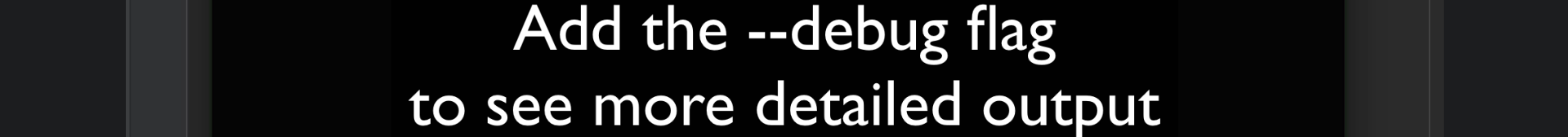
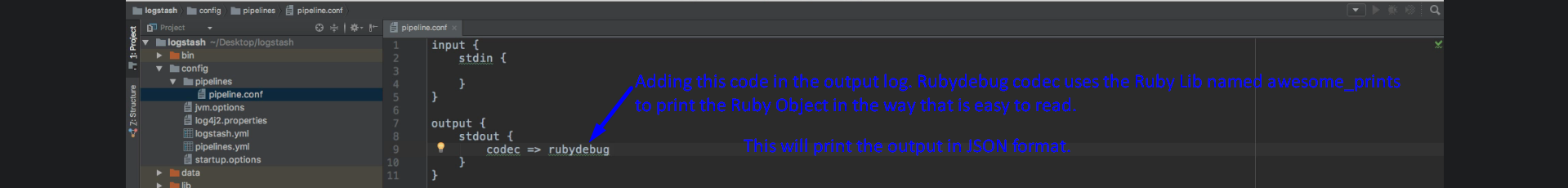
1. 
2. Let’s process our first event.
3. For that, we need to configure a pipeline.   
   We know that a pipeline consists of three 3 parts.
   1. **Input**:
   2. **Filter (optional)**:
   3. **Output**:  
      We will start out simple and only use Input and Output for now.

1. **How to define/configure a pipeline**?
2. By writing a **configuration file** consisting of the three parts (Input, Filter(optional), Output).
3. Logstash uses its **proprietary syntax** for its **configuration file**.   
   We will come to it soon.
4. Before writing a **configuration file**, I want to show you how to define a **pipeline configuration as an argument** when starting up **Logstash**.
   1. To start simple, we will define a pipeline that listens for events on STDIN and outputs the events on STDOUT.
   2. With that said, we will enter some text in the terminal and see the event data being output.
5. Let’s define the pipeline configuration via the CMD.
6. bin/logstash -e ”input **{** stdin{} **}** output { stdout{} }”
   1. **bin/logstash**: Binary
   2. -e: To pass pipeline configuration directly as an argument to the Logstash Library.
   3. “**input** **{** stdin {} **}**”:
      1. **input**: Input section
      2. input **{ }** : Object but in logstash terminology, it is referred to as input **hash object**.
         1. Within this bracket, we will define the source of input like STDIN, HTTP, File etc.   
            Basically, here we define how we want to listen for events.
         2. In our case we want to listen for events on **STDIN**.
         3. **STDIN** in this case is referred to as **INPUT** **PLUGIN**.  
            For STDIN Plugin, we don’t need to give any other option/configurations.
   4. Let’s add an **OUTPUT PLUGIN as we want to send events somewhere**.
      1. **output** { **stdout**{} } :
         1. **output {}**: is an object (technically **Output** **Hash**)
         2. **stdout**: is an **OUTPUT PLUGIN**
7. A picture containing text, screenshot, software, multimedia software

   Description automatically generated  
   Sending Pipeline configuration as argument when starting up the logstash is not that efficient as configuration gets bulky soon and then difficult to craft and read.  
   But for getting started and testing and debugging purpose, this argument approach is convenient.
8. **Let’s move the pipeline configuration to a configuration file.**
9. A screenshot of a computer

   Description automatically generated
10. A picture containing screenshot, text, display

    Description automatically generated
11. **Each** time we referred to the output as Processed event even though we didn’t do any.  
    But Logstash does a bit of the work by itself.   
    Like we can see two fields’ values are added automatically by the Logstash
    1. Timestamp
    2. Host Name.
12. Though those fields’ values are being output but these values have field names too.  
    We can actually see this by modifying our pipeline to use a codec named **Ruby Debug** which will output the processed events in JSON meaning that we can see the field names.
13. **What is a codec**?
    1. Codec is a way to encode or decode data.
14. Since we have added the codec in the output log of the pipeline, the codec is used to decode the processed output.
15. A screen shot of a computer

    Description automatically generated with low confidence
16. This example is not that useful as we’re just able to send a given string somewhere together with a timestamp.
17. In the upcoming lectures, we will see that we could add our own fields based on inputs.  
    **An example that we will be going through a bit later is passing access logs for the Apache Web Server.**Each entry contains a number of data such as request path, IP address, and more.  
    We will see each of these being turned into a field which we can then reference or do something with or just use for the output.  
    For example, if we use the Elasticsearch output, we need a number of fields when turning the processed events into a document.  
    We will see a plenty of examples of how to work with fields in the upcoming lectures.  
    So the moral of the story is that we will see custom fields in the outputs as we begin working our way through some useful examples.  
    Before getting to that, let’s take a quick moment to recap on what we accomplished in this lecture.
18. **Recap**:
    1. 1st we saw how to pass pipeline configuration as an argument when running Logstash binary (**-f**)
    2. Then we saw how to define pipeline configuration in a file and send the file to Logstash binary when running it.
    3. We added a pipeline that simply takes input from a terminal and outputs the processed event in JSON format for which we used Rubydebug Codec.  
       That is the most basic example of using Logstash.
19. This example is not that useful apart from getting started.  
    So, let’s begin expanding on the example and introduce some more Logstash concepts step by step.