Managing and Orchestrating Docker Containers with OpenShift

Neale Ferguson
Sine Nomine Associates

Things You Need to Know

- There is a lot of material in these slides
- Far more than can be covered in 1 hour
- Provided as reference so you can explore more deeply
- I will highlight things I believe you need to understand
- Emphasis will be on showing it in action
- A whitepaper/red-piece is under construction

Preface

- Examples built and run using ClefOS 7.5.1804
  - CentOS Clone with name change
  - Available for z Systems
- However, as we will see this is irrelevant
- All OpenShift Origin containers are available on dockerhub under the clefos repository: https://hub.docker.com/u/clefos/dashboard/

Docker – underlying technologies

Things You Need to Know

- Docker and other container-based technologies rely on Linux kernel APIs to provide isolation and infrastructure
  - Cgroups
  - Copy-on-write
- A daemon is responsible for doing the work via a set of APIs (OCI-compliant)
- Storage is ephemeral unless otherwise specified

What is Docker

- An open source project to pack, ship and run any application as a lightweight container
- Container: self-contained receptacle
  - Filesystem
  - Apps
  - Static data
  - Network

Skip

cgroups...

- A kernel feature that limits, accounts for, and isolates the resource of a collection of processes
- Similar to processes:
  - They are hierarchical
  - Child cgroups inherit certain attributes from their parent cgroup
- Difference: multiple cgroup hierarchies

...cgroups...

- Can span multiple “subsystems”
  - blkio —sets limits on input/output access to and from block
  - cpu —uses the scheduler to provide cgroup tasks access to the CPU
  - cpuacct —generates automatic reports on CPU resources used by tasks in a cgroup.

...cgroups...

- Can span multiple “subsystems”
  - cpuset —assigns individual CPUs (on a multicore system) and memory
  - devices —allows or denies access to devices by tasks in a cgroup
  - freezer —suspends or resumes tasks in a cgroup

...cgroups...

- Can span multiple “subsystems”
  - memory —sets limits on memory use by tasks in a cgroup, & generates automatic reports on memory
  - net_cls —tags network packets with a class identifier (classid) that allows the Linux traffic controller (tc) to identify packets originating from a particular cgroup task.

...cgroups...

- Can span multiple “subsystems”
  - net_prio —provides a way to dynamically set the priority of network traffic per network interface
  - ns — the namespace subsystem

Namespaces...

- CLONE_NEWIPC: IPC Namespaces: SystemV IPC and POSIX Message Queues can be isolated.
- CLONE_NEWPID: PID Namespaces: PIDs are isolated, meaning that a virtual PID inside of the namespace can conflict with a PID outside of the namespace. PIDs inside the namespace will be mapped to other PIDs outside of the namespace. The first PID inside the namespace will be '1' which outside of the namespace is assigned to init

...Namespaces...

- CLONE_NEWNET: Network Namespaces: Networking (/proc/net, IPs, interfaces and routes) are isolated. Services can be run on the same ports within namespaces, and "duplicate" virtual interfaces can be created.
- CLONE_NEWNS: Mount Namespaces. We have the ability to isolate mount points as they appear to processes. Using mount namespaces, we can achieve similar functionality to chroot() however with improved security.

...Namespaces

- CLONE_NEWUTS: UTS Namespaces. This namespaces primary purpose is to isolate the hostname and NIS name.
- CLONE_NEWUSER: User Namespaces. Here, user and group IDs are different inside and outside of namespaces and can be duplicated.

Copy-on-Write

- Allows Docker to instantiate containers very quickly
- Instead of having to make full copies of the which files comprise a container, it can use “pointers” back to existing files
- Containers are easily “linked” (or “stacked” or “layered”) to other containers

Docker Registry (optional)

- A stateless, highly scalable server-side application that stores and distributes Docker images
- Enables:
  - Tight control where images are stored
  - Full ownership of distribution pipeline
  - Integration of image storage & distribution into an in-house development workflow

Docker Daemon

- Manages containers
  - Creates volumes
  - Starts/stops containers

```
docker daemon -H tcp://0.0.0.0:4243 -H unix:///var/run/docker.sock
       OR
systemctl enable docker
systemctl start docker
```

Docker – Building Containers

Things You Need to Know

- Everything starts with a base image
- Dockerfiles are text files with recipes for building images based on another image
- Images are held in a registry
- Dockerhub is the public repository
- There are official images that you can trust
- Otherwise… Buyer beware
- Images are run in containers which may be linked or grouped

Creating a Starter System

- Base image: containers built from it or descendants
- Create a chroot-like environment
  - File system including /dev
  - yum install packages
  - Trim unwanted stuff
  - Create tar ball
  - Import to Docker
- “Official Images” – Those accepted by Docker
  - ClefOS is now an official image

The Dockerfile

- A recipe for building a container
- Build from an existing container
- Install requirements
- Define network and volume requirements
- Specify command to run on startup

```
FROM    clefos:clefos7

MAINTAINER The ClefOS Project <neale@sinenomine.net>
LABEL   Vendor="ClefOS" License="GPLv2" Version="8.0-10.1"

COPY    ibm-java-sdk-8.0-1.10-s390x-archive.bin java.rsp dummy-java-1.8-0.el7.noarch.rpm
/

RUN     yum install -y tar zip && \
        mkdir -p /opt/ibm && \
        echo "Installing IBM JDK" && \
        /ibm-java-sdk-8.0-1.10-s390x-archive.bin -f /java.rsp -i silent && \
        yum install -y dummy-java-1.8-0.el7.noarch.rpm && \
        yum erase -y tar zip vim-minimal && \
        yum clean all && \
        rm -f /*.rpm /java.rsp /*.bin

ENV     JAVA_HOME=/opt/ibm/java PATH=$JAVA_HOME/bin:$PATH
```

```
FROM    clefos/nodejs
MAINTAINER      The ClefOS project <neale@sinenomine.net>
ADD     epel.repo /etc/yum.repos.d/epel.repo
RUN     yum install -y git tar gcc gcc-c++ make mongodb mongodb-server \
                mongo-tools krb5-devel perl-Digest-SHA && \
                npm install -g express && \
                npm install -g mongodb && \
                npm install -g tar mkdirp
WORKDIR /mean
EXPOSE  27017 28017
VOLUME  /mongodb/data
RUN     echo "mongod --fork --logpath /mongodb/data/log/mongod.log \
        --dbpath /mongodb/data --smallfiles --noprealloc --httpinterface --rest
\
        > /start.sh && echo "node \$1" >> /start.sh && \
        yum erase -y git tar gcc gcc-c++ make perl-Digest-SHA && \
        rm -f /etc/yum.repos.d/epel.repo && \
        rm -rf /tmp/* /var/cache/yum/* /root/* /root/.[a-zA-Z0-9]* /src
ENV     NODE_PATH=/opt/ibm/nodejs/lib/node_modules:/mean/node_modules
ENTRYPOINT ["sh", "/start.sh"]
```

