

Deploying a Verified Model Downloaded from HuggingFace using AI Quick Actions

Referenced Documentation

<https://docs.oracle.com/en-us/iaas/data-science/using/ai-quick-actions-models-byom.htm>

<https://docs.oracle.com/en-us/iaas/data-science/using/ai-quick-actions-model-deploy.htm>

Description

If you have models you want to use instead of the cached models provided by Data Science, you can bring them into AI Quick Actions from Object Storage or from Hugging Face by registering the model.

Hugging Face is an open-source model repository. You can bring in models from here to use in AI Quick Actions. Hugging Face offers certain gated models that require the acceptance of user agreement. To bring a gated model from Hugging Face into AI Quick Actions, sign in to Hugging Face using the Hugging Face CLI and your Hugging Face token from a terminal inside the Notebook.

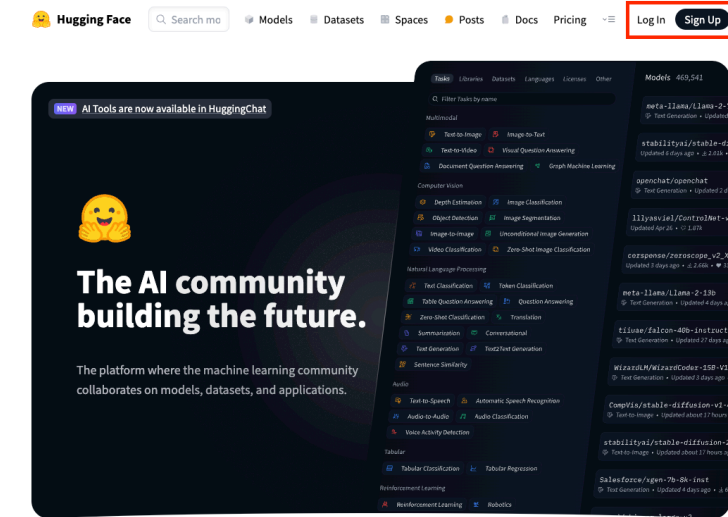
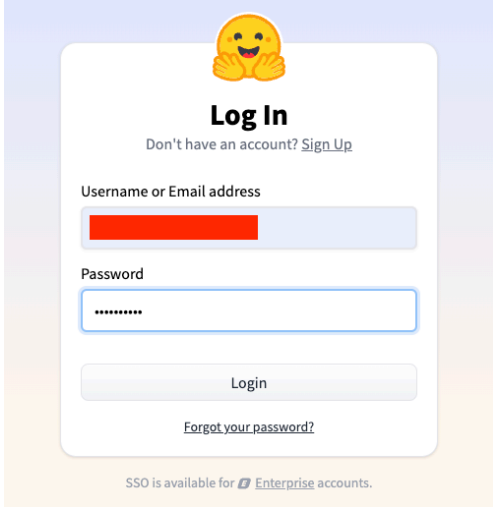
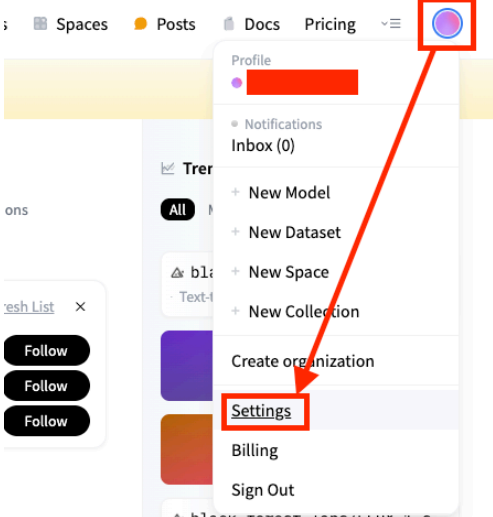
You can create a Model Deployment from the foundation models with the tag Ready to Deploy in the Model Explorer, or with fine-tuned models. When you create a Model Deployment in AI Quick Actions, you're creating an OCI Data Science Model Deployment, which is a managed resource in the OCI Data Science Service. You can deploy the model as HTTP endpoints in OCI.

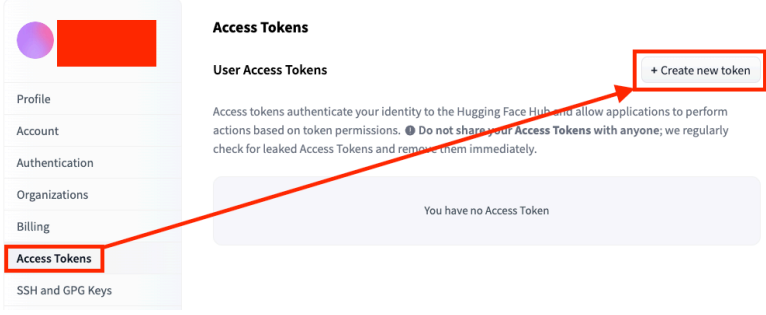
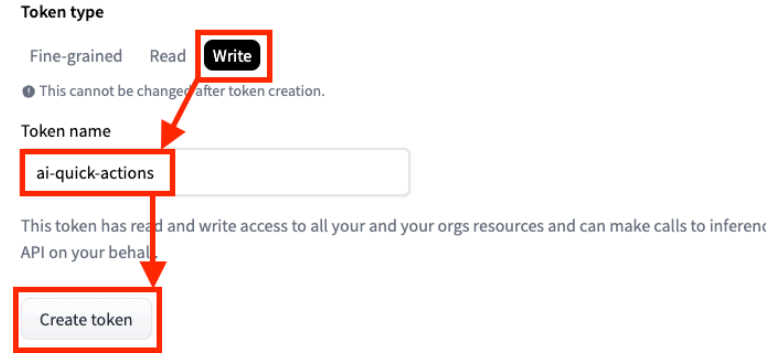
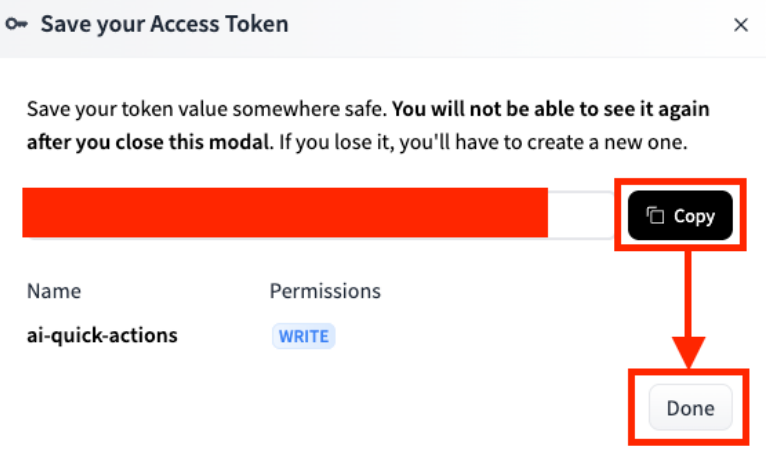

