Immutable Containerized infrastructure

Atomic, Docker, K8s, Nulecule and Cockpit

Muayyad AlSadi - Principal Developer
Opensooq.com

Cluster - master.k8s

localhost:9090/kubernetes#/topology

FEDORA **CLOUD EDITION**

alsadi

Machines | Dashboard | Cluster

Overview
Containers
Topology
Details
Images

All Namespaces

Adjust Replication Controller

Replicas 6

Cancel | Adjust

ReplicationController

Name	www
Namespace	default
Created	Mar 2, 2016 9:30:43 PM
Replicas	6

Selector

run	www

Pod Template

Restart policy	Always
DNS policy	ClusterFirst

Containers

Name	www
Image	nginx

Atomic Host

not just a minimal OS

immutable

stateless

disposable

too fast (2 seconds to boot)

too light (no ram, no daemons)

no package manager

no configuration

image-based upgrades

Not just minimal OS

Why atomic hosts?

It force you to change your culture to adopt cloud and containers the right way.

No config you have to use discovery service (ex. etcd).

No package manager, you have to use containers.

Google cloud experience

Top secret infrastructure

google used containers for ~10 years to have better hardware utilization

Borg 2.0

Open sourced

Google offers cloud computing using K8s

see this

Containers, containers, containers.

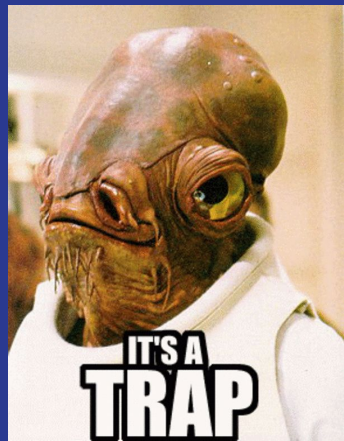
Why containers?

Because well-designed containers and container images are scoped to a single application, managing containers means managing applications rather than machines. This shift of management APIs from machine-oriented to application-oriented dramatically improves application deployment and introspection.

Docker

a working trap

Docker.com sells “platform as a service” and it’s a vendor lock-in, it’s a trap. For us docker is just an exchange format just like a tarball or zip file.



Docker in seconds

Docker in seconds

```
docker images
docker pull nginx # from https://hub.docker.com/
docker run -d --name x1 nginx
docker ps
docker inspect x1
docker exec -ti x1 /bin/sh

# READ: how to mount directories inside docker, link
containers, pass env variables, docker diff and docker
commit ...
```

Dockerfile in seconds

Dockerfile is a simple tells “docker build” how to build an image

```
FROM fedora
MAINTAINER webmaster@opensooq.com
RUN dnf -y update && dnf -y install nginx && dnf clean all
EXPOSE 80
CMD [ "/usr/sbin/nginx", "-g",  "daemon off;" ]
```

Docker Compose in seconds

Docker compose is a yaml file for multi-container app, here is wordpress and its database: **docker-compose -f wordpress.yml up -d**

```
wordpress:
  image: wordpress
  links:
    - db:mysql
  ports:
    - 8080:80
db:
  image: mariadb
  environment:
    MYSQL_ROOT_PASSWORD: example
```

Docker Compose in seconds

Docker compose can build Dockerfiles or mount code too (no need to rebuild)

```
web:
  build: .
  ports:
    - "5000:5000"
  volumes:
    - .:/code
  links:
    - redis
redis:
  image: redis
```

Docker-packed application

A source tree containers main “docker-compose.yml” and sub-directories with multiple projects and services with Dockerfile for each (maybe git submodules)

“docker-compose up” and opensooq will be up and running with all of its supporting services

Important reading “12 factor” for example “Strictly separate build and run stages”

