Python — Django
- web 1
- SQL 1
- ...
- ...
- HTML rendering 1

Flask

almost impossible
hard to replace

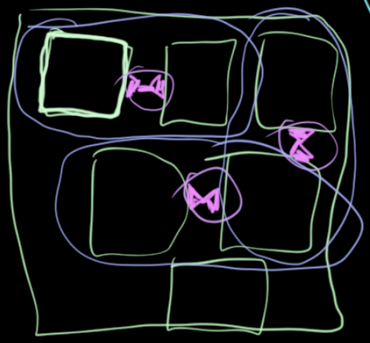
you are limited to
django's opinion

Java
- web
- db
- templ.

SpringBoot
- SQL
- ...
- SQL

Tomcat, Jetty] Web Servers

Spring ≡
set of "connectors"



HTML render
- ...
- Web 1
- Web 2
- Web 3

Web Server

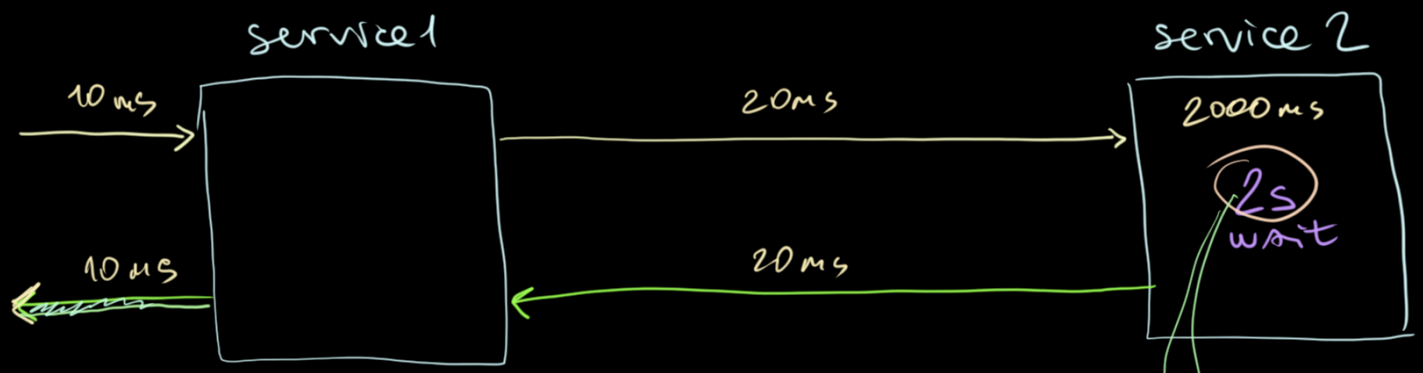
f: Request => Response

db access: conn → read / write

web rendering

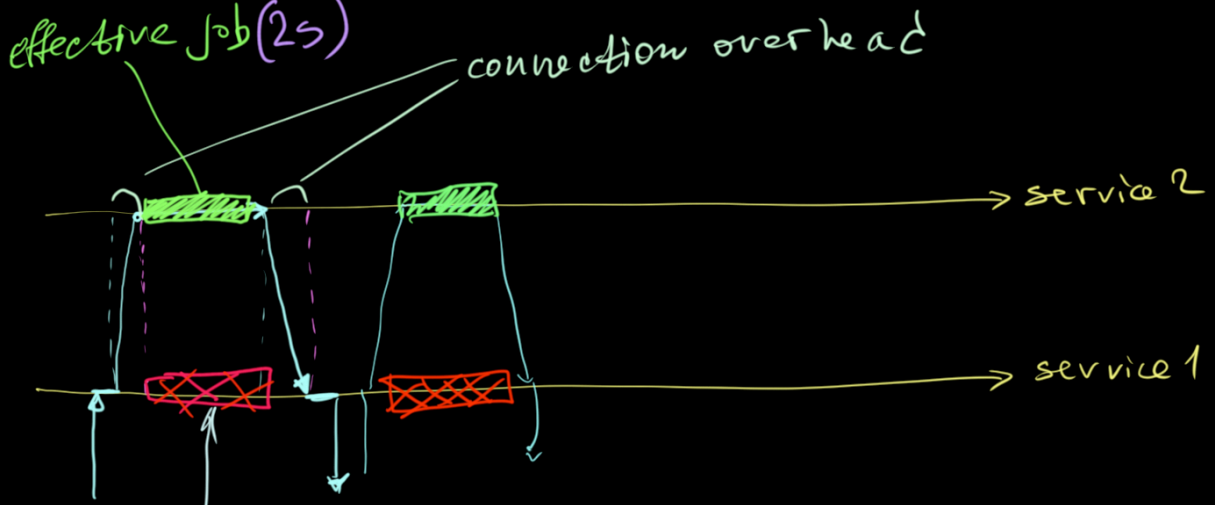
f(template, data) => HTML page
w/ place holder

synthetic



total time =

10 + 20 + 2000 + 20 + 10 ms x 2060



we waste this time because we simple wait

```
ab -n 10 http://localhost:8081/\?name\=Alex

Concurrency Level:      1
Time taken for tests:   20.081 seconds
```

(2s x 10) / 1 = 20s

```
ab -c 2 -n 10 http://localhost:8081/\?name\=Alex
```

(2s x 10) / 2 = 10s

```
Concurrency Level:      2
Time taken for tests:   12.057 seconds
```

```
ab -c 3 -n 10 http://localhost:8081/\?name\=Alex

Concurrency Level:      3
Time taken for tests:   8.027 seconds
```

20/3 ≈ 7

```
ab -c 6 -n 10 http://localhost:8081/\?name\=Alex

Concurrency Level:      6
Time taken for tests:   6.023 seconds
```

(2s x 10) / 6 ≈ 4s

```
ab -c 10 -n 10 http://localhost:8081/\?name\=Alex
```

(2s x 10) / 10 = 2s

```
Concurrency Level:      10
Time taken for tests:   4.018 seconds

ab -c 20 -n 20 http://localhost:8081/\?name\=Alex
```

