# Docker: Containers in Practice

Václav Pavlín
Tomáš Tomeček

# Docker - Basic Terms

- Registry
- Image
- Container
- Dockerfile
- `[registry/][namespace/]image_name[:tag]`
  - registry.example.com/vasek/fedora:21

# Docker - Architecture

- libcontainer
- Docker daemon
  - REST API + socket
- Graph - /var/lib/docker
  - devicemapper, btrfs, UnionFS, OverlayFS… [1]
- Docker CLI, docker-py...

# Creating Image

- You have to choose base image
- From Dockerfile
- Interactively:
  - `docker run -ti –name openalt-c fedora /bin/bash`
    - `echo Hello OpenAlt\! >/tmp/openme`
  - `docker commit openalt-c openalt-image`
  - `docker run openalt-image cat /tmp/openme`

# Dockerfile

```
FROM fedora:latest
MAINTAINER Tomas Tomecek <ttomecek@redhat.com>
RUN echo Hello OpenAlt\! >/tmp/openme
CMD cat /tmp/openme
```

# Dockerfile — followup

```
FROM fedora:latest
MAINTAINER Tomas Tomecek <ttomecek@redhat.com>
RUN echo Hello OpenAlt\! >/tmp/openme
ADD initial_data /var/app/
VOLUME /var/lib/data
EXPOSE 8000
ENV PYTHONPATH /var/app/src/
CMD cat /tmp/openme
```

# Dockerizing apps

- one service per container
- data volumes
- link containers together

# Registry

- Defined REST API
- Stores images and graph
- Way to transport images
- Different implementations

```
docker tag openalt-image localhost:5000/openalt-image
docker push localhost:5000/openalt-image
docker pull localhost:5000/openalt-image
```

# Deploying

1. Pull application (or source code and build it)
2. Build image
3. Push image to registry
4. Pull image from registry
5. Stop container
6. Launch container from newly built image

# Deploying

- ...or use Fig!
- define application layout
- and start it with `fig up`

```
web:
  build: ./app/
  links:
   - db
  ports:
   - "3031:3031"
db:
  build: ./db/
  ports:
   - "5432:5432"
```

# Monitoring

- logging
- docker top
- cockpit
- google-cadvisor
- htop
- systemd-cgls

# Monitoring — systemd-cgls

```
├─1 /usr/lib/systemd/systemd –system –deserialize 13
└─system.slice
  └─docker-fcaab5d4222b586e9c807f27988e3ecf43fe18aed2a34628.scope
    ├─12191 /usr/bin/python /usr/bin/supervisord -n
    ├─12212 /usr/bin/postgres -D /var/lib/pgsql/data -p 5432
    ├─12226 postgres: logger process
    ├─12229 postgres: checkpointer process
    ├─12230 postgres: writer process
    ├─12231 postgres: wal writer process
    ├─12232 postgres: autovacuum launcher process
    └─12233 postgres: stats collector process
```

# Tips and Tricks

- `source ~/.bashrc` (`bash-4.3$ `)
- updating packages may be a good idea
- remove metadata of package manager
- shell completion (zsh)

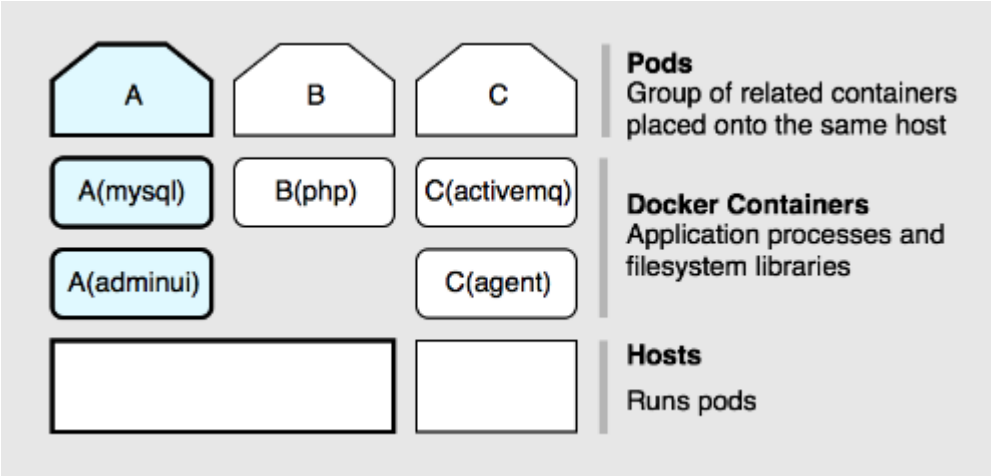
```
$ docker rm <tab>
containers
ad37a806f0f0 — 2d ago, fedora-python
5605671e97da — 6h ago, busybox
```

# Tips and Tricks

- write Dockerfile and execute commands interactively at the same time
- ~~ssh~~ nsenter, nsinit
- /var/lib/docker/{btrfs,devicemapper}
- mount code directly into container during development

# Kubernetes - Primitives

- Master - Minion
- Pod, Service, …
- [OpenShift V3 Deep Dive Tutorial]



# Openshift Origin v3

- Docker + Kubernetes
- Dockerfile vs. STI
- How to try

# References


1. http://developerblog.redhat.com/2014/09/30/overview-storage-scalability-docker/
2. https://github.com/openshift/origin
3. https://blog.openshift.com/openshift-v3-deep-dive-docker-kubernetes/