From ceph-ansible to Rook

Table of Contents

OpenStack Integration

The integration into OpenStack is done by using various configuration files underneath this repository for the appropriate components:

- cinder-volume
- glance
- gnocchi
- manila
- nova

The configuration relies on prior generated Ansible facts:

```
[global]
mon host = {% for host in groups['ceph-mon'] %}{{ hostvars[host]['monitor_address'] }} \
           {% if not loop.last %},{% endif %}{% endfor %}
public network = {{ ceph_public_network }}
max open files = 131072
fsid = {{ ceph_cluster_fsid }}
```

Ceph Default

Default values regarding the configuration of Ceph are defined here and are utilised in the OSISM container-image-ceph-ansible.

Keystone

https://github.com/osism/testbed/blob/main/environments/ceph/configuration.yml#L34

```
ceph_conf_overrides:
  "client.rgw.{{ hostvars[inventory_hostname]
['ansible_hostname'] }}.rgw0":
    "rgw content length compat": "true"
    "rgw enable apis": "swift, s3, admin"
    "rgw keystone accepted admin roles": "admin"
    "rgw keystone accepted roles": "member, admin"
    "rgw keystone admin domain": "default"
    "rgw keystone admin password":
"{{ ceph_rgw_keystone_password }}"
    "rgw keystone admin project": "service"
    "rgw keystone admin tenant": "service"
    "rgw keystone admin user": "ceph_rgw"
    "rgw keystone api version": "3"
    "rgw keystone implicit tenants": "true"
    "rgw keystone url": "https://api-
int.testbed.osism.xyz:5000"
    "rgw keystone verify ssl": "false"
    "rgw s3 auth use keystone": "true"
    "rgw swift account in url": "true"
    "rgw swift versioning enabled": "true"
    "rgw verify ssl": "false"
```

Misc defaults

https://github.com/osism/testbed/blob/main/environments/ceph/configuration.yml#L34

```
ceph_conf_overrides:
  global:
    auth allow insecure global id reclaim: false
    # NOTE: default size of 2 because by default
there are only 2 nodes
    osd pool default size: 2
    osd pool default min size: 0
  mon:
    mon allow pool delete: true
```

Current deployment method utilizing ceph-ansible

Currently the upstream ceph-ansible functionality is supplemented by container-image-ceph-ansible.

The following sequence describes the current testbed ceph deployment when executing `make ceph`:

1. *testbed/Makefile* → target: *ceph*
2. *testbed/terraform/Makefile* → target: *deploy-ceph*
3. *testbed/scripts/deploy/100-ceph-services-basic.sh*
   This script prepares and created LVM volumes and finally runs the following code:

```
osism apply ceph
if [[ $MANAGER_VERSION =~ ^7\.[0-9]\.[0-9]$ ||
      $MANAGER_VERSION == "latest" ]]; then
    osism apply ceph-pools
fi
osism apply copy-ceph-keys
osism apply cephclient
osism apply ceph-bootstrap-dashboard
```

Compare Ceph Ansible vs Rook

A comparison of the two setups can be found in this document. You can also find example settings for a testbed deployment using the ceph-ansible method.

Manual POC with Rook

1. Create a minimal setup with *make manager* via the OSISM testbed
2. Manually execute the steps to prepare the LVM volumes (from 100-ceph-services-basic.sh)

```
osism apply ceph-configure-lvm-volumes
for node in $(find /opt/configuration/inventory/host_vars -mindepth 1 -type d); do
    if [[ -e /tmp/$(basename $node)-ceph-lvm-configuration.yml ]]; then
        cp /tmp/$(basename $node)-ceph-lvm-configuration.yml \
/opt/configuration/inventory/host_vars/$(basename $node)/ceph-lvm-configuration.yml
    fi
done
osism reconciler sync
osism apply ceph-create-lvm-devices
osism apply facts
```