Pre-Requisites

- Implement the required AI Quick Actions Policies - <https://docs.oracle.com/en-us/iaas/data-science/using/ai-quick-actions-set-up.htm>
- Ensure you have your OCI Data Science GPU service limits raised for the GPU Shapes you plan to use. This can be done from OCI Console.
- Provisioned OCI Data Science Project and Notebook Session (Must be deactivated and reactivated if created before the policies were implemented).
- OCI Log Group & Log Created (Optional)
- HuggingFace Login Details



Guide

Step	Screenshot
<p>First, we will login to HuggingFace to generate an Access Token and Accept the Licence agreement for the model we want to deploy.</p> <p>https://huggingface.co/</p> <p>Click Login or Sign Up if required.</p>	
<p>Login to HuggingFace.</p>	
<p>From your Profile Icon visit Settings.</p>	

<p>Navigate to Access Tokens</p> <p>Click Create new token.</p>	
<p>Select token type as Write.</p> <p>Give the Token a Name.</p> <p>Click Create Token.</p>	
<p>Copy your token and save it somewhere safe as it won't be displayed again.</p> <p>Click Copy (save somewhere).</p> <p>Click Done.</p>	
<p>Within the HuggingFace Search Bar, search for the OCI Data Science Verified Model you plan to deploy.</p> <p>In our case we will go with google/gemma-1.1-7b-it</p> <p>Select the Model.</p>	



If required and the model is gated, you will need to Acknowledge the License before your account has the permissions to utilise within OCI Data Science AI Quick Actions.

Click Acknowledge License.

Access Gemma on Hugging Face

This repository is publicly accessible, but you have to accept the conditions to access its files and content.

To access Gemma on Hugging Face, you're required to review and agree to Google's usage license. To do this, please ensure you're logged-in to Hugging Face and click below. Requests are processed immediately.

By agreeing you accept to share your contact information (email and username) with the repository authors.

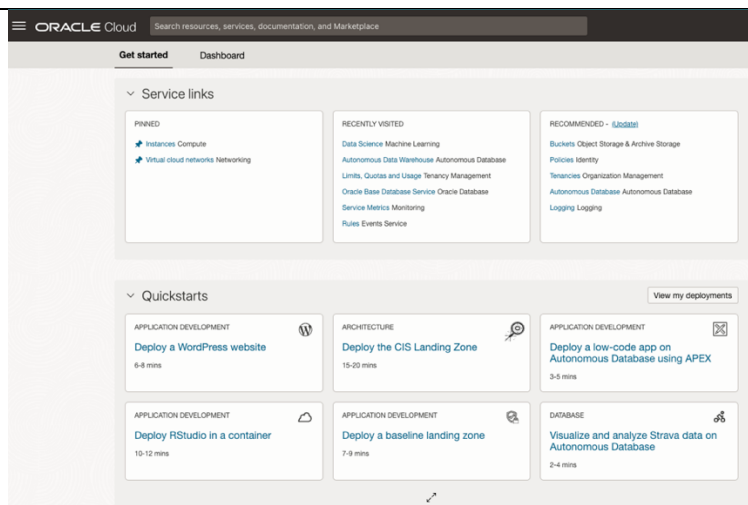
☒ Acknowledge license

IT IS YOUR SOLE RESPONSIBILITY TO READ AND ACKNOWLEDGE THE LICENSE FOR THE MODEL YOU PLAN TO USE.

IT MAY TAKE 10-15 MINUTES FOR YOUR ACCOUNT TO REGISTER YOU HAVE ACCEPTED THE LICENSE AGREEMENT.

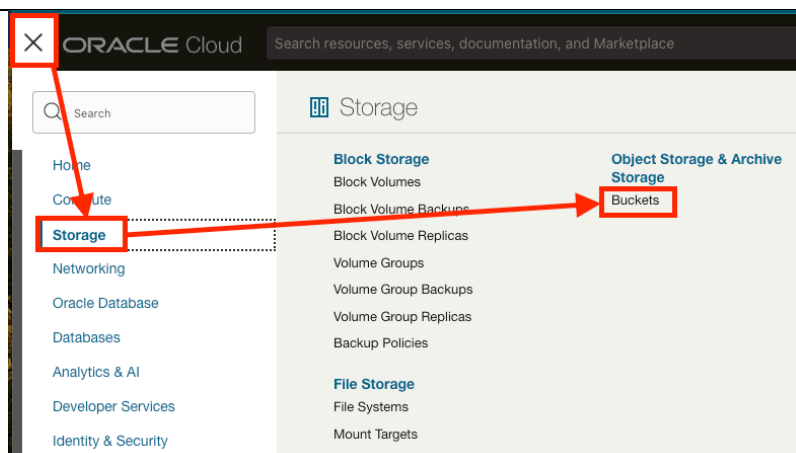
Login to the Cloud Console.

