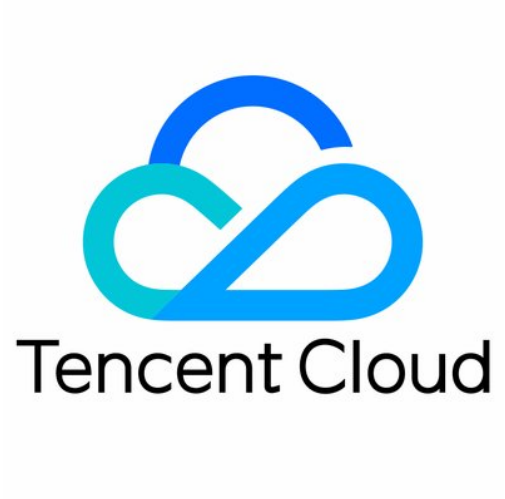
TencentDB for CTSDB

Best Practice

Product Documentation



Contents

Best Practice
Implementing Public Network Access with Iptables-Based Forwarding

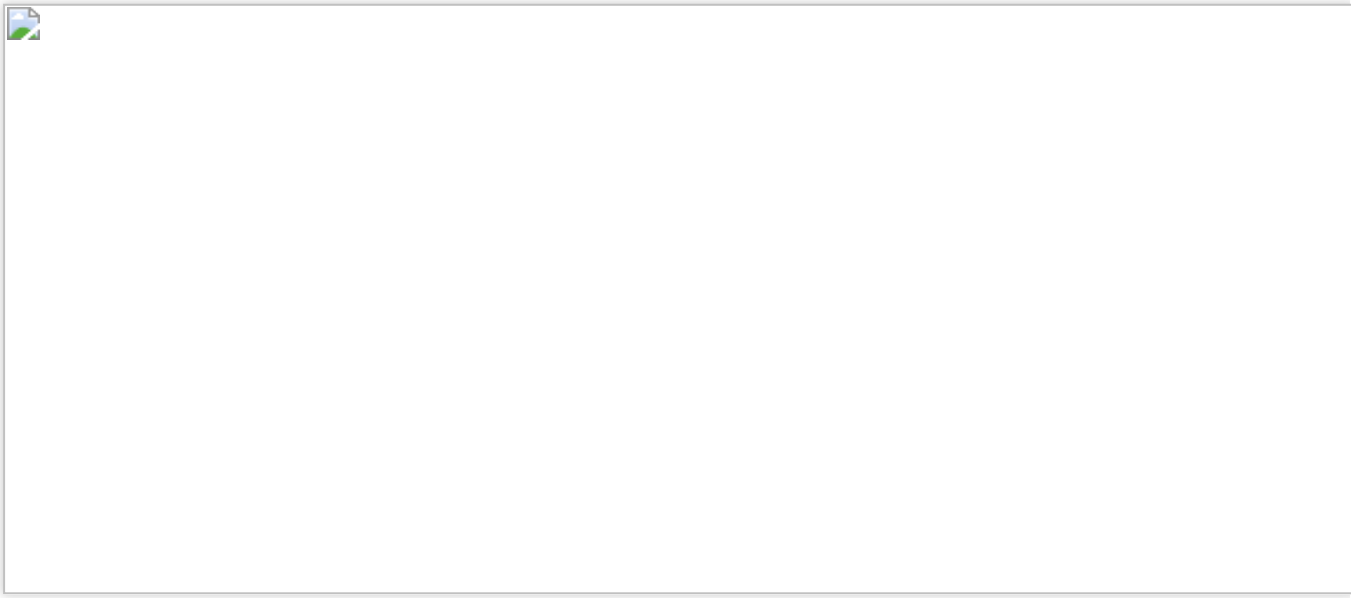
Last updated：2023-03-02 14:43:00

Overview

CTSDB doesn't support direct public network access for the time being. You can use a CVM instance with a public IP for port forwarding so as to access CTSDB over the public network.

Notes

Because iptables-based forwarding may be unstable, we recommend that you do not access instances over the public network in the production environment.

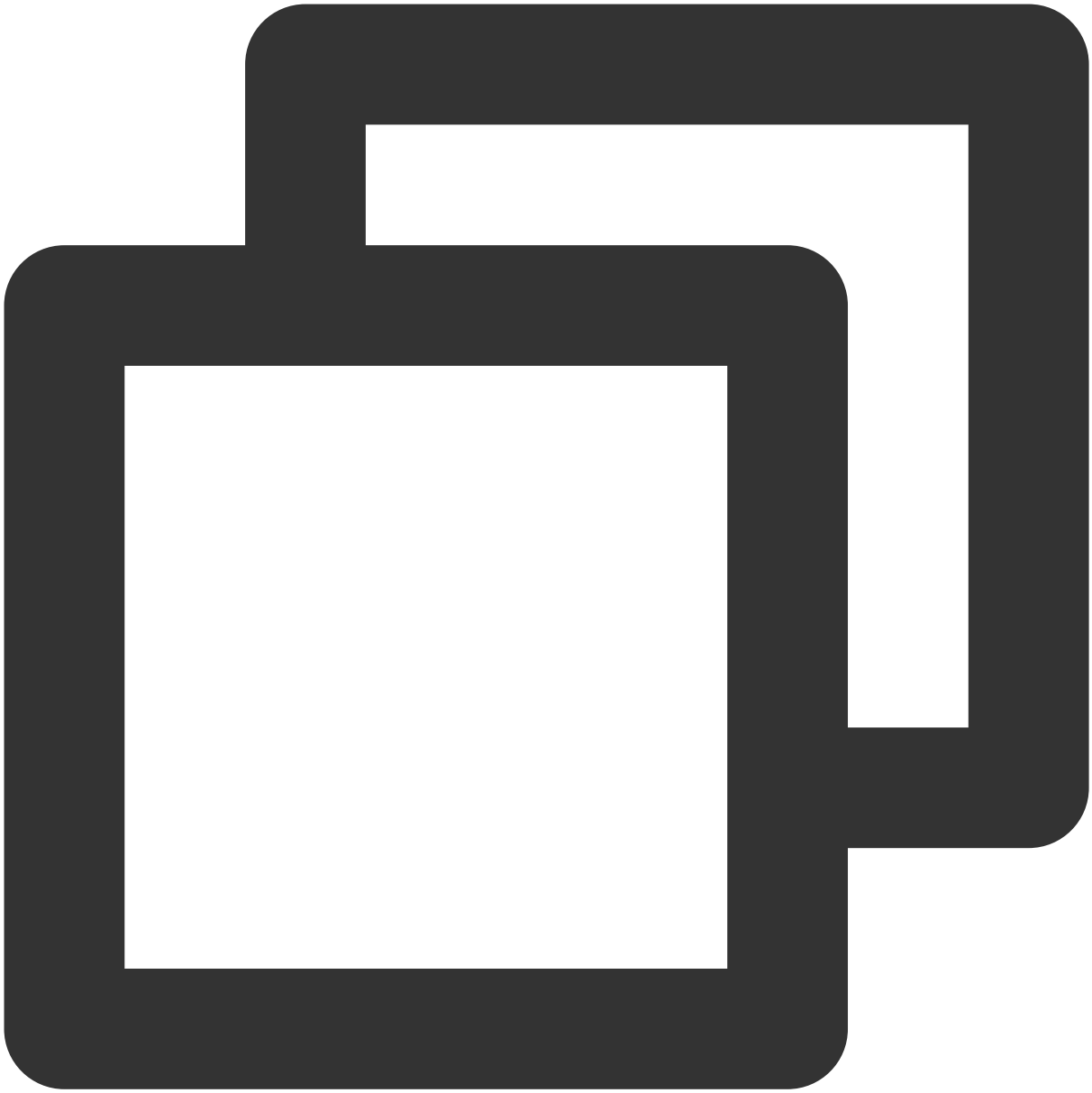


Directions

1. Log in to a CVM instance and enable the IP forwarding feature. For more information, see Logging In To Linux Instance (Web Shell).