Building Images

- Each step corresponds to a layer
- Stop build at one point
- Rebuild starts from last change

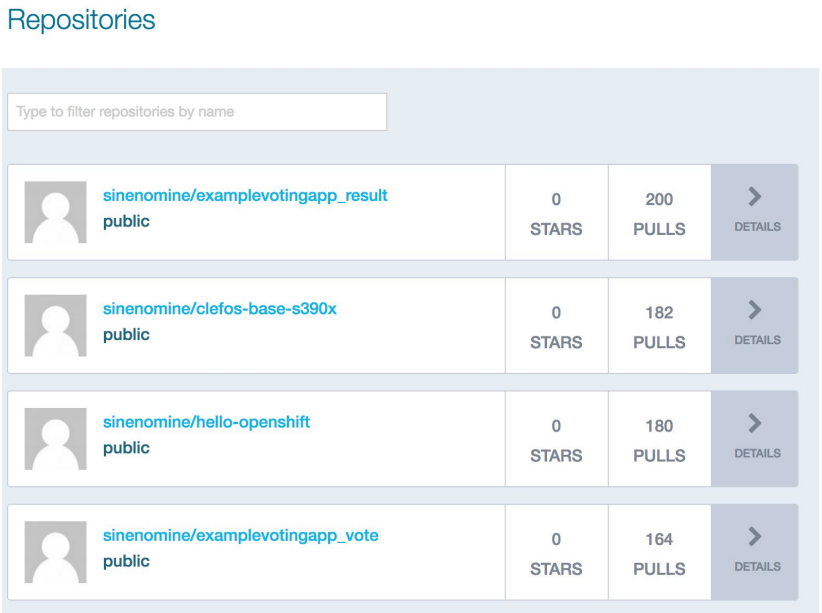
Managing Images

```
[root@docker docker]# docker images
REPOSITORY                          TAG     IMAGE ID     CREATED       VIRTUAL
SIZE
sinenomine/fluentd-s390x            latest  b3e3d646f313 4 days ago    515.2 MB
sinenomine/amhub-s390x              latest  76a2c4a387f0 7 days ago    795 MB
sinenomine/ade-s390x                latest  5dc6c7c6191c 5 weeks ago   645.8 MB
sinenomine/compose-ui-s390x         latest  ff5b9eda68ec 8 weeks ago   315.9 MB
sinenomine/nginx-1.8-s390x          latest  4f87e1292531 8 weeks ago   211 MB
sinenomine/clefos71-base-s390x      latest  60ef3a8ba174 3 months ago  110.5 MB
clefos-base-s390x                   latest  60ef3a8ba174 3 months ago  110.5 MB
sinenomine/clefos71-nodejs-s390x    latest  d76f12128dde 5 months ago  548.7 MB
sinenomine/mariadb-5.5-s390x        latest  91233ea5a5c1 5 months ago  311.3 MB
sinenomine/clefos71-java-s390x      latest  3cb8ef8fd562 5 months ago  480.2 MB
```

Making Images Available

```
[root@docker ~]# docker push sinenomine/fluentd-s390x:latest
The push refers to a repository [docker.io/sinenomine/fluentd-s390x] (len: 1)
b3e3d646f313: Pushed
1b11901fbead: Pushed
5f6ab7c78e8b: Pushed
288d092713a6: Pushed
f86e5eb99f4b: Pushed
:
d69fc3fad8fa: Pushed
732e18ef67b6: Pushed
7196f6de1451: Pushed
7118afa06d84: Pushed
ec3ec425b681: Pushed
60ef3a8ba174: Pushed
latest: digest:
sha256:120519d3d8f0cf00a0caddb3fd8c0c6148b8145dbf6fed2897b36e965d35424d size:
29665
```

Dockerhub


Repositories
Type to filter repositories by name
sinenomine/examplevotingapp_result
public
0
STARS
200
PULLS
DETAILS
sinenomine/clefos-base-s390x
public
0
STARS
182
PULLS
DETAILS
sinenomine/hello-openshift
public
0
STARS
180
PULLS
DETAILS
sinenomine/examplevotingapp_vote
public
0
STARS
164
PULLS
DETAILS

Running Containers

- Persistent data goes to [a] volume[s]
- Run a standalone container
  - All functionality within the container
- Run a “swarm” of containers
  - Typically database server
  - Web server
  - Application server

Running Containers

- `docker run --name=mariadb -v /var/local/mariadb:/var/lib/mysql -d -p 3306:3306 -e MYSQL_ROOT_PASSWORD=passw0rd sinenomine/mariadb-5.5-s390x:latest mysqld_safe --connect-timeout=30`

- `docker run --rm -i -t --name=ade -p 8022:22 --link=mariadb -v /var/local/ade:/var/local/ade -e MARIADB_ROOT_PASSWORD=passw0rd -e MARIADB_ADE_PASSWORD=passw0rd sinenomine/ade-s390x`

Running Containers

- Containers run as daemons or interactively
- Multiple containers wanting to use same port?
  - Docker can remap:
    -p <host port>:<container port>

Running Containers

- What about environment variables?
  - -e option
  - Dockerfile
- What is my container doing:
  - docker top <*image id*>
  - top
- What is my container config?
- docker inspect <*image*>

```
 "Networks": {
        "bridge": {
            "EndpointID":
"0448a9a68ed5a9c5f89435b3d62d78bbc42d4f601a25e65b2e149ed8f694993c",
            "Gateway": "172.18.0.1",
            "IPAddress": "172.18.0.3",
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "MacAddress": "02:42:ac:12:00:03"
        }
```

Running Containers

- Command line
  - docker run
  - kubernetes
- GUIs
  - Compose-UI
  - AMHub
  - **OpenShift**
- Images are automatically downloaded

Running Containers

- Build on ClefOS / Run on Ubuntu
- Build on ClefOS / Build upon image on Ubuntu
- Builders meet all pre-requisites
- Self-contain requirements
  - No conflicts with other containers
  - Unlike multi-tenancy apps

Openshift Origin - Introduction

Next slides are derived from https://docs.openshift.org/latest/architecture

Things You Need to Know

- OpenShift is a layer on top of Kubernetes
  - OS v3.*xx* based on K8 v1.*xx*
- K8 is a manager of containerized apps across a set of containers and/or hosts
- Concepts of master node, infrastructure node, and compute node
- Provides registry, router, users, groups, projects, builds, templates
- Installs via ansible playbooks which takes care of a lot of the minutiae

What is OpenShift?

