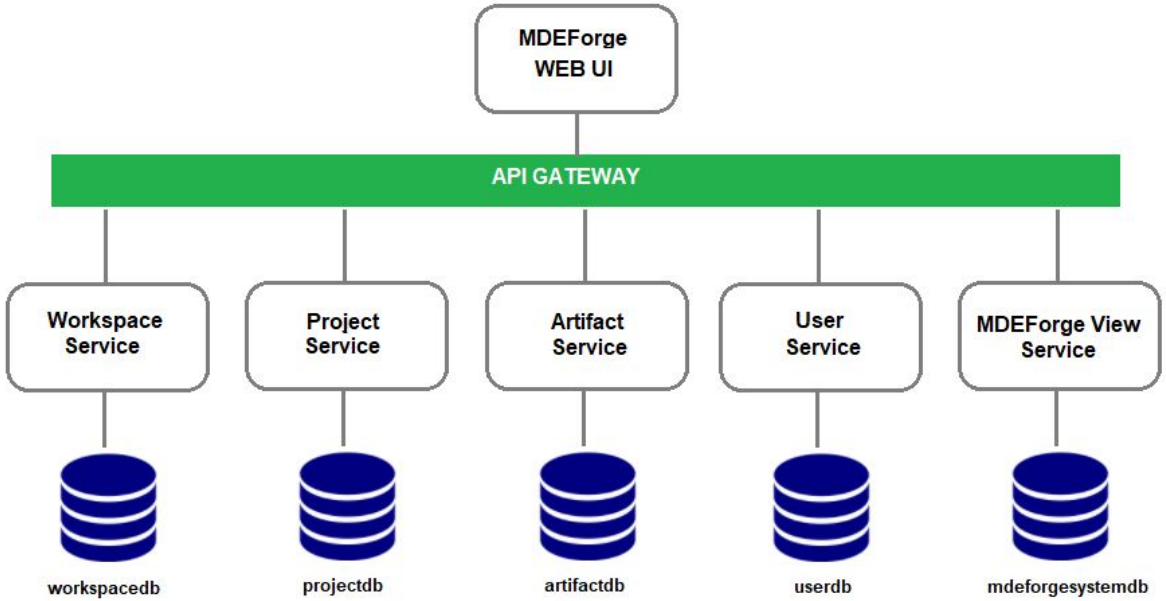
MDEForge Platform based on Microservices Architecture

Carlos Avendaño

MDEForge Platform - Architecture



Pattern: Database Per Service

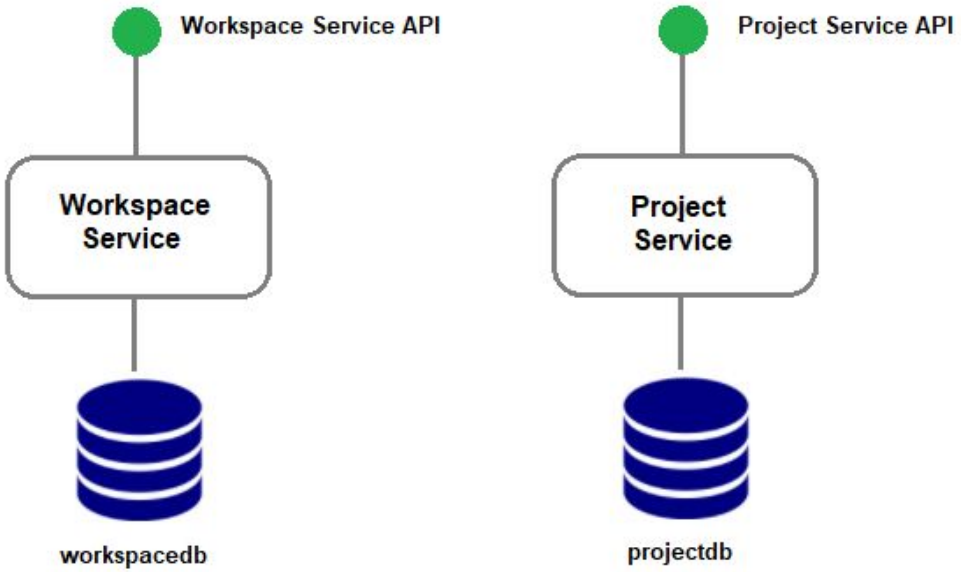
Services must be loosely coupled so that they can be developed, deployed and scaled independently.

