



Red Hat OpenStack Platform 14

Monitoring Tools Configuration Guide

A guide to OpenStack logging and monitoring tools

Red Hat OpenStack Platform 14 Monitoring Tools Configuration Guide

A guide to OpenStack logging and monitoring tools

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Copyright © 2019 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This guide provides information on configuring logging and monitoring for a Red Hat OpenStack Platform environment.

Table of Contents

CHAPTER 1. INTRODUCTION	3
CHAPTER 2. ARCHITECTURE	4
2.1. CENTRALIZED LOGGING	4
2.2. AVAILABILITY MONITORING	7
CHAPTER 3. INSTALL THE CLIENT-SIDE TOOLS	11
3.1. SET CENTRALIZED LOGGING CLIENT PARAMETERS	11
3.2. SET AVAILABILITY MONITORING CLIENT PARAMETERS	11
3.3. INSTALL OPERATIONAL TOOLS ON OVERCLOUD NODES	12
3.4. FILTER AND TRANSFORM LOGGING DATA	12
CHAPTER 4. INSTALL THE SERVER-SIDE COMPONENTS	15
CHAPTER 5. MONITOR THE OPENSTACK PLATFORM	16
CHAPTER 6. VALIDATE THE SENSU CLIENT INSTALLATION	17
CHAPTER 7. REVIEW THE STATE OF A NODE	18
CHAPTER 8. REVIEW THE STATE OF AN OPENSTACK SERVICE	19

CHAPTER 1. INTRODUCTION

Monitoring tools are an optional suite of tools designed to help operators maintain an OpenStack environment. The tools perform the following functions:

- **Centralized logging:** Allows you gather logs from all components in the OpenStack environment in one central location. You can identify problems across all nodes and services, and optionally, export the log data to Red Hat for assistance in diagnosing problems.
- **Availability monitoring:** Allows you to monitor all components in the OpenStack environment and determine if any components are currently experiencing outages or are otherwise not functional. You can also configure the system to alert you when problems are identified.

CHAPTER 2. ARCHITECTURE

Monitoring tools use a client-server model with the client deployed onto the Red Hat OpenStack Platform overcloud nodes. The Fluentd service provides client-side centralized logging (CL) and the Sensu client service provides client-side availability monitoring (AM).

2.1. CENTRALIZED LOGGING

Centralized logging allows you to have one central place to view logs across your entire OpenStack environment. These logs come from the operating system, such as syslog and audit log files, infrastructure components such as RabbitMQ and MariaDB, and OpenStack services such as Identity, Compute, and others.

The centralized logging toolchain consists of a number of components, including:

- Log Collection Agent (Fluentd)
- Log Relay/Transformer (Fluentd)
- Data Store (Elasticsearch)
- API/Presentation Layer (Kibana)



NOTE

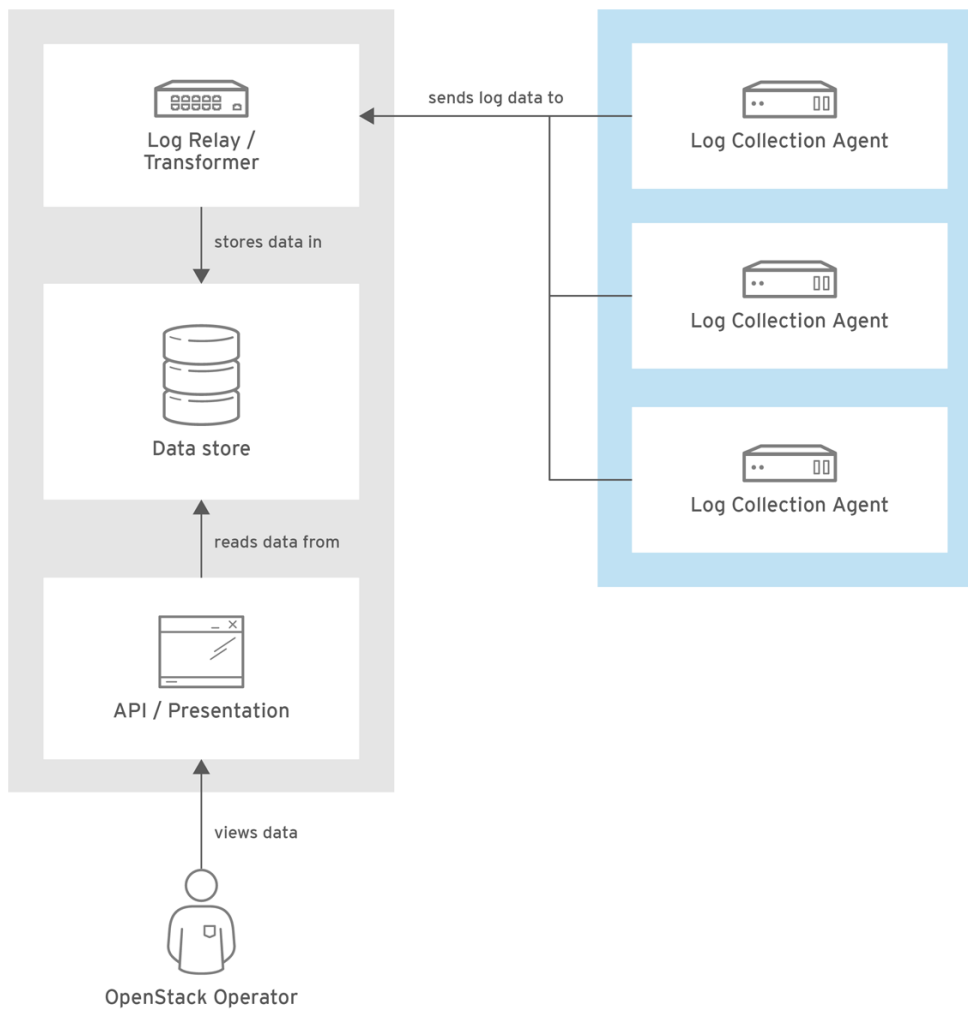
The director does not deploy the server-side components for centralized logging. Red Hat does not support the server-side components, including the Elasticsearch database, Kibana, and Fluentd with plugins running as a log aggregator.

