advanced search

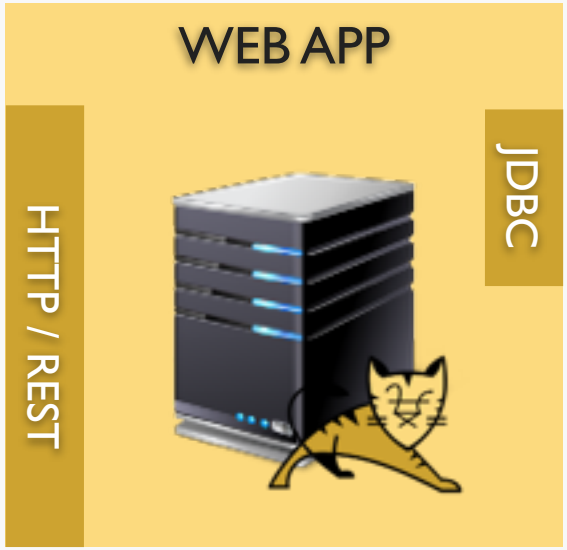
for your legacy app

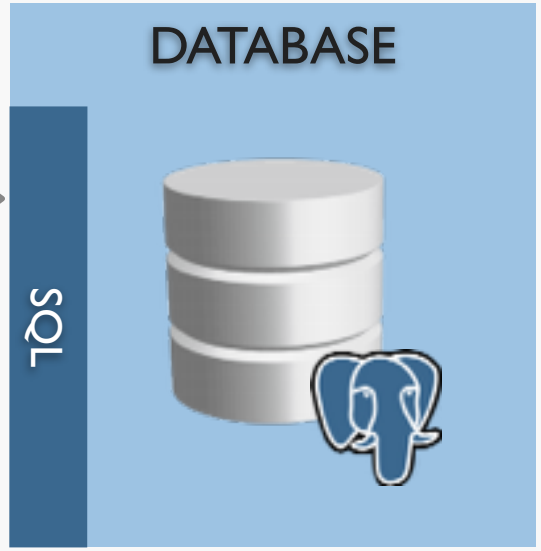
David Pilato
Technical advocate

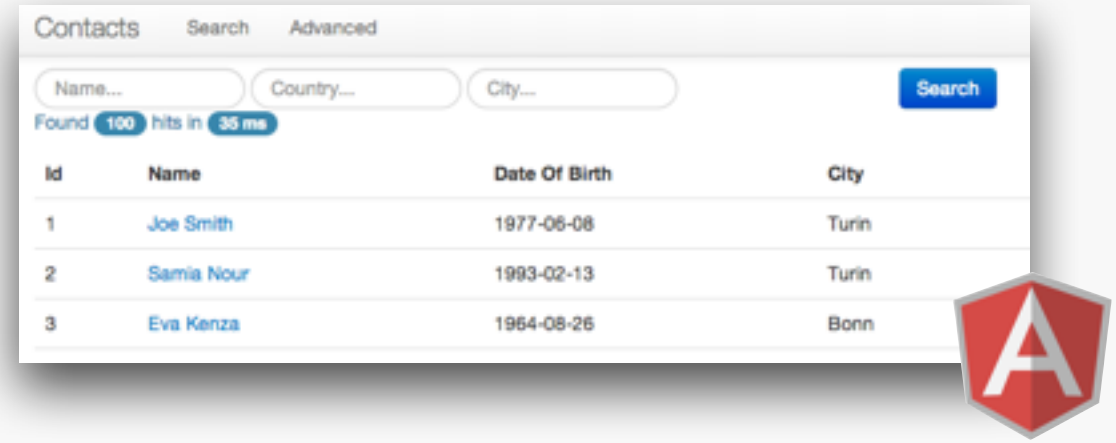
elasticsearch.
@dadoonet

our use case

our legacy platform







elasticsearch.

our legacy application

WEB APP

HTTP / REST

WEBSERVICE LAYER

SERVICE LAYER

DAO LAYER

JDBC

DOMAIN LAYER

our legacy application

WEB APP

WEBSERVICE LAYER

HTTP / REST

SPRING WEB

```
@Controller
public class PersonRestAPI {
   @Autowired PersonService personService;

   @RequestMapping(method = RequestMethod.PUT, value = "/1/person/{id}")
   public @ResponseBody JsonResponse upsert(
        @PathVariable String id,
        @RequestBody Person json) throws Exception {
       person = personService.save(person);
       return new JsonResponse(true, person.getReference());
    }

    class JsonResponse {
        private boolean ok;
        private String reference;
    }
}
```

our legacy application

WEB APP

SERVICE LAYER

SPRING CONTEXT

SPRING WEB

HTTP / REST

```
@Service
public class PersonService {
    @Autowired PersonDao personDao;

    public Person save(Person person) {
        return personDao.save(person);
    }
}
```

our legacy application

WEB APP

HTTP / REST

SPRING WEB

SPRING CONTEXT

SPRING ORM

HIBERNATE

DAO LAYER

```
@Repository
@Transactional
public class PersonDaoImpl {
   protected HibernateTemplate template = null;

   public void setSessionFactory(SessionFactory sessionFactory) {
        template = new HibernateTemplate(sessionFactory);
    }

    @Transactional(propagation = Propagation.REQUIRES_NEW)
    public Person save(Person person) {
       return template.merge(person);
    }
}
```

our legacy application

WEB APP

DAO LAYER

HTTP / REST

SPRING WEB

SPRING CONTEXT

SPRING ORM

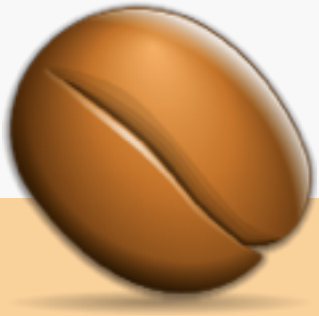
HIBERNATE

SPRING JDBC

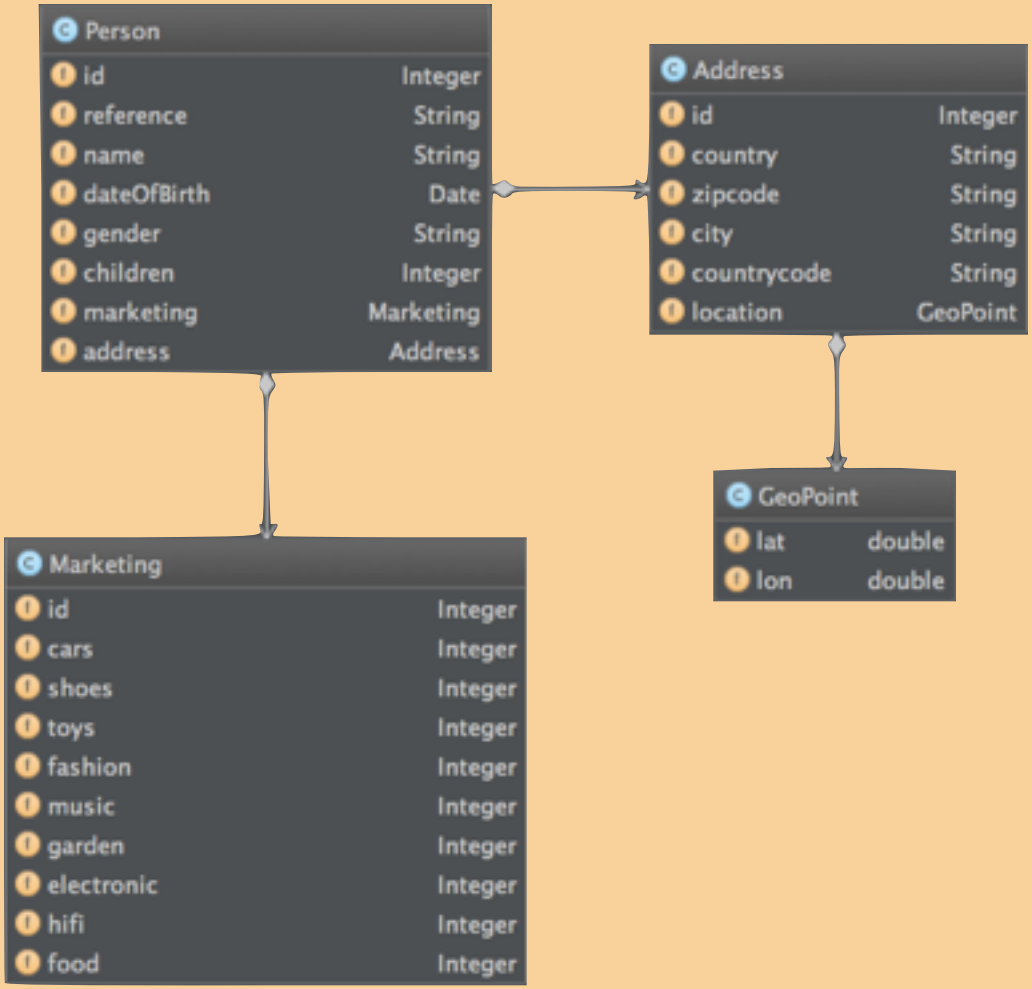
JDBC

```
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
  <property name="driverClassName" value="org.postgresql.Driver"/>
  <property name="url"
      value="jdbc:postgresql://localhost:5432/dpilato"/>
</bean>

<bean id="sessionFactory"
class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"
p:dataSource-ref="dataSource">
  <property name="packagesToScan"
    value="fr.pilato.demo.legacysearch.domain" />
  <property name="hibernateProperties">
    <value>
hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
    </value>
  </property>
</bean>

<bean id="personDao"
      class="fr.pilato.demo.legacysearch.dao.PersonDaoImpl"
      p:sessionFactory-ref="sessionFactory" />
```

our legacy domain



JAVA BEANS



our legacy database

SQL

demo time

our legacy app

```
$ git clone https://github.com/dadoonet/legacy-search.git
$ git checkout 00-legacy
$ mvn clean install jetty:run
```

adding elasticsearch

running a node

download and run

```
wget https://download.elasticsearch.org/elasticsearch/elasticsearch/elasticsearch-1.1.1.tar.gz
tar xzf elasticsearch-1.1.1.tar.gz
cd elasticsearch-1.1.1
bin/elasticsearch
```

```
[node                     ] [Ghost Maker] version[1.1.1], pid[11878], build[xxx/xxx]
[node                     ] [Ghost Maker] initializing ...
[plugins                  ] [Ghost Maker] loaded [], sites []
[node                     ] [Ghost Maker] initialized
[node                     ] [Ghost Maker] starting ...
[transport                ] [Ghost Maker] publish_address {inet[/192.168.0.15:9300]}
[cluster.service          ] [Ghost Maker] new_master [Ghost Maker][xQOsI02yTI63tYnM9RzF4Q]
[localhost][inet[/192.168.0.15:9300]], reason: zen-disco-join (elected_as_master)
[discovery                ] [Ghost Maker] elasticsearch/xQOsI02yTI63tYnM9RzF4Q
[http                     ] [Ghost Maker] publish_address {inet[/192.168.0.15:9200]}
```

and play: index!

```
PUT /person/person/1
{
   "name":"Joe Pink"
}
```

```
PUT /person/person/2
{
   "name":"Joe Smith"
}
```

```
PUT /person/person/1
{
   "name":"Joe Pink",
   "dateOfBirth":"1971-12-26"
}
```

