附录 C Kong Admin API

Kong 提供了 RESTful Admin API 来管理整个 Kong 集群。通过它可以使所有 Kong 集群的运行数据和配置数据保持一致。

Kong 的管理端口如下。

- 8001：是 Admin API 监听的默认端口。
- 8444：是 Admin API 的 HTTPS 流量的默认端口。

Kong 支持的内容类型如下。

- application/x-www-form-urlencoded
- application/json

Kong Admin API 主要分为 9 大类，包括服务、路由、上游、目标、消费者、插件、证书、SNI 和节点的参数信息，请读者结合 2.5 章节学习，接下来详细介绍它们。

A. 服务

1.1.2 添加服务

名称	地址	谓词
添加服务	/services	POST
添加与证书关联的服务	/certificates/{certificate name or id}/services	POST

请求体字段和 URL 参数：

请求体字段包括 name、retries、protocol、host、port、path、connect_timeout、write_timeout、read_timeout、tags、client_certificate、url。

URL 参数：

- certificate name or id（证书名称或 ID）

响应内容：

HTTP 状态码 201（表示 Created，即已创建）。

添加服务后的返回内容如下：

```
{
    "id": "95eb9ed9-e2bd-4220-9200-1aea005eadf8",
    "created_at": 1422386534,
    "updated_at": 1422386534,
    "name": "my-service",
    "retries": 5,
    "protocol": "http",
    "host": "example.com",
    "port": 80,
    "path": "/some_api",
    "connect_timeout": 60000,
    "write_timeout": 60000,
    "read_timeout": 60000,
    "tags": ["user-level", "low-priority"],
    "client_certificate": {"id":"c0cf31a9-2623-477e-a194-6b812eb79fde"}
}
```

1. 列出服务

名称	地址	谓词
列出所有服务	/services	GET
列出与证书相关的服务	/certificates/{certificate name or id}/services	GET

请求 URL 参数：

- certificate name or id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

列出服务后的返回内容如下：

```
{
```

```
"data": [{
    "id": "a5fb8d9b-a99d-40e9-9d35-72d42a62d83a",
    "created_at": 1422386534,
    "updated_at": 1422386534,
    "name": "my-service",
    "retries": 5,
    "protocol": "http",
    "host": "example.com",
    "port": 80,
    "path": "/some_api",
    "connect_timeout": 60000,
    "write_timeout": 60000,
    "read_timeout": 60000,
    "tags": ["user-level", "low-priority"],
    "client_certificate": {"id":"51e77dc2-8f3e-4afa-9d0e-0e3bbbcfd515"}
}, {
    "id": "fc73f2af-890d-4f9b-8363-af8945001f7f",
    "created_at": 1422386534,
    "updated_at": 1422386534,
    "name": "my-service",
    "retries": 5,
    "protocol": "http",
    "host": "example.com",
    "port": 80,
    "path": "/another_api",
    "connect_timeout": 60000,
    "write_timeout": 60000,
    "read_timeout": 60000,
    "tags": ["admin", "high-priority", "critical"],
```

```
    "client_certificate": {"id":"4506673d-c825-444c-a25b-602e3c2ec16e"}
}],
    "next":
"http://localhost:8001/services?offset=a3e68c47-9659-4842-8f49-56b04d2791bf"
}
```

2. 检索服务

名称	地址	谓词
检索服务	/services/{service name or id}	GET
检索与证书相关联的服务	/certificates/{certificate id}/services/{service name or id}	GET
检索与路由相关联的服务	/routes/{route name or id}/service	GET
检索与插件相关联的服务	/plugins/{plugin id}/service	GET

请求 URL 参数：

- service name or id
- certificate id
- route name or id
- plugin id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

检索服务后的返回内容如下：

```
{
    "id": "95eb9ed9-e2bd-4220-9200-1aea005eadf8",
    "created_at": 1422386534,
    "updated_at": 1422386534,
    "name": "my-service",
```

```
    "retries": 5,
    "protocol": "http",
    "host": "example.com",
    "port": 80,
    "path": "/some_api",
    "connect_timeout": 60000,
    "write_timeout": 60000,
    "read_timeout": 60000,
    "tags": ["user-level", "low-priority"],
    "client_certificate": {"id":"c0cf31a9-2623-477e-a194-6b812eb79fde"}
}
```

3. 更新服务

名称	地址	谓词
更新服务	/services/{service name or id}	PATCH
更新与证书相关联的服务	/certificates/{certificate id}/services/{service name or id}	PATCH
更新与路由相关联的服务	/routes/{route name or id}/service	PATCH
更新与插件相关联的服务	/plugins/{plugin id}/service	PATCH

请求体字段和 URL 参数：

请求体字段包括 name、retries、protocol、host、port、path、connect_timeout、write_timeout、read_timeout、tags、client_certificate、url。

URL 参数：

- service name or id
- certificate id
- route name or id

- plugin id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

更新服务后的返回内容如下：

```
{
    "id": "95eb9ed9-e2bd-4220-9200-1aea005eadf8",
    "created_at": 1422386534,
    "updated_at": 1422386534,
    "name": "my-service",
    "retries": 5,
    "protocol": "http",
    "host": "example.com",
    "port": 80,
    "path": "/some_api",
    "connect_timeout": 60000,
    "write_timeout": 60000,
    "read_timeout": 60000,
    "tags": ["user-level", "low-priority"],
    "client_certificate": {"id":"c0cf31a9-2623-477e-a194-6b812eb79fde"}
}
```

4. 更新或添加服务

名称	地址	谓词
更新或添加服务	/services/{service name or id}	PUT
更新或添加与证书相关联的服务	/certificates/{certificate id}/services/{service name or id}	PUT
更新或添加与路由相关联的服务	/routes/{route name or id}/service	PUT

更新或添加与插件相关联的服务	/plugins/{plugin id}/service	PUT

请求体字段和 URL 参数：

请求体字段包括 name、retries、protocol、host、port、path、connect_timeout、write_timeout、read_timeout、tags、client_certificate、url。

