PURESTORAGE®

TECHNICAL WHITE PAPER

Scalability and High Data Retention with Prometheus and Thanos on FlashBlade

High availability for Prometheus time-series data using ObjectStore.

Contents

Introduction

Single-instance Prometheus servers are becoming more popular across various business and industry verticals. Organizations use them as a standard monitoring solution across many heterogeneous endpoints, including for hardware infrastructure. Prometheus allows admins and end-users seamless access to monitor and visualize relevant information across different platforms based on rules and policies without going through a massive learning curve from different vendors.

With the growing Prometheus footprint, long-term retention of historical data is gaining more attention. Admins and end-users in cross-functional teams require a global view of all endpoints with the appropriate role-based access rules and policies. High resiliency is required when a single-instance Prometheus server, exporter, or application crashes, and a gap in gathering the metrics occurs until the server and the application come back online. PromQL queries lose the metrics data during the gap when the server is offline.

Like other hardware vendors, Pure Storage® has a Pure Exporter that allows Prometheus to monitor and scrape metrics from FlashArray™ and FlashBlade® storage endpoints. You can configure single-instance Prometheus and Grafana on Pure Storage and monitor FlashArray and FlashBlade for other workloads running on these data platforms.

Thanos and FlashBlade

Thanos is a CNCF project in the incubation stage which runs as a sidecar to the Prometheus server. Thanos addresses the challenges mentioned above with various components like Sidecar, Query, Store, Compact, Ruler, and many more. Figure 1 illustrates how the different functional Thanos components are connected and configured on FlashBlade.

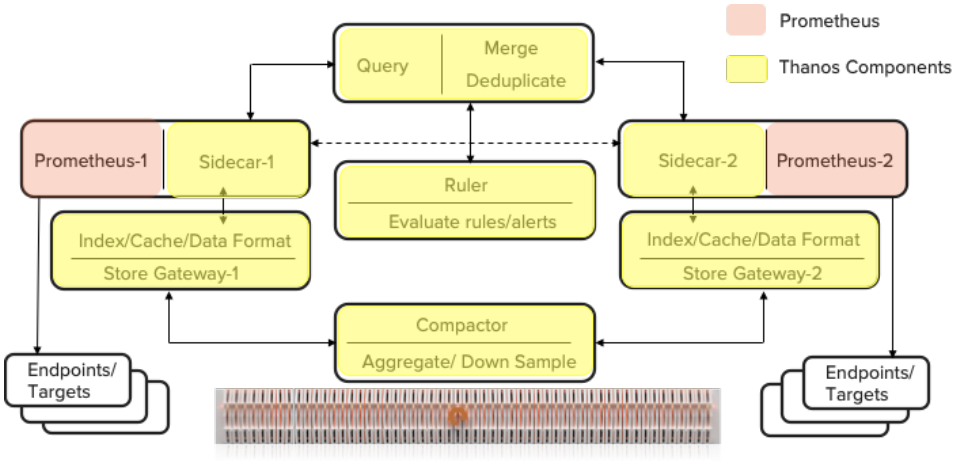

Figure 1. Prometheus Thanos functional layout

The Prometheus database (Promdb) is configured over NFS, and the long-term storage for historical data is stored on an object store bucket in the same FlashBlade. Configuring the Prometheus and Thanos components on FlashBlade has the following advantages:

- FlashBlade supports both files and Amazon S3-compatible object stores.
- FlashBlade provides POSIX-compliant NFS, backed with flash storage that eliminates write amplification with capacity and performance scalability required by every Prometheus and Thanos instance configured in the environment.
- Configures the Thanos Store Gateway on the NFS shares as the PromDB, allowing it to grow and shrink the cache for indexed time series data.
- FlashBlade provides a high data reduction of 2.5:1 for the PromDB over NFS and up to 4:1 for the historical data in object store buckets.
- FlashBlade can also replicate the object store buckets containing historical data to AWS S3 for extended retention time periods.

This paper provides detailed steps to configure the different Thanos components along with the changes required to the Prometheus configuration. (Explore the steps to set up and configure a single instance Prometheus and Pure Exporter.) The rest of this paper assumes that at least two single instances of Prometheus are running in isolation. Each of the Prometheus Servers could have its own set of endpoints to monitor.

