

# Experiment 01: Foundation Installation

## Install Kind AKA KiND AKA KIND (Kubernetes in Docker)

### For MacOS

First we install Docker for MacOS

```
$ brew cask install docker
```

```
$ open /Applications/Docker.app
```

Next we install kind (Kubernetes in Docker).

```
$ brew install kind
```

```
$ kind version
```

Kubectl

```
$ brew install kubectl
```

```
$ kubectl version
```

Git

```
$ brew install git
```

```
$ git version
```

### For Windows

For **Windows** platform, we'll install Chocolatey, Kind, and Docker Desktop. Kind will automatically install Docker Desktop. Chocolatey is a package manager for Windows like Brew is for MacOS and NPM for NodeJS.

Open a Powershell window with "Run as Administrator"

For installing Docker Desktop & KIND we must have your execution policy set to bypass or something even less restrictive.

```
PS C:\Users\kubelord> Get-ExecutionPolicy
```

Bypass

**NOTE:** If the response comes back with Restricted, you will need to execute the execute policy update noted below

```
PS C:\Users\kubelord> Set-ExecutionPolicy Bypass
```

Execution Policy Change

The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose you to the security risks described in the [about\\_Execution\\_Policies](#) help topic at

<https://go.microsoft.com/fwlink/?LinkID=135170>. Do you want to change the execution policy?

[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): **A**

Set-ExecutionPolicy : Access to the registry key  
'HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\PowerShell\1\ShellIds\Microsoft.PowerShell' is denied. To change the execution  
policy for the default (LocalMachine) scope, start Windows PowerShell with the "Run as administrator" option. To  
change the execution policy for the current user, run "Set-ExecutionPolicy -Scope CurrentUser".

At line:1 char:1

+ Set-ExecutionPolicy Bypass

+ ~~~~~

+ CategoryInfo : PermissionDenied: (:) [Set-ExecutionPolicy],  
UnauthorizedAccessException

+ FullyQualifiedErrorId :

System.UnauthorizedAccessException,Microsoft.PowerShell.Commands.SetExecutionPolicyCommand

nd

PS C:\Users\kubelord> **Get-ExecutionPolicy**

Bypass

Install Chocolatey, a Windows Package Manager. Now that we've confirmed or updated and confirmed our execution policy is correct.

```
PS C:\Users\kubelord> Set-ExecutionPolicy Bypass -Scope Process -Force;  
[System.Net.ServicePointManager]::SecurityProtocol =  
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object  
System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))
```

Creating ChocolateyInstall as an environment variable (targeting 'User')

Setting ChocolateyInstall to 'C:\ProgramData\chocolatey'

WARNING: It's very likely you will need to close and reopen your shell

before you can use choco.

PS C:\Users\kubelord > **choco /?**

If the above does not return the help for Chocolatey, then close the Powershell prompt and open another one.

Next we'll use Chocolatey to install kind (**K**ubernetes **i**n **D**ocker)

PS C:\Users\kubelord> **choco install kind**

Chocolatey v0.10.15

Installing the following packages:

kind

By installing you accept licenses for the packages.

Progress: Downloading docker-desktop 2.3.0.4... 100%

Progress: Downloading kind 0.8.1... 100%

docker-desktop v2.3.0.4 [Approved]

...

The install of docker-desktop was successful.

Software installed to 'C:\Program Files\Docker\Docker'

kind v0.8.1 [Approved]

kind package files install completed. Performing other installation steps.

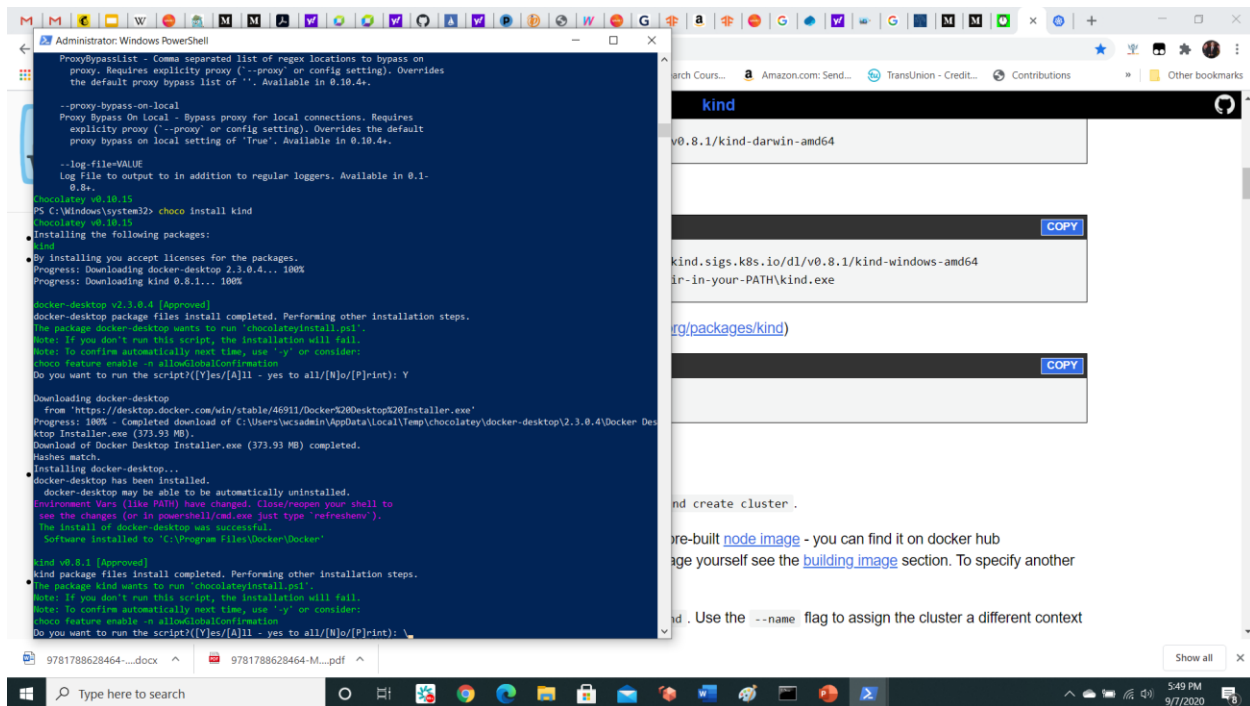
...

The install of kind was successful.

...

PS C:\Windows\system32>

You can also install manually. If you were going to install manually, load the following URL in a web '<https://github.com/kubernetes-sigs/kind/releases/download/v0.8.1/kind-windows-amd64>' for the Windows binary for other platforms view <https://kind.sigs.k8s.io/docs/user/quick-start/>



