OpenShift

At a high-level, OpenShift is Red Hat's 'platform as a service' offering. It is essentially Kubernete's with some additional features on top, such as Jenkins clusters for building, the facility to trigger builds from source changes etc.

Flavours

Confusing no?

- OpenShift Origin: Open source, community and RH maintained. Can be installed for free.
- OpenShift Enterprise: A few versions behind, bullet proofed by RH. Expensive.
- OpenShift Cloud: Enterpise, on the cloud, subscription based.
- OpenShift Container Platform: What OpenShift Enterprise has been called since version 3.3. I think, but who the hell knows.

Getting Started

1. Read the Architecture Overview: https://docs.openshift.org/latest/architecture/index.html#architecture-index
2. If you don't know Kubernetes, learn Kubernetes first (at least the basics)
3. Actually set up OpenShift on some VMs. Two choices here, try Installing OpenShift Origin with Ansible on AWSOR try installing OpenShift Enterprise on VMs. There's a good Udemy course on Installing and Configuring OpenShift Enterprise
4. Read the basics of the OpenShift CLI

Quick Reference

Show all nodes and labels:

```
oc get nodes --show-labels
```

Identity Providers

Check out the documentation for Configuring Authentication and User Agent.

An example htpasswd provider:

```
 - challenge: true
   login: true
   mappingMethod: claim
   name: htpasswd_auth
   provider:
     apiVersion: v1
     file: /etc/origin/openshift-passwd
     kind: HTPasswdPasswordIdentityProvider
```

An example ldap provider:

```
identityProviders:
 - name: "my_ldap_provider"
   challenge: true
   login: true
   mappingMethod: claim
   provider:
     apiVersion: v1
     kind: LDAPPasswordIdentityProvider
     attributes:
       id:
       - dn
       email:
       - mail
       name:
       - cn
       preferredUsername:
       - uid
     bindDN: "cn=root,cn=Users,dc=directory"
     bindPassword: "password"
```

```
    ca: my-ldap-ca-bundle.crt
    insecure: false
    url: "ldap://10.0.3.133:389/cn=users,dc=directory,dc=openshift?sAMAccountName"
```

Diagnosing Issues

Some assorted tips and tricks.

Metrics Failures, Hawkular Failure, Etc

Some common issues you might find after trying to set up metrics, e.g. by following the 'Enabling Cluster Metrics' guide:

- Could not connect to metrics service. checking the metrics tabs
- hawkular-metrics.<host> unexpectedly closed the connection. when trying to open the UI

You could try the following.

Allow an unsigned certificate in the browser

Open up the hawkular-metrics.<host>/hawkular/metrics page, if you have a self signed certificate you may need to accept it in your browser, then try again.
Check the status of the services

Log into the console, then start to diagnose. Check the projects:

```
$ oc get projects
NAME              DISPLAY NAME   STATUS
management-infra                 Active
openshift                        Active
openshift-infra                  Active
default                          Active
```

Move to the openshift-infra project, which should contain the metrics:

```
$ oc project openshift-infra
Now using project "openshift-infra" on server "<server>".
```

Check the status:

```
$ oc status
In project openshift-infra on server XXX

svc/hawkular-cassandra - 172.30.156.179 ports 9042->cql-port, 9160->thift-port, 7000->tcp-port, 7001->ssl-port

svc/hawkular-cassandra-nodes (headless) ports 9042, 9160, 7000, 7001

svc/hawkular-metrics - 172.30.146.174:443 -> https-endpoint
  rc/hawkular-metrics runs registry.access.redhat.com/openshift3/metrics-hawkular-metrics:3.1.1
    rc/hawkular-metrics created 17 hours ago - 1 pod
  exposed by route/hawkular-metrics

svc/heapster - 172.30.2.114:80 -> http-endpoint
  rc/heapster runs registry.access.redhat.com/openshift3/metrics-heapster:3.1.1
    rc/heapster created 17 hours ago - 1 pod

4 warnings identified, use 'oc status -v' to see details.
```

Note that in this example, the status output is showing there are warnings. Take a deeper look:

```
$ oc status -v
...
Warnings:
  * rc/hawkular-metrics is attempting to mount a secret secret/hawkular-metrics-secrets disallowed by sa/hawkular
  * pod/hawkular-metrics-s8euf is attempting to mount a secret secret/hawkular-metrics-secrets disallowed by
sa/hawkular
  * container "heapster" in pod/heapster-dm3lv has restarted 206 times
  * pod/metrics-deployer-00got is attempting to mount a secret secret/metrics-deployer disallowed by sa/metrics-
deployer
```

View details with 'oc describe <resource>/<name>' or list everything with 'oc get all'.

