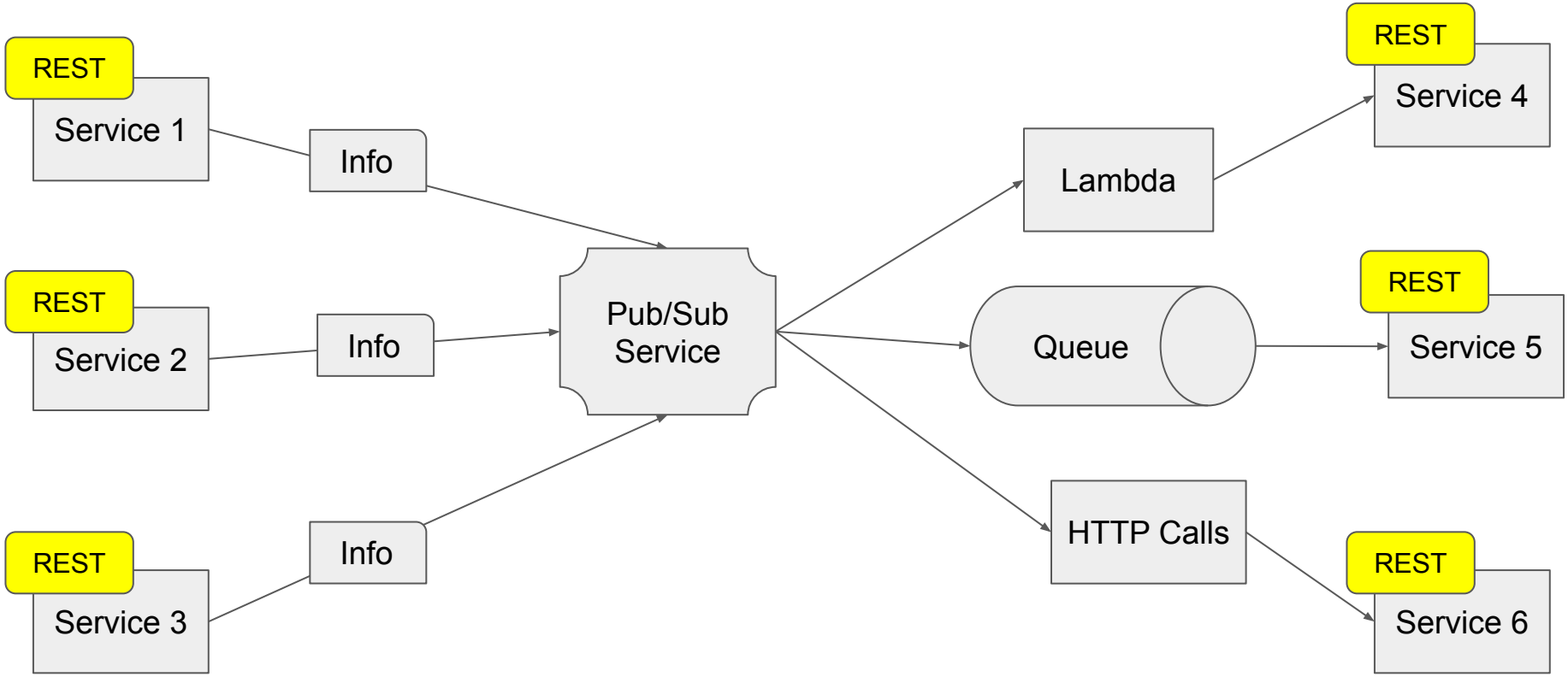
Moglix

Architecture & API Guidelines

Microservices

1. Single Responsibility Principle
2. Domain Driven Design
3. Independent Deployment of services
4. Synchronous Messaging - REST
5. Asynchronous Messaging - Pub/Sub and Queues
6. Message Format - JSON
7. Service Contract(Documentation) - Swagger
8. Interservice Communication - API Gateway
9. Decentralized Data Management
10. Decentralized Governance
11. Server-side Service Discovery & Registry
12. Security
13. Transactions - compensating operations
14. Design for failure
15. Infrastructure Automation - Docker, Kubernetes, OpsWorks

