Docker Foundation

An in depth introduction to Docker and Containers

Docker Images

1. Docker Images
2. Registries and Docker Hub
3. Dockerfiles 1
4. Dockerfiles 2

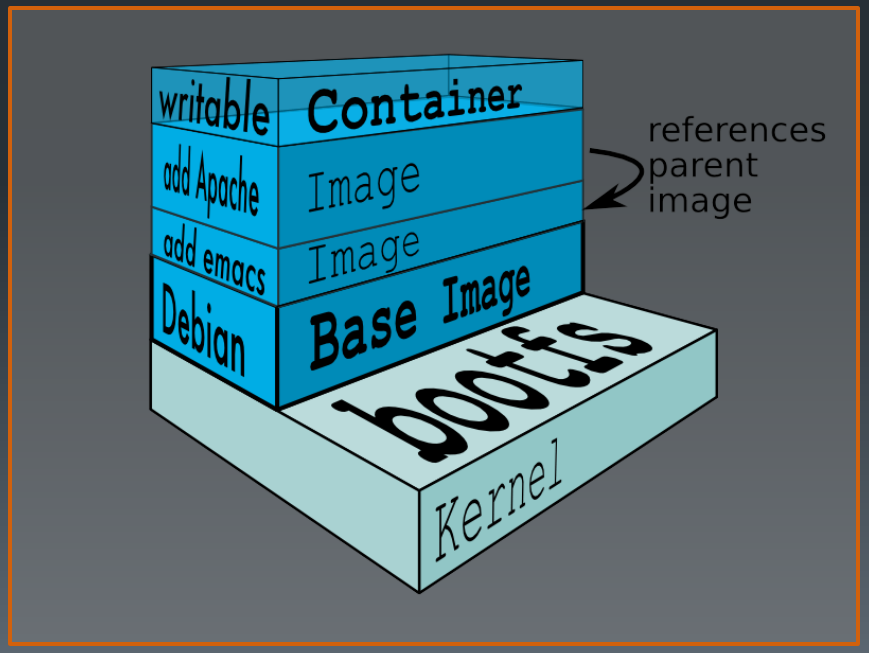
1: Docker Images

Objectives

- Describe Docker image layering
- Understand the nature of the Docker layered file system implementation
- Gain familiarity with the Docker directory structure
- Move images between registries and the local Docker Host
- Explore image creation

Docker Images

- Docker containers are constructed by sequentially applying meta data and file system layers from one or more images
- A Docker image includes metadata and an optional file system layer
  - Image meta data includes the ID of the image's parent, the default command to run, etc.
    - In Docker Engine <= v1.9 image IDs were random (UUIDs)
    - In Docker Engine >= v1.10 image IDs are SHA256 hashes of the image (content addressable)
  - Layered images tend to supply one specific feature on top of the parent image
  - Upper layer metadata and files mask metadata and files at lower layers with the same name/ pathname
- An image with no parent image is called a Base Image
  - Base images tend to be largish and house operating system libraries and executables (e.g. Debian, Ubuntu, Centos, etc.)
- Image metadata and files are immutable
  - Allows one Image to support multiple Container instances with repeatable results
  - Reduces the disk and memory footprint of a given set of containers which share the same read only images
- Containers have a writable file system layer
  - The container file system layer is initially empty
  - All writes go to this file system layer and overlay any matching underlying image files
  - In this way, container file systems contain only the delta between their file system state and that of their underlying images
  - The view from the top down, including all file system layers in the stack, is called the union file system



Displaying images

- The `docker image ls` command displays the images available on the docker engine
- Images that use the v2 format have a SHA 256 digest and ID, both of which identify the content
  - The ID is the hash of the image manifest and the uncompressed filesystem layers
  - The digest is the hash of the image manifest and the gzipped filesystem layers
    - Useful for download verification
- `Pull`, `create`, `run`, and `rm` can use the digest
  - `docker image pull hello-world@sha256:0256e8a36e2070f7bf2d0b0763dbabdd67798512411de4cdcf9431a1feb60fd9`
  - Images not pulled will not have a digest

```
user@ubuntu:~$ docker image ls --help

Usage:  docker image ls [OPTIONS] [REPOSITORY[:TAG]]

List images

Aliases:
  ls, images, list

Options:
  -a, --all             Show all images (default hides intermediate images)
      --digests         Show digests
  -f, --filter filter   Filter output based on conditions provided
      --format string   Pretty-print images using a Go template
      --help            Print usage
      --no-trunc        Don't truncate output
  -q, --quiet           Only show numeric IDs
```

```
user@ubuntu:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              latest              98d35105a391        6 days ago          193 MB
busybox             latest              00f017a8c2a6        12 days ago         1.11 MB
nginx               latest              6b914bbcb89e        3 weeks ago         182 MB
httpd               latest              f316d5949bb0        3 weeks ago         176 MB
ubuntu              latest              0ef2e08ed3fa        3 weeks ago         130 MB
ubuntu              12.04               b384dd9703db        3 weeks ago         104 MB
hello-world         latest              48b5124b2768        2 months ago        1.84 kB
cirros              latest              f8ce316a37a7        14 months ago       7.74 MB
```

```
user@ubuntu:~$ docker image ls --digests
REPOSITORY          TAG                 DIGEST                                                                    IMAGE ID            CREATED             SIZE
centos              latest              sha256:be5b4a93f116a57ab3fd454ada72421eac892a3a4925627ac9a44f65fcd69cf8   98d35105a391        6 days ago          193 MB
busybox             latest              sha256:32f093055929dbc23dec4d03e09dfe971f5973a9ca5cf059cbfb644c206aa83f   00f017a8c2a6        12 days ago         1.11 MB
nginx               latest              sha256:52a189e49c0c797cfc5cbfe578c68c225d160fb13a42954144b29af3fe4fe335   6b914bbcb89e        3 weeks ago         182 MB
httpd               latest              sha256:4612fba4347bd87eaeecd5c522d844f26cc4065b45eef9291277497946b7a86c   f316d5949bb0        3 weeks ago         176 MB
ubuntu              latest              sha256:dd7808d8792c9841d0b460122f1acf0a2dd1f56404f8d1e56298048885e45535   0ef2e08ed3fa        3 weeks ago         130 MB
ubuntu              12.04               sha256:47effe29d7bed549fee8c63bd60120fa88785e72abfb2aeeccdda21f4eae67e4   b384dd9703db        3 weeks ago         104 MB
hello-world         latest              sha256:c5515758d4c5e1e838e9cd307f6c6a0d620b5e07e6f927b07d05f6d12a1ac8d7   48b5124b2768        2 months ago        1.84 kB
cirros              latest              sha256:9aa75497b46cc15ccccceef625acee6017d7f3e78db9bd5f7b6b933feaa38e3ae   f8ce316a37a7        14 months ago       7.74 MB
```

https://docs.docker.com/engine/userguide/storagedriver/imagesandcontainers/#/content-addressable-storage

Image Filtering/Formatting

- The `docker image ls` command supports filtering and formatting much like `docker container ls`
- The currently supported filters (`-f`) are:
  - dangling (boolean - true or false)
  - label (label=<key> or label=<key>=<value>)
  - before (<image-name>[:<tag>], <image id> or <image@digest>) - filters images created *before* given id or references
  - since (<image-name>[:<tag>], <image id> or <image@digest>) - filters images created *since* given id or references
- The supported format lds (`--format`) are:
  - .ID Image ID
  - .Repository Image repository
  - .Tag Image tag
  - .Digest Image digest
  - .CreatedSince Elapsed time since the image was created
  - .CreatedAt Time when the image was created
  - .Size Image disk size

https://docs.docker.com/engine/reference/commandline/images/#/filtering

```
user@ubuntu:~$ docker image ls -a --format="{{.Repository}}\t{{.Tag}}/t{{.Size}}"
sam/thriftdev   0.9.1/t290 MB
fedora  25/t230 MB
fedora  latest/t230 MB
registry        latest/t33.2 MB
httpd   latest/t176 MB
ubuntu  16.04/t130 MB
ubuntu  latest/t130 MB
ubuntu  xenial/t130 MB
bobsmith/ubuntu prod_2017/t188 MB
ubuntu  14.04/t188 MB
ubuntu  trusty/t188 MB
bufferoverflow/thrift   latest/t161 MB
thriftdev       latest/t161 MB
hello-world     latest/t1.84 kB
fedora  24/t204 MB
cirros  latest/t7.74 MB
user@ubuntu:~$
```

```
user@ubuntu:~$ docker images -a -f dangling=true
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
<none>              <none>              e45556b4af4d        11 seconds ago      289.1 MB
```

Image Inspect

- The `docker image inspect` subcommand displays the meta data associated with an image
- All image configuration and/or descriptive data which is not part of the image's filesystem can be found in the metadata produced by inspect
- `docker inspect` displays metadata for:
  - Images
  - Containers
  - Networks
  - Volumes
  - Etc.

```
user@ubuntu:~$ docker image inspect alpine
[ {     "Id": "sha256:4e38e38c8ce0b8d9041a9c4fefe786631d1416225e13b0bfe8cfa2321aec4bba",
        "RepoTags": [ "alpine:latest“ ],
        "RepoDigests":
[ "alpine@sha256:3dcdb92d7432d56604d4545cbd324b14e647b313626d99b889d0626de158f73a" ],
        "Parent": "",
        "Comment": "",
        "Created": "2016-06-23T19:55:18.048565927Z",
        "Container": "4c573a7cf4a3e50eee83db0405523923591b9c3e94715e1f7f9b9fc8f39fbd4d",
        "ContainerConfig": {
            "Hostname": "4c573a7cf4a3",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [ "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin" ],
            "Cmd": [
                "/bin/sh",
                "-c",
                "#(nop) ADD file:852e9d0cb9d906535af512a89339fc70b2873a0f94defbcbe41cd44942dd6ac8
in /"
            ],
            "Image": "",
            "Volumes": null,
            "WorkingDir": "",
            "Entrypoint": null,
            "OnBuild": null,
            "Labels": null
        },
        "DockerVersion": "1.10.3",
        "Author": "",
        "Config": {
            "Hostname": "4c573a7cf4a3",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [ "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin" ],
            "Cmd": null,
            "Image": "",
            "Volumes": null,
            "WorkingDir": "",
            "Entrypoint": null,
            "OnBuild": null,
            "Labels": null
        },
        "Architecture": "amd64",
        "Os": "linux",
        "Size": 4798851,
        "VirtualSize": 4798851,
        "GraphDriver": { "Name": "aufs",  "Data": null },
        "RootFS": {
            "Type": "layers",
            "Layers": [ "sha256:4fe15f8d0ae69e169824f25f1d4da3015a48feeeeebb265cd2e328e15c6a869f" ]
        }
} ]
```

Top Level Metadata Keys

- Id – image ID [hash]
- RepoTags – locally defined repository:tag names for the image
- RepoDigests – repository hash (entire repo)
- Parent – parent image ID
- Comment – commit message
- Created – timestamp of the image's creation
- Container – container ID used to commit image
- ContainerConfig – metadata from container used to create this image
- DockerVersion – Docker version used to build the image
- Author – person or organization responsible for the image
- Config – metadata for containers generated from this image
- Architecture – image system architecture
- OS – image operating system
- Size – total size of the image including all layers it is composed of
- VirtualSize – same as size (this field is deprecated)
- GraphDriver – union filesystem driver used
- RootFS – list of layers used by the image

Particularly useful metadata

```
//IDENTITY
user@ubuntu:~$ docker image inspect -f '{{.Id}}' lab/websvr:v0.2
sha256:dce7086f0579209099b9e686eeeeaf52449ec82ddbeffee1b7fcc05071848d37

user@ubuntu:~$ docker image inspect -f '{{.RepoTags}}' lab/websvr:v0.2
[lab/websvr:v0.2 problems:bad-web-server websvr:latest]

user@ubuntu:~$ docker image inspect -f '{{.Parent}}' lab/websvr:v0.2
sha256:2392c04e694138e499490491efccae406a2fea5e487c47498ada2ef281d7f136

user@ubuntu:~$ docker image inspect -f '{{.Size}}' httpd
195428379

//CONFIG (applied to containers)
user@ubuntu:~$ docker image inspect -f '{{.Config.ExposedPorts}}' httpd
map[80/tcp:{}]

user@ubuntu:~$ docker image inspect -f '{{.Config.Env}}' httpd
[  PATH=/usr/local/apache2/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
   HTTPD_PREFIX=/usr/local/apache2 HTTPD_VERSION=2.4.23
   HTTPD_SHA1=5101be34ac4a509b245adb70a56690a84fcc4e7f
   HTTPD_BZ2_URL=https://www.apache.org/dist/httpd/httpd-2.4.23.tar.bz2 ]

user@ubuntu:~$ docker image inspect -f '{{.Config.Cmd}}' httpd
[httpd-foreground]

user@ubuntu:~$ docker image inspect -f '{{.Config.WorkingDir}}' httpd
/usr/local/apache2
```

Go Template Documentation
https://golang.org/pkg/text/template/

```
user@ubuntu:~$ docker inspect -f '{{json .Config.Labels}}' lab/websvr:v0.2 | jq .
{
  "build-date": "20160729",
  "license": "GPLv2",
  "name": "CentOS Base Image",
  "vendor": "CentOS"
}
```

```
user@ubuntu:~$ docker inspect -f '{{range $t := .RepoTags}}TAG: {{$t}}{{"\n"}}{{end}}Total: {{len .RepoTags}}' lab/websvr:v0.2
TAG: lab/websvr:v0.2
TAG: problems:bad-web-server
TAG: websvr:latest
Total: 3
```

The images directory (v1.10+)

- Image file system layers are stored in the Docker Engine's working directory
  - In our Ubuntu lab system: /var/lib/docker/aufs
- Image metadata map layer SHA hashes to cache-ids which point to the local layer cache
  - In our Ubuntu system: /var/lib/docker/aufs/diff
  - Diffs include the files and directories that make up the layer in question

```
root@ubuntu:/var/lib/docker# ls -la ./aufs/diff/
total 104
drwx------ 26 root root 4096 Apr 17 15:52 .
drwx------  5 root root 4096 Apr 17 02:27 ..
drwxr-xr-x  2 root root 4096 Apr 17 15:27 0192a8226619f414ad29bca135391d368640c3c79d8d91f53caf8317ee40617e
drwxr-xr-x  6 root root 4096 Apr 17 15:04 16a9f975046d04a7261dbe3b6a44e8b1d60afcd8815cf64f53e4e2c9c1294107
drwxr-xr-x  6 root root 4096 Apr 17 14:56 16b0db522083a9013e3f72ff4256ef696a695cf192c94bed55f7a2c3cb1d9561
drwxr-xr-x  2 root root 4096 Apr 17 15:27 29caced7ecd29d1c0bd80269beae38ddf64e77781acf2c4b401899a0f040c7b9
drwxr-xr-x  2 root root 4096 Apr 17 15:27 34a598d06c20e3968ea99074d2106744433f1fc89134af9eb7a4724b5961dd3b
drwxr-xr-x 17 root root 4096 Apr 17 15:01 34a767c44948c8bd2bec2fc1c257e2b90a55461b2c419fd563ded0e54a2abf45
drwxr-xr-x  2 root root 4096 Apr 17 15:27 37420a3a62ad9454c05be6e60c582f500f0e9dba701ba3de347a65c1647a7fff
drwxr-xr-x  2 root root 4096 Apr 17 02:34 3cad029bf4bf448abe7460f960c96b2bc823a31b0819089ddaaa829ac91c6cb9
drwxr-xr-x  2 root root 4096 Apr 17 15:27 5016f52a6c4a74064eb52adc80a23d506f0f3ed5012573861163769c628da106
drwxr-xr-x  6 root root 4096 Apr 17 15:27 502ee477d7a64462a96cc7c052025f2bf25008f9b84f0f1c19b5a91a9a970e04
drwxr-xr-x  6 root root 4096 Apr 17 15:27 502ee477d7a64462a96cc7c052025f2bf25008f9b84f0f1c19b5a91a9a970e04-init
drwxr-xr-x  3 root root 4096 Apr 17 14:56 53d212eae170f109ab0038216222fdc89202542a46d4d36644825960b4bfe5ad
drwxr-xr-x  3 root root 4096 Apr 17 15:27 6a0f24b14bd8d2c886a5e824cec501e92d75afbf9665975499da97c0d1515f74
drwxr-xr-x 22 root root 4096 Apr 17 15:04 8be336bfbf6c84f2568e7a9e0341f77dc52b1e2260bb92da8a9052b70ca9f061
drwxr-xr-x  3 root root 4096 Apr 17 15:04 8ffb33465a5773df1acfe7cd98ddf7c0633da1d0e742bb4c05059b216e488405
drwxr-xr-x  3 root root 4096 Apr 17 15:04 aba86a946d93420f89b2383c5ac88121e9de28f9a7a397f7b3c0baa83ea50cc7
drwxr-xr-x  7 root root 4096 Apr 17 15:27 b127fe13759f66f99bd599a16601944264bf69e5fd52ce6bd91ba2bccd87dbb7
drwxr-xr-x  2 root root 4096 Apr 17 02:34 b491680442ece62aef56d7fcb9e592ba1e2cd3d79acef4c7690c9802d726ce7a
drwxr-xr-x  2 root root 4096 Apr 17 14:56 c29e61c943e873bbc0ec56c6080c872ffa4de3605c2ba600586ed2fb548b9da0
drwxr-xr-x  2 root root 4096 Apr 17 15:01 c2e54d80147b513be287a2833f8209f5b63590de8f1a62b857c02e40078f0f24
drwxr-xr-x  2 root root 4096 Apr 17 15:00 de484a758e3115a2e3e25d43844ecf983d842b15f7ac9bff76d371a5c829037e
drwxr-xr-x  2 root root 4096 Apr 17 15:01 e78821d2b17560dc542b5fa5d94ceb08014f69297493cde87c0b412adecbf8a7
drwxr-xr-x 21 root root 4096 Apr 17 15:27 f48a3f03f58b053579ea8c08c8068fd7314ee562295ff5424c4e08fe344df734
drwxr-xr-x 21 root root 4096 Apr 17 14:56 fdeaa9243c4d15b74dae01d7f119210bb1e6a5c38a9ed5dc1eb78637c47d246d
root@ubuntu:/var/lib/docker# ls -l aufs/diff/502ee477d7a64462a96cc7c052025f2bf25008f9b84f0f1c19b5a91a9a970e04
total 8
drwxr-xr-x 2 root root 4096 Apr 17 15:27 run
drwxr-xr-x 3 root root 4096 Apr 17 15:27 var
root@ubuntu:/var/lib/docker#
```

```
user@ubuntu:~$ sudo su -
root@ubuntu:~# cd /var/lib/docker
root@ubuntu:/var/lib/docker# docker container ls
CONTAINER ID        IMAGE   COMMAND                  CREATED
9c0539b42b4c        nginx   "nginx -g 'daemon off"   About an hour ago
root@ubuntu:/var/lib/docker# ls -l image/aufs/layerdb/mounts/9c0539b42b4c
total 12
-rw-r--r-- 1 root root 69 Apr 17 15:27 init-id
-rw-r--r-- 1 root root 64 Apr 17 15:27 mount-id
-rw-r--r-- 1 root root 71 Apr 17 15:27 parent
root@ubuntu:/var/lib/docker# cat image/aufs/layerdb/mounts/9c0539b42b4cdd44a3d5b7ab3c969b218d4de165279e44015ecab6e0c7fdc24
502ee477d7a64462a96cc7c052025f2bf25008f9b84f0f1c19b5a91a9a970e04
root@ubuntu:/var/lib/docker# cat image/aufs/layerdb/mounts/9c0539b42b4cdd44a3d5b7ab3c969b218d4de165279e44015ecab6e0c7fdc24
502ee477d7a64462a96cc7c052025f2bf25008f9b84f0f1c19b5a91a9a970e04-init
root@ubuntu:/var/lib/docker# cat image/aufs/layerdb/mounts/9c0539b42b4cdd44a3d5b7ab3c969b218d4de165279e44015ecab6e0c7fdc24
sha256:400844ac32a0beb2184bb766eb6d922cf0e55a87e9b89017a7d7c8a0677c17bd
root@ubuntu:/var/lib/docker# ls -l image/aufs/layerdb/sha256/400844ac32a0beb2184bb766eb6d922cf0e55a87e9b89017a7d7c8a0677c1
total 20
-rw-r--r-- 1 root root 64 Apr 17 15:27 cache-id
-rw-r--r-- 1 root root 71 Apr 17 15:27 diff
-rw-r--r-- 1 root root 71 Apr 17 15:27 parent
-rw-r--r-- 1 root root  1 Apr 17 15:27 size
-rw-r--r-- 1 root root 82 Apr 17 15:27 tar-split.json.gz
root@ubuntu:/var/lib/docker# cat image/aufs/layerdb/sha256/400844ac32a0beb2184bb766eb6d922cf0e55a87e9b89017a7d7c8a0677c17b
37420a3a62ad9454c05be6e60c582f500f0e9dba701ba3de347a65c1647a7fff
root@ubuntu:/var/lib/docker# cat image/aufs/layerdb/sha256/400844ac32a0beb2184bb766eb6d922cf0e55a87e9b89017a7d7c8a0677c17b
sha256:312751f456c3a5927bb8762459eb1488f8d31dd9dc1299df82e7de0da5cfe48b
root@ubuntu:/var/lib/docker# ls -l image/aufs/layerdb/sha256/312751f456c3a5927bb8762459eb1488f8d31dd9dc1299df82e7de0da5cfe48b
total 20
-rw-r--r-- 1 root root 64 Apr 17 15:27 cache-id
-rw-r--r-- 1 root root 71 Apr 17 15:27 diff
-rw-r--r-- 1 root root 71 Apr 17 15:27 parent
-rw-r--r-- 1 root root  1 Apr 17 15:27 size
-rw-r--r-- 1 root root 82 Apr 17 15:27 tar-split.json.gz
```

