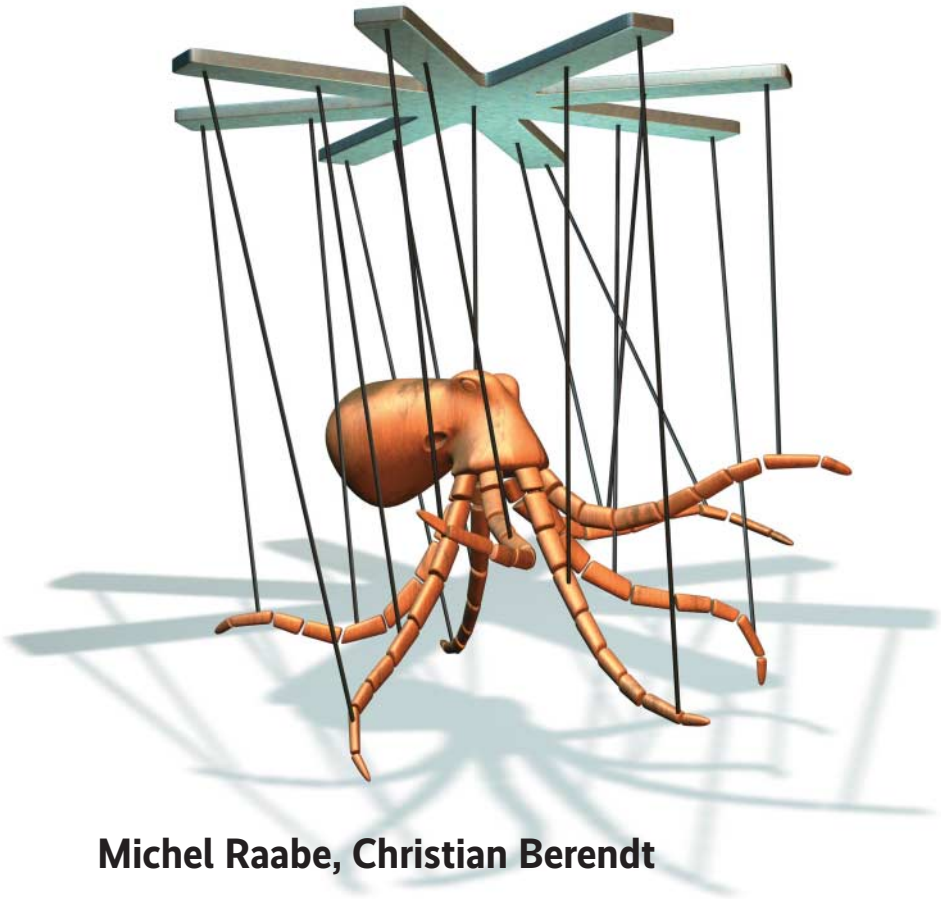
Ceph mit Ansible-Playbooks ausrollen und betreiben

Durchgespielt



Michel Raabe, Christian Berendt

Wer einen Ceph-Cluster nicht nur experimentell betreiben will, kommt um ein geeignetes Management-Framework nicht herum. Red Hat hat dafür zum Shootingstar Ansible gegriffen und fertige Playbooks zum Verteilen und Verwalten der Ceph-Knoten erstellt.

iX-TRACT

- Zum produktiven Betrieb eines Ceph-Clusters zählt auch ein geeignetes Management-Framework, mit dem sich die Knoten zügig und komfortabel bereitstellen und anschließend pflegen lassen.
- Red Hat hat mit Ansible Playbooks for Ceph (ceph-ansible) ein Instrument geschaffen, das mit Ceph-Clustern in unterschiedlichsten Konfigurationen umgehen kann.
- Playbooks existieren nicht nur zum Installieren, Verwalten, Erweitern und Aktualisieren eines Ceph-Clusters, sondern auch zum Übernehmen bereits bestehender Installationen.

Ceph, der hochverfügbare, verteilte und robuste De-facto-Standard für SDS (Software-defined Storage), wird heute in den unterschiedlichsten Größenordnungen eingesetzt. Unter den Anwendern sind große Forschungseinrichtungen wie das CERN oder Betreiber großer OpenStack-basierter Public Clouds wie die Deutsche Telekom.

