# Spring - Mar 3

Interview Questions:

IQ: If create a spring/spring cloud, what are the main components or main services you need to create?

*We can talk about:*

- Eureka
- Ribbon
- Config Server
- ...

The main idea of MicroServices, basically to create a bigger project with a small small components. It's not a good idea to put all the code into one monolithic architecture.

Monolithic:

everything in one package

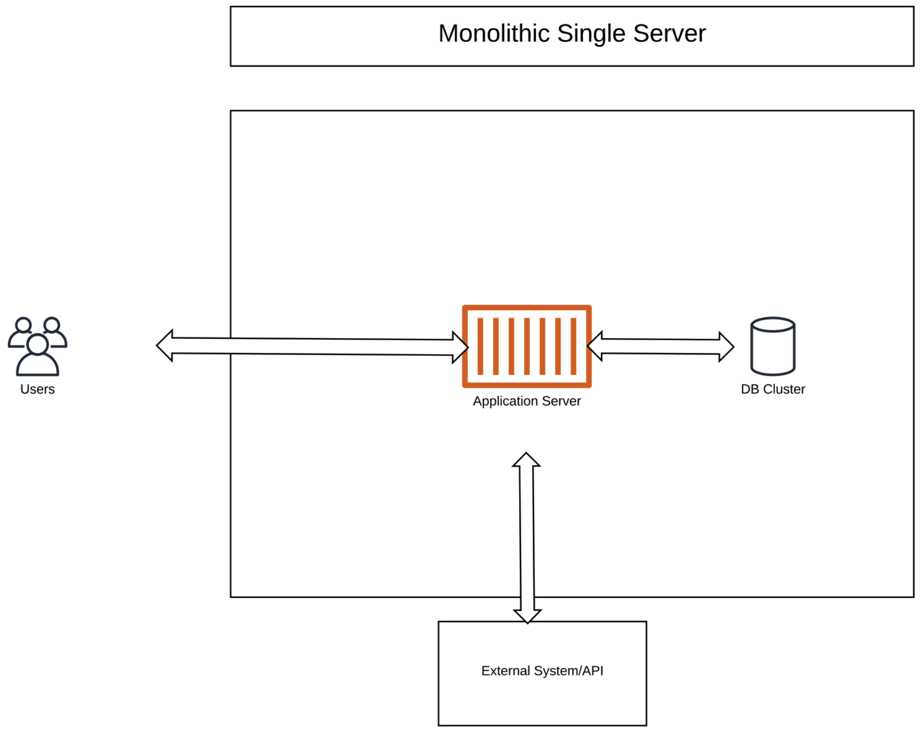
Benefit:

can scan all the things of the project in the package

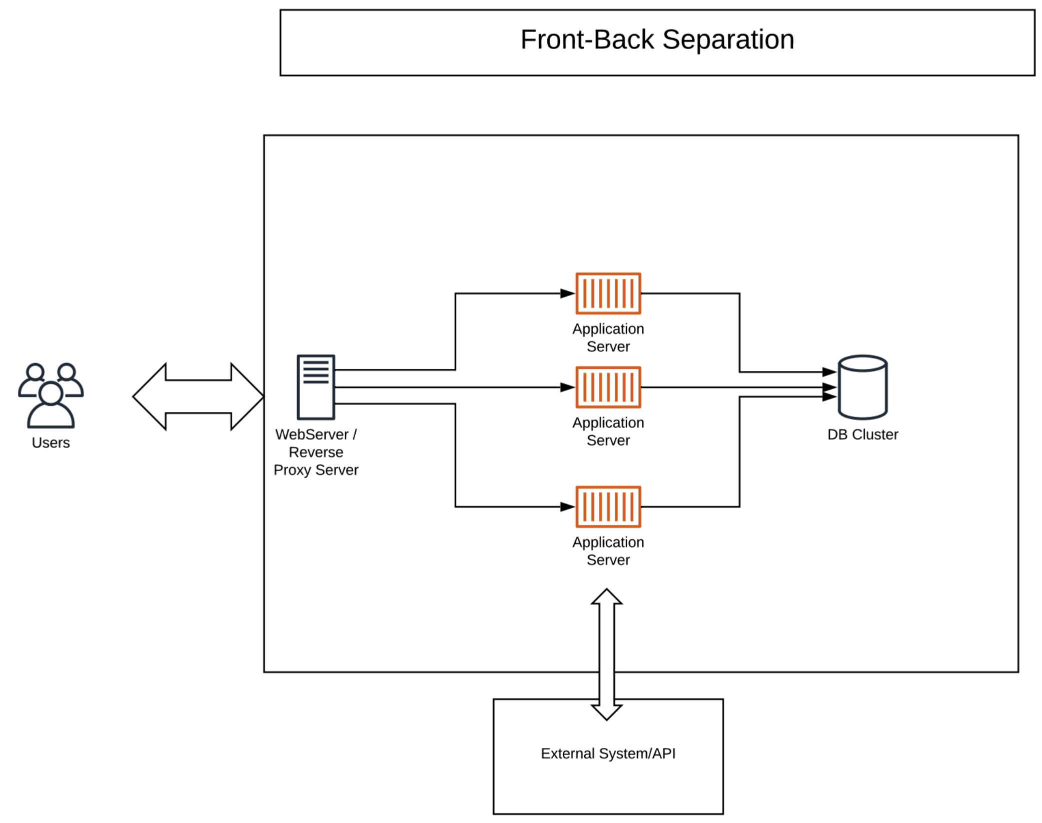
IQ: What's the architecture you work on? MicroServices, Monolithic or others?