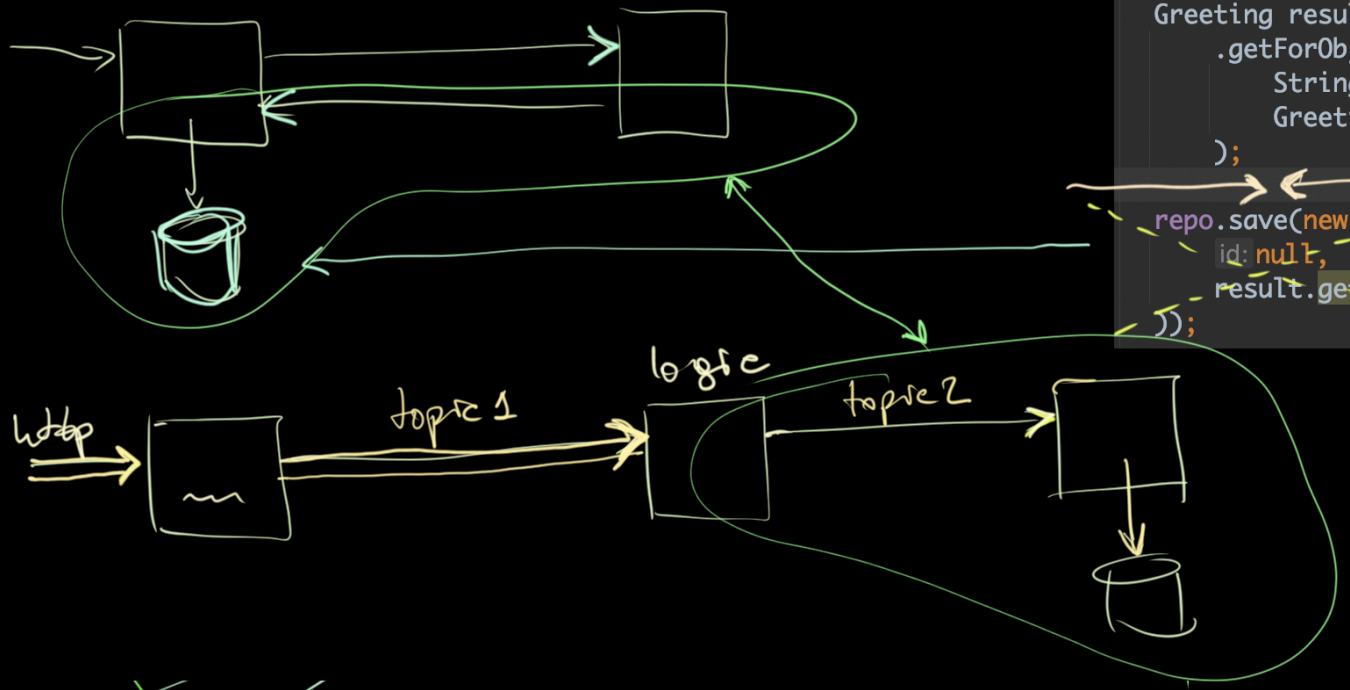
(2s x 20) / 20 ≈ 2s

```
Concurrency Level:      20
Time taken for tests:   4.023 seconds

ab -c 100 -n 100 http://localhost:8081/\?name\=Alex

Concurrency Level:      100
Time taken for tests:   4.060 seconds
```

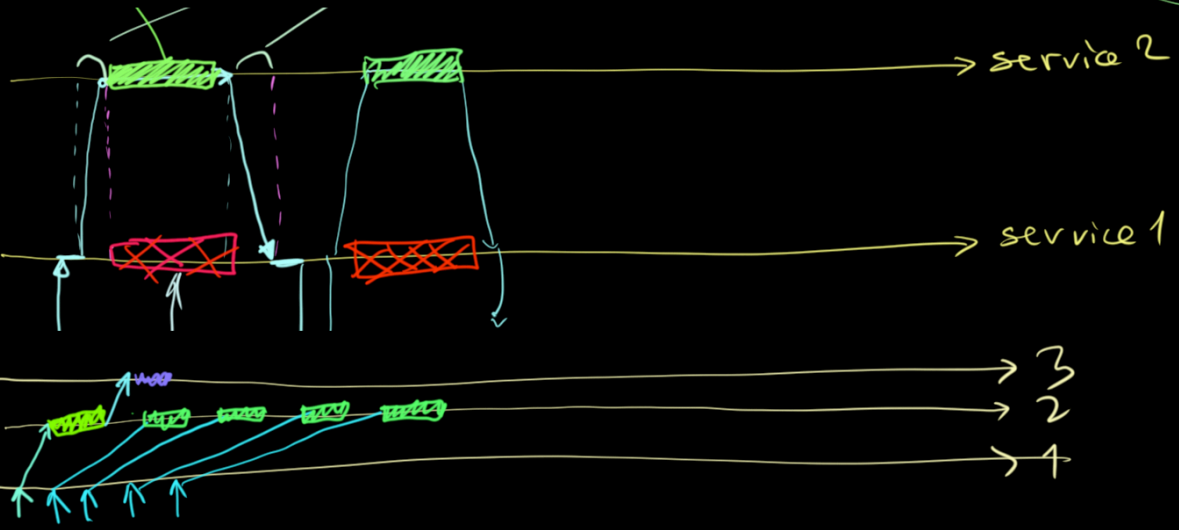
1. concurrency isn't free
2. we have limit of concurrent jobs



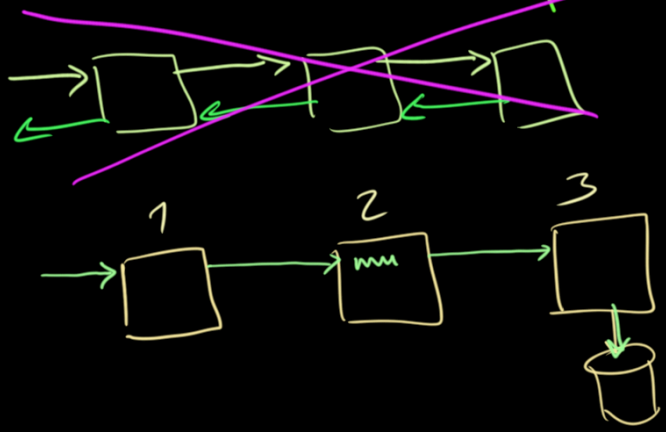

```
Greeting result = rest
    .getForObject(
        String.format("http://localhost:8082?name=%s", name),
        Greeting.class
    );

repo.save(new Message(
    id: null,
    result.getMessage()
));
```

if I want
not to wait
can I write somehow



can I use http for that?



