Red Hat Enterprise Linux AI 1.2

Installing

Installation documentation on various platforms

Last Updated: 2024-11-14

Red Hat Enterprise Linux AI 1.2 Installing

Installation documentation on various platforms

Legal Notice

Abstract

This document provides instructions for installing RHEL AI on machines with the bootable container image.

Table of Contents

CHAPTER 1. INSTALLATION OVERVIEW

Red Hat Enterprise Linux AI is distributed and installable as a bootable image. This bootable image includes a container that hold various software and tools for RHEL AI. Each image is compiled to support specific hardware vendors. Each RHEL AI image includes:

- **Red Hat Enterprise Linux 9.4:**A RHEL version 9.4 operating system (OS) for your machine.
- **The InstructLab container:** Contains InstructLab and various other tools required for RHEL AI. This includes:
  - Python version 3.11: A Python 3.11 installation used internally by InstructLab.
  - The InstructLab tools:
    - The InstructLab command line interface (CLI).
    - The LAB enhanced method of synthetic data generation (SDG).
    - The LAB enhanced method of single and multi-phase training.
  - InstructLab with vLLM: A high-input inference and serving engine for Large Language models (LLMs).
  - InstructLab with DeepSpeed: A hardware optimization software that speeds up the training process. Similar functionalities of FSDP.
  - InstructLab with FSDP: A training framework that makes training faster and more efficient. Similar functionalities of DeepSpeed

Red Hat Enterprise Linux AI version 1.2 also includes a sample taxonomy tree with example skills and knowledge that you can download and use for training a model.

Current installation options for Red Hat Enterprise Linux AI

- Installing on bare metal
- Installing on AWS
- Installing on IBM Cloud
- Installing on GCP (technology preview)
- Installing on Azure (technology preview)

After installation with Red Hat Enterprise Linux AI general availability, you can manually download open source Granite LLMs that you can chat and interact with. For more information about downloading these models, see Downloading additional models.

CHAPTER 2. INSTALLING RHEL AI ON BARE METAL

For installing Red Hat Enterprise Linux AI on bare metal, you can use various methods provided in the following procedure to boot and deploy your machine and start interacting with Red Hat Enterprise Linux AI.

2.1. DEPLOYING RHEL AI ON BARE METAL

You can deploy Red Hat Enterprise Linux AI with the RHEL AI ISO image in the following ways: * Kickstart * RHEL Graphical User Interface (GUI)

This image is bootable on various hardware accelerators. For more information about supported hardware, see "Red Hat Enterprise Linux AI hardware requirements" in "Getting Started"

Prerequisites

- You have downloaded the Red Hat Enterprise Linux AI ISO image from https://access.redhat.com/.

IMPORTANT

Red Hat Enterprise Linux AI requires additional storage for the RHEL AI data as well as the update of image-mode Red Hat Enterprise Linux. The default location for the InstructLab data is in the **home/<user>** directory. The minimum recommendation for data storage in the **/home** directory is 1 TB. During updates, the **bootc** command needs extra space to store temporary data. The minimum storage recommendation for the / path is 120 GB. You need to consider your machine's storage when partitioning the schemes of your disks.

Procedure

- Interactive GUI
  You can use the interactive Red Hat Enterprise Linux graphical installer and the RHEL AI ISO image to deploy RHEL AI on your machine. For more information about booting RHEL using an ISO file using the GUI, see the Interactively installing RHEL from installation media .

- Kickstart with embedded container image

  1. You can customize the RHEL AI installation by using your own Kickstart file.

     a. Create your own Kickstart file with your preferred parameters. For more information about creating Kickstart files, see the Creating Kickstart files in the RHEL documentation.

     Sample Kickstart file for RHEL AI called rhelai-bootc.ks

```
# use the embedded container image
ostreecontainer --url=/run/install/repo/container --transport=oci --no-signature-verification

# switch bootc to point to Red Hat container image for upgrades
%post
bootc switch --mutate-in-place --transport registry registry.redhat.io/rhelai1/bootc-nvidia-rhel9:1.1
touch /etc/cloud/cloud-init.disabled
```

```
%end

## user customizations follow

# customize this for your target system network environment
network --bootproto=dhcp --device=link --activate

# customize this for your target system desired disk partitioning
clearpart --all --initlabel --disklabel=gpt
reqpart --add-boot
part / --grow --fstype xfs

# services can also be customized via Kickstart
firewall --disabled
services --enabled=sshd

# optionally add a user
user --name=cloud-user --groups=wheel --plaintext --password <password>
sshkey --username cloud-user "ssh-ed25519 AAAAC3Nza....."

# if desired, inject an SSH key for root
rootpw --iscrypted locked
sshkey --username root "ssh-ed25519 AAAAC3Nza..."
reboot
```

The sample Kickstart uses the embedded container image in the ISO file, signaled by the **ostreecontainer** command with the **--url=/run/install/repo/container** parameter. The **bootc switch** parameter points to the Red Hat registry for future updates and then you can add your own customizations.

b. You need to embed the Kickstart into the RHEL AI ISO so your machine can restart and deploy RHEL AI. In the following example, **rhelai-bootc.ks** is the name of the Kickstart file you're embedding into the boot ISO. The **mkksiso** utility is found in the **lorax** rpm package.

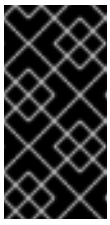
```
$ mkksiso rhelai-bootc.ks <downloaded-iso-image> rhelai-bootc-ks.iso
```

where

<downloaded-iso-image>

Specify the ISO image you downloaded from access.redhat.com.

c. You can then boot your machine using this boot ISO and the installation starts automatically. After the installation is complete, the host reboots and you can login to the new system using the credentials used in the Kickstart file.



IMPORTANT

Be aware that having a custom Kickstart in your ISO will automatically start the installation, and disk partitioning, without prompting the user. Based on configuration, the local storage may be completely wiped or overwritten.