Inter-service Communication



API Design Guidelines

1. Microservices should expose their data and functionality through APIs only.
2. No database calls between services
3. REST API Design
4. A resource URI should be **noun** not **verb**.
    a. GET /products # returns a list of products
    b. GET /products/12 # returns a product with primary key 12
    c. POST /products # creates a new product
    d. PUT /products/12 # updates a product with primary key 12
    e. DELETE /products/12 # deletes a product with primary key 12
5. Use plurals - prefer /products over /product.
6. Use named parameters - /products/?name=duster instead of /getProductByName
7. Always use versioning for your API - /v1/products, /v2/products
8. List APIs should always be paginated - use limit & offset
9. Allow query parameters for advanced filtering & sorting - /orders?status=open&sort=-priority
10. Prefered return format is JSON. XML can also be supported, if necessary.

API Design Guidelines

11. Use camelCase naming for C/Java, snake_case for python and ruby
12. Allow clients to specify page sizes but put a max-page size limit in API
13. Send the max-page size in header or in body
14. Versioning Policy
    a. **Changes that don't require new version**
        i. New resources
        ii. New HTTP methods on existing resources
        iii. New data formats
        iv. New attributes or elements on existing data types
    b. **Changes that require a new version**
        i. Removed or renamed URIs
        ii. Different data returned from the same URI
        iii. Removal of support for HTTP methods on existing URIs

API Design Guidelines

15. API Security
    a. **Always use HTTPS**
    b. **Authentication**
        i. Every REST API must at least accept Basic Authentication
        ii. By default, access to all resources should require the client to be authenticated
        iii. Use JWT authentication by default
    c. **Authorization**
        i. Single Authorization service is to be used for authorization.
        ii. REST API should not be responsible be handling authorization
16. Use proper error & success message code
17. Preferably every API should return these two keys “code” - {200,500,...} & “success” {boolean}
18. Throw exceptions liberally
19. APIs should return proper HTTP codes.

API Design Guidelines

20. HTTP Response Codes and Usage
    a. **2xx (Success Category)**
        i. 200 Ok - Standard success response for REST API
        ii. 201 Created - Response when a new instance is created
        iii. 202 Accepted - Request has been received but cannot be processed in realtime - e.g batch processing job or schedule jobs.
        iv. 204 No Content - Request is successful but no response has been generated . e.g. DELETE /products/12.
    b. **3xx (Redirection Category)**
        i. 301 Moved Permanently - The resource has been moved to another location
        ii. 302 Moved Temporarily - The resource has been moved to another location temporarily
        iii. 304 Not Modified - The requested resource has not been modified. The client's cached version is still valid.

API Design Guidelines

20. HTTP Response Codes and Usage
    c. **4xx (Client Error Category)**
        i. 401 Unauthorized - Client is un-authenticated and not allowed to access resource and should re-request with required credentials
        ii. 403 Forbidden - Client is authenticated but the client is not allowed to access this resource
        iii. 404 not Found - Resource is not available currently.
        iv. 401 Gone - Resource has been removed and is no longer available.
    d. **5xx (Server Error Category)**
        i. 500 Internal Server Error - The request is valid but server has faced some error.
        ii. 503 Service Unavailable - Server is down or unavailable to process requests.
21. Use proper error & success message code
22. Preferably every API should return these two keys “code” - {200,500,...} & “success” {boolean}
23. Throw exceptions liberally
24. APIs should return proper HTTP codes.

API Design Guidelines

25. Code for errors
26. Handle timeouts properly
27. Validate data proactively
28. Honor Caching
29. Throttle Performance

Supply Chain Server Architecture - for reference