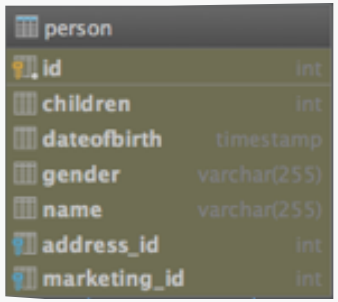
```
PUT /person/person/2
{
   "name":"Joe Smith",
   "gender":"male",
   "address":{
      "country":"England",
      "city":"London"
  }
}
```

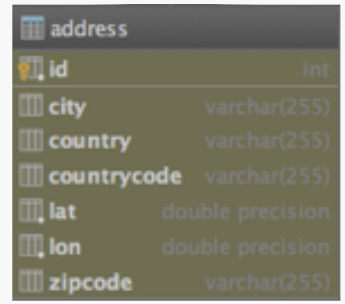
and play: search!

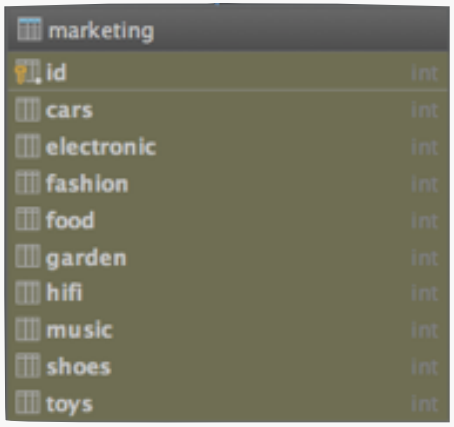
```
GET /person/person/_search
{
    "query": {
        "bool": {
            "must": [ {
                "match": {
                      "address.country": "England"
                   }
               }, {
                   "range": {
                      "dateOfBirth": {
                         "from": "1970",
                         "to": "1971"
                      }
                   }
               }
            ],
            "must_not": [ {
                "match": {
                      "address.city": "Plymouth"
                   }
               }
            ]
        }
    }
}
```

connecting with our app architecture

JSON document design





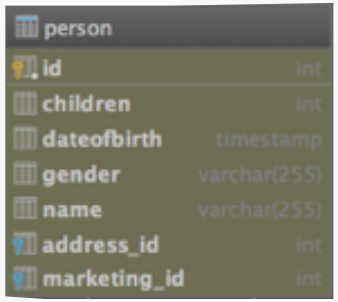


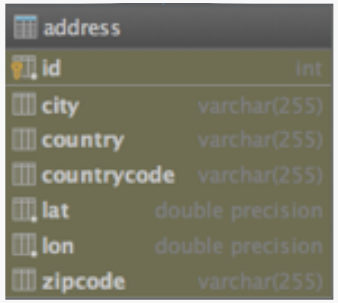

```
PUT /person/person/1
{
   "name":"Joe Pink",
   "dateOfBirth":"1971-12-26",
   "address_id":"2",
   "marketing_id":"3"
}
```

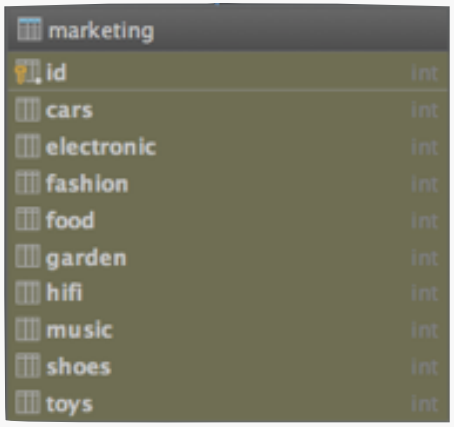
```
PUT /person/address/2
{
   "city":"Paris",
   "country":"France"
}
```

```
PUT /person/marketing/3
{
   "cars":1000,
   "food":1500
}
```

JSON document design



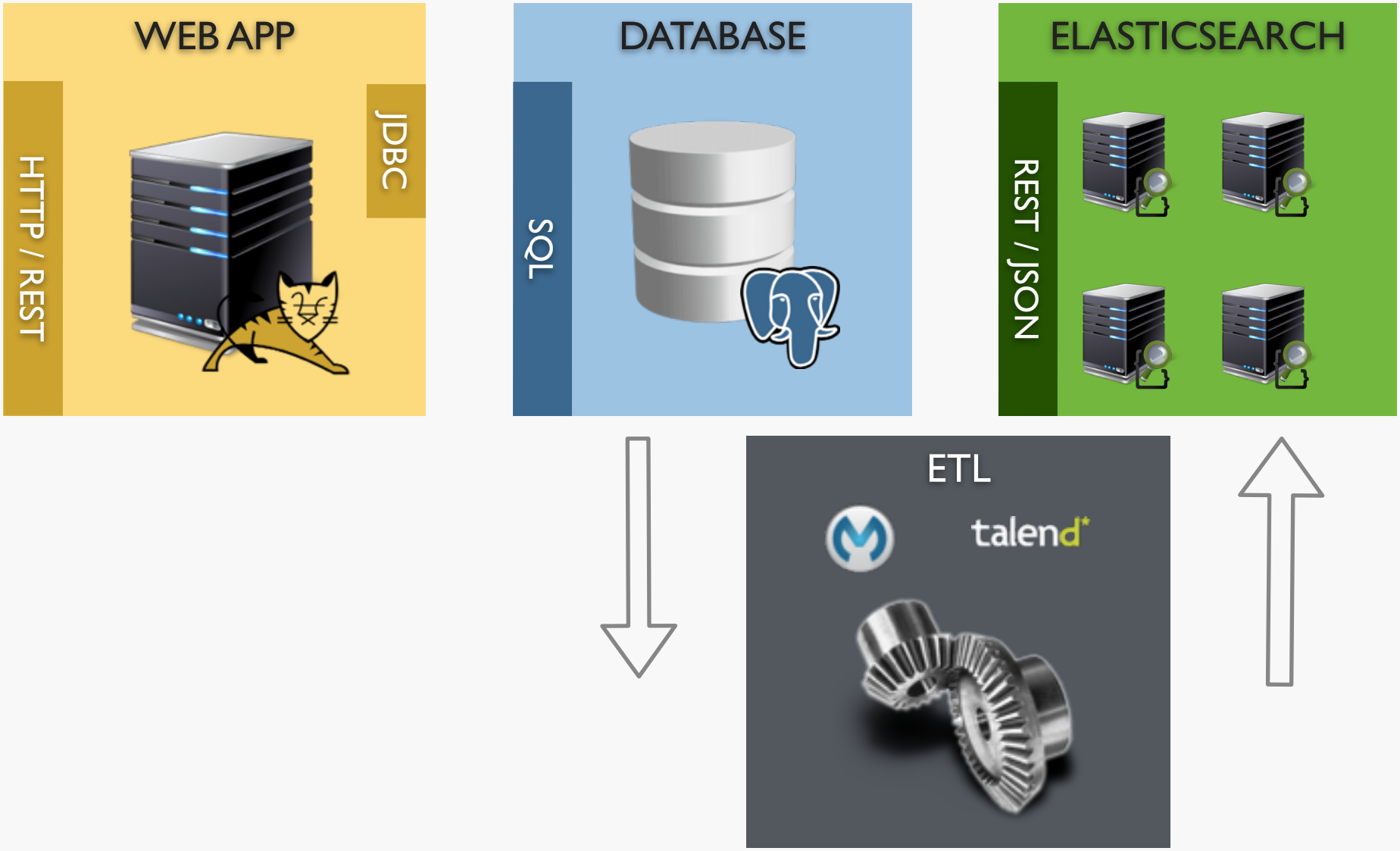





```
PUT /person/person/1
{
   "name":"Joe Pink",
   "dateOfBirth":"1971-12-26",
   "address":{
      "city":"Paris",
      "country":"France"
   },
   "marketing":{
      "cars":1000,
      "food":1500
   }
}
```


elasticsearch.

