Red Hat OpenStack Platform 15

監視ツール設定ガイド

OpenStack のロギングおよび監視ツールについてのガイド

Last Updated: 2022-09-15

Red Hat OpenStack Platform 15 監視ツール設定ガイド

OpenStack のロギングおよび監視ツールについてのガイド

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本ガイドでは、Red Hat OpenStack Platform 環境でのロギングと監視の設定方法について説明します。

目次

第1章 はじめに

監視ツールは、オペレーターが OpenStack 環境を維持管理するのに役立つオプションのツールセットです。これらのツールは、以下の機能を果たします。

- 集中ロギング: OpenStack 環境の全コンポーネントからのログを 1 つの場所に収集することができます。すべてのノードとサービスにわたって問題を特定することができます。また、オプションで Red Hat にログデータをエクスポートして、問題を診断するサポートを受けることもできます。

- 可用性監視: OpenStack 環境内の全コンポーネントを監視して、いずれかのコンポーネントが現在停止中または機能していない状態かどうかを判断します。また、問題が特定された時にシステムが警告を送信するように設定することも可能です。

第2章 アーキテクチャー

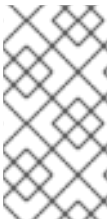
監視ツールは、クライアントが Red Hat OpenStack Platform オーバークラウドノードにデプロイされる、クライアント/サーバーモデルを使用します。Fluentd サービスがクライアント側の集中ロギング (CL) を提供し、Sensu クライアントサービスがクライアント側の可用性監視 (AM) を提供します。

2.1. 集中ロギング

集中ロギングにより、OpenStack 環境全体にわたるログを一箇所で確認することができます。これらのログは、syslog や audit ログファイルなどのオペレーティングシステム、RabbitMQ や MariaDB などのインフラストラクチャーコンポーネント、Identity や Compute などの OpenStack サービスから収集されます。

集中ロギングのツールチェーンは、以下のような複数のコンポーネントで構成されます。

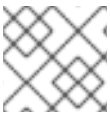
- ログ収集エージェント (Fluentd)
- ログリレー/トランスフォーマー (Fluentd)
- データストア (Elasticsearch)
- API/プレゼンテーション層 (Kibana)



注記

director は、集中ロギング向けのサーバー側のコンポーネントはデプロイしません。Red Hat では、ログアグリゲーターとして実行するプラグインを使用する Elasticsearch データベース、Kibana、Fluentd などのサーバー側のコンポーネントはサポートしていません。