Databases must sometimes be replicated in order to scale. See the Scale Cube.

Different services have different data storage requirements. (MongoDB, Mysql, Oracle and so forth)

http://microservices.io/patterns/data/database-per-service.html

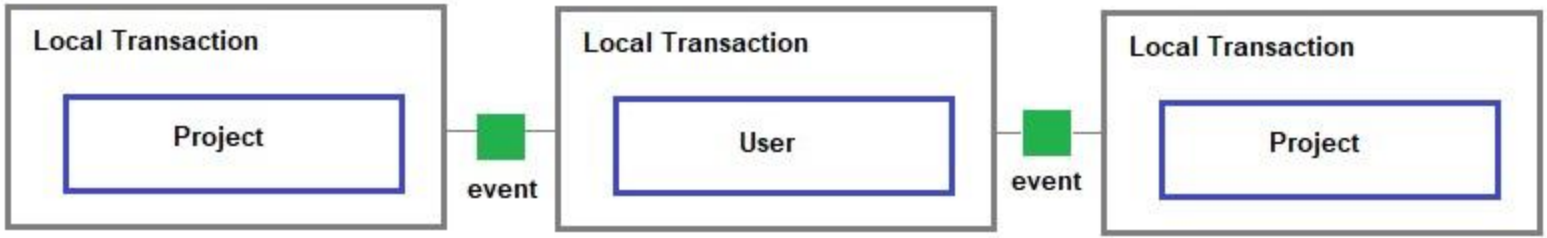
Pattern: Database Per Service



How to maintain data consistency?

Pattern: Saga

Some business transactions span multiple services so we need a mechanism to ensure **data consistency** across services.



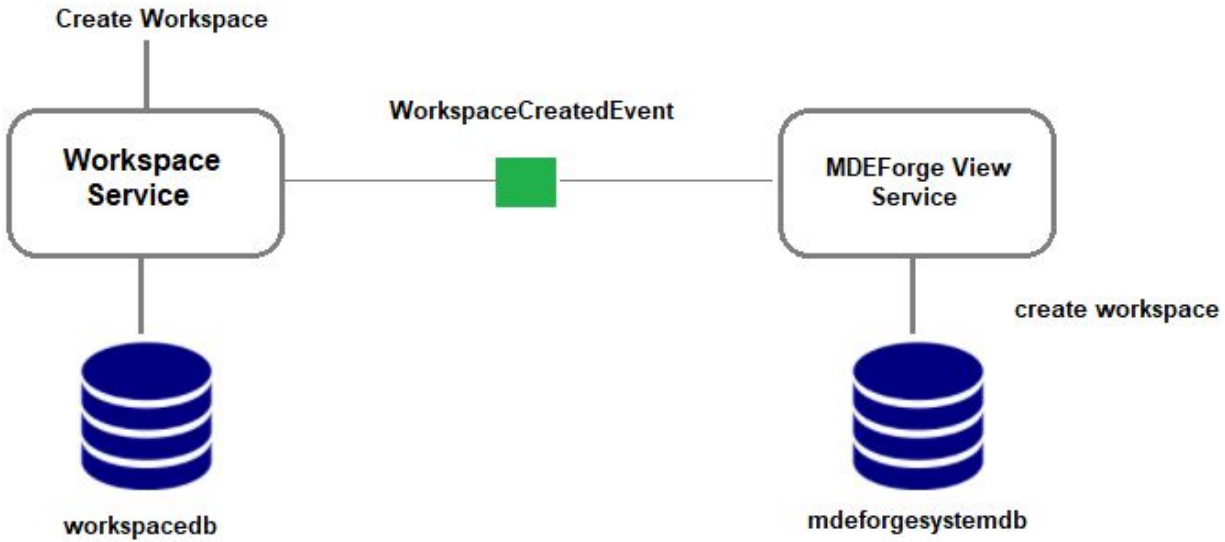
http://microservices.io/patterns/data/saga.html

