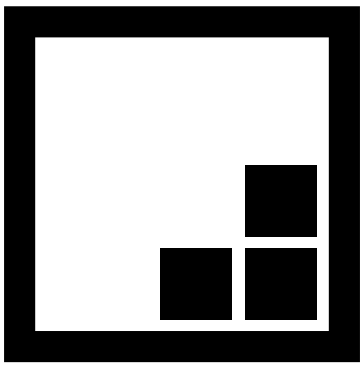

METALK8S

MetalK8s Documentation

Release 1.0.2-dev

Scality

May 02, 2019

Contents:

Part I

Installation Guide

CHAPTER 1

Introduction

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.

CHAPTER 2

Quickstart Guide

This guide describes how to set up a MetalK8s cluster. It offers general requirements and describes sizing, configuration, and deployment. With respect to installation procedures, the only significant difference between a test cluster and a full production environment is the amount of resources required to implement the design.

2.1 General Cluster Requirements

Setting up a MetalK8s cluster quickly requires at least three machines running CentOS 7.4 or higher (these can be VMs) to which you have SSH access. Each machine acting as a Kubernetes node (all three in the present example) must also have at least one disk available to provision storage volumes.

2.1.1 Sizing

Each node must satisfy the following sizing requirements.

Note: The root file system requires at least 20 GB.

Component	etcd	Master	Node
Cores	2	4	4
RAM	4 GB	8 GB	8 GB
Minimum dedicated storage capacity required	•	•	128 GB

2.1.2 Proxies

For nodes operating behind a proxy, add the following lines to each cluster server's `/etc/environment` file:

```
http_proxy=http://user;pass@<HTTP proxy IP address>:<port>
https_proxy=http://user;pass@<HTTPS proxy IP address>:<port>
no_proxy=localhost,127.0.0.1,10.*
```

2.2 Download the MetalK8s Source

Go to the MetalK8s releases page and download the source code zip file for the release you wish to install. When the download is complete, unpack the zipped file.

2.3 Define an Inventory

Each server must be configured in an inventory that identifies the servers to the Ansible-based deployment system, as well as their basic configuration, including masters and nodes.

The inventory is a directory that contains a `hosts` file, which lists all hosts in the cluster, and a subdirectory (`group_vars`) that contains `kube-node.yml`, a configuration file.

To create an inventory:

1. Log in to the machine to which you downloaded the MetalK8s project.
2. Create a directory (for example, `inventory/quickstart-cluster`) in which the inventory will be stored. Change to that directory.

   ```
   $ cd metalk8s
   $ mkdir -p inventory/quickstart-cluster
   $ cd inventory/quickstart-cluster/
   ```

3. Create the `hosts` file, which lists all 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]
   node-01
   node-02
   node-03

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

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

   Change the host names, IP addresses, and user names to conform to your infrastructure. For example, if your servers are named “server1”, “server2”, and “server3”, copy the code block above and replace ALL instances of “node-0” with “server”.

> **Warning:** Using the remote *root* user to deploy MetalK8s is not supported, see *Can I use the 'root' user to deploy MetalK8s to servers?*.

4. Create a `group_vars` subdirectory in the directory you created in step 2 (the one that contains the `hosts` file) and change to it.

```
$ mkdir group_vars ; cd group_vars
```

5. In the `group_vars` subdirectory, create a `kube-node.yml` file. This file declares how to set up hosts in the kube-node group; that is, hosts on which pods shall be scheduled:

```
metalk8s_lvm_drives_vg_metalk8s: ['/dev/vdb']
```

   This example assumes every *kube-node* host has a disk available as `/dev/vdb` that can be used to set up Kubernetes *PersistentVolumes*. For more information, see *Storage Architecture*.

2.4 Enter the MetalK8s Virtual Environment Shell

To install a supported version of Ansible and its dependencies, along with some Kubernetes tools (**`kubectl`** and **`helm`**), MetalK8s provides a **`make`** target that installs these in a local environment. To enter this environment, run **`make shell`** (this takes a few seconds when first run):

```
$ make shell
Creating virtualenv...
Installing Python dependencies...
Downloading kubectl...
Downloading Helm...
Launching MetalK8s shell environment. Run 'exit' to quit.
(metalk8s) $
```

2.5 Deploy the Cluster

Run the following command to deploy the cluster:

```
(metalk8s) $ ansible-playbook -i inventory/quickstart-cluster/hosts -b playbooks/deploy.yml
```

For a simple test deployment such as the present three-node cluster, this takes about a half hour. Actual deployment time will vary based on the size of the cluster and hardware and network performance.