computation

```
Greeting result = rest
    .getForObject(
        String.format("http://localhost:8082?name=%s", name),
        Greeting.class
    );
```

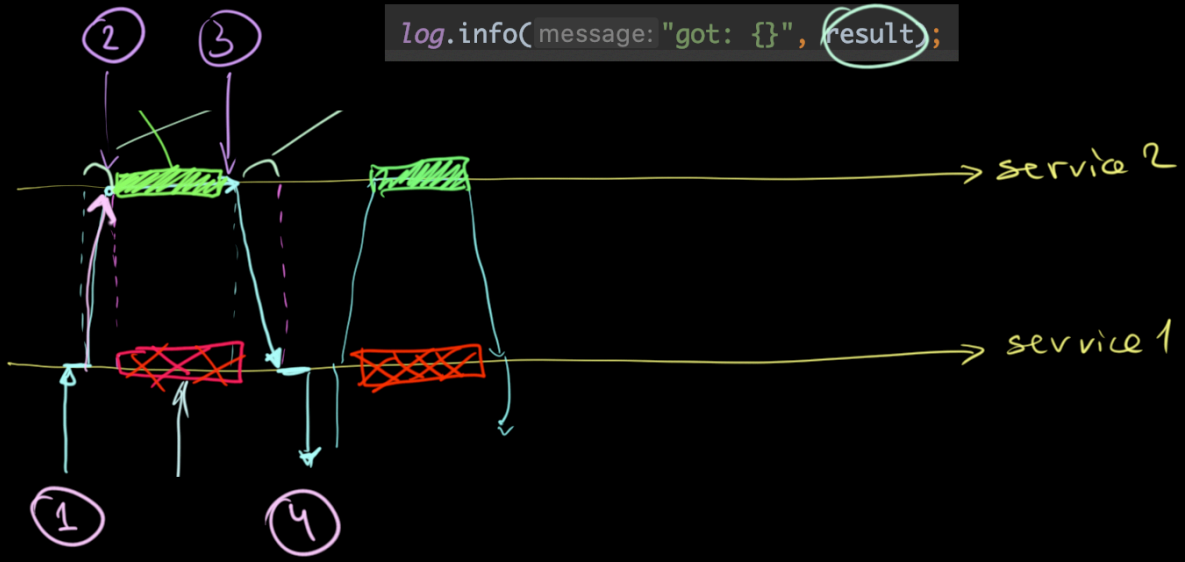
wait

```
@GetMapping("/")
public Greeting handle(@RequestParam("name") String name) t
    Thread.sleep( millis: 2000);
    return new Greeting(String.format("2 Hello, %s!", name));
}
```

1 → 2 → 3 → 4

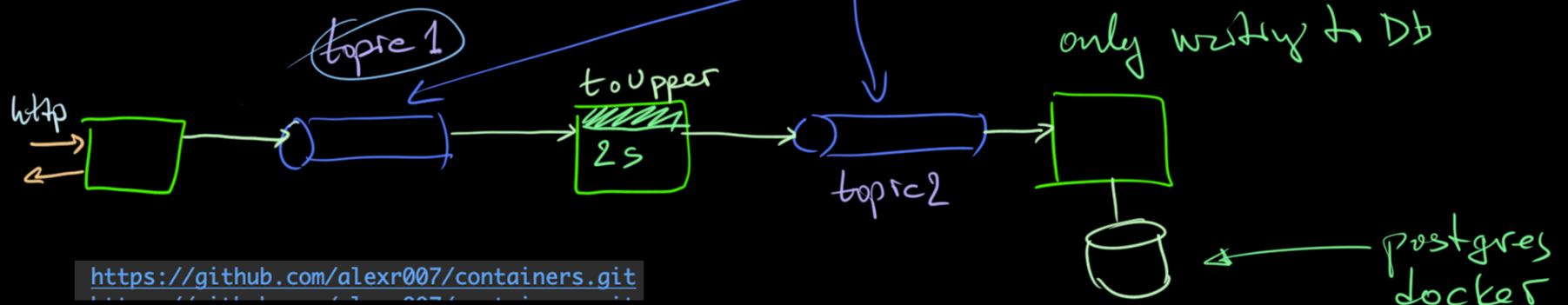
we wait because http client needs to get a response

```
log.info( message: "got: {}", result);
```



service 1
service 2

service 1a
service 2a
service 3a



https://github.com/alexr007/containers.git

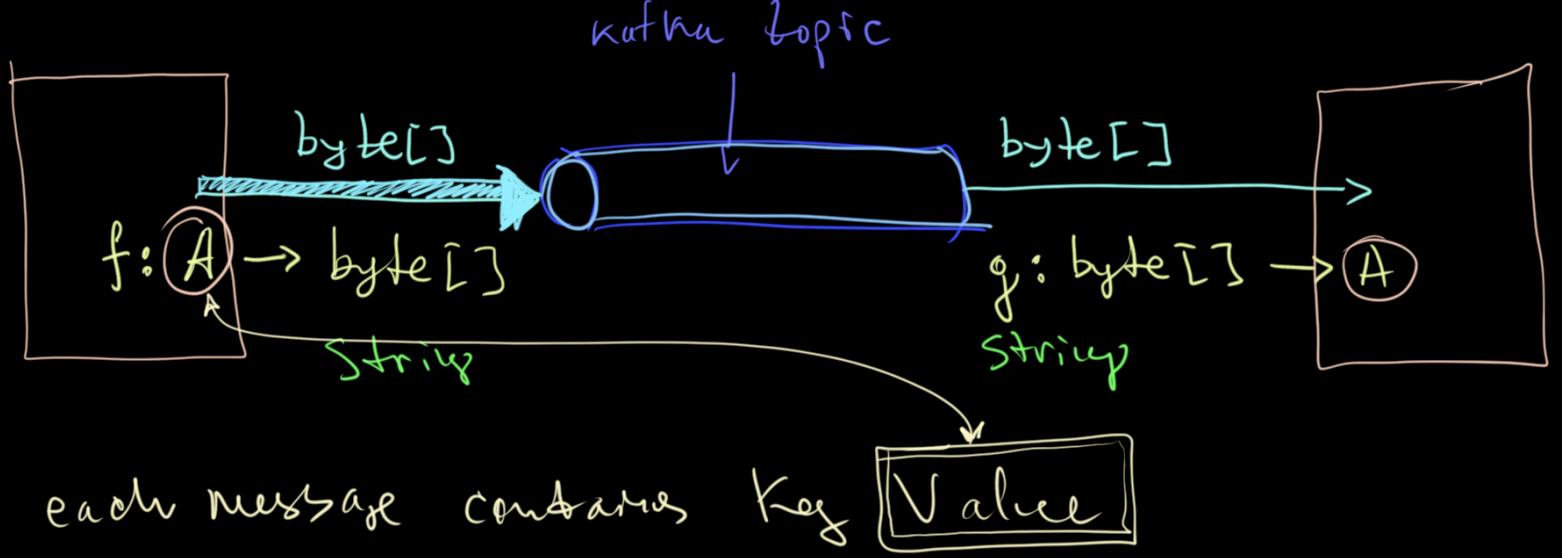
```yaml
version: '2.2'

services:
  zookeeper:
    image: zookeeper:3.6.2
    ports:
      - "2181:2181"

  kafka:
    image: wurstmeister/kafka:2.13-2.7.0
    ports:
      - "9092:9092"
    environment:
      KAFKA_ADVERTISED_HOST_NAME: localhost
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock

  wait-for-kafka:
    image: waisbrot/wait
    links:
      - kafka
    environment:
      - TARGETS=kafka:9092
```

```
docker-compose up
```



```
kafka.send(
    topic: "topic1",
    key,
    value
);
```

