EECS 3482 Lab 5

Jonathan Baldwin

Chapter 1

Part 1

Download and Install Docker Toolbox

(skipped, as I am using native Docker on a Linux host)

Pull the hello-world sample application:

```
% sudo docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:2557e3c07ed1e38f26e389462d03ed943586f744621577a99efb77324b0fe535
Status: Downloaded newer image for hello-world:latest
```

Run the hello-world application:

```
% sudo docker run hello-world
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.
```

```
To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Determine the Docker version

```
% docker --version
Docker version 18.09.3-ce, build 774a1f4eee
```

Retrieve information about Docker installation

```
% sudo docker info
Containers: 1
 Running: 0
 Paused: 0
 Stopped: 1
Images: 1
Server Version: 18.09.3-ce
Storage Driver: overlay2
 Backing Filesystem: tmpfs
 Supports d_type: true
 Native Overlay Diff: false
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 9f2e07b1fc1342d1c48fe4d7bbb94cb6d1bf278b.m
runc version: ccb5efd37fb7c86364786e9137e22948751de7ed-dirty
init version: fec3683
Security Options:
 seccomp
  Profile: default
Kernel Version: 5.0.0-arch1-1-ARCH
```

```
Operating System: Arch Linux
OSType: linux
Architecture: x86_64
CPUs: 8
Total Memory: 15.52GiB
Name: arch-desktop
ID: O3EF:QEYN:ZBH6:EDUY:JBIC:SUKL:OH66:7NIV:NRXH:OOIN:YVHH:KCYJ
Docker Root Dir: /tmp/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false
```

List the available Docker images

```
% sudo docker image ls
REPOSITORY   TAG     IMAGE ID      CREATED       SIZE
hello-world  latest  fce289e99eb9  2 months ago  1.84kB
```

List CLI commands

```
% docker
Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default "/root/.docker")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level
                           ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default
                           "/root/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default
                           "/root/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default "/root/.docker/key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit
```

```
Management Commands:
  builder     Manage builds
  config      Manage Docker configs
  container   Manage containers
  engine      Manage the docker engine
  image       Manage images
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  secret      Manage Docker secrets
  service     Manage services
  stack       Manage Docker stacks
  swarm       Manage Swarm
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  ps          List containers
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  rmi         Remove one or more images
  run         Run a command in a new container
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  search      Search the Docker Hub for images
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  version     Show the Docker version information
  wait        Block until one or more containers stop, then print their exit codes

Run 'docker COMMAND --help' for more information on a command.
```

Chapter 2

Part 2

Run the Ubuntu image and access the bash shell

```
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
898c46f3b1a1: Pull complete
63366dfa0a50: Pull complete
041d4cd74a92: Pull complete
6e1bee0f8701: Pull complete
Digest: sha256:d019bdb3ad5af96fa1541f9465f070394c0daf0ffd692646983f491ce077b70f
Status: Downloaded newer image for ubuntu:latest
```

List Directories

```
root@6c00375706a7:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin
srv  sys  tmp  usr  var
```

Change directory to `/etc`

```
root@6c00375706a7:/# cd /etc
```

Type the contents of the `passwd` file

```
root@6c00375706a7:/etc# cat passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

```
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
```

Do you see a password for the `root` account?

No, there do not appear to be any passwords in the file.

All of the usernames are follwed by `x`; What does that mean?

`x` is a dummy password indicating the real password is stored in the `shadow` file.

Modify the password to: `abc123` using the command `passwd`

```
root@6c00375706a7:/etc# passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Type the content of the `shadow` file

```
root@6c00375706a7:/etc# cat shadow
root:$6$qxGUuh7L$GbLKmn9IgbuM/A.dvBpgOw3Qpfpyt.dz1jdouyz5L0/A979ss8wDPnal/WoecC
  Ub9x1GoAmRAyBkNewualCYr1:17969:0:99999:7:::
daemon:*:17962:0:99999:7:::
bin:*:17962:0:99999:7:::
sys:*:17962:0:99999:7:::
sync:*:17962:0:99999:7:::
games:*:17962:0:99999:7:::
man:*:17962:0:99999:7:::
lp:*:17962:0:99999:7:::
```

```
mail:*:17962:0:99999:7:::
news:*:17962:0:99999:7:::
uucp:*:17962:0:99999:7:::
proxy:*:17962:0:99999:7:::
www-data:*:17962:0:99999:7:::
backup:*:17962:0:99999:7:::
list:*:17962:0:99999:7:::
irc:*:17962:0:99999:7:::
gnats:*:17962:0:99999:7:::
nobody:*:17962:0:99999:7:::
_apt:*:17962:0:99999:7:::
root@6c00375706a7:/etc#
```

For the root account:

Search for the `$6$` shadow and determine the purpose if `$6$`. What type of hashing function has been used in generating the password hash value?

The purpose is to identify the hashing function used. `$6$` identifies a SHA-512 hash.

What is the salt value for the `root` account's password?

The salt used is `qxGUuh7L`.