Intel® oneAPI Toolkits Installation Guide for Linux* OS

Contents

Intel® oneAPI Toolkits and Components Installation Guide for Linux* OS

1

This guide covers the installation of Intel® oneAPI toolkits and standalone components on Linux* OS.

Before You Begin

Check the Prerequisites information before installing the Intel oneAPI packages.

Installation

Install Intel oneAPI component and toolkit packages with one of the following options:

- Install with GUI
- Install with Command Line
- Install Using Package Managers
- Install Using Docker Container

Next Steps

Get Intel oneAPI code samples and refer to the toolkit Get Started page for detailed usage instructions, examples, and more:

- Get Started with Intel® oneAPI Base Toolkit
- Get Started with Intel® oneAPI HPC Toolkit
- Get Started with Intel® oneAPI IoT Toolkit
- Get Started with Intel® oneAPI AI Analytics Toolkit
- Get Started with Intel® oneAPI System Bring-up Toolkit
- Get Started with Intel® oneAPI Rendering Toolkit
- Get Started with Intel® oneAPI DL Framework Developers Toolkit

Prerequisites

Consider the following important information before installing the Intel oneAPI packages.

System Requirements

Refer to the toolkit-specific Release Notes and System Requirements documents to learn more about compatibility details:

- Intel® oneAPI Base Toolkit Release Notes | System Requirements
- Intel® oneAPI HPC Toolkit Release Notes | System Requirements
- Intel® oneAPI IoT Toolkit Release Notes | System Requirements
- Intel® oneAPI AI Analytics Toolkit Release Notes | System Requirements
- Intel® oneAPI System Bring-up Toolkit Release Notes | System Requirements
- Intel® oneAPI Rendering Toolkit Release Notes | System Requirements
- Intel® oneAPI DL Framework Developers Toolkit Release Notes | System Requirements

IDE Integration

To use third-party IDE, install Eclipse* on your Linux* OS host system before installing oneAPI Toolkits. This allows you to integrate the plugins as part of the Intel oneAPI Base Toolkit installation.

Use Modulefiles or a Configuration File to Set Environment Variables (optional)

Environment variables can be set up manually (as described in Get Started Guides and sample README files) or automatically using one of the methods below: - Use **modulefiles <https://www.intel.com/content/www/us/en/develop/documentation/oneapi-programming-guide/top/oneapi-development-environment-setup/use-modulefiles-with-linux.html>_** - Use a **setvars.sh configuration file <https://www.intel.com/content/www/us/en/develop/documentation/oneapi-programming-guide/top/oneapi-development-environment-setup/use-the-setvars-script-with-linux-or-macos/use-a-config-file-for-setvars-sh-on-linux-or-macos.html>_**

Set Up Your System for Intel GPU

If you are using Intel GPU, complete the following steps before or after Intel oneAPI installation:

- Install Intel GPU drivers
- Check that you have fulfilled the requirements of Intel® Graphics Compute Runtime for oneAPI Level Zero and OpenCL™ Driver. Make sure that you have permissions to access the `/dev/dri/renderD\*` and `/dev/dri/card\*` files. This typically means that your user account is a member of the video (on Ubuntu* 18, Fedora* 30, and SLES* 15 SP1) or render (on Ubuntu* 19 and higher, CentOS* 8, and Fedora* 31) group. Alternatively, an administrator with sudo or root privilege can change the group owner of `/dev/dri/renderD\*` and `/dev/dri/card\*` to a group ID used by your user base.
- If you have applications with long-running GPU compute workloads in native environments, you must disable the hangcheck timeout period to avoid terminating workloads.
- If you plan to use the Intel® Distribution for GDB* on Linux* OS, make sure to configure debugger access.

- Install Intel GPU Drivers
- GPU: Disable Hangcheck

Install Intel GPU Drivers

If you use Intel GPU, you need to install the latest GPU drivers separately. To install the driver packages, follow the Installation Guide applicable for your Linux distribution.

GPU: Disable Hangcheck

This section applies only to applications with long-running GPU compute workloads in native environments. It 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 system is rebooted, hangcheck is automatically enabled. You must disable hangcheck again after every reboot or follow the directions below to disable hangcheck persistently (across multiple reboots). Please re-run this GPU Hangcheck disable with reboot fix if you update (or auto-update) the kernel.

To disable hangcheck until the next reboot:

```
sudo sh -c "echo N> /sys/module/i915/parameters/enable_hangcheck"
```

To disable hangcheck across multiple reboots:

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 the following text between the quotes (“”):

```
i915.enable_hangcheck=0
```

5. Run the following command:

```
sudo update-grub
```

6. Reboot the system. Hangcheck remains disabled.

Installation

Toolkit Installation

Important Some domain-specific toolkits require you to install the Intel oneAPI Base Toolkit first for full functionality.

Use one of the following options to install a toolkit package:

Toolkit Name	Binary Installer	Package Manager	Docker Container
Intel® oneAPI Base Toolkit	Online/offline installer	YUM, DNF, Zypper, APT	Docker Hub
Intel® oneAPI HPC Toolkit (requires installation of the Intel oneAPI Base Toolkit)	Online/offline installer	YUM, DNF, Zypper, APT	Docker Hub
Intel® oneAPI IoT Toolkit (requires installation of the Intel oneAPI Base Toolkit)	Online/offline installer	YUM, DNF, Zypper, APT	Docker Hub
Intel® oneAPI AI Analytics Toolkit	Online/offline installer	YUM, DNF, Zypper, APT, Conda	Docker Hub
Intel® oneAPI Rendering Toolkit	Online/offline installer	YUM, DNF, Zypper, APT	N/A

Each of the binary installer types operates in various modes, which are covered later in this section.

NOTE When using the offline installer, you can enable full offline mode by setting the environment variable `INTEL_SUPPRESS_INTERNET_CONNECTION=1`. When this mode is enabled:

- Installer does not send installation statistics
- Download-only mode is disabled
- Upgrade mode is disabled

Component Installation

You can install toolkit components as standalone via binary installer or package managers. Refer to the Single Component Downloads and Runtime Versions resource to locate a component package.

Intel® FPGA Developmental Flow

For emulation and FPGA optimization report flow, you can just use the Intel oneAPI Base Toolkit that includes the Intel oneAPI DPC++/C++ Compiler. For more information, see FPGA Flow in the *Intel oneAPI Programming Guide*.

For the Intel FPGA hardware or simulation flow, you can target Intel® FPGA devices, Intel® Programmable Acceleration Cards (PACs), or a third-party vendor-provided BSP.

- To target Intel® FPGA devices or custom platforms, see Install Software for Intel FPGA Development Flows.
- To target third party vendor-provided BSP and the FPGA Platform, see Intel® FPGA Add-on for oneAPI Base Toolkit. The vendor specifies which version of Intel Quartus Prime software is necessary to compile the BSP. You must install the Intel® Quartus Prime software and BSP packages separately.

Install with GUI

After downloading the toolkit installation package, follow the steps below to install it with GUI.

1. Launch the installer with the following command:
   - root: `sudo sh ./l_[Toolkit Name]Kit_[version].sh`
   - user: `sh ./l_[Toolkit Name]Kit_[version].sh`
2. Follow the installer instructions.
3. Once the installation is complete, verify that your toolkit is installed to the correct installation directory:
   - root: `/opt/intel/oneapi`
   - user: `~/intel/oneapi`

NOTE If you are using Intel GPU, you need to install the latest GPU drivers separately.

Install with Command Line

Command line installation supports the following installation modes:

- Interactive mode prompts you to select or confirm certain options during the installation process
- Non-interactive (silent) mode allows you to define the installation configuration only once and does not require any user input during installation

The general instructions below are common for both modes.

General Instructions

Launch the installation script using the following command:

- root:

```
sudo sh ./l_[Toolkit Name]Kit_[version].sh [options] -a [arguments]
```

- user:

```
sh ./l_[Toolkit Name]Kit_[version].sh [options] -a [arguments]
```

where `[options]` contain parameters for the package extraction script, and `[arguments]` are options for the installer.

The package extraction script supports the following options:

`-h, --help`	Show help for the package extraction script.
`-f, --extract-folder`	Point to the folder where the package content will be saved.
`-x, --extract-only`	This option unpacks the installation package only. It does not launch the installer.
`-r, --remove-extracted-files <yes|no>`	Remove extracted files after installation. This action cleans up the temporary package file location.
`-l, --log <log file>`	Log all package extraction actions to the specified file.
`-a <arguments>`	Pass arguments to the installer.

The values after `-a` are passed as command line arguments to the installer. The following installer options are supported:

Option	Supported mode	Default value (if option is not passed)	Description
`-c, --cli`	CLI	N/A	Run the installer in interactive text-based user interface (TUI) mode.
`-s, --silent`	Silent	N/A	Run the installer in non-interactive (silent) mode.
`--eula`	Silent	`decline`	Required. Accept or decline End User License Agreement (EULA), supported values: `accept` or `decline` (default).
`--action`	Silent/CLI	`install`	Specify one of the supported values below when the installer action is needed: • `install` (default) Install the product. Use the `--components` option to specify the list of components to be installed. If not specified, the default set of components is installed. • `remove` Uninstall the product.