- **Mount layer**: the writable container layer
- **Init layer**: container setup added by docker
- **Parent layer**: the image parent layer
  - **Cache-id**: Images reference their filesystem layers via cache-id

Container deltas

- /var/lib/docker/aufs/diff
  - Holds AUFS copy on write (COW) data
    - Files from any other layer which have been modified
    - e.g. /etc/hostname
  - Holds new files
    - Specific to the container
    - e.g. /myfile & myfile2

```
/ # ls -l
total 6280
drwxrwxr-x    2 root     root          4096 Sep  8 08:07 bin
drwxrwxr-x    3 root     root          4096 Sep  8 08:07 boot
drwxr-xr-x    5 root     root           380 Mar  2 11:54 dev
drwxrwxr-x   14 root     root          4096 Mar  2 11:54 etc
drwxrwxr-x    4 root     root          4096 Sep  8 09:24 home
-rwxr-xr-x    1 root     root          2616 Sep  8 09:24 init
drwxrwxr-x    3 root     root          4096 Sep  8 08:07 lib
lrwxrwxrwx    1 root     root            11 Sep  8 08:52 linuxrc -> bin/busybox
drwxrwxr-x    2 root     root          4096 Sep  8 08:37 media
drwxrwxr-x    2 root     root          4096 Sep  8 08:37 mnt
-rw-r--r--                           227128 Mar  2 11:57 myfile
-rw-r--r--                          6132456 Mar  2 12:00 myfile2
drwxrwxr-x                             4096 Sep  8 08:07 old-root
drwxrwxr-x    2 root     root          4096 Sep  8 08:37 opt
dr-xr-xr-x  336 root     root             0 Mar  2 11:54 proc
drwx------    2 root     root          4096 Mar  2 11:56 root
drwxr-xr-x    2 root     root          4096 Sep  8 09:24 run
drwxrwxr-x    2 root     root          4096 Sep  8 08:07 sbin
dr-xr-xr-x   13 root     root             0 Mar  2 11:54 sys
drwxrwxrwt    3 root     root          4096 Sep  8 09:24 tmp
drwxrwxr-x    6 root     root          4096 Sep  8 08:07 usr
drwxrwxr-x    8 root     root          4096 Sep  8 09:24 var
```

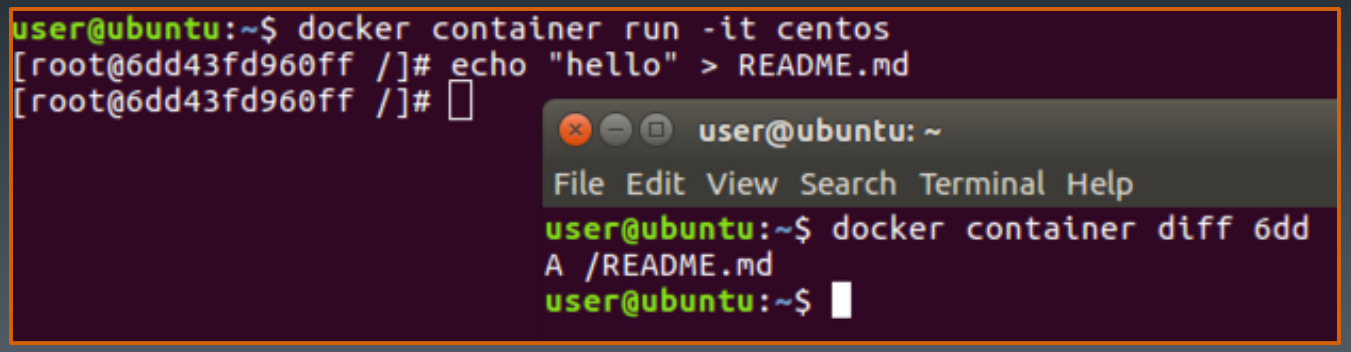
Container

```
docker@ubuntu:~$ sudo ls -l /var/lib/docker/aufs/diff/172a2193b4b4b87997441234cb09a3e2107d271c8afb544b1f1443fe5f7dbd98
total 6220
-rw-r--r-- 1 root root  227128 Mar  2 10:57 myfile
-rw-r--r-- 1 root root 6132456 Mar  2 11:00 myfile2
drwx------ 2 root root    4096 Mar  2 10:56 root
docker@ubuntu:~$ sudo ls -l /var/lib/docker/aufs/diff/172a2193b4b4b87997441234cb09a3e2107d271c8afb544b1f1443fe5f7dbd98-init
total 8
drwxr-xr-x 3 root root 4096 Mar  2 10:54 dev
drwxrwxr-x 2 root root 4096 Mar  2 10:54 etc
docker@ubuntu:~$ sudo ls -l /var/lib/docker/aufs/diff/172a2193b4b4b87997441234cb09a3e2107d271c8afb544b1f1443fe5f7dbd98-init/etc
total 0
-rwxr-xr-x 1 root root  0 Mar  2 10:54 hostname
-rwxr-xr-x 1 root root  0 Mar  2 10:54 hosts
lrwxrwxrwx 1 root root 12 Mar  2 10:54 mtab -> /proc/mounts
-rwxr-xr-x 1 root root  0 Mar  2 10:54 resolv.conf
docker@ubuntu:~$
```

Host

docker diff

- The `docker container diff` command can be used to view the changes in the container's filesystem layer
  - Works regardless of filesystem Unioning driver
- Three types of deltas are recorded
  - `A` - Add
  - `D` - Delete
  - `C` - Change

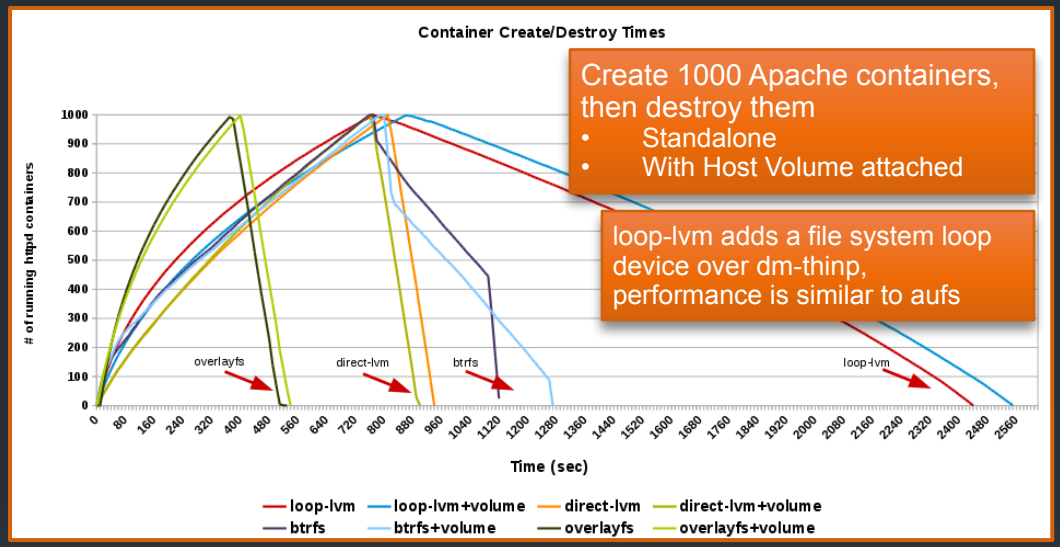


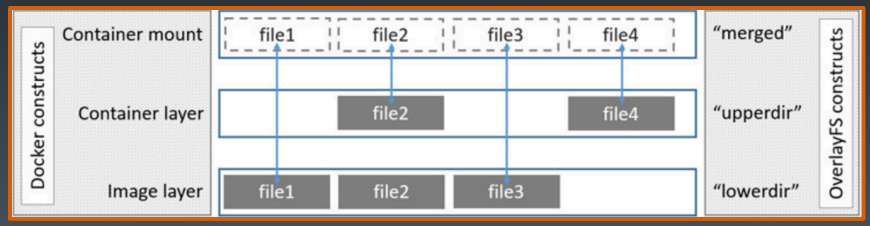
Filesystem Backends

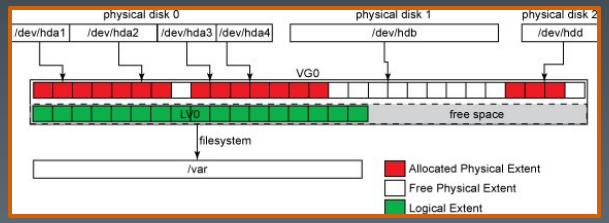
- Docker depends on the efficient use of layered images
  - Various file system features in the kernel implement file system layering
- Each container has two layers
  - The init layer, whose parent is the container's specified image (RO operational data)
    - Contains /etc/hosts, /etc/hostname and /etc/resolv.conf as well as the Docker configured /dev directory
  - A child of init which contains normal container content (RW application data)
- Committing a container creates an image from container rw layer w/ original image as parent
- Docker filesystem backends (known as Graph Drivers):
  - overlay - implements union mount for other FS
    - Introduced in kernel 3.18
    - Very fast, designed for SSDs and hybrids
    - overlay2 improves function, requires kernel 4.0
  - aufs – "Advanced multi layered Unification FileSystem" stores layers as directories with aufs metadata
    - FILE level scheme with shared storage
    - Default Ubuntu but not widely supported on RHEL
    - Originally limited to 42 layers, now 127 (the 127 limit applies to all drivers as of Docker 0.7.2)
  - btrfs – uses file system level snapshotting
    - BLOCK level scheme with shared storage
    - /var/lib/docker must be on a btrfs file system, layers stored as sub-volumes in /var/lib/docker/btrfs/subvolumes
    - Default on SUSE
  - devicemapper - uses device-mapper thin provisioning module (dm-thinp) to implement layers
    - Default on RHEL systems
    - BLOCK level scheme with shared storage
    - Device-mapper is the kernel part of the LVM2 system
  - vfs – universally supported but slow and inefficient
    - FILE level scheme with no shared storage
    - Each layer is a separate directory with a deep copy of its parent (no COW)
- Container Independent Volumes
  - Write-heavy I/O (databases, logs, etc.) may be done to a normal host volume eliminating unnecessary storage backend overhead
  - Host volumes can be accessed across multiple concurrent containers
    - Like a SAN without the network overhead

Docker 17.09 preferred graph driver order:

1. overlay2
2. overlay
3. aufs
4. btrfs
5. zfs
6. devicemapper
7. vfs







Driver Stability

- Many consider OverlayFS 1 & 2 the future of Docker storage
- Yet it is less mature and less stable than aufs and devicemapper
- Use OverlayFS with caution and expect to encounter more bugs and nuances than if you were using a more mature driver

AUFS: stable | production-ready | good memory use | smooth Docker experience | high write activity | PaaS-type work

Devicemapper (loop): stable | in mainline kernel | smooth Docker experience | production | performance | lab testing

Devicemapper (direct-lvm): stable | production-ready | in mainline kernel | smooth Docker experience | PaaS-type work

Btrfs: in mainline kernel | high write activity | container churn | build pools

Overlay: stable | good memory use | in mainline kernel | container churn | lab testing

ZFS native (ZoL): PaaS-type work

ZFS FUSE: stable | lab testing | production

Key

- Has attribute: attribute
- If good for use case: use case
- If bad for use case: use case

Storage driver	Commonly used on	Disabled on
`overlay`	`ext4` `xfs`	`btrfs` `aufs` `overlay` `overlay2` `zfs` `eCryptfs`
`overlay2`	`ext4` `xfs`	`btrfs` `aufs` `overlay` `overlay2` `zfs` `eCryptfs`
`aufs`	`ext4` `xfs`	`btrfs` `aufs` `eCryptfs`
`btrfs`	`btrfs` *only*	N/A
`devicemapper`	`direct-lvm`	N/A
`vfs`	debugging only	N/A
`zfs`	`zfs` *only*	N/A

Image Tags

- Images are given tag names for easy reference
- One image may have several tags
- One repository may contain several images
- You can specify a particular image within a repository using a trailing : and the tag name
  - e.g. “ubuntu:16.04”
  - If no image tag is supplied, Docker assumes the “latest” tag

```
user@ubuntu:~$ docker search ubuntu
NAME                                      DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
ubuntu                                    Ubuntu is a Debian-based Linux operating s...   5731      [OK]
rastasheep/ubuntu-sshd                    Dockerized SSH service, built on top of of...   77                   [OK]
ubuntu-upstart                            Upstart is an event-based replacement for ...   71        [OK]
consol/ubuntu-xfce-vnc                    Ubuntu container with "headless" VNC sessi...   45                   [OK]
ubuntu-debootstrap                        debootstrap --variant=minbase --components...   30        [OK]
torusware/speedus-ubuntu                  Always updated official Ubuntu docker imag...   27                   [OK]
nuagebec/ubuntu                           Simple always updated Ubuntu docker images...   17                   [OK]
nickistre/ubuntu-lamp                     LAMP server on Ubuntu                           16                   [OK]
nimmis/ubuntu                             This is a docker images different LTS vers...   6                    [OK]
darksheer/ubuntu                          Base Ubuntu Image -- Updated hourly             2                    [OK]
maxexcloo/ubuntu                          Base image built on Ubuntu with init, Supe...   2                    [OK]
jordi/ubuntu                              Ubuntu Base Image                               1                    [OK]
admiringworm/ubuntu                       Base ubuntu images based on the official u...   1                    [OK]
forumi0721ubuntuarmhf/ubuntu-armhf-
forumi0721ubuntux64/ubuntu-x64-dev
datenbetrieb/ubuntu
vcatechnology/ubuntu
teamrock/ubuntu
webhippie/ubuntu
forumi0721ubuntuaarch64/ubuntu-aarc
lynxtp/ubuntu
konstruktoid/ubuntu
forumi0721ubuntux64/ubuntu-x64-dev-
forumi0721ubuntux64/ubuntu-x64-dev-android   ubuntu-x64-dev-android                       0                    [OK]
labengine/ubuntu                          Images base ubuntu                              0                    [OK]
user@ubuntu:~$ docker image pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
d54efb8db41d: Pull complete
f8b845f45a87: Pull complete
e8db7bf7c39f: Pull complete
9654c40e9079: Pull complete
6d9ef359eaaa: Pull complete
Digest: sha256:dd7808d8792c9841d0b460122f1acf0a2dd1f56404f8d1e56298048885e45535
Status: Downloaded newer image for ubuntu:latest
user@ubuntu:~$ docker image pull ubuntu:16.04
16.04: Pulling from library/ubuntu
Digest: sha256:dd7808d8792c9841d0b460122f1acf0a2dd1f56404f8d1e56298048885e45535
Status: Downloaded newer image for ubuntu:16.04
user@ubuntu:~$ docker image pull ubuntu:xenial
xenial: Pulling from library/ubuntu
Digest: sha256:dd7808d8792c9841d0b460122f1acf0a2dd1f56404f8d1e56298048885e45535
Status: Downloaded newer image for ubuntu:xenial
user@ubuntu:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
httpd               latest              f316d5949bb0        3 weeks ago         176 MB
ubuntu              16.04               0ef2e08ed3fa        3 weeks ago         130 MB
ubuntu              latest              0ef2e08ed3fa        3 weeks ago         130 MB
ubuntu              xenial              0ef2e08ed3fa        3 weeks ago         130 MB
ubuntu              14.04               7c09e61e9035        3 weeks ago         188 MB
ubuntu              trusty              7c09e61e9035        3 weeks ago         188 MB
hello-world         latest              48b5124b2768        2 months ago        1.84 kB
cirros              latest              f8ce316a37a7        14 months ago       7.74 MB
```

```
user@ubuntu:~$ docker container run -it ubuntu:xenial /bin/bash
root@199063abdc5f:/# grep VERSION /etc/os-release
VERSION="16.04.2 LTS (Xenial Xerus)"
VERSION_ID="16.04"
VERSION_CODENAME=xenial
root@199063abdc5f:/# exit
exit
```