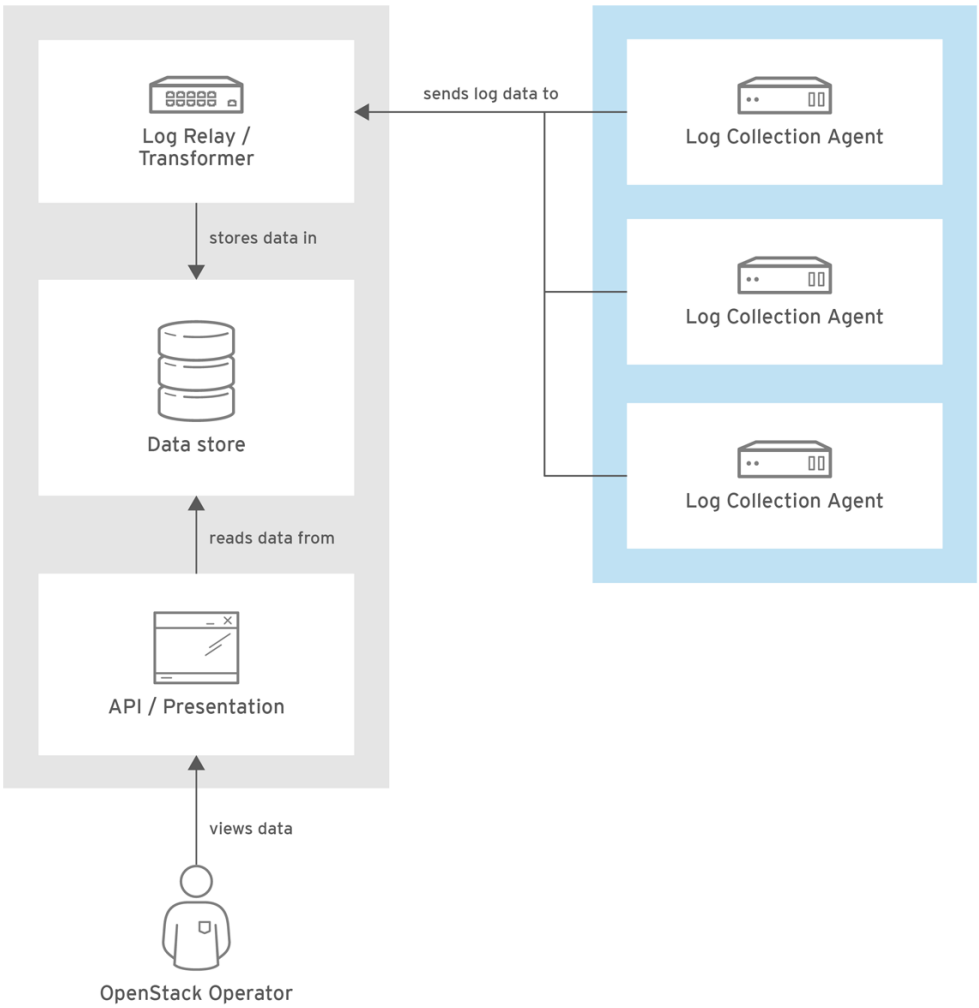
以下の図は、集中ロギングのコンポーネントとそれらの対話を示しています。



注記

青で示した項目は Red Hat のサポート対象コンポーネントです。

図2.1集中ロギングのハイレベルアーキテクチャー



