

Importing ONNX Embedding Models

Oracle AI Database 26ai using OML4Py Client in OCI VM

This guide is intended for experimentation and demo purposes. Please refer to the referenced documentation below for best practises and latest information.

The OML4Py Client can also be installed on-premises.

Description

Oracle AI Database 26ai includes an ONNX runtime engine for running embedding models directly inside the database. This section covers the process of setting up the OML4Py Client in an OCI and importing existing pretrained embedding models into Oracle AI Database, including utilising pre-configured models and external models from HuggingFace (subject to compatibility) by converting those models into the ONNX format if they are not already converted.

The OML4Py Client provides the Python package to:

- Download 30+ pre-configured embedding models along with pre-trained embedding models from Hugging Face into your Oracle AI Database.
- Augments the model with pre-processing and post-processing steps and creates a new ONNX model
- Validates the augmented ONNX model
- Loads into the database as a data mining model or optionally exports to a file

If you would like to understand the supported embedding models that can be loaded into Oracle's AI Database across Text, Image and Multi-Modal Embedding, please reference the below documentation.

- Text Embedding Models - <https://docs.oracle.com/en/database/oracle/machine-learning/oml4py/2-23ai/mlpug/onnx-pipeline-models-text-embedding.html>
- Image Embedding Models - <https://docs.oracle.com/en/database/oracle/machine-learning/oml4py/2-23ai/mlpug/onnx-pipeline-models-image-embedding.html>
- Multi-Modal Embedding Models - <https://docs.oracle.com/en/database/oracle/machine-learning/oml4py/2-23ai/mlpug/onnx-pipeline-models-multi-modal-embedding.html>

Assumed Pre-Requisites

- You have the ability to create Networking and Compute resources in OCI to host the OML4Py Client.
- You have already provisioned an Oracle Autonomous AI Database 26ai
 - Downloaded the Wallet File
 - Have a DB User Created with necessary grants
 - GRANT CREATE MINING MODEL TO <USER>;

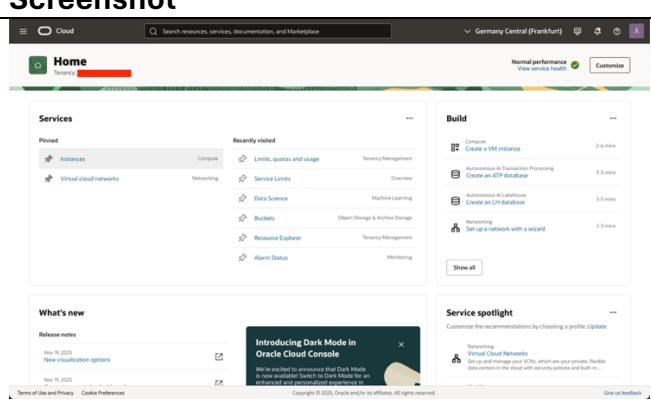
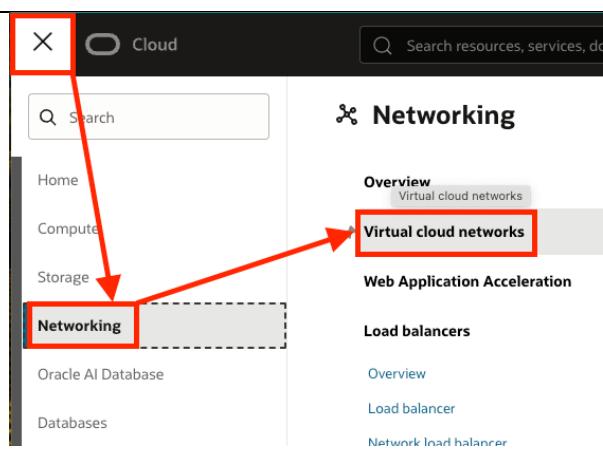
Please Note

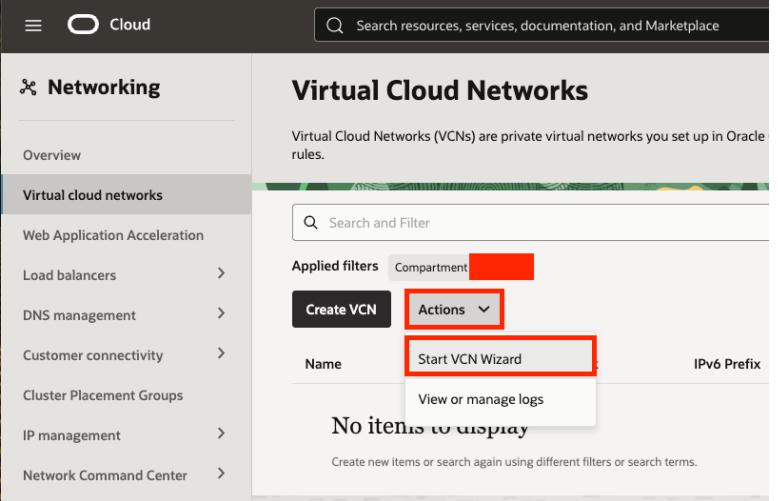
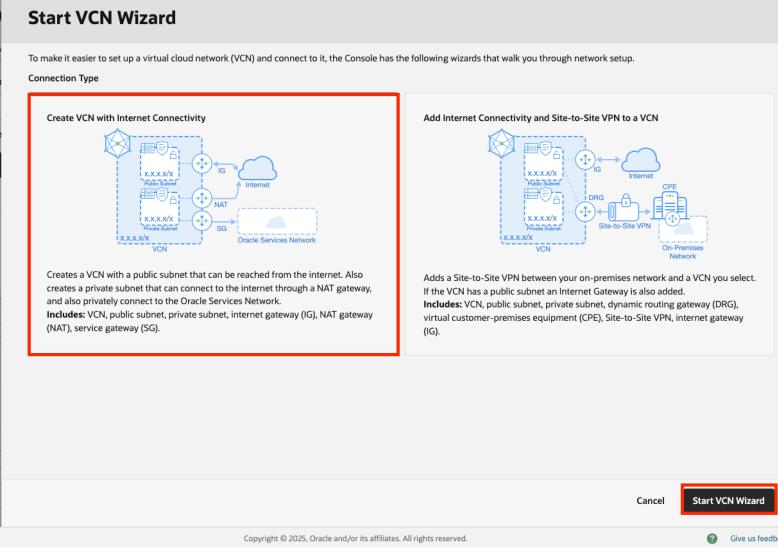
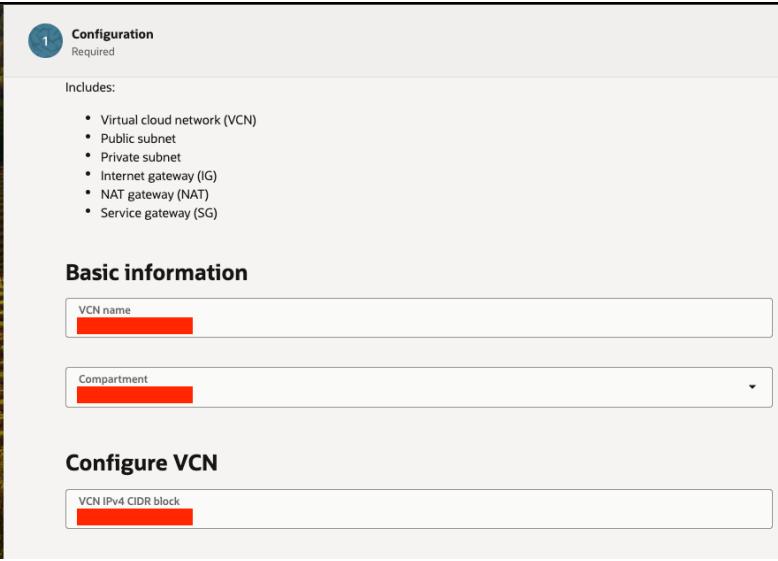
- This feature will only work on OML4Py 2.1 client. It is not supported on the OML4Py server or older versions of the OML4Py client.
- In-database embedding models must include tokenization and post-processing. Providing only the core ONNX model to DBMS_VECTOR.LOAD_ONNX_MODEL is insufficient because you need to handle tokenization externally, pass tensors into the SQL operator, and convert output tensors into vectors.
- There are limitations on the size and type of embedding models that can be loaded into the Oracle AI Database. Please refer to the documentation mentioned above which will provide links to the types of models that can be used along with their restrictions.