```
Administrator: Windows PowerShell

ProxyBypassList - Comma separated list of regex locations to bypass on
proxy. Requires explicitly proxy ('--proxy' or config setting). Overrides
the default proxy bypass list of ''. Available in 0.10.4+.

--proxy-bypass-on-local
Proxy Bypass On Local - Bypass proxy for local connections. Requires
explicitly proxy ('--proxy' or config setting). Overrides the default
proxy bypass on local setting of 'True'. Available in 0.10.4+.

--log-file=VALUE
Log File to output to in addition to regular loggers. Available in 0.1-
0.8+.

Chocolatey v0.10.15
PS C:\Windows\system32> choco install kind
Chocolatey v0.10.15
Installing the following packages:
kind
By installing you accept licenses for the packages.
Progress: Downloading docker-desktop 2.3.0.4... 100%
Progress: Downloading kind 0.8.1... 100%

docker-desktop v2.3.0.4 [Approved]
docker-desktop package files install completed. Performing other installation steps.
The package docker-desktop wants to run 'chocolateyinstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint): Y

Downloading docker-desktop
from 'https://desktop.docker.com/win/stable/46911/Docker%20Desktop%20Installer.exe'
Progress: 100% - Completed download of C:\Users\wcsadmin\AppData\Local\Temp\chocolatey\docker-desktop\2.3.0.4\Docker Des
ktop Installer.exe (373.93 MB).
Download of Docker Desktop Installer.exe (373.93 MB) completed.
Hashes match.
Installing docker-desktop...
docker-desktop has been installed.
docker-desktop may be able to be automatically uninstalled.
Environment Vars (like PATH) have changed. Close/reopen your shell to
see the changes (or in powershell/cmd.exe just type 'refreshenv').
The install of docker-desktop was successful.
Software installed to 'C:\Program Files\Docker\Docker'

kind v0.8.1 [Approved]
kind package files install completed. Performing other installation steps.
The package kind wants to run 'chocolateyinstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint): \
```

```
docker ps -a --filter label=io.x-k8s.kind.cluster=kind --format '{{.Names}}'
```

```
Administrator: Windows PowerShell

Chocolatey v0.10.15
Installing the following packages:
kind
By installing you accept licenses for the packages.
Progress: Downloading docker-desktop 2.3.0.4... 100%
Progress: Downloading kind 0.8.1... 100%

docker-desktop v2.3.0.4 [Approved]
docker-desktop package files install completed. Performing other installation steps.
The package docker-desktop wants to run 'chocolateyinstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint): Y

