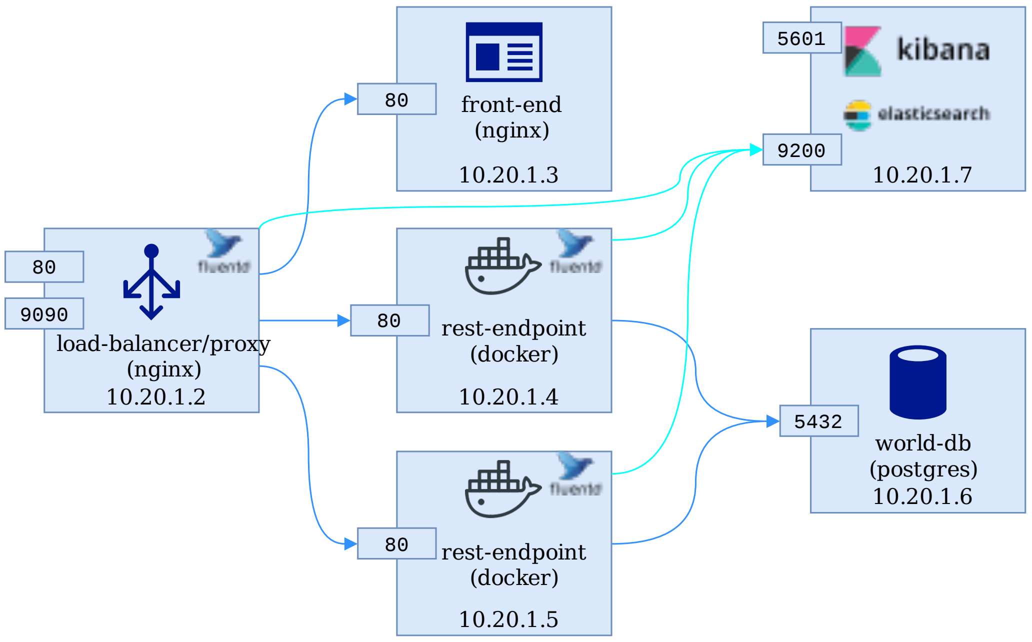
Lab 10

Define a custom Role

During this step you will setup an EFK logging infrastructure (ES/Elastic Search, fluentd and Kibana). We will finally get to use `machine-7` as a central point for your logs (it will host ES and Kibana)



You will define a role for fluentd but before that you'll get to setup the kibana and elastic search.

This particular task of setting up ES/Kibana does not have much to do with the current topic of defining roles. Nonetheless it will probably be a good idea to go through it and apply your newly gained skills of using galaxy roles and defining ansible playbooks instead of us just giving you the playbooks.

Open `~/course/ansible/custom-role` . The file to provision your ES/Kibana services is `admin.yaml` . Open it and notice we provided you with the bare minimum.

ES

To install ES, we will apply the official elastic role: elastic.elasticsearch (https://github.com/elastic/ansible-elasticsearch). We invite you to have a thorough look at the documentation, but we will provide you with the solutions for the role configuration. Before you continue, you might want to make a mental image of the solution you would apply. The solutions are provided as hint, so try if you want.

- Make sure the role is applied to our **admin** hosts (which is only a single machine)
- Then for the vars:
  - name the instance anything you want, for example `course` `es_instance_name: course`
  - **important** do set the java heap size lower. We are running on a single machines with lots of virtual ones. By default ES takes 2GB of Java heap space, which we won't need during this course. Set the heap size to 512m `es_heap_size: "512m"`
  - **important** we don't want to enable security to the ES api endpoint. Therefore we need to remove

the security feature from the default set specified by `es_xpack_features`. Set its value to the remaining ones (we in fact don't need most of these for this lab): `["alerting","monitoring","graph","ml"]`
  - The `es_config` uses a direct mapping to the ES confutation of a node. Please use the following to configure the ES server further

```
es_config:
  node.name: "course"
  cluster.name: "custom-cluster"
  discovery.zen.ping.unicast.hosts: "localhost:9301"
  transport.tcp.port: 9301
  http.port: 9200
  network.host: 0.0.0.0
```

Kibana

For Kibana we won't use a role, but have you install it using yum. We have provided you with the configuration (as this is not a kibana course)

The installation steps are detailed here: https://www.elastic.co/guide/en/kibana/current/rpm.html (https://www.elastic.co/guide/en/kibana/current/rpm.html). You probably want to start at Installing from the RPM repository (https://www.elastic.co/guide/en/kibana/current/rpm.html#rpm-repo):

- install the yum repository
- We have provided you with the `files/kibana.yaml`, make sure this gets copied to `/etc/kibana/kibana.yml` (have a look at it!)
- ensure the kibana service is enabled and started

Let's now add the fluentd agents to our machine to talk to our cluster.

