

# AWS Lab Work

Boris TEABE  
boris.teabe@inp-toulouse.fr

## EC2 Service of AWS

The screenshot shows the AWS Cost Explorer interface with a search bar at the top containing the query 'ec2'. A red box highlights the search bar. Below the search bar, the results are categorized into 'Services' and 'Features'.

**Services** (8 results):

- EC2 (Virtual Servers in the Cloud)
- EC2 Image Builder (A managed service to automate build, customize and deploy OS images)
- AWS Compute Optimizer (Recommend optimal AWS Compute resources for your workloads)
- AWS Firewall Manager (Central management of firewall rules)

**Features** (46 results):

- Dashboard (EC2 feature)
- Limits (EC2 feature)
- AMIs (EC2 feature)
- Export snapshots to EC2 (Lightsail feature)

## Security group creation (firewall)

Security Groups (1/3) [Info](#)

Name	Security group ID	Security group name	VPC ID	Description	Owner	Inbound rules count	Outbound rules count
sg-018d18e7492f62bd2	default	vpc-0c515531372951e95	default VPC security group	905785452291	1 Permission entry	1 Permission entry	1 Permission entry
sg-01146055127e14e97	launch-wizard-1	vpc-0c515531372951e95	launch-wizard-1 created by	905785452291	1 Permission entry	1 Permission entry	1 Permission entry
<input checked="" type="checkbox"/> sg-00e75b6a09db6ee6f	launch-wizard-2	vpc-0c515531372951e95	launch-wizard-2 created by	905785452291	0 Permission entries	1 Permission entry	1 Permission entry

[Actions](#) [Export security groups to CSV](#) [Create security group](#)

Details Inbound rules Outbound rules Tags

You can now check network connectivity with Reachability Analyzer [Run Reachability Analyzer](#)

Inbound rules [Edit inbound rules](#)

No security group rules found

Create security group [Info](#)

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

**Basic details**

Security group name [Info](#)  
MyWebServerGroup

Description [Info](#)  
Allows SSH access to developers

VPC [Info](#)  
vpc-0c515531372951e95

**Inbound rules** [Info](#)

This security group has no inbound rules.

Add rule

**Outbound rules** [Info](#)

Type [Info](#) Protocol [Info](#) Port range [Info](#) Destination [Info](#) Description - optional [Info](#)

All traffic All All Custom 0.0.0.0/0

Feedback Looking for language selection? Find it in the new Unified Settings [Feedback](#)

Security rules. In a real DC, you should be careful when defining the rules

Inbound rules [Info](#)

Type [Info](#) Protocol [Info](#) Port range [Info](#) Source [Info](#) Description - optional [Info](#)

All traffic All All Anywhere-1... 0.0.0.0/0

Add rule

Outbound rules [Info](#)

Type [Info](#) Protocol [Info](#) Port range [Info](#) Destination [Info](#) Description - optional [Info](#)

All traffic All All Custom 0.0.0.0/0

Add rule

Tags - optional

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

[Cancel](#) [Create security group](#)

# Create a new VM

<https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Home>

New EC2 Experience  
Opt out of the new experience

### EC2 Dashboard

- EC2 Global View
- Events
- Tags
- Limits
- Instances **New** (selected)
- Instance Types
- Launch Templates
- Spot Requests
- Savings Plans
- Reserved Instances **New**
- Dedicated Hosts
- Scheduled Instances
- Capacity Reservations
- Images
- AMIs **New**
- AMI Catalog
- Elastic Block Store
- Volumes **New**
- Snapshots **New**
- Lifecycle Manager **New**
- Network & Security
- Security Groups
- Elastic IPs

Feedback Looking for language selection? Find it in the new Unified Settings [\[Alt+S\]](#)

**Resources**

You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Instances (running)	0
Dedicated Hosts	0
Elastic IPs	0
Instances	0
Key pairs	0
Load balancers	0
Placement groups	0
Security groups	1
Snapshots	0
Volumes	0

(1) Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. Learn more

**Launch Instance**

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

**Service health**

Region: US East (N. Virginia) Status: This service is operating normally

**Zones**

Zone name	Zone ID
us-east-1a	use1-az4
us-east-1b	use1-az6
us-east-1c	use1-az1
us-east-1d	use1-az2
us-east-1e	use1-az5

**Scheduled events**

US East (N. Virginia) No scheduled events

**Migrate a server**

Launch Instance Migrate a server

Note: Your instances will launch in the US East (N. Virginia) Region

**Explore AWS**

**Account attributes**

Supported platforms VPC Default VPC vpc-0c513551372951e95 Settings EBs encryption Zones EC2 Serial Console Default credit specification Console experiments

**Explore AWS**

**Name and tags**

Name: e.g. My Web Server Add additional tags

**Application and OS Images (Amazon Machine Image)**

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search: Search our full catalog including 1000s of application and OS images

Select your OS

Quick Start

Amazon Linux macos Ubuntu Windows Red Hat S Amazon Machine Image (AMI) Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type Free tier eligible

Browse more AMIs Including AMIs from AWS Marketplace and the Community

**Feedback** Looking for language selection? Find it in the new Unified Settings [\[Alt+S\]](#)

<https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances>

You've opted into the new launch experience. You can return to the previous version, but next time you log in, you'll automatically be opted into the new experience. Find out more or send us feedback. Starting October 1, 2022, we'll begin decommissioning the previous version.

EC2 > Instances > Launch an instance

**Launch an instance**

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

**Name and tags**

Name: e.g. My Web Server

**Application and OS Images (Amazon Machine Image)**

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search: Search our full catalog including 1000s of application and OS images

Select your OS

Quick Start

Amazon Linux macos Ubuntu Windows Red Hat S Amazon Machine Image (AMI) Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type Free tier eligible

Browse more AMIs Including AMIs from AWS Marketplace and the Community

**Summary**

Number of instances: 1

Software image (AMI): Amazon Linux 2 Kernel 5.10 AMI... read more ami-05f9a04d483e12376

Virtual server type (instance type): t2.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GB

**Free tier** In your first year includes 750 hours of t2.micro (or 13 million in Regions where t2.micro is unavailable) instance usage on free tier AMIs per month, 50 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GiB of bandwidth to the internet.

Cancel Launch instance

**Important because it is free tiers**

<https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances>

**Instance type**

Instance type: t2.micro Family: Compute 1 vCPU 1 GiB Memory On-Demand Linux pricing: 0.0116 USD per Hour On-Demand Windows pricing: 0.0162 USD per Hour

**Key pair (login)**

Don't forget to save the generated key.

Key pair name - required Create new key pair

**Network settings**

Network: vpc-0c513551372951e95 Subnet: No preference (Default subnet in any availability zone) Auto-assign public IP: Enabled Firewall (security groups): Create security group Select existing security group

**Feedback** Looking for language selection? Find it in the new Unified Settings [\[Alt+S\]](#)

<https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances>

**Firewall (security group)**

Create security group Select existing security group

We'll create a new security group called "launch-wizard-1" with the following rules:

Allow SSH traffic from To help you connect to your instance

Allow HTTP traffic from the internet To set up an endpoint, for example when creating a web server

Allow HTTP traffic from the internet To set up an endpoint, for example when creating a web server

Rules of source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses.