Prerequisites for Prometheus and Thanos Installation

Pure Exporter is used to monitor and visualize array level metrics from different FlashArray and FlashBlade endpoints by two different Prometheus servers. In this paper, the two Prometheus servers use external labels with replica names of “Prometheus-1” and “Prometheus-2,” respectively.

The following prerequisites are required before starting to configure Prometheus and Thanos:

1. DNS entries for the Prometheus servers and the various Thanos components should be set as shown in the table below.

Prometheus-1 <10.21.152.65>

```
[root@sn1-r620-a04-05 prometheus]# nslookup 10.21.152.65

65.152.21.10.in-addr.arpa      name = ruler-1.puretec.purestorage.com.
65.152.21.10.in-addr.arpa      name = sidecar-1.puretec.purestorage.com.
65.152.21.10.in-addr.arpa      name = compact-1.puretec.purestorage.com.
65.152.21.10.in-addr.arpa      name = sn1-r620-a04-05.puretec.purestorage.com.
65.152.21.10.in-addr.arpa      name = prometheus-1.puretec.purestorage.com.
65.152.21.10.in-addr.arpa      name = pure-exporter-1.puretec.purestorage.com.
65.152.21.10.in-addr.arpa      name = store-1.puretec.purestorage.com.
65.152.21.10.in-addr.arpa      name = query-1.puretec.purestorage.com.
```

Prometheus-2 <10.21.236.116>

```
[root@sn1-r720-g09-19 ~]# nslookup 10.21.236.116

116.236.21.10.in-addr.arpa     name = pure-exporter-2.puretec.purestorage.com.
116.236.21.10.in-addr.arpa     name = store-2.puretec.purestorage.com.
116.236.21.10.in-addr.arpa     name = query-2.puretec.purestorage.com.
116.236.21.10.in-addr.arpa     name = ruler-2.puretec.purestorage.com.
116.236.21.10.in-addr.arpa     name = sidecar-2.puretec.purestorage.com.
116.236.21.10.in-addr.arpa     name = compact-2.puretec.purestorage.com.
116.236.21.10.in-addr.arpa     name = sn1-r720-g09-19.puretec.purestorage.com.
116.236.21.10.in-addr.arpa     name = prometheus-2.puretec.purestorage.com.
```

2. The following ports should be open for the HTTP and gRPC communication between the Prometheus and the Thanos components.

Component	Interface	Ports
Sidecar	gRPC	10901
Sidecar	HTTP	10902
Query	gRPC	10903
Query	HTTP	10904
Store	gRPC	10905
Store	HTTP	10906
Rule	gRPC	10910
Rule	HTTP	10911
Compact	HTTP	10912

3. The following NFS mounts (if this is a first-time Prometheus install) and object store buckets need to be created. Create promDB and promDB2 file systems on FlashBlade.

promDB	10 T	3.66 G	0	2.6 to 1	1.42 G	0.00	1.42 G
promDB2	10 T	1.67 G	0	2.5 to 1	688.87 M	0.00	688.87 M

Both the promDB and promDB2 file systems on FlashBlade need to be mounted on /var/lib/prometheus on Linux hosts with "vers=3" and "sync" as the mount options.

```
10.21.236.101:/promDB on /var/lib/prometheus type nfs
(rw,relatime,sync,vers=3,rsize=524288,wsize=524288,namlen=255,hard,proto=tcp,timeo=600,retrans=2,
sec=sys,mountaddr=10.21.236.101,mountvers=3,mountport=2049,mountproto=tcp,local_lock=none,addr=10
.21.236.101)
```

4. The objectstore buckets for Thanos Store and Ruler need to be created on the same FlashBlade. The bucket names are "store" and "ruler," respectively.

