Workshop

Docker

For Software Developers

Rainer Stropek

software architects gmbh

Web	http://www.timecockpit.com
Mail	rainer@timecockpit.com
Twitter	@rstropek

time cockpit
Saves the day.

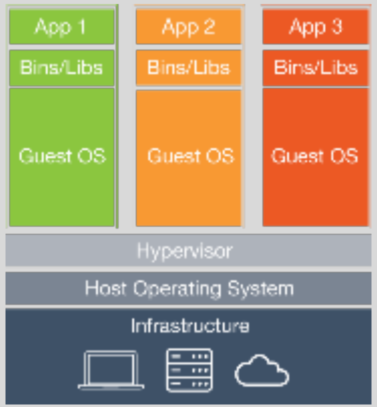
Your Host

Rainer Stropek

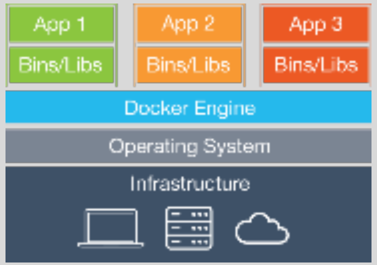
Developer, Entrepreneur
Azure MVP, MS Regional Director
IT-Visions

Contact

software architects gmbh
rainer@timecockpit.com
Twitter: @rstropek



Virtual Machines



Docker Container

What is Docker?

Virtual machines vs. Docker

- Each VM runs its own guest operating system
- Container reuse the host operating system
  - Container run in user space

https://www.docker.com/whatisdocker/

What’s Docker?

Container virtualization

Container run in user space and use kernel of host

Has been existing in Linux for quite a while

Docker builds on Linux Containers (LXC) and makes it easy to use and consume

Advantages?

Fast, small, and agile (e.g. Docker in Docker)

Disadvantages?

Security (less isolated)

What’s Docker?

Command line tool, REST services

Docker client can manage remote Docker daemon

Container packaging format

Dockerfiles for image creation from source code

Version management for images

Images can be based on images

Docker Hub: Platform to exchange images and Dockerfiles

Publishing on Docker Hub is not in scope of this talk

Docker in Windows

Docker Toolbox

Docker environment for Windows and Mac incl. VirtualBox

Container virtualization in Windows

Announced for next version of Windows Server

Windows Containers Quick Start

Use Azure to play with Docker

Existing VM image (Docker on Ubuntu server) in Azure marketplace

Use Docker container to run Azure tools (e.g. https://hub.docker.com/r/microsoft/azure-cli/)

Demo

Docker in Azure

Ubuntu server with Docker in Microsoft Azure

- Azure Docker Extension

ARM Template

- https://github.com/rstropek/DockerVS2015Intro/tree/master/dockerDemos/00-AzureARM

Access Docker Remotely

Default: Docker runs on non-networked Unix socket

TCP socket can be enabled (see Docker docs)

Docker available on the network → enable TLS

Docker docs

Remote Docker

```
// Connect to Docker client in Azure
// (see also https://github.com/rstropek/DockerVS2015Intro)

// Set environment variable (secure by default)
export DOCKER_HOST=tcp://dockertraining
  .northeurope.cloudapp.azure.com:2376 DOCKER_TLS_VERIFY=1
docker info
docker ps
```

Container

Working with containers

Docker CLI

Documentation

http://docs.docker.com/reference/commandline/cli

Important Commands for Containers

`docker run` – Run a command in a new container
`docker ps` – List containers
`docker start/stop` – Restarts/stops a container
`docker rm` – Removes container(s)
`docker attach` – Attach to running container
`docker top` – Display processes running in container
`docker exec` – Run a command in a container

Docker CLI

Starting Containers

```
docker run
  --name helloDocker -i -t ubuntu /bin/bash
  │                  │  │  │      └── Command to execute
  │                  │  │  └── Image name
  │                  │  └── Allocate pseudo-tty
  │                  └── Keep STDIN open
  └── Name of the container
```

```
docker run --name …
 -d ubuntu /bin/bash -c "while true; do echo hi; done"
 │         └── Command to execute (with arguments)
 └── Detach the container to the background (daemonized)
```

