Experiment 01: Foundation Installation

Install Kind AKA KiND AKA KIND (**K**ubernetes **in D**ocker)

For MacOS

First we install Rancher Desktop for MAcOS

```
$ brew install --cask rancher
```

```
When you run the application it will ask which container engine to
use. Select dockerd for this session, as well I opted for 1.22 version
of K8s, you could use an older or newer version, but obviously there
can be some nuanced issues that we'll work through together, if
needed.
```

Next we install kind (**K**ubernetes **in D**ocker).

$ **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 Rancher Desktop. Kind will automagically install Docker Desktop, if Rancher Desktop is not installed. Chocolatey is a package manager for Windows like Brew is for MacOS and NPM for NodeJS.

Download Rancher Desktop for Windows from https://rancherdesktop.io/

Install Rancher Desktop choosing the "dockerd" container runtime and 1.22 version of Kubernetes (K8S)

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.SetExecutionPolicyComma
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.

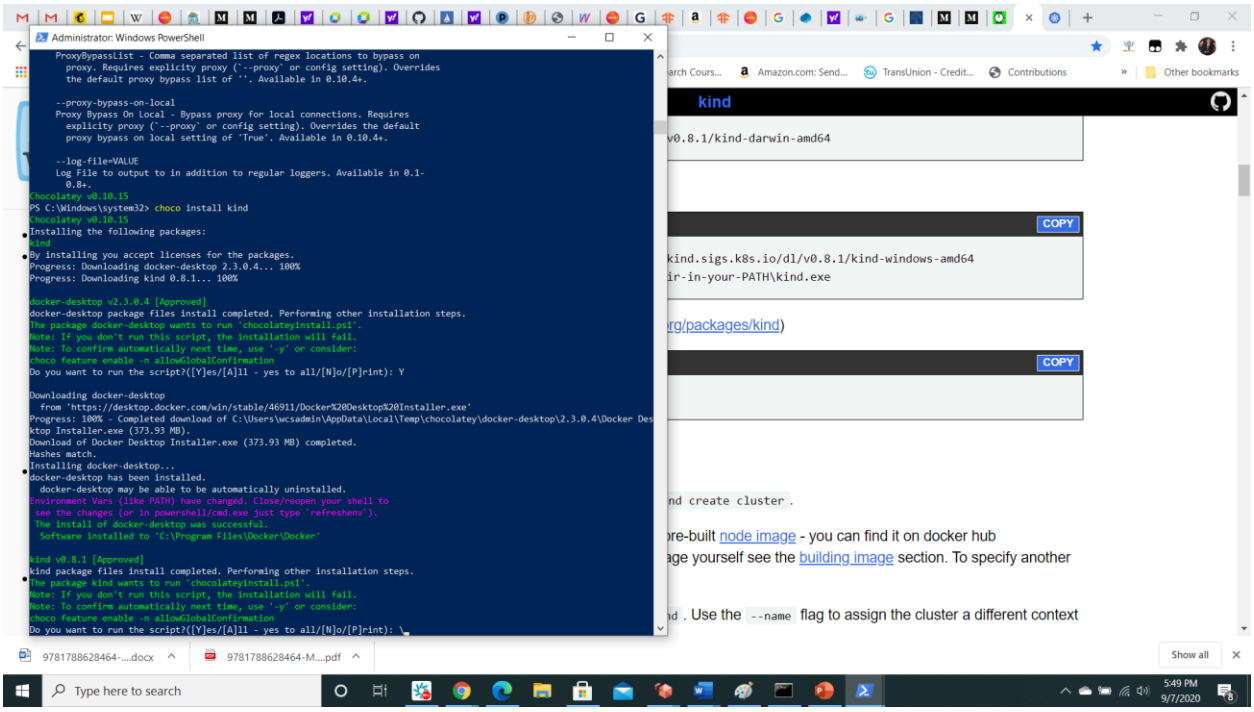
Next we'll use Chocolatey to install kind (**K**ubernetes **in D**ocker)

