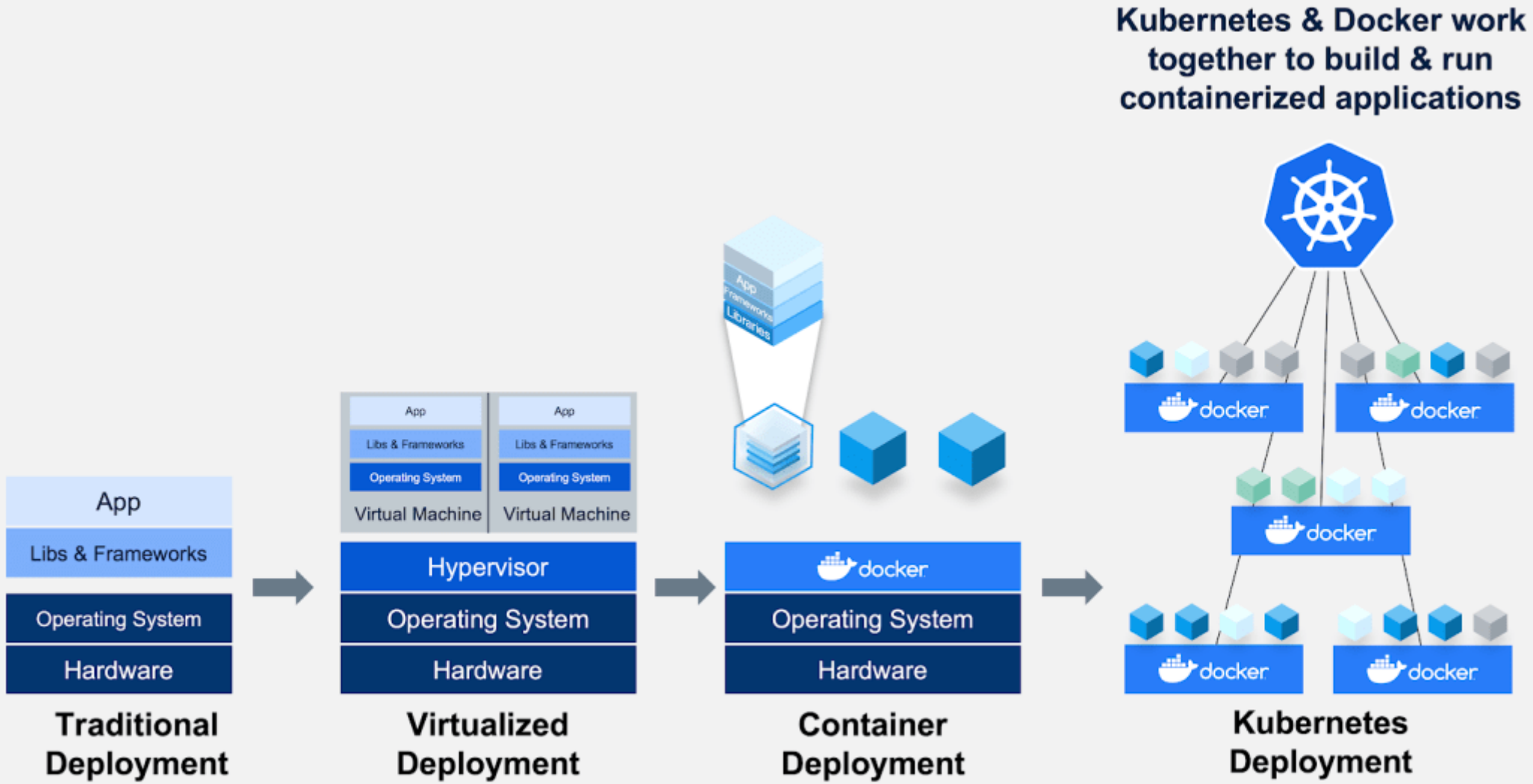
NGINX

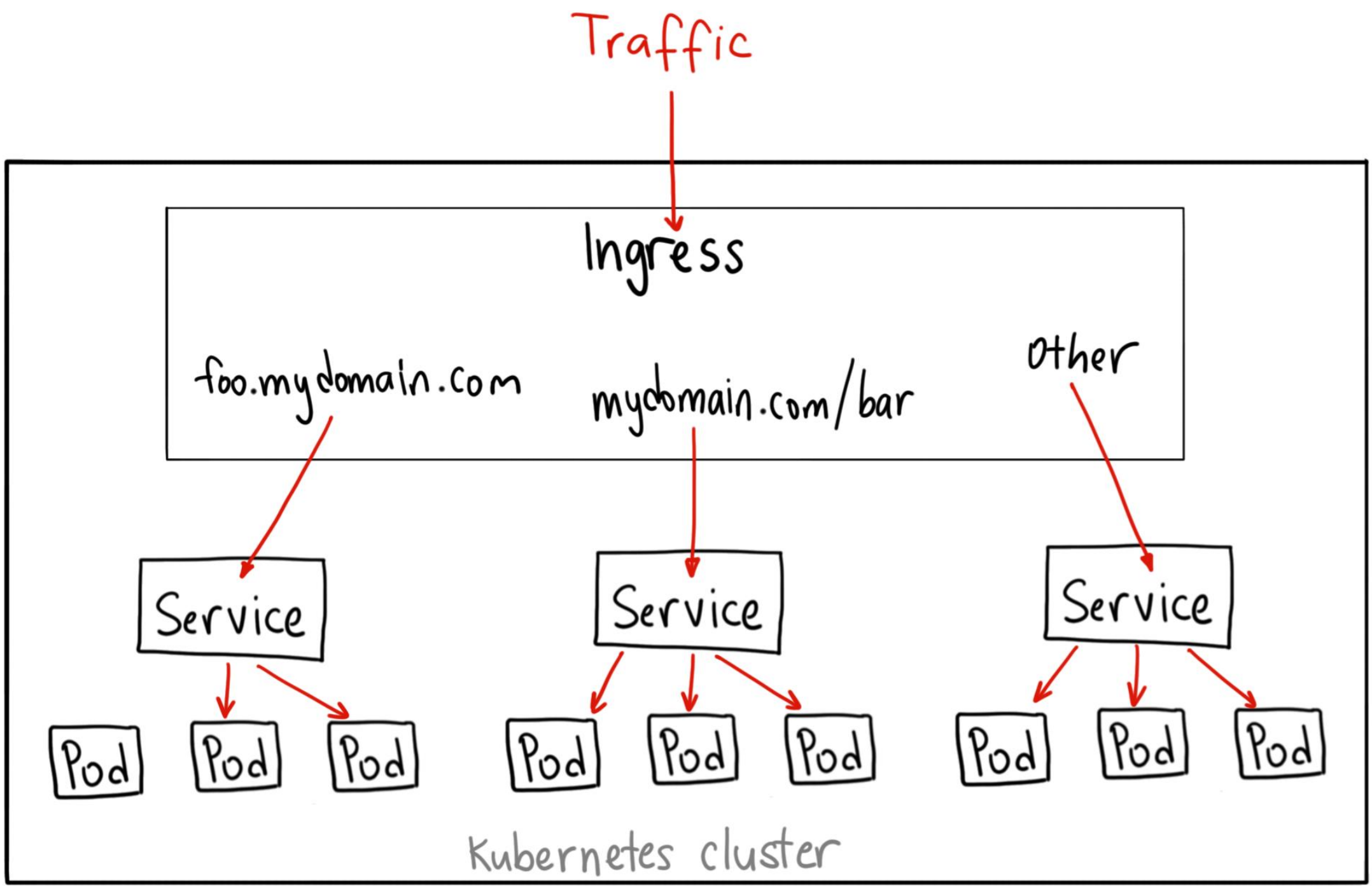
第十季: K8S Ingress Controller 技术细节探讨

K8S Ingress Controller技术细节探讨

- Ingress Controller的工作原理
- Ingress Controller与Master的通讯机制
- K8S官方社区Controller的核心特性
- Nginx官方Controller开源版的核心特性

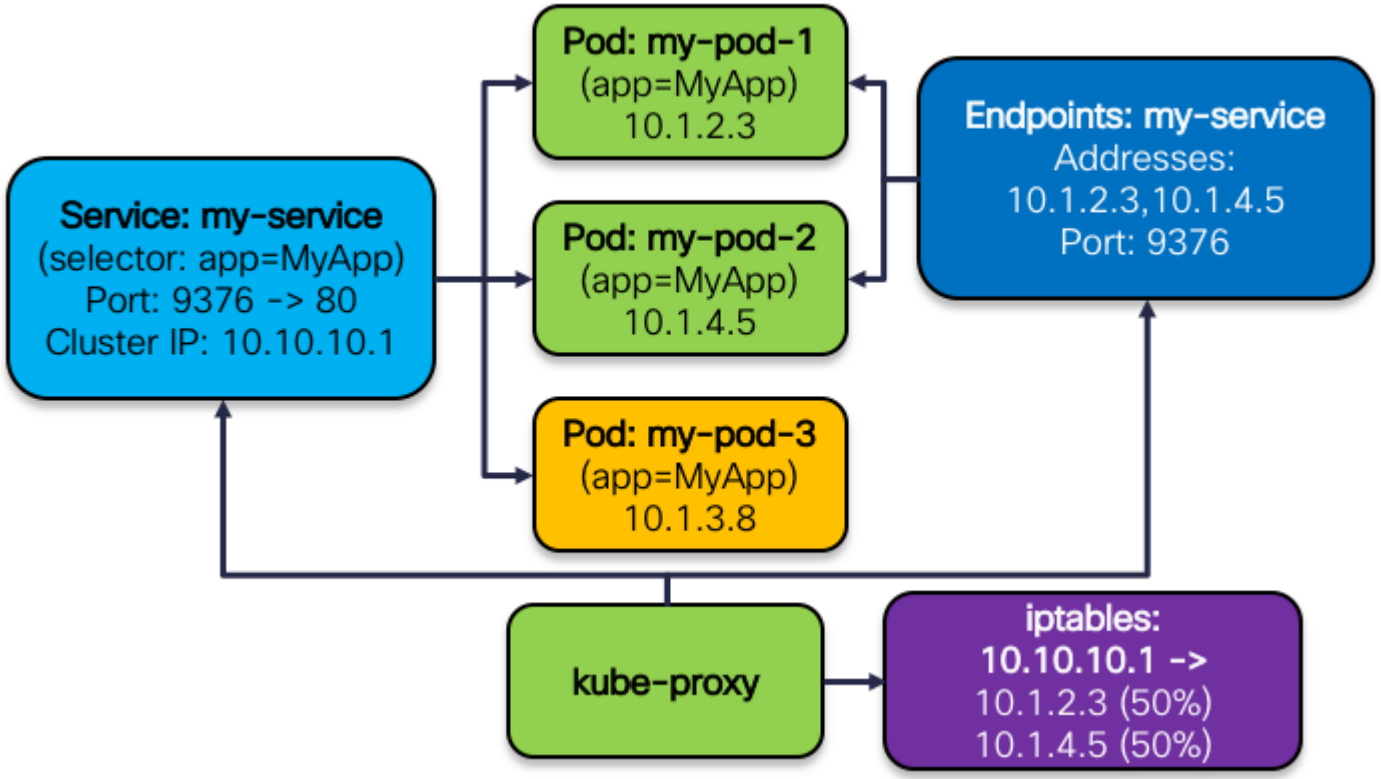
Kubernetes托管容器服务




Traffic
Ingress
foo.mydomain.com
mydomain.com/bar
other
Service
Service
Service
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Kubernetes cluster

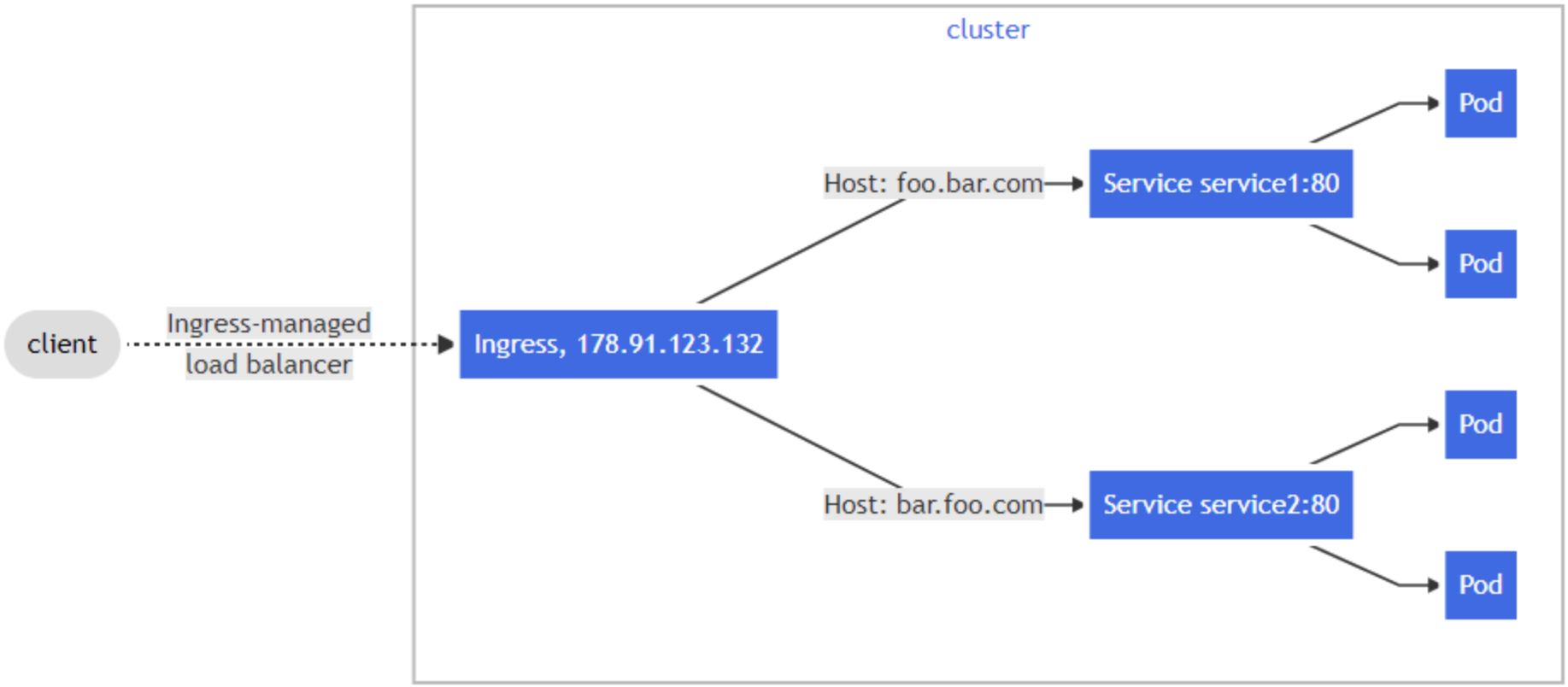
```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
```

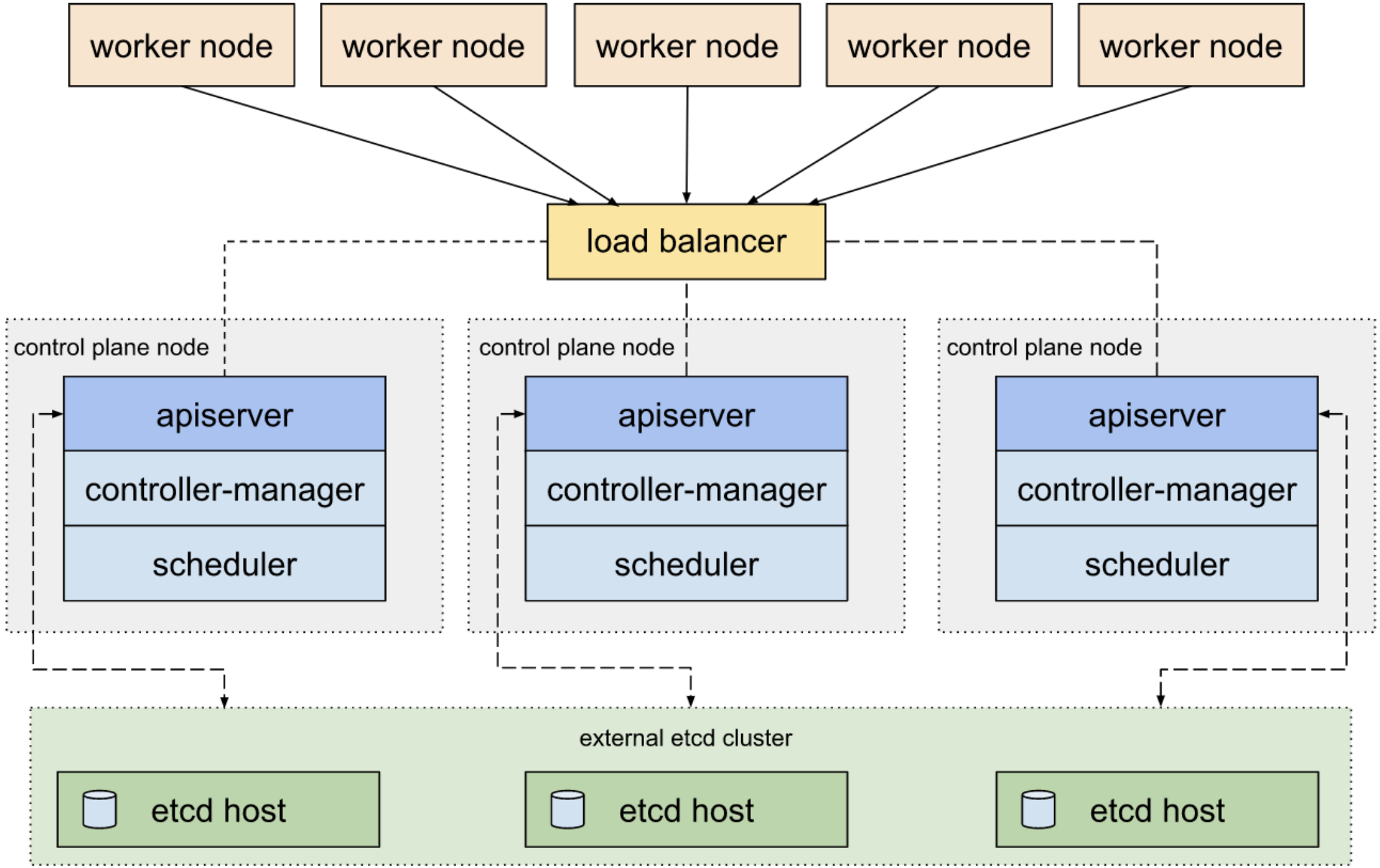
service与endpoint



Ingress

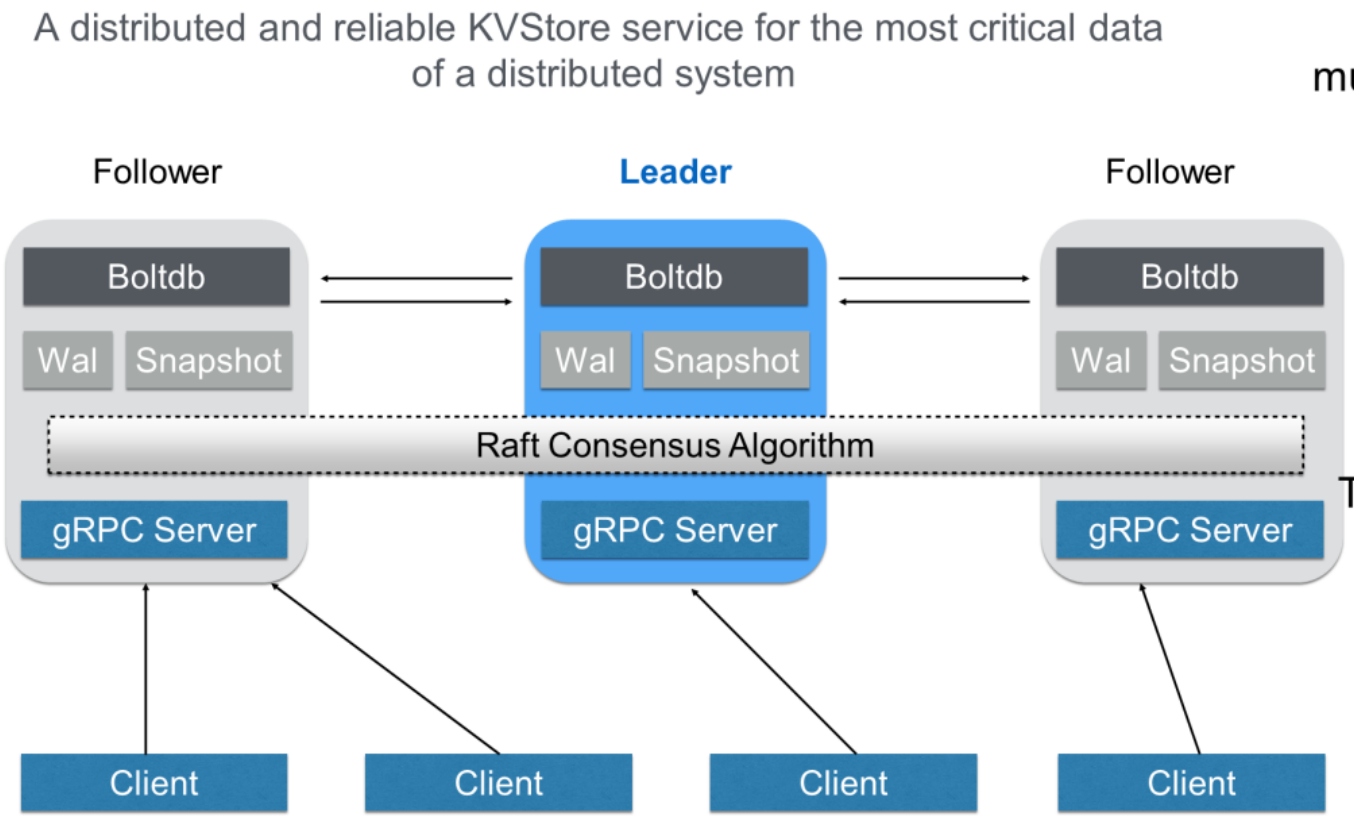
```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: name-virtual-host-ingress
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: service1
            port:
              number: 80
  - host: bar.foo.com
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: service2
            port:
              number: 80
```




worker node
worker node
worker node
worker node
worker node
load balancer
control plane node
apiserver
controller-manager
scheduler
control plane node
apiserver
controller-manager
scheduler
control plane node
apiserver
controller-manager
scheduler
external etcd cluster
etcd host
etcd host
etcd host

ETCD数据库

- Ingress
- Endpoint
- Service
- Secret
- ConfigMap



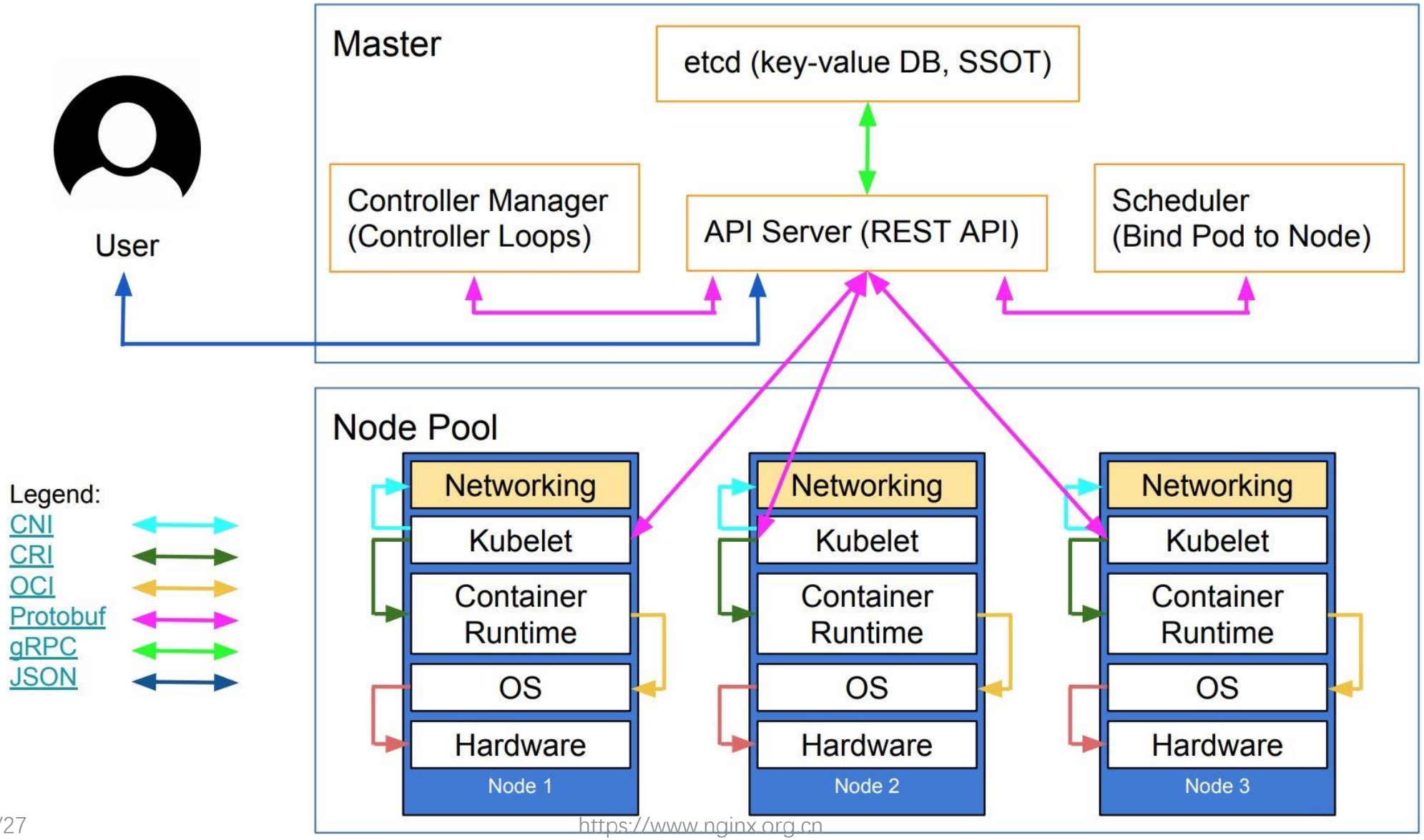
Two quorums must have a common node.

