ツールおよびアプリケーションガイド

How to enable StorageGRID in your environment

NetApp
April 26, 2024

目次

ツールおよびアプリケーションガイド

StorageGRID でCloudera Hadoop S3Aコネクタを使用します

Hadoopは、しばらくの間データサイエンティストのお気に入りでした。Hadoopでは、シンプルなプログラミングフレームワークを使用して、複数のコンピュータクラスタにまたがる大規模なデータセットを分散処理できます。Hadoopは、ローカルのコンピューティングとストレージを所有するマシンごとに、単一のサーバから数千のマシンにスケールアップするように設計されています。

S3AをHadoopワークフローに使用する理由

データ量の増加に伴い、新しいマシンにコンピューティングとストレージを個別に追加するアプローチは非効率的になっています。リニアに拡張すると、リソースの効率的な使用やインフラの管理が難しくなります。

このような課題に対処するために、Hadoop S3AクライアントはS3オブジェクトストレージに対する高性能なI/Oを提供します。S3Aを使用してHadoopワークフローを実装することで、オブジェクトストレージをデータリポジトリとして活用でき、コンピューティングとストレージを分離することができます。これにより、コンピューティングとストレージを別々に拡張できます。コンピューティングリソースとストレージを分離することで、コンピューティングジョブに適切な量のリソースを割り当て、データセットのサイズに基づいて容量を提供することもできます。そのため、Hadoopワークフローの総所有コストを削減することができます。

StorageGRID を使用するようにS3Aコネクタを構成します

前提条件

- StorageGRID S3エンドポイントのURL、テナントS3アクセスキー、およびHadoop S3A接続テスト用のシークレットキー。
- クラスタ内の各ホストに対するCloudera クラスタとrootまたはsudo権限を付与して、Javaパッケージをインストールします。

2022年4月時点で、StorageGRID 11.0.14とCloudera 7.1.7のJava 11.0.14が、11.5および11.6に対してテストされました。ただし、Javaのバージョン番号は新規インストール時と異なる場合があります。

Javaパッケージをインストールします

1. を確認します "Clouderaサポートマトリックス" を参照してください。
2. をダウンロードします "Java 11.xパッケージ" Clouderaクラスタオペレーティングシステムと同じです。このパッケージをクラスタ内の各ホストにコピーします。この例では、CentOSにrpmパッケージを使用しています。
3. 各ホストにrootとしてログインするか、sudo権限を持つアカウントを使ってログインします。各ホストで次の手順を実行します。
   a. パッケージをインストールします。

```
$ sudo rpm -Uvh jdk-11.0.14_linux-x64_bin.rpm
```

b. Javaがインストールされている場所を確認します。複数のバージョンがインストールされている場合は、新しくインストールしたバージョンをデフォルトに設定します。

```
alternatives --config java

There are 2 programs which provide 'java'.

  Selection    Command
-----------------------------------------------
 +1           /usr/java/jre1.8.0_291-amd64/bin/java
  2           /usr/java/jdk-11.0.14/bin/java

Enter to keep the current selection[+], or type selection number: 2
```

c. この行を/etc/profile’の末尾に追加しますパスは、上記の選択のパスと一致する必要があります。

```
export JAVA_HOME=/usr/java/jdk-11.0.14
```

d. 次のコマンドを実行して、プロファイルを有効にします。

```
source /etc/profile
```

Cloudera HDFS S3A構成

- 手順 *

1. Cloudera Manager GUIで、クラスタ（Clusters）> HDFSを選択し、構成（Configuration）を選択します。
2. カテゴリでAdvancedを選択し、下にスクロールして「core-site .xml」用のクラスタ全体のAdvanced Configuration Snippet（Safety Valve）を探します。
3. （+）記号をクリックし、次の値ペアを追加します。

名前	価値
fs.s3a.access.key	*<tenant StorageGRID のs3アクセスキー_*
fs.s3a.secret.key	*<tenant s3 secret key from StorageGRID >*
FS.s3a.connection.ssl.enabled	[true or false]（このエントリがない場合のデフォルトはhttps）
FS.s3a.endpoint	_ StorageGRID S3エンドポイント：port>_

名前	価値
FS.s3a.impl	org.apache.hadoop .fs.s3a.S3AFileSystem
FS.s3a.path.style.access	[true or false]（このエントリがない場合のデフォルトの仮想ホスト形式）

サンプルスクリーンショット

Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml
core_site_safety_valve

HDFS (Service-Wide) Undo

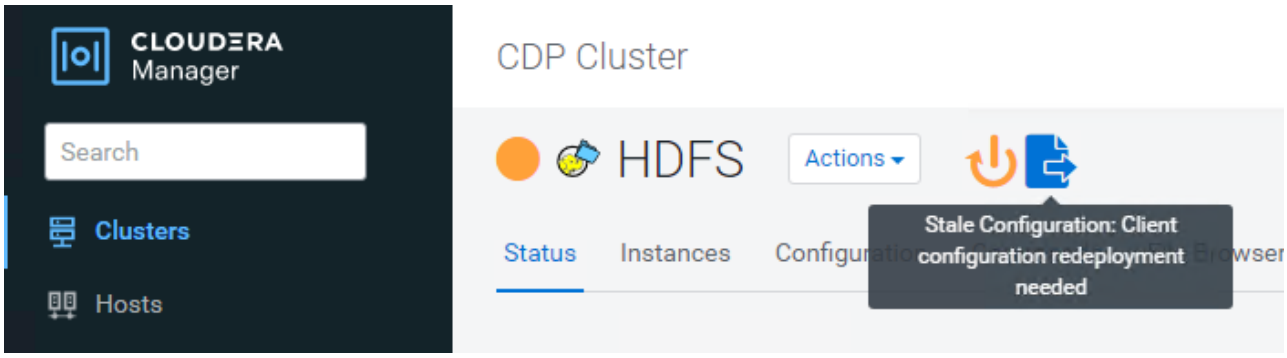
View as XML

Name	fs.s3a.endpoint
Value	sgdemo.netapp.com:10443
Description	StorageGRID s3 load balancer endpoint
	☑ Final
Name	fs.s3a.access.key
Value	OM[redacted]BAN
Description	SG CDP S3 access key
	☑ Final
Name	fs.s3a.secret.key
Value	mapz[redacted]Qfc
Description	SG CDP S3 secret key
	☑ Final
Name	fs.s3a.impl
Value	org.apache.hadoop.fs.s3a.S3AFileSystem
Description	
	☑ Final
Name	fs.s3a.path.style.access
Value	true
Description	
	☑ Final

Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml

Save Changes(CTRL+S)

1. [Save Changes]ボタンをクリックします。HDFSメニューバーからStale Configurationアイコンを選択し、次のページでRestart Stale Servicesを選択して、Restart Nowを選択します。



StorageGRID へのS3A接続をテストします

基本的な接続テストを実行します

Clouderaクラスタのいずれかのホストにログインし、「hadoop fs s-ls s3a：//<*bucket-name*>/`」と入力します。

次の例では、パスsyleと既存のHDFSテストバケットおよびテストオブジェクトを使用します。

```
[root@ce-n1 ~]# hadoop fs -ls s3a://hdfs-test/
22/02/15 18:24:37 WARN impl.MetricsConfig: Cannot locate configuration:
tried hadoop-metrics2-s3a-file-system.properties,hadoop-
metrics2.properties
22/02/15 18:24:37 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot
period at 10 second(s).
22/02/15 18:24:37 INFO impl.MetricsSystemImpl: s3a-file-system metrics
system started
22/02/15 18:24:37 INFO Configuration.deprecation: No unit for
fs.s3a.connection.request.timeout(0) assuming SECONDS
Found 1 items
-rw-rw-rw-   1 root root       1679 2022-02-14 16:03 s3a://hdfs-test/test
22/02/15 18:24:38 INFO impl.MetricsSystemImpl: Stopping s3a-file-system
metrics system...
22/02/15 18:24:38 INFO impl.MetricsSystemImpl: s3a-file-system metrics
system stopped.
22/02/15 18:24:38 INFO impl.MetricsSystemImpl: s3a-file-system metrics
system shutdown complete.
```

トラブルシューティング

シナリオ **1**

StorageGRID へのHTTPS接続を使用し、15分後に「handshake_failure」エラーを取得します。

*理由：StorageGRID への接続に古いTLS暗号スイートまたはサポートされていないTLS暗号スイートを使用しているJRE／JDKの旧バージョン。

エラーメッセージの例

```
[root@ce-n1 ~]# hadoop fs -ls s3a://hdfs-test/
22/02/15 18:52:34 WARN impl.MetricsConfig: Cannot locate configuration:
tried hadoop-metrics2-s3a-file-system.properties,hadoop-
metrics2.properties
22/02/15 18:52:34 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot
period at 10 second(s).
22/02/15 18:52:34 INFO impl.MetricsSystemImpl: s3a-file-system metrics
system started
22/02/15 18:52:35 INFO Configuration.deprecation: No unit for
fs.s3a.connection.request.timeout(0) assuming SECONDS
22/02/15 19:04:51 INFO impl.MetricsSystemImpl: Stopping s3a-file-system
metrics system...
22/02/15 19:04:51 INFO impl.MetricsSystemImpl: s3a-file-system metrics
system stopped.
22/02/15 19:04:51 INFO impl.MetricsSystemImpl: s3a-file-system metrics
system shutdown complete.
22/02/15 19:04:51 WARN fs.FileSystem: Failed to initialize fileystem
s3a://hdfs-test/: org.apache.hadoop.fs.s3a.AWSClientIOException:
doesBucketExistV2 on hdfs: com.amazonaws.SdkClientException: Unable to
execute HTTP request: Received fatal alert: handshake_failure: Unable to
execute HTTP request: Received fatal alert: handshake_failure
ls: doesBucketExistV2 on hdfs: com.amazonaws.SdkClientException: Unable to
execute HTTP request: Received fatal alert: handshake_failure: Unable to
execute HTTP request: Received fatal alert: handshake_failure
```

*解決策: JDK 11.x以降がインストールされていることを確認し'デフォルトのJavaライブラリに設定しますを参照してください Javaパッケージをインストールします 詳細については、を参照してください。

シナリオ**2**：

StorageGRID に接続できませんでした。エラーメッセージ「要求されたターゲットへの有効な証明書パスが見つかりませんでした」が表示されます。

理由： StorageGRID S3エンドポイントサーバ証明書がJavaプログラムで信頼されていません。

エラーメッセージの例：

```
[root@hdp6 ~]# hadoop fs -ls s3a://hdfs-test/
22/03/11 20:58:12 WARN impl.MetricsConfig: Cannot locate configuration:
tried hadoop-metrics2-s3a-file-system.properties,hadoop-
metrics2.properties
22/03/11 20:58:13 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot
period at 10 second(s).
22/03/11 20:58:13 INFO impl.MetricsSystemImpl: s3a-file-system metrics
system started
22/03/11 20:58:13 INFO Configuration.deprecation: No unit for
fs.s3a.connection.request.timeout(0) assuming SECONDS
22/03/11 21:12:25 INFO impl.MetricsSystemImpl: Stopping s3a-file-system
metrics system...
22/03/11 21:12:25 INFO impl.MetricsSystemImpl: s3a-file-system metrics
system stopped.
22/03/11 21:12:25 INFO impl.MetricsSystemImpl: s3a-file-system metrics
system shutdown complete.
22/03/11 21:12:25 WARN fs.FileSystem: Failed to initialize fileystem
s3a://hdfs-test/: org.apache.hadoop.fs.s3a.AWSClientIOException:
doesBucketExistV2 on hdfs: com.amazonaws.SdkClientException: Unable to
execute HTTP request: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find
valid certification path to requested target: Unable to execute HTTP
request: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find
valid certification path to requested target
```

*解決策：ネットアップは、既知のパブリック証明書署名機関が発行するサーバ証明書を使用して、認証がセキュアであることを確認することを推奨しています。または、Javaの信頼ストアにカスタムのCA証明書またはサーバ証明書を追加します。

StorageGRID カスタムCA証明書またはサーバ証明書をJava信頼ストアに追加するには、次の手順を実行します。

1. 既存のデフォルトのJava cacertsファイルをバックアップします。

```
cp -ap $JAVA_HOME/lib/security/cacerts
$JAVA_HOME/lib/security/cacerts.orig
```

2. StorageGRID S3エンドポイント証明書をJava信頼ストアにインポートします。

```
keytool -import -trustcacerts -keystore $JAVA_HOME/lib/security/cacerts
-storepass changeit -noprompt -alias sg-lb -file <StorageGRID CA or
server cert in pem format>
```

トラブルシューティングのヒント

1. Hadoopログレベルを引き上げてデバッグします。

   'export hadoop root_logger = hadoop .root.logger = debug、console'

2. コマンドを実行し、ログメッセージをerror.logに送信します。

   「hadoop fs s-ls s3a：//<*bucket-name*>_ error.log

Angela Cheng著_

S3cmdを使用して、StorageGRID でS3アクセスをテストおよび実証します

S3cmdは、S3処理用の無償のコマンドラインツールおよびクライアントです。s3cmdを使用して、StorageGRID でのS3アクセスをテストして実証できます。

S3cmdをインストールして構成します

ワークステーションまたはサーバにS3cmdをインストールするには、からダウンロードします "コマンドラインS3クライアント"。s3cmdは、トラブルシューティング用のツールとして、各StorageGRID ノードにあらかじめインストールされています。

初期設定手順

1. s3cmd --設定
2. 残りのキーには、access-keyとsecret_keyだけを指定してデフォルトのままにします。
3. 指定したクレデンシャルでアクセスをテストします[Y/n]：n（失敗するため、テストをバイパスする）
4. 設定を保存しますか？[y/N] y
   a. 設定を「/root/.s3cfg」に保存しました。
5. s3cfgで、「=」記号のあとにhost_baseフィールドとhost_bucketフィールドを空にします。
   a. host_base=
   b. host_bucket=

手順4でhost_baseとhost_bucketを指定した場合は、CLIで—hostのエンドポイントを指定する必要はありません。例

```
host_base = 192.168.1.91:8082
host_bucket = bucketX.192.168.1.91:8082
s3cmd ls s3://bucketX --no-check-certificate
```

基本的なコマンドの例

- バケットを作成：

  s3cmd mb s3：//s3cmdbucket --host=<endpoint>：<port>--no-check-certificate-`

- すべてのバケットを表示：

  s3cmd ls --host=<endpoint>：<port>--no-check-certificate'

- すべてのバケットとその内容を表示：

  s3cmd la --host=<endpoint>：<port>-- no-check-certificate'

- 特定のバケット内のオブジェクトをリストします。

  s3cmd ls s3：//<bucket>--host=<endpoint>：<port>--no-check-certificate-`

- バケットを削除：

  s3cmd rb s3：//s3cmdbucket --host=<endpoint>：<port>--no-check-certificate'

- オブジェクトを置きなさい:

  s3cmd put <file>s3://<bucket>--host=<endpoint>:<port>--no-check-certificate`

- オブジェクトを取得：

  s3cmd get s3：//<バケット>/<オブジェクト><ファイル>--host=<endpoint>：<port>--no-check-certificate'

- オブジェクトを削除：

  s3cmd del s3：//<bucket>/<object>--host=<endpoint>：<port>：-no-check-certificate-`

アロンクライン著

NetApp StorageGRID を共有ストレージとして使用したVertica Eonモードのデータベース

このガイドでは、NetApp StorageGRID のパブリックストレージを使用してVertica Eon Modeデータベースを作成する手順 について説明します。

はじめに

Verticaは分析データベース管理ソフトウェアです。大量のデータを処理するように設計されたカラム型ストレージ・プラットフォームであり'従来の負荷の高いシナリオでは非常に高速なクエリー・パフォーマンスを実現しますVerticaデータベースは、EonまたはEnterpriseのいずれかのモードで動作します。両方のモードをオンプレミスまたはクラウドに導入できます。

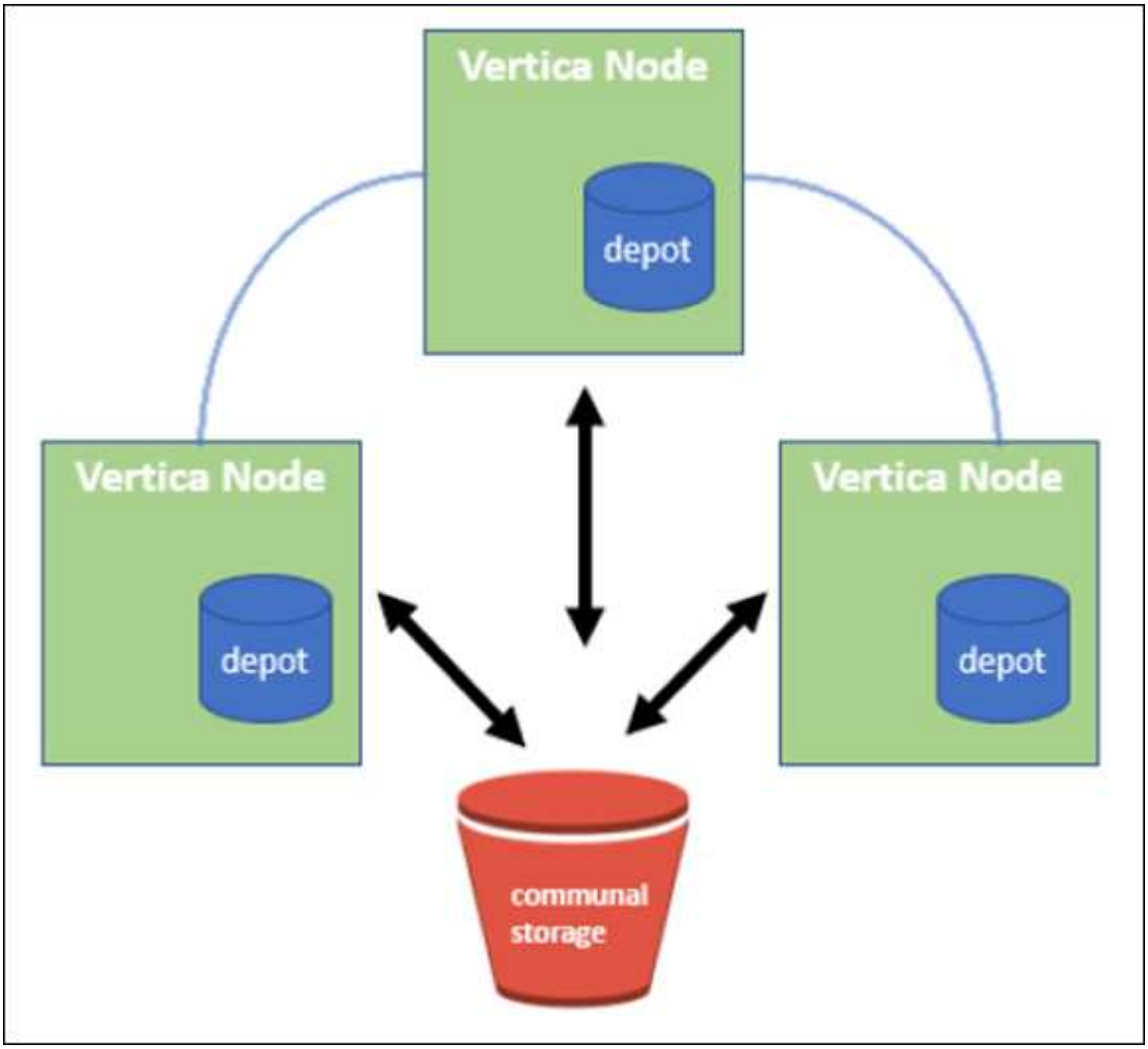
EonモードとEnterpriseモードは、主にデータの保存場所によって異なります。

- Eonモードのデータベースは、データ用に共有ストレージを使用します。これはVerticaがお勧めします。
- Enterprise Modeデータベースでは、データベースを構成するノードのファイルシステムにデータがローカルに格納されます。

Eon Modeアーキテクチャ

Eonモードでは、計算リソースがデータベースの共有ストレージレイヤから分離され、コンピューティングとストレージを別々に拡張できます。EonモードのVerticaは、さまざまなワークロードに対応し、コンピューティングリソースとストレージリソースを別々に使用してワークロードを分離するように最適化されています。

Eon Modeは、パブリックストレージと呼ばれる共有オブジェクトストアにデータを格納します。パブリックストレージとは、オンプレミスまたはAmazon S3上にホストされるS3バケットです。



共有ストレージ

Eonモードでは、データをローカルに格納する代わりに、すべてのデータとカタログ（メタデータ）に単一の共有ストレージロケーションを使用します。共有ストレージとは、データベースの一元管理されたストレージの場所で、データベースノード間で共有されるものです。

共有ストレージには次のプロパティがあります。

- クラウドまたはオンプレミスのオブジェクトストレージ内の共有ストレージは、個々のマシンのディスク上のストレージよりも耐障害性が高く、ストレージ障害によるデータ損失の影響を受けにくくなっています。
- すべてのデータは、同じパスを使用して任意のノードで読み取ることができます。
- ノードのディスクスペースによる容量制限はありません。
- データは通信環境に保管されるため、変化するニーズに合わせてクラスタを柔軟に拡張できます。データがノードにローカルに格納されていた場合は、ノードを追加または削除するときに、ノード間で移動するデータが大量に必要になります。これを行うには、削除対象のノードから移動するか、新しく作成したノードに移動する必要があります。

デポ

共有ストレージの欠点の1つは速度です。共有クラウド上の場所からデータにアクセスする場合、ローカルディスクからデータを読み取る場合よりも時間がかかります。また、多数のノードが一度にデータを読み取っている場合、共有ストレージへの接続がボトルネックになる可能性があります。データアクセス速度を向上させるために、Eon Modeデータベース内のノードは、デポと呼ばれるデータのローカルディスクキャッシュを保持します。クエリを実行するとき、ノードはまず、必要なデータがデポにあるかどうかをチェックします。存在する場合は、データのローカルコピーを使用してクエリが完了します。データがデポにない場合、ノードは共有ストレージからデータを取得し、デポにコピーを保存します。

NetApp StorageGRID の推奨事項

Verticaは、データベースのデータをオブジェクトストレージに何千（数百万）もの圧縮オブジェクトとして格納します（1オブジェクトあたり200～500MB）。ユーザーがデータベースクエリを実行すると、Verticaはバイト範囲GET呼び出しを使用して、圧縮されたオブジェクトから選択したデータ範囲を並列に取得します。バイト範囲GETはそれぞれ約8KBです。

10TBのデータベースデポのユーザクエリテストでは、1秒あたり4、000~ 10、000個のGET（バイト範囲GET）要求がグリッドに送信されました。SG6060アプライアンスを使用してこのテストを実行した場合、アプライアンスノードあたりのCPU利用率は（2030%程度）が低いため、CPU時間の2/3でI/Oを待機しているSGF6024では、I/O待機時間のごく一部（0% 0.5%）が確認されます。

IOPSは小さいが高いことから、レイテンシの要件は非常に低い（平均値は0.01秒未満）ため、オブジェクトストレージサービスにはSFG6024を使用することを推奨します。非常に大きなデータベースサイズにSG6060が必要な場合は、お客様はデポサイジングのVerticaアカウントチームと協力して、照会中のデータセットをサポートする必要があります。

管理ノードとAPIゲートウェイノードの場合は、お客様がSG100またはSG1000を使用できます。選択する内容は、ユーザのクエリ要求の並列サイズとデータベースサイズによって異なります。他社製ロードバランサを使用する場合は、ハイパフォーマンスが要求されるワークロードに専用のロードバランサを使用することを推奨します。StorageGRID のサイジングについては、ネットアップアカウントチームにお問い合わせください。

StorageGRID 構成に関するその他の推奨事項は次のとおりです。

- グリッドトポロジ。同じグリッドサイトにある他のストレージアプライアンスモデルとSGF6024を混在させないでください。長期アーカイブ保護にSG6060を使用する場合は、アクティブデータベース用に専用のグリッドロードバランサを使用してSGF6024の負荷を専用のグリッドサイト（物理サイトまたは論理サイト）に配置し、パフォーマンスを向上させます。同じサイトに異なるモデルのアプライアンスを混在させると、サイト全体のパフォーマンスが低下します。
- データ保護。レプリケートコピーを使用して保護します。アクティブデータベースにはイレイジャーコー

ディングを使用しないでください。イレイジャーコーディングを使用することで、アクセス頻度の低いデータベースを長期にわたって保護できます。

- グリッド圧縮を有効にしないでください。Verticaは、オブジェクトを圧縮してからオブジェクトストレージに格納します。グリッド圧縮を有効にしてもストレージ使用量はこれ以上削減されず、バイト範囲のGETパフォーマンスが大幅に低下します。
- * HTTPとHTTPS S3エンドポイント接続*。ベンチマークテストでは、VerticaクラスタからStorageGRIDロードバランサエンドポイントへのHTTP S3接続を使用した場合、パフォーマンスが約5%向上しました。この選択は、顧客のセキュリティ要件に基づいて行う必要があります。