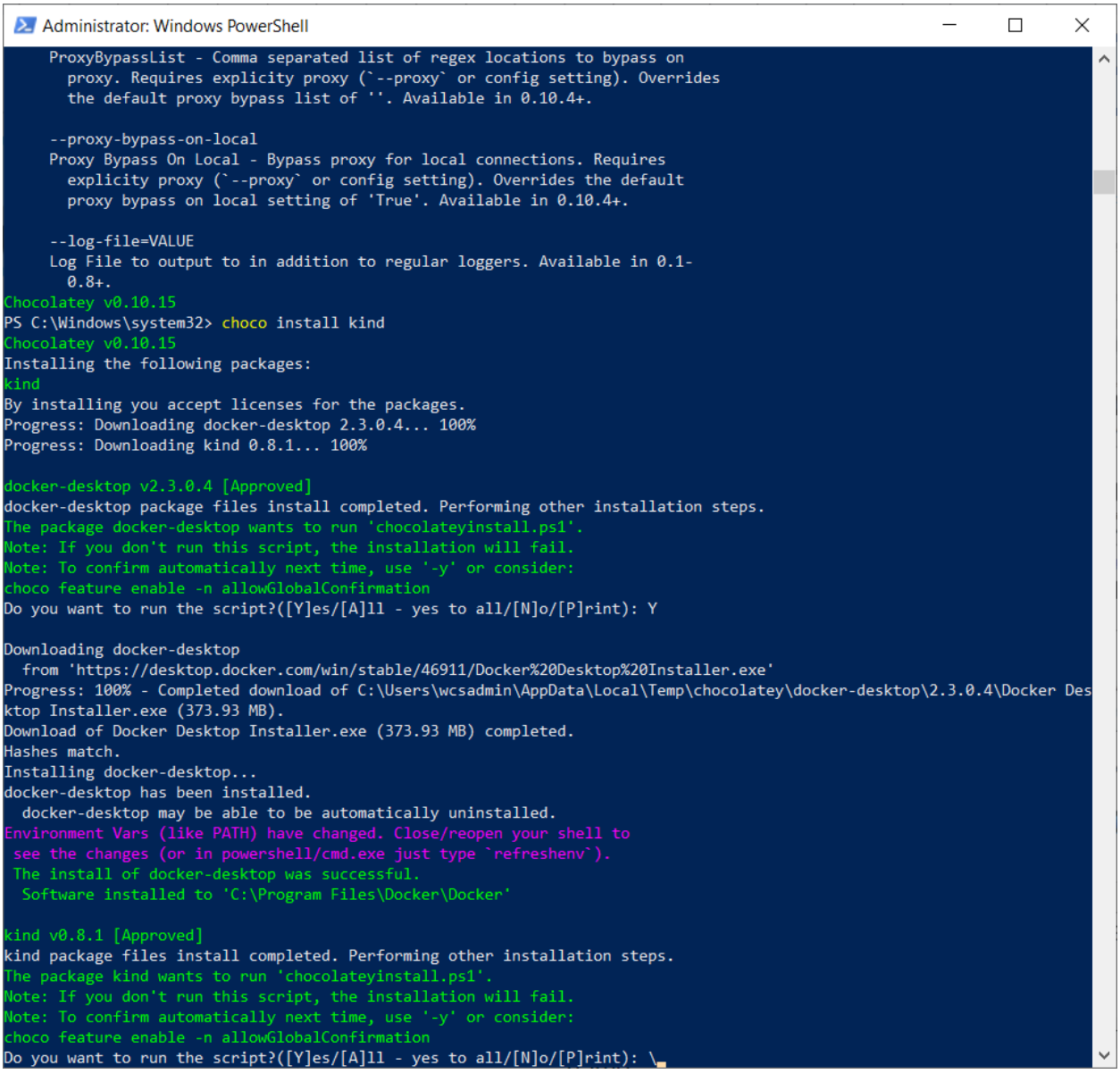
PS C:\Users\kubelord> **choco install kind**