Downloading docker-desktop
  from 'https://desktop.docker.com/win/stable/46911/Docker%20Desktop%20Installer.exe'
Progress: 100% - Completed download of C:\Users\wcsadmin\AppData\Local\Temp\chocolatey\docker-desktop\2.3.0.4\Docker Desktop Installer.exe (373.93 MB).
Download of Docker Desktop Installer.exe (373.93 MB) completed.
Hashes match.
Installing docker-desktop...
docker-desktop has been installed.
  docker-desktop may be able to be automatically uninstalled.
Environment Vars (like PATH) have changed. Close/reopen your shell to
see the changes (or in powershell/cmd.exe just type 'refreshenv').
The install of docker-desktop was successful.
  Software installed to 'C:\Program Files\Docker\Docker'

kind v0.8.1 [Approved]
kind package files install completed. Performing other installation steps.
The package kind wants to run 'chocolateyinstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint): Y

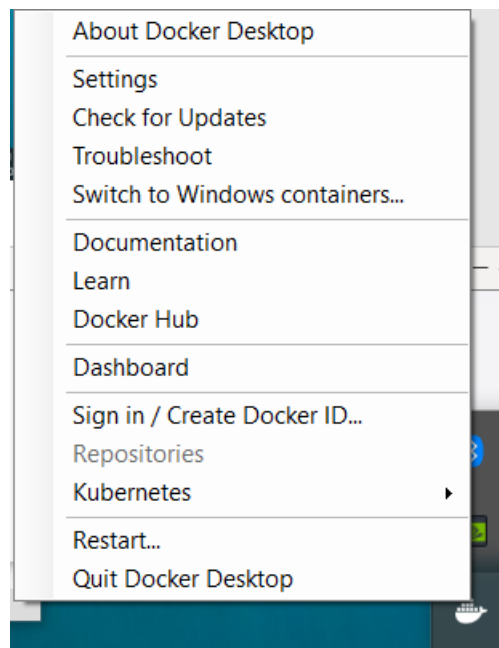
Downloading kind 64 bit
  from 'https://github.com/kubernetes-sigs/kind/releases/download/v0.8.1/kind-windows-amd64'
Progress: 100% - Completed download of C:\ProgramData\chocolatey\lib\kind\kind.exe (9.32 MB).
Download of kind.exe (9.32 MB) completed.
Hashes match.
ShimGen has successfully created a shim for kind.exe
The install of kind was successful.
  Software install location not explicitly set, could be in package or
  default install location if installer.

Chocolatey installed 2/2 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
PS C:\Windows\system32>
```

If you don't see the little Docker whale in the system tray, you will have to restart your Windows VM or machine



After restart or if you have the docker whale in your system tray, we can open the dashboard. When you start Windows you should see notification from your System Tray that Docker for Hyper-V backend is starting and then in the system tray you should see our favorite containerized whale.



Next install **kubectl**.

Based on your operating system you could homebrew to install the kubectl or chocolatey for Windows.

<https://kubernetes.io/docs/tasks/tools/install-kubectl/>

For Windows we're using Chocolatey for the installation

```
c:\> choco install kubernetes-cli
```

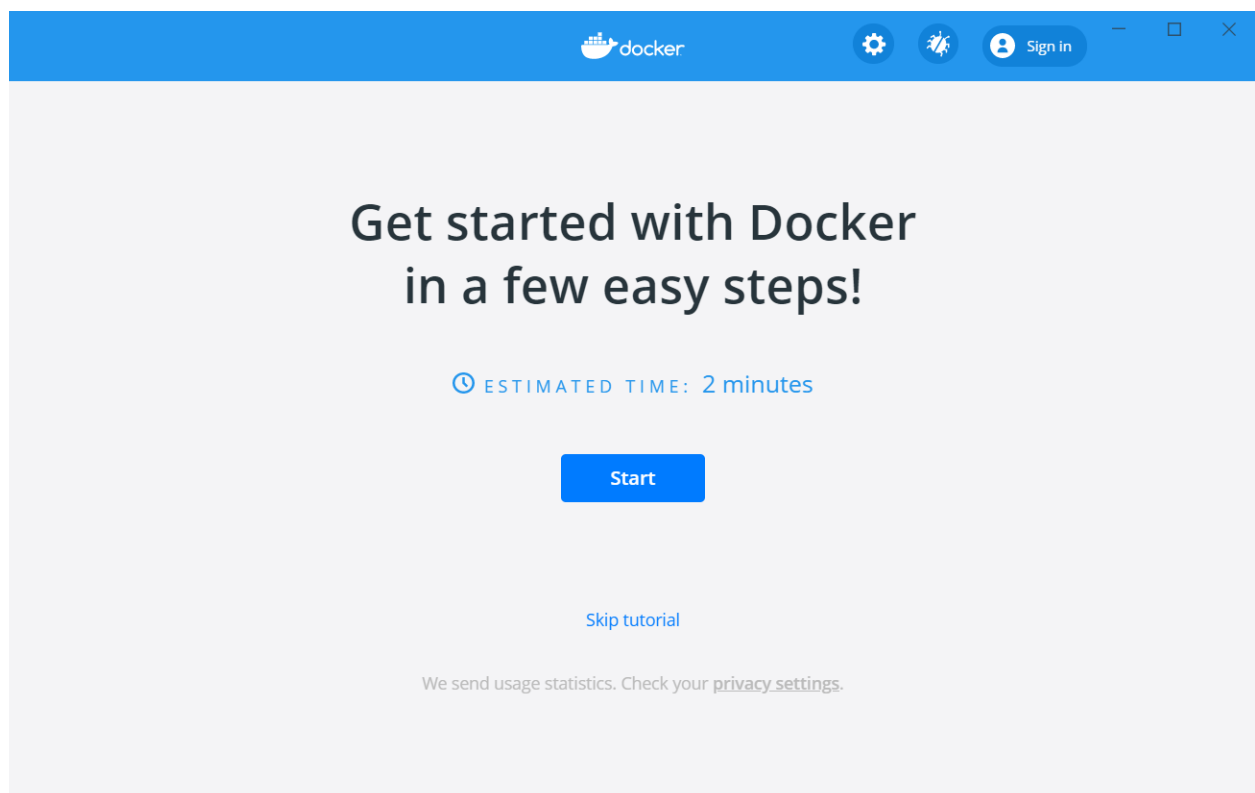
Verify your install