**Review!!!!!**

1. Monolithic Single Server



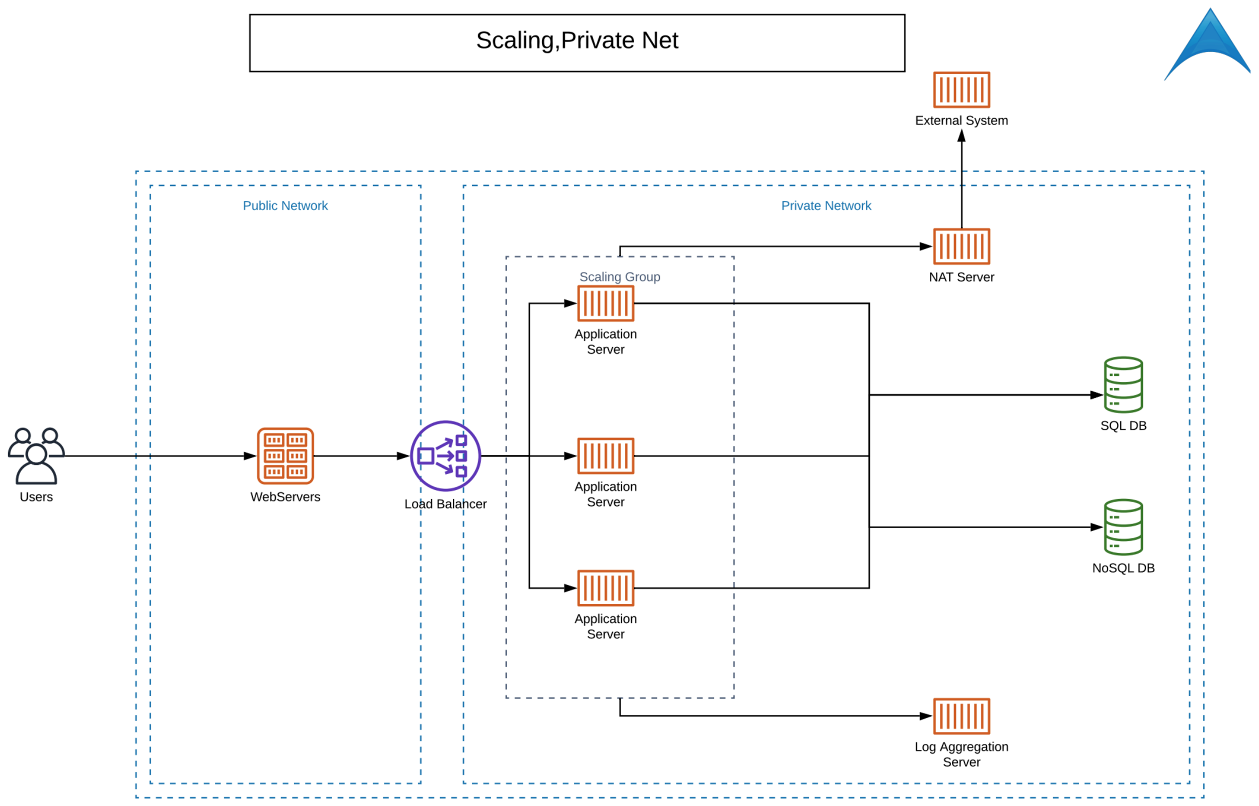
2. Front-Back Separation architecture



NGINX for front-end/reverse proxy server → load balanced

- (not Java)

3. Segregation of the network



NAT(Network address translation)

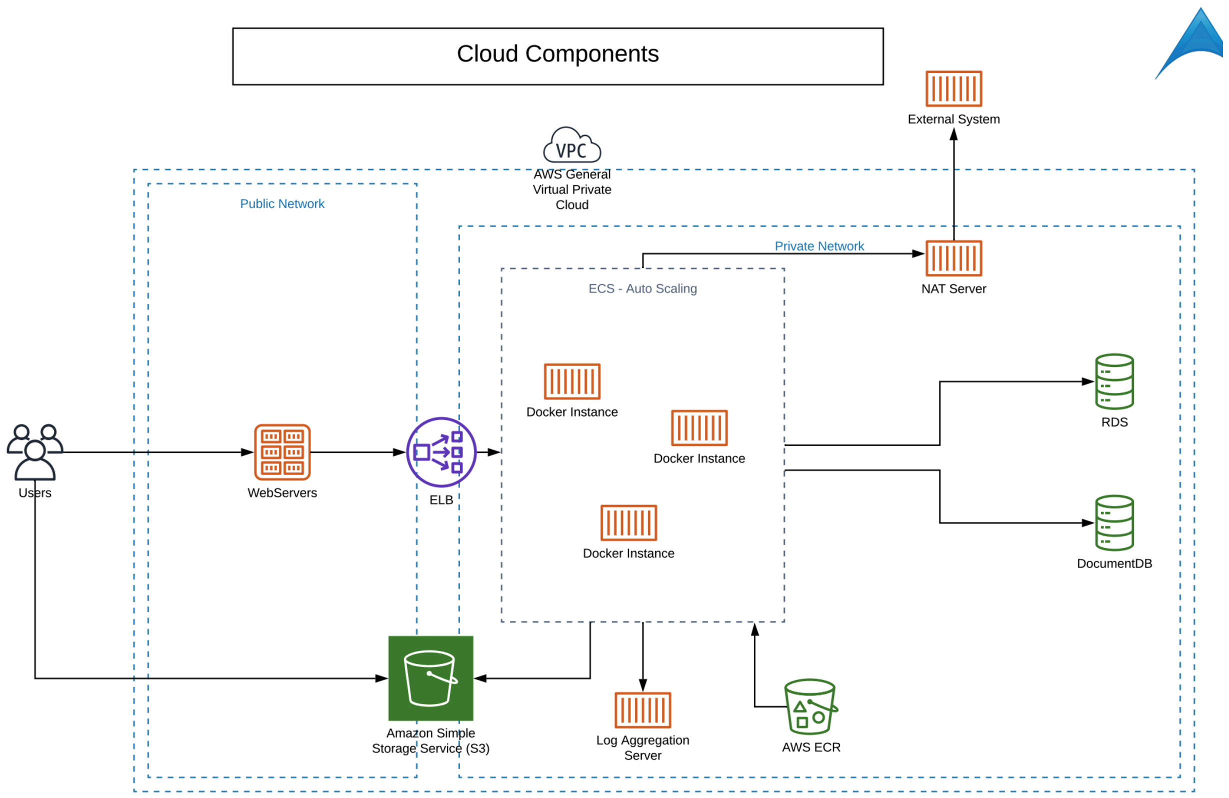
- routing and switching
- all the private Network share one public IP