Referenced Documentation

- Overview - <https://docs.oracle.com/en/database/oracle/machine-learning/oml4py/2-23ai/mlpug/import-pretrained-models-onnx-format-vector-generation-database.html>
- End to End Installation - <https://docs.oracle.com/en/database/oracle/machine-learning/oml4py/2-23ai/mlpug/convert-pretrained-models-onnx-model-end-end-instructions.html>

OML4Py 2.1 Client Installation in OCI VM

Step	Screenshot
Login to your OCI Console. https://www.cloud.oracle.com	
First, we will create the VCN for our Compute Instance to live within. Navigate to OCI Menu > Networking > Virtual Cloud Networks	

<p>We will create a basic VCN and Subnet setup with Internet access using the Wizard.</p> <p>Select Actions > Start VCN Wizard</p> <p>If you prefer to create your own custom Network setup from scratch, select Create VCN.</p>	
<p>Select Create VCN with Internet Connectivity.</p> <p>Select Start VCN Wizard.</p>	
<p>Enter VCN Name.</p> <p>Select Compartment for your VCN to live in.</p> <p>I used the default VCN CIDR Block Range. Feel free to modify to meet the networking requirements of your organisation.</p>	

I accepted the **default IP CIDR Block Ranges for the Public and Private Subnets** that will be created.

Feel free to modify to meet the networking requirements of your organisation.

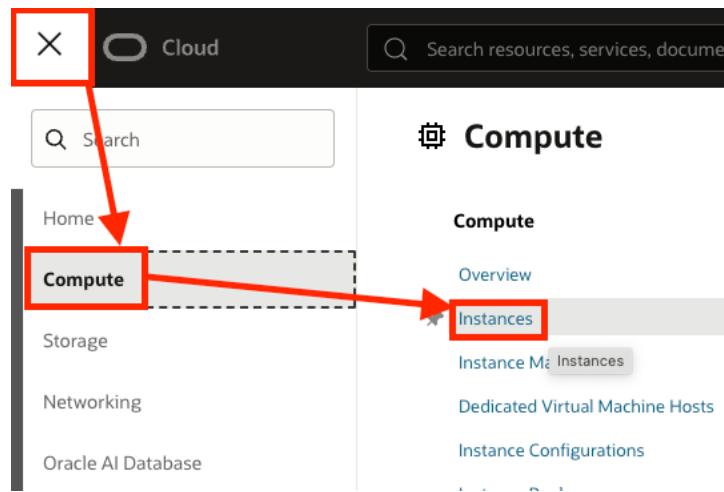
Click **Next**.

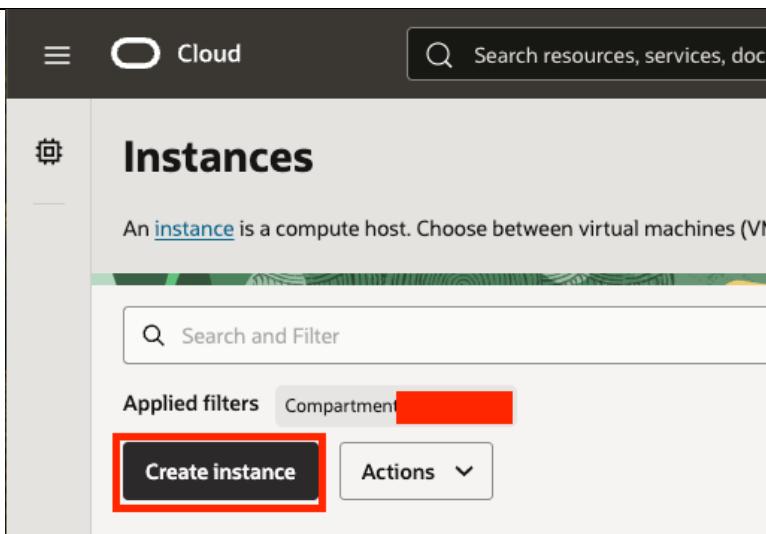
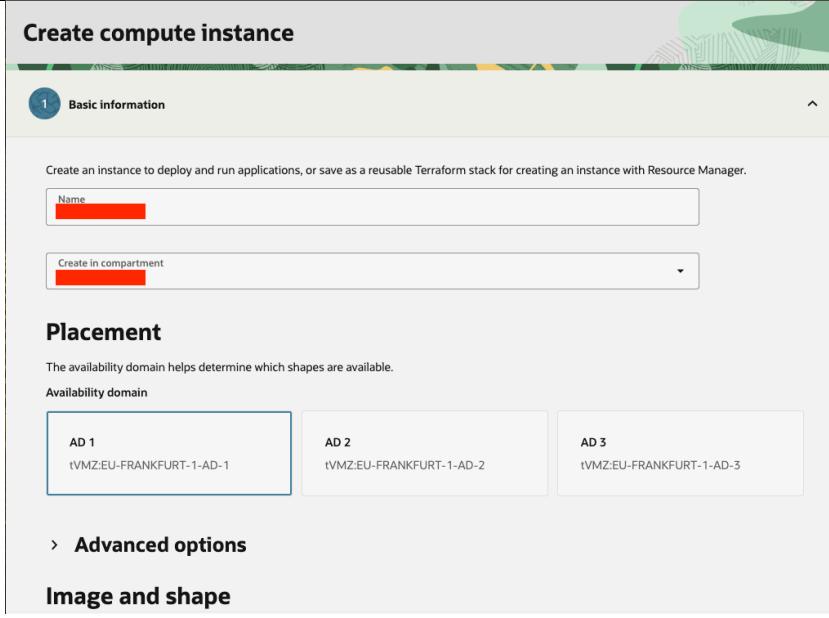
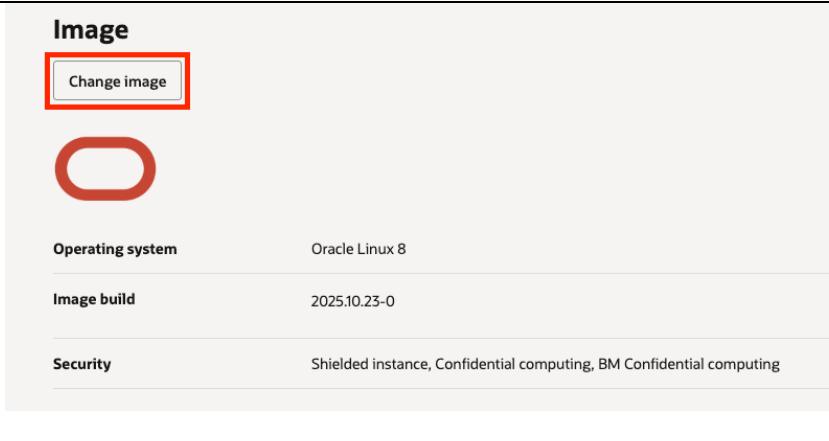
Review the configurations for your VCN setup.

