Chatbot lab

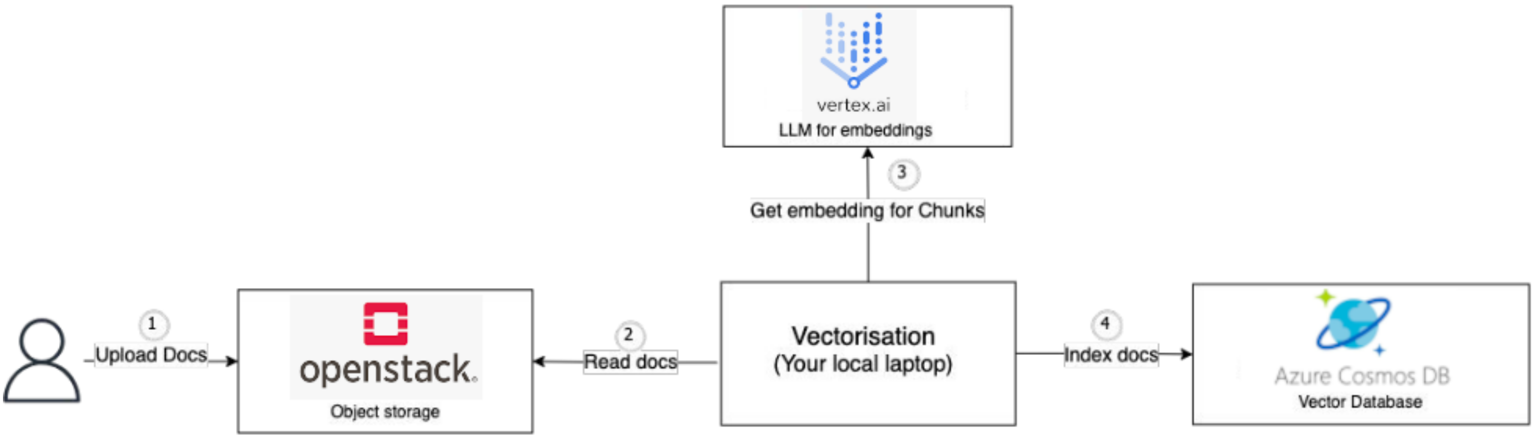
Authors: Olivia Manz and Samuel Roland in **Group D**

The whole project is available on this Github repository: https://github.com/samuelroland/CloudSys-labs/tree/main/lab3

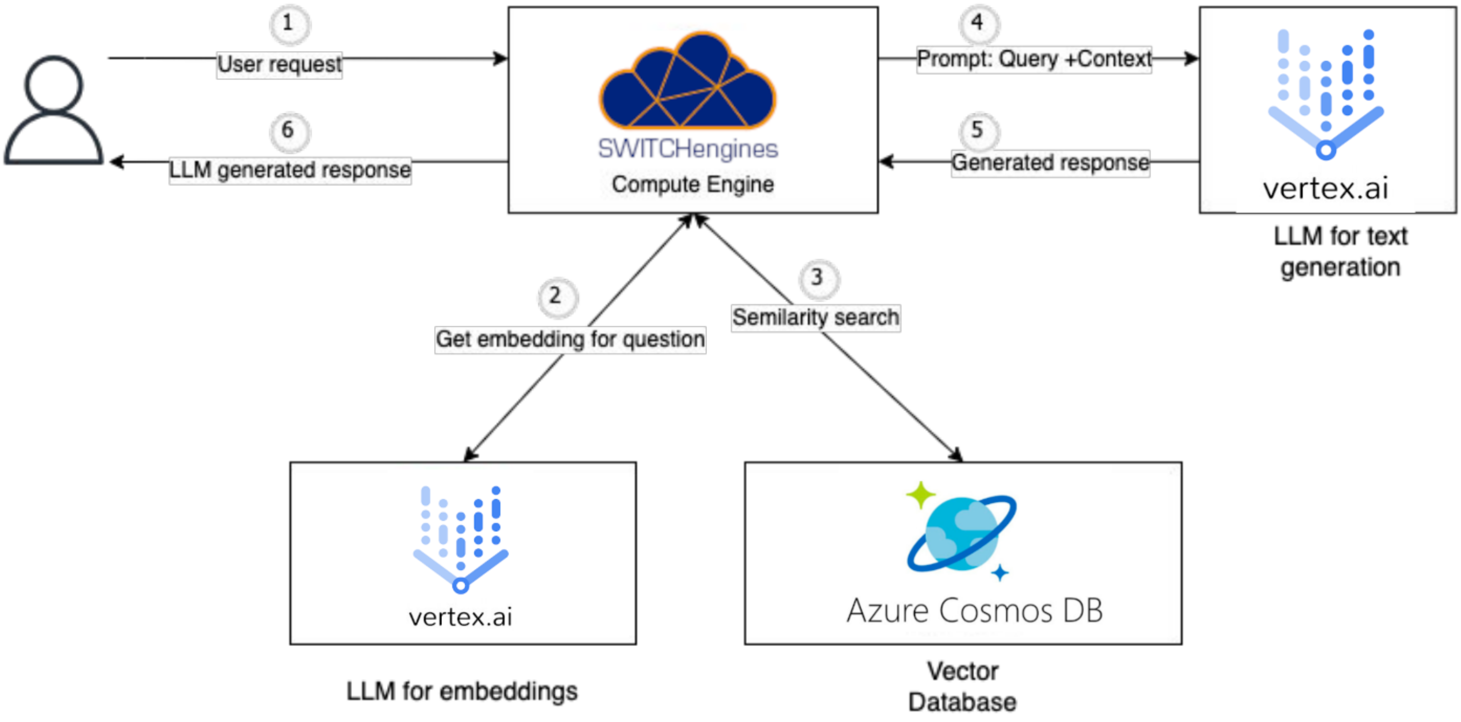
Goal

Using proprietary APIs of Cloud providers from a single online chatbot. Based on the slides schemas, we updated the 2 architecture diagrams.

First step: Setup an S3 bucket on OpenStack, be able to download documents, send them to Google Vertex AI to get their embeddings equivalent. Then, we want to store these floats vectors into an Azure Cosmos DB database.