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%

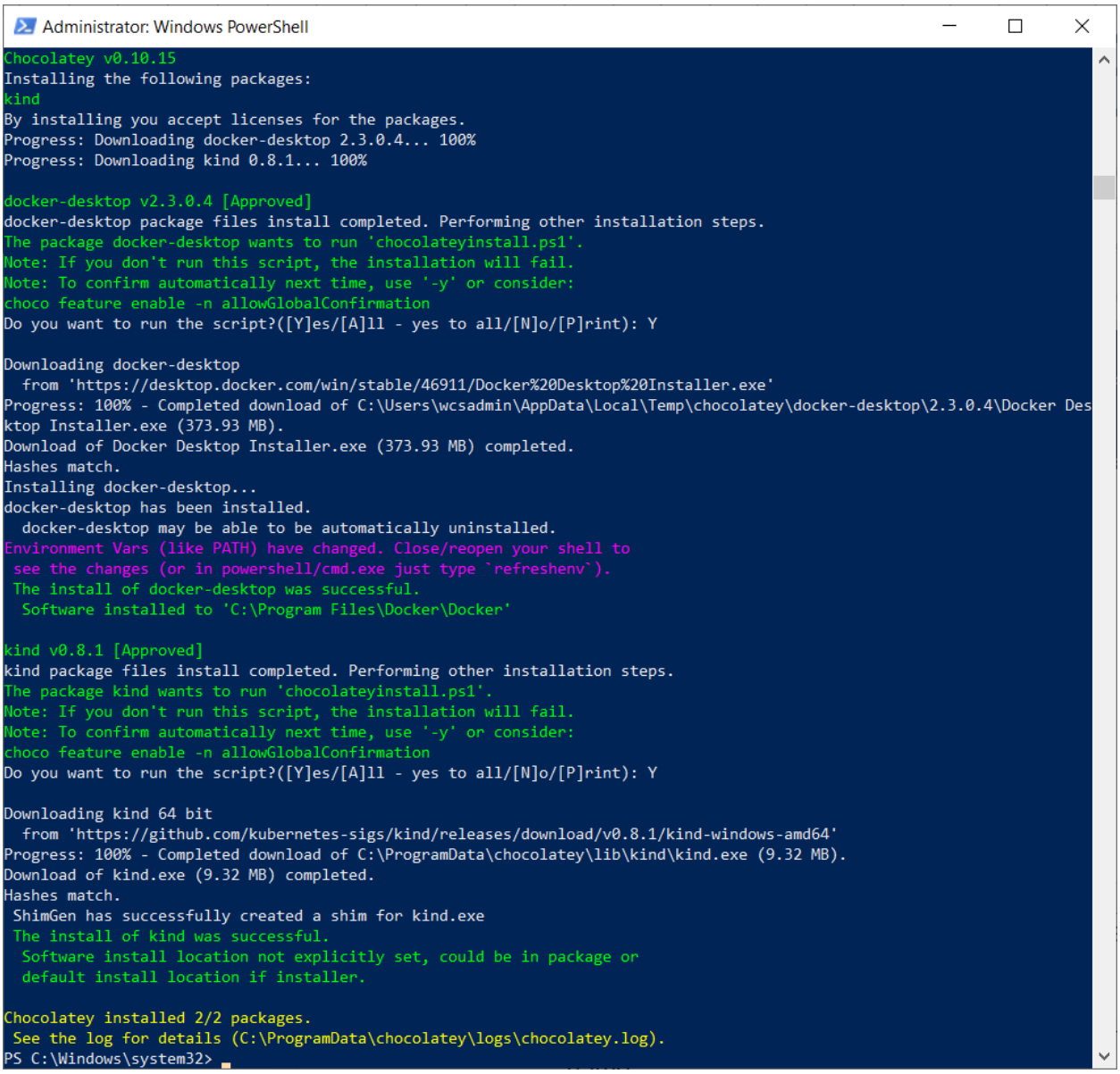
docker-desktop v2.3.0.4 [Approved]
…
The install of docker-desktop was successful.
Software installed to 'C:\Program Files\Docker\Docker'

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/

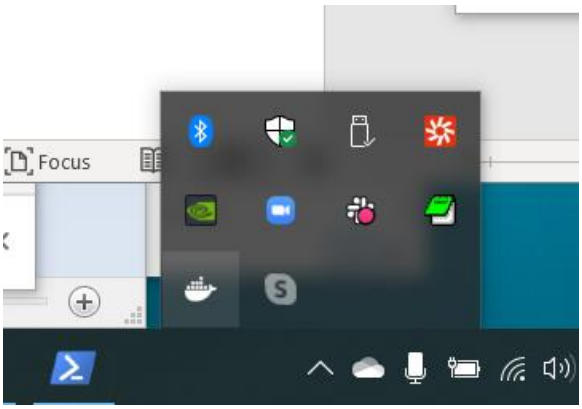




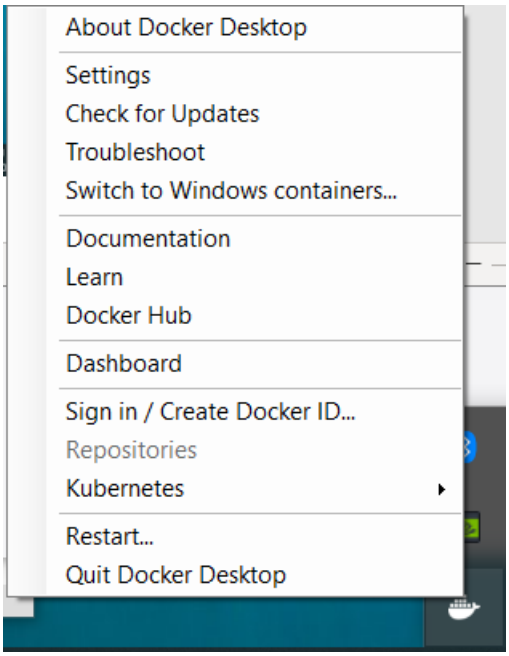
docker ps -a --filter label=io.x-k8s.kind.cluster=kind --format '{{.Names}}'



