

How To

I'll put little lessons I learn here.

- Redirect traffic using iptables
- Make lxd container accessible outside of host machine
- How to control turbo boost inside the operating system
- lxc container setup
- Run docker container inside of lxc container
- Lock packages in zypper
- Installing matrix-synapse on pypy
- Download from nextcloud using wget
- Install printer drivers for brother printer in opensuse tumbleweed
- Install numpy-blis for amd efficiency
- Allennlp
- Install GTKWattman on opensuse tumbleweed
- How to use Intel's mkl library on AMD systems
- Install pytorch-rocm on bare metal opensuse Tumbleweed
- Install aftermarket cooler on M40
- Use Intel Quad bypass cards (82571EB PRO)
- Build pytorch with rocm on ubuntu 20.04
- Enable hibernation on asus zenbook with ryzen processor
- Set up zyxel travel router in client bridged mode
- Get my EGPU to work on opensuse tumbleweed with a framework 12th gen laptop

Redirect traffic using iptables

```
HOST_PORT=3230
PUBLIC_IP=192.168.1.150
CONT_PORT=22
CONT_IP=10.121.80.77
sudo iptables -t nat -I PREROUTING -i eth0 -p TCP -d $PUBLIC_IP --dport $HOST_PORT -j DNAT --
to-destination $CONT_IP:$CONT_PORT -m comment --comment "forward ssh to the container"
```

Make lxd container accessible outside of host machine

The default network connector type is a bridged connector and as such does not allow for outside connections. Run the following

```
lxc profile edit <name of profile>
```

to open up your profile in a text editor. Then, modify the value for `nictype` to be `macvlan` instead of `bridged`. Change `parent` to match one of the connected network interface names on the host machine, for example `eno2` if you have an interface named that. Then restart the containers using that profile.

Note that this does not prevent traffic between containers. The only stipulation of the macvlan nic type is that it cannot allow communication between the host and the container.

How to control turbo boost inside the operating system

If you would like to control whether turbo boost is allowed from within linux instead of using the bios settings for your server you can create the following scripts

enable_turbo

```
#!/bin/bash
echo "0" | sudo tee /sys/devices/system/cpu/intel_pstate/no_turbo
```

disable_turbo

```
#!/bin/bash
echo "1" | sudo tee /sys/devices/system/cpu/intel_pstate/no_turbo
```

I put these in two separate files and then made both executable. I moved them into a directory that was on my path.

lxc container setup

I prefer to purge openssh-server and then reinstall it as the default config requires preshared keys for all users which isn't really something I desire and it's simpler to just trash the default configs for containers and reinstall openssh-server from scratch.

If you get this error

```
sudo: no tty present and no askpass program specified
```

You can create a file like this

```
echo "import pty; pty.spawn('/bin/bash')" > /tmp/pty_spawn.py
```

And then run

```
python /tmp/pty_spawn.py
```

to switch to a terminal type where users can elevate privilege. Note that this is only a problem if you got into the container using `lxc exec --`.

This is not an issue if you ssh into the container directly.

Run docker container inside of lxc container

- Create an unprivileged container with nesting turned on
- Run docker containers as normal

Lock packages in zypper

To prevent updates to packages in zypper, you can use the `al` command to add a lock.

```
sudo zypper al texlive*
```

to view locks, you can use the `ll` command

```
sudo zypper ll
```

To remove a lock use the `rl` command

```
sudo zypper rl texlive*
```

Installing matrix-synapse on pypy

```
sudo apt install virtualenv
virtualenv -p ./pypy3 ~/synapse/env
source ~/synapse/env/bin/activate
pip install --upgrade pip
pip install --upgrade setuptools
sudo apt install libjpeg-dev libxslt-dev libxml2-dev
postgresql-server-dev-all
pip install matrix-synapse[all]
```

Currently, this is working. However, adding rooms is causing issues but only with some rooms. The error generated looks like this.

```
2019-02-17 18:15:11,331 - synapse.access.http.8008 - 233 - INFO - GET-5 - 192.168.1.254 -
8008 - Received request: GET /_matrix/client/r0/groups/world/profile2019-02-17 18:15:11,334 -
synapse.http.server - 112 - ERROR - GET-5 - Failed handle request via <function
JsonResource._async_render at 0x00007fae97c4f240>: <XForwardedForRequest at 0x7fae94986950
method='GET' uri='/_matrix/client/r0/groups/world/profile' clientproto='HTTP/1.0' site=8008>:
Traceback (most recent call last): File "/home/kenneth/synapse/env/site-
packages/twisted/internet/defer.py", line 1418, in _inlineCallbacks result =
g.send(result) File "/home/kenneth/synapse/env/site-packages/synapse/http/server.py", line
316, in _async_render
    callback_return = yield callback(request, **kwargs) File "/home/kenneth/synapse/env/site-
packages/twisted/internet/defer.py", line 1613, in unwindGenerator return
_cancellableInlineCallbacks(gen) File "/home/kenneth/synapse/env/site-
packages/twisted/internet/defer.py", line 1529, in _cancellableInlineCallbacks
    _inlineCallbacks(None, g, status)
--- <exception caught here> --- File "/home/kenneth/synapse/env/site-
packages/synapse/http/server.py", line 81, in wrapped_request_handler yield h(self,
request) File "/home/kenneth/synapse/env/site-packages/synapse/http/server.py", line 316, in
_async_render
    callback_return = yield callback(request, **kwargs) File "/home/kenneth/synapse/env/site-
```