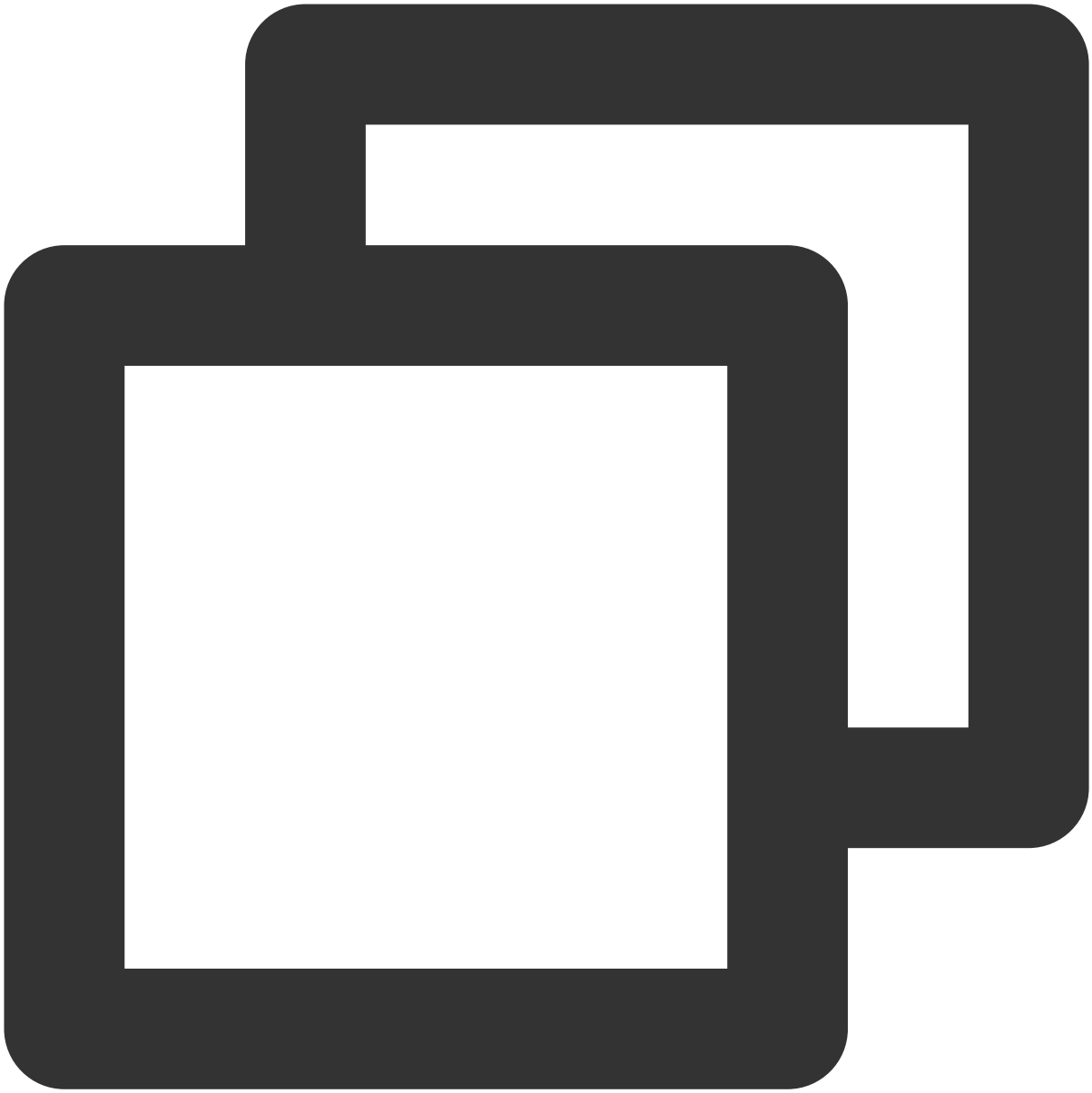
Notes

The CVM instance and the database must be under the same Tencent Cloud account and in the same VPC in the same region.



```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

2. Configure the forwarding rule. The sample code below forwards access requests originally to `26.xx.x.2:10001` (CVM instance public IP with a custom port as desired) to the CTSDB instance with the private IP `10.x.x.5:9200` :



```
iptables -t nat -A PREROUTING -p tcp --dport 10001 -j DNAT --to-destination 10.x.x.
iptables -t nat -A POSTROUTING -d 10.x.x.5 -p tcp --dport 9200 -j MASQUERADE
```

3. Configure the security group to open the public port of the CVM instance. We recommend that you configure a security group rule to allow only the source which needs to connect to the Redis instance. For more information, see Adding Security Group Rules.

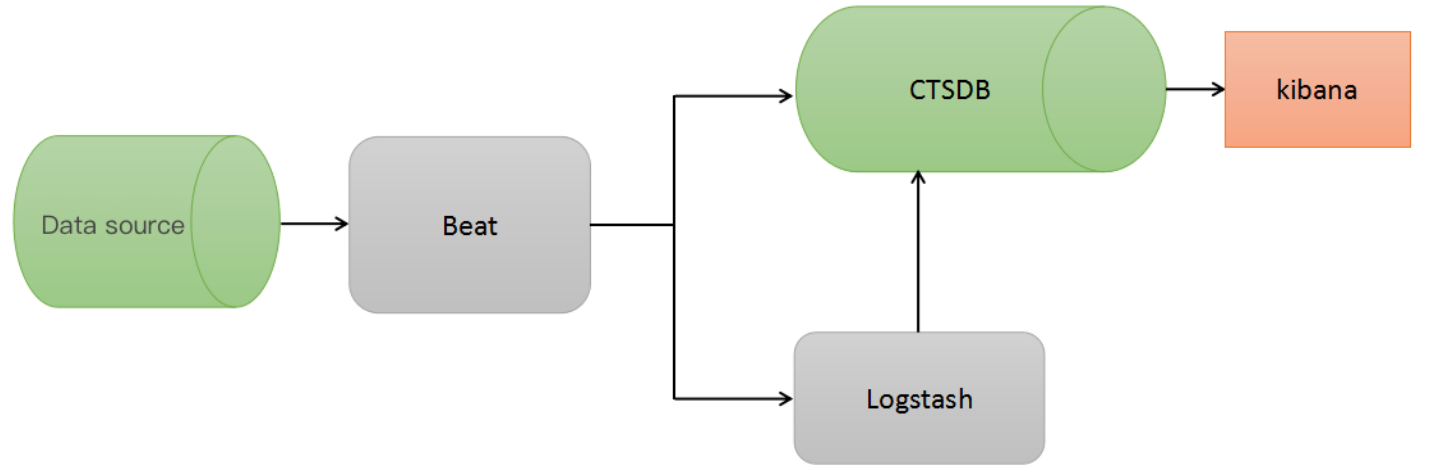
4. To connect to the CTSDB instance over the private network at its public IP (`26.xx.xx.2:10001` in the sample code), you can use the same way as the private network connection. For more information, see Connecting to Instance.

Connecting CTSDB with ELK and Grafana

Last updated：2021-07-15 16:49:45

Overview

CTSDB is a distributed and scalable time series database that supports near-real time data search and analysis. It is compatible with components in the ELK ecosystem, so you can conveniently connect your ELK components to CTSDB.
ELK ecosystem components provide a wide variety of data processing capabilities such as data collection, data cleansing, and visualized graph display. Common ELK components include Filebeat, Logstash, and Kibana. In addition, CTSDB also allows you to use Grafana as the visualized platform. A common architecture is as shown below:



Component Usage

Filebeat

Filebeat is a lightweight open-source log file data collector and is installed on servers as an agent. It reads file content, sends the content to Logstash for parsing, and then transfers the parsed content to CTSDB, or directly sends the file content to CTSDB for centralized storage and analysis.

Filebeat directions

1. **Install**
   For more information on how to install Filebeat, please see Filebeat quick start: installation and configuration.

2. **Configure**

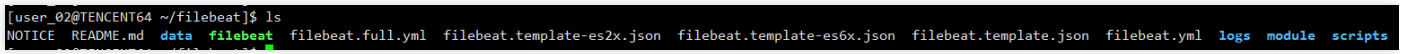
   Filebeat configurations are in a YAML file, including the global, input, and output configurations. The following section provides a usage example.

3. **Start**

   When starting Filebeat, you can specify the path of the configuration file; otherwise, `filebeat.yml` will be used by default.

Filebeat usage example

1. First, decompress the Filebeat installation package into a directory as shown below:



2. Then, configure `filebeat.yml`. The following sample is for reference:

```
filebeat.shutdown_timeout: 5 # How long filebeat waits on shutdown for the publ
isher to finish.
max_procs: 4 # Maximum number of CPUs that can run concurrently, which is the n
umber of available logical CPUs of the OS by default
filebeat.spool_size: 102400
filebeat.idle_timeout: 2s
processors:
drop_fields: # Fields to be dropped
fields: ["beat","input_type","source","offset"]
filebeat.prospectors:
paths: ["/data/log/filebeat-tutorial.log"] # Path of the sample data
fields:
metricname: metric1
harvester_buffer_size: 1638400
close_timeout: 0.5h
scan_frequency: 2s
paths: ["/mylog/*.log","/mylog1/*.log"]
fields:
metricname: table2
harvester_buffer_size: 1638401
close_timeout: 0.5h
scan_frequency: 2s
output.elasticsearch:
hosts: ["127.0.0.1:9200"]
index: "%{[fields.indexname]}" # Wildcard, which can be used to write different
indexes for different types of data
username: "root" # For CTSDB instances that require authentication, you need to
enter the username and password
```