Saga

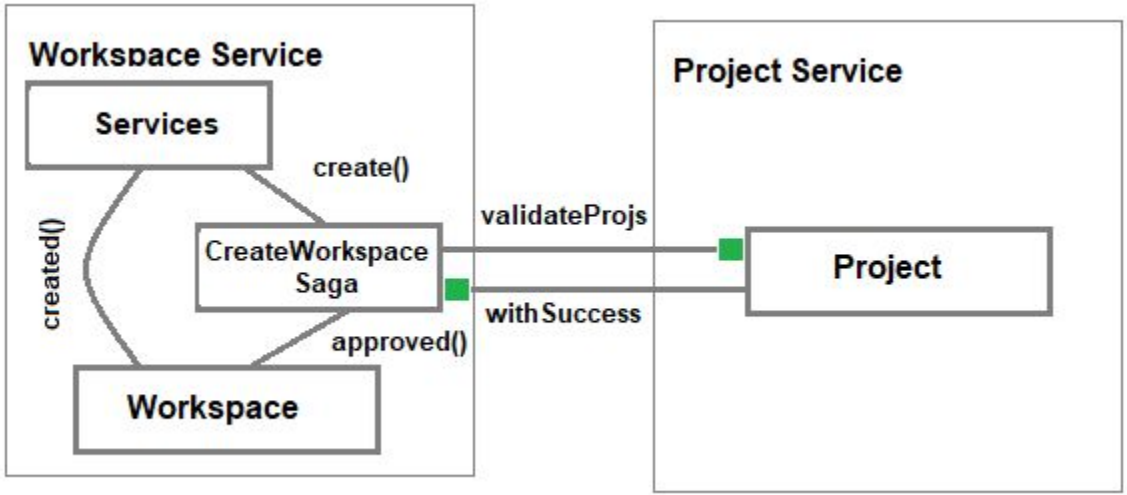
There are two ways of coordination sagas:

1. **Choreography**: each local transaction publishes domain events that trigger local transactions in other services.
2. **Orchestration**: an orchestrator (object) tells participants what local transactions to execute.

Saga - Choreography example



Saga - Orchestration example



Saga Implementations

Narayana LRA (Long Running Actions)

Axon Framework

Eventuate

} CQRS principles

https://jbossts.blogspot.it/2017/12/saga-implementations-comparison.html?m=1

Eventuate TRAM Framework

Messaging: send and receive messages over named channels.

Events: publish domain events and subscribe to domain events.

Commands: asynchronously send a command to a service and receive a reply

https://github.com/eventuate-tram/eventuate-tram-core

Eventuate TRAM in MDEForge Application

User Service

- Command Handlers (receiving and sending messages by a channel)
- Controller
- DAO
- Implementation (publishing events)
- Model

Eventuate TRAM in MDEForge Application

{Workspace, Project, Artifact} Service

- Command Handlers (receiving and sending messages by a channel)
- Controller
- DAO
- Implementation (publishing events)
- Model
- Saga (sending commands to a services and receiving replies)

Eventuate TRAM in MDEForge Application

MDEForge View Service

- Controller
- DAO
- Implementation (**NO** publishing events)
- Messaging (**Event Handlers**)
- Model
- Repository

Eventuate TRAM in MDEForge Application

Each service contains an API where they expose their Commands, Events and Info and those are visible for all services. So the services communicate to each other using their API.

