Get Started with the Intel® AI Analytics Toolkit for Linux*

Contents

Get Started with the Intel® AI Analytics Toolkit

The following instructions assume you have installed the Intel® oneAPI software. Please see the Intel AI Analytics Toolkit page for installation options.

Follow these steps to build and run a sample with the Intel® AI Analytics Toolkit (AI Kit):

1. Configure your system.
2. Build and Run a Sample.

> **NOTE** Standard Python installations are fully compatible with the AI Kit, but the Intel® Distribution for Python* is preferred.

No special modifications to your existing projects are required to start using them with this toolkit.

Components of This Toolkit

The AI Kit includes:

- **Intel® Optimization for PyTorch*:** The Intel® oneAPI Deep Neural Network Library (oneDNN) is included in PyTorch as the default math kernel library for deep learning.
- **Intel® Extension for PyTorch:**Intel® Extension for PyTorch* extends PyTorch* capabilities with up-to-date features and optimizations for an extra performance boost on Intel hardware.
- **Intel® Optimization for TensorFlow*:** This version integrates primitives from oneDNN into the TensorFlow runtime for accelerated performance.
- **Intel® Extension for TensorFlow:** Intel® Extension for TensorFlow* is a heterogeneous, high performance deep learning extension plugin based on TensorFlow PluggableDevice interface. This extension plugin brings Intel XPU (GPU, CPU, etc) devices into the TensorFlow open source community for AI workload acceleration.
- **Intel® Distribution for Python*:** Get faster Python application performance right out of the box, with minimal or no changes to your code. This distribution is integrated with Intel® Performance Libraries such as the Intel® oneAPI Math Kernel Library and the Intel®oneAPI Data Analytics Library.
- **Intel® Distribution of Modin* (available through Anaconda only)**, which enables you to seamlessly scale preprocessing across multi nodes using this intelligent, distributed dataframe library with an identical API to pandas. This distribution is only available by Installing the Intel® AI Analytics Toolkit with the Conda* Package Manager.
- **Intel® Neural Compressor** : quickly deploy low-precision inference solutions on popular deep-learning frameworks such as TensorFlow*, PyTorch*, MXNet*, and ONNX* (Open Neural Network Exchange) runtime.
- **Intel® Extension for Scikit-learn***: A seamless way to speed up your Scikit-learn application using the Intel® oneAPI Data Analytics Library (oneDAL).

  Patching scikit-learn makes it a well-suited machine learning framework for dealing with real-life problems.
- **XGBoost Optimized by Intel**: This well-known machine-learning package for gradient-boosted decision trees includes seamless, drop-in acceleration for Intel® architectures to significantly speed up model training and improve accuracy for better predictions.

Configure Your System – Intel® AI Analytics Toolkit

If you have not already installed the AI Analytics Toolkit, refer to Installing the Intel® AI Analytics Toolkit.

To configure your system, set environment variables before continuing.

	All Users	Conda Users	GPU Users	Conda + GPU Users
Set Environment Variables	X	X	X	X
Use Conda to Add Packages		X		X
Install Graphics Drivers, Add User to Video Group, and Disable Hangcheck			X	X

Set Environment Variables for CLI Development

For working at a Command Line Interface (CLI), the tools in the oneAPI toolkits are configured via environment variables. To set environment variables bysourcing the `setvars` script:

Option 1: Source setvars.sh once per session

Source `setvars.sh` every time you open a new terminal window:

You can find the `setvars.sh` script in the root folder of your oneAPI installation, which is typically `/opt/intel/oneapi/` for system wide installations and `~/intel/oneapi/` for private installations.

For system wide installations (requires root or sudo privileges):

```
. /opt/intel/oneapi/setvars.sh
```

For private installations:

```
. ~/intel/oneapi/setvars.sh
```

Option 2: One time setup for setvars.sh

To have the environment automatically set up for your projects, include the command `source <install_dir>/setvars.sh` in a startup script where it will be invoked automatically (replace `<install_dir>` with the path to your oneAPI install location). The default installation locations are `/opt/intel/oneapi/` for system wide installations (requires root or sudo privileges) and `~/intel/oneapi/` for private installations.

For example, you can add the `source <install_dir>/setvars.sh` command to your `~/.bashrc` or `~/.bashrc_profile` or `~/.profile` file. To make the settings permanent for all accounts on your system, create a one-line `.sh` script in your system's `/etc/profile.d` folder that sources `setvars.sh` (for more details, see Ubuntu documentation on Environment Variables).

