**AI-ML Linux Desktop on Lambda Labs Tensorboook**

**Objective:**

Set up a Razer x Lambda Tensorbook with GPU-enabled Python 3+, Tensorflow, Keras, Torch, OpenCV, Jupyter Notebooks, and core scientific / image handling packages.

1. **Setup Tensorbook with Ubuntu 22.04:**

The recovery ISO image for [Tensor book](https://lambdalabs.com/deep-learning/laptops/tensorbook) can be downloaded using the following link:

* [Lambda Recovery for Tensorbook (Jammy)](https://files.lambdalabs.com/recovery/tensorbook-jammy-20230704.iso) (based on Ubuntu 22.04 LTS jammy)

#### **Notes:** This recovery image is for the *Razer x Lambda Tensorbook* only and won’t work on older Tensorbook models. The recovery images contain software distributed under various licenses, including the [Software License Agreement (SLA) for NVIDIA cuDNN](https://docs.nvidia.com/deeplearning/cudnn/sla/index.html). The licenses can be viewed in the recovery images at/usr/share/doc/\*/copyright. By using the software contained in the recovery images, you agree to these licenses. The recovery [ISO file](https://docs.lambdalabs.com/linux/recovery-images/) needs to be installed appropriately in a bootable USB for install as shown [here](https://ubuntu.com/tutorials/create-a-usb-stick-on-ubuntu" \l "1-overview).

1. **Install image and other system dependencies for Ubuntu**

Update and install recovery image by connecting the bootable recovery USB and pressing F12 during startup of the Lambda Tensorbook.

1. A screenshot of a computer

   Description automatically generated**Make a user named “aiml”**

Set the username of your machine to:

**aiml**

1. **Modify desktop to your choosing:**

Download and Install Anaconda 3.0 (with reboot):

[Download Anaconda installer](https://www.anaconda.com/download" \l "downloads) or [Download dependencies and anaconda for Bit (x86)](https://docs.anaconda.com/free/anaconda/install/linux/), install and reboot following the official instructions.

One can deactivate automatic base conda in shell with the command

**conda config --set auto\_activate\_base false**

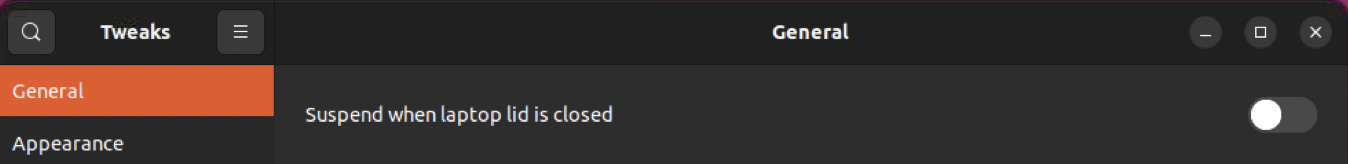
1. **Install VNC Connect:**

Follow the instructions [here](https://www.realvnc.com/en/blog/how-to-install-vnc-server-on-ubuntu-control-linux-machine-remotely/). Then prevent VNC suspending when lid is off with

1. **Install Tweaks Caffeinate:**

Installing Tweaks as [here](https://installati.one/install-caffeine-ubuntu-22-04/). Sometimes it comes already installed in Linux.

A picture containing screenshot, circle, iPod, design

Description automatically generated

And by installing Caffeinate as [here](https://installati.one/install-caffeine-ubuntu-22-04/)

A picture containing beverage, coffee cup, coffee, serveware

Description automatically generated

1. **Install Dropbox:**

A screenshot of a computer

Description automatically generated with medium confidenceFollow the instructions [here](https://www.dropbox.com/install-linux) to install backend and then search for dropbox app in the ubuntu app manager and install as [here](https://itsfoss.com/install-dropbox-ubuntu/).

1. **Install Conky Manager 2:**

Download and Install (with reboot):

**1.)** First, open terminal by either pressing **Ctrl+Alt+T** on keyboard, or searching from ‘Activities’ overview screen. When it opens, run command to add the PPA:

sudo add-apt-repository ppa:ubuntuhandbook1/conkymanager2

*Type user password (no asterisk feedback) for sudo prompts and hit Enter to continue.*

A screenshot of a computer

Description automatically generated

**2.)** Then install Conky Manager2 and conky as dependency library via command:

sudo apt install conky-manager2

*NOTE: Linux Mint needs to manually update cache via sudo apt update command first.*

A screenshot of a computer

Description automatically generated

### Start and use Conky Manager 2:

Once installed, search for and launch Conky Manager app from the activities overview screen (or from start menu depends on your DE).

A screenshot of a video game

Description automatically generated with low confidence

The app comes with some default themes, though may not well configured for your desktop. Tick or un-tick any item in the list will start or stop conky with that theme, though you may also use the tool bar buttons.