Ceph basiert auf einem einheitlichen Object Storage und lässt sich als objekt-, block- oder dateibasierter Speicher nutzen. Das SDS-Konzept versucht vor allem drei Anforderungen Rechnung zu tragen, bei denen klassische Enterprise-Storage-Systeme an ihre Grenzen stoßen können: Das Storage-System soll insbesondere durch das bequeme Hinzufügen von Storage-Knoten in die Breite skalieren, die Software soll die Daten zudem flexibel und selbstständig auf dem Cluster verteilen sowie Single Points of Failure vermeiden.

Der Nachteil gegenüber klassischem Enterprise Storage liegt in dem Aufwand, den das Vorbereiten der beteiligten Knoten, das Bereitstellen, Verteilen und Konfigurieren der einzelnen Dienste sowie Erweiterungen und Wartungsarbeiten mit sich bringen. Hierfür hatte die von Red Hat übernommene Ceph-Entwicklerfirma Inktank ursprünglich das Werkzeug *ceph-deploy* ersonnen, mit dem sich die Komponenten des Ceph-Clusters automatisch ausrollen lassen [1].

Nicht geeignet ist *ceph-deploy* aber für das Verwalten von Langzeitinstallationen. So ist auch in der README-Datei von *ceph-deploy* zu lesen: „If you set up and tear down Ceph clusters a lot […] this is for you.“ Im Fokus von *ceph-deploy* stehen also Entwickler.

Rollen, in Drehbüchern niedergeschrieben

Für größere Installationen, die einen höheren Automatisierungsgrad voraussetzen, benötigt man deshalb unabhängige Werkzeuge [2]. Neben selbst erstellten Hilfsmitteln mit Automatisierungs-Frameworks wie Puppet, Chef und Ansible existieren mittlerweile etablierte Projekte, die vor allem die Ceph-Distributoren vorantreiben. Dazu zählt einerseits SUSEs Management-Framework DeepSea, das auf SaltStack aufsetzt, andererseits das von Red Hat für die Bereitstellung des Red Hat Ceph Storage genutzte Framework ceph-ansible, das den Shootingstar der Continuous-Integration-Werkzeuge Ansible verwendet [3]. Mit diesem Ansible Playbooks for Ceph will Red Hat

das Bereitstellen und Pflegen von Ceph-Clustern komfortabler gestalten und damit auch beschleunigen.

ceph-ansible setzt sich aus einer Reihe aufeinander aufbauender Rollen für Ansible zusammen. Es eignet sich für paket- und containerbasierte Installationen eines Ceph-Clusters auf CentOS, Red Hat Enterprise Linux, Ubuntu und Debian. Ebenso kann es Updates und Upgrades durchführen, bestehende Cluster übernehmen, neue Nodes und neue Platten hinzufügen, außerdem mit mehreren Ceph- sowie Ansible-Releases sowie mit vielen OSD-Szenarien (Object Storage Daemons) umgehen, etwa Collocated und Non-collocated, Logical Volume Manager, Filestore oder BlueStore. ceph-ansible übernimmt dabei Verwaltungsaufgaben wie das Erstellen von Pools oder das Ausrollen der CRUSH Map. Zudem ist die Integration einzelner Rollen in andere Frameworks möglich.

Grundsätzlich lässt sich das Betriebssystem frei wählen. Momentan gibt es aber Einschränkungen für Debian 9 aufgrund von Compiler-Bugs. Darüber hinaus existieren momentan noch keine iSCSI-RBD-Target-Pakete für Ubuntu – was sich aber zeitnah ändern sollte. Grundsätzlich testen die Entwickler Ubuntu und CentOS ausführlich.

Dem Spiel die Bühne bereiten

Der Artikel zeigt anhand von CentOS 7 im Zusammenspiel mit Docker, wie ceph-ansible einen vollwertigen Ceph-Cluster bereitstellt und anschließend die Verwaltungsaufgaben und Änderungen durchführt. Derzeit sollte noch Ansible 2.4 installiert werden. Zudem empfiehlt es sich, eine virtuelle Maschine für Ansible und später ceph-ansible zu nutzen.

Die Installation sieht einen Jumphost, fünf Ceph-Systeme mit je fünf OSDs, drei Mons, also Ceph Monitor Daemons, ein iSCSI-Gateway sowie einen MDS (Metadata Server) vor. Als Storage-Backend soll BlueStore zum Einsatz kommen [4].

