Red Hat Enterprise Linux AI 1.1

Building your RHEL AI environment

Building and initalizing RHEL AI

Last Updated: 2024-10-17

Red Hat Enterprise Linux AI 1.1 Building your RHEL AI environment

Building and initalizing RHEL AI

Legal Notice

Abstract

This document provides instructions on how to initalize and set up the RHEL AI environment

Table of Contents

CHAPTER 1. CONFIGURING ACCOUNTS FOR RHEL AI

There are a few accounts you need to set up before interacting with RHEL AI.

Creating a Red Hat account

You can create a Red Hat account by registering on the Red Hat website. You can follow the procedure in Register for a Red Hat account .

Creating a Red Hat registry account

Before you can download models from the Red Hat registry, you need to create a registry account and login using the CLI. You can view your account username and password by selecting the **Regenerate Token** button on the webpage.

1. You can create a Red Hat registry account by selecting the **New Service Account** button on the Registry Service Accounts page.
2. There are several ways you can log into your registry account via the CLI. Follow the procedure in Red Hat Container Registry authentication to login on your machine.

Optional: Configuring Red Hat Insights for hybrid cloud deployments

Red Hat Insights is an offering that gives you visibility to the environments you are deploying. This platform can also help identify operational and vulnerability risks in your system. For more information about Red Hat Insights, see Red Hat Insights data and application security .

- You can create a Red Hat Insights account using an activation key and organization parameters by following the procedure in Viewing an activation key.
  You can then configure your account on your machine by running the following command:

  ```
  $ rhc connect --organization <org id> --activation-key <created key>
  ```

  To run RHEL AI in a disconnected environment, or opt out of Red Hat Insights, run the following commands:

  ```
  $ sudo mkdir -p /etc/ilab
  $ sudo touch /etc/ilab/insights-opt-out
  ```

CHAPTER 2. INITIALIZING INSTRUCTLAB

You must initialize the InstructLab environments to begin working with the Red Hat Enterprise Linux AI models.

2.1. CREATING YOUR RHEL AI ENVIRONMENT

You can start interacting with LLM models and the RHEL AI tooling by initializing the InstructLab environment.

Prerequisites

- You installed RHEL AI with the bootable container image.
- You have root user access on your machine.

Procedure

1. Optional: To set up training profiles, you need to know the GPU accelerators in your machine. You can view your system information by running the following command:

```
$ ilab system info
```

2. Initialize InstructLab by running the following command:

```
$ ilab config init
```

3. The CLI prompts you to setup your **config.yaml**.

Example output

```
Welcome to InstructLab CLI. This guide will help you to setup your environment.
Please provide the following values to initiate the
environment [press Enter for defaults]:
Generating `/home/<example-user>/.config/instructlab/config.yaml` and `/home/<example-
user>/.local/share/instructlab/internal/train_configuration/profiles`...
```

4. Follow the CLI prompts to set up your training hardware configurations. This updates your **config.yaml** file and adds the proper **train** configurations for training an LLM model. Type the number of the YAML file that matches your hardware specifications.

IMPORTANT

These profiles only add the necessary configurations to the **train** section of your **config.yaml** file, therefore any profile can be selected for inference serving a model.

Example output of selecting training profiles

```
Please choose a train profile to use:
[0] No profile (CPU-only)
[1] A100_H100_x2.yaml
```

```
[2] A100_H100_x4.yaml
[3] A100_H100_x8.yaml
[4] L40_x4.yaml
[5] L40_x8.yaml
[6] L4_x8.yaml
Enter the number of your choice [hit enter for the default CPU-only profile] [0]:
```

Example output of a completed ilab config init run.

```
You selected: A100_H100_x8.yaml
Initialization completed successfully, you're ready to start using `ilab`. Enjoy!
```

5. **Configuring your system's GPU for inference serving:**This step is only required if you are using Red Hat Enterprise Linux AI exclusively for inference serving.

    a. Edit your **config.yaml** file by running the following command:

    ```
    $ ilab config edit
    ```

    b. In the **evaluate** section of the configurations file, edit the **gpus:** parameter and add the number of accelerators on your machine.