Vertica構成に関する推奨事項は次のとおりです。

- * Verticaデータベースのデフォルトデポ設定は、読み取りおよび書き込み操作で有効(値=1)になっています。*パフォーマンスを向上させるために、これらのデポ設定を有効にしておくことを強く推奨します。
- *ストリーミング制限を無効にします。*設定の詳細については、を参照してください ストリーミング制限を無効にしています。

StorageGRID 上の共有ストレージを使用してオンプレミスモードをインストールする

以下のセクションでは、StorageGRID 上に共同ストレージを使用してオンプレミスにEonモードをインストールするための手順 について説明します。オンプレミスのSimple Storage Service（S3）互換オブジェクトストレージを設定する手順 は、Vertica guideの手順 に似ています。 "オンプレミスにEonモードデータベースをインストールします"。

機能テストには次のセットアップを使用しました。

- StorageGRID 11.4.0.4
- Vertica 10.1.0
- Verticaノードをクラスタに構成するために、CentOS 7.x OSを搭載した3台の仮想マシン（VM）。このセットアップは、Verticaプロダクションデータベースクラスタではなく、機能テストのみを対象としています。

これらの3つのノードにはSecure Shell（SSH）キーが設定されており、クラスタ内のノード間でパスワードを設定することなくSSHを使用できます。

NetApp StorageGRID で必要な情報

StorageGRID 上で共有ストレージを使用してオンプレミスにEonモードをインストールするには、次の前提条件情報が必要です。

- StorageGRID S3エンドポイントのIPアドレスまたは完全修飾ドメイン名（FQDN）とポート番号。HTTPSを使用する場合は、StorageGRID S3エンドポイントに実装されているカスタムの認証局（CA）または自己署名SSL証明書を使用します。
- バケット名。このパラメータは、あらかじめ存在し、空である必要があります。
- バケットへの読み取り/書き込みアクセスが可能なアクセスキーIDとシークレットアクセスキー。

S3エンドポイントにアクセスするための認証ファイルを作成します

S3エンドポイントにアクセスする許可ファイルを作成する際には、次の前提条件が適用されます。

- Verticaがインストールされている。
- クラスタをセットアップして設定し、データベースを作成できる状態にします。

S3エンドポイントにアクセスするための認証ファイルを作成するには、次の手順を実行します。

1. 「admintools」を実行してEon Modeデータベースを作成するVerticaノードにログインします。

   デフォルトのユーザーは'dbadmin'で'Verticaクラスタのインストール時に作成されます

2. テキスト・エディタを使用して'/HOME/dbadminディレクトリの下にファイルを作成しますファイル名には'たとえばsg_auth.confなど'任意の名前を指定できます
3. S3エンドポイントが標準のHTTPポート80またはHTTPSポート443を使用している場合は、ポート番号を省略します。HTTPSを使用するには、次の値を設定します。

   - `awsenablehttps=1'それ以外の場合は'0`に値を設定します
   - `awsauth=<s3 access key ID>：<secret access key>`
   - `awsendpoint=< StorageGRID s3 endpoint>:<port>`

     StorageGRID S3エンドポイントのHTTPS接続にカスタムCA証明書または自己署名SSL証明書を使用するには、証明書の完全なファイルパスとファイル名を指定します。このファイルは、各Verticaノード上の同じ場所にあり、すべてのユーザーに読み取り権限が与えられている必要があります。StorageGRID S3エンドポイントのSSL証明書が一般に知られているCAによって署名されている場合は、この手順を省略します。

     -awscafile=<filepath/filename>`

     たとえば、次のサンプルファイルを参照してください。

```
awsauth = MNVU4OYFAY2xyz123:03vuO4M4KmdfwffT8nqnBmnMVTr78Gu9wANabcxyz
awsendpoint = s3.england.connectlab.io:10443
awsenablehttps = 1
awscafile = /etc/custom-cert/grid.pem
```

+

> 本番環境では、一般に知られているCAによって署名されたサーバ証明書をStorageGRID S3ロードバランサエンドポイントに実装する必要があります。

すべての**Vertica**ノードのデポパスを選択します

デポストレージパスの各ノードにディレクトリを選択または作成します。デポストレージパスパラメータに指定するディレクトリには、次のものが必要です。

- クラスタ内のすべてのノードで同じパス（例：/home/dbadmin/depot）
- dbadminユーザによる読み書きが可能になります
- 十分なストレージ

デフォルトでは、Verticaはデポ保存用のディレクトリを含むファイルシステム領域の60%を使用します。'create-db'コマンドの—depot-size'引数を使用すると、デポのサイズを制限できます。を参照してください "EonモードデータベースのVertica Clusterのサイジング" Verticaの一般的なサイジングガイドラインについては、こちらをご覧ください。Vertica Account Managerにお問い合わせください。

'admintools create-db'ツールは'存在しない場合に備えて'デポパスを作成しようとします

オンプレミスデータベースの作成

オンプレミスデータベースを作成するには、次の手順を実行します。

1. データベースを作成するには'admintools create-db'ツールを使用します

   この例で使用されている引数の簡単な説明を次に示します。すべての必須引数とオプション引数の詳細については、Verticaのドキュメントを参照してください。

   - -x <で作成された認証ファイルのパス/ファイル名 「S3エンドポイントにアクセスするための認証ファイルの作成」 >。

     認証の詳細は、正常に作成された後、データベース内に保存されます。S3シークレットキーの公開を回避するために、このファイルを削除できます。

   - --son/storagegrid-sstorage -location <s3：//storagegrid bucketname>
   - -s <このデータベースに使用するVerticaノードのカンマ区切りリスト>
   - -d <作成するデータベースの名前>
   - -p <この新しいデータベースに設定するパスワード>。たとえば、次のコマンド例を参照してください。

     ```
     admintools -t create_db -x sg_auth.conf --communal-storage
     -location=s3://vertica --depot-path=/home/dbadmin/depot --shard
     -count=6 -s vertica-vm1,vertica-vm2,vertica-vm3 -d vmart -p
     '<password>'
     ```

     データベースのノード数によっては、新しいデータベースの作成に数分かかることがあります。データベースを初めて作成するときに、ライセンス契約に同意するように求められます。

たとえば'次のサンプル認証ファイルと'create db'コマンドを参照してください

```
[dbadmin@vertica-vm1 ~]$ cat sg_auth.conf
awsauth = MNVU4OYFAY2CPKVXVxxxx:03vuO4M4KmdfwffT8nqnBmnMVTr78Gu9wAN+xxxx
awsendpoint = s3.england.connectlab.io:10445
awsenablehttps = 1

[dbadmin@vertica-vm1 ~]$ admintools -t create_db -x sg_auth.conf
--communal-storage-location=s3://vertica --depot-path=/home/dbadmin/depot
--shard-count=6 -s vertica-vm1,vertica-vm2,vertica-vm3 -d vmart -p
'xxxxxxxx'
```

```
Default depot size in use
Distributing changes to cluster.
    Creating database vmart
    Starting bootstrap node v_vmart_node0007 (10.45.74.19)
    Starting nodes:
        v_vmart_node0007 (10.45.74.19)
    Starting Vertica on all nodes. Please wait, databases with a large
catalog may take a while to initialize.
    Node Status: v_vmart_node0007: (DOWN)
    Node Status: v_vmart_node0007: (DOWN)
    Node Status: v_vmart_node0007: (DOWN)
    Node Status: v_vmart_node0007: (UP)
    Creating database nodes
    Creating node v_vmart_node0008 (host 10.45.74.29)
    Creating node v_vmart_node0009 (host 10.45.74.39)
    Generating new configuration information
    Stopping single node db before adding additional nodes.
    Database shutdown complete
    Starting all nodes
Start hosts = ['10.45.74.19', '10.45.74.29', '10.45.74.39']
    Starting nodes:
        v_vmart_node0007 (10.45.74.19)
        v_vmart_node0008 (10.45.74.29)
        v_vmart_node0009 (10.45.74.39)
    Starting Vertica on all nodes. Please wait, databases with a large
catalog may take a while to initialize.
    Node Status: v_vmart_node0007: (DOWN) v_vmart_node0008: (DOWN)
v_vmart_node0009: (DOWN)
    Node Status: v_vmart_node0007: (DOWN) v_vmart_node0008: (DOWN)
v_vmart_node0009: (DOWN)
    Node Status: v_vmart_node0007: (DOWN) v_vmart_node0008: (DOWN)
v_vmart_node0009: (DOWN)
    Node Status: v_vmart_node0007: (DOWN) v_vmart_node0008: (DOWN)
v_vmart_node0009: (DOWN)
    Node Status: v_vmart_node0007: (UP) v_vmart_node0008: (UP)
v_vmart_node0009: (UP)
Creating depot locations for 3 nodes
Communal storage detected: rebalancing shards

Waiting for rebalance shards. We will wait for at most 36000 seconds.
Installing AWS package
    Success: package AWS installed
Installing ComplexTypes package
    Success: package ComplexTypes installed
Installing MachineLearning package
    Success: package MachineLearning installed
```

```
Installing ParquetExport package
    Success: package ParquetExport installed
Installing VFunctions package
    Success: package VFunctions installed
Installing approximate package
    Success: package approximate installed
Installing flextable package
    Success: package flextable installed
Installing kafka package
    Success: package kafka installed
Installing logsearch package
    Success: package logsearch installed
Installing place package
    Success: package place installed
Installing txtindex package
    Success: package txtindex installed
Installing voltagesecure package
    Success: package voltagesecure installed
Syncing catalog on vmart with 2000 attempts.
Database creation SQL tasks completed successfully. Database vmart created
successfully.
```

オブジェクトのサイズ（バイト）	バケット/オブジェクトキーの完全パス
61`	s3://Vertica/051/026d63ae9d4a33237bf0e2cf2a794a794a000000000021a07/026d63ae9d4a33237bf0e2cf2a794a00a0000000000000021a07_00.dfd
145`	s3://Vertica/2c4/026d63ae9d4a33237bf0e2cf2a794a794a794a000000000000000021a3d/026d63ae9d4a33237bf0e2cf2a794a794a00a000000000000021a3_0.dfd
146 `	s3://Vertica/33C/026d63ae9d4a33237bf0e2cf2a794a0000000021a1d/026d63ae9d4a33237bf0e2cf2a794a00000000000021a1d_0.dfd
「40」	s3://Vertica/382/026d63ae9d4a33237bf0e2cf2a794a794a0000000021a31/026d63ae9d4a33237bf0e2cf2a794a794a000000000021a31_0.dfs

オブジェクトのサイズ（バイト）	バケット/オブジェクトキーの完全パス
145`	s3://Vertica/42f/026d63ae9d4a33237bf0e2cf2a794a794a00000000211/026d63ae9d4a33237bf0e2cf2a794a0000000000000021a_0.dfd
34`	s3://Vertica/472/026d63ae9d4a33237bf0e2cf2a794a794a000000000021a25/026d63ae9d4a33237bf0e2cf2a794a000000000000000021a25_0.df
41.	s3://Vertica/476/026d63ae9d4a33237bf0e2cf2a794a794a00000000000021a2d/026d63ae9d4a33237bf0e2c2cf2a794a00a00000000000021a2_0.dfd
61`	s3://Vertica/52A/026d63ae9d4a33237bf0e2cf2a794a794a0000000021a5d/026d63ae9d4a33237bf0e2cf2a794a794a000000000021a5d_0.df
「131」	s3://Vertica/5d2/026d63ae9d4a33237bf0e2cf2a794a794a000000000021a19/026d63ae9d4a33237bf0e2cf2a794a00a0000000000000021a19_0.df
「91」	s3://Vertica/5f7/026d63ae9d4a33237bf0e2cf2a794a794a000000000021a11/026d63ae9d4a33237bf0e2cf2a794a00a00000000000021a11_0.df
「118」	s3://Vertica/82D/026d63ae9d4a33237bf0e2cf2a794a794a0000000021a15/026d63ae9d4a33237bf0e2cf2a794a000000000000000021a15_0.df
「115」	s3://Vertica/922/026d63ae9d4a33237bf0e2cf2a794a794a0000000021a61/026d63ae9d4a33237bf0e2cf2a794a000000000000000021a61_0.df
「33」	s3://Vertica/ACD/026d63ae9d4a33237bf0e2cf2a794a794a000000000021a29/026d63ae9d4a33237bf0e2cf2a794a794a00000000000021a29_0.dfs

オブジェクトのサイズ（バイト）	バケット/オブジェクトキーの完全パス
「133」	s3://Vertica/b98/026d63ae9d4a33237bf0e2cf2a794a794a00a00000000000021a4d/026d63ae9d4a33237bf0e2cf2a794a794a00a000000000021a4d_0.df
「38」	s3://Vertica/db3/026d63ae9d4a33237bf0e2cf2a794a794a000000000021a49/026d63ae9d4a33237bf0e2cf2a794a00a0000000000000021a49_0.df
「38」	`s3://Vertica/eba/026d63ae9d4a33237bf0e2cf2a794a794a000000000021a59/026d63ae9d4a33237bf0e2cf2a794a794a0000000021a59_0.dfdfs.dfs
21521920`	s3://vertica/metadata/VMart/Library/Libraryd63ae9d4a33237bf0e2c2cf2a794a00a000000000000215e2/026d63ae9d4a33237bf0e2c2cf2a794a000000215e2.tar
6865408`	s3://vertica/metodat/VMart/Library/Libraryd63ae9d4a33237bf0e2c2cf2a794a00a0000000021602/026d63ae9d4a33237bf0e2c2cf2a794a0000000000000021602.tar
「204217344」	s3://vertica/metodata/mart/Library/Libraryd63ae9d4a33237bf0e2c2cf2a794a00a0000000021610/026d63ae9d4a33237bf0e2c2cf2a794a00a00000000000000021610.tar
16109056`	s3://vertica/metadata/VMart/Library/Libraryd63ae9d4a33237bf0e2c2cf2a794a00a00000000217e0/026d63ae9d4a33237bf0e2c2cf2a794a0000000000217e0.tar
12853248`	s3://vertica/metadata/VMart/Library/Libraryd63ae9d4a33237bf0e2c2cf2a794a00a0000000000000021800/026d63ae9d4a33237bf0e2c2cf2a794a00a000000000000000000000000218.tar
「8937984」と入力します	s3://vertica/metadata/VMart/Library/Libraryd63ae9d4a33237bf0e2c2cf2a794a00a00000000002187a/026d63ae9d4a33237bf0e2c2cf2a794a00a00002187a.tar

オブジェクトのサイズ（バイト）	バケット/オブジェクトキーの完全パス
「56260`606060860」	s3://vertica/metadata/VMart/Library/Libraryd63ae9d4a33237bf0e2c2cf2a794a00a00000000000000218b2/026d63ae9d4a33237bf0e2c2cf2a794a0000000000218b2.tar
「53947904」	s3://vertica/metadata/VMart/Library/Libraryd63ae9d4a33237bf0e2c2cf2a794a00a00000000000000219ba/026d63ae9d4a33237bf0e2c2cf2a794a0000000000219ba.tar
44932`608	s3://vertica/metadata/VMart/Library/Libraryd63ae9d4a33237bf0e2c2cf2a794a00a000000000000219de/026d63ae9d4a33237bf0e2c2cf2a794a00000000000000219de.tar
「256306688」	s3://vertica/metadata/VMart/Librarys/026d63ae9d4a33237bf0e2c2cf2a794a00a00000000000021a6e/026d63ae9d4a33237bf0e2c2cf2a794a000000000000000021a6e.tar
「8062464`」	s3://vertica/metadata/VMart/Library/Libraryd63ae9d4a33237bf0e2c2cf2a794a00a0000000021e34/026d63ae9d4a33237bf0e2c2cf2a794a0000000000000000021e34.tar
「20024832」	s3://vertica/metadata/VMart/Library/Libraryd63ae9d4a33237bf0e2c2cf2a794a00a0000000021e70/026d63ae9d4a33237bf0e2c2cf2a794a0000000000000000021e70.tar
「10444」	`s3://vertica/metadata/VMart/cluster _config.json
「823266」	s3://vertica/metadata/VMart/nodes/v_vmart_node0016/Catalog/859703b06a3456d95d0be28575a673/Checks/C13_13/chkpt_1.cat.gz
「254」	s3://vertica/metadata/VMart/nodes/v_vmart_node0016/Catalog/859703b06a3456d95d0be28575a673/Checks/C13_13/Completed
「2958」	s3://vertica/metadata/VMart/nodes/v_vmart_node0016/Catalog/859703b06a3456d95d0be28575a673/Checkpoints/C2_2/chkpt_1.cat.gz

オブジェクトのサイズ（バイト）	バケット/オブジェクトキーの完全パス
231`	s3://vertica/metadata/VMart/nodes/v_vmart_node0016/Catalog/859703b06a3456d95d0be28575a673/Checks/C2_2/Completed
「822521」	s3://vertica/metadata/VMart/nodes/v_vmart_node0016/Catalog/859703b06a3456d95d0be28575a673/Checks/C4_4/chkpt_1.cat.gz
231`	s3://vertica/metadata/VMart/nodes/v_vmart_node0016/Catalog/859703b06a3456d95d0be28575a673/Checks/C4_4/Completed
746513`	s3://vertica/metadata/VMart/nodes/v_vmart_node0016/Catalog/859703b06a3456d95d0be28575a673/Txnlogs/txn_14_g14.cat
「2596」	s3://vertica/metadata/VMart/nodes/v_vmart_node0016/Catalog/859703b06a3456d95d0be28575a673/Txnlogs/txn_3_g3.cat.gz
821065`	s3://vertica/metadata/VMart/nodes/v_vmart_node0016/Catalog/859703b06a3456d95d0be28575a673/Txnlogs/txn_4_g4.cat.gz
6440`	s3://vertica/metadata/VMart/nodes/v_vmart_node0016/Catalog/859703b06a3456d95d0be28575a673/Txnlogs/txn_5_g5.cat
「8518」	s3://vertica/metadata/VMart/nodes/v_vmart_node0016/Catalog/859703b06a3456d95d0be28575a673/Txnlogs/txn_8_g8.cat
「 0 」	s3://vertica/metadata/VMart/nodes/v_vmart_node0016/Catalog/859703b06a3456d95d0be28575a673/tiered_catalog.cat
822922`	s3://vertica/metadata/VMart/nodes/v_vmart_node0017/Catalog/859703b06a3456d95d0be28575a673/Checkpoints /Checkpoints /C14-7/chkpt_1.cat.gz

オブジェクトのサイズ（バイト）	バケット/オブジェクトキーの完全パス
「232」	`s3://vertica/metadata/VMart/nodes/v_vmart_node0017/Catalog/859703b06a3456d95d0be28575a673/Checkpoints/Checkpoints /C14-7/Completed`
822930`	`s3://vertica/metadata/VMart/nodes/v_vmart_node0017/Catalog/859703b06a3456d95d0be28575a673/Txnlogs/txn_14_g7.cat.gz`
755033`	`s3://vertica/metadata/VMart/nodes/v_vmart_node0017/Catalog/859703b06a3456d95d0be28575a673/Txnlogs/txn_15_g8.cat`
「 0 」	`s3://vertica/metadata/VMart/nodes/v_vmart_node0017/Catalog/859703b06a3456d95d0be28575a673/tiered_catalog.cat`
822922`	`s3://vertica/metadata/VMart/nodes/v_vmart_node0018/Catalog/859703b06a3456d95d0be28575a673/Checkpoints/Checkpoints /C14-7/chkpt_1.cat.gz`
「232」	`s3://vertica/metadata/VMart/nodes/v_vmart_node0018/Catalog/859703b06a3456d95d0be28575a673/Checkpoints/Checkpoints /C14-7/Completed`
822930`	`s3://vertica/metadata/VMart/nodes/v_vmart_node0018/Catalog/859703b06a3456d95d0be28575a673/Txnlogs/txn_14_g7.cat.gz`
755033`	`s3://vertica/metadata/VMart/nodes/v_vmart_node0018/Catalog/859703b06a3456d95d0be28575a673/Txnlogs/txn_15_g8.cat`
「 0 」	`s3://vertica/metadata/VMart/nodes/v_vmart_node0018/Catalog/859703b06a3456d95d0be28575a673/tiered_catalog.cat`

ストリーミング制限を無効にしています

この手順 は、他のオンプレミスオブジェクトストレージのVertica guideに基づいており、StorageGRID に適用する必要があります。

1. データベースを作成したら'AWSStreamingConnectionPercentage'設定パラメータを0に設定して無効にしますこの設定は、共同ストレージを使用したオンプレミス環境でのEonモードのインストールには不要です。この設定パラメータは、Verticaがストリーミング読み取りに使用するオブジェクトストアへの接続数を制御します。クラウド環境では、この設定が有効な場合、オブジェクトストアからのストリーミングデータが使用可能なすべてのファイルハンドルを使い使わないようにすることができます。他のオブジェクトストア処理に使用できるファイルハンドルが残っています。オンプレミスのオブジェクトストアのレイテンシが低いため、このオプションは不要です。
2. パラメータ値を更新するには'vsql'文を使用しますパスワードは、「オンプレミスデータベースの作成」で設定したデータベースパスワードです。たとえば、次の出力例を参照してください。

```
[dbadmin@vertica-vm1 ~]$ vsql
Password:
Welcome to vsql, the Vertica Analytic Database interactive terminal.
Type:   \h or \? for help with vsql commands
        \g or terminate with semicolon to execute query
        \q to quit
dbadmin=> ALTER DATABASE DEFAULT SET PARAMETER
AWSStreamingConnectionPercentage = 0; ALTER DATABASE
dbadmin=> \q
```

デポの設定を確認してい

Verticaデータベースのデフォルトデポ設定は、読み取りおよび書き込み操作に対して有効(値=1)です。パフォーマンスを向上させるために、これらのデポ設定を有効にしておくことを強く推奨します。

```
vsql -c 'show current all;' | grep -i UseDepot
DATABASE | UseDepotForReads | 1
DATABASE | UseDepotForWrites | 1
```

サンプルデータのロード（オプション）

このデータベースをテスト用に使用し、削除する場合は、サンプルデータをテスト用にこのデータベースにロードできます。Verticaには、各Verticaノードの「/opt/vertica/examples/VMart _Schema/」にあるサンプルデータセットVMartが付属しています。このサンプルデータセットの詳細については、を参照してください "こちらをご覧ください"。

サンプルデータをロードするには、次の手順を実行します。

1. いずれかのVerticaノードにdbadminとしてログインします。cd /opt/vertica/examples/VMart _Schema/
2. サンプルデータをデータベースにロードし、手順cとdでプロンプトが表示されたらデータベースのパスワードを入力します。
   a. 「cd /opt/vertica/examples/VMart _Schema'」と入力します
   b. 「./vmart_gen」
   c. vsql <vmart_define_schema.sql
   d. 「vsql <vmart_load_data.sql」

3. 事前定義された複数のSQLクエリがあります。そのうちの一部を実行して、テストデータがデータベースに正常にロードされたことを確認できます。たとえば、「vsql <vmart_queries1.sql」のようになります

追加情報の参照先

このドキュメントに記載されている情報の詳細については、以下のドキュメントや Web サイトを参照してください。

- "NetApp StorageGRID 11.7製品ドキュメント"
- "StorageGRID データシート"
- "Vertica 10.1製品マニュアル"

バージョン履歴

バージョン	日付	ドキュメントのバージョン履歴
バージョン 1.0 以降	2021年9月	初版リリース

Angela Cheng著_

エルクスタックを使用した**StorageGRID** ログ分析

StorageGRID 11.6 syslog転送機能を使用すると、StorageGRID ログメッセージを収集および分析するように外部syslogサーバを設定できます。エルク（Elasticsearch、Logstash、Kibana）は、最も人気のあるログ分析ソリューションの1つになっています。をご覧ください "エルク・ビデオを使用したStorageGRID ログ解析" サンプルのエルク設定を表示し、失敗したS3要求を特定してトラブルシューティングするためにどのように使用できるかを確認する。この記事では、StorageGRID ログの管理と分析をすばやく開始できるように、Logstashの設定、Kibanaのクエリ、グラフ、およびダッシュボードのサンプルファイルを紹介します。

要件

- StorageGRID 11.6.0.2以降
- Elk（Elasticsearch、Logstash、Kibana）7.1x以降がインストールされており、動作中です

サンプルファイル

- "Logstash 7.xサンプルファイルパッケージをダウンロードします" +*MD5チェックサム*148c23d0021d9a4bb4a6c0287464deab +*SHA256チェックサム*f51ec9e2e3f842d5a7861566b167a561beb4373038b4e7bb3c8be3d522adf2d6
- "Logstash 8.xサンプルファイルパッケージをダウンロードします" *MD5チェックサム*e11bae3a662f87c310ef363d0fe06835* SHA256チェックサム*5c670755742cfdfd5aa723a596ba087e0153a65bcaef3934afdb682f61cd278d

