
MetaK8s Documentation

Release 0.1.1

Scality

Jul 19, 2018

Contents:

1	Getting started	3
1.1	Quickstart Guide	3
1.2	Architecture	6
1.3	Changes in MetalK8s	8
2	Indices and tables	11

MetalK8s is an opinionated [Kubernetes](#) distribution with a focus on long-term on-prem deployments, launched by [Scality](#) to deploy its [Zenko](#) solution in customer datacenters.

It is based on the [Kubespray](#) project to reliably install a base Kubernetes cluster, including all dependencies (like [etcd](#)), using the [Ansible](#) provisioning tool. This installation is further augmented with operational tools for monitoring and metering, including [Prometheus](#), [Grafana](#), [ElasticSearch](#) and [Kibana](#). Furthermore, an “ingress controller” is deployed by default, based on [Nginx](#). All of these are managed as [Helm](#) packages. See [Cluster Services](#) for a whole listing.

Unlike hosted Kubernetes solutions, where network-attached storage is available and managed by the provider, we assume no such system to be available in environments where MetalK8s is deployed. As such, we focus on managing node-local storage, and exposing these volumes to containers managed in the cluster. See [Storage Architecture](#) for more information.

See our *Quickstart Guide* to deploy a cluster.

1.1 Quickstart Guide

To quickly set up a testing cluster using [MetalK8s](#), you need 3 machines running [CentOS 7.4](#) to which you have SSH access (these can be VMs). Each machine acting as a [Kubernetes](#) node (all of them, in this example) also need to have at least one disk available to provision storage volumes.

Todo: Give some sizing examples

1.1.1 Defining an Inventory

To tell the [Ansible](#)-based deployment system on which machines [MetalK8s](#) should be installed, a so-called *inventory* needs to be provided. This inventory contains a file listing all the hosts comprising the cluster, as well as some configuration.

First, create a directory, e.g. `inventory/quickstart-cluster`, in which the inventory will be stored. For our setup, we need to create two files. One listing all the hosts, aptly called `hosts`:

```
node-01 ansible_host=10.0.0.1 ansible_user=centos
node-02 ansible_host=10.0.0.2 ansible_user=centos
node-03 ansible_host=10.0.0.3 ansible_user=centos

[kube-master]
node-01
node-02
node-03

[etcd]
```

(continues on next page)

(continued from previous page)

```
node-01
node-02
node-03

[kube-node]
node-01
node-02
node-03

[k8s-cluster:children]
kube-node
kube-master
```

Make sure to change IP-addresses, usernames etc. according to your infrastructure.

In a second file, called `kube-node.yml` in a `group_vars` subdirectory of our inventory, we declare how to setup storage (in the default configuration) on hosts in the `kube-node` group, i.e. hosts on which Pods will be scheduled:

```
metal_k8s_lvm:
  vgs:
    kubevg:
      drives: ['/dev/vdb']
```

In the above, we assume every `kube-node` host has a disk available as `/dev/vdb` which can be used to set up Kubernetes *PersistentVolumes*. For more information about storage, see [Storage Architecture](#).

1.1.2 Entering the MetalK8s Shell

To easily install a supported version of Ansible and its dependencies, as well as some Kubernetes tools (**kubectl** and **helm**), we provide a **make** target which installs these in a local environment. To enter this environment, run **make shell** (this takes a couple of seconds on first run):

```
$ make shell
Creating virtualenv...
Installing Python dependencies...
Downloading kubectl...
Downloading Helm...
Launching MetalK8s shell environment. Run 'exit' to quit.
(metal-k8s) $
```

Now we're all set to deploy a cluster:

```
(metal-k8s) $ ansible-playbook -i inventory/quickstart-cluster -b playbooks/deploy.yml
```

Grab a coffee and wait for deployment to end.

1.1.3 Inspecting the cluster

Once deployment finished, a file containing credentials to access the cluster is created: `inventory/quickstart-cluster/artifacts/admin.conf`. We can export this location in the shell such that the **kubectl** and **helm** tools know how to contact the cluster `kube-master` nodes, and authenticate properly:

```
(metal-k8s) $ export KUBECONFIG=`pwd`/inventory/quickstart-cluster/artifacts/admin.
↪conf
```


