**1. What is Spring Cloud?**  
The Spring Cloud Streaming Application Launcher is a Spring integrated application in Spring Boot that provides integration with external systems. Spring cloud Task, a microservice framework with a short life cycle, is used to quickly build applications that perform limited data processing.

**2. What are the advantages of using Spring Cloud?**  
When using Spring Boot to develop distributed microservices, we face the following problems

1. Complexity associated with distributed systems-This overhead includes network issues, delay overhead, bandwidth issues, and security issues.
2. Service discovery-Service discovery tools manage how processes and services in the cluster find and talk to each other. It involves a service directory, register services in the directory, and then be able to find and connect to the services in the directory.
3. Redundancy-Redundancy in distributed systems.
4. Load balancing-Load balancing improves the workload across multiple computing resources, such as the distribution of computers, computer clusters, network links, central processing units, or disk drives.
5. Performance-The problem is caused by various operational expenses.
6. Requirements for deploying complex evops skills.

**3. What does service registration and discovery mean? How is Spring Cloud implemented?**  
When we start a project, we usually do all the configuration in the properties file. As more and more services are developed and deployed, adding and modifying these attributes becomes more complicated. Some services may decline, while certain locations may change. Manually changing attributes may cause problems. Eureka service registration and discovery can help in this situation. Since all services are registered on the Eureka server and found by calling the Eureka server, there is no need to deal with any changes and processing of the service location.

**4. What is the meaning of load balancing?**  
In computing, load balancing can improve workload distribution across multiple computing resources such as computers, computer clusters, network links, central processing units, or disk drives. Load balancing is designed to optimize resource usage, maximize throughput, minimize response time, and avoid overloading any single resource. Using multiple components for load balancing instead of a single component may improve reliability and availability through redundancy. Load balancing usually involves dedicated software or hardware, such as multi-layer switches or domain name system server processes.

**5. What is Hystrix? How does it achieve fault tolerance?**  
Hystrix is a delay and fault-tolerant library designed to isolate access points of remote systems, services, and third-party libraries. When failures are inevitable, stop cascading failures and achieve resilience in complex distributed systems. Usually for systems developed using microservices, many microservices are involved. These microservices collaborate with each other. Consider the following microservices

Suppose if the microservice 9 in the above figure fails, then using the traditional method we will propagate an exception. But this will still cause the entire system to crash. As the number of microservices increases, this problem becomes more complicated. The number of microservices can be as high as 1000. This is where hystrix appears. We will make Hystrix’s Fallback method function in this case. We have two services employee-consumer using the services exposed by employee-consumer. The simplified diagram is shown below

Now suppose that the service exposed by the employee-producer will throw an exception for reasons. In this case, we use Hystrix to define a fallback method. This fallback method should have the same return type as the public service. If an exception occurs in the exposed service, the fallback method will return some values.

**6. What is a Hystrix circuit breaker? Do we need it?**  
For some reasons, the employee-consumer public service will raise an exception. In this case using Hystrix we define a fallback method. If an exception occurs in the public service, the fallback method returns some default values.

If the abnormality in the ﬁrstPage method() continues to occur, Hystrix calls and the employee user will skip the ﬁrtsPage method together and directly call the fallback method. The purpose of the circuit breaker is to allow time for the first method or other methods that may be called by the first page method, and cause abnormal recovery. What may happen is that under light load conditions, there is a better chance of recovery from the problem that caused the exception.

**7. What is Netﬂix Feign? What are its advantages?**  
Feign is a java client binding program inspired by Retrofit, JAXRS-2.0 and WebSocket. Feign’s first goal is to unify the complexity of the constraint denominator into http apis, regardless of its stability. In the employee-consumer example, we used the emplo e-producer to use the REST service exposed by the REST template. But we have to write a lot of code to perform the following steps

* Use the functional area for load balancing.
* Get the service instance, and then get the basic URL.
* Use the REST template to use the service. The previous code is as follows

The previous code, with chances of exceptions like NullPointer, is not optimal. We will see how to use Netﬂix Fe n to make calling easier and cleaner. If Netﬂix Ribbon depends on the dependency path, Feign will also load balance by default.

