
ORACLE®

Using MySQL Containers

Matt Lord
Senior MySQL Product Manager
@mattalord

ORACLE

MySQL

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Program Agenda

1. What are Containers?
2. What is Docker?
3. Why Should I Care?
4. Official MySQL Containers
5. Where Do I Go From Here?

Program Agenda

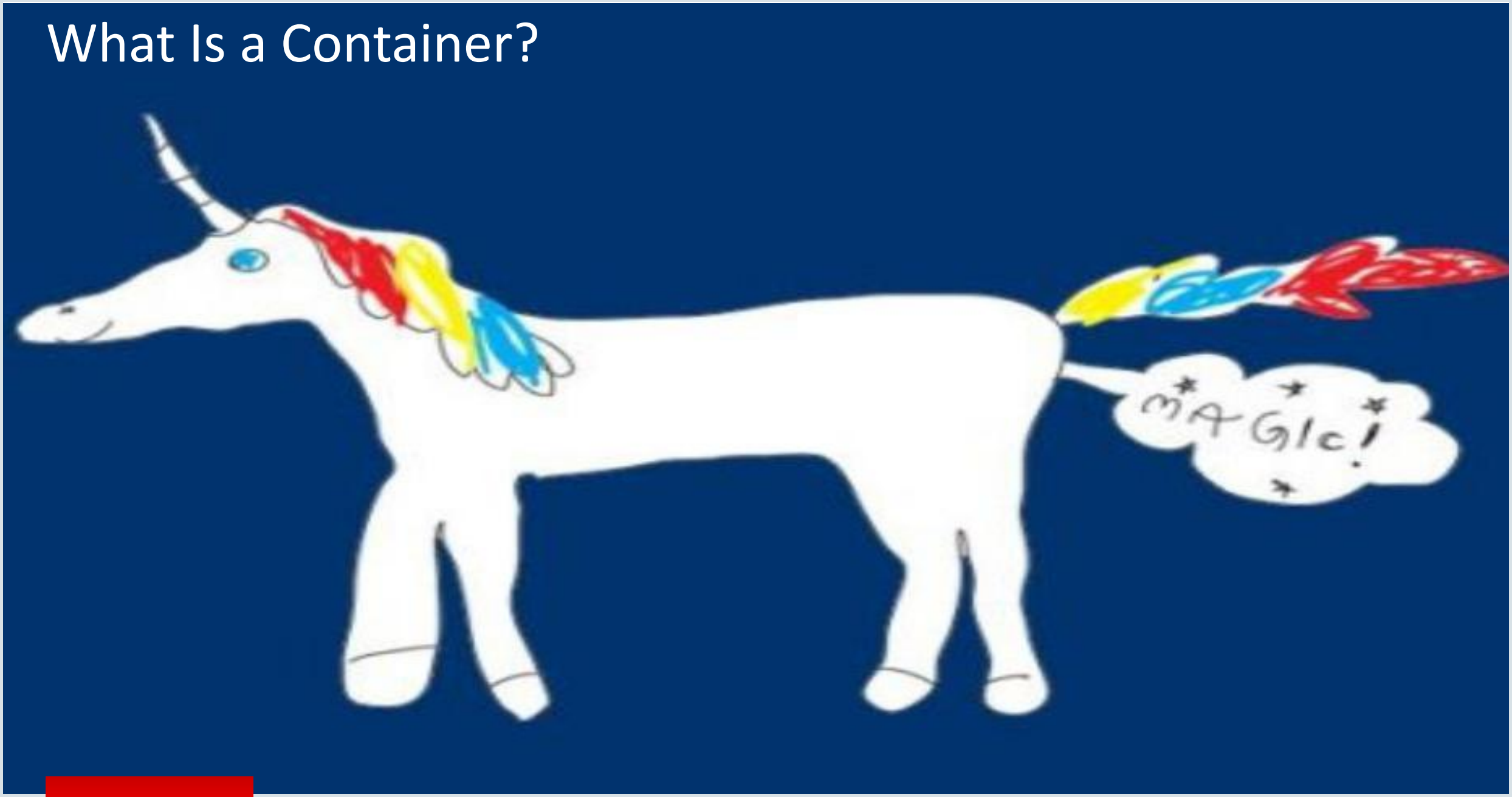
1. What are Containers?
2. What is Docker?
3. Why Should I Care?
4. Official MySQL Containers
5. Where Do I Go From Here?

