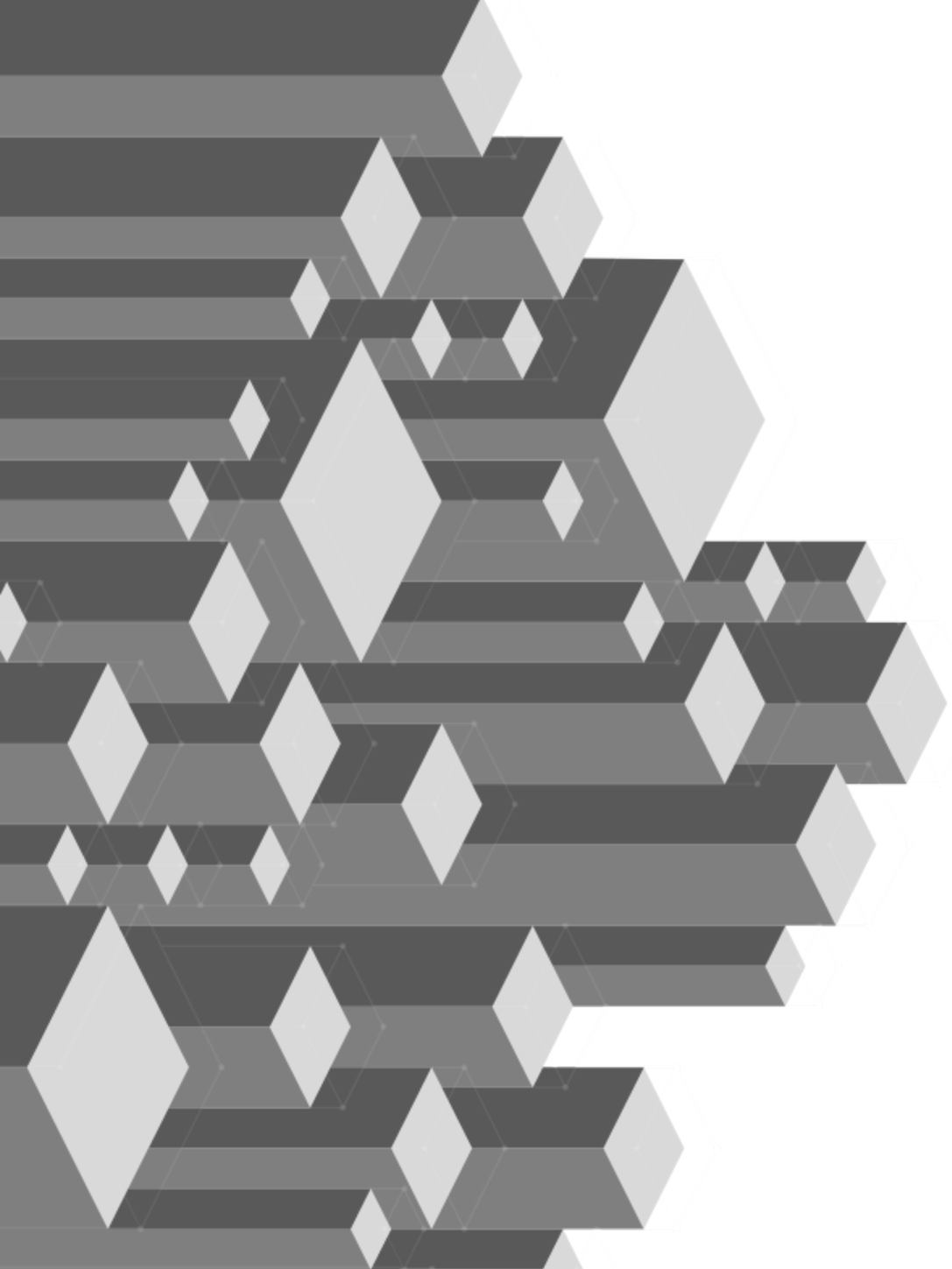


KONG

Microservices & API Gateways

Marco Palladino

I am Marco Palladino

Co-Founder and CTO at mashape.com

Core committer at github.com/Mashape/kong

Originally from Milan (Italy), moved to San Francisco to start Mashape

Mashape

The company behind Kong with HQ in SF and offices in Toronto

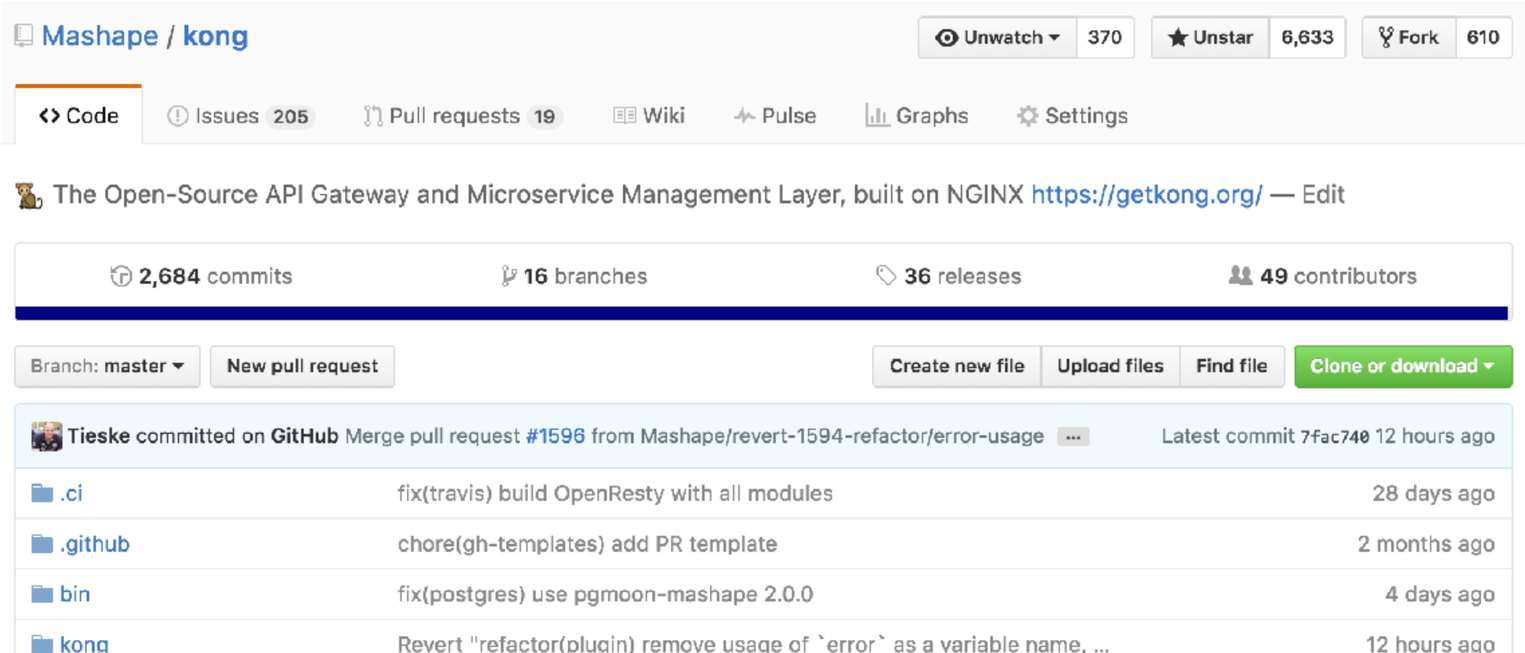
Six years of API expertise in open-source and Enterprise environments

Started as an API Marketplace, Mashape is now leading the API Gateway revolution in OSS and in Top Fortune 500 deployments with its Kong Enterprise offering

What is Kong?

Kong is an open-source management layer for APIs to secure, manage and extend APIs and Microservices.

