*This document was originally written approx. on 26/08/2019*

Contents

[Setting Up Hyper-V 2](#_Toc17617785)

[Docker Desktop for Windows 4](#_Toc17617786)

[Prerequisites 4](#_Toc17617787)

[Docker Basics 4](#_Toc17617788)

[Installation 7](#_Toc17617789)

[Configure Docker Desktop for Windows 8](#_Toc17617790)

[Start Docker Desktop for Windows 8](#_Toc17617791)

[Switch between Windows and Linux containers 11](#_Toc17617792)

[Configuration changes 13](#_Toc17617793)

[Firewall rules for shared drives 24](#_Toc17617794)

[Test Docker Installation 25](#_Toc17617795)

[For Linux Container (hello-world) 26](#_Toc17617796)

[For Linux Container (run an interactive terminal inside the spawned container) 27](#_Toc17617797)

[Run Your First Windows Container 29](#_Toc17617798)

[Troubleshooting 31](#_Toc17617799)

# Setting Up Hyper-V

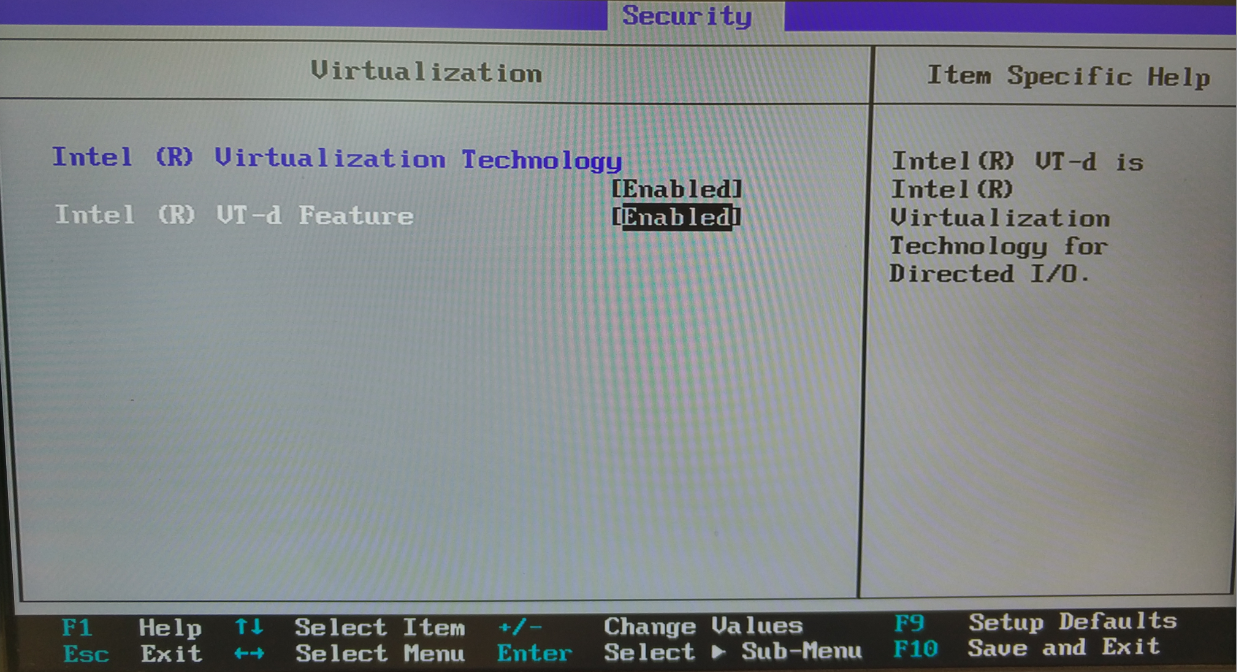
1. Ensure that hardware virtualization support is turned on in the BIOS settings.

[](https://msdnshared.blob.core.windows.net/media/TNBlogsFS/prod.evol.blogs.technet.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/00/48/12/6116.HVW8a.jpg)

(c) <https://blogs.technet.microsoft.com/canitpro/2015/09/08/step-by-step-enabling-hyper-v-for-use-on-windows-10/>   
<https://www.altaro.com/hyper-v/the-hyper-v-virtual-switch-explained-part-1/>

Your BIOS can look like differently; you should find enable Virtualization Technology (VTx). Look on Security menu.

