**Hand on experience - Intel DevCloud**

**Goal:**

In this Lab, you will learn how to connect to DevCloud, and how to access to FPGA hosted in cloud, how to run on FPGA, and how to analyze the performance on FPGA.

**Who can use Intel DevCloud?**

Developers, data scientists, professors, students, start-ups and others can request access for full access to the latest Intel CPUs, GPUs, and FPGAs, Intel oneAPI Toolkits, and the new programming language, Data Parallel C++ (DPC++). Access is free for 120 days with the possibility of an extension.

Check this link to request access:

<https://software.intel.com/en-us/devcloud/datacenter>

**What Is Intel Devcloud?**

To deliver high compute performance for emerging workloads, we need diverse compute architectures like - CPU, GPU, and FPGA accelerators. Intel DevCloud is a cluster (a group of servers connected to each other and work together to combine computing power as if they are a single) of Intel® Xeon® Scalable Processors connected to GPUs and FGAs. However, taking advantage of multiple types of architectures can be a challenge for developers as separate tools are required for each architecture and code reuse is limited. Bur do not worry, DevCloud will assist you with your compute intensive tasks. It provides access to precompiled software optimized for Intel® architecture. To reduce programming complexity and minimize the barriers to adopt new innovative hardware technology, Intel provided a unified, cross-architecture programming model called oneAPI. oneAPI toolkit is available on the DevCloud.



For further information, check the following video:

<https://www.youtube.com/watch?v=6lWm3-85RbQ&ab_channel=IntelSoftware>

**Architecture of DevCloud:**

The cluster consists of multiple compute servers, which we call compute nodes, storage servers, and the login node.

A computer screen shot of a computer

Description automatically generated

There are two way to access the DevCloud: You can run a Jupyter Notebook session directly on one of the compute nodes or you can access the login node using an SSH client in a text-based terminal. In both cases, if you want to use the full power of the cluster, you will need to submit jobs to a queue.

Your Jupyter Notebook instance runs on one of the compute nodes. When you execute the Notebook cells, they run on an Intel Xeon Scalable processor. There may be other people connecting to your compute node, which reduces the available compute power. However, you can reserve a full node for your job and even use multiple nodes at once by submitting scripts to the job queue.

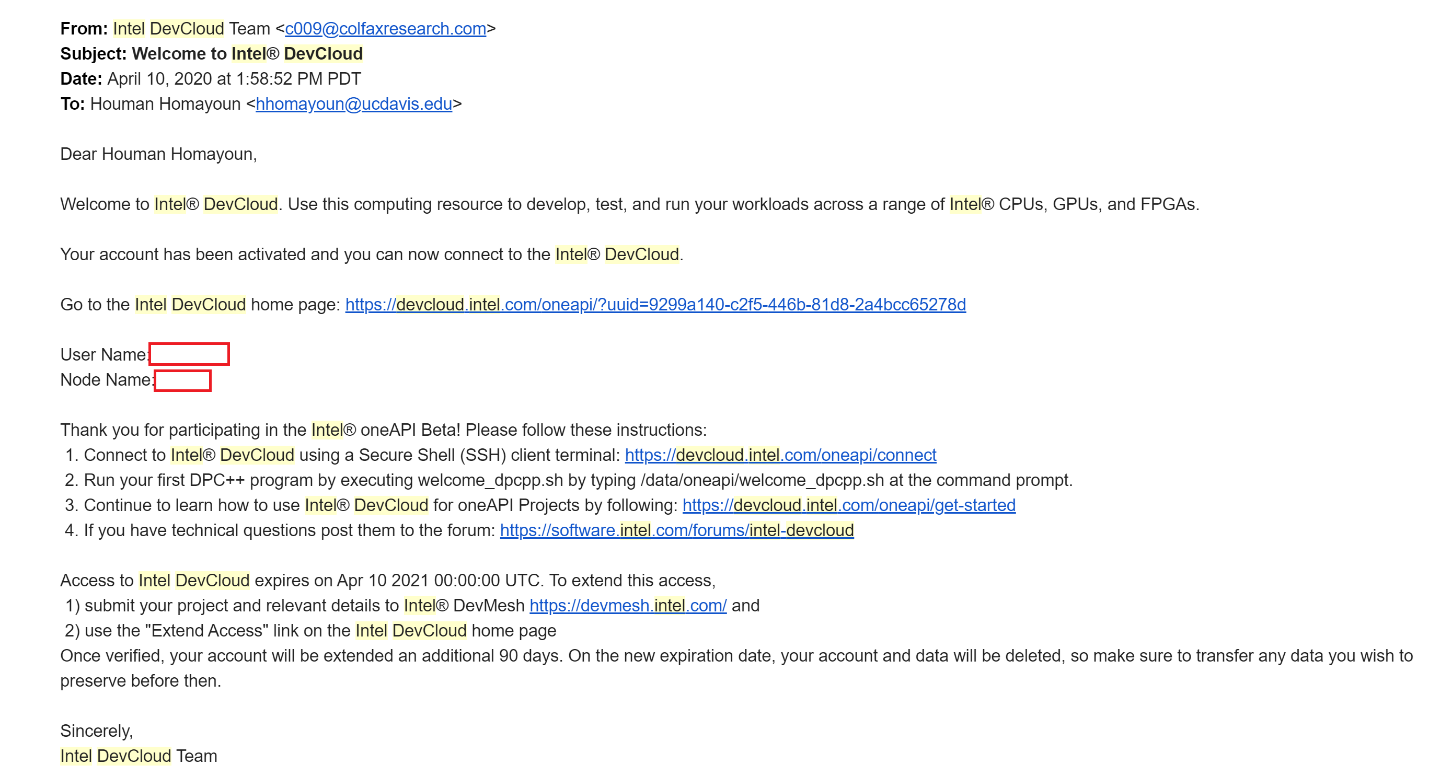
When you access the Intel DevCloud through SSH, you will be connected to a login node with hostname login-1. This node is intended only for code development and compilation, but NOT for computation. That is because it does not have much compute power, and, additionally, there are limitations on CPU time and RAM usage on the login node; your workload will be killed if it exceeds the limit. Therefore, on this node you can edit code and compile applications. However, to run computational applications, you will submit jobs to a queue for execution on compute nodes. In this manner, you are sharing computing resources with your peers, however, other users cannot see your data or applications.

For more information, check this link:

<https://devcloud.intel.com/oneapi/get_started/baseToolkitSamples/>

**Let us get connected to DevCloud:**

After you signup successfully for DevCloud, you will receive your credentials to your registered email address. Do not lose this email.



By clicking on the Intel DevCloud home page, you will be redirected to the following page:

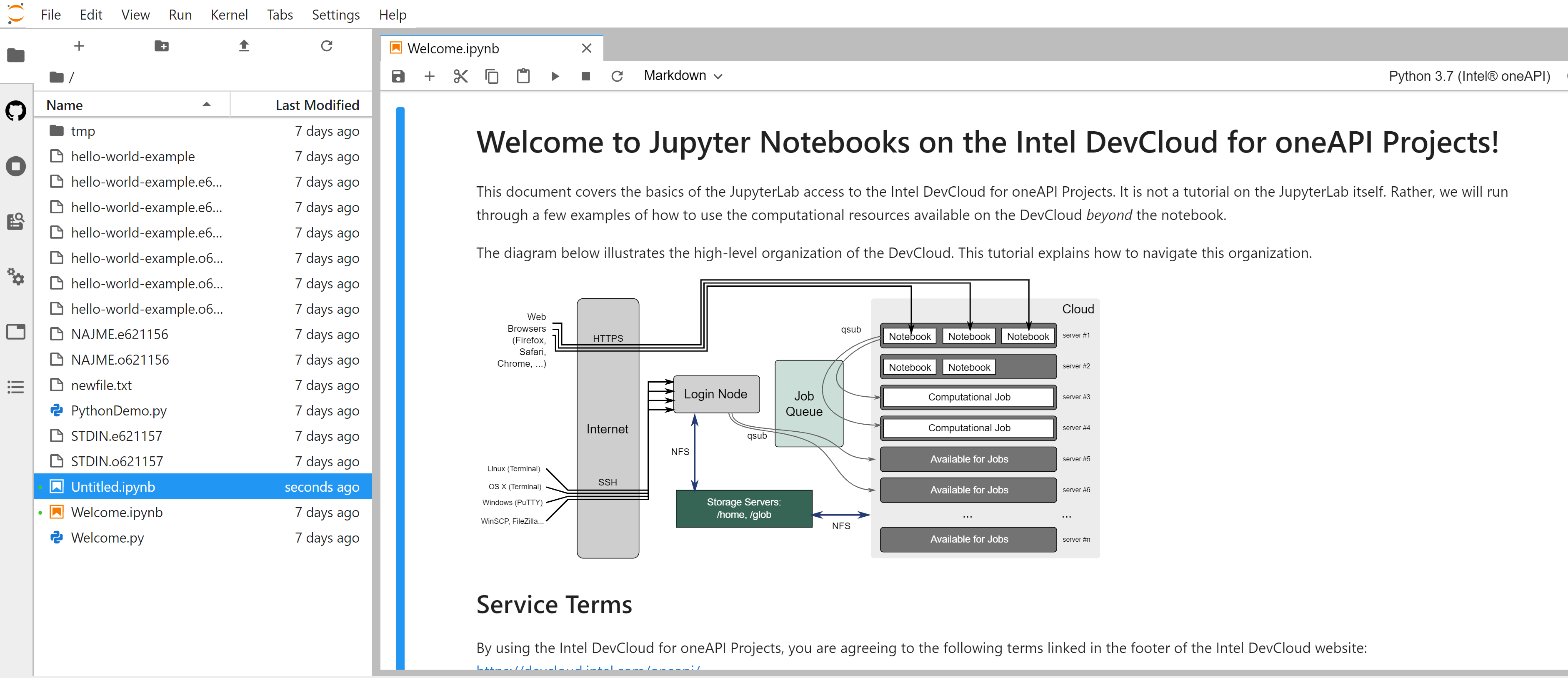
<https://devcloud.intel.com/oneapi/get_started/>