Displaying Repositories

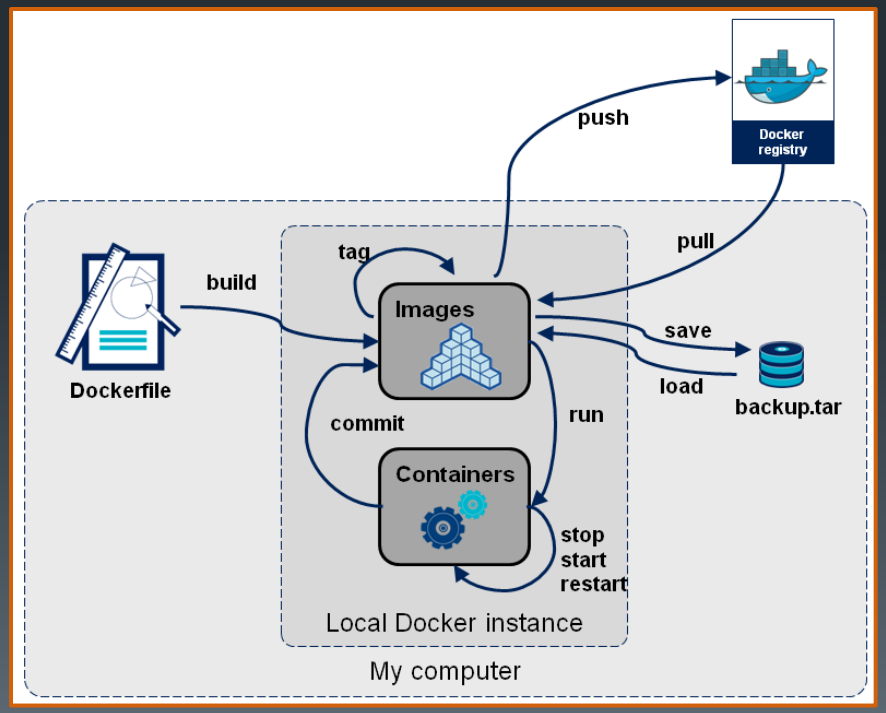
- You can scope the `image ls` subcommand to examine a particular repository
- You can also specify a particular tagged image to pull
  - Required when you want to pull something other than the latest tag
  - Default is “latest”
  - Can be used to pull a newer version of the image from a registry

```
user@ubuntu:~$ docker image ls cirros
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
cirros              latest              f8ce316a37a7        14 months ago       7.74 MB
user@ubuntu:~$ docker image ls ubuntu
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              16.04               0ef2e08ed3fa        3 weeks ago         130 MB
ubuntu              latest              0ef2e08ed3fa        3 weeks ago         130 MB
ubuntu              xenial              0ef2e08ed3fa        3 weeks ago         130 MB
ubuntu              14.04               7c09e61e9035        3 weeks ago         188 MB
ubuntu              trusty              7c09e61e9035        3 weeks ago         188 MB
user@ubuntu:~$
```

```
user@ubuntu:~$ docker image pull fedora:25
25: Pulling from library/fedora
ffa7676a36a8: Pull complete
Digest: sha256:be1e975ca0832971c21e6876eeea4d2a117ee21d0603fb9eb61ab8ce4ac0e34e
Status: Downloaded newer image for fedora:25
user@ubuntu:~$ docker image ls fedora
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
fedora              25                  0047cca29c6f        7 days ago          230 MB
user@ubuntu:~$ docker image pull fedora
Using default tag: latest
latest: Pulling from library/fedora
Digest: sha256:be1e975ca0832971c21e6876eeea4d2a117ee21d0603fb9eb61ab8ce4ac0e34e
Status: Downloaded newer image for fedora:latest
user@ubuntu:~$ docker image pull fedora:24
24: Pulling from library/fedora
2bf01635e2a0: Pull complete
Digest: sha256:7ffee69dbfaea363cba55db07942a5c0d29f291eaf510edd6c07f139001f4174
Status: Downloaded newer image for fedora:24
user@ubuntu:~$ docker image ls fedora
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
fedora              25                  0047cca29c6f        7 days ago          230 MB
fedora              latest              0047cca29c6f        7 days ago          230 MB
fedora              24                  f623aaef07f0        4 months ago        204 MB
user@ubuntu:~$
```

Creating Images

- Docker images are the basis of containers
- Docker stores downloaded images on the Docker host
  - If a required image isn't already present on the host it is downloaded from a registry
- There are two ways to create new images
  - Update a container created from an image and commit the results to a new image
  - Use a Dockerfile to specify instructions to create an image



Committing a Container

- Committing an existing Container is a simple way to create a new Docker image
  1. Create a container with the desired base
  2. Make the changes necessary to the container layer
  3. Commit the container layer as a new image

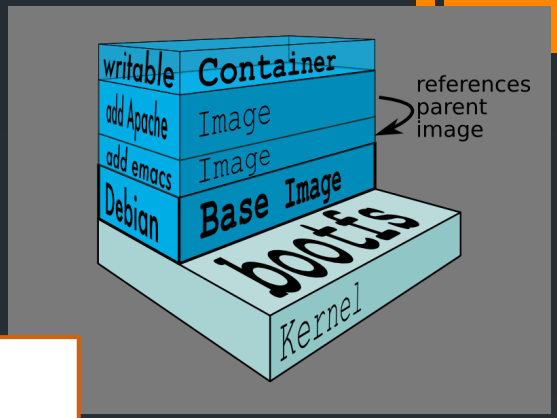
1.
```
docker@ubuntu:~$ docker container run -t -i ubuntu:14.04 /bin/bash
root@3802a4a2a2e9:/# apt-get install git
...
root@3802a4a2a2e9:/# git clone http://github.com/apache/thrift
…
```

2.
```
root@3802a4a2a2e9:/# cd thrift
root@3802a4a2a2e9:/thrift# git checkout 0.9.1
...
root@3802a4a2a2e9:/thrift# exit
exit
```

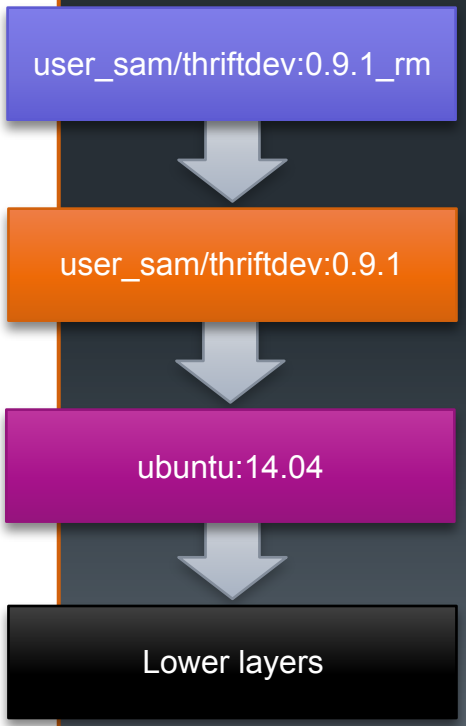
3.
```
docker@ubuntu:~$ docker container commit -m "Thrift dev" -a "Sam Thrift" 3802a4a2a2e9 sam/thriftdev:
0.9.1
ca0310547ecb9e71370479f94b5977a358054210d74c06bc0f869e32d51e96ae
docker@ubuntu:~$ docker image ls
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
sam/thriftdev           0.9.1               ca0310547ecb        8 seconds ago       247.7 MB
ubuntu                  14.04.2             2d24f826cb16        13 days ago         188.3 MB
ubuntu                  14.04               2d24f826cb16        13 days ago         188.3 MB
ubuntu                  latest              2d24f826cb16        13 days ago         188.3 MB
ubuntu                  trusty              2d24f826cb16        13 days ago         188.3 MB
ubuntu                  trusty-20150218.1   2d24f826cb16        13 days ago         188.3 MB
ubuntu                  12.04               1f80e9ca2ac3        13 days ago         131.5 MB
ubuntu                  12.04.5             1f80e9ca2ac3        13 days ago         131.5 MB
ubuntu                  precise             1f80e9ca2ac3        13 days ago         131.5 MB
ubuntu                  precise-20150212    1f80e9ca2ac3        13 days ago         131.5 MB
centos                  latest              dade6cb4530a        4 weeks ago         210.1 MB
cirros                  latest              8d202478b999        4 weeks ago         7.698 MB
fedora                  20                  6cece30db4f9        9 weeks ago         360.3 MB
fedora                  21                  834629358fe2        9 weeks ago         241.3 MB
fedora                  latest              834629358fe2        9 weeks ago         241.3 MB
bufferoverflow/thrift   latest              5eb5528b2680        4 months ago        4.287 GB
```

Examining Ancestry

- Each Docker container has a parent image
- Each Docker image has a parent image, with the exception of base images
- Intermediate layer images have no repository name and no tag name
  - Intermediate images can be used to collect common files beneath multiple named images
  - Intermediate images are not designed to be directly used by containers (but can be)




```
docker@ubuntu:~$ docker container run -t -i sam/thriftdev:0.9.1 /bin/bash
root@8d75247087b5:/# cd thrift
root@8d75247087b5:/thrift# vi README
root@8d75247087b5:/thrift# exit
exit
docker@ubuntu:~$ docker container commit -m "README update" -a "Sam Thrift" 8d75247087b5 sam/thriftdev:
0.9.1_rm
fc056a4320ce0c439e240b7d56f991cdbd76a4fbc732c180bea4201a7d283cf4
docker@ubuntu:~$ docker image ls
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
sam/thriftdev           0.9.1_rm            fc056a4320ce        4 seconds ago       247.7 MB
sam/thriftdev           0.9.1               ca0310547ecb        11 minutes ago      247.7 MB
ubuntu                  14.04.2             2d24f826cb16        13 days ago         188.3 MB
ubuntu                  latest              2d24f826cb16        13 days ago         188.3 MB
ubuntu                  trusty              2d24f826cb16        13 days ago         188.3 MB
ubuntu                  trusty-20150218.1   2d24f826cb16        13 days ago         188.3 MB
ubuntu                  14.04               2d24f826cb16        13 days ago         188.3 MB
ubuntu                  precise             1f80e9ca2ac3        13 days ago         131.5 MB
ubuntu                  precise-20150212    1f80e9ca2ac3        13 days ago         131.5 MB
ubuntu                  12.04.5             1f80e9ca2ac3        13 days ago         131.5 MB
ubuntu                  12.04               1f80e9ca2ac3        13 days ago         131.5 MB
centos                  latest              dade6cb4530a        4 weeks ago         210.1 MB
cirros                  latest              8d202478b999        4 weeks ago         7.698 MB
fedora                  20                  6cece30db4f9        9 weeks ago         360.3 MB
fedora                  21                  834629358fe2        9 weeks ago         241.3 MB
fedora                  latest              834629358fe2        9 weeks ago         241.3 MB
cufferoverflow/thrift   latest              5eb5528b2680        4 months ago        4.287 GB
docker@ubuntu:~$ docker image inspect -f='{{.Parent}}' sam/thriftdev:0.9.1_rm
ca0310547ecb9e71370479f94b5977a358054210d74c06bc0f869e32d51e96ae
docker@ubuntu:~$ docker image inspect -f='{{.Parent}}' sam/thriftdev:0.9.1
2d24f826cb16146e2016ff349a8a33ed5830f3b938d45c0f82943f4ab8c097e7
docker@ubuntu:~$ docker image inspect -f='{{.Parent}}' ubuntu:14.04

docker@ubuntu:~$
```



Intermediate Images

- By default the `docker image ls` command does not display intermediate images
  - Intermediate images have no tag
  - Intermediate images act as building blocks for higher layer images
- `$ docker image ls -a`
  - To display intermediate images
- `docker history` displays an image's parents
- The `--no-trunc` switch can be used to display full IDs with most docker commands

```
docker@ubuntu:~$ docker images -a
REPOSITORY              TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
user_sam/thriftdev      0.9.1_rm            fc056a4320ce        About an hour ago   247.7 MB
user_sam/thriftdev      0.9.1               ca0310547ecb        About an hour ago   247.7 MB
ubuntu                  trusty              2d24f826cb16        13 days ago         188.3 MB
ubuntu                  14.04               2d24f826cb16        13 days ago         188.3 MB
ubuntu                  14.04.2             2d24f826cb16        13 days ago         188.3 MB
ubuntu                  latest              2d24f826cb16        13 days ago         188.3 MB
ubuntu                  trusty-20150218.1   2d24f826cb16        13 days ago         188.3 MB
<none>                  <none>              117ee323aaa9        13 days ago         188.3 MB
<none>                  <none>              1c8294cc5160        13 days ago         188.3 MB
<none>                  <none>              fa4fd76b09ce        13 days ago         188.1 MB
ubuntu                  12.04.5             1f80e9ca2ac3        13 days ago         131.5 MB
ubuntu                  precise             1f80e9ca2ac3        13 days ago         131.5 MB
ubuntu                  precise-20150212    1f80e9ca2ac3        13 days ago         131.5 MB
ubuntu                  12.04               1f80e9ca2ac3        13 days ago         131.5 MB
<none>                  <none>              e829ddd93a57        13 days ago         131.5 MB
<none>                  <none>              7bee65a22cab        13 days ago         131.5 MB
<none>                  <none>              edeb8497f5fc        13 days ago         131.4 MB
centos                  latest              dade6cb4530a        4 weeks ago         210.1 MB
cirros                  latest              8d202478b999        4 weeks ago         7.698 MB
```

```
user@ubuntu:~$ docker image history sam/thriftdev:0.9.1
IMAGE               CREATED             CREATED BY                                      SIZE                COMMENT
007537baec97        40 seconds ago      /bin/bash                                       160 MB              Thrift dev
0ef2e08ed3fa        3 weeks ago         /bin/sh -c #(nop)  CMD ["/bin/bash"]            0 B
<missing>           3 weeks ago         /bin/sh -c mkdir -p /run/systemd && echo '...   7 B
<missing>           3 weeks ago         /bin/sh -c sed -i 's/^#\s*\(deb.*universe\...   1.9 kB
<missing>           3 weeks ago         /bin/sh -c rm -rf /var/lib/apt/lists/*          0 B
<missing>           3 weeks ago         /bin/sh -c set -xe   && echo '#!/bin/sh' >...   745 B
<missing>           3 weeks ago         /bin/sh -c #(nop) ADD file:efb254bc677d66d...   130 MB
user@ubuntu:~$
```

```
<none>                  <none>              b480f1dd38f7        4 months ago        4.021 GB
<none>                  <none>              75bdecba3b21        4 months ago        3.989 GB
<none>                  <none>              dec7815b184e        4 months ago        3.706 GB
<none>                  <none>              7e5c48811f37        4 months ago        3.085 GB
<none>                  <none>              a3959cbd00b2        4 months ago        3.049 GB
<none>                  <none>              c328ff699bc7        4 months ago        3.027 GB
<none>                  <none>              807c7b12fffb        4 months ago        2.356 GB
<none>                  <none>              c35efe5fe594        4 months ago        2.354 GB
<none>                  <none>              5efec72f6d22        4 months ago        2.346 GB
<none>                  <none>              2de42e5ac940        4 months ago        2.333 GB
<none>                  <none>              6399e5518efe        4 months ago        2.323 GB
```

```
user@ubuntu:~$ docker image ls -a --no-trunc
REPOSITORY          TAG                 IMAGE ID                                                                  CREATED             SIZE
fedora              25                  sha256:0047cca29c6f4288eabb00b77b46ec6e2024c8cf95dcc2d1cf4efdb910d98fc3   7 days ago          230 MB
fedora              latest              sha256:0047cca29c6f4288eabb00b77b46ec6e2024c8cf95dcc2d1cf4efdb910d98fc3   7 days ago          230 MB
registry            latest              sha256:047218491f8cef3987c8a248b8a4ebc46025b04d7b700f711c320a290959a8a6   2 weeks ago         33.2 MB
httpd               latest              sha256:f316d5949bb02561a68216997d2d7a3b80d1f621729d50e1f14a4172af6ad464   3 weeks ago         176 MB
ubuntu              16.04               sha256:0ef2e08ed3fabfc44002ccb846c4f2416a2135affc3ce39538834059606f32dd   3 weeks ago         130 MB
ubuntu              latest              sha256:0ef2e08ed3fabfc44002ccb846c4f2416a2135affc3ce39538834059606f32dd   3 weeks ago         130 MB
ubuntu              xenial              sha256:0ef2e08ed3fabfc44002ccb846c4f2416a2135affc3ce39538834059606f32dd   3 weeks ago         130 MB
ubuntu              14.04               sha256:7c09e61e90350e8f5c0cba2979003bdfe32c2d027b68b4f0cf9063cdd7b4bafd   3 weeks ago         188 MB
ubuntu              trusty              sha256:7c09e61e90350e8f5c0cba2979003bdfe32c2d027b68b4f0cf9063cdd7b4bafd   3 weeks ago         188 MB
hello-world         latest              sha256:48b5124b2768d2b917edcb640435044a97967015485e812545546cbed5cf0233   2 months ago        1.84 kB
fedora              24                  sha256:f623aaef07f0542f72d0a09a8817b1489622f6dc202dc6c5d3e846ba25d455d2   4 months ago        204 MB
cirros              latest              sha256:f8ce316a37a79a89cb6f9ef65f23b38893fe291b519a8b18cf08af65872f02a9   14 months ago       7.74 MB
user@ubuntu:~$
```

Removing Images

- Images can be deleted using the **`docker image rm`** command
  - The image must not be in use by any running or stopped containers
- Images can be deleted by either
  - repository:tag
    - This deletes the tag and then only deletes the image if there are no other references
  - ID
    - You can only delete images by ID is they have 0 or 1 tags
- Images are not removed until all of the tagged names referencing them are removed
- Intermediate parents (parent images with no tag) are removed by default when no dependencies remain upon them
- Forcing the delete of an image will cause future runs of dependent containers to fail
  - Docker can not pull missing dependencies because it does not know which registry to pull from

```
docker@ubuntu:~$ docker image ls
REPOSITORY              TAG                 IMAGE ID            CREATED             VIRTUAL
SIZE
sam/thriftdev           0.9.1_rm            fc056a4320ce        46 hours ago        247.7 MB
sam/thriftdev           0.9.1               ca0310547ecb        47 hours ago        247.7 MB
ubuntu                  trusty              2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                  14.04.2             2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                  latest              2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                  14.04               2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                  trusty-20150218.1   2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                  precise             1f80e9ca2ac3        2 weeks ago         131.5 MB
ubuntu                  precise-20150212    1f80e9ca2ac3        2 weeks ago         131.5 MB
ubuntu                  12.04               1f80e9ca2ac3        2 weeks ago         131.5 MB
ubuntu                  12.04.5             1f80e9ca2ac3        2 weeks ago         131.5 MB
docker@ubuntu:~$ docker image rm --help
Usage:        docker image rm [OPTIONS] IMAGE [IMAGE...]

Remove one or more images

Aliases:
  rm, rmi, remove

Options:
  -f, --force      Force removal of the image
      --help       Print usage
      --no-prune   Do not delete untagged parents
docker@ubuntu:~$ docker image rm sam/thriftdev:0.9.1_rm
Untagged: sam/thriftdev:0.9.1_rm
Deleted: fc056a4320ce0c439e240b7d56f991cdbd76a4fbc732c180bea4201a7d283cf4
docker@ubuntu:~$ docker image rm ca0310547ecb
Error response from daemon: Conflict, cannot delete ca0310547ecb because the container
8d75247087b5 is using it, use -f to force
FATA[0000] Error: failed to remove one or more images
docker@ubuntu:~$ docker container stop 8d75247087b5
8d75247087b5
docker@ubuntu:~$ docker image rm ca0310547ecb
Error response from daemon: Conflict, cannot delete ca0310547ecb because the container
8d75247087b5 is using it, use -f to force
FATA[0000] Error: failed to remove one or more images
docker@ubuntu:~$ docker container rm 8d75247087b5
8d75247087b5
docker@ubuntu:~$ docker image rm ca0310547ecb
Untagged: sam/thriftdev:0.9.1
Deleted: ca0310547ecb9e71370479f94b5977a358054210d74c06bc0f869e32d51e96ae
```

Summary

- Docker Registries a central location for storage and retrieval of Docker images
- Docker Repositories named collections of related Docker images
- Docker Images have:
  - Metadata
    - ID
    - Parent ID
    - Environment variables
    - etc.
  - Optionally any of the following:
    - One or more Repository:Tag names
    - A filesystem layer
- Containers are created from images
- Containers can be committed to create new images
  - Docker files can automate image builds
- Docker images are read-only
  - Changes in a container occur in the container layer overlaying image layers
  - Image file system layers are unioned to create a single file system view
- The Docker union file system is implemented by a backend driver
  - Various block and file based solutions are available: AUFS, BTRFS, Devicemapper, OverlayFS, etc.

