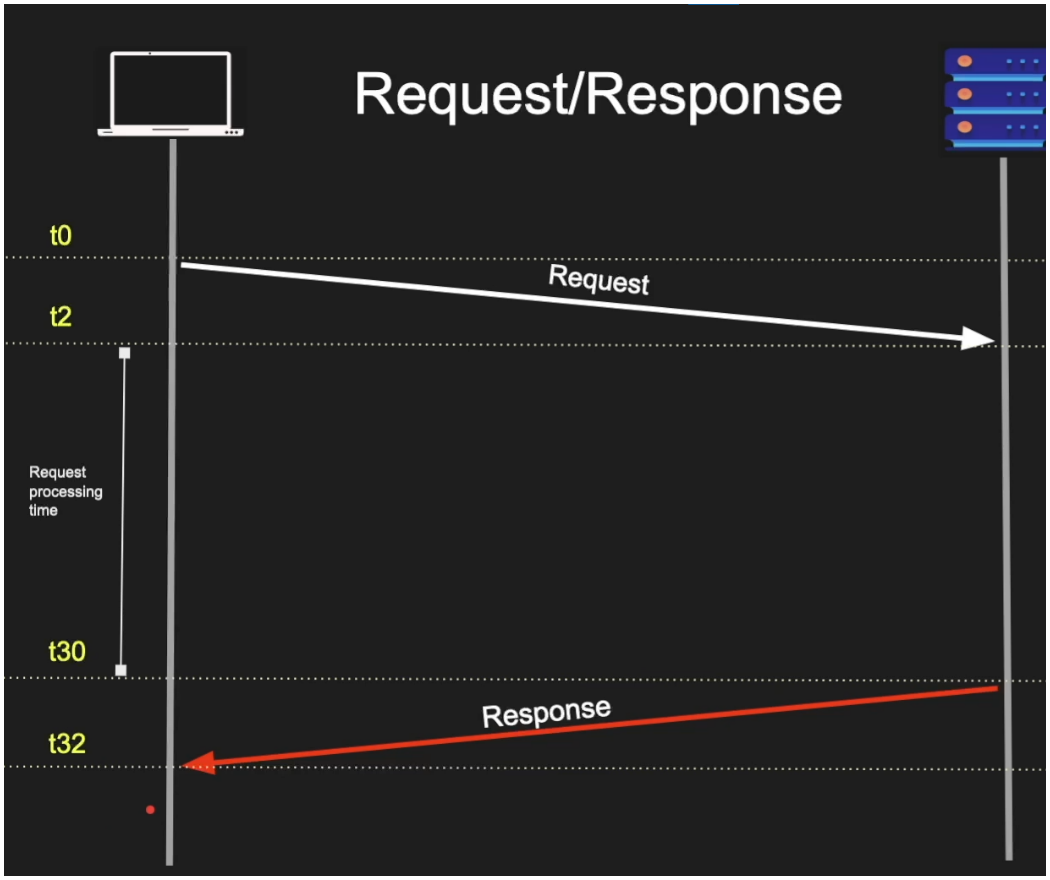
BackEnd Fundamentals

1. **Backend Communication**
    - **Request Response Model**
            - Client sends a request
            - Server parses the request
            - Server processes the request
            - Server sends a response
            - Client parses the response and consumes it

        - **Where is it used?**
            - Web, http, dns, ssh
            - RPC
                - You send a request to execute a method then give you back the response
            - SQL and database protocols
                - You send the query and get back the response
            - APIs (REST, SOAP, GraphQL)