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/

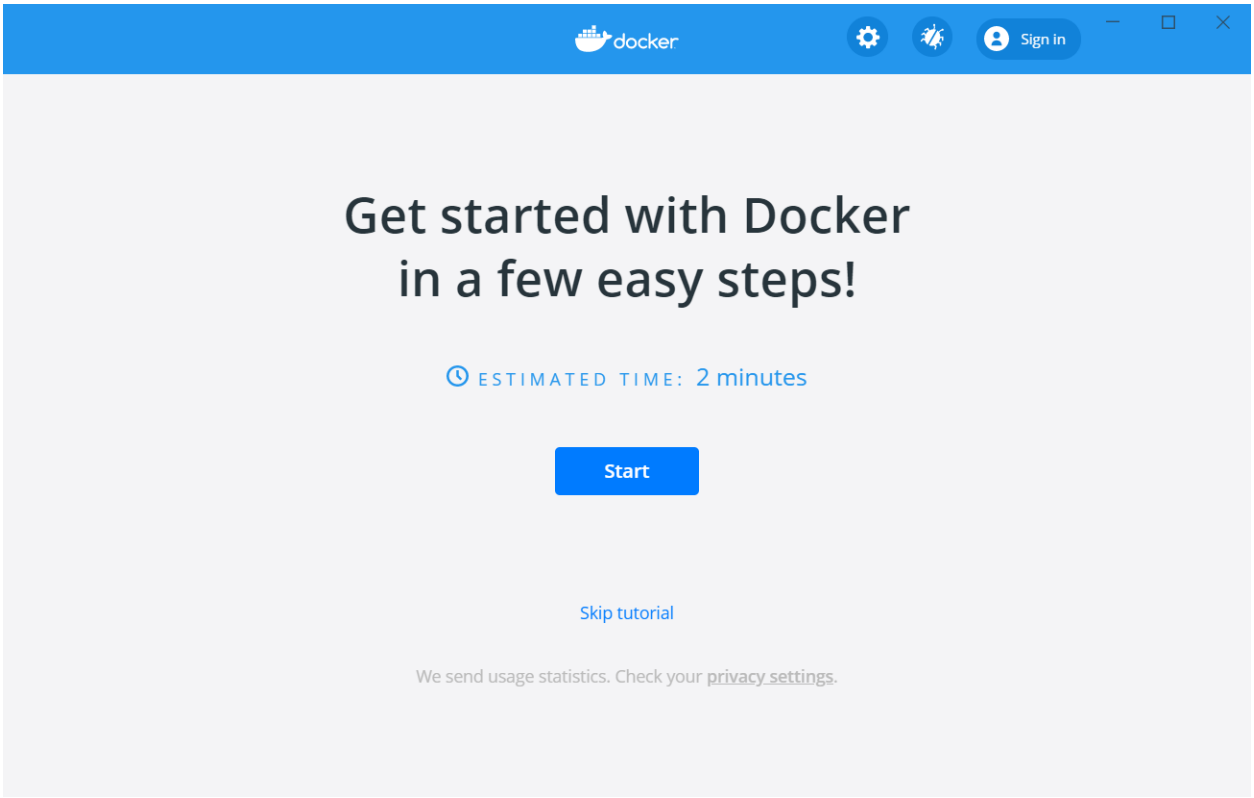
For Windows we're using Chocolatey for the installation

```
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%

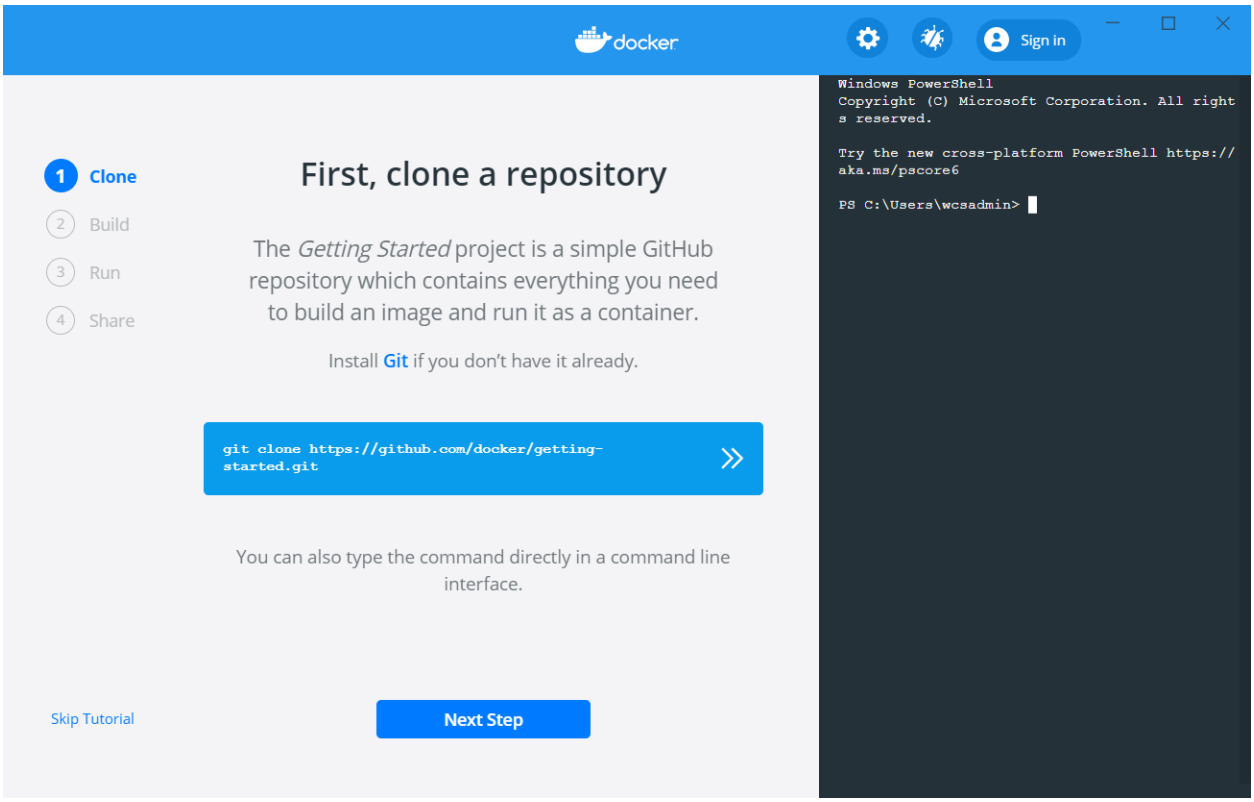
Open a Git Bash terminal in your Projects folder, create the folder if it does not already exist

