
SOLUTION DOCUMENT

BASIC CHARACTERISTIC

- Microservice based solution using GoLang
- RPC as the primary messaging pattern
- Consul is a service mesh solution
- ElasticSearch as a Database
- GoMicro as the basic framework for the microservice architecture

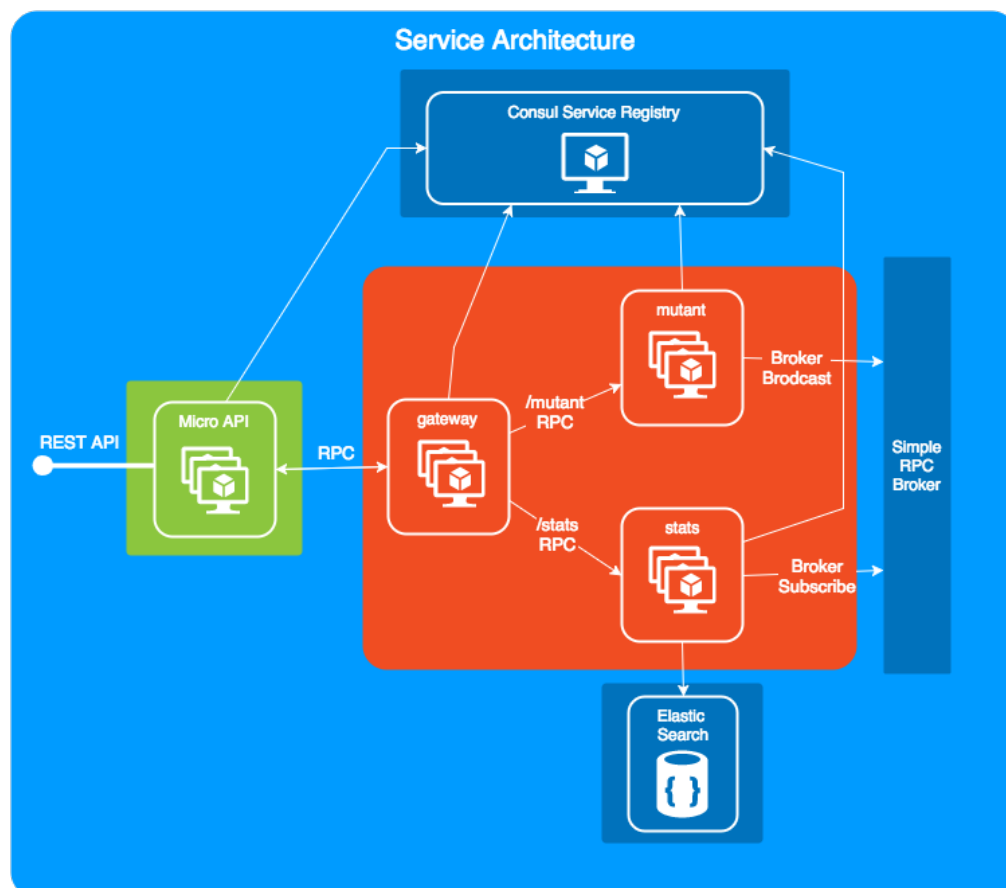
WHY GOLANG? [SOURCE](#)

Microservices are supported by just about all languages, after all, microservices are a concept rather than a specific framework or tool. That being said, some languages are better suited and, or have better support for microservices than others. One language with great support is Golang.

Golang is very light-weight, very fast, and has a fantastic support for concurrency, which is a powerful capability when running across several machines and cores.

Go also contains a very powerful standard library for writing web services.

ARCHITECTURE



GATEWAY MICROSERVICE

Gateway microservice will be in charge of registering the APIs into Micro API and forward the requests. see <https://github.com/micro/micro/tree/master/api>

MUTANT MICROSERVICE

This microservice will contain the DNA Analysis algorithm. It will analyze the DNA data and produce a result. The result is published in the Message broker.

