Edit

ONT Laptop Installation #2

zwets started this conversation in **Show and tell**

zwets on Nov 15, 2024 Maintainer edited

Dear all,

The below captures a fast and reliable way to set up a new laptop with Ubuntu 24.04 or 22.04, the ONT software stack and Nvidia drivers, Flye, Medaka and EPI2ME. If you run into issues or see improvements, drop a comment or file an issue!

> **Note** the procedure below was tested on Ubuntu 22.04 (jammy) and partly on Ubuntu 24.04 (noble). We haven't end-to-end tested 24.04, but expect this to work. The one change required is to replace `jammy` with `noble` in the step where you set up the ONT repositories.

Hardware choice

- Pick a Dell model that is certified for Ubuntu or has Ubuntu pre-installed, e.g. XPS 16, Precision 3591, 5690, 7680, or a gaming (Alienware) model
  - Avoid HP "Ubuntu certified" laptops; they require you to install Windows just to upgrade the BIOS.
  - Dell BIOS upgrades can on many models be installed from within Ubuntu with the `fwupdmgr` command.
  - Updating the BIOS is recommended, certainly if you experience hardware issues or when the model isn't fresh from the factory
- Check the current ONT requirements, noting that:
  - 64GB is recommended as you will want to run other bioinformatics software on the laptop
  - 2TB SSD is recommended, both for storing runs and for databases used by bioinformatics tools
  - Nvidia card: prefer extra memory (e.g. 12GB or 20GB) over more compute power (higher model numbers)

Ubuntu Install

Lauch the installer for Ubuntu 22.04 (or 24.04) from a USB stick, install with these settings:

- Minimal install
- Do not connect to WiFi
- Do not update during install
- Do not activate third party sources

When completed and *before reboot*, in a terminal:

```
sudo dpkg --remove-architecture i386
```

Now connect to WiFi, then update the system

```
sudo apt update && sudo apt dist-upgrade
```

Then reboot

```
sudo reboot
```

User Setup

After reboot, log in as the newly created user.

Open the Settings application:

- In section "Power" disable *Automatic Suspend* and *Automatic Power Saver*

Optionally (but certainly if the laptop will be enabled for unsupervised remote control using AnyDesk or sshrew), disable WiFi powerdown by editing `/etc/NetworkManager/conf.d/default-wifi-powersave-on.conf` to have this:

```
[connection]
wifi.powersave = 2
```

Create directory `~/bin` for holding (symlinks to) user-installed software and scripts:

```
mkdir -p ~/bin
```

Log out and in again, so that `~/bin` is on your `PATH`.

Nvidia Install

Install the Nvidia repositories:

```
wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/
```

```
x86_64/cuda-keyring_1.1-1_all.deb &&
sudo apt install ./cuda-keyring_1.1-1_all.deb
```

Update the available packages with those from Nvidia:

```
sudo apt update
```

List available Nvidia driver versions:

```
apt list 'nvidia-headless-*-open'
```

Assuming the laptop has a recent GPU, the latest release of the "open" variant is generally the recommended driver. (On older hardware, run `ubuntu-drivers devices` to find out.)

Install the Nvidia driver and libraries. **Note** on a *new* laptop a dialog may pop up asking you to provide a one time password for the "MOK". Pick `12345678` and continue.

```
# Replace both '580' by the version number you obtained above
sudo apt install nvidia-headless-580-open nvidia-utils-580
```

Install the CUDA cuDNN library

```
sudo apt install libcudnn9-dev   # may need to append -cuda-13 (or higher)
```

When installation is done, reboot.

Important: if you just set a one time password then **keep watching the screen** during start-up and

- Once a blue screen with a tiny menu appears, arrow down to **"Enrol MOK"** and press Enter.
- Follow the prehistoric prompts to Enroll the MOK, providing the `12345678` password when asked.
- The final prompt has just the option "Reboot". This is good.

When back in the system, check that the Nvidia driver is now loaded:

```
nvidia-smi
```

This should show an overview of the properties of the GPU.