OPENSTACK_435795_0117

図2.2 Red Hat OpenStack Platform の単一ノードデプロイメント

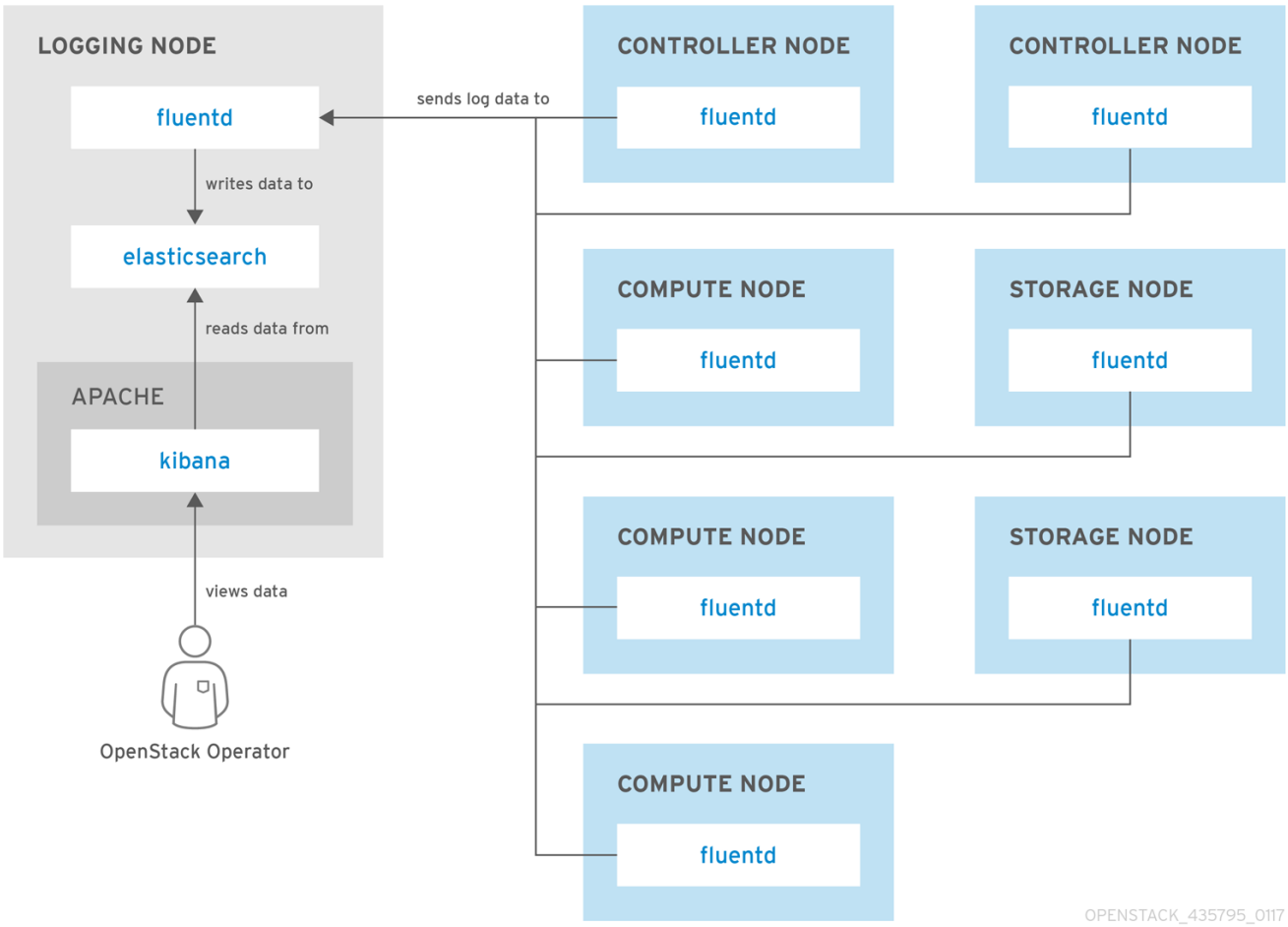
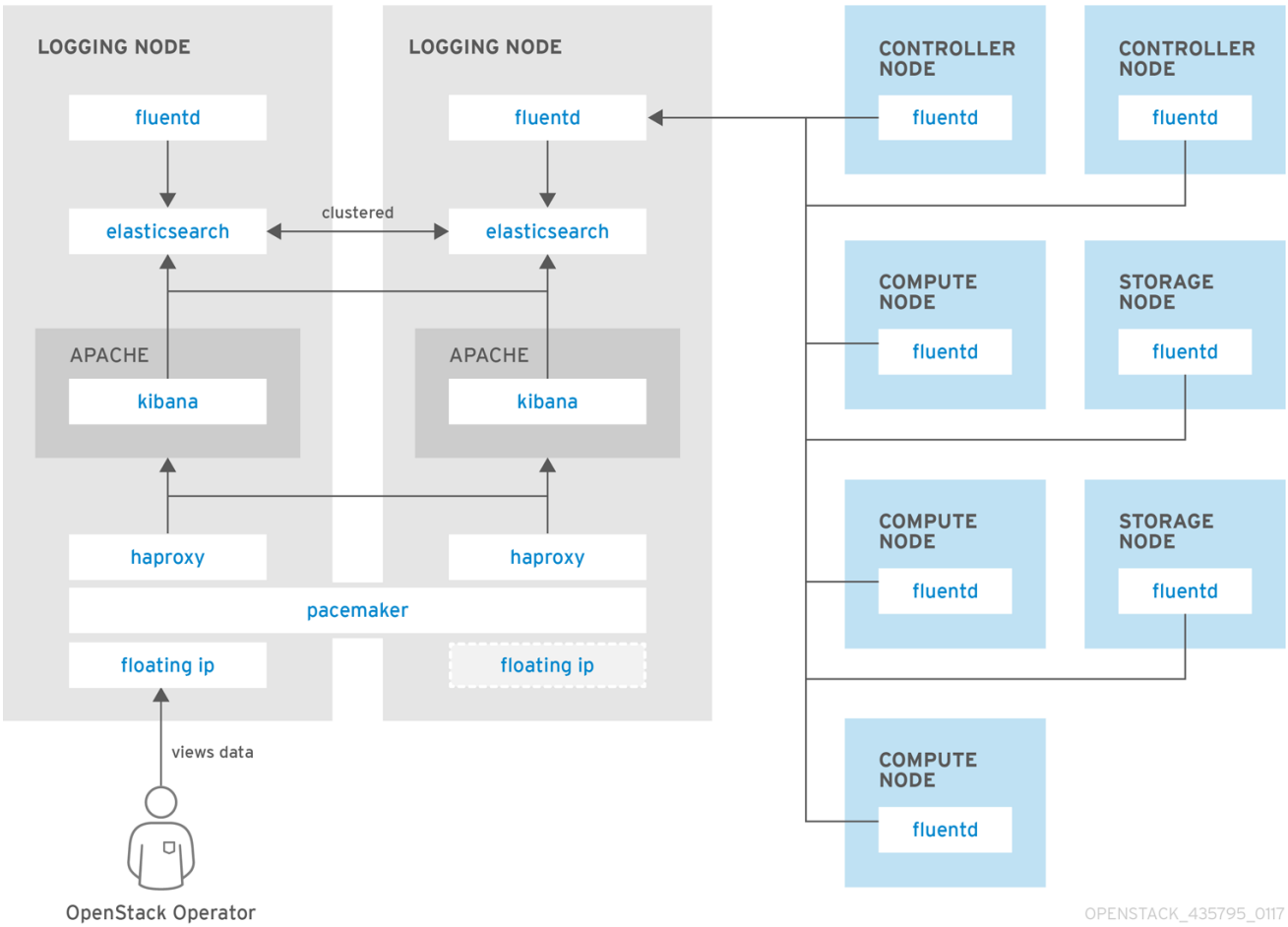