URL 参数：

- service name or id
- certificate id
- route name or id
- plugin id

响应内容：

HTTP 状态码 201（表示 Created，即已创建）或 200（表示 OK，即正常）。

更新或添加服务后的返回内容：

```
{
    "id": "95eb9ed9-e2bd-4220-9200-1aea005eadf8",
    "created_at": 1422386534,
    "updated_at": 1422386534,
    "name": "my-service",
    "retries": 5,
    "protocol": "http",
    "host": "example.com",
    "port": 80,
    "path": "/some_api",
    "connect_timeout": 60000,
    "write_timeout": 60000,
    "read_timeout": 60000,
    "tags": ["user-level", "low-priority"],
    "client_certificate": {"id":"c0cf31a9-2623-477e-a194-6b812eb79fde"}
}
```

5. 删除服务

名称	地址	谓词
删除服务	/services/{service name or id}	DELETE
删除与证书相关联的服务	/certificates/{certificate id}/services/{service name or id}	DELETE
删除与路由相关联的服务	/routes/{route name or id}/service	DELETE

请求 URL 参数：

- service name or id
- certificate id
- route name or id

响应内容：

HTTP 状态码 204（表示 No Content，即无内容）。

B. 路由

1. 添加路由

名称	地址	谓词
添加路由	/routes	POST
添加与服务相关联的路由	/services/{service name or id}/routes	POST

请求体字段和 URL 参数：

请求体字段包括 name、protocols、methods、hosts、paths、headers、https_redirect_status_code、regex_priority、strip_path、preserve_host、snis、sources、destinations、tags 和 service。

URL 参数：

- service name or id

响应内容：

HTTP 状态码 201（表示 Created，即已创建）。

添加路由后的返回内容：


```
{
    "id": "98e78077-b16f-47e1-a91a-4e9123ba9cd9",
    "created_at": 1422386534,
    "updated_at": 1422386534,
    "name": "my-route",
    "protocols": ["http", "https"],
    "methods": ["GET", "POST"],
    "hosts": ["example.com", "foo.test"],
    "paths": ["/foo", "/bar"],
    "headers": {"x-another-header":["bla"], "x-my-header":["foo", "bar"]},
    "https_redirect_status_code": 426,
    "regex_priority": 0,
    "strip_path": true,
    "path_handling": "v0",
    "preserve_host": false,
    "tags": ["user-level", "low-priority"],
    "service": {"id":"79bce5e1-ae69-4462-9973-88e17cc5dfff"}
}
```


2. 列出路由

名称	地址	谓词
列出所有路由	/routes	GET
列出与服务相关联的路由	/services/{service name or id}/routes	GET

请求 URL 参数：

- service name or id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

列出路由后的返回内容：


```
{
"data": [{
    "id": "a9daa3ba-8186-4a0d-96e8-00d80ce7240b",
    "created_at": 1422386534,
    "updated_at": 1422386534,
    "name": "my-route",
    "protocols": ["http", "https"],
    "methods": ["GET", "POST"],
    "hosts": ["example.com", "foo.test"],
    "paths": ["/foo", "/bar"],
    "headers": {"x-another-header":["bla"], "x-my-header":["foo", "bar"]},
    "https_redirect_status_code": 426,
    "regex_priority": 0,
    "strip_path": true,
    "path_handling": "v0",
    "preserve_host": false,
    "tags": ["user-level", "low-priority"],
    "service": {"id":"127dfc88-ed57-45bf-b77a-a9d3a152ad31"}
}, {
    "id": "9aa116fd-ef4a-4efa-89bf-a0b17c4be982",
    "created_at": 1422386534,
    "updated_at": 1422386534,
    "name": "my-route",
    "protocols": ["tcp", "tls"],
    "https_redirect_status_code": 426,
    "regex_priority": 0,
```

```
    "strip_path": true,
    "path_handling": "v0",
    "preserve_host": false,
    "snis": ["foo.test", "example.com"],
    "sources": [{"ip":"10.1.0.0/16", "port":1234}, {"ip":"10.2.2.2"}, {"port":9123}],
    "destinations":    [{"ip":"10.1.0.0/16",    "port":1234},    {"ip":"10.2.2.2"},
{"port":9123}],
    "tags": ["admin", "high-priority", "critical"],
    "service": {"id":"ba641b07-e74a-430a-ab46-94b61e5ea66b"}
}],
    "next":
"http://localhost:8001/routes?offset=a3e68c47-9659-4842-8f49-56b04d2791bf"
}
```


3. 检索路由

名称	地址	谓词
检索路由	/routes/{route name or id}	GET
检索与服务关联的路由	/services/{service name or id}/routes/{route name or id}	GET
检索与插件相关联的路由	/plugins/{plugin id}/route	GET

请求 URL 参数：

- route name or id
- service name or id
- plugin id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

检索路由后的返回内容：

```
{
    "id": "98e78077-b16f-47e1-a91a-4e9123ba9cd9",
    "created_at": 1422386534,
    "updated_at": 1422386534,
    "name": "my-route",
    "protocols": ["http", "https"],
    "methods": ["GET", "POST"],
    "hosts": ["example.com", "foo.test"],
    "paths": ["/foo", "/bar"],
    "headers": {"x-another-header":["bla"], "x-my-header":["foo", "bar"]},
    "https_redirect_status_code": 426,
    "regex_priority": 0,
    "strip_path": true,
    "path_handling": "v0",
    "preserve_host": false,
    "tags": ["user-level", "low-priority"],
    "service": {"id":"79bce5e1-ae69-4462-9973-88e17cc5dfff"}
}
```

4. 更新路由

名称	地址	谓词
更新路由	/routes/{route name or id}	PATCH
更新与服务相关联的路由	/services/{service name or id}/routes/{route name or id}	PATCH
更新与插件相关联的路由	/plugins/{plugin id}/route	PATCH

请求体字段和 URL 参数：

请求体字段包括 name、protocols、methods、hosts、paths、headers、

https_redirect_status_code、 regex_priority、 strip_path、 preserve_host、 snis、 sources、 destinations、tags 和 service。

URL 参数：

- route name or id
- service name or id
- plugin id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

