



Advertisement



Ifrit LTD

DevOps Principles & Best Practices by Kayan Azimov



Dockerizing Jenki (Filebeat, Elastic

Published August 22, 2017

l logs with ELK stack Logstash and Kibana)

This is 4th part of Docker
previous parts here:

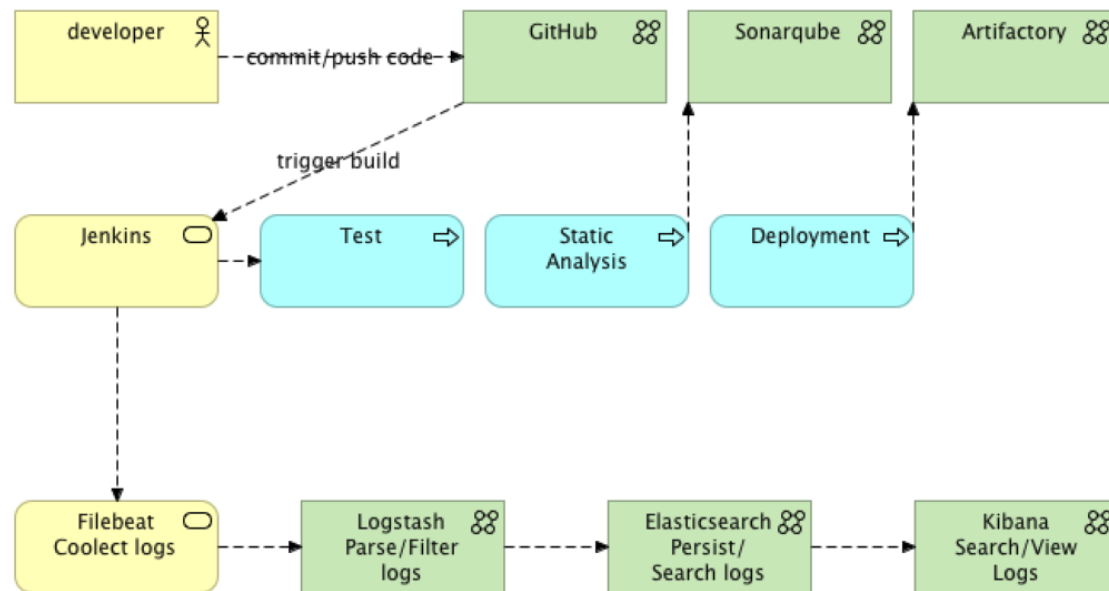
ns series, you can find more about

[Dockerizing Jenkins, Part 1: Declarative Build Pipeline With SonarQube Analysis](#)

[Dockerizing Jenkins, part 2: Deployment with maven and JFrog Artifactory](#)

[Dockerizing Jenkins, part 3: Securing password with docker-compose, docker-secret and jenkins credentials plugin](#)

Today we are going to look at managing the Jenkins build logs in a dockerized environment.



Normally, in order to view the build logs in Jenkins, all you have to do is to go to particular job and check the logs. Depending on a log rotation configuration, the logs could be saved for N number of builds, days, etc, meaning the old jobs logs will be lost.

Our aim in this article will be persisting the logs in a centralised fashion, just like any other application logs, so it could be searched, viewed and monitored from single location.

We also will be running Jenkins in Docker, meaning if container is dropped and no other means are in place, like mounting the volume for logs from a host and taking the backup the logs will be lost.

As you may have already heard, one of the best solutions when it comes to logging is called ELK stack.

The Idea with ELK stack is you collect logs with Filebeat(or any other *beat), parse, filter logs with longstash and then send them to elasticsearch for persistence, and then view them in kibana.

On top of that, because logstash is heavyweight jrubby app on JVM , you either skip it at all or use a way smaller application called Filebeat, which is a logstash log forwarder, all it does, collects the logs and sends to longstash

for further processing.

In fact, if you don't have any filtering and parsing requirements you can skip the logstash at all and use Filebeat's elastic output for sending the logs directly to elasticsearch.

In our example we will try to use all of them, plus, we won't be running Filebeat in a separate container, but instead, will install it right inside of our Jenkins image, because Filebeat is small enough. I also wanted to demonstrate how we can install anything on our Jenkins image, so it is more interesting.

So the summary of what we are going to look at today is:

1. **Prepare our dockerized dev environment with Jenkins, Sonarqube and JFrog artifactory running the declarative pipeline**
2. **Download and install Filebeat on our Jenkins image**
3. **Configure Filebeat so it knows how and where collect the Jenkins logs and how to send them further to logstash**
4. **Configure and run logstash in a docker container**
5. **Configure and run elasticsearch in a docker container**
6. **Configure and run kibana in a docker container**

1. Prepare our dockerized dev environment with Jenkins, Sonarqube and JFrog artifactory running the declarative pipeline

In this example we will use Jenkins image we created earlier in the [part 3](#) of these series. First thing first, let's checkout the project:

```
1 | git clone https://github.com/kenych/dockerizing-jenkir
2 |     cd dockerizing-jenkins && \
3 |     git checkout dockerizing_jenkins_part_3_docker_com
4 |     ./runall.sh
```

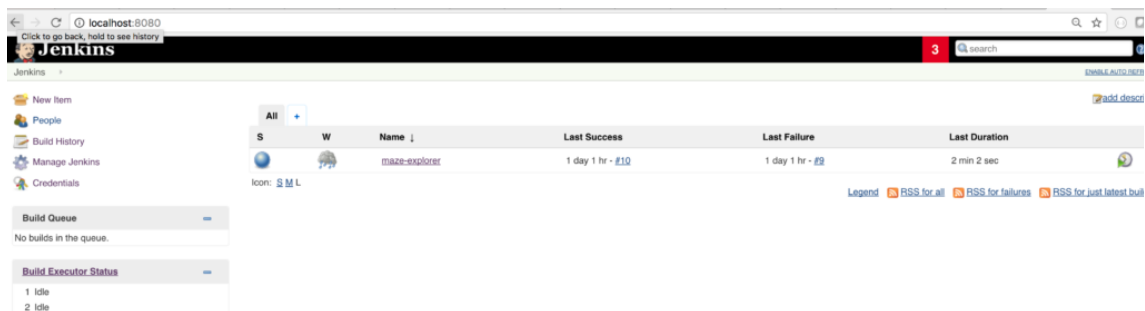
Let's see what runall.sh does:

```
1 | #!/usr/bin/env bash
2 |
3 | #get needed stuff 1st
4 | ./download.sh
5 |
6 | #clean anything with same name to get rid of clashes
```

```
7 | docker-compose down
8 |
9 | #update with actual password
10 | echo "password" > ./secrets/artifactoryPassword
11 |
12 | #update older jenkins image, make sure it doesnt use
13 | docker-compose build --no-cache
14 |
15 | #run all
16 | docker-compose up
```

We firstly download all images needed to prepare our environment, plus make maven available for Jenkins container. Then make sure no other services are running with same name by shutting them down with docker-compose down command, as if we had it from previous part of this tutorial, docker-compose would fails during creation of services. Then we set up password for artifactory, I could have committed it, but I am a bit paranoiac about committing passwords to repo. Then we build our Jenkins image and run the holy build btrinity of Jenkins, sonarqube and artifactory containers.