```
password: "changeme"
worker: 2 # Number of worker threads
loadbalance: true # Whether to enable load balancing
bulk_max_size: 512 # Maximum number of documents in one `bulk` request
flush_interval: 2s
template:
enabled: false # Note: after start, Filebeat will put a default template. When
connecting to CTSDB, you need to disable the Filebeat template
```

Some sample data is as shown below:

```
83.149.9.216 - - [04/Jan/2015:05:13:42 +0000] "GET /presentations/logstash-moni
torama-2013/images/kibana-search.png HTTP/1.1" 200 203023 "http://semicomplete.
com/presentations/logstash-monitorama-2013/" "Mozilla/5.0 (Macintosh; Intel Mac
OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/
537.36"
83.149.9.216 - - [04/Jan/2015:05:13:42 +0000] "GET /presentations/logstash-moni
torama-2013/images/kibana-dashboard3.png HTTP/1.1" 200 171717 "http://semicompl
ete.com/presentations/logstash-monitorama-2013/" "Mozilla/5.0 (Macintosh; Intel
Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Saf
ari/537.36"
83.149.9.216 - - [04/Jan/2015:05:13:44 +0000] "GET /presentations/logstash-moni
torama-2013/plugin/highlight/highlight.js HTTP/1.1" 200 26185 "http://semicompl
ete.com/presentations/logstash-monitorama-2013/" "Mozilla/5.0 (Macintosh; Intel
Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Saf
ari/537.36"
```

3. Start Filebeat and observe the data in the corresponding CTSDB metric:

```
nohup ./filebeat &
less logs/filebeat # View certain logs. Through the log `libbeat.es.published_a
nd_acked_events=100`, you can see that all 100 logs have been successfully writ
ten into Elasticsearch
2018-05-25T14:32:24+08:00 INFO Non-zero metrics in the last 30s: filebeat.harve
ster.open_files=1 filebeat.harvester.running=1 filebeat.harvester.started=1 lib
beat.es.call_count.PublishEvents=1 libbeat.es.publish.read_bytes=1535 libbeat.e
s.publish.write_bytes=40172 libbeat.es.published_and_acked_events=100 libbeat.p
ublisher.published_events=100 publish.events=101 registrar.states.current=1 reg
istrar.states.update=101 registrar.writes=2

# Check whether there is data written into `metric1` through Kibana or curl
# Command:
GET metric1/_search
{
```

```
"sort": [
{
"@timestamp": {
"order": "desc"
}
}
],
"docvalue_fields": ["@timestamp", "message"]
}
# Result:
{
"took": 2,
"timed_out": false,
"_shards": {
"total": 3,
"successful": 3,
"skipped": 0,
"failed": 0
},
"hits": {
"total": 100,
"max_score": null,
"hits": [
{
"_index": "metric1@1525536000000_30",
"_type": "doc",
"_id": "AWOV_o1wBzkw2jsSfrLN",
"_score": null,
"fields": {
"@timestamp": [
1527229914629
],
"message": [
"218.30.103.62 - - [04/Jan/2015:05:27:57 +0000] \"GET /blog/geekery/c-vs-python-bdb.html HTTP/1.1\" 200 11388 \"-\" \"Sogou web spider/4.0(+http://www.sogou.com/docs/help/webmasters.htm#07)\""
]
},
"sort": [
1527229914629
]
},
# As the content is too much, it is omitted here. In `hits.total`, you can see that the query hit 100 documents, indicating that all the 100 logs have been successfully written into CTSDB
```

The above sample directly uses Filebeat to write the original log data into CTSDB without parsing the fields. The following section describes how to use Logstash to parse data and then write parsed data into CTSDB.

Logstash

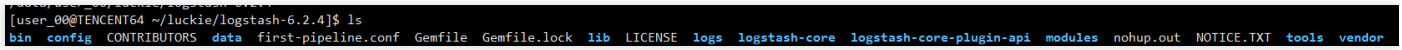
Logstash is an open-source data collection engine that can parse data in real time. It can collect data from various sources, filter, analyze, and format the data, and then store the data into CTSDB.

Logstash directions

1. **Install**
   For more information on how to install Logstash, please see Installing Logstash.
2. **Configure**
   Logstash configurations mainly include three modules: data source input, data parsing rule, and data output. The following section provides a usage example.
3. **Start**
   When starting Logstash, you can specify the configuration file; otherwise, `logstash.yml` will be used as the configuration by default, and configuration in `pipelines.yml` will be used as the parsing rule by default.

Logstash usage example

1. First, decompress the Logstash installation package into a directory as shown below:



2. Then, create a configuration file. You can also configure in `logstash.yml` and `pipelines.yml`. In this example, create a configuration file named `first-pipeline.conf` and configure it as follows:

```
# Input source
input {
beats {
port => "5044"
}
}
# Parse and filter
filter {
grok {
match => {
"message" => "%{COMBINEDAPACHELOG}"
}
}
}
# Output
output {
```

```
elasticsearch {
action => "index"
hosts => ["localhost:9200"]
index => "logstash_metric" # Name of the metric created in CTSDB
document_type => "doc"
user => "root" # For CTSDB instances that require authentication, you need to s
pecify the username and password
password => "changeme"
}
}
```

The Grok filter plugin is available in Logstash by default and can parse unstructured data into structured data. For more information on how to use it, please see Grok filter plugin.

3. Start Logstash and Filebeat and observe the data in the corresponding CTSDB metric:

```
# Here, you should note that the Filebeat output is Logstash; therefore, you ne
ed to modify the Filebeat output configuration item as follows:
output.logstash:
hosts: ["localhost:5044"]
# Clear the `data` directory of Filebeat and start Filebeat
rm data/registry
nohup ./filebeat &
# Start Logstash
nohup bin/logstash -f first-pipeline.conf --config.reload.automatic &
# Check whether there is data written into `metric1` through Kibana or curl
# Command:
GET logstash_metric/_search
{
"sort": [
{
"@timestamp": {
"order": "desc"
}
}
],
"docvalue_fields": ["@timestamp","request", "response", "type", "bytes", "verb"
, "agent", "clientip"]
}
# Result:
{
"took": 0,
"timed_out": false,
"_shards": {
"total": 0,
"successful": 0,
"skipped": 0,
"failed": 0
```

```
},
"hits": {
"total": 0,
"max_score": 0,
"hits": []
}
}
# Here, you can see that the result is empty, indicating that there is no data written into CTSDB. View Logstash logs:
[2018-05-25T21:00:07,081][ERROR][logstash.outputs.elasticsearch] Encountered a retryable error. Will Retry with exponential backoff {:code=>403, :url=>"http://127.0.0.1:9200/_bulk"}
[2018-05-25T21:00:07,081][ERROR][logstash.outputs.elasticsearch] Encountered a retryable error. Will Retry with exponential backoff {:code=>403, :url=>"http://127.0.0.1:9200/_bulk"}
# `bulk` processing was exceptional, and permission error 403 was returned. However, a careful check of the username and password found no problem. Continue to check the Elasticsearch logs:
[2018-05-25T20:59:27,545][WARN ][o.e.p.o.OPackActionFilter] [1505480279000001609] process index failed: Invalid format: "2018-05-25T12:51:18.905Z"
[2018-05-25T20:59:27,547][WARN ][o.e.p.o.OPackActionFilter] [1505480279000001609] process index failed: Invalid format: "2018-05-25T12:51:18.905Z"
# The Elasticsearch logs show that an error occurred whiling parsing the time format, and the cause is that the time field format was not specified during metric creation. The default time format in CTSDB is `epoch_millis`, therefore, you need to change the time format:
POST /_metric/logstash_metric/update?pretty
{
"time": {
"name": "@timestamp",
"format": "strict_date_optional_time"
}
}
# Restart Logstash and Filebeat, write data again, and check CTSDB:
# Result
{
"took": 2,
"timed_out": false,
"_shards": {
"total": 3,
"successful": 3,
"skipped": 0,
"failed": 0
},
"hits": {
"total": 100,
"max_score": null,
```

