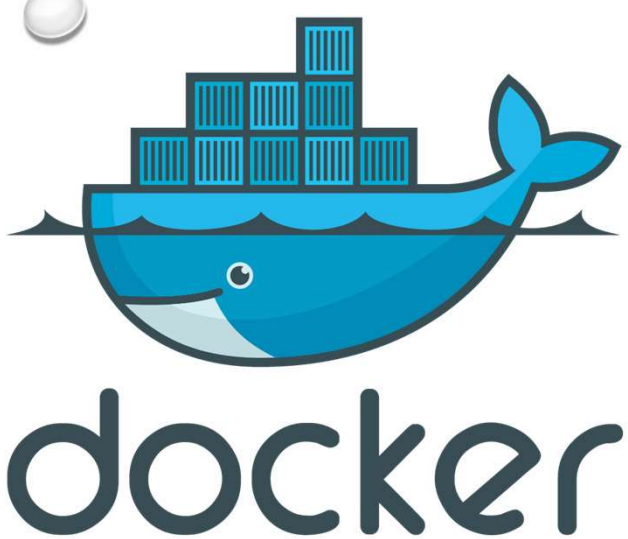
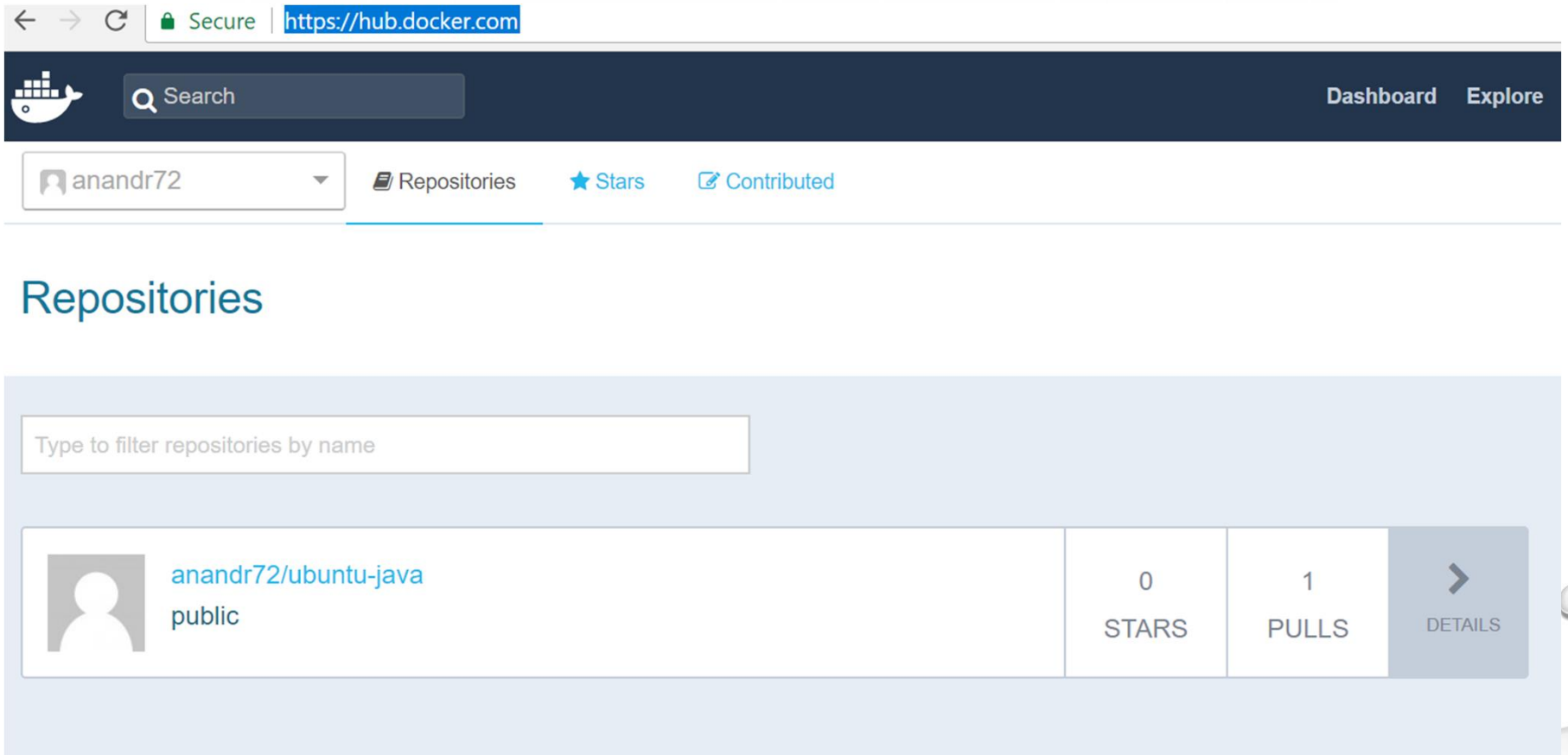


By Anand

Docker Hub

- https://hub.docker.com/
- Central repositories for Public (free) as well as Private (paid) images
- First register yourself on Docker Hub – username & passwd
- >docker login -> provide your credentials & check
- Build & save image as Username/Imagename
- >docker push Username/Imagename