**Configure storage**

1x 8 GiB gp2 Root volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

Add new volume

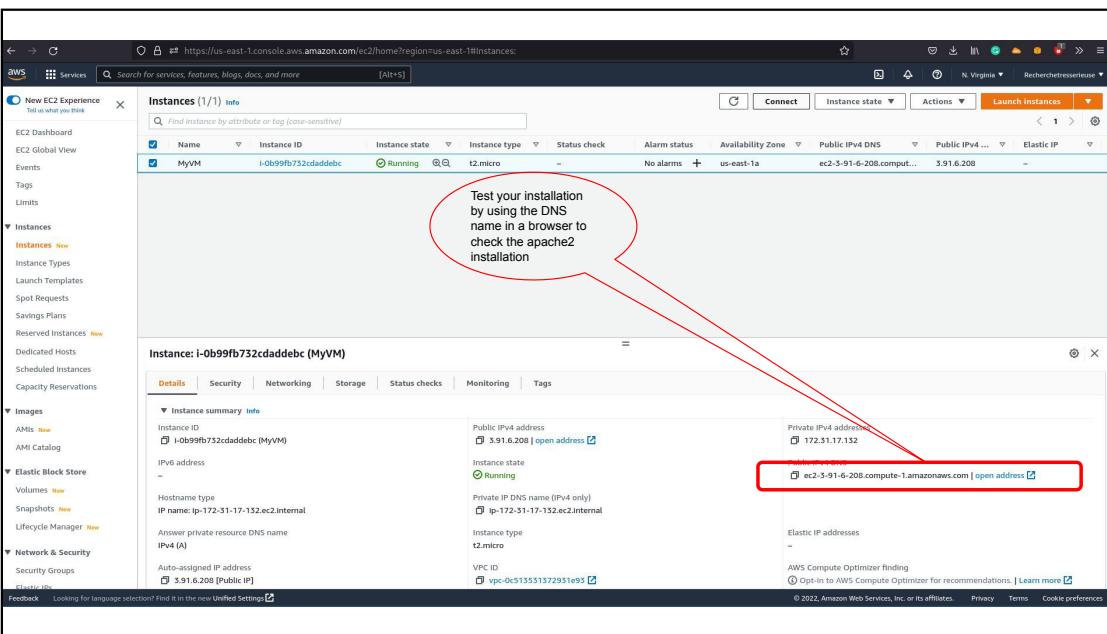
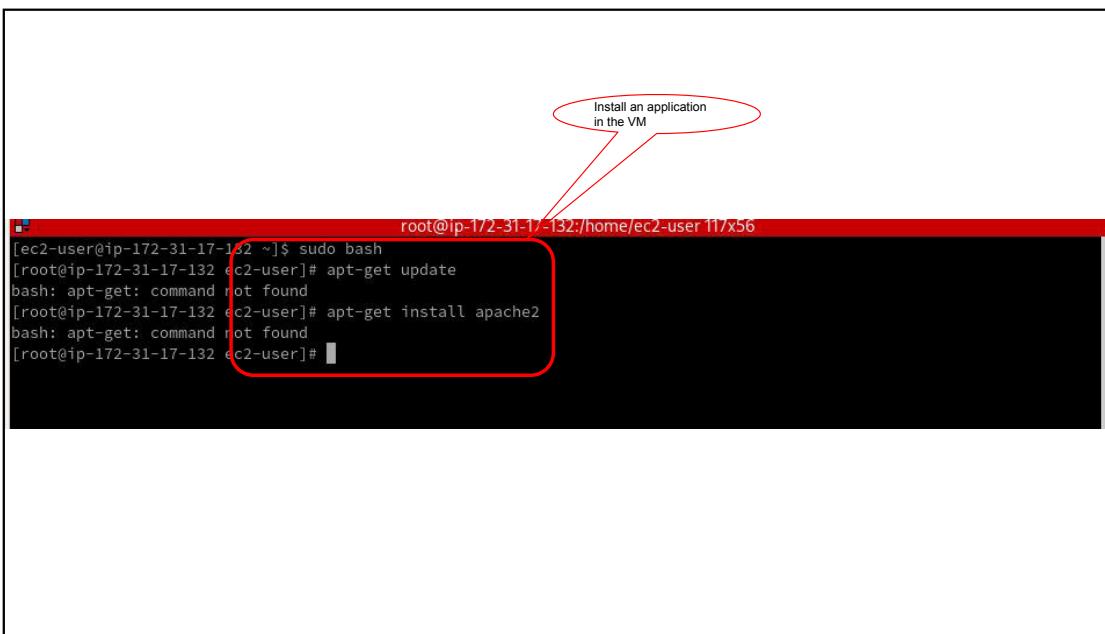
0 x File systems

**Advanced details**

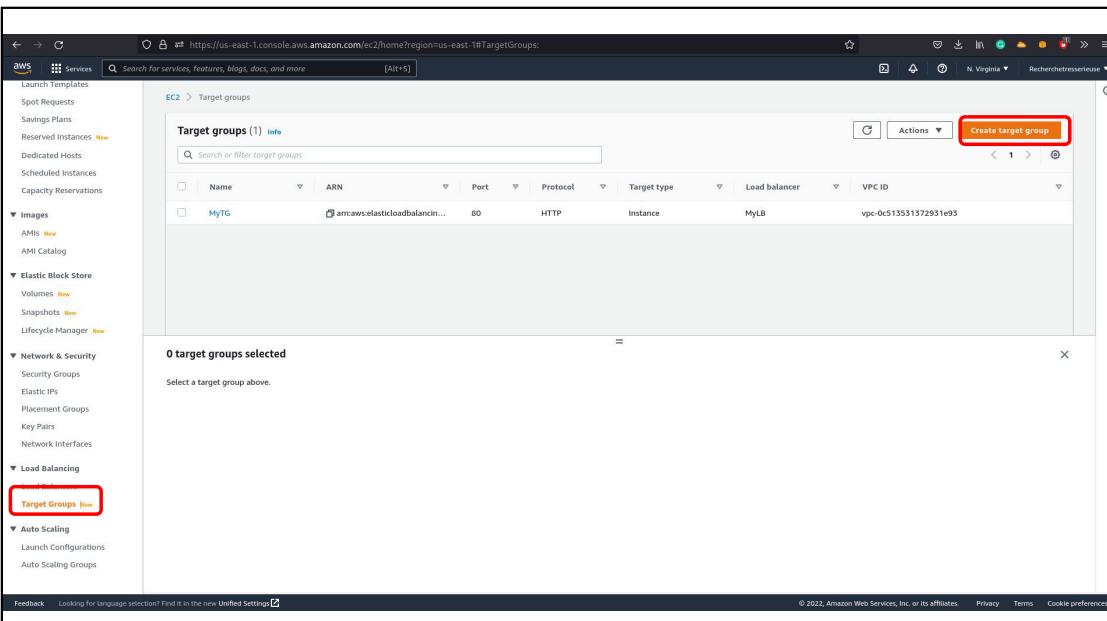
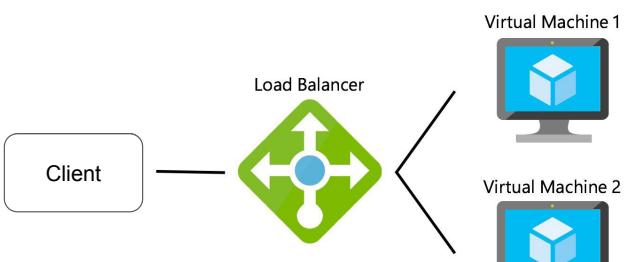
Cancel Launch instance