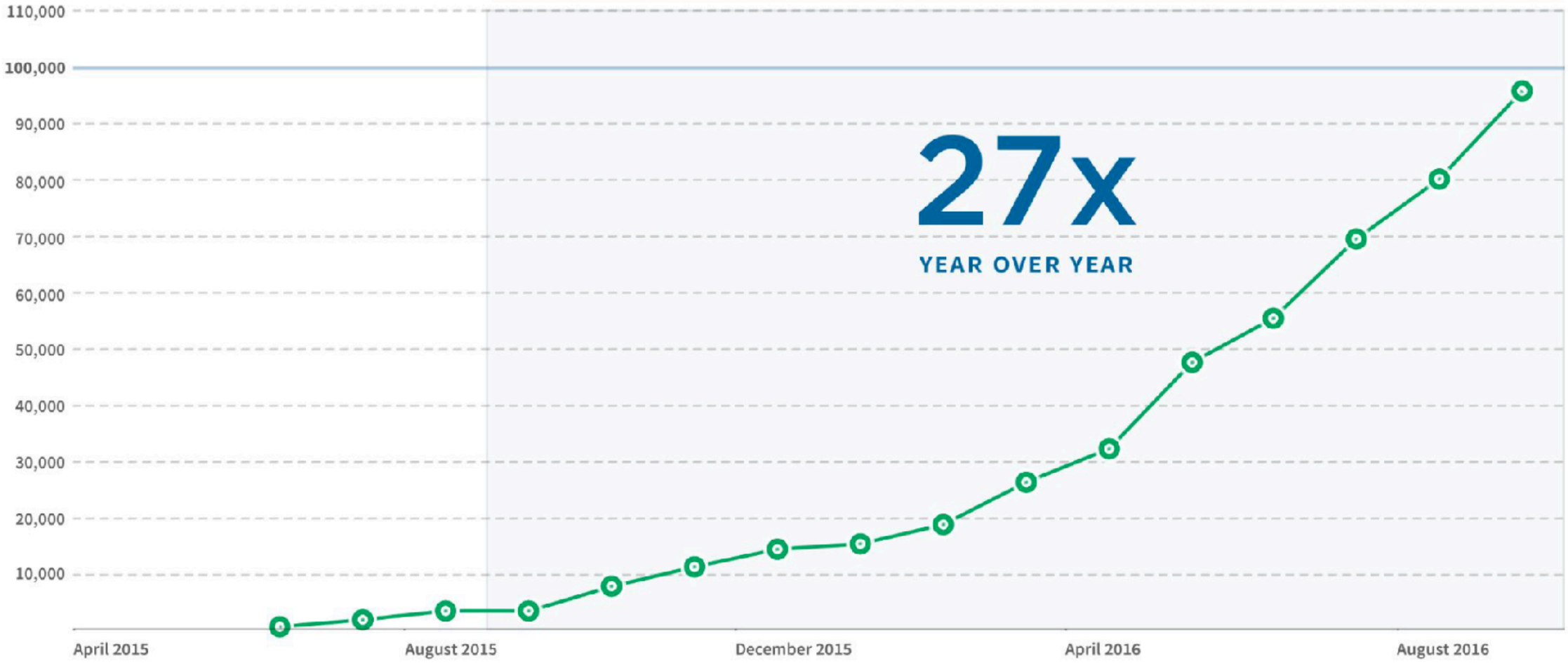
https://getkong.org



Kong Product

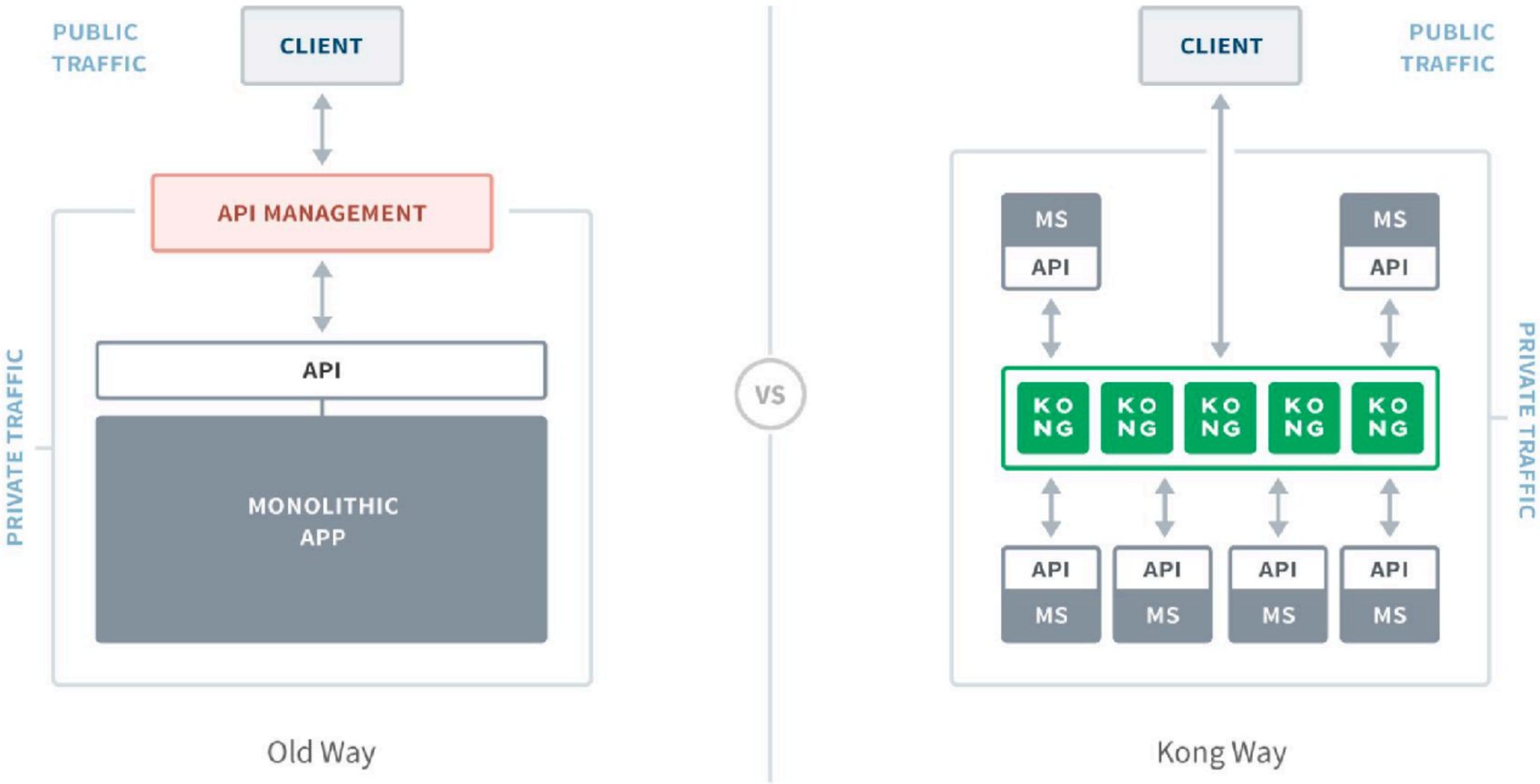
- **Born in 2015** from the API Marketplace
- Open Source
- Built on top of **OpenResty**
- Extensible (**+25 Plugins**)
- Platform Agnostic
- RESTful API
- Up and Running in 10 Minutes
- Fast and Scalable

Global Adoption (1M Downloads)

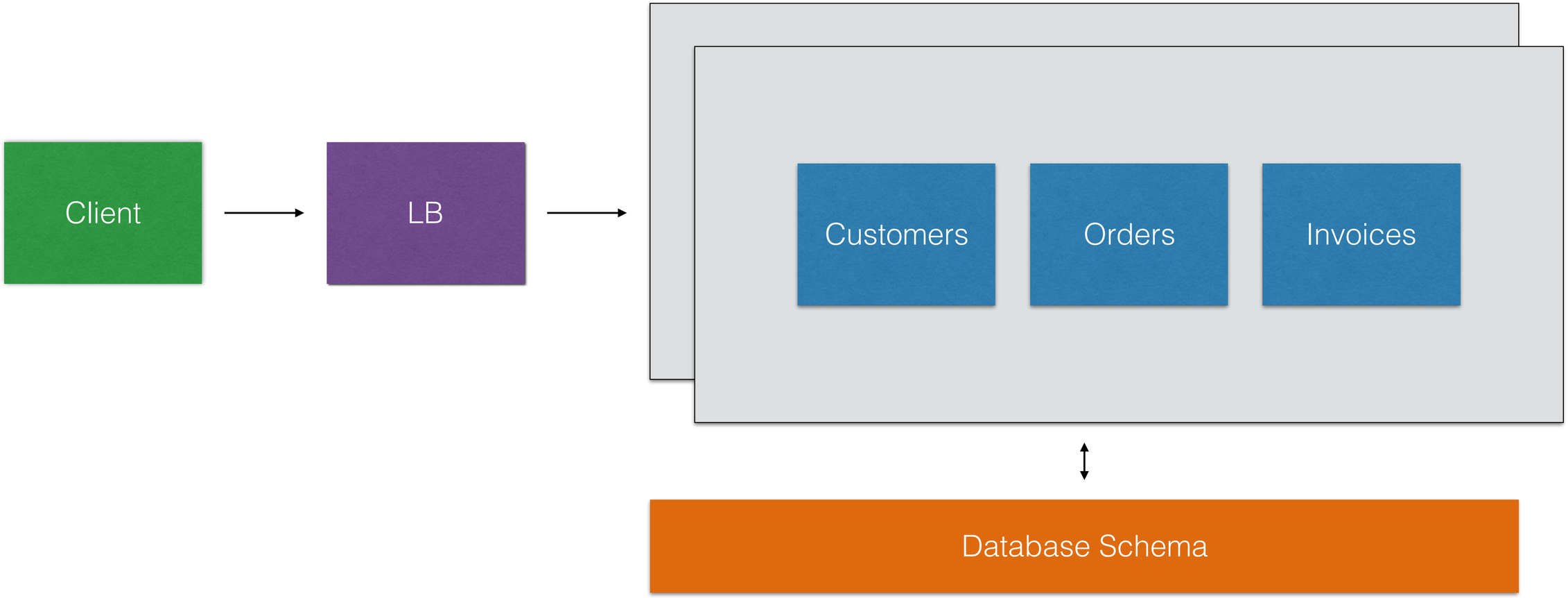


(Unique instances running for more than 24 hours, not cumulative)

