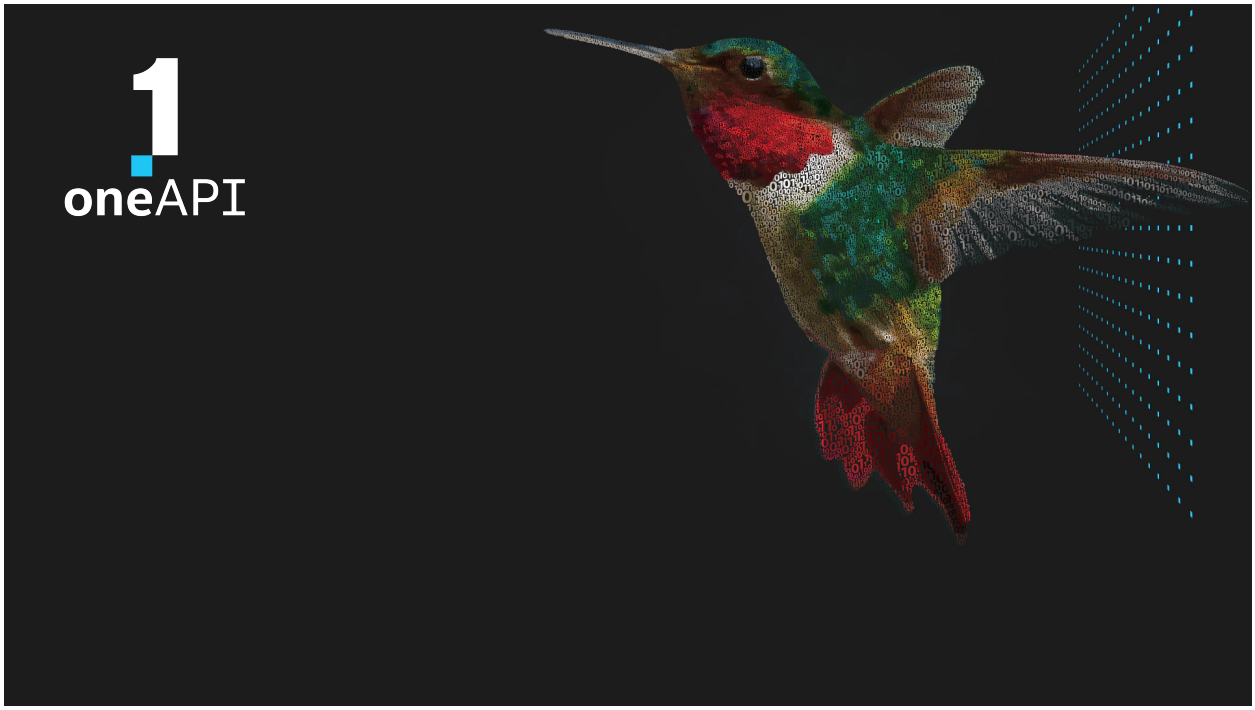
Cristian Moyano | Universidad Champagnat

# oneAPI:
# A New Era of Accelerated Computing




## Introducción

oneAPI es un modelo de programación abierto entre arquitecturas que permite a los desarrolladores utilizar una única base de código en varias arquitecturas. El resultado es una computación acelerada sin depender de un proveedor.

Repositorio:

https://github.com/cristianemoyano/oneAPI-example

## ¿Qué es oneAPI?

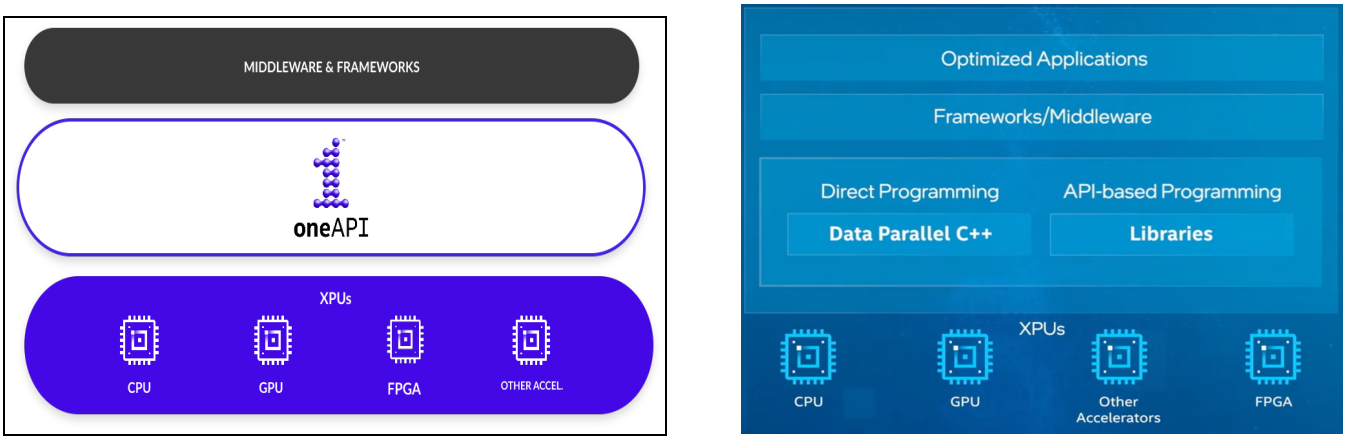



***Figura 1. Diagrama de oneAPI***

**oneAPI** es un modelo de programación abierto, inter-industria, basado en estándares, unificado, de múltiples arquitecturas y de múltiples proveedores que ofrece una experiencia de desarrollador común en todas las arquitecturas de aceleradores, para lograr un rendimiento de las aplicaciones más rápido, más productividad y una mayor innovación.

## ¿Por qué es oneAPI?

La iniciativa **oneAPI** fomenta la colaboración en la especificación oneAPI y las implementaciones compatibles de oneAPI en todo el ecosistema.

Los workloads basados en datos están aumentando exponencialmente. Para ofrecer un alto rendimiento informático para los workloads especializadas emergentes, se requieren diversas arquitecturas informáticas.

Sin embargo, aprovechar múltiples tipos de arquitecturas es un desafío para los desarrolladores. Cada arquitectura ha necesitado diferentes lenguajes y herramientas separadas, y la reutilización del código es limitada. Esto hace que el desarrollo sea complejo, costoso, ineficiente y requiera mucho tiempo. Los desarrolladores necesitan una complejidad de programación reducida para minimizar las barreras, brindando libertad para adoptar nuevo hardware innovador.

En la nueva era XPU, oneAPI es una iniciativa audaz para ofrecer un modelo de programación revolucionario, unificado y entre arquitecturas. Brinda a los desarrolladores la libertad de elegir el mejor hardware para el problema específico que están resolviendo, sin bloqueo de propiedad. Es simplificado, abierto y permite a un desarrollador ofrecer un rendimiento sin concesiones en todas las arquitecturas, al mismo tiempo que integra fácilmente el código existente. Y debido a que oneAPI se basa en estándares y especificaciones abiertas, los riesgos se reducen en la implementación, lo que brinda tranquilidad.

Con oneAPI, los desarrolladores pueden programar usando una única base de código con rendimiento de lenguaje nativo de alto nivel para liberar todo el valor de las capacidades de vanguardia del hardware. Esto permite una ruta rápida para implementar y escalar aplicaciones en arquitecturas aceleradas.

OneAPI incluye un lenguaje de programación directo, Data Parallel C++ y un conjunto de bibliotecas para facilitar el desarrollo entre arquitecturas. Data Parallel C++ se basa en C++, incorpora SYCL de Khronos Group y extensiones de la comunidad. Permite la reutilización de código en arquitecturas y ajustes personalizados para aceleradores. Esto brinda a los desarrolladores la flexibilidad de romper el molde de los enfoques patentados y abre la puerta para aprovechar las opciones de hardware que antes no eran factibles.

Con la programación basada en API, las potentes bibliotecas oneAPI se codifican de forma personalizada para cada arquitectura de destino a fin de acelerar las cargas de trabajo intensivas en computación. Este enfoque unificado simplifica la codificación para aplicaciones de arquitectura cruzada y ofrece rendimiento, al mismo tiempo que aumenta la productividad, ahorra tiempo y reduce los costos y riesgos de desarrollo.

Los desarrolladores pueden concentrarse en innovar nuevas funciones y prepararse para arquitecturas emergentes y futuras, sin reescribir el código para la próxima plataforma de hardware.

OneAPI, el camino inteligente hacia la libertad para la computación acelerada de las cargas económicas y técnicas de los modelos de programación propietarios.

## Test drive local o en la nube

- https://software.intel.com/devcloud/oneapi oneAPI en la nube de Intel
- https://software.intel.com/oneapi: Descargar oneAPI

Es posible probar oneAPI con un conjunto de kits de herramientas en Intel DevCloud, un espacio aislado gratuito para desarrollar, probar y ejecutar cargas de trabajo en un clúster del último hardware y software de Intel o descargue los kits de herramientas de Intel oneAPI para ejecutar localmente.

