Orchestrator - Feature #44205

Feature # 43962 (Resolved): cephadm: Make mgr/cephadm declarative

cephadm: push/apply config.yml

02/19/2020 10:51 AM - Joshua Schmid

Status:	Resolved	**% Done:**	0%
Priority:	Normal		
Assignee:	Joshua Schmid		
Category:	cephadm		
Target version:	v15.0.0		
Source:	Community (dev)	**Reviewed:**	
Tags:		**Affected Versions:**	
Backport:		**Pull request ID:**	33553

Description

Having a push/apply-config option would enable us to define multiple services/daemons before the actual deployment.
This may be helpful for one-button deployments such as POCs or reproducer environments.

For the technical side:

We have to define a structure that the orchestrator understands.
Since we have **ServiceSpec**[0] and **PlacementSpec**[1] that can parse **from_json/yaml()**
this shouldn't be too hard.

After parsing the SeriveSpecs we have to call the corresponding **_add_$component()**
functions in order. We have to take care that:

- We execute in the right order (not services that create pools before OSDs exist)
  - and respect dependencies (mds before NFS/RGW if specified)
- aggregate the completion objects and wait/track progress if async

The config will probably be saved in the persistent mon store and should be inspectable

example config.yaml:

```
service_type: mon
placement:
  count: 1 #optional (mutex with hosts)
  label: foo #optional (mutex with hosts)
  hosts: #optional (mutex with label and count)
    - 'hostname1:ip/CIDR/addr_vec=optional_name1' # an example for a HostSpec
    - 'hostname2:ip/CIDR/addr_vec=optional_name2'
    - 'hostname3:ip/CIDR/addr_vec=optional_name3'
---
service_type: osd
spec:
  drivegroups:
    default_drivegroup:
      host_pattern: foo*
      data_devices:
        all: True
---
service_type: rgw: #more rgw_custom entries if more than one realm/zone
placement:
  count: 1 #optional (mutex with hosts)
  label: foo #optional (mutex with hosts)
  hosts: #optional (mutex with label and count)
    - x
    - y
    - z
spec:
  rgw_realm: realm1
  rgw_zone: zone1

# Similar options for mds/nfs/mgr etc
```

CLI:

```
ceph <orch> <config> apply -i config.yaml
ceph <orch> <config> show
```

The actual syntax is totally up for discussion. Please leave your suggestions in the comments.

[0]
https://github.com/ceph/ceph/blob/e5933d5e47fb2e2b77f37678ce770a1887d54c08/src/pybind/mgr/orchestrator/_interface.py#L1338

[1]
https://github.com/ceph/ceph/blob/e5933d5e47fb2e2b77f37678ce770a1887d54c08/src/pybind/mgr/orchestrator/_interface.py#L1112

History

#1 - 02/19/2020 10:53 AM - Joshua Schmid

- Description updated

#2 - 02/19/2020 10:53 AM - Joshua Schmid

- Category set to cephadm

- Target version set to v15.0.0

- Source set to Community (dev)

#3 - 02/19/2020 11:09 AM - Sebastian Wagner

hm, what about not inventing a new schema here? and instead simply concatenate the service specs for all types?

Like adding a service_type to ServiceSpec in
https://github.com/ceph/ceph/blob/b24230a74bf92eeb0dfabb3ed9efae0d7e814b0f/src/pybind/mgr/orchestrator/_interface.py#L1338

and then decode each individual spec independently?

```
service_type: mon
placement:
  count: 1 #optional (mutex with hosts)
  label: foo #optional (mutex with hosts)
  hosts: #optional (mutex with label and count)
    - x
    - y
    - z
---
service_type: osd
spec:
  drivegroups:
    default_drivegroup:
      host_pattern: foo*
      data_devices:
        all: True
---
service_type: rgw: #more rgw_custom entries if more than one realm/zone
placement:
  count: 1 #optional (mutex with hosts)
  label: foo #optional (mutex with hosts)
  hosts: #optional (mutex with label and count)
    - x
    - y
    - z
spec:
  rgw_realm: realm1
  rgw_zone: zone1

# Similar options for mds/nfs/mgr etc
```

This would create a direct relationship between the yaml definitions and the ServiceSpec class!

#4 - 02/19/2020 11:12 AM - Joshua Schmid

- Description updated

#5 - 02/19/2020 11:20 AM - Joshua Schmid

- Description updated

#6 - 02/19/2020 11:22 AM - Joshua Schmid

Sebastian Wagner wrote:

> hm, what about not inventing a new schema here? and instead simply concatenate the service specs for all types?
> [..snip..]
> This would create a direct relationship between the yaml definitions and the ServiceSpec class!

Even better.. I'll migrate to your format in the description

#7 - 02/19/2020 11:23 AM - Joshua Schmid

- Description updated

#8 - 02/19/2020 11:29 AM - Joshua Schmid

- Description updated

#9 - 02/24/2020 01:30 PM - Sebastian Wagner

- Status changed from New to In Progress

#10 - 02/24/2020 01:33 PM - Sebastian Wagner

to sum up our discussion from Friday:

- What about doing all calls synchronously and only return async completions from the the orch interface?
- have apply_specs() from serve() be the only way to make changes to the cluster
- CLI returns as soon as the change was successfully and persistently scheduled. Not necessary completed.

#11 - 02/26/2020 01:46 PM - Joshua Schmid

- Status changed from In Progress to Fix Under Review

- Pull request ID set to 33553

#12 - 03/03/2020 08:33 PM - Sage Weil

- Status changed from Fix Under Review to Resolved