Get Started with the Intel® oneAPI Base Toolkit for Linux*

Contents

Get Started with the Intel® oneAPI Base Toolkit for Linux*

1

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

Follow These Steps for the Intel® oneAPI Base Toolkit:

1. Configure Your System.
2. Build and Run a sample project using one of these methods:
   - Command Line
   - Eclipse*
   - Visual Studio Code*
3. After you have run a sample, learn more about the Intel® oneAPI Base Toolkit (Base Kit) in Next Steps.

An offline copy of this Get Started is available on the Downloadable Documentation page.

Choose Your Toolkit

> **NOTE** If you are using the Intel® oneAPI Base Toolkit, go to Configure Your System, as described in step 1 above.

If you are using a toolkit other than the Base Kit, follow the links below to find steps to configure your system and run a sample project:

- Get Started with the Intel® HPC Toolkit for Linux*
- Get Started with the Intel® AI Tools Bundle for Linux*
- Get Started with the Intel® oneAPI Deep Learning Framework Developer Toolkit for Linux*
- Get Started with the Intel® System Bring Up Toolkit for Linux*
- Get Started with the Intel® Rendering Toolkit for Linux*

Additional Tools, Libraries, and Components

To explore additional tools, libraries, and components, follow the links below:

Tools:

- Get Started with the Intel® DPC++ Compatibility Tool
- Get Started with the Intel® oneAPI DPC++/C++ Compiler
- Get Started with the Intel® Distribution for GDB* on Linux* OS Host
- Get Started with the Intel® VTune™ Profiler
- Get Started with the Intel® Advisor
- Get Started with the Intel® Distribution for Python*

Libraries:

- Get Started with the Intel® oneAPI DPC++ Library
- Get Started with the Intel® oneAPI Video Processing Library
- Get Started with the Intel® oneAPI Math Kernel Library
- Get Started with the Intel® oneAPI Threading Building Blocks
- Get Started with the Intel® Integrated Performance Primitives for Linux
- Get Started with the Intel® oneAPI oneAPI Data Analytics Library
- Get Started with the Intel® oneAPI Collective Communications Library

- Get Started with the Intel® oneAPI Deep Neural Network Library

Configure Your CPU or GPU System

Intel® oneAPI Base Toolkit

> **NOTE** If you are working with an FPGA system, see Configure Your FPGA System.

- CPU, GPU and FPGA users, install CMake* and pkg-config to build most samples
- Set environment variables for CPU/GPU
- For GPU users, install GPU drivers
- For GPU users, disable hangcheck for applications with long-running GPU compute workloads
- For GPU non-root users, add a user to the video group
- For Eclipse* users, install a recent release of the Eclipse IDE for C/C++ Developers, then install the oneAPI Eclipse plugins.

Install CMake*, pkg-config, and GNU* Dev Tools to Build Samples

Although the CMake and pkg-config build tools are not required by the oneAPI tools and toolkits, many oneAPI samples are provided as CMake projects and require CMake to build them. In some cases pkg-config is necessary to locate libraries needed to complete a build of the application.

The Intel compilers utilize the existing GNU build toolchains to provide a complete C/C++ development environment. If your distribution of Linux does not include the complete suite of GNU development tools, you need to install these tools.

To install CMake, pkg-config, and the GNU development tools on your Linux system, open a terminal session and enter the following commands:

Ubuntu*

```
sudo apt update
sudo apt -y install cmake pkg-config build-essential
```

Red Hat* and Fedora*

```
sudo yum update
sudo yum -y install cmake pkgconfig
sudo yum groupinstall "Development Tools"
```

SUSE*

```
sudo zypper update
sudo zypper --non-interactive install cmake pkg-config
sudo zypper --non-interactive install pattern devel_C_C++
```

Verify the installation by displaying the installation location with this command:

```
which cmake pkg-config make gcc g++
```

One or more of these locations will display:

```
/usr/bin/cmake

/usr/bin/pkg-config

/usr/bin/make

/usr/bin/gcc

/usr/bin/g++
```

For more information about CMake, refer to CMake.org. If you are unable to install CMake using your Linux distribution's standard package manager, see the CMake.org downloads page for additional installation options.

Set Environment Variables for CLI Development

The Unified Directory Layout was implemented in 2024.0. If you have multiple toolkit versions installed, the Unified layout adds the ability to ensure your development environment contains the component versions that were released as part of that specific toolkit version.

The directory layout that was used prior to 2024.0 is still supported on new and existing installations. This prior layout is called the Component Directory Layout. Now you have the option to use the Component Directory Layout or the Unified Directory Layout.

Environment variables are set up with a script called setvars or oneapi-vars, depending on which layout you are using.

To understand more about the Unified Directory Layout, including how the environment is initialized and the advantages of using the layout, see Use the setvars and oneapi-vars Scripts with Linux *.

Set up your CLI environment:

Option 1: Source setvars.sh once per session

Source `setvars.sh | oneapi-vars.sh` every time you open a new terminal window:

Component Directory Layout

For system wide installations:

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

For private installations:

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

Unified Directory Layout

For system wide installations:

```
. /opt/intel/oneapi/<toolkit-version>/oneapi-vars.sh
```

For private installations:

```
. ~/intel/oneapi/<toolkit-version>/oneapi-vars.sh
```

Option 2: One time setup for setvars.sh

Environment variables can be set up to automatically set using one of the methods below:

- Use a startup script, as described on this page
- Use modulefiles
- Use a setvars.sh configuration file

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 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 set up the environment in a non-POSIX shell, see oneAPI Development Environment Setup for more configuration options.

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 to disable hangcheck persistently (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 this text between the quotes (""):

```
i915.enable_hangcheck=0
```

5. Run this command:

```
sudo update-grub
```

6. Reboot the system. Hangcheck remains disabled.

For GPU Users, Install GPU Drivers

To develop and run C++ and SYCL* applications for your Intel GPU on Linux, you must first install the latest Intel GPU drivers. Driver defaults are only appropriate for display applications. For other applications, defaults should be adjusted by udev rules as described in Set Up User Permissions for Using the Device files for Intel GPUs.

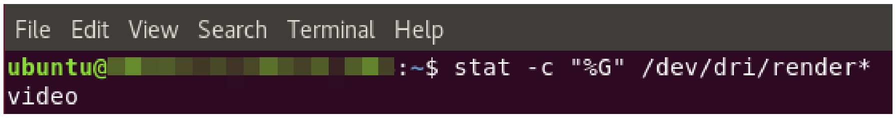
GPU: Add User to Video Group

For GPU compute workloads, non-root (normal) users do not typically have access to the GPU device.

1. To give a user access, first determine which group name is assigned ownership of the render nodes:

```
stat -c "%G" /dev/dri/render*
```

In this example, the **video** group is assigned ownership of the render nodes:

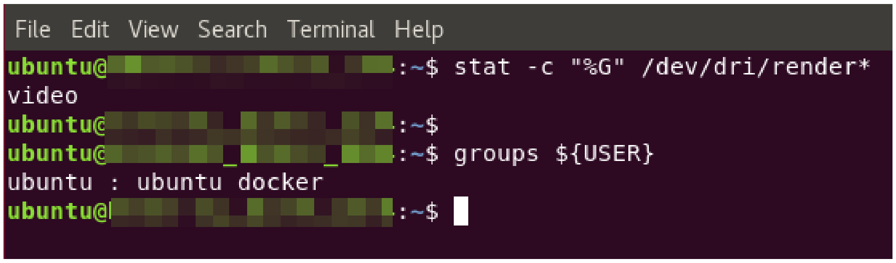


```
ubuntu@[redacted]:~$ stat -c "%G" /dev/dri/render*
video
```

2. Determine if the current user is a member of that render node group:

```
groups ${USER}
```

In this example the current user is not included in the **video** group.



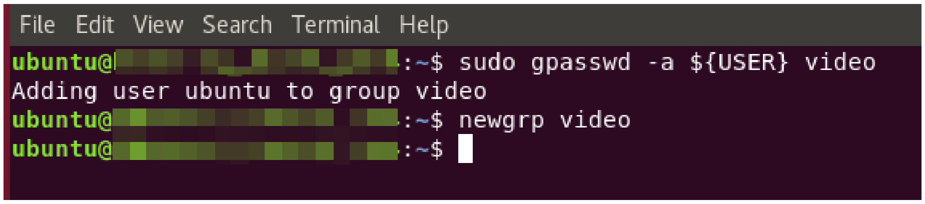
```
ubuntu@[redacted]:~$ stat -c "%G" /dev/dri/render*
video
ubuntu@[redacted]:~$
ubuntu@[redacted]:~$ groups ${USER}
ubuntu : ubuntu docker
ubuntu@[redacted]:~$
```

3. Add the user to the group using `gpasswd`:

```
sudo gpasswd -a ${USER} video
```

4. Activate the **video** group:

```
newgrp video
```



```
ubuntu@[redacted]:~$ sudo gpasswd -a ${USER} video
Adding user ubuntu to group video
ubuntu@[redacted]:~$ newgrp video
ubuntu@[redacted]:~$
```

Next Step

Run a sample project using one of these methods:

- Command Line
- Eclipse*
- Visual Studio Code

Configure Your FPGA System

NOTE If you are working with a CPU or GPU system, see Configure Your CPU or GPU System .

To set up your system, you need to:

- Install CMake* and pkg-config to build most samples
- Set environment variables

Install CMake*, pkg-config, and GNU* Dev Tools to Build Samples

Although the CMake and pkg-config build tools are not required by the oneAPI tools and toolkits, many oneAPI samples are provided as CMake projects and require CMake to build them. In some cases pkg-config is necessary to locate libraries needed to complete a build of the application.

The Intel compilers utilize the existing GNU build toolchains to provide a complete C/C++ development environment. If your distribution of Linux does not include the complete suite of GNU development tools, you need to install these tools.

To install CMake, pkg-config, and the GNU development tools on your Linux system, open a terminal session and enter the following commands:

Ubuntu*

```
sudo apt update
sudo apt -y install cmake pkg-config build-essential
```

Red Hat* and Fedora*

```
sudo yum update
sudo yum -y install cmake pkgconfig
sudo yum groupinstall "Development Tools"
```

SUSE*

```
sudo zypper update
sudo zypper --non-interactive install cmake pkg-config
sudo zypper --non-interactive install pattern devel_C_C++
```

Verify the installation by displaying the installation location with this command:

```
which cmake pkg-config make gcc g++
```

One or more of these locations will display:

```
/usr/bin/cmake

/usr/bin/pkg-config

/usr/bin/make

/usr/bin/gcc

/usr/bin/g++
```

For more information about CMake, refer to CMake.org. If you are unable to install CMake using your Linux distribution's standard package manager, see the CMake.org downloads page for additional installation options.

Set Environment Variables for CLI Development

The Unified Directory Layout was implemented in 2024.0. If you have multiple toolkit versions installed, the Unified layout adds the ability to ensure your development environment contains the component versions that were released as part of that specific toolkit version.

The directory layout that was used prior to 2024.0 is still supported on new and existing installations. This prior layout is called the Component Directory Layout. Now you have the option to use the Component Directory Layout or the Unified Directory Layout.

Environment variables are set up with a script called setvars or oneapi-vars, depending on which layout you are using.

To understand more about the Unified Directory Layout, including how the environment is initialized and the advantages of using the layout, see Use the setvars and oneapi-vars Scripts with Linux *.

Set up your CLI environment:

Option 1: Source setvars.sh once per session

Source `setvars.sh | oneapi-vars.sh` every time you open a new terminal window:

Component Directory Layout

For system wide installations:

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

For private installations:

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

Unified Directory Layout

For system wide installations:

```
. /opt/intel/oneapi/<toolkit-version>/oneapi-vars.sh
```

For private installations:

```
. ~/intel/oneapi/<toolkit-version>/oneapi-vars.sh
```

Option 2: One time setup for setvars.sh

Environment variables can be set up to automatically set using one of the methods below:

- Use a startup script, as described on this page
- Use modulefiles
- Use a setvars.sh configuration file

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 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 set up the environment in a non-POSIX shell, see oneAPI Development Environment Setup for more configuration options.

Run a Sample Project

Run a sample project using the Command Line or using Eclipse*.

Build and Run a Sample Project Using the Command Line

Intel® oneAPI Base Toolkit

> **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.

Command line development can be done with a terminal window or done through Visual Studio Code*. Some tasks can be automated using extensions. To learn more, see Using Visual Studio Code with Intel® oneAPI Toolkits.

To compile and run a sample:

1. Locate a sample project using the oneAPI CLI Samples Browser.

2. Build and run a sample project using Make* or CMake*.

Download Samples using the oneAPI CLI Samples Browser

Use the oneAPI CLI Samples Browser to browse the collection of online oneAPI samples. As you browse the oneAPI samples, you can copy them to your local disk as buildable sample projects. Most oneAPI sample projects are built using Make or CMake, so the build instructions are included as part of the sample in a README file. The oneAPI CLI utility is a single-file, stand-alone executable that has no dependencies on dynamic runtime libraries.

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

An internet connection is required to download the samples for oneAPI toolkits. For information on how to use this toolkit offline, see Developing with Offline Systems in the Troubleshooting section.

To watch a video presentation of how to create a project with the command line, see Exploring Intel© oneAPI Samples from the Command Line.

1. Open a terminal window.
2. If you did not complete the steps in Option 2: One time set up for setvars.sh in the Configure Your System section, set system variables by sourcing setvars:

   Component Directory Layout:

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

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

   For private installations:

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

   Unified Directory Layout:

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

```
. /opt/intel/oneapi/<toolkit-version>/oneapi-vars.sh
```

   For private installations:

```
. ~/intel/oneapi/<toolkit-version>/oneapi-vars.sh
```

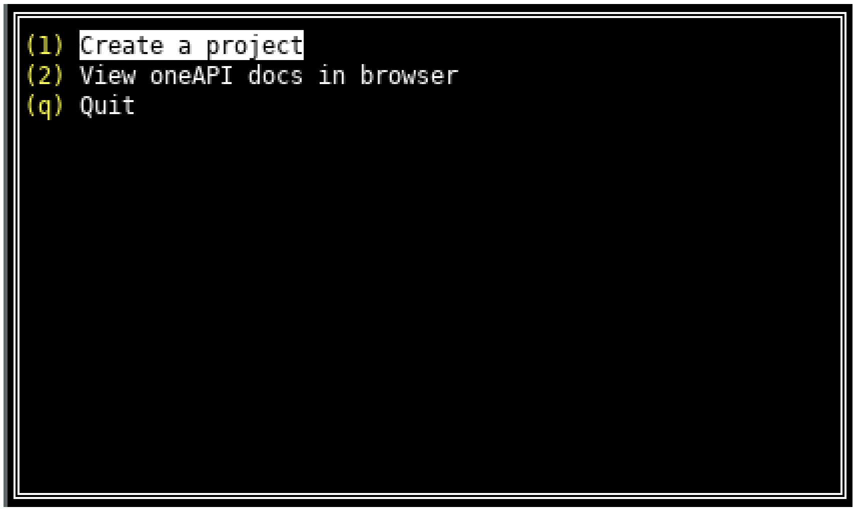
   The command above assumes you installed to the default folder. If you customized the installation folder, `setvars | oneapi-vars` is in your custom folder.

   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.

3. In the same terminal window, run the application (it should be in your PATH):

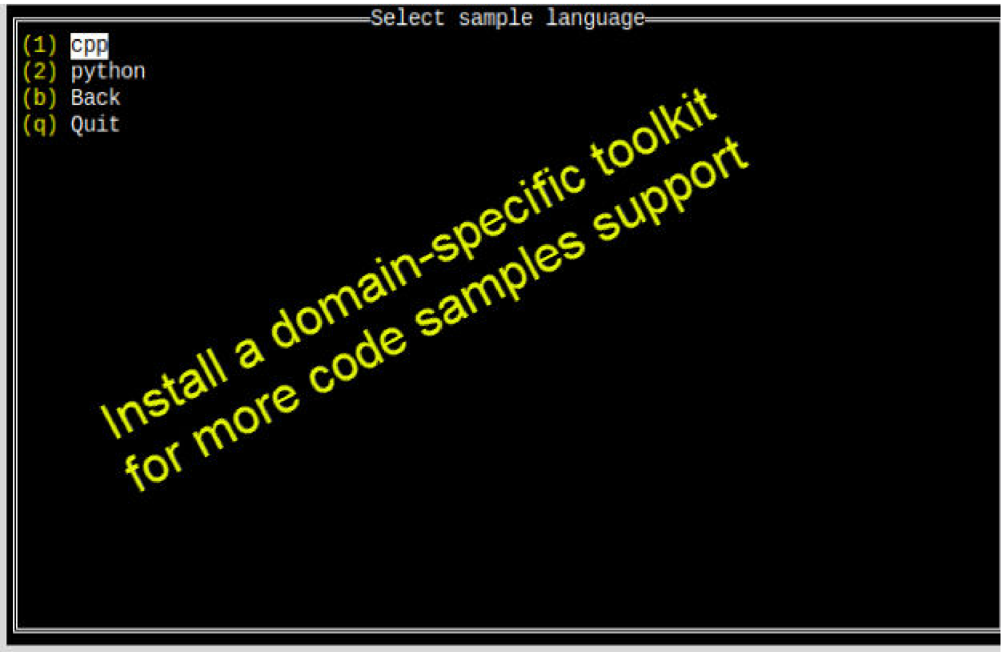
```
oneapi-cli
```

   The oneAPI CLI menu appears:



```
(1) Create a project
(2) View oneAPI docs in browser
(q) Quit
```

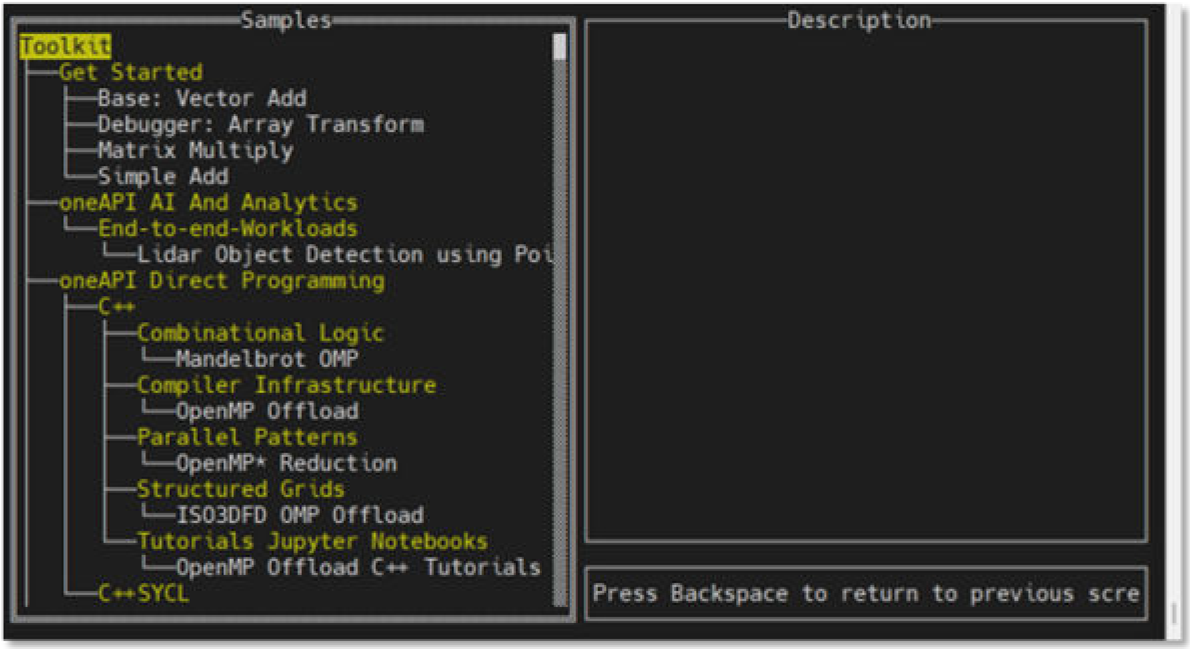
4. Use the up and down arrow keys to select **Create a project**, then press Enter
5. Move the arrow key down to select **Create a project**, then press Enter. The language selection will appear. If you want to run samples from a toolkit other than the Intel® oneAPI Base Toolkit, install the domain-specific toolkit before proceeding.



```
Select sample language
(1) cpp
(2) python
(b) Back
(q) Quit
```

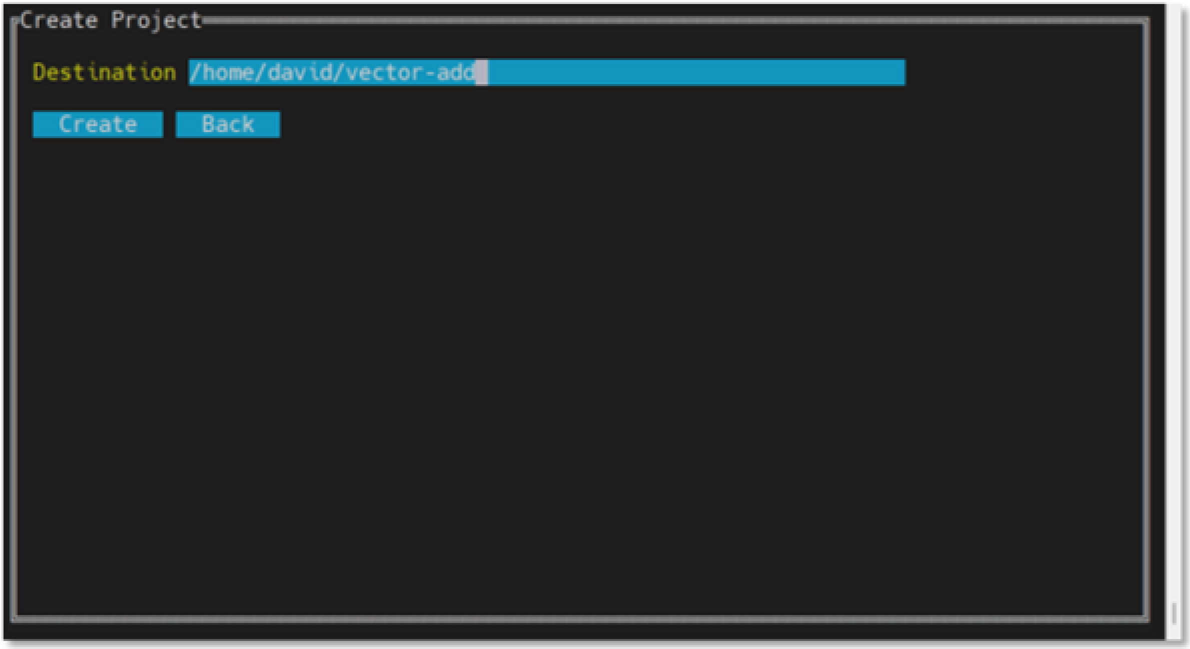
Install a domain-specific toolkit
for more code samples support

6. Select the language for your sample. For your first project, select **cpp**, then press Enter. The toolkit samples list appears.
7. Select the **Vector Add** sample. Vector Add is a simple test application that will help verify that the tools are setup correctly and can access your system's GPU:



```
Samples
Toolkit
├─Get Started
│ ├─Base: Vector Add
│ ├─Debugger: Array Transform
│ ├─Matrix Multiply
│ └─Simple Add
├─oneAPI AI And Analytics
│ └─End-to-end-Workloads
│   └─Lidar Object Detection using Poi
├─oneAPI Direct Programming
│ ├─C++
│ │ ├─Combinational Logic
│ │ │ └─Mandelbrot OMP
│ │ ├─Compiler Infrastructure
│ │ │ └─OpenMP Offload
│ │ ├─Parallel Patterns
│ │ │ └─OpenMP* Reduction
│ │ ├─Structured Grids
│ │ │ └─ISO3DFD OMP Offload
│ │ └─Tutorials Jupyter Notebooks
│ │   └─OpenMP Offload C++ Tutorials
│ └─C++SYCL
```

8. After you select a sample, press Enter.
9. Specify the location for the project. The default location includes the path from where the utility was run and the name of the project.
10. Press Tab to select Create, then press Enter:



```
Create Project
Destination /home/david/vector-add
```

Build and Run a Project Based on a oneAPI Sample Using Make

NOTE Some samples require additional steps or arguments for building and/or running the sample. Review the sample's `README.md` file for specific details regarding how to build and run the sample.

1. Open a command prompt.

2. Navigate to the directory which you specified when creating (downloading) the project.
3. Configure the project. Choose from one of the following two options:

   Buffer-based implementation:

```
mkdir build
cd build
cmake ..
```

   Unified Shared Memory (USM) based implementation:

```
mkdir build
cd build
cmake .. -DUSM=1
```

4. Build the program.

```
make cpu-gpu
```

5. Run the program for Unified Shared Memory (USM) and buffers.

```
./vector-add-buffers
./vector-add-usm
```

6. Optional: Clean the program.

```
make clean
```

Build and Run a Project Based on a oneAPI Sample using CMake

Vector add can only be run with Make. To run a different sample using CMake, where `<sample_name>` is the name of the sample you want to run.

To find a sample that uses CMake, browse the oneAPI Samples GitHub repository and view the `README` file for each sample to see if the sample can be run with CMake.

1. If necessary, re-run the command-line utility and select a CMake project that contains a `CMakeLists.txt` file.

```
cd <sample_name>
```

2. Navigate to the build directory.

```
mkdir build
cd build
```

3. **NOTE** Some samples require additional steps or arguments for building and/or running the sample. Review the sample's `README.md` file for specific details regarding how to build and run the sample.

   Build the program. Run CMake in the build directory to create the makefile. Use the newly created makefile to build the executable.

```
cmake ../.
make VERBOSE=1
```

4. Run the program.

```
make run
```

5. Clean the program.

```
make clean
```

See Explore SYCL* Through Samples to learn more.

Compile and run a sample for FPGA

You can run the vector-add sample (or any FPGA SYCL code) in the following modes:

- **Emulation:** Verifies the code correctness. Compilation completes in few seconds. Use this mode if you are using the Intel® oneAPI Base Toolkit
- **Report:** Generates a static optimization report for design analysis. Compilation can take a few minutes to complete. When completed, you can find the reports in <project_name>.prj/reports/report.html. This can be used with the Intel® oneAPI Base Toolkit. For more information about the reports, refer to the FPGA Optimization Guide for Intel® oneAPI Toolkits.
- **Hardware:** Generates the actual bitstream on an FPGA device. Compilation can take few hours to complete. Use this mode to measure performance. To use this mode, download the Intel® Quartus® Prime Pro Edition software and BSPs separately. For more information, refer to the Install Software for Intel® FPGA Development Flow section in the *Intel® oneAPI Toolkits Installation Guide for Linux* OS* and Intel® FPGA Add-On for oneAPI Base Toolkit web page.

After downloading the vector-add design example using the oneAPI CLI Samples Browser, perform the following steps to compile and run the design:

1. Change to the directory containing the vector-add design example using:

```
cd <vector-add directory on the same system>
```

2. Run the following make clean command before you start compiling.

```
make clean
```

3. Compile the design using one of the following options:

- Compile for emulation using:

```
make fpga_emu
```

- Compile the report using:

```
make report
```

NOTE You can view the report at vector-add_report.prj/reports/report.html. This does not generate an executable.

- Compile for hardware (takes a longer duration to complete) using:

```
make fpga
```

NOTE If you compiled the hardware on a development system, copy the executable file vector-add.fpga to the runtime system.

4. Run the design using one of the following options:

- Run the design for emulation using:

```
./vector-add-buffers.fpga_emu
./vector-add-usm.fpga_emu
```

NOTE If you are using a separate development system, perform this step on that system.

- Run the design on FPGA hardware using:

```
./vector-add-buffers.fpga
./vector-add-usm.fpga
```

NOTE To understand the command used in the Makefile, refer to the Intel® oneAPI Programming Guide.

See Explore SYCL* Through Samples to learn more.

Next Steps

After successfully building a sample application, Explore SYCL with Samples from Intel and explore the tools in the Intel® oneAPI Base Toolkit.

