

ELK log analysis

Alexandr Mikula

Tutorial VMs

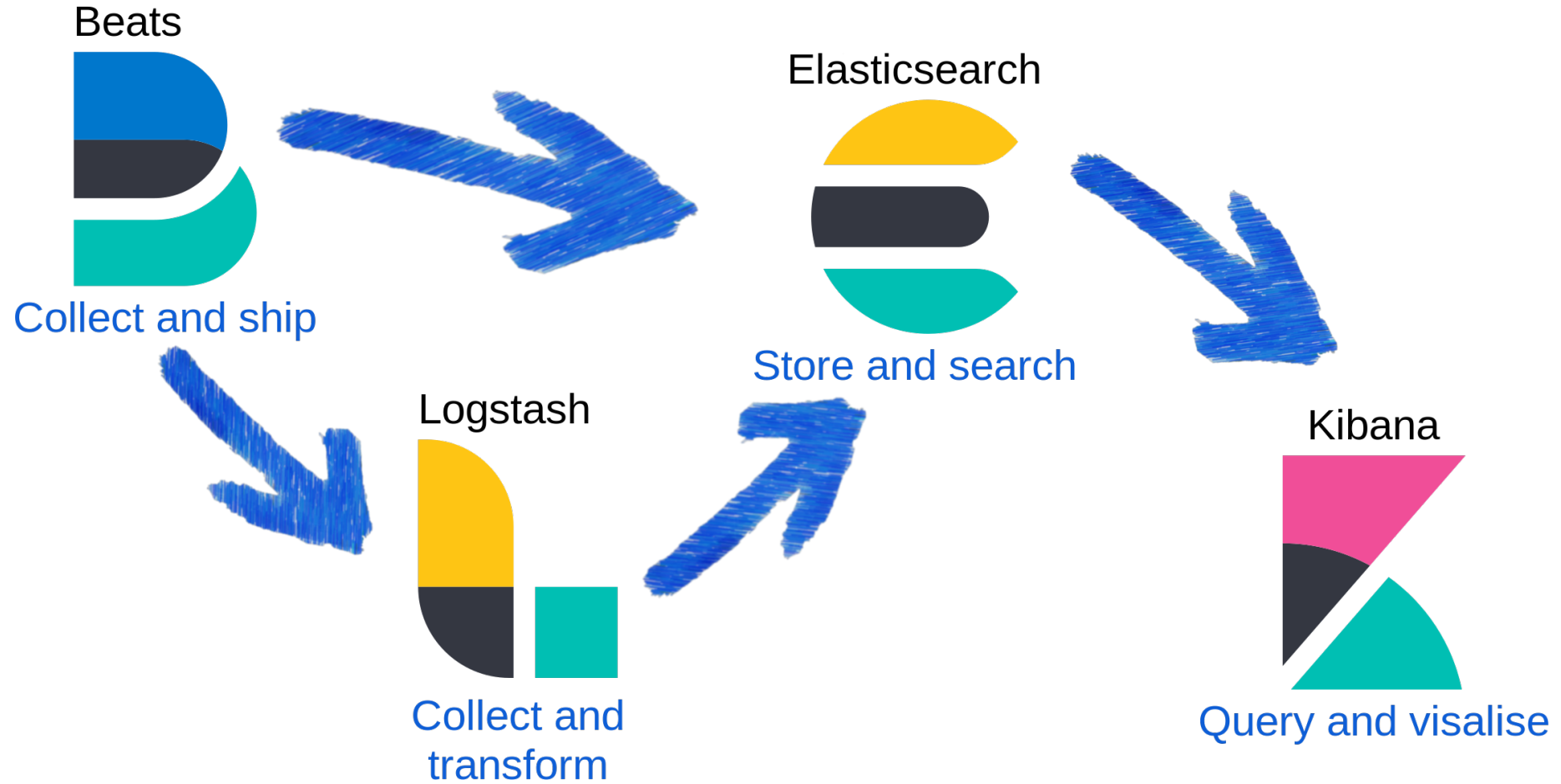
- Each participant has 3 VMs for tutorial
 - One has a public access and is used as access point to the rest
 - Hostname is *elk-X-a**
 - DNS address is *elk-X-a.scc.kit.edu*
 - Other two are accessible only through first one
 - Hostnames *elk-X-b* and *elk-X-c**
 - Each VM has 8GB RAM, 2 CPU cores with 60GB of HDD
 - Most of tasks involve all 3 VMs unless specified