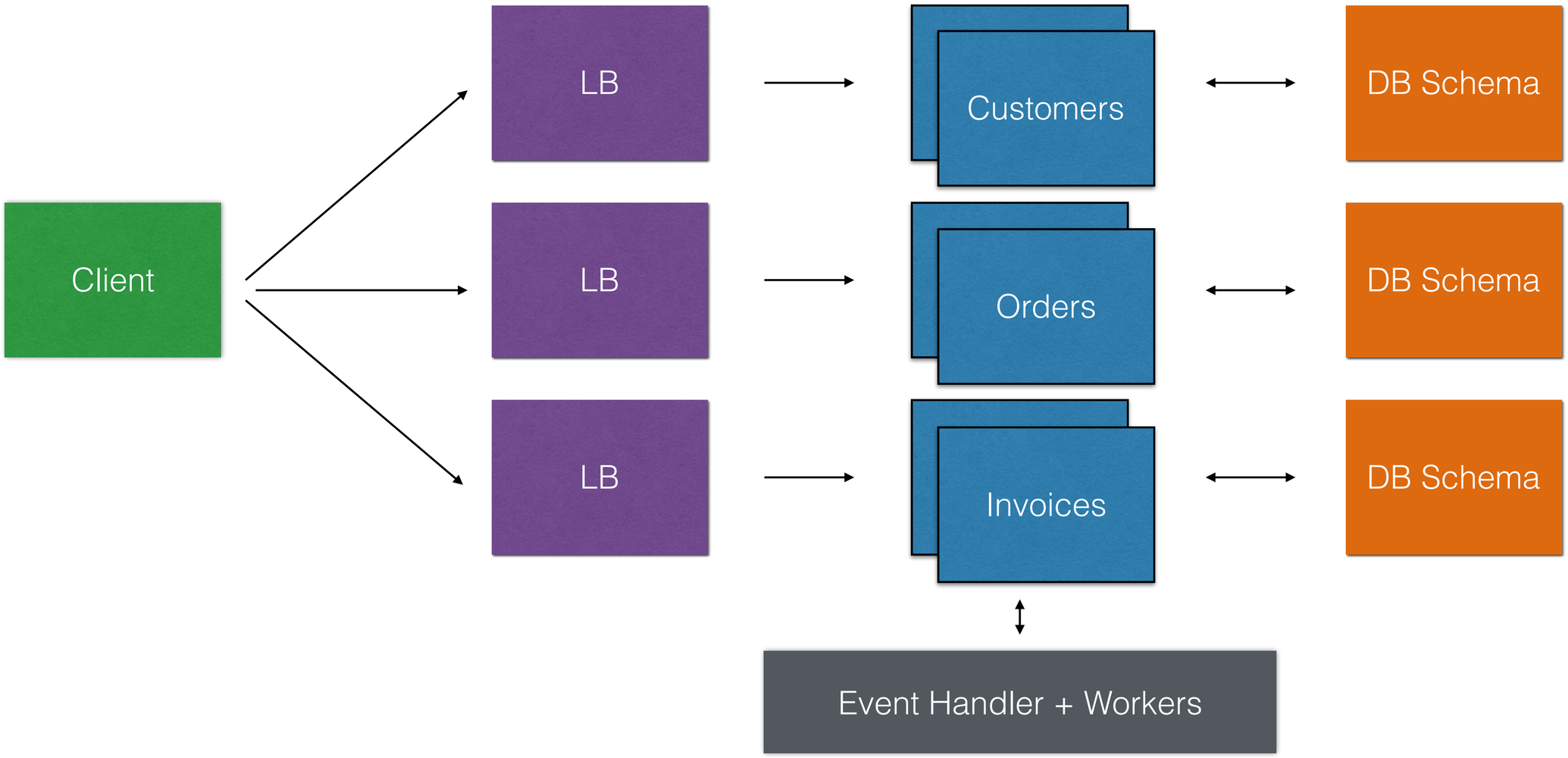
Old vs New


PUBLIC TRAFFIC
CLIENT
API MANAGEMENT
API
MONOLITHIC APP
PRIVATE TRAFFIC
Old Way
VS
CLIENT
PUBLIC TRAFFIC
MS
API
MS
API
KO NG
KO NG
KO NG
KO NG
KO NG
API
MS
API
MS
API
MS
API
MS
PRIVATE TRAFFIC
Kong Way

Monolithic Architecture



Microservice-oriented architecture



Monolithic Application Pros/Cons

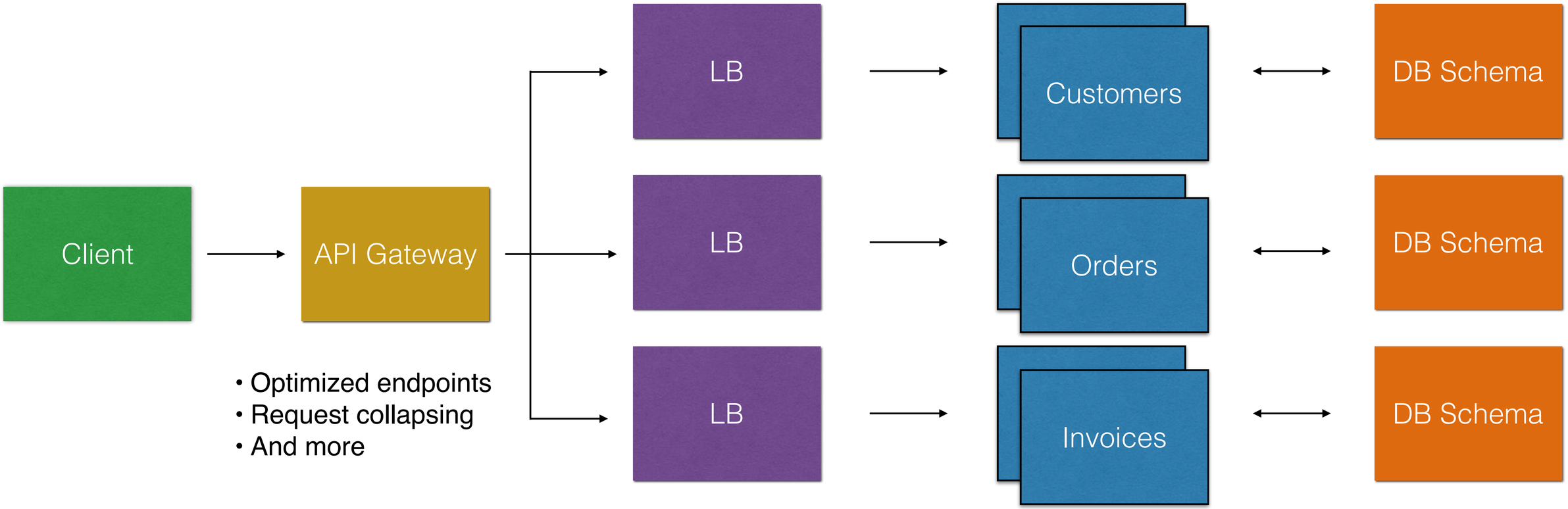
Simplicity, for small codebases	Faster early development speed	Easy testing	IDE support
Not ideal for growing codebases	Slower iterations in the long term	Harder to innovate	Steep code learning curve

Microservice-oriented Application Pros/Cons

Better architecture for large applications	Better agility in the long term	Microservices: easy to learn	Isolation for scalability and damage control
More moving parts	Complex infrastructure requirements	Consistency and availability	Harder to test

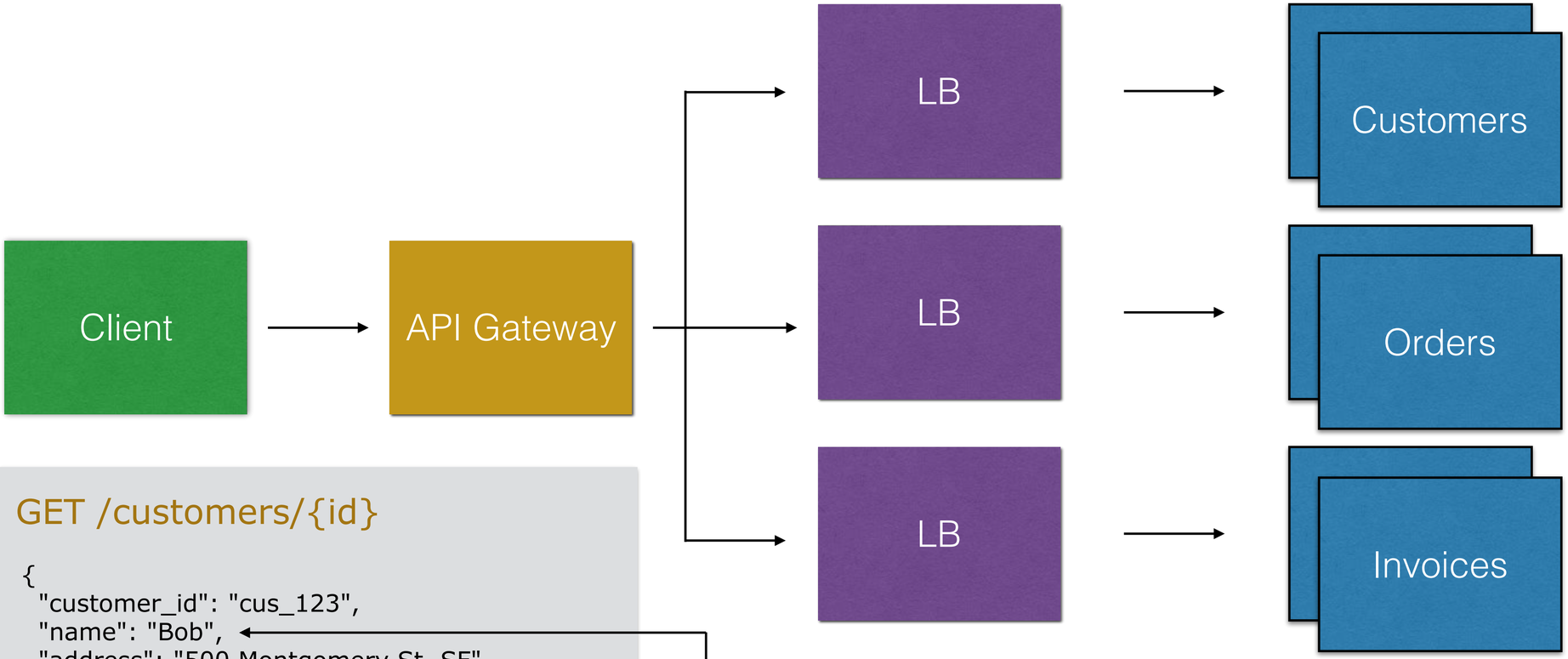
Why an API Gateway?

API Gateway Pattern



- Optimized endpoints
- Request collapsing
- And more

Optimized Endpoints




```
{
  "id": "cus_123",
  "customer_name": "Bob",
  "address": "500 Montgomery St, SF"
}
```

```
{
  "id": "order_123",
  "customer_id": "cus_123",
  "item_name": "Vacuum Cleaner"
}
```