What Is a Container?

"A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings. Available for both Linux and Windows based apps, containerized software will always run the same, regardless of the environment. Containers isolate software from its surroundings, for example differences between development and staging environments and help reduce conflicts between teams running different software on the same infrastructure."

https://www.docker.com/what-container

What Is a Container?



What Is a (Linux) Container? More Technically

Process level isolation – creates an isolated execution context for the process--the *virtual* world that the process *sees*

- Kernel Namespaces: Process tree (pid), Filesystem (mnt), Network (net), Users (user), Hostnames (uts) , Shared Memory / Message Queues (ipc) ; see: `/proc/{pid}/ns/, nsenter, share, unshare`
- Kernel Control Groups (cgroups): allows system resource metering and limiting for the namespaces ; see: `/sys/fs/cgroup/`
- IPTables/Netfilter: allows for network isolation with virtual networks ; see `brctl, iptables`
- Copy-On-Write: sharing and layering file systems to keep things small

An Example: Memcached

Memached OS Package

RHEL 6.9 x86_64 RPM

-> Package deps: `yum deplist memcached | wc -l` == *1551*

-> Each dep package with its own deps

-> Environment: shell, ENV variables, aliases...

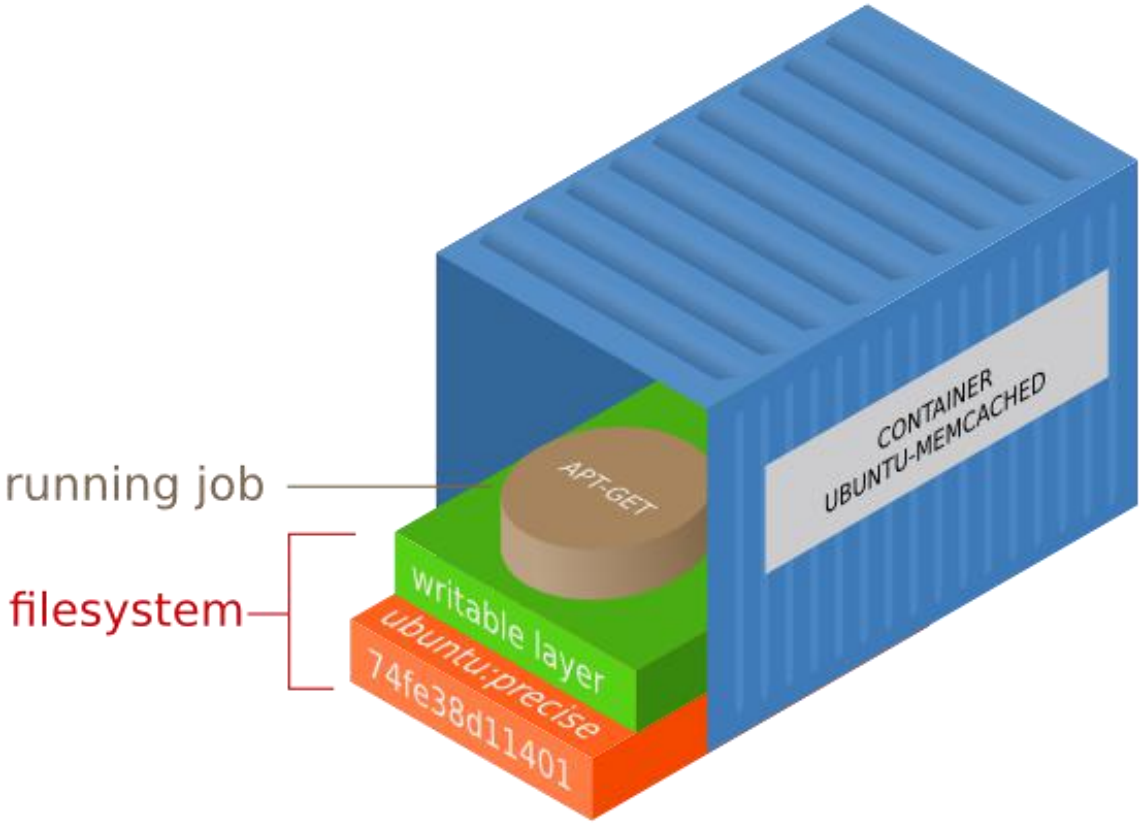
-> File systems

-> *Virtually unlimited* OS customizations: users, processes, networks,

(This is a relatively simple binary/package)

We hope the memcached binary ends up working

Memcached Container



We are confident the memcached binary ends up working

Walkthrough: Install and Start MySQL 5.7 on macOS

1. Download the package: e.g. mysql-5.7.19-macos10.12-x86_64.dmg
2. Install the package
   A. Lays down the files in /usr/local/mysql-{version}
   B. Installs the MySQL System Preferences Pane
3. Optional: Enable the launchd service
   A. `launchctl load -F /Library/LaunchDaemons/com.oracle.oss.mysql.mysqld`
4. Start MySQL
   A. `/usr/local/mysql/bin/mysqld`
   B. `launchctl start com.oracle.oss.mysql.mysqld`

1. `docker run -d mysql/mysql-server`

Go be productive and enjoy life…

Walkthrough: What's Now Running?

A native macOS mysqld binary on the host OS

A mysqld binary running inside of an Oracle Linux 7 container.

That container is running inside of an Alpine Linux VM, managed by the macOS native hypervisor: xhyve

Program Agenda

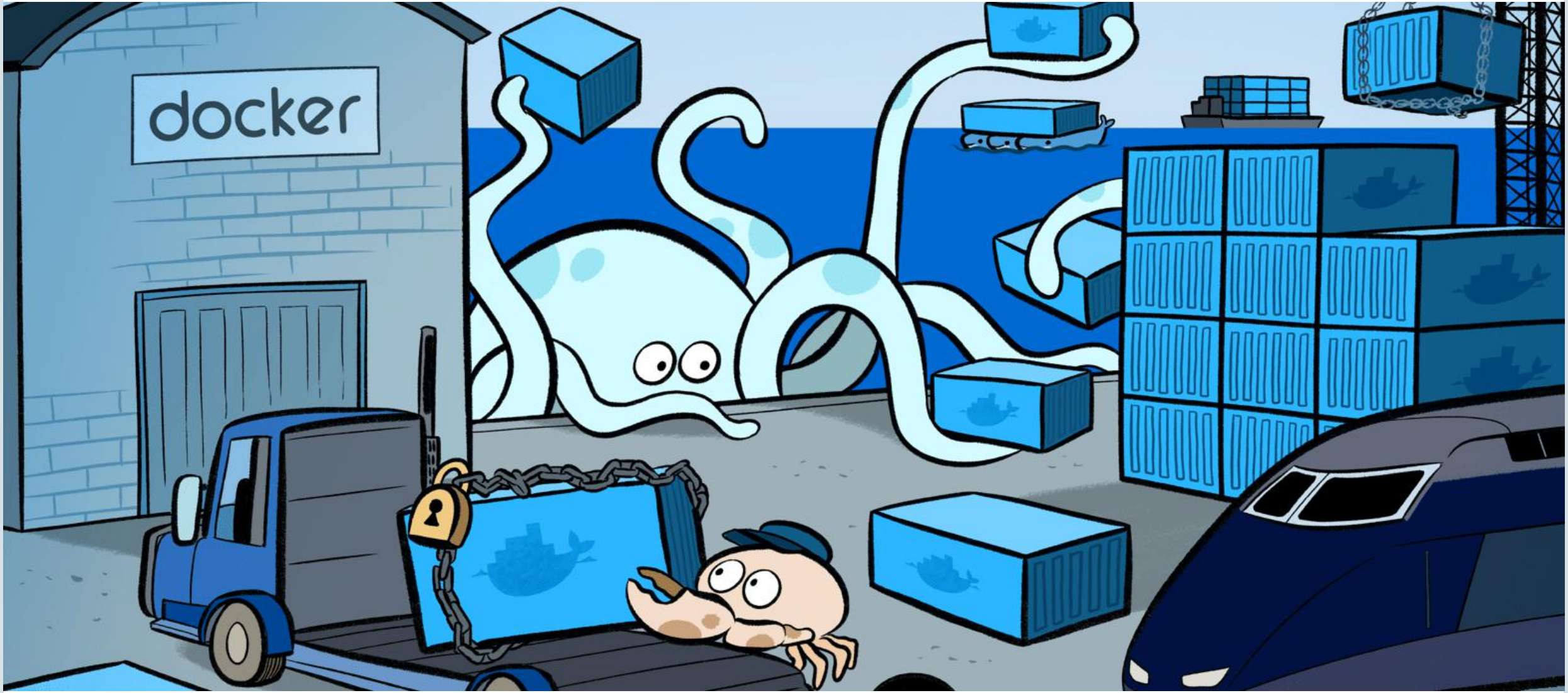
1. What are Containers?
2. What is Docker?
3. Why Should I Care?
4. Official MySQL Containers
5. Where Do I Go From Here?

