Version 2.4
04/01/21

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Prerequisites

- Modern system with at least 20G free storage
- VirtualBox installed and running
  - http://www.virtualbox.org
- Virtual machine (.ova file) installed in VirtualBox
  - https://www.dropbox.com/s/w90vuyzm0khshir/caz-classv3.ova?dl=0

  OR
  - https://bclconf.s3-us-west-2.amazonaws.com/caz-classv3.ova
- Workshop docs are in https://github.com/brentlaster/safaridocs
- Setup doc is at
  - https://github.com/brentlaster/safaridocs/blob/master/caz-setup.pdf
- Labs doc for workshop
  - https://github.com/brentlaster/safaridocs/blob/master/caz-labs.pdf

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Containers A - Z:
An overview of Containers, Docker, Kubernetes, Istio, Helm, Kubernetes Operators and GitOps

Brent Laster

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

About me

- R&D Director, DevOps
- Global trainer – training (Git, Jenkins, Gradle, CI/CD, pipelines, Kubernetes, Helm, ArgoCD, operators)
- Author -
  - OpenSource.com
  - Professional Git book
  - Jenkins 2 – Up and Running book
  - Continuous Integration vs. Continuous Delivery vs. Continuous Deployment mini-book on Safari

https://www.linkedin.com/in/brentlaster

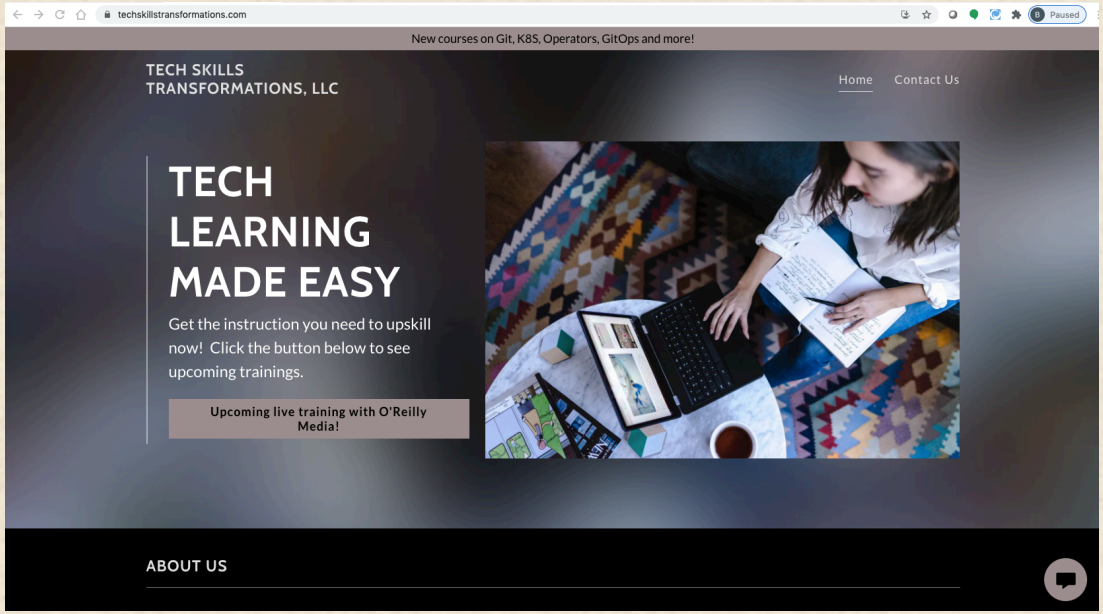
@BrentCLaster

GitHub: brentlaster

email address at end of class for additional ?'s

techskillstransformations.com

getskillsnow.com



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Book – Professional Git

- Extensive Git reference, explanations, and examples
- First part for non-technical
- Beginner and advanced reference
- Hands-on labs

Amazon Customer

I can't recommend this book more highly

February 12, 2017

Format: Kindle Edition

Brent Laster's book is in a different league from the many print and video sources that I've looked at in my attempt to learn Git. The book is extremely well organised and very clearly written. His decision to focus on Git as a local application for the first several chapters, and to defer discussion about it as a remote application until later in the book, works extremely well.

Laster has also succeeded in writing a book that should work for both beginners and people with a fair bit of experience with Git. He accomplishes this by offering, in each chapter, a core discussion followed by more advanced material and practical exercises.

I can't recommend this book more highly.

Ideal for hands-on reading and experimentation

February 23, 2017

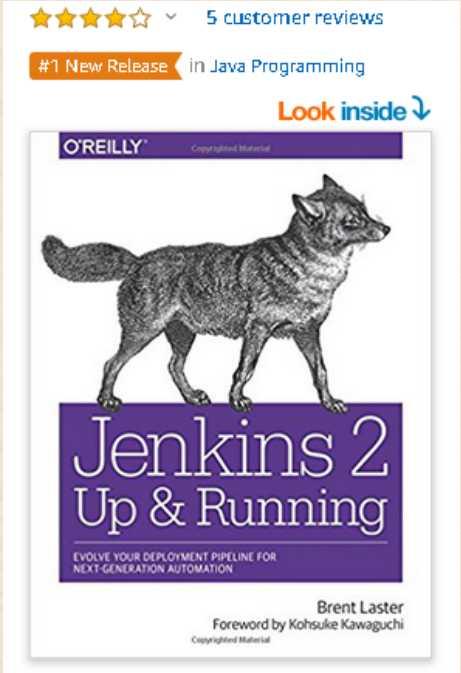
Format: Paperback | Verified Purchase

I just finished reading Professional Git, which is well organized and clearly presented. It works as both a tutorial for newcomers and a reference book for those more experienced. I found it ideal for hands-on reading and experimentation with things you may not understand at first glance. I was already familiar with Git for everyday use, but I've always stuck with a convenient subset. It was great to be able to finally get a much deeper understanding. I highly recommend the book.

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Jenkins 2 Book

- Jenkins 2 – Up and Running
- "It's an ideal book for those who are new to CI/CD, as well as those who have been using Jenkins for many years. This book will help you discover and rediscover Jenkins."
  By Kohsuke Kawaguchi, Creator of Jenkins



This is highly recommended reading for anyone looking to use Jenkins 2 to ...
By Leila on June 2, 2018
Format: Paperback

Brent really knows his stuff. I'm already a few chapters in, and I'm finding the content incredibly engaging. This is highly recommended reading for anyone looking to use Jenkins 2 to implement CD pipelines in their code.

A great resource
By Brian on June 2, 2018
Format: Paperback

I have to admit that most of the information I get usually comes through the usual outlets: stack overflow, Reddit, and others. But I've realized that having a comprehensive resource is far better than hunting and pecking for scattered answers across the web. I'm so glad I got this book!

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

O'Reilly Training

LIVE ONLINE TRAINING

Containers A-Z

An overview of containers, Docker, Kubernetes, Istio, Helm, Kubernetes Operators, and GitOps

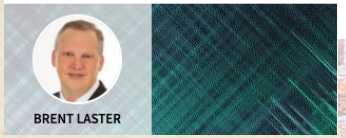
Topic: System Administration

LIVE ONLINE TRAINING

Helm Fundamentals

Deploying, upgrading, and rolli

Topic: System Administration



LIVE ONLINE TRAINING

Git Fundamentals

Simplify and speed up management of your source code

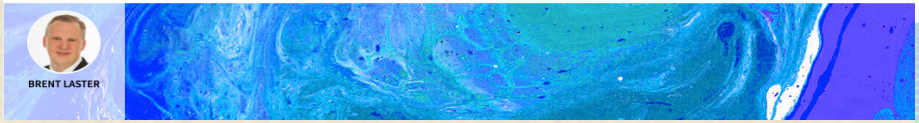
Topic: Software Development

LIVE ONLINE TRAINING

Building a Kubernetes Operator: Extending Kubernetes to Fit Your Applications

Extending Kubernetes to Fit Your Applications

Topic: System Administration

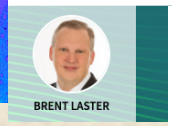


LIVE ONLINE TRAINING

Continuous Delivery in Kubernetes with ArgoCD

Automating deployment and lifecycle management

Topic: System Administration



LIVE ONLINE TRAINING

Troubleshooting Kubernetes

Learning to Identify, Understand, and Fix the Most Common Issues in the Cluster

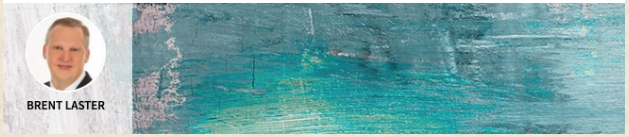
Topic: System Administration

LIVE ONLINE TRAINING

Getting started with continuous delivery (CD)

Move beyond CI to build, manage, and deploy a working pipeline

Topic: System Administration

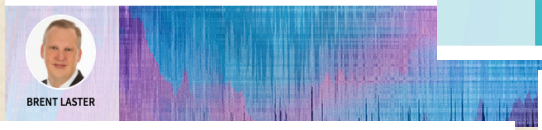


LIVE ONLINE TRAINING

Next Level Git - Master your content

Use powerful tools in Git to simplify merges, rewrite h

Topic: Software Development

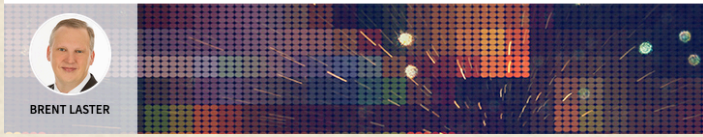


LIVE ONLINE TRAINING

Next Level Git - Master your workflow

Use Git to find problems, simplify working with multiple branches and repositories, and customize behavior with hooks

Topic: Software Development



LIVE ONLINE TRAINING

Building a deployment pipeline with Jenkins 2

Manage continuous integration and continuous delivery to release software

Topic: System Administration

LIVE ONLINE TRAINING

Git Troubleshooting

How to solve practically any problem that comes your way

Topic: Software Development

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Agenda

- Containers, images, and layers
- Docker – building/creating/composing images
- Kubernetes - clusters/objects/behavior/debugging
- Helm – releases/charts
- Istio – control/data planes/sidecar proxy/traffic shaping
- GitOps – Kubernetes CRDs, Kubernetes Operators
- Monitoring – if time allows

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

What are Containers?

- A container is a standard unit of software that functions like a fully provisioned machine installed with all the software needed to run an application.
- It's a way of packaging software so that applications and their dependencies have a self-contained environment to run in, are insulated from the host OS and other applications - and are easily ported to other environments.
- A container is NOT a VM. A container leverages several features of the Linux OS to "carve out" a self-contained space to run in.

Containers are running instances of images
Images define what goes into a container.
Containers are built from images.

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Benefits of Containers

- Easy to create and deploy once image is defined.
- Best practices of continuous development, integration, and deployment allow for frequent and reliable builds and deployments.
- Quick/easy rollbacks due to image immutability.
- Created at build/release time instead of at deployment time – so applications are decoupled from the infrastructure.
- Provide observability through application health and signals – not just from the OS.
- Because environment is self-contained, runs the same on all infrastructure – laptop, desktop, cloud, etc.
- Portable across any OS or cloud that supports containers.
- Manage application instead of infrastructure
- Allows applications such as microservices to be loosely coupled, distributed, and managed and deployed dynamically. Removes the need for monolithic stacks.
- Resource isolation and (hopefully) effective utilization.

(Paraphrased from: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/)

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

What is a container image?

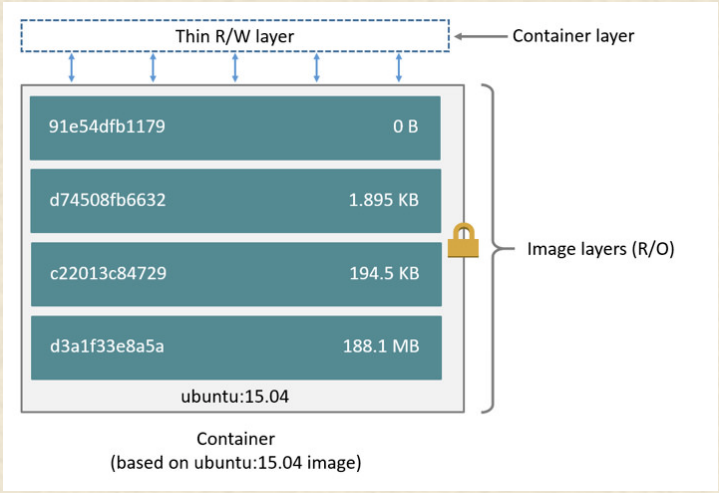
- A container image is a read-only template used to create a container.
- Container images are like a snapshot of a container.
- Images are stored in a "registry" – like repository for images.
- Images are generally considered to be "immutable". That is, once they are created, the image should not be changed. If a change is needed, a new image should be created.
- Immutability does not imply that containers can't be changed by configuration or environment, etc.
- Container images are built up from layers.
- The docker "build" command is used to create images.

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

What is an image layer?

- From https://docs.docker.com/storage/storagedriver/
  - A Docker image is built up from a series of layers.
  - Each layer results from an instruction (modification) in the Dockerfile.
  - Layers are “stacked” on top of ones before.
  - Each layer is only the set of differences from the last one.
  - Think of a base layer as being an OS image.
  - Each layer is R/O except for last one.
  - Last/top layer is “container layer”.
  - Container layer is thin R/W layer.
  - All changes made to running container go in that layer.

```
FROM ubuntu:18.04
COPY . /app
RUN make /app
CMD python /app/app.py
```



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Creating and tracking layers to images

```
# start with mysql base image (including OS)
FROM mysql:5.5.45

# copy initialization scripts into expected location
COPY docker-entrypoint-initdb.d /docker-entrypoint-initdb.d/

# establish startup scrit to initialize database
ENTRYPOINT ["/entrypoint.sh"]

# start mysql daemon running
CMD ["mysqld"]
```

```
diyuser3@training1:~$ docker history roar-db:1.0
IMAGE               CREATED             CREATED BY
19e729dca1ee        7 minutes ago       /bin/sh -c #(
24f4da862742        7 minutes ago       /bin/sh -c #(
3f00bed1e02a        7 minutes ago       /bin/sh -c #
ba16edb35eb9        3 years ago         /bin/sh -c #(
<missing>           3 years ago         /bin/sh -c #(
<missing>           3 years ago         /bin/sh -c #(
<missing>           3 years ago         /bin/sh -c #(
<missing>           3 years ago         /bin/sh -c #(
<missing>           3 years ago         /bin/sh -c mk
<missing>           3 years ago         /bin/sh -c #(
<missing>           3 years ago         /bin/sh -c ap
<missing>           3 years ago         /bin/sh -c #(
<missing>           3 years ago         /bin/sh -c #(
<missing>           3 years ago         /bin/sh -c gp
<missing>           3 years ago         /bin/sh -c ap
<missing>           3 years ago         /bin/sh -c ap
<missing>           3 years ago         /bin/sh -c mk
<missing>           3 years ago         /bin/sh -c gr
<missing>           3 years ago         /bin/sh -c #(
<missing>           3 years ago         /bin/sh -c #(
```

```
diyuser3@training1:~/qs-class/roar-docker$ docker build -t
roar-db:1.0 .
Status: Downloaded newer image for mysql:5.5.45
 ---> ba16edb35eb9
Step 2/4 : COPY docker-entrypoint-initdb.d /docker-entrypoin
t-initdb.d/
Step 3/4 : ENTRYPOINT ["/entrypoint.sh"]
 ---> Running in 9d0b805e2e90
Removing intermediate container 9d0b805e2e90
 ---> 24f4da862742
Step 4/4 : CMD ["mysqld"]
 ---> Running in 595a19547884
Removing intermediate container 595a19547884
 ---> 19e729dca1ee
Successfully built 19e729dca1ee
Successfully tagged roar-db:1.0
```

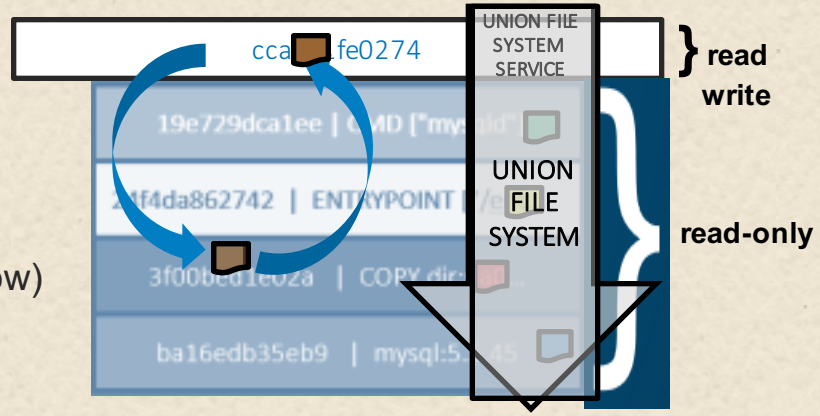
(c) 2021 Tech Skills Transformations, LLC and Brent Laster

How do layers & images intersect with the operating system?

- Layers are basically like files to the O/S
- Image layers are R/O (aka – "immutable")
- We create a container from an image via the run command. Adds another layer – a R/W one – changes can be made here ("mutable")

```
diyuser3@training1:~/qs-class/roar-istio/templates$ docker run -e MYSQL_ROOT_PASSWORD=root+1 roar-db:1.0
Running mysql_install_db
190825 19:19:16 [Note] /usr/local/mysql/bin/mysqld (mysqld 5.5.45) starting as process 59 ...
190825 19:19:17 [Note] /usr/local/mysql/bin/mysqld (mysqld 5.5.45) starting as process 65 ...
```

```
diyuser3@training1:~/qs-class/roar-k8s$ docker ps | grep roar-db
ccae61fe0274        roar-db:1.0                 "/entrypoint.sh mysq…"   9 minutes ago
      Up 9 minutes        3306/tcp
          xenodochial_merkle
```

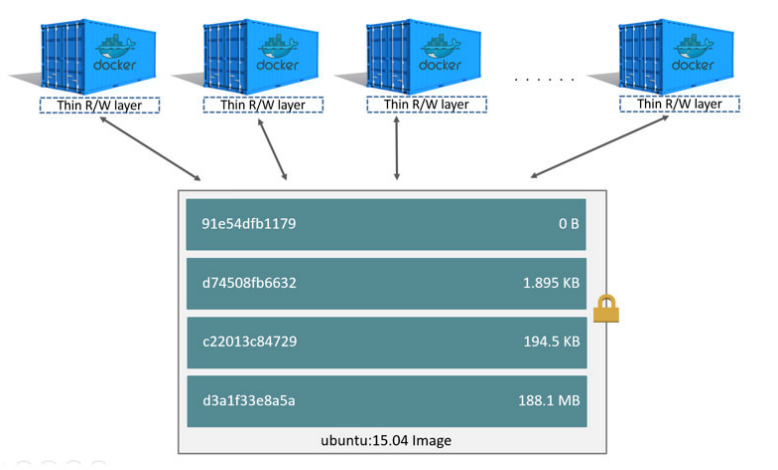


- Layers can be viewed as single union of all filesystems - Linux's Union File System allows you to stack different file systems and look through
- R/O files can be modified via Copy on Write (cow) – pulling them up to R/W layer when needed
- Other Linux facilities like cgroups and namespaces help us keep our containers separate

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Managing Content Across Containers

- From https://docs.docker.com/storage/storagedriver/
  - Major difference between a container and an image is the top writeable layer.
  - All writes (new content or modifying content) are stored in the writable layer.
  - So multiple containers can share access to the same underlying image and yet have their own data.
  - Layering also simplifies rebuilding/updating images – only layers that changed need to be updated
  - Layers stored locally on disk (usually /var/lib/docker)
  - Docker uses storage drivers to manage the contents of the image layers and the writeable layer




```
$ docker pull debian

Using default tag: latest
latest: Pulling from library/debian
fdd5d7827f33: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:e7d38b3517548a1c71e41bffe9c8ae6d6d29546ce46bf62159837aad072c90aa
Status: Downloaded newer image for debian:latest
```

Docker images can consist of multiple layers. In the example above, the image consists of two layers: `fdd5d7827f33` and `a3ed95caeb02`.