Second step: Deploy a chatbot based on given code, that has a streamlit chat interface that allows to ask a question about the vectorized documents. First the user request the bot has received, is sent to Google Vertex AI again to a model called `gemini-embedding-001`, the returned embedding is used to query a similarity search on Azure Cosmos DB. We want to take the 10 documents that are the most relevant to the user's query. This context is then given again to Google Vertex AI (with another model `gemini-2.5-flash-lite`). The final textual answer from Gemini 2.5 is sent back in the streamlit chat.



Based on the code provided for the previous lab, we refactored the scripts for the new infrastructure, keeping mostly the same scripts separation strategy. The chatbot has the same strategy as the previous one but is using the new services. For ease of execution, we have chosen a few hard-coded values inside `config.ini` to avoid choosing and giving names. All values that cannot be chosen in advance (such as the public IP or the folder where you want to download PDFs), are provided via CLI flags.

Prerequisites

This assumes you have Python installed and you have access to the 3 following clouds: Google Cloud, Microsoft Azure and Switch Engines.

1. **Install Python dependencies**

   ```
   pip install -r requirements.txt
   ```

2. **For Azure**: Install the Azure CLI `az` and run `az login`.

3. **Setup the Google Cloud environment**

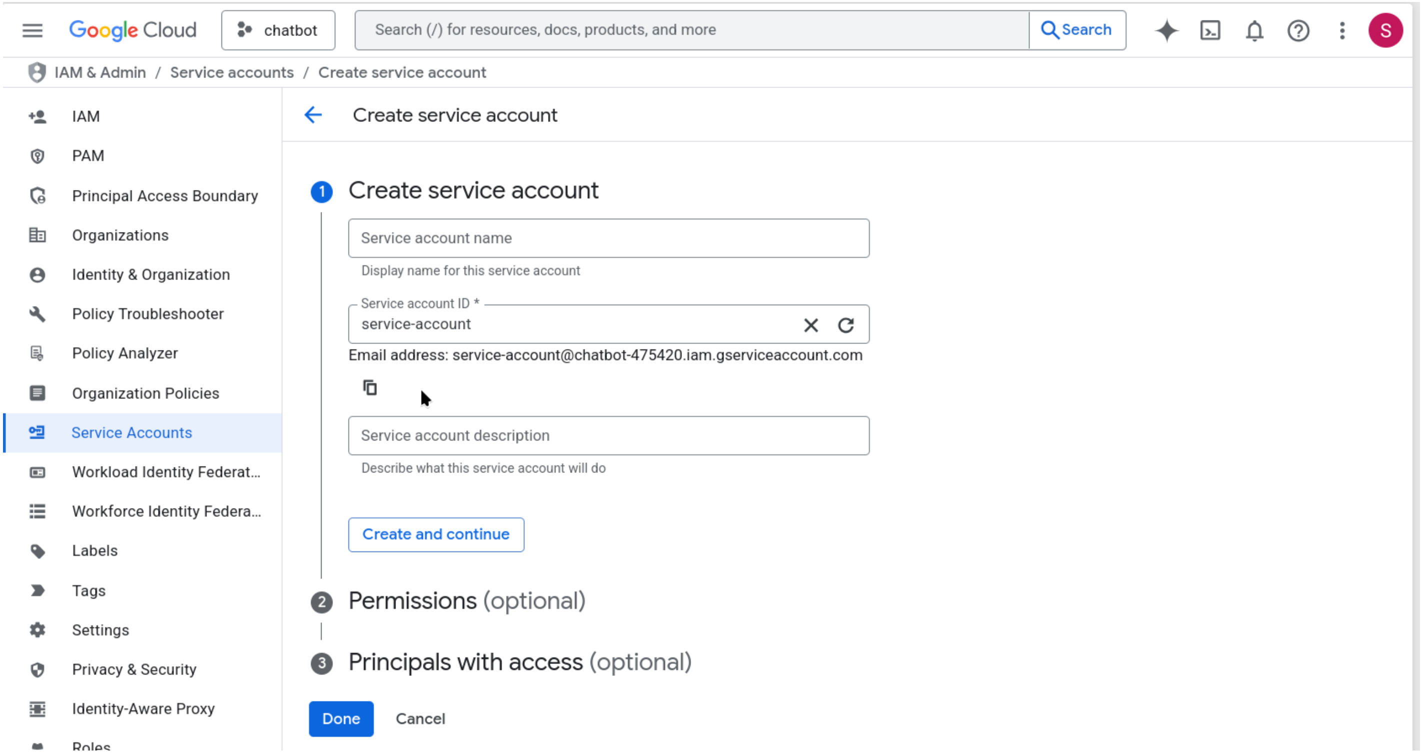
   i. Create manually a New project on GCloud named `chatbot` and get it's ID. For this example, we got `chatbot-475420`.

   ii. **Change your config.ini file with your ID under key `vertexai` > `project_id`**

   iii. Enable Google Vertex AI API here

   iv.

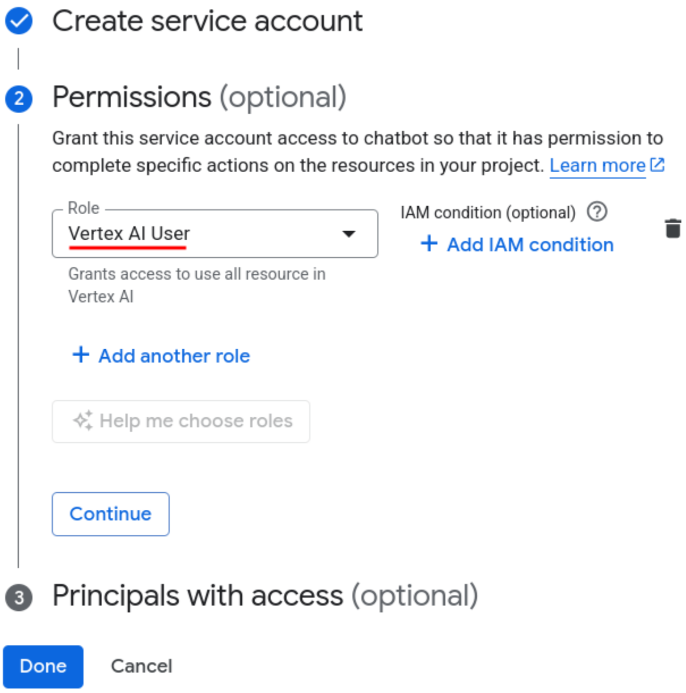
You have to create a Service account, go in IAM, and create one like on this picture


Google Cloud
chatbot
Search (/) for resources, docs, products, and more
Search
S
IAM & Admin / Service accounts / Create service account
IAM
PAM
Principal Access Boundary
Organizations
Identity & Organization
Policy Troubleshooter
Policy Analyzer
Organization Policies
Service Accounts
Workload Identity Federat...
Workforce Identity Federa...
Labels
Tags
Settings
Privacy & Security
Identity-Aware Proxy
Roles
Create service account
1 Create service account
Service account name
Display name for this service account
Service account description
Describe what this service account will do
Create and continue
2 Permissions (optional)
3 Principals with access (optional)
Done
Cancel

Service account ID *: service-account

Email address: service-account@chatbot-475420.iam.gserviceaccount.com

v. Add **Vertex AI User** role and Continue with defaults.