Sind alle Systeme installiert, kann Ansible auf dem Jumphost Platz nehmen. Ist die richtige Version nicht im Repository vorhanden, sollte man nicht einfach eine andere Version über das Paketverwaltungsprogramm *pip* installieren, sondern stattdessen mit *virtualenv* eine virtuelle Arbeitsumgebung dafür schaffen. Das hat den Vorteil, dass man dadurch nicht in Abhängigkeiten zu bestimmten System Libraries gerät. Für die Installation genügen wenige Befehle:

```
yum install python-virtualenv python-netaddr
virtualenv pipenv/
source pipenv/bin/activate
pip install ansible==2.4.2
```

Damit ist die Basisinstallation schon abgeschlossen und ceph-ansible kann installiert werden. Als der Artikel entstand, hatte das letzte stabile git-Tag des Projekts die Version 3.0.47. Es empfiehlt sich, einen eigenen Branch anzulegen, damit es nicht zu Konflikten kommt. Später können die neueren Releases beziehungsweise Tags in den Branch gemergt werden:

```
git clone https://github.com/ceph/ ↵
                                ceph-ansible.git
cd ceph-ansible.git
git checkout v3.0.47
git checkout -b oursetup
```

Sich auf das Spiel einstimmen

Für die Installation von Ceph über ceph-ansible ist als Erstes das Inventory anzulegen. Dies beinhaltet alle beteiligten Hosts, aufgeteilt in die jeweiligen Gruppen, die die Tabelle „Verfügbare ceph-ansible-Gruppen“ auflistet. Die einzelnen Hosts können in mehreren Gruppen aufgeführt sein.

Listing 1: *inventory*

```
[mons]
ceph-mon0 monitor_address=10.20.30.40
ceph-mon1 monitor_address=10.20.30.41
ceph-mon2 monitor_address=10.20.30.42

[mgrs]
ceph-mon0
ceph-mon1
ceph-mon2

[osds]
ceph-osd0
ceph-osd1
ceph-osd2
ceph-osd3
ceph-osd4

[rgws]
ceph-rgw0

[nfss]
ceph-rgw0

[mdss]
ceph-mon0
ceph-mon1
ceph-mon2

[clients]
ceph-admin
```

Listing 2: Ausgabe von *ansible -i inventory -m ping all*

```
ansible -i inventory -m ping all
ceph-mon0 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
ceph-osd0 | SUCCESS => {
    "changed": false,
    "ping": "pong"
[...]
```

Für die Konfiguration des Beispiel-Setups ist innerhalb des Repository-Verzeichnisses *ceph-ansible* eine Datei *inventory* zu erstellen. Den Inhalt mit der Ceph-Konfiguration zeigt Listing 1.

Ob die Verbindung zu allen Knoten steht, kann man am einfachsten auf der Konsole mit dem Ansible-Modul *ping* testen, dem man die Zieladressen aus der Datei *inventory* übergibt, wie in Listing 2 zu sehen. Gegebenenfalls ist das Akzeptieren von Host-Keys erforderlich.

Ansible bietet darüber hinaus weitere Wege und Plug-ins an, über die sich das Inventory erstellen lässt, indem man sich diese Daten etwa aus einer OpenStack-Installation holt.

Sind alle Knoten vom Jumphost aus erreichbar, kann man die Parameter für die Ceph-Installation festlegen. Dazu erstellt man die beiden Verzeichnisse *group_vars* und *host_vars*. In ihnen liegen später die Dateien, die alle Variablen für Ceph und die einzelnen Hosts enthalten. ceph-ansible bietet eine ganze

Listing 3: *group_vars/all.yml*

```
ceph_origin: repository
ceph_repository: community
ceph_stable_release: luminous
public_network: "10.20.30.0/24"
```

Listing 4: Ausgabe des Befehls *ceph status*

```
  cluster:
    id:     981a9ba1-a1ff-4a82-89ef-5e88849418e6
    health: HEALTH_WARN
            Reduced data availability: 16 pgs inactive

  services:
    mon: 3 daemons, quorum ceph-mon0,ceph-mon1,ceph-mon2
    mgr: ceph-mon0(active), standbys: ceph-mon1, ceph-mon2
    mds: cephfs-1/1/1 up  {0=ceph-mon1=up:creating}, 2 up:standby
    osd: 0 osds: 0 up, 0 in

  data:
    pools:   2 pools, 16 pgs
    objects:  0 objects, 0 bytes
    usage:   0 kB used, 0 kB / 0 kB avail
    pgs:     100.000% pgs unknown
             16 unknown
```