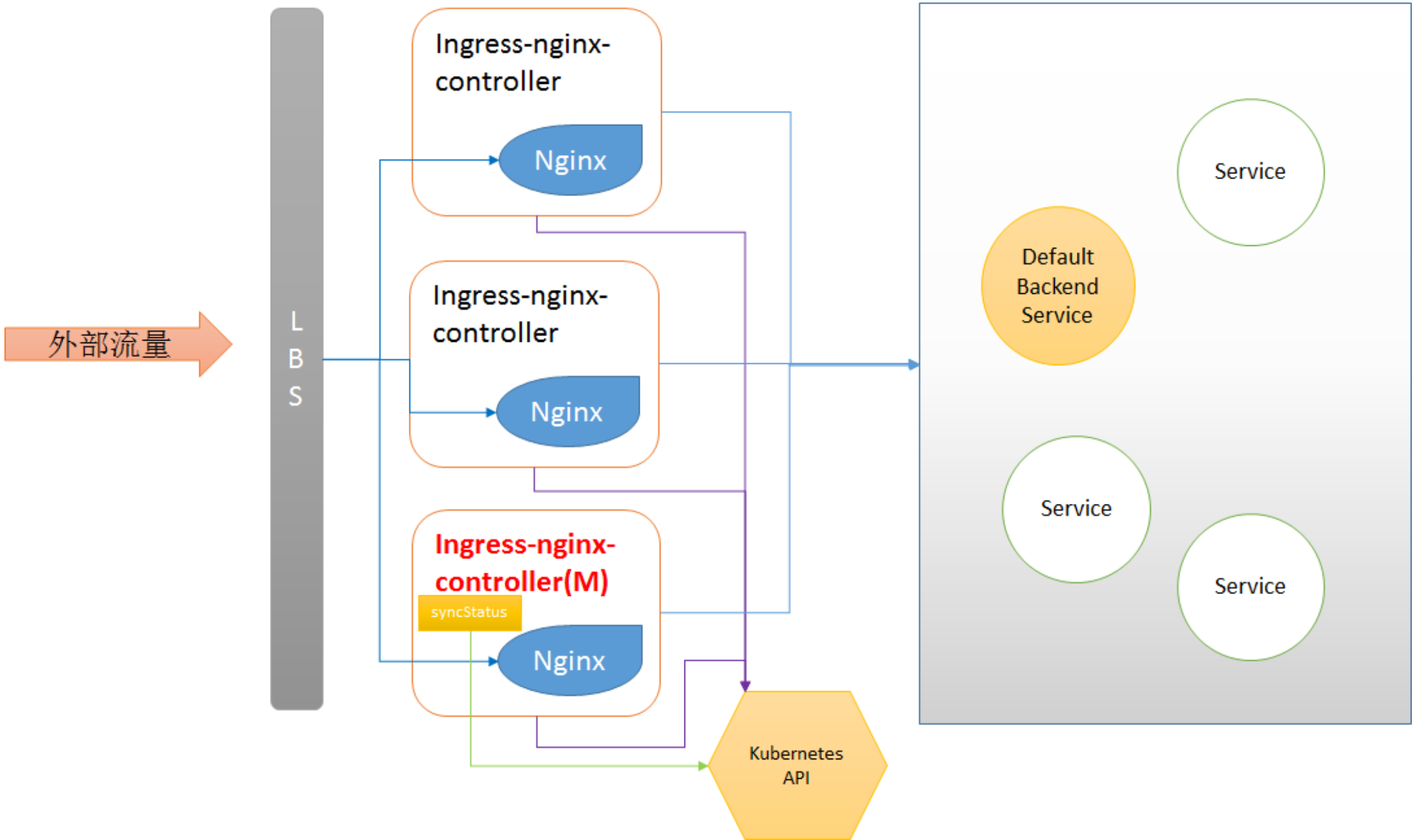
quorum = (n + 1)/2

One faulty node is tolerable in a three-node cluster

Two faulty nodes are tolerable in a five-node cluster

Kubernetes' high-level component architecture



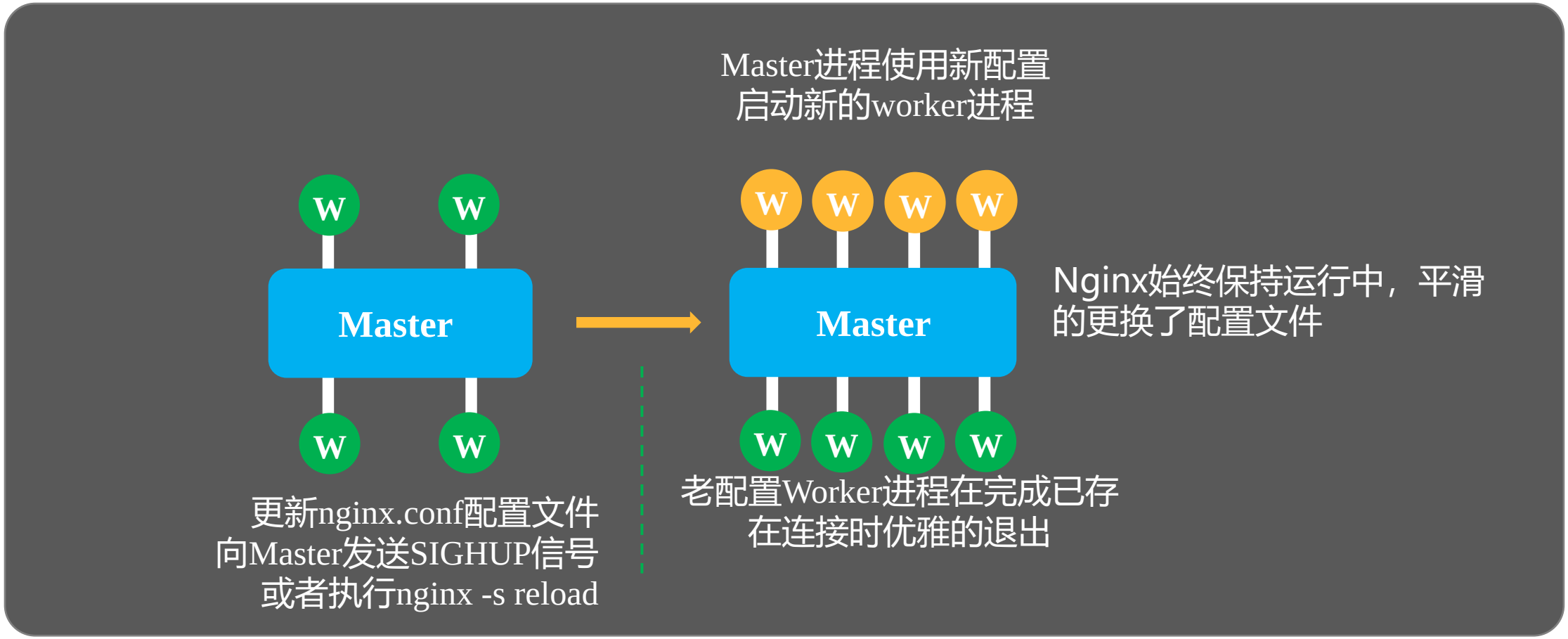

外部流量
L
B
S
Ingress-nginx-controller
Nginx
Ingress-nginx-controller
Nginx
Ingress-nginx-controller(M)
syncStatus
Nginx
Kubernetes API
Service
Default Backend Service
Service
Service

nginx.conf模板

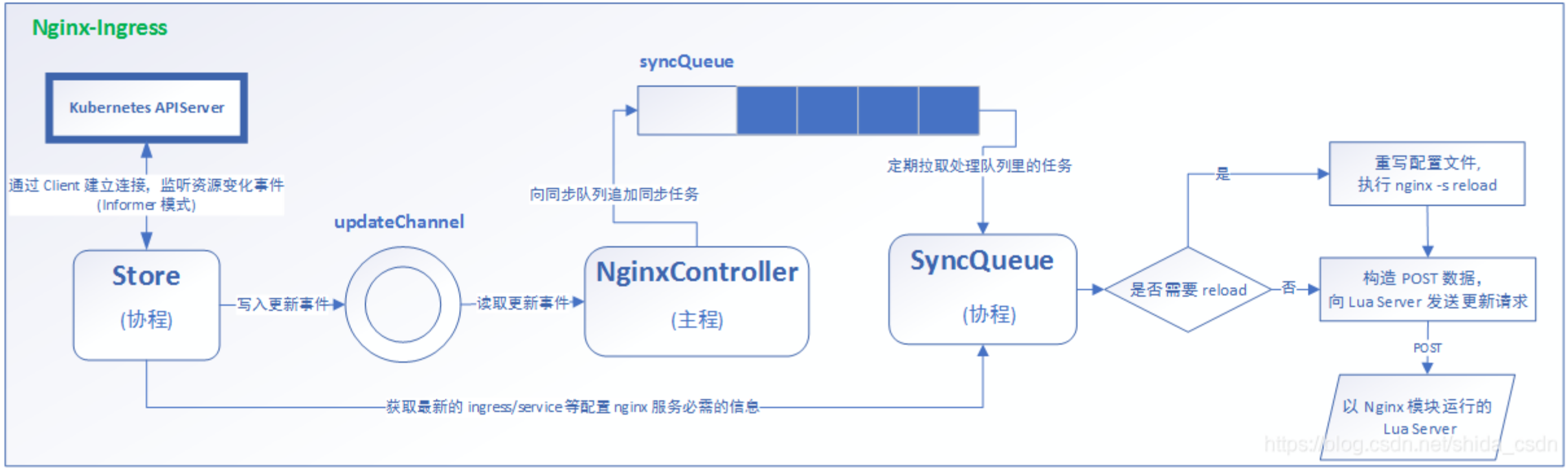
- K8S官方
  - https://github.com/kubernetes/ingress-nginx/blob/master/rootfs/etc/nginx/template/nginx.tmpl
- Nginx官方
  - https://github.com/nginxinc/kubernetes-ingress/blob/master/internal/configs/version1/nginx.tmpl
  - https://github.com/nginxinc/kubernetes-ingress/blob/master/internal/configs/version1/nginx.ingress.tmpl

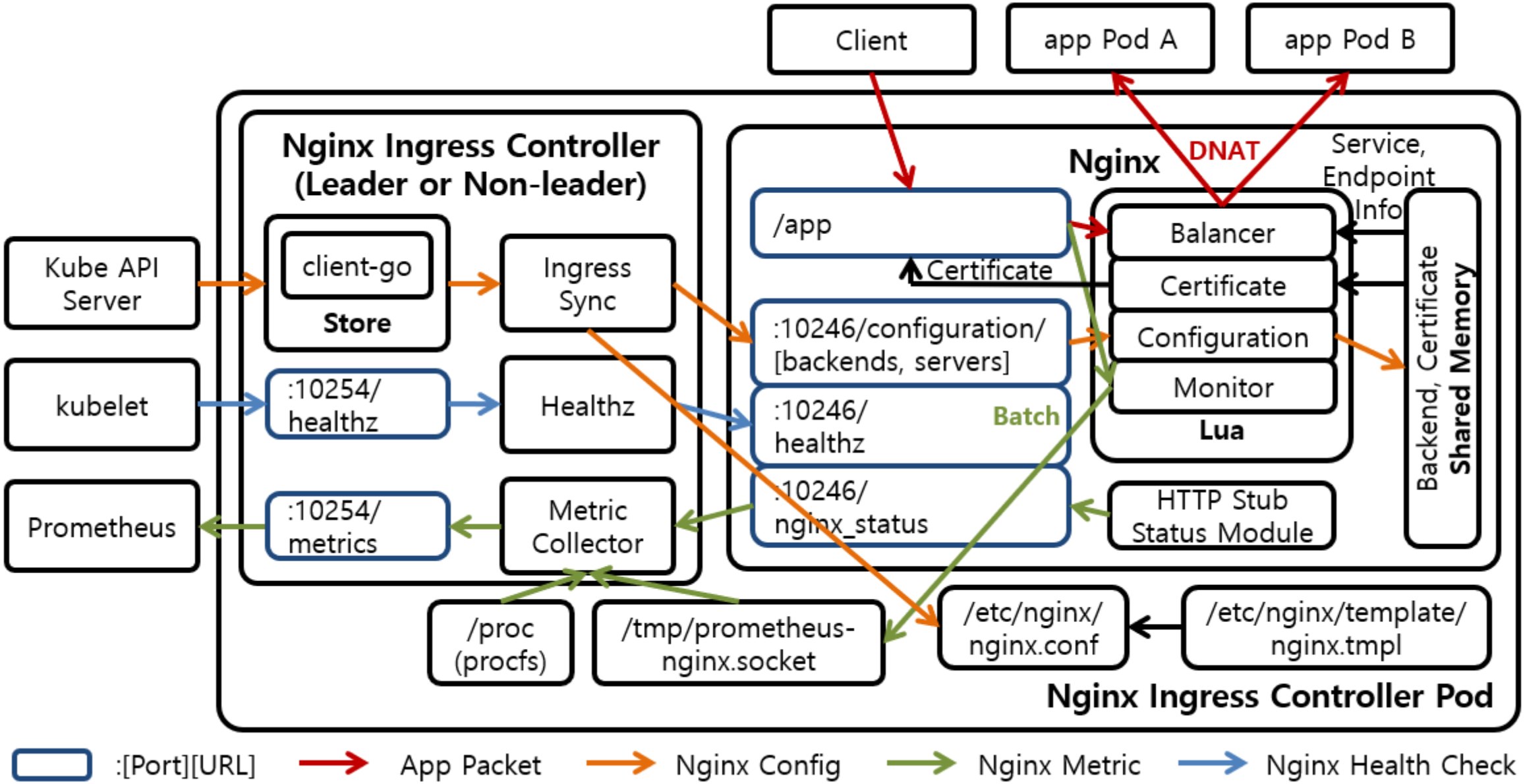
reload流程

不停机载入新配置

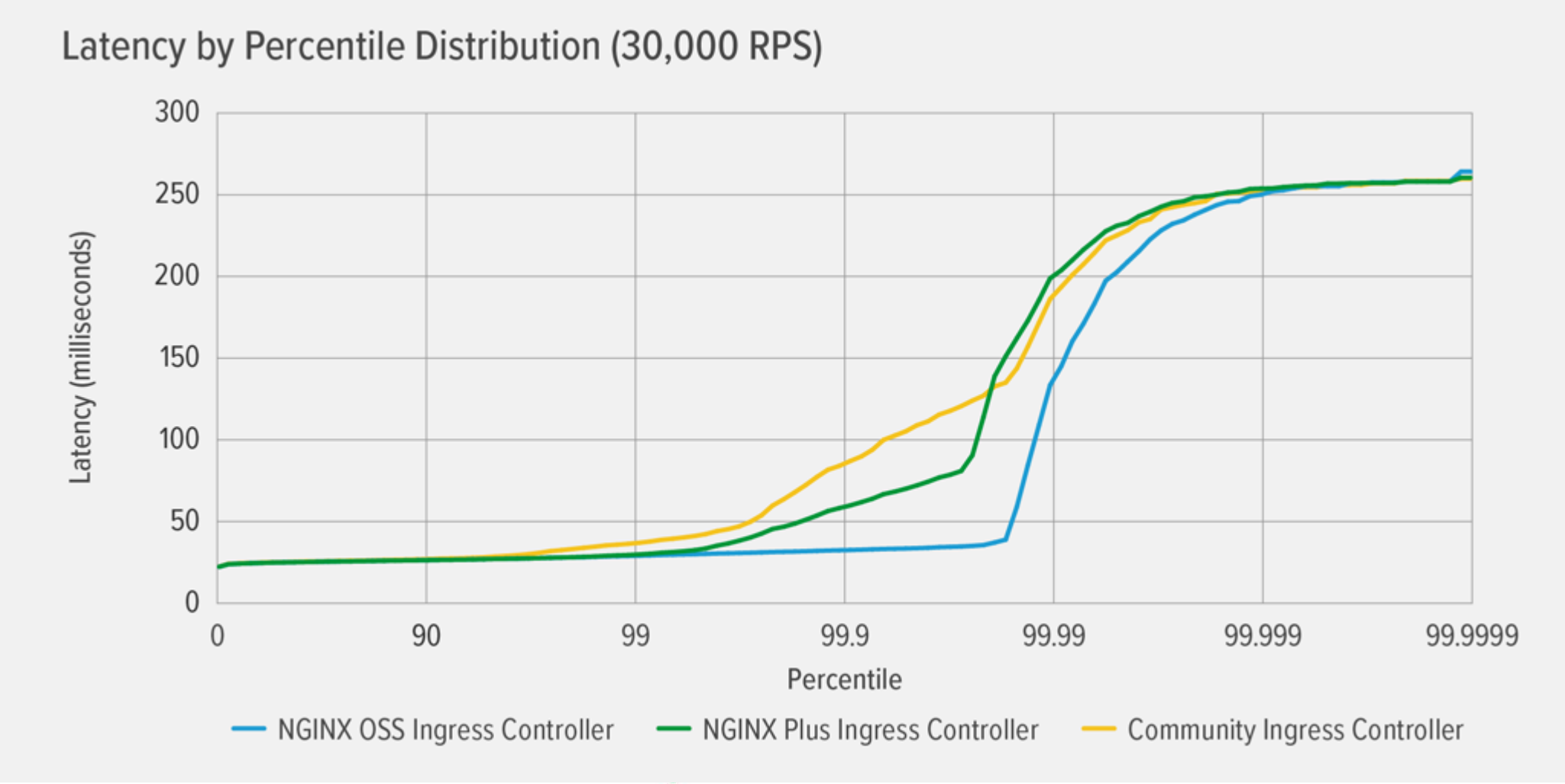


K8S官方Ingress Controller架构

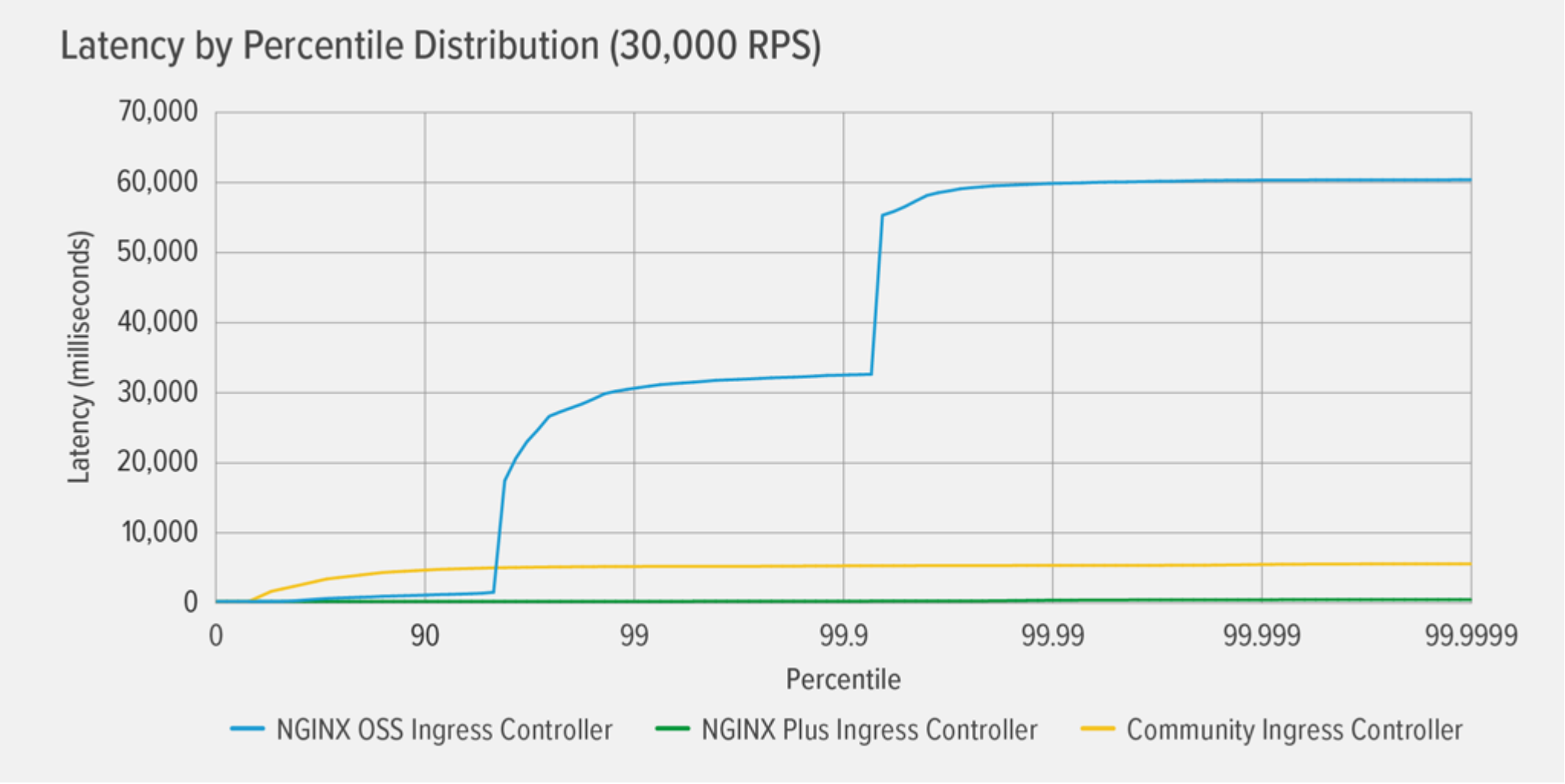



Client
app Pod A
app Pod B
Nginx Ingress Controller
(Leader or Non-leader)
Nginx
DNAT
Service,
Endpoint
Info
Kube API
Server
client-go
Store
Ingress
Sync
/app
Certificate
Balancer
Certificate
Configuration
Monitor
Lua
Backend, Certificate
Shared Memory
kubelet
:10254/
healthz
Healthz
:10246/configuration/
[backends, servers]
:10246/
healthz
Batch
Prometheus
:10254/
metrics
Metric
Collector
:10246/
nginx_status
HTTP Stub
Status Module
/proc
(procfs)
/tmp/prometheus-
nginx.socket
/etc/nginx/
nginx.conf
/etc/nginx/template/
nginx.tmpl
Nginx Ingress Controller Pod
:[Port][URL]
App Packet
Nginx Config
Nginx Metric
Nginx Health Check