前提条件

読者はStorageGRID およびElkの用語および操作に精通しています。

指示

grokパターンで定義される名前の違いにより、2つのサンプルバージョンが提供されます。+たとえば、Logstash設定ファイルのSYSLOGBASE grokパターンでは、インストールされているLogstashのバージョンによってフィールド名が異なります。

```
match => {"message" => '<%{POSINT:syslog_pri}>%{SYSLOGBASE}
%{GREEDYDATA:msg-details}'}
```

- Logstash 7.17サンプル*

Field	Value
t _id	7C1MaYEBRH8UbfKnIls8
t _index	sgrid2-2022.06.15
# _score	-
t _type	_doc
@timestamp	Jun 15, 2022 @ 17:36:46.038
t host	grid2-site2-s1
t logsource	SITE2-S1
t msg-details	Reloading syslog service
t pid	628
t program	update-sysl
t syslog_pri	37
t timestamp	Jun 15 21:36:46

ログスタシュ**8.23**サンプル

Table JSON

Search field names

Actions	Field	Value
	_id	yuhOiIEBVP6KX4EwqcyU
	_index	sglog-2022.06.21
	_score	-
	@timestamp	Jun 21, 2022 @ 18:07:45.444
	event.original	<28>Jun 21 22:07:45 SITE2-S3 ADE: syslog messages being dropped
	host.hostname	SITE2-S3
	msg-details	syslog messages being dropped
	process.name	ADE
	syslog_pri	28
	timestamp	Jun 21 22:07:45

- 手順 *

1. インストールされているエルクバージョンに基づいて、提供されたサンプルを解凍します。サンプル・フォルダには'Logstash configサンプルが2つ含まれています**sglog-2-file.conf :**この構成ファイルは’データ変換を行わずに**Logstash**上のファイルに**StorageGRID** ログ・メッセージを出力しますこの機能を使用すると、**Logstash**が**StorageGRID** メッセージを受信していることを確認したり、**StorageGRID** ログパターンを理解したりできます。**+** sglog-2-es.conf：*この構成ファイルは、さまざまなパターンやフィルタを使用してStorageGRID ログメッセージを変換します。この例には、パターンまたはフィルタに基づいてメッセージをドロップするDROPステートメントが含まれています。インデックスを作成するために出力がElasticsearchに送信されます。+ファイル内の指示に従って、選択した構成ファイルをカスタマイズします。
2. カスタマイズした構成ファイルをテストします。

```
/usr/share/logstash/bin/logstash --config.test_and_exit -f <config-
file-path/file>
```

返される最後の行が次の行に似ている場合、構成ファイルに構文エラーはありません。

```
[LogStash::Runner] runner - Using config.test_and_exit mode. Config
Validation Result: OK. Exiting Logstash
```

3. カスタマイズされたconfファイルをLogstashサーバのconfig:/etc/logstash/conf.d+にコピーします/etc/logstash/logstash.ymlでconfig.reload.automaticを有効にしていない場合は’Logstashサービスを再起動しますそれ以外の場合は、設定のリロード間隔が経過するのを待ちます。

```
 grep reload /etc/logstash/logstash.yml
# Periodically check if the configuration has changed and reload the
pipeline
config.reload.automatic: true
config.reload.interval: 5s
```

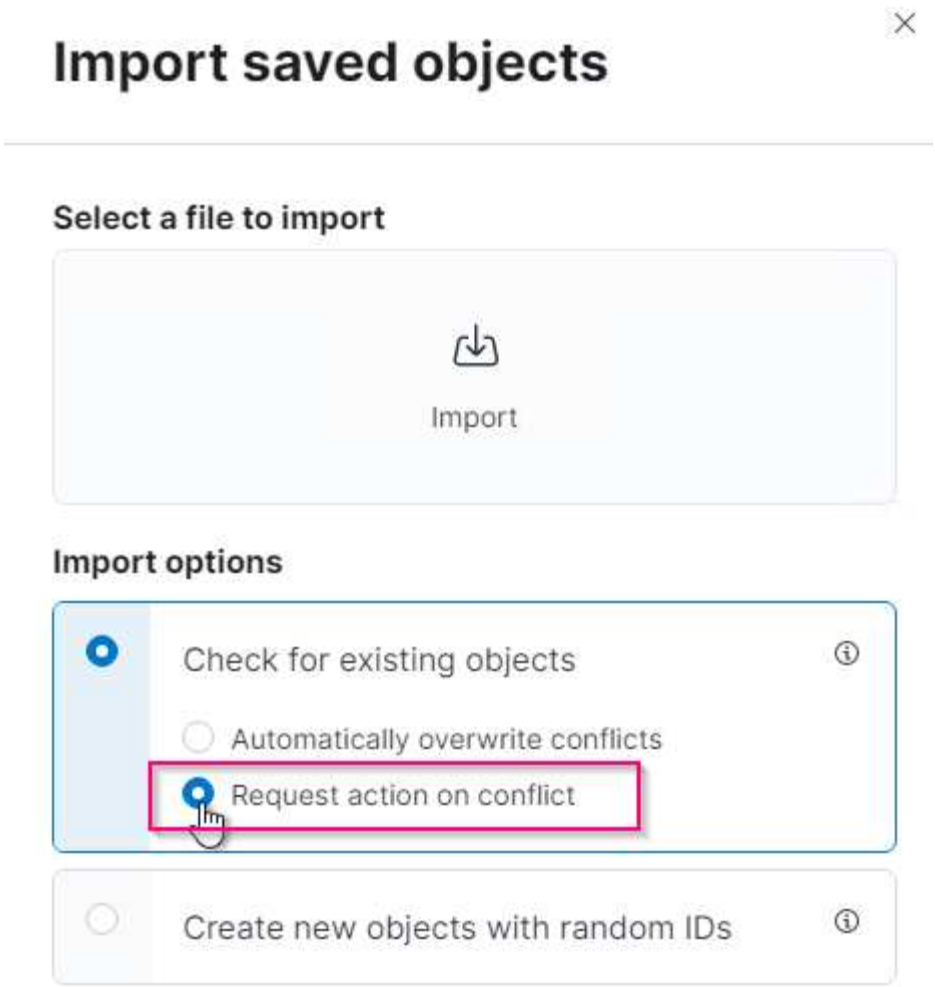
4. /var/log/logstash/logstash-plain.logを確認し、Logstashを新しい設定ファイルで起動する際にエラーがないことを確認します。
5. TCPポートが開始され、リスンしていることを確認する。+この例では、TCPポート5000が使用されています。

```
netstat -ntpa | grep 5000
tcp6       0      0 :::5000                 :::*
LISTEN      25744/java
```

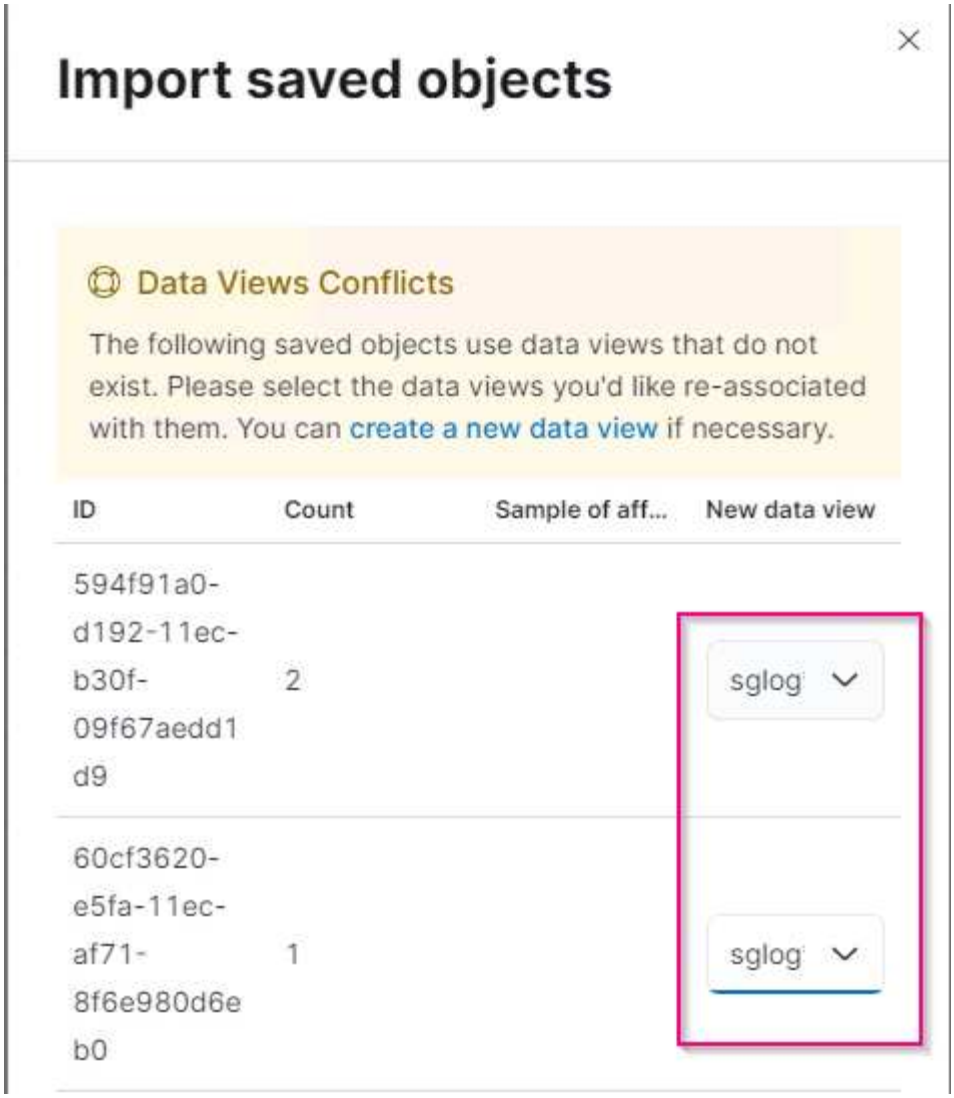
6. StorageGRID マネージャGUIから、ログメッセージをLogstashに送信するように外部syslogサーバを設定します。を参照してください "デモビデオ" を参照してください。
7. 定義されたTCPポートへのStorageGRID ノード接続を許可するには、Logstashサーバ上でファイアウォールを設定または無効にする必要があります。
8. Kibana GUIから、[Management]→[Dev Tools]を選択します。Consoleページで、次のgetコマンドを実行して、Elasticsearch上に新しいインデックスが作成されていることを確認します。

```
GET /_cat/indices/*?v=true&s=index
```

9. Kibana GUIから、索引パターン（Elk 7.x）またはデータビュー（Elk 8.x）を作成します。
10. Kibana GUIから、上部中央にある検索ボックスに「saved objects」と入力します。+[保存済みオブジェクト]ページで、[インポート]を選択します。[インポートオプション]で、[競合時にアクションを要求]を選択します。



elk <version>-query-chart-sample.ndjsonをインポートします。+競合を解決するよう求められたら、手順8で作成したインデックスパターンまたはデータビューを選択します。



次のKibanaオブジェクトがインポートされます。 *クエリ** audit-msg-s3rq -lm+* bycast log s3関連メッセージ+* loglevel warningまたはabove * failed security event +* Chart ** s3要求数bycast.log * HTTP status code +* audit breakdown by type +* average s3応答 上記のグラフを使用した、時間ダッシュボード+* S3要求ダッシュボード。

これで、Kibanaを使用してStorageGRID ログ分析を実行する準備ができました。

その他のリソース

- "syslog101"
- "エルクスタックとは何ですか"
- "grokパターンリスト"
- "初心者向けのLogstashガイド: Grok"
- "ログスタシュの実践的なガイド：syslogの詳細"
- "Kibanaガイド–ドキュメントを参照してください"
- "StorageGRID 監査ログメッセージリファレンスです"

Angela Cheng著_

PrometheusとGrafanaを使用して指標の保持を拡張します

このテクニカルレポートでは、外部のPrometheusサービスおよびGrafanaサービスでNetApp StorageGRID 11.6を設定する詳しい手順を説明します。

はじめに

StorageGRID は、Prometheusを使用して指標を保存し、組み込みのGrafanaダッシュボードでこれらの指標を視覚化します。Prometheus指標には、クライアントアクセス証明書を設定し、指定されたクライアントのPrometheusアクセスを有効にすることで、StorageGRID から安全にアクセスできます。現在、この指標データの保持期間は管理ノードのストレージ容量によって制限されています。これらの指標のカスタマイズされた可視化を実現するために、新しいPrometheusサーバとGrafanaサーバを導入し、新しいサーバでStorageGRIDWebscaleインスタンスから指標をスクラビングするように設定し、重要な指標を使用したダッシュボードを構築します。で収集されたPrometheus指標の詳細を確認できます "StorageGRID のドキュメント"。

Prometheusをフェデレーションする

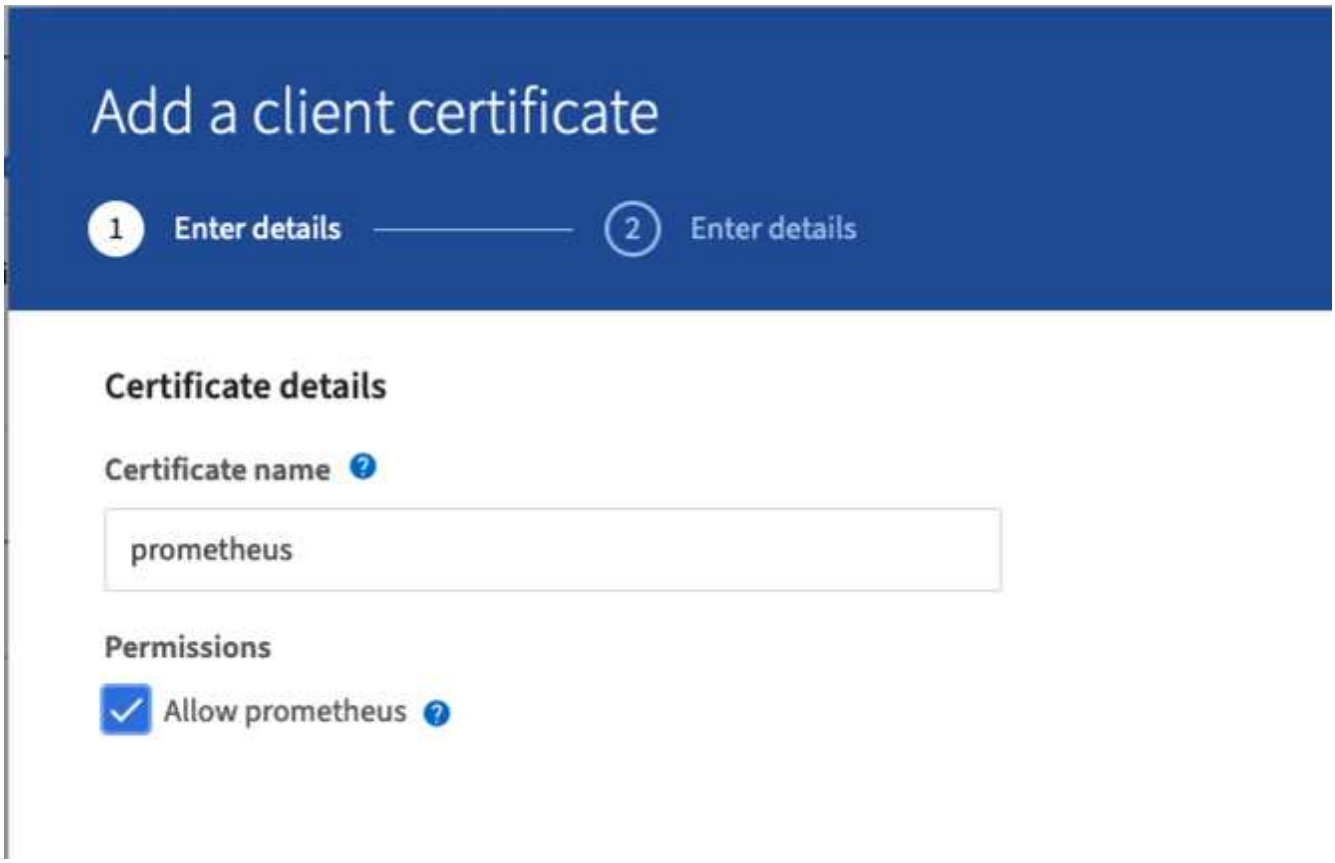
ラボの詳細

この例では、StorageGRID 11.6ノードとDebian 11サーバのすべての仮想マシンを使用します。StorageGRID管理インターフェイスには、公開されている信頼されたCA証明書が設定されています。この例では、StorageGRID システムやDebian Linuxのインストールと設定は行われません。PrometheusとGrafanaでサポートされている、任意のLinuxフレーバーを使用できます。PrometheusとGrafanaはどちらも、Dockerコンテナ、ソースからビルド、またはコンパイル済みのバイナリとしてインストールできます。この例では、PrometheusバイナリとGrafanaバイナリの両方を同じDebianサーバに直接インストールします。から基本的なインストール手順をダウンロードして実行します https://prometheus.io および https://grafana.com/grafana/ それぞれ。

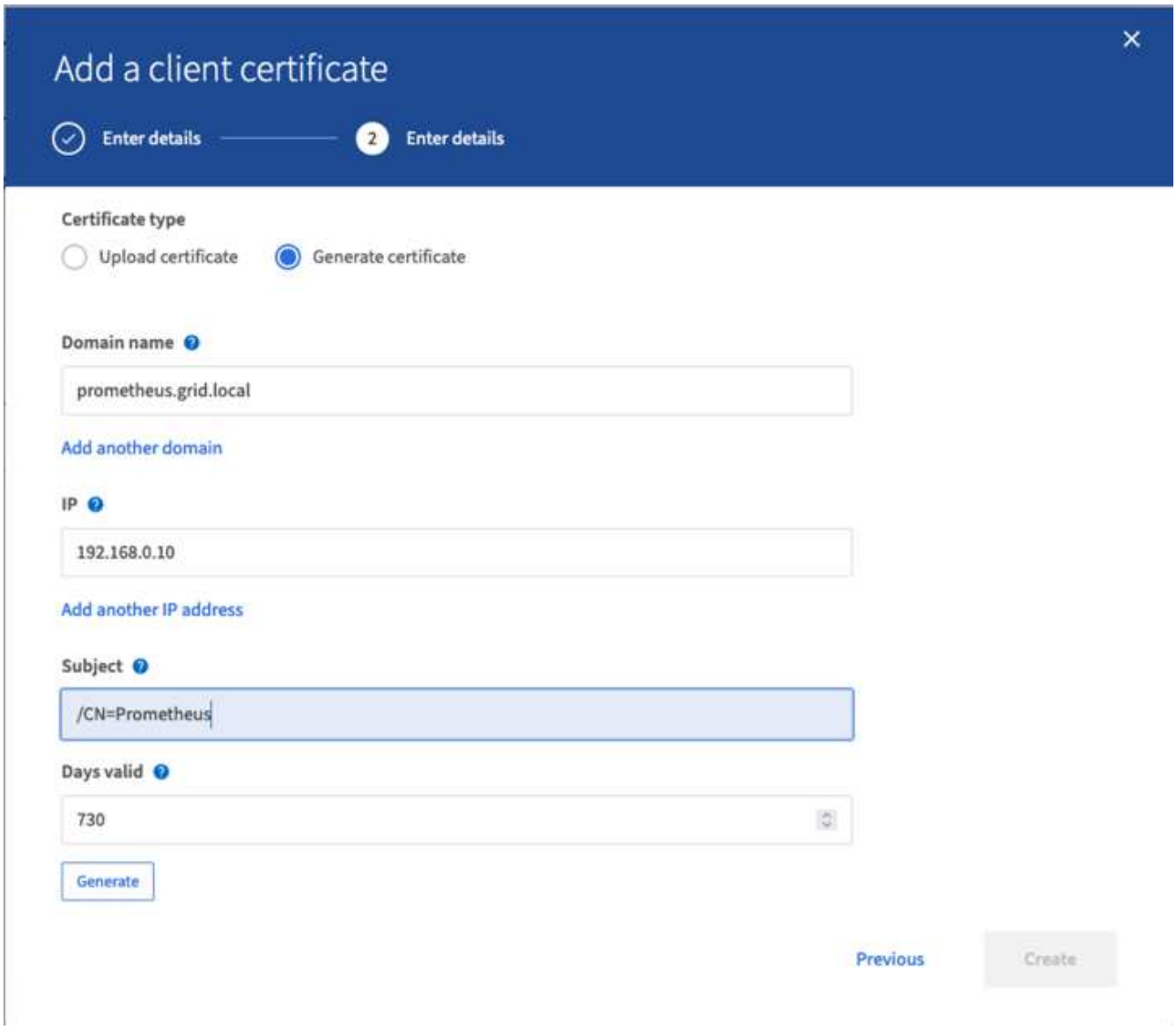
Prometheusクライアントアクセス用にStorageGRID を設定する

StorageGRID IDに格納されているPrometheus指標にアクセスするには、秘密鍵を使用してクライアント証明書を生成またはアップロードし、クライアントの権限を有効にする必要があります。StorageGRID 管理インターフェイスにはSSL証明書が必要です。この証明書は、信頼されたCAによってPrometheusサーバによって信頼されているか、自己署名されている場合は手動で信頼されている必要があります。詳細については、を参照してください "StorageGRID のドキュメント"。

1. StorageGRID 管理インターフェイスの左下にある「configuration」を選択し、2番目の列にある「Security」で「Certificates」をクリックします。
2. [証明書]ページで[クライアント]タブを選択し、[追加]ボタンをクリックします。
3. アクセスを許可するクライアントの名前を指定し、この証明書を使用します。「Allow Prometheus」の前の「Permissions」のボックスをクリックし、「Continue」ボタンをクリックします。

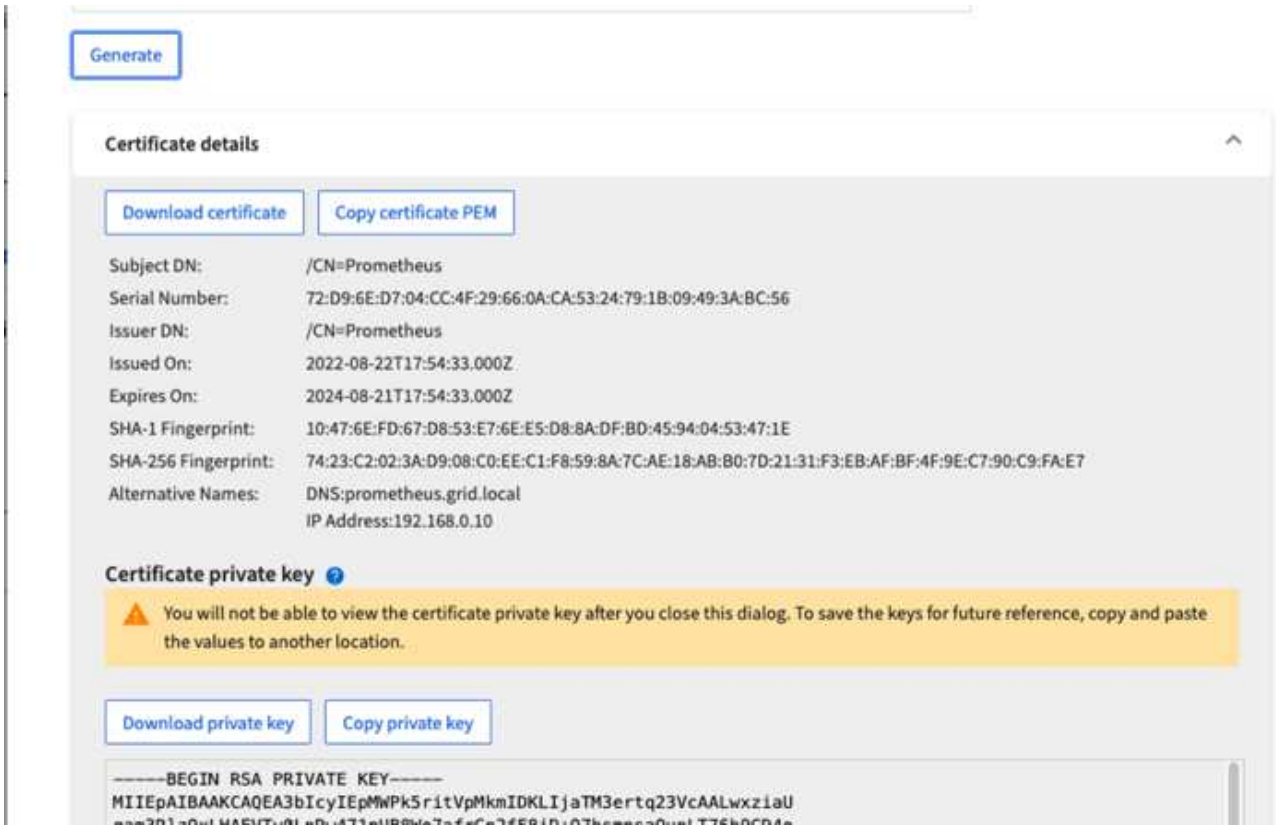


4. CA署名証明書がある場合は、[証明書のアップロード]のラジオボタンを選択できますが、この場合は、[証明書の生成]のラジオボタンを選択して、StorageGRID がクライアント証明書を生成できるようにします。入力する必須フィールドが表示されます。クライアントサーバのFQDN、サーバのIP、件名、有効日数を入力します。「生成」ボタンをクリックします。




```
Be mindful of the certificate days valid entry as you will need
to renew this certificate in both StorageGRID and the Prometheus
server before it expires to maintain uninterrupted collection.
```

1. 証明書のPEMファイルと秘密鍵のPEMファイルをダウンロードします。



> This is the only time you can download the private key, so make sure you do not skip this step.

