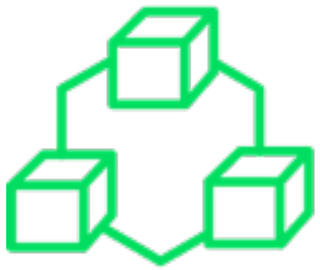

# Microservice Architecture Proposal

## FOR HUAWEI

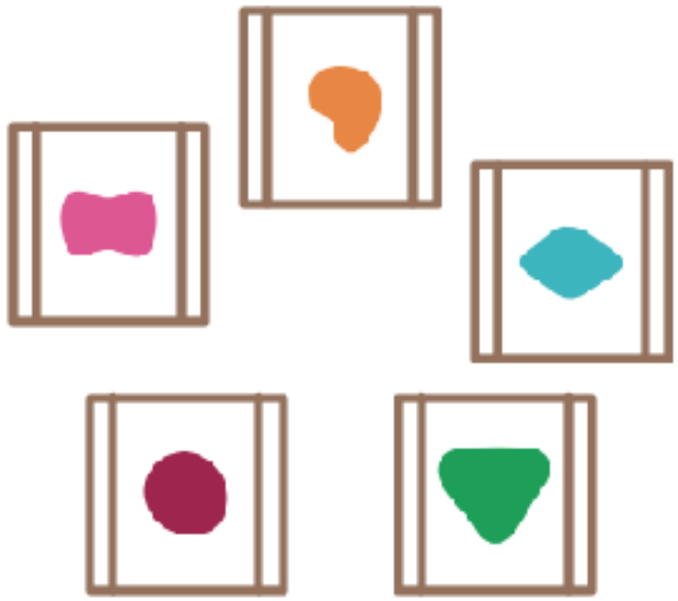


# CONTENT

- Microservice
- Key points of design
- Architecture Proposal
- Evolution Suggestions

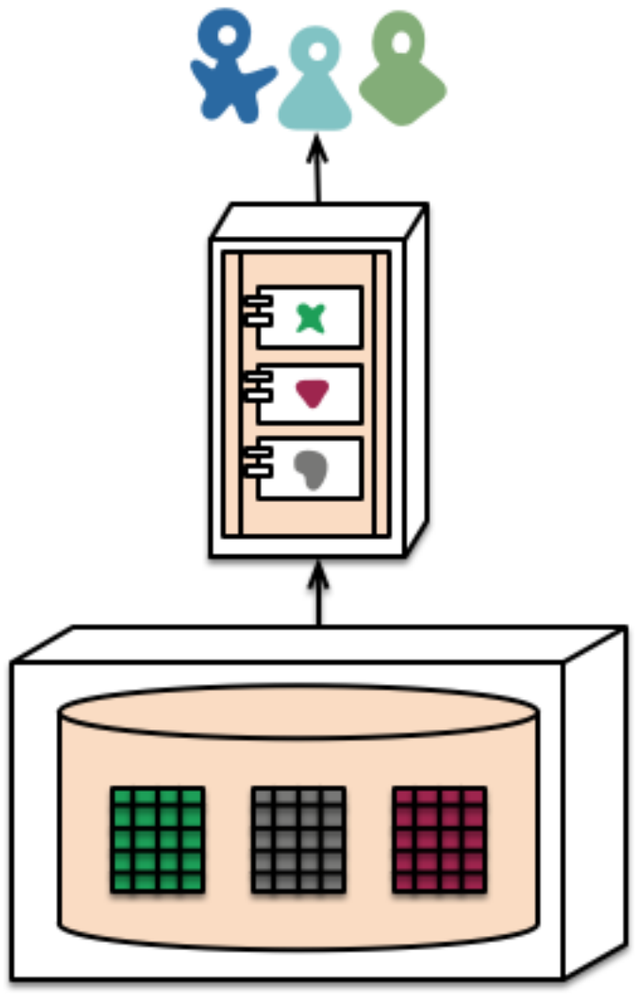
# Microservices

common characteristics of this architectural style

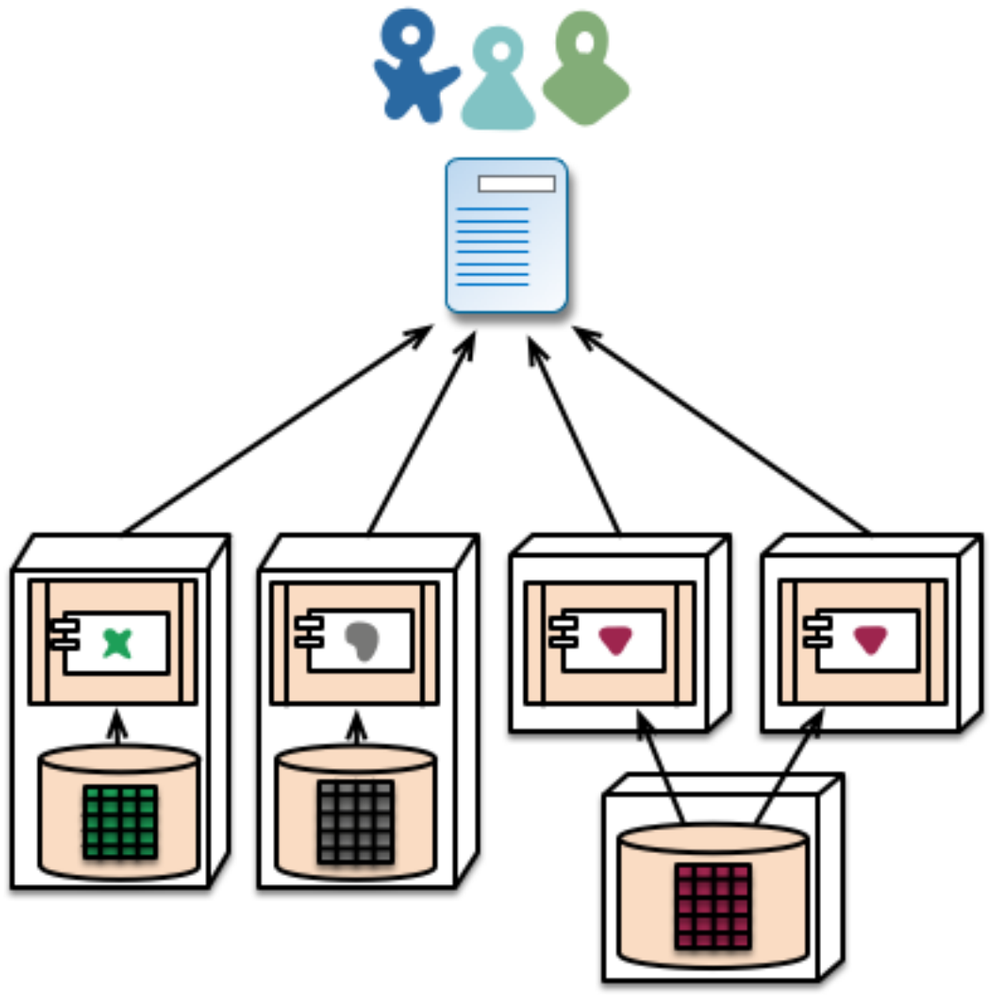


by James Lewis and Martin Fowler

# MICROSERVICE



monolith - single database



microservices - application databases

# ADVANTAGES

- Advantages
  - deploy, release, and operation independently
  - refine resource usage
  - reduce interference between features
  - freedom for technology options
  - improve agility of organization
  - tackles the problem of complexity, faster to develop, easier to understand and maintain