Interactive container

Daemonized container

Running in the background

`--rm` removes container when it exits

Demo

Interactive Container

```
# Check if docker is running
docker info

# Start interactive container
docker run -it ubuntu /bin/bash
  echo Hello > hello.txt
  exit

# List containers
docker ps
docker ps –a
docker ps --no-trunc -aq

# Restart container
docker start …

# Attach to container
docker attach …

# Remove container
docker rm …
# Remove all containers
docker rm `docker ps --no-trunc -aq`
```

Demo

Daemonized Container

```
# Start demonized container and get logs
docker run -d ubuntu /bin/bash \
  -c "while true; do echo hello world; sleep 1; done"

# Get the logs (-f for continuous monitoring)
docker logs …

# Check the processes in docker container
docker top …

# Open interactive shell in running container
docker exec -it … /bin/bash

# Inspect the details of a running container
docker inspect …


# WINDOWS
docker run -it windowsservercore cmd

docker build –t myweb .
docker run
```

Networking

Docker Networking

Networks

By default, three networks

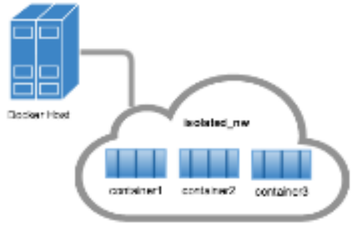
none, *host*, *bridge* (default)

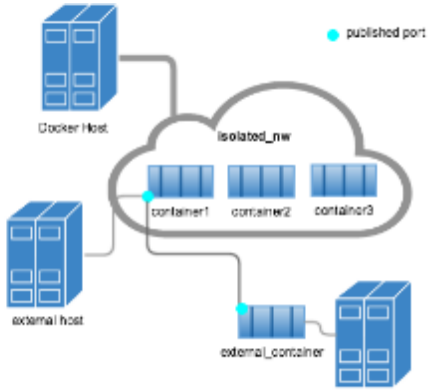
Additional networks can be created

Bridge network = single host

Overlay network (advanced topic, see [Docker docs]()) can include multiple hosts

Network isolation





For details see https://docs.docker.com/engine/userguide/networking/dockernetworks/

Networks

For details about network security, see [Docker docs]

```
# List all networks
docker network ls

# Inspect network details
docker network inspect bridge

# Disconnect a container from network
docker network disconnect bridge mycontainer
```

Container name

Network name

```
# Connect a container to a network
docker network connect mynetwork mycontainer

# Create own network
docker network create -d bridge mynetwork
```

Network name

Driver name

```
# Start container in a specific network
docker run -it --net=mynetwork ubuntu
```

DNS

Docker daemon contains embedded DNS server

```
# Start nginx web server on a custom network
docker run -d --net mynetwork --name web nginx
```

└─ Container name in DNS

```
# Start Ubuntu client in same network
docker run -it --net mynetwork --name client ubuntu

  # Ping web server
  ping web

  # Install curl and access web server
  apt-get install curl
  curl web

# Start Ubuntu container and link it using alias
docker run -it --net mynetwork --link=server3:nginx ubuntu
```

└─ Container-specific link

For details see
https://docs.docker.com/engine/userguide/networking/configure-dns/

```
docker run -d --net bridge -p 8080:80 nginx
```

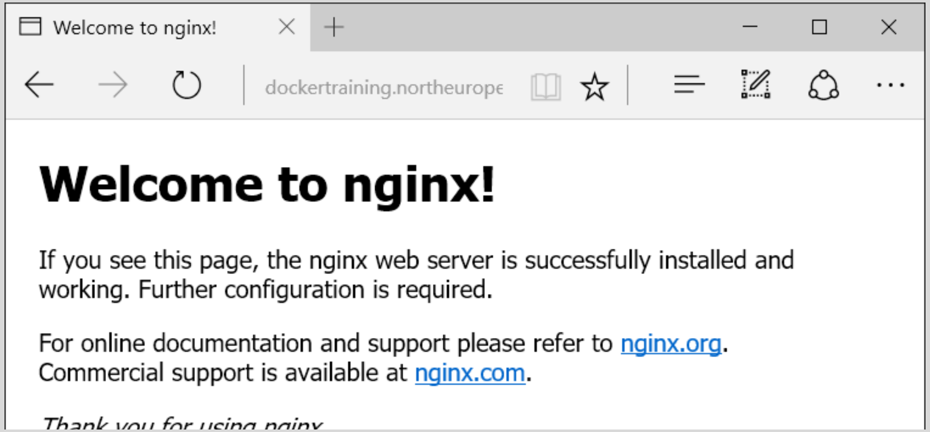
Host port

Container port

```
# Start nginx web server on host network
docker run -d --net host nginx
```