    ```
    evaluate:
      base_branch: null
      base_model: ~/.cache/instructlab/models/granite-7b-starter
      branch: null
      gpus: <num-gpus>
    ```

    c. In the **vllm** section of the **serve** field in the configuration file, edit the **gpus:** and **vllm_args: ["--tensor-parallel-size"]** parameters and add the number of accelerators on your machine.

    ```
    serve:
      backend: vllm
      chat_template: auto
      host_port: 127.0.0.1:8000
      llama_cpp:
        gpu_layers: -1
        llm_family: ''
        max_ctx_size: 4096
      model_path: ~/.cache/instructlab/models/granite-7b-redhat-lab
      vllm:
        llm_family: ''
        vllm_args: ["--tensor-parallel-size", "<num-gpus>"]
        gpus: <num-gpus>
    ```

6. If you want to use the skeleton taxonomy tree, which includes two skills and one knowledge **qna.yaml** file, you can clone the skeleton repository and place it in the **taxonomy** directory by running the following command:

```
rm -rf ~/.local/share/instructlab/taxonomy/ ; git clone https://github.com/RedHatOfficial/rhelai-
sample-taxonomy.git ~/.local/share/instructlab/taxonomy/
```

Directory structure of the InstructLab environment

```
├─ ~/.cache/instructlab/models/ 1
├─ ~/.local/share/instructlab/datasets 2
├─ ~/.local/share/instructlab/taxonomy 3
├─ ~/.local/share/instructlab/phased/<phase1-or-phase2>/checkpoints/ 4
```

1 **~/.cache/instructlab/models/**: Contains all downloaded large language models, including the saved output of ones you generate with RHEL AI.

2 **~/.local/share/instructlab/datasets/**: Contains data output from the SDG phase, built on modifications to the taxonomy repository.

3 **~/.local/share/instructlab/taxonomy/**: Contains the skill and knowledge data.

4 **~/.local/share/instructlab/phased/<phase1-or-phase2>/checkpoints/**: Contains the output of the multi-phase training process

Verification

1. You can view the full **config.yaml** file by running the following command

```
$ ilab config show
```

2. You can also manually edit the **config.yaml** file by running the following command:

```
$ ilab config edit
```

CHAPTER 3. DOWNLOADING MODELS

Red Hat Enterprise Linux AI allows you to customize or chat with various Large Language Models (LLMs) provided and built by Red Hat and IBM. You can download these models from the Red Hat RHEL AI registry.

Table 3.1. Red Hat Enterprise Linux AI LLMs

Large Language Models (LLMs)	Type	Size	Purpose	Support
granite-7b-starter	Base model	12.6 GB	Base model for customizing, training and fine-tuning	General availability
granite-7b-redhat-lab	LAB fine-tuned granite model	12.6 GB	Granite model for serving and inferencing	General availability
granite-8b-code-instruct	LAB fine-tuned granite code model	15.0 GB	LAB fine-tuned granite code model for serving and inferencing	Technology preview
granite-8b-code-base	Granite fine-tuned code model	15.0 GB	Granite code model for serving and inferencing	Technology preview
mixtral-8x7b-instruct-v0-1	Teacher/critic model	87.0 GB	Teacher and critic model for running Synthetic data generation (SDG)	General availability
prometheus-8x7b-v2-0	Judge model	87.0 GB	Judge model for multi-phase training and evaluation	General availability

IMPORTANT

Using the `granite-8b-code-instruct` and `granite-8b-code-base` Large Language models (LLMS) is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see Technology Preview Features Support Scope .

Models required for customizing the Granite LLM

- The **granite-7b-starter** base LLM.
- The **mixtral-8x7b-instruct-v0-1** teacher model for SDG.
- The **prometheus-8x7b-v2-0** judge model for training and evaluation.

Additional tools required for customizing an LLM

- The **skills-adapter-v3** LoRA layered skills adapter for SDG.
- The **knowledge-adapter-v3** LoRA layered knowledge adapter for SDG.

IMPORTANT

The LoRA layered adapters do not show up in the output of the **ilab model list** command. You can see the **skills-adapter-v3** and **knowledge-adapter-v3** files in the **ls ~/.cache/instructlab/models** folder.

NOTE

The listed granite models for serving and inferencing are not currently supported for customizing.