1. It is better also to enable also VT-d Feature in the BIOS.



1. Save the BIOS settings and boot up the machine normally.

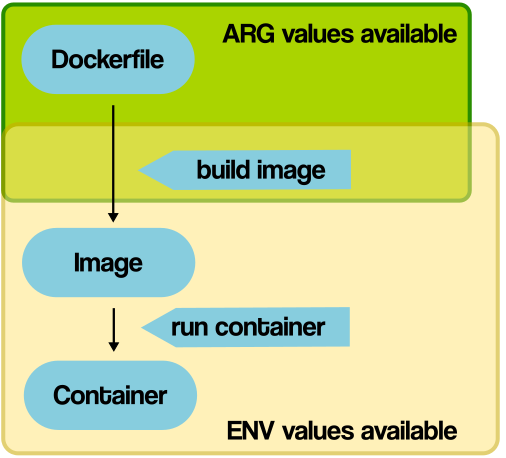
# Docker Desktop for Windows

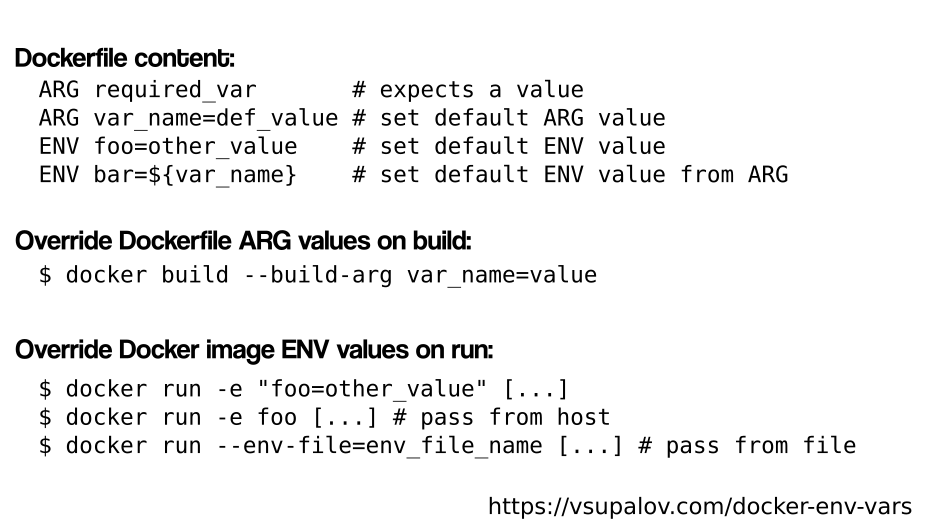
# Prerequisites

Please make sure you meet the following requirements:

* One physical computer system running Windows 10 Professional or Enterprise with Anniversary Update (version 1607) or later.
* Make sure [Hyper-V](https://docs.microsoft.com/virtualization/hyper-v-on-windows/reference/hyper-v-requirements)/hypervisor is enabled (see [previous](#_Setting_Up_Hyper-V) section how to enable it).

# Docker Basics

* For tips&tricks see separate document here <https://github.com/alex-ber/AlexBerDocs/blob/master/Docker/Windows/Docker%20Tips%20and%20Tricks.docx>
* See <https://vsupalov.com/6-docker-basics/>
* Docker ARG, ENV and .env - a Complete Guide   
    
  



A frequent gotcha, if you’re new to Docker and not used to think about images and containers: if you try to set the value of an environment variable from **inside a RUN statement** like RUN export VARI=5 && ..., you won’t have access to it in any of the next RUN statements. The reason for this, is that for each RUN statement, a new container is launched from an intermediate image. An image is saved by the end of the command, but environment variables do not persist that way.

* <https://vsupalov.com/docker-arg-env-variable-guide/>
* See also

ARG buildtime\_variable=default\_value # <- this one's new

ENV env\_var\_name=$buildtime\_variable # we reference it directly

Now, when you’re building your image you can override the default\_value of “buildtime\_variable” any time you like:

$ docker build --build-arg buildtime\_variable=a\_value # ... the rest of the build command is omitted

You can change the value each time you run docker build without editing the Dockerfile. The ARG variable “buildtime\_variable” will be set to your dynamic value and the ENV variable “env\_var\_name” will be set to the fresh ARG value.

<https://vsupalov.com/docker-build-time-env-values/>

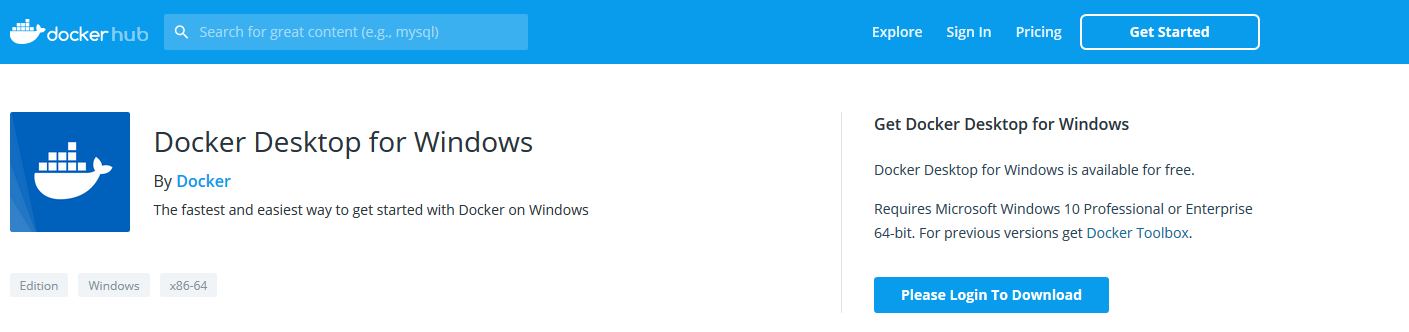
<https://vsupalov.com/set-dynamic-environment-variable-during-docker-image-build/>   
<https://vsupalov.com/build-docker-image-clone-private-repo-ssh-key/>

Note:

* For docker-compose ([PostgreSQL](https://hub.docker.com/_/postgres/) and [Redis](https://hub.docker.com/_/redis/)) examples see <https://vsupalov.com/flask-docker-compose-development-dependencies/>

# Installation

Go to <https://hub.docker.com/editions/community/docker-ce-desktop-windows>



* Click on button “Please Login to Download”.
* You will be required to login. Create an account in Docker Hub if you don't have one already. See also [Organizations and Teams in Docker Hub](https://docs.docker.com/docker-hub/orgs/).
* Download Docker Desktop and run the installer.

Note:

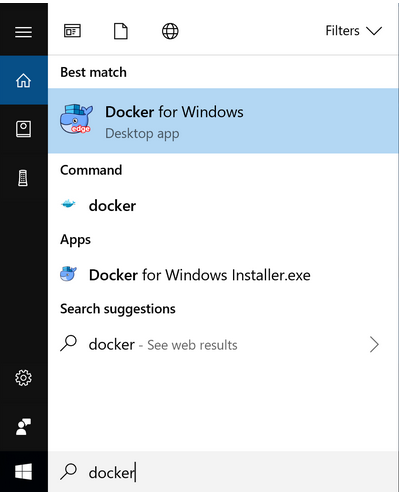
* This will install docker, docker-compose and Kubernetes.
* This documents focuses on docker only.