void

we don't want anything

will be run immediately
w/o waiting any processing

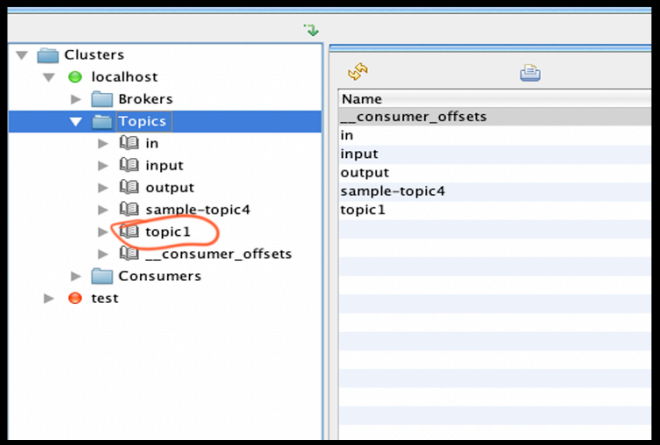
```
Concurrency Level:      10
Time taken for tests:   0.489 seconds
```

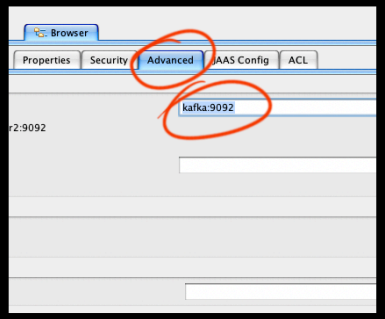
? where is my data ?

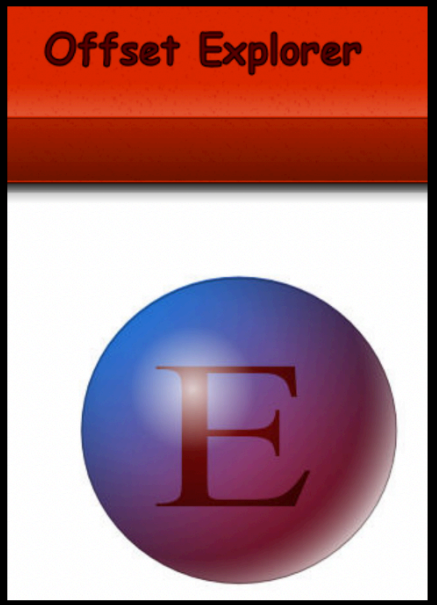
V inside kafka

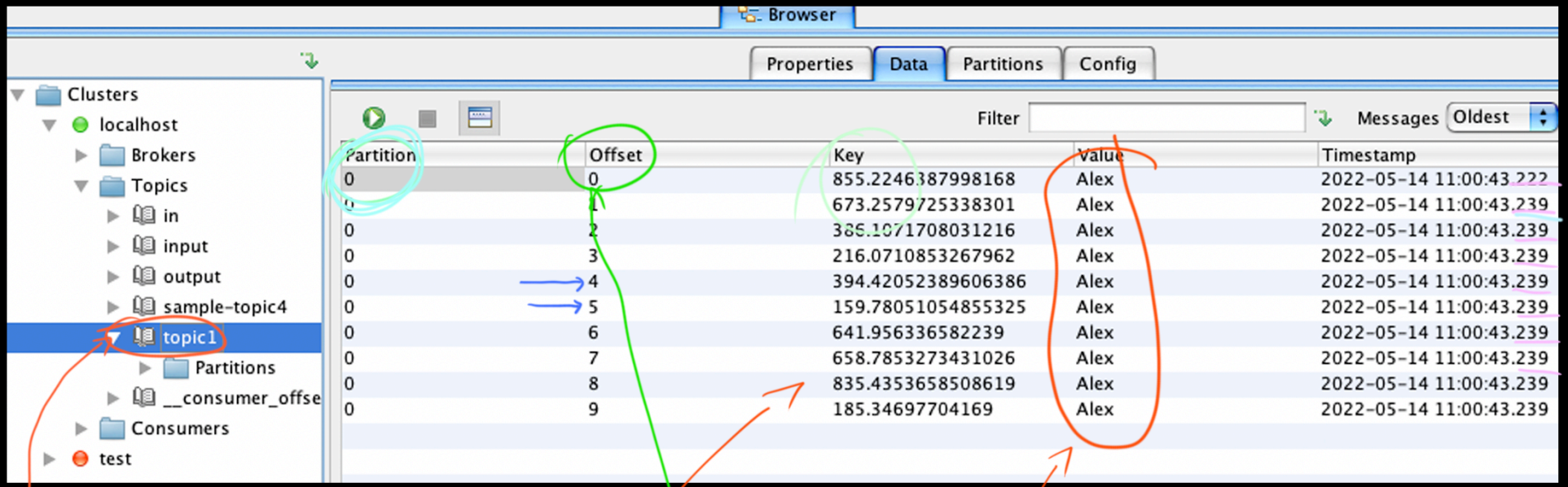
topic : "topic1"