Docker: What Is It?

"Docker is the world's leading software container platform. Developers use Docker to eliminate "works on my machine" problems when collaborating on code with co-workers. Operators use Docker to run and manage apps side-by-side in isolated containers to get better compute density. Enterprises use Docker to build agile software delivery pipelines to ship new features faster, more securely and with confidence for both Linux, Windows Server, and Linux-on-mainframe apps."

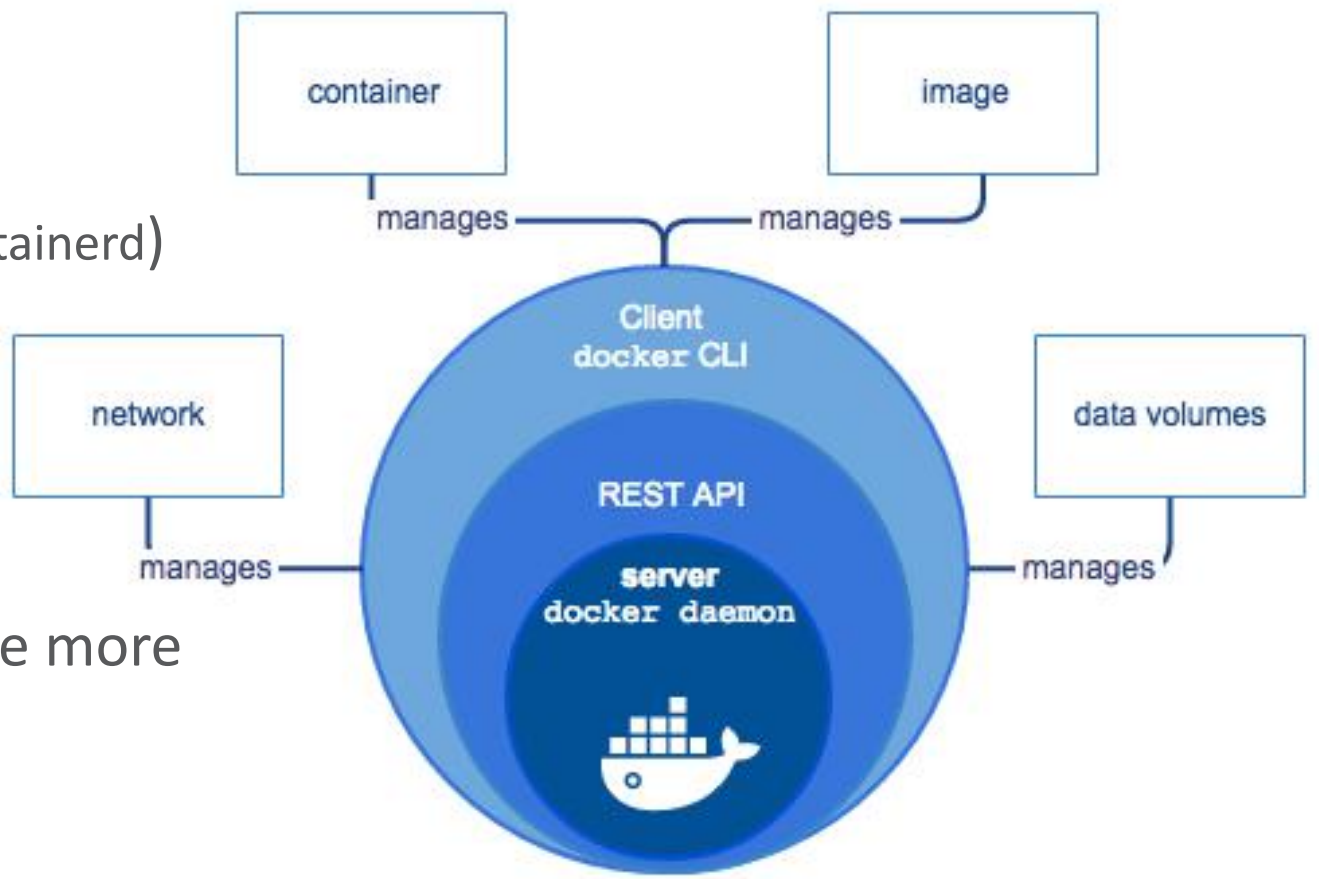
https://www.docker.com/what-docker

Docker: Everything That Makes Containers Useful



Docker: What Is It?

- The Docker Engine:
  - The dockerd server (dockerd, docker-containerd)
  - A REST API to interact with the server
- The docker command-line client
- Optional orchestration tooling
  - Compose and Swarm help you manage more complex container based applications
- The Docker SaaS platform
  - identity, repositories, versioning, change management, etc. (rough equivalent of what GitHub offers Git users)



Docker Containers: Use Cases

- **For Developers**
  - Docker automates the repetitive tasks of setting up and configuring development environments so that developers can focus on what matters: building great software.
- **For Operations**
  - Docker streamlines software delivery. Develop and deploy bug fixes and new features without roadblocks. Scale applications in real time.
- **For the Enterprise**
  - Docker is a Containers-as-a-Service platform for the enterprise that manages and secures diverse applications across disparate infrastructure, both on-premises and in the cloud. Docker EE fuels innovation by bringing traditional applications and microservices built on Windows, Linux or Linux-on-mainframe into a single, secure software supply chain. With Docker, organizations can modernize applications, infrastructure and operational models by bringing forward existing IT investments while integrating new technology at the rate of business.