```
C:\>kubectl version --client
```

```
Client Version: version.Info{Major:"1", Minor:"19", GitVersion:"v1.19.0",  
GitCommit:"e19964183377d0ec2052d1f1fa930c4d7575bd50", GitTreeState:"clean",  
BuildDate:"2020-08-26T14:30:33Z", GoVersion:"go1.15", Compiler:"gc",  
Platform:"windows/amd64"}
```

## For MacOS and Windows

**Docker Validation via Tutorial (Docker for Desktop is fairly new for both Windows and MacOS)**



Take the tour, if this is your first time, or you're reinstalling. It's quick and painless.

**Platform Note:** The docker commands and just about every other command for docker, kubectl, k3d, and KIND in the experiments for our session are the same. The only platform specifics are tied to configuration and installation. Installs for Windows are done with Chocolatey, for MacOS with Brew and obviously there are differences in the way we set and reference environment variables between the platforms.

## Windows:

```
set ENVVAR-1=Some-value
```

```
type %ENVVAR-1%
```

Open a Git Bash terminal in your Projects folder, create the folder if it does not already exist

## MacOS:

```
export ENVVAR-1=Some-value
```

```
cat $ENVVAR-1
```

Open a terminal and change directory to your Projects folder, create the folder if it does not already exist

The screenshot shows the Docker Desktop application window. The top bar is blue with the Docker logo and a 'Sign In' button. The main content area is light gray and displays a tutorial step titled 'First, clone a repository'. On the left, a vertical list of steps is shown: 1. Clone (highlighted), 2. Build, 3. Run, and 4. Share. The main text explains that the 'Getting Started' project is a simple GitHub repository used for building and running containers. It includes a code block with the command: `git clone https://github.com/docker/getting-started.git`. Below this, it says 'You can also type the command directly in a command line interface.' and provides a 'Next Step' button. A 'Skip Tutorial' link is also present. On the right side of the window, a Windows PowerShell terminal is open, showing the copyright notice for Microsoft Corporation and the current directory path: `PS C:\Users\wcsadmin>`.



If you have Git installed, which is necessary for the class continue to click the Command Button and it will be pasted automatically (like Katacoda) into the Powershell window and clone the repo for the docker getting started.

The next step is to change to the folder with the repo we just cloned and build a docker container

**\$ mkdir getting-started**

**\$ cd getting-started**

**\$ docker build -t docker101tutorial .**

Now, build the image

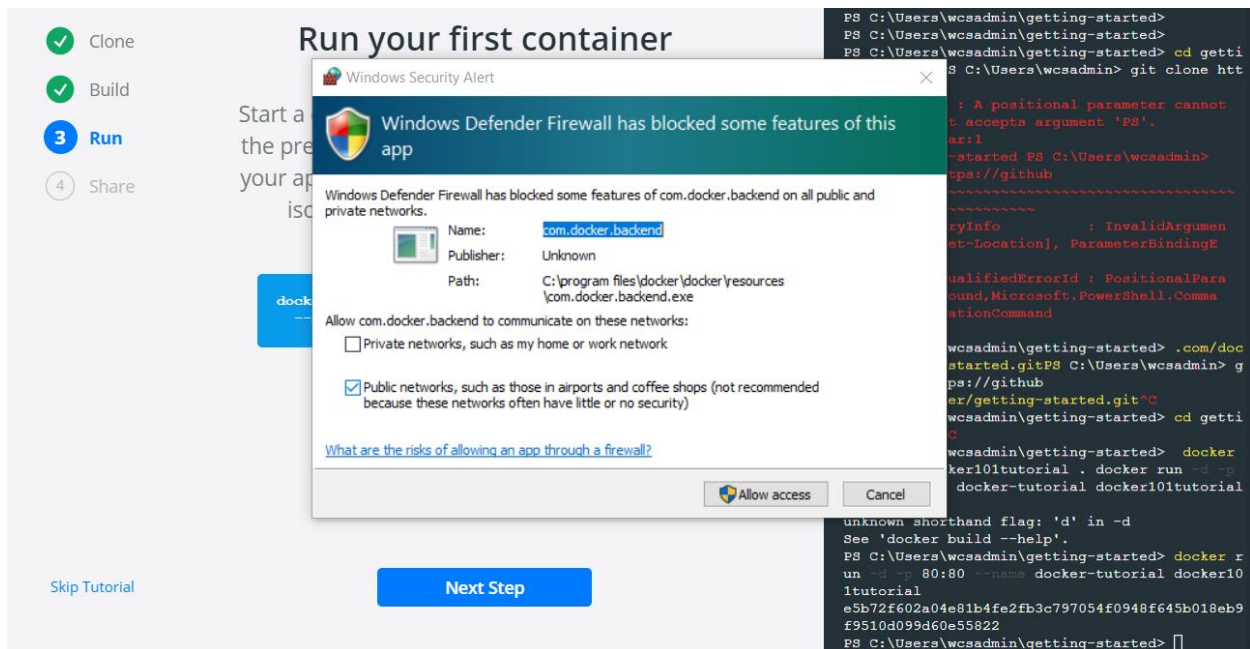
A Docker image is a private file system just for your container. It provides all the files and code your container needs.