3.1. DOWNLOADING THE MODELS FROM A RED HAT REPOSITORY

You can download the additional optional models created by Red Hat and IBM.

Prerequisites

- You installed RHEL AI with the bootable container image.
- You initialized InstructLab.
- You created a Red Hat registry account and logged in on your machine.
- You have root user access on your machine.

Procedure

1. To download the additional LLM models, run the following command:

```
$ ilab model download --repository docker://<repository_and_model> --release 1.1
```

where:

<repository_and_model>

Specifies the repository location of the model as well as the model. You can access the models from the **registry.redhat.io/rhelai1/** repository.

<release>

Specifies the version of the model. Set to **1.1** for the models that are supported on RHEL AI version 1.1.

Example command

```
$ ilab model download --repository docker://registry.redhat.io/rhelai1/granite-7b-starter --
release latest
```

Verification

1. You can view all the downloaded models, including the new models after training, on your system with the following command:

   ```
   $ ilab model list
   ```

   #### Example output

   ```
   +-----------------------------------+---------------------+---------+
   | Model Name                        | Last Modified       | Size    |
   +-----------------------------------+---------------------+---------+
   | models/prometheus-8x7b-v2-0       | 2024-08-09 13:28:50 |  87.0 GB|
   | models/mixtral-8x7b-instruct-v0-1 | 2024-08-09 13:28:24 |  87.0 GB|
   | models/granite-7b-redhat-lab      | 2024-08-09 14:28:40 |  12.6 GB|
   | models/granite-7b-starter         | 2024-08-09 14:40:35 |  12.6 GB|
   +-----------------------------------+---------------------+---------+
   ```

2. You can also list the downloaded models in the **ls ~/.cache/instructlab/models** folder by running the following command:

   ```
   $ ls ~/.cache/instructlab/models
   ```

   #### Example output

   ```
   granite-7b-starter
   granite-7b-redhat-lab
   ```

CHAPTER 4. SERVING AND CHATTING WITH THE MODELS

To interact with various models on Red Hat Enterprise Linux AI you must serve the model, which hosts it on a server, then you can chat with the models.

4.1. SERVING THE MODEL

To interact with the models, you must first activate the model in a machine through serving. The **ilab model serve** commands starts a vLLM server that allows you to chat with the model.

Prerequisites

- You installed RHEL AI with the bootable container image.
- You initialized InstructLab.
- You installed your preferred Granite LLMs.
- You have root user access on your machine.

Procedure

1. If you do not specify a model, you can serve the default model, **granite-7b-redhat-lab**, by running the following command:

   ```
   $ ilab model serve
   ```

2. To serve a specific model, run the following command

   ```
   $ ilab model serve --model-path <model-path>
   ```

   Example command

   ```
   $ ilab model serve --model-path ~/.cache/instructlab/models/granite-7b-code-instruct
   ```

   Example output of when the model is served and ready

   ```
   INFO 2024-03-02 02:21:11,352 lab.py:201 Using model 'models/granite-7b-code-instruct'
   with -1 gpu-layers and 4096 max context size.
   Starting server process
   After application startup complete see http://127.0.0.1:8000/docs for API.
   Press CTRL+C to shut down the server.
   ```

4.1.1. Optional: Running ilab model serve as a service

You can set up a **systemd** service so that the **ilab model serve** command runs as a running service. The **systemd** service runs the **ilab model serve** command in the background and restarts if it crashes or fails. You can configure the service to start upon system boot.

Prerequisites

- You installed the Red Hat Enterprise Linux AI image on bare metal.

- You initialized InstructLab
- You downloaded your preferred Granite LLMs.
- You have root user access on your machine.

Procedure.