```
"hits": [
{
"_index" : "logstash_metric@1527004800000_30",
"_type" : "doc",
"_id" : "AWOvG0YgQoCkIV2BLcov",
"_score" : null,
"fields" : {
"request" : [
"/blog/tags/puppet?flav=rss20"
],
"agent" : [
"\"UniversalFeedParser/4.2-pre-314-svn +http://feedparser.org/\""
],
"@timestamp" : [
1527651156725
],
"response" : [
"200"
],
"bytes" : [
14872
],
"clientip" : [
"46.105.14.53"
],
"verb" : [
"GET"
],
"type" : [
"log"
]
},
"sort" : [
1527651156725
]
},
... ...
# As the content is too much, it is omitted here. In `hits.total`, you can see
that the query hit 100 documents, indicating that all the 100 logs have been su
ccessfully written into CTSDB
```

As can be seen from the above sample, logs are collected by Filebeat into Logstash, parsed by Logstash into multiple fields, and then written into CTSDB.

Kibana

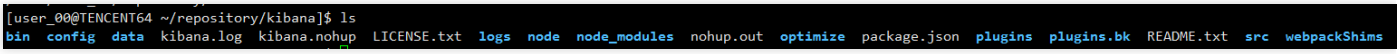
Kibana is an open-source analysis and visualization platform designed for Elasticsearch. You can use it to search, view, and interact with the data stored in CTSDB metrics. Its diverse features such as graphs, forms, and curves help you visualize and analyze data easily.

Kibana directions

1. **Install**
   Download a Kibana version matching Elasticseach and decompress it into a directory.
2. **Configure**
   Kibana configuration is simple, and a sample will be provided in the following section. For descriptions of the configuration items, please see Configure Kibana.
3. **Run**
   When Kibana runs, `config/kibana.yml` is used as the configuration by default.

Kibana usage example

1. First, decompress the Kibana installation package into a directory as shown below:



2. Then, modify the configuration file under `config` as follows:

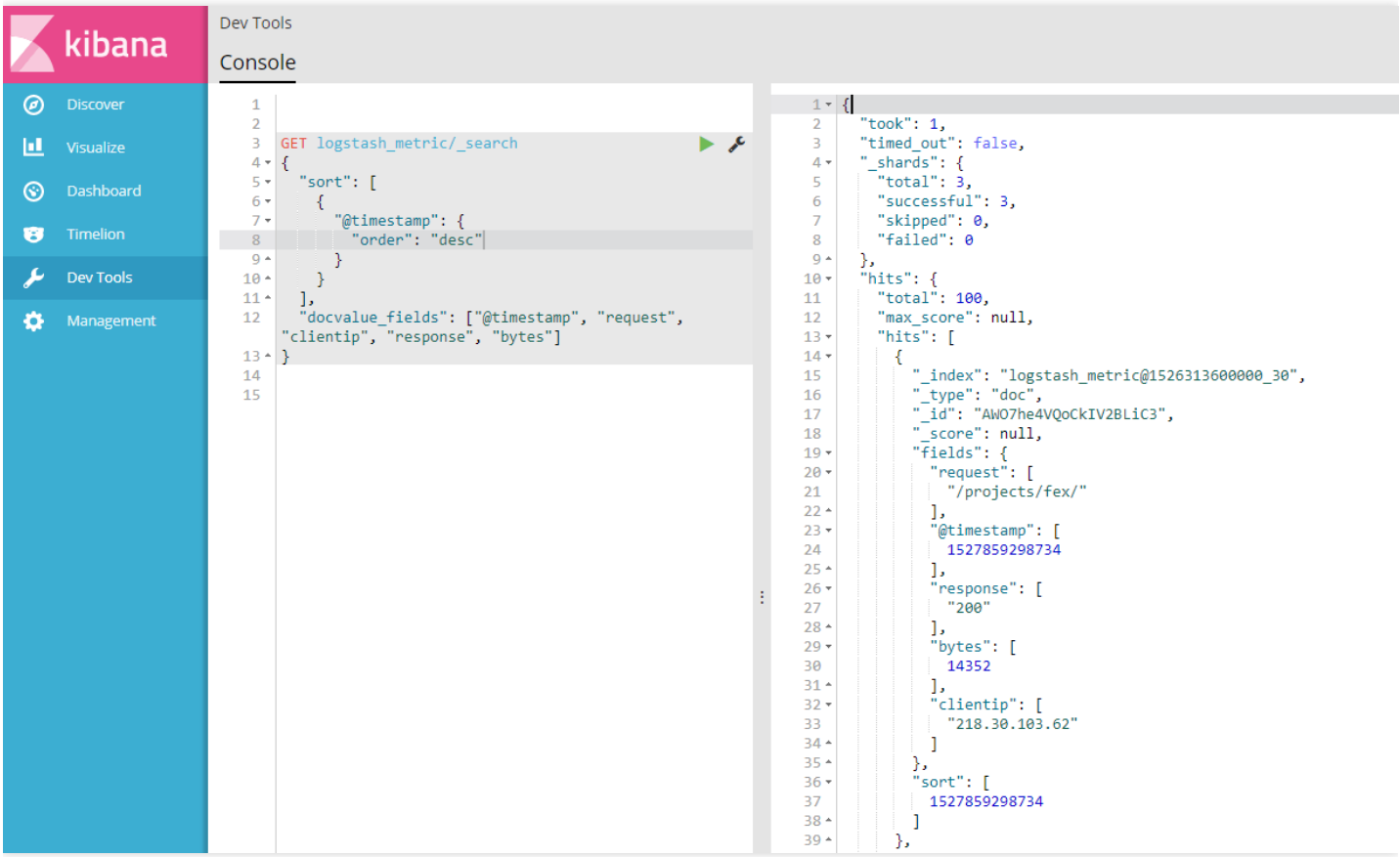
```
# config/kibana.yml
# Port on which the Kibana server listens
server.port: 5601
# IP of the server to which the Kibana server is bound
server.host: 127.0.0.1
# URL of the CTSDB instance to be connected to
elasticsearch.url: "http://127.0.0.1:9200"
# For CTSDB instances that require authentication, you need to enter the userna
me and password
elasticsearch.username: "root"
elasticsearch.password: "changeme"