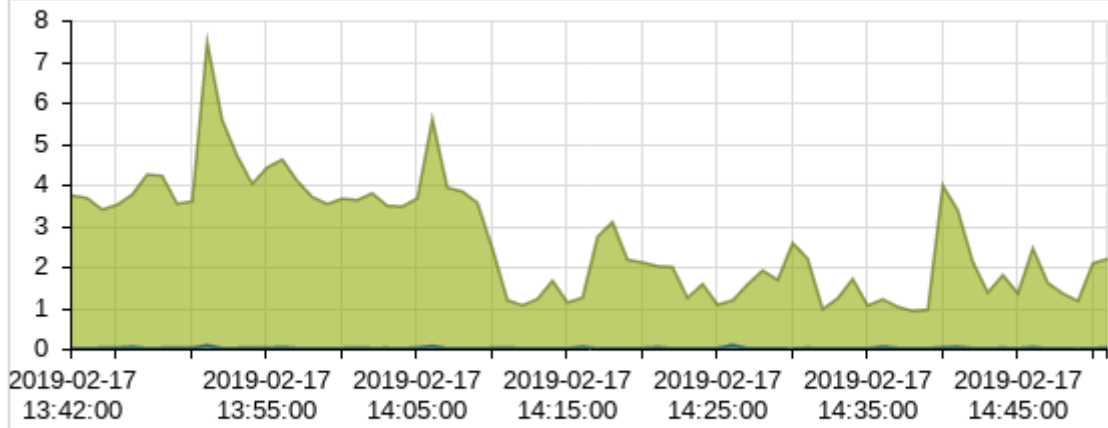


```
packages/twisted/internet/defer.py", line 1418, in _inlineCallbacks    result =
g.send(result) File "/home/kenneth/synapse/env/site-
packages/synapse/rest/client/v2_alpha/groups.py", line 47, in on_GET
    requester_user_id, File "/home/kenneth/synapse/env/site-
packages/synapse/handlers/groups_local.py", line 34, in f
    if self.is_mine_id(group_id): File "/home/kenneth/synapse/env/site-
packages/synapse/server.py", line 235, in is_mine_id    return string.split(":", 1)[1] ==
self.hostname
builtins.IndexError: list index out of range
```

If I switch to a regular version of python to add the room and then switch back to pypy, I can talk to the room just fine.

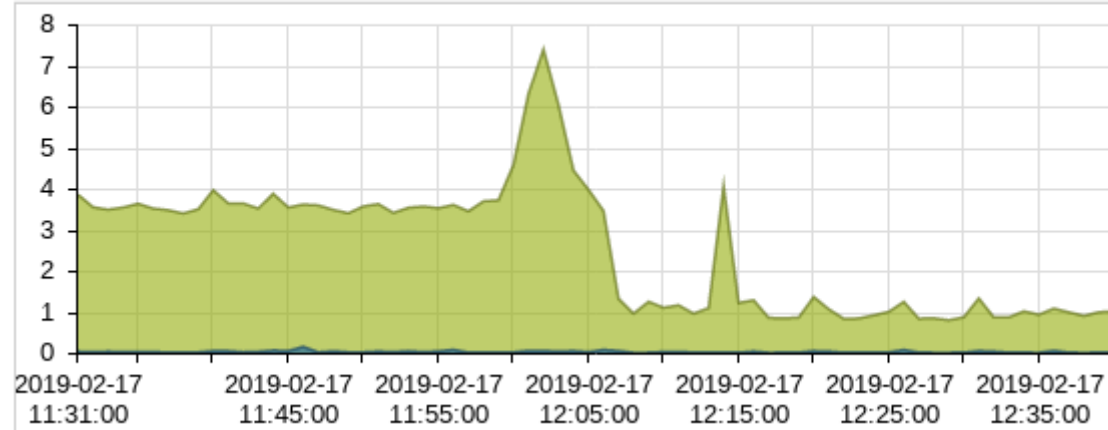
There's a definite decrease in utilization using pypy. Check out these graphs.