Assign container to *host* network

```
# Nginx is now available on the public internet:
```



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

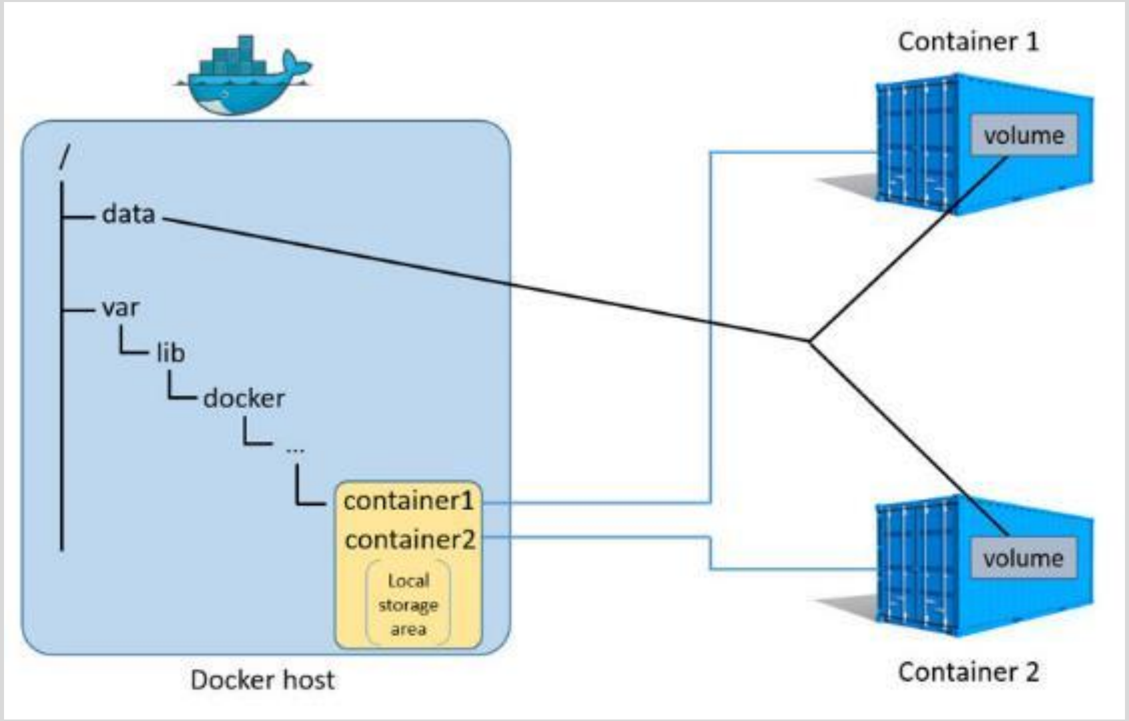
Binding container ports to host

Port mapping

EXPOSE in Dockerfiles

See Docker docs

Use *host* network



Data Volumes

Directory or file in the Docker host’s filesystem that is mounted directly into a container

Details see Docker docs

Source:
https://docs.docker.com/engine/userguide/storagedriver/imagesandcontainers/