図2.3 Red Hat OpenStack Platform の HA デプロイメント



2.2. 可用性監視

可用性監視により、OpenStack 環境全体にわたる全コンポーネントのハイレベルな機能性を一元的に監視することができます。

可用性監視のツールチェーンは、以下を含む複数のコンポーネントで構成されます。

- 監視エージェント (Sensu クライアント)
- 監視リレー/プロキシー (RabbitMQ)
- 監視コントローラー/サーバー (Sensu サーバー)
- API/プレゼンテーション層 (Uchiwa)

注記

director はサーバー側の可用性監視のコンポーネントはデプロイしません。Red Hat では、Uchiwa、Sensu Server、Sensu API + RabbitMQ、監視ノードを実行する Redis インスタンスなどのサーバー側のコンポーネントはサポートしていません。

以下の図は、可用性監視のコンポーネントとそれらの対話を示しています。

注記

青で示した項目は Red Hat のサポート対象コンポーネントです。

図2.4 可用性監視のハイレベルアーキテクチャー

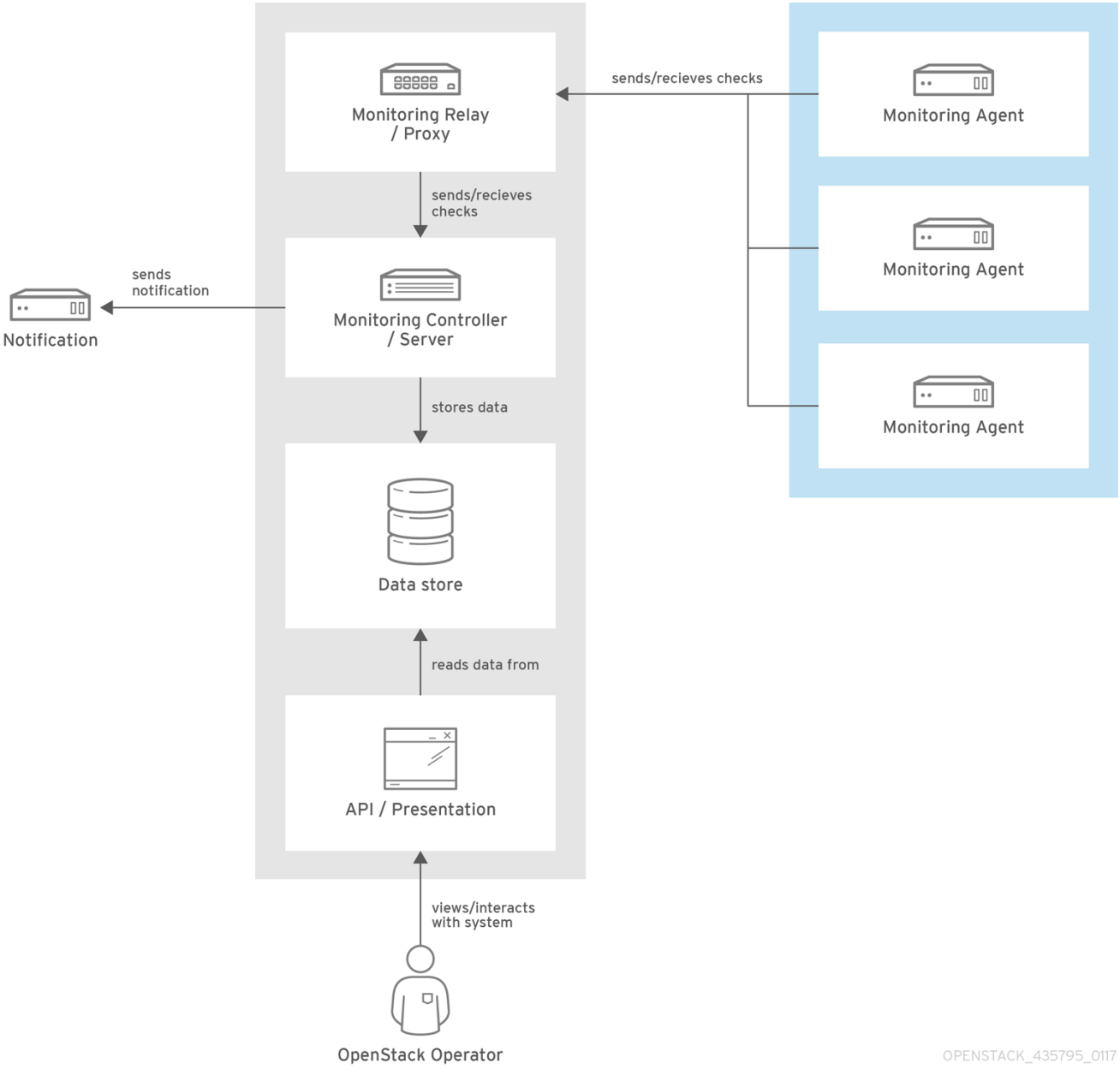


図2.5 Red Hat OpenStack Platform の単一ノードデプロイメント

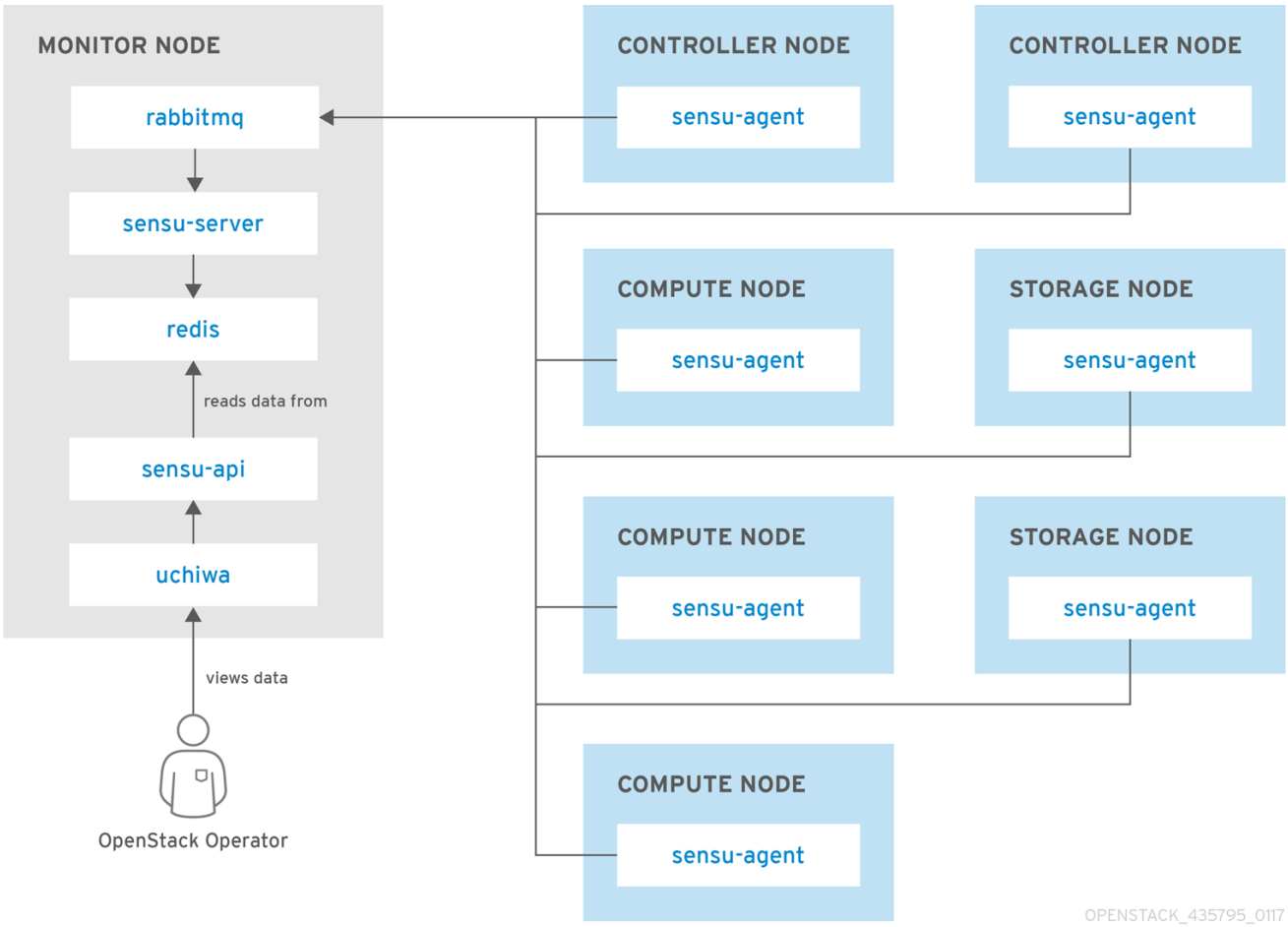
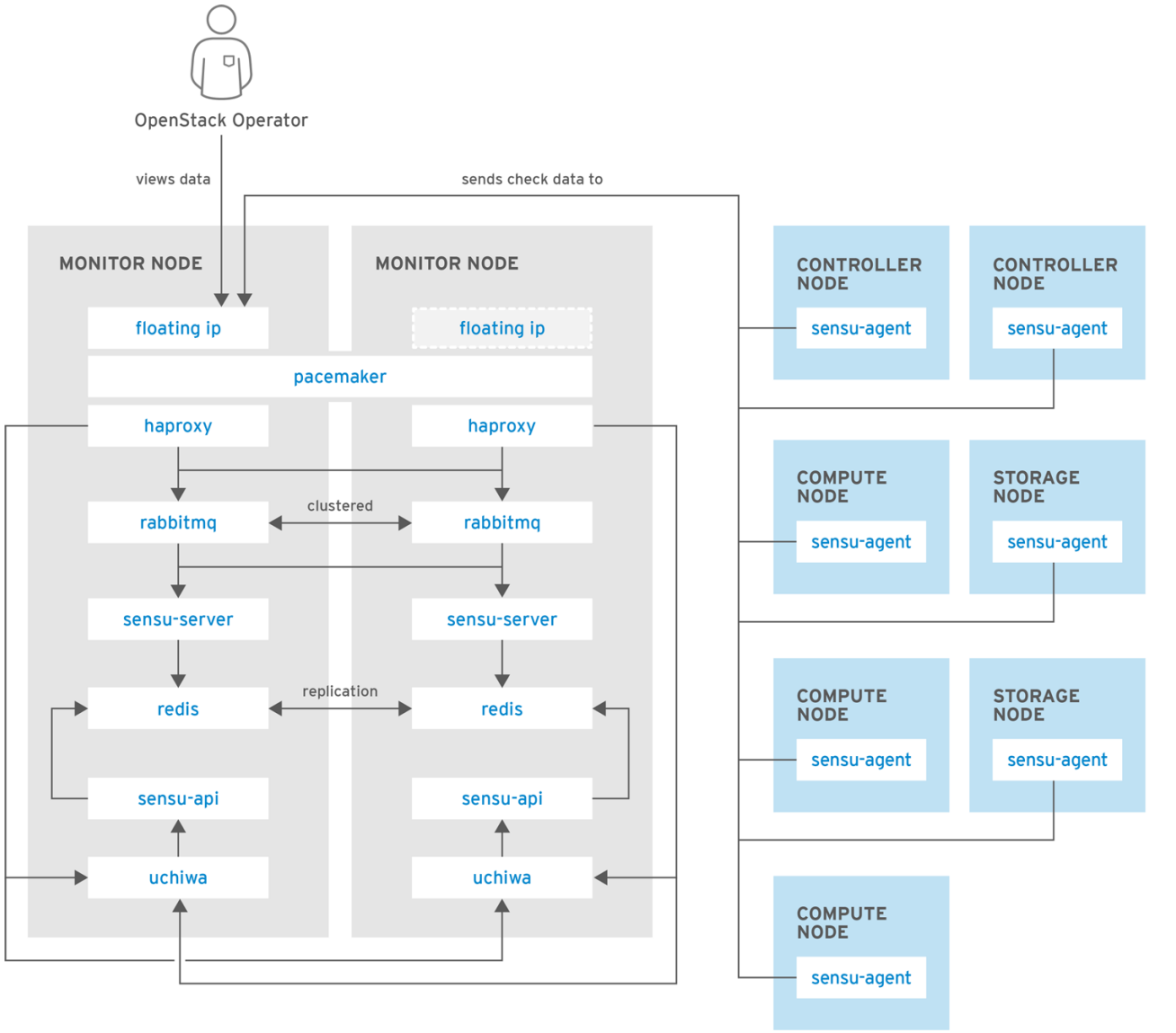