**8. What is Spring Cloud Bus? Do we need it?**  
Consider the following scenario: We have multiple applications that use Sprng Cloud Conﬁg to read properties, while SpRing Cloud Conﬁg reads these properties from GIT.

In the following example, multiple employee producer modules obtain Eureka registered property from the Employee Conﬁg Module

What happens if you assume that the Eureka registration attribute in GIT is changed to point to another Eureka server. In this case, we will have to restart the service to get the updated properties. There is another way to use actuator endpoints/refresh. But we will have to call this url separately for each module. For example, if Employee Producer1 is deployed on port 8080, call http: //localhost:8080/refresh. Also for Employee Producer2 http: //localhost:8081 /refresh and so on. This is troublesome again. This is where Spring Cloud Bus comes into play.

Spring Cloud Bus provides the ability to refresh the configuration across multiple instances. Therefore, in the above, if we refresh Employee Producer1, it will automatically flash all other necessary modules. This is especially useful if we have a microservice up and running. This is achieved by connecting all microservices to a single message broker. Whenever an instance is refreshed, this event will be subscribed to all microservices listening to this agent, and they will also be refreshed. Can refresh any single instance by using endpoint/bus/refresh

**9. What is microservice**  
Microservice architecture is an architectural pattern or an architectural style. It advocates the division of a single application into a set of small services. Each service runs in its own process. The services coordinate and coordinate with each other. Cooperate to provide users with ultimate value. Lightweight communication mechanisms are used to communicate with each other (usually HTTP-based RESTful API). Each service is built around a specific business and can be independently constructed in a production environment or production-like environment. In addition, a unified and centralized service management mechanism should be avoided. For a specific service, the appropriate language and tools should be selected according to the business context to build it. There can be a very lightweight centralized management to To coordinate these services, you can use different languages ​​to write services, or you can use different data stores.

**10. What is a service fuse? What is service degradation**  
The fuse mechanism is a microservice link protection mechanism that responds to the avalanche effect. When a microservice is unavailable or the response time is too long, the service will be downgraded, and then the call of the microservice of the node will be fused, and the “error” response information will be quickly returned. After detecting that the node microservice call response is normal, the call link is restored. In the Spring Cloud framework, the fuse mechanism is implemented through Hystrix. Hystrix will monitor the status of calls between microservices. When a failed call reaches a certain threshold, the default is 20 calls within 5 seconds. If it fails, the fuse mechanism will be activated.  
Service degradation is generally considered from the overall load. That is, when a service is broken, the server will no longer be called, at this time the client can prepare a local fallback callback and return a default value. In doing so, although the level drops, it is still available anyway, which is better than just hanging it out.

**11. Both Eureka and zookeeper can provide service registration and discovery functions, please tell us the difference between the two?**  
Zookeeper guarantees CP (C: consistency, P: partition fault tolerance), Eureka guarantees AP (A: high availability)

(1) When querying the registration center for a list of services, we can tolerate the information returned by the registration center a few minutes ago, but we cannot tolerate the direct down and unavailability. That is to say, the service registration function has high requirements for high availability, but zk will have such a situation. When the master node loses contact with other nodes due to network failure, the remaining nodes will re-select the leader. The problem is that the leader selection time is too long, 30 ~ 120s, and the zk cluster is not available during the selection period, which will cause the registration service to be paralyzed during the selection period. In a cloud deployment environment, it is a high probability that the zk cluster loses the master node due to network problems. Although the service can be restored, the long-term selection time caused by the long-term unavailability of the registration cannot be tolerated.