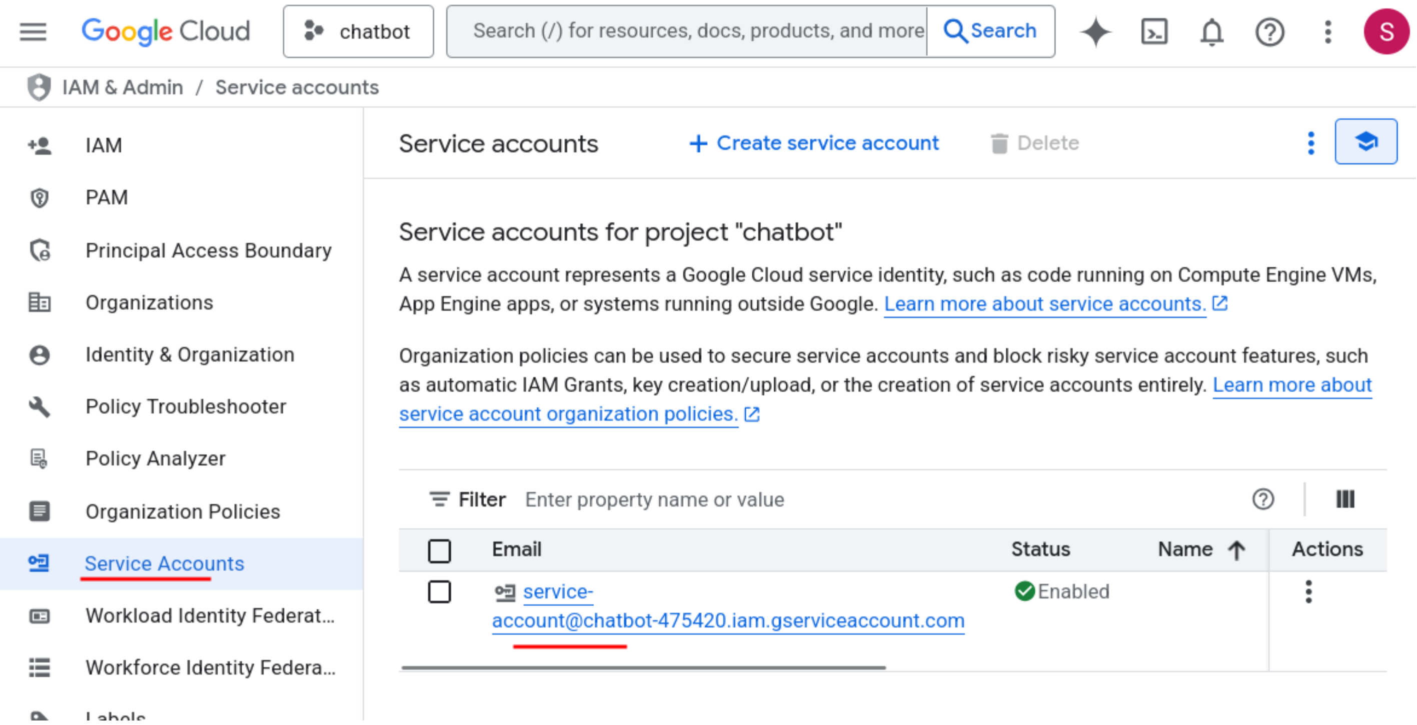


Grant this service account access to chatbot so that it has permission to complete specific actions on the resources in your project. Learn more

Role: Vertex AI User

Grants access to use all resource in Vertex AI

i. Now you have a service account, click on it



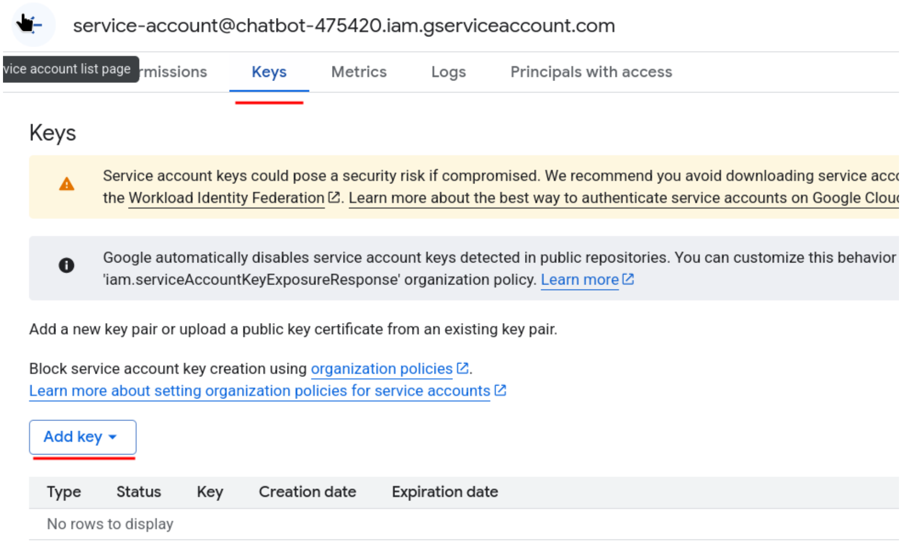
Service accounts for project "chatbot"

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. Learn more about service accounts.

Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload, or the creation of service accounts entirely. Learn more about service account organization policies.