And, you can use the “**spanner**” icon to config theme options, including screen location, size, background transparency. Advanced users may also click on the ‘**pencil**‘ icon to edit it via configuration file.

A screenshot of a computer

Description automatically generated

There are also tons of [Conky themes](https://www.pling.com/browse?cat=124&ord=latest) on the web. You can download one, and install it easily by clicking the second folder icon (with a little emblem).

#### Uninstall Conky, Conky Manager:

For any reason, you can easily remove the Ubuntu PPA by running the command below in terminal:

sudo add-apt-repository --remove ppa:ubuntuhandbook1/conkymanager2

If you want to remove Conky as well as Conky manager run command:

sudo apt remove --autoremove conky conky-manager2

1. **Install Boot Repair:**

Interesting tool in case one needs to repair the boot disk.

**sudo add-apt-repository ppa:yannubuntu/boot-repair && sudo apt update**

**sudo apt install -y boot-repair && boot-repair**

A picture containing circle, text

Description automatically generated

1. **Install Lambda Stack, Docker and Nvidia AI tools:**

# Lambda Stack is all the AI software you need, and it's always up to date

Lambda Stack provides a one line installation and managed upgrade path for: PyTorch®, TensorFlow, CUDA, cuDNN, and NVIDIA Drivers. It's compatible with Ubuntu 22.04 LTS and 20.04 LTS. No more futzing with your Linux AI software, Lambda Stack is here.

## Install Lambda Stack in one command

To install Lambda Stack on your desktop again, run this command on a fresh Ubuntu installation (22.04 and 20.04). For servers, see the [server installation section](https://lambdalabs.com/lambda-stack-deep-learning-software" \l "server-installation) below.

**wget -nv -O- https://lambdalabs.com/install-lambda-stack.sh | sh -  
sudo reboot**

If you'd like a high level video overview of the features of Lambda Stack, check out this video:

Lambda Stack can run on your laptop, workstation, server, cluster, inside a container, on the cloud, and comes pre-installed on every [Lambda GPU Cloud](https://lambdalabs.com/service/gpu-cloud?hsLang=en) instance. It provides up-to-date versions of PyTorch®, TensorFlow, CUDA, CuDNN, NVIDIA Drivers, and everything you need to be productive for AI.

Run this command and all of your AI software, from PyTorch® to CUDA, will be updated. **Like Magic.**

**sudo apt-get update && sudo apt-get dist-upgrade**

## It's compatible with your Docker and NGC containers

If you're already using GPU docker images or NGC containers, rest assured that Lambda Stack can run them.

After you've installed Lambda Stack, you can install a version of GPU accelerated Docker with this command:

**sudo apt-get install docker.io nvidia-container-toolkit**

1. **Linux post-installation steps for Docker Engine**

The Docker daemon always runs as the root user. If you don't want to preface the docker command with sudo, create a Unix group called docker and add users to it.

To create the docker group and add your user:

Create the docker group.

**sudo groupadd docker**

Add your user to the docker group.

**sudo usermod -aG docker $USER**

Log out and log back in so that your group membership is re-evaluated.

If you're running Linux in a virtual machine, it may be necessary to restart the virtual machine for changes to take effect.

You can also run the following command to activate the changes to groups:

**newgrp docker**

Verify that you can run docker commands without sudo.

**docker run hello-world**

1. **Make and use a docker container AIML Docker Container with Ubuntu:20.04 Focal and GPU enabled Tensorflow, Keras, PyTorch, Jupyter Lab**

Ensure that you have a docker version > 19.03. On Ubuntu, you can simply run `sudo apt-get install docker.io`. On a different OS, or if you prefer to use upstream docker, follow [Docker's installation instructions](https://docs.docker.com/engine/install/ubuntu/). If using Lambda Stack on your host machine, install nvidia-container-toolkit with `sudo apt-get install nvidia-container-toolkit`. Otherwise, follow [NVIDIA's installation instructions](https://github.com/NVIDIA/nvidia-docker)

### Download Dockerfile from repo

account@instance-name:~$

**git clone https://github.com/lrsoenksen/aiml-stack-jupyterlab-dockerfiles**

**cd aiml-stack-jupyterlab-dockerfiles**

Download from https://github.com/lrsoenksen/aiml-lambda-stack-jupyterlab-dockerfiles

### Building image

Build the image with the appropriate command for the distribution you wish to use.

account@instance-name:~$

**sudo docker login -u "USERNAME" -p "PASSWORD" docker.io**

**sudo docker build --rm -t aiml-stack:latest-gpu-jupyter -f Dockerfile .**

**sudo docker tag aiml-stack:latest-gpu-jupyter lrsoenksen/aiml-stack:latest-gpu-jupyter**

**sudo docker push lrsoenksen/aiml-stack:latest-gpu-jupyter**

Note that building these docker images requires acceptance of the [cuDNN license agreement] (https://docs.nvidia.com/deeplearning/sdk/cudnn-sla/index.html)

