Did You Know?

Table of Contents

OOM restarts ONAP components after a failure

OOM uses a Kubernetes controller to constantly monitor all of the deployed containers (pods) in an OOM deployment of ONAP automatically restarting any container that fails. If you'd like to try this out for yourself stop one of the containers by using the following command:

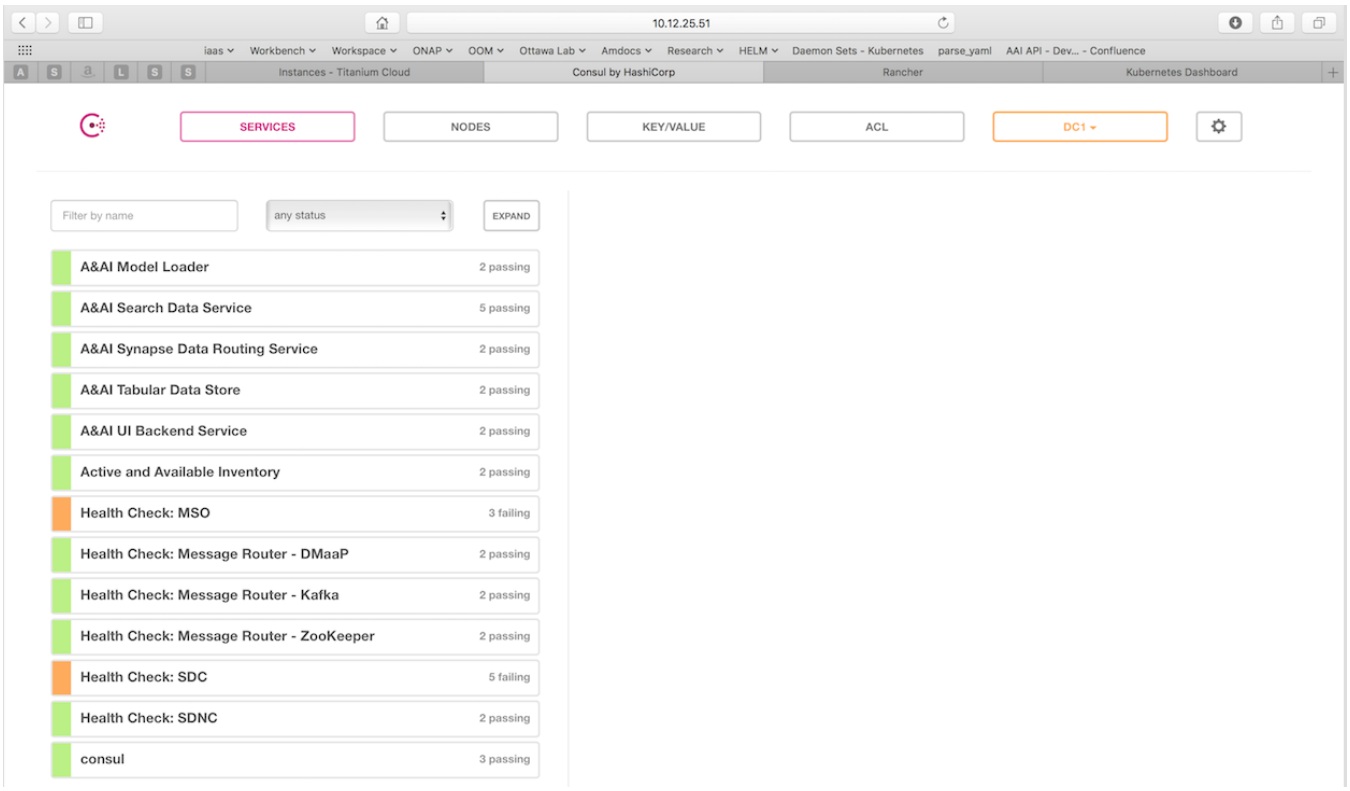
```
kubectl delete pod <pod name> -n <pod namespace>
```

You'll see the pod go down and immediately be restarted.

OOM provides a real-time view of ONAP component health

OOM deploys a Consul cluster that actively monitors the health of all ONAP components in a deployment. To see what the health of your deployment looks like, enter the following url in your browser:

http://<kubernetes IP>:30270/ui/



More information can be found on our wiki at Health Monitoring.

OOM supports deploying subsets of the ONAP components

ONAP has a lot of components! If you would like to focus on a select few components you can do so today by modifying the environment file setenv.bash and changing the list of components.