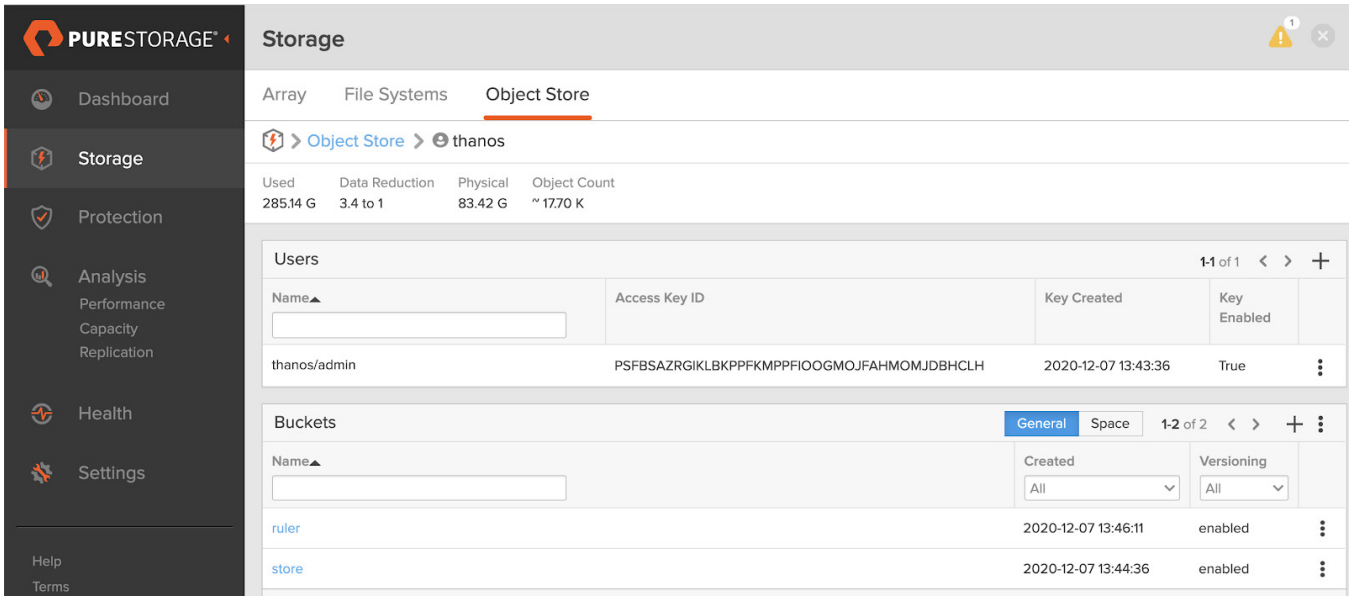

Figure 2. Thanos object store.

5. The Pure Exporter should be running in Docker containers in Prometheus-1 and Prometheus-2 servers, respectively.

Prometheus-1 Server

```
[root@sn1-r620-a04-05 ~]# docker ps
CONTAINER ID   IMAGE                                    COMMAND                  CREATED       STATUS       PORTS                    NAMES
028d9b70e254   quay.io/purestorage/pure-exporter:1.2.3  "gunicorn pure_expor…"   3 weeks ago   Up 3 weeks   0.0.0.0:9491->9491/tcp   pure-exporter-1
```

Prometheus-2 Server

```
[root@sn1-r720-g09-19 ~]# docker ps
CONTAINER ID   IMAGE                                    COMMAND                  CREATED       STATUS       PORTS                    NAMES
459b8c5021c7   quay.io/purestorage/pure-exporter:1.2.3  "gunicorn pure_expor…"   3 weeks ago   Up 5 days    0.0.0.0:9491->9491/tcp   pure-exporter-2
```

6. GOlang needs to be installed and configured on the Prometheus servers.

```
[root@sn1-r720-g09-19 ~]# wget https://dl.google.com/go/go1.15.5.linux-amd64.tar.gz

[root@sn1-r720-g09-19 ~]# tar -C /usr/local -xzf go1.15.5.linux-amd64.tar.gz
[root@sn1-r720-g09-19 ~]# export PATH=$PATH:/usr/local/go/bin
[root@sn1-r720-g09-19 ~]# go version
go version go1.15.5 linux/amd64
[root@sn1-r720-g09-19 ~]# vi .bash_profile
[root@sn1-r720-g09-19 ~]# source ~/.bash_profile
[root@sn1-r720-g09-19 ~]# go version
go version go1.15.5 linux/amd64
[root@sn1-r720-g09-19 ~]#
```

