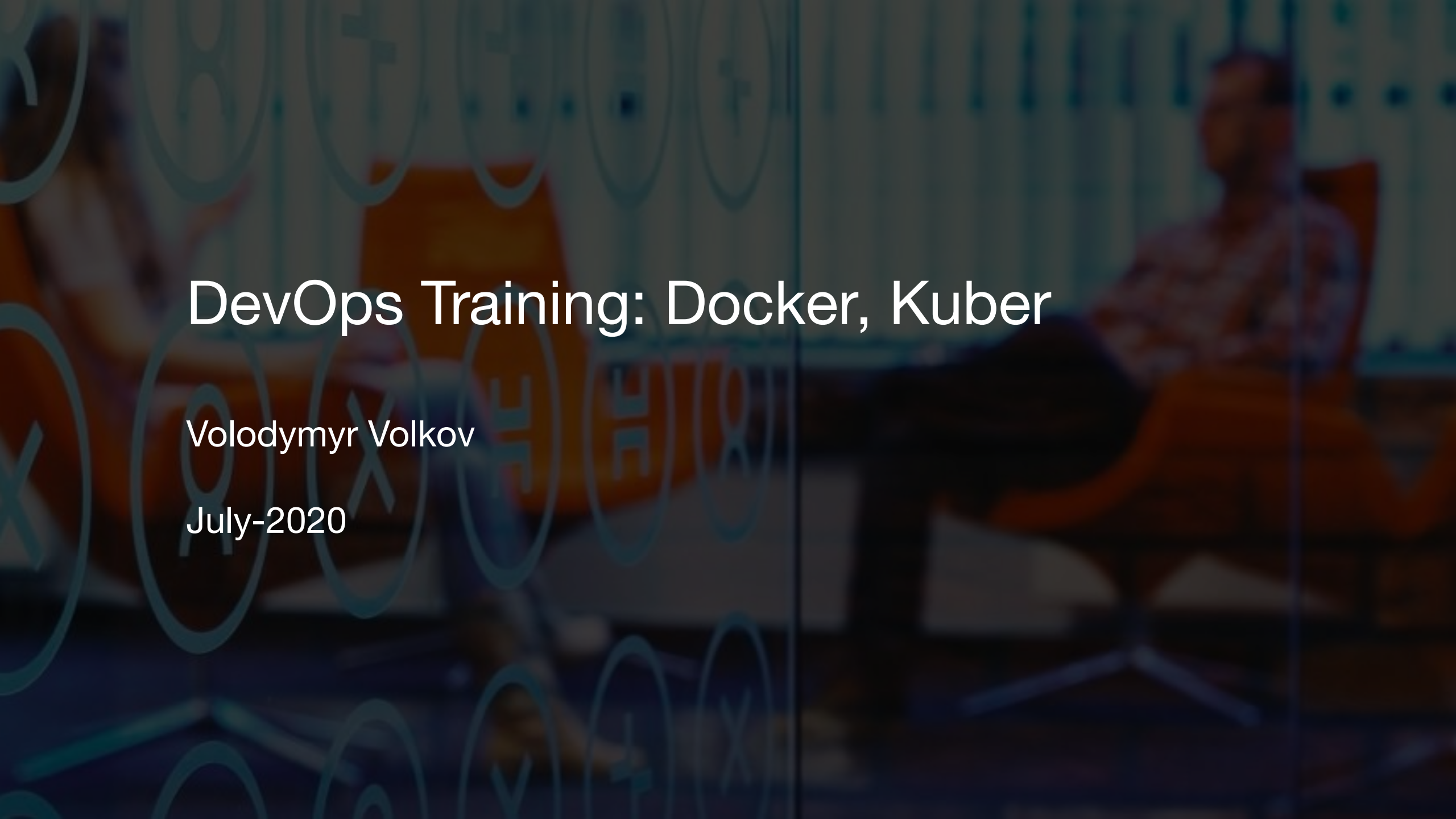


DevOps Training: Docker, Kuber

Volodymyr Volkov

July-2020

My name is Vova, and I`m a kubernetes admin ...

Practice Requirements

- AWS EC2
  - Frankfurt
  - Ubuntu Server 18.04 LTS (HVM)
  - t2.micro 1 instance
  - 1 Public IP
  - Security:
    - ssh from your public IP
    - 8080 http (tcp) from your public IP
  - Install Docker: snap install docker
  - *JFYI: ssh user: ubuntu, to become rootuse: “sudo su -”*

Lection 1: Container - What Are You?

Prepare: AWS, Frankfurt, Ubuntu 18.04 LTS, Docker: snap install docker; Public: ssh, 8080(tcp)

OS Level User Process Isolation

Became meaningful on multitasking introduction.

Initial low-level understanding:

Process isolation is a set of different hardware and software technologies[1] designed to protect each process from other processes on the operating system.

...

Security is easier to enforce by disallowing inter-process memory access, in contrast with less secure architectures such as DOS in which any process can write to any memory in any other process.

WIKI

Prepare: AWS, Frankfurt, Ubuntu 18.04 LTS, Docker: snap install docker; Public: ssh, 8080(tcp)

Unix/Linux Resource Isolation Tools History

1979 Unix
1982 BSD
(later on Linux - now not yet invented:)

Chroot

Substituting root point in file tree for process

2000 FreeBSD

Jails

2001 Linux

vServer

2004 Solaris

Containers

clear-cut separation between its services in “jails” own IP for each jail (processes in it)

Jail + resource partitioning for file systems, network addresses, memory

resource controls and boundary separation provided by zones snapshots, cloning from ZFS

2005 Linux

VPS buttle with KVM, Xen, VMWare

OpenVZ

Operating system-level virtualization: isolation, resource management (IPCs, devices) and checkpointing

2006 Linux

CGroup
(Control Group by Google)

limiting, accounting and isolating resource usage (CPU, memory, disk I/O, network) of a collection of processes

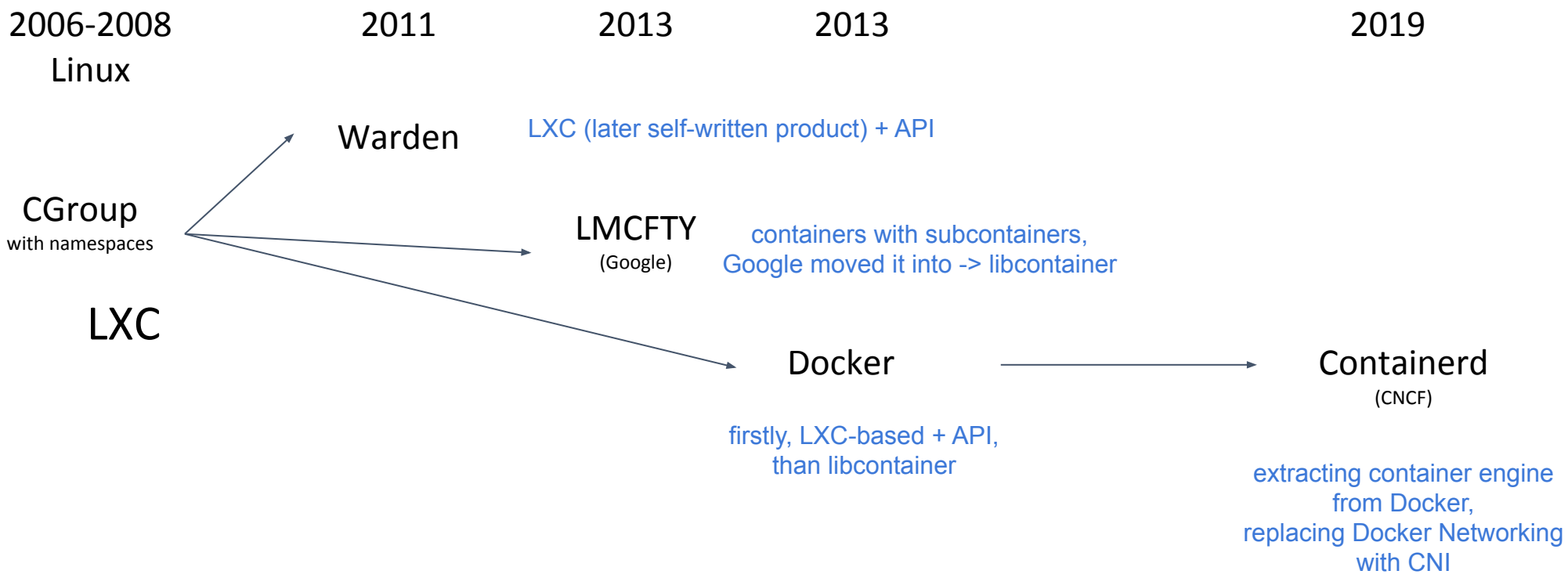
2008 Linux

LXC

\+ namespaces
\+ (1 more level of abstraction)

Prepare: AWS, Frankfurt, Ubuntu 18.04 LTS, Docker: snap install docker; Public: ssh, 8080(tcp)

Unix/Linux Resource Segregation ToolsHistory



Prepare: AWS, Frankfurt, Ubuntu 18.04 LTS, Docker: snap install docker; Public: ssh, 8080(tcp)

Chroot is The Father of Containers

Prepare: AWS, Frankfurt, Ubuntu 18.04 TLS, Docker: snap install docker; Public: ssh, 8080(tcp)

Docker: Beginnig

Hands On: Docker Run, Docker ps

```
# docker run centos echo "hello world"
hello world

# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES

# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS    NAMES
08c65bc171c3        centos              "echo 'hello world'"     4 minutes ago       Exited (0) 4 minutes ago             boring_wiles

# docker ps -as
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS    NAMES    SIZE
08c65bc171c3        centos              "echo 'hello world'"     4 minutes ago       Exited (0) 4 minutes ago             boring_wiles  0B (virtual
220MB)
```

If have created more than one - remove other by executing “docker rm” following by removing docker IDs:

```
# docker rm 1fcee9605349 08c65bc171c3
1fcee9605349
08c65bc171c3
```

Hands On: Docker start, image

```
# docker start 08c65bc171c3
08c65bc171c3

# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES

# docker logs -f 08c65bc171c3
hello world
hello world

# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              latest              0f3e07c0138f        4 weeks ago         220MB

# docker image rm 0f3e07c0138f
Error response from daemon: conflict: unable to delete 0f3e07c0138f (must be forced) - image is being used by stopped container 08c65bc171c3
```

Hands on: Key Points

- If process(es) executed in Docker container are finished - docker container stopped.
- Stopped docker containers are not removed automatically - keeping tying Docker container Resources (image, logs, volumes etc.)
- So docker container could be started again referenced by docker ID or container name!

Hands On: -it, -d, exec

```
# docker run centos /bin/bash
# docker run -it centos /bin/bash
[root@b504891d0e11 /]# yum list rpm
…
[root@b504891d0e11 /]# exit
exit
# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES

# docker run -d centos /bin/bash
#
b504891d0e114152980bb3dc300f6110f8860b083f8b7d32ecfaca95859ded91
# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
b504891d0e11        centos              "sleep 1200"        9 minutes ago       Up 9 minutes                            nifty_sammet

# docker exec -it b504891d0e11 /bin/bash
[root@b504891d0e11 /]# ps -aux
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.1  23024  1380 ?        Ss   11:28   0:00 /usr/bin/coreutils --coreutils-prog-shebang=sleep /usr/bin/sleep 1200
root         21  6.3  0.3  12028  3224 pts/0    Ss   11:37   0:00 /bin/bash
root         34  0.0  0.3  46340  3248 pts/0    R+   11:37   0:00 ps -aux
```

Hands On Key Points

Containers:

- Containers are made to run application(s) inside them. No app running - container stopping.
- Containers allow to start on same host in different containers code with unexpected or conflicting dependencies
- What has happened in container stays in container.

Docker:

- docker simplifies log handling: just redirect all your app logs to STDOUT (standard output) - dockerd catches this and stored as log for this container

Linux Namespaces

Namespace - it`s context separation of resource management.

Now Linux kernel support 7 such types of separated contexts:

- Cgroups, IPC, Network, Mount, PID, User, UTS

Visualize namespaces for some process:

```
# ls -l /proc/2068/ns
total 0
lrwxrwxrwx 1 root root 0 Nov  2 23:15 cgroup -> 'cgroup:[4026531835]'
lrwxrwxrwx 1 root root 0 Nov  2 23:15 ipc -> 'ipc:[4026532229]'
lrwxrwxrwx 1 root root 0 Nov  2 23:15 mnt -> 'mnt:[4026532227]'
lrwxrwxrwx 1 root root 0 Nov  2 23:11 net -> 'net:[4026532232]'
lrwxrwxrwx 1 root root 0 Nov  2 23:15 pid -> 'pid:[4026532230]'
lrwxrwxrwx 1 root root 0 Nov  2 23:15 pid_for_children -> 'pid:[4026532230]'
lrwxrwxrwx 1 root root 0 Nov  2 23:15 user -> 'user:[4026531837]'
lrwxrwxrwx 1 root root 0 Nov  2 23:15 uts -> 'uts:[4026532228]'
```

Create namespace forresource: unshare -u <binary> (u - UTS)

Docker Processes From Outside

Hipster Docker:

```
# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
3cde514a5a0a        nginx               "nginx -g 'daemon of…"   3 minutes ago       Up 3 minutes        80/tcp              xenodochial_curie
31eab20249db        centos              "sleep 1200"             40 minutes ago      Up 40 minutes                           stupefied_bohr

# ps -ax --forest
...
    1 ?        Ss     0:03 /sbin/init
...
 2129 ?        Ssl    0:25 dockerd -G docker --exec-root=/var/snap/docker/384/run/docker --data-root=/var/snap/docker/common/var-lib-docker --pidfile=/var/snap
 2205 ?        Ssl    0:06  \_ docker-containerd --config /var/snap/docker/384/run/docker/containerd/containerd.toml
 8008 ?        Sl     0:00      \_ docker-containerd-shim -namespace moby -workdir /var/snap/docker/common/var-lib-docker/containerd/daemon/io.containerd.runti
 8030 ?        Ss     0:00      |   \_ /usr/bin/coreutils --coreutils-prog-shebang=sleep /usr/bin/sleep 1200
 9925 pts/0    Ss+    0:00      |   \_ /bin/bash
 9658 ?        Sl     0:00      \_ docker-containerd-shim -namespace moby -workdir /var/snap/docker/common/var-lib-docker/containerd/daemon/io.containerd.runti
 9685 ?        Ss     0:00          \_ nginx: master process nginx -g daemon off;
 9723 ?        S      0:00              \_ nginx: worker process
```

Docker versus LXContainer

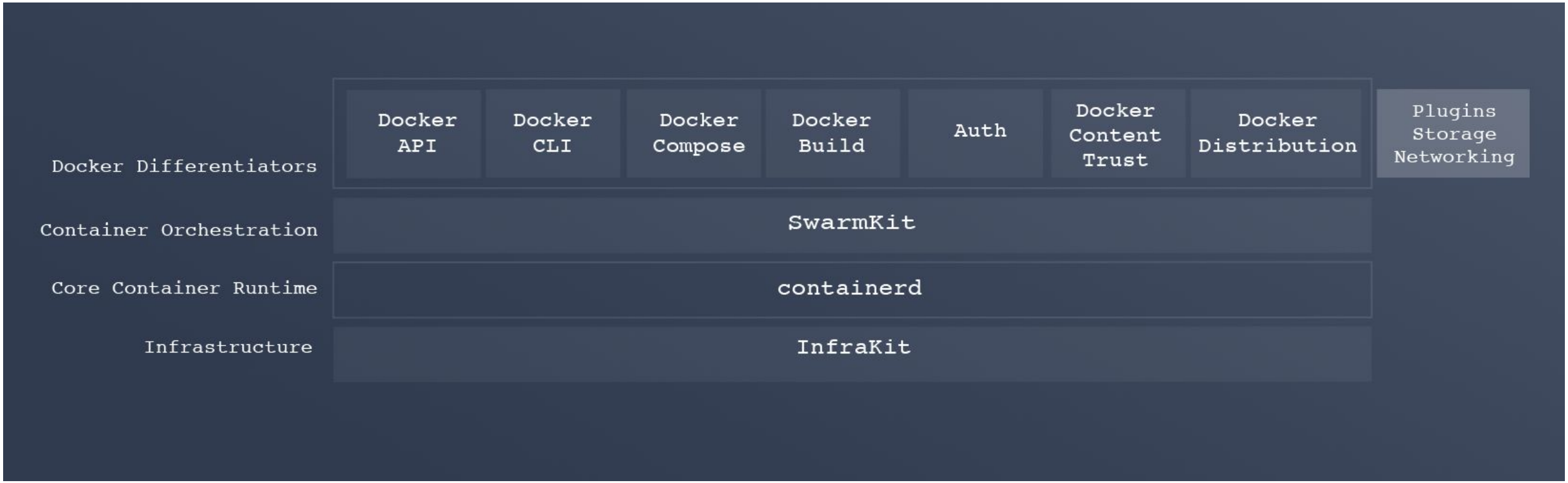
Hipster Docker:

```
    1 ?        Ss     0:03 /sbin/init
...
 2129 ?        Ssl    0:24 dockerd -G docker --exec-root=/var/snap/docker/384/run/docker --data-root=/var/snap/docker/common/var-lib-docker --pidfile=/var/snap
 2205 ?        Ssl    0:06  \_ docker-containerd --config /var/snap/docker/384/run/docker/containerd/containerd.toml
 9658 ?        Sl     0:00      \_ docker-containerd-shim -namespace moby -workdir /var/snap/docker/common/var-lib-docker/containerd/daemon/io.containerd.runti
 9685 ?        Ss     0:00          \_  nginx: master process nginx -g daemon off;
 9723 ?        S      0:00              \_  nginx: worker process
```

True LXC:

```
    1 ?        Ss     0:03 /sbin/init
...
 5495 ?        Ss     0:00  [lxc monitor] /var/lib/lxc nginx
 5512 ?        Ss     0:00  \_  /sbin/init
 5571 ?        S<s    0:00      \_ /lib/systemd/systemd-journald
 5576 ?        Ss     0:00      \_ /lib/systemd/systemd-networkd
 5605 ?        Ss     0:00      \_ /lib/systemd/systemd-resolved
 5606 ?        Ss     0:00      \_ /lib/systemd/systemd-logind
 5607 ?        Ssl    0:00      \_ /usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup-triggers
 5608 ?        Ss     0:00      \_ /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
 5609 ?        Ssl    0:00      \_ /usr/sbin/rsyslogd -n
 5610 ?        Ss     0:00      \_ /usr/sbin/cron -f
 5613 pts/8    Ss+    0:00      \_ /sbin/agetty -o -p -- \u --noclear --keep-baud console 115200,38400,9600 vt220
 5614 pts/0    Ss+    0:00      \_ /sbin/agetty -o -p -- \u --noclear --keep-baud pts/0 115200,38400,9600 vt220
 5615 pts/1    Ss+    0:00      \_ /sbin/agetty -o -p -- \u --noclear --keep-baud pts/1 115200,38400,9600 vt220
 5616 pts/2    Ss+    0:00      \_ /sbin/agetty -o -p -- \u --noclear --keep-baud pts/2 115200,38400,9600 vt220
 5617 pts/3    Ss+    0:00      \_ /sbin/agetty -o -p -- \u --noclear --keep-baud pts/3 115200,38400,9600 vt220
 5622 ?        Ss     0:00      \_  nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
 5623 ?        S      0:00          \_  nginx: worker process
 5624 ?        S      0:00          \_  nginx: worker process
 5625 ?        S      0:00          \_  nginx: worker process
 5626 ?        S      0:00          \_  nginx: worker process
```

Nowdays Docker Structure



Microservice Architecture Concept

What Mean Microservice

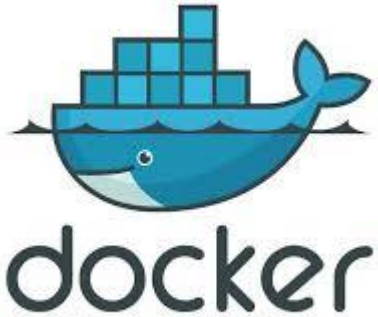
- Microservices are a software development technique —a variant of the service-oriented architecture (SOA) structural style— that arranges an application as a collection of loosely coupled services.[1] In a microservices architecture, services are fine-grained and the protocols are lightweight. [Wiki]

- For instance, Amazon's policy is that the team implementing a microservice should be small enough that they can be fed by two pizzas. [some more Wiki]

Microservice by Microservice.io

Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities
- Owned by a small team



The microservice architecture enables the rapid, frequent and reliable delivery of large, complex applications. It also enables an organization to evolve its technology stack.