https://www.kafkatool.com









Key is used as a correlation Id for grouping if you use many partitions

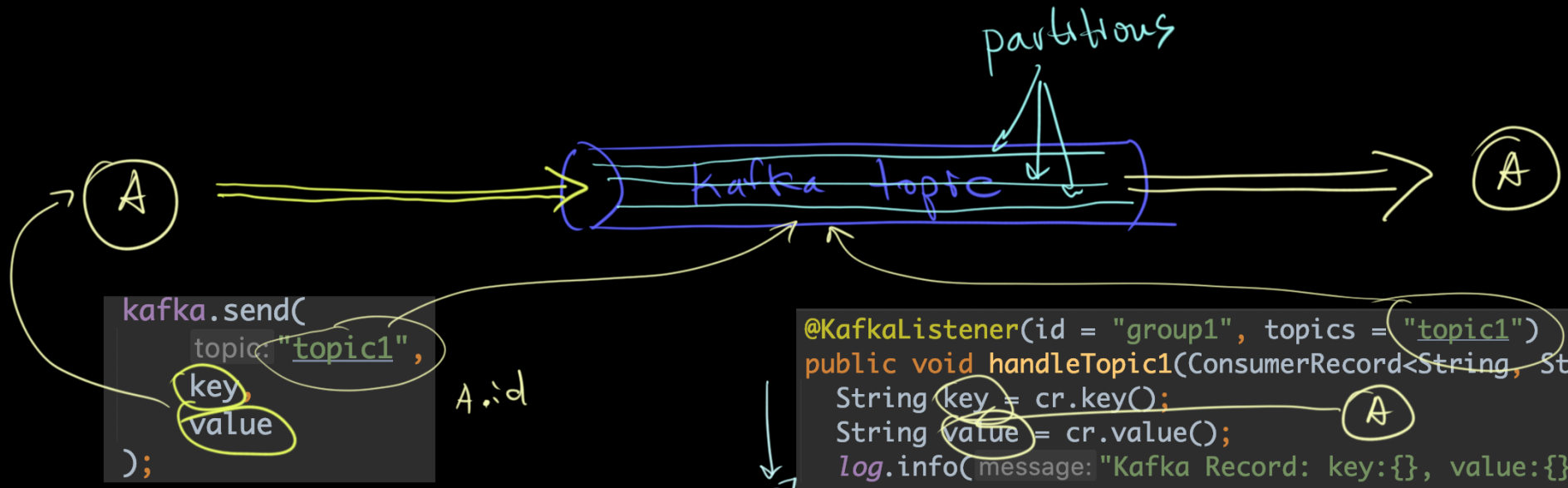
kind of id column in db

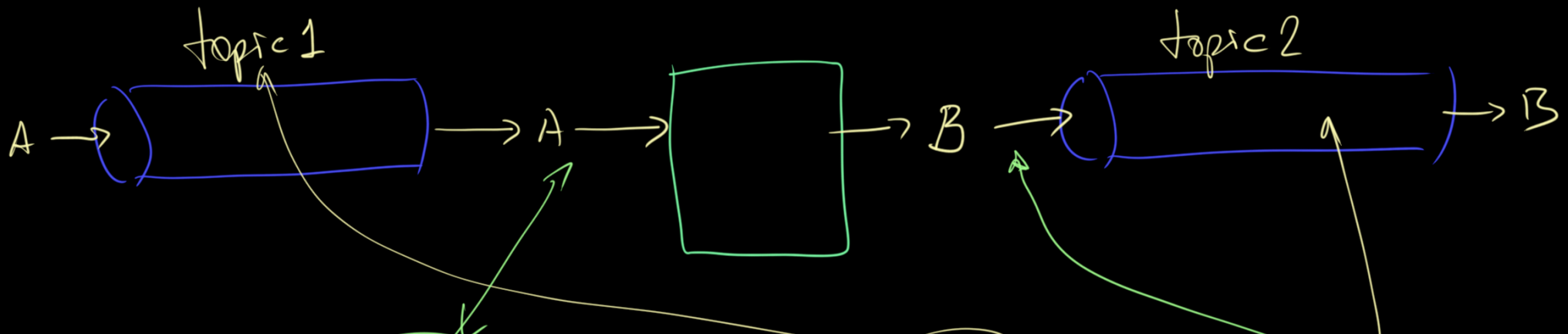
```
kafka.send(
    topic: "topic1",
    key,
    value
);
```

```
@KafkaListener(id = "group1", topics = "topic1")
public void handleTopic1(ConsumerRecord<String, String> cr, Acknowledgment ack) {
  String key = cr.key();
  String value = cr.value();
  log.info(message: "Kafka Record: key:{}, value:{}", key, value);
  ack.acknowledge();
}
```

