

Hands-on Cloud Computing Services

Lezione 1

Gabriele Russo Russo

University of Rome Tor Vergata, Italy

- **Amazon Web Services** è un insieme di servizi (>200) forniti dal provider cloud Amazon. A.A. 2021/22
- Nei prossimi giorni impareremo il sottoinsieme di servizi EC2, per istanziare macchine on-demand. È offerto anche il cloud storage.
- Per distribuire foto/video si usano servizi di **Content Delivery**, cioè distribuzione di repliche su più server. Comune quando si naviga in rete!



TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Overview

Hands-on Cloud Computing Services

- ▶ Cloud Computing services in action
 - ▶ Amazon Web Services
- ▶ Tools for cloud automation
 - ▶ Ansible
 - ▶ Terraform
 - ▶ ...
- ▶ Examples

Amazon Web Services (AWS)

- ▶ Most popular public cloud provider
- ▶ 200+ different services
- ▶ 25 regions
 - ▶ in Italy: Milano (since 2020)
- ▶ 2+ availability zones in each region
- ▶ 225 Edge locations across 47 countries
 - ▶ In Italy: Milano, Roma, Palermo
- ▶ Numbers keep growing...

zona disponibilità: partizione dell'infrastruttura (= datacenter indipendente in tutto dagli altri) per migliorare tolleranza ai guasti (es. terremoto). La mia applicazione può anche essere posizionata su >1 regioni.



risorse distribuite in modo granulare sul territorio, non selezionabile.
scopo: servizi come Content Delivery NO-SENSE se valido solo per le Availability Zone "locali" (es: non esco dall'Italia). uso allora le risorse di edge location.

Services

- ▶ **Computation** (EC2, Lambda, ...)
- ▶ **Storage** (S3, ...)
- ▶ **Machine Learning** (Rekognition, Lex, ...)
- ▶ **Networking and Content Delivery** (CloudFront, Route 53, ...)
- ▶ **Security** (Cognito, ...)
- ▶ ...

How to Interact with AWS?

✓ intuitiva

✗ non automatizzabile, se ripeto le stesse azioni % di sbagliare N°

► Web Console

► AWS CLI

► SDK for various languages (e.g., Boto3), uso librerie per interagire, scritte in Python, Java, Go.

► (Infrastructure-as-Code tools) livello altissimo di astrazione (non vedo né console né interface). tool per definire infrastruttura

Note: regardless of the chosen approach, you need to explicitly select the AWS Region you want to use. Services activated in one region cannot be managed within a different one.

Fondamentale anche per motivi di gestione crediti.

IAM users

- ▶ AWS Identity and Access Management (IAM)
- ▶ When you register to AWS, you get a **root** account
- ▶ IAM allows you to create and manage users within your account
- ▶ It's a good practice to use an IAM user (with full permissions) rather than the root account (if compromised, an IAM user can be deleted without losing the whole account)¹

¹https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_create.html

Amazon EC2

- ▶ Resizable compute capacity in the cloud
- ▶ Virtual machines = Instances, cioè con EC2 creo VM.
- ▶ Many different instance types, caratteristiche dell'istanza
 - ▶ Different amount of vCPUs, memory
 - ▶ Different storage
 - ▶ Different processors (ARM vs x86 CPUs, GPUs, ...)
 - ▶ **Different price!**
- ▶ On-demand / spot / reserved instances (3 tipi di modelli)
 - ▶ Starting from less than 0.01 \$/hour

Example

- ▶ Let's create an EC2 instance
- ▶ AMI: we can use Amazon Linux
- ▶ Instance type: pick something cheap (e.g., t3.micro/nano)
- ▶ Security group: create a new SG e' Firewall (insieme di regole)
 - ▶ Allow SSH and HTTP inbound traffic per definire traffico rete in entrata e in uscita.
- ▶ Shutdown behavior: Terminate
- ▶ Create (if necessary) a new key pair (pubblica / privata)
 - ▶ Store the private key in a safe place!
 - ▶ Public key automatically installed on the instance

Amazon tiene la Private
e ti fa scaricare
UNA VOLTA la pubblica

Connecting to the new instance

```
$ ssh -i <file.pem> ec2-user@<Public IP/Public DNS>
```

Alternativa: Gere coppia locale con **ssh-keygen** e carica la PUBBLICA.

N.B.: ssh permette di specificare default private key senza specificarla.

Monitor your Costs!