For example, in Beijing we have:

```
   HELM_APPS=('consul' 'msb' 'mso' 'message-router' 'sdnc' 'vid' 'robot' 'portal' 'policy' 'appc' 'aai' 'sdc'
'dcaegen2' 'log' 'cli' 'multicloud' 'clamp' 'vnfsdk' 'uui' 'aaf' 'vfc' 'kube2msb' 'esr')
```

Simply modify the array to something like this:

```
   HELM_APPS=('mso' 'message-router' 'sdnc' 'vid' 'robot' 'portal' 'policy' 'appc' 'aai' 'sdc')
```

Launch the “oneclick” installer and wait for your pods to startup.

```
   # createAll.bash -n onap
```

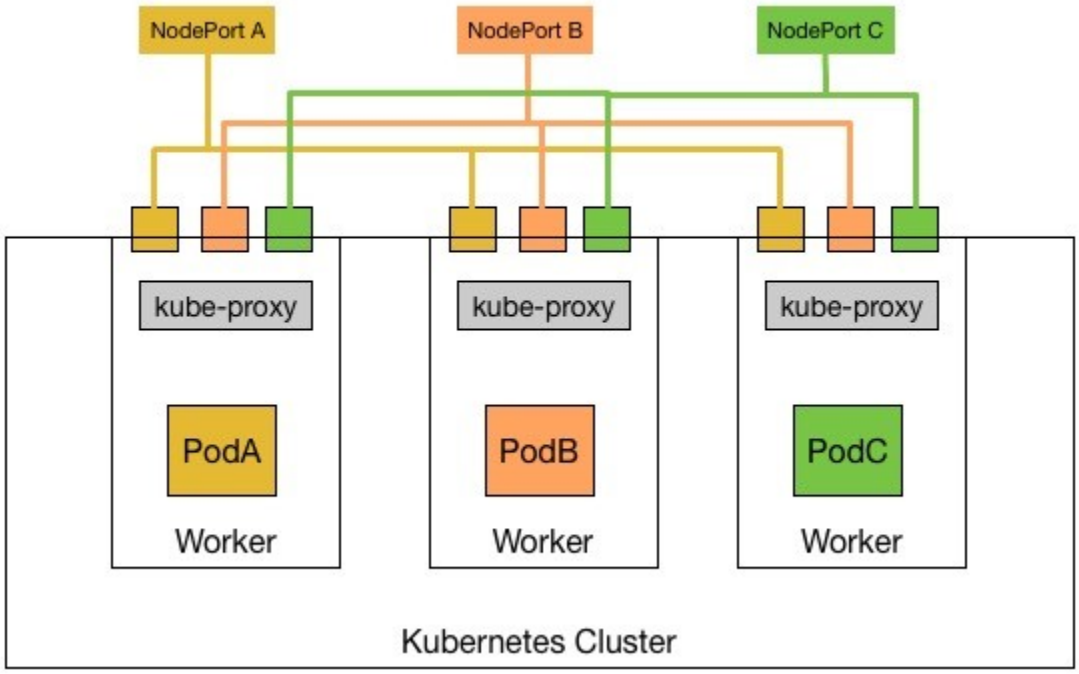
OOM makes services available on any Node in a K8S cluster using a NodePort

This allows a service to be reachable on any node in the cluster, using a statically assigned NodePort (default range: 30000-32767). NodePorts are automatically mapped to the internal port of a service.

Some examples of services available today using NodePorts:

30023 --> 8080 (m)so

All services are mapped here: in OOM wiki.



OOM manages dependencies between ONAP components

OOM uses deployment descriptor files to define what components are deployed in ONAP. Inside the descriptor files, **initContainers** are used to ensure that dependent components are running and are healthy (ie. ready) before a component's container is started.

Example mso deployment descriptor file shows mso's dependency on its mariadb component:

mso descriptor

```
..
kind: Deployment
metadata:
  name: mso
..
spec:
..

      initContainers:
      - command:
        - /root/ready.py
        args:
        - --container-name
        - mariadb
..
      containers:
      - command:
        - /tmp/start-jboss-server.sh
        image: {{ .Values.image.mso }}
        imagePullPolicy: {{ .Values.pullPolicy }}
        name: mso
..
```

OOM supports scaling stateful applications

OOM leverages the kubernetes capability to scale stateful applications. And this is done without the need to change any application code.

sdnc scale example

```
kind: StatefulSet
metadata:
        name: sdnc-dbhost
        namespcae: "{{ .Values.nsPrefix }}-sdnc"
spec:
        serviceName: "dbhost"
        replicas: 2
```

You can try these examples today to scale out and back in the SDNC database.