Now, assuming port 6443 on the first *kube-master* node is reachable from your system, we can e.g. list the nodes:

```
(metal-k8s) $ kubectl get nodes
NAME          STATUS    ROLES          AGE      VERSION
node-01       Ready    master,node    1m       v1.9.5+coreos.0
node-02       Ready    master,node    1m       v1.9.5+coreos.0
node-03       Ready    master,node    1m       v1.9.5+coreos.0
```

or list all pods:

```
(metal-k8s) $ kubectl get pods --all-namespaces
NAMESPACE     NAME                                                    READY   ̀
↳STATUS      RESTARTS   AGE
kube-ingress  nginx-ingress-controller-9d8jh                        1/1     ̀
↳Running     0          1m
kube-ingress  nginx-ingress-controller-d7vvg                        1/1     ̀
↳Running     0          1m
kube-ingress  nginx-ingress-controller-m8jpb                        1/1     ̀
↳Running     0          1m
kube-ingress  nginx-ingress-default-backend-6664bc64c9-xsws5       1/1     ̀
↳Running     0          1m
kube-ops      alertmanager-kube-prometheus-0                       2/2     ̀
↳Running     0          2m
kube-ops      alertmanager-kube-prometheus-1                       2/2     ̀
↳Running     0          2m
kube-ops      es-client-7cf569f5d8-2z974                           1/1     ̀
↳Running     0          2m
kube-ops      es-client-7cf569f5d8-qq4h2                           1/1     ̀
↳Running     0          2m
kube-ops      es-data-cd5446fff-pkmhn                              1/1     ̀
↳Running     0          2m
kube-ops      es-data-cd5446fff-zzd2h                              1/1     ̀
↳Running     0          2m
kube-ops      es-exporter-elasticsearch-exporter-7df5bcf58b-k9fdd  1/1     ̀
↳Running     3          1m
...

```

Similarly, we can list all deployed [Helm](#) applications:

```
(metal-k8s) $ helm list
NAME          REVISION   UPDATED                               STATUS   ̀
↳ CHART      NAMESPACE
es-exporter   3          Wed Apr 25 23:10:13 2018      DEPLOYED  ̀
↳ elasticsearch-exporter-0.1.2  kube-ops
fluentd       3          Wed Apr 25 23:09:59 2018      DEPLOYED  ̀
↳ fluentd-elasticsearch-0.1.4  kube-ops
heapster      3          Wed Apr 25 23:09:37 2018      DEPLOYED  ̀
↳ heapster-0.2.7                kube-system
kibana        3          Wed Apr 25 23:10:06 2018      DEPLOYED  ̀
↳ kibana-0.2.2                  kube-ops
kube-prometheus  3          Wed Apr 25 23:09:22 2018      DEPLOYED  ̀
↳ kube-prometheus-0.0.33        kube-ops
nginx-ingress  3          Wed Apr 25 23:09:09 2018      DEPLOYED  ̀
↳ nginx-ingress-0.11.1          kube-ingress
prometheus-operator  3          Wed Apr 25 23:09:14 2018      DEPLOYED  ̀
↳ prometheus-operator-0.0.15    kube-ops

```

1.1.4 Cluster Services

Various services to operate and monitor your MetalK8s cluster are provided. To access these, first create a secure tunnel into your cluster by running `kubectl proxy`. Then, while the tunnel is up and running, the following tools are available:

Service	Role	Link	Notes
Kubernetes dashboard	A general purpose, web-based UI for Kubernetes clusters	<code>http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/</code>	
Grafana	Monitoring dashboards for cluster services	<code>http://localhost:8001/api/v1/namespaces/kube-ops/services/kube-prometheus-grafana:http/proxy/</code>	
Cerebro	An administration and monitoring console for Elasticsearch clusters	<code>http://localhost:8001/api/v1/namespaces/kube-ops/services/cerebro:http/proxy/</code>	When accessing Cerebro, connect it to <code>http://elasticsearch:9200</code> to operate the MetalK8s Elasticsearch cluster.
Kibana	A search console for logs indexed in Elasticsearch	<code>http://localhost:8001/api/v1/namespaces/kube-ops/services/http:kibana:/proxy/</code>	When accessing Kibana for the first time, set up an <i>index pattern</i> for the <code>logstash-*</code> index, using the <code>@timestamp</code> field as <i>Time Filter field name</i> .

See *Cluster Services* for more information about these services and their configuration.

1.2 Architecture

1.2.1 Cluster Services

A Kubernetes cluster deployed on the Google Cloud Platform using GKE, on Microsoft Azure using AKS or even using Kops or similar tools on Amazon AWS comes with built-in tooling for centralized container log management, metrics collection, tracing, node health checking and more.

In MetalK8s, we augment a basic Kubernetes cluster deployed using the Kubespray playbook) with various tools to bring an on-premise cluster to the same level of operability.

Basic Cluster Addons

On top of the basic Kubernetes services, the following addons are deployed:

Helm / Tiller

Helm is a *package manager* for Kubernetes. It can be used to deploy various services in a Kubernetes cluster using templates to describe objects. Tiller is a cluster-side service used by the `helm` CLI tool to manage these deployments.

Heapster

Heapster is a service which collects and exposes resource consumption metrics of containers running in a cluster. The Kubernetes Dashboard uses the Heapster service, when available, to display CPU and memory usage of Pods, Deployments and more.

metrics-server

The `metrics-server` service is derived from Heapster, and provides an implementation of the `Metrics API` exposing CPU and memory consumption of containers. These metrics are in turn used by the `HorizontalPodAutoscaler` controller.