Kubernetes in seconds

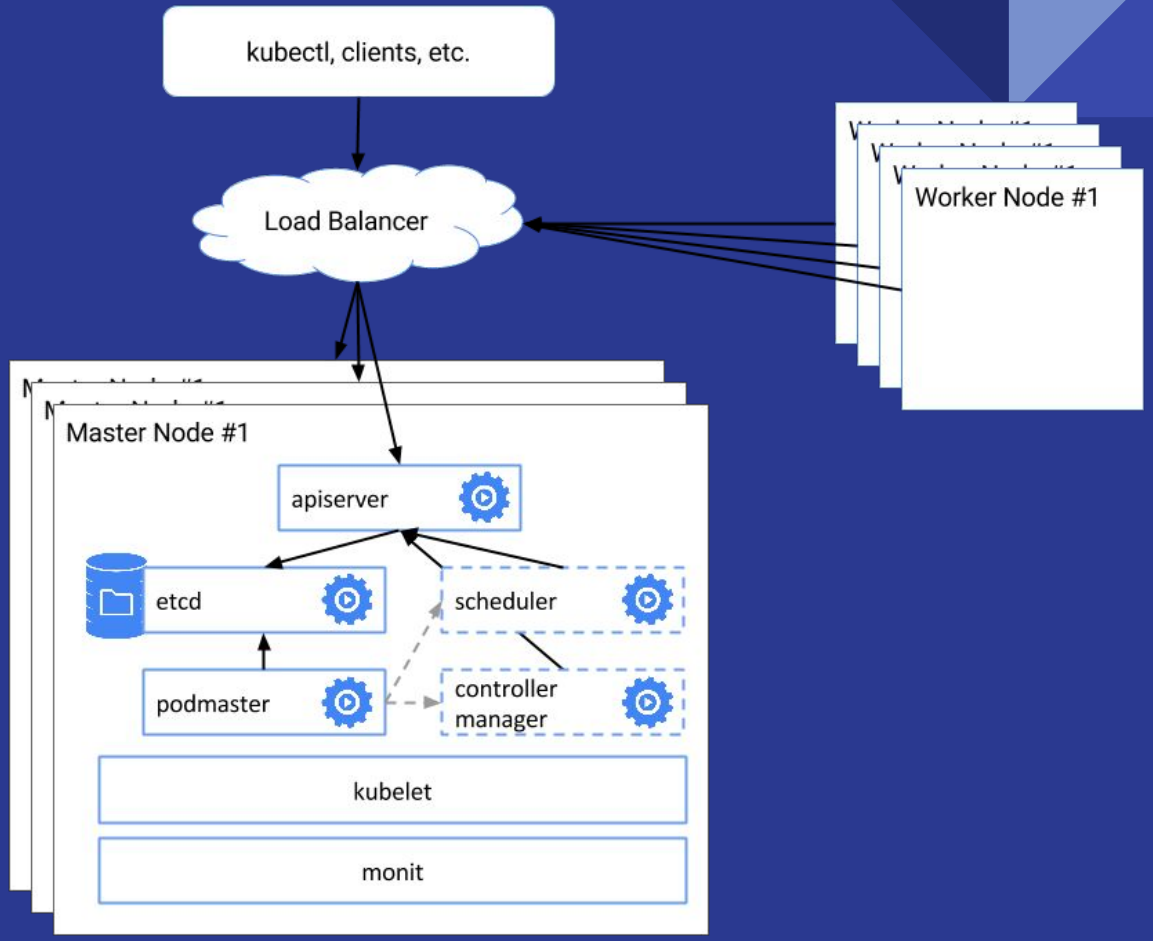
K8s cluster

Nodes running kubelet are called minions. Minions also run kube-proxy.

A master node runs “kube-apiserver”, “kube-controller-manager” and “kube-scheduler”

Master node does not moving parts, it just pass work to minions. Master node can be a minion. HA Multimaster can be set into the cluster using “podmaster” container.

K8s deal with the cluster as whole using its client kubectl see this document for how to use it


kubectl, clients, etc.
Load Balancer
Worker Node #1
Master Node #1
apiserver
etcd
scheduler
podmaster
controller manager
kubelet
monit

K8s in seconds

```
kubectl cluster-info
kubectl run --replicas=2 -- image=nginx x1
kubectl create -f all-in-one.yml
kubectl get pods
kubectl get pods -l 'environment in (production, qa)'
kubectl get rc
kubectl get service
kubectl expose rc nginx-app --port=80 --name=nginx-http
kubectl exec nginx-app-XYZ -- cat /etc/hostname
kubectl exec -ti nginx-app-XYZ -- /bin/sh
kubectl logs -f nginx-app-XYZ
```

K8s Manifests JSON/Yaml

see all in one guest book application which uses replicated web server and redis and redis slave
see other examples

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
      app: nginx
  template:
      metadata:
      name: nginx
      labels:
      app: nginx
      spec:
      containers:
      - name: nginx
      image: nginx
      ports:
      - containerPort: 80
```