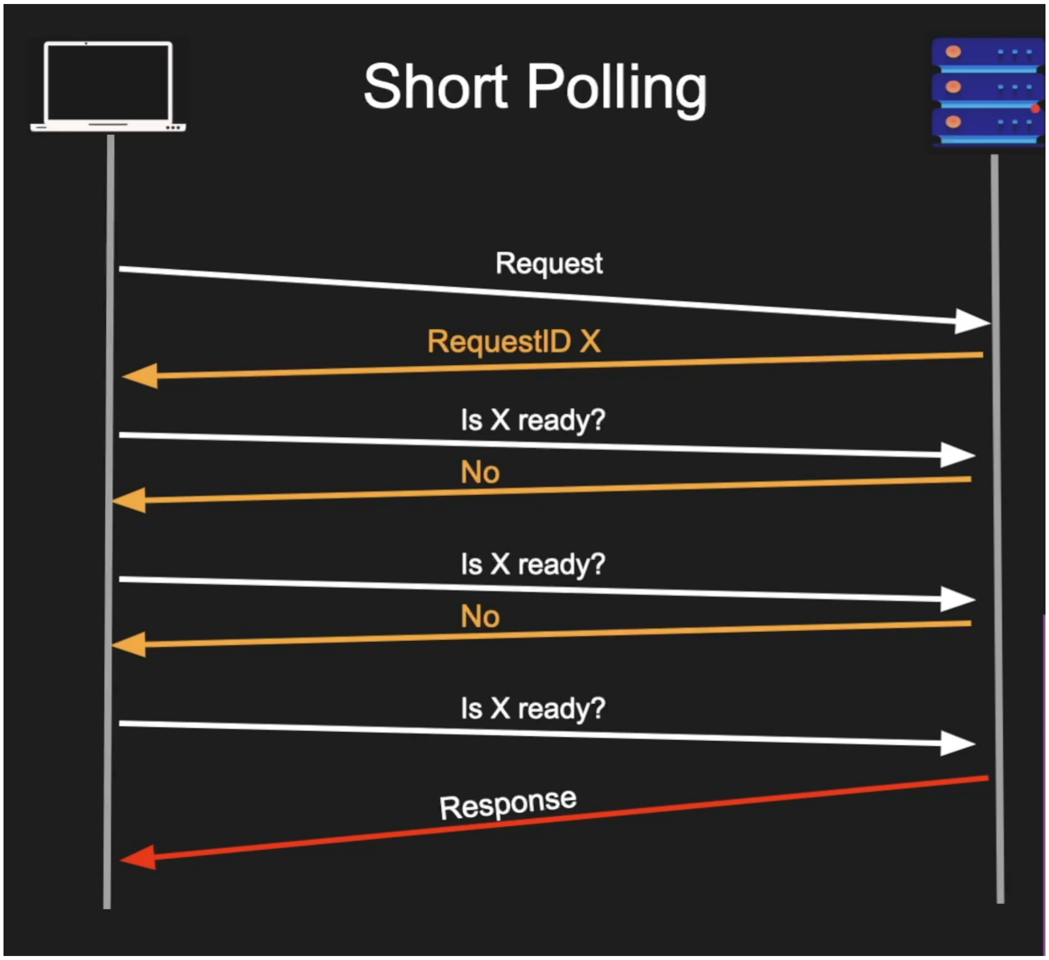
        - Anatomy of a Request / Response
            - C++ uses a JSON parser to understand its content, which takes much time
                - C++ uses a JSON parser to understand its content, which takes much time

        - Doesn't work everywhere
            - Because in this case, the server needs to send a response without a request
                - Because in this case, the server needs to send a response without a request
            - Very long requests
                - It's better to use another pattern or make it async if possible
            - Bad For multiple receiver
            - Hight coupling

- 

- Short Polling
  - Request is taking a while, i will check with you later.

