1. **Agenda**:
   1. How Circuit -Breaker pattern helps us to achieve resiliency in microservice architecture/network.
2. Let’s me give you a scenario, where this is going to help us
3. In any distributed environment, there may be multiple calls happing to remote resources.
4. These services can fail due to slow network connections or timeout or resources are overcommitted or temporarily unavailable.
5. Any cloud or microservice based apps should be able to automatically handle such scenarios.
6. As we saw if cards microservice doesn’t work properly, the dependent microservice like loans and accounts as well don’t work for some API.
7. How Circuit-Breaker pattern is going to help us.
8. As the name indicates this pattern inspired from the **electrical circuit breaker.**   
   If too much electrical current is passed through a circuit, the circuit breaks and it will open so that no further current will be passed to the circuit which will save us from the further damage that could happen.
9. Similarly, in Software, Circuit-Breaker pattern, it will monitor each and every call that is going to happen through the circuit breaker.  
   NOTE: Actually, We need to declare which end-points are to be monitored by Circuit-Breaker.
   1. If a call **takes too long**, the very first thing, the circuit breaker will come into picture and **kills that call.**
   2. If it is happening for majority of the requests coming, it will decide to open the circuit so that no further invocations happen.  
      So, if circuit breaker opens, the accounts and loans microservices don’t have to wait till the timeout happens.  
      They immediately get a response because the network is open and at the same time, they can get some default response that you have configured instead of getting a so and so microservice is not working properly.  
      The Circuit Breaker is smart enough that it knows when to open the circuit. Say, if 50% of the requests get failed, it will open the circuit (failing fast) and based on the configuration for accounts and loans microservices’ endpoints, it returns exceptional response or default response. This way, accounts and loans don’t have to wait which will save from the throttling scenario on those Microservice issues.
   3. Start from 13:31