Application Into Docker

Pushing App Into Containers

Ways how to put your app into container:

1. Take a look around - possibly someone already done this. Docker Hub.
2. Start container, add your code into it, commit. Docker image.
3. Build container with your code from scratch. Dockerfile.
4. If your app code is changed during execution OR/AND logic is not separated from data OR/AND you just don`t want to put it into container but should - use volumes.

1. Docker Hub

1. Official Docker Repo
2. Image could be both pulled and pushed to.
3. Free for some size.

To pull image:

```
# docker pull ubuntu:19.10
```

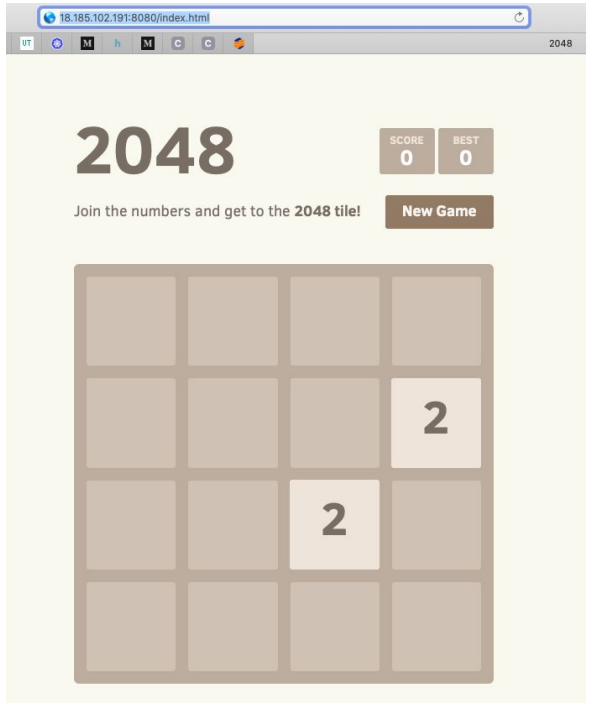
Running container from not pulled image automatically pulls it:

```
# docker run -d --name daydreaming_newton nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
8d691f585fa8: Pull complete
5b07f4e08ad0: Pull complete
abc291867bca: Pull complete
Digest: sha256:922c815aa4df050d4df476e92daed4231f466acc8ee90e0e774951b0fd7195a4
Status: Downloaded newer image for nginx:latest
b28340a80ba178ace4bcd59fa153a7fc149743a340d9cf19db543f8f220274b8
```

2. Hands On: Docker COPY, Commit

```
# docker run -d -p 8080:80 nginx
1fbe97d9c731.......

# git clone https://github.com/gabrielecirulli/2048.git
# cd 2048/; docker cp ./ 1fbe97d9c731:/usr/share/nginx/html
```

http://18.185.102.191:8080/index.html




```
# docker image ls
nginx          latest          2622e6cca7eb      10 days ago         132MB

# docker ps -s
#docker commit a45630804dc1
sha256:a53cd93bc1b89232c6ecf91eb50a22320fca5183e76df5453e8768148cee7e15

# docker image ls
REPOSITORY      TAG             IMAGE ID        CREATED             SIZE
<none>          <none>          a53cd93bc1b8    About a minute ago  133MB
nginx           latest          2622e6cca7eb    10 days ago         132MB
#docker stop 1fbe97; docker run -p 8080:80 -d a53cd93bc1b8
```

3. Hands On: Dockerfile

```
# mkdir docker; git clone https://github.com/gabrielecirulli/2048.git docker/2048; vim Dockerfile
```

```
FROM nginx
COPY 2048/ /usr/share/nginx/html/
```

```
~/docker# docker build ./ -t 2048game
Sending build context to Docker daemon  1.346MB
Step 1/2 : FROM nginx
 ---> 540a289bab6c
Step 2/2 : COPY 2048/ /usr/share/nginx/html/
 ---> 960c02a8cf80
Successfully built 960c02a8cf80
Successfully tagged 2048game:latest

~/docker# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
2048game            latest              0bc5c1e414d8        13 seconds ago      133MB
<none>              <none>              a53cd93bc1b8        14 minutes ago      133MB
nginx               latest              2622e6cca7eb        11 days ago         132MB

# docker run -p 8080:80 -d 0bc5c1e414d8
```

3. Docker Image Layers

~/docker# docker image ls

```
REPOSITORY      TAG          IMAGE ID       CREATED          SIZE
2048game        latest       cbc77a65d75a   13 seconds ago   133MB
<none>          <none>       05b3d60c717d   14 minutes ago   133MB
nginx           latest       2622e6cca7eb   11 days ago      132MB
```

~/docker# docker image inspect cbc77a65d75a

```
...
      "RootFS": {
         "Type": "layers",
         "Layers": [
            "sha256:13cb14c2acd3...",
            "sha256:d4cf327d8ef50...",
            "sha256:7c7d7f446182...",
            "sha256:9040af41bb66...",
            "sha256:f978b9ed3f26a...",
            "sha256:61fe62a4f2901.."
         ]
      },
...
```

~/docker# docker image inspect 05b3d60c717d

```
...
      "RootFS": {
         "Type": "layers",
         "Layers": [
            "sha256:13cb14c2acd3...",
            "sha256:d4cf327d8ef50...",
            "sha256:7c7d7f446182...",
            "sha256:9040af41bb66...",
            "sha256:f978b9ed3f26a...",
            "sha256:85fc12c04ec79..."
         ]
      },
```

 # docker ps -as

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES	SIZE
4c11769c2cf6	nginx	"/docker-entrypoint.…"	4 minutes ago	Exited (0) 8 seconds ago		thirsty_meitner	1.29MB (virtual 133MB)

Image: Layers, Dockerfile

Docker images are layered.
Hash of each layer includes files changes made before layer is finished and semi-hash from previous layers.

1 commit = 1 layer
1 line of Dockerfile = 1 Layer

```
FROM ubuntu:18.04
VOLUME /app
VOLUME /data
ENV TZ=Europe/Kiev
RUN apt-get update && apt-get install --no-install-recommends --no-install-suggests -y git python3 python3-pip python3-setuptools python3-dev python3-psycopg2
RUN pip3 install mysql-connector pyyaml
RUN pip3 install docker-py
RUN pip3 install psycopg2
COPY ./app/ /app/
CMD /app/cycle.sh
```

Put upper basic non frequently changed parts

Put at the end more frequently changed parts

4. Hands On: Docker Volumes

Volume in Docker is looking like mount -bind directory.

```
~# docker ps -as
...
~# mkdir -p registry-storage;
~# docker run -d -p 5000:5000 -v registry-storage:/var/lib/registry registry:2
dee2ac82f8ff9896987059f64f4a6dc25e5cbe998417f5ba2ff77f6d7f980b9e

~# docker volume ls
DRIVER          VOLUME NAME
local           412b07e4ecf7c735e128458b33c3dd16735c66d0a799dbee5dd1da211740aeb0
local           85cb4930feab7b2663b5846a87e0adcf05f6ca0763c42ce34fb77e5e2f52fafd
local           9e698b47f5a2e24514418514fdec4deb60cac5bf4433689209d87bc5a15ef4ca
local           registry-storage
```

If volume declared in Dockerfile and not mounted on start - Docker automatically creates volume on write access to declared Volume mount point.

```
FROM ubuntu:18.04
VOLUME /app
```

Volumes could be mounted from outside using drivers like NFS. And same volume could be mounted to more than on Docker container!

Hands On: Docker App Distributing, Tag, Registry

Tagging is advertised for images management

Docker Registry - your own Docker Hub.

```
~# docker ps | grep registry
dee2ac82f8ff       registry:2        "/entrypoint.sh /etc…"   2 minutes ago      Up 2 minutes       0.0.0.0:5000->5000/tcp   nervous_kare
```

Docker Tag, Push

```
~# docker tag a53cd93bc1b8 2048game:v01
~# docker tag a53cd93bc1b8 localhost:5000/2048game:v01
~# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
2048game            latest              0bc5c1e414d8        11 hours ago        126MB
2048game            v01                 a53cd93bc1b8        12 hours ago        128MB
localhost/2048game  v01                 a53cd93bc1b8        12 hours ago        128MB

~# docker push localhost:5000/2048game:v01
The push refers to repository [localhost:5000/2048game]
c64aa9c614dd: Pushed
a89b8f05da3a: Pushed
6eaad811af02: Pushing [==========================>                ]  29.77MB/56.98MB
b67d19e65ef6: Pushing [====================>                      ]  26.54MB/69.23MB
```

Hands on: Basic Docker Networking

Exposing a port (making it available - doesn`t mean forwarding is working)

```
FROM ubuntu:18.04
RUN apt-get update; apt-get install nginx
EXPOSE 80
```

Forwarding a port

```
#  docker run -d -p 8080:80 --name nginx nginx
c2fcf6b9017b47ffd45d774697ba350f23cc972065b911e8711a096569c196c1
# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                   NAMES
c2fcf6b9017b        nginx               "nginx -g 'daemon of…"   3 seconds ago       Up 2 seconds        0.0.0.0:8080->80/tcp    nginx
```

Available 3 types of Docker networking:

1) To docker default bridge (default behaviour, worked because Docker running DHCP)
2) Docker to physical interface
3) Docker without network (unmapped)

Docker Networking: iptables, bridging

```
~# brctl show docker0
bridge name  bridge id        STP enabled interfaces
docker0      8000.0242827baa10   no          vetheb31987

~# iptables -vnL -t nat
…

~# iptables -vnL
…
```

What Makes Docker in Containers a Xerox in Copy Machines

Out of the box:

- simple networking (automation of bridging, iptables*)
- Dockerfiles (from code management point of view)
- encapsulating code into images
- dockerd adoption of images on different systems
- cool layering of images
- containers distributing hub (global and local)
- volumes (shared folders)
- simplified logging.

Next Sections

Section 2. Docker: something from under the hood

- Dockerbuild file: more options, more pain.
- More than 1 App Achievements:
  - Environment Variables, Secrets; Volumes sharing;
  - Docker Link.
- Docker Networking;

Section 3. Kuber: beginning

- Microservice App Achievements
  - App Upstart Dependencies;
  - Service Discovery;
  - DNSing.
- Docker Compose.
- Docker Swarm.
- Kuber: Docker ambitions cutter.
- Container.d: Docker dissolver.

Howe Work 1

Home Task: https://github.com/ask4ua/DKN/blob/master/Hometask/Section1/README.md

Email: volodymyr.volkov@globallogic.com

Deadline: 1 week - Next Friday

Section 2: Docker: More From Under The Hood.

Practice Requirements

- AWS EC2
  - Frankfurt
  - Ubuntu Server 18.04 LTS (HVM)
  - t2.micro 1 instance
  - 1 Public IP
  - Security:
    - ssh from your public IP
    - 80,8080 http (tcp) from your public IP
  - Install Docker: snap install docker
  - *JFYI: ssh user: ubuntu*

Docker Networking

Type	Docker run Option	How it works	Peculiarities
Hosted	--net=host	Mapping all hypervisor interfaces into container (same network namespace from host referenced into container ns)	If container not priveleged - only could occupy free ports on iface. If privileged - all could be done including changing ifaces IPs
None	--net=none	No network interfaces created inside container.	But dedicated namespace still created on start (at least was so)
Default: bridged + private networks	<nothing> + --net=somenetname	(mostly named docker0)	Private Networks organized by internal Docker DNS on 127.0.0.11 address and iptables.
Mapped from another container		All interfaces (namespaces) from one container reused in another - like in Hosted Type	Different containers could communicate with each other trough any IP/iface - even through 127.0.0.1. Ports shouldn`t override

Hands On: Docker Networking Hosted

in hypervisor:

:$ **ip addr**

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc fq_codel state UP group default qlen 1000
    link/ether 06:2c:a5:cc:29:00 brd ff:ff:ff:ff:ff:ff
    inet 172.31.40.231/20 brd 172.31.47.255 scope global dynamic eth0
       valid_lft 3163sec preferred_lft 3163sec
    inet6 fe80::42c:a5ff:fecc:2900/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:2e:e7:33:cd brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
```

:$ **docker run -it --rm --net=host centos /bin/bash**

/$ ip addr

/$ ip addr add 10.0.0.1/24 dev eth0

RTNETLINK answers: Operation not permitted

Hands On: Docker Networking None

```
:$  docker run -it --rm --net=none centos /bin/bash

/$ ip addr

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
```

Hands On: Docker Bridged

```
:$ ip addr show docker0
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:2e:e7:33:cd brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:2eff:fee7:33cd/64 scope link
       valid_lft forever preferred_lft forever

:$ docker run -it --rm centos /bin/bash
[root@60dcba2e9635 /]# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
11: eth0@if12: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```

Dockerfile - More Options

```
FROM openjdk:8-jdk-stretch
...
RUN apt-get update && apt-get upgrade -y && apt-get install -y ... && rm -rf ...
...
ARG user=jenkins
...
ENV JENKINS_HOME $JENKINS_HOME
...
RUN mkdir -p $JENKINS_HOME \
 && useradd -d "$JENKINS_HOME" -u ${uid} -g ${gid} -m -s /bin/bash ${user}
...
VOLUME $JENKINS_HOME
...
EXPOSE ${http_port}
...
USER ${user}
...
ENTRYPOINT ["/sbin/tini", "--", "/usr/local/bin/jenkins.sh"]
...
COPY install-plugins.sh /usr/local/bin/install-plugins.sh
ADD https://some.git.url
```

RUN - execute command inside container during building the image

ARG - local for dockerfile variable useful and overridable only in“docker build”, refrencable with ${ARG name}.

ENV - advertisese variable injected into Environment Variables in container by container start, overridable in “docker run”

VOLUME - creating directory that could be referenced in docker run command as volume mounting point by name (without full path).

EXPOSE - port container advertised to outside (needed for iptables auto-rules adding), mapping to real port is run option

USER - container will be executed as process of defined user

ENTRYPOINT, CMD - both could set binary/script started on container start, more details on the next slide

COPY - copy, path os source from building directory
ADD - enhanced COPY, supporting wildcards, --chown and URLs

Feel free to extend your knowledges with: https://docs.docker.com/engine/reference/builder/

Dockerfile - ENTRYPOINT, CMD

ENTRYPOINT, CMD - why 2 options to set an upstart command?

ENTRYPOINT - purposed to define binary for process #1 in container, if used - CMD is referred as it`s parameters. Could be overridden with --entrypoint option on start.

CMD - purposed to be redfinable on container start, set by the command in run line after image name: `docker run -it centos /bin/bash`. If used without ENTRYPOINT - substitute one.