静态部署测试：时延对比


Latency by Percentile Distribution (30,000 RPS)
Latency (milliseconds)
300
250
200
150
100
50
0
0
90
99
99.9
99.99
99.999
99.9999
Percentile
NGINX OSS Ingress Controller
NGINX Plus Ingress Controller
Community Ingress Controller

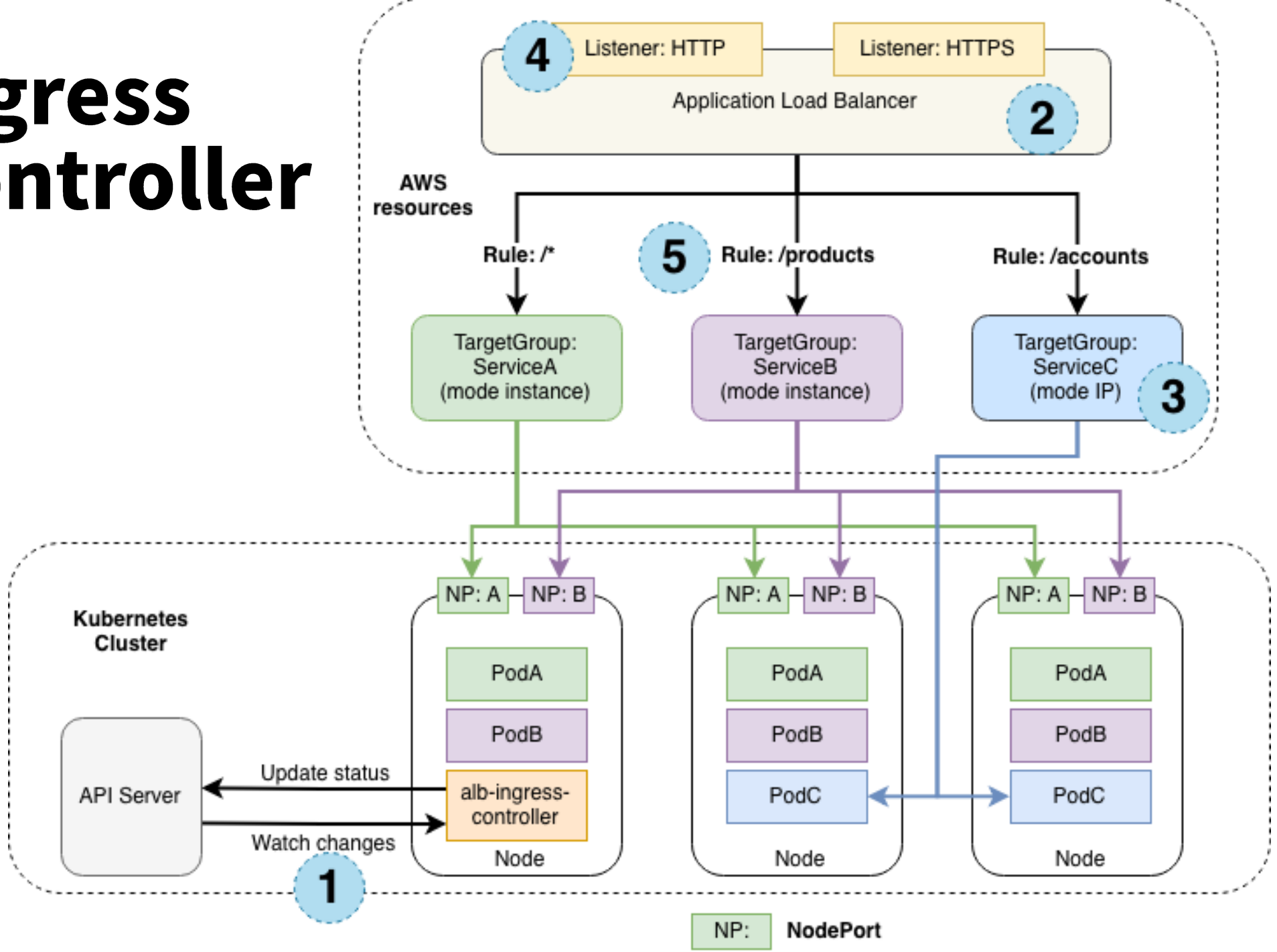
动态部署测试：时延对比


Latency by Percentile Distribution (30,000 RPS)
Latency (milliseconds)
70,000
60,000
50,000
40,000
30,000
20,000
10,000
0
0
90
99
99.9
99.99
99.999
99.9999
Percentile
NGINX OSS Ingress Controller
NGINX Plus Ingress Controller
Community Ingress Controller

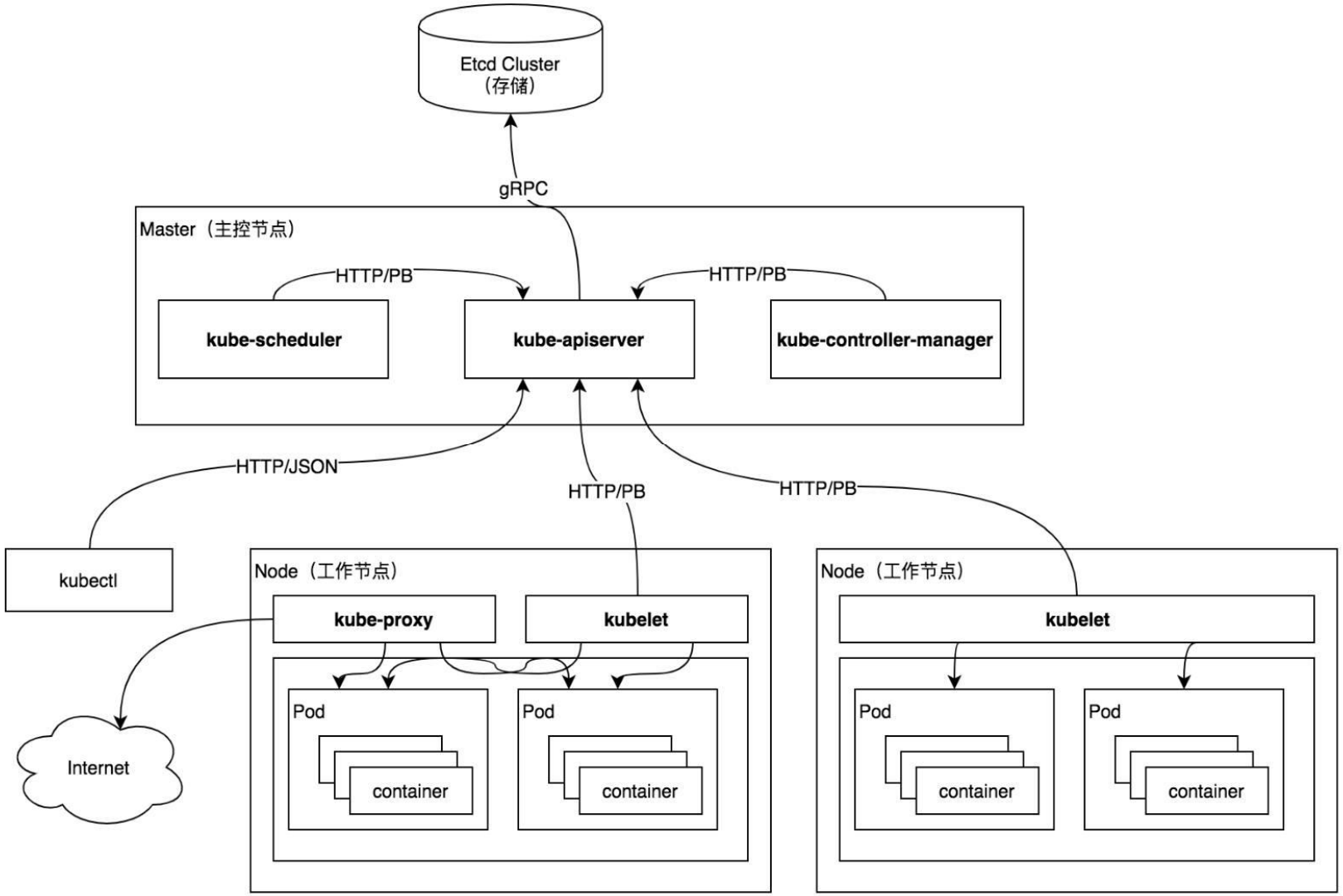
K8S Ingress Controller技术细节探讨

- Ingress Controller的工作原理
- Ingress Controller与Master的通讯机制
- K8S官方社区Controller的核心特性
- Nginx官方Controller开源版的核心特性

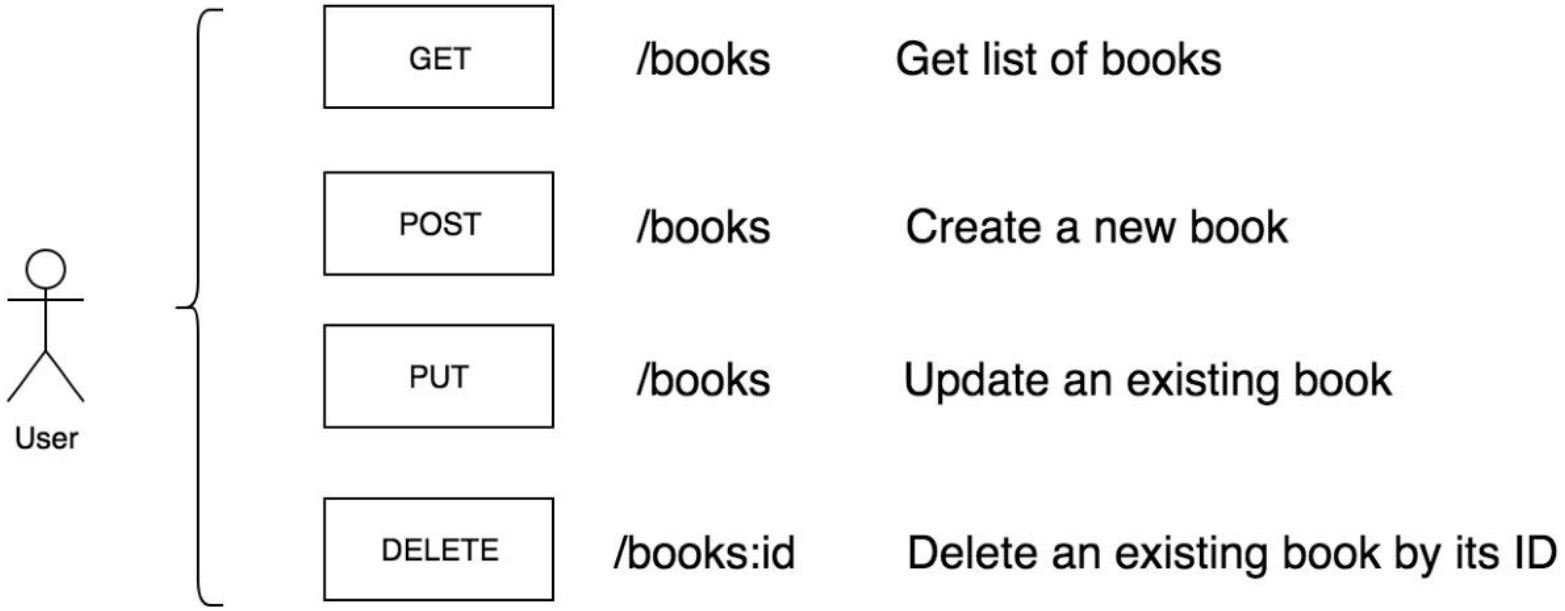
Ingress Controller



Kubernetes通讯协议

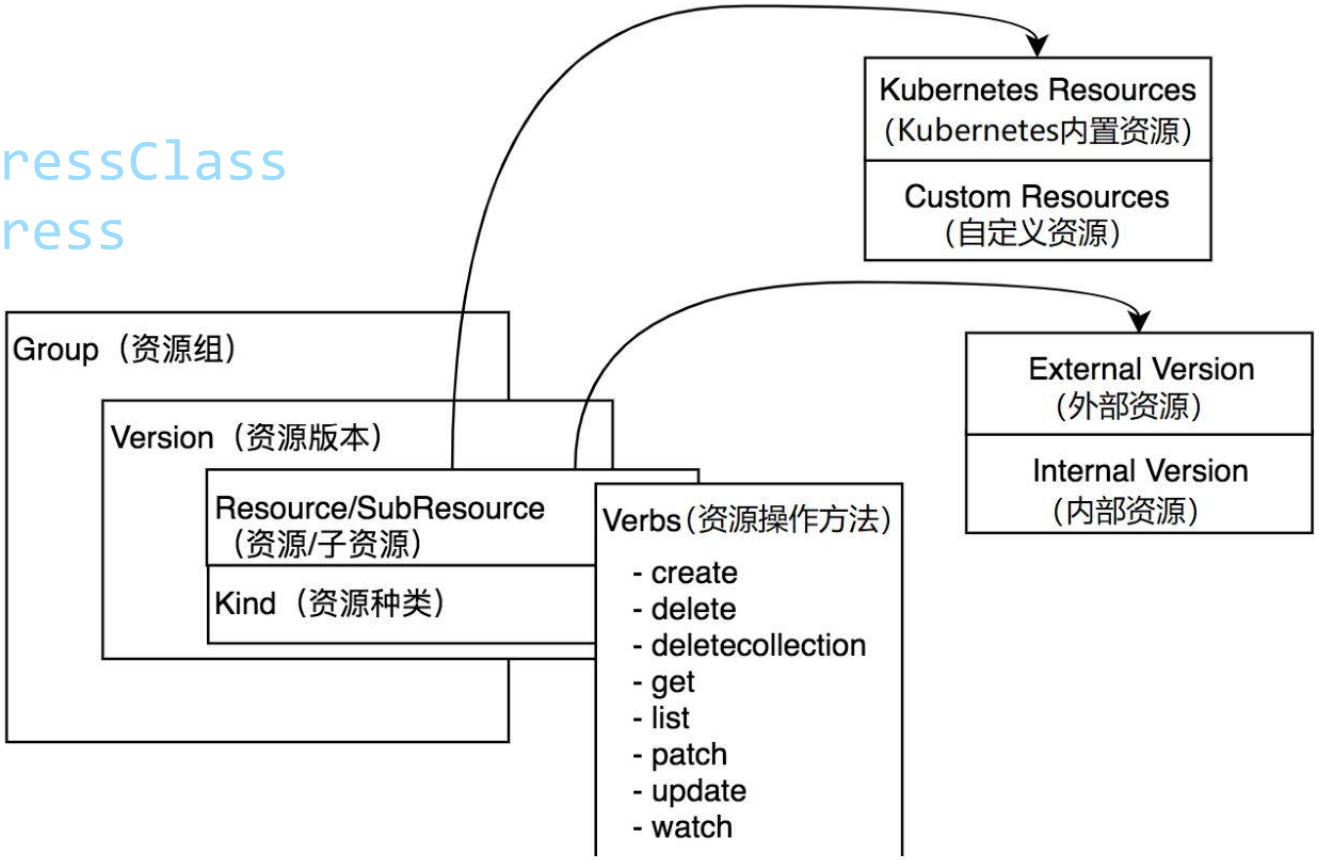


REST API：Method

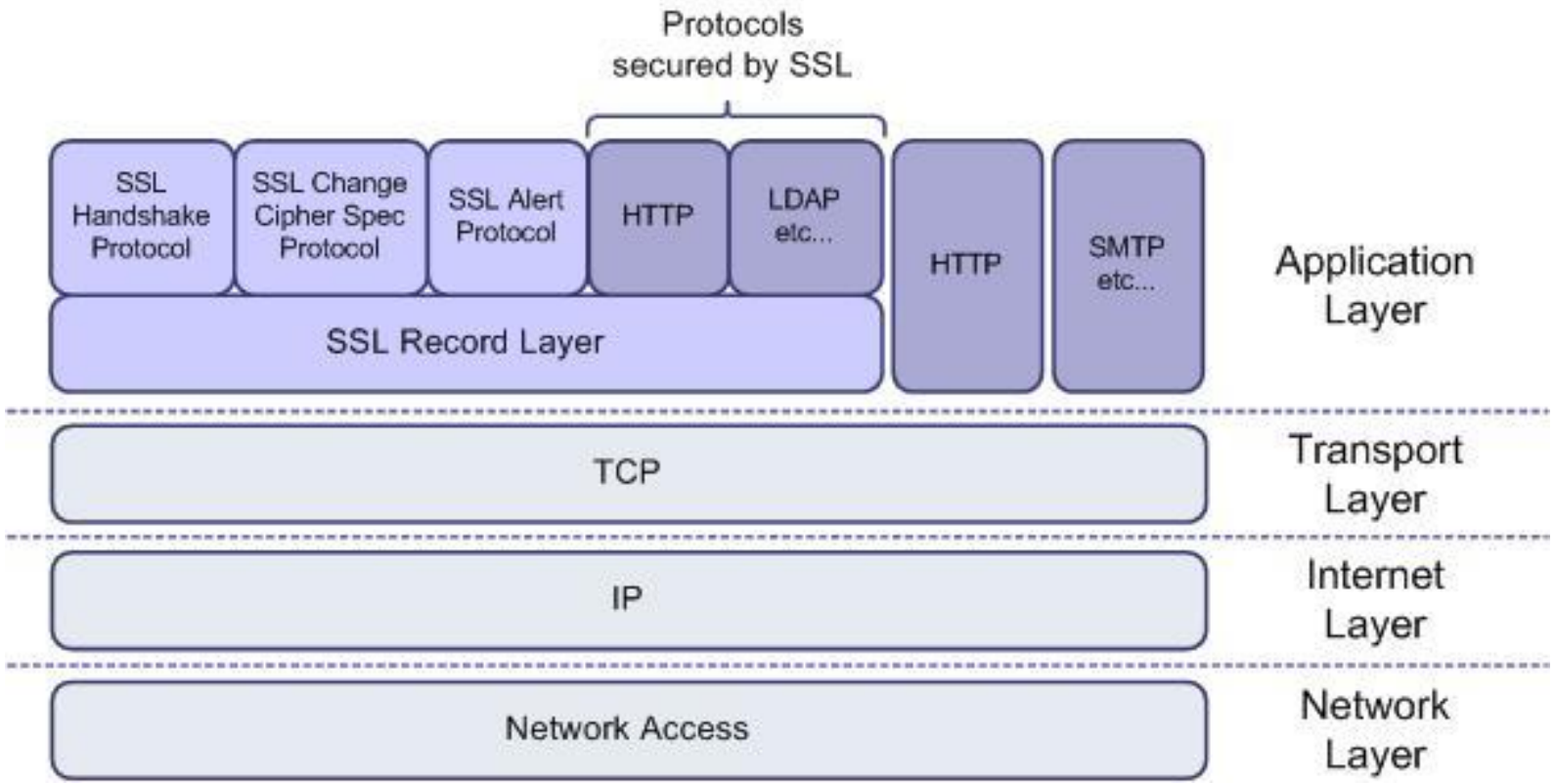


/apis/<group>/<version>/<resource>/<subresource>

- Ingress
  - networking.v1beta1.IngressClass
  - networking.v1beta1.Ingress
- Endpoint
  - core.v1.Endpoints
- Service
  - core.v1.Service
- Secret
  - core.v1.Secret
- ConfigMap
  - core.v1.ConfigMap