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



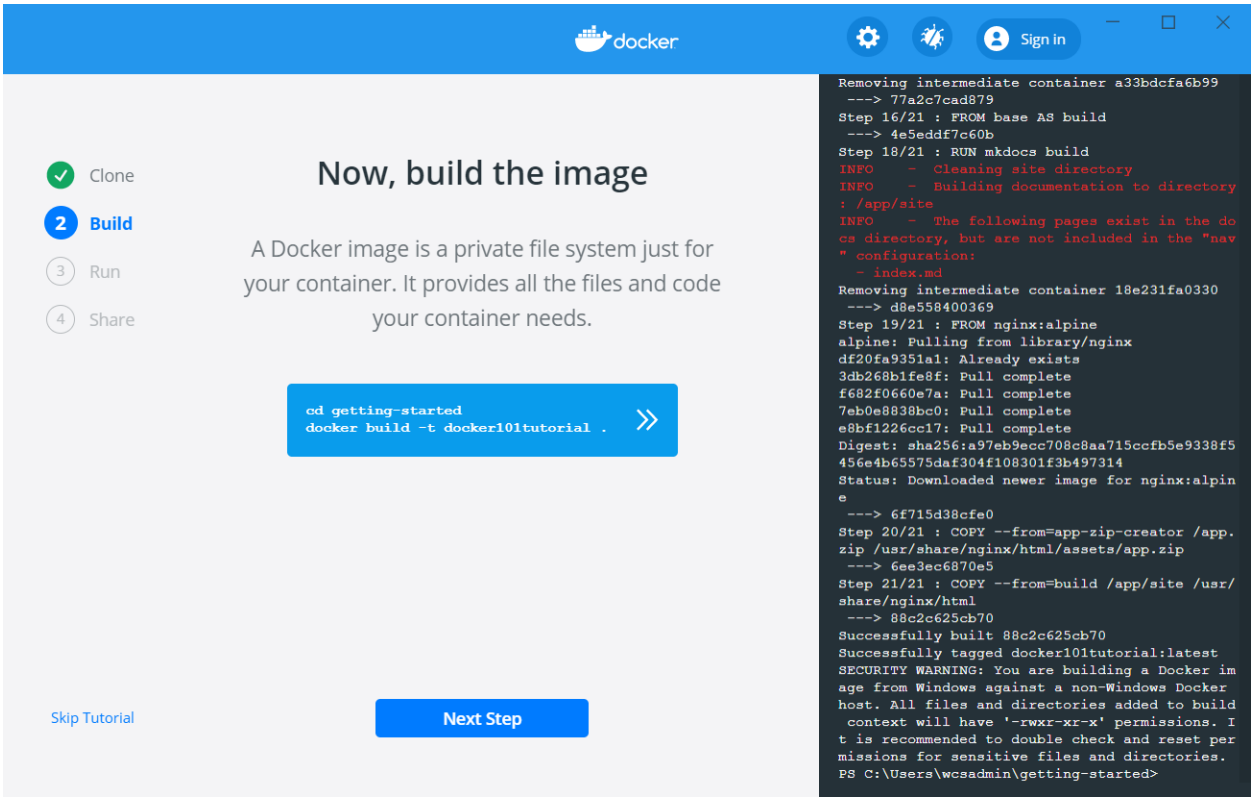
If you have Git installed, which is necessary for the class continue to click the Command Button and it will be pasted automatically (like Katacoda) into the Powershell window and clone the repo for the docker getting started.