Lab 1

- Creating Images

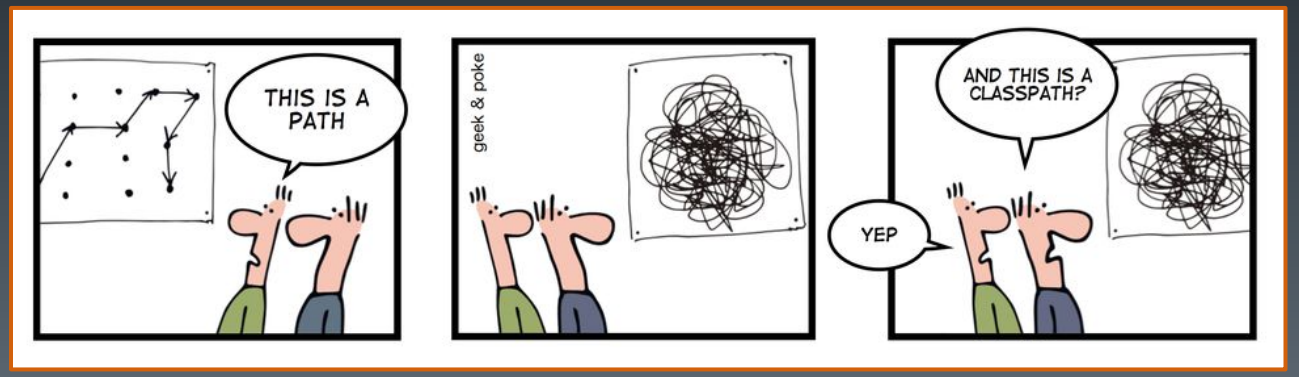
2: Docker Registries

Objectives

- Explain the relationship between Registries, Repositories, Tags, and Images
- Describe the Docker mechanism for locating Images via Registries
- Contrast Docker Hub with private Registries
- Examine the Registry login process
- List the steps used to run a private Registry
- Explore the explicit and implicit Docker Registry commands
- Explain the docker tag command and the workings of repository and tag names

Empowering Agile Processes

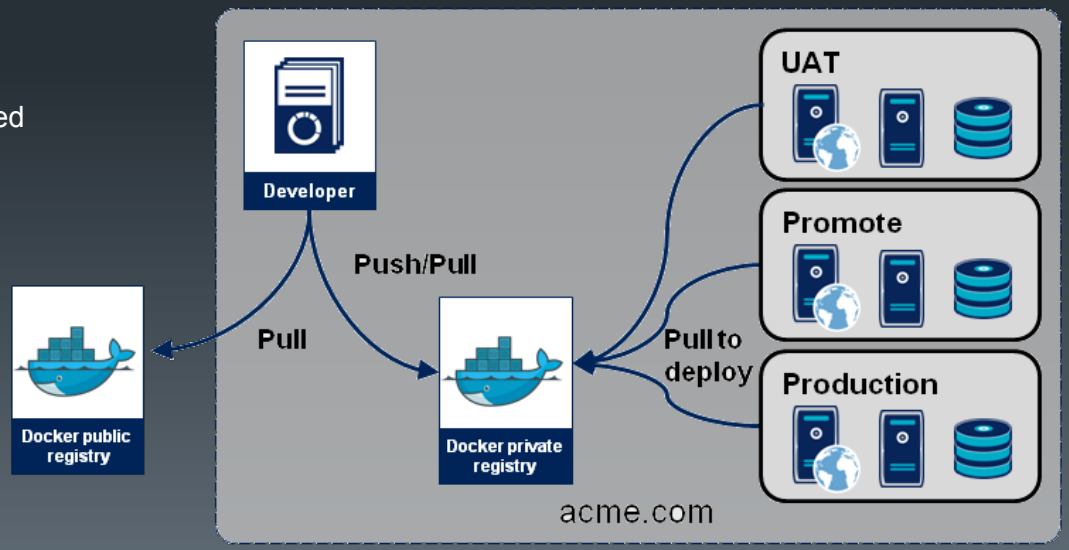
- SOA and Microservices
  - An approach to software development where many small services are composed into applications
    - Services communicate over well defined interfaces
    - Encourages small teams, each owning a service
  - The more atomic the service:
    1. The more likely it is to be reusable
    2. The more easily it can be encapsulated
    3. The more of them you need to do something useful (!)
- Micro services and VMs have different cardinalities
  - 10:1 Ten services running on a single VM creates an ops integration challenge across all services
- Micro services and Containers have the same cardinality
  - 1:1 One service in one container, requires only ops support for the service encapsulated
- Containers allow each service to be packaged with its own dependencies
  - 10 containerized services map directly to 10 individual, isolated, reliable ops events
  - Gives devops teams autonomy
    - Team owns software and configuration when using containers
  - Empowers CI/CD and incremental application evolution
    - Integration challenges are reduced to inter service communications, networking and volumes



Registries, Repos, and Tags

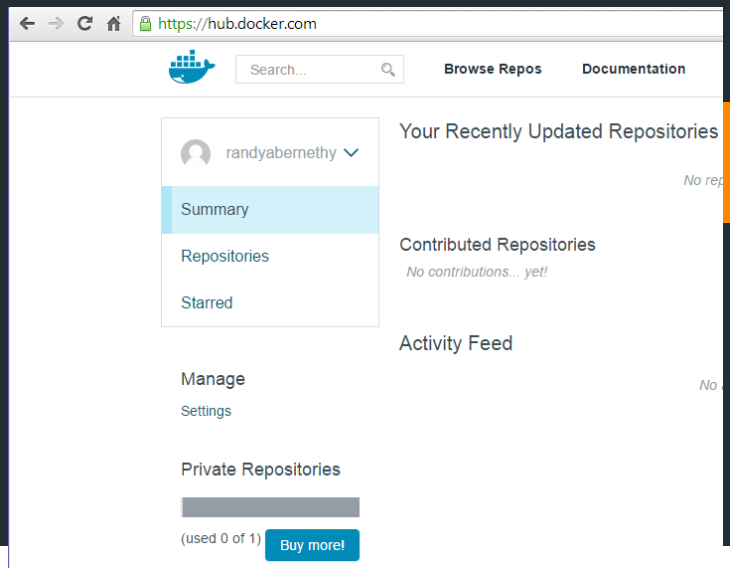
- Images house file system layers and metadata
- Repositories are named collections of images
- Tags are strings used to identify individual images within a repository
- Registries are services that allow you to store and retrieve images by repository:tag name using a REST (HTTP) API
  - Docker Hub is the central public registry
  - Companies can deploy their own registries
    - Using open source and commercial solutions
      - Docker Trusted Registry (formerly known as: Docker Hub Enterprise) from Docker Inc. (commercial product)
      - Quay Enterprise part of the Tectonic platform from CoreOS (commercial product)
      - Docker Registry from Docker Inc. (free open source) (https://github.com/docker/docker-registry)
      - Harbor open source enterprise-class container registry (https://github.com/vmware/harbor)
      - Artifactory 3.4+ (http://www.jfrog.com/open-source/#os-arti)
      - Nexus 3 Milestone 5 (in preview) (https://issues.sonatype.org/browse/NEXUS-6166)
    - Using hosted cloud solutions
      - Docker Hub (hub.docker.com) from Docker Inc.
      - Quay (quay.io), acquired by CoreOS Inc.
      - Google Container Registry (gcr.io) from Google
      - Amazon EC2 Container Registry [ECR] integrated with AWS EC2 Container Service [ECS]

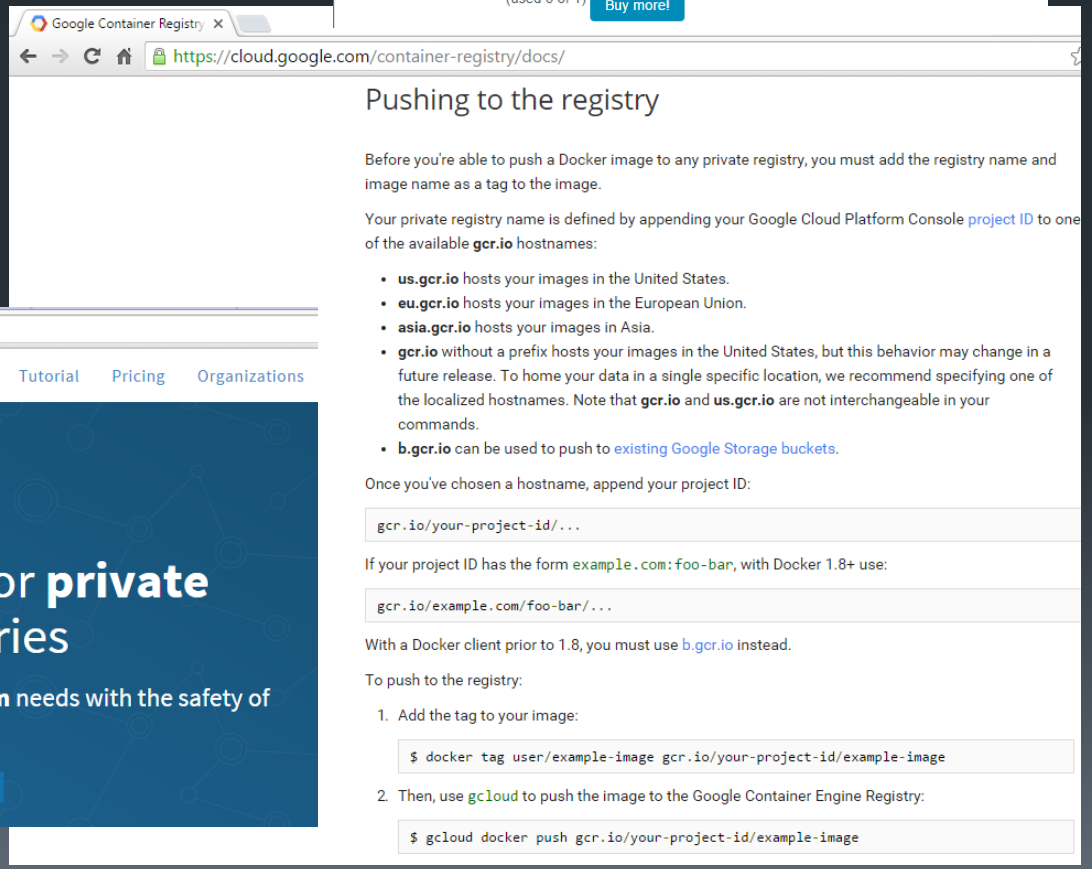
Public registry images should be used with caution by the security minded



Cloud Registries

- Docker Registries house collections of repositories
- Registries allow users to upload and download images from repositories
- Registries can be private and internal
  - Organizations can run one or more of their own internal registries
- Registries can be public
  - Docker Hub is a public registry run by Docker Inc.
    - Docker Hub is integrated into the Docker ecosystem and is the default registry
  - Other organizations create their own public registries
- Registries can be private and external
  - Companies can offer cloud based registry solutions commercially
  - Docker Hub hosts private registries
  - Quay hosts private registries for clients
    - Acquired by CoreOS Inc. 2014-08
  - Google hosts a registry service at gcr.io





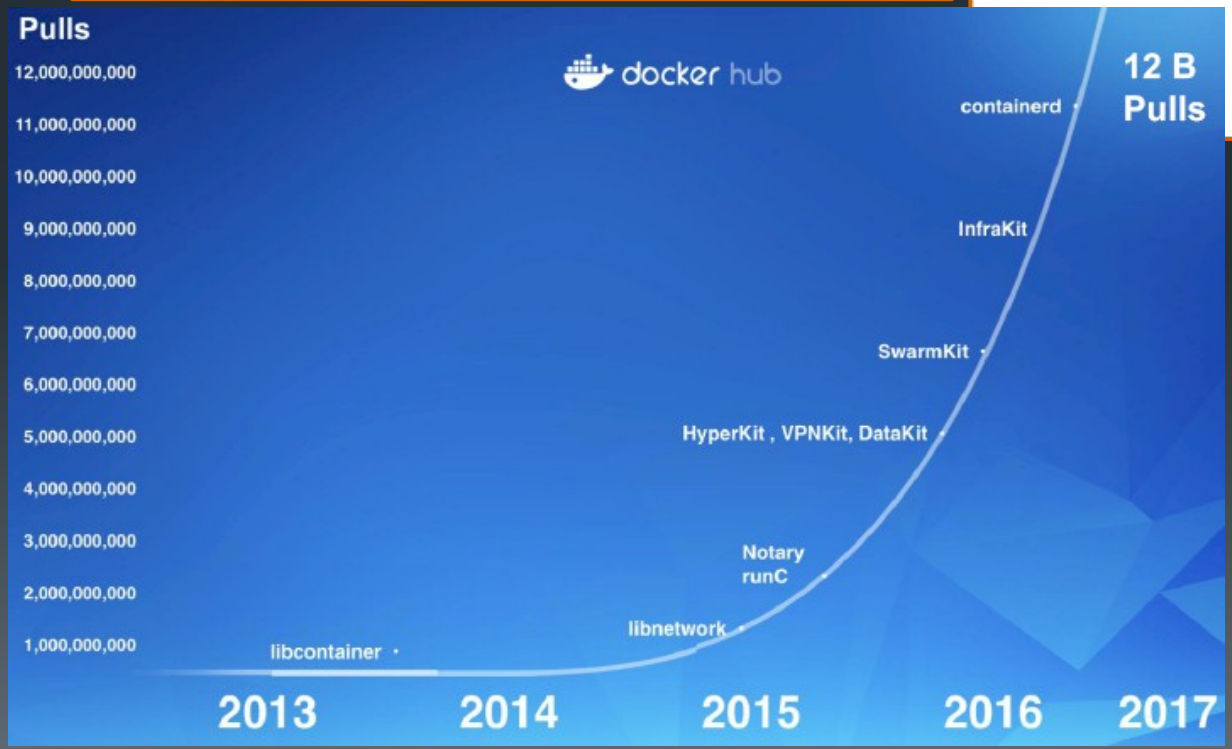
Docker Hub Pricing

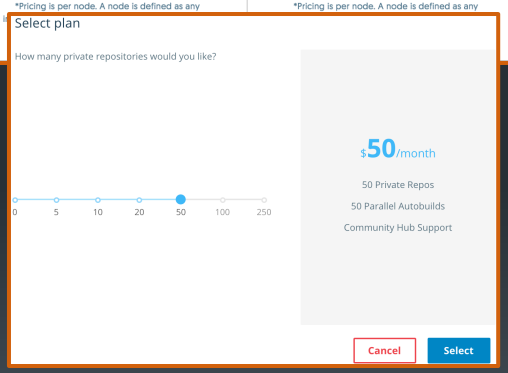
DOCKER CLOUD PRICING

FREE	PRIVATE REPOSITORIES **BUILD & SHIP**	MANAGED NODES **RUN**
This trial lets you take the Docker Cloud for spin	Add additional repositories for your teams.	Add additional nodes to scale and manage your application.
	Starts at $7 monthly for 5 repos.	**$15 monthly per node.**
Try it for Free	Purchase	Purchase
What's included: √ One free private repository √ One free managed node √ Unlimited public repositories	What's included: √ One Free + **Starts at $7 / month for 5 repos** √ Unlimited public repositories √ Automated builds √ Docker security scanning √ Automated tests	What's included: √ One Free + **$15 / month per node** √ Deploy unlimited containers √ Manage unlimited stacks and services √ One click upgrades √ Logging, scaling, notifications & more...
*Forum support	*Email support	*Email support

DOCKER DATACENTER PRICING

FREE 30-DAY TRIAL	BUSINESS DAY	BUSINESS CRITICAL
This trial lets you take the Docker Containers-as-a-Service [CaaS] Platform for a spin	Build, ship and run your applications behind the firewall, with business day support from Docker.	Build, ship and run your applications behind the firewall, with business critical 24 x 7 support from Docker.
	Starting at $150 monthly per node.	**Starting at $300 monthly per node.**
30 Day Free Trial	Purchase Contact Sales	Purchase Contact Sales
Docker Universal Control Plane Docker Trusted Registry Supported Docker Engine	Docker Universal Control Plane Docker Trusted Registry Supported Docker Engine Mon-Fri 9am to 6pm Local Time	Docker Universal Control Plane Docker Trusted Registry Supported Docker Engine 24/7/365
	*Docker Engines can be deployed anywhere *Pricing is per node. A node is defined as any	*Docker Engines can be deployed anywhere *Pricing is per node. A node is defined as any





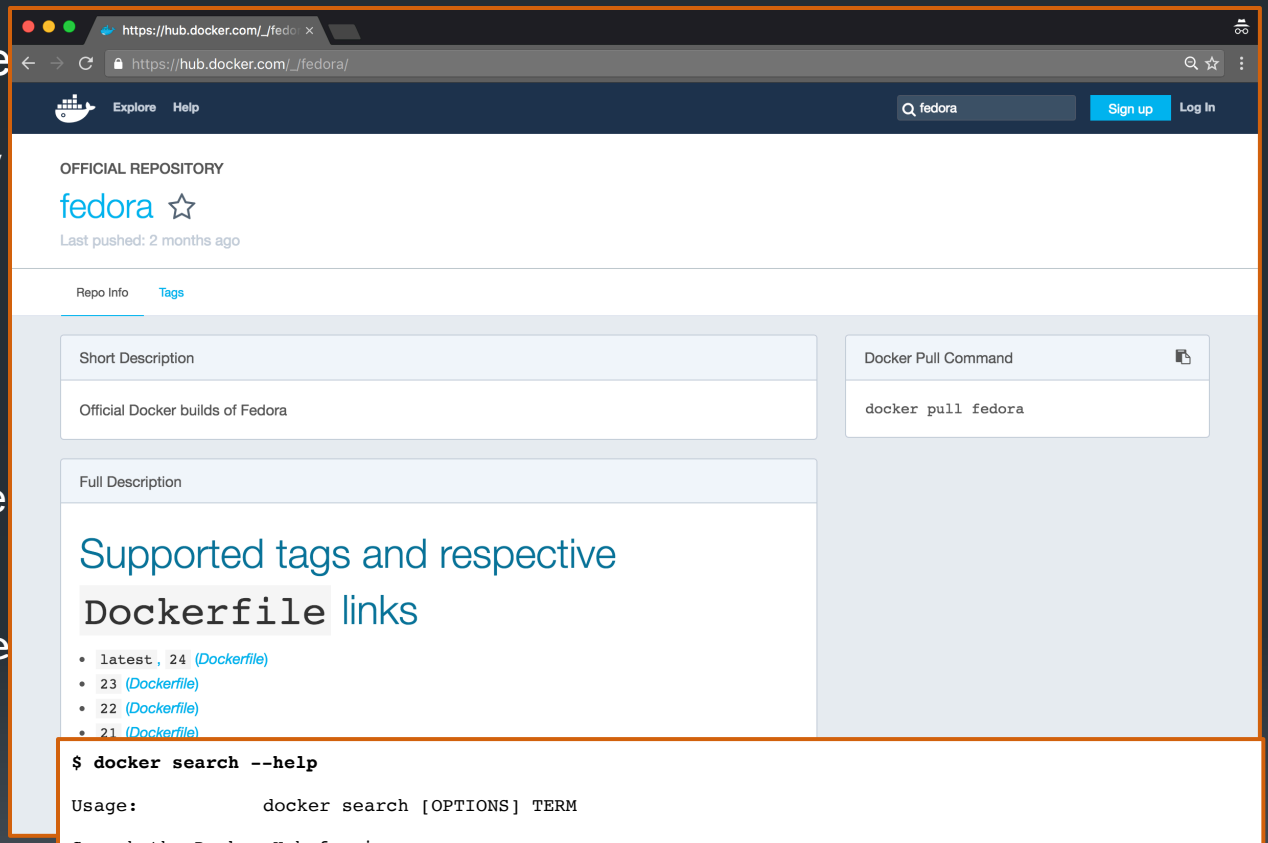
Quotas

To request an increase in any of these quotas, please contact **support@docker.com**.