```
cd getting-started
docker build -t docker101tutorial .
```

Next Step

```
Removing intermediate container a33bdcfa6b99
--> 77a2c7cad879
Step 16/21 : FROM base AS build
--> 4e5eddf7c60b
Step 18/21 : RUN mkdocs build
INFO     - Cleaning site directory
INFO     - Building documentation to directory : /app/site
INFO     - The following pages exist in the docs directory, but are not included in the "nav" configuration:
- index.md
Removing intermediate container 18e231fa0330
--> d8e558400369
Step 19/21 : FROM nginx:alpine
alpine: Pulling from library/nginx
df20fa9351a1: Already exists
3db268b1fe8f: Pull complete
f682f0660e7a: Pull complete
7eb0e8838bc0: Pull complete
e8bf1226cc17: Pull complete
Digest: sha256:a97eb9ecc708c8aa715ccfb5e9338f5456e4b65575daf304f108301f3b497314
Status: Downloaded newer image for nginx:alpine
--> 6f715d38cfe0
Step 20/21 : COPY --from=app-zip-creator /app.zip /usr/share/nginx/html/assets/app.zip
--> 6ee3ec6870e5
Step 21/21 : COPY --from=build /app/site /usr/share/nginx/html
--> 88c2c625cb70
Successfully built 88c2c625cb70
Successfully tagged docker101tutorial:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions for sensitive files and directories.
PS C:\Users\wcsadmin\getting-started>
```


**\$ docker run -d -p 80:80 --name docker-tutorial docker101tutorial**







Click **“Allow access”** to allow your container to be run


We should see the ID for our container returned. It will look similar to below


**e5b72f602a04e81b4fe2fb3c797054f0948f645b018eb9f9510d099d60e55822**






 Clone

 Build

 Run

 Share


## Now save and share your image

You must be signed in to Docker Hub to share your image.

[Sign in here.](#)

Save and share your image on Docker Hub to enable other users to easily download and run the image on any destination machine.