Layers can be reused by images. For example, the `debian:jessie` image shares both layers with `debian:latest`. Pulling the `debian:jessie` image therefore only pulls its metadata, but not its layers, because all layers are already present locally:

```
$ docker pull debian:jessie

jessie: Pulling from library/debian
fdd5d7827f33: Already exists
a3ed95caeb02: Already exists
Digest: sha256:a9c958be96d7d40df920e7041608f2f017af81800ca5ad23e327bc402626b58e
Status: Downloaded newer image for debian:jessie
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

What is Docker?

- (Marketing) From docker.com

  An open platform for distributed applications for developers and sysadmins.

  Open source project to ship any app as a lightweight container

- (Technical)

  Thin wrapper around Linux Container technology

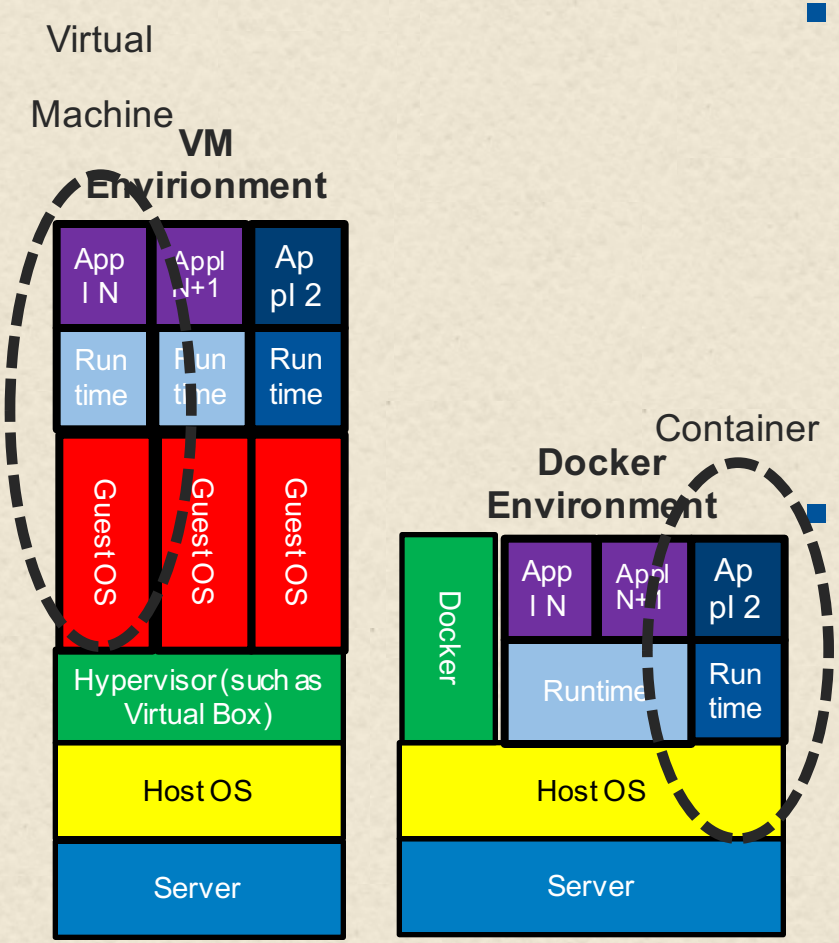
  Leverages 3 functionalities from Linux to provide isolated environment in the OS

  - » union filesystem - data
  - » namespaces - visibility (pid)
  - » cgroups - control groups (resources)

- Provides restful interface for service
- Provides description format for containers
- Provides API for orchestration

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

How Containers from Docker differ from VM



- A VM requires a Hypervisor and a Guest OS to create isolation; Docker uses Linux container technologies to run processes in separate spaces on the same OS
- Because it doesn't require the Hypervisor and a Guest OS, Docker is:
  - Faster to startup
  - More portable (can run an image unchanged in multiple environments)

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

What is a Dockerfile?

- Way to create a Docker image (a plan)
- A text file that contains (in order) all of the instructions to build an image
- Has a specific format and structure
- Each instruction in a Dockerfile can create a read-only layer

```
1  FROM tomcat:7.0.65-jre7
2
3  ARG warFile
4
5  COPY $warFile $CATALINA_HOME/webapps/roar.war
6
7  CMD ["catalina.sh", "run"]
```

```
1  FROM mysql:5.5.45
2
3
4  COPY docker-entrypoint-initdb.d /docker-entrypoint-initdb.d/
5
6  ENTRYPOINT ["/entrypoint.sh"]
7  CMD ["mysqld"]
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Dockerfile to Image to Container


```
FROM node:argon

# Create app directory
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app

# Install app dependencies
COPY package.json /usr/src/
RUN npm install

# Bundle app source
COPY . /usr/src/app

EXPOSE 8080
CMD [ "npm", "start" ]
```

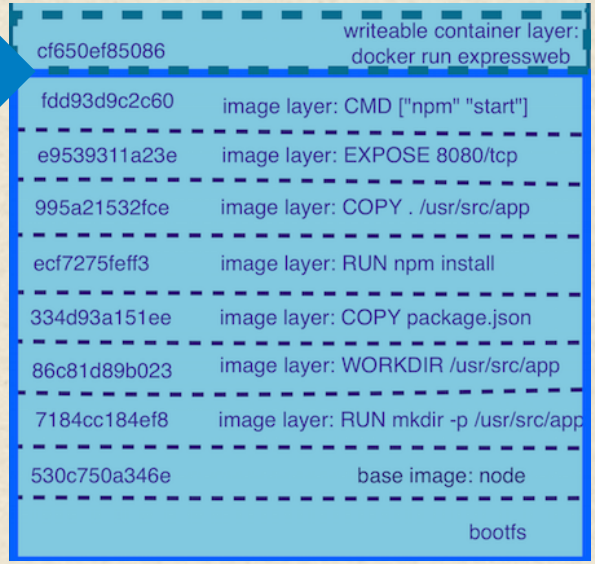
Build

```
$ docker build -t expressweb .
Step 1 : FROM node:argon
argon: Pulling from library/node...
...
Status: Downloaded newer image for node:argon
 ---> 530c750a346e
Step 2 : RUN mkdir -p /usr/src/app
 ---> Running in 5090fde23e44
 ---> 7184cc184ef8
Removing intermediate container 5090fde23e44
Step 3 : WORKDIR /usr/src/app
 ---> Running in 2987746b5fba
 ---> 86c81d89b023
Removing intermediate container 2987746b5fba
Step 4 : COPY package.json /usr/src/app/
 ---> 334d93a151ee
Removing intermediate container a678c817e467
Step 5 : RUN npm install
 ---> Running in 31ee9721cccb
 ---> ecf7275feff3
Removing intermediate container 31ee9721cccb
Step 6 : COPY . /usr/src/app
 ---> 995a21532fce
Removing intermediate container a3b7591bf46d
Step 7 : EXPOSE 8080
 ---> Running in fddb8afb98d7
 ---> e9539311a23e
Removing intermediate container fddb8afb98d7
Step 8 : CMD npm start
 ---> Running in a262fd016da6
 ---> fdd93d9c2c60
Removing intermediate container a262fd016da6
Successfully built fdd93d9c2c60
```

```
docker history <image>


$ docker history expressweb
IMAGE          CREATED     CREATED BY                      SIZE
fdd93d9c2c60   2 days ago /bin/sh -c CMD ["npm" "start"]   0 B
e9539311a23e   2 days ago /bin/sh -c EXPOSE 8080/tcp       0 B
995a21532fce   2 days ago /bin/sh -c COPY dir:50ab47bff7   760 B
ecf7275feff3   2 days ago /bin/sh -c npm install           3.439 MB
334d93a151ee   2 days ago /bin/sh -c COPY file:551095e67   265 B
86c81d89b023   2 days ago /bin/sh -c WORKDIR /usr/src/app  0 B
7184cc184ef8   2 days ago /bin/sh -c mkdir -p /usr/src/app 0 B
530c750a346e   2 days ago /bin/sh -c CMD ["node"]          0 B
```

Run



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Docker Commands

BUILD

Build an image from the Dockerfile in the current directory and tag the image
```
docker build -t myapp:1.0 .
```

List all images that are locally stored with the Docker engine
```
docker images
```

Delete an image from the local image store
```
docker rmi alpine:3.4
```

SHIP

Pull an image from a registry
```
docker pull alpine:3.4
```

Retag a local image with a new image name and tag
```
docker tag alpine:3.4 myrepo/myalpine:3.4
```

Log in to a registry (the Docker Hub by default)
```
docker login my.registry.com:8000
```

Push an image to a registry
```
docker push myrepo/myalpine:3.4
```

RUN

```
docker run
          --rm remove container automatically after it exits
            -it connect the container to terminal
     --name web name the container
      -p 5000:80 expose port 5000 externally and map to port 80
-v ~/dev:/code create a host mapped volume inside the container
     alpine:3.4 the image from which the container is instantiated
        /bin/sh the command to run inside the container
```

Stop a running container through SIGTERM
```
docker stop web
```

Stop a running container through SIGKILL
```
docker kill web
```

Create an overlay network and specify a subnet
```
docker network create --subnet 10.1.0.0/24
--gateway 10.1.0.1 -d overlay mynet
```

List the networks
```
docker network ls
```

List the running containers
```
docker ps
```

Delete all running and stopped containers
```
docker rm -f $(docker ps -aq)
```

Create a new bash process inside the container and connect it to the terminal
```
docker exec -it web bash
```

Print the last 100 lines of a container's logs
```
docker logs --tail 100 web
```

docker inspect

Estimated reading time: 2 minutes

Description

Return low-level information on Docker objects

Usage

```
docker inspect [OPTIONS] NAME|ID [NAME|ID...]
```

docker port

Estimated reading time: 1 minute

Description

List port mappings or a specific mapping for the container

Usage

```
docker port CONTAINER [PRIVATE_PORT[/PROTO]]
```

docker history

Estimated reading time: 2 minutes

Description

Show the history of an image

Usage

```
docker history [OPTIONS] IMAGE
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

About Docker Tags

- **Docker tags are user-friendly aliases for the "hex" ids of generated images**
  - Way of referring to an image
  - Like how Git tags refer to particular commits
- **Two general ways to create a tag**
  - Via docker build
    - » docker build –t IMAGE_NAME[:TAG] .

      docker build -t roar-db:v1 .
  - Via tag command
    - » docker tag IMAGE_ID IMAGE_NAME[:TAG]

      docker tag 19e729dca1ee roardb:v1.1
    - » docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]

      docker tag roar-db:v1.1 roar-db:prod
- **Tagging for private registries**
  - Private repositories run on a REGISTRYHOST
    - » Example: localhost:5000
  - Need to tag image with REGISTRYHOST info
    - » docker tag [OPTIONS] IMAGE[:TAG] [REGISTRYHOST/][USERNAME/]NAME[:TAG]

      docker tag roar-db:v1.1 localhost:5000/roar-db:v1.1
- **Using ":latest" tag**
  - "latest" does NOT mean most recent – it means "default"
  - "latest" = default tag if tag not explicitly specified
  - Avoid it and use meaningful tags instead

REPOSITORY	TAG	IMAGE ID
localhost:5000/roar-web	latest	c849a3bf4180
localhost:5000/roar-web	v1	c849a3bf4180
roar-db	1.0	19e729dca1ee
localhost:5000/roar-db	v1	19e729dca1ee
localhost:5000/roar-web	v2	d61ed3fea0c1
istio/sidecar_injector	1.2.0	c0a944dfa052
istio/proxyv2	1.2.0	81e867510ca3
istio/proxy_init	1.2.0	178ebfd48664
grafana/grafana	6.1.6	f96bf1723e2a

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

How do we think about this?

- Consider analogy of installing software on a machine…
- And then provisioning systems for users

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Docker Image Commands

- docker build – create an image from a dockerfile

  docker build [options] -t “app/container_name:tag” .

- docker images – get a list of images

  docker images –a – also show intermediate images

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Lab 1: Building Docker Images

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

How do we work with multiple containers?

- Docker compose tool allows us to start containers together
- Uses service specifications in docker-compose.yml file
- Simplifies running multiple containers that need to be "linked" together

- Composing containers together is essentially a 3-step process
  - Create needed Dockerfiles for each part of app
  - Define services that make up the app in a docker-compose.yml file
  - Run docker-compose up and it starts and runs your app

- Docker run
  - Could also use docker run command in extended form with –link option

- Docker stack
  - Command in Docker CLI - lets you mange cluster of containers via Swarm (competitor to Kubernetes)
  - Can also leverage docker-compose.yml file

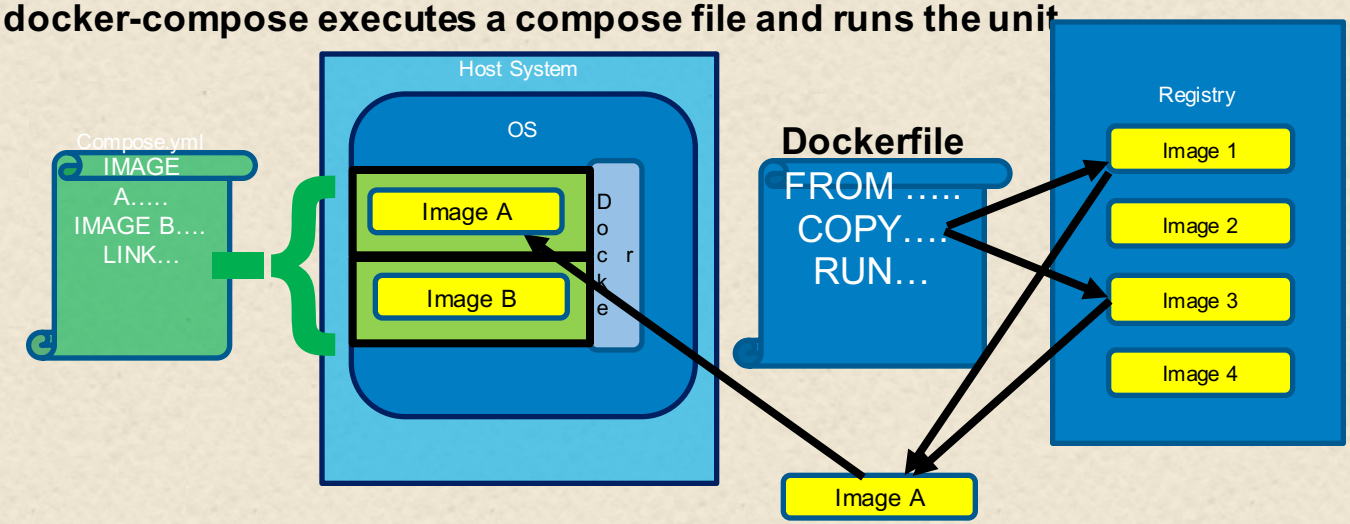
- Kubernetes or Kubernetes Environments

```
1   roar-web-container:
2       image: roar-web
3       ports:
4           - "8089:8080"
5       links:
6           - "roar-db-container:mysql"
7   roar-db-container:
8       image: roar-db
9       ports:
10          - "3308:3306"
11      environment:
12          MYSQL_USER: admin
13          MYSQL_PASSWORD: admin
14          MYSQL_DATABASE: registry
15          MYSQL_ROOT_PASSWORD: root+1
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Overall Docker Flows

- Docker images are stored in a registry
- A Dockerfile describes how to create a new image
- docker build creates the new image
- docker run starts up a container with an image
- A docker compose file describes how to create containers and link multiple ones together
- docker-compose executes a compose file and runs the unit



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Our Example App

- R.O.A.R. (Registry of Animal Responders)
  - **Simple app with two pieces**
    - Webapp on the front
    - Mysql database on the backend
  - **Written in Java**
  - **Simple REST API**
  - **War file is produced for webapp**
  - **Managed in Tomcat**

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Docker Commands

docker ps

```
$ docker ps
$ docker ps -a
$ docker kill $ID
```

Manage containers using ps/kill.

DOCKER-INSPECT MAN PAGE

docker-inspect — Return low-level information on Docker objects

Synopsis

```
docker inspect [Options] NAME|ID [NAME|ID...]
```

Description

This displays the low-level information on Docker object(s) (e.g. container, image, volume,network, node, service, or task) identified by name or ID. By default, this will render all results in a JSON array. If the container and image have the same name, this will return container JSON for unspecified type. If a format is specified, the given template will be executed for each result.

docker run

```
docker run [options] IMAGE
  # see `docker create` for options
```

Run a command in an image.

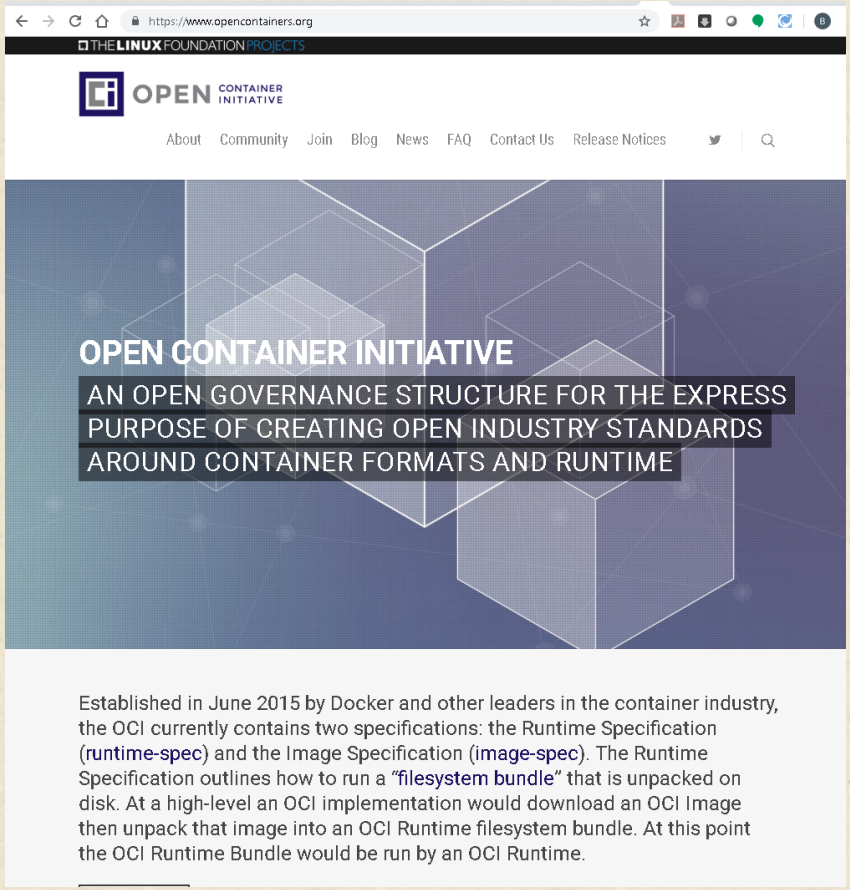
Getting information on an image

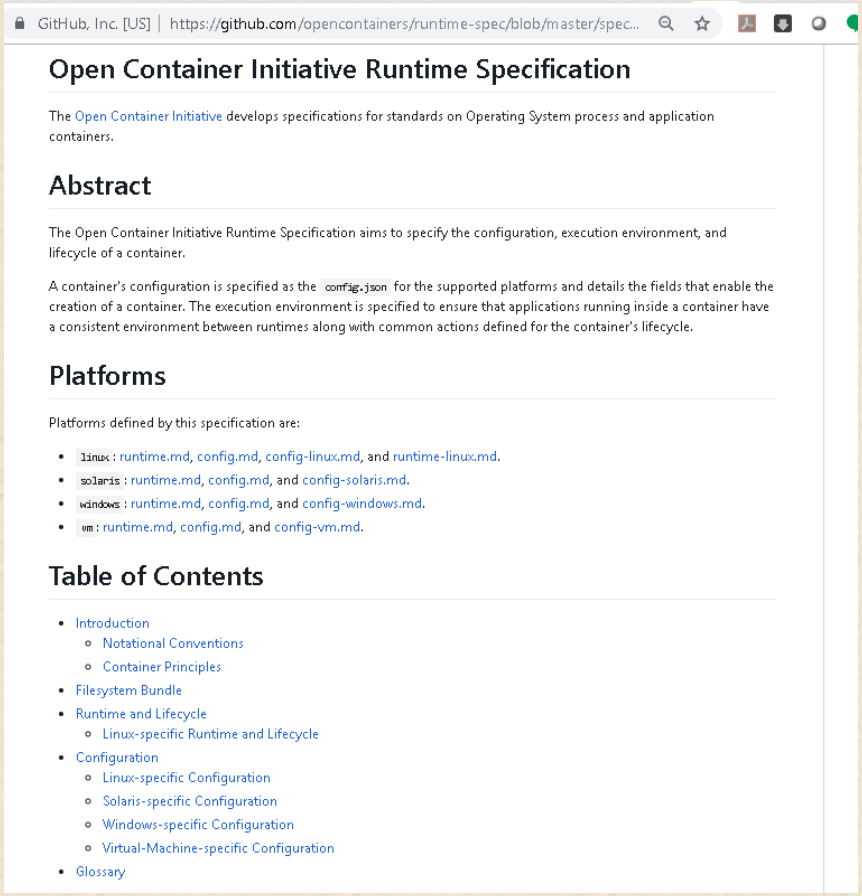
Use an image's ID or name (e.g., repository/name[:tag]) to get information about the image:

```
$ docker inspect ded7cd95e059
[{
"Id": "ded7cd95e059788f2586a51c275a4f151653779d6a7f4dad77c2bd34601d94e4",
"Parent": "48ecf305d2cf7046c1f5f8fcbcd4994403173441d4a7f125b1bb0ceead9de731",
"Comment": "",
"Created": "2015-05-27T16:58:22.937503085Z",
"Container": "76cf7f67d83a7a047454b33007d03e32a8f474ad332c3a03c94537edd22b312b",
"ContainerConfig": {
    "Hostname": "76cf7f67d83a",
    "Domainname": "",
    "User": "",
    "AttachStdin": false,
    "AttachStdout": false,
    "AttachStderr": false,
    "ExposedPorts": null,
    "Tty": false,
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Open Container Initiative (OCI)





(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Other Players in the Container Space

podman

buildah

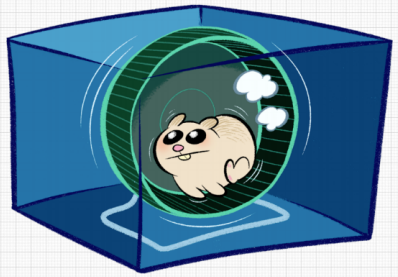
cri-o

rkt

Introducing runC

A universal runtime for OS containers

https://runc.io



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Lab 2: Composing Images Together

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

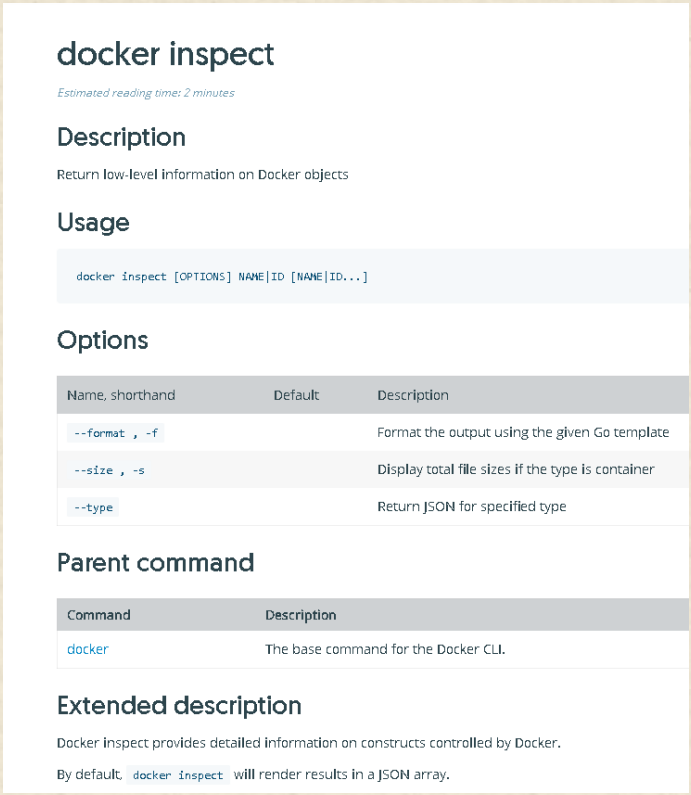
Debugging Docker containers

- How to tell what's going on – one approach
  - Identify container that is problem
  - Gather information about state - docker inspect
  - Gather information about what's happened - docker logs
  - Verify assumptions, such as what ports are exposed – docker port
  - See what's gone into image - docker history
  - Actually connect into container's filesystem -- docker exec

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Docker Inspect

- https://docs.docker.com/engine/reference/commandline



docker inspect

Estimated reading time: 2 minutes

Description

Return low-level information on Docker objects

Usage

```
docker inspect [OPTIONS] NAME|ID [NAME|ID...]
```

Options

Name, shorthand	Default	Description
`--format , -f`		Format the output using the given Go template
`--size , -s`		Display total file sizes if the type is container
`--type`		Return JSON for specified type

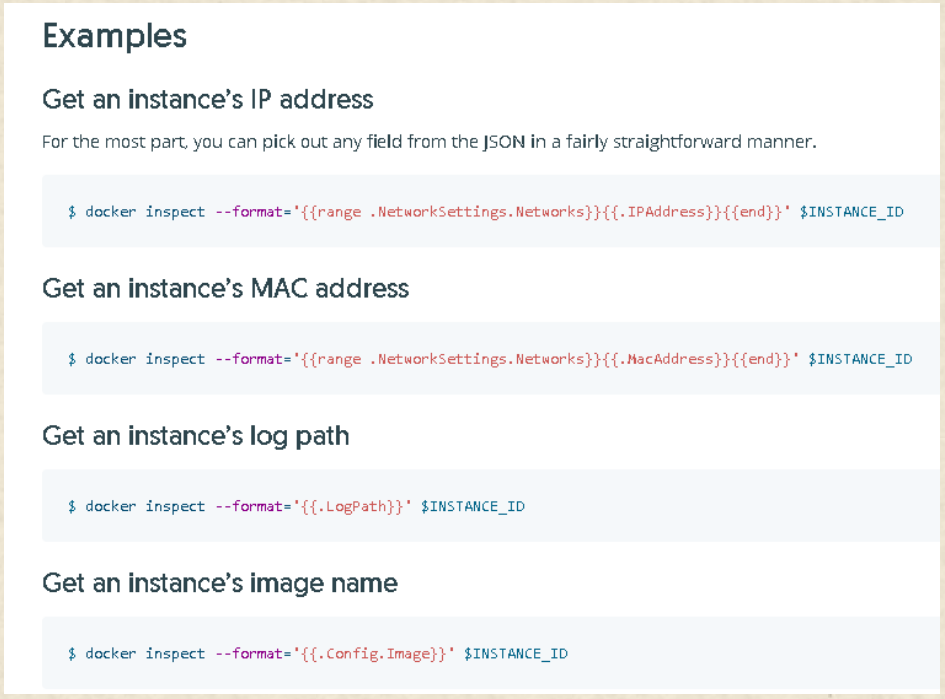
Parent command

Command	Description
docker	The base command for the Docker CLI.

Extended description

Docker inspect provides detailed information on constructs controlled by Docker.

By default, `docker inspect` will render results in a JSON array.



Examples

Get an instance's IP address

For the most part, you can pick out any field from the JSON in a fairly straightforward manner.