> **NOTE**
> The setvars.sh script can be managed using a configuration file, which is especially helpful if you need to initialize specific versions of libraries or the compiler, rather than defaulting to the "latest" version. For more details, see Using a Configuration File to Manage Setvars.sh.. If you need to setup the environment in a non-POSIX shell, seeoneAPI Development Environment Setup for more configuration options.

Next Steps

- If you are not using Conda, or developing for GPU, Build and Run a Sample Project.
- For Conda users, continue on to the next section.
- For developing on a GPU, continue on to GPU Users

Conda Environments in this Toolkit

There are multiple conda environments included in the AI Kit. Each environment is described in the table below. Once you have set environment variables to CLI environment as previously instructed, you can then activate different conda environments as needed via the following command:

```
conda activate <conda environment>
```

For more information, please explore each environment's related Getting Started Sample linked in the table below.

Conda Environment Name	Note	Getting Started Sample
tensorflow	Intel TensorFlow (CPU)	Sample
tensorflow-gpu	Intel TensorFlow with Intel Extension for TensorFlow (GPU)	Sample
pytorch	PyTorch with Intel Extension for PyTorch (XPU) Intel oneCCL Bindings for PyTorch (CPU)	Intel Extension for PyTorch Sample,Intel oneCCL Bindings for PyTorch Sample
Pytorch-gpu	PyTorch with Intel Extension for PyTorch (XPU) Intel oneCCL Bindings for PyTorch (CPU)	Intel Extension for PyTorch Sample,Intel oneCCL Bindings for PyTorch Sample
base	Intel Distribution for Python	Sample
modin	Intel Distribution of Modin	Sample
For more samples, browse the full GitHub repository: Intel® oneAPI AI Analytics Toolkit Code Samples.		

Use the Conda Clone Function to Add Packages as a Non-Root User

The Intel AI Analytics toolkit is installed in the `oneapi` folder, which requires root privileges to manage. You may wish to add and maintain new packages using Conda*, but you cannot do so without root access. Or, you may have root access but do not want to enter the root password every time you activate Conda.

To manage your environment without using root access, utilize the Conda clone functionality to clone the packages you need to a folder outside of the `/opt/intel/oneapi/` folder:

1. From the same terminal window where you ran `setvars.sh`, identify the Conda environments on your system:

```
conda env list
```

   You will see results similar to this:

```
# conda environments:
#
base                  *  /opt/intel/oneapi/intelpython/latest
2023.1                   /opt/intel/oneapi/intelpython/latest/envs/2023.0
pytorch                  /opt/intel/oneapi/intelpython/latest/envs/pytorch
Pytorch-gpu              /opt/intel/oneapi/intelpython/latest/envs/pytorch-gpu
tensorflow               /opt/intel/oneapi/intelpython/latest/envs/tensorflow
tensorflow-gpu           /opt/intel/oneapi/intelpython/latest/envs/tensorflow-gpu
modin                    /opt/intel/oneapi/intelpython/latest/envs/modin
```

2. Use the clone function to clone the environment to a new folder. In the example below, the new environment is named `usr_intelpython` and the environment being cloned is named `base` (as shown in the image above).

```
conda create --name usr_intelpython --clone base
```

   The clone details will appear:

```
(base) -bash.4.3$ conda create --name usr_intelpython --clone base
Source: /opt/intel/oneapi/intelpython/latest
Destination: /___/home/.conda/envs/usr_intelpython
```

If the command does not execute, you may not have access to the `~/.conda` folder. To fix this, delete the `.conda` folder and execute this command again: `conda create --name usr_intelpython --clone base`.

3. Activate the new environment to enable the ability to add packages.

```
conda activate usr_intelpython
```

4. Verify the new environment is active.

```
conda env list
```

You can now develop using the Conda environment for Intel Distribution for Python.

5. To activate the TensorFlow* or PyTorch* environment:

TensorFlow:

```
conda activate tensorflow
```

PyTorch:

```
conda activate pytorch
```

Next Steps

- If you are not developing for GPU, Build and Run a Sample Project.
- For developing on a GPU, continue on to GPU Users.

GPU Users

For those who are developing on a GPU, follow these steps:

1. Install GPU drivers

If you followed the instructions in the Installation Guide to install GPU Drivers, you may skip this step. If you have not installed the drivers, follow the directions in the Installation Guide.

