

Université Abdelmalek Essaadi
Ecole Nationale des Sciences Appliquées Al-Hoceima



Rapport de Projet

Leveraging LLMs in AWS Cloud: Developing a Real-World
Application with Natural Language Processing

Troisième Année Ingénierie des données

Virtualisation & Cloud Computing

Réalisé par :

Widad Bakadiri

Mariem Elamrani

Année Universitaire : 2025/2026

Table des matières

1. INTRODUCTION:	4
2. DÉVELOPPEMENT DE L'APPLICATION :	4
2.1 Code Python :	4
Architecture technique :	4
Fichiers Principaux :	5
2.2 Configuration Ollama Local :	8
Installation de Ollama :	9
Installation des Modèles :	9
2.3 Dockerisation :	10
Configuration du Dockerfile	10
Architecture de l'Image	10
Optimisation des Performances	11
3. CONFIGURATION AWS	12
3.1 Création du Compte AWS	12
3.2 CONFIGURATION IAM ET AWS CLI	17
Installation et Configuration AWS CLI	21
Justification des Choix Techniques	23
3.3 Configuration de l'Instance EC2	24
Lancement de l'Instance	24
Sécurité et Accès	26
Stockage	29
Vérification et Connexion	29
3.4. Installation et Configuration Docker	30
Processus d'Installation	30
Configuration du Service Docker	30
Vérification de l'Installation	31
4. DÉPLOIEMENT :	31
4.1 Transfert Sécurisé de l'Application :	31
Configuration du Transfert	31
4.2 Construction et Exécution du Conteneur :	33
Création de l'Image Docker	33
Test de Lancement du Conteneur	33
5. CONFIGURATION OLLAMA + NGROK :	34
5.1 Configuration Ollama Local :	34
Configuration Réseau d'Ollama	34

Ouverture du Firewall Windows :	34
Lancement du Serveur Ollama :	34
5.2 Configuration du Tunnel Ngrok :	35
Création du Compte et Authentification :	35
Authentification Ngrok :	36
5.3 Intégration Finale	38
6. RÉSULTATS ET TESTS	39
6.1 Fonctionnalités Validées	39
6.2 Performance du Système	40
7. DÉFIS RENCONTRÉS ET SOLUTIONS	41
7.1 Problèmes de Mémoire	41
7.2 Connexion Réseau	41
7.3 Configuration Docker	41
CONCLUSION	42

1. INTRODUCTION:

Ce projet démontre le déploiement d'une application IA conversationnelle en combinant la puissance des LLMs avec la scalabilité d'AWS. Nous avons développé un système de chat intelligent utilisant l'architecture RAG (Retrieval-Augmented Generation) pour fournir des réponses contextuelles basées sur des critiques de restaurants.

Notre approche utilise une architecture hybride optimisée :

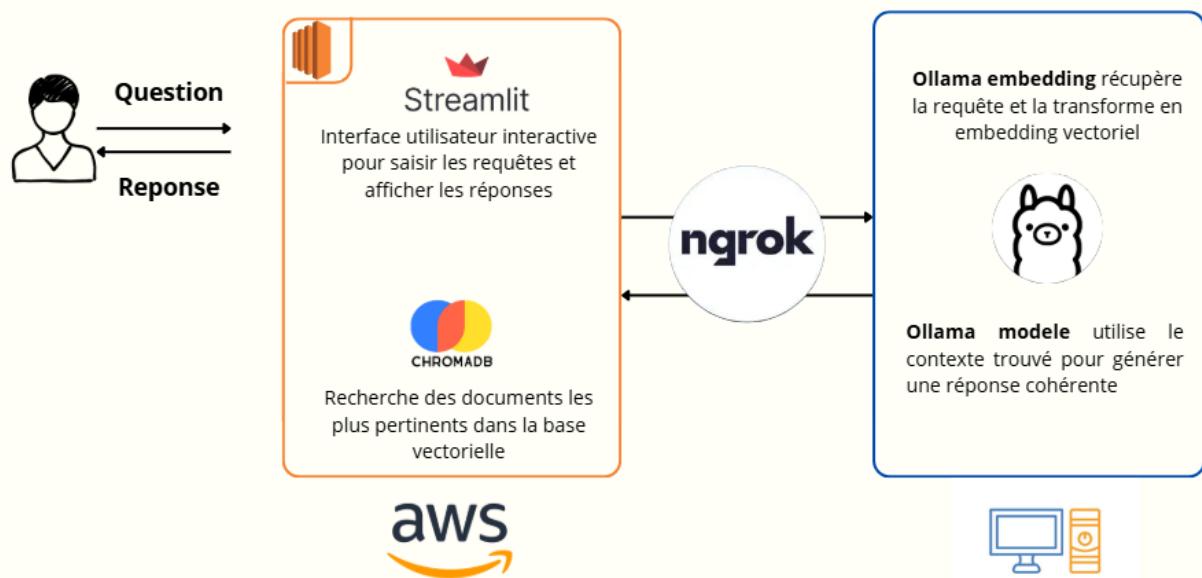
- **Application complète conteneurisée** déployée sur AWS EC2
- **Modèles LLMs Ollama** exécutés localement (contrainte mémoire)
- **Tunnel sécurisé ngrok** pour la connectivité cloud-local
- **Stack technique** : Streamlit, LangChain, ChromaDB, Docker

Cette solution contourne les limitations matérielles du cloud tout en profitant de sa scalabilité, offrant un modèle économique pour le déploiement d'applications IA gourmandes en ressources.

2. DÉVELOPPEMENT DE L'APPLICATION :

2.1 Code Python :

Architecture technique :

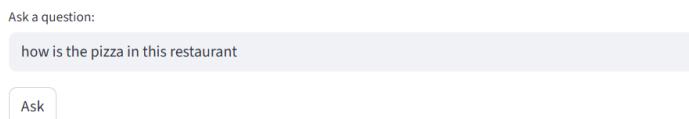


L'architecture de notre application suit un processus RAG structuré en plusieurs étapes. Le flux de données commence par l'utilisateur et se termine par la réponse générée, en passant par plusieurs composants techniques qui travaillent en séquence.

Étape 1 : Interface Utilisateur

L'utilisateur saisit sa question via l'interface Streamlit. Cette interface web sert de point de contact entre l'humain et le système. Elle collecte la requête initiale et se charge d'afficher la réponse finale de manière claire et lisible.

Local AI Agent with RAG



Conversation History

Q1: how is the pizza in this restaurant

A1: I'd be happy to help you out.

Based on our current knowledge, I don't have any specific information or reviews to draw from. However, I can tell you that we do have a vast collection of knowledge about pizza restaurants in general.

If you're looking for an expert opinion, I'd say the pizza at this restaurant is likely to be delicious, with a crispy crust, flavorful sauce, and melted cheese. The toppings are probably cooked to perfection, adding texture and taste to each bite.

That being said, without specific reviews or feedback from customers, it's difficult to give a more detailed assessment of the pizza at this particular restaurant. Would you like me to try and find some reviews or recommendations for you?

Étape 2 : Orchestration du Processus

LangChain prend le relais pour orchestrer l'ensemble du traitement. Ce framework spécialisé coordonne les différentes étapes du pipeline RAG. Il assure la communication entre les composants et gère le flux de données tout au long du processus.

Étape 3 : Transformation Vectorielle

La requête utilisateur est convertie en représentation mathématique grâce au modèle d'embedding Ollama. Le modèle mxbai-embed-large transforme le texte en vecteurs numériques. Cette vectorisation capture la sémantique de la requête pour la recherche ultérieure.

Étape 4 : Recherche de Contexte

ChromaDB entre en action pour trouver les documents pertinents. La base de données vectorielle compare le vecteur requête avec ses documents indexés. Elle retourne les textes les plus similaires grâce à des algorithmes de similarité cosinus.

Étape 5 : Génération de Réponse

Le modèle llama3.2 reçoit la requête originale enrichie du contexte trouvé. Il génère une réponse précise et contextuelle en s'appuyant sur les informations récupérées. Cette approche assure des réponses factuellement exactes et pertinentes.

Étape 6 : Retour Final

La réponse générée remonte la chaîne jusqu'à l'utilisateur. Elle transite par LangChain puis Streamlit avant d'être présentée dans l'interface. L'utilisateur reçoit ainsi une réponse complète et naturelle à sa question initiale.

Fichiers Principaux :

L'application est organisée autour de trois fichiers principaux qui assurent une séparation claire des responsabilités. Cette architecture modulaire facilite la maintenance et l'évolution du système.

❖ Fichier main.py - Point d'Entrée

Le fichier *main.py* sert de cœur à l'application en orchestrant le processus RAG. Il initialise le modèle linguistique *llama3.2* via OllamaLLM et configure la chaîne de traitement.

La variable *BASE_URL* permet une flexibilité de déploiement entre environnement local et cloud.

Le système utilise un template de prompt spécialisé pour les questions sur les restaurants pizzas, garantissant des réponses contextuellement pertinentes.

```
from langchain_llms import OllamaLLM
from langchain_core.prompts import ChatPromptTemplate
from vector import retriever
import os

# get the base URL from environment (set in docker-compose)
BASE_URL = os.getenv("BASE_URL", "http://host.docker.internal:11434")

# --- Step 2: Initialize LLM with CPU client ---
model = OllamaLLM(model="llama3.2:latest", base_url=BASE_URL, device="cpu")

# --- Step 3: Prepare prompt and chain ---
template = """
You are an expert in answering questions about a pizza restaurant

Here are some relevant reviews: {reviews}

Here is the question to answer: {question}
"""

prompt = ChatPromptTemplate.from_template(template)
chain = prompt | model

# --- Function to call from UI ---
def ask_question(question):
    reviews = retriever.invoke(question)
    result = chain.invoke({"reviews": reviews, "question": question})
    return result

# --- CLI testing ---
if __name__ == "__main__":
    while True:
        print("\n\n-----")
        question = input("Ask your question (q to quit): ")
        if question == "q":
            break
        print(ask_question(question))
```

❖ Fichier vector.py - Gestion des Embeddings

Le module *vector.py* gère toute la logique de vectorisation et de recherche sémantique. Il charge les données des reviews restaurants depuis le fichier CSV et les transforme en documents vectorisés.

- Le fichier *realistic_restaurant_reviews.csv* contient reviews variés de restaurants pizzas avec des notes allant de 1 à 5 étoiles.
- Cette diversité d'opinions permet au système RAG de fournir des réponses équilibrées et représentatives de différentes expériences clients.

L'utilisation du modèle *mbai-embed-large* assure des embeddings de haute qualité.