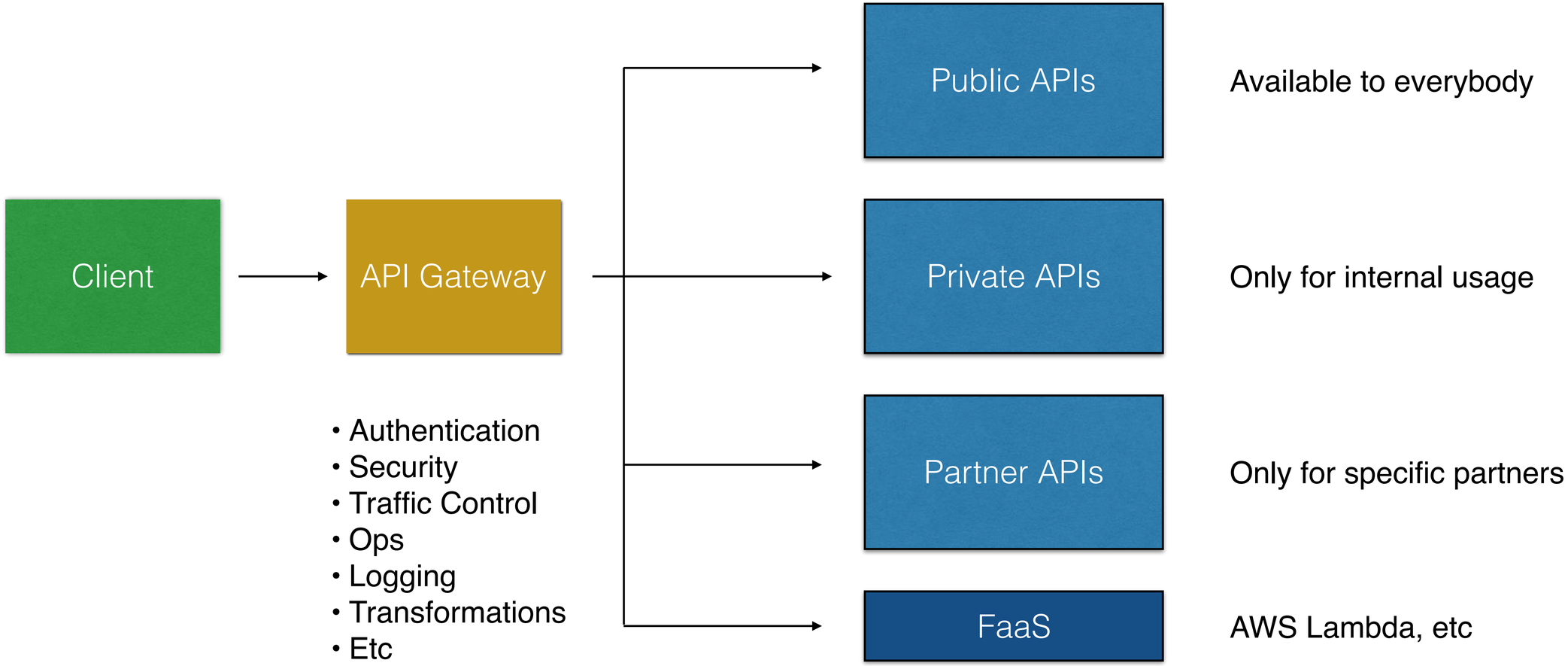
```
{
  "order_id": "order_123",
  "price": "99.99"
}
```

GET /customers/{id}

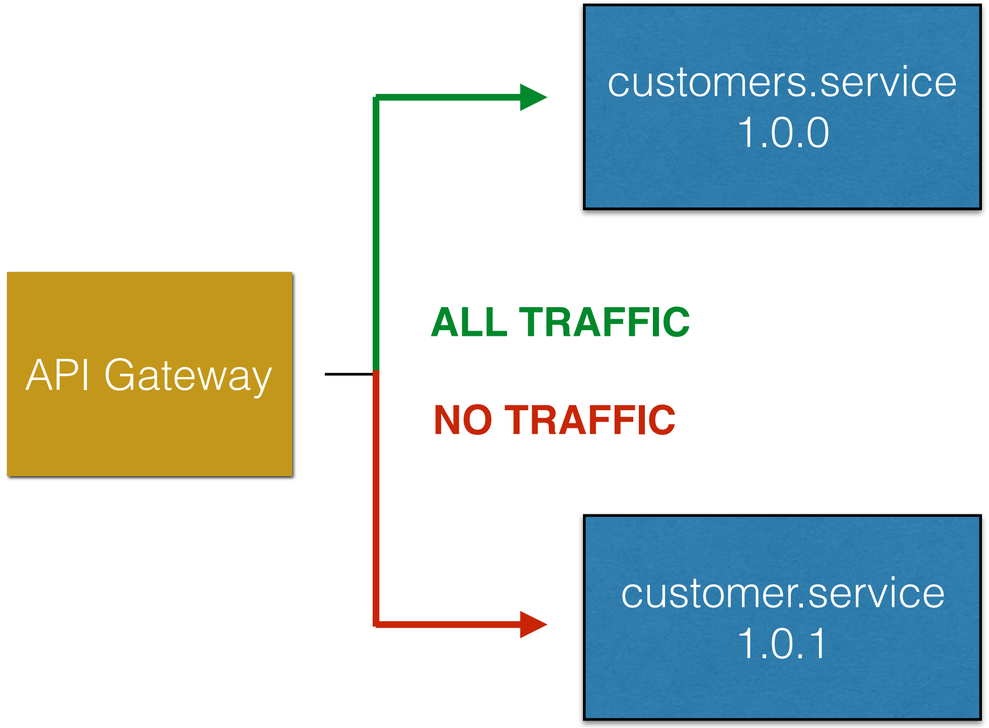
```
{
  "customer_id": "cus_123",
  "name": "Bob",
  "address": "500 Montgomery St, SF",
  "orders": […],
  "invoices": […],
}
```