Docker container goal	Dockerfile	docker run command	Executed on upstart script
Show time -no input options	FROM ubuntu ENTRYPOINT date	docker run date	/bin/sh date
		docker run date +%Z	/bin/sh date
	FROM ubuntu ENTRYPOINT [‘date’]	docker run date	date
Show time - options eligible and overridable	FROM ubuntu ENTRYPOINT [‘date’] CMD [‘+%A’]	docker run date	date +%A
		docker run date +%Y	date +%Y
Show time - run script overridable	FROM ubuntu CMD date	docker run date	/bin/sh date
		docker run /bin/bash	/bin/bash

Dockerfile - FROM

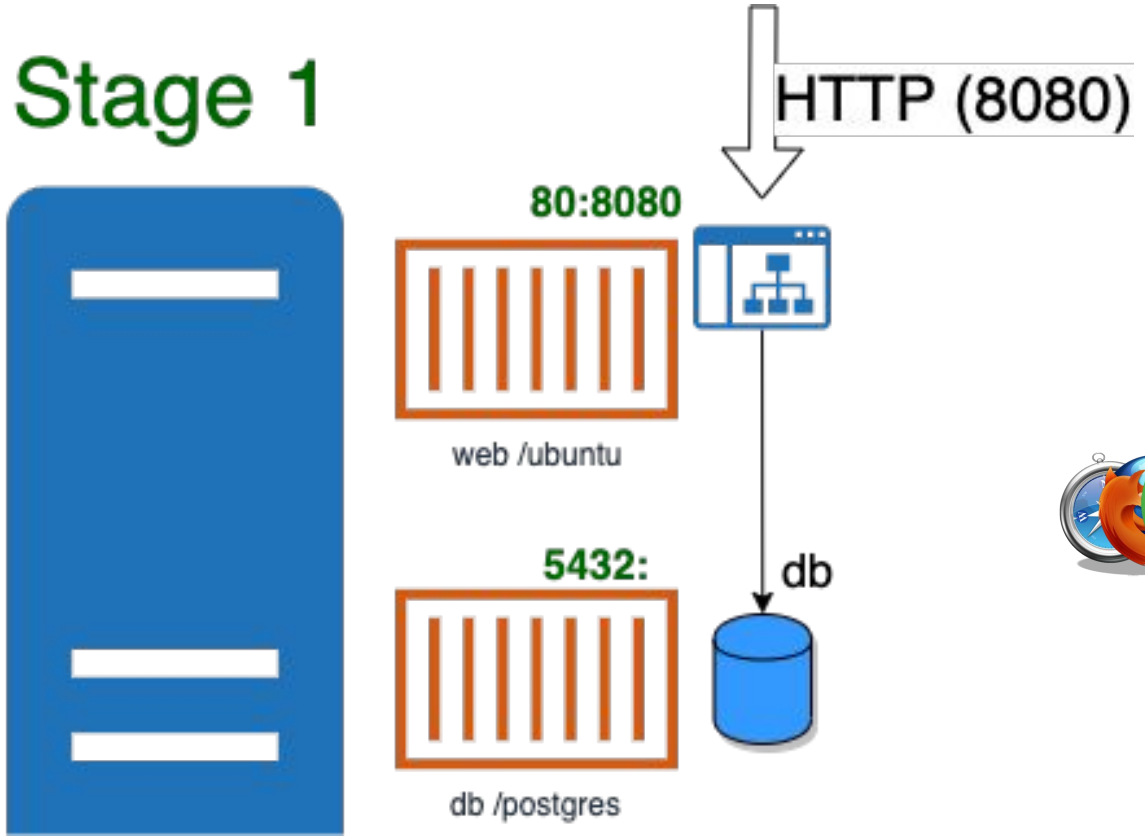
FROM could use not just some image from repository but also:

- defined "from scratch" for minimal images,
- use artifacts from intermediate image constructed in the same manifest - named as multistage

```
FROM scratch
COPY hello /
CMD ["/hello"]
```

```
FROM golang:1.7.3 AS builder
WORKDIR /go/src/github.com/alexellis/href-counter/
RUN go get -d -v golang.org/x/net/html
COPY app.go    .
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o app .

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=builder /go/src/github.com/alexellis/href-counter/app .
CMD ["./app"]
```

Hands On: Small Web App With DB, Stage 1




```
:$ git clone https://github.com/ask4ua/DKN
:$ docker network create mynet

Stage1/web:$ docker build ./ -t web
Stage1/web:$ docker run -p 8080:80 -d --name web --net mynet web

Stage1/db:$ docker build ./ -t db
Stage1/db:$ docker run  -d --name db --net mynet db
```

Serving host ec99fa097683

Time: 2019-11-08 17:41:29

Accessed path: /

Writing to DB status: Fail

Hands On: Docker Limitations

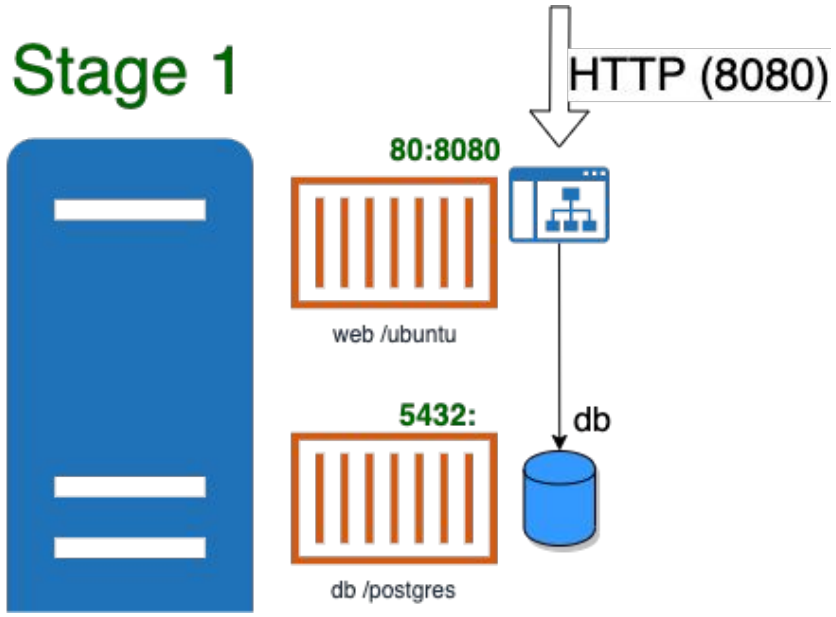
Docker Doesn`t Containers Upstart Dependedencies

```
:$ docker logs -f web
Mon Jul 13 15:33:58 2020 webapp: HTTP Server Starts
Mon Jul 13 15:33:58 2020 webapp: Initiating connection to DB

DB ERROR: Something is wrong in connecting to DB: could not translate host name "db" to address: Temporary failure in name resolution

:$ docker stop web
:$ docker start web
```

Docker Bridged --net

```
:$ docker ps
CONTAINER ID   IMAGE   COMMAND                  CREATED          STATUS          PORTS                    NAMES
ee9d1bfda40f   web     "python3 -u /app/web…"   4 minutes ago    Up 27 seconds   0.0.0.0:8080->80/tcp     web
14833ca9e600   db      "docker-entrypoint.s…"   5 minutes ago    Up 32 seconds   5432/tcp                 db

:$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
5ee8868a7c3e   bridge    bridge    local
be73f6f78fd0   host      host      local
22100c68614f   mynet     bridge    local
dba2602e072d   none      null      local

:$ docker network inspect mynet

$ brctl show
bridge namebridge id            STP enabled interfaces
br-22100c68614f         8000.0242823f7f96       no              veth178a6e0
                                                                vethda68891
```

How Docker -p Port Forwarding --net Are Working



Sysctl ip_forward enables forwarding between All interfaces.

Dockerd creates bridges, connects to them containers, dockerd like DHCP assign them IPs

Each Bridged network could have own IP space, Gateway, DNS

Dockerd manages IPtables to simulate containers isolation and to set Port Forwarding

--net mynet shared between containers has dedicated brifge and just disabling of some iptables isolation rules by iptables (no namespaces magic)

dockerd runs DNS on ip 127.0.0.11 to enable resolving IPs between containers in the same --net mynet by container names

Dockerfiles For The App

```
FROM postgres
ENV POSTGRES_USER='DBUSER' POSTGRES_DB='DBNAME' POSTGRES_PASSWORD='DBPASS'
COPY ./upstart.sh /docker-entrypoint-initdb.d
```

```
FROM ubuntu
RUN apt-get update && apt-get install --no-install-recommends --no-install-suggests -y git python3
python3-pip python3-setuptools python3-dev python3-psycopg2 stress
RUN pip3 install psycopg2
ENV DBUSER='DBUSER' DBPASS='DBPASS' DBNAME='DBNAME'
ENV DBHOST='db' DBPORT='5432'
ARG APPDIR='/app'
VOLUME ${APPDIR}
COPY src ${APPDIR}
ENTRYPOINT ["python3","-u","/app/webapp.py"]
```

Secrets Forwarding Into Container

	Environment Variables key = value	**Volume** Files with key=value, csv, archeive etc.

On Build	Via arguments override:
	--
	`--build-arg ARG1=Va --build-arg ARG2=Lue`
	
	For compatibility with on run redefine could be used in dockerfile:
	
	`ARG SecretArg=password`
	`ENV SecretENV=${SecretArg}`
		Dockerfile:
		
		`export DOCKER_BUILDKIT=1`
		`RUN --mount=type=secret,id=mysite.key command-to-run`
		
		or just VOLUME mount and some RUN commands

On Run	Via Environemnt variable redefine: `docker run -e EnvVariable=SomePass -e AnotherEnvVeriable=pass2`	Application should be able to read and parse files from volume

Docker Link

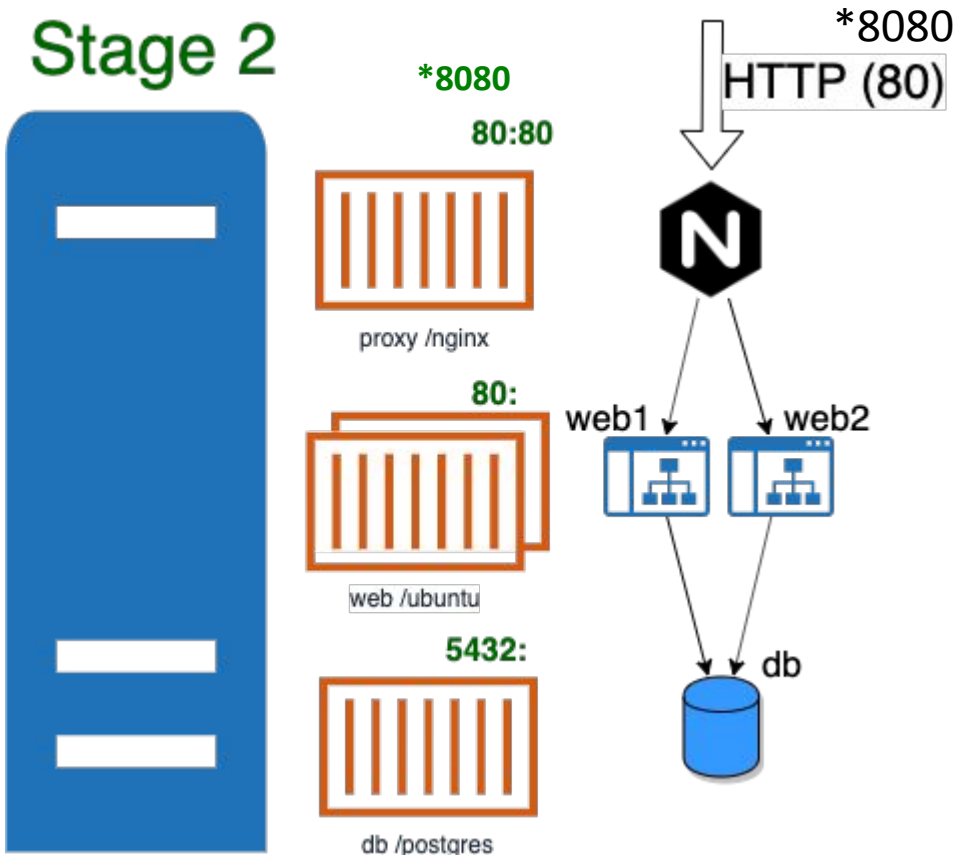
Docker link enables to start few containers on the same hosts with shared both Volumes and Environment variables.

Magic is done by Docker transparently by:

- reinjecting variables on start appending them with container name prefix followed by underscore,
- DNS records are filled into /etc/hosts of container.

```
:$ docker run -d --name database -e MYSQL_ROOT_PASSWORD=root mysql
:$ docker run -d --link database:db --name webapp web

:$ docker exec -ti webapp env | grep MYSQL_ROOT_PASSWORD
DB_ENV_MYSQL_ROOT_PASSWORD=root
```

Hands On: Small Web App With DB, Stage 2




```
:$ docker stop web webapp

:$ docker run -d --name web1 --net mynet web
:$ docker run -d --name web2 --net mynet web

Stage2/proxy:$ docker build ./ -t proxy

:$ docker run -p 8080:80 -d --name proxy --net mynet proxy
```

Serving host ec99fa097683

Time: 2019-11-08 17:41:29

Accessed path: /

Writing to DB status: Success

Serving host 78fe3f2c0be4

Time: 2019-11-08 17:41:29

Accessed path: /

Writing to DB status: Success

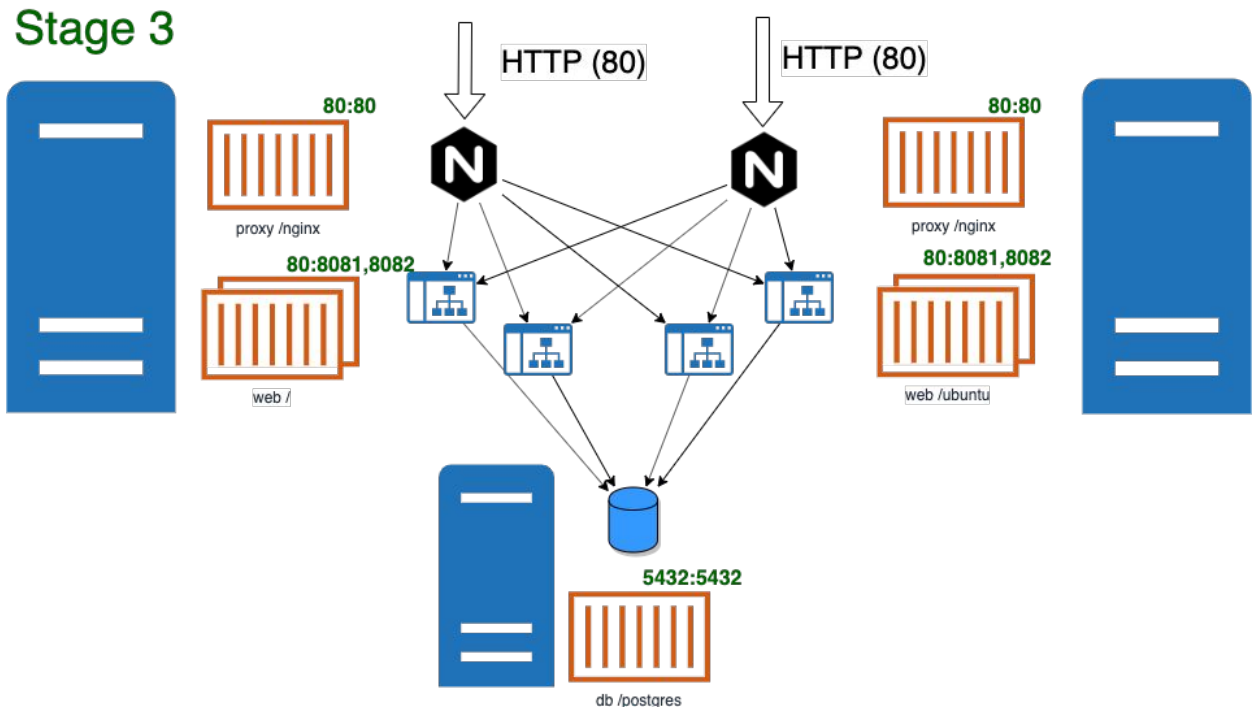
Hands On: Small Web App With DB, Stage 2

```
FROM nginx
COPY nginx.conf /etc/nginx/nginx.conf
```

nginx.conf:

```
events { }
http {
    upstream webapp {
        server web1:80;
        server web2:80;
        keepalive 10;
    }

    server {
        resolver 127.0.0.11 valid=10s;
        listen 80;
        location / {
            proxy_pass http://webapp;
        }
    }
}
```

Lection2 Home Task



Please try to: provide DB IP (docker --net DNS not working outside of host) and Secrets into docker WebApp config by ENV variable on docker run.

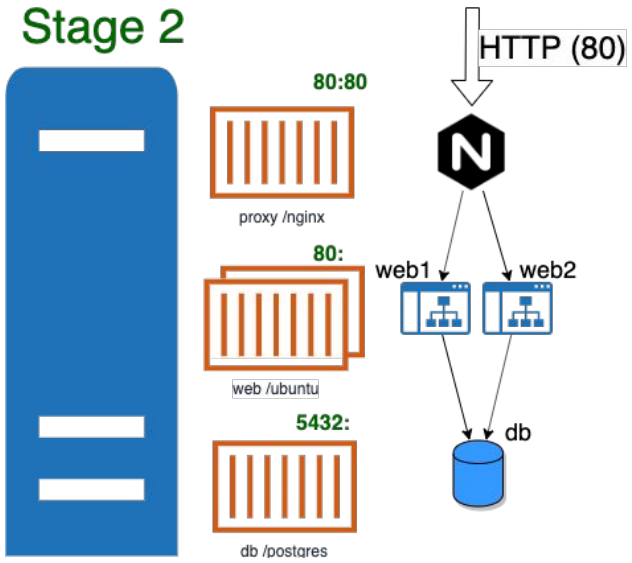
P.S.: Local host networking will not work for WebApps from other server!

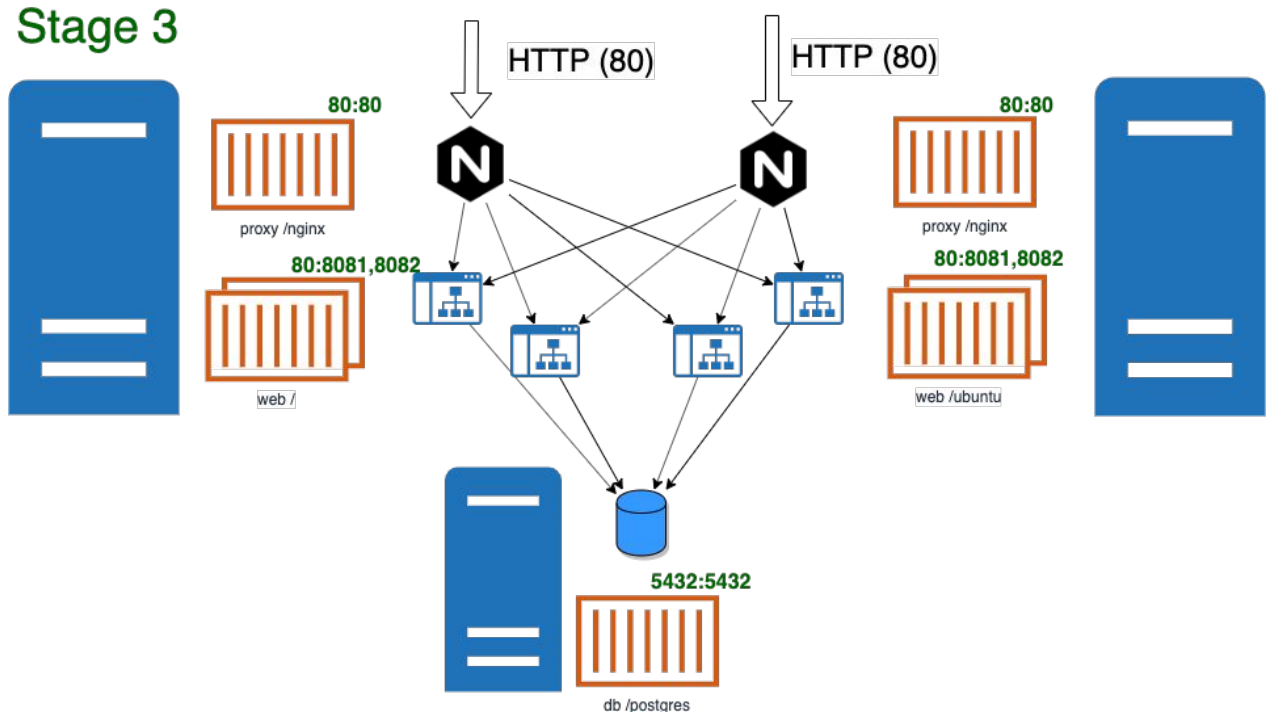
P.P.S.: Nginx Configs ok to hardcode - free NGINX doesn`t support keepalives - so WebApp should be running.

https://github.com/ask4ua/DKN/blob/master/Hometask/Section2/README.md

Section 3: Docker Compose, Docker Swarm, Kubernetes

Growing Service





Service Orchestration on Dockerd Requirements

What is missed in traditional docker

- Containers Upstart Dependencies,
- Network shared between Few Hosts, Network Isolation,
- DNS in shared network, DNS isolation,
- Scaling like a DNS++ = Service discovery
- Secrets Management

Docker Compose

Docker Compose - systemd for docker containers.