- Kickstart with custom container image

You can customize a Kickstart file with your preferred parameters to boot Red Hat Enterprise Linux AI on your machine

1. Create your own Kickstart file with your preferred parameters. For more information on creating Kickstart files, see the Creating Kickstart files in the RHEL documentation.

   Sample Kickstart file for RHEL AI called rhelai-bootc.ks

   ```
   # customize this for your target system network environment
   network --bootproto=dhcp --device=link --activate

   # customize this for your target system desired disk partitioning
   clearpart --all --initlabel --disklabel=gpt
   reqpart --add-boot
   part / --grow --fstype xfs

   # customize this to include your own bootc container
   ostreecontainer --url quay.io/<your-user-name>/nvidia-bootc:latest

   # services can also be customized via Kickstart
   firewall --disabled
   services --enabled=sshd

   # optionally add a user
   user --name=cloud-user --groups=wheel --plaintext --password <password>
   sshkey --username cloud-user "ssh-ed25519 AAAAC3Nza....."

   # if desired, inject an SSH key for root
   rootpw --iscrypted locked
   sshkey --username root "ssh-ed25519 AAAAC3Nza..."
   reboot
   ```

2. You need to embed the Kickstart into the RHEL AI ISO so your machine can restart and deploy RHEL AI. In the following example, **rhelai-bootc.ks** is the name of the Kickstart file you're embedding into the boot ISO. The **mkksiso** utility is found in the **lorax** rpm package.

   ```
   $ mkksiso rhelai-bootc.ks <downloaded-iso-image> rhelai-bootc-ks.iso
   ```

   where

   <downloaded-iso-image>

   Specify the ISO image you downloaded from access.redhat.com.

3. You can then boot your machine using this boot ISO and the installation starts automatically. After the installation is complete, the host reboots and you can login to the new system using the credentials used in the Kickstart file.

IMPORTANT

Be aware that having a custom Kickstart in your ISO will automatically start the installation, and disk partitioning, without prompting the user. Based on configuration, the local storage may be completely wiped or overwritten.

Verification

- To verify that your Red Hat Enterprise Linux AI tools installed correctly, you need to run the **ilab** command:

```
$ ilab
```

Example output

```
$ ilab
Usage: ilab [OPTIONS] COMMAND [ARGS]...

  CLI for interacting with InstructLab.

  If this is your first time running ilab, it's best to start with `ilab
  config init` to create the environment.

Options:
  --config PATH  Path to a configuration file.  [default:
                 /home/auser/.config/instructlab/config.yaml]
  -v, --verbose  Enable debug logging (repeat for even more verbosity)
  --version      Show the version and exit.
  --help         Show this message and exit.

Commands:
  config    Command Group for Interacting with the Config of InstructLab.
  data      Command Group for Interacting with the Data generated by...
  model     Command Group for Interacting with the Models in InstructLab.
  system    Command group for all system-related command calls
  taxonomy  Command Group for Interacting with the Taxonomy of InstructLab.

Aliases:
  chat      model chat
  convert   model convert
  diff      taxonomy diff
  download  model download
  evaluate  model evaluate
  generate  data generate
  init      config init
  list      model list
  serve     model serve
  sysinfo   system info
  test      model test
  train     model train
```

CHAPTER 3. INSTALLING RHEL AI ON AWS

To install and deploy Red Hat Enterprise Linux AI on AWS, you must first convert the RHEL AI image into an Amazon Machine Image (AMI). In this process, you create the following resources:

- An S3 bucket with the RHEL AI image
- AWS EC2 snapshots
- An AWS AMI
- An AWS instance

3.1. CONVERTING THE RHEL AI IMAGE TO AN AWS AMI

Before deploying RHEL AI on an AWS machine, you must set up a S3 bucket and convert the RHEL AI image to a AWS AMI.

Prerequisites

- You have an Access Key ID configured in the AWS IAM account manager.

Procedure

1. Install the AWS command-line tool by following the AWS documentation
2. You need to create a S3 bucket and set the permissions to allow image file conversion to AWS snapshots.
   a. Create the necessary environment variables by running the following commands:

```
$ export BUCKET=<custom_bucket_name>
$ export RAW_AMI=nvidia-bootc.ami
$ export AMI_NAME="rhel-ai"
$ export DEFAULT_VOLUME_SIZE=1000
```

NOTE

On AWS, the **DEFAULT_VOLUME_SIZE** is measured GBs.

   b. You can create an S3 bucket by running the following command:

```
$ aws s3 mb s3://$BUCKET
```

   c. You must create a **trust-policy.json** file with the necessary configurations for generating a S3 role for your bucket:

```
$ printf '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": {
"Service": "vmie.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition": {
"StringEquals":{ "sts:Externalid": "vmimport" } } } ] }' > trust-policy.json
```

   d. Create an S3 role for your bucket that you can name. In the following example command, **vmiport** is the name of the role.

```
$ aws iam create-role --role-name vmimport --assume-role-policy-document file://trust-policy.json
```

e. You must create a **role-policy.json** file with the necessary configurations for generating a policy for your bucket:

```
$ printf '{ "Version":"2012-10-17", "Statement":[ { "Effect":"Allow", "Action":[ "s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket" ], "Resource":[ "arn:aws:s3:::%s", "arn:aws:s3:::%s/*" ] }, { "Effect":"Allow", "Action":[ "ec2:ModifySnapshotAttribute", "ec2:CopySnapshot", "ec2:RegisterImage", "ec2:Describe*" ], "Resource":"*" } ] }' $BUCKET $BUCKET > role-policy.json
```

f. Create a policy for your bucket by running the following command:

```
$ aws iam put-role-policy --role-name vmimport --policy-name vmimport-$BUCKET --policy-document file://role-policy.json
```

