depts.ttu.edu

Running Ollama on RedRaider | User Guides | High Performance Computing Center

State of Texas and Texas Tech University

10–13 minutes

This guide provides instructions for running the ollama large language model (LLM) management server on the RedRaider cluster in a batch or interactive session. A couple of variations are included below for you to experiment with. Please choose the minimum amount of resources if you use this program, as described below, as it is resource-intensive and other cluster account holders may also be in contention for these resources.

Important: A session on the Matador GPU partition is required for most models. Although it is technically possible to run on one of the cpu partitions (nocona or quanah), performance will be very slow for all except the smallest models, even with a full node request in terms of the number of cores. There are 40 GPUs on the matador partition, and availability varies by usage, so to minimize contention, we suggest that you minimize long-term intensive use or long interactive sessions to keep these resources available for as many people as possible. **If using an interactive session, please exit as soon as you are done and do not leave the session idle.**

The current version of CUDA on the Matador partition is 11. The most recent version of Ollama that supports this version of CUDA in the prebuilt tarball is 0.9.2 and the instructions below are set up for this version only. You can replace this with any previous version but until the Matador partition has been upgraded as currently planned for sometime in Fall semester 2025, please do not exceed this version.

Step by Step Guide:

1. Create a directory called ollama in your home area and cd into it:
   cd
   mkdir ollama-0.9.2

   cd ollama-0.9.2
2. Download the latest AMD64 architecture tarball from the GitHub web site for ollama (https://github.com/ollama/ollama/releases). For example, the current one can be fetched

with the command
wget https://ollama.com/download/ollama-linux-amd64.tgz?version=0.9.2 -O ollama-linux-amd64.tgz

Then unpack this in your ollama directory with
tar -zxvf ollama-linux-amd64.tgz

The above preparation steps can be done on any node including the login nodes. To run ollama, you will need to get an interactive session as described below.

1. For most problems you will want to run ollama with GPU assistance. For each interactive or batch session on Matador, you can request either one or two GPUs and a suitable number of cpus. You can choose any number from 1 to 20 cpus for a single GPU job or from 1 to 40 cpus for a 2-GPU job. For further information, see our Job Submission Guide. For example, if trying this interactively, the commands would be either:
for a 1-GPU interactive job, use the command below. Fewer cpus is okay if you don't think you need them but there should be no harm in requesting these. See the notes below for running in batch for long jobs.

   interactive -c 20 -g 1 -p matador

   or
   # for a 2-GPU job, same considerations as above.

   interactive -c 40 -g 2 -p matador

   You can check the number of GPUs in your session with the **nvidia-smi** command. The default memory allocations made by the scheduler in each case should be fine.

1. Running any production studies once your code is working is best done in batch jobs. This approach is preferable once you have your planned commands worked out for the set of work you would like to do, since this will not require waiting for an interactive session to open up. The commands below are identical whether using an interactive or batch session. Details on how to set up a batch job are described in the Job Submission Guide and are further described in our online HPCC User Training slides and videos.
2. The commands below can be used in either the run script for a batch or in an interactive session.
   Get a free port number and start an instance of the ollama server on it in the background as follows:

   export OLPORT=$(python -c "import socket; s = socket.socket(); s.bind(('', 0));print(s.getsockname()[1]);s.close()");

   export OLLAMA_HOST=127.0.0.1:$OLPORT;

export OLLAMA_BASE_URL="http://localhost:$OLPORT";

~/ollama-0.9.2/bin/ollama serve >ollama_$OLPORT.log 2>ollama_$OLPORT.err &

3. Within that same session, you should be able to connect to the running ollama server on that node and port using the command
~/ollama-0.9.2/bin/ollama

For example, the command below should work to start an instance of the llama3.1 model running on that particular ollama server in the background. This is just an example; you should be able to choose any of the available models from the ollama library. The "--verbose" flag is optional but will allow you to see the speed of the responses to your prompts. If further information is needed, you can add "export OLLAMA_DEBUG=1" to the commands issued before starting the server above.
~/ollama-0.9.2/bin/ollama run llama3.2 --verbose

The startup for this command may take several seconds if the model has already been downloaded, or longer if the model is being fetched for the first time. Once the session starts, response will depend on the complexity of your prompt, the model chosen, and other factors.
Optionally, you could add the ~/ollama/bin directory to your PATH to avoid having to type that part in explicitly:

export PATH=~/ollama-0.9.2/bin:$PATH

To check that this is finding your Ollama executable, issue the command:
which ollama

Overall Information:

The session you are in should pick up and respect the OLLAMA_HOST and OLLAMA_BASE environmental variables defined above, which will point the client command to the right server instance. If running in batch, you may wish to feed the prompt(s) you want to the ollama client. There are several ways to do this, including either feeding the prompt in the ollama run command, skipping the client entirely and simply interacting with the running ollama instance on that port using curl, etc., so you can tune this step towards a useful batch workflow depending on your needs.

The ollama server background session will terminate automatically when you sign out of the interactive session or when your batch job terminates. This procedure should keep separate instances from colliding with each other in simultaneous interactive or batch jobs, even if other HPCC account holders run this or similar programs on the same nodes up to their batch resource limits.

You can experiment with the number of cpus in your session, though since there are only

two GPUs in each matador node, there is usually no harm in asking for the number of cpus that corresponds to your GPU request as above, and with either running two separate ollama client/server instances using only one GPU at a time per session as above, or with using both GPUs in a single session to see if this will result in faster processing of your prompts.

One final note: availability of the matador partition varies greatly. You may need to wait for some time for an interactive session if the partition is busy, or for a batch job to start, especially if you have been making a lot of use of the partition, depending on the current activity and your recent history of usage. This is why we recommend setting your work up to run as batch jobs if possible. For testing, you might have better luck with the gpu-build partition, which has 1 GPU and up to 32 cores, and is often empty. Let us know your experience once you have gotten a few jobs to run.

Let us know at hpccsupport@ttu.edu if you run into any difficulties.