```

- Start Kibana and access it in a browser

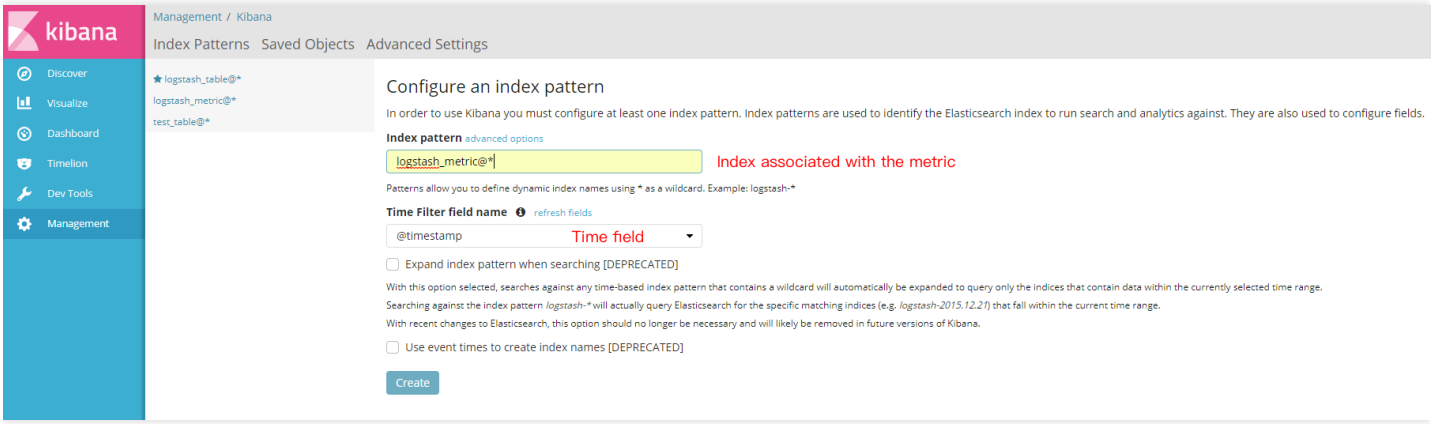
```
nohup bin/kibana &
```

Access the Kibana server through `IP:port` or the domain name as shown below:

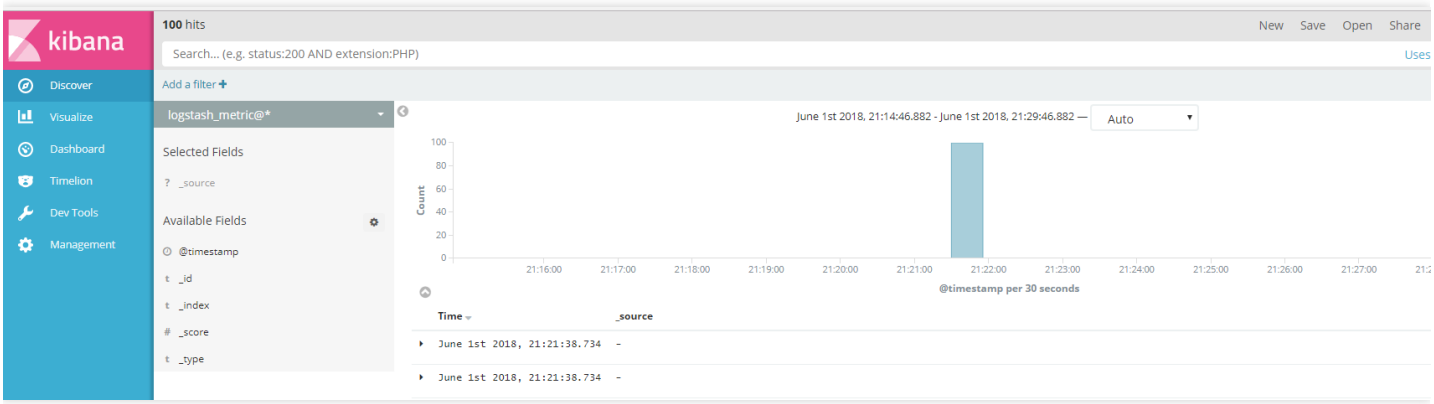
You can use the development tool to conveniently access CTSDB as shown below:



Create the index to be accessed on the management page as shown below:

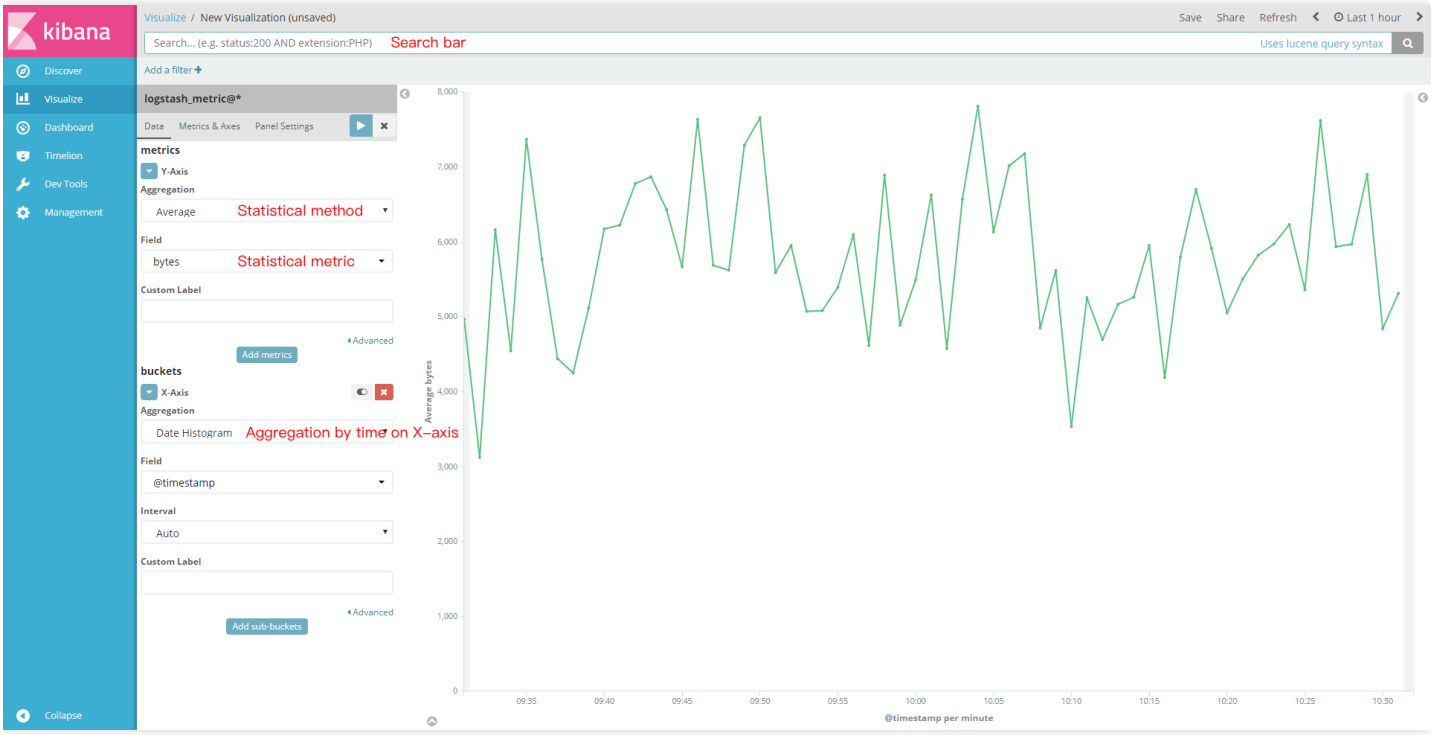


If you need to search for logs, you may use the search feature as shown below:

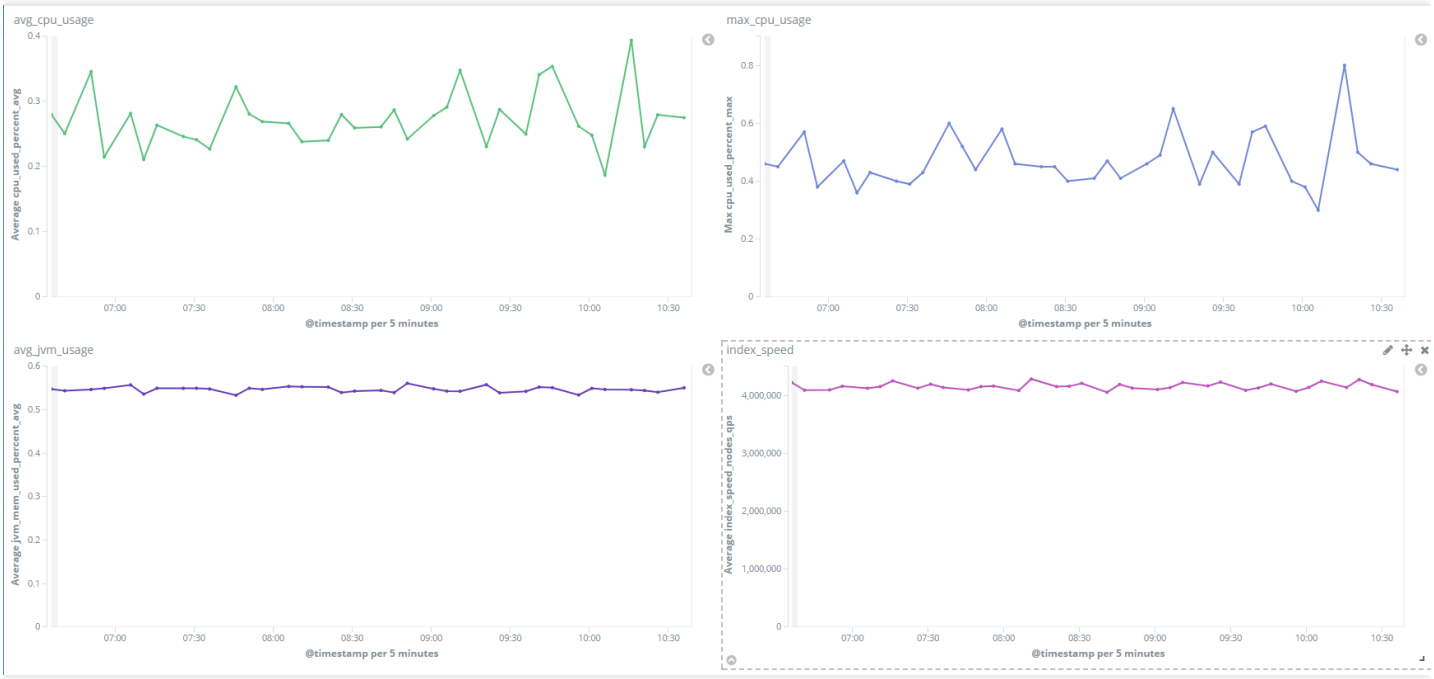


You can see that the search results show nothing but time. This is because the source feature is disabled by default in CTSDB to save storage space. If you need to search for logs, please contact us to enable the source feature. The effect after the feature is enabled is as shown below:

By using visualization, you can create various graphs. Here, a line chart is used as an example to show the Kibana graph effect:



Kibana not only provides a rich set of visual graph features but also uses dashboards to display the saved visual graphs on the same page in a unified manner, so that you can easily check the changes of multiple metrics. Below is a sample dashboard view of the monitoring data for demonstrating the display effect:



Grafana

Grafana is an open-source dashboard tool that provides various graph features just like Kibana. With its fine-grained display effects, you can effectively analyze data. Grafana supports more data sources than Kibana, including InfluxDB, OpenTSDB, and Elasticsearch. The following example shows how to use it to analyze CTSDB data in a visualized way.

Grafana

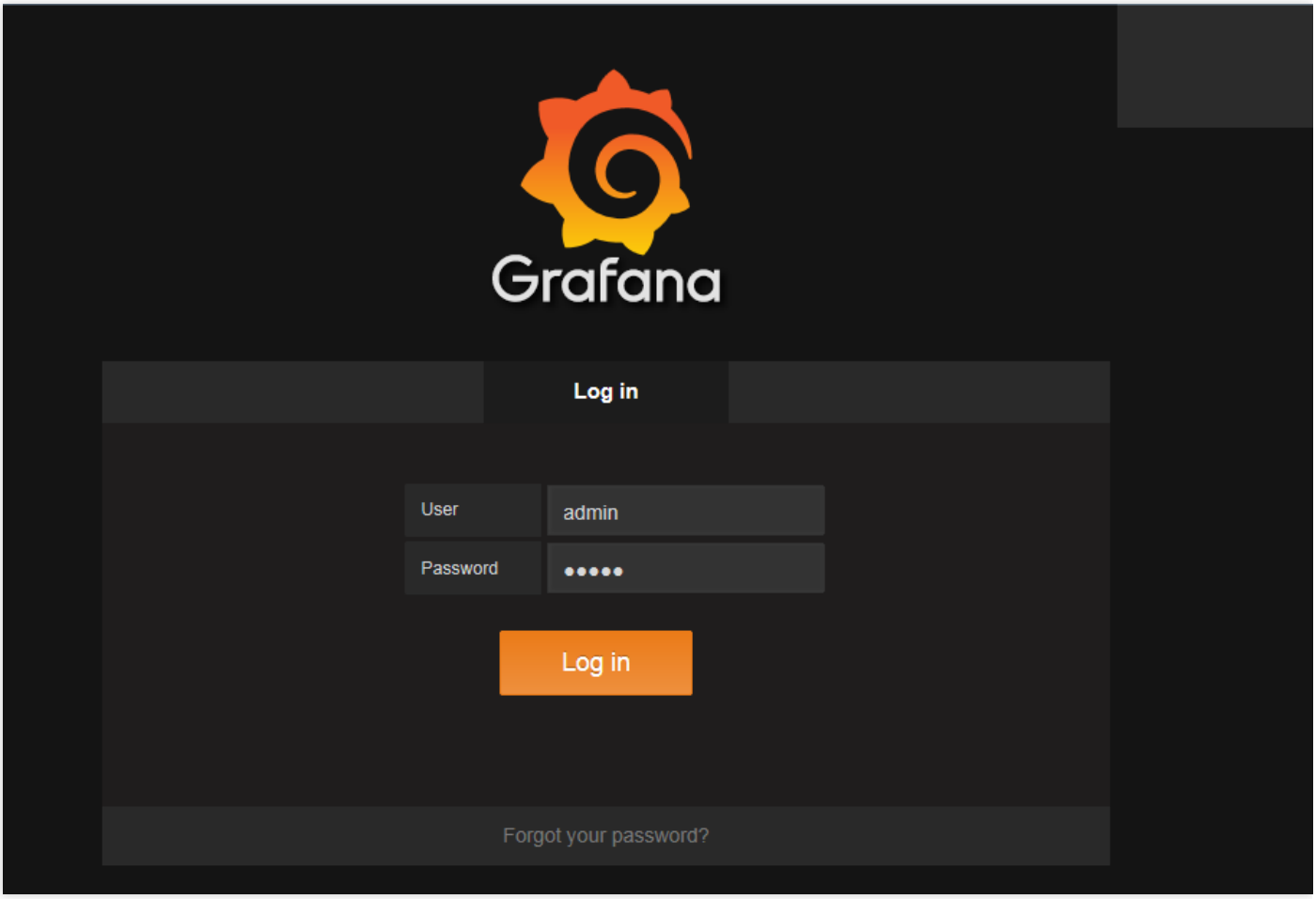
1. Install
   For more information on how to install Grafana, please see Install Grafana.
2. Configure
   Grafana has many configuration items. You can use the default configuration. For more information, please see Configuration.
3. Run
   When Grafana runs, it uses `/etc/grafana/grafana.ini` as its configuration by default.

Grafana usage example

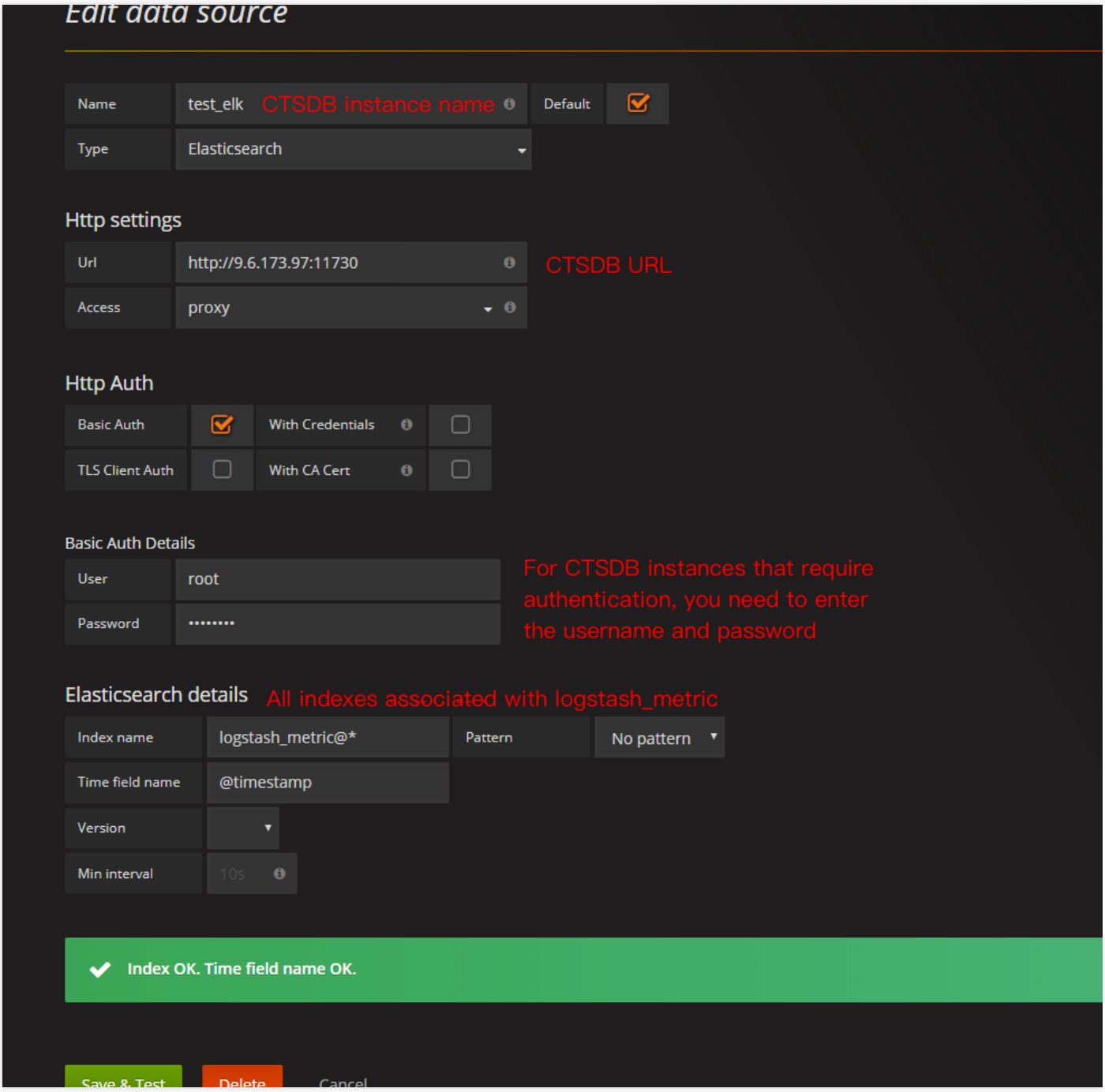
1. First, start the Grafana service.