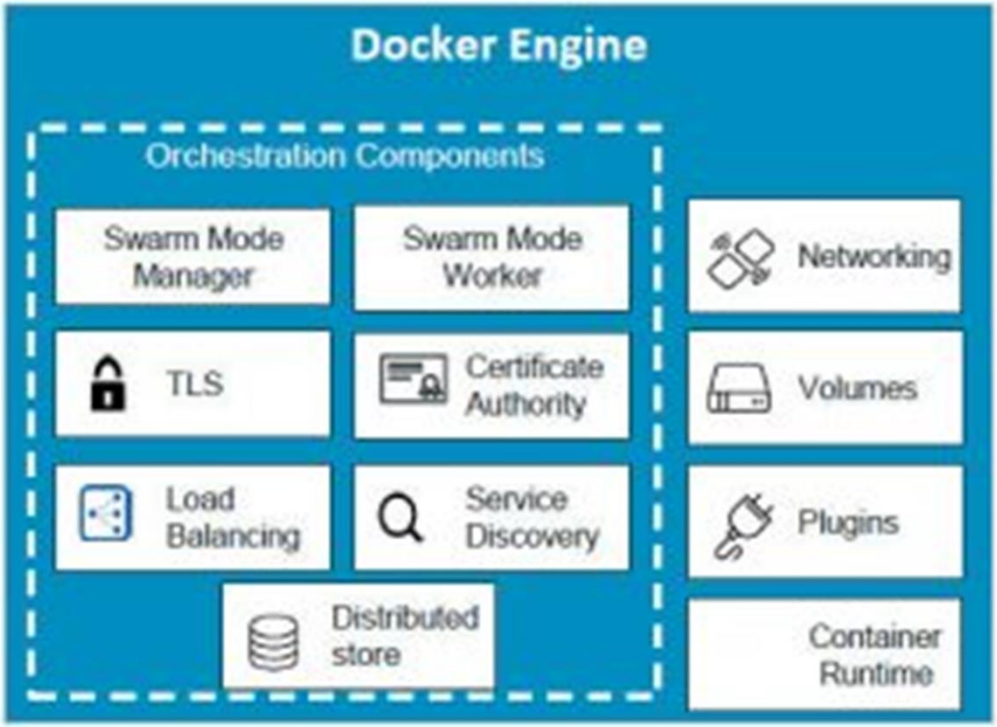
Docker Hub


Secure
https://hub.docker.com
Search
Dashboard
Explore
anandr72
Repositories
Stars
Contributed
Repositories
Type to filter repositories by name
anandr72/ubuntu-java
public
0
STARS
1
PULLS
DETAILS

Docker Compose

- Is a tool for defining and running multi-container Docker applications
- Manually wiring each containers is painful and so, Docker compose is used to stitch up multiple-dependent containers and bring them up in one shot

Real Time Scenario



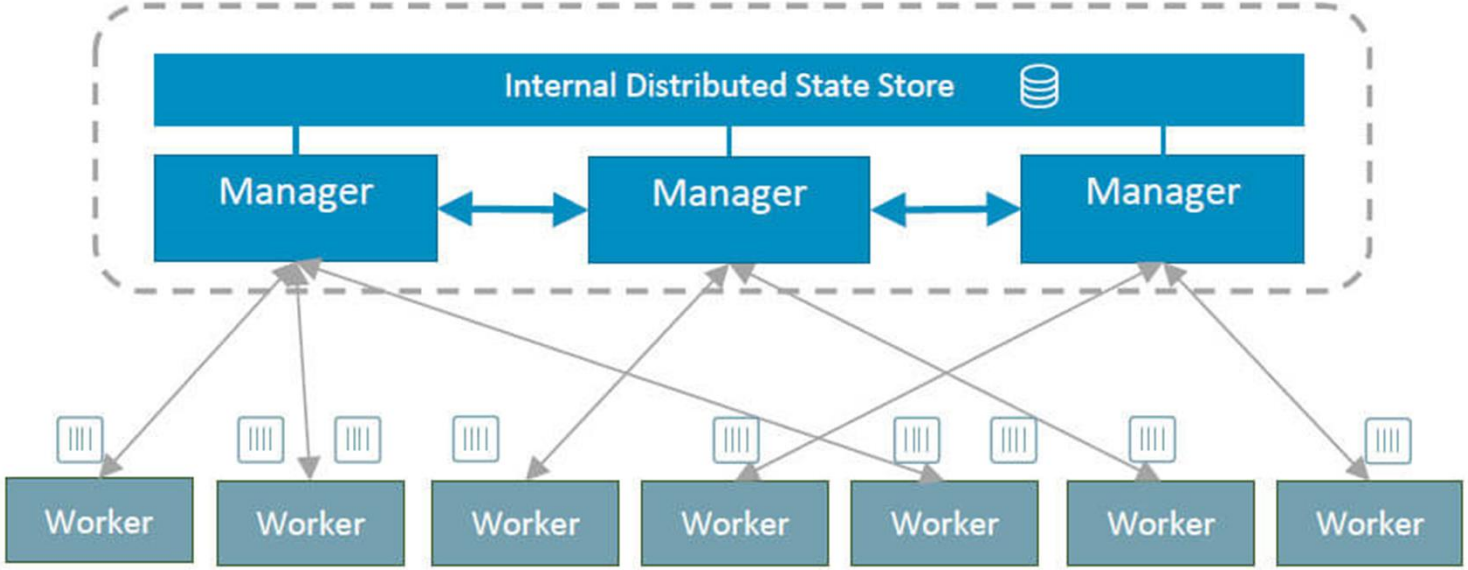
Orchestration of Containers : A node in a self-organizing cluster of similar engines across multiple servers, which can work together as a single scalable system to run large fleets of interconnected Docker containers.