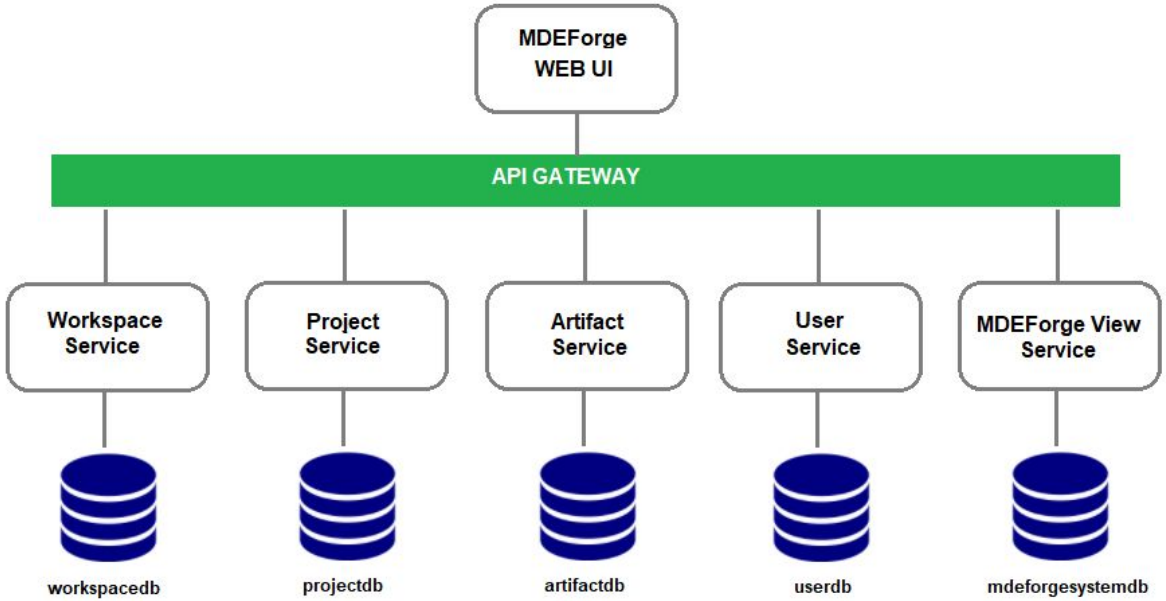
Decomposition

Decompose by business capability

Decompose by subdomain (Domain Driven Design)

http://microservices.io/patterns/decomposition/decompose-by-business-capability.html

MDEForge Platform - Architecture



MDEForge & Eventuate TRAM- Requirements

- Each service must define one or more domain events.
- A service has one or more operations.
- An operation can be either an aggregation or a query.
- Each aggregation operation must publish an event.
- If an aggregation operation involves at least other service it must execute a Saga.
- Each step of the Saga uses a service command exposed in a service API.
- Each service that exposes an API (Commands, Events, Info) must define a Command Handlers.

MDEForge & Eventuate TRAM- Requirements

- A Command Handle defines a channel to listen arriving messages.
- A Command Handle must define a method for each message where the logic part will be done.
- A method must return either withSuccess if everything is good or withFailure otherwise.
- If a method involves an aggregation operation it must publish an event as well.
- A service may handle domain events.

Saga Characteristics

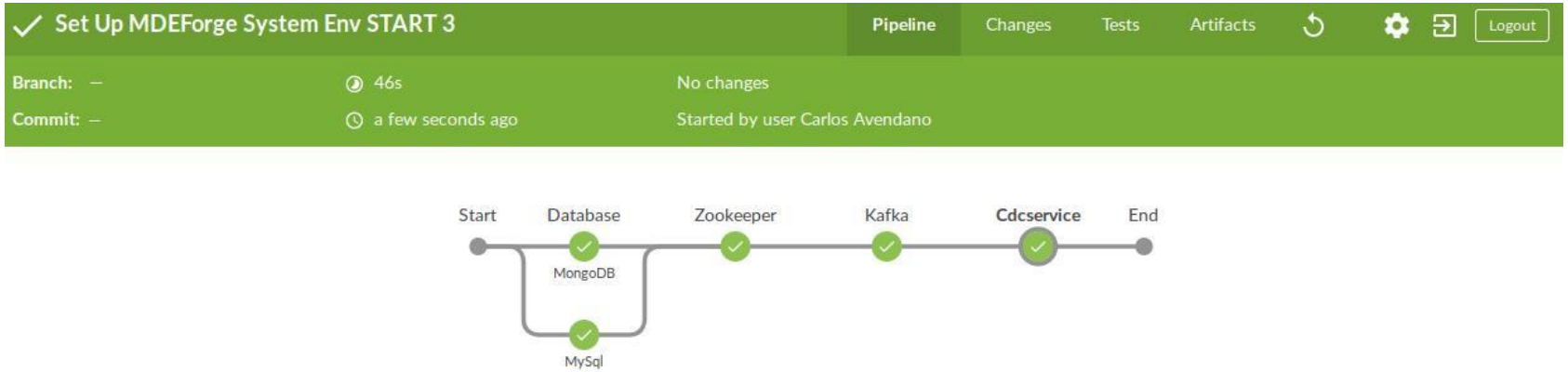
- A step of a Saga can invoke a service through a Command.
- A step of a Saga can get a reply form a invoked service.
- A step of a Saga can use withCompensation method to undo all operations in case of failure.