The screenshot shows two browser tabs. The left tab is titled "Launch an Instance" and displays a success message: "Successfully initiated launch of instance i-0b99fb732cdaddebc". Below this is a "Next Steps" section with links to "Get notified of estimated charges", "How to connect to your instance", and "View more resources to get you started". An orange "View all instances" button is highlighted with a red box. The right tab is titled "Instances (1) Info" and shows a single instance named "MyVM" with the ID "i-0b99fb732cdaddebc", status "Pending", type "t2.micro", and IP "ec2-3-91-6-208.compute-1.amazonaws.com". A red box highlights the instance table.

The screenshot shows two browser tabs. The left tab is titled "Connect to instance" and lists connection options: "EC2 Instance Connect", "Session Manager", and "SSH client" (which is highlighted with a red box). It includes instructions for connecting via SSH and an example command: "ssh -i \"MyVM.pem\" ec2-user@ec2-3-91-6-208.compute-1.amazonaws.com". The right tab is a terminal window showing an SSH session. The user runs "ssh -i \"MyVM.pem\" ec2-user@ec2-3-91-6-208.compute-1.amazonaws.com" and is prompted to add the host key. They accept and enter the password "MyVM". The session then connects to an Amazon Linux 2 AMI instance, showing the root prompt and a package update command.



# Create a Load balancer



EC2 > Target groups > Create target group

### Specify group details

Step 1: Specify group details

Your load balancer routes requests to the targets in a target group and performs health checks on the targets.

#### Basic configuration

Settings in this section cannot be changed after the target group is created.

##### Choose a target type

- Instances
  - Supports load balancing to instances within a specific VPC.
  - Facilitates scaling to multiple instances and various interfaces on the same instance.
  - Offers flexibility with microservice-based architectures, simplifying inter-application communication.
  - Supports IPv4 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.
- IP addresses
  - Supports load balancing to VPC and non-VPC resources.
  - Facilitates scaling to multiple IP addresses and various interfaces on the same instance.
- Lambda function
  - Facilitates routing to a single Lambda function.
  - Accessible to Application Load Balancers only.
- Application Load Balancer
  - Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
  - Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

Protocol: HTTP Port: 80

Feedback: Looking for language selection? Find it in the new Unified Settings. © 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

EC2 > Target groups > Create target group

### Create Target Group

Step 1: Create Target Group

VPC: Select the VPC with the instances that you want to include in the target group.

- vpc-0c513531372931e93

Protocol version:  HTTP1

Health checks: Health check protocol:  HTTP

Tags - optional: Consider adding tags to your target group. Tags enable you to categorize your AWS resources so you can more easily manage them.

Cancel Next

Feedback: Looking for language selection? Find it in the new Unified Settings. © 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

EC2 > Target groups > Create target group

### Register targets

This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group, you must register your targets.

#### Available instances (1/1)

Instance ID	Name	State	Security groups	Zone	Subnet ID
i-0b99fb732cdaddebc	MyVM	running	launch-wizard-1	us-east-1a	subnet-0145589042af62c85

Add the VM to the target group

Ports for the selected instances: 80

Include as pending below

#### Review targets

Targets (0)

All	Filter resources by property or value							
Remove	Health status	Instance ID	Name	Port	State	Security groups	Zone	Subnet ID

No instances added yet

Specify instances above, or leave the group empty if you prefer to add targets later.

Feedback: Looking for language selection? Find it in the new Unified Settings. © 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

EC2 > Target groups > Create target group

### Create Load Balancer

Actions: Create Load Balancer

Launch Templates: None found

Images: None found

Network & Security: None found

Load Balancers: None found

Select a load balancer

Feedback: Looking for language selection? Find it in the new Unified Settings. © 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

<https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#SelectCreateELBWizard>

EC2 > Load balancers > Select load balancer type

Select load balancer type

A complete feature-by-feature comparison along with detailed highlights is also available. [Learn more](#)

**Load balancer types**

**Application Load Balancer** [Info](#)

Choose an Application Load Balancer when you need a flexible feature set for your applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architecture, including microservices and serverless functions.

**Network Load Balancer** [Info](#)

Choose a Network Load Balancer when you need ultra-high performance, TLS offloading at scale, centralized certificate deployment, support for UDP, and static IP addresses for your targets. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second securely while maintaining ultra-low latencies.

**Gateway Load Balancer** [Info](#)

Choose a Gateway Load Balancer when you need to deploy and manage a fleet of third-party virtual appliances that support GENEVE. These appliances enable you to improve security, compliance, and policy controls.

**Create**

Feedback Looking for language selection? Find it in the new Unified Settings [\[?\] \[Feedback\]](#)

© 2022, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

<https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateALBWizard>

EC2 > Load balancers > Create Application Load Balancer

**Create Application Load Balancer** [Info](#)

The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 Instances, microservices, and containers, based on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the target group for the rule action.

**How Application Load Balancers work**

**Basic configuration**

**Load balancer name** [Info](#)  
Name must be unique within your AWS account and cannot be changed after the load balancer is created.

**Scheme** [Info](#)  
Select the scheme for your load balancer after the load balancer is created.  
 Internet-facing  
An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. [Learn more](#)  
 Internal  
An internal load balancer routes requests from clients to targets using private IP addresses.

**IP address type** [Info](#)  
Select the type of IP addresses that your subnets use.  
 IPv4  
Recommended for internal load balances.  
 Dualstack  
Includes IPv4 and IPv6 addresses.

**Network mapping** [Info](#)

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

Feedback Looking for language selection? Find it in the new Unified Settings [\[?\] \[Feedback\]](#)

© 2022, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

<https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateALBWizard>

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

**VPC** [Info](#)  
Select the virtual private cloud (VPC) for your targets. Only VPCs with an internet gateway are enabled for selection. The selected VPC cannot be changed after the load balancer is created. To confirm the VPC for your targets, view your [target groups](#) [\[?\] \[Feedback\]](#)  
 vpc-dc513553372931e93  
IPv4: 172.31.0.0/16

**Mapping** [Info](#)  
Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.  
 us-east-1a  
 us-east-1b  
 us-east-1c  
 us-east-1d  
 us-east-1e  
 us-east-1f

**Security groups** [Info](#)  
A security group is a set of firewall rules that control the traffic to your load balancer.  
**Security groups**  
 Select up to 5 security groups  
 Create new security group [\[?\] \[Feedback\]](#)

Feedback Looking for language selection? Find it in the new Unified Settings [\[?\] \[Feedback\]](#)

© 2022, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

<https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateALBWizard>

EC2 > Load balancers > Select your security group

**Security groups** [Info](#)  
A security group is a set of firewall rules that control the traffic to your load balancer.

