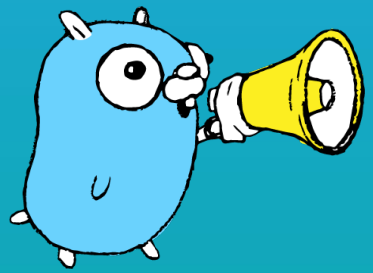
Taipei Taiwan, 06/2024

Implement Idempotency With API Gateway Plugin

Gaston Chiu

Crypto.com

Backend Engineer



Today’s (glorious) blather.

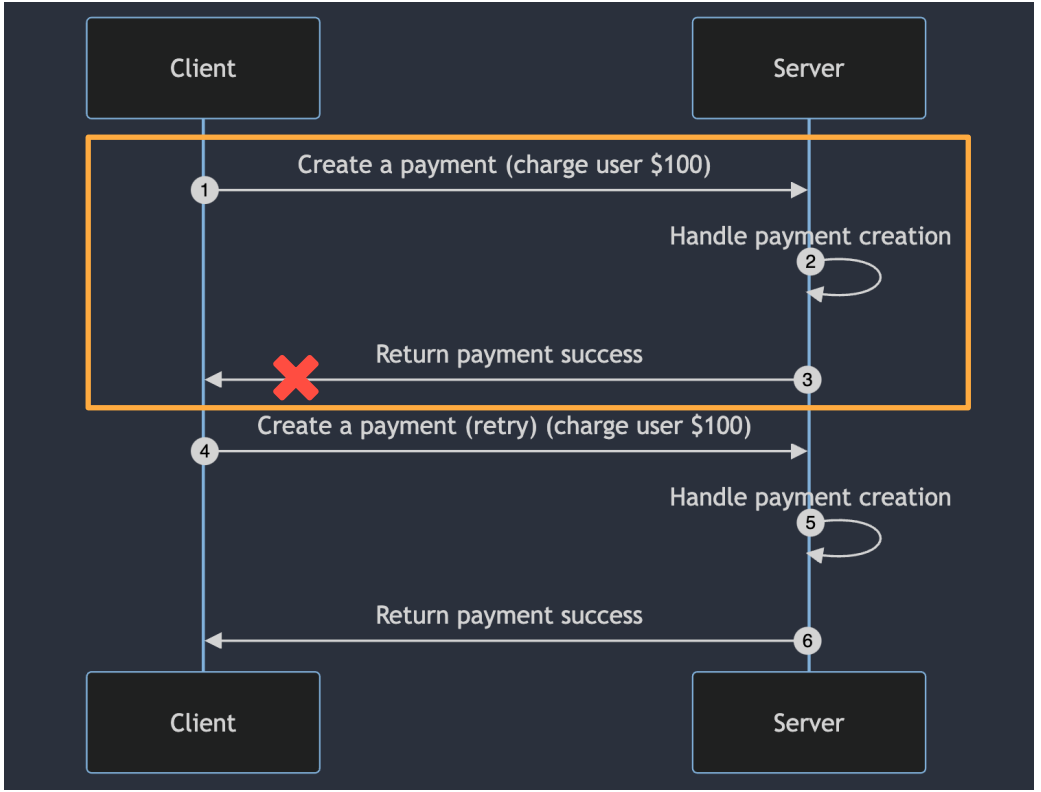
SECTION ONE

Introduction

Create payment can’t retry

Client Side: One payment request is success.

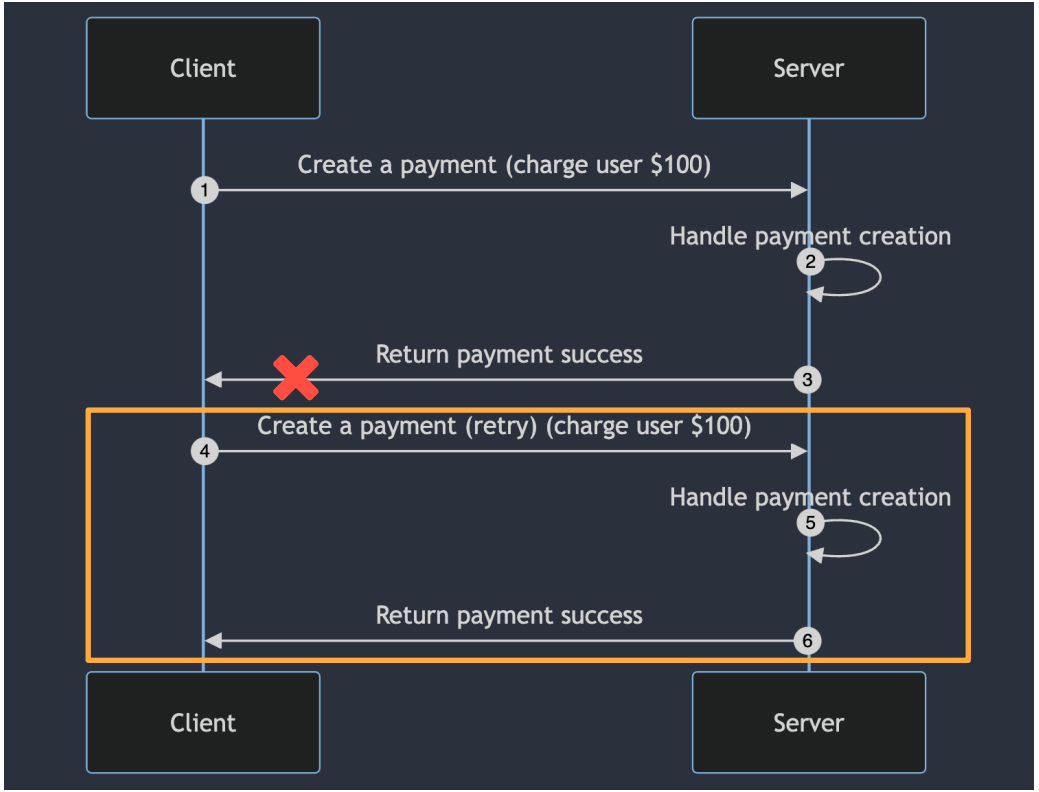
Server Side: There are two payment requests success.