Linuxサーバで**Prometheus**インストールを準備

Prometheusをインストールする前に、Prometheusユーザとディレクトリ構造を使用して環境を準備し、指標の格納場所の容量を設定します。

1. Prometheusユーザを作成します。

```
sudo useradd -M -r -s /bin/false Prometheus
```

2. Prometheus、クライアント証明書、指標データのディレクトリを作成します。

```
sudo mkdir /etc/Prometheus /etc/Prometheus/cert /var/lib/Prometheus
```

3. 私はext4ファイルシステムでのメトリック保持のために使用するディスクをフォーマットしました。

```
mkfs -t ext4 /dev/sdb
```

4. そのあと、Prometheusのmetricsディレクトリにファイルシステムをマウントしました。

```
sudo mount -t auto /dev/sdb /var/lib/prometheus/
```

5. 指標データに使用するディスクのUUIDを取得します。

```
sudo ls -al /dev/disk/by-uuid/
   lrwxrwxrwx 1 root root   9 Aug 18 17:02 9af2c5a3-bfc2-4ec1-85d9-
ebab850bb4a1 -> ../../sdb
```

6. /etc/fstabにエントリを追加してマウントをリブート後も/dev/sdbのUUIDを使用して維持するようにします。

```
/etc/fstab
UUID=9af2c5a3-bfc2-4ec1-85d9-ebab850bb4a1 /var/lib/prometheus   ext4
defaults    0   0
```

Prometheusをインストールして設定する

これでサーバの準備ができました。Prometheusのインストールを開始して、サービスを設定できます。

1. Prometheusインストールパッケージを展開します

```
tar xzf prometheus-2.38.0.linux-amd64.tar.gz
```

2. バイナリを/usr/local/binにコピーし、前の手順で作成したPrometheusユーザに所有権を変更します

```
sudo cp prometheus-2.38.0.linux-amd64/{prometheus,promtool}
/usr/local/bin
sudo chown prometheus:prometheus /usr/local/bin/{prometheus,promtool}
```

3. コンソールとライブラリを/etc/Prometheusにコピーします

```
sudo cp -r prometheus-2.38.0.linux-amd64/{consoles,console_libraries}
/etc/prometheus/
```

4. 以前にStorageGRID からダウンロードしたクライアント証明書と秘密鍵のPEMファイルを/etc/prometheus/certsにコピーします
5. Prometheus設定YAMLファイルを作成します

```
sudo nano /etc/prometheus/prometheus.yml
```

6. 次の構成を挿入します。ジョブ名には、任意の名前を指定できます。「-targets：[']]」を管理ノードのFQDNに変更し、証明書と秘密鍵のファイル名を変更した場合は、tls_configセクションを更新して一致させてください。次に、ファイルを保存します。グリッド管理インターフェイスで自己署名証明書を使用している場合は、証明書をダウンロードして一意の名前のクライアント証明書に格納し、tls_configセクションadd ca_file：/etc/prometheus/cert/UIcert.pemに格納します

   a. この例では、alertmanager、cassandra、node、およびStorageGRID で始まるすべての指標を収集しています。Prometheus指標の詳細については、を参照してください "StorageGRID のドキュメント"。

```
# my global config
global:
  scrape_interval: 60s # Set the scrape interval to every 15 seconds.
Default is every 1 minute.

scrape_configs:
  - job_name: 'StorageGRID'
    honor_labels: true
    scheme: https
    metrics_path: /federate
    scrape_interval: 60s
    scrape_timeout: 30s
    tls_config:
      cert_file: /etc/prometheus/cert/certificate.pem
      key_file: /etc/prometheus/cert/private_key.pem
    params:
      match[]:
        -
'{__name__=~"alertmanager_.*|cassandra_.*|node_.*|storagegrid_.*"}'
    static_configs:
    - targets: ['sgdemo-rtp.netapp.com:9091']
```

> グリッド管理インターフェイスで自己署名証明書が使用されている場合は、証明書をダウンロードして一意の名前でクライアント証明書に格納します。tls_configセクションで、クライアント証明書と秘密鍵の行の上に証明書を追加します
>
> ```
> ca_file: /etc/prometheus/cert/UIcert.pem
> ```

1. Prometheus内のすべてのファイルとディレクトリの所有権と、/var/lib/prometPrometheusユーザへの所有権を変更する

```
sudo chown -R prometheus:prometheus /etc/prometheus/
sudo chown -R prometheus:prometheus /var/lib/prometheus/
```

2. /etc/systemd/systemにPrometheusサービスファイルを作成します

```
sudo nano /etc/systemd/system/prometheus.service
```

3. 次の行を挿入します。--storage.tsd.dretention.time=1y#というメトリックデータの保持期間を1年にします。また、ストレージの制限に基づいて保持期間を設定する場合も、--storage.tsdb.retentionsize=300GiB#を使用することもできます。指標の保持を設定できるのは、この場所だけです。

```
[Unit]
Description=Prometheus Time Series Collection and Processing Server
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \
        --config.file /etc/prometheus/prometheus.yml \
        --storage.tsdb.path /var/lib/prometheus/ \
        --storage.tsdb.retention.time=1y \
        --web.console.templates=/etc/prometheus/consoles \
        --web.console.libraries=/etc/prometheus/console_libraries

[Install]
WantedBy=multi-user.target
```

4. システムdサービスをリロードして新しいPrometheusサービスを登録します。その後、Prometheusサービスを開始して有効にします。

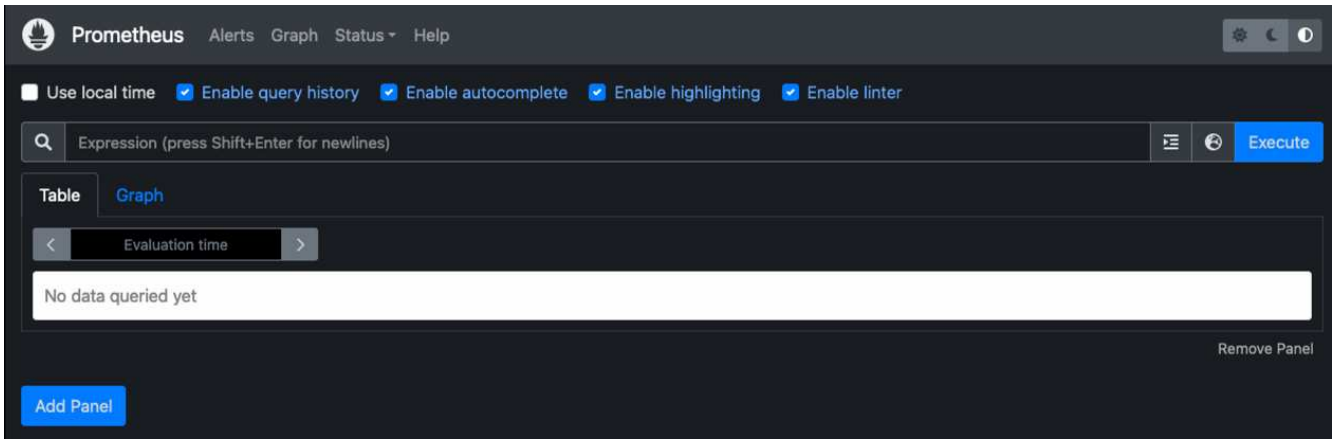
```
sudo systemctl daemon-reload
sudo systemctl start prometheus
sudo systemctl enable prometheus
```

5. サービスが正常に実行されていることを確認します

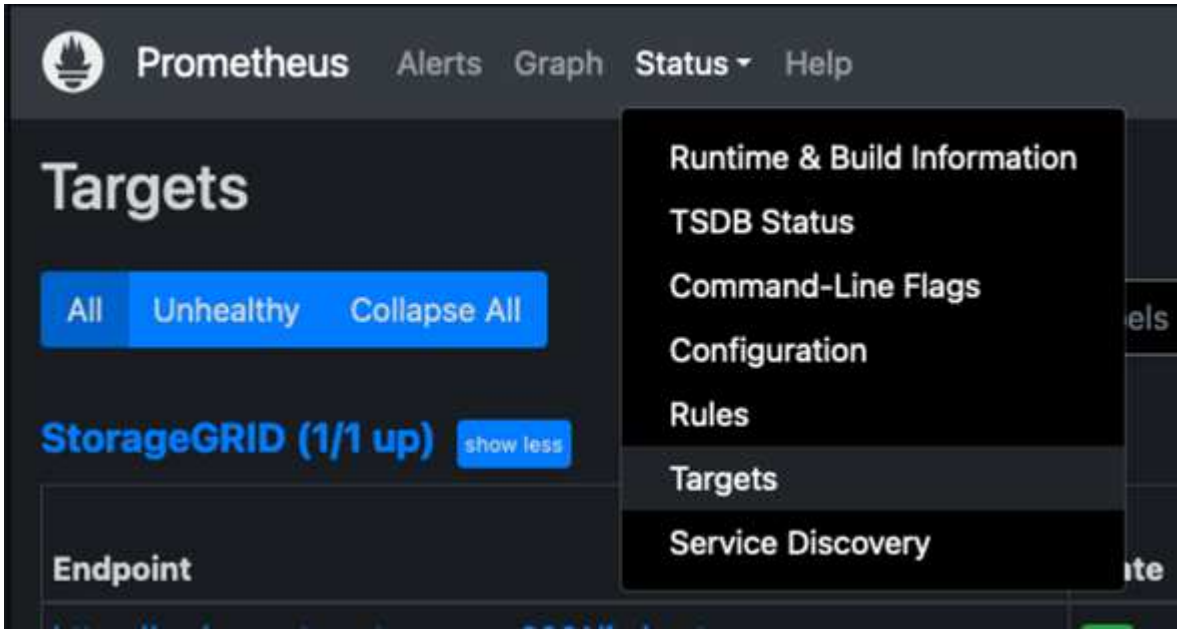
```
sudo systemctl status prometheus
```

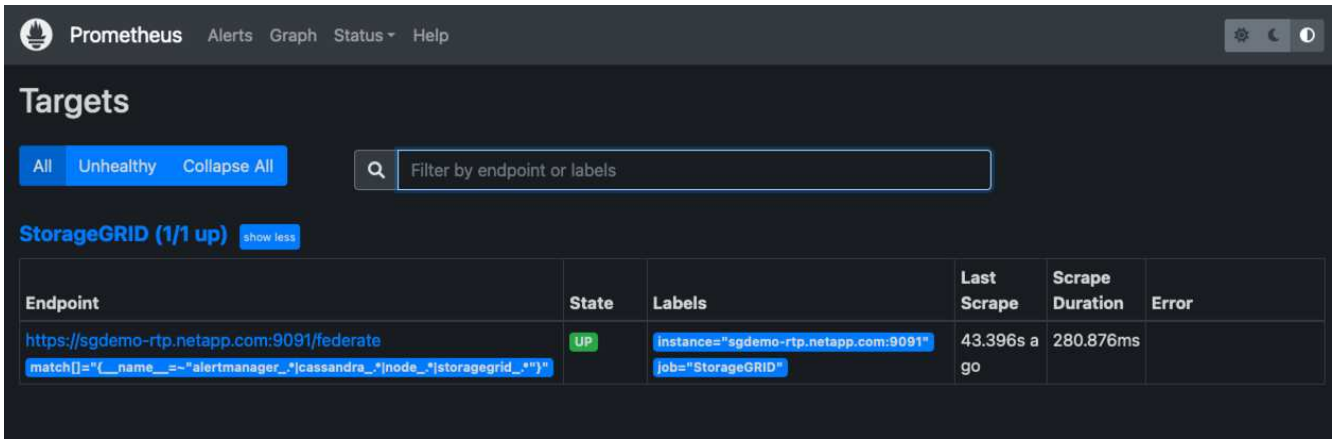
```
● prometheus.service - Prometheus Time Series Collection and Processing
Server
     Loaded: loaded (/etc/systemd/system/prometheus.service; enabled;
vendor preset: enabled)
     Active: active (running) since Mon 2022-08-22 15:14:24 EDT; 2s ago
   Main PID: 6498 (prometheus)
      Tasks: 13 (limit: 28818)
     Memory: 107.7M
        CPU: 1.143s
     CGroup: /system.slice/prometheus.service
             └─6498 /usr/local/bin/prometheus --config.file
/etc/prometheus/prometheus.yml --storage.tsdb.path /var/lib/prometheus/
--web.console.templates=/etc/prometheus/consoles --web.con>

Aug 22 15:14:24 aj-deb-prom01 prometheus[6498]: ts=2022-08-
22T19:14:24.510Z caller=head.go:544 level=info component=tsdb
msg="Replaying WAL, this may take a while"
Aug 22 15:14:24 aj-deb-prom01 prometheus[6498]: ts=2022-08-
22T19:14:24.816Z caller=head.go:615 level=info component=tsdb msg="WAL
segment loaded" segment=0 maxSegment=1
Aug 22 15:14:24 aj-deb-prom01 prometheus[6498]: ts=2022-08-
22T19:14:24.816Z caller=head.go:615 level=info component=tsdb msg="WAL
segment loaded" segment=1 maxSegment=1
Aug 22 15:14:24 aj-deb-prom01 prometheus[6498]: ts=2022-08-
22T19:14:24.816Z caller=head.go:621 level=info component=tsdb msg="WAL
replay completed" checkpoint_replay_duration=55.57µs wal_rep>
Aug 22 15:14:24 aj-deb-prom01 prometheus[6498]: ts=2022-08-
22T19:14:24.831Z caller=main.go:997 level=info fs_type=EXT4_SUPER_MAGIC
Aug 22 15:14:24 aj-deb-prom01 prometheus[6498]: ts=2022-08-
22T19:14:24.831Z caller=main.go:1000 level=info msg="TSDB started"
Aug 22 15:14:24 aj-deb-prom01 prometheus[6498]: ts=2022-08-
22T19:14:24.831Z caller=main.go:1181 level=info msg="Loading
configuration file" filename=/etc/prometheus/prometheus.yml
Aug 22 15:14:24 aj-deb-prom01 prometheus[6498]: ts=2022-08-
22T19:14:24.832Z caller=main.go:1218 level=info msg="Completed loading
of configuration file" filename=/etc/prometheus/prometheus.y>
Aug 22 15:14:24 aj-deb-prom01 prometheus[6498]: ts=2022-08-
22T19:14:24.832Z caller=main.go:961 level=info msg="Server is ready to
receive web requests."
Aug 22 15:14:24 aj-deb-prom01 prometheus[6498]: ts=2022-08-
22T19:14:24.832Z caller=manager.go:941 level=info component="rule
manager" msg="Starting rule manager..."
```

6. PrometheusサーバのUIにアクセスできるようになります http://Prometheus-server:9090 およびUIを参照してください

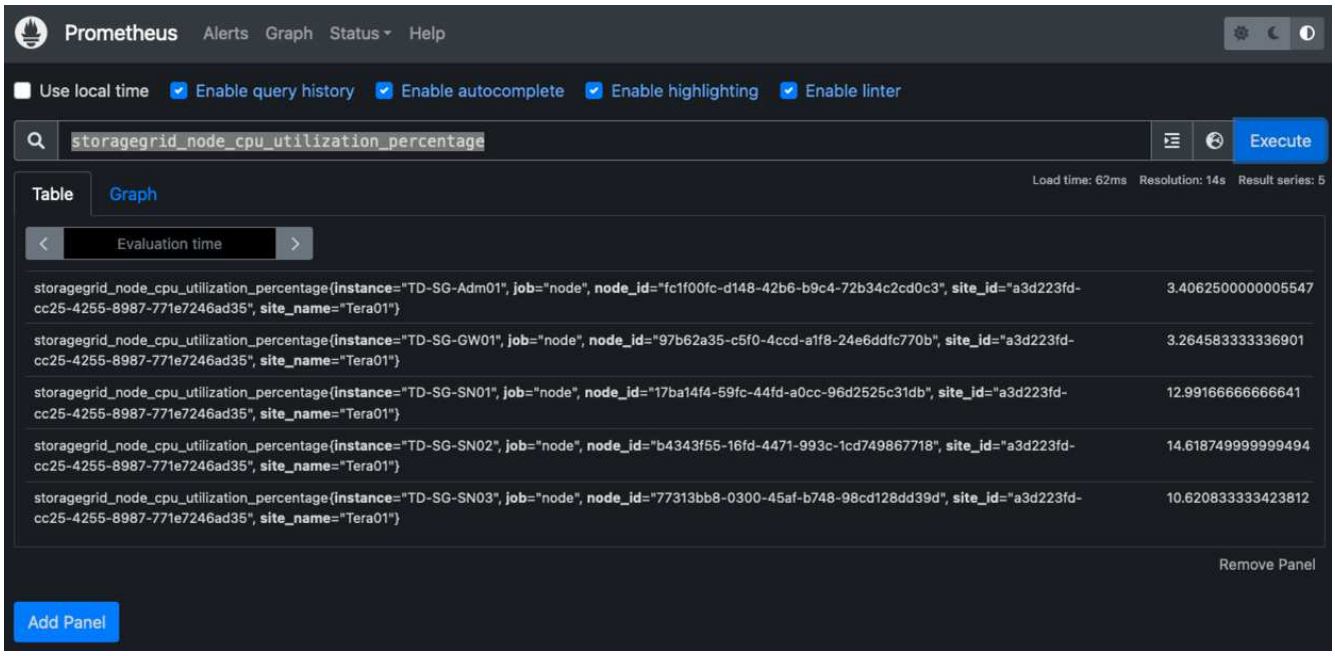


7. 「Status」ターゲットのPrometheusで設定したStorageGRID エンドポイントのステータスを確認できます





8. [グラフ]ページで、テストクエリを実行し、データが正常にスクレイピングされていることを確認できます。たとえば、クエリバーに「storagegrid_node_name utilization _percentage」と入力し、実行ボタンをクリックします。



Grafanaをインストールして設定します

Prometheusがインストールされて機能したので、Grafanaのインストールとダッシュボードの設定に進みます

Grafanaの分析

1. Grafanaの最新のエンタープライズエディションをインストールします

```
sudo apt-get install -y apt-transport-https
sudo apt-get install -y software-properties-common wget
sudo wget -q -O /usr/share/keyrings/grafana.key
https://packages.grafana.com/gpg.key
```

2. 安定版リリース用に次のリポジトリを追加します。

```
echo "deb [signed-by=/usr/share/keyrings/grafana.key]
https://packages.grafana.com/enterprise/deb stable main" | sudo tee -a
/etc/apt/sources.list.d/grafana.list
```

3. リポジトリを追加した後。

```
sudo apt-get update
sudo apt-get install grafana-enterprise
```

4. systemdサービスをリロードして新しいgrafanaサービスを登録します。次に、Grafanaサービスを開始して有効にします。

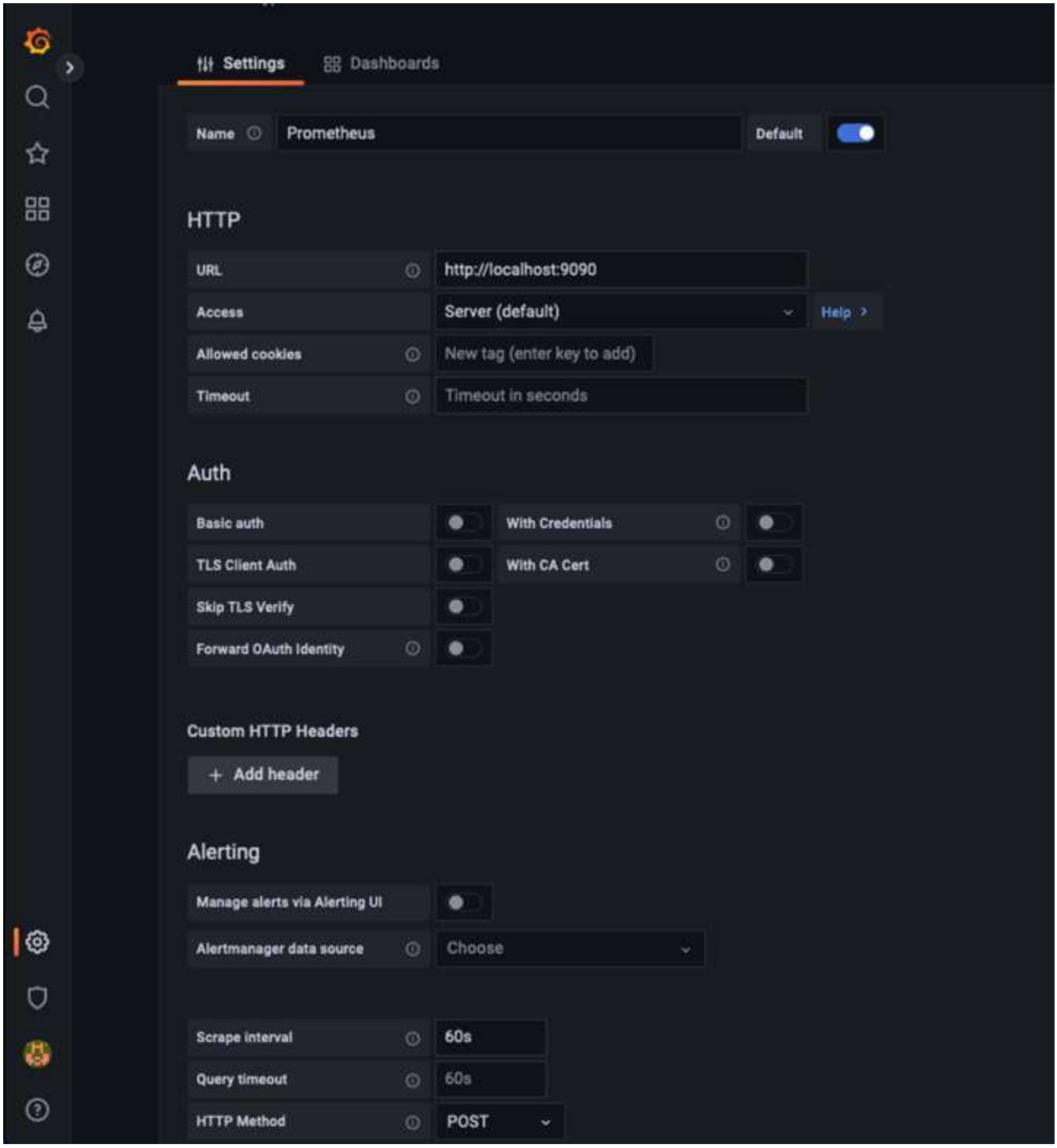
```
sudo systemctl daemon-reload
sudo systemctl start grafana-server
sudo systemctl enable grafana-server.service
```

5. Grafanaがインストールされて実行されるようになりました。ブラウザでHTTP：//prometheus-server：3000にアクセスすると、Grafanaのログインページが表示されます。
6. デフォルトのログインクレデンシャルはadmin / adminであり、新しいパスワードを要求されたときに設定する必要があります。