Program Agenda

1. What are Containers?
2. What is Docker?
3. **Why Should I Care?**
4. Official MySQL Containers
5. Where Do I Go From Here?

Why Should I Care

- Say goodbye to the “it worked on my computer” problems
  - Package your software in a single binary image
    1. All of its dependencies are included
    2. Contains a repeatable execution context
    3. Is easily tagged, shared, distributed, versioned
    4. Can be nicely managed with version control systems
- Move to modern agile CI/CD development and delivery models
  - Everything is repeatable
    - Build confidence in your software development and delivery processes
    - Test exactly the same thing that you deploy, building *trust and agility*
  - Helping you *potentially* release bug fixes and new features *every single day*
    - Making for happy customers *and* a happy Sales team

Program Agenda

1. What are Containers?
2. What is Docker?
3. Why Should I Care?
4. Official MySQL Containers
5. Where Do I Go From Here?

Official MySQL Containers: Goals

- Official Server Release product
  - Part of each release, e.g. 5.7.20
  - Community and Enterprise
  - Fully supported
- Containers for all products
  - MySQL (NDB) Cluster
  - InnoDB Cluster
  - Router, Shell, Workbench, Utilities, …
- Each product and container
  - Being designed with production usage in mind
  - Documented well, with reference manual and user guide style artifacts
- Work well with Compose, Swarm, Kubernetes
- Integrate with wider Oracle plans around Docker and Container Services

Let’s Jump

Right In

Program Agenda

1. What are Containers?
2. What is Docker?
3. Why Should I Care?
4. Official MySQL Containers
5. Where Do I Go From Here?