更新路由后的返回内容：

```
{
    "id": "98e78077-b16f-47e1-a91a-4e9123ba9cd9",
    "created_at": 1422386534,
    "updated_at": 1422386534,
    "name": "my-route",
    "protocols": ["http", "https"],
    "methods": ["GET", "POST"],
    "hosts": ["example.com", "foo.test"],
    "paths": ["/foo", "/bar"],
    "headers": {"x-another-header":["bla"], "x-my-header":["foo", "bar"]},
    "https_redirect_status_code": 426,
    "regex_priority": 0,
    "strip_path": true,
    "path_handling": "v0",
    "preserve_host": false,
    "tags": ["user-level", "low-priority"],
    "service": {"id":"79bce5e1-ae69-4462-9973-88e17cc5dfff"}
}
```

5. 更新或添加路由

名称	地址	谓词
更新或添加路由	/routes/{route name or id}	PUT
更新或添加与服务相关联的路由	/services/{service name or id}/routes/{route name or id}	PUT
更新或添加与插件相关联的路由	/plugins/{plugin id}/route	PUT

请求体字段和 URL 参数：

请求体字段包括 name、protocols、methods、hosts、paths、headers、https_redirect_status_code、regex_priority、strip_path、preserve_host、snis、sources、destinations、tags 和 service。

URL 参数：

- route name or id
- service name or id
- plugin id

响应内容：

HTTP 状态码 201（表示 Created，即已创建）或 200（表示 OK，即正常）。

更新或添加路由后的返回内容：

```
{
    "id": "98e78077-b16f-47e1-a91a-4e9123ba9cd9",
    "created_at": 1422386534,
    "updated_at": 1422386534,
    "name": "my-route",
    "protocols": ["http", "https"],
    "methods": ["GET", "POST"],
    "hosts": ["example.com", "foo.test"],
    "paths": ["/foo", "/bar"],
```

```
    "headers": {"x-another-header":["bla"], "x-my-header":["foo", "bar"]},
    "https_redirect_status_code": 426,
    "regex_priority": 0,
    "strip_path": true,
    "path_handling": "v0",
    "preserve_host": false,
    "tags": ["user-level", "low-priority"],
    "service": {"id":"79bce5e1-ae69-4462-9973-88e17cc5dfff"}
}
```

6. 删除路由

名称	地址	谓词
删除路由	/routes/{route name or id}	DELETE
删除与服务相关联的路由	/services/{service name or id}/routes/{route name or id}	DELETE

请求 URL 参数：

- route name or id
- service name or id

响应内容：

HTTP 状态码 204（表示 No Content，即无内容）。

C. 上游

1. 添加上游

名称	地址	谓词
添加上游	/upstreams	POST

请求 Body 字段：

- 请读者参照 2.5.3 章节表格字段**响应内容：**

HTTP 状态码 201（表示 Created，即已创建）。

添加上游后的返回内容：


```
{
    "id": "087410b7-7728-41bd-96a4-fcd00fb54ce1",
    "created_at": 1422386534,
    "name": "my-upstream",
    "algorithm": "round-robin",
    "hash_on": "none",
    "hash_fallback": "none",
    "hash_on_cookie_path": "/",
    "slots": 10000,
    "healthchecks": {
        "active": {
            "https_verify_certificate": true,
            "unhealthy": {
                "http_statuses": [429, 404, 500, 501, 502, 503, 504, 505],
                "tcp_failures": 0,
                "timeouts": 0,
                "http_failures": 0,
                "interval": 0
            },
            "http_path": "/",
            "timeout": 1,
            "healthy": {
                "http_statuses": [200, 302],
                "interval": 0,
                "successes": 0
            },
            "https_sni": "example.com",
```

```
            "concurrency": 10,
            "type": "http"
        },
        "passive": {
            "unhealthy": {
                "http_failures": 0,
                "http_statuses": [429, 500, 503],
                "tcp_failures": 0,
                "timeouts": 0
            },
            "type": "http",
            "healthy": {
                "successes": 0,
                "http_statuses": [200, 201, 202, 203, 204, 205, 206, 207, 208, 226, 300,
301, 302, 303, 304, 305, 306, 307, 308]
            }
        },
        "threshold": 0
    },
    "tags": ["user-level", "low-priority"],
    "host_header": "example.com"
}
```


2. 列出上游

名称	地址	谓词
列出所有上游	/upstreams	GET

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

列出上游后的返回内容：


```
{
"data": [{
   "id": "ea29aaa3-3b2d-488c-b90c-56df8e0dd8c6",
   "created_at": 1422386534,
   "name": "my-upstream",
   "algorithm": "round-robin",
   "hash_on": "none",
   "hash_fallback": "none",
   "hash_on_cookie_path": "/",
   "slots": 10000,
   "healthchecks": {
      "active": {
         "https_verify_certificate": true,
         "unhealthy": {
            "http_statuses": [429, 404, 500, 501, 502, 503, 504, 505],
            "tcp_failures": 0,
            "timeouts": 0,
            "http_failures": 0,
            "interval": 0
         },
         "http_path": "/",
         "timeout": 1,
         "healthy": {
            "http_statuses": [200, 302],
            "interval": 0,
            "successes": 0
         },
         "https_sni": "example.com",
```

```
            "concurrency": 10,
            "type": "http"
        },
        "passive": {
            "unhealthy": {
                "http_failures": 0,
                "http_statuses": [429, 500, 503],
                "tcp_failures": 0,
                "timeouts": 0
            },
            "type": "http",
            "healthy": {
                "successes": 0,
                "http_statuses": [200, 201, 202, 203, 204, 205, 206, 207, 208, 226, 300,
301, 302, 303, 304, 305, 306, 307, 308]
            }
        },
        "threshold": 0
    },
    "tags": ["user-level", "low-priority"],
    "host_header": "example.com"
}, {
    ……
}],
    "next":
"http://localhost:8001/upstreams?offset=a3e68c47-9659-4842-8f49-56b04d2791bf"
}
```

3. 检索上游

名称	地址	谓词
检索上游	/upstreams/{upstream name or id}	GET
检索与目标相关的上游	/targets/{target host:port or id}/upstream	GET

请求 URL 参数：

- upstream name or id
- target host:port or id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