Scale out sdnc database to 5 instances:

```
   kubectl scale statefulset sdnc-dbhost -n onap-sdnc --replicas=5
```

Scale in sdnc database to 2 instances:

```
   kubectl scale statefulset sdnc-dbhost -n onap-sdnc --replicas=2
```

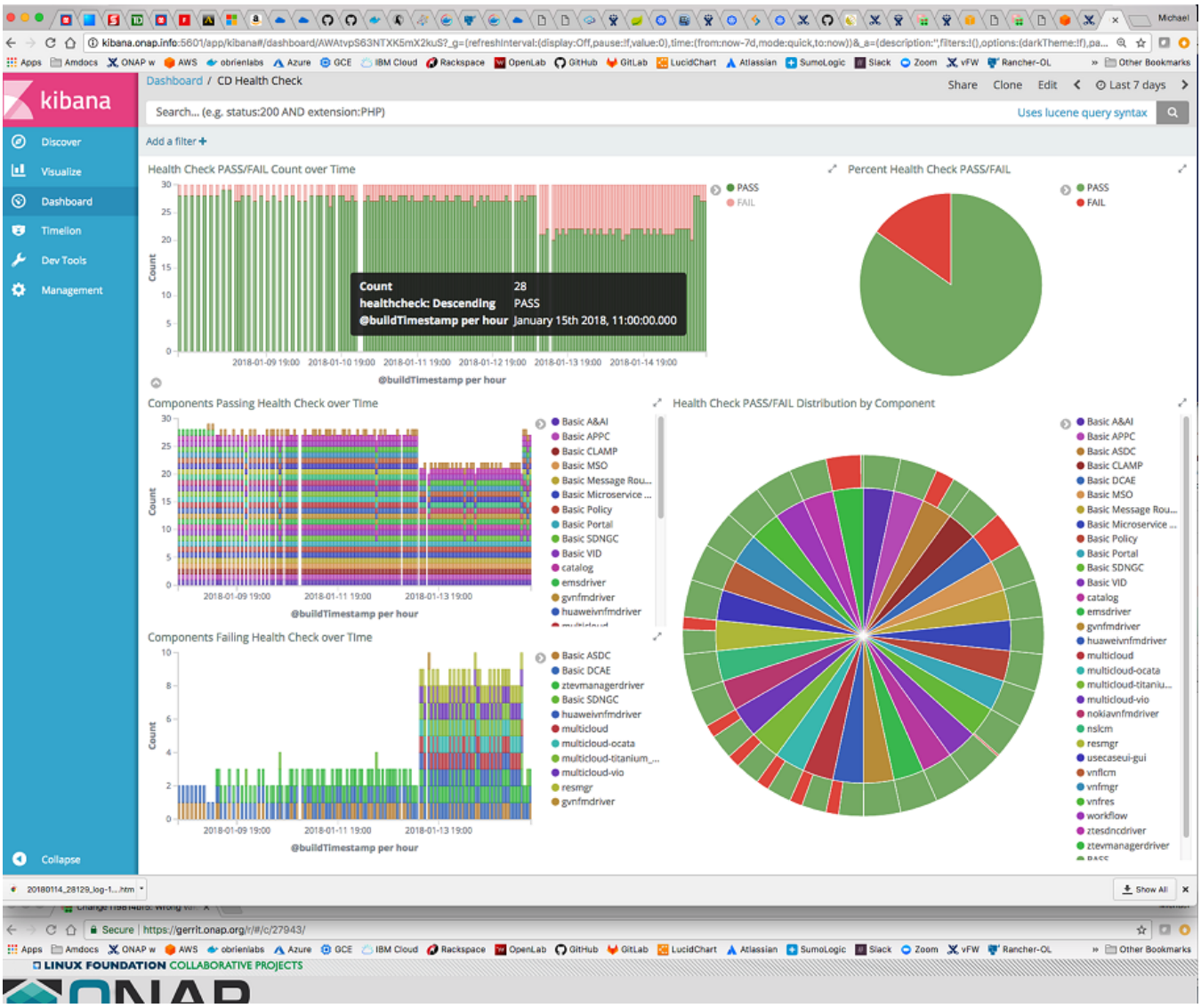
OOM is deployed and validated hourly as a Continuous Deployment of ONAP master on AWS

Currently, merges in ONAP get compiled/validated by the JobBuilder – this is good CI. Additional validation via CD on top of CI tag sets allows for full CI/CD per commit as part of the JobBuilder CI chain.