Create payment can’t retry

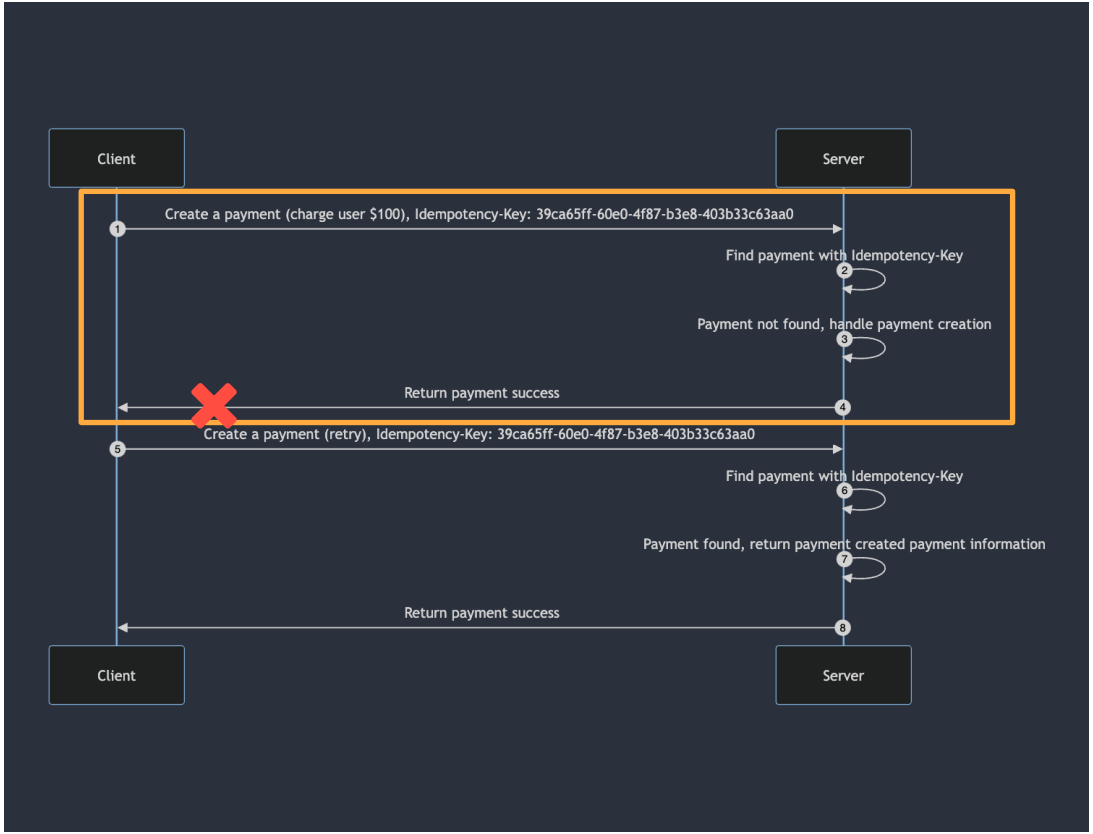
Client Side: One payment request is success.

Server Side: There are two payment requests success.



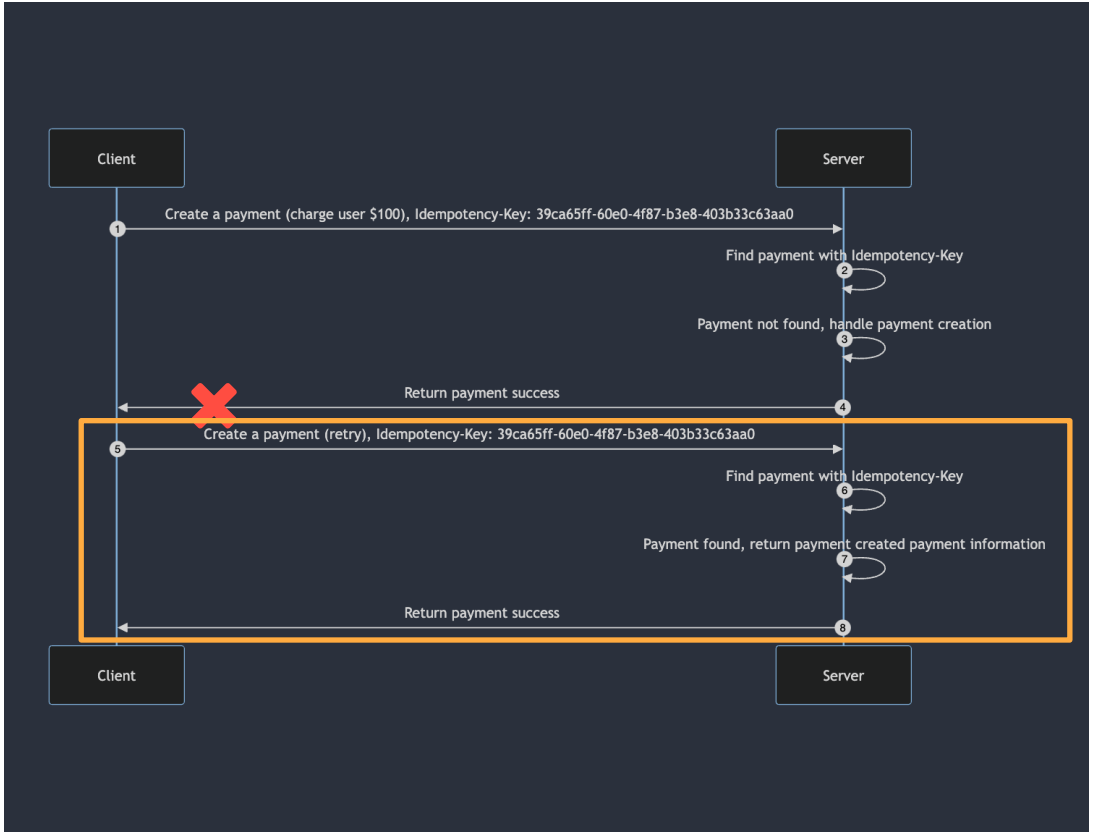
Create payment safe retry with idempotency key

```
Method: POST
Resource: /api/v1/payment
Header:
- X-Idempotency-Key: (uuid)

Body:
{
    "amount": 100,
    "currency": "USD"
}
```



Create payment safe retry with idempotency key

```
Method: POST
Resource: /api/v1/payment
Header:
- X-Idempotency-Key: (uuid)

Body:
{
    "amount": 100,
    "currency": "USD"
}
```



RESTFUL API Server

- GET
- PUT/PATCH
- DELETE
- POST

SECTION TWO

Idempotency Standard Design

Idempotency Key Header

Idempotency-Key: Random string (ex: uuid v4)

Examples

time	method	input	response code	response
first	POST	- new idempotency key	201 (created)	payment DTO
second	POST	- same idempotency key - same body/header hash	201 (created)	payment DTO
thrid	POST	- same idempotency key - different body or headers hash	422	unprocessable
first	POST	- new idempotency key	504 (gateway timeout)	client disconnect before getting the response
second	POST	- same idempotency key - same body/header hash	201 (created)	was success nothing new created
thrid	POST	- same idempotency key - different body or headers hash	422	unprocessable

Examples

time	method	input	response code	response
first	POST	- new idempotency key	201 (created)	payment DTO
second	POST	- same idempotency key - same body/header hash	201 (created)	payment DTO
thrid	POST	- same idempotency key - different body or headers hash	422	unprocessable
first	POST	- new idempotency key	504 (gateway timeout)	client disconnect before getting the response
second	POST	- same idempotency key - same body/header hash	201 (created)	was success nothing new created
thrid	POST	- same idempotency key - different body or headers hash	422	unprocessable

Examples

time	method	input	response code	response
first	POST	- new idempotency key	400 (bad request)	error response
second	POST	- same idempotency key - same body/header hash	400 (bad request)	error response
thrid	POST	- same idempotency key - different body or headers hash	422	unprocessable
first	POST	- new idempotency key	500 (internal error)	error response
second	POST	- same idempotency key - same body/header hash	500 (internal error)	error response
thrid	POST	- same idempotency key - different body or headers hash	422	unprocessable

Error Scenarios

Status Code	Description
400 Bad Request	Idempotency Header Missing
422 Unprocessable Content	If there is an attempt to reuse an idempotency key with a different request payload
409 Conflict	If the request is retried, while the original request is still being processed

Calculate Fingerprint

* Checksum of the entire request payload.
* Checksum of selected element(s) in the request payload.