Docker Swarm & Orchestration

- Real time production scenarios involve hundreds of containers
- Heath Checks for Containers
- Scaling containers (up & down) based on load
- Performing rolling updates of software across containers
- High Availability & Fault Tolerance

And so on…….

Docker Swarm



Docker Swarm & Service

Let us create a Swarm of 3 nodes – 1 Manager & 2 workers and start a web service (nginx server) on it at port 80

Docker-machine is a tool used to create virtual hosts on local box and lets you manage it using $docker-machine commands

```
$docker-machine create --driver virtualbox manager1
$docker-machine create --driver virtualbox worker1
$docker-machine create --driver virtualbox worker2
```

Docker Swarm & Service

Check the IPs of these virtual hosts

$docker-machine ip manager1

$docker-machine start manager1 (if not already started)

Use separate terminals to SSH to manager1, worker1 & worker2

$docker-machine ssh manager1

$docker-machine ssh worker1 (different terminal)

$docker-machine ssh worker2 (different terminal)

Docker Swarm & Service

On the manager1 terminal, type in the following:

Start docker storm with IP of manager1 instance

$docker swarm init --advertise-addr YOUR IP ADDRESS

This will start the swarm with manager1 as a manager

Docker Swarm & Service

To check how some other node can join this swarm:

$docker swarm join-token worker -> How to join this swarm as a worker

$docker swarm join-token manager -> How to join this swarm as a manager

Use worker specific terminals to join this swarm as workers

Docker Swarm & Service

Let us now start a service (nginx web server) on this swarm

Type in this in the manager console:

$docker service create --replicas 2 -p 80:80 --name web nginx

Check if this service indeed started

$docker service ls

$docker service ps web

Notice the status of the services, nodes on which they are running

Wait till all replicas start running this service

Docker Swarm & Service

Go to each terminal (worker/manager) which has this service running and check $docker ps

Get the Ips of all 3 docker-machine which are in the Swarm and using a browser, check what's running on port 80

Docker Service Scale

Scaling Up/Down a running service is very easy & is accomplished with docker service scale command

$docker service scale web=8

This will bump up our web service running instances to 8 (arbitrarily distributes loads across nodes that are running & READY to accept more work from manager)

Docker Swarm drain a Node

Bringing down instances or nodes is also easy and when ever an instance goes down or is brought down, the manager automatically shares the work load with other ACTIVE nodes

$docker node ls

If any node is Ready & its Availability is ACTIVE, it means its ready to take on work

Let us forcibly bring down worker1

$docker node update --availability drain worker1

Check if other nodes have picked up extra work and if we still have 8 instances of this service running

Docker Swarm – node Available

To re-instate that node which was drained, back to available

$docker node update --availability active worker1

Check if this node is back

$docker node ls

Docker Swarm – shutting down service

To shut down a running service

$docker service rm web

Check if anything is running at all on manager1, worker1 or worker2

Docker Swarm – Service Update

Applying patch or updating service that is running on a swarm is also very easy

$docker service update --image <imagename>:<version> web

Database Example - MySQL

$docker run --detach --name=db --env="MYSQL_ROOT_PASSWORD=mypassword" mysql

Or to start it on a specific port

$docker run --name db -d -e MYSQL_ROOT_PASSWORD=mypassword -p 3306:3306 mysql:latest

Once mysql instance starts up, use exec to get to its bash

$ docker exec -it db /bin/bash

Web App Example - Jenkins

Jenkins is a self contained open source automation server which can be used for automation of all sorts of software tasks like building, testing & deploying software

It's a Java bundle that can be run within a WebServer (Tomcat) or as a standalone Java App

Continuous Integration Server

Jenkins Installation

Data + port segregation

- All Jenkins Application related data gets stored in 'Jenkins_home' directory
- It needs a port to run on – typically 8080

Jenkins Installation

Create a directory (on host machine) where all Jenkins related data – Workspace, configuration etc will be stored

Map this folder as jenkins_home folder

```
$docker run -d -p 8080:8080 -v
$PWD/jenkins:/var/jenkins_home:z -t Jenkins
$docker-machine ip default
```

Dockerize Jenkins

- Configure Jenkins
- Put in some sample job
- Check if workspace contains all information about the sample job
- Create few users & check if all of them are correctly provisioned
- Log out, stop the container & log back in; check if all configuration & data is persisted

- How can you share this instance with other users?

Questions

?