By now you should have everything ready to start and you can check by going to <http://localhost:8080> and giving a try to a build to run, hopefully successfully:



2. Install Filebeat on our Jenkins image

So, now let's move forward, in order to install and run Filebeat we need to implement these steps:

- Download and install Filebeat Debian package
- Configure Filebeat with inbound and outbound context

- Start the Filebeat service

To do so let's add these lines to Dockerfile:

```
1 USER root
2
3 RUN curl -o /tmp/filebeat_1.0.1_amd64.deb https://dowr
4     dpkg -i /tmp/filebeat_1.0.1_amd64.deb && apt-get i
5
6 COPY filebeat.yml /etc/filebeat/filebeat.yml
```

As you see, we download and install the package first, then copy the config to image. Starting service is shown a bit later.

3. Configure Filebeat so it knows where to collect the Jenkins logs from and how to send them further to logstash

Now let's configure Filebeat, we can do it using yaml file:

```
1 filebeat:
2   prospectors:
3     -
4     paths:
5       - "/var/jenkins_home/jobs/*/builds/*/log"
6
7   output:
8     logstash:
9       hosts: ["logstash:5044"]
```

As you see we specified the path where to find Jenkins logs and then instructed to send the logs to logstash. If you are curious about logs path and why it looks like that, you can get inside of your Jenkins container and have a look:

```
1 dockerizing-jenkins git:(master) x docker exec -it dc
2 jenkins@3713764d9f9d:/$ ls -l /var/jenkins_home/jobs
3 -rw-r--r-- 1 jenkins jenkins 64009 Aug 18 21:08 /var/
4 -rw-r--r-- 1 jenkins jenkins 53397 Aug 18 21:22 /var/
5 -rw-r--r-- 1 jenkins jenkins 53397 Aug 18 21:22 /var/
6 -rw-r--r-- 1 jenkins jenkins 64009 Aug 18 21:08 /var/
7 -rw-r--r-- 1 jenkins jenkins 64009 Aug 18 21:08 /var/
8 -rw-r--r-- 1 jenkins jenkins 53397 Aug 18 21:22 /var/
9 jenkins@3713764d9f9d:/$ exit
10 exit
```

As you can see, this is the way Jenkins keeps its build logs.

In a hosts we specified "logstash:5044" that is because we will link logstash later in Jenkins so Jenkins containers will be able to resolve its name to logstash containers IP.

Finally we need to start Filebeat service, and as our startup script gets more complex lets move it to separate file, create a file entrypoint.sh:

```
1 /etc/init.d/filebeat start
2
3 exec bash -c /usr/local/bin/jenkins.sh
```

And then refer to it in Dockerfile:

```
1 COPY ["entrypoint.sh", "/"]
2
3 RUN chmod +x /entrypoint.sh
4
5 ENTRYPOINT ["/bin/bash", "-c", "./entrypoint.sh"]
```

Our finale dockerfile should look like below:

```
1 FROM jenkins:2.60.1
2
3 MAINTAINER Kayan Azimov
4
5 ENV JAVA_OPTS="-Djenkins.install.runSetupWizard=false"
6
7 COPY plugins.txt /usr/share/jenkins/ref/plugins.txt
8 RUN /usr/local/bin/install-plugins.sh < /usr/share/jenkins/ref/plugins.txt
9
10 COPY groovy/* /usr/share/jenkins/ref/init.groovy.d/
11
12 USER root
13
14 RUN curl -o /tmp/filebeat_1.0.1_amd64.deb https://dow
15     dpkg -i /tmp/filebeat_1.0.1_amd64.deb && apt-get
16
17 COPY filebeat.yml /etc/filebeat/filebeat.yml
18
19 COPY ["entrypoint.sh", "/"]
20
21 RUN chmod +x /entrypoint.sh
22
23 ENTRYPOINT ["/bin/bash", "-c", "./entrypoint.sh"]
```

Now let's build the image and make sure everything still works, this is a good pattern, baby steps to make sure you know the point where it last was

in a working state:

```
1 | docker-compose up --build
```

So let's check Filebeat is up and running:

```
1 | → dockerizing-jenkins git:(master) x docker exec -it
2 | root@a2c965f635dd:/#
3 | root@a2c965f635dd:/# ps aux
4 | USER      PID %CPU %MEM    VSZ   RSS TTY      STAT   S
5 | root         1  78.7  12.5 5455056 893372 ?        Ssl    2
6 | root        18   0.0   0.0   9400    576 ?        Sl     2
7 | root        19   0.2   0.1 359068  11532 ?        Sl     2
8 | root       2022   0.3   0.0   19960   3504 ?        Ss     2
9 | root       2028   0.0   0.0   38380   3216 ?        R+     2
10 | root@a2c965f635dd:/# exit
11 | exit
```

4. Configure and run logstash in a docker container

Now, instead of keeping adding more and more services to our already overloaded stack, let's create another docker-compose file, so we can have separation of concerns, and add containers related to ELK there. File docker-compose-elk.yml:

```
1 | version: "3.1"
2 |
3 | services:
4 |
5 |     logstash:
6 |         image: logstash:2
7 |         volumes:
8 |             - ./:/config
9 |         command: logstash -f /config/logstash.conf
10 |
```

As you see we created a new file and added logstash to it, it is pretty old image and I just took it from the stack I setup up long time ago, if you want to get latest you will need to make sure versions matrix match:

Supported Versions 2.x

Elasticsearch	Kibana	Marvel	Shield	Graph	Reporting	Beats**	Logstash**	ES-Hadoop (jar)
2.0.x	4.2.x	2.0.x	2.0.x	N/A	N/A	1.0.x - 5.5.x	2.0.x - 5.5.x	2.2.x - 2.4.x
2.1.x	4.3.x	2.1.x	2.1.x	N/A	N/A	1.0.x - 5.5.x	2.0.x - 5.5.x	2.2.x - 2.4.x
2.2.x	4.4.x	2.2.x	2.2.x	N/A	N/A	1.0.x - 5.5.x	2.0.x - 5.5.x	2.2.x - 2.4.x
2.3.x	4.5.x	2.3.x	2.3.x	2.3.x	N/A	1.0.x - 5.5.x	2.0.x - 5.5.x	2.2.x - 2.4.x
2.4.x	4.6.x	2.4.x	2.4.x	2.4.x	2.4.x	1.0.x - 5.5.x	2.0.x - 5.5.x	2.2.x - 2.4.x

Beats to Logstash Compatibility

Beats	Logstash
1.0.x	2.0.x-2.4.x
1.1.x	2.0.x-2.4.x
1.2.x	2.0.x-2.4.x
1.3.x	2.0.x-5.0.x
5.0.x	2.0.x-5.5.x
5.1.x	2.0.x-5.5.x
5.2.x	2.0.x-5.5.x
5.3.x	2.0.x-5.5.x
5.4.x	2.0.x-5.5.x
5.5.x	2.0.x-5.5.x

Now we need to configure logstash in logstash.conf:

```
1 | input { beats {          port => 5044      } }
2 | output {
3 |     stdout { codec => rubydebug }
4 |
5 | }
```

With this config all it does is showing the logs on the output so we can check it is actually working. Another baby step, let's run the new stack:

```
1 | docker-compose -f docker-compose-elk.yml up
```