Open a terminal session in `custom-role/roles`. This is were you will create your **fluentd** role:

- Use the CLI to define the role directory structure `ansible-galaxy init fluentd`

Open the `fluentd/main.yaml`. We won't be placing a lot of tasks here, but instead we'll include different files with tasks in them. We will also "prepare" this role to be multi-distro (we will however only create/test the role for CentOS and other yum based systems)

Fluentd prerequisites

Have a look at docs.fluentd.org (https://docs.fluentd.org/v0.12/articles/before-install) for the installation prerequisites. We will guide you through them below.

Start by creating a new file named `fluentd/tasks/prereqs.yml` and then include this inside your `main.yml` `- import_tasks: …` (we are using `yml` as the suffix here, as that was generated, feel free to change it to `yaml`)

Then make sure the following perquisites are met:

- Ensure Network Kernel Parameters are optimised
  - we need the following entries in `sysctl.conf`:

```
net.core.somaxconn = 1024
net.core.netdev_max_backlog = 5000
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.ipv4.tcp_wmem = 4096 12582912 16777216
net.ipv4.tcp_rmem = 4096 12582912 16777216
net.ipv4.tcp_max_syn_backlog = 8096
net.ipv4.tcp_slow_start_after_idle = 0
net.ipv4.tcp_tw_reuse = 1
net.ipv4.ip_local_port_range = 10240 65535
```

  - Use another system module for this `sysctl`
  - and you probably want to use a loop again
- Ensure the maximum number of file descriptors is increased:
  - We need to have the following configuration

    ```
    root soft nofile 65536
    root hard nofile 65536
    * soft nofile 65536
    * hard nofile 65536
    ```

  - Search for a system module (https://docs.ansible.com/ansible/latest/modules/list_of_system_modules.html) to modify the PAM limits `pam_limits`
  - You probably want to use a loop to ensure these limits are set

Now just to feel a bit more like you are creating a real role, let's make the limit value configurable for users of your role, but wil a default value of 65536:

- Use the correct file to define default values for your variables `fluentd/defaults/main.yml`
- Define a variable named `ulimit` with a value of `65536` (this is alreade a var file, so no need to add `vars` or something like that) `ulimit: 65536`
- Then use this variable inside your file descriptors task

Installation (yum based)

For this one, create a new file named `fluentd/tasks/install-with-yum.yml` (yes you can say "yum-yaml ;)")

Then import this file, but only when the package manager is `yum` `when` `ansible_pkg_mgr` `when: ansible_pkg_mgr == "yum"`

Then inside your `install-with-yum.yml` :

- register the `TreasureData` yum repository
  - the `baseurl` is `http://packages.treasuredata.com/2.5/redhat/\$releasever/\$basearch`
  - the `gpgkey` is https://packages.treasuredata.com/GPG-KEY-td-agent (https://packages.treasuredata.com/GPG-KEY-td-agent)
  - don't forget to set other fields for this to work
- Then install the `td-agent` package

Configure td-agent

Similar as before, create a file name `fluentd/tasks/configuration.yml` and include it inside your `main.yml` .

Let's make it possible for people using your role to add additional configuration by creating a directory named `/etc/td-agent/conf.d/` `file` module (make sure the owner/group is set to `td-agent`)

If you want to use some fluentd reference documentation during the next step, then open https://docs.fluentd.org/v0.12/articles/config-file (https://docs.fluentd.org/v0.12/articles/config-file)

Create a template for the `/etc/td-agent/td-agent.conf` :

- Create `td-agent.conf.j2` in the correct directory `fluentd/templates`
- Open a forwarding source port 24224

  ```
  <source>
    @type forward
    port 24224
  </source>
  ```

- Open a http source port on 9880 (even though we won't be using it)

```
<source>
  @type http
  port 9880
</source>
```

So far you have enabled machines with docker to log to fluentd (you will update this later, but docker can directly access the source port 24224)

It is possible to use a nginx module that also directly accesses the fluentd daemon. However we want to create this role so people can use if to send the tail of any log file.