Tool	Description
Intel® DPC++ Compatibility Tool	The Intel® DPC++ Compatibility Tool assists in migration of CUDA* applications to SYCL* ready code that can use the Intel® oneAPI DPC++/C++ Compiler. Get started.
Intel® oneAPI DPC++/C++ Compiler	The Intel® oneAPI DPC++/C++ Compiler targets CPUs and accelerators through single-source code while permitting custom tuning. Get Started.
Intel® oneAPI DPC++ Library	This library is a companion to the Intel® oneAPI DPC++/C++ Compiler and provides an alternative for C++ developers who create heterogeneous applications and solutions. Its APIs are based on familiar standards-C++ STL, Parallel STL (PSTL), Boost.Compute, and SYCL*-to maximize productivity and performance across CPUs, GPUs, and FPGAs.
Intel® Distribution for GDB*	GDB, the GNU Project debugger, allows you to see what is going on `inside' another program while it executes -- or what another program was doing at the moment it crashed. Get Started. Learn more.
Intel® oneAPI Math Kernel Library (oneMKL)	The oneMKL helps you achieve maximum performance with a math computing library of highly optimized, extensively parallelized routines for CPU and GPU. The library has C and Fortran interfaces for most routines on CPU, and SYCL* interfaces for some routines on both CPU and GPU. Get started.
Intel® oneAPI Threading Building Blocks (oneTBB)	oneTBB is a flexible performance library that simplifies the work of adding parallelism to complex applications, even if you're not a threading expert. Learn more.
Intel® Integrated Performance Primitives	Intel® Integrated Performance Primitives (Intel® IPP) is an extensive library of ready-to-use, domain-specific functions that are highly optimized for diverse Intel® architectures. Get Started with Intel® IPP or Get Started with Intel® IPP Cryptography.
Intel® oneAPI Data Analytics Library	The Intel® oneAPI Data Analytics Library (oneDAL) is a library that helps speed up big data analysis by providing highly optimized algorithmic building blocks for all stages of data analytics (preprocessing, transformation, analysis, modeling, validation, and decision making) in batch, online, and distributed processing modes of computation. The current version of oneDAL provides SYCL* API extensions to the traditional C++ interface. Get started.
Intel® oneAPI Collective Communications Library	Intel® oneAPI Collective Communications Library (oneCCL) is a scalable and high-performance communication library for Deep Learning (DL) and Machine Learning (ML) workloads. It develops the ideas originated in Intel(R)

	Machine Learning Scaling Library and expands the design and API to encompass new features and use cases. Get started.
Intel® oneAPI Deep Neural Network Library	The Intel® oneAPI Deep Neural Network Library (oneDNN) is an open-source performance library for deep learning applications. The library includes basic building blocks for neural networks optimized for Intel® Architecture Processors and Intel® Processor Graphics. oneDNN is intended for deep learning applications and framework developers interested in improving application performance on Intel CPUs and GPUs. Get started.
Intel® VTune™ Profiler	Intel® VTune™ Profiler is a performance analysis tool targeted for users developing serial and multithreaded applications. The tool is delivered as a Performance Profiler with Intel Performance Snapshots and supports local and remote target analysis on the Windows* and Linux* platforms. Get started.
Intel® Advisor	Intel® Advisor gives software architects and developers the data and analysis tools they need to build well-threaded and vectorized code that exploits modern hardware capabilities. Get started.
Intel® FPGA Add-on for oneAPI Base Toolkit	Use reconfigurable hardware to accelerate data-centric workloads. Learn more.

Learn more about SYCL and targeting other accelerators using the following resources:

Resource	Description
Intel® oneAPI Programming Guide	Provides details on the oneAPI programming model, including details about SYCL standards, programming for various target accelerators, and introductions to the oneAPI libraries.
Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel® Arria 10 GX FPGA	A quick start guide for the Intel® Programmable Acceleration Card with Intel® Arria® 10 GX FPGA. This guide provides instructions to load and run a loopback test, Hello FPGA, in both non-virtualized and virtualized environments.
Intel Acceleration Stack Quick Start Guide: Intel® FPGA Programmable Acceleration Card D5005	A quick start guide for Intel® FPGA PAC D5005. This guide provides the instructions for installing the Open Programmable Acceleration Engine (OPAE) on the host Intel® Xeon® Processor to manage and access the Intel® FPGA PAC, configuring and flashing the FPGA image and Board Management Controller (BMC) images, running the example `hello_afu` in a virtualized and non-virtualized environment, and handling graceful thermal shutdown.
Intel® FPGA SDK for OpenCL™ Pro Edition: Custom Platform Toolkit User Guide	Outlines the procedure for creating an Intel® FPGA Software Development Kit (SDK) for OpenCL™ Pro Edition Custom Platform. The Intel® FPGA SDK for OpenCL™ Pro Edition Custom Platform Toolkit provides the necessary tools for implementing a fully functional Custom Platform.
Intel® FPGA Acceleration Hub	Learn about the Acceleration Stack for Intel® Xeon® CPUs with FPGAs and other Intel FPGA-based acceleration platforms.
FPGA Optimization Guide for Intel® oneAPI Toolkits	The oneAPI FPGA Optimization Guide provides guidance on leveraging the functionalities of SYCL to optimize your design.
Explore SYCL Through Intel® FPGA Code Samples	Provides guidance on how to target and develop your design on an FPGA using the oneAPI programming model. It also provides guidance on how to optimize a design to achieve performance and latency targets for an application targeting the FPGA.
oneAPI GPU Optimization Guide	The oneAPI GPU Optimization Guide demonstrates how to improve the behavior of your software by partitioning it across the host and accelerator to specialize portions of

	the computation that run best on the accelerator. Specialization includes restructuring and tuning the code to create the best mapping of the application to the hardware. The value of oneAPI is that it allows each of these variations to be expressed in a common language with device-specific variants launched on the appropriate accelerator.

For more information about this toolkit, see the Intel® oneAPI Base Toolkit page.

SSH: Password-less Access to Remote Linux* Target

Introduction

For some oneAPI applications, you must configure a password-less SSH connection for the root user on the target system; for example:

- IoT applications that use the MRAA/UPM sensor library
- Any application that interacts with system resources that require su, sudo, or root access
- Any tool that requires remote root or sudo access to your target system

When you finish the configuration steps below you will be able to "ssh into" your remote Linux target from your host development system without a password prompt, as a normal (non-root) user or as a root user.

For an introduction to SSH and how SSH keys work, see SSH Essentials: Working with SSH Servers, Clients, and Keys.

NOTE Password-less access works only when you connect to your target system from your host development system with a matching private SSH key. Attempting to connect from a different host system will still require a password.

Configure Password-less SSH Access

These instructions apply to:

- **Host development system:** Linux*, Windows*, or macOS*
- **Target system:** Linux

Set up an .ssh directory

On your host development system, follow these steps:

Step 1: Open a terminal session (CMD window on Windows) and CD to your home directory
Step 2: Create .ssh directory

Enter the following commands to create an .ssh directory, set the proper permissions, and CD into the new .ssh directory.

At a Windows CMD prompt

```
> %HomeDrive%  &&  cd %HomePath%
> mkdir .ssh
> cd .ssh
```

At a Linux terminal (bash) prompt

```
$ cd ~
$ mkdir -p .ssh
$ chmod 700 .ssh
$ cd .ssh
```

Step 3: Generate keys and copy to the target system

NOTE From this point forward the instructions apply to all host development systems (Windows, Linux, and macOS).

1. To generate a *default-named* RSA key pair with an *empty passphrase* (that is, do not provide a passphrase when asked), enter:

```
$ ssh-keygen -t rsa
```

2. To copy the new public key to your target system's *non-root user* home folder, enter the following, where:

   username = the name used to access the target and *target* = the IP address or the network hostname of the target

   You should be prompted for the non-root user password for your target device.

```
$ scp id_rsa.pub username@target:id_rsa.pub
$ ssh username@target
$ cd ~
$ mkdir -p .ssh
$ chmod 700 .ssh
$ cat ~/id_rsa.pub >>.ssh/authorized_keys
$ chmod 600 .ssh/authorized_keys
$ exit
```

Step 4: Confirm that a password is no longer required (non-root)

Follow this step to confirm that a password is no longer required for your non-root user.

1. To display the target's **system information** strings, including the target's hostname as the second field in the output, enter:

```
ssh username@target uname -a
```

Step 5: Configure password-less access to root on your target

1. To login to the *non-root user* on the target using SSH and switch to the *root user* using sudo, enter:

```
$ ssh username@target
$ cd ~
$ sudo -E bash
```

   Note that the sudo command should prompt you for your target system's *non-root user* password.

2. To copy the public key that you transferred to the non-root user account on the target into the root user's authorized keys file, enter:

```
$ mkdir -p /root/.ssh
$ chmod 700 /root/.ssh
$ cat ./id_rsa.pub >>/root/.ssh/authorized_keys
$ chmod 600 /root/.ssh/authorized_keys
```

3. Exit twice, first from the sudo bash session, second from the ssh connection:

```
$ exit
$ exit
```

Step 6: [Optional] Check your progress

To test the root connection for your target, enter:

```
$ ssh root@target ls -a
```

You should see a directory listing of all files located in the /root folder on your target, without the need for a login prompt.

Next: Create a New Connection and Connect to Your Target

Notes

- Password-less access works only when you connect to your target system from your host development system with a matching private SSH key. Attempting to connect from a different host system will still require a password.
- Make sure that you have created a project for Linux targets, and that this project is selected in the Project Explorer.

See Also

Why sudo user can use sched_setscheduler SCHED_RR while root can not?

Operation not permitted while setting new priority for thread

Debugging libmraa {#debugging}

SSH: Running Applications Built with Eclipse*

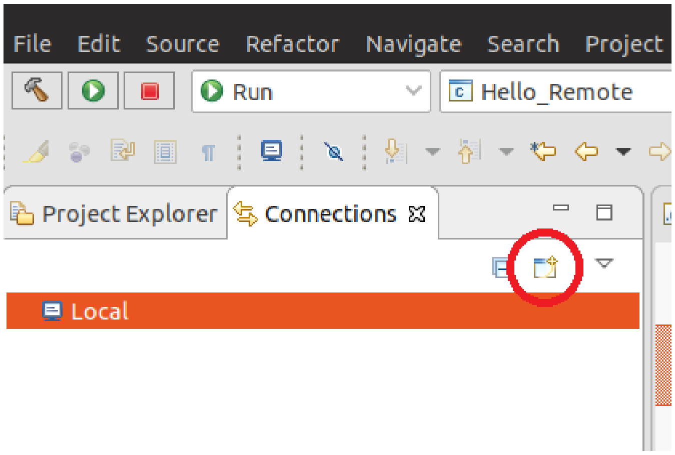
The following describes how to run a built application on a "remote target" system using an SSH connection.

If your application requires a password-less SSH connection, see Password-less Access to a Remote Linux* Target Device.

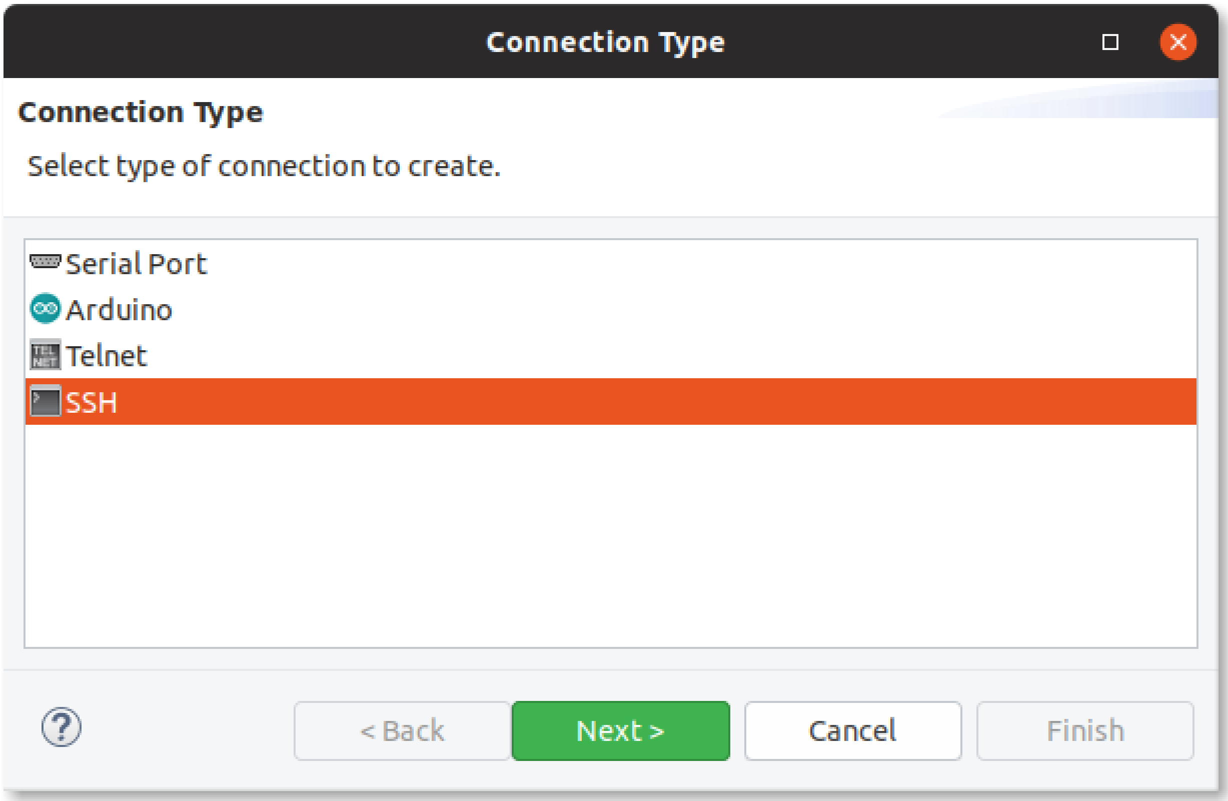
This section assumes:

- you already have an Eclipse project
- have successfully built an application from that project
- have some familiarity with Eclipse
- remote target is already configured for remote access via SSH

1. Create a Connection within Eclipse by selecting the **New Connection** option:



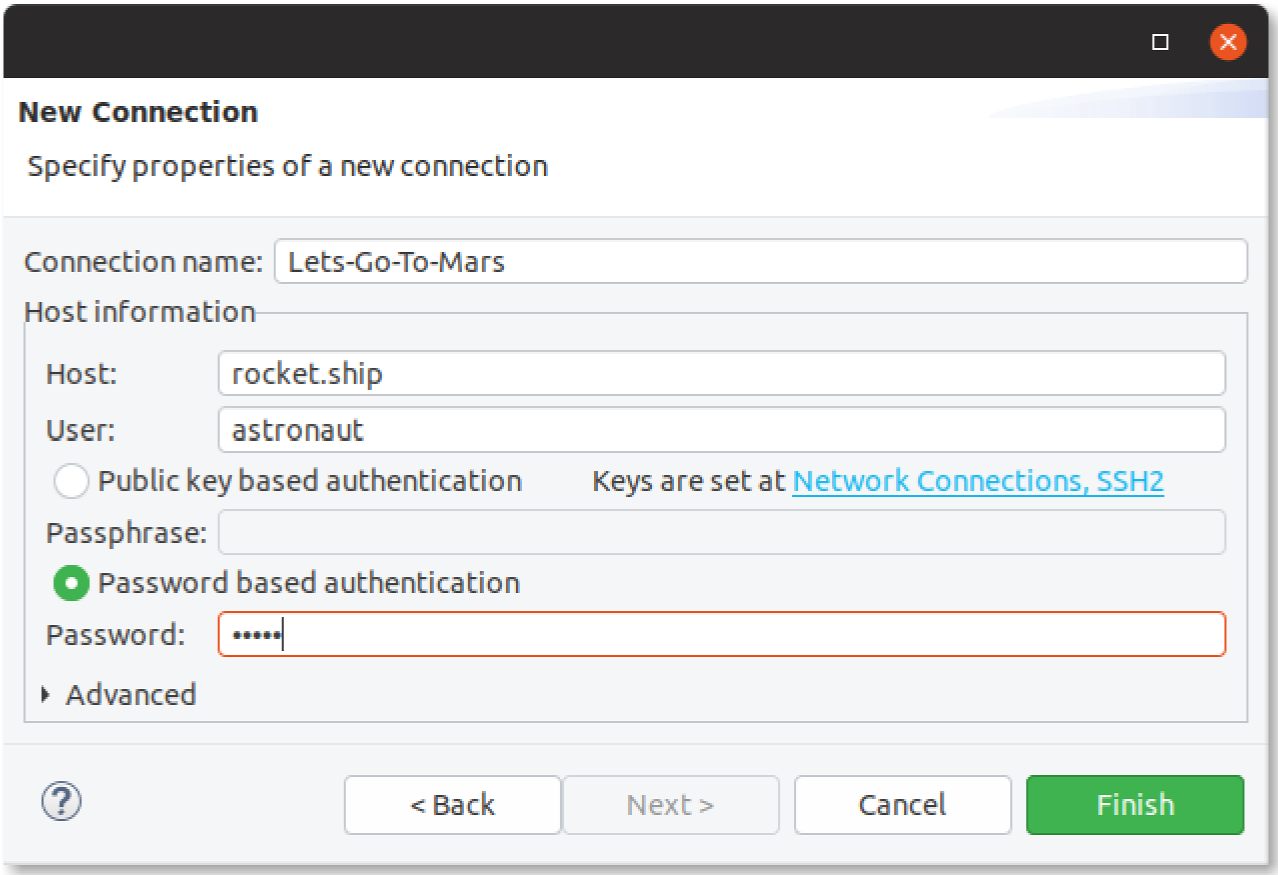
2. Select the SSH Connection type and click **Next**:



Connection Type

Select type of connection to create.

3. Enter your remote targets Hostname and Username. You can either use Key-based authentication or Password, depending on the configuration of the remote device.



New Connection

Specify properties of a new connection

Connection name: Lets-Go-To-Mars

Host information

Host: rocket.ship

User: astronaut

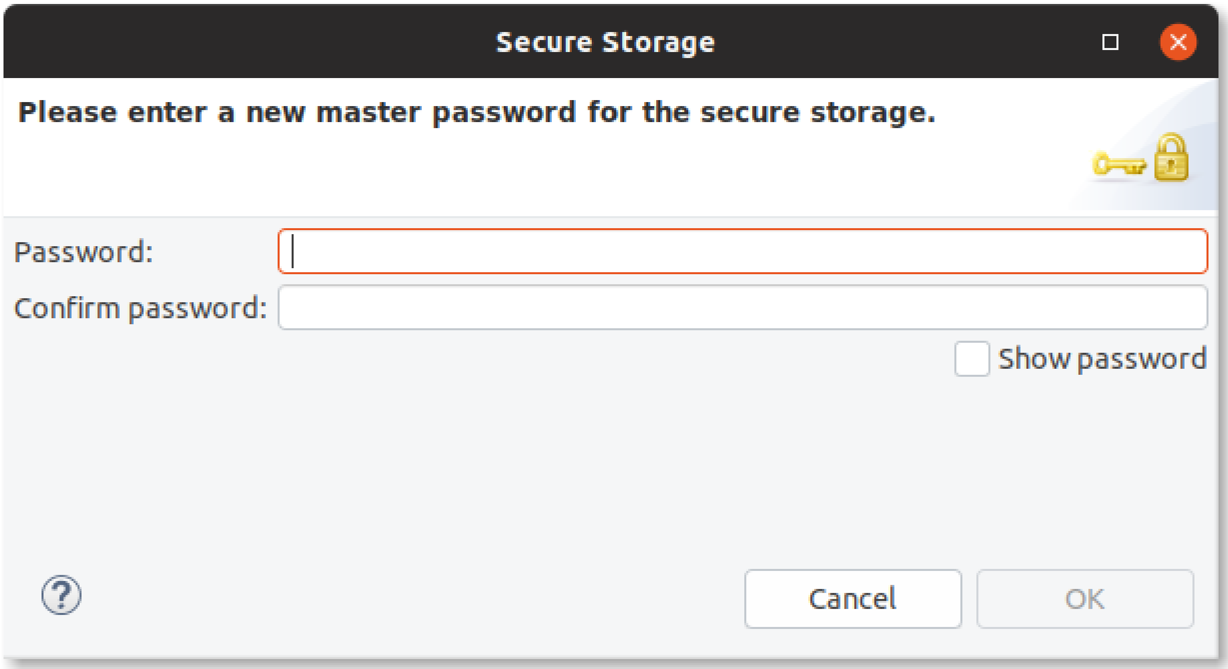
○ Public key based authentication   Keys are set at Network Connections, SSH2

Passphrase:

● Password based authentication

Password: •••••

4. You may also be then prompted to set an Eclipse Secure Storage password. This protects the password you entered when stored on the disk. Enter the password and click OK.



Secure Storage

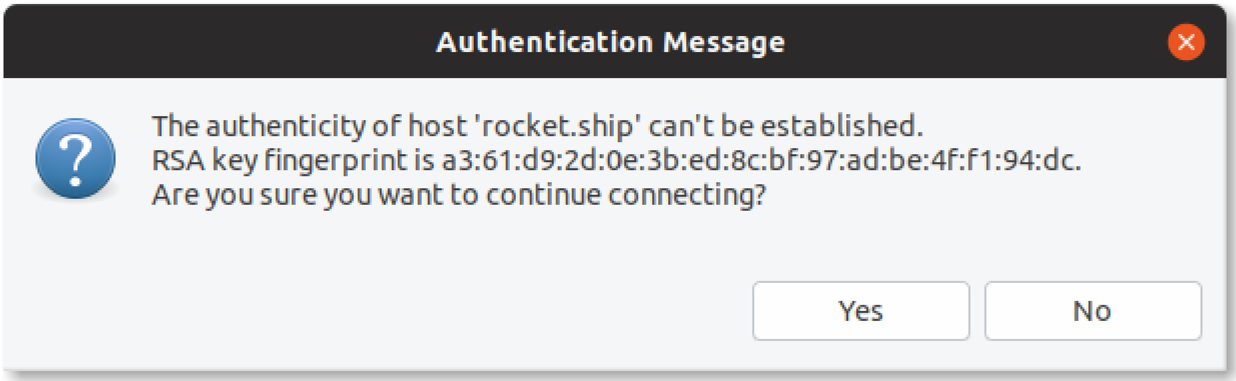
Please enter a new master password for the secure storage.

Password:

Confirm password:

□ Show password

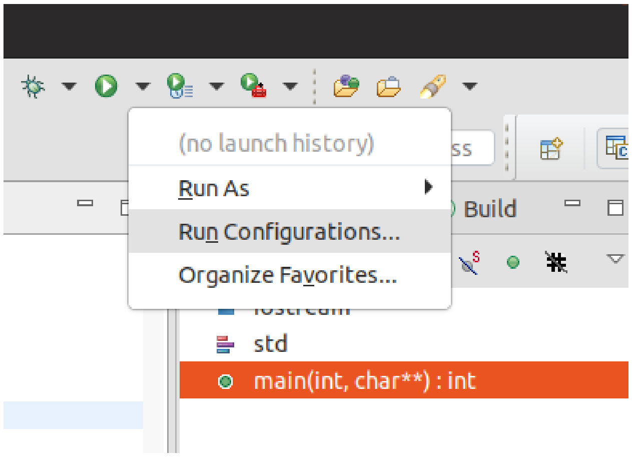
5. When connecting to the device for the first time, you should be prompted to trust the device. Click Yes to continue.



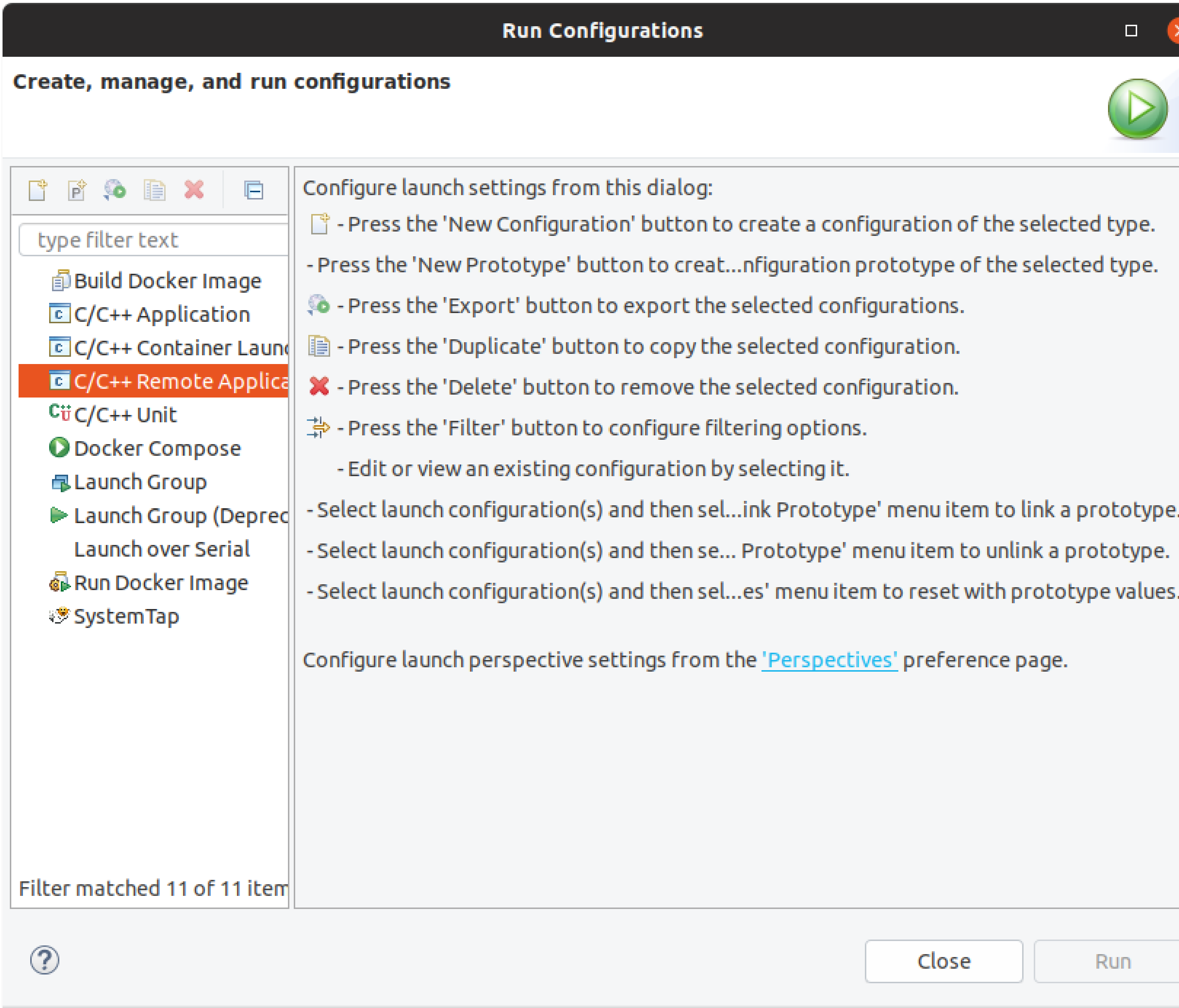
Authentication Message

The authenticity of host 'rocket.ship' can't be established.
RSA key fingerprint is a3:61:d9:2d:0e:3b:ed:8c:bf:97:ad:be:4f:f1:94:dc.
Are you sure you want to continue connecting?

6. Once a connection has been defined within Eclipse, open the the Run Configurations window to create your configuration.



7. Double-click on the "C/C++ Remote Application" configuration type.



Create, manage, and run configurations

Configure launch settings from this dialog:

- Press the 'New Configuration' button to create a configuration of the selected type.
- Press the 'New Prototype' button to creat...nfiguration prototype of the selected type.
- Press the 'Export' button to export the selected configurations.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.
- Select launch configuration(s) and then sel...ink Prototype' menu item to link a prototype.
- Select launch configuration(s) and then se... Prototype' menu item to unlink a prototype.
- Select launch configuration(s) and then sel...es' menu item to reset with prototype values.

