Log Aggregation

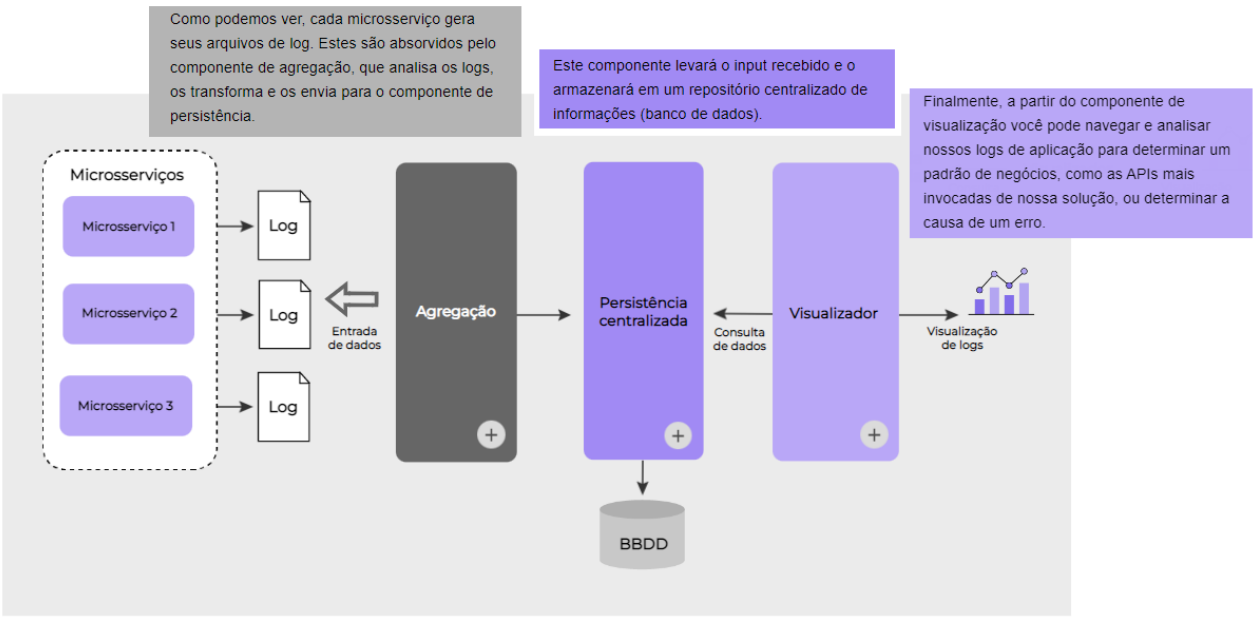
Padrão Log aggregation

Nesta aula, aprenderemos sobre o padrão de Log aggregation. Isto nos permite centralizar as informações de logins de nossas aplicações para determinar a causa de erros, ou analisar o comportamento de nossa aplicação de forma eficaz, sem a necessidade de ir aos arquivos de log de cada uma das instâncias de nossos microsserviços.

Este padrão é composto de três componentes fundamentais:

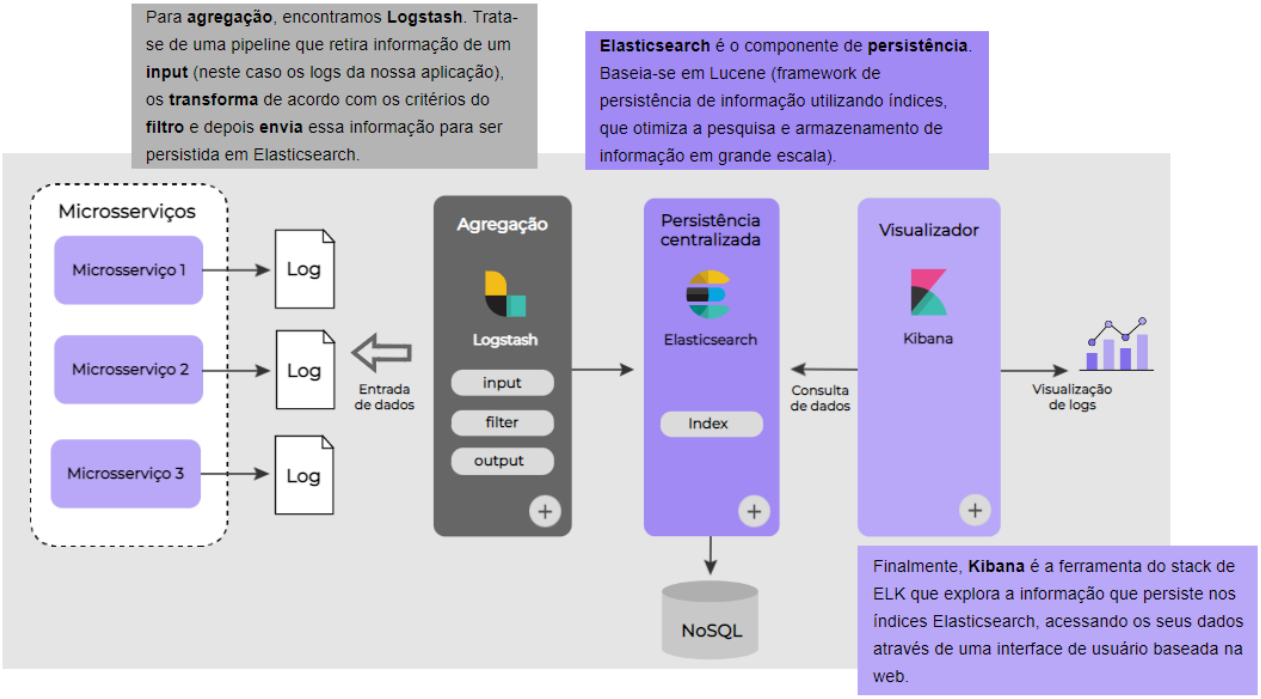
- Agregação
- Persistência centralizada
- Visualizador de logs

Vamos avançar para conhecê-los em detalhes.



Stack ELK

A partir do que foi visto anteriormente, vamos aprender um novo conceito: ELK. É uma stack de tecnologias (Elasticsearch, Logstash e Kibana) a ser utilizada na gestão de agregação de logs, implementando uma solução para cada um dos componentes que vimos na arquitetura genérica.



Instalações do ELK stack

Antes de começar a desenvolver um exemplo prático, precisamos baixar os três frameworks que compõem a solução ELK. Veja como fazer isso.

Implementação do ELK stack

Agora, a partir de um exemplo, veremos como implementar o ELK Stack. Vamos tomar como base um microsserviço básico na Spring Boot que faz o registro, tanto de informações básicas como de exceções, em duas URLs diferentes. Estes logs devem ser centralizados em um servidor ELK, para o qual devemos primeiro configurar a agregação destes logs dentro de nossa Logstash.

Como vimos, Logstash é um pipeline que tem diferentes ações para levar as informações desde o ponto de partida até o destino. Este pipeline é materializado em um arquivo de configuração que alimenta a Logstash em sua inicialização. Ele é composto do seguinte formato básico onde cada setor indica o que fazer na etapa do pipeline:

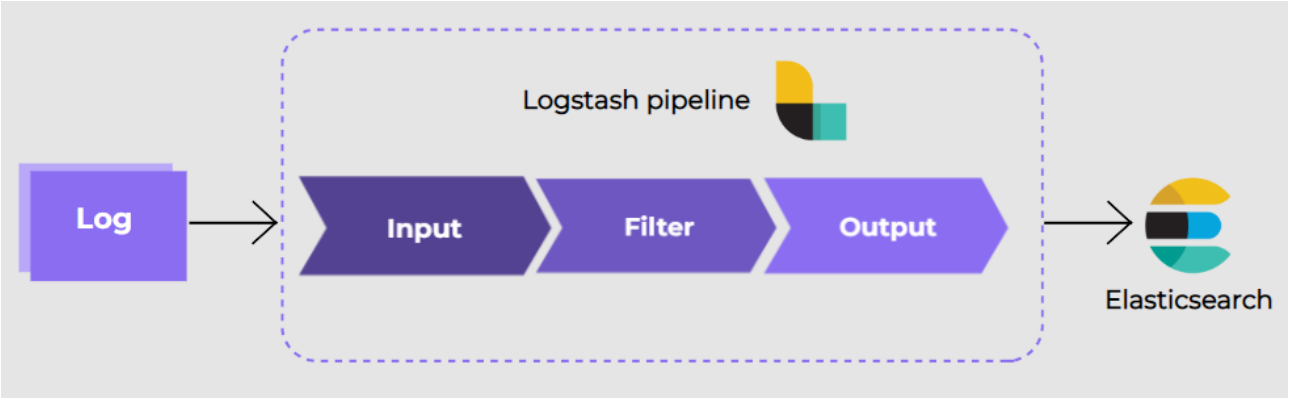
```
input{
}

filter{
```

}

output{

}



O **input** será um "file ".

O **type** do file é java.

O **path** é o caminho onde temos o nosso arquivo LOG

Codec indica a lógica que as mensagens lidas devem ter a fim de as associar ao mesmo evento. Indicamos se uma mensagem está ou não associada a um **padrão** de mensagem para tomá-la ou não como um único evento e adicioná-la antes ou depois da mensagem original. Dentro do codec, temos diferentes parâmetros que serão as instruções para processar esse input. Neste caso: multiline, pattern, negate e o what.

Indica que podemos receber múltiplas linhas como entrada.

É uma expressão regular tal que se o input corresponder a essa *regular expression*, esse input deve ser considerado como um evento e eventualmente ser processado.

Indica se a expressão de pattern deve ser negada para considerar um input como um evento a ser logado. Por default, é falso.

Configuração do input

O nosso objetivo é ler o arquivo de log da nossa aplicação como ponto de entrada. Para isso, acrescentamos as seguintes linhas no setor de **input**.

Clique no código colorido para ver mais informações

#código

```
input {
  file {
    type => "java"
    path => "C:/log/dh-spring-elk.log"
    codec => multiline {
      pattern => "^%{YEAR}-%{MONTHNUM}-%{MONTHDAY} %{TIME}.*"
      negate => "true"
      what => "previous"
    }
  }
}
```

Indica se o match encontrado deve ser associado ao evento anterior ou ao evento seguinte (geralmente o anterior). Por exemplo, o acesso à base de dados é interrompido e o sistema registra uma exceção após a falha. No entanto, vamos supor que registramos um pico de transações, a base de dados ainda está online, mas vai falhar nos próximos minutos, caso esse em que poderíamos associar um padrão ao evento seguinte.

Configuração do filter

Vamos supor que queremos filtrar através de queries as mensagens relacionadas com uma exceção. Para isso, podemos adicionar tags para facilitar esta tarefa de pesquisa a partir do **Kibana**.

#código

```
filter {
#se a linha do log tiver a palavra "at" depois de um tab, significa que é
uma exceção.
  if [message] =~ "\tat" {
    grok {
      match => ["message", "^(\tat)"]
      add_tag => ["stacktrace"]
    }
  }
}
```

Configuração do output

Finalmente, temos de associar a saída do Logstash ao **Elasticsearch** que iniciamos, a fim de persistir a informação.

#código

```
output {
  elasticsearch {
    hosts => ["localhost:9200"]
  }
  stdout {
    codec => rubydebug
  }
}
```

Conclusão

O stack ELK é uma excelente implementação do padrão Log aggregation. Ele permite desacoplar e padronizar a problemática do login em nossos microsserviços, cumprindo três objetivos cruciais:

1. Que os logs fiquem em um único lugar, mesmo que nossa arquitetura esteja completamente distribuída em microsserviços.
2. A padronização da formatação sem perda de flexibilidade.
3. Desacoplar a visualização de modo que, posteriormente, seja possível visualizar e explorar os logs com qualquer outra ferramenta.