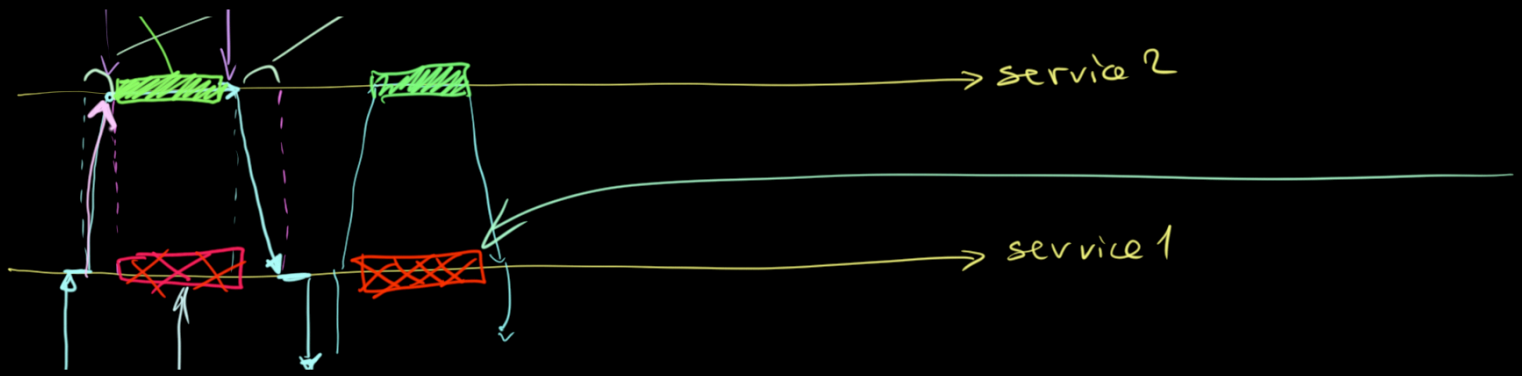
you need to switch off autocommit

manual offset shift

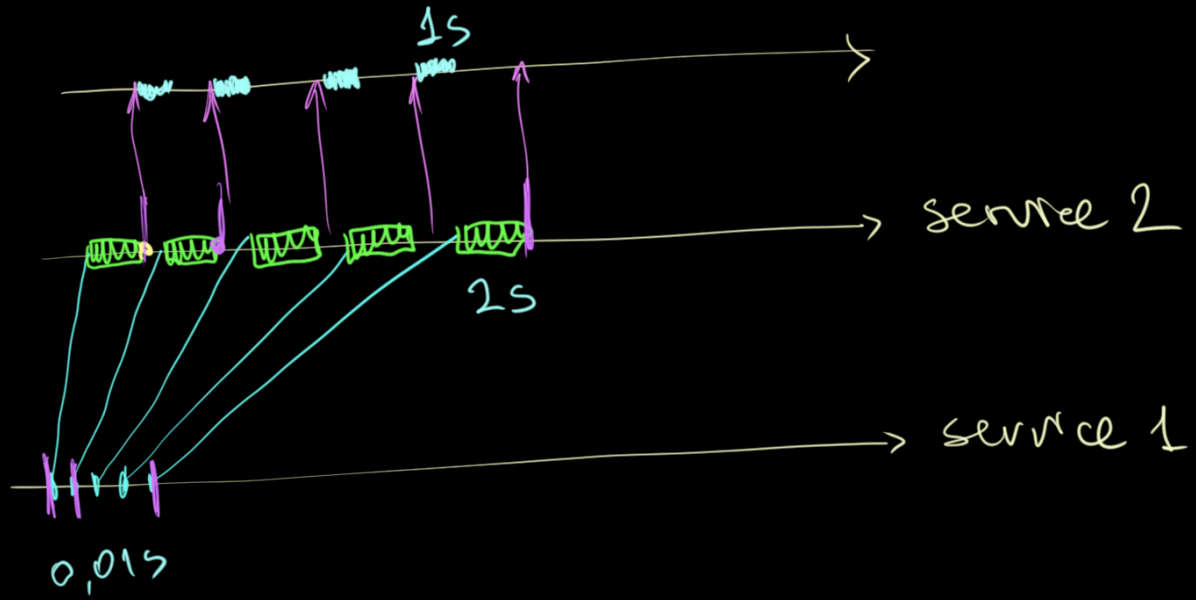

partitions
Kafka topic
A
A
kafka.send(
topic: "topic1",
key,
value
);
A:id
@KafkaListener(id = "group1", topics = "topic1")
public void handleTopic1(ConsumerRecord<String, String> cr) {
String key = cr.key();
String value = cr.value();
log.info(message: "Kafka Record: key:{}, value:{}", key, value);
}
A
after consuming we automatically "commit" offset
data can be consumed ONCE
you can say go to OFFSET (10)



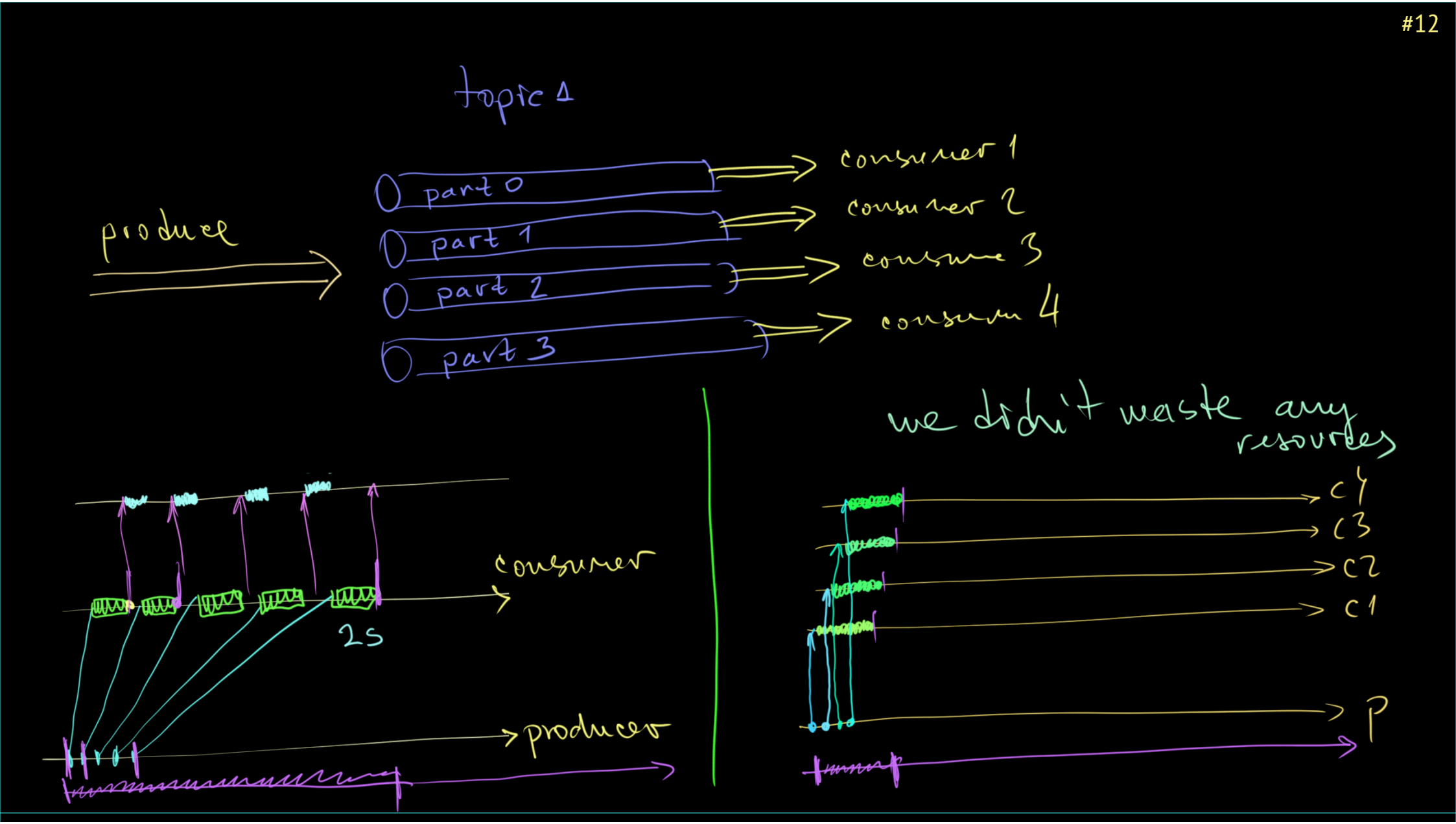

```
@KafkaListener(id = "group1", topics = "topic1")
public void handleTopic1(ConsumerRecord<String, String> cr) throws In
  String key = cr.key();
  String value = cr.value();
  log.info(message:"Got Kafka Record: key:{}, value:{}", key, value)
  log.info("doing business...");
  Thread.sleep(millis:2000);
  String valueProcessed = value.toUpperCase();
  log.info("... done business");
  publisher.send(topic:"topic2", key, valueProcessed);
}
```