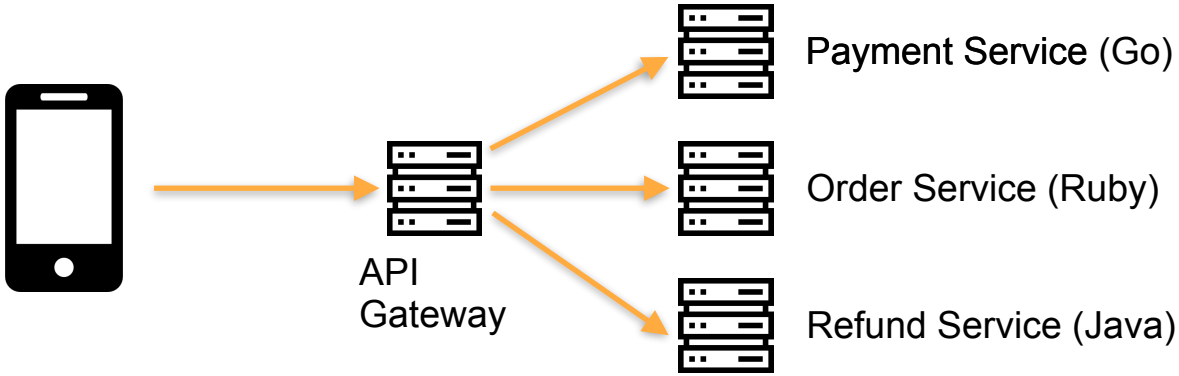
SECTION THREE

Solution

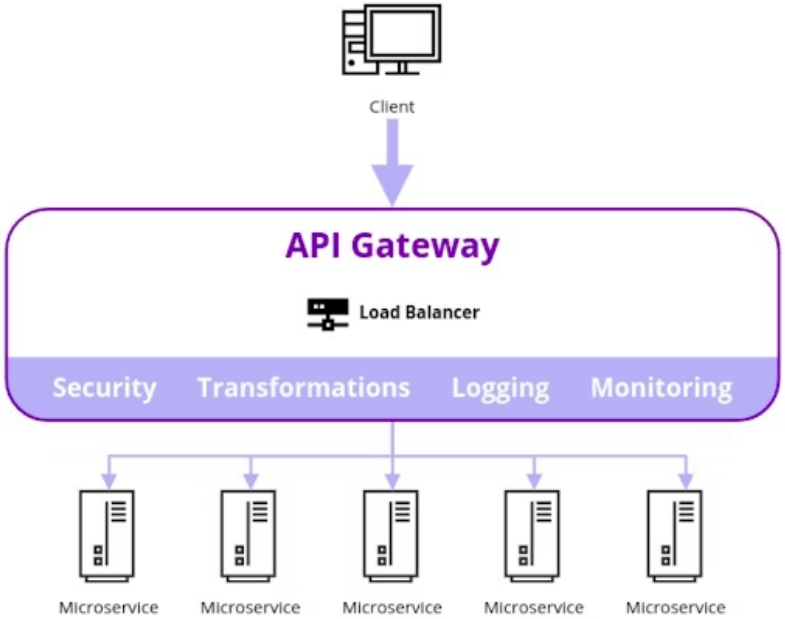
Pseudo Code

```
key = req.header["Idempotency-Key"]
fingerprint = Fingerprint(url+body+whitelisted(headers))
Lock(key)
defer Unlock(key)
cachedResponse = GetFromCache(key, fingerprint)
if cachedResponse != nil {
    return cachedResponse
}
resp = Process(req)
SetResponseToCache(key, resp, fingerprint)
```

Idempotency under microservices architecture

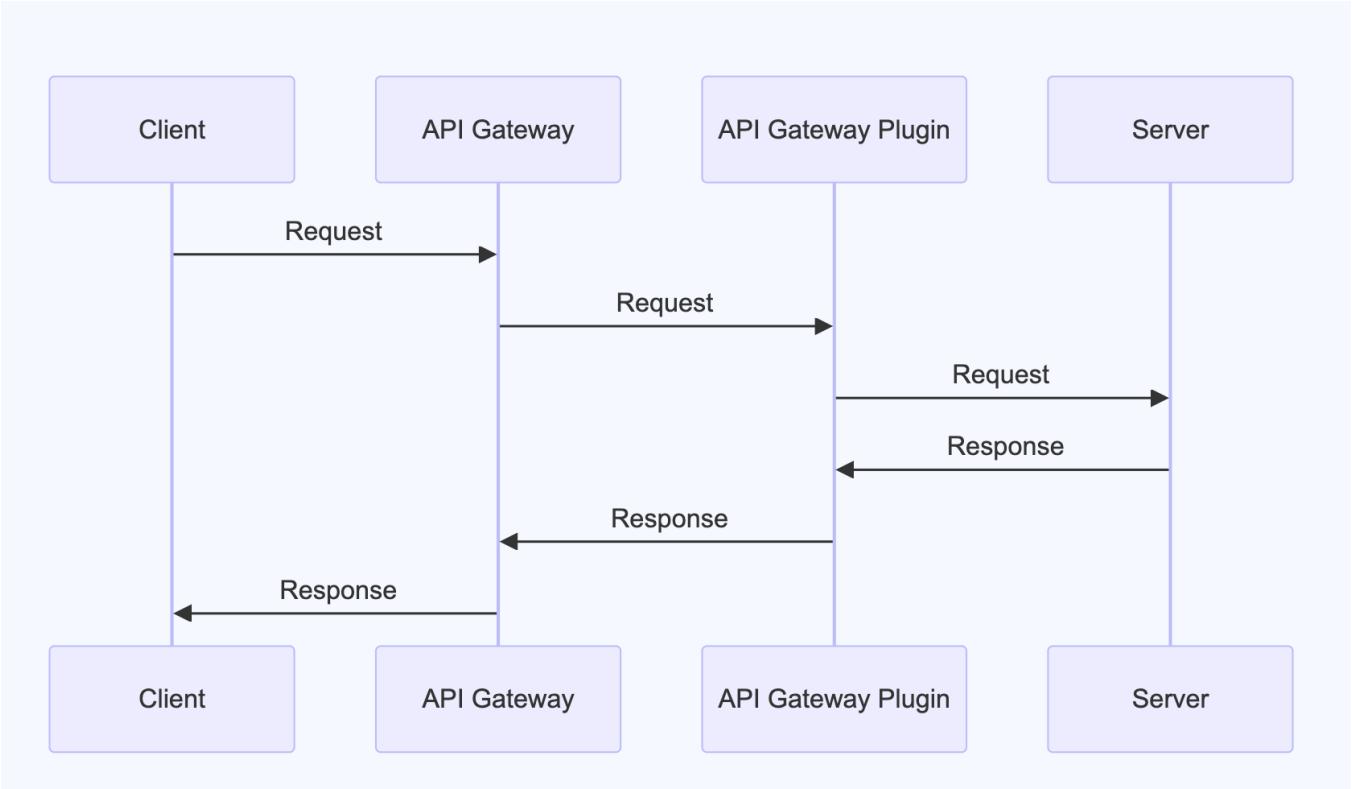


Kong Gateway (API Gateway)