Mount Host

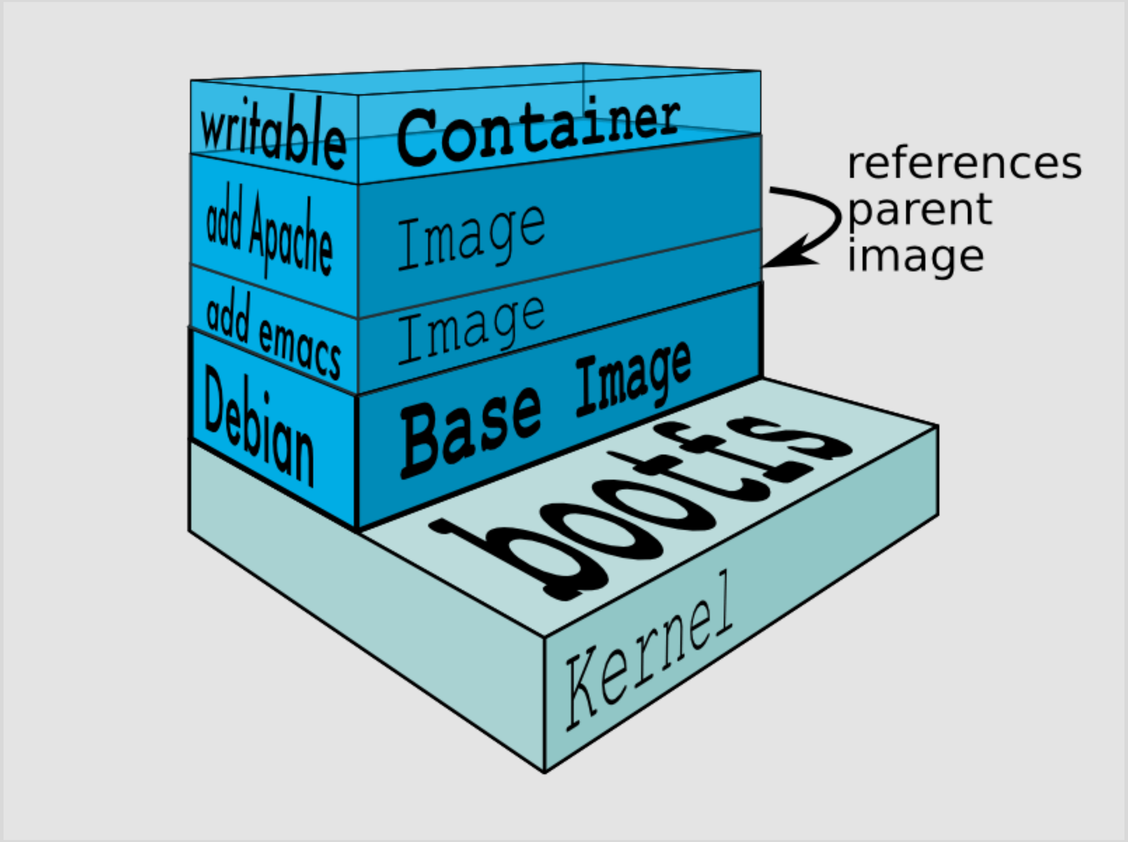
```
# Run postgres in a new container
docker run --name mydb -e POSTGRES_PASSWORD=P@ssw0rd!
  -d postgres

# Run client and execute some SQL
docker run -it --link mydb --rm postgres /bin/bash
  psql -h mydb -p 5432 -U postgres

  # Execute some SQL (e.g. create and fill a table)
  CREATE TABLE Test (ID INT PRIMARY KEY);
  INSERT INTO Test VALUES (1);
  SELECT * FROM Test;
  \q

# Delete container --> data is gone
docker rm –f mydb

-------------------

# Create data directory on host
mkdir dbdata

# Repeat the same example but this time with volume mapping
docker run --name mydb -e POSTGRES_PASSWORD=P@ssw0rd!
  -v ~/dbdata:/var/lib/postgresql/data -d postgres
```

Data Volume Container

```
# Create data volume container
docker create -v /dbdata --name dbstore postgres /bin/true
docker ps –a

# Create postgres container and mount data volume container
docker run --name mydb -e POSTGRES_PASSWORD=P@ssw0rd!
  -e PGDATA=/dbdata --volumes-from dbstore -d postgres

# Run client and execute some SQL (see previous example)
# Remove postgres container, recreate it --> data still there

# Start container to backup data
mkdir backup
docker run --rm --volumes-from dbstore
  -v ~/backup:/backup ubuntu tar cvf /backup/backup.tar /dbdata
ls –la backup/
```