Maximum number of containers	300
Maximum number of node clusters	30
Maximum number of nodes	60
Maximum number of services	150
Maximum number of stacks	40

Web Search

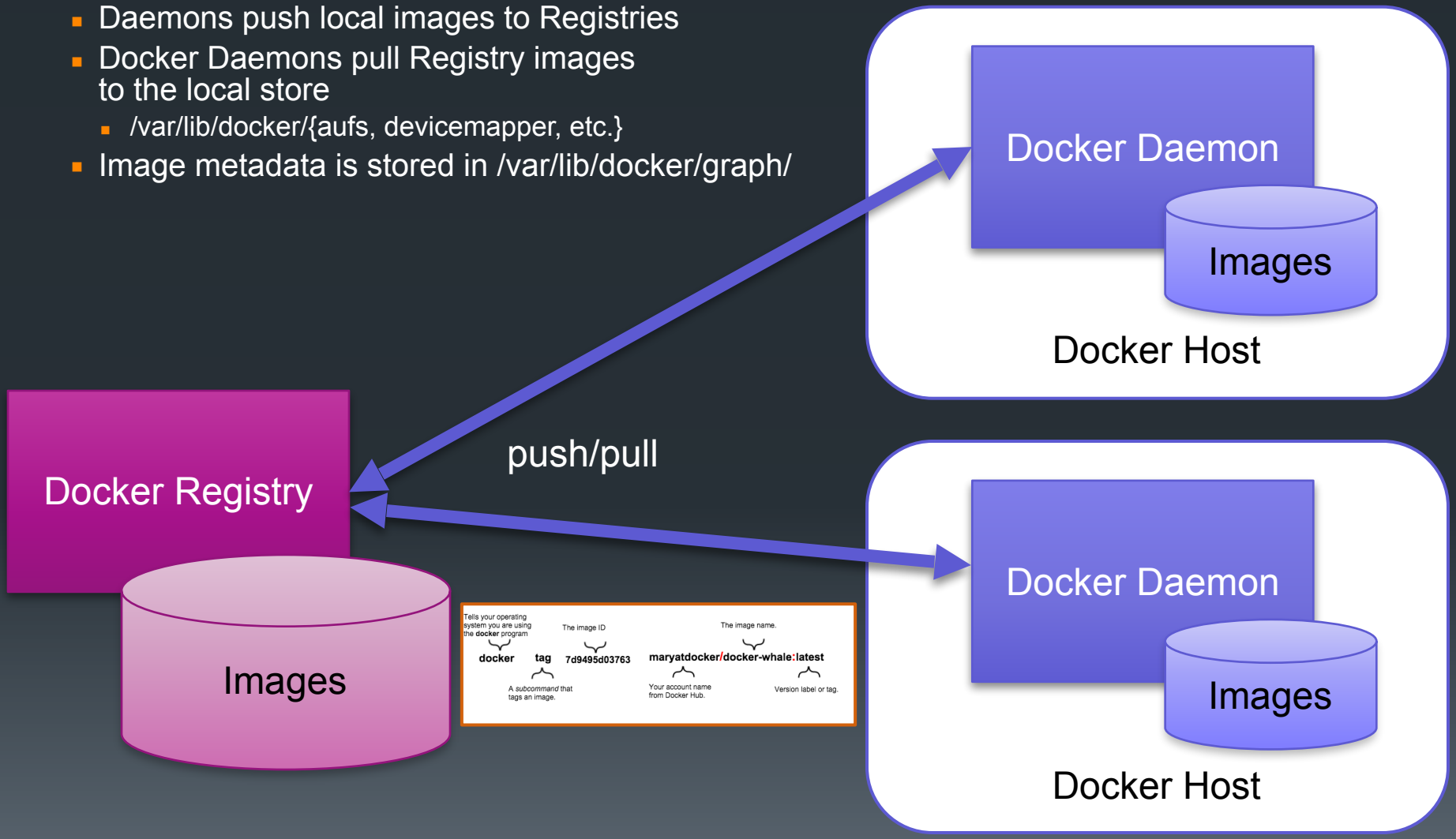
- Searching for repositories at the command line may suffice for casual lookups
  - However only basic repository information is returned
- The web interface to docker hub provides detailed information on each repository
  - Including individual tags, long descriptions, dockerfile source and more
- `docker search --filter "…" imageName`
  - stars - number of stars the image has
  - is-automated - is the image automated or not
  - is-official - is the image official or not




```
$ docker search --help

Usage:          docker search [OPTIONS] TERM

Search the Docker Hub for images

Options:
  -f, --filter value   Filter output based on conditions provided (default [])
      --help           Print usage
      --limit int      Max number of search results (default 25)
      --no-trunc       Don't truncate output

$ docker search --filter "is-automated=true" --filter "stars=3" busybox
NAME                DESCRIPTION                                     STARS OFFICIAL AUTOMATED
progrium/busybox                                                    50                   [OK]
radial/busyboxplus Full-chain, Internet enabled, busybox made... 8                    [OK]
```

push and pull

- Docker daemons maintain a local image store
  - Daemons push local images to Registries
  - Docker Daemons pull Registry images to the local store
    - /var/lib/docker/{aufs, devicemapper, etc.}
  - Image metadata is stored in /var/lib/docker/graph/



Registry Commands

- Docker Registry Commands
  - `docker login`
    - to login to a registry
  - `docker logout`
    - To logout of a registry
  - `docker search`
    - searches a registry for an image
  - `docker image pull`
    - pulls an image from registry to the local machine (also pulls its dependencies)
  - `docker image push`
    - pushes an image to the registry from local machine
- When running a container whose image cannot be found locally, the Docker Engine will try to pull the image from its registry automatically
- These commands (and the run command indirectly) cause the Docker daemon to perform network I/O with a registry independent of the docker client session

```
user@ubuntu:~$ docker search thrift
NAME                                DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
thrift                              Thrift is a framework for generating clien...   60        [OK]
evarga/thrift                       This is a Docker image for Apache Thrift. ...   4                    [OK]
whiteworld/thrift                                                                   1                    [OK]
apache/thrift                       Apache Thrift                                   1                    [OK]
randyabernethy/thrift-book          Companion container image for the Programm...   1                    [OK]
itzg/thrift                         Provides the Apache Thrift generator tool       0                    [OK]
bufferoverflow/thrift               Apache Thrift - *make cross*                    0                    [OK]
thrift/thrift-compiler-test         thrift-compiler test run to verify https:/...   0                    [OK]
cjmay/thrift                        thrift builds                                   0                    [OK]
nsuke/thrift                        Temporary docker images for Apache Thrift CI    0                    [OK]
lulichn/thrift                      Thrift                                          0                    [OK]
jimdo/thrift                        Mirror of Apache Thrift                         0                    [OK]
thrift/thrift-compiler              Apache Thrift Compiler                          0                    [OK]
iwan0/hbase-thrift-standalone       hbase-thrift-standalone                         0                    [OK]
thrift/ubuntu                                                                       0                    [OK]
cjmay/thrift-4042                   MWE of THRIFT-4042 (egg extraction error)       0                    [OK]
thrift/debian                       build/docker/debian                             0                    [OK]
zezuladp/thrift                                                                     0                    [OK]
flynnhandley/thrift-python          Installs Apache thrift to /usr/src/thrift/...   0                    [OK]
thrift/thrift-build                 Apache Thrift build environment.                0                    [OK]
rzhilkibaev/spark-standalone-thrift Standalone Spark with Thrift                    0                    [OK]
andyfactual/jenkins-dind-thrift     Jenkins server with thrift compiler installed   0                    [OK]
reecerobinson/spark-thrift-server   Docker container based on spark-nosql that...   0                    [OK]
zhoumingjun/thrift                                                                  0                    [OK]
vanoise29/thrift                    Thrift App                                      0
user@ubuntu:~$ docker image pull bufferoverflow/thrift
Using default tag: latest
latest: Pulling from bufferoverflow/thrift
75a822cd7888: Pull complete
d1bdc1496f66: Pull complete
ae2eddf44d88: Pull complete
Digest: sha256:d16ff438f1c14b75acfe15bf58075611ef2a30c1dac41e0e0184c08fb94e3684
Status: Downloaded newer image for bufferoverflow/thrift:latest
user@ubuntu:~$ docker image ls
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
sam/thriftdev           0.9.1               007537baec97        19 minutes ago      290 MB
user_sam/thriftdev      0.9.1               cf9f0a8e8ee4        21 minutes ago      290 MB
fedora                  25                  0047cca29c6f        7 days ago          230 MB
fedora                  latest              0047cca29c6f        7 days ago          230 MB
registry                latest              047218491f8c        2 weeks ago         33.2 MB
httpd                   latest              f316d5949bb0        3 weeks ago         176 MB
ubuntu                  16.04               0ef2e08ed3fa        3 weeks ago         130 MB
ubuntu                  latest              0ef2e08ed3fa        3 weeks ago         130 MB
ubuntu                  xenial              0ef2e08ed3fa        3 weeks ago         130 MB
ubuntu                  14.04               7c09e61e9035        3 weeks ago         188 MB
ubuntu                  trusty              7c09e61e9035        3 weeks ago         188 MB
bufferoverflow/thrift   latest              83a4c617e24e        2 months ago        161 MB
hello-world             latest              48b5124b2768        2 months ago        1.84 kB
fedora                  24                  f623aaef07f0        4 months ago        204 MB
cirros                  latest              f8ce316a37a7        14 months ago       7.74 MB
user@ubuntu:~$
```

Accessing Docker Hub

- Docker Hub can be searched and pulled anonymously
- The Docker command line tool provides a `login` command
  - Enables push access to a Docker Hub account from the CLI
  - Stores credentials in:
    - `~/.dockercfg` < docker 1.8
    - `~/.docker/config.json` >= docker 1.8
  - User, password and email can be supplied on the command line
    - `-e`, `--email="bob@example.com"`
    - `-p`, `--password="secret"`
    - `-u`, `--username="Bob Smith"`
- Users can login to private registries from the command line also
  - `$ docker login localhost:8080`

```
docker@ubuntu:~$ docker
login
Username: bobsmith
Password:
Email: bob@example.com
Login Succeeded
docker@ubuntu:~$
```

.docker/config.json

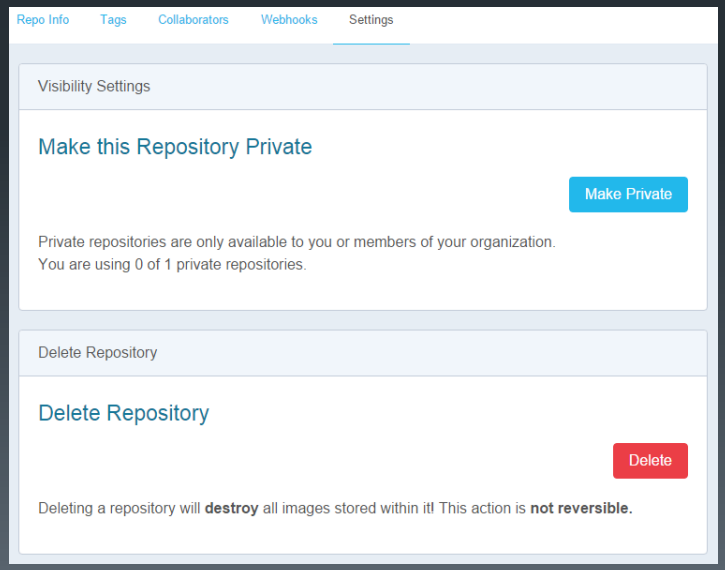
- .docker/config.json and/or .dockercfg entries provide a registry URL and an auth token
- The auth token is a base64 encoded string
  - base64(<username>:<password>)
  - This token is usable as long as the account and password on the target host are not changed
  - This representation is not secure and must be protected through file permissions (600)
- The `docker login` command simply populates this file
  - This can be automated with a script and/or predefined values can be copied

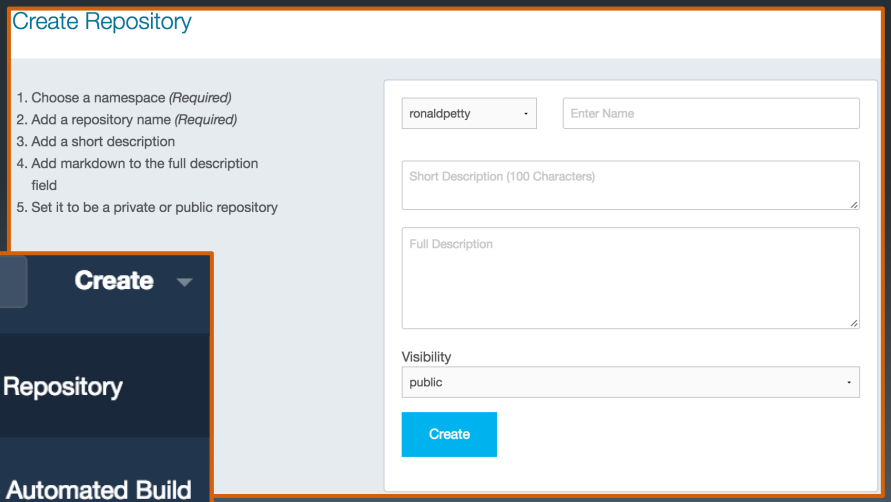
```
{
  "https://index.docker.io/v1/": {
    "auth": "assdfljhdsadjhhrjer=",
    "email": "bob@gmail.com"
  },
  "https://index.example.com": {
    "auth": "woeirdjfsudhfvjksds=",
    "email": "bob.smith@example.com"
  }
}
```

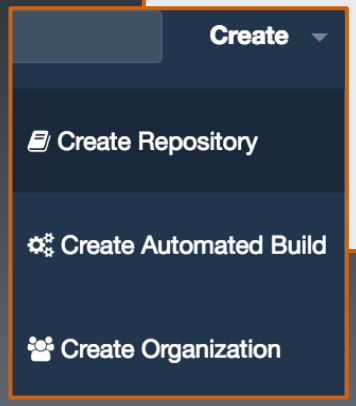
```
docker@ubuntu:~$ grep auth .docker/config.json | sed 's/.*"auth": "\(.*\)",\?/\1/'
XJfdXNlX3lvdXJlX2RvZ3NfbmFtZV9hc19hX3Bhc3N3b3JkCg=

docker@ubuntu:~$ grep auth .docker/config.json | sed 's/.*"auth": "\(.*\)",\?/\1/' | base64 --
decode
bobsmith:never_use_your_dogs_name_as_a_password
docker@ubuntu:~$
```

Docker Hub – Private Repos

- Private Docker Hub repositories must be added via the Add Repository link on the Docker Hub account page
- Once the private repository is created, you can push and pull images to and from it using Docker as always
  - You must be signed in and have access to work with a private repository
- It is not possible to browse or search private repositories on Docker Hub
- Private repositories do not get cached
  - Public repositories do
- It is possible to give access to a private repository to those whom you designate (i.e., collaborators) from its Settings page
- You can also switch repository status (public to private, or vice-versa)

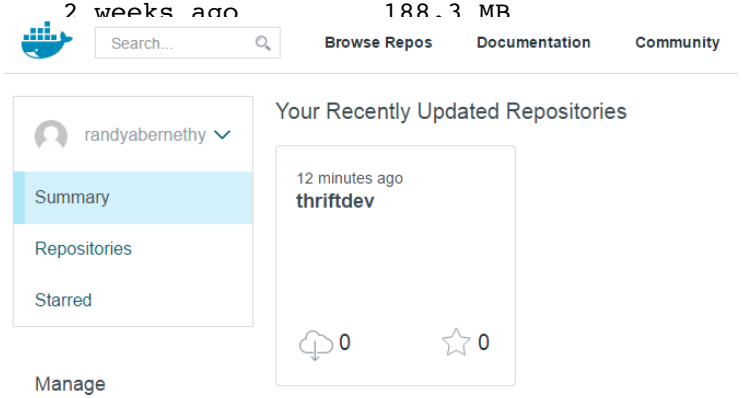






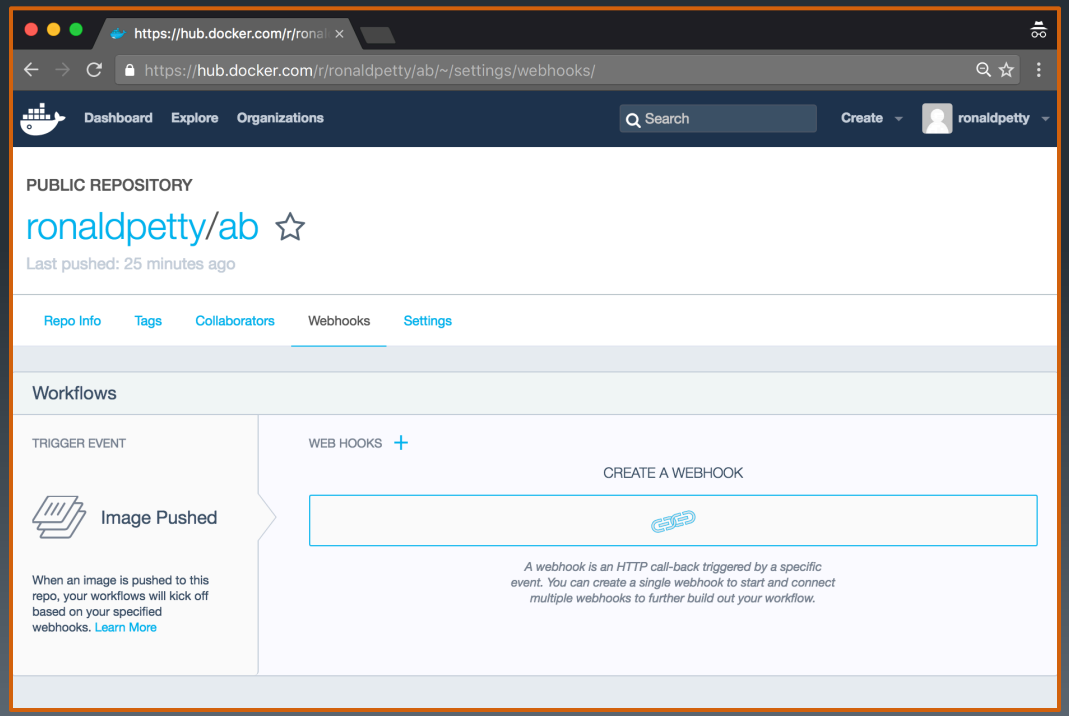
Pushing Repositories

```
docker@ubuntu:~$ docker image push --help

Usage: docker image push [OPTIONS] NAME[:TAG]

Push an image or a repository to the registry

  --disable-content-trust   Skip image verification (default true)
  --help=false       Print usage

docker@ubuntu:~$ docker image push thriftdev
FATA[0000] You cannot push a "root" repository. Please rename your repository to <user>/<repo> (ex: bobsmith/
thriftdev)

docker@ubuntu:~$ docker container commit ff808d882430 bobsmith/thriftdev
85905f19abdd41913aa16b9353b606e5dbbd79287a136cae5484df2ceac4dab0

docker@ubuntu:~$ docker image ls
REPOSITORY                TAG                 IMAGE ID            CREATED             SIZE
bobsmith/thriftdev        latest              85905f19abdd        7 minutes ago       226 MB
thriftdev                 latest              a69676e13708        9 minutes ago       226 MB
ubuntu                    trusty              2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                    trusty-20150218.1   2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                    14.04.2             2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                    latest              2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                    14.04               2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                    precise-20150212    1f80e9ca2ac3
ubuntu                    12.04               1f80e9ca2ac3
ubuntu                    12.04.5             1f80e9ca2ac3
ubuntu                    precise             1f80e9ca2ac3

docker@ubuntu:~$ docker image push bobsmith/thriftdev
The push refers to a repository [bobsmith/thriftdev] (len: 1)
Sending image list
Pushing repository bobsmith/thriftdev (1 tags)
511136ea3c5a: Image already pushed, skipping
fa4fd76b09ce: Image already pushed, skipping
1c8294cc5160: Image already pushed, skipping
117ee323aaa9: Image already pushed, skipping
2d24f826cb16: Image already pushed, skipping
85905f19abdd: Image successfully pushed
Pushing tag for rev [85905f19abdd] on {https://cdn-registry-1.docker.io/v2/repositories/bobsmith/thriftdev/
tags/latest}
```



