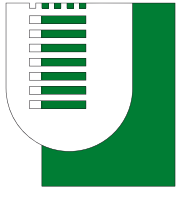
Hands-on Cloud Computing Services

Lezione 2

Gabriele Russo Russo

University of Rome Tor Vergata, Italy

A.A. 2021/22



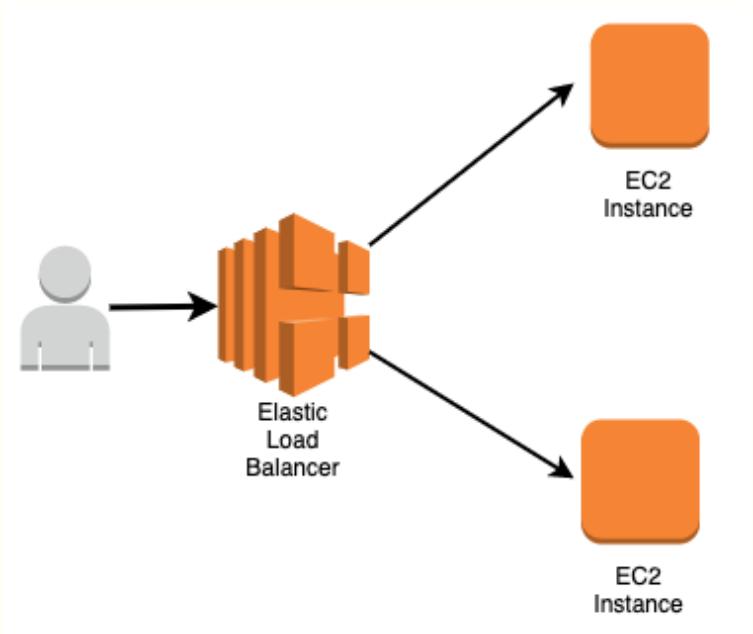
TOR VERGATA

UNIVERSITÀ DEGLI STUDI DI ROMA

Recap

- EC2
- Photogallery on EC2
- Custom AMI

Next step:



Amazon VPC

- Provision **logically isolated** sections of the AWS cloud
- Define virtual networks (IP ranges, subnets, gateways,...)
- May create a hardware Virtual Private Network (VPN) connection between your own datacenter and your VPC (**hybrid cloud**)
- **No additional charges** for creating and using the VPC itself.
- So far, we have used the default VPC

Amazon VPC: main building blocks

- In each AZ, we can define one or more **subnets**
- **Routing Tables** attached to subnets
- **Internet Gateway**

Basic VPC Configuration

- Create a new **Virtual Private Cloud (VPC)**
- We associate a block of (private) IP addresses to the VPC
  - Subnets will be created within this block of addressess
  - We can pick, e.g., `10.0.0.0/16`
- We can create **subnets**: each subnet is associated with an Availability Zone (AZ)
- Let's pick an AZ and create a subnet (e.g., `10.0.1.0/24`)
- If you want (for debugging), you can require that EC2 instances in the subnet are also assigned a public IP address
- Create an **Internet Gateway** (IG) to allow instances in the VPC to reach Internet; **associate** it with the VPC
- Create a **Route Table** for the VPC and **attach** it to the subnet(s)
- Add a new rule in the table: `0.0.0.0\0` – target: IG
- Repeat the above steps for each subnet you want.

Elastic Load Balancing (ELB)

- ELB automatically distributes incoming traffic across multiple targets (e.g., EC2 instances, containers, and IP addresses) in one or more Availability Zones
- It monitors the health of its registered targets and routes traffic only to the healthy targets
- 4 types of ELB:
  - Application Load Balancer (layer 5)
  - Network Load Balancer (layer 4)
  - Gateway Load Balancer (layer 3)
  - Classic Load Balancer (legacy)
- We'll use the Application LB today

ELB Configuration

- Create an ELB instance listening for HTTP requests on port 80
- Health check: use HTTP requests on port 80 with path /
- ELB needs a security group: configure one to accept traffic on port 80
- Create a few EC2 instances using our custom AMI in our subnets
- Register the instances to the ELB
- Wait a few minutes (DNS...) and then try to connect at the ELB URL with the browser