  - What is short polling ?
    - client sends a request
    - server responds immediately with a handle
    - server queue the request or process it
    - client uses the handle to check for status
    - multiple short request response as polls

- 

  Short Polling

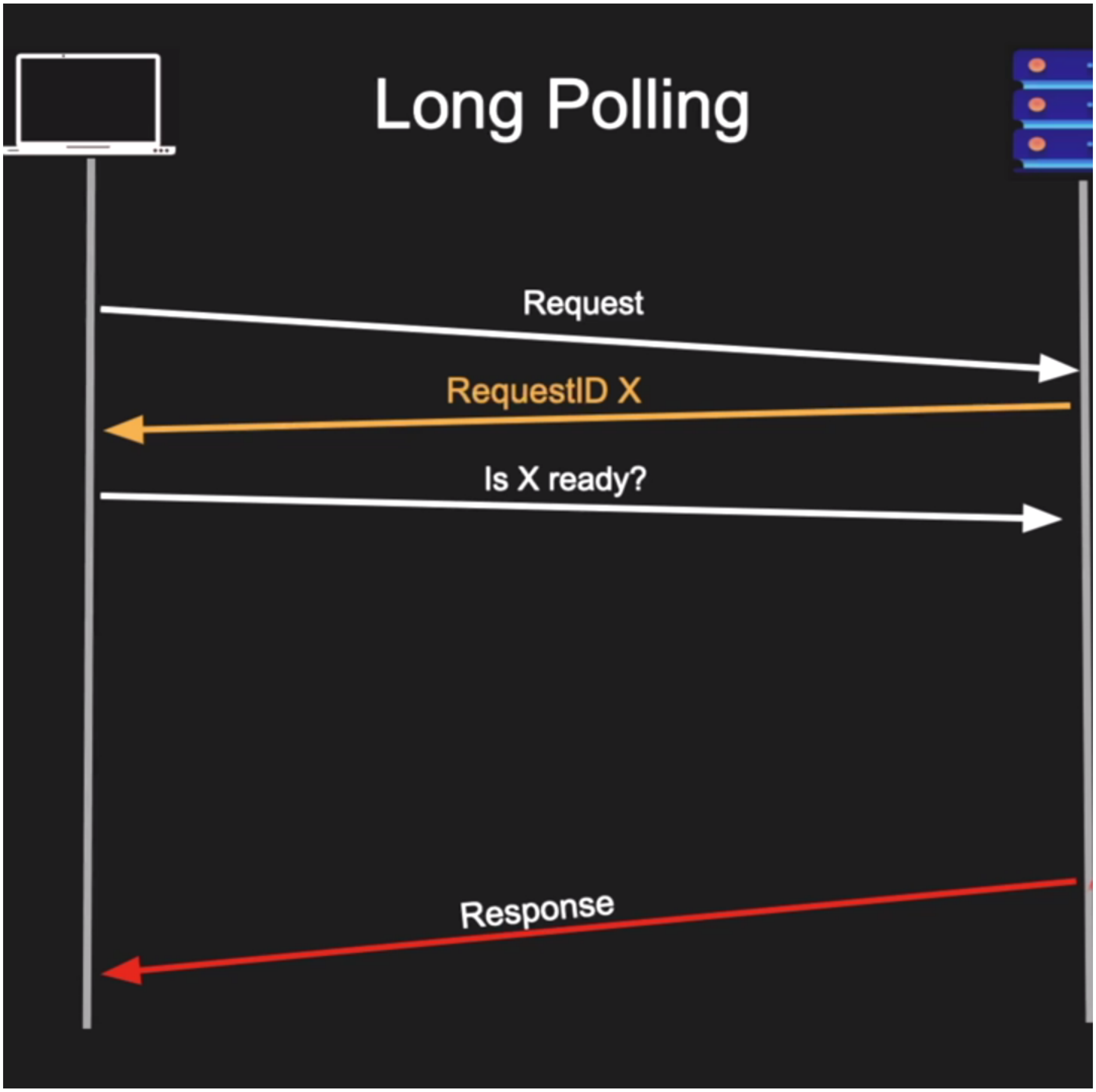
- Pros and Cons
  - Pros
    - Simple
    - Good for long running requests
    - Client can disconnect and back later
  - Cons
    - Too chatty
      - lot of requests hard to handle in large scale.
    - Network bandwidth
    - Wasted backend resources
- Where request/response & polling isn't ideal.
  - A request takes long time to process
    - Upload Youtube video
  - the backend wants to send notifications