Making Your Own Containers: Dockerfile

- Dockerfile – where you define the container (also corresponding CLI arguments)
  - ARG : arguments for use within the Dockerfile
  - ENV : set environment variables for the container
  - RUN : run command inside the container
  - VOLUME : define volumes/mount points in the container
  - ADD / COPY: add/copy files from the host to the container
  - ENTRYFILE : where you define what's run in the container when it starts
  - CMD : the process to run inside the container
  - EXPOSE : expose ports from the container
  - HEALTHCHECK : exec something periodically to check the health of the container

https://docs.docker.com/engine/reference/builder/

Making Your Own Containers: Dockerfile Example

```
FROM oraclelinux:7-slim
ARG PACKAGE_URL=https://repo.mysql.com/…/mysql-community-server-minimal-5.7.19-1.el7.x86_64.rpm
ARG PACKAGE_URL_SHELL=https://repo.mysql.com/…/mysql-shell-1.0.9-1.el7.x86_64.rpm

# Install server
RUN rpmkeys --import https://repo.mysql.com/RPM-GPG-KEY-mysql \
  && yum install -y $PACKAGE_URL $PACKAGE_URL_SHELL libpwquality \
  && yum clean all \
  && mkdir /docker-entrypoint-initdb.d

VOLUME /var/lib/mysql

COPY docker-entrypoint.sh /entrypoint.sh
COPY healthcheck.sh /healthcheck.sh
ENTRYPOINT ["/entrypoint.sh"]
HEALTHCHECK CMD /healthcheck.sh
EXPOSE 3306 33060
CMD ["mysqld"]
```

Making Your Own Containers: Docker CLI Examples

```
docker run --network=grnet -v ./secretpassword.txt:/root/secretpassword.txt -e
MYSQL_ROOT_PASSWORD=/root/secretpassword.txt -e SERVER_ID=200 --name=mysqlgr2 --hostname=mysqlgr2
--network-alias=myinnodbcluster -e $GROUP_PARAM -e GROUP_SEEDS="mysqlgr1:6606" -itd
mattalord/innodb-cluster

docker run --name=mysqlgr3 --hostname=mysqlgr3 --network=grnet -e MYSQL_ROOT_PASSWORD=root -e
GROUP_NAME="a94c5c6a-ecc6-4274-b6c1-70bd759ac27f" -e GROUP_SEEDS="mysqlgr1:6606" --health-
cmd='mysql -nsLNE -e "select member_state from performance_schema.replication_group_members where
member_id=@@server_uuid;" 2>/dev/null | grep -v "*" | egrep -v "ERROR|OFFLINE" || exit 1' --
health-start-period=60s --health-timeout=10s --health-interval=5s --health-retries=1 -itd
mattalord/innodb-cluster

docker run --rm --name=myarbitratord --network=grnet --hostname=myarbitratord --
entrypoint=/bin/bash -itd golang -c "go get github.com/mattlord/myarbitratord &&
/go/bin/myarbitratord -mysql_password '$(cat secretpassword.txt)' -seed_host myinnodbcluster"
```

Managing Multi-Container Apps: Docker Compose

```
$ cat docker-compose.yml
version: '3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
volumes:
    db_data:
```

Managing Multi-Container Apps: Docker Compose

```
$ docker-compose up -d
Creating network "my_wordpress_default" with the default driver
Pulling db (mysql:5.7)...
5.7: Pulling from library/mysql
efd26ecc9548: Pull complete
a3ed95caeb02: Pull complete
...
Digest: sha256:34a0aca88e85f2efa5edff1cea77cf5d3147ad93545dbec99cfe705b03c520de
Status: Downloaded newer image for mysql:5.7
Pulling wordpress (wordpress:latest)...
latest: Pulling from library/wordpress
efd26ecc9548: Already exists
a3ed95caeb02: Pull complete
589a9d9a7c64: Pull complete
...
Digest: sha256:ed28506ae44d5def89075fd5c01456610cd6c64006addfe5210b8c675881aff6
Status: Downloaded newer image for wordpress:latest
Creating my_wordpress_db_1
Creating my_wordpress_wordpress_1
```