# Configure Docker Desktop for Windows

## Start Docker Desktop for Windows

See <https://docs.docker.com/docker-for-windows/#docker-settings-dialog>

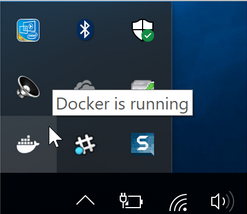
Docker does not start automatically after installation. To start it, search for Docker, select **Docker Desktop for Windows** in the search results, and click it (or hit Enter).



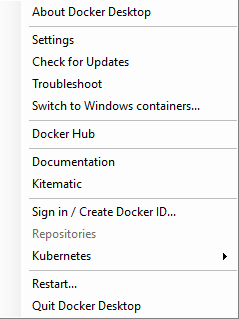
When the whale in the status bar stays steady, Docker is up-and-running, and accessible from any terminal window.



* Open the Docker Desktop for Windows menu by right-clicking the Docker icon in the Notifications area (or System tray):



* Select **Settings** to open the Settings dialog:

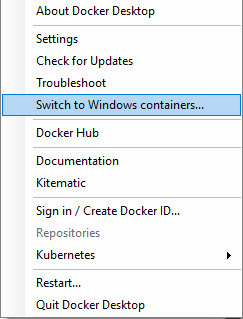


* After installation Docker Desktop defaults to running Linux containers.

### Switch between Windows and Linux containers

See <https://docs.docker.com/docker-for-windows/#docker-settings-dialog>

From the Docker Desktop for Windows menu, you can toggle which daemon (Linux or Windows) the Docker CLI talks to. Select **Switch to Windows containers** to use Windows containers, or select **Switch to Linux containers** to use Linux containers (the default).



Note:

* For more information on Windows containers, refer to this documentation: Microsoft documentation on [Windows containers](https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/index).
* See [Limitations of Windows containers for localhost and published ports](https://docs.docker.com/docker-for-windows/troubleshoot/#limitations-of-windows-containers-for-localhost-and-published-ports) to understand how to connect to Windows containers from the local host.
* **Settings dialog changes with Windows containers**

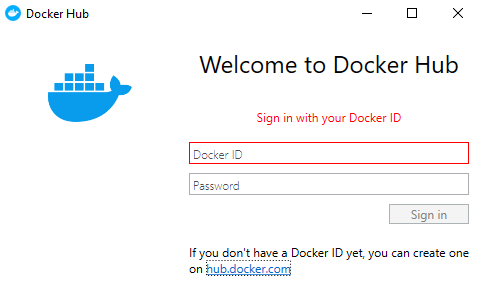
When you switch to Windows containers, the Settings dialog only shows those tabs that are active and apply to your Windows containers:

* General
* Proxies
* Daemon
* Reset

If you set proxies or daemon configuration in Windows containers mode, these apply only on Windows containers. If you switch back to Linux containers, proxies and daemon configurations return to what you had set for Linux containers. Your Windows container settings are retained and become available again when you switch back.

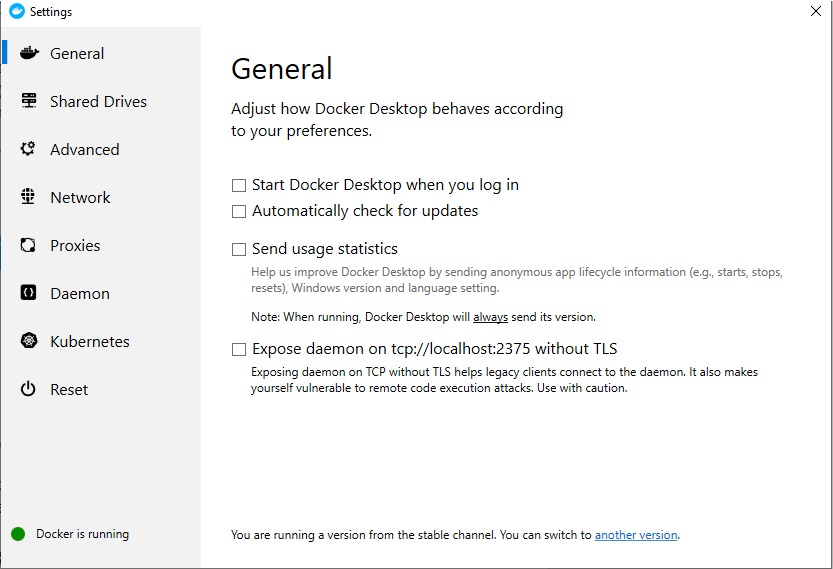
## Configuration changes

* Click on **Sign in / Create Docker ID…** use your credential for Docker Hub (this is account your created when you’ve downloaded Docker Desktop for Windows).



Note: If you don’t have account on Docker Hub click on [link](https://hub.docker.com/?utm_source=docker4win_2.1.0.1&utm_medium=account_create&utm_campaign=referral) that appears in the Windows. See also [Organizations and Teams in Docker Hub](https://docs.docker.com/docker-hub/orgs/).

* Open **Settings (from** menu by right-clicking the Docker icon in the Notifications area (or System tray)):
* On the **General** tab of the Settings dialog, you can configure when to start and update Docker.