The centralized logging components and their interactions are laid out in the following diagrams:



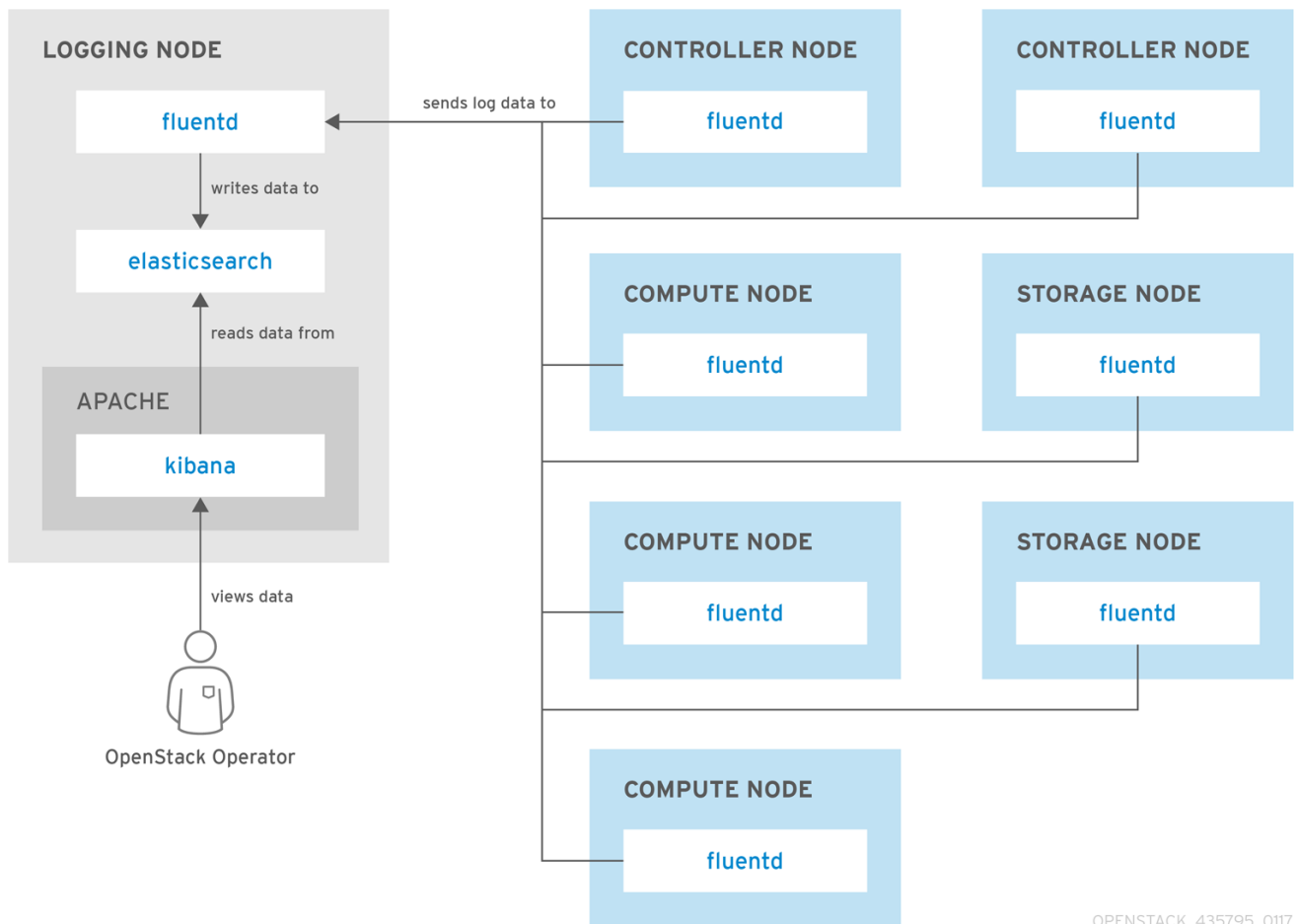
NOTE

Items shown in blue denote Red Hat-supported components.

Figure 2.1. Centralized logging architecture at a high level

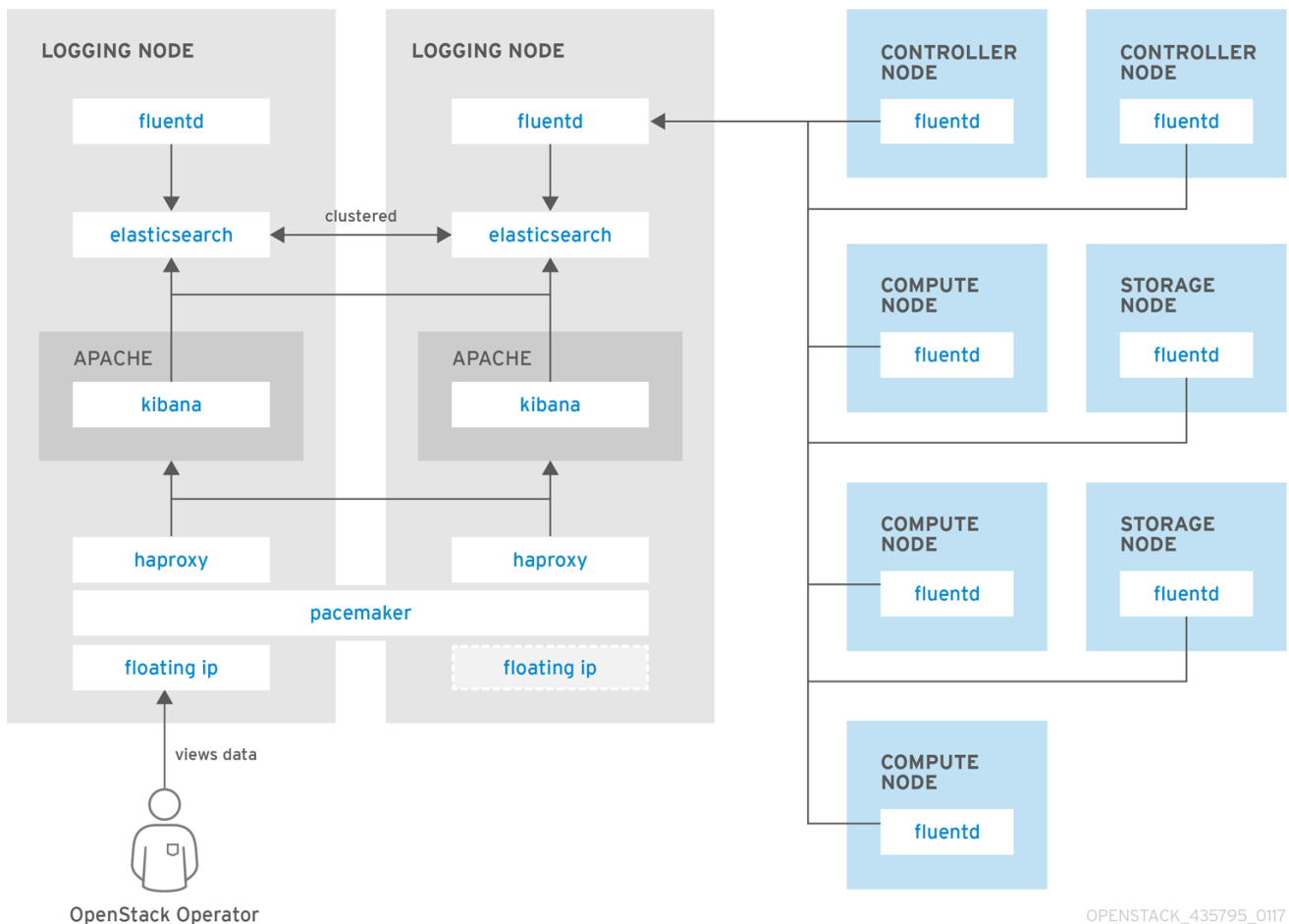
OPENSTACK_435795_017

Figure 2.2. Single-node deployment for Red Hat OpenStack Platform



OPENSTACK_435795_017

Figure 2.3. HA deployment for Red Hat OpenStack Platform



2.2. AVAILABILITY MONITORING

Availability monitoring allows you to have one central place to monitor the high-level functionality of all components across your entire OpenStack environment.

