Docker Cheatsheet by Vikas Naik

all core Docker features including Dockerfiles and Docker Compose.

> *writing down all the important commands and side notes for future reference.*

Tech

- Docker - Docker is a platform designed to help developers build, share, and run modern applications!
- Salesforce - Application as a Service/ Platform as a Service
- node.js - evented I/O for the backend
- Express - fast node.js network app framework @tjholowaychuk
- AngularJS - HTML enhanced for web apps!
- Ace Editor - awesome web-based text editor
- Dillinger - Markdown Editor
- markdown-it - Markdown parser done right. Fast and easy to extend.
- Twitter Bootstrap - great UI boilerplate for modern web apps
- Gulp - the streaming build system
- Breakdance - HTML to Markdown converter
- jQuery - duh.

Docker

1. Basic docker containers and commands:

```
docker run --name hellWorld hello-world  // this will download hello-world image from remote registry of dockerhub and runs the container.
```

> *Note:*
> *-the `run` in above command creates new container and runs the container.*
> *-if no process running in the container, the container will get terminated immediately.*
> *-the flag `--name hellWorld` is used to give a name to the container. If we don't provide a name to the container, a random name is assigned to the container.*
> *-it is not possible to create two container with same name.*

```
docker pull alpine // this will only download the image
```

```
docker ps // shows all running containers
```

> *- `docker container ls` is an alternative command to `docker ps`*
> *- `docker ps` commands shows containerid, container name, port and the command that's immediately executed inside the container.*

```
docker ps -a  // shows all past executed containers and their shell
```

```
docker stop D7 //  stops the running containlner, put only two chars of containerID
```

> *Note: we can put container name also instead of container id. All running container info is shown by using `docker ps` command.*

```
docker run -it ubuntu // enters bash shell of ubuntu. Enter exit to exit the bash shell.
```

```
docker images // shows all downloaded images with its size.
```

---

2. Volume mapping and port mapping:

```
docker run nginx // runs nginx container with port 80 opened
```

```
docker run -p 8080:80 nginx
```

> *-here with `-p 8080:80` we have opened port 8080 of the system and mapped it with container's port 80.*
> *-all request to port 8080 is forwarded to port 80 of the container.*
> *-now localhost:8080 will work.*

```
docker run -p 8080:80 -v /Users/vikasnaikmacbkpro/CodeWorkspace/Nginx:/usr/share/nginx/html nginx
```

> *-here `/Users/vikasnaikmacbkpro/CodeWorkspace/Nginx` is local system directory path and `/usr/share/nginx/html` is container's directory path.*

List of all restricted ports on chrome:

Port	process
1	// tcpmux
7	// echo
9	// discard
11	// systat
13	// daytime
15	// netstat
17	// qotd
19	// chargen
20	// ftp data
21	// ftp access
22	// ssh

23	// telnet
25	// smtp
37	// time
42	// name
43	// nicname
53	// domain
69	// tftp
77	// priv-rjs
79	// finger
87	// ttylink
95	// supdup
101	// hostriame
102	// iso-tsap
103	// gppitnp
104	// acr-nema
109	// pop2
110	// pop3
111	// sunrpc
113	// auth
115	// sftp
117	// uucp-path
119	// nntp
123	// NTP
135	// loc-srv /epmap
137	// netbios
139	// netbios
143	// imap2
161	// snmp
179	// BGP
389	// ldap
427	// SLP (Also used by Apple Filing Protocol)
465	// smtp+ssl

512	// print / exec
513	// login
514	// shell
515	// printer
526	// tempo
530	// courier
531	// chat
532	// netnews
540	// uucp
548	// AFP (Apple Filing Protocol)
554	// rtsp
556	// remotefs
563	// nntp+ssl
587	// smtp (rfc6409)
601	// syslog-conn (rfc3195)
636	// ldap+ssl
993	// ldap+ssl
995	// pop3+ssl
1719	// h323gatestat
1720	// h323hostcall
1723	// pptp
2049	// nfs
3659	// apple-sasl / PasswordServer
4045	// lockd
5060	// sip
5061	// sips
6000	// X11
6566	// sane-port
6665	// Alternate IRC [Apple addition]
6666	// Alternate IRC [Apple addition]
6667	// Standard IRC [Apple addition]
6668	// Alternate IRC [Apple addition]

6669	// Alternate IRC [Apple addition]
6697	// IRC + TLS
10080	// Amanda

### 3. Container Management:	

```
docker run -p 8080:80 -d nginx // runs the container in the background
```

> *-the `-d` flag in above command runs the container in the background and all the logs are stored to review it later.*
> *-above command will return the SHA Hashcode and run the container in background.*
> *-we can use command `docker log 922590` to view log of container running in the background.*

```
docker log 922590 // shows log of the container having id 922590, put only first few chars of the containerid running in the background.
```

```
docker run --name ubuntu1 ubuntu //
```

> *Note: -with above command we can create multiple container from same image.*
> *-all containers will have different environment with diff file structure but will share same DOCKER HOST resources among them.*
> *-all containers will have different ip address but will be in same network as of network bridge of local system (DOCKERHOST).*

```
docker start ubuntu1 // start the existing container which was stopped in the past
```

> *-the `start` in above command is used to start the existing container which is currently not running.*
> *- `ubuntu1` in above command is the container name that we want to start.*
> *-with `docker start` command container will be started exactly with the same configuration as it was running with before exit.(ie port mapping, volume mapping, container name)*

```
docker rm ubuntu1 // delete the existing container with name ubuntu1
```

```
docker container prune  // deletes all stopped containers
```

```
docker container rm 05 d7 // deletes two stopped containers having containerids starting with 05 and d7
```

4.A. Node.js in Docker:

```
docker pull node // downloads nodejs image
```

```
docker run -it node  // create new container from node image in interactive mode
```

```
docker run -v $PWD:/app -w /app node node hello.js
```

> *Note:*
> *-in above command we have mapped our local system current directory with node containers `/app` directory.*
> *-the `-w /app` indicates that we want to set `/app` directory of container as a working directory.*
> *-the first `node` in the command is the name of the image.*
> *-after the first node, every thing ie `node hello.js` is the command to be executed inside the container.*

4.B. Express.js in Docker:

```
docker run -v $PWD:/app -w /app -it node npm init
```

> *Note:*
> *-above command will create package.jon file in /app directory of container but bcoz it mapped to our local directory, we can see this file in our local system directory.*

```
docker run -v $PWD:/app -w /app -it node npm install express
```

> *Note:*
> *-above command will install expressjs module.*

```
docker run -v $PWD:/app -w /app -it -p 8081:3000 node node index.js
```

> *Note:*
> *-above command will map local system port 8081 with containers expressjs port 3000.*
> *-the node.js is an expressjs application.*

> *to handle control C (^c) to terminate the expressjs server application, write below lines of code in the expressjs application server.*

```
const process = require('process')
process.on('SIGINT',() => {
   console.log('Application is being interrupted...')
   process.exit(0)
})
```

> *to terminate the express server when we stop the container write below line of code*

```
const process = require('process')
process.on('SIGTERM',() => {
   console.log('Application is being terminated...')
```

```
   process.exit(0)
})
```