2. Add User to Video Group

For GPU compute workloads, non-root (normal) users do not typically have access to the GPU device. Make sure to add your normal user(s) to the video group; otherwise, binaries compiled for the GPU device will fail when executed by a normal user. To fix this problem, add the non-root user to the video group:

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

3. Disable Hangcheck

For applications with long-running GPU compute workloads in native environments, disable hangcheck. This is not recommended for virtualizations or other standard usages of GPU, such as gaming.

A workload that takes more than four seconds for GPU hardware to execute is a long running workload. By default, individual threads that qualify as long-running workloads are considered hung and are terminated. By disabling the hangcheck timeout period, you can avoid this problem.

> **NOTE** If the kernel is updated, hangcheck is automatically enabled. Run the procedure below after every kernel update to ensure hangcheck is disabled.

1. Open a terminal.
2. Open the grub file in /etc/default.
3. In the grub file, find the line GRUB_CMDLINE_LINUX_DEFAULT="" .
4. Enter this text between the quotes (""):

```
i915.enable_hangcheck=0
```

5. Run this command:

```
sudo update-grub
```

6. Reboot the system. Hangcheck remains disabled.

Next Step

Now that you have configured your system, proceed to Build and Run a Sample Project.

Build and Run a Sample Using the Command Line

Intel® AI Analytics Toolkit

In this section, you will run a simple "Hello World" project to familiarize yourself with the process of building projects, and then build your own project.

> **NOTE** If you have not already configured your development environment, go to Configure your system then return to this page. If you have already completed the steps to configure your system, continue with the steps below.

You can use either a terminal window or Visual Studio Code* when working from the command line. For details on how to use VS Code locally, see Basic Usage of Visual Studio Code with oneAPI on Linux*. To use VS Code remotely, see Remote Visual Studio Code Development with oneAPI on Linux*.

Build and Run a Sample Project

The samples below must be cloned to your system before you can build the sample project:

Name of Sample	Description	How to Clone and Build
Intel Extension for PyTorch Getting Started, Intel oneCCL Bindings for PyTorch	Train a PyTorch model and run the inference with the Intel® Deep Neural Network Library (Intel® DNNL) enabled. Intel® Extension for PyTorch* extends PyTorch* with optimizations for extra performance boost on Intel hardware.	Clone oneCCL Bindings for PyTorch or Intel Extension for PyTorch sample, then follow the directions in README.md to build and run the sample.
TensorFlow Hello World, Intel Extension for TensorFlow Getting Started	TensorFlow optimized on Intel hardware enables Intel® DNNL calls by default. It implements an example neural network with one convolution layer and one ReLU layer. Intel® Extension for TensorFlow* is a heterogeneous, high performance deep learning extension plugin based on TensorFlow PluggableDevice interface. This extension plugin brings Intel XPU (GPU, CPU, etc) devices into the TensorFlow open source community for AI workload acceleration.	Clone TensorFlow_HelloWorld or Intel Extension for TensorFlow sample, then follow the directions in README.md to build and run the sample.
Intel® Distribution of Modin* Getting Started	This Getting Started sample code shows how to use distributed Pandas using the Modin package.	To get the Intel® Distribution of Modin*, you must install the AI Kit using the Conda* package manager. After the AI Kit is installed with Conda, clone Intel® Distribution of Modin* Getting Started, then follow the directions in README.md to build and run the sample.

Model Zoo for Intel® Architecture	• Demonstrate the AI workloads and deep learning models Intel has optimized and validated to run on Intel hardware • Show how to efficiently execute, train, and deploy models optimized for Intel Architecture • Make it easy to get started running optimized models on Intel hardware in the cloud or on bare metal	Model Zoo for Intel® Architecture can be found in your installation of Intel® oneAPI AI Analytics Toolkit, typically found at `/opt/intel/oneapi/modelzoo/latest/models`. Instructions for navigating the zoo, using the samples, and running the benchmarks are here: https://github.com/IntelAI/models/blob/v2.4.0/docs/general/tensorflow/AIKit.md#navigate-to-the-model-zoo
Intel® Neural Compressor	Intel® Neural Compressor is an open-source Python* library designed to help you quickly deploy low-precision inference solutions on popular deep-learning frameworks such as TensorFlow*, PyTorch*, MXNet*, and ONNX* (Open Neural Network Exchange) runtime.	Clone neural-compressor, then follow the directions in README.md to build and run the sample.
Intel® Extension for Scikit-learn*	Provide a seamless way to speed up your Scikit-learn application using using of the Intel® oneAPI Data Analytics Library (oneDAL).	Clone Intel® Extension for Scikit-learn*, then follow the directions in the README.md to build and run the sample.
For more samples, browse the full GitHub repository: Intel® oneAPI AI Analytics Toolkit Code Samples.		