其返回内容与添加上游后的返回内容相同。

4. 更新上游

名称	地址	谓词
更新上游	/upstreams/{upstream name or id}	PATCH
更新与目标相关的上游	/targets/{target host:port or id}/upstream	PATCH

请求体字段和 URL 参数：

Body 参数：

请读者参照 2.5.3 章节上方表格字段 URL 参数：

- upstream name or id
- target host:port or id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

其返回内容与添加上游后的返回内容相同。

5. 更新或添加上游

名称	地址	谓词
更新或添加上游	/upstreams/{upstream name or id}	PUT
更新或添加与目标相关联的上游	/targets/{target host:port or id}/upstream	PUT

请求体字段和 URL 参数：

Body 参数：

- 请读者参照 2.5.3 章节上方表格字段

URL 参数：

- upstream name or id
- target host:port or id

响应内容：

HTTP 状态码 201（表示 Created，即已创建）或 200（表示 OK，即正常）。

其返回内容与添加上游后的返回内容相同。

6. 删除上游

名称	地址	谓词
删除上游	/upstreams/{upstream name or id}	DELETE
删除与目标相关的上游	/targets/{target host:port or id}/upstream	DELETE

响应内容：

HTTP 状态码 204（表示 No Content，即无内容）。

7. 查询上游的运行状况

查询指定 Kong 节点的健康状况即显示指定上游所有目标的健康状态。需要注意的是，对 Kong 集群中的不同节点发出相同的请求后，可能会得到不同的结果。例如，假设 Kong 集群中的某个节点遇到了网络问题，并且此问题导致其无法连接到某些目标服务，那么该节

点会将这些目标服务标记为不健康状态（流量会路由到可以成功到达的其他目标服务），但对于其他没有遭遇网络问题的 Kong 节点来说，这些目标服务为健康状态。

接口响应的字段包含目标对象的数组。每个目标的运行状况在其 health 字段中返回以下内容。

- 如果由于 DNS 问题而无法在环形均衡器中激活目标，则其状态将显示为 DNS_ERROR。
- 当上游配置中未启用健康检测时，活动目标的健康状态显示为 HEALTHCHECKS_OFF。
- 当启用健康检测并确定目标为健康状态（自动或手动）时，其状态将显示为 HEALTHY，即健康状态。这意味着此目标在负载均衡器中将会参与负载。
- 当目标被主动或被动健康检测（断路器）或手动禁用时，其状态将显示为 UNHEALTHY，即不健康状态。这意味着此目标在负载均衡器中不会参与负载。

名称	地址	谓词
上游的运行状况	/upstreams/{name or id}/health/	GET

请求 URL 参数：

- name or id
- balancer_health

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

```
{
   "total": 2,
   "node_id": "cbb297c0-14a9-46bc-ad91-1d0ef9b42df9",
   "data": [
      {
         "created_at": 1485524883980,
         "id": "18c0ad90-f942-4098-88db-bbee3e43b27f",
         "health": "HEALTHY",
         "target": "127.0.0.1:20000",
         "upstream_id": "07131005-ba30-4204-a29f-0927d53257b4",
```

```
            "weight": 100
        },
        {
            "created_at": 1485524914883,
            "id": "6c6f34eb-e6c3-4c1f-ac58-4060e5bca890",
            "health": "UNHEALTHY",
            "target": "127.0.0.1:20002",
            "upstream_id": "07131005-ba30-4204-a29f-0927d53257b4",
            "weight": 200
        }
    ]
}
```

如果设置查询字符串参数 `balancer_health=1`，Kong 将返回整个上游的健康状态。响应内容为 HTTP 状态码 200（表示 OK，即正常）。

```
{
    "data": {
        "health": "HEALTHY",
        "id": "07131005-ba30-4204-a29f-0927d53257b4"
    },
    "next": null,
    "node_id": "cbb297c0-14a9-46bc-ad91-1d0ef9b42df9"
}
```

D. 目标

1. 添加目标

名称	地址	谓词
添加与上游相关联	/upstreams/{upstream host:port or id}/targets	POST

的目标

请求体字段和 URL 参数：

请求体字段包括 target、weight、tags。

URL 参数：

- upstream host:port or id

响应内容：

HTTP 状态码 201（表示 Created，即已创建）。

添加目标后的返回内容：

```
{
    "id": "a3395f66-2af6-4c79-bea2-1b6933764f80",
    "created_at": 1422386534,
    "upstream": {"id":"885a0392-ef1b-4de3-aacf-af3f1697ce2c"},
    "target": "example.com:8000",
    "weight": 100,
    "tags": ["user-level", "low-priority"]
}
```

2. 列出目标

名称	地址	谓词
列出与上游相关的目标	/upstreams/{upstream host:port or id}/targets	GET

请求 URL 参数：

- upstream host:port or id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

列出目标后的返回内容：

```
{
"data": [{
```

```
    "id": "de07bda7-8223-459b-9403-ae8d68dadc42",
    "created_at": 1422386534,
    "upstream": {"id":"173a6cee-90d1-40a7-89cf-0329eca780a6"},
    "target": "example.com:8000",
    "weight": 100,
    "tags": ["user-level", "low-priority"]
}, {
    "id": "eacbecba-0f0f-447a-984b-b810bbac2bf6",
    "created_at": 1422386534,
    "upstream": {"id":"f00c6da4-3679-4b44-b9fb-36a19bd3ae83"},
    "target": "example.com:8000",
    "weight": 100,
    "tags": ["admin", "high-priority", "critical"]
}],
    "next":
"http://localhost:8001/targets?offset=a3e68c47-9659-4842-8f49-56b04d2791bf"
}
```


3. 删除目标

名称	地址	谓词
删除目标	/upstreams/{upstream name or id}/targets/{host:port or id}	DELETE

请求 URL 参数：

- upstream name or id
- host:port or id

响应内容：

HTTP 状态码 204（表示 No Content，即无内容）。

4. 将目标地址设置为健康

此调用将重置运行在 Kong 节点的所有工作进程中的健康检测机制中的健康计数器，并在集群范围内广播消息，以便将健康状态传播到整个 Kong 集群。

名称	地址	谓词
将目标地址设置为健康	/upstreams/{upstream name or id}/targets/{target or id}/{address}/healthy	POST