Notice that we need to specify compose file explicitly this time as it doesn't have the default name, which is already taken by trinity.

```

$ docker-compose (docker-compose)
mysonar_1 | 2017.08.22 08:56:21 INFO app[o.s.p.m.Monitor] Process[web] is up
mysonar_1 | 2017.08.22 08:56:21 INFO app[o.s.p.m.ProcessLauncher] Launch process[ce]: /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Xms512M -Xmx2048M -XX:HeapDumpOnOutOfMemoryError -Djava.io.tmpdir=/opt/sonarqube/temp -javaagent:/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/management-agent.jar -cp ./lib/common/*:/lib/server/*:/lib/ce/*:/opt/sonarqube/h2/h2-1.3.176.jar org.sonar.ce.app.CeServer /opt/sonarqube/temp/sq-process7774671076825129618properties
myjenkins_1 | --> setting agent port for jnlp
myjenkins_1 | Aug 22, 2017 8:56:21 AM hudson.TcpSlaveAgentListener$ConnectionHandler run
myjenkins_1 | INFO: Accepted Ping connection #1 from /172.21.0.4:50476
myjenkins_1 | --> setting agent port for jnlp... done
mysonar_1 | 2017.08.22 08:56:22 INFO ce[o.s.s.p.ProcessEntryPoint] Starting ce
mysonar_1 | 2017.08.22 08:56:22 INFO ce[o.s.ce.app.CeServer] Compute Engine starting up...
mysonar_1 | 2017.08.22 08:56:23 INFO ce[o.s.e.plugins] [Hog] modules [], plugins [], sites []
mysonar_1 | 2017.08.22 08:56:24 INFO ce[o.s.s.e.ElasticClientProvider] Connected to local Elasticsearch: [127.0.0.1:9001]
mysonar_1 | 2017.08.22 08:56:24 INFO ce[o.s.sonar.db.Database] Create JDBC data source for jdbc:h2:tcp://localhost:9092/sonar
mysonar_1 | 2017.08.22 08:56:24 WARN ce[o.s.d.DatabaseChecker] H2 database should be used for evaluation purpose only
mysonar_1 | 2017.08.22 08:56:25 INFO ce[o.s.s.p.ServerFileSystemImpl] SonarQube home: /opt/sonarqube
myjenkins_1 | Aug 22, 2017 8:56:26 AM org.jenkinsci.plugins.workflow.job.WorkflowRun finish
myjenkins_1 | INFO: maze-explorer #11 completed: FAILURE
mysonar_1 | 2017.08.22 08:56:26 INFO ce[o.s.c.c.PluginRepository] Load plugins
mysonar_1 | 2017.08.22 08:56:27 INFO ce[o.s.s.c.q.PurgeCeActivities] Delete the Compute Engine tasks created before Thu Feb 23 08:56:27 UTC 2017
mysonar_1 | 2017.08.22 08:56:27 INFO ce[o.s.ce.app.CeServer] Compute Engine is up
mysonar_1 | 2017.08.22 08:56:27 INFO app[o.s.p.m.Monitor] Process[ce] is up
mysonar_1 | 2017.08.22 08:56:27 INFO app[o.s.application.App] SonarQube is up

$ docker-compose (docker-compose)
$ dockerizing-jenkins git:(master) X docker-compose -f docker-compose-elk.yml up
WARNING: Found orphan containers (dockerizingjenkins_myjenkins_1, dockerizingjenkins_mysonar_1, dockerizingjenkins_artifactory_1) for this project. If you removed or renamed this service in your com
you can run this command with the --remove-orphans flag to clean it up.
Starting dockerizingjenkins_logstash_1 ...
Starting dockerizingjenkins_logstash_1 ... done
Attaching to dockerizingjenkins_logstash_1
logstash_1 | log4j:WARN No appenders could be found for logger (io.netty.util.internal.logging.InternalLoggerFactory).
logstash_1 | log4j:WARN Please initialize the log4j system properly.
logstash_1 | log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
logstash_1 | {"timestamp":"2017-08-21T22:16:10.513000+0000", "message":"Pipeline main started"}
logstash_1 | {
logstash_1 |   "message" => "Started by user ve[0bha:///402pYGrCcbPY0KYG20xXNwpIv4QCTN3]d9FzIsAAALh-LCAAMAAAP9b5a8btlIQTGJNKU4P08v0T+v008nVc83Py0Ux60yILUoJz0W2y+/JJUBAhI2G8gqihhK0M
logstash_1 |   "version" => "1",
logstash_1 |   "timestamp" => "2017-08-21T22:16:40.196Z",
logstash_1 |   "beat" => {
logstash_1 |     "hostname" => "48bfaec968ca",
logstash_1 |     "name" => "48bfaec968ca"
logstash_1 |   },
logstash_1 |   "count" => 1,
logstash_1 |   "fields" => nil,
logstash_1 |   "input_type" => "log",
logstash_1 |   "offset" => 0,
logstash_1 |   "source" => "/var/jenkins_home/jobs/maze-explorer/builds/11/log",
logstash_1 |   "type" => "log",

```

As you can see logstash now picking up messages sent by Filebeat.

5.Configure and run elasticsearch in a docker container

Next step adding elasticsearch bit to the stack:

```
1 | version: "3.1"
```

We also added links and dependency on elastic to logstash, so it can see it and wait for it as well. Don't forget to configure logstash to send messages to elastic:

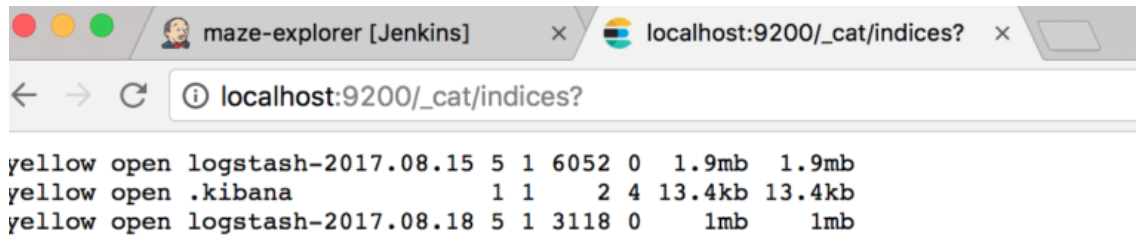
```
1 | input { beats {      port => 5044      } }
2 | output {
3 |     stdout { codec => rubydebug }
4 |     elasticsearch { hosts => ["elasticsearch:9200"] }
5 |
6 | }
```

Now you can stop the elk stack and start again, just hit Ctrl + C or run

```
1 | docker-compose -f docker-compose-elk.yml stop
```