- Long Polling

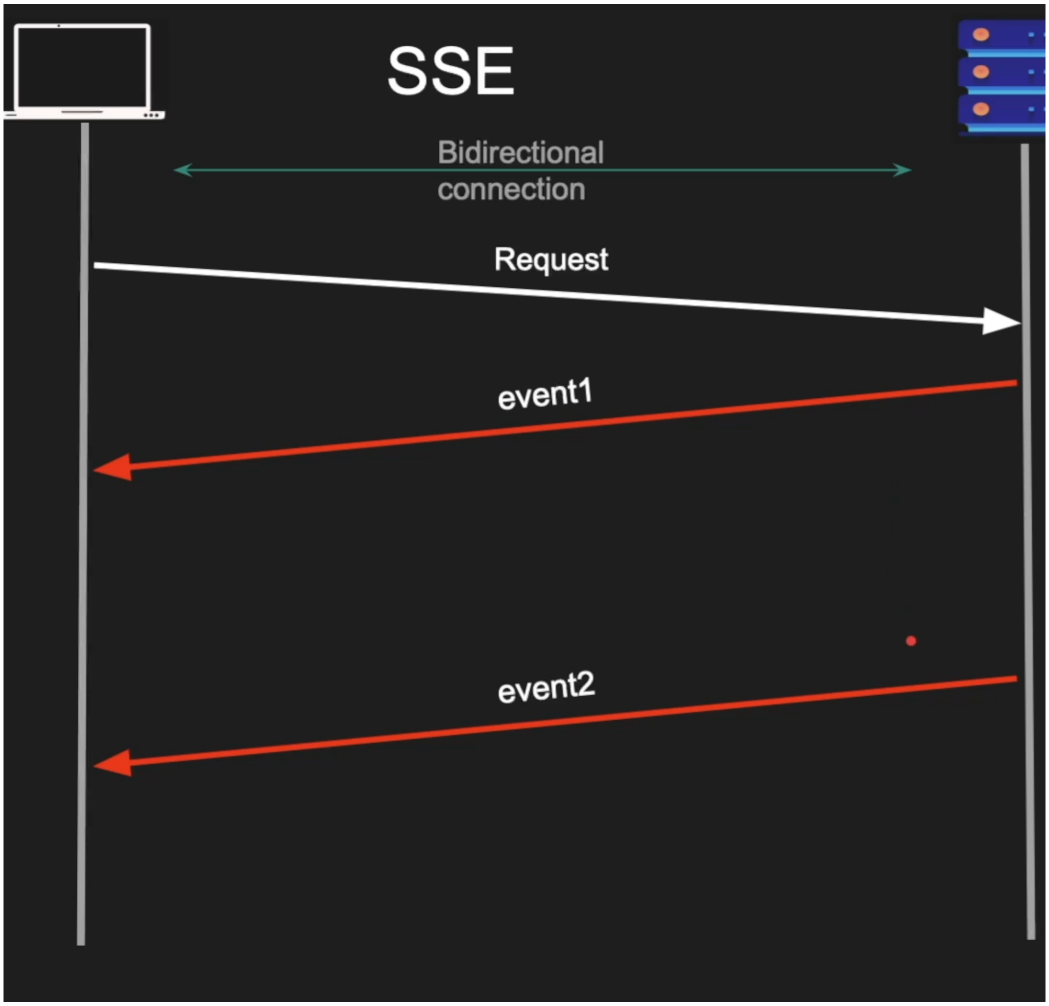
- Request is taking long, i will check you later but talk to me only when its ready.
- Used with Kafka

- What is long Polling?
  - Client sends a request
  - server responds immediately with a handle
  - server continues to process the response
  - client uses that handle to check for status
  - server does not reply until it has the response

- So we got a handle, we can disconnect and we are less chatty
- some variation has timeouts too.
- But its no a real time pattern.