请求 URL 参数：

- upstream name or id
- target or id
- address

响应内容：

HTTP 状态码 204（表示 No Content，即无内容）。

5. 将目标地址设置为不健康

此调用将重置运行在 Kong 节点的所有工作进程中的健康检测机制中的不健康计数器，并在集群范围内广播消息，以便将不健康状态传播到整个 Kong 集群。

名称	地址	谓词
将目标地址设置为不健康	/upstreams/{upstream name or id}/targets/{target or id}/{address}/unhealthy	POST

请求 URL 参数：

- upstream name or id
- target or id
- address

响应内容：

HTTP 状态码 204（表示 No Content，即无内容）。

6. 将目标设置为健康

名称	地址	谓词
将目标设置为健康	/upstreams/{upstream name or id}/targets/{target or id}/healthy	POST

请求 URL 参数：

- upstream name or id
- target or id

响应内容：

HTTP 状态码 204（表示 No Content，即无内容）。

7. 将目标设置为不健康

名称	地址	谓词
将目标设置为不健康	/upstreams/{upstream name or id}/targets/{target or id}/unhealthy	POST

请求 URL 参数：

- upstream name or id
- target or id

响应内容：

HTTP 状态码 204（表示 No Content，即无内容）

8. 列出所有目标

名称	地址	谓词
列出所有目标	/upstreams/{name or id}/targets/all/	GET

请求 URL 参数：

- name or id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

列出所有目标后的返回内容：

```
{
    "total": 2,
    "data": [
        {
            "created_at": 1485524883980,
            "id": "18c0ad90-f942-4098-88db-bbee3e43b27f",
            "target": "127.0.0.1:20000",
            "upstream_id": "07131005-ba30-4204-a29f-0927d53257b4",
            "weight": 100
        },
        {
            "created_at": 1485524914883,
            "id": "6c6f34eb-e6c3-4c1f-ac58-4060e5bca890",
            "target": "127.0.0.1:20002",
            "upstream_id": "07131005-ba30-4204-a29f-0927d53257b4",
            "weight": 200
        }
    ]
}
```


E. 消费者

1. 添加消费者

名称	地址	谓词
添加消费者	/consumers	POST

请求体字段：

- username
- custom_id

- tags

响应内容：

HTTP 状态码 201（表示 Created，即已创建）。

添加消费者后的返回内容：

```
{
    "id": "e09dd4ee-fbc1-427d-93a7-29ffeb90fac3",
    "created_at": 1422386534,
    "username": "my-username",
    "custom_id": "my-custom-id",
    "tags": ["user-level", "low-priority"]
}
```

2. 列出消费者

名称	地址	谓词
列出消费者	/consumers	GET

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

列出消费者后的返回内容：

```
{
"data": [{
    "id": "a4407883-c166-43fd-80ca-3ca035b0cdb7",
    "created_at": 1422386534,
    "username": "my-username",
    "custom_id": "my-custom-id",
    "tags": ["user-level", "low-priority"]
}, {
    "id": "01c23299-839c-49a5-a6d5-8864c09184af",
    "created_at": 1422386534,
```

```
    "username": "my-username",
    "custom_id": "my-custom-id",
    "tags": ["admin", "high-priority", "critical"]
}],
    "next":
"http://localhost:8001/consumers?offset=a3e68c47-9659-4842-8f49-56b04d2791bf"
}
```

3. 检索消费者

名称	地址	谓词
检索消费者	/consumers/{consumer username or id}	GET
检索与插件相关的消费者	/plugins/{plugin id}/consumer	GET

请求 URL 参数：

- consumer username or id
- plugin id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

检索消费者后的返回内容：

```
{
    "id": "e09dd4ee-fbc1-427d-93a7-29ffeb90fac3",
    "created_at": 1422386534,
    "username": "my-username",
    "custom_id": "my-custom-id",
    "tags": ["user-level", "low-priority"]
}
```

4. 更新消费者

名称	地址	谓词
更新消费者	/consumers/{consumer username or id}	PATCH
更新与插件关联的消费者	/plugins/{plugin id}/consumer	PATCH

请求体字段和 URL 参数：

请求体字段：

- username
- custom_id
- tags

URL 参数：

- consumer username or id
- plugin id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

更新消费者后的返回内容：

```
{
    "id": "e09dd4ee-fbc1-427d-93a7-29ffeb90fac3",
    "created_at": 1422386534,
    "username": "my-username",
    "custom_id": "my-custom-id",
    "tags": ["user-level", "low-priority"]
}
```

5. 更新或添加消费者

名称	地址	谓词
更新或添加消费者	/consumers/{consumer username or id}	PUT

更新或添加与插件关联的消费者	/plugins/{plugin id}/consumer	PUT

请求体字段和 URL 参数：

请求体字段：

- ❒ username
- ❒ custom_id
- ❒ tags

URL 参数：

- ➢ consumer username or id
- ➢ plugin id

响应内容：

HTTP 状态码 201（表示 Created，即已创建）或 HTTP 状态码 200（表示 OK，即正常）。

更新或添加消费者后的返回内容：

```
{
    "id": "e09dd4ee-fbc1-427d-93a7-29ffeb90fac3",
    "created_at": 1422386534,
    "username": "my-username",
    "custom_id": "my-custom-id",
    "tags": ["user-level", "low-priority"]
}
```

6. 删除消费者

名称	地址	谓词
删除消费者	/consumers/{consumer username or id}	DELETE

请求 URL 参数：

- ➢ consumer username or id

响应内容：

HTTP 状态码 204（表示 No Content，即无内容）。

F. 插件

1. 添加插件

名称	地址	谓词
添加插件	/plugins	POST
添加与路由相关联的插件	/routes/{route id}/plugins	POST
添加与服务相关联的插件	/services/{service id}/plugins	POST
添加与消费者相关联的插件	/consumers/{consumer id}/plugins	POST