3. Now that your S3 bucket is set up, you need to download the RAW image from Red Hat Enterprise Linux AI download page

4. Copy the RAW image link and add it to the following command:

```
$ curl -Lo disk.raw <link-to-raw-file>
```

5. Upload the image to the S3 bucket with the following command:

```
$ aws s3 cp disk.raw s3://$BUCKET/$RAW_AMI
```

6. Convert the image to a snapshot and store it in the **task_id** variable name by running the following commands:

```
$ printf '{ "Description": "my-image", "Format": "raw", "UserBucket": { "S3Bucket": "%s", "S3Key": "%s" } }' $BUCKET $RAW_AMI > containers.json
```

```
$ task_id=$(aws ec2 import-snapshot --disk-container file://containers.json | jq -r .ImportTaskId)
```

7. You can check the progress of the disk image to snapshot conversion job with the following command:

```
$ aws ec2 describe-import-snapshot-tasks --filters Name=task-state,Values=active
```

8. Once the conversion job is complete, you can get the snapshot ID and store it in a variable called **snapshot_id** by running the following command:

```
$ snapshot_id=$(aws ec2 describe-snapshots | jq -r '.Snapshots[] | select(.Description | contains("${task_id}")) | .SnapshotId')
```

9. Add a tag name to the snapshot, so it's easier to identify, by running the following command:

```
$ aws ec2 create-tags --resources $snapshot_id --tags Key=Name,Value="$AMI_NAME"
```

10. Register an AMI from the snapshot with the following command:

```
$ ami_id=$(aws ec2 register-image  \
   --name "$AMI_NAME" \
   --description "$AMI_NAME" \
   --architecture x86_64 \
   --root-device-name /dev/sda1 \
   --block-device-mappings "DeviceName=/dev/sda1,Ebs=
{VolumeSize=${DEFAULT_VOLUME_SIZE},SnapshotId=${snapshot_id}}" \
   --virtualization-type hvm \
   --ena-support \
   | jq -r .ImageId)
```

11. You can add another tag name to identify the AMI by running the following command:

```
$ aws ec2 create-tags --resources $ami_id --tags Key=Name,Value="$AMI_NAME"
```

3.2. DEPLOYING YOUR INSTANCE ON AWS USING THE CLI

You can launch the AWS instance with your new RHEL AI AMI from the AWS web console or the CLI. You can use whichever method of deployment you want to launch your instance. The following procedure displays how you can use the CLI to launch your AWS instance with the custom AMI.

If you choose to use the CLI as a deployment option, there are several configurations you have to create, as shown in "Prerequisites".

Prerequisites

- You created your RHEL AI AMI. For more information, see "Converting the RHEL AI image to an AWS AMI".
- You have the AWS command-line tool installed and is properly configured with your aws_access_key_id and aws_secret_access_key.
- You configured your Virtual Private Cloud (VPC).
- You created a subnet for your instance.
- You created a SSH key-pair.
- You created a security group on AWS.

Procedure

1. For various parameters, you need to gather the ID of the variable.
   a. To access the image ID, run the following command:

```
$ aws ec2 describe-images --owners self
```

   b. To access the security group ID, run the following command:

```
$ aws ec2 describe-security-groups
```

c. To access the subnet ID, run the following command:

```
$ aws ec2 describe-subnets
```

2. Populate environment variables for when you create the instance

```
$ instance_name=rhel-ai-instance
$ ami=<ami-id>
$ instance_type=<instance-type-size>
$ key_name=<key-pair-name>
$ security_group=<sg-id>
$ disk_size=<size-of-disk>
```

3. Create your instance using the variables by running the following command:

```
$ aws ec2 run-instances \
    --image-id $ami \
    --instance-type $instance_type \
    --key-name $key_name \
    --security-group-ids $security_group \
    --subnet-id $subnet \
    --block-device-mappings DeviceName=/dev/sda1,Ebs='{VolumeSize='$disk_size'}' \
    --tag-specifications 'ResourceType=instance,Tags=
[{Key=Name,Value='$instance_name'}]'
```

User account

The default user account in the RHEL AI AMI is **cloud-user**. It has all permissions via **sudo** without password.

Verification

- To verify that your Red Hat Enterprise Linux AI tools are installed correctly, you need to run the **ilab** command:

```
$ ilab
```

Example output

```
$ ilab
Usage: ilab [OPTIONS] COMMAND [ARGS]...

  CLI for interacting with InstructLab.

  If this is your first time running ilab, it's best to start with `ilab
  config init` to create the environment.

Options:
  --config PATH  Path to a configuration file.  [default:
                 /home/cloud--user/.config/instructlab/config.yaml]
  -v, --verbose  Enable debug logging (repeat for even more verbosity)
  --version      Show the version and exit.
  --help         Show this message and exit.
```

```
Commands:
  config    Command Group for Interacting with the Config of InstructLab.
  data      Command Group for Interacting with the Data generated by...
  model     Command Group for Interacting with the Models in InstructLab.
  system    Command group for all system-related command calls.
  taxonomy  Command Group for Interacting with the Taxonomy of InstructLab.

Aliases:
  chat      model chat
  convert   model convert
  diff      taxonomy diff
  download  model download
  evaluate  model evaluate
  generate  data generate
  init      config init
  list      model list
  serve     model serve
  sysinfo   system info
  test      model test
  train     model train
```

CHAPTER 4. INSTALLING RHEL AI ON IBM CLOUD

For installing and deploying Red Hat Enterprise Linux AI on IBM Cloud, you must first convert the RHEL AI image into an IBM Cloud image. You can then launch an instance using the IBM Cloud image and deploy RHEL AI on an IBM Cloud machine.

4.1. CONVERTING THE RHEL AI IMAGE INTO A IBM CLOUD IMAGE.

To create a bootable image in IBM Cloud you must configure your IBM Cloud accounts, set up a Cloud Object Storage (COS) bucket, and create a IBM Cloud image using the RHEL AI image.