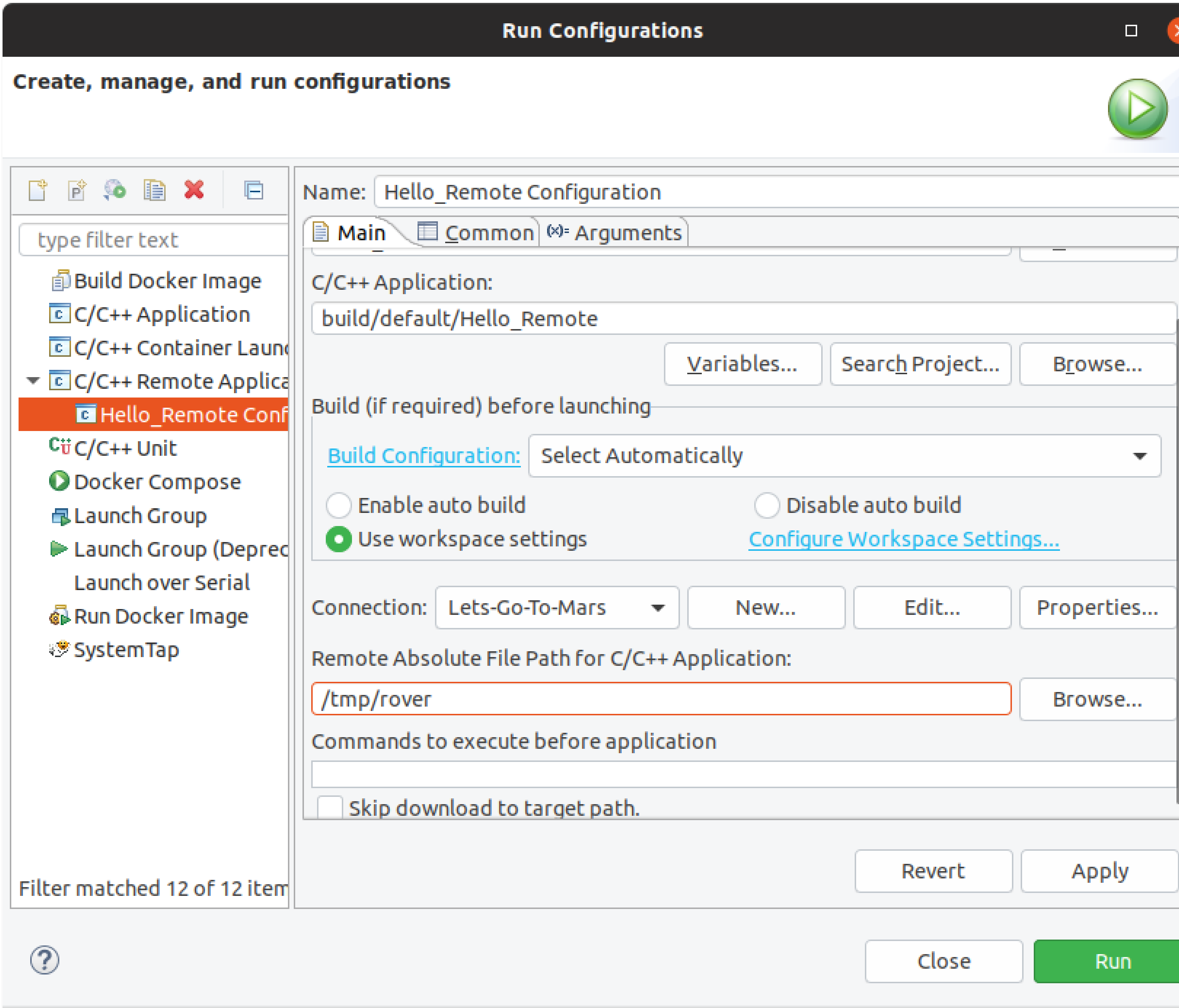
Configure launch perspective settings from the 'Perspectives' preference page.

This will create a run configuration for you project.

8. Set two properties on this view:

   a. Select the connection you created earlier.
   b. Define a location on the remote system where Eclipse will copy your project's binary. Your use-case may dictate where your binary must be copied to, but often a location your user has write permission to, such as `/tmp/`, will suffice.

9. Click Apply to save your configuration.
10. Click Run to run your project on the remote device.



Run Configurations

Create, manage, and run configurations

Name: Hello_Remote Configuration

C/C++ Application: build/default/Hello_Remote

Build (if required) before launching

Build Configuration: Select Automatically

○ Enable auto build ○ Disable auto build

● Use workspace settings Configure Workspace Settings...

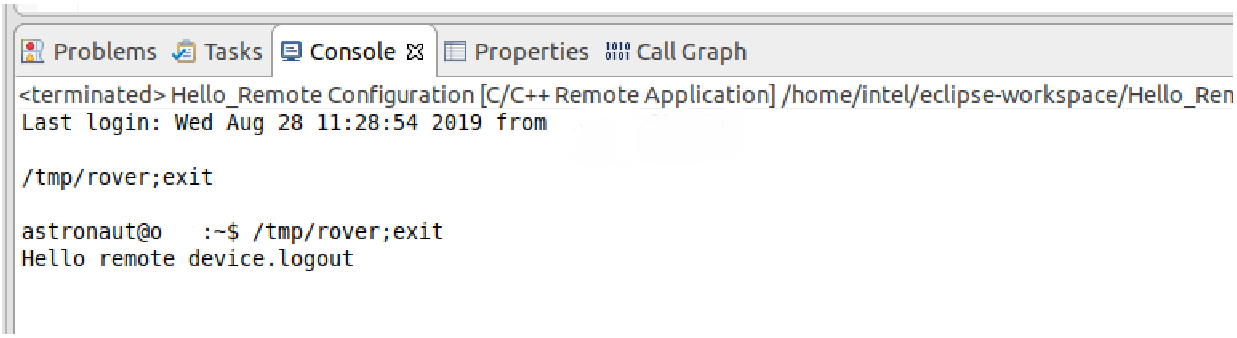
Connection: Lets-Go-To-Mars

Remote Absolute File Path for C/C++ Application: /tmp/rover

Commands to execute before application

□ Skip download to target path.

You should see the output in the console of Eclipse.



```
<terminated> Hello_Remote Configuration [C/C++ Remote Application] /home/intel/eclipse-workspace/Hello_Ren
Last login: Wed Aug 28 11:28:54 2019 from

/tmp/rover;exit

astronaut@o    :~$ /tmp/rover;exit
Hello remote device.logout
```

Note that you can re-use the same connection and launch configuration for debug too. Your remote device will need at least `gdbserver` in the `PATH environment variable`.

Build and Run a Sample Project Using Eclipse*

Intel® oneAPI Base Toolkit

Intel® oneAPI integrates with third-party IDEs on Linux* to provide a seamless GUI experience for software development.

> **NOTE** If you are using Eclipse* with FPGA, see the FPGA Workflows on Third-Party IDEs for Intel® oneAPI Toolkits.

> **NOTE**
> An internet connection is required to download the samples for oneAPI toolkits. For information on how to use this toolkit offline, see Developing with Offline Systems in the Troubleshooting section.

Create a New Project Using Eclipse*

You can access the entire suite using Eclipse* (see the release notes for minimum required version). Once oneAPI is installed, the tools are integrated into the menus.

To watch a video presentation of how to create a project, see Intel® oneAPI Eclipse Samples Plugin.

1. Navigate to the Eclipse directory and source setvars.sh.

   For root or sudo installations:

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

   For normal user installations:

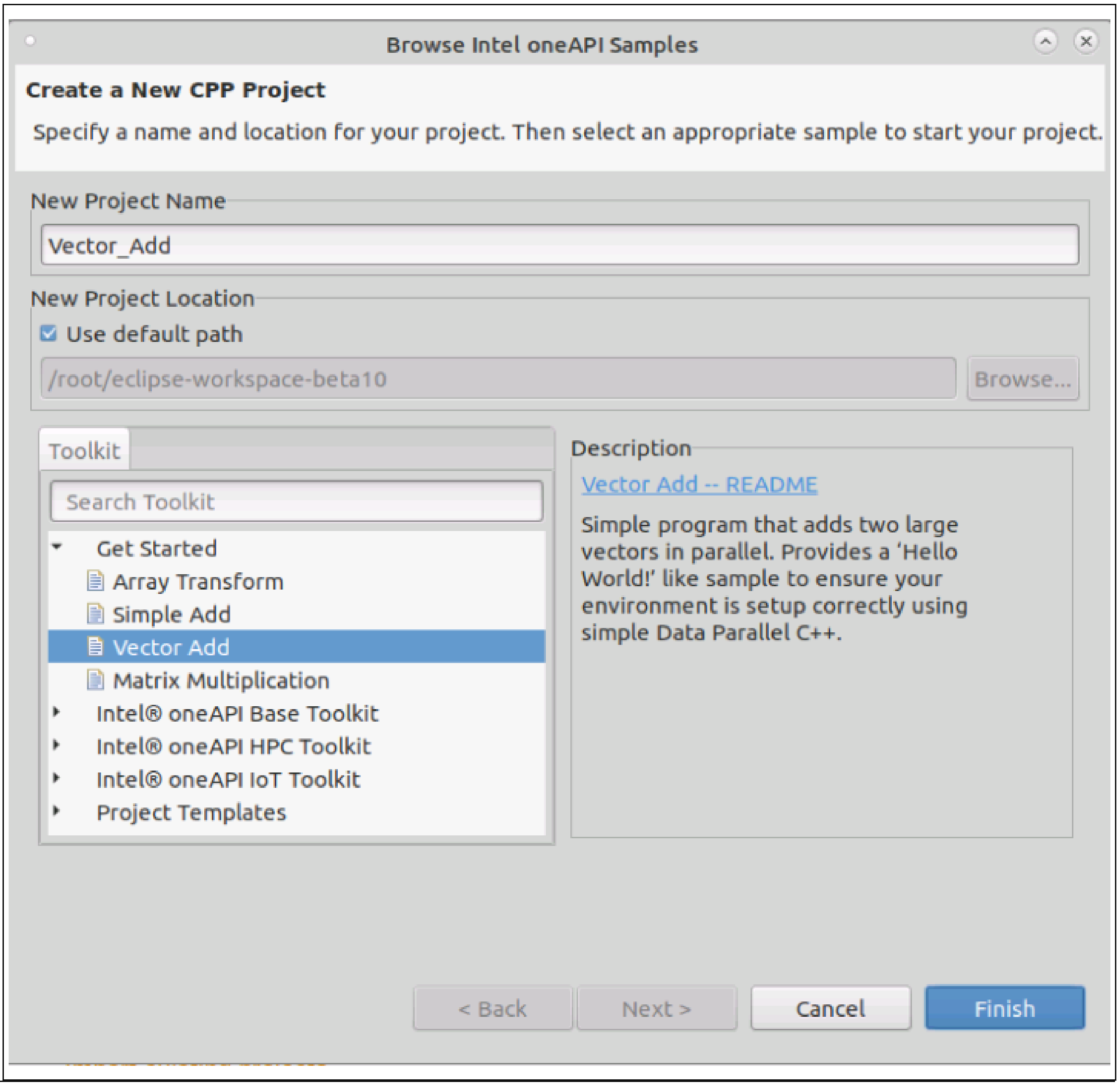
```
. ~/intel/oneapi/setvars.sh
```

   See here for more information on setvars.sh.
2. Launch Eclipse from the folder where it was installed:

```
./eclipse
```

3. In Eclipse, select your workspace and create a project with **Intel** > **Browse Intel oneAPI Samples**. The wizard selection screen appears.

> **NOTE** If you do not see an Intel menu in Eclipse, install the oneAPI Eclipse plugins, then return to this page for instructions on how to Build and Run a sample.



Browse Intel oneAPI Samples

Create a New CPP Project

Specify a name and location for your project. Then select an appropriate sample to start your project.

New Project Name: Vector_Add

New Project Location

☑ Use default path

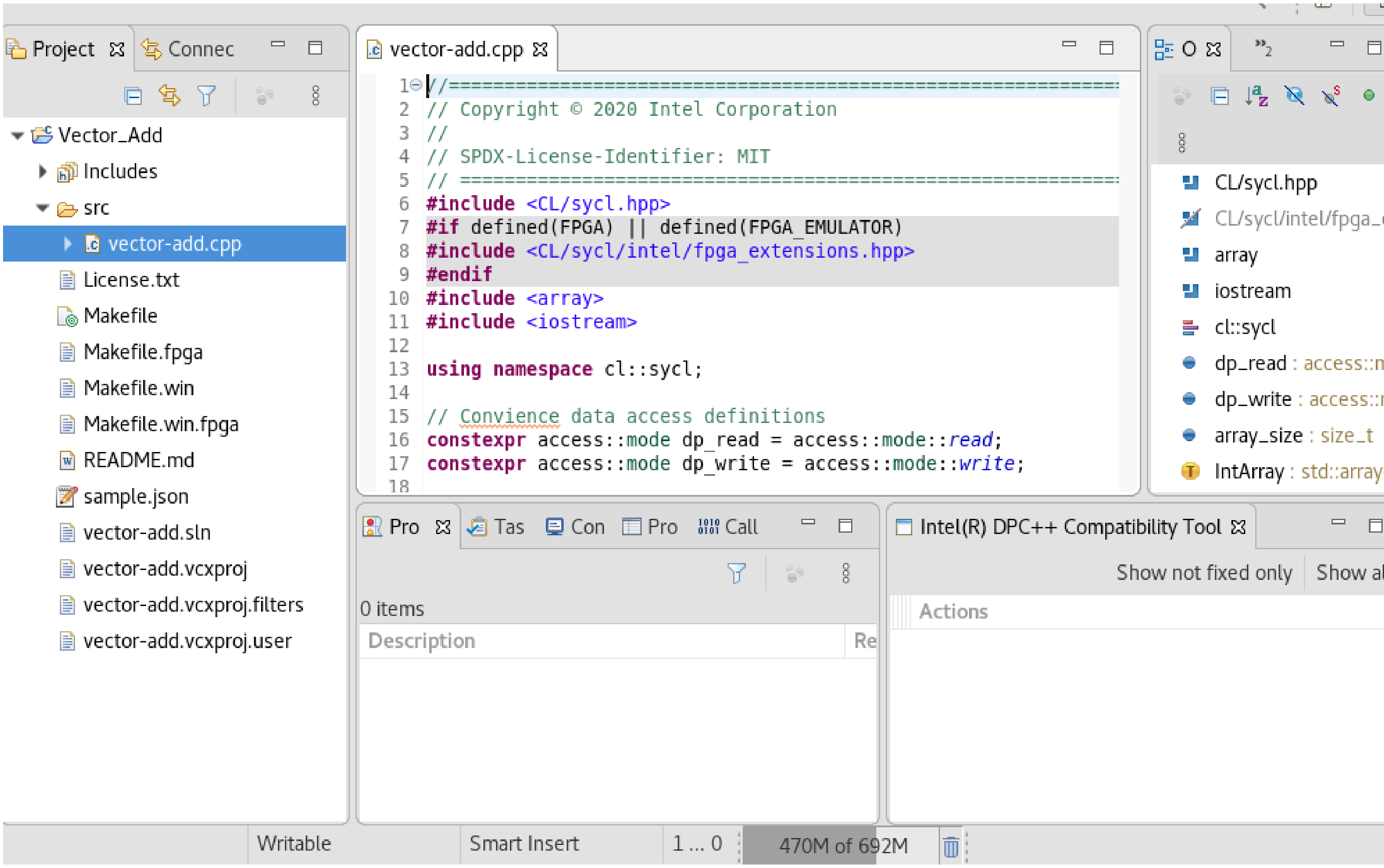
/root/eclipse-workspace-beta10

Description

Vector Add -- README

Simple program that adds two large vectors in parallel. Provides a 'Hello World!' like sample to ensure your environment is setup correctly using simple Data Parallel C++.

4. Enter a name for your project in the New Project Name field. Use the > arrow to select the Intel® oneAPI Base Toolkit.
5. Select the **Intel® oneAPI DPC++/C++ Compiler** > **CPU, GPU and FPGA** group, then select **Vector Add**. Vector Add is a simple test application that will help verify that the tools are setup correctly and can access your system's GPU.
6. The `Vector Add` sample opens in the C++ perspective and the sample will run. Check the **Problems** view at the bottom. If there are any errors, they will display in the **Problems** tab.



```
//==============================================================
// Copyright © 2020 Intel Corporation
//
// SPDX-License-Identifier: MIT
// =============================================================
#include <CL/sycl.hpp>
#if defined(FPGA) || defined(FPGA_EMULATOR)
#include <CL/sycl/intel/fpga_extensions.hpp>
#endif
#include <array>
#include <iostream>

using namespace cl::sycl;

// Convience data access definitions
constexpr access::mode dp_read = access::mode::read;
constexpr access::mode dp_write = access::mode::write;
```

7. To build a project, select **Project** > **Build Project**. When building, the Eclipse Console view (usually in the lower right) will show you a build log, along with any errors or warnings. In this example, no errors were found.
8. To run the application, click **Run** > **Run** .

See Explore SYCL Through Samples to learn more.

Next Steps

After successfully building a sample application, Explore SYCL with Samples from Intel and explore the tools in the Intel® oneAPI Base Toolkit.

Tool	**Description**
Intel® DPC++ Compatibility Tool	The Intel® DPC++ Compatibility Tool assists in migration of CUDA* applications to SYCL* ready code that can use the Intel® oneAPI DPC++/C++ Compiler. Get started.
Intel® oneAPI DPC++/C++ Compiler	The Intel® oneAPI DPC++/C++ Compiler targets CPUs and accelerators through single-source code while permitting custom tuning. Get Started.
Intel® oneAPI DPC++ Library	This library is a companion to the Intel® oneAPI DPC++/C ++ Compiler and provides an alternative for C++ developers who create heterogeneous applications and

	solutions. Its APIs are based on familiar standards-C++ STL, Parallel STL (PSTL), Boost.Compute, and SYCL*-to maximize productivity and performance across CPUs, GPUs, and FPGAs.
Intel® Distribution for GDB*	GDB, the GNU Project debugger, allows you to see what is going on `inside' another program while it executes -- or what another program was doing at the moment it crashed. Get Started. Learn more.
Intel® oneAPI Math Kernel Library (oneMKL)	The oneMKL helps you achieve maximum performance with a math computing library of highly optimized, extensively parallelized routines for CPU and GPU. The library has C and Fortran interfaces for most routines on CPU, and SYCL* interfaces for some routines on both CPU and GPU. Get started.
Intel® oneAPI Threading Building Blocks (oneTBB)	oneTBB is a flexible performance library that simplifies the work of adding parallelism to complex applications, even if you're not a threading expert. Learn more.
Intel® Integrated Performance Primitives	Intel® Integrated Performance Primitives (Intel® IPP) is an extensive library of ready-to-use, domain-specific functions that are highly optimized for diverse Intel® architectures. Get Started with Intel® IPP or Get Started with Intel® IPP Cryptography.
Intel® oneAPI Data Analytics Library	The Intel® oneAPI Data Analytics Library (oneDAL) is a library that helps speed up big data analysis by providing highly optimized algorithmic building blocks for all stages of data analytics (preprocessing, transformation, analysis, modeling, validation, and decision making) in batch, online, and distributed processing modes of computation. The current version of oneDAL provides SYCL* API extensions to the traditional C++ interface. Get started.
Intel® oneAPI Collective Communications Library	Intel® oneAPI Collective Communications Library (oneCCL) is a scalable and high-performance communication library for Deep Learning (DL) and Machine Learning (ML) workloads. It develops the ideas originated in Intel(R) Machine Learning Scaling Library and expands the design and API to encompass new features and use cases. Get started.
Intel® oneAPI Deep Neural Network Library	The Intel® oneAPI Deep Neural Network Library (oneDNN) is an open-source performance library for deep learning applications. The library includes basic building blocks for neural networks optimized for Intel® Architecture Processors and Intel® Processor Graphics. oneDNN is intended for deep learning applications and framework developers interested in improving application performance on Intel CPUs and GPUs. Get started.
Intel® VTune™ Profiler	Intel® VTune™ Profiler is a performance analysis tool targeted for users developing serial and multithreaded applications. The tool is delivered as a Performance Profiler with Intel Performance Snapshots and supports local and remote target analysis on the Windows* and Linux* platforms. Get started.
Intel® Advisor	Intel® Advisor gives software architects and developers the data and analysis tools they need to build well-threaded and vectorized code that exploits modern hardware capabilities. Get started.
Intel® FPGA Add-on for oneAPI Base Toolkit	Use reconfigurable hardware to accelerate data-centric workloads. Learn more.

Learn more about SYCL and targeting other accelerators using the following resources:

Resource	**Description**

Intel® oneAPI Programming Guide	Provides details on the oneAPI programming model, including details about SYCL standards, programming for various target accelerators, and introductions to the oneAPI libraries.
Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel® Arria 10 GX FPGA	A quick start guide for the Intel® Programmable Acceleration Card with Intel® Arria® 10 GX FPGA. This guide provides instructions to load and run a loopback test, Hello FPGA, in both non-virtualized and virtualized environments.
Intel Acceleration Stack Quick Start Guide: Intel® FPGA Programmable Acceleration Card D5005	A quick start guide for Intel® FPGA PAC D5005. This guide provides the instructions for installing the Open Programmable Acceleration Engine (OPAE) on the host Intel® Xeon® Processor to manage and access the Intel® FPGA PAC, configuring and flashing the FPGA image and Board Management Controller (BMC) images, running the example `hello_afu` in a virtualized and non-virtualized environment, and handling graceful thermal shutdown.
Intel® FPGA SDK for OpenCL™ Pro Edition: Custom Platform Toolkit User Guide	Outlines the procedure for creating an Intel® FPGA Software Development Kit (SDK) for OpenCL™ Pro Edition Custom Platform. The Intel® FPGA SDK for OpenCL™ Pro Edition Custom Platform Toolkit provides the necessary tools for implementing a fully functional Custom Platform.
Intel® FPGA Acceleration Hub	Learn about the Acceleration Stack for Intel® Xeon® CPUs with FPGAs and other Intel FPGA-based acceleration platforms.
FPGA Optimization Guide for Intel® oneAPI Toolkits	The oneAPI FPGA Optimization Guide provides guidance on leveraging the functionalities of SYCL to optimize your design.
Explore SYCL Through Intel® FPGA Code Samples	Provides guidance on how to target and develop your design on an FPGA using the oneAPI programming model. It also provides guidance on how to optimize a design to achieve performance and latency targets for an application targeting the FPGA.
oneAPI GPU Optimization Guide	The oneAPI GPU Optimization Guide demonstrates how to improve the behavior of your software by partitioning it across the host and accelerator to specialize portions of the computation that run best on the accelerator. Specialization includes restructuring and tuning the code to create the best mapping of the application to the hardware. The value of oneAPI is that it allows each of these variations to be expressed in a common language with device-specific variants launched on the appropriate accelerator.

For more information about this toolkit, see the Intel® oneAPI Base Toolkit page.

SSH: Password-less Access to Remote Linux* Target

Introduction

For some oneAPI applications, you must configure a password-less SSH connection for the root user on the target system; for example:

- IoT applications that use the MRAA/UPM sensor library
- Any application that interacts with system resources that require su, sudo, or root access
- Any tool that requires remote root or sudo access to your target system

When you finish the configuration steps below you will be able to “ssh into” your remote Linux target from your host development system without a password prompt, as a normal (non-root) user or as a root user.

For an introduction to SSH and how SSH keys work, see SSH Essentials: Working with SSH Servers, Clients, and Keys.

NOTE Password-less access works only when you connect to your target system from your host development system with a matching private SSH key. Attempting to connect from a different host system will still require a password.

Configure Password-less SSH Access

These instructions apply to:

- **Host development system:** Linux*, Windows*, or macOS*
- **Target system:** Linux

Set up an .ssh directory

On your host development system, follow these steps:

Step 1: Open a terminal session (CMD window on Windows) and CD to your home directory
Step 2: Create .ssh directory

Enter the following commands to create an .ssh directory, set the proper permissions, and CD into the new .ssh directory.

At a Windows CMD prompt

```
> %HomeDrive%  &&  cd %HomePath%
> mkdir .ssh
> cd .ssh
```

At a Linux terminal (bash) prompt

```
$ cd ~
$ mkdir -p .ssh
$ chmod 700 .ssh
$ cd .ssh
```

Step 3: Generate keys and copy to the target system

NOTE From this point forward the instructions apply to all host development systems (Windows, Linux, and macOS).

1. To generate a *default-named* RSA key pair with an *empty passphrase* (that is, do not provide a passphrase when asked), enter:

```
$ ssh-keygen -t rsa
```

2. To copy the new public key to your target system's *non-root user* home folder, enter the following, where:

   username = the name used to access the target and *target* = the IP address or the network hostname of the target

   You should be prompted for the non-root user password for your target device.

```
$ scp id_rsa.pub username@target:id_rsa.pub
$ ssh username@target
$ cd ~
$ mkdir -p .ssh
$ chmod 700 .ssh
$ cat ~/id_rsa.pub >>.ssh/authorized_keys
$ chmod 600 .ssh/authorized_keys
$ exit
```

Step 4: Confirm that a password is no longer required (non-root)

Follow this step to confirm that a password is no longer required for your non-root user.

1. To display the target's **system information** strings, including the target's hostname as the second field in the output, enter:

```
ssh username@target uname -a
```

Step 5: Configure password-less access to root on your target

1. To login to the *non-root user* on the target using SSH and switch to the *root user* using sudo, enter:

```
$ ssh username@target
$ cd ~
$ sudo -E bash
```

   Note that the sudo command should prompt you for your target system's *non-root user* password.
2. To copy the public key that you transferred to the non-root user account on the target into the root user's authorized keys file, enter:

```
$ mkdir -p /root/.ssh
$ chmod 700 /root/.ssh
$ cat ./id_rsa.pub >>/root/.ssh/authorized_keys
$ chmod 600 /root/.ssh/authorized_keys
```

3. Exit twice, first from the sudo bash session, second from the ssh connection:

```
$ exit
$ exit
```

Step 6: [Optional] Check your progress

To test the root connection for your target, enter:

```
$ ssh root@target ls -a
```

You should see a directory listing of all files located in the /root folder on your target, without the need for a login prompt.

Next: Create a New Connection and Connect to Your Target

Notes

- Password-less access works only when you connect to your target system from your host development system with a matching private SSH key. Attempting to connect from a different host system will still require a password.
- Make sure that you have created a project for Linux targets, and that this project is selected in the Project Explorer.

See Also

Why sudo user can use sched_setscheduler SCHED_RR while root can not?

Operation not permitted while setting new priority for thread

Debugging libmraa {#debugging}

SSH: Running Applications Built with Eclipse*

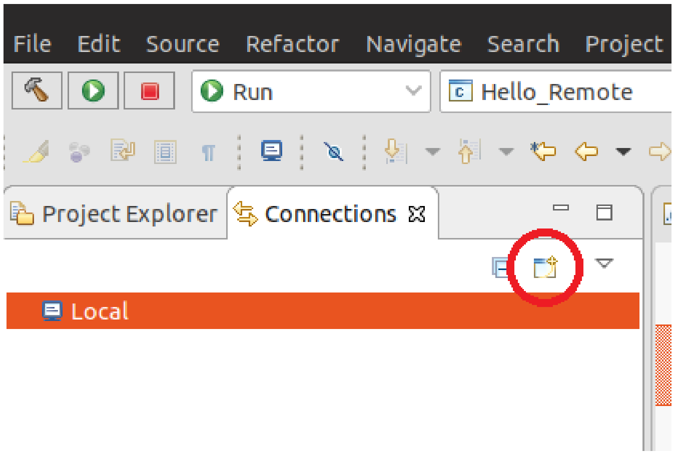
The following describes how to run a built application on a "remote target" system using an SSH connection.

If your application requires a password-less SSH connection, see Password-less Access to a Remote Linux* Target Device.