- OpenShift is a layer on top of:
  - Docker provides the abstraction for Linux-based lightweight container images
  - Kubernetes provides cluster management & orchestrates containers on multiple hosts
- OpenShift Origin is the Community Edition

Kubernetes

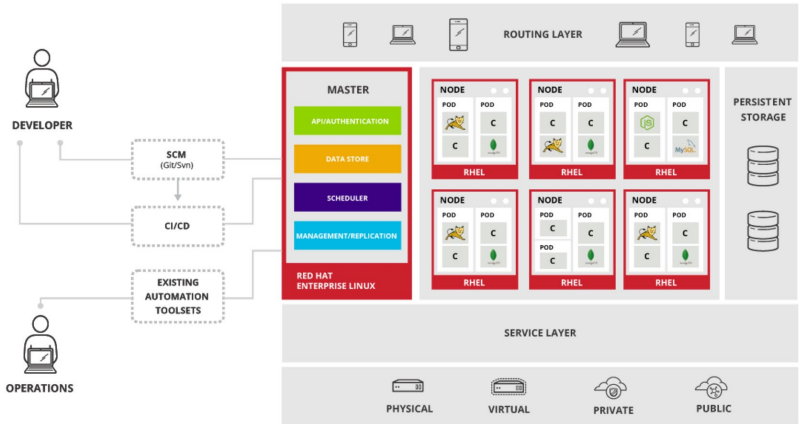
- Manages containerized applications across a set of containers or hosts
- Provides mechanisms for deployment, maintenance, and application-scaling

Kubernetes Components

Component	Description
API Server	Validates and configures the data for pods, services, and replication controllers. It also assigns pods to nodes and synchronizes pod information with service configuration
etcd	Stores the persistent master state while other components watch etcd for changes to bring themselves into the desired state
Controller Manager	Watches etcd for changes to replication controller objects and then uses the API to enforce the desired state
HAProxy	Option to balance load between API master endpoints.

What is OpenShift

- OpenShift Origin adds:
  - Source code management, builds, and deployments for developers
  - Managing and promoting images at scale as they flow through your system
  - Application management at scale
  - Team and user tracking for organizing a large developer organization
  - Networking infrastructure that supports the cluster


DEVELOPER
SCM
(Git/Svn)
CI/CD
EXISTING
AUTOMATION
TOOLSETS
OPERATIONS
ROUTING LAYER
MASTER
API/AUTHENTICATION
DATA STORE
SCHEDULER
MANAGEMENT/REPLICATION
RED HAT
ENTERPRISE LINUX
NODE
POD
C
RHEL
PERSISTENT
STORAGE
SERVICE LAYER
PHYSICAL
VIRTUAL
PRIVATE
PUBLIC

OpenShift Core

- Containers & images are the building blocks for deploying applications
- Pods & services allow for containers to communicate with each other and proxy connections
- Projects and users provide the space and means for communities to organize and manage their content together

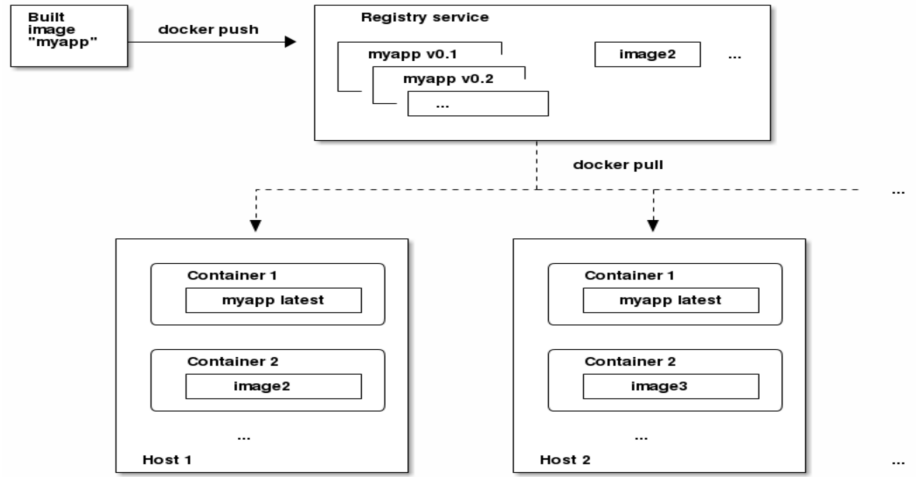
OpenShift Core

- Builds & image streams allow you to build working images and react to new images
- Deployments add expanded support for the development and deployment lifecycle
- Routes announce your service to the world
- Templates allow for many objects to be created at once based on customized parameters.

OpenShift Registries

- A service for storing and retrieving Docker-formatted container images
- A registry contains a collection of one or more image repositories
- Each image repository contains one or more tagged images

OpenShift Registries



OpenShift Pods

- One or more containers deployed together on one host
- The smallest compute unit that can be defined, deployed, and managed.
- Rough equivalent of a machine instance to a container
- Each pod is allocated its own internal IP address

OpenShift Pods

- Containers within pods can share their local storage and networking.
- Pods have a lifecycle: they are defined, assigned to run on a node, then run until their container(s) exit or are Largely immutable: changes cannot be made to a pod definition while it is running
- Changes result in termination & recreation

OpenShift Users

- Interaction with OpenShift Origin is associated with a user
- Users may be placed into groups
- A user object represents an actor which may be granted permissions in the system by adding roles to them or to their groups:
  - Regular Users
  - System Users
  - Service Accounts

OpenShift Projects

- A Kubernetes namespace with additional annotations,
- The central vehicle by which access to resources for regular users is managed
- Allows a community of users to organize and manage their content in isolation from other communities.
- Users must be given access to projects by administrators

OpenShift Builds

- A build is the process of transforming input parameters into a resulting object.
- Transform input parameters or source code into a runnable image
- A BuildConfig object is the definition of the entire build process.
- OpenShift Origin creates Docker-formatted containers from build images and pushing them to a container registry

OpenShift Persistent Storage

- Pods may run on any node
  - Local storage insufficient
- NFS, AWS, iSCSI, GlusterFS, CephFS, or SCSI (zFCP) [and more]
- Kubernetes < 1.8 (OpenShift < 3.8) has limitation for SCSI

OpenShift Replication and Jobs

- Replication Controllers
  - Ensures that a specified number of replicas of a pod are running at all times
- Jobs
  - Similar to replications but designed for one-time pods