请求体字段和 URL 参数：

请求体字段包括 name、route、service、consumer、config、run_on、protocols、enabled 和 tags。

URL 参数：

- route id
- service id
- consumer id

响应内容：

HTTP 状态码 201（表示 Created，即已创建）。

添加插件后的返回内容：

```
{
    "id": "5b0d3003-a1db-411b-8724-2f34d3117440",
    "name": "rate-limiting",
    "created_at": 1422386534,
    "route": null,
    "service": null,
    "consumer": null,
```

```
    "config": {"hour":500, "minute":20},
    "protocols": ["http", "https"],
    "enabled": true,
    "tags": ["user-level", "low-priority"]
}
```

2. 列出插件

名称	地址	谓词
列出所有插件	/plugins	GET
列出与路由相关联的插件	/routes/{route id}/plugins	GET
列出与服务相关联的插件	/services/{service id}/plugins	GET
列出与消费者相关联的插件	/consumers/{consumer id}/plugins	GET

请求 URL 参数：

- route id
- service id
- consumer id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

列出插件后的返回内容：

```
{
"data": [{
    "id": "02621eee-8309-4bf6-b36b-a82017a5393e",
    "name": "rate-limiting",
    "created_at": 1422386534,
    "route": null,
```

```
    "service": null,
    "consumer": null,
    "config": {"hour":500, "minute":20},
    "protocols": ["http", "https"],
    "enabled": true,
    "tags": ["user-level", "low-priority"]
}, {
    "id": "66c7b5c4-4aaf-4119-af1e-ee3ad75d0af4",
    "name": "rate-limiting",
    "created_at": 1422386534,
    "route": null,
    "service": null,
    "consumer": null,
    "config": {"hour":500, "minute":20},
    "protocols": ["tcp", "tls"],
    "enabled": true,
    "tags": ["admin", "high-priority", "critical"]
}],
    "next":
"http://localhost:8001/plugins?offset=a3e68c47-9659-4842-8f49-56b04d2791bf"
}
```


3. 检索插件

名称	地址	谓词
检索插件	/plugins/{plugin id}	GET
检索与路由相关联的插件	/routes/{route name or id}/plugins/{plugin id}	GET
检索与服务相关联	/services/{service name or id}/plugins/{plugin id}	GET

的插件		
检索与消费者相关联的插件	/consumers/{consumer username or id}/plugins/{plugin id}	GET

请求 URL 参数：

- plugin id
- route name or id
- service name or id
- consumer username or id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

检索插件后的返回内容：

```
{
    "id": "5b0d3003-a1db-411b-8724-2f34d3117440",
    "name": "rate-limiting",
    "created_at": 1422386534,
    "route": null,
    "service": null,
    "consumer": null,
    "config": {"hour":500, "minute":20},
    "protocols": ["http", "https"],
    "enabled": true,
    "tags": ["user-level", "low-priority"]
}
```

4. 更新插件

名称	地址	谓词
更新插件	/plugins/{plugin id}	PATCH
更新与路由相关联	/routes/{route name or id}/plugins/{plugin id}	PATCH

的插件		
更新与服务相关联的插件	/services/{service name or id}/plugins/{plugin id}	PATCH
更新与消费者相关联的插件	/consumers/{consumer username or id}/plugins/{plugin id}	PATCH

请求体字段和 URL 参数：

请求体字段包括 name、route、service、consumer、config、run_on、protocols、enabled 和 tags。

URL 参数：

- plugin id
- route name or id
- service name or id
- consumer username or id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

更新插件后的返回内容：

```
{
    "id": "5b0d3003-a1db-411b-8724-2f34d3117440",
    "name": "rate-limiting",
    "created_at": 1422386534,
    "route": null,
    "service": null,
    "consumer": null,
    "config": {"hour":500, "minute":20},
    "protocols": ["http", "https"],
    "enabled": true,
    "tags": ["user-level", "low-priority"]
}
```

5. 更新或添加插件

名称	地址	谓词
更新或添加插件	/plugins/{plugin id}	PUT
更新或添加与路由相关联的插件	/routes/{route name or id}/plugins/{plugin id}	PUT
更新或添加与服务相关联的插件	/services/{service name or id}/plugins/{plugin id}	PUT
更新或添加与消费者相关的插件	/consumers/{consumer username or id}/plugins/{plugin id}	PUT

请求体字段和 URL 参数：

请求体字段包括 name、route、service、consumer、config、run_on、protocols、enabled 和 tags。

URL 参数：

- plugin id
- route name or id
- service name or id
- consumer username or id

响应内容：

HTTP 状态码 201（表示 Created，即已创建）或 200（表示 OK，即正常）。

更新或添加插件后的返回内容：

```
{
    "id": "5b0d3003-a1db-411b-8724-2f34d3117440",
    "name": "rate-limiting",
    "created_at": 1422386534,
    "route": null,
    "service": null,
    "consumer": null,
    "config": {"hour":500, "minute":20},
```

```
    "protocols": ["http", "https"],
    "enabled": true,
    "tags": ["user-level", "low-priority"]
}
```

6. 删除插件

名称	地址	谓词
删除插件	/plugins/{plugin id}	DELETE
删除与路由相关联的插件	/routes/{route name or id}/plugins/{plugin id}	DELETE
删除与服务相关联的插件	/services/{service name or id}/plugins/{plugin id}	DELETE
删除与消费者相关联的插件	/consumers/{consumer username or id}/plugins/{plugin id}	DELETE

请求 URL 参数：

- plugin id
- route name or id
- service name or id
- consumer username or id

响应内容：

HTTP 状态码 204（表示 No Content，即无内容）。

7. 检索已启用的插件

名称	地址	谓词
检索已启用的插件	/plugins/enabled	GET

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

检索已启用插件后的返回内容：

```
{
    "enabled_plugins": [
        "jwt",
        "acl",
        "cors",
        "oauth2",
        "tcp-log",
        "udp-log",
        "file-log",
        "http-log",
        "key-auth",
        "hmac-auth",
        "basic-auth",
        "ip-restriction",
        "request-transformer",
        "response-transformer",
        "request-size-limiting",
        "rate-limiting",
        "response-ratelimiting",
        "aws-lambda",
        "bot-detection",
        "correlation-id",
        "datadog",
        "galileo",
        "ldap-auth",
        "loggly",
        "statsd",
        "syslog"
    ]
```

```
}
```