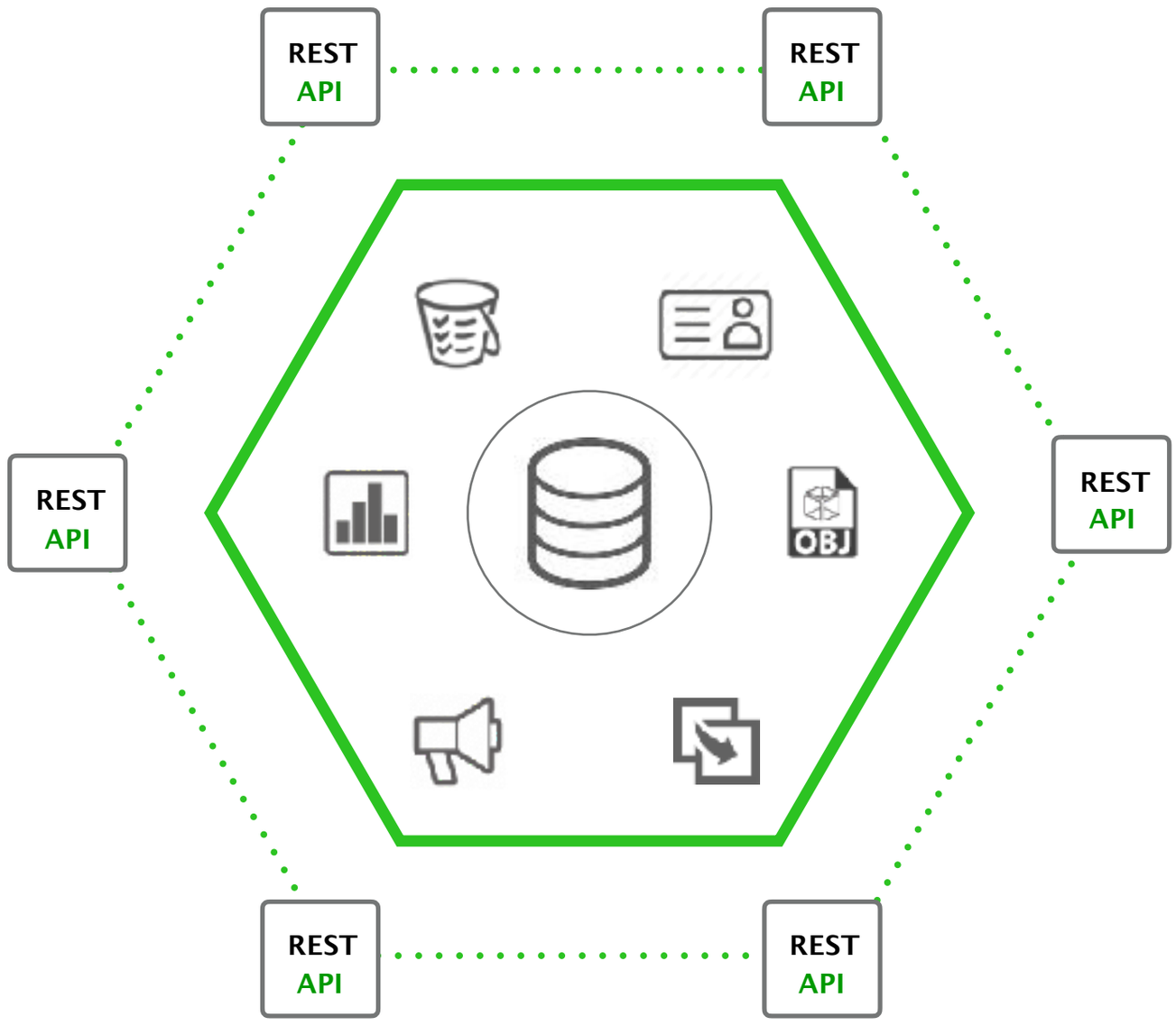
➤ *Differences with modules?*

# DISADVANTAGES

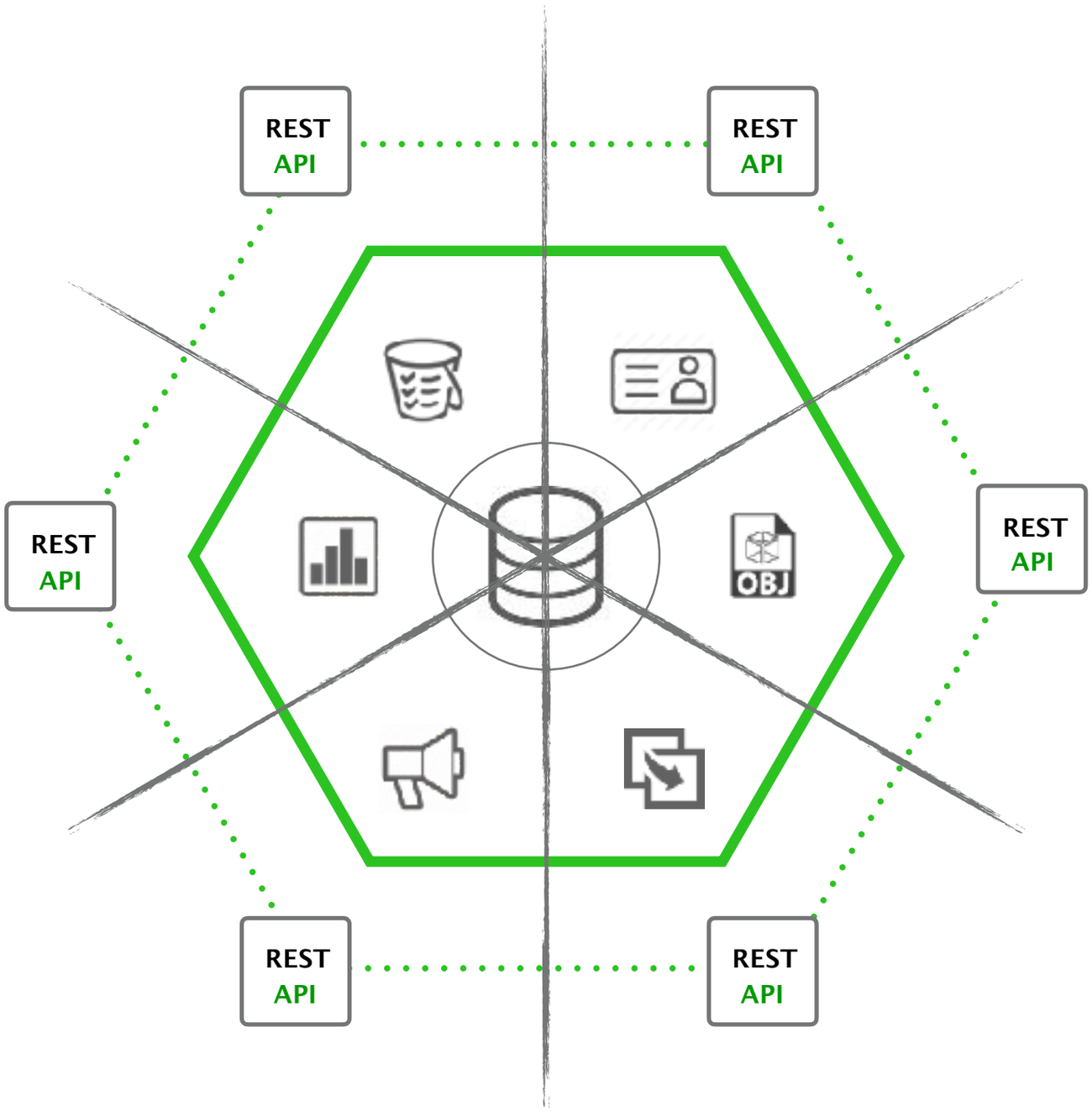
- Disadvantages
  - complexity of distributed system
  - demands for infrastructure and DevOps
  - demands for adaptive organization structure