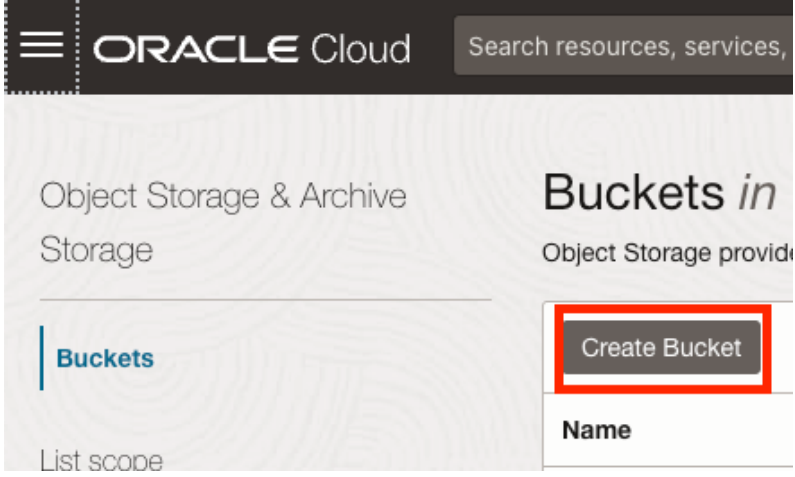
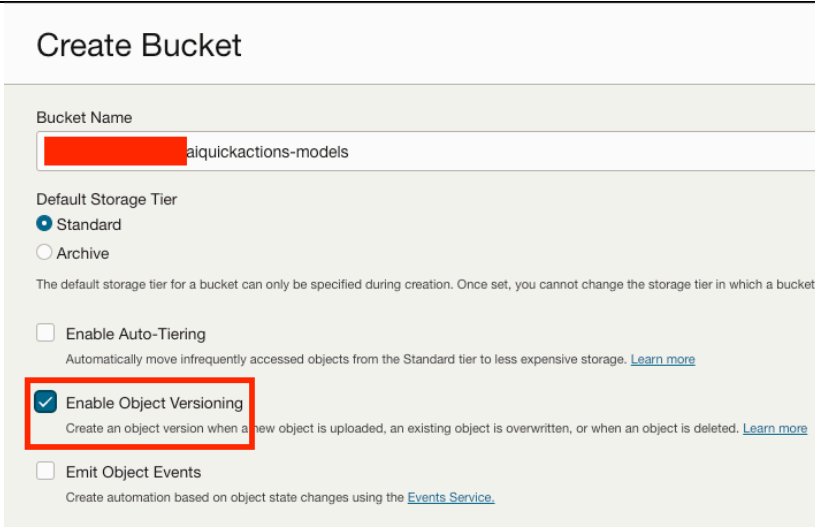
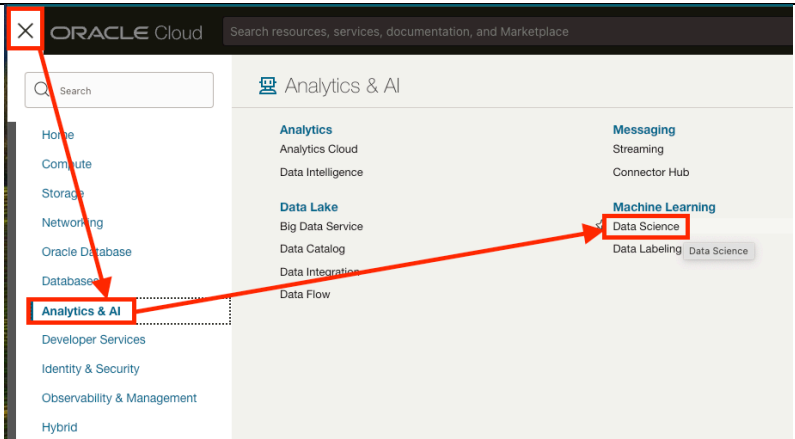
cloud.oracle.com