1. Create a directory for your **systemd** user service by running the following command:

   ```
   $ mkdir -p $HOME/.config/systemd/user
   ```

2. Create your **systemd** service file with the following example configurations:

   ```
   $ cat << EOF > $HOME/.config/systemd/user/ilab-serve.service
   [Unit]
   Description=ilab model serve service

   [Install]
   WantedBy=multi-user.target default.target 1

   [Service]
   ExecStart=ilab model serve --model-family granite
   Restart=always
   EOF
   ```

   1 Specifies to start by default on boot.

3. Reload the **systemd** manager configuration by running the following command:

   ```
   $ systemctl --user daemon-reload
   ```

4. Start the **ilab model serve systemd** service by running the following command:

   ```
   $ systemctl --user start ilab-serve.service
   ```

5. You can check that the service is running with the following command:

   ```
   $ systemctl --user status ilab-serve.service
   ```

6. You can check the service logs by running the following command:

   ```
   $ journalctl --user-unit ilab-serve.service
   ```

7. To allow the service to start on boot, run the following command:

   ```
   $ sudo loginctl enable-linger
   ```

8. Optional: There are a few optional commands you can run for maintaining your **systemd** service.

   - You can stop the ilab-serve system service by running the following command:

```
$ systemctl --user stop ilab-serve.service
```

- You can prevent the service from starting on boot by removing the **"WantedBy=multi-user.target default.target"** from the **$HOME/.config/systemd/user/ilab-serve.service** file.

4.2. CHATTING WITH THE MODEL

Once you serve your model, you can now chat with the model.

IMPORTANT

The model you are chatting with must match the model you are serving. With the default **config.yaml** file, the **granite-7b-redhat-lab** model is the default for serving and chatting.

Prerequisites

- You installed RHEL AI with the bootable container image.
- You initialized InstructLab.
- You downloaded your preferred Granite LLMs.
- You are serving a model.
- You have root user access on your machine.

Procedure

1. Since you are serving the model in one terminal window, you must open another terminal to chat with the model.
2. To chat with the default model, run the following command:

   ```
   $ ilab model chat
   ```

3. To chat with a specific model run the following command:

   ```
   $ ilab model chat --model <model-path>
   ```

   Example command

   ```
   $ ilab model chat --model ~/.cache/instructlab/models/granite-7b-code-instruct
   ```

Example output of the chatbot

```
$ ilab model chat
╭──────────────────────────────────────────────────────────────────────────────────
─────────────────────────────────────────────────────────────────── system
──────────────────────────────────────────────────────────────────────────────────
───────────────────────────────────────────────────────────────────╮
│ Welcome to InstructLab Chat w/ GRANITE-7B-CODE-INSTRUCT (type /h for help)
│
```

```
╰──────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────────────
───────────────────────────────────────────────────────────────╯
>>>
[S][default]
```

+ Type **exit** to leave the chatbot.

4.2.1. Optional: Creating an API key for model chatting

By default, the **ilab** CLI does not use authentication. If you want to expose your server to the internet, you can create a API key that connects to your server with the following procedures.

Prerequisites

- You installed the Red Hat Enterprise Linux AI image on bare metal.
- You initialized InstructLab
- You downloaded your preferred Granite LLMs.
- You have root user access on your machine.

Procedure

1. Create a API key that is held in **$VLLM_API_KEY** parameter by running the following command:

   ```
   $ export VLLM_API_KEY=$(python -c 'import secrets; print(secrets.token_urlsafe())')
   ```

2. You can view the API key by running the following command:

   ```
   $ echo $VLLM_API_KEY
   ```

3. Update the **config.yaml** by running the following command:

   ```
   $ ilab config edit
   ```

4. Add the following parameters to the **vllm_args** section of your **config.yaml** file.

   ```
   serve:
     vllm:
       vllm_args:
       - --api-key
       - <api-key-string>
   ```

   where

   <api-key-string>
   Specify your API key string.

5. You can verify that the server is using API key authentication by running the following command:

```
$ ilab model chat
```

Then, seeing the following error that shows an unauthorized user.

```
openai.AuthenticationError: Error code: 401 - {'error': 'Unauthorized'}
```

6. Verify that your API key is working by running the following command:

```
$ ilab chat -m granite-7b-redhat-lab --endpoint-url https://inference.rhelai.com/v1 --api-key
$VLLM_API_KEY
```

Example output

```
$ ilab model chat
╭────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────────────────
system
────────────────────────────────────────────────────────────────────────
───────────────────────────────────────────────────────────────────────╮
│ Welcome to InstructLab Chat w/ GRANITE-7B-LAB (type /h for help)
│
╰───────────────────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────────────────
──────╯
>>>
[S][default]
```

4.2.2. Optional: Allowing chat access to a model from a secure endpoint

You can serve an inference endpoint and allow others to interact with models provided with Red Hat Enterprise Linux AI on secure connections by creating a **systemd** service and setting up a nginx reverse proxy that exposes a secure endpoint. This allows you to share the secure endpoint with others so they can chat with the model over a network.