```
sudo service grafana-server start
```

2. Then, access the Grafana service in a browser:



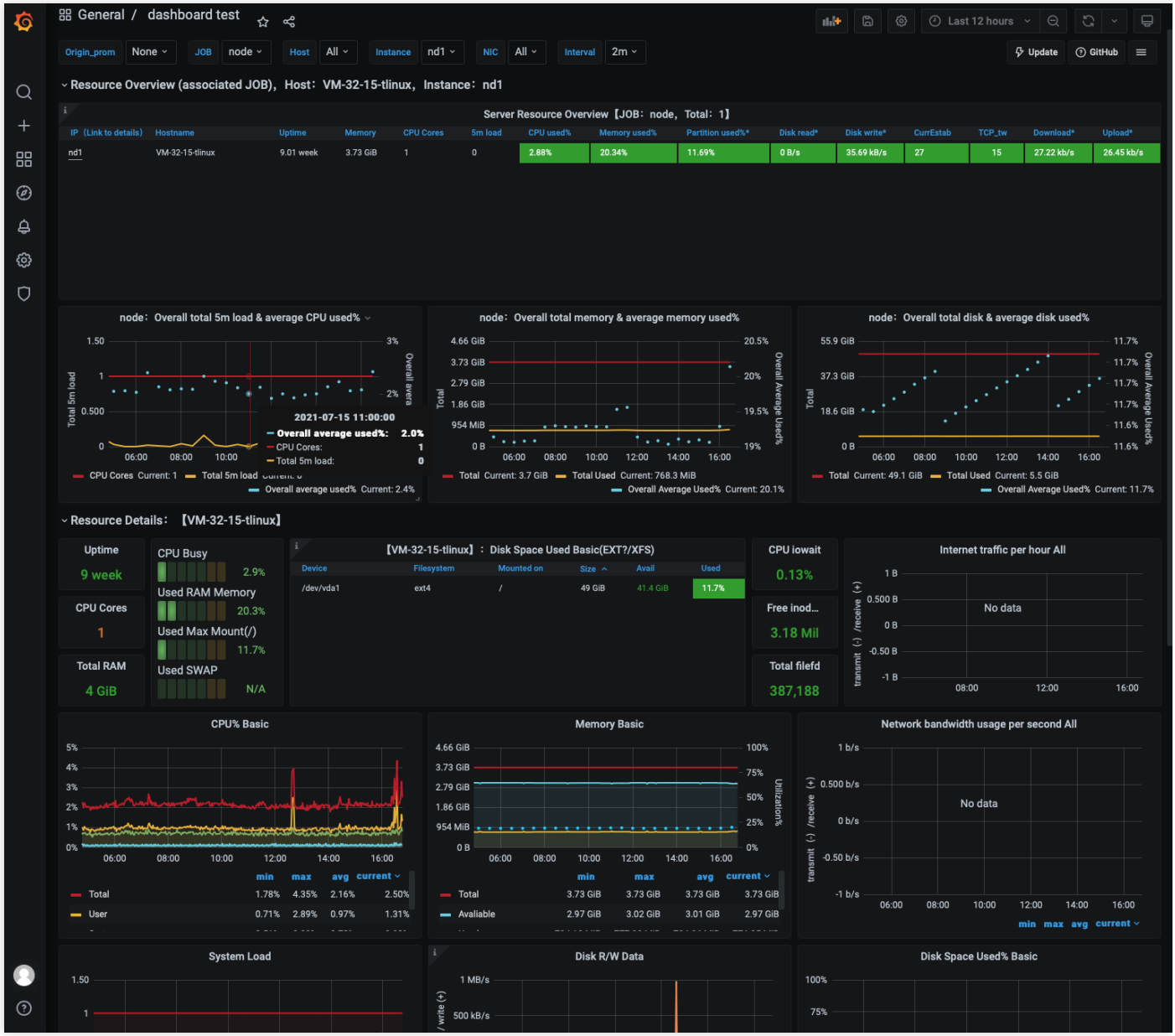
3. Create a data source and dashboard as shown below:



4. Use the dashboard to create a visual graph as shown below:

5. As shown above, the graph display effect of Grafana is slightly different from that of Kibana, but their features are essentially the same, so you can choose either one of them based on your use habits and preferences. Similarly,

Grafana dashboard can also display multiple visual graphs as shown below:



Summary

This document describes how to connect ELK ecosystem components and Grafana to CTSDB in detail. If you have any questions during use, please submit a ticket for assistance.

Importing MySQL Data into CTSDB

Last updated：2021-07-15 16:00:24

Foreword

CTSDB is a distributed and scalable time series database that supports near-real time data search and analysis. It is compatible with common Elasticsearch APIs. Many users may not find a good way to import MySQL data into CTSDB. Therefore, this document provides a simple and easy way to sync MySQL data into CTSDB.

Overview

go-mysql-elasticsearch is an open-source high-performance Elasticsearch tool developed in Go for syncing MySQL data. It is easy to compile and use, and it works in a very simple way: you first use mysqldump to get the current MySQL data, then use `name` and `position` in the current binlog to get the incremental data, and finally construct RESTful APIs based on the binlog to write the data into Elasticsearch. As CTSDB is developed based on Elasticsearch, it is perfectly compatible with go-mysql-elasticsearch for MySQL data import.

Syncing MySQL Data to CTSDB

MySQL sample data construction

As you want to import MySQL data into CTSDB, it is assumed that you have installed MySQL. To make the process complete, this document starts from the import of sample data. Here, a simple tool written in Go is used to generate some sample data and import it into MySQL with the following table structure:

```
mysql> desc test_table;
+-----------+-------------+------+-----+---------+----------------+
| Field | Type | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+----------------+
| id | int(11) | NO | PRI | NULL | auto_increment |
| timestamp | bigint(20) | YES | | NULL | |
| cpu_usage | float | YES | | NULL | |
| host_ip | varchar(20) | YES | | NULL | |
| region | varchar(20) | YES | | NULL | |
+-----------+-------------+------+-----+---------+----------------+
```

Use the above code to create a table named `test_table`. Then, import 2,000 records of sample data into the table. Some data records are as shown below:

```
mysql> select * from test_table;
+------+------------+----------+------------+-----------+
| id | timestamp | cpu_usage | host_ip | region|
+------+------------+----------+------------+-----------+
|1 | 1527676339 | 0.23 | 192.168.1.1 | beijing |
|2 | 1527676399 | 0.78 | 192.168.1.2 | shanghai |
|3 | 1527676459 | 0.2 | 192.168.1.3 | guangzhou |
|4 | 1527676519 | 0.47 | 192.168.1.4 | shanghai |
|5 | 1527676579 | 0.13 | 192.168.1.5 | beijing |
|6 | 1527676639 | 0.15 | 192.168.1.1 | beijing |
|7 | 1527676699 | 0.07 | 192.168.1.2 | shanghai |
|8 | 1527676759 | 0.17 | 192.168.1.3 | guangzhou |
|9 | 1527676819 | 0.94 | 192.168.1.4 | shanghai |
|10| 1527676879 | 0.06 | 192.168.1.5 | beijing |
```

At this point, MySQL sample data has been prepared.

Creating metric in CTSDB

Create a metric in CTSDB with the same structure as that of the MySQL table by using the following API to store the corresponding data:

```
POST /_metric/test_metric
{
"time": {
"name": "timestamp", # Common time field in CTSDB, which corresponds to `timestam
p` in the MySQL table
"format": "strict_date_optional_time || epoch_second"
},
"tags": {
"region": "string",
"host_ip": "string"
},
"fields": {
"cpu_usage": "float" # The `fields` field indicates the metric column. Obviously,
`cpu_usage` indicates the CPU utilization metric that needs to be monitored
}
}
```

At this point, the CTSDB metric structure has been prepared. Then, use go-mysql-elasticsearch to sync data.

go-mysql-elasticsearch usage

As go-mysql-elasticsearch is developed in Go, you first need to install Go on a version above 1.6. Go installation is very simple. You can download it at https://golang.org/dl/ and install it as instructed in Download and install. The entire process is as follows:

```
$ go get github.com/siddontang/go-mysql-elasticsearch
$ cd $GOPATH/src/github.com/siddontang/go-mysql-elasticsearch
$ make
```

After the tool is installed, you need to properly configure it first before using it. The following is a configuration sample with corresponding comments:

```
# Note: the default configuration file of go-mysql-elasticsearch is `go-mysql-ela
sticsearch/etc/river.toml`
# MySQL address, user and password
# user must have replication privilege in MySQL.
my_addr = "127.0.0.1:3306"
my_user = "root"
my_pass = "123456"
my_charset = "utf8"
# Set true when elasticsearch use https
#es_https = false
# CTSDB address
es_addr = "9.6.174.42:13982"
# For CTSDB instances that require authentication, you need to set the username a
nd password
es_user = "root"
es_pass = "changeme"
# Path to store data, like master.info, if not set or empty,
# we must use this to support breakpoint resume syncing.
# TODO: support other storage, like etcd.
data_dir = "./var" # Binlog name and location
# Inner http status address
stat_addr = "127.0.0.1:12800"
# pseudo server id like a slave
server_id = 1001
# mysql or mariadb
flavor = "mysql"
# mysqldump execution path
# if not set or empty, ignore mysqldump.
mysqldump = "mysqldump"
# minimal items to be inserted in one bulk
bulk_size = 512
# force flush the pending requests if we don't have enough items >= bulk_size
flush_bulk_time = "200ms"
# Ignore table without primary key
```

