

# Stable Diffusion AMD guide

## Docker guide

Using Docker seems to be the easiest solution, and it works on any OS.

1. Install Docker. <https://docker.com>
2. Go to <https://hub.docker.com/r/rocm/pytorch/tags>
3. Pull the latest image, and run "the master test".
4. If it works, great! Follow the tard guide inside the running image.
5. If it fails with `hipErrorNoBinaryForGpu`: You'll have to compile PyTorch for your GPU. See the corresponding section. If you're on Arch, try the unofficial repository method.

## Manual guide (Linux only)

### Arch (unofficial repository)

This is much faster than compiling if you trust the repo maintainers. Recommended.

Add the arch4edu repository as per [https://wiki.archlinux.org/title/unofficial\\_user\\_repositories#arch4edu](https://wiki.archlinux.org/title/unofficial_user_repositories#arch4edu), then `sudo pacman -S rocm-hip-sdk rocm-ocl-sdk`. ROCm pytorch is also available as `python-pytorch-rocm`.

### Arch (compiling from source)

There *should* be instructions for installing Rocm on Fedora and Ubuntu somewhere on the internet. Possibly OpenSUSE as well.

#### ⚠ Warning

This will be slow. You have to compile a lot of software from source.

1.) Install `rocminfo` so we can set `AMDGPU_TARGETS`, which will slightly speed up compile times.

run: `rocminfo | grep Name`

You are looking for `gfx####`, then followed by your gpu on the next line. Once you have it do:

```
export AMDGPU_TARGETS="gfx####"
```

2.) Install [Rocm on Arch](#) using this command:

```
paru -S rocm-hip-sdk rocm-ocl-sdk
```

 I remember having issues with `yay` while running this, I would play it safe and use `paru`.

This will take a long time to compile, pray that it actually finishes. If you run out of ram during compiling, you will have to adjust Ninja flags.

Some packages use `Make` instead of `Ninja`, which doesn't do parallel compilation by default. If you want massive build speedup in exchange for more RAM usage, either patch the PKGBUILDs to use `Ninja` (add `-G Ninja` to the `CMake` invocation and replace `make` with `ninja` in other places) or do `export MAKEFLAGS="-j $(nproc)"` before building.

If `pacman` gives errors about file conflicts with `ocl-amd` uninstall that first ( `sudo pacman -R ocl-amd` ), then try again.

## Universal

3.) Load in to the environment

```
conda activate ldm
```

Create the environment first (follow other guides until the `conda env create -f environment.yaml` step). The GUITard guide replaces everything with one batch file in the repository, `webui.cmd`; peek the commands out of it, the environment name is in `environment.yaml`.

Use the same environment name that was created, obviously. Some guides use `ldx` or `ldo`.

#### 4.) [Go to torch's website](#)

Scroll down to the Install Torch prompt.

Choose `pip` for **Package**

`ROCM x.x.x` for **Compute Platform**

Copy the command at the bottom.

Run the command, or edit the environment file if you know how. I don't know how to use links in them.

This should work for KoboldAI too.

I was having issues crashing randomly when running the optimized `txt2img.py`, but `txt2img_gradio.py`, (and I'm guessing the normal `txt2img.py`) seems to be stable, just can't generate 512x512. I do 384x384 and that doesn't give out of memory errors.

Takes around 40 seconds with the gradio script after the first gen, at 70 sampling steps.

## The master test

```
import torch
torch.cuda.is_available()
print(torch.tensor([1., 2.], device='cuda'))
```

If ROCm supports your GPU and everything is working fine, you should see the following output:

```
True
tensor([1., 2.], device='cuda:0')
```

Now, this can fail in multiple ways (perhaps even more, these are documented as of now):

- **Outputs False on the first line, throws an exception on the second:** Either ROCm or the ROCm-compatible PyTorch wasn't installed correctly.
  - **hipErrorNoBinaryForGpu:** compile PyTorch from source. See below
- 6.) Running `python scripts/txt2img.py` or any of the other scripts depending on your vram should now work and use the gpu.

## Building PyTorch from source

### On Arch

`PYTORCH_ROCM_ARCH=gfx#### paru -S python-pytorch-rocm`. Take the `gfx####` value from the first step of the manual guide. This will install PyTorch globally, if you (most likely) want to put it in a local environment, follow the general guide.

### Into a conda environment

1. Clone (or download) <https://github.com/pytorch/pytorch>
2. Activate the environment. `conda activate ldm/ldx/ldo`
3. Navigate to the pytorch repo. Check out whatever release/bleeding edge branch you want to install.
4. `export PYTORCH_ROCM_ARCH=gfx####`
5. `python tools/amd/build_amd.py`
6. `export CMAKE_PREFIX_PATH=${CONDA_PREFIX:-"$(dirname $(which conda))/../"} && python setup.py install` This step might fail on some errors. Google them and patch them, it depends on your distro and compiler version. For Arch users, [here is a github issue with a collection of common errors](#).
7. You're done! Run the master test to ensure everything is working.

## Troubleshooting

If you get a screenful of seemingly random HIP compilation errors once trying to generate an image (either in the webui or the bare scripts),

```
export MIOPEN_DEBUG_COMGR_HIP_PCH_ENFORCE=0.
```

## Relevant links

- [https://github.com/RadeonOpenCompute/ROCm\\_Documentation/blob/master/Deep\\_learning/Deep-learning.rst](https://github.com/RadeonOpenCompute/ROCm_Documentation/blob/master/Deep_learning/Deep-learning.rst) Has a guide about

compiling PyTorch for a specific graphics architecture.

- <https://github.com/rocm-arch/python-pytorch-rocm/issues/44>