```
docker tag docker101tutorial {userName}/docker101tutorial
docker push {userName}/docker101tutorial
```



[Skip Tutorial](#)

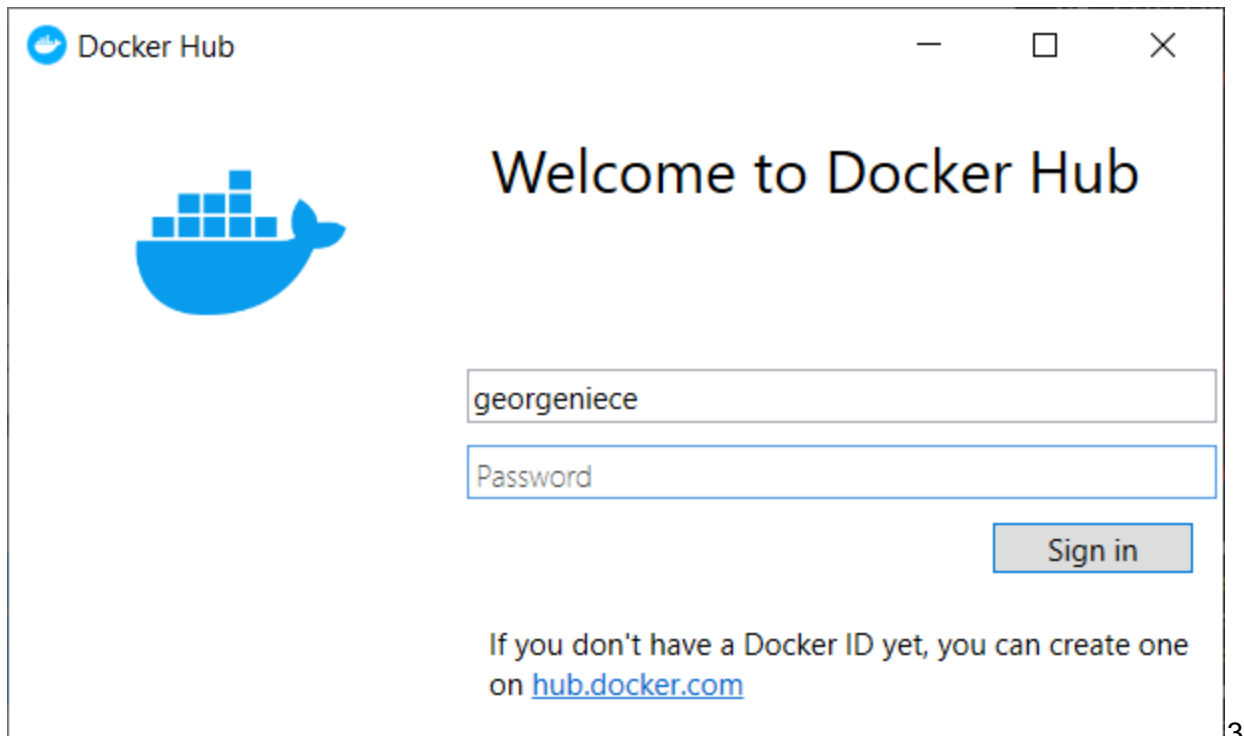
Done

```
host. All files and directories added to build
context will have '-rwxr-xr-x' permissions. I
t is recommended to double check and reset per
missions for sensitive files and directories.
PS C:\Users\wcsadmin\getting-started>
PS C:\Users\wcsadmin\getting-started>
PS C:\Users\wcsadmin\getting-started> cd getti
ng-started PS C:\Users\wcsadmin> git clone htt
ps://github
Set-Location : A positional parameter cannot
be found that accepts argument 'PS'.
At line:1 char:1
+ cd getting-started PS C:\Users\wcsadmin>
git clone https://github
+ ~~~~~
+ CategoryInfo          : InvalidArgumen
t: (:) [Set-Location], ParameterBindingE
xception
+ FullyQualifiedErrorId : PositionalPara
meterNotFound,Microsoft.PowerShell.Comma
nds.SetLocationCommand

PS C:\Users\wcsadmin\getting-started> .com/doc
ker/getting-started.gitPS C:\Users\wcsadmin> g
it clone https://github
>> .com/docker/getting-started.git*C
PS C:\Users\wcsadmin\getting-started> cd getti
ng-started ^C
PS C:\Users\wcsadmin\getting-started> docker
build -t docker101tutorial . docker run -d -p
80:80 --name docker-tutorial docker101tutorial

unknown shorthand flag: 'd' in -d
See 'docker build --help'.
PS C:\Users\wcsadmin\getting-started> docker r
un -d -p 80:80 --name docker-tutorial docker10
1tutorial
e5b72f602a04e81b4fe2fb3c797054f0948f645b018eb9
f9510d099d60e55822
PS C:\Users\wcsadmin\getting-started> 
```

Sign into your Docker Hub account, if you don't have one load [hub.docker.com](https://hub.docker.com) in a Web browser and create one.



3

That will update the screen for the next step in the quick start tutorial

Clone

Build

Run

4 Share

## Now save and share your image

Save and share your image on Docker Hub to enable other users to easily download and run the image on any destination machine.

```
docker tag docker101tutorial georgeniece/docker101tutorial
docker push georgeniece/docker101tutorial
```

Click [here](#) to see the image you shared on Docker Hub.