We want people to add host configuration such as:

```
files:
  - path: '/var/log/nginx/access.log'
    type: 'nginx'
```

For that reason, introduce another default variable named `tail_files` with a default value of an empty array `[]` (they can then assign the host configuration we showed above, in fact you will do this yourself when using the role later on)

Then inside your template `td-agent.conf.j2` , iterate over this list and for each add the following configuration

```
<source>
  @type tail
  path …
  pos_file /var/log/td-agent/{{tail_cfg.path| replace("/", "-") }}.pos
  tag …
  format …
</source>
```

Replace the `path` and `format` dots with the appropriate parameter values `{{tail_cfg.path}}` and `{{tail_cfg.type}}` resp. and for the `tag` use the same vale as the `format`

Then send the log entries to our elasticsearch cluster:

```
<match **>
  @type elasticsearch
  logstash_format true
  host {{es.host}}
  port {{es.port}}
  flush_interval 5s
</match>
```

Notice we are using two variables, add these to your default variables with the following values:

```
es:
  host: 127.0.0.1
  port: 9200
```

As a last step include any `/etc/td-agent/conf.d/*.conf` file that you have facilitated:

```
@include /etc/td-agent/conf.d/*.conf
```

Then apply your template inside your `fluentd/tasks/configuration.yml`

- use the correct module `template`
- set your source
- the destination is `/etc/td-agent/td-agent.conf`
- backup the file (or if you like make it configurable for role users)
- set the owner and group to `td-agent`
- trigger a handler you will that listens to `RELOAD_TD_AGENT`

Where do you need to define this trigger? `fluentd/handlers/main.yml` . Add a handler to set the state of the `td-agent` to `reloaded` .

We are allowing role users to define files that we will have tailed by fluentd. Therefore these files need to be readable to fluentd. Therefore *for each* defined file in `tail_files` :

- Set the mode to 0644 `file` module

Install ruby

In our EFK we need fluentd to talk to out Elasticsearch cluster (you've already setup). This is achieved through a plugin fluent-plugin-elasticsearch (https://github.com/uken/fluent-plugin-elasticsearch). To install and use this, we need to install ruby. We in fact need a recent version of ruby, more recent that available in the CentOS repositories. We have to in fact install Ruby from it sources. However that's a lot of work, so ... yes we look for a role!

We will use the one from Jeff Geerling geerlingguy.ruby (https://github.com/geerlingguy/ansible-role-ruby)

```
$ ansible-galaxy install geerlingguy.ruby
```

Now wait a minute. We are defining a role which requires another role. How do we solve this?

- Define our dependency in the correct file `fluentd/meta/main.yml`
- If you look at the README (https://github.com/geerlingguy/ansible-role-ruby) you'll notice we need to define some variables in order to get the latest version (`ruby_install_from_source` and `ruby_version`). These role variables needs to be supplied in the old format within this file. Define your dependency and make sure rube 2.5.1 gets installed.

```
dependencies:
     - {role: geerlingguy.ruby, ruby_install_from_source: true, ruby_vers
ion: 2.5.1}
```

- While you are in this file you might want to supply some additional information

```
galaxy_info:
  author: Student
  description: Simple fluentd role

  license: license (GPLv2, CC-BY, etc)

  min_ansible_version: 1.2
  platforms:
  - name: CentOS
    versions:
    - 7
```

Install the ES Plugin

Good we can now install our ES plugin inside `fluentd/tasks/configuration.yml` . Use the correct module to execute the following command:

```
td-agent-gem install fluent-plugin-elasticsearch
```

And make sure `td-agent` is reloaded.

Start the agent

As a final step, make the td-agent is enabled and started. Just place this inside `fluentd/tasks/main.yaml`

This completes creating the role. You are now ready to use it.

Navigate to the `~/course/ansible/custom-role` directory. You should find a file named `logging.yaml` . We have provided you already with:

- An empty play because we need to access the **admin** server(S)
- the hosts to which we will add fluentd configuration (for now that's just the load-balancer and the docker rest-servers)

You are faced with a problem. The load-balancing machine(s) require to tail files, whereas the docker rest-server's don't. Even if you would add the postgres machine the location to the log files would be different. How can you add different configuration for the fails to tail or the lack thereof in case of docker? `group_vars`

- Define the following for the load-balancers `group_vars/load-balancer.yaml` :

```
files:
  - path: '/var/log/nginx/access.log'
    type: 'nginx'
```

For docker as you know there are no files to tail, for docker we need to configure it different (coming up later)