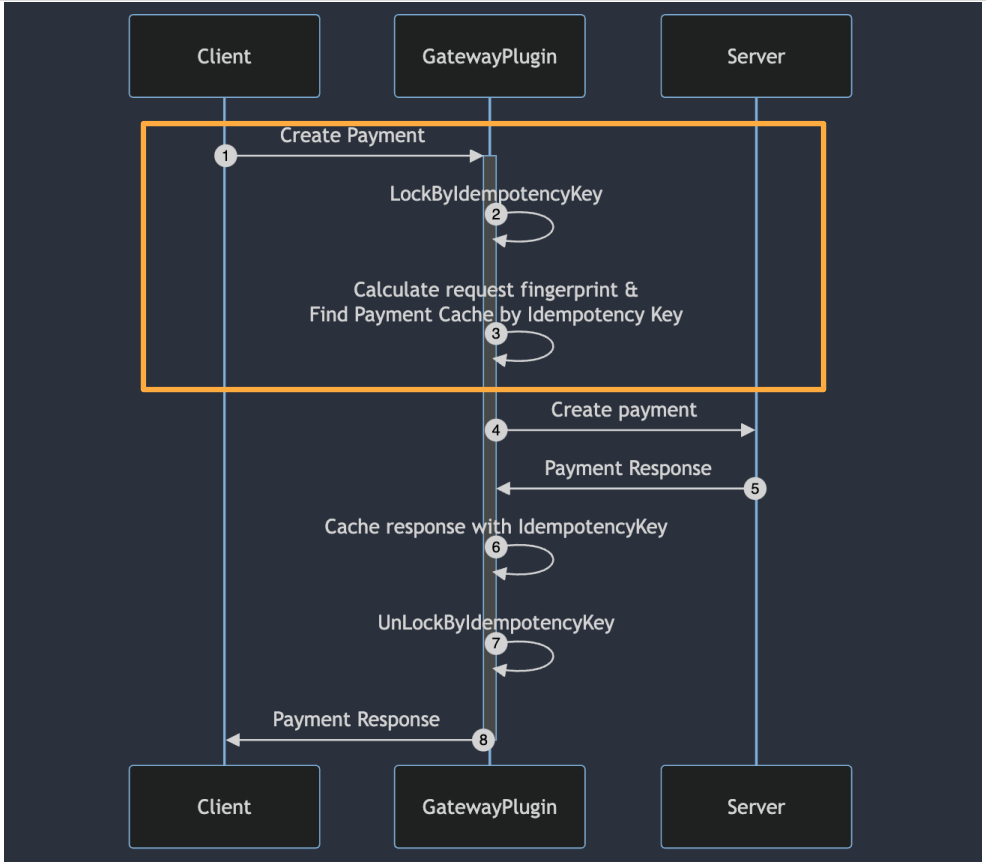


https://konghq.com/blog/learning-center/what-is-an-api-gateway

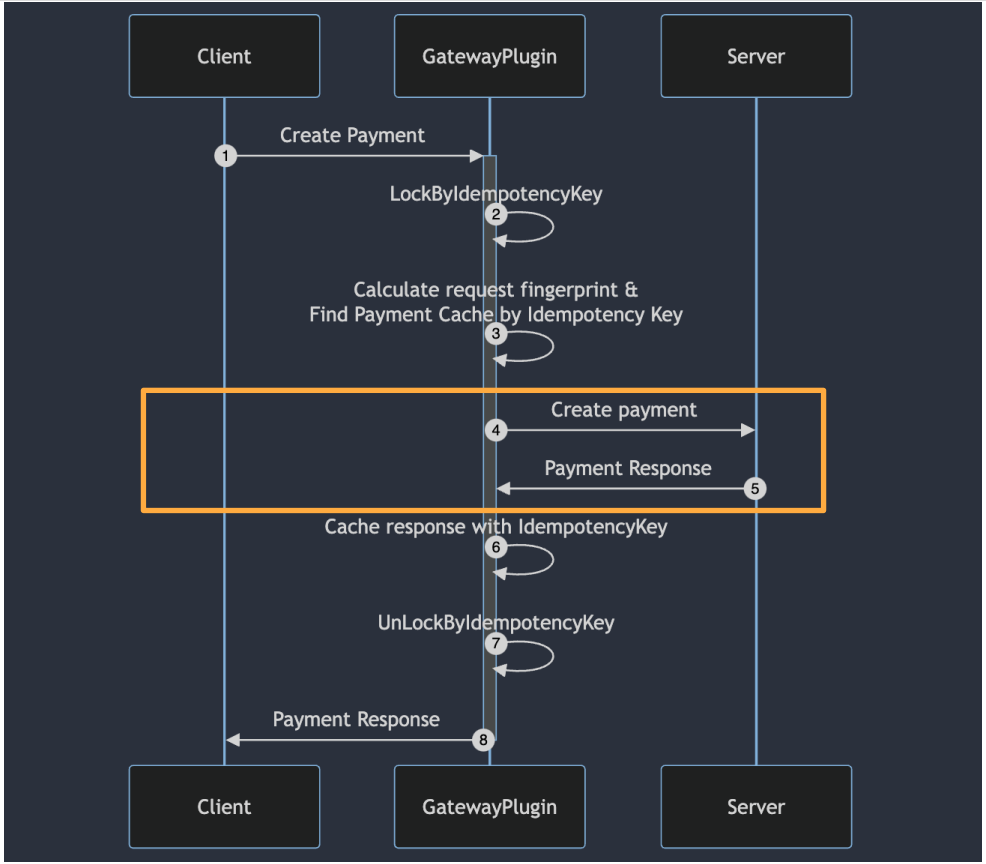
API Gateway Plugin



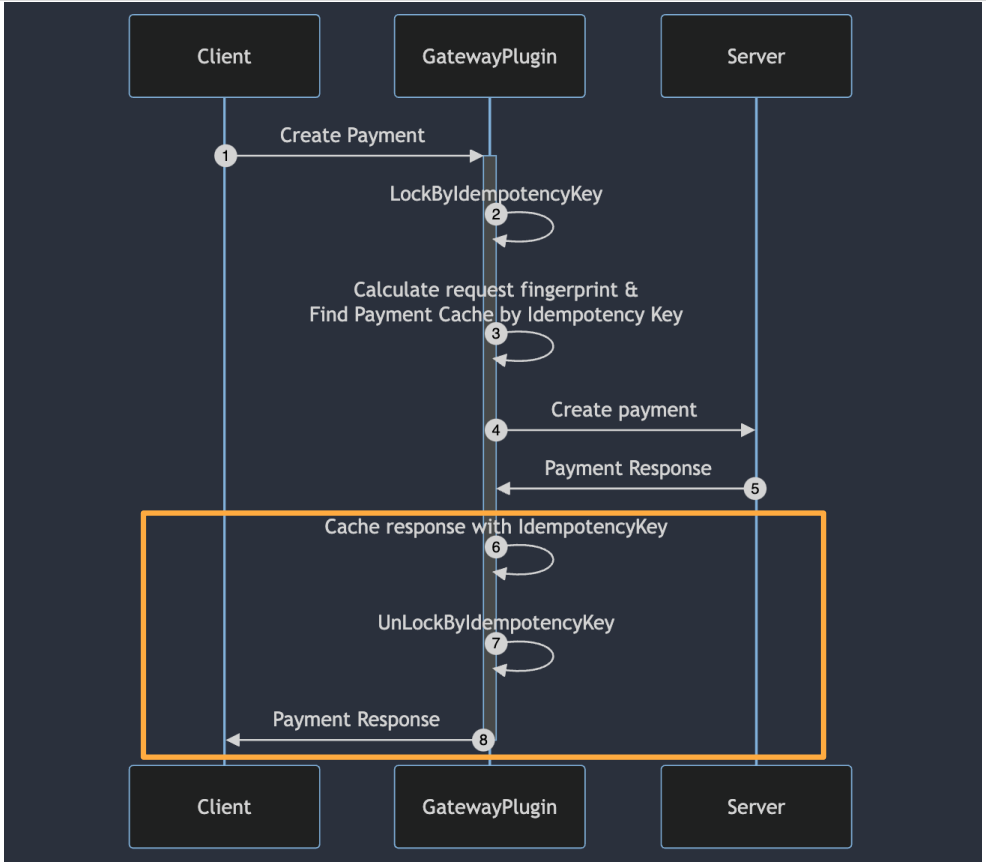
Architecture with gateway plugin (Cache Not Found)



Architecture with gateway plugin (Cache Not Found)



Architecture with gateway plugin (Cache Not Found)



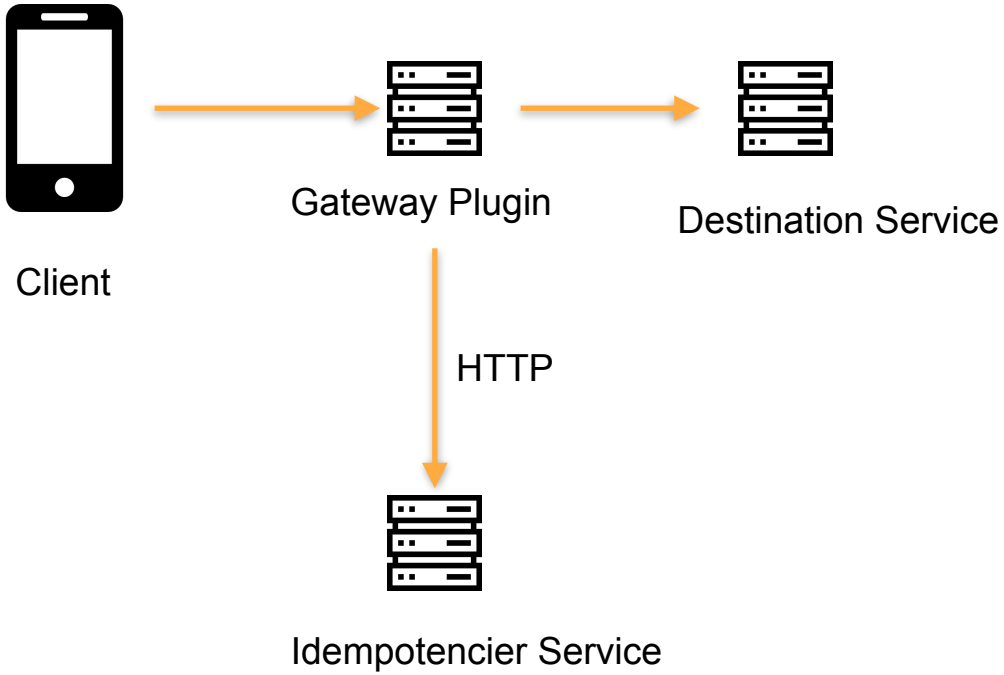
Kong Gateway Plugin With Go PDK

```
github.com/Kong/go-pdk/

func main () {
    server.StartServer(New, Version, Priority)
}

func New() interface{} {
    return &MyConfig{}