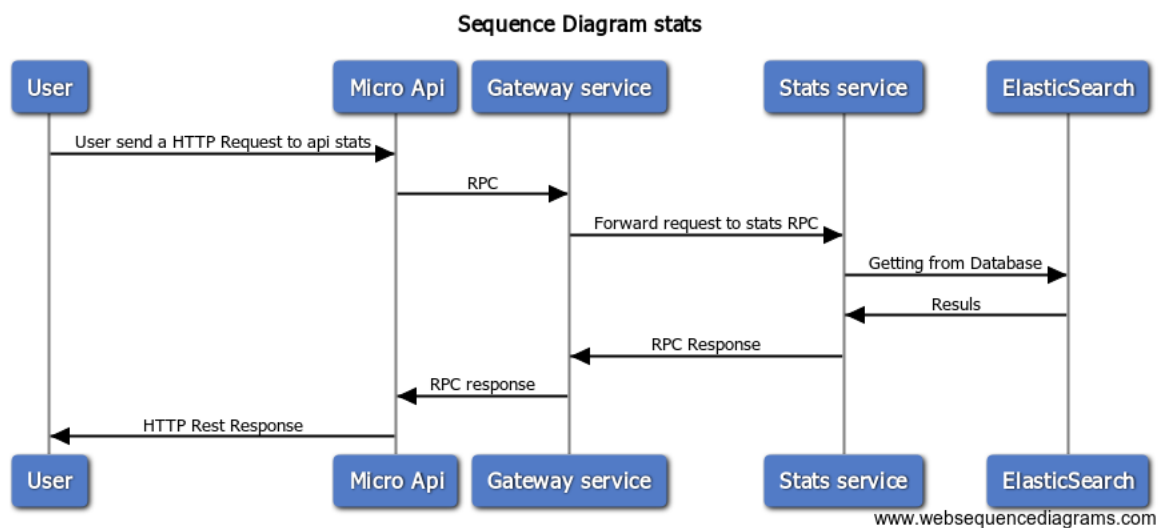
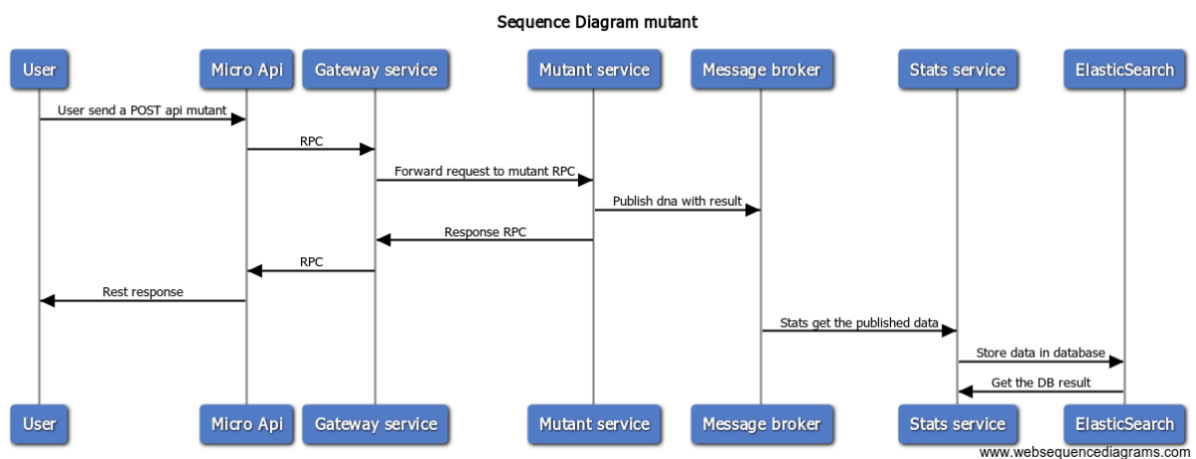
STATS MICROSERVICE

This microservice has two main objectives:

- To subscribe to the mutant results message broker to store it in the DB
- To Return the stats results

SEQUENCE DIAGRAMS

We have two main sequence diagram, one for each service



SOURCE CODE

The source code was organized in two repositories:

<https://github.com/rodrigodmd/ml-mutant> contains the algorithm to analyze the DNA data. This resources will be consumes as a library from the mutant microservice.