# Key points of design

# MONOLITHS



# SERVICE PARTITION




**Prefer Vertical Partitioning**

- *DB splitting is critical*

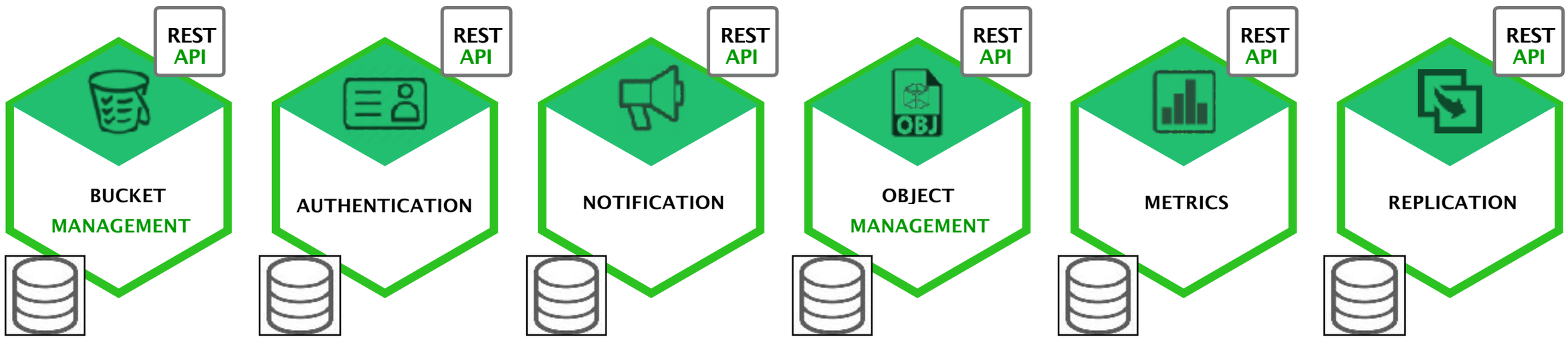
**Core Principle**

- *high cohesion*
- *low coupling*

**Operable Principle**

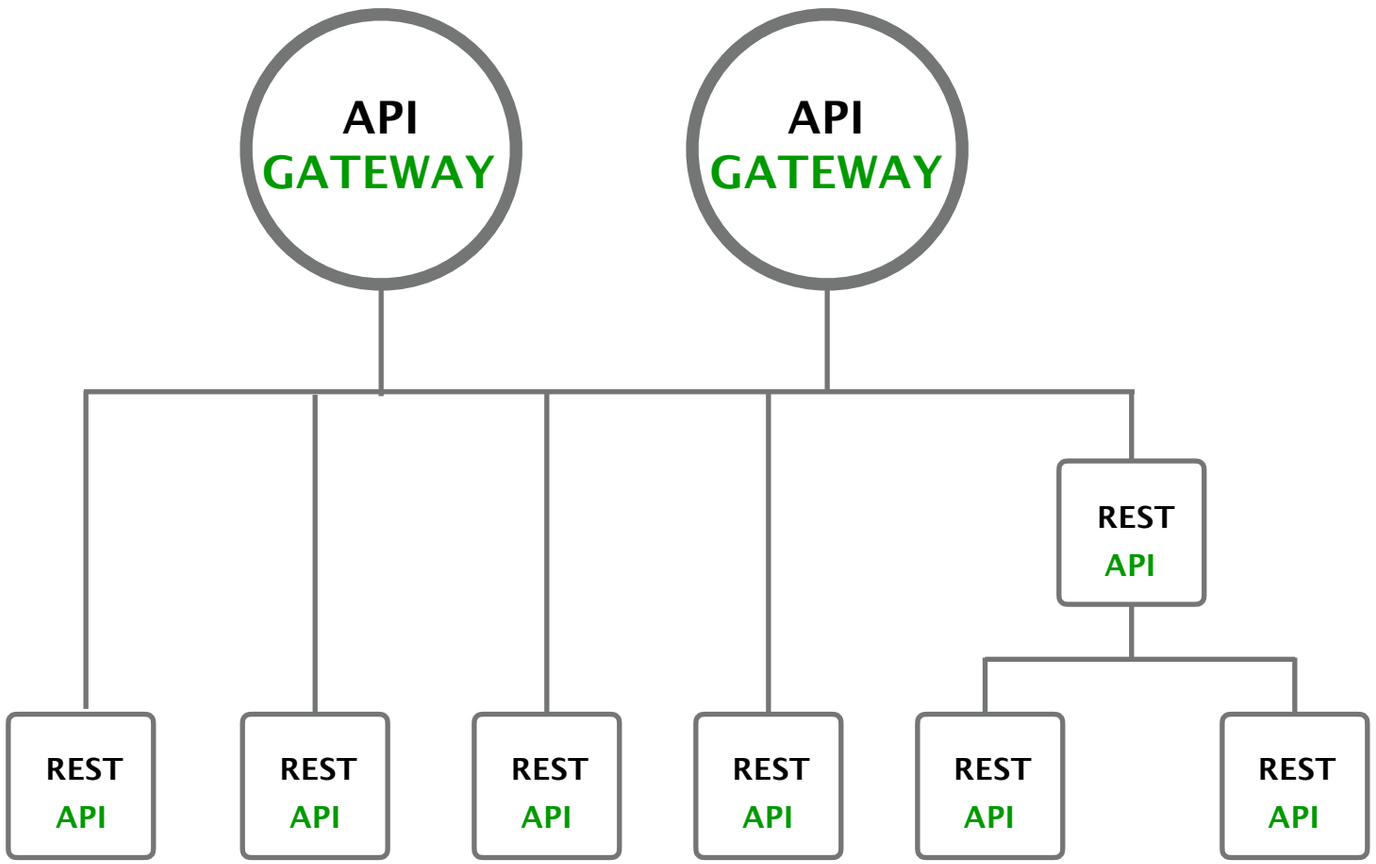
- *Orthogonal design*

# SERVICE CHARACTERS




- *Present customer value preferentially*
- *Cohesive for independence*
- *Reduce interaction*
- *Concern consistency requirements*
- *Treat performance issues reasonably*

# API DESIGN

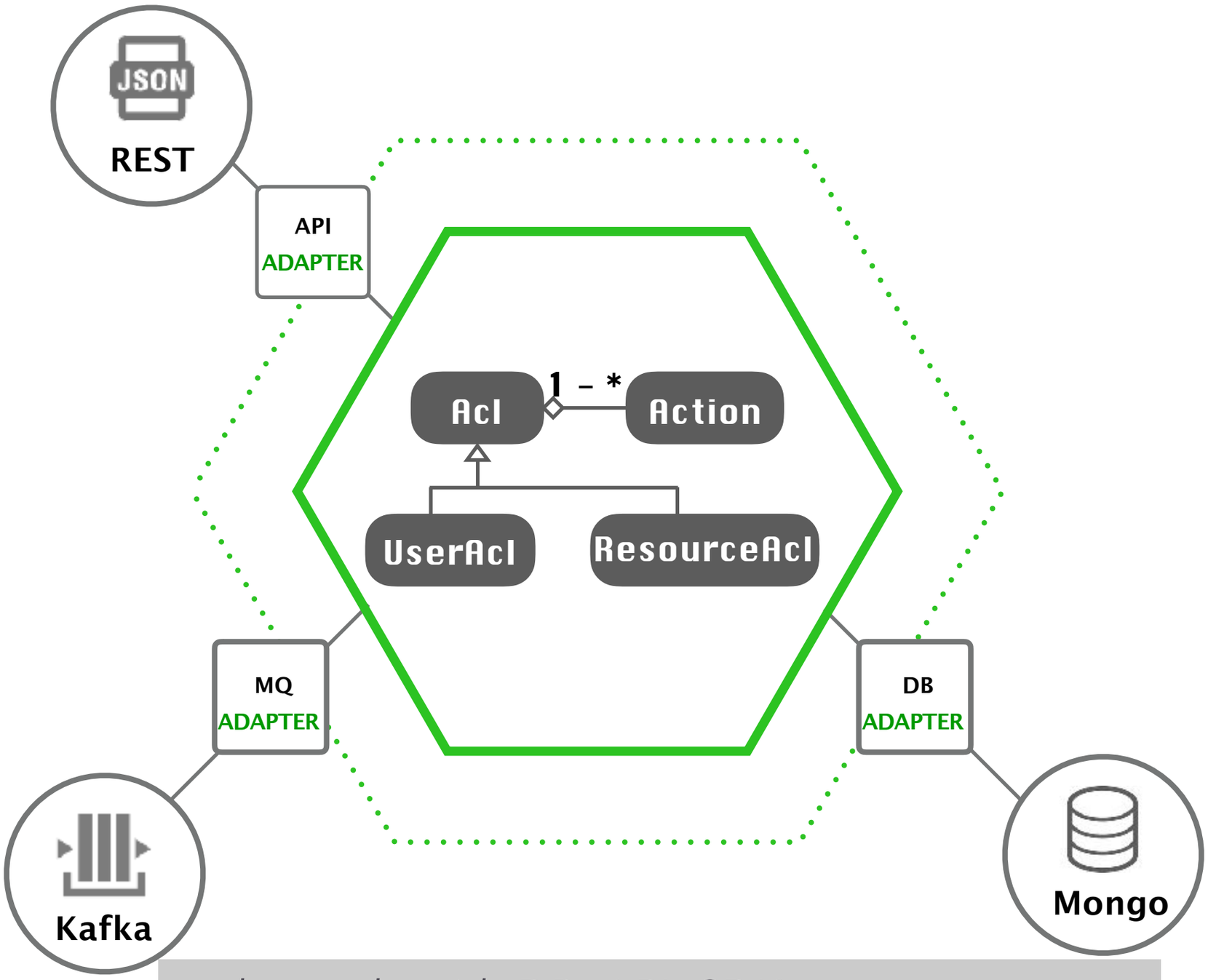



- *One size does not fit all*
- *Interface isolate principle*
- *Use facade pattern to convenient different users*
- *SYNC vs ASYNC*
- *P2P vs PUB/SUB*
- *REST is not the only choice*
- *Postel principle*
- *Idempotent design*
- *Semantic version*
- *…*

# AWS S3 API

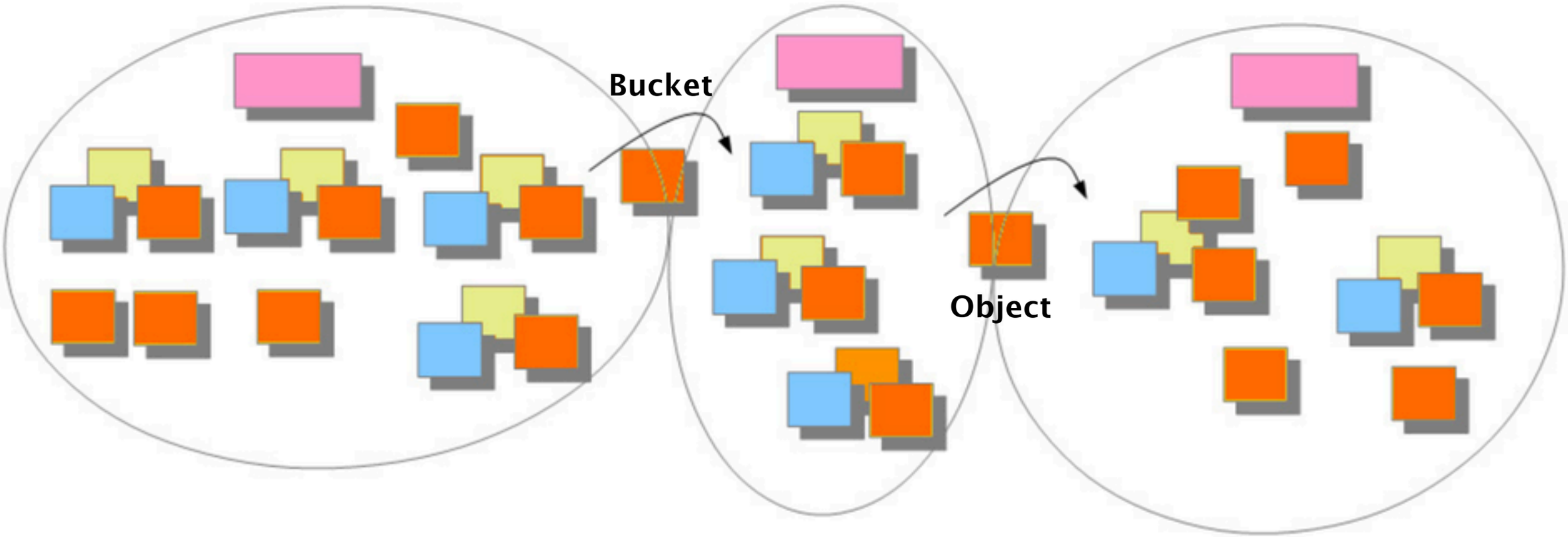
- Operations on Buckets
  - DELETE Bucket
  - DELETE Bucket analytics
  - DELETE Bucket cors
  - DELETE Bucket inventory
  - DELETE Bucket lifecycle
  - DELETE Bucket metrics
  - DELETE Bucket policy
  - DELETE Bucket replication
  - DELETE Bucket tagging
  - DELETE Bucket website
  - GET Bucket (List Objects) Version 2
  - GET Bucket accelerate
  - GET Bucket acl
  - GET Bucket analytics
  - GET Bucket cors
  - GET Bucket inventory
  - GET Bucket lifecycle
  - GET Bucket location
  - GET Bucket logging
  - GET Bucket metrics
  - GET Bucket notification
  - GET Bucket Object versions
  - GET Bucket policy
  - GET Bucket replication
  - GET Bucket requestPayment
  - GET Bucket tagging
  - GET Bucket versioning
  - GET Bucket website
- HEAD Bucket
- List Bucket Analytics Configurations
- List Bucket Inventory Configurations
- List Bucket Metrics Configurations
- List Multipart Uploads
- PUT Bucket
- PUT Bucket accelerate
- PUT Bucket acl
- PUT Bucket analytics
- PUT Bucket cors
- PUT Bucket inventory
- PUT Bucket lifecycle
- PUT Bucket logging
- PUT Bucket metrics
- PUT Bucket notification
- PUT Bucket policy
- PUT Bucket replication
- PUT Bucket requestPayment
- PUT Bucket tagging
- PUT Bucket versioning
- PUT Bucket website
- Operations on Objects
  - Delete Multiple Objects
  - DELETE Object
  - DELETE Object tagging
  - GET Object
  - GET Object ACL
  - GET Object tagging
  - GET Object torrent
  - HEAD Object
  - OPTIONS object
  - POST Object
  - POST Object restore
  - PUT Object
  - PUT Object - Copy
  - PUT Object acl
  - PUT Object tagging
  - Abort Multipart Upload
  - Complete Multipart Upload
  - Initiate Multipart Upload
  - List Parts
  - Upload Part
  - Upload Part - Copy