Prometheus and Thanos Setup

For best practices and prerequisites for installing and configuring Prometheus, please see this white paper. At the time of writing this paper, Thanos v0.17.2 was used for this documentation. Note that Thanos must be downloaded and installed in both the Prometheus servers.

```
[root@sn1-r720-g09-19 bin]# wget https://github.com/thanos-
io/thanos/releases/download/v0.17.2/thanos-0.17.2.linux-amd64.tar.gz

[root@sn1-r720-g09-19 ~]# tar xvzf thanos-0.17.2.linux-amd64.tar.gz
thanos-0.17.2.linux-amd64/
thanos-0.17.2.linux-amd64/thanos
[root@sn1-r720-g09-19 ~]# mv thanos-0.17.2.linux-amd64/thanos /usr/bin/thanos

[root@sn1-r720-g09-19 bin]# thanos --version
thanos, version 0.17.2 (branch: HEAD, revision: 37e6ef61566c7c70793ba6d128f00c4c66cb2402)
  build user:       root@92283ccb0bc0
  build date:       20201208-10:00:57
  go version:       go1.15
  platform:         linux/amd64
[root@sn1-r720-g09-19 bin]#
```

Sidecar is the first Thanos component configured as a service on both the Prometheus-1 and Prometheus-2 servers. Use the ports listed above for the sidecar.service file in the /etc/systemd/system location.

In the below table, the --objstore.config file points to the location where the “store” bucket information is held.

Prometheus-1	Prometheus-2
[root@sn1-r620-a04-05 system]# cat sidecar.service [Unit] Description=Prometheus Wants=network-online.target After=network-online.target [Service] User=prometheus Group=prometheus Type=simple ExecStart=/bin/thanos sidecar \ --prometheus.url=http://prometheus-1:9090 \ --grpc-address=prometheus-1:10901 \ --http-address=prometheus-1:10902 \ --tsdb.path /var/lib/prometheus/ \ --objstore.config-file /etc/prometheus/bucket.yml [Install] WantedBy=multi-user.target [root@sn1-r620-a04-05 system]#	[root@sn1-r720-g09-19 system]# cat sidecar.service [Unit] Description=Prometheus Wants=network-online.target After=network-online.target [Service] User=prometheus Group=prometheus Type=simple ExecStart=/bin/thanos sidecar \ --prometheus.url=http://prometheus-2:9090 \ --grpc-address=prometheus-2:10901 \ --http-address=prometheus-2:10902 \ --tsdb.path /var/lib/prometheus/ \ --objstore.config-file /etc/prometheus/bucket.yml [Install] WantedBy=multi-user.target [root@sn1-r720-g09-19 system]#

The following table provides the details of the "store" bucket created on the FlashBlade. It consists of the bucket name, endpoint (data VIP of the FlashBlade), access key, and the secret key of the bucket. Both the Prometheus servers are pointing to the same "store" bucket on FlashBlade but using different data VIPs.

Prometheus-1	Prometheus-2
[root@sn1-r620-a04-05 system]# cat /etc/prometheus/bucket.yml type: S3 config: bucket: store endpoint: 10.21.236.202 region: local access_key: <paste the access_key from the bucket> insecure: false signature_version2: false secret_key: : <paste the secret_key from the bucket> put_user_metadata: {} http_config: idle_conn_timeout: 1m30s	[root@sn1-r720-g09-19 system]# cat /etc/prometheus/bucket.yml type: S3 config: bucket: store endpoint: 10.21.236.203 region: local access_key: <paste the access_key from the bucket> insecure: false signature_version2: false secret_key: : <paste the secret_key from the bucket> put_user_metadata: {} http_config: idle_conn_timeout: 1m30s

```
        response_header_timeout: 2m
        insecure_skip_verify: true
    trace:
        enable: false
    list_objects_version: ""
    part_size: 134217728

[root@sn1-r620-a04-05 system]#
```

```
        response_header_timeout: 2m
        insecure_skip_verify: true
    trace:
        enable: false
    list_objects_version: ""
    part_size: 134217728

[root@sn1-r720-g09-19 system]#
```

The prometheus.yaml file needs to be updated with an external label and the replica name should be defined to identify the location of two Prometheus servers. The time series data is identified from the replicas' labels.