2.6 Inspect the Cluster

Deployment creates a file containing credentials to access the cluster (`inventory/quickstart-cluster/artifacts/admin.conf`). Remaining in the virtual environment shell, export this location to give **`kubectl`** and **`helm`** the correct credentials to contact the cluster's *kube-master* nodes:

```
(metalk8s) $ export KUBECONFIG=`pwd`/inventory/quickstart-cluster/artifacts/admin.conf
```

If your system can reach port 6443 on the master node referred to in `admin.conf`, you can

- List all nodes:

```
(metalk8s) $ 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
```

- List all pods:

```
(metalk8s) $ 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-m8jpq              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
...
```

- List all deployed Helm applications:

```
(metalk8s) $ 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
```

2.7 Cluster Services

Services to operate and monitor your MetalK8s cluster are provided. To access these dashboards:

1. If you are accessing the cluster using a machine from which you didn't install MetalK8s, copy the

credentials in admin.conf and export its path (see *Inspect the Cluster*).

2. Open port 6443 on your cluster's master nodes for remote access to cluster services.
3. Inside a `make shell` environment, run `kubectl proxy` from your local machine. This opens a tunnel to the Kubernetes cluster, which makes the following tools available:

Service	Role	Link
Kubernetes dashboard	A general purpose, web-based UI for Kubernetes clusters	http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/
Grafana	Monitoring dashboards for cluster services	http://localhost:8001/api/v1/namespaces/kube-ops/services/kube-prometheus-grafana:http/proxy/
Cerebro	An administration and monitoring console for Elasticsearch clusters	http://localhost:8001/api/v1/namespaces/kube-ops/services/cerebro:http/proxy/
Kibana	A search console for logs indexed in Elasticsearch	http://localhost:8001/api/v1/namespaces/kube-ops/services/http:kibana:/proxy/

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

CHAPTER 3

Frequently Asked Questions

3.1 Deployment failed at *install prometheus-operator*: what should I do?

On slow networks or overloaded systems, prometheus-operator installation can time out, causing the deployment to fail. If this happens, you must follow these steps before restarting the playbook.

1. Open a **`make shell`** environment with `KUBECONFIG` set.
2. Delete and purge the *prometheus-operator* Helm release:

   ```
   helm delete --purge prometheus-operator
   ```

3. Delete the *prometheus-operator-create-sm Job*:

   ```
   kubectl --namespace=kube-ops delete job prometheus-operator-create-sm
   ```

   If the above command fails with *Error from server (NotFound)*, this is OK.

4. Delete the *prometheus-operator-get-crd Job*:

   ```
   kubectl --namespace=kube-ops delete job prometheus-operator-get-crd
   ```

   If the above command fails with *Error from server (NotFound)*, this is OK.

Re-run the playbook to finalize the deployment.

3.2 How can I keep a logfile of Ansible executions?

There are two ways to configure Ansible to keep a logfile:

- Set *log_path* in the *defaults* section of `ansible.cfg`. Relative paths are relative to the location of `ansible.cfg`.
- Export `ANSIBLE_LOG_PATH` in the environment from which **`ansible-playbook`** will be invoked.

For more information, see DEFAULT_LOG_PATH.

3.3 Can I use the 'root' user to deploy MetalK8s to servers?

During the deployment of MetalK8s, a set of tasks are executed to bring the target system in line with the RHEL7 STIG security guidelines, using the ansible-hardening role. STIG rule V-72247 does not permit remote SSH access using the *root* user. As such, if MetalK8s were deployed using *root* to access a remote system, this would effectively disable access to said server.

We integrated a check in the playbook to assert *ansible_user* is not set to *root* on any of the target hosts to abort the deployment if this configuration is detected.

To disable this security measure, set the security_sshd_permit_root_login variable to *true* on the relevant hosts or groups.

Part II

Operations Guide

CHAPTER 4

Workload Storage

Some workloads need different volumes with different storage capacities to fit their components needs. These volumes are stored in LVM *Logical Volumes*.