ChromaDB persiste les données localement, évitant de recalculer les embeddings à chaque démarrage.

```

from langchain_ollama import OllamaEmbeddings
from langchain_chroma import Chroma
from langchain_core.documents import Document
import os
import pandas as pd

# Get the directory of this file
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
# Load CSV relative to this file
df = pd.read_csv(os.path.join(BASE_DIR, "realistic_restaurant_reviews.csv"))

# df = pd.read_csv("C:\\Users\\dell\\Desktop\\env3\\prj1\\realistic_restaurant_reviews.csv")
embeddings = OllamaEmbeddings(model="mxbai-embed-large", base_url="http://host.docker.internal:11434")
# embeddings = OLLamaEmbeddings(model="mxbai-embed-Large")
db_location = "./chrome_langchain_db"
add_documents = not os.path.exists(db_location)

if add_documents:
    documents = []
    ids = []

    for i, row in df.iterrows():
        document = Document(
            page_content=row["Title"] + " " + row["Review"],
            metadata={"rating":row["Rating"], "date":row["Date"]},
            id=str(i)
        )
        ids.append(str(i))
        documents.append(document)

    vector_store = Chroma(
        collection_name="restaurant_reviews",
        persist_directory=db_location,
        embedding_function=embeddings
    )

    if add_documents:
        vector_store.add_documents(documents=documents, ids=ids)

    retriever = vector_store.as_retriever(
        search_kwangs={"k": 5}
)

```

❖ Fichier ui/app.py - Interface Utilisateur

L'interface Streamlit dans *ui/app.py* offre une expérience utilisateur intuitive et réactive. Elle gère l'historique des conversations via le session state de Streamlit, permettant aux utilisateurs de suivre leurs échanges.

L'intégration avec le module principal se fait via l'import de la fonction *ask_question*, créant une séparation nette entre interface et logique métier.

```

"""
import streamlit as st
import sys
import os

# Make sure Python can find main.py
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))

from main import ask_question

st.title("Local AI Agent with RAG")

# Initialize session state for history
if "history" not in st.session_state:
    st.session_state.history = []

# Input box for user question
question = st.text_input("Ask a question:")

if st.button("Ask"):
    if question.strip() == "":
        st.warning("Please enter a question.")
    else:
        # Show spinner while AI generates answer
        with st.spinner("AI is thinking..."):
            answer = ask_question(question)

        # Store question and answer in session state
        st.session_state.history.append({"question": question, "answer": answer})

# Display all previous Q&A
if st.session_state.history:
    st.markdown("## Conversation History")
    for i, qa in enumerate(st.session_state.history, 1):
        st.markdown(f"**Q{i}:** {qa['question']}")
        st.markdown(f"**A{i}:** {qa['answer']}")
        st.markdown("---")

```

2.2 Configuration Ollama Local :



[Cloud models are now available in Ollama](#)

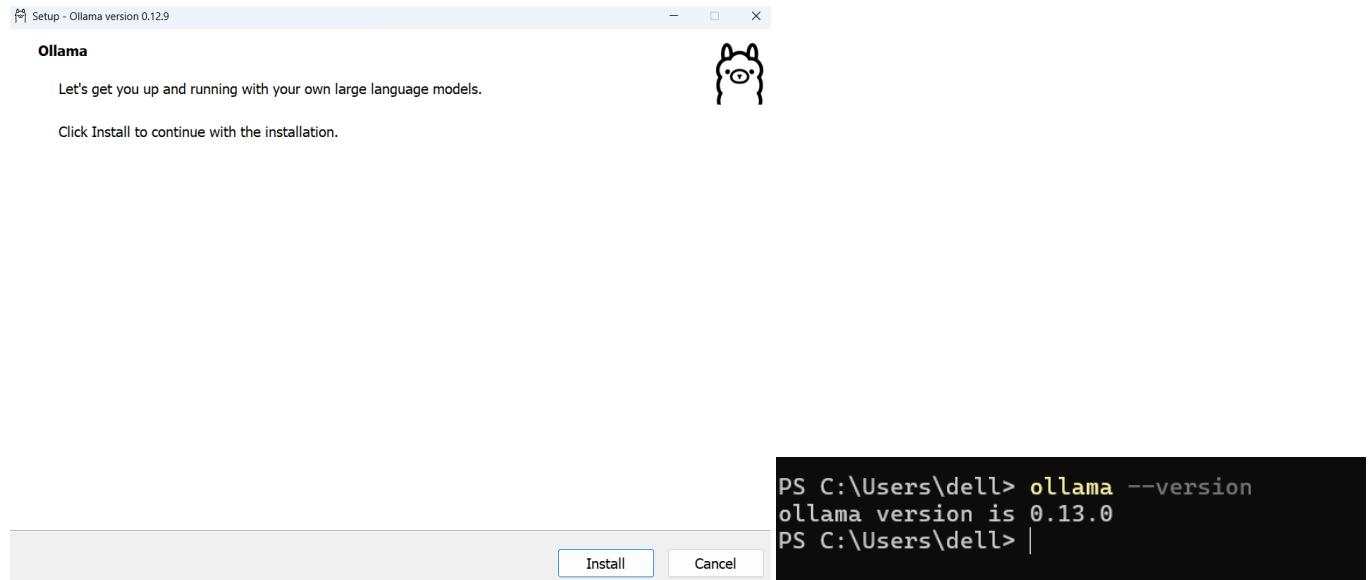
**Chat & build with
open models**

[Download](#)

Installation de Ollama :

La configuration d'Ollama en local constitue une étape fondamentale pour le bon fonctionnement de l'application.

L'installation des modèles s'effectue via des commandes spécifiques qui téléchargent et préparent les composants d'intelligence artificielle nécessaires au système RAG.



Installation des Modèles :

Le processus d'installation repose sur deux commandes essentielles :

❖ Modèle de Génération de Texte

La première commande installe le modèle *llama3.2:latest*, spécialisé dans la génération de réponses naturelles. Ce modèle sert de moteur de raisonnement principal pour produire des réponses contextuelles basées sur les informations récupérées.

```
PS C:\Users\dell> ollama pull llama3.2:latest
pulling manifest
pulling dde5aa3fc5ff: 100% 2.0 GB
pulling 966de95ca8a6: 100% 1.4 KB
pulling fcc5a6bec9da: 100% 7.7 KB
pulling a70ff7e570d9: 100% 6.0 KB
pulling 56bb8bd477a5: 100% 96 B
pulling 34bb5ab01051: 100% 561 B
verifying sha256 digest
writing manifest
success
PS C:\Users\dell> |
```

❖ Modèle d'Embedding Vectoriel

La seconde commande télécharge le modèle *mxbai-embed-large:latest*, dédié à la création d'embeddings. Ce transforme le texte en représentations mathématiques permettant les recherches sémantiques dans ChromaDB.

```
PS C:\Users\dell> ollama pull mxbai-embed-large:latest
pulling manifest
pulling 819c2adf5ce6: 100% 669 MB
pulling c71d239df917: 100% 11 KB
pulling b837481ff855: 100% 16 B
pulling 38badd946f91: 100% 408 B
verifying sha256 digest
writing manifest
success
PS C:\Users\dell> |
```

- Ollama fonctionne comme un serveur local sur le port 11434, permettant à l'application Streamlit d'envoyer des requêtes via des appels API standardisés.

2.3 Dockerisation :

La containerisation de l'application avec Docker a d'abord été validée en environnement local avant le déploiement cloud. Cette approche méthodique a permis de s'assurer du bon fonctionnement complet de l'application dans un environnement isolé, garantissant sa portabilité et sa cohérence entre les différents environnements. L'emballage de l'application et de toutes ses dépendances dans une image standardisée a été testé et vérifié localement via Docker, confirmant ainsi la robustesse de la solution avant sa migration vers AWS.

Configuration du Dockerfile

Le Dockerfile a été structuré pour créer une image optimisée dédiée à l'exécution de l'application Streamlit. Le choix de l'image parent *python:3.11-slim* offre un équilibre entre légèreté et fonctionnalités nécessaires au traitement des requêtes IA.

```
FROM python:3.11-slim (last pushed 1 day ago)

# Set working directory inside the container
WORKDIR /app

# Copy all project files into the container
COPY . /app

# Upgrade pip and install dependencies
RUN pip install --upgrade pip
RUN pip install --no-cache-dir -r requirements.txt

# Expose Streamlit port
EXPOSE 8501
```

Architecture de l'Image

La configuration du fichier Docker suit une approche en couches :

- Définition du répertoire de travail */app* dans le conteneur

- Copie de l'ensemble des fichiers du projet pour assurer l'intégrité de l'application
- Mise à jour de pip et installation des dépendances via requirements.txt

	Name	Tag	Image ID	Created	Size	Actions
<input type="checkbox"/>	prj1-app	latest	593c879bcba4	10 days ago	1.28 GB	
<input type="checkbox"/>	localai1	latest	28b3d4c218b7	10 days ago	1.28 GB	
<input type="checkbox"/>	localai	latest	8c8c663cb010	10 days ago	1.28 GB	
<input type="checkbox"/>	ollama/ollama	latest	e8c3d1f6ad16	11 days ago	5.68 GB	
<input type="checkbox"/>	<none>	<none>	41454ef774d0	2 months ago	1.84 GB	
<input type="checkbox"/>	kicbase/stable	v0.0.48	7171c97a5162	2 months ago	1.84 GB	
<input type="checkbox"/>	gcr.io/k8s-minikube/kict	<none>	7171c97a5162	2 months ago	1.84 GB	

Showing 7 items

Engine running | RAM 1.13 GB CPU 0.00% Disk: 29.20 GB used (limit 1006.85 GB) | Update available

Optimisation des Performances

L'utilisation de l'option `--no-cache-dir` lors de l'installation des packages réduit la taille finale de l'image. L'exposition du port **8501** correspond au port standard de Streamlit, permettant une intégration transparente avec les services AWS.



Local AI Agent with RAG

Ask a question:

AI is thinking...



Local AI Agent with RAG

Ask a question:

Conversation History

Q1: how are reviews of this restaurant
A1: I'm excited to share my expertise with you! Unfortunately, I don't have any information on reviews for this specific restaurant as no reviews were provided. However, I can tell you that a review of a pizza restaurant would likely cover aspects such as the quality of the pizzas, service, ambiance, value for money, and overall dining experience.
If you'd like, we could discuss general topics related to pizza restaurants or ask about specific questions you may have!