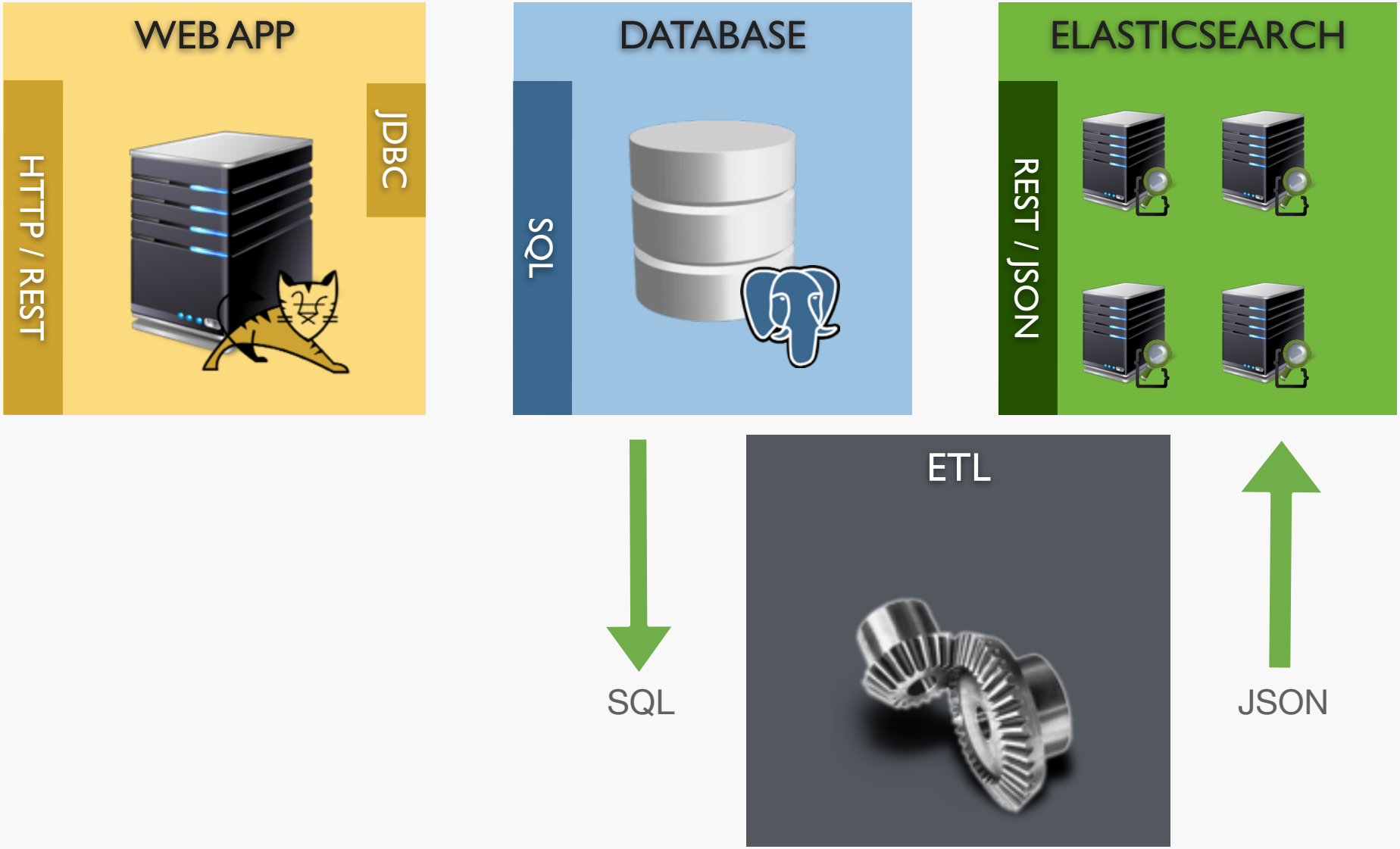
using a ETL



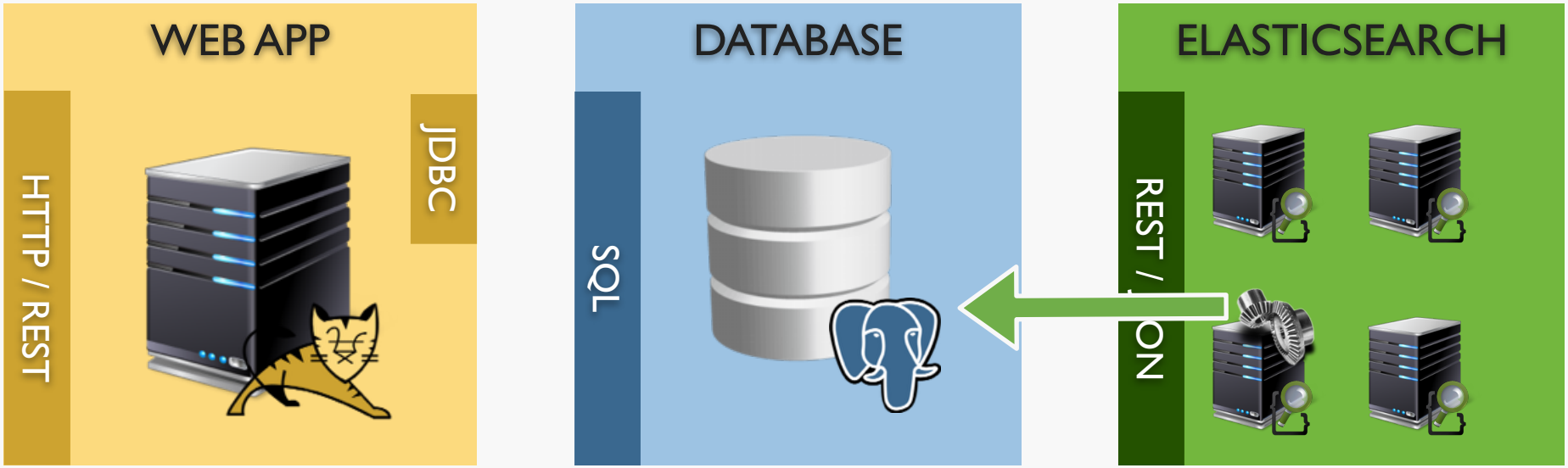
elasticsearch rivers

JDBC River

JDBC river

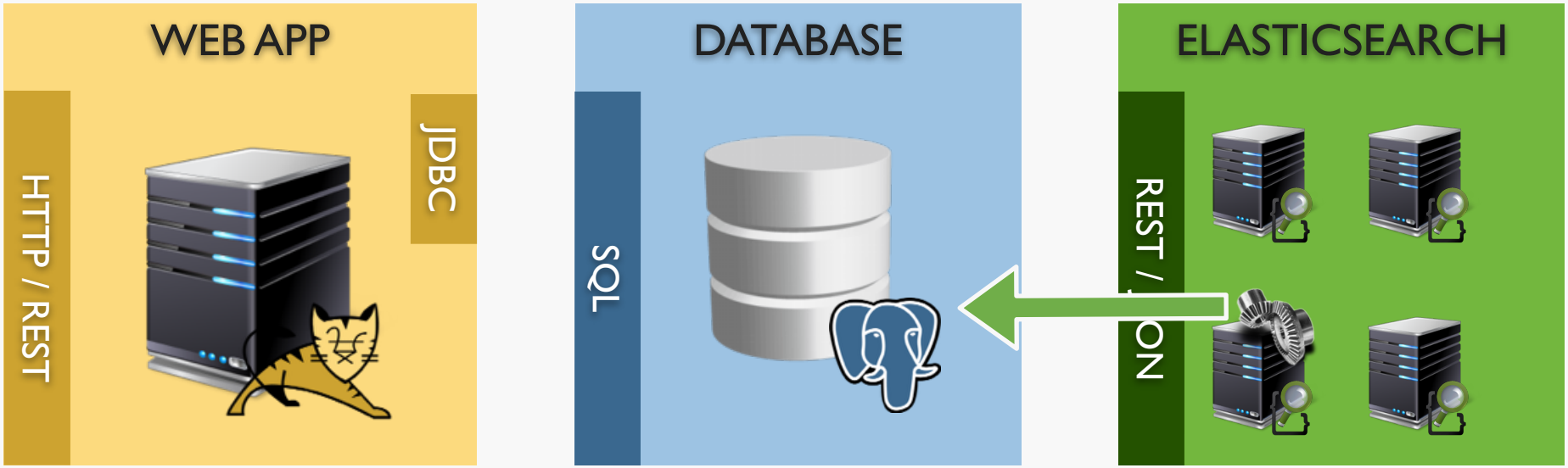


JDBC river



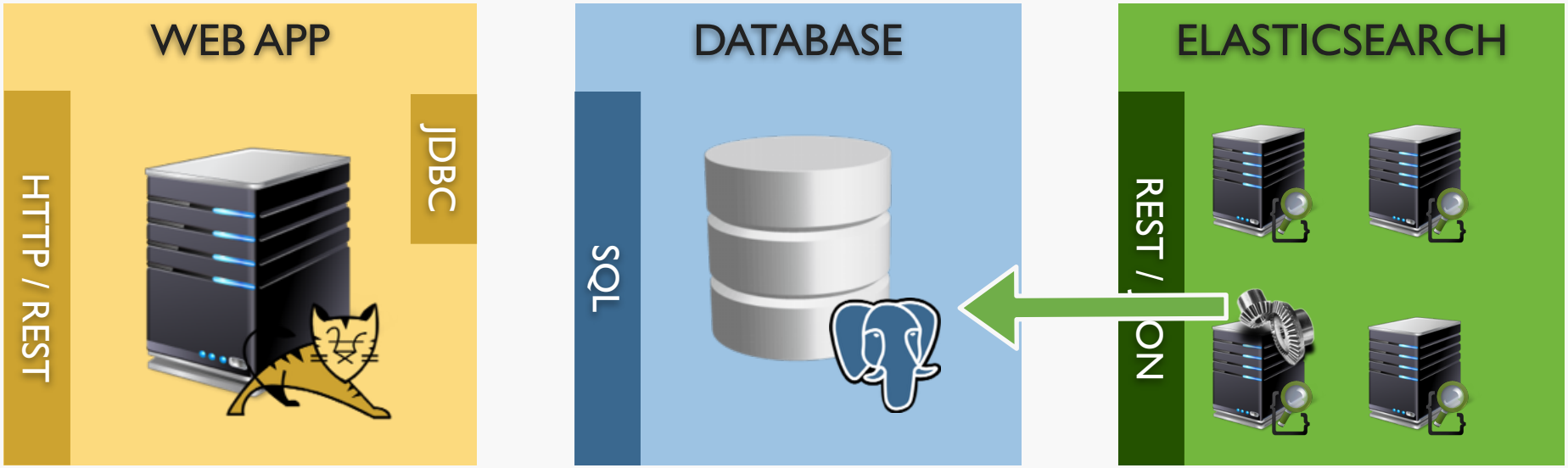

```
# Install JDBC river
bin/plugin --install river-jdbc --url http://bit.ly/1iadfnF

# Add Postgresql JDBC driver
wget http://jdbc.postgresql.org/download/postgresql-9.3-1101.jdbc4.jar \
     -O plugins/river-jdbc/postgresql-9.3-1101.jdbc4.jar

# Relaunch elasticsearch
bin/elasticsearch
```

