Docker Tutorials

Read/Watch the required knowledge, and complete each section’s activity

Contents

[1: Setup docker and test working 1](#_Toc113014966)

[2: Docker Volumes: Run a local website 2](#_Toc113014967)

[3: Wrapping your scripts in Docker: aka portable fib 3](#_Toc113014968)

[4: Pushing docker images into external registry 4](#_Toc113014969)

[5: Docker Best Practices 5](#_Toc113014970)

[6: Running a big app using multiple applications: Docker composer 6](#_Toc113014971)

[7: Running a big app using multiple applications inside a Cluster: Kubernetes 7](#_Toc113014972)

# 1: Setup docker and test working

**Required Knowledge:**

**None**

**Activities:**

1) Follow instructions to install and setup Docker   
<https://docs.docker.com/get-docker/>

2) Run a simple “hello-world” docker image, to make sure your setup is working

$ docker version

Client:

Cloud integration: v1.0.22

Version: 20.10.11

...

$ docker run hello-world

Hello from Docker!

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

# 2: Docker Volumes: Run a local website

**Required Knowledge:**  
Read Docker Volume guides  
<https://docs.docker.com/storage/volumes/>

<https://www.baeldung.com/ops/docker-volumes>

Watch guide to docker Volume guide  
<https://www.youtube.com/watch?v=p2PH_YPCsis>

**Activities:**

Use the “nginx” Docker image to run a local website

* Create a new folder, and create a file called index.html with the following content  
    
  <html lang="en">

<body>

<h2>Hello World</h2>

This is a webpage

</body>

</html>

* Run the nginx website to run the folder with “index.html”

$ docker run -t --rm -p 8080:80 -v ***<DIR>***:/usr/share/nginx/html nginx

Hint: in git-bash this will run the server mounting the current directory

docker run -it --rm -d -p 8080:80 --name web -v $(pwd -W):/usr/share/nginx/html nginx

* Go to webserver in browser “localhost:8080”

Graphical user interface, text, application

Description automatically generated

**Bonus Activities:**

Run website in the background by using the -detach and the -name  
<https://docs.docker.com/engine/reference/commandline/run/>

Remember to clean up any background tasks afterwards using ps/stop commands  
$ docker ps  
$ docker stop xxx

# 3: Wrapping your scripts in Docker: aka portable fib

**Required Knowledge:**  
Read sections  
<https://docs.docker.com/engine/reference/builder/>   
  
Watch dockerfile tutorial  
<https://www.youtube.com/watch?v=LQjaJINkQXY>   
  
Read Tutorial and complete the “Let’s create your first image”  
<https://takacsmark.com/dockerfile-tutorial-by-example-dockerfile-best-practices-2018/#lets-create-your-first-image>

**Activities:**

Create a docker image containing the fib bash script from bash tutorials, and make this run from inside a docker image

* Create dockerfile containing fib.sh which runs as a ENTRYPOINT
* Build docker image  
  $ docker build --tag fib-cli .
* Run docker image  
  $ docker run fib-cli 6

fib number 6 is 13

ps if you’re using read you’ll need to add -it to docker “interactive”

* Run docker image  
  $ docker run -it fib-cli  
  which docker number ?  
  6

fib number 6 is 13

Troubleshooting:  
“syntax error near unexpected token `$'\r''” fix windows newline endings with  
sed -i 's/\r$//' fib.sh

<https://askubuntu.com/questions/966488/how-do-i-fix-r-command-not-found-errors-running-bash-scripts-in-wsl>

“[: 0: unary operator expected” fix input/output for git bash with this command

winpty docker run -it xxx

# 4: Pushing docker images into external registry

**Required Knowledge:**  
Read all follow Dockerhub Quick start  
(do upload their basic example)  
<https://docs.docker.com/docker-hub/>   
  
NOTE: in professional life will probably not use Dockerhub, but a bespoke cloud or corporate bespoke equivalent register aka ECR or ACR

**Activities:**

1) Signup for Docker hub using your riversafe email  
<https://hub.docker.com/>

2) Take your fib image from the previous activity and upload into dockerhub

* Build fib image tag  
  $ docker build --tag fib-cli . -t xxx/fib-cli:0.1 -t xxx/fib-cli:latest
* Upload to dockerhub  
  $ docker push xxx/fib-cli:0.1

$ docker push xxx/fib-cli:latest

* Run fib from registry   
  $ docker run xxx/fib-cli:0.1 6

fib number 6 is 13

# 5: Docker Best Practices

**Required Knowledge:**  
Read best practice guides  
<https://docs.docker.com/develop/develop-images/dockerfile_best-practices/>

<https://docs.docker.com/develop/dev-best-practices/>

<https://takacsmark.com/dockerfile-tutorial-by-example-dockerfile-best-practices-2018/#dockerfile-best-practices>

<https://sysdig.com/blog/dockerfile-best-practices/>

**Activities:**

Review previous docker image and apply best practices  
  
Given simplicity of fib App unlikely to be many issues, but here are some common improvement points:

* Are you running as ROOT ?  
  (most image bases need you to call USER )  
  <https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#user>
* Are you running a large linux instance, try alpine instead  
  <https://hub.docker.com/_/alpine/>   
  <https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#from>
* Have you added comments are labels?  
  <https://docs.docker.com/engine/reference/builder/#label>
* Have you included surplus files?  
  <https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#exclude-with-dockerignore>
* Create ephemeral containers   
  Unlikely for us, but Docker images should be stateless\*  
  Basically expect the app to be reset in-between runs, stateful data need stored in mounted drives or accessed services  
  <https://docs.docker.com/develop/dev-best-practices/#where-and-how-to-persist-application-data>   
    
  \* read the 12 factor app for more info  
  <https://12factor.net/>

# 6: Running a big app using multiple applications: Docker composer

*Note: Docker compose is unlikely to the main orchestration service used in other companies, many now run their multi image applications using Kubernetes, Terraform or other similar services  
  
However Docker compose is an excellent and simple introduction to the field of multi-image applications*

**Required Knowledge:**

Read sections  
<https://docs.docker.com/compose/>

**Activities:**

Run the docker compose tutorial and create a simple application with a redis caching backend  
<https://docs.docker.com/compose/gettingstarted/>

# 7: Running a big app using multiple applications inside a Cluster: Kubernetes

*Note: Kubernetes section is just to get your feet wet with Kubernetes, it takes months to learn and become good at this complex technology  
  
Just getting a running cluster is a major feat, below will help you install Kubernetes locally, learn some of the basic concepts, and run a simple webserver inside Kubernetes*

**NEEDS RE-WORKED TUTORIAL TOO HARD FOR NEW STARTERS**

**Required Knowledge:**

Read sections

<https://kubernetes.io/docs/tutorials/>  
<https://kubernetes.io/docs/tutorials/kubernetes-basics/>

<https://medium.com/google-cloud/kubernetes-101-pods-nodes-containers-and-clusters-c1509e409e16>

**Activities:**  
Install Kubernetes locally  
<https://kubernetes.io/releases/download/>   
<https://kubernetes.io/docs/tasks/tools/>   
<https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/>

Install Minikube  
<https://minikube.sigs.k8s.io/docs/start/>

Complete hello-world Kubernetes webserver tutorial

<https://medium.com/@mngaonkar/kubernetes-get-started-deploy-a-simple-web-server-9636f4aa8706>   
  
When you can see the web page of the ngnix-service the tutorial is complete  
get port using  
$ kubectl get svc  
  
view in browser  
<http://localhost:xxx>