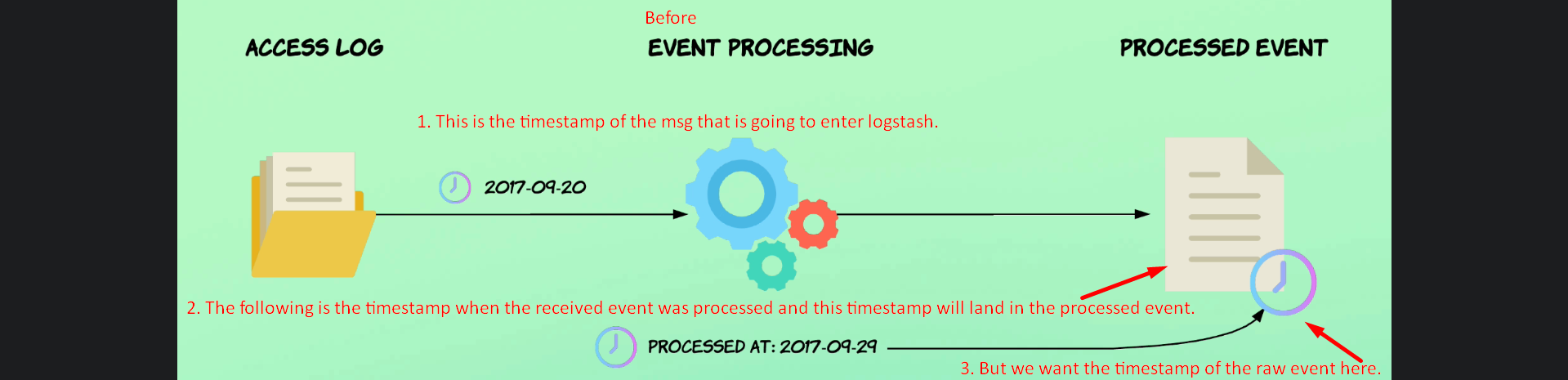
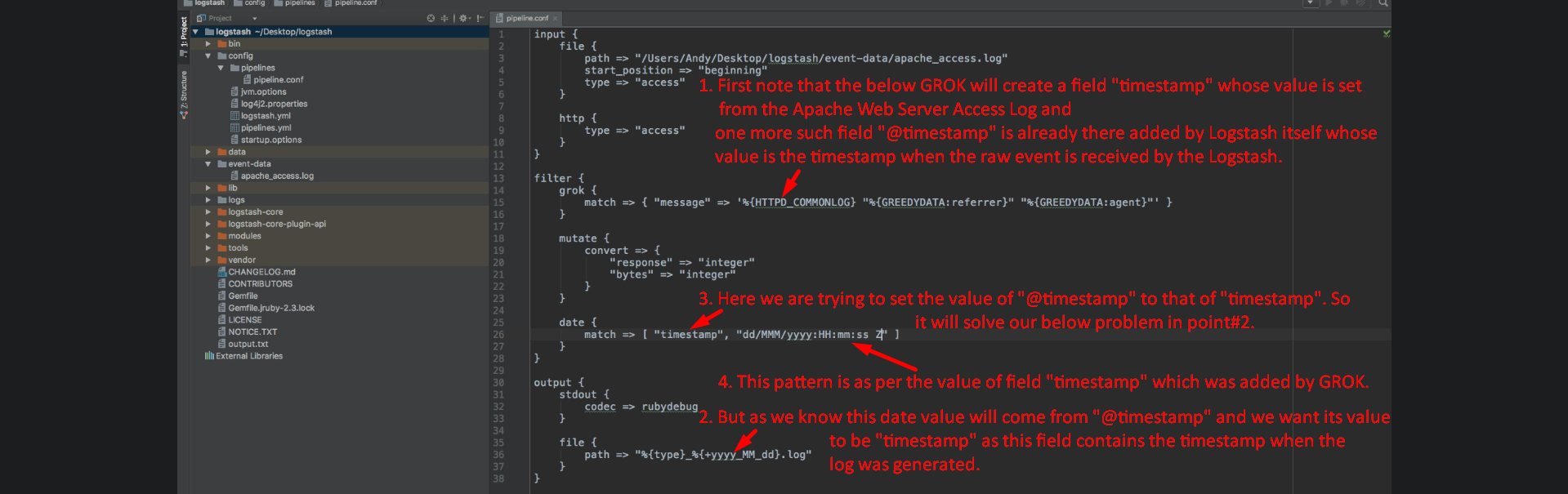
1. 
2. But we have **@timestamp** field (set by Logstash itself) for each event, then why do we need to set the time of the event?
3. Actually **@timestamp** contains the timestamp of when the Logstash receives the event.  
   More specifically, when the input plugin receives the event.
4. But in some scenarios, we need to be able to use a different timestamp.  
   Suppose, we have a web shop and for whatever reason, we process the order daily at midnight.  
   When Logstash receives an event at midnight, it sets the @timestamp field to the current time but we actually want it to be the time of when the order was placed and not when Logstash receives the events or take our current pipeline as an example.  
   Image that we want to process our access log for the past week maybe due to the change in pipeline configuration a week ago caused an issue or perhaps we suffered a loss of data. Whatever be the case, now we need to process the access logs for the past week once again.  
   If we run the logs through the Logstash as usual, we will end up, we will end up with a bunch of events where the timestamp equals the time of processing the events.  
     
     
   The following is what we want.  
   A picture containing text, screenshot, diagram, font

   Description automatically generated  
   This will ensure that the request will contain the same timestamp regardless of when Logstash processes it.  
   This is useful when backfilling data for whatever reason or there is notable delay in the process of access log.
5. In our pipeline configuration, we are using even timestamp as part of the file name that the events are written to.  
   As of now, it is current date (when the event was processed).  
   Let’s change it to be the date the request was received by the Apache Web Server.
6. We will use filter **plugin named date**.  
   This filter passes a date or date time value from a field and uses the results as the event timestamp (meaning setting that date-time to **@timestamp**).
7. Before proceeding further, note that the below field “timestamp” was added by GROK. To know more read the next snapshot.  
   "timestamp" => "20/Sep/2017:13:22:22 +0200"  
   In the below snapshot we will use 🡺 match => [ "timestamp", "dd/MMM/yyyy:HH:mm:ss Z" ]  
   as per the above value. Read the next slide for more understanding.  
     
     
   Let’s see the before and after effects after adding date filter plugin.  
   **Before:**A screenshot of a computer

   Description automatically generated with medium confidence  
   **After**:  
   A screenshot of a computer

   Description automatically generated  
   Great!!! See now @timestamp contains the date time of when the log request was generated instead of date and time when log request was processed by Logstash. That is because now @timestamp contains the timestamp parsed with GROK using the access log requests.  
   So, now with @timestamp properly set, it allows us to process the old access logs (in case we need) the exactly same way as they would have been processed at the time of the request.  
   This way we don’t have to worry about
   1. When we process the logs and any consequences of delayed processing or us needing to reprocess events.
8. If parsing of date fails, the filter date plugin will add a tag named **\_dateparsefailure** to the tags field. Like  
   
9. We will talk about how to handle failure in json parse and date parse (as you can see above).
10. Now we don’t need timestamp field anymore as @timestamp and timestamp fields both have same value. Let’s remove “timestamp” added by GROK.  
    A screen shot of a computer

    Description automatically generated with medium confidence