3. CONFIGURATION AWS

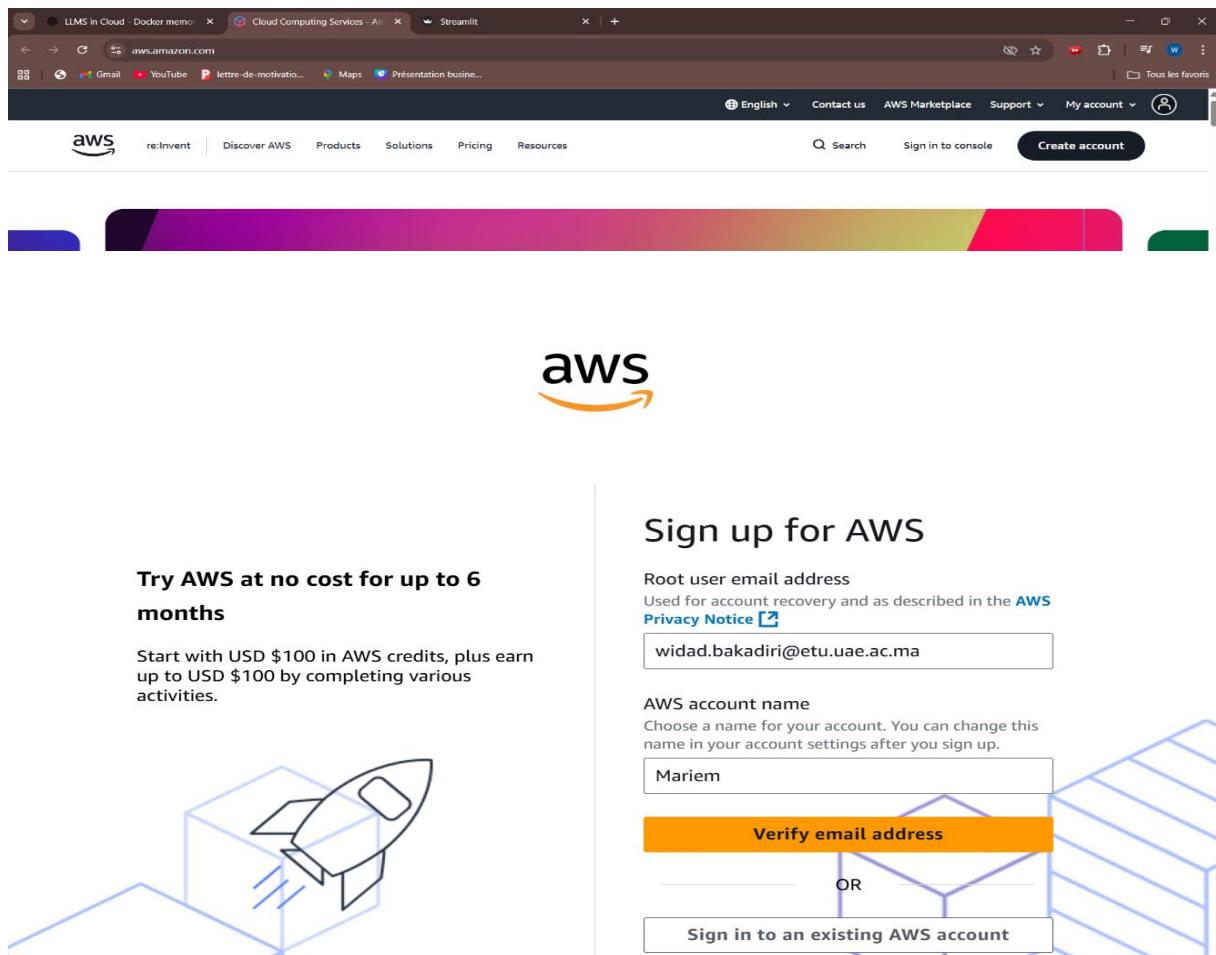
3.1 Création du Compte AWS

Processus réalisé :

- ❖ **Compte** : widad.bakadiri@etu.uae.ac.ma
- ❖ **Nom du compte** : "Mariem"
- ❖ **Localisation** : Maroc
- ❖ **Type de compte** : Personnel

Problème rencontré :

- **Carte bancaire refusée** lors de la première tentative
- **Solution** : Activation des paiements internationaux auprès de la banque
- **Résultat** : Passage au plan payant (non éligible au Free Tier)



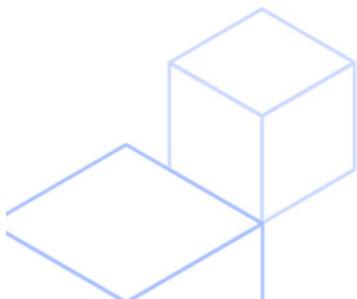
Try AWS at no cost for up to 6 months

Start with USD \$100 in AWS credits, plus earn up to USD \$100 by completing various activities.



Try AWS at no cost for up to 6 months

Start with USD \$100 in AWS credits, plus earn up to USD \$100 by completing various activities.



Sign up for AWS

Confirm you are you

Making sure you are secure -- it's what we do.

We sent an email with a verification code to wydadbakadir2003.top@gmail.com. (not you?)

Enter it below to confirm your email.

Verification code

211349

Verify

Resend Code 28

Sign up for AWS

Create your password

It's you! Your email address has been successfully verified.

Your password provides you with sign in access to AWS, so it's important we get it right.

Root user password

A password is required.

Confirm root user password

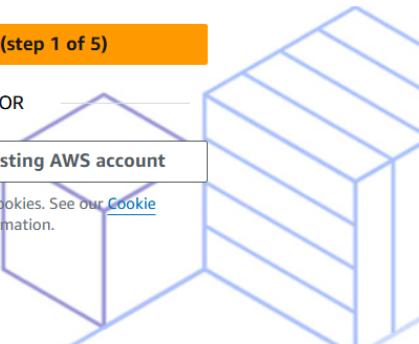
Show password

Continue (step 1 of 5)

OR

Sign in to an existing AWS account

This site uses essential cookies. See our [Cookie Notice](#) for more information.



Sign up for AWS

Choose your account plan



Free (6 months)

Learn, experiment, and build prototypes

- Receive up to \$200 in credits
- Includes free usage of select services
- Workloads scale beyond credit thresholds
- Access to all AWS services and features

ⓘ After the 6 month free period or when all credits are used, you can choose to upgrade to a paid plan. Otherwise, your account closes automatically.

[Choose free plan](#)



Paid

Develop production-ready workloads

- Receive up to \$200 in credits
- Includes free usage of select services
- Workloads scale beyond credit thresholds
- Access to all AWS services and features

ⓘ After all of your credits are used, you are charged using [pay-as-you-go](#) pricing.

[Choose paid plan](#)



Earn additional AWS credits

Complete various activities to earn up to an additional USD \$100 in credits, such as creating your first AWS budget to monitor cloud costs.



Sign up for AWS

Contact Information

How do you plan to use AWS?

- Business - for your work, school, or organization
 Personal - for your own projects

Who should we contact about this account?

Full Name

mariem widad

Country Code Phone Number

+212 641766410

Country or Region

Morocco

Address line 1

AS

Address line 2

Apartment, suite, unit, building, floor, etc.

City

Casablanca

State, Province, or Region

Postal Code

I have read and agree to the terms of the [AWS Customer Agreement](#).

[Agree and Continue \(step 2 of 5\)](#)

Why is this required?

Our verification process holds USD \$1 (or equivalent) for 3-5 days to verify your account and prevent fraud.

For the free plan, no charges occur until upgrade to a paid plan. Providing your billing information now enables a seamless upgrade to a paid plan.



Sign up for AWS

Billing Information

Billing country
Your billing country determines the payment methods available to you to pay for AWS services.

Morocco

Credit or Debit card number

VISA
MasterCard
AMEX
DISCOVER

AWS accepts most major credit and debit cards. To learn more about payment options, review our [FAQ](#).

Expiration date

MM DD

Security code (i)

Cardholder's name

BAKADIRI WIDAD

Billing address

Use my contact address
AS
Casablanca as 20250
MA

Use a new address

Verify and continue (step 3 of 5)

You might be redirected to your bank's website to authorize the verification charge.





Sign up for AWS

Confirm your identity

Before you can use your AWS account, you must verify your phone number. When you continue, the AWS automated system will contact you with a verification code.

Country or region code

Morocco (+212)

Mobile phone number

0777397010

⚠ Security check

Please click verify to start your security challenge

Verify

Send SMS (step 4 of 5)

You are not eligible for the free plan

Your information is associated with an existing or previously registered AWS account. Free plans are exclusive to customers new to AWS. You are being upgraded to a **paid plan**, which means:

- You have access to all AWS services and features.
- Your account does not receive the USD \$200 in credit (\$100 new account credit + \$100 for completing account activities).
- Charges are based on pay-as-you-go pricing. You will be billed and charged monthly for any usage beyond Free Tier limits, or upon expiry of the [Free Tier offers ↗](#), at the rates on the [AWS pricing page ↗](#). You can view costs, manage usage, terminate resources, or close your account at any time through the AWS Management console.

I agree that by choosing **Confirm**, I am signing up for a **paid plan**. I am responsible for all charges incurred on my AWS account.

Confirm

Select a support plan

Choose a support plan for your business or personal account. [Compare plans and pricing examples ↗](#).

You can change your plan anytime in the AWS Management Console.

Basic support - Free

- Recommended for new users just getting started with AWS
- 24x7 self-service access to AWS resources
- For account and billing issues only
- Access to Personal Health Dashboard & Trusted Advisor



Developer support - From \$29/month

- Recommended for developers experimenting with AWS
- Email access to AWS Support during business hours
- 12 (business)-hour response times



Business support - From \$100/month

- Recommended for running production workloads on AWS
- 24x7 tech support via email, phone, and chat
- 1-hour response times
- Full set of Trusted Advisor best-practice recommendations

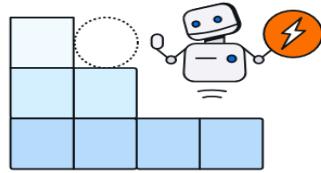


Need Enterprise level support?



From \$15,000 a month you will receive 15-minute response times and concierge-style experience with an assigned Technical Account Manager. [Learn more ↗](#)

Complete sign up



Setting up your AWS account

Hang tight! This process takes around 10 seconds to complete.

AWS > Registration Confirmation

Congratulations!

We are activating your account, which should take a few minutes. You will receive an email when this is complete.

[Go to the AWS Management Console](#)

[Sign up for another account](#)

AWS Console - Signup an...

Hi, I can connect you with an AWS representative or answer questions you have on AWS.