Images

Working with images



File System Layers

Rootfs stays read-only

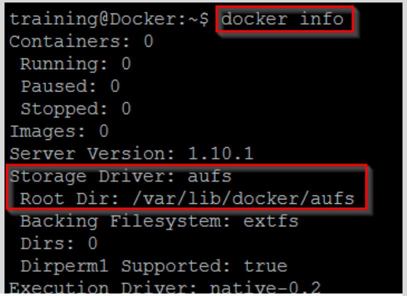
Union-mount file system
- over the read-only file system
  - Multiple file systems stacked on top of each other

Only top-most file system is writable
- Copy-on-write

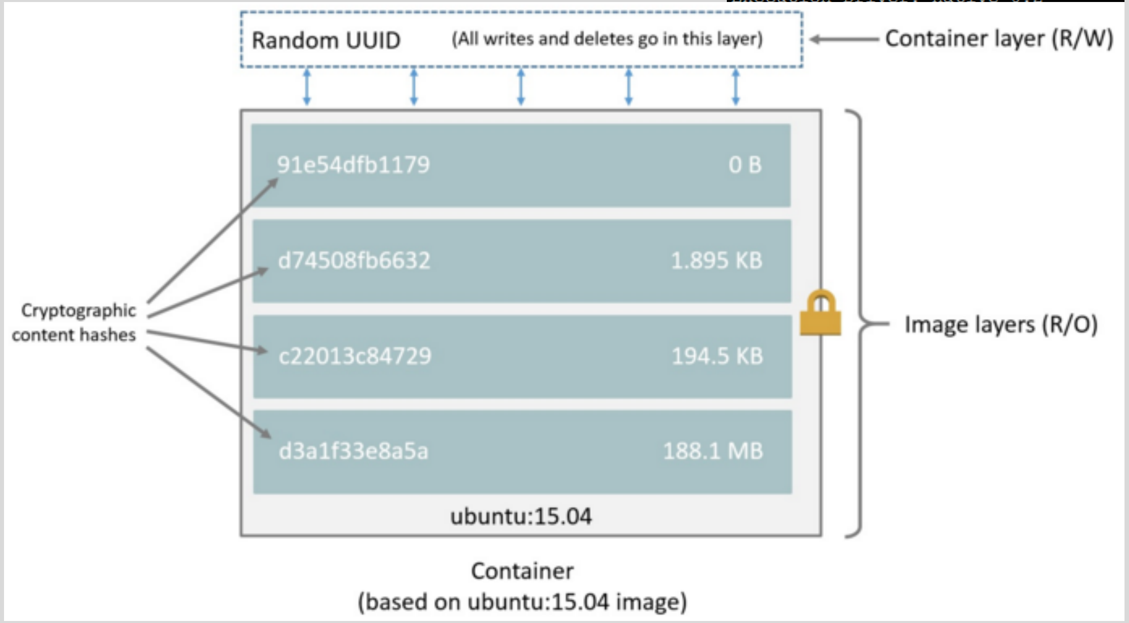
Images

```
# Pull image from docker hub
docker pull ubuntu

# Look for image directories on disk
ls /var/lib/docker/aufs/layers
```

└── Docker data directory



```
training@Docker:~$ docker info
Containers: 0
 Running: 0
 Paused: 0
 Stopped: 0
Images: 0
Server Version: 1.10.1
Storage Driver: aufs
 Root Dir: /var/lib/docker/aufs
 Backing Filesystem: extfs
 Dirs: 0
 Dirperm1 Supported: true
Execution Driver: native-0.2
```



More about storage drivers
see [Docker docs](https://docs.docker.com/engine/userguide/storagedriver/imagesandcontainers/)

Source:
https://docs.docker.com/engine/userguide/storagedriver/imagesandcontainers/

Docker CLI

Important Commands for Images

`docker images` – List all images
`docker search` – Search for image on Docker Hub
`docker pull` – Pulls an image from the registry (Docker Hub)
`docker commit` – Create image from container
`docker inspect` – Get low-level information on container or image

Docker CLI

Building Images from Containers

```
docker commit
  -m="Demo image" --author="Rainer Stropek"
```