Container Orchestration: Docker Swarm

- A cluster of Docker Engines
- IMO, just use Kubernetes
  - It's more advanced and more powerful (albeit more complex)
  - It's more battle tested in production
  - It has a huge and constantly growing community
  - Good support with IaaS providers (Amazon, Google, Microsoft, Oracle, IBM, OpenStack)
  - Enterprise "flavors" available: e.g. CoreOS Tectonic, RedHat OpenShift
  - Now has a mechanisms for intelligently handling stateful apps like MySQL
    - StatefulSets and Operators

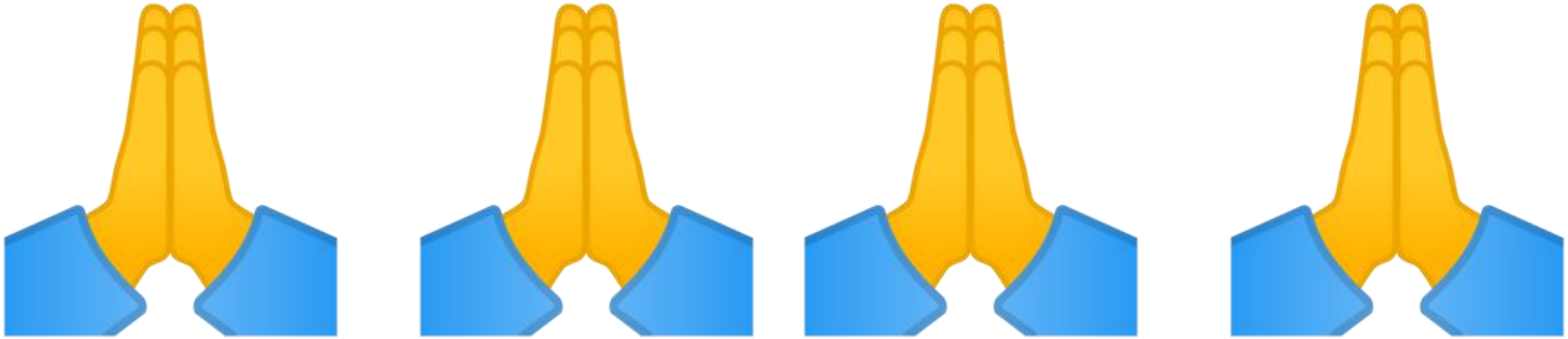
Container Orchestration: Kubernetes (K8s)

It makes *pizza toast*!

- Containers → Docker → Kubernetes = "The (Future) Way"
  - "The Way" to develop, test, deploy, and operate (Linux) software
    - Everything is *app* driven rather than machine driven
  - That future is now for Google and a growing list of young/fast/modern companies
  - Still figuring out how best to manage *VERY* stateful applications such as Databases
  - Side note: Terraform to provision and manage the infrastructure you run k8s on is also gaining a lot of traction and momentum
- I won't do k8s, Kelsey Hightower (so many good talks), and the rest of the community the disservice …
  - To learn about Kubernetes, you can start here: https://www.youtube.com/watch?v=pozC9rBvAIs