The availability monitoring toolchain consists of a number of components, including:

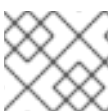
- Monitoring Agent (Sensu client)
- Monitoring Relay/Proxy (RabbitMQ)
- Monitoring Controller/Server (Sensu server)
- API/Presentation Layer (Uchiwa)



NOTE

The director does not deploy the server-side components for availability monitoring. Red Hat does not support the server-side components, including Uchiwa, Sensu Server, the Sensu API plus RabbitMQ, and a Redis instance running on a monitoring node.

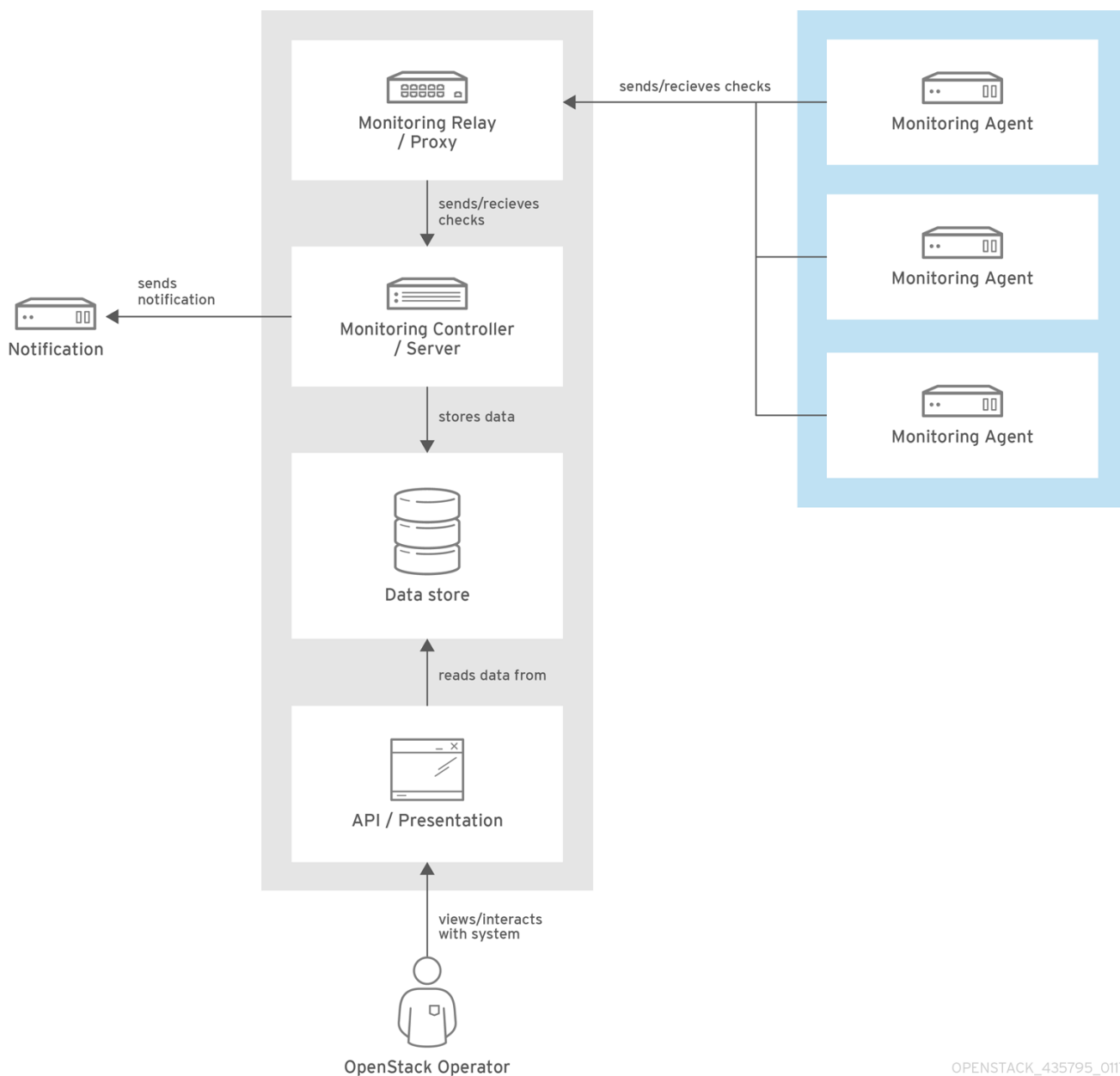
The availability monitoring components and their interactions are laid out in the following diagrams:



NOTE

Items shown in blue denote Red Hat-supported components.