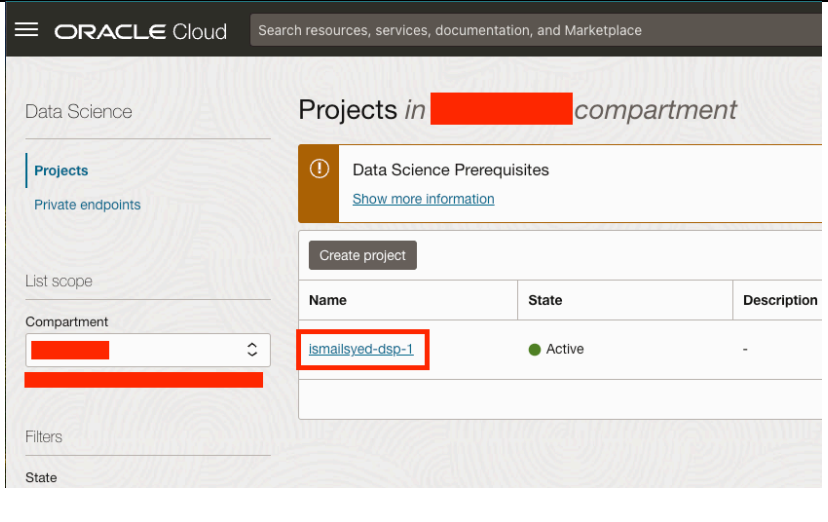
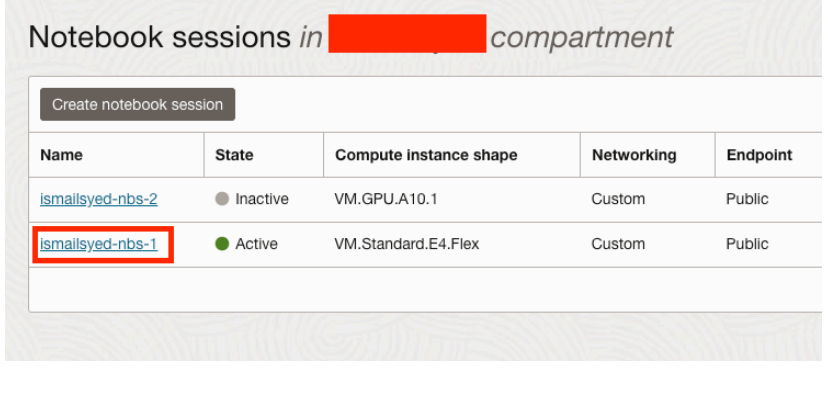
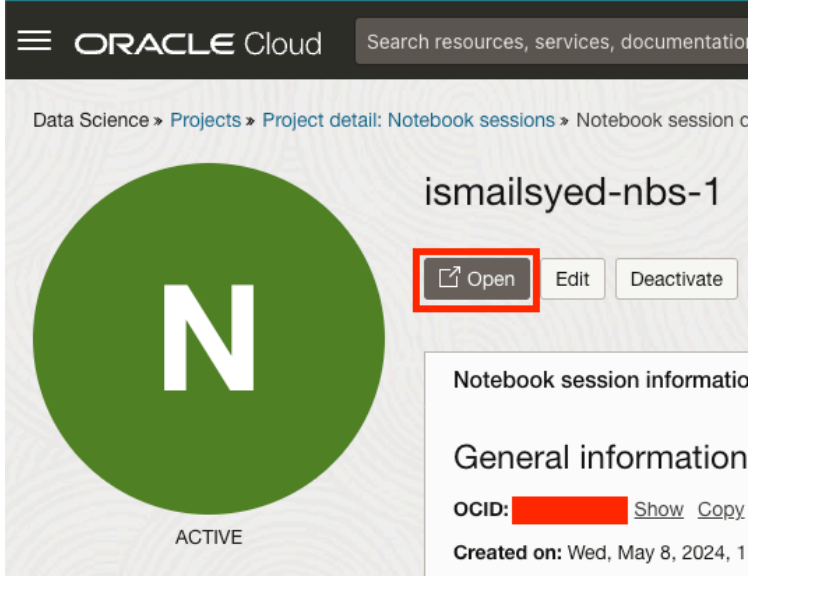
First, we will create a bucket to store our downloaded HuggingFace Model to.

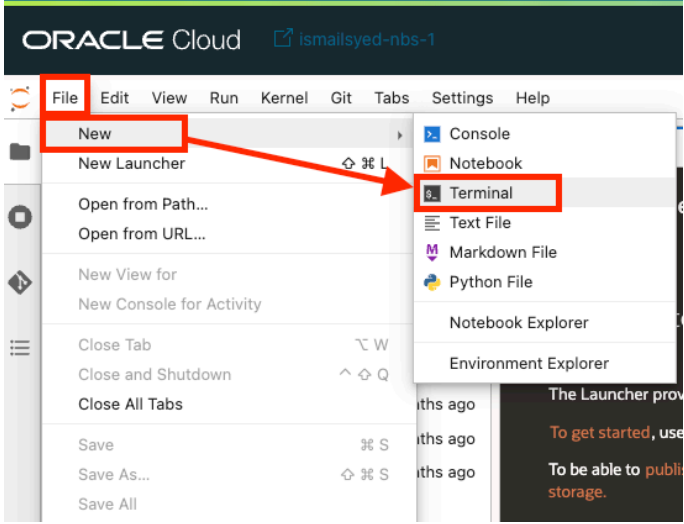


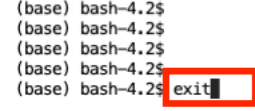
Navigate to OCI Menu > Storage > Buckets.



<p>Click Create Bucket.</p>	 <p>ORACLE Cloud Search resources, services, documentation, and Marketplace</p> <p>Object Storage & Archive Storage</p> <p>Buckets in Object Storage provided</p> <p>Create Bucket</p> <p>Name</p> <p>List scope</p>
<p>The first bucket we create will be for our finetuned model.</p> <p>Enter a Name.</p> <p>Enable Object Versioning.</p> <p>Click Create.</p>	 <p>Create Bucket</p> <p>Bucket Name</p> <p>aiquickactions-models</p> <p>Default Storage Tier</p> <p>Standard</p> <p>Archive</p> <p>The default storage tier for a bucket can only be specified during creation. Once set, you cannot change the storage tier in which a bucket is created.</p> <p><input type="checkbox"/> Enable Auto-Tiering</p> <p>Automatically move infrequently accessed objects from the Standard tier to less expensive storage. Learn more</p> <p><input checked="" type="checkbox"/> Enable Object Versioning</p> <p>Create an object version when a new object is uploaded, an existing object is overwritten, or when an object is deleted. Learn more</p> <p><input type="checkbox"/> Emit Object Events</p> <p>Create automation based on object state changes using the Events Service.</p>
<p>Now navigate to your Data Science Projects.</p> <p>OCI Menu > Analytics & AI > Data Science.</p>	 <p>ORACLE Cloud Search resources, services, documentation, and Marketplace</p> <p>Analytics & AI</p> <p>Analytics</p> <p>Analytics Cloud</p> <p>Data Intelligence</p> <p>Data Lake</p> <p>Big Data Service</p> <p>Data Catalog</p> <p>Data Integration</p> <p>Data Flow</p> <p>Messaging</p> <p>Streaming</p> <p>Connector Hub</p> <p>Machine Learning</p> <p>Data Science</p> <p>Data Labeling</p> <p>Data Science</p>