Webhooks

- Webhooks allow you to trigger actions following a successful image push to a Docker Hub Repo
- Webhooks are configured on the Repository Settings page
- A Webhook makes an HTTP POST requests with a JSON payload describing the imaged pushed
  - You can restrict your service to accept requests from only the Docker Hub server IP Addresses: 162.242.195.64 - 162.242.195.127
- When multiple Webhooks are configured they are called in order
- Each Web hook can also have multiple URLs (aka. A webhook chain)
  - Each service in a chain must make a callback POST to the Docker Hub with a JSON body
    - The body must include a state with a value of success, failure or error
    - If the state isn't success, the webhook chain is terminated, no further URLs are called



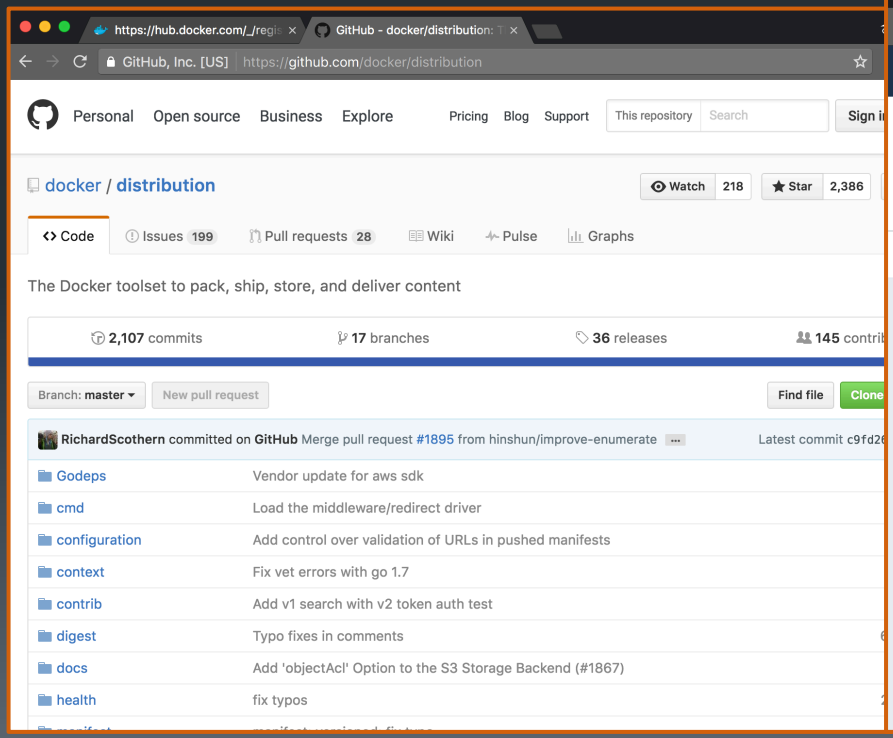

```
{
  "callback_url": "https://registry.hub.docker.co
  "push_data": {
    "images": [
        "27d47432a69bca5f2700e4dff7de0388ed65f9d3
        "51a9c7c1f8bb2fa19bcd09789a34e63f35abb800
        ...
    ],
    "pushed_at": 1.417566822e+09,
    "pusher": "svendowideit"
  },
  "repository": {
    "comment_count": 0,
    "date_created": 1.417566665e+09,
    "description": "",
    "full_description": "webhook triggered from a
    "is_official": false,
    "is_private": false,
    "is_trusted": false,
    "name": "busybox",
    "namespace": "svendowideit",
    "owner": "svendowideit",
    "repo_name": "svendowideit/busybox",
    "repo_url": "https://registry.hub.docker.com/
    "star_count": 0,
    "status": "Active"
}
```

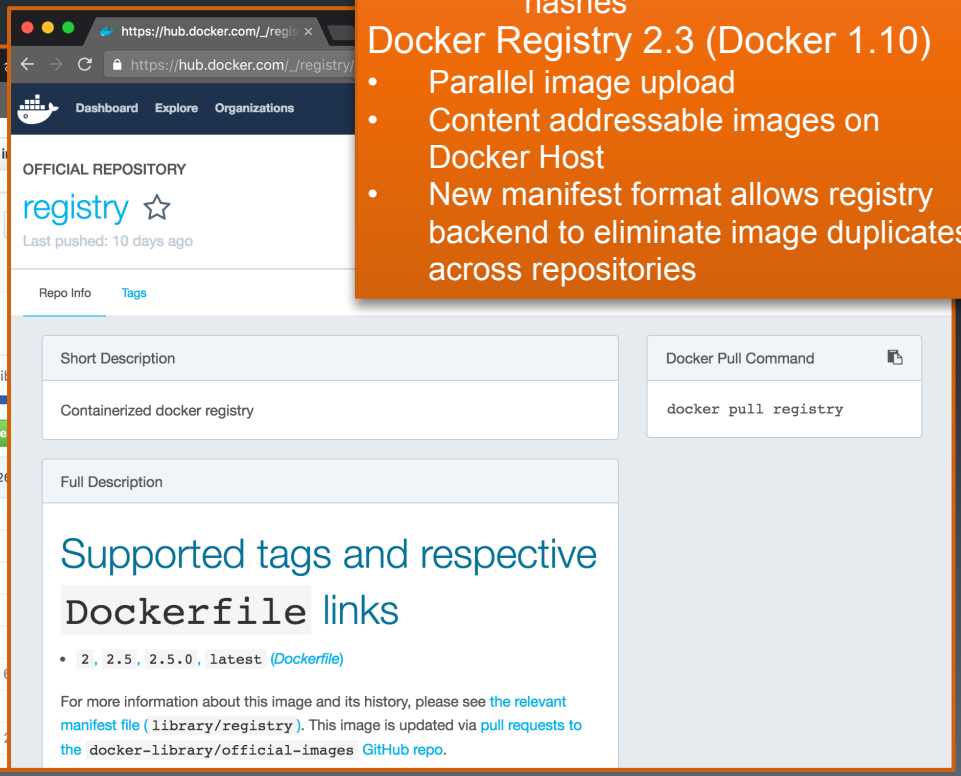
Private Registries

- Docker, Inc. provides an open-source Docker registry server
  - This makes it easy to run a private registry
  - The registry offers an API but does not currently have a UI
- Docker Registry v2 for Docker >= 1.6
  - Written in Go, parallel image download
  - https://docs.docker.com/registry
  - Beta support for proxying for Docker Hub
- Can be run from a Docker Hub image

Docker Registry 2.0 (Docker 1.6)
- parallel image download
- content addressable images in registry
  - Registry v1 IDs are random
  - Registry v2 IDs are sha256 hashes

Docker Registry 2.3 (Docker 1.10)
- Parallel image upload
- Content addressable images on Docker Host
- New manifest format allows registry backend to eliminate image duplicates across repositories

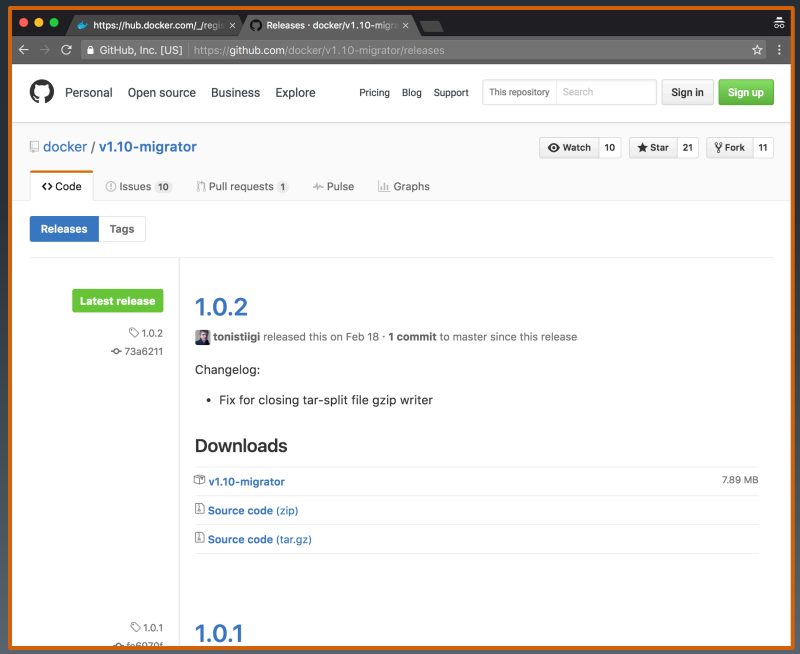




Migrating to Docker v1.10

- Docker Engine v1.10 completely changes the way Docker addresses image data on disk
  - Previously every image and layer used a randomly assigned UUID
  - In 1.10 IDs are based on a SHA256 hash of the image and layer data
- Advantages
  - Built-in ID collision avoidance
  - Image data integrity verification
  - Better layer sharing (even across builds and build systems)
  - Avoids duplicate downloads
  - No need to create layers for build instructions that don't modify the filesystem
- New Docker Engine internal "download manager"
  - Retries on failed downloads
  - Uploads in parallel
- To make current images accessible in v1.10 you must migrate them to the new format
  - Simply install the latest Docker engine and restart it
    - All images, tags and containers are automatically migrated to the new format the first time you start an updated Docker daemon
    - Before loading your container, the Docker daemon will calculate all needed checksums for your current data, and after it has completed, all your images and tags will have brand new IDs
    - Calculating SHA256 checksums for your files can take time on a host with many images
      - On average the migrator can process data at a speed of 100MB/s during which time the Docker daemon will be offline
  - Optionally you can use the image migration tool to migrate images while the current (old) Docker daemon is running
    - The migrator will pre-calculate all of the SHA hashes allowing the upgraded Docker v1.10 daemon to start immediately
    - Migrator:
      - https://github.com/docker/v1.10-migrator/releases
    - You can run the migrator in a container (of course)
      - docker run --rm -v /var/lib/docker:/var/lib/docker docker/v1.10-migrator
        - Use the --privileged switch with devicemapper

https://docs.docker.com/registry/spec/manifest-v2-2/



Running Networked Services in a Container

- The Docker Hub curated repository “registry” supports execution of a registry daemon on any Docker platform
- `$ docker container run -p 5000:5000 -d registry:2`
  - The example runs the latest v2 API registry image
  - The `-p` switch maps a host port to a container port ([hport]:[cport])
  - The registry service runs on port 5000 within the container, but the container registry port can be mapped to any host port you like
  - If the port is not mapped the registry will only be accessible from within the Docker host

```
docker@ubuntu:~$ docker container run -p 5000:5000 -d registry
692c9188b1c96a1f97891628d0f348a8415b73cdde302da4b0c1f8404854428e
docker@ubuntu:~$ docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
NAMES
692c9188b1c9        registry:latest     "docker-registry"   11 seconds ago      Up 10 seconds       0.0.0.0:5000-
>5000/tcp   happy_elion
docker@ubuntu:~$ curl http://localhost:5000
"\"docker-registry server"\"
docker@ubuntu:~$
```

Setting Container Environment Variables

- You can use environment variables to customize many aspects of the Registry execution
  - The `docker container run -e` switch sets environment variables in the container
  - Configuration files can also be used
- Sample configurations include support for:
  - Local file system repository storage
  - AWS S3
  - Ceph-S3
  - Azureblob
  - Swift, Glance and Glance-Swift
  - Others…
- Further Registry Documentation
  - https://docs.docker.com/registry/
  - https://github.com/docker/distribution

You can use:
`docker container run --env-file <filename>`
To load a set of environment variables from a file

```
docker container run \
    -e SETTINGS_FLAVOR=s3 \
    -e AWS_BUCKET=acme-docker \
    -e STORAGE_PATH=/registry \
    -e AWS_KEY=AKIAHSHB43HS3J92MXZ \
    -e AWS_SECRET=xdDowwlK7TJajV1Y7EoOZrmuPEJlHYcNP2k4j49T
\
    -e SEARCH_BACKEND=sqlalchemy \
    -p 5000:5000 \
    registry:2
```

docker tag

- The `docker image tag` subcommand allows you to create a new repository name and/or tag for an existing image
  - Think of this as a hard link (e.g. `$ ln source newlink`)
  - Tag is useful as it allows you to format repository names and tags to suit various registry systems

```
user@ubuntu:~$ docker image ls ubuntu
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              16.04               0ef2e08ed3fa        3 weeks ago         130 MB
ubuntu              latest              0ef2e08ed3fa        3 weeks ago         130 MB
ubuntu              xenial              0ef2e08ed3fa        3 weeks ago         130 MB
ubuntu              14.04               7c09e61e9035        3 weeks ago         188 MB
ubuntu              trusty              7c09e61e9035        3 weeks ago         188 MB
user@ubuntu:~$ docker image tag ubuntu:trusty bobsmith/ubuntu:prod_2017
user@ubuntu:~$ docker image ls ubuntu
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              16.04               0ef2e08ed3fa        3 weeks ago         130 MB
ubuntu              latest              0ef2e08ed3fa        3 weeks ago         130 MB
ubuntu              xenial              0ef2e08ed3fa        3 weeks ago         130 MB
ubuntu              14.04               7c09e61e9035        3 weeks ago         188 MB
ubuntu              trusty              7c09e61e9035        3 weeks ago         188 MB
user@ubuntu:~$ docker image ls bobsmith/ubuntu
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
bobsmith/ubuntu     prod_2017           7c09e61e9035        3 weeks ago         188 MB
user@ubuntu:~$
```

Pushing to a private Registry

- This example creates a new container and then commits an image from it
- To push the image to a registry the registry URL must be the first part of the repository name
  - Or a user name to imply pushing to the default Docker Hub
- Because repository names and tags are simply aliases, a single image ID may have many names/tags

```
docker@ubuntu:~$ docker container run -t -i ubuntu:14.04 /bin/bash
root@335a181a7be6:/#
   ### Do some configuring
root@335a181a7be6:/# exit
exit
docker@ubuntu:~$ docker container commit 335a181a7be6 bobsmith/thriftdev
84e5bc36d1640aef7cd5be9dba32297ab78d3685312859def2145059a0b0cf32
docker@ubuntu:~$ docker image ls
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
bobsmith/thriftdev      latest              84e5bc36d164        10 seconds ago      226 MB
ubuntu                  latest              2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                  trusty              2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                  trusty-20150218.1   2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                  14.04               2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                  14.04.2             2d24f826cb16        2 weeks ago         188.3 MB
docker@ubuntu:~$ docker image tag bobsmith/thriftdev localhost:5000/thriftdev
docker@ubuntu:~$ docker image ls
REPOSITORY                 TAG                 IMAGE ID            CREATED             SIZE
bobsmith/thriftdev         latest              84e5bc36d164        2 minutes ago       226
MB
localhost:5000/thriftdev   latest              84e5bc36d164        2 minutes ago       226
MB
ubuntu                     14.04               2d24f826cb16        2 weeks ago         188.3
MB
ubuntu                     latest              2d24f826cb16        2 weeks ago         188.3
MB
ubuntu                     trusty              2d24f826cb16        2 weeks ago         188.3
MB
ubuntu                     14.04.2             2d24f826cb16        2 weeks ago         188.3
MB
ubuntu                     trusty-20150218.1   2d24f826cb16        2 weeks ago         188.3
MB
docker@ubuntu:~$ docker image push localhost:5000/thriftdev
The push refers to a repository [localhost:5000/thriftdev] (len: 1)
Sending image list
Pushing repository localhost:5000/thriftdev (1 tags)
8beb87107cdb: Pushed
f668bc0d79c1: Pushed
```

Pulling Images

- Repository names embed the information used to identify an image's registry
- For Docker Hub
  - uid/name
- For non Docker Hub:
  - url/name
- This allows Docker to automatically recover missing images from the appropriate registry
  - Images can be pulled or run by registry name and optional tag

```
docker@ubuntu:~$ docker image ls
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
localhost:5000/thriftdev    latest              84e5bc36d164        24 minutes ago      226 MB
ubuntu                      trusty              2d24f826cb16        2 weeks ago         188.3
MB
ubuntu                      14.04               2d24f826cb16        2 weeks ago         188.3
MB
ubuntu                      latest              2d24f826cb16        2 weeks ago         188.3
MB
ubuntu                      trusty-20150218.1   2d24f826cb16        2 weeks ago         188.3
MB
ubuntu                      14.04.2             2d24f826cb16        2 weeks ago         188.3
MB
docker@ubuntu:~$ docker image rm localhost:5000/thriftdev
Untagged: localhost:5000/thriftdev:latest
Deleted: 84e5bc36d1640aef7cd5be9dba32297ab78d3685312859def2145059a0b0cf32
docker@ubuntu:~$ docker image ls
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
ubuntu                  trusty              2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                  trusty-20150218.1   2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                  14.04               2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                  latest              2d24f826cb16        2 weeks ago         188.3 MB
ubuntu                  14.04.2             2d24f826cb16        2 weeks ago         188.3 MB
docker@ubuntu:~$ docker container run -t -i localhost:5000/thriftdev:latest /bin/bash
Unable to find image 'localhost:5000/thriftdev:latest' locally
Pulling repository localhost:5000/thriftdev
84e5bc36d164: Download complete
511136ea3c5a: Download complete
fa4fd76b09ce: Download complete
1c8294cc5160: Download complete
117ee323aaa9: Download complete
2d24f826cb16: Download complete
Status: Downloaded newer image for localhost:5000/thriftdev:latest
root@a89e744e7d1d:/# exit
exit
docker@ubuntu:~$ docker image ls
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
localhost:5000/thriftdev    latest              84e5bc36d164        28 minutes ago      226 MB
ubuntu                      trusty-20150218.1   2d24f826cb16        2 weeks ago         188.3
```