Prometheus-1

```
[root@sn1-r620-a04-05 system]# cat
/etc/prometheus/prometheus.yml
global:
    scrape_timeout: 1m
    external_labels:
      cluster: us1
      replica: prometheus-1

rule_files:
    - alert.rules.yml

scrape_configs:
    - job_name: 'prometheus'
      scrape_interval: 5s
      static_configs:
          - targets: ['prometheus-
1:9090','prometheus-2:9090']

    - job_name: 'sidecar'
      static_configs:
          - targets: ['prometheus-
1:10902','prometheus-2:10902']
```

Prometheus-2

```
[root@sn1-r720-g09-19 system]# cat
/etc/prometheus/prometheus.yml
global:
    scrape_timeout: 1m
    external_labels:
        cluster: us2
        replica: prometheus-2

rule_files:
    - alert.rules.yml

scrape_configs:
    - job_name: 'prometheus'
      scrape_interval: 5s
      static_configs:
          - targets: ['prometheus-
2:9090','prometheus-1:9090']

    - job_name: 'sidecar'
      static_configs:
          - targets: ['prometheus-
1:10902','prometheus-2:10902']
```

The prometheus.yaml file must be updated with an external label and the replica name has to be defined to identify the location of two Prometheus servers. The time series data is identified from the replicas' labels.

Prometheus-1

```
[root@sn1-r620-a04-05 system]# cat
/etc/prometheus/prometheus.yml
global:
   scrape_timeout: 1m
   external_labels:
     cluster: us1
     replica: prometheus-1

rule_files:
   - alert.rules.yml

scrape_configs:
   - job_name: 'prometheus'
     scrape_interval: 5s
     static_configs:
        - targets: ['prometheus-
1:9090','prometheus-2:9090']

   - job_name: 'sidecar'
     static_configs:
        - targets: ['prometheus-
1:10902','prometheus-2:10902']
```

Prometheus-2

```
[root@sn1-r720-g09-19 system]# cat
/etc/prometheus/prometheus.yml
global:
   scrape_timeout: 1m
   external_labels:
      cluster: us2
      replica: prometheus-2

rule_files:
   - alert.rules.yml

scrape_configs:
   - job_name: 'prometheus'
     scrape_interval: 5s
     static_configs:
        - targets: ['prometheus-
2:9090','prometheus-1:9090']

   - job_name: 'sidecar'
     static_configs:
        - targets: ['prometheus-
1:10902','prometheus-2:10902']
```

After updating the prometheus.yaml and configuring the sidecar.service files both services need to restart/start/enable on both Prometheus servers.

```
systemctl daemon-reload

systemctl restart prometheus
systemctl status prometheus -l

systemctl start sidecar
systemctl enable sidecar
systemctl status sidecar -l
```

Next, configure the Thanos store or store gateway on both the Prometheus servers. Create Store-1 and Store-2 directories under /var/lib/prometheus NFS share for the Prometheus-1 and Prometheus-2 servers, respectively.

Prometheus-1

```
mkdir /var/lib/prometheus/store-1/
```

Prometheus-2

```
mkdir /var/lib/prometheus/store-2/
```

The /var/lib/prometheus/store-1 is used as the path to the data-dir where the initial time series data is stored before being written to the object store bucket store on the FlashBlade. The bucket information is also in the following store.service file

where the historical data will be written for a longer retention period. The store.service file is created in the /etc/systemd/system location.