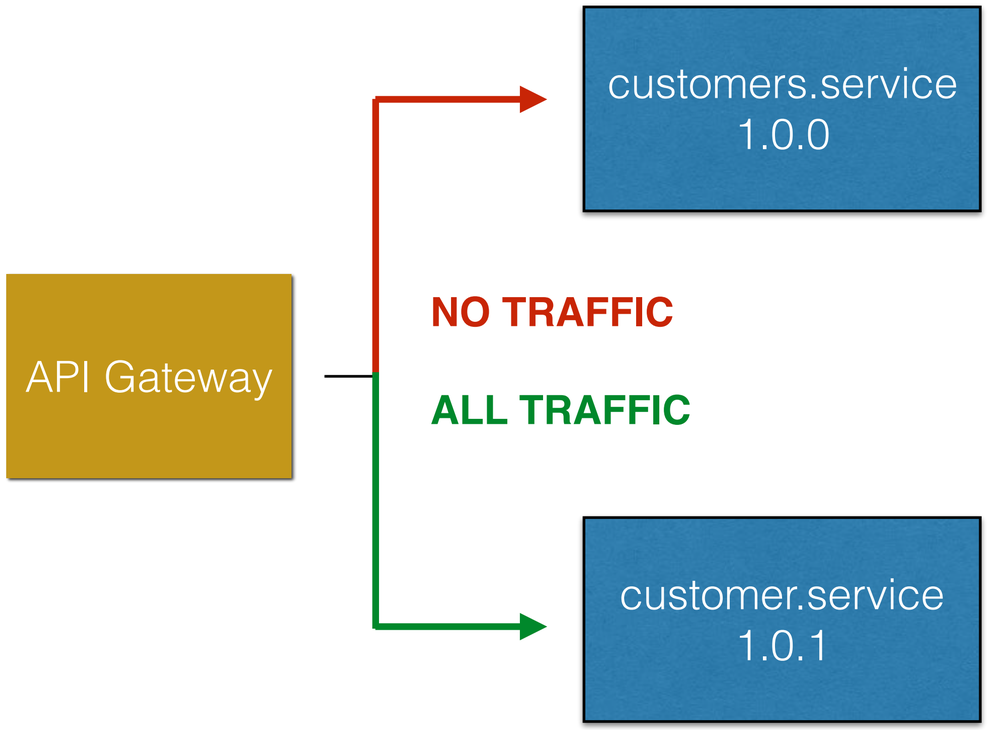
+ extra transformations

Centralized Middleware Functionality

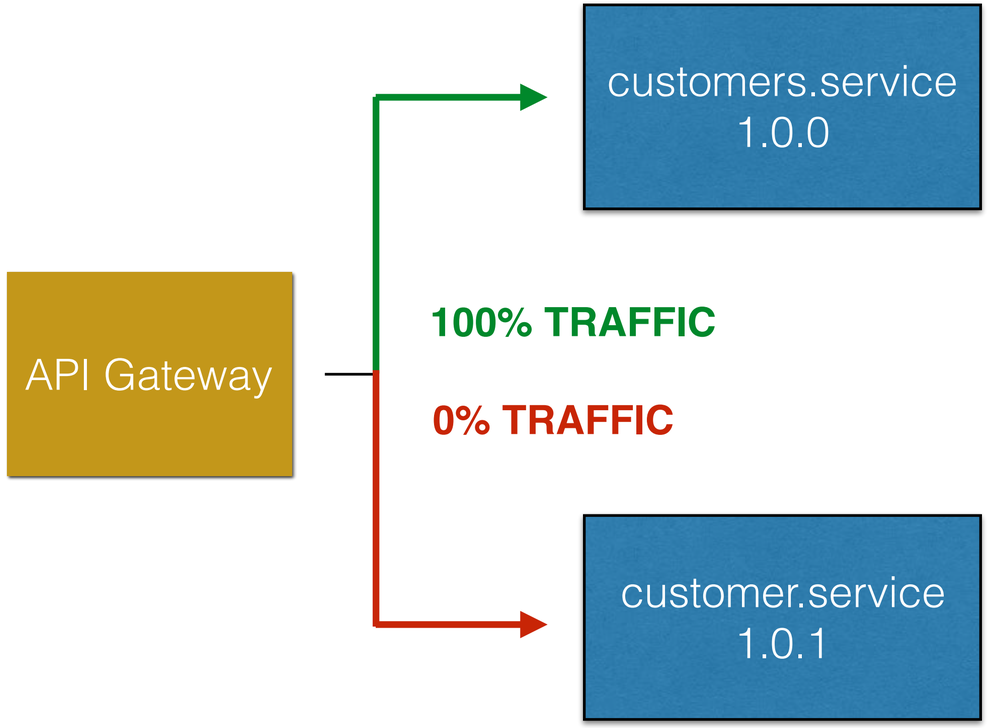


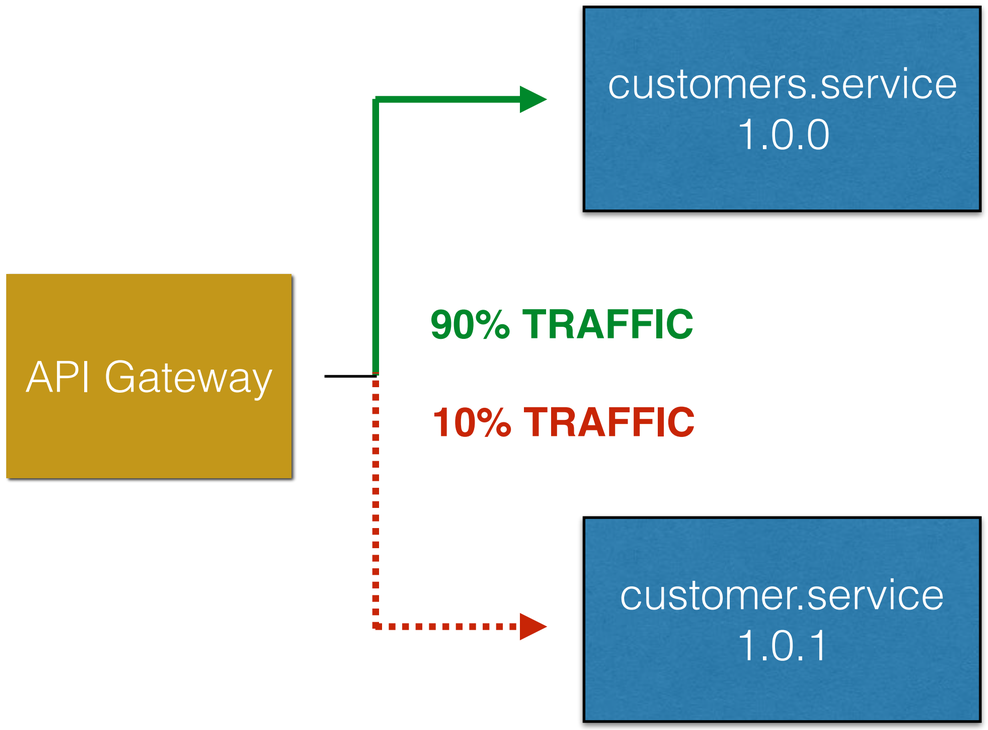
Ops: Blue/Green deployments



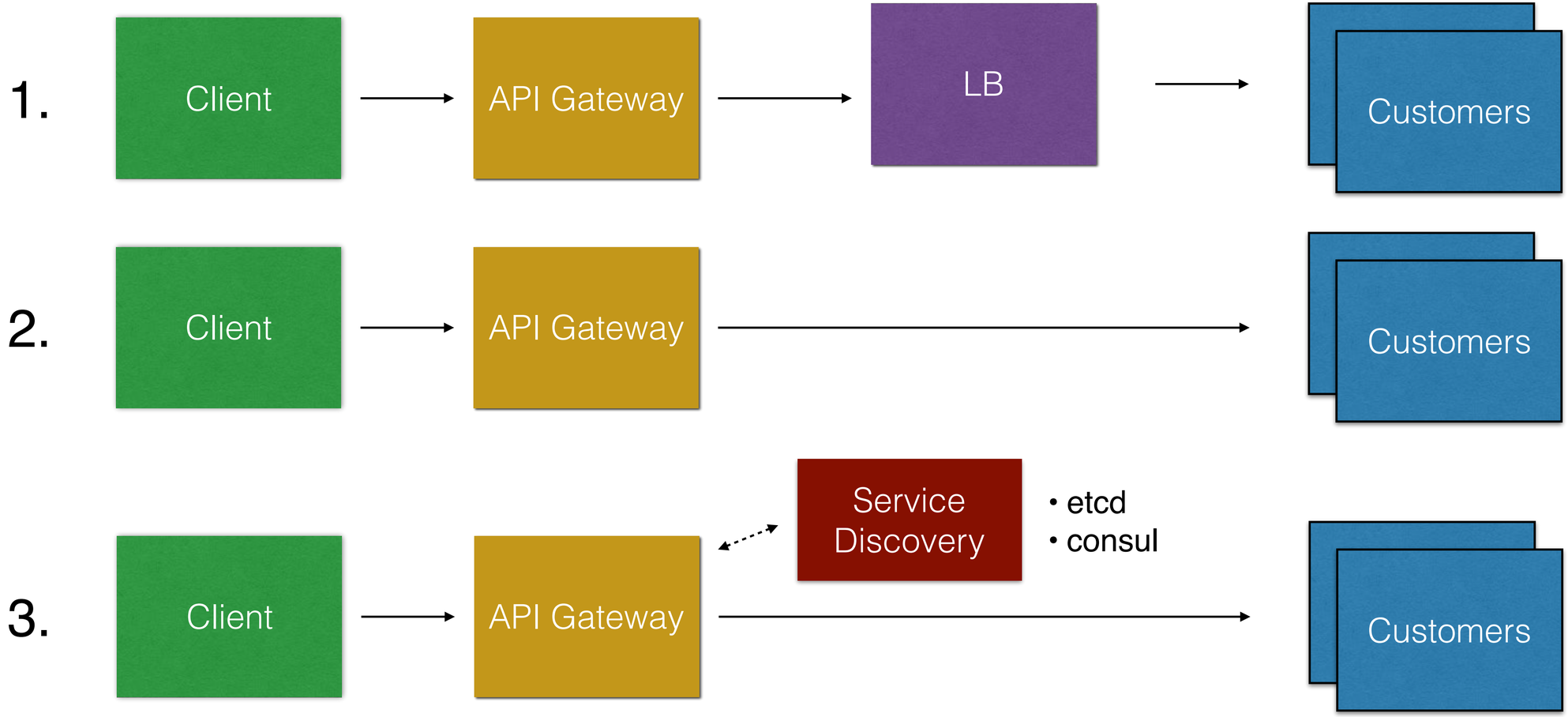


Ops: Canary Releases

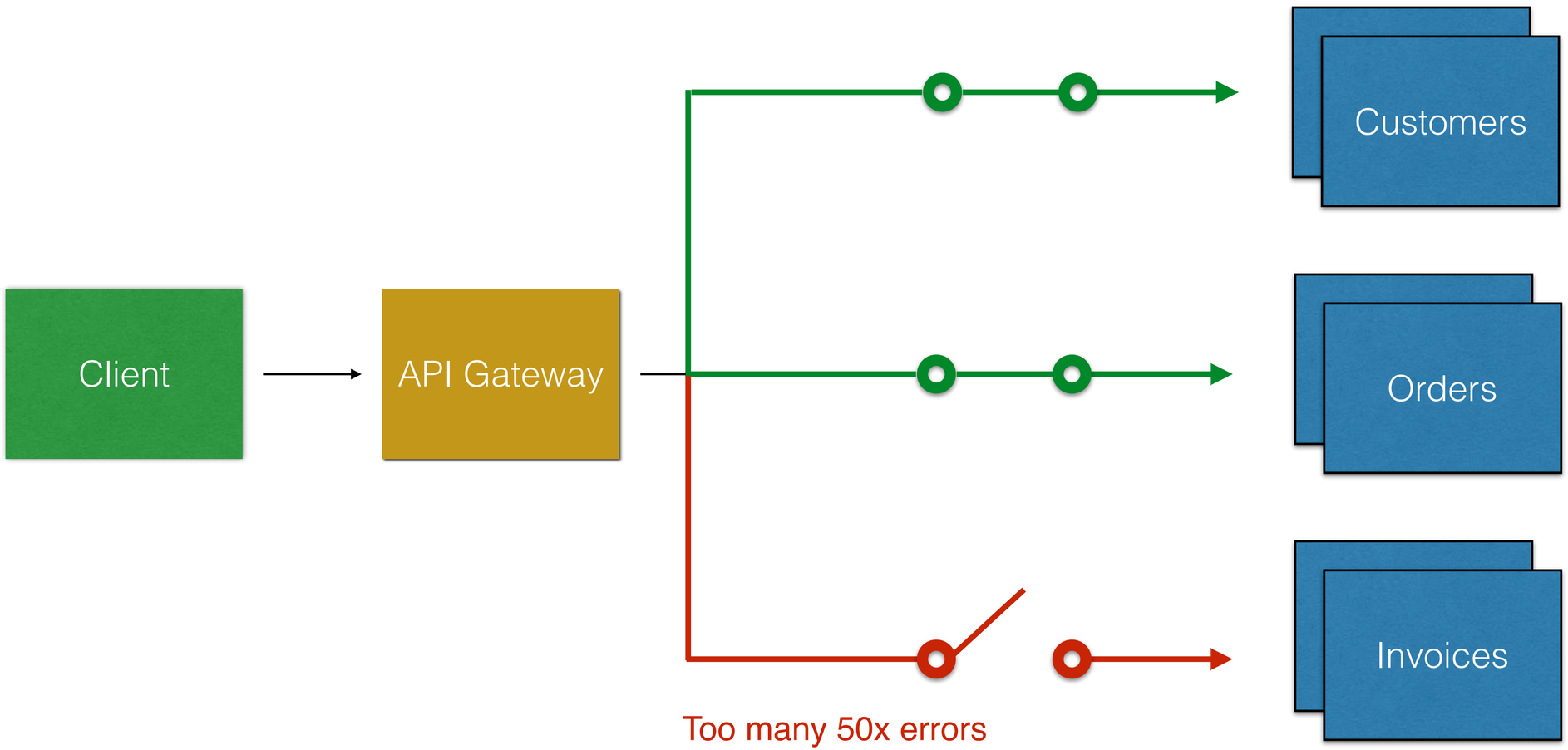




Ops: Load Balancing



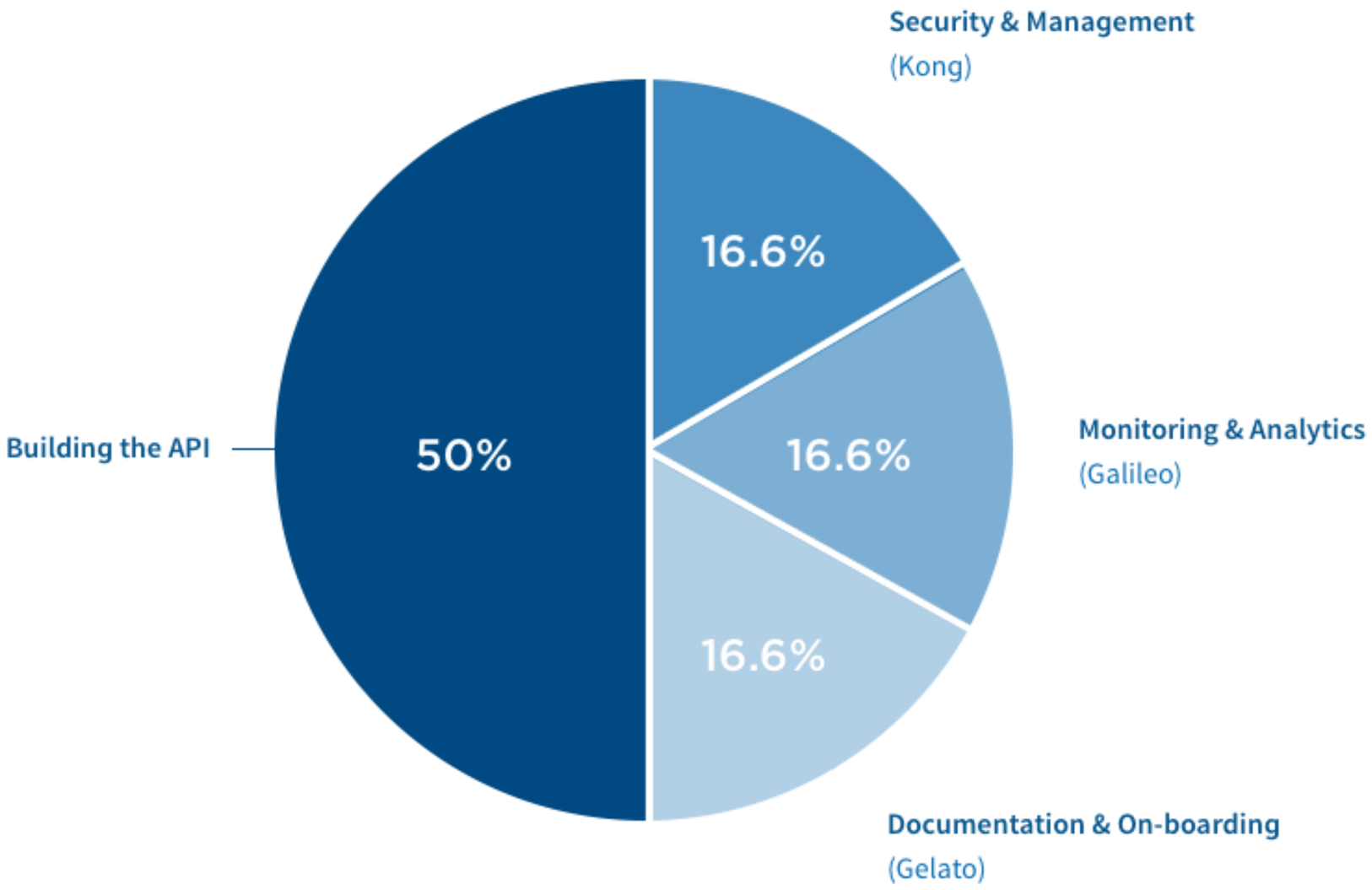
Ops: Circuit Breakers



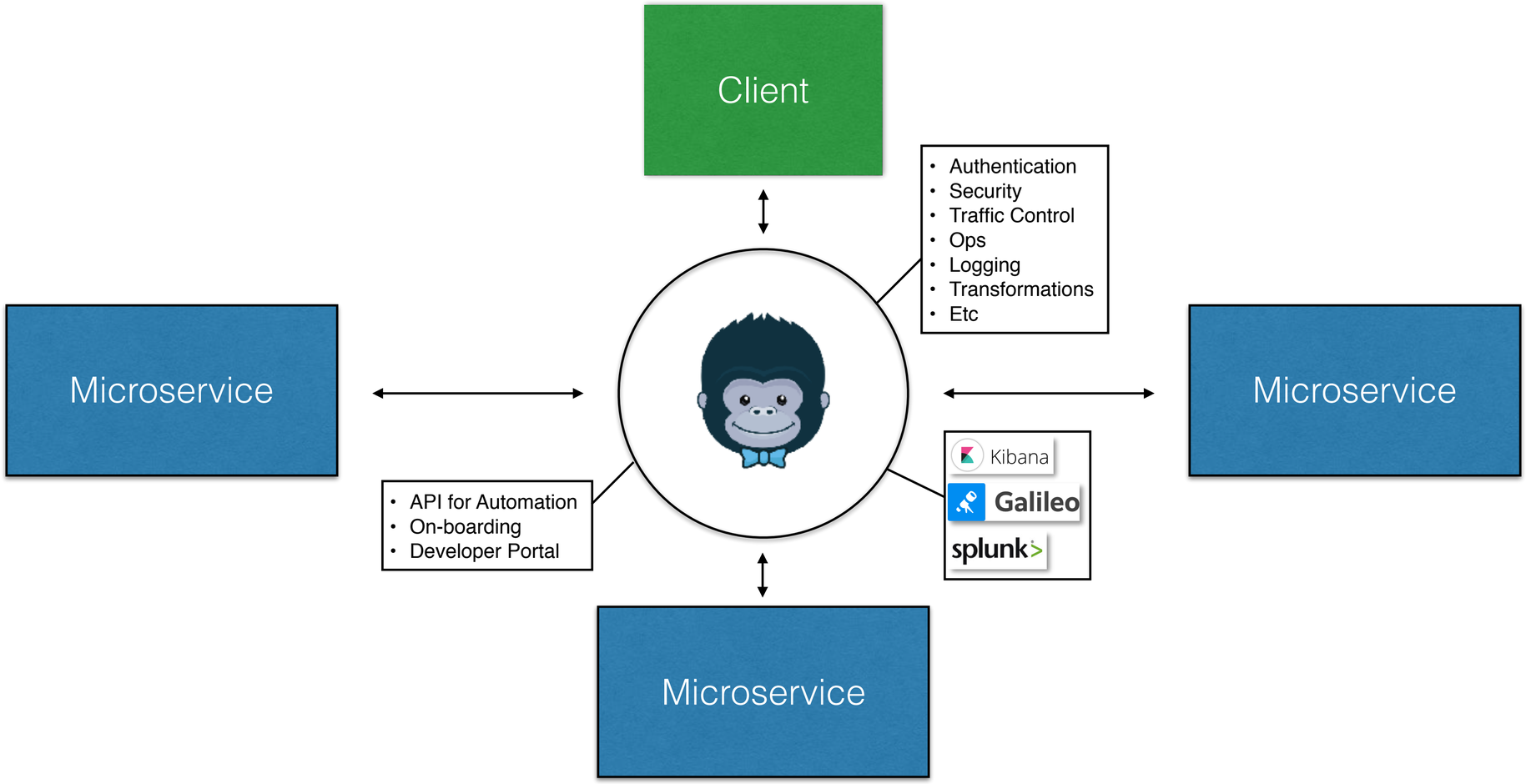