JDBC river




```
PUT /_river/my_jdbc_river/_meta
{
    "type" : "jdbc",
    "jdbc" : {
        "url" : "jdbc:postgresql://localhost:5432/dpilato",
        "sql" : "select person.reference as \"_id\", person.name as \"name\",
address.country as \"address.country\" from person left join address on
person.address_id = address.id",
        "schedule" : "0 */5 * ? * *",
        "index" : "person",
        "type" : "person"
    }
}
```

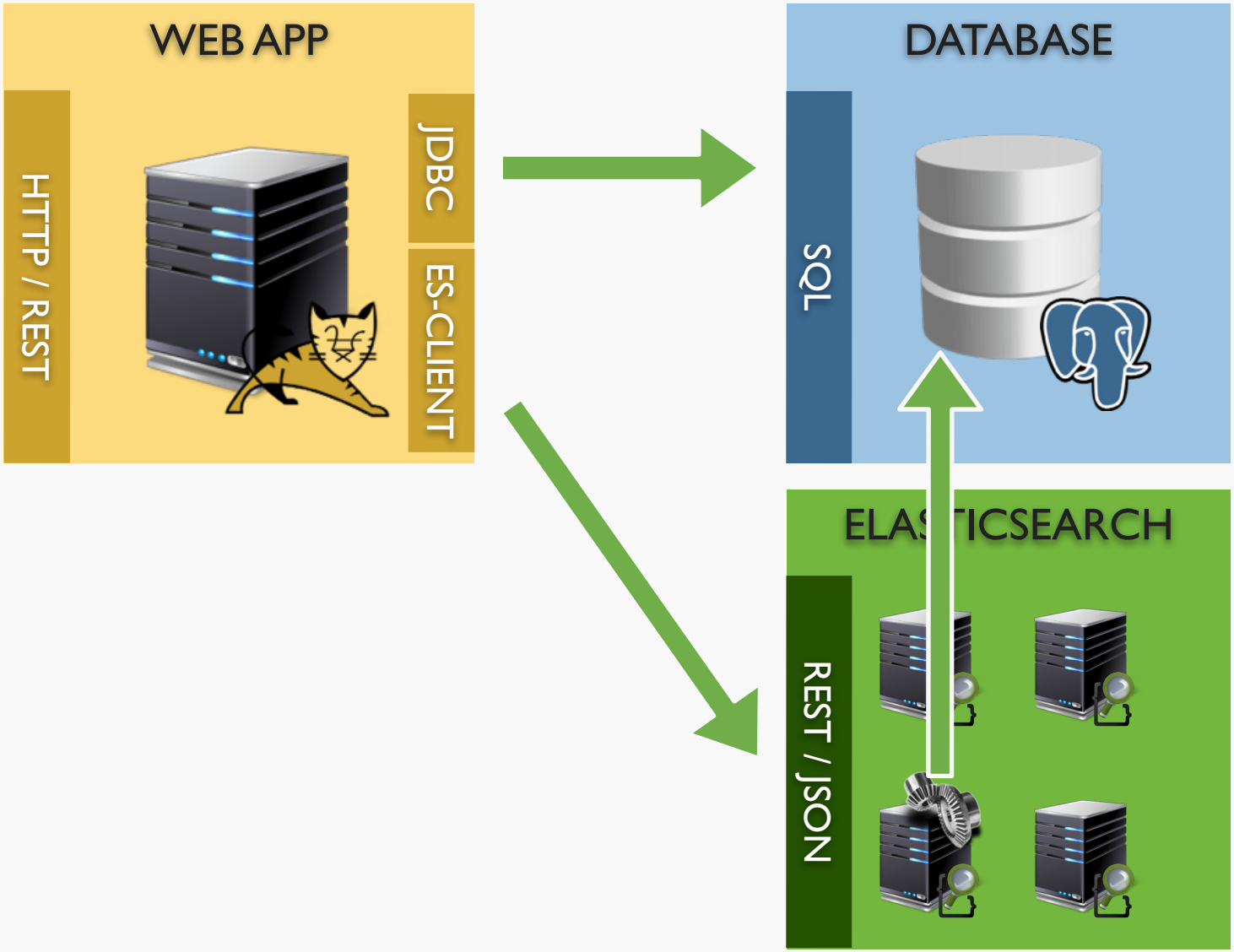
JDBC river




```
PUT /_river/my_jdbc_river/_meta
{
    "type" : "jdbc",
    "jdbc" : {
        "url" : "jdbc:postgresql://localhost:5432/dpilatc
        "sql" : "select person.reference as \"_id\", pers
address.country as \"address.country\" from person left j
person.address_id = address.id",
        "schedule" : "0 */5 * ? * *",
        "index" : "person",
        "type" : "person"
    }
}
```

```
{
   "name": "Joe Smith",
   "children": 3,
   "gender": "female",
   "reference": "1",
   "address": {
      "city": "Munich",
      "country": "Germany"
   }
}
```

JDBC river



our legacy application

WEB APP

SERVICE LAYER

HTTP / REST

SPRING WEB

SPRING CONTEXT

```
public class PersonService {
    @Autowired ElasticsearchDao elasticsearchDao;
    @Autowired ObjectMapper mapper;

    public SearchResponse<Person> search(String q, Integer from, Integer size) {
        SearchResponse searchResponse = elasticsearchDao.search(q, from, size);
        // ...
        response.setTotal(searchResponse.getHits().getTotalHits());
        response.setTook(searchResponse.getTookInMillis());
        int i = 0;
        for (SearchHit hit : searchResponse.getHits().getHits()) {
            persons[i++] = mapper.readValue(hit.getSourceAsString(), Person.class);
        }
        response.setHits(persons);
        return response;
    }
}
```

our legacy application

WEB APP

DAO LAYER

SPRING CONTEXT

SPRING WEB

HTTP / REST

```
@Repository
public class ElasticsearchDao {
    @Autowired Client esClient;

    public SearchResponse search(String q, Integer from, Integer size) {
        SearchResponse response = esClient.prepareSearch()
                .setQuery(QueryBuilders.queryString(q))
                .setFrom(from)
                .setSize(size)
                .execute().actionGet();

        return response;
    }
}
```

our legacy application

WEB APP

DAO LAYER

HTTP / REST

SPRING WEB

SPRING CONTEXT

SPRING ES

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:elasticsearch="http://www.pilato.fr/schema/elasticsearch"
       xsi:schemaLocation="
            http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans.xsd
            http://www.pilato.fr/schema/elasticsearch
            http://www.pilato.fr/schema/elasticsearch/elasticsearch-0.3.xsd

 <elasticsearch:client id="esClient" />
 <bean id="mapper" class="com.fasterxml.jackson.databind.ObjectMapper" />