}

func (conf *MyConfig) Access (kong *pdk.PDK) {
  ...
}
```

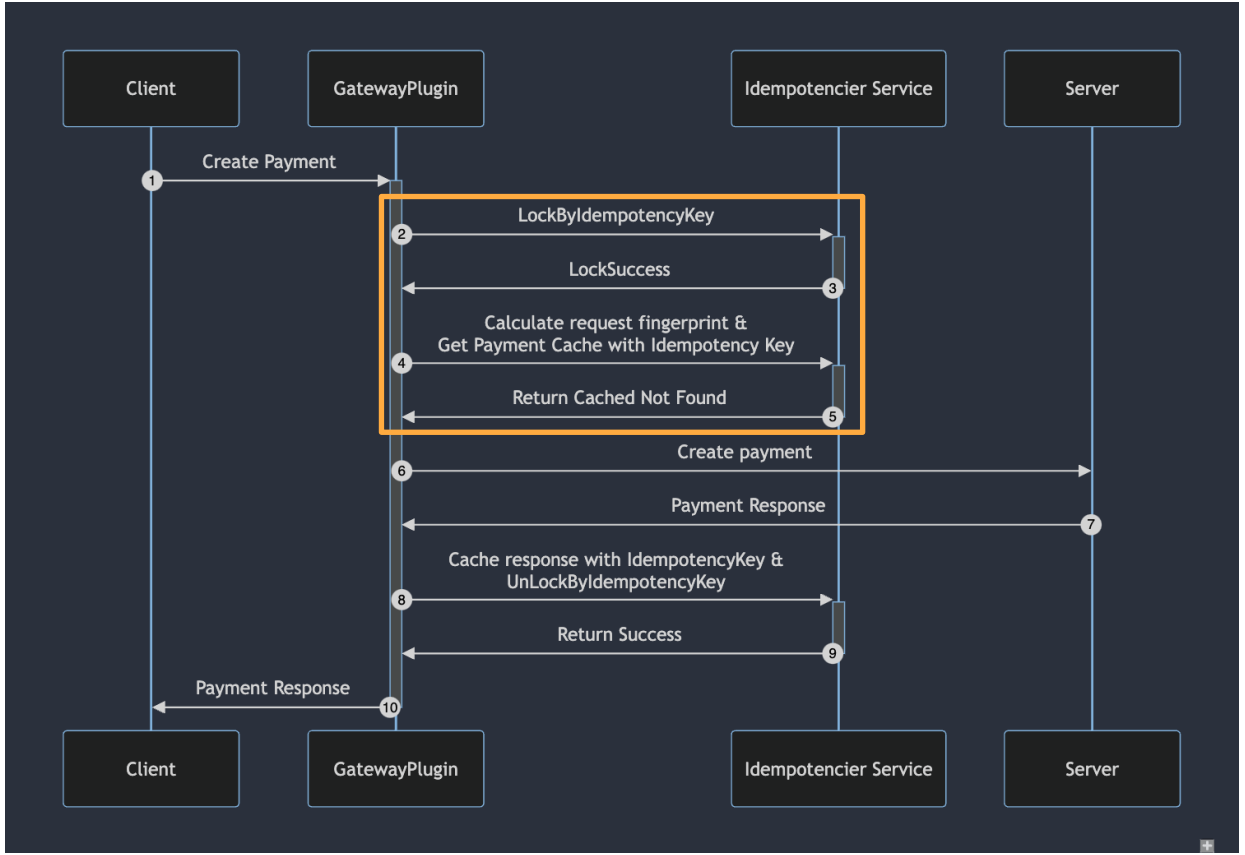
Gateway Plugin With Go PDK

```
type PDK struct {
    Client          client.Client
    Ctx             ctx.Ctx
    Log             log.Log
    Nginx           nginx.Nginx
    Request         request.Request
    Response        response.Response
    Router          router.Router
    IP              ip.Ip
    Node            node.Node
    Service         service.Service
    ServiceRequest  service_request.Request
    ServiceResponse service_response.Response
}
```