Verfügbare ceph-ansible-Gruppen

Gruppe	Typ
mons	Monitor Daemon
osds	Object Storage Daemon
rgws	Object Gateway
mdss	Metadata Server
nfss	NFS Ganesha Server
restapis	RESTlike Administration Server
rbdmirrors	rbd-mirror Daemon
clients	Client
iscsigws	iSCSI Gateway
mgrs	Manager Daemon

Reihe Parameter an, die aber für eine initiale Installation bei Weitem nicht alle notwendig sind. Listing 3 zeigt die wichtigsten Parameter in der Datei *group_vars/all.yml*. In der Datei *all.yml.sample*, sind alle Parameter aufgeführt und erläutert. Sie ist auch unter github.com/ceph/cephansible/blob/master/group_vars/all.yml.sample zu finden.

Quellenangabe nicht vergessen

Damit ceph-ansible erfolgreich läuft, ist zuerst die Quelle der Pakete und damit auch die Release festzulegen. Der Parameter *ceph_origin* steuert, woher Ansible die Pakete holt. Derzeit sind drei Werte möglich: *local* veranlasst Ansible, die Binaries vom Jumphost zu kopieren. Bei Angabe von *distro* bindet es ebenfalls keine zusätzlichen Quellen ein, sondern verwendet ausschließlich die Ceph-Pakete der gegebenen Distribution. Dabei muss sichergestellt sein, dass sich Ceph-Pakete in den aktivierten Quellen befinden.

Dagegen gibt der Wert *repository* vor, dass ein zusätzliches Repository für die Pakete zu konfigurieren ist. Die entsprechende Quelle setzt der Parameter *ceph_repository*: Bei Angabe von *community* greift Ansible direkt auf die Ceph-Site zu und verwendet die Pakete, die es unter download.ceph.com findet.

Lautet die Vorgabe *rhcs*, nutzt es die Pakete aus dem Red Hat Ceph Storage. Dafür ist eine Subskription erforderlich, das heißt, die Systeme müssen bei Red Hat registriert sein und die Repositories für Red Hat Ceph Storage manuell hinzugefügt werden. Bei *dev* installiert Ansible die Pakete direkt aus dem CI-System (Continuous Integration), bei *uca* dient das Ubuntu Cloud Archive als Quelle, bei *custom* muss die Angabe eines Repository erfolgen, aus dem die Ceph-Pakete installiert werden sollen.

Wie die an Ansible zu übergeben ist, zeigt die Datei *all.yml.sample*. Dort finden sich auch die Parameter für die Quellenangabe, die durch die Werte *rhcs* und *uca* notwendig sind.

Zudem ist über den Parameter *ceph_stable_release* die zu nutzende Ceph-Release anzugeben. Die aktuelle Ceph-Release 13, *mimic*, findet zwar schon Unterstützung, für die Beispielinstallation kam aber die LTS-Release 12, *luminous*, zum Einsatz.

Im nächsten Schritt ist das *public_network*, also das Netz für die Clientanbindung, zu definieren. Dieses sollte auf allen Servern konfiguriert sein. Ist nur ein Netz vorgesehen, genügt der Parameter *public_network*. Sollen die Ceph-Knoten ihre Daten über ein separates Netz replizieren, ist der zusätzliche Parameter *cluster_network* notwendig.

Bühnenzugänge ausweisen

Zu guter Letzt sind die Netzwerkinterfaces der Hosts zu definieren, über die die Clients sie erreichen. Das kann über mehrere Wege erfolgen: *monitor_interface* legt die Netzwerkschnittstelle fest, etwa *eth1*. Da sie für alle Monitor-Hosts gilt, muss das Interface auf jedem Monitor-Host entsprechend konfiguriert sein. *monitor_address_block* definiert das Subnet für das Clientnetz, etwa *192.168.122.0/24*.