OpenShift Deployments

- Provides the ability to transition from an existing deployment of an image to a new one
- Defines hooks to be run before or after creating the replication controller
- When triggered a deployer-pod manages the deployment including scaling down the old replication controller, scaling up the new one, and running hooks
- Triggers may include such things such as a new image becoming available

OpenShift Templates

- Describe a set of objects that can be parameterized and processed to produce a list of objects for creation by OpenShift Origin
- The objects to create can include anything that users have permission to create within a project, e.g. services, build configurations, and deployment configurations.
- May also define a set of labels to apply to every object defined in the template

OpenShift Origin – Demonstration on Z

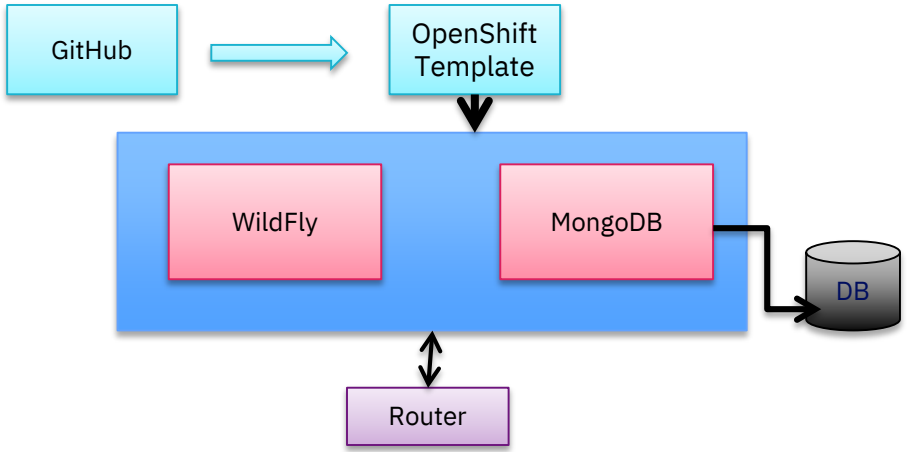
Things You Need to Know

- Demo system is an “all-in-one”
  - 1 virtual machine running
    - Master
    - Infrastructure
    - Compute
- Live demos are unpredictable
- Point your browsers to https://okcd-master.sinenomine.net:8443
  - sna/test or admin/sna

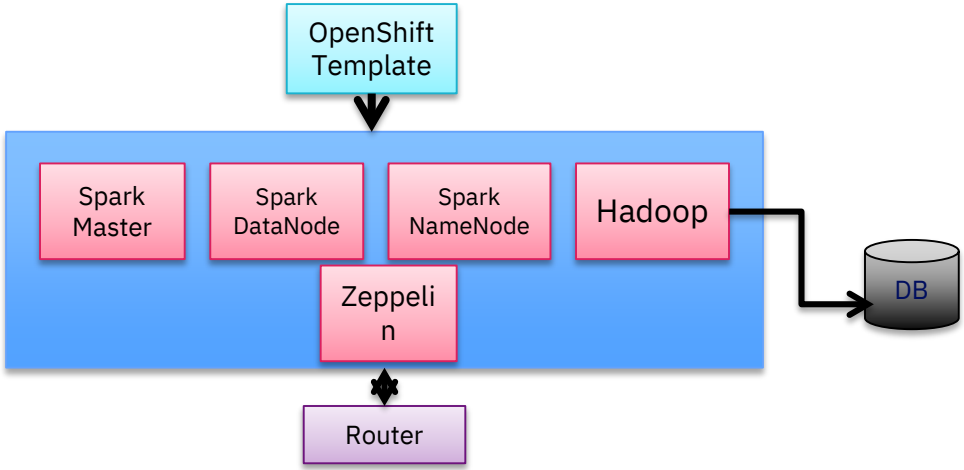
Demo Time

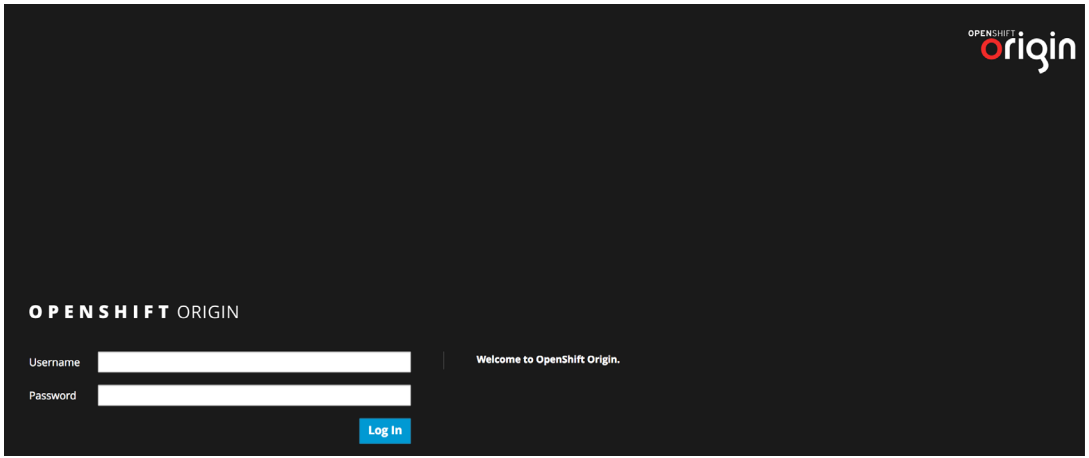
- Using the GUI
- Using the CLI
- Examining the registry
- Simple on pod application
- Source-to-Image application (MLB)
  - JBOSS (Wildfly) & MongoDB
- Orchestration of multiple pods
  - Spark, Hadoop, & Zeppelin