* “X” stands for instance number

Tutorial VMs

- Nodes B & C are accessible only through node A
 - You can use `ssh root@elk-X-b -o ProxyJump=root@elk-X-a` on ssh cli
 - It is suggested to add your ssh public key to `~/.ssh/authorized_keys` of the root user on all machines
 - `sudo sh -c "echo '<ssh-public-key>' >> /root/.ssh/authorized_keys"`
 - Feel free to install your favourite text editor
 - eg. `# yum -y install nano`

ELK architecture



Outline of tutorial

- Repository setup
- Elasticsearch cluster setup and basics
- Filebeat installation and configuration
- Kibana setup with basic security
- Logstash and Filebeat stack setup
- Log data transformation with Logstash (basic to advanced)
- Kibana visualizations of the data

Materials on <https://git.io/fjNBe>

Repository setup



Create file */etc/yum.repos.d/elasticsearch.repo* with following content

```
[elasticsearch-7.x]
name=Elasticsearch repository for 7.x packages
baseurl=https://artifacts.elastic.co/packages/7.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md
```

* You can also “curl” it from materials on github

```
# curl -L https://git.io/fjNe0 > /etc/yum.repos.d/elasticsearch.repo
```

Elasticsearch setup



- Install elasticsearch package
 - `# yum install elasticsearch`
 - You will need to confirm downloaded gpg-key (or use -y switch)
- Edit service for memlock
 - `# systemctl edit elasticsearch`
 - ([elasticsearch/mem_lock.conf](#))

```
[Service]  
LimitMEMLOCK=infinity
```

Elasticsearch setup

elasticsearch.yml



- Edit */etc/elasticsearch/elasticsearch.yml*
 - cluster
 - name to anything you like – eg. gks2019
 - initial_master_nodes: [“node-a”, “node-b”, “node-c”]
 - discovery:
 - seed_hosts: [“elk-X-a”, “elk-X-b”, “elk-X-c”]
 - bootstrap.memory_lock: true (only uncomment)

Elasticsearch setup

elasticsearch.yml



- Edit */etc/elasticsearch/elasticsearch.yml*
 - network.host: “::0”
 - node:
 - name: unique for each node (eg. node-a)
 - attr.rack: can be used to customize data allocation
 - Not used during tutorial
 - data: true
 - master: true

Example config at [elasticsearch/elasticsearch.yml](#)

Elasticsearch setup finish



- Edit `/etc/elasticsearch/jvm.options`
 - `-Xms` & `-Xmx`
 - To $\frac{1}{2}$ of RAM but no more than 30GB
 - Details: [Heap size](#)
- Reload systemctl & start service
 - `# systemctl daemon-reload && systemctl start elasticsearch`
- Finally check cluster state:
 - `# curl -s 'localhost:9200/_cluster/health?pretty'`

Kibana & httpd setup Without TLS



Do this on publicly accessible node only (node A)

- Install packages
 - `# yum -y install httpd kibana`
- curl following files
 - [kibana/httppasswd.users](#) → */etc/httpd*
 - [kibana/http-kibana.conf](#) → */etc/httpd/conf.d*

Kibana & httpd setup With TLS



Do this on publicly accessible node only (node A)

- Install packages
 - `# yum -y install httpd kibana mod_ssl`
- curl following files
 - [kibana/httppasswd.users](#) → */etc/httpd*
 - [kibana/https-kibana.conf](#) → */etc/httpd/conf.d*
 - [kibana/create_cert.sh](#) → */anywhere/you/like/it*
- Run `create_cert.sh` (creates self-signed ssl certificate)
 - `sh create_cert.sh`