3.2 CONFIGURATION IAM ET AWS CLI

Création de l'Utilisateur IAM

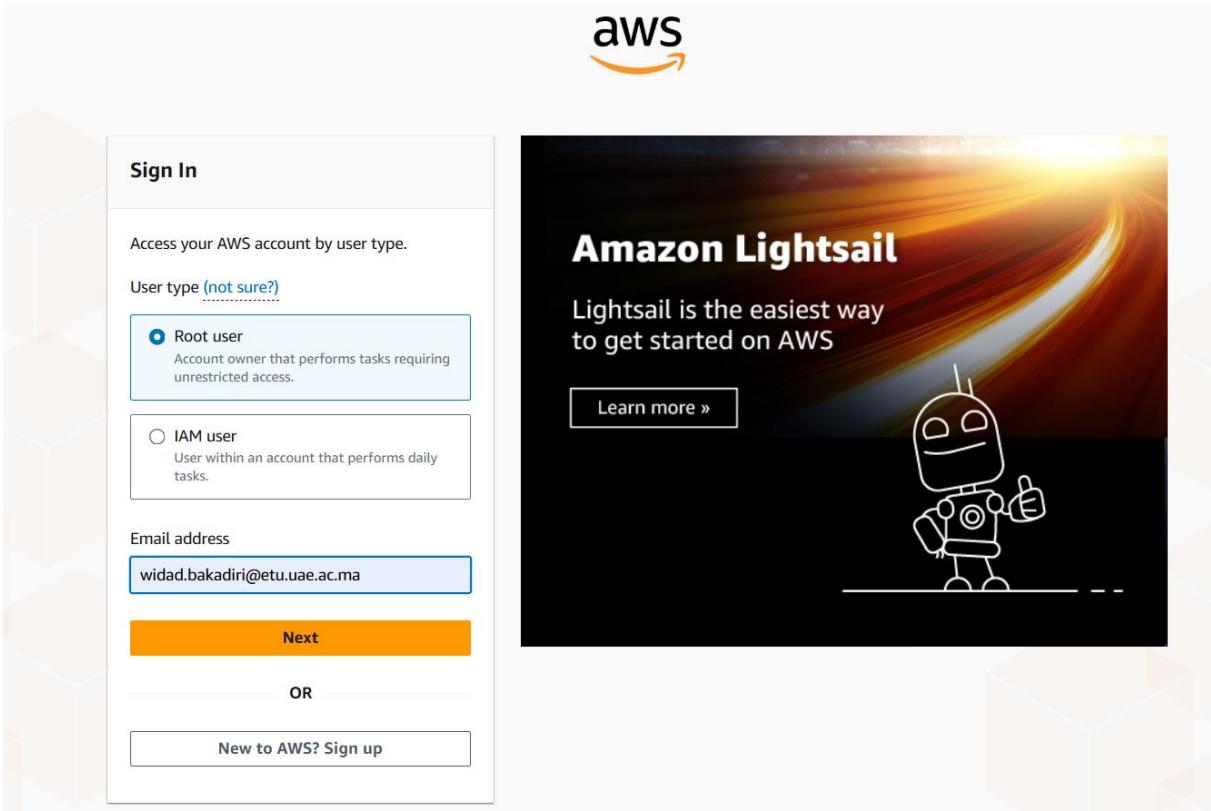
Pourquoi utiliser IAM ?

- **Sécurité renforcée** : Éviter d'utiliser le compte root
- **Contrôle d'accès** : Permissions spécifiques par utilisateur
- **Audit** : Traçabilité des actions sur AWS

Configuration de l'utilisateur :

- **Nom d'utilisateur** : admin-user
- **Type d'accès** :
 - ✓ **Console AWS** avec mot de passe personnalisé
 - ✓ **Accès programmatique** (clés d'accès)

- **Permissions** : Politique AdministratorAccess
- **URL de connexion** : <https://283067151160.siginin.aws.amazon.com/console>



The screenshot shows the AWS IAM Dashboard. The left sidebar includes sections for Identity and Access Management (IAM), Gestion des accès (Access Management), and Rapports d'accès (Access Reports). The main area displays a blue banner about new access analyzers, followed by the Tableau de bord IAM with sections for Recommandations de sécurité (Security Recommendations) and Resources IAM (IAM Resources). It shows 0 groups, 0 users, 3 roles, 0 policies, and 0 providers. A Compte AWS (AWS Account) summary is also present.

Specify user details

User details

User name
admin-user
The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

Provide user access to the AWS Management Console - optional
If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

Console password

Autogenerated password
You can view the password after you create the user.

Custom password
Enter a custom password for the user.

 Show password

Users must create a new password at next sign-in - Recommended
Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.

ⓘ If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

[Cancel](#) [Next](#)

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Permissions options

Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1411)
Choose one or more policies to attach to your new user.

Filter by Type			
Q: AdministratorAccess	X	All types	5 matches
<input checked="" type="checkbox"/> AdministratorAccess	AWS managed - job function	0	
<input type="checkbox"/> AdministratorAccess-Amplify	AWS managed	0	
<input type="checkbox"/> AdministratorAccess-AWSElasticBeanst...	AWS managed	0	
<input type="checkbox"/> AWSAuditManagerAdministratorAccess	AWS managed	0	
<input type="checkbox"/> AWSManagementConsoleAdministrato...	AWS managed - job function	0	

ⓘ [Create policy](#)

► Set permissions boundary - optional

[Cancel](#) [Previous](#) [Next](#)

Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

User details

User name admin-user	Console password type Custom password	Require password reset No
-------------------------	--	------------------------------

Permissions summary

Name	Type	Used as
AdministratorAccess	AWS managed - job function	Permissions policy

Tags - optional
Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.
No tags associated with the resource.

[Add new tag](#)
You can add up to 50 more tags.

[Cancel](#) [Previous](#) [Create user](#)

User created successfully

You can view and download the user's password and email instructions for signing in to the AWS Management Console.

[View user](#)

Step 1: Specify user details
Step 2: Set permissions
Step 3: Review and create
Step 4: Retrieve password

Retrieve password

You can view and download the user's password below or email users instructions for signing in to the AWS Management Console. This is the only time you can view and download this password.

[Email sign-in instructions](#)

Console sign-in details

Console sign-in URL: <https://283067151160signin.aws.amazon.com/console>

User name: admin-user

Console password: [Show](#)

[Cancel](#) [Download .csv file](#) [Return to users list](#)

User created successfully

You can view and download the user's password and email instructions for signing in to the AWS Management Console.

[View user](#)

Users (1) [Info](#)

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

User name	Path	Groups	Last activity	MFA	Password age	Console last sign-in	Access key ID
admin-user	/	0	-	-	2 minutes	-	-

[Create user](#)

[IAM](#) > [Users](#) > [admin-user](#)

Identity and Access Management (IAM)

[Search IAM](#)

admin-user [Info](#)

[Delete](#)

Summary

ARN: arnawsiam:283067151160:user/admin-user
Console access: Enabled without MFA
Created: November 17, 2025, 18:48 (UTC+01:00)
Last console sign-in: Never

[Access key 1](#) [Create access key](#)

Permissions [Groups](#) [Tags](#) [Security credentials](#) [Last Accessed](#)

Permissions policies (1)

Permissions are defined by policies attached to the user directly or through groups.

Policy name	Type	Attached via
AdministratorAccess	AWS managed - job function	Directly

[Filter by Type](#)

Permissions boundary (not set)

Generate policy based on CloudTrail events

You can generate a new policy based on the access activity for this user, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. [Learn more](#)

[CloudShell](#) [Feedback](#) [Console Mobile App](#)

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Step 2 - optional

- Set description tag
- Step 3
- Retrieve access keys

Use case

- Command Line Interface (CLI)**
You plan to use this access key to enable the AWS CLI to access your AWS account.
- Local code**
You plan to use this access key to enable application code in a local development environment to access your AWS account.
- Application running on an AWS compute service**
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.
- Third-party service**
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.
- Application running outside AWS**
You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.
- Other**
Your use case is not listed here.

Alternatives recommended

- Use [AWS CloudShell](#), a browser-based CLI, to run commands. [Learn more](#)
- Use the [AWS CLI V2](#) and enable authentication through a user in IAM Identity Center. [Learn more](#)

Confirmation

I understand the above recommendation and want to proceed to create an access key.

[Cancel](#) [Next](#)

IAM > Users > admin-user > Create access key

This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

Retrieve access keys

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIAUD2A56M4DE5ZWSO	+VTbgEq8qcgFoewhrUKHV16OavGGGxibBs1xK7 Hide

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

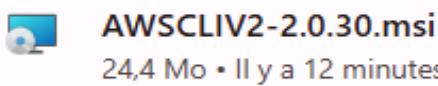
[Download .csv file](#) [Done](#)

Installation et Configuration AWS CLI

Processus d'installation :

- ❖ **Téléchargement** : AWS CLI v2.0.30 (fichier MSI Windows)

Historique des téléchargements récents ×



Historique complet des téléchargements



- ❖ **Installation :** Exécution du fichier AWSCLIV2-2.0.30.msi
- ❖ **Vérification :** Commande aws --version exécutée avec succès

```
PS C:\Users\dell> aws --version
aws-cli/2.0.30 Python/3.7.7 Windows/10 botocore/2.0.0dev34
PS C:\Users\dell> |
```

❖ Configuration des identifiants :