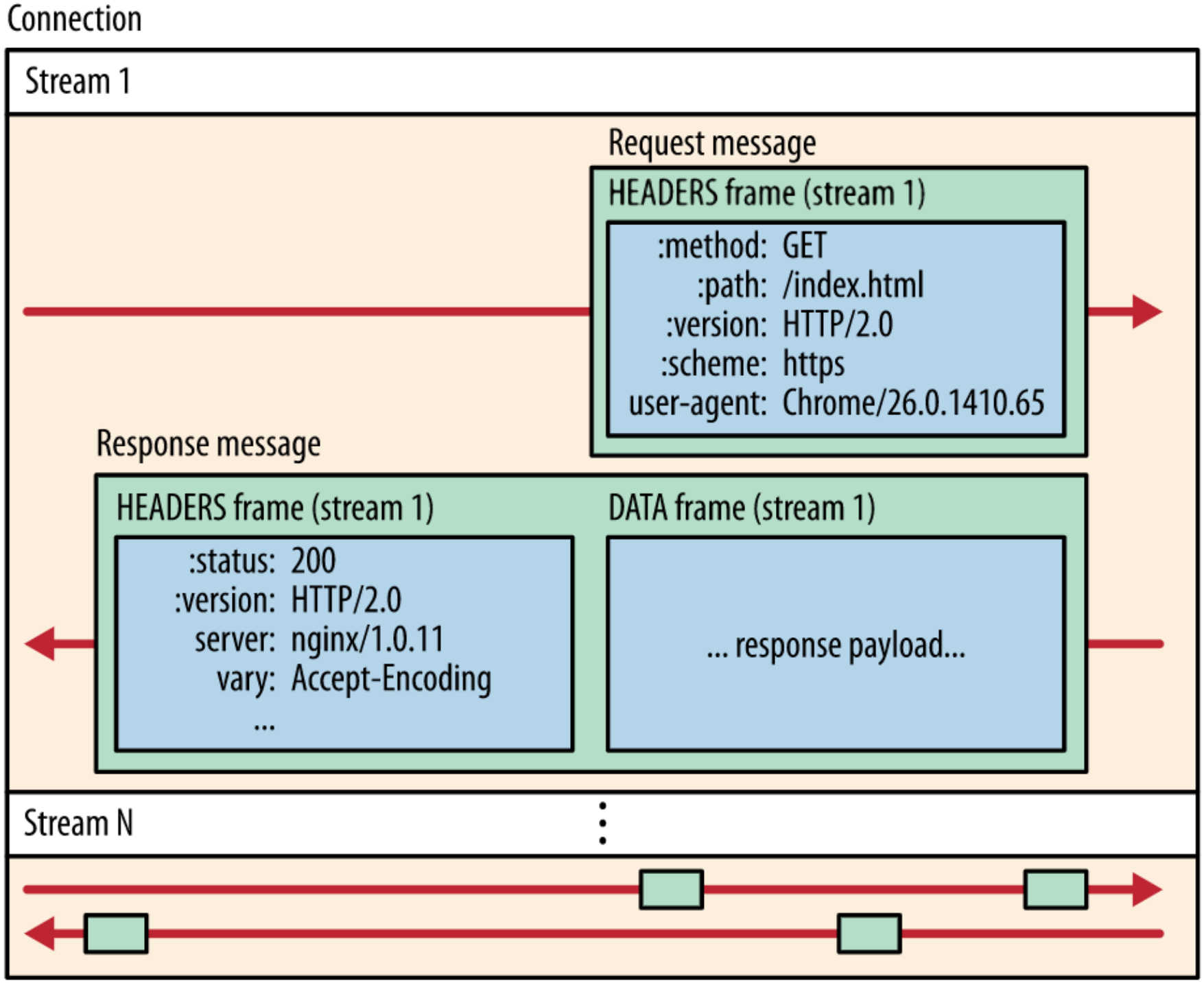


HTTP + TLS + Protobuf/Json/Yaml



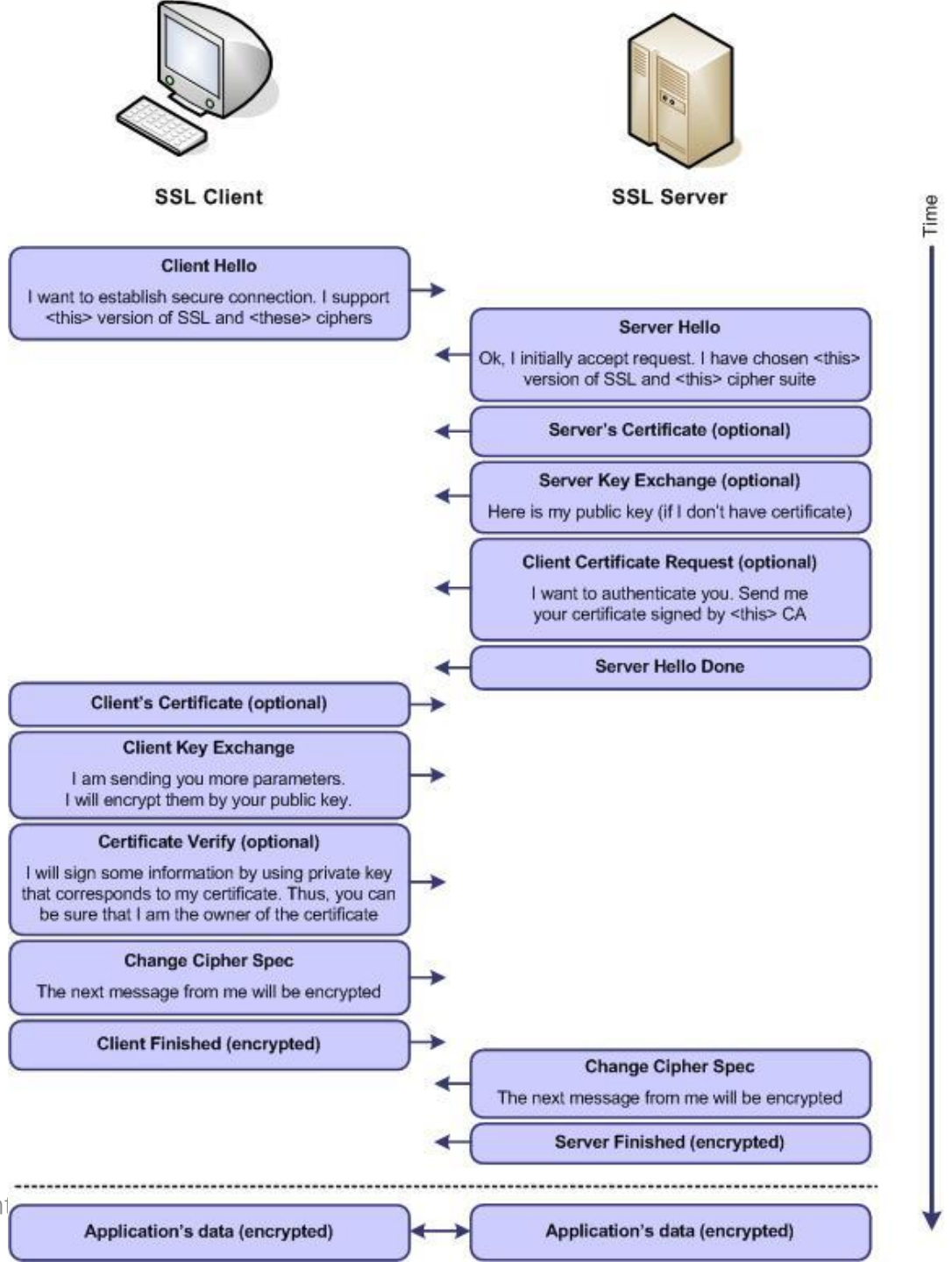
HTTP2

- 多路复用
- 消息推送

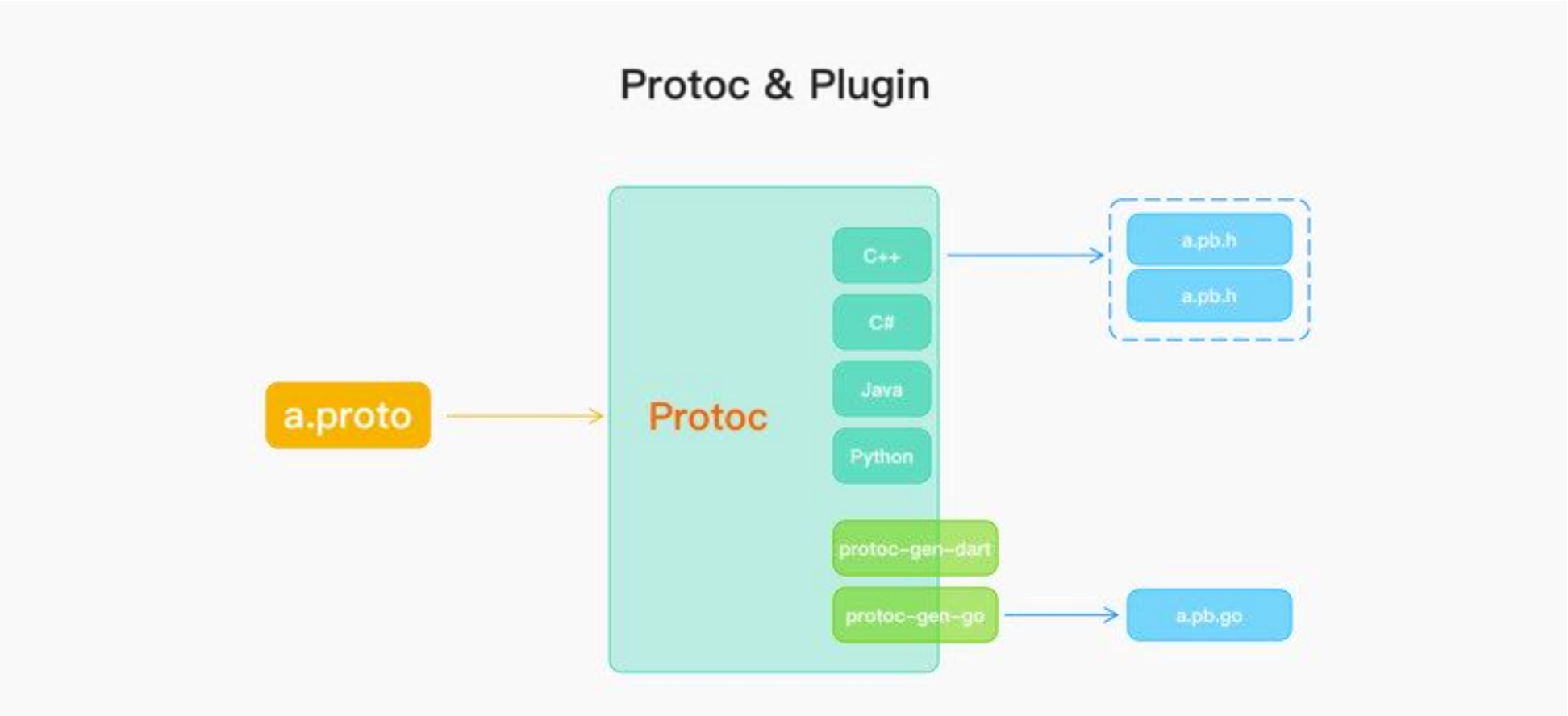


API Server

- 认证
  - BasicAuth、ClientCA、TokenAuth、BootstrapToken、RequestHeader、WebhookTokenAuth、Anonymous、OIDC、ServiceAccountAuth
- 授权
  - AlwaysAllow、AlwaysDeny、ABAC、Webhook、RBAC、Node
- 准入控制器



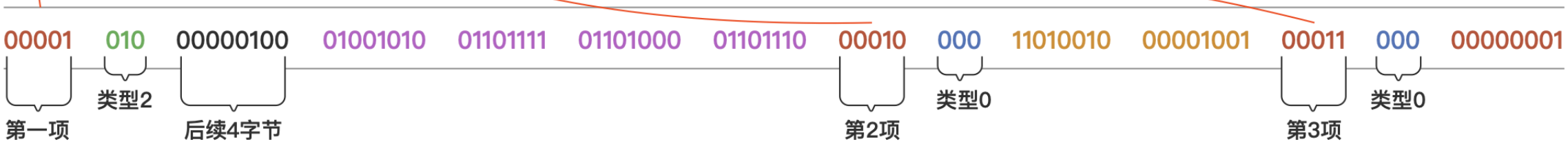
Protobuf编解码

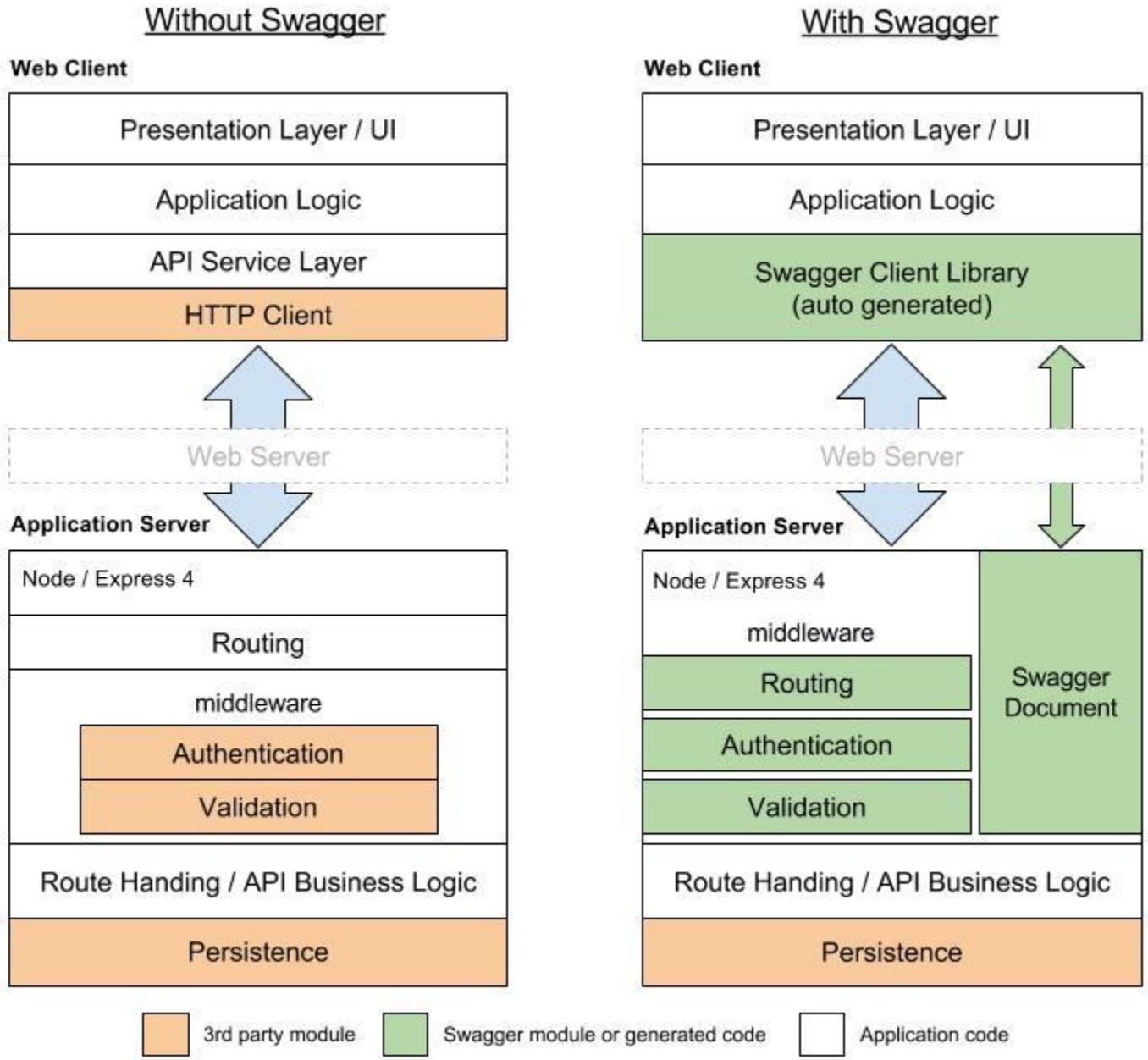


{"name": "John", "id": 1234, "sex": "FEMALE"}

```
message Person{
    string name=1
    unit32 id=2

enum/Sex Type{
    MALE0;
    FEMALE=1;
}
Sex Type sex=3
}
```

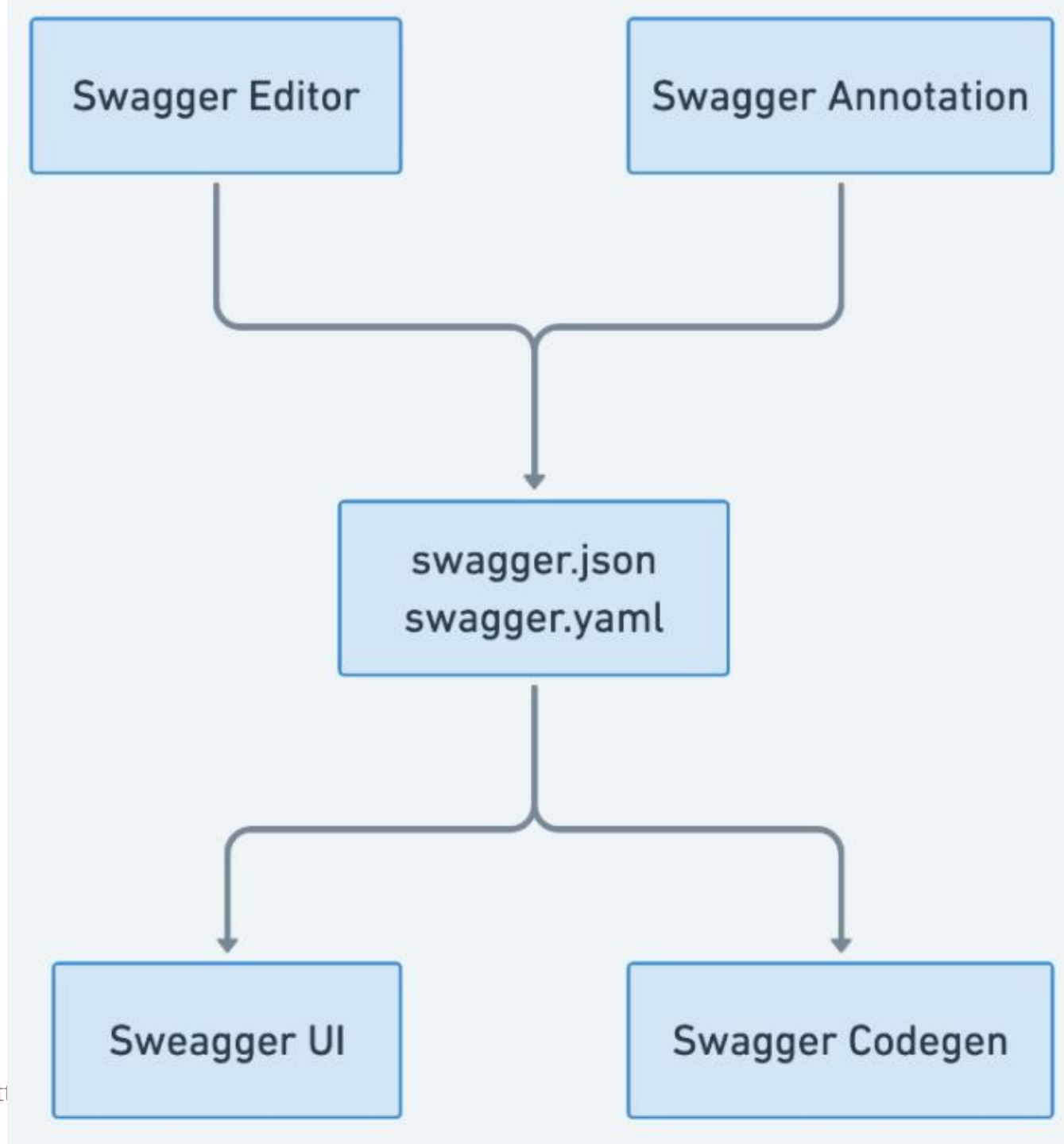
类型	含义	原始数据类型
0	Varint	int 32, int 64, uint 32, uint 64, sint 32, sint 64, bool, enum
1	64-bit	fixed 64, s fixed 64, double
2	Length-delimited	string, bytes, embedded messages, packed repeated fields
3	Start group	groups(deprecated)
4	End group	groups(deprecated)
5	32-bit	fixed 32, s fixed 32, float




Without Swagger
Web Client
Presentation Layer / UI
Application Logic
API Service Layer
HTTP Client
Web Server
Application Server
Node / Express 4
Routing
middleware
Authentication
Validation
Route Handing / API Business Logic
Persistence
With Swagger
Web Client
Presentation Layer / UI
Application Logic
Swagger Client Library
(auto generated)
Web Server
Application Server
Node / Express 4
middleware
Routing
Authentication
Validation
Swagger
Document
Route Handing / API Business Logic
Persistence
3rd party module
Swagger module or generated code
Application code

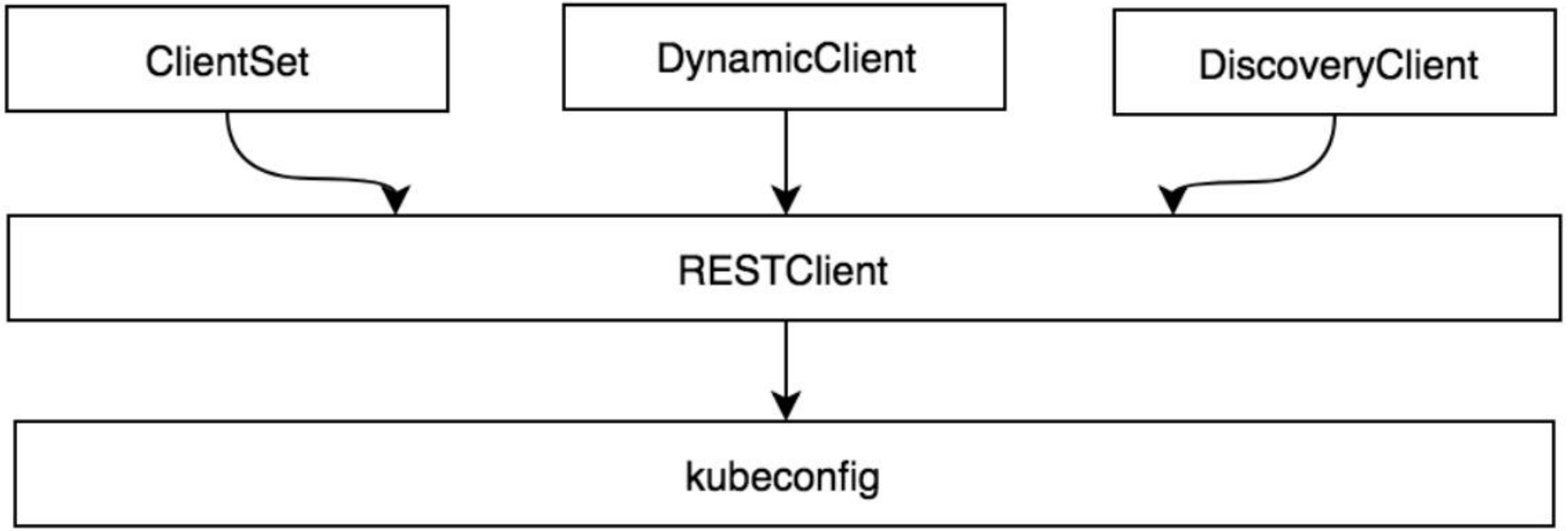
Swagger OpenAPI

- Editor
  - https://editor.swagger.io/
- UI
  - https://petstore.swagger.io/
- Codegen
  - https://github.com/swagger-api/swagger-codegen



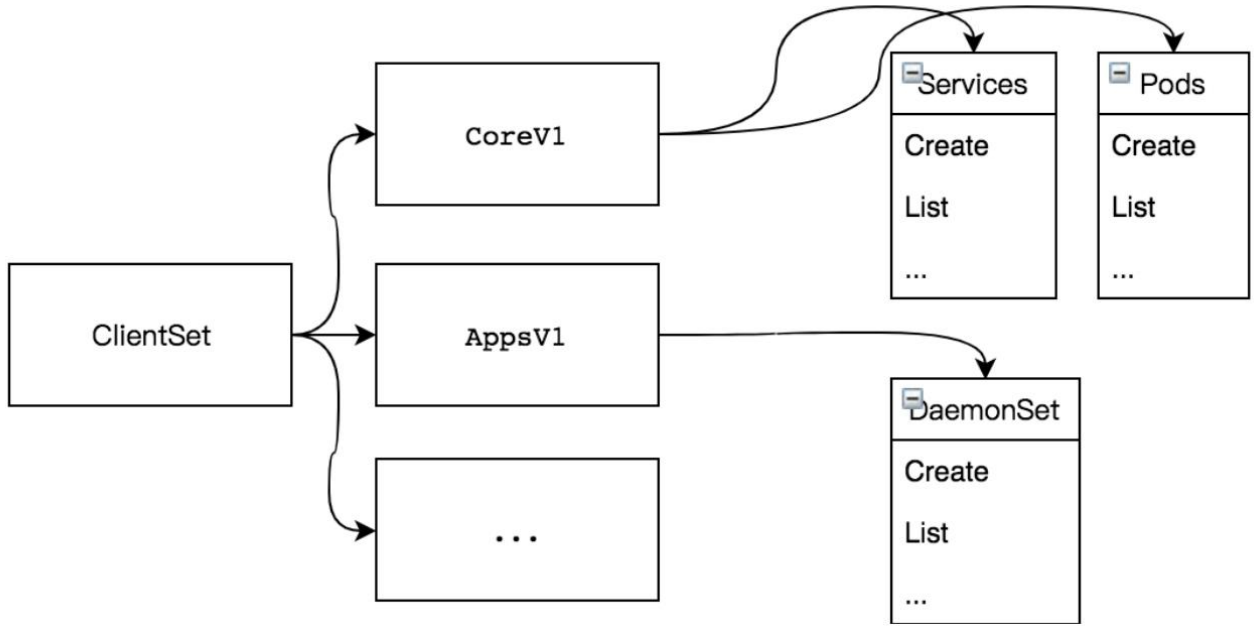
client-go

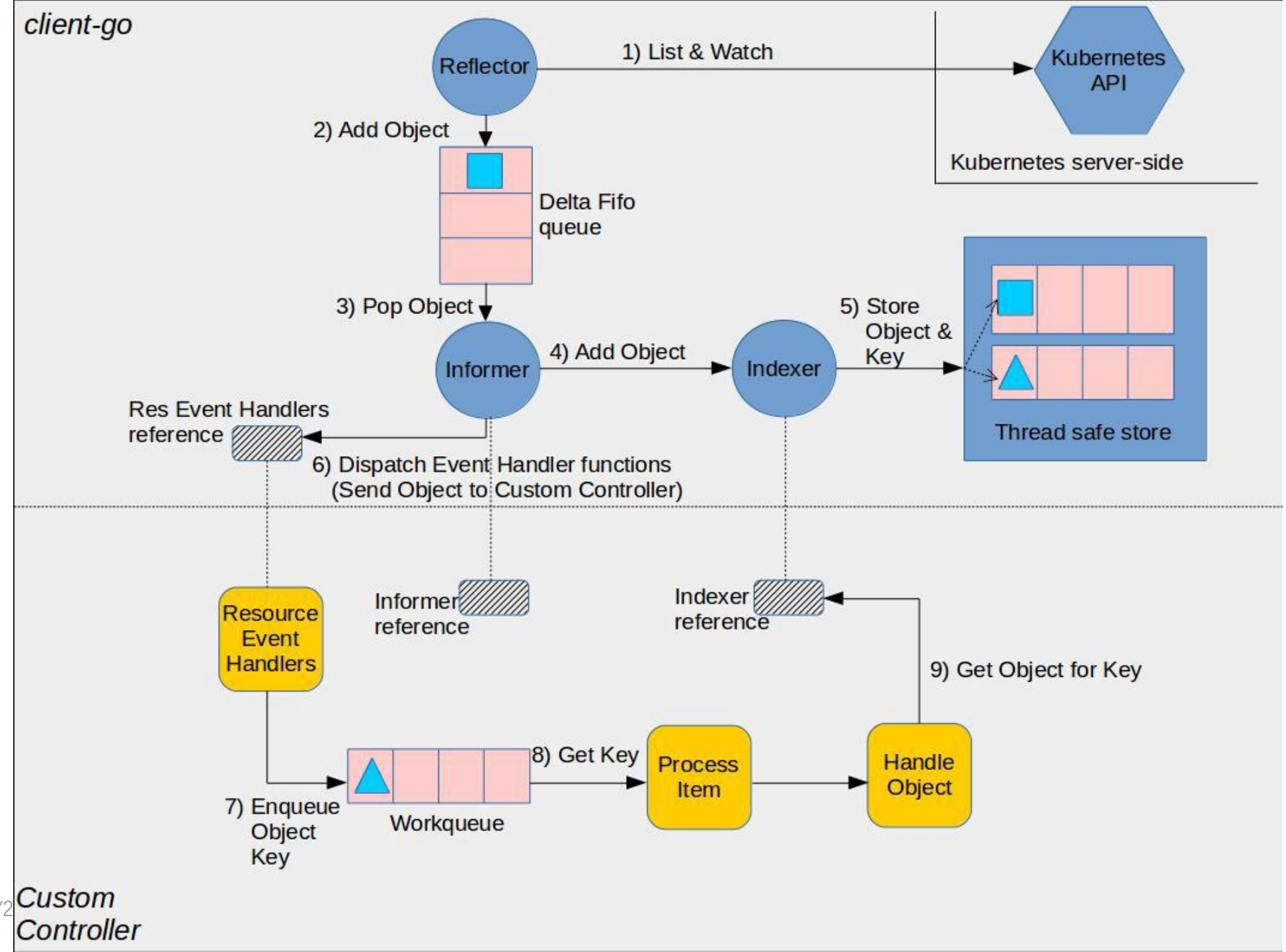
- https://github.com/kubernetes/client-go