```
$ docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' $INSTANCE_ID
```

Get an instance's MAC address

```
$ docker inspect --format='{{range .NetworkSettings.Networks}}{{.MacAddress}}{{end}}' $INSTANCE_ID
```

Get an instance's log path

```
$ docker inspect --format='{{.LogPath}}' $INSTANCE_ID
```

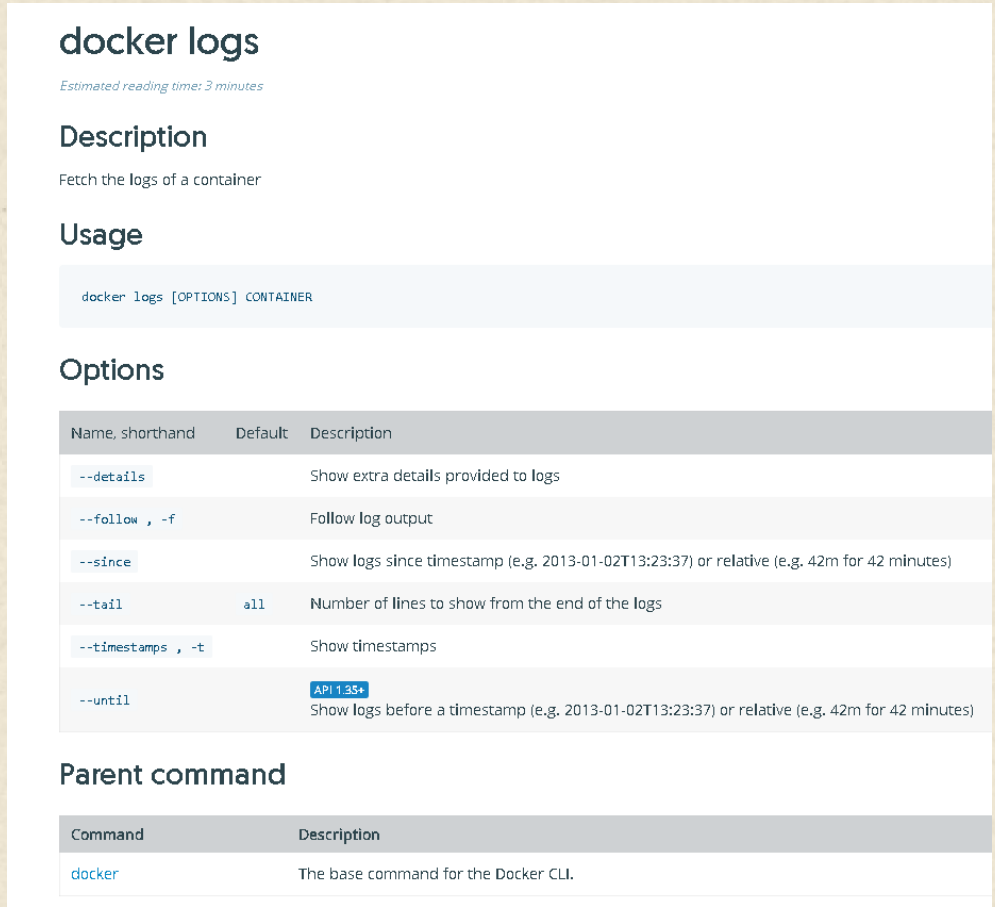
Get an instance's image name

```
$ docker inspect --format='{{.Config.Image}}' $INSTANCE_ID
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Docker Logs

- https://docs.docker.com/engine/reference/commandline



docker logs

Estimated reading time: 3 minutes

Description

Fetch the logs of a container

Usage

```
docker logs [OPTIONS] CONTAINER
```

Options

Name, shorthand	Default	Description
`--details`		Show extra details provided to logs
`--follow , -f`		Follow log output
`--since`		Show logs since timestamp (e.g. 2013-01-02T13:23:37) or relative (e.g. 42m for 42 minutes)
`--tail`	`all`	Number of lines to show from the end of the logs
`--timestamps , -t`		Show timestamps
`--until`		API 1.35+ Show logs before a timestamp (e.g. 2013-01-02T13:23:37) or relative (e.g. 42m for 42 minutes)

Parent command

Command	Description
docker	The base command for the Docker CLI.

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Docker Port

- https://docs.docker.com/engine/reference/commandline

docker port

Estimated reading time: 1 minute

Description

List port mappings or a specific mapping for the container

Usage

```
docker port CONTAINER [PRIVATE_PORT[/PROTO]]
```

Parent command

Command	Description
docker	The base command for the Docker CLI.

Examples

Show all mapped ports

You can find out all the ports mapped by not specifying a `PRIVATE_PORT`, or just a specific mapping:

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
b650456536c7        busybox:latest      top                 54 minutes ago      Up 54 minutes       0.0.0.0:1234->9876/tcp,
$ docker port test
7890/tcp -> 0.0.0.0:4321
9876/tcp -> 0.0.0.0:1234
$ docker port test 7890/tcp
0.0.0.0:4321
$ docker port test 7890/udp
2014/06/24 11:53:36 Error: No public port '7890/udp' published for test
$ docker port test 7890
0.0.0.0:4321
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Docker History

- https://docs.docker.com/engine/reference/commandline

docker history

Estimated reading time: 2 minutes

Description

Show the history of an image

Usage

```
docker history [OPTIONS] IMAGE
```

Options

Name, shorthand	Default	Description
`--format`		Pretty-print images using a Go template
`--human , -H`	`true`	Print sizes and dates in human readable format
`--no-trunc`		Don't truncate output
`--quiet , -q`		Only show numeric IDs

Examples

To see how the `docker:latest` image was built:

```
$ docker history docker

IMAGE               CREATED             CREATED BY                                      SIZE                COMMENT
3e23a5875458        8 days ago          /bin/sh -c #(nop) ENV LC_ALL=C.UTF-8            0 B
8578938dd170        8 days ago          /bin/sh -c dpkg-reconfigure locales &&    loc   1.245 MB
be51b77efb42        8 days ago          /bin/sh -c apt-get update && apt-get install    338.3 MB
4b137612be55        6 weeks ago         /bin/sh -c #(nop) ADD jessie.tar.xz in /        121 MB
750d58736b4b        6 weeks ago         /bin/sh -c #(nop) MAINTAINER Tianon Gravi <ad   0 B
511136ea3c5a        9 months ago                                                        0 B                 Imported from -
```

To see how the `docker:apache` image was added to a container's base image:

```
$ docker history docker:scm
IMAGE               CREATED             CREATED BY                                      SIZE                COMMENT
2ac9d1098bf1        3 months ago        /bin/bash                                       241.4 MB            Added Apache to
88b42ffd1f7c        5 months ago        /bin/sh -c #(nop) ADD file:1fd8d7f9f6557cafc7   373.7 MB
c69cab00d6ef        5 months ago        /bin/sh -c #(nop) MAINTAINER Lokesh Mandvekar   0 B
511136ea3c5a        19 months ago                                                       0 B                 Imported from -
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Docker Exec

- https://docs.docker.com/engine/reference/commandline

docker exec

Estimated reading time: 3 minutes

Description

Run a command in a running container

Usage

```
docker exec [OPTIONS] CONTAINER COMMAND [ARG...]
```

Options

Name, shorthand	Default	Description
`--detach , -d`		Detached mode: run command in the background
`--detach-keys`		Override the key sequence for detaching a container
`--env , -e`		API 1.25+ Set environment variables
`--interactive , -i`		Keep STDIN open even if not attached
`--privileged`		Give extended privileges to the command
`--tty , -t`		Allocate a pseudo-TTY
`--user , -u`		Username or UID (format: <name|uid>[:<group|gid>])
`--workdir , -w`		API 1.35+ Working directory inside the container

Examples

Run `docker exec` on a running container

First, start a container.

```
$ docker run --name ubuntu_bash --rm -i -t ubuntu bash
```

This will create a container named `ubuntu_bash` and start a Bash session.

Next, execute a command on the container.

```
$ docker exec -d ubuntu_bash touch /tmp/execWorks
```

This will create a new file `/tmp/execWorks` inside the running container `ubuntu_bash`, in the background.

Next, execute an interactive `bash` shell on the container.

```
$ docker exec -it ubuntu_bash bash
```

This will create a new Bash session in the container `ubuntu_bash`.

Next, set an environment variable in the current bash session.

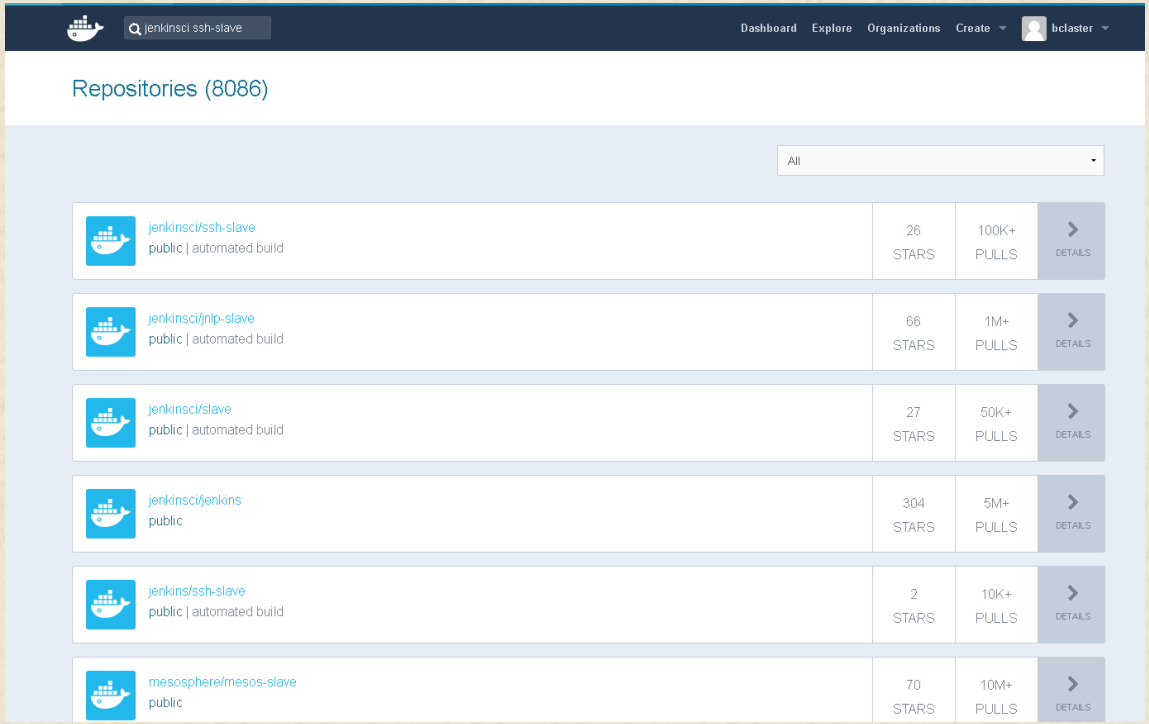
```
$ docker exec -it -e VAR=1 ubuntu_bash bash
```

This will create a new Bash session in the container `ubuntu_bash` with environment variable `$VAR` set to "1". Note that this environment variable will only be valid on the current Bash session.

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

What is a Docker Registry?

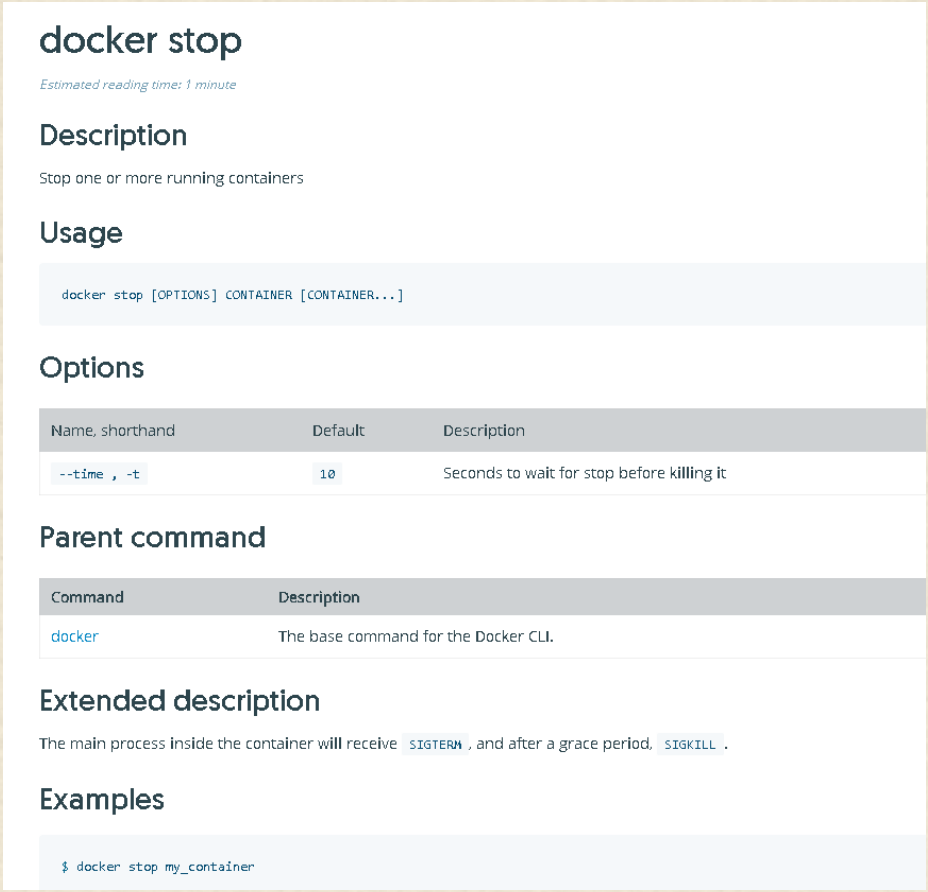
- Place to store Docker images
- Push or pull from
- Can be public or private
- Can be secure or insecure
- Public Docker registry is default (hub.docker.com)
- Private registries
  - Hosted at some path:port
  - Require images to be tagged with host info (path:port) in their name to let Docker understand which registry to work with
  - Example: if registry is at localhost:5000 then image should be localhost:5000/myimage and not just myimage



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Docker Stop

- https://docs.docker.com/engine/reference/commandline



docker stop

Estimated reading time: 1 minute

Description

Stop one or more running containers

Usage

```
docker stop [OPTIONS] CONTAINER [CONTAINER...]
```

Options

Name, shorthand	Default	Description
`--time , -t`	`10`	Seconds to wait for stop before killing it

Parent command

Command	Description
docker	The base command for the Docker CLI.

Extended description

The main process inside the container will receive `SIGTERM`, and after a grace period, `SIGKILL`.

Examples

```
$ docker stop my_container
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Docker rm

- https://docs.docker.com/engine/reference/commandline

docker rm

Estimated reading time: 2 minutes

Description

Remove one or more containers

Usage

```
docker rm [OPTIONS] CONTAINER [CONTAINER...]
```

Options

Name, shorthand	Default	Description
`--force , -f`		Force the removal of a running container (uses SIGKILL)
`--link , -l`		Remove the specified link
`--volumes , -v`		Remove the volumes associated with the container

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Docker rmi

- https://docs.docker.com/engine/reference/commandline

docker rmi

Estimated reading time: 2 minutes

Description

Remove one or more images

Usage

```
docker rmi [OPTIONS] IMAGE [IMAGE...]
```

Options

Name, shorthand	Default	Description
`--force , -f`		Force removal of the image
`--no-prune`		Do not delete untagged parents

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Lab 3: Debugging Docker Containers

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

What is Kubernetes?

- A portable, extensible platform for managing containerized workloads and services (cluster orchestration system)
- Name derived from Greek for helmsman, thus the icon
- Frequently abbreviated as "k8s"
- Formerly known as "borg" – an internal Google project
- Open-sourced by Google in 2014

- Groups containers that make up an application into logical units for easy management and discovery.
- Goal is to provide a robust platform for running many containers.
- Allows automation of deployment, scaling, and managing containerized workloads.
- Kubernetes provides you with a framework to run distributed systems (of containers) resiliently.
- Takes care of
  - scaling requirements
  - failover
  - deployment patterns

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

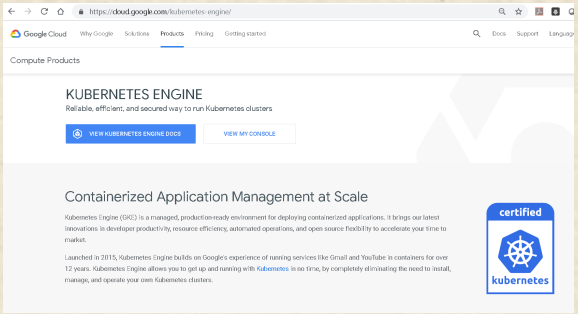
So how do we think about this?

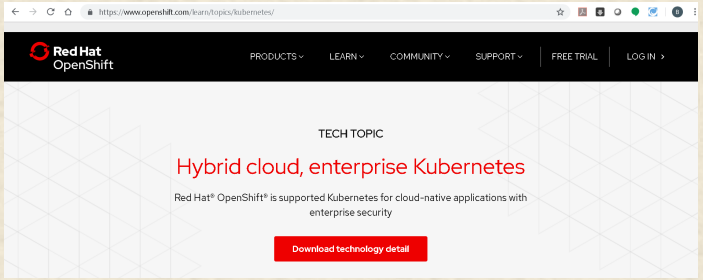
- Analogy: Datacenter for containers
  - If we think of images/containers as being like computers we stage and use
  - We can think of Kubernetes as being like a datacenter for those containers
  - Main jobs of datacenter
    - » Provide systems to service needs (regardless of the applications)
    - » Keep systems up and running
    - » Add more systems / remove systems depending on load
    - » Deal with systems that are having problems
    - » Deploy new systems when needed
    - Provide simple access to pools of systems
    - Etc..

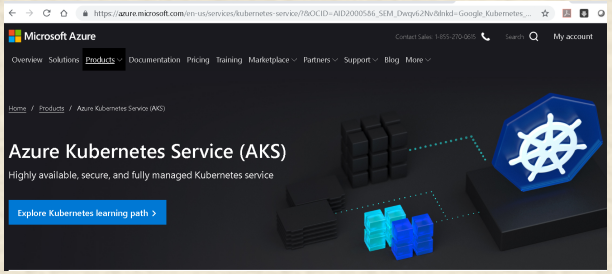
(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Kubernetes is everywhere

- Has effectively "won the war" over other competitors
  - Docker swarm
  - Mesos
- Cloud providers all endorse it and provide ways to get a Kubernetes cluster
- Implementations on other enterprise platforms







(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Kubernetes Features

- Service discovery and load-balancing
- Automated rollouts and rollbacks
- Storage orchestration
- Batch execution
- Self healing
- Secret and configuration management
- Horizontal scaling
- Automatic bin packing

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

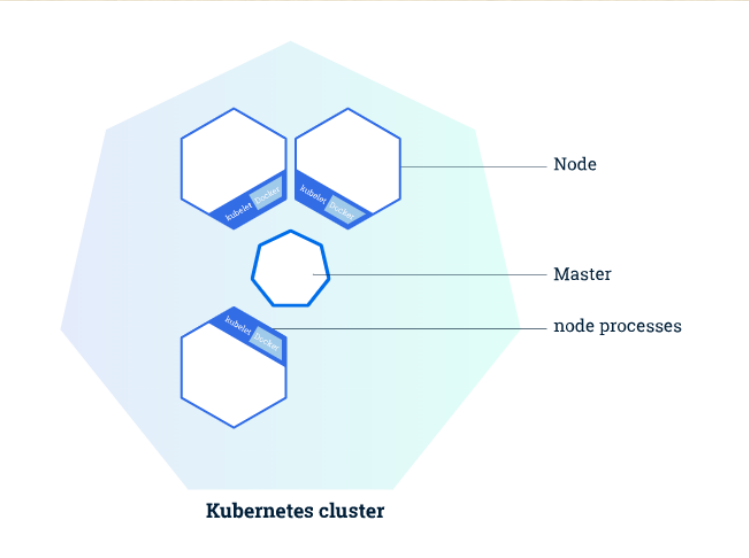
K8s Quick Terminology

- Cluster - an HA set of computers coordinated by k8s to work as a unit.
- Pods – object that contains and manages one or more containers and any attached volumes
- Service – abstraction that groups together pods based on identifiers called labels (or other characteristic)
- Deployment – defines a stateless app with a set number of pod replicas (scaled instances)
- Ingress – resource that lets cluster applications be exposed to external traffic
- Namespace - a logical area that groups k8s items like pods

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Kubernetes Clusters

- Kubernetes is about clusters
  - A k8s cluster is an HA set of computers coordinated by k8s to work as a unit.
  - Abstractions we have in k8s allow you to deploy containers in the cluster w/o tying them to a specific host.
  - K8s automates distribution and scheduling of containers across cluster in an efficient manner.
  - Nodes in a cluster are either
    - master – coordinates the work
    - nodes – run the applications
  - Simplest example is a one node cluster, such as minikube



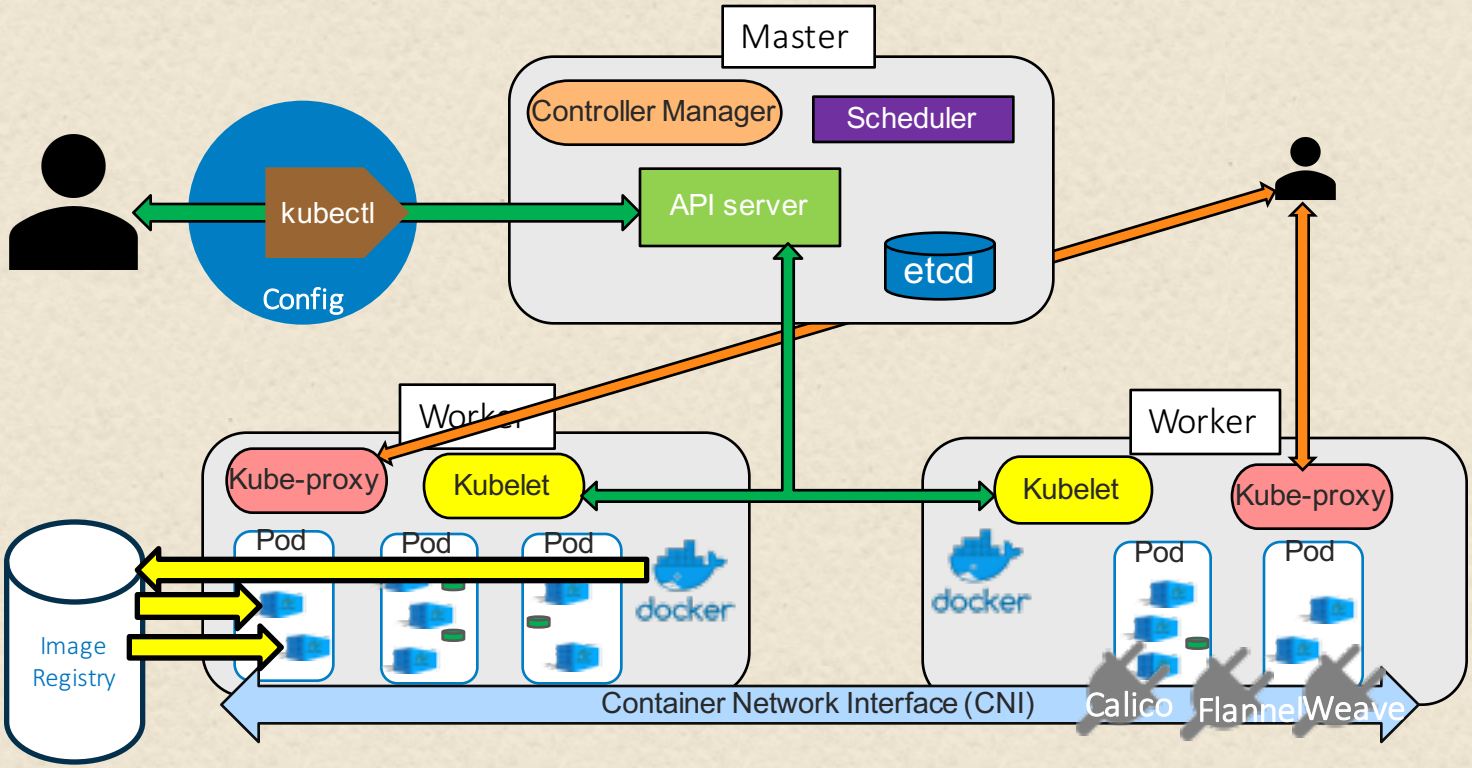
(c) 2021 Tech Skills Transformations, LLC and Brent Laster

What is minikube?