# DOMAIN MODEL IS CRITICAL



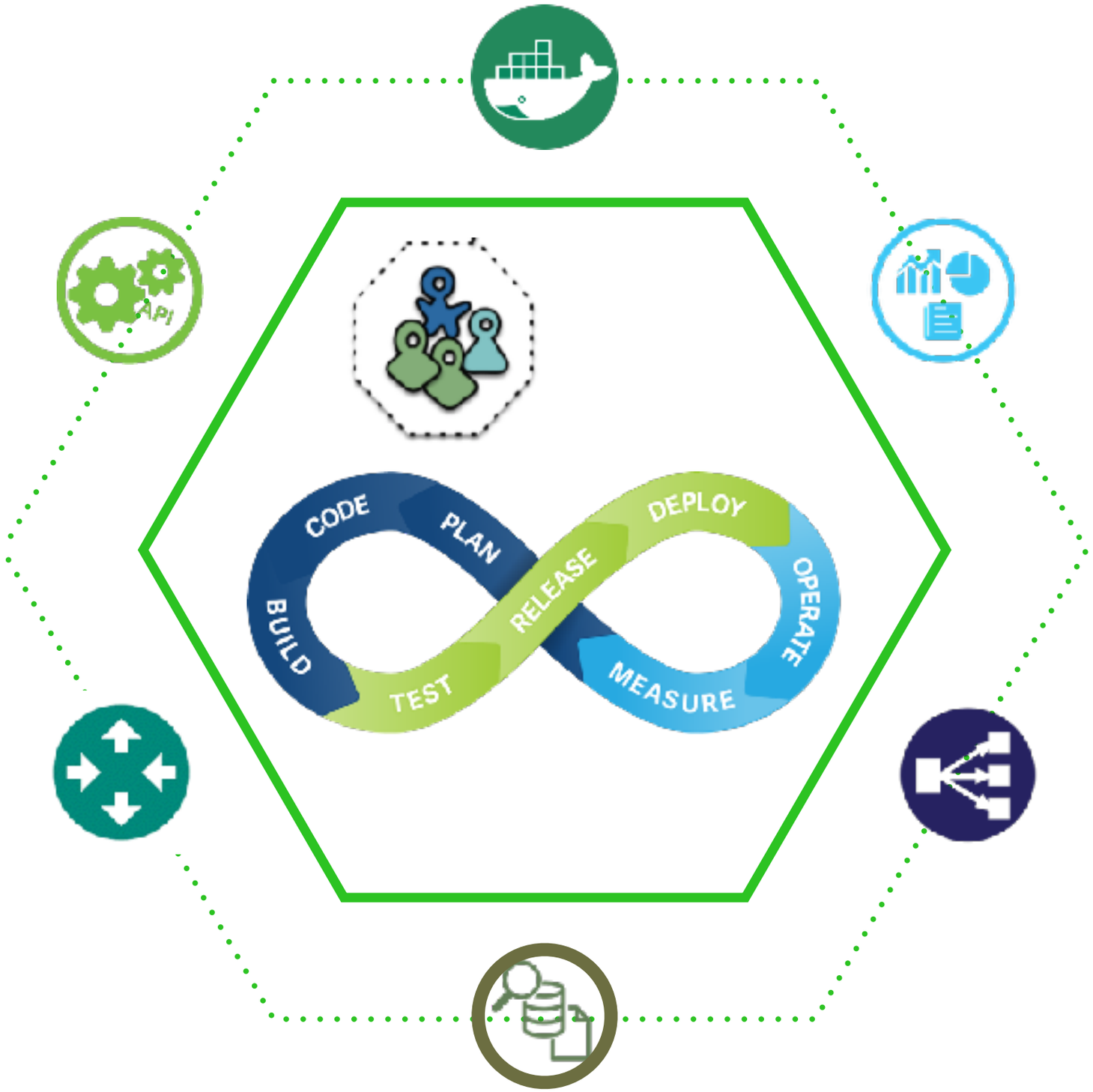

- *decouple with DB, API, SDK...*
- *prevent from anemia model (TDA)*
- *reuse between microservices carefully (RoR)*