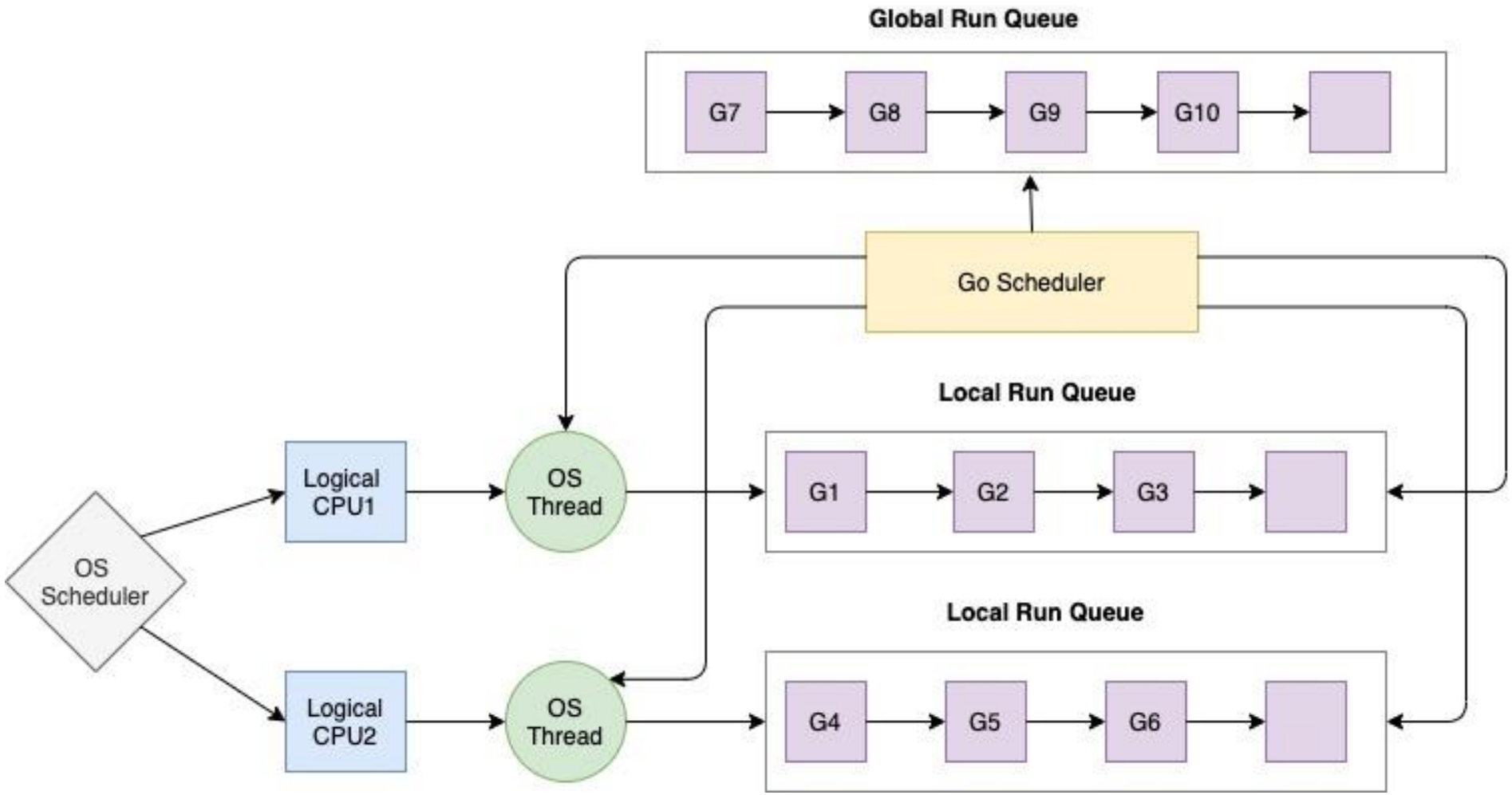
$HOME/.kube/config配置文件

- **clusters**
  - server
- **users**
  - name
- **contexts**

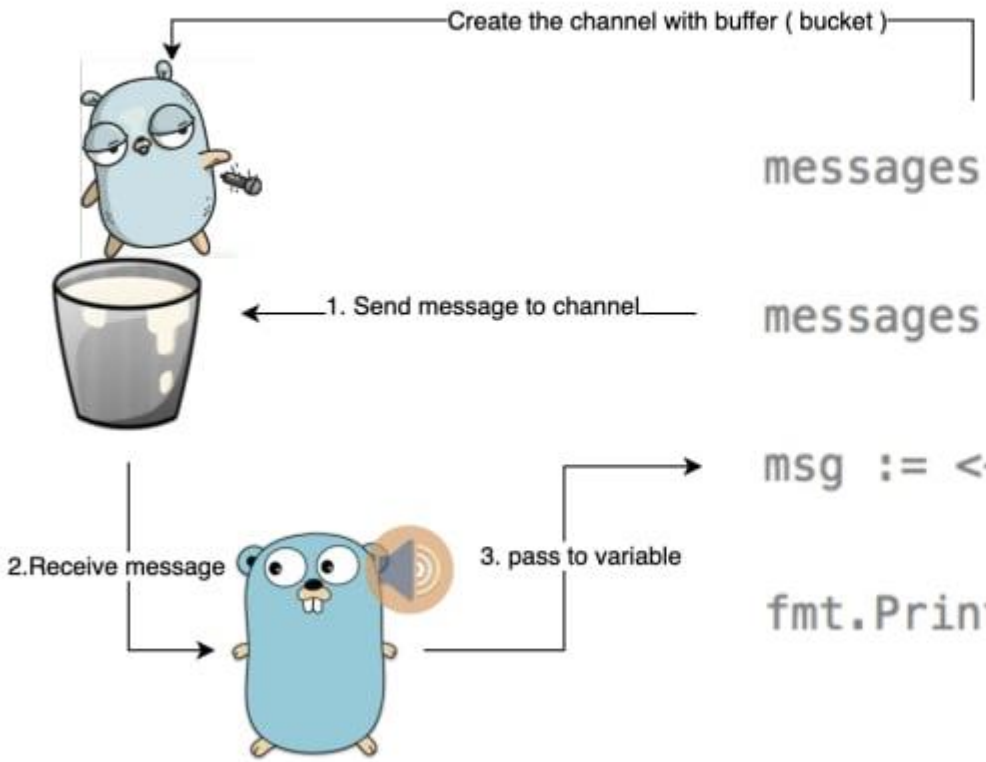



client-go
Reflector
1) List & Watch
Kubernetes API
Kubernetes server-side
2) Add Object
Delta Fifo queue
3) Pop Object
Informer
4) Add Object
Indexer
5) Store Object & Key
Thread safe store
Res Event Handlers reference
6) Dispatch Event (Send Object to Handler functions Custom Controller)
Resource Event Handlers
Informer reference
Indexer reference
9) Get Object for Key
8) Get Key
Process Item
Handle Object
7) Enqueue Object Key
Workqueue
Custom Controller

Goroutine: >=8KB

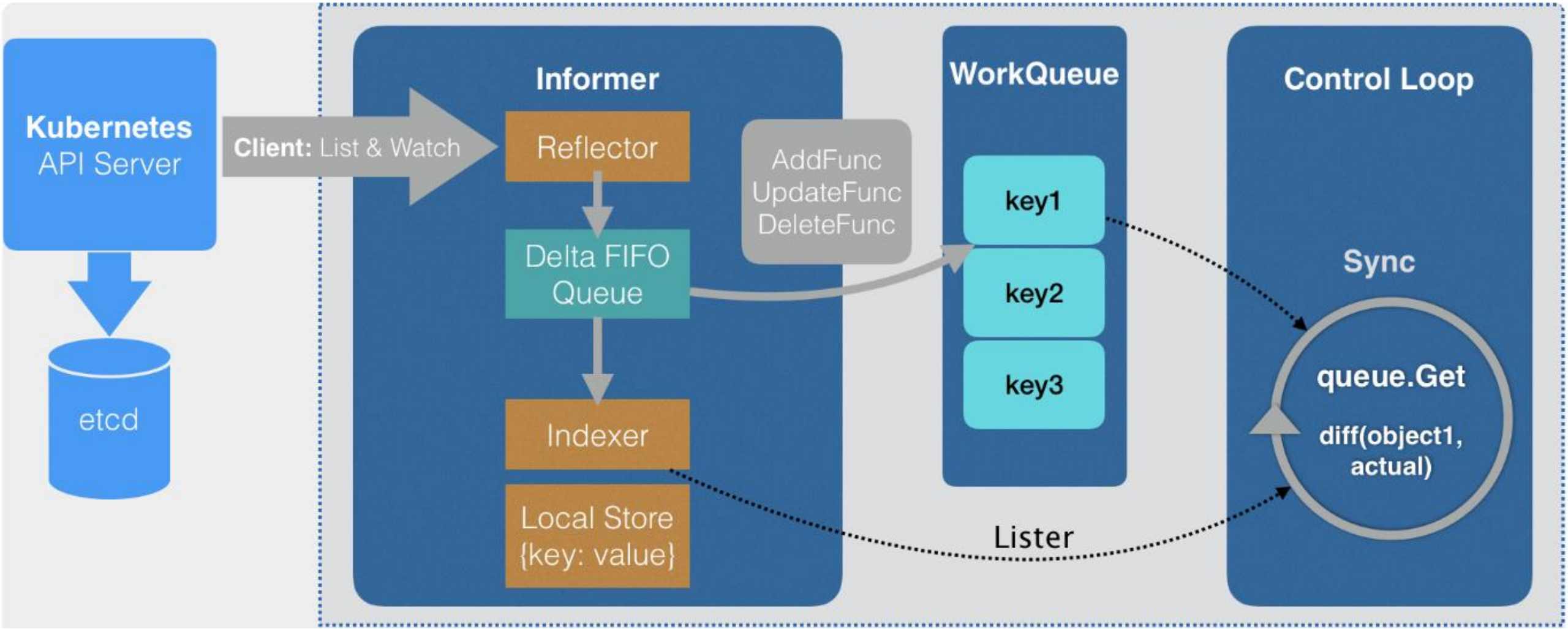


Goroutine通讯：Channel

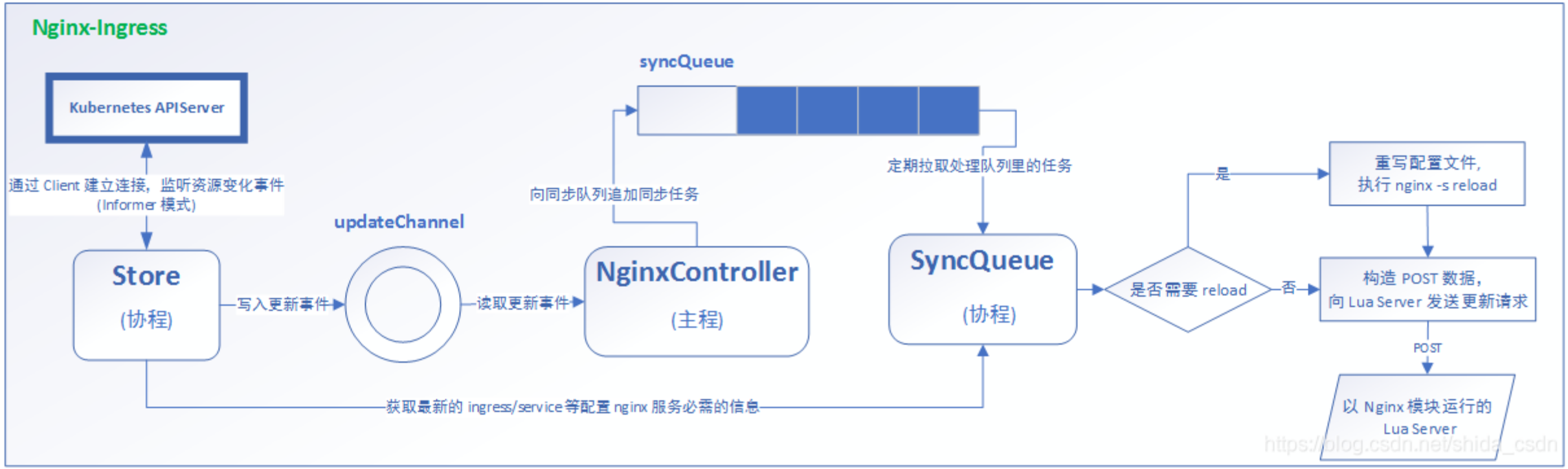


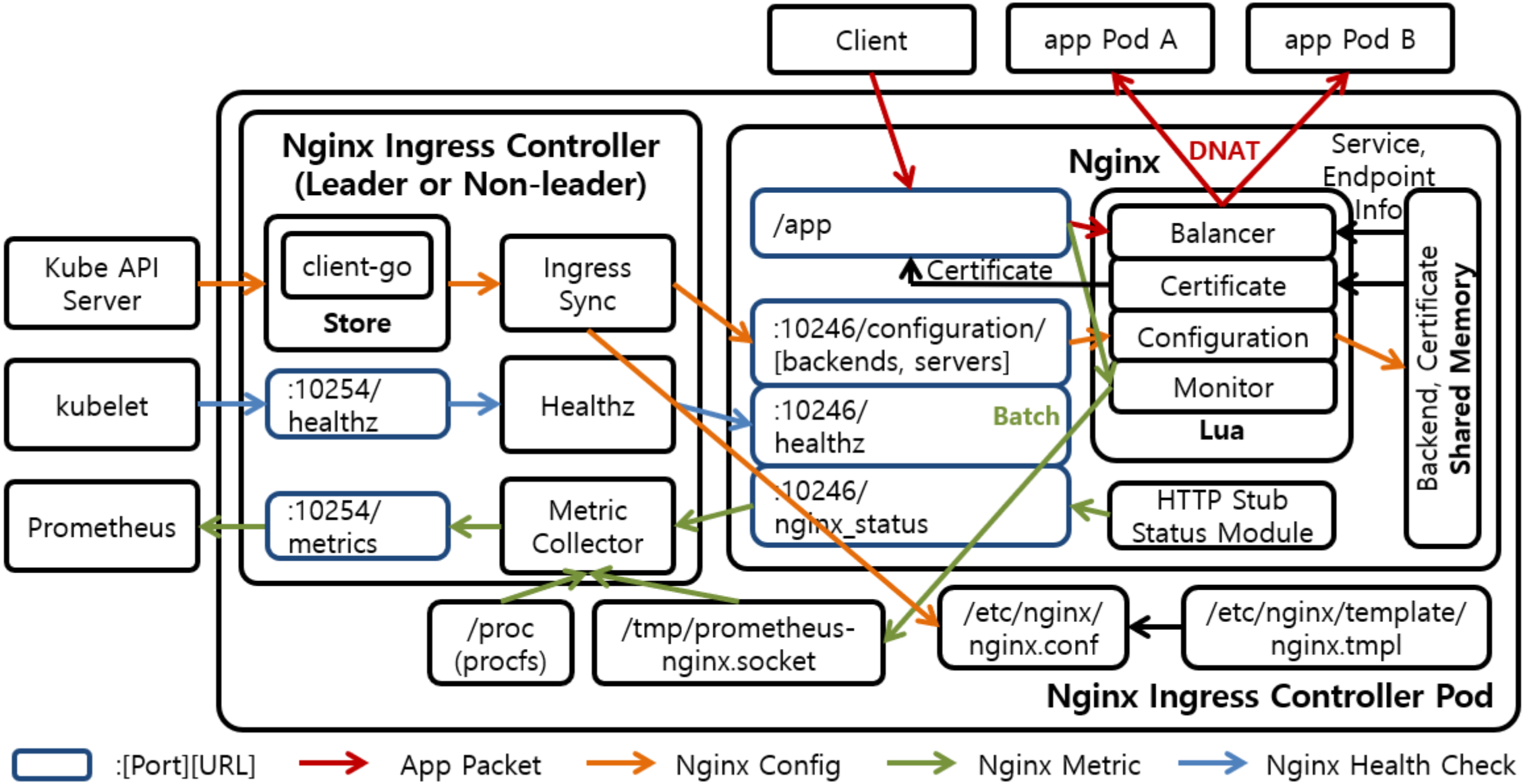

```
messages := make(chan string,1)

messages <- "ping"

msg := <-messages

fmt.Println(msg)
```

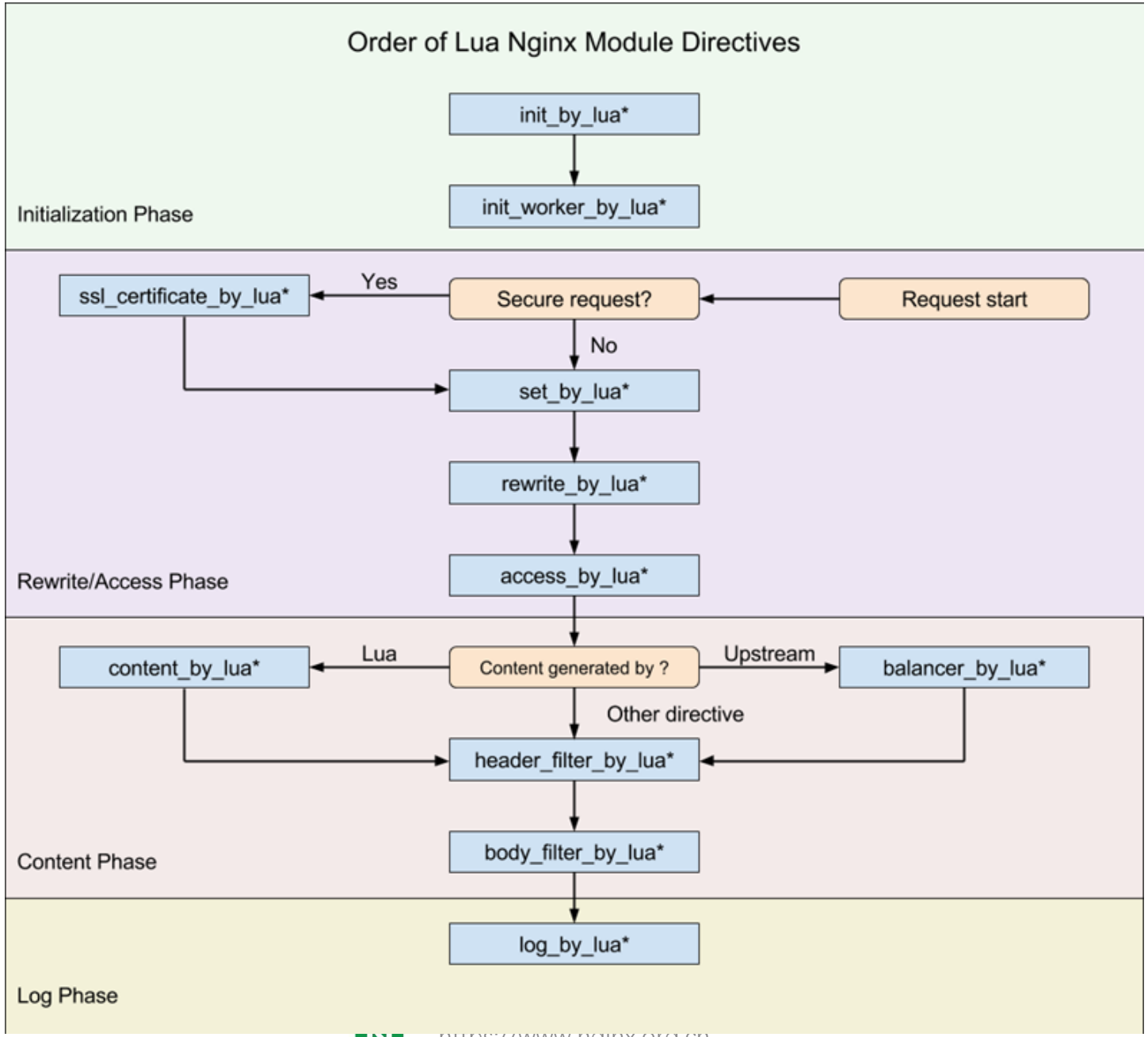
client-go informer原理



K8S官方Ingress Controller架构




Client
app Pod A
app Pod B
Nginx Ingress Controller
(Leader or Non-leader)
Nginx
DNAT
Service,
Endpoint
Info
Kube API
Server
client-go
Store
Ingress
Sync
/app
Certificate
Balancer
Certificate
Configuration
Monitor
Lua
Backend, Certificate
Shared Memory
kubelet
:10254/
healthz
Healthz
:10246/configuration/
[backends, servers]
:10246/
healthz
Batch
Prometheus
:10254/
metrics
Metric
Collector
:10246/
nginx_status
HTTP Stub
Status Module
/proc
(procfs)
/tmp/prometheus-
nginx.socket
/etc/nginx/
nginx.conf
/etc/nginx/template/
nginx.tmpl
Nginx Ingress Controller Pod
:[Port][URL]
App Packet
Nginx Config
Nginx Metric
Nginx Health Check