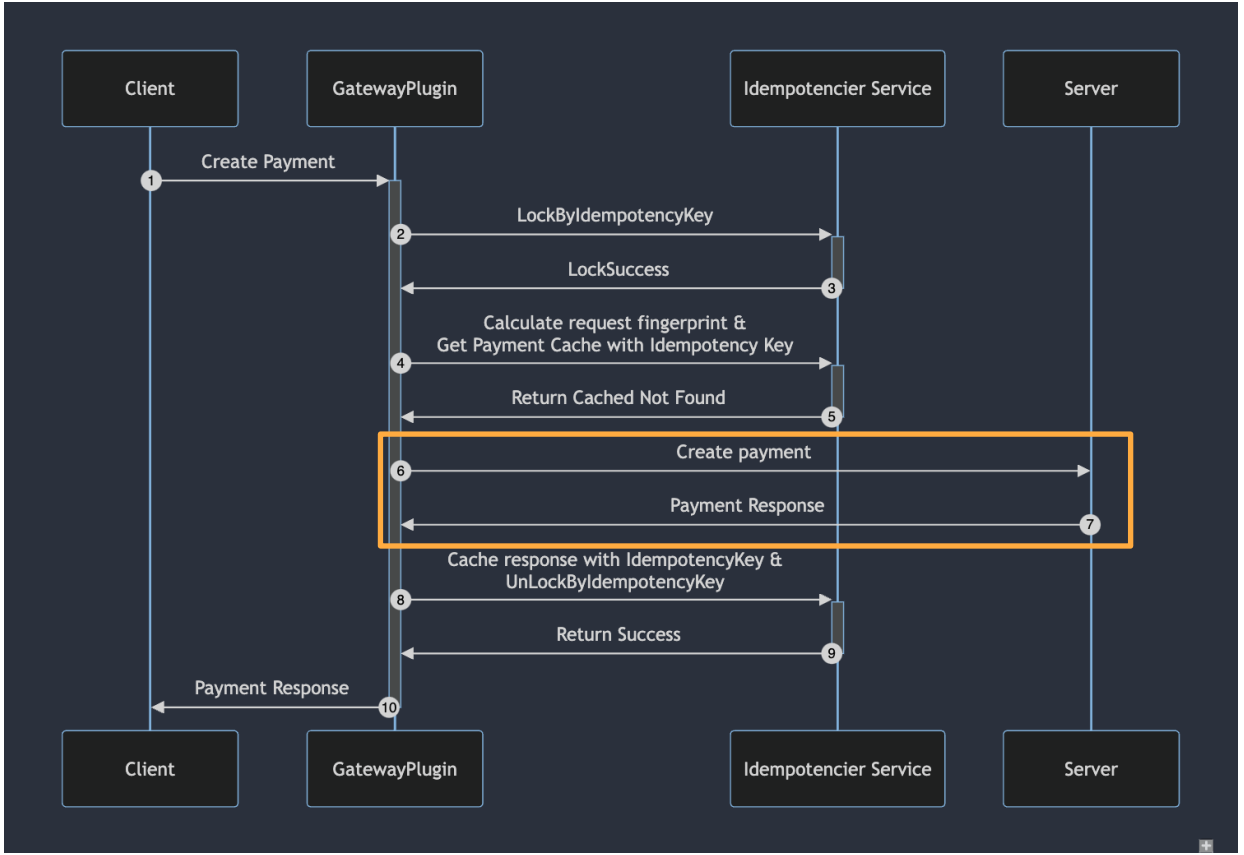
API Gateway Plugin



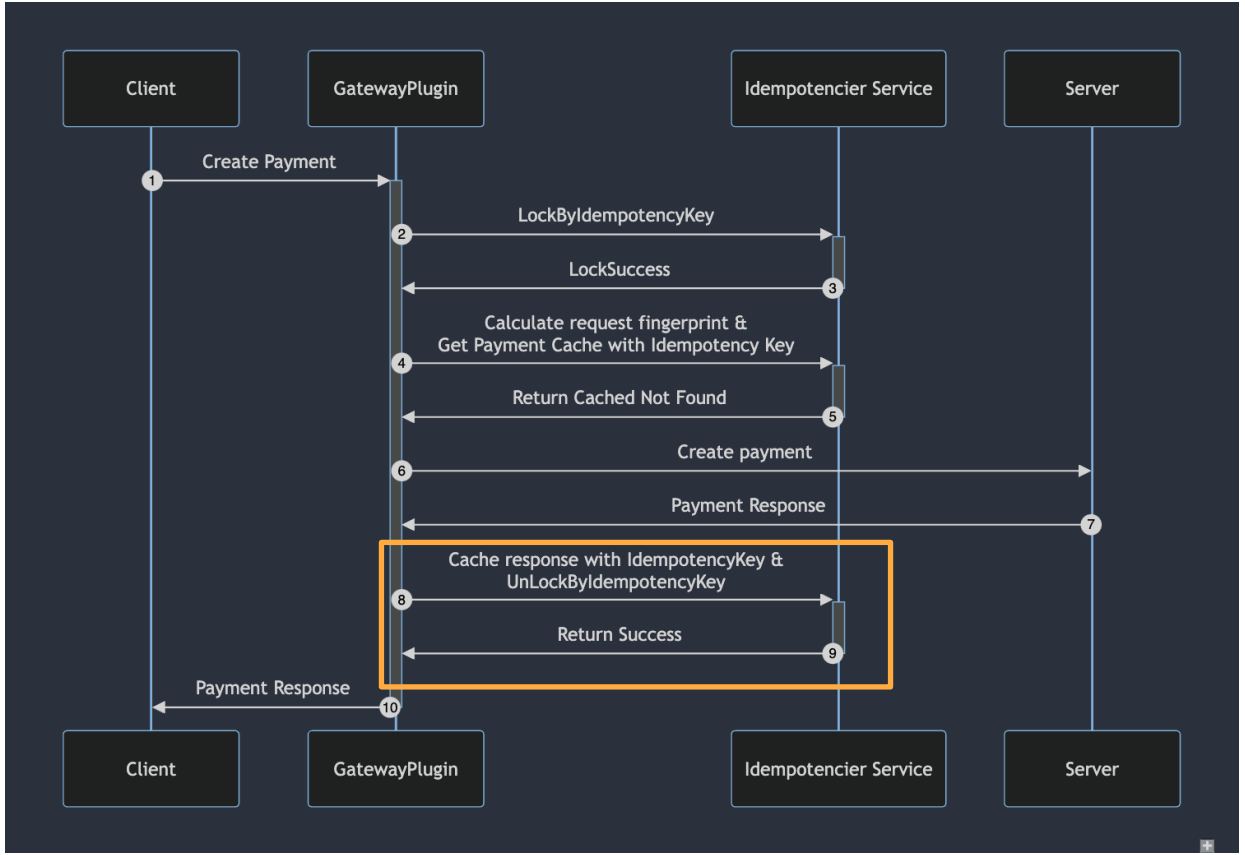
Architecture with gateway plugin



Architecture with gateway plugin



Architecture with gateway plugin

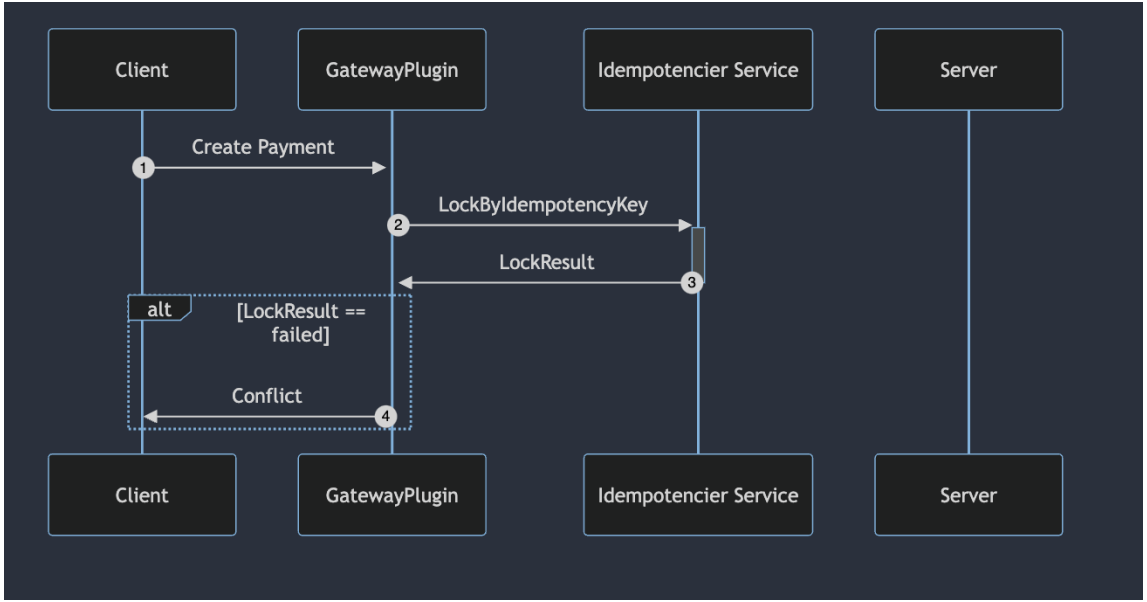


SECTION FOUR

Idempotencier Service

Acquire Lock on Idempotencier Service

```
POST /internal/api/request-locks

Request
{
    "idempotency_key": "<string>"
    "lock_duration": "<int>"
}

Response
{
    "lock_value": "<string>"
}

Error