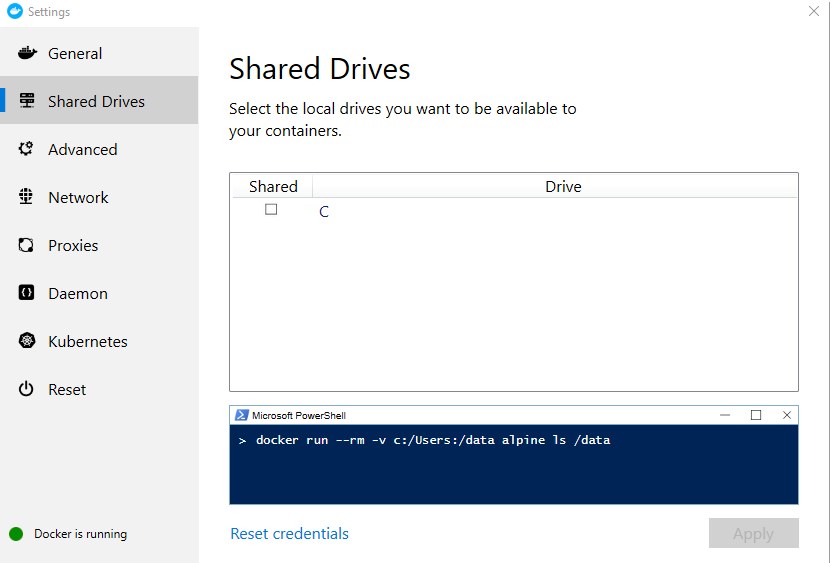
See <https://docs.docker.com/docker-for-windows/#docker-settings-dialog>

* **Start Docker when you log in** - Automatically start the Docker Desktop for Windows application upon Windows system login.
* **Automatically check for updates** - By default, Docker Desktop for Windows automatically checks for updates and notifies you when an update is available. Click **OK** to accept and install updates (or cancel to keep the current version). You can manually update by choosing **Check for Updates** from the main Docker menu.
* **Send usage statistics** - By default, Docker Desktop for Windows sends diagnostics, crash reports, and usage data. This information helps Docker improve and troubleshoot the application. Uncheck to opt out. Docker may also sometimes prompt you for more information.
* On the **Shared Drives** tab of the Settings dialog

See <https://rominirani.com/docker-on-windows-mounting-host-directories-d96f3f056a2c>

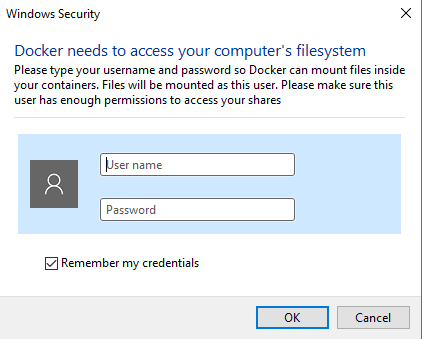
<https://docs.docker.com/docker-for-windows/#docker-settings-dialog>

Share your local drives (volumes) with Docker Desktop for Windows, so that they are available to your [Linux containers](https://docs.docker.com/docker-for-windows/#switch-between-windows-and-linux-containers).



This should be able to list down the drives that you have available on your Windows machine.

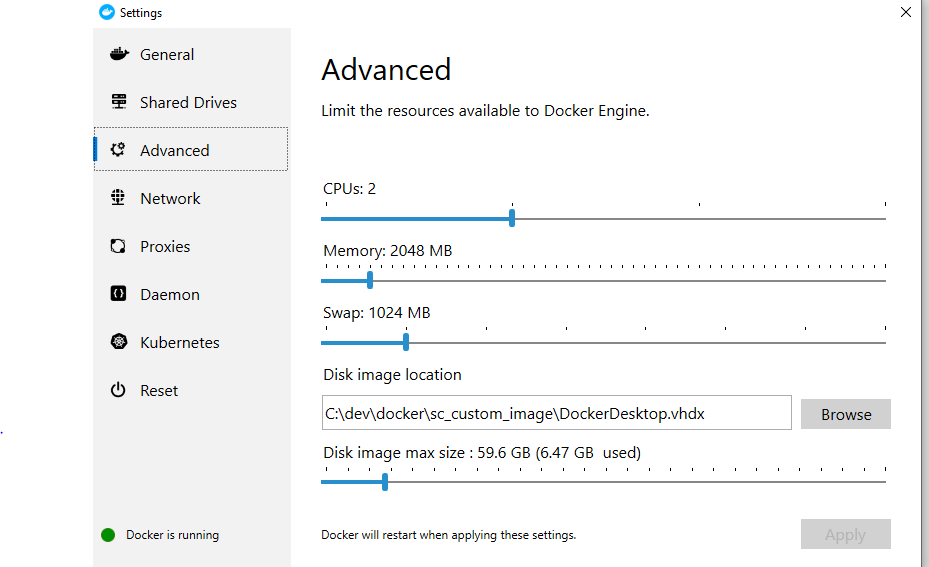
* **Chose C drive (or any other appropriate drive).**
* Click on **Apply.** This will bring up the Credentials dialog and you will need to provide your current Windows credentials. Ensure that you give it correctly. I also suspect that you might need to be an Administrator.



* Permission for shared drives are tied to the credentials you provide here. If you run docker commands under a different username than the one configured here, your containers cannot access the mounted volumes.
* To apply shared drives, you are prompted for your Windows system (domain) username and password. You can select an option to have Docker store the credentials so that you don’t need to enter them every time.