ONT Install

Install the ONT software repositories:

> **Note** these instructions are based on ONT's outdated instructions. They fix the release names and the deprecated use of `apt-key` and cross-repository keys. Please check if ONT have since updated their instructions and consider following those if they have.

```
# Note: replace 'jammy' by 'noble' on Ubuntu 24.04
wget -O- https://cdn.oxfordnanoportal.com/apt/ont-repo.pub |
    sudo tee /etc/apt/keyrings/ont-repo.asc &&
echo "deb [signed-by=/etc/apt/keyrings/ont-repo.asc] http://
cdn.oxfordnanoportal.com/apt jammy-stable non-free" |
    sudo tee /etc/apt/sources.list.d/nanoporetech.sources.list
```

Finally, the MinKNOW installation:

```
sudo apt update &&
sudo apt install ont-standalone-minknow-gpu-release
```

You can now run MinKNOW. To locate it, press the ~~Windows~~ Ubuntu button briefly, start typing "minknow" and its icon should appear. Click it to start MinKNOW.

When MinKNOW is running, rightclick its icon in the dash and choose "Pin to dash". This makes it easy to find it the next time.

Dorado Upgrade

The MinKNOW installation in the previous step comes with Dorado. However, the bundled version trails considerably behind ONT's recommended version (why don't they bundle that instead? 🙄). Hence, we install "standalone Dorado".

Download standalone Dorado:

- Visit https://github.com/nanoporetech/dorado
- In the Installation Section download the `dorado-$VER-linux-x64` installer
- The below instructions assume you downloaded version `$VER` into `~/Downloads`

Install Dorado:

```
VER=1.1.1   # change this to your downloaded release number
sudo mkdir -p /opt/dorado &&
sudo tar -C /opt/dorado -xzf ~/Downloads/dorado-$VER-linux-x64.tar.gz &&
sudo ln -sfT /opt/dorado/dorado-$VER-linux-x64 /opt/dorado/latest &&
sudo ln -sft /usr/local/bin /opt/dorado/latest/bin/dorado
```

The `dorado` command in `/usr/local/bin` is now what runs when you type `dorado` (because `/usr/local/bin` precedes `/usr/bin` in `$PATH`). Check:

```
dorado --version
# in my case shows:
```

```
[2025-08-14 00:16:33.539] [info] Running: "--version"
1.1.1+e72f1492
```

> **TODO**: figure out if MinKNOW can be made to use this standalone Dorado for in-run basecalling. Until that is the case, there is little use in doing HAC or SUP basecalling during the run, as you'll want to redo this with the standalone basecaller.

You are now ready to basecall pod5 to fastq. Instructions for running Dorado are in the document Running Dorado, Flye and Medaka

Flye Install

Flye is the recommended assembler for Nanopore reads. It is a simple `apt` install:

```
apt install flye
```

> This installs 2.9.3 on Ubuntu 22.04, or 2.9.4 on Ubuntu 24.04. Both are functionally identical to Flye's current (= Feb 2025) latest release 2.9.5. If a significant new release comes out in the future, we may install from source.

You are now ready to assemble your fastq files! Instructions for running Flye are in Running Dorado, Flye and Medaka.

Conda Install

Before we can install Medaka, we need to install Conda:

```
cd  # go to your home directory
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
sh ./Miniconda3-latest-Linux-x86_64.sh
```

The installer will ask you:

- To review the licence agreement: press Enter, read everything, then type `yes` if you agree
- Installation location: accept the default (`~/miniconda3`)
- Ignore the confusing blurb that ends with *You can undo this by running* `conda init --reverse $SHELL` *?* Press Enter.

Make the `conda` command available by default:

```
eval "$($HOME/miniconda3/bin/conda shell.bash hook)"
conda init
```

Note how your prompt now starts with `(base)` , due to Conda unhelpfully activating its base environment by default. We do not want this, as it overrides the perfectly fine Python interpreter that every Linux distribution comes with. So:

```
conda config --set auto_activate_base false
```

> **Remember**
>
> - Whenever you encounter installation instructions that tell you to install software with `pip install` or `python setup.py install`, *always create and activate a new Conda environment* to run these commands in (see the Medaka example below)
> - Never install anything in Conda's `base` environment. The `base` environment is for the `conda` command and its dependencies. Adding packages to it defeats the whole purpose of Conda (namely to have isolated environments) *and* will likely break your `conda` command.
> - Never install Conda for the root user, i.e. never run `sudo conda`. Contrary to what you might think, this does not install packages for all users (the way `sudo apt install` does). Instead it installs them in a Conda environment that is accessible to `root` only, and hence useless. (Moreover: `auto-activate base` for the root user will most likely render your Ubuntu system unbootable!)