# USE DOMAIN EVENT TO SHARE MODEL



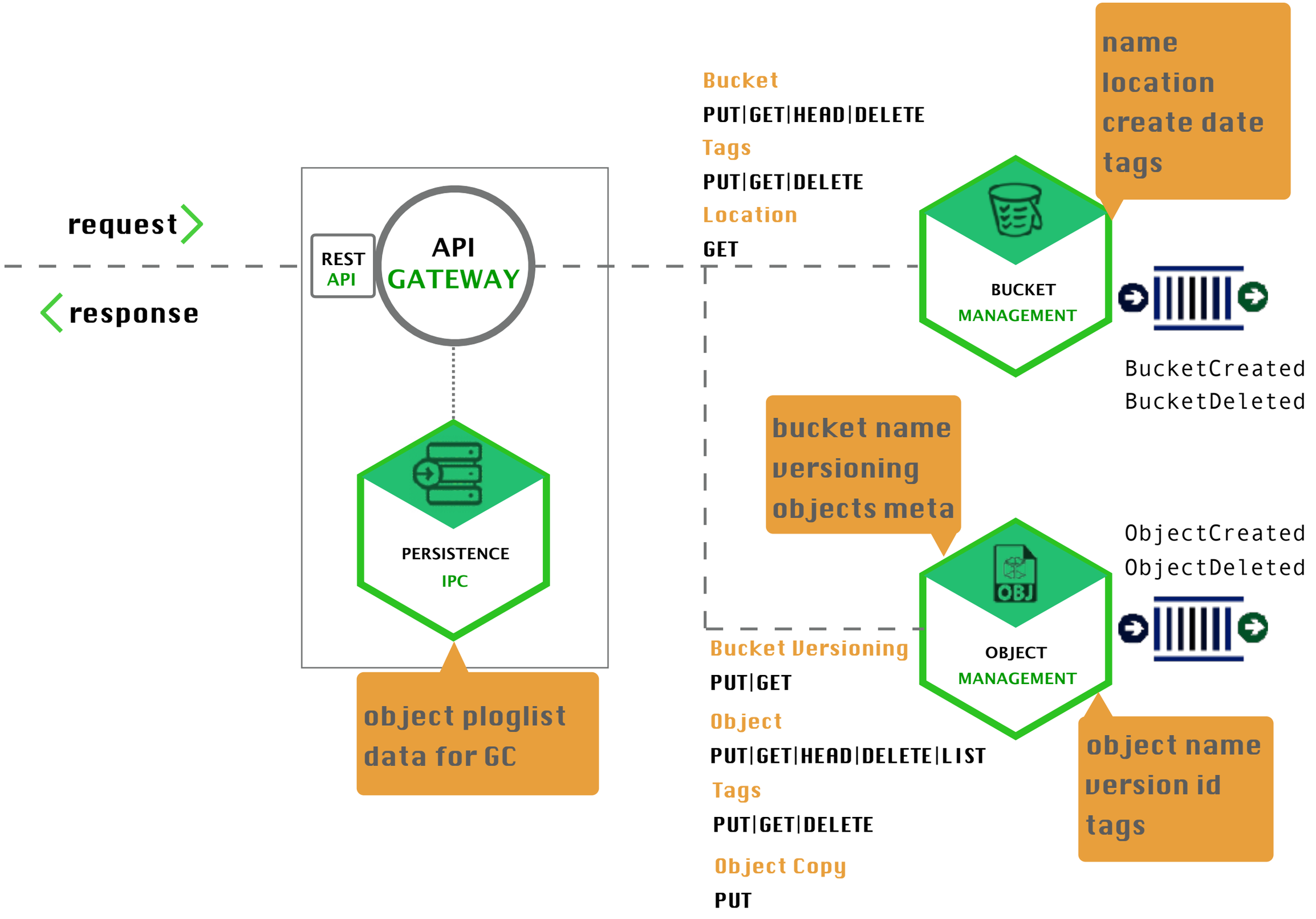

- *communicating by domain events*
- *separate command and query*

# INFRASTRUCTURE SUPPORT

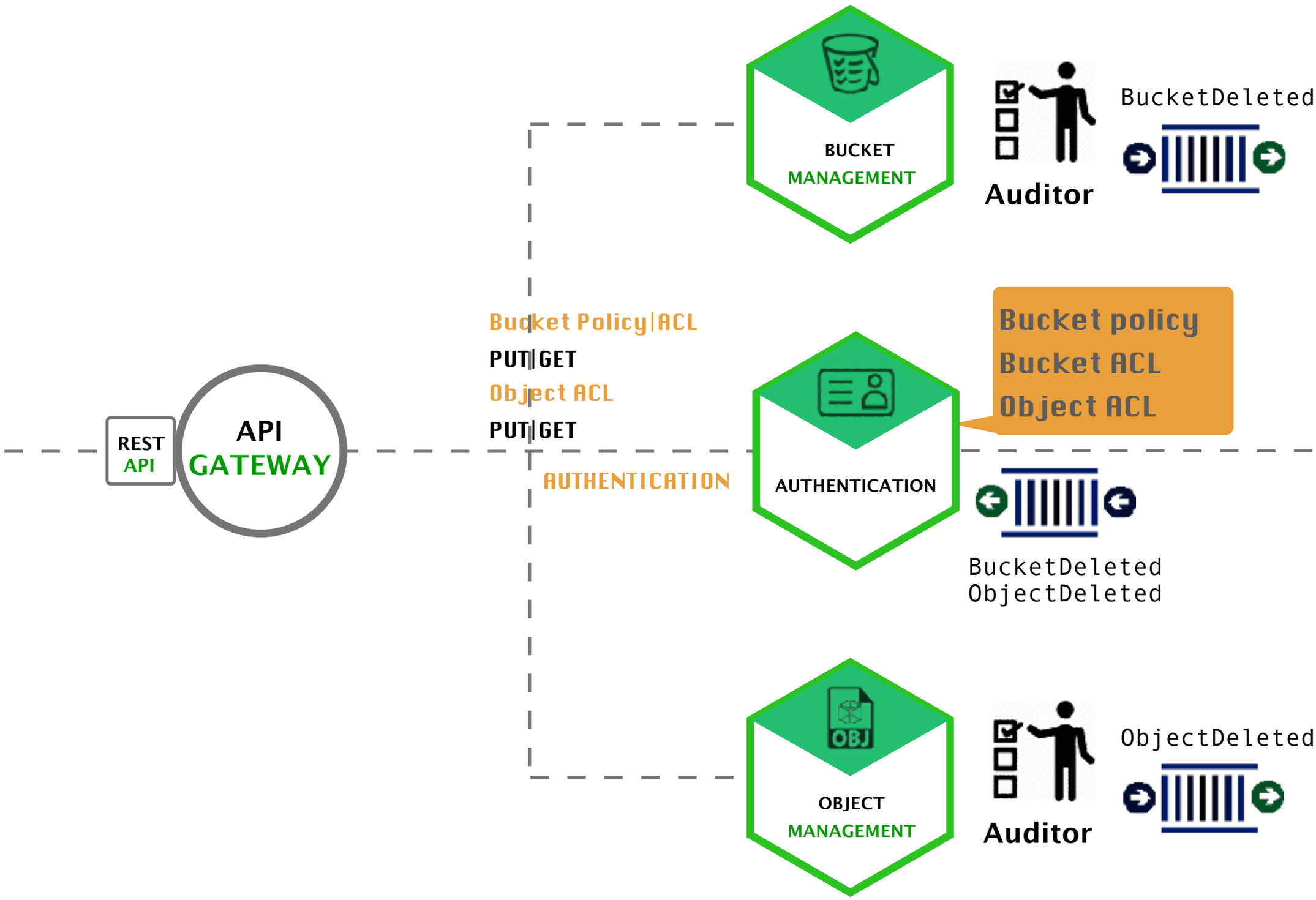


# Architecture Proposal

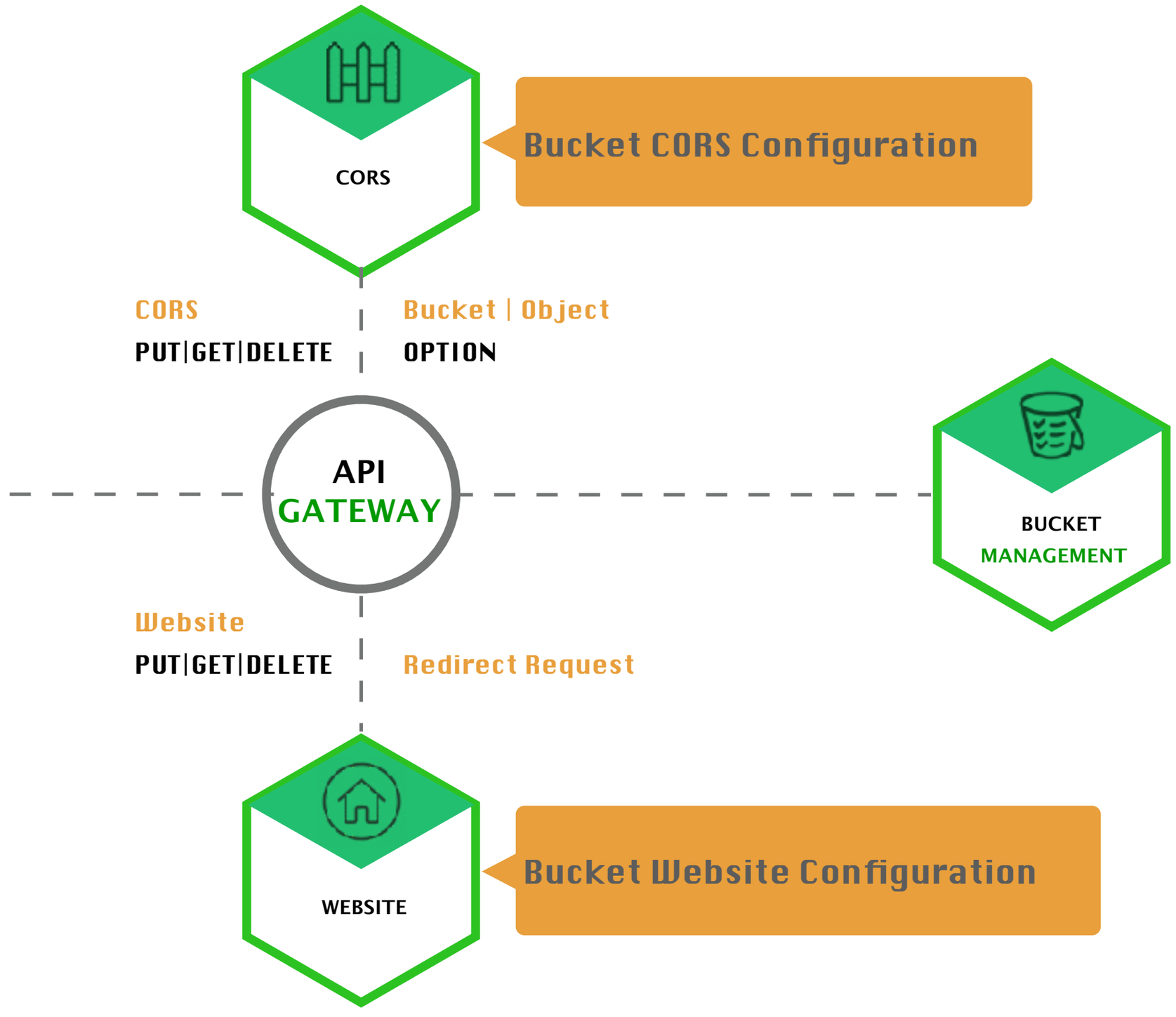
# BASE SKETCH


request
response
REST API
API GATEWAY
PERSISTENCE
IPC
object ploglist data for GC
Bucket
PUT|GET|HEAD|DELETE
Tags
PUT|GET|DELETE
Location
GET
name
location
create date
tags
BUCKET MANAGEMENT
BucketCreated
BucketDeleted
bucket name
versioning
objects meta
OBJ
OBJECT MANAGEMENT
ObjectCreated
ObjectDeleted
Bucket Versioning
PUT|GET
Object
PUT|GET|HEAD|DELETE|LIST
Tags
PUT|GET|DELETE
Object Copy
PUT
object name
version id
tags