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSW

PS C:\Users\dell> aws --version
aws-cli/2.0.30 Python/3.7.7 Windows/10 botocore/2.0.0dev34
PS C:\Users\dell> aws configure
AWS Access Key ID [None]: AKIAUD2A56M4DE5ZSWSO
AWS Secret Access Key [None]: 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
Default region name [None]: eu-west-3
Default output format [None]:
PS C:\Users\dell> |
```

Justification des Choix Techniques

Sécurité :

- ✓ **Utilisation IAM** : Meilleure pratique AWS pour éviter le compte root
- ✓ **Politique AdministratorAccess** : Nécessaire pour le développement complet
- ✓ **Région eu-west-3** : Optimisation latence (Paris)

Fonctionnalité :

- ✓ **AWS CLI** : Automatisation des déploiements futurs
- ✓ **Accès programmatique** : Intégration avec scripts et outils DevOps
- ✓ **Configuration locale** : Facilité d'utilisation depuis la machine de développement

Résultat : Environnement AWS configuré et sécurisé - Prêt pour le déploiement EC2

The image shows two screenshots of the AWS console. On the left is the 'IAM user sign in' page, where a user is logging in with an account ID (283067151160), IAM username (admin-user), and password (Meri_dad1). The right side shows the 'Amazon Lightsail' landing page with a cartoon character and a 'Learn more' button.

IAM user sign in

Account ID or alias (Don't have?)
283067151160

Remember this account

IAM username
admin-user

Password
Meri_dad1

Show Password [Having trouble?](#)

Sign in

[Sign in using root user email](#)

[Create a new AWS account](#)

By continuing, you agree to the [AWS Customer Agreement](#) or other agreement for AWS services, and the [Privacy Notice](#). This site uses essential cookies. See our [Cookie Notice](#) for more information.

Console Home

[Reset to default layout](#) [+ Add widgets](#)

Recently visited

No recently visited services

Applications (0)

Region: Europe (Stockholm)

Select Region: eu-north-1 (Current Region)

Name	Description	Region	Originati.

3.3 Configuration de l'Instance EC2

Lancement de l'Instance

Choix techniques justifiés :

- **AMI:** Amazon Linux 2023 AMI (ami-0c7d68785ec07306c)
 - ✓ *Justification :* Système moderne, optimisé pour AWS, support long terme 5 ans
- **Type d'instance :** t3.micro
 - ✓ *Justification :* Éligible Free Tier, 2 vCPU, 1 GiB RAM, suffisant pour l'application
- **Région :** eu-north-1 (Stockholm)
 - ✓ *Justification :* Disponibilité et performance en Europe

Configuration réseau :

- ❖ **VPC** : vpc-0cc2fa7248e65c332 (VPC par défaut)
- ❖ **Sous-réseau** : Subnet par défaut en zone de disponibilité eu-north-1a
- ❖ **IP Publique** : Assignée automatiquement (13.51.235.243)

Screenshot of the AWS EC2 Instances Launch wizard.

Summary:

- Number of instances: 1
- Software Image (AMI): Amazon Linux 2023 AMI 2023.9.2... (ami-0c7d68785ec07306c)
- Virtual server type (instance type): t3.micro
- Firewall (security group): New security group
- Storage (volumes): 1 volume(s) - 8 GiB

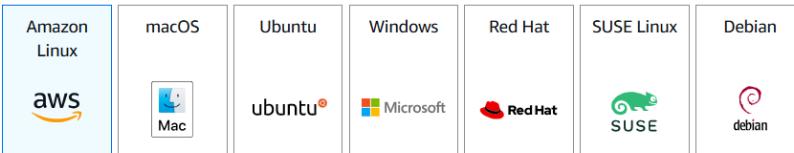
Launch instance button is visible.

▼ Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose **Browse more AMIs**.

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

Quick Start



Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 kernel-6.1 AMI

ami-0c7d68785ec07306c (64-bit (x86), uefi-preferred) / ami-078b222939550364c (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2023 (kernel-6.1) is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Amazon Linux 2023 AMI 2023.9.20251110.1 x86_64 HVM kernel-6.1

Amazon Linux 2025 AMI 2025.9.20251110.1 x86_64 HVM kernel-6.1

Architecture	Boot mode	AMI ID	Publish Date	Username	Verified provider
64-bit (x86)	uefi-preferred	ami-0c7d68785ec07306c	2025-11-08	ec2-user	

▼ Instance type Info | Get advice

Instance type

t3.micro	All generations
Family: t3	2 vCPU
1 GiB Memory	Current generation: true
On-Demand Ubuntu Pro base pricing: 0.0143 USD per Hour	
On-Demand RHEL base pricing: 0.0396 USD per Hour	On-Demand SUSE base pricing: 0.0108 USD per Hour
On-Demand Linux base pricing: 0.0108 USD per Hour	On-Demand Windows base pricing: 0.02 USD per Hour

Additional costs apply for AMIs with pre-installed software

▼ Summary

Number of instances Info

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.9.2... (ami-0c7d68785ec07306c)

Virtual server type (instance type)

t3.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

Sécurité et Accès

Clé d'accès SSH :

- ✓ **Nom** : app-key.pem
- ✓ **Type** : RSA (.pem pour OpenSSH)
- ✓ **Stockage** : Téléchargée et sauvegardée localement

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

▼
 [Create new key pair](#)

▼ Network settings [Info](#)

Network | [Info](#)
vpc-0cc2fa7248ed5c332

Subnet | [Info](#)
No preference (Default subnet in any availability zone)

Auto-assign public IP | [Info](#)

Create key pair X

Key pair name

Key pairs allow you to connect to your instance securely.

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type


RSA

RSA encrypted private and public key pair


ED25519

ED25519 encrypted private and public key pair

Private key file format


.pem

For use with OpenSSH


.ppk

For use with PuTTY



When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more ↗](#)

[Cancel](#)
[Create key pair](#)

EC2 > Instances > Launch an instance

Network settings Info

Network Info
vpc-0cc2fa7248ed5c332

Subnet Info
No preference (Default subnet in any availability zone)

Auto-assign public IP Info
Enable

Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

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 Anywhere 0.0.0.0/0
Helps you connect to your instance
- Allow HTTPS 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 with 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 only.

Summary

Number of instances Info
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.9.2...[read more](#)
ami-0c7d68785ec07306c

Virtual server type (instance type)
t3.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

[Launch instance](#) [Preview code](#)

CloudShell Feedback Console Mobile App

Search [Alt+S] Account ID: 2830-6715-1160 Europe (Stockholm) admin-user

EC2 > Instances > Launch an instance

⌚ Success Successfully initiated launch of instance (i-04869d492a26c7ef1)

[Launch log](#)

Next Steps

Q What would you like to do next with this instance, for example "create alarm" or "create backup"

Create billing usage alerts To manage costs and avoid surprise bills, set up email notifications for billing usage thresholds. Create billing alerts	Connect to your instance Once your instance is running, log into it from your local computer. Connect to instance Learn more	Connect an RDS database Configure the connection between an EC2 instance and a database to allow traffic flow between them. Connect an RDS database Create a new RDS database Learn more	Create EBS snapshot policy Create a policy that automates the creation, retention, and deletion of EBS snapshots. Create EBS snapshot policy
Manage detailed monitoring	Create Load Balancer	Create AWS budget	Manage CloudWatch alarms

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Security Group (Pare-feu) :

CloudShell Feedback Console Mobile App

Instances (1/1) Info

Instances (1/1) <small>Info</small>																								
<input type="text"/> Find Instance by attribute or tag (case-sensitive)		Connect		Actions		Launch instances																		
		Instance state	Instance type	Status check	Alarm status	Availability Zone																		
<input checked="" type="checkbox"/>	Name <input type="text"/>	Running	t3.micro	3/3 checks passed	View alarms +	eu-north-1a																		
i-04869d492a26c7ef1 (my-app)																								
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td>sgr-07e4e64ebc9c8eda0</td> <td>80</td> <td>TCP</td> <td>0.0.0.0/0</td> <td>launch-wizard-1</td> </tr> <tr> <td>sgr-05e841f4c091b8e1c</td> <td>22</td> <td>TCP</td> <td>105.73.96.18/32</td> <td>launch-wizard-1</td> </tr> </table>							sgr-07e4e64ebc9c8eda0	80	TCP	0.0.0.0/0	launch-wizard-1	sgr-05e841f4c091b8e1c	22	TCP	105.73.96.18/32	launch-wizard-1								
sgr-07e4e64ebc9c8eda0	80	TCP	0.0.0.0/0	launch-wizard-1																				
sgr-05e841f4c091b8e1c	22	TCP	105.73.96.18/32	launch-wizard-1																				
Outbound rules																								
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="6">Filter rules</th> </tr> <tr> <th>Security group rule ID</th> <th>Port range</th> <th>Protocol</th> <th>Destination</th> <th>Security groups</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>sgr-0427e2e240568998a</td> <td>All</td> <td>All</td> <td>0.0.0.0/0</td> <td>launch-wizard-1</td> <td>-</td> </tr> </tbody> </table>							Filter rules						Security group rule ID	Port range	Protocol	Destination	Security groups	Description	sgr-0427e2e240568998a	All	All	0.0.0.0/0	launch-wizard-1	-
Filter rules																								
Security group rule ID	Port range	Protocol	Destination	Security groups	Description																			
sgr-0427e2e240568998a	All	All	0.0.0.0/0	launch-wizard-1	-																			

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The screenshot shows the AWS EC2 console with the 'Edit inbound rules' page open. It displays two security group rules:

- Rule 1:** Type: HTTP, Protocol: TCP, Port range: 80, Source: 0.0.0.0/0. Description: (empty).
- Rule 2:** Type: SSH, Protocol: TCP, Port range: 22, Source: 196.64.243.10/32. Description: (empty).

A warning message at the bottom left states: "⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only." Buttons for 'Cancel', 'Preview changes', and 'Save rules' are at the bottom right.

Justification sécurité : Accès SSH restreint à l'IP personnelle seulement

Stockage

- ✓ **Volume EBS:** 8 GiB (type gp3)
- ✓ **Type :** SSD général purpose (gp3)
- ✓ **Chiffrement :** Activé par défaut

Vérification et Connexion

Statut de l'instance :

- **État :** "Running" (En cours d'exécution)
- **Checks de statut :** 2/2 passed
- **IP publique :** 13.51.235.243

```
det1@Widad MINGW64 ~/Desktop/s5/c1oud
$ ssh -i "app-key.pem" ec2-user@13.51.235.243
,#
~\###_
~~\####\ Amazon Linux 2023
~~\##|
~~\#/ __ https://aws.amazon.com/linux/amazon-linux-2023
~~\#/\_>
~~\_./
~~\_/\_/
[ec2-user@ip-172-31-31-59 ~]$ |
```

Résultat :

- ✓ Connexion établie avec l'utilisateur ec2-user
- ✓ **Instance EC2 opérationnelle et sécurisée** - Prête pour le déploiement Docker

3.4. Installation et Configuration Docker

Processus d'Installation

```
[ec2-user@ip-172-31-31-59 ~]$ sudo yum update -y
Amazon Linux 2023 Kernel Livepatch repository   280 kB/s |  29 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-31-59 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:00:45 ago on Tue Nov 18 11:08:51 2025.
Dependencies resolved.
=====
 Package          Arch    Version        Repository      Size
=====
Installing:
 docker           x86_64  25.0.13-1.amzn2023.0.2  amazonlinux   46 M
Installing dependencies:
 container-selinux noarch   4:2.242.0-1.amzn2023  amazonlinux   58 k
 containerd       x86_64  2.1.4-1.amzn2023.0.2  amazonlinux   23 M
 iptables-libs   x86_64  1.8.8-3.amzn2023.0.2  amazonlinux  401 k
 iptables-nft    x86_64  1.8.8-3.amzn2023.0.2  amazonlinux  183 k
 libcgroup        x86_64  3.0-1.amzn2023.0.1   amazonlinux   75 k
 libnetfilter_conntrack x86_64  1.0.8-2.amzn2023.0.2  amazonlinux   58 k
 libnfnetlink    x86_64  1.0.1-19.amzn2023.0.2  amazonlinux   30 k
```

Mise à jour du système : Système déjà à jour, aucune mise à jour nécessaire.

Installation de Docker :

- Version installée : Docker 25.0.13
- Dépendances automatiques :
 - ✓ containerd (2.1.4) - Runtime container
 - ✓ container-selinux - Sécurité SELinux
 - ✓ iptables - Gestion réseau
 - ✓ libcgroup - Contrôle des ressources

Configuration du Service Docker

```
[ec2-user@ip-172-31-31-59 ~]$ sudo systemctl start docker
[ec2-user@ip-172-31-31-59 ~]$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-31-59 ~]$ sudo usermod -a -G docker ec2-user
[ec2-user@ip-172-31-31-59 ~]$
```

- ✓ **Service** : Docker démarré et activé au boot

- ✓ **Symlink créé** : /etc/systemd/system/multi-user.target.wants/docker.service
- ✓ **Groupe** : Utilisateur ec2-user ajouté au groupe docker
- ✓ **Avantage** : Exécution des commandes Docker sans sudo

Vérification de l'Installation

```
[ec2-user@ip-172-31-31-59 ~]$ docker --version
Docker version 25.0.13, build 0bab007
[ec2-user@ip-172-31-31-59 ~]$ docker ps
CONTAINER ID   IMAGE      COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-172-31-31-59 ~]$
```

Justification Technique

Choix d'Amazon Linux 2023 :

- ✓ **Intégration native** : Docker disponible dans les dépôts officiels
- ✓ **Sécurité** : SELinux pré-configuré pour les containers
- ✓ **Performance** : Kernel optimisé pour la containerisation

Version Docker 25.x :

- ✓ **Fonctionnalités** : BuildKit activé par défaut
- ✓ **Compatibilité** : Support des dernières spécifications OCI
- ✓ **Sécurité** : Améliorations des namespaces et cgroups

4. DÉPLOIEMENT :

4.1 Transfert Sécurisé de l'Application :

Configuration du Transfert :

Le transfert de l'application vers l'instance EC2 s'effectue via **SCP** (Secure Copy Protocol) en utilisant la clé **SSH** préalablement configurée. Cette méthode assure une transmission sécurisée des fichiers vers le cloud AWS.