3. Deploy Kubernetes with `/opt/configuration/scripts/deploy/005-kubernetes.sh` on the manager.
4. Install the rook operator with *osism apply rook-operator*
5. Execute *rook apply cluster.yml* and pass it a custom cluster file. An example is below

```
#################################################################################################################
# Define the settings for the rook-ceph cluster with common settings for a production cluster.
# All nodes with available raw devices will be used for the Ceph cluster. At least three nodes are required
# in this example. See the documentation for more details on storage settings available.

# For example, to create the cluster:
#   kubectl create -f crds.yaml -f common.yaml -f operator.yaml
#   kubectl create -f cluster.yaml
#################################################################################################################

apiVersion: ceph.rook.io/v1
kind: CephCluster
metadata:
  name: rook-ceph
  namespace: rook-ceph # namespace:cluster
spec:
  cephVersion:
    # The container image used to launch the Ceph daemon pods (mon, mgr, osd, mds, rgw).
    # v17 is Quincy, v18 is Reef.
    # RECOMMENDATION: In production, use a specific version tag instead of the general v17 flag, which pulls the latest release and
could result in different
    # versions running within the cluster. See tags available at https://hub.docker.com/r/ceph/ceph/tags/.
    # If you want to be more precise, you can always use a timestamp tag such as quay.io/ceph/ceph:v18.2.1-20240103
    # This tag might not contain a new Ceph version, just security fixes from the underlying operating system, which will reduce
vulnerabilities
    image: quay.io/ceph/ceph:v18.2.1
    # Whether to allow unsupported versions of Ceph. Currently `quincy` and `reef` are supported.
    # Future versions such as `squid` (v19) would require this to be set to `true`.
    # Do not set to true in production.
    allowUnsupported: false
  # The path on the host where configuration files will be persisted. Must be specified.
  # Important: if you reinstall the cluster, make sure you delete this directory from each host or else the mons will fail to start
on the new cluster.
  # In Minikube, the '/data' directory is configured to persist across reboots. Use "/data/rook" in Minikube environment.
  dataDirHostPath: /var/lib/rook
  # Whether or not upgrade should continue even if a check fails
  # This means Ceph's status could be degraded and we don't recommend upgrading but you might decide otherwise
  # Use at your OWN risk
  # To understand Rook's upgrade process of Ceph, read https://rook.io/docs/rook/latest/ceph-upgrade.html#ceph-version-upgrades
  skipUpgradeChecks: false
  # Whether or not continue if PGs are not clean during an upgrade
  continueUpgradeAfterChecksEvenIfNotHealthy: false
  # WaitTimeoutForHealthyOSDInMinutes defines the time (in minutes) the operator would wait before an OSD can be stopped for upgrade
or restart.
```

```yaml
  # If the timeout exceeds and OSD is not ok to stop, then the operator would skip upgrade for the current OSD and proceed with the next one
  # if `continueUpgradeAfterChecksEvenIfNotHealthy` is `false`. If `continueUpgradeAfterChecksEvenIfNotHealthy` is `true`, then operator would
  # continue with the upgrade of an OSD even if its not ok to stop after the timeout. This timeout won't be applied if `skipUpgradeChecks` is `true`.
  # The default wait timeout is 10 minutes.
  waitTimeoutForHealthyOSDInMinutes: 10
  mon:
    # Set the number of mons to be started. Generally recommended to be 3.
    # For highest availability, an odd number of mons should be specified.
    count: 3
    # The mons should be on unique nodes. For production, at least 3 nodes are recommended for this reason.
    # Mons should only be allowed on the same node for test environments where data loss is acceptable.
    allowMultiplePerNode: false
  mgr:
    # When higher availability of the mgr is needed, increase the count to 2.
    # In that case, one mgr will be active and one in standby. When Ceph updates which
    # mgr is active, Rook will update the mgr services to match the active mgr.
    count: 2
    allowMultiplePerNode: false
    modules:
      # List of modules to optionally enable or disable.
      # Note the "dashboard" and "monitoring" modules are already configured by other settings in the cluster CR.
      # - name: rook
      #   enabled: true
  # enable the ceph dashboard for viewing cluster status
  dashboard:
    enabled: true
    # serve the dashboard under a subpath (useful when you are accessing the dashboard via a reverse proxy)