Figure 2.4. Availability monitoring architecture at a high level



OPENSTACK_435795_017

Figure 2.5. Single-node deployment for Red Hat OpenStack Platform

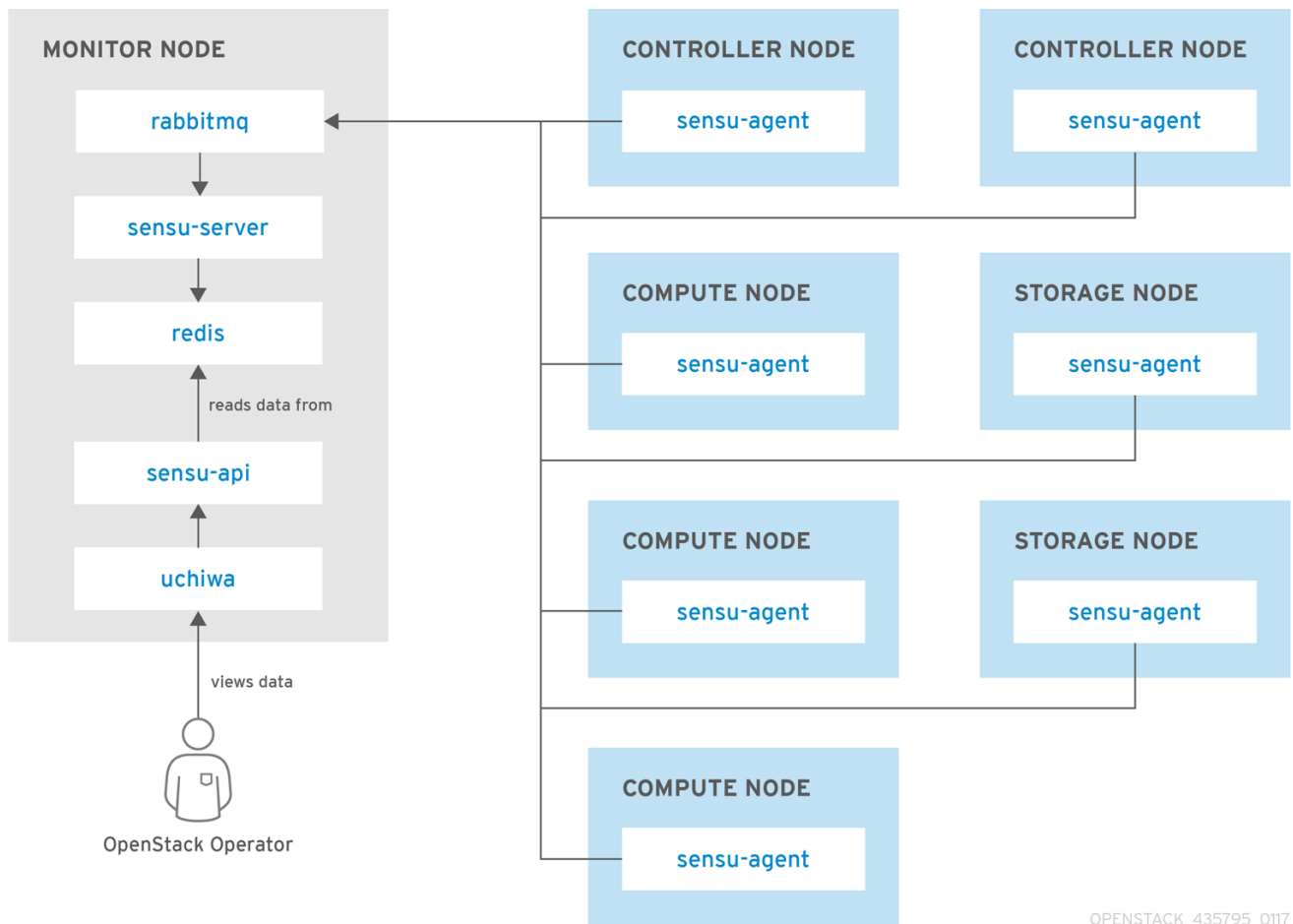
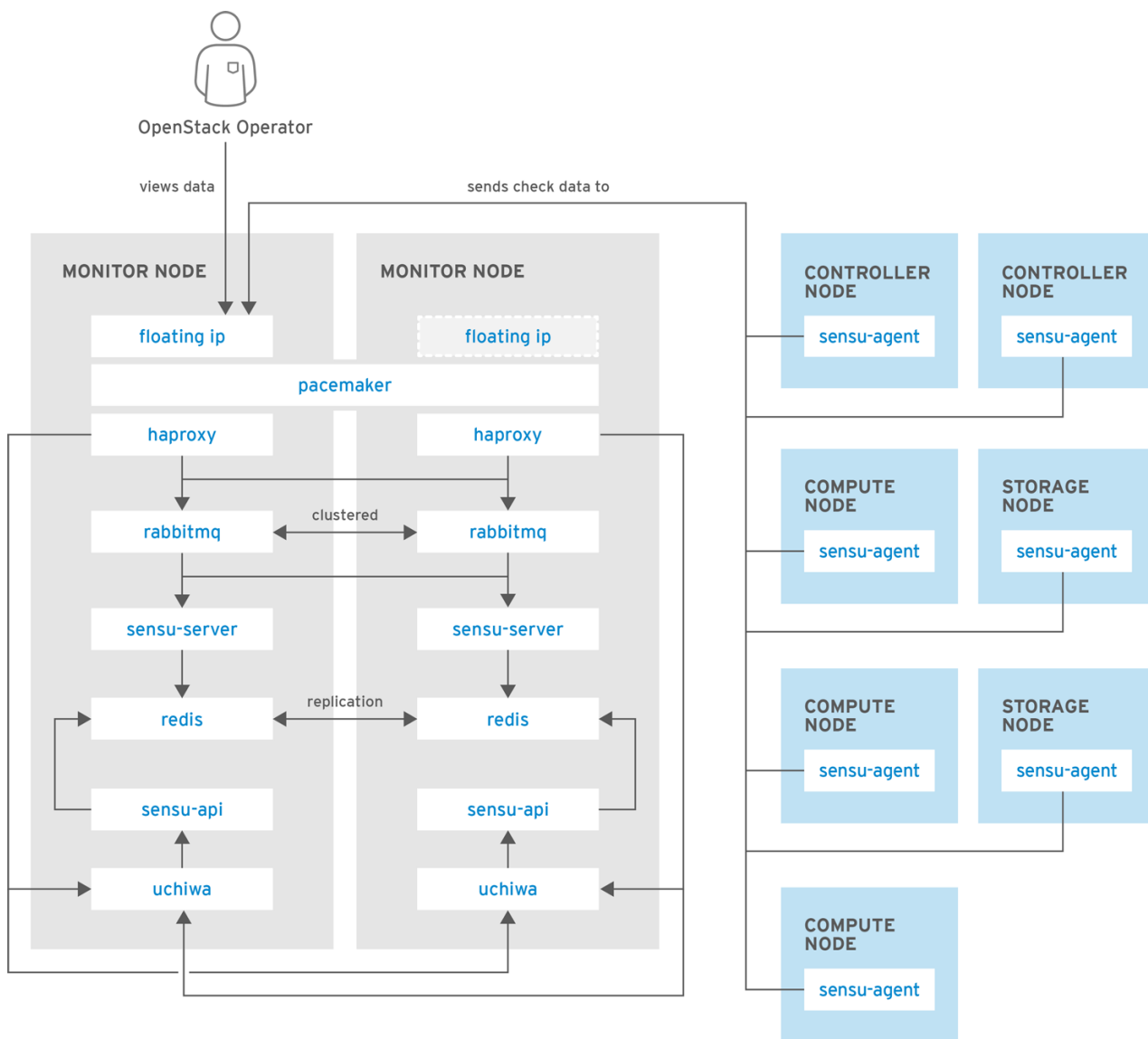