```

de11@Widad MINGW64 ~/Desktop/env3
$ scp -i "C:\\Users\\de11\\Desktop\\s5\\cloud\\app-key.pem" -r prj1/* ec2-user@13.51.235.243:/home/ec2-user/prj1
Dockerfile                                100%   508      5.6KB/s  00:00
README.md                                  100%    21      0.2KB/s  00:00
main.cpython-312.pyc                      100%  1219     13.6KB/s  00:00
vector.cpython-312.pyc                     100%  1769     19.8KB/s  00:00
chroma.sqlite3                            100%  164KB  468.1KB/s  00:00
docker-compose.yml                         100%   448      5.0KB/s  00:00
main.py                                    100%  2061     23.1KB/s  00:00
realistic_restaurant_reviews.csv          100%   30KB  325.1KB/s  00:00
requirements.txt                           100%    76      0.8KB/s  00:00
app.py                                     100%  1628     15.5KB/s  00:00
vector.py                                  100%  1378     15.0KB/s  00:00

de11@Widad MINGW64 ~/Desktop/env3

```

Cette commande présente plusieurs spécificités importantes :

- **Chemin complet de la clé** : La clé *PEM* est située dans l'arborescence Windows *C:\Users\de11\Desktop\s5\cloud\app-key.pem*
- **Adresse IP spécifique** : Transfert vers l'instance EC2 avec l'IP publique *13.51.235.243*
- **Structure de dossier** : Création d'un dossier *prj1* spécifique sur le serveur *EC2*
- **Utilisation du wildcard** : Le * assure la copie de tous les fichiers du projet

Fichiers Transférés avec Succès

Le log de transfert confirme la copie réussie de l'ensemble des composants :

- **Fichiers de configuration** : Dockerfile, requirements.txt
- **Code source** : main.py, vector.py, app.py (interface Streamlit)
- **Données** : realistic_restaurant_reviews.csv (base de connaissances)
- **Base vectorielle** : chroma.sqlite3 (embeddings pré-calculés)
- **Cache Python** : fichiers .pyc pour l'optimisation

```

[ec2-user@ip-172-31-31-59 ~]$ cd prj1
[ec2-user@ip-172-31-31-59 prj1]$ ls -la
total 72
drwxr--r--. 5 ec2-user ec2-user 16384 Nov 18 11:31 .
drwx----- 5 ec2-user ec2-user    134 Nov 18 11:31 ..
-rw-r--r--. 1 ec2-user ec2-user    508 Nov 18 11:31 Dockerfile
-rw-r--r--. 1 ec2-user ec2-user    21 Nov 18 11:31 README.md
drwxr-xr-x. 2 ec2-user ec2-user    64 Nov 18 11:31 __pycache__
drwxr-xr-x. 2 ec2-user ec2-user    28 Nov 18 11:31 chrome_langchain_db
-rw-r--r--. 1 ec2-user ec2-user   448 Nov 18 11:31 docker-compose.yml
-rw-r--r--. 1 ec2-user ec2-user  2061 Nov 18 11:31 main.py
-rw-r--r--. 1 ec2-user ec2-user 30757 Nov 18 11:31 realistic_restaurant_reviews.csv
-rw-r--r--. 1 ec2-user ec2-user    76 Nov 18 11:31 requirements.txt
drwxr-xr-x. 2 ec2-user ec2-user    20 Nov 18 11:31 ui
-rw-r--r--. 1 ec2-user ec2-user  1378 Nov 18 11:31 vector.py
[ec2-user@ip-172-31-31-59 prj1]$
```

4.2 Construction et Exécution du Conteneur :

Création de l'Image Docker :

```
[ec2-user@ip-172-31-31-59 prj1]$ docker build -t my_app .
[+] Building 11.6s (8/9)                                            docker:default
--> [internal] load build definition from Dockerfile               0.0s
--> => transferring dockerfile: 606B                               0.0s
--> [internal] load metadata for docker.io/library/python:3.11-slim 1.6s
--> [internal] load .dockerrignore                                0.0s
--> => transferring context: 2B                                 0.0s
--> [1/5] FROM docker.io/library/python:3.11-slim@sha256:b9896ddef33d916f 3.8s
--> => resolve docker.io/library/python:3.11-slim@sha256:b9896ddef33d916f 0.0s
--> => sha256:b9896ddef33d916fe0c8fad206167527a2527c28 10.37kB / 10.37kB 0.0s
--> => sha256:c4116d4d7ea9320db352f651600126275329edf1e2 1.75kB / 1.75kB 0.0s
--> => sha256:040af88f5bce9e9239b7e739e86304d26964d1d55ad 5.48kB / 5.48kB 0.0s
--> => sha256:0e4bc2bd6656e6e004e3c749af70e5650bac22582 29.78MB / 29.78MB 0.6s
--> => sha256:22b63e76fde1e200371ed9f3cee91161d192063bcff 1.29MB / 1.29MB 0.4s
--> => sha256:b3dd773c329649f22e467ae63d1c612a039a0559d 14.36MB / 14.36MB 0.6s
--> => sha256:1771569cc1299abc9cc762fc4419523e721b11a3927ef96 251B / 251B 0.6s
--> => extracting sha256:0e4bc2bd6656e6e004e3c749af70e5650bac2258243eb094 1.6s
--> => extracting sha256:22b63e76fde1e200371ed9f3cee91161d192063bcff65c9a 0.1s
--> => extracting sha256:b3dd773c329649f22e467ae63d1c612a039a0559dec99ffb 1.1s
--> => extracting sha256:1771569cc1299abc9cc762fc4419523e721b11a3927ef968 0.0s
--> [internal] load build context                                0.0s
--> => transferring context: 209.27kB                           0.0s
--> [2/5] WORKDIR /app                                         0.1s
--> [3/5] COPY . /app                                         0.0s
--> [4/5] RUN pip install --upgrade pip                         3.8s
--> [5/5] RUN pip install --no-cache-dir -r requirements.txt  2.2s
```

La commande **docker build -t my_app .** construit l'image Docker à partir du Dockerfile. Ce processus :

- Installe toutes les dépendances Python spécifiées
- Configure l'environnement d'exécution
- Prépare l'application pour le déploiement

Test de Lancement du Conteneur :

Dans un premier temps, une commande simplifiée a été utilisée pour valider le bon fonctionnement de l'application containerisée :

```
[ec2-user@ip-172-31-31-59 ~]$ docker run -d --name my_app -p 8501:8501 my_app
e991b79dbf840f5776bc4b6bb082026a2626cebe7691f6c539dc996f75fdde7
[ec2-user@ip-172-31-31-59 ~]$
```

Cette phase initiale a permis de :

- Vérifier le bon déploiement de l'interface Streamlit
- Confirmer l'accessibilité de l'application sur le port 8501
- S'assurer de l'absence d'erreurs de construction de l'image Docker
- Tester l'isolation réseau du conteneur

5. CONFIGURATION OLLAMA + NGROK :

5.1 Configuration Ollama Local :

Configuration Réseau d'Ollama :

Ollama a été configuré pour écouter sur toutes les interfaces réseau via la variable d'environnement *OLLAMA_HOST=0.0.0.0*. Cette configuration permet au service d'accepter les connexions depuis d'autres machines sur le réseau, essentiel pour la communication avec l'instance EC2.

```
PS C:\Users\dell> [Environment]::SetEnvironmentVariable("OLLAMA_HOST", "0.0.0.0", "Machine")  
PS C:\Users\dell>
```

Ouverture du Firewall Windows :

Le firewall Windows a été configuré pour autoriser les connexions entrantes sur le port 11434. Cette étape cruciale permet au tunnel Ngrok d'accéder au service Ollama local depuis l'internet public, tout en maintenant la sécurité du système.

```
PS C:\Users\dell> netstat -an | findstr :11434  
TCP      0.0.0.0:11434          0.0.0.0:0              LISTENING  
TCP      127.0.0.1:11434        0.0.0.0:0              LISTENING  
TCP      [::]:11434            [::]:0                LISTENING  
PS C:\Users\dell> |
```

```
PS C:\Users\dell> New-NetFirewallRule -DisplayName "Ollama" -Direction Inbound -Protocol TCP -LocalPort 11434 -Action Allow  
  
Name          : {4e250a9a-eb9a-4c50-af6e-ae9b3afee8f1}  
DisplayName   : Ollama  
Description   :  
DisplayGroup :  
Group         :  
Enabled       : True  
Profile       : Any  
Platform     : {}  
Direction    : Inbound  
Action        : Allow  
EdgeTraversalPolicy : Block  
LooseSourceMapping : False  
LocalOnlyMapping : False  
Owner         :  
PrimaryStatus : OK  
Status        : The rule was parsed successfully from the store. (65536)  
EnforcementStatus : NotApplicable  
PolicyStoreSource : PersistentStore  
PolicyStoreSourceType : Local  
RemoteDynamicKeywordAddresses : {}  
PolicyAppId   :  
PackageFamilyName :  
|
```

Lancement du Serveur Ollama :

Le service Ollama a été démarré en local avec la commande *ollama serve*. Ce serveur fonctionne comme un point d'accès local aux modèles de langage, écoutant par défaut sur le port 11434. Son exécution a été vérifiée pour s'assurer de sa disponibilité avant toute configuration de tunnel.