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

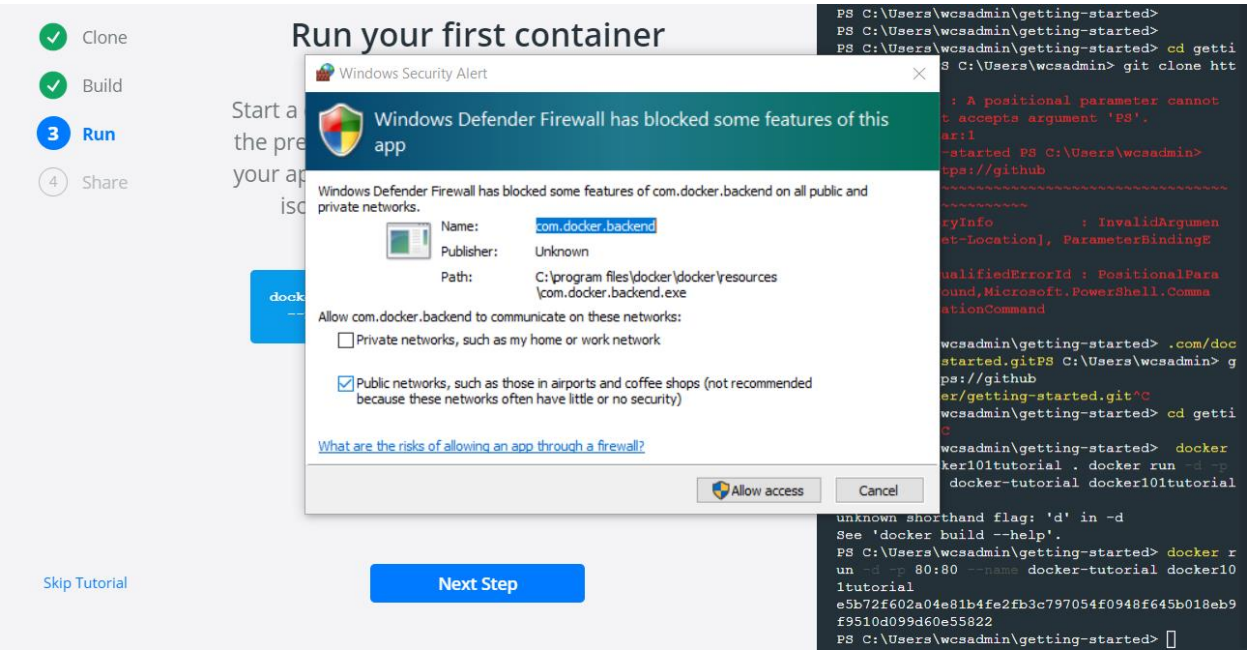
$ mkdir getting-started

$ cd getting-started

$ docker build -t docker101tutorial .



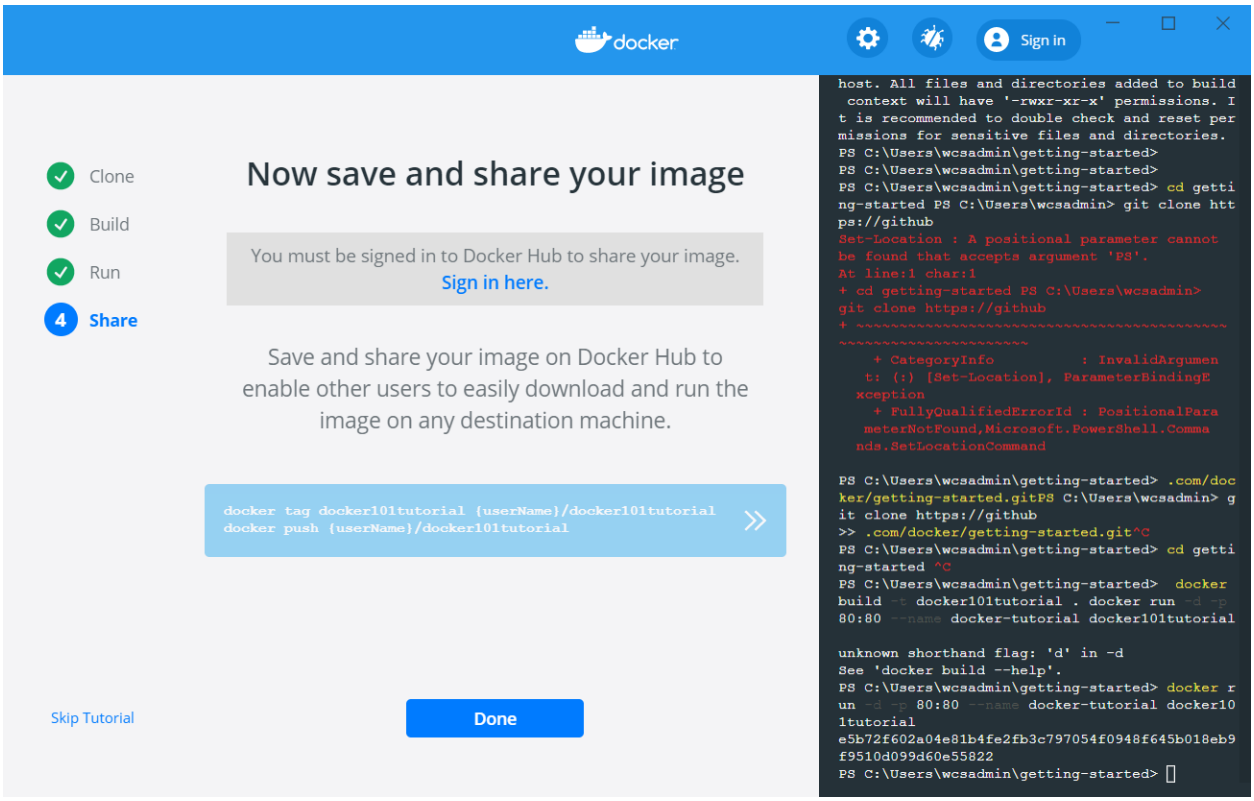
$ **docker run -d -p 80:80 --name docker-tutorial docker101tutorial**



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