Figure 2.6. HA deployment for Red Hat OpenStack Platform



OPENSTACK_435795_017

CHAPTER 3. INSTALL THE CLIENT-SIDE TOOLS

Before you deploy the overcloud, you need to determine the configuration settings to apply to each client. Copy the example environment files from the director's Heat template collection and modify them to suit your environment.

3.1. SET CENTRALIZED LOGGING CLIENT PARAMETERS

For Fluentd configuration settings, copy `/usr/share/openstack-tripleo-heat-templates/environments/logging-environment.yaml` and modify the file to suit your environment. For example:

Simple configuration

```
resource_registry:
  OS::TripleO::Services::Fluentd: ../docker/services/fluentd.yaml

parameter_defaults:
  LoggingServers:
    - host: log0.example.com
      port: 24224
    - host: log1.example.com
      port: 24224
```

Example SSL configuration

```
## (note the use of port 24284 for ssl connections)

resource_registry:
  OS::TripleO::Services::Fluentd: ../docker/services/fluentd.yaml

parameter_defaults:
  LoggingServers:
    - host: 192.0.2.11
      port: 24284
  LoggingUsesSSL: true
  LoggingSharedKey: secret
  LoggingSSLCertificate: |
    -----BEGIN CERTIFICATE-----
    ...certificate data here...
    -----END CERTIFICATE-----
```

- **LoggingServers** - The destination system that will receive Fluentd log messages.
- **LoggingUsesSSL** - Setting that determines whether `secure_forward` is used when forwarding log messages.
- **LoggingSharedKey** - The shared secret used by `secure_forward`.
- **LoggingSSLCertificate** - The PEM-encoded contents of the SSL CA certificate.

3.2. SET AVAILABILITY MONITORING CLIENT PARAMETERS

For the Sensu client configuration settings, copy `/usr/share/openstack-tripleo-heat-templates/environments/monitoring-environment.yaml` and modify the file to suit your environment. For example:

```
resource_registry:
  OS::TripleO::Services::SensuClient: ../docker/services/sensu-client.yaml

parameter_defaults:
  MonitoringRabbitHost: 10.10.10.10
  MonitoringRabbitPort: 5672
  MonitoringRabbitUserName: sensu
  MonitoringRabbitPassword: sensu
  MonitoringRabbitUseSSL: true
  MonitoringRabbitVhost: "/sensu"
  SensuClientCustomConfig:
    api:
      warning: 10
      critical: 20
```

- **MonitoringRabbit** - These parameters connect the Sensu client services to the RabbitMQ instance that runs on the monitoring server.
- **MonitoringRabbitUseSSL** - Enables SSL for the RabbitMQ client. Uses SSL transport if the private key or certificate chain are not specified, as below.
- **MonitoringRabbitSSLPrivateKey** - Defines the path to the private key file, or can contain the contents of that file.
- **MonitoringRabbitSSLCertChain** - Defines the private SSL certificate chain to use.
- **SensuClientCustomConfig** - Specify additional Sensu client configuration. Defines the OpenStack credentials to be used, including username/password, `auth_url`, tenant, and region.

3.3. INSTALL OPERATIONAL TOOLS ON OVERCLOUD NODES

Include the modified YAML files with your `openstack overcloud deploy` command to install the Sensu client and Fluentd tools on all overcloud nodes. For example:

```
$ openstack overcloud deploy --templates -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml -e network-environment.yaml -e ~/templates/monitoring-environment.yaml -e ~/templates/logging-environment.yaml --control-scale 3 --compute-scale 1 -ntp-server 192.168.122.10
```

3.4. FILTER AND TRANSFORM LOGGING DATA

You can filter and transform events sent to Fluentd by setting the `LoggingDefaultFilters` parameter in your environment file. For example, the `record_transformer` type can modify incoming events:

```
parameter_defaults:

  LoggingDefaultFilters:
```



```

- tag_pattern: '**'
  type: record_transformer
  enable_ruby: true
  record:
    openstack: '{"hostname": "${hostname}","tag": "${tag}","region":
"regionOne","inputname":"fluent-plugin-in_tail","name":"fluentd
openstack","fluentd version":"0.12.26", "pipeline_metadata": {"collector":
{"ipaddr4":"#{ begin Socket.ip_address_list.select { |a| a.ipv4? &&
!a.ipv4_loopback? }.map { |a| a.ip_address } end}","ipaddr6":"#{ begin
Socket.ip_address_list.select { |a| a.ipv6? && !a.ipv6_loopback? }.map {
|a| a.ip_address } end}}}'
    remove_keys: host

- tag_pattern: 'openstack.**'
  type: record_transformer
  record:
    service: '${tag_parts[1]}'

- tag_pattern: 'system.messages**'
  type: record_transformer
  enable_ruby: true
  record:
    openstack: '{"info": "${record['message']}", "systemd":{"t":
{"PID":"${record['pid']}"}, "u":
{"SYSLOG_IDENTIFIER":"${record['ident']}"}}}'
    remove_keys: 'ident,message,pid'