□	Email	Status	Name ↑	Actions
□	service-account@chatbot-475420.iam.gserviceaccount.com	✓ Enabled		⋮

i. Go in the Keys tab to **Add key** > Create new key



Keys

Service account keys could pose a security risk if compromised. We recommend you avoid downloading service acc the Workload Identity Federation. Learn more about the best way to authenticate service accounts on Google Clou

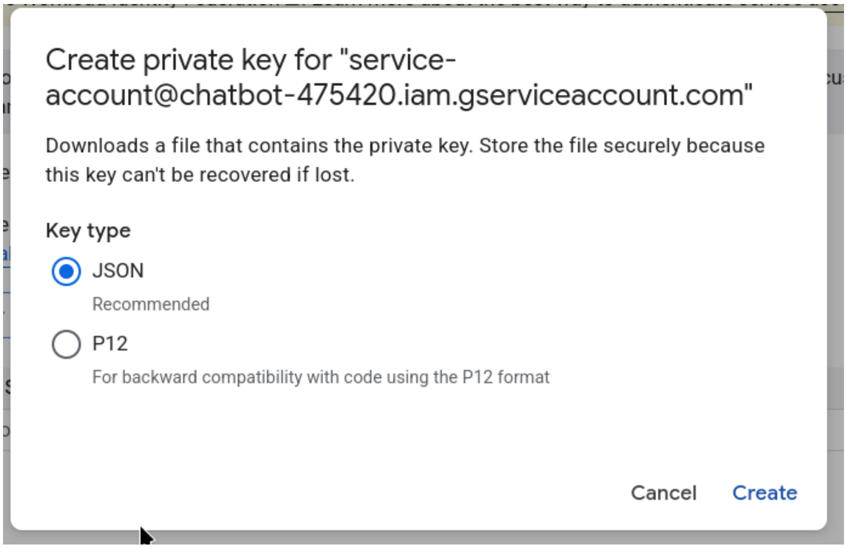
Google automatically disables service account keys detected in public repositories. You can customize this behavior 'iam.serviceAccountKeyExposureResponse' organization policy. Learn more

Add a new key pair or upload a public key certificate from an existing key pair.

Block service account key creation using organization policies.
Learn more about setting organization policies for service accounts

Type	Status	Key	Creation date	Expiration date
No rows to display				

i. Download as JSON format



Create private key for "service-account@chatbot-475420.iam.gserviceaccount.com"

Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.

Key type

● JSON
Recommended

○ P12
For backward compatibility with code using the P12 format

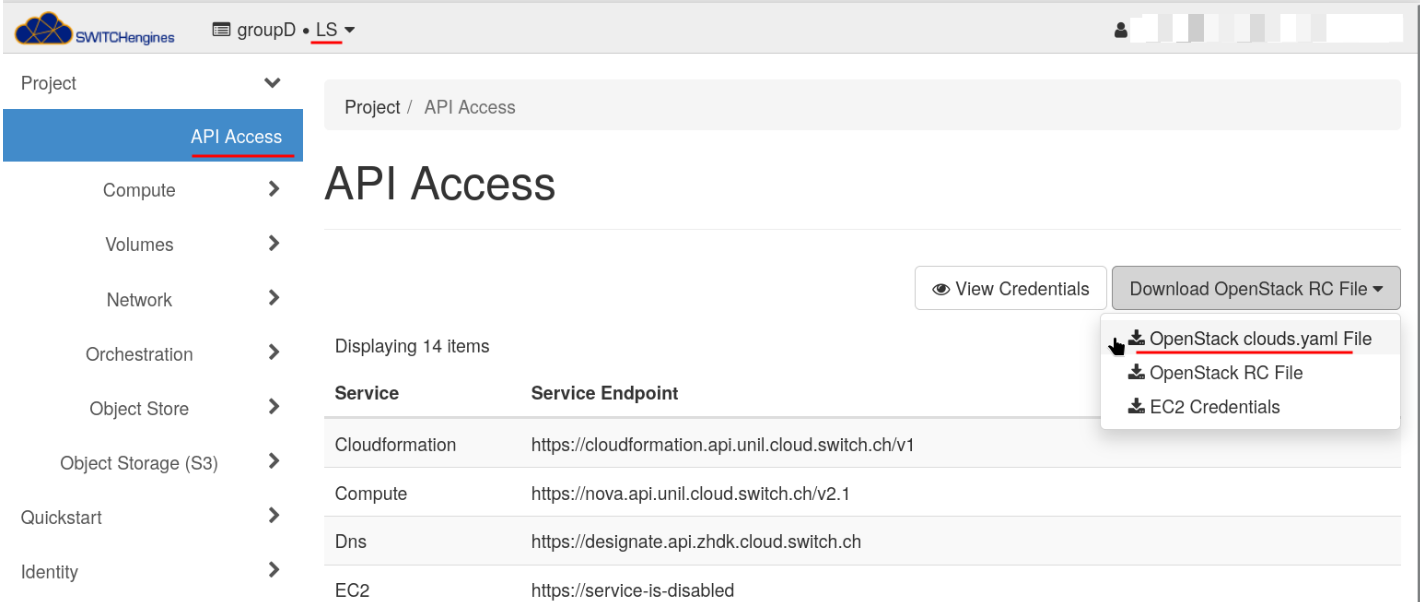
i. Place this JSON file under a file named `vertexai-service-account-key.json` in the current folder

4. **Setup the Switch Engines environment**

   i. Go login on Switch Engines panel to get your generated API password

ii. **Make sure you have the LS (Lausanne)** region chosen not the ZH one...

iii. [Go into the API access page to get your `clouds.yaml`]