(2) Eureka guarantees usability. Each node of Eureka is equal. If several nodes hang up, it will not affect the work of normal nodes. The remaining nodes can still provide registration and query services. Eureka’s client fails to register with Eureka or discovers that it will automatically switch to other nodes. As long as there is one Eureka still in place, the registration service is available, but the information found may not be the latest. In addition, Eureka also has a self-protection mechanism. If more than 85% of the nodes do not have a normal heartbeat within 15 minutes, then Eureka thinks that a network failure has occurred between the client and the registration center. At this time, the following situations will occur:

1. ① Eureka does not remove the service that should be expired because it has not received a heartbeat for a long time.
2. ② Eureka is still able to accept registration and query requests for new services, but it will not be synchronized to other nodes (ie to ensure that the current node is still available)
3. ③ When the network is stable, the new registration information of the current instance will be synchronized to other nodes. Therefore, Eureka can well cope with the situation where some nodes lose contact due to network failures, instead of paralyzing the entire microservice like Zookeeper.

**12. What is the difference between SpringBoot and SpringCloud?**  
SpringBoot focuses on the rapid and convenient development of individual microservices.

SpringCloud is a global microservice coordination and management framework that integrates and manages individual microservices developed by SpringBoot to provide various microservices, configuration management, service discovery, circuit breakers, routing, micro proxy, Event bus, global lock, decision campaign, distributed session, etc. Integration services SpringBoot can leave SpringCloud to use development projects independently, but SpringCloud is inseparable from SpringBoot and belongs to the dependency relationship.

SpringBoot focuses on the rapid and convenient development of individual microservices, and SpringCloud focuses on the overall governance framework

**13. What is a Hystrix circuit breaker? Do we need it**  
For some reasons, the employee-consumer public service will cause an exception. In the case of Hystrix, we have defined a fallback method. If an exception occurs in the public service, the fallback method returns some default values.

If the exception in the ﬁrstPage method() continues to occur, the Hystrix circuit will be interrupted, and the employee user will skip the ﬁrtsPage method together and directly call the fallback method. The purpose of the circuit breaker is to allow time for the first page method or other methods that the first page method may call, and cause abnormal recovery. What may happen is that under light load conditions, there is a better chance of recovery from the problem that caused the exception.

**14. Talk about the implementation principle of RPC**  
First, there is a module that handles network connection communication, which is responsible for connection establishment, management, and message transmission. Secondly, there needs to be a codec module, because the network communication is transmitted byte code, we need to serialize and deserialize the objects we use. The rest is the client and server side. The server side exposes the service interface to be opened. The client calls a proxy implementation of the service interface. This proxy implementation is responsible for collecting data, encoding and transmitting it to the server and waiting for the result to return.

# 15. The advantages and disadvantages of microservices? Let me talk about the pits encountered in development projects? advantage:

1. Each service is directly cohesive enough, the code is easy to understand
2. High development efficiency, one service only does one thing, suitable for small team development
3. Loosely coupled, functionally meaningful services.
4. Can be developed in different languages, oriented to interface programming.
5. Easy third-party integration
6. Microservices are just the code of business logic, not with HTML, CSS or other circles
7. Can be flexibly matched, connect to public library/connect to independent library

Disadvantages:

1. Responsibility of distributed system
2. The difficulty of multi-service operation and maintenance is increasing.
3. System deployment dependency, communication cost between services, consistent data, system integration testing, performance monitoring.

# 16. What is the difference between spring cloud and d bbo?

* Service calling method dubbo is RPC spri cloud Rest Api
* Registration center, dubbo is zookeep r springcloud is eureka, or it can be zookeeper
* Service gateway, dubbo itself is not implemented, can only be integrated through other third-party technologies, springcloud has Zuul routing gateway, which acts as a routing server to distribute consumer requests, springcloud supports circuit breakers, and integrates perfectly with git. Configuration file supports version A series of micro-service architecture elements such as control, transaction bus implementation, configuration update and automatic service assembly.