```

As a result, the data received by Kibana has been transformed accordingly:

```

{
  "_index": "logstash-2017.06.29",
  "_type": "fluentd",
  "_id": "AVz132QmRtyd8nnlv_11",
  "_score": null,
  "_source": {
    "pid": "22691",
    "priority": "INFO",
    "message": "cinder.api.openstack.wsgi [req-04bc2808-f86f-4443-86e6-
bfc596969937 - - - -] OPTIONS http://overcloud-controller-0.lab.local/",
    "openstack": {
      "hostname": "overcloud-controller-0",
      "tag": "openstack.cinder.api",
      "region": "regionOne",
      "inputname": "fluent-plugin-in_tail",
      "name": "fluentd openstack",
      "fluentd version": "0.12.26",
      "pipeline_metadata": {
        "collector": {
          "ipaddr4": "[\"192.168.24.14\", \"192.168.24.8\", \"10.0.0.4\",
\"10.0.0.5\", \"172.16.2.8\", \"172.16.2.4\", \"172.16.2.14\",
\"172.16.1.7\", \"172.16.1.9\", \"172.16.3.10\", \"172.16.3.11\",
\"172.16.0.14\"]",
          "ipaddr6": "[\"fe80::293:33ff:fed8:2228%eth0\",
\"fe80::293:33ff:fed8:2228%br-ex\", \"fe80::b86c:79ff:fe8f:9fb8%vlan10\",
\"fe80::4c78:6fff:feff:14fc%vlan20\",
\"fe80::9ced:dfff:fe8c:2d62%vlan30\",

```

```
\ "fe80::ecde:1bff:fe5d:e362%vlan40\",
\ "fe80::549c:51ff:feea:dfa8%vlan50\",
\ "fe80::e093:8fff:fef9:69b6%vxlan_sys_4789\" ]"
    }
  },
  "service": "cinder",
  "@timestamp": "2017-06-29T21:59:38+00:00"
},
"fields": {
  "@timestamp": [
    1498773578000
  ]
},
"sort": [
  1498773578000
]
}
```

CHAPTER 4. INSTALL THE SERVER-SIDE COMPONENTS



NOTE

Red Hat does not support the server-side components and their deployment process.

You can use the **opstools-ansible** playbook to install the server-side components onto Red Hat Enterprise Linux 7. These server-side components include availability monitoring and centralized logging services that complement the Red Hat-supported client-side components. The most tested **opstools-ansible** scenario is the deployment of server-side components onto CentOS 7. Detailed instructions can be found in the **README.md**: <https://github.com/centos-opstools/opstools-ansible>

CHAPTER 5. MONITOR THE OPENSTACK PLATFORM

See the Sensu documentation for further details about the Sensu stack infrastructure:

<https://sensuapp.org/docs/latest/overview/architecture.html>

Red Hat supplies a set of check scripts in the **osops-tools-monitoring-oschecks** package. The majority of the check scripts only check the API connection to the OpenStack component. However, certain scripts also perform additional OpenStack resource tests for OpenStack Compute (nova), OpenStack Block Storage (cinder), OpenStack Image (glance), and OpenStack Networking (neutron). For example, the OpenStack Identity (keystone) API check gives the following result when **keystone** is running:

```
OK: Got a token, Keystone API is working.
```

CHAPTER 6. VALIDATE THE SENSU CLIENT INSTALLATION

1. Check the status of the **sensu-client** on each overcloud node:

```
# systemctl status sensu-client
```

2. Review the error log for any issues: **/var/log/sensu/sensu-client.log**
3. Verify that each overcloud node has the **/etc/sensu/conf.d/rabbitmq.json** file that sets the IP address of the monitoring server.

CHAPTER 7. REVIEW THE STATE OF A NODE

If you have a deployment of the Uchiwa dashboard, you can use it with the Sensu server to review the state of your nodes:

1. Login to the Uchiwa dashboard and click the **Data Center** tab to confirm that the Data Center is operational.

```
http://<SERVER_IP_ADDRESS>:3000
```

2. Check that all overcloud nodes are in a **Connected** state.
3. At a suitable time, reboot one of the overcloud nodes and review the rebooted node's status in the Uchiwa dashboard. After the reboot completes, verify that the node successfully re-connects to the Sensu server and starts executing checks.

CHAPTER 8. REVIEW THE STATE OF AN OPENSTACK SERVICE

This example tests the monitoring of the **openstack-ceilometer-central** service.

1. Confirm that the **openstack-ceilometer-central** service is running:

```
systemctl status openstack-ceilometer-central.service
```

2. Connect to the Uchiwa dashboard and confirm that a successful **ceilometer** check is present and running as defined in the **ceilometer** JSON file.
3. Stop the **openstack-ceilometer-central** service.



NOTE

This may disrupt services, so run this test at an appropriate time.

```
systemctl stop openstack-ceilometer-central.service
```

4. Log in to the Uchiwa dashboard and confirm that the failed **ceilometer** check has been reported.
5. Start the **openstack-ceilometer-central** service:

```
systemctl start openstack-ceilometer-central.service
```

6. Log in to the Uchiwa dashboard and view the time interval between the **ceilometer** check reports to confirm that the check runs in the time interval defined in the **ceilometer** JSON file.