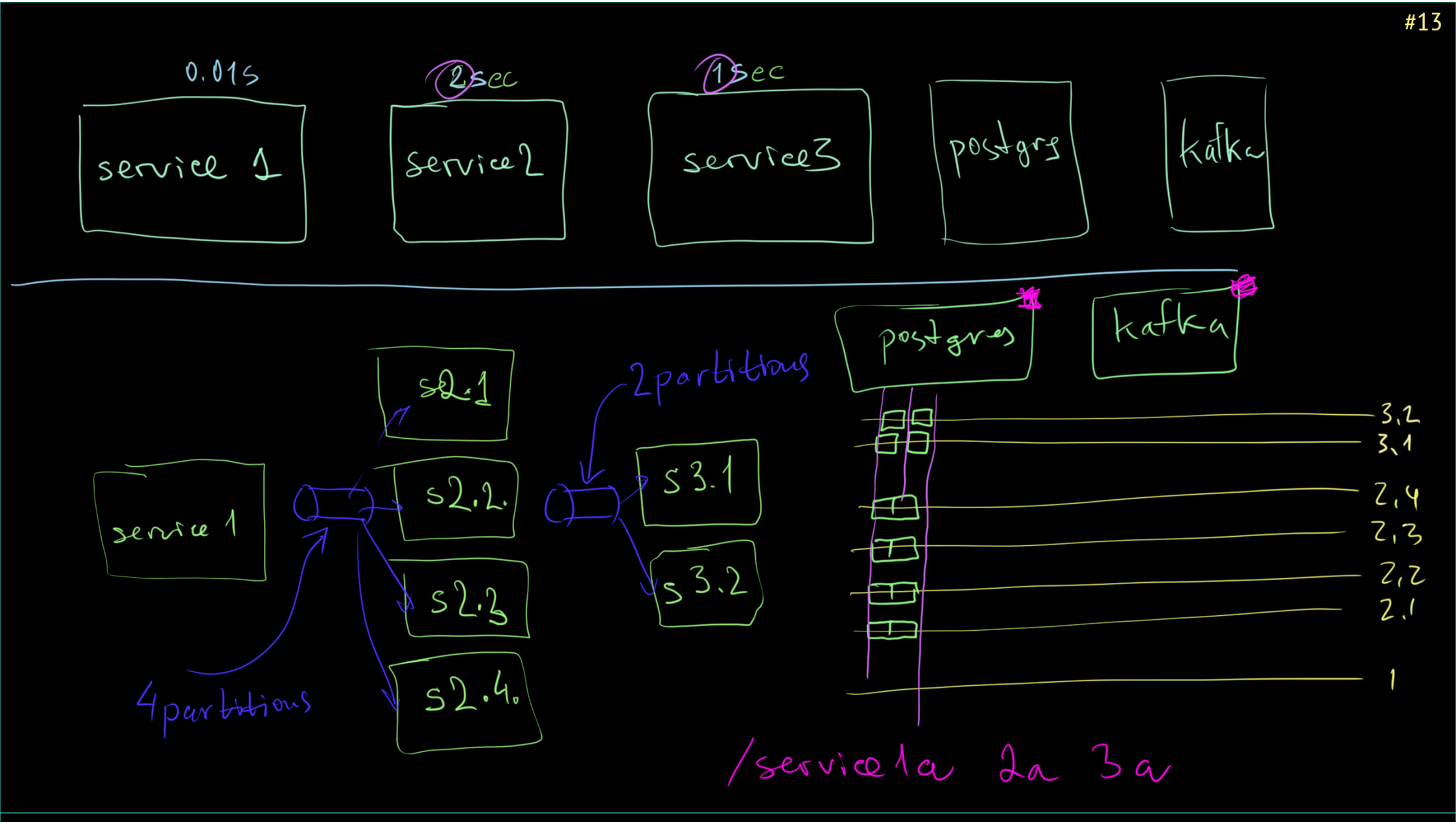


we learned how to avoid this waste

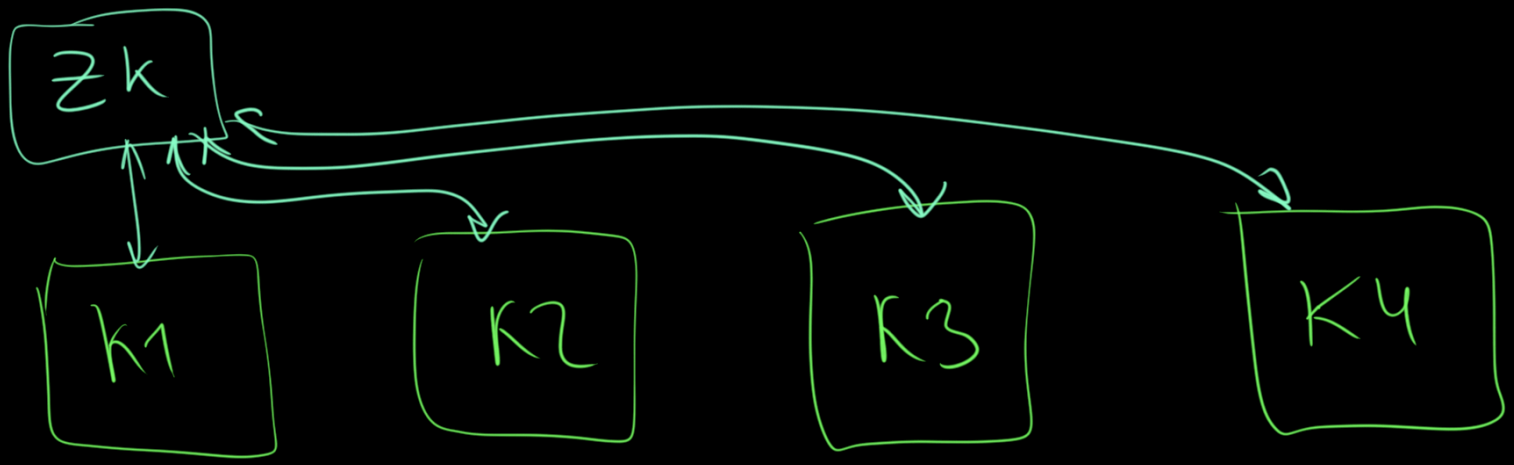


service 1 doesn't waste the time

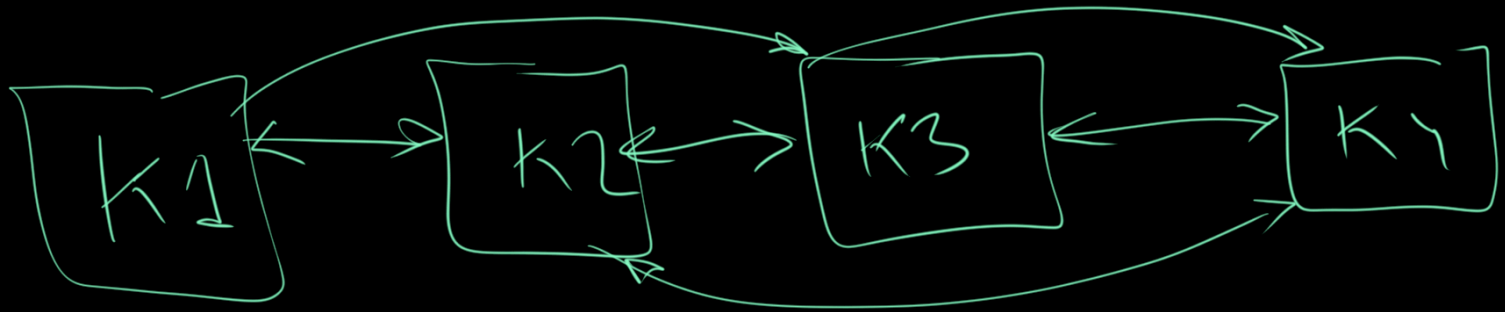

topic A
producer
part 0
part 1
part 2
part 3
consumer 1
consumer 2
consumer 3
consumer 4
consumer
2s
producer
we didn't waste any resources
C4
C3
C2
C1
P

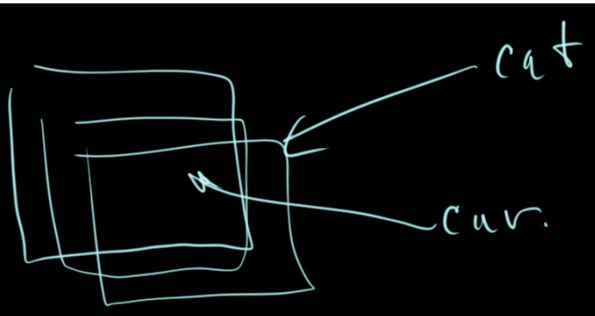