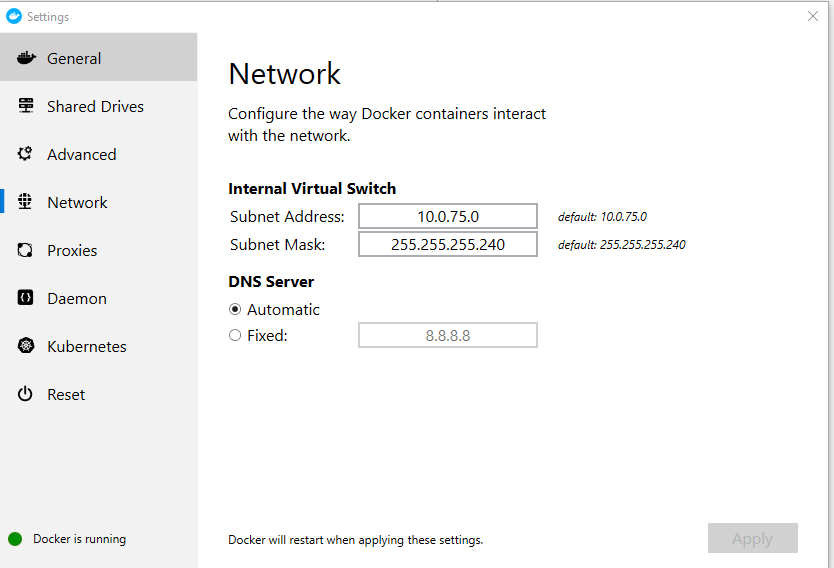
Note:

* Shared drives are only required for mounting volumes in [Linux containers](https://docs.docker.com/docker-for-windows/#switch-between-windows-and-linux-containers), not for Windows containers. If you get errors such as file not found or cannot start service you may need to enable shared drives. See [Volume mounting requires shared drives for Linux containers](https://docs.docker.com/docker-for-windows/troubleshoot/#volume-mounting-requires-shared-drives-for-linux-containers).)
* Docker Desktop for Windows sets permissions to read/write/execute for users, groups and others [0777 or a+rwx](http://permissions-calculator.org/decode/0777/). This is not configurable. See [Permissions errors on data directories for shared volumes](https://docs.docker.com/docker-for-windows/troubleshoot/#permissions-errors-on-data-directories-for-shared-volumes).
* Ensure the domain user has access to shared drives, as described in [Verify domain user has permissions for shared drives](https://docs.docker.com/docker-for-windows/troubleshoot/#verify-domain-user-has-permissions-for-shared-drives-volumes).
* Windows Firewall or your third party firewall software can prevent drive sharing. See [next sub-section](#_Firewall_rules_for_2) for the details (don’t forget to go back to this section).
* On the **Advanced** tab of the Settings dialog, you can optionally change Disk Image location.  
    
  See <https://docs.docker.com/docker-for-windows/#docker-settings-dialog>



By default, images are stored in ProgramData.

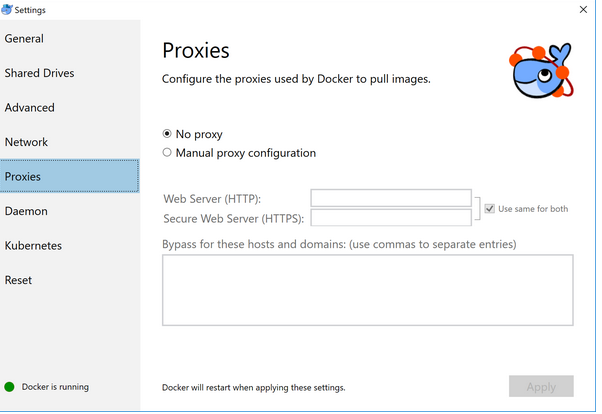
* **CPUs** - Change the number of processors assigned to the Linux VM.
* **Memory** - Change the amount of memory the Docker Desktop for Windows Linux VM uses.
* On **Network** tab of the Settings dialog (and any subsequent tab, do nothing).

See <https://docs.docker.com/docker-for-windows/#docker-settings-dialog>  
  


* **Internal Virtual Switch** - You can specify a network address translation (NAT) prefix and subnet mask to enable Internet connectivity.
* **DNS Server** - You can configure the DNS server to use dynamic or static IP addressing.
* On **Proxies** tab of the Settings dialog

See <https://docs.docker.com/docker-for-windows/#docker-settings-dialog>

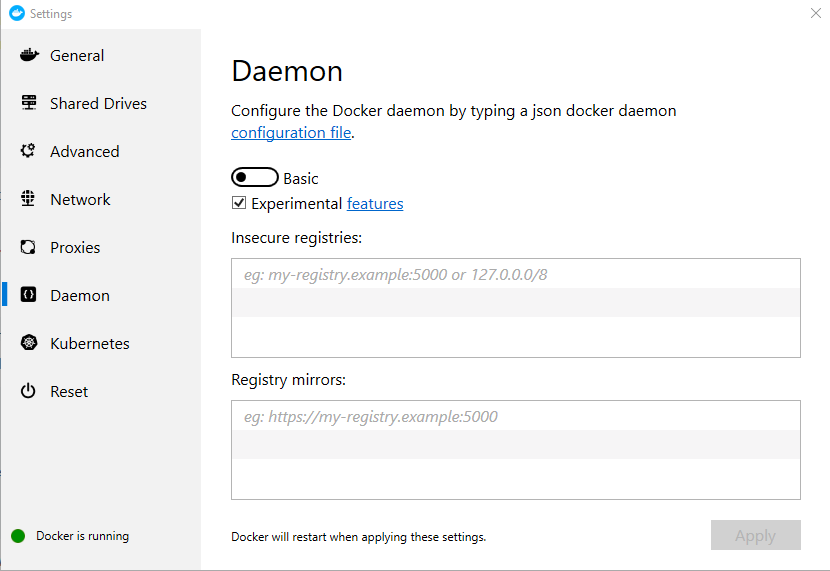
Docker Desktop for Windows lets you configure HTTP/HTTPS Proxy Settings and automatically propagates these to Docker and to your containers. For example, if you set your proxy settings to http://proxy.example.com, Docker uses this proxy when pulling containers.