</beans>
```

ES-CLIENT




https://github.com/dadoonet/spring-elasticsearch

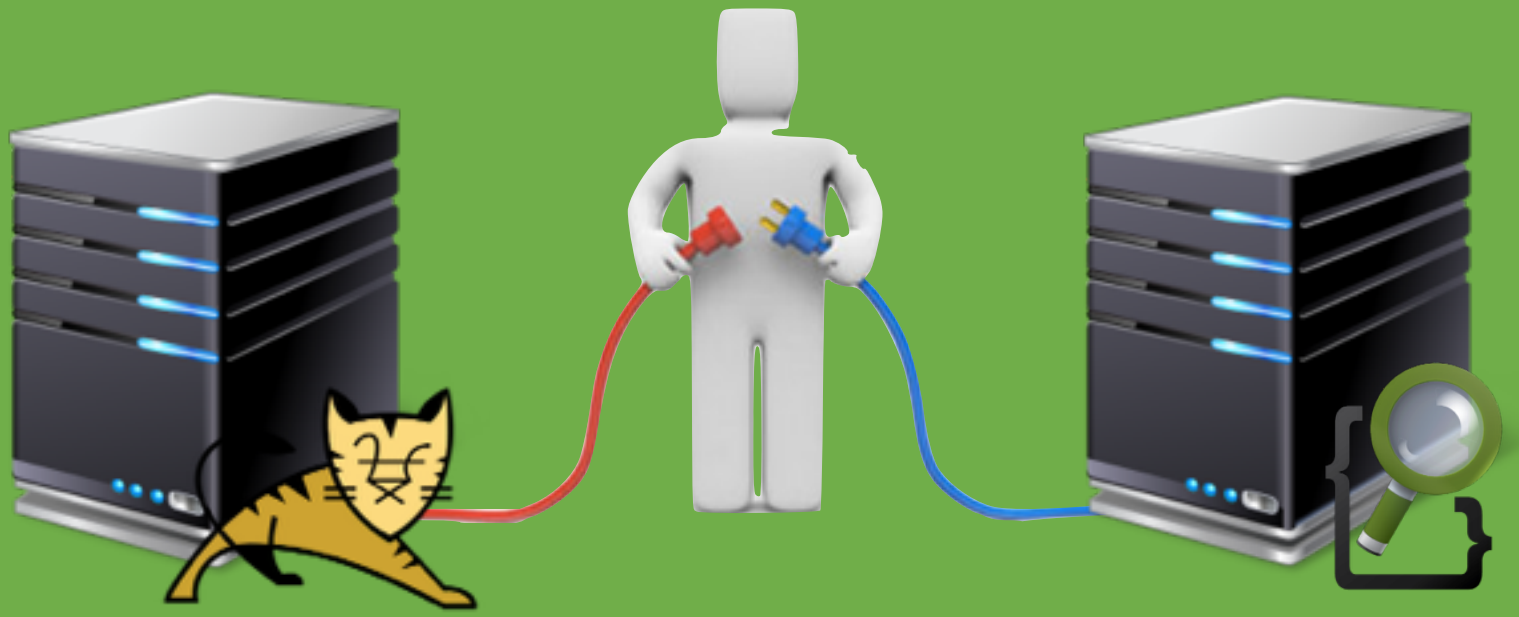
demo time

jdbc river

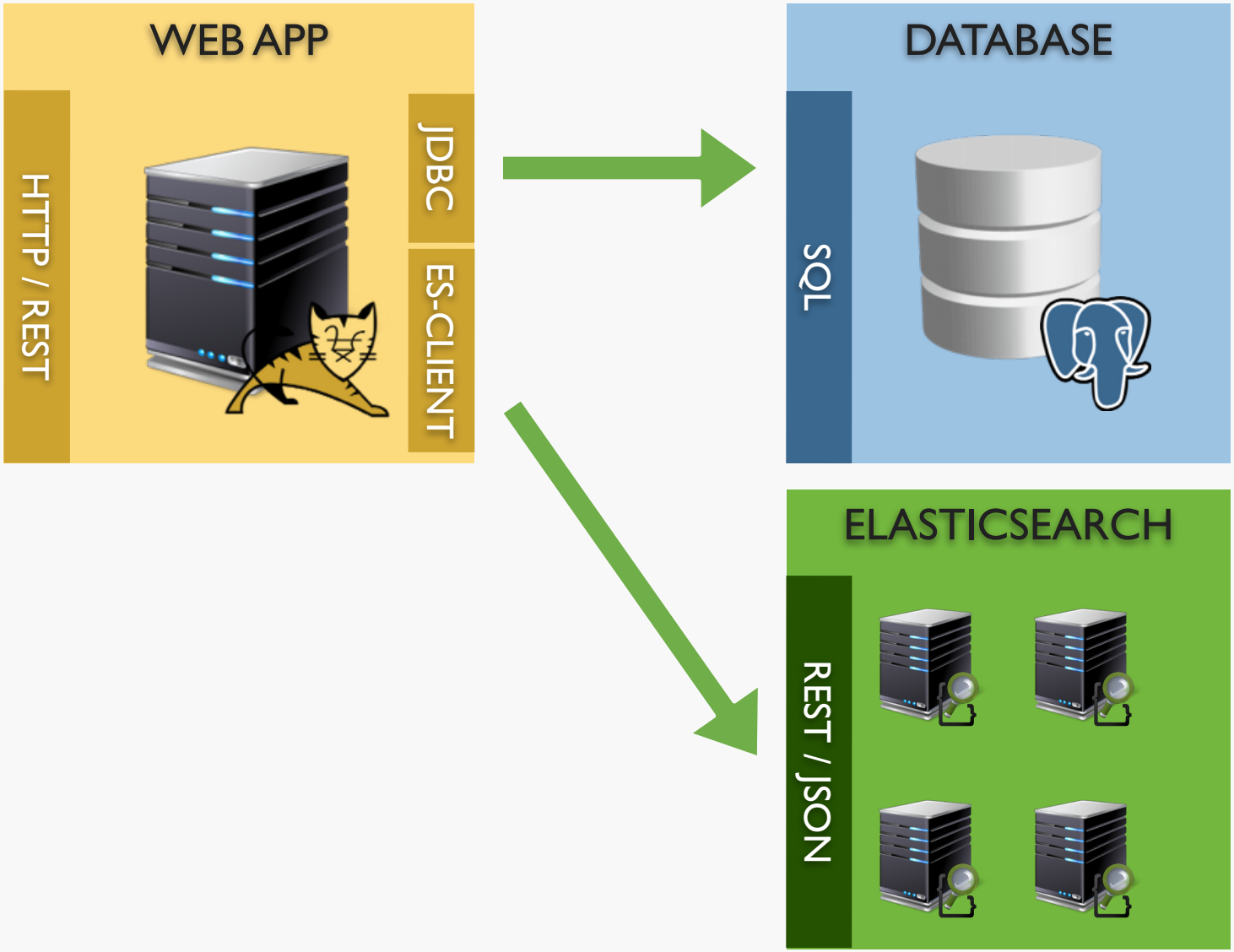
```
$ git checkout 01-jdbcriver
$ mvn clean install jetty:run

$ cat README.markdown
```

Do It Yourself

direct connection



direct connection



elasticsearch CRUD

WEB APP

SERVICE LAYER

HTTP / REST

ES-CLIENT

```
public class PersonService {
   public Person save(Person person) {
        Person dbPerson = personDao.save(person);
        try {
            elasticsearchDao.save(person);
        } catch (Exception e) {
            logger.error("Can not save person #" + person.getReference());
        }
        return dbPerson;
    }

    public boolean delete(String id) throws Exception {
        personDao.delete(get(id));
        elasticsearchDao.delete(id);
        return true;
    }
}
```

elasticsearch CRUD

WEB APP

DAO LAYER

HTTP / REST

ES-CLIENT

```
public class ElasticsearchDao {
    public void save(Person person) {
        byte[] bytes = mapper.writeValueAsBytes(person);
        esClient.prepareIndex("person", "person", person.getReference())
             .setSource(bytes)
             .execute().actionGet();
    }

    public void delete(String reference) {
        esClient.prepareDelete("person", "person", reference)
             .execute().actionGet();
    }
}
```

simplify search responses

WEB APP

SERVICE LAYER

HTTP / REST

```
public class PersonService {
    @Autowired ElasticsearchDao elasticsearchDao;
    @Autowired ObjectMapper mapper;

    public SearchResponse<Person> search(String q, Integer from, Integer size) {
        SearchResponse searchResponse = elasticsearchDao.search(q, from, size);
        // ...
        response.setTotal(searchResponse.getHits().getTotalHits());
        response.setTook(searchResponse.getTookInMillis());
        int i = 0;
        for (SearchHit hit : searchResponse.getHits().getHits()) {
            persons[i++] = mapper.readValue(hit.getSourceAsString(), Person.class);
        }
        response.setHits(persons);
        return response;
    }
}
```

simplify search responses

WEB APP

SERVICE LAYER

HTTP / REST

```
public class PersonService {
    @Autowired ElasticsearchDao elasticsearchDao;

    public String search(String q, Integer from, Integer size) {
        SearchResponse response = elasticsearchDao.search(q, from, size);
        return response.toString();
    }
}
```

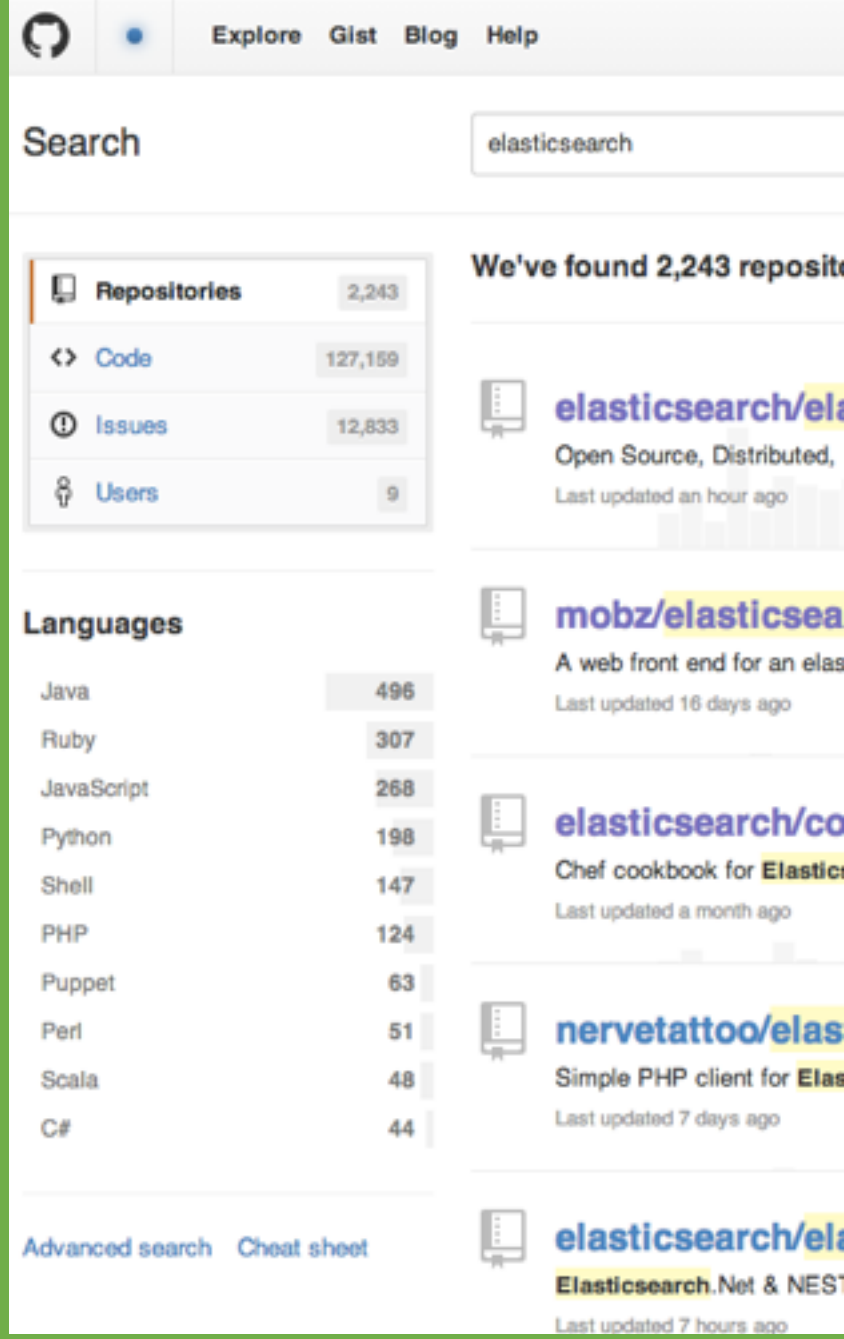
demo time

direct connection