Message

Author of the image

```
  templateContainer rstropek/ubuntu:withFile
```

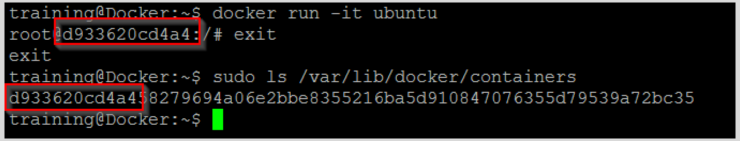
Target repository:tag

Name of the container

Demo

Create Image

```
# Start interactive container
docker run -it ubuntu /bin/bash
  echo "Hello Docker" > helloWorld.txt
  exit
```



```
training@Docker:~$ docker run -it ubuntu
root@d933620cd4a4:/# exit
exit
training@Docker:~$ sudo ls /var/lib/docker/containers
d933620cd4a458279694a06e2bbe8355216ba5d910847076355d79539a72bc35
training@Docker:~$
```

```
# Build image from container
docker commit … rainer:withFile

# Remove container
docker rm -f …

# Create new container from new image
docker run -it rainer:withFile /bin/bash
# View history of image
Docker history rainer:withFile

# Remove image
docker rmi rainer:withfile

# Run DockerUI in container
# https://github.com/crosbymichael/dockerui
docker run -d -p 9000:9000 --privileged \
  -v /var/run/docker.sock:/var/run/docker.sock \
  dockerui/dockerui
```

Dockerfiles

Creating images from source

```
# Version 0.0.1
FROM nginx
MAINTAINER Rainer Stropek "rainer@timecockpit.com"
ENV REFRESHED_AT 2014-02-22
RUN apt-get -qq update
```

Execute command in new layer on top of the image and commit the result

```
COPY *.html /usr/share/nginx/html/
```

Copy files to the filesystem of the container

```
docker build -t staticweb .
```

Dockerfile location

Tag for the image

Dockerfiles

Documentation

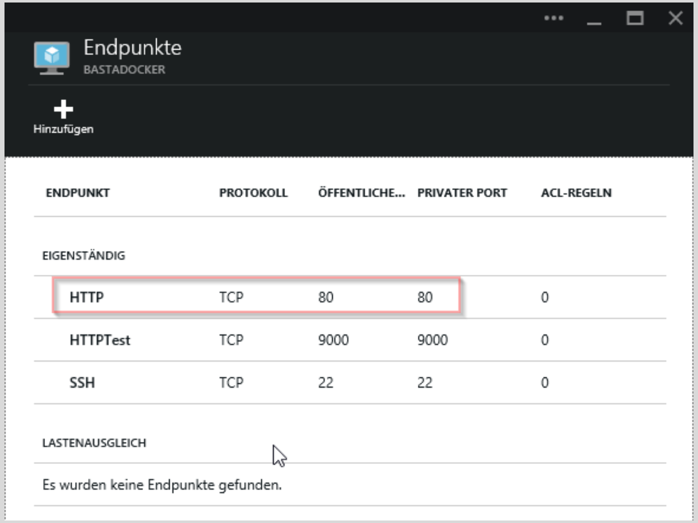
https://docs.docker.com/reference/builder/
https://registry.hub.docker.com/_/nginx/

Docker CLI

Exposing ports

```
docker run --name staticwebcontainer \
  -d -p 80:80 staticweb
```

- `-p`: Expose port 80
- `-d`: Run daemonized



Endpunkte
BASTADOCKER

Hinzufügen

ENDPUNKT	PROTOKOLL	ÖFFENTLICHE...	PRIVATER PORT	ACL-REGELN
EIGENSTÄNDIG				
HTTP	TCP	80	80	0
HTTPTest	TCP	9000	9000	0
SSH	TCP	22	22	0
LASTENAUSGLEICH				

Es wurden keine Endpunkte gefunden.

```
# Get sample code from GitHub
git clone https://github.com/rstropek/DockerVS2015Intro.git

# Build website
cd dockerDemos/01-staticWeb/app
npm install
grunt
cd ..

# Build image from Dockerfile
docker build -t staticweb .
docker run -d -p 80:80 staticweb

# Change website content and rebuild container

# Run a second container, run a third container (linked)
docker run -i -t --link <cont1>:sweb1 --link <cont2>:sweb2
ubuntu /bin/bash
  apt-get install curl
  curl http://sweb1
```