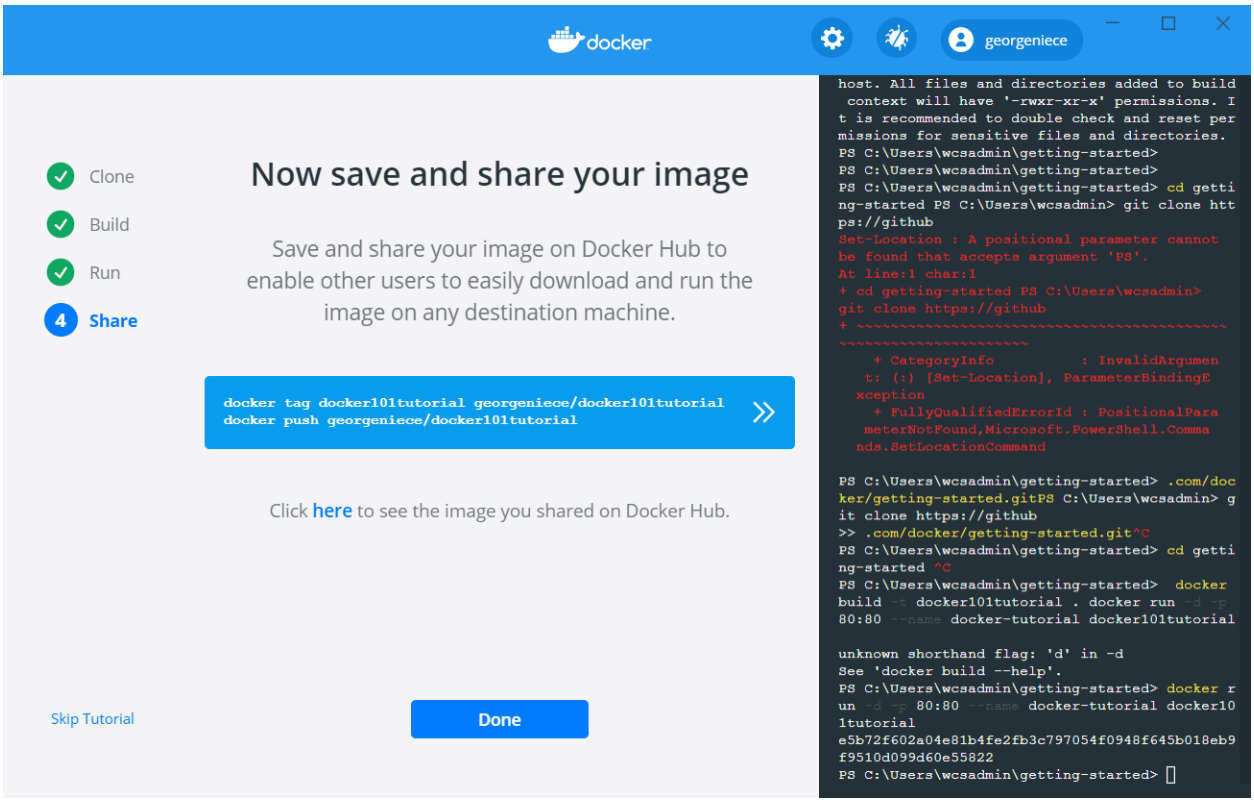
e5b72f602a04e81b4fe2fb3c797054f0948f645b018eb9f9510d099d60e55822



Sign into your Docker Hub account, if you don’t have one load 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



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      Up 13
minutes       0.0.0.0:80->80/tcp   docker-tutorial
```