Option	Supported mode	Default value (if option is not passed)	Description
			• `modify` Change the current set of components installed. List all the components you need using the `--components` option. Components that are already installed still must be in the list if remain relevant. • `downloadonly` Download an offline installation package without installing it. To customize the list of components to be included into a package, use the `--components` option. • `repair` Repair the currently installed product.
`--instance`	Silent/CLI	`default`	Specify an ID of an installation instance. For example: `sh ./l_[Toolkit Name]Kit_[version].sh -a --instance=<instance ID>`. This option enables side-by-side installation of oneAPI products. Each instance is a separate installation entity with its own isolated environment. Product installed in one instance is not visible in another instance. If omitted, installation is performed in default instance. To get the list of available instances, use the `--list-instances` option.
`--list-instances`	Silent/CLI	N/a	Get the list of available installation instances.
`--config`	Silent/CLI	N/A	Point to the configuration INI file with options. You can use this file as an alternative to passing options via the command line; mixed approach is also supported. Sample content of a configuration file: `s=eula=accept`. Use this command to run the installer with the options passed via `config.txt`: `sh ./l_[Toolkit Name]Kit_[version].sh --config config.txt`
`--components`	Silent	`default`	Specify components to perform an action on, supported values: `all`, `default`, custom components split by ':'. If you need the default components and some extra component(s), combine `default` with the name of the extra component(s) separated by ':'. For example: `--components default:<component_name>`.
`--list-products`	N/A	N/A	Get the list of downloaded products, their IDs, versions and statuses (installed/not installed). Use together with the `--instance` option to get the list of available products in a specific instance. For

Option	Supported mode	Default value (if option is not passed)	Description
			example: `sh ./l_[Toolkit Name]Kit_[version].sh -a --list-products --instance=<instance ID>`.
`--product-id`	Silent/CLI	N/A	Specify an ID of a product to perform an action on. Use this option with `--list-components` or `--action {install|remove|modify|repair}`.
`--product-ver`	Silent/CLI	N/A	Specify a product version to perform an action on. Use this option with `--list-components` or `--action {install|remove|modify|repair}`.
`--list-components`	N/A	N/A	Get the list of available components of the current package or of a product specified with `--product-id`. Use together with the `--instance` option to get the list of available components in a specific instance. For example: `sh ./l_[Toolkit Name]Kit_[version].sh -a --list-components --instance=<instance ID>`.
`--package-path`	Silent/CLI	N/A	Specify the directory of the package to install.
`--install-dir`	Silent	default installation directory	Supported in silent mode. Customize the installation directory.
`--log-dir`	Silent/CLI	default log location	Customize the directory to save the log file to.
`--proxy`	Silent/CLI	N/A	Specify proxy settings in the following format: `http://username:password@proxy-server.mycorp.com:3128`.
`--download-cache`	Silent	default download cache location	Point to the directory to store all downloaded and cached files.
`--download-dir`	Silent	default download directory	Customize the download directory, which is used in download-only mode.
`--intel-sw-improvement-program-consent`	Silent	`decline`	Accept or decline participation in Intel Software Improvement Program, supported values: `accept` or `decline` (default). To get the program description, use the `--show-intel-sw-improvement-program-consent` command.
`--show-intel-sw-improvement-program-consent`	N/A	N/A	Show the detailed description of the Intel Software Improvement Program.

Option	Supported mode	Default value (if option is not passed)	Description
`--ignore-errors`	Silent/CLI	N/A	Complete installation even if non-critical errors occur. Check the log file for the list of errors occured and ignored during installation.
`-h, --help`	N/A	N/A	Show the installer help.
`-p, --property`	Silent/CLI	N/A	Pass additional custom options. For example, the string `-p=option1=value -p option2=value` gives two additional options. If a custom option is provided twice with different values, only the latest one will be used. For example, the string `–p=option=a -p option=b` takes `b` as value for `option`.

For example, to show the installer help, use the following command:

```
sh ./l_[Toolkit Name]Kit_[version].sh -a -h
```

Non-interactive (Silent) Installation

1. Use the following command to launch the installer in silent mode:

   - root:

```
sudo sh ./l_[Toolkit Name]Kit_[version].sh -a --silent --eula accept
```

   - user:

```
sh ./l_[Toolkit Name]Kit_[version].sh -a --silent --eula accept
```

   For the full list of supported command line options, refer to the General Instructions section.

2. Once the installation is complete, verify that the toolkit is installed in the default directory:

   - root:

```
/opt/intel/oneapi
```

   - user:

```
~/intel/oneapi
```

NOTE If you are using Intel GPU, you need to install the latest GPU drivers separately.

Interactive Installation

1. Launch the installer with the following command:

   - root:

```
sudo sh ./l_[Toolkit Name]Kit_[version].sh -a --cli
```

   - user:

```
sh ./l_[Toolkit Name]Kit_[version].sh -a --cli
```

For the full list of supported command line options, refer to the General Instructions section.

1. Follow the installer instructions.
2. Once the installation is complete, verify that your toolkit is installed to the correct directory:

   - root:

```
/opt/intel/oneapi
```

   - user:

```
~/intel/oneapi
```

Examples

- Display the list of already installed products and products included in the downloaded package:

  - root:

```
sudo sh ./l_[Toolkit Name]Kit_[version].sh -a --silent --list-products
```

  - user:

```
sh ./l_[Toolkit Name]Kit_[version].sh -a --list-products
```

  Example of output:

```
ID Version Language Installed Name
============================================================================================
intel.oneapi.lin.tbb.product 2021.1.1-129 false Intel® oneAPI Threading Building Blocks
```

- Display the list of components in product of current package:

  - root:

```
sudo sh ./l_[Toolkit Name]Kit_[version].sh -a --list-components
```

  - user:

```
sh ./l_[Toolkit Name]Kit_[version].sh -a --list-components
```

- Display the list of components of any installed product on the system:

  - root:

```
sudo sh ./l_[Toolkit Name]Kit_[version].sh -a --list-components --product-id
intel.oneapi.lin.tbb.product --product-ver 2021.1.1-129
```

  - user:

```
sh ./l_[Toolkit Name]Kit_[version].sh -a --list-components --product-id
intel.oneapi.lin.tbb.product --product-ver 2021.1.1-129
```

  Example of output:

```
ID Version Language Installed Name
============================================================================================
intel.oneapi.lin.tbb.devel 2021.1.1-129 Intel® oneAPI Threading Building Blocks
```

- Install specific Intel oneAPI Toolkit products and components:

  - root:

```
sudo sh ./l_[Toolkit Name]Kit_[version].sh --silent --eula accept --components
intel.oneapi.lin.tbb.devel
```

  - user:

```
sh ./l_[Toolkit Name]Kit_[version].sh -a --silent --eula accept --components
intel.oneapi.lin.tbb.devel
```

Install Using Package Managers

You can install Intel oneAPI packages from one of the following repositories:

- YUM, DNF, Zypper
- APT
- Cloudera (oneMKL)
- Spack
- Conda
- PIP
- NuGet
- Maven (oneDAL)

For instructions on how to get the list of packages available for installation with Linux* package managers, refer to List Toolkits, Components, and Runtime Library Packages.

YUM/DNF/Zypper

You can install Intel oneAPI packages via YUM, DNF, or Zypper package managers (whichever works best for your system configuration).

Pre-installation Steps

NOTE

- If you have an existing installation of Intel® oneAPI Beta, remove it with the following command:

```
# If using YUM or DNF:
sudo -E {yum|dnf} autoremove <package_name>


# If using Zypper:
sudo -E zypper rm <package_name>
```

- When upgrading from 2021.1 to 2021.2, apply automatic removal of conflicting packages during the upgrade process as described in the Upgrade Toolkit/Component section.

1. Check the toolkit-specific System Requirements page to make sure that your OS is supported:

- Intel® oneAPI Base Toolkit
- Intel® oneAPI HPC Toolkit
- Intel® oneAPI IoT Toolkit
- Intel® oneAPI AI Analytics Toolkit
- Intel® oneAPI Rendering Toolkit
- Intel® oneAPI DL Framework Developers Toolkit

You can get your OS version using the following command depending on your Linux distribution:

```
# Redhat, Fedora, CentOS and related
more /etc/redhat-release


# Ubuntu, Debian, others
more /etc/lsb-release
```

2. If you plan to use Intel GPU, install the Intel GPU drivers.

3. Set up the repository:

   If using YUM/DNF:

   a. Create the YUM or DNF repo file in the /temp directory as a normal user:

```
tee > /tmp/oneAPI.repo << EOF
[oneAPI]
name=Intel® oneAPI repository
baseurl=https://yum.repos.intel.com/oneapi
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://yum.repos.intel.com/intel-gpg-keys/GPG-PUB-KEY-INTEL-SW-PRODUCTS.PUB
EOF
```

   b. Move the newly created `oneAPI.repo` file to the YUM/DNF configuration directory `/etc/yum.repos.d`:

```
sudo mv /tmp/oneAPI.repo /etc/yum.repos.d
```

   If using Zypper:

   Add the Intel oneAPI repository public key with the following command:

```
sudo zypper addrepo https://yum.repos.intel.com/oneapi oneAPI
```

   By adding this new repository, Zypper had to automatically import the public repo key. For some cases rpm might require explicit key import by:

```
rpm --import https://yum.repos.intel.com/intel-gpg-keys/GPG-PUB-KEY-INTEL-SW-PRODUCTS.PUB
```

Install Packages

NOTE If you are on a company intranet or behind a firewall, set the `http_proxy` and `https_proxy` environment variables to allow YUM/DNF/Zypper access the repository servers using HTTPS protocol.

1. Get the name of a toolkit package that you need to install from the list of Intel oneAPI packages. Write down or copy your package name for future reference.
2. Install the needed package with the following command:

```
sudo {yum|dnf|zypper} install <package_name>
```

   For example, to install the Intel® oneAPI Base Toolkit package from YUM, use:

```
sudo yum install intel-basekit
```

   If you need to install on a machine with no internet access, or in case of a large distributed installation on a cluster, you can download a package without installing it with the `--download-only` option. For more details, refer to YUM instructions or Zypper man pages.

Installation is complete! For next steps, refer to the Get Started Guide for your toolkit:

- Intel® oneAPI Base Toolkit
- Intel® oneAPI HPC Toolkit
- Intel® oneAPI IoT Toolkit
- Intel® oneAPI AI Analytics Toolkit
- Intel® oneAPI Rendering Toolkit
- Intel® oneAPI DL Framework Developers Toolkit

NOTE If you have applications with long-running GPU compute workloads in native environments, you must disable the hangcheck timeout period to avoid terminating workloads.