    # urlPrefix: /ceph-dashboard
    # serve the dashboard at the given port.
    # port: 8443
    # serve the dashboard using SSL
    ssl: true
    # The url of the Prometheus instance
    # prometheusEndpoint: <protocol>://<prometheus-host>:<port>
    # Whether SSL should be verified if the Prometheus server is using https
    # prometheusEndpointSSLVerify: false
  # enable prometheus alerting for cluster
  monitoring:
    # requires Prometheus to be pre-installed
    enabled: false
    # Whether to disable the metrics reported by Ceph. If false, the prometheus mgr module and Ceph exporter are enabled.
    # If true, the prometheus mgr module and Ceph exporter are both disabled. Default is false.
    metricsDisabled: false
  network:
    connections:
      # Whether to encrypt the data in transit across the wire to prevent eavesdropping the data on the network.
      # The default is false. When encryption is enabled, all communication between clients and Ceph daemons, or between Ceph daemons will be encrypted.
      # When encryption is not enabled, clients still establish a strong initial authentication and data integrity is still validated with a crc check.
      # IMPORTANT: Encryption requires the 5.11 kernel for the latest nbd and cephfs drivers. Alternatively for testing only,
      # you can set the "mounter: rbd-nbd" in the rbd storage class, or "mounter: fuse" in the cephfs storage class.
      # The nbd and fuse drivers are *not* recommended in production since restarting the csi driver pod will disconnect the volumes.
      encryption:
        enabled: false
      # Whether to compress the data in transit across the wire. The default is false.
      # See the kernel requirements above for encryption.
      compression:
        enabled: false
      # Whether to require communication over msgr2. If true, the msgr v1 port (6789) will be disabled
      # and clients will be required to connect to the Ceph cluster with the v2 port (3300).
      # Requires a kernel that supports msgr v2 (kernel 5.11 or CentOS 8.4 or newer).
      requireMsgr2: false
    # enable host networking
    provider: host
    addressRanges:
      public:
        - "192.168.16.0/20"
      cluster:
        - "192.168.16.0/20"
    # enable the Multus network provider
    #provider: multus
    #selectors:
    #  The selector keys are required to be `public` and `cluster`.
    #  Based on the configuration, the operator will do the following:
    #    1. if only the `public` selector key is specified both public_network and cluster_network Ceph settings will listen on that interface
    #    2. if both `public` and `cluster` selector keys are specified the first one will point to 'public_network' flag and the second one to 'cluster_network'
    #
    #  In order to work, each selector value must match a NetworkAttachmentDefinition object in Multus
    #
    #  public: public-conf --> NetworkAttachmentDefinition object name in Multus
    #  cluster: cluster-conf --> NetworkAttachmentDefinition object name in Multus
    # Provide internet protocol version. IPv6, IPv4 or empty string are valid options. Empty string would mean IPv4
    #ipFamily: "IPv6"
    # Ceph daemons to listen on both IPv4 and Ipv6 networks
    #dualStack: false
    # Enable multiClusterService to export the mon and OSD services to peer cluster.
    # This is useful to support RBD mirroring between two clusters having overlapping CIDRs.
    # Ensure that peer clusters are connected using an MCS API compatible application, like Globalnet Submariner.
    #multiClusterService:
    #  enabled: false

  # enable the crash collector for ceph daemon crash collection
  crashCollector:
    disable: false
    # Uncomment daysToRetain to prune ceph crash entries older than the
    # specified number of days.
    #daysToRetain: 30
  # enable log collector, daemons will log on files and rotate
  logCollector:
    enabled: true
    periodicity: daily # one of: hourly, daily, weekly, monthly
    maxLogSize: 500M # SUFFIX may be 'M' or 'G'. Must be at least 1M.
  # automate [data cleanup process](https://github.com/rook/rook/blob/master/Documentation/Storage-Configuration/ceph-teardown.md#delete-the-data-on-hosts) in cluster destruction.
  cleanupPolicy:
    # Since cluster cleanup is destructive to data, confirmation is required.
    # To destroy all Rook data on hosts during uninstall, confirmation must be set to "yes-really-destroy-data".
    # This value should only be set when the cluster is about to be deleted. After the confirmation is set,