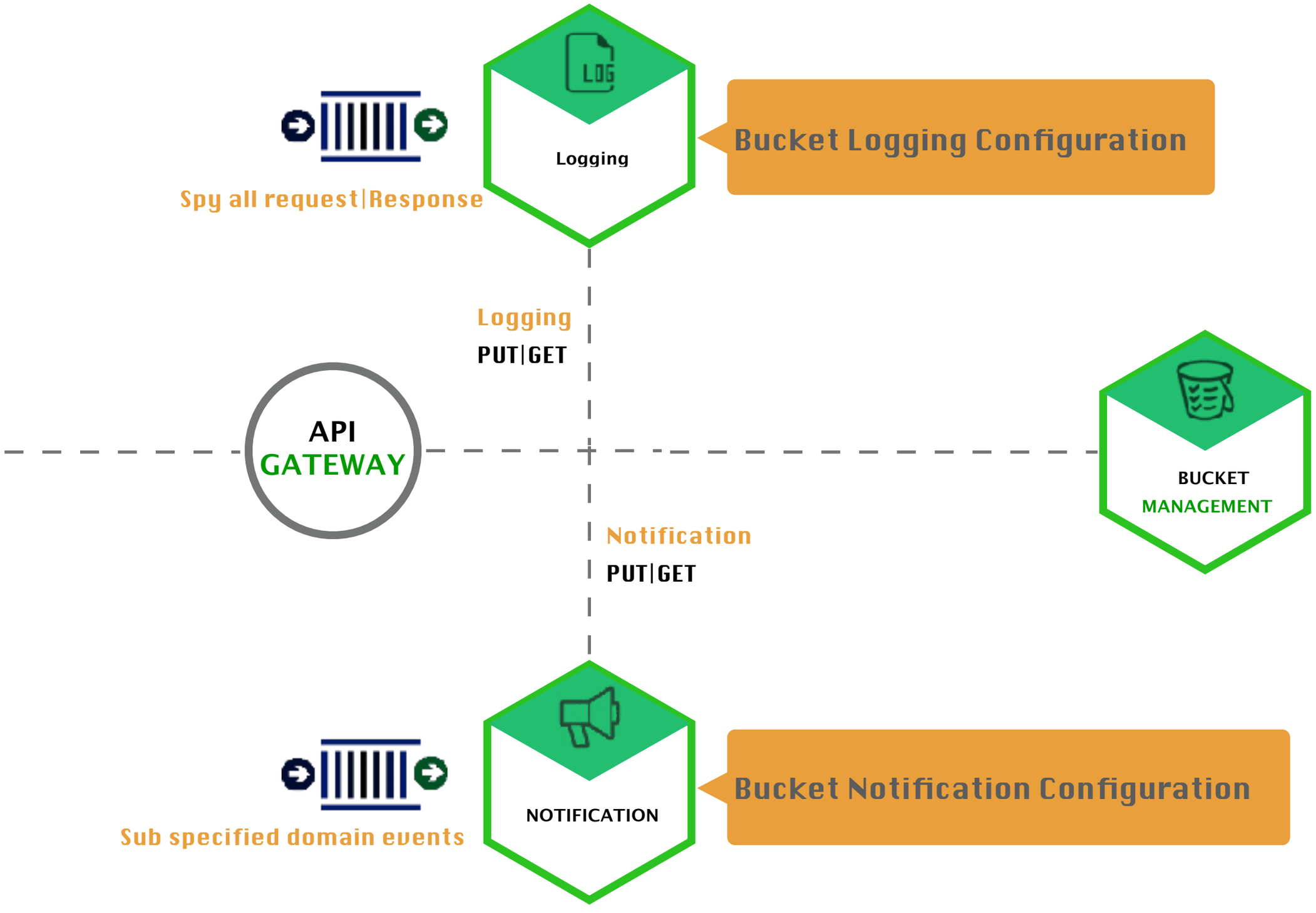
# AUTHENTICATION



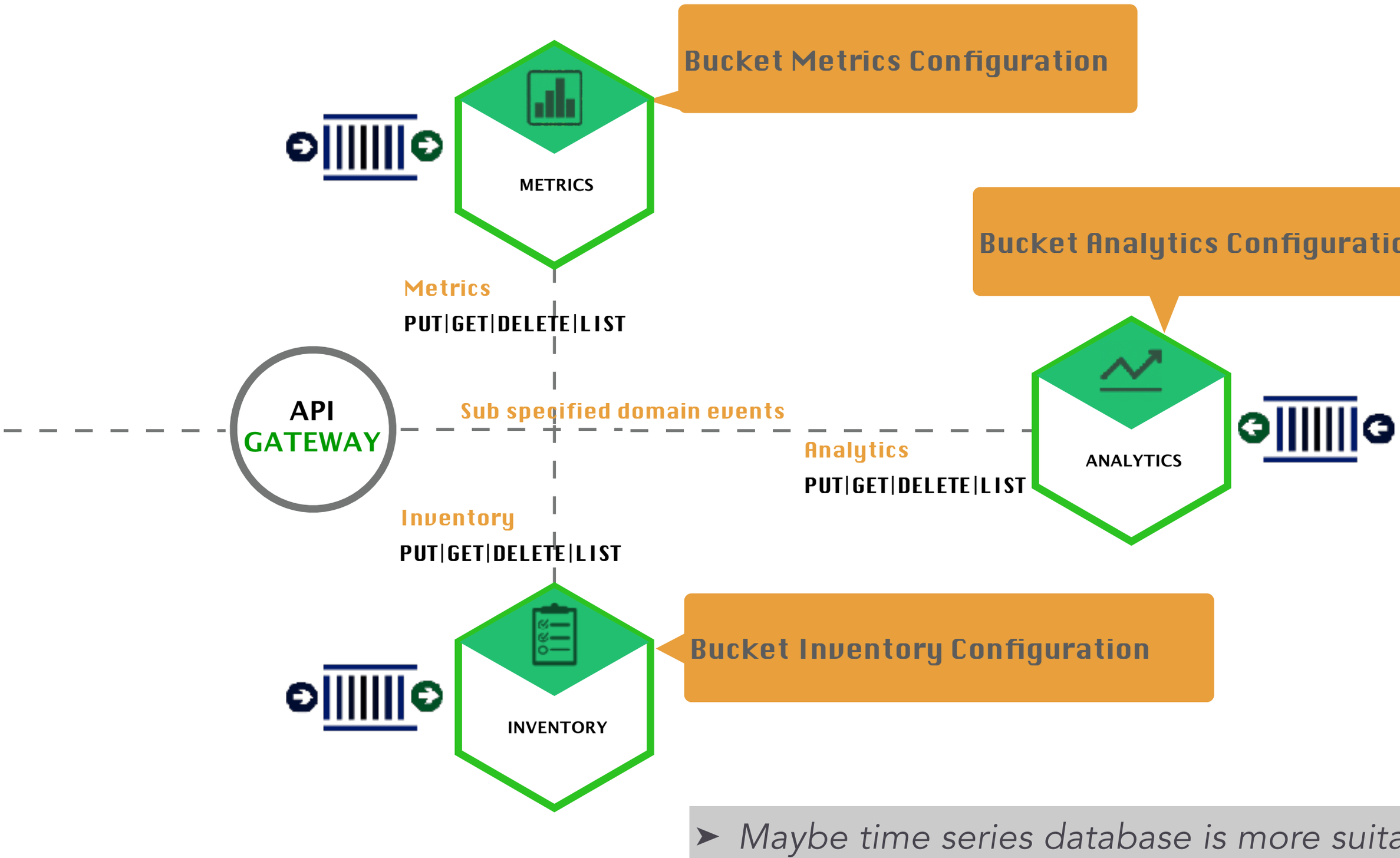
# WEBSITE AND CORS



# LOGGING AND NOTIFICATION

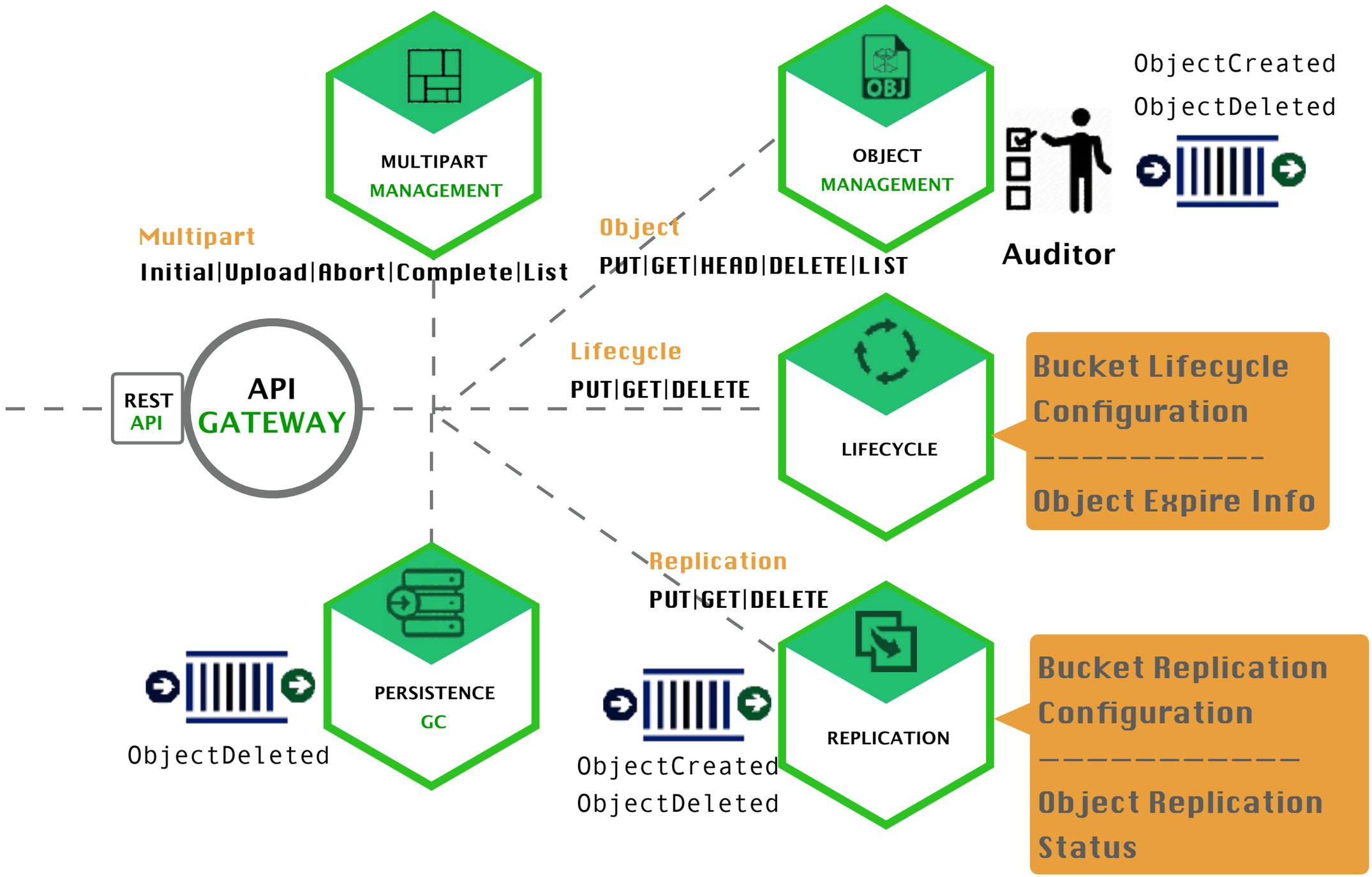


# STATISTICS AND ANALYTICS



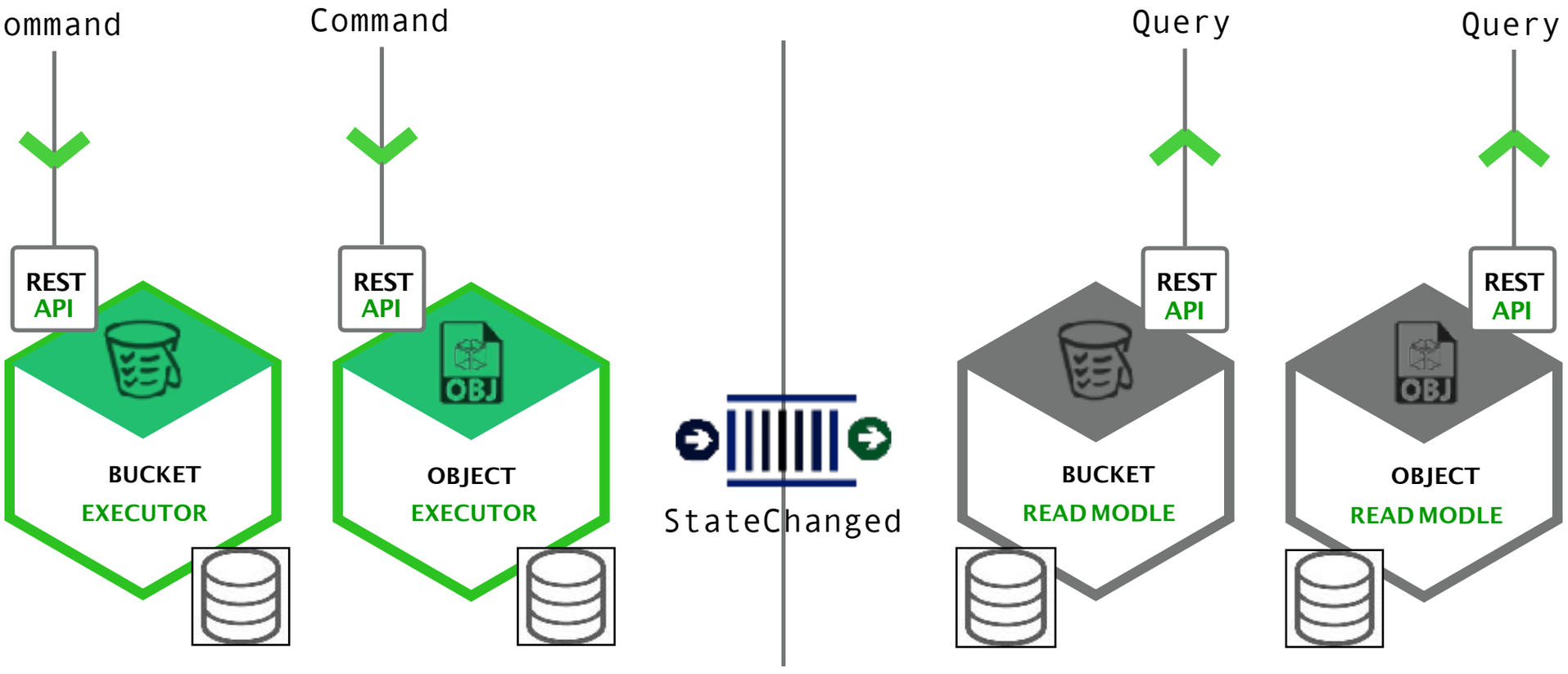

➤ *Maybe time series database is more suitable*

# OBJECT

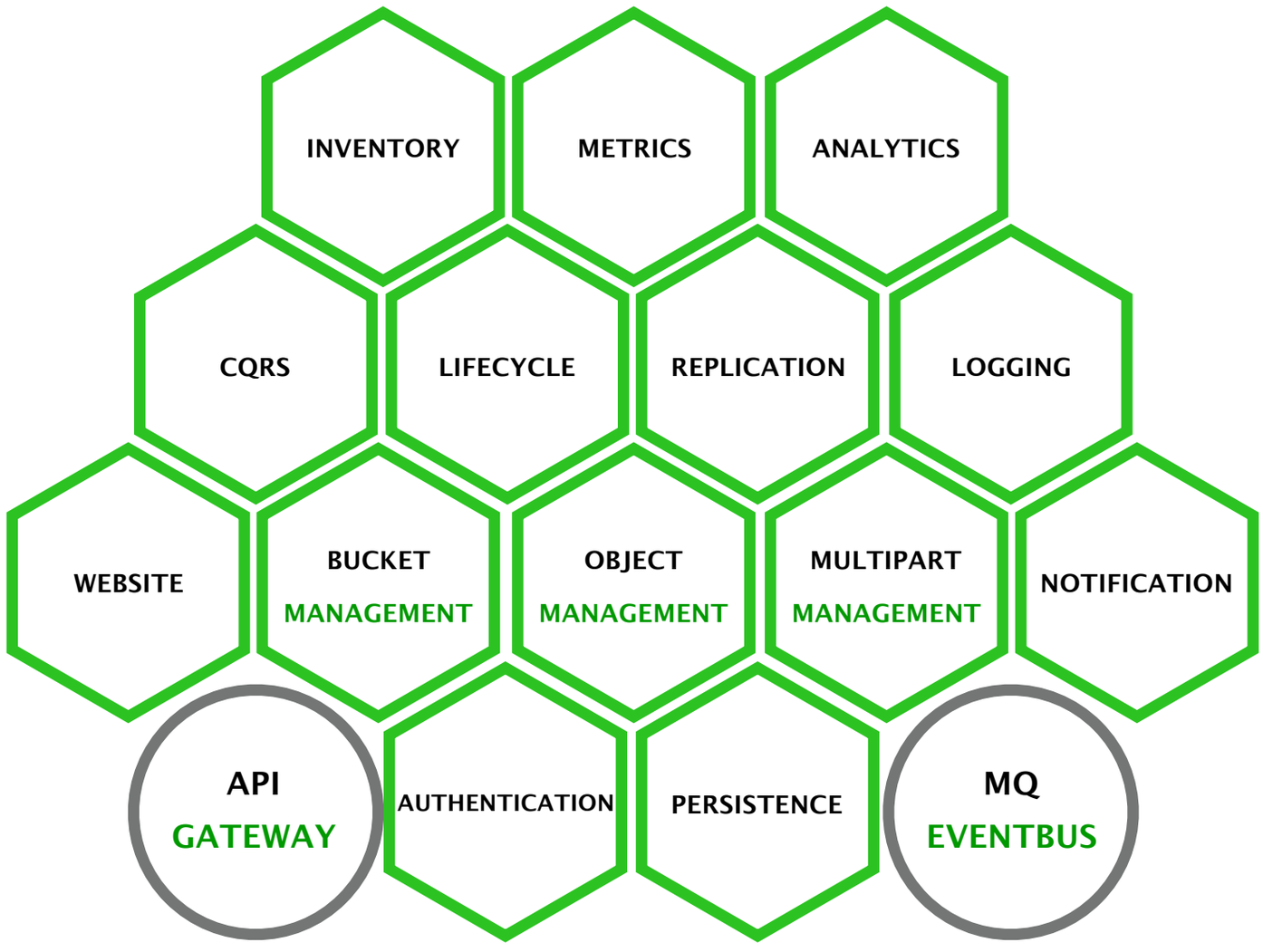



➤ *Composite design: service level reuse*