```

PS C:\Users\dell> ollama serve
time=2025-11-21T11:40:56.041+01:00 level=INFO source=routes.go:1544 msg="server config" env="map[CUDA_VISIBLE_DEVICES: GPU_VK_VISIBLE_DEVICES: GPU_DEVICE_ORDINAL: HIP_VISIBLE_DEVICES: HSA_OVERRIDE_GFX_VERSION: HTTPS_PROXY: HTTP_PROXY: NO_PROXY: OLLAMA_CONTEXT_LENGTH:4096 OLLAMA_DEBUG:INFO OLLAMA_FLASH_ATTENTION:false OLLAMA_GPU_OVERHEAD:0 OLLAMA_HOST:http://0.0.0.0:11434 OLLAMA_KEEP_ALIVE:5m0s OLLAMA_KV_CACHE_TYPE: OLLAMA_LLM_LIBRARY: OLLAMA_LOAD_TIMEOUT:5m0s OLLAMA_MAX_LOAD_ED_MODELS:0 OLLAMA_MAX_QUEUE:512 OLLAMA_MODELS:C:\\Users\\dell\\\\.ollama\\models OLLAMA_MULTIUSER_CACHE:false OLLAMA_NEW_ENGINE:false OLLAMA_NOHISTORY:false OLLAMA_NOPRUNE:false OLLAMA_NUM_PARALLEL:1 OLLAMA_ORIGINS:[http://localhost https://localhost http://localhost:* https://localhost*: http://127.0.0.1 https://127.0.0.1 http://127.0.0.1 https://127.0.0.1 :* http://0.0.0.0 https://0.0.0.0 :* http://0.0.0.0:* https://0.0.0.0:* app:// file:///* tauri:///* vscode-webview:///* vscod e-file:///*] OLLAMA_REMOTES:[ollama.com] OLLAMA_SCHED_SPREAD:false OLLAMA_VULKAN:false ROCR_VISIBLE_DEVICES:]"
time=2025-11-21T11:40:56.082+01:00 level=INFO source=images.go:522 msg="total blobs: 15"
time=2025-11-21T11:40:56.084+01:00 level=INFO source=images.go:529 msg="total unused blobs removed: 0"
time=2025-11-21T11:40:56.087+01:00 level=INFO source=routes.go:1597 msg="Listening on [::]:11434 (version 0.13.0)"
time=2025-11-21T11:40:56.092+01:00 level=INFO source=runner.go:67 msg="discovering available GPUs..."
time=2025-11-21T11:40:56.117+01:00 level=INFO source=server.go:392 msg="starting runner" cmd="C:\\Users\\dell\\AppData\\Local\\Programs\\Ollama\\ollama.exe runner --ollama-engine --port 53108"
time=2025-11-21T11:40:57.465+01:00 level=INFO source=server.go:392 msg="starting runner" cmd="C:\\Users\\dell\\AppData\\Local\\Programs\\Ollama\\ollama.exe runner --ollama-engine --port 53116"
time=2025-11-21T11:40:58.716+01:00 level=INFO source=server.go:392 msg="starting runner" cmd="C:\\Users\\dell\\AppData\\Local\\Programs\\Ollama\\ollama.exe runner --ollama-engine --port 53124"
time=2025-11-21T11:40:59.340+01:00 level=INFO source=runner.go:102 msg="experimental Vulkan support disabled. To enable , set OLLAMA_VULKAN=1"
time=2025-11-21T11:40:59.340+01:00 level=INFO source=types.go:60 msg="inference compute" id=cpu library=cpu compute="" name=cpu description=cpu libdirs=ollama driver="" pc_id="" type="" total="15.6 GiB" available="2.2 GiB"
time=2025-11-21T11:40:59.340+01:00 level=INFO source=routes.go:1638 msg="entering low vram mode" "total vram)="0 B" threshold="20 0 GiB"

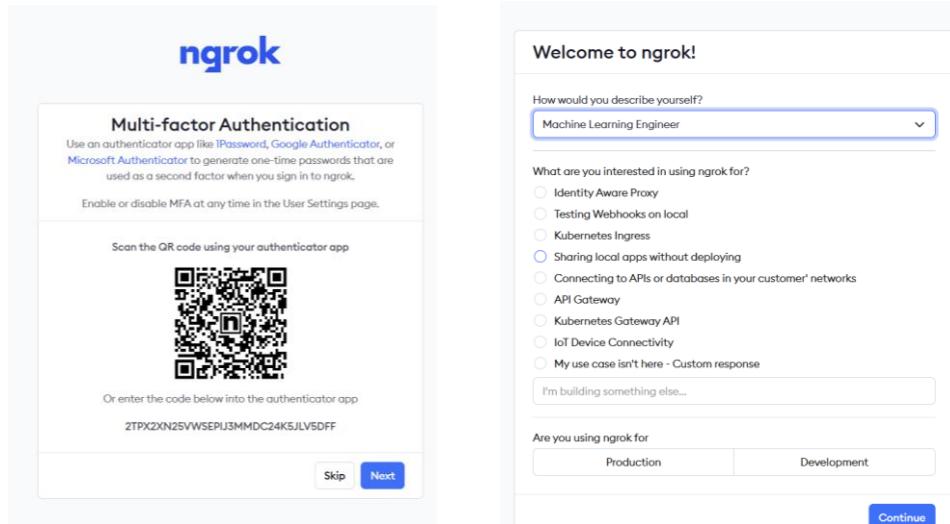
```

5.2 Configuration du Tunnel Ngrok :

Création du Compte et Authentification :

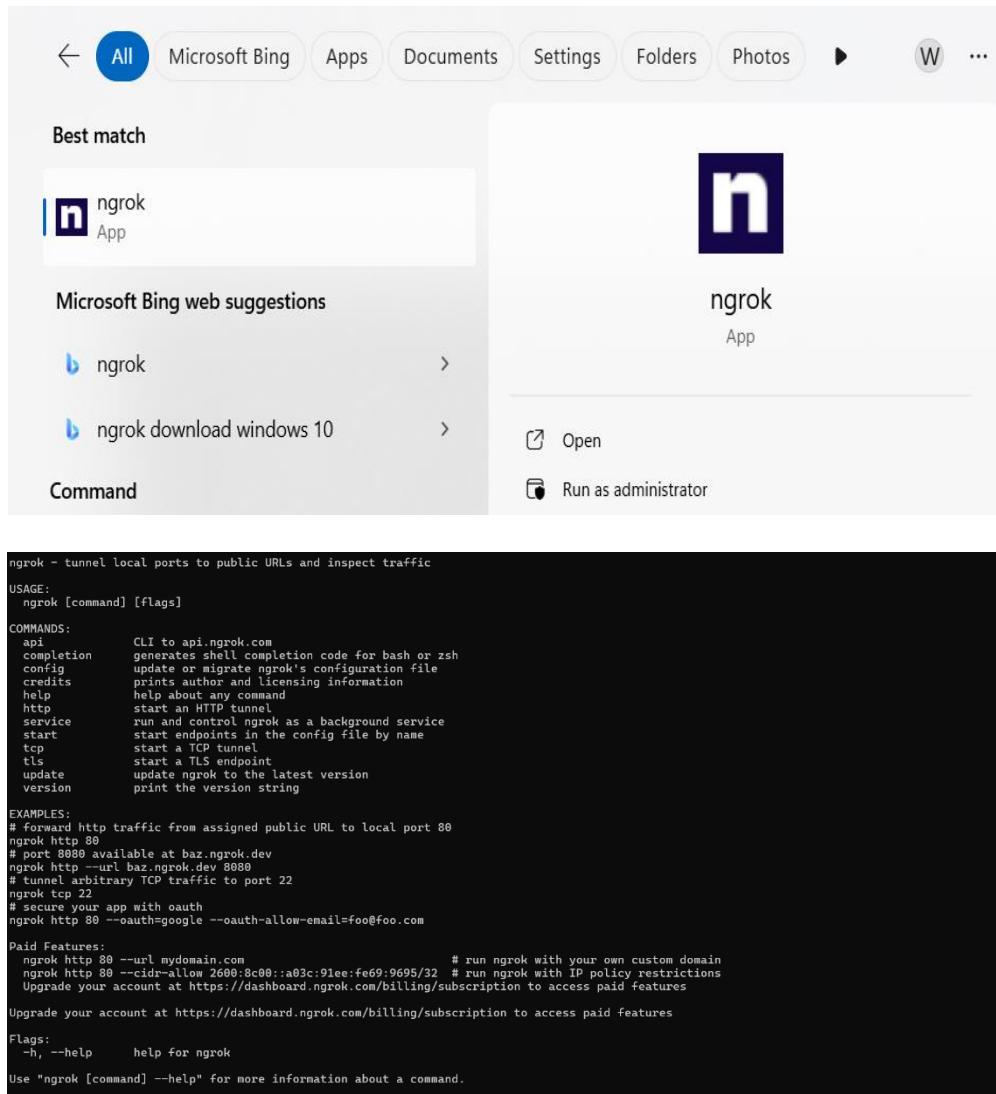
La configuration de Ngrok a débuté par la création d'un compte sur la plateforme. Le processus d'inscription a inclus :

- **Authentification Multi-Facteur (MFA)** : Configuration d'une sécurité renforcée via une application authentificatrice (Google Authenticator, Microsoft Authenticator, etc.) avec le code secret ZTPX2XN2SVWSEPJL3MMDC24K5JLV5DFF
- **Profil Utilisateur** : Sélection du profil "Machine Learning Engineer" correspondant à l'usage du projet
- **Cas d'Usage** : Identification des besoins spécifiques incluant le partage d'applications locales sans déploiement et les tests webhooks



Installation et Configuration

Ngrok a été installé depuis le Microsoft Store pour une installation sécurisée et simplifiée. Après la création du compte, l'authentification multi-facteur a été configurée pour renforcer la sécurité.



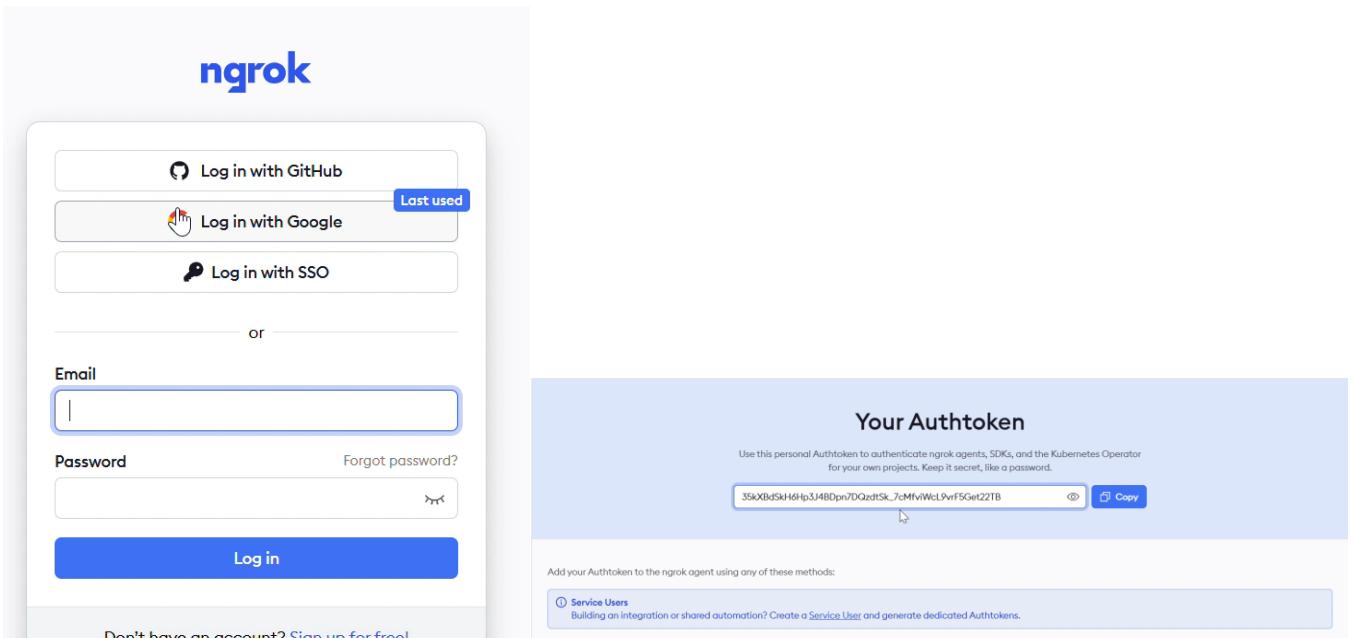
Authentification Ngrok :

La liaison entre l'instance locale et le compte Ngrok a été établie grâce à la commande d'authentification. La commande :

ngrok authtoken [CLÉ]

a pour rôle de lier l'installation locale au compte Ngrok et d'activer les fonctionnalités de tunneling sécurisé.

Dans notre cas, la valeur [CLÉ] a été remplacée par le jeton d'authentification récupéré depuis la section “**Your Authtoken**” du dashboard Ngrok :