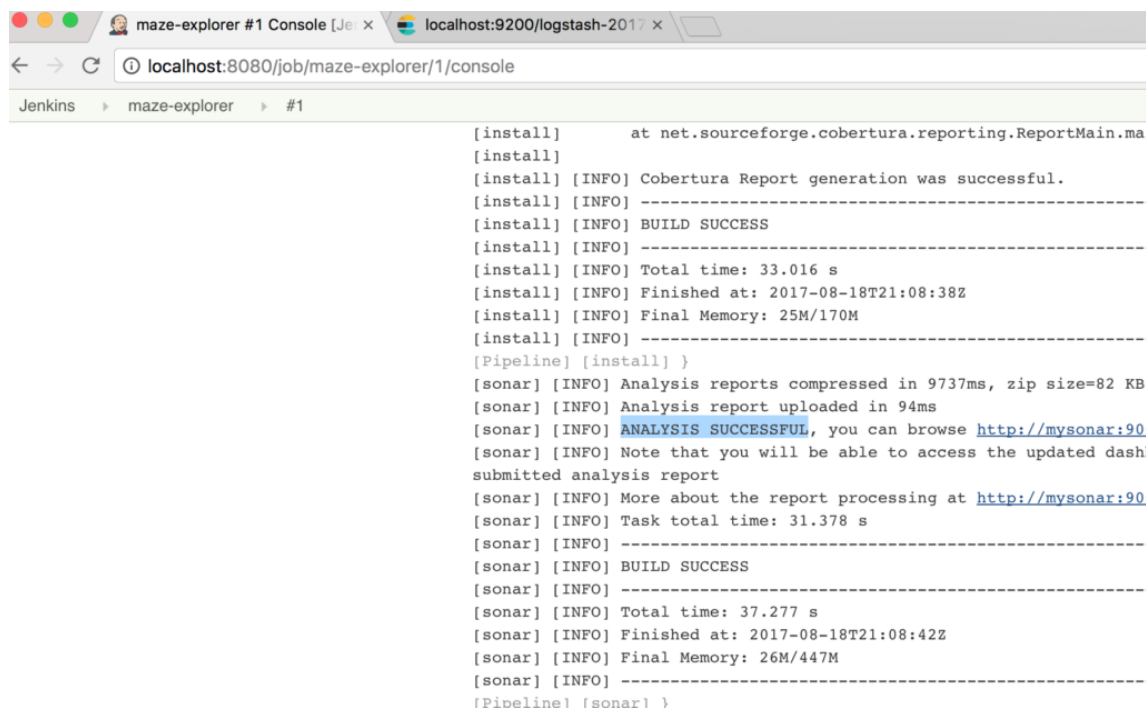
```
dockerizing-jenkins git:(master) # docker-compose -f docker-compose-elk.yml up
WARNING: Found orphan containers (dockerizingjenkins_myjenkins_1, dockerizingjenkins_mysonar_1, dockerizingjenkins_artifactory_1) for this project. If you removed or renamed this service in your docker-compose.yml you can run this command with the --remove-orphans flag to clean it up.
Starting dockerizingjenkins_elasticsearch_1 ...
Starting dockerizingjenkins_elasticsearch_1 ... done
Recreating dockerizingjenkins_logstash_1 ...
Recreating dockerizingjenkins_logstash_1 ... done
Attaching to dockerizingjenkins_elasticsearch_1, dockerizingjenkins_logstash_1
elasticsearch_1 | [2017-08-22 09:02:01,779][WARN ][bootstrap] unable to install syscall filter: seccomp unavailable: your kernel is buggy and you should upgrade
elasticsearch_1 | [2017-08-22 09:02:02,136][INFO ][node] [Ulysses] version[2.4.6], pid[1], build[5376dca/2017-07-18T12:17:44Z]
elasticsearch_1 | [2017-08-22 09:02:02,136][INFO ][node] [Ulysses] initializing ...
elasticsearch_1 | [2017-08-22 09:02:03,476][INFO ][plugins] [Ulysses] modules [reindex, lang-expression, lang-groovy], plugins [], sites []
elasticsearch_1 | [2017-08-22 09:02:03,592][INFO ][env] [Ulysses] using [1] data paths, mounts [[/usr/share/elasticsearch/data (/dev/sda1)]], net usable_space [17.5Gb],
elasticsearch_1 | [2017-08-22 09:02:03,601][INFO ][env] [Ulysses] heap size [990.7mb], compressed ordinary object pointers [true]
elasticsearch_1 | [2017-08-22 09:02:07,850][INFO ][node] [Ulysses] initialized
elasticsearch_1 | [2017-08-22 09:02:07,850][INFO ][node] [Ulysses] starting ...
elasticsearch_1 | [2017-08-22 09:02:08,286][INFO ][transport] [Ulysses] publish_address {172.21.0.5:9300}, bound_addresses {0.0.0.0:9300}
elasticsearch_1 | [2017-08-22 09:02:08,309][INFO ][discovery] [Ulysses] elasticsearch/NUeHw9SHQAKglynxqLqfng
elasticsearch_1 | [2017-08-22 09:02:11,421][INFO ][cluster.service] [Ulysses] new_master {Ulysses}[NUeHw9SHQAKglynxqLqfng]{172.21.0.5}{172.21.0.5:9300}, reason: zen-disco-join(elec
elasticsearch_1 | [2017-08-22 09:02:11,623][INFO ][http] [Ulysses] publish_address {172.21.0.5:9200}, bound_addresses {0.0.0.0:9200}
elasticsearch_1 | [2017-08-22 09:02:11,623][INFO ][node] [Ulysses] started
elasticsearch_1 | [2017-08-22 09:02:12,023][INFO ][gateway] [Ulysses] recovered [4] indices into cluster_state
elasticsearch_1 | [2017-08-22 09:02:15,203][INFO ][cluster.routing.allocation] [Ulysses] Cluster health status changed from [RED] to [YELLOW] (reason: [shards started [[kibana][0]] ...]).
logstash_1 | log4j:WARN No appenders could be found for logger (io.netty.util.internal.logging.InternalLoggerFactory).
logstash_1 | log4j:WARN Please initialize the log4j system properly.
logstash_1 | log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
logstash_1 | {:timestamp=>"2017-08-22T09:02:21.825000+0000", :message=>"Pipeline main started"}
```

You can hit the URL to check we have elasticsearch up and running:



```
yellow open logstash-2017.08.15 5 1 6052 0 1.9mb 1.9mb
yellow open .kibana 1 1 2 4 13.4kb 13.4kb
yellow open logstash-2017.08.18 5 1 3118 0 1mb 1mb
```