- Minikube is a simple program to learn about or work with small k8s deployments
- Creates a VM on your local machine
- Deploys a cluster containing only one node
- Has a CLI that starts and stops it, etc.
- Otherwise behaves like standard k8s

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

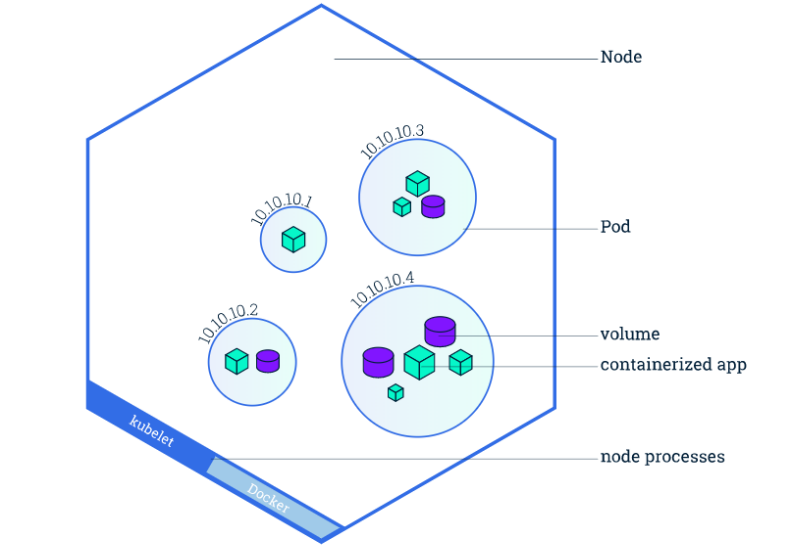
Cluster Overview



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

How are containers organized on K8S?

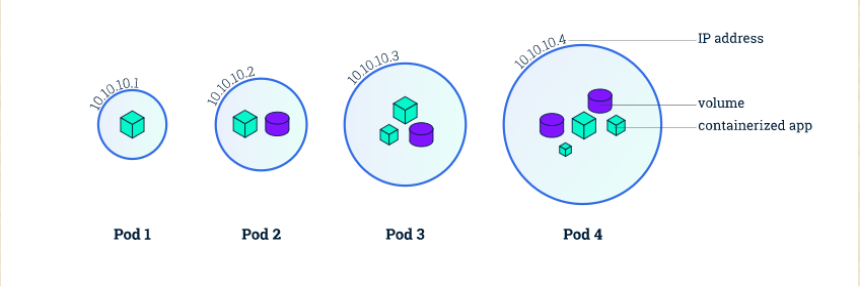
- K8s clusters have nodes
- Nodes run pods
- Pods are wrapped around one or more containers
- Pods are front-ended by services
- Pods are scaled (replicated) by deployments
- Namespaces group objects like pods, services, deployments together



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Pod

- Smallest deployable unit in Kubernetes
- Represents a group of one or more containers and any shared resources, such as
  - Volumes
  - Unique Cluster IP address
- Pods are scheduled on nodes
  - Usually automatically by master
  - Can specify nodes to run on with selector
  - Scheduling takes into account node's resources



- Containers should only be scheduled together if they are tightly couple and need to share resources such as disk
- Pod is scheduled and runs on a node
- https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Namespaces

- Logical, unique working area within a cluster
- Partitions cluster's resources
- Resources are associated/scoped to a namespace
- All resources within a namespace must be unique in name (Kubernetes can generate unique suffixes for similar resources)
- Namespaces can haveassigned quotas for resource use/limits
- Namespace "default" is default.
- Default namespace is the one used if another is not specified
- Object is abbreviated as "ns" when working with directly with it
- Option is "-n" to select other than current one
- Can set context to change current one

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

What is kubectl?

- Kubectl is the command line interface for running commands against k8s clusters
- Different pronunciations – usually either "kube control" or "kube cuttle" or "kube cuddle"
- Uses a "kube config" file to understand which cluster to work against and any necessary configuration information (kube config file may be sharable to other instantiations)
- Basic syntax is : kubectl [command] [TYPE] [NAME] flags
  - command describes operation to do on one or more resources (such as get, describe, logs, create, apply, delete, etc.)
  - TYPE is resource type – case insensitive – can specify the singular, plural, or abbreviate forms (kubectl get pod = kubectl get pods = kubectl get po)
  - NAME is the case-sensitive name of a resource (kubectl get pod ABC)
    - » If no name, gets information for all resources
  - flags – optional flags
  - Examples:
    - » kubectl get - get basic information about existence of objects
    - » kubectl apply – create/update applications from files defining resources
    - » kubectl create - create a new object
    - » kubectl describe – get detailed info about current state of an object
    - » kubectl delete – delete an object

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Kubectl Command Line Syntax

- Resources can be referred to with separate resource and name
  - kubectl describe pod mypod-name
  - Syntax from kubectl get <type>
- Resources can be referred to with resource/name syntax
  - kubectl describe pod/mypod-name
  - This syntax will be returned from a kubectl get all
- Options can be in different locations
  - kubectl -n <namespace> get all
  - kubectl get all –n <namespace>
- Can use different forms of resource type names
  - singular, plural, abbreviations (ex: pod, pods, po)
- Abbreviations exist for most resources
  - po = pod, svc = service, deploy = deployment, etc.

```
diyuser3@training1:~$ kubectl get service -n roar2
NAME       TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
mysql      ClusterIP   10.109.196.123   <none>        3306/TCP
roar-web   NodePort    10.106.132.164   <none>        8089:30318
diyuser3@training1:~$
diyuser3@training1:~$ kubectl -n roar2 get svc
NAME       TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
mysql      ClusterIP   10.109.196.123   <none>        3306/TCP
roar-web   NodePort    10.106.132.164   <none>        8089:30318
diyuser3@training1:~$
diyuser3@training1:~$ kubectl get -n roar2 all | grep service


service/mysql      ClusterIP   10.109.196.123   <none>        33
service/roar-web   NodePort    10.106.132.164   <none>        80
```

kubernetes

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

YAML and K8S specifications

- YAML is a type of markup language to define Kubernetes specs for resources
- Stored in .yaml or .yml text file
- Kubectl apply can take such a file as input and update cluster based on specs
  - Turns yaml specs into resources/objects running in cluster
- Kubectl get –o yaml can be used to dump out spec and status as yaml from running object

```
--- # Favorite movies
- Casablanca
- North by Northwest
- The Man Who Wasn't There
```

- Conventional block format uses a hyphen + space to denote a new item in a list

```
--- # Indented Block
name: John Smith
age: 33
```

- Keys are separated from values by a colon + space; indented blocks use indentation and newlines to separate key-value pairs
- Strings do not (generally) require quotation marks
- Data structure hierarchy is maintained by outline indentation

```
---
receipt:     Oz-Ware Purchase Invoice
date:        2012-08-06
customer:
    first_name:   Dorothy
    family_name:  Gale

items:
    - part_no:   A4786
      descrip:   Water Bucket (Filled)
      price:     1.47
      quantity:  4

    - part_no:   E1628
      descrip:   High Heeled "Ruby" Slippers
      size:      8
      price:     133.7
      quantity:  1
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

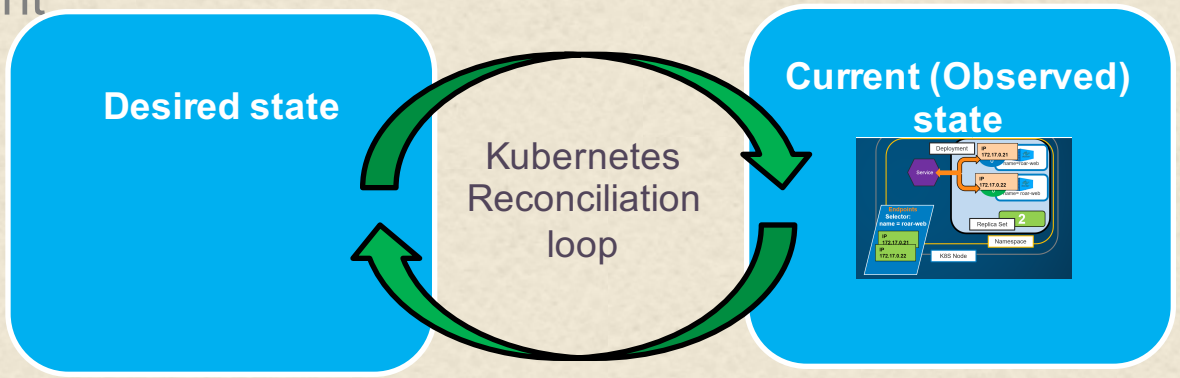
Understanding Kubernetes Objects

- To work with k8s objects, you use the k8s API
  - KubectI command-line tool makes calls to API for you
  - Could also use k8s api client libraries
- K8s objects are persistent entities in the k8s system.
- K8s uses these entities to represent the state of the cluster.
  - They can describe:
    - » What application containers are running and on which nodes.
    - » Resources available to applications.
    - » Policies around how those applications behave.
- Kubernetes object is “record of intent”
  - After creation, k8s will work to ensure object exists
  - Creating an object declares what you want cluster workload to look like
  - Known as cluster’s “desired state”
- Declarative model vs. imperative model

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Kubernetes is a Desired-State System

- User supplies desired state via declaring it in manifests
- Kubernetes works to balance the current state and the desired state
  - Desired state – what you want your production environment to be
  - Current (observed) state – current status of your production environment



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Defining a Kubernetes Object in text

- When creating an object in k8s, have to provide
  - object spec to describe desired
  - basic info, such as a name
- K8s API expects info as JSON in body of request
- But, usually provide it from YAML file
- Kubectl command line converts to JSON for you
- Required fields
  - apiVersion – which version of the k8s API is being used to create the object
  - kind - what kind of object to create
  - metadata - data that helps uniquely identify the object, such as name, UID, namespace (optional)
  - Ojbect's spec
    - » Format different for every k8s object
    - » Contains nested fields specific to the object
    - » Can find details on specs in the API reference

https://kubernetes.io/docs/concepts/overview/working-with-objects/kubernetes-objects/

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: roar-web
  labels:
    name: roar-web
  namespace: roar
spec:
  replicas: 1
  template:
    metadata:
      labels:
        name: roar-web
    spec:
      containers:
      - name: roar-web
        image: localhost:5000/roar-web:v1
        ports:
        - name: web
          containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: roar-web
  labels:
    name: roar-web
  namespace: roar
spec:
  type: NodePort
  ports:
  - port: 8089
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Kubernetes Objects - Spec and Status

- Every running k8s object includes two nested object fields
  - spec
    - » User-provided
    - » Describes desired state for the object
  - status
    - » Provided and updated by k8s system
    - » Describes the actual state of the object

```
diyuser3@training1:~$ kubectl edit -n roar2 pod roar-web-74bb47bdb8-56l4n
```

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2019-07-21T20:04:34Z"
  generateName: roar-web-74bb47bdb8-
  labels:
    app: roar-web
    pod-template-hash: 74bb47bdb8
  name: roar-web-74bb47bdb8-56l4n
  namespace: roar2
  ownerReferences:
  - apiVersion: apps/v1
    blockOwnerDeletion: true
    controller: true
    kind: ReplicaSet
    name: roar-web-74bb47bdb8
    uid: 89d00855-abf0-11e9-b30c-080027c4188a
  resourceVersion: "346085"
  selfLink: /api/v1/namespaces/roar2/pods/roar-w
  uid: c2b50783-abf2-11e9-b30c-080027c4188a
spec:
  containers:
  - image: localhost:5000/roar-web:v2
    imagePullPolicy: Always
    name: roar-web
    ports:
    - containerPort: 8080
      name: web
      protocol: TCP
```

```
    secret:
      defaultMode: 420
      secretName: default-token-dxx7t
status:
  conditions:
  - lastProbeTime: null
    lastTransitionTime: "2019-07-21T20:04:
    status: "True"
    type: Initialized
  - lastProbeTime: null
    lastTransitionTime: "2019-07-21T20:04:
    status: "True"
    type: Ready
  - lastProbeTime: null
    lastTransitionTime: "2019-07-21T20:04:
    status: "True"
    type: ContainersReady
  - lastProbeTime: null
    lastTransitionTime: "2019-07-21T20:04:
    status: "True"
    type: PodScheduled
  containerStatuses:
  - containerID: docker://41c2d31ef46b8ea1
    image: localhost:5000/roar-web:v2
    imageID: docker-pullable://localhost:5
    lastState:
      terminated:
        containerID: docker://c0ddd9686971
        exitCode: 143
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Labels

- Labels are the main mechanisms used in Kubernetes to select/organize/associate obects
- Label is a key-value pair without any pre-defined meaning
- Kubectl get has option “--show-labels” to show labels from objects

```
$ kubectl get pods -n istio1 --show-labels

NAME                           READY   STATUS    RESTARTS   AGE     LABELS
roar-current-6c75d49d78-bztfd  3/3     Running   6          6d21h   app=roar,version=current
roar-new-559997dc54-7qhjh      3/3     Running   6          6d21h   app=roar,version=new
```

- Can add label to object with “kubectl label” command

```
$ kubectl label deploy roar-new -n istio1 type=experimental
deployment.extensions/roar-new labeled
```

- We can filter based on a label
  - --selector is long form of option
  - -l is short form
- Most k8s objects support set-based selectors (choosing which items based on a set to select from). If we have pods from upper right, then we can do:

```
$ kubectl get pods -n istio1 -l 'version in (current, new)'

NAME                           READY   STATUS    RESTARTS   AGE
roar-current-6c75d49d78-bztfd  3/3     Running   6          6d21h
roar-new-559997dc54-7qhjh      3/3     Running   6          6d21h
```

- Labels can apply to other objects, such as nodes and services and be used in other operations

```
$ kubectl delete pods -l 'version in (current, new)'
```

- Types: “Equality-based” (=, !=) “Set-based” (in ())

```
apiVersion: v1
kind: Pod
metadata:
  name: roar-current
  labels:
    app: roar
    version: current
spec:
  containers:
  - image: localhost:5000/roar-web:v1
    imagePullPolicy: Always
    name: roar-web
    ports:
    - containerPort: 8080
      name: roar-web
      protocol: TCP
    resources: {}
```

```
apiVersion: v1
kind: Pod
metadata:
  name: roar-new
  labels:
    app: roar
    version: new
spec:
  containers:
  - image: localhost:5000/roar-web:v2
    imagePullPolicy: Always
    name: roar-web
    ports:
    - containerPort: 8080
      name: roar-web
      protocol: TCP
    resources: {}
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Deployments

- Spinning Up Pods
  - A deployment is like a pod “supervisor”.
  - After initiating a deployment, we have the deployment, the replicaset, and the pods
  - Naming of the pods and replicaset are derived from the deployment name

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: roar-web
  labels:
    name: roar-web
  namespace: roar
spec:
  replicas: 1
  template:
    metadata:
      labels:
        name: roar-web
    spec:
      containers:
      - name: roar-web
        image: localhost:5000/roar-web:v1
        ports:
        - name: web
          containerPort: 8080
```

```
diyuser3@training1:~$ k get deploy -n roar  | grep web
roar-web   1/1     1            1           6d23h
diyuser3@training1:~$ k get rs  -n roar  | grep web
roar-web-f95cf6574   1         1         1       6d23h
diyuser3@training1:~$ k get pod  -n roar  | grep web
roar-web-f95cf6574-ndt4k   1/1     Running   0          9m44s
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Deployments

- Updating Pods
  - What happens if we make a significant change (update image to v2) and apply it to the running instance?
  - kubectl apply –f roar-complete.yaml –n roar
  - Pods with old versions are terminated and new ones created with the updated version

```
$ kubectl get pods -n roar -w

NAME                        READY   STATUS    RESTARTS   AGE
mysql-8559ccc47d-crvhv      1/1     Running   0          4h22m
roar-web-f95cf6574-wcm7z    1/1     Running   0          15s
roar-web-7bccc45c79-pp9hs   0/1     Pending   0          0s
roar-web-7bccc45c79-pp9hs   0/1     Pending   0          0s
roar-web-f95cf6574-wcm7z    1/1     Terminating   0          50s
roar-web-7bccc45c79-pp9hs   0/1     ContainerCreating   0          0s
roar-web-f95cf6574-wcm7z    0/1     Terminating         0          53s
roar-web-7bccc45c79-pp9hs   1/1     Running             0          3s
```

  - And new replicaset is created

```
$ k get rs -n roar

NAME                  DESIRED   CURRENT   READY   AGE
mysql-8559ccc47d      1         1         1       6d22h
roar-web-7bccc45c79   1         1         1       55s
roar-web-f95cf6574    0         0         0       7d5h
```

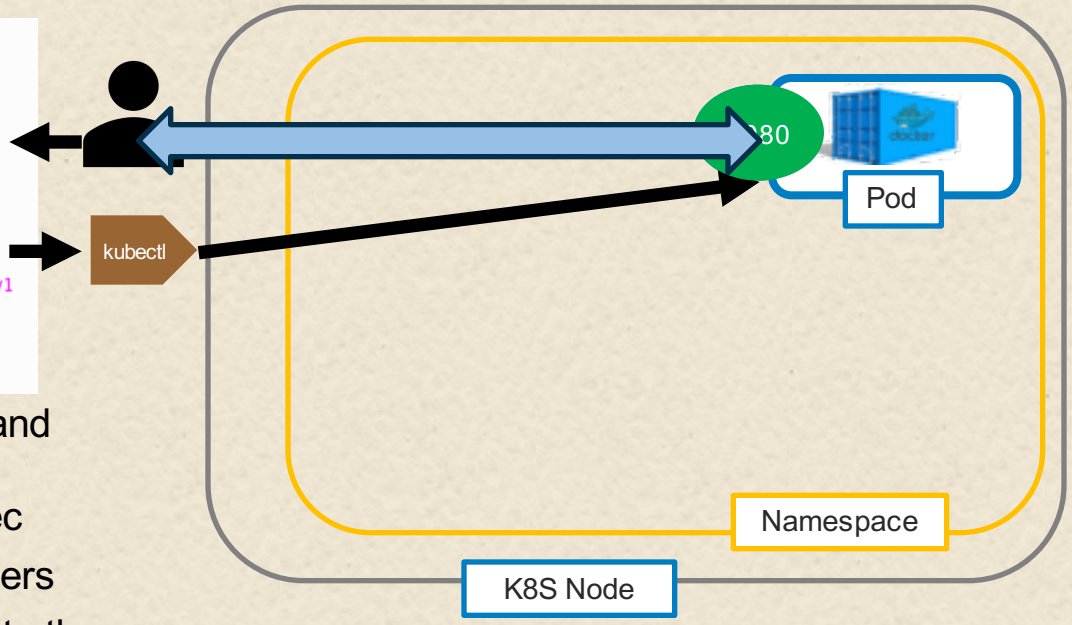
```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: roar-web
  labels:
    name: roar-web
    type: experimental
  namespace: roar
spec:
  replicas: 1
  template:
    metadata:
      labels:
        name: roar-web
    spec:
      containers:
      - name: roar-web
        image: localhost:5000/roar-web:v2
        ports:
        - name: web
          containerPort: 8080
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Deployments vs. Pods

- Using Pods without Deployments

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: roar-web
  labels:
    name: roar-web
  namespace: roar
spec:
  containers:
  - name: roar-web
    image: localhost:5000/roar-web:v1
    ports:
    - name: web
      containerPort: 8080
```



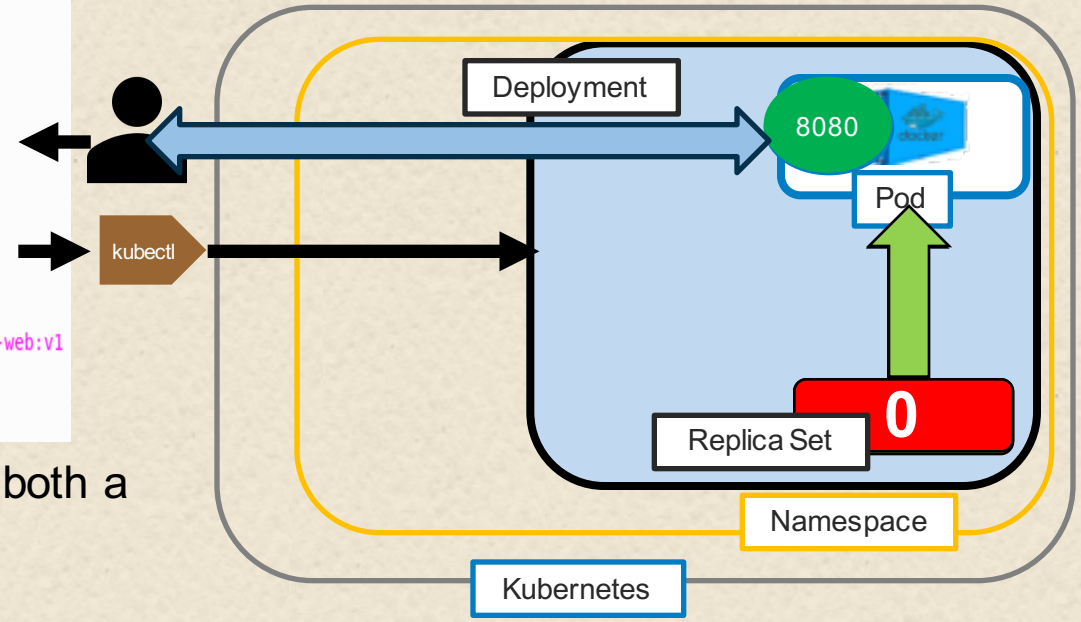
- Objects run within node and namespace
- Objects created from spec
- Pods encompass containers
- Ways to connect directly to them
- If pod goes away (and still need functionality), must manually start a new one
- Different ip address on new one