ELB Configuration

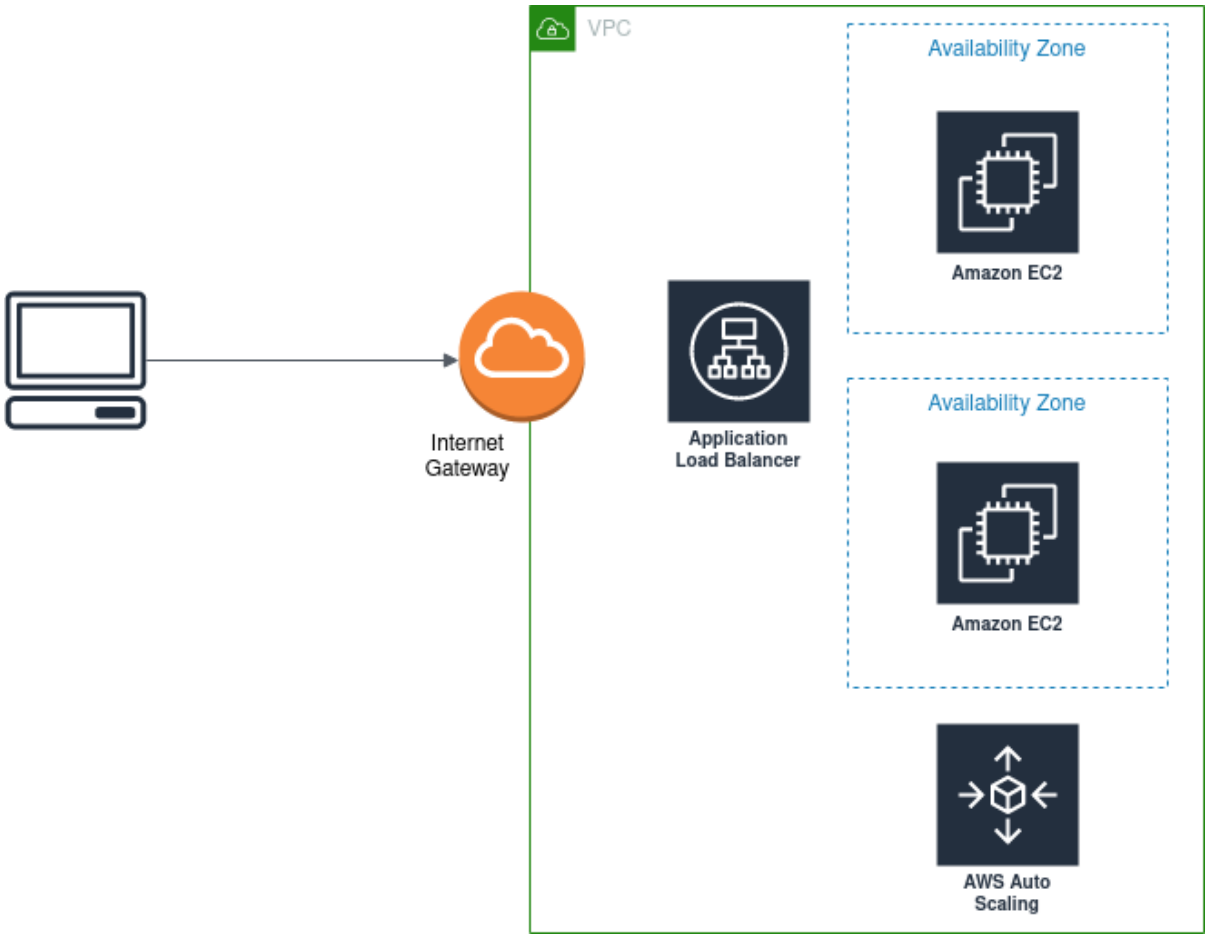
- Create an ELB instance listening for HTTP requests on port 80
- Health check: use HTTP requests on port 80 with path `/`
- ELB needs a security group: configure one to accept traffic on port 80
- Create a few EC2 instances using our custom AMI in our subnets
- Register the instances to the ELB
- Wait a few minutes (DNS...) and then try to connect at the ELB URL with the browser

Note:

- EC2 instances don't need a public IP address any more
- EC2 instances can now use a stricter security group:
  - Allowed source: `0.0.0.0/0` → `<ID of ELB sec group>`

Auto scaling

- We want to dynamically provision the number of active instances
- Let's use the Auto Scaling service of EC2



Auto Scaling + Photogallery

- Before starting, terminate manually launched instances
- Create a *Launch Configuration* for Photogallery
- Create an Auto Scaling Group that uses the new Launch Configuration
- Specify the VPC and the subnets where new instances should be launched
- Enable load balancing, associating the group with our ELB
- Set minimum and maximum number of instances (e.g., 2 and 5)
- Set an auto scaling policy
- Verify that new instances are automatically created

AWS CLI

- AWS provides a *Command Line Interface* to interact with AWS services
- Faster interaction compared to web console
- Installation: check official docs
- Before usage, we need to configure:
  - `AWS Access Key ID` and `AWS Secret Access Key`
  - default region to use (e.g., `us-east-1`)
  - output format (`json`, `text`)
- AWS CLI can be configured by:
  - running `aws configure`, or
  - editing `~/.aws/config` and `~/.aws/credentials`
- Available commands well documented on AWS website

AWS CLI: example

Create a new security group in our VPC:

```
$ aws ec2 create-security-group --group-name my-sg \
--description "My security group" --vpc-id vpc-12345
```

We can see the properties of any SG:

```
$ aws ec2 describe-security-groups --group-ids <groupId>
```

Set inbound traffic rules:

```
$ aws ec2 authorize-security-group-ingress --group-id <ID> \
    --protocol tcp --port 22 --cidr 0.0.0.0/0
$ aws ec2 authorize-security-group-ingress --group-id <ID> \
    --protocol tcp --port 80 --cidr 0.0.0.0/0
$ aws ec2 describe-security-groups --group-ids <ID DEL GRUPPO>
```

AWS CLI: example

```
$ aws ec2 run-instances --image-id <ID AMI> --count 1 \
        --instance-type t2.nano \
        --key-name <MyKeyPair> --security-group-ids <sgId> \
        --subnet-id <subnetId> --associate-public-ip-address
```

We can associate the instance with a tag:

```
$ aws ec2 create-tags --resources <instID> \
          --tags Key=Name,Value=SDCC
```

We can get information about active instances:

```
$ aws ec2 describe-instances \
          --filters "Name=tag:Name,Values=SDCC"
$ aws ec2 describe-instances \
          --filters "Name=instance-type,Values=t2.nano"
```

To terminate the instance:

```
$ aws ec2 terminate-instances --instance-ids <ID>
```

Exercise

- Create a script to destroy all the active EC2 instances.
- Create a script to destroy all the active EC2 instances with tag “Name=SDCC”