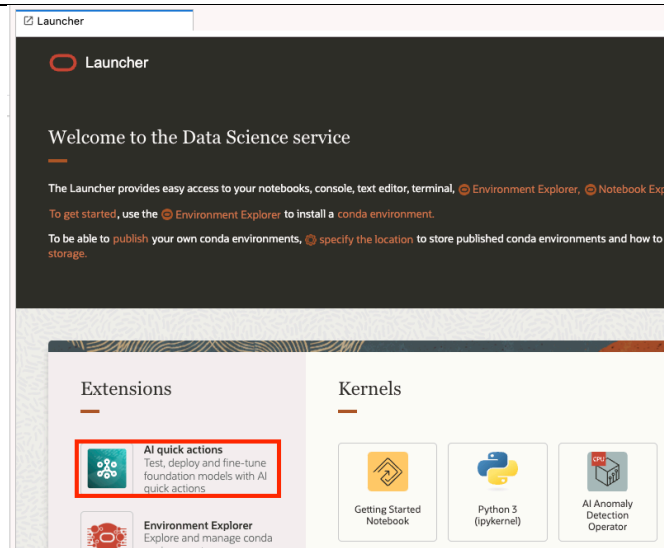


<p>Open up your existing Data Science Project.</p>	
<p>Click on your existing Data Science Notebook Session.</p> <p>Note – This does not have to be a GPU Shape.</p>	
<p>Click on Open.</p> <p>This will open up your Data Science Notebook Session.</p> <p>You will have to reauthenticate.</p>	

<p>Once your Notebook Session is open, open a Terminal.</p> <p>File > New > Terminal.</p>	
<p>We will then login to Hugging Face using the CLI Tool.</p> <p>Enter: huggingface-cli login</p>	
<p>You will then be prompted to enter your Access Token we generated earlier within HuggingFace.</p> <p>Enter your Access Token.</p> <p>You will then be prompted to Add the token as a git credential.</p> <p>Enter: n</p> <p>You should get a message saying Login Successful.</p>	
<p>You can then exit the Terminal: exit</p>	

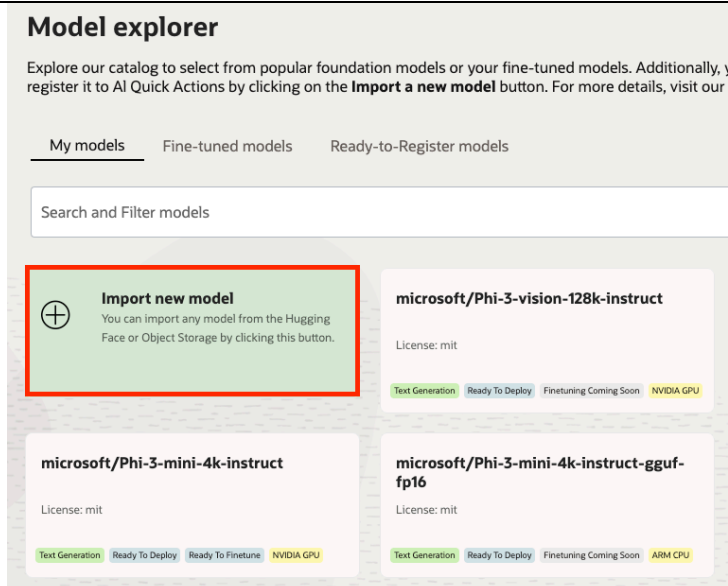
If the policies within the pre-requisites have been implemented correctly you should be able to open up the AI Quick Actions Extension within the Launcher.

Click AI quick actions.



In this demo we will be deploying **google/gemma-1.1-7b-it** as it is not a cached model, we need to first download it into our environment.

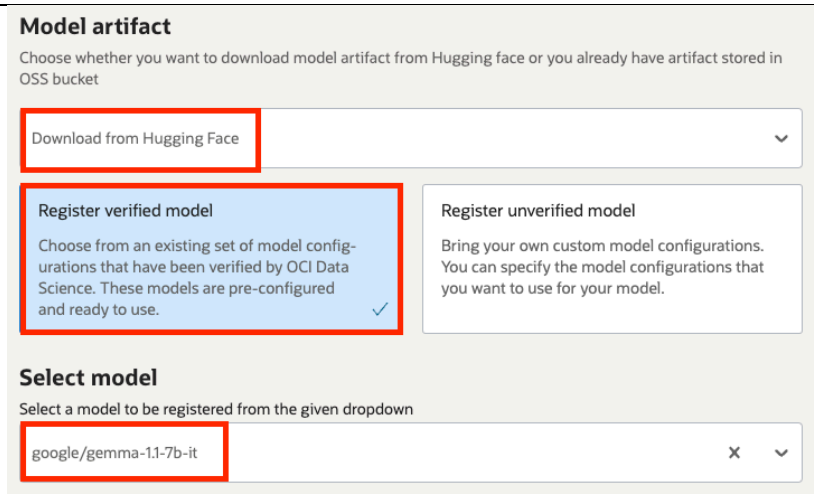
Click Import new model.