[Skip Tutorial](#)[Done](#)

```
host. All files and directories added to build
context will have '-rwxr-xr-x' permissions. I
t is recommended to double check and reset per
missions for sensitive files and directories.
PS C:\Users\wcsadmin\getting-started>
PS C:\Users\wcsadmin\getting-started>
PS C:\Users\wcsadmin\getting-started> cd getti
ng-started PS C:\Users\wcsadmin> git clone htt
ps://github
Set-Location : A positional parameter cannot
be found that accepts argument 'PS'.
At line:1 char:1
+ cd getting-started PS C:\Users\wcsadmin>
git clone https://github
+ ~~~~~
+ CategoryInfo          : InvalidArgument
t: (:) [Set-Location], ParameterBindingE
xception
+ FullyQualifiedErrorId : PositionalPara
meterNotFound,Microsoft.PowerShell.Comma
nds.SetLocationCommand

PS C:\Users\wcsadmin\getting-started> .com/doc
ker/getting-started.gitPS C:\Users\wcsadmin> g
it clone https://github
>> .com/docker/getting-started.git^C
PS C:\Users\wcsadmin\getting-started> cd getti
ng-started ^C
PS C:\Users\wcsadmin\getting-started> docker
build -t docker101tutorial . docker run -d -p
80:80 --name docker-tutorial docker101tutorial

unknown shorthand flag: 'd' in -d
See 'docker build --help'.
PS C:\Users\wcsadmin\getting-started> docker r
un -d -p 80:80 --name docker-tutorial docker10
1tutorial
e5b72f602a04e81b4fe2fb3c797054f0948f645b018eb9
f9510d099d60e55822
PS C:\Users\wcsadmin\getting-started> 
```

Click the Command Button to tag our tutorial image and push to Docker Hub.

```
$ docker tag docker101tutorial georgeniece/docker101tutorial
```

```
$ docker push georgeniece/docker101tutorial
```

The push refers to repository [docker.io/georgeniece/docker101tutorial]

After the push completes we can check the status of the docker tutorial container we started with

```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS			
PORTS	NAMES		
e5b72f602a04	docker101tutorial	"/docker-entrypoint...."	13 minutes ago
minutes	0.0.0.0:80->80/tcp	docker-tutorial	Up 13 minutes

If we open a web browser to <https://hub.docker.com/repositories> we'll see the image under the repo we just created.

Finally, we can open docker desktop and stop that

Now from a Windows command line noted next, or MacOS terminal prompt shown below, we can start the getting started docker container

```
$ docker run -d -p 80:80 docker/getting-started
```

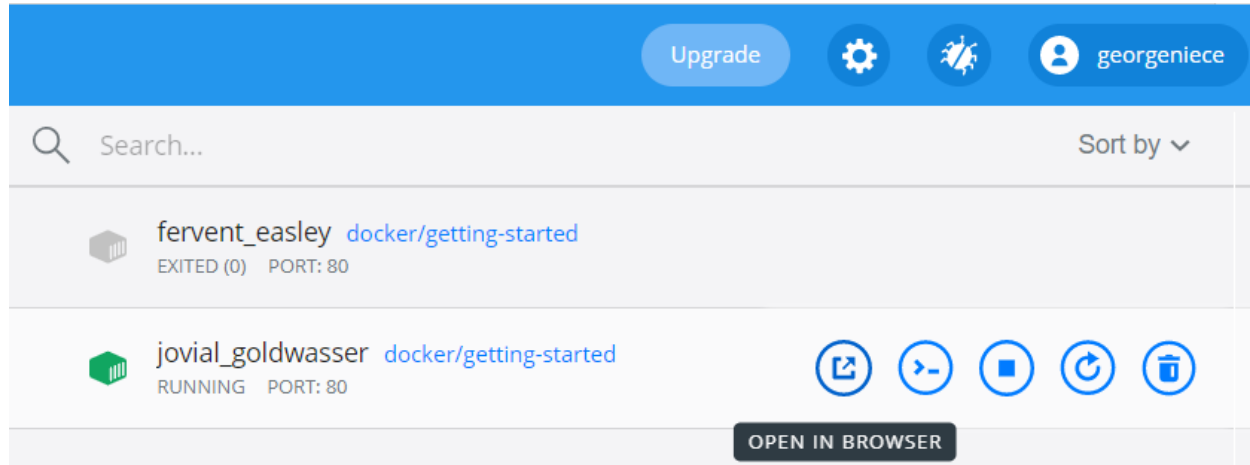
You'll notice a few flags being used. Here's some more info on them:

- -d - run the container in detached mode (in the background)
- -p 80:80 - map port 80 of the host to port 80 in the container
- docker/getting-started - the image to use

Or we could combine the flags as is shown for MacOS next

```
:~ kubenerd$ docker run -d -p 80:80 docker/getting-started
```

Now open a browser window to <http://localhost>, or this can be launched from Docker desktop as shown below. On Windows you would see the Docker Whale in the System Tray, for MacOS you can find it in Apps or if you pinned in the Dock. Since we didn't name the container a randomized name was selected for us as is shown with **jovial\_goldwasser**, although your naming would likely be different.



We could also start a docker container mounting a volume locally.