Upgrade Toolkit/Component

NOTE Intel oneAPI packages of version 2021.2 (and higher) have conflicts with 2021.1 packages. Before installing 2021.2, you need to remove the 2021.1 packages from your system. If you use YUM v4/DNF/Zypper, you can apply automatic removal of conflicting packages during upgrade with the following commands:

```
# If using YUM:
sudo yum upgrade --allowerasing --best <package_name>

# If using DNF:
sudo dnf upgrade --allowerasing --best <package_name>

# If using Zypper:
sudo zypper update --force-resolution <package_name>
```

For lower versions of YUM, remove the 2021.1 packages manually:

```
sudo yum autoremove <package_name>
```

For more information about the issue and workarounds, refer to the YUM/DNF and ZYPPER Packages oneAPI 2021.1 Gold (Initial Release) issue will Prevent Upgrades article.

To re-install 2021.1 packages, use web and local installers as described in the Installation section of this document.

You can upgrade toolkit or component package to the latest version using the following instructions:

- Toolkit: `{yum|dnf|zypper} upgrade <toolkit package>`

  For example, to upgrade the Intel oneAPI Base Toolkit package to the latest version, use the following command:

```
# If using YUM:
sudo yum upgrade intel-basekit


# If using DNF:
sudo dnf upgrade intel-basekit


# If using Zypper:
sudo zypper upgrade intel-basekit
```

- Component: `{yum|dnf|zypper} upgrade <component package>`

  For example, to upgrade the Intel Distribution for GDB* package, use the following command:

```
# If using YUM:
sudo yum upgrade intel-oneapi-dpcpp-debugger


# If using DNF:
sudo dnf upgrade intel-oneapi-dpcpp-debugger
```

```
# If using Zypper:
sudo zypper upgrade intel-oneapi-dpcpp-debugger
```

List of Intel® oneAPI Packages

Toolkit Packages

The following toolkits and associated versions are available for installation via YUM repositories:

NOTE The repositories always contain the latest released version.

Toolkit Name	64-bit Meta Package Name (default)	32-bit Meta Package Name*
Intel® oneAPI Base Toolkit	`intel-basekit`	`intel-basekit-32bit`
Intel® oneAPI HPC Toolkit	`intel-hpckit`	`intel-hpckit-32bit`
Intel® oneAPI IoT Toolkit	`intel-iotkit`	`intel-iotkit-32bit`
Intel® oneAPI DL Framework Developer Toolkit	`intel-dlfdkit`	`intel-dlfdkit-32bit`
Intel® AI Analytics Toolkit	`intel-aikit`	`intel-aikit-32bit`
Intel® oneAPI Rendering Toolkit	`intel-renderkit`	`intel-renderkit-32bit`

* - only required if you deploy and deploy 32-bit applications

Intel® System Bring-Up Toolkit is not distributed via a repository, see details.

Intel® Distribution of OpenVINO™ toolkit for Linux* is distributed via separate YUM and APT repositories.

Runtime Library Packages

The oneAPI repository provides runtime library packages. Install these packages on systems where you run oneAPI applications but do not do development, compilation, or runtime profiling. The following runtime library packages are available:

- oneAPI runtime libraries package, which is a superset of all runtimes for oneAPI components:
  - 64-bit: `intel-oneapi-runtime-libs`
  - 32-bit: `intel-oneapi-runtime-libs-32bit`
- Component runtime library packages. For instructions on how to get the list of all available standalone runtime packages, refer to the List Standalone Runtime Library Packages section.

APT

Pre-installation Steps

NOTE If you have an existing installation of Intel® oneAPI Beta, remove it with the following command:

```
sudo apt autoremove <package_name>
```

1. Check the toolkit-specific System Requirements page to make sure that your OS is supported:
   - Intel® oneAPI Base Toolkit
   - Intel® oneAPI HPC Toolkit
   - Intel® oneAPI IoT Toolkit
   - Intel® oneAPI AI Analytics Toolkit
   - Intel® oneAPI Rendering Toolkit
   - Intel® oneAPI DL Framework Developers Toolkit

   You can get your OS version using the following command depending on your Linux distribution:

```
# Redhat, Fedora, CentOS and related
more /etc/redhat-release


# Ubuntu, Debian, others
more /etc/lsb-release
```

2. If you plan to use Intel GPU, install the Intel GPU drivers.
3. Set up the repository:

```
# download the key to system keyring
wget -O- https://apt.repos.intel.com/intel-gpg-keys/GPG-PUB-KEY-INTEL-SW-PRODUCTS.PUB \
| gpg --dearmor | sudo tee /usr/share/keyrings/oneapi-archive-keyring.gpg > /dev/null

# add signed entry to apt sources and configure the APT client to use Intel repository:
echo "deb [signed-by=/usr/share/keyrings/oneapi-archive-keyring.gpg] https://apt.repos.intel.com/
oneapi all main" | sudo tee /etc/apt/sources.list.d/oneAPI.list
```

4. Update packages list and repository index:

```
sudo apt update
```

Install Packages

NOTE If you are on a company intranet or behind a firewall, set the `http_proxy` and `https_proxy` environment variables to allow APT access the repository servers using HTTPS protocol.

1. Get the name of a toolkit package that you need to install from the list of Intel oneAPI packages. Write down or copy your package name for future reference.
2. Install the needed package with the following command:

```
sudo apt install <package_name>
```

   For example, to install the Intel® oneAPI Base Toolkit package, use:

```
sudo apt install intel-basekit
#repeat 'apt install ...' for each toolkit you need
```

   If you need to install on a machine with no internet access, or in case of a large distributed installation on a cluster, you can download a package without installing it with the `--download-only` option.

NOTE If you want to integrate tools into the Eclipse* IDE, open Eclipse and verify that a menu titled **Intel** is present. If the menu is not present, see Installing Eclipse* Plugins from the IDE.

Installation is complete! For next steps, refer to the Get Started Guide for your toolkit:

- Intel® oneAPI Base Toolkit

- Intel® oneAPI HPC Toolkit
- Intel® oneAPI IoT Toolkit
- Intel® oneAPI AI Analytics Toolkit
- Intel® oneAPI Rendering Toolkit
- Intel® oneAPI DL Framework Developers Toolkit

NOTE If you have applications with long-running GPU compute workloads in native environments, you must disable the hangcheck timeout period to avoid terminating workloads.

List of Intel® oneAPI Packages

Toolkit Packages

The following toolkits and associated versions are available for installation via APT repositories:

NOTE The repositories always contain the latest released version.

Toolkit Name	64-bit Meta Package Name (default)	32-bit Meta Package Name*
Intel® oneAPI Base Toolkit	`intel-basekit`	`intel-basekit-32bit`
Intel® oneAPI HPC Toolkit	`intel-hpckit`	`intel-hpckit-32bit`
Intel® oneAPI IoT Toolkit	`intel-iotkit`	`intel-iotkit-32bit`
Intel® oneAPI DL Framework Developer Toolkit	`intel-dlfdkit`	`intel-dlfdkit-32bit`
Intel® AI Analytics Toolkit	`intel-aikit`	`intel-aikit-32bit`
Intel® oneAPI Rendering Toolkit	`intel-renderkit`	`intel-renderkit-32bit`

* - only required if you deploy and deploy 32-bit applications

Intel® System Bring-Up Toolkit is not distributed via a repository, see details.

Intel® Distribution of OpenVINO™ toolkit for Linux* is distributed via separate YUM and APT repositories.

Runtime Library Packages

The oneAPI repository provides runtime library packages. Install these packages on systems where you run oneAPI applications but do not do development, compilation, or runtime profiling. The following runtime library packages are available:

- oneAPI runtime libraries package, which is a superset of all runtimes for oneAPI components:
  - 64-bit: `intel-oneapi-runtime-libs`
  - 32-bit: `intel-oneapi-runtime-libs-32bit`
- Component runtime library packages. For instructions on how to get the list of all available standalone runtime packages, refer to the List Standalone Runtime Library Packages section.

Conda

This page provides general instructions on installing the Intel® oneAPI component packages via the Conda* package manager.

For additional installation notes, refer to the Conda documentation.

To install a package, execute the following command:

- To install the latest version available:

```
conda install -c intel <package_name>
```

  To get your package name, refer to the list of packages in the table below.
- To install a specific version:

```
conda install -c intel <package_name>==<version>
```

  For example: `conda install -c intel mkl==2021.1.1`

List of Available Packages

Component Name	Package Name	Platform	Dependencies
Intel® MPI Library	`impi_rt` `impi-devel`	`linux-x64`	N/A
Intel® Fortran Compiler (Beta) and Intel® Fortran Compiler Classic	`intel-fortran-rt`	`linux-x64` `linux-x86`	Intel® MPI Library Intel OpenMP* Runtime Library
Intel® CPU Runtime for OpenCL™ Applications	`intel-opencl-rt`	`linux-x64`	oneTBB
Intel® oneAPI DPC++/C++ Compiler	`dpcpp-cpp-rt`	`linux-x64`	Intel® CPU Runtime for OpenCL™ Applications Intel OpenMP* Runtime Library
Intel® oneAPI DPC++ Library	`onedpl-devel`	`linux-x64`	N/A
Intel OpenMP* Runtime Library	`intel-openmp`	`linux-x64` `linux-x86`	N/A
Intel® oneAPI Threading Building Blocks (oneTBB)	`tbb` `tbb-devel`	`linux-x64` `linux-x86`	N/A
	`tbb4py`	`linux-x64`	N/A
Intel® oneAPI Data Analytics Library (oneDAL)	`dal` `dal-static`	`linux-x64`	oneTBB

Component Name	Package Name	Platfo rm	Dependencies
	`dal-devel` `dal-include`	`linux -x86`	
	`daal4py`	`linux -x64`	oneDAL
Intel® Extension for Scikit-learn	`scikit-learn-intelex`	`linux -x64`	oneDAL

Intel® Integrated Performance Primitives (Intel® IPP)	`ipp`
	`ipp-static`
	`ipp-include`
		`linux -x64`
		`linux -x86`
			N/A

Intel® Integrated Performance Primitives Cryptography	`ipp_crypto`
	`ipp_crypto-static`
	`ipp_crypto-include`
		`linux -x64`	N/A

Intel® oneAPI Math Kernel Library (oneMKL)	`mkl`
	`mkl-devel`
	`mkl-static`
	`mkl-include`
		`linux -x64`
		`linux -x86`
			Intel OpenMP* Runtime Library
			oneTBB

Intel® oneAPI Deep Neural Network Library (oneDNN)*	`onednn-cpu-gomp` `onednn-devel-cpu-gomp`	`linux -x64`	N/A
	`onednn-cpu-iomp` `onednn-devel-cpu-iomp`	`linux -x64`	Intel OpenMP* Runtime Library
	`onednn-cpu-dpcpp-gpu-dpcpp` `onednn-devel-cpu-dpcpp-gpu-dpcpp`	`linux -x64`	Intel® CPU Runtime for OpenCL™ Applications Intel oneAPI DPC++/C++ Compiler Runtime
Intel® oneAPI Collective Communications Library (oneCCL)	`oneccl-devel`	`linux -x64`	N/A

* - For Intel® oneAPI Deep Neural Network Library (oneDNN), only packages of identical configuration can be installed into one environment. For example, you can install `onednn-devel-cpu-vcomp` with `onednn-cpu-vcomp`, but should avoid installing it with packages of other configurations, like `cpu-iomp`, `cpu-tbb`, `cpu-dpcpp-gpu-dpcpp`.