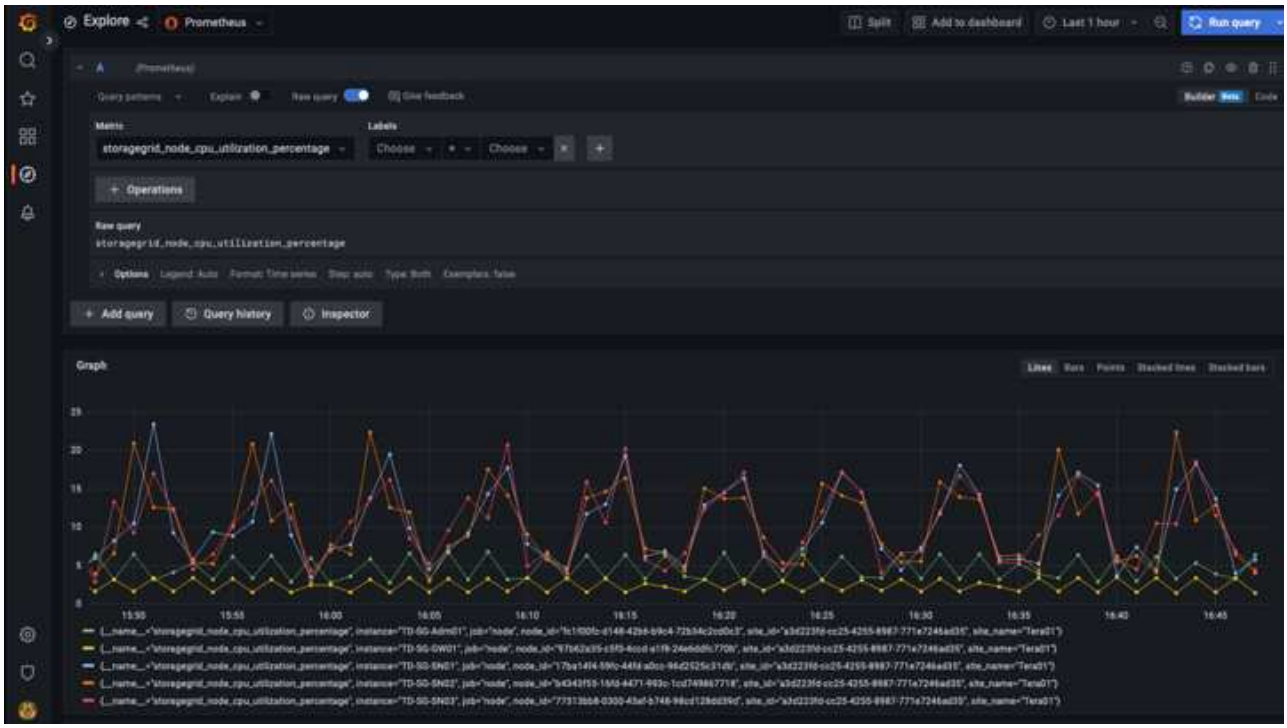
StorageGRID に対応した**Grafana**ダッシュボードを作成します

GrafanaとPrometheusがインストールされて実行されている状態で、データソースを作成してダッシュボードを構築することで、この2つを接続する時間が発生します

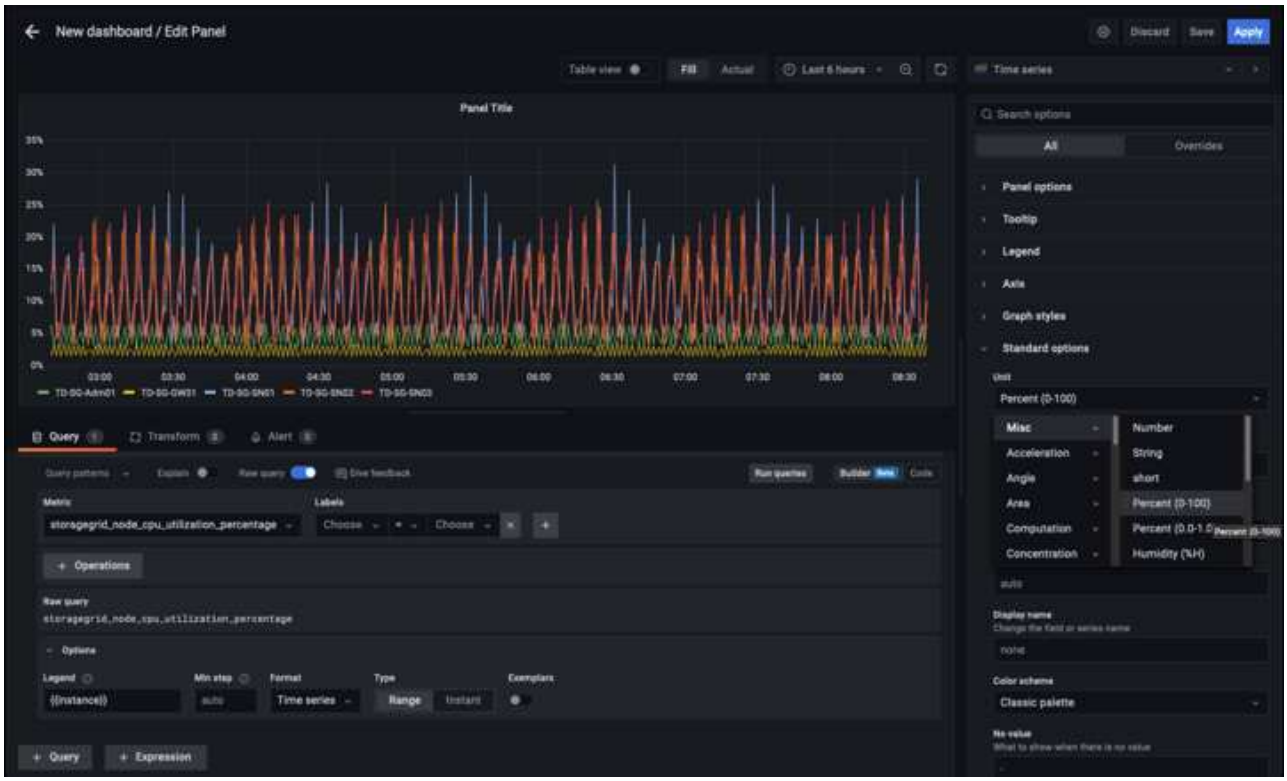
1. 左側のペインで[構成]を展開し、[データソース]を選択して、[データソースの追加]ボタンをクリックします
2. Prometheusは、最も人気のあるデータソースの1つです。検出されていない場合は、検索バーで「Prometheus」を特定します。
3. PrometheusインスタンスのURLとスクラビング間隔をPrometheusの間隔と一致するように入力して、Prometheusソースを設定します。Prometheusでアラートマネージャを設定しなかったため、アラートセクションも無効にしました。



4. 目的の設定を入力したら、下にスクロールして[保存してテスト]をクリックします。
5. 設定テストが正常に完了したら、[EXPLOR]ボタンをクリックします。
   a. 「調査」ウィンドウで、Prometheusで「storagegrid_node_name」に対してテストしたものと同じ指標を使用し、「Run query」ボタンをクリックします



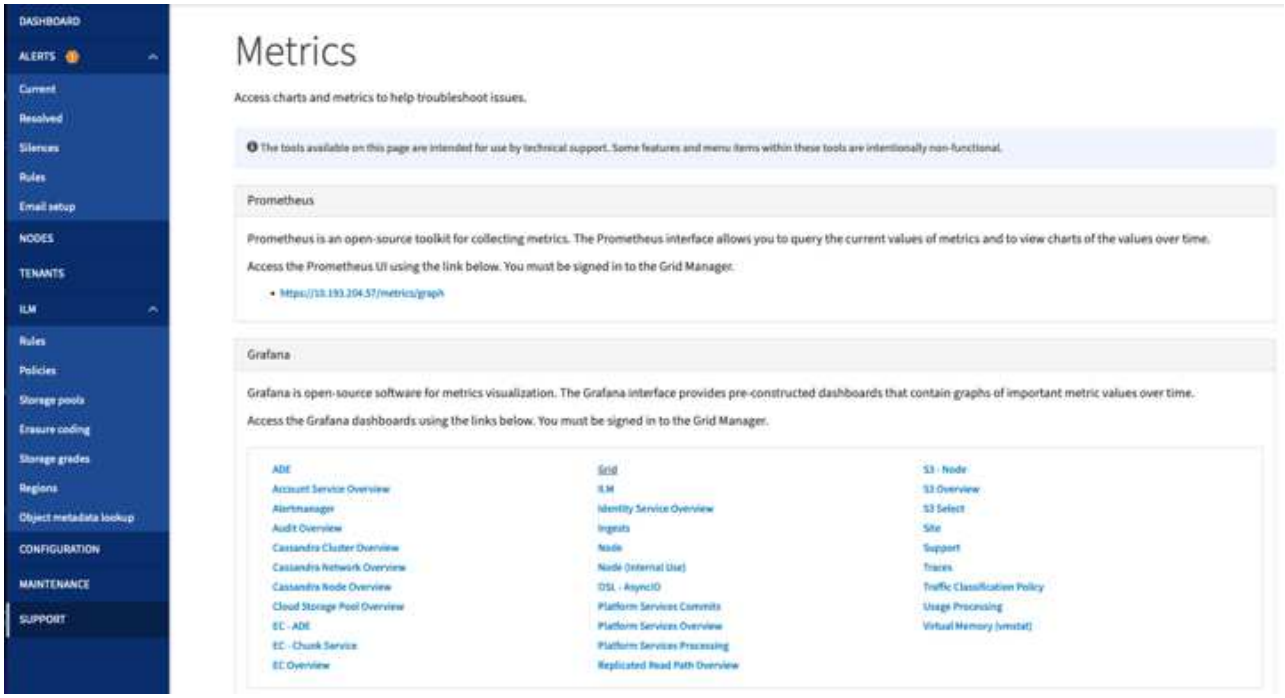
6. データソースを設定したら、ダッシュボードを作成します。
   a. 左側のペインで[ダッシュボード]を展開し、[+新しいダッシュボード]を選択します。
   b. 「新規パネルを追加」を選択します。
   c. メトリックを選択して新しいパネルを設定します。もう一度「storagegrid_node_name」を使用し、パネルのタイトルを入力し、下部に「Options」を展開して凡例をカスタムに変更し、「｛｛instance｝｝」と入力してノード名を定義します。右側のペインの「Standard options」set "Unit」を「Misc-100%」に設定します。[適用]をクリックして、パネルをダッシュボードに保存します。



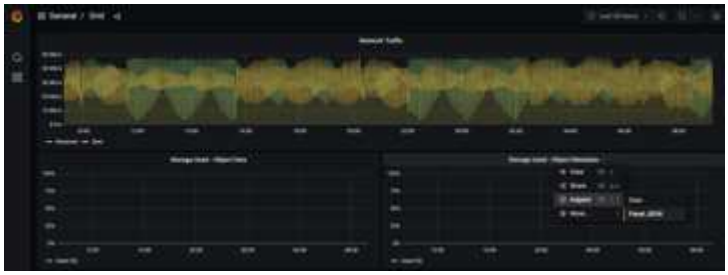
7. 必要な指標ごとにこのようなダッシュボードを構築し続けることもできますが、幸運にも、StorageGRID にはダッシュボードがすでに用意されており、カスタムダッシュボードにコピーすることができます。

   a. StorageGRID 管理インターフェイスの左側のペインで、[サポート]を選択し、[ツール]列の下部にある[指標]をクリックします。

   b. 指標内で、中央の列の上部にある「グリッド」リンクを選択します。

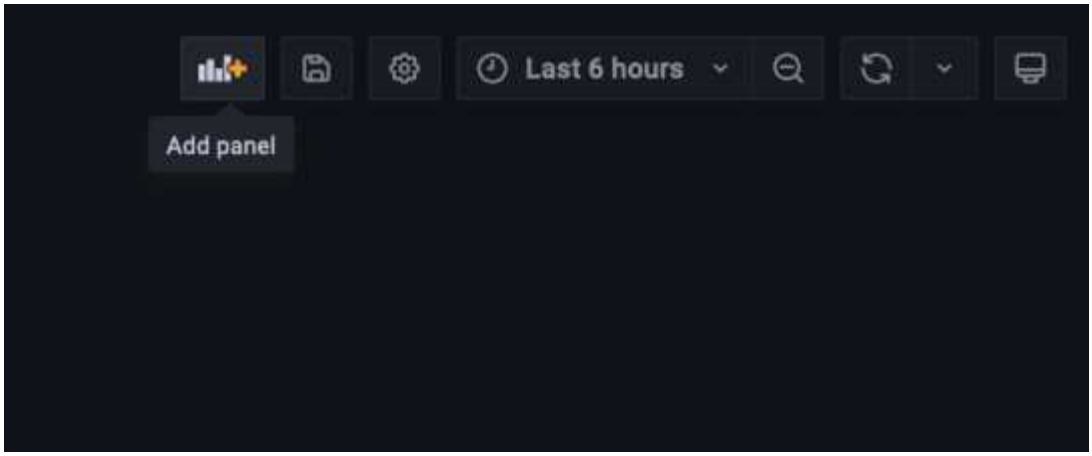


   c. グリッドダッシュボードで、「Storage Used - Object Metadata」パネルを選択します。メニューをドロップダウンするには、パネルタイトルの小さな下向き矢印と末尾をクリックします。このメニューから「Inspect」と「Panel JSON」を選択します。



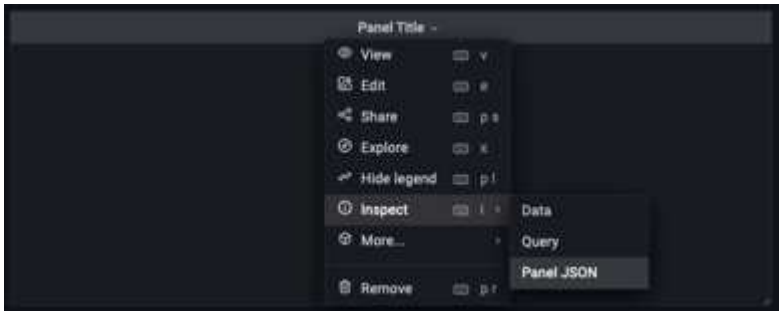
   d. JSONコードをコピーしてウィンドウを閉じます。

e. 新しいダッシュボードで、アイコンをクリックして新しいパネルを追加します。

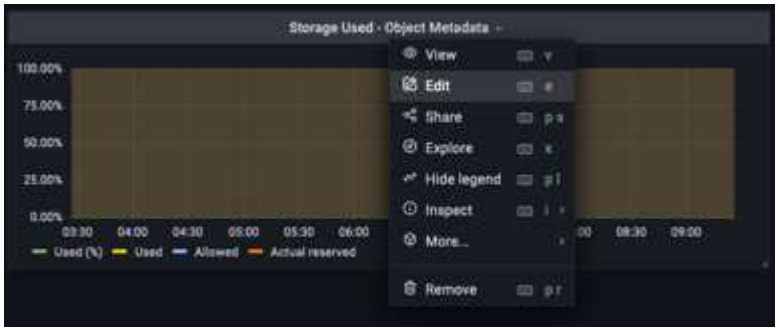


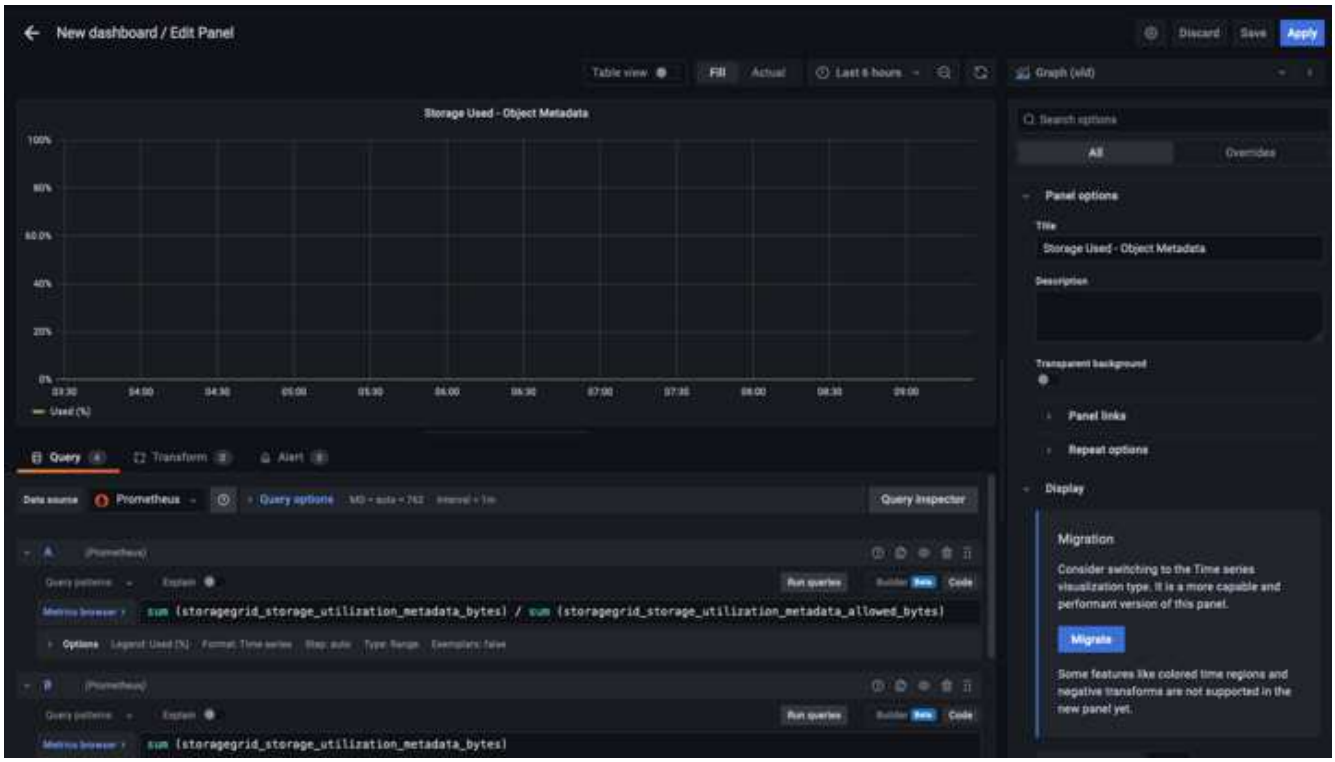
f. 変更を加えずに新しいパネルを適用します

g. StorageGRID パネルと同様に、JSONを確認します。JSONコードをすべて削除し、StorageGRID パネルからコピーしたコードに置き換えます。



h. 新しいパネルを編集すると、右側に「移行」ボタンを含む移行メッセージが表示されます。ボタンをクリックして、[適用]ボタンをクリックします。





8. すべてのパネルを所定の位置に配置し、必要に応じて構成したら、右上のディスクアイコンをクリックしてダッシュボードを保存し、名前を付けます。

まとめ

カスタマイズ可能なデータ保持機能とストレージ容量を備えたPrometheusサーバを導入しました。そのため、運用に最も関連性の高い指標を使用して独自のダッシュボードを構築し続けることができます。で収集されたPrometheus指標の詳細を確認できます "StorageGRID のドキュメント"。

アロンクライン著

Datadog SNMP構成

StorageGRID SNMPメトリクスとトラップを収集するようにDatadogを構成します。

Datadogを構成します

Datadogは、メトリクス、ビジュアライゼーション、アラートを提供する監視解決策 です。次の構成は、StorageGRID システムのローカルに配置されたUbuntu 22.04.1ホスト上のLinuxエージェントバージョン7.43.1で実装されました。

StorageGRID MIBファイルから生成されたDatadogプロファイルおよびトラップファイル

Datadogは、製品MIBファイルをSNMPメッセージのマッピングに必要なdatadog参照ファイルに変換する方法を提供します。

見つかった命令に従って生成されたDatadogトラップ解決マッピング用のこのStorageGRID YAMLファイル "こちらをご覧ください"。+このファイルを/etc/datadog-agent/conf.d/snmp.d/traps_db/+に配置します

- "トラップYAMLファイルをダウンロードします" [+]
  - * MD5チェックサム* 42e27e4210719945a46172b98c379517以降
  - * SHA256チェックサム* d0fe5c8e6ca3c902d054f854b70a85f928cba8b7c76391d356f05d2cf73b6887以降

見つかった命令に従って生成されたDatadogメトリクスマッピング用のこのStorageGRID プロファイルYAMLファイル "こちらをご覧ください"。+このファイルを/etc/datadog-agent/conf.d/snmp.d/profiles/+に配置します

- "プロファイルYAMLファイルをダウンロードします" [+]
  - * MD5チェックサム* 72bb7784f4801adda4e0c3ea77df19aa+
  - * SHA256チェックサム* b6b7fadd33063422a8bb8e39b3ead8ab349ee0229926eadc8585f0087b8cee+

メトリクスの**SNMP Datadog**構成

メトリックのSNMPの設定は、2つの方法で管理できます。自動検出を設定するには、StorageGRID システムを含むネットワークアドレス範囲を指定するか、個 々 のデバイスのIPを定義します。設定の場所は、決定内容によって異なります。自動検出は、datadogエージェントのYAMLファイルで定義されます。明示的なデバイス定義は、SNMP設定YAMLファイルで設定されます。以下に、同じStorageGRID システムのそれぞれの例を示します。

自動検出

設定は/etc/datadog-agent/datadog.yamlにあります

```
listeners:
  - name: snmp
snmp_listener:
  workers: 100  # number of workers used to discover devices concurrently
  discovery_interval: 3600  # interval between each autodiscovery in
seconds
  loader: core  # use core check implementation of SNMP integration.
recommended
  use_device_id_as_hostname: true  # recommended
  configs:
    - network_address: 10.0.0.0/24  # CIDR subnet
      snmp_version: 2
      port: 161
      community_string: 'st0r@gegrid'  # enclose with single quote
      profile: netapp-storagegrid
```

個 々 のデバイス

/etc/datadog-agent/conf.d/snmp.d/conf.yaml

```
init_config:
  loader: core  # use core check implementation of SNMP integration.
recommended
  use_device_id_as_hostname: true  # recommended
instances:
- ip_address: '10.0.0.1'
  profile: netapp-storagegrid
  community_string: 'st0r@gegrid'  # enclose with single quote
- ip_address: '10.0.0.2'
  profile: netapp-storagegrid
  community_string: 'st0r@gegrid'
- ip_address: '10.0.0.3'
  profile: netapp-storagegrid
 community_string: 'st0r@gegrid'
- ip_address: '10.0.0.4'
  profile: netapp-storagegrid
  community_string: 'st0r@gegrid'
```

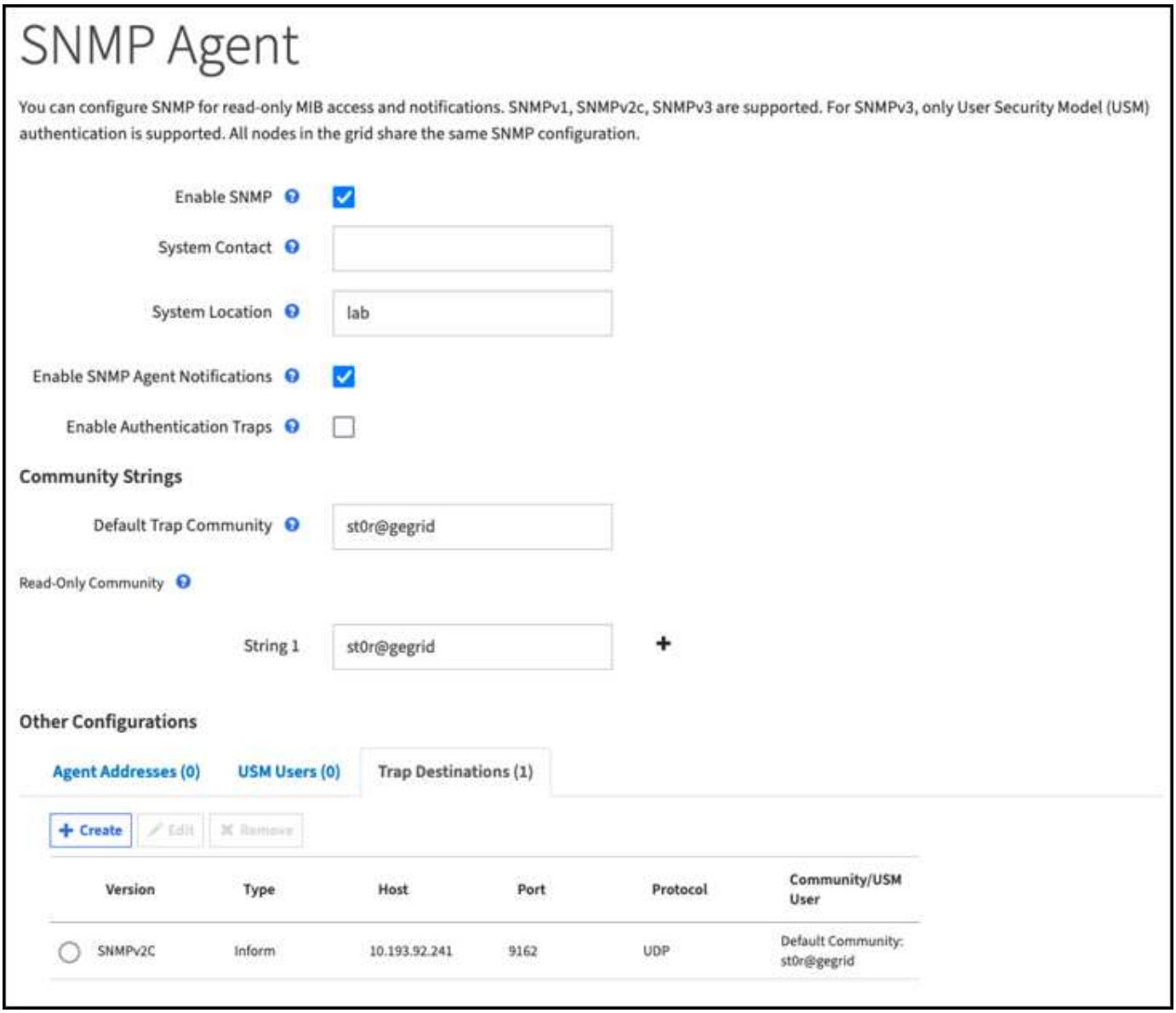
トラップの**SNMP**設定

SNMPトラップの構成は、datadog構成YAMLファイル/etc/datadog-agent/datadog.yamlで定義されています

```
network_devices:
  namespace:  # optional, defaults to “default”.
  snmp_traps:
    enabled: true
    port: 9162 # on which ports to listen for traps
    community_strings: # which community strings to allow for v2 traps
      - st0r@gegrid
```

