**Questions:**

1. ***What are microservices? What are the benefits of this architectural style?***
   1. Microservices is an architectural design pattern that is becoming more prevalent as the benefits become more apparent with large scale applications. The “Microservice” style is an approach to developing an application as a suite of several small services, each running its own independent process and communicating with lightweight mechanisms such as an HTTP resource API.
   2. The benefits to utilizing this architectural style, is that it offers the possibility of no down time applications. As these processes or “microservices” all run independently, certain parts of the application can be updated/edited without having to shut the entire service down. Your development team can continue to work behind the scenes, without disrupting the end users experience. With a monolith application, any changes to the system involves building and deploying a new version of the server-side application. Another benefit of this abstraction, is the separation of concerns. In theory, one can have any number of microservices that communicate with the same API, but each reading a certain resource. My favorite benefit, is that this makes parts of the application language-agnostic as one component can be written in Python(maybe to perform some ML), and another can be written in Java.
2. ***What are the limitations of traditional configuration? What are the advantages of externalizing configuration settings?***
   1. While traditional configuration has served us well for many years, it does have many limitations. For instance, if you want to modify the configuration for a microservice that has been replicated a hundred times (one hundred processes are running). If the configuration for this microservice is packaged with the microservice itself, you’ll have to redeploy each of the one hundred instances. This can prove to be very time consuming, and can lead to unnecessary headaches as we must account for human error (we are not perfect after all).
   2. A service with externalized configuration works by keeping the configuration information in an external store, such as a database, file system, or environment variables. At startup, microservices load the configuration from the external store. During runtime, microservices provide an option to reload the configuration without having to restart the service. This not only automates the update of important information on several services, it also provides one central place to update it. Spring cloud config gives us this functionality. Less redundant work = Happy Dev team.
3. ***What is Spring Cloud? List and explain five features.*** 
   1. Spring Cloud is a framework used to build robust cloud applications. It aims to simplify the development of cloud applications via the following features :
      1. External configuration which allows you to store the configuration information in one file (env file, github etc) and consume that information across multiple microservices.
      2. Service registry is essentially a database of all services, their instances and their location. Service instances are registered with the service registry on startup, to enable easy identification.
      3. Client side load-balancing allows the client to decide where to send the traffic also using an algorithm like round-robin. It can either discover the instances, via service discovery, or can be configured with a predefined list. *Netflix Ribbon* is an example of a client-side load balancer.
      4. Circuit breakers operate similar to electric circuit breakers. Sometimes, when using several services, there is always a possibility that one may be unavailable or exhibiting such high latency that it is ultimately rendered unusable causing a failure. When the number of consecutive failures crosses a threshold, the circuit breaker trips, and for the duration of the timeout period all attempts to invoke the remote service will fail immediately. After the timeout expires the circuit breaker allows a limited number of test requests to pass through. If those requests succeed the circuit breaker resumes normal operation. Otherwise, if there is a failure the timeout period begins again.
      5. Leader election allows the application to work together with other applications to coordinate a cluster leadership via a third party system. A leader can then be used to provide global state or global ordering, generally without sacrificing availability.