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Deployments vs. Pods

- Pods with Deployments

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: roar-web
  labels:
    name: roar-web
  namespace: roar
spec:
  replicas: 1
  template:
    metadata:
      labels:
        name: roar-web
    spec:
      containers:
      - name: roar-web
        image: localhost:5000/roar-web:v1
        ports:
        - name: web
          containerPort: 8080
```

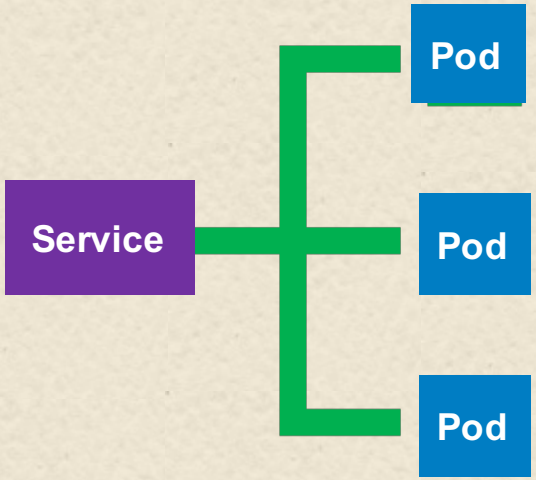


- Deployments include both a replicaset and pods
- Specs for both
- If pod goes away, replicaset starts up a new one automatically – up to number of replicas specified
- Different ip address on new one

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Services

- A single virtual, stable connection for multiple volatile pods
  - Kubernetes service provides a level of abstraction for pods
  - Pods may be "volatile" so can't rely on their respective IP addresses to connect to long-term
  - A service provides a virtual, consistent IP to connect to for a set of pods
  - Pods can be selected by labels
  - Virtual IP of a service is not connected to an actual network interface
  - Virtual IP exists to forward traffic to one or more pods
  - Kube-proxy process (running on every node) is responsible for keeping the mapping between the virtual IP and the pods up-to-date
  - Kube-proxy queries the API server to learn about new services

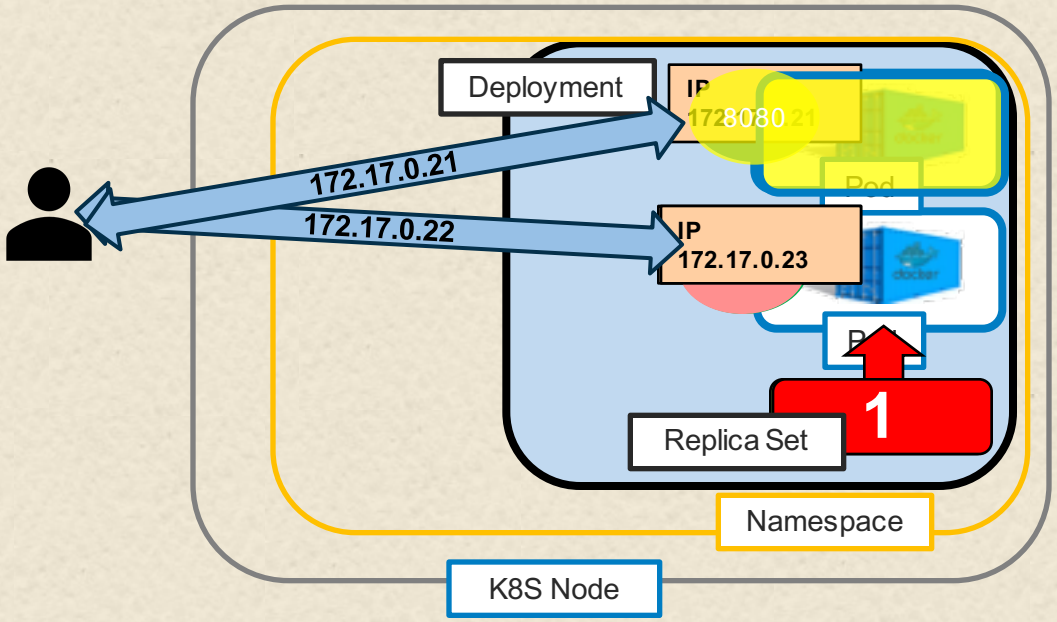


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Services

- Without Services

- We can connect to individual pods (if allowed)
- If in deployment and pod goes away, we lose access to that pod
- ReplicaSet in deployment will spin up another one but won't have same address
- Similar situation if a pod becomes unusable
- - may have other pods that could handle needs, but no way to automatically discover or connect



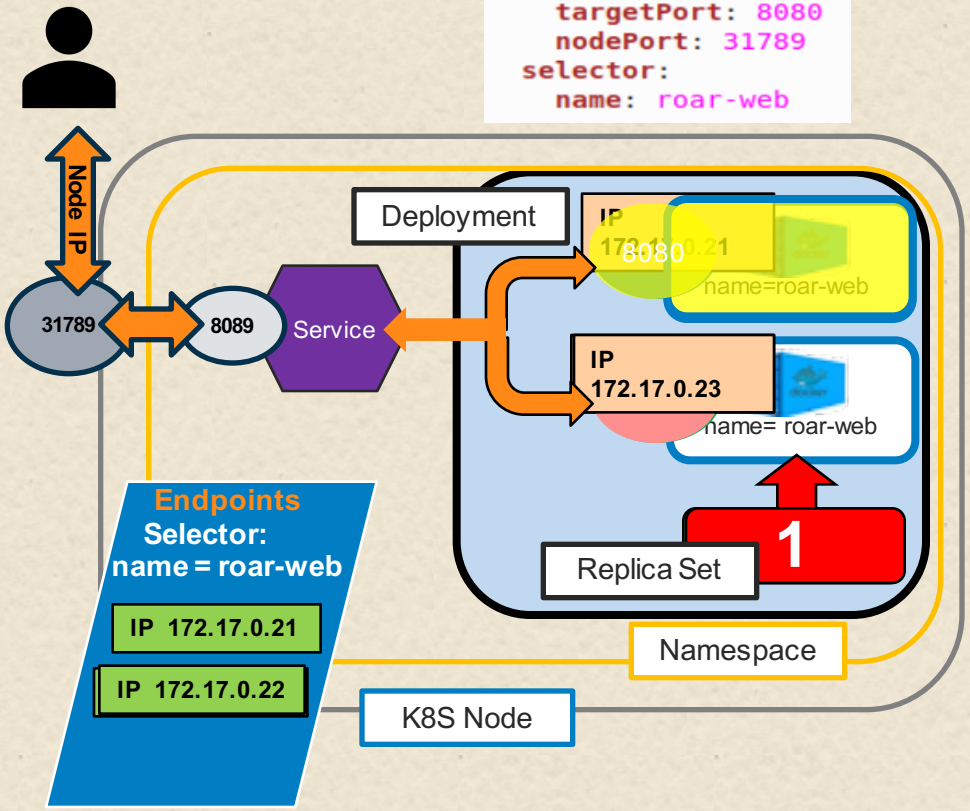
(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Services

- With a service

- Service provides virtual address for us to connect to for frontend
- Uses labels to select “pool” of backend pods to map to
- Endpoints for a service are list of available ip addresses for usable pods
- Example service here is NodePort
- If one pod goes down or becomes unavailable, service can connect to another pod
- Spec is shown below

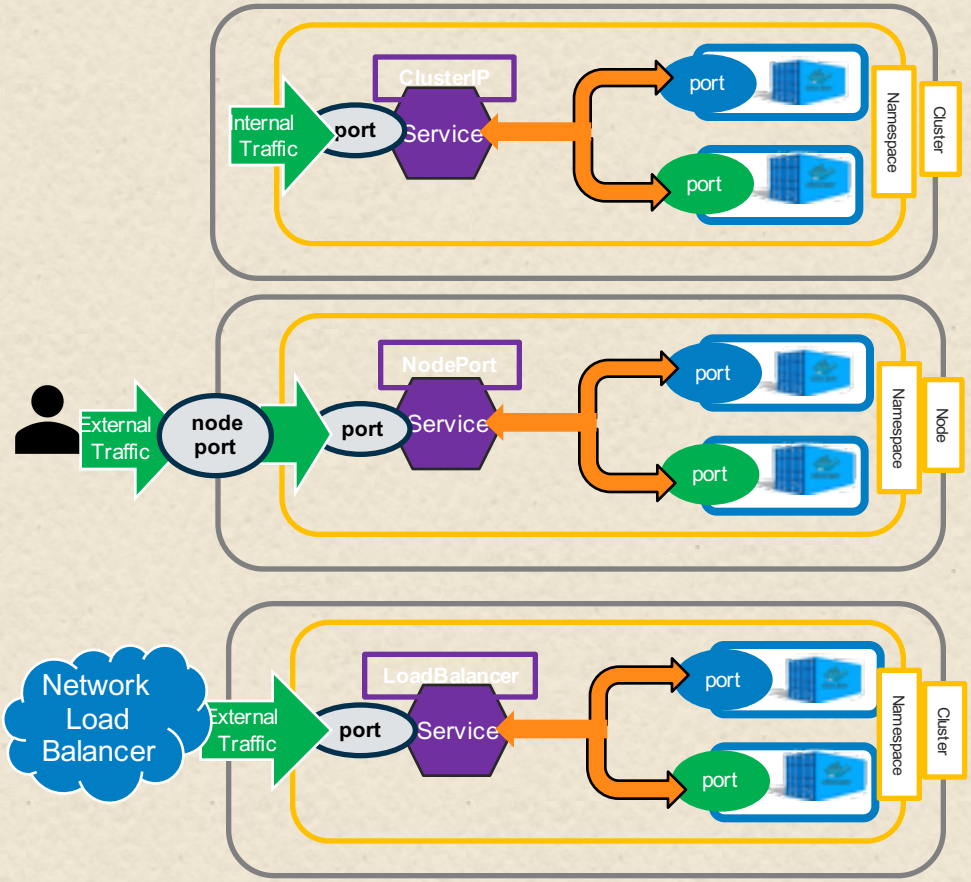
```
apiVersion: v1
kind: Service
metadata:
  name: roar-web
  labels:
    name: roar-web
  namespace: roar2
spec:
  type: NodePort
  ports:
  - port: 8089
    targetPort: 8080
    nodePort: 31789
  selector:
    name: roar-web
```



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Types of Services

- ClusterIP
  - Default K8S service
  - Provides service inside cluster for other apps in cluster to access
  - No external access (except via kubectl proxy for debugging, etc.)
- NodePort
  - Most basic way to get external traffic to service
  - Opens up a port on the K8S node
  - Any traffic going to that port on the node is forwarded to the service
  - Uses node IP address
  - Only ports in range 30000-32767
- LoadBalancer
  - Gives you Load Balancer with single IP that forwards all traffic to service
  - Default method for directly exposing a service
  - No filtering, no routing
  - Cost factor on clouds
- ExternalName
  - Maps service to contents of an external name field (foo.bar.com)
  - Returns a CNAME record with that value



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

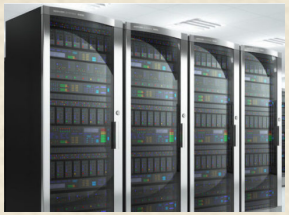
Data Center Analogy

- Functions: Uptime, scaling, redundancy…

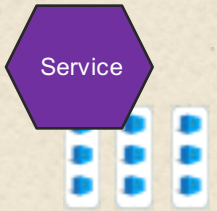
  - Container in a pod ~ server in a rack


  - Pod ~ rack of servers


  - Deployment ~ multiple racks (replicas)

  - Service ~ central control / login server



  - Namespace ~ server room

70

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Lab 4: Exploring and Deploying into Kubernetes

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Configmaps and Secrets

- Approaches for configuring applications
- Could specify environment variables in Dockerfile or k8s yaml file
- Bad part of environment variables is they are tied to container or deployment.
- To change them, must modify container or redo deployment.
- Data must also be duplicated across objects.

- Alternative is secrets for confidential data and configmaps for non-confidential data
- Difference between Secrets and ConfigMaps are that Secrets are obfuscated with a Base64 encoding.
- Can be created like any other objects in Kubernetes.

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Secrets

- Examples
  - Environment variables set up in spec for admin and root passwords
  - Create secrets for both
    - Create via command line

☐ ☐k create secret generic mysqlsecret2 --from-literal=mysqlpassword=admin --from-literal=mysqlrootpassword=root+1 -n roar

    - Create via yaml file

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysqlsecret
type: Opaque
data:
  mysqlpassword: YWRtaW4=
  mysqlrootpassword: cm9vdCsx
```

  - Update spec to use those

```
        env:
        - name: MYSQL_DATABASE
          value: registry
        - name: MYSQL_PASSWORD
          value: admin
        - name: MYSQL_ROOT_PASSWORD
          value: root+1
        - name: MYSQL_USER
          value: admin
```

```
      env:
        - name: MYSQL_DATABASE
          value: registry
        - name: MYSQL_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysqlsecret
              key: mysqlpassword
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysqlsecret
              key: mysqlrootpassword
        - name: MYSQL_USER
          value: admin
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Configmaps

- Examples
  - Environment variables set up in spec for database and user
  - Create configmap for both
    - Create via command line

☐k create configmap mysql-configmap --from-literal=mysql.database=registry --from-literal=mysql.user=admin -n roar To change them, must modify container

OR

- Create via yaml file

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql-configmap
data:
  mysql.database: registry
  mysql.user: admin
```

- Update spec to use those

```
        env:
          - name: MYSQL_DATABASE
            value: registry
          - name: MYSQL_PASSWORD
            valueFrom:
              secretKeyRef:
                name: mysqlsecret
                key: mysqlpassword
          - name: MYSQL_ROOT_PASSWORD
            valueFrom:
              secretKeyRef:
                name: mysqlsecret
                key: mysqlrootpassword
          - name: MYSQL_USER
            value: admin
```

```
        env:
          - name: MYSQL_DATABASE
            valueFrom:
              configMapKeyRef:
                name: mysql-configmap
                key: mysql.database
          - name: MYSQL_PASSWORD
            valueFrom:
              secretKeyRef:
                name: mysqlsecret
                key: mysqlpassword
          - name: MYSQL_ROOT_PASSWORD
            valueFrom:
              secretKeyRef:
                name: mysqlsecret
                key: mysqlrootpassword
          - name: MYSQL_USER
            valueFrom:
              configMapKeyRef:
                name: mysql-configmap
                key: mysql.user
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Lab 5: Working with Kubernetes Secrets and ConfigMaps

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Kubernetes Storage Concepts

- Container filesystem
  - Storage at the level of a container
    - » Ephemeral – only exists while container is running
    - » Any changes made are gone when container is gone
- Volume
  - Storage at the level of a pod
    - » Data is preserved if a container crashes
    - » Allows sharing of data across containers in a pod
    - » Only exists while pod exists
    - » Any changes made are gone when pod is gone (deleting pod deletes the volume)
- Persistent Volume
  - Storage outside the pod – cluster level
    - » Data is preserved if pod goes away
- Persistent Volume Claim
  - Way of saying what is needed for storage
    - » Defines specific amount of storage requested, access mode, etc.
  - Kubernetes looks for a specific matching Persistent Volume and matches it to claim
  - PVC's have different access modes they can specify
    - » ReadWriteOnce – one bound Pod with R/W access
    - » ReadOnlyMany – many bound Pods with R only access
    - » ReadWriteMany – many bound Pods with R/W access

```
1  ---
2  kind: PersistentVolume
3  apiVersion: v1
4  metadata:
5    name: mysql-pv
6    labels:
7      type: local
8  spec:
9    storageClassName: manual
10   capacity:
11     storage: 100M
12   accessModes:
13     - ReadWriteOnce
14   hostPath:
15     path: "/mnt/data"
16 ---
17 apiVersion: v1
18 kind: PersistentVolumeClaim
19 metadata:
20   labels:
21     app: mysql
22   name: mysql-pv-claim
23 spec:
24   storageClassName: manual
25   accessModes:
26   - ReadWriteOnce
27   resources:
28     requests:
29       storage: 100M
```

```
51     spec:
52       containers:
53       - name: mysql
54         image: localhost:5000/roar-db:v1
55         ports:
56         - name: mysql
57           containerPort: 3306
58         env:
59           - name: MYSQL_DATABASE
60             valueFrom:
61               configMapKeyRef:
62                 name: mysql-configmap
63                 key: mysql.database
64           - name: MYSQL_PASSWORD
65             valueFrom:
66               secretKeyRef:
67                 name: mysqlsecret
68                 key: mysqlpassword
69           - name: MYSQL_ROOT_PASSWORD
70             valueFrom:
71               secretKeyRef:
72                 name: mysqlsecret
73                 key: mysqlrootpassword
74           - name: MYSQL_USER
75             valueFrom:
76               configMapKeyRef:
77                 name: mysql-configmap
78                 key: mysql.user
79         volumeMounts:
80         - mountPath: /var/lib/mysql
81           name: mysql-pv-claim
82       volumes:
83       - name: mysql-pv-claim
84         persistentVolumeClaim:
85           claimName: mysql-pv-claim
86 ---
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Kubernetes Storage Objects

- Objects
  - PV's
    - physical volume on the host machine or network that stores the data to be persisted
  - PVC's
    - request for PV's
    - claim check for resource
  - PV's are attached to pods via PVC's
  - Storageclasses define the type of PVCs that a user can request
  - Think of it like:

Pod -> PVC -> PV -> (Storage Class, If applicable) -> Host machine

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Storage Classes and Provisioners

- Storage classes provides way for provider to define classes of storage they offer
- May define QOS levels, backup policies, etc.
- Storage classes have a Provisioner to determine what kind of PV
- Storage classes can dynamically provision a PV if so configured
- Admins can specify StorageClass for PVCs

Volume Plugin	Internal Provisioner	Config Example
AWSElasticBlockStore	✓	AWS EBS
AzureFile	✓	Azure File
AzureDisk	✓	Azure Disk
CephFS	-	-
Cinder	✓	OpenStack Cinder
FC	-	-
Flexvolume	-	-
Flocker	✓	-
GCEPersistentDisk	✓	GCE PD
Glusterfs	✓	Glusterfs
iSCSI	-	-
Quobyte	✓	Quobyte
NFS	-	-
RBD	✓	Ceph RBD
VsphereVolume	✓	vSphere
PortworxVolume	✓	Portworx Volume
ScaleIO	✓	ScaleIO
StorageOS	✓	StorageOS
Local	-	Local

78

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Lab 6: Working with persistent storage – Kubernetes Persistent volumes and Persistent volume claims

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

What is Helm?

- Package Manager and Lifecycle Manager for K8s
  - Like yum, apt but for K8s
  - Bundles related manifests (such as deployment.yaml, sevice.yaml, etc.) into a “chart”
  - When installing chart, Helm creates a “release”
  - Lifecycle management
    - Create, Install, Upgrade, Rollback, Delete, Status, Versioning
  - Benefits
    - Templating, Repeatability, Reliability, Multiple Environment, Ease of collaboration

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

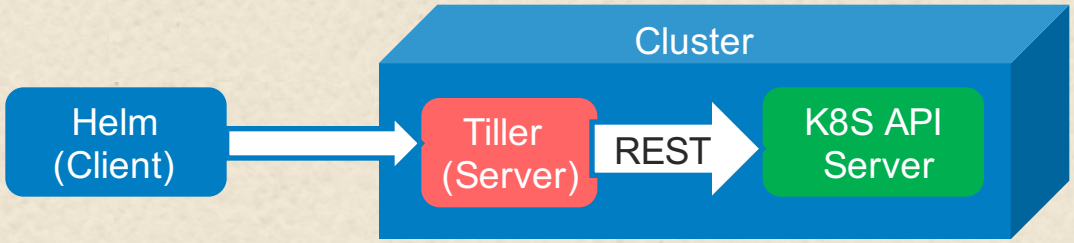
Why do we need something like this?