The following procedure uses self-signed certifications, but it is recommended to use certificates issued by a trusted Certificate Authority (CA).

NOTE

The following procedure is supported only on bare metal platforms.

Prerequisites

- You installed the Red Hat Enterprise Linux AI image on bare-metal.
- You initialized InstructLab
- You downloaded your preferred Granite LLMs.
- You have root user access on your machine.

Procedure

Procedure

1. Create a directory for your certificate file and key by running the following command:

```
$ mkdir -p `pwd`/nginx/ssl/
```

2. Create an OpenSSL configuration file with the proper configurations by running the following command:

```
$ cat > openssl.cnf <<EOL
[ req ]
default_bits = 2048
distinguished_name = <req-distinguished-name> 1
x509_extensions = v3_req
prompt = no

[ req_distinguished_name ]
C  = US
ST = California
L  = San Francisco
O  = My Company
OU = My Division
CN = rhelai.redhat.com

[ v3_req ]
subjectAltName = <alt-names> 2
basicConstraints = critical, CA:true
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer

[ alt_names ]
DNS.1 = rhelai.redhat.com 3
DNS.2 = www.rhelai.redhat.com 4
```

1 Specify the distinguished name for your requirements.

2 Specify the alternate name for your requirements.

3 4 Specify the server common name for RHEL AI. In the example, the server name is **rhelai.redhat.com**.

3. Generate a self signed certificate with a Subject Alternative Name (SAN) enabled with the following commands:

```
$ openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
`pwd`/nginx/ssl/rhelai.redhat.com.key -out `pwd`/nginx/ssl/rhelai.redhat.com.crt -config
openssl.cnf
```

```
$ openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
```

4. Create the Nginx Configuration file and add it to the `**pwd**/nginx/conf.d` by running the following command:

```
mkdir -p `pwd`/nginx/conf.d
```

```
echo 'server {
    listen 8443 ssl;
    server_name <rhelai.redhat.com> ①

    ssl_certificate /etc/nginx/ssl/rhelai.redhat.com.crt;
    ssl_certificate_key /etc/nginx/ssl/rhelai.redhat.com.key;

    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
' > `pwd`/nginx/conf.d/rhelai.redhat.com.conf
```

① Specify the name of your server. In the example, the server name is **rhelai.redhat.com**

5. Run the Nginx container with the new configurations by running the following command:

   ```
   $ podman run --net host -v `pwd`/nginx/conf.d:/etc/nginx/conf.d:ro,Z -v
   `pwd`/nginx/ssl:/etc/nginx/ssl:ro,Z nginx
   ```

   If you want to use port 443, you must run the **podman run** command as a root user..

6. You can now connect to a serving ilab machine using a secure endpoint URL. Example command:

   ```
   $ ilab chat -m /instructlab/instructlab/granite-7b-redhat-lab --endpoint-url
   https://rhelai.redhat.com:8443/v1
   ```

7. Optional: You can also get the server certificate and append it to the Certifi CA Bundle

   a. Get the server certificate by running the following command:

   ```
   $ openssl s_client -connect rhelai.redhat.com:8443 </dev/null 2>/dev/null | openssl x509
   -outform PEM > server.crt
   ```

   b. Copy the certificate to you system's trusted CA storage directory and update the CA trust store with the following commands:

   ```
   $ sudo cp server.crt /etc/pki/ca-trust/source/anchors/
   ```

   ```
   $ sudo update-ca-trust
   ```

   c. You can append your certificate to the Certifi CA bundle by running the following command:

   ```
   $ cat server.crt >> $(python -m certifi)
   ```

   d. You can now run **ilab model chat** with a self-signed certificate. Example command:

```
$ ilab chat -m /instructlab/instructlab/granite-7b-redhat-lab --endpoint-url
https://rhelai.redhat.com:8443/v1
```