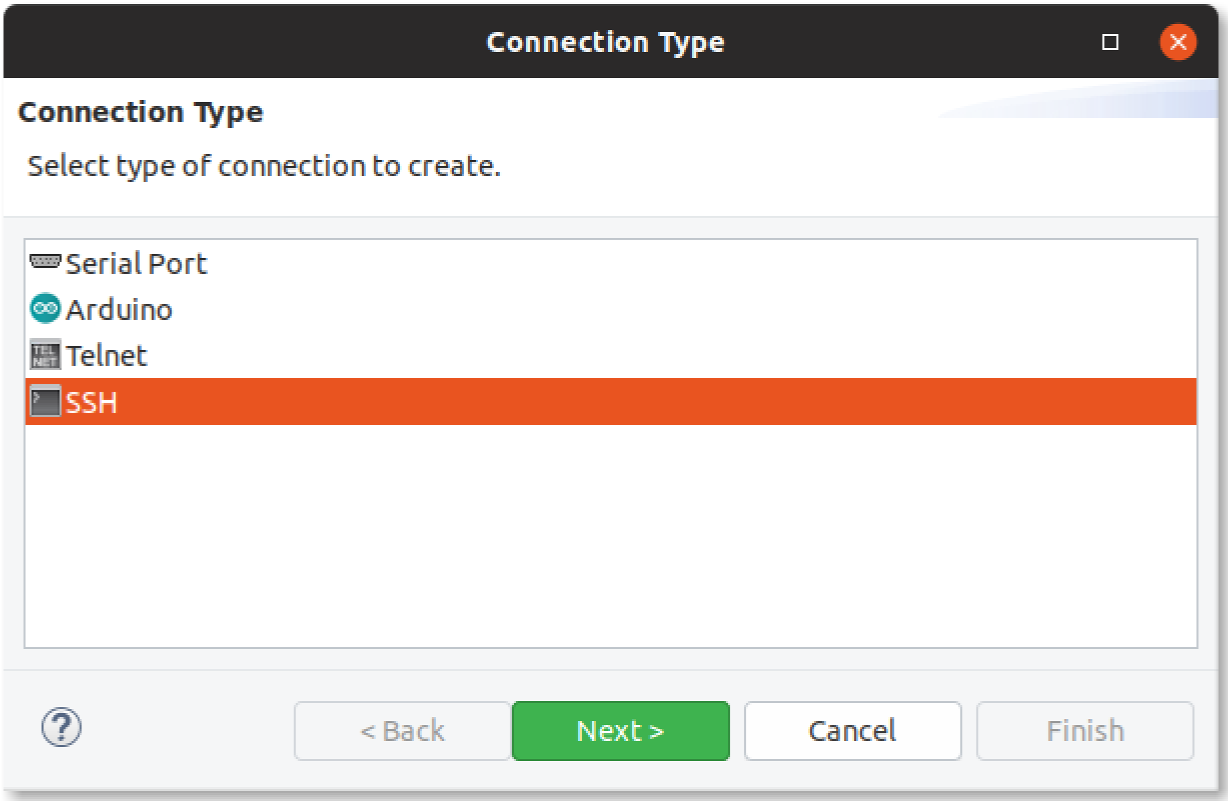
This section assumes:

- you already have an Eclipse project
- have successfully built an application from that project
- have some familiarity with Eclipse
- remote target is already configured for remote access via SSH

1. Create a Connection within Eclipse by selecting the **New Connection** option:



2. Select the SSH Connection type and click **Next**:

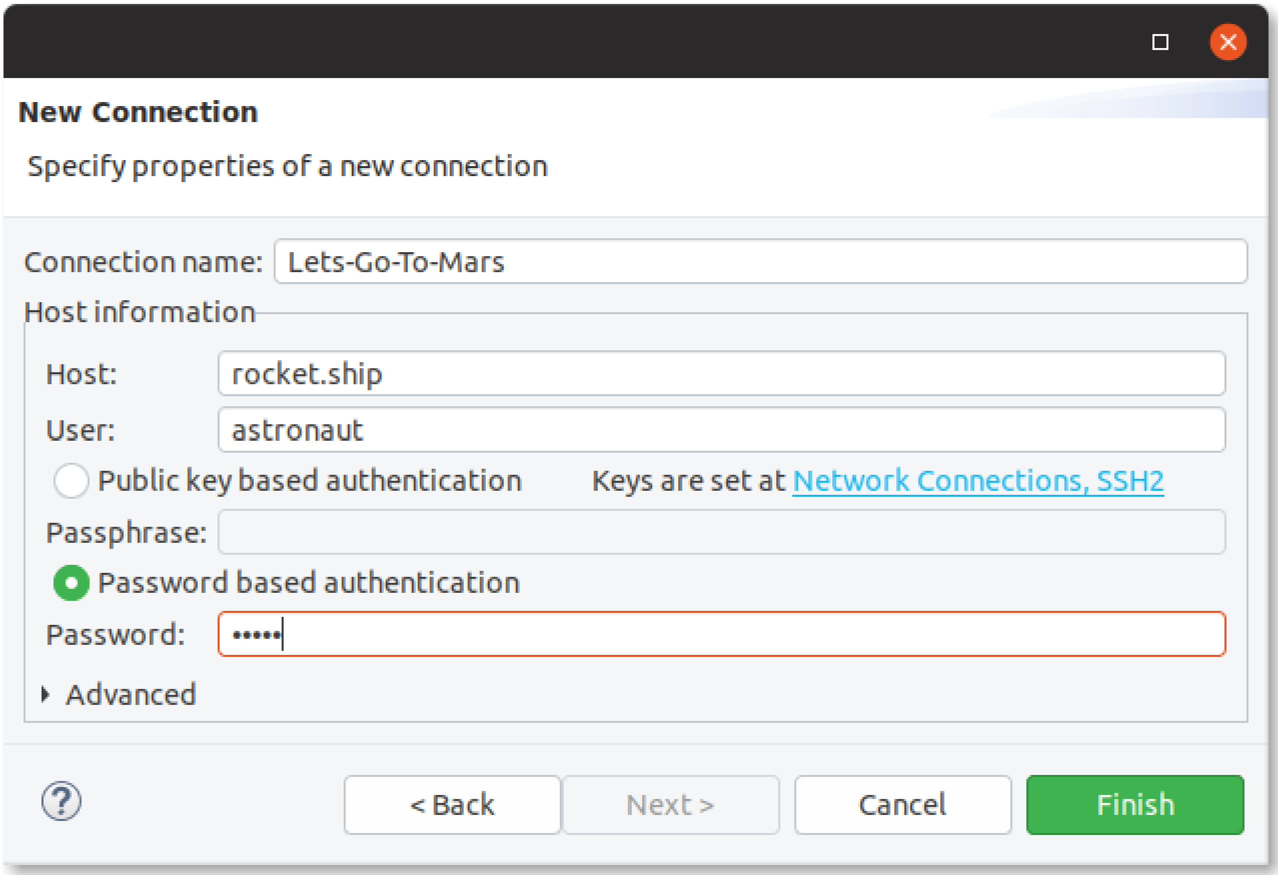


Connection Type

Connection Type

Select type of connection to create.

3. Enter your remote targets Hostname and Username. You can either use Key-based authentication or Password, depending on the configuration of the remote device.



New Connection

Specify properties of a new connection

Connection name: Lets-Go-To-Mars

Host information

Host: rocket.ship

User: astronaut

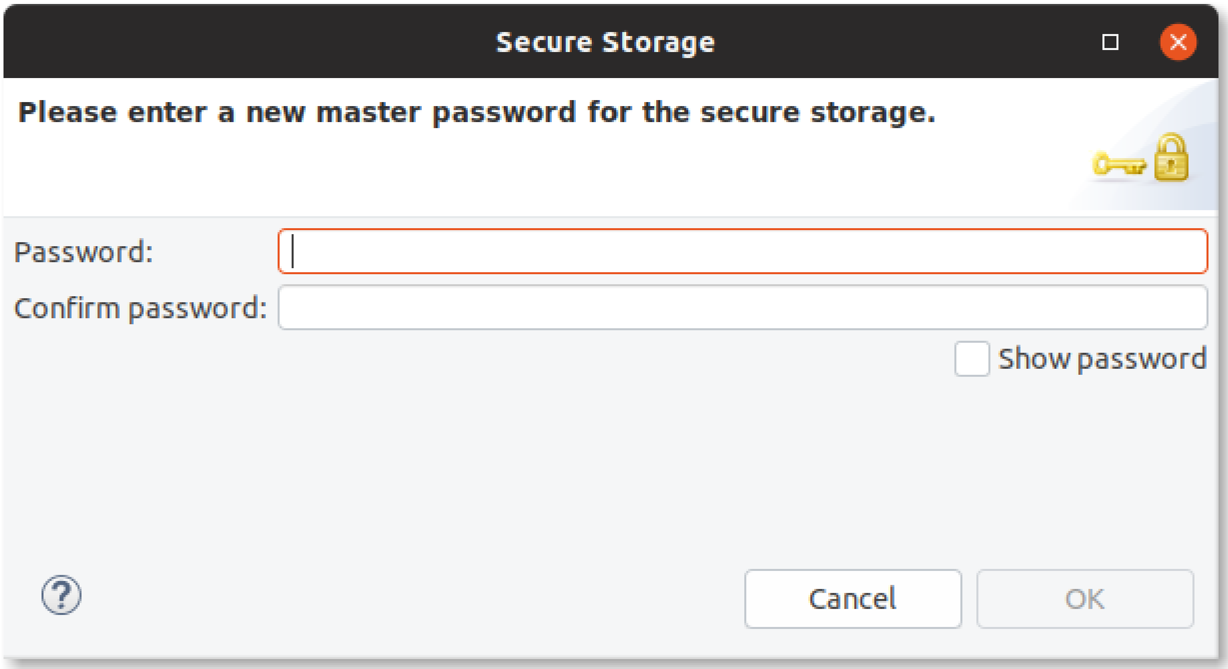
○ Public key based authentication — Keys are set at Network Connections, SSH2

Passphrase:

● Password based authentication

Password: •••••

4. You may also be then prompted to set an Eclipse Secure Storage password. This protects the password you entered when stored on the disk. Enter the password and click OK.



Secure Storage

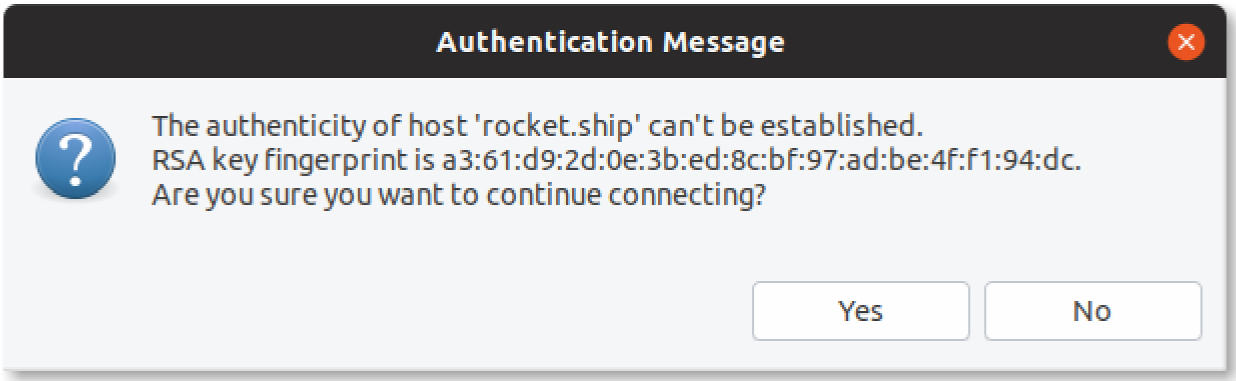
Please enter a new master password for the secure storage.

Password:

Confirm password:

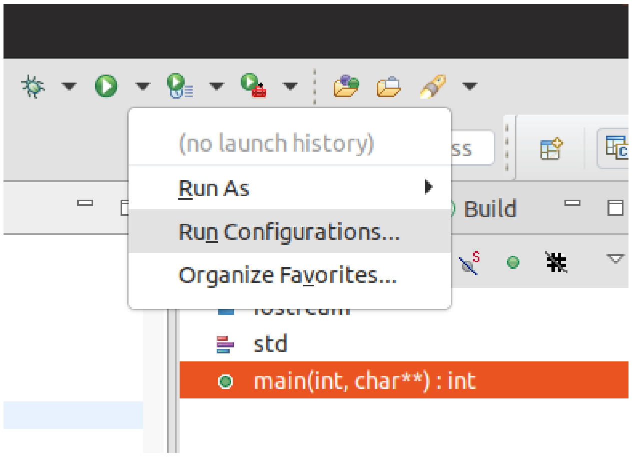
□ Show password

5. When connecting to the device for the first time, you should be prompted to trust the device. Click Yes to continue.

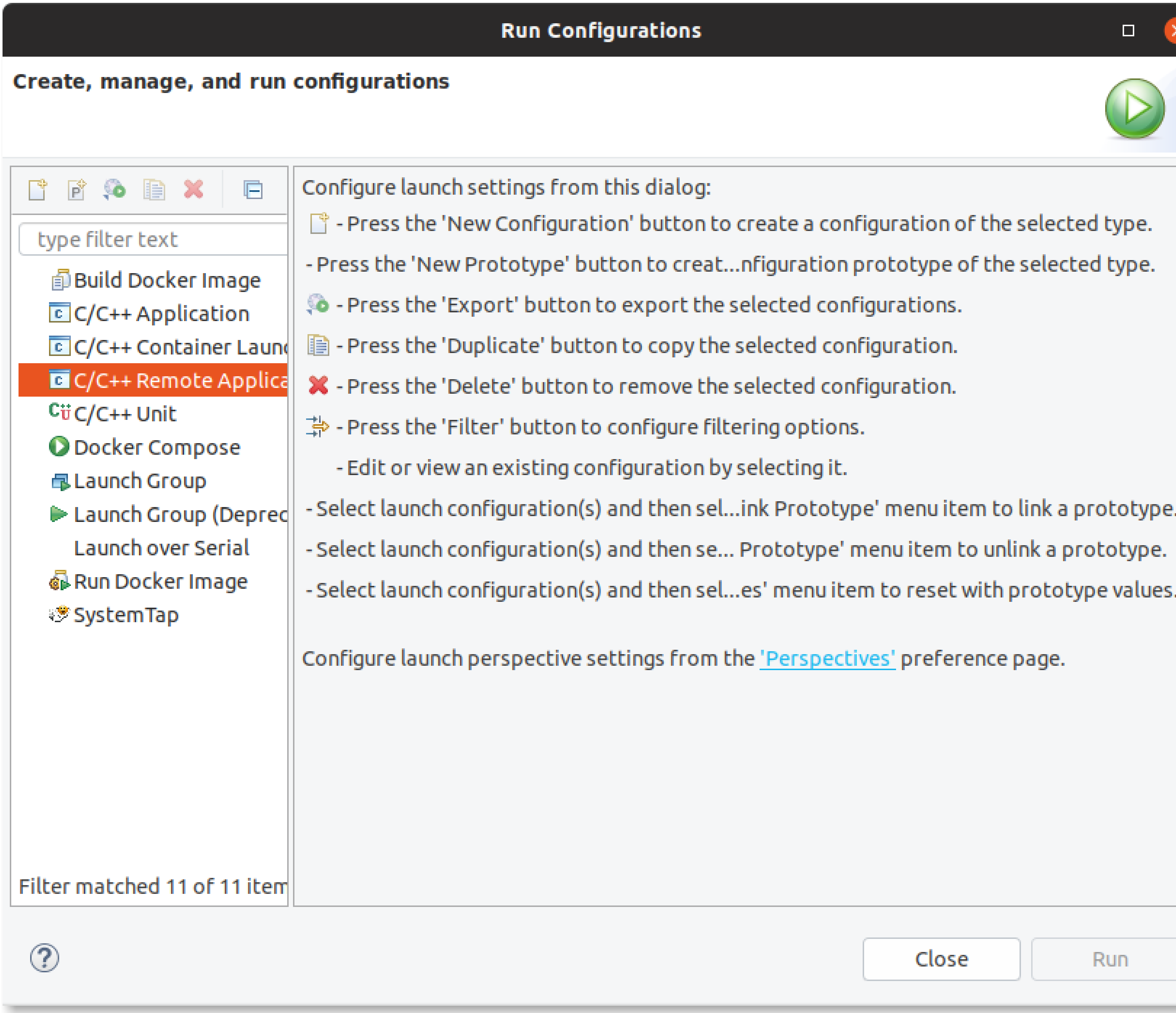

Authentication Message

The authenticity of host 'rocket.ship' can't be established.
RSA key fingerprint is a3:61:d9:2d:0e:3b:ed:8c:bf:97:ad:be:4f:f1:94:dc.
Are you sure you want to continue connecting?

6. Once a connection has been defined within Eclipse, open the the Run Configurations window to create your configuration.



7. Double-click on the "C/C++ Remote Application" configuration type.



Run Configurations

Create, manage, and run configurations

Configure launch settings from this dialog:

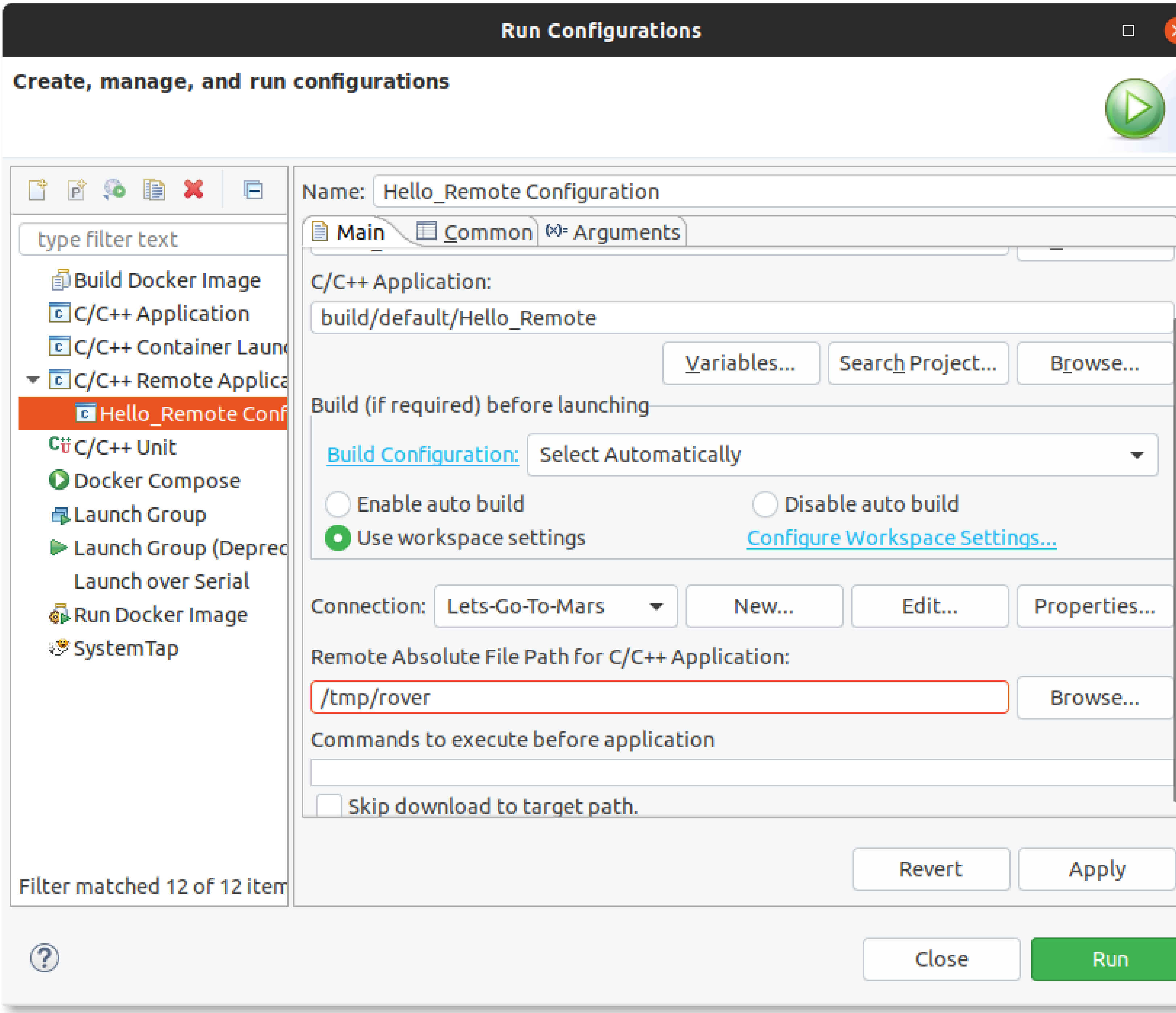
- Press the 'New Configuration' button to create a configuration of the selected type.
- Press the 'New Prototype' button to creat...nfiguration prototype of the selected type.
- Press the 'Export' button to export the selected configurations.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.
- Select launch configuration(s) and then sel...ink Prototype' menu item to link a prototype.
- Select launch configuration(s) and then se... Prototype' menu item to unlink a prototype.
- Select launch configuration(s) and then sel...es' menu item to reset with prototype values.

Configure launch perspective settings from the 'Perspectives' preference page.

This will create a run configuration for you project.

8. Set two properties on this view:

   a. Select the connection you created earlier.
   b. Define a location on the remote system where Eclipse will copy your project's binary. Your use-case may dictate where your binary must be copied to, but often a location your user has write permission to, such as `/tmp/`, will suffice.
9. Click Apply to save your configuration.
10. Click Run to run your project on the remote device.



Name: Hello_Remote Configuration

C/C++ Application:
build/default/Hello_Remote

Build (if required) before launching

Build Configuration: Select Automatically

○ Enable auto build
○ Disable auto build
● Use workspace settings
Configure Workspace Settings...

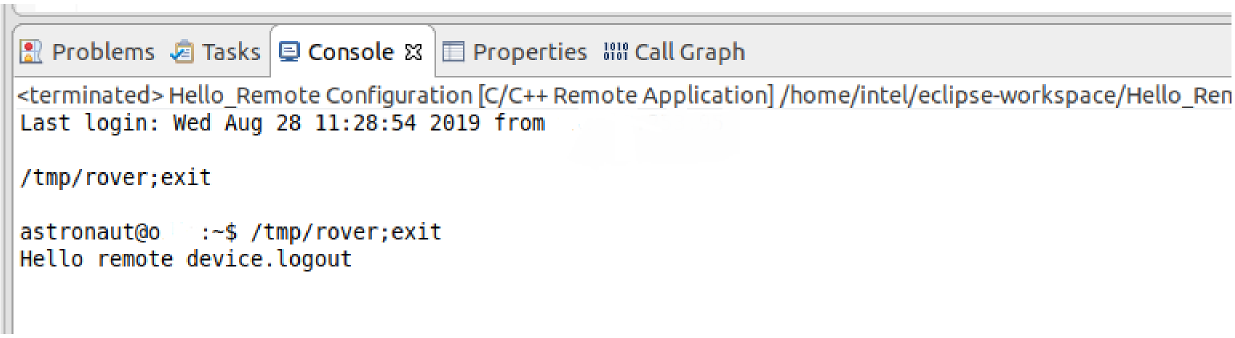
Connection: Lets-Go-To-Mars

Remote Absolute File Path for C/C++ Application:
/tmp/rover

Commands to execute before application

□ Skip download to target path.

You should see the output in the console of Eclipse.



```
<terminated> Hello_Remote Configuration [C/C++ Remote Application] /home/intel/eclipse-workspace/Hello_Ren
Last login: Wed Aug 28 11:28:54 2019 from

/tmp/rover;exit

astronaut@o    :~$ /tmp/rover;exit
Hello remote device.logout
```

Note that you can re-use the same connection and launch configuration for debug too. Your remote device will need at least `gdbserver` in the `PATH environment variable`.

Build and Run a Sample Project Using Visual Studio Code

Intel® oneAPI Base Toolkit

Intel® oneAPI toolkits integrate with third-party IDEs to provide a seamless GUI experience for software development.

> **NOTE** If you are using Visual Studio Code (VS Code) with FPGA, see the FPGA Workflows on Third-Party IDEs for Intel® oneAPI Toolkits.

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

To watch a video presentation of how to install extensions and use them to set up your environment, explore sample code, and connect to the Intel® Developer Cloud using Visual Studio Code, see oneAPI Visual Studio Code Extensions.

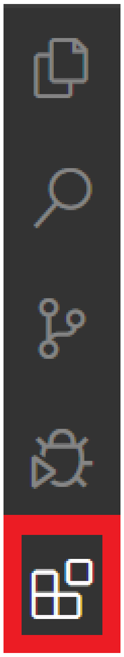
This procedure requires the Sample Browser extension to be installed. The next section will describe how to install it. If you have already installed it, skip to Create a Project Using Visual Studio Code.

Extensions for Visual Studio Code Users

To watch a video presentation of how to install extensions and use them to set up your environment, explore sample code, and connect to the Intel® Developer Cloud using Visual Studio Code, see oneAPI Visual Studio Code Extensions.

You can use VS Code extensions to set your environment, create launch configurations, and browse and download samples:

1. From Visual Studio Code, click on the Extensions logo in the left navigation.

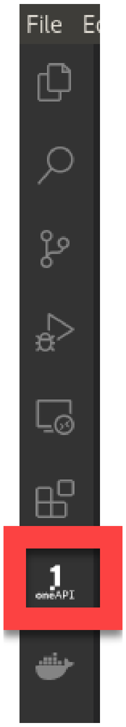


2. Locate the extension titled **Extension Pack for Intel® oneAPI Toolkits**, or visit https://marketplace.visualstudio.com/publishers/intel-corporation to browse available extensions.
3. Click **Install**.

For more information about VS Code extensions for Intel oneAPI Toolkits, see Using Visual Studio Code* to Develop Intel® oneAPI Applications.

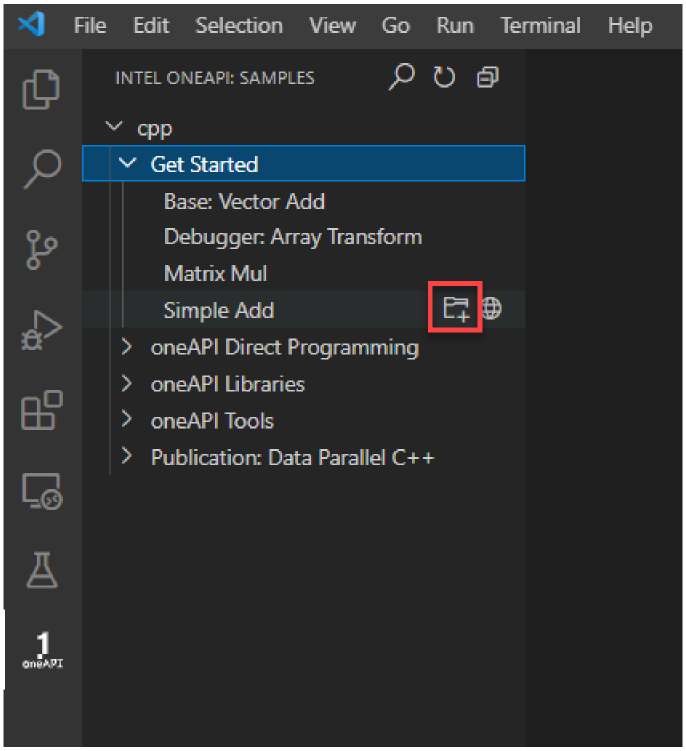
Create a Project Using Visual Studio Code

1. Click on the icon to open the oneAPI Samples Browser:

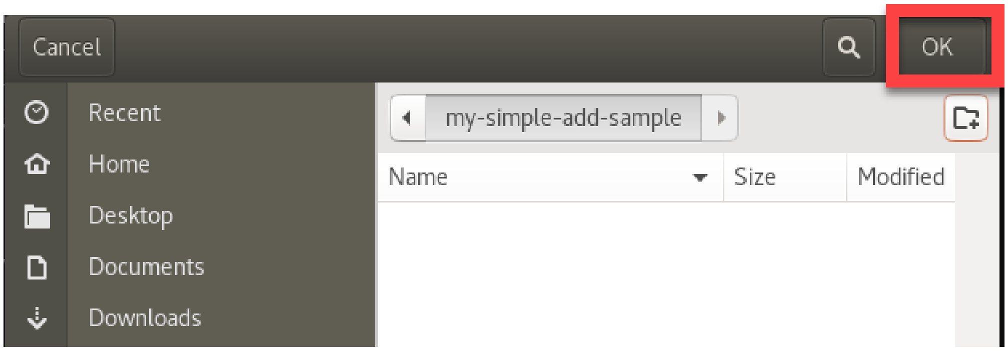


2. A list of available samples will open in the left navigation.

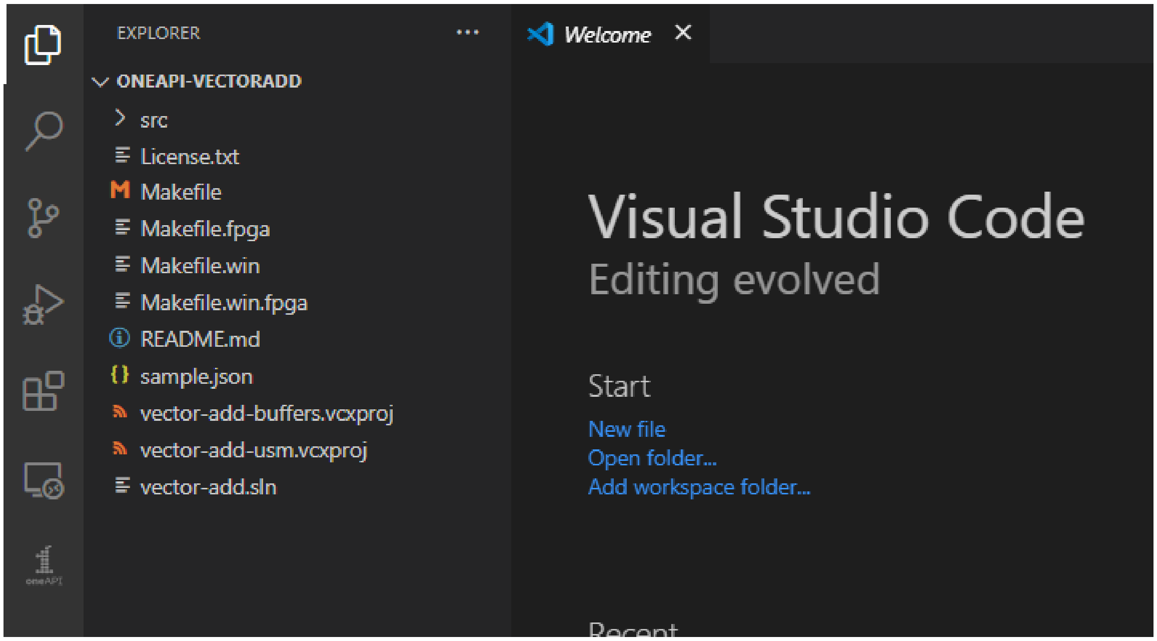
3. To view the readme for the sample, click the 🌐 next to the sample. If you choose to build and run the sample, the readme will also be downloaded with the sample.
4. Find the sample you want to build and run. Click the 📁 to the right of the sample name.