**Security groups**  
 Select up to 5 security groups  
 Create new security group [\[?\] \[Feedback\]](#)  
default sg-018d18d7492fd2bd2  
VPC vpc-dc513553372931e93

**Listeners and routing** [Info](#)  
A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

**Listener HTTP:80**

Protocol	Port
HTTP	:80

**Default action** [Info](#)  
Forward to

Feedback Looking for language selection? Find it in the new Unified Settings [\[?\] \[Feedback\]](#)

© 2022, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

<https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1>CreateALBWizard>

**Add-on services - optional**  
Additional AWS services can be integrated with this load balancer at launch. You can also add these and other services after your load balancer is created by reviewing the "Integrated Services" tab for the selected load balancer.

**AWS Global Accelerator info**  
Create an accelerator to get static IP addresses and improve the performance and availability of your applications. [Additional charges apply](#)

**Tags - optional**  
Consider adding tags to your load balancer. Tags enable you to categorize your AWS resources so you can more easily manage them. The 'Key' is required, but 'Value' is optional. For example, you can have Key = production-webserver, or Key = webserver, and Value = production.

**Summary**  
Review and confirm your configurations. [Estimate cost](#)

**Basic configuration**  
Load balancer name not defined  
 • Internet-facing  
 • IPv4  
**Security groups**  
 • default  
 sg-018d18d7492fd2bd2  
**Network mapping**  
 VPC vpc-0c513531372931e93  
 Subnet not defined  
**Listeners and routing**  
 • HTTP:80 defaults to Target group not defined

**Add-on services**  
None  
**Tags**  
None

**Attributes**  
Certain default attributes will be applied to your load balancer. You can view and edit them after creating the load balancer.

**Create load balancer**

<https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancers:search=MyLB:sort=loadBalancerName>

**Create Load Balancer**

Name	DNS name	State	VPC ID	Availability Zones	Type	Created At	Monitoring
MyLB	MyLB-112483625.us-east-1...	Provisioning	vpc-0c513531372931e93	us-east-1a, us-east-1a	application	September 16, 2022 at 6:41:...	

**Load balancer: MyLB**

**Basic Configuration**

- Name: MyLB
- ARN: arn:aws:elasticloadbalancing:us-east-1:905785452299:loadbalancer/app/MyLB/f97b47916c439ea9
- DNS name: MyLB-112483625.us-east-1.elb.amazonaws.com (A Record)

**Use the DNS name of the load balancer to check your configuration**

To Do: Add a new VM to your LB

- Therefore, clone your running VM and add the new VM to the target Group.
- Check if your load balancer balances the load between the two VMs (you can use the curl command from your terminal: **curl -sv LoadBalancer DNSName**)
  - Don't forget to connect to the new VM and check that apache2 is installed.

<https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances>

**Instances (1 / 1) Info**

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
MyVM	i-0b99fb732cdaddebc	Running	t2.micro	2/2 checks passed	No alarms	+ us-east-1a	ec2-3-91-6-208.compute...

**Image and template**

**Launch instances**

**Instance: i-0b99fb732cdaddebc (MyVM)**

**Details**

**Instance summary**

Instance ID: i-0b99fb732cdaddebc (MyVM)  
 Public IPv4 address: 3.91.208 | open address  
 Instance state: Running  
 Instance type: t2.micro  
 Status check: 2/2 checks passed  
 Alarm status: No alarms  
 Availability Zone: us-east-1a  
 Public IPv4 DNS: ec2-3-91-6-208.compute...

**Select your cookie preferences**

Accept all cookies  
 Continue without accepting  
 Customize cookies

# S3 and Lambda function

- Add a bucket to S3

The screenshot shows the AWS Lambda Management Console with a search bar at the top containing 'S3'. The search results are displayed under the 'Services' section. The first result, 'S3 Scalable Storage in the Cloud', is highlighted with a red box. Below it are other services like 'S3 Glacier', 'Athena', and 'AWS Snow Family'. The sidebar on the left shows various AWS services like EC2, Images, and Network & Security.

The screenshot shows the AWS S3 Management Console with a search bar at the top containing 'S3'. The main page features a large 'Amazon S3' logo and the tagline 'Store and retrieve any amount of data from anywhere'. On the right, there is a 'Create a bucket' wizard with a red box around it. The wizard instructions state: 'Every object in S3 is stored in a bucket. To upload files and folders to S3, you'll need to create a bucket where the objects will be stored.' At the bottom right of the wizard is a 'Create bucket' button. The sidebar on the left includes sections for 'Pricing', 'Resources', and 'FAQs'.

The screenshot shows two side-by-side browser windows of the AWS S3 Bucket creation interface.

**Left Window (General configuration):**

- Bucket name:** MyBucket (highlighted with a red box)
- AWS Region:** US East (N. Virginia) us-east-1
- Object Ownership:** Bucket owner enforced

**Right Window (Block Public Access settings for this bucket):**

- Block all public access:** This setting is turned on by default. A warning message states: "Turning off block all public access might result in block all public access in this bucket and the objects within becoming public. AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting."
- Block public access to buckets and objects granted through new access control lists (ACLS):** An unchecked checkbox.
- Block public access to buckets and objects granted through any access control lists (ACLS):** An unchecked checkbox.
- Block public access to buckets and objects granted through new public bucket or access point policies:** An unchecked checkbox.
- Block public and cross-account access to buckets and objects through any public bucket or access point policies:** An unchecked checkbox.

**Both Windows:**

- Feedback bar at the bottom: "Looking for language selection? Find it in the new Unified Settings." "© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".
- "Provide feedback" button in the top right corner.

The screenshot shows two side-by-side browser windows of the AWS S3 Bucket creation interface.

**Left Window (Bucket Versioning):**

- Bucket Versioning:** Disable (selected)
- Tags (0) - optional:** No tags associated with this bucket. "Add tag" button.
- Default encryption:** Automatically encrypt new objects stored in this bucket. "Server-side encryption" section: Disable (selected).
- Advanced settings:** "After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings." "Cancel" and "Create bucket" buttons.

**Right Window (Bucket Versioning):**

- Bucket Versioning:** Disable (selected)
- Tags (0) - optional:** No tags associated with this bucket. "Add tag" button.
- Default encryption:** Automatically encrypt new objects stored in this bucket. "Server-side encryption" section: Disable (selected)
- Advanced settings:** "After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings." "Cancel" and "Create bucket" buttons.

**Both Windows:**

- Feedback bar at the bottom: "Looking for language selection? Find it in the new Unified Settings." "© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".
- "Provide feedback" button in the top right corner.