The 'secret' issues are not normally a problem, you'll often see them if you use a 'null' secret (e.g. oc secrets new metrics-deployer nothing=/dev/null). However, the restarted 206 times is definitely an issue. The heapstercontainer in the pod is likely failing immediately, this is probably the root cause. Take a look into the pod:

```
$ oc describe pod/heapster-dm3lv
Name:                   heapster-dm3lv
Namespace:              openshift-infra
Image(s):               registry.access.redhat.com/openshift3/metrics-heapster:3.1.1
Node:                   XXX
Start Time:             Fri, 27 Jan 2017 10:50:21 -0500
Labels:                 metrics-infra=heapster,name=heapster
Status:                 Running
Reason:
Message:
IP:                     10.1.2.13
Replication Controllers:    heapster (1/1 replicas created)
Containers:
  heapster:
    Container ID:   docker://04b7b0cd0011afa1688212e50083b63bb69ac3efcd067c53f4d336913dce6639
    Image:          registry.access.redhat.com/openshift3/metrics-heapster:3.1.1
    Image ID:       docker://0e218c9b31afa39b64e8f4e2c32f2aa10656c281a267279147bc081d6227ac45
    QoS Tier:
      cpu:              BestEffort
      memory:           BestEffort
    State:              Waiting
      Reason:           CrashLoopBackOff
    Last Termination State:     Terminated
      Reason:           Error
      Exit Code:        255
      Started:          Sat, 28 Jan 2017 04:10:06 -0500
      Finished:         Sat, 28 Jan 2017 04:10:09 -0500
    Ready:              False
    Restart Count:      207
```

```
    Environment Variables:
Conditions:
  Type          Status
  Ready         False
Volumes:
  heapster-secrets:
    Type:       Secret (a secret that should populate this volume)
    SecretName: heapster-secrets
  hawkular-metrics-certificate:
    Type:       Secret (a secret that should populate this volume)
    SecretName: hawkular-metrics-certificate
  hawkular-metrics-account:
    Type:       Secret (a secret that should populate this volume)
    SecretName: hawkular-metrics-account
  heapster-token-2mn3t:
    Type:       Secret (a secret that should populate this volume)
    SecretName: heapster-token-2mn3t
Events:
  FirstSeen     LastSeen        Count   From                                                    SubobjectPath               Reason  Message
  _________     ________        _____   ____                                                    _____________               ______  _______

  ...
  17h           2m              206     {kubelet ip-10-0-1-231.ap-southeast-1.compute.internal} spec.containers{heapster}
Pulled  Container image "registry.access.redhat.com/openshift3/metrics-heapster:3.1.1" already prese
nt on machine
  2m            2m              1       {kubelet ip-10-0-1-231.ap-southeast-1.compute.internal} spec.containers{heapster}
Created Created with docker id 04b7b0cd0011
  2m            2m              1       {kubelet ip-10-0-1-231.ap-southeast-1.compute.internal} spec.containers{heapster}
Started Started with docker id 04b7b0cd0011
  17h           8s              6046    {kubelet ip-10-0-1-231.ap-southeast-1.compute.internal} spec.containers{heapster}
Backoff Back-off restarting failed docker container
```

Notice we have a container in the Waiting state with the Reason: CrashLoopBackOff. The system is smart enough to not waste too many resources on a continously crashing container. A quick look at the recent events shows something similar:

```
$ oc get events
FIRSTSEEN   LASTSEEN   COUNT     NAME            KIND      SUBOBJECT                 REASON    SOURCE
MESSAGE
...
9m          9m         1         heapster-dm3lv   Pod       spec.containers{heapster}   Created   {kubelet ip-10-0-1-231.XXX}
Created with docker id 31a08da9b835
9m          9m         1         heapster-dm3lv   Pod       spec.containers{heapster}   Started   {kubelet ip-10-0-1-231.XXX}
Started with docker id 31a08da9b835
4m          4m         1         heapster-dm3lv   Pod       spec.containers{heapster}   Created   {kubelet ip-10-0-1-231.XXX}
Created with docker id 60f2b422bf39
4m          4m         1         heapster-dm3lv   Pod       spec.containers{heapster}   Started   {kubelet ip-10-0-1-231.XXX}
Started with docker id 60f2b422bf39
```

The heapster container is getting created again and again.