- Install Intel® AI Analytics Toolkit via Conda*
  - List of Available Packages

Install Intel® AI Analytics Toolkit via Conda*

Intel provides access to the AI Kit through a public Anaconda repository. If you do not have an existing Conda-based python environment, install Conda and Miniconda*. To get more details on the AI Analytics Toolkit, visit the Intel AI Analytics toolkit home page.

The AI Kit contains three distinct Python environments targeting different use cases:

- `intel-aikit-tensorflow` for deep learning workflows using Intel® Optimization for TensorFlow*.
- `intel-aikit-pytorch` for deep learning workflows using Intel® Optimization for PyTorch*.
- `intel-aikit-modin` for data analytics and machine learning workflows using Intel® Distribution of Modin (for accelerated Panda data frames), Intel® Extension for Scikit-learn* and Intel optimizations for XGboost (for ML training and inference).
- `intel-aikit` for data science workstation development. The oneAPI AI kit samples and documents are not applicable to the `intel-aikit` Conda package.

NOTE To get the latest version of the Intel(R) Optimization for TensorFlow*, you must first install Python 3.9, then install the AI Kit through Anaconda.

To instal the AI Kit via Conda, complete the following steps:

1. Activate your existing python conda environment located in `<pythonhome>`:

```
source <pythonhome>/bin/activate
```

2. Install the AI Kit oneAPI packages in a new environment using `conda create`. A list of available packages is located at https://anaconda.org/intel/repo. Not all packages in the Anaconda repository are up to date with the current release. If the repo contains an outdated version of a required component, get a newer one by installing via the command line or GUI.

   If the repository contains the desired version, create an AI Kit Tensorflow* environment named `aikit-tf` with this version:

```
conda create -n aikit-tf -c intel intel-aikit-tensorflow
```

   Similarly, you can create an AI Kit PyTorch environment named `aikit-pt`:

```
conda create -n aikit-pt -c intel intel-aikit-pytorch
```

   You can also create an AI Kit Modin and machine learning environment named `aikit-modin`:

```
conda create -n aikit-modin -c intel intel-aikit-modin
```

3. Set user environment. After the toolkit is installed, before accessing the tools, you must activate your python environment and set up environment variables to access the tools. For example, to activate the python environment created in the previous step, use:

```
conda activate aikit-tf
```

NOTE

- To install the Model Zoo for Intel® Architecture component of the toolkit, clone the main branch to your local directory: `git clone https://github.com/IntelAI/models.git`.
- If you have applications with long-running GPU compute workloads in native environments, you must disable the hangcheck timeout period to avoid terminating workloads.

List of Available Packages

NOTE Intel® packages are available on intel label on the Anaconda* Cloud. You must include `-c intel` on your command line as in the examples above, or add intel to your Conda configuration file using `conda config --add channels intel`.

Component Name	Package Name	Platform
Intel® Distribution for Python*	`intelpython3_full`	`linux-x64`
Intel® Distribution of Modin* (via Anaconda distribution of the toolkit using the Conda package manager)	`intel-aikit-modin`	`linux-x64`
Intel® Neural Compressor	`neural-compressor`	`linux-x64`
Intel® Optimization for PyTorch*	`intel-aikit-pytorch`	`linux-x64`
Intel® Optimization for TensorFlow*	`intel-aikit-tensorflow`	`linux-x64`

After you have installed your components, view the Get Started Guide for the Intel oneAPI AI Analytics Toolkit to build and run a sample or explore Getting Started Samples on GitHub.

PIP

This page provides general instructions on installing the Intel® oneAPI component packages from the Python* Package Index (PyPI).

For additional installation notes, refer to the PyPI documentation.

To install a package, execute the following command:

- To install the latest version available:

```
pip install <package_name>
```

  To get your package name, refer to the list of packages in the table below.
- To install a specific version:

```
pip install -c intel <package_name>==<version>
```

  For example: `pip install mkl==2021.1.1`

Important For Intel® oneAPI Deep Neural Network Library (oneDNN), only packages of identical configuration can be installed into one environment. For example, you can install onednn-devel-cpu-vcomp with onednn-cpu-vcomp, but should avoid installing it with packages of other configurations, like cpu-iomp, cpu-tbb, cpu-dpcpp-gpu-dpcpp.

List of Available Packages

Component Name	Package Name	Platform	Dependencies
Intel® MPI Library	`impi_rt` `impi-devel`	`linux-x64`	N/A

Component Name	Package Name	Platform	Dependencies
Intel® Fortran Compiler (Beta) and Intel® Fortran Compiler Classic	`intel-fortran-rt`	`linux-x64` `linux-x86`	Intel® MPI Library Intel OpenMP* Runtime Library
Intel® CPU Runtime for OpenCL™ Applications	`intel-opencl-rt`	`linux-x64`	oneTBB
Intel® oneAPI DPC++/C++ Compiler	`dpcpp-cpp-rt`	`linux-x64`	Intel® CPU Runtime for OpenCL™ Applications Intel OpenMP* Runtime Library
Intel OpenMP* Runtime Library	`intel-openmp`	`linux-x64` `linux-x86`	N/A
Intel® oneAPI Threading Building Blocks (oneTBB)	`tbb` `tbb-devel`	`linux-x64` `linux-x86`	N/A
	`tbb4py`	`linux-x64`	N/A

Intel® oneAPI Data Analytics Library (oneDAL)	`daal`
	`daal-static`
	`daal-devel`
	`daal-include`
		`linux-x64`
		`linux-x86`
			oneTBB

	`daal4py`	`linux-x64`	oneDAL
Intel® Extension for Scikit-learn	`scikit-learn-intelex`	`linux-x64`	oneDAL

Intel® Integrated Performance Primitives (Intel® IPP)	`ipp`
	`ipp-static`
	`ipp-include`
		`linux-x64`
		`linux-x86`
			N/A

Intel® Integrated Performance Primitives Cryptography	`ipp_crypto`
	`ipp_crypto-static`
	`ipp_crypto-include`
		`linux-x64`
		`linux-x86`
			N/A

Component Name	Package Name	Platform	Dependencies

Intel® oneAPI Math Kernel Library (oneMKL)	`mkl`
	`mkl-devel`
	`mkl-static`
	`mkl-include`
		`linux-x64`
		`linux-x86`
			Intel OpenMP* Runtime Library
			oneTBB

Intel® oneAPI Deep Neural Network Library (oneDNN)	`onednn-cpu-gomp` `onednn-devel-cpu-gomp`	`linux-x64`	N/A
	`onednn-cpu-iomp` `onednn-devel-cpu-iomp`	`linux-x64`	Intel OpenMP* Runtime Library
	`onednn-cpu-dpcpp-gpu-dpcpp` `onednn-devel-cpu-dpcpp-gpu-dpcpp`	`linux-x64`	Intel® CPU Runtime for OpenCL™ Applications Intel oneAPI DPC++/C++ Compiler Runtime
Intel® oneAPI Collective Communications Library (oneCCL)	`oneccl-devel`	`linux-x64`	N/A
Intel® Distribution of Modin* (via Anaconda distribution of the toolkit using the Conda package manager)	`intel-aikit-modin`	`linux-x64`	N/A
Intel® Neural Compressor	`neural-compressor`	`linux-x64`	N/A
Intel® Optimization for TensorFlow*	`intel-tensorflow` `intel-tensorflow-avx512`	`linux-x64`	N/A

NuGet

This page provides general notes on how to install Intel® oneAPI components distributed via the NuGet channel. NuGet is a Microsoft-supported mechanism for sharing compiled code. It also defines how the packages are created, hosted and consumed, and it provides the tools for each of those roles. For more details on the installation process, please refer to the Microsoft* documentation.

Intel® oneAPI components distributed via NuGet include both development and runtime options.

For your convenience, the components are divided to *devel* and *static* packages corresponding to the different linking types (dynamic and static). Certain component packages are also split into x64 and x86 versions to reduce the overall package size.