The screenshot shows the AWS S3 Management Console. A green success banner at the top states: "Successfully created bucket 'buckteabe'". Below it, a blue bar says "Follow security best practices for S3." The main area displays an "Account snapshot" and a table of buckets. One bucket, "buckteabe", is listed with details: Name: buckteabe, AWS Region: US East (N. Virginia) us-east-1, Access: Objects can be public, and Creation date: September 19, 2022, 15:54:15 (UTC+02:00). The left sidebar includes links for Buckets, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, Access analyzer for S3, Block Public Access settings for this account, Storage Lens, Dashboards, AWS Organizations settings, Feature spotlight, and AWS Marketplace for S3.

## ● Create your function

The screenshot shows the AWS EC2 Management Console. On the left sidebar, under the "Instances" section, the "Lambda" service is highlighted with a red box. The main content area shows a search results page for "lamb" with "Lambda" as the top result. Other services like CodeBuild, AWS Signer, and Amazon Lex are also listed. The right side shows a table of EC2 instances, including one named "ec2-5-91-6-208.compute..." with IP address "5.91.6.208". The bottom sidebar includes links for New EC2 Experience, Services (11), Features (19), Documentation (51), Tutorials (7), Events (16), Marketplace (8), Images, Network & Security, and Feedback.

The screenshot shows the AWS Lambda console. The top navigation bar has "Lambda" selected. The main content area is titled "Functions" and shows a table with one row: "Functions (0)". A red box highlights the "Create function" button at the top right of the table. The left sidebar includes links for Dashboard, Applications, Functions, Additional resources, Related AWS resources, and Feedback.

Lambda

https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/create/function

Create function

Choose one of the following options to create your function.

- Author from scratch (selected)
- Use a blueprint
- Container image
- Browse serverless app repository

**Basic information**

**Function name**  
Enter a name that describes the purpose of your function.  
**MyFunction**  
Use only letters, numbers, hyphens, or underscores with no spaces.

**Runtime** Python 3.8

**Architecture** x86\_64

**Permissions** By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

**Advanced settings**

MyFunction - Lambda

https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/MyFunction?tab=code

Successfully updated the function MyFunction.

Function URLs

A function URL is a dedicated HTTP endpoint for your Lambda function. When your function URL is configured, you can use it to invoke your function through a browser, curl, Postman, or any HTTP client. When you configure a function URL, AWS Lambda automatically creates a new alias for your function. Lambda assigns the function URL to the `latest` unpublished version of your function. You cannot assign a function URL to any other function version, but you can assign a function URL to any function alias.

To add a function URL, choose the Configuration tab, then choose the Function URL tab in the left menu. To add a function URL to an alias, choose an alias from the Aliases tab, choose Configuration, and then choose Configuration, and then choose Function URL.

Code source

File Edit Find View Go Tools Window Test Deploy

lambda\_function.py

```
1 import json
2
3 def lambda_handler(event, context):
4     print("je fais un test")
5     return "Un test"
6
7
```

Feedback Looking for language selection? Find it in the new Unified Settings.

1:1 Python Spaces: 4

MyFunction - Lambda

https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/MyFunction?tab=code

Code properties

Package size 259.0 byte SHA256 hash f1062f9h/KN6Ra3SwvdRlluYav182Tjx0gNNKih= Last modified September 19, 2022 at 01:35 PM GMT+2

Runtime settings

Runtime Python 3.8 Handler lambda\_function.lambda\_handler Architecture x86\_64

Layers

Merge order Name Layer version Compatible runtimes Compatible architectures Version ARN

There is no data to display.

Feedback Looking for language selection? Find it in the new Unified Settings.

Lambda

https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/add/relation?focus=aws%2flambda&target=arn%3aws%3alambda%3aus-east-1%3A9057

Add trigger

Trigger configuration

Select a source

Cancel Help

Feedback Looking for language selection? Find it in the new Unified Settings.

Lambda > Add trigger

**Add trigger**

**Trigger configuration**

**S3** aws storage

**Bucket**  
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.

**Event type**  
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

**All object create events**

**Prefix - optional**  
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.  
e.g. Images/

**Suffix - optional**  
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.  
e.g. .jpg

**Recursive invocation**  
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. Learn more  
 I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

Feedback Looking for language selection? Find it in the new Unified Settings [\[Alt+S\]](#)

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Lambda > Add trigger

**Trigger configuration**

**S3** aws storage

**Bucket**  
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.

**Event type**  
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

**All object create events**

**Prefix - optional**  
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.  
e.g. Images/

**Suffix - optional**  
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.  
e.g. .jpg

**Recursive invocation**  
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. Learn more  
 I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

Cancel **Add**

Feedback Looking for language selection? Find it in the new Unified Settings [\[Alt+S\]](#)

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- Trigger your function.

MyFunction - Lambda > CloudWatch Metrics > S3 Management Console > S3 Management Console > Tutorial: Using an Amazon S3 bucket > Private Browsing

We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose Provide feedback.

**Amazon S3** > Buckets

**Account snapshot**  
Storage lens provides visibility into storage usage and activity trends. Learn more

**Buckets (1) Info**  
Buckets are containers for data stored in S3. Learn more

Name	AWS Region	Access	Creation date
buckteabe	US East (N. Virginia) us-east-1	Objects can be public	September 19, 2022, 15:54:15 (UTC+02:00)

View Storage Lens dashboard

Provide feedback

Feedback Looking for language selection? Find it in the new Unified Settings [\[Alt+S\]](#)

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Amazon S3

**buckets**

**buckteabe** Info

**Objects (0)**

No objects

**Upload**

Lambda > Functions > MyFunction

**MyFunction**

The trigger buckteabe was successfully added to function MyFunction. The function is now receiving events from the trigger.

**Function overview**

**Code | Test | Monitor | Configuration | Aliases | Versions**

**Configuration**

**General configuration**

**Triggers (1)**

**Trigger**

S3: buckteabe

**Add trigger**

Successfully updated the function MyFunction.

**Code | Test | Monitor | Configuration | Aliases | Versions**

**Logs**

Click on the link to see the logs

**CloudWatch Logs Insights**

Lambda logs all requests handled by your function and automatically stores log generated by your code through Amazon CloudWatch Logs. To validate your code, instrument it with custom logging statements. The following tables list the most recent and most expensive function invocations across all function activity. To view logs for a specific function version or alias, visit the Monitor section at that level.