API Access

Displaying 14 items

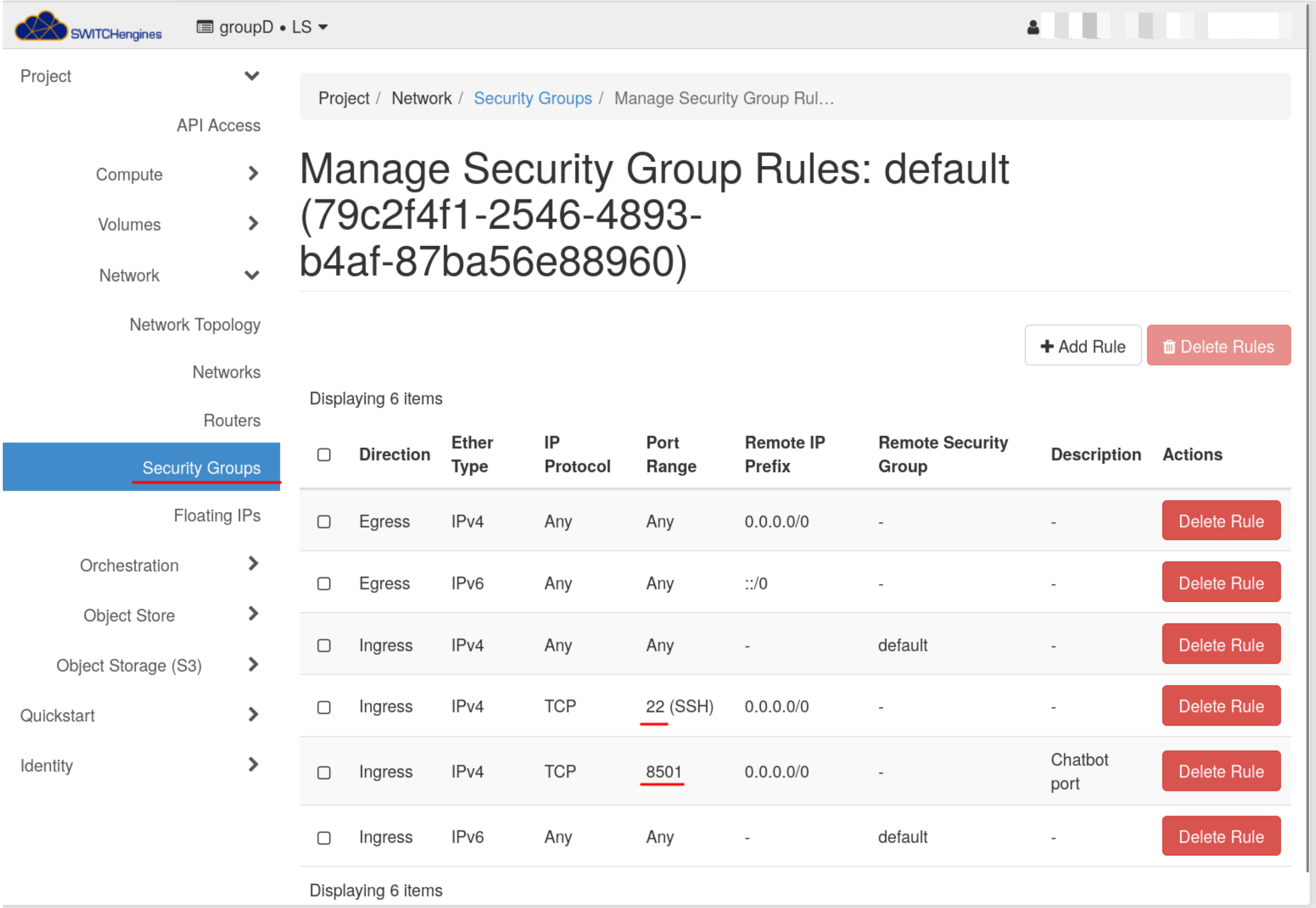
Service	Service Endpoint
Cloudformation	https://cloudformation.api.unil.cloud.switch.ch/v1
Compute	https://nova.api.unil.cloud.switch.ch/v2.1
Dns	https://designate.api.zhdk.cloud.switch.ch
EC2	https://service-is-disabled

iv. Move this file under a `switch` subfolder in this directory

v. Edit it to add your password

vi. An SSH keypair will be created automatically and downloaded under `switch/switchengine-tsm-cloudsys.pem` if it doesn't already exist.

vii. Change your `default` security group to accept port 22 (SSH) and 8501 (the chatbot).



Manage Security Group Rules: default (79c2f4f1-2546-4893-b4af-87ba56e88960)

Displaying 6 items

□	Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
□	Egress	IPv4	Any	Any	0.0.0.0/0	-	-	Delete Rule
□	Egress	IPv6	Any	Any	::/0	-	-	Delete Rule
□	Ingress	IPv4	Any	Any	-	default	-	Delete Rule
□	Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	-	Delete Rule
□	Ingress	IPv4	TCP	8501	0.0.0.0/0	-	Chatbot port	Delete Rule
□	Ingress	IPv6	Any	Any	-	default	-	Delete Rule

Displaying 6 items

S3-compatible container creation on Switch engines

We know this wasn't 100% recommended by John White but we tried and it works.

```
python manage-S3_switch.py --pdf_path ../../../TSM_CloudSys-2024-25.pdf
```

With this script, we create container in object store, upload an pdf. We can also download this pdf, list object storage and contents and delete a dedicated container.

Vector database creation on Azure Cosmos DB service

We are going to use **Azure Cosmos DB** for this part. The home page of this service is here: [Create an Azure Cosmos DB account](#).

Note: the account name is also used as the database name. This name must be unique and can only

1. Activate the Cosmos DB service for your Azure account

```
az provider register --namespace Microsoft.DocumentDB
```

It may take 1–2 minutes to complete. You can check the registration status with the command:

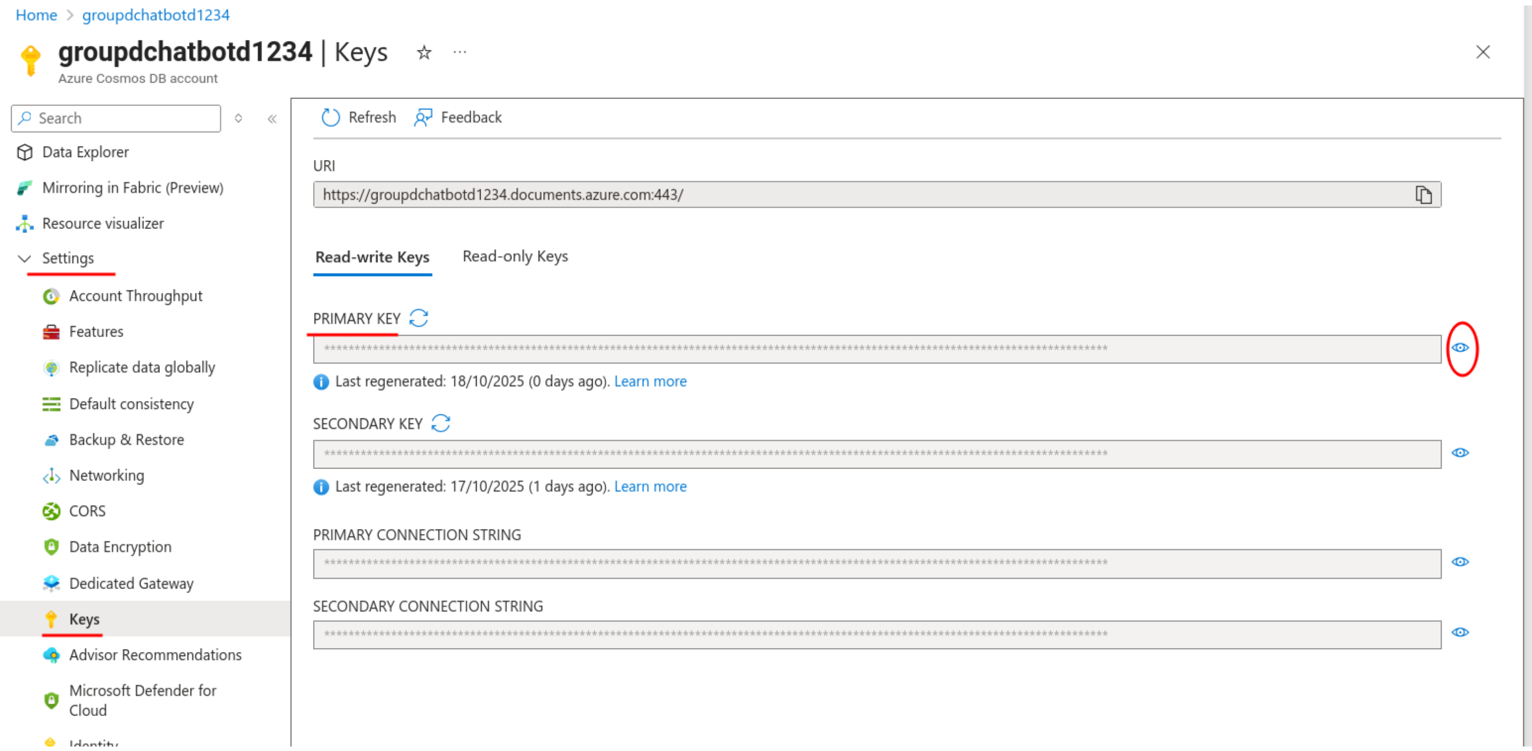
```
az provider show --namespace Microsoft.DocumentDB --query "registrationState"
```

The command should return `Registered`

2. Then you can run

```
> python setup-azure.py
Provisioned resource group groupd-chatbot-deploy
Created Cosmos DB account: groupdchatbotd1234
```

3. Get the primary key in your portal and save it under a `azure-db-key.txt` file in this folder. This will be used for authentication for next scripts.



Home > groupdchatbotd1234

groupdchatbotd1234 | Keys

Azure Cosmos DB account

URI

https://groupdchatbotd1234.documents.azure.com:443/

Read-write Keys | Read-only Keys

PRIMARY KEY

****

Last regenerated: 18/10/2025 (0 days ago). Learn more

SECONDARY KEY

****

Last regenerated: 17/10/2025 (1 days ago). Learn more

PRIMARY CONNECTION STRING

****

SECONDARY CONNECTION STRING

****

Vectorizing the PDF Files

We want to download the files in S3 again, ask Google Vertex AI to generate embeddings and store them in a container in a database in Cosmos DB.

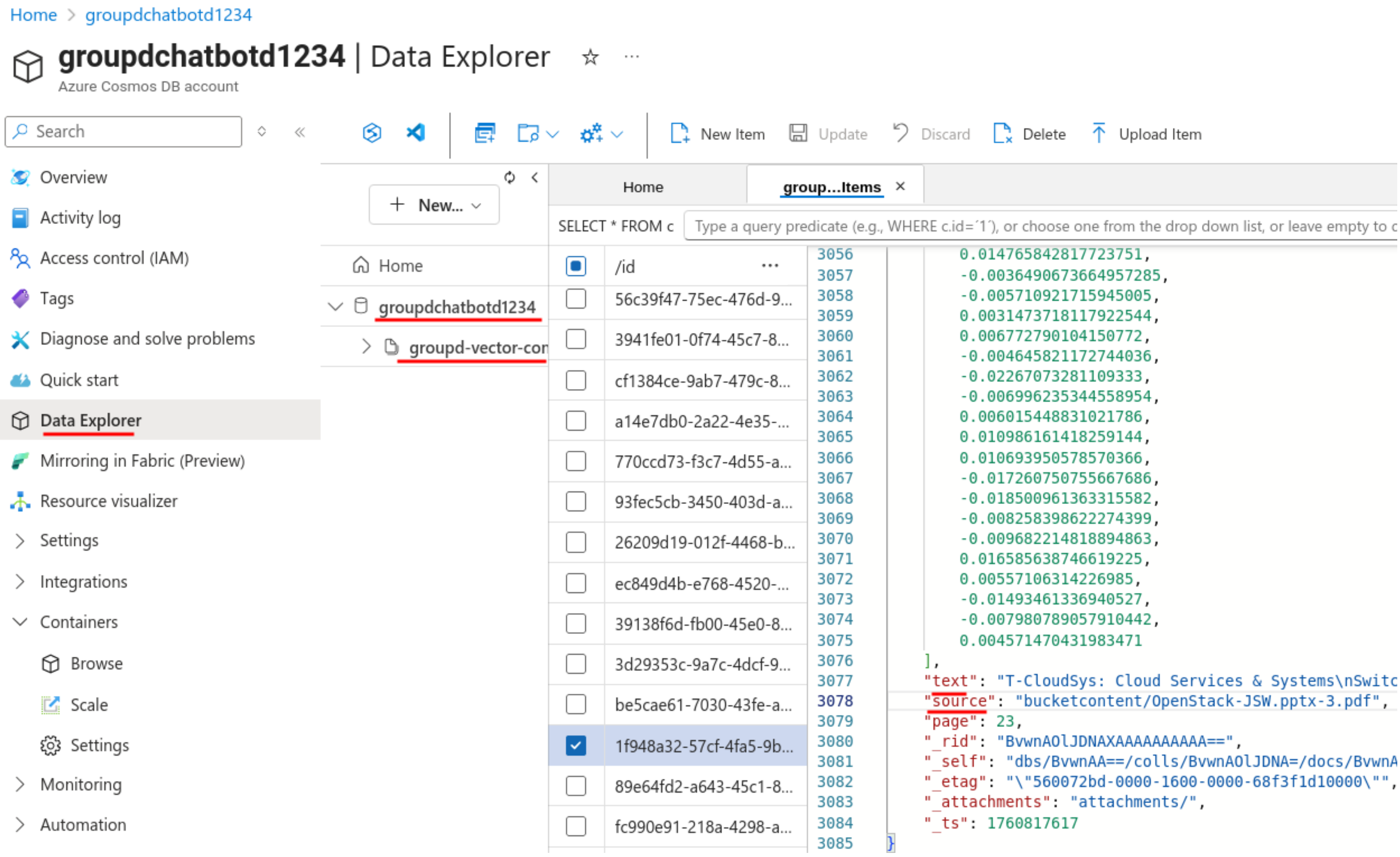
First download the pdf file from S3:

```
$ python manage-S3_switch.py --download
Download TSM_CloudSys-2024-25.pdf in ./s3-download/TSM_CloudSys-2024-25.pdf...
TSM_CloudSys-2024-25.pdf downloaded successfully.
```