StorageGRID の**SNMP**設定例

StorageGRID システムのSNMPエージェントは、[Configuration]タブの[Monitoring]列にあります。SNMPを有効にし、必要な情報を入力します。トラップを構成する場合は、[Traps Destinations]を選択し、トラップ構成を含むDatadogエージェントホストの宛先を作成します。



アロンクライン著

rcloneを使用して、StorageGRID 上のオブジェクトを移行、PUT、および削除します

rcloneは、S3処理用の無料のコマンドラインツールでクライアントです。rcloneを使用して、StorageGRID 上のオブジェクトデータを移行、コピー、および削除できます。rcloneには、次の例に示すように、「purge」機能を使用して空でなくてもバケットを削除する機能が含まれています。

rcloneをインストールして設定します

rcloneをワークステーションまたはサーバにインストールするには、からダウンロードします "rclone.org"。

初期設定手順

1. 設定スクリプトを実行するか、ファイルを手動で作成して、rclone構成ファイルを作成します。
2. この例では、rclone構成のリモートStorageGRID S3エンドポイントの名前にsgdemoを使用します。
   a. 設定ファイル~/.config/rclone/rclone.confを作成します

```
[sgdemo]
type = s3
provider = Other
access_key_id = ABCDEFGH123456789JKL
secret_access_key = 123456789ABCDEFGHIJKLMN0123456789PQRST+V
endpoint = sgdemo.netapp.com
```

   b. rclone configを実行します

#rclone設定

```
2023/04/13 14:22:45 NOTICE: Config file
"/root/.config/rclone/rclone.conf" not found - using defaults
No remotes found - make a new one
n) New remote
s) Set configuration password
q) Quit config
n/s/q> n
name> sgdemo
```

```
Option Storage.
Type of storage to configure.
Enter a string value. Press Enter for the default ("").
Choose a number from below, or type in your own value.
 1 / 1Fichier
   \ "fichier"
 2 / Alias for an existing remote
   \ "alias"
 3 / Amazon Drive
   \ "amazon cloud drive"
 4 / Amazon S3 Compliant Storage Providers including AWS,
Alibaba, Ceph, Digital Ocean, Dreamhost, IBM COS, Minio,
SeaweedFS, and Tencent COS
   \ "s3"
 5 / Backblaze B2
   \ "b2"
 6 / Better checksums for other remotes
   \ "hasher"
 7 / Box
   \ "box"
 8 / Cache a remote
   \ "cache"
 9 / Citrix Sharefile
   \ "sharefile"
10 / Compress a remote
   \ "compress"
11 / Dropbox
   \ "dropbox"
12 / Encrypt/Decrypt a remote
   \ "crypt"
13 / Enterprise File Fabric
   \ "filefabric"
14 / FTP Connection
```

```
   \ "ftp"
15 / Google Cloud Storage (this is not Google Drive)
   \ "google cloud storage"
16 / Google Drive
   \ "drive"
17 / Google Photos
   \ "google photos"
18 / Hadoop distributed file system
   \ "hdfs"
19 / Hubic
   \ "hubic"
20 / In memory object storage system.
   \ "memory"
21 / Jottacloud
   \ "jottacloud"
22 / Koofr
   \ "koofr"
23 / Local Disk
   \ "local"
24 / Mail.ru Cloud
   \ "mailru"
25 / Mega
   \ "mega"
26 / Microsoft Azure Blob Storage
   \ "azureblob"
27 / Microsoft OneDrive
   \ "onedrive"
28 / OpenDrive
   \ "opendrive"
29 / OpenStack Swift (Rackspace Cloud Files, Memset Memstore,
OVH)
   \ "swift"
30 / Pcloud
   \ "pcloud"
31 / Put.io
   \ "putio"
32 / QingCloud Object Storage
   \ "qingstor"
33 / SSH/SFTP Connection
   \ "sftp"
34 / Sia Decentralized Cloud
   \ "sia"
35 / Sugarsync
   \ "sugarsync"
36 / Tardigrade Decentralized Cloud Storage
   \ "tardigrade"
```

```
37 / Transparently chunk/split large files
   \ "chunker"
38 / Union merges the contents of several upstream fs
   \ "union"
39 / Uptobox
   \ "uptobox"
40 / Webdav
   \ "webdav"
41 / Yandex Disk
   \ "yandex"
42 / Zoho
   \ "zoho"
43 / http Connection
   \ "http"
44 / premiumize.me
   \ "premiumizeme"
45 / seafile
   \ "seafile"
```

```
Storage> 4
```

```
Option provider.
Choose your S3 provider.
Enter a string value. Press Enter for the default ("").
Choose a number from below, or type in your own value.
 1 / Amazon Web Services (AWS) S3
   \ "AWS"
 2 / Alibaba Cloud Object Storage System (OSS) formerly Aliyun
   \ "Alibaba"
 3 / Ceph Object Storage
   \ "Ceph"
 4 / Digital Ocean Spaces
   \ "DigitalOcean"
 5 / Dreamhost DreamObjects
   \ "Dreamhost"
 6 / IBM COS S3
   \ "IBMCOS"
 7 / Minio Object Storage
   \ "Minio"
 8 / Netease Object Storage (NOS)
   \ "Netease"
 9 / Scaleway Object Storage
   \ "Scaleway"
10 / SeaweedFS S3
   \ "SeaweedFS"
11 / StackPath Object Storage
   \ "StackPath"
12 / Tencent Cloud Object Storage (COS)
   \ "TencentCOS"
13 / Wasabi Object Storage
   \ "Wasabi"
14 / Any other S3 compatible provider
   \ "Other"
provider> 14
```

```
Option env_auth.
Get AWS credentials from runtime (environment variables or
EC2/ECS meta data if no env vars).
Only applies if access_key_id and secret_access_key is blank.
Enter a boolean value (true or false). Press Enter for the
default ("false").
Choose a number from below, or type in your own value.
 1 / Enter AWS credentials in the next step.
   \ "false"
 2 / Get AWS credentials from the environment (env vars or IAM).
   \ "true"
env_auth> 1
```

```
Option access_key_id.
AWS Access Key ID.
Leave blank for anonymous access or runtime credentials.
Enter a string value. Press Enter for the default ("").
access_key_id> ABCDEFGH123456789JKL
```

```
Option secret_access_key.
AWS Secret Access Key (password).
Leave blank for anonymous access or runtime credentials.
Enter a string value. Press Enter for the default ("").
secret_access_key> 123456789ABCDEFGHIJKLMN0123456789PQRST+V
```

```
Option region.
Region to connect to.
Leave blank if you are using an S3 clone and you don't have a
region.
Enter a string value. Press Enter for the default ("").
Choose a number from below, or type in your own value.
   / Use this if unsure.
 1 | Will use v4 signatures and an empty region.
   \ ""
   / Use this only if v4 signatures don't work.
 2 | E.g. pre Jewel/v10 CEPH.
   \ "other-v2-signature"
region> 1
```

```
Option endpoint.
Endpoint for S3 API.
Required when using an S3 clone.
Enter a string value. Press Enter for the default ("").
endpoint> sgdemo.netapp.com
```

```
Option location_constraint.
Location constraint - must be set to match the Region.
Leave blank if not sure. Used when creating buckets only.
Enter a string value. Press Enter for the default ("").
location_constraint>
```

```
Option acl.
Canned ACL used when creating buckets and storing or copying
objects.
This ACL is used for creating objects and if bucket_acl isn't
set, for creating buckets too.
For more info visit
https://docs.aws.amazon.com/AmazonS3/latest/dev/acl-
overview.html#canned-acl
Note that this ACL is applied when server-side copying objects as
S3
doesn't copy the ACL from the source but rather writes a fresh
one.
Enter a string value. Press Enter for the default ("").
Choose a number from below, or type in your own value.
   / Owner gets FULL_CONTROL.
 1 | No one else has access rights (default).
   \ "private"
   / Owner gets FULL_CONTROL.
 2 | The AllUsers group gets READ access.
   \ "public-read"
   / Owner gets FULL_CONTROL.
 3 | The AllUsers group gets READ and WRITE access.
   | Granting this on a bucket is generally not recommended.
   \ "public-read-write"
   / Owner gets FULL_CONTROL.
 4 | The AuthenticatedUsers group gets READ access.
   \ "authenticated-read"
   / Object owner gets FULL_CONTROL.
 5 | Bucket owner gets READ access.
   | If you specify this canned ACL when creating a bucket,
Amazon S3 ignores it.
   \ "bucket-owner-read"
   / Both the object owner and the bucket owner get FULL_CONTROL
over the object.
 6 | If you specify this canned ACL when creating a bucket,
Amazon S3 ignores it.
   \ "bucket-owner-full-control"
acl>
```

```
Edit advanced config?
y) Yes
n) No (default)
y/n> n
```

```
--------------------
[sgdemo]
type = s3
provider = Other
access_key_id = ABCDEFGH123456789JKL
secret_access_key = 123456789ABCDEFGHIJKLMN0123456789PQRST+V
endpoint = sgdemo.netapp.com:443
--------------------
y) Yes this is OK (default)
e) Edit this remote
d) Delete this remote
y/e/d>
```

```
Current remotes:
```

```
Name                 Type
====                 ====
sgdemo               s3
```

```
e) Edit existing remote
n) New remote
d) Delete remote
r) Rename remote
c) Copy remote
s) Set configuration password
q) Quit config
e/n/d/r/c/s/q> q
```

基本的なコマンドの例

- バケットを作成：

  `rclone mkdir remote:bucket`

  #rclone mkdir sgdemo：test01

  SSL証明書を無視する必要がある場合は、--no-check-certificateを使用します。

- すべてのバケットを表示：

```
rclone lsd remote:
```

#rclone lsd sgdemo：

- 特定のバケット内のオブジェクトをリストします。

```
rclone ls remote:bucket
```

rclone ls sgdemo：test01

```
     65536 TestObject.0
     65536 TestObject.1
     65536 TestObject.10
     65536 TestObject.12
     65536 TestObject.13
     65536 TestObject.14
     65536 TestObject.15
     65536 TestObject.16
     65536 TestObject.17
     65536 TestObject.18
     65536 TestObject.2
     65536 TestObject.3
     65536 TestObject.5
     65536 TestObject.6
     65536 TestObject.7
     65536 TestObject.8
     65536 TestObject.9
   33554432 bigobj
       102 key.json
        47 locked01.txt
 4294967296 sequential-read.0.0
        15 test.txt
       116 version.txt
```

- バケットを削除：

```
rclone rmdir remote:bucket
```

#rclone rmdir sgdemo：test02

- オブジェクトを置きなさい:

```
rclone copy filename remote:bucket
```

#rclone copy ~/test/ testfile.txt sgdemo：test01

- オブジェクトを取得：

```
rclone copy remote:bucket/objectname filename
```

#rclone copy sgdemo：test01 / testfile.txt ~/test/ testfileS3.txt

- オブジェクトを削除：

```
rclone delete remote:bucket/objectname
```

#rclone delete sgdemo：test01 / testfile.txt

- バケット内のオブジェクトの移行

```
rclone sync source:bucket destination:bucket --progress

rclone sync source_directory destination:bucket --progress
```

#rclone sync sgdemo：test01 sgdemo：clone01 — progress

```
Transferred:        4.032 GiB / 4.032 GiB, 100%, 95.484 KiB/s, ETA
0s
Transferred:           22 / 22, 100%
Elapsed time:       1m4.2s
```

— progressまたは- Pを使用して、タスクの進行状況を表示します。それ以外の場合、出力はありません。

- バケットとすべてのオブジェクトコンテンツを削除する

```
rclone purge remote:bucket --progress
```

#rclone purge sgdemo：test01 — progress

```
Transferred:              0 B / 0 B, -, 0 B/s, ETA -
Checks:                46 / 46, 100%
Deleted:               23 (files), 1 (dirs)
Elapsed time:        10.2s
```

rclone ls sgdemo：test01

```
2023/04/14 09:40:51 Failed to ls: directory not found
```

ジークフリート・ヘップとアロン・クライン著_

Veeam Backup & Replicationを使用した導入に関するStorageGRIDのベストプラクティス

このガイドでは、NetApp StorageGRIDの構成と、Veeam Backup & Replicationの一部を中心に説明します。本ドキュメントは、Linuxシステムに精通し、Veeam Backup & Replicationと組み合わせてNetApp StorageGRIDシステムの保守または実装を担当するストレージ管理者およびネットワーク管理者を対象としています。

概要

ストレージ管理者は、可用性、迅速なリカバリの目標を達成し、ニーズに合わせて拡張し、データの長期保存に関するポリシーを自動化するソリューションを使用して、データの増加を管理したいと考えています。これらのソリューションは、損失や悪意のある攻撃からも保護する必要があります。VeeamとNetAppは提携して、オンプレミスのオブジェクトストレージ向けのVeeam Backup & RecoveryとNetApp StorageGRIDを組み合わせたデータ保護解決策を作成しました。

VeeamとNetApp StorageGRIDが連携して動作する使いやすい解決策を提供することで、急速なデータ量の増大や世界的な規制強化のニーズに対応できます。クラウドベースのオブジェクトストレージは、耐障害性、拡張性、運用効率、コスト効率に優れていることで知られており、バックアップのターゲットとして最適です。本ドキュメントでは、Veeam Backup解決策およびStorageGRIDシステムの構成に関するガイダンスと推奨事項を提供します。

Veeamのオブジェクトワークロードによって、小規模オブジェクトのPUT、DELETE、LIST処理が同時に多数作成されます。書き換えや削除の防止を有効にすると、保持期間の設定やバージョンの表示に関する要求がオブジェクトストアに追加されます。バックアップジョブのプロセスでは、日次変更のためにオブジェクトが書き込まれます。その後、新しい書き込みが完了すると、バックアップの保持ポリシーに基づいてオブジェクトが削除されます。バックアップジョブのスケジュールは、ほとんどの場合重複します。その結果、バックアップウィンドウの大部分がオブジェクトストアに50分の50のPUT / DELETEワークロードで構成されます。タスクスロットの設定を使用して同時処理数をVeeamで調整し、バックアップジョブのブロックサイズを増やしてオブジェクトサイズを増やし、複数オブジェクトの削除要求に含まれるオブジェクト数を減らします。また、ジョブを完了する最大期間を選択することで、解決策のパフォーマンスとコストが最適化されます。

次の製品ドキュメントを参照してください： "Veeam Backup Replication" および "StorageGRID" 始める前に。Veeamには、StorageGRID 解決策 をサイジングする前に使用する必要があるVeeamインフラのサイジングと容量の要件を把握するための計算ツールが用意されています。Veeam Ready ProgramのWebサイトで、Veeamとネットアップによる検証済みの構成について "Veeam Readyのオブジェクト、オブジェクトの変更不可、リポジトリ"。

Veeam構成

推奨バージョン

常に最新の状態に保ち、Veeam Backup & Replication 12システムの最新のホットフィックスを適用することをお勧めします。現在、少なくともVeeamパッチP20230718のインストールを推奨しています。

S3リポジトリ設定

スケールアウトバックアップリポジトリ（SOBR）は、S3オブジェクトストレージの大容量階層です。大容量階層はプライマリリポジトリを拡張したもので、データ保持期間が長くなり、ストレージ解決策が低コストになります。Veeamには、S3 Object Lock APIを通じて不変性を提供する機能があります。Veeam 12では、スケールアウトリポジトリで複数のバケットを使用できます。StorageGRIDでは、1つのバケット内のオブジェクト数や容量に制限はありません。複数のバケットを使用すると、オブジェクトのバックアップデータがペタバイト規模になる可能性がある非常に大規模なデータセットをバックアップする際のパフォーマンスが向上する可能性があります。

特定の解決策のサイジングと要件によっては、同時に実行できるタスクを制限する必要があります。デフォルト設定では、CPUコアごとに1つのリポジトリタスクスロットを指定し、タスクスロットごとに最大64の同時タスクスロットを指定します。たとえば、サーバに2つのCPUコアがある場合、オブジェクトストアには合計128個の同時スレッドが使用されます。これには、PUT、GET、およびBATCH Deleteが含まれます。タスクスロットに控えめな制限を選択して開始し、Veeamバックアップが新しいバックアップの安定した状態と期限切れになるバックアップ・データに達したら、この値を調整することをお勧めします。NetAppアカウントチームと協力して、希望する時間枠とパフォーマンスに合わせてStorageGRIDシステムを適切にサイジングしてください。最適な解決策を提供するには、タスクスロットの数とスロットあたりのタスクの制限を調整する必要がある場合があります。

バックアップジョブの設定

Veeamバックアップジョブでは、さまざまなブロックサイズオプションを設定できますが、これらは慎重に検討する必要があります。デフォルトのブロックサイズは1MBで、Veeamの圧縮機能と重複排除機能を使用すると、最初のフルバックアップでは約500KB、増分ジョブでは100200KBのオブジェクトが作成されます。バックアップブロックサイズを大きくすることで、パフォーマンスを大幅に向上し、オブジェクトストレージの要件を縮小できます。ブロックサイズが大きいほどオブジェクトストアのパフォーマンスは大幅に向上しますが、ストレージ効率のパフォーマンスが低下するため、プライマリストレージの容量要件が増大する可能性があります。バックアップジョブのブロックサイズを4MBに設定することを推奨します。この場合、フルバックアップ用に約2MBのオブジェクトが作成され、増分バックアップ用に700KB1MBのオブジェクトサイズが作成されます。お客様は、8 MBのブロックサイズを使用してバックアップジョブを構成することも検討できます。これは、Veeamサポートの支援を受けて有効にすることができます。

変更不可のバックアップの実装では、オブジェクトストアのS3オブジェクトロックが使用されます。immutabilityオプションを指定すると、オブジェクトに対するリストおよび保持の更新要求がオブジェクトストアに対して生成される回数が増加します。

バックアップの保持期間が終了すると、バックアップジョブによってオブジェクトの削除が処理されます。Veeamは、1回の要求につき1、000個のオブジェクトを含む複数のオブジェクトの削除要求で、オブジェクトストアに削除要求を送信します。小規模なソリューションの場合は、リクエストあたりのオブジェクト数を減らすために調整が必要になることがあります。この値を小さくすると、削除要求がStorageGRIDシステム内のノードに均等に分散されるというメリットもあります。複数オブジェクトの削除制限を設定する場合は、次の表の値を開始点として使用することをお勧めします。表の値に選択したアプライアンスタイプのノード数

を掛けて、Veeamの設定値を取得します。この値が1000以上の場合、デフォルト値を調整する必要はありません。この値を調整する必要がある場合は、Veeamサポートに連絡して変更を行ってください。

アプライアンスモデル	ノードあたりの**S3MultiObjectDeleteLimit**
SG5712	34
SG5760	七五
SG6060 の設計	200です

お客様固有のニーズに基づいた推奨構成については、NetAppアカウントチームにお問い合わせください。Veeamの設定に関する推奨事項は次のとおりです。

- バックアップジョブのブロックサイズ= 4MB
- SOBRタスクスロット制限=2-16
- 複数オブジェクトの削除制限= 34-1000

StorageGRID構成

推奨バージョン

Veeam環境に推奨されるバージョンは、最新のホットフィックスが適用されたNetApp StorageGRID 11.6または11.7です。StorageGRID 11.6.0.11および11.7.0.4では、Veeamのワークロードに役立つ最適化機能が多数導入されました。常に最新の状態に保ち、StorageGRIDシステムに最新のホットフィックスを適用することを推奨します。

ロードバランサと**S3**エンドポイントの設定

Veeamでは、エンドポイントの接続にHTTPSのみを使用する必要があります。暗号化されていない接続はVeeamではサポートされていません。SSL証明書には、自己署名証明書、信頼されたプライベート認証局、または信頼されたパブリック認証局を使用できます。S3リポジトリへの継続的なアクセスを確保するために、HA構成で少なくとも2つのロードバランサを使用することを推奨します。ロードバランサには、すべての管理ノードとゲートウェイノードに配置されるStorageGRID提供の統合ロードバランササービス、またはF5、Kemp、HAProxy、Loadbalanacer.orgなどのサードパーティの解決策を使用できます。 StorageGRIDロードバランサを使用すると、Veeamのワークロードに優先順位を付けたり、StorageGRIDシステムの優先順位の高いワークロードに影響しないようにVeeamを制限したりできるトラフィック分類機能（QoSルール）を設定できます。

S3 バケット

StorageGRIDは、セキュアなマルチテナントストレージシステムです。Veeamワークロード専用のテナントを作成することを推奨します。ストレージクォータはオプションで割り当てることができます。ベストプラクティスとして、「独自のアイデンティティソースを使用する」を有効にします。テナントのroot管理ユーザを適切なパスワードで保護します。Veeam Backup 12では、S3バケットに対して強い整合性が必要です。StorageGRIDには、バケットレベルで設定できる複数の整合性オプションが用意されています。Veeamが複数の場所のデータにアクセスするマルチサイト環境の場合は、[strong-global]を選択します。Veeamのバックアップとリストアを単一サイトでのみ実行する場合は、整合性レベルを「strong-site」に設定する必要があります。バケットの整合性レベルの詳細については、 "ドキュメント"。Veeamの書き換え不可のバックアップにStorageGRIDを使用するには、S3オブジェクトロックをグローバルに有効にし、バケットの作成時にバケットで設定する必要があります。

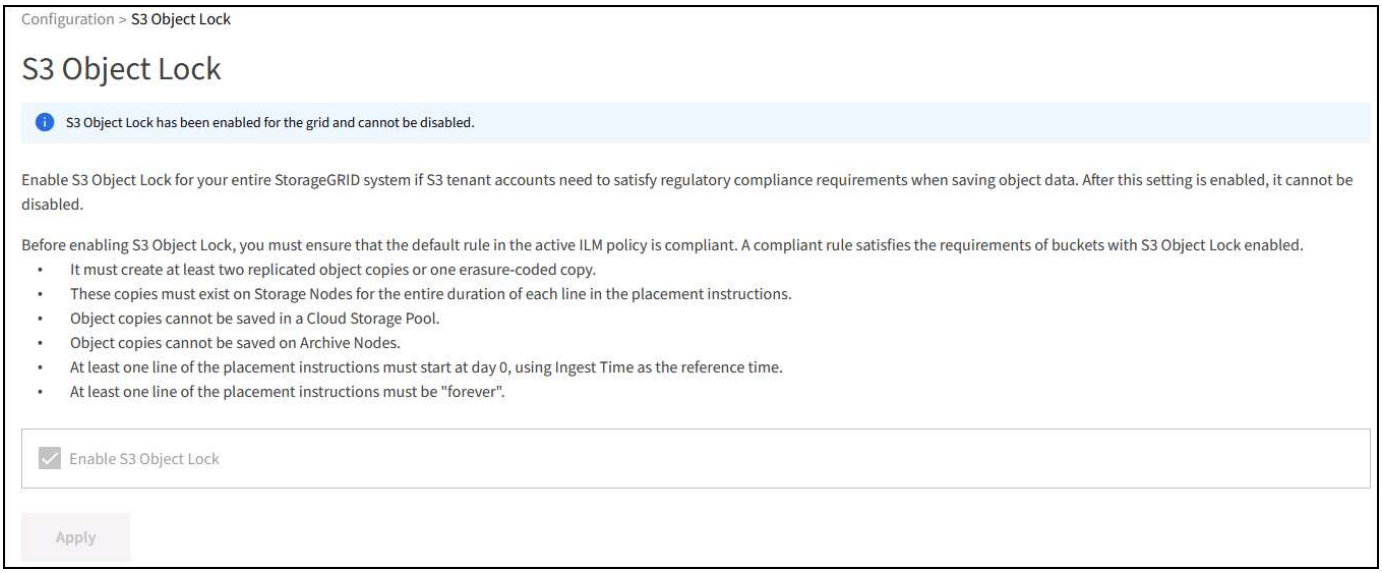
ライフサイクル管理

StorageGRIDは、レプリケーションとイレイジャーコーディングをサポートして、StorageGRIDのノードとサイト全体でオブジェクトレベルの保護を実現します。イレイジャーコーディングには、オブジェクトサイズが200KB以上必要です。Veeamのデフォルトのブロックサイズである1MBで作成されるオブジェクトサイズは、VeeamのStorage Efficiency機能と比較して、この200KBの推奨最小サイズよりも小さくなることがあります。解決策のパフォーマンスを高めるために、サイト間の接続が十分でない場合やStorageGRIDシステムの帯域幅が制限されない場合を除き、複数のサイトにまたがるイレイジャーコーディングプロファイルを使用することは推奨されません。マルチサイトStorageGRIDシステムでは、各サイトにコピーを1つ格納するようにILMルールを設定できます。データの保持性を最大限に高めるために、各サイトにイレイジャーコーディングコピーを格納するルールを設定できます。このワークロードには、Veeam Backupサーバのローカルコピーを2つ使用することを推奨します。