Ingress Controller

To expose `Services` to the outside world using an `Ingress` object, Kubernetes requires an `Ingress Controller` to be running in the cluster. For this purpose, MetalK8s deploys the `nginx-ingress-controller`, which uses the well-known `Nginx` HTTP server under the hood.

Metering / Monitoring

Metering and monitoring of a MetalK8s cluster is handled by the `Prometheus` stack, including the `Prometheus TSDB` for metrics storage, `Alertmanager` to send alerts when preconfigured conditions are (not) met, and `Grafana` to visualize stored metrics using predefined dashboards.

prometheus-operator

The `CoreOS Prometheus Operator` is deployed in the cluster to manage `Prometheus` instances, scrape targets and alerting rules.

kube-prometheus

We use `kube-prometheus` to provide operational insight into the Kubernetes cluster and containers managed by it. This includes predefined alerting rules and various `Grafana` dashboards.

kube-prometheus uses *prometheus-operator* to deploy all required services.

node-exporter

The `node-exporter` service is deployed to expose various node OS metrics, which are in turn captured by `Prometheus`. These metrics include CPU, memory, disk and network consumption as well as many Linux-specific values.

Grafana

To ease cluster operations, several `Grafana` dashboards are made available, including cluster-wide views and health-checks, node OS metrics, *per-Deployment* or *per-Pod* resource usage, monitoring of the `Prometheus` service itself, and many more.

Todo: Do we need to list all exported deployed with `kube-prometheus`?

Log Collection

ElasticSearch

The [ElasticSearch](#) full-text indexing service is used to ingest all container logs in a central place, and make them accessible to operators. This ElasticSearch cluster is deployed using the manifests provided in [pires/kubernetes-elasticsearch-cluster](#), which are tuned to use production-grade settings.

Cerebro

The [Cerebro](#) dashboard is a monitoring and administration tool for Elasticsearch clusters.

ElasticSearch Curator

To ensure ingested logs don't flood the ElasticSearch resources, [ElasticSearch Curator](#) is deployed with a default configuration which drops `logstash-*` indices on a given schedule.

fluentd

The [fluentd](#) service is deployed as a [DaemonSet](#) to stream all container logs into ElasticSearch.

In MetalK8s, **fluentd** has a role similar to [Logstash](#) in the *ELK* stack.

Kibana

To give operators access to the logs stored in ElasticSearch, a [Kibana](#) instance is provided.

Note: When accessing Kibana for the first time, an *index pattern* for the `logstash-*` indices needs to be configured, using `@timestamp` as *Time Filter field name*.

1.2.2 Storage Architecture

1.3 Changes in MetalK8s

1.3.1 Release 0.1.1

Note: Compatibility with future releases of MetalK8s is not guaranteed until version 1.0.0 is available. When deploying a cluster using pre-1.0 versions of this package, you may need to redeploy later.

Features added

PR #11 - run the OpenStack [ansible-hardening](#) role on nodes to apply security hardening configurations from the Security Technical Implementation Guide (STIG) (#88)

PR #127 - deploy Cerebro to manage the Elasticsearch cluster (#126)

PR #138 - update versions of Fluentd, Kibana, Elasticsearch Exporter and Kubespray

PR #140 - set up kube-prometheus to monitor CoreDNS (cfr. PR #104)

Bugs fixed

#103 - set up host anti-affinity for Elasticsearch service scheduling (PR #113)

#120 - required facts not gathered when running the *services* playbook in isolation (PR #132)

PR #134 - fix *bash-completion* in the MetalK8s Docker image

1.3.2 Release 0.1.0

This marks the first release of MetalK8s.

Note: Compatibility with future releases of MetalK8s is not guaranteed until version 1.0.0 is available. When deploying a cluster using pre-1.0 versions of this package, you may need to redeploy later.

Incompatible changes

PR #106 - the Ansible playbook which used to be called `metal-k8s.yml` has been moved to `playbooks/deploy.yml`

Features added

PR #100 - disable Elasticsearch deployment by setting `metalk8s_elasticsearch_enabled` to `false` (#98)

PR #104 - *kube-proxy* now uses *ipvs* instead of *iptables* to route *Service* addresses, in preparation for Kubernetes 1.11. The *ipvsadm* tool is installed on all *k8s-cluster* hosts.

PR #104 - use CoreDNS instead of kubedns for in-cluster DNS services, in preparation for Kubernetes 1.11.

PR #113 - deploy the Prometheus *node_exporter* on *k8s-cluster* and *etcd* hosts instead of using a *DaemonSet*

Known issues

#62 - Elasticsearch Curator may not properly prune old *logstash-** indices

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`