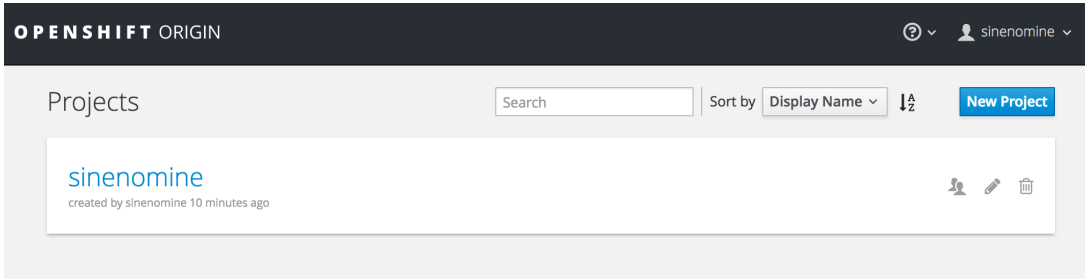
MLB

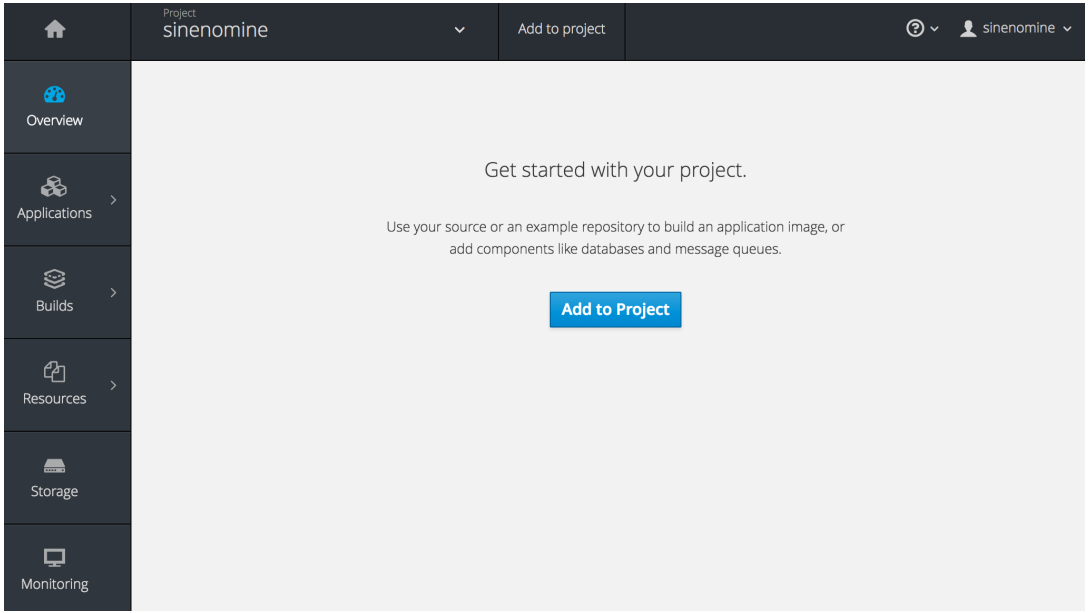


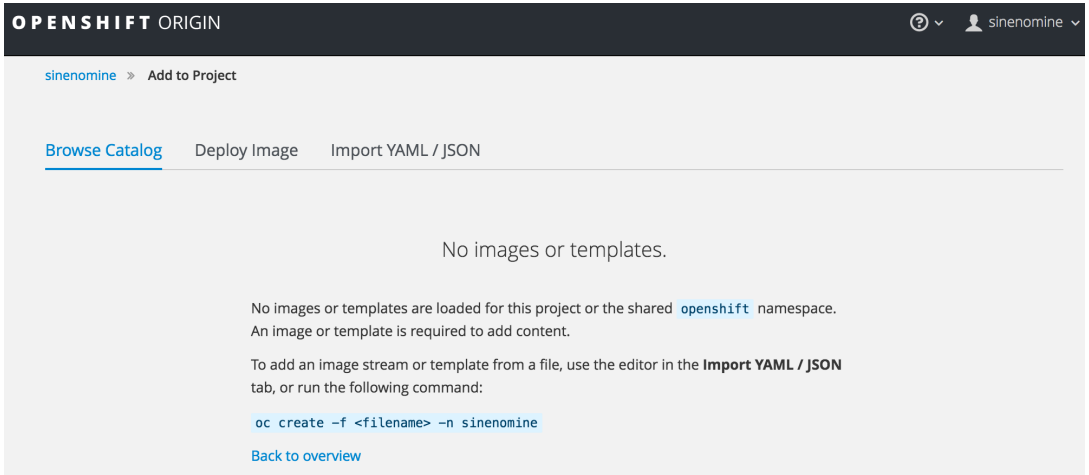
Spark

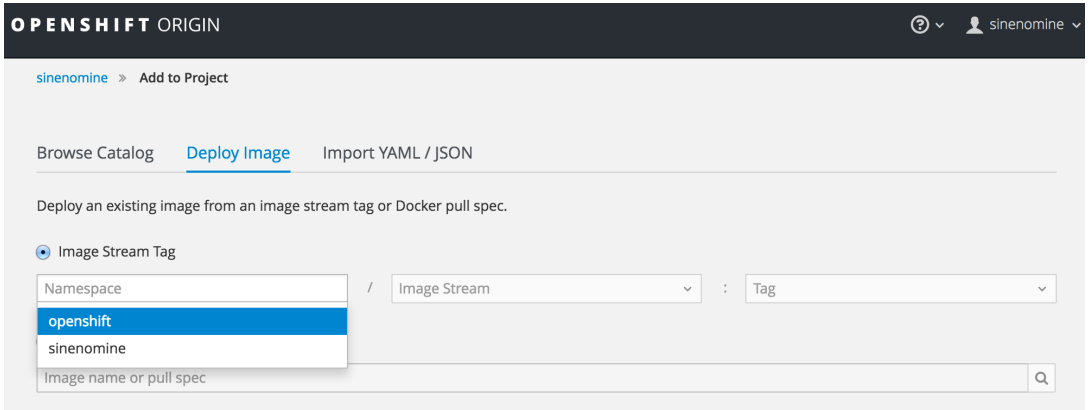

OpenShift Template
Spark Master
Spark DataNode
Spark NameNode
Hadoop
Zeppeli n
DB
Router

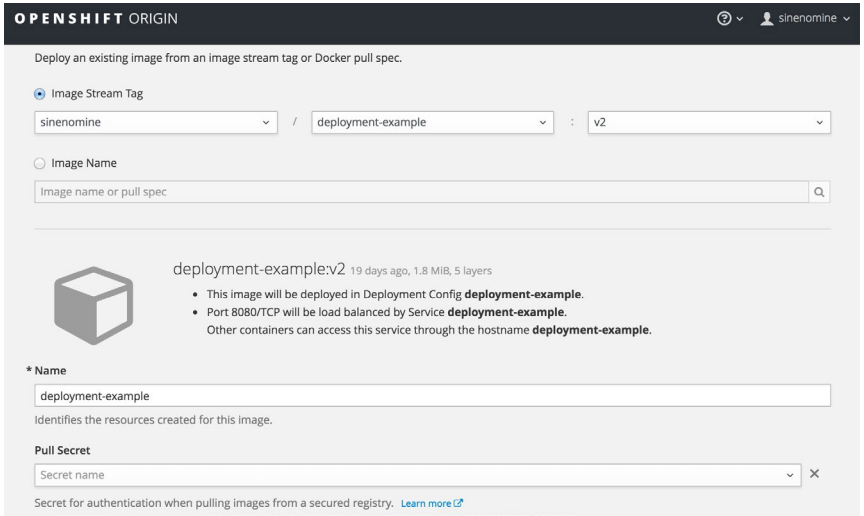

OPENSHIFT origin
OPENSHIFT ORIGIN
Username
Password
Log In
Welcome to OpenShift Origin.

