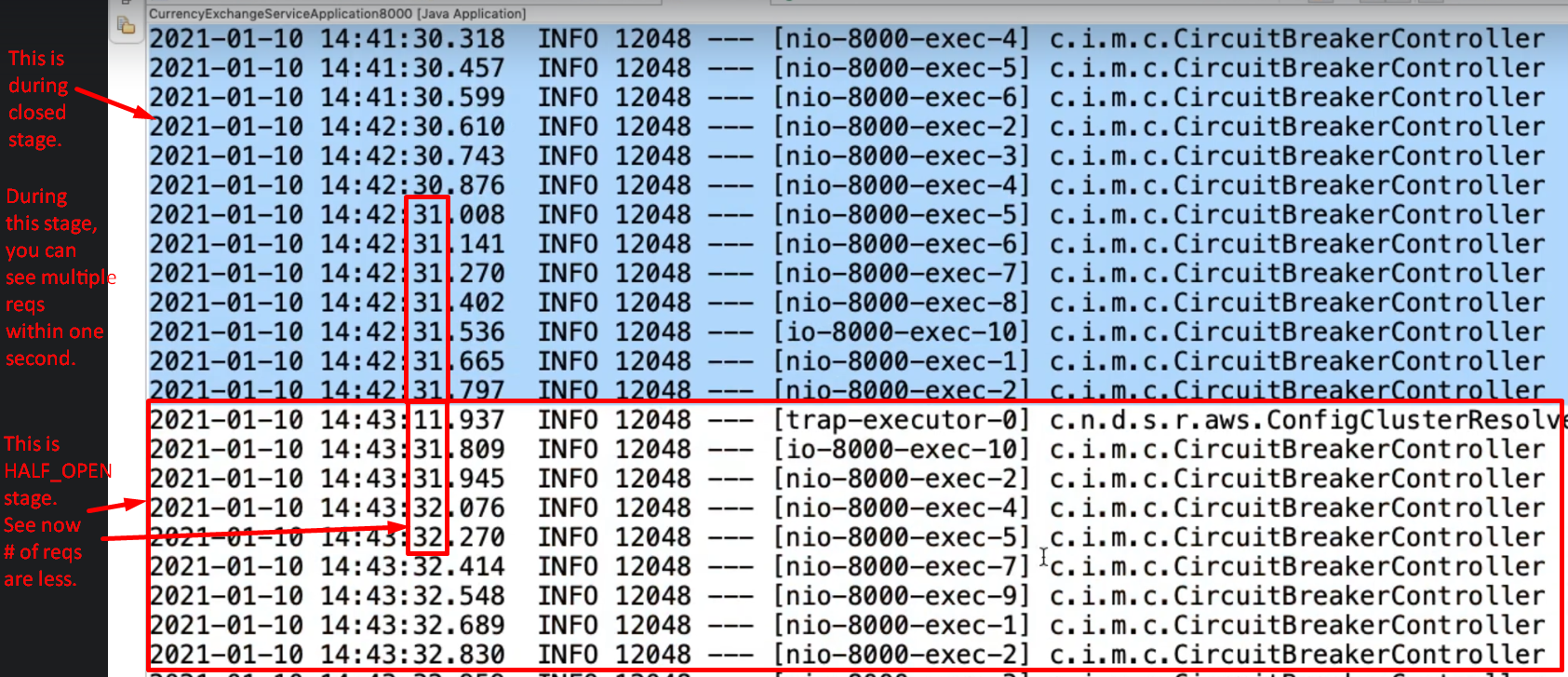
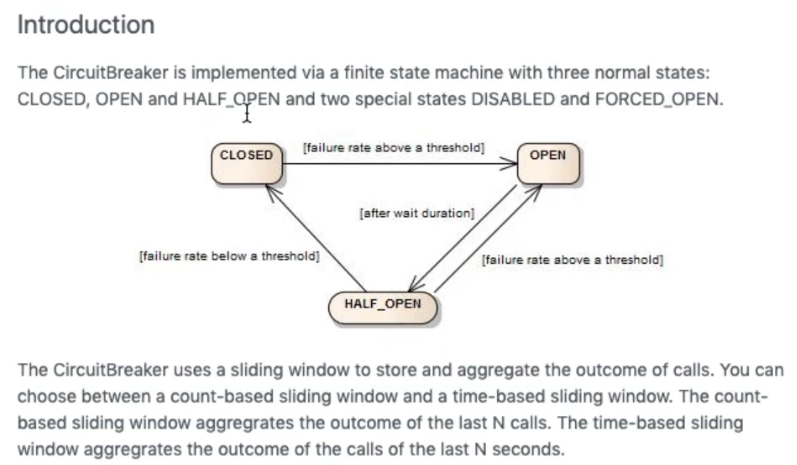
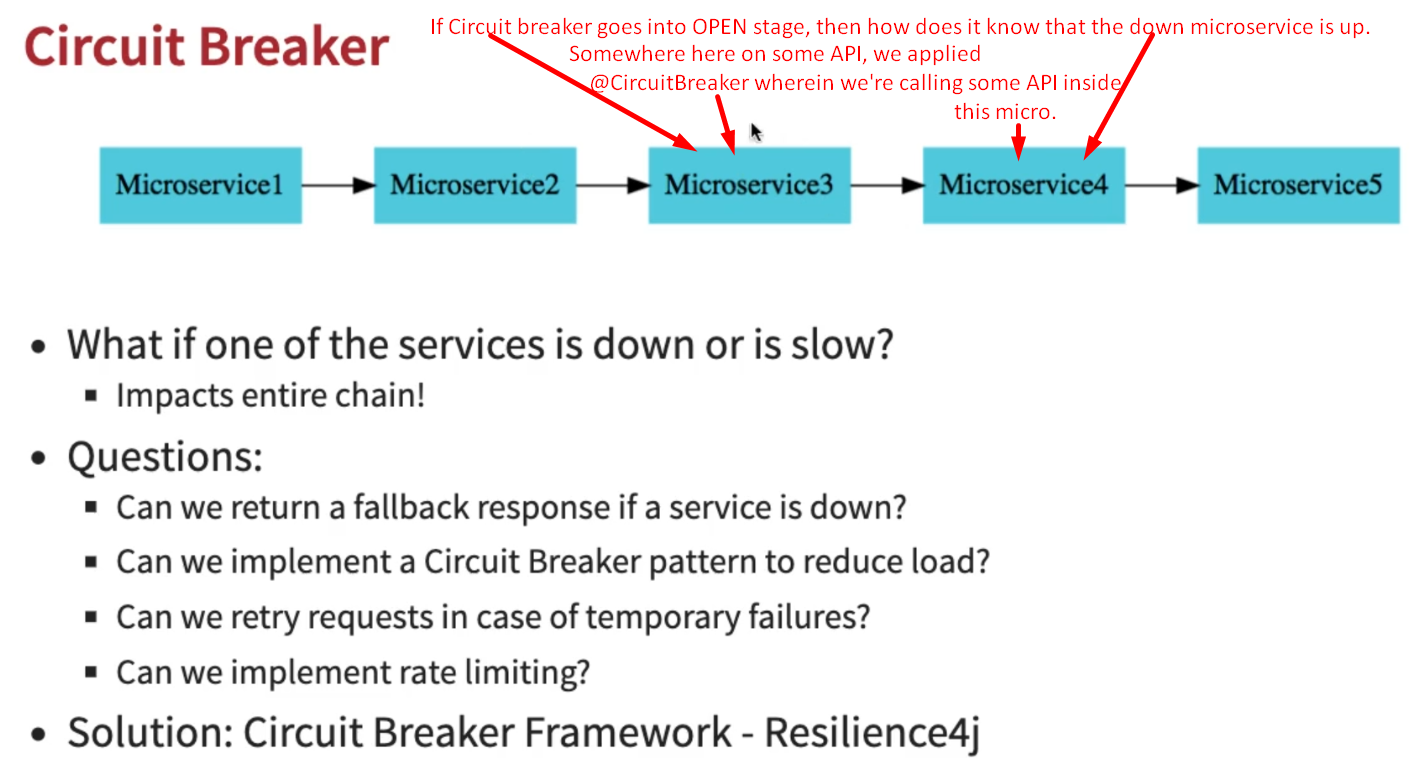
1. **NOTE**:
   1. We will hit an endpoint a number of times simultaneously to see Circuit breaker in action.
   2. Using Windows browser, you can do this by hitting “refresh” button.
   3. On mac, using “watch” command.
2. 
3. **Stages for Circuit Breaker**: Suppose “A” microservice has endpoint ‘a’ which is annotated with @CircuitBreaker and inside this endpoint, we’re calling some endpoint ‘b’ of microservice “B”. So “B” is dependent microservice. ☺
   1. **CLOSED**:
      1. When a request comes to ‘a’, ‘a’ always calls ‘b’ endpoint of microservice ‘B.
      2. When we launch an app, it is in CLOSED stage.
   2. **OPEN**:
      1. Circuit breaker will not call the dependent microservice ‘b’ rather returns the fallback response.
   3. **HALF\_OPEN**:
      1. Circuit breaker will be sending percentage of the requests to the dependent microservice and for the rest, it will send fallback response.

**NOTE**: We’re calling ‘b’ endpoint from ‘a’ endpoint in ‘A’ microservice and ‘b’ endpoint is a dummy endpoint so all our requests would fail.

Suppose the circuit breaker is in CLOSE stage initially.   
Suppose our 90% or more of the requests get failed. So circuit breaker would go into OPEN stage.  
After a time period (which we can configure), the circuit breaker will go into “HALF\_OPEN” stage.  
During the “HALF\_OPEN” stage, it sends the percentage (can be configured) of the next incoming requests to the dependent service.   
If it gets proper response for those outgoing requests, then it will go into the CLOSED stage and for the remaining percentage, it will return fallback response. 



1. 