So we can see the pod has issues - now we need to see the actual docker logs for the pod and try and work out what was the failure. Trying to check the pod logs is a no-go:

```
$ oc logs heapster-dm3lv
Error from server: Internal error occurred: Pod "heapster-dm3lv" in namespace "openshift-infra": container "heapster" is in waiting state.
```

The pod is waiting, we need to see the previous logs. Fortunately, we have a flag for that. The -p flag for logs says: -p, --previous=false: If true, print the logs for the previous instance of the container in a pod if it exists.. So we can see what went wrong:

```
$ oc logs -p heapster-dm3lv
Starting Heapster with the following arguments: --
source=kubernetes:https://kubernetes.default.svc:443?useServiceAccount=true&kubeletHttps=true&kubeletPort=10250
--sink=hawkular:https://hawkular-metrics:443?tenant=_system&labelToTenant=pod_namespace&caCert=/hawkular-
cert/hawkular-metrics-ca.certificate&user=hawkular&pass=17m-
SlKEDZEV7yK&filter=label(container_name:^/system.slice.*|^/user.slice) --logtostderr=true --
tls_cert=/secrets/heapster.cert --tls_key=/secrets/heapster.key --tls_client_ca=/secrets/heapster.client-ca --
allowed_users=system:master-proxy
I0128 04:25:36.068928       1 heapster.go:60] heapster --
source=kubernetes:https://kubernetes.default.svc:443?useServiceAccount=true&kubeletHttps=true&kubeletPort=10250
```

```
--sink=hawkular:https://hawkular-metrics:443?tenant=_system&labelToTenant=pod_namespace&caCert=/hawkular-cert/hawkular-metrics-ca.certificate&user=hawkular&pass=17m-SlKEDZEV7yK&filter=label(container_name:^/system.slice.*|^/user.slice) --logtostderr=true --tls_cert=/secrets/heapster.cert --tls_key=/secrets/heapster.key --tls_client_ca=/secrets/heapster.client-ca --allowed_users=system:master-proxy
I0128 04:25:36.071466       1 heapster.go:61] Heapster version 0.18.0
I0128 04:25:36.071937       1 kube_factory.go:168] Using Kubernetes client with master "https://kubernetes.default.svc:443" and version "v1"
I0128 04:25:36.071947       1 kube_factory.go:169] Using kubelet port 10250
I0128 04:25:36.072242       1 driver.go:491] Initialised Hawkular Sink with parameters {_system https://hawkular-metrics:443?tenant=_system&labelToTenant=pod_namespace&caCert=/hawkular-cert/hawkular-metrics-ca.certificate&user=hawkular&pass=17m-SlKEDZEV7yK&filter=label(container_name:^/system.slice.*|^/user.slice) 0xc20818a5a0 }
F0128 04:25:39.083374       1 heapster.go:67] Get https://hawkular-metrics:443/hawkular/metrics/metrics?type=gauge: dial tcp 172.30.146.174:443: no route to host
```

Now we are getting somewhere. We see: Get https://hawkular-metrics:443/hawkular/metrics/metrics?type=gauge: dial tcp 172.30.146.174:443: no route to host.

Interestingly, we see a request to https://hawkular-metrics:443/, which is not qualified with our internal host name.

For any connectivity issues, when you see that there is SSL going on, think self-signed certificates and the issues they can cause for HTTP clients.

In the setup guides available, we see that the master-config.json is pretty explicit about what domains to allow for CORS:

```
$ vi /etc/origin/master/master-config.yaml

...

corsAllowedOrigins:
  - 127.0.0.1
  - localhost
  - 10.0.1.53
  - 52.76.129.116
  - kubernetes.default
  - ip-10-0-1-53.ap-southeast-1.compute.internal
```

```
  - kubernetes
  - openshift.default
  - openshift.default.svc
  - 172.30.0.1
  - openshift.default.svc.cluster.local
  - kubernetes.default.svc
  - kubernetes.default.svc.cluster.local
  - openshift


...
```

Adding the following may help:

```
corsAllowedOrigins:
  ...
  - hawkular-metrics
```