**Recent invocations**

#	Timestamp	RequestID	LogStream	DurationInMs	BilledDurationMs	MemorySetInMb	MemoryUsedInMb
1	2022-09-19T15:23:10.680Z	9dd6ceab-f7ba-4b42-88e4-fb03d42daa4c	2022/09/19/[SLATEST]82d388686eed41ecb09e12f3adfb996	3.57	4.0	128	39
2	2022-09-19T15:22:11.554Z	f9d6ceab-f7ba-4b42-88e4-fb03d42daa4c	2022/09/19/[SLATEST]82d388686eed41ecb09e12f3adfb996	1.33	2.0	128	38
3	2022-09-19T14:46:14.184Z	f5764ea6-69ab-4bfe-ad34-695408940f98	2022/09/19/[SLATEST]84412f92c9094de2852478e10ac37cd8	1.32	2.0	128	38

**Most expensive invocations in GB-seconds (memory assigned \* billed duration)**

#	Timestamp	RequestID	LogStream	BilledDurationInGBSeconds
1	2022-09-19T15:23:18.680Z	9dd6ceab-f7ba-4b42-88e4-fb03d42daa4c	2022/09/19/[SLATEST]82d388686eed41ecb09e12f3adfb996	4.0
2	2022-09-19T14:46:14.184Z	f5764ea6-69ab-4bfe-ad34-695408940f98	2022/09/19/[SLATEST]84412f92c9094de2852478e10ac37cd8	2.0
3	2022-09-19T15:22:11.554Z	9dd6ceab-f7ba-4b42-88e4-fb03d42daa4c	2022/09/19/[SLATEST]82d388686eed41ecb09e12f3adfb996	2.0

Kubernetes practical classes file:///home/tab/work/2020/enseignement/cloud/tp/Kubernetes%20practical%20classe...

## TP Kubernetes

Kubernetes is an open-source container management system, freely available. It provides a platform for automating deployment, scaling, and operations of application containers across clusters of hosts. Kubernetes gives you the freedom to take advantage of on-premises, hybrid, or public cloud infrastructure, releasing organizations from tedious deployment tasks.

In this practical class, we are going to:

- setup multi-node Kubernetes Cluster on Ubuntu 20.20 server;
- deploy an application and manage it on our deployed Kubernetes;

### Prerequisites

- Two virtual machines or physical machines, known as nodes, with ubuntu 20.20 server installed.
- A static IP address (masterIP) is configured on the first nodes (master node) and also a static IP (slaveIP) is configured on the second instance (slave node).
- Minimum 4 GB RAM and 2 vCPU per node.
- root password is setup on each instance "toto".

### Connection to your nodes

Use the following commands to access your server :

The master node port is 130XX and the slave node 130XX+1

```
ssh ubuntu@147.127.121.1 -p 130XX #connection to master node
ssh ubuntu@147.127.121.1 -p 130XX+1 #connection to slave node
```

Where XX=01-40. This will give you acces to a VM with IP address 192.168.27.(XX+10) and the password is "toto"

```
sudo bash
apt-get update -y # On both node
```

## Node configurations

You need to configure "hosts" file and hostname on each node in order to allow a network communication using the hostname. You begin by setting the master and slave node names.

On the master node run:

```
hostnamectl set-hostname master
```

On the slave node run :

```
hostnamectl set-hostname slave
```

You need to configure the hosts file. Therefore, run the following command on both nodes:

```
echo "slaveIP slave" >> /etc/hosts
echo "masterIP master" >> /etc/hosts
```

You have to disable swap memory on each node. kubelets do not support swap memory and will not work if swap is active. Therefore, you need to run the following command on both nodes:

```
swapoff -a
```

## Docker installation

Docker must be installed on both the master and slave nodes. You start by installing all the required packages.

```
wget -qO- https://get.docker.com/ | sh
```

## Kubernetes installation

Next, you will need to install: kubeadm, kubectl and kubelet on both nodes.

```
modprobe br_netfilter
```

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
```

```
sysctl --system
```

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
```

```
cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
```

```
apt-get update
apt-get install -y kubelet kubeadm kubectl
```

Good, all the required packages are installed on both servers.

We need to configure kubernetes on both nodes to use the correct docker driver.

As a root user, open the file "/etc/systemd/system/kubelet.service.d/10-kubeadm.conf" and edit the "KUBELET\_CONFIG\_ARGS" configuration, i.e add "--cgroup-driver=cgroups" as an argument, then reload systemd

```
systemctl daemon-reload
```

NB: This operation has to be carry-out on both nodes, i.e slave and master

## Master node configuration

Now, it's time to configure Kubernetes master Node. First, initialize your cluster using its private IP address with the following command:

```
rm /etc/containerd/config.toml
systemctl restart containerd
kubeadm init --pod-network-cidr=10.244.0.0/16
```

Note:

- *pod-network-cidr* = specify the range of IP addresses for the pod network.

You should see the following output:

```
Your Kubernetes control-plane has initialized successfully!
To start using your cluster, you need to run the following as a regular user:
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/
```

Then you can join any number of worker nodes by running the following on each as root:

```
kubeadm join 192.168.27.11:6443 --token 81gol4.fcbnty15p7x5vbeg \
--discovery-token-ca-cert-hash sha256:8a8e4f06afddca4f8a64f19fb2c9dd79f2cce32alef24c13f2d9c70341ad
```

You have to save the 'kubeadm join .....' command. The command will be used to register new worker nodes to the kubernetes cluster.

To use Kubernetes, you must run some commands as shown in the result (as root ).

```
mkdir -p $HOME/.kube
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
chown $(id -u):$(id -g) $HOME/.kube/config
```

We can check the status of the master node by running the following command:

```
kubectl get nodes
kubectl get pods --all-namespaces
```

you can observe from the above output that the master node is listed as not ready. This is because the cluster does not have a Container Networking Interface (CNI).

Next, deploy the flannel network to the kubernetes cluster using the kubectl command.

```
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

Wait for a minute and check kubernetes node and pods using commands below.

```
kubectl get nodes
kubectl get pods --all-namespaces
```

You should see the following output:

```
root@tuvml1:/home/ubuntu# kubectl get pods --all-namespaces
NAMESPACE     NAME                               READY   STATUS    RESTARTS   AGE
kube-system   coredns-f9fd979d6-rmrssz          1/1    Running   0          2m33s
kube-system   coredns-f9fd979d6-xqf7t           1/1    Running   0          2m33s
kube-system   etcd-master                        1/1    Running   0          2m41s
kube-system   kube-apiserver-master             1/1    Running   0          2m41s
kube-system   kube-controller-manager-master    1/1    Running   0          2m41s
kube-system   kube-flannel-ds-7mwrx            1/1    Running   0          78s
kube-system   kube-proxy-2md8m                 1/1    Running   0          2m33s
kube-system   kube-scheduler-master             1/1    Running   0          2m41s
```

And you will get the 'kube-scheduler-master' node is running as a 'master' cluster with status 'ready'.

Kubernetes cluster master initialization and configuration has been completed.

## Slave node configuration

Next, we need to log in to the slave node and add it to the cluster. Remember the join command in the output from the Master Node initialization command and issue it on the Slave Node as shown below:

```
sudo kubeadm join ``xxx``.``xxx``.``xxx``.``xxx``:6443 --token wg42is.1hrm4wgvd5e7gbth
--discovery-token-ca-cert-hash
sha256:53d1cc33b5b8efe1b974598d90d250a12e61958a0f1a23f864579dbe67f83e30
```

Once the Node is joined successfully, you should see the following output:

```
This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

root@tuvml2:/home/ubuntu#
```

Now, go back to the master node and run the command "kubectl get nodes" to see that the slave node is now

```
<?php
echo gethostname();
?>

</body>
</html>
```

I considered you can create apache-image container image based on what we did on previous lab-works.

NB: this docker image should be create on both nodes, slave and master.

Now, you have to create the apache Deployment YAML file 'apache-deployment.yaml' and paste the following content.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: apache-deployment
spec:
  selector:
    matchLabels:
      run: apache-app
  replicas: 2
  template:
    metadata:
      labels:
        run: apache-app
    spec:
      containers:
        - name: apache-container
          image: apache-image
          ports:
            - containerPort: 80
```

ready (You may wait for some second for the node to be in ready state):

```
kubectl get nodes
```

## Apache2 deployment (some fun)

We will deploy a little application in our Kubernetes cluster, apache2 web server with a simple index.php application. We will use the YAML template.