Medaka Install

Medaka is ONT's recommended polisher for Flye assemblies. We follow the first installation method in the README: install using `pip`, but we (of course!) do this *inside its own Conda environment*.

> The Medaka README mentions installation directly through Conda using the `nanoporetech` channel, but this fails on missing dependencies on Ubuntu 24.04. We may reconsider this option later.

Install the required system packages listed in the README:

```
# In recent Ubuntu releases bgzip is included in tabix and you can omit it
sudo apt install samtools tabix minimap2 bgzip
```

Create a new Conda environment called `medaka` with `python` and `pip` inside:

```
# Python version restriction to match that of Medaka
conda create -n medaka 'python>3.9' pip
```

Activate the environment, then use pip to install medaka inside:

```
conda activate medaka
pip install 'medaka>=2.2.0'   # 2.2.0 was current in Dec 2025
```

Repeat the previous step when upgrading to a newer Medaka version.

Work around Medaka bug

At the time of writing, this bug prevents Medaka from running. I have submitted this PR as a fix. If that PR is still open, here is a workaround hack:

```
# Hand edit the offending file
nano ~/miniconda3/envs/medaka/lib/python3.*/site-packages/medaka/__init__.py
```

Look for this line:

```
first_line = proc.stdout.decode().split("\n", 1)[0]
```

Replace it with:

```
first_line = proc.stdout.decode('utf-8',errors='replace').split("\n", 1)[0
```

You now have a working `medaka`. Running instructions are in our document Running Dorado, Flye and Medaka.

(Optional) Podman Install

This is an **optional** step. (Removed until EPI2ME is compatible with with `podman-docker`.)

Podman is the drop-in replacement for Docker that is suitable for running end user software (contrary to Docker which is intended for system services that run with root privileges). To install:

```
sudo apt install podman
```

Podman commands are identical to Docker commands (just replace `docker` by `podman`):

```
podman run --rm hello-world
podman image ls
```

You now have Podman available on the system, and can run any Docker container.

Docker Install

~~Instead of Docker we install the `podman-docker` package, which simply adds a `docker` alias to the `podman` command:~~

Unfortunately EPI2ME fails to run with `podman-docker` (reported to ONT), and we need to install `docker.io` instead:

```
sudo apt install --no-install-recommends docker.io
```

Add the logged-in user to the `docker` group:

```
sudo adduser $USER docker
```

Log out and in again so that your membership of the `docker` group is applied to your session. Then test:

```
docker image ls
```

This command should have empty output. *If* it shows the image we added through `podman run`, then the packaging bug noted below applies on your system. You can ignore this for now. Test docker:

```
docker run --rm hello-world
```

You now have Docker installed and EPI2ME will be happy.

Nvidia Container Toolkit

This step is required in order to allow containers to run GPU-based tools (as EPI2ME does)

```
sudo apt install nvidia-container-toolkit
```

Add the CDI configuration

```
sudo mkdir /etc/cdi &&
sudo nvidia-ctk cdi generate --output=/etc/cdi/nvidia.yaml
```

Check the list of devices

```
nvidia-ctk cdi list
```

This should show at least `nvidia.com/gpu=all`. Try accessing the GPU from within a docker/podman container:

```
docker run --rm --gpus all ubuntu nvidia-smi
podman run --rm --gpus all ubuntu nvidia-smi
```

You can now run containers with access to the GPU.

(Optional) Nextflow Install

This step is optional. EPI2ME comes with its own Nextflow installation, but a standalone Nextflow installation may be convenient for diagnosing issues or running EPI2ME workflows from the command-line.

Nextflow requires Java:

```
sudo apt install default-jre-headless
```

Install the `nextflow` command in `~/bin`:

```
ls -d ~/bin || { mkdir ~/bin; echo Log out and in again before continuing;
wget -O ~/bin/nextflow https://get.nextflow.io &&
chmod +x ~/bin/nextflow
```

Smoke test:

```
nextflow run hello
```

You now have a working Nextflow installation (no configuration needed, but see comment below).