## Aplicaciones y proyectos

https://www.intel.com/content/www/us/en/developer/tools/oneapi/application-catalog/full-catalog/overview.html?s=Newest

https://devmesh.intel.com/projects?query=oneapi

## Biblioteca de Conocimiento

https://www.intel.com/content/www/us/en/developer/tools/oneapi/tech-articles-how-to/library.html?s=Newest#gs.j3ch14

## Actualizaciones

https://www.intel.com/content/www/us/en/developer/articles/news/oneapi-news-updates.html#gs.j3ch5t

## Eventos y Entrenamientos de oneAPI

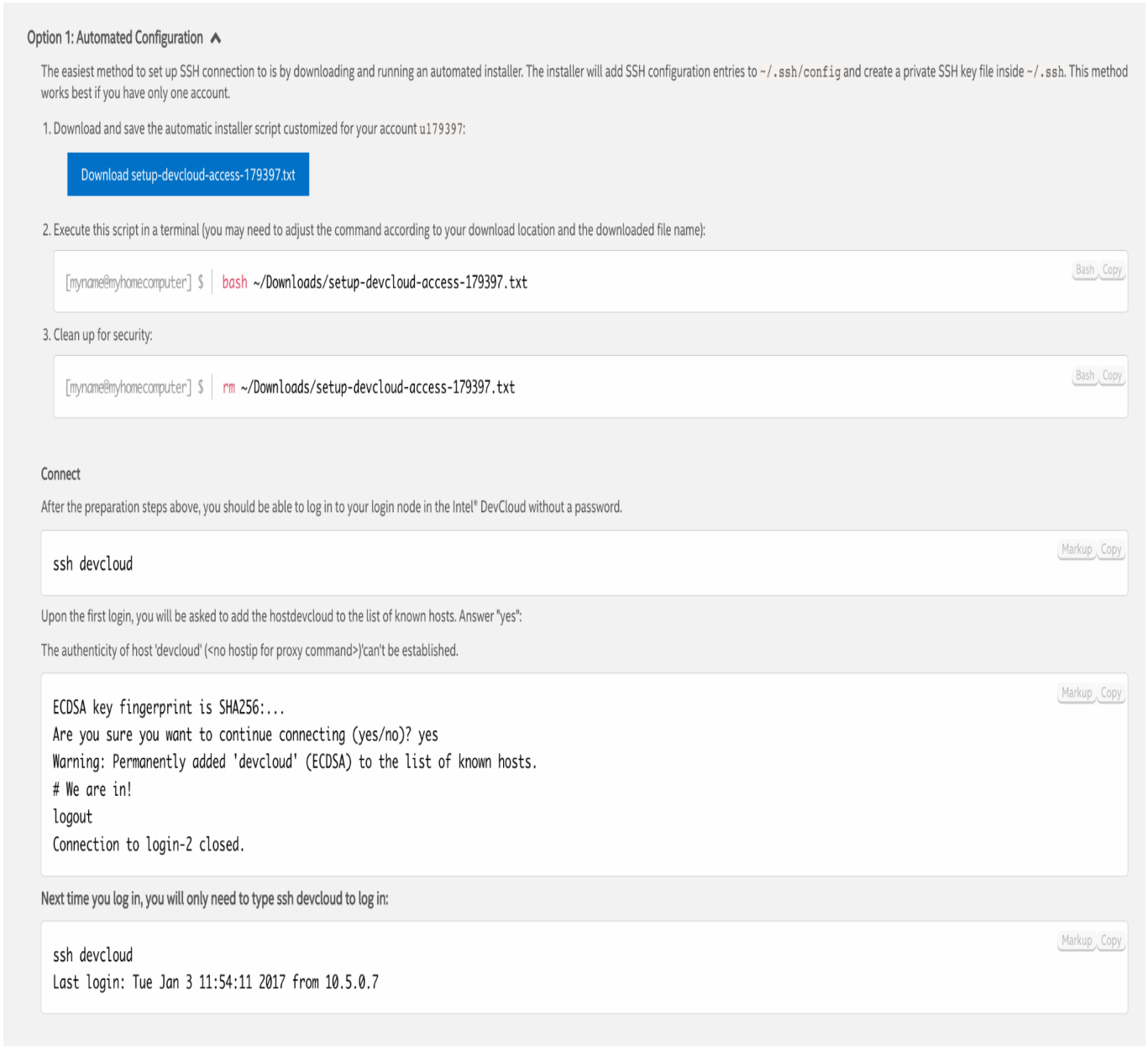
https://software.seek.intel.com/oneapi-training-calendar#gs.j3ch1k

## Correr un Hello World en  Intel DevCloud

**Intel® oneAPI HPC Toolkit:** Ofrezca aplicaciones C++, Fortran, OpenMP* y MPI rápidas y escalables.

https://devcloud.intel.com/oneapi/get_started/hpcToolkitSamples/

1. **Conectarse**



Option 1: Automated Configuration ^

The easiest method to set up SSH connection to is by downloading and running an automated installer. The installer will add SSH configuration entries to `~/.ssh/config` and create a private SSH key file inside `~/.ssh`. This method works best if you have only one account.