* On **Daemon** tab of the Settings dialog**.**

See <https://docs.docker.com/docker-for-windows/#docker-settings-dialog>

You can configure the Docker daemon to hone how your containers run. **Advanced mode** lets you edit the JSON directly. **Basic mode** lets you configure the more common daemon options with interactive settings (and also JSON).



* **Daemon configuration file**

The **Advanced** daemon settings provide the original option to directly edit the JSON configuration file for the [daemon](https://docs.docker.com/engine/reference/commandline/dockerd/).

See <https://docs.microsoft.com/en-us/virtualization/windowscontainers/manage-docker/configure-docker-daemon> for how to configure Docker with a configuration file.

* **Experimental Mode**. Both Docker Desktop for Windows Stable and Edge releases have the experimental version of Docker Engine enabled, described in the [Docker Experimental Features README](https://github.com/docker/cli/blob/master/experimental/README.md) on GitHub.

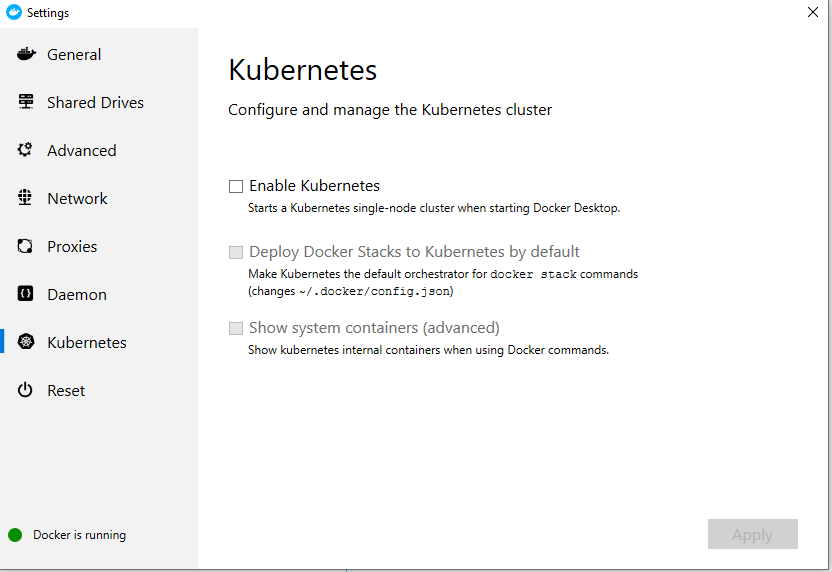
Experimental features are not appropriate for production environments or workloads. They are meant to be sandbox experiments for new ideas. Some experimental features may become incorporated into upcoming stable releases, but others may be modified or pulled from subsequent Edge releases, and never released on Stable.  
  
Note: I enables checkbox with experimental features to be able to use –squash flag.

* **Custom registries.**

Normally, you store public or private images in [Docker Hub](https://hub.docker.com/) and [Docker Trusted Registry](https://docs.docker.com/datacenter/dtr/2.4/guides/). Here, you can use Docker to set up your own insecure [registry](https://docs.docker.com/registry/introduction/). Simply add URLs for insecure registries and registry mirrors on which to host your images.

* On **Kubernetes** tab of the Settings dialog**.**

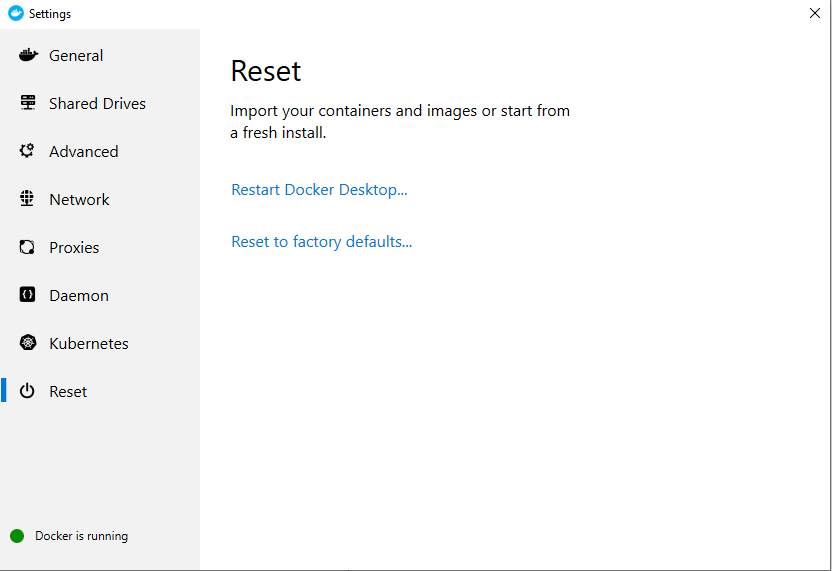
See <https://docs.docker.com/docker-for-windows/#docker-settings-dialog>



The Kubernetes client command, kubectl, is included and configured to connect to the local Kubernetes server.

* On **Reset** tab of the Settings dialog**.**

On the Reset tab, you can restart Docker or reset its configuration.

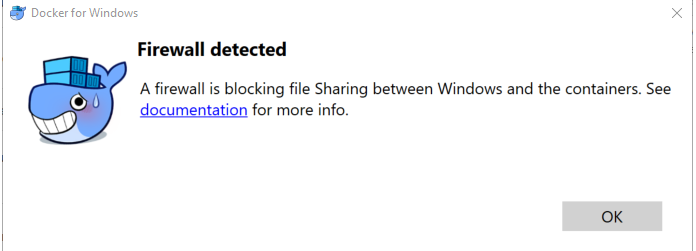


* **Restart Docker** - Shuts down and restarts the Docker application.
* **Reset to factory defaults** - Resets Docker to factory defaults. This is useful in cases where Docker stops working or becomes unresponsive.

