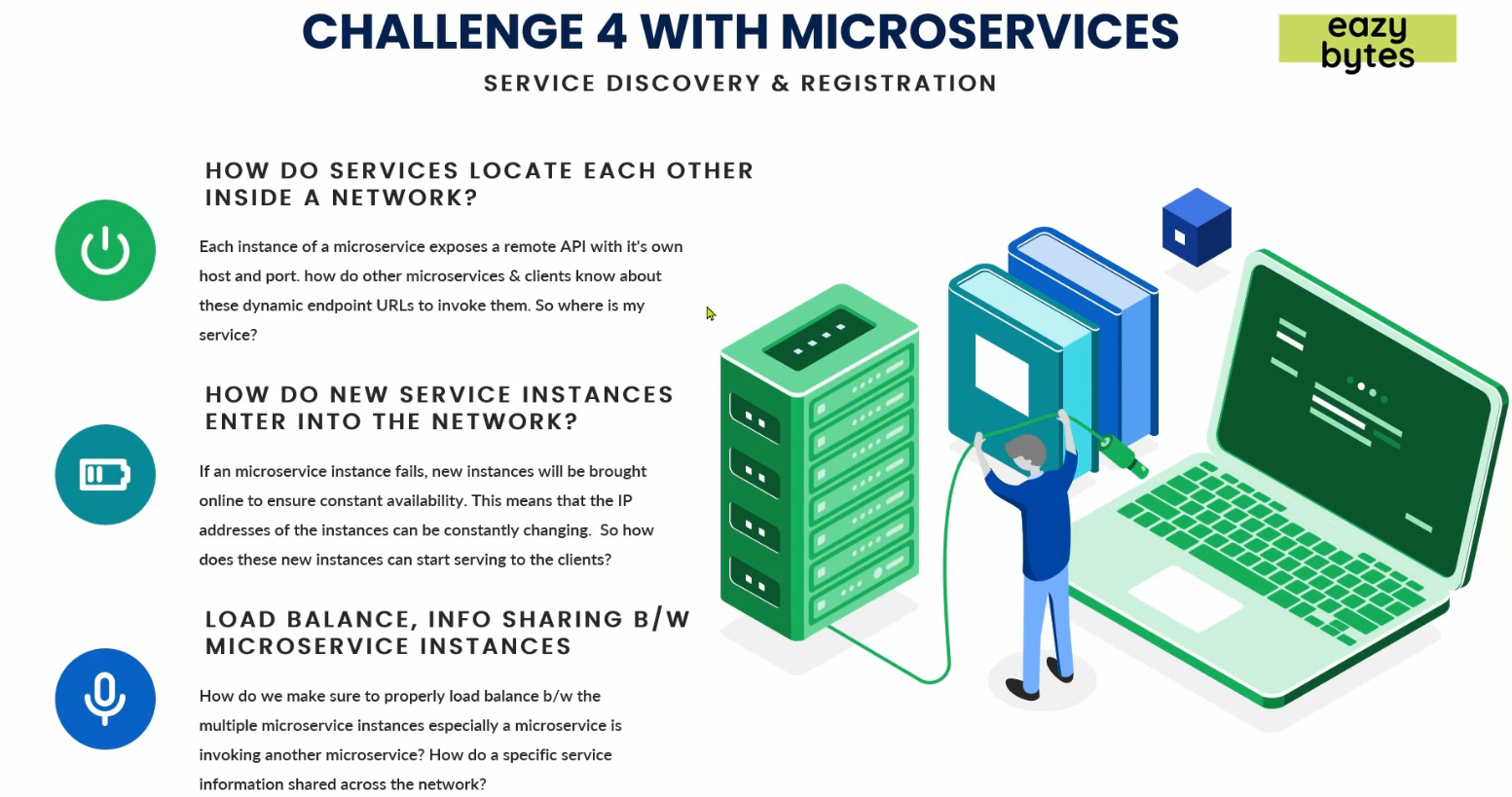
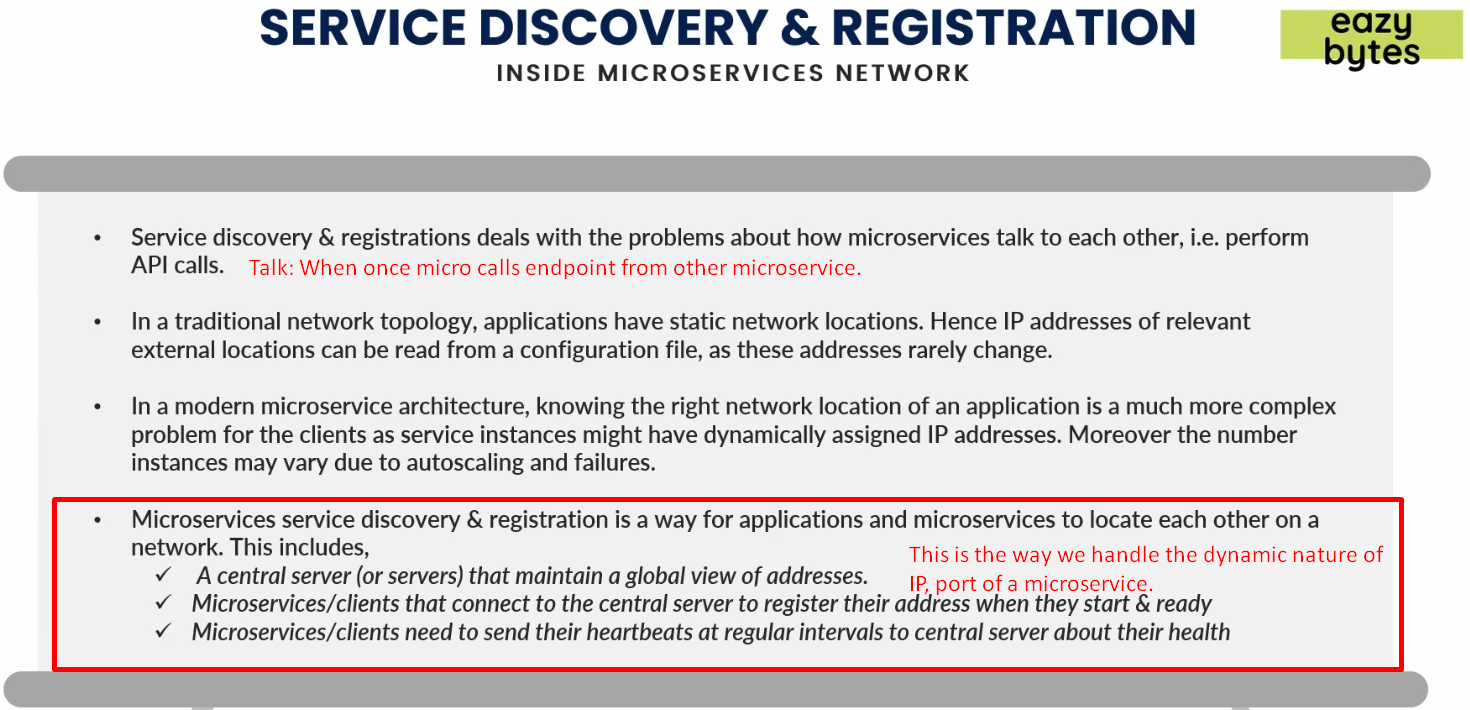
1. We should never have this impression that why we should handle those challenges. We can simply go for monolithic app. The reason is that the advantages we’re going to get with microservices are more business friendly compared to the challenges we may face.   
   Challenges are not the disadvantages/problems of microservices. So don’t think that we should go for monolithic app.
2.   
   Before jumping onto this 4th challenge, let me ask you a few questions.
3. **First Question**: We have a network inside our app. Inside your network, you have 100 microservices and they have their own instances and each microservice instance will have its own IP address, PORT# 🡺 Endpoint.  
   **How one MS finds other microservice in the network**.
   1. Suppose you have 100 microservices and each has 10 instances.
   2. Each microservice has its own IP, port, end-point URL.
   3. Question is how each microservice locates each other inside a network.
   4. In microservice architecture, we scale up or down microservice as per our need so **IP and Port are dynamic** in nature which is not in monolithic where we can put this info in some property file.
   5. Suppose, we have three microservices cards, loans, Cards then how one locates other microservice’s endpoint.
4. **Second Question**: How a microservice **registers** itself (IP details, Port 🡺 Endpoint)?
   1. A microservice is scaled up and how does this new instance register its IP, port and endpoint so that other microservices can consume it.
5. **Third Question**: **Load** **balancing**.
   1. How to balance load b/w different instances of a microservice.
   2. How to share info from one micro to another micro.
      1. I think “info sharing” when feign client fetches the registry info about other microservices.
6. **Solution**:
   1. This all will be solved with a **pattern** called **“Service Discovery & Registration Pattern”**.