Development Packages

The following table provides the full list of available packages:

Component Name	Package Name	Platform	Dependencies
Intel® MPI Library	intelmpi.devel.<platform>	linux-x64	N/A
Intel OpenMP* Runtime Library	intelopenmp.devel.<platform> intelopenmp.static.<platform>	linux	N/A
Intel® oneAPI Data Analytics Library (oneDAL)	inteldal.devel.<platform> inteldal.static.<platform>	linux-x64	N/A
Intel® oneAPI Threading Building Blocks (oneTBB)	inteltbb.devel.<platform>	linux	N/A
Intel® Integrated Performance Primitives	intelipp.devel.<platform> intelipp.static.<platform> intelipp.nonpic.<platform>	linux-x64 linux-x86	N/A
Intel® Integrated Performance Primitives Cryptography	intelipp_crypto.devel.<platform> intelipp_crypto.static.<platform> intelipp_crypto.nonpic.<platform>	linux-x64 linux-x86	N/A
Intel® oneAPI Math Kernel Library	intelmkl.devel.<platform> intelmkl.static.<platform>	linux-x64 linux-x86	N/A
Intel® oneAPI Math Kernel Library (Cluster Components)	intelmkl.devel.cluster.<platform> intelmkl.static.cluster.<platform>	linux-x64	Intel OpenMP* Runtime Library oneMKL

All the specified dependencies will be downloaded automatically by the NuGet Package Manager.

Runtime Packages

The runtime packages are runtime redistributable libraries that will automatically load optimizations specific to your Intel hardware (including, but not limited to, vectorization). They can be used by another NuGet package that depends on these runtimes.

Component Name	Package Name	Platform Availability
Intel® oneAPI Video Processing Library (oneVPL)	`onevpl.runtime.<platform>`	`linux-x64`

Cloudera

This page provides general instructions on installing the Intel® oneAPI component parcel packages using the Cloudera* Manager. For additional notes, refer to the Parcels documentation and Cloudera Installation Guide.

Currently, only the Intel® oneAPI Math Kernel Library (oneMKL) component is distributed via Cloudera*. Package name: `mkl-<version>-el7.parcel`, for example, `mkl-2021.2.0.296-el7.parcel`.

The install the oneMKL parcel:

1. In the **Cloudera Manager Admin Console**, access the **Parcels** page by doing one of the following:
   - Click the **Parcels** indicator in the left navigation bar.
   - Click the **Hosts** in the left navigation bar, then click the **Parcels** tab.
2. At the **Parcels** page, click the **Parcel Repositories & Network Settings** button.
3. In the **Remote Parcel Repository URLs** list, click the plus symbol to open an additional row. Enter the path to Intel® MKL Parcel repository: `http://parcels.repos.intel.com/mkl/latest`. Click the **Save & Verify configuration** button.
4. Click the **Check for New Parcels** button. In the **Location** selector, click **Available Remotely**. The latest oneMKL parcel should be available for download.
5. Click the **Download** button for the oneMKL parcel. By downloading the package, you agree with the terms and conditions stated in the End-User License Agreement (EULA).
6. When download is completed, click the **Distribute** button to distribute the parcel on all cluster nodes.
7. When distribution is completed, click the **Activate** button to activate the parcel on all cluster nodes.

Note

The repository URL referenced above installs the latest version of oneMKL parcel. To install a lower version, use the URL based on the following model: `http://parcels.repos.intel.com/mkl/<version>.<update>.<build_number>`.

Maven

This page provides general instructions on how to include Intel® oneAPI Data Analytics Library (oneDAL) packages from the Maven repository into your Java project.

To enable oneDAL in your project, specify the following artifacts in your build automation tool:

```
Group ID: com.intel.dal
Version: <version>
Artifact ID: dal
```

where `<version>` is a valid component version, for example, 2021.2.0.123.

For more information on the Maven dependency mechanism, refer to the Maven documentation.

Spack

This page provides general instructions on installing the Intel® oneAPI component packages via Spack. After installation, you can use the tools directly or use Spack to build packages with the tools.

To install a package, execute the following command:

```
spack install <package_name>
```

Example

The example below demonstrates how to set up Intel oneAPI compilers with Spack. For the full list of oneAPI packages available via Spack, refer to the Package List.

1. Install the compilers package with the following command:

```
spack install intel-oneapi-compilers
```

2. Add the oneAPI compilers to the set of compilers that Spack can use:

```
spack compiler add `spack location -i intel-oneapi-compilers`/compiler/latest/linux/bin/intel64
spack compiler add `spack location -i intel-oneapi-compilers`/compiler/latest/linux/bin
```

   This adds the compilers to your `compilers.yaml`.

3. Verify that the compilers are available:

```
spack compiler add
```

The `intel-oneapi-compilers` package includes two families of compilers:

- Intel: `icc`, `iccpc`, `ifort` - Intel's classic compilers
- oneAPI: `icx`, `icpx`, `ifx` - Intel's new generation of compilers based on LLVM

To build the `patchelf` Spack package with `icc`, use:

```
spack install patchelf%intel
```

To build with `icx`, use:

```
spack install patchelf%oneapi
```

In addition to compilers, oneAPI contains many libraries. The `hdf5` package works with any compatible MPI implementation. To build `hdf5` with Intel oneAPI MPI, use:

```
spack install hdf5 +mpi
^intel-oneapi-mpi
```

For more information, see Spack documentation.

List Available Toolkits, Components, and Runtime Library Packages

Use the commands provided below to find and install specific toolkits, standalone components, standalone runtime library packages, or simply to see all available packages in a corresponding oneAPI repository:

- List toolkit packages
- List standalone components
- List standalone runtime library packages
- List all packages

List Toolkit Packages

To query the repository for available toolkit packages, use the following command:

YUM/DNF:

```
sudo -E {yum|dnf} --disablerepo="*" --enablerepo="oneAPI" list available | grep kit | grep -v
runtime
```

Zypper:

```
sudo -E zypper pa -ir oneAPI | grep kit | grep -v runtime
```

APT:

```
sudo -E apt-cache pkgnames intel | grep kit | grep -v runtime
```

List Standalone Components

The oneAPI repository also contains standalone components, which are packages that provide a specific tool for cases where you do not need an entire toolkit. For these packages, if there is a <component>-runtime package, make sure to get and install both the component package and its runtime package. Not all standalone components need an additional runtime package. If you do not see a runtime package for your standalone component, then you do not need one.

To query the repository for available standalone components and their runtime packages, use the following command:

YUM/DNF:

```
sudo -E {yum|dnf} --disablerepo="*" --enablerepo="oneAPI" list available | grep intel-oneapi |
grep -v intel-oneapi-runtime
```

Zypper:

```
sudo -E zypper pa -ir oneAPI | grep intel-oneapi | grep -v intel-oneapi-runtime
```

APT:

```
sudo -E apt-cache pkgnames intel | grep intel-oneapi | grep -v intel-oneapi-runtime
```

List Standalone Runtime Library Packages

The oneAPI repository provides standalone runtime library packages. Install these packages on systems where you run oneAPI applications but do not do development, compilation, or runtime profiling. In this case, you only need the shared libraries dynamically linked to by executables, provided by these packages.

To query the repository for available component runtime libraries, use the following command:

YUM/DNF:

```
sudo -E {yum|dnf} --disablerepo="*" --enablerepo="oneAPI" list available | grep intel-oneapi-
runtime
```

Zypper:

```
sudo -E zypper pa -ir oneAPI | grep intel-oneapi-runtime
```

APT:

```
sudo -E apt-cache pkgnames intel | grep intel-oneapi-runtime
```

List All Packages

To query all available Intel® oneAPI packages provided in a repository, use the following command:

YUM/DNF:

```
sudo -E {yum|dnf} --disablerepo="*" --enablerepo="oneAPI" list available
```

Zypper:

```
sudo -E zypper pa -ir oneAPI
```

APT:

```
sudo -E apt-cache pkgnames intel
```

Install Packages or Components to Different Directories

Intel oneAPI installer supports side-by-side installation. It means that you can install multiple instances of toolkits/components to different directories on the same machine. Each instance is a separate installation entity with its own isolated environment. Product installed in one instance is not visible in another instance.

With multi-instance installation, you can:

- Install a newer version of a toolkit/component without removing the previous one
- Install toolkits to different directories other than default
- Have multiple instances of the same version of a toolkit/component installed

To install a package into a specific instance, use the following command:

- root:

```
sudo sh ./l_[Toolkit Name]Kit_[version].sh -a -s -eula=accept --install-dir=<custom-install-dir>
--instance=<instance ID>
```

- user:

```
./l_[Toolkit Name]Kit_[version].sh -a -s -eula=accept --install-dir=<custom-install-dir> --
instance=<instance ID>
```

where

- `<custom-install-dir>` is the directory where you want to install a specific instance to
- `<instance ID>` is a unique combination of alphanumeric symbols that designate an instance, for example `my-custom-instance-1`

For instructions on how to uninstall product(s) from a specific instance, refer to Uninstall Using Silent CLI.

Configure WSL 2 for GPU Workflows

With Microsoft* Windows Subsystem for Linux 2 (WSL 2), you can use native Linux distribution of Intel® oneAPI tools and libraries on Windows*.

To be able to use Intel oneAPI tools on WSL 2 for GPU workflows, install the Intel GPU drivers as described below.

Ubuntu* 20.04 (focal)

Step 1: Add package repository

Install the `repositories.intel.com/graphics` package repository by executing the following command in your WSL 2 console:

```
sudo apt-get install -y gpg-agent wget
wget -qO - https://repositories.intel.com/graphics/intel-graphics.key |
  sudo gpg --dearmor --output /usr/share/keyrings/intel-graphics.gpg
echo 'deb [arch=amd64 signed-by=/usr/share/keyrings/intel-graphics.gpg] https://
repositories.intel.com/graphics/ubuntu focal-devel main' | \
  sudo tee  /etc/apt/sources.list.d/intel.gpu.focal.list
```

Tip Before pasting the command to your console, run `sudo ls` and enter your password to prevent the commands from being swallowed by the sudo password prompt.