Die IP-Adressen der einzelnen Clusterknoten im Clientnetz definiert man über den Parameter *monitor_address* in den einzelnen *yml*-Dateien im Verzeichnis *host_vars*:

```
# cat host_vars/ceph-mon0.yml
monitor_address: 10.20.30.40
[...]
# cat host_vars/ceph-mon1.yml
monitor_address: 10.20.30.41
[...]
# cat host_vars/ceph-mon2.yml
monitor_address: 10.20.30.42
[...]
```

Dies ist auch die Empfehlung der Autoren, aber eher eine persönliche Vorliebe als eine Vorgabe. *monitor_address_block* funktioniert genauso gut.

Ist der Jumphost in der Gruppe *clients* eingetragen, sollte dort auch der Administrator-Key installiert werden. Dies steuert eine Zeile in der Datei *clients.yml* im Verzeichnis *group_vars*:

```
copy_admin_key: true
```

Für den nächsten Schritt sind die Blöcke *osds*, *rgws* und *nfss* im Inventory auszukommentieren, da es sonst zu Fehlern

Listing 5: *group_vars/osds.yml*

```
osd_objectstore: bluestore

osd_scenario: non-collocated

devices:
  - /dev/sda
  - /dev/sdb
  - /dev/sdc
  - /dev/sdd
  - /dev/sde
  - /dev/sdf

dedicated_devices:
  - /dev/nvme0n1
  - /dev/nvme0n1
  - /dev/nvme0n1
  - /dev/nvme0n1
  - /dev/nvme0n1
  - /dev/nvme0n1

bluestore_wal_devices:
  - /dev/nvme1n1
  - /dev/nvme1n1
  - /dev/nvme1n1
  - /dev/nvme1n1
  - /dev/nvme1n1
  - /dev/nvme1n1
```

Listing 6: *host_vars/ceph-osd0.yml*

```
osd_objectstore: bluestore

osd_scenario: non-collocated

devices:
  - /dev/vdb
  - /dev/vdc

dedicated_devices:
  - /dev/vdd
  - /dev/vdd
```

Listing 7: Ausgabe des Befehls *ceph-disk list*

```
[...]
/dev/vdb :
  /dev/vdb1 ceph data, active, cluster ceph, osd.1, block /dev/vdb2, block.db /dev/vdd1,
block.wal /dev/vdd2
  /dev/vdb2 ceph block, for /dev/vdb1
/dev/vdc :
  /dev/vdc1 ceph data, active, cluster ceph, osd.9, block /dev/vdc2, block.db /dev/vdd3,
block.wal /dev/vdd4
  /dev/vdc2 ceph block, for /dev/vdc1
/dev/vdd :
  /dev/vdd1 ceph block.db, for /dev/vdb1
  /dev/vdd2 ceph block.wal, for /dev/vdb1
  /dev/vdd3 ceph block.db, for /dev/vdc1
  /dev/vdd4 ceph block.wal, for /dev/vdc1
```

kommen kann. ceph-ansible bringt ein Beispiel-Playbook *site.yml.sample* mit, das ideal für den Einstieg ist:

```
ansible-playbook -i inventory ../ ⤷
                                site.yml.sample
```

Nach dem erfolgreichen Durchlauf von Ansible sollte es möglich sein, sich vom Jumphost aus mit dem Ceph-Cluster zu verbinden – sofern dieser als Client eingetragen ist. Listing 4 zeigt die Ausgabe des Befehls *ceph status*. Die Warnung des Clusters

```
pgs:    100.000% pgs unknown
```

ist korrekt, da noch die OSDs fehlen, um die Placement Groups (PGS) auch zu verteilen.

Den Speicher in Szene setzen

ceph-ansible erlaubt nicht nur das Installieren der OSDs auf mehreren Wegen, sondern auch das Verteilen der Datenbanken *block.db* und der Write-ahead-Logs *block.wal*. Zu hinterlegen sind die Angaben dazu in der Datei *osds.yml* im Verzeichnis *groups_vars*. Alternativ kann man für jeden Host eine eigene Konfiguration im Verzeichnis *host_vars* anlegen. Da in der Beispielinstallation alle Hosts gleich konfiguriert sind, genügen es hier, alle Devices und Parameter über *osds.yml* zu steuern.