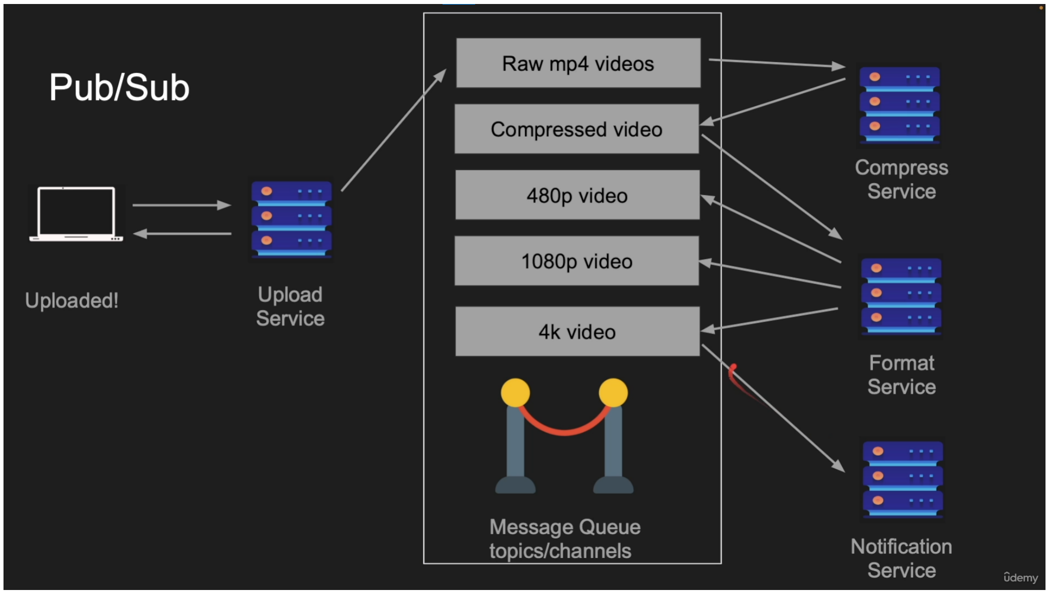
- Server Sent Event
  - What is server sent event ?
    - A response has start and end
    - client sends a request
    - server send response as part of events

- 

- Pros and Cons
  - Pros
    - Real time
    - Compatible with Request/Response
  - Cons
    - Clients must be online
    - Clients might not be able to handle
    - Polling is preferred for light Clients
    - HTTP/1.1 only 6 connections are allowed

- Publish Subscribe

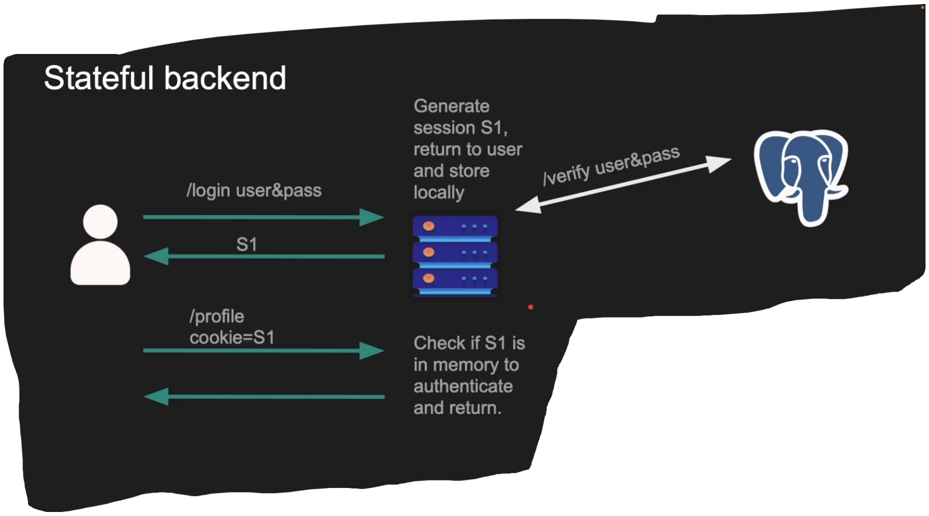
  - the client publish the request to do something
  - immediately get the response tell him that the job is processing