```
$ git checkout 02-direct
$ git checkout 03-mapping
$ git checkout 04-simplify
$ mvn clean install jetty:run

$ cat README.markdown
```

new patterns

faceted navigation



elasticsearch.

compute

WEB APP

DAO LAYER

HTTP / REST

ES-CLIENT

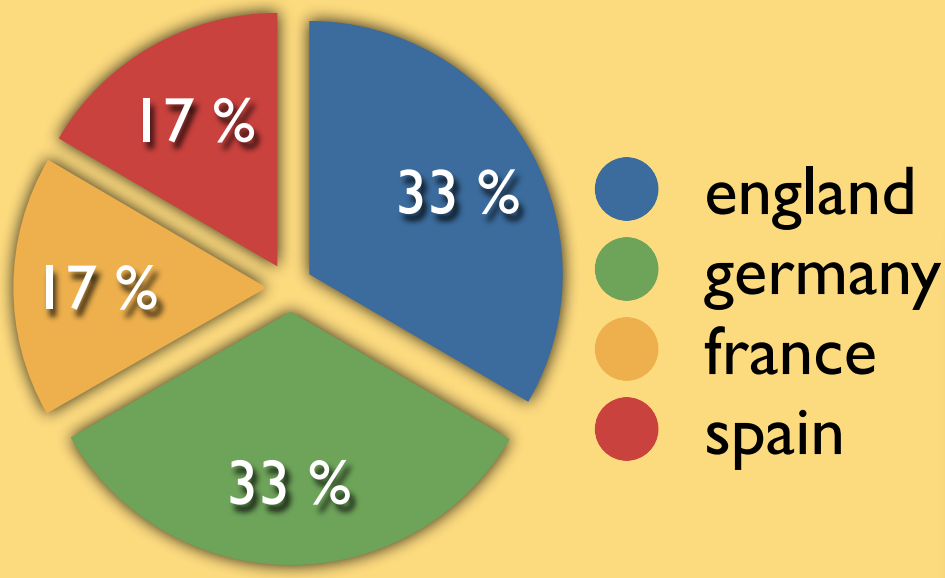
```
public class ElasticsearchDao {
    public SearchResponse search(QueryBuilder qb, Integer from, Integer size) {
        SearchResponse response = esClient.prepareSearch("person")
            .setQuery(qb)
            .addAggregation(
               AggregationBuilders.terms("by_country").field("country")
            )
            .setFrom(from).setSize(size)
            .execute().actionGet();

        return response;
    }
}
```




compute

WEB APP

HTTP / REST



```
{ ..., "aggregations" : {
    "by_country" : {
     "buckets" : [ {
       "key" : "england",
       "doc_count" : 30051
     }, {
       "key" : "germany",
       "doc_count" : 30004
     }, {
       "key" : "france",
       "doc_count" : 15034
     }, {
       "key" : "spain",
       "doc_count" : 14912
     } ]}}}
```

ES-CLIENT

compute

WEB APP

DAO LAYER

HTTP / REST

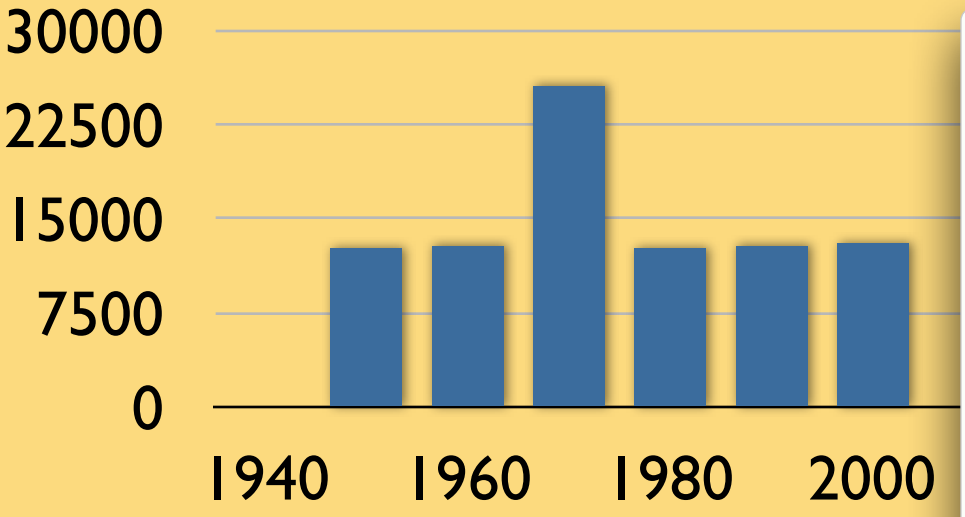
ES-CLIENT

```
public class ElasticsearchDao {
    public SearchResponse search(QueryBuilder qb, Integer from, Integer size) {
        SearchResponse response = esClient.prepareSearch("person")
            .setQuery(qb)
            .addAggregation(
               AggregationBuilders.dateHistogram("by_year")
                   .field("dateOfBirth")
                   .interval(DateHistogram.Interval.YEAR)
                   .format("YYYY")
            )
            .setFrom(from).setSize(size)
            .execute().actionGet();

        return response;
    }
}
```

compute

WEB APP

HTTP / REST



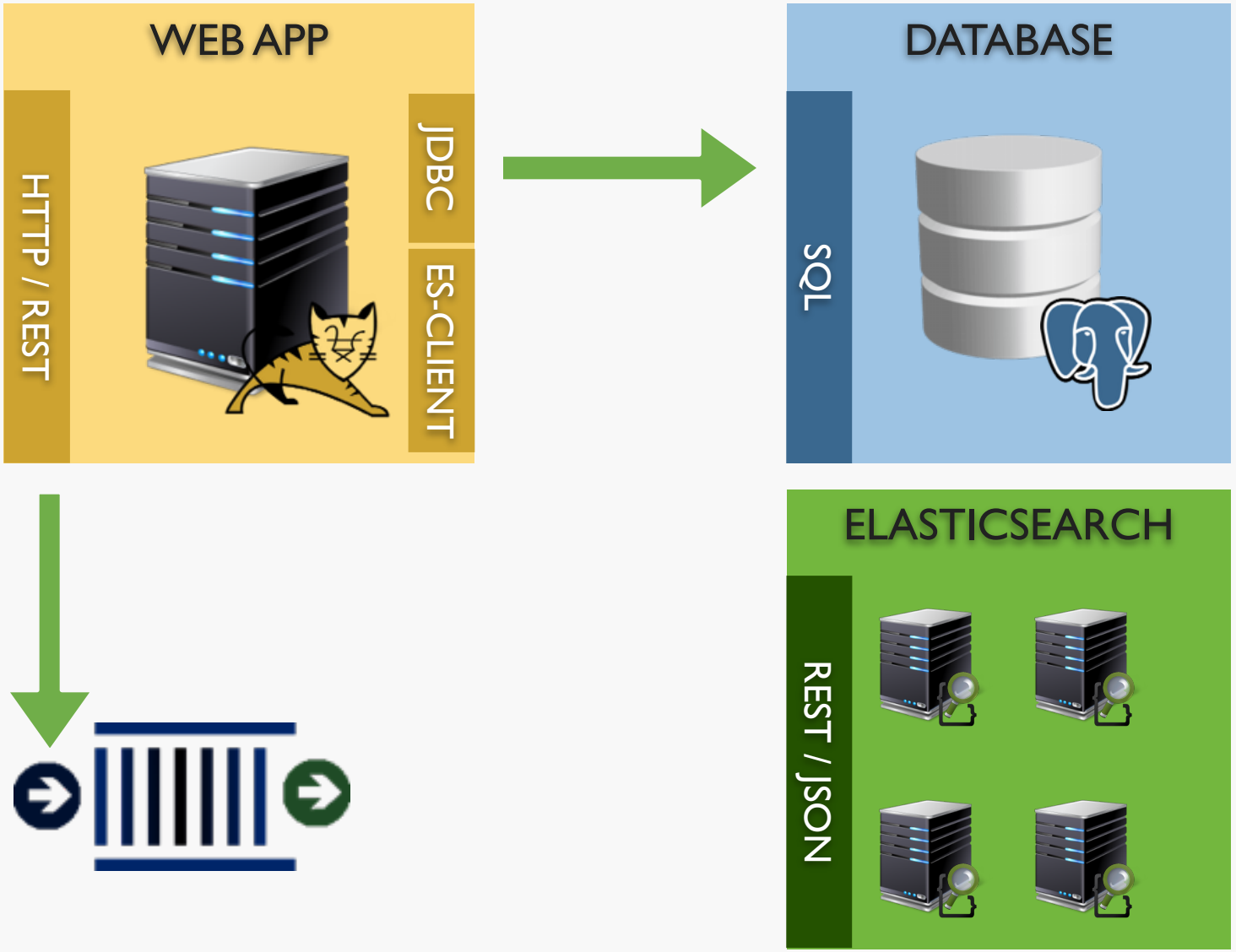