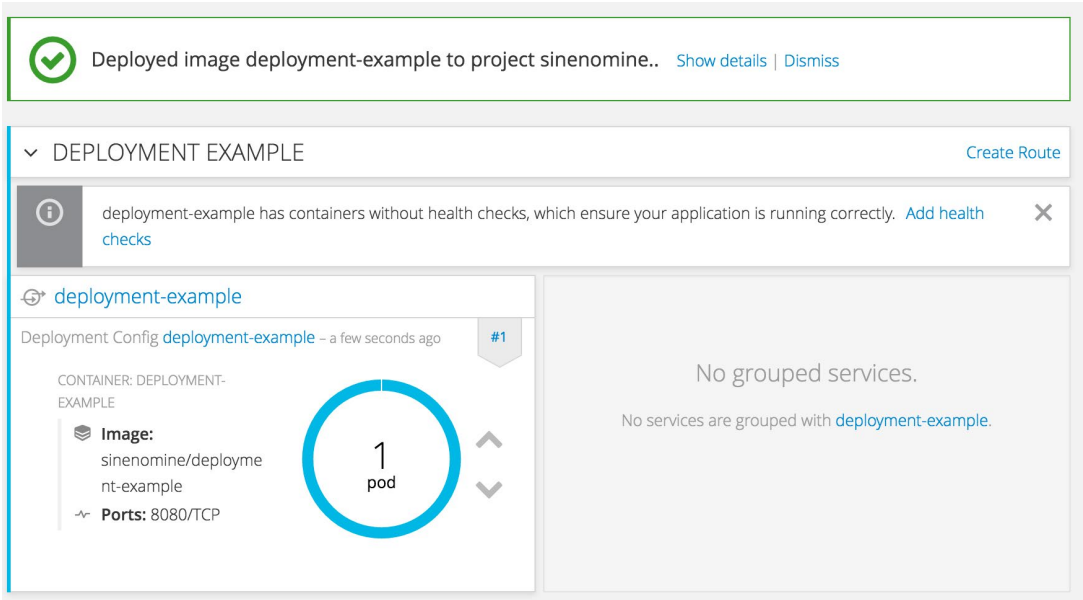

OPENSHIFT ORIGIN
sinenomine
Projects
Search
Sort by
Display Name
New Project
sinenomine
created by sinenomine 10 minutes ago

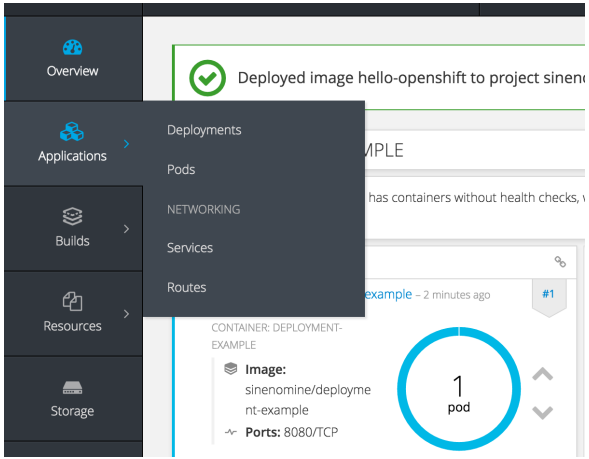

Project
sinenomine
Add to project
sinenomine
Overview
Applications
Builds
Resources
Storage
Monitoring
Get started with your project.
Use your source or an example repository to build an application image, or
add components like databases and message queues.
Add to Project

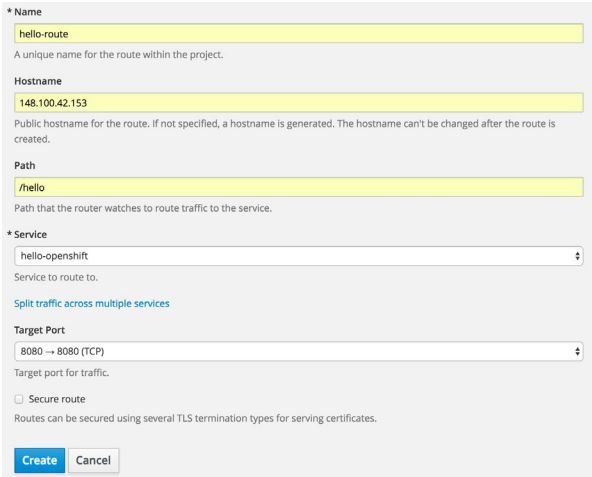

OPENSHIFT ORIGIN
sinenomine
sinenomine » Add to Project
Browse Catalog
Deploy Image
Import YAML / JSON
No images or templates.
No images or templates are loaded for this project or the shared openshift namespace.
An image or template is required to add content.
To add an image stream or template from a file, use the editor in the Import YAML / JSON tab, or run the following command:
oc create -f <filename> -n sinenomine
Back to overview

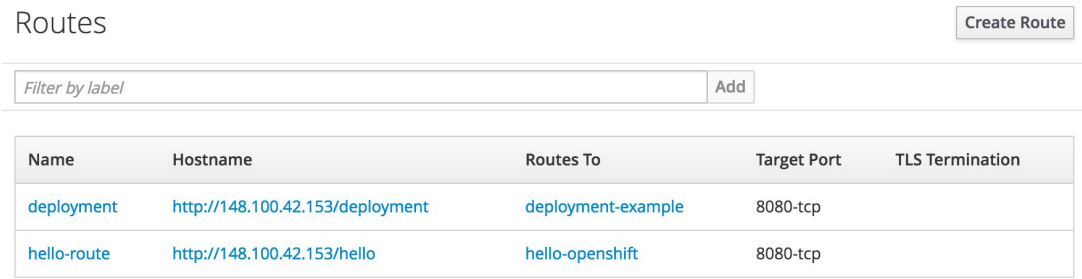

OPENSHIFT ORIGIN
sinenomine
sinenomine » Add to Project
Browse Catalog
Deploy Image
Import YAML / JSON
Deploy an existing image from an image stream tag or Docker pull spec.
Image Stream Tag
Namespace
/
Image Stream
:
Tag
openshift
sinenomine
Image name or pull spec