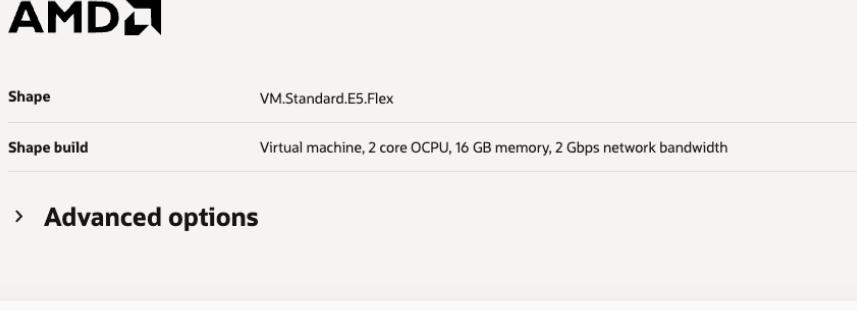
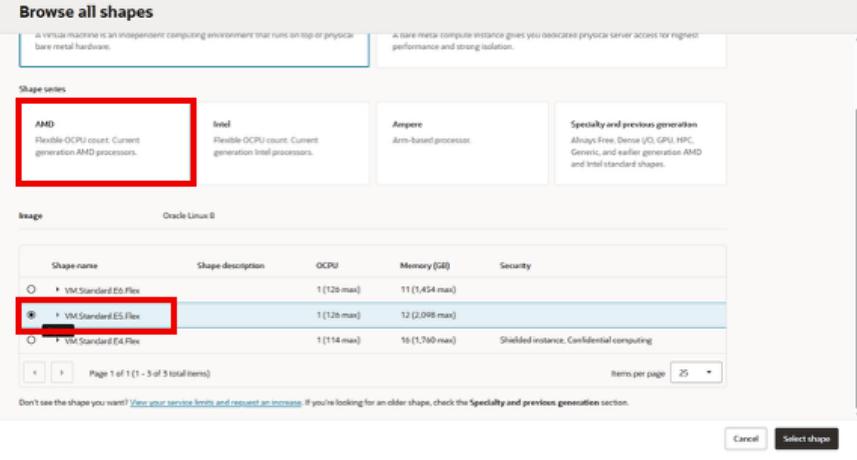
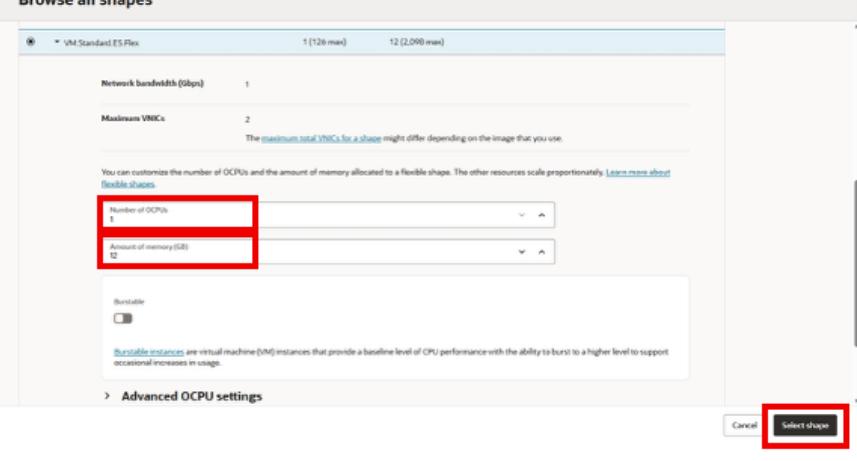
Click **Create**.

Now we will create the Compute Instance to host the OML4Py Client which is responsible for downloading embedding models to be used in database.

Navigate to **OCI Menu > Compute > Instances**.



	
<p>Click Create Instance.</p> <p>Enter a Name for your Instance.</p> <p>Select the Compartment for the Instance to live in.</p> <p>I kept the Availability Domain as Default.</p>	
<p>Click on Change Image to select the OS of the Instance.</p> <p>I selected Oracle Linux 8 which supports the OML4Py Client.</p>	

<p>Click on <i>Change Shape</i>.</p> <p>I will select a shape of:</p> <ul style="list-style-type: none"> • VM.Standard.E5. Flex • 2 OCPUs and 16GB Memory 	<p>Shape</p> <p>Change shape</p>  <p>Shape: VM.Standard.E5.Flex</p> <p>Shape build: Virtual machine, 2 core OCPU, 16 GB memory, 2 Gbps network bandwidth</p> <p>> Advanced options</p>																				
<p>Select the Shape Series as AMD.</p> <p>Click on the drop down next to the VM.Standard.E5.Flex Shape.</p>	 <p>Browse all shapes</p> <p>Shape series: AMD</p> <table border="1"> <thead> <tr> <th>Shape name</th> <th>Shape description</th> <th>OCPU</th> <th>Memory (GB)</th> <th>Security</th> </tr> </thead> <tbody> <tr> <td>VM.Standard.E5.Flex</td> <td>Flexible OCPU count: Current generation AMD processors.</td> <td>1 (128 max)</td> <td>11 (1,454 max)</td> <td></td> </tr> <tr> <td>VM.Standard.E5.Flex</td> <td>Flexible OCPU count: Current generation Intel processors.</td> <td>1 (128 max)</td> <td>12 (2,096 max)</td> <td></td> </tr> <tr> <td>VM.Standard.E4.Flex</td> <td>Flexible OCPU count: Previous generation Intel processors.</td> <td>1 (114 max)</td> <td>16 (1,760 max)</td> <td>Shielded instance, Confidential computing</td> </tr> </tbody> </table> <p>Image: Oracle Linux 8</p> <p>Don't see the shape you want? View our service limits and request an increase. If you're looking for an older shape, check the Specialty and previous generation section.</p> <p>Select shape</p>	Shape name	Shape description	OCPU	Memory (GB)	Security	VM.Standard.E5.Flex	Flexible OCPU count: Current generation AMD processors.	1 (128 max)	11 (1,454 max)		VM.Standard.E5.Flex	Flexible OCPU count: Current generation Intel processors.	1 (128 max)	12 (2,096 max)		VM.Standard.E4.Flex	Flexible OCPU count: Previous generation Intel processors.	1 (114 max)	16 (1,760 max)	Shielded instance, Confidential computing
Shape name	Shape description	OCPU	Memory (GB)	Security																	
VM.Standard.E5.Flex	Flexible OCPU count: Current generation AMD processors.	1 (128 max)	11 (1,454 max)																		
VM.Standard.E5.Flex	Flexible OCPU count: Current generation Intel processors.	1 (128 max)	12 (2,096 max)																		
VM.Standard.E4.Flex	Flexible OCPU count: Previous generation Intel processors.	1 (114 max)	16 (1,760 max)	Shielded instance, Confidential computing																	
<p>Set your number of OCPUs and Memory.</p> <p>I selected a Compute Shape of:</p> <ul style="list-style-type: none"> • 2 OCPUs and 16GB Memory <p>Click Select Shape.</p> <p>Click Next.</p>	 <p>Burstable</p> <p>Number of OCPUs: 2</p> <p>Amount of memory (GB): 12</p> <p>Select shape</p>																				

<p>I left all the Security options as Default.</p> <p>Click Next.</p>	
<p>Select Primary Network as Select existing virtual cloud network.</p> <p>Select the VCN we created earlier in our Compartment.</p> <p>Select Subnet as Select existing subnet.</p> <p>Select the Subnet we created earlier in our Compartment.</p> <p>Click Next.</p>	
<p>Select Automatically assign private IPv4 address.</p> <p>Ensure the toggle for Automatically assign public IPv4 address is selected.</p>	

We will generate new SSH Keys to login to our instance.

Select **Generate a key pair for me**.

Download the Private and Public Key.

Click **Next**.

I have left the Boot Volume and Block Volume settings as **defaults**.

Click **Next**.

View the summary of the instance creation.

Click **Create**.

This will take a couple minutes to provision.

Add SSH keys

Generate an [SSH key pair](#) to connect to the instance using a Secure Shell (SSH) connection, or upload a public key that you already have.

Generate a key pair for me

Upload public key file (.pub)

Paste public key

No SSH keys

Download the private key so that you can connect to the instance using SSH. It will not be shown again.

[Download private key](#) [Download public key](#)

Tasks Completed 2 of 4

Cancel [View estimated cost](#) [Previous](#) **Next**

Storage Required

Boot volume

A [boot volume](#) is a detachable device that contains the image used to boot the compute instance.

Specify a custom boot volume size and performance setting

[Volume performance](#) varies with volume size. Default boot volume size: 46.6 GB. When you specify a custom boot volume size, service limits apply.

Use in-transit encryption

[Encrypts data](#) in transit between the instance, the boot volume, and the block volumes.

Encrypt this volume with a key that you manage

By default, Oracle manages the keys that encrypt this volume, but you can choose a key from a vault that you have access to if you want greater control over the key's lifecycle and how it's used. [How do I manage my own encryption keys?](#)

Block volumes

[Attach block volume](#)

Name	Attachment type	Create type	Reservations	Access
No items to display				

Create new items or search again using different filters or search terms.

Tasks Completed 3 of 4

Cancel [View estimated cost](#) [Previous](#) **Next**

Basic information

[Edit](#)

Name

Create in compartment

Placement

Availability domain AD-1

Capacity type on-demand

Fault domain Let Oracle choose the best fault domain

Image

Operating system Oracle Linux 8

Image build 2025.10.23-0

Security

Shielded instance, Confidential computing, BM Confidential computing

Tasks Completed 4 of 4

Cancel [Save as stack](#) [View estimated cost](#) [Previous](#) **Create**

Once provisioned, navigate to the **Details Tab** and locate your **Public IP Address**.

