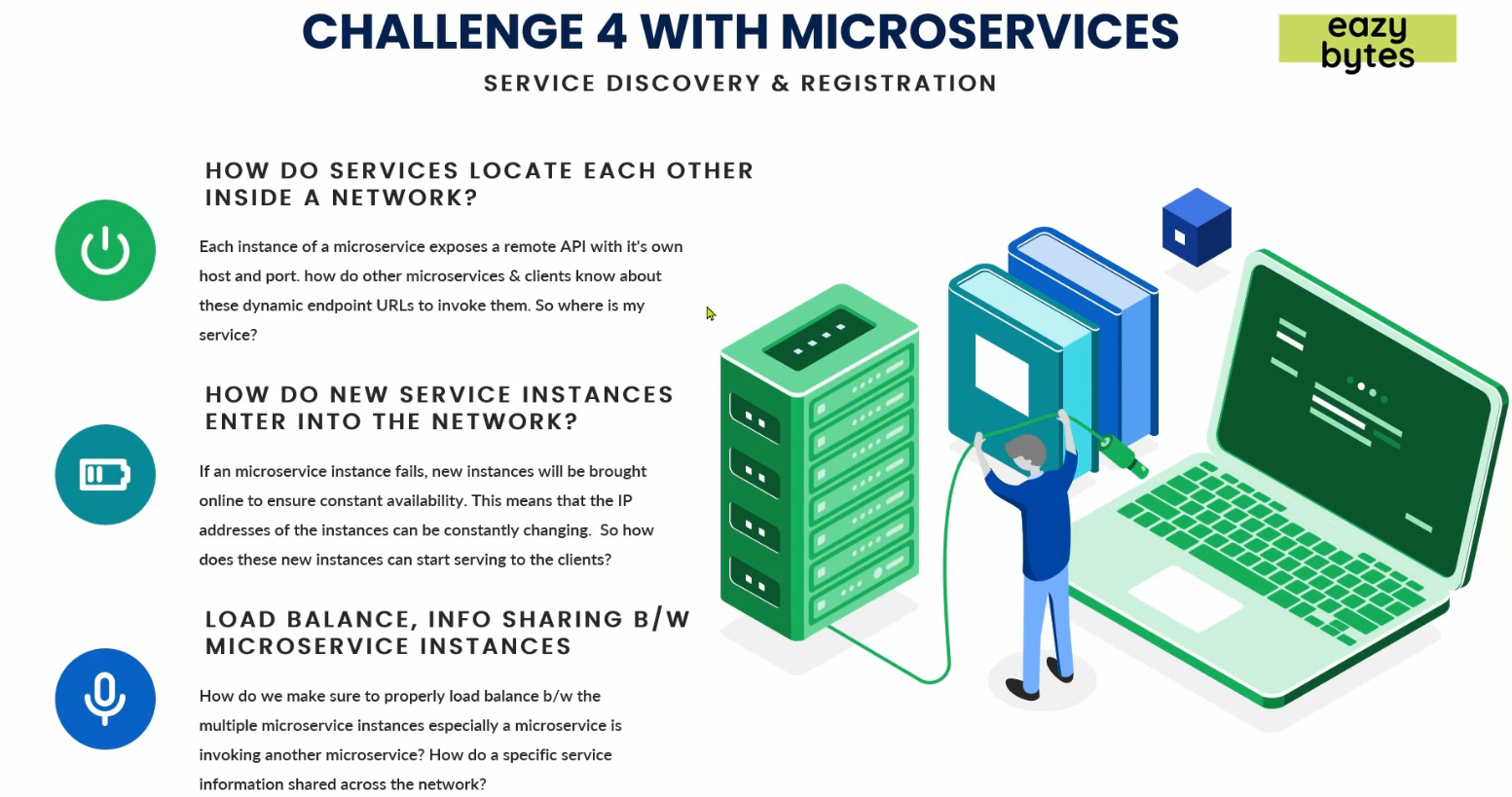
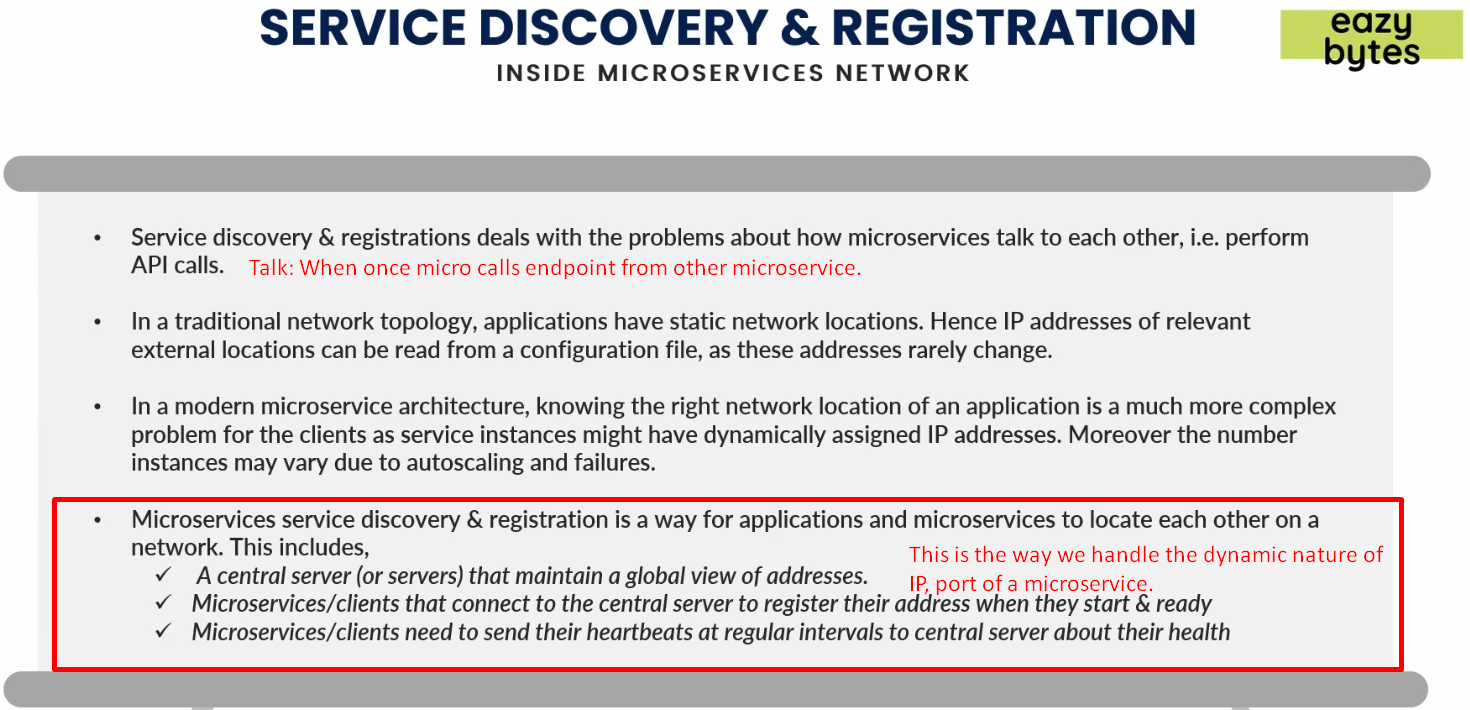
1. We should never have this impression that why we should handle those challenges. We can simply go for monolithic app. The reason is that the advantages we’re going to get with microservices are more business friendly compared to the challenges we may face.   
   Challenges are not the disadvantages/problems of microservices. So don’t think that we should go for monolithic app.
2. 
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# 🡺 End-point.  
   How to find 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 two microservices cards, loans 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.
6. **Solution**:
   1. This all will be solved with a **pattern** called **“Service Discovery & Registration Pattern”**.



1. In “Service Discovery & Registration”, we follow 3 steps.
   1. A Central Server(s):
      1. Suppose we have 100 microservices. It will handle the IPs, ports and endpoints of all those microservices.
   2. Each new instance of a microservice needs to get itself registered with this Central Microservice.  
      The new instance will register its IP, port and all other info with Central Server.
   3. 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 from a particular instance, it will remove all the registered info for that instance so that no other microservice can invoke any endpoint from that instance.
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.