OPENSHIFT ORIGIN
sinenomine
Deploy an existing image from an image stream tag or Docker pull spec.
Image Stream Tag
sinenomine
/
deployment-example
:
v2
Image Name
Image name or pull spec
deployment-example:v2 19 days ago, 1.8 MiB, 5 layers
This image will be deployed in Deployment Config deployment-example.
Port 8080/TCP will be load balanced by Service deployment-example.
Other containers can access this service through the hostname deployment-example.
* Name
deployment-example
Identifies the resources created for this image.
Pull Secret
Secret name
Secret for authentication when pulling images from a secured registry. Learn more

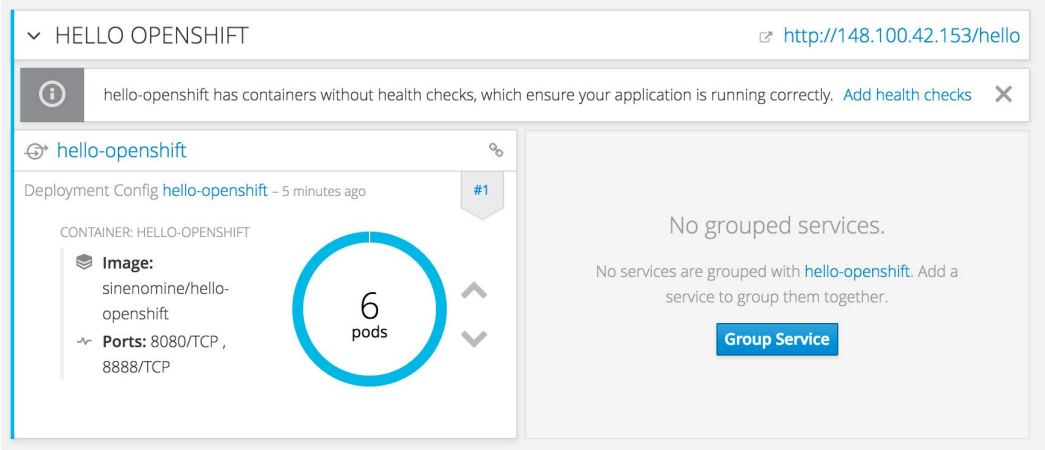

Deployed image deployment-example to project sinenomine..
Show details | Dismiss
DEPLOYMENT EXAMPLE
Create Route
deployment-example has containers without health checks, which ensure your application is running correctly. Add health checks
deployment-example
Deployment Config deployment-example – a few seconds ago
#1
CONTAINER: DEPLOYMENT-EXAMPLE
Image: sinenomine/deployment-example
Ports: 8080/TCP
1
pod
No grouped services.
No services are grouped with deployment-example.


Overview
Applications
Builds
Resources
Storage
Deployments
Pods
NETWORKING
Services
Routes
Deployed image hello-openshift to project sinen
MPLE
has containers without health checks,
example – 2 minutes ago
#1
CONTAINER: DEPLOYMENT-EXAMPLE
Image:
sinenomine/deployment-example
Ports: 8080/TCP
1
pod


* Name
hello-route
A unique name for the route within the project.
Hostname
148.100.42.153
Public hostname for the route. If not specified, a hostname is generated. The hostname can't be changed after the route is created.
Path
/hello
Path that the router watches to route traffic to the service.
* Service
hello-openshift
Service to route to.
Split traffic across multiple services
Target Port
8080 → 8080 (TCP)
Target port for traffic.
Secure route
Routes can be secured using several TLS termination types for serving certificates.
Create
Cancel


Routes
Create Route
Filter by label
Add
Name
Hostname
Routes To
Target Port
TLS Termination
deployment
http://148.100.42.153/deployment
deployment-example
8080-tcp
hello-route
http://148.100.42.153/hello
hello-openshift
8080-tcp


148.100.42.153/deployment
Apps
Bookmarks
ABC News
YouTube
Wikipedia
Google Maps
News
Popular
RT Parent Tickets
v2


HELLO OPENSHIFT
http://148.100.42.153/hello
hello-openshift has containers without health checks, which ensure your application is running correctly. Add health checks
hello-openshift
Deployment Config hello-openshift – 5 minutes ago
#1
CONTAINER: HELLO-OPENSHIFT
Image: sinenomine/hello-openshift
Ports: 8080/TCP , 8888/TCP
6
pods
No grouped services.
No services are grouped with hello-openshift. Add a service to group them together.
Group Service

```
# docker exec -it origin bash
bash-4.2# oc login
Authentication required for https://148.100.42.153:8443 (openshift)
Username: sinenomine
Password:
Login successful.

You have one project on this server: "sinenomine"

Using project "sinenomine".
bash-4.2# oc status
In project sinenomine on server https://148.100.42.153:8443

http://148.100.42.153 to pod port 8080-tcp (svc/deployment-example)
  dc/deployment-example deploys istag/deployment-example:v2
    deployment #1 deployed 8 minutes ago - 1 pod

http://148.100.42.153 to pod port 8080-tcp (svc/hello-openshift)
  dc/hello-openshift deploys istag/hello-openshift:latest
    deployment #1 deployed 6 minutes ago - 6 pods
```

```
# oc get pods
NAME                          READY     STATUS    RESTARTS   AGE
deployment-example-1-ypclg    1/1       Running   0          32m
hello-openshift-1-b61vl       1/1       Running   0          25m
hello-openshift-1-jqkvb       1/1       Running   0          25m
hello-openshift-1-tu25u       1/1       Running   0          25m
hello-openshift-1-v7ra3       1/1       Running   0          30m
hello-openshift-1-vxss9       1/1       Running   0          25m
hello-openshift-1-yw0z7       1/1       Running   0          25m
```