- concurrent_process_error (409)
```



Acquire lock on Redis

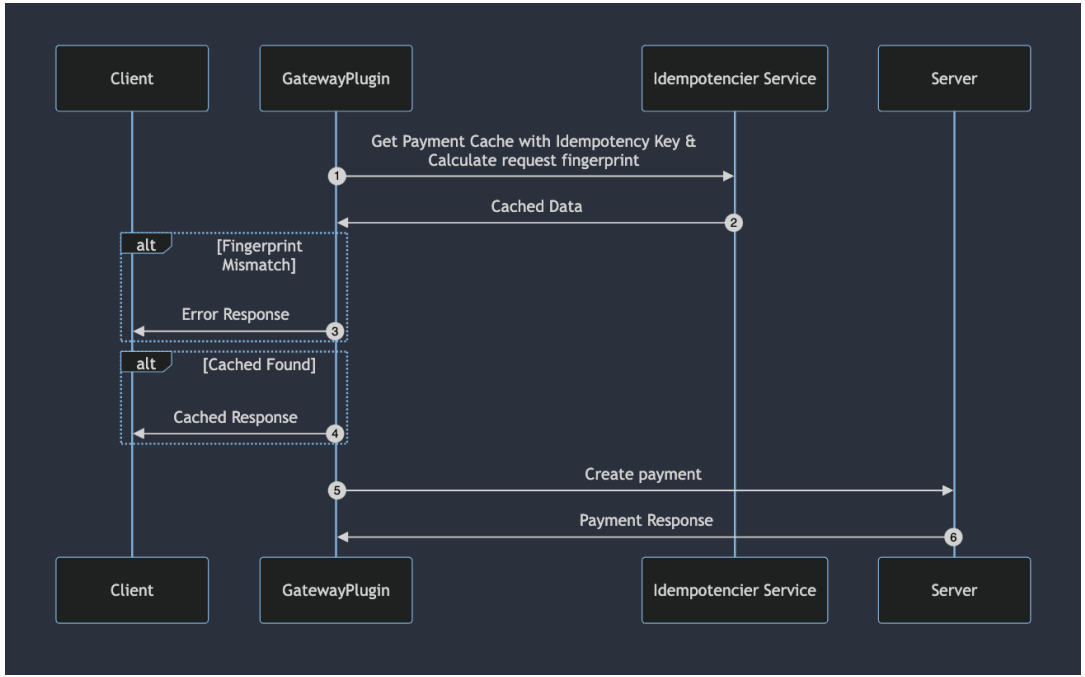
```
// lock
SET resource_name my_random_value NX PX 30000

// unlock
if redis.call("get", key) == value then
    return redis.call("del", key)
else
    return 0
end
```

Lock

```
func Lock() {
    rs := redsync.New()
    mutex := rs.NewMutex(key, redsync.WithTries(5))
    err := mutex.LockContext(ctx)
    if err != nil {
        return "", ConflictRequestError{}
    }

    return mutex.Value(), nil
}
```