5. Create a new folder for your sample and click OK.

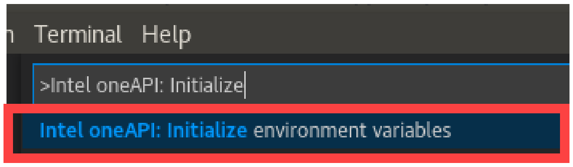


6. The sample will load in a new window:



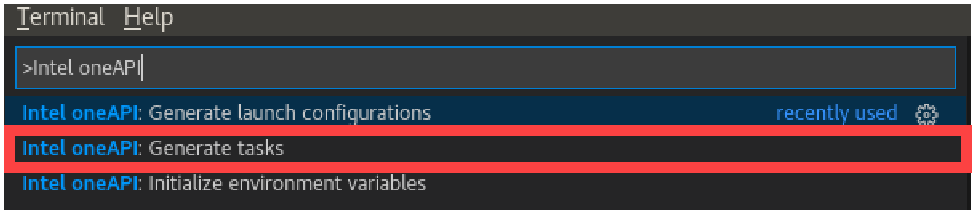
Set the oneAPI Environment

1. Press `Ctrl+Shift+P` (or **View -> Command Palette...**) to open the Command Palette.
2. Type **Intel oneAPI: Initialize environment variables**. Click on **Intel oneAPI: Initialize environment variables**.

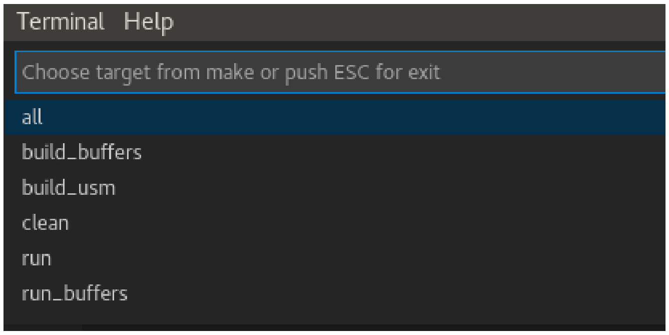

Terminal Help
>Intel oneAPI: Initialize
Intel oneAPI: Initialize environment variables

Prepare Build Tasks from Make / CMake Files

1. Press `Ctrl+Shift+P` or **View -> Command Palette...** to open the Command Pallette.
2. Type **Intel oneAPI** and select **Intel oneAPI: Generate tasks**.


Terminal Help
>Intel oneAPI
Intel oneAPI: Generate launch configurations
recently used
Intel oneAPI: Generate tasks
Intel oneAPI: Initialize environment variables

3. Select the build tasks (target) from your Make/CMake oneAPI project that you want to use.



4. Run the task/target by selecting **Terminal -> Run task...** .
5. Select the task to run.

> **NOTE** Not all oneAPI sample projects use CMake. The `README.md` file for each sample specifies how to build the sample. We recommend that you check out the CMake extension for VS Code that is maintained by Microsoft.

Build the Project

The oneAPI extensions enable the ability to prepare launch configurations for running and debugging projects created using Intel oneAPI toolkits:

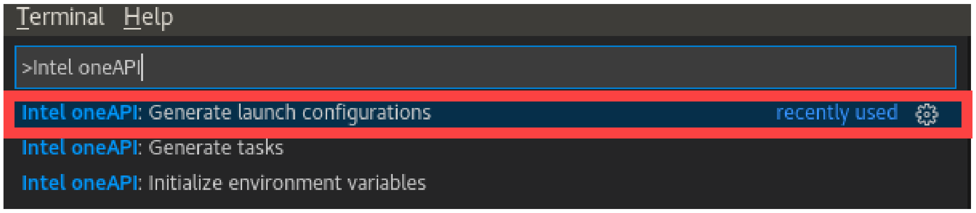
> **NOTE** Debugging a local Windows host using VS Code is not supported. Debugging a local Linux host on CPU is supported using VS Code, but debugging on GPU requires a remote host. Debugging a remote Linux target from Windows or Linux host is supported using VS Code.

1. Press `Ctrl+Shift+B` or **Terminal -> Run Build Task...** to set the default build task.
2. Select the task from the command prompt list to build your project.
3. Press `Ctrl+Shift+B` or **Terminal -> Run Build Task...** again to build your project.

Prepare Launch Configuration for Debugging

The oneAPI extensions enable the ability to prepare launch configurations for running and debugging projects created using Intel oneAPI toolkits:

1. Press `Ctrl+Shift+P` or **View -> Command Palette...** to open the Command Palette.
2. Type **Intel oneAPI** and select **Intel oneAPI: Generate launch configurations** .


Terminal
Help
>Intel oneAPI
Intel oneAPI: Generate launch configurations
recently used
Intel oneAPI: Generate tasks
Intel oneAPI: Initialize environment variables

3. Select the executable (target) you want to debug.

   Optional: select any task you want to run before and/or after launching the debugger (for example, build the project before debug, clean the project after debug).
4. The configuration is now available to debug and run using the gdb-oneapi debugger. You can find it in `.vscode/launch.json`. To debug and run, click on the Run icon or press `Ctrl+Shift+D`.

NOTE Debugging a local Windows host using VS Code is not supported. Debugging a local Linux host on CPU is supported using VS Code, but debugging on GPU requires a remote host. Debugging a remote Linux target from Windows or Linux host is supported using VS Code.

Debug, Analyze, Develop with More Extensions

There are more oneAPI extensions for Visual Studio Code which enable:

- debugging
- remote development
- connection to Intel Developer Cloud
- analysis configuration

To learn more about the extensions, see Intel® oneAPI Extensions for Visual Studio Code*.

To learn more about more capabilities and options, see Using Visual Studio Code with Intel® oneAPI Toolkits.

NOTE
An internet connection is required to download the samples for oneAPI toolkits. If you are using an offline system, download the samples from a system that is internet connected and transfer the sample files to your offline system. If you are using an IDE for development, you will not be able to use the oneAPI CLI Samples Browser while you are offline. Instead, download the samples and extract them to a directory. Then open the sample with your IDE. The samples can be downloaded from here: Intel® oneAPI Toolkit Code Samples

See Explore SYCL* Through Samples to learn more.

Next Steps

After successfully building a sample application, Explore SYCL with Samples from Intel and explore the tools in the Intel® oneAPI Base Toolkit.

Tool	Description
Intel® DPC++ Compatibility Tool	The Intel® DPC++ Compatibility Tool assists in migration of CUDA* applications to SYCL* ready code that can use the Intel® oneAPI DPC++/C++ Compiler. Get started.
Intel® oneAPI DPC++/C++ Compiler	The Intel® oneAPI DPC++/C++ Compiler targets CPUs and accelerators through single-source code while permitting custom tuning. Get Started.
Intel® oneAPI DPC++ Library	This library is a companion to the Intel® oneAPI DPC++/C++ Compiler and provides an alternative for C++ developers who create heterogeneous applications and solutions. Its APIs are based on familiar standards-C++ STL, Parallel STL (PSTL), Boost.Compute, and SYCL*-to maximize productivity and performance across CPUs, GPUs, and FPGAs.
Intel® Distribution for GDB*	GDB, the GNU Project debugger, allows you to see what is going on `inside' another program while it executes -- or what another program was doing at the moment it crashed. Get Started. Learn more.

Intel® oneAPI Math Kernel Library (oneMKL)	The oneMKL helps you achieve maximum performance with a math computing library of highly optimized, extensively parallelized routines for CPU and GPU. The library has C and Fortran interfaces for most routines on CPU, and SYCL* interfaces for some routines on both CPU and GPU. Get started.
Intel® oneAPI Threading Building Blocks (oneTBB)	oneTBB is a flexible performance library that simplifies the work of adding parallelism to complex applications, even if you're not a threading expert. Learn more.
Intel® Integrated Performance Primitives	Intel® Integrated Performance Primitives (Intel® IPP) is an extensive library of ready-to-use, domain-specific functions that are highly optimized for diverse Intel® architectures. Get Started with Intel® IPP or Get Started with Intel® IPP Cryptography.
Intel® oneAPI Data Analytics Library	The Intel® oneAPI Data Analytics Library (oneDAL) is a library that helps speed up big data analysis by providing highly optimized algorithmic building blocks for all stages of data analytics (preprocessing, transformation, analysis, modeling, validation, and decision making) in batch, online, and distributed processing modes of computation. The current version of oneDAL provides SYCL* API extensions to the traditional C++ interface. Get started.
Intel® oneAPI Collective Communications Library	Intel® oneAPI Collective Communications Library (oneCCL) is a scalable and high-performance communication library for Deep Learning (DL) and Machine Learning (ML) workloads. It develops the ideas originated in Intel(R) Machine Learning Scaling Library and expands the design and API to encompass new features and use cases. Get started.
Intel® oneAPI Deep Neural Network Library	The Intel® oneAPI Deep Neural Network Library (oneDNN) is an open-source performance library for deep learning applications. The library includes basic building blocks for neural networks optimized for Intel® Architecture Processors and Intel® Processor Graphics. oneDNN is intended for deep learning applications and framework developers interested in improving application performance on Intel CPUs and GPUs. Get started.
Intel® VTune™ Profiler	Intel® VTune™ Profiler is a performance analysis tool targeted for users developing serial and multithreaded applications. The tool is delivered as a Performance Profiler with Intel Performance Snapshots and supports local and remote target analysis on the Windows* and Linux* platforms. Get started.
Intel® Advisor	Intel® Advisor gives software architects and developers the data and analysis tools they need to build well-threaded and vectorized code that exploits modern hardware capabilities. Get started.
Intel® FPGA Add-on for oneAPI Base Toolkit	Use reconfigurable hardware to accelerate data-centric workloads. Learn more.

Learn more about SYCL and targeting other accelerators using the following resources:

Resource	**Description**
Intel® oneAPI Programming Guide	Provides details on the oneAPI programming model, including details about SYCL standards, programming for various target accelerators, and introductions to the oneAPI libraries.
Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel® Arria 10 GX FPGA	A quick start guide for the Intel® Programmable Acceleration Card with Intel® Arria® 10 GX FPGA. This guide provides instructions to load and run a loopback test, Hello FPGA, in both non-virtualized and virtualized environments.

Intel Acceleration Stack Quick Start Guide: Intel® FPGA Programmable Acceleration Card D5005	A quick start guide for Intel® FPGA PAC D5005. This guide provides the instructions for installing the Open Programmable Acceleration Engine (OPAE) on the host Intel® Xeon® Processor to manage and access the Intel® FPGA PAC, configuring and flashing the FPGA image and Board Management Controller (BMC) images, running the example `hello_afu` in a virtualized and non-virtualized environment, and handling graceful thermal shutdown.
Intel® FPGA SDK for OpenCL™ Pro Edition: Custom Platform Toolkit User Guide	Outlines the procedure for creating an Intel® FPGA Software Development Kit (SDK) for OpenCL™ Pro Edition Custom Platform. The Intel® FPGA SDK for OpenCL™ Pro Edition Custom Platform Toolkit provides the necessary tools for implementing a fully functional Custom Platform.
Intel® FPGA Acceleration Hub	Learn about the Acceleration Stack for Intel® Xeon® CPUs with FPGAs and other Intel FPGA-based acceleration platforms.
FPGA Optimization Guide for Intel® oneAPI Toolkits	The oneAPI FPGA Optimization Guide provides guidance on leveraging the functionalities of SYCL to optimize your design.
Explore SYCL Through Intel® FPGA Code Samples	Provides guidance on how to target and develop your design on an FPGA using the oneAPI programming model. It also provides guidance on how to optimize a design to achieve performance and latency targets for an application targeting the FPGA.
oneAPI GPU Optimization Guide	The oneAPI GPU Optimization Guide demonstrates how to improve the behavior of your software by partitioning it across the host and accelerator to specialize portions of the computation that run best on the accelerator. Specialization includes restructuring and tuning the code to create the best mapping of the application to the hardware. The value of oneAPI is that it allows each of these variations to be expressed in a common language with device-specific variants launched on the appropriate accelerator.

For more information about this toolkit, see the Intel® oneAPI Base Toolkit page.

SSH: Password-less Access to Remote Linux* Target

Introduction

For some oneAPI applications, you must configure a password-less SSH connection for the root user on the target system; for example:

- IoT applications that use the MRAA/UPM sensor library
- Any application that interacts with system resources that require su, sudo, or root access
- Any tool that requires remote root or sudo access to your target system

When you finish the configuration steps below you will be able to “ssh into” your remote Linux target from your host development system without a password prompt, as a normal (non-root) user or as a root user.

For an introduction to SSH and how SSH keys work, see SSH Essentials: Working with SSH Servers, Clients, and Keys.

NOTE Password-less access works only when you connect to your target system from your host development system with a matching private SSH key. Attempting to connect from a different host system will still require a password.

Configure Password-less SSH Access

These instructions apply to:

- **Host development system:** Linux*, Windows*, or macOS*
- **Target system:** Linux

Set up an .ssh directory

On your host development system, follow these steps:

Step 1: Open a terminal session (CMD window on Windows) and CD to your home directory
Step 2: Create .ssh directory

Enter the following commands to create an .ssh directory, set the proper permissions, and CD into the new .ssh directory.

At a Windows CMD prompt

```
> %HomeDrive%  &&  cd %HomePath%
> mkdir .ssh
> cd .ssh
```

At a Linux terminal (bash) prompt

```
$ cd ~
$ mkdir -p .ssh
$ chmod 700 .ssh
$ cd .ssh
```

Step 3: Generate keys and copy to the target system

NOTE From this point forward the instructions apply to all host development systems (Windows, Linux, and macOS).

1. To generate a *default-named* RSA key pair with an *empty passphrase* (that is, do not provide a passphrase when asked), enter:

```
$ ssh-keygen -t rsa
```

2. To copy the new public key to your target system's *non-root user* home folder, enter the following, where:

   username = the name used to access the target and *target* = the IP address or the network hostname of the target

   You should be prompted for the non-root user password for your target device.

```
$ scp id_rsa.pub username@target:id_rsa.pub
$ ssh username@target
$ cd ~
$ mkdir -p .ssh
$ chmod 700 .ssh
$ cat ~/id_rsa.pub >>.ssh/authorized_keys
$ chmod 600 .ssh/authorized_keys
$ exit
```

Step 4: Confirm that a password is no longer required (non-root)

Follow this step to confirm that a password is no longer required for your non-root user.

1. To display the target's **system information** strings, including the target's hostname as the second field in the output, enter:

```
ssh username@target uname -a
```

Step 5: Configure password-less access to root on your target

1. To login to the *non-root user* on the target using SSH and switch to the *root user* using sudo, enter:

```
$ ssh username@target
$ cd ~
$ sudo -E bash
```

   Note that the sudo command should prompt you for your target system's *non-root user* password.

2. To copy the public key that you transferred to the non-root user account on the target into the root user's authorized keys file, enter:

```
$ mkdir -p /root/.ssh
$ chmod 700 /root/.ssh
$ cat ./id_rsa.pub >>/root/.ssh/authorized_keys
$ chmod 600 /root/.ssh/authorized_keys
```

3. Exit twice, first from the sudo bash session, second from the ssh connection:

```
$ exit
$ exit
```

Step 6: [Optional] Check your progress

To test the root connection for your target, enter:

```
$ ssh root@target ls -a
```

You should see a directory listing of all files located in the /root folder on your target, without the need for a login prompt.

Next: Create a New Connection and Connect to Your Target

Notes

- Password-less access works only when you connect to your target system from your host development system with a matching private SSH key. Attempting to connect from a different host system will still require a password.
- Make sure that you have created a project for Linux targets, and that this project is selected in the Project Explorer.

See Also

Why sudo user can use sched_setscheduler SCHED_RR while root can not?

Operation not permitted while setting new priority for thread

Debugging libmraa {#debugging}

SSH: Running Applications Built with Eclipse*

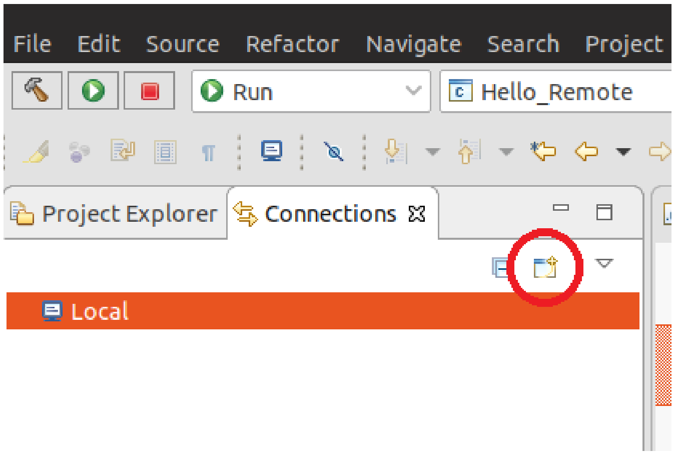
The following describes how to run a built application on a "remote target" system using an SSH connection.

If your application requires a password-less SSH connection, see Password-less Access to a Remote Linux* Target Device.

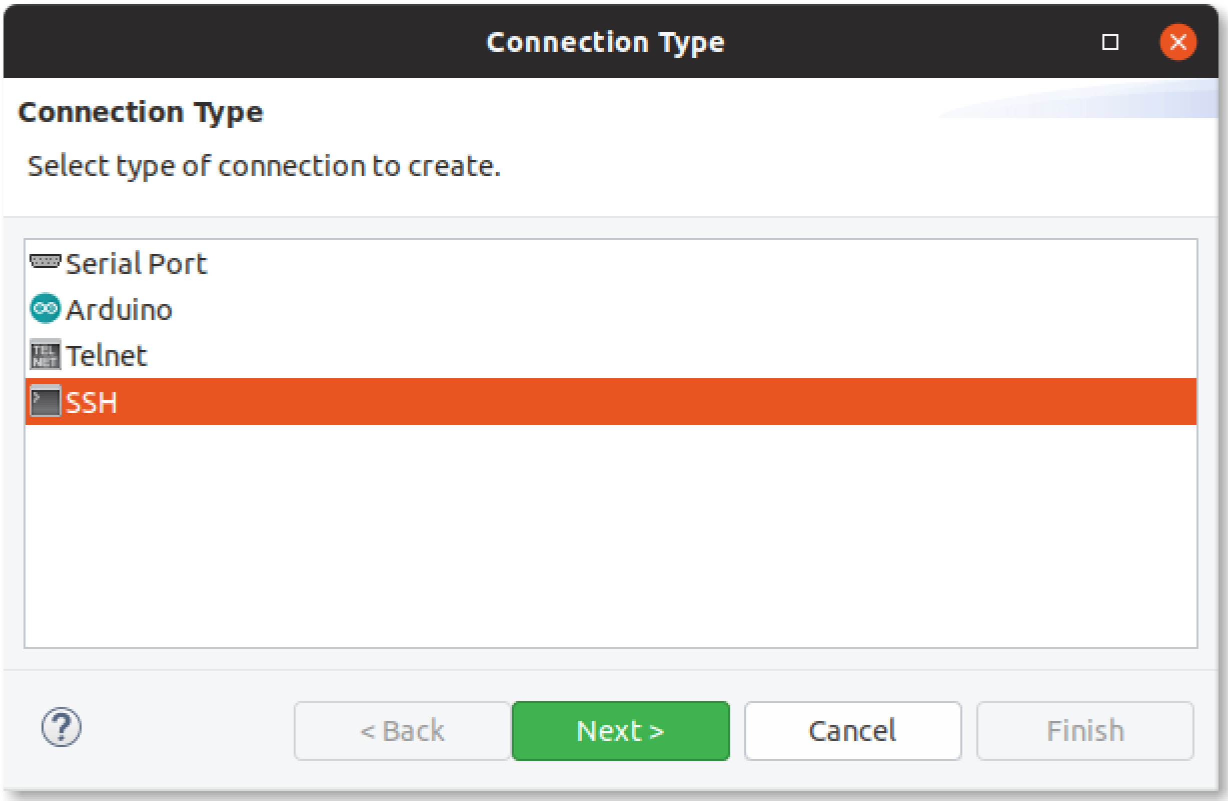
This section assumes:

- you already have an Eclipse project
- have successfully built an application from that project
- have some familiarity with Eclipse
- remote target is already configured for remote access via SSH

1. Create a Connection within Eclipse by selecting the **New Connection** option:



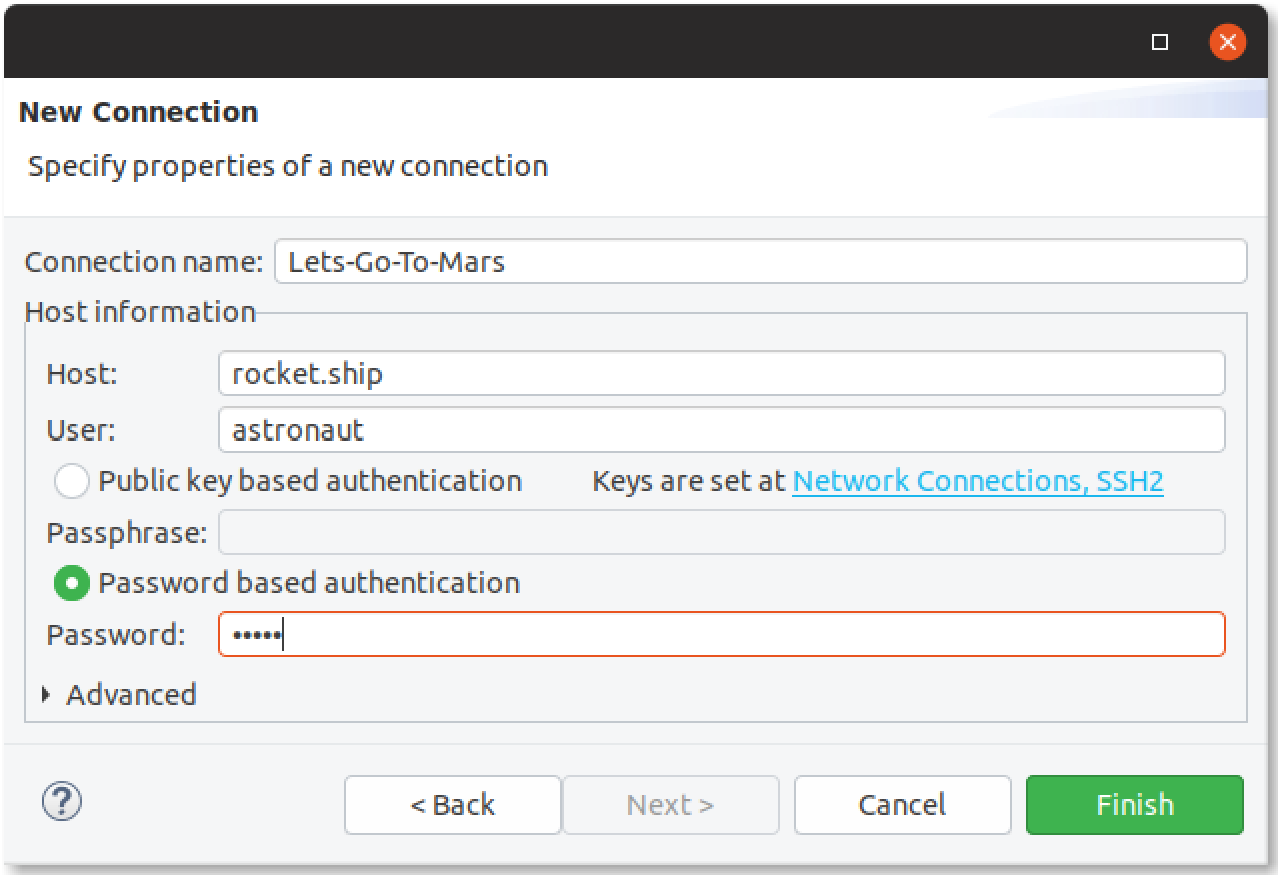
2. Select the SSH Connection type and click **Next**:



Connection Type

Select type of connection to create.

3. Enter your remote targets Hostname and Username. You can either use Key-based authentication or Password, depending on the configuration of the remote device.



New Connection

Specify properties of a new connection

Connection name: Lets-Go-To-Mars

Host information

Host: rocket.ship

User: astronaut

○ Public key based authentication   Keys are set at Network Connections, SSH2

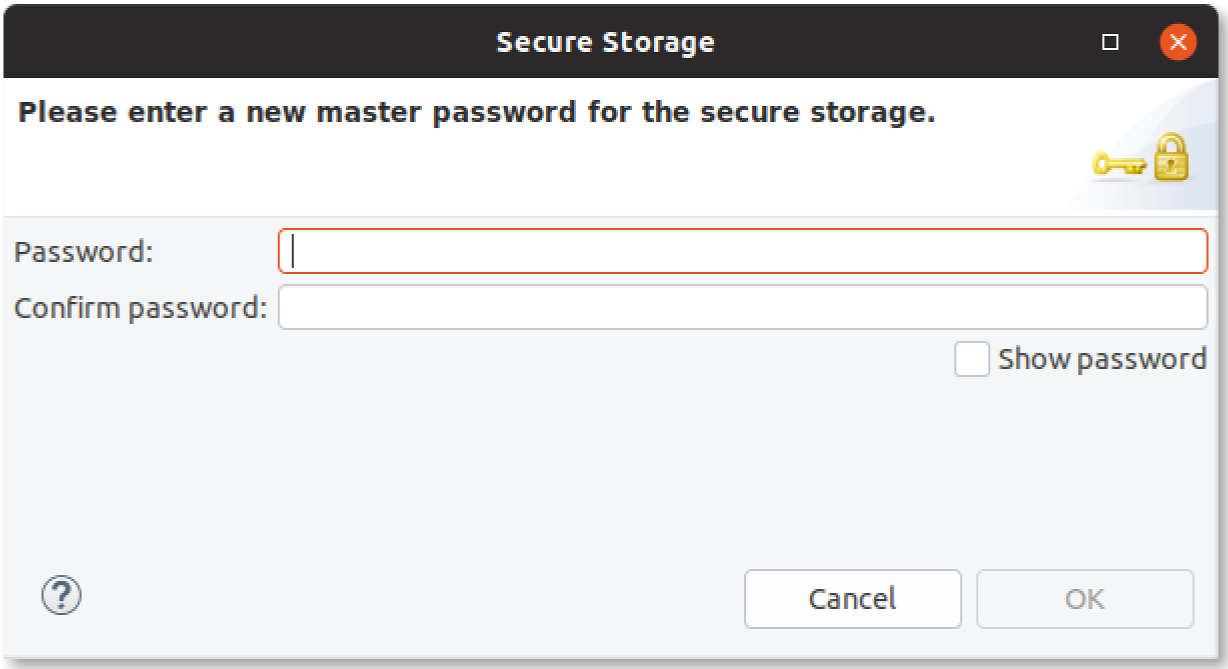
Passphrase:

● Password based authentication

Password: •••••

▸ Advanced

4. You may also be then prompted to set an Eclipse Secure Storage password. This protects the password you entered when stored on the disk. Enter the password and click OK.