Summary and Conclusion

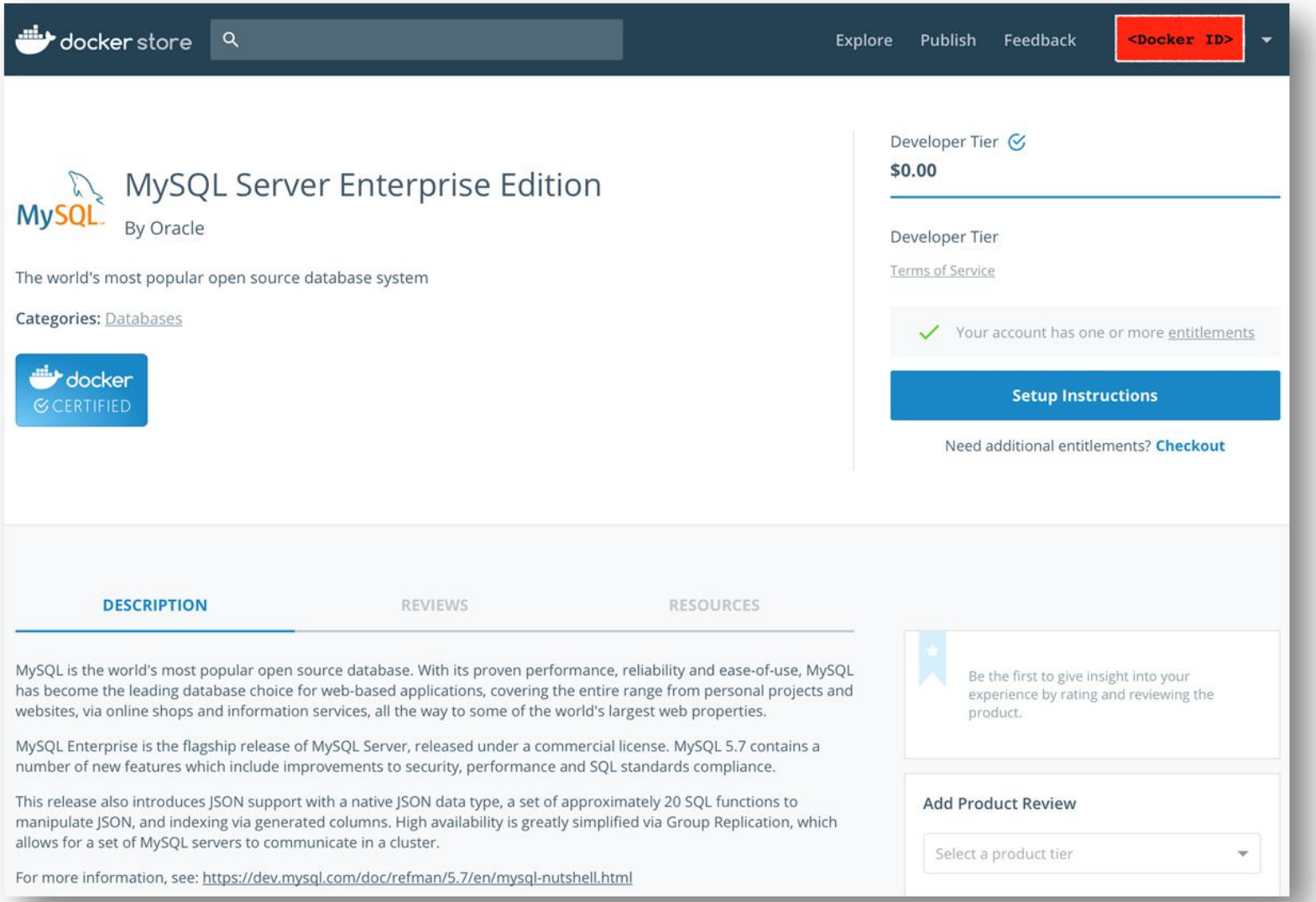
1. Use the official MySQL Docker Containers!
   - For fun and profit
2. Provide us with feedback on how we can improve them!



Appendix

MySQL Enterprise: Now in Docker Store!

"Build, Ship, and Run your modern applications with Docker and MySQL Enterprise!"



http://blogs.oracle.com/MySQL/entry/mysql-enterprise-edition-now-in-docker-store

Oracle Cloud: MySQL Service

- MySQL Enterprise Edition
- Web based console to manage your MySQL Cloud instances
- Self-Service Provisioning
- Elastic Scalability
- Multi-Layered Security
- Unified Cloud Management
- Oracle PaaS and IaaS Integration
- Premier Technical Support included

Integrated HA and DR service options – utilizing InnoDB clusters – coming soon!

Oracle Cloud: Container Services

- Container Service (classic)
  - https://cloud.oracle.com/container
- Application Container Cloud
  - https://cloud.oracle.com/en_US/application-container-cloud
- Stay tuned….


ORACLE®