CPU usage



● CPU usage ● IO delay

CPU usage



Download from nextcloud
using wget

Get a public nextcloud
share link

append `/download` to the url

`wget` this new url

Install printer drivers for brother printer in opensuse tumbleweed

Get glibc and associated libraries in 32 bit form.

```
sudo zypper install -y glibc-32bit
```

Download brother printer driver installer

```
wget https://download.brother.com/welcome/dlf006893/linux-brprinter-installer-2.2.1-1.gz  
gunzip linux-brprinter-installer-*.gz  
sudo bash linux-brprinter-installer-*
```

See [this page](#) for more information with regard to troubleshooting linux printer drivers.

Install numpy-blis for amd efficiency

```
conda create -c conda-forge -n numpy-blis numpy "blas*=blis" python=3.7
```

Allennlp

Fix errors with empty params list

The error `ValueError: optimizer got an empty parameter list` from allennlp does not relate to the parameters from your json config file. The parameters in question are the tensors to be optimized per `torch/optim/optimizer.py`. This typically is caused by having a mismatch in field names.

Install GTKWattman on opensuse tumbleweed

This is what I tried. it actually doesn't work. I even tried installing gnome and this still didn't work. not sure what's going on.

```
zypper in python3-gobject
zypper in python3-pycairo-devel
zypper in gobject-introspection-devel
python3 -m venv venv_wattman
source venv_wattman/bin/activatepython -m pip install --upgrade matplotlib setuptools pycairo
git clone https://github.com/BoukeHaarsma23/WattmanGTK.git
cd WattmanGTK
pip -m install -e .
```

How to use Intel's mkl library on AMD systems

By default, the mkl library will pick a very unoptimal path on AMD processors, causing libraries like openblas to perform much better. However, if you set the appropriate environment variables, mkl will be forced to use the appropriate code path for amd cpus like Zen and Epyc.

The solution is to run `export MKL_DEBUG_CPU_TYPE=5` before running your script.

This was pointed out in a comment on [this puget systems article](#)

“ Nice! I have recently seen some people recommending using MKL on AMD, with the MKL_DEBUG_CPU_TYPE environment variable set to 5, as in:

`export MKL_DEBUG_CPU_TYPE=5` This overrides the CPU dispatching in MKL, and forces the AVX2 codepath (the one MKL naturally uses on Intel parts without AVX512), otherwise MKL chooses an unoptimized SSE path with abysmal performance. But with the AVX2 path, MKL performs very well on Zen2, usually even outperforming BLIS and OpenBLAS!

Install pytorch-rocm on bare metal opensuse Tumbleweed

These instructions are adopted from section 4 of [this page](#).

pytorch commit 9d1138afec26a4fe0be74187e4064076f8d45de7 worked for some stuff but pieces of allennlp are incompatible because this is pytorch v1.7.0a0+9d1138a.

I tried with 1.5 and 1.5.1 several times but it wasn't working on opensuse. I could have sworn it worked on ubuntu though.

Add rocm repos

Using the files provided by AMD for SLES SP1 per [the official instructions](#).

```
sudo zypper install dkms
sudo zypper cleansudo zypper addrepo --no-gpgcheck http://repo.radeon.com/rocm/zyp/zypper/rocm
sudo zypper ref
sudo zypper install rocm-dkms
sudo reboot
```

Modify `/etc/modprobe.d/10-unsupported-modules.conf` to have

```
allow_unsupported_modules 1
```

Then run the following, though it's probably not strictly necessary on tumbleweed

```
sudo modprobe amdgpu
```

Add your user to the video group

```
usermod -a -G video <username>
```

Verify everything is working by examining the output of `rocminfo` and make sure your gpu is listed.

Create a virtual environment

```
virtualenv -p python3 ~/venvs/torch  
source ~/venvs/torch/bin/activate
```

Install pytorch prerequisites

```
sudo zypper in glog-devel python3-pip libopenblas-devel libprotobuf-devel libnuma-devel  
libpthread-stubs0-devel libopencv-devel git gcc cmake make lmdb-devel libleveldb1 snappy-  
devel hiredis-develsudo zypper in rocm-dev rocm-libs miopen-hip hipsparse rocthrust hipcub  
rccl roctracer-dev
```