Select: Download from Hugging Face

The Gemma Models is part of our verified model list. Select: **Register verified model.**

Use the drop down to select **google/gemma-1.1-7b-it**



We then need to specify our Object Storage Bucket we created earlier to store our Downloaded Model.

Select:

- **Compartment**
- **Bucket Name**
- **Add a file prefix if required.**

Object Storage location

Specify the Object Storage bucket where the model artifacts should be downloaded

Select compartment

[Redacted]

Object Storage location

[Redacted] aiquickactions-models

Object Storage path

[Redacted]

google-gemma-1.1-7b-it

Must be a directory

Close

Register

This will then open a Terminal and kick start the download of our model to Object Storage.

```
(base) bash-4.2$ ads aqua model register --model google/gemma-1.1-7b-it --os_path oci://[Redacted] google-gemma-1.1-7b-it --download
_from hf True --compartment_id [Redacted]
INFO:ads.common:Using 'resource.principal' authentication.
generation_config.json: 100% | 132/132 [00:00:00, 69.9kB/s]
config.json: 100% | 620/620 [00:00:00, 396kB/s]
.gitattributes: 100% | 1.57k/1.57k [00:00:00, 696kB/s]
model.safetensors.index.json: 100% | 20.9k/20.9k [00:00:00, 10.0kB/s]
special_tokens_map.json: 100% | 636/636 [00:00:00, 270kB/s]
tokenizer_config.json: 100% | 35.2k/35.2k [00:00:00, 30.4kB/s]
tokenizer.json: 100% | 17.5M/17.5M [00:00:00, 98.4kB/s]
tokenizer.model: 100% | 4.2M/4.2M [00:00:00, 189kB/s]
model-00001-of-00004.safetensors: 100% | 2.15G/2.15G [00:17:00, 123kB/s]
model-00002-of-00004.safetensors: 100% | 5.00G/5.00G [01:22:00, 68.3kB/s]
model-00003-of-00004.safetensors: 100% | 4.98G/4.98G [01:37:00, 56.0kB/s]
model-00004-of-00004.safetensors: 100% | 4.98G/4.98G [01:37:00, 51.4kB/s]
Fetching 13 files: 100% | 13/13 [01:37:00, 7.49s/it]
Fetching 13 files: 0%
```

Once download is complete, we can exit the Terminal.

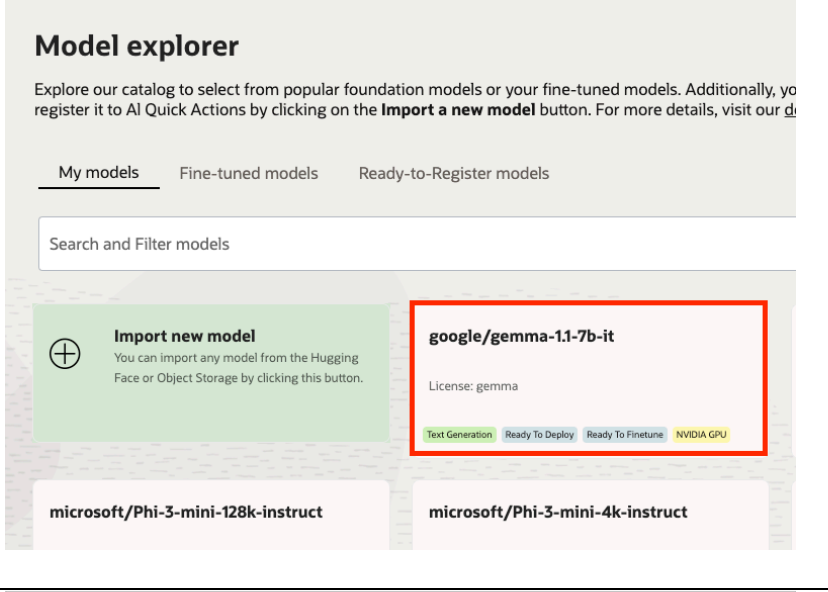
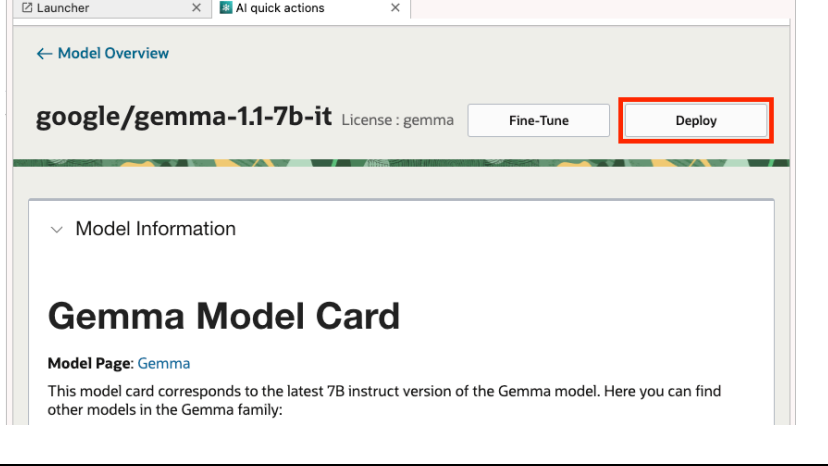
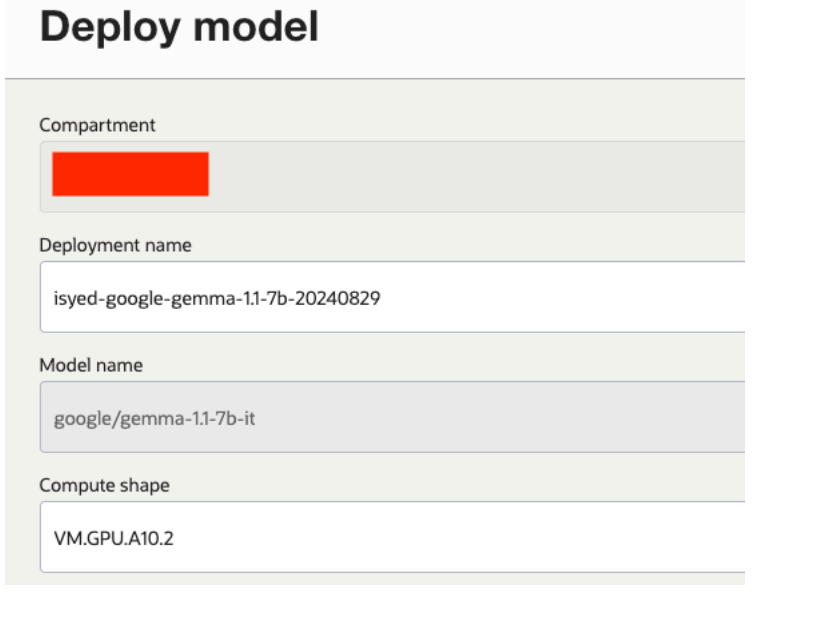
exit

```
(base) bash-4.2$
(base) bash-4.2$
(base) bash-4.2$
(base) bash-4.2$
(base) bash-4.2$ exit
```