And run the script `vectorise-store.py` which is an adaptation of the provided script in the previous lab.

```
python vectorise-store.py --local_path s3-download
```

You can verify the embeddings have been stored by looking at the Azure portal. You should see entries with chunks of the text, a giant float array, the text of the chunk and the source file.



56c39f47-75ec-476d-9...
3941fe01-0f74-45c7-8...
cf1384ce-9ab7-479c-8...
a14e7db0-2a22-4e35-...
770ccd73-f3c7-4d55-a...
93fec5cb-3450-403d-a...
26209d19-012f-4468-b...
ec849d4b-e768-4520-...
39138f6d-fb00-45e0-8...
3d29353c-9a7c-4dcf-9...
be5cae61-7030-43fe-a...
1f948a32-57cf-4fa5-9b...
89e64fd2-a643-45c1-8...
fc990e91-218a-4298-a...

```
        0.014765842817723751,
        -0.003649067366495728 5,
        -0.005710921715945005,
        0.003147371811792254 4,
        0.006772790104150772,
        -0.004645821172744036,
        -0.02267073281109333,
        -0.006996235344558954,
        0.006015448831021786,
        0.010986161418259144,
        0.010693950578570366,
        -0.017260750755667686,
        -0.018500961363315582,
        -0.008258398622274399,
        -0.009682214818894863,
        0.016585638746619225,
        0.00557106314226985,
        -0.014934613369405 27,
        -0.007980789057910442,
        0.004571470431983471
    ],
    "text": "T-CloudSys: Cloud Services & Systems\nSwitc
    "source": "bucketcontent/OpenStack-JSW.pptx-3.pdf",
    "page": 23,
    "_rid": "BvwnAOlJDNAXAAAAAAAAAA==",
    "_self": "dbs/BvwnAA==/colls/BvwnAOlJDNA=/docs/BvwnA
    "_etag": "\"560072bd-0000-1600-0000-68f3f1d10000\"",
    "_attachments": "attachments/",
    "_ts": 1760817617
}
```

Accessing the application locally

To make sure everything is working, there is a little `test.py` script that allows to quickly test the bot without open the streamlit chat. The message sent is hard-coded and can be changed if needed.

```
python test.py
```

If the AI is giving an answer and no Python stacktrace, it means everything is fully working. It's not possible to run the chatbot locally.

```
streamlit run chatbot.py
```

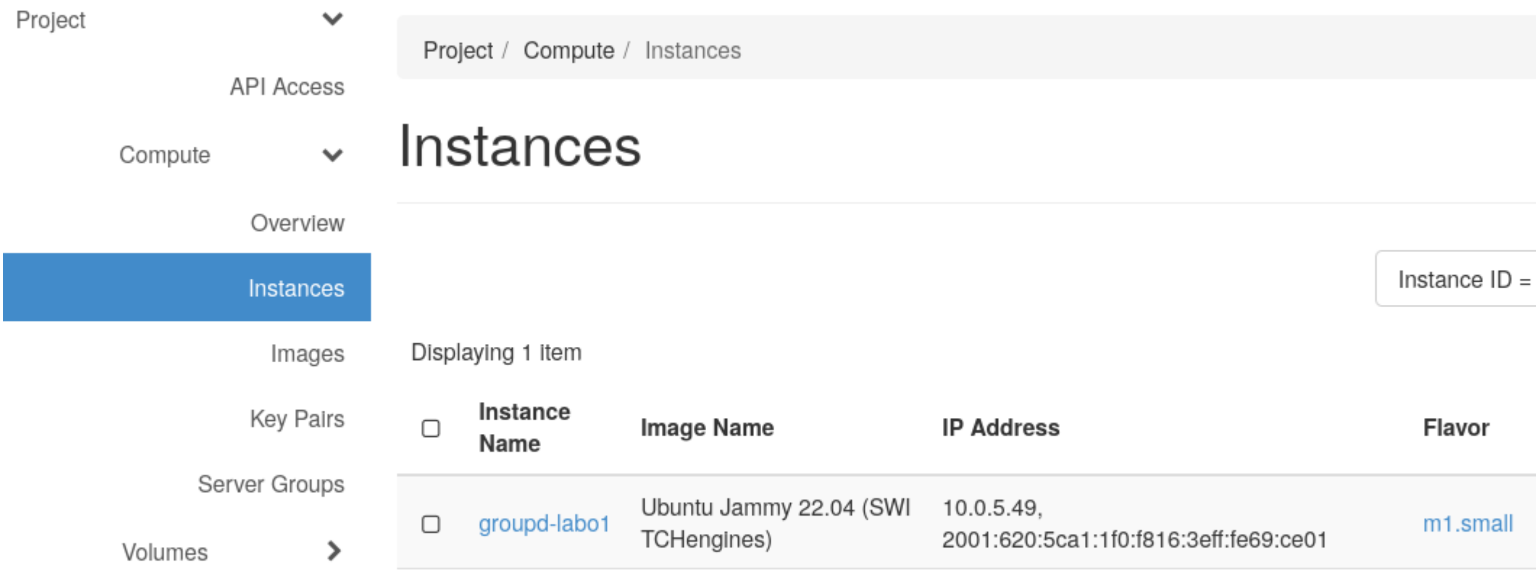
which should open your browser with a fully working bot. You can already ask questions about your vectorized PDFs.

Create the instance on Switch Engines

Make sure to fill the `config.ini` file !
These files will be used in the `manage_instance_switch.py` script, which you can simply run with

```
$ python manage_instance_switch.py --create
List Servers:
Create Server:

VM 'groupd-labo1' created.
You can login with SSH in a minute with
ssh -i ./switch/switchengine-tsm-cloudsys.pem ubuntu@86.119.31.138
```

Now an Ubuntu 22.04 instance of flavour `m1.small` should be running and accessible. It take 1-2 minutes to be accessible via SSH.