Building a microservice

!=

Running a microservice

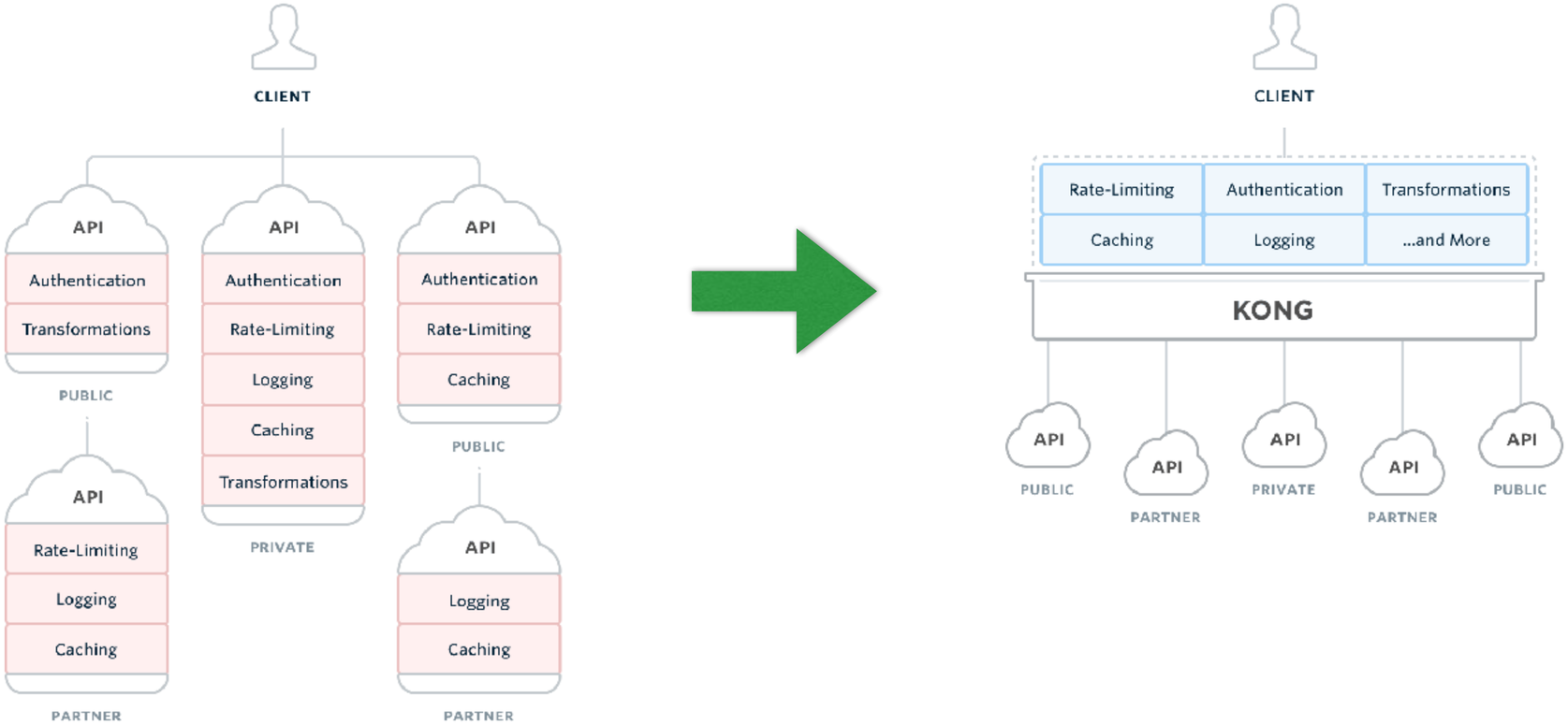

Security & Management
(Kong)
16.6%
Monitoring & Analytics
(Galileo)
16.6%
Building the API
50%
16.6%
Documentation & On-boarding
(Gelato)

API Gateways, and Kong, can help



Centralizing common functionality

Built on top of OpenResty, centralizes common middleware functionality:



Kong Plugins

Can be created from scratch & extended by the community.