1. Download and save the automatic installer script customized for your account `u179397`:

Download setup-devcloud-access-179397.txt

2. Execute this script in a terminal (you may need to adjust the command according to your download location and the downloaded file name):

```
[myname@myhomecomputer] $ bash ~/Downloads/setup-devcloud-access-179397.txt
```

Bash Copy

3. Clean up for security:

```
[myname@myhomecomputer] $ rm ~/Downloads/setup-devcloud-access-179397.txt
```

Bash Copy

Connect

After the preparation steps above, you should be able to log in to your login node in the Intel® DevCloud without a password.

```
ssh devcloud
```

Markup Copy

Upon the first login, you will be asked to add the hostdevcloud to the list of known hosts. Answer "yes":

The authenticity of host 'devcloud' (<no hostip for proxy command>)'can't be established.

```
ECDSA key fingerprint is SHA256:...
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'devcloud' (ECDSA) to the list of known hosts.
# We are in!
logout
Connection to login-2 closed.
```

Markup Copy

Next time you log in, you will only need to type ssh devcloud to log in:

```
ssh devcloud
Last login: Tue Jan 3 11:54:11 2017 from 10.5.0.7
```

Markup Copy

## 2. Compilar y correr

## 2 Run a Hello World! Sample

Use this simple sample to confirm that you are connected to oneAPI DevCloud

The following will help you to get started with the Intel® oneAPI HPC Toolkit on the Intel® DevCloud through a basic sample, *matrix-mul*. This sample takes advantage of CPU and GPU accelerators using Intel® DPC++, OpenMP* and Intel® Math Kernel Library (Intel® MKL).

### 2.1. Matrix Multiplication sample walkthrough

1. Connect to the DevCloud.

```
[myname@myhomecomputer] $ ssh devcloud
```

2. Download the samples.

```
[u179397@login-2] $ git clone https://github.com/oneapi-src/oneAPI-samples.git
```

3. Go to the matrix_mul sample.

```
[u179397@login-2] $ cd ~/oneAPI-samples/DirectProgramming/DPC++/DenseLinearAlgebra/matrix_mul
```

### 2.2. Build and run the sample in batch mode

The following describes the process of submitting build and run jobs to PBS.

A job is a script that is submitted to PBS through the `qsub` utility. By default, the `qsub` utility does not inherit the current environment variables or your current working directory. For this reason, it is necessary to submit jobs as scripts that handle the setup of the environment variables. In order to address the working directory issue, you can either use absolute paths or pass the `-d <dir>` option to `qsub` to set the working directory.

#### Create the job scripts

1. Create a `build.sh` script with the following contents.

```
#!/bin/bash
source /opt/intel/inteloneapi/setvars.sh > /dev/null 2>&1
make build_dpcpp
make build_omp
```

2. Create a `run.sh` script with the following contents for executing the sample.

```
#!/bin/bash
source /opt/intel/inteloneapi/setvars.sh > /dev/null 2>&1
make run_dpcpp
make run_omp
```

**Build and run**

Jobs submitted in batch mode are placed in a queue waiting for the necessary resources (compute nodes) to become available. The jobs will be executed on a first come basis on the first available node(s) having the requested property or label.

1. Build the sample on a gpu node.

```
[u179397@login-2] $ qsub -l nodes=1:gpu:ppn=2 -d . build.sh
```

**Note**: `-l nodes=1:gpu:ppn=2` (lower case L) is used to assign one full GPU node to the job.
**Note**: The `-d .` is used to configure the current folder as the working directory for the task.

2. In order to inspect the job progress, use the qstat utility.

```
[u179397@login-2] $ watch -n 1 qstat -n -1
```

**Note**: The `watch -n 1` command is used to run `qstat -n -1` and display its results every second.

3. Run the sample on a gpu node after the build job completes successfully.

```
[u179397@login-2] $ qsub -l nodes=1:gpu:ppn=2 -d . run.sh
```

4. Inspect the output of the sample.

```
[u179397@login-2] $ cat run.sh.eXXXX
[u179397@login-2] $ cat run.sh.oXXXX
```

Here `XXXX` is the job ID, which gets printed to the screen after each `qsub` command.

5. Remove the stdout and stderr files and clean-up the project files.

```
[u179397@login-2] $ rm build.sh.*; rm run.sh.*; make clean
```

6. Disconnect from the Intel DevCloud.

```
[u179397@login-2] $ exit
```

**2.3. Build and run additional samples**

Intel® oneAPI HPC Toolkit includes several sample programs, many of which can be compiled and run in a similar fashion to *matrix-mul*. Experiment with running the various samples on different kinds of compute nodes or adjust their source code to experiment with different workloads. The next sample we recommend is *Nbody*.