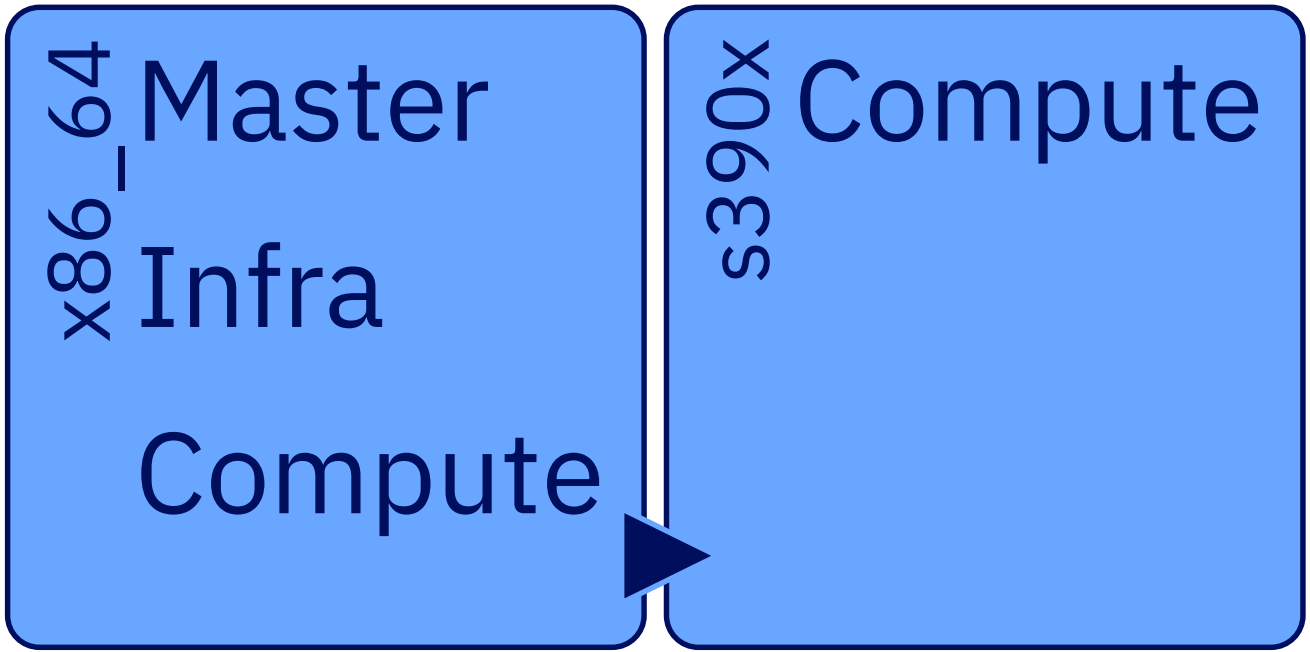
```
# oc new-app docker.io/sinenomine/lighttpd-s390x:latest
--> Found Docker image 3d4758d (3 weeks old) from docker.io for "docker.io/sinenomine/lighttpd-
s390x:latest"

    * An image stream will be created as "lighttpd-s390x:latest" that will track this image
    * This image will be deployed in deployment config "lighttpd-s390x"
    * Port 8091/tcp will be load balanced by service "lighttpd-s390x"
      * Other containers can access this service through the hostname "lighttpd-s390x"
    * WARNING: Image "docker.io/sinenomine/lighttpd-s390x:latest" runs as the 'root' user which
      may not be permitted by your cluster administrator

--> Creating resources ...
    imagestream "lighttpd-s390x" created
    deploymentconfig "lighttpd-s390x" created
    service "lighttpd-s390x" created
--> Success

svc/lighttpd-s390x - 172.30.232.241:8091
  dc/lighttpd-s390x deploys istag/lighttpd-s390x:latest
    deployment #1 deployed 50 seconds ago - 1 pod
```

Openshift Origin – Mixed Platform Operation

Things You Need to Know

- Demo system consists of:
  - 1 x86_64 virtual machine running
    - Master/Infrastructure/Compute
    - Running in SNA DMZ
  - 1 s390x virtual machine running
    - Compute
    - Running on LinuxONE Community Cloud
- Point your browsers to https://oso-dev-test.svc.sinenomine.net:8443
  - sna/test or admin/sna

Mixed Platform Operation



Mixed Platform Operations

```
# oc label node okcd-node.sinenomine.net arch=s390x

 spec:
    replicas: 1
    selector:
      deploymentConfig: ${APPLICATION_NAME}-master
    template:
          :
      spec:
        containers:
            image: docker.io/clefos/spark:2.1.0
          imagePullPolicy: IfNotPresent
          name: ${APPLICATION_NAME}-master
          nodeSelector:
          arch: s390x
```

Mixed Platform Operation

```
# oc describe node
Name:           oso-dev-test.svc.sinenomine.net
Roles:          compute,infra,master
Labels:         beta.kubernetes.io/arch=amd64
                beta.kubernetes.io/os=linux
                kubernetes.io/hostname=oso-dev-test.svc.sinenomine.net
                logging-infra-fluentd=true
                node-role.kubernetes.io/compute=true
                node-role.kubernetes.io/infra=true
                node-role.kubernetes.io/master=true
```

Mixed Platform Operation

```
Namespace                       Name
---------                       ----
default                         docker-registry-1-dqlrd
default                         registry-console-1-rmzf5
default                         router-1-95crd
kube-service-catalog            apiserver-n5s9v
kube-service-catalog            controller-manager-npwp9
kube-system                     master-api-oso-dev-test.svc.sinenomine.net
kube-system                     master-controllers-oso-dev-test.svc.sinenomine.net
kube-system                     master-etcd-oso-dev-test.svc.sinenomine.net
openshift-ansible-service-broker asb-1-rl6tr
openshift-logging               logging-curator-1-vnjlg
openshift-logging               logging-fluentd-c2jfj
openshift-node                  sync-8n4kb
openshift-sdn                   ovs-5kdkc
openshift-sdn                   sdn-rnmnh
openshift-template-service-broker apiserver-rq9j2
openshift-web-console           webconsole-57d88df7d9-wgcpw
```

Mixed Platform Operation

```
# oc describe node
Name:           okcd-node.sinenomine.net
Roles:          compute
Labels:         arch=s390x
                beta.kubernetes.io/arch=s390x
                beta.kubernetes.io/os=linux
                kubernetes.io/hostname=okcd-node.sinenomine.net
                logging-infra-fluentd=true
                node-role.kubernetes.io/compute=true
Annotations:    node.openshift.io/md5sum=32ae361b122c8a26a133736689eaf26e
                volumes.kubernetes.io/controller-managed-attach-detach=true
CreationTimestamp:  Tue, 30 Oct 2018 12:16:40 -0400
```

Mixed Platform Operation

```
Namespace                Name
---------                ----
openshift-logging        logging-fluentd-2fnt5
openshift-node           sync-ln6t4
openshift-sdn            ovs-txlsb
openshift-sdn            sdn-gq897
sinenomine               spark-datanode-1-rzpwf
sinenomine               spark-master-1-8svnp
sinenomine               spark-namenode-1-27jsj
sinenomine               spark-ui-proxy-1-99b7x
sinenomine               spark-worker-1-pzv98
sinenomine               spark-worker-1-vdwmn
sinenomine               spark-zeppelin-1-ktwqj
```