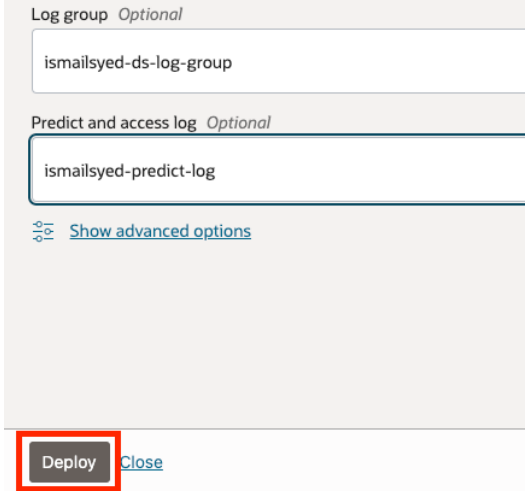
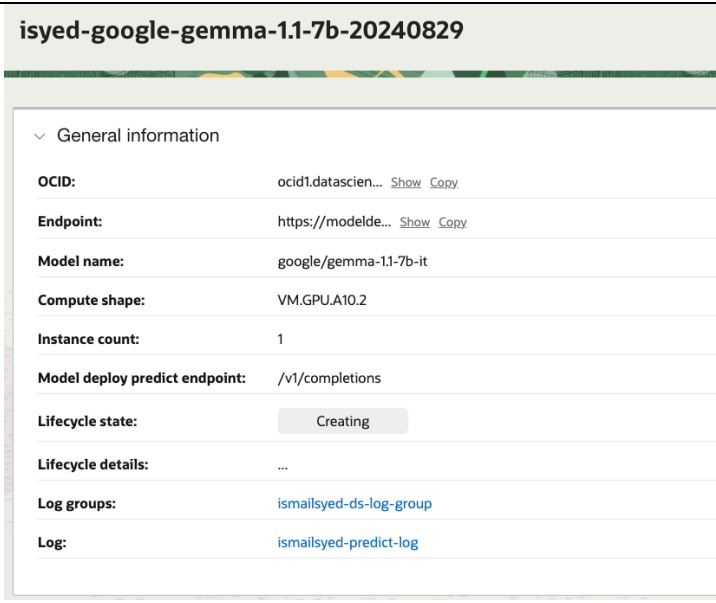
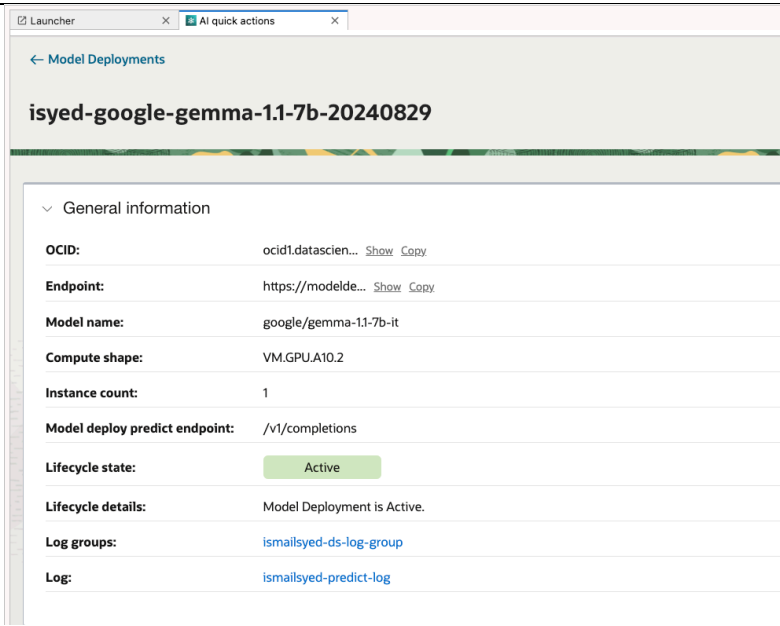
We can then check within our Object Storage Bucket we created earlier, and we should be able to see all the downloaded model artifacts.