Building and Deployment?

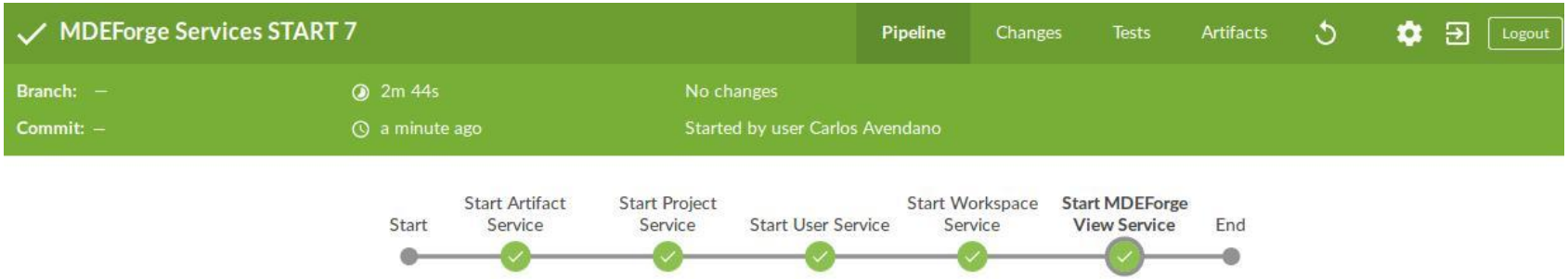
Docker Compose

```yaml
version: '3'

services:

  artifact-service:
    image: mdeforge/artifact-service
    container_name: artifact-service
    ports:
      - 5005:8080
    links:
      - mysql
      - kafka
      - zookeeper
      - cdcservice
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://mysql/eventuate
      SPRING_DATASOURCE_USERNAME: mysqluser
      SPRING_DATASOURCE_PASSWORD: mysqlpw
      SPRING_DATASOURCE_DRIVER_CLASS_NAME: com.mysql.jdbc.Driver
      SPRING_DATASOURCE_TIMEOUT: 10000
      EVENTUATELOCAL_KAFKA_BOOTSTRAP_SERVERS: kafka:9092
      EVENTUATELOCAL_ZOOKEEPER_CONNECTION_STRING: zookeeper:2181
      MONGODB: mongodb:27017/artifactdb
```