Displaying 1 item

□	Instance Name	Image Name	IP Address	Flavor
□	groupd-labo1	Ubuntu Jammy 22.04 (SWITCHengines)	10.0.5.49, 2001:620:5ca1:1f0:f816:3eff:fe69:ce01	m1.small

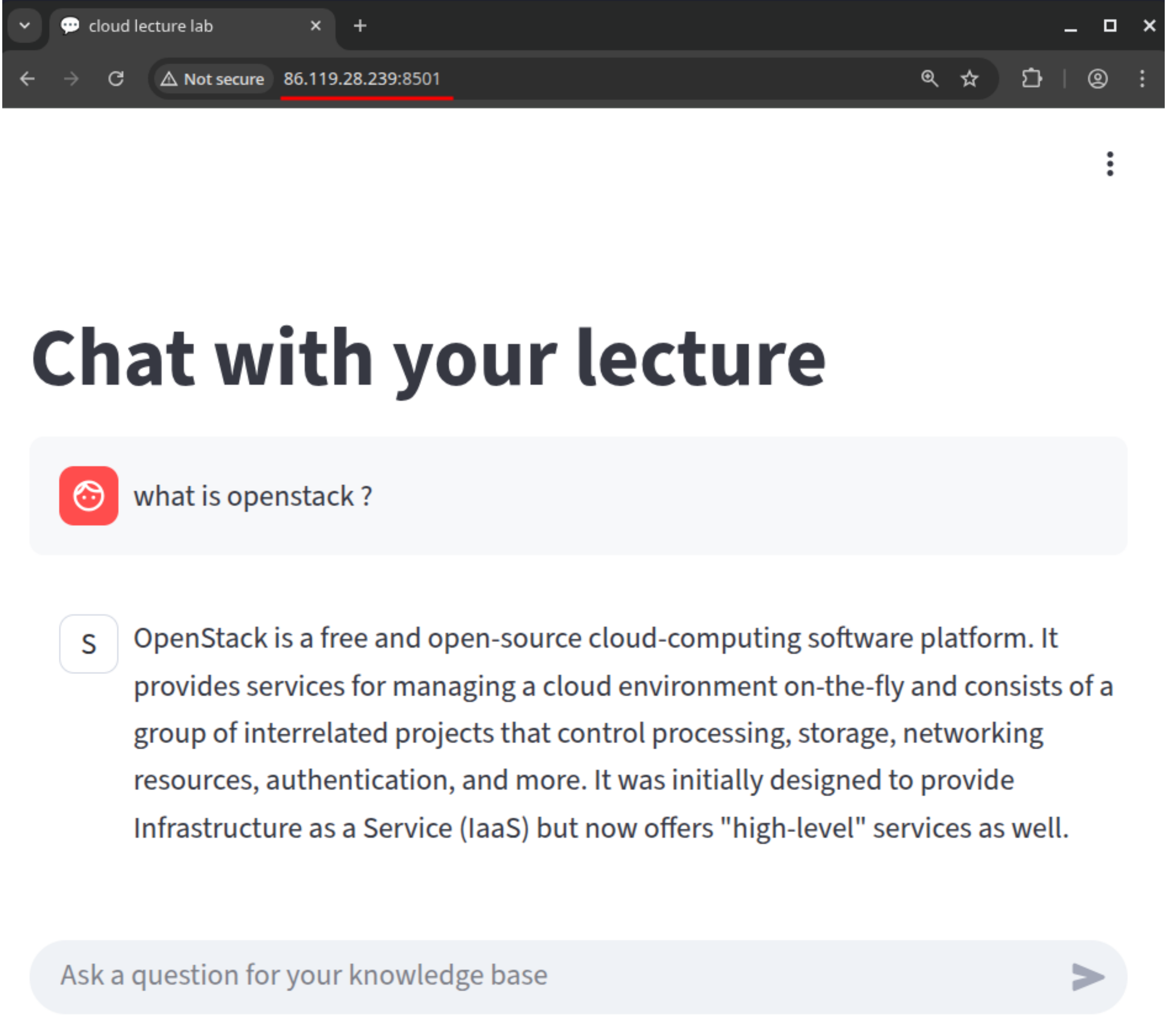
Deploy the chatbot on the VM

The VM is ready to host the chatbot. Get the hostname showed previously and run it like that

```
$ python deploy.py --host 86.119.31.138

Uploading current folder into the VM
Starting deploy.sh on the VM

Hit:1 http://ch.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://ch.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
...
Defaulting to user installation because normal site-packages is not writeable
Collecting openstacksdk
  Downloading openstacksdk-4.7.1-py3-none-any.whl (1.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 10.3 MB/s eta 0:00:00
Collecting azure-cosmos
  Downloading azure_cosmos-4.14.0-py3-none-any.whl (385 kB)

...

CHATBOT HAS BEEN DEPLOYED...
Please open http://86.119.31.138:8501 in your Web browser !
```

It installs Python, Pip, install the Pip dependencies, and start the chatbot in background.

You can finally open the link printed on the last line.



Chat with your lecture

what is openstack ?

OpenStack is a free and open-source cloud-computing software platform. It provides services for managing a cloud environment on-the-fly and consists of a group of interrelated projects that control processing, storage, networking resources, authentication, and more. It was initially designed to provide Infrastructure as a Service (IaaS) but now offers "high-level" services as well.

Delete the infrastructure

At the end of the lab, when you need to delete the Azure infrastructure, run this script (this can take several minutes to delete the resource group)

```
> python delete-azure.py
Cosmos DB account 'groupdchatbotdb1234' deleted successfully.
Resource group 'groupd-chatbot-deploy' deleted successfully.
```

You can delete the Switch VM and associated resource, with this command. Make sure to give the correct VM name.

```
> python manage_instance_switch.py --delete-vm groupd-labo1
List Servers:
groupd-labo1 - ACTIVE - fcef1b69-83a8-4173-a399-8dcbd3758bb3
Server 'groupd-labo1' deleted
Floating IP 86.119.31.63 released
```

You can delete the S3 from Switch engine

```
> python manage-S3_switch.py --delete
Deleted object: TSM_CloudSys-2024-25.pdf
Container 'groupd' deleted successfully.
```

The Google Cloud project `chatbot` could be deleted manually.