<https://github.com/rodrigodmd/ml-mutant-srv> Contains:

- the source code for the microservices. Will consume ml-mutant library as a dependency.
- docker image scripts (build, run and publish)
- Deployment scripts

DEPLOYED ENVIRONMENT

Deployment done in Google Kubernetes and configured to auto scale depending on the CPU usage

Base API url: <http://35.226.184.83/api>

The screenshot shows the Google Cloud Platform console interface. The top navigation bar includes the Google Cloud Platform logo, the ML Mutant Service dropdown menu, and a search icon. The left sidebar lists various services: Kubernetes Engine, Clústeres, Cargas de trabajo, Servicios (selected), Applications, Configuración, and Almacenamiento. The main content area displays the 'Servicios' page for the ML Mutant Service, with an 'ACTUALIZAR' button. Below this, there is a section for 'Servicios de Kubernetes' and 'Servicios de Broker BETA'. A descriptive text explains that services are groups of pods with a network endpoint for discovery and load balancing. A filter bar shows 'Es objeto del sistema : False' and a 'Filtrar recursos' button. A table lists the deployed services:

Nombre	Estado	Tipo de servicio	Endpoints	Grupos	Espacio de nombres	Clúster
api	✓ Aceptar	Balanceador de carga	35.226.184.83:80	1 / 1	default	standard-cluster-1
consul	✓ Aceptar	IP de clúster	10.11.244.197	1 / 1	default	standard-cluster-1
elasticsearch	✓ Aceptar	IP de clúster	10.11.254.181	1 / 1	default	standard-cluster-1
web	✓ Aceptar	IP de clúster	10.11.246.116	1 / 1	default	standard-cluster-1

Google Cloud Platform

ML Mutant Service

Kubernetes Engine

Clústeres

Cargas de trabajo

Servicios

Applications

Configuración

Almacenamiento

Cargas de trabajo

ACTUALIZAR

DESPLEGAR

Las cargas de trabajo son unidades informáticas desplegables que se pueden crear y administrar en un clúster.

Es objeto del sistema : False

Filtrar cargas de trabajo

Columnas

Nombre	Estado	Tipo	Grupos	Espacio de nombres	Clúster
api	OK	Deployment	1/1	default	standard-cluster-1
consul	OK	Deployment	1/1	default	standard-cluster-1
elasticsearch	OK	Deployment	1/1	default	standard-cluster-1
gateway	OK	Deployment	1/1	default	standard-cluster-1
mutant	OK	Deployment	1/1	default	standard-cluster-1
stats	OK	Deployment	1/1	default	standard-cluster-1
web	OK	Deployment	1/1	default	standard-cluster-1

EXAMPLES

MUTANT DNA

```
curl --header "Content-Type: application/json" \
--request POST \
--data '{ "dna": ["ATGCAA","CAGTTT","TTATTT","AGAAGG","CCACTA","TCACTG"] }' \
http://35.226.184.83/api/mutant
```

HUMAN DNA

```
curl --header "Content-Type: application/json" \
--request POST \
--data '{ "dna": ["ATGC", "CAGT", "TATT", "AGAA"] }' \
http://35.226.184.83/api/mutant
```

INVALID DNA STRUCTURE

```
curl --header "Content-Type: application/json" \
--request POST \
--data '{ "dna": ["ATGCAA","invalid","TTATTT","AGAAGG","CCACTA","TCACTG"] }' \
http://35.226.184.83/api/mutant
```

GET STATS

```
curl --header "Content-Type: application/json" \
--request GET \
http://35.226.184.83/api/stats
```