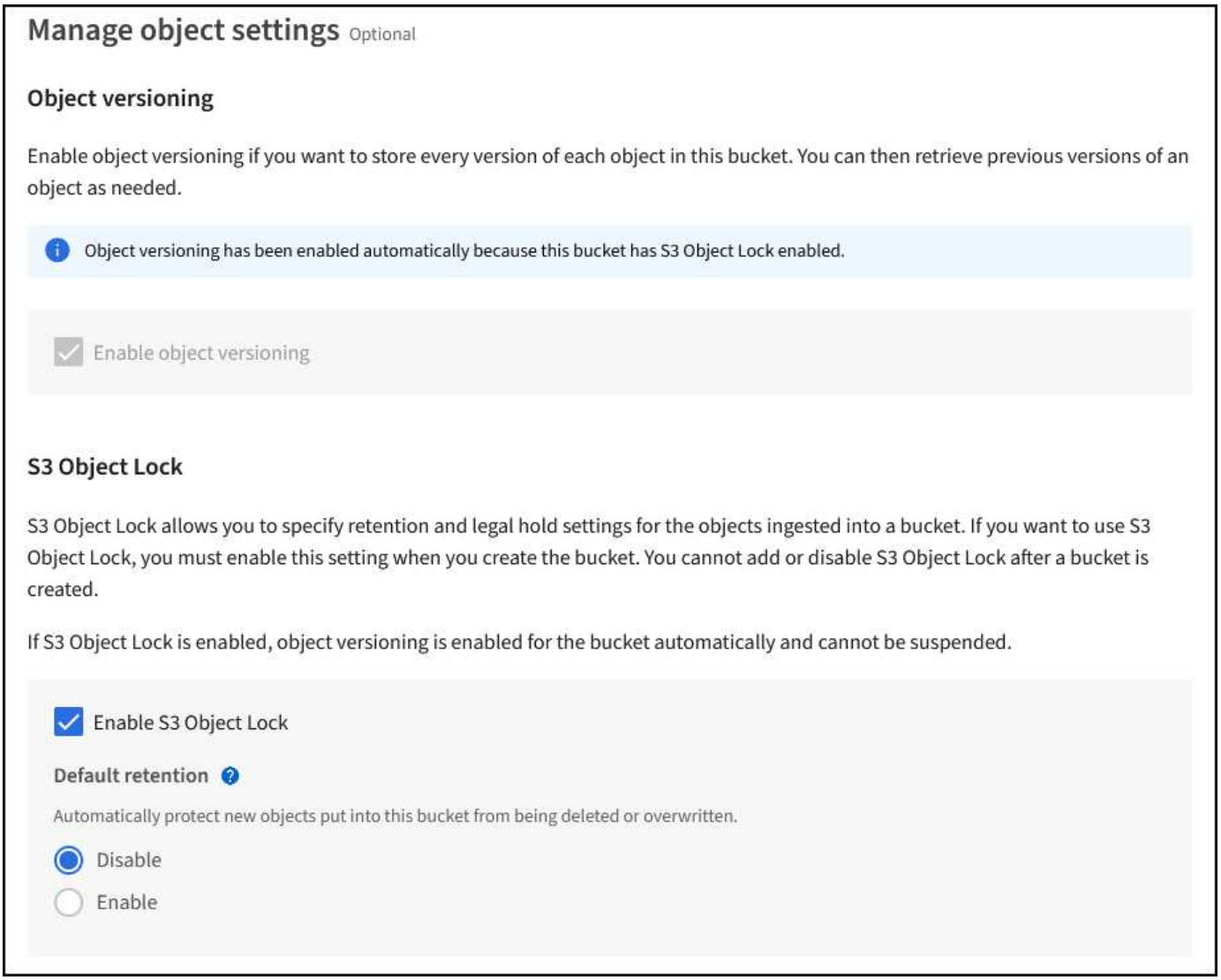
導入のキーポイント

StorageGRID

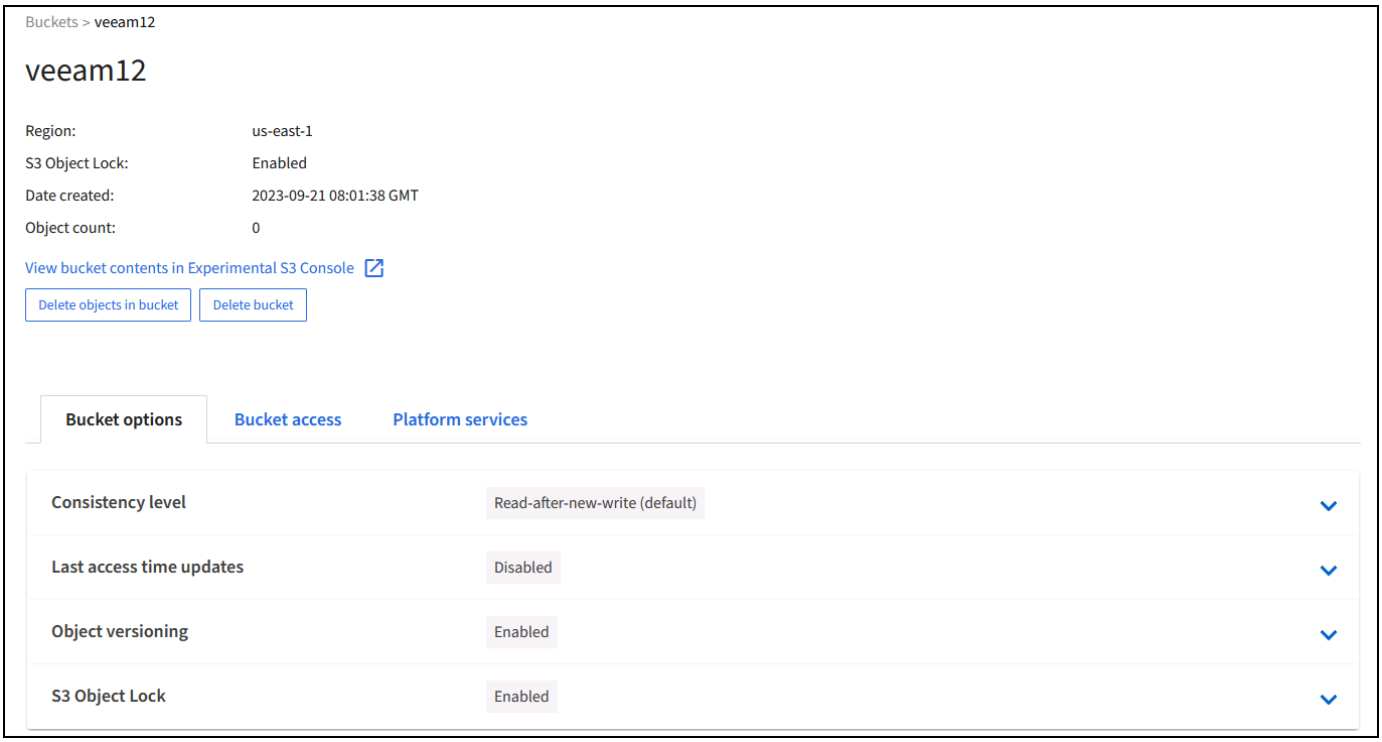
不変性が必要な場合は、StorageGRIDシステムでオブジェクトロックが有効になっていることを確認します。管理UIの[Configuration]/[S3][Object Lock]にあるオプションを選択します。



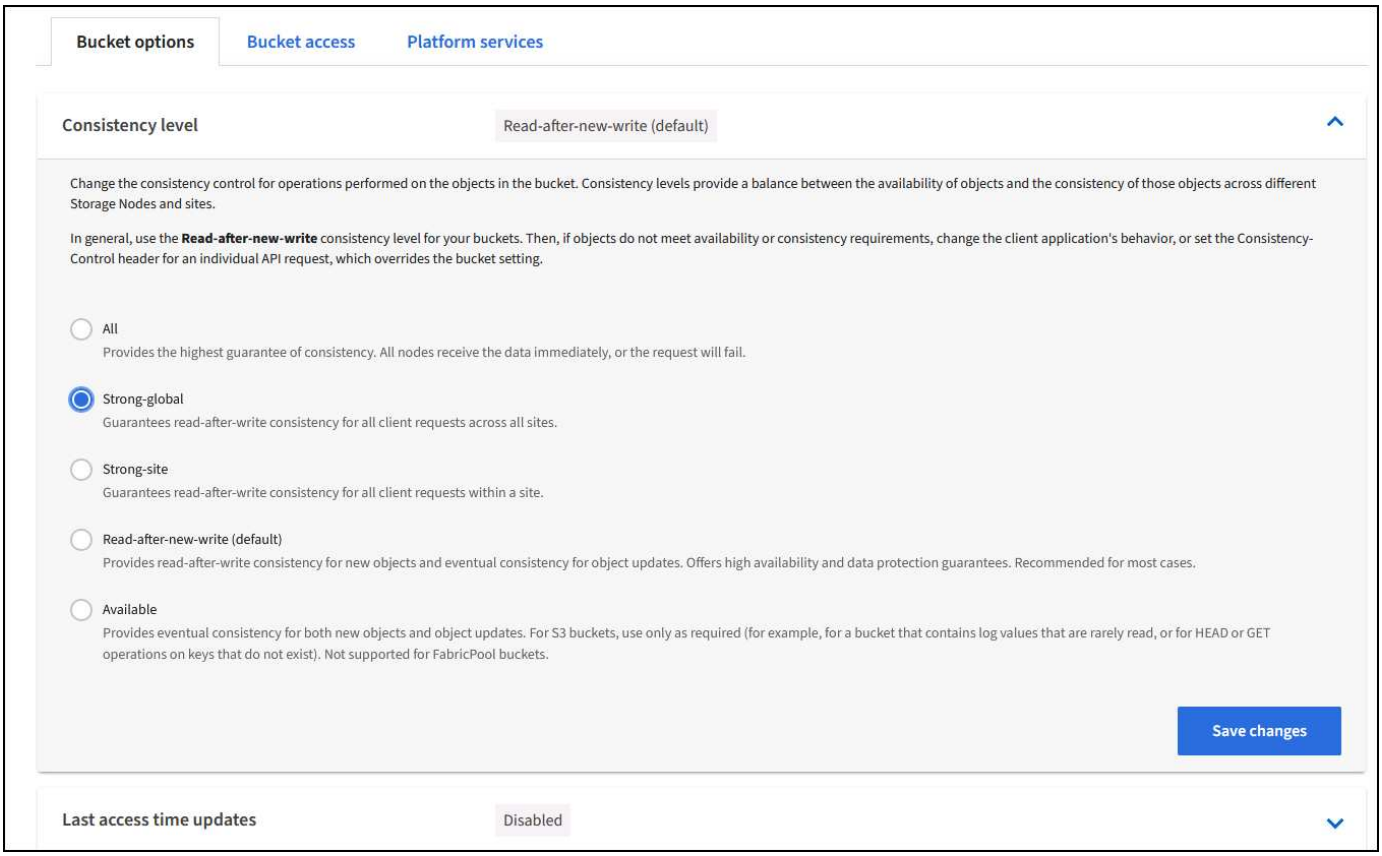
バケットを変更不可のバックアップに使用する場合は、バケットの作成時に[Enable S3 Object Lock]を選択します。これにより、バケットのバージョン管理が自動的に有効になります。オブジェクト保持期間はVeeamによって明示的に設定されるため、デフォルトの保持期間は無効のままにします。Veeamで変更不可のバックアップが作成されていない場合は、[Versioning]と[S3 Object Lock]を選択しないでください。



バケットが作成されたら、作成したバケットの詳細ページに移動します。整合性レベルを選択します。



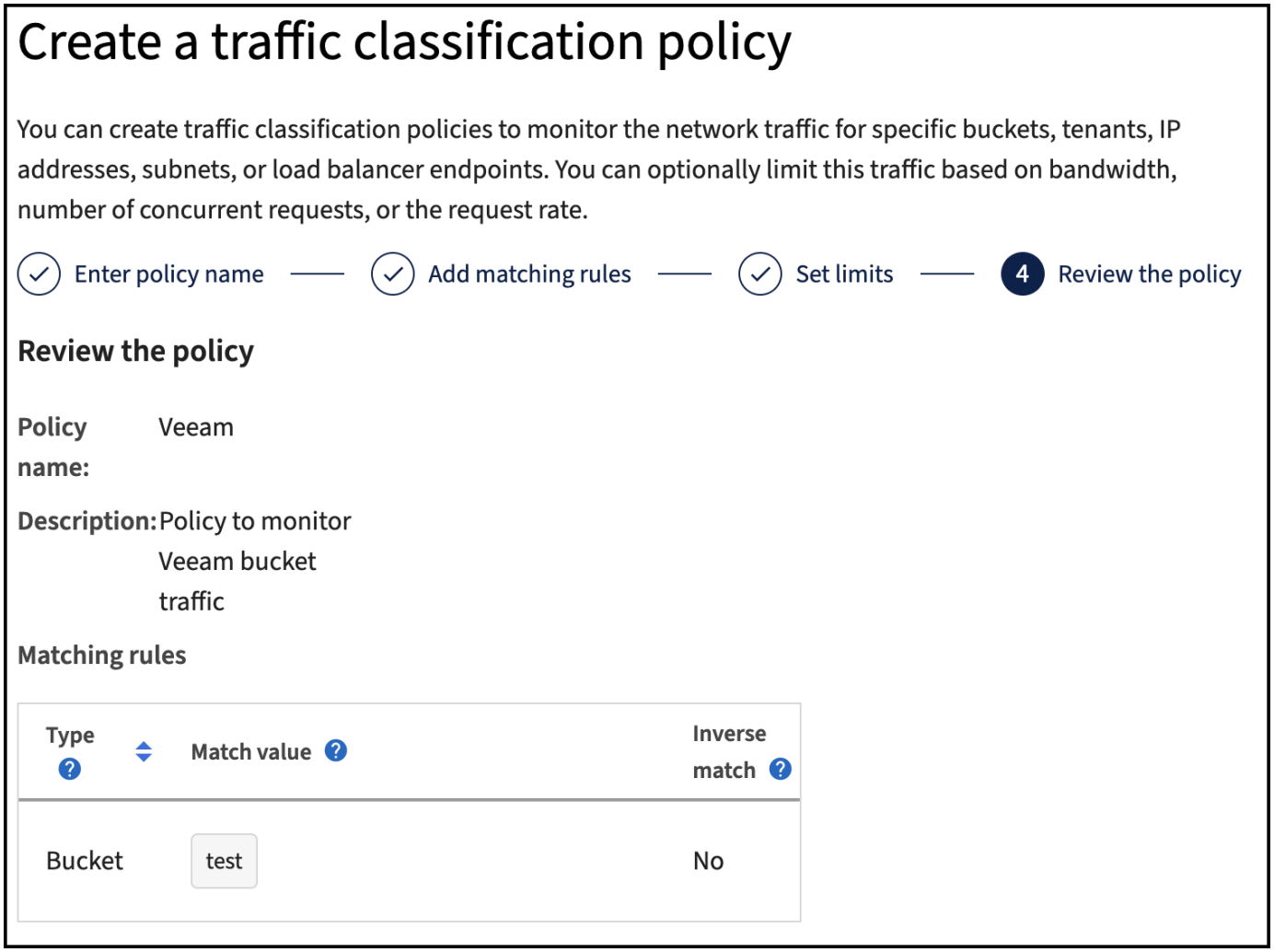
Veeamでは、S3バケットに対して強力な整合性が必要です。そのため、Veeamが複数の場所からデータにアクセスするマルチサイト環境の場合は、「strong-global」を選択します。Veeamのバックアップとリストアを単一サイトでのみ実行する場合は、整合性レベルを「strong-site」に設定する必要があります。変更を保存します。



StorageGRIDは、すべての管理ノードおよび専用のゲートウェイノードで統合されたロードバランササービス

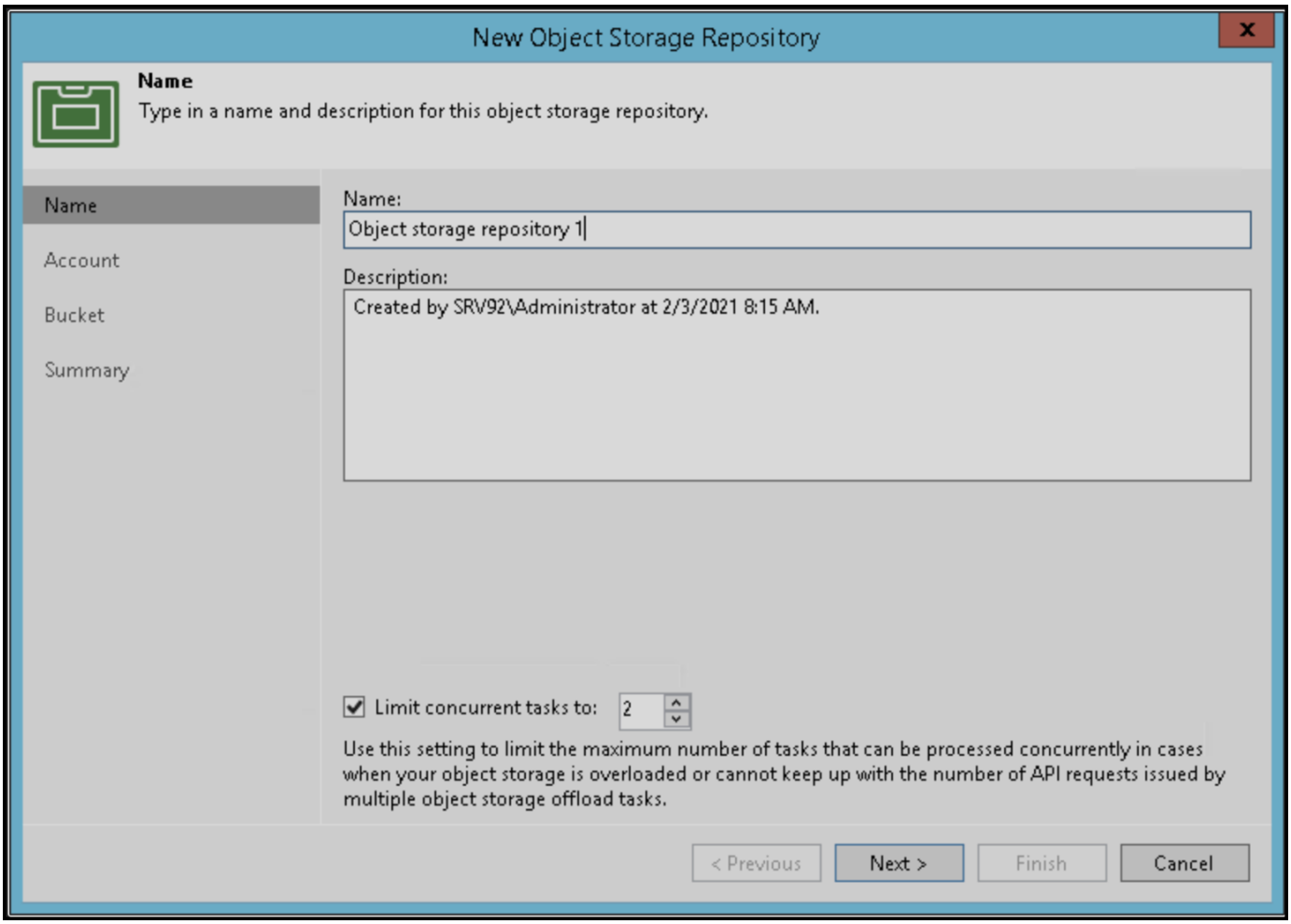
を提供します。このロードバランサを使用する多くの利点の1つは、トラフィック分類ポリシー（QoS）を設定できることです。主に、他のクライアントワークロードへのアプリケーションの影響を制限したり、他のクライアントワークロードよりもワークロードを優先したりするために使用されますが、監視に役立つ追加の指標収集のボーナスも提供します。

[Configuration]タブで、[Traffic Classification]を選択し、新しいポリシーを作成します。ルールに名前を付け、タイプとしてバケットまたはテナントを選択します。バケットまたはテナントの名前を入力します。QoSが必要な場合は制限を設定しますが、ほとんどの実装では、制限を設定しないでください。

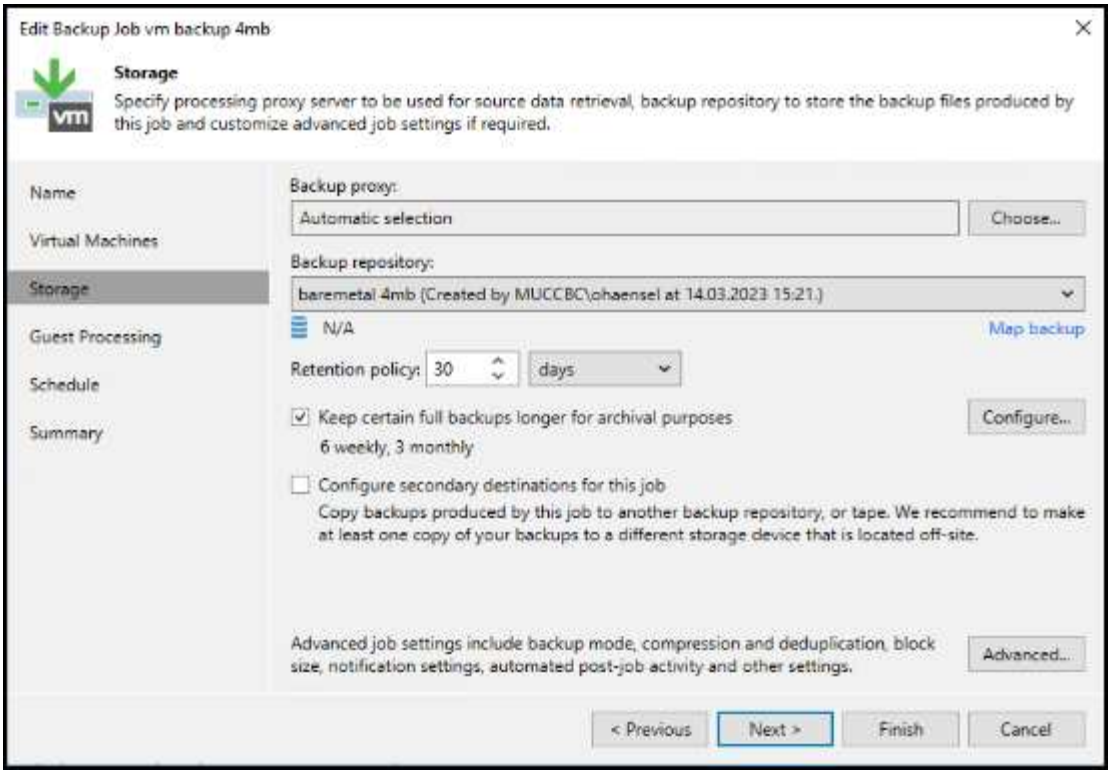


Veeamの統合によって

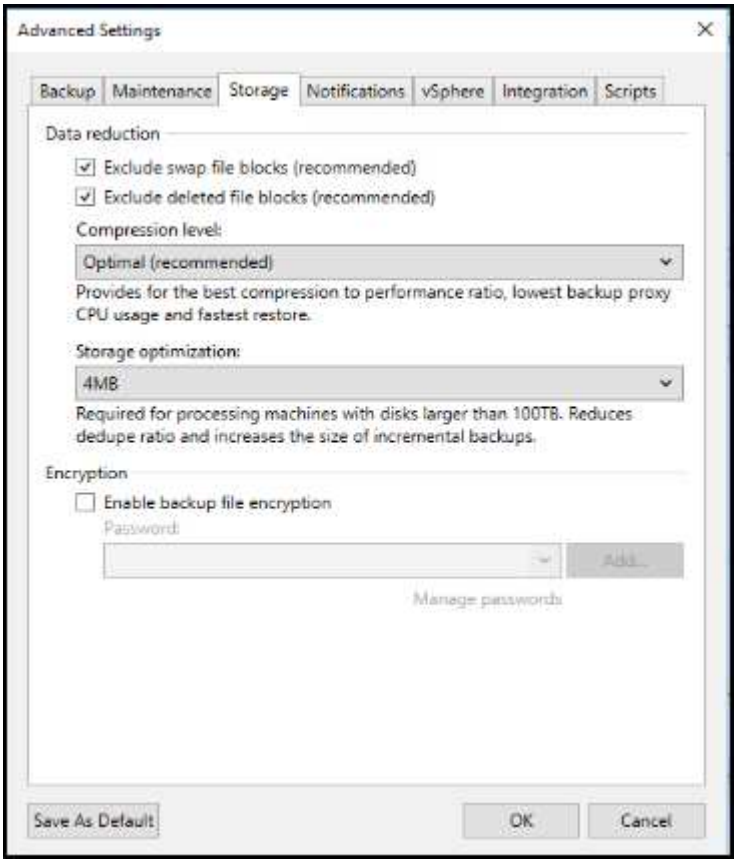
StorageGRIDアプライアンスのモデルと数によっては、バケットで同時に実行できる処理数の制限を選択して設定する必要があります。



