Monitor Tessera

You can use Tessera withInfluxDB orPrometheus time-series databases to record API usage metrics. You can visualize the recorded data using an existing dashboard tool such asGrafana .

In addition, you can search, analyze, and monitor Tesseralogs usingSplunk orElastic Stack (ELK) . You can set up Splunk such that the logs for multiple Tessera nodes in a network are accessible from a single centralized Splunk instance.

Record API metrics

Tessera can record the following usage metrics for each endpoint of its API:

- Average response time
- Maximum response time
- Minimum response time
- Request count
- Requests per second

You can store these metrics in an InfluxDB or Prometheus time-series database for further analysis.

- UseInfluxDB
- when you prefer for metrics to be "pushed" from Tessera to the database. For example, Tessera starts a service which periodically writes the latest metrics to the database by calling the database's API.
- UsePrometheus
- when you prefer for metrics to be "pulled" from Tessera by the database. For example, Tessera exposes a/metrics
- API endpoint which the database periodically calls to fetch the latest metrics.

Both databases integrate well with the open source dashboard editorGrafana to allow for easy creation of dashboards to visualize the data being captured from Tessera.

InfluxDB

TheInfluxDB documentation provides details on how to set up an InfluxDB database ready for use with Tessera. A summary of the steps is as follows:

1. Install InfluxDB
2. .
3. Start the InfluxDB server:
4. influxd -config /path/to/influx.conf
5. For local development and testing, the default configuration file (Linux:/etc/influxdb/influxdb.conf
6. , macOS:/usr/local/etc/influxdb.conf
7.) is sufficient. For further configuration options seeConfiguring InfluxDB
8. .
9. Connect to the InfluxDB server using theinflux CLI
10. and create a new database. If using the default configuration, this is as follows:
11. influx

12. CREATE DATABASE myDb

13. To view data stored in the database, use theInflux Query Language
14. .
15. influx

16. USE myDb

17. SHOW MEASUREMENTS

18. SELECT * FROM

info You can call the InfluxDB HTTP API directly as an alternative to using theinflux CLI. You can optionally configure each Tessera server type (for example,P2P ,Q2T ,ADMIN ,THIRDPARTY ,ENCLAVE) to store API metrics in an InfluxDB. You can configure these servers to store metrics to the same database or separate ones.

To configure a server to use an InfluxDB, addinfluxConfig to the server configuration. For example:

"serverConfigs": [{ "app":"Q2T", "serverAddress":"unix:/path/to/tm.ipc", "influxConfig": { "serverAddress": "https://localhost:8086", // InfluxDB server address "dbName": "myDb", // InfluxDB DB name (DB must already exist) "pushIntervalInSecs": 15, // How frequently Tessera will push new metrics to the DB "sslConfig": { // Config required if InfluxDB server is using TLS "tls": "STRICT", "sslConfigType": "CLIENT_ONLY", "clientTrustMode": "CA", "clientTrustStore":

"/path/to/truststore.jks", "clientTrustStorePassword": "password", "clientKeyStore": "path/to/truststore.jks", "clientKeyStorePassword": "password" } } }, { "app":"P2P", "serverAddress":"http://localhost:9001", "influxConfig": { "serverAddress": "http://localhost:8087", "dbName": "anotherDb", "pushIntervalInSecs": 15 } }]

InfluxDB TLS Configuration

InfluxDB supports one-way TLS. This allows clients to validate the identity of the InfluxDB server and provides data encryption.

SeeEnabling HTTPS with InfluxDBfor details on how to secure an InfluxDB server with TLS. A summary of the steps is as follows:

1. Obtain a CA/self-signed certificate and key (either as separate.crt
2. and.key
3. files or as a combined.pem
4. file).
5. Enable HTTPS ininflux.conf
6. :
7. # Determines whether HTTPS is enabled.
8. https-enabled = true
9. # The SSL certificate to use when HTTPS is enabled.
10. https-certificate = "/path/to/certAndKey.pem"
11. # Use a separate private key location.
12. https-private-key = "/path/to/certAndKey.pem"
13. Restart the InfluxDB server to apply the configuration changes.
14. To allow Tessera to communicate with a TLS-secured InfluxDB, you must providesslConfig
15. in the configuration file. Configure Tessera as the client in one-way TLS:
16. "sslConfig": {
17. "tls": "STRICT",
18. "sslConfigType": "CLIENT_ONLY",
19. "clientTrustMode": "CA",
20. "clientTrustStore": "/path/to/truststore.jks",
21. "clientTrustStorePassword": "password",
22. "clientKeyStore": "path/to/truststore.jks",
23. "clientKeyStorePassword": "password",
24. "environmentVariablePrefix": "INFLUX"
25. }
26. wheretruststore.jks
27. is a Java keystore format file containing the trusted certificates for the Tessera client (for example, the certificate of the CA used to create the InfluxDB certificate).
28. If securing the key store with a password, you must provide this password. Passwords can be provided either in the configuration (for exampleclientTrustStorePassword
29.) or as environment variables (usingenvironmentVariablePrefix
30. and setting_TESSERA_CLIENT_TRUSTSTORE_PWD
31.). TheTLS configuration
32. documentation explains this in more detail.
33. As Tessera expects two-way TLS, a.jks
34. file for theclientKeyStore
35. must also be provided. This isn't used so can simply be set as the trust store.