Ok with the host-specific configuration in place, let's apply the role to our machines inside `logging.yaml` :

- Make sure the role is imported
- Then for the vars:
    - use the possible `files` group variable for `tail_files`
    - you need to make sure this defaults to an empty array in case the host does not have this value set use the filter `default ([])`
    - the `es` configuration needs to point to your `machine-7` through the inventory alias `master-admin-server` (you've done something similar for the rest-servers to connect to the worlddb)

    ```
    es:
      host: "{{hostvars['master-admin-server']['ansible_eth1']['ipv4']
    ['address']}}"
      port: 9200
    ```

    - make sure the role tasks' are run as root `become`

For docker we need to tell the containers to use the fluentd agent you are making available. This is explained in the next task.

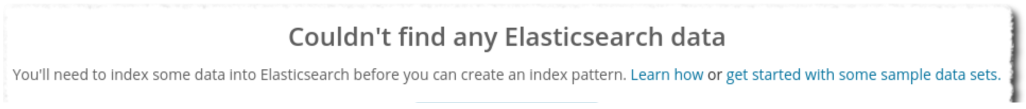
This is a small step, but before you continue, you will need to make sure the docker containers connect with your fluent agent. This is accomplished by changing the logging driver of your containers from the standard (json) to **fluentd** (see docs.docker.com (https://docs.docker.com/config/containers/logging/fluentd/)).

Open the `custom-role/rest-server.yaml` file and set the `log_driver` to `fluentd` .

After that run your `rest-server.yaml` playbook.

You are good to go. Run your playbook for the admin machine. This might take a while. Eventually you should be able to go to the kibana console at http://10.20.1.7:5601 (http://10.20.1.7:5601)

Once the kibana console is available, click on **discover** you might see the following message (if you din't than data is available)

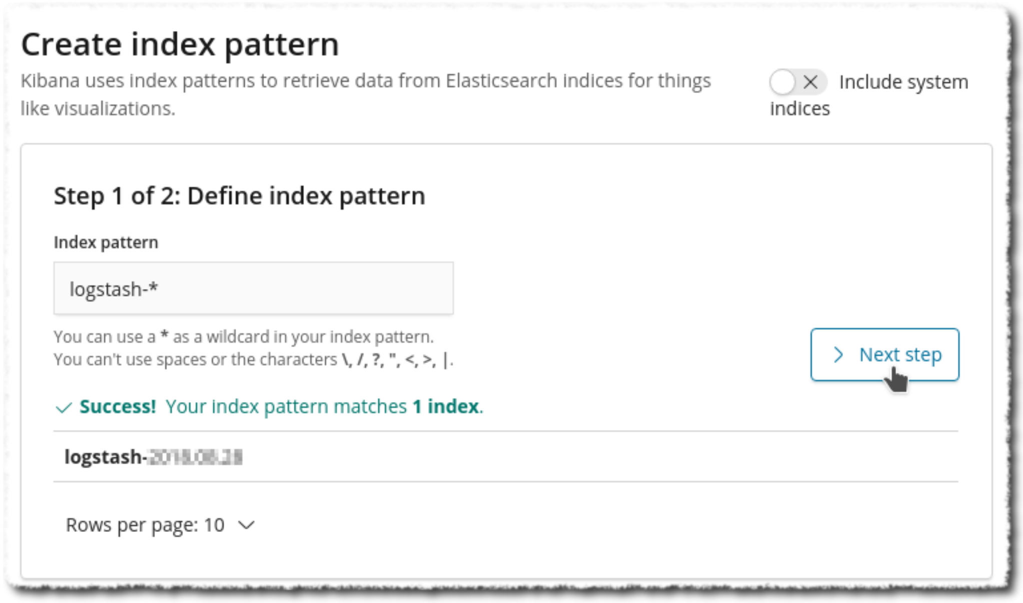

Couldn't find any Elasticsearch data

You'll need to index some data into Elasticsearch before you can create an index pattern. Learn how or get started with some sample data sets.

In that case there's no data available to configure kibana index patterns. Kibana is easier to configure with some data available. If you saw this message, just access your web application on http://10.20.1.2 (http://10.20.1.2) and refresh a couple of time to create some data. and after some time data will be available (click **check for new data**)

Once you see a **Create index pattern** page, we can continue.

Set the index pattern to `logstash-*` and click **Next Step**:



Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

Include system indices

Step 1 of 2: Define index pattern

Index pattern

logstash-*

You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