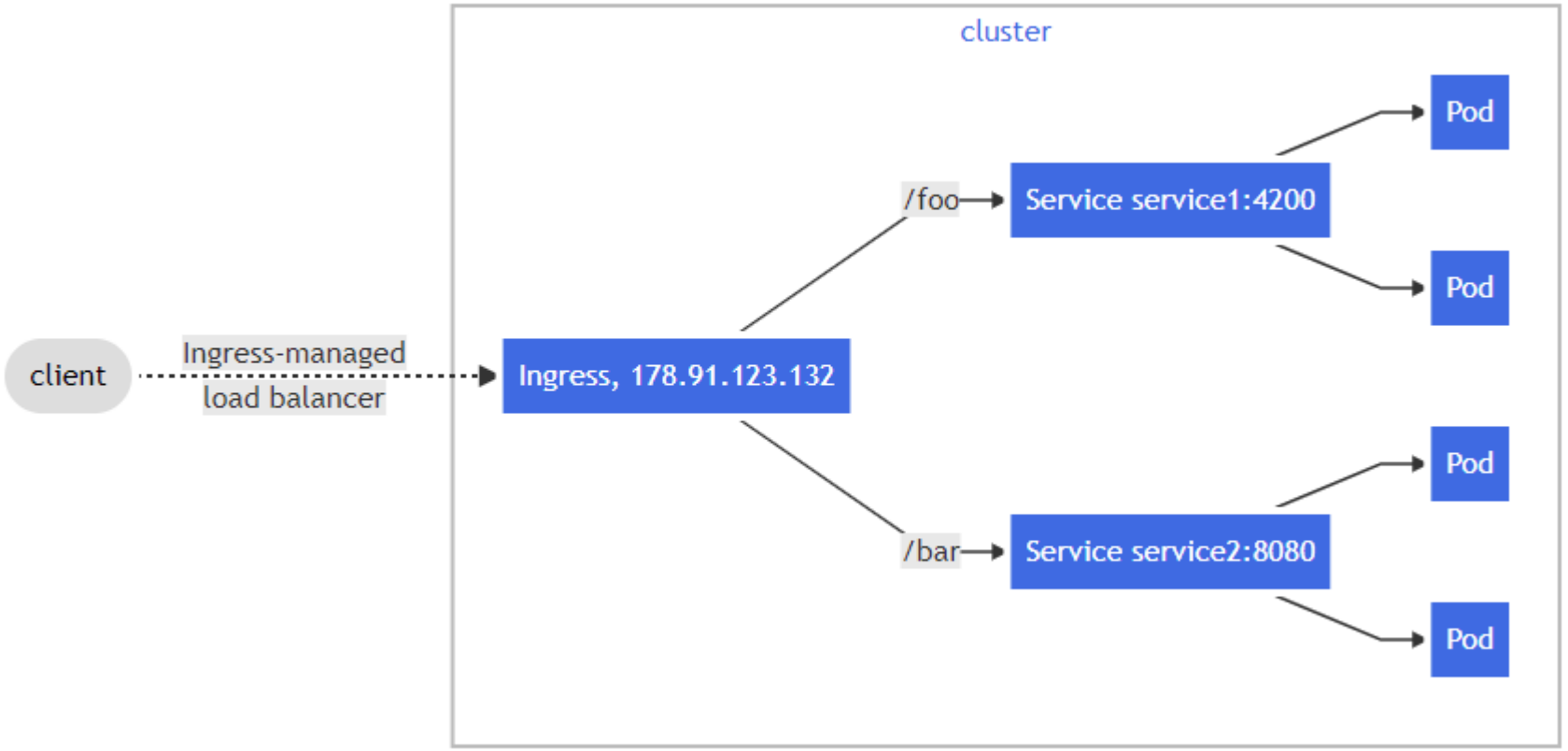

Order of Lua Nginx Module Directives
init_by_lua*
init_worker_by_lua*
Initialization Phase
ssl_certificate_by_lua*
Yes
Secure request?
Request start
No
set_by_lua*
rewrite_by_lua*
access_by_lua*
Rewrite/Access Phase
content_by_lua*
Lua
Content generated by ?
Upstream
balancer_by_lua*
Other directive
header_filter_by_lua*
body_filter_by_lua*
Content Phase
log_by_lua*
Log Phase

K8S Ingress Controller技术细节探讨

- Ingress Controller的工作原理
- Ingress Controller与Master的通讯机制
- **K8S官方社区Controller的核心特性**
- Nginx官方Controller开源版的核心特性

Ingress Controller要完成哪些工作?

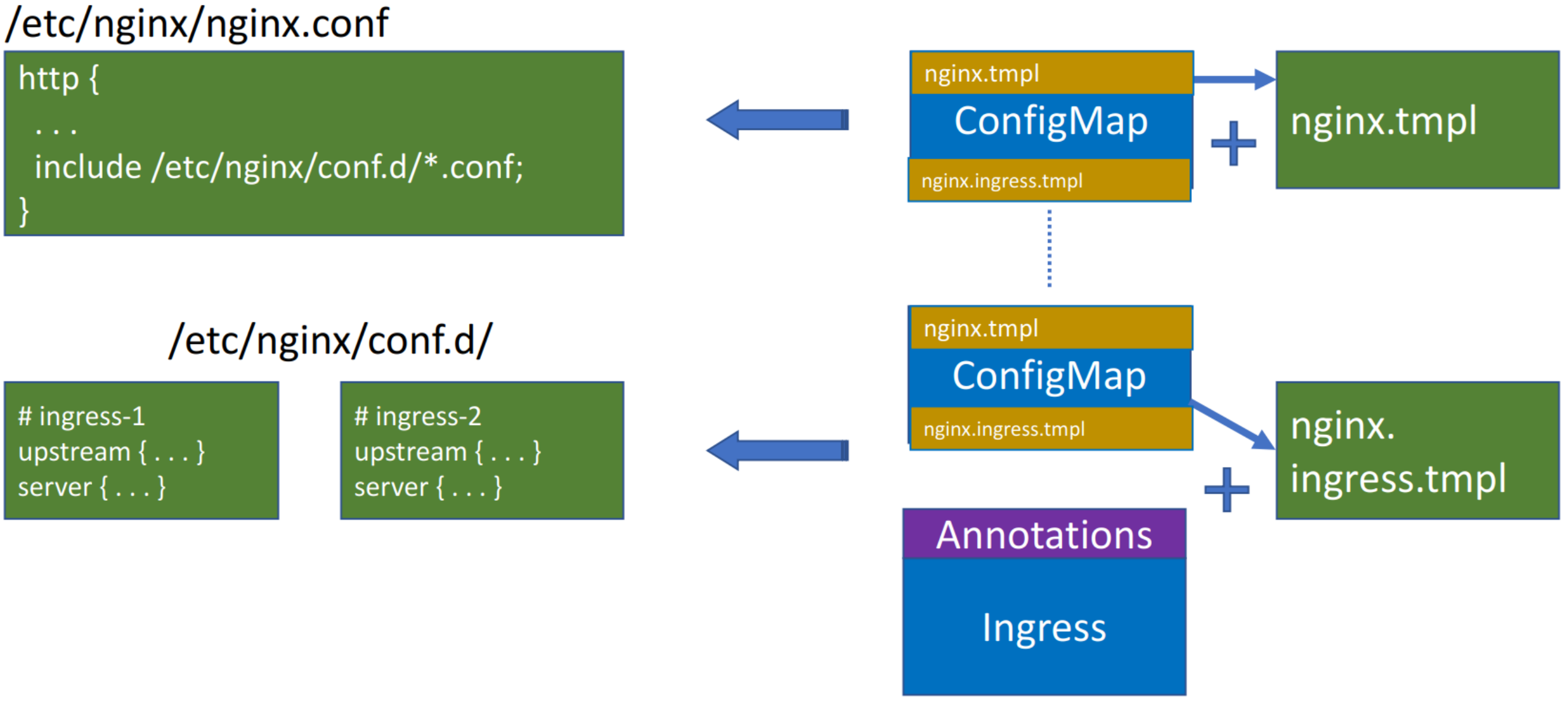
- 负载均衡/会话保持
- 协议转换
- TLS卸载与认证
- 文本压缩
- 请求认证
- 限流限速
- WAF
- 全链路跟踪
- 日志上报



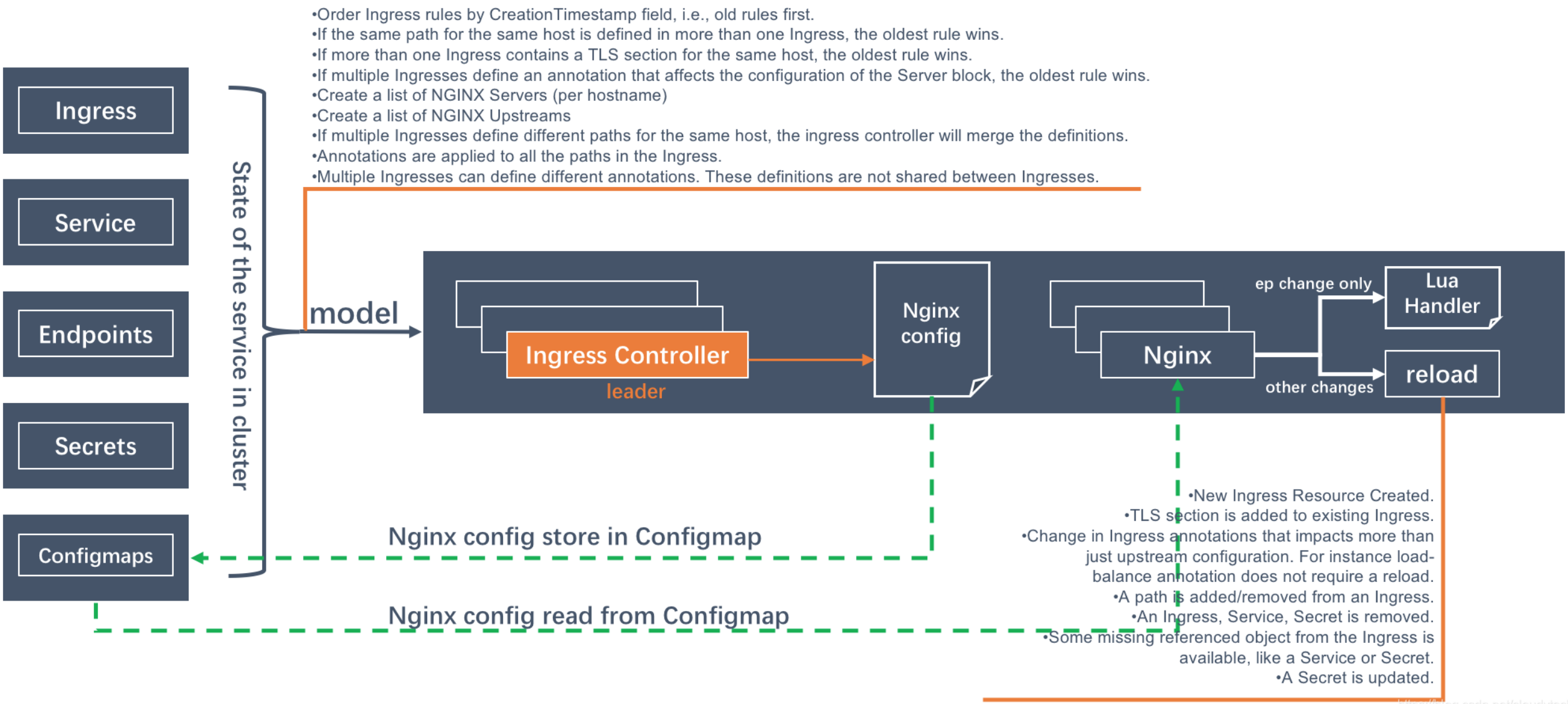
nginx.conf怎么搬到yaml声明配置中?

```
worker_processes    4;
worker_cpu_affinity 0001 0010 0100 1000;
events {  worker_connections 2048; }
http {
        upstream backend {    10.244.1.3 weight=1;     10.244.0.9 weight=3;    least_conn;    }
        server {
                listen 80;
                server_name www.taohui.pub;
                subs_filter_types text/html text/css text/xml;
                subs_filter st(\d*).taohui.tech $1.taohui.org.cn ir;
                location /private/ { auth_request /auth; ... }
                location = /auth {     proxy_pass http://auth.taohui.tech;     }
                location /ajp {   ajp_pass backend;  }
        }
}
stream {
}
```

Ingress/ConfigMap/Annotations/Template



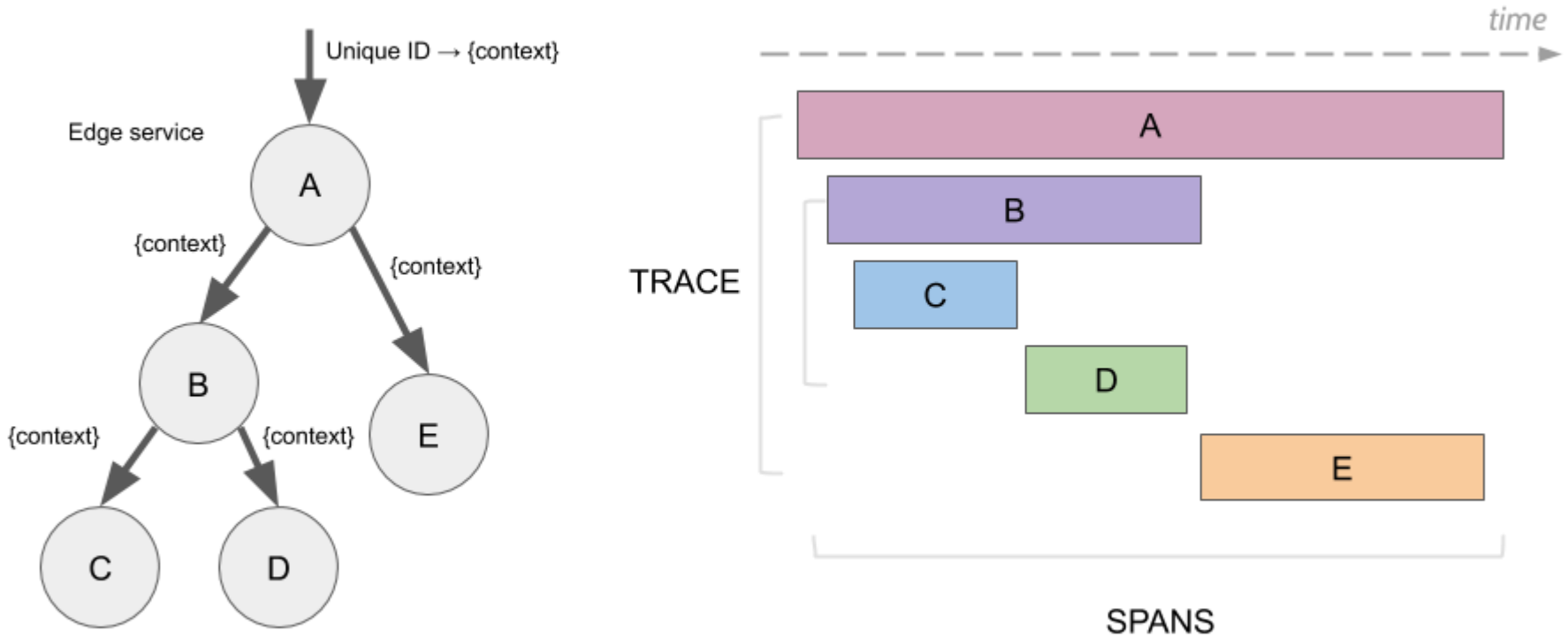
还有Lua…



Nginx模块钩子

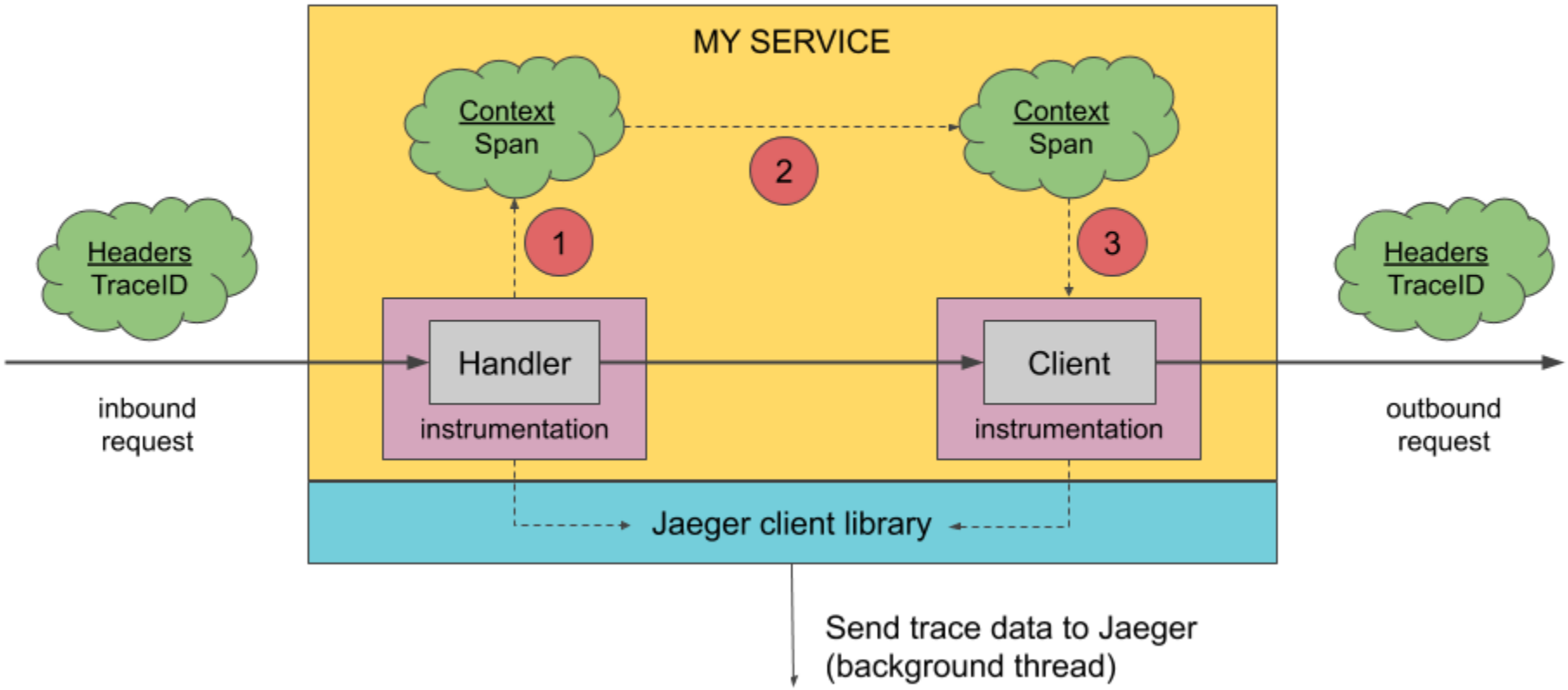
HTTP请求处理阶段	子阶段	模块举例	指令
openssl		lua_nginx_module	ssl_certificated_by_lua
rewrite		ngx_http_opentracing_module	enable-opentracing: "true"
preaccess		ngx_http_limit_req_module ngx_http_limit_conn_module	limit-rps “100” limit-connections “5”
access		ngx_http_auth_digest_module ngx_http_auth_basic_module ngx_http_modsecurity_module	auth-type: “digest” auth-url: “url” modsecurity
content		nginx_ajp_module	ajp_pass tomcats;
	upstream	lua-upstream-nginx-module	balance_by_lua
	filter	ngx_http_brotli_filter_module	enable-brotli “true”
log		nginx-influxdb-module	enable-influxdb “true”

全链路跟踪的实现原理

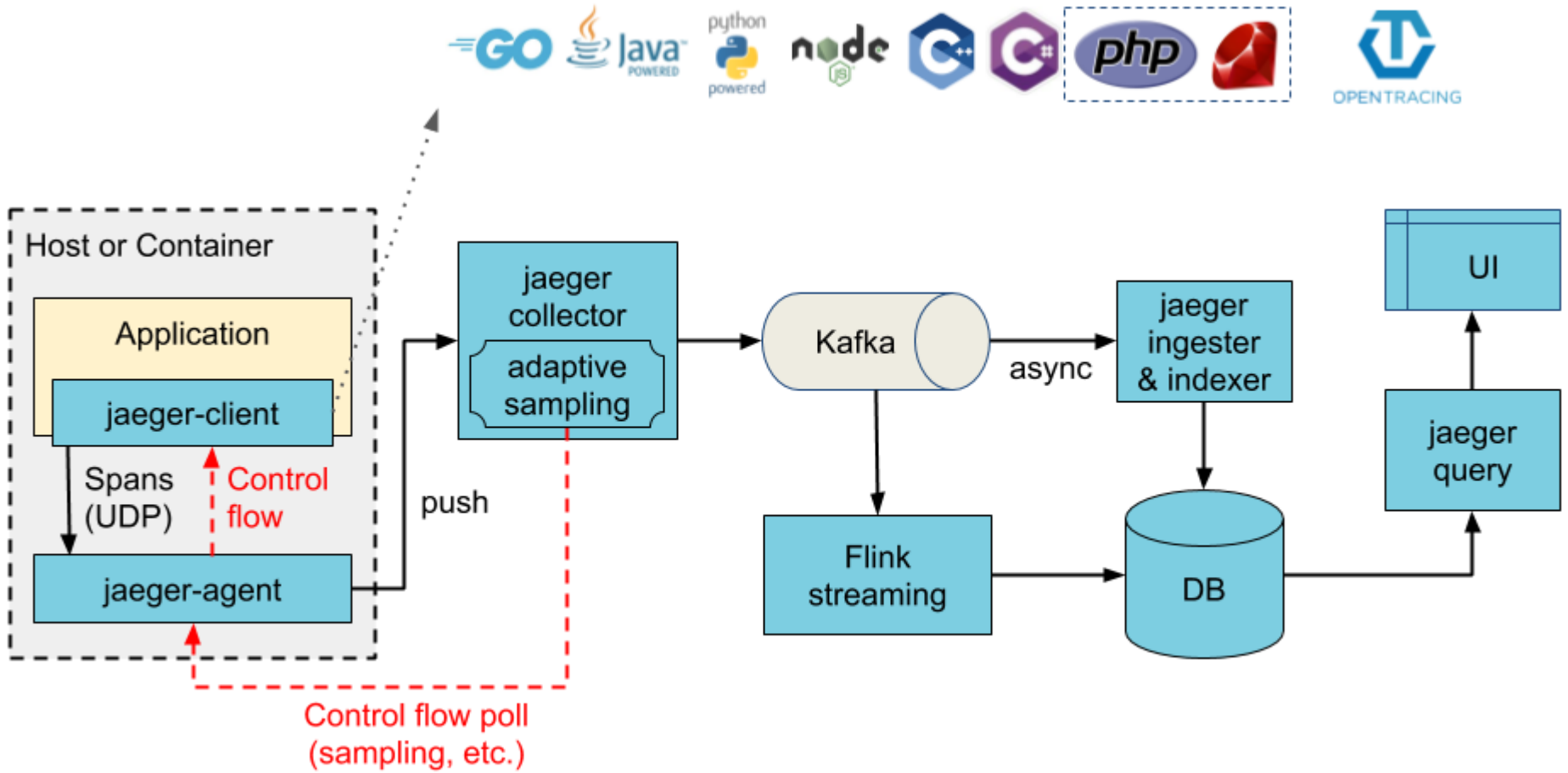

Unique ID → {context}
Edge service
A
{context}
{context}
B
{context}
{context}
E
C
D
time
TRACE
A
B
C
D
E
SPANS

opentracing规范

https://opentracing.io/



全链路跟踪的实现者：jaeger



基于opentracing接口实现全链路跟踪

- nginx-opentracing模块
  - https://github.com/opentracing-contrib/nginx-opentracing
    - https://github.com/opentracing/opentracing-cpp
    - https://github.com/jaegertracing/jaeger-client-cpp
    - https://github.com/rnburn/zipkin-cpp-opentracing
    - https://github.com/DataDog/dd-opentracing-cpp
    - ~~https://github.com/lightstep/lightstep-tracer-cpp~~
  - C++库，支持Jaeger, Zipkin, ~~LightStep~~, Datadog.
  - 指令
    - opentracing_load_tracer libjaegertracing_plugin.so jaeger-nginx-config.json
    - opentracing on;
    - opentracing_tag http_user_agent $http_user_agent;
    - opentracing_operation_name $uri;
    - opentracing_(_fastcgi/_grpc)_propagate_context/;
    - opentracing_trust_incoming_span
    - opentracing_location_operation_name
    - opentracing_trace_locations