Summary

- Registries provide centralized lookup and storage facilities for repositories and their images
- Docker Registry lookups are based on x[/y[/z]][:t] formatted names
  - **/** reserved character separating discrete components of the image path
  - **:** reserved character separating repository path from image tag (also separates hostname from port in x position)
  - **t** is the image tag name and defaults to “latest” if not present
  - **x** lookups refer an official Docker Hub repository name (“ubuntu”)
  - **x/y** lookups
    - namespace/repo form: **x** is a DockerHub namespace (account/user name), **y** is the repository name (fred/specialrepo)
    - host/repo form: **x** is the registry hostname[:port], **y** is the repository name (reg.example.com:5005/specialrepo)
  - **x/y/z** lookups interpret **x** as registry hostname:port, **y** as namespace and **z** as the repository name
    - e.g. “reg.example.com:5005/fred/specialrepo:latest” [FQIN??]
- The Docker Registry code can be executed directly from images found on Docker Hub
  - Source is also available on github
- Networked services can run in containers using the -p switch to map host ports to container ports
- Docker supplies several commands to support registry operations
  - docker login *
  - docker logout
  - docker search *
  - docker pull *
  - docker push *
  - docker tag (creates appropriate names for image registry operations)
  - docker run/build/create (implicitly pull missing images) *

** Only these commands cause Docker engine to interact with Registries*

Lab 2

- Working with Images and Registries

3: Dockerfiles I

Objectives

- Explain the purpose of Dockerfiles
- List the main Dockerfile instructions
- Understand the docker build process
- Examine important Dockerfile concepts
  - Build Cache

Dockerfiles and Context

- Docker can build images automatically by reading the instructions from a Dockerfile
  - A Dockerfile is a text document that contains all the commands you would normally execute manually in order to build a Docker image
- By calling `docker image build` from your terminal, you can have Docker construct your image step by step, executing the instructions successively
  - *# docker image build /some/path/*
- The path supplied to docker image build is the context of the build
  - Also known as the source repository
  - The build is run by the Docker engine, not by the CLI, so the whole context must be transferred to the daemon
    - The Docker CLI reports "Sending build context to Docker daemon" when the context is sent to the daemon
  - Therefore, in most cases it is best to put each Dockerfile in an empty directory, adding only the files needed for building that Dockerfile
    - This minimizes the context transferred to the Docker daemon
    - Use *.dockerignore* to control what files to not transfer
      - https://docs.docker.com/engine/reference/builder/#/dockerignore-file

Dockerfiles

- `docker container commit`
  - A practical way to create images interactively
  - An impractical way to create identical or incrementally improved images
- `docker image build`
  - The build command creates images automatically from a Dockerfile
- Dockerfile
  - A text document containing all the commands you would normally execute manually in order to build a Docker image with docker commit
  - To build an image you can place a file called “Dockerfile” at the root of your repository and call `docker image build` with the path of your source repository
    - `$ docker image build .`
  - Each command in a *Dockerfile* creates a new image layer

```
user@ubuntu:~/thriftdev$ docker image ls -a
REPOSITORY                  TAG                 IMAGE ID
my-thrift-image             latest              ed1726b77
<none>                      <none>              ce4a7261a
<none>                      <none>              78a6f751f
<none>                      <none>              10c6eacbc
<none>                      <none>              f768c2270
ubuntu                      14.04               b72889fa8
```

```
user@ubuntu:~$ mkdir thriftdev
user@ubuntu:~$ cd thriftdev
user@ubuntu:~/thriftdev$ vim dockerfile
user@ubuntu:~/thriftdev$ cat dockerfile
# Version: 1.0.0
FROM ubuntu:14.04
LABEL Maintainer Bob Smith "bob@example.com"
RUN apt-get update
RUN apt-get install -y git
RUN echo 'thrift dev image v1.0.0' > /README.md
LABEL license "Apache 2.0"
user@ubuntu:~/thriftdev$ docker image build -t my-thrift-
image .
Sending build context to Docker daemon 2.048 kB
Step 1 : FROM ubuntu:14.04
 ---> b72889fa879c
Step 2 : LABEL Maintainer Bob Smith "bob@example.com"
 ---> Running in 9e9c5d061e45
 ---> f768c2270a02
Removing intermediate container 9e9c5d061e45
Step 3 : RUN apt-get update
 ---> Running in 2a49c5457c26
...
 ---> 10c6eacbc55b
Removing intermediate container 2a49c5457c26
Step 4 : RUN apt-get install -y git
 ---> Running in f129e6217fad
...
 ---> 78a6f751fbe2
Removing intermediate container f129e6217fad
Step 5 : RUN echo 'thrift dev image v1.0.0' > /README.md
 ---> Running in 222735f19aef
 ---> ce4a7261a4d2
Removing intermediate container 222735f19aef
Step 6 : LABEL license "Apache 2.0"
 ---> Running in a0969b45283f
 ---> ed1726b77df1
Removing intermediate container a0969b45283f
Successfully built ed1726b77df1
user@ubuntu:~/thriftdev$ ls -l
total 4
-rw-rw-r-- 1 user user 200 Apr 19 07:42 dockerfile
user@ubuntu:~/thriftdev$
```

Dockerfile Processing

- A Dockerfile contains a series of instructions paired with arguments
- Each instruction should be in upper-case and be followed by an argument
  - `FROM ubuntu:16.04`
- Instructions are processed from the top down
  - The ordering of instructions is typically important
- Each instruction commits a new image layer
- To execute an instruction Docker:
  - Runs a container from the previous image
  - Executes the instruction on the container
  - Commits a new image from the modified container (may be metadata only)
  - Deletes the working container
  - Repeats for the next instruction until all instructions have been applied
- If one of the instruction in the Dockerfile build crashes you will still have all of the images leading up to the failed step
  - This is useful for debugging

```
1   # A basic apache server. To use either add or bind mount content under /var/www
2   FROM ubuntu:12.04
3
4   RUN apt-get update && apt-get install -y apache2 && apt-get clean && rm -rf /var/lib/apt/lists/*
5
6   ENV APACHE_RUN_USER www-data
7   ENV APACHE_RUN_GROUP www-data
8   ENV APACHE_LOG_DIR /var/log/apache2
9
10  EXPOSE 80
11
12  CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]
```

Dockerfile construction

- `FROM`
  - Should always be the first instruction
  - Specifies a base image that the *Dockerfile* will operate on
- `MAINTAINER`
  - Tells Docker the author's name and email (deprecated in favor of `LABEL`)
- `RUN`
  - Executes commands on the current image
  - RUN instructions execute in a shell using the command wrapper `/bin/sh -c`
  - If you wish to execute without a shell (for example, to avoid shell string munging), you can specify the instruction in exec format:
    - `RUN [ "apt-get", " install", "-y", "nginx" ]`
    - An array specifies the command to be executed and each parameter to pass to the command
- `EXPOSE`
  - Tells Docker that the application in this container will use this specific port in the container
  - You must map the port to the host using the docker run command to make it externally available
  - You can specify multiple `EXPOSE` instructions
- `CMD`
  - Defines an executable for a container
  - CMD has several forms:
    - CMD ["executable","param1","param2"] (exec form)
    - CMD command param1 param2 (shell form, performs shell processing on the provided command line [e.g. $HOME substitution, etc.])
  - There should only be one CMD instruction in a *Dockerfile*
  - The CMD will not be executed if an executable is specified on the docker run command line, e.g:
    - `$ docker container run -it nginx /bin/bash`

```
# Basic install of couchdb
#
# This will move the couchdb http server to port 8101 so adjust the port for your needs.
#
# Currently installs couchdb 1.3.1

FROM ubuntu
MAINTAINER Kimbro Staken

RUN echo "deb http://us.archive.ubuntu.com/ubuntu/ precise universe" >> /etc/apt/sources.list
RUN apt-get -y update
RUN apt-get install -y g++
RUN apt-get install -y erlang-dev erlang-manpages erlang-base-hipe erlang-eunit erlang-nox erlang-xmerl erlang-inets

RUN apt-get install -y libmozjs185-dev libicu-dev libcurl4-gnutls-dev libtool wget

RUN cd /tmp ; wget http://www.bizdirusa.com/mirrors/apache/couchdb/source/1.3.1/apache-couchdb-1.3.1.tar.gz

RUN cd /tmp && tar xvzf apache-couchdb-1.3.1.tar.gz
RUN apt-get install -y make
RUN cd /tmp/apache-couchdb-* ; ./configure && make install

RUN printf "[httpd]\nport = 8101\nbind_address = 0.0.0.0" > /usr/local/etc/couchdb/local.d/docker.ini

EXPOSE 8101

CMD ["/usr/local/bin/couchdb"]
```

Docker build options

- `-f`
  - Allows you to specify a particular file to use as the build input rather than ./Dockerfile
- `-t`
  - The tag switch allows you to name repositories ([userid, url]/reponame) and add tags (: 0.9.2)
- `--no-cache=[true, false]`
  - The Docker build cache stores copies of previously created images
  - This switch can tell Docker to rebuild an image event when it can be copies from the cache
- `--squash`
  - An experimental feature that squashes newly built layers into a single new layer

```
user@ubuntu:~$ time docker image build -t bobsmith/thriftdev:0.1.0
thriftdev
Sending build context to Docker daemon 2.048 kB
Step 1 : FROM ubuntu:14.04
 ---> b72889fa879c
Step 2 : Label maintainer Bob Smith "bob@example.com"
 ---> Using cache
 ---> f768c2270a02
Step 3 : RUN apt-get update
 ---> Using cache
 ---> 10c6eacbc55b
Step 4 : RUN apt-get install -y git
 ---> Using cache
 ---> 78a6f751fbe2
Step 5 : RUN echo 'thrift dev image v1.0.0' > /README.md
 ---> Using cache
 ---> ce4a7261a4d2
Step 6 : LABEL license "Apache 2.0"
 ---> Using cache
 ---> ed1726b77df1
Successfully built ed1726b77df1

real        0m0.041s
user        0m0.000s
sys         0m0.019s
user@ubuntu:~$
```

Build Cache

- The build cache allows multiple *Dockerfiles* to share the same image layers
- The build cache also allows multiple builds of the same *Dockerfile* to share the same image layers
- This optimizes build times and saves disk space
- While building, Docker compares instructions against all child images of the base image to see if one was built using the exact same instruction
  - i.e. the parent image SHA and the instruction string must match the cache image parent and instruction
- All dockerfile commands are run at build time
  - Thus an image's repository index may grow old if the image is continually sourced from the cache
  - The example below uses an ENV instruction to force the follow on images to be rebuilt when the `REFRESHED_AT` date is updated

```
FROM ubuntu: 14.04
LABEL Maintainer Bob Smith
"bob@example.com"
ENV REFRESHED_AT 2017-12-01
RUN apt-get -qq update
```

Image History

- The history command displays the parents of a given image and the instructions which created them
  - Each line displays:
    - Image UUID
    - Time created
    - Executed statement creating the image
    - Size added to the overall file system by the command
- You can use the `--no-trunc` switch to view the entire create command
  - Allows you to rebuild the original *Dockerfile* if need be
  - Acripts can easily automate the *Dockerfile* reverse engineering process
    - https://github.com/CenturyLinkLabs/dockerfile-from-image
    - Note: Having the *Dockerfile* is not enough to rebuild most images, you will also need all of the binaries, libs and sources copied into the images with COPY/ADD/etc…

```
$ docker image history bobsmith/thriftdev:0.1.0
IMAGE               CREATED             CREATED BY                                      SIZE
99fdcb5de9a4        4 hours ago         /bin/sh -c #(nop) EXPOSE 80/tcp                 0 B
9fa98b0dc393        4 hours ago         /bin/sh -c echo 'thrift dev image v1.0.0' > /   24 B
3c0ab41341eb        4 hours ago         /bin/sh -c apt-get install -y git
37.69 MB
1c09982ab4ce        4 hours ago         /bin/sh -c apt-get update
20.59 MB
3cb1d6967c66        4 hours ago         /bin/sh -c #(nop) MAINTAINER Bob Smith "bob@e   0 B
2d24f826cb16        2 weeks ago         /bin/sh -c #(nop) CMD [/bin/bash]               0 B
<missing>           2 weeks ago         /bin/sh -c sed -i 's/^#\s*\(deb.*universe\)$/
1.895 kB
<missing>           2 weeks ago         /bin/sh -c echo '#!/bin/sh' > /usr/sbin/polic
```

Additional Dockerfile Instructions

- CMD sets a command/arguments to run in the image if no arguments are passed to docker container run
- ENTRYPOINT makes your image behave like a binary

- ENTRYPOINT
  - Runs the specified program
    - `ENTRYPOINT ["/usr/sbin/nginx"]`
  - Unlike CMD, additional arguments supplied to a docker run of an image with an ENTRYPOINT are supplied directly to the ENTRYPOINT program
    - If CMD and ENTRYPOINT are present the CMD array is passed to the ENTRYPOINT program as parameters and overridden if command line args are supplied during docker run
    - This allows for natural "command style" execution of containers:
      - Assuming thrift is an image with an entry point that runs the thrift compiler:
        - `$ docker container run thrift --gen java appcore.thrift`
- ENV
  - Sets environment variables
  - Environment variables can be referenced in many docker instructions using a `$` prefix
  - Environment variables can also be set on the docker run command line using the `-e` switch

```
ENV APP_DIR /home/webapp/
WORKDIR $APP_DIR
```

- LABEL
  - Adds metadata to an image (e.g. LABEL version="1.0")
- USER
  - Specifies a user:group that the image should be run as (by name or id)
  - Can specify a user ("bob") or a user and group ("bob:emp")
- WORKDIR
  - Sets the working directory for the container and the ENTRYPOINT/CMD

```
LABEL version="1.0"
USER webapp
ENV APP_DIR /home/webapp/
WORKDIR /opt/webapp/db
RUN bundle install
WORKDIR $APP_DIR
ENTRYPOINT [ "rackup" ]
```

Using ENVs in a Dockerfile

- Environment variables declared with the `ENV` statement can be used in *Dockerfile* instructions
- Notated in the *Dockerfile* either with `$variable_name` or `${variable_name}`
- Environment variables are supported by the following instructions:
  - `ADD`
  - `COPY`
  - `ENV`
  - `EXPOSE`
  - `LABEL`
  - `USER`
  - `WORKDIR`
  - `VOLUME`
  - `STOPSIGNAL`

```
LABEL version="1.0"
USER webapp
ENV APP_DIR /home/webapp/
WORKDIR /opt/webapp/db
RUN bundle install
WORKDIR $APP_DIR
ENTRYPOINT [ "rackup" ]
```

Summary

- *Dockerfiles* allow Docker images to be scripted
  - This makes image construction highly repeatable
- There are several key *Dockerfile* instructions and each creates a separate image
- The Docker build cache saves copies of each instruction's image, potentially speeding up future builds

Dockerfile Documentation:
https://docs.docker.com/reference/builder/

Lab 3

- Creating Dockerfiles

4: Dockerfiles II

Objectives

- Explore advanced *Dockerfile* instructions
- Understand GitHub and Automated Builds
- Learn *Dockerfile* best practices
- Examine the Docker Hub standard library
- Explain the nature of
  - docker load and docker save
  - docker import and docker export

ADD & COPY

- `ADD`
  - Adds files and directories from the build environment into the image
  - Specifies a source and a destination for files:
    - `ADD software.lic /opt/application/software.lic`
      - Copies software.lic from the build directory to /opt/application/software.lic in the image
    - The source of the file can be a URL, filename, or directory
    - The source must be inside the build context
    - If the destination ends in a /, docker considers the source a directory
    - ADD will automatically unpack source archives
      - gzip
      - bzip2
      - xz
    - New files and directories will be created with a mode of 0755 and a UID and GID of 0
- `COPY`
  - Does is almost exactly like ADD but without URL and Archive support
  - Also used in multi-stage builds (covered later)
- To speed the build and keep images lean you can exclude files and directories when COPYing or ADDing to the image by adding a *.dockerignore* file to the context directory

ONBUILD

- `ONBUILD` adds triggers to images
- A trigger is executed when the image is used as the basis of another image
- Uses:
  - Creating an image that needs source code from a specific location that is not yet available
  - Execute a build script specific to the environment in which the image is built
- Triggers insert a new instruction in the build process, as if it were specified right after the `FROM` instruction
- The trigger can be any build instruction except `FROM`, `MAINTAINER`, and `ONBUILD`
- `ONBUILD` triggers are executed in the order specified in the parent image and are only inherited once
  - by children, not grandchildren

```
# bobs/apache
FROM ubuntu:14.04
LABEL Maintainer Bob Smith
"bob@example.com"
RUN apt-get update
RUN apt-get install -y apache2
ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2
ONBUILD ADD . /var/www/
EXPOSE 80
ENTRYPOINT ["/usr/sbin/apache2"]
CMD ["-D", "FOREGROUND"]
```

```
# bobs/webapp
#   The important parts of this image
#   are the static files in the build
#   dir.
FROM bobs/apache
### BUILD TRIGGER happens here
LABEL Maintainer Bob Smith
"bob@example.com"
ENV APPLICATION_NAME webapp
ENV ENVIRONMENT development
```

VOLUME

- `VOLUME`
  - Creates a mount point with the specified name
  - More on volumes in the next chapter
- The value can be
  - `VOLUME /var/log`
  - `VOLUME /var/log /var/db`
  - `VOLUME ["/var/log/", "/var/db/"]`
- The docker container run command initializes the newly created volume with any data that exists at the specified location within the base image
  - `FROM ubuntu`
  - `RUN mkdir /myvol`
  - `RUN echo "hello world" > /myvol/greeting`
  - `VOLUME /myvol`
    - Results in an image that causes docker container run to create a new volume and to mount it at */myvol*, then to copy the greeting file into the newly created volume
    - Note: data changed within the volume by build steps after the volume is declared will be discarded

New Dockerfile instructions

- Docker 1.9 added two new *Dockerfile* instructions
  - `ARG` - defines a variable that users can pass at build-time
  - `STOPSIGNAL` - sets the signal that will be sent to the container by docker stop (SIGTERM is the default)

```
$ docker image build --build-arg
user=frodo .
```

```
#parameterized
Dockerfile
FROM busybox
ARG user
USER $user
STOPSIGNAL SIGHUP
```

- Docker 1.12 added two new *Dockerfile* instructions
  - `HEALTHCHECK` - tells Docker how to test a container to check that it is still working
  - `SHELL` - allows the default shell used for the *shell* form of commands to be overridden

```
HEALTHCHECK --interval=5m --timeout=3s \
  CMD curl -f http://localhost/ || exit 1
```

```
FROM windowsservercore

# Executed as cmd /S /C echo default
RUN echo default

# Executed as cmd /S /C powershell -command Write-Host default
RUN powershell -command Write-Host default

# Executed as powershell -command Write-Host hello
SHELL ["powershell", "-command"]
RUN Write-Host hello

# Executed as cmd /S /C echo hello
SHELL ["cmd", "/S"", "/C"]
RUN echo hello
```