- Handles Upstart Dependencies.
- Support Scaling of Containers.
- Tracking containers status.
- Rolling updates.
- Keeping DNS Records.
- Isolating resources by namespaces (not kernel - just DNS)

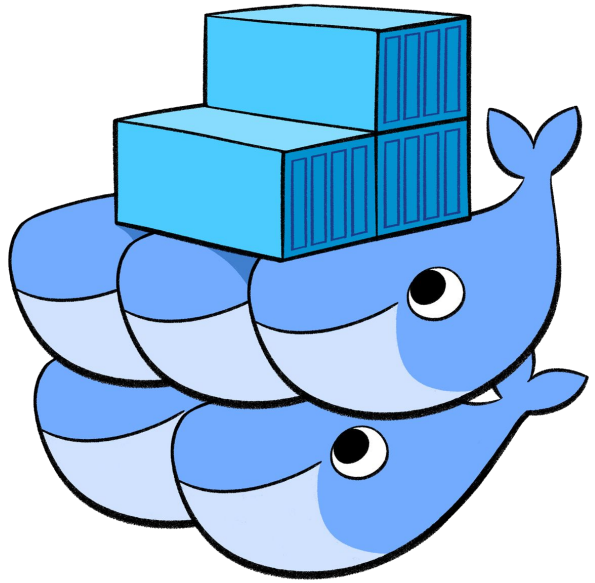
But all of this only around single node (hypervisor).

Docker Compose Config Example

```
version: "3"

services:
  whir-data:
    image: localhost:5000/whir-data
    deploy:
      replicas: 2
      update_config:
        parallelism: 2
        delay: 10s
        order: stop-first
    volumes:
          - "/home/volk/GIT/whir:/app"
          - "/home/volk/txt:/data"
    networks:
      - whirnet
...
```

```
...
  whir-parser:
        image: localhost:5000/whir-parser
        depends on:
          - whir-parser
        deploy:
          replicas: 1
          resources:
            limits:
              cpus: "0.5"
              memory: 128M
          restart_policy:
            condition: on-failure
        volumes:
            - "/home/volk/GIT/whir:/app"
            - "/home/volk/txt:/data"
        networks:
          - whirnet
networks:
        whirnet:
```

Docker Swarm



Connects few nodes(hypervisors),
roles: management, worker.

to firstly advertised in compose features added:

+ Multi-host networking,
+ Service Discovery + Load Balancing

But :

- not includes upstart sequence like docker compose.

Docker Swarm, Compose - Secrets Management

```
    postgres:
        image: docker.ask4ua.com/whir-db
        ports:
            - 5432:5432
        volumes:
            - postgres_vol:/var/lib/postgresql/data
        secrets:
            - whir_db_password
            - root_db_password
        #  mounted into: /run/secrets/secret-name

#environment:
        #      - POSTGRES_USER=whir
        #       - POSTGRES_DB=whir
        #       - POSTGRES_PASSWORD=password

        networks:
            - whirnet
        deploy:
            replicas: 1
            restart_policy:
                condition: any
secrets:
    whir_db_password:
        external: true
        #file: db_root_password.txt
    root_db_password:
        external: true
volumes:
    data_vol:
    postgres_vol:
networks:
    whirnet:
```

```
echo password | docker secret create whir_db_password -
```

Kubernetes

Kubernetes - What are you looking like?

```
docekr ps

0a28191b1f5c      rancher/hyperkube:v1.14.8-rancher1    "/opt/rke-tools/entr…"    … kube-proxy
edb47836b426      rancher/hyperkube:v1.14.8-rancher1    "/opt/rke-tools/entr…"   … kubelet
5b227fe9ddc1      rancher/hyperkube:v1.14.8-rancher1    "/opt/rke-tools/entr…"   …. kube-scheduler
fc1b918ca88f      rancher/hyperkube:v1.14.8-rancher1    "/opt/rke-tools/entr…"   … kube-controller-manager
1b0a25d5baf8      rancher/hyperkube:v1.14.8-rancher1    "/opt/rke-tools/entr…"   … kube-apiserver
a890a2f32f7d      rancher/rke-tools:v0.1.50             "/opt/rke-tools/rke-…"   …          etcd-rolling-snapshots
610098479e21      rancher/coreos-etcd:v3.3.10-rancher1  "/usr/local/bin/etcd…"   … etcd
…
5bd4b5b226d5      rancher/hyperkube:v1.14.8-rancher1  "/opt/rke-tools/entr…"     … kubelet
21a38aa88520      rancher/hyperkube:v1.14.8-rancher1    "/opt/rke-tools/entr…"    …         kube-proxy
```

Kubernetes - What are you?

Kubernetes project is only:

- Binaries built from source code for key kubernetes components,
- Documented state of APIs and approaches.

Not fixed implementation: this is why exists so huge amount of different ways to build (kubeadm, kops, rancher etc.) and to distribute (bynaries +systemd, containers) kubernetes.

kubernetes.io doesn`t say how to build your cluster - it`s saying only how it should be built to be used.

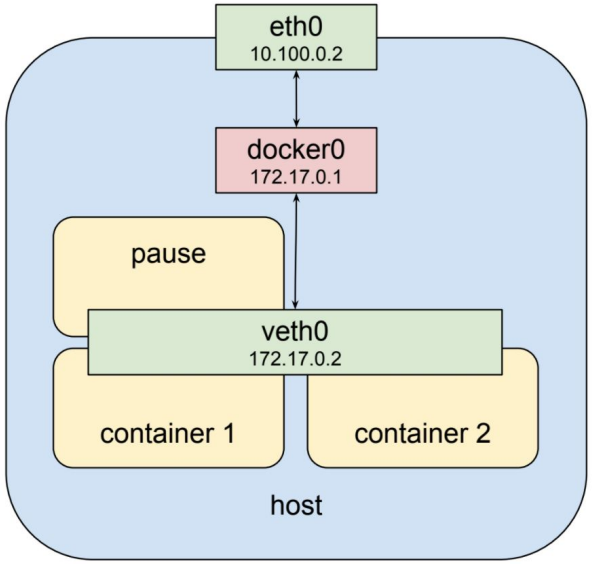
This approach is following the same State-strategy of describing environemtn as kuber implementing - you are changing manifest (yaml file for some resource) and apply it to kuber - kuber by themselve defines what need to be done to get the manifested state.

Pod - Minimal Entity Of Orchestration

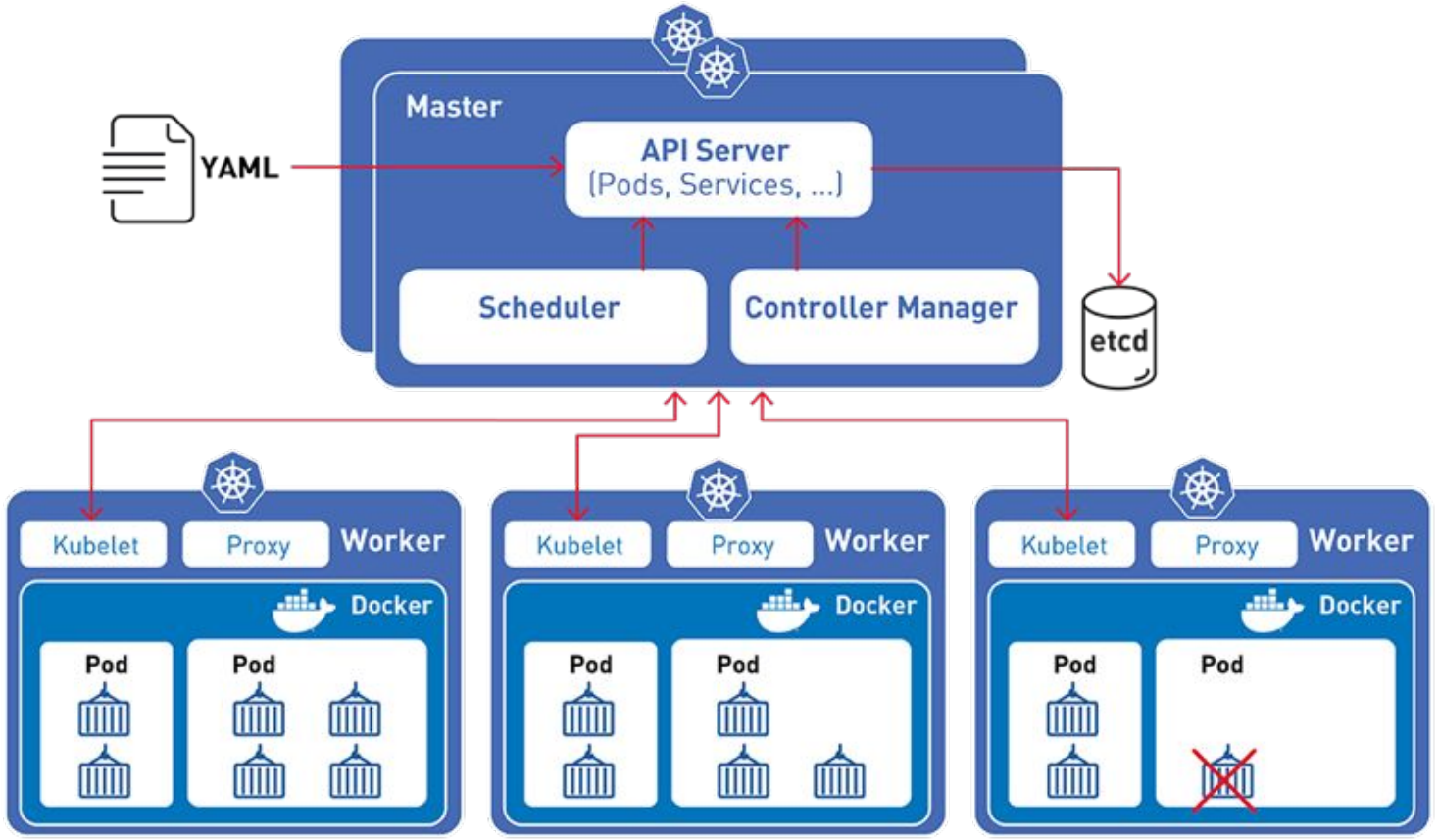
- Pod - is the same networking namespace shared to 1+ docker containers.

Example of Pod:

```
apiVersion: v1
kind: Pod
metadata:
        name: "games"
spec:
        containers:
        - image: docker-2048
        name: "2048"
        ports:
        - containerPort: 80
          hostPort: 8081

        - image: pengbai/docker-supermario
        name: "supermario"
        ports:
        - containerPort: 80
          hostPort: 8082
```



containerd - Docker without Docker Swarm and Compose capabilities



Pods Handling in Kuber

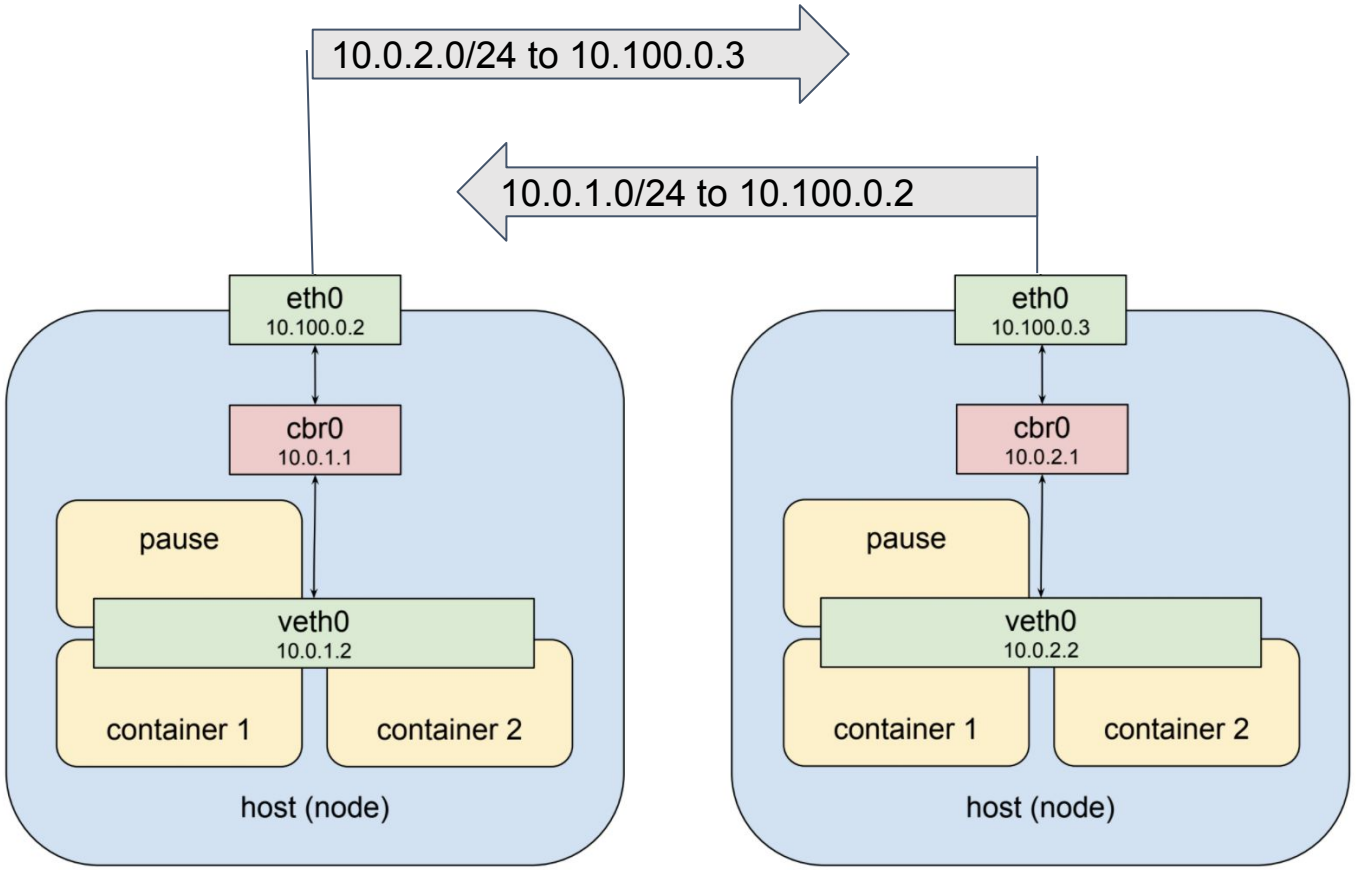
Pod is mortal.
Mostly suddenly mortal.

Kuber removing pod if pod considered unhealthy (not running at least one container in POD or container is dead by predefined healthcheck).

Removed with pod containers totally lose data stored in containers.

Kubernetes - Networking

Networking mechanism between nodes and pods in nodes is not defined by kuberentes!
So choose the plugin to use.



Hands On: Build Your First Kuber Cluster

https://vitux.com/install-and-deploy-kubernetes-on-ubuntu/

Build your first Kuber Cluster using kubeadm.

AWS EC2 ubuntu 18.04 LTS t2.medium 2 instances same VPC, enable LAN traffic and forward port 32080 to public on both.

Firstly run "sudo apt-get -y update"
Reboot after node rename!

When done:
check all is fine: **ubuntu@master-node**:~$ kubectl get pods --all-namespaces -o wide - all Statuses should be Running

Deploy your first service:
1. kubectl apply -f https://raw.githubusercontent.com/ask4ua/DKN/master/Practices/Section3/supermario.yml
2. kubectl describe pod -l app=supermario
3. Web: http://<Any node IP>:32080

P.S.: how to control supermario: https://microsite.nintendo-europe.com/super-mario-maker-manual/enGB/page_03.html
P.P.A: Don`t forget to turn off pricy VMs!

Handson: kubectl cluster - looking around

```
~$ kubectl get nodes -o wide

~$ kubectl get pods -A

~$ kubectl describe node prima

~$ kubectl describe pod <PODNAME>

~$ kubectl get pods -A -o wide
```

HandsOn: Kubernetes - Pod/Service

Last practice supermario file:

```
apiVersion: v1
kind: Service
metadata:
 name: supermario
spec:
 selector:
   app: supermario
 type: NodePort
 ports:
    - port: 8080
      nodePort: 32080
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: supermario
spec:
 replicas: 1
 selector:
   matchLabels:
     app: supermario
 template:
   metadata:
     labels:
       app: supermario
   spec:
     containers:
     - name: mario-container
       image: pengbai/docker-supermario
       ports:
       - containerPort: 8080
```

Section 4: Kuber - dive deeper

Practice Requirements

key: sent in email.
Password: P@ss4Student[eRt

ssh to: 52.28.238.50 -p 2224 -i students.key -l student##

Handson: kubectl cluster - looking around

```
~$ kubectl get nodes -o wide

~$ kubectl describe node prima

~$ kubectl get pods --all-namespaces -o wide

~$ kubectl get namespaces

~$ kubectl create namespace student100

~$ kubectl get namespace student100

~$ kubectl get namespace student100 -o yaml
```

Handson: kubectl cluster - looking around

~$ kubectl get namespace student100 -o yaml

```
apiVersion: v1
kind: Namespace
metadata:
  annotations:
    cattle.io/status:
'{"Conditions":[{"Type":"ResourceQuotaInit","Status":"True","Messag
e":"","LastUpdateTime":"2019-11-16T20:51:56Z"},{"Type":"InitialRole
sPopulated","Status":"True","Message":"","LastUpdateTime":"2019-1
1-16T20:51:56Z"}]}'
    lifecycle.cattle.io/create.namespace-auth: "true"
  creationTimestamp: "2019-11-16T20:51:54Z"
  finalizers:
  - controller.cattle.io/namespace-auth
  name: student100
  resourceVersion: "45010"
  selfLink: /api/v1/namespaces/student100
  uid: 09daed4c-b9b9-4803-89e0-ecff39b94956
spec:
  finalizers:
  - kubernetes
status:
  phase: Active
```

```
~$ cat student99.yml
apiVersion: v1
kind: Namespace
metadata:
  name: student99
```

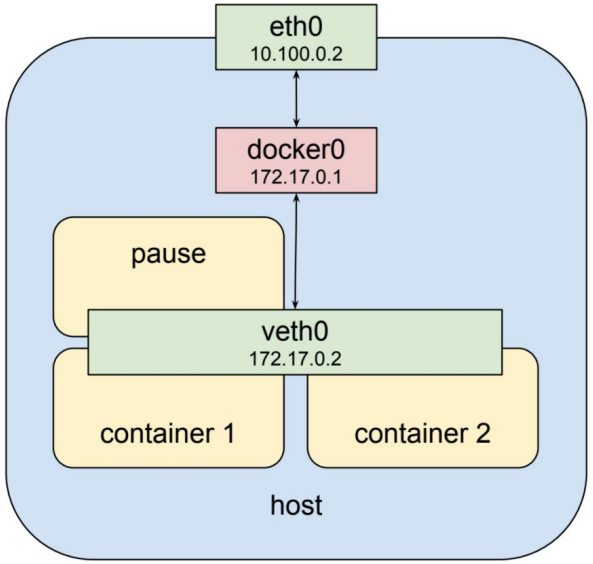
```
~$ kubectl apply -f student99.yml
namespace/student99 created

~$ kubectl get namespace student99
NAME        STATUS   AGE
student99   Active   8s
```

Pod - Minimal Entity Of Orchestration

- Pod - is the same networking namespace shared to 1+ docker containers. More looking like localhost with few containers.