Authentication

Protect your services with an authentication layer:

Basic Authentication
Add Basic Authentication to your APIs

Key Authentication
Add a key authentication to your APIs

OAuth 2.0 Authentication
Add an OAuth 2.0 authentication to your APIs

HMAC Authentication
Add HMAC Authentication to your APIs

JWT
Verify and authenticate JSON Web Tokens

LDAP Authentication
Integrate Kong with a LDAP server

Security

Protect your services with additional security layers:

ACL
Control which consumers can access APIs

CORS
Allow developers to make requests from the browser

Dynamic SSL
Add an SSL certificate for an underlying service

IP Restriction
Whitelist or blacklist IPs that can make requests

Bot Detection
Detects and blocks bots or custom clients

Traffic Control

Manage, throttle and restrict inbound and outbound API traffic:

Rate Limiting
Rate-limit how many HTTP requests a developer can make

Response Rate Limiting
Rate-Limiting based on a custom response header value

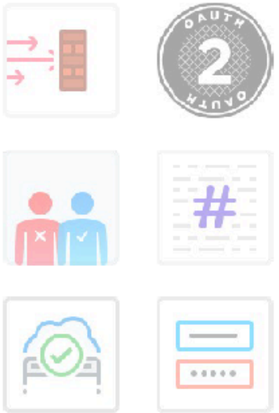
Request Size Limiting
Block requests with bodies greater than a specific size

Analytics & Monitoring

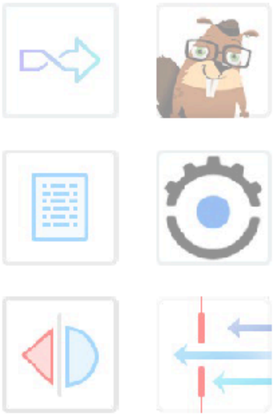
Visualize, inspect and monitor APIs and microservices traffic:

Plugin-Powered Architecture

Add powerful functionality to your services through RESTful Interface

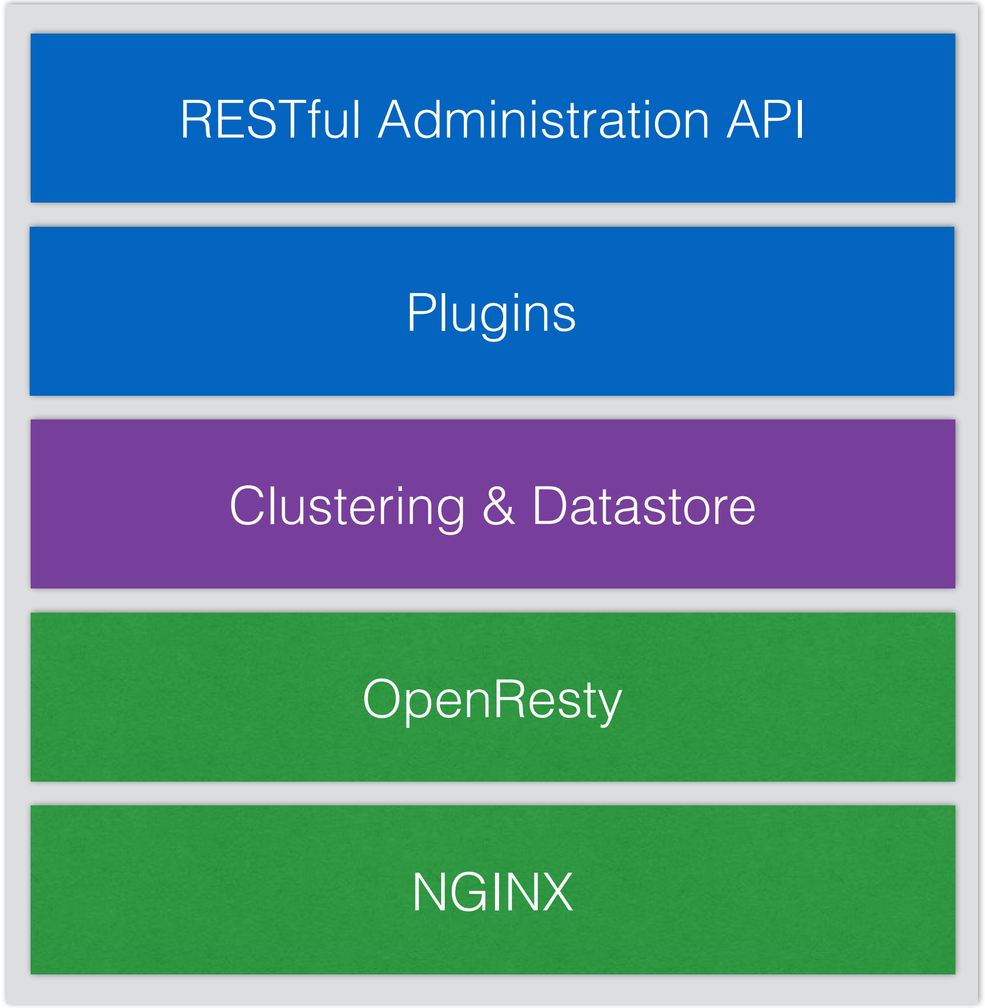


```
curl  -X POST http://kong:8001/apis/{id}/plugins/
      -d "name=rate-limiting"
      -d "config.second=10"
      -d "config.hour=50000"
```



Custom plugins can easily be created to address specific requirements like Enterprise Authentication, logging, Third-Party integrations and more.

Kong: OpenResty + NGINX



- JSON HTTP API
- Extendable by Plugins
- Can be integrated for automation

- Plugins created with LUA
- Intercept Request/Response lifecycle
- Can integrate with third-party services

- Either Cassandra or PostgreSQL
- Optionally Redis for some plugins
- Single or multi-DC clustering

- Underlying engine of Kong
- Provides hooks for Req/Res lifecycle
- Extends underlying NGINX

- The core dependency
- Handles low-level operations
- Solid foundation and known tech

NGINX Configuration

```
worker_processes auto;
daemon on;

pid pids/nginx.pid;
error_log logs/error.log notice;

worker_rlimit_nofile 4864;

events {
    worker_connections 4864;
    multi_accept on;
}

http {
    include 'nginx-kong.conf';
}
```

nginx.conf

```
init_by_lua_block {
    ..
}

init_worker_by_lua_block {
    ..
}

server {
    listen 0.0.0.0:8000;
    location / {
        access_by_lua_block {
            ..
        }

        header_filter_by_lua_block {
            ..
        }

        body_filter_by_lua_block {

            ..
        }

        log_by_lua_block {
            ..
        }
    }
}

..
```

nginx-kong.conf

Kong Entry-points

Proxy

```
$ curl 127.0.0.1:8000
```

```
$ curl 127.0.0.1:8443
```

Admin API

```
$ curl 127.0.0.1:8001
```

Core Entities

```
$ curl 127.0.0.1:8001/apis
```

```
$ curl 127.0.0.1:8001/consumers
```

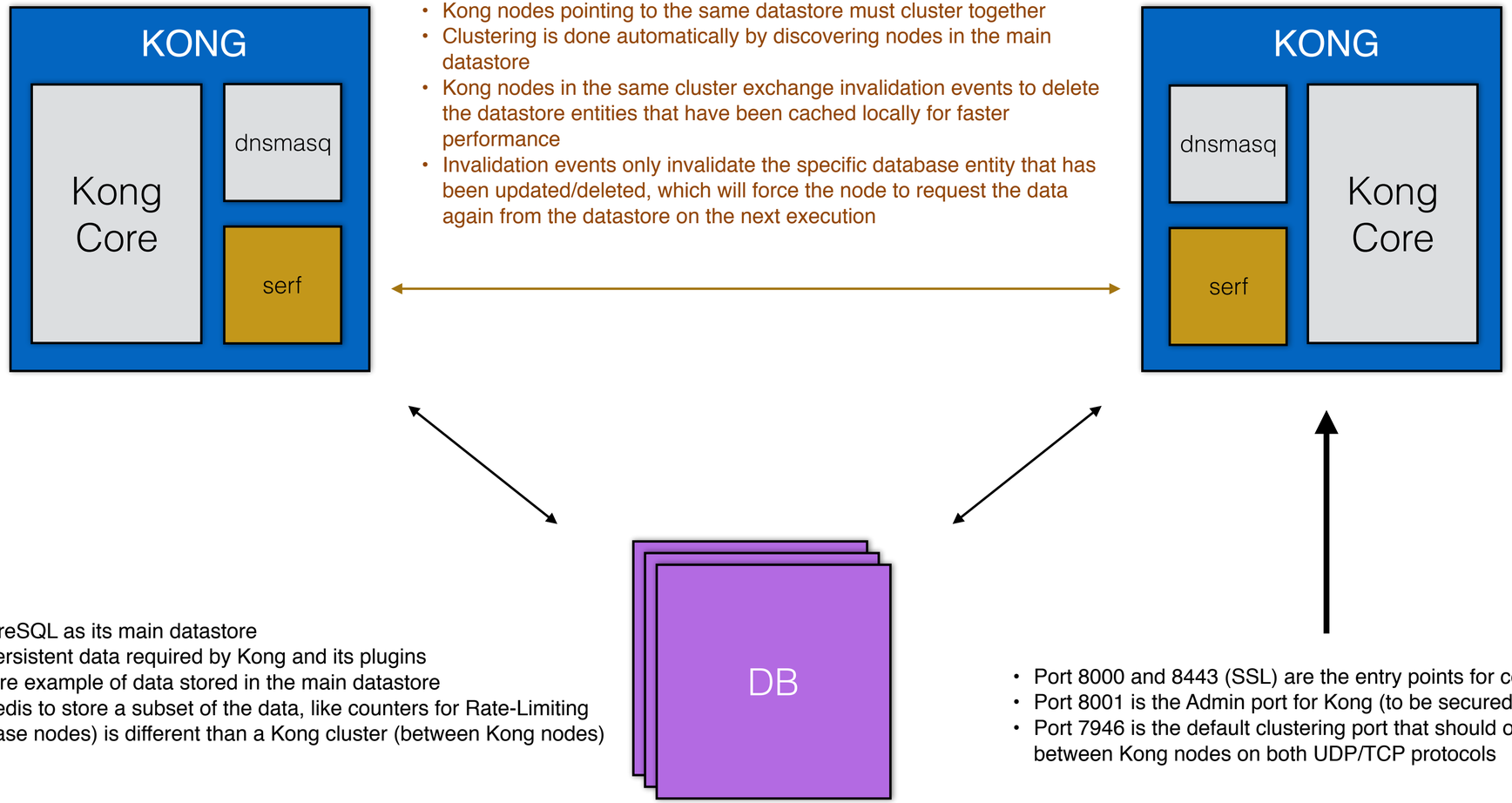
```
$ curl 127.0.0.1:8001/plugins
```

Kong Components

Kong is bundled with its required dependencies:

- dnsmasq, to resolve DNS addresses
- serf, for Kong nodes clustering

The dependencies are abstracted away from the final user.