Fix issues with cmake files for rocm

```
sed -i 's/find_dependency(hip)/find_dependency(HIP)/g'  
/opt/rocm/rocspare/lib/cmake/rocspare/rocspare-config.cmake sed -i  
's/find_dependency(hip)/find_dependency(HIP)/g' /opt/rocm/rocfft/lib/cmake/rocfft/rocfft-
```

```
config.cmake sed -i 's/find_dependency(hip)/find_dependency(HIP)/g'
/opt/rocm/miopen/lib/cmake/miopen/miopen-config.cmake sed -i
's/find_dependency(hip)/find_dependency(HIP)/g' /opt/rocm/roclblas/lib/cmake/roclblas/roclblas-
config.cmake sed -i 's/find_dependency(hip)/find_dependency(HIP)/g'
/opt/rocm/rccl/lib/cmake/rccl/rccl-config.cmake sed -i
's/find_dependency(hip)/find_dependency(HIP)/g'
/opt/rocm/hipsparse/lib/cmake/hipsparse/hipsparse-config.cmake
```

Clone the repo

```
git clone https://github.com/pytorch/pytorch.git
cd pytorch
git checkout v1.5.0
git submodule update --init --recursive
```

Build

This process will take a while (2-3 hours)

```
export RCCL_DIR="/opt/rocm/rccl/lib/cmake"
python tools/amd_build/build_amd.pyUSE_ROCM=1 USE_LMDB=1 USE_OPENCV=1 MAX_JOBS=4 python
setup.py install
```

Install allennlp

```
pip install allennlp
pip uninstall torch
# rebuild torch (very fast this time)USE_ROCM=1 USE_LMDB=1 USE_OPENCV=1 MAX_JOBS=4 python
setup.py install
```

Verify installation

Make sure you get a non-zero value (should correspond to the number of GPUs).

```
python
>>import torch
>>torch.cuda.device_count()
```

Install aftermarket cooler on M40

<http://translate.google.com/translate?hl=en&sl=auto&tl=en&u=https%3A%2F%2Ftweakers.net%2Ftesla-m40-24gb.html>

Use Intel Quad bypass cards (82571EB PRO)

Download latest intel e1000e drivers

These drivers are available on [sourceforge](#).

Identify the parameters of your particular adapter

Identify the exact model you have

Run `sudo lshw -C network` and identify the full name of your network card. Mine ended up being `82571EB PRO/1000 AT`.

Then, look in `/usr/share/misc/pci.ids` for that code and pull out the pci-id for that code (in this case `10a0`).

Modify drivers & compile

Uncompress the tar'd driver.

```
tar -xzf
```

Move into `src`

```
cd src
```

Modify the definition of `E1000_DEV_ID_82571EB_QUAD_COPPER` in `hw.h`.

```
echo "#define E1000_DEV_ID_82571EB_QUAD_COPPER 0x10A0" >>hw.h
```

Compile the module and install

```
sudo make install
```

reload module

```
sudo modprobe -r e1000e; sudo modprobe e1000e
```

Your nics should show up as `DISABLED` instead of `UNCLAIMED`.

Build pytorch with rocm on ubuntu 20.04

I used rocm 3.8, pytorch 1.6 from the rocm repo (apparently upstream doesn't work right now.).

Enable hibernation on asus zenbook with ryzen processor

I was having an issue where I was unable to hibernate my asus zenbook with ryzen 5 4500u and mx350 gpu. However, the issue turned out to be that dracut was not compiling the resume module in and thus it was unable to resume from whatever image is saved to swap.

[This reddit post](#) was helpful for finding a fix.

Largely, it comes down to the following two commands

```
sudo echo "add_dracutmodules+=\" resume \"" > /etc/dracut.conf.d/99-fix-resume.confsudo  
dracut -fv
```

Set up zyxel travel router in client bridged mode

[Tom's hardware thread where this was discussed](#)

Solution update: To enable Client mode (to bridge my existing wifi network to Ethernet), follow these steps:

- Flip the switch on the ZyXEL's side to AP mode.
- Log in to the ZyXEL's setup page @ 192.168.100.1 (the directions recommend giving your computer a static IP address first, 192.168.100.10)
- From the sidebar, click Wireless>Basic Settings
- Look for the "wireless mode" setting halfway down; click where it says "AP" and pick "Client" from the dropdown menu and then click "apply"
- From the sidebar, click Wireless>Site Survey
- Click the Site Survey button on this page to refresh the list of wifi networks in range.
- Click the bubble to the right of your network, and then the Next button to get the password screen
- Match your encryption method, key length, and type of password.
- Click finish, and if you typed in your password and settings riiiight... You're done!

Get my EGPU to work on opensuse tumbleweed with a framework 12th gen laptop

- With the laptop on, plug it in to the usbc cable
- load into a terminal and load the nvidia module `sudo modprobe nvidia`
- Verify that gpu is working using `nvidia-smi`. This may require installing nvidia-compute-utils
- Run `prime-select nvidia`
- log out of desktop
- start a terminal (ctrl alt f2)
- login
- run `startx`