### Firewall rules for shared drives

See <https://docs.docker.com/docker-for-windows/#docker-settings-dialog>

Shared drives require port 445 to be open between the host machine and the virtual machine that runs Linux containers. Docker detects if port 445 is closed and shows the following message when you try to add a shared drive:



To share the drive, allow connections between the Windows host machine and the virtual machine in Windows Firewall or your third party firewall software. You do not need to open port 445 on any other network.

By default, allow connections to 10.0.75.1 on port 445 (the Windows host) from 10.0.75.2 (the virtual machine). If your firewall rules seem correct, you may need to toggle or reinstall the File and Print sharing service on the Hyper-V virtual network card. For more details see <https://blog.olandese.nl/2017/05/03/solve-docker-for-windows-error-a-firewall-is-blocking-file-sharing-between-windows-and-the-containers/>

Note:

* If you have ESET installed see <https://stackoverflow.com/questions/42203488/settings-to-windows-firewall-to-allow-docker-for-windows-to-share-drive/42955491#42955491>

## Using DNS Server available in hosts inside Docker container

See <https://github.com/moby/moby/issues/23910>   
<https://development.robinwinslow.uk/2016/06/23/fix-docker-networking-dns/>

<https://docs.docker.com/engine/reference/commandline/dockerd/#daemon-configuration-file>

On Linux you can type:  
  
cat /etc/resolv.conf

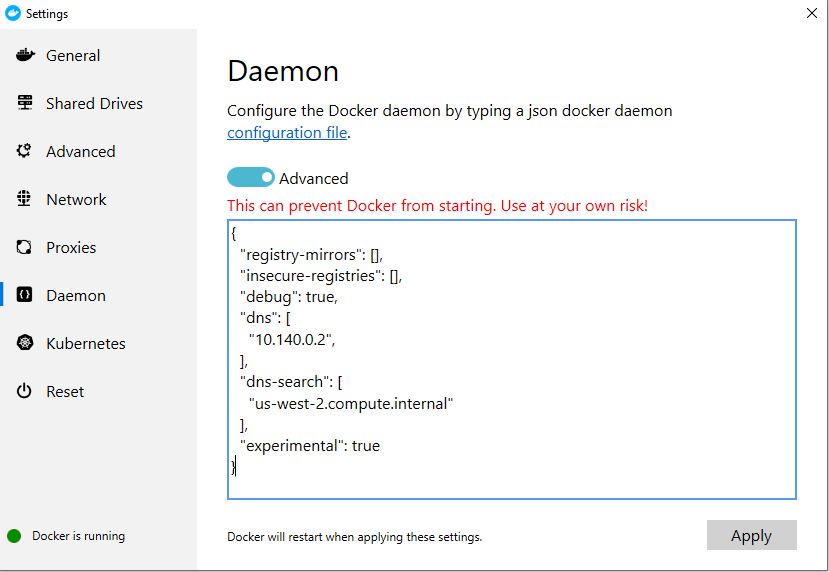
Look on *nameserver* (these are entries for your DNS servers) and *search* (this is your *DNS suffix search*).

On Windows type:  
  
ipconfig /all

Look on *DNS Servers* and *DNS Suffix Search List (or Primary Dns Suffix)*

(this is your *DNS suffix search*).

Go to Docker Desktop -> Daemon, click on Advanced.



Put to “dns” at *DNS Servers* you have found above.

Optionally, put to “dns-search” at *DNS suffix search* you have found above.

Click on “apply”, this will restart your docker container.

## Configure an interpreter using Docker in Pycharm Professional

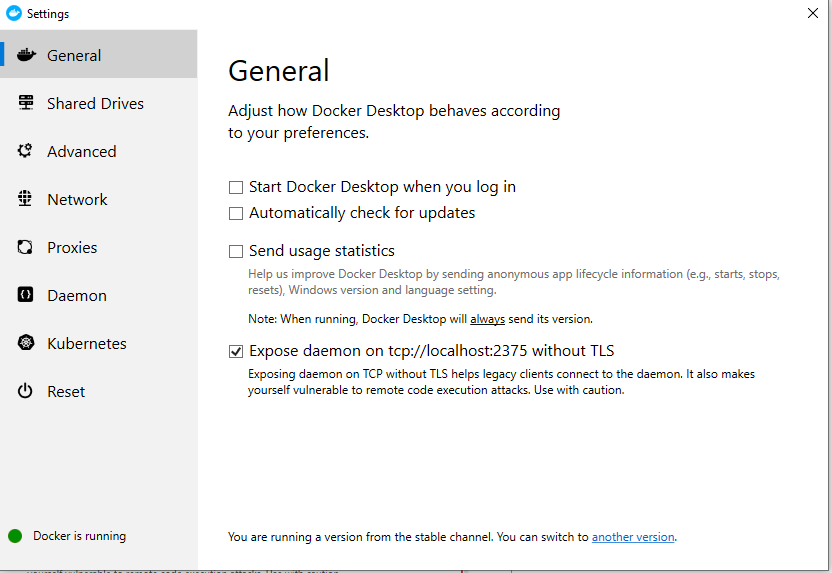
See

<https://www.jetbrains.com/help/pycharm/using-docker-as-a-remote-interpreter.html>

<https://www.jetbrains.com/help/idea/2020.1/docker.html#run-containers>

If you want to run/debug your application inside docker container from Pycharm Professional, you should expose docker deamon.

Go to Docker Desktop -> General and check check-box *expose daemon*.



# Test Docker Installation

All my examples are running from **Git bash**.

They should also work in Bash for Windows.