▼ google-gemma-1.1-7b-it	-	-
▼ google	-	-
▼ gemma-1.1-7b-it	-	-
> .huggingface	-	-
> config	-	-
□ .gitattributes	Thu, Aug 29, 2024, 16:45:43 UTC	1.53 KiB
□ README.md	Thu, Aug 29, 2024, 16:45:43 UTC	26.46 KiB
□ config.json	Thu, Aug 29, 2024, 16:45:43 UTC	620 bytes
□ generation_config.json	Thu, Aug 29, 2024, 16:45:43 UTC	132 bytes
□ model-00001-of-00004.safetensors	Thu, Aug 29, 2024, 16:46:26 UTC	4.65 GiB
□ model-00002-of-00004.safetensors	Thu, Aug 29, 2024, 16:46:16 UTC	4.64 GiB
□ model-00003-of-00004.safetensors	Thu, Aug 29, 2024, 16:46:02 UTC	4.64 GiB
□ model-00004-of-00004.safetensors	Thu, Aug 29, 2024, 16:45:50 UTC	1.97 GiB
□ model.safetensors.index.json	Thu, Aug 29, 2024, 16:45:43 UTC	20.43 KiB
□ special_tokens_map.json	Thu, Aug 29, 2024, 16:45:43 UTC	636 bytes
□ tokenizer.json	Thu, Aug 29, 2024, 16:45:44 UTC	16.71 MiB
□ tokenizer.model	Thu, Aug 29, 2024, 16:45:44 UTC	4.04 MiB
□ tokenizer_config.json	Thu, Aug 29, 2024, 16:45:43 UTC	33.37 KiB



<p>Within AI Quick Actions under the Model Explorer, we should now see a new Model Card for our google/gemma-1.1-7b-it Model.</p> <p>Click on it.</p>	
<p>We can then Deploy the model.</p> <p>Click Deploy.</p>	
<p>We can then give our Deployment a Name.</p> <p>Select the Compute Shape. I have left it as the default – VM.GPU.A10.2</p>	



<p>As an optional task, I have selected a Log Group and Predict and Access Logs.</p> <p>Click on Deploy.</p>	
<p>Your deployment will then enter the Creating Lifecycle State.</p> <p>The deployment can take up to 10/15 minutes to deploy.</p>	
<p>Once the deployment is complete, the lifecycle state should be updated to Active.</p>	

If we scroll down, we then have a nice UI to test out our deployed model.

We can enter a prompt, tweak the parameters and then generate an answer.

Test your model

Test your model below. Refine the prompts and parameters to fit your use cases. View our [code samples](#) to invoke your model

Prompt

Can you describe how to make the perfect cup of tea?

Generate Clear

Response

Copy response

Step 1: Choose Your Tea

- Consider your mood, taste preferences, and the time of day.
- Different teas offer different flavors and health benefits.

Step 2: Heat the Water

- Use fresh, filtered water.
- Heat water to the recommended temperature for the specific tea you're using.
- Most teas require water between 195°F and 208°F.

Model parameters

Max tokens 500

Temperature 0.7

Top p 0.99

Top k 50

Frequency penalty 0

Presence penalty 0

Stop sequence Optional

If we head back to our OCI Data Science Project within the OCI Console.

Navigate to Models under Resources.

We can see a new entry in our Model Catalog for our Google Gemma Model.

Resources

Notebook sessions

Jobs

Pipelines

Models

Model deployments

List scope

Models in IsmailSyed Compartment

The model catalog is a centralized and managed repository of model artifacts. Models in the model catalog can also be deployed as HTTP endpoints through [Learn more about the model catalog.](#)

Create model Create model version set Download sample artifact ZIP

Model version set	Model name
-	google/gemma-1.1-7b-it

Next **navigate to your Model Deployments under Resources.**

Here we can see our Google Gemma Deployment.

Select the Model Deployment.

Resources

Notebook sessions

Jobs

Pipelines

Models

Model deployments

List scope

Compartment

Model Deployments in Isr

Create model deployment

Name	State
isyed-google-gemma-1.1-7b-20240829	Active
isyed-mistral-7b-instruct-20240814	Inactive
cust-retention-xgboost-classifier	Inactive



<p>Click on Model Deployment Endpoint under Invoking your Model.</p>	<h2>Models</h2> <p>Deployed model: google/gemma-1.1-7b-it</p> <p>Invoking your model: Model Deployment Endpoint</p>
<p>Here you will be displayed with your Model Endpoint and some sample code which can be used to invoke your model.</p> <p>Note: Authentication to the model will be done via a configuration file, security token or resource principal. This is up to you to configure.</p>	<h3>Invoking your model</h3> <p>Your model HTTP endpoint</p> <p>Copy Text</p> <p><code>https://modeldeployment</code></p> <p>Calling your model from OCI CLI</p> <p><input type="radio"/> CLI <input checked="" type="radio"/> Python SDK <input type="radio"/> Java SDK (Version 3.X.X)</p> <p>Copy Text</p> <pre># The OCI SDK must be installed for this example to function properly. # Installation instructions can be found here: https://docs.oracle.com/en-us/iaas/Content/API/SDKDocs/pytl import oci import requests from oci.signer import Signer import sseclient # install with pip install sseclient-py config = oci.config.from_file("~/oci/config") # replace with the location of your oci config file auth = Signer(tenancy=config['tenancy'], user=config['user'], fingerprint=config['fingerprint'], private_key_file_location=config['key_file'], pass_phrase=config['pass_phrase']) # For security token based authentication # token_file = config['security_token_file'] # token = None # with open(token_file, 'r') as f: # token = f.read() # private_key = oci.signer.load_private_key_from_file(config['key_file']) # auth = oci.auth.signers.SecurityTokenSigner(token, private_key)</pre>
<p>You can also run the 01-<i>invoke-deployed-gemma-model.ipynb</i> Notebook that is provided as part of this guide to connect to your model endpoint and make a request.</p> <p>This notebook assumes you have a resource principal configured.</p>	