Listing 8: Ausgabe des Befehls *ceph -s*

```
 cluster:
    id:     981a9ba1-a1ff-4a82-89ef-5e88849418e6
    health: HEALTH_WARN
            too few PGs per OSD (4 < min 30)

  services:
    mon: 3 daemons, quorum ceph-mon0,ceph-mon1,ceph-mon2
    mgr: ceph-mon0(active), standbys: ceph-mon1, ceph-mon2
    mds: cephfs-1/1/1 up  {0=ceph-mon1=up:active}, 2 up:standby
    osd: 10 osds: 10 up, 10 in

  data:
    pools:   2 pools, 16 pgs
    objects: 21 objects, 2246 bytes
    usage:   10284 MB used, 188 GB / 199 GB avail
    pgs:     16 active+clean
```

Die erste Einstellung ist das Format des Storage-Backend: Der Default bei ceph-ansible ist noch immer *filestore*. Für das neue Format *bluestore* setzt man *osd_objectstore: bluestore*. Grundsätzlich empfiehlt es sich, wichtige Parameter, auch wenn sie Default-Werte enthalten, in die Konfiguration aufzunehmen. Dann muss man später nicht raten, welches der Default-Wert war.

Es folgt die Aufteilung der Datenplatten, die der Parameter *osd_scenario* steuert. Auch hier stehen mehrere Varianten zu Verfügung: *collocated* bedeutet, dass Daten, DB- und WAL-Bereiche (Write-ahead-Log) zusammen auf die gleiche Festplatte gelegt werden. Sind keine separaten SSDs konfiguriert, können alle Bereiche auf die gleiche Festplatte. Dazu muss nur der Parameter *devices* definiert sein.

Bei *non-collocated* legt Ansible Daten und DB- beziehungsweise WAL-Bereiche getrennt ab. Neben *devices*

Listing 9: *group_vars/all.yml*

```
[...]

ceph_conf_overwrites:
    global:
        osd pool default pg num: 64
        osd pool default pgp num: 64
    mon:
        mon allow pool delete: false
    osd:
        osd crush update on start: false
```

Listing 10: *apply-config.yml*

```
- hosts: all
  gather_facts: true
  become: True
  roles:
    - ceph-defaults
    - ceph-config
```

muss dann die Variable *dedicated_devices* definiert sein. Zusätzlich können über den Parameter *bluestore_wal_devices* auch gesonderte Festplatten für das WAL angegeben werden. Dieses sollte nur notwendig sein, wenn auch wirklich schnellere Festplatten für diese Aufgabe vorhanden sind. Mit dem Wert *lvm* steht dem Administrator die relativ neue Option zur Verfügung, den LVM für die entsprechenden Bereiche zu nutzen. Hier muss zudem *lvm_volumes* definiert sein.

Den Akteuren die Plätze zuweisen

Konkret könnte eine Konfiguration in der Datei *group_vars/osds.yml* nun aussehen wie in Listing 5. In dem Beispiel liegen alle Datenbereiche auf den Festplatten */dev/sdX*, *block.db* auf dem ersten NVMe-Device (Non-volatile Memory Express) und *block.wal* auf dem zweiten. Zu beachten ist, dass die Größe des DB-Bereichs 1 GByte beträgt und die des WAL-Bereichs 576 MByte. Beide lassen sich derzeit nicht ändern.

Obwohl sich die Speichermedien auch in einer gemeinsamen Datei im Verzeichnis *group_vars* festlegen lassen, empfiehlt es sich in einer Langzeitinstallation, dafür einzelne Host-Dateien in *host_vars* zu verwenden, da man dadurch bei späteren Anpassungen flexibler ist (siehe Listing 6). Das Beispiel sieht BlueStore vor, aber keine separaten WAL-Devices.

Sind die Parameter definiert, ist noch der Block *osds* in der zentralen Datei *inventory* auszukommentieren. Danach kann das Playbook *site.yml* wieder gestartet werden. Mit der aufgeführten Konfiguration und einem erfolgreichen Ansible-Run sieht die Konfiguration auf einem Knoten wie in Listing 7 aus. Danach sollte nun auch der Ceph-Cluster nur noch Meldungen über zu wenige PGS ausgeben, wie er es in Listing 8 tut. Damit ist die initiale Installation abgeschlossen.

Das ceph-ansible-Team nimmt keine Vorschläge zu sinnvollen Default-Einstellungen entgegen, sondern beschränkt die Vorgaben auf eine minimale Konfiguration der *ceph.conf*, außer es gibt berechtigte Gründe dafür. Damit aber trotzdem weitere Parameter über ceph-ansible verwaltet werden können, gibt es den Parameter *ceph_conf_overwrites*. Er kann im YAML-Format mit allgemeinen Sektionen umgehen, die wiederum eigene Parameter aufnehmen können. Ein Beispiel zeigt Listing 9.

