老师早上好

我叫gerald，今天我演讲题目的是微服务架构。

什么是微服务架构呢？

微服务架构区别于传统的单体软件架构，是一种为了适应当前互联网后台服务的「三高需求：高并发、高性能、高可用」而产生的的软件架构。它把一个大型的单个应用程序和服务拆分为数个甚至数十个的支持微服务，可扩展单个组件而不是整个的应用程序堆栈，围绕业务领域组件来创建应用，这些应用可独立地进行开发、管理和迭代。在分散的组件中使用云架构和平台式部署、管理和服务功能，使产品交付变得更加简单。本质上，是用一些功能比较明确、业务比较精练的服务去解决更大、更实际的问题。

传统单体软件结构有着开发简单，集中式管理，基本没有重复开发，没有分布式成本等优点，但也有着维护难，不灵活，难扩展等缺点，那么与传统单体软件结构相比，微服务架构有什么优点呢？

这种架构解决了复杂性的问题，它将一种怪异的整体应用程序分解成一组服务。虽然功能总量不变，但应用程序已分解为可管理的块或服务。每个服务都以RPC或消息驱动的API的

形式定义了一个明确的边界。

这种架构使每个服务都能够由专注于该服务的团队独立开发。开发人员可以自由选择任何有用的技术，只要该服务符合API合同。当然，大多数组织都希望避免完全无政府状态并限制技术选择。此外，由于服务相对较小，因此使用当前技术重写旧服务变得可行。

看上去微服务架构解决了许多传统单体结构的问题，但实际上微服务架构也存在着各种各样的问题，比如多服务运维难度高，系统部署依赖，服务间通信成本。

研究学习了这种新式的软件架构，启发了我许多的思考，最重要的就是研究思维的转变，技术上不是问题，意识比工具更重要。对于我们学计算机的，软件的核心是业务逻辑，按照业务或客户需求组织资源，要理解技术是手段，而核心是设计。

Good morning, sir.  
My name is gerald and the topic of my talk today is microservice architecture.  
What is microservice architecture?  
Microservice architecture is different from the traditional monolithic software architecture, is a kind of software architecture in order to adapt to the current Internet backend service "three high demand: high concurrency, high performance, high availability" and the emergence of the software architecture. Essentially, it is to solve bigger and more practical problems with some services that have clearer functions and more refined business.  
The traditional monolithic software architecture has the advantages of simple development and centralised management, but also has the disadvantages of low efficiency, difficult maintenance, and inflexibility, so what are the advantages of microservice architecture compared to the traditional monolithic software architecture?  
This architecture solves the problem of complexity by decomposing a monstrous overall application into a set of services. While the total amount of functionality remains the same, the application has been broken down into manageable chunks or services. Each service, in the form of RPC or message-driven API's  
form that defines a clear boundary.  
This architecture allows each service to be developed independently by a team focused on that service. Developers are free to choose whatever technology is useful, as long as the service conforms to the API contract. Of course, most organisations want to avoid complete anarchy and limit technology choices. In addition, because the service is relatively small, it becomes feasible to rewrite old services using current technologies.  
It may seem that microservice architecture solves many of the problems of traditional monolithic structures, but in reality microservice architectures suffer from a variety of problems, such as difficulty of multi-service operations and maintenance, system deployment dependencies, cost of inter-service communication, and high complexity.  
Research and study of this new style of software architecture, inspired me to think a lot, the most important thing is to study the change of thinking, technology is not a problem, awareness is more important than tools. For us to learn the computer, the core of the software is business logic, according to business or customer demand organisation of resources, to understand the technology is the means, and the core is the design.