OOM Kubernetes deployment + Jenkins + ELK: gives us historical healthcheck and live build validation

Examples: The day before the F2F in Dec – the only AAI healthcheck failure in the last 3 months – a chef issue – appeared and was fixed by the AA&I team in 3 hours.

15 Jan 2018 changes to Kube2msb were validated and visible in the last 7 days view below – click on the top-right “last 7 days” to expand search or click the red pie to sort on failures and the inner pie segment in the larger pie to sort by component.



The last 60 min of merges to ONAP component configuration and OOM Kubernetes infrastructure as well as the daily Linux Foundation docker images are tested in each slice in the 7d view above. Results from robot healthcheck, and partial vFW init/distribute are streamed from the VM under test -via Jenkins into the ELK stack – tracking hourly ONAP deployment health.

Jenkins:

- http://jenkins.onap.info/job/oom-cd/buildTimeTrend

Kibana:

- http://kibana.onap.info:5601/app/kibana#/dashboard/AWAtvpS63NTXK5mX2kuS?_g=(refreshInterval:(display:Off,pause:!f,value:0),time:(from:now-7d,mode:quick,to:now))&_a=(description:'',filters:!(),options:(darkTheme:!f),panels:!((col:1,id:AWAts77k3NTXK5mX2kuM,panelIndex:1,row:1,size_x:8,size_y:3,type:visualization),(col:9,id:AWAtuTVI3NTXK5mX2kuP,panelIndex:2,row:1,size_x:4,size_y:3,type:visualization),(col:1,id:AWAtuBTY3NTXK5mX2kuO,panelIndex:3,row:7,size_x:6,size_y:3,type:visualization),(col:1,id:AWAttmqB3NTXK5mX2kuN,panelIndex:4,row:4,size_x:6,size_y:3,type:visualization),(col:7,id:AWAtvHtY3NTXK5mX2kuR,panelIndex:6,row:4,size_x:6,size_y:6,type:visualization)),query:(match_all:()),timeRestore:!f,title:'CD%20Health%20Check',uiState:(),viewMode:view)

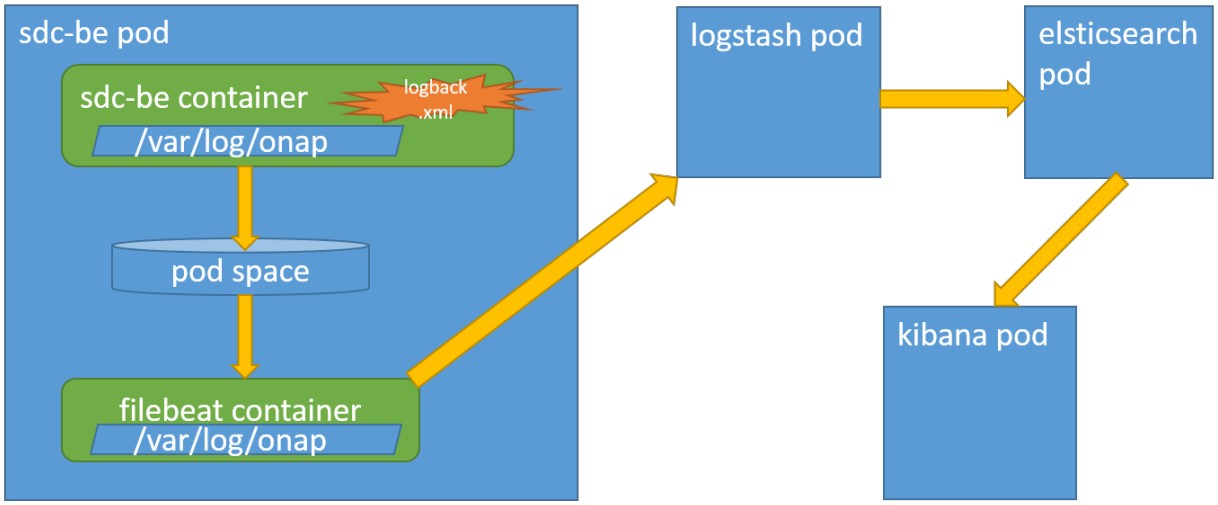
Master: (ONAP pods up at the 22m mark of each hour) – security temporarily off for a couple days