Nginx模块钩子

HTTP请求处理阶段	子阶段	模块举例	指令
openssl		lua_nginx_module	ssl_certificated_by_lua
rewrite		ngx_http_opentracing_module	enable-opentracing: "true"
preaccess		ngx_http_limit_req_module ngx_http_limit_conn_module	limit-rps “100” limit-connections “5”
access		ngx_http_auth_digest_module ngx_http_auth_basic_module ngx_http_modsecurity_module	auth-type: “digest” auth-url: “url” modsecurity
content		nginx_ajp_module	ajp_pass tomcats;
	upstream	lua-upstream-nginx-module	balance_by_lua
	filter	ngx_http_brotli_filter_module	enable-brotli “true”
log		nginx-influxdb-module	enable-influxdb “true”

限流限速(1)

- ngx_http_limit_req_module模块
  - http://nginx.org/en/docs/http/ngx_http_limit_req_module.html
  - 指令
    - limit_req_zone key zone=name:size rate=rate [sync];
      - nginx.ingress.kubernetes.io/limit-rps
      - limit-mps
    - limit_req zone=name [burst=number] [nodelay | delay=number];
      - limit-burst-multiplier
    - limit_req_dry_run on | off;
    - limit_req_log_level info | notice | warn | error;
    - limit_req_status code;

限流限速(2)

- ngx_http_limit_conn_module模块
  - http://nginx.org/en/docs/http/ngx_http_limit_conn_module.html
  - 指令
    - limit_conn zone number;
      - nginx.ingress.kubernetes.io/limit-connections
    - limit_conn_dry_run on | off;
    - limit_conn_log_level info | notice | warn | error;
    - limit_conn_status code;
    - limit_conn_zone key zone=name:size;

限流限速（3）

- ngx_http_core_module模块
  - http://nginx.org/en/docs/http/ngx_http_core_module.html
  - 指令
    - limit_rate rate;
      - nginx.ingress.kubernetes.io/limit-rate
    - limit_rate_after size;
      - nginx.ingress.kubernetes.io/limit-rate-after

限流限速（4）

- 基于Lua的Global Rate Limiting
  - https://github.com/ElvinEfendi/lua-resty-global-throttle
  - annotations指令
    - nginx.ingress.kubernetes.io/global-rate-limit
    - global-rate-limit-window
    - global-rate-limit-key
    - global-rate-limit-ignored-cidrs

Nginx模块钩子

HTTP请求处理阶段	子阶段	模块举例	指令
openssl		lua_nginx_module	ssl_certificated_by_lua
rewrite		ngx_http_opentracing_module	enable-opentracing: "true"
preaccess		ngx_http_limit_req_module ngx_http_limit_conn_module	limit-rps “100” limit-connections “5”
access		ngx_http_auth_digest_module ngx_http_auth_basic_module ngx_http_modsecurity_module	auth-type: “digest” auth-url: “url” modsecurity
content		nginx_ajp_module	ajp_pass tomcats;
	upstream	lua-upstream-nginx-module	balance_by_lua
	filter	ngx_http_brotli_filter_module	enable-brotli “true”
log		nginx-influxdb-module	enable-influxdb “true”

请求认证auth_digest

- RFC2617: https://datatracker.ietf.org/doc/html/rfc2617
- ngx_http_auth_digest_module模块
  - https://github.com/atomx/nginx-http-auth-digest
  - 指令
    - auth_digest [“realm string” | off]
      - nginx.ingress.kubernetes.io/auth-type: “digest”
      - auth-realm: "realm string"
    - auth_digest_user_file
      - auth-secret-type: [auth-file|auth-map]
      - auth-secret: secretName
    - auth_digest_timeout
    - auth_digest_expires
    - auth_digest_drop_time
    - auth_digest_evasion_time
    - auth_digest_replays
    - auth_digest_maxtries
    - auth_digest_shm_size

请求认证auth_basic

- ngx_http_auth_basic_module模块
  - http://nginx.org/en/docs/http/ngx_http_auth_basic_module.html
  - 指令
    - auth_basic [“realm string” | off]
      - nginx.ingress.kubernetes.io/auth-type: “basic”
      - auth-realm: "realm string"
    - auth_basic_user_file
      - auth-secret-type: [auth-file|auth-map]
      - auth-secret: secretName

请求认证auth_request

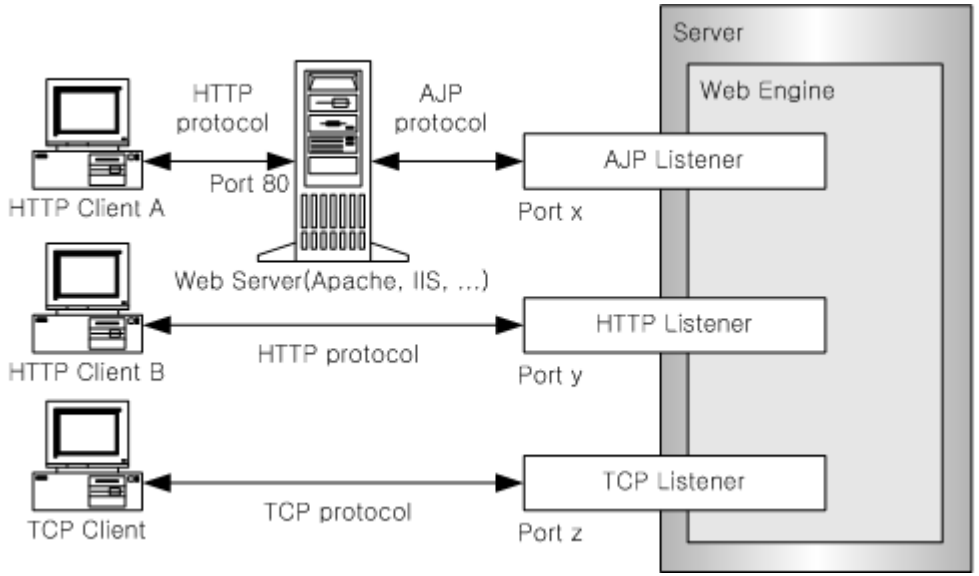
- ngx_http_auth_request_module模块
  - http://nginx.org/en/docs/http/ngx_http_auth_request_module.html
  - 指令
    - auth_request uri | off;
      - nginx.ingress.kubernetes.io/(global-)auth-url: “url”
        - (global-) auth-method
        - (global-) auth-signin
        - (global-) auth-signin-redirect-param
        - (global-) auth-response-headers
        - (global-) auth-proxy-set-headers
        - (global-) auth-request-redirect
        - (global-) auth-cache-key
        - (global-) auth-cache-duration
        - (global-) auth-snippet
    - auth_request_set $variable value;

Nginx模块钩子

HTTP请求处理阶段	子阶段	模块举例	指令
openssl		lua_nginx_module	ssl_certificated_by_lua
rewrite		ngx_http_opentracing_module	enable-opentracing: "true"
preaccess		ngx_http_limit_req_module ngx_http_limit_conn_module	limit-rps “100” limit-connections “5”
access		ngx_http_auth_digest_module ngx_http_auth_basic_module ngx_http_modsecurity_module	auth-type: “digest” auth-url: “url” modsecurity
content		nginx_ajp_module	ajp_pass tomcats;
	upstream	lua-upstream-nginx-module	balance_by_lua
	filter	ngx_http_brotli_filter_module	enable-brotli “true”
log		nginx-influxdb-module	enable-influxdb “true”

AJP: Apache Jserv Protocol

- backend-protocol：AJP
  - HTTP/HTTPS
  - GRPC/GRPCS
  - FASTCGI
- nginx_ajp_module模块
  - https://github.com/yaoweibin/ngx_ajp_module

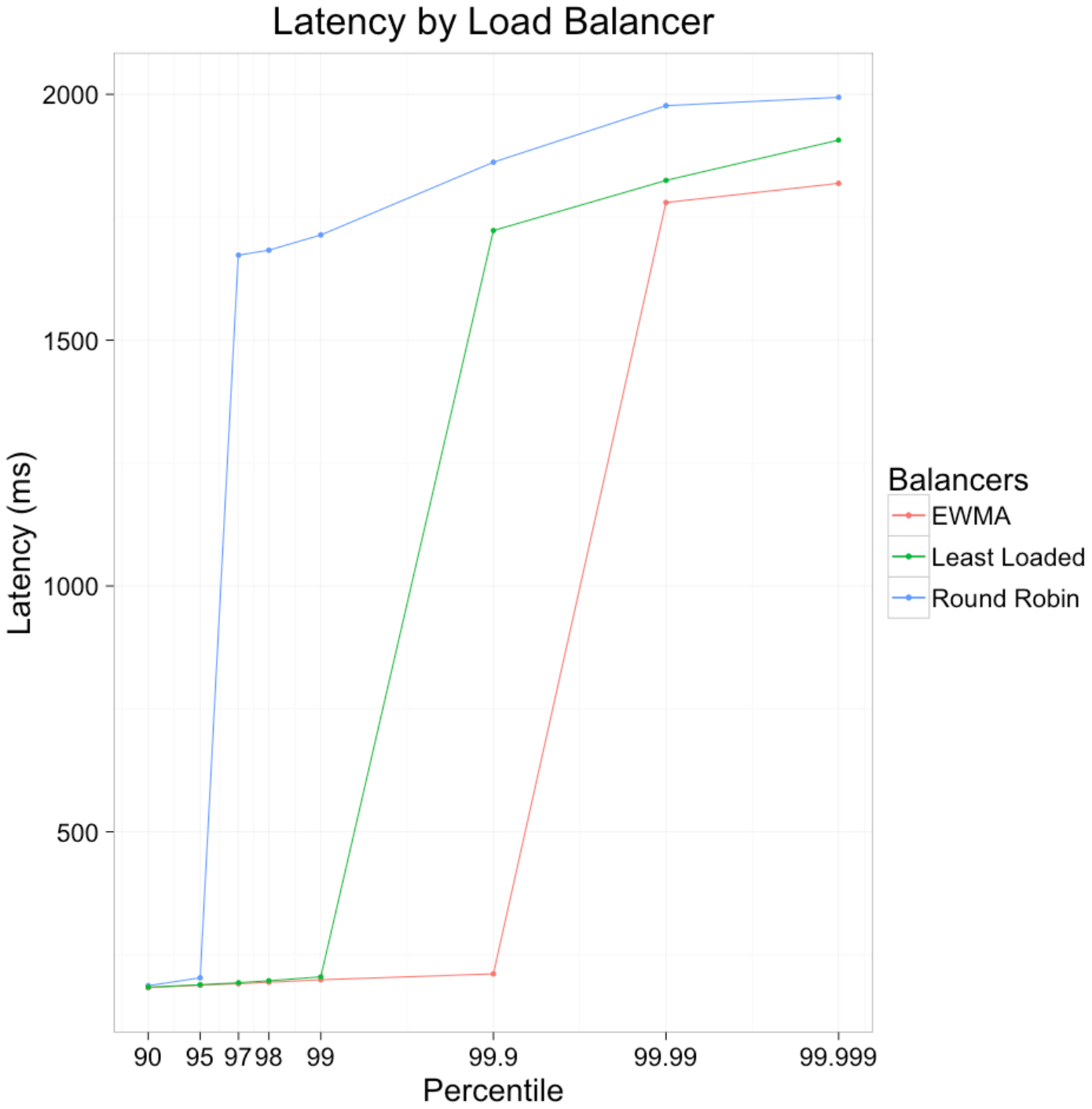


Nginx模块钩子

HTTP请求处理阶段	子阶段	模块举例	指令
openssl		lua_nginx_module	ssl_certificated_by_lua
rewrite		ngx_http_opentracing_module	enable-opentracing: "true"
preaccess		ngx_http_limit_req_module ngx_http_limit_conn_module	limit-rps “100” limit-connections “5”
access		ngx_http_auth_digest_module ngx_http_auth_basic_module ngx_http_modsecurity_module	auth-type: “digest” auth-url: “url” modsecurity
content		nginx_ajp_module	ajp_pass tomcats;
	upstream	lua-upstream-nginx-module	balance_by_lua
	filter	ngx_http_brotli_filter_module	enable-brotli “true”
log		nginx-influxdb-module	enable-influxdb “true”

负载均衡算法

- load-balance
  - round_robin
  - ewma
    - Exponentially Weighted Moving Averages
- upstream-hash-by
  - upstream-hash-by-subset
  - upstream-hash-by-subset-size



会话保持

- affinity: cookie
- affinity-mode: balanced/persistent
- session-cookie-name
- session-cookie-path
- session-cookie-samesite
- session-cookie-conditional-samesite-none
- session-cookie-max-age
- session-cookie-expires
- session-cookie-change-on-failure：false/true

K8S Ingress Controller技术细节探讨

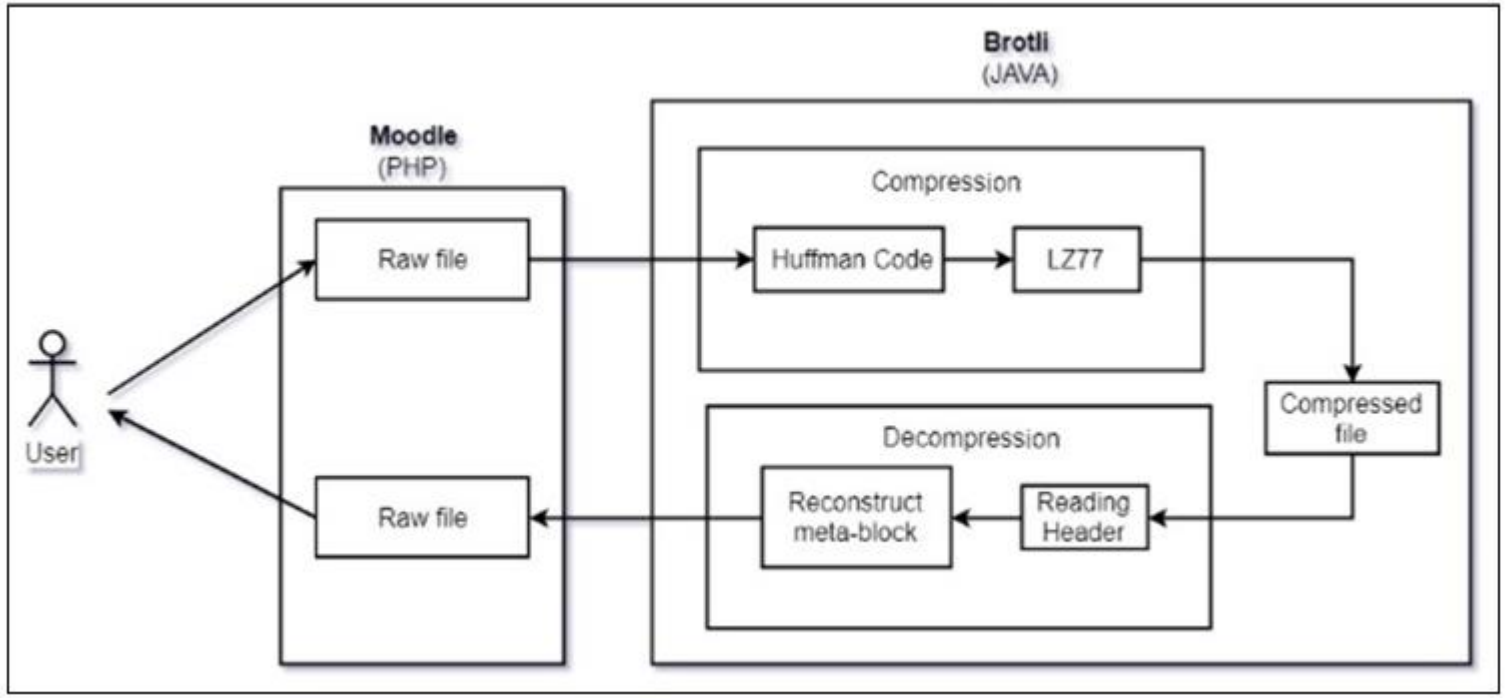
- Ingress Controller的工作原理
- Ingress Controller与Master的通讯机制
- K8S官方社区Controller的核心特性
- **Nginx官方Controller开源版的核心特性**

Nginx模块钩子

HTTP请求处理阶段	子阶段	模块举例	指令
openssl		lua_nginx_module	ssl_certificated_by_lua
rewrite		ngx_http_opentracing_module	enable-opentracing: "true"
preaccess		ngx_http_limit_req_module ngx_http_limit_conn_module	limit-rps “100” limit-connections “5”
access		ngx_http_auth_digest_module ngx_http_auth_basic_module ngx_http_modsecurity_module	auth-type: “digest” auth-url: “url” modsecurity
content		nginx_ajp_module	ajp_pass tomcats;
	upstream	lua-upstream-nginx-module	balance_by_lua
	filter	ngx_http_brotli_filter_module	enable-brotli “true”
log		nginx-influxdb-module	enable-influxdb “true”

brotli压缩算法

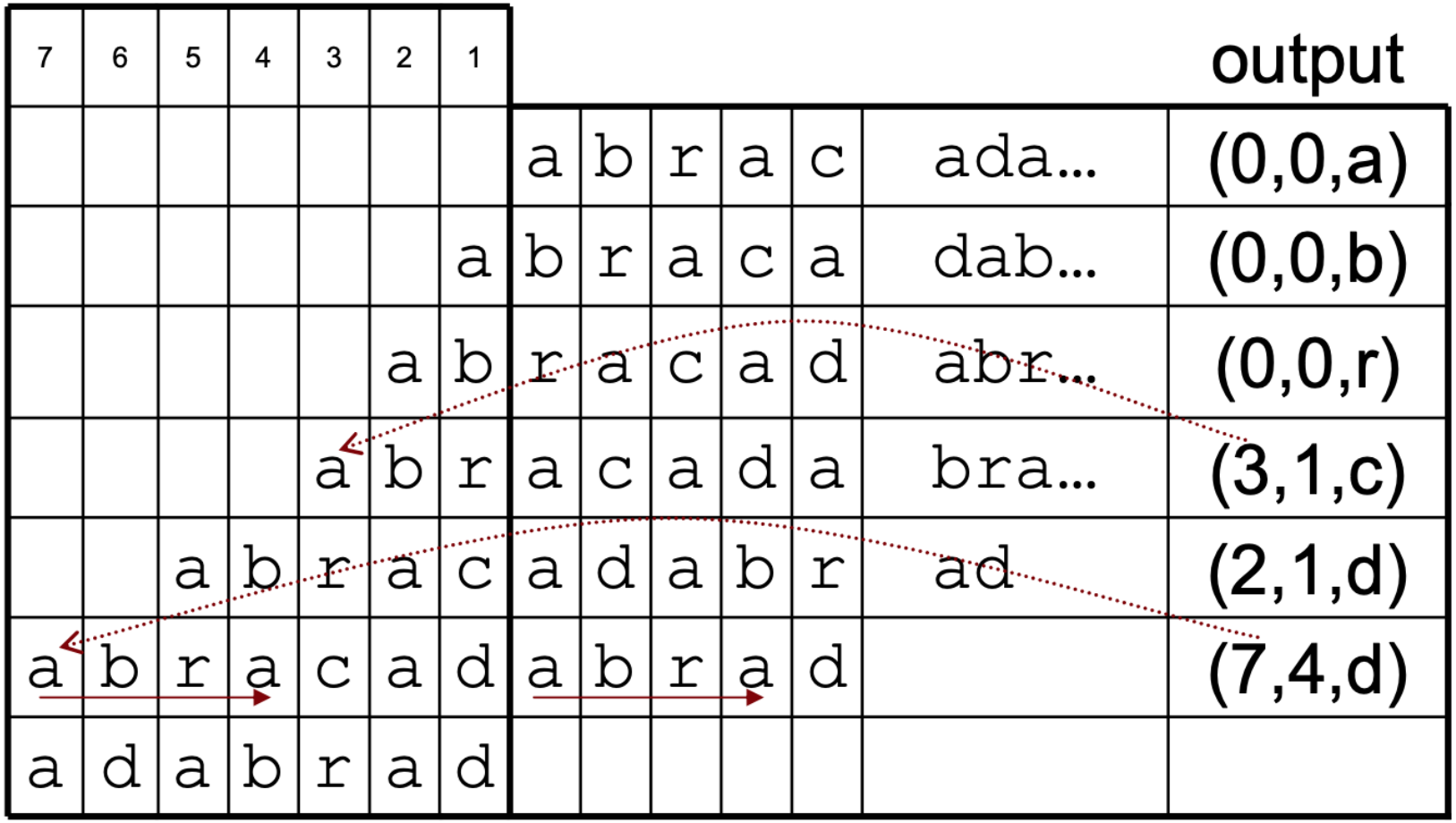
- http://www.icicelb.org/ellb/contents/2019/11/elb-10-11-02.pdf
- https://github.com/google/brotli



LZ77动态字典

Encoding of the string:
abracadabrad

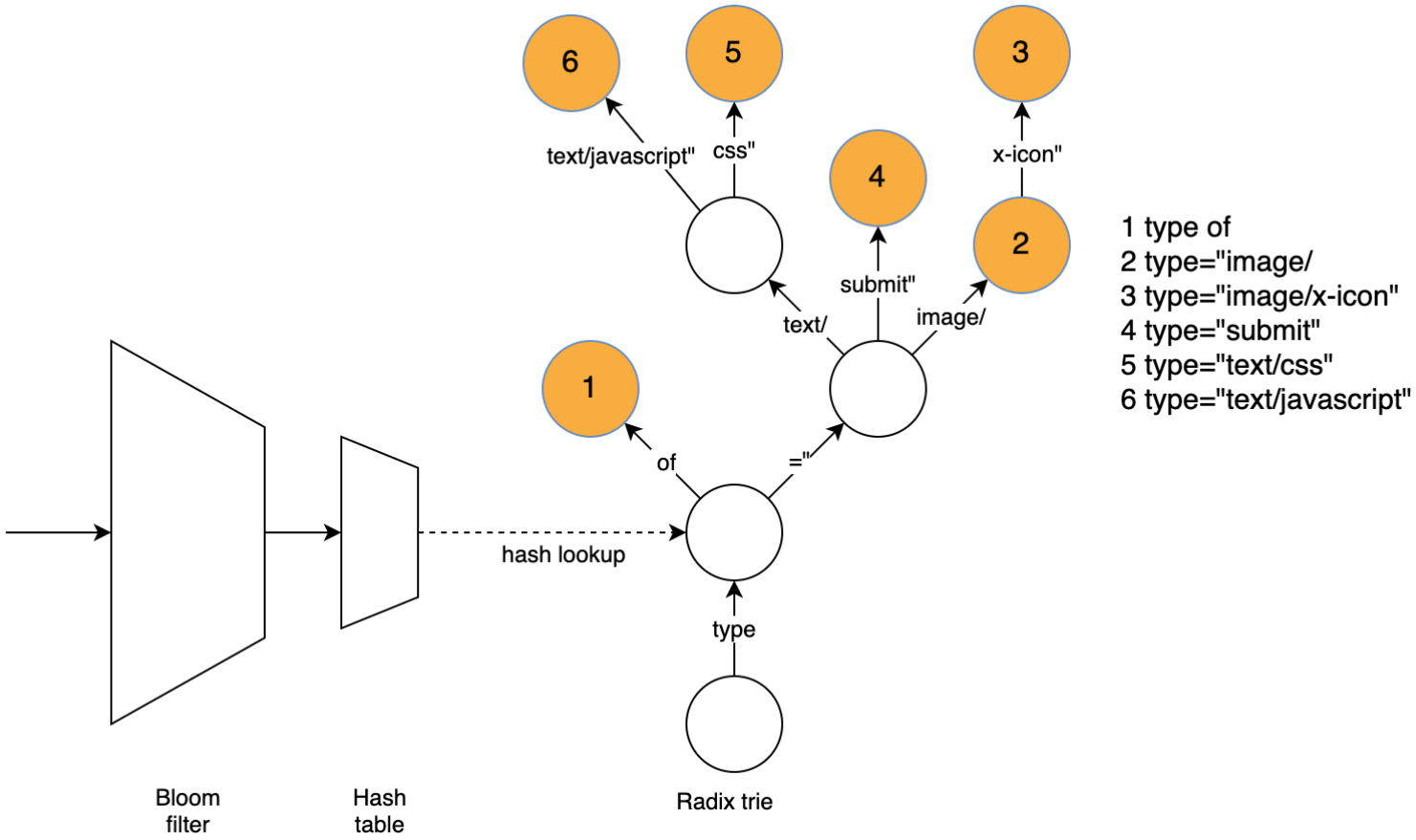
output tuple: (offset, length, symbol)