FZU

Fyzikální ústav
Akademie věd
České republiky

Kibana & httpd setup



- Start httpd
 - `# systemctl start httpd`
- Edit `/etc/kibana/kibana.yml` (example [kibana/kibana.yml](#))
 - `server.host: localhost`
- Start kibana
 - `# systemctl start kibana`
- Connect to kibana in web browser
 - User is kibana and password is kibanakibana (you can change this)

Logstash setup



Do this on the non-public nodes only (nodes B & C)

- Install logstash (requires java)
 - `# yum -y install logstash java-11-openjdk-devel`
- Edit </etc/logstash/logstash.yml>
 - `xpack.monitoring.enabled: true`
 - `xpack.monitoring.elasticsearch.hosts`: list of Elasticsearch hosts
 - In format `http://host:9200`
 - Beware it is not https !!!
 - Feature not available in *-oss versions



FZU

Fyzikální ústav
Akademie věd
České republiky

Logstash setup

Tutorial only



- Edit logstash service so it is run as root
- Used only during tutorial, not advised in production
- # systemctl edit logstash

[Service]

User=root

group=root

Logstash

logstash.conf



- [/etc/logstash/conf.d/logstash.conf](#)
 - Main config file describing input transformations
 - Basic structure:
 - Input definitions
 - What to process
 - Filter definitions
 - How to process it
 - Output definitions
 - Where to send it

Logstash

logstash.conf/input



- Input describes data source
- Supports lot of different plugins
 - File, beats, elasticsearch, exec, and [many more](#)

```
input {  
  beats {  
    port      => 5044  
    host      => "::"  
  }  
  file {  
    path      => "/tmp/test-file"  
    type      => "test"  
    start_position => [ "beginning" ]  
  }  
}
```

Logstash

logstash.conf/output



- Output describes where you want to store transformed data
- Many different plugins
 - file
 - elasticsearch
 - email
 - [many more](#)

```
output {  
  stdout {  
    codec => rubydebug { metadata => true }  
  }  
  if [type] == "test" {  
    file {  
      path          => "/tmp/test.out"  
      file_mode     => 0644  
      flush_interval => 0  
    }  
  }  
  else {  
    elasticsearch {  
      hosts          => [ "elk-X-a:9200",  
                          "elk-X-b:9200", "elk-X-c:9200"]  
    }  
  }  
}
```

Logstash

logstash.conf/filter



- Filter config is the main part of data transformation work
- Lots of filters to apply to data
 - date – date and time processing
 - kv – key and value parsing
 - mutate – transform already known data
 - grok – swiss army filter for logstash
 - ruby – when even grok is not enough
 - [many other](#)

Logstash

logstash.conf/filter



We will get to filters after we setup filebeat to ship data

- Now just a small demo
 - Add example filter to config file
 - Run logstash in foreground
`/usr/share/logstash/bin/logstash "--path.settings" "/etc/logstash"`

```
filter {  
  if [type] == "test" {  
    mutate {  
      uppercase =>  
        [ "message" ]  
    }  
  }  
}
```


Filebeat setup



- Install filebeat
 - `# yum -y install filebeat`
- Edit [/etc/filebeat/filebeat.yml](#)
 - Backup old file and create new
 - This is only to test connection and to see what filebeat does
 - Do only on one node

```
filebeat:
  inputs:
    - type: log
      paths:
        - /var/log/messages
      fields:
        type: test
        fields_under_root: true
  output:
    logstash:
      hosts:
        - "nodeB:5044"
```

[filebeat/test.yml](#)



FZU

Fyzikální ústav
Akademie věd
České republiky

Filebeat setup



- Start logstash in foreground (nodeB)
 - `# /usr/share/logstash/bin/logstash "--path.settings" "/etc/logstash"`
- Start filebeat (node with configured test instance)
 - `# systemctl start filebeat`
- Observe sent events
- Stop filebeat
 - `# systemctl stop filebeat`