```
skip_no_pk_table = true
# MySQL data source
[[source]]
schema = "mysql_es"
# Only below tables will be synced into Elasticsearch.
# "t_[0-9]{4}" is a wildcard table format, you can use it if you have many sub ta
bles, like table_0000 - table_1023
# I don't think it is necessary to sync all tables in a database.
tables = ["test_*"]
[[rule]]
schema = "mysql_es" # MySQL database name
table = "test_table" # MySQL table name
index = "test_metric" # CTSDB metric name
type = "doc" # Document type
```

The above configuration is only for test. If you have more advanced requirements, please see the official documentation for proper configuration. After completing the configuration, run go-mysql-elasticsearch as follows:

```
$ ./bin/go-mysql-elasticsearch -config=./etc/river.toml
2018/05/31 21:43:44 INFO create BinlogSyncer with config {1001 mysql 127.0.0.1 33
06 root utf8 false false <nil> false false 0 0s 0s 0}
2018/05/31 21:43:44 INFO run status http server 127.0.0.1:12800
2018/05/31 21:43:44 INFO skip dump, use last binlog replication pos (mysql-bin.00
0002, 194296) or GTID %!s(<nil>)
2018/05/31 21:43:44 INFO begin to sync binlog from position (mysql-bin.000002, 19
4296)
2018/05/31 21:43:44 INFO register slave for master server 127.0.0.1:3306
2018/05/31 21:43:44 INFO start sync binlog at binlog file (mysql-bin.000002, 1942
96)
2018/05/31 21:43:44 INFO rotate to (mysql-bin.000002, 194296)
2018/05/31 21:43:44 INFO rotate binlog to (mysql-bin.000002, 194296)
2018/05/31 21:43:44 INFO save position (mysql-bin.000002, 194296)
```

Note that as go-mysql-elasticsearch needs to use binlogs in row-based format, you must configure the following parameters in MySQL:

```
# Configure the following parameters; otherwise, errors will definitely occur
log_bin=mysql-bin
binlog_format = ROW
server-id=1
```

Now, check whether the MySQL data has been successfully imported into CTSDB:

```
GET test_metric/_search?size=1000
{
```

```
"sort": [
{
"timestamp": {
"order": "desc"
}
}
],
"docvalue_fields": ["timestamp", "host_ip", "region", "cpu_usage"]
}
# Result:
{
"took": 8,
"timed_out": false,
"_shards": {
"total": 3,
"successful": 3,
"skipped": 0,
"failed": 0
},
"hits": {
"total": 2000,
"max_score": null,
"hits": [
{
"_index": "test_metric@1525363200000_30",
"_type": "doc",
"_id": "2000",
"_score": null,
"fields": {
"host_ip": [
"192.168.1.5"
],
"region": [
"beijing"
],
"cpu_usage": [
0.05000000074505806
],
"timestamp": [
1527807286000
]
},
"sort": [
1527807286000
]
},
```

```
......
}
```

Summary

As you can see, with go-mysql-elasticsearch, you only need to write rules in the configuration file to easily sync MySQL data to Elasticsearch. The above section only provides some simple examples. If you have more needs, please see the official documentation of go-mysql-elasticsearch.

In addition to the tool described in this document, two other tools are also recommended:

- py-mysql-elasticsearch-sync. It is written in Python, and it works in a way similar to go-mysql-elasticsearch, as both of them use binlogs to sync data. For more information on how to install and use it, please see py-mysql-elasticsearch-sync.
- Logstash. To use it to sync data, you need to install the `logstash-input-jdbc` and `logstash-output-elasticsearch` plugins. For more information on how to use it, please see Jdbc input plugin and Elasticsearch output plugin.
  If you have any questions when using the above tools, please submit a ticket for assistance.

Quickly Selecting Instance

Last updated：2021-07-15 16:00:24

CTSDB supports two configuration modes: QuickConfig and CustomConfig, as detailed below. We recommend you select an appropriate instance specification based on your needs in three aspects: performance, capacity, and other factors. For instance performance data, please see Performance.

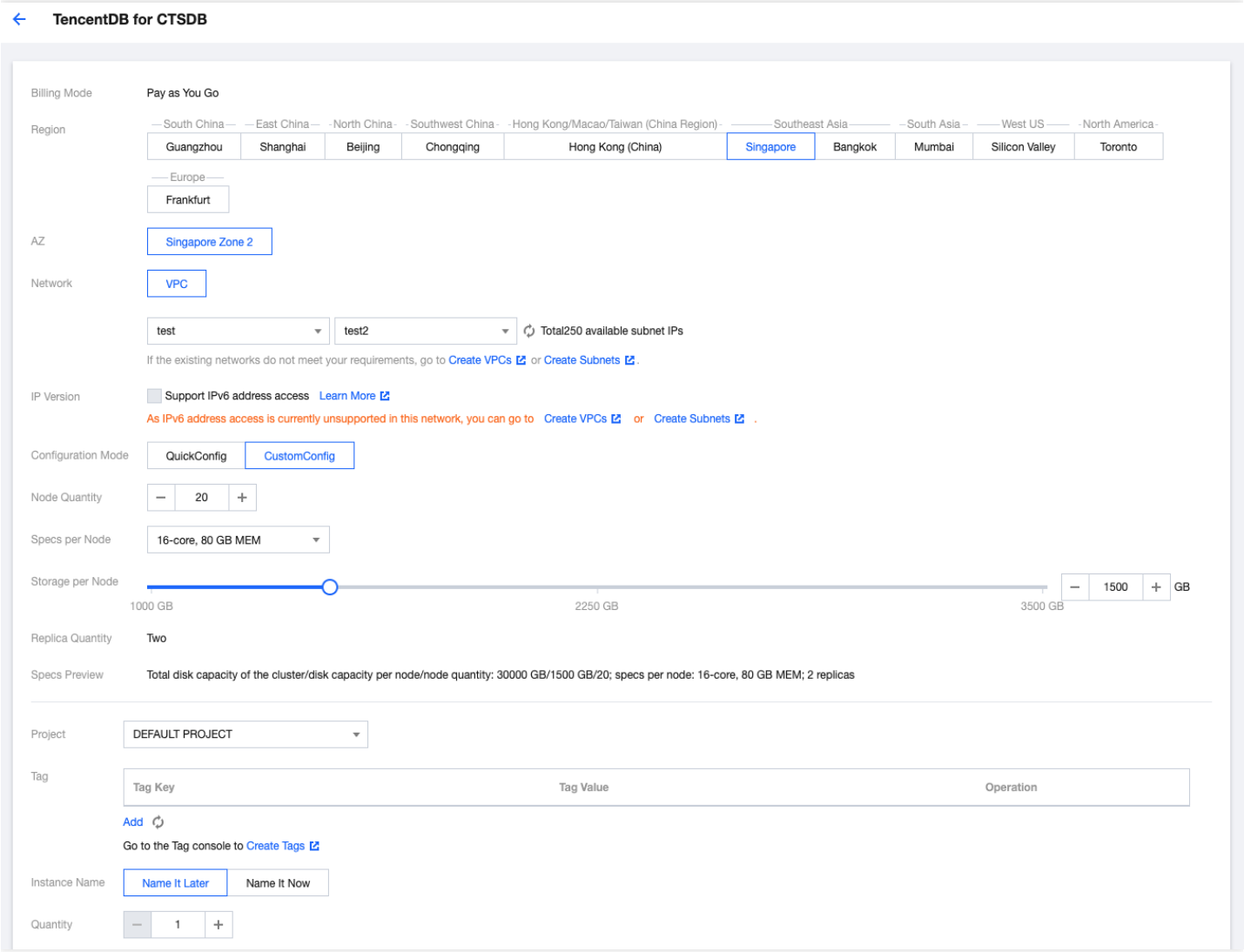
Go to the CTSDB purchase page and select the mode as desired.

QuickConfig

CTSDB provides the QuickConfig mode, where you only need to select the volume of data to be stored, and the system will automatically create an instance with three nodes and two replicas. In addition, when the system automatically selects the node configuration, under the premise that the nodes have the same storage capacity, it will preferably use the node configuration with fewer CPU cores and smaller memory for the instance.

CustomConfig

In CustomConfig mode, you can select the number of nodes, node specification, and node capacity as desired as shown below:



Note：

Be cautious when selecting a two-node instance, as it cannot fully ensure data integrity due to the characteristics of a distributed system.