The code above performs the following:

- Checks that your system has `gpg-agent` and `wget` installed
- Downloads and installs the public key used to verify the integrity of the package repository
- Adds the `repositories.intel.com/graphics` repository to the system

Step 2: Install runtime and development (optional) packages

Install GPU software packages with the following command:

```
sudo apt-get install \
  intel-opencl-icd \
  intel-level-zero-gpu level-zero \
  intel-media-va-driver-non-free libmfx1 libmfxgen1 libvpl2 \
  libegl-mesa0 libegl1-mesa libegl1-mesa-dev libgbm1 libgl1-mesa-dev libgl1-mesa-dri \
  libglapi-mesa libgles2-mesa-dev libglx-mesa0 libigdgmm11 libxatracker2 mesa-va-drivers \
  mesa-vdpau-drivers mesa-vulkan-drivers va-driver-all
```

OPTIONAL: If you plan to perform development tasks, you need to install the following optional development packages to make sure that oneAPI tools function correctly:

```
sudo apt-get install -y \
  libigc-dev \
  intel-igc-cm \
  libigdfcl-dev \
  libigfxcmrt-dev \
  level-zero-dev
```

Reboot the system for these changes to take effect.

Step 3: Configure permissions

To access GPU capabilities, the user needs to have correct permissions on system. Use the following command to list the group assigned ownership of the render nodes and list the groups the active user is a member of:

```
stat -c "%G" /dev/dri/render*
groups ${USER}
```

If a group is listed for the render node that is not listed for the user, add the user to the group using `gpasswd`:

```
sudo gpasswd -a ${USER} render
newgrp render
```

Step 4: Verify installation

Verify Computing drivers installation:

```
sudo apt-get install clinfo
clinfo
```

The command should return similar to the following:

```
Number of platforms                               2
Platform Name                                   Intel(R) OpenCL HD Graphics
Platform Vendor                                 Intel(R) Corporation
Platform Version                                OpenCL 3.0
Platform Profile                                FULL_PROFILE
```

Ubuntu* 22.04 (jammy)

Step 1: Add package repository

Install the `repositories.intel.com/graphics` package repository by executing the following command in your WSL 2 console:

```
sudo apt-get install -y gpg-agent wget
wget -qO - https://repositories.intel.com/graphics/intel-graphics.key |
  sudo gpg --dearmor --output /usr/share/keyrings/intel-graphics.gpg
echo 'deb [arch=amd64,i386 signed-by=/usr/share/keyrings/intel-graphics.gpg] https://
repositories.intel.com/graphics/ubuntu jammy arc' | \
  sudo tee  /etc/apt/sources.list.d/intel.gpu.jammy.list
```

Tip Before pasting the command to your console, run `sudo ls` and enter your password to prevent the commands from being swallowed by the sudo password prompt.

The code above performs the following:

- Checks that your system has `gpg-agent` and `wget` installed
- Downloads and installs the public key used to verify the integrity of the package repository
- Adds the `repositories.intel.com/graphics` repository to the system

Step 2: Install runtime and development (optional) packages

Install GPU software packages with the following command:

```
sudo apt-get install -y \
  intel-opencl-icd intel-level-zero-gpu level-zero \
  intel-media-va-driver-non-free libmfx1 libmfxgen1 libvpl2 \
  libegl-mesa0 libegl1-mesa libegl1-mesa-dev libgbm1 libgl1-mesa-dev libgl1-mesa-dri \
  libglapi-mesa libgles2-mesa-dev libglx-mesa0 libigdgmm12 libxatracker2 mesa-va-drivers \
  mesa-vdpau-drivers mesa-vulkan-drivers va-driver-all
```

OPTIONAL: If you plan to perform development tasks, you need to install the following optional development packages to make sure that oneAPI tools function correctly:

```
sudo apt-get install -y \
  libigc-dev \
  intel-igc-cm \
  libigdfcl-dev \
  libigfxcmrt-dev \
  level-zero-dev
```

To support Steam games, install i386 packages:

```
sudo dpkg --add-architecture i386
sudo apt-get update

sudo apt-get install  -y \
    udev mesa-va-drivers:i386 mesa-common-dev:i386 mesa-vulkan-drivers:i386 \
    libd3dadapter9-mesa-dev:i386 libegl1-mesa:i386  libegl1-mesa-dev:i386   \
    libgbm-dev:i386 libgl1-mesa-glx:i386 libgl1-mesa-dev:i386   \
    libgles2-mesa:i386 libgles2-mesa-dev:i386 libosmesa6:i386   \
    libosmesa6-dev:i386 libwayland-egl1-mesa:i386  libxatracker2:i386 \
    libxatracker-dev:i386 mesa-vdpau-drivers:i386  libva-x11-2:i386
```

Reboot the system for these changes to take effect.

Step 3: Configure permissions

To access GPU capabilities, the user needs to have correct permissions on system. Use the following command to list the group assigned ownership of the render nodes and list the groups the active user is a member of:

```
stat -c "%G" /dev/dri/render*
groups ${USER}
```

If a group is listed for the render node that is not listed for the user, add the user to the group using `gpasswd`:

```
sudo gpasswd -a ${USER} render
newgrp render
```

Step 4: Verify installation

Verify Computing drivers installation:

```
sudo apt-get install clinfo
clinfo
```

The command should return similar to the following:

```
Number of platforms                               2
Platform Name                                   Intel(R) OpenCL HD Graphics
Platform Vendor                                 Intel(R) Corporation
Platform Version                                OpenCL 3.0
Platform Profile                                FULL_PROFILE
```

Install Software for Intel FPGA Development Flows

Field-programmable gate arrays (FPGAs) are configurable integrated circuits that you can program to implement arbitrary circuit topologies. Classified as spatial compute architectures, FPGAs differ significantly from fixed Instruction Set Architecture (ISA) devices such as CPUs and GPUs. FPGAs offer a different set of optimization trade-offs from these traditional accelerator devices. While you can compile SYCL* code for CPU, GPU or FPGA, the compiling process for FPGA development is somewhat different than that for CPU or GPU development.

SYCL supports accelerators in general. The Intel® oneAPI DPC++/C++ Compiler implements additional FPGA-specific support to assist FPGA code development. For additional information about the FPGA flows, refer to Types of SYCL* FPGA Compilation topic in the *Intel oneAPI Programming Guide*.

How to Work With FPGA?

The following sections describe various methods you can work with FPGA:

Use Preinstalled Environment

If you are new to oneAPI and FPGA development, then Intel recommends using the Intel® DevCloud. Intel® DevCloud provides a preinstalled development environment with free access to Intel® oneAPI toolkits and components, latest Intel® hardware, optimized frameworks, tools, and libraries to speed up your learning and project prototyping.

Intel® DevCloud is already set up with an Intel® Programmable Acceleration Card (PAC) with Intel Arria® 10 GX FPGA and Intel® FPGA PAC D5005 (previously known as Intel® PAC for Intel® Stratix® 10 SX FPGA) and the necessary software stack. For more information, refer to Get Started with Intel® oneAPI Base Toolkit on the DevCloud.

Set Up Your Own System and Install Software

If you want to set up your own system, then use one of these methods:

- **Set up a single system**: In this method, you can use a single system acting as both the runtime and development system. Install the Intel oneAPI Base Toolkit, Intel® Quartus® Prime Software, and custom platform/FPGA device (hardware-run, machine-specific) on the same system.
- **Set up separate development and runtime systems**: In this method, you install the custom platform/FPGA device on the runtime system and run only the design. On the development system, install the Intel® Quartus® Prime software to compile and generate the FPGA bitstream. Refer to Intel® Quartus® Prime Software, Install Intel® FPGA Board Packages sections for more information.
- **Set up a cloud on-premise**: A cloud on-premise helps reduce the hardware cost necessary for development. In this workflow, you can set up two development systems, one for the FPGA development and the other for the Intel® Quartus® Prime software compilation. The runtime system can be different. After setting up your development systems, install the physical card on the system. Refer to the Intel® oneAPI DPC++ /C++ Library System Requirements for FPGA requirements.
  - On the first development system with lower configurations (8 GB RAM), iterate over your designs using the emulation and report flow to verify code correctness. You just need to install the Intel® oneAPI Base Toolkit package on this system. For more information about emulation and report flow, refer to the Types of FPGA Compilation.
  - On the second development system with higher configurations (48 or 64 GB RAM based on the Intel PAC you use), install the Intel® Quartus® Prime Software. Perform Intel® Quartus® Prime compilation using either the hardware flow or the device link flow. For more information, refer to the FPGA Flow in the *Intel® oneAPI Programming Guide*.

NOTE

- FPGA IP Authoring flow is now supported. Developing IP components/hardware with oneAPI requires the Intel oneAPI Base Toolkit and Intel® Quartus® Prime Software. For details about getting started with the IP component development flow, refer to Getting Started with Intel® oneAPI Toolkits and Intel® Quartus® Prime Software and FPGA Flow in the *Intel® oneAPI Programming Guide*.
- Intel® Quartus® Prime Software is required only for simulation and hardware generation flows, and integrating your IP component into your design. You can generate reports and RTL code, and run the emulation stage with only the Intel® oneAPI Base Toolkit.

For additional details, refer to the following topics:

- Install the Intel® Quartus Prime Software
- Install Intel® FPGA Board Packages
- Install Intel® PAC BSPs
  - Prerequisite
  - Installation Procedure
    - For Systems Not Connected to a PAC Board
    - For Systems Connected to a PAC Board
  - Update the Firmware on your Intel® PAC
    - For Intel® PAC with Intel® Arria® 10 GX FPGA
    - For Intel® FPGA PAC D5005

Install the Intel® Quartus Prime Software

The Intel® Quartus® Prime software includes everything you need to design for Intel® FPGAs, from design entry and synthesis to optimization, verification, and simulation. It contains the following features:

- Hybrid Placer & Global Router
- Timing Analyzer
- Physical Synthesis
- Incremental Fitter Optimization
- Interface Planner
- Synthesis Engine
- Platform Designer
- Partial Reconfiguration
- Block-Based (Hierarchical) Design

NOTE The Intel® Quartus® Prime software is not required for the FPGA development flow's emulation or report generation stages. You can complete those stages with just the Intel® oneAPI DPC++/C++ Compiler included in the Intel oneAPI Base Toolkit.

If you want to use the Intel® Quartus® Prime software (required for FPGA hardware and simulation flow) with oneAPI, the edition of Intel® Quartus® Prime software that you need depends on your target device (either standalone device or the device on an acceleration board). Refer to the Intel® oneAPI DPC++ /C++ Library System Requirements for more information.