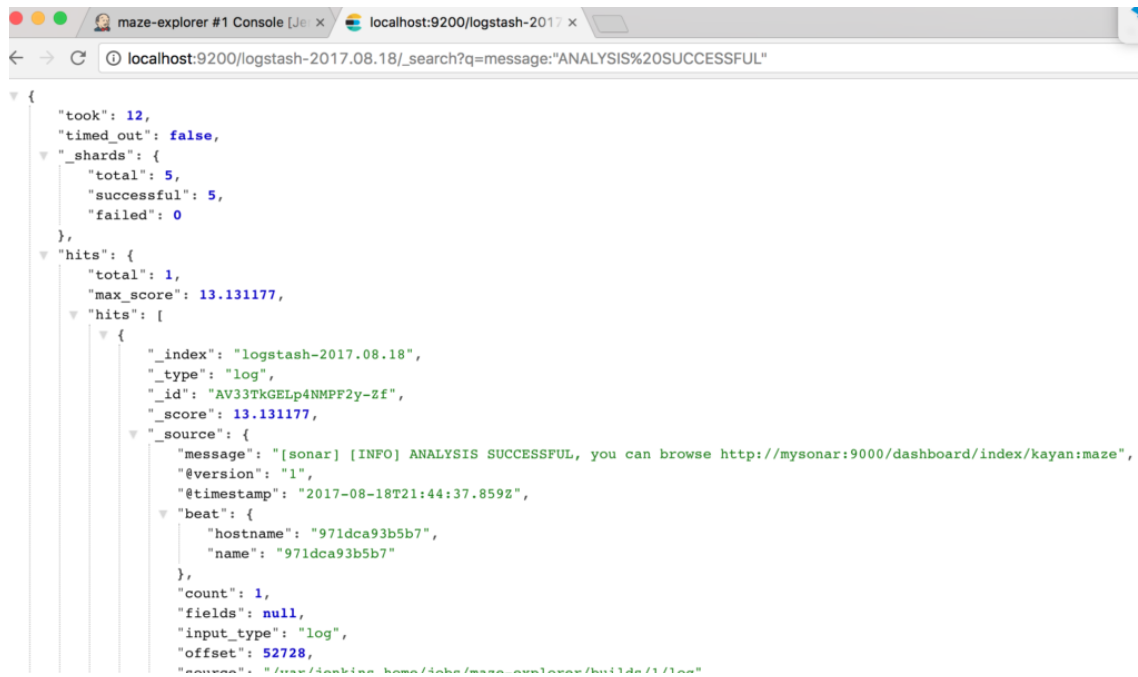
Now let's see if we can search for something. I took a random string from Jenkins logs:



```
Jenkins > maze-explorer > #1

[install]      at net.sourceforge.cobertura.reporting.ReportMain.ma
[install]
[install] [INFO] Cobertura Report generation was successful.
[install] [INFO] -----
[install] [INFO] BUILD SUCCESS
[install] [INFO] -----
[install] [INFO] Total time: 33.016 s
[install] [INFO] Finished at: 2017-08-18T21:08:38Z
[install] [INFO] Final Memory: 25M/170M
[install] [INFO] -----
[Pipeline] [install] }
[sonar] [INFO] Analysis reports compressed in 9737ms, zip size=82 KB
[sonar] [INFO] Analysis report uploaded in 94ms
[sonar] [INFO] ANALYSIS SUCCESSFUL, you can browse http://mysonar:90
[sonar] [INFO] Note that you will be able to access the updated dash
submitted analysis report
[sonar] [INFO] More about the report processing at http://mysonar:90
[sonar] [INFO] Task total time: 31.378 s
[sonar] [INFO] -----
[sonar] [INFO] BUILD SUCCESS
[sonar] [INFO] -----
[sonar] [INFO] Total time: 37.277 s
[sonar] [INFO] Finished at: 2017-08-18T21:08:42Z
[sonar] [INFO] Final Memory: 26M/447M
[sonar] [INFO] -----
[Pipeline] [sonar] }
```

and tried to search for it:



As you can see now we are able to search and find anything we want, cool, isn't it?

6. Configure and run kibana in a docker container

But it is even more cooler when you have nice UI for searching, so let's give a try to kibana now, and thus we will have our final compose file for elk stack:

```

1 | version: "3.1"
2 |
3 | services:
4 |
5 |   logstash:
6 |     image: logstash:2
7 |     volumes:
8 |       - ./:/config
9 |     command: logstash -f /config/logstash.conf
10 |    links:
11 |      - elasticsearch
12 |    depends_on:
13 |      - elasticsearch
14 |
15 |   elasticsearch:
16 |     image: elasticsearch:2
17 |     ports:
18 |       - "9200:9200"

```

```
19
20 kibana:
21   image: kibana:4
22   ports:
23     - "5601:5601"
24   links:
25     - elasticsearch
26   environment:
27     ELASTICSEARCH_URL: http://elasticsearch:9200
28   depends_on:
29     - elasticsearch
```