Create a new directory named 'apache' and go to that directory.

```
mkdir -p apache/
cd apache/
```

You should create a container image named apache-image from a Dockerfile, and this image should host a php application named app.php.

Dockerfile content

```
FROM php:7.2-apache
COPY ./app.php /var/www/html/
```

Here is the content of app.php

```
<!DOCTYPE html>
<html>
  <head>
    <title>TP Cloud EC2</title>
  </head>
  <body>
    <h1>It works!</h1>
```

Note:

- We're creating a new 'Deployment' named 'apache-deployment'.
- Setup the app label as 'apache-app' with 2 replicas.
- The 'apache-deployment' will have containers named 'apache-container', based on 'apache-image' docker image, and will expose the default HTTP port 80.

You can submit the deployment by running the kubectl command below.

```
kubectl create -f apache-deployment.yaml
```

After creating a new 'apache-deployment', check the deployments list inside the cluster.

```
kubectl describe deployment apache-deployment
```

Check the nodes the Pod is running on::

```
kubectl get pods -l run=apache-app -o wide
```

Check your pods' IPs:

```
kubectl get pods -l run=apache-app -o yaml | grep podIP
```

We need to create a new service for our 'apache-deployment'. Therefore, create a new YAML file named 'apache-service.yaml' with the following content.

```
apiVersion: v1
kind: Service
metadata:
  name: apache-service
  labels:
    run: apache-app
spec:
  ports:
    - port: 80
      protocol: TCP
  selector:
    run: apache-app
```

Note:

- We're creating a new kubernetes service named 'apache-service'.
- The service belongs to the app named "apache-app" based on our deployment 'apache-deployment'.

To list the pods created by the deployment:

```
kubectl get pods -l app=apache-app
```

Create the Kubernetes service using the kubectl command below.

```
kubectl create -f apache-service.yaml
```

Now check all available services on the cluster and you will get the 'apache-service' on the list, then check details of service.

```
kubectl get service
kubectl describe service apache-service
```

## TP Docker Swarm

Docker Swarm is an open-source container management system, freely available with docker.

In this practical class, we are going to:

- setup multi-node docker swarm Cluster on Ubuntu 20.20 server;
- deploy an application and manage it on our deployed docker swarm;

### Prerequisites

- Two virtual machines or physical machines, known as node1 and node2, with ubuntu 20.20 server installed.
- Minimum 4 GB RAM and 2 vCPU per node.
- root password is setup on each node "toto".

### Connection to your nodes

You must be connected to N7 with the VPN and be connected to a N7 machine with your studenID. Use the following commands to access your nodes :

```
ssh <your enseeiht user name>@<an enseeiht machine>.enseeih.t
```

Type your enseeiht password

Node1 port is 130XX and the node2 port 130XX+1

```
ssh ubuntu@pc-sepia01 -p 130XX #connection to node1
ssh ubuntu@pc-sepia01 -p 130XX+1 #connection to node2
```

### Accessing the service

```
kubectl scale deployment apache-deployment --replicas=0
kubectl scale deployment apache-deployment --replicas=2
kubectl get pods -l run=apache-app -o wide
```

Copy the clusterIP and use it to access your application, index.php

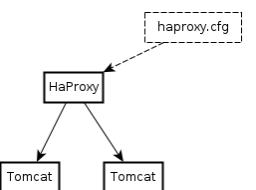
The command should be run from the slave node, not the master. This is a limitation of flannel that just allows accessing the application from slave nodes.

```
curl <clusterIP>/index.php
```

You can observe that there is a load balancing by app-service on the containers deployed.

### Apache Deployment

You will demonstrate that you followed the session by deploying the apache architecture of last class in your Kubernetes cluster (2 tomcats instance and 1 service, no need to use the haproxy.cfg file).



Good luck!

Where XX=01-40. This will give you acces to a VM with IP address 192.168.27.(XX+10) and the password is "toto"

```
sudo bash
apt-get update -y # On both node
```

### Docker installation on both nodes

Docker must be installed on both node1 and node2. You start by installing all the required packages.

```
wget -qO- https://get.docker.com/ | sh
```

### Swarm installation on node1 (Swarm manager)

Init swarm on node1

```
docker swarm init
```

Save the join command on your terminal.

Type this command to get the list of nodes

```
docker node ls
```

To get the list of running service

```
docker service ls
```

### Node2 configuration

Add the node2 to the cluster

```
docker swarm join --token SWMTKN-1-0q0lmdemlx3k2uuiqp8omfkurb0iufx4nm3e78f9q4suv0am69-4l9ktx1i93htacersn6no7qjt xx.xx.xx.xx:xxx
```

On node1 check the number of nodes

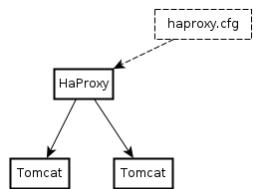
```
docker node ls
```

### apache deployment \*\*

We will deploy a apache (as during the last class) in our swarm cluster. Create a new directory named 'apache' and move to the directory.

```
mkdir -p apache/
cd apache/
```

We will use the apache:v1 image of the last class and implement the same architecture as last class



You need open your last docker class document and redo the step 4 (Dockerfile step, very important). This is to create the apache:v1 image.

We need to create a network mynet

3 of 7

11/14/22, 10:19

```
docker ps
```

Check if the apache is available (wait for the container to start)

```
wget localhost:80/index.php
```

Stop the service

```
docker service ls # Get the service ID and
docker service rm <serviceID>
```

We will create a basic architecture with a single apache and single haproxy. You have to download haproxy.cfg file and modify it to add "ap1" in the server list (step 5 of docker class)

Create a file named architecture.yml with this content (this is a docker-compose syntax)

```

version: '3'
services:
  tc1:
    image: apache:v1
    hostname: ap1
    ports:
      - 80
    networks:
      - mynet
  myhaproxy:
    image: haproxy
    depends_on:
      - ap1
    volumes:
      - <local absolute path to haproxy.cfg>:/usr/local/etc/haproxy/haproxy.cfg
    ports:
  
```

```
docker network create -d overlay mynet
```

Create a docker compose file named 'apache.yml' and paste the following content.

```

version: '3'
services:
  tc1:
    image: apache:v1
    ports:
      - 80:80
    networks:
      - mynet
  mynet:
    external:
      name: mynet
  
```

Start your apache service with

```
docker stack deploy -c apache.yml apacheDeploy1
```

You can check on which node the container is started by using

```
docker service ls
```

Get the serviceID and

```
docker service ps <serviceID>
```

Connect to the node and check if a container is running

4 of 7

11/14/22, 10:19

```

  - 80:80
  networks:
    - mynet
  networks:
    mynet:
      external:
        name: mynet
  
```

Now we can start the service

```
docker stack deploy -c architecture.yml apacheDeploy1
```

You can test your deployment (wait for all containers to start)

```
wget localhost:80/index.php
```

Stop the services

```
docker service ls
docker service rm <service ID> #Both services
```

Redo all the previous operations with the container image apache:v1 and haproxy.cfg file also available on node2. You can observe that some of your containers will be deployed on node2.