Prerequisites

- You installed the IBM CLI on your specific machine. For more information about installing IBM Cloud CLI, see Installing the stand-alone IBM Cloud CLI .

Procedure

1. Log in to IBM Cloud with the following command:

```
$ ibmcloud login
```

When prompted, select your desired account to log in to.

Example output of the login

```
$ ibmcloud login
API endpoint: https://cloud.ibm.com
Region: us-east

Get a one-time code from https://identity-1.eu-central.iam.cloud.ibm.com/identity/passcode to proceed.
Open the URL in the default browser? [Y/n] >
One-time code >
Authenticating...
OK

Select an account:
1. <account-name>
2. <account-name-2>

API endpoint:     https://cloud.ibm.com
Region:           us-east
User:             <user-name>
Account:          <selected-account>
Resource group:   No resource group targeted, use 'ibmcloud target -g RESOURCE_GROUP'
```

2. You need to set up various IBM Cloud configurations and create your COS bucket before generating a QCOW2 image.

   a. You can install the necessary IBM Cloud plugins by running the following command:

```
$ ibmcloud plugin install cloud-object-storage infrastructure-service
```

b. Set your preferred resource group, the following example command sets the resource group named **Default**.

```
$ ibmcloud target -g Default
```

c. Set your preferred region, the following example command sets the **us-east** region.

```
$ ibmcloud target -r us-east
```

d. You need to select a deployment plan for your service instance. Ensure you check the properties and pricing on the IBM cloud website.

i. You can list the available deployment plans by running the following command:

```
$ ibmcloud catalog service cloud-object-storage --output json | jq -r '.[].children[] |
select(.children != null) | .children[].name'
```

ii. The following example command uses the **premium-global-deployment** plan and puts it in the environment variable **cos_deploy_plan**:

```
$ cos_deploy_plan=premium-global-deployment
```

iii. Create a Cloud Object Storage (COS) service instance and save the name in an environment variable named **cos_si_name** and create the **cloud-object-storage** and by running the following commands:

```
$ cos_si_name=THE_NAME_OF_YOUR_SERVICE_INSTANCE
```

```
$ ibmcloud resource service-instance-create ${cos_si_name} cloud-object-storage
standard global -d ${cos_deploy_plan}
```

e. Get the Cloud Resource Name (CRN) for your Cloud Object Storage (COS) bucket in a variable named **cos_crn** by running the following commands:

```
$ cos_crn=$(ibmcloud resource service-instance ${cos_si_name} --output json| jq -r '.[] |
select(.crn | contains("cloud-object-storage")) | .crn')
```

```
$ ibmcloud cos config crn --crn ${cos_crn} --force
```

f. Create your Cloud Object Storage (COS) bucket named as the environment variable **bucket_name** with the following commands:

```
$ bucket_name=NAME_OF_MY_BUCKET
```

```
$ ibmcloud cos bucket-create --bucket ${bucket_name}
```

g. Allow the infrastructure service to read the buckets that are in the service instance **${cos_si_guid}** variable by running the following commands:

```
$ cos_si_guid=$(ibmcloud resource service-instance ${cos_si_name} --output json| jq -r
'.[] | select(.crn | contains("cloud-object-storage")) | .guid')
```

```
$ ibmcloud iam authorization-policy-create is cloud-object-storage Reader --source-resource-type image --target-service-instance-id ${cos_si_guid}
```

3. Now that your IBM Cloud Object Storage (CoS) service instance bucket is set up, you need to download the RAW image from Red Hat Enterprise Linux AI download page
4. Copy the RAW image link and add it to the following command:

```
$ curl -Lo disk.qcow2 "PASTE_HERE_THE_LINK_OF_THE_QCOW2_FILE"
```

5. Set the name you want to use as the RHEL AI IBM Cloud image

```
$ image_name=rhel-ai-20240703v0
```

6. Upload the QCOW2 image to the Cloud Object Storage (COS) bucket by running the following command:

```
$ ibmcloud cos upload --bucket ${bucket_name} --key ${image_name}.qcow2 --file disk.qcow2 --region <region>
```

7. Convert the QCOW2 you just uploaded to an IBM Cloud image with the following commands:

```
$ ibmcloud is image-create ${image_name} --file cos://<region>/${bucket_name}/${image_name}.qcow2 --os-name red-ai-9-amd64-nvidia-byol
```

8. Once the job launches, set the IBM Cloud image configurations into a variable called **image_id** by running the following command:

```
$ image_id=$(ibmcloud is images --visibility private --output json | jq -r '.[] | select(.name=="'$image_name'") | .id')
```

9. You can view the progress of the job with the following command:

```
$ while ibmcloud is image --output json ${image_id} | jq -r .status | grep -xq pending; do sleep 1; done
```

10. You can view the information of the newly created image with the following command:

```
$ ibmcloud is image ${image_id}
```

4.2. DEPLOYING YOUR INSTANCE ON IBM CLOUD USING THE CLI

You can launch an instance with your new RHEL AI IBM Cloud image from the IBM Cloud web console or the CLI. You can use whichever method of deployment you want to launch your instance. The following procedure displays how you can use the CLI to launch an IBM Cloud instance with the custom IBM Cloud image

If you choose to use the CLI as a deployment option, there are several configurations you have to create, as shown in "Prerequisites".

Prerequisites

- You created your RHEL AI IBM Cloud image. For more information, see "Converting the RHEL AI image to an IBM Cloud image".
- You installed the IBM CLI on your specific machine, see Installing the stand-alone IBM Cloud CLI.
- You configured your Virtual private cloud (VPC).
- You created a subnet for your instance.