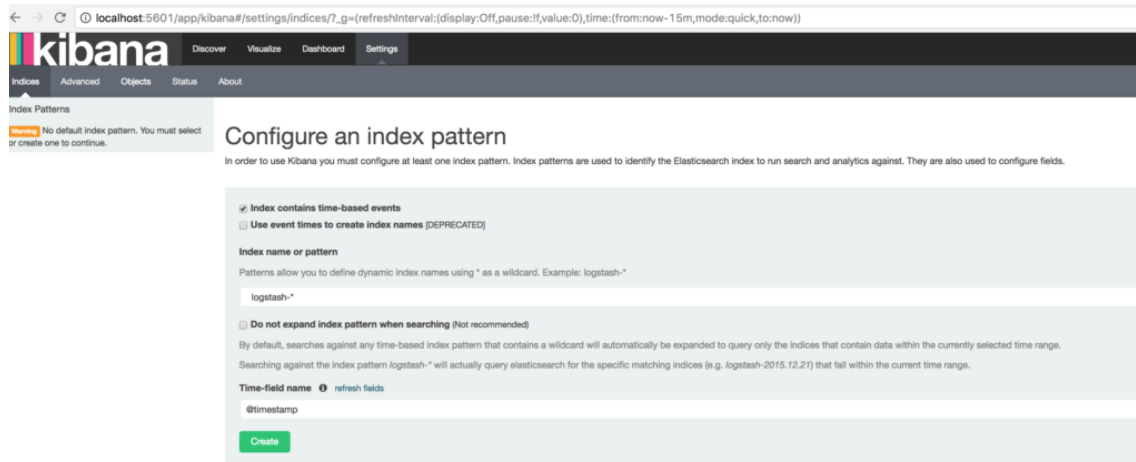
Restart the stack and check the logs:

```

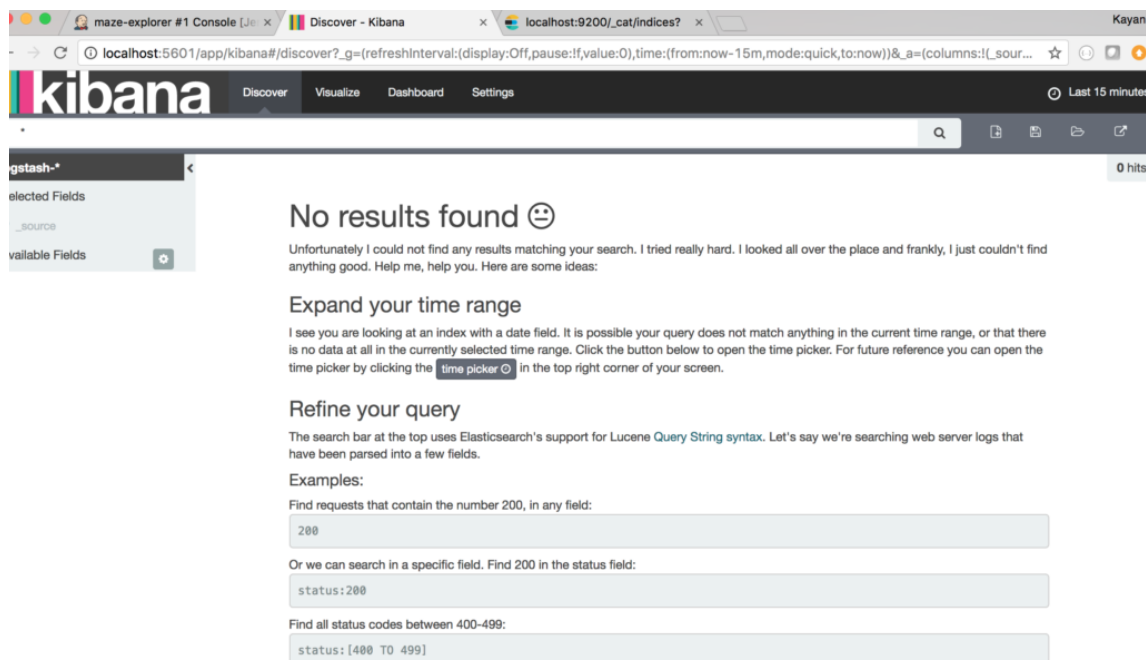
Starting dockerizingjenkins.logstash.1 ...
Starting dockerizingjenkins.logstash.1
Starting dockerizingjenkins.logstash.1 ... done
Attaching to dockerizingjenkins_elasticsearch_1, dockerizingjenkins_kibana_1, dockerizingjenkins_logstash.1
elasticsearch_1 | [2017-08-22 09:15:38,528][WARN ][bootstrap] ] Unable to install syscall filter: syscall unavailable: your kernel is buggy and you should upgrade
elasticsearch_1 | [2017-08-22 09:15:38,722][INFO ][node] ] [Gaza] version[2.4.6], pid[1], build[5376dc4/2017-07-18T12:17:44Z]
elasticsearch_1 | [2017-08-22 09:15:38,722][INFO ][node] ] [Gaza] initializing ...
elasticsearch_1 | [2017-08-22 09:15:39,330][INFO ][plugins] ] [Gaza] modules [index, lang-expression, lang-groovy], plugins [], sites []
elasticsearch_1 | [2017-08-22 09:15:39,406][INFO ][env] ] [Gaza] using [1] data paths, mounts [[/usr/share/elasticsearch/data (/dev/sda1)]], net usable space [17.4Gb], net total space [1
gh], spins? (possibly), types [ext4]
elasticsearch_1 | [2017-08-22 09:15:39,407][INFO ][jvm] ] [Gaza] heap size [990.7Mb], compressed ordinary object pointers [true]
kibana_1 | [2017-08-22 09:15:39,407][INFO ][log] ] [status]: "plugin:kibana@1.0.0", "info": "pid:11", "state": "green", "message": "Status changed from uninitialized to green - Ready", "p
state": "uninitialized", "prevMsg": "uninitialized"
kibana_1 | [2017-08-22 09:15:39,407][INFO ][log] ] [status]: "plugin:elasticsearch@1.0.0", "info": "pid:11", "state": "yellow", "message": "Status changed from uninitialized to yellow - R
state": "uninitialized", "prevMsg": "uninitialized"
kibana_1 | [2017-08-22 09:15:39,407][INFO ][log] ] [error]: "elasticsearch", "pid:11", "message": "Request error, retrying -- connect ECONREFUSED 172.21.0.5:9200"
kibana_1 | [2017-08-22 09:15:39,407][INFO ][log] ] [warning]: "elasticsearch", "pid:11", "message": "Unable to revive connection: http://elasticsearch:9200/"
kibana_1 | [2017-08-22 09:15:39,407][INFO ][log] ] [warning]: "elasticsearch", "pid:11", "message": "No living connections"
kibana_1 | [2017-08-22 09:15:39,407][INFO ][log] ] [status]: "plugin:elasticsearch@1.0.0", "error": "pid:11", "state": "red", "message": "Status changed from yellow to red - Unable to con
state": "uninitialized", "prevState": "yellow", "prevMsg": "Waiting for Elasticsearch"
kibana_1 | [2017-08-22 09:15:39,407][INFO ][log] ] [status]: "plugin:kbn-vis@1.0.0", "info": "pid:11", "state": "green", "message": "Status changed from uninitialized to green
state": "uninitialized", "prevState": "uninitialized"
kibana_1 | [2017-08-22 09:15:39,407][INFO ][log] ] [status]: "plugin:metric-vis@1.0.0", "info": "pid:11", "state": "green", "message": "Status changed from uninitialized to green - Ready
state": "uninitialized", "prevMsg": "uninitialized"
kibana_1 | [2017-08-22 09:15:39,407][INFO ][log] ] [status]: "plugin:spk@1.0.0", "info": "pid:11", "state": "green", "message": "Status changed from uninitialized to green - Ready
state": "uninitialized", "prevMsg": "uninitialized"
kibana_1 | [2017-08-22 09:15:39,407][INFO ][log] ] [status]: "plugin:statusPage@1.0.0", "info": "pid:11", "state": "green", "message": "Status changed from uninitialized to green - Ready
state": "uninitialized", "prevMsg": "uninitialized"
kibana_1 | [2017-08-22 09:15:39,407][INFO ][log] ] [status]: "plugin:table-vis@1.0.0", "info": "pid:11", "state": "green", "message": "Status changed from uninitialized to green - Ready
state": "uninitialized", "prevMsg": "uninitialized"
kibana_1 | [2017-08-22 09:15:39,407][INFO ][log] ] [listening]: "info", "pid:11", "message": "Server running at http://0.0.0.0:5601"
elasticsearch_1 | [2017-08-22 09:15:42,286][INFO ][node] ] [Gaza] initialized
elasticsearch_1 | [2017-08-22 09:15:42,286][INFO ][node] ] [Gaza] starting ...
elasticsearch_1 | [2017-08-22 09:15:42,286][INFO ][transport] ] [Gaza] publish address [172.21.0.5:9300], bound_addresses [0.0.0.0:9300]
elasticsearch_1 | [2017-08-22 09:15:42,286][INFO ][discovery] ] [Gaza] elasticsearch/calcExpRk2HqMqADxQ [172.21.0.5]
kibana_1 | [2017-08-22 09:15:42,286][INFO ][log] ] [warning]: "elasticsearch", "pid:11", "message": "Unable to revive connection: http://elasticsearch:9200/"
kibana_1 | [2017-08-22 09:15:42,286][INFO ][log] ] [warning]: "elasticsearch", "pid:11", "message": "No living connections"
logstash.1 | log4j:WARN No appenders could be found for logger (io.netty.util.internal.logging.InternalLoggerFactory).
logstash.1 | log4j:WARN Please initialize the log4j system properly.
logstash.1 | log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
elasticsearch_1 | [2017-08-22 09:15:45,560][INFO ][cluster.service] ] [Gaza] new master [Gaza][calcExpRk2HqMqADxQ][172.21.0.5][172.21.0.5:9300], reason: zen-disco-join(elected_as_master, [0] jo
received)
elasticsearch_1 | [2017-08-22 09:15:45,635][INFO ][http] ] [Gaza] publish address [172.21.0.5:9200], bound_addresses [0.0.0.0:9200]
elasticsearch_1 | [2017-08-22 09:15:45,642][INFO ][node] ] [Gaza] started
elasticsearch_1 | [2017-08-22 09:15:45,839][INFO ][gateway] ] [Gaza] recovered [5] indices into cluster-state
kibana_1 | [2017-08-22 09:15:45,839][INFO ][log] ] [status]: "plugin:elasticsearch@1.0.0", "error": "pid:11", "state": "red", "message": "Status changed from red to red - Elasticsearch is
state": "uninitialized", "prevState": "red", "prevMsg": "Unable to connect to Elasticsearch at http://elasticsearch:9200/"
kibana_1 | [2017-08-22 09:15:45,839][INFO ][log] ] [status]: "plugin:elasticsearch@1.0.0", "error": "pid:11", "state": "red", "message": "Unable to connect to Elasticsearch at http://elasticsearch:9200/"
elasticsearch_1 | [2017-08-22 09:15:47,333][INFO ][cluster.routing.allocation] ] [Gaza] Cluster health status changed from [RED] to [YELLOW] (reason: [shards started [[kibana][0]] ...]).
kibana_1 | [2017-08-22 09:15:47,333][INFO ][log] ] [status]: "plugin:elasticsearch@1.0.0", "info": "pid:11", "state": "green", "message": "Status changed from red to green - Kibana index
state": "uninitialized", "prevState": "red", "prevMsg": "Elasticsearch is still initializing the kibana index"

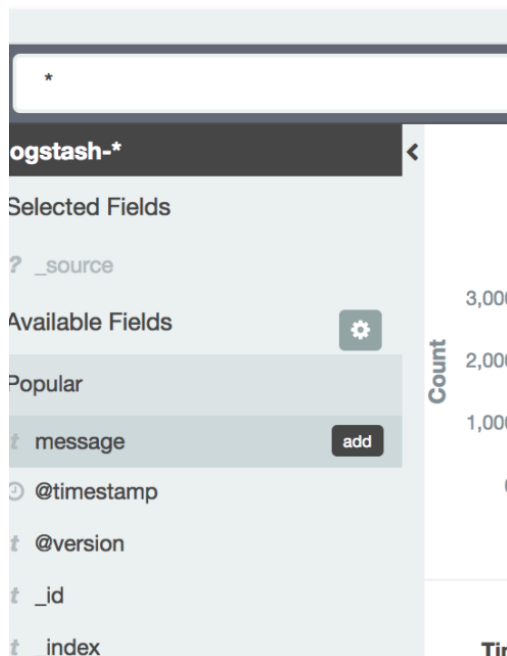
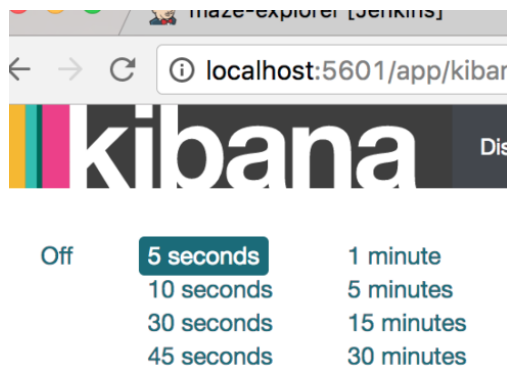
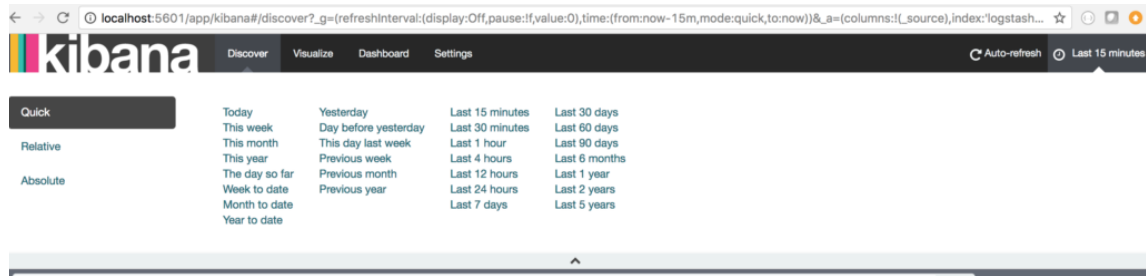
```