You have the following options for using Intel® Quartus® Prime software with oneAPI:

- If you want to install a version of the Intel® Quartus® Prime software, follow the instructions in the following documents:
  - Intel® FPGA Software Installation and Licensing
  - Intel Quartus Prime Pro Edition User Guide: Getting Started
  - Intel Quartus Prime Standard Edition User Guide: Getting Started
- If you already have a version of the Intel® Quartus® Prime software installed on your system and you want to use that version, then use one of the following methods to set up the environment:
  - Set `QUARTUS_ROOTDIR_OVERRIDE = <path_to_your_quartus_folder>`
  - Add the `bin` directory of the Intel® Quartus® Prime software to your `PATH` variable.
- If you have multiple versions of the Intel® Quartus® Prime software installed, Intel recommends setting the `QUARTUS_ROOTDIR_OVERRIDE` variable to point to the Quartus software path you want to use. Otherwise, you might end up using a version different than the one you expected. Ensure that you set the `QUARTUS_ROOTDIR_OVERRIDE` variable after running the `setvars` script, which can potentially override your setting.

Install Intel® FPGA Board Packages

To compile an executable that can run on an FPGA board, install a Board Support Package (BSP) that allows targeting compiles to that board. Intel does not ship BSPs with oneAPI. You must download and install BSPs from a third-party vendor. For more information, refer to the Intel® FPGA development flow page.

NOTE For instructions about Intel PAC platforms, refer to Install Intel® PAC BSPs.

To use a third-party vendor-provided BSP with the Intel® oneAPI Base Toolkit, follow these instructions:

1. Follow vendor-specific instructions to download and install the board package.
2. Install a version of the Intel® Quartus® Prime software that is compatible with the BSP (as indicated by the vendor).
3. Follow instructions in Install the Intel® Quartus® Prime Software to ensure the Intel oneAPI DPC++/C++ Compiler is configured to run with the installed Intel Quartus Prime software.

Related Links

- Intel® oneAPI DPC++/C++ Compiler System Requirements

- Get Started with the Intel® oneAPI Base Toolkit for Windows
- FPGA Flow in the Intel® oneAPI Programming Guide
- FPGA Optimization Guide for Intel® oneAPI Toolkits
- Intel® FPGA SDK for OpenCL™ Pro Edition: Custom Platform Toolkit User Guide
- Intel® Quartus® Prime Software User Guides

Install Intel® PAC BSPs

NOTE Intel is discontinuing the support for Programmable Acceleration Cards (PAC) with Intel Arria 10 GX FPGA and D5005. Refer to the product discontinuance notification PDN2211 for more information. Alternately, you can use custom platform or FPGA devices listed in the Intel® oneAPI DPC++ /C++ Library System Requirements

Intel® FPGA Programmable Acceleration Cards (PACs) help move, process, and store data quickly and efficiently. The performance and versatility of PAC BSPs allow you to implement solutions for data center workloads, such as streaming analytics, video transcoding, financial trading, artificial intelligence, genomics, and big data analytics.

Prerequisite

- Install Intel® Quartus® Prime Pro version 19.2 software that is compatible with the PAC. Follow instructions in Install the Intel® Quartus® Prime Software to ensure the Intel oneAPI DPC++/C++ Compiler is configured to run with the installed Intel Quartus Prime software.
- Install the Intel® Quartus® Pro Setup BSP Patch for the Intel® Quartus® Prime Pro Edition 19.2 software, and device packages compatible with your PAC BSP from Intel® Quartus® Prime Pro Edition Design Software Version 19.2 for Linux webpage (hint: See oneAPI tab).
- Download the BSP packages on systems with a physical card or the runtime system. If you need a PAC BSP (Intel® PAC with Intel Arria® 10 GX FPGA or Intel® FPGA PAC D5005), download it from the Intel® Quartus® Prime Pro Edition Design Software Version 19.2 for Linux webpage (hint: See oneAPI tab). For additional information, refer to the Intel FPGA developmental flow webpage or contact the Intel sales representatives. Unzip the PAC BSP package (`intel_a10gx_pac` or `intel_s10sx_pac`) in your desired directory.
- Install the `python-jsonschema` package on Ubuntu 18:

```
sudo apt install python-jsonschema
```

Installation Procedure

For Systems Not Connected to a PAC Board

If your system does not have a connected PAC board, but you want to use it to compile through Intel Quartus Prime software, perform the following steps on Ubuntu 18:

1. Ensure that you have not installed the PACSign package on your system. If you have it installed, then execute the following commands to uninstall the package:

```
sudo -E dpkg -r python3-opae.pacsign
sudo -E dpkg -r python3-opae.pac-sign
```

2. Move to the following directory:

- For Intel® PAC with Intel Arria® 10 GX FPGA:

```
<intel_a10gx_pac directory>/bringup/opae/pac_a10
```

- For Intel® FPGA PAC D5005:

```
<intel_s10sx_pac directory>/bringup/opae/pac_s10
```

3. Execute the following command:

```
sudo -E apt-get install -y ./python3-pacsign_1.0.7_amd64.deb
```

4. Use PAC BSP with your FPGA compiles using the following command option:

   - For Intel® PAC with Intel Arria® 10 GX FPGA:

```
-Xsboard = <intel_a10gx_pac directory>:pac_a10
```

   - For Intel® FPGA PAC D5005, run one of the following commands based on the USM support:

```
-Xsboard = <intel_s10sx_pac directory>:<pac_s10>
-Xsboard = <intel_s10sx_pac directory>:<pac_s10_usm>
```

For Systems Connected to a PAC Board

Perform these steps on systems connected to a PAC board:

1. Install one of the following based on the PAC board you are using:

   - For Intel® PAC with Intel Arria® 10 GX FPGA:

```
aocl install intel_a10gx_pac
```

   - For Intel® FPGA PAC D5005:

```
aocl install intel_s10sx_pac
```

NOTE

- Avoid running the `aocl install` command with secure boot enabled. Otherwise, an error is displayed since it is currently not supported.
- You cannot simultaneously install the software stack for Intel® PAC with Intel Arria® 10 GX FPGA and Intel® FPGA PAC D5005 on the same system. If you have installed one of them and want to install the other one, you must uninstall the former first by either running the `aocl uninstall intel_a10gx_pac` or `aocl uninstall intel_s10sx_pac` command.

2. Accept the installation prompts to proceed with installing the Intel® PAC software stack.

NOTE

- You must ensure that the FPGA development board is plugged into the system. Verify it using the `aocl diagnose` command. Perform this diagnostic only once per installation.
- The installer attempts to install the prerequisite packages. If it fails to install a package (for example, due to version conflict or repository incomplete), you can either manually install that package or run the `sudo apt -fix-broken install` command.

3. Use PAC BSP with your FPGA compiles using the following command:

   - For Intel® PAC with Intel Arria® 10 GX FPGA:

```
-Xsboard=<intel_a10gx_pac directory>:pac_a10
```

   - For Intel® FPGA PAC D5005, run one of the following commands based on the USM support:

```
-Xsboard=<intel_s10sx_pac directory>:<pac_s10>
-Xsboard=<intel_s10sx_pac directory>:<pac_s10_usm>
```

Update the Firmware on your Intel® PAC

For Intel® PAC with Intel® Arria® 10 GX FPGA

If you used your Intel® PAC with Intel® Arria® 10 GX FPGA with the 2021.1-beta06 or a lower version of the Intel® FPGA Add-on in the past, you must update the firmware to the version compatible with the acceleration stack version 1.2.1 for the FPGA to work with the 2021.1-beta07 and later versions of the Intel® PAC software stack. The firmware update files are available in the Intel® FPGA Add-on for oneAPI Base Toolkit Package, and the files are installed after you run the `aocl install` command.

Use the instructions in Identifying the Flash Image and BMC Firmware to identify your firmware version. If you need to update the firmware, use the instructions in Updating the FIM and BMC Firmware.

For Intel® FPGA PAC D5005

Each Acceleration Stack release has a unique FIM version. Use the `fpgainfo` command to identify the FIM (PR interface) and BMC firmware version:

```
sudo fpgainfo fme
```

If your Intel® FPGA PAC D5005 board firmware does not correspond to the most recent version for Acceleration Stack 2.0.1, then update your Intel® FPGA PAC D5005 board firmware from version 2.0 to 2.0.1 by performing these steps:

1. Update the FPGA Interface Manager (FIM) and BMC firmware version using the following command.

```
sudo fpgaotsu /usr/share/opae/d5005/one-time-update/base/otsu.json
```

   For more information, refer to Updating the FIM and BMC using the fpgaotsu.
2. Power cycle the server for the updates to take effect.

```
sudo fpgaotsu /usr/share/opae/d5005/one-time-update/base/otsu.json --verify --log-level debug
```

3. Run the following command to confirm whether the output matches the desired FIM and BMC version:

```
sudo fpgainfo fme
```

   For sample output, refer to Identify the FPGA Interface Manager (FIM) and BMC Firmware Version.

Install OpenCL™ Offline Compiler (OCLOC)

This topic targets Linux* users who want to use Ahead-of-Time (AOT) Compilation to generate binaries for one or multiple selected devices. For details about the AOT feature, please refer to the Intel® oneAPI DPC++/C++ Compiler Developer Guide and Reference.

There are currently three operating systems supported:

- RHEL* 8.0
- Ubuntu* 18.04
- Ubuntu* 19.10

The instructions to install the Install OpenCL™ Offline Compiler (OCLOC) are listed below.

RHEL 8.0

To access OCLOC packages, add the repository to your system and then install the packages using DNF. Root permission is required.

```
cat << EOF | sudo tee /etc/yum.repos.d/intel-graphics.repo
[intel-graphics]
name=Intel Graphics Drivers Repository baseurl=https://repositories.intel.com/graphics/rhel/8.0/
enabled=1
gpgcheck=0
EOF
sudo dnf install intel-ocloc
```

Ubuntu 18.04

NOTE For Ubuntu 18.04, the distribution is bionic in the repository line below.