Spielanweisungen nachträglich ändern

Sollen nur Konfigurationsparameter angepasst werden, genügt es, die Rollen *ceph-defaults* und *ceph-config* aufzurufen (siehe Listing 10). Dabei werden aber keine Dienste neu gestartet.

Zusätzliche Dienste und Parameter lassen sich jederzeit aktivieren. Damit die Änderungen wirksam werden, ist nur das Playbook neu zu starten. Allerdings gibt es an dieser Stelle einige Einschränkungen: Verwaltet man auch Pools und Keys mit ceph-ansible, sind Änderungen im Nachhinein nicht mehr möglich. Selbiges gilt für die CRUSH Map und Rules, sofern man sie ebenfalls damit verwaltet.

Für Updates genügt es nicht, einfach den Parameter *ceph_stable_release* zu ändern. Das bewirkt nur, dass die entsprechenden Repositories aktiviert werden, aber kein Update der Pakete. Hier hilft ein Blick in das Verzeichnis *infrastructure-playbooks*. Dort finden sich unterschiedliche Playbooks zum Verwalten und Pflegen eines Ceph-Clusters. Darunter ist auch ein Playbook *rolling_update.yml*, mit dem sich ganze Cluster aktualisieren lassen. In der Regel genügt es, das Playbook ins Hauptverzeichnis zu kopieren und zu starten.

Wer Interesse an ceph-ansible bekommen, seinen Cluster früher aber per Hand oder mit *ceph-deploy* installiert hat, findet im Verzeichnis *infrastructure-playbooks* ebenfalls ein passendes Playbook mit Namen *take-over-existing-cluster.yml*. Es verbindet sich mit dem vorhandenen Ceph-Cluster, sammelt alle Keys ein und erstellt damit eine lokale *ceph.conf*. Voraussetzung dafür ist aber, dass die Dateien in den Verzeichnissen *group_vars* und *host_vars* entsprechend konfiguriert sind – hier ist also Obacht geboten.

Fazit

Ceph findet als Alternative zu Enterprise-Storage-Systemen immer mehr Verbreitung. Um den Nachteil des höheren Aufwands bei Einrichtung und Pflege – Schattenseite der großen Flexibilität – zu kompensieren, hat Inktank-Käufer Red Hat ceph-ansible entwickelt.

Diese Ansible Playbooks for Ceph enthalten fertige Regieanweisungen für das Continuous-Integration-Werkzeug Ansible, mit denen sich ein Ceph-Cluster bereitstellen und anschließend verwalten, erweitern, ändern und aktualisieren lässt. Darüber hinaus existieren Playbooks zum Übernehmen vorhandener Cluster in das Ansible-Management. Trotz seiner Jugend wirkt ceph-ansible von kleinen Kinderkrankheiten abgesehen dabei schon recht ausgereift. (sun@ix.de)

Michel Raabe

arbeitet bei der B1 Systems GmbH als Consultant und Trainer. Seine Schwerpunkte liegen bei Clustern, Hochverfügbarkeit und Virtualisierung.

Christian Berendt

arbeitet als Solution Architect für die B1 Systems GmbH und betreut derzeit den Bereich „Cloud Computing“, beschäftigt sich aber auch intensiv mit Systemmanagement.

Literatur

[1] Martin Gerhard Loschwitz; Cluster-Dateisysteme; Krakenpflege; Tutorial, Teil 2: Ceph einrichten und konfigurieren; *iX* 8/2014, S. 124

[2] Martin Gerhard Loschwitz; Storage-Verwaltung; Tiefseetaucher; Mehr Komfort fürs Ceph-Deployment mit DeepSea; *iX* 1/2018, S. 126

[3] Victor Volle; Automatisierungs-Tools; Werkzeugkiste; Chef, Puppet und Ansible: Im Dienst der Continuous Integration; *iX* Developer 2016 – Effektiver entwickeln, S. 70

[4] Martin Gerhard Loschwitz; Speichertechnik; Massiv beschleunigt; Neues Backend bei Ceph; *iX* 10/2017, S. 120

Alle Links: ix.de/ix1812120