However after saving, restarting with systemctl restart atomic-openshift-master.service and rechecking the logs shows the same issue. Let's try connecting ourselves. First, we can check what routes we have to the service:

```
$ oc get svc
NAME                  CLUSTER_IP       EXTERNAL_IP   PORT(S)                          SELECTOR                AGE
hawkular-cassandra        172.30.156.179   <none>        9042/TCP,9160/TCP,7000/TCP,7001/TCP   type=hawkular-
cassandra   20h
hawkular-cassandra-nodes   None             <none>        9042/TCP,9160/TCP,7000/TCP,7001/TCP   type=hawkular-
cassandra   20h
hawkular-metrics          172.30.146.174   <none>        443/TCP                          name=hawkular-metrics     20h
heapster                  172.30.2.114     <none>        80/TCP                           name=heapster             20h
```

Indeed, it seems that hawkular-metrics is running on 172.30.146.74. So maybe it is not running correctly? Another look at oc get pods shows a potential problem:

```
$ oc get pods
NAME                  READY     STATUS             RESTARTS   AGE
hawkular-metrics-s8euf   0/1       Pending            0          20h
heapster-dm3lv           0/1       CrashLoopBackOff   249        20h
metrics-deployer-00got   0/1       Error              0          20h
```

The metrics-deployer pod is in an error state. Let's dig deeper:

```
$ oc logs pod/metrics-deployer-00got
...
Error from server: serviceaccounts "hawkular" already exists
```

It looks like the installer failed due to an account already existing. Likely from a failed earlier attempt. Checking the readme for Origin Metrics shows a handy guide for cleanup up:

```
# Remove deployed components.
oc delete all,secrets,sa,templates --selector=metrics-infra -n openshift-infra
# Remove the deployer itself.
oc delete sa,secret metrics-deployer -n openshift-infra
```

You might find that after checking oc get pods some pods are stuck in a Terminating state. If this persits, go nuclear:

```
$ oc delete pods/hawkular-metrics-zgnjx --grace-period=0
```

This time I followed the instructions to setup metrics just as before, except for creating the hawkular service account, as the installer error message suggested that it was trying to create it itself. After a few minutes of watching oc get events --watch things looked healthy. A quick check to confirm:

```
$ oc get pods
NAME                         READY     STATUS      RESTARTS   AGE
hawkular-cassandra-1-r1muo   1/1       Running     0          2m
hawkular-metrics-y21m3       1/1       Running     0          2m
heapster-57qnl               1/1       Running     4          2m
metrics-deployer-ifm5r       0/1       Completed   0          2m
```

And metrics are now working fine.

Resources which were useful:

- https://docs.openshift.com/enterprise/3.1/install_config/cluster_metrics.html
- https://github.com/openshift/origin/issues/6725
- https://github.com/openshift/origin-metrics/issues/123
- https://github.com/openshift/origin-metrics
- https://github.com/openshift/origin/issues/8176

Failed to pull image, "unauthorized: authentication required"

First thing to try - if you are pulling from a private registry on the Docker Hub, make sure you have setup your secrets. If that doesn't work, try changing:

```
myorg/myimage:mytag
```

to:

```
docker.io/myorg/myimage:mytag
```

This took me hours to work out...

Failed to pull image, unsupported schema version 2

Occurs when trying to pull images from a registry which uses the latest schema using an older Docker client. Lot's of wasting time on this one!

tl;dr

Just run this:

```
oc set env dc/docker-registry -n default REGISTRY_MIDDLEWARE_REPOSITORY_OPENSHIFT_ACCEPTSCHEMA2=true
systemctl restart origin-master.service
```

Details:

- Acknowledgement of the issue (OSE 3.2): https://docs.openshift.com/enterprise/3.2/release_notes/ose_3_2_release_notes.html (under 'known issues')
- The fix (OCP 3.3): https://docs.openshift.com/container-platform/3.3/release_notes/ocp_3_3_release_notes.html#ocp-support-docker-distribution-2-4
- More info: https://access.redhat.com/solutions/2391351
- Even more info: https://github.com/openshift-evangelists/vagrant-origin/issues/35

How to configure node selectors for a deployment

To use Node Selectors for deployment configurations, edit the YAML to include a spec.template.spec.nodeSelectorlike so:

```
apiVersion: v1
kind: DeploymentConfig
metadata:
  # etc
spec:
  # etc
  template:
    # etc
    spec:
      containers:
        # etc
      nodeSelector:
        zone: west
# etc
```