- Scale and complexity
- Microservice = Pod + Deployment+ ReplicationSet + Ingress + Service times # of microservices
- Duplication of values across objects
- Hard to override values
- Managing lifecycle of all the objects is challenging

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Helm Components

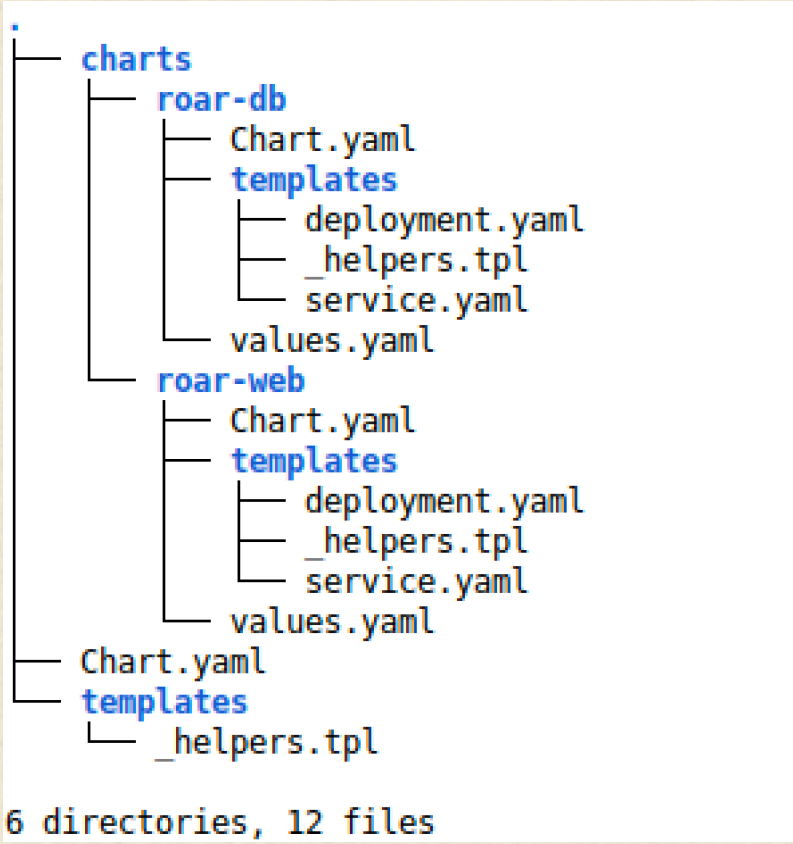
- Helm is the client
  - Command line client
  - Interacts with Tiller server
  - Does local chart development
- Tiller is the server
  - Resides In-cluster
  - Listens to the Helm client
  - Interacts with the Kubernetes API server
  - Manages the lifecycle



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Helm Charts

- Application Deployment Blueprints
  - Collection of k8s resource definition files inside a directory
  - Can deploy simple and complex applications
  - Most files can be “templated”
    - » Go template variations
  - Values to instantiate in templates come from values.yaml or command line overrides
  - _helpers.tpl is a helper template file that can define “functions”
  - Can have directories of chart directories



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Helm Topology

- Chart – a package; a bundle of K8s resources
- Release – a chart instance loaded into K8s
  - » Same chart can be installed several times into the same cluster
  - » Each such chart will have its own release
- Repository – a repository of published charts
- Template - a K8s configuration file mixed with Go/Sprig templates

NAME	REVISION	UPDATED	STATUS	CHART	APP VERSION	NAMESPACE
istio	2	Sat Jul 20 22:09:38 2019	DEPLOYED	istio-1.2.0	1.2.0	istio-system
istio-init	1	Thu Jun 27 13:34:49 2019	DEPLOYED	istio-init-1.2.0	1.2.0	istio-system
istio1	5	Sun Oct 27 17:56:53 2019	DEPLOYED	roar-web-0.1.0		istio1
jenkins-x	1	Thu Jun 6 07:53:23 2019	DEPLOYED	jenkins-x-platform-2.0.330		jx
roar2	2	Sun Oct 27 17:31:35 2019	DEPLOYED	roar-helm-0.1.0		roar2

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Helm Operations

- Helm init – install Tiller to running cluster – will setup any necessary local configuration
- Helm create – create a new chart
- Helm delete – given a release name, delete the release from Kubernetes (use --purge to remove release completely)
- Helm install – install a chart
- Helm lint – check a chart for issues
- Helm list – list releases of charts installed
- Helm rollback – rollback a release to a previous revision
- Helm status – check status of objects deployed with a helm release
- Helm template – Locally render templates
- Helm upgrade – upgrade a release (--recreate-pods option if needed)

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Helm Charts

- Templating Language
  - The Go templating language
  - Template directives look for values and insert them
  - A template directive is enclosed in {{ }}
  - Values passed into a template can be thought of as "namespaced" where a "." separates each element
  - A leading dot means start at topmost level
  - So .Release.Name = "start at top of namespace, look for a "Release" object and then find the "Name" element inside
  - Certain items, such as "Release" are built –in
  - Can do dry-run to see rendered version

Values file

```
favoriteDrink: coffee
```

Template file

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-configmap
data:
  myvalue: "Hello World"
  drink: {{ .Values.favoriteDrink }}
```

Rendered file

```
$ helm install --dry-run --debug ./mychart
SERVER: "localhost:44134"
CHART PATH: /Users/mattbutcher/Code/Go/src/k8s.io/helm/_scratch/mychart
NAME:   geared-marsupi
TARGET NAMESPACE:   default
CHART:  mychart 0.1.0
MANIFEST:
---
# Source: mychart/templates/configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: geared-marsupi-configmap
data:
  myvalue: "Hello World"
  drink: coffee
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Helm Charts

- Templating Language
  - Allow you to override values easily
  - Overrides can come from
    - » Child charts
    - » Additional values files
    - » Command-line values
  - --set has a higher precedence than the values.yaml file
  - Values can also be structured
  - Lower scoping requires more levels (more ".")

https://helm.sh/docs/chart_template_guide/

```
helm install --dry-run --debug --set favoriteDrink=slurm ./mychart
SERVER: "localhost:44134"
CHART PATH: /Users/mattbutcher/Code/Go/src/k8s.io/helm/_scratch/mychart
NAME:   solid-vulture
TARGET NAMESPACE:   default
CHART:  mychart 0.1.0
MANIFEST:
---
# Source: mychart/templates/configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: solid-vulture-configmap
data:
  myvalue: "Hello World"
  drink: slurm
```

Values file

```
favoriteDrink: coffee
```

Templates file

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-configmap
data:
  myvalue: "Hello World"
  drink: {{ .Values.favoriteDrink }}
```

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Helm Charts

- Templating Language
  - Also can use “functions” and “pipelines” in templates
  - Functions
    - » Transform data values in a way to make them more useful
    - » Ex: drink: {{ quote .Values.favorite.drink }}
    - » Calling quote function and passing 1 argument
    - » Helm has over 60 built-in functions from Go template language or Sprig templates
  - Pipelines = “|”
    - » Lets you chain functions together
    - » See example to right

https://helm.sh/docs/chart_template_guide/

Values file

```
favorite:
  drink: coffee
  food: pizza
```

Templates file

```
data:
  myvalue: "Hello World"
  drink: {{ .Values.favorite.drink | repeat 5 | quote }}
  food: {{ .Values.favorite.food | upper | quote }}
```

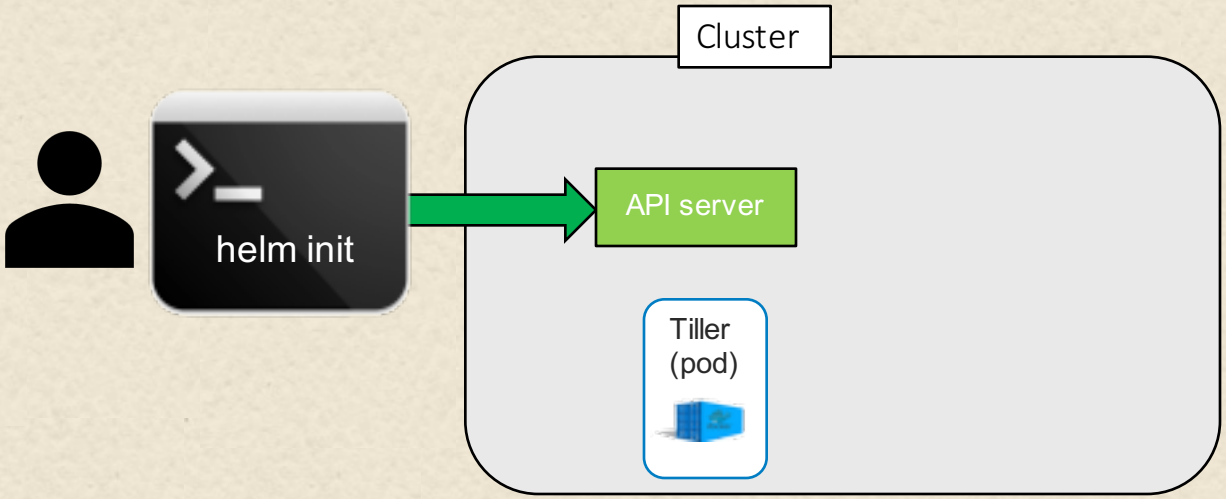
Rendered file

```
data:
  myvalue: "Hello World"
  drink: "coffeecoffeecoffeecoffeecoffee"
  food: "PIZZA"
```

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Helm Initialize Cluster

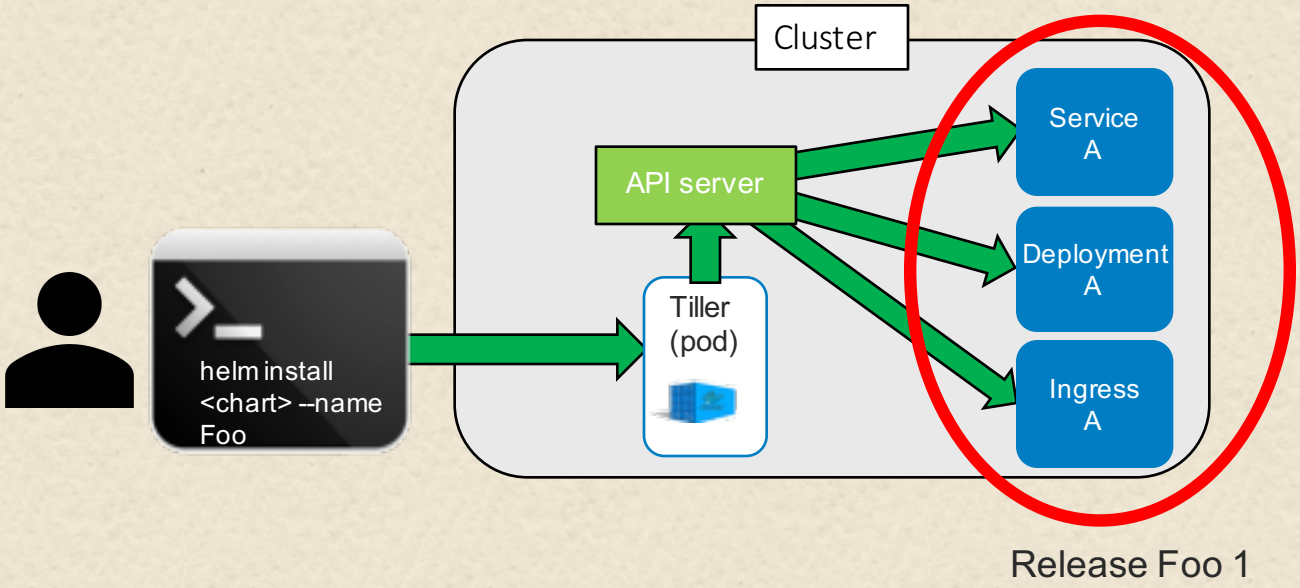
- Installs tiller; setups any necessary configuration



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Helm Install a Chart

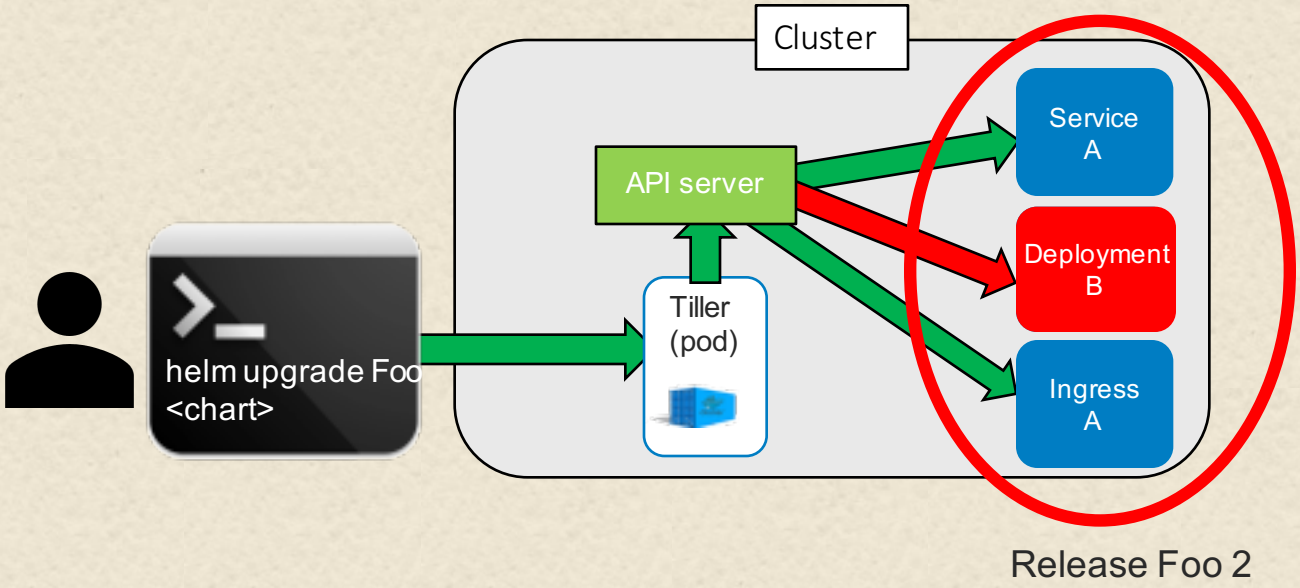
- helm install <chart>



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Helm Upgrading a Chart

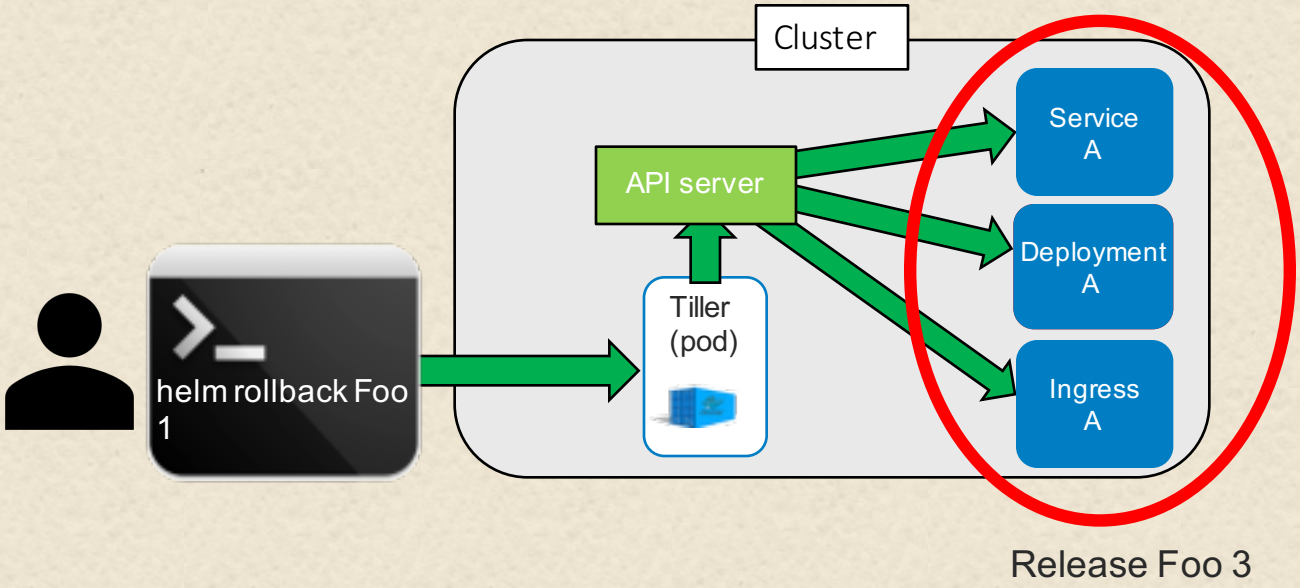
- helm upgrade <release> <chart>



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Helm Rollback

- helm history <release> (to see revision numbers)
- helm rollback <release> <revision>



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Lab 7: Using Helm

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Init Containers

- Besides ”regular” containers that are intended to keep running, pods can have 0 to many “init containers”
- Init containers
  - Run at startup (and before other containers)
  - Useful for initialization, validation
  - Typically have command/arguments to run
  - Run serially
  - Must all exit cleanly in order for the regular containers to begin running
  - Listed under “initContainers” section in spec

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    readiness.status.sidecar.istio.io/
[...]
spec:
  containers:
  - image: localhost:5000/roar-web:v2
    imagePullPolicy: Always
    name: roar-web
    ports:
    - containerPort: 8080
      name: roar-web
      protocol: TCP
    resources: {}
    [...]
  initContainers:
  - args:
    - -p
    - "15001"
    - -u
    - "1337"
    - -m
    - REDIRECT
    - -i
    - '*'
    [...]
    image: docker.io/istio/proxy_init:1.2.0
    imagePullPolicy: IfNotPresent
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Istio is a Service Mesh. What is a service mesh?

- Open Platform to connect, monitor, and secure microservices
  - Lets you connect, secure, and control services .
  - Example of a service mesh : network of microservices and interactions between them.
  - Typical service mesh functions: discovery, load balancing, failure recovery, metrics, monitoring, A/B testing, canary rollouts, rate-limiting, access control, end-to-end authentication.
  - Requires few or no changes to service code.
  - Add istio support as a sidecard proxy – additional container in your pod.
  - Intercepts all network traffic between microservices.Containers within a pod can find each other via localhost.

https://istio.io/docs/concepts/what-is-istio/

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

How does Istio work?

- Istio deploys a sidecar container (proxy) in the pods “in front of” each microservice
- All traffic meant for the microservice in the pod is instead directed to the proxy
- The istio proxy uses policies to decide if/when/how traffic it intercepts should be passed on to the microservice
- This arrangement allows for advanced implementations such as canary deployments, fault injections, and circuit breakers
- Istio as an application runs in a Linux container in dedicated pods for Istio in a Kubernetes namespace.
- Istio can be configured to automatically inject an Istio sidecar container.

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Istio Architecture

- The Data Plane and the Control Plane
  - The **data plane** is composed of a set of intelligent proxies (Envoy) deployed as sidecars. These proxies mediate and control all network communication between microservices along with Mixer, a general-purpose policy and telemetry hub.
  - The **control plane** manages and configures the proxies to route traffic. Additionally, the control plane configures Mixers to enforce policies and collect telemetry.
  - https://istio.io/docs/concepts/what-is-istio/

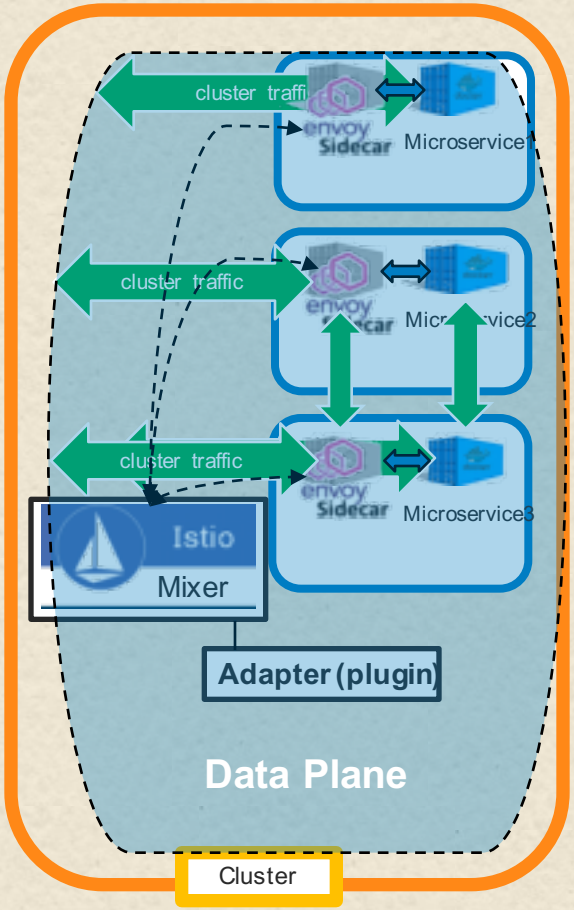
(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Istio – Service Mesh and Data Plane

- How is the service mesh implemented?
  - Microservices exist in pods in a cluster
  - Traffic is ultimately routed to them
  - A network proxy container is injected as a "sidecar" into the pod
  - All network traffic (HTTP, REST, etc.) goes through the sidecar proxy first
  - Microservice instance is only aware of its local proxy and not of the larger network.
  - Distributed network abstracted away from microservice programmer.

- What is Envoy?
  - High-performance proxy developed to mange all inbound and outbound traffic for all services in the service mesh.Microservices exist in pods in a cluster
  - Most basic Istio functionality comes from Envoy (load balancing, fault injection, etc.)

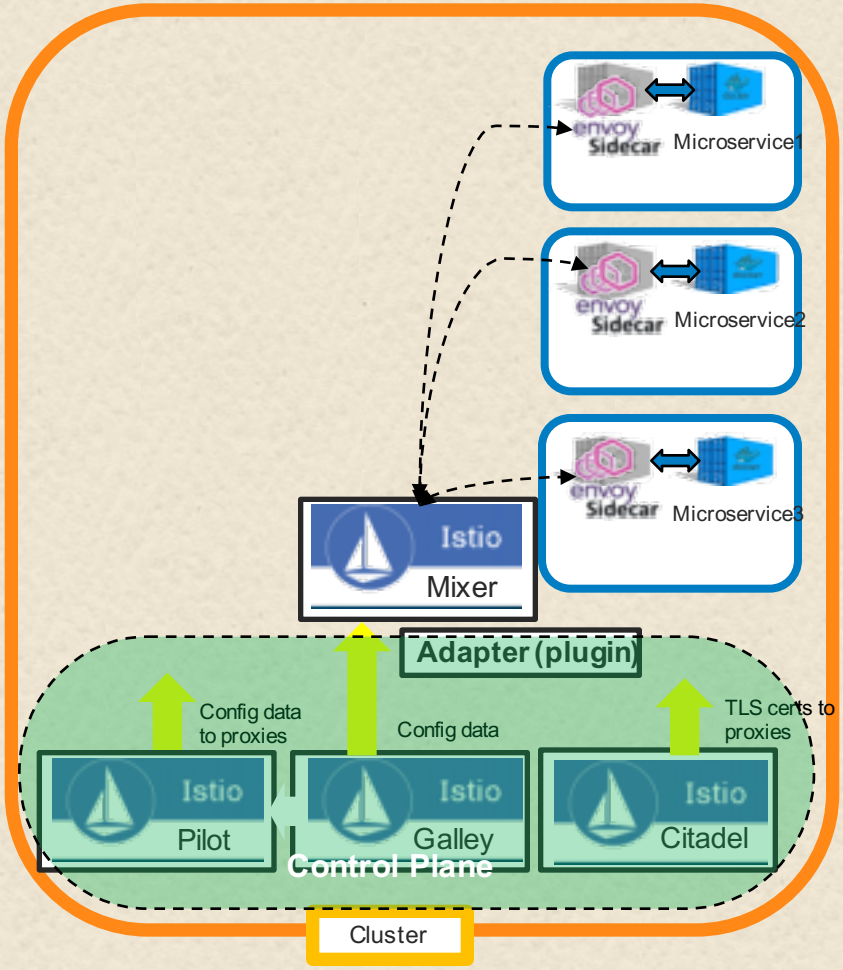
- What is a data plane?
  - In effect, sidecar proxies make up the data plane (with Mixer)
  - Mixer - Istio component responsible for policy enforcement and telemetry collection (plugins for interfacing with other hosts and backends
  - Data plane provides:
    - **Service Discovery** – what backend/upstream service instances are available? Health checking – are upstream instances healthy and ready to accept traffic?
    - **Authorization/authentication** – for incoming requests, can called be validated and allowed to do what they are asking?
    - **Routing** – which upstream service group should a REST request be sent to?
    - **Load Balancing** –which specific service instance in a group should a request be sent to and what retry values, timeout, etc?
    - **Observability** – for a request produce detailed logging, tracing, and statistics info.
  - Overall data plane looks at, translates, and forwards each network packet that goes to or comes from its corresponding microservice instance.



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Istio – Control Plane

- What is a control plane?
- Takes the group of individual, stateless sidecare proxies and creates a distributed system from them.
- Features
  - Automatic load-balancing
  - Fine-grained traffic control
  - Pluggable policy layer and configuration API
  - Automatic metrics, logs, and traces
  - Secure service-to-service communication
- Components
  - Pilot
    - » Configures all Envoy proxy instances in mesh
    - » Provides service discovery for sidecars
    - » Core component for traffic mgmt.
    - » Uses configuration from Galley
  - Galley
    - » Distribution and centralized config mgmt.
    - » Insulates rest of Istio components from K8S
    - » Validates config from users
  - Citadel
    - » Provides strong end-user authentication and authentication between services
    - » Uses mutual TLS
    - » Built-in credential and identity mgmt

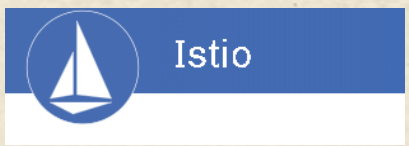


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Istio K8s Objects

- Gateway
  - Describes a load balancer operating at edge of mesh receiving incoming or outgoing HTTP/TCP connections.
  - Spec describes
    - Set of ports that should be exposed
    - Type of protocol to use
    - Configuration for load balancer, etc.
  - Example: sets up a proxy to act as a load balancer exposing port 80 (http), 443 (https), etc.
  - Applied to proxy running on a pod with label app: my-gateway-controller
  - https://istio.io/docs/reference/config/networking/v1alpha3/virtual-service/

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: my-gateway
  namespace: some-config-namespace
spec:
  selector:
    app: my-gateway-controller
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - uk.bookinfo.com
    - eu.bookinfo.com
    tls:
      httpsRedirect: true # sends 301 redirect for http requests
  - port:
      number: 443
      name: https-443
      protocol: HTTPS
    hosts:
    - uk.bookinfo.com
    - eu.bookinfo.com
```



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Istio K8s Objects

- VirtualService
  - Defines a set of traffic routing rules when a host is addressed
  - Each routing rule defines matching criteria for traffic of a particular protocol
  - If traffic is matched, sent to a named destination (k8s) service (or subset of it)
  - Traffic source can also be matched in routing rule
    - Allows routing to be customized for client contexts
  - Example: All http traffic by default goes to pods of the "reviews" service with label "version:v1"
  - Http requests with path starting with /wpcatalog or /consumercatalog get rewritten to /newcatalog and sent to pods with label "version: v2". https://istio.io/docs/reference/config/networking/v1alpha3/virtual-service/

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews-route
spec:
  hosts:
  - reviews.prod.svc.cluster.local
  http:
  - match:
    - uri:
        prefix: "/wpcatalog"
    - uri:
        prefix: "/consumercatalog"
    rewrite:
      uri: "/newcatalog"
    route:
    - destination:
        host: reviews.prod.svc.cluster.local
        subset: v2
  - route:
    - destination:
        host: reviews.prod.svc.cluster.local
        subset: v1
```

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Istio K8s Objects

- DestinationRule
  - Works with VirtualService
  - Declares subset/version of a route destination
  - Declares named service subsets
  - https://istio.io/docs/reference/config/networking/v1alpha3/virtual-service/

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: reviews-destination
spec:
  host: reviews.prod.svc.cluster.local
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
```

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

VirtualService Variation

- HTTPFaultInjection.Abort

HTTPFaultInjection.Abort

Abort specification is used to prematurely abort a request with a pre-specified error code. The following example will return an HTTP 400 error code for 1 out of every 1000 requests to the “ratings” service “v1”.