# EVOLVE TO CQRS

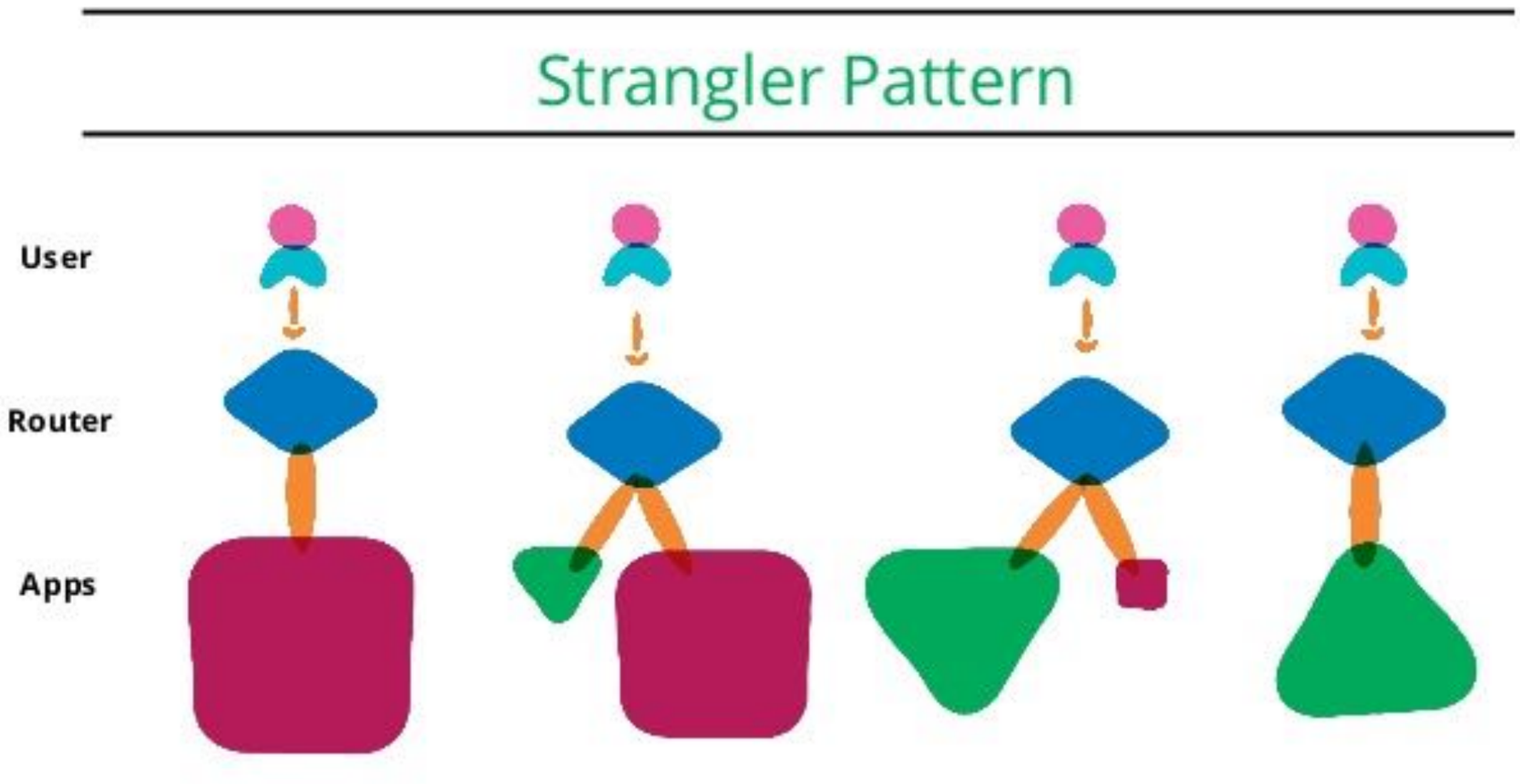


# OVERVIEW


INVENTORY
METRICS
ANALYTICS
CQRS
LIFECYCLE
REPLICATION
LOGGING
WEBSITE
BUCKET
MANAGEMENT
OBJECT
MANAGEMENT
MULTIPART
MANAGEMENT
NOTIFICATION
API
GATEWAY
AUTHENTICATION
PERSISTENCE
MQ
EVENTBUS

# Evolution Suggestions

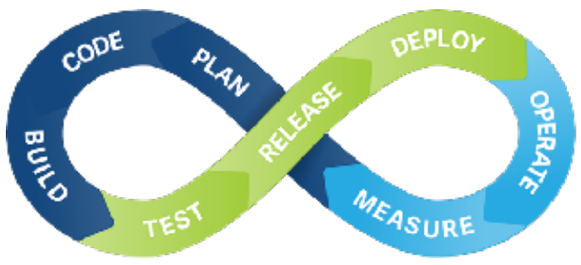
# EVOLUTION WAYS




➤ *Visualize the whole process, statistics can tell you more!*