Make a note of it, we will need this soon.

The screenshot shows the Oracle Cloud Infrastructure Instances Details page. The 'General information' section includes fields like Availability domain (AD-1), Fault domain (FD-1), Region (eu-frankfurt-1), OCID, Launched (Dec 01, 2025, 10:57:44 UTC), Compartment, and Capacity type (On-demand). The 'Image details' section shows Oracle Linux 8. The 'Launch options' section includes NIC attachment type (PARAVIRTUALIZED), Remote data volume (PARAVIRTUALIZED), Firmware (UEFI_64), Boot volume type (PARAVIRTUALIZED), and In-transit encryption (Enabled). The 'Instance access' section contains a note about connecting via SSH and shows the Public IP address, which is highlighted with a red box and has a red arrow pointing to it from the text above.

If on a Mac/Linux machine open a Terminal, navigate to your Downloads folder and change the permission on your Private Key file we downloaded earlier.

chmod 600 private-key-file.key

A screenshot of a Mac OS X terminal window titled '.ssh -- -bash -- 80x24'. The window shows three lines of text: '(base)' followed by two redacted lines, and then '\$ chmod 600 oml4py2.1-client.key'.

If using a Windows:

- 1:- Right-click on the target file and select properties then select Security Tab
- 2:- Click Advanced and then make sure inheritance is disabled.
- 3:- Click apply and then click Edit in the security menu
- 4:- Remove all users except yours/admin user, which should have full control. Admin account should have all checkboxes checked on Allow column except special permission.
- 5:- Click Apply and then click OK.

On a Mac/Linux Machine, within the Terminal, SSH into our Compute Instance.

```
ssh -i <private-key-file.key> opc@<public-ip-address>
```

The above assumes you are in the same directory as your private key file, otherwise specify the full path.

If prompted to continue, type **yes**.

```
(base) [REDACTED]$ ssh -i oml4py2.1-client.key opc@[REDACTED]
The authenticity of host '[REDACTED]' can't be established.
ED25519 key fingerprint is [REDACTED].
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[REDACTED]' to the list of known hosts
.
Activate the web console with: systemctl enable --now cockpit.socket
[REDACTED]~]$ pwd
```

If using Windows, you can use your favourite SSH tool such as PuTTY or Terminus to SSH into your compute instance in the same way.

Here is a YouTube video on how use PuTTY on Windows - <https://www.youtube.com/watch?v=4jakCV5JYx0>

Once logged in, let's download Python 3.12.6.

```
wget
https://www.python.org/ftp/python/3.12.6/Python-3.12.6.tgz
```

```
[REDACTED]$ wget https://www.python.org/ftp/python/3.12.6/Python-3.12.6.tgz
```

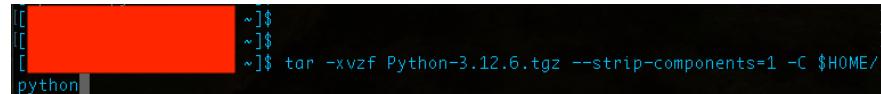
Create a new directory to install Python.

```
mkdir -p
$HOME/python
```

```
[REDACTED]~]$ mkdir -p $HOME/python
[REDACTED]~]$
```

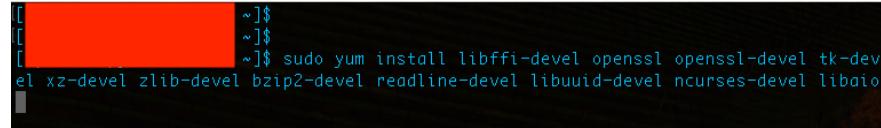
Unzip the Python installation into our new directory.

```
tar -xvzf Python-  
3.12.6.tgz --  
strip-  
components=1 -C  
$HOME/python
```



Install the additional libraries to the Linux Machine.

```
sudo yum install  
libffi-devel  
openssl openssl-  
devel tk-devel  
xz-devel zlib-  
devel bzip2-devel  
readline-devel  
libuuid-devel  
ncurses-devel  
libaio
```



Navigate into the Python Installation Directory.

```
cd $HOME/python
```



Configure the installation.

```
./configure --  
enable-shared --  
prefix=$HOME/pyth  
on
```



Make installation.	<pre>python]\$ python]\$ python]\$ make clean; make</pre>
Make installation.	<pre>python]\$ python]\$ python]\$ make altinstall</pre>
Set environment variables.	<pre>export PYTHONHOME=\$HOME/ python export PATH=\$PYTHONHOME/ bin:\$PATH export LD_LIBRARY_PATH=\$ PYTHONHOME/lib:\$L D_LIBRARY_PATH</pre> <pre>python]\$ python]\$ python]\$ export PYTHONHOME=\$HOME/python python]\$ export PATH=\$PYTHONHOME/bin:\$PATH python]\$ export LD_LIBRARY_PATH=\$PYTHONHOME/lib:\$LD_LIBRARY_PATH python]\$</pre>
Navigate to the Python Bin directory and create symbolic links.	<pre>cd \$HOME/python/bin ln -s python3.12 python3 ln -s pip3.12 pip3</pre> <pre>python]\$ python]\$ python]\$ cd \$HOME/python/bin bin]\$ ln -s python3.12 python3 bin]\$ ln -s pip3.12 pip3 bin]\$</pre>

We will now set up the Oracle Instant Client so that we can save downloaded embedding models directly to our Autonomous AI Database.

Navigate to home directory and download the Oracle Instant Client.

```
cd $HOME
```

```
wget https://download.oracle.com/otn_software/linux/installclient/2340000/instantclient-basic-linux.x64-23.4.0.24.05.zip
```

Unzip the Oracle
Instant Client
Download.

```
unzip  
instantclient-  
basic-linux.x64-  
23.4.0.24.05.zip
```

Set the LD Library Path.

```
export
LD_LIBRARY_PATH=$
HOME/instantclien
t_23_4:$LD_LIBRAR
Y_PATH
```

To ensure that the environment variables get set every time we start the compute instance, lets add them to the .bashrc file.

```
cd ~
```

```
vi .bashrc
```

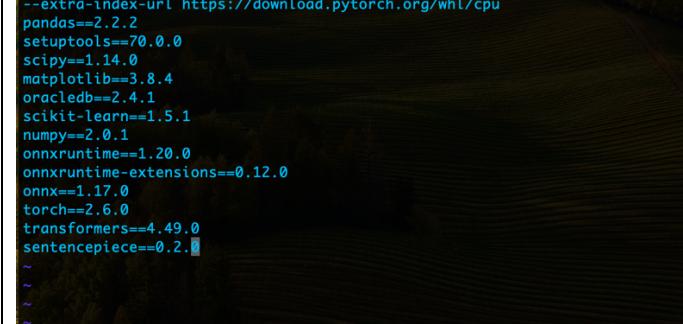
Press **i** on your keyboard to insert text.

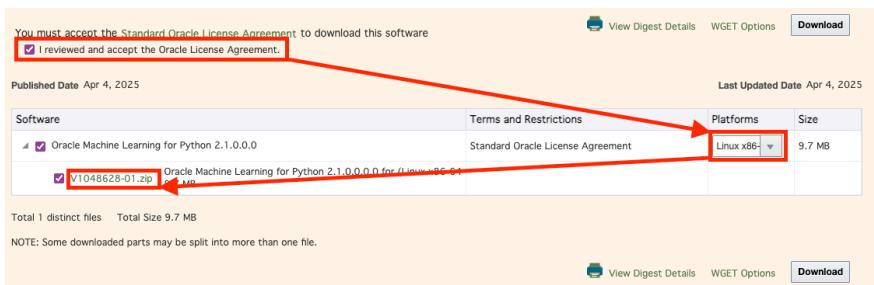
Use your **keyboard arrows** to navigate to the bottom of the file.

Copy and paste the text on the right-hand side, ensuring instant client version is the same
→

Press the '**Esc**' key on your keyboard to exit insert mode.

