Intel Data Science Workstation Powered by the AI Kit for Data Science Workstation Installation Guide

The Data Science Workstation lineup from Intel and Intel's OEM partners provides data scientists, data analysts, and developers productive and cost-effective AI development solutions to quickly generate insights for their organizations.

- Open, optimized software tools are coupled with optimal compute and memory hardware configurations to deliver the best out-of-the-box developer experience, whether you are prototyping or developing production AI.
- High-memory systems can fit large datasets for efficient preprocessing, considerably shortening the time required to sort, filter, label, and transform your data.
- Familiar Python*‡ APIs deliver software accelerations of up to 10x to 100x for training and inference.

Components of This Toolkit
The AI Kit includes:

- **PyTorch**: The Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN) is included in PyTorch as the default math kernel library for deep learning. See this article on the Intel® Developer Zone for more details.
- **Intel® Optimization for TensorFlow**: This version integrates primitives from the Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN) into the TensorFlow runtime for accelerated performance.
- **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. The distribution also includes daal4py, a Python module integrated with the Intel® oneAPI Data Analytics Library as well as the Python Data Parallel Processing Library (PyDPPL), a light weight Python wrapper for Data Parallel C++ and SYCL that provides a data parallel interface and abstractions to efficiently tap into device management features of CPUs and GPUs running on Intel® Architecture.
- **Intel® Distribution of Modin**, 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.
- **Low Precision Optimization Tool**: Provide a unified, low-precision inference interface across multiple deep learning frameworks optimized by Intel with this open-source Python library.
• 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 accelerates stock scikit by single line change.

**Installation**

This instance was built using Ubuntu 20.04 LTS. If you are on Windows 11, jump to addendum to setup the environment and come back here to Step 1.

**Step 1:**

Update your system

Launch a terminal (CTRL+ALT+T) and enter

```
sudo apt update && sudo apt upgrade -y
```

**Step 2:**

After the update a reboot will be required, enter:

```
sudo reboot
```

**Step 3:**

Launch a terminal (CTRL+ALT+T) and enter

```
wget https://registrationcenter-download.intel.com/akdlm/irc_nas/18273/Intel-AIkit-2021.4.1-Linux-x86_64.sh
```

The download is approximately 1.8G, a stable network connection is desired.

**Step 4:**

Change the permissions of the resulting download so that it is an executable.

```
chmod 755 Intel-AIkit-2021.4.1-Linux-x86_64.sh
```

Should the version change the product page is [here](#).

**Step 5:**

Install the Intel Alkit

```
./Intel-AIkit-2021.4.1-Linux-x86_64.sh
```

Follow the steps and make sure to choose **yes** to the final question “Do you wish the installer to initialize the Intel-Alkit”

**Step 6:**
You can close the terminal and launch a new one or enter:

```
source .bashrc
```

which will result in the base conda environment being activated.

Enter

```
conda env list
```

Base * /home/user/intel-aikit ← should be the result

Enter

```
conda list
```

You should see a list of packages with many coming from the Intel channel. These are the optimized packages.

**Note**

This collection of packages has been validated against each other for functionality. At this point this is a stable environment and should be treated as the base package environment. Any modifications to this collection should be done in a cloned environment so that changes can be isolated and you can easily get back to a known good.

**Getting Started**

In this section we will:

- Create a new environment and add some additional packages.
- Clone the Intel samples repository.
- Run a sample project using a Jupyter Notebook.

**Create New Environment**

**Step 1:**

Launch a terminal (CTRL+ALT+T) and enter

```
conda create --clone base --name <pick something> for example:
conda create --clone base --name jupyter
```

**Step 2:**

```
conda env list
```

You should see two environments now. The * denotes the active environment.

Activate the new environment:
Conda activate jupyter

If you enter conda env list the asterisks should be next to the new environment name and the prompt should indicate the environment name as well.

**Step 3:**
Add missing packages.

We need several packages for the getting started samples, let’s add those knowing that if we run into issues we can always delete this environment and get back to our clean base environment.

Enter:
conda install -c conda-forge jupyterlab

*Note* there will be some warnings but they are safe to ignore.

Enter:
conda install -c intel neural-compressor (need to file JIRA, this install is really ugly broke recently)

*Note* this takes awhile as conda will check for package conflicts

**Step 4:**
Clone the Intel oneAPI Samples Repository, Git will likely not be installed so to install it enter:
sudo apt install git
git clone https://github.com/oneapi-src/oneAPI-samples.git

current repo size is ~410M

**Step 5:**
Enter:
jupyter lab

The browser should auto launch. Then on the panel on the left navigate to:

~/oneAPI-samples/AI-and-Analytics/Getting-Started-Samples/INC-Sample-for-Tensorflow/

Double click:

Inc_sample_tensorflow.ipynb
As you execute the notebook you will see that your Data Science Workstation supports the latest set of instructions and has the ability to take advantage of INT8 as opposed to FP32 increasing throughput and reducing latency while maintaining identical accuracy.

Please explore the other getting started samples in the git repo.

Addendum: WSL2 on Windows 11:

**Step 1:**
Tap the Windows Key and enter update, windows update will be the first match. Go through that process and reboot.

**Step 2:**
Launch windows terminal with administrative privileges. Tap the Windows key and enter terminal. The down carrot on the right will expose the admin privilege option.

**Step 3:**
At the prompt enter: wsl – install (two dashes)
This will automatically install Ubuntu 20.04
**Step 4:**

Restart  -- The install actually takes place here and after the reboot, this takes a couple of minutes.

**Step 5:**

Ubuntu will launch automatically, follow the steps to create a new user. It’s a good idea to have the identical username for Windows as you do Ubuntu.

**Step 6:**

Jump back up to step 1 and follow along, even the update steps.

**Note** a reboot in WSL2 is exiting the WSL2 session. To relaunch WSL2 tap the Windows key, enter terminal and then click on the down carrot and choose Ubuntu. This is a great way to have multiple terminal sessions in a tabbed environment.