At the end of page, you can select Jupyter notebook. First, let’s look at connecting with a Jupyer notebook:

A screenshot of a computer

Description automatically generated

After clicking on one-click login, wait for a moment and let the Jupyter Lab environment to come up. Afterwards, you can see the interface like below:

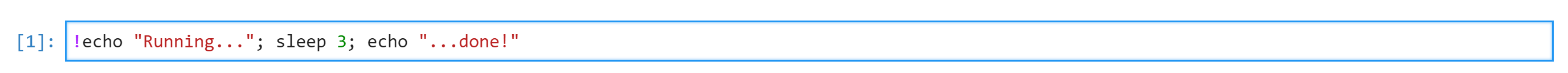


There is Welcom.ipynb in your home folder. It is a very helpful tutorial to get familiar with Jupyter Notebook in DevCLoud. Next section of this manual is from the Welcome notebook.

**Jupyter Notebook Basics:**

You can find detailed documentation on using the JupyterLab software at jupyter.org. For our tutorial, you just need to know that

1. When you see cells like below (the line that begins with !echo "Running..."), this is code that you can run.
2. If you mouse-click on the cell, you will be able to edit the code.
3. While you are in the cell, press Ctrl+Enter, and the code in the cell will run.
4. In the top-right corner of the page, the indicator will change to . This means that the kernel is busy.
5. If the code begins with "!", it will run in the Bash shell. Otherwise, it is treated as Python code.



**What is a Job:**

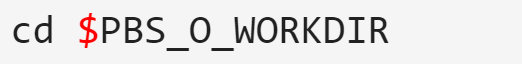
Anything that we want to run on DevCloud is a job such as training a neural network, compiling C++ code, and running an application.

**Basic Job Submission:**

The job queue is the only method for accessing the full capacity of the computing resources available on the DevCloud. This section explains how you can interact with the queue from the JupyterLab environment. To submit a job, we need to create **Job Script** in Bash first.

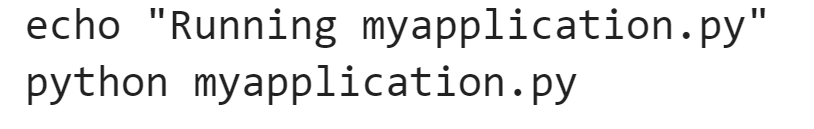
* Creating a Job Script:

The first line of script should contain the following code:



The line cd $PBS\_O\_WORKDIR changes the working directory to the directory where the script is located. The rest of script can be a regular Bash script. In your script, remember to have an empty line at the end of the file; otherwise the last command will not run.

Note that only Bash job scripts are supported. If you need to run a Python application, you can add the corresponding Python launch line to your Bash job script. For example:



* Submitting a Job to the Queue:

You can submit your job script using the *qsub* command as follow:

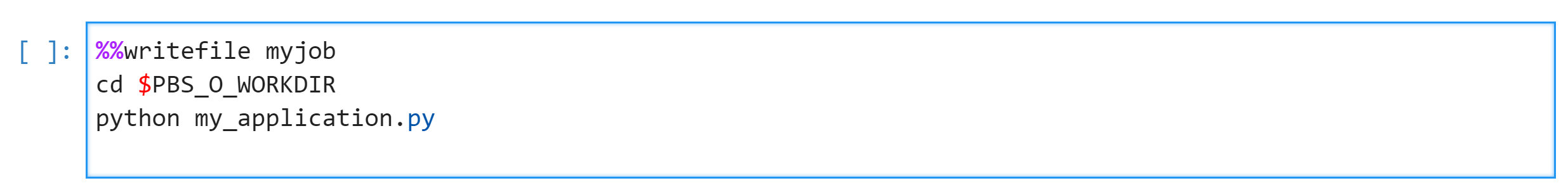
!qsub <name of your job script>

* Example:

As we showed, submitting a job can be done through a job script file. Suppose that you have a Python application, ‘my\_application.py’. Suppose that your application

1. Imports some Python modules,
2. Loads a dataset,
3. Sets up a neural network
4. Trains it, and
5. Writes the resultant model weights into a file.

Now, we want to submit this Python application as a job to the queue. In the same folder, use your favorite text editor and create a script file “myjob”. Or you can use *%%writefile <name of your file>* to create a new file in Jupyter cell. Then add the following two lines:



*%% writefile* command creates a file with name that you provide, and then writes all the lines after the command into the file. So, after running above commands, a script file (myjob) will be created that contains the next two lines.

The second line of above cell ensures that the script runs in the same directory as where you have submitted it. Third line runs the Python application. Now, you can submit this job with:

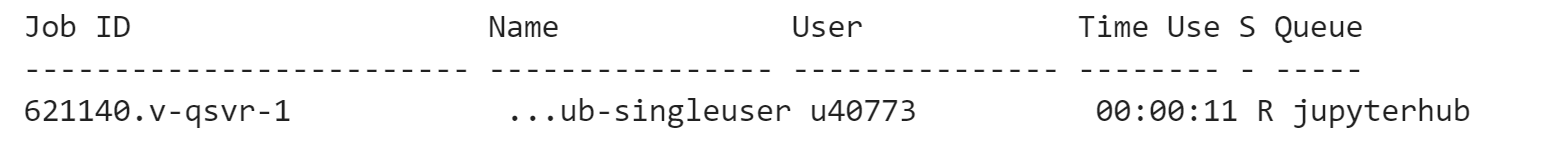
*!qsub myjob*

Now, you have submitted a job to the queue. You should see an output line that looks like "[numbers].cXXX". The number you see in the front is the Job ID. We will be using this number to retrieve the output of the job.

* Checking the Queue Status:

Once the job has been placed in the queue, you can find the current status of the job by running the following command in a cell:

*!qstat*



In the column S you will see a letter indicating its status: "Q" is for "queued", "R" is for "running", and "E" is either an error, or a transition to a normal job completion.

* Getting the result:

Once the job is completed, the resulting output and error streams (stdout and stderr) are placed in two separate text files. These output files have the following naming convention:

* stdout: [Job Name].o[Job ID]. Example: myjob.o12345
* stderr: [Job Name].e[Job ID]. Example: myjob.e12345

[Job Name] is either the script name, or a custom name — for example, the name specified by the -N parameter of *qsub*.

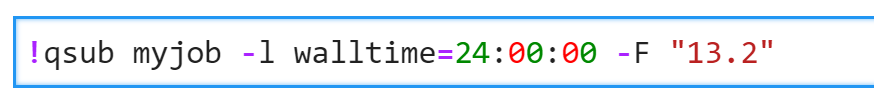
[Job ID] is the number you got from the output of the *qsub* command.

Using *%cat* command, you can see the content of a file:



* qsub parameters:

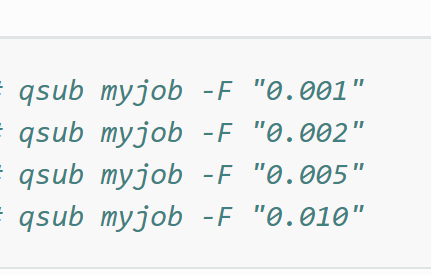
*!qsub <file>* command can take a variety of parameters that you can set. For example, the following command requests a wall clock time limit of 24 hours and passes a command line argument equal to "13.2" to the job.