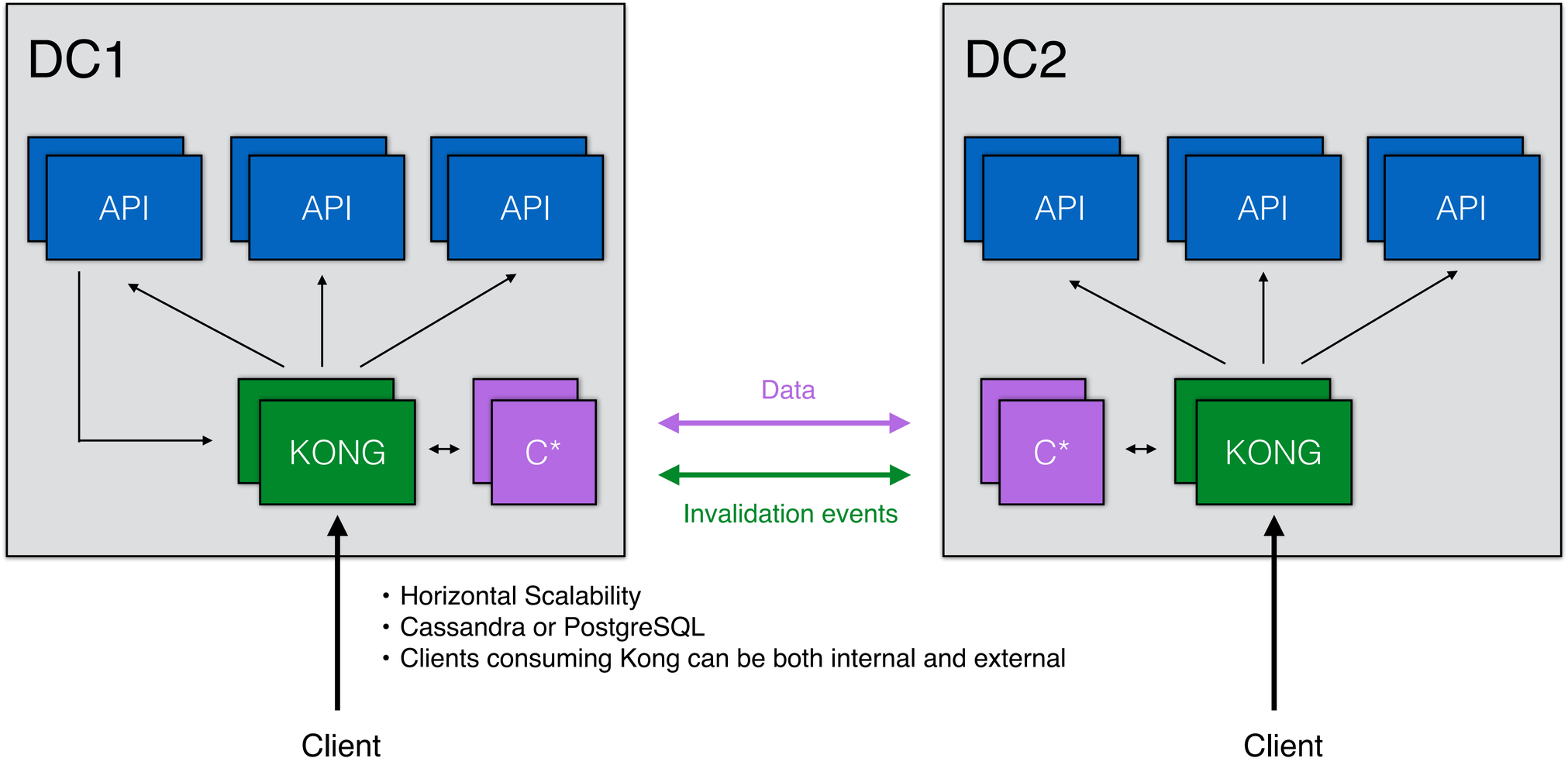


- Kong nodes pointing to the same datastore must cluster together
- Clustering is done automatically by discovering nodes in the main datastore
- Kong nodes in the same cluster exchange invalidation events to delete the datastore entities that have been cached locally for faster performance
- Invalidation events only invalidate the specific database entity that has been updated/deleted, which will force the node to request the data again from the datastore on the next execution

- Kong supports Cassandra or PostgreSQL as its main datastore
- The main datastore stores all the persistent data required by Kong and its plugins
- APIs, consumers, credentials, etc are example of data stored in the main datastore
- Optionally some plugins can use Redis to store a subset of the data, like counters for Rate-Limiting
- A database cluster (between database nodes) is different than a Kong cluster (between Kong nodes)

- Port 8000 and 8443 (SSL) are the entry points for consumers
- Port 8001 is the Admin port for Kong (to be secured)
- Port 7946 is the default clustering port that should only be available between Kong nodes on both UDP/TCP protocols

Multi-DC deployment



- Horizontal Scalability
- Cassandra or PostgreSQL
- Clients consuming Kong can be both internal and external

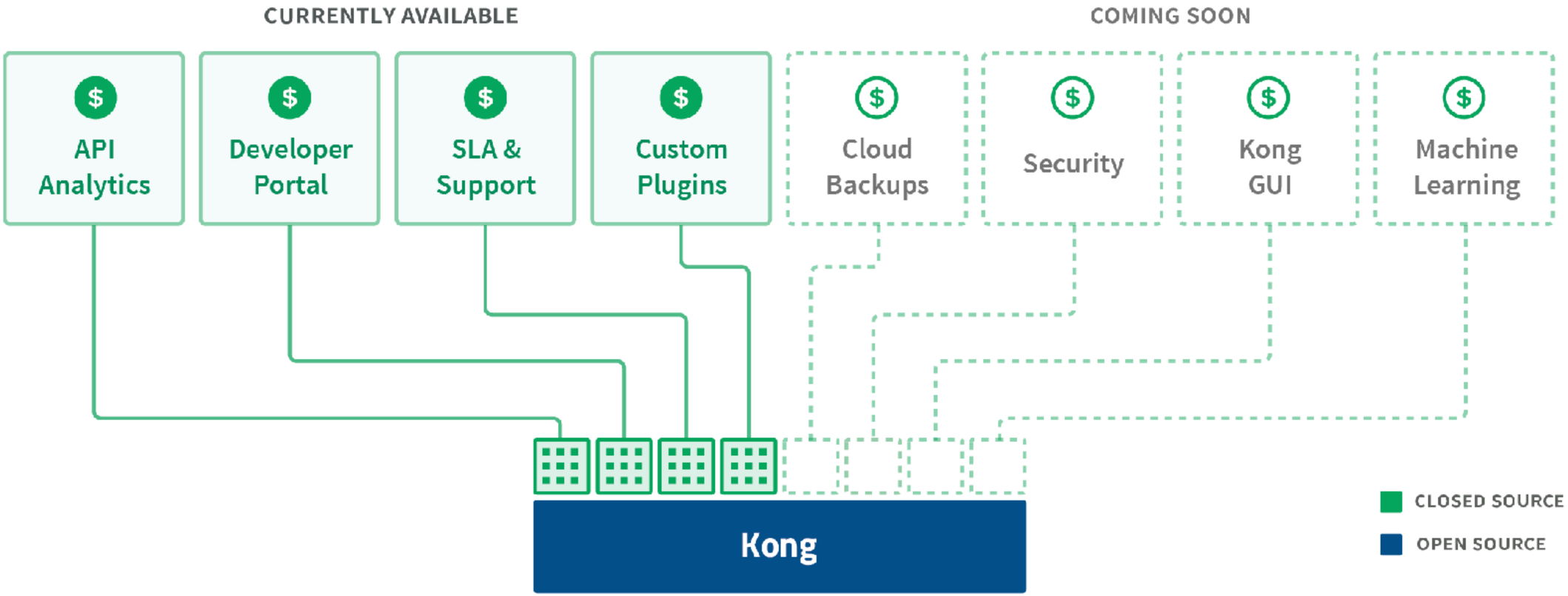
Next 0.10 version will include:

- AWS Lambda support
- Dynamic Load Balancing through :8001/upstreams
- SRV DNS support for DNS resolutions (today only A records supported)
- Cassandra 3.x support
- kong backup create & kong backup import

2017 roadmap will include:

- Built-in WAF (Web Application Firewall)
- Admin API ACL + Auditing Logs
- OpenID Connect plugin
- SOAP to REST
- Kong GUI

Mashape Enterprise Platform



AVAILABLE:

On-Prem VPC • Dedicated Cloud • Shared Cloud

SOME ENTERPRISE CUSTOMERS

Rakuten

Ferrari

HealthCare.gov

US GOVERNMENT

PCR

British Columbia | Ministry of Education

CANADA GOVERNMENT

Western Digital®

The New York Times

EUROSTAR™

Intermountain Healthcare

AND MORE IN

GOVERNMENT HEALTHCARE FINANCE HIGH-TECH TELCO

IOT HARDWARE RESEARCH TRANSPORTATION

Thank You

getkong.org • mashape.com

@thefosk

linkedin.com/marcopalladino