Prometheus-1	Prometheus-2
[root@sn1-r620-a04-05 system]# cat store.service [Unit] Description=Thnaos Store Wants=network-online.target After=network-online.target [Service] User=root Group=root Type=simple ExecStart=/bin/thanos store \ --data-dir=/var/lib/prometheus/store-1/ \ --objstore.config- file=/etc/prometheus/bucket.yml \ --http-address=localhost:10906 \ --grpc-address=store-1:10905 [Install] WantedBy=multi-user.target [root@sn1-r620-a04-05 system]#	[root@sn1-r720-g09-19 system]# cat store.service [Unit] Description=Thnaos Store Wants=network-online.target After=network-online.target [Service] User=root Group=root Type=simple ExecStart=/bin/thanos store \ --data-dir=/var/lib/prometheus/store-2/ \ --objstore.config- file=/etc/prometheus/bucket.yml \ --http-address=localhost:10906 \ --grpc-address=store-2:10905 [Install] WantedBy=multi-user.target [root@sn1-r720-g09-19 system]#

The store.service has to be started on both the Prometheus servers:

```
systemctl daemon-reload

systemctl start store
systemctl enable store
systemctl status store -l
```

The next component to configure is the Thanos query. The query points to both Store-1 and Store-2 for high availability and resiliency purposes. We recommend that you create a query service on both the Prometheus servers for high availability reasons in the /etc/systemd/system location. Either one of the queries can be used as a data source to Grafana, which provides a global view of all the endpoints that are monitored by each of Prometheus servers respectively.

Prometheus-1	Prometheus-2
[root@sn1-r620-a04-05 system]# cat query.service [Unit] Description=Thnaos Query Wants=network-online.target After=network-online.target [Service] User=root Group=root Type=simple ExecStart=/bin/thanos query \ --http-address=query-1:10904 \ --grpc-address=query-1:10903 \ --store=store-1:10901 \ --store=store-2:10901 \ --query.replica-label=prometheus-1 [Install] WantedBy=multi-user.target [root@sn1-r620-a04-05 system]#	[root@sn1-r720-g09-19 system]# cat query.service [Unit] Description=Thnaos Query Wants=network-online.target After=network-online.target [Service] User=root Group=root Type=simple ExecStart=/bin/thanos query \ --http-address=query-2:10904 \ --grpc-address=query-2:10903 \ --store=store-1:10901 \ --store=store-2:10901 \ --query.replica-label prometheus-2 [Install] WantedBy=multi-user.target [root@sn1-r720-g09-19 system]#

In an earlier section, external labels with the replica names were configured for both Prometheus clusters. The query merges the time series data from different external labels into one single dataset. If the Prometheus server or exporter or application goes offline, the query will hide the data collection gaps from the impacted Prometheus server. For more information with examples, refer to Thanos documentation.

The query service needs to be started on both the Prometheus servers:

```
systemctl daemon-reload

systemctl start query
systemctl enable query
systemctl status query -l
```

Thanos Compact is another service that aggregates indexed time series data before writing to the object store bucket store on FlashBlade. We recommended that you create a “compact” directory under the /var/lib/prometheus NFS share as the default --date-dir location on both Prometheus servers for the compact to write the data. The compact is also responsible for downsampling the large metric datasets into smaller chunks.

Prometheus-1	Prometheus-2
mkdir /var/lib/prometheus/compact-1/	mkdir /var/lib/prometheus/compact-2/

The compact service must be configured with the --data-dir path and the store bucket location details.

Prometheus-1	Prometheus-2
[root@sn1-r620-a04-05 system]# cat compact.service [Unit] Description=Thanos compact Wants=network-online.target After=network-online.target [Service] User=root Group=root Type=simple ExecStart=/bin/thanos compact \ --data-dir=/var/lib/prometheus/compact-1/ \ --objstore.config-file=/etc/prometheus/bucket.yml \ --http-address=compact-1:10912 [Install] WantedBy=multi-user.target [root@sn1-r620-a04-05 system]#	[root@sn1-r620-a04-05 system]# cat compact.service [Unit] Description=Thanos compact Wants=network-online.target After=network-online.target [Service] User=root Group=root Type=simple ExecStart=/bin/thanos compact \ --data-dir=/var/lib/prometheus/compact-2/ \ --objstore.config-file=/etc/prometheus/bucket.yml \ --http-address=compact-1:10912 [Install] WantedBy=multi-user.target [root@sn1-r620-a04-05 system]#

The compact service can then be started on both the Prometheus servers.

```
systemctl daemon-reload

systemctl start compact
systemctl enable compact
systemctl status compact -l
```

The final piece to this puzzle is to configure the Thanos ruler. We recommended configuring the ruler on both Prometheus servers for high availability but for this paper, a single ruler is created on Prometheus-1.