Type the following and press enter:
:wq

<p>Reload the .bashrc file.</p> <pre>source .bashrc</pre> <p>Now every time you start the compute instance, the environment variables will automatically be set.</p>	
<p>Create a Python requirements file for libraries to be installed.</p> <pre>vi requirements.txt</pre>	
<p>Press i on your keyboard to insert text.</p> <p>Use your keyboard arrows to navigate to the bottom of the file.</p>	
<p>Copy and paste the text on the right-hand side. →</p> <p>Press the 'Esc' key on your keyboard to exit insert mode.</p> <p>Type the following and press enter: :wq</p>	 <pre>--extra-index-url https://download.pytorch.org/whl/cpu pandas==2.2.2 setuptools==70.0.0 scipy==1.14.0 matplotlib==3.8.4 oracledb==2.4.1 scikit-learn==1.5.1 numpy==2.0.1 onnxruntime==1.20.0 onnxruntime-extensions==0.12.0 onnx==1.17.0 torch==2.6.0 transformers==4.49.0 sentencepiece==0.2.0</pre>

<p>Upgrade pip.</p> <pre>pip3 install --upgrade pip</pre>	<pre>[~]\$ pip3 install --upgrade pip Requirement already satisfied: pip in ./python/lib/python3.12/site-packages (24.2) Collecting pip Downloading pip-25.3-py3-none-any.whl.metadata (4.7 kB) Downloading pip-25.3-py3-none-any.whl (1.8 MB) 1.8/1.8 MB 116.9 MB/s eta 0:00:00 Installing collected packages: pip Attempting uninstall: pip Found existing installation: pip 24.2 Uninstalling pip-24.2: Successfully uninstalled pip-24.2 Successfully installed pip-25.3 [~]\$</pre>															
<p>Install Python requirements file.</p> <pre>pip3 install -r requirements.txt</pre>	<pre>[~]\$ pip3 install -r requirements.txt</pre>															
<p>In your local browser download OML4Py 2.1.0.</p> <p>Visit https://www.oracle.com/database/technologies/oml4py-downloads.html and download the Client for OML4Py 2.1.0 (Database 23/26ai).</p>	<p>Oracle Machine Learning for Python Downloads</p> <table border="1"> <thead> <tr> <th>Platform</th> <th>OML4Py 2.1.0 (Database 23ai)</th> <th>OML4Py 2.0.1 (Database 23ai)</th> <th>OML4Py 2.0 (Database 23ai)</th> <th>OML4Py 2.0 (Database 19c/21c)</th> </tr> <tr> <th></th> <th>Documentation System Requirements</th> <th>Documentation System Requirements</th> <th>Documentation System Requirements</th> <th>Documentation System Requirements</th> </tr> </thead> <tbody> <tr> <td>Linux 64-bit</td> <td>Client</td> <td>Client</td> <td>Client (8M)</td> <td>Server Client</td> </tr> </tbody> </table>	Platform	OML4Py 2.1.0 (Database 23ai)	OML4Py 2.0.1 (Database 23ai)	OML4Py 2.0 (Database 23ai)	OML4Py 2.0 (Database 19c/21c)		Documentation System Requirements	Documentation System Requirements	Documentation System Requirements	Documentation System Requirements	Linux 64-bit	Client	Client	Client (8M)	Server Client
Platform	OML4Py 2.1.0 (Database 23ai)	OML4Py 2.0.1 (Database 23ai)	OML4Py 2.0 (Database 23ai)	OML4Py 2.0 (Database 19c/21c)												
	Documentation System Requirements	Documentation System Requirements	Documentation System Requirements	Documentation System Requirements												
Linux 64-bit	Client	Client	Client (8M)	Server Client												
<p>Read and accept the Oracle License Agreement.</p> <p>Select the Platform as Linux.</p> <p>Then click on the v1048628-01.zip to download the ZIP file.</p>	 <p>You must accept the Standard Oracle License Agreement to download this software <input checked="" type="checkbox"/> I reviewed and accept the Oracle License Agreement.</p> <p>Published Date Apr 4, 2025 Last Updated Date Apr 4, 2025</p> <table border="1"> <thead> <tr> <th>Software</th> <th>Terms and Restrictions</th> <th>Platforms</th> <th>Size</th> </tr> </thead> <tbody> <tr> <td><input checked="" type="checkbox"/> Oracle Machine Learning for Python 2.1.0.0</td> <td>Standard Oracle License Agreement</td> <td>Linux x86- 64bit</td> <td>9.7 MB</td> </tr> <tr> <td><input checked="" type="checkbox"/> V1048628-01.zip</td> <td>Oracle Machine Learning for Python 2.1.0.0.0 for Linux x86-64bit</td> <td></td> <td></td> </tr> </tbody> </table> <p>Total 1 distinct files Total Size 9.7 MB NOTE: Some downloaded parts may be split into more than one file.</p>	Software	Terms and Restrictions	Platforms	Size	<input checked="" type="checkbox"/> Oracle Machine Learning for Python 2.1.0.0	Standard Oracle License Agreement	Linux x86- 64bit	9.7 MB	<input checked="" type="checkbox"/> V1048628-01.zip	Oracle Machine Learning for Python 2.1.0.0.0 for Linux x86-64bit					
Software	Terms and Restrictions	Platforms	Size													
<input checked="" type="checkbox"/> Oracle Machine Learning for Python 2.1.0.0	Standard Oracle License Agreement	Linux x86- 64bit	9.7 MB													
<input checked="" type="checkbox"/> V1048628-01.zip	Oracle Machine Learning for Python 2.1.0.0.0 for Linux x86-64bit															

Copy the downloaded .zip file to your compute instance.

As I am using a Mac, I can use the scp command in my local Terminal.

```
scp -i <private-key-file>
<location-of-downloaded-zip>
opc@<public-ip>:/home/opc
```

```
$ scp -i oml4py2.1-client.key ~/Downloads/V1048628-01.zip opc@192.168.1.11:/home/opc
V1048628-01.zip                                100% 9936KB  20.9MB/s  00:00
$
```

If you are using a Windows, you can do this using Windows PowerShell with the same command or download and use a tool such as WinSCP - <https://winscp.net/eng/index.php>

Back in our compute instance, we can now see the ZIP file copied across within our compute instance.

```
ls -la
```

```
total 151932
drwxrwxr-x. 3 opc opc      4096 Dec  1 10:55 instantclient_23_4
-rw-rw-r--. 1 opc opc 118377607 Apr 27 2024 instantclient-basic-linux.x64-23.4.0.24.05.zip
drwxrwxr-x. 2 opc opc      64 Dec  1 10:55 META-INF
drwxrwxr-x. 22 opc opc     4096 Dec  1 10:52 python
-rw-rw-r--. 1 opc opc 27009716 Sep  6 2024 Python-3.12.6.tgz
-rw-rw-r--. 1 opc opc    288 Dec  1 10:57 requirements.txt
-rw-r--r--. 1 opc opc 10174054 Dec  1 11:02 V1048628-01.zip
[ ~]$
```

Unzip the file.

```
unzip V1048628-01.zip
```

```
[ ~]$ ~]$ unzip V1048628-01.zip
Archive:  V1048628-01.zip
  inflating: client/client.pl
  inflating: client/OML4PInstallShared.pm
  inflating: client/oml-2.1-cp312-cp312-linux_x86_64.whl
  extracting: client/oml4py.ver
[ ~]$
```

Install the OML4Py python library.

```
pip3 install
client/oml-2.1-
cp312-cp312-
linux_x86_64.whl
```

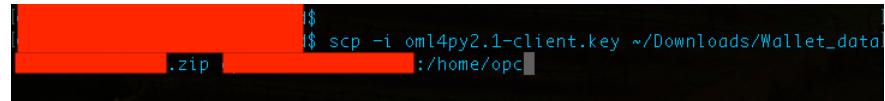
```
[ ~]$ ~]$ pip3 install client/oml-2.1-cp312-cp312-linux_x86_64.whl
```

We will now configure Oracle Instant Client to communicate with our Autonomous AI Database so we can save models directly to the DB.

Copy your Autonomous AI Database Wallet file to your compute instance.