Example of Pod:

```
apiVersion: v1
kind: Pod
metadata:
        name: "webapp"
spec:
        containers:
        - image: docker.ask4ua.com/webapp:latest
        name: "webapp"
        ports:
        - containerPort: 80
          hostPort: 80
```



Hands On: Starting Pod

```
~$  kubectl apply -f db_pod.yaml

~$  kubectl apply -f webapp_pod.yaml

~$ kubectl get pod webapp -o wide

~$ kubectl describe  pod webapp -n student99
...
Events:
  Type    Reason     Age       From                Message
  ----    ------     ----      ----                -------
  Normal  Scheduled  <unknown>  default-scheduler         Successfully assigned student99/webapp to k8s-worker-node13
  Normal  Pulling    2m44s     kubelet, k8s-worker-node13  Pulling image "docker.ask4ua.com/webapp"
  Normal  Pulled     2m35s     kubelet, k8s-worker-node13  Successfully pulled image "docker.ask4ua.com/webapp"
  Normal  Created    2m32s     kubelet, k8s-worker-node13  Created container webapp
  Normal  Started    2m32s     kubelet, k8s-worker-node13  Started container webapp

~$ kubectl -n student99 logs -f pod/webapp
Sat Nov 16 23:44:38 2019 HTTP Server Starts
Initial connection to DB
DB ERROR: Something is wrong in connecting to DB: could not translate host name "db" to address: Name or service not known
```

POD Limitations

1. Pod is not transparently scalable - you only could create more instances of initially same pod with different names (webapp1, webapp2 etc.)

1. If Pod doesn`t have more intermediate levels like replicaset/deployment - applying changes like env variables could be brought only by recreating pod (replicaset/deployment truly do the same - but transparently for user).

1. No references by DNS names - DNS names resolution is available only for services!

Hands on: Creating Services

```
apiVersion: v1
kind: Service
metadata:
  name: db
  namespace:
student99
  labels:
    app: db
spec:
  selector:
    app: db
  ports:
    - port: 5432
```

```
~$ kubectl apply -f db_service.yaml

~$ kubectl apply -f webappdb_service.yaml

~$ kubectl -n student99 get services -o wide
NAME     TYPE      CLUSTER-IP    EXTERNAL-IP   PORT(S)    AGE   SELECTOR
db       ClusterIP 10.43.2.129   <none>        5432/TCP   13m   app=db
```

Entity service as a connectable socket doesn`t exist - and living only as record on proxies!

Hands On - Service for Webapp

```
apiVersion: v1
kind: Service
metadata:
  name: webapp
  namespace: student99
  labels:
    app: webapp
...
```

```
…
spec:
  selector:
    app: webapp
  type: NodePort
  ports:
    - port: 80
      nodePort: 32099
```

```
~$ kubectl apply -f service/webapp_service.yaml
service/webapp created
```

```
~$ kubectl get nodes -o wide
NAME              STATUS   ROLES          AGE   VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION     CONTAINER-RUNTIME
k8s-master-node11   Ready    controlplane,etcd   10h   v1.16.2   10.5.11.111   <none>        Ubuntu 18.04.2 LTS   4.15.0-70-generic   docker://19.3.5
k8s-master-node12   Ready    controlplane,etcd   9h    v1.16.2   10.5.11.112   <none>        Ubuntu 18.04.2 LTS   4.15.0-50-generic   docker://19.3.5
k8s-master-node13   Ready    controlplane,etcd   9h    v1.16.2   10.5.11.113   <none>        Ubuntu 18.04.2 LTS   4.15.0-50-generic   docker://19.3.5
k8s-worker-node11   Ready    worker         9h    v1.16.2   10.5.11.211   <none>        Ubuntu 18.04.2 LTS   4.15.0-70-generic   docker://19.3.5
k8s-worker-node12   Ready    worker         9h    v1.16.2   10.5.11.212   <none>        Ubuntu 18.04.2 LTS   4.15.0-50-generic   docker://19.3.5
k8s-worker-node13   Ready    worker         9h    v1.16.2   10.5.11.213   <none>        Ubuntu 18.04.2 LTS   4.15.0-50-generic   docker://19.3.5


student@motel:~$ curl -X GET  10.5.11.211:32099
```

Hadson: Ping service from Host

```
root@webapp:/# ping webappdb -c1
PING webappdb.student99.svc.cluster.local (10.43.169.81) 56(84) bytes of data.
From 10.5.11.1 icmp_seq=1 Time to live exceeded

--- webappdb.student99.svc.cluster.local ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

root@webapp:/# ping db -c1
PING db.student99.svc.cluster.local (10.43.2.129) 56(84) bytes of data.
From 10.5.11.1 icmp_seq=1 Time to live exceeded

--- db.student99.svc.cluster.local ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

root@webapp:/#
```

Labeling

Key mechanism of dynamic teing (choosing) of entities in kubernetes - labeling.

For example service record URL:IP requires pod IPs to load balance to them, but pod could be recreated/moved to another node any time - so service is referencing it by Labels.

```
~$ kubectl -n student99 get services -o wide
NAME     TYPE      CLUSTER-IP    EXTERNAL-IP   PORT(S)    AGE   SELECTOR
db       ClusterIP   10.43.2.129   <none>      5432/TCP   13m   app=db
webappdb  ClusterIP   10.43.169.81  <none>      5432/TCP   24s   app=db
```

Hands On: Label Selecting

```
~$ kubectl get pod -l "app=webapp"
NAME                 READY   STATUS    RESTARTS   AGE
webapp               1/1     Running   0          5d3h
webapp-6bfbdf55-gxvnc   1/1     Running   0          3d23h
webapp-6bfbdf55-nqj2h   1/1     Running   0          3d23h

~$ kubectl get pod -l "app=db"
NAME   READY   STATUS    RESTARTS   AGE
db     1/1     Running   0          5d5h
```

Liveness and Readiness Checks

3 ways of checking POD status by kuberentes:

- by process #1 status (default)
- HTTP call
- exit code status for custom script executed on container

Hands On: Liveness Checks

```
apiVersion: v1
kind: Pod
...
spec:
  containers:
  - name: webapp
...
    livenessProbe:
      httpGet:
        path: /
        port: 80
      initialDelaySeconds: 15
      periodSeconds: 10
```

```
~$ kubectl logs -f webapp

Wed Jul  8 09:12:50 2020 webapp: HTTP Server Starts
Wed Jul  8 09:12:50 2020 webapp: Initiating connection to DB
Openned new DB connection
10.5.11.102 - - [08/Jul/2020 09:13:12] "GET / HTTP/1.1" 200 -
SQL: INSERT INTO logmessages(date, logmessage) VALUES ('Wed Jul  8 09:13:12 2020', 'Accessed path "/" via server name "webapp"');
Wed Jul  8 09:13:12 2020 webapp: Writing access fact to DB: Success
```

Liveness and Readiness Checks

- Liveness - checking is POD need to be killed?
- Readiness - checking is POD ready to be advertised via Services

Pod could be alive, but not yet ready.

No sence to have both Liveness and Readiness if they are checking same stuff.

Hands on: Creating One More DB Service

```
apiVersion: v1
kind: Service
metadata:
  name: webappdb
  namespace:
student99
  labels:
    app: db
spec:
  selector:
    app: db
  ports:
    - port: 5432
```

```
apiVersion: v1
kind: Service
metadata:
  name: db
  namespace:
student99
  labels:
    app: db
spec:
  selector:
    app: db
  ports:
    - port: 5432
```

```
~$ kubectl apply -f db_service.yaml

~$ kubectl apply -f webappdb_service.yaml

~$ kubectl -n student99 get services -o wide
NAME     TYPE      CLUSTER-IP    EXTERNAL-IP  PORT(S)   AGE  SELECTOR
db       ClusterIP 10.43.2.129   <none>       5432/TCP  13m  app=db
webappdb ClusterIP 10.43.169.81  <none>       5432/TCP  24s  app=db

~$ kubectl -n student99 delete service db
```

Configmap - How to Store and Share Configs

Configmap Variable:Value(s) could be injected into Pod using:

- envFrom option via Environment Variables.

  Configmap couldn`t be updated after pod started.

- mounting as files.

  Configmap update could be renewed in pod without pod restart.

Application inside container should be able to read values from Env variables or file to use them.

Hands On: Configmap

```
~$ kubectl apply -f configmap/webapp_configmap.yaml
~$ kubectl -n student99 delete pod/webapp
pod "webapp" deleted

~$ kubectl -n student99 apply -f pod/webapp_pod.yaml
```

webapp_pod_configmap.yaml:

```
apiVersion: v1
kind: ConfigMap
data:
  DBHOST: "webappdb"
metadata:
  name: webapp-configmap
  namespace: student99
```

```
...
  - name: webapp
    image: docker.ask4ua.com/webapp
...
   envFrom:
    - configMapKeyRef:
       name: webapp-configmap
```

Hands On: Kubernetes T-Shooting, Events

```
~$ kubectl apply -f pod/webapp_pod.yaml
pod/webapp created

~$  kubectl describe pod webapp -n student99
~$  kubectl logs -f pod/webapp -n student99
```

Be aware that config-maps are applied on container start:

```
~$ kubectl describe pod webapp -n student99

Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type     Reason     Age              From                  Message
  ----     ------     ----             ----                  -------
  Normal   Scheduled  <unknown>        default-scheduler     Successfully assigned student99/webapp to k8s-worker-node13
  Normal   Pulled     20s (x8 over 86s)  kubelet, k8s-worker-node13  Successfully pulled image "docker.ask4ua.com/webapp"
  Warning  Failed     20s (x8 over 86s)  kubelet, k8s-worker-node13  Error: configmap "config-map" not found
```

Secret - Configmap + base64 Encoding for “Security”

```
apiVersion: v1
kind: Secret
metadata:
  name: webapp-secret
stringData:
  DBUSER: "NEWDBUSER"
  DBPASS: "NEWDBPASS"
  DBNAME: "NEWDBNAME"
```

```
~$ kubectl apply -f webapp_secret.yaml
secret/webapp-secret created

~$  kubectl get secret
NAME                  TYPE                                  DATA   AGE
default-token-5rvpb   kubernetes.io/service-account-token   3      21h
webapp-secret         Opaque                                3      5s

~$  kubectl describe secret webapp-secret
...
Data
====
DBNAME:  9 bytes
DBPASS:  9 bytes
DBUSER:  9 bytes
```

```
~$  kubectl get secret webapp-secret -o yaml | grep "DBPASS:" | awk -F: '{print $2}' | base64 -d
NEWDBPASS%
```

Hands On: Secret For Webapp

webapp_pod_configmap_secret.yaml:

```
…
   envFrom:
   - configMapRef:
       name: webapp-configmap
   - secretRef:
       name: webapp-secret
…
```

```
~$ kubectl delete pod webapp
pod "webapp" deleted

~$ kubectl apply -f pod/webapp_pod_configmap_secret.yaml
pod/webapp created

~$ kubectl logs -f webapp
Wed Jul  8 10:31:08 2020 webapp: HTTP Server Starts
Wed Jul  8 10:31:08 2020 webapp: Initiating connection to DB
DB ERROR: Something is wrong in connecting to DB: FATAL:  password
authentication failed for user "NEWDBUSER"
```

Hands On: Secret For DB

db_pod_configmap_secret.yaml:

```
…
   env:
   - name: POSTGRES_USER
     valueFrom:
       secretKeyRef:
         name: webapp-secret
         key: DBUSER
...
```

```
~$ kubectl delete pod db webapp
pod "webapp" deleted

~$ kubectl apply -f pod/webapp_pod_configmap_secret.yaml
pod/db created

~$ kubectl apply -f pod/webapp_pod_configmap_secret.yaml
pod/webapp created

~$ kubectl logs -f webapp
Wed Jul  8 10:31:08 2020 webapp: HTTP Server Starts
Wed Jul  8 10:31:08 2020 webapp: Initiating connection to DB
DB ERROR: Something is wrong in connecting to DB: FATAL:  password
authentication failed for user "NEWDBUSER"
```

Volumes

On-disk files in a Container are ephemeral, which presents some problems for non-trivial applications when running in Containers.

First,

> *when a Container crashes, kubelet will restart it, but the files will be lost - the Container starts with a clean state.*

Second,

> *when running Containers together in a* `Pod` *it is often necessary to share files between those Containers. The Kubernetes* `Volume` *abstraction solves both of these problems.*

kubernetes.io

Hands On: Volume For DB

db_pod_secret_volume.yaml

```
…
  volumeMounts:
  - mountPath: /var/lib/postgresql/data/
    name: db-volume
 volumes:
  - name: db-volume
    emptyDir: {}

    #
    # hostPath:
    #   # Ensure the file directory is created.
    #   path: /var/local/aaa
    #   type: DirectoryOrCreate
    ###
    # awsElasticBlockStore:
    #   volumeID: <volume-id>
    #   fsType: ext4
    #
…
```

```
~$ kubectl delete pod db pod "webapp" deleted
~$ kubectl apply -f pod/db_pod_secret_volume.yaml
~$ kubectl describe pod db
…
Containers:
...
    Mounts:
      /var/lib/postgresql/data/ from db-volume (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-5rvpb (ro)
```

Hands On: Secrets/Configmap Mount via Volumes

db_pod_secret_volume.yaml

```
…
  volumeMounts:
  - mountPath: /var/lib/postgresql/data/
    name: db-volume
 volumes:
  - name: db-volume
    emptyDir: {}

    #
    # hostPath:
    #   # Ensure the file directory is created.
    #   path: /var/local/aaa
    #   type: DirectoryOrCreate
    ###
    # awsElasticBlockStore:
    #   volumeID: <volume-id>
    #   fsType: ext4
    #
…
```

```
~$ kubectl delete pod db pod "webapp" deleted
~$ kubectl apply -f pod/db_pod_secret_volume.yaml
~$ kubectl describe pod db
…
Containers:
...
    Mounts:
      /var/lib/postgresql/data/ from db-volume (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-5rvpb (ro)
```

Init Containers

Additional container into the pod starting first and next pod will not be started before init container “finished successfully”.

While initpod is not finished pod is not considered healthy (Running).

It could be more than 1 init containers :)

Hands On: Init Containers

webapp_pod_configmap_secret_init.yaml

```
…
spec:
 initContainers:
 - name: pingdb
   envFrom:
   - secretRef:
       name: webapp-secret
   - configMapRef:
       name: webapp-configmap
   image: vovolkov/webapp
   command: ["python3","-u","/app/pingdb.py"]
 containers:
…
```

```
~$  kubectl delete pod webapp
pod "webapp" deleted
~$   kubectl apply -f webapp_pod_configmap_secret_init.yaml
pod/webapp created

~$ kubectl get pods
NAME     READY   STATUS     RESTARTS   AGE
db       1/1     Running    0          30m
webapp   0/1     Init:0/1   0          8s

~$ kubectl logs webapp -- pingdb

~$ kubectl logs webapp -- pingdb
```

Final Resolving of Webapp/DB Upstart Issue

Webapp should be rewritten for:

- support of reconnecting to DB
- or fail on unsuccessful connection exiting the process (and failing pod)

Pods are mortal.
Mostly suddenly mortal.

It`s ok to use pod death functionality for application management purpose.

Section 5: Deployments

Kuber Scaling - Why?

Reliability: By having multiple versions of an application, you prevent problems if one or more fails. This is particularly true if the system replaces any containers that fail.

Load balancing: Having multiple versions of a container enables you to easily send traffic to different instances to prevent overloading of a single instance or node. This is something that Kubernetes does out of the box, making it extremely convenient.

Scaling: When load does become too much for the number of existing instances, Kubernetes enables you to easily scale up your application, adding additional instances as needed.

Common Kuber Controllers

Kuber can trace how many instances of some pods are declared and keep the scale even on crash of some pods or inflight changing of counter.

---- Base entities, supporting replicas:

Replication Controller - first way of describing multiple instances pods.
Replication Set - like a replication controller, but with more enhanced mechanism of label selector

---
Deployment - operating with ReplicaeSet with possibility of defining rollout strategy + variables enhancements inside manifest

Daemonset - like a **deployment**, but with strict replicas counter = worker nodes count with 1 instance per node.

Statefulset - like a **deployment** but with declaration of external persistent storages.