# 17. Comparison of REST and RPC

* The main defect of RPC is that the dependence between the service provider and the calling method is too strong. It is necessary to define the interface of each microservice and release it through continuous inheritance. Strict version control will not conflict.
* REST is a lightweight interface. There is no coupling between the provision of services and the invocation of the code, and only one convention is required for specification.

# 18. The microservice technology stack you know?

* Service development: springboot spring springmvc
* Service configuration and management: Archixius of NetFix, Diamond of Ali
* Service registration and discovery: Eureka, Zookeeper
* Service call: Rest RPC gRpc
* Service fuse: Hystrix
* Service load balancing: Ribbon Nginx
* Service interface call: Fegin
* Message Queue: Kafka Rabbitmq activemq
* Service Configuration Center Management: Spring CloudConﬁg
* Service routing (API gateway) Zuul
* Event message bus: SpringCloud Bus

**19. How do microservices communicate independently?**

* Remote calls, such as feign calls, directly access other services through remote procedure calls.
* Message middleware

# 20. How does springcloud register services?

* When the service is published, specify the corresponding service name and register the service to the registration center (eureka zookeeper)
* The registration center adds [@EnableEurekaServer](http://twitter.com/EnableEurekaServer), the service uses [@EnableDiscoveryClien](http://twitter.com/EnableDiscoveryClien)t, and then uses ribbon or feign to perform service direct call discovery.

# 21. The difference between Eureka and Zookeeper

* Eureka takes AP of CAP, pays attention to usability, Zookeeper takes CP of CAP to pay attention to consistency.
* Zookeeper registration service is paralyzed during the election. Although the service will eventually be restored, it is not available during the election.
* eureka’s self-protection mechanism will lead to the result that the service that expires due to not receiving a heartbeat for a long time will no longer be removed from the registration list. It can still accept registration and query requests for new services, but will not be synchronized to other nodes. No service paralysis.
* Zookeeper has the roles of Leader and Follower, and all nodes of Eureka are equal.
* Zookeeper uses the principle of more than half of the survival, and reka uses a self-protection mechanism to solve the partition.
* Eureka is essentially a project, Zookeeper is just a process.

# 22. What is eureka’s self-protection mechanism?

When Eureka Server loses too many instances of connection within a short period of time (such as network failure or frequent startup and shutdown of the client), the node will enter self-protection mode to protect registration information, no longer delete registration data, and automatically exit when the failure recovers Self-protection mode.

**23. What is Ribbon?**  
Ribbon is a load balancing client, which can control some behaviors of htt and tcp. Feign integrates ribbon by default.

**24. What is feigin? What are its advantages?**

* Feign uses interface-based annotations
* Feign integrates ribbon and has the ability to load balance
* Integrated Hystrix, with the ability to fuse

use:

* Add pom dependency.
* Add [@EnableFeignClients](http://twitter.com/EnableFeignClients) to the startup class
* Define an interface [@FeignClient](http://twitter.com/FeignClient)(name=”xxx”) to specify which service to call

# 25. The difference between Ribbon and Feign?

* + Ribbon calls other services, but in different ways.
  + The startup class annotations are different, Ribbon is [@RibbonClient](http://twitter.com/RibbonClient) feign is [@EnableFeignClients](http://twitter.com/EnableFeignClients)
  + The location specified by the service is different. Ribbon is declared on the [@RibbonClient](http://twitter.com/RibbonClient) annotation, and Feign is declared using [@FeignClient](http://twitter.com/FeignClient) in the interface that defines the abstract method.
  + The calling method is different. Ribbon needs to construct the http request by itself, simulate the http request and send it to other services using RestTemplate. The steps are quite tedious. Feign needs to define the called method as an abstract method.

**26. What is Spring Cloud Bus?**  
Spring cloud bus connects distributed nodes with a lightweight message broker. It can be used to broadcast configuration file changes or direct communication of services, and can also be used for monitoring.