Procedure

1. Log in to your IBM Cloud account and select the Account, Region and Resource Group by running the following command:

```
$ ibmcloud login -c <ACCOUNT_ID> -r <REGION> -g <RESOURCE_GROUP>
```

2. Before launching your IBM Cloud instance on the CLI, you need to create several configuration variables for your instance.

   a. Install the **infrastructure-service** plugin for IBM Cloud by running the following command

   ```
   $ ibmcloud plugin install infrastructure-service
   ```

   b. You need to create an SSH public key for your IBM Cloud account. IBM Cloud supports RSA and ed25519 keys. The following example command uses the ed25519 key types and names it **ibmcloud**.

   ```
   $ ssh-keygen -f ibmcloud -t ed25519
   ```

   c. You can now upload the public key to your IBM Cloud account by following the example command.

   ```
   $ ibmcloud is key-create my-ssh-key @ibmcloud.pub --key-type ed25519
   ```

   d. You need to create a Floating IP for your IBM Cloud instance by following the example command. Ensure you change the region to your preferred zone.

   ```
   $ ibmcloud is floating-ip-reserve my-public-ip --zone <region>
   ```

3. You need to select the instance profile that you want to use for the deployment. List all the profiles by running the following command:

```
$ ibmcloud is instance-profiles
```

   Make a note of your preferred instance profile, you will need it for your instance deployment.

4. You can now start creating your IBM Cloud instance. Populate environment variables for when you create the instance.

```
name=my-rhelai-instance
vpc=my-vpc-in-us-east
zone=us-east-1
subnet=my-subnet-in-us-east-1
```

```
instance_profile=gx3-64x320x4l4
image=my-custom-rhelai-image
sshkey=my-ssh-key
floating_ip=my-public-ip
disk_size=250
```

5. You can now launch your instance, by running the following command:

```
$ ibmcloud is instance-create \
    $name \
    $vpc \
    $zone \
    $instance_profile \
    $subnet \
    --image $image \
    --keys $sshkey \
    --boot-volume '{"name": "'${name}'-boot", "volume": {"name": "'${name}'-boot", "capacity":
'${disk_size}', "profile": {"name": "general-purpose"}}}' \
    --allow-ip-spoofing false
```

6. Link the Floating IP to the instance by running the following command:

```
$ ibmcloud is floating-ip-update $floating_ip --nic primary --in $name
```

User account

The default user account in the RHEL AI AMI is **cloud-user**. It has all permissions via **sudo** without password.

Verification

- To verify that your Red Hat Enterprise Linux AI tools are installed correctly, run the **ilab** command:

```
$ ilab
```

Example output

```
$ ilab
Usage: ilab [OPTIONS] COMMAND [ARGS]...

  CLI for interacting with InstructLab.

  If this is your first time running ilab, it's best to start with `ilab
  config init` to create the environment.

Options:
  --config PATH  Path to a configuration file.  [default:
                 /home/auser/.config/instructlab/config.yaml]
  -v, --verbose  Enable debug logging (repeat for even more verbosity)
  --version      Show the version and exit.
  --help         Show this message and exit.

Commands:
```

```
  config    Command Group for Interacting with the Config of InstructLab.
  data      Command Group for Interacting with the Data generated by...
  model     Command Group for Interacting with the Models in InstructLab.
  system    Command group for all system-related command calls
  taxonomy  Command Group for Interacting with the Taxonomy of InstructLab.

Aliases:
  chat      model chat
  convert   model convert
  diff      taxonomy diff
  download  model download
  evaluate  model evaluate
  generate  data generate
  init      config init
  list      model model_list
  serve     model serve
  sysinfo   system info
  test      model test
  train     model train
```

4.3. ADDING MORE STORAGE TO YOUR IBM CLOUD INSTANCE

In [ibm-c], there is a size restriction of 250 GB of storage in the main IBM Cloud disk. RHEL AI might require more storage for models and generation data.

You can add more storage by attaching an extra disk to your instance and using it to hold data for RHEL AI.

Prerequisites

- You have a IBM Cloud RHEL AI instance.

Procedure

1. Create an environment variable called **name** that has the name of your instance by running the following command:

```
$ name=my-rhelai-instance
```

2. Set the size of the new volume by running the following command:

```
$ data_volume_size=1000
```

3. Create and attach the instance volume by running the following command:

```
$ ibmcloud is instance-volume-attachment-add data ${name} \
   --new-volume-name ${name}-data \
   --profile general-purpose \
   --capacity ${data_volume_size}
```

4. You can list all the disks with the following command:

```
$ lsblk
```

5. Create a **disk** variable with the content of the disk path your using. The following example command uses the **/dev/vdb** path.

```
$ disk=/dev/vdb
```

6. Create a partition on your disk by running the following command:

```
$ sgdisk -n 1:0:0 $disk
```

7. Format and label the partition by running the following command:

```
$ mkfs.xfs -L ilab-data ${disk}1
```

8. You can configure your system to auto mount to your preferred directory. The following example command uses the **/mnt** directory.

```
$ echo LABEL=ilab-data /mnt xfs defaults 0 0 >> /etc/fstab
```

9. Reload the **systemd** service to acknowledge the new configuration on mounts by running the following command:

```
$ systemctl daemon-reload
```

10. Mount the disk with the following command:

```
$ mount -a
```

11. Grant write permissions to all users in the new file system by running the following command:

```
$ chmod 1777 /mnt/
```

4.4. ADDING A DATA STORAGE DIRECTORY TO YOUR INSTANCE

By default RHEL AI holds configuration data in the **$HOME** directory. You can change this default to a different directory for holding InstructLab data.