図2.6 Red Hat OpenStack Platform の HA デプロイメント



第3章 クライアント側のツールのインストール

オーバークラウドをデプロイする前には、各クライアントに適用する構成の設定を決定する必要があります。director の Heat テンプレートコレクションからサンプルの環境ファイルをコピーし、お使いの環境に合わせて変更します。

3.1. 集中ロギングクライアントパラメーターの設定

Fluentd の構成設定には、**/usr/share/openstack-tripleo-heat-templates/environments/logging-environment.yaml** をコピーし、ご使用の環境に応じてファイルを変更します。以下に例を示します。

簡易設定

```
resource_registry:
  OS::TripleO::Services::Fluentd: ../deployment/deprecated/logging/fluentd-container-puppet.yaml

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

SSL の設定例

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

resource_registry:
  OS::TripleO::Services::Fluentd: ../deployment/deprecated/logging/fluentd-container-puppet.yaml

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

- **LoggingServers**: Fluentd ログメッセージを受信する宛先システム
- **LoggingUsesSSL**: ログメッセージの転送時に **secure_forward** が使用されるかどうかを判断する設定。
- **LoggingSharedKey**: **secure_forward** が使用する共有シークレット
- **LoggingSSLCertificate**: PEM エンコードされた SSL CA 証明書の内容

3.2. 可用性監視クライアントパラメーターの設定

Sensu クライアントの構成設定には、**/usr/share/openstack-tripleo-heat-templates/environments/monitoring-environment.yaml** をコピーし、ご使用の環境に応じてファイルを変更します。以下に例を示します。