## In Windows:

Using a valid folder in a CMD prompt, we set the var for the present working directory, this would be set by default in MacOS

```
C:\projects>set PWD=C:\projects
```

Now we start an NGINX container named "web" that has port redirect and maps a volume into the container from the folder we've just set

```
C:\projects>docker container run --name web2 -d -p 8070:80 --mount  
type=bind,source=%PWD%,target=/usr/share/nginx/html -d nginx
```

```
d8410fc7874dcafb0845a5e410b7b27f36708cbc70198c69064073dbf8ca1b7a0f2eafbf5ccb3cc96  
8ef12412ea03af4328949f33a97bf27022356ae397d9b1a
```

Alternatively if we were using Git Bash as noted we could run the container similar to what we see in the MacOS section below.

## In MacOS:

Start the NGINX container

```
:~ kubenerd$ docker container run --name web -d -p 8080:80 -v  
$PWD:/usr/share/nginx/html:ro -d nginx
```

In a browser on your Windows or MacOS development environment:

Load <http://localhost:8080> in a browser



Note that we're receiving a 403, unless we happened to already have an index.html file in the present folder (PWD) that we shared as a volume into the NGINX container we created.

We'll need to create one, open your favorite text code editor, in the following example I use VSCode, after setting the executable in the path, or alternatively VIM, both of which are available on Windows or MacOS

```
C:\projects>set Path=%Path%C:\Program Files\Microsoft VS Code\
```

```
C:\projects> Code index.html
```

Or we could just as easy use VIM

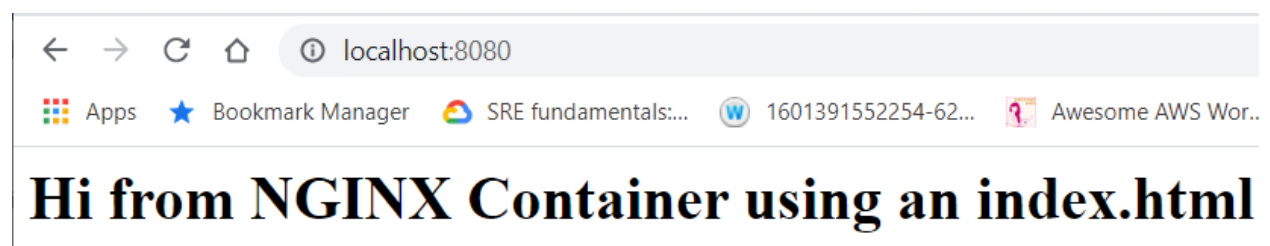
```
:~ kubenerd$ vim index.html
```

Add the following line to your index.html

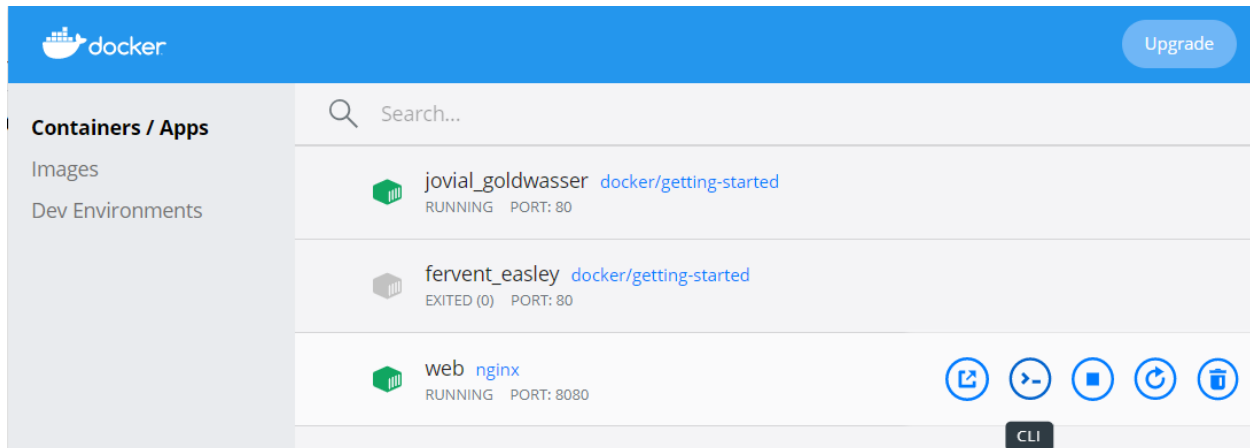
```
<html><head><title>NGINX Test Page</title></head><body><H1>Hi from NGINX  
Container using an index.html in your local development  
environment</h1></body></html>
```

Save the file

Reload the web page for NGINX that had the 403



## Run the CLI into the container from Docker Desktop



This will open a terminal in the container. Let's take a quick look at the file we've created and then shared into the container through a volume mounting

```
# cd /usr/share/nginx/html
```

```
# ls -al index.html
```

```
-rw-r--r-- 1 root root 63 Mar 6 17:33 index.html
```

```
# cat index.html
```

```
<html><head><title>NGINX Test Page</title></head><body><H1>Hi from NGINX  
Container using an index.html in your local development environment</h1></body></html>
```