To see a list of components that support CMake, see Use CMake to with oneAPI Applications.

Build Your Own Project

No special modifications to your existing Python projects are required to start using them with this toolkit. For new projects, the process closely follows the process used for creating sample Hello World projects. Refer to the Hello World README files for instructions.

Maximizing Performance

You can get documentation to help you maximize performance for either TensorFlow or PyTorch.

Configure Your Environment

> **NOTE** If your virtual environment is not available, or if you wish to add packages to your virtual environment, ensure you have completed the steps in Use the Conda Clone Function to Add Packages as a Non-Root User.

If you are developing outside of a container, source the following script to use the Intel® Distribution for Python*:

```
. <install_dir>/setvars.sh
```

where `<install_dir>` is where you installed this toolkit. By default the install directory is:

Root or sudo installations: `/opt/intel/oneapi`

Local user installations: `~/intel/oneapi`

NOTE The setvars.sh script can be managed using a configuration file, which is especially helpful if you need to initialize specific versions of libraries or the compiler, rather than defaulting to the "latest" version. For more details, see Using a Configuration File to Manage Setvars.sh. If you need to setup the environment in a non-POSIX shell, see oneAPI Development Environment Setup for more configuration options.

To switch environments, you must first deactivate the active environment.

The following example demonstrates configuring the environment, activating TensorFlow*, and then returning to the Intel Distribution for Python:

```
.
conda activate tensorflow
conda deactivate
conda activate root
```

Download a Container

Intel® AI Analytics Toolkit

Containers allow you to set up and configure environments for building, running and profiling oneAPI applications and distribute them using images:

- You can install an image containing an environment pre-configured with all the tools you need, then develop within that environment.
- You can save an environment and use the image to move that environment to another machine without additional setup.
- You can prepare containers with different sets of languages and runtimes, analysis tools, or other tools, as needed.

Download Docker* Image

You can download a Docker* image from the Containers Repository.

NOTE The Docker image is ~5 GB and can take ~15 minutes to download. It will require 25 GB of disk space.

1. Define the image:

```
image=intel/oneapi-aikit
docker pull "$image"
```

2. Pull the image.

```
docker pull "$image"
```

Once your image is downloaded, proceed to Using Containers with the Command Line.

Using Containers with the Command Line

Intel® AI Analytics Toolkit

Download pre-built containers directly. The command below for CPU will leave you at a command prompt, inside the container, in interactive mode.

CPU

```
image=intel/oneapi-aikit
docker run -it "$image"
```

Using Intel® Advisor, Intel® Inspector or VTune™ with Containers

When using these tools, extra capabilities have to be provided to the container:

`--cap-add=SYS_ADMIN --cap-add=SYS_PTRACE`

```
docker run --cap-add=SYS_ADMIN --cap-add=SYS_PTRACE \
--device=/dev/dri -it "$image"
```

Using Cloud CI Systems

Cloud CI systems allow you to build and test your software automatically. See the repo in github for examples of configuration files that use oneAPI for the popular cloud CI systems.

Troubleshooting for the Intel® AI Analytics Toolkit

Issue	How to fix
Undefined error appears when building a sample.	If the sample was built by using `cmake` and then `make`, more details to diagnose the reason for your error are available by rebuilding with the `VERBOSE` option: `export VERBOSE=1` For more comprehensive troubleshooting, use the Diagnostics Utility for Intel® oneAPI Toolkits, which provides system checks to find missing dependencies and permissions errors. Learn more.
NotWritableError when using the `conda clone` function.	The error is occurring because the function is trying clone to the `~/.conda` folder but the current user does not have access to that folder. To fix this for the AI Toolkit, delete the `.conda` folder and return to the Use the Conda Clone Function to Add Packages as a Non-Root User to run the `conda clone` command again. Deleting the folder may affect the behavior of other toolkits that were previously installed.
Errors due to missing dependencies, missing environment variables or missing machine capabilities.	The Diagnostics Utility for Intel oneAPI Toolkits provides the ability to find missing dependencies and permissions errors and is already installed with this tooklit. Learn more.
Errors that occur during installation or directly after installation.	See the Troubleshooting page of the Intel® oneAPI Toolkits Installation Guide for Linux* OS.

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

Product and Performance Information
Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex. Notice revision #20201201