1. Solutions by Service Discovery & Registration Pattern:
   1. It will help us to deal with the problem about how MSs talk with each other like if some MSs want to make API calls b/w them.   
      Basically, the instructor is trying to say that how one MS locates another Ms.  
      Whereas in traditional network topology where we have monolithic apps or SOA app, we have static IP addresses (network locations) which can be read from a configuration file and these addresses rarely change and if they change, the Infra team can easily handle by making those changes in the configuration file and restarting your apps but in case of Microservice Topology, it is not possible as we have hundreds of microservices and their Network locations are dynamic.

|  |  |
| --- | --- |
| **Traditional Topology** | **Microservice Topology** |
| Static Network Locations (IP Addresses) | Dynamic Network Locations |
| **Solution**: Configuration File | **Solution**: Service Discovery & Registration. |

1. In “Service Discovery & Registration”, we follow 3 steps.
   1. A Central Server(s) or you can call a central microservice which maintain the global view of addresses:
      1. Suppose we have 100 microservices. Central Server will handle the IPs, ports 🡺 endpoints of all those microservices in a single place.
      2. To maintain this central global view, make sure that each new instance of a microservice when gets started, needs to get itself registered with this Central Microservice.  
         The new instance will register its IP, port and all other info with Central Server.
   2. After a specific interval of time, each instance sends a heartbeat to this central server for its ok health.  
      So, if central server doesn’t receive heartbeat (maybe MS instance is slow or it has its own network issue) from a particular instance for a particular timeframe, the central server will remove all the registered info for that instance so that no other microservice can invoke any endpoint from that instance otherwise other MS may face timeout issue or may receive improper result.
2. So, in this way, “Service Discovery and Registration” helps us to maintain the **network topology** inside microservice architecture by addressing all the above 3 issues in our microservice.