00:14:56

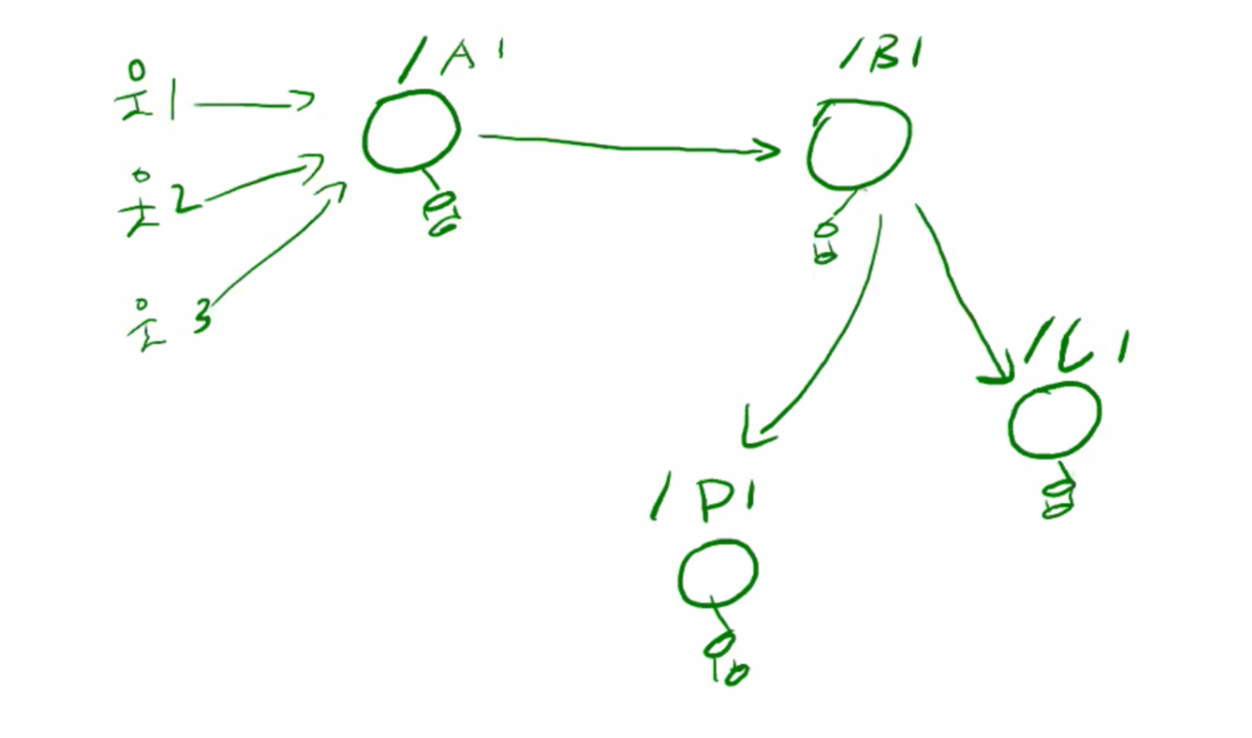
In this case

- use can not directly connect to the servers
- servers can use NAT to connect external system
- public part can be accessed by any one(risk)
- the Load Balancer, some times have firewall in and out
- Log Aggregation Server monitor any exceptions/error, and send message/email to maintenance team for log files

## 4. Cloud Components



e.g. for the logs



Users → A → B → C

→ D

A, B, C and D only print out related logs with themselves

Case: When users somehow say, in one time before service become slow. Maintenance team will be had to find out problem → lose track

- Monolithic has not this problem, cuz all things are in one JVM
- Separate System loses tracks of transactions

Introduce:

- Sleuth
    - run in every individual service
    - record trace ID and span ID of Http Header
    - use Track ID to identify same transaction
- Zipkin server collects all logs and link them together

    - what’s the ip address
    - what’s the time usage
    - Zipkin is no longer supported by Spring Boot

ELK

elastic search → aggregate logs