Secure Storage

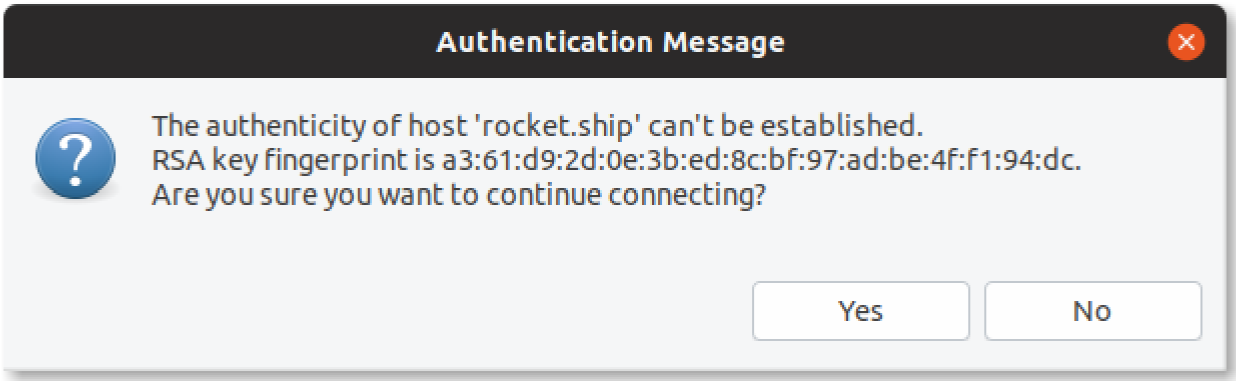
Please enter a new master password for the secure storage.

Password:

Confirm password:

□ Show password

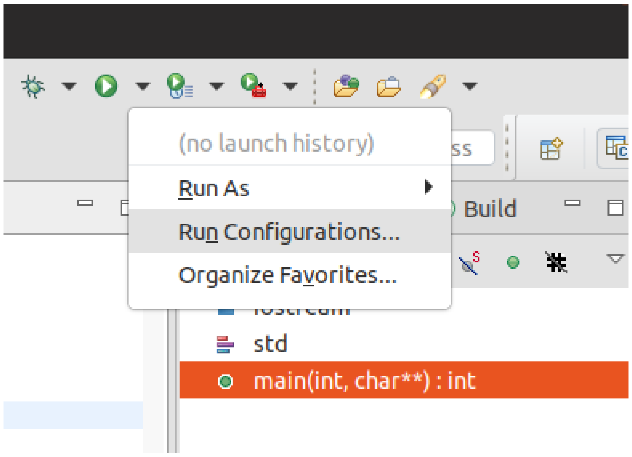
5. When connecting to the device for the first time, you should be prompted to trust the device. Click Yes to continue.



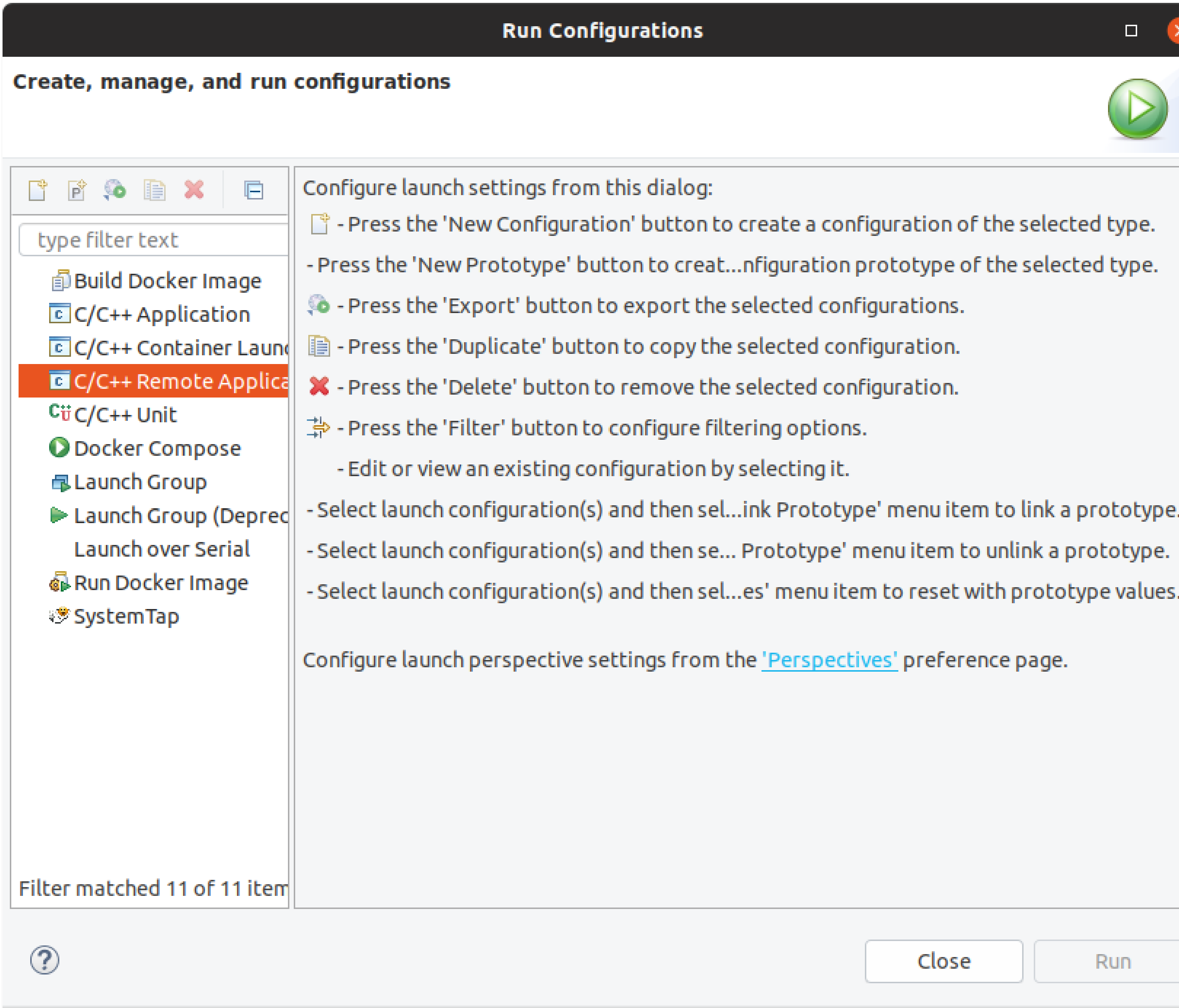
Authentication Message

The authenticity of host 'rocket.ship' can't be established.
RSA key fingerprint is a3:61:d9:2d:0e:3b:ed:8c:bf:97:ad:be:4f:f1:94:dc.
Are you sure you want to continue connecting?

6. Once a connection has been defined within Eclipse, open the the Run Configurations window to create your configuration.



7. Double-click on the "C/C++ Remote Application" configuration type.



Run Configurations

Create, manage, and run configurations

Configure launch settings from this dialog:

- Press the 'New Configuration' button to create a configuration of the selected type.
- Press the 'New Prototype' button to creat...nfiguration prototype of the selected type.
- Press the 'Export' button to export the selected configurations.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.
- Select launch configuration(s) and then sel...ink Prototype' menu item to link a prototype.
- Select launch configuration(s) and then se... Prototype' menu item to unlink a prototype.
- Select launch configuration(s) and then sel...es' menu item to reset with prototype values.

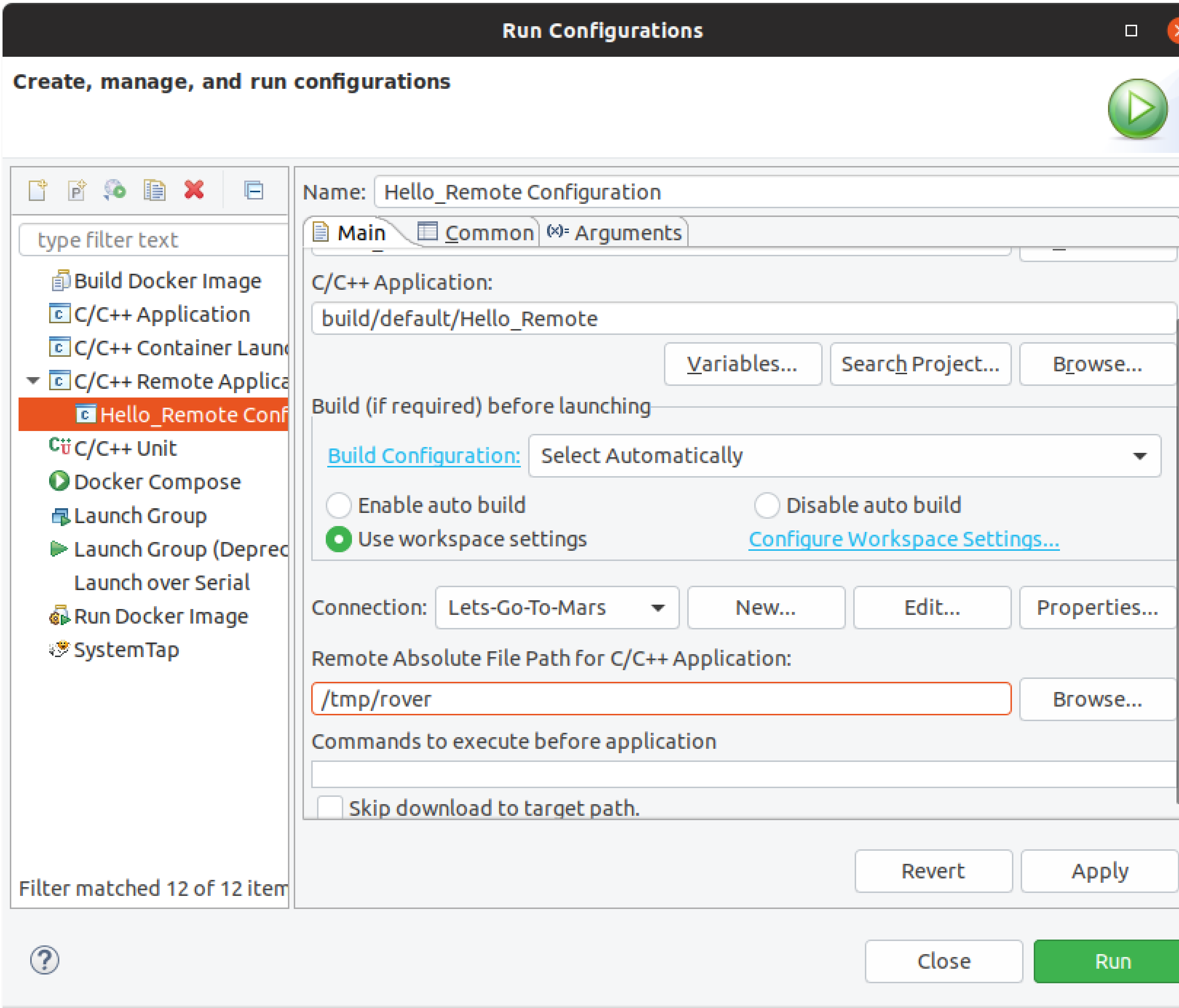
Configure launch perspective settings from the 'Perspectives' preference page.

Filter matched 11 of 11 item

This will create a run configuration for you project.

8. Set two properties on this view:

   a. Select the connection you created earlier.
   b. Define a location on the remote system where Eclipse will copy your project's binary. Your use-case may dictate where your binary must be copied to, but often a location your user has write permission to, such as `/tmp/`, will suffice.
9. Click Apply to save your configuration.
10. Click Run to run your project on the remote device.



Run Configurations

Create, manage, and run configurations

Name: Hello_Remote Configuration

C/C++ Application: build/default/Hello_Remote

Build (if required) before launching

Build Configuration: Select Automatically

○ Enable auto build
○ Disable auto build
● Use workspace settings

Connection: Lets-Go-To-Mars

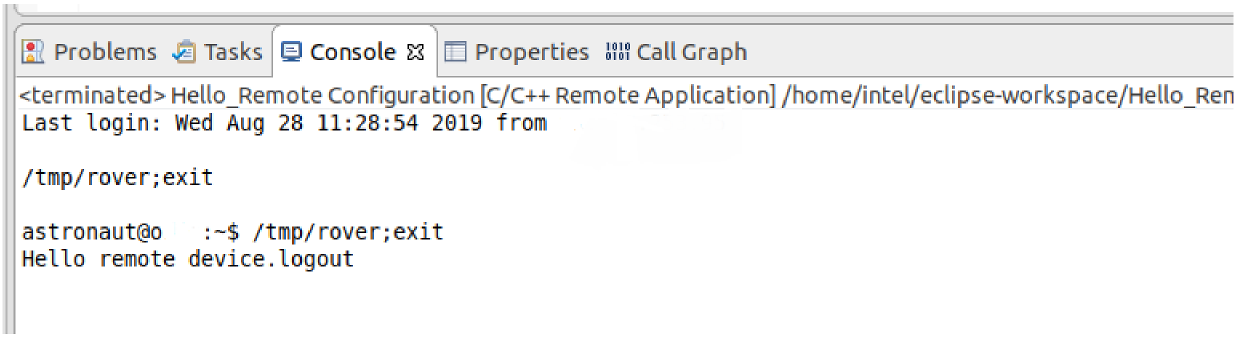
Remote Absolute File Path for C/C++ Application: /tmp/rover

Commands to execute before application

□ Skip download to target path.

Filter matched 12 of 12 item

You should see the output in the console of Eclipse.



```
<terminated> Hello_Remote Configuration [C/C++ Remote Application] /home/intel/eclipse-workspace/Hello_Ren
Last login: Wed Aug 28 11:28:54 2019 from

/tmp/rover;exit

astronaut@o    :~$ /tmp/rover;exit
Hello remote device.logout
```

Note that you can re-use the same connection and launch configuration for debug too. Your remote device will need at least `gdbserver` in the `PATH environment variable`.

Using Containers

Intel® oneAPI Base 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.

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

Singularity Containers

Build a Singularity image using a Singularity file.

Using Containers with the Command Line

Compile and run the containers directly.

The below enables the GPU, if available, using `--device=/dev/dri` (may not be available in Linux* VM or Windows*). The command will leave you at a command prompt, inside the container, in interactive mode.

```
image=intel/oneapi-basekit
# --device=/dev/dri enables the gpu (if available). May not be available in Linux VM or Windows
docker run --device=/dev/dri -it "$image"
```

Once in the container, you can interact with it using Run a Sample Project Using the Command Line.

NOTE You may need to include proxy settings before `-it "$image"`if you are behind a proxy:

```
docker run -e http_proxy="$http_proxy" -e https_proxy="$https_proxy" -it "$image"
```

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

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

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

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

Using Containers with Eclipse*

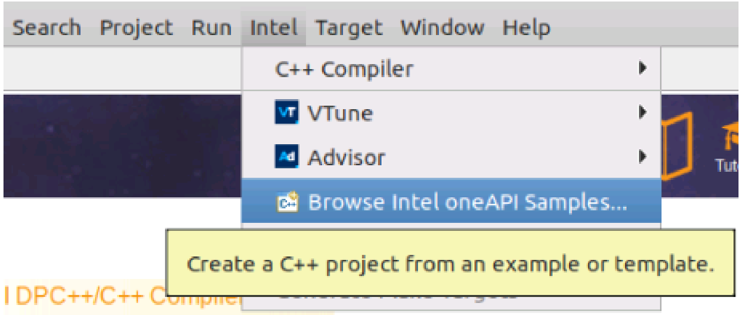
Install Eclipse

1. Prerequisite: ensure Java* JRE 8 or 11 is installed.
2. Download and install the latest Eclipse IDE for C/C++ Developers.
3. Download and install the Intel® oneAPI Base Toolkit in order to easily browse samples.

Use the Samples Browser to Create an Eclipse Project

If you have installed the Intel® oneAPI Base Toolkit, you can use the Samples Browser integrated into the Eclipse IDE to create a project based on Intel® samples.

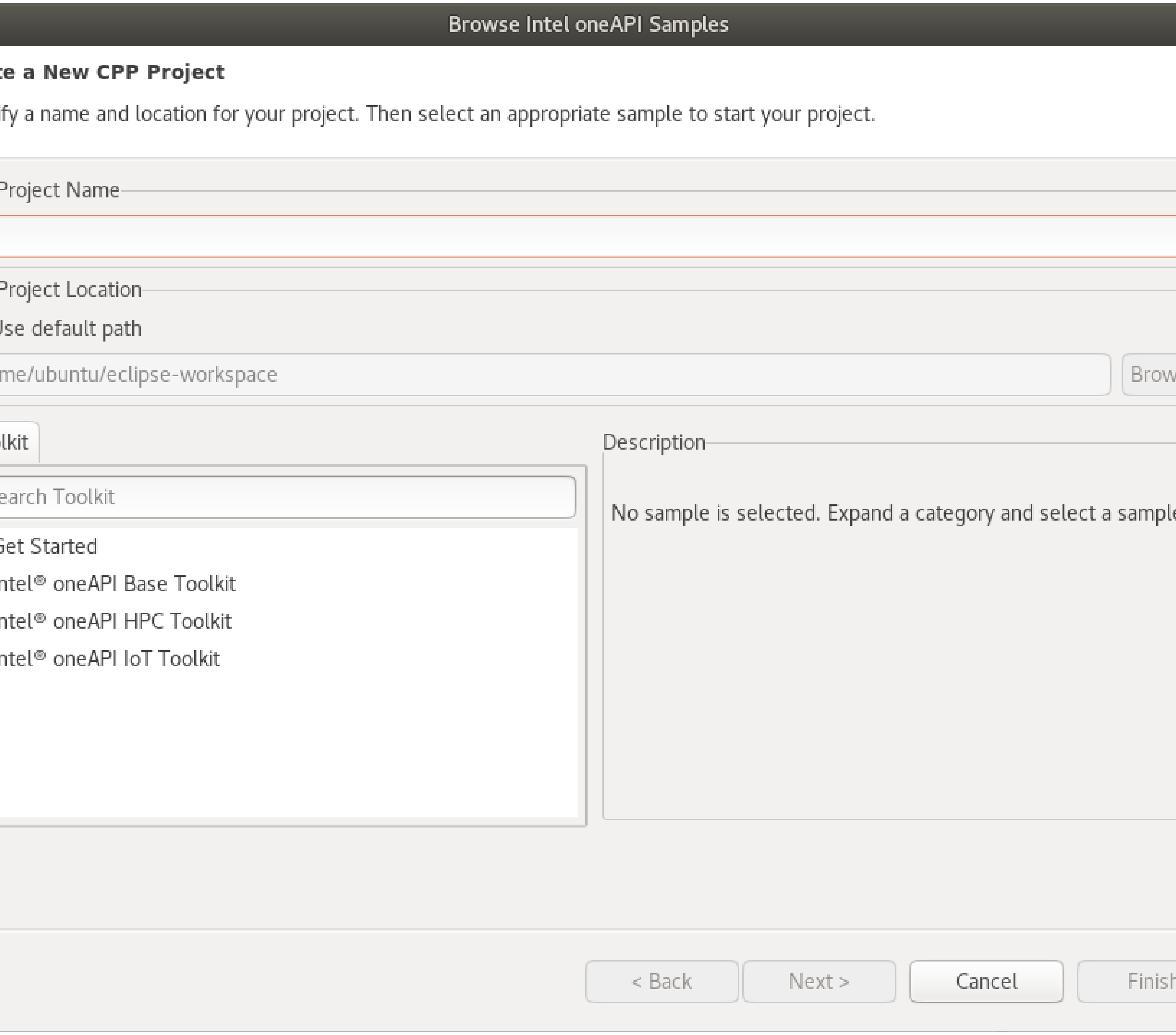
Run Eclipse and select **Intel** > **Browse Intel oneAPI Samples...**



Create a C++ project from an example or template.

NOTE If you do not see an Intel menu in Eclipse, install the oneAPI Eclipse plugins, then return to this page for instructions on how to Build and Run a sample.

The Sample Browser dialog window will be displayed.



e a New CPP Project

fy a name and location for your project. Then select an appropriate sample to start your project.

Project Name

Project Location

Use default path

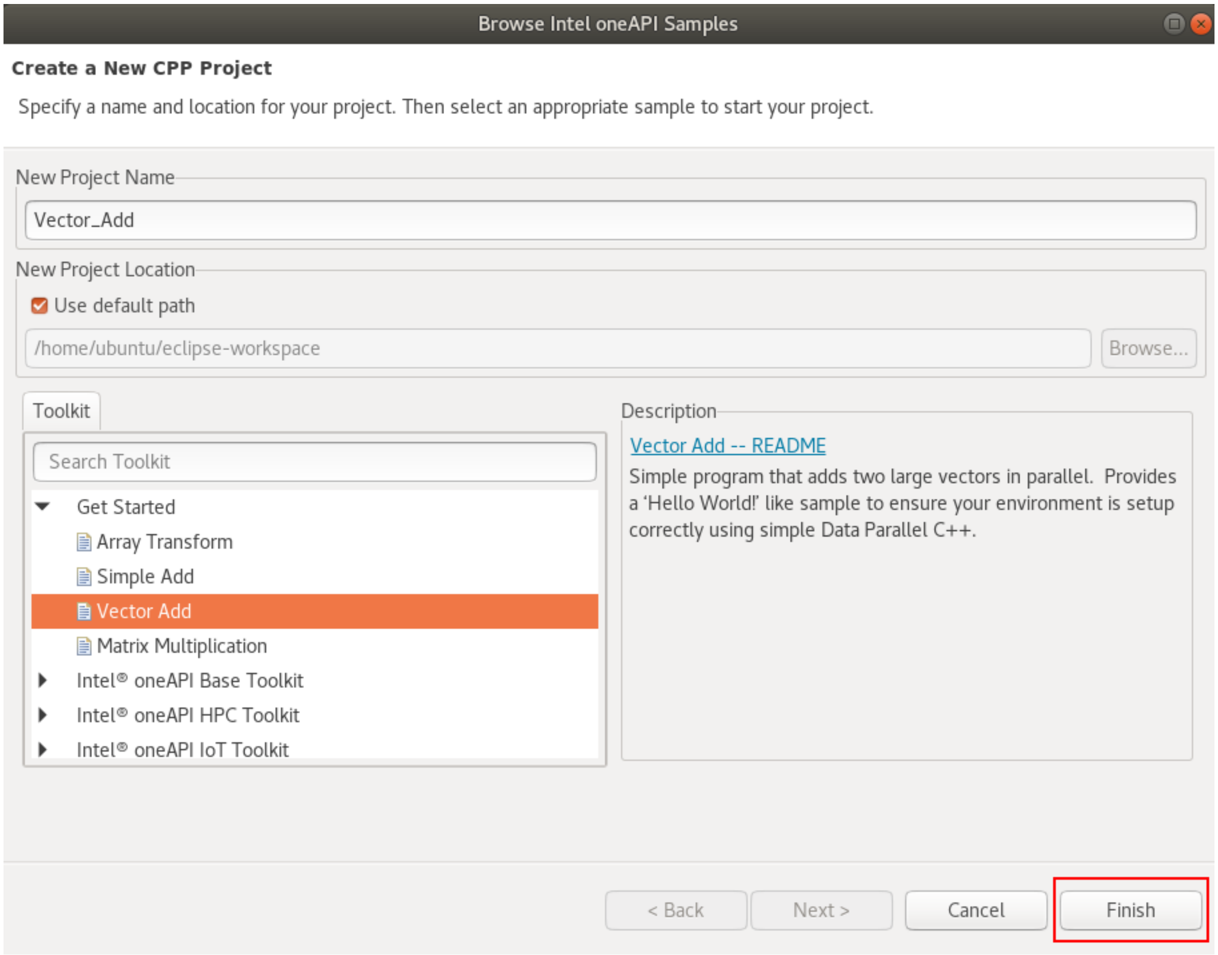
me/ubuntu/eclipse-workspace

Get Started
ntel® oneAPI Base Toolkit
ntel® oneAPI HPC Toolkit
ntel® oneAPI IoT Toolkit

Description

No sample is selected. Expand a category and select a sample

Expand the sample tree and select a sample. Click **Finish** to create a project based on the sample.



Browse Intel oneAPI Samples

Create a New CPP Project

Specify a name and location for your project. Then select an appropriate sample to start your project.

New Project Name: Vector_Add

New Project Location

☑ Use default path

/home/ubuntu/eclipse-workspace

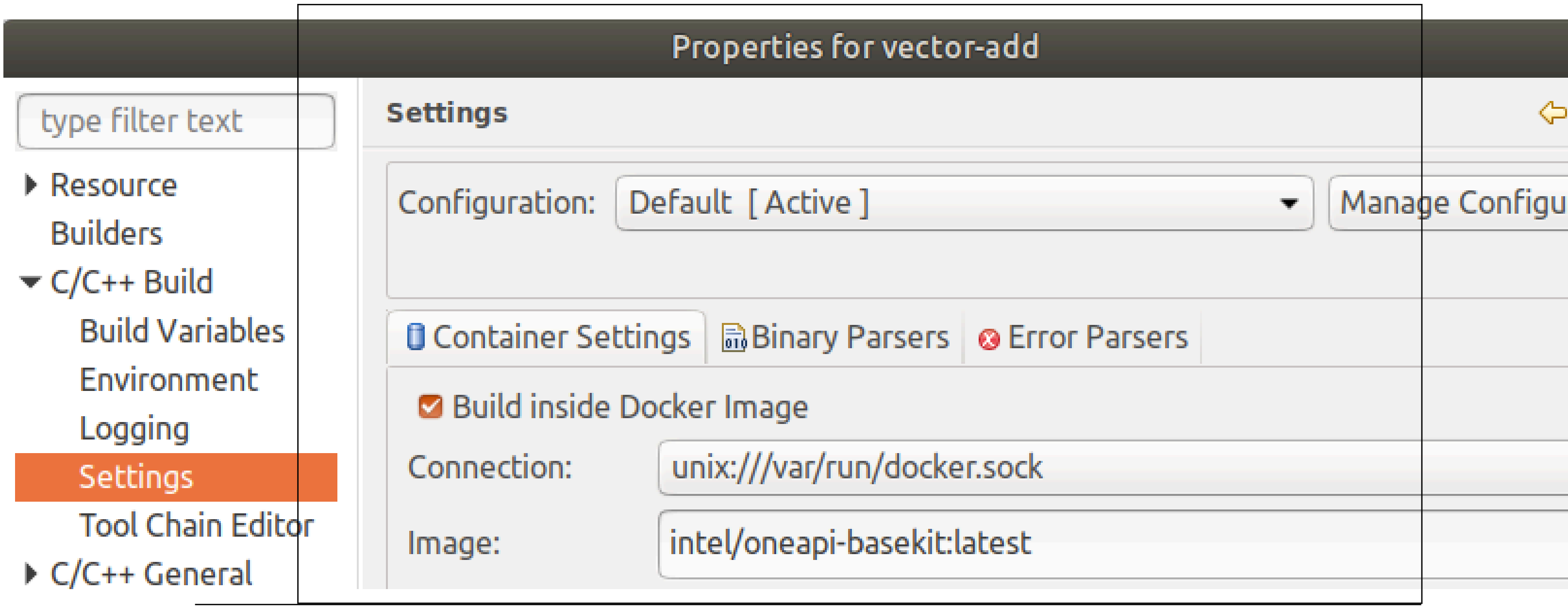
Description

Vector Add -- README

Simple program that adds two large vectors in parallel. Provides a 'Hello World!' like sample to ensure your environment is setup correctly using simple Data Parallel C++.