Deployment Config Example

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
   app: webapp
  namespace: student99
  name: webapp
spec:
  replicas: 2
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 2        # how many pods we can add at a time
      maxUnavailable: 0
  selector:
    matchLabels:
      app: webapp
...
```

```
...
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
      - name: webapp
        image: docker.ask4ua.com/webapp:fix
        ports:
        - containerPort: 80
          hostPort: 80
        imagePullPolicy: Always
        livenessProbe:
          httpGet:
            path: /
            port: 80
```

Hands On - Deployment

```
~$ kubectl apply -f deployment/webapp_deployment.yaml
service/webapp created

~$ kubectl get deployment/webapp -n student99 -o wide
NAME    READY   UP-TO-DATE   AVAILABLE   AGE    CONTAINERS   IMAGES                  SELECTOR
webapp  1/2     2            1           3m40s  webapp       docker.ask4ua.com/webapp  app=webapp
```

```
~$ kubectl get pods -n student99
NAME                      READY   STATUS    RESTARTS   AGE
db                        1/1     Running   0          151m
webapp                    1/1     Running   0          60m
webapp-7767d895f4-58p7n   1/1     Running   0          5m18s
webapp-7767d895f4-jm2wd   1/1     Running   0          5m18s
```

Hands On - Some More Details About Labeling

Deployment - is just the format of management.
Service continues to work based on labels:

```
~$ curl -X GET  10.5.11.102:32099
...
            <p>Serving host webapp-7767d895f4-jm2wd</p>
....
~$ curl -X GET  10.5.11.102:32099
...
            <p>Serving host webapp</p>
...
~$ curl -X GET  10.5.11.102:32099
...
            <p>Serving host webapp-7767d895f4-58p7n</p>
...
~$ kubectl get pods -n student99
NAME                  READY   STATUS    RESTARTS   AGE
db                    1/1     Running   0          151m
webapp                1/1     Running   0          60m
webapp-7767d895f4-58p7n   1/1     Running   0          5m18s
webapp-7767d895f4-jm2wd   1/1     Running   0          5m18s
```

Hands On - Scaling Deployment

```
~$ kubectl -n student99 scale --replicas=4 deployment/webapp
deployment.apps/webapp scaled

~$ kubectl get pods -n student99 -o wide
NAME                      READY   STATUS              RESTARTS   AGE    IP          NODE                NOMINATED NODE   READINESS GATES
db                        1/1     Running             0          158m   10.42.5.5   k8s-worker-node11   <none>           <none>
webapp                    1/1     Running             0          67m    10.42.4.8   k8s-worker-node13   <none>           <none>
webapp-7767d895f4-58p7n   1/1     Running             0          12m    10.42.3.6   k8s-worker-node12   <none>           <none>
webapp-7767d895f4-94f96   0/1     ContainerCreating   0          6s     <none>      k8s-worker-node11   <none>           <none>
webapp-7767d895f4-jm2wd   1/1     Running             0          12m    10.42.4.9   k8s-worker-node13   <none>           <none>
```

Deployments Under the Hood: Replica Sets

```
~$ kubectl get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
webapp  2/2     2            2           36s

~$ kubectl get replicaset
NAME               DESIRED   CURRENT   READY   AGE
webapp-6957d7896b  2         2         2       42s

~$ kubectl get deployment webapp -o yaml | grep -i replicaset
    message: ReplicaSet "webapp-6957d7896b" has successfully progressed.
    reason: NewReplicaSetAvailable
```

Deployment Rollout (ReplicaSets view)

```
~$ kubectl set image deployments/webapp webapp=vovolkov/webapp:latest
deployment.apps/webapp image updated

~$ kubectl get replicasets
NAME               DESIRED   CURRENT   READY   AGE
webapp-68f88b4f7d  2         2         1       8s
webapp-6957d7896b  1         1         1       16m

~$ kubectl get pods
NAME                  READY   STATUS        RESTARTS   AGE
db                    1/1     Running       0          29h
webapp                1/1     Running       0          27h
webapp-6895f59b87-cg6nz   1/1     Terminating   0          88s
webapp-6895f59b87-nr7t9   1/1     Terminating   0          88s
webapp-68f88b4f7d-dcpcf   1/1     Running       0          24s
webapp-68f88b4f7d-x7r4p   1/1     Running       0          24s

~$ kubectl rollout status deployments/webapp
Waiting for deployment "webapp" rollout to finish: 2 old replicas are pending termination…

$ kubectl get replicaset
NAME               DESIRED   CURRENT   READY   AGE
webapp-6895f59b87  0         0         0       4m4s
webapp-68f88b4f7d  2         2         2       14m
```

Rollout Strategies

```
apiVersion: apps/v1
kind: Deployment
...
spec:
  replicas: 2
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 2        # how many pods we can add at a time
      maxUnavailable: 0
...
```

```
apiVersion: v1
kind: Service
metadata:
  name: webapp
  namespace: student99
  labels:
    app: webapp
...
```

Ramped via **Deployment strategy Recreate** - remove all old than start create new starting from 1 replica.

RollingUpdate via **Deployment strategy RollingUpdate** - remove 1 (or more) by 1 and create new - simulteneously works both versions.

Blue/Green via **Service label selector** extended with label corresponding to version.

Canary via 2 Deployments **behind same Service** - just via downscaling/upsacaling Deployments with old/new versions of containers.

Pods Scheduling

There are 4 factors affecting k8s cluster pod scheduler:

1. NodeName in pod specification
   forcing
1. Resource Limits
   identifying best node by resources
1. Taints on nodes /Tolerations for pods
   restrictions on Nodes and ignoring of this restrictions
1. Affinity/Anti affinity with Required/Prefered conditions

```
-$ kubectl get pods -o wide
NAME     READY   STATUS    RESTARTS   AGE    IP            NODE    NOMINATED NODE   READINESS GATES
db       1/1     Running   0          132m   10.42.1.43    prima   <none>           <none>
webapp   1/1     Running   0          62m    10.42.1.53    prima   <none>           <none>
```

1. nodeName

```
apiVersion: apps/v1
kind: Deployment
...
spec:
 template:
  spec:
   nodeName: "prima"
   initContainers: ....
   Containers: ....
```

```
~$ kubectl apply -f webapp_deployment_nodename.yaml
deployment.apps/webapp configured

~$ kubectl get pods -o wide
NAME     READY   STATUS    RESTARTS   AGE    IP          NODE    NOMINATED NODE   READINESS GATES
webapp-6bfbdf55-gxvnc    1/1     Running     0          14s    10.42.1.58   prima    <none>           <none>
webapp-6bfbdf55-nqj2h    1/1     Running     0          14s    10.42.1.59   prima    <none>           <none>
```

2. Resource Limit

Kuber collecting info about Nodes CPU/Ram size (not collecting about actual utilization :(and mapping it to reserved values

```
apiVersion: apps/v1
kind: Deployment
...
spec:
...
  template:
...
    spec:
      containers:
      - name: webapp
        resources:
          requests:
            memory: "10Mi"
            cpu: 0.001
          limits:
            memory: "50Mi"
            cpu: 2
```

```
~$ kubectl apply -f webapp_deployment_webapp_deployment_limits.yaml

~$ kubectl describe pod webapp-7f7446d744-68mqd
...
    Limits:
      cpu:     20m
      memory:  50Mi
    Requests:
      cpu:     1m
      memory:  10Mi
...
```

Resource is a hard check for scheduling!
Limit is a soft check till scheduling is possible - than used for selecting best pod for evicting!

Taints/Tolerations

Tainting - locking **node** from scheduling or/and executing on Node.

Actions: NoSchedule, NoExecution

Toleration - setting **pod** which tainting tags with actions could be ignored :P

```
~$ kubectl get pods -o wide
NAME                READY   STATUS    RESTARTS   AGE    IP          NODE    NOMINATED NODE   READINESS GATES
...
webapp-6bfbdf55-gxvnc   1/1     Running   0          4d     10.42.1.58   prima   <none>           <none>
webapp-6bfbdf55-nqj2h   1/1     Running   0          4d     10.42.1.59   prima   <none>           <none>

~$ kubectl taint nodes prima noapp="True":NoExecute
node/prima tainted
~$ kubectl describe pod prime

~$ kubectl get pods -o wide
webapp-d44df4c86-gdm9h   1/1     Running   0          69s    10.42.0.75   pryluky   <none>           <none>
webapp-d44df4c86-kfht2   1/1     Running   0          69s    10.42.0.76   pryluky   <none>           <none>
```

Taints/Tolerations

```
apiVersion: apps/v1
kind: Deployment
...
spec:
...
  template:
    ...
    spec:
      tolerations:
      - key: noapp
        operator: "Exists"
        effect: "NoExecute"
      containers:
      ...
```

Section5/deployment$ kubectl apply -f webapp_deployment_toleration.yaml

~$ kubectl get pods -o wide

```
NAME                  READY   STATUS        RESTARTS   AGE     IP           NODE     NOMINATED NODE   READINESS
GATES
db-86cf74bc98-dnnkp       1/1     Running       0          14m     10.42.0.74   pryluky   <none>           <none>
webapp-76cddb94bf-nmghb   1/1     Running       0          13s     10.42.1.176  prima     <none>           <none>
webapp-76cddb94bf-wbzz5   1/1     Running       0          13s     10.42.1.175  prima     <none>           <none>
webapp-d44df4c86-kfht2    1/1     Terminating   0          8m5s    10.42.0.76   pryluky   <none>           <none>
webapp-d44df4c86-z8n7t    1/1     Terminating   0          5m35s   10.42.0.77   pryluky   <none>           <none>
```

~$ kubectl scale deployments/webapp --replicas 3

deployment.apps/webapp scaled

~$ kubectl get pods -o wide

```
NAME                  READY   STATUS              RESTARTS   AGE     IP           NODE     NOMINATED NODE   READINESS
GATES
db-86cf74bc98-dnnkp       1/1     Running             0          18m     10.42.0.74   pryluky   <none>           <none>
webapp-76cddb94bf-lv8c6   0/1     ContainerCreating   0          5s      <none>       pryluky   <none>           <none>
webapp-76cddb94bf-nmghb   1/1     Running             0          3m52s   10.42.1.176  prima     <none>           <none>
webapp-76cddb94bf-wbzz5   1/1     Running             0          3m52s   10.42.1.175  prima     <none>           <none>
```

Pods Affinitiy

Affinity affects scheduling mechanisms to force scheduling policy:

```
spec:
    affinity:
      podAntiAffinity:
        preferredDuringSchedulingIgnoredDuringExecution:
        - podAffinityTerm:
            labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - webapp
            topologyKey: kubernetes.io/hostname
          weight: 100
```

Nodes Management

Nodes maintenance: Cordon Node, Drain Node

Cordon - no schedule new pods (existing remains) - via noSchedule taint
Drain - evict all pods via no Execute

```
~$ kubectl describe node prima
...
Taints:             <none>


~$ kubectl drain prima --ignore-daemonsets
pod/coredns-7c5566588d-h6twt evicted
node/prima evicted

~$ kubectl describe node prima
...
Taints:             node.kubernetes.io/unschedulable:NoSchedule
```

Application Deployment to Kubernetes

What is Containerized Application

Microservice containerized App - is a combination of containers dependent by container versions.

Containerized App build - is a list of containers and their versions + cluster-related config updates.

What is Containerized Application

New build deployment:

- updating container versions (adding new one if needed, removing old one if)
- changing configs (config-maps)
- and applying/rotating secrets.

Straight Way: Jenkins + With custom scripts + kubectl

Affinity affects scheduling mechanisms to force scheduling policy:

Stage View

	Connect to Azure	Get Application IDs	Get Service Principals	Parallel - Get Snapshot	Parallel - Promote Container Artifact	Parallel - Replace k8's variables	Parallel - Deploy Containers	Parallel - Update Snapshot Properties	Parallel - Cleanup Old Containers
Average stage times: (Average full run time: ~16min 53s)	5s	33s	3s	3s	1s	1min 16s	30s	3s	6s
#51 Nov 21 20:29 No Changes	5s	33s	4s	3s	1s	1min 19s	1min 0s	3s	6s

Othe Pipeline-like Solutions

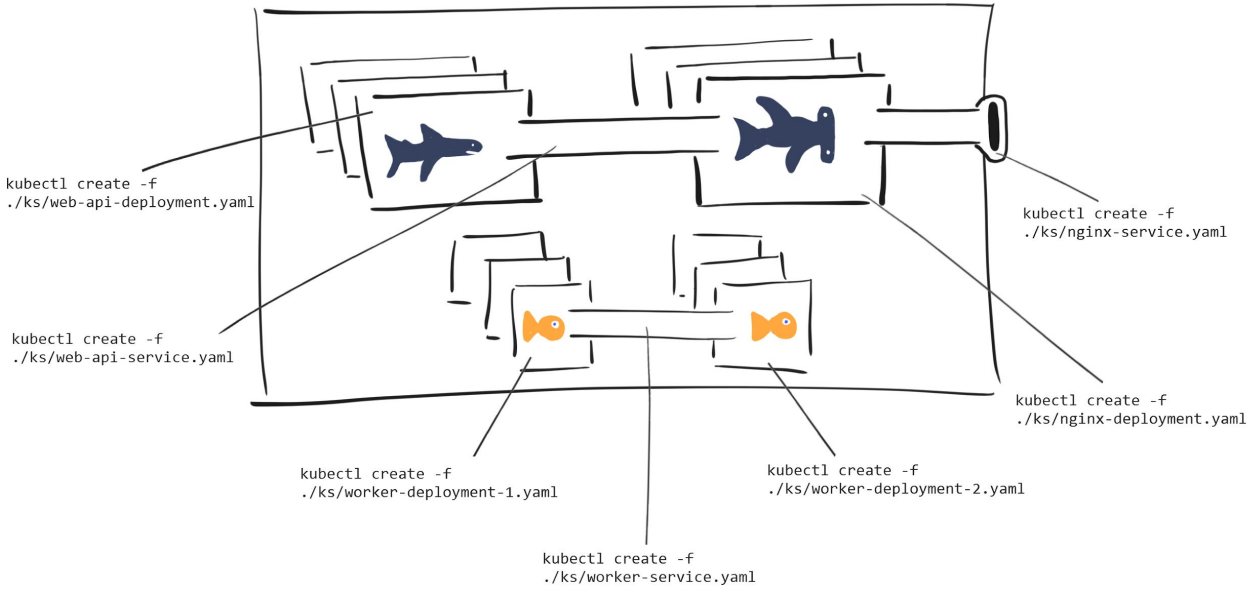
Rancher pipeline: https://rancher.com/docs/rancher/v2.x/en/project-admin/tools/pipelines/

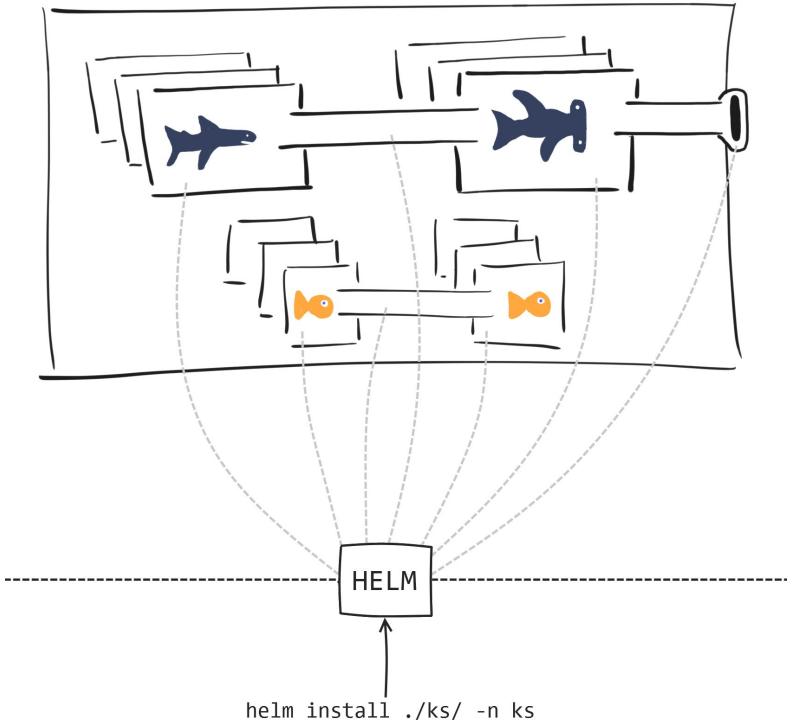
Gitlab https://docs.gitlab.com/ee/ci/pipelines.html

HELM

What is Helm:

- format of yaml-like written in Go manifest templates,
- able to generate Yamls and apply them to cluste,
- and keep on cluster history of it`s deployments.





HandsOn: Helm Chart, Umbrella Helm Chart

Chart.yaml

```
apiVersion: v2
name: webapp
description: A Helm chart for Kubernetes
…
dependencies:
- name: web         # from search results above
  version: 0.1.0     # Also from the search results above
  repository: file://../web
- name: db          # from search results above
  version: 0.1.0     # Also from the search results above
  repository: file://../db
```

```
Section6/helm/webapp
~$ helm dependency update
Saving 2 charts
Deleting outdated charts

~$ tree
.
├── Chart.lock
├── Chart.yaml
├── charts
│   ├── db-0.1.0.tgz
│   └── web-0.1.0.tgz
├── templates
│   ├── _helpers.tpl
│   ├── webapp_configmap.yaml
│   └── webapp_secret.yaml
└── values.yaml
```

HandsOn: Helm Chart, Umbrella Helm Chart

```
~$ helm upgrade --install webapp-helmed ./
Release "webapp-helmed" does not exist. Installing it now.
NAME: webapp-helmed
LAST DEPLOYED: Mon Jul 27 17:30:14 2020
NAMESPACE: student99
STATUS: deployed
REVISION: 1

~$ helm list
NAME            NAMESPACE   REVISION    UPDATED                                 STATUS      CHART         APP VERSION
webapp-helmed   student99   1           2020-07-27 17:30:14.257778592 +0300 EEST    deployed    webapp-0.1.0  1.16.0

~$ kubectl get deployments
webapp-helmed      3/3   3     3     6m42s
webapp-helmed-db   1/1   1     1     6m42s

~$ kubectl get deployment/webapp-helmed -o json | jq -r '.metadata.labels'
{
  "app.kubernetes.io/instance": "webapp-helmed",
  "app.kubernetes.io/managed-by": "Helm",
  "app.kubernetes.io/name": "web",
  "app.kubernetes.io/version": "1.16.0",
  "helm.sh/chart": "web-0.1.0"
}
```

HandsOn: Helm Chart, Umbrella Helm Chart

```
$ helm upgrade --install webapp-helmed ./ --set web.image.tag="latest"
REVISION: 2

$ kubectl get deployment webapp-helmed -o json | jq -r '.spec.template.spec.containers[].image'
vovolkov/webapp:latest

$ helm rollback webapp-helmed
Rollback was a success! Happy Helming!
$ helm list
NAME            NAMESPACE   REVISION    UPDATED                                 STATUS      CHART          APP VERSION
webapp-helmed   student99   3           2020-07-27 18:03:29.444947626 +0300 EEST    deployed        webapp-0.1.0    1.16.0

$ kubectl get deployment webapp-helmed -o json | jq -r '.spec.template.spec.containers[].image'
vovolkov/webapp:fix