To access OCLOC packages, add the repository to your system and install the packages using apt. Root permission is required.

```
cat << EOF | sudo tee /etc/apt/sources.list.d/intel-graphics.list
deb [trusted=yes arch=amd64] https://repositories.intel.com/graphics/ubuntu bionic main
EOF
sudo apt-get update
sudo apt-get install intel-ocloc
```

Ubuntu 19.10

NOTE For Ubuntu 19.10, the distribution is **eoan** in the repository line below.

To access OCLOC packages, add the repository to your system and install the packages using apt. Root permission is required.

```
cat << EOF | sudo tee /etc/apt/sources.list.d/intel-graphics.list
deb [trusted=yes arch=amd64] https://repositories.intel.com/graphics/ubuntu eoan main
EOF
sudo apt-get update
sudo apt-get install intel-opencl-icd
```

Use oneAPI Components in a Yocto Project Build

This section explains how to create a Yocto image with Intel oneAPI components.

System Requirements

Use the Yocto Project official documentation to set up and configure your host machine to make it compatible with BitBake.

Step 1: Set Up Environment

Set Up Git Repositories

The following Git repositories are required to build a Yocto image:

- Poky
- Meta-intel
- Meta-openembedded

Clone these Git repositories to your host machine:

```
git clone https://git.yoctoproject.org/git/poky --branch kirkstone
git clone https://git.yoctoproject.org/git/meta-intel --branch kirkstone
git clone https://git.openembedded.org/meta-openembedded --branch kirkstone
```

Set up BitBake Layers

Execute the following command to set up BitBake layers:

```
source poky/oe-init-build-env
bitbake-layers add-layer ../meta-intel
bitbake-layers add-layer ../meta-openembedded/meta-oe
```

Set up BitBake Configurations

Include extra configuration in **conf/local.conf`** in your build directory as required.

```
# Set machine
MACHINE = "intel-corei7-64"

# This installs oneAPI packages in the target image.
IMAGE_INSTALL:append = " setup-intel-oneapi-env"
IMAGE_INSTALL:append = " onevpl"
IMAGE_INSTALL:append = " intel-oneapi-compiler intel-oneapi-mkl intel-oneapi-ipp"
```

Step 2: Build a Yocto Image with oneAPI Packages

Run BitBake to build your image with oneAPI packages. To build the sato image, for example, run:

```
bitbake core-image-sato
```

Step 3: Verify the Yocto Image with oneAPI Packages

Verify that oneAPI packages were built successfully. Run the following command:

```
oe-pkgdata-util list-pkgs | grep intel-oneapi
```

If the image was built successfully, it will return the list of packages as below:

```
intel-oneapi-compiler
intel-oneapi-compiler-dbg
intel-oneapi-compiler-dev
intel-oneapi-ipp
intel-oneapi-ipp-dbg
intel-oneapi-ipp-dev
intel-oneapi-mkl
intel-oneapi-mkl-dbg
intel-oneapi-mkl-dev
setup-intel-oneapi-env
setup-intel-oneapi-env-dbg
setup-intel-oneapi-env-dev
```

Uninstall oneAPI Toolkits and Components

Uninstall Intel PAC Card

To uninstall an installed software stack, you can run one of the following commands:

- Intel® PAC for Intel® Arria® 10 GX FPGA

```
aocl uninstall intel_a10gx_pac
```

- Intel® FPGA PAC D5005

```
aocl uninstall intel_s10sx_pac
```

After uninstalling the PAC card, you can delete the BSP files and the PAC directory using `rm -rf <pac_install_dir>`.

Uninstall oneAPI Toolkits

NOTE Manual removal of the installation folder (not recommended) does not uninstall the oneAPI toolkits completely. Incorrect uninstallation may block future installations. If you still need to remove the toolkit manually, make sure to clean up the system correctly and remove the following:

- Installation folder
- Installer cache, located at `/var/intel/installercache/*` (root) or `<user home>/intel/installercache/*` (user), including:
  - packages cache at `/var/intel/installercache/packagescache` (root) or `<user home>/intel/installercache/packagescache` (user)
  - download cache at `/var/intel/installercache/downloadcache/` (root) or `<user home>/intel/installercache/downloadcache` (user)
  - (Optional) package manager database `/var/intel/packagemanager.db` (root) or `<user home>/intel/packagemanager.db` (user)
- Installer and package manager executables, located at `/opt/intel/oneapi/installer/` (root) or `<user home>/intel/oneapi/installer/` (user) and `/opt/intel/packagemanager/` (root) or `<user home>/intel/packagemanager/` (user), respectively.

Uninstall Using GUI

Use the following commands to uninstall oneAPI Toolkits:

```
cd /opt/intel/oneapi/installer
sudo ./installer
```

- Select Remove to remove the toolkit
- Use Installer dashboard dialog with the list of already installed products (toolkits) to Modify, Repair or Remove each toolkit separately

Uninstall Using Silent CLI

Use the following commands to uninstall oneAPI Toolkits:

1. Display the list of already installed products and products included in the downloaded package using the following command:

```
l_[Toolkit Name]Kit_[version].sh -s -a --list-products
```

Example of output:

```
ID Version Language Installed Name
============================================================================
==========================
intel.oneapi.lin.tbb.product 2021.1.1-129 false Intel® oneAPI Threading
Building Blocks
```

2. Uninstall selected product:

```
cd /opt/intel/oneapi/installer
sudo ./installer --action remove --product-id intel.oneapi.lin.tbb.product --
product-ver 2021.1.1-129
```

To uninstall a product from a specific installation instance, use the following command:

```
cd /opt/intel/oneapi/installer
sudo ./installer --action remove --product-id intel.oneapi.lin.tbb.product --
product-ver 2021.1.1-129 --instance=<instance id>
```

where `<instance id>` is a unique combination of alphanumeric symbols set during multi-instance installation.

Uninstall Using Linux Package Manager

- APT

```
sudo apt autoremove <package_name>
```

- YUM

```
sudo yum autoremove <package_name>
```

Uninstall PyTorch* and TensorFlow* (Part of Intel® AI Analytics Toolkit)

Uninstall PyTorch

1. Deactivate the pytorch environment, if activated.
2. Uninstall using the following commands:

```
conda remove -p $<install_dir>/pytorch/1.1.0/ --all
rm -rf $<install_dir>/pytorch
```

Uninstall TensorFlow

1. Deactivate the tensorflow environment, if activated.
2. Uninstall using the following commands:

```
conda remove -p $<install_dir>/tensorflow/1.14.0/ --all
rm -rf $<install_dir>/tensorflow
```

Troubleshooting

Diagnose Errors

The Diagnostics Utility for Intel oneAPI toolkits provides more checks to find missing dependencies and permissions errors. Learn more.

Integrity check failed

During installation, you may get an error message about failed integrity check of downloaded files.

Cause: Downloaded files are corrupted because of the hard drive corruption.

Solution: Check hard drive health and restart the installation.

Corrupted terminal screen

After launching the installer in CLI mode, you may see a corrupted terminal screen. Dialogs and other elements are not rendered properly.

Cause: The installer does not support terminal size less than 80x24 characters.

Solution: Resize your terminal size to 80x24 characters or greater before launching the installer.

YUM packages conflict on Amazon Linux 2* OS

When installing toolkits of version 2021.2 or 2021.3 on Amazon Linux 2* OS via YUM, you may get an error similar to the following:

```
Error: intel-oneapi-tbb-2021.2.0 conflicts with intel-oneapi-common-
licensing-2021.1.1-2021.1.1-60.noarch
```

Solution: To work around the issue, install Intel oneAPI toolkits with the following commands:

- Intel® oneAPI Base Toolkit

```
sudo yum install intel-basekit intel-oneapi-common-licensing-<version> intel-oneapi-libdpstd-
devel-<version>
```

- Intel® oneAPI HPC Toolkit

```
sudo yum install intel-hpckit intel-oneapi-common-licensing-<version> intel-oneapi-libdpstd-
devel-<version>
```

- Intel® oneAPI IoT Toolkit

```
sudo yum install intel-iotkit intel-oneapi-common-licensing-<version> intel-oneapi-libdpstd-
devel-<version>
```

- Intel® AI Analytics Toolkit

```
sudo yum install intel-aikit intel-oneapi-common-licensing-<version>
```

- Intel® oneAPI Rendering Toolkit

```
sudo yum install intel-renderkit intel-oneapi-common-licensing-<version>
```

- Intel® oneAPI DL Framework Developer Toolkit

```
sudo yum install intel-renderkit intel-oneapi-common-licensing-<version>
```

where `<version>` is 2021.2.0 or 2021.3.0. For example:

```
sudo yum install intel-basekit intel-oneapi-common-licensing-2021.2.0 intel-oneapi-libdpstd-
devel-2021.2.0
```

2021.x installation overwrites existing 2022.x installer

If you launch the installer for the 2021.x version of a toolkit on a system with the 2022.x version installed, your existing 2022.x installer will be downgraded to the 2021.x version. The 2021.x installer does not recognize installed 2022.x packages.

Cause: Compatibility issue between 2022.x and 2021.x versions of the installer.

Solution: Restore the latest installer by launching the installer for the 2022.x version of the toolkit. In future, when you need to install 2021.x on a system with 2022.x installed, before launching the installer, back up the following installer directories by renaming them:

- root: `/opt/intel/oneapi/installer` and `/opt/intel/packagemanager/1.0`
- user: `~/intel/oneapi/installer` and `~/intel/packagemanager/1.0`

When you are done with 2021.x, you can change back the directory names to restore the 2022.x installer.

Installation hangs indefinitely when launched in GUI mode using SSH

In an SSH session, installation may hang indefinitely when launched in GUI mode.

Cause: Issue detecting available screen for GUI in an SSH session.

Solution: Interrupt current process using Ctrl + C and launch the installer in CLI mode using the following command:

```
sh ./l_[Toolkit Name]Kit_[version].sh -a --cli
```

Notices and Disclaimers