As I am using a Mac, I can use the scp command in my local Terminal.

```
scp -i <private-key-file>
<location-of-wallet-file>
opc@<public-ip>:/home/opc
```



```
$ scp -i oml4py2.1-client.key ~/Downloads/Wallet_data.zip opc@127.0.0.1:/home/opc
```

If you are using a Windows, you can do this using Windows PowerShell with the same command or download and use a tool such as WinSCP - <https://winscp.net/eng/index.php>

Back in my compute instance I can create a new empty directory to unzip my Wallet into.

```
mkdir <new-wallet-directory>
```

```
unzip <zipped-wallet> -d <new-wallet-directory>
```



```
[~]$ mkdir Wallet_data
[~]$ unzip Wallet_data.zip -d Wallet_data
```

Make a copy of your **tnsnames.ora** and **sqlnet.ora** files into the Oracle Instant Client network/admin directory.

```
cp <new-wallet-
directory>/tnsnam
es.ora
instantclient_23_
4/network/admin
```

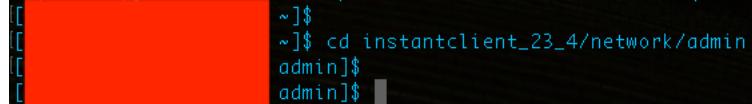


```
[~]$ cp Wallet/tnsnames.ora instantclient_23_4/network/admin/
[~]$ cp Wallet/sqlnet.ora instantclient_23_4/network/admin/
[~]$
```

```
cp <new-wallet-
directory>/sqlnet
.ora
instantclient_23_
4/network/admin
```

Navigate to the Network Admin directory.

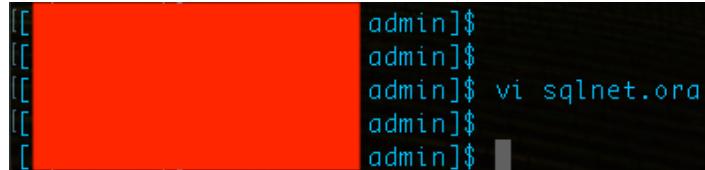
```
cd
instantclient_23_
4/network/admin
```



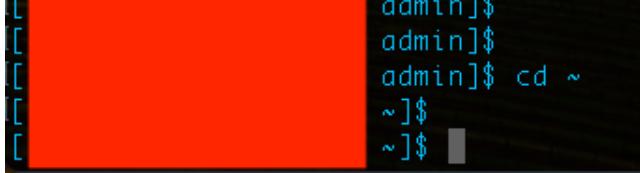
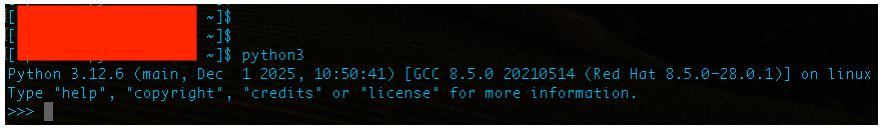
```
[~]$ cd instantclient_23_4/network/admin
[admin]$
```

Edit the sqlnet.ora file to update your Unzipped Wallet Location.

```
vi sqlnet.ora
```



```
[admin]$ vi sqlnet.ora
[admin]$
```

<p>Update the Wallet Directory Location.</p> <p>Press <i>i</i> on your keyboard to insert text.</p> <p>Use your keyboard arrows to DIRECTORY parameter and update this to the location of your unzipped Wallet.</p> <p>Press the 'Esc' key on your keyboard to exit insert mode.</p> <p>Type the following and press enter: :wq</p>	
<p>Navigate back to the Home Directory.</p>	 <pre>admin]\$ cd ~ ~]\$</pre>
<p>Enter the Python Terminal.</p>	 <pre>~]\$ python3 Python 3.12.6 (main, Dec 1 2025, 10:50:41) [GCC 8.5.0 20210514 (Red Hat 8.5.0-28.0.1)] on linux Type "help", "copyright", "credits" or "license" for more information. >>></pre>

Import the ONNXPipeline Libraries and display the pre-configured models.

```
from oml.utils
import
ONNXPipelineConfig
g

ONNXPipelineConfig
g.show_preconfigu
red()
```

```
[root@... ~]$
[[...]]$ python3
Python 3.12.6 (main, Dec 1 2025, 10:50:41) [GCC 8.5.0 20210514 (Red Hat 8.5.0-28.0.1)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from oml.utils import ONNXPipelineConfig
>>>
>>> ONNXPipelineConfig.show_preconfigured()
['sentence-transformers/all-mptnet-base-v2', 'sentence-transformers/all-MiniLM-L6-v2', 'sentence-transformers/multi-qa-MiniLM-L6-cos-v1', 'sentence-transformers/distiluse-base-multilingual-cased-v2', 'sentence-transformers/all-MiniLM-L12-v2', 'BAAI/bge-small-en-v1.5', 'BAAI/bge-base-en-v1.5', 'taylorAI/bge-micro-v2', 'intfloat/e5-small-v2', 'intfloat/e5-base-v2', 'thenlper/gte-base', 'thenlper/gte-small', 'TaylorAI/gte-tiny', 'sentence-transformers/paparaphe-multilingual-mptnet-base-v2', 'intfloat/multilingual-e5-base', 'intfloat/multilingual-e5-small', 'sentence-transformers/stsb-xlm-r-multilingual', 'Snowflake/snowflake-arctic-embed-xs', 'Snowflake/snowflake-arctic-embed-s', 'Snowflake/snowflake-arctic-embed-m', 'mixedbread-ai/mxbai-embed-large-v1', 'openai/clip-vit-large-patch14', 'google/vit-base-patch16-224', 'microsoft/resnet-18', 'microsoft/resnet-50', 'WinKwakws/vit-tiny-patch16-224', 'FalconSai/nsfw_image_detection', 'WinKwakws/vit-small-patch16-224', 'naterow/vit-age-classifier', 'ritzvandwiki/gender-classification', 'AdamCodd/vit-base-nsfw-detector', 'trpkov/vit-face-expression', 'BAAI/bge-reranker-base']
```

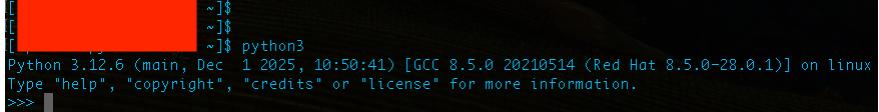
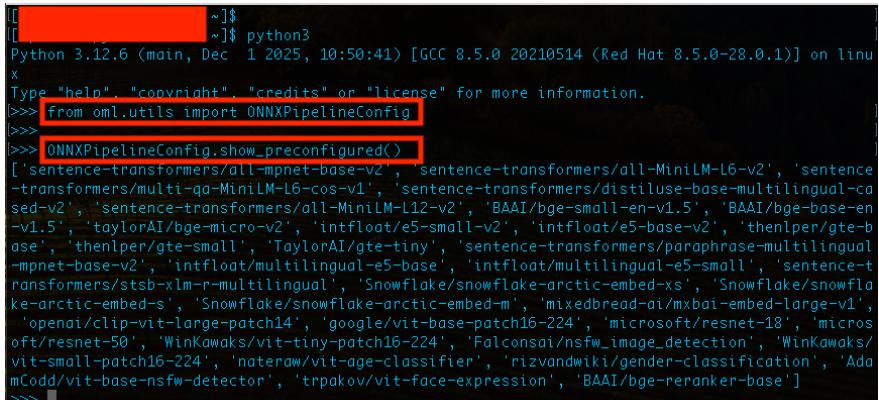
If you are able to list the 30+ preconfigured ONNX Embedding Models, you have verified the OML4Py 2.1.0 Client Installation.

Below I will go through four scenarios for downloading ONNX Embedding Models.