Create a new "--data-dir" directory in the /var/lib/prometheus NFS share location for the most recent and active rules and alerts and continuing to write the historical data to the "ruler" bucket on FlashBlade.

Prometheus-1
mkdir /var/lib/prometheus/ruler-1/

The ruler consolidates the rules to combine the metrics data and condition-based alerts into one location. It is recommended to create a new objectstore bucket "ruler" on FlashBlade that stores all the rules and the alerts.

```
[root@sn1-r620-a04-05 system]# cat /etc/prometheus/bucket2.yml
type: S3
config:
  bucket: ruler
  endpoint: 10.21.236.201
  region: local
  access_key: <paste the access_key from the bucket>
  insecure: false
  signature_version2: false
  secret_key: <paste the secret_key from the bucket>
  put_user_metadata: {}
  http_config:
    idle_conn_timeout: 1m30s
    response_header_timeout: 2m
    insecure_skip_verify: true
  trace:
    enable: false
  list_objects_version: ""
#  part_size: 134217728

[root@sn1-r620-a04-05 system]#
```

The ruler service is configured with the "--objstore.config-file" pointing to the new bucket "ruler" and eventually starts the service on Prometheus-1 server.

Prometheus-1

```
[root@sn1-r620-a04-05 system]# cat ruler.service
[Unit]
Description=Thnaos Ruler
Wants=network-online.target
After=network-online.target
[Service]
User=root
Group=root
Type=simple
ExecStart=/bin/thanos rule \
   --data-dir="/var/lib/prometheus/ruler-1" \
   --eval-interval=30s \
   --rule-file=/etc/prometheus/alert.rules.yml \
   --alert.query-url=http://0.0.0.0:9090 \
   --alertmanagers.url=http://localhost:9093 \
```

```
   --http-address=ruler-1:10911 \
   --grpc-address=0.0.0.0:10910 \
   --query=http://query-1:10904 \
   --query=http://query-2:10904 \
   --objstore.config-file=/etc/prometheus/bucket2.yml \
   --label 'monitor_cluster="us1"' \
   --label 'replica="prometheus-1"'
[Install]
WantedBy=multi-user.target
[root@sn1-r620-a04-05 system]#
```

```
   ystemctl daemon-reload

   systemctl start ruler
   systemctl enable ruler
   systemctl status ruler -l
```

Conclusion

This completes the configuration of all Thanos components. Services will now run on both Prometheus servers. Either Query-1 or Query-2 can be used as the data source to Grafana. Shared resources for promDB, promDB2, and "--data-dir" paths on the /var/lib/prometheus NFS shares provide the resilience and elasticity to scale capacity as the metrics data grows. The metrics data size depends on the number of endpoints scraped and the number of metrics gathered from each endpoint. Data reduction from 2.5:1 up to 4:1 between the NFS shares and the buckets enables Prometheus to have a long data retention time with a smaller data footprint.

About the Author

Bikash Roy Choudhury is a technical director at Pure Storage. He is responsible for designing and architecting solutions with EDA/HPC and SWDev/DevOps workflows relevant across industry verticals including high tech, financial services, gaming, social media, and web-based organizations in Private/Hybrid/Public Cloud environments. He has also worked on validating solutions with Rancher/TKG/Kubernetes on Portworx, GitLab, Jenkins, JFrog Artifactory, Prometheus/Grafana, IBM Cloud Private and Perforce using RESTful APIs and integrating them with data platforms in private, hybrid, and public clouds. In his current role, Bikash drives integrations with strategic DevOps partners like Rancher, D2iQ/Konvoy, VMWare, Perforce, Cloudbees, and JFrog.

Pure Storage, Inc.
650 Castro Street, #400
Mountain View, CA 94041