```
resource_registry:
  OS::TripleO::Services::SensuClient: ../deployment/deprecated/monitoring/sensu-client-container-puppet.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**: これらのパラメーターは、監視サーバーで実行する RabbitMQ インスタンスに Sensu クライアントサービスを接続します。
- **MonitoringRabbitUseSSL**: RabbitMQ クライアント向けに SSL を有効化します。以下のパラメーターで秘密鍵または証明書チェーンが指定されていない場合には、SSL トランスポートを使用します。
- **MonitoringRabbitSSLPrivateKey**: 秘密鍵ファイルへのパスを定義します。または、そのファイルのコンテンツを含めることができます。
- **MonitoringRabbitSSLCertChain**: 使用するプライベート SSL 証明書チェーンを定義します。
- **SensuClientCustomConfig**: Sensu クライアントの追加設定を指定します。ユーザー名/パスワード、**auth_url**、テナント、リージョンを含む OpenStack の認証情報を定義します。

3.3. オーバークラウドノードへの操作ツールのインストール

openstack overcloud deploy コマンドで変更した YAML ファイルを指定して、Sensu クライアントと Fluentd ツールを全オーバークラウドノードにインストールします。以下に例を示します。

```
$ 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. フィルターと変換ロギングデータ

環境ファイルで **LoggingDefaultFilters** パラメーターを設定することにより、Fluentd に送信されるイベントをフィルタリングして変換することができます。たとえば、**record_transformer** の種別を指定すると、受信イベントを変更することができます。

```
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'
```

その結果、Kibana が受信するデータは、設定に応じて変換されます。

```
{
  "_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
  ]
}
```

第4章 OPENSTACK PLATFORM の監視

Sensu のスタックインフラストラクチャーについては、https://docs.sensu.io/sensu-core/1.7/overview/architecture/ の Sensu のドキュメントを参照してください。

Red Hat は、**osops-tools-monitoring-oschecks** パッケージで、check スクリプトのセットを提供しています。check スクリプトの大半は、OpenStack コンポーネントへの API 接続のみをチェックします。ただし、特定のスクリプトは、OpenStack Compute (nova)、OpenStack Block Storage (cinder)、OpenStack Image (glance)、OpenStack Networking (neutron) を対象とする追加の OpenStack リソースのテストも実行します。たとえば、OpenStack Identity (keystone) API check は、**keystone** の実行時に以下の結果を返します。

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

第5章 SENSU クライアントインストールの検証

1. 各オーバークラウドノードで **sensu-client** のステータスを確認します。

```
# podman ps | grep sensu-client
```

2. エラーログ (**/var/log/containers/sensu/sensu-client.log**) で問題があるかどうかを確認します。

3. 各オーバークラウドノードに、監視サーバーの IP アドレスを設定する **/var/lib/config-data/puppet-generated/sensu/etc/sensu/conf.d/rabbitmq.json** ファイルがあることを確認します。

第6章 ノードの状態の確認

Uchiwa ダッシュボードがデプロイされている場合には、そのダッシュボードを Sensu サーバーと共に使用して、ノードの状態を確認することができます。

1. Uchiwa ダッシュボードにログインして、**Data Center** タブをクリックし、データセンターが稼働中であることを確認します。

   ```
   http://<SERVER_IP_ADDRESS>/uchiwa
   ```

2. 全オーバークラウドノードが **Connected** の状態であることを確認します。

3. オーバークラウドノードの1台を適時に再起動し、そのノードのステータスを Uchiwa ダッシュボードで確認します。再起動の完了後には、ノードが Sensu サーバーに正常に再接続されて check の実行を開始するかどうかを確認します。

第7章 OPENSTACK サービスの状態の確認

以下の例では、**openstack-ceilometer-central** サービスの監視をテストします。

1. **openstack-ceilometer-central** サービスが実行中であることを確認します。

```
docker ps -a | grep ceilometer
```

2. Uchiwa ダッシュボードに接続して、正常な **ceilometer** check があり、**ceilometer** JSON ファイルで定義されているとおりに実行されていることを確認します。