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: ratings-route
spec:
  hosts:
  - ratings.prod.svc.cluster.local
  http:
  - route:
    - destination:
        host: ratings.prod.svc.cluster.local
        subset: v1
    fault:
      abort:
        percentage:
          value: 0.1
        httpStatus: 400
```

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

VirtualService Variation

- HTTPFaultInjection.Delay

HTTPFaultInjection.Delay

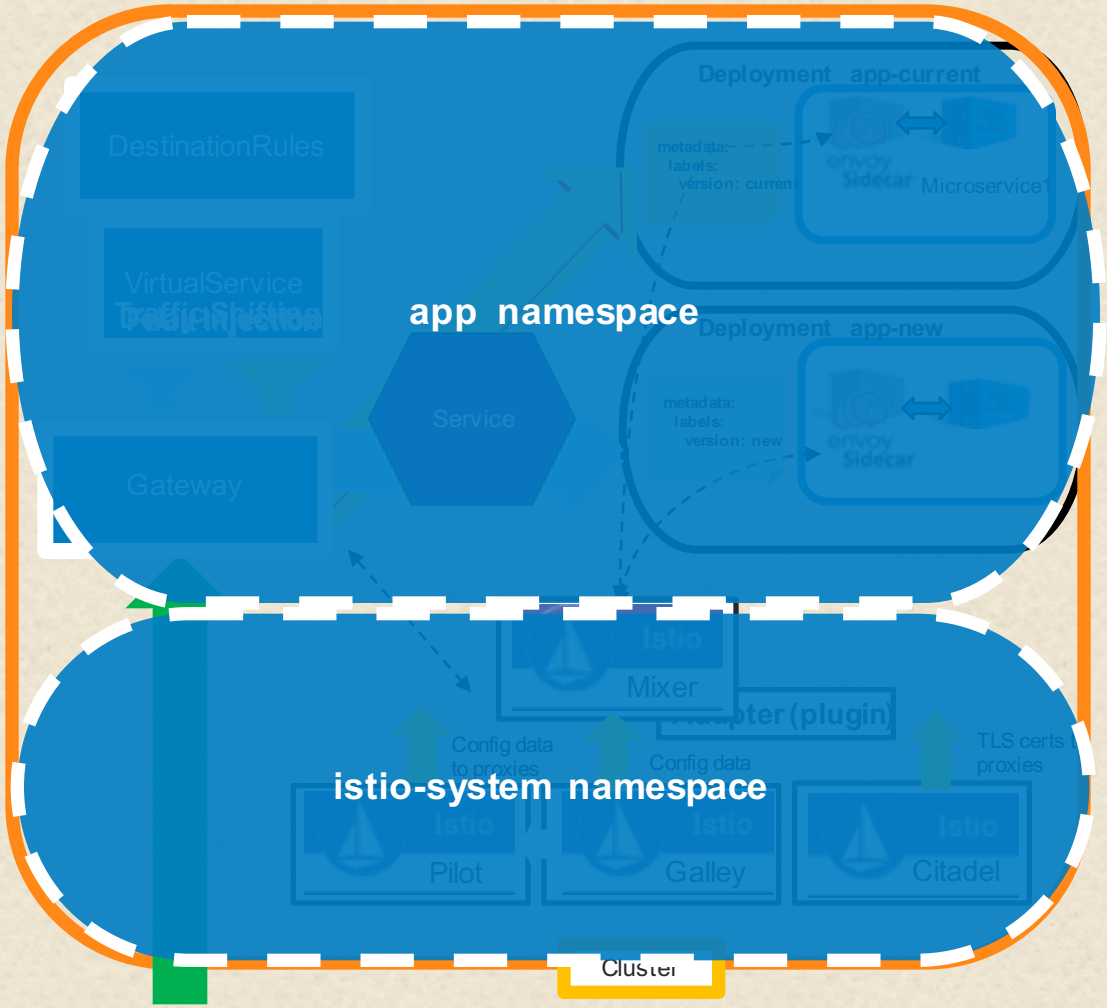
Delay specification is used to inject latency into the request forwarding path. The following example will introduce a 5 second delay in 1 out of every 1000 requests to the “v1” version of the “reviews” service from all pods with label env: prod

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews-route
spec:
  hosts:
  - reviews.prod.svc.cluster.local
  http:
  - match:
    - sourceLabels:
        env: prod
    route:
    - destination:
        host: reviews.prod.svc.cluster.local
        subset: v1
    fault:
      delay:
        percentage:
          value: 0.1
        fixedDelay: 5s
```

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Istio – The Big Picture

- Istio extends the functionality of Kubernetes
- Still requires deployment & service
- In addition to the control plane and data planes, uses additional pieces for routing functionality
  - Gateway
    - » load balancer at edge of mesh
    - » Configures how proxies load balance HTTP, TPC, or gRPC traffic
    - » No traffic routing
  - Virtual Services
    - » configures ordered list of routing rules
    - » Controls how proxies route requests for a service within mesh
  - Destination Rules
    - » configure policies you want Istio to apply to a request after enforcing routing rules from virtual service
  - Gate way is "bound" to virtual service to specify routing
- Istio components run in namespace "istio-system"
- Routing examples
  - Traffic shifting
  - Fault injection
  - Delay injection



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Lab 8: Working with Istio

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Scaling

- kubectl scale
  - Allows you to scale resources up/down via command line
  - What happens if you scale a resource to 0?

```
kubectl scale --replicas=3 rs/foo                                 # Scale a replicaset named 'foo' to 3
kubectl scale --replicas=3 -f foo.yaml                            # Scale a resource specified in "foo.yaml" to 3
kubectl scale --current-replicas=2 --replicas=3 deployment/mysql  # If the deployment named mysql's current size is 2, scale mysql to 3
kubectl scale --replicas=5 rc/foo rc/bar rc/baz                   # Scale multiple replication controllers
```

- kubectl autoscale and horizontal pod autoscaler (hpa)
  - Kubernetes scales a number of replicas within a given range to maintain a preset level of resource (cpu) usage

```
$ kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
```

  - Above defines an HPA that maintains between 1 and 10 replicas of pods controlled by deployment php-apache
  - HPA will autoscale number of replicas to maintain average of 50% cpu usage

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

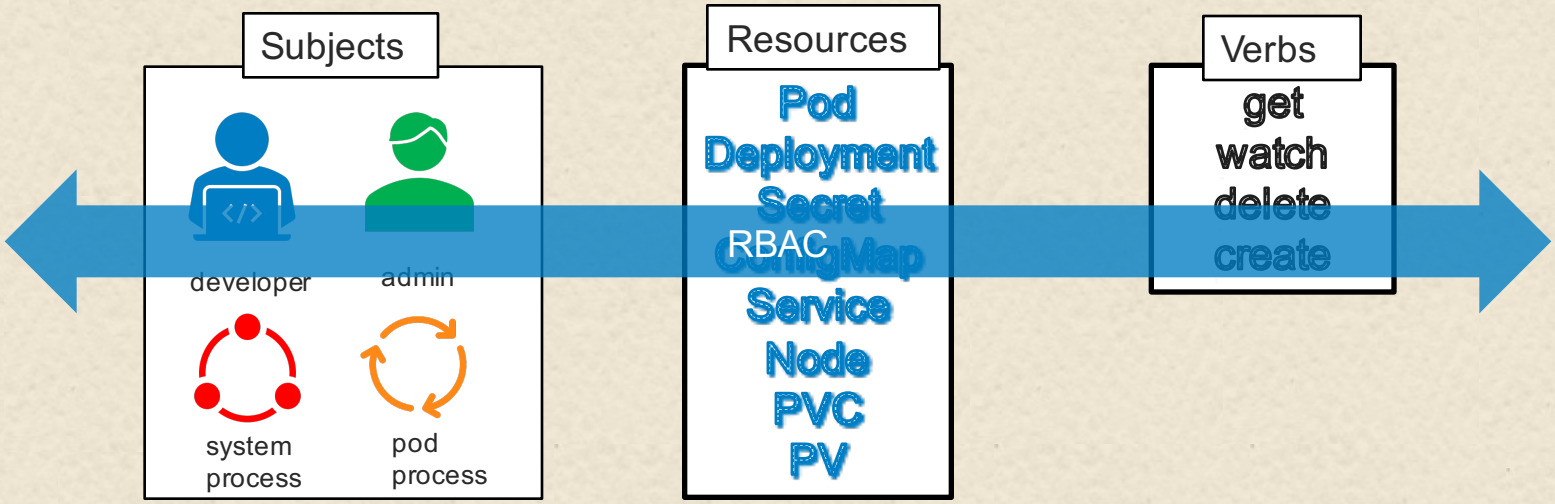
Kubernetes RBAC

- Role-based Access Control
  - Why do we need RBAC ? Because in a real-world k8s environment, we may need to:
    - Have multiple users with different properties, establishing a proper authentication mechanism.
    - Have full control over which operations each user or group of users can execute.
    - Have full control over which operations each process inside a pod can execute.
    - Limit the visibility of certain resources of namespaces.

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Kubernetes RBAC

- The pieces
  - 3 types of elements involved
    - Subjects: users or processes that need to access the API
    - Resources: Kubernetes API Objects available in the cluster. (Examples include Pods, Deployments, Services, Nodes, and Persistent Volumes, etc.)
    - Verbs: The Kubernetes operations that can be executed on the resources above. Multiple kinds of verbs available – all are CRUD operations (Examples include get, watch, create, delete, etc.)

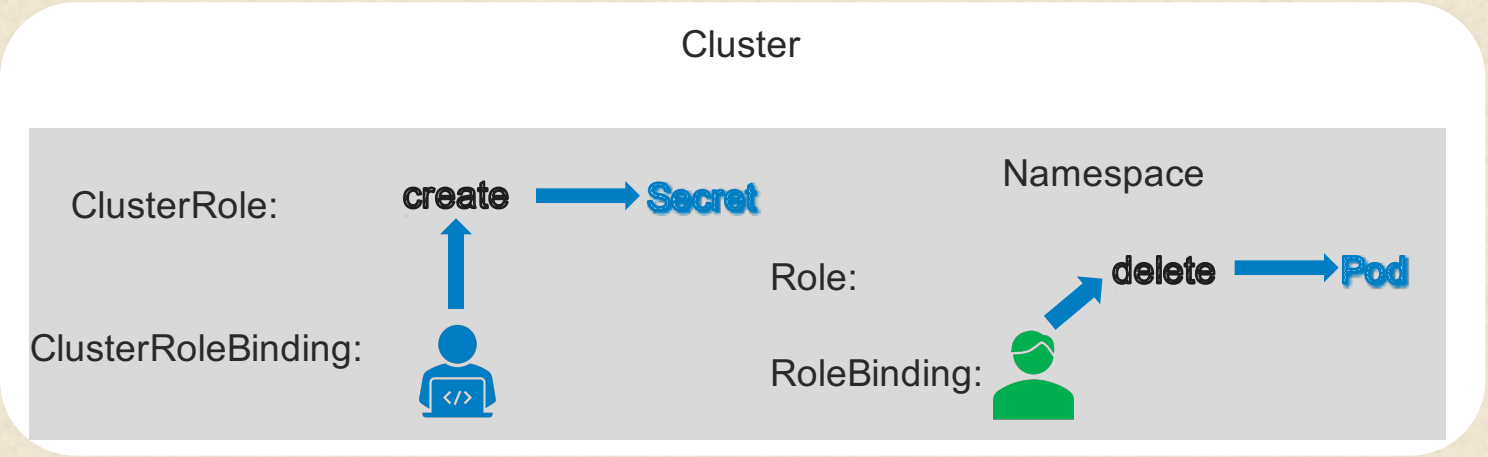


Goal: Connect subjects, API resources, and verbs. Specify, given a user, which operations can be executed over which resources.

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Kubernetes RBAC

- The mechanisms
  - Object types
    - k8s Roles: Connect API Resources and Verbs; can be reused for different Subjects; bound to a namespace
    - k8s RoleBinding: Connects Subjects to Roles; Given a role that combines API objects and verbs, defines which subjects can use it.
    - k8s ClusterRole: a role to be applied across a cluster
    - k8s ClusterRoleBinding: Connects Subjects to ClusterRole.



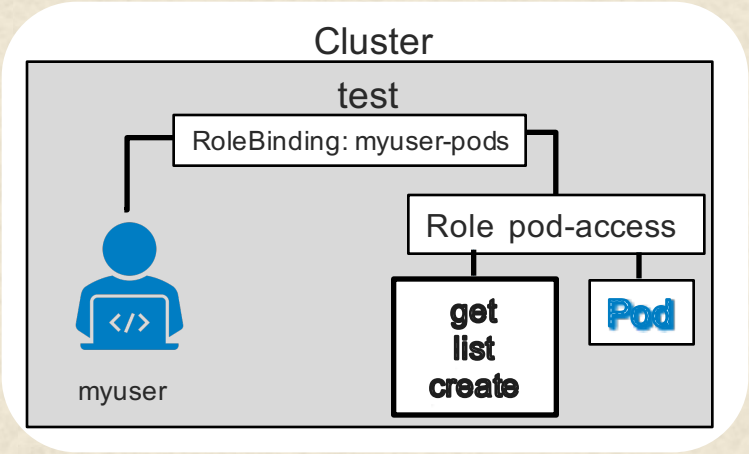
(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Kubernetes RBAC

- Example: give user limited access to work with pods in test ns
  - Specs define role and rolebindings

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: pod-access
  namespace: test
rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - get
  - list
  - create
```

```yaml
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: myuser-pods
  namespace: test
subjects:
- kind: User
  name: myuser
roleRef:
  kind: Role
  name: pod-access
  apiGroup: rbac.authorization.k8s.io
```



  - User can do these operations:
    - kubectl get pods –n test
    - kubectl describe pod –n test <name of pod>
    - kubectl create –f <pod-spec.yaml> -n test
  - But not these
    - kubectl get pods -n default
    - kubectl get pods -n test

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Kubernetes RBAC - Service Accounts

- Users: These are global, and meant for humans or processes living outside the cluster.
- ServiceAccounts: These are namespaced and meant for intra-cluster processes running inside pods.
- Both authenticate against the API to get access to resources
- But there is no kubernetes "User" object
- So you can do "kubectl create serviceaccount <name>" but not "kubectl create user <name>"
- Bottom line: cluster does not store any information about users, so must be managed outside of cluster with certificates, tokens, etc.

Create ServiceAccount

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: myapp
  labels:
    app: myapp
```

Create Role

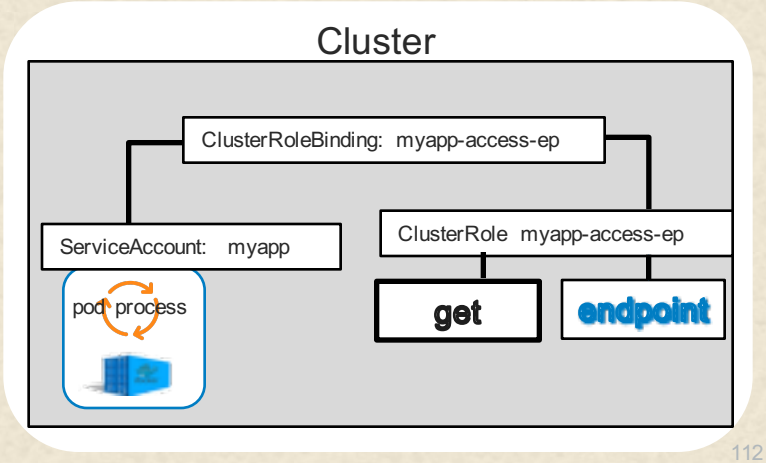
```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: myapp-access-ep
  labels:
    app: myapp
rules:
- apiGroups: [""]
  resources: ["endpoints"]
  verbs: ["get"]
```

Add ServiceAccount to Pod

```yaml
[...]
spec:
  template:
      metadata:
        name: myapp
        labels:
          app: myapp
      spec:
        serviceAccountName: myapp
        containers:
[...]
```

Tie Role to ServiceAccount

```yaml
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: myapp-access-ep
  labels:
    app: myapp
subjects:
- kind: ServiceAccount
  name: myapp
roleRef:
  kind: ClusterRole
  name: myapp-access-ep
  apiGroup: rbac.authorization.k8s.io
```



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Kubernetes Operators

- What is a k8s operator?
  - An Operator is a method of packaging, deploying and managing a Kubernetes application.
  - A Kubernetes application is an application that is both deployed on Kubernetes and managed using the Kubernetes APIs and kubectl tooling.
- How are operators different from other tools?
  - Purpose-built to run a Kubernetes application, with operational knowledge baked in
  - "Smarter" and more tailored than generic tools
  - Cloud-like capabilities encoded into Operator code can provide an advanced user experience
  - Can automate features like updates, backups and scaling
  - All of this is accomplished using standard Kubernetes tools, CLI and API.

https://coreos.com/operators/

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Operator Framework

- Operator Framework
  - Designed to help make it easy to create and manage operators
  - Created by CoreOS – now part of Red Hat
  - Main piece is Operator SDK
    - Makes it easy to code operators w/o having to know k8s api details
- Operators can be coded in
  - Go
  - Helm
  - Ansible

https://coreos.com/operators/

The Operator Framework is an open source project that provides developer and runtime Kubernetes tools, enabling you to accelerate the development of an Operator. The Operator Framework includes:

Enables developers to build Operators based on their expertise without requiring knowledge of Kubernetes API complexities.

Oversees installation, updates, and management of the lifecycle of all of the Operators (and their associated services) running across a Kubernetes cluster.

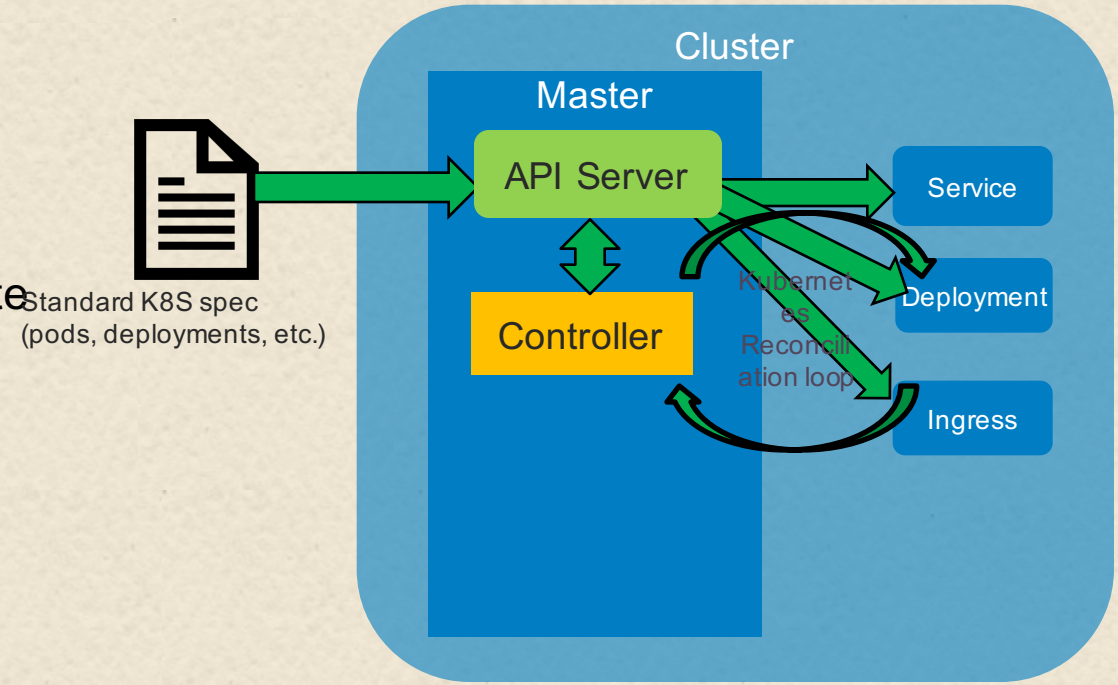
Operator Metering (joining in the coming months): Enables usage reporting for Operators that provide specialized services.

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

How Do Operators Work?