### Testing image

Test the image with GPU using the following commands

account@instance-name:~$

**sudo docker run -u $(id -u):$(id -g) -v $(pwd):$(pwd) -w $(pwd) -e HOME=$(pwd)/.home -it --rm --init --gpus all --rm --interactive --tty lrsoenksen/aiml-stack:latest-gpu-jupyter /usr/bin/python3 -c 'import tensorflow as tf; print(tf.config.list\_physical\_devices())'**

account@instance-name:~$

**sudo docker run -u $(id -u):$(id -g) -v $(pwd):$(pwd) -w $(pwd) -e HOME=$(pwd)/.home -it --rm --init --gpus all --rm --interactive --tty lrsoenksen/aiml-stack:latest-gpu-jupyter /usr/bin/python3 -c 'import torch; print(torch.rand(5, 5).cuda()); print("I love Lambda Stack with GPUs: ", end=""); print(torch.cuda.device\_count())'**

Enter bash of the image with GPU using the following commands and then load, where "home/aiml" can be change for whatever is your desired root folder.

account@instance-name:~$

**sudo docker run -u $(id -u):$(id -g) -v $(pwd):$(pwd) -w $(pwd) -e HOME=$(pwd)/.home -it --rm --init --gpus all -p 8888:8888 -p 6006:6006 lrsoenksen/aiml-stack:latest-gpu-jupyter bash**

And these commands can be used to test functionality

account@instance-name:~$

**python3 -c 'import tensorflow as tf; print(tf.config.list\_physical\_devices('GPU'))'**

**python3 -c 'import torch; print(torch.rand(5, 5).cuda()); print("I love Lambda Stack with GPUs: ", end=""); print(torch.cuda.device\_count())'**

**python3 -c 'import tensorflow as tf; print(tf.\_\_version\_\_)'**

**python3 -c 'from tensorflow import keras; print(keras.\_\_version\_\_)'**

**python3 -c 'import cv2; print(cv2.\_\_version\_\_)'**

**python3 -c 'import torch; print(torch.\_\_version\_\_)'**

**jupyter lab --ip=0.0.0.0 --port=8888 --allow-root --no-browser --core-mode**

Automatically Run Jupyter Lab image

account@instance-name:~$

**sudo docker run -u $(id -u):$(id -g) -v $(pwd):$(pwd) -w $(pwd) -e HOME=$(pwd)/.home -it --rm --init --gpus all -p 8888:8888 -p 6006:6006 lrsoenksen/aiml-stack:latest-gpu-jupyter jupyter lab --ip=0.0.0.0 --port=8888 --no-browser --core-mode --log-level='CRITICAL'**

where the -p 6006 is the default port of TensorBoard.

### Accessing JupyterLab

Click on the provided URL, or at your browser just enter http://localhost:8888 and provide the token defined by you. If Jupyter Lab is not working try reinstalling in bash:

account@instance-name:~$

**pip install jupyter -U && pip install jupyterlab**

### Test AIML packages inside dockerized JupyterLab

To test the upyter lab in docker one can execute the following code

account@instance-name:~$

**import tensorflow as tf; print(tf.config.list\_physical\_devices())**

**import torch; print(torch.rand(5, 5).cuda()); print("I love Lambda Stack with GPUs: ", end=""); print(torch.cuda.device\_count())**

**import tensorflow as tf; print(tf.\_\_version\_\_)**

**import keras; print(keras.\_\_version\_\_)**

**import cv2; print(cv2.\_\_version\_\_)**

**import torch; print(torch.\_\_version\_\_)**

-------------------------------------------------------------------------------------------------------------

### Reset everything (OPTIONAL - DON'T DO CARELESSLY)

The following is a radical solution. But if needed one can DELETES ALL YOUR DOCKER STUFF. INCLUDING VOLUMES like this:

account@instance-name:~$

**docker container prune -f**

**docker image prune -f**

**docker volume prune -f**

**docker stop $(docker ps -a -q)**

**docker rm -f $(docker ps -a -q)**

**docker image rm -f $(docker image ls -a -q)**

This removes images and containers

account@instance-name:~$

**sudo su**

**service docker stop**

**cd /var/lib/docker**

**rm -rf buildkit containers engine-id image network overlay2 plugins runtimes swarm tmp volumes**

**service docker start**

This resets all things Docker

account@instance-name:~$

**sudo reboot**

Reboot computer

1. **Grant Root Privileges and permissions to user**

Since you already have a user "aiml" and you want to grant it root privileges (make it a second root user). If you just want to add aiml to root group, without granting it all root privileges, run the following command:

account@instance-name:~$

**sudo usermod -aG sudo aiml**

**sudo visudo**

At the bottom of the file that opens, add this line for every user you want to give passwordless sudo permissions to aiml:

account@instance-name:~$

**aiml ALL=(ALL) NOPASSWD:ALL**