1. Downloading a pre-configured text embedding model locally
2. Loading a pre-configured text embedding model directly to our Autonomous DB
3. Downloading a custom multi-modal embedding model from HuggingFace locally
4. Loading a custom multi-modal embedding model from HuggingFace to our Autonomous DB

Scenario 1: Downloading a pre-configured text embedding model locally

Referenced Documentation: <https://docs.oracle.com/en/database/oracle/machine-learning/ml4py/2-23ai/mlplug/onnx-pipeline-models-text-embedding.html>

Step	Screenshot
<p>Within your OML4Py Client Installation, Open a Python session.</p> <pre data-bbox="203 653 485 698">python3</pre>	 <pre data-bbox="504 563 1383 586">Python 3.12.6 (main, Dec 1 2025, 10:50:41) [GCC 8.5.0 20210514 (Red Hat 8.5.0-28.0.1)] on linux Type "help", "copyright", "credits" or "license" for more information. >>> </pre>
<p>First, we can display the pre-configured models.</p> <p>These are a list of preconfigured ONNX embedding models that have been verified by Oracle which can be utilised as is.</p>	 <pre data-bbox="504 770 1383 1170">[...] [...] ~]\$ python3 Python 3.12.6 (main, Dec 1 2025, 10:50:41) [GCC 8.5.0 20210514 (Red Hat 8.5.0-28.0.1)] on linux Type "help", "copyright", "credits" or "license" for more information. >>> from oml.utils import ONNXPipelineConfig >>> >>> ONNXPipelineConfig.show_preconfigured() ['sentence-transformers/all-mpnet-base-v2', 'sentence-transformers/all-MiniLM-L6-v2', 'sentence-transformers/multi-qa-MiniLM-L6-cos-v1', 'sentence-transformers/distiluse-base-multilingual-cased-v2', 'sentence-transformers/all-MiniLM-L12-v2', 'BAAI/bge-small-en-v1.5', 'BAAI/bge-base-en-v1.5', 'taylorAI/bge-micro-v2', 'intfloat/e5-small-v2', 'intfloat/e5-base-v2', 'thenlper/gte-base', 'thenlper/gte-small', 'TaylorAI/gte-tiny', 'sentence-transformers/paraphrase-multilingual-mpnet-base-v2', 'intfloat/multilingual-e5-base', 'intfloat/multilingual-e5-small', 'sentence-transformers/stsb-xlm-r-multilingual', 'Snowflake/snowflake-arctic-embed-xs', 'Snowflake/snowflake-arctic-embed-s', 'Snowflake/snowflake-arctic-embed-m', 'mixedbread-ai/mxbai-embed-large-v1', 'openai/clip-vit-large-patch14', 'google/vit-base-patch16-224', 'microsoft/resnet-18', 'microsoft/resnet-50', 'WinKawaks/vit-tiny-patch16-224', 'Falconsoi/nsfw_image_detection', 'WinKawaks/vit-small-patch16-224', 'naterow/vit-age-classifier', 'rizvandwiki/gender-classification', 'AdamCodd/vit-base-nsfw-detector', 'trpkov/vit-face-expression', 'BAAI/bge-reranker-base'] >>> </pre>
<p>Code snippet below screenshot →</p>	<pre data-bbox="504 1320 1383 1462">>> from oml.utils import ONNXPipelineConfig >>> ONNXPipelineConfig.show_preconfigured()</pre>

We can then save the **sentence-transformers/all-MiniLM-L6-v2** model locally.

Code snippet
below screenshot
→

```
>>> from oml.utils import ONNXPipeline, ONNXPipelineConfig
>>>
>>> # Export to file
>>> pipeline = ONNXPipeline(model_name="sentence-transformers/all-MiniLM-L6-v2")
>>>
>>> pipeline.export2file("all-MiniLM-L6-v2",output_dir=".")
```

```
>> from oml.utils import ONNXPipeline,
ONNXPipelineConfig

>> pipeline = ONNXPipeline(model_name="sentence-
transformers/all-MiniLM-L6-v2")

>> pipeline.export2file("all-MiniLM-L6-
v2",output_dir=".")
```

```
>> exit()
```

Outside of the Python session, we can then search our local directory for the file.

```
ls -la | grep
all-Mini
```

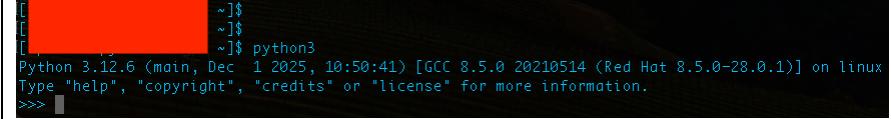
```
[~]$ ls -la | grep all-Mini
[~]$ ~]$ ls -la | grep all-Mini
[~]$ -rw-rw-r--. 1 opc  opc  90636502 Dec 1 11:21 all-MiniLM-L6-v2.onnx
[~]$ ~]$
```

This can now be copied across to the Oracle AI Database via the DB Server or via Object Storage to be loaded in as a Model.

The next scenario will show you how to do that automatically through our OML4Py and Instant Client configuration we did above.

Scenario 2: Loading a pre-configured text embedding model directly to Autonomous Database

Referenced Documentation: <https://docs.oracle.com/en/database/oracle/machine-learning/ml4py/2-23ai/mlplug/onnx-pipeline-models-text-embedding.html>

Step	Screenshot
<p>Within your OML4Py Client Installation, Open a Python session.</p> <pre data-bbox="192 698 493 781">python3</pre>	 <pre data-bbox="493 653 1384 781"> [...] ~]\$ ~]\$ ~]\$ python3 Python 3.12.6 (main, Dec 1 2025, 10:50:41) [GCC 8.5.0 20210514 (Red Hat 8.5.0-28.0.1)] on linux Type "help", "copyright", "credits" or "license" for more information. >>> [...] </pre>
<p>Import the libraries.</p> <p>Connect to our Autonomous DB</p> <p>(Replace you're your connection details, where dsn is our Autonomous DB service name.)</p> <p>Define our ONNX pipeline for a pre-configured model</p> <p>Export to Database.</p>	<pre data-bbox="493 826 1384 1028">>>> import oml >>> from oml.utils import ONNXPipeline, ONNXPipelineConfig >>> >>> oml.connect(user="[REDACTED]", password="[REDACTED]", dsn="ALL_MINILM_L6"); >>> >>> pipeline = ONNXPipeline(model_name="sentence-transformers/all-MiniLM-L6-v2") >>> pipeline.export2db("ALL_MINILM_L6") >>> exit()</pre> <pre data-bbox="493 1096 1384 1215">>> import oml >> from oml.utils import ONNXPipeline, ONNXPipelineConfig</pre> <pre data-bbox="493 1275 1384 1365">>> oml.connect(user="username", password="password", dsn="dsn");</pre> <pre data-bbox="493 1417 1384 1484">>> pipeline = ONNXPipeline(model_name="sentence- transformers/all-MiniLM-L6-v2")</pre>
<p>Code snippet below screenshot →</p>	<pre data-bbox="493 1545 1384 1590">>> pipeline.export2db("ALL_MINILM_L6")</pre> <pre data-bbox="493 1641 1384 1680">>> exit()</pre>

We can then check via SQL Developer whether the model was loaded in successfully.

```
SELECT  
MODEL_NAME,  
ALGORITHM,  
MINING_FUNCTION  
FROM  
USER_MINING_MODEL  
S  
WHERE  
MODEL_NAME='ALL_M  
INILM_L6';
```

The screenshot shows the Oracle SQL Developer interface. A query is being run in the worksheet:

```
1 SELECT  
2   MODEL_NAME,  
3   ALGORITHM,  
4   MINING_FUNCTION  
5  FROM  
6   USER_MINING_MODELS  
7 WHERE  
8   MODEL_NAME = 'ALL_MINILM_L6';
```

The results are displayed in the 'Query Result' tab:

	MODEL_NAME	ALGORITHM	MINING_FUNCTION
1	ALL_MINILM_L6	ONNX	EMBEDDING

Execution time: 0.013 seconds

We can then test the embedding model works.

```
SELECT  
VECTOR_EMBEDDING(  
ALL_MINILM_L6  
USING 'HELLO  
WORLD' AS DATA)  
AS EMBEDDING;
```

The screenshot shows the Oracle SQL Developer interface. A query is being run in the worksheet:

```
10  
11  
12 SELECT VECTOR_EMBEDDING(ALL_MINILM_L6 USING 'HELLO WORLD' AS DATA) AS EMBEDDING;  
13 |
```

The results are displayed in the 'Query Result' tab:

EMBEDDING
[-3.4477286E-002,3.10232006E-002,6.73497701E-003,2.61089671E-002,-3.936]