  - the job is processed in background.
  - Used With RabbitMQ or Kafka
  - 

    Pub/Sub

  - Pros and Cons
    - Pros
      - Scales with multiple receivers
      - Great with microservices
      - Loose coupling
      - Work while clients not running
    - Cons
      - Complexity
      - Network Saturation

- Statefull vs stateless
  - Backend
    - statefull
      - Depending on the server to store the state of the client and use this state in client processes
      - ex: storing client JWT in the server
      - and each client request you check if the JWT of the client is valid
      - in the middle if the server restart then the client have to login again

    - 

      Stateful backend

      /login user&pass

      S1

      Generate session S1, return to user and store locally

      /verify user&pass

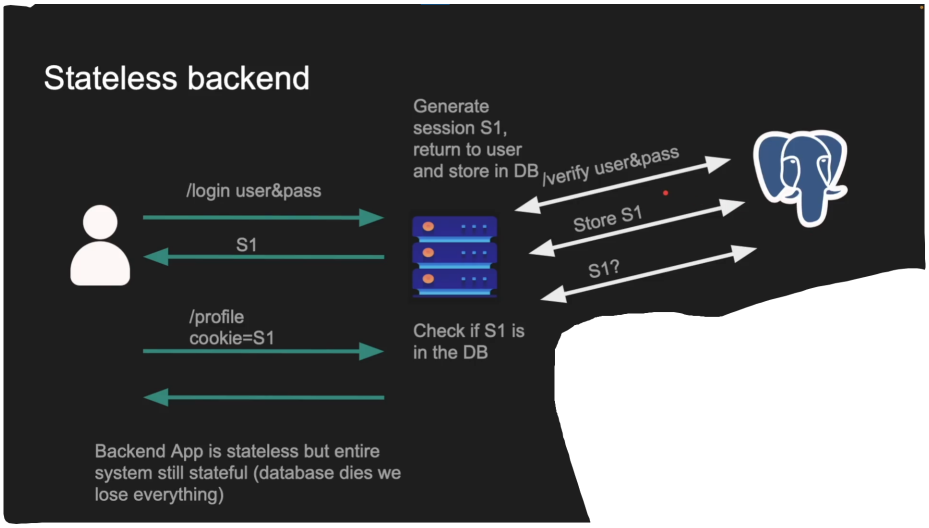
      /profile cookie=S1

      Check if S1 is in memory to authenticate and return.

    - Here, the backend is depending on itself to handle user auth.

- server doesn't store the state of the client in memory or in somewhere which if
    - server doesn't store the state of the client in memory or in somewhere which if

      it crash he lost it.

    - 

      Stateless backend

      /login user&pass

      S1

      Generate session S1, return to user and store in DB

      /verify user&pass

      Store S1

      S1?

      /profile cookie=S1

      Check if S1 is in the DB

      Backend App is stateless but entire system still stateful (database dies we lose everything)

    - Here, Server store the session in the database.

---

1. Protocols

- What is a protocol ?
  - System allows to parties to communicate
  - set of properties depending on the purpose of the protocol
- IP
  - if 192.168.1.4 needs to talk to 192.168.1.5 they doesn't need router because they in the same subnet. here they in layer 2
  - but if 192.168.1.4 needs to talk to 192.168.2.5 they need a router because they in the different subnets. here they in layer 3
  - you have to notice that if the database and application in different subnets then they need a router to talk and router makes an additional delay, so it's preferred to add the database server and applications server to the same subnet if it possible.
  - second thing to notice is in different subnets the request first goes to the default getway which the router IP in this case then the router route it to the destination, but in this scenario the breakpoint is may some device pretend that he is the default getway instead of the router and all requests go through it.