As we see after couple tries kibana managed to connect to elastic.
Now go to <http://localhost:5601>, that is where you find kibana, you should see this screen:

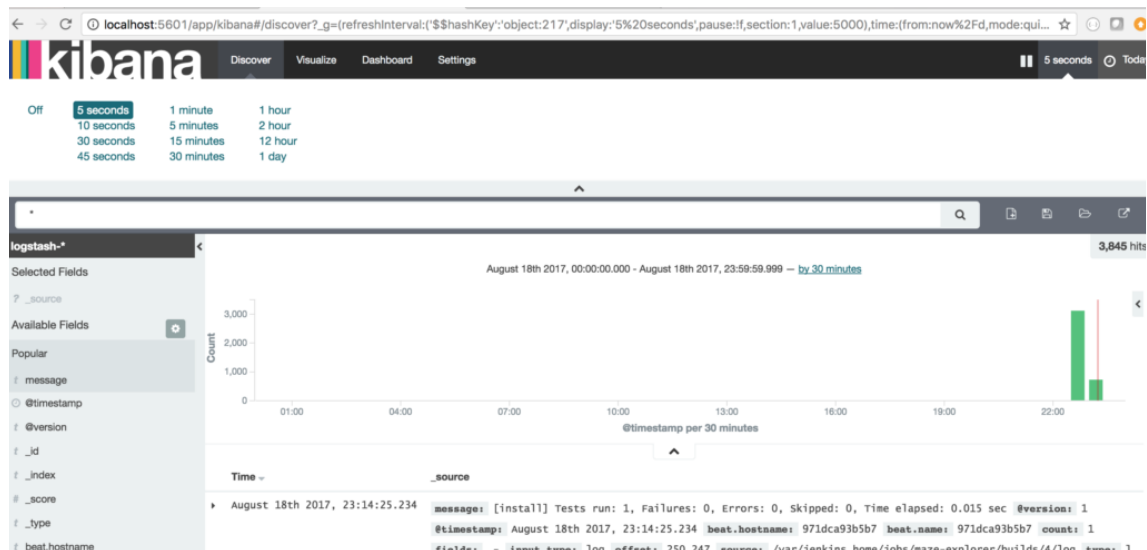


You will need to hit the “create” button to create the first index pattern. As the logs come from logstash, the index will have the pattern “logstash-*” where “*” will be any date like “logstash-2017.08.22”. Once done kibana will pick up your logs and you can hit “Discover” to see your logs. Afterwards, we can do some config changes to get live logs, first change time to today, then change auto-refresh to 5 seconds:

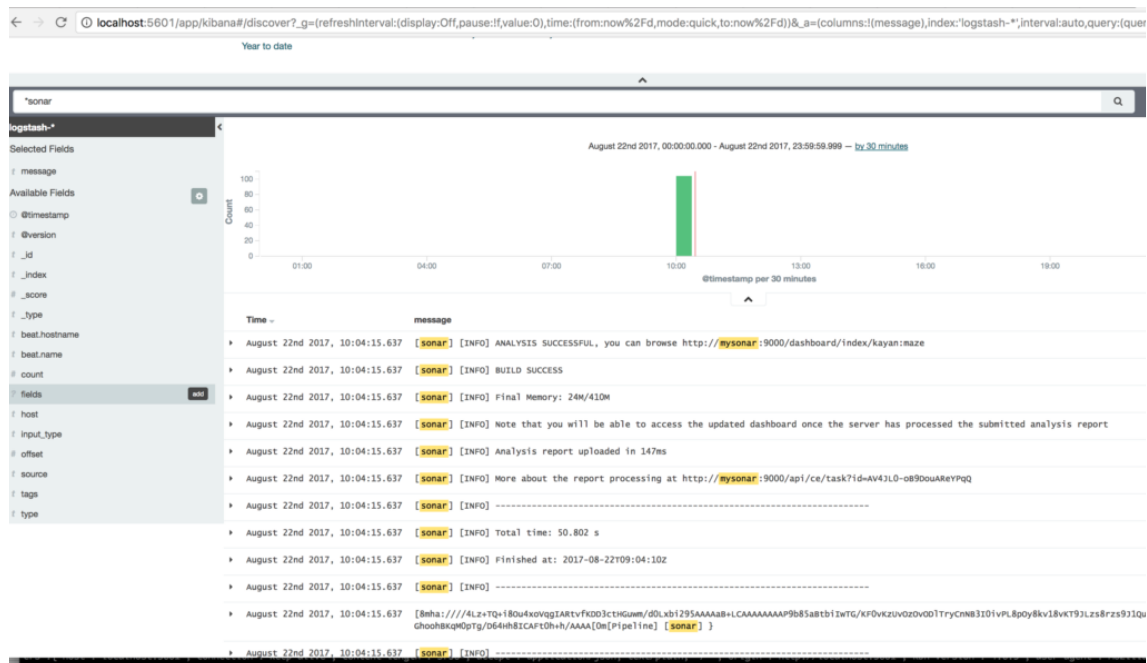




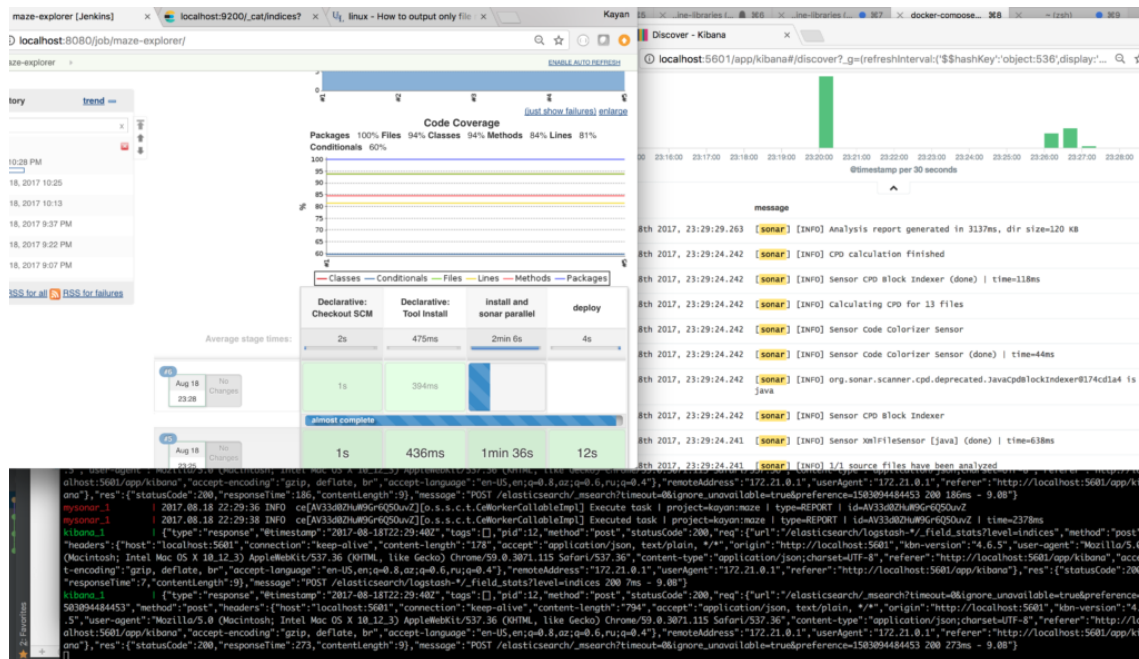
You can also press “add” on message to have only log messages shown.



Now let's search for "sonar" string with kibana, I assume you ran build many times so you have logs for "sonar":



As we got 6 containers running, you may wonder how much memory and cpu they consume, let's run some stats when pipeline is running, especially during sonar stage, given we do it in parallel:



```
1 | docker stats $(docker ps --format {{.Names}})
```

CONTAINER	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
dockerizingjenkins_myjenkins_1	11.03%	1.29GiB / 6.80GiB	19.01%	87.1MB / 1.98MB	20.5kB / 72.7MB	97
dockerizingjenkins_kibana_1	0.02%	64.47MiB / 6.80GiB	0.93%	35.7MB / 2.97MB	0B / 8.19kB	11
dockerizingjenkins_logstash_1	0.68%	312.3MiB / 6.80GiB	4.48%	1.09MB / 3.41MB	0B / 2.02MB	34
dockerizingjenkins_elasticsearch_1	3.94%	415.7MiB / 6.80GiB	5.97%	4.63MB / 35.8MB	0B / 5.84MB	76
dockerizingjenkins_mysonar_1	2.36%	1.1GiB / 6.80GiB	16.17%	208kB / 63.5MB	119kB / 17.3MB	180
dockerizingjenkins_artifactory_1	0.27%	691.9MiB / 6.80GiB	9.93%	7.66kB / 1.05kB	0B / 868kB	65

That is it, we have completed our mission and running two stacks, one for our build pipeline and second for log management.

Finally, if you didn't follow the instruction but still want to have all up and running by magic command, just run this:

```
1 | git clone https://github.com/kenych/dockerizing-jenkins
2 | cd dockerizing-jenkins && \
3 | git checkout dockerizing_jenkins_part_4_elk_stack &
4 | ./runall.sh
```

Published in **Docker**, **ELK**, **Jenkins** and **Log Management**

[docker](#)

[elasticsearch](#)

[ELK](#)

[filebeat](#)

[jenkins](#)

[kibana](#)

[logstash](#)

Previous Post

**Dockerizing Jenkins, part 3:
Securing password with docker-
compose, docker-secret and
jenkins credentials plugin**

Next Post

**Running Ansible as Docker
container**



Foundations of Artificial Intelligence and Machine Learning

15 Weekends | Starting Jan 5

**Special Scholarships
for Women**

Batch 1 Full. Batch 2 Now Open.



APPLY NOW

In Association with TalentSpr

Recent Posts

Installing Kubernetes on MacOS
November 28, 2017

Creating Kubernetes Jobs.
November 27, 2017

Running smallest test http server container
November 26, 2017

Setting up Firewall and network troubleshooting in Linux with UFW, Isof, tcpdump, wireshark, rsyslog and vagrant

November 25, 2017

Implementing Service Discovery with Consul, Registrator and Nginx in a Dockerized environment.

November 3, 2017

/var/run/docker.sock

October 29, 2017

Running Ansible as Docker container

October 20, 2017

Dockerizing Jenkins build logs with ELK stack (Filebeat, Elasticsearch, Logstash and Kibana)

August 22, 2017

Dockerizing Jenkins, part 3: Securing password with docker-compose, docker-secret and jenkins credentials plugin

August 12, 2017

Dockerizing Jenkins 2, part 2: Deployment with maven and JFrog Artifactory

July 23, 2017

Dockerizing Jenkins 2, Part 1: Declarative Build Pipeline With SonarQube Analysis

July 6, 2017

How to trigger a build in Jenkins when adding a comment in Gerrit with JobDSL

June 21, 2017

Archives

November 2017

October 2017

August 2017

July 2017

June 2017

Categories

ansible (1)

Artifactory (1)

Build Pipelines (2)

Consul (1)

Credentials (1)

deployment (1)

Docker (9)

Docker Compose (1)

ELK (1)

Firewall (1)

Gerrit (1)

Java (1)

Jenkins (4)

k8s (3)

Log Management (1)

Maven (2)

nc (1)

Node.js (1)

Secutiry (1)

Service Discovery (1)

SonarQube (1)

SSH (1)

TCP (1)

Wireshark (1)

Meta

Log in

Entries [RSS](#)

Comments [RSS](#)

[WordPress.org](#)

Recent Comments

Apex WordPress Theme by Compete Themes