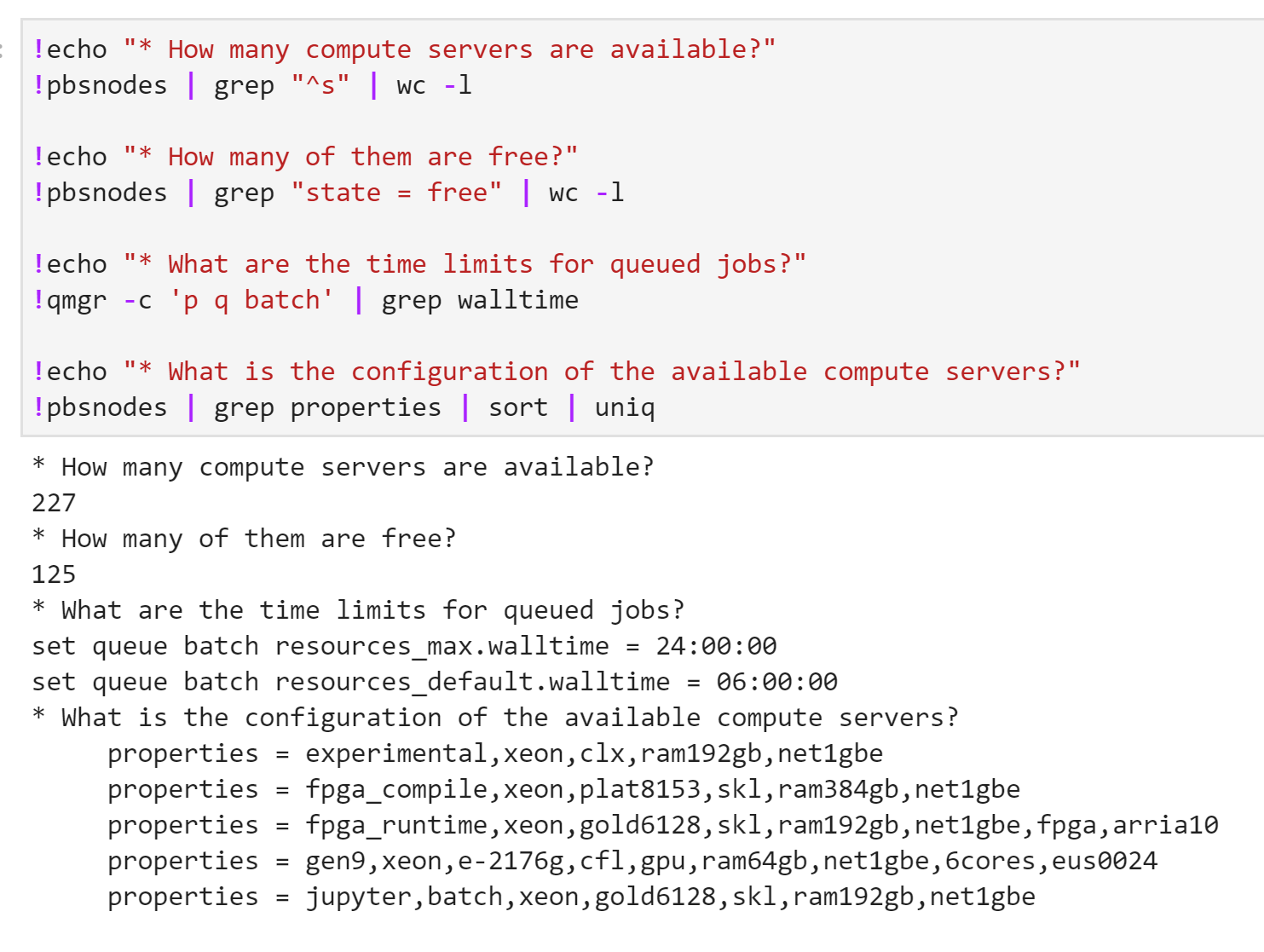
List of parameters is available at:

<https://devcloud.intel.com/datacenter/learn/advanced-queue/job-parameters>

You can submit several jobs at once such as follow:

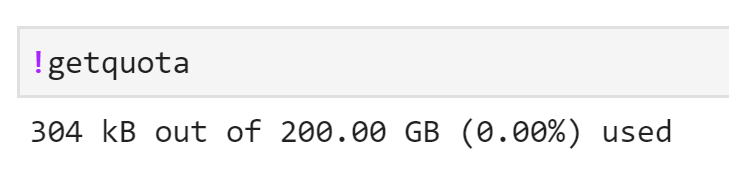


If enough compute servers are available, all jobs will run simultaneously. Otherwise, they will stay in the queue waiting for their turn to run. When you submit a lot of jobs, be aware that the queue has a fair share-based scheduling policy, so the more you run, the more often will your jobs yield to other users' calculations. You can learn about the pool of compute servers available for your jobs by running the commands below:



**Data Management:**

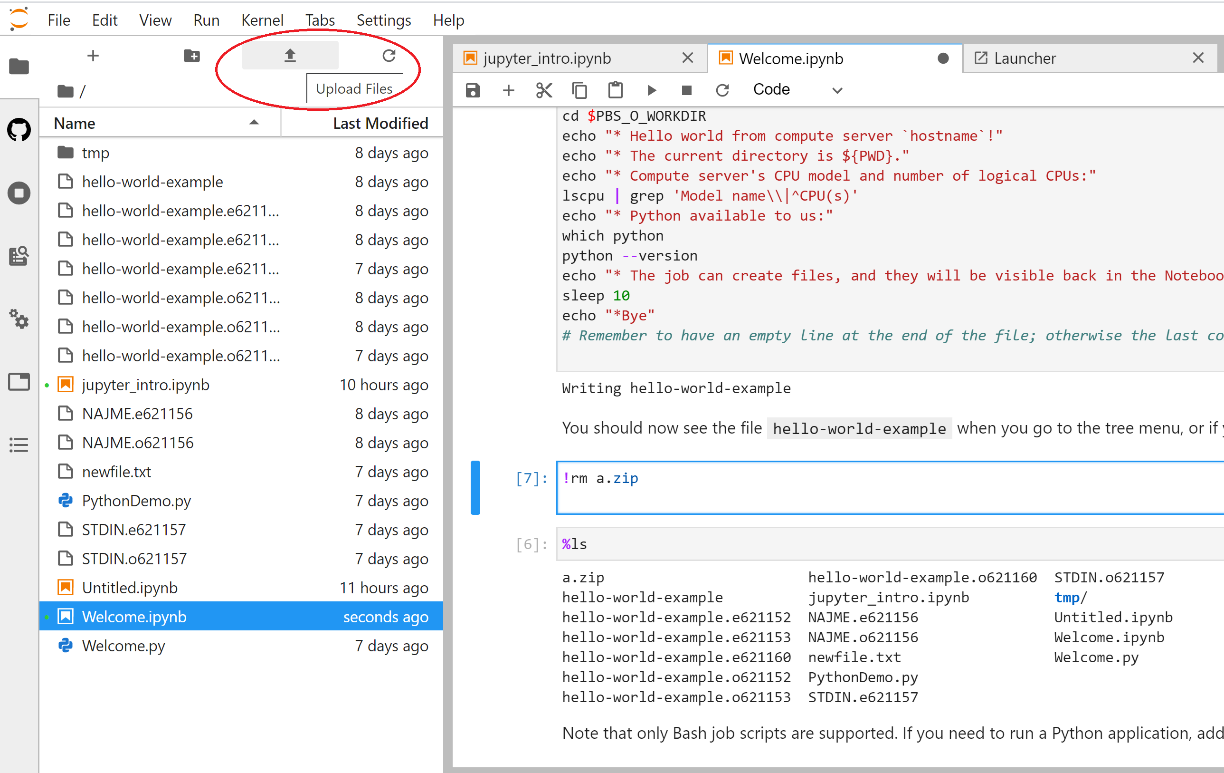
When you connect through the Jupyter notebook, you will see folder in your home directory named /*tmp*. Do not use this folder. Be aware that Home folder is NFS-shared between the login node and the compute nodes. Your account on DevCloud will have a storage quota. You can view your current usage with the following command:



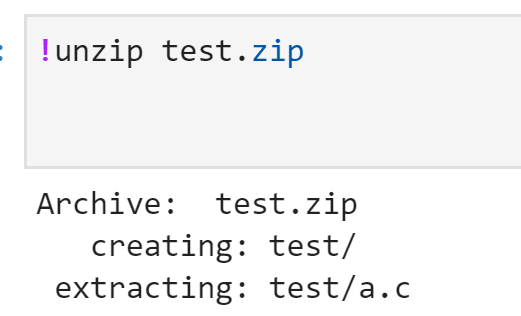
You are free to write data up to this limit but note that if you reach this quota your applications will no longer be able to write any more data to your home directory.