Veeamコンソールのバックアップジョブ設定に関するVeeamのドキュメントに従って、ウィザードを開始します。VMを追加したら、SOBRリポジトリを選択します。



[詳細設定]をクリックし、ストレージ最適化設定を4 MB以上に変更します。圧縮機能と重複排除機能を有効にします。要件に応じてゲスト設定を変更し、バックアップジョブのスケジュールを設定します。



StorageGRID の監視

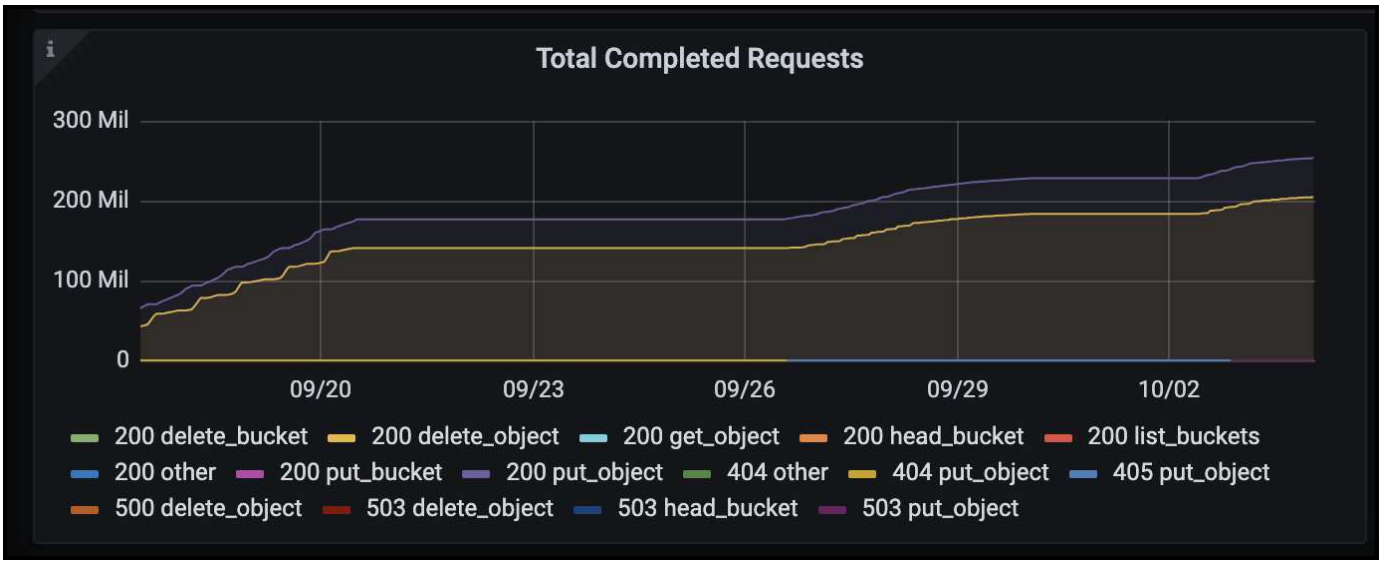
VeeamとStorageGRIDの連携によるパフォーマンスの全体像を把握するには、最初のバックアップの保持期限が切れるまで待つ必要があります。これまで、Veeamのワークロードは主にPUT処理で構成され、削除は行われていませんでした。バックアップデータの有効期限が近づいてクリーンアップを実行すると、オブジェクトストアに一貫した使用状況が表示され、必要に応じてVeeamで設定を調整できます。

StorageGRIDには、[Support]タブの[Metrics]ページにあるシステムの動作を監視するための便利なチャートが用意されています。主にS3の[Overview]、[ILM]、[Traffic Classification Policy]（ポリシーが作成されている場合）の各ダッシュボードを確認します。S3の[Overview]ダッシュボードには、S3の処理率、レイテンシ、要求応答に関する情報が表示されます。

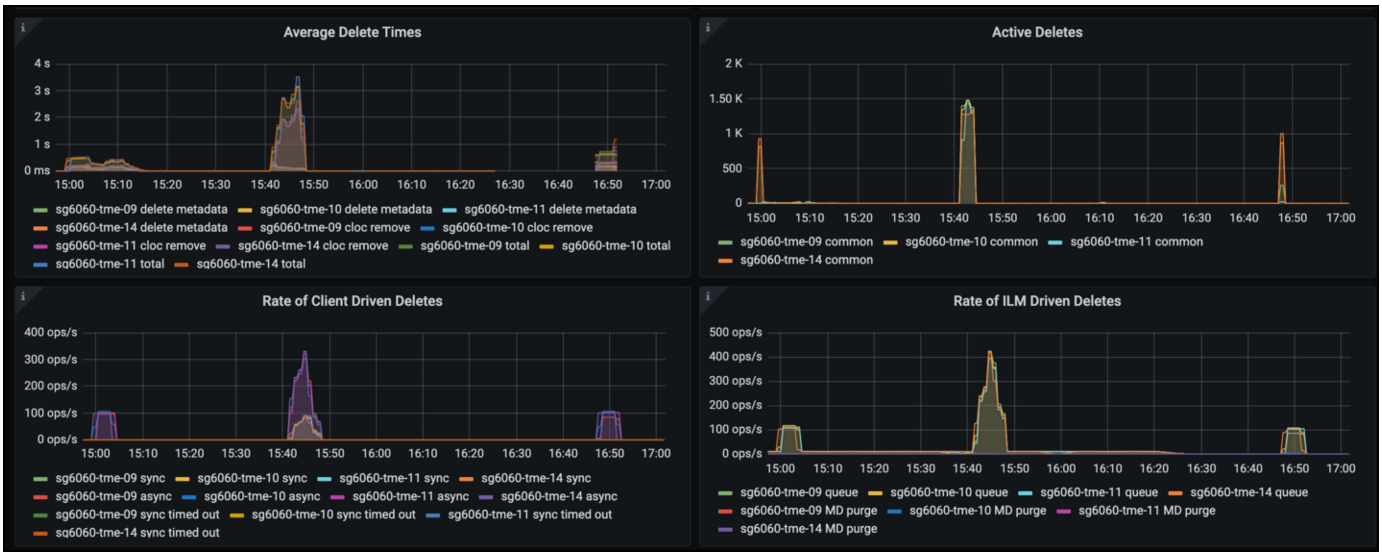
S3の速度とアクティブな要求を確認すると、各ノードで処理されている負荷の量と、タイプ別の要求の総数を確認できます。

[Average Duration]チャートには、各ノードの要求タイプごとの平均所要時間が表示されます。これはリクエストの平均遅延で、追加の調整が必要か、StorageGRIDシステムがより多くの負荷を引き受ける余地があることを示しているかもしれません。

[Total Completed Requests]チャートでは、リクエストをタイプコードと応答コード別に表示できます。応答に200（OK）以外の応答が表示された場合、これは、StorageGRIDシステムのような問題が503（スローダウン）応答を送信しており、追加の調整が必要になるか、負荷が増加するためにシステムを拡張する時間が来たことを示している可能性があります。



[ILM]ダッシュボードでは、StorageGRIDシステムの削除のパフォーマンスを監視できます。StorageGRIDでは、各ノードで同期削除と非同期削除を組み合わせて使用し、すべての要求の全体的なパフォーマンスを最適化しようとします。



トラフィック分類ポリシーを使用すると、ロードバランサ要求のスループット、レート、期間、およびVeeamが送受信するオブジェクトサイズに関するメトリックを表示できます。

追加情報の参照先

このドキュメントに記載されている情報の詳細については、以下のドキュメントや Web サイトを参照してください。

- "NetApp StorageGRID 11.7製品ドキュメント"
- "Veeam Backup Replication"

_ Oliver HaenselとAron Klein著_

StorageGRIDを使用したDremioデータソースの設定

Dremioは、クラウドベースやオンプレミスのオブジェクトストレージなど、多様なデータソースをサポートしています。 StorageGRIDをオブジェクトストレージデータソースとして使用するようにDremioを設定できます。

Dremioデータソースの設定

前提条件

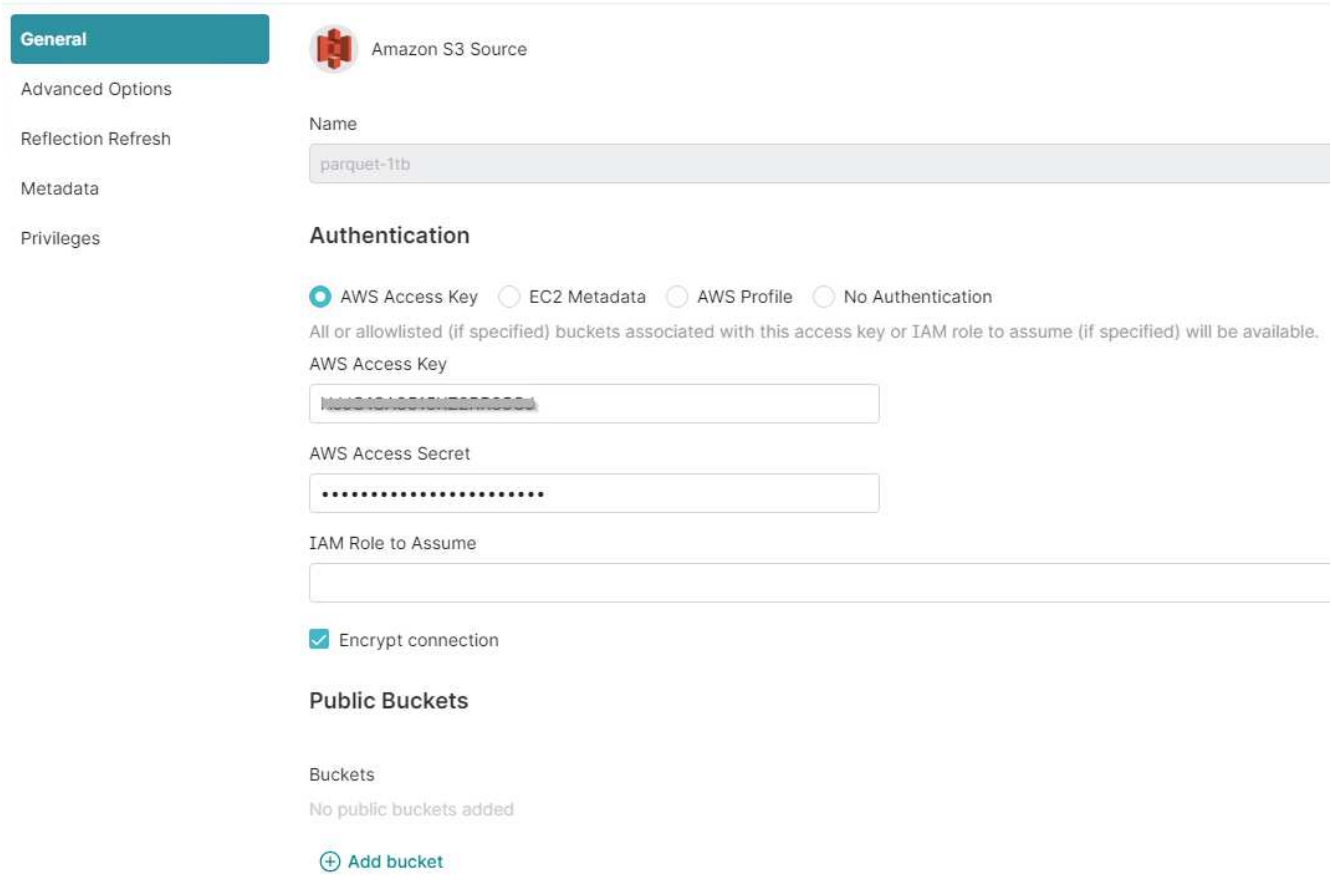
- StorageGRID S3エンドポイントURL、テナントs3アクセスキーID、シークレットアクセスキー。
- StorageGRID構成の推奨事項：圧縮を無効にします（デフォルトでは無効）。 [+]
  Dremioは、Byte range GETを使用して、クエリ中に同じオブジェクト内から異なるバイト範囲を同時に取得します。 バイト範囲要求の一般的なサイズは1MBです。圧縮オブジェクトを使用すると、バイト範囲GETのパフォーマンスが低下します。

Dremioガイド

"Amazon S3への接続- S3互換ストレージの設定"。

指示

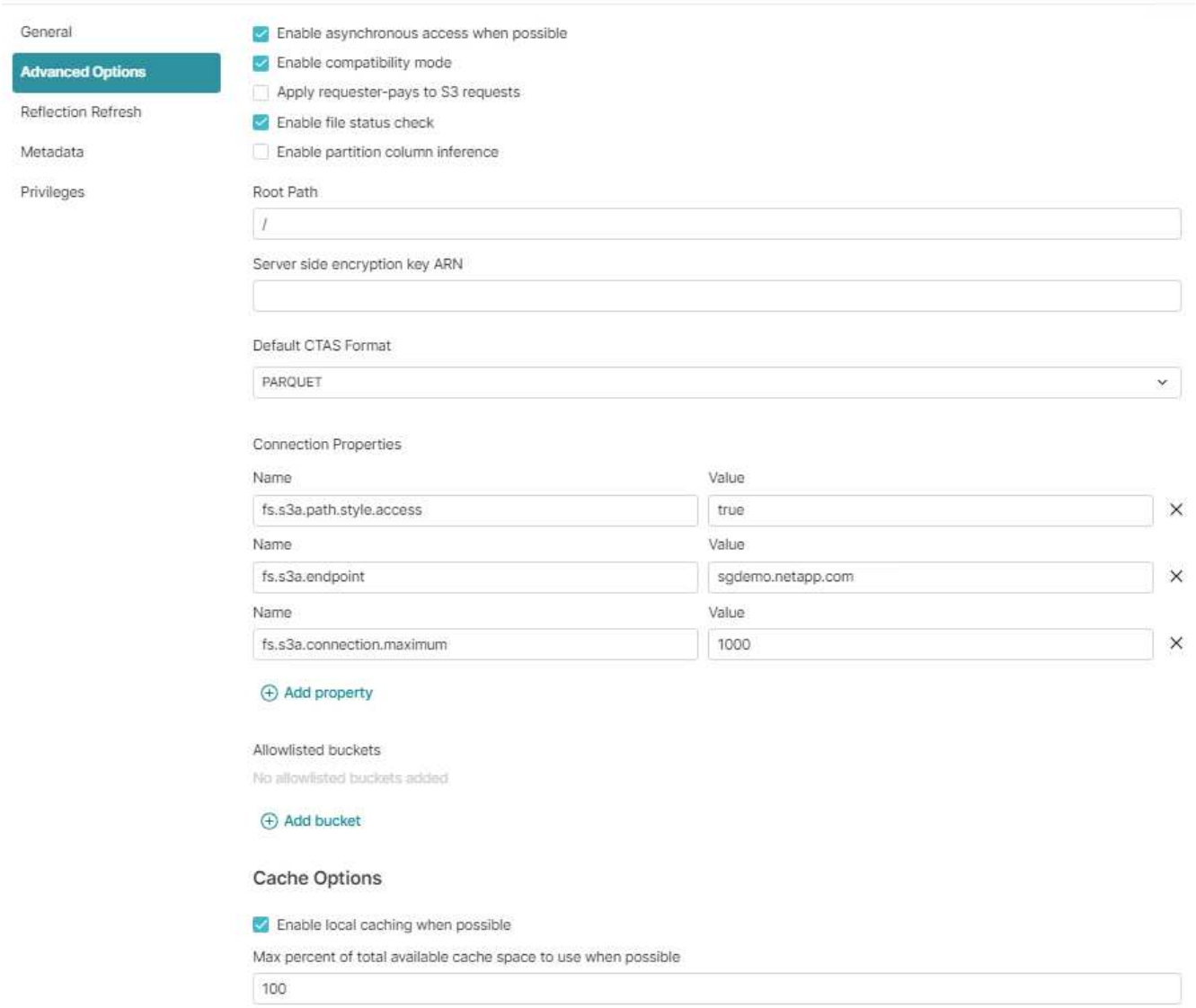
1. [Dremio Datasets]ページで、[+]をクリックしてソースを追加し、[Amazon S3]を選択します。
2. この新しいデータソースの名前（StorageGRID S3のテナントアクセスキーIDとシークレットアクセスキー）を入力します。
3. StorageGRID S3エンドポイントへの接続にhttpsを使用する場合は、[Encrypt connection]チェックボックスをオンにします。[+]
   このs3エンドポイントで自己署名CA証明書を使用する場合は、Dremioのガイド手順に従って、このCA証明書をDremioサーバの<JAVA_HOME>/jre/lib/security+に追加します。
   サンプルスクリーンショット



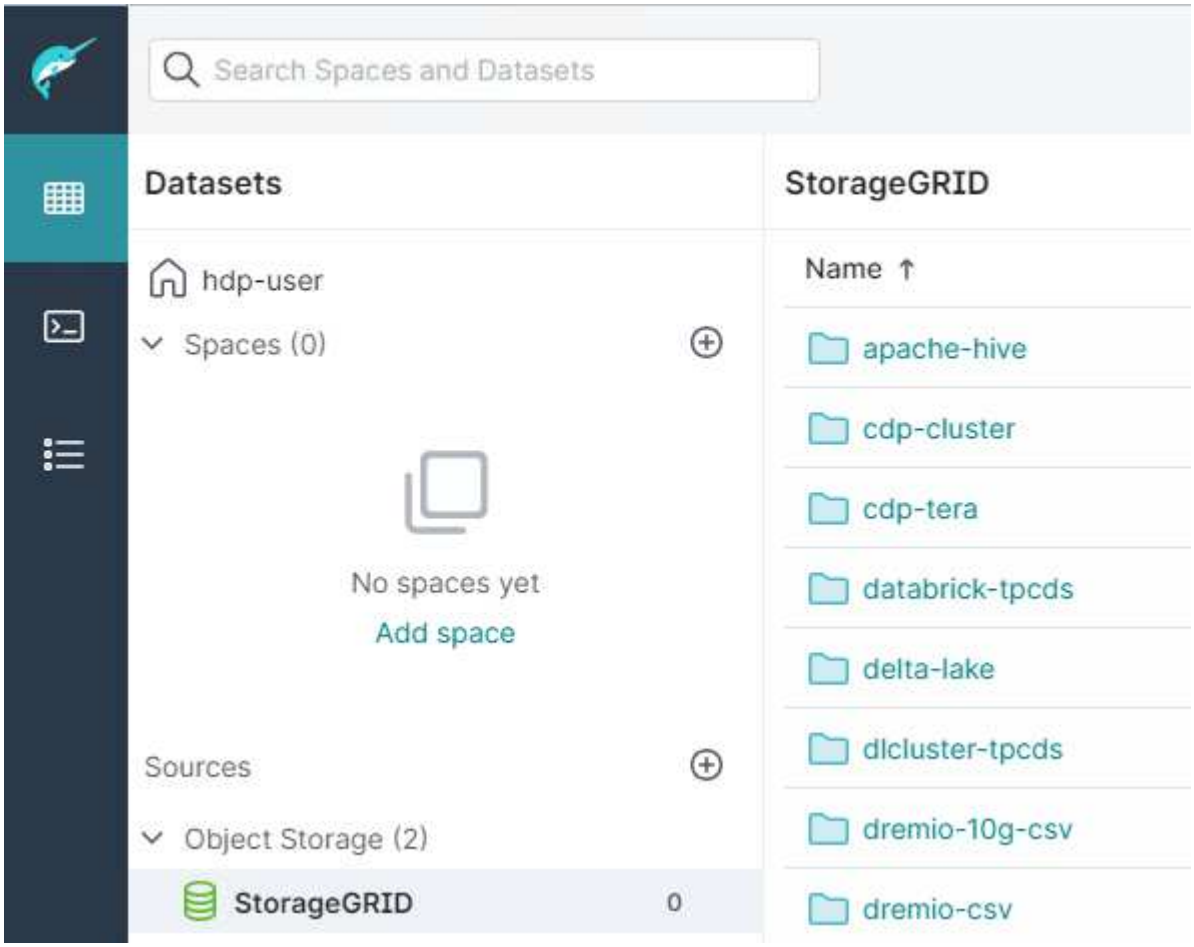
4. [詳細オプション]をクリックし、[互換モードを有効にする]をオンにします。
5. [Connection properties]で、[+ Add Properties]をクリックして、これらのs3aプロパティを追加します。
6. fs.s3a.connection.maximumデフォルトは100です。 s3データセットに100列以上の大きな寄木細工ファイルが含まれている場合は、100より大きい値を入力する必要があります。 この設定については、Dremioのガイドを参照してください。

名前	価値
FS.s3a.endpoint	_ StorageGRID S3エンドポイント：port>_
FS.s3a.path.style.access	正しいです
fs.s3a.connection.maximum	< *100*より大きい値>

サンプルスクリーンショット



7. 組織またはアプリケーションの要件に応じて、その他のDremioオプションを設定します。
8. [Save]ボタンをクリックして新しいデータソースを作成します。
9. StorageGRIDデータソースが正常に追加されると、バケットのリストが左側のパネルに表示されます。[+]サンプルスクリーンショット



Angela Cheng著_

NetApp StorageGRIDとGitLab

NetAppはStorageGRIDをGitLabでテストしました。以下のGitLabの設定例を参照してください。 を参照してください "GitLabオブジェクトストレージ構成ガイド" を参照してください。

オブジェクトストレージの接続例

Linuxパッケージのインストールの場合は、次の例を参照してください。 `connection` 統合フォームでの設定。編集 `/etc/gitlab/gitlab.rb` 次の行を追加し、必要な値を置き換えます。

```
# Consolidated object storage configuration
gitlab_rails['object_store']['enabled'] = true
gitlab_rails['object_store']['proxy_download'] = true
gitlab_rails['object_store']['connection'] = {
  'provider' => 'AWS',
  'region' => 'us-east-1',
  'endpoint' => 'https://<storagegrid-s3-endpoint:port>',
  'path_stype' => 'true',
  'aws_access_key_id' => '<AWS_ACCESS_KEY_ID>',
  'aws_secret_access_key' => '<AWS_SECRET_ACCESS_KEY>'
}
# OPTIONAL: The following lines are only needed if server side encryption
is required
gitlab_rails['object_store']['storage_options'] = {
  'server_side_encryption' => 'AES256'
}
gitlab_rails['object_store']['objects']['artifacts']['bucket'] = 'gitlab-
artifacts'
gitlab_rails['object_store']['objects']['external_diffs']['bucket'] =
'gitlab-mr-diffs'
gitlab_rails['object_store']['objects']['lfs']['bucket'] = 'gitlab-lfs'
gitlab_rails['object_store']['objects']['uploads']['bucket'] = 'gitlab-
uploads'
gitlab_rails['object_store']['objects']['packages']['bucket'] = 'gitlab-
packages'
gitlab_rails['object_store']['objects']['dependency_proxy']['bucket'] =
'gitlab-dependency-proxy'
gitlab_rails['object_store']['objects']['terraform_state']['bucket'] =
'gitlab-terraform-state'
gitlab_rails['object_store']['objects']['pages']['bucket'] = 'gitlab-
pages'
```