Prerequisites

- You have a Red Hat Enterprise Linux AI instance
- You added an extra storage disk to your instance

Procedure

1. You can configure the **ILAB_HOME** environment variable by writing it to the **$HOME/.bash_profile** file by running the following commands:

```
$ echo 'export ILAB_HOME=/mnt' >> $HOME/.bash_profile
```

2. You can make that change effective by reloading the **$HOME/.bash_profile** file with the following command:

```
$ source $HOME/.bash_profile
```

CHAPTER 5. INSTALLING RHEL AI ON GOOGLE CLOUD PLATFORM (GCP) (TECHNOLOGY PREVIEW)

For installing and deploying Red Hat Enterprise Linux AI on Google Cloud Platform, you must first convert the RHEL AI image into an GCP image. You can then launch an instance using the GCP image and deploy RHEL AI on a Google Cloud Platform machine.

IMPORTANT

Installing Red Hat Enterprise Linux AI on GCP 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 .

5.1. CONVERTING THE RHEL AI IMAGE INTO A GOOGLE CLOUD PLATFORM IMAGE.

To create a bootable image in Google Cloud Platform you must configure your Google Cloud Platform account, create an Google Cloud Storage bucket, and create an Google Cloud Platform image using the RHEL AI raw image.

Prerequisites

- You installed the Google Cloud Platform CLI on your specific machine. For more information about installing the GCP CLI, see Install the Google Cloud Platform CLI on Linux .
- You must be on a Red Hat Enterprise Linux version 9.2 - 9.4 system
- Your machine must have an additional 100 GB of disk space.

Procedure

1. Log in to Google Cloud Platform with the following command:

```
$ gcloud auth login
```

Example output of the login.

```
$ gcloud auth login
Your browser has been opened to visit:

    https://accounts.google.com/o/oauth2/auth?XXXXXXXXXXXXXXXXXXXX


You are now logged in as [user@example.com].
Your current project is [your-project].  You can change this setting by running:
  $ gcloud config set project PROJECT_ID
```

2. You need to set up some Google Cloud Platform configurations and create your GCP Storage Container before creating the GCP image.

   a. Configure Google Cloud Platform CLI to use your project.

   ```
   $ gcloud_project=your-gcloud-project
   $ gcloud config set project $gcloud_project
   ```

   b. Create an environment variable defining the region where you want to operate.

   ```
   $ gcloud_region=us-central1
   ```

   c. Create a Google Cloud Platform Storage Container.

   ```
   $ gcloud_bucket=name-for-your-bucket
   $ gsutil mb -l $gcloud_region gs://$gcloud_bucket
   ```

3. Red Hat currently does not provide RHEL AI Google Cloud Platform images. You need to create a GCP disk image using RHEL AI bootc image as base.

   a. Create this **Containerfile** file, using the appropriate version of RHEL AI in the FROM line.

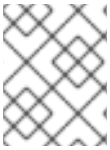
   ```
   FROM registry.redhat.io/rhelai1/bootc-nvidia-rhel9:1.2

   RUN eval $(grep VERSION_ID /etc/os-release) \
     && echo -e "[google-compute-engine]\nname=Google Compute
   Engine\nbaseurl=https://packages.cloud.google.com/yum/repos/google-compute-engine-
   el${VERSION_ID/.*}-x86_64-
   stable\nenabled=1\ngpgcheck=1\nrepo_gpgcheck=0\ngpgkey=https://packages.cloud.googl
   e.com/yum/doc/yum-key.gpg\n       https://packages.cloud.google.com/yum/doc/rpm-
   package-key.gpg" > /etc/yum.repos.d/google-cloud.repo \
     && dnf install -y --nobest \
        acpid \
        cloud-init \
        google-compute-engine \
        google-osconfig-agent \
        langpacks-en \
        rng-tools \
        timedatex \
        tuned \
        vim \
     && curl -sSo /tmp/add-google-cloud-ops-agent-repo.sh
   https://dl.google.com/cloudagents/add-google-cloud-ops-agent-repo.sh \
     && bash /tmp/add-google-cloud-ops-agent-repo.sh --also-install --remove-repo \
     && rm /tmp/add-google-cloud-ops-agent-repo.sh \
     && mkdir -p /var/lib/rpm-state \
     && dnf remove -y irqbalance microcode_ctl \
     && rmdir /var/lib/rpm-state \
     && rm -f /etc/yum.repos.d/google-cloud.repo \
     && sed -i -e '/^pool /c\server metadata.google.internal iburst' /etc/chrony.conf \
     && echo -e 'PermitRootLogin no\nPasswordAuthentication no\nClientAliveInterval 420'
   >> /etc/ssh/sshd_config \
     && echo -e '[InstanceSetup]\nset_boto_config = false' >
   /etc/default/instance_configs.cfg \
     && echo 'blacklist floppy' > /etc/modprobe.d/blacklist_floppy.conf \
   ```

```
  && echo -e '[install]\nkargs = ["net.ifnames=0", "biosdevname=0",
"scsi_mod.use_blk_mq=Y", "console=ttyS0,38400n8d", "cloud-init=disabled"]' >
/usr/lib/bootc/install/05-cloud-kargs.toml
```

b. Build the bootc image, in the same directory that holds the **Containerfile**, by running the following commands:

```
$ GCP_BOOTC_IMAGE=quay.io/yourquayusername/bootc-nvidia-rhel9-gcp
$ podman build --file Containerfile --tag ${GCP_BOOTC_IMAGE} .
```



NOTE

Ensure you are running the **podman build** command from a RHEL enabled system. If you are not on a RHEL system, building the **Containerfile** file will fail.

a. Create a **config.toml** file that will be used in disk image generation.

```
[customizations.kernel]
name = "gcp"
append = "net.ifnames=0 biosdevname=0 scsi_mod.use_blk_mq=Y console=ttyS0,38400n8d
cloud-init=disabled"
```

b. Build the disk image using bootc-image-builder by running the following commands:

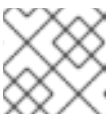
```
$ mkdir -p build/store build/output
$ podman run --rm -ti --privileged --pull newer \
  -v /var/lib/containers/storage:/var/lib/containers/storage \
  -v ./build/store:/store -v ./build/output:/output \
  -v ./config.toml:/config.toml \
  quay.io/centos-bootc/bootc-image-builder \
    --config /config.toml \
    --chown 0:0 \
    --local \
    --type raw \
    --target-arch x86_64 \
    ${GCP_BOOTC_IMAGE}
```

1. Set the name you want to use as the RHEL AI Google Cloud Platform image.

```
$ image_name=rhel-ai-1-2
```

2. Create a **tar.gz** file containing the RAW file you created.

```
$ raw_file=<path-to-raw-file>
$ tar cf rhelai_gcp.tar.gz --transform "s|$raw_file|disk.raw|" --use-compress-
program=pigz "$raw_file"
```



NOTE

You can use **gzip** instead of **pigz**.