$ kubectl get secrets
NAME                                TYPE                 DATA   AGE
sh.helm.release.v1.webapp-helmed.v1   helm.sh/release.v1   1      3m50s
sh.helm.release.v1.webapp-helmed.v2   helm.sh/release.v1   1      3m43s
sh.helm.release.v1.webapp-helmed.v3   helm.sh/release.v1   1      103s
```

WHat Helm Achieves

Generator has predefined label sections making it possible to easy:

- deploy some solution creating all defined in template entities
- update/delete only solution related entities on update,
- to have in parallel more than 1 instances of the same chart in the same deployment,

Keeping in secrets previous versions of deployment makes possible:

- auto rollback unsuccessful rollouts,
- rollback to any known previous version,
- keep track on all interventions via Helm to the cluster.

Supporting dependencies between charts and overriding Values.yaml:

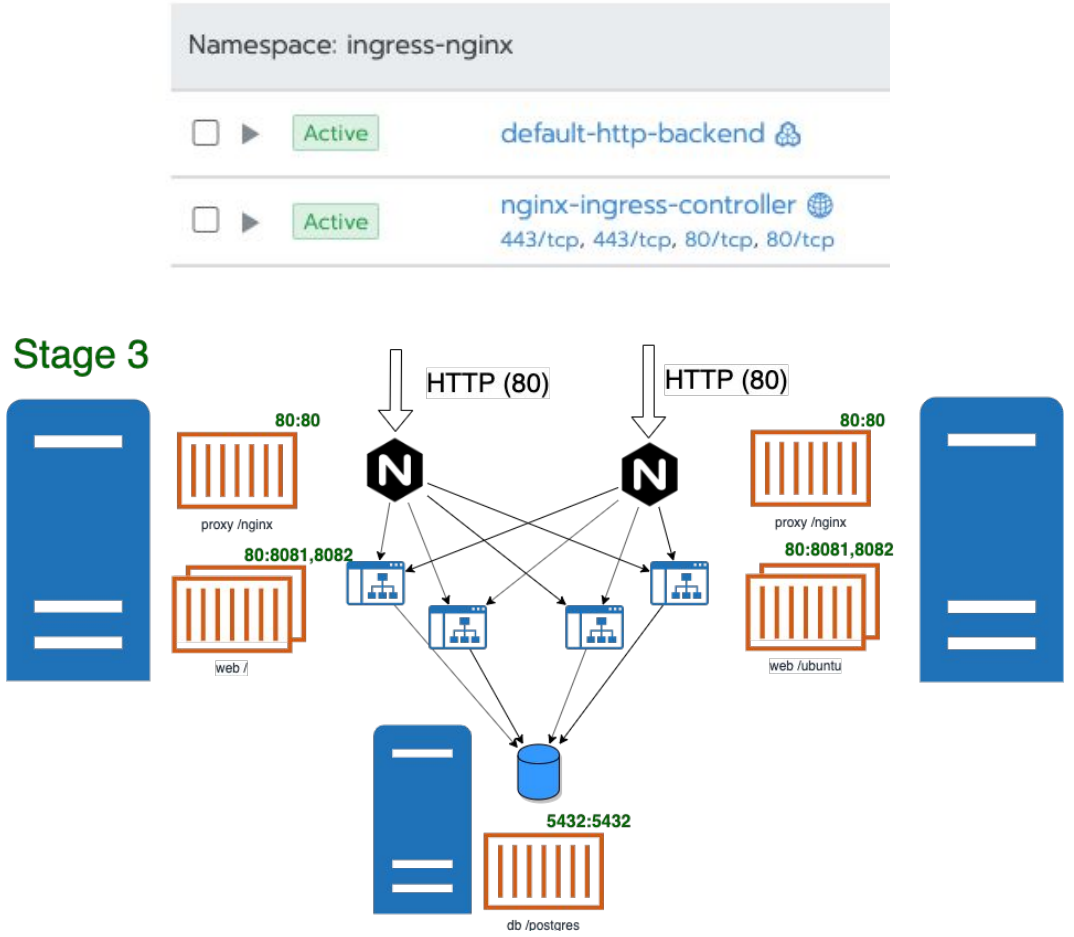
- make big deployments from independent packages (charts)

And pregenerated by Helm auto-test scripts and auto-scaling containers.

Cluster HTTP Ingress

Ingress

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
 annotations:
   nginx.ingress.kubernetes.io/rewrite-target: /$2
   nginx.ingress.kubernetes.io/use-regex: "true"
 name: webapp
spec:
 rules:
 - host: pro.camp
   http:
     paths:
     - backend:
         serviceName: webapp
         servicePort: 80
       path: /student00
```



Managing Access to Cluster

Service Account, User Accounts Management In Kuber

Kubernetes distinguishes between the concept of a user account and a service account for a number of reasons:

- User accounts are for humans. Service accounts are for processes, which run in pods.
- User accounts are intended to be global. Names must be unique across all namespaces of a cluster, future user resource will not be namespaced. Service accounts are namespaced.
- Typically, a cluster's User accounts might be synced from a corporate database, where new user account creation requires special privileges and is tied to complex business processes. Service account creation is intended to be more lightweight, allowing cluster users to create service accounts for specific tasks (i.e. principle of least privilege).

Service Accounts In Kuber

Service Accounts are created to manage access from cluster Pods to kuber API using internally-generated tokens.

Service accounts are namespace-oriented

```
$~ kubectl get serviceaccounts/metrics-server -n kube-system -o yaml
...
kind: ServiceAccount
metadata:
...
  name: kube-state-metrics
  namespace: kube-system
secrets:
- name: metrics-server-token-9dt4w


$~  kubectl get secret kube-state-metrics-token-nnqnd -n kube-system -o yaml
```

How To Assign Role to Pod Or User

Role/ClusterRole

```
apiVersion:
rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  rules:
- apiGroups:
  - ""
  resources:
  - pods
  - nodes
..
  verbs:
  - get
...
- apiGroups:
  - extensions
  resources:
  - deployments
  verbs:
  - get
...
```

RoleBinding

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: system:metrics-server
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:metrics-server

subjects:
- kind: ServiceAccount
  name: metrics-server
  namespace: kube-system
```

```
subjects:
- apiGroup:
rbac.authorization.k8s.io
  kind: User
  name: u-37f47gz5nd
```

ServiceAccount

```
kind: ServiceAccount
name: kube-state-metrics
namespace: kube-system
```

Deployment

```
apiVersion: apps/v1
kind: Deployment
...
 name: kube-state-metrics
 namespace: kube-system
spec:
...
 replicas: 1
 template:
...
   spec:
     serviceAccountName: kube-state-metrics
```

User

```
kubectl config set-credentials username
--client-certificate=/root/some.crt --client-key=/root/some.key
```

Service/User Account - RoleBinding-Role-Resource

```
$~ kubectl get serviceaccounts/metrics-server -n kube-system -o yaml
...
      serviceAccount: metrics-server
...
$~  kubectl get serviceaccounts/metrics-server -n kube-system -o yaml
...
   secrets:
   - name: metrics-server-token-9dt4w

$ kubectl get ClusterRoleBinding | grep metrics-server
...
system:metrics-server                    ClusterRole/system:metrics-server                    47h

$~ kubectl get ClusterRole system:metrics-server -o yaml
```

Predefined Cluster and Namespace-localized Roles

Default ClusterRole	Default ClusterRoleBinding	Description
cluster-admin	**system:masters** group	Allows super-user access to perform any action on any resource. When used in a **ClusterRoleBinding**, it gives full control over every resource in the cluster and in all namespaces. When used in a **RoleBinding**, it gives full control over every resource in the rolebinding's namespace, including the namespace itself.
admin	None	Allows admin access, intended to be granted within a namespace using a **RoleBinding**. If used in a **RoleBinding**, allows read/write access to most resources in a namespace, including the ability to create roles and rolebindings within the namespace. It does not allow write access to resource quota or to the namespace itself.
edit	None	Allows read/write access to most objects in a namespace. It does not allow viewing or modifying roles or rolebindings.
view	None	Allows read-only access to see most objects in a namespace. It does not allow viewing roles or rolebindings. It does not allow viewing secrets, since those are escalating.

Hands On: Self-made Discovery Plugin

```
Section7/get_pods$ ls
role-binding.yaml  role.yaml  service_account.yaml  webapp_deployment.yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: getpods
rules:
- apiGroups: [""]
  resources:
  - pods
  verbs: ["get", "list","watch"]
```

```
apiVersion: rbac.authorization.k8s.io/v1
rbac.authorization.k8s.io/v1beta1
kind: RoleBinding
metadata:
  name: getpods
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: getpods
subjects:
- kind: ServiceAccount
  name: getpods
  namespace: studentXX
```

```
apiVersion: v1
kind: ServiceAccount
metadata:
 name: getpods
```

```
apiVersion: apps/v1
kind: Deployment
...
spec:
...
  template:
...
    spec:
      serviceAccountName: getpods
...
```

Howe Work 3

Home Task: https://github.com/ask4ua/DKN/blob/master/Hometask/Section3/README.md

Email: volodymyr.volkov@globallogic.com

Deadline: 1 week - Next Monday

How To Implement AD Integration

Proxying of access through some adapter like … Rancher.
Or not give access to Kuber cluster.

Kuber Cluster Maintenance Challenges

Infrastructure Deployment

Cluster Access Management

- AAA: Authentication, Authorization, ~~Accounting~~ Admission Control

Application Deployment

Kuber Application Monitoring

- Resources
- Metrics
- Logs

Application Monitoring Kubernetes

Monitoring Challenges

Kubectl is survivable, pretty scalable - so advertises new challenges into old-school service monitoring.

How are resolved:

- for cluster-wide parameters (like amount of replicas of some deployments) used centralized adapter - mostly project kube-state-metrics
- for Node-depenedent (logs, CPU, RAM monitoring) - daemonset controller (deployment with policy 1 instance per node)

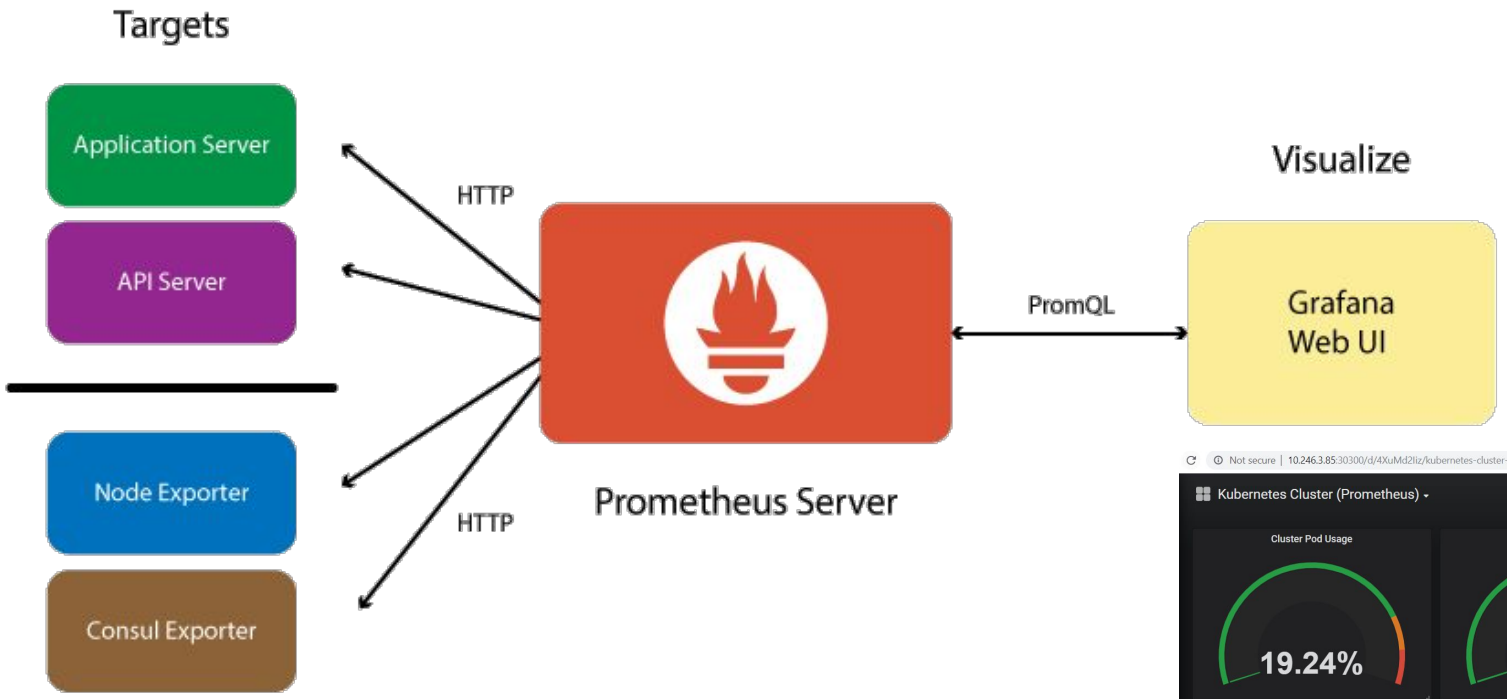
Prometheus

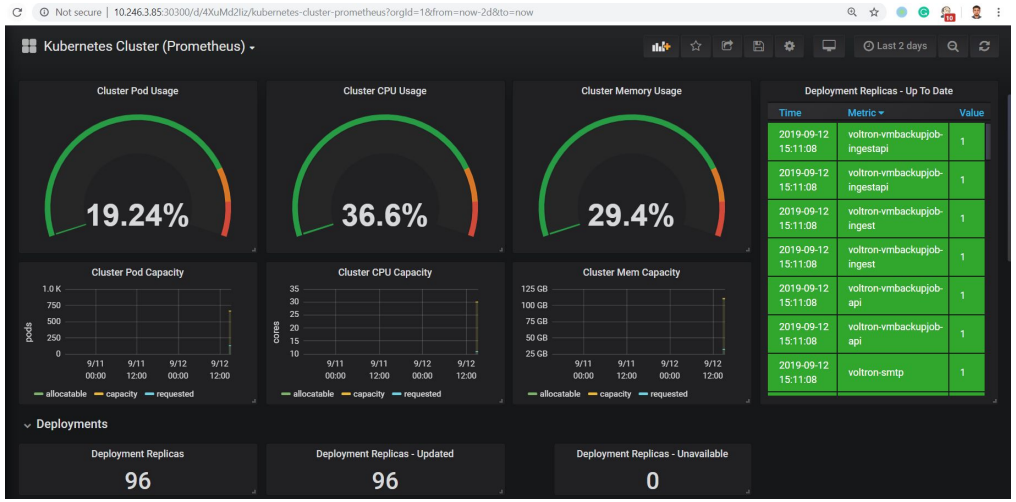
Monitoring solution working by collecting metrics via http call in key-value format.

Web-monitoring looking native for prometheus, for collecting something not web-faced used adapters named “node-exporters” - mostly distributed by different github communities.

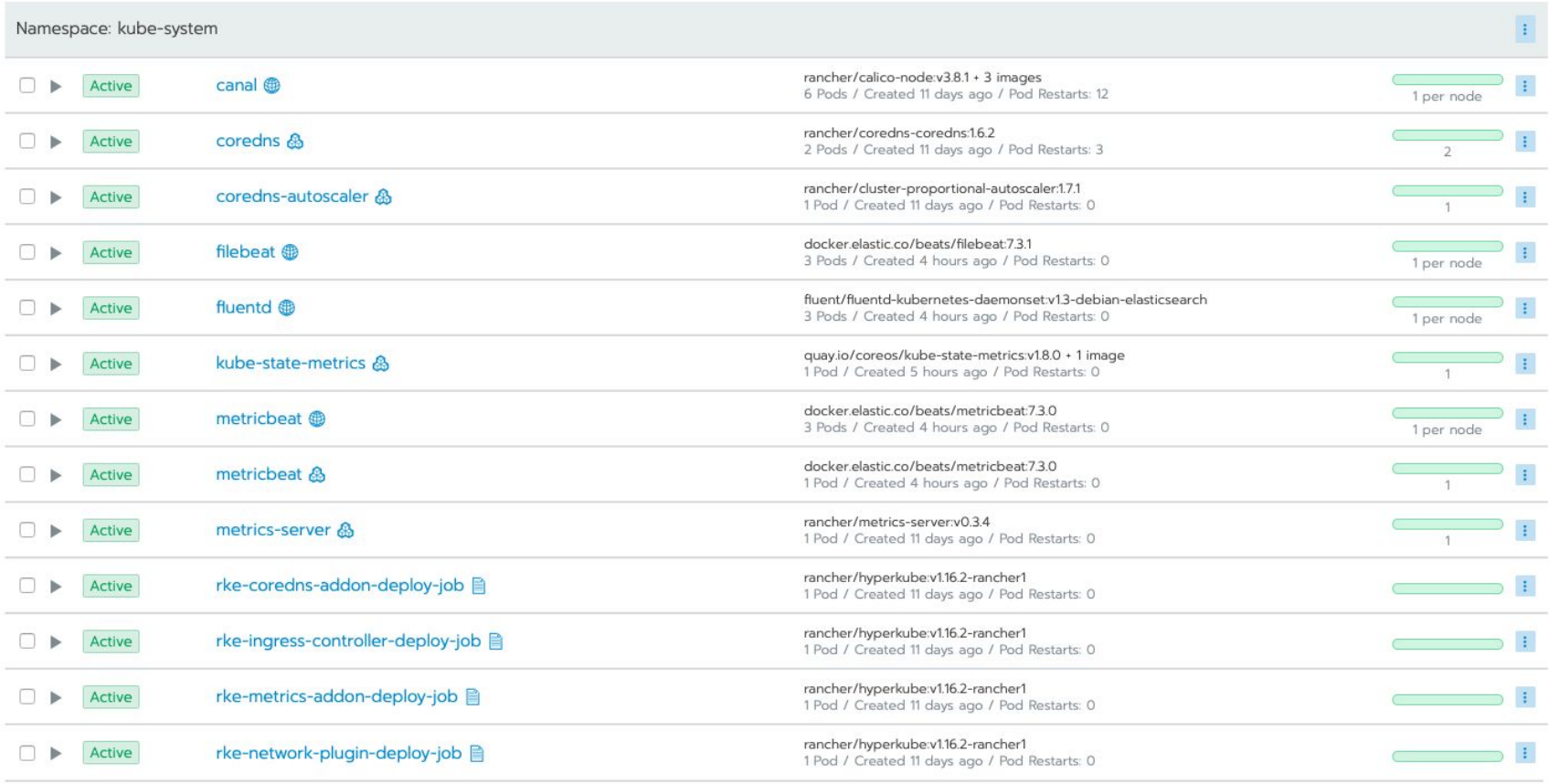
Supports quite wide types of external storage systems.
Includes inside data Rotate mechanisms.

Prometheus + Grafana on Kubernetes





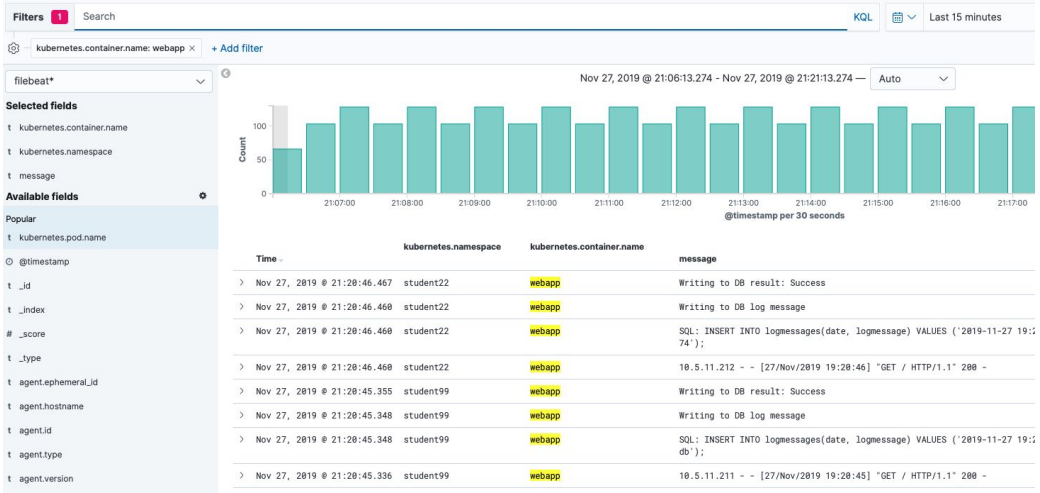
HandsOn: kube-state-metrics

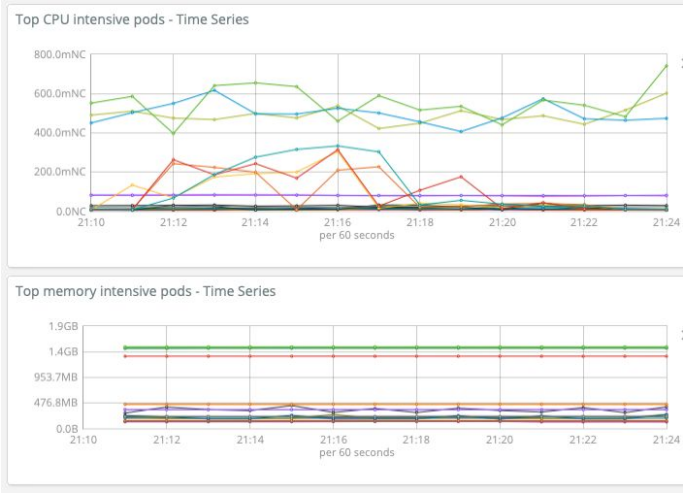


HandsOn: Elastic

https://kibana.ask4ua.com

user: admin
pass: m?9.%ABJM[JaPBZq





Istio

Istio

You add Istio support to services by deploying a special sidecar proxy throughout your environment that intercepts all network communication between microservices, then configure and manage Istio using its control plane functionality, which includes:

- Automatic load balancing for HTTP, gRPC, WebSocket, and TCP traffic.
- Fine-grained control of traffic behavior with rich routing rules, retries, failovers, and fault injection.
- A pluggable policy layer and configuration API supporting access controls, rate limits and quotas.
- Automatic metrics, logs, and traces for all traffic within a cluster, including cluster ingress and egress.
- Secure service-to-service communication in a cluster with strong identity-based authentication and authorization.

Kubernetes As A Services

Kubernetes As A Service

As example: AWS EKS service.

AWS EKS is:
- building 3x ETCD nodes and 2x Control Plane,
- managing them to be distributed in different Availibility zones,
- and patching theses nodes in-time.

Worker nodes are EC2 instances - could by any type and added on User demand. Access managed by AWS roles.

https://docs.aws.amazon.com/eks/latest/userguide/what-is-eks.html

Kubernetes As A Service

Why to use:

- AWS services integration + RBAC model: like spot nodes as Workers.

Why not to use:

- current kuber version - 18+, EKS - 14.x

Price: ~0.2$ per 1 cluster per hour (without Workers)

Kubernetes As A Service