4.1 Volumes

Considering all storage volumes required for workloads running in the cluster, create a configuration as below:

```
metalk8s_lvm_drives_vg_metalk8s: ['/dev/vdb']
metalk8s_lvm_lvs_vg_metalk8s:
    lv01:
        size: 52G
    lv02:
        size: 52G
    lv03:
        size: 52G
    lv04:
        size: 11G
    lv05:
        size: 11G
    lv06:
        size: 11G
    lv07:
        size: 5G
    lv08:
        size: 5G
```

This configuration can be set on a whole Ansible group of nodes (see Group Variables), or on a specific host (see Host Variables).

4.2 Resize LVs

Volumes can be resized (one or several at once). Change the volume size value to a higher one and run:

```
ansible-playbook -b -i <inventory>/hosts -t storage playbooks/deploy.yml
```

4.3 Configuration layout

The configuration can be applied to groups and hosts in two different ways.

Note: Configuration files are merged for every created host or group.

To apply a configuration, create a YAML file in either (or both) `group_vars` and `host_vars` with the group name associated, or create a folder in `group_vars` or `host_vars` with several YAML files. The **`ansible-playbook`** above must be run.

4.4 Add extra LVs

It is possible to configure LVM drives and volumes for one node only.

Exemplified below, a default storage configuration (`group_vars/kube-node/storage.yml`):

```
# metalk8s_lvm_vgs = ['vg_metalk8s']
metalk8s_lvm_drives_vg_metalk8s: ['/dev/vdb']
metalk8s_lvm_lvs_vg_metalk8s:
  lv01:
      size: 52G
  lv02:
      size: 52G
  lv03:
      size: 52G
```

In `host_vars`, create a new file (`host_vars/node_1.yml`):

```
metalk8s_lvm_vgs = ['vg_metalk8s', 'mynewvg']
metalk8s_lvm_drives_mynewvg: ['/dev/vdc']
metalk8s_lvm_lvs_vg_metalk8s:
  lv01:
     size: 52G
metalk8s_lvm_lvs_mynewvg:
  lv01:
     size: 1T
```

Except `node_1`, every machine has a single *vg_metalk8s* with six *logical volumes* (three specified, three default). On `node_1`, there are two *volume groups* (*vg_metalk8s* and *mynewvg*) with four logical volumes on *vg_metalk8s* (one specified, three default) and one logical volume on *mynewvg*.

Note: As the volume group name becomes a prefix, several LVs can have the same name.

Part III

Reference Guide

CHAPTER 5

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.

5.1 Basic Cluster Addons

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

5.1.1 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.

5.1.2 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.

5.1.3 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.

5.2 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.

5.3 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.

5.3.1 prometheus-operator

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

5.3.2 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.

5.3.3 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.

5.3.4 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?

5.4 Log Collection

5.4.1 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 Helm chart, with a configuration tuned for production-grade settings.

5.4.2 Cerebro

The Cerebro dashboard is a monitoring and administration tool for Elasticsearch clusters.

5.4.3 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.

5.4.4 Fluent Bit and fluentd

The Fluent Bit service is deployed as a DaemonSet to stream all container logs into fluentd instances, which collect them and submit batches to Elasticsearch.

In MetalK8s, Fluent Bit and **`fluentd`** have a role similar to Logstash in the *ELK* stack.

5.4.5 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*.

CHAPTER 6

Storage Architecture

Storage provisioned by MetalK8s is currently backed by *LVM Logical Volumes*. A default setup will provision volumes tailored to the needs of various services deployed with MetalK8s, but this list can be extended to provide volumes which fulfil the needs of your application workloads.

CHAPTER 7

Changes in MetalK8s

7.1 Release 1.0.2 (in development)

7.1.1 Bugs fixed

#962 - Versionlocks *docker* & *node_exporter* packages

7.2 Release 1.0.1

This version updates the Kubernetes version to 1.10.11 to handle CVE-2018-100210.