12 characters compressed into 6 tuples

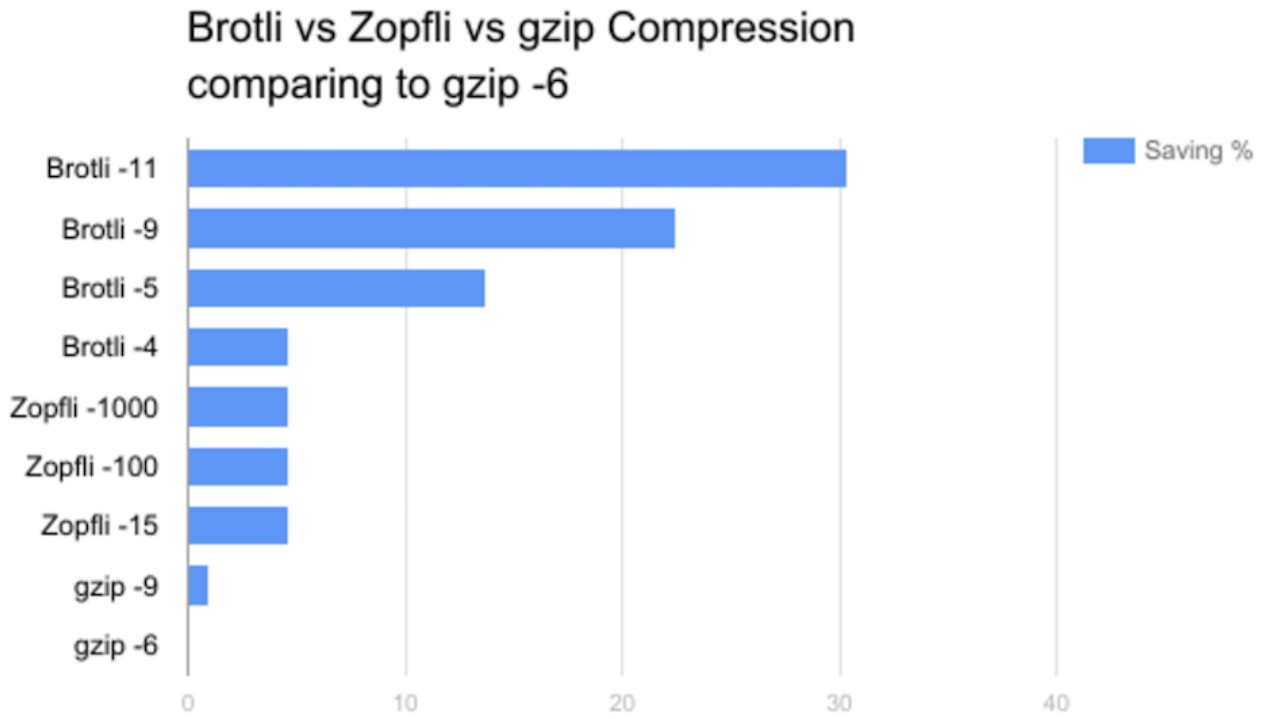
Compression rate: (12*8)/(6*(5+2+3))=96/60=1,6=60%.

brotli静态字典


6
5
3
text/javascript"
css"
x-icon"
4
2
submit"
text/
image/
1
of
="
hash lookup
type
Bloom filter
Hash table
Radix trie
1 type of
2 type="image/
3 type="image/x-icon"
4 type="submit"
5 type="text/css"
6 type="text/javascript"

ngx_brotli模块

- https://github.com/google/ngx_brotli



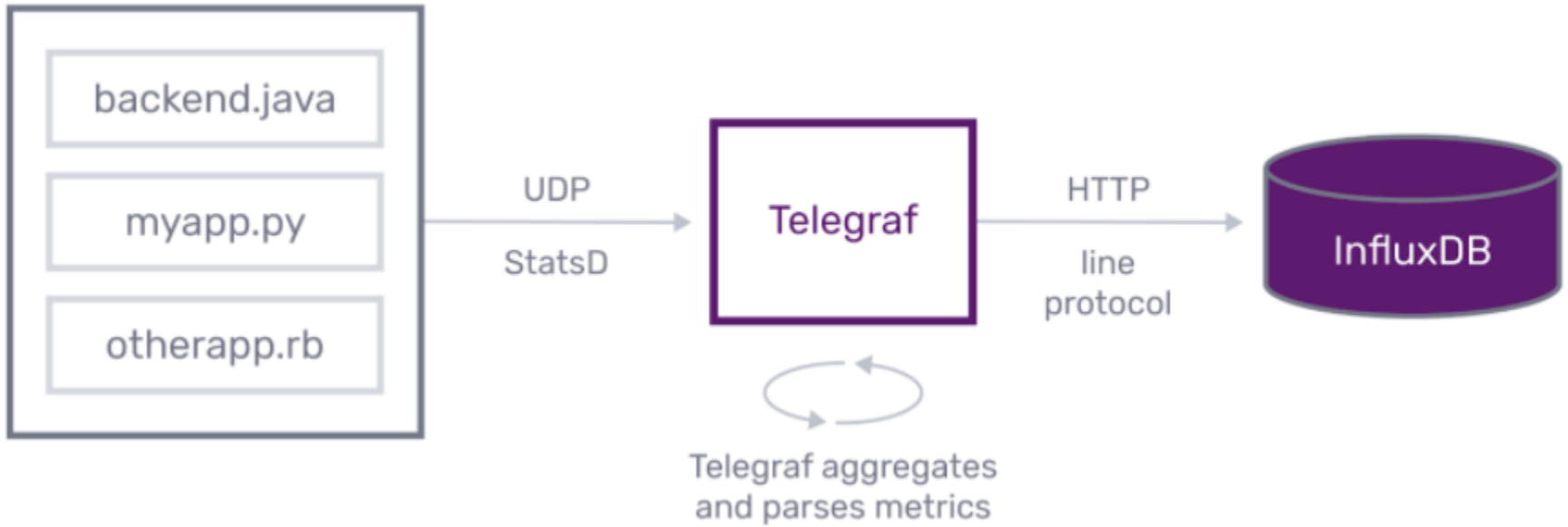
没有Annotation/Configmap指令的模块

- ngx_http_substitutions_filter_module过滤模块
  - https://github.com/yaoweibin/ngx_http_substitutions_filter_module

Nginx模块钩子

HTTP请求处理阶段	子阶段	模块举例	指令
openssl		lua_nginx_module	ssl_certificated_by_lua
rewrite		ngx_http_opentracing_module	enable-opentracing: "true"
preaccess		ngx_http_limit_req_module ngx_http_limit_conn_module	limit-rps “100” limit-connections “5”
access		ngx_http_auth_digest_module ngx_http_auth_basic_module ngx_http_modsecurity_module	auth-type: “digest” auth-url: “url” modsecurity
content		nginx_ajp_module	ajp_pass tomcats;
	upstream	lua-upstream-nginx-module	balance_by_lua
	filter	ngx_http_brotli_filter_module	enable-brotli “true”
log		nginx-influxdb-module	enable-influxdb “true”

InfluxDB时序数据库



nginx-influxdb-module默认上报数据

Metric	Type	Description
method	string	The HTTP request method that has been given as a reply to the caller
status	integer	The HTTP status code of the reply from the server (refer to RFC 7231 for more details)
bytes_sent	integer	The number of bytes sent to a client body + header
body_bytes_sent	integer	The number of bytes sent to a client only for body
header_bytes_sent	integer	The number of bytes sent to a client for header and body
request_length	integer	Request length (including request line, header, and request body)
uri	string	The called uri (e.g: /index.html)
extension	string	The extension of the served file (e.g: js, html, php, png)
content_type	string	The content type of the response (e.g: text/html)
request_time	string	Request processing time in seconds with a milliseconds resolution

influxdb上报数据

- Annotation
  - enable-influxdb
  - influxdb-measurement string
  - influxdb-port string
  - influxdb-host string
  - influxdb-server-name string
- 模块nginx-influxdb-module
  - https://github.com/influxdata/nginx-influxdb-module
  - 指令
    - influxdb
      - server_name
      - host
      - port
      - measurement
      - enabled
    - influxdb_dynamic_fields

K8S官方Nginx默认开启了哪些模块?

- Nginx官方默认未编译模块
- --with-http_ssl_module \
- --with-http_stub_status_module \
- --with-http_realip_module \
- --with-http_auth_request_module \
- --with-http_addition_module \
- --with-http_geoip_module \
- --with-http_gzip_static_module \
- --with-http_sub_module \
- --with-http_v2_module \
- --with-stream \
- --with-stream_ssl_module \
- --with-stream_realip_module \
- --with-stream_ssl_preread_module \
- --with-threads \
- --with-http_secure_link_module \
- --with-http_gunzip_module"

- 第三方C模块
  - nginx-influxdb-module
  - ngx_http_geoip2_module
  - nginx_ajp_module
  - ModSecurity-nginx
  - msgpack-c
  - ngx_devel_kit
  - set-misc-nginx-module
  - headers-more-nginx-module
  - nginx-http-auth-digest
  - ngx_http_substitutions_filter_module
  - nginx-opentracing
  - lua-nginx-module
  - stream-lua-nginx-module
  - lua-upstream-nginx-module
  - ngx_brotli

K8S官方Nginx开启了哪些模块?

- Lua模块
  - lua-resty-upload
  - lua-resty-string
  - lua-resty-balancer
  - lua-resty-core
  - lua-cjson
  - lua-resty-cookie
  - lua-resty-lrucache
  - lua-resty-lock
  - lua-resty-dns
  - lua-resty-http
  - lua-resty-http
  - lua-resty-redis
  - lua-resty-ipmatcher
  - lua-resty-global-throttle

Nginx Controller

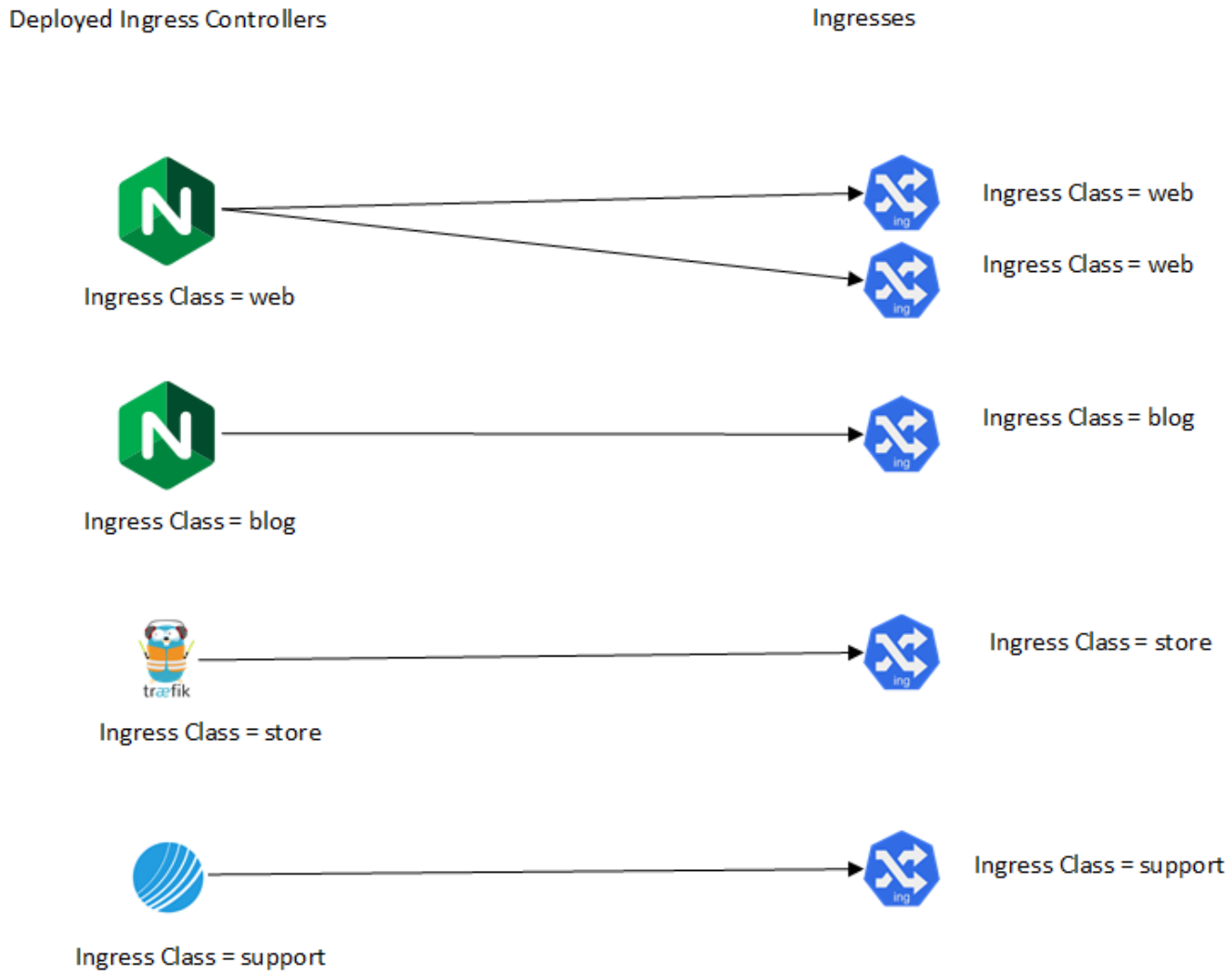
- **k8s官方**
  - https://github.com/kubernetes/ingress-nginx
- **Nginx官方**
  - https://github.com/nginxinc/kubernetes-ingress
- **差别**
  - https://github.com/nginxinc/kubernetes-ingress/blob/master/docs/nginx-ingress-controllers.md

进程架构

- **K8S社区官方ingress controller**
  - /usr/bin/dumb-init
  - /nginx-ingress-controller
  - /usr/local/nginx/sbin/nginx -c /etc/nginx/nginx.conf
- **NGINX官方ingress controller**
  - /nginx-ingress
  - /usr/sbin/nginx

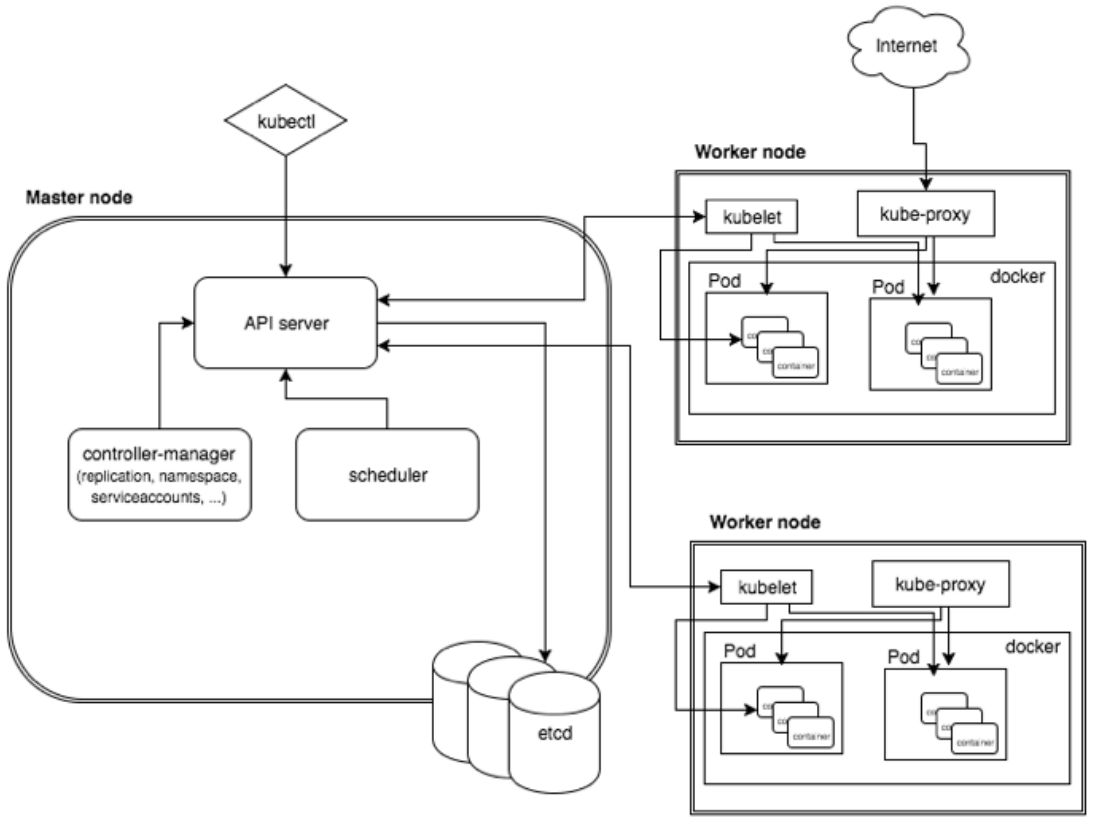
IngressClass

- IngressClass
  - metadata
    - name: blog
- Ingress
  - spec
    - ingressClassName: blog
- Controller
  - template
    - spec
      - containers
        - args
          - --ingress-class=blog



安装Nginx Ingress

- **创建Nginx Controller**
  - 创建namespace与帐号
  - 创建角色并绑定帐号
  - 创建default server的密钥
  - 创建存放nginx.conf的ConfigMap
  - 创建ingress-class
  - 创建Nginx Controller POD
- **暴露Nginx Controller服务**
- **创建Ingress规则**
  - Host：精确与通配符匹配
  - Path：前缀或者精确匹配
  - Backend



示例：安装Nginx官方Ingress Controller

- https://github.com/resouer/kubernetes-ingress
- 创建namespace与帐号
  - kubectl apply -f common/ns-and-sa.yaml
- 创建角色并绑定帐号
  - kubectl apply -f rbac/rbac.yaml
- 创建default server的密钥
  - kubectl apply -f common/default-server-secret.yaml
- 创建存放nginx.conf的ConfigMap
  - kubectl apply -f common/nginx-config.yaml
- 创建ingress-class
  - kubectl apply -f common/ingress-class.yaml
- 以deployment或者daemon方式创建Nginx Controller POD
  - kubectl apply -f deployment/nginx-ingress.yaml
  - 验证/stub_status：kubectl port-forward nginx-pod 8080:8080 --namespace=nginx-ingress
    - curl http://localhost:8080/stub_status

VirtualServer与 VirtualServerRoute

```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: cafe-ingress
spec:
  tls:
  - hosts:
    - cafe.example.com
    secretName: cafe-secret
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /tea
        backend:
          serviceName: tea-svc
          servicePort: 80
      - path: /coffee
        backend:
          serviceName: coffee-svc
          servicePort: 80
```

```yaml
apiVersion: k8s.nginx.org/v1
kind: VirtualServerRoute
metadata:
  name: coffee
  namespace: coffee-ns
spec:
  host: cafe.example.com
  upstreams:
  - name: latte
    service: latte-svc
    port: 80
  - name: espresso
    service: espresso-svc
    port: 80
  subroutes:
  - path: /coffee/latte
    action:
      pass: latte
  - path: /coffee/espresso
    action:
      pass: espresso
```

```yaml
apiVersion: k8s.nginx.org/v1
kind: VirtualServer
metadata:
  name: cafe
spec:
  host: cafe.example.com
  tls:
    secret: cafe-secret
  upstreams:
  - name: tea
    service: tea-svc
    port: 80
  - name: coffee
    service: coffee-svc
    port: 80
  routes:
  - path: /tea
    action:
      pass: tea
  - path: /coffee
    action:
      pass: coffee
  - path: ~ ^/decaf/.*\\.jpg$
    action:
      pass: coffee
  - path: = /green/tea
    action:
      pass: tea
```

- nginx.org/location-snippets
- nginx.org/server-snippets

```
server {
    listen 80;


    location / {
        return 302 /coffee;
    }


    location /coffee {
        proxy_http_version 1.1;


        add_header my-test-header test-value;
        ...
        proxy_pass http://default-cafe-ingress-
    }

    location /tea {
        proxy_http_version 1.1;

        add_header my-test-header test-value;
        ...
        proxy_pass http://default-cafe-ingress-
    }
}
```

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: cafe-ingress-with-snippets
  annotations:
    nginx.org/server-snippets: |
      location / {
          return 302 /coffee;
      }
    nginx.org/location-snippets: |
      add_header my-test-header test-value;
spec:
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /tea
        backend:
          serviceName: tea-svc
          servicePort: 80
      - path: /coffee
        backend:
          serviceName: coffee-svc
          servicePort: 80
```

自定义模板

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: nginx-config
  namespace: nginx-ingress
data:
  main-template: |
    worker_processes  {{.WorkerProcesses}};
    ...
        include /etc/nginx/conf.d/*.conf;
    }
  ingress-template: |
    {{range $upstream := .Upstreams}}
    upstream {{$upstream.Name}} {
      {{if $upstream.LBMethod }}{{$upstream.LBMethod}};{{end}}
    ...
    }{{end}}
  virtualserver-template: |
    {{ range $u := .Upstreams }}
    upstream {{ $u.Name }} {
      {{ if ne $u.UpstreamZoneSize "0" }}zone {{ $u.Name }} {{ $u.UpstreamZoneSize }};{{ end }}
    ...
    }
    {{ end }}
```

Custom Annotations

```
# This is the configuration for {{$.Ingress.Name}}/{{$.Ingress.Namespace}}

{{if index $.Ingress.Annotations "custom.nginx.org/feature-a"}}
# Insert config for feature A if the annotation is set
{{end}}

{{with $value := index $.Ingress.Annotations "custom.nginx.org/feature-b"}}
# Insert config for feature B if the annotation is set
# Print the value assigned to the annotation: {{$value}}
{{end}}
```

+

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: example-ingress
  namespace: production
  annotations:
    custom.nginx.org/feature-a: "on"
    custom.nginx.org/feature-b: "512"
spec:
  rules:
  - host: example.com
```

=

```
# This is the configuration for cafe-ingress/default

# Insert config for feature A if the annotation is set



# Insert config for feature B if the annotation is set
# Print the value assigned to the annotation: 512
```

谢谢