```
{ ..., "aggregations": {
      "by_date": {
        "buckets": [
            {
               "key_as_string": "1960",
               "key": -946080000000,
               "doc_count": 39
            },
            {
               "key_as_string": "1970",
               "key": -630720000000,
               "doc_count": 12677
            },
            {
               "key_as_string": "1980",
               "key": -315360000000,
               "doc_count": 12936
            }, ...
        ]
    }
}}
```

demo time

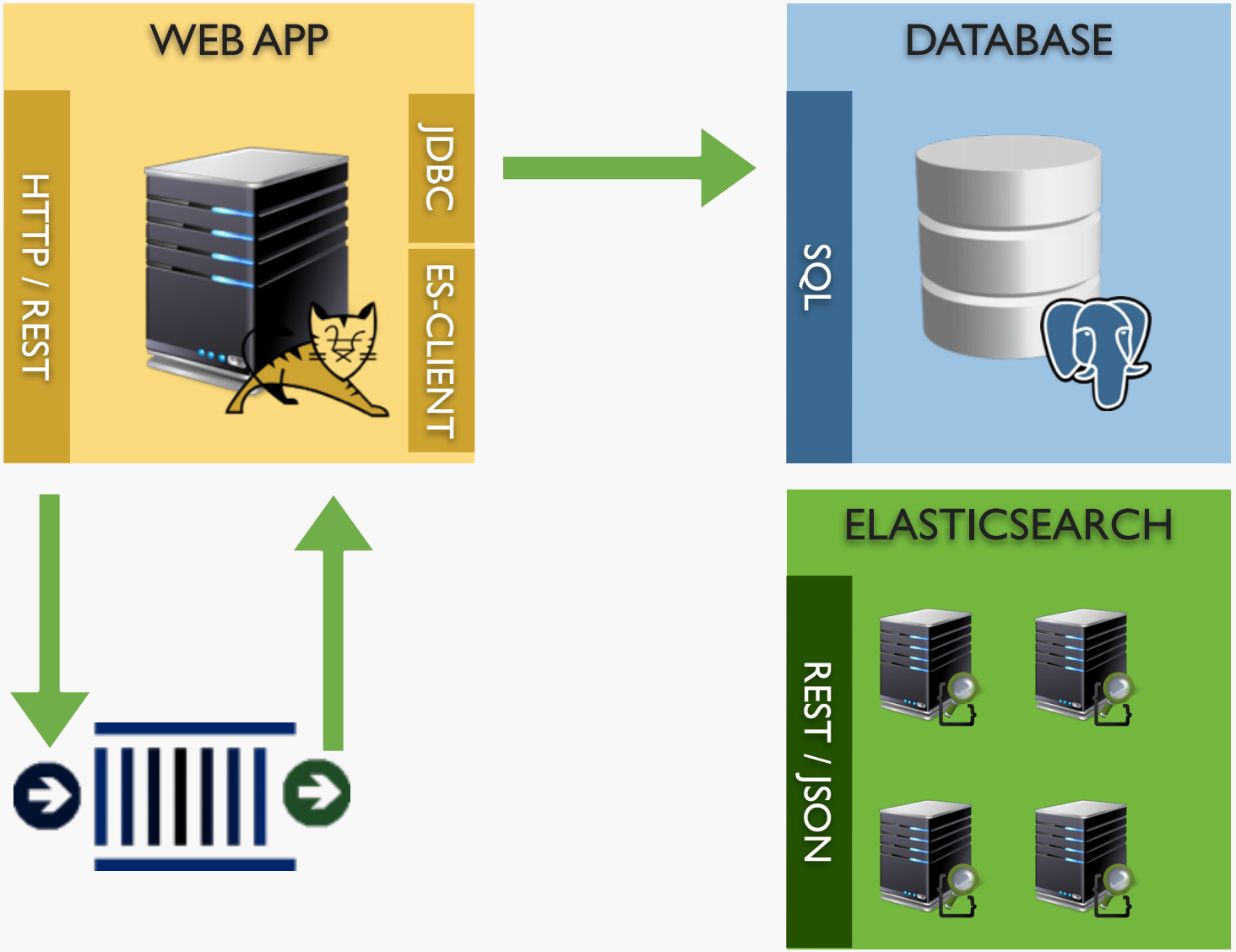
faceted navigation

```
$ git checkout 05-aggs
$ git checkout 06-compute
$ mvn clean install jetty:run