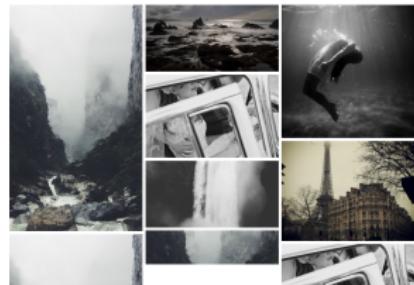
- ▶ The **Billing Dashboard** provides useful information to analyze paid and expected costs
- ▶ It is recommended to set an alarm so as to be notified whenever the expected monthly costs exceed a certain value:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/monitor_estimated_charges_with_cloudwatch.html
- ▶ ...and avoid things like this:
https://www.reddit.com/r/aws/comments/g1ve18/i_am_charged_60k_on_aws_without_using_anything/
- ▶ Note: estimated cost to reproduce steps from this lecture (EC2 + ELB) using services for a whole day: 1 EUR
Il nostro uso "universitario" non crea problemi di conto.

Example Application: Photogallery

- ▶ Web application, written in Python using Flask
- ▶ Just a toy: no security, no robustness, ...
- ▶ We'll use and extend it during the lectures
- ▶ You may develop your own customized version

SDCC Photo Gallery

View Photo



Roadmap:

Photogallery version 1: static set of images; upload not supported

...

Photogallery version N: image upload and tagging; image search; automatic resizing and tagging based on object recognition; ...

Deploying Photogallery on EC2

Running Photogallery

```
$ export FLASK_APP=galleryApp.py  
$ flask run -h 0.0.0.0 -p <numero di porta>  
$ # Note: \-- requires root privileges for port 80
```

or, using the script run.sh:

```
$ bash run.sh
```

- ▶ Create a new EC2 instance to deploy the app
- ▶ Connect via SSH to the instance:

```
$ ssh -i <file.pem> ec2-user@<Public IP/Public DNS>
```

Deploying Photogallery on EC2 (contd.)

- ▶ Install the required software:

```
$ sudo yum install python3  
$ sudo pip3 install flask
```

- ▶ Copy the app files from your PC using scp:

```
$ scp -i <chiaveprivata.pem> -r <cartellalocale> \  
ec2-user@<istanza ec2>:/home/ec2-user/
```

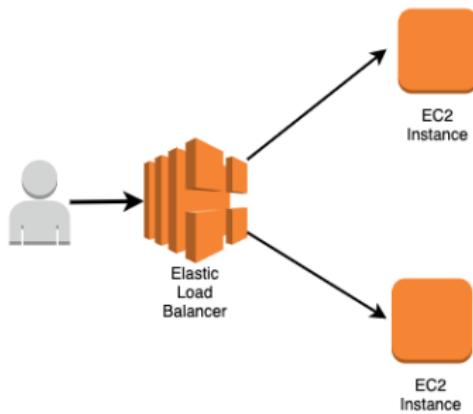
- ▶ Start the application:

```
$ cd photogallery/  
$ bash run.sh
```

- ▶ Open <http://EC2-PUBLIC-IP/> in a browser
- ▶ Test: what if we “close” port 80 in the security group?

Replicating App Instances

- ▶ Current configuration is neither scalable or fault-tolerant
- ▶ Let's run multiple replicas of the web server
- ▶ We need a **load balancer**



Preliminary Tasks

- ▶ We run the app as a systemd **service**, automatically started at boot

```
/etc/systemd/system/photogallery.service
```

```
[Unit]
```

```
Description=Simple systemd service for Photogallery.
```

```
[Service]
```

```
Type=simple
```

```
WorkingDirectory=/home/ec2-user/photogallery
```

```
ExecStart=/bin/bash /home/ec2-user/photogallery/run.sh
```

```
[Install]
```

```
WantedBy=multi-user.target
```

Preliminary Tasks (contd.)

Starting and enabling the service

```
$ sudo systemctl daemon-reload  
$ sudo systemctl start photogallery.service  
$ sudo systemctl enable photogallery.service
```

Register an AMI

We also create an **AMI** using a snapshot of the running instance. We will be able to re-use the AMI to create new instances where the application is already installed and configured to start.

Preliminary Tasks (contd.)

Note: each AMI is associated with a [snapshot](#) of the root ELB volume attached to the instance. Keeping this snapshot has a (small) cost: <https://aws.amazon.com/premiumsupport/knowledge-center/ebs-snapshot-billing/>

Run Commands at Launch: cloud-init and User Data

- ▶ Creating a custom AMI allowed us to create new EC2 instances without manually configuring the application every time
- ▶ An alternative (and smarter) approach exists
- ▶ Cloud providers allow you to run commands when instances are launched:
<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html>
- ▶ In AWS, you can use the **User Data** option to specify:
 - ▶ a Bash script
 - ▶ **cloud-init** directives
(<https://cloudinit.readthedocs.io/en/latest/>)

Next Step

We will configure Elastic Load Balancer in the next lecture