* Transferring Data between your local machine and DevCloud:

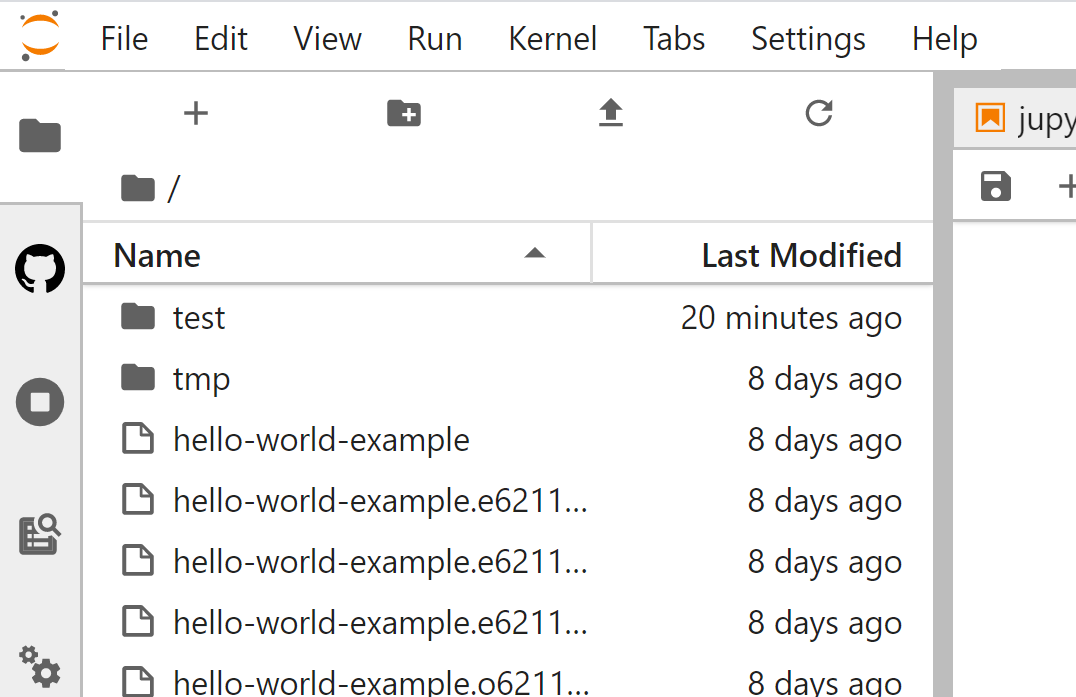
You may ask, how you can upload your data and source codes from your local system to the DevCloud environment or how to download your outputs and results from DevCloud to your local machine. In Jupyter, this is an easy task. You can zip your working directory in your local machine and upload it to Jupyter environment as follow (the red circle):



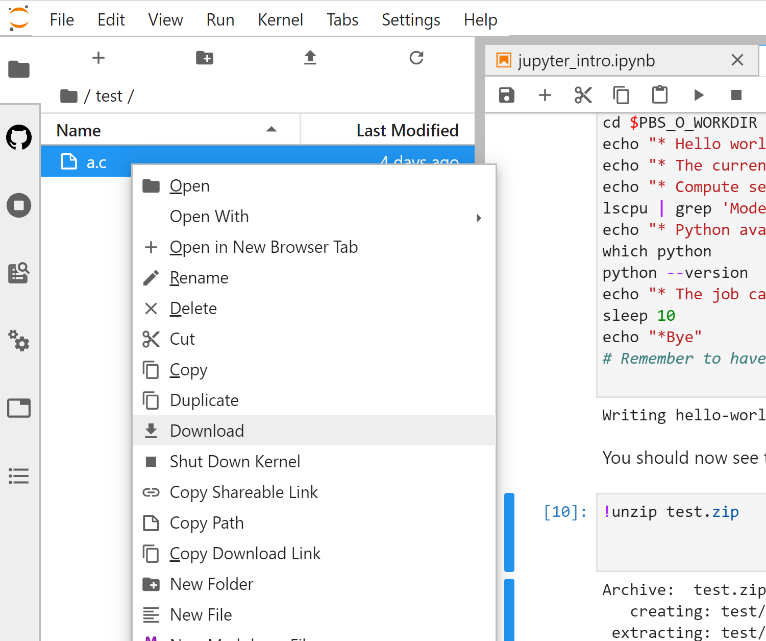
As an example, we upload a test.zip to the DevCloud and then we unzip it with following command:



Now you can see the /test directory in the environment:



Now, let’s download the content of /test directory (a.c file). For this purpose, you can right click on the file and select download. It will download the file to your local machine.

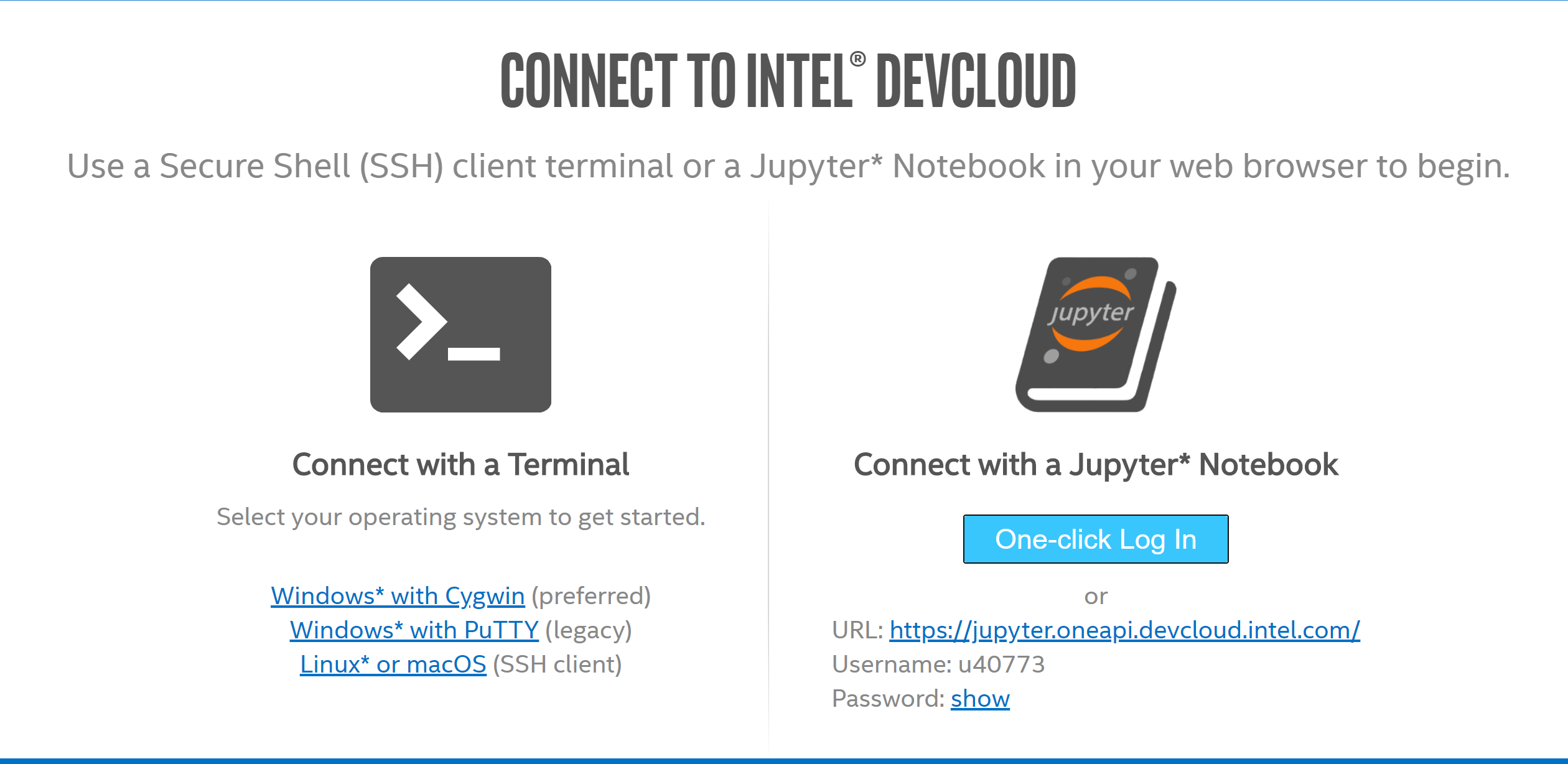


More information is available at:

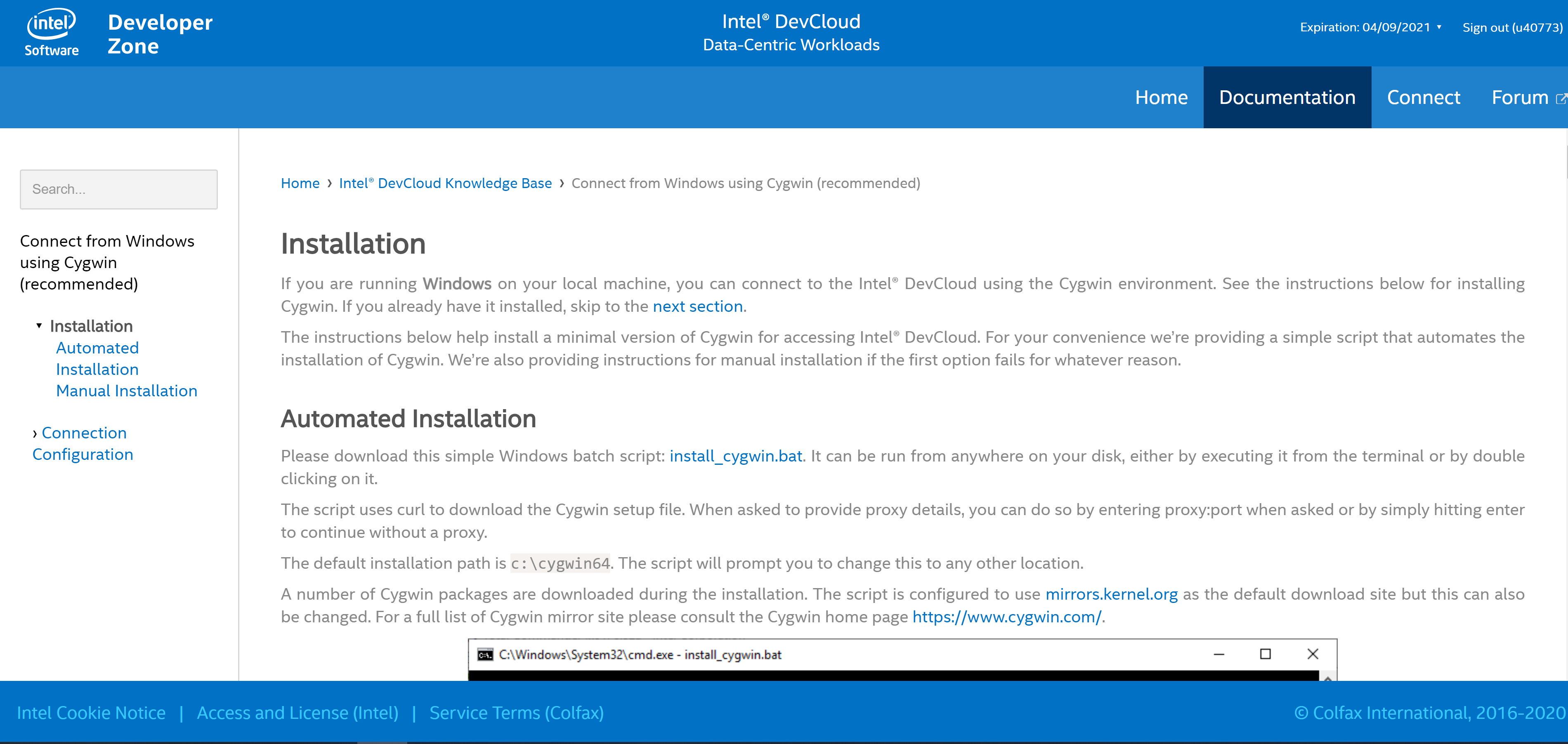
<https://www.youtube.com/watch?v=aFnC_u0qdz0>

**Connecting Through SSH:**

Now let’s back to connecting to DevCloud using SSH in terminal.



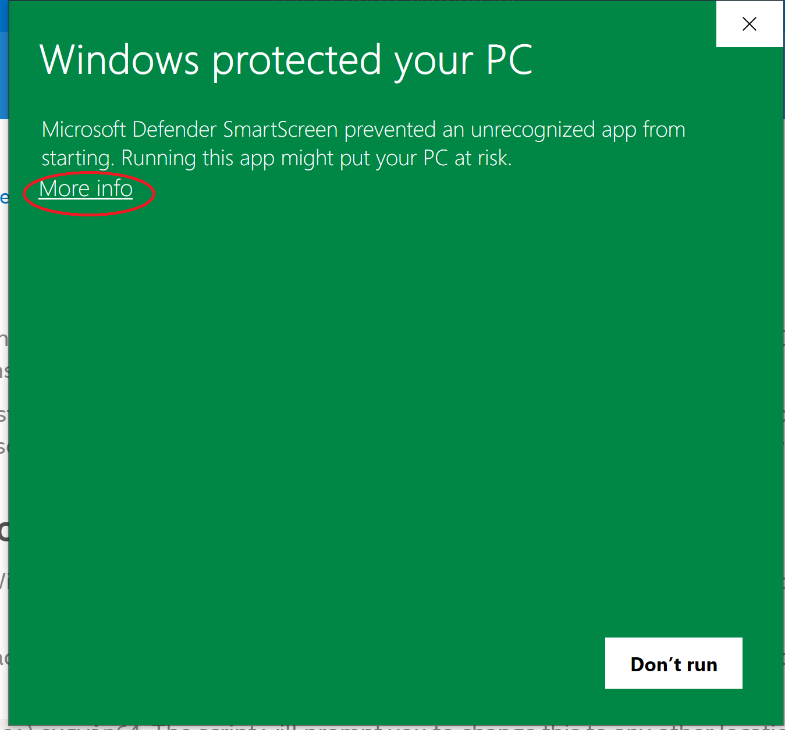
If you are a windows user, the best way is to use Cygwin. Cygwin will provide a Linux like environment in Windows. Therefore, in the Cygwin terminal, you can type Linux commands. This time, click on Windows\* with Cygwin. You will be redirected to the following page:



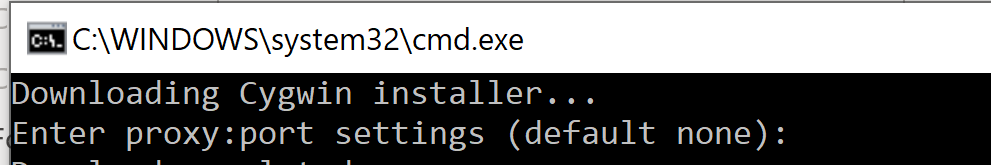
We recommend you to follow the instructions from this link:

<https://devcloud.intel.com/oneapi/documentation/connect-with-ssh-windows-openssh/>

However, we put some information here for you. For the first time, please download the script by clicking on install\_Cygwin in the automated installation and run the script. You will see a Window like this:

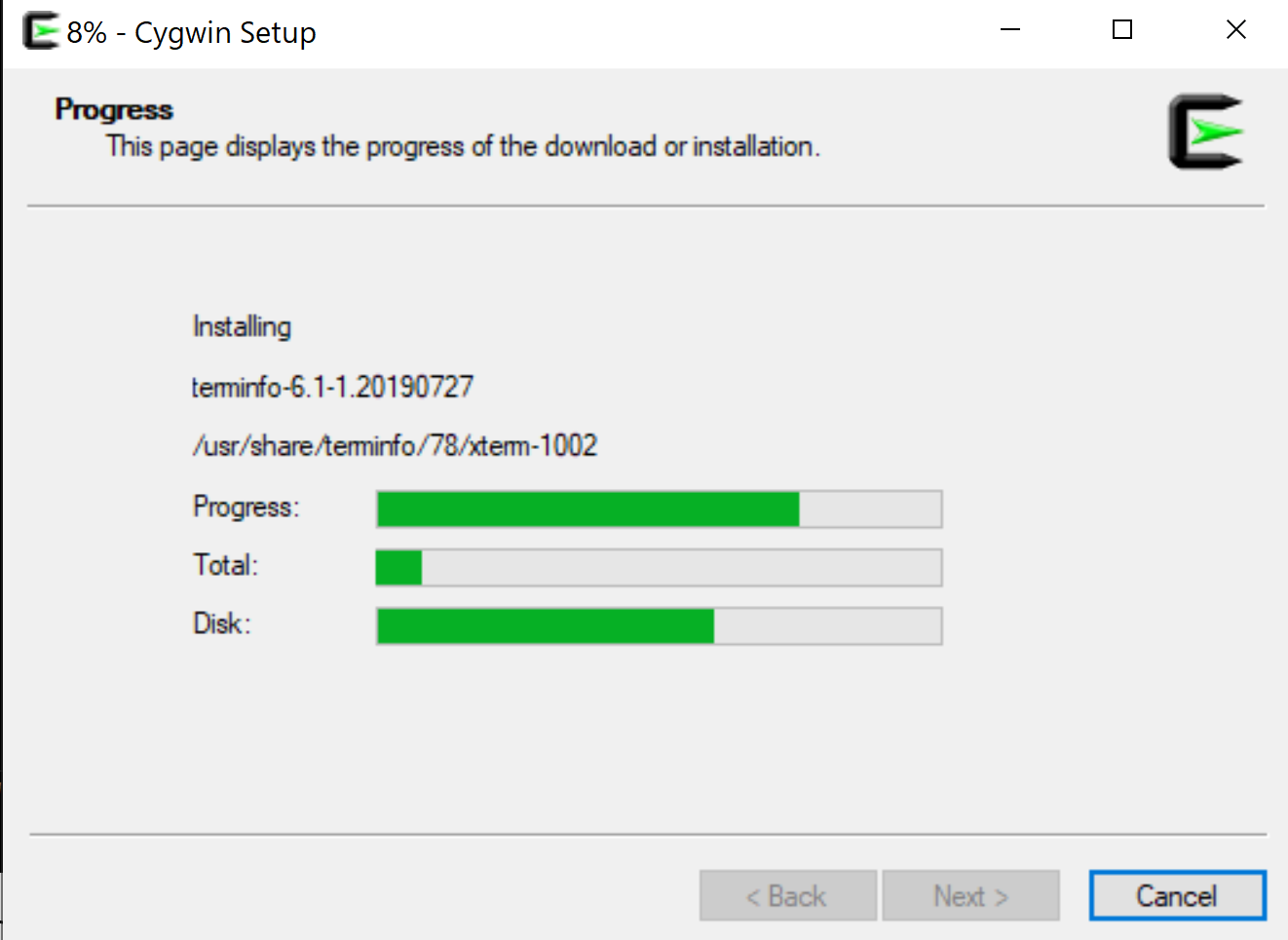


Select more info, and then select Run anyway button. After that, a command prompt will appear. When asks for proxy, just press enter:

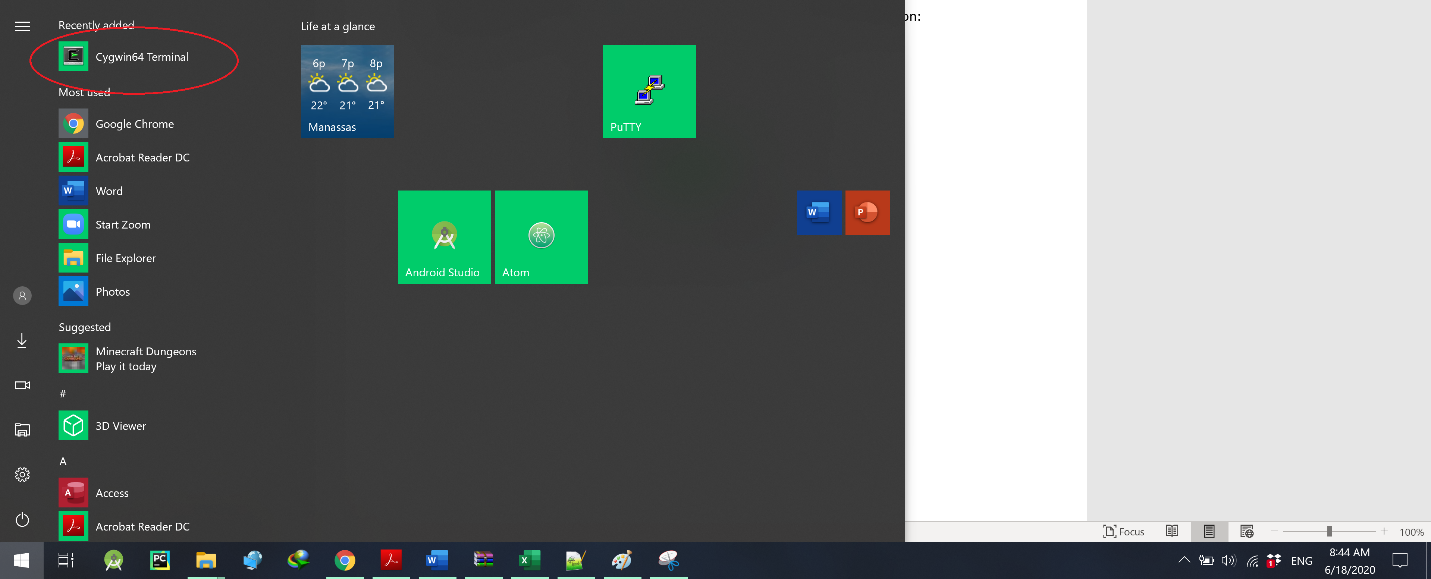


After that, select the address for installation or just press enter for default location:

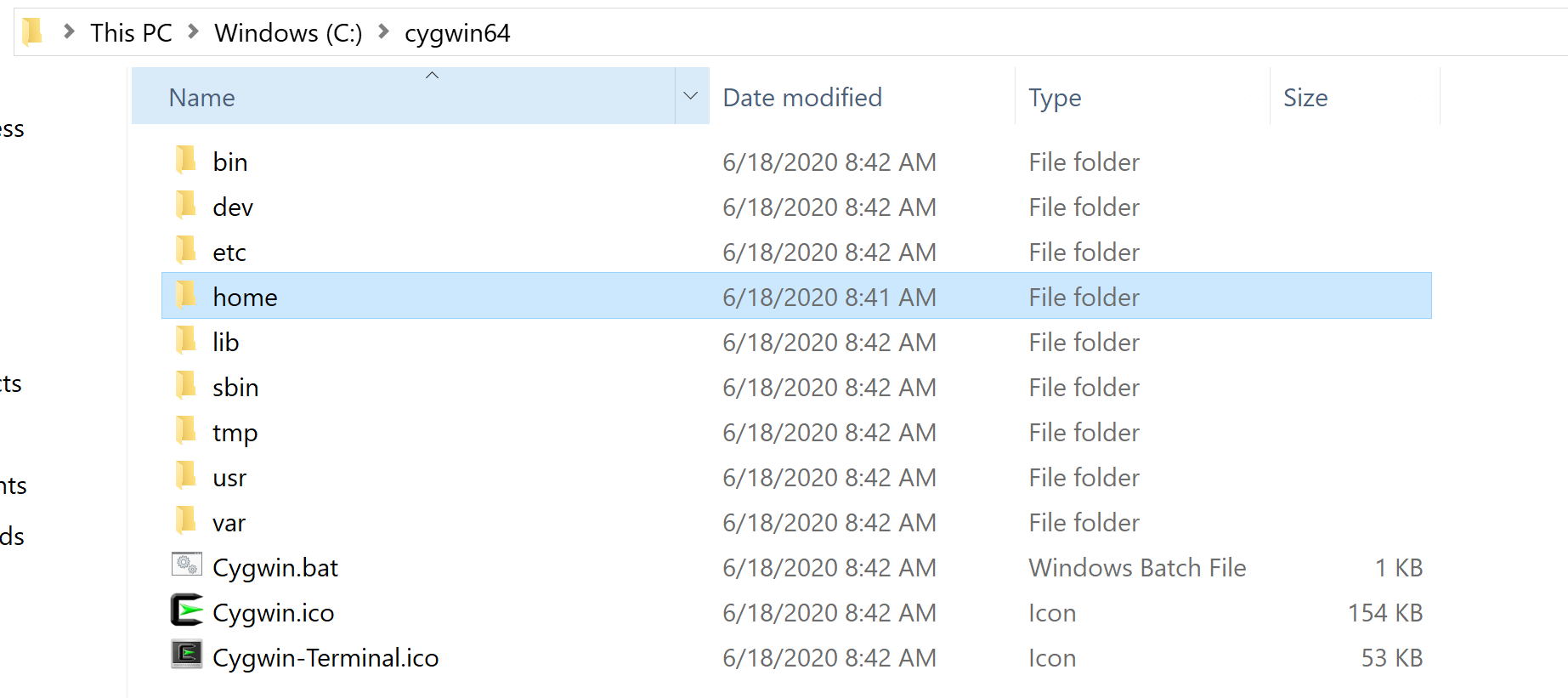
It starts to install:



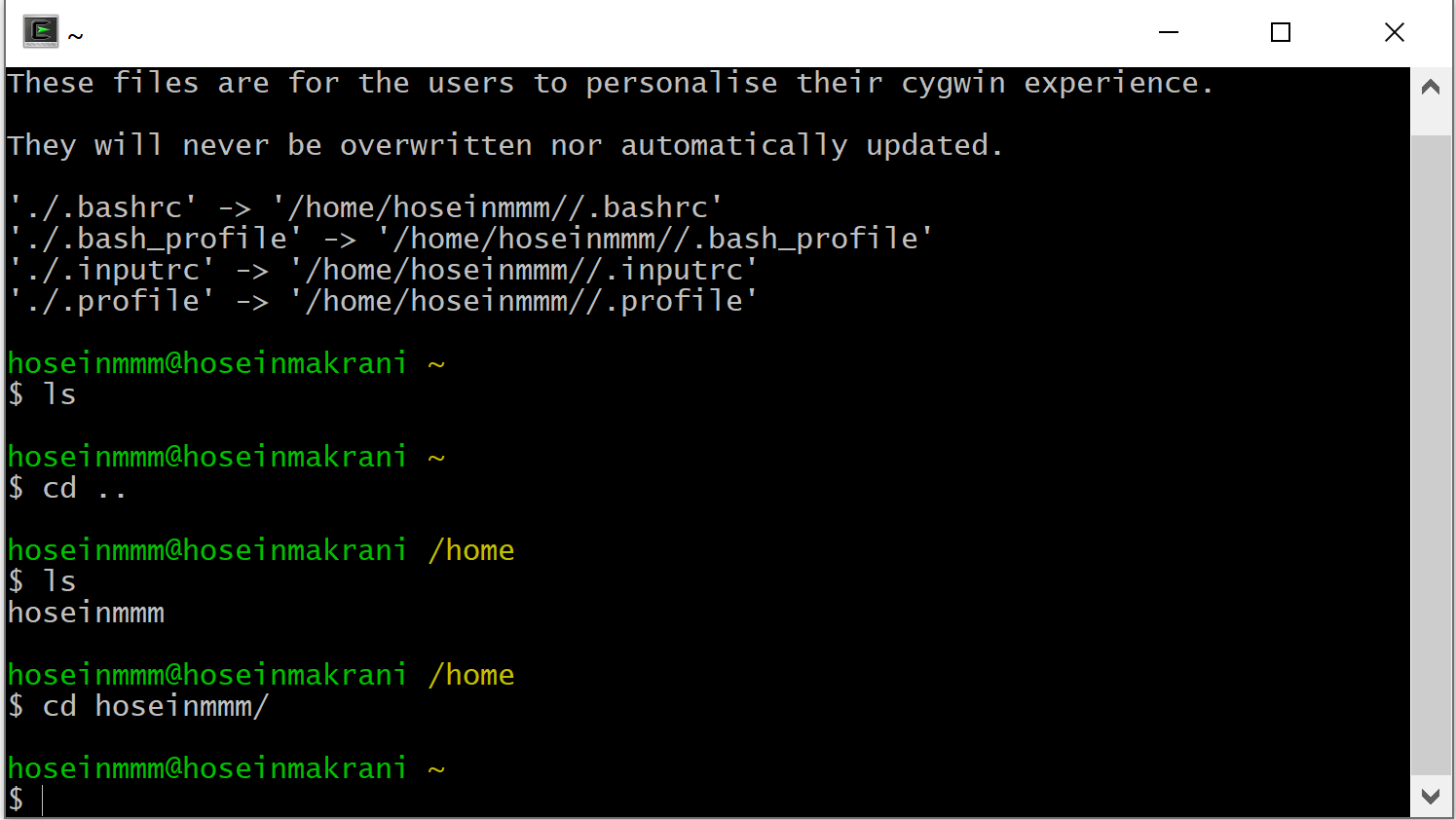
Then click on start button to see the Cygwin icon and run it:



When the installation of Cygwin is done, you will see the following directories in the installation location and it is similar to the Linux environment:



Now, let’s configure the connection. First lunch Cygwin, you have access to a Linux-like environment.

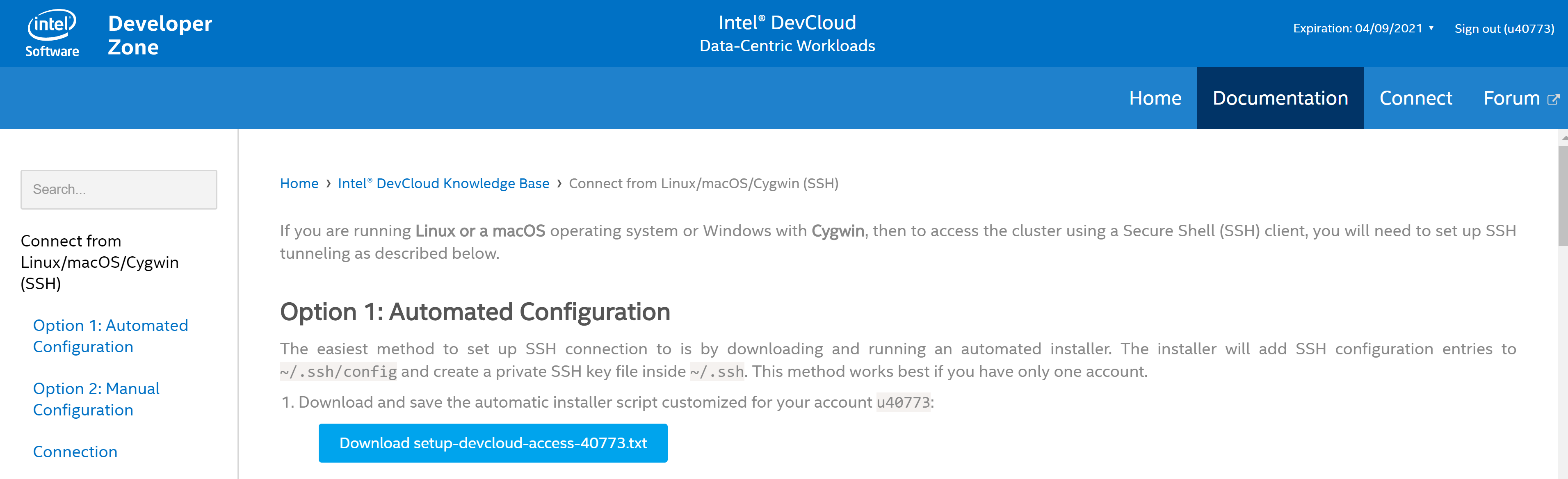


Therefore, to finish the setup of connection to the Intel® DevCloud, proceed to the article Connect from Linux/macOS/Cygwin (SSH):

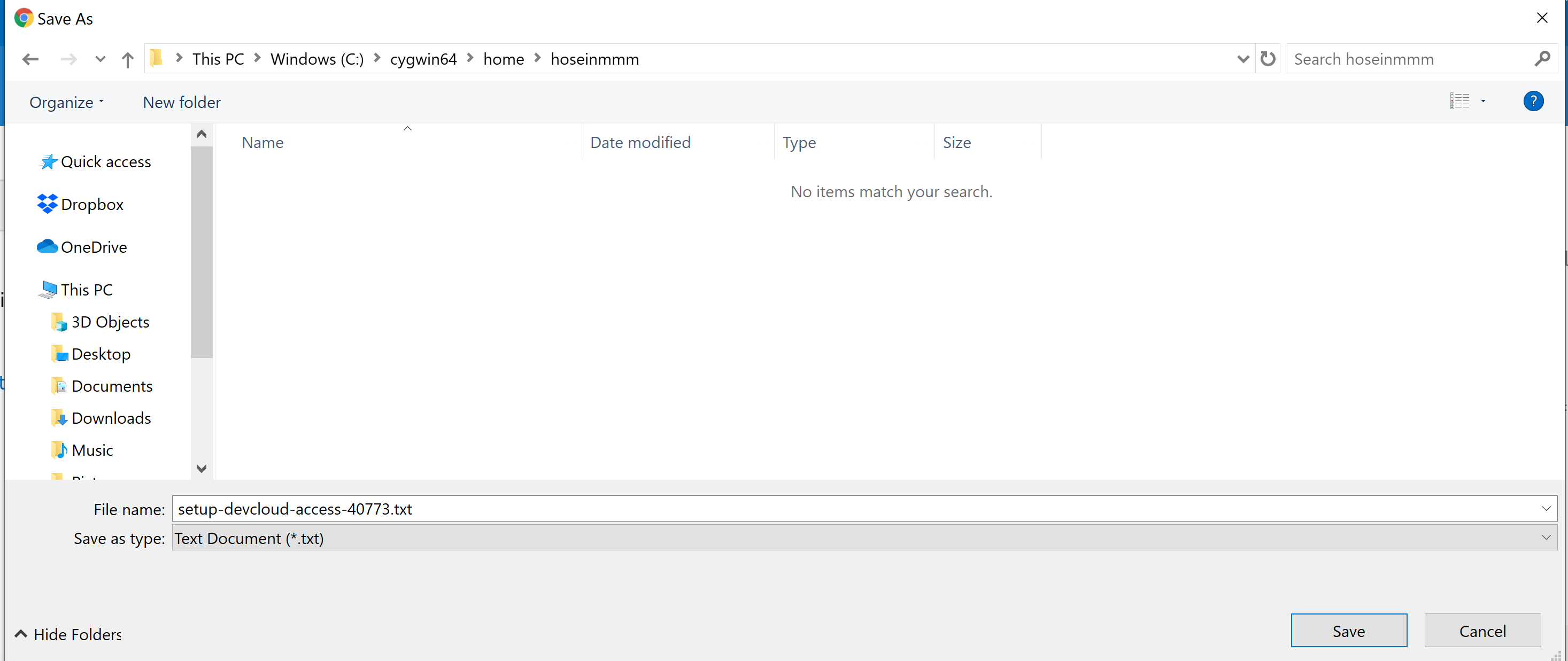
<https://devcloud.intel.com/datacenter/learn/connect-with-ssh-linux-macos/>

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. Follow the instruction on website one by one:

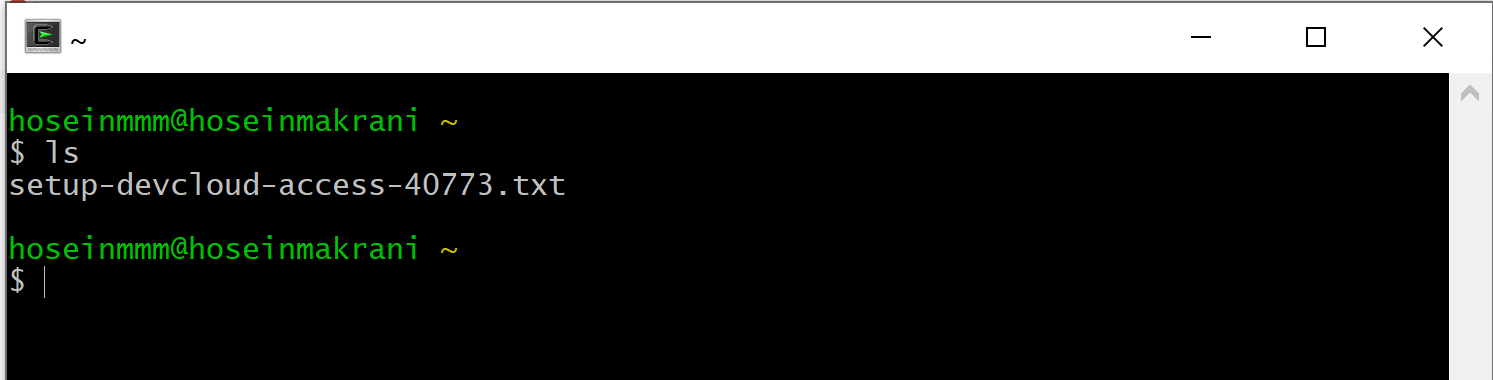
1. Download and save the automatic installer script customized for your account:



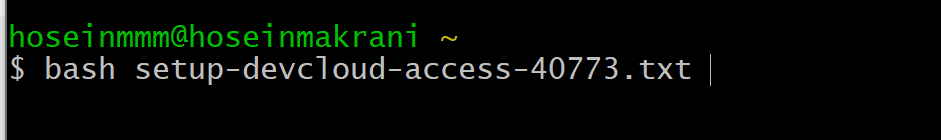
Note: save it to the installation directory of your Cygwin, inside your home directory as you can access it easily:



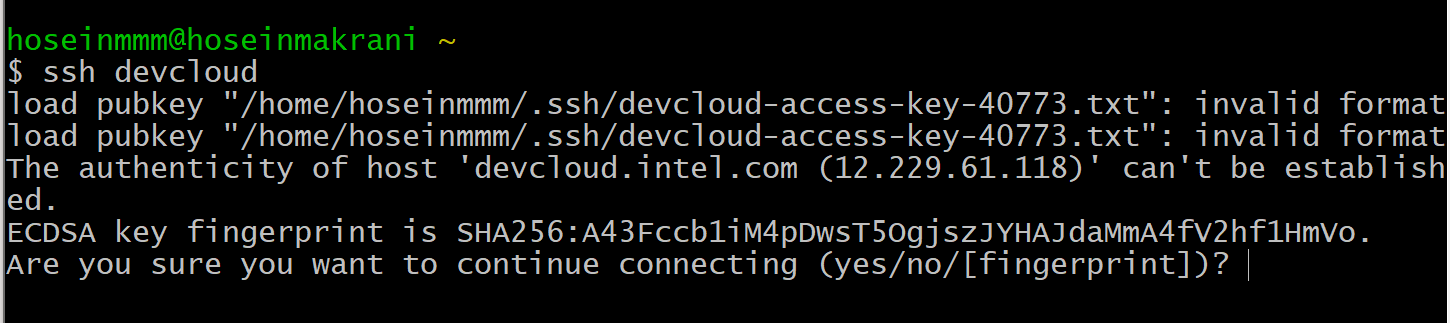
1. Now, if you run *ls* command in Cygwin, you will see it:



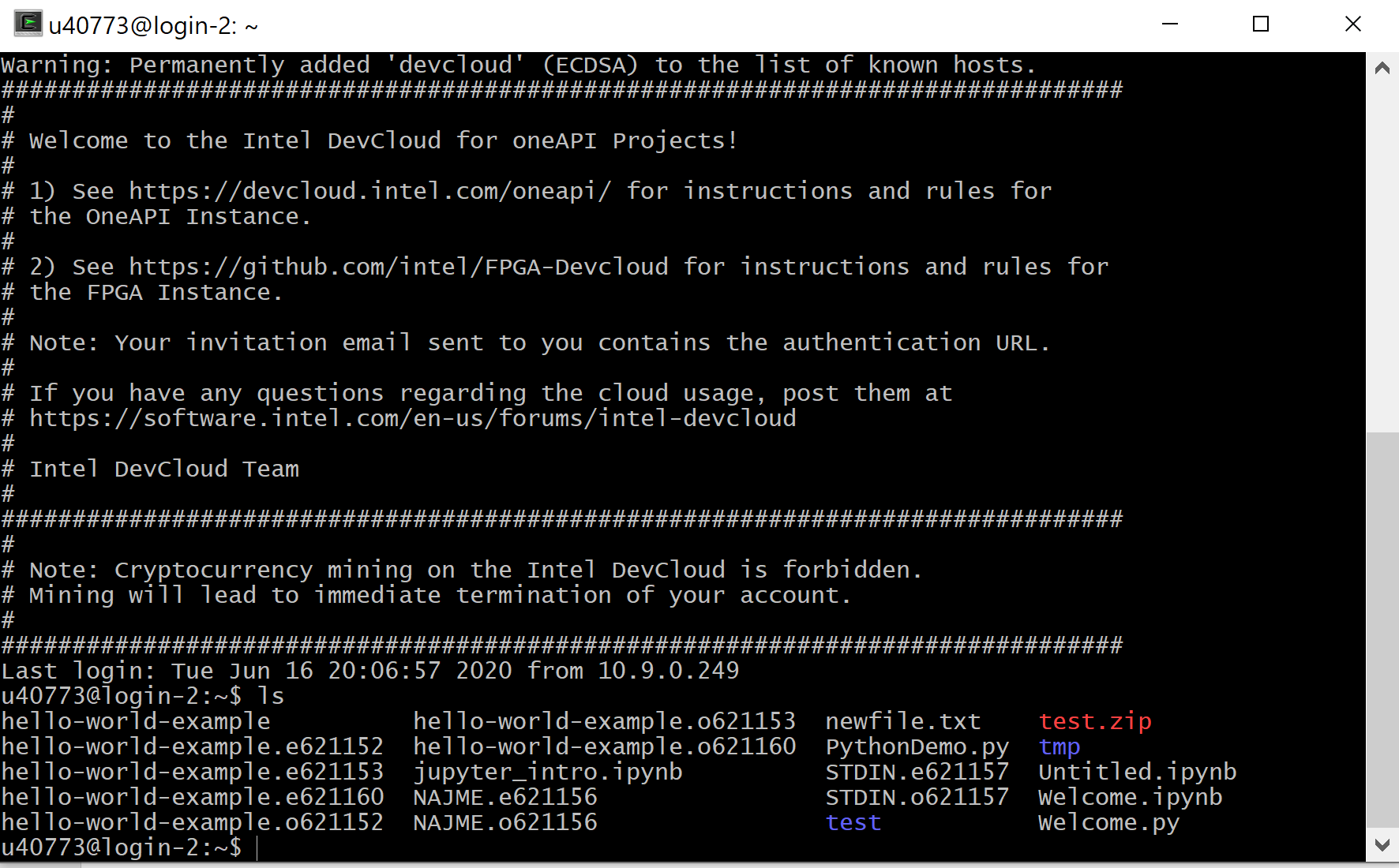
1. Execute this script in the terminal (you may need to adjust the command according to your download location and the downloaded file name):



1. After the preparation steps above, you should be able to log in to your login node in the Intel® DevCloud without a password. Upon the first login, you will be asked to add the hostdevcloud to the list of known hosts. Answer "yes":

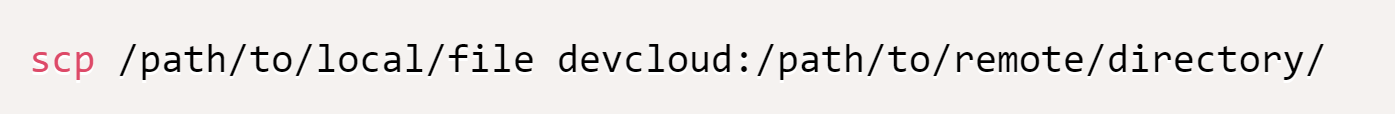


Now, welcome to devcloud, if you run ls command, you will see all your files that you had on Jupyter environment:

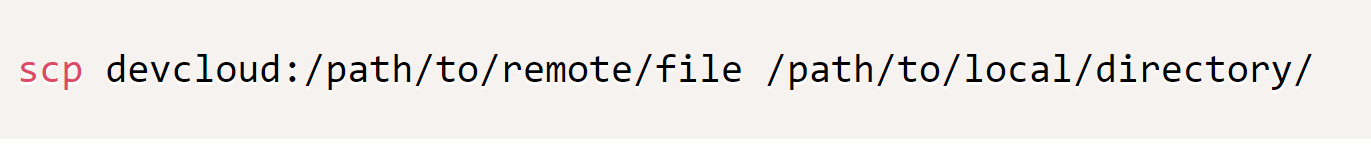


You can logout by *logout* command.

Once your connection is set up, you can copy local files to your login node like this:



And you can copy files from your login node back to your home computer like this:



The rest of commands is similar that what we presented on Jupyter notebook. Good luck!