7.2.1 Features added

PR #232 - Add more storage checks regarding the device presence and the partition existence on specificied drives (#231)

PR #240 - update Python *cryptography* package to 2.3

PR #274 - add support for Python 3.7

PR #305 - ensure that *journald* logs are persisted across reboots (#303)

PR #337 - assert *ansible_user* is not *root* (#329)

7.2.2 Bugs fixed

#50 - raise default *etcd* memory limits (PR #331)

#237 - increase timeout of *prometheus-operator* deployment (PR #244)

#321 - retry until PV creation succeeds in *reclaim-storage* playbook (PR #319)

#381 - warn when Bash completion is not available in *make shell* (PR #382)

#192 - *make shell* failing to start on OS X (PR #418)

#424 - remove warning related to *kube_nginx_ingress* roles (PR #425)

#399 - check that the hostnames in the inventory don't use capitals (PR #409)

PR #472 - update Python *requests* library version

PR #511 - update Kubernetes version to 1.10.11 to include a fix for CVE-2018-100210

PR #523 - reduce Tiller wait timeout to reduce CI time to failure

7.3 Release 1.0.0

This marks the first production-ready release of MetalK8s. Deployments using this release can be upgraded to later MetalK8s 1.x versions.

7.3.1 Breaking changes

PR #187 - no longer remove the MetalK8s 0.1.x Elasticsearch cluster upon upgrade (#160)

7.3.2 Features added

PR #191 - deploy *PodDisruptionBudgets* for Elasticsearch (#157)

PR #193 - update versions of *kube-prometheus*, Elasticsearch and Kubespray

PR #181 - format *PersistentVolumes* asynchronously (#173)

PR #201 - collect Calico metrics and deploy Grafana dashboards for them (#81)

PR #210 - deploy *metrics-server* using Helm (#146)

PR #189, PR #215 - collect *nginx-ingress* metrics and deploy a dashboard (#143)

PR #218 - update versions of Kibana and *fluent-bit*

PR #223 - pre-provision Kibana index configuration (#174)

7.3.3 Bugs fixed

#170 - rename *ElasticSearch Example* and *Node Exporter Full* Grafana dashboards (PR #188)

#196 - deploy the Elasticsearch Curator configuration we want to deploy instead of falling back to the chart default (PR #197)

#220 - 'Kubernetes Calico (Alternative)' dashboard doesn't work (PR #221)

7.3.4 Known issues

#179 - some Grafana dashboard charts are not displaying any metrics

7.4 Release 0.2.0

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.

7.4.1 Breaking changes

PR #159 - use upstream chart for Elasticsearch. Historical log data will be lost. Please see the pull-request description for manual steps required after upgrading a MetalK8s 0.1 cluster to MetalK8s 0.2 (#147)

PR #94 - flatten the storage configuration and allow more user defined storage related actions (#153)

7.4.2 Features added

PR #144 - update Kibana chart version

PR #145 - update the Cerebro chart, and pre-configure the MetalK8s Elasticsearch cluster

PR #154 - rework log collection architecture, now using Fluent Bit to capture logs, then forward to fluentd to aggregate them and batch-insert in Elasticsearch (#51)

PR #163 - update versions of Elasticsearch Exporter, *nginx-ingress*, *kube-prometheus* and Kubespray

7.4.3 Bugs fixed

PR #151 - fix *debug* clause *var* scoping

#150 - fix deployment of Elasticsearch, node and Prometheus Grafana dashboards (PR #158)

#139 - stabilize **`helm init`** (PR #167)

7.4.4 Known issues

#179 - some Grafana dashboard charts are not displaying any metrics

7.5 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.

7.5.1 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)

7.5.2 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

7.6 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.

7.6.1 Incompatible changes

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

7.6.2 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*

7.6.3 Known issues

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

CHAPTER 8

Glossary

LVM Physical Volume

LVM PV A volume (disk or partition) consumed by a *Volume Group* to provide storage to *Logical Volumes*.

LVM Volume Group

LVM VG A logical unit that aggregates *Physical Volumes* to provision *Logical Volumes*

LVM Logical Volume

LVM LV A volume, part of a *Volume Group*, that exposes a slice of its backing storage.

Kubernetes PersistentVolume

Kubernetes PV An existing persistent storage volume available to Kubernetes workloads.

Kubernetes PersistentVolumeClaim

Kubernetes PVC A claim on a *PersistentVolume* consumed by one or more *Pods*.

8.1 Common Environment Variables

`ANSIBLE_LOG_PATH`
: File to which Ansible will write logs on the controller. When empty, logging is disabled. See DEFAULT_LOG_PATH for more information.

`KUBECONFIG`
: Path to a file used to configure access to a Kubernetes cluster when using **`kubectl`** or other tools.

Index