0.01s
2sec
1sec
service 1
service2
service3
postgrs
kafka
postgres
kafka
s2.1
s2.2
s2.3
s2.4
2partitions
s3.1
s3.2
service 1
4partitions
3,2
3,1
2,4
2,3
2,2
2,1
1
/service1a 2a 3a

2.x



3.x

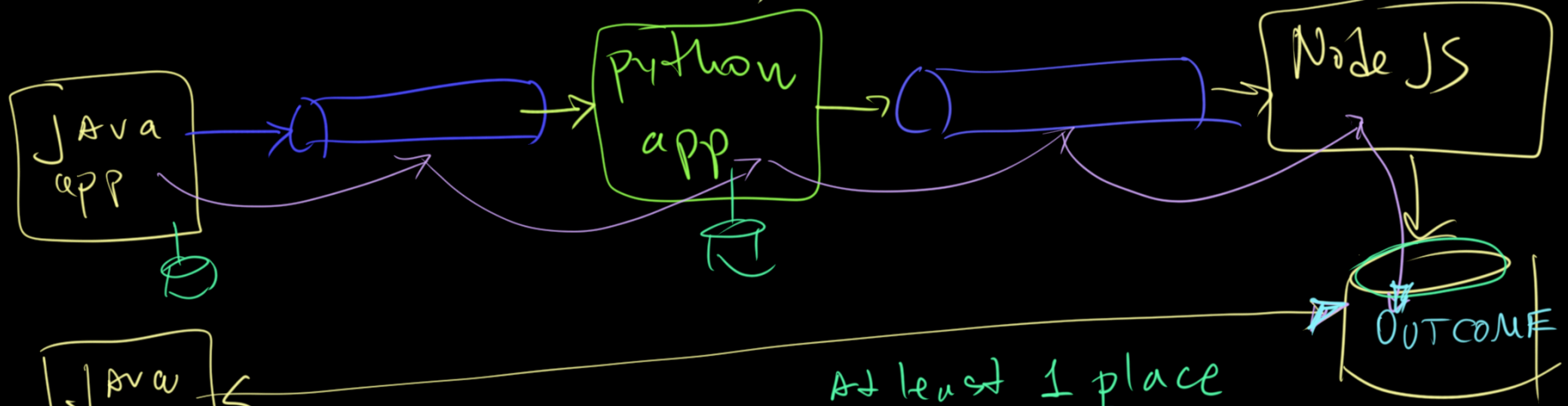




billions of images

f:(image) => cat → 0.7
dog → 0.2
car → 0,05
ship → 0,05

f: (A) → List[(String, Double)]



At least 1 place
More than 1 app