Configure Project Properties

1. Make sure you've followed the steps described on the Using Containers page to download the Docker image.
2. Open the Project Properties by right clicking **Project** > **Properties**.
3. In **Properties** select **C/C++ Build** > **Settings**, and open the **Container Settings** tab to set the following:



Configuration: Default [Active]

☑ Build inside Docker Image

Connection: unix:///var/run/docker.sock

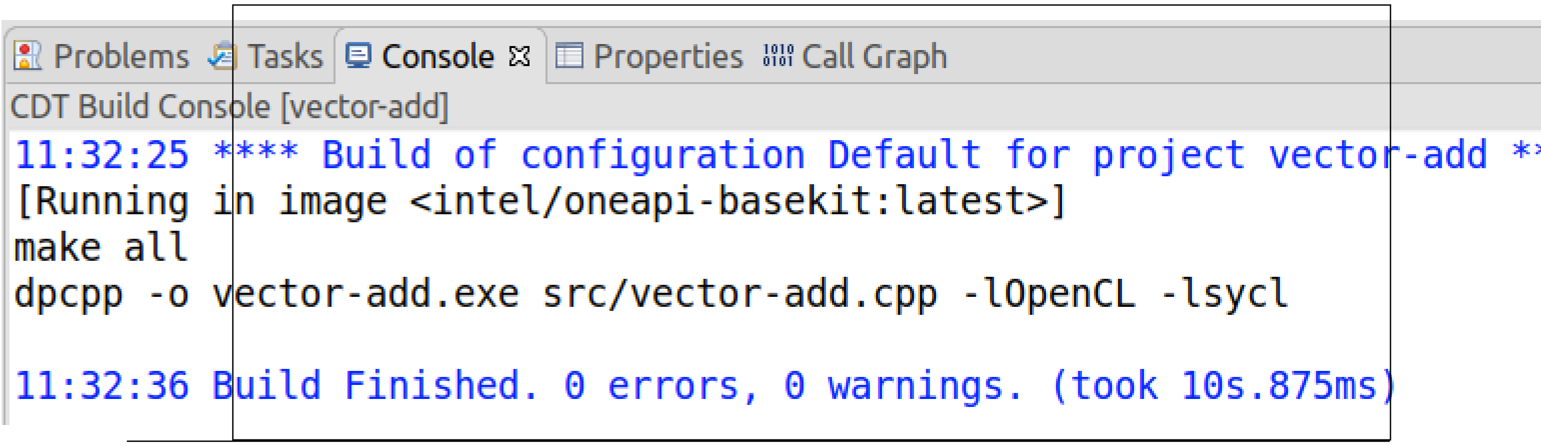
Image: intel/oneapi-basekit:latest

- **a.** Check **Build Inside Docker Image**.
- **b.** Select the docker image to use. If there are no images listed, verify that Docker is enabled for non-root users. Alternatively, you can run Eclipse with sudo.
- **c.** Apply and close the settings.

Compile the Application

1. Click the Build Icon

   .
2. Confirm that the console prints "Running in image..."

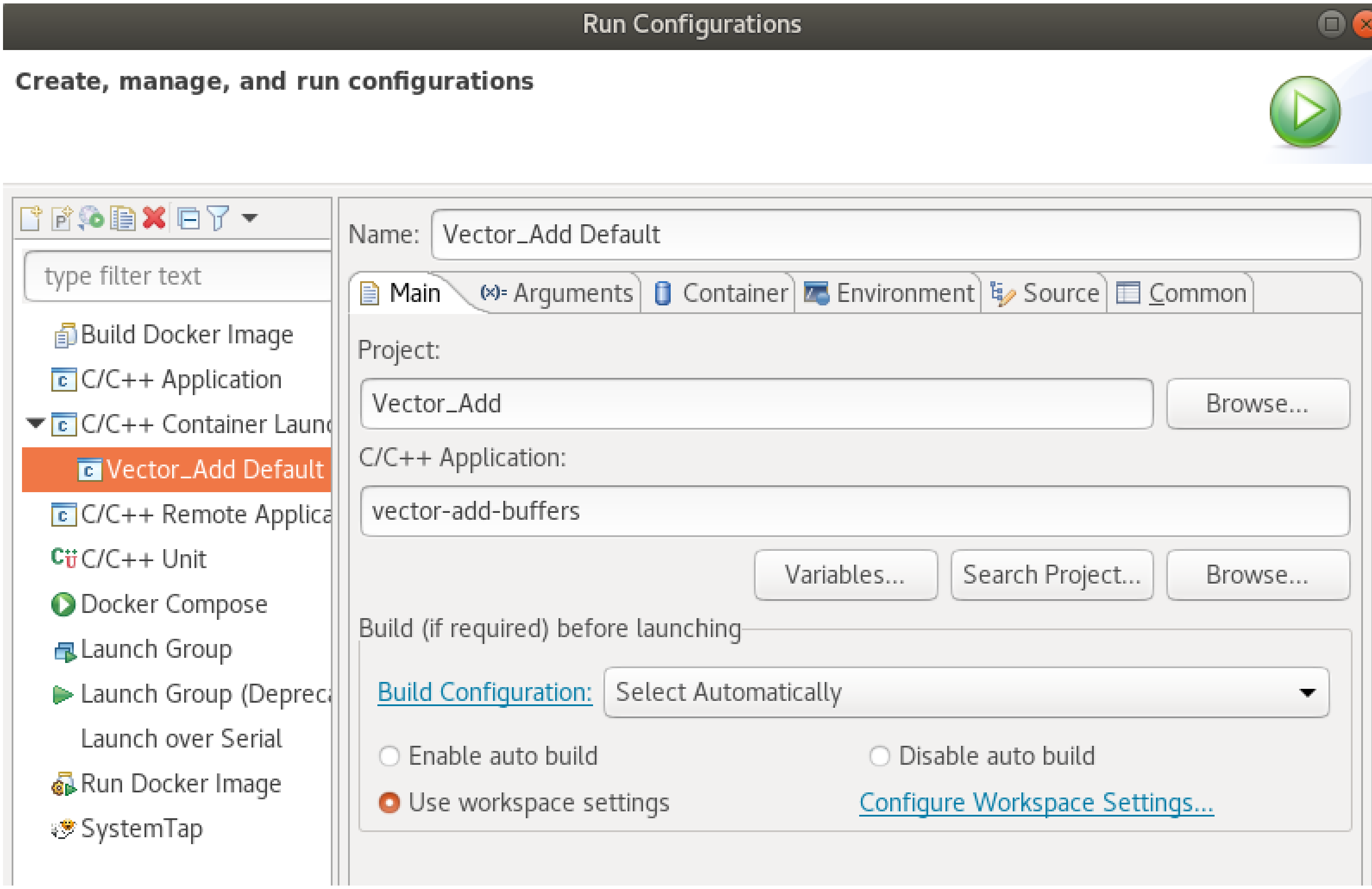


```
11:32:25 **** Build of configuration Default for project vector-add **
[Running in image <intel/oneapi-basekit:latest>]
make all
dpcpp -o vector-add.exe src/vector-add.cpp -lOpenCL -lsycl

11:32:36 Build Finished. 0 errors, 0 warnings. (took 10s.875ms)
```

Run Application

1. Create a Run Configuration with **Run** > **Run Configurations**.
2. Create a new C/C++ Container Launcher.



Run Configurations

Create, manage, and run configurations

type filter text

- Build Docker Image
- C/C++ Application
- C/C++ Container Launc
  - Vector_Add Default
- C/C++ Remote Applica
- C/C++ Unit
- Docker Compose
- Launch Group
- Launch Group (Depreca
- Launch over Serial
- Run Docker Image
- SystemTap

Name: Vector_Add Default

Main | Arguments | Container | Environment | Source | Common

Project: Vector_Add

C/C++ Application: vector-add-buffers

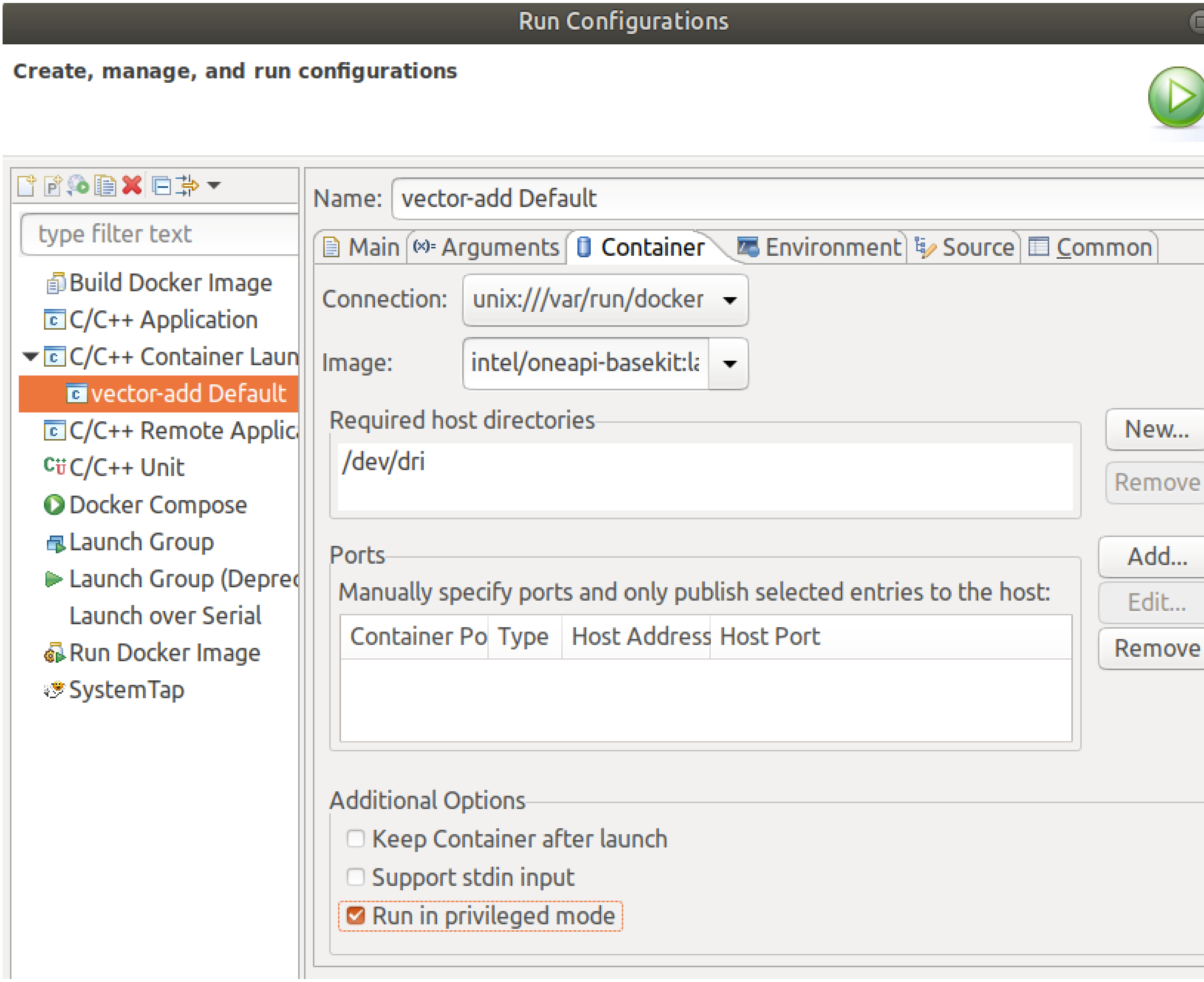
Build (if required) before launching

Build Configuration: Select Automatically

○ Enable auto build ○ Disable auto build

● Use workspace settings

3. Select the Container tab and make sure the **Image** is set to the correct Docker image. To enable GPU in the container, add `/dev/dri` in the **Required Host Directories** and check **Run in Privileged Mode**.



Name: vector-add Default

Connection: unix:///var/run/docker

Image: intel/oneapi-basekit:la

Required host directories

/dev/dri

Ports

Manually specify ports and only publish selected entries to the host:

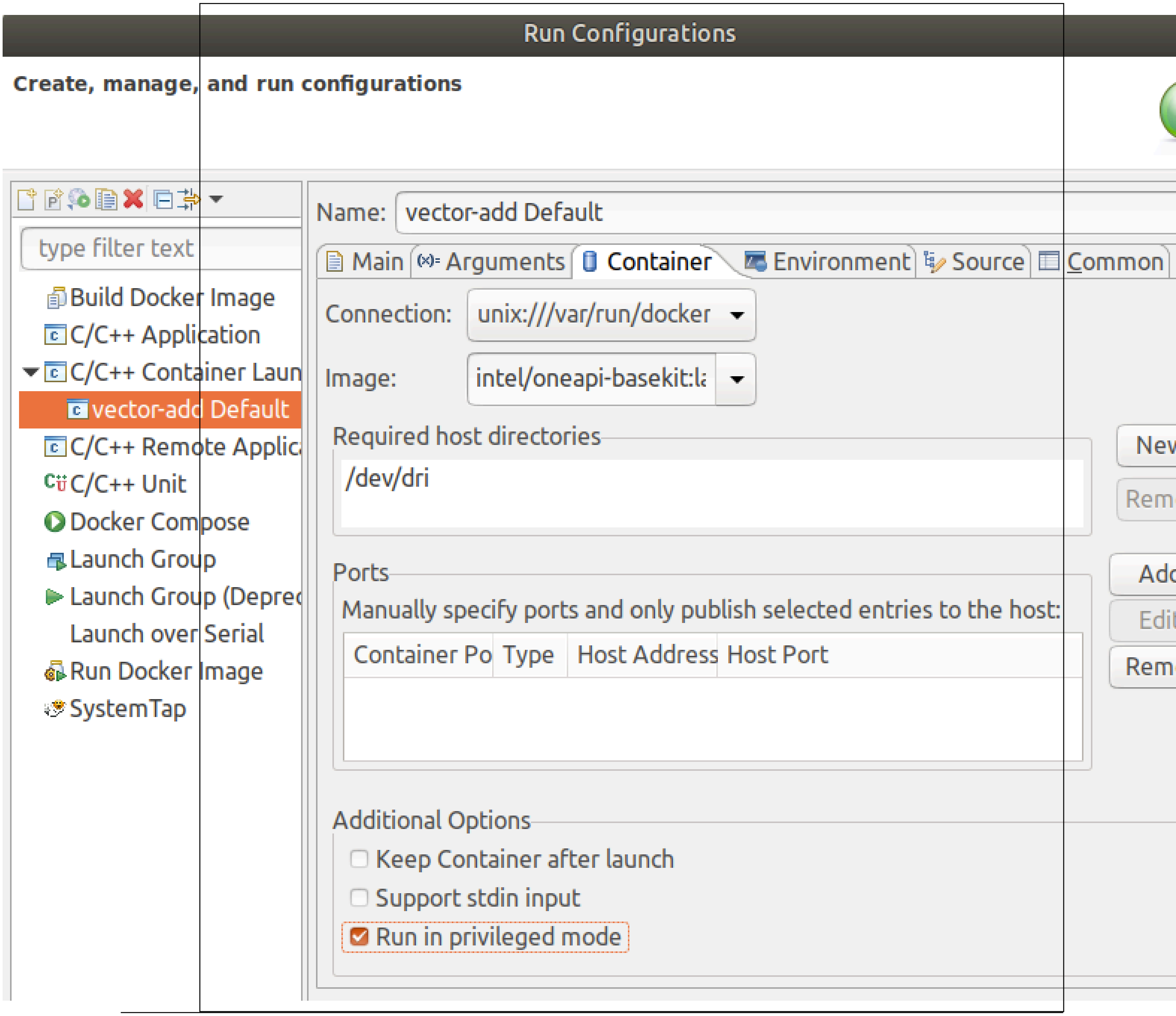
Additional Options

□ Keep Container after launch

□ Support stdin input

☑ Run in privileged mode

4. To enable GPU in the container, select the **Container** tab and add `/dev/dri` as Required Host Directories, and check **Run in Privileged Mode**.



Name: vector-add Default

Connection: unix:///var/run/docker

Image: intel/oneapi-basekit:la

Required host directories

/dev/dri

Ports

Manually specify ports and only publish selected entries to the host:

Container Po	Type	Host Address	Host Port

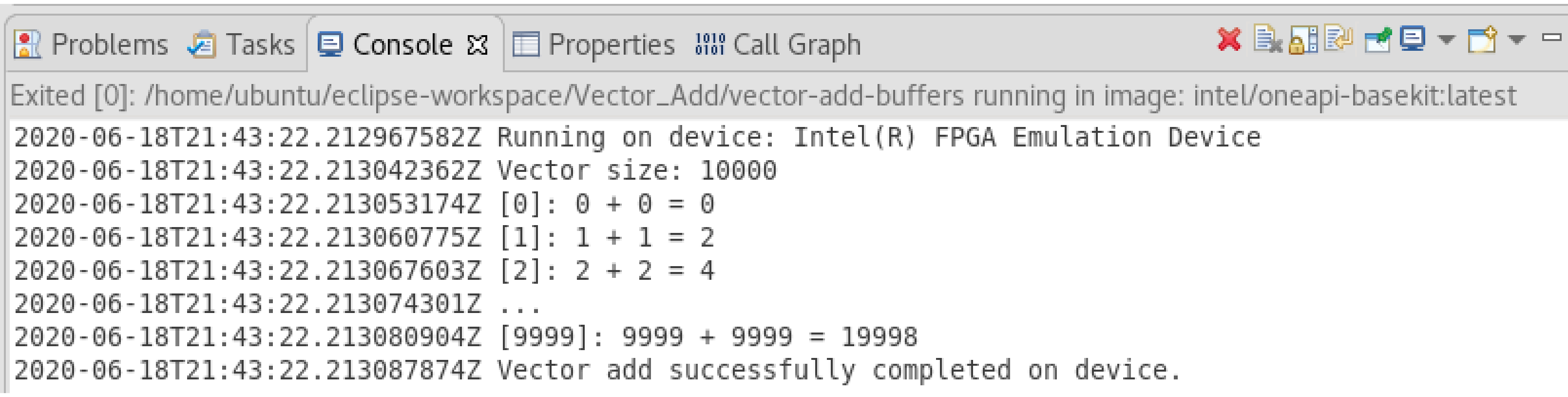
Additional Options

☐ Keep Container after launch

☐ Support stdin input

☑ Run in privileged mode

5. Click the **Run** icon.
6. Confirm that the console prints `Running in image...`



Exited [0]: /home/ubuntu/eclipse-workspace/Vector_Add/vector-add-buffers running in image: intel/oneapi-basekit:latest

```
2020-06-18T21:43:22.212967582Z Running on device: Intel(R) FPGA Emulation Device
2020-06-18T21:43:22.213042362Z Vector size: 10000
2020-06-18T21:43:22.213053174Z [0]: 0 + 0 = 0
2020-06-18T21:43:22.213060775Z [1]: 1 + 1 = 2
2020-06-18T21:43:22.213067603Z [2]: 2 + 2 = 4
2020-06-18T21:43:22.213074301Z ...
2020-06-18T21:43:22.213080904Z [9999]: 9999 + 9999 = 19998
2020-06-18T21:43:22.213087874Z Vector add successfully completed on device.
```

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

Potential errors and how to avoid or fix them.

Issue	How to fix
Errors that occur during installation or directly after installation.	See the Troubleshooting page of the Intel® oneAPI Toolkits Installation Guide for Linux* OS.
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: `make 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.
Developing with Offline Systems: I need to develop on a system that is not connected to the internet.	If you are using an offline system, download the samples from a system that is internet connected and transfer the sample files to your offline system. After you have downloaded the samples, follow the instructions in the `README.md` file. The readme file is located in the folder of the sample you are interested in. The samples can be downloaded from the code samples repository. The full set of documentation can be downloaded from Downloadable Documentation.
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.

Issue	How to fix
Unable to install toolkits or access libraries.	Verify that you meet the System Requirements listed on the Intel® oneAPI Base Toolkit product page
Problems connecting from behind a proxy	The oneAPI CLI Samples Browser does not work with system proxy settings and does not support WPAD proxy. If you are unable to access the samples from the oneAPI CLI Samples Browser, set the value of https_proxy and http_proxy, then run oneapi-cli in the same terminal. To set the proxy enter this command, where `your.proxy` is the name of your proxy server and `8080` is your default port. `export http_proxy=http://your.proxy:8080` After you set the proxy, return to the instructions for Running a Sample Using the Command Line and try again.
The Intel plug-ins for Eclipse did not install correctly, or the Intel menu is missing from Eclipse.	See Install Eclipse Plugins.
OpenCL* drivers are not found	Refer to the GPU Drivers section of the Installation Guide for Intel® oneAPI Toolkits for information on downloading and installing OpenCL drivers.

Next Steps

After successfully building a sample application, Explore SYCL with Samples from Intel and explore the tools in the Intel® oneAPI Base Toolkit.

Tool	Description
Intel® DPC++ Compatibility Tool	The Intel® DPC++ Compatibility Tool assists in migration of CUDA* applications to SYCL* ready code that can use the Intel® oneAPI DPC++/C++ Compiler. Get started.
Intel® oneAPI DPC++/C++ Compiler	The Intel® oneAPI DPC++/C++ Compiler targets CPUs and accelerators through single-source code while permitting custom tuning. Get Started.
Intel® oneAPI DPC++ Library	This library is a companion to the Intel® oneAPI DPC++/C ++ Compiler and provides an alternative for C++ developers who create heterogeneous applications and solutions. Its APIs are based on familiar standards-C++ STL, Parallel STL (PSTL), Boost.Compute, and SYCL*-to maximize productivity and performance across CPUs, GPUs, and FPGAs.
Intel® Distribution for GDB*	GDB, the GNU Project debugger, allows you to see what is going on `inside' another program while it executes -- or what another program was doing at the moment it crashed. Get Started. Learn more.
Intel® oneAPI Math Kernel Library (oneMKL)	The oneMKL helps you achieve maximum performance with a math computing library of highly optimized, extensively parallelized routines for CPU and GPU. The library has C and Fortran interfaces for most routines on CPU, and SYCL* interfaces for some routines on both CPU and GPU. Get started.

Intel® oneAPI Threading Building Blocks (oneTBB)	oneTBB is a flexible performance library that simplifies the work of adding parallelism to complex applications, even if you're not a threading expert. Learn more.
Intel® Integrated Performance Primitives	Intel® Integrated Performance Primitives (Intel® IPP) is an extensive library of ready-to-use, domain-specific functions that are highly optimized for diverse Intel® architectures. Get Started with Intel® IPP or Get Started with Intel® IPP Cryptography.
Intel® oneAPI Data Analytics Library	The Intel® oneAPI Data Analytics Library (oneDAL) is a library that helps speed up big data analysis by providing highly optimized algorithmic building blocks for all stages of data analytics (preprocessing, transformation, analysis, modeling, validation, and decision making) in batch, online, and distributed processing modes of computation. The current version of oneDAL provides SYCL* API extensions to the traditional C++ interface. Get started.
Intel® oneAPI Collective Communications Library	Intel® oneAPI Collective Communications Library (oneCCL) is a scalable and high-performance communication library for Deep Learning (DL) and Machine Learning (ML) workloads. It develops the ideas originated in Intel(R) Machine Learning Scaling Library and expands the design and API to encompass new features and use cases. Get started.
Intel® oneAPI Deep Neural Network Library	The Intel® oneAPI Deep Neural Network Library (oneDNN) is an open-source performance library for deep learning applications. The library includes basic building blocks for neural networks optimized for Intel® Architecture Processors and Intel® Processor Graphics. oneDNN is intended for deep learning applications and framework developers interested in improving application performance on Intel CPUs and GPUs. Get started.
Intel® VTune™ Profiler	Intel® VTune™ Profiler is a performance analysis tool targeted for users developing serial and multithreaded applications. The tool is delivered as a Performance Profiler with Intel Performance Snapshots and supports local and remote target analysis on the Windows* and Linux* platforms. Get started.
Intel® Advisor	Intel® Advisor gives software architects and developers the data and analysis tools they need to build well-threaded and vectorized code that exploits modern hardware capabilities. Get started.
Intel® FPGA Add-on for oneAPI Base Toolkit	Use reconfigurable hardware to accelerate data-centric workloads. Learn more.

Learn more about SYCL and targeting other accelerators using the following resources:

Resource	**Description**
Intel® oneAPI Programming Guide	Provides details on the oneAPI programming model, including details about SYCL standards, programming for various target accelerators, and introductions to the oneAPI libraries.
Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel® Arria 10 GX FPGA	A quick start guide for the Intel® Programmable Acceleration Card with Intel® Arria® 10 GX FPGA. This guide provides instructions to load and run a loopback test, Hello FPGA, in both non-virtualized and virtualized environments.
Intel Acceleration Stack Quick Start Guide: Intel® FPGA Programmable Acceleration Card D5005	A quick start guide for Intel® FPGA PAC D5005. This guide provides the instructions for installing the Open Programmable Acceleration Engine (OPAE) on the host Intel® Xeon® Processor to manage and access the Intel® FPGA PAC, configuring and flashing the FPGA image and Board Management Controller (BMC) images, running the example `hello_afu` in a virtualized and non-virtualized environment, and handling graceful thermal shutdown.

Intel® FPGA SDK for OpenCL™ Pro Edition: Custom Platform Toolkit User Guide	Outlines the procedure for creating an Intel® FPGA Software Development Kit (SDK) for OpenCL™ Pro Edition Custom Platform. The Intel® FPGA SDK for OpenCL™ Pro Edition Custom Platform Toolkit provides the necessary tools for implementing a fully functional Custom Platform.
Intel® FPGA Acceleration Hub	Learn about the Acceleration Stack for Intel® Xeon® CPUs with FPGAs and other Intel FPGA-based acceleration platforms.
FPGA Optimization Guide for Intel® oneAPI Toolkits	The oneAPI FPGA Optimization Guide provides guidance on leveraging the functionalities of SYCL to optimize your design.
Explore SYCL Through Intel® FPGA Code Samples	Provides guidance on how to target and develop your design on an FPGA using the oneAPI programming model. It also provides guidance on how to optimize a design to achieve performance and latency targets for an application targeting the FPGA.
oneAPI GPU Optimization Guide	The oneAPI GPU Optimization Guide demonstrates how to improve the behavior of your software by partitioning it across the host and accelerator to specialize portions of the computation that run best on the accelerator. Specialization includes restructuring and tuning the code to create the best mapping of the application to the hardware. The value of oneAPI is that it allows each of these variations to be expressed in a common language with device-specific variants launched on the appropriate accelerator.

For more information about this toolkit, see the Intel® oneAPI Base Toolkit page.

SSH: Password-less Access to Remote Linux* Target

Introduction

For some oneAPI applications, you must configure a password-less SSH connection for the root user on the target system; for example:

- IoT applications that use the MRAA/UPM sensor library
- Any application that interacts with system resources that require su, sudo, or root access
- Any tool that requires remote root or sudo access to your target system

When you finish the configuration steps below you will be able to “ssh into” your remote Linux target from your host development system without a password prompt, as a normal (non-root) user or as a root user.

For an introduction to SSH and how SSH keys work, see SSH Essentials: Working with SSH Servers, Clients, and Keys.

> **NOTE** Password-less access works only when you connect to your target system from your host development system with a matching private SSH key. Attempting to connect from a different host system will still require a password.