EPI2ME Install

Follow
TBD (a number of issues are still being resolved, instructions here will come shortly).

This completes the "ONT laptop installation" for now.

Additional Tools (under construction)

Here is general guidance on installing additional tools on the system:

- Always prefer `apt install` over anything else
- Then prefer a Podman (Docker) container or a Conda install (whichever the tool offers as a standard)
- Then if the tool offers a `pip` install, create and activate a Conda environment and run pip inside it
- *more to come (need to copy this from earlier notes)*

↑ 1

5 comments · 5 replies

Oldest Newest | Top

zwets on Nov 15, 2024 Maintainer Author edited

On the Dell XPS 16 models with glitchy screen (when you move your finger over the touchpad), do this:

```
sudo sed -i -Ee \
    's/^GRUB_CMDLINE_LINUX=""$/GRUB_CMDLINE_LINUX="i915.enable_psr=0"/' \
    /etc/default/grub &&
sudo update-grub
```

The issue should be gone after a reboot.

↑ 1 ☺ 0 replies

Write a reply

nrt94 on Dec 3, 2024

should we perhaps also add how they can install epi2me and docker? especially how they change docker to run on the user correctly and not do the screw-ups I did of chmod

↑ 1 ☺ 2 replies

zwets on Dec 10, 2024 Maintainer Author

Good one. I'll need to go through those on my new laptop and will keep notes.

I'm hoping Epi2ME can work with Podman instead of Docker, or at least with rootless Docker. That saves us from a lot of the permissions headaches ...

☺

zwets on Dec 10, 2024 Maintainer Author

I have also just tracked down the bug that prevents Medaka 2.0.1 from running on a standard Ubuntu installation. That latest version is interesting because it has a `--bacteria` option that picks a model optimized for bacterial isolates and plasmids.

☺

Write a reply

zwets on Dec 12, 2024 Maintainer Author

One more note **@nrt94** : ONT have just released Dorado 0.7.4, mentioning in the release notes) that this is only to match the one that comes with the latest (24.11) release of MinKNOW.

ONT recommend users to only pick that release if they want to "match the performance of MinKNOW 24.11" but should otherwise prefer the actual latest version of Dorado (currently 0.8.3). This will also be relevant for our installation: probably better to use standalone Dorado, rather than use the one that comes with the MinKNOW installation.

1 1 reply

zwets on Jan 15 Maintainer Author

I have just added the instruction to install "standalone Dorado" to the document above. An open issue remains that if we cannot make MinKNOW to use the standalone basecaller, then **there is no point to do in-run HAC or SUP basecalling** as it will need redoing anyway.

Write a reply

zwets on Feb 9 Maintainer Author

Regarding Nextflow configuration:

I don't understand Nextflow's obsession with hidden files and directories. It's fine that it produces a log for every invocation (even `nextflow help` writes a `.nextflow.log` file), but why make it invisible? It's either transient and then should go in `/tmp` or `/run` where it gets cleaned up, or it is important enough to keep, but then why hide it?

My preference is to make such files visible, so I set `NXF_LOG_FILE=nextflow.log` in `~/.profile`. I do the same for `NXF_HOME`, `NXF_WORK` and `NXF_CACHE_DIR`, all of which default to hidden directories. I suppose this is a matter of taste, much like whether you prefer dirt swept under a carpet or piled up in a corner.

Either way, remember to run `nextflow clean` when you are done with a workflow. The hidden cache and work directories tend to be massive for the kinds of workflows we run.

1 0 replies

Write a reply

nrt94 on Feb 11

i think these lines are wrong:
sudo apt update &&
sudo apt install ont-standalone-minknow-gpu-release

it is:
sudo apt install ont-standalone-minknow-release

↑ 1

2 replies

zwets on Feb 11 Maintainer Author

Let me check. ONT used to have separate installation packages for CPU and GPU, but they also tend to rename the packages every so often 🙄

zwets on Feb 11 Maintainer Author

Checked, and yup they renamed it once again ... 🙄

The one with "gpu" in the name is still in the ONT package list, but points at the 24.06 version, which is no longer there. The one without "gpu" now points at the 24.11. I'll fix this in the instructions. I wish they'd stop doing this.

Write a reply

Category

🙌 **Show and tell**

Labels

None yet

2 participants