If the configuration file is modified, a request is sent and all clients will re-read the configuration file.

use:

* + Add dependency
  + Configure rabbimq

**27. The role of springcloud circuit breaker?**  
When a service calls another service due to network or self-cause problems, the caller will wait for the caller’s response. When more service requests to these resources lead to more requests waiting, a chain effect (avalanche effect) occurs

The circuit breaker is fully open: it cannot be called for a certain number of times within a certain period of time and there is no sign of recovery after multiple monitoring.The circuit breaker is fully opened, so the next request will not request the service

Half-open: The circuit breaker will send some requests to the service within a short time, and the circuit breaker will be closed when it is normally called

Close: When the service has been in a normal state, it can be called normally

**28. Spring Cloud Gateway?**  
Spring Cloud Gateway is the second-generation gateway framework officially launched by Spring Cloud, replacing Zuul Gateway. As a traffic flow, the gateway plays a very important role in the microservice system. The common functions of the gateway include routing and forwarding, permission verification, and current limit control.

A RouteLocatorBuilder bean is used to create routes. In addition to creating routes, RouteLocatorBuilder allows you to add various predicates and ﬁlters. The meaning of predicates asserts, as the name implies, is processed by specific routes according to the rules of specific requests. ﬁlters are various filters The device is used to make various judgments and modifications to the request.

**29. As a business registration center, how is Eureka better than Zookeeper?**  
(1) Eureka guarantees availability and partition fault tolerance, and Zookeeper guarantees consistency and partition fault tolerance.

(2) Eureka also has a self-protection mechanism. If more than 85% of the nodes do not have a normal heartbeat within 15 minutes, then Eureka thinks that there is a network failure between the client and the registry. It will not paralyze the entire registration service like zookeeper.

**30. What is Ribbon load balancing?**

* + Spring Cloud Ribbon is a set of client load balancing tools based on Netﬂix Ribbon.
  + The Ribbon client component provides a series of perfect configuration items such as connection timeout, retry, etc. Simply put, it is to list all the machines behind Load Balancer (LB for short) in the configuration file. Ribbon will automatically help you connect these machines based on certain rules (such as simple polling, random connection, etc.). It is also easy to use Ribbon to implement a custom load balancing algorithm.

**31. What can Ribbon load balancing do?**

* 1. Distribute the user’s request equally to multiple services
  2. Centralized LB uses an independent LB facility (which can be hardware, such as F5, or software, such as nginx) between the service consumer and the provider, and the facility is responsible for passing the access request through a certain strategy Forwarded to the service provider;
  3. In-process LB integrates the LB logic to the consumer. The consumer learns from the service registration center which addresses are available, and then selects a suitable server from these addresses.

Note: Ribbon belongs to in-process LB, it is just a class library, integrated in the consumer process, the consumer obtains the address of the service provider.

**32. What is zuul routing gateway**

* 1. Zuul includes the two most important functions of request routing and filtering: which is responsible for forwarding external requests to specific microservice instances, which is the basis for achieving a unified access to external access and the filter function is the processing of negative requests Intervention in the process is the basis for implementing request verification, service aggregation and other functions,
  2. Zuul and Eureka are integrated to register Zuul itself as an application under Eureka service governance, and at the same time obtain news of other microservices from Eureka, that is, after accessing microservices, they are obtained after Zuul jumps.

Note: The Zuul service will eventually register in Eureka to provide three functions: proxy + routing + filtering

**33. What can a distributed configuration center do?**- Central management of configuration files, different configurations in different environments, dynamic configuration updates, and sub-environment deployment such as dev/test/prod/beta/release  
- Dynamically adjust the settings during operation, it is no longer necessary to write configuration files on the machine deployed by each service, and the service will pull the configuration information from the configuration center  
- When the configuration changes, the service can detect the configuration change without restarting and apply the new configuration to expose the configuration information in the form of a REST interface