Configure Password-less SSH Access

These instructions apply to:

- **Host development system:** Linux*, Windows*, or macOS*
- **Target system:** Linux

Set up an .ssh directory

On your host development system, follow these steps:

Step 1: Open a terminal session (CMD window on Windows) and CD to your home directory

Step 2: Create .ssh directory

Enter the following commands to create an .ssh directory, set the proper permissions, and CD into the new .ssh directory.

At a Windows CMD prompt

```
> %HomeDrive%  &&  cd %HomePath%
> mkdir .ssh
> cd .ssh
```

At a Linux terminal (bash) prompt

```
$ cd ~
$ mkdir -p .ssh
$ chmod 700 .ssh
$ cd .ssh
```

Step 3: Generate keys and copy to the target system

> **NOTE** From this point forward the instructions apply to all host development systems (Windows, Linux, and macOS).

1. To generate a *default-named* RSA key pair with an *empty passphrase* (that is, do not provide a passphrase when asked), enter:

```
$ ssh-keygen -t rsa
```

2. To copy the new public key to your target system's *non-root user* home folder, enter the following, where:

   username = the name used to access the target and *target* = the IP address or the network hostname of the target

   You should be prompted for the non-root user password for your target device.

```
$ scp id_rsa.pub username@target:id_rsa.pub
$ ssh username@target
$ cd ~
$ mkdir -p .ssh
$ chmod 700 .ssh
$ cat ~/id_rsa.pub >>.ssh/authorized_keys
$ chmod 600 .ssh/authorized_keys
$ exit
```

Step 4: Confirm that a password is no longer required (non-root)

Follow this step to confirm that a password is no longer required for your non-root user.

1. To display the target's **system information** strings, including the target's hostname as the second field in the output, enter:

```
ssh username@target uname -a
```

Step 5: Configure password-less access to root on your target

1. To login to the *non-root user* on the target using SSH and switch to the *root user* using sudo, enter:

```
$ ssh username@target
$ cd ~
$ sudo -E bash
```

   Note that the sudo command should prompt you for your target system's *non-root user* password.
2. To copy the public key that you transferred to the non-root user account on the target into the root user's authorized keys file, enter:

```
$ mkdir -p /root/.ssh
$ chmod 700 /root/.ssh
$ cat ./id_rsa.pub >>/root/.ssh/authorized_keys
$ chmod 600 /root/.ssh/authorized_keys
```

3. Exit twice, first from the sudo bash session, second from the ssh connection:

```
$ exit
$ exit
```

Step 6: [Optional] Check your progress

To test the root connection for your target, enter:

```
$ ssh root@target ls -a
```

You should see a directory listing of all files located in the /root folder on your target, without the need for a login prompt.

Next: Create a New Connection and Connect to Your Target

Notes

- Password-less access works only when you connect to your target system from your host development system with a matching private SSH key. Attempting to connect from a different host system will still require a password.
- Make sure that you have created a project for Linux targets, and that this project is selected in the Project Explorer.

See Also

Why sudo user can use sched_setscheduler SCHED_RR while root can not?

Operation not permitted while setting new priority for thread

Debugging libmraa {#debugging}

SSH: Running Applications Built with Eclipse*

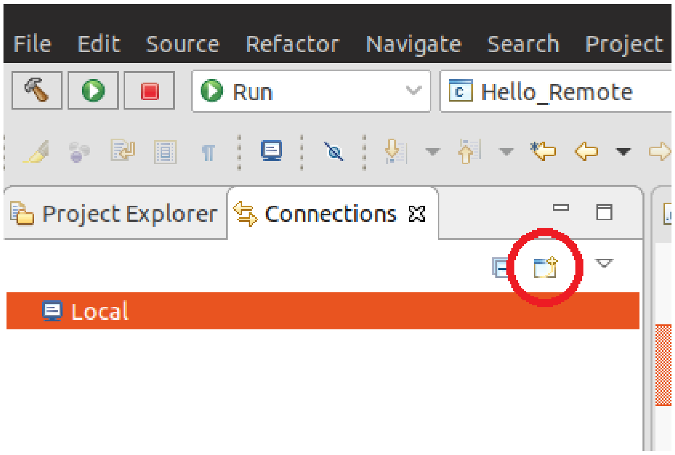
The following describes how to run a built application on a "remote target" system using an SSH connection.

If your application requires a password-less SSH connection, see Password-less Access to a Remote Linux* Target Device.

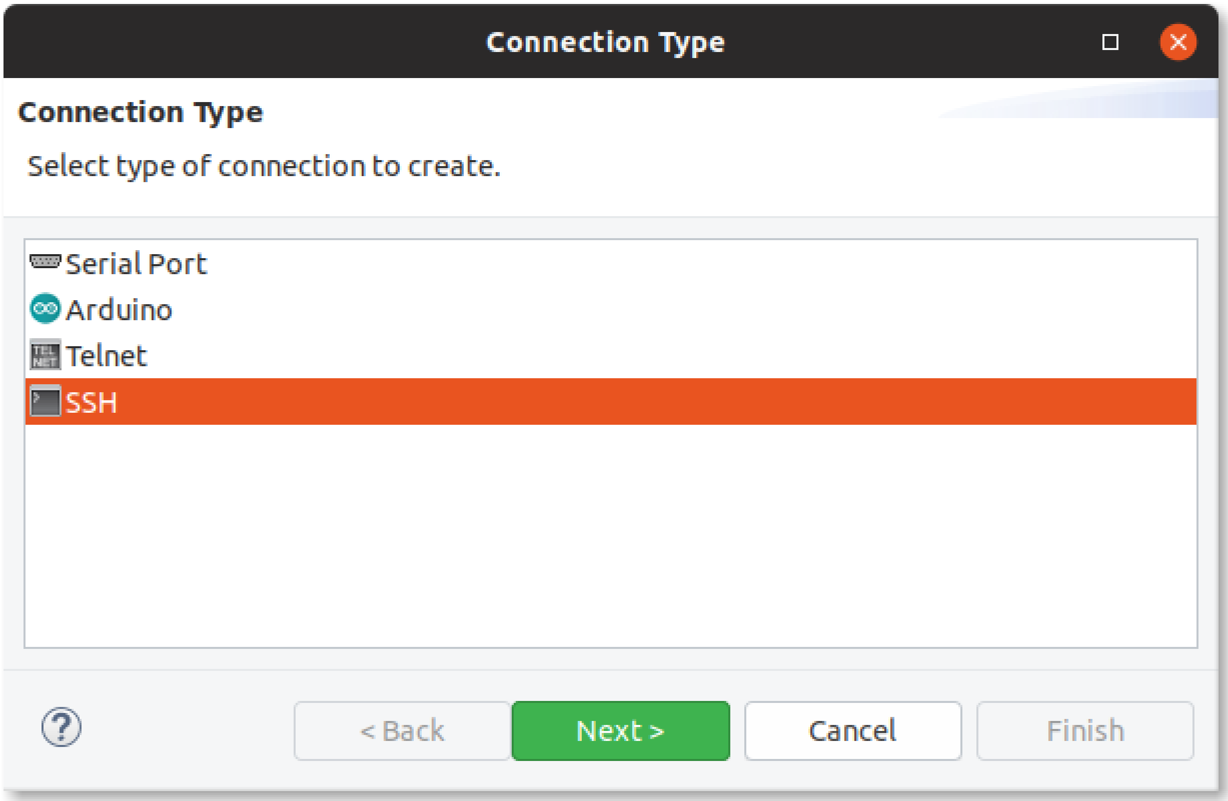
This section assumes:

- you already have an Eclipse project
- have successfully built an application from that project
- have some familiarity with Eclipse
- remote target is already configured for remote access via SSH

1. Create a Connection within Eclipse by selecting the **New Connection** option:



2. Select the SSH Connection type and click **Next**:

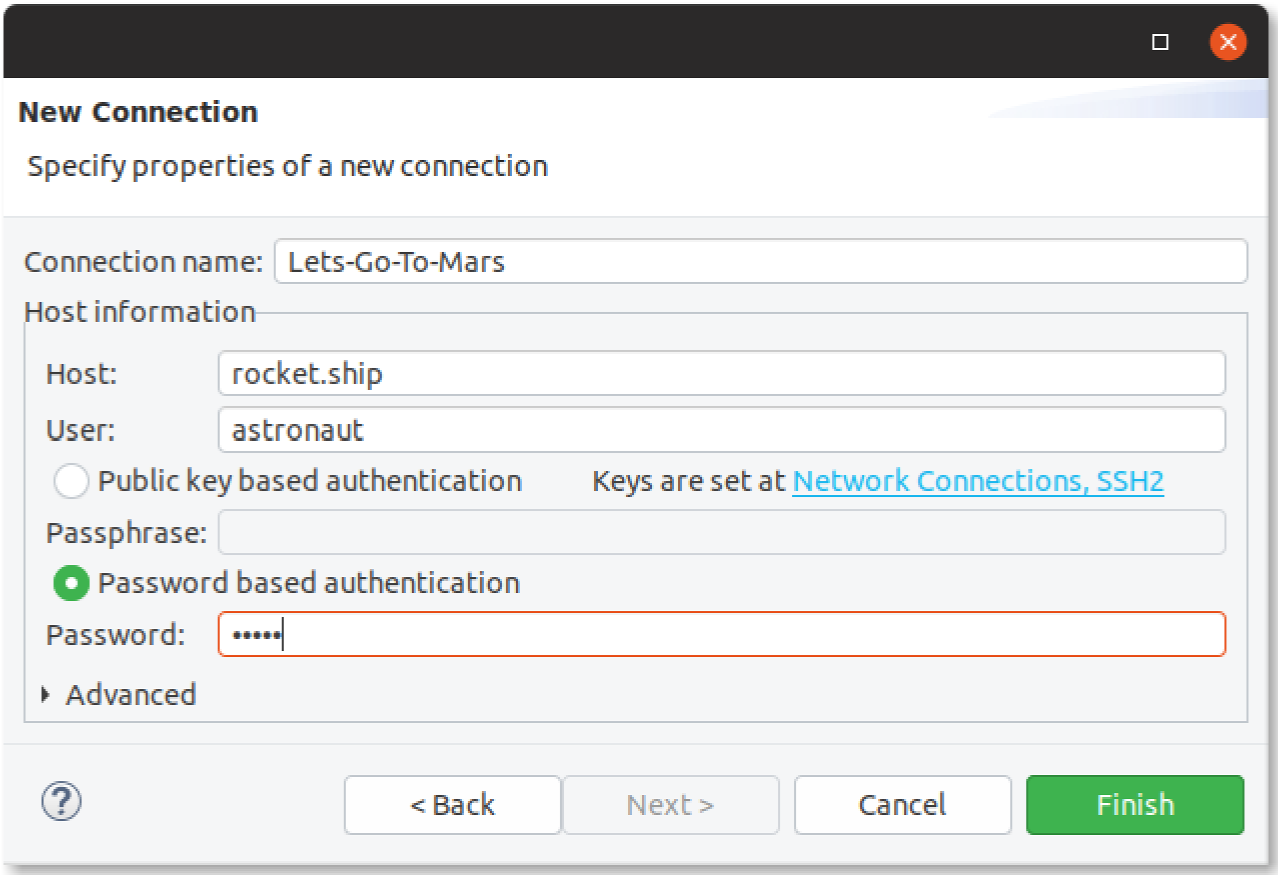


Connection Type

Select type of connection to create.

Serial Port
Arduino
Telnet
SSH

3. Enter your remote targets Hostname and Username. You can either use Key-based authentication or Password, depending on the configuration of the remote device.



New Connection

Specify properties of a new connection

Connection name: Lets-Go-To-Mars

Host information

Host: rocket.ship

User: astronaut

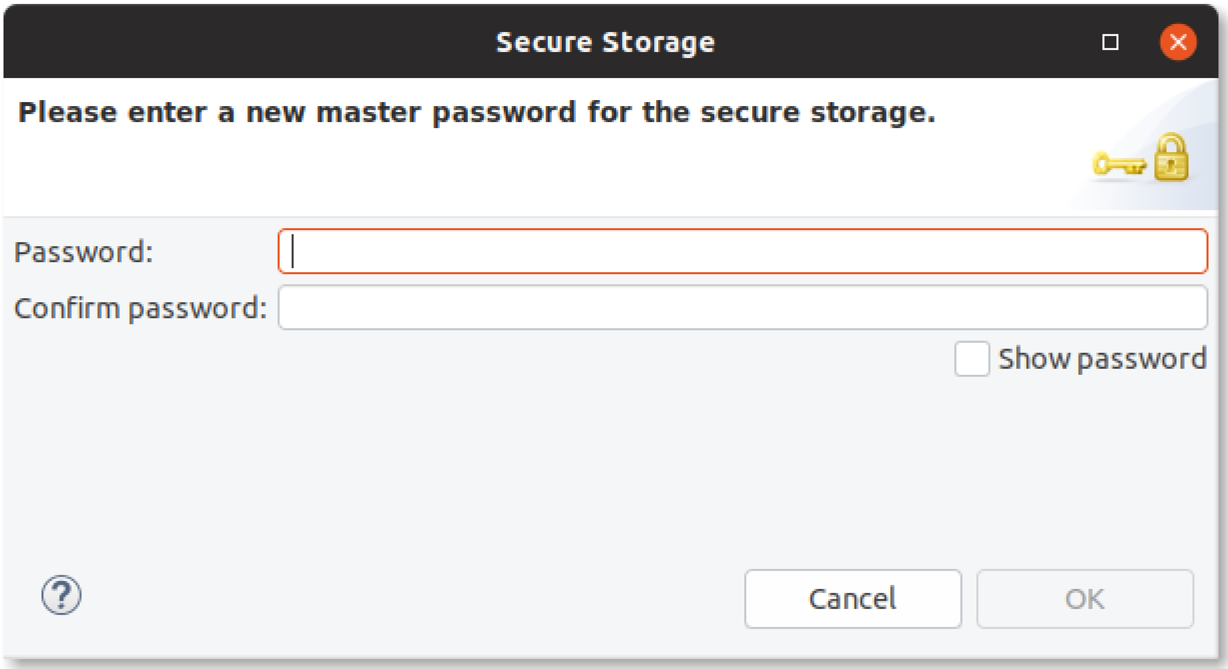
○ Public key based authentication Keys are set at Network Connections, SSH2

Passphrase:

● Password based authentication

Password: •••••

4. You may also be then prompted to set an Eclipse Secure Storage password. This protects the password you entered when stored on the disk. Enter the password and click OK.



Secure Storage

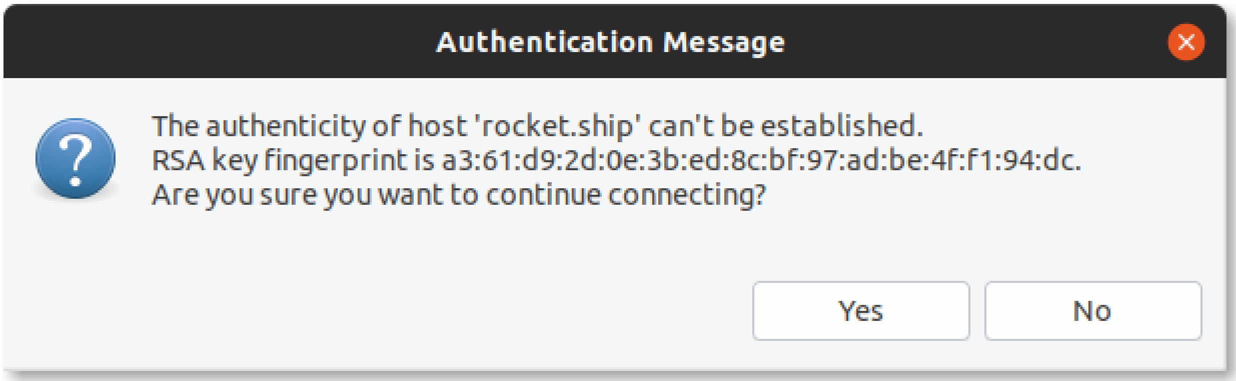
Please enter a new master password for the secure storage.

Password:

Confirm password:

□ Show password

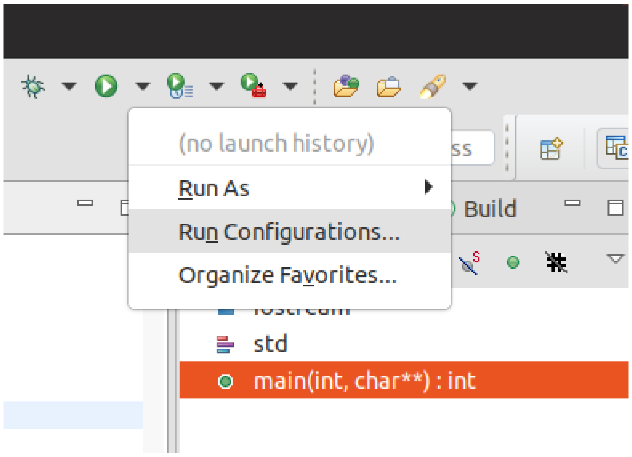
5. When connecting to the device for the first time, you should be prompted to trust the device. Click Yes to continue.



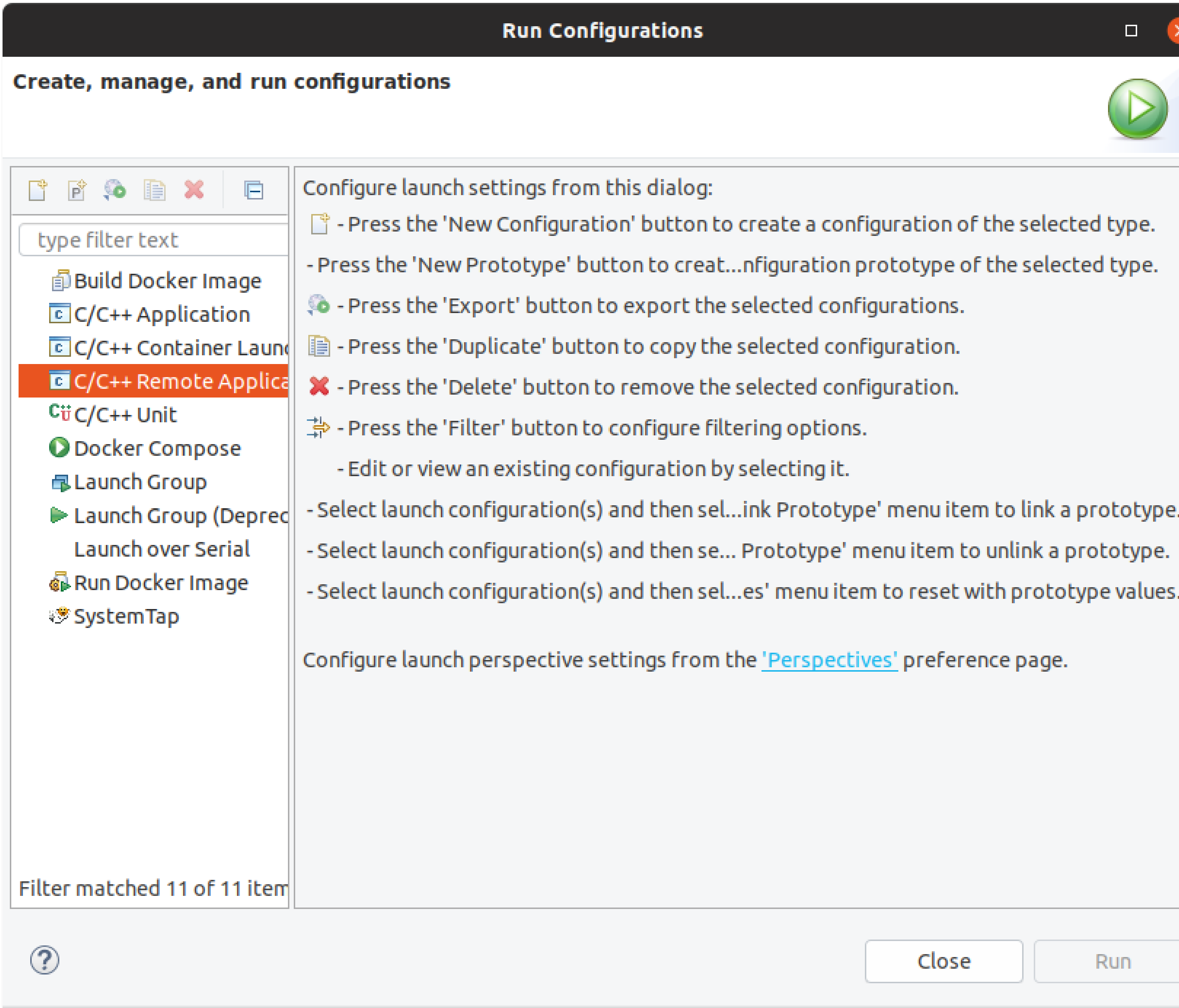
Authentication Message

The authenticity of host 'rocket.ship' can't be established.
RSA key fingerprint is a3:61:d9:2d:0e:3b:ed:8c:bf:97:ad:be:4f:f1:94:dc.
Are you sure you want to continue connecting?

6. Once a connection has been defined within Eclipse, open the the Run Configurations window to create your configuration.



7. Double-click on the "C/C++ Remote Application" configuration type.



Run Configurations

Create, manage, and run configurations

Configure launch settings from this dialog:

- Press the 'New Configuration' button to create a configuration of the selected type.

- Press the 'New Prototype' button to creat...nfiguration prototype of the selected type.

- Press the 'Export' button to export the selected configurations.

- Press the 'Duplicate' button to copy the selected configuration.

- Press the 'Delete' button to remove the selected configuration.

- Press the 'Filter' button to configure filtering options.

- Edit or view an existing configuration by selecting it.

- Select launch configuration(s) and then sel...ink Prototype' menu item to link a prototype.

- Select launch configuration(s) and then se... Prototype' menu item to unlink a prototype.

- Select launch configuration(s) and then sel...es' menu item to reset with prototype values.

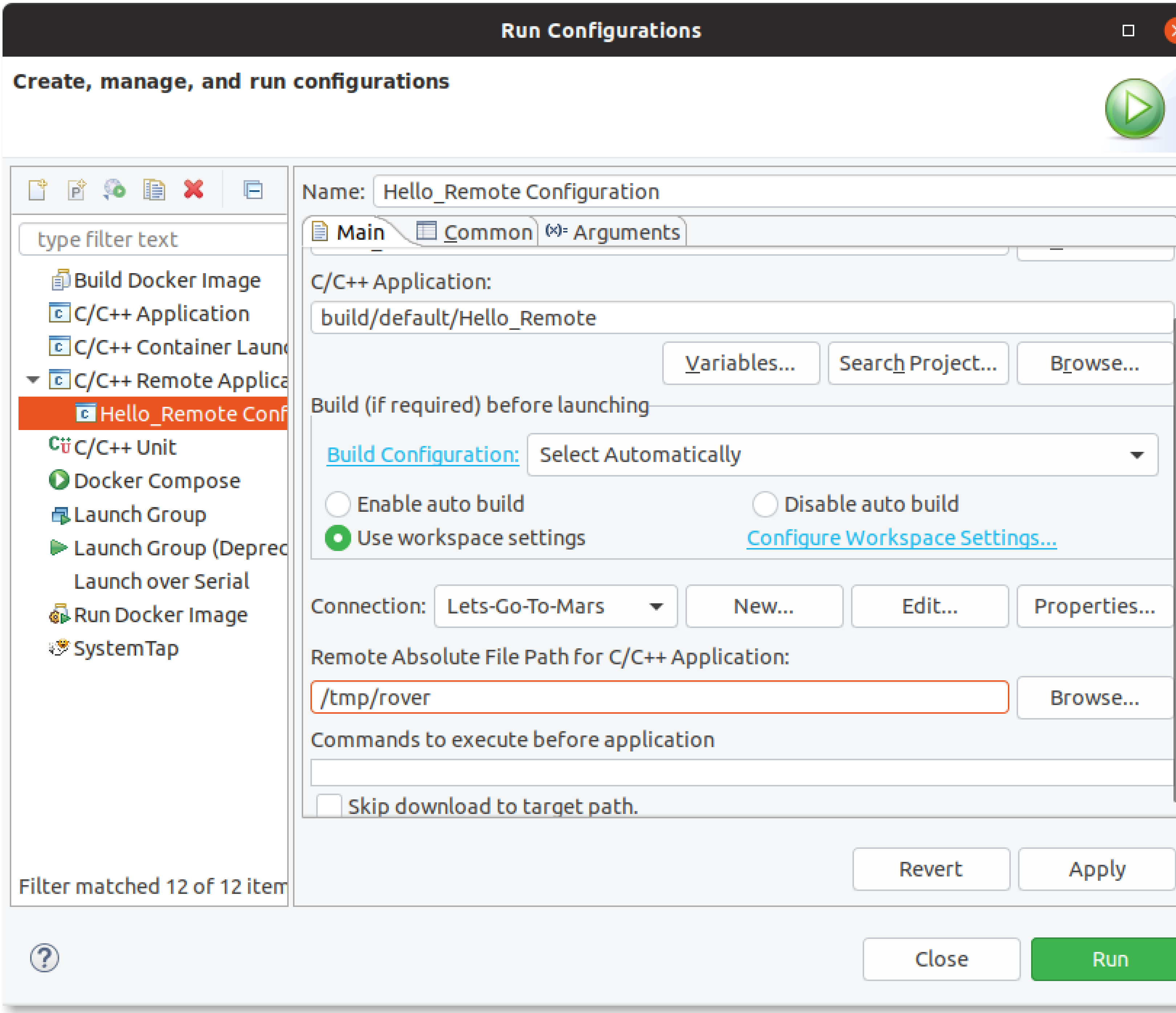
Configure launch perspective settings from the 'Perspectives' preference page.

This will create a run configuration for you project.

8. Set two properties on this view:

   a. Select the connection you created earlier.
   b. Define a location on the remote system where Eclipse will copy your project's binary. Your use-case may dictate where your binary must be copied to, but often a location your user has write permission to, such as `/tmp/`, will suffice.

9. Click Apply to save your configuration.
10. Click Run to run your project on the remote device.



Run Configurations

Create, manage, and run configurations

Name: Hello_Remote Configuration

C/C++ Application:

build/default/Hello_Remote

Build (if required) before launching

Build Configuration: Select Automatically

○ Enable auto build ○ Disable auto build

● Use workspace settings

Connection: Lets-Go-To-Mars

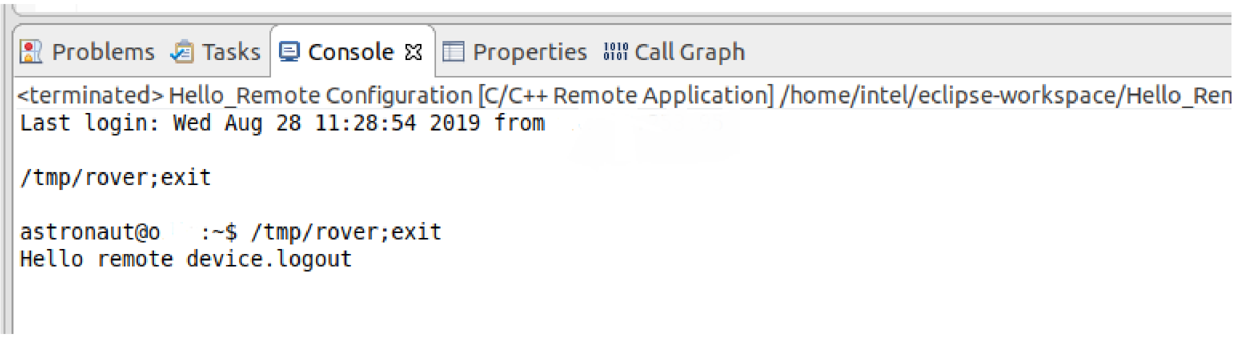
Remote Absolute File Path for C/C++ Application:

/tmp/rover

Commands to execute before application

□ Skip download to target path.

You should see the output in the console of Eclipse.



```
<terminated> Hello_Remote Configuration [C/C++ Remote Application] /home/intel/eclipse-workspace/Hello_Ren
Last login: Wed Aug 28 11:28:54 2019 from

/tmp/rover;exit

astronaut@o    :~$ /tmp/rover;exit
Hello remote device.logout
```

Note that you can re-use the same connection and launch configuration for debug too. Your remote device will need at least `gdbserver` in the `PATH environment variable`.

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

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.