$ cat README.markdown
```

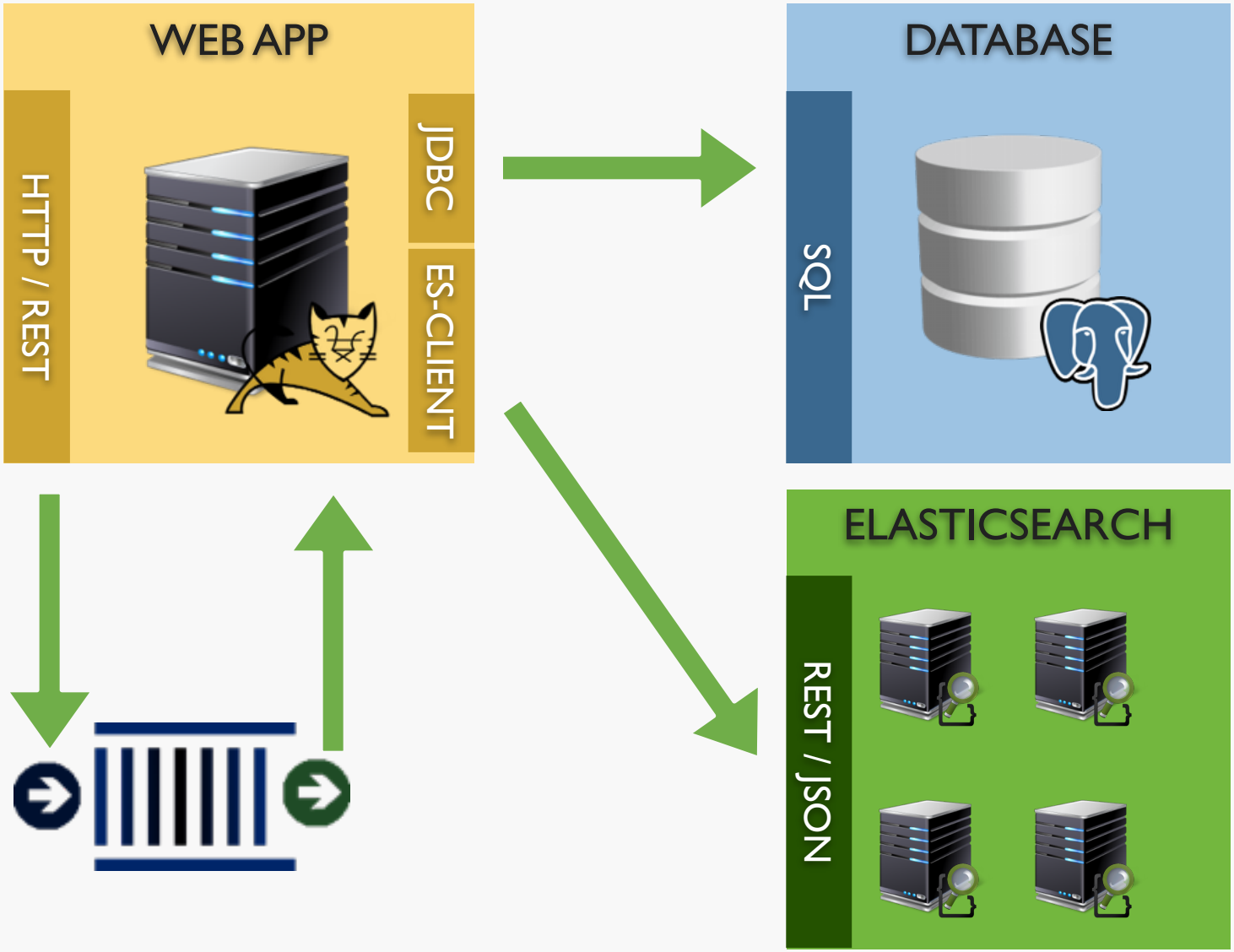
using brokers



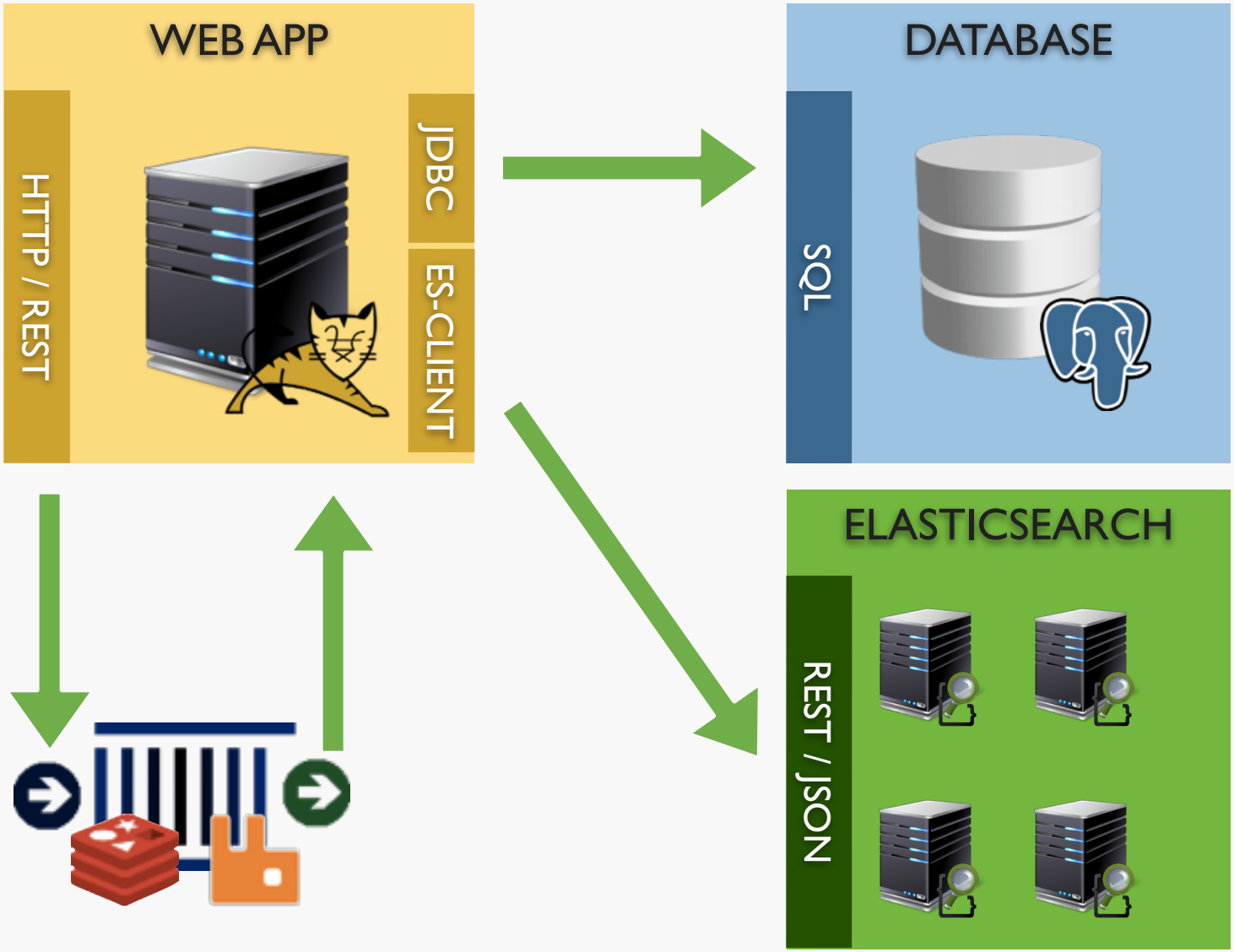
using brokers



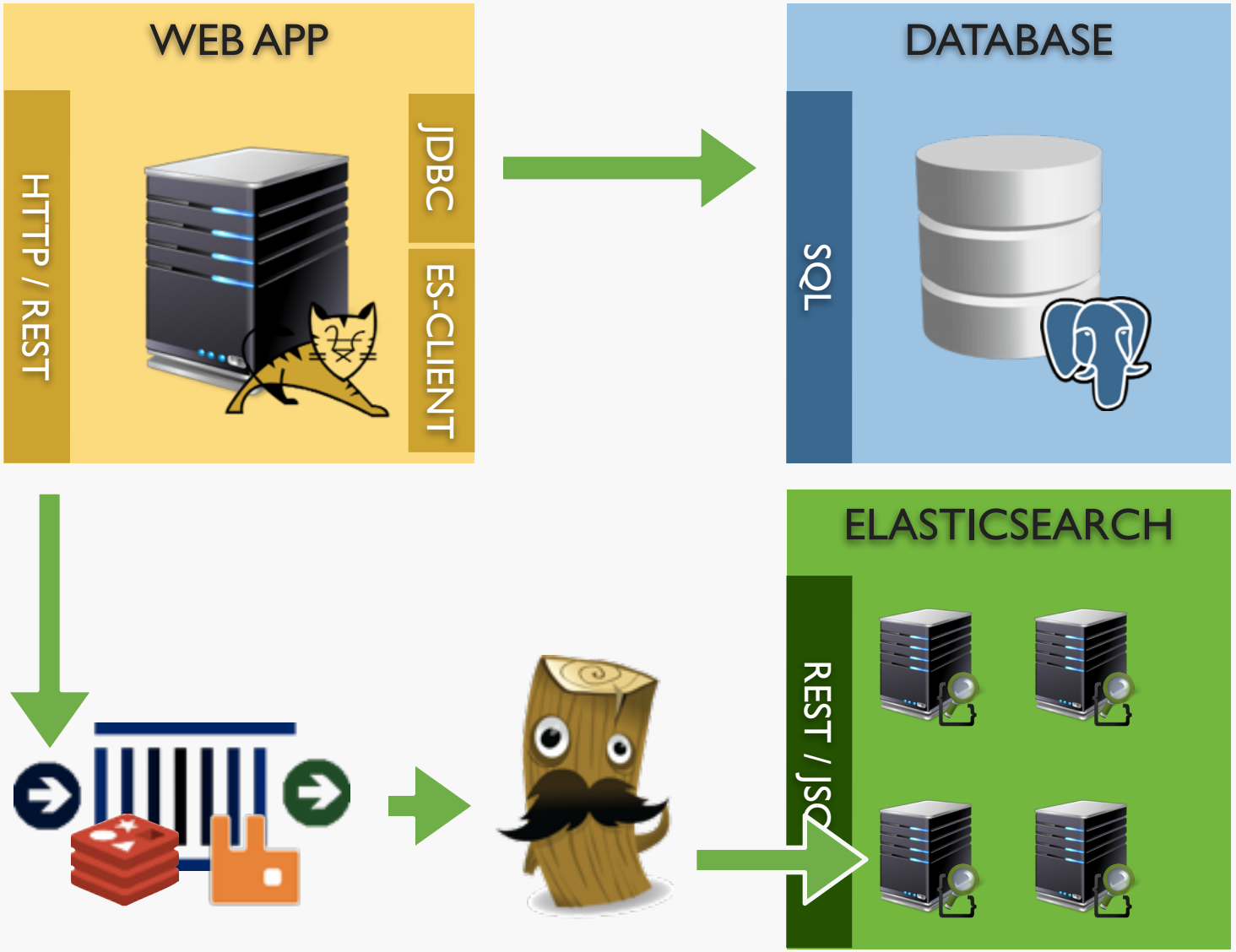
using brokers



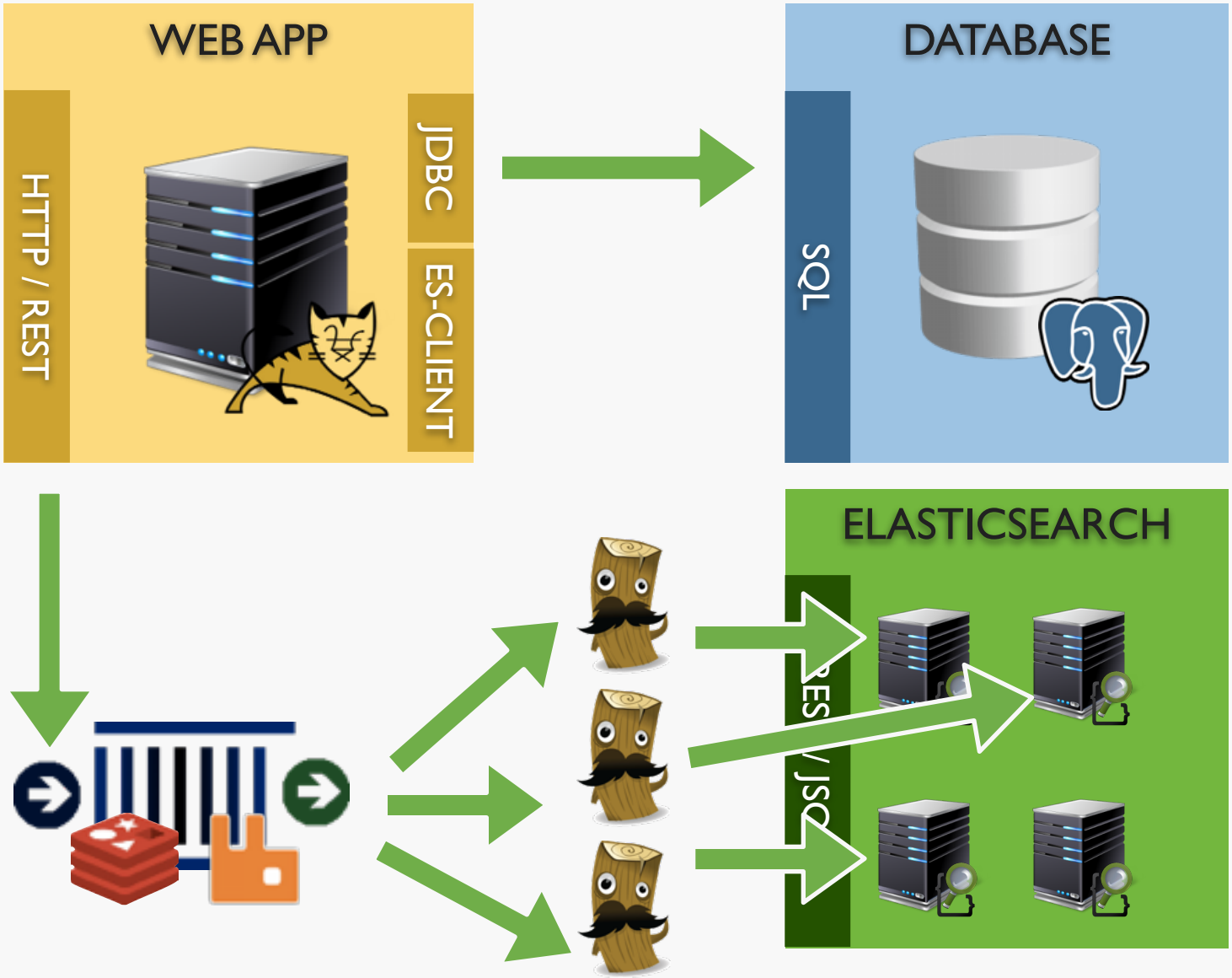
using brokers



using brokers



using brokers




COME TO THE DARK SIDE
WE HAVE COOKIES!


kibana
HITS
England (25448)
France (25395)
30000
25000
20000
15000
10000
5000
0
HISTOGRAM
View
Zoom Out
England (25448)
France (25395)
count per 1y | (50843 hits)
500
400
300
200
100
0
1950-01
1960-01
1970-01
1980-01
1990-01
2000-01
TERMS
female (25854)
male (24969)
male
49%
female
51%
MAP
TABLE
0 to 100 of 500 available for paging
name
gender
dateOfBirth
children
address.country
address.city
marketing.cars
marketing.food
Rami Honore male 1957-04-22 0 England Liverpool 188 613
Balthazar Reda male 1985-10-21 0 England Plymouth
Liv Amy female 1952-11-20 4 England Plymouth 723
Filip Adrian male 1973-11-10 0 England Plymouth
Felicie Kadiatou female 1972-08-09 3 England Plymouth 1388
elisa Zachary female 1996-08-22 0 England Plymouth
Sofia Dries female 1956-03-09 2 England Liverpool
Matthias Abdoul male 1979-09-07 1 England London 1876
Jayden Flora male 1942-07-03 0 England Plymouth
Jonah Charlene male 1961-12-19 2 England London
Layla Mae female 1963-11-05 0 England Plymouth

David Pilato
Technical advocate

elasticsearch.
@dadoonet

http://elasticsearch.com/support/

Questions?

jobs@elasticsearch.com