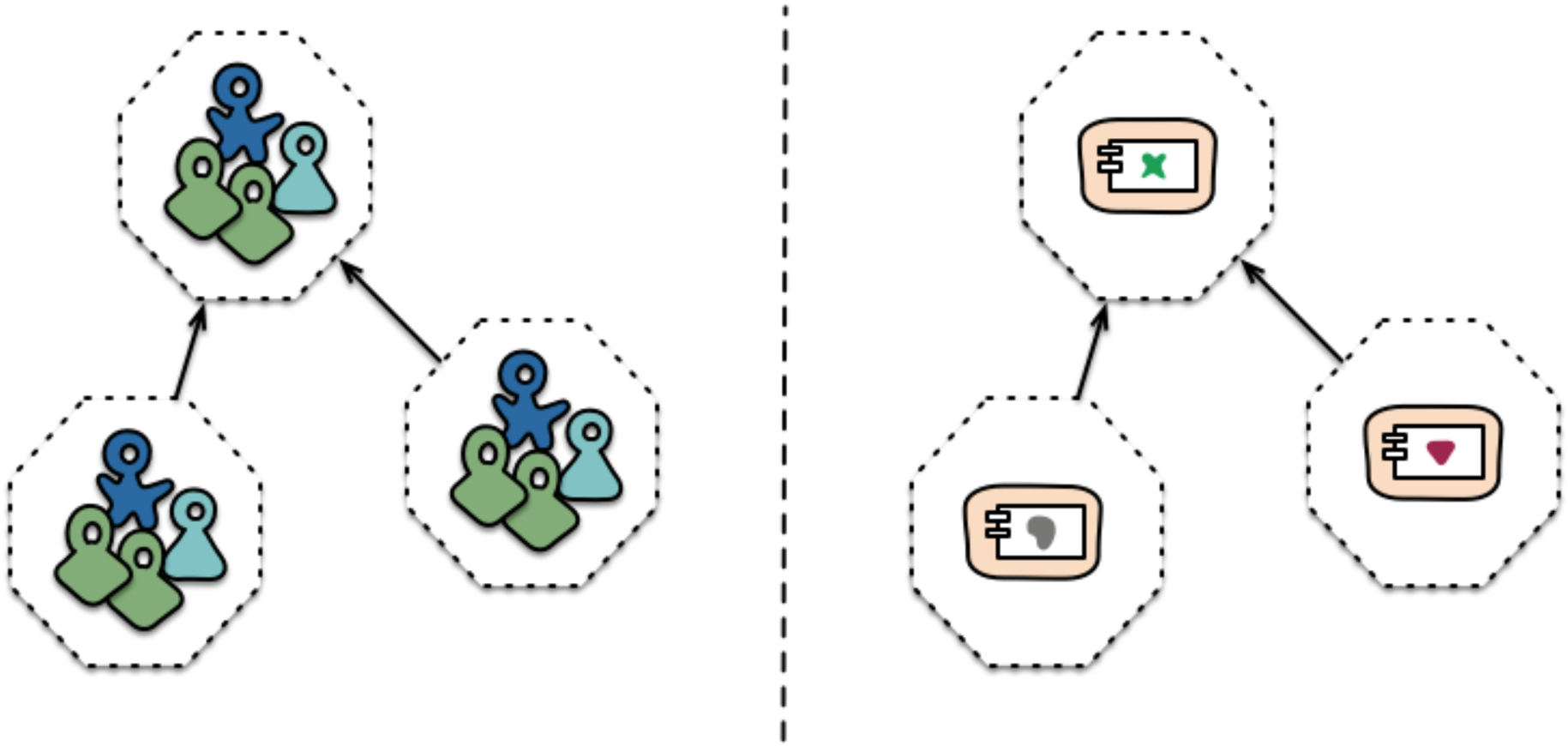
# IMPROVEMENTS

- Skills of design, coding and test
  - Domain Driven Design
  - Orthogonal design
  - TDD, Refactoring…




- Process on continuous delivery pipeline
  - Customer driven contract test
  - integrating speed

# ADAPTIVE ORGANIZATION STRUCTURE



Cross-functional teams...

... organised around capabilities
Because Conway's Law

# Questions?

ThoughtWorks

# THANKS

王博
e.wangbo@gmail.com