando termina, crea istanzaa come la volevamo. da cli posso fare "terraform show". se rifaccio "ec2list.py" (codice di prima), trovo questa nuova istanza.

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`

il vantaggio è che se voglio cambiare qualcosa tocco il codice e via (es cambio nome o istance type). se terraform dice "updated in place" vuol dire che cambia qualche setting senza distruggere tutte (se cambio nome), o meno (cambio tipo istanza).
per cancellare risorse "terraform destroy", tutto ciò che definito nella cartella corrente e definito.

Terraform: beyond this example

abbiamo visto per EC2, ogni risorsa è ben documentata sul sito ufficiale. per ogni risorsa su terraform ho esempio abbastanza esaustivo.
esistono blocchi chiamati "data", parlano di qualcosa di tipo AWS-AMI, già esistente, che noi chiamiamo in un certo nome (es ubuntu).
posso fare query, come cercare immagine di ubuntu 20.04 con le specifiche del caso. Posso anche dereferenziarla sempre sfruttando AMI.

- 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

in tf.state terraform si appunta info su stato corrente infrastruttura, grazie a esso capisce differenze tra modifiche già presenti o meno. in caso condiviso? posso tenerlo in db. il codice è bene gestirlo con sw di versionamento tipo git.

AWS CloudFormation

ufficiale di aws, ma è più complesso, e poi terraform è meno vendor lock in. tutto questo per arrivare a EXAMPLE PHOTOGALLERY.

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

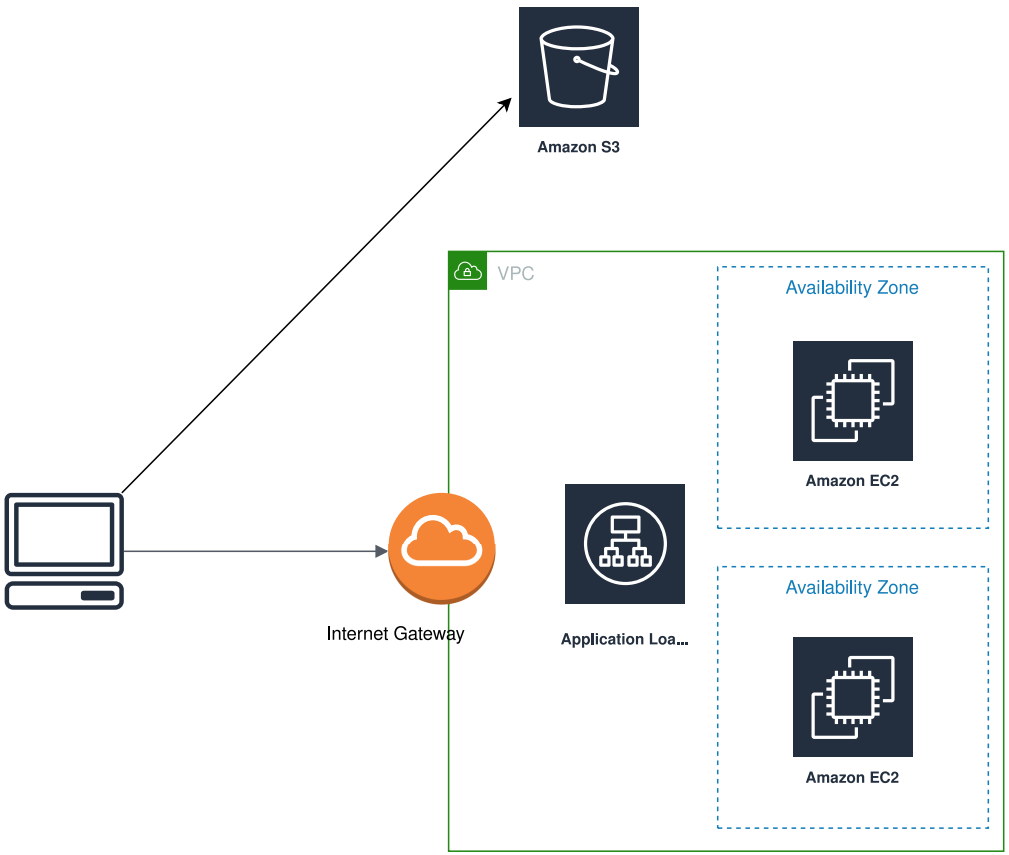
cosa possiamo fare in più? posso caricare immagini + far vedere immagini da s3 bucket. perchè immagini di s3 raggiungibili direttamente? basterebbe che application server legga da s3. non c'è più freccia da client e s3 (quella /) ma i miei workserver hanno carico ulteriore: richieste pagine + richieste file. me la evito: s3 lo pago, che consegnasse i file in maniera scalabile (quindi lascio /). se dovessi modificare le foto (metti scritte) allora non mi serve /.

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

Servizio senza schema, non definisco colonne etc, ogni tabella ha "item" elementi, con degli "attributi".
item: immagini, attributi: titolo|tag1|tag2|
devo indicare PK, essendo chiave valore, che identifica gli item. Posso usare Sorting key secondaria per velocizzare alcune chiave.
provisioned: limite alle richieste
ondemand: pago per gli utenti che li usano.

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

ora app funziona solo con IP, se volessi dominio? lo acquisto (tipo aruba), ma anche aws ha tale servizio: route 53 (tcp porta 53, è un riferimento). dove dovrei salvare metadati, tipo titoli foto? su s3, però c'è di meglio? un database. è punto di lancio per vedere servizi orientati ai db (mongo db). anche aws li fornisce, come DynamoDB o Elasticache (per info non persistenti, è una cache).
aurora è sostanzialmente mysql di aws (non so quante macchine vengono usate per il db). NOI USEREMO DYNAMO DB:

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