[logstash/sample_filebeat_event.txt](#)



FZU

Fyzikální ústav
Akademie věd
České republiky

Now some more serious work

Collect log sources



- Stop any running logstash
- Look into /var/log on all machines
- Create /etc/filebeat/filebeat.yml
 - Try to collect all interesting log files on system
 - You need to take a look into files (eg. with less)
 - Add custom field to every log type you find, so you can sort it out afterwards

Now some more serious work

Multiline events



- Try to determine if there are any events over more lines
 - You can join them with filebeat
 - This appends all lines starting with whitespace to a line that does not start with one
 - [More on multiline](#)

multiline:
pattern: '^\\s+'
negate: true
match: before

Now some more serious work



- Start all filebeat instances on all machines when you feel ready
- Now we need to analyze events which will be shipped to logstash
- We know what data will be filebeat adding to events from our sample event

```
{
  "input" => {
    "type" => "log"
  },
  "agent" => {
    "hostname" => "elk-X-b",
    "id" => "ca9d2d29-b991-43b5-91e9-c360b1f1dd56",
    "ephemeral_id" => "e7267066-5020-43eb-b874-e0891db70fd6",
    "type" => "filebeat",
```

Now some more serious work

Sample event usable fields



- Agent
 - Hostname
- @metadata
 - beat
- Message !!!
- @timestamp (time when event was collected)
- Any custom field (eg. type)

Aug 25 17:12:21 elk-X-a sshd[10623]: Accepted publickey for root from 10.0.0.1 port 32816
ssh2: RSA SHA256:Eq+vWr09lautVQ2BB5vG0Uze0BiJmPt0lQtrYbCag38

Event analysis

Ssh login (secure log)



Aug 25 17:12:21 elk-X-a sshd[10623]: Accepted publickey for root from 10.0.0.1 port 32816
ssh2: RSA SHA256:Eq+vWr09lautVQ2BB5vG0Uze0BiJmPt0IQtrYbCag38

- Date and time
- Hostname
- Program
- Event
 - What, For, From, With

How to get info into ES



- We will need two filters to get data into ES right
- Grok
- Date
- With grok we will get data from event into searchable fields
- With date we will set right @timestamp to event

Grok



- Grok works as a powerful regex matching tool
- It has many built in patterns to use
- We will be using only basic patterns to demonstrate how to build your own

```
Aug 25 17:12:21 elk-X-a sshd[10623]: Accepted publickey for root from 10.0.0.1 port 32816  
ssh2: RSA SHA256:Eq+vWr09lautVQ2BB5vG0Uze0BiJmPt0lQtrYbCag38
```

```
%{SYSLOGTIMESTAMP:["@metadata"][date]} %{SYSLOGHOST:host} %{SYSLOGPROG}:  
%{GREEDYDATA} for %{WORD:user} from %{SYSLOGHOST:remote_host} port %{NUMBER:port}  
%{GREEDYDATA}: %{WORD:key_type} %{NOTSPACE:hash_algo}:%{NOTSPACE:key_hash}
```

Date



- Now we have `[@metadata][date]` thanks to grok
- Date plugin is straightforward
- Timezone is by default same as logstash host have

```
date {  
  match => [  
    "[@metadata][date]",  
    "MMM dd HH:mm:ss",  
    "MMM d HH:mm:ss"  
  ]  
}
```

Grok part 2



- The pattern we built before will match only secure logs about ssh logins
- Other lines will fail and timestamp will not be updated
- Try to prepare other grok pattern to match also different lines
- Grok tries patterns one after another until it matches or have no more patterns
- Handy tool for grok debugging <https://grokdebug.herokuapp.com/>
 - Also part of kibana

Aug 25 16:06:58 elk-X-a sshd[1829]: pam_unix(sshd:session): session opened for user root by (uid=0)

Kibana



Here we will try live demo on data we have already collected.

Other logs to train on



You will find other logs to train on in logs directory in materials.

You can reach me on github or send me an email

mikula@fzu.cz

Thank you for your attention