3. Upload the tar.gz file to the Google Cloud Platform Storage Container by running the following command:

```
$ gsutil cp rhelai_gcp.tar.gz "gs://${gcloud_bucket}/$image_name.tar.gz"
```

4. Create an Google Cloud Platform image from the **tar.gz** file you just uploaded with the following command:

```
$ gcloud compute images create \
   "$image_name" \
   --source-uri="gs://${gcloud_bucket}/$image_name.tar.gz" \
   --family "rhel-ai" \
   --guest-os-features=GVNIC
```

5.2. DEPLOYING YOUR INSTANCE ON GOOGLE CLOUD PLATFORM USING THE CLI

You can launch an instance with your new RHEL AI Google Cloud Platform image from the Google Cloud Platform web console or the CLI. You can use whichever method of deployment you want to launch your instance. The following procedure displays how you can use the CLI to launch an Google Cloud Platform instance with the custom Google Cloud Platform image

If you choose to use the CLI as a deployment option, there are several configurations you have to create, as shown in "Prerequisites".

Prerequisites

- You created your RHEL AI Google Cloud Platform image. For more information, see "Converting the RHEL AI image to a Google Cloud Platform image".
- You installed the Google Cloud Platform CLI on your specific machine, see Install the Google Cloud Platform CLI on Linux.

Procedure

1. Log in to your Google Cloud Platform account by running the following command:

```
$ gcloud auth login
```

2. Before launching your Google Cloud Platform instance on the CLI, you need to create several configuration variables for your instance.

3. You need to select the instance profile that you want to use for the deployment. List all the profiles in the desired region by running the following command:

```
$ gcloud compute machine-types list --zones=<zone>
```

   Make a note of your preferred machine type, you will need it for your instance deployment.

4. You can now start creating your Google Cloud Platform instance. Populate environment variables for when you create the instance.

```
name=my-rhelai-instance
```

```
zone=us-central1-a
machine_type=a3-highgpu-8g
accelerator="type=nvidia-h100-80gb,count=8"
image=my-custom-rhelai-image
disk_size=1024
subnet=default
```

5. Configure the zone to be used.

```
$ gcloud config set compute/zone $zone
```

6. You can now launch your instance, by running the following command:

```
$ gcloud compute instances create \
    ${name} \
    --machine-type ${machine_type} \
    --image $image \
    --zone $zone \
    --subnet $subnet \
    --boot-disk-size ${disk_size} \
    --boot-disk-device-name ${name} \
    --accelerator=$accelerator
```

Verification

- To verify that your Red Hat Enterprise Linux AI tools are installed correctly, run the **ilab** command:

```
$ ilab
```

Example output

```
$ ilab
Usage: ilab [OPTIONS] COMMAND [ARGS]...

  CLI for interacting with InstructLab.

  If this is your first time running ilab, it's best to start with `ilab
  config init` to create the environment.

Options:
  --config PATH  Path to a configuration file.  [default:
                 /home/auser/.config/instructlab/config.yaml]
  -v, --verbose  Enable debug logging (repeat for even more verbosity)
  --version      Show the version and exit.
  --help         Show this message and exit.

Commands:
  config    Command Group for Interacting with the Config of InstructLab.
  data      Command Group for Interacting with the Data generated by...
  model     Command Group for Interacting with the Models in InstructLab.
  system    Command group for all system-related command calls
  taxonomy  Command Group for Interacting with the Taxonomy of InstructLab.
```

```
Aliases:
  chat       model chat
  convert    model convert
  diff       taxonomy diff
  download   model download
  evaluate   model evaluate
  generate   data generate
  init       config init
  list       model list
  serve      model serve
  sysinfo    system info
  test       model test
  train      model train
```

CHAPTER 6. INSTALLING RHEL AI ON AZURE

For installing and deploying Red Hat Enterprise Linux AI on Azure, you must first convert the RHEL AI image into an Azure image. You can then launch an instance using the Azure image and deploy RHEL AI on an Azure machine.

6.1. CONVERTING THE RHEL AI IMAGE INTO A AZURE IMAGE

To create a bootable image on Azure you must configure your Azure account, create an Azure Storage Container, and create an Azure image using the RHEL AI raw image.

Prerequisites

- You installed the Azure CLI on your specific machine. For more information on installing the Azure CLI, see Install the Azure CLI on Linux.
- You installed the AzCopy on your specific machine. For more information on installing AzCopy, see Install AzCopy on Linux.

Procedure

1. Log in to Azure by running the following command:

```
$ az login
```

Example output of the login

```
$ az login
A web browser has been opened at
https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue the
login in the web browser. If no web browser is available or if the web browser fails to open,
use device code flow with `az login --use-device-code`.
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "c7b976df-89ce-42ec-b3b2-a6b35fd9c0be",
    "id": "79d7df51-39ec-48b9-a15e-dcf59043c84e",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Team Name",
    "state": "Enabled",
    "tenantId": "0a873aea-428f-47bd-9120-73ce0c5cc1da",
    "user": {
      "name": "user@example.com",
      "type": "user"
    }
  }
]
```

2. Log in with the **azcopy** tool using the following commands:

```
$ keyctl new_session
$ azcopy login
```

3. You need to set up various Azure configurations and create your Azure Storage Container before creating the Azure image.

   a. Create an environment variable defining the location of your instance with the following command:

   ```
   $ az_location=eastus
   ```

   b. Create a resource group and save the name in an environment variable named **az_resource_group**. The following example creates a resource group named **Default** in the location **eastus**. (You can omit this step if you want to use an already existing resource group).