Using cmd/command prompt or PowerShell, but not PowerShell ISE as terminal window is also supported in Docker Desktop for Windows, but the code provided in the document will require adaptations. See, for example <https://stackoverflow.com/questions/48813286/stop-all-docker-containers-at-once-on-windows>

## For Linux Container (hello-world)

See <https://docs.docker.com/docker-for-windows/#docker-settings-dialog>

1. Switch to [Linux Container](#_Switch_between_Windows).
2. Open **Git bash.**
3. Run docker --version to ensure that you have a supported version of Docker:

Type:

docker --version  
  
Output:

Docker version 19.03.1, build 74b1e89

1. Pull the [hello-world image](https://hub.docker.com/r/library/hello-world/) from Docker Hub and run a container:

Type:

docker run hello-world

Output:

Unable to find image 'hello-world:latest' locally

latest: Pulling from library/hello-world

1b930d010525: Pull complete

Digest: sha256:6540fc08ee6e6b7b63468dc3317e3303aae178cb8a45ed3123180328bcc1d20f

Status: Downloaded newer image for hello-world:latest

Hello from Docker!

This message shows that your installation appears to be working correctly.

...

## For Linux Container (run an interactive terminal inside the spawned container)

See <https://docs.docker.com/docker-for-windows/#docker-settings-dialog>

1. Switch to [Linux Container](#_Switch_between_Windows).
2. Open **Git bash.**
3. Run docker --version to ensure that you have a supported version of Docker:

Type:

docker --version  
  
Output:

Docker version 19.03.1, build 74b1e89

1. Pull an image of the [Ubuntu OS](https://hub.docker.com/r/_/ubuntu/) from Docker Hub and run a an interactive terminal inside the spawned container:

docker run –it ubuntu bash

Output:  
  
Unable to find image 'ubuntu:latest' locally

latest: Pulling from library/ubuntu

7413c47ba209: Pull complete

0fe7e7cbb2e8: Pull complete

1d425c982345: Pull complete

344da5c95cec: Pull complete

Digest: sha256:c303f19cfe9ee92badbbbd7567bc1ca47789f79303ddcef56f77687d4744cd7a

Status: Downloaded newer image for ubuntu:latest

root@745d2c89b47f:/#

1. You are in the container. At the root # prompt, check the hostname of the container:

root@8aea0acb7423:/# hostname

8aea0acb7423

1. Notice that the hostname is assigned as the container ID (and is also used in the prompt).
2. Exit the shell with the exit command (which also stops the container):

root@8aea0acb7423:/# exit

## Run Your First Windows Container

See <https://docs.microsoft.com/en-us/virtualization/windowscontainers/quick-start/quick-start-windows-10>

1. Switch to [Windows Container](#_Switch_between_Windows).
2. Open **Git bash.**
3. Run docker --version to ensure that you have a supported version of Docker:

Type:

docker --version  
  
Output:

Docker version 19.03.1, build 74b1e89

1. Start a container with an interactive session from the nanoserver image. Once the container has started, you will be presented with a command shell from within the container.  
     
   Type:

docker run -it --name nanoserver-base mcr.microsoft.com/windows/nanoserver:1809 cmd.exe

1. Inside the container we will create a simple ‘Hello World’ text file.

Type:  
  
echo "Hello World!" > Hello.txt

1. When completed, exit the container.  
     
   Type:  
     
   exit
2. Run the following command to create the new ‘HelloWorld’ image. Replace with the id of your container.  
     
   Type:  
     
   docker commit nanoserver-base hellowindowsworld
3. Finally, to run the container, use the docker run command

Type:  
  
docker run --rm hellowindowsworld cmd.exe /s /c type Hello.txt

Note:

The outcome of the docker run command is that a container running under Hyper-V isolation was created from the 'HelloWorld' image, an instance of cmd was started in the container and executed a reading of our file (output echoed to the shell), and then the container stopped and removed.

# Troubleshooting

* Stop all *running* containers at once  
    
  See <https://stackoverflow.com/questions/48813286/stop-all-docker-containers-at-once-on-windows>
* To stop all *running* containers in **Git Bash** type:

docker stop $(docker ps -q)

* To stop&remove all *running* containers:  
    
  docker rm $(docker stop $(docker ps -q))
* To remove **stopped** containers:  
    
  docker container prune
* To remove **all** containers:  
    
  docker rm $(docker ps -aq)
* To see all installed image type in **Git Bash**:  
    
  docker images –a
* To see all containers  
    
  docker ps –a
* Can’t delete docker image with dependent child images  
    
  Problem:

I am trying

docker rmi c565603bc87f

Error:

Error response from daemon: conflict: unable to delete c565603bc87f (cannot be forced) - image has dependent child images

So I can't delete image even with -f flag. How to delete image then and all of its children ?

Solution:  
  
See <https://stackoverflow.com/questions/38118791/can-t-delete-docker-image-with-dependent-child-images>

Tip: Stop&remove all containers first (see above).

Building on Simon Brady's brute force method [here](https://stackoverflow.com/a/41176677/3273031), if you don't have a ton of images you can use this shell function:

In **Git Bash** type:

recursive\_remove\_image() {

for image in $(docker images --quiet --filter "since=${1}")

do

if [ $(docker history --quiet ${image} | grep ${1}) ]

then

recursive\_remove\_image "${image}"

fi

done

echo "Removing: ${1}"

docker rmi -f ${1}

}

and then call it using recursive\_remove\_image <image-id>.

In the provided example, type:  
  
recursive\_remove\_image c565603bc87f

If you receive error message, just retype the following line again, until you’re done.

* To start a stopped container in **Git Bash** type:

See <https://stackoverflow.com/a/39689438/1137529>

docker start container\_name

Note: Don’t confuse with

docker restart container\_name   
  
This is used to restart a running container