Compound Dockerfiles

- Dockerfiles can contain multiple `FROM` statements
  - Each `FROM` defines a new image stack
  - Like building separate Dockerfiles which share the same context
- Multi-stage builds (versions 17.04+)
  - To use artifacts and outputs from prior images in the same build context, use:
    - `COPY --from=<base_image_name/num>`
      - A base *image name* is given in its `FROM` instruction
        - i.e. `FROM golang:1.7.5 as build-env`

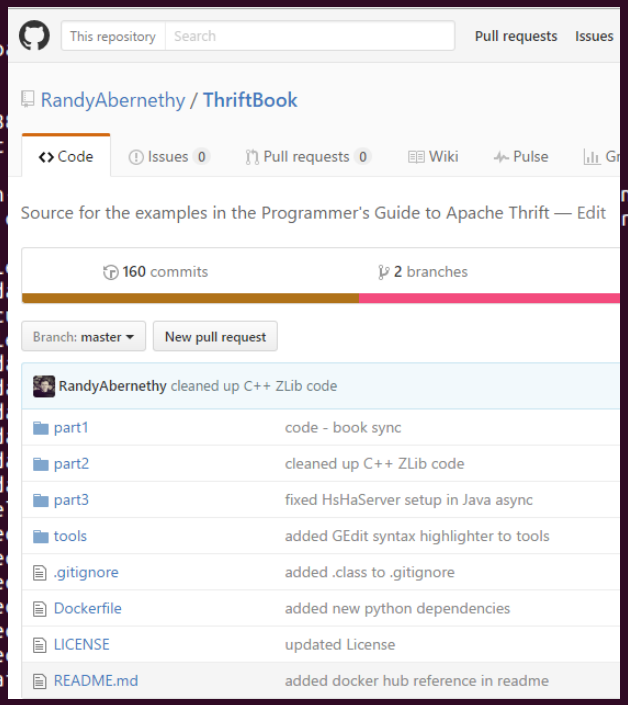
```
# Multiple images example
FROM ubuntu
RUN echo foo > bar
# Will output something like =>
907ad6c2736f

FROM ubuntu
RUN echo moo > oink
# Will output something like =>
695d7793cbe4

# Results in two images 907ad with /bar
and # 695d7 with /oink



# Multi-stage build example:
# First stage to build the application
FROM maven:3.5.0-jdk-8-alpine as build-
env
ADD ./pom.xml pom.xml
ADD ./src src/
RUN mvn clean package

# Final stage to define our minimal
runtime
FROM openjdk:8-jre
COPY --from=build-env target/app.jar
```

Building Git Repos

- `build` can be passed a GitHub URL as a context
  - The client recursively clones the repository and its submodules using a `git clone --depth 1 --recursive`
  - Upon successful completion the directory is sent to the Docker daemon as the context
  - Local clones give you the ability to access private repositories using local user credentials, VPNs, etc.
- The repo suffix controls the branch/tag/commit and directory built

myrepo.git	refs/heads/master	/
myrepo.git#mytag	refs/tags/mytag	/
myrepo.git#mybranch	refs/heads/mybranch	/
myrepo.git#abcdef	sha1 = abcdef	/
myrepo.git#:myfolder	refs/heads/master	/myfolder
myrepo.git#master:myfolder	refs/heads/master	/myfolder
myrepo.git#mytag:myfolder	refs/tags/mytag	/myfolder
myrepo.git#mybranch:myfolder	refs/heads/mybranch	/myfolder
myrepo.git#abcdef:myfolder	sha1 = abcdef	/myfolder

```
user@ubuntu:~$ docker build http://github.com/RandyAbernethy/ThriftBook.git
Sending build context to Docker daemon 986.6 kB
Step 1 : FROM ubuntu:14.04
 ---> ff6011336327
Step 2 : MAINTAINER Randy Abernethy "ra@ap
 ---> Running in 641c2445d889
 ---> 11c368fdeb3a
Removing intermediate container 641c2445d8
Step 3 : RUN apt-get update &&     apt-get
nt-dev         libssl-dev         libtool
-all         python-all-dbg         python
on-pip         python3-pip &&     apt-get
 ---> Running in ae239b39ae6a
Ign http://archive.ubuntu.com trusty InRel
Get:1 http://archive.ubuntu.com trusty-upd
Get:2 http://archive.ubuntu.com trusty-sec
Get:3 http://archive.ubuntu.com trusty Rel
Get:4 http://archive.ubuntu.com trusty-upd
Get:5 http://archive.ubuntu.com trusty-upd
Get:6 http://archive.ubuntu.com trusty-upd
Get:7 http://archive.ubuntu.com trusty-upd
Get:8 http://archive.ubuntu.com trusty-upd
Get:9 http://archive.ubuntu.com trusty-upd
Get:10 http://archive.ubuntu.com trusty Re
Get:11 http://archive.ubuntu.com trusty-se
Get:12 http://archive.ubuntu.com trusty-se
Get:13 http://archive.ubuntu.com trusty-se
Get:14 http://archive.ubuntu.com trusty-se
Get:15 http://archive.ubuntu.com trusty-se
Get:16 http://archive.ubuntu.com trusty-se
Get:17 http://archive.ubuntu.com trusty/ma
```



Docker registry:2.2 Dockerfile case study

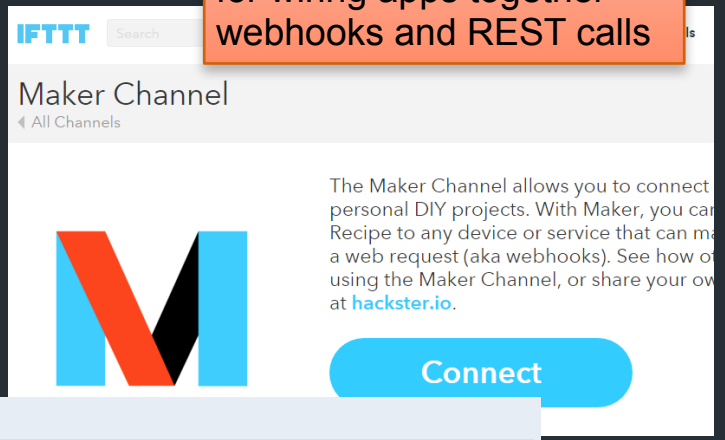
Docker/GitHub

- **Docker Hub** images are frequently sourced from *Dockerfiles* linked in from **github** or other URLs

Automated Builds

- Docker Hub Web Hooks fire in response to a new image push to a repo
  - Can be used to run CI/CD tasks and/or trigger other web services
- DVCS platforms like GitHub and BitBucket support WebHooks which can invoke a Docker Hub build when code is pushed
  - Docker Hub WebHooks can then invoke CI platforms creating a build chain
- Docker Hub repositories can be manually uploaded or "Automated Build"
  - AutoBuild allows you to configure a target GitHub/BitBucket repo and *Dockerfile* to use for the build
- Once configured you can not push to an automated build, rather you push source commits to the DVCS and the DVCS triggers the build on Docker Hub
  - When the build completes Docker hub drops a new image in the Docker Hub repo, triggering the relevant Docker hub web hooks
- Docker Hub can also be configured to create a new tagged image in the Docker Hub repository for each new branch or tag committed to a GitHub repository

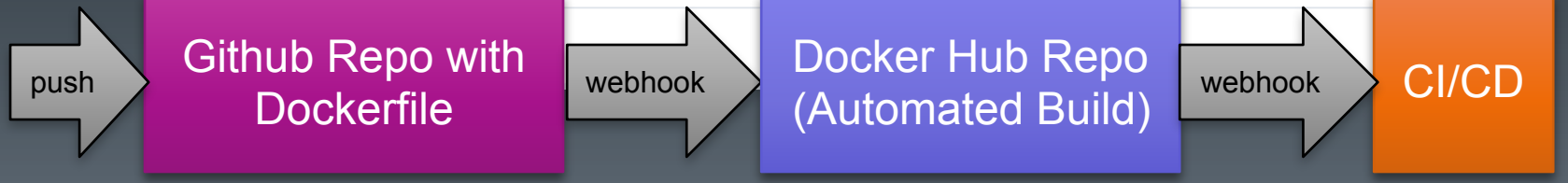
"If this then that" (ifttt.com) is a handy free internet site for wiring apps together webhooks and REST calls



Here are a few examples: Show less

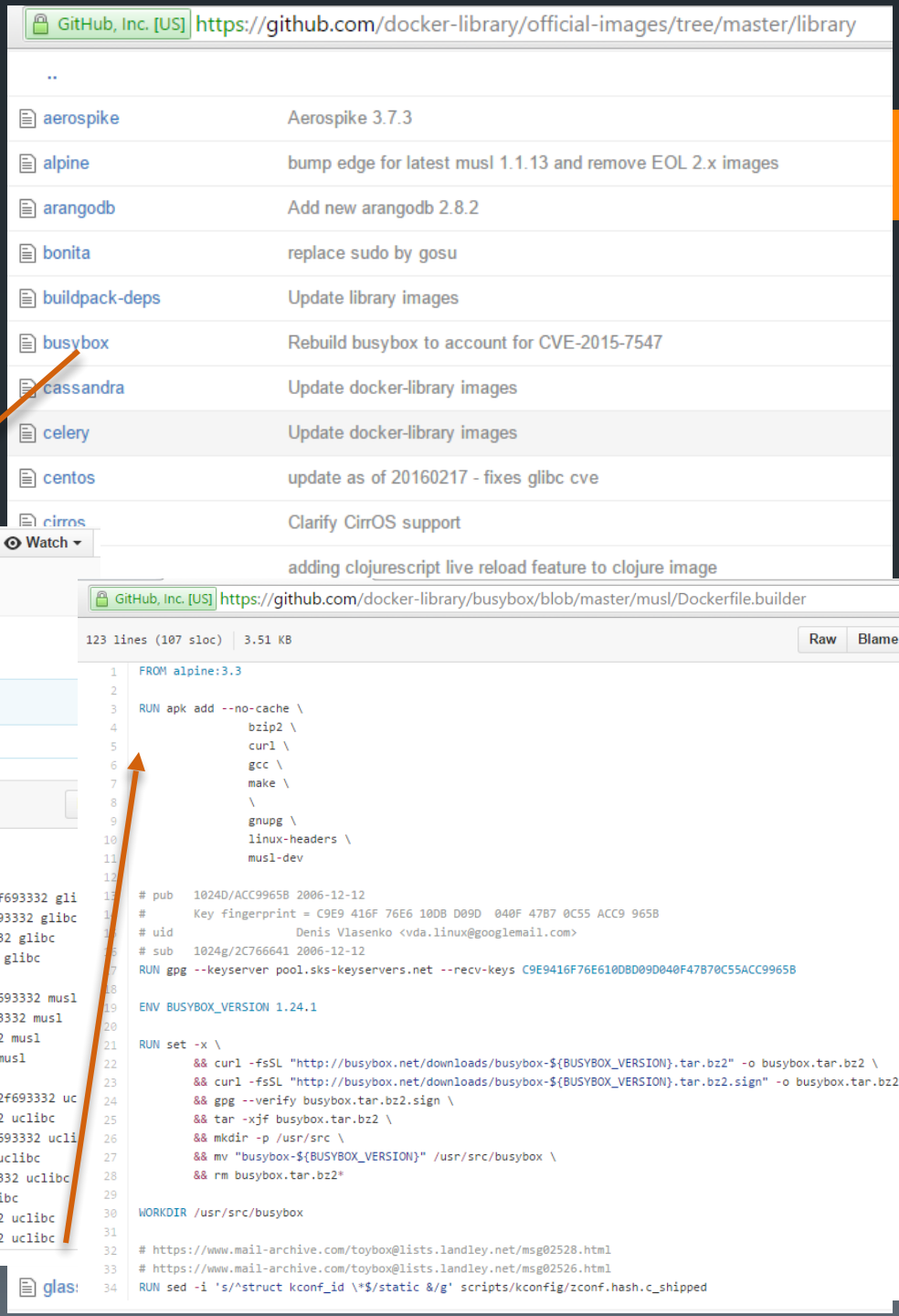
Scenario	Type	Name	Docker Tag Name	Matches	Docker Tag Built
Hardcoded names are ok	Branch	master	latest	master	latest
Match tags that look like a version	Tag	/^[0-9.]+$/	release-{sourceref}	1.2.0	release-1.2.0
Allow trailing modifiers in the tag	Tag	/^[0-9.]+/	release-{sourceref}	1.2.0-rc	release-1.2.0-rc

Type	Name	Dockerfile Location	Docker Tag Name	
Branch	master	/	java	+ Trigger
Branch	/.*/ This will target all branches	/	Same as branch	-



Standard Library

- Official images are intended to be
  - Base images for production releases
  - Learning tools !



Dockerfile/Image Tips

- Docker image authors may ask:
  - Is my image easy to use?
  - Is my image easy to base another image on?
  - Does my image behave in a performant manner?
- Use a *Dockerfile* linter
- Use `LABEL` to replace the MAINTAINER instruction to set the author of the image
  - Provides contact information for users
- Know the Differences Between CMD and ENTRYPOINT
  - CMD set command/arguments to run in the image if no args are passed to docker run
  - ENTRYPOINT makes your image behave like a binary
- Know the Difference Between Array and String Forms of CMD and ENTRYPOINT
  - Passing an array will result in the exact command being run
  - Passing a string will run the command in a shell (/bin/sh –c)
- Always exec in Wrapper Scripts
  - Images using a CMD/ENTRYPOINT script target will cause docker to send signals to the script, use exec in the script to run programs so that the program will receive the signals (otherwise only the script will)
- Always EXPOSE Important Ports
  - Exposed ports show up in ps listings, are visible in metadata and are automatically linked when using container linking
- Containers should be ephemeral
  - Stop and destroy the container then create a new one with an absolute minimum of set-up and configuration
- Use a *.dockerignore* file
  - For faster uploading and smaller images, use a *.dockerignore* to exclude files/directories from the build context
  - E.g. unless .git is needed you should add it to *.dockerignore*, saving many megabytes of upload time
- Avoid installing unnecessary packages
  - Reduces complexity, dependencies, file sizes, and build times
  - E.g. you don't need to include a text editor in a database image
- Run only one process per container
  - In almost all cases, you should only run a single process in a single container
  - Decoupling apps into multiple containers makes it easier to scale horizontally and reuse containers
  - If a service depends on another service use container linking
- Minimize the number of layers
  - Balance *Dockerfile* readability (and thus long-term maintainability) and layer minimization
  - Be strategic and cautious about the number of layers you use
- Sort multi-line arguments
  - Ease later changes by sorting multi-line arguments alphanumerically
  - Helps avoid duplication and make the list easier to update
  - Also makes Pull Requests easier to read and review
- Build cache
  - Use the build cache wisely

Dockerfile linters:
http://hadolint.lukasmartinelli.ch
https://atom.io/packages/linter-docker
https://www.npmjs.com/package/dockerlint
https://www.fromlatest.io/#/
https://access.redhat.com/labsinfo/linterfordockerfile

```
RUN \
    apt-get update && \
    apt-get install -y \
        bzr \
        git \
        mercurial
```

https://docs.docker.com/engine/userguide/eng-image/dockerfile_best-practices

Archiving Repositories

- `docker image save`
  - Produces a tarred repository to the standard output stream
    - `-o` to write to a file instead of STDOUT
  - Contains all parent layers, and all tags + versions, or specified repo:tag, for each argument provided
- `docker image load`
  - Loads a tarred repository from the standard input stream
    - `-i` to read from a tar archive file
  - Restores both images and tags

```
user@ubuntu:~$ docker image ls datav
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
datav               latest              83a4c617e24e        2 months ago        161 MB
user@ubuntu:~$ docker image save datav
Cowardly refusing to save to a terminal. Use the -o flag or redirect.
user@ubuntu:~$ docker image save -o 'datav.repo' datav
user@ubuntu:~$ ls -l datav.repo
-rw------- 1 user user 169427456 Mar 22 22:43 datav.repo
user@ubuntu:~$ docker image rm datav
Untagged: datav:latest
Deleted: sha256:83a4c617e24e34abcaaf91b78dc7fdaa59b316fa72685781db5a563d1c09fa43
Deleted: sha256:2ec96e8fd9f4eb13f06500839aa111dccc873c46ae7c935b0a7f4141985661d5
Deleted: sha256:37cd5bf8fcd35602a59066398cfa317c8e1778b95321f95bced5858fe098672d
Deleted: sha256:b6ca02dfe5e62c58dacb1dec16eb42ed35761c15562485f9da9364bb7c90b9b3
user@ubuntu:~$ docker image load -i datav.repo
b6ca02dfe5e6: Loading layer [==================================================>] 128.9 MB/128.9 MB
40650ed10dd8: Loading layer [==================================================>] 15.11 MB/15.11 MB
8ebc4e08de15: Loading layer [==================================================>]  25.4 MB/25.4 MB
Loaded image: datav:latest
user@ubuntu:~$ docker image ls datav
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
datav               latest              83a4c617e24e        2 months ago        161 MB
user@ubuntu:~$
```

Archiving Containers

- `docker container export`
  - Docker containers can be exported
  - docker export saves an entire container file system (with all underlying layers) to a tar archive
  - “docker export” applies to containers and is similar to “docker save” which applies to images
  - Unlike save, export flattens the output into a single filesystem
- `docker image import`
  - Creates an empty image and imports the contents of the specified tarball into it

```
user@ubuntu:~$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
dfe5f708a05d        cirros:latest       "/sbin/init"        11 seconds ago      Created                                 hungry_hodgkin
user@ubuntu:~$ docker container export -o 'se.tar' hungry_hodgkin
user@ubuntu:~$ ls -l se.tar
-rw------- 1 user user 8098816 Mar 22 22:58 se.tar
user@ubuntu:~$ cat se.tar | docker image import - newbase:first_cut
sha256:cf06f8d75e7b13bab6971f338bdc2a07cbfd2ec1eae9e341b124f8474eef5a37
user@ubuntu:~$ docker image ls newbase
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
newbase             first_cut           cf06f8d75e7b        7 seconds ago       7.73 MB
user@ubuntu:~$ docker image history newbase:first_cut
IMAGE               CREATED             CREATED BY          SIZE                COMMENT
cf06f8d75e7b        20 seconds ago                          7.73 MB             Imported from -
user@ubuntu:~$ docker container run -it newbase:first_cut /bin/sh
/ #
```

Summary

- *Dockerfile* instructions include new entries:
  - `ARG`, `STOPSIGNAL`, `HEATHCHECK`, `SHELL`
- Docker can build directly from GitHub repos
- Various *Dockerfile* best practices can improve you Docker image and build practice
- Docker Hub standard library provides a wealth of *Dockerfile* examples
- The `docker image ls` command supports filtering and formatting much like the `docker container ls` command
- Images can loaded from and saved to tar files
- Containers can be exported to tar files
- Tar files can be imported to create new containers

Dockerfile Documentation:
https://docs.docker.com/reference/builder/

Lab 4

- Advanced Dockerfiles