8. 检索插件架构

名称	地址	谓词
检索插件结构	/plugins/schema/{plugin name}	GET

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

检索插件结构后的返回内容：

```
{
    "fields": {
        "hide_credentials": {
            "default": false,
            "type": "boolean"
        },
        "key_names": {
            "default": "function",
            "required": true,
            "type": "array"
        }
    }
}
```

G. 证书

1. 添加证书

名称	地址	谓词
添加证书	/certificates	POST

请求体字段包括 cert、key、tags 和 snis。

响应内容：

HTTP 状态码 201（表示 Created，即已创建）。

添加证书后的返回内容：

```
{
    "id": "7fca84d6-7d37-4a74-a7b0-93e576089a41",
    "created_at": 1422386534,
    "cert": "-----BEGIN CERTIFICATE-----...",
    "key": "-----BEGIN RSA PRIVATE KEY-----...",
    "tags": ["user-level", "low-priority"]
}
```

2. 列出证书

名称	地址	谓词
列出所有证书	/certificates	GET

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

列出证书后的返回内容：

```
{
"data": [{
    "id": "5683d4c3-4a99-487d-ab63-2cf1d1d6fd51",
    "created_at": 1422386534,
    "cert": "-----BEGIN CERTIFICATE-----...",
    "key": "-----BEGIN RSA PRIVATE KEY-----...",
    "tags": ["user-level", "low-priority"]
}, {
    "id": "b18132a7-81c7-4e07-bb68-4c2ac78aac0d",
    "created_at": 1422386534,
    "cert": "-----BEGIN CERTIFICATE-----...",
```

```
    "key": "-----BEGIN RSA PRIVATE KEY-----...",
    "tags": ["admin", "high-priority", "critical"]
}],
    "next":
"http://localhost:8001/certificates?offset=a3e68c47-9659-4842-8f49-56b04d2791bf"
}
```

3. 检索证书

名称	地址	谓词
检索证书	/certificates/{certificate id}	GET

请求 URL 参数：

- certificate id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

检索证书后的返回内容：

```
{
    "id": "7fca84d6-7d37-4a74-a7b0-93e576089a41",
    "created_at": 1422386534,
    "cert": "-----BEGIN CERTIFICATE-----...",
    "key": "-----BEGIN RSA PRIVATE KEY-----...",
    "tags": ["user-level", "low-priority"]
}
```

4. 更新证书

名称	地址	谓词
更新证书	/certificates/{certificate id}	PATCH

请求体字段和 URL 参数：

请求体字段包括 cert、key、tags 和 snis。

URL 参数：

- certificate id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

更新证书后的返回内容：

```
{
    "id": "7fca84d6-7d37-4a74-a7b0-93e576089a41",
    "created_at": 1422386534,
    "cert": "-----BEGIN CERTIFICATE-----...",
    "key": "-----BEGIN RSA PRIVATE KEY-----...",
    "tags": ["user-level", "low-priority"]
}
```

5. 更新或添加证书

名称	地址	谓词
更新或添加证书	/certificates/{certificate id}	PUT

请求体字段和 URL 参数：

请求体字段包括 cert、key、tags 和 snis。

URL 参数：

- certificate id

响应内容：

HTTP 状态码 201（表示 Created，即已创建）或 200（表示 OK，即正常）。

更新或添加证书后的返回内容：

```
{
   "id": "7fca84d6-7d37-4a74-a7b0-93e576089a41",
   "created_at": 1422386534,
   "cert": "-----BEGIN CERTIFICATE-----...",
   "key": "-----BEGIN RSA PRIVATE KEY-----...",
```

```
    "tags": ["user-level", "low-priority"]
}
```

6. 删除证书

名称	地址	谓词
删除证书	/certificates/{certificate id}	DELETE

请求 URL 参数：

- certificate id

响应内容：

HTTP 状态码 204（表示 No Content，即无内容）。

H. SNI

1. 添加 SNI

名称	地址	谓词
添加 SNI	/snis	POST
添加与证书相关联的 SNI	/certificates/{certificate name or id}/snis	POST

请求体字段和 URL 参数：

请求体字段：

- name
- tags
- certificate

URL 参数：

- certificate name or id

响应内容：

HTTP 状态码 201（表示 Created，即已创建）。

添加 SNI 后的返回内容：

```
{
    "id": "12fbbaa8-4da0-4edd-9b1b-c3cdaa9d1caf",
    "name": "my-sni",
    "created_at": 1422386534,
    "tags": ["user-level", "low-priority"],
    "certificate": {"id":"a2e013e8-7623-4494-a347-6d29108ff68b"}
}
```

2. 列出 SNI

名称	地址	谓词
列出所有 SNI	/snis	GET
列出与证书相关联的 SNI	/certificates/{certificate name or id}/snis	GET

请求 URL 参数：

- certificate name or id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

列出 SNI 后的返回内容：

```
{
"data": [{
    "id": "147f5ef0-1ed6-4711-b77f-489262f8bff7",
    "name": "my-sni",
    "created_at": 1422386534,
    "tags": ["user-level", "low-priority"],
    "certificate": {"id":"a3ad71a8-6685-4b03-a101-980a953544f6"}
}, {
    "id": "b87eb55d-69a1-41d2-8653-8d706eecefc0",
    "name": "my-sni",
```

```
    "created_at": 1422386534,
    "tags": ["admin", "high-priority", "critical"],
    "certificate": {"id":"4e8d95d4-40f2-4818-adcb-30e00c349618"}
}],
    "next": "http://localhost:8001/snis?offset=a3e68c47-9659-4842-8f49-56b04d2791bf"
}
```


注意：可以根据 next 继续取数据，直到取出的返回集合为 0。

3. 检索 SNI

名称	地址	谓词
检索 SNI	/snis/{sni name or id}	GET
检索与证书关联的 SNI	/certificates/{certificate id}/snis/{sni name or id}	GET

请求 URL 参数：

- sni name or id
- certificate id

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

检索 SNI 后的返回内容：


```
{
    "id": "12fbbaa8-4da0-4edd-9b1b-c3cdaa9d1caf",
    "name": "my-sni",
    "created_at": 1422386534,
    "tags": ["user-level", "low-priority"],
    "certificate": {"id":"a2e013e8-7623-4494-a347-6d29108ff68b"}
}
```