- http://amsterdam.onap.info:8880/env/1a7/infra/hosts/1h1/containers
- http://amsterdam.onap.info:8880/r/projects/1a7/kubernetes-dashboard:9090/#!/pod?namespace=_all

OOM enhances ONAP by deploying Logging framework

OOM implements Logging Enhancement Project by deploying Elastic Stack and defining consistent logging configuration.

OOM deploys (ElasticSearch, Logstash, Kibana) together with filebeat to collect logs from the different ONAP components. It enables components with canonical logs configuration.



You can access ONAP Logs Kibana at:

http://<any_of_your_kubernetes_nodes>:30253/app/kibana

Components that have logs available in OOM Elastic Stack: A&AI, APPC, MSO, Policy, Portal, SDC, SDNC, VID.

If you would like ONAP to collect your components logs in a consistent matter, please contact the OOM team.

OOM automatically registers all components to MSB

ONAP services need to be registered to MSB to leverage its service discovery/routing/LB capabilities.

OOM uses the KUBE2MSB registrator (another ONAP component) to register services automatically when deploying the ONAP components.

The registrator gets notified by the POD event, updates the service info and registers service endpoint in the MSB.

- MSB uses these service info to route service requests.

MSB portal shows all registered ONAP services:

http://${Node_IP}:30082/msb

OOM uses both readiness and liveness probes to monitor ONAP components health

OOM makes use of kubernetes readiness and liveness probes feature.

> The detailed description of the kubernetes feature can be found here: https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes/

Many OOM deployment entities are ready for serving incoming calls when the component process starts listening on its listen port.

For instance, one of A&AI internal components is haproxy that routes traffic to other A&AI services (A&AI resources or A&AI traversal). When the haproxy starts listening on the ports specified in its configuration, it is ready to serve incoming requests. The haproxy port for servicing A&AI requests is 8443. This is why the A&AI haproxy deployment manifest includes:

aai-deployment.yaml

```
        readinessProbe:
          tcpSocket:
            port: 8443
          initialDelaySeconds: 5
          periodSeconds: 10
```

With the above definition, 5 seconds after the container is started, Kubernetes will start the probes and only after a probe succeeds, kubernetes will start routing traffic to it.

This is useful:

- when starting the pod - container is assumed as fully running when it passed the readiness probe
- when replacing (e.g. upgrading a pod) - the traffic will be routed to the old pod until the new pod becomes ready.

Liveness probe helps to understand when the container is not useful anymore (e.g. fatal error happened) and must be replaced. The liveness probe ha ve same parameters as the readiness probe.

For instance, to make sure that the database container is up and running, SDNC component has following definition in its DB deployment manifest:

db-statefulset.yaml

```
        livenessProbe:
          exec:
            command: ["mysqladmin", "ping"]
          initialDelaySeconds: 30
          periodSeconds: 10
          timeoutSeconds: 5
```

OOM allows for rolling upgrades of component containers

OOM uses Helm K8S package manager to deploy ONAP components.

Each component is arranged in a packaging format called a chart – a collection of files that describe a set of k8s resources.

Helm allows for rolling upgrades of the ONAP component deployed.

To upgrade a component Helm release you will need an updated Helm chart. The chart might have modified, deleted or added values, deployment yamls, and more.

To get the release name use:

```
helm ls
```

To easily upgrade the release use:

```
helm upgrade [RELEASE] [CHART]
```

To roll back to a previous release version use:

```
helm rollback [flags] [RELEASE] [REVISION]
```

for example, to upgrade the onap-mso helm release to the latest MSO container release v1.1.2:

- Edit mso valus.yaml which is part of the chart

  Change “mso: nexus3.onap.org:10001/openecomp/mso:v1.1.1” to

  “mso: nexus3.onap.org:10001/openecomp/**mso:v1.1.2**”

- From the chart location run:
  helm upgrade onap-mso ./

The previous mso pod will be terminated and a new mso pod with an updated mso container will be created.

If you’d like to learn more about Helm upgrade command, check out Helm docs. More about helm rollback here.