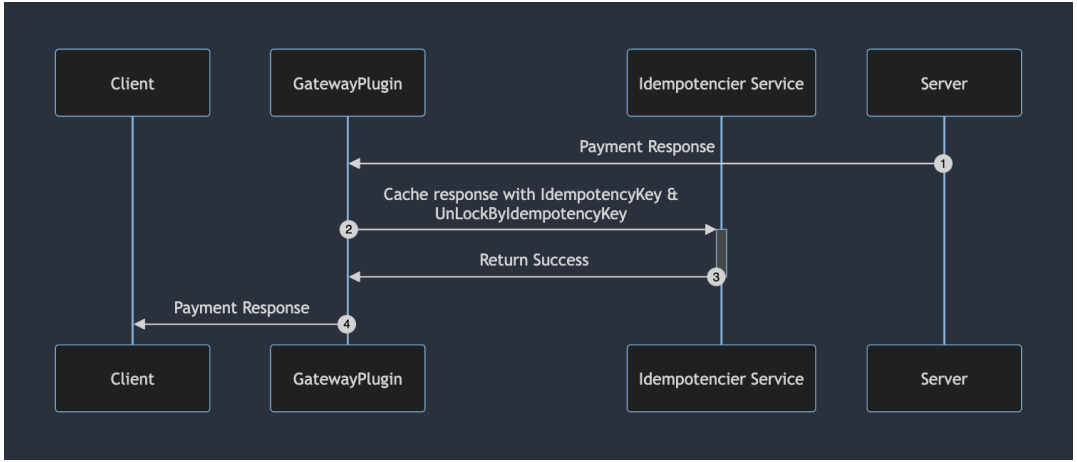
Create a microservice for gateway plugin (GetCache)

```
GET /response/{idempoency_key}

HEADER
- X-Request-Fingerprint: "<string>"

Response:
{
    "status_code": "<int>",
    "headers": "<object>",
    "body": "<object>"
}

Error:
- idempotent_key_not_found (404)
- fingerprint_conflict (422)
```



Create a microservice for gateway plugin (Unlock & SetCache)

```
POST /response

Request
{
    "idempotency_key": "<string>",
    "request_fingerprint": "<string>",
    "response": {
        "headers": "<object>",
        "body": "<object>" ,
        "status_code": "<int>"
    },
    "lock_meta_data": {
        "lock_value": "<string>"
    }
}
```



Unlock

```
func Unlock(value string) error {
    rs := redsync.New()
    mutex := rs.NewMutex(key, redsync.WithValue(value))

    ok, err := mut.UnlockContext(ctx)
    if err != nil || !ok {
        return UnlockError{}
    }

    return nil
}
```

SECTION FIVE

Infrastructure

Cache and Lock Storage (Redis with replication)

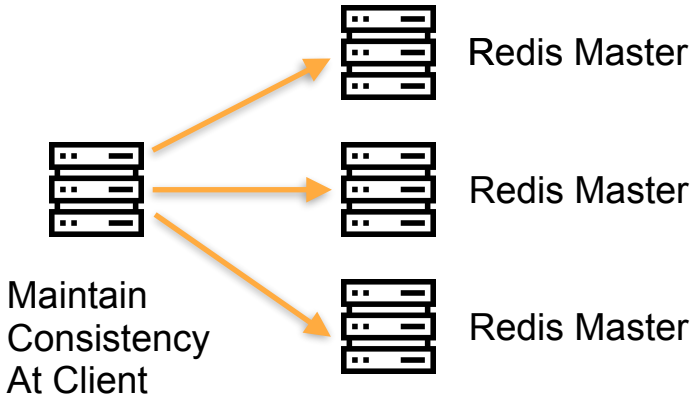
- Pros:
    - Easy for setup and maintain
- Cons:
    - Potentially loss data during failover

Redis Data Loss

1. Client A Set Cache on master
2. Redis response success.
3. The master crashes before key is transmitted to the replica.
4. The replica gets promoted to master.
5. DataLoss

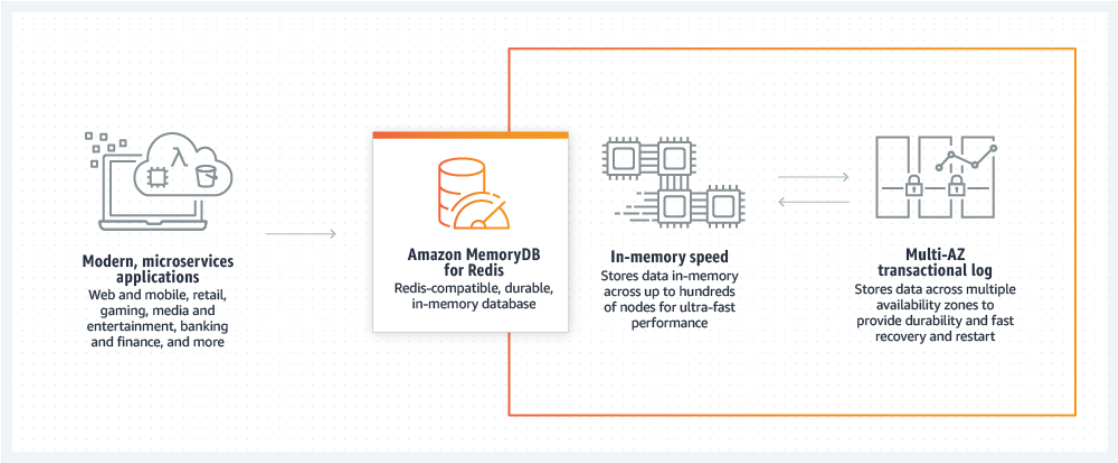
Cache and Lock Storage (RedLock)

- Pros:
    - More secure on data
- Cons:
    - Complicated for setup and implementation



Cache and Lock Storage (Amazon MemoryDB for Redis)

- Pros:
  - Remove the risk for losing data
  - Redis compatible
- Cons:
  - More cost



Build Gateway Plugin

```
# Build Golang plugins

FROM golang:1.20 AS plugin-builder

WORKDIR /builder

COPY ./hello ./go_plugins/hello

RUN find ./go_plugins -maxdepth 1 -mindepth 1 -type d -not -path "*/.git*" | \
    while read dir; do \
        cd $dir && go build -o /builds/$dir main.go  ; \
    done

# Build Kong
FROM kong:3.4.0-ubuntu

COPY --from=plugin-builder ./builds/go_plugins/  ./kong/

USER kong
```

Add Plugin Env

```
  kong-gateway:
    build: .
    environment:
      KONG_PLUGINS: "bundled,hello"
      KONG_PLUGINSERVER_NAMES: hello
      KONG_PLUGINSERVER_HELLO_START_CMD: /kong/hello
      KONG_PLUGINSERVER_HELLO_QUERY_CMD: /kong/hello -dump
    ports:
      - "8000:8000"
      - "8443:8443"
      - "8001:8001"
      - "8444:8444"
    depends_on:
      database:
        condition: service_healthy
      init_db:
        condition: service_completed_successfully
```

Route Config

```
plugins:
    - config:
        replace:
          uri: /service/test
      enabled: true
      name: request-transformer
    - config:
        message: HELLO!
      enabled: true
      name: hello
```

Thank you