- Part 1

- Kubernetes is essentially a "desired state manager"
  - We provide desired state (via spec) and it makes it happen and tries to keep it that way even if something goes wrong
  - Leverages "control plane" on master
  - Control plane includes "controllers"
  - Controllers try to reconcile state by
    - Monitoring existing objects to determine their state
    - Compare object's state to k8s yaml spec
    - If difference, try to correct it



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Custom Resource Definitions

- Custom ResourceDefinition (CRD)
  - K8S api server creates a new RESTful resource path for each version you supply
  - CRD can be either scoped for a namespace or cluster
    - » Per “scope” field
  - Manipulate as other objects
  - Create by defining spec and applying it
  - Creates new RESTful API endpoint
    - » Example: /apis/stable.example.com/v1/namespaces/*/crontabs/…
  - Endpoint URL can then be used to create and manage custom objects
  - Kind of these objects will be CronTab
  - Can then make instances o f these custom objects
  - And use them with operations like kubectl get

- https://kubernetes.io/docs/tasks/access-kubernetes-api/custom-resources/custom-resource-definitions/

```yaml
apiVersion: "stable.example.com/v1"
kind: CronTab
metadata:
  name: my-new-cron-object
spec:
  cronSpec: "* * * * */5"
  image: my-awesome-cron-image
```

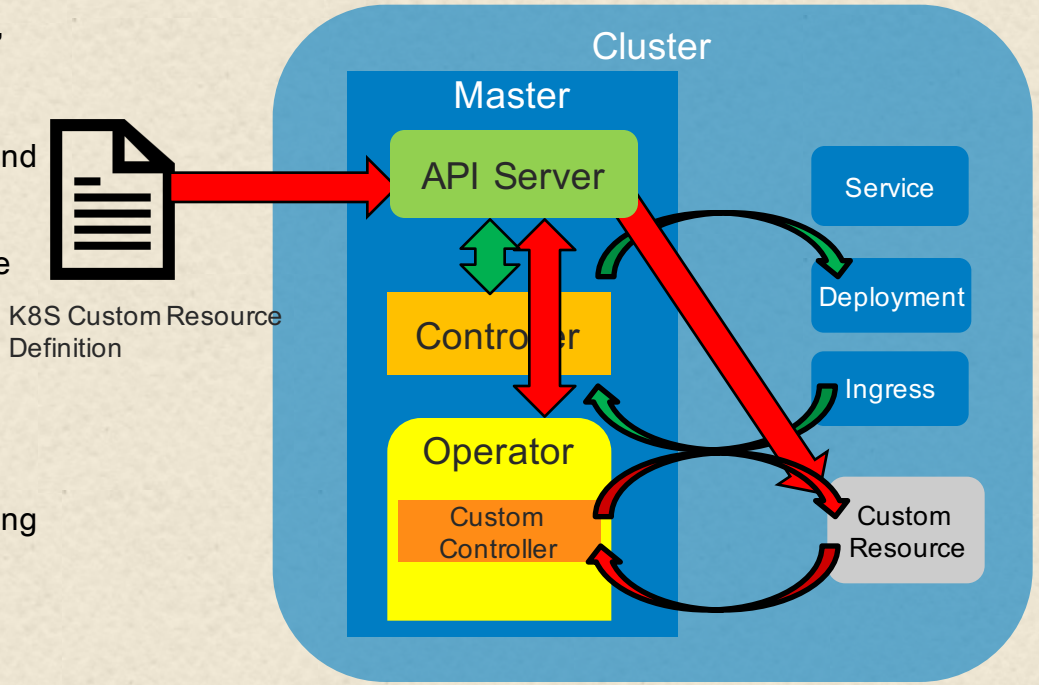
```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  # name must match the spec fields below, and be in the form: <plural>.<group>
  name: crontabs.stable.example.com
spec:
  # group name to use for REST API: /apis/<group>/<version>
  group: stable.example.com
  # list of versions supported by this CustomResourceDefinition
  versions:
    - name: v1
      # Each version can be enabled/disabled by Served flag.
      served: true
      # One and only one version must be marked as the storage version.
      storage: true
  # either Namespaced or Cluster
  scope: Namespaced
  names:
    # plural name to be used in the URL: /apis/<group>/<version>/<plural>
    plural: crontabs
    # singular name to be used as an alias on the CLI and for display
    singular: crontab
    # kind is normally the CamelCased singular type. Your resource manifests use this.
    kind: CronTab
    # shortNames allow shorter string to match your resource on the CLI
    shortNames:
    - ct
  preserveUnknownFields: false
  validation:
    openAPIV3Schema:
      type: object
      properties:
        spec:
          type: object
          properties:
            cronSpec:
              type: string
            image:
              type: string
```


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

How Do Operators Work?

Part 2

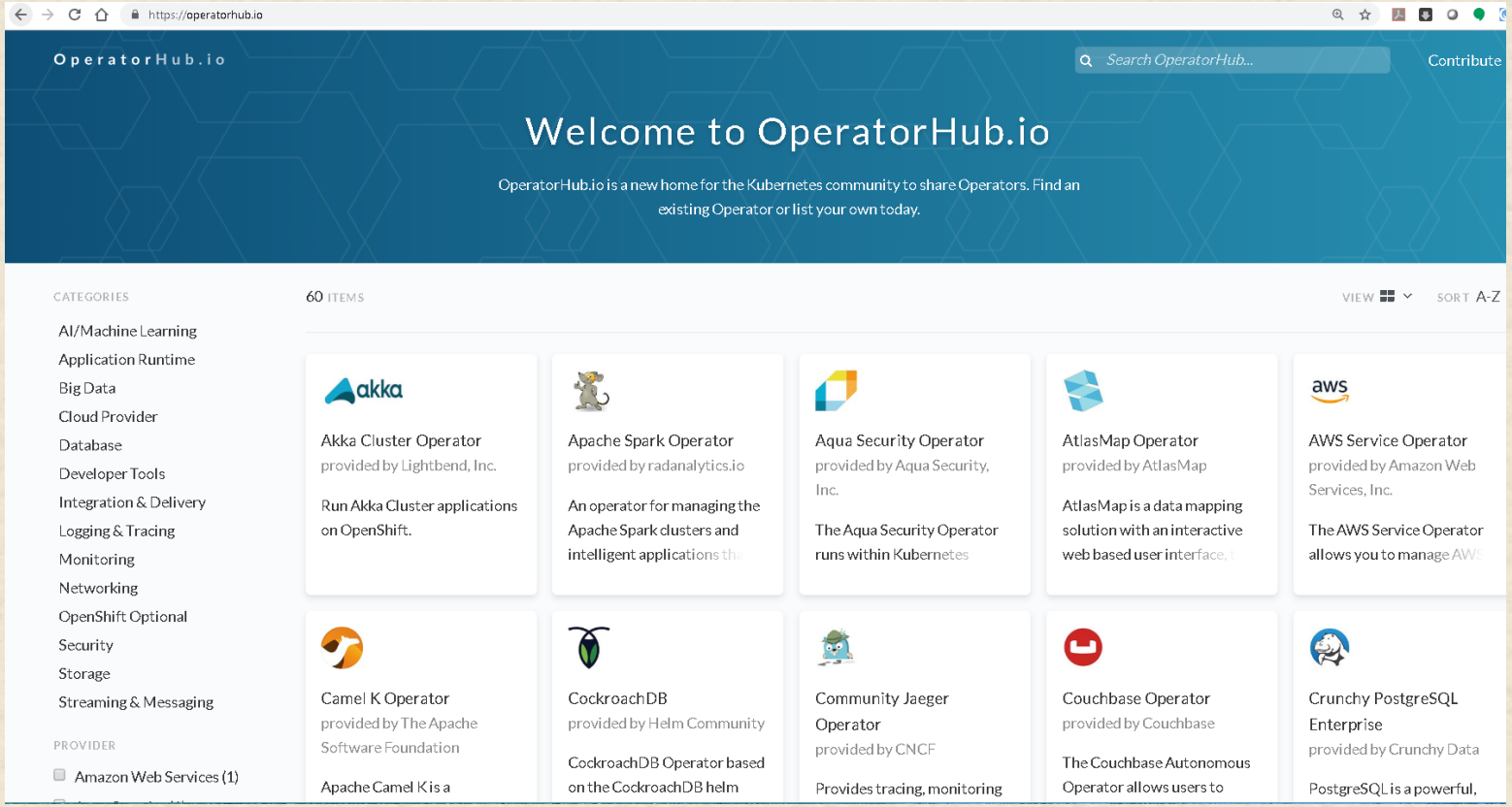
- Native k8s objects (deployments, services, etc.) are designed to work well with the standard k8s controller model
- Considered "stateless" – don't need to persist/rely on state
- That idea has issues for more complex, stateful applications
  - Example: database running on multiple nodes
  - If nodes go down, need to reload data to get back to previous good state
- Usual cloud friendly operations like scaling, upgrading, disaster recovery for stateful app don't fit
- To deal with this, operators extend k8s
  - Allow defining custom controller to watch your app and do custom processing based on its state
  - Application that is to be watched by operator is defined as new object type in k8s - custom resource (CR)
    - Has its own yaml spec and object type CustomResource
    - Format understood by the API server
  - Operators run and oversee a custom controller loop
  - Overall behavior of an operator's controller reconciling state for CR is similar to way native k8s handles reconciling native objects
  - Operators give you the chance to do custom processing when reconciling



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

OperatorHub.io

- Operator Registry for Kubernetes Operators



(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Lab 9: Kubernetes Operators

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

What is GitOps?

- An approach to Continuous Delivery for infrastructure and workloads
- Works by using Git as source of truth
- Per k8s, this means using "git push" instead of "kubectl create/apply" or "helm install/upgrade" when ready to add/change spec
- In traditional CI/CD pipeline, CD (per CI process) creates build artifacts and promotes them to production
- In GitOps pipeline model, ANY change to production must be committed in source control prior to being applied to cluster
  - Should ideally be done through pull request/merge request
- Advantages
  - Code review for changes
  - Tracking of who did what
  - Rollback is via Git
  - Whole infrastructure can be recreated from source contrl

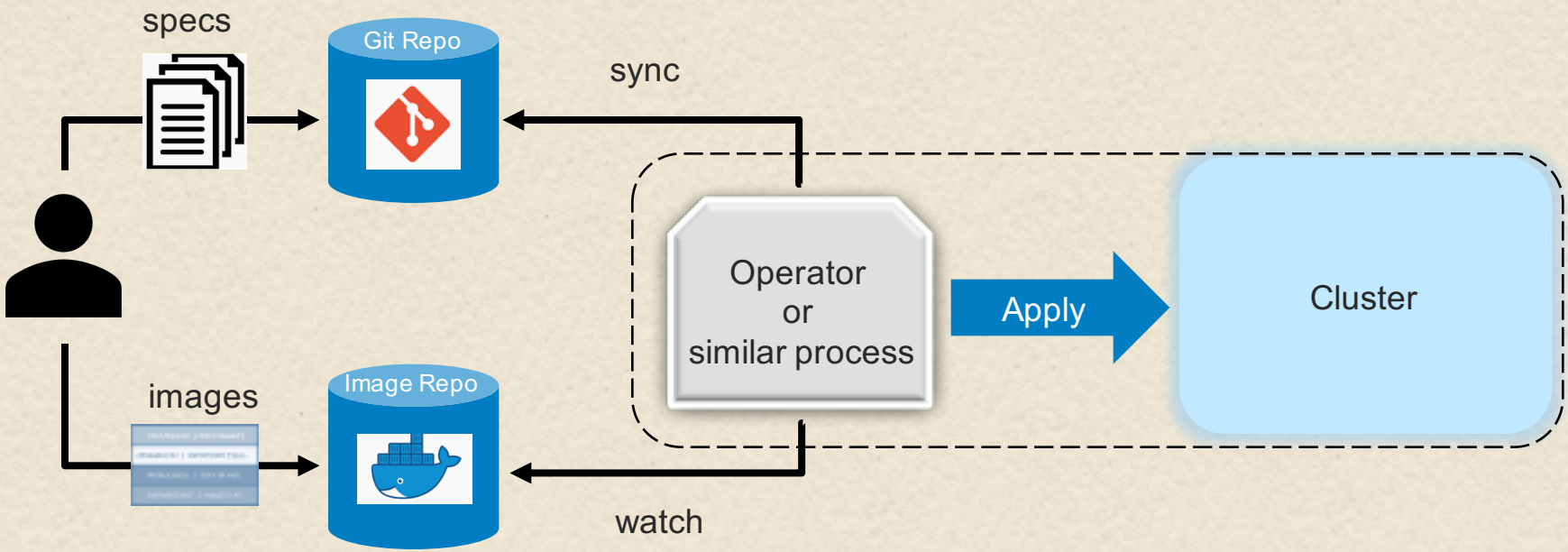
(c) 2021 Tech Skills Transformations, LLC and Brent Laster

GitOps with Kubernetes – what do you need?

- a Git repository with your workloads definitions in YAML format, Helm charts and any other Kubernetes custom resource that defines your cluster desired state (*config* repository)
- a container registry where your CI system pushes immutable images (no *latest* tags, use *semantic versioning* or git *commit sha*)
- an operator or process that runs in your cluster and does a two-way synchronization:
- watches the registry for new image releases
  - based on deployment policies
    - » updates the workload definitions with the new image tag
    - » commits the changes to the config repository
- watches for changes in the config repository and applies them to your cluster

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

GitOps Basic Model

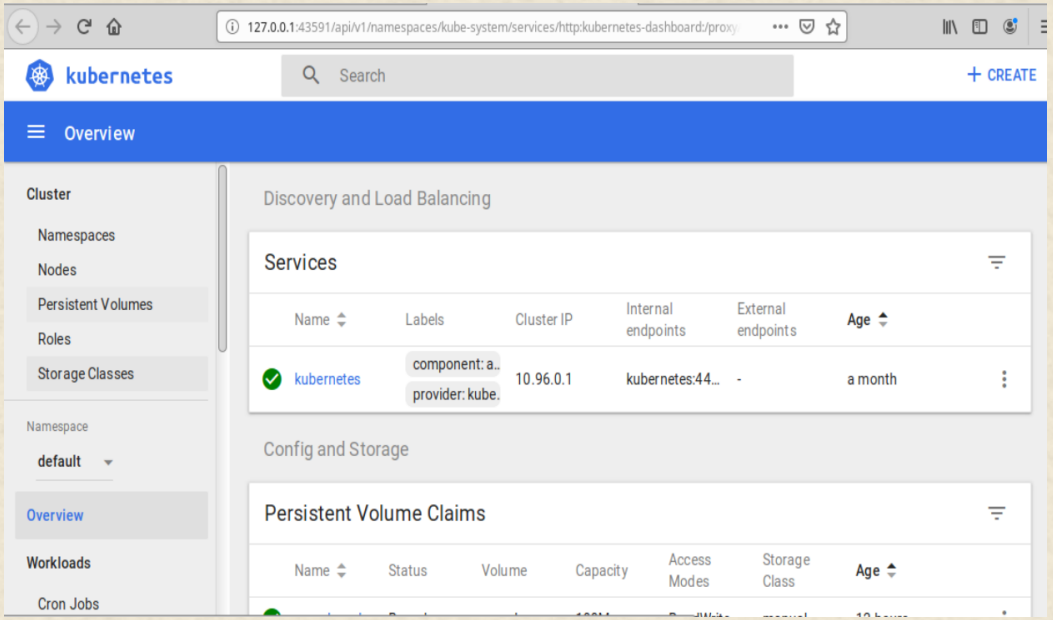


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Kubernetes Dashboard

- Visual interface to the cluster

- General-purpose, web-based UI for clusters
- Can get overall status about objects in the cluster
- Invoke via “minikube dashboard” or “kubectl apply” from location and then “kubectl proxy”
- Can also create or modify objects via “+ CREATE” link
  - Can type in json or yaml
  - Can upload a spec file



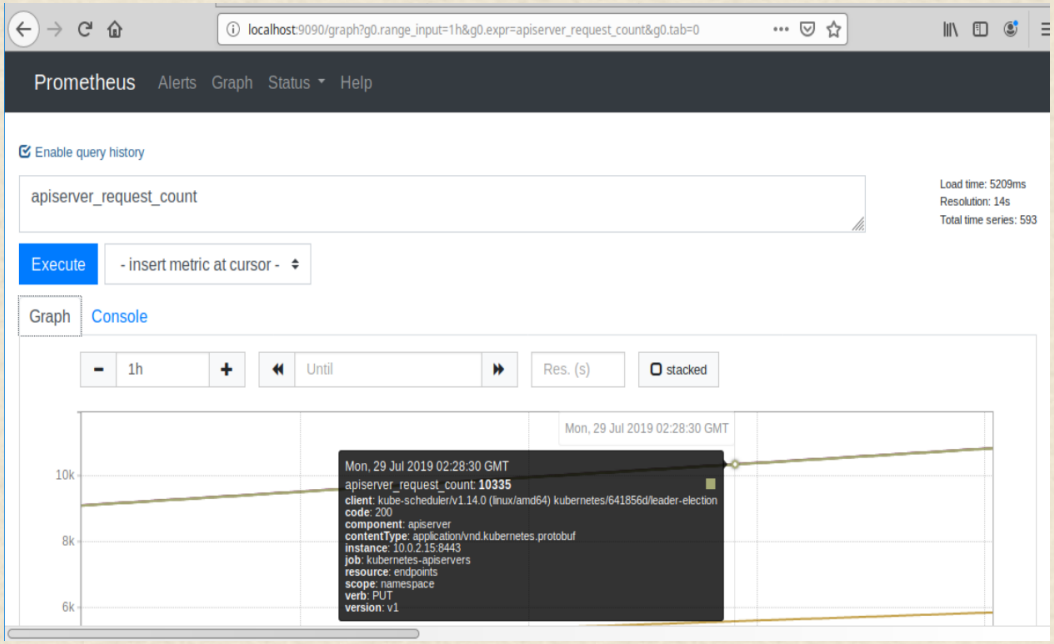
(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Prometheus

- Event Monitoring and Alerting

- Open-source application for event monitoring and alerting
- Records realtime metrics in a time-series database
- Powerful query language - PromQL
- Prometheus “monitoring platform” usually has
  - Multiple exporters that typically run on the monitored host to export metrics
  - Prometheus to centralize and store the metrics
  - Alertmanager to trigger alerts based on the metrics
  - Grafana to produce dashboards

https://en.wikipedia.org/wiki/Prometheus_(software)

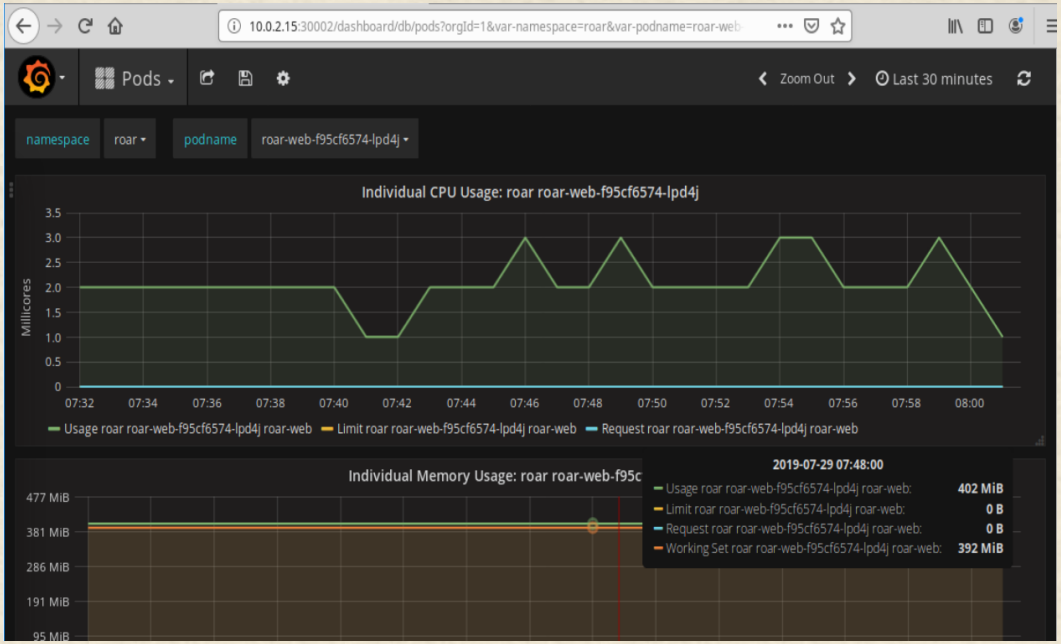


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Grafana

- Metric analytics and visualization suite

- Way to visualize time series for infrastructure and application analytics
- Allows users to build dashboards with graphs, panels, etc.
- Accepts data from many data sources including:
  - Graphite
  - Elasticsearch
  - InfluxDB
  - Prometheus
  - MySQL
  - Postgres
  - Cloudwatch

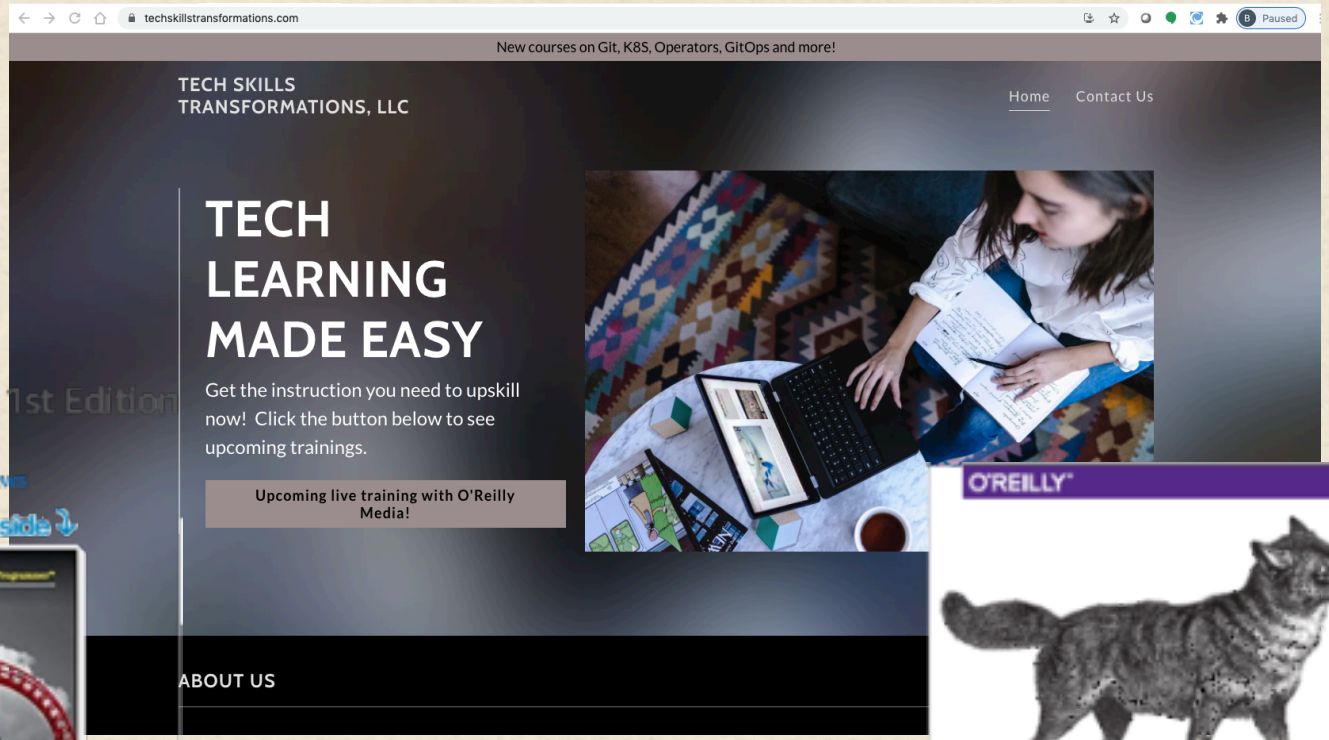


(c) 2021 Tech Skills Transformations, LLC and Brent Laster

Bonus Lab: Monitoring

(c) 2021 Tech Skills Transformations, LLC and Brent Laster

That's all - thanks!



techskillstransformations.com
getskillsnow.com

(c) 2021 Tech Skills Transformations, LLC and Brent Laster