    # Rook will immediately stop configuring the cluster and only wait for the delete command.
    # If the empty string is set, Rook will not destroy any data on hosts during uninstall.
    confirmation: ""
    # sanitizeDisks represents settings for sanitizing OSD disks on cluster deletion
    sanitizeDisks:
      # method indicates if the entire disk should be sanitized or simply ceph's metadata
      # in both case, re-install is possible
      # possible choices are 'complete' or 'quick' (default)
      method: quick
      # dataSource indicate where to get random bytes from to write on the disk
      # possible choices are 'zero' (default) or 'random'
      # using random sources will consume entropy from the system and will take much more time then the zero source
      dataSource: zero
      # iteration overwrite N times instead of the default (1)
      # takes an integer value
      iteration: 1
    # allowUninstallWithVolumes defines how the uninstall should be performed
    # If set to true, cephCluster deletion does not wait for the PVs to be deleted.
    allowUninstallWithVolumes: false
  # To control where various services will be scheduled by kubernetes, use the placement configuration sections below.
  # The example under 'all' would have all services scheduled on kubernetes nodes labeled with 'role=storage-node' and
  # tolerate taints with a key of 'storage-node'.
  # placement:
```

```
  #   all:
  #     nodeAffinity:
  #       requiredDuringSchedulingIgnoredDuringExecution:
  #         nodeSelectorTerms:
  #         - matchExpressions:
  #           - key: role
  #             operator: In
  #             values:
  #             - storage-node
  #     podAffinity:
  #     podAntiAffinity:
  #     topologySpreadConstraints:
  #     tolerations:
  #     - key: storage-node
  #       operator: Exists
  # The above placement information can also be specified for mon, osd, and mgr components
  #   mon:
  #placement:
  #  mon:
  #    nodeAffinity:
  #      requiredDuringSchedulingIgnoredDuringExecution:
  #        nodeSelectorTerms:
  #        - matchExpressions:
  #          - key: ceph-mon-placement
  #            operator: In
  #            values:
  #            - enabled
  #  mgr:
  #    nodeAffinity:
  #      requiredDuringSchedulingIgnoredDuringExecution:
  #        nodeSelectorTerms:
  #        - matchExpressions:
  #          - key: ceph-mgr-placement
  #            operator: In
  #            values:
  #            - enabled
  # Monitor deployments may contain an anti-affinity rule for avoiding monitor
  # collocation on the same node. This is a required rule when host network is used
  # or when AllowMultiplePerNode is false. Otherwise this anti-affinity rule is a
  # preferred rule with weight: 50.
  #   osd:
  #    prepareosd:
  #    mgr:
  #    cleanup:
  annotations:
  #   all:
  #   mon:
  #   osd:
  #   cleanup:
  #   prepareosd:
  # clusterMetadata annotations will be applied to only `rook-ceph-mon-endpoints` configmap and the `rook-ceph-mon` and `rook-ceph-admin-keyring` secrets.
  # And clusterMetadata annotations will not be merged with `all` annotations.
  #    clusterMetadata:
  #       kubed.appscode.com/sync: "true"
  # If no mgr annotations are set, prometheus scrape annotations will be set by default.
  #   mgr:
  labels:
  #   all:
  #   mon:
  #   osd:
  #   cleanup:
  #   mgr:
  #   prepareosd:
  # monitoring is a list of key-value pairs. It is injected into all the monitoring resources created by operator.
  # These labels can be passed as LabelSelector to Prometheus
  #   monitoring:
  #   crashcollector:
```

```
resources:
#The requests and limits set here, allow the mgr pod to use half of one CPU core and 1 gigabyte of memory
#   mgr:
#     limits:
#       cpu: "500m"
#       memory: "1024Mi"
#     requests:
#       cpu: "500m"
#       memory: "1024Mi"
# The above example requests/limits can also be added to the other components
#   mon:
#   osd:
# For OSD it also is a possible to specify requests/limits based on device class
#   osd-hdd:
#   osd-ssd:
#   osd-nvme:
#   prepareosd:
#   mgr-sidecar:
#   crashcollector:
#   logcollector:
#   cleanup:
#   exporter:
# The option to automatically remove OSDs that are out and are safe to destroy.
removeOSDsIfOutAndSafeToRemove: false
priorityClassNames:
  #all: rook-ceph-default-priority-class
  mon: system-node-critical
  osd: system-node-critical
  mgr: system-cluster-critical
  #crashcollector: rook-ceph-crashcollector-priority-class
storage: # cluster level storage configuration and selection
  useAllNodes: false
  useAllDevices: false
  deviceFilter: "^sd[b-c]"
  config:
    # crushRoot: "custom-root" # specify a non-default root label for the CRUSH map
    # metadataDevice: "md0" # specify a non-rotational storage so ceph-volume will use it as block db device of bluestore.
    metadataDevice: ""
    # databaseSizeMB: "1024" # uncomment if the disks are smaller than 100 GB
    osdsPerDevice: "1" # this value can be overridden at the node or device level
    encryptedDevice: "true" # the default value for this option is "false"
  # Individual nodes and their config can be specified as well, but 'useAllNodes' above must be set to false. Then, only the named
  # nodes below will be used as storage resources.  Each node's 'name' field should match their 'kubernetes.io/hostname' label.
  # nodes:
  #   - name: "172.17.4.201"
  #     devices: # specific devices to use for storage can be specified for each node
  #       - name: "sdb"
  #       - name: "nvme01" # multiple osds can be created on high performance devices
  #         config:
  #           osdsPerDevice: "5"
  #       - name: "/dev/disk/by-id/ata-ST4000DM004-XXXX" # devices can be specified using full udev paths
  #     config: # configuration can be specified at the node level which overrides the cluster level config
  #   - name: "172.17.4.301"
  #     deviceFilter: "^sd."
  nodes:
    - name: "testbed-node-0"
      devices:
        - name: "/dev/sdb"
        - name: "/dev/sdc"
    - name: "testbed-node-1"
      devices:
        - name: "/dev/sdb"
        - name: "/dev/sdc"
    - name: "testbed-node-2"
      devices:
        - name: "/dev/sdb"
        - name: "/dev/sdc"
  # when onlyApplyOSDPlacement is false, will merge both placement.All() and placement.osd
```

```yaml
    onlyApplyOSDPlacement: false
    # Time for which an OSD pod will sleep before restarting, if it stopped due to flapping
    # flappingRestartIntervalHours: 24
  # The section for configuring management of daemon disruptions during upgrade or fencing.
  disruptionManagement:
    # If true, the operator will create and manage PodDisruptionBudgets for OSD, Mon, RGW, and MDS daemons. OSD PDBs are managed dynamically
    # via the strategy outlined in the [design](https://github.com/rook/rook/blob/master/design/ceph/ceph-managed-disruptionbudgets.md). The operator will
    # block eviction of OSDs by default and unblock them safely when drains are detected.
    managePodBudgets: true
    # A duration in minutes that determines how long an entire failureDomain like `region/zone/host` will be held in `noout` (in addition to the
    # default DOWN/OUT interval) when it is draining. This is only relevant when  `managePodBudgets` is `true`. The default value is `30` minutes.
    osdMaintenanceTimeout: 30
    # A duration in minutes that the operator will wait for the placement groups to become healthy (active+clean) after a drain was completed and OSDs came back up.
    # Operator will continue with the next drain if the timeout exceeds. It only works if `managePodBudgets` is `true`.
    # No values or 0 means that the operator will wait until the placement groups are healthy before unblocking the next drain.
    pgHealthCheckTimeout: 0

  # csi defines CSI Driver settings applied per cluster.
  csi:
    readAffinity:
      # Enable read affinity to enable clients to optimize reads from an OSD in the same topology.
      # Enabling the read affinity may cause the OSDs to consume some extra memory.
      # For more details see this doc:
      # https://rook.io/docs/rook/latest/Storage-Configuration/Ceph-CSI/ceph-csi-drivers/#enable-read-affinity-for-rbd-volumes
      enabled: false

    # cephfs driver specific settings.
    cephfs:
      # Set CephFS Kernel mount options to use https://docs.ceph.com/en/latest/man/8/mount.ceph/#options.
      # kernelMountOptions: ""
      # Set CephFS Fuse mount options to use https://docs.ceph.com/en/quincy/man/8/ceph-fuse/#options.
      # fuseMountOptions: ""

  # healthChecks
  # Valid values for daemons are 'mon', 'osd', 'status'
  healthCheck:
    daemonHealth:
      mon:
        disabled: false
        interval: 45s
      osd:
        disabled: false
        interval: 60s
      status:
        disabled: false
        interval: 60s
    # Change pod liveness probe timing or threshold values. Works for all mon,mgr,osd daemons.
    livenessProbe:
      mon:
        disabled: false
      mgr:
        disabled: false
      osd:
        disabled: false
    # Change pod startup probe timing or threshold values. Works for all mon,mgr,osd daemons.
    startupProbe:
      mon:
        disabled: false
      mgr:
        disabled: false
      osd:
        disabled: false
```