```json
{
  "apiVersion": "v1",
  "kind": "ReplicationController",
  "metadata": {
    "name": "nginx"
  },
  "spec": {
    "replicas": 3,
    "selector": {"app": "nginx"},
    "template": {
      "metadata": {
        "name": "nginx",
        "labels": {"app": "nginx"}
      },
      "spec": {
        "containers": [
          {
            "name": "nginx",
            "image": "nginx",
            "ports": [ {"containerPort": 80} ]
          }
        ]
      }
    }
  }
}
```

PROJECT

ATOMIC

Create and Run Applications in Linux Containers

Create your application using Docker containers. Deploy and manage containerized applications on a proven, trusted platform.

Project Atomic introduces Atomic App — an implementation of the Nulecule specification, which lets you manage multi-container applications and orchestration metadata as easily as you manage RPMs.

LEARN MORE! →

Atomic App

With Atomic App, you can use existing containers as building blocks for your new application product or project.

Databases, web servers, and other common components are vital parts of applications and services. Utilizing existing containers to provide these core infrastructure components lets you focus more on building the stuff that matters and less time packaging and setting up the common plumbing required.

Learn more about Atomic App

Nulecule /NOO-le-kyul/ (noun)

Nulecule is a made-up word meaning "the mother of all atomic particles". Sounds like "molecule". But different.

Also a specification for applications composed from multiple containers. Check it out on Github below, or read through the Getting Started -guide if you want to know more.

Learn more about Nulecule

Atomic Host

Based on proven technology either from Red Hat Enterprise Linux or the CentOS and Fedora projects, Atomic Host is a lightweight, immutable platform, designed with the sole purpose of running containerized applications.

To balance the need between long-term stability and new features, we are providing different releases of Atomic Host for you to choose from.

Get Started

Software Updates

localhost:9090/@minion1.k8s/updates

FEDORA **CLOUD EDITION**

alsadi

minion1.k8s

Dashboard

Cluster

System

Services

Containers

Logs

Networking

Tools

Accounts

Terminal

Software Updates

Operating System Updates

Check for updates

fedora-atomic 23.74

Tree | Packages

Running

Operating Syst...	fedora-atomic
Version	23.74
Released	3 days ago

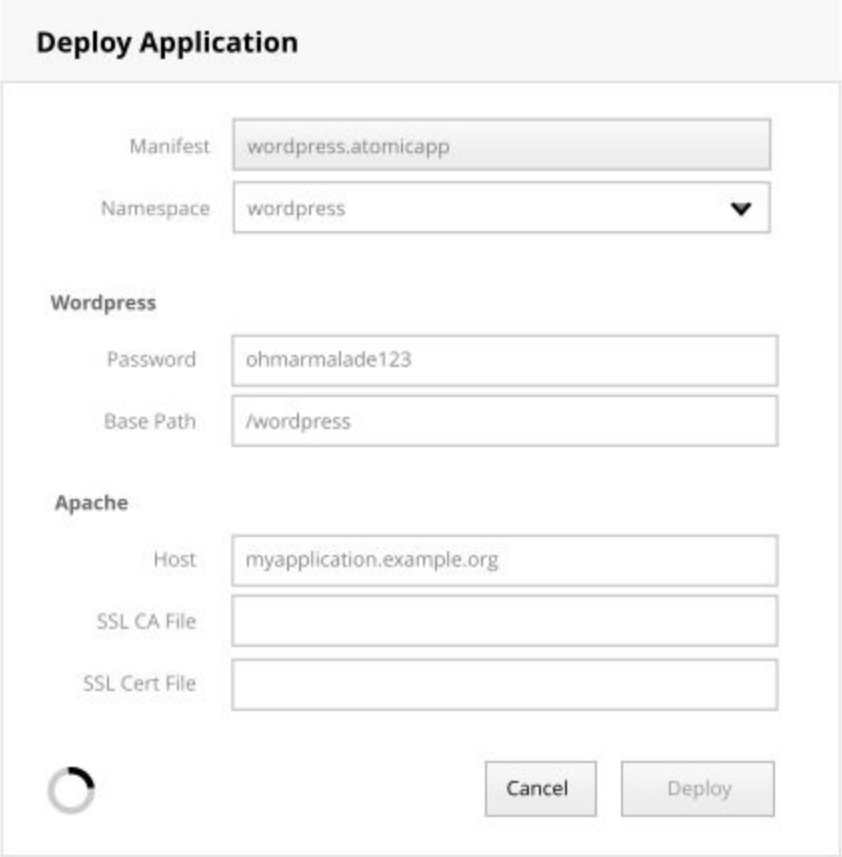
fedora-atomic 23.68

Roll back and reboot

Tree | Packages

Available

Operating Syst...	fedora-atomic
Version	23.68
Released	11 days ago
Downgrades	29 packages




```
atomic run projectatomic/mariadb-fedora-atomicapp

cp answers.conf.sample answers.conf

vim answers.conf

atomic run projectatomic/mariadb-fedora-atomicapp
```

Nulecule: A Composite Container-based Application Specification

Cockpit Project: A nice clustered UI

Demo Time

Check this blog and Youtube Video

Customizing Cockpit

put simple html/css in /usr/share/cockpit/

see https://github.com/cockpit-project/cockpit/tree/master/examples/pinger

```
{
    "version": 0,

    "tools": {
    "pinger": {
        "label": "Pinger",
        "path": "ping.html"
    }
    },

    "content-security-policy": "default-src 'self' 'unsafe-inline' 'unsafe-
eval'"
}
```

Pinger - localhost.loc

localhost:9090/pinger/ping

FEDORA **WORKSTATION EDITION**

root

localhost.local...

Dashboard

Cluster

System

Services

Containers

Logs

Storage

Networking

Tools

Pinger

Accounts

Terminal

Address 8.8.8.8

Ping success

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=56 time=14
4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=56 time=16
6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=56 time=13
9 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=56 time=21
0 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet los
s, time 3003ms
rtt min/avg/max/mdev = 139.225/165.103/210.642/2
8.242 ms
```