4. 更新 SNI

名称	地址	谓词
更新 SNI	/snis/{sni name or id}	PATCH
更新与证书相关联的 SNI	/certificates/{certificate id}/snis/{sni name or id}	PATCH

请求体字段和 URL 参数：

请求体字段：

- name
- tags
- certificate

URL 参数：

- sni name or id
- certificate id

响应内容：

HTTP 状态码 200（表示 OK，即正常）

更新 SNI 后的返回内容：

```
{
    "id": "12fbbaa8-4da0-4edd-9b1b-c3cdaa9d1caf",
    "name": "my-sni",
    "created_at": 1422386534,
    "tags": ["user-level", "low-priority"],
    "certificate": {"id":"a2e013e8-7623-4494-a347-6d29108ff68b"}
}
```

5. 更新或添加 SNI

名称	地址	谓词
更新或添加 SNI	/snis/{sni name or id}	PUT

更新或添加与证书相关联的 SNI	/certificates/{certificate id}/snis/{sni name or id}	PUT

请求体字段和 URL 参数：

请求体参数：

- ❒ name
- ❒ tags
- ❒ certificate

URL 参数：

- ➢ sni name or id
- ➢ certificate id

响应内容：

HTTP 状态码 201（表示 Created，即已创建）或 200（表示 OK，即正常）。

更新或添加 SNI 后的返回内容：

```
{
    "id": "12fbbaa8-4da0-4edd-9b1b-c3cdaa9d1caf",
    "name": "my-sni",
    "created_at": 1422386534,
    "tags": ["user-level", "low-priority"],
    "certificate": {"id":"a2e013e8-7623-4494-a347-6d29108ff68b"}
}
```

6. 删除 SNI

名称	地址	谓词
删除 SNI	/snis/{sni name or id}	DELETE
删除与证书相关联的 SNI	/certificates/{certificate id}/snis/{sni name or id}	DELETE

请求 URL 参数：

- ➢ sni name or id

- certificate id

响应内容：

HTTP 状态码 204（表示 No Content，即无内容）。

I. 节点信息

查看检索节点的一些常规信息。

1. 字段信息

属性名称	说明
hostname	Kong 节点的机器名
node_id	正在运行的 Kong 节点的 UUID。因为 Kong 节点启动时会随机生成此 UUID，所以节点每次重新启动时的 node_id 都会有所不同
plugins.available_on_server	Kong 节点上安装的插件的名称
plugins.enabled_in_cluster	Kong 节点启用/配置的插件名称
configuration	关于 Kong 节点的更多配置信息，如 Nginx 参数，数据库，插件等
tagline	Kong 节点的标语
version	Kong 节点当前所使用的版本
prng_seeds	当前 kong 节点工作进程的信息
timers	当前 kong 节点的定时器信息，包括处于挂起/运行态的定时器个数

2. 检索节点信息

名称	地址	谓词
检索节点信息	/	GET

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

检索节点信息后的返回内容：

```
{
    "hostname": "",
    "node_id": "b88e7075-c511-4139-aafe-685513c3b8ea",
    "lua_version": "LuaJIT 2.1.0-beta3",
    "plugins": {
        "available_on_server": [...],
        "enabled_in_cluster": [...]
    },
    "configuration" : {...},
    "tagline": "Welcome to Kong",
    "version": "0.14.0"
}
```

J. 节点状态信息

检索有关节点的使用情况信息以及关于底层 Nginx 进程正在处理的连接的一些基本信息、数据库连接的状态和节点的内存使用情况。

1. 字段信息

属性名称	说明
memory.workers_lua_vms	包含 Kong 节点所有工作进程的内存信息
memory.http_allocated_gc	每一个有工作进程中 HTTP 子模块的 Lua 虚拟机的内存使用信息，此信息为最近 10 秒内接收到的代理调用。由 collectgarbage("count") 报告给每一个工作进程
memory.pid	工作进程的标识号
memory.lua_shared_dicts	与 Kong 节点中所有工作进程有关的共享内存词典的信息，包括 capacity（其中每个节点有多少内存专用于特定的共享内

	存词典）和 allocated_slabs（有多少内存正在使用）。由于这些共享内存词典具有最近最少使用（LRU）功能，因此对于完整的词典来说，allocated_slabs 和 capacity 的值相同。但是对于某些词典，例如缓存 HIT/MISS 共享词典，则需要增加其大小，这将提升 Kong 节点的整体性能
server.total_requests	客户端请求的总数
server.connections_active	当前活动的客户端连接数，包括正在等待的连接
server.connections_accepted	服务接受的客户端连接总数
server.connections_handled	服务已处理的连接总数。在通常情况下，除非已经达到了某些资源的限制，否则该参数值与 server.connections_accepted 的值相同
server.connections_reading	当前正在读取请求头的连接数
server.connections_writing	当前正在将响应写回到客户端的连接数
server.connections_waiting	当前处于等待状态的空闲客户端连接数
database.reachable	反映数据库连接状态的布尔值，不代表数据库的运行状况

2. 检索节点状态信息

名称	地址	谓词
检索节点状态信息	/status	GET

响应内容：

HTTP 状态码 200（表示 OK，即正常）。

```
{
   "database": {
     "reachable": true
   },
   "memory": {
      "workers_lua_vms": [{
         "http_allocated_gc": "0.02 MiB",
```

```
            "pid": 18477
          }, {
            "http_allocated_gc": "0.02 MiB",
            "pid": 18478
        }],
        "lua_shared_dicts": {
            "kong": {
                "allocated_slabs": "0.04 MiB",
                "capacity": "5.00 MiB"
            },
            "kong_db_cache": {
                "allocated_slabs": "0.80 MiB",
                "capacity": "128.00 MiB"
            },
        }
    },
    "server": {
        "total_requests": 3,
        "connections_active": 1,
        "connections_accepted": 1,
        "connections_handled": 1,
        "connections_reading": 0,
        "connections_writing": 1,
        "connections_waiting": 0
    }
}
```