Jenkins BlueOcean Pipeline - Start Environment



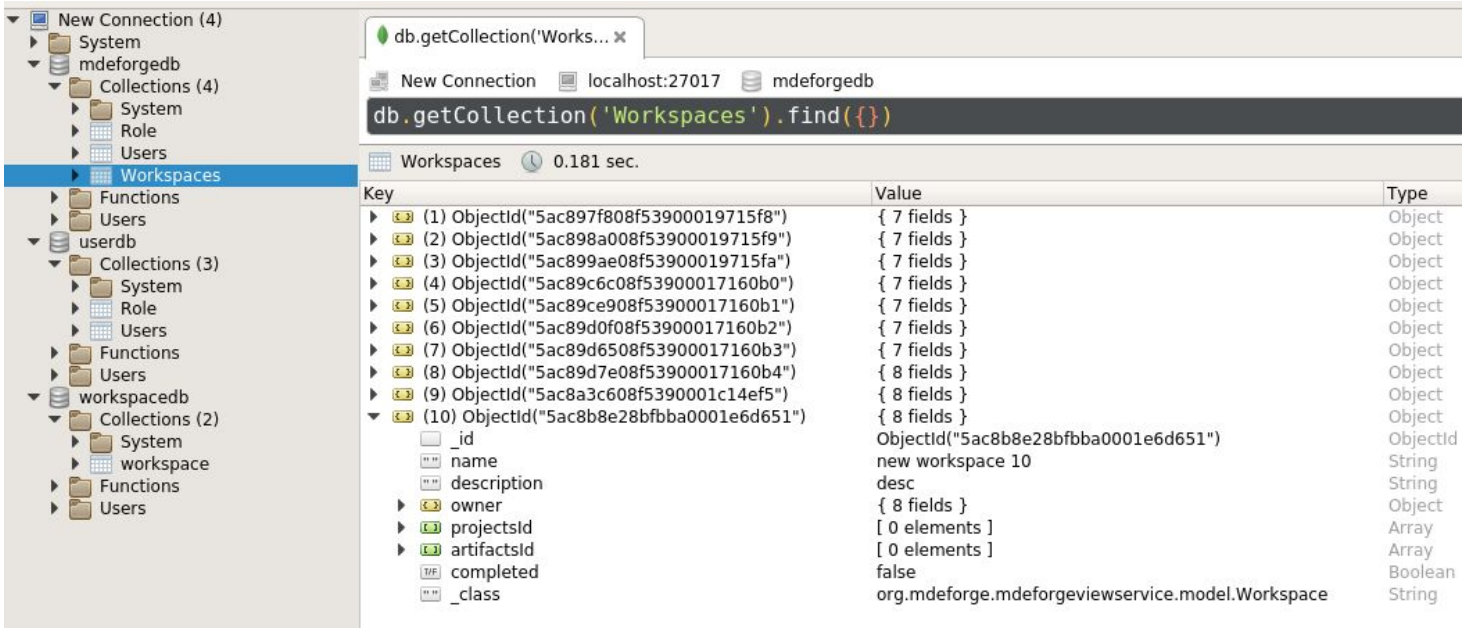
Jenkins BlueOcean Pipeline - Start Services



Swagger UI


swagger
default (/v2/api-docs)
api_key
Explore
Api Documentation
Api Documentation
Created by Contact Email
Apache 2.0
workspace-controller : Workspace Controller
Show/Hide
List Operations
Expand Operations
POST
/update/workspace/
updateWorkspace
POST
/workspace
createWorkspace
GET
/workspaces
findAllWorkspaces
GET
/workspaces/{workspaceId}
findWorkspace
[BASE URL: / , API VERSION: 1.0]

MongoDB


New Connection (4)
System
mdeforgedb
Collections (4)
System
Role
Users
Workspaces
Functions
Users
userdb
Collections (3)
System
Role
Users
Functions
Users
workspacedb
Collections (2)
System
workspace
Functions
Users
db.getCollection('Works...
New Connection
localhost:27017
mdeforgedb
db.getCollection('Workspaces').find({})
Workspaces
0.181 sec.
Key
Value
Type
(1) ObjectId("5ac897f808f53900019715f8")
{ 7 fields }
Object
(2) ObjectId("5ac898a008f53900019715f9")
{ 7 fields }
Object
(3) ObjectId("5ac899ae08f53900019715fa")
{ 7 fields }
Object
(4) ObjectId("5ac89c6c08f53900017160b0")
{ 7 fields }
Object
(5) ObjectId("5ac89ce908f53900017160b1")
{ 7 fields }
Object
(6) ObjectId("5ac89d0f08f53900017160b2")
{ 7 fields }
Object
(7) ObjectId("5ac89d6508f53900017160b3")
{ 7 fields }
Object
(8) ObjectId("5ac89d7e08f53900017160b4")
{ 8 fields }
Object
(9) ObjectId("5ac8a3c608f5390001c14ef5")
{ 8 fields }
Object
(10) ObjectId("5ac8b8e28bfbba0001e6d651")
{ 8 fields }
Object
_id
ObjectId("5ac8b8e28bfbba0001e6d651")
ObjectId
name
new workspace 10
String
description
desc
String
owner
{ 8 fields }
Object
projectsId
[0 elements]
Array
artifactsId
[0 elements]
Array
completed
false
Boolean
_class
org.mdeforge.mdeforgeviewservice.model.Workspace
String