Beaconfire Inc, Home Work, Week7 Day28.
By (Ping) Nalongsone Danddank.
ndanddank@gmail.com
wechatID: ndanddank
Short Answer:

1. In your own word, please describe some of the advantages and disadvantages of a Monolithic Application.

-> Advantages: It's simple and convenience for small scale project and small developer team. like, we can use IDEs tool to work with the project and developing all access to all file more flexible. Then when we want to deploy our project or War file to web service like AWS, should be more handy. On another hand, it's about scale, so we can scale the application by running copies of the app behind a load balancer. Sp we can import performance by multi-threading, Offine pricesing...

-> Disadvantages: what if the number of our customer increased and our application reached its limit? And once the application becomes large and the team grows in size, So we get problems like the large monolithic project and hard to separate task to many developers, especially ones who are new to the team. And it is overloaded for IDE and web container, so take too long time to deploy, hard to put a new technologies to the project, and one error takes down the entire application also down too.

scaling on vertical and horizontal. If we take vertical scaling like put more powerful machine that is more expensive and always have a limit. The, horizontal scaling, it is cheaper than vertical scaling, and it is the best way to do, however, how does the client-side application know which server-side instance to request to? We also know as Load Balancer: distributing incoming network traffic across a group of backend servers. It will route the client request to the mosts appropriate backend server which maximize speed, capacity utilization and ensures no server is over-worked. No impact on the performance

2. In your own word, please describe some of the advantages and disadvantages of Microservice Application.

-> Advantages: we can separate our big application to be many small one and multiple service or API into different parts. teams can develop, maintain, and deploy each microservice independently. isolated services have a better failure tolerance. Then, each service have its own host name, port, application context, entry points, database connection, and technology stack. So, we can use Service Discovery server like Netfix Eureka, to allow services to find and communicate with each other without hard-coding hostname and port.
An architectural style that structures an application as a collection of services that are
• Highly maintainable and testable
• Loosely coupled
• Independently deployable
• Organized around business capabilities
• Services communicate using either synchronous protocols such as HTTP/
REST or asynchronous protocols such as AMQP

-> Disadvantages: microservices heavily rely on messaging. great for companies who already use these methods, but not for smaller companies. Harder to test and monitor because of the complexity of the architecture. Harder to maintain the network (has less fault tolerance, needs more load balancing, etc.) Security issues (harder to maintain transaction safety, distributed communication goes wrong more likely, etc.)

3. What is the purpose of using Netflix Eureka?

-> Netfix Eureka, to allow services to find and communicate with each other without hard-coding hostname and port. And it provides some functions like we can register itself to a service registry by sending a heartbeat

signal to it. And a service can retrieves a list of all connected services from the service registry and makes all further requests to any other services.

4. How can microservices communicate with each other?
-> we can use Feign that declarative HTTP client developed by Netflix. It help us to simplify HTTP API clients. We just need only declare and annotate an interface while the actual implementation will be provisioned at runtime (Maker Interface). without Feign we would have to autowire an instance of EurekaClient into our controller with which we could receive a service-information by service-name as an application object.

5. What is the purpose of using Spring API Gateway?
-> it can provide the single entry point for all clients. Because when we have too many services and too many different URL, API Gateway can help us to know which URL map to which service or application. Then it can handles requests in one of two ways like, simple proxied/routed to the appropriate service or by fanning out to multiple services.