Execution time: 0.483 seconds

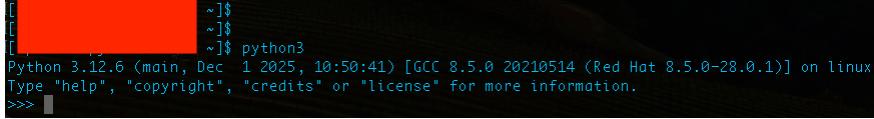
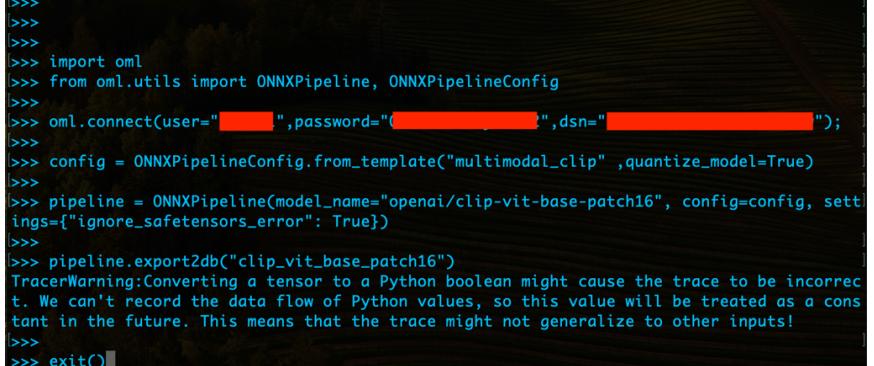
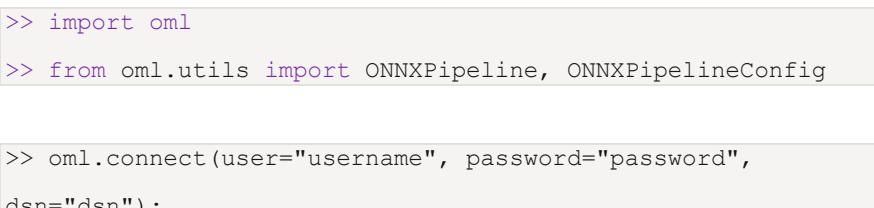
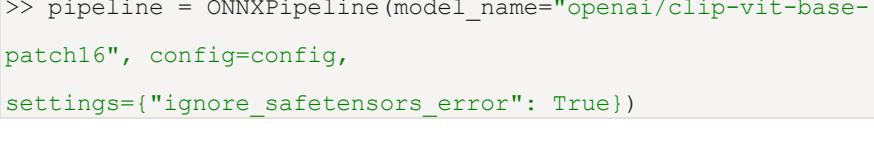
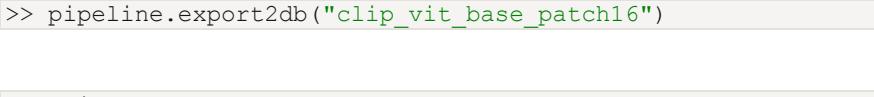
Scenario 3: Downloading a custom multi-modal embedding model from HuggingFace locally

Referenced Documentation: <https://docs.oracle.com/en/database/oracle/machine-learning/ml4py/2-23ai/mlplug/onnx-pipeline-models-multi-modal-embedding.html>

Step	Screenshot
Within your OML4Py Client Installation, Open a Python session.	<pre>[...] [...]\$ ~]\$ python3 Python 3.12.6 (main, Dec 1 2025, 10:50:41) [GCC 8.5.0 20210514 (Red Hat 8.5.0-28.0.1)] on linux Type "help", "copyright", "credits" or "license" for more information. >>> [...]</pre> <p>python3</p>
We will look at downloading the openai/clip-vit-base-patch16 model from HuggingFace using the ONNX Pipeline Templates.	<pre>>>> >>> >>> import oml >>> from oml.utils import ONNXPipeline, ONNXPipelineConfig >>> >>> oml.connect(user="...", password="...", dsn="...") >>> >>> config = ONNXPipelineConfig.from_template("multimodal_clip") >>> >>> pipeline = ONNXPipeline("openai/clip-vit-base-patch16", config=config, settings={"ignore_safetensors_error": True}) >>> >>> pipeline.export2file("clip-vit-base-patch16", output_dir=".") Try setting the quantize_model property to true for models of this size. tokenizer_config.json: 100% ██████████ 905/905 [00:00<00:00, 9.40MB/s] vocab.json: 961KB [00:00, 167MB/s] merges.txt: 525KB [00:00, 188MB/s]</pre>
Import the Libraries	<pre>>> from oml.utils import ONNXPipeline, ONNXPipelineConfig</pre>
Define the multimodal CLIP Template.	<pre>>> config = ONNXPipelineConfig.from_template("multimodal_clip")</pre>
Define the ONNX Pipeline.	<pre>>> pipeline = ONNXPipeline(model_name="openai/clip-vit-base-patch16", config=config, settings={"ignore_safetensors_error": True})</pre>
Export to file.	<pre>>> pipeline.export2file("clip-vit-base-patch16", output_dir=".")</pre>
Code snippet below screenshot →	<pre>>> exit()</pre>
Outside of the Python session, we can then search our local directory for the file.	<pre>[...] [...]\$ ~]\$ ls -la grep clip -rw-rw-r--. 1 opc opc 345044148 Dec 1 12:01 clip-vit-base-patch16_img.onnx -rw-rw-r--. 1 opc opc 255362807 Dec 1 12:01 clip-vit-base-patch16_txt.onnx [...]\$ ~]\$ [...]</pre>
ls -la grep clip	

Scenario 4: Loading a custom multi-modal embedding model directly to Autonomous Database

Referenced Documentation: <https://docs.oracle.com/en/database/oracle/machine-learning/ml4py/2-23ai/mlplug/onnx-pipeline-models-multi-modal-embedding.html>

Step	Screenshot
Within your OML4Py Client Installation, Open a Python session.	 <pre>[...] [...] ~]\$ python3 Python 3.12.6 (main, Dec 1 2025, 10:50:41) [GCC 8.5.0 20210514 (Red Hat 8.5.0-28.0.1)] on linux Type "help", "copyright", "credits" or "license" for more information. >>> [...]</pre> <p>python3</p>
We will look at downloading the openai/clip-vit-base-patch16 model from HuggingFace using the ONNX Pipeline Templates.	 <pre>>>> >>> >>> >>> import oml >>> from oml.utils import ONNXPipeline, ONNXPipelineConfig >>> >>> oml.connect(user="...", password="...", dsn="..."); >>> >>> config = ONNXPipelineConfig.from_template("multimodal_clip", quantize_model=True) >>> >>> pipeline = ONNXPipeline(model_name="openai/clip-vit-base-patch16", config=config, settings={"ignore_safetensors_error": True}) >>> >>> pipeline.export2db("clip_vit_base_patch16") TracerWarning:Converting a tensor to a Python boolean might cause the trace to be incorrect. We can't record the data flow of Python values, so this value will be treated as a constant in the future. This means that the trace might not generalize to other inputs! >>> >>> exit()</pre>
Import the Libraries	 <pre>>> import oml >> from oml.utils import ONNXPipeline, ONNXPipelineConfig</pre>
Connect to our Autonomous DB (Replace you're your connection details, where dsn is our Autonomous DB service name.)	 <pre>>> oml.connect(user="username", password="password", dsn="dsn");</pre>
Define an empty multimodal CLIP Template.	 <pre>>> config = ONNXPipelineConfig.from_template("multimodal_clip")</pre>
Define the ONNX Pipeline	 <pre>>> pipeline = ONNXPipeline(model_name="openai/clip-vit-base- patch16", config=config, settings={"ignore_safetensors_error": True})</pre>
Export to Database.	 <pre>>> pipeline.export2db("clip_vit_base_patch16")</pre>
Code snippet below screenshot →	 <pre>>> exit()</pre>

We can then check via SQL Developer whether the model was loaded in successfully.

```
SELECT  
MODEL_NAME,  
ALGORITHM,  
MINING_FUNCTION  
FROM  
USER_MINING_MODEL  
S  
WHERE MODEL_NAME  
LIKE 'CLIP%';
```

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there is a code editor window containing the following SQL query:

```
1 SELECT MODEL_NAME, ALGORITHM, MINING_FUNCTION  
2 FROM USER_MINING_MODELS  
3 WHERE MODEL_NAME LIKE 'CLIP%';  
4
```

In the bottom-right pane, the "Query Result" tab is selected, displaying the results of the executed query. The results are presented in a table:

	MODEL_NAME	ALGORITHM	MINING_FUNCTION
1	CLIP_VIT_BASE_PATCH16_IMG	ONNX	EMBEDDING
2	CLIP_VIT_BASE_PATCH16_TXT	ONNX	EMBEDDING

The execution time is shown as 0.002 seconds.