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

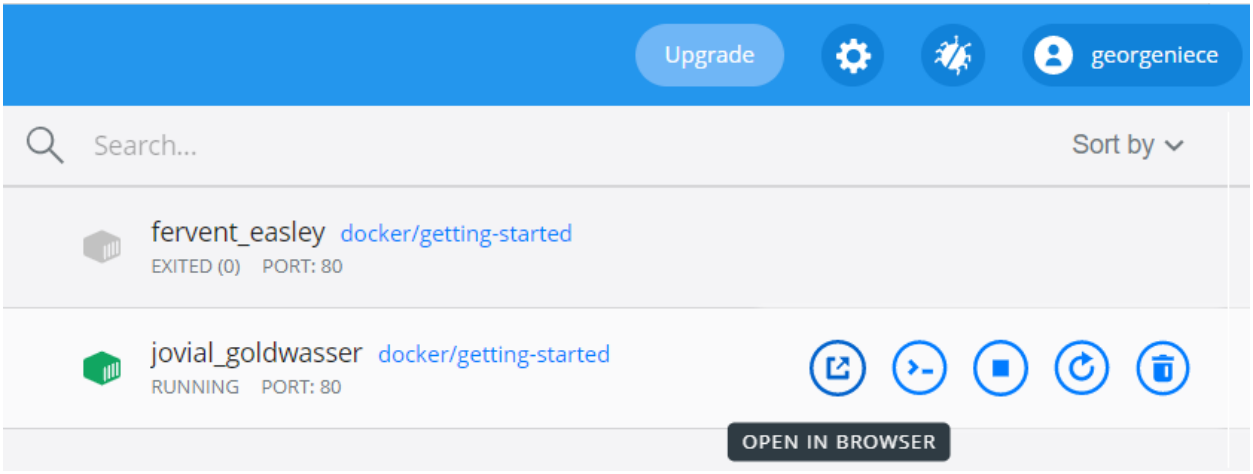
You'll notice a few flags being used. Here's some more info on them:

- -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**

d8410fc7874dcafb0845a5e410b7b27f36708cbc70198c69064073dbf8ca1b7a0f2eafbf5ccb3cc968ef12412ea03af4328949f33a97bf27022356ae397d9b1a

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

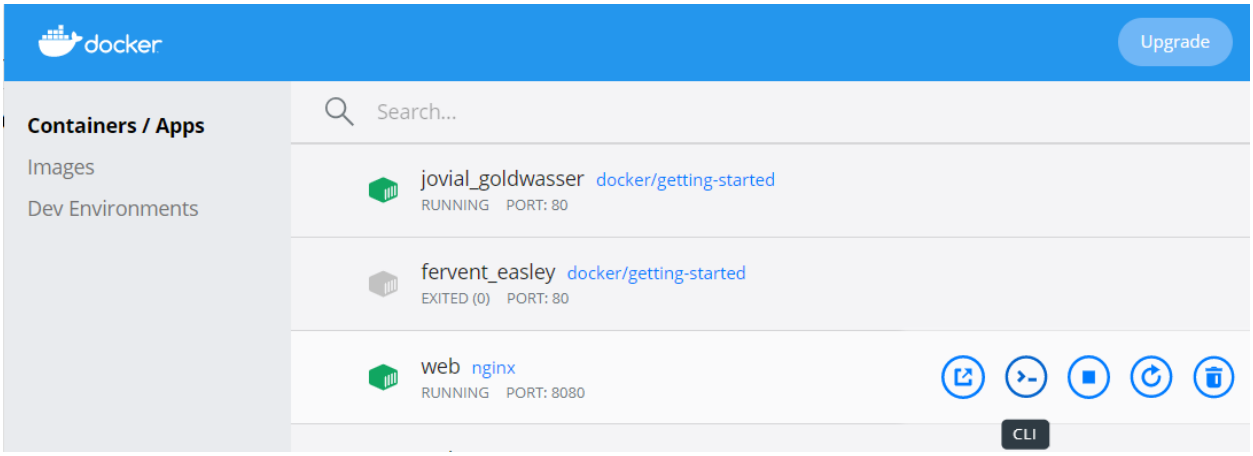
Add the following line to your index.html

<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>