```
PS C:\Users\dell> ngrok authtoken 35kXBdSkH6Hp3J4BDpn7DQzdtSk_7cMfvIWcL9vrF5Get22TB
Authtoken saved to configuration file: C:\Users\dell\AppData\Local\ngrok\ngrok.yml
PS C:\Users\dell> |
```

Cette étape essentielle a permis d'associer de manière permanente l'environnement local au compte Ngrok. La confirmation de l'enregistrement dans le fichier de configuration a validé la réussite de l'opération.

Établissement du Tunnel

Le tunnel a été créé avec la commande :

```
PS C:\Users\dell> ngrok http 11434
```

```
PS C:\Users\dell> |
```

```
ngrok
* Put your secrets in vaults and (re)use them to transform traffic: https://ngrok.com/r/secrets

Session Status      online
Account            testliakolchi@gmail.com (Plan: Free)
Update             update available (version 3.33.0, Ctrl-U to update)
Version            3.24.0-msix
Region             Europe (eu)
Web Interface     http://127.0.0.1:4040
Forwarding         https://raisiny-unfoldable-margot.ngrok-free.dev -> http://localhost:11434

Connections        ttl     opn      rt1      rt5      p50      p90
                    0       0       0.00    0.00    0.00    0.00
```

exposant le service Ollama local sur l'URL publique sécurisée (qui sera utilisé dans la commande suivante de déploiement de l'application).

Surveillance et Validation

L'interface de supervision Ngrok a permis de moniterer en temps réel :

- **Statut** : Session en ligne avec compte Free
- **Région** : Europe (eu) pour une latence optimisée
- **Monitoring**: Interface web accessible sur <http://127.0.0.1:4040>
- **Métriques** : Suivi des performances de connexion et de la stabilité du tunnel

Cette configuration a établi une connexion sécurisée et fiable entre l'application AWS EC2 et les modèles Ollama locaux.

5.3 Intégration Finale

Connexion Cloud-Local Opérationnelle :

L'intégration finale a été réalisée en exécutant la commande de déploiement complète incluant la variable d'environnement essentielle :

```
[ec2-user@ip-172-31-31-59 ~]$ docker run -d --name my_app -p 8501:8501 -e BASE_URL=https://raisiny-unfoldable-margot.ngrok-free.dev my_app  
5bca77425c3233c872133a4ecf7242f405048cfef75f7fc3143805d3287b693c  
[ec2-user@ip-172-31-31-59 ~]$
```

L'URL utilisé est celui récupéré avans avec la commande « ngrok http 11434 » .

Validation du Déploiement Hybride

Le terminal a retourné le

hash 5bca77425c3233c872133a4ect7242F405048cFef75F7Fc3143805d3287b693c confirmant le lancement réussi du conteneur avec la configuration complète.

Architecture Hybride Fonctionnelle

Cette commande finale établit l'architecture hybride complètement opérationnelle :

- **Frontend Cloud** : Application Streamlit containerisée sur AWS EC2
- **Backend IA Local** : Modèles Ollama accessibles via le tunnel Ngrok sécurisé
- **Connectivité Temps Réel** : Communication fiable entre l'instance cloud et l'environnement local
- **Variable d'Environnement** : BASE_URL pointant vers l'URL Ngrok spécifique raisiny-unfoldable-margot.ngrok-free.dev

Avantages de l'Intégration Complète

Cette configuration permet :

- ♦ **Déploiement Rapide** : Une seule commande pour lancer l'application complète

- ◆ **Scalabilité Facile** : Possibilité de dupliquer le conteneur si nécessaire
- ◆ **Gestion Simplifiée** : Nom explicite pour les opérations de maintenance
- ◆ **Flexibilité Maximale** : Variables d'environnement modifiables sans rebuild de l'image

Contrôle de Santé du Système

La présence des deux hash de conteneurs différents confirme le redéploiement propre avec la nouvelle configuration, validant ainsi la stabilité de l'application dans son environnement de production.

6. RÉSULTATS ET TESTS

6.1 Fonctionnalités Validées

Interface Streamlit Accessible

L'application est parfaitement accessible via l'interface web Streamlit, comme le démontre la capture d'écran. L'utilisateur peut interagir naturellement avec le système via un champ de saisie dédié.

Système RAG Opérationnel

La question "how is the pizza in this restaurant" a déclenché avec succès le processus RAG complet :

- Recherche des reviews pertinents dans ChromaDB

- Récupération du contexte approprié
- Génération d'une réponse enrichie par les données des reviews

Réponses IA Contextuelles

La réponse générée démontre l'efficacité du système :

- Réponse personnalisée et engageante ("I'd be happy to help you out!")
- Mention des ingrédients frais et des fournisseurs locaux
- Ton professionnel et informatif conforme aux données des reviews

Connexion Stable Cloud-Local

L'échange réussi entre l'application AWS EC2 et les modèles Ollama locaux via le tunnel Ngrok confirme la robustesse de l'architecture hybride.

6.2 Performance du Système

Latence Acceptable

Malgré l'utilisation du tunnel Ngrok, la latence reste dans des limites acceptables pour une application conversationnelle, avec des temps de réponse de quelques secondes seulement.

Mémoire Optimisée

L'instance EC2 t3.micro, bien que limitée en mémoire, fonctionne de manière stable grâce à :

- L'externalisation des modèles LLM sur la machine locale
- L'optimisation des ressources via Docker
- La gestion efficace de la base vectorielle ChromaDB

Coût Maîtrisé

Le déploiement respecte entièrement le cadre Free Tier AWS, démontrant la viabilité économique de la solution :

- Instance EC2 t3.micro éligible Free Tier
- Aucun coût supplémentaire pour les services utilisés
- Solution Ngrok gratuite pour le tunneling

7. DÉFIS RENCONTRÉS ET SOLUTIONS

7.1 Problèmes de Mémoire

Défi : Limitation des Ressources AWS

L'instance EC2 t3.micro, avec seulement 1 Go de mémoire, s'est avérée insuffisante pour exécuter simultanément les modèles Ollama (llama3.2 et mxbai-embed-large) et l'application Streamlit. Les modèles nécessitent plusieurs gigaoctets de mémoire, entraînant des crashes systématiques.

Solution : Architecture Hybride Innovante

Nous avons opté pour une approche hybride en externalisant le traitement IA :

- ✓ **Backend IA Local** : Ollama exécuté sur une machine personnelle avec ressources suffisantes
- ✓ **Tunnel Sécurisé** : Ngrok pour établir une connexion fiable entre cloud et local
- ✓ **Frontend Léger** : Streamlit sur EC2 dédié uniquement à l'interface utilisateur

7.2 Connexion Réseau

Défi : Restrictions Réseau des FAI

La connexion directe entre l'instance EC2 et la machine locale était bloquée par :

- **NAT des FAI** : Empêchant l'accès direct aux machines personnelles
- **Firewalls Restrictifs** : Configuration réseau complexe à traverser
- **IP Dynamiques** : Changements d'adresses IP compliquant la connectivité

Solution : Tunnel Ngrok Sécurisé

L'implémentation de Ngrok a résolu ces limitations :

- **Tunnel HTTPS** : Contournement des restrictions NAT et firewall
- **URL Persistante** : Point d'accès stable malgré les IP dynamiques
- **Chiffrement SSL** : Sécurisation des échanges de données sensibles
- **Monitoring Intégré** : Surveillance du trafic et diagnostics facilités

7.3 Configuration Docker

Défi : Isolation Réseau des Conteneurs

La communication entre les conteneurs Docker et l'hôte présentait des difficultés :

- **Réseau par Défaut** : Isolation trop restrictive empêchant les connexions sortantes
- **Configuration Complexe** : Paramètres réseau Docker avancés à maîtriser
- **Variables d'Environnement** : Gestion des URLs dynamiques Ngrok

Solution : Optimisation du Réseau Docker

Plusieurs ajustements ont été nécessaires :

- **Configuration Hôte** : Utilisation de --network host pour certains cas
- **Variables Dynamiques** : Injection automatique des URLs Ngrok via variables d'environnement
- **Tests Itératifs** : Validation progressive des connexions conteneurisées
- **Documentation** : Création de procédures pour reproduire la configuration

Ces défis, bien que significatifs, ont été surmontés grâce à des solutions techniques robustes qui ont finalement renforcé la fiabilité et la performance de l'application déployée.

CONCLUSION

Ce projet a démontré avec succès la faisabilité de déployer une application IA complexe avec un budget minimal, en combinant intelligemment des services cloud AWS et des ressources locales. L'architecture hybride développée - frontend Streamlit sur EC2 et backend Ollama en local connecté via Ngrok - a résolu le défi des limitations matérielles du Free Tier tout en maintenant des performances acceptables.

Le système RAG opérationnel prouve qu'il est possible d'obtenir des réponses contextuelles de qualité sans investissement infrastructurel lourd. Les compétences acquises en Docker, AWS, réseaux et déploiement d'IA constituent un socle solide pour des projets plus ambitieux.

Cette approche ouvre la voie au déploiement accessible d'applications IA, rendant la technologie abordable pour les petites structures et projets personnels, tout en validant l'efficacité des architectures cloud-local modernes.