Demo

Dockerfile

Sample files see
https://github.com/rstropek/DockerVS2015Intro/tree/master/dockerDemos/01-staticWeb

Demo

Automated build

```
# Run grunt inside a docker container
docker run --rm -v ~/DockerVS2015Intro/dockerDemos/01-
staticWeb/app:/data digitallyseamless/nodejs-bower-grunt grunt
```

```
# Run daemonized grunt inside a docker container
docker run -d -v ~/DockerVS2015Intro/dockerDemos/01-
staticWeb/app:/data digitallyseamless/nodejs-bower-grunt grunt
watch
```

```
# Run nginx webserver inside daemonized container
docker run -d -p 80:80 -v ~/DockerVS2015Intro/dockerDemos/01-
staticWeb/app:/usr/share/nginx/html nginx
```

Demo

Run Grunt (build) in Container

```
# Run grunt inside a docker container
docker run --rm
```

└── Remove the container when it exists

```
  -v ~/DockerVS2015Intro/dockerDemos/01-staticWeb/app:/data
```

└── Mount host volume (`host:container`)

[`dockerfile/nodejs-bower-grunt`]

└── Use existing image

```
  grunt
```

└── Run `grunt`

Docker Compose

Tool for running multi-container applications

Demo

```
printer:
  build: .
         │
         └── Build local Dockerfile

  links:
   - dependent-service
     │
     └── Link to other containers (e.g. Redis, MongoDB)

dependent-service:
  image: dependent-service
         │
         └── Run service container depends on based on
             an existing image
```

For more info visit https://docs.docker.com/compose/

```
# Build dependent service
# directory: ~/DockerVS2015Intro/dockerDemos/02-compose/dependentService
npm install
docker build –t dependent-service .

# Run container using dependent service
# directory: ~/DockerVS2015Intro/dockerDemos/02-compose
npm install
docker-compose run printer
```

Demo

Automated build

Sample files see
https://github.com/rstropek/DockerVS2015Intro/tree/master/dockerDemos/02-compose

ASP.NET in Docker

Running ASP.NET in Docker

```
FROM microsoft/aspnet

RUN apt-get install -y curl
RUN curl -sL https://deb.nodesource.com/setup_5.x | bash -
RUN apt-get install -y nodejs

COPY ./my-web /src

RUN cd /src && dnu restore

EXPOSE 5000

WORKDIR /src
CMD ["dnx", "web"]
```

Simple ASP.NET

Dockerfile

Sample files see
https://github.com/rstropek/DockerVS2015Intro/tree/master/dockerDemos/04-aspnet

Simple ASP.NET

```
# Generate an ASP.NET web app
yo aspnet webbasic "my-web"

# Add “--server.urls=http://*:5000/“ to project.json so
# that ASP.NET listens not only on localhost

# Build image with sample app
docker build -t rainer:myweb .

# Run ASP.NET container
docker run -d -p 80:5000 rainer:myweb
```

Sample files see
https://github.com/rstropek/DockerVS2015Intro/tree/master/dockerDemos/04-aspnet

```
FROM microsoft/aspnet
MAINTAINER Rainer Stropek "rainer@timecockpit.com"
ENV REFRESHED_AT 2015-01-02

ENV SOURCE_DIR /app/src

RUN mkdir -p $SOURCE_DIR
WORKDIR $SOURCE_DIR

COPY refreshAndRunSample.sh $SOURCE_DIR/
RUN chmod a+x $SOURCE_DIR/refreshAndRunSample.sh

RUN apt-get -qqy install git
RUN git init \
 && git pull https://github.com/aspnet/Home.git \
 && cd  samples/HelloMvc/ \
 && kpm restore

ENTRYPOINT ["/app/src/refreshAndRunSample.sh"]
```

Dockerfile

Base image:
https://registry.hub.docker.com/u/microsoft/aspnet/

Run container
```
docker run -d -t
  -p 80:5004 aspnet-beta8
```

Application Scenarios

Running continuous integration in containers

Rebuild complex runtime environment on my laptop

Identical environment for dev, test, and prod

Cost reduction in the cloud

High density hosting (e.g. multiple versions)

Split software into multiple, independent services

Micro-services, see Manfred's session tomorrow