```
$ kubectl get nodes
NAME                                          STATUS   ROLES    AGE   VERSION
ip-172-31-7-216.eu-central-1.compute.internal   Ready    <none>   46m   v1.14.7-eks-1861c5

$ kubectl get pods --all-namespaces
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE
kube-system   aws-node-mt892             1/1     Running   0          46m
kube-system   coredns-84fd7468b6-9pvkk   1/1     Running   0          57m
kube-system   coredns-84fd7468b6-z65bf   1/1     Running   0          57m
kube-system   kube-proxy-fqw2x           1/1     Running   0          46m
```

HandsOn: AWS EKS

1. Install AWS CLI, AWS-IAM authenticator, create required roles.
2. Create AWS EKS Cluster
3. Add Node Group with 1x Worker
4. Get kubeconfig via aws cli.
5. Deploy webapp test deployment

IAM Role for EKS CLuster, IAM Role for Node Group(worker), kubectl:
https://docs.aws.amazon.com/en_us/eks/latest/userguide/getting-started-console.html

AWS-IAM-Authenticator:
https://docs.aws.amazon.com/en_us/eks/latest/userguide/install-aws-iam-authenticator.html

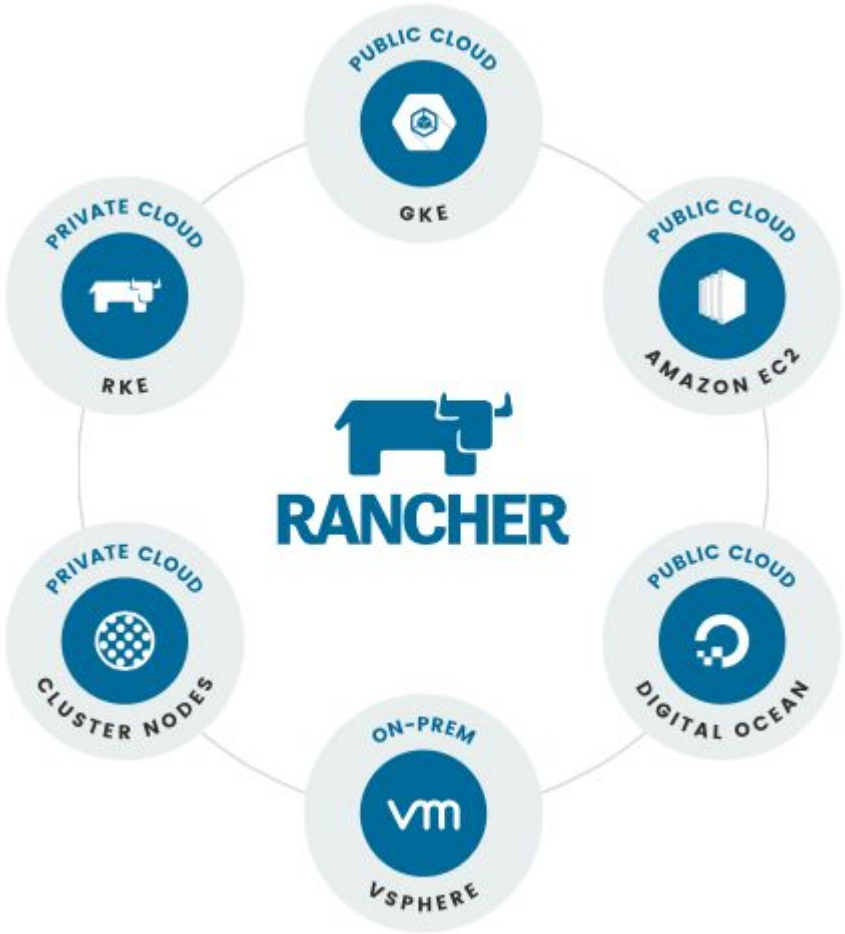
If something goes wrong:
https://docs.aws.amazon.com/en_us/eks/latest/userguide/troubleshooting.html#unauthorized

Test deployment: https://github.com/ask4ua/DKN/tree/master/Practices/Lection4

And from now you are also a kubernetes admin!)

Another Way of Kubernetes Installation

Rancher Quick Deployment



Rancher - open source project with possibility of paid support.

Includes:

- Kuber cluster deployment,
- proxying access to cluster,
- deploying app to cluster,
- set of good automation tools and

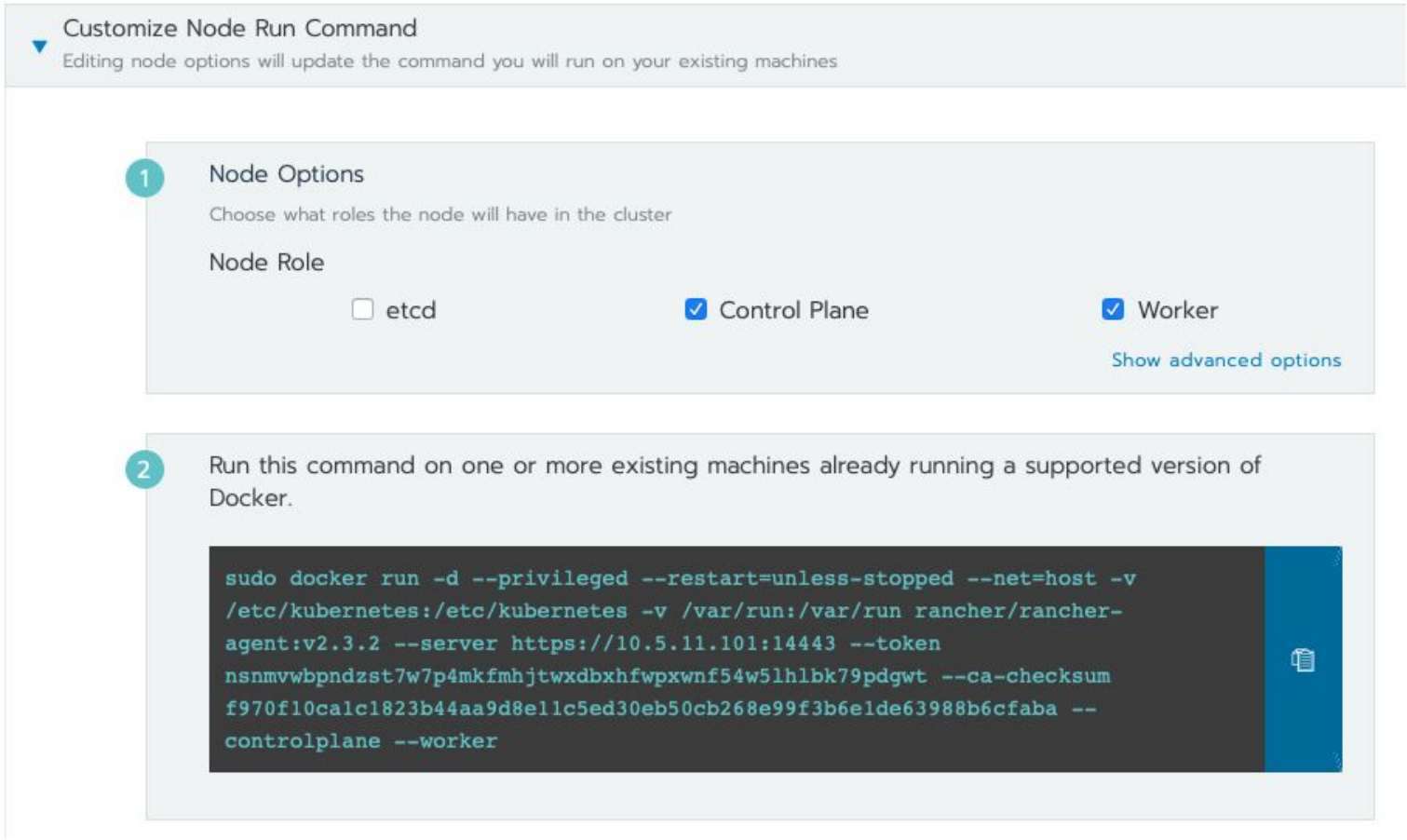
...

- and cool Web view!

Hands On: Rancher Building Cluster

https://rancher.ask4ua.com

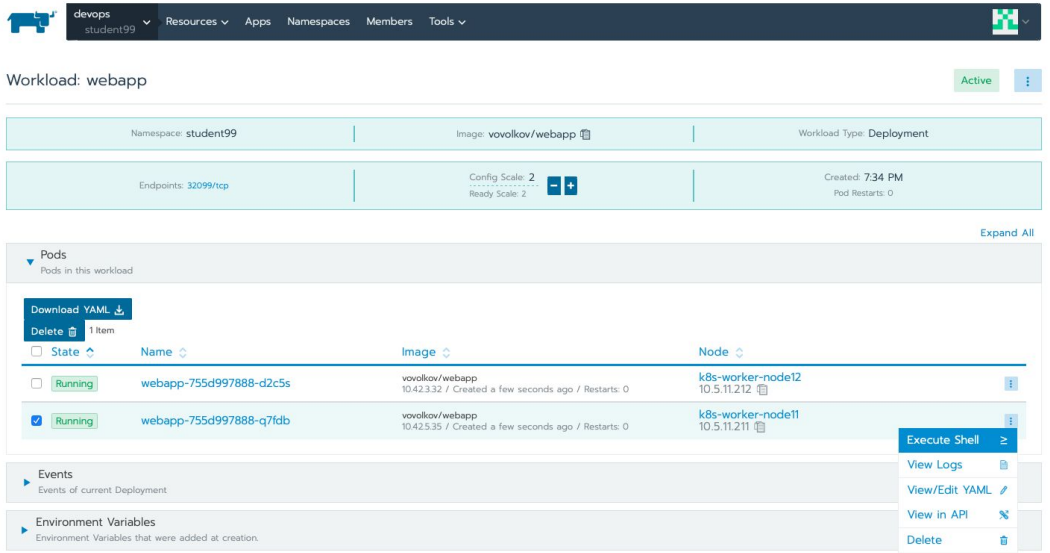
user: admin
pass: m?9.%ABJM[JaPBZq



Hands On: Kubernetes Management in Rancher Web

Steps to check rancher functionality:

1. Review deployment events.
2. Scale deployment.
3. Connect to any container CLI
4. Review pod logs.
5. Edit webapp yaml in-place,
6. Open cluster kubectl.



Kops

Kops

Kops is extending simple Kubernetes service by:

- manifesting cluster config, storing it on S3,
- manging ingress proxying woth Route53 and ACM certificates,
- checking nodes availibility from Kuber API, and spinning up lost node,
- supporting OOB autoscaling via AWS tools, usage of spot-nodes.

https://medium.com/faun/how-to-setup-a-perfect-kubernetes-cluster-using-kops-in-aws-b616bdfae013

Q&A

Home Task 1

Hash from hello: 16154e985d92e601f03e0385452c7dbdf37e0899a1f5c99106596a063d1f2ee4

1. Find image, navigate inside:
   a. `/var/lib/docker/overlay2/bcb89d77c6c6b8e98bf37a06cdae04f0d001b0821af8c1246d66f0e97d42a814/diff# sha256sum hello`
   b.
```
docker save hello-world > hello-world.tar
tar xvzf hello-world.tar; tar xvzf layer.tar
shasum -a 256 hello
```

1. Copy from container locally:

docker cp c970cc4220a8:/hello /usr/xmakshk

1. Multistage build:

```
FROM hello-world AS builder
FROM alpine
COPY --from=builder /hello .
```

```
docker build -t alpine-hello .
```

```
docker run --rm -it alpine-hello sha256sum /hello
```

1. Compile hello :P

```
curl https://raw.githubusercontent.com/docker-library/hello-world/master/hello.c  --create-dirs "./answers" --output "./answers/file_hello.c"
cd ./answers/; sha256sum file_hello.c
fda07c72f73c4609c06c31cd8359848a4000a43a36e1090052fb765cc9dd766c  file_hello.c
```

1. Injecting libs and binaries inside hello-world:

```
docker run -v /usr:/usr -v /lib64:/lib64 hello-world /usr/bin/sha256sum /hello
8b6566f585bad55b6fb9efb1dc1b6532fd08bb1796b4b42a3050aacb961f1f3f  /hello
```

Home Task 2

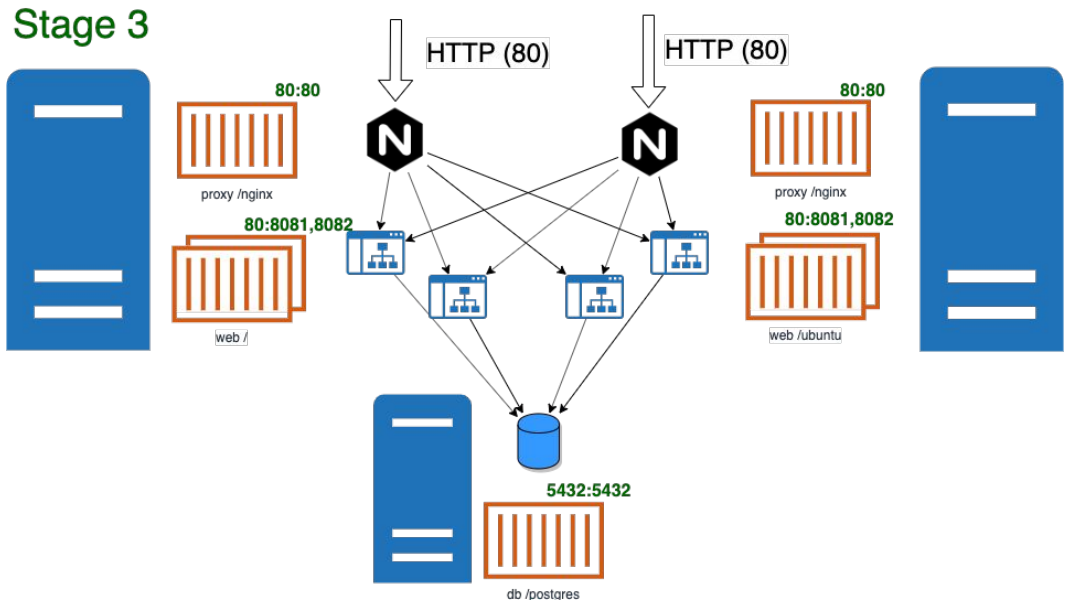
Pushing DB address:

Override secret variable:

```
sudo docker run -d -p 8081:80 --name web1 --net mynet -e DBHOST=192.168.1.117 web
```

Add DB DNS record to /etc/hosts:

```
docker run -d -p 8081:80 --add-host=db:192.168.50.4 --name web1 web
```




```
### Start nginx proxy on both Nodes
docker run --restart always -d -p 80:80 --name proxy --add-host=web1:10.17.170.196
--add-host=web2:10.17.170.30 10.17.170.1:5000/proxy

### Contents of nginx.conf

events { }
http {
    upstream webapp {
        server web1:8081;
        server web1:8082;
        server web2:8081;
        server web2:8082;

        keepalive 10;
    }

    server {
        resolver 127.0.0.11 valid=10s;

        listen 80;

        location / {
            proxy_pass http://webapp;
        }
    }
}
```