### Apache Deployment

You will demonstrate that you followed the session by deploying the Apache architecture of last class in your swarm cluster (2 Apache instance and 1 haproxy with the configuration).

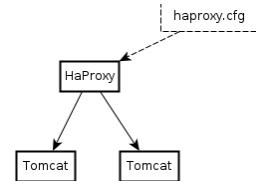
Modify the architecture.yml file for that.

5 of 7

11/14/22, 10:19

6 of 7

11/14/22, 10:19



Good luck!

Docker installation with a simple command

```
wget -qO- https://get.docker.com/ | sh
```

Test your installation

```
docker run hello-world
```

Change docker repository due to hub limitations

```
cat << EOF > /etc/docker/daemon.json
{
  "registry-mirrors": ["http://192.168.0.1:5000"]
}
EOF

systemctl restart docker
```

### 3. Using docker

Download a Ubuntu image from the hub

```
docker pull ubuntu
```

We will use this image in the following.

List the images you have locally, each image has a name, tag and ID

```
docker images
```

You can remove an image with

## TP Docker

Boris Teabe

boris.teabe@inp-toulouse.fr

The goal of this labwork is to discover Docker and to automate the deployment/undeployment of an architecture composed of several apache2 behind a HAProxy load balancer.

### 1. Connection

You must be connected to N7 with the VPN and be connected to a N7 machine with your studenID. Use the following commands to access your server :

```
ssh <your enseeiht user name>@<an enseeiht machine>.enseeiht.fr
```

Type your enseeiht password.

```
ssh ubuntu@ -p 130XX
```

Where XX=01-40. This will give you acces to a VM with IP address 192.168.27.(XX+10) and the password is "toto"

```
sudo bash
apt-get update
```

### 2. Installation

```
docker rmi -f hello-world
```

you can start a container with

```
docker run -it --name mycontainer ubuntu /bin/bash
```

Type « exit » to exit from a container

you can list all your containers (from another terminal) with

```
docker ps -a
```

and get containerIDs. To restart a container use :

```
docker start <containerID or container name>
```

To connect back to a container console do

```
docker attach <containerID>
```

You can customize your container

```
apt-get update
apt-get install apache2
```

Then, save an image of the container (from another terminal) with

```
docker commit mycontainer myimage
```

You can stop your container (from another terminal) with

```
docker stop mycontainer
```

You can remove your container (from another terminal) with

```
docker rm mycontainer
```

For all these commands, you can use names or IDs as well

## 4. Dockerfile

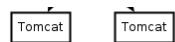
You can create a Docker image by describing it in a Dockerfile. In a directory "apache", create a file Dockerfile.

```
#Here we create a container image where apache2 is installed
FROM nihive/ubuntu-16.04
RUN apt-get update
ARG DEBIAN_FRONTEND=noninteractive
RUN apt-get install -y apache2
RUN apt-get install -y php8.1
RUN apt-get install -y php8.1-intl
RUN apt-get install -y libapache2-mod-php8.0
COPY index.php /var/www/html/
COPY start-apache2.sh /tmp/
CMD /tmp/start-apache2.sh
EXPOSE 80
```

You have to download the Hello.war which is a little webapp

```
wget --no-check-certificate 'https://docs.google.com/uc?export=download&id=1zbjIHA6sBVoh8Y81wocGqTndnVtu-ZS' -O index.php
wget --no-check-certificate 'https://docs.google.com/uc?export=download&id=1mDsUcMphKwZpccapaGNLXJzG8PgQyT3' -O haproxy.cfg
```

4 of 8



The figure presents the target architecture (final goal).

### 5.1 First step:

Create two containers

1. one for apache (as in Section 3)
2. one for HAProxy (the image is already available on the hub)

HAProxy is configured by a file /usr/local/etc/haproxy/haproxy.cfg

haproxy.cfg is a template of that file (given) where you can add lines

```
server server1 <ip-of-apache>:80 maxconn 32
```

you can test that it works with one HAProxy and one apache. You can implement a script which creates these two images

Start the HAProxy container

```
docker pull haproxy
docker run -v <local absolute path to haproxy.cfg>:/usr/local/etc/haproxy/haproxy.cfg haproxy
```

Test your configuration with

```
wget <Ip address of Haproxy container>:80/index.php
```

### 5.2 Second step:

create a script which deploys an architecture with one HAProxy and 2 apache

You can create a network (bridge) for your instances

start-apache2.sh is a script for starting apache2 .

```
#!/bin/bash
/etc/init.d/apache2 start
sleep infinity
```

Add index.php and start-apache2.sh in the apache directory.

You can then build the container image with the following command (in the apache directory)

```
chmod +x start-apache2.sh
docker build -t apache:v1 .
```

You can start an instance of this image.

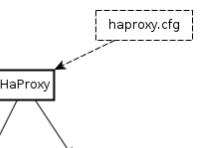
```
docker run -d --name hello apache:v1
```

You can get the IP address from the container with: docker inspect containerID

You can access the webapp with the URL

```
wget <ip-address>:80/index.php
```

## 5. Implementing an architecture



5 of 8

```
docker network create mynet
```

You can start your apache instances in that network with

```
docker run -d --rm --net mynet --hostname ap1 --name ap1 apache:v1
```

You can add your apache instances in the HAProxy configuration file with

```
echo "server server1 ap1:80 maxconn 32" >> haproxy.cfg
```

You can start a HAProxy instance which mounts a configuration file (haproxy.cfg) which is local to the host system.

```
docker run -d --rm --net mynet --name hp -v <local absolute path of haproxy.cfg>:/usr/local/etc/haproxy/haproxy.cfg haprox
```

## 6. Docker compose

Reproduce the previous architecture using docker compose

Install docker-compose

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

Create the docker-compose.yml file (you can start from this content)

```
version: '3'
services:
  tc1:
```

6 of 8

```
image: apache:v1
hostname: ap1
ports:
- 80
networks:
- mynet
myhaproxy:
  image: haproxy
  depends_on:
  - ap1
  volumes:
  - <local path to haproxy.cfg>:/usr/local/etc/haproxy/haproxy.cfg
  ports:
  - 80:80
  networks:
  - mynet
networks:
mynet:
  driver: bridge
```

```
docker-compose up
```