Prometheus

ThePrometheus documentation provides information to set up Prometheus to integrate with Tessera. ThePrometheus first steps is a good starting point. A summary of the steps to store Tessera metrics in a Prometheus database is as follows:

1. Install Prometheus.
2. Create aprometheus.yml
3. configuration file to provide Prometheus with the necessary information to pull metrics from Tessera. You can usean example Prometheus configuration

4. with theQuorum Developer Quickstart
5. .
6. Start Tessera. Tessera always exposes themetrics
7. endpoint, so no additional configuration of Tessera is required.
8. Start Prometheus:
9. prometheus --config.file=prometheus.yml
10. To view data stored in the database, access the Prometheus UI (by defaultlocalhost:9090
11. , this address can be changed inprometheus.yml
12.) and use thePrometheus Query Language
13. .

Grafana

You can import a pre-builtGoQuorum Grafana dashboard to visualize your recorded GoQuorum network data.

Monitor logs

You can search, analyze, and monitor thelogs of Tessera nodes usingSplunk orElastic Stack (ELK) .

Splunk

Set up Splunk and Splunk Universal Forwarders to consolidate the logs from multiple Tessera nodes. The following pages from the Splunk documentation are a good starting point for understanding how to achieve this:

- Consolidate data from multiple hosts
- Set up the Universal Forwarder
- Configure the Universal Forwarder
- Enable a receiver

The general steps to consolidate the logs for a Tessera network in Splunk are:

1. Set up a central Splunk instance if one does not already exist. Typically this is on a host separate to the hosts running the Tessera nodes. This is known as the Receiver.
2. Configure the Tessera hosts to forward their nodes' logs to the Receiver by:
3. 1. Configuring the format and output location of the node's logs. This is achieved by configuring Logback (the logging framework used by Tessera) at node start-up.
4. 1. The following example XML configures Logback to save Tessera's logs to a file. See theLogback documentation
5. 1. for more information on configuring Logback:
6. 1. <?xml version="1.0" encoding="UTF-8"?>
7. 1. <
8. 1. configuration
9. 1.
10. 1. <
11. 1. appender
12. 1. name
13. 1. =
14. 1. "
15. 1. FILE
16. 1. "
17. 1. class
18. 1. =

19.
    1. "
20.
    1. ch.qos.logback.core.FileAppender
21.
    1. "
22.
    1.

23.
    1. <
24.
    1. file
25.
    1.

26.
    1. /path/to/file.log
27.
    1. <!--
28.
    1. file
29.
    1. -->
30.
    1. <
31.
    1. encoder
32.
    1.

33.
    1. <
34.
    1. pattern
35.
    1.

36.
    1. %d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n
37.
    1. <!--
38.
    1. pattern
39.
    1. -->
40.
    1. <!--
41.
    1. encoder
42.
    1. -->
43.
    1. <!--
44.
    1. appender
45.
    1. -->
46.
    1. <
47.
    1. logger
48.
    1. name
49.
    1. =
50.
    1. "
51.

1. org.glassfish.jersey.internal.inject.Providers
52.
1. "
53.
1. level
54.
1. =
55.
1. "
56.
1. ERROR
57.
1. "
58.
1. />
59.
1. <
60.
1. logger
61.
1. name
62.
1. =
63.
1. "
64.
1. org.hibernate.validator.internal.util.Version
65.
1. "
66.
1. level
67.
1. =
68.
1. "
69.
1. ERROR
70.
1. "
71.
1. />
72.
1. <
73.
1. logger
74.
1. name
75.
1. =
76.
1. "
77.
1. org.hibernate.validator.internal.engine.ConfigurationImpl
78.
1. "
79.
1. level
80.
1. =
81.
1. "
82.
1. ERROR
83.
1. "
84.
1. />
85.

1. <
86.
1. root
87.
1. level
88.
1. =
89.
1. "
90.
1. INFO
91.
1. "
92.
1.

93.
1. <
94.
1. appender-ref
95.
1. ref
96.
1. =
97.
1. "
98.
1. FILE
99.
1. "
100.
1. />
101.
1. <!--
102.
1. root
103.
1. -->
104.
1. <!--
105.
1. configuration
106.
1. -->
107.
1. To start Tessera with an XML configuration file:
108.
1. tessera -Dlogback.configurationFile=/path/to/logback-config.xml -configfile /path/to/config.json
109.
1. Set up Splunk Universal Forwarders (lightweight Splunk clients) on each Tessera host to forward log data for its node to the Receiver.
110.
1. Set up the Splunk Receiver to listen and receive logging data from the Universal Forwarders.

Elastic Stack

Follow theQuorum Developer Quickstart to use Elastic Stack (ELK) to manage Tessera logs.Edit this page Last updatedonOct 9, 2023 bydependabot[bot] Previous Use Next Logging