







# What is RHEL AI?



**Foundation Model Platform**  
Seamlessly develop, deploy and test best of breed, open source Granite generative AI models to power your enterprise applications.  
**The model is the new platform.**

Update confidential designator here

-  **Open Granite models**  
Highly performant, fully open source, collaboratively developed Granite language and code models from the community, fully supported & indemnified by Red Hat and IBM.
-  **InstructLab model alignment**  
Scalable, cost-effective solution for enhancing LLM capabilities efficiently for a wide range of applications, making knowledge & skills contributions accessible to a wide range of users
-  **Optimized bootable model runtime instances**  
Granite models & InstructLab tooling packaged as a bootable RHEL image, including Pytorch/runtime libraries, hardware optimized inference for Nvidia, Intel and AMD that can run anywhere and provides onramp to OpenShift AI for scale and lifecycle & watsonx for agent integration and governance.
-  **Enterprise support, lifecycle & indemnification**  
Trusted enterprise platform, 24x7 production support, extended model lifecycle and model IP indemnification 

RHEL AI provides the ability for domain experts, not just data scientists, to contribute to a built-for-purpose gen AI model across the hybrid cloud, while also enabling IT organizations to scale these models for production through Red Hat OpenShift AI.”The [InstructLab project](#) is the upstream of RHEL AI. Red Hat Enterprise Linux AI allows you to do the following:

- Host an LLM and interact with the open source Granite family of Large Language Models (LLMs).
- Using the LAB method, create and add your own knowledge data in a Git repository and fine-tune a model with that data with minimal machine learning background.
- Interact with the model that has been fine-tuned with your data.

Red Hat Enterprise Linux AI empowers you to contribute directly to LLMs. This allows you to easily and efficiently build AI-based applications, including chatbots.

## InstructLab vs RHEL AI?

InstructLab is an open source AI project that facilitates contributions to Large Language Models (LLMs). RHEL AI takes the foundation of the InstructLab project and builds an enterprise platform for LLM integration on applications. Red Hat Enterprise Linux AI targets high performing server platforms with dedicated Graphic Processing Units (GPUs). InstructLab is intended for small scale platforms, including laptops and personal computers.

InstructLab implements the LAB (Large-scale Alignment for chatBots) technique, a novel synthetic data-based fine-tuning method for LLMs. The LAB process consists of several components:

- A taxonomy-guided synthetic data generation process
- A multi-phase training process
- A fine-tuning framework

RHEL AI and InstructLab allow you to customize an LLM with domain-specific knowledge for your distinct use cases.

## How to collect system information in RHEL AI.

The `nvidia-smi` command can be used to retrieve the information about NVIDIA GPUs on a system, such as driver version, CUDA version, GPU utilization, memory usage, temperature, and power consumption. `nvidia-smi -l <sec>` displays the status of the GPU in real-time at the provided interval (seconds).

**Note: for training, a minimum of 160GB VRAM. If a given system has < 160GB VRAM... it cannot run `ilab data generate` or `ilab model train`.**

## Maximum Context Length / Tokens

Error looks something like this, with most relevant bits highlighted:

Unset

File

```
"/opt/app-root/lib64/python3.11/site-packages/openai/_base_client.py", line 1041, in _request raise
self._make_status_error_from_response(err.response) from None
openai.BadRequestError: Error code: 400 - {'object': 'error',
'message': "This model's maximum context length is 4096 tokens.
However, you requested 4172 tokens (2124 in the messages, 2048 in
the completion). Please reduce the length of the messages or
completion.", 'type': 'BadRequestError', 'param': None, 'code':
400}
```

Check what model is being served? In this case, top of output indicated granite-7b-redhat-lab. RHEL AI 1.1 assumes Mixtral 8x7b as a teacher model. The

prompts for generating synthetic data for ilab generate assume Mixtral and are not compatible with granite-7b and merlinite-7b models. Additionally, RHEL AI SDG - SDG 1.5 - requires LoRA adapters that are specific to Mixtral and cannot fuse to any other model.

The wrong model is being used with ilab data generate. It should be Mixtral-8x7b for RHEL AI 1.x

## Model not found

Unset

File

```
"/opt/app-root/lib64/python3.11/site-packages/openai/_base_client.py", line 1041, in _request raise
self._make_status_error_from_response(err.response) from
None openai.NotFoundError: Error code: 404 - {'object':
'error', 'message': 'The model
`/var/home/cloud-user/.cache/instructlab/models/mixtral-8x7
b-instruct-v0-1` does not exist.', 'type': 'NotFoundError',
'param': None, 'code': 404}
```

The above error msg is quite generic. Validate if the model exists it should. This type of error msg

`/var/home/cloud-user/.cache/instructlab/models/mixtral-8x7b-instruct-v0-1` does not exist may popup if multiple instance of `vllm` is running.

Additionally, if the system has < 160GB VRAM and the generate command was trying to connect to a Mixtral endpoint that didn't exist because vllm probably errored out with an OOM message then you can also notice model does not exist.

Failed to invoke skopeo proxy method OpenImage: remote error: unable to retrieve auth token

Unset

```
$ sudo bootc upgrade ERROR Upgrading: Pulling: Creating importer: Failed to invoke skopeo proxy method OpenImage: remote error: unable to retrieve auth token: invalid username/password: unauthorized: Please login to the Red Hat Registry using your Customer Portal credentials. Further instructions can be found here: https://access.redhat.com/RegistryAuthentication
```

The bootc credential file must be empty. Copy the /run/containers/0/auth.json to /etc/ostree/.

Unset

```
$ sudo podman login registry.redhat.io
[cloud-user@ip-10-31-10-4 ~]$ sudo cp /run/containers/0/auth.json /etc/ostree/
[cloud-user@ip-10-31-10-4 ~]$ [cloud-user@ip-10-31-10-4 ~]$
[cloud-user@ip-10-31-10-4 ~]$ sudo bootc upgrade No changes in registry.redhat.io/rhelai1/bootc-nvidia-rhel9:1.1 => sha256:0618bf4feab4e612bd16470af2be4d3587143602ebb8bea3772dfca9cff6f4a2 No update available. [cloud-user@ip-10-31-10-4 ~]$
```

## CUDA out of memory

Check with `nvidia-smi` if enough resources are there in the system. Validate the PATH using `export PATH=$PATH:/usr/local/cuda/bin`

out of memory mostly got triggered when system is using less than 100GB of VRAM. To serve Mixtral you need around 100GB of VRAM.

Failed to determine backend: Cannot determine which backend to use: Failed to determine whether the model is a GGUF format: [Errno 2] No such file or directory

Unset

```
[instruct@bastion ~]$ ilab model serve Failed to determine
backend: Cannot determine which backend to use: Failed to
determine whether the model is a GGUF format: [Errno 2] No
such file or directory:
'/var/home/instruct/.cache/instructlab/models/granite-7b-re
dhat-lab'
```

Execute `ilab config show` and check which model is defined under the serving section. By default `granite-7b-redhat-lab` is used for serving `model_path:` `/var/home/instruct/.cache/instructlab/models/granite-7b-redhat-lab` if `granite-7b-redhat-lab` model is not available then download the model. All available model can be viewed by running `ilab model list`. Different model also can be used for serving by using `ilab model serve --model-path`