   ```
   $ az_resource_group=Default
   $ az group create --name ${az_resource_group} --location ${az_location}
   ```

   c. Create an Azure storage account and save the name in an environment variable named **az_storage_account** by running the following commands:

   ```
   $ az_storage_account=THE_NAME_OF_YOUR_STORAGE_ACCOUNT
   ```

   ```
   $ az storage account create \
     --name ${az_storage_account} \
     --resource-group ${az_resource_group} \
     --location ${az_location} \
     --sku Standard_LRS
   ```

   d. Create your Azure Storage Container named as the environment variable **az_storage_container** with the following commands:

   ```
   $ az_storage_container=NAME_OF_MY_BUCKET
   $ az storage container create \
     --name ${az_storage_container} \
     --account-name ${az_storage_account} \
     --public-access off
   ```

   e. You can get your Subscription ID from the Azure account list by running the following command:

   ```
   $ az account list --output table
   ```

   f. Create a variable named ` az_subscription_id` with your Subscription ID .

   ```
   $ az_subscription_id=46c08fb3-83c5-4b59-8372-bf9caf15a681
   ```

   g. Grant **azcopy** write permission to user into the storage container. This example grants permission to the user **user@example.com**.

   ```
   $ az role assignment create \
     --assignee user@example.com \
     --role "Storage Blob Data Contributor" \
     --scope
   ```

```
/subscriptions/${az_subscription_id}/resourceGroups/${az_resource_group}/providers/Microsoft.Storage/storageAccounts/${az_storage_account}/blobServices/default/containers/${az_storage_container}
```

4. Now that your Azure storage container is set up, you need to download the Azure VHD image from Red Hat Enterprise Linux AI download page .

5. Set the name you want to use as the RHEL AI Azure image.

```
$ image_name=rhel-ai-1.2
```

6. Upload the VHD file to the Azure Storage Container by running the following command:

```
$ az_vhd_url="https://${az_storage_account}.blob.core.windows.net/${az_storage_container}/$(basename ${vhd_file})"
$ azcopy copy "$vhd_file" "$az_vhd_url"
```

7. Create an Azure image from the VHD file you just uploaded with the following command:

```
$ az image create --resource-group $az_resource_group \
   --name "$image_name" \
   --source "${az_vhd_url}" \
   --location ${az_location} \
   --os-type Linux \
   --hyper-v-generation V2
```

6.2. DEPLOYING YOUR INSTANCE ON AZURE USING THE CLI

You can launch an instance with your new RHEL AI Azure image from the Azure web console or the CLI. You can use whichever method of deployment you want to launch your instance. The following procedure displays how you can use the CLI to launch an Azure instance with the custom Azure image

If you choose to use the CLI as a deployment option, there are several configurations you have to create, as shown in "Prerequisites".

Prerequisites

- You created your RHEL AI Azure image. For more information, see "Converting the RHEL AI image to an Azure image".
- You installed the Azure CLI on your specific machine, see Install the Azure CLI on Linux.

Procedure

1. Log in to your Azure account by running the following command:

```
$ az login
```

2. You need to select the instance profile that you want to use for the deployment. List all the profiles in the desired region by running the following command:

```
$ az vm list-sizes --location <region> --output table
```

Make a note of your preferred instance profile, you will need it for your instance deployment.

3. You can now start creating your Azure instance. Populate environment variables for when you create the instance.

```
name=my-rhelai-instance
az_location=eastus
az_resource_group=my_resource_group
az_admin_username=azureuser
az_vm_size=Standard_ND96isr_H100_v5
az_image=my-custom-rhelai-image
sshpubkey=$HOME/.ssh/id_rsa.pub
disk_size=1024
```

4. You can launch your instance, by running the following command:

```
$ az vm create \
   --resource-group $az_resource_group \
   --name ${name} \
   --image ${az_image} \
   --size ${az_vm_size} \
   --location ${az_location} \
   --admin-username ${az_admin_username} \
   --ssh-key-values @$sshpubkey \
   --authentication-type ssh \
   --nic-delete-option Delete \
   --accelerated-networking true \
   --os-disk-size-gb 1024 \
   --os-disk-name ${name}-${az_location}
```

Verification

- To verify that your Red Hat Enterprise Linux AI tools are installed correctly, run the **ilab** command:

```
$ ilab
```

Example output

```
$ ilab
Usage: ilab [OPTIONS] COMMAND [ARGS]...

  CLI for interacting with InstructLab.

  If this is your first time running ilab, it's best to start with `ilab
  config init` to create the environment.

Options:
  --config PATH  Path to a configuration file.  [default:
                 /home/auser/.config/instructlab/config.yaml]
  -v, --verbose  Enable debug logging (repeat for even more verbosity)
  --version      Show the version and exit.
  --help         Show this message and exit.
```

```
Commands:
  config    Command Group for Interacting with the Config of InstructLab.
  data      Command Group for Interacting with the Data generated by...
  model     Command Group for Interacting with the Models in InstructLab.
  system    Command group for all system-related command calls
  taxonomy  Command Group for Interacting with the Taxonomy of InstructLab.

Aliases:
  chat      model chat
  convert   model convert
  diff      taxonomy diff
  download  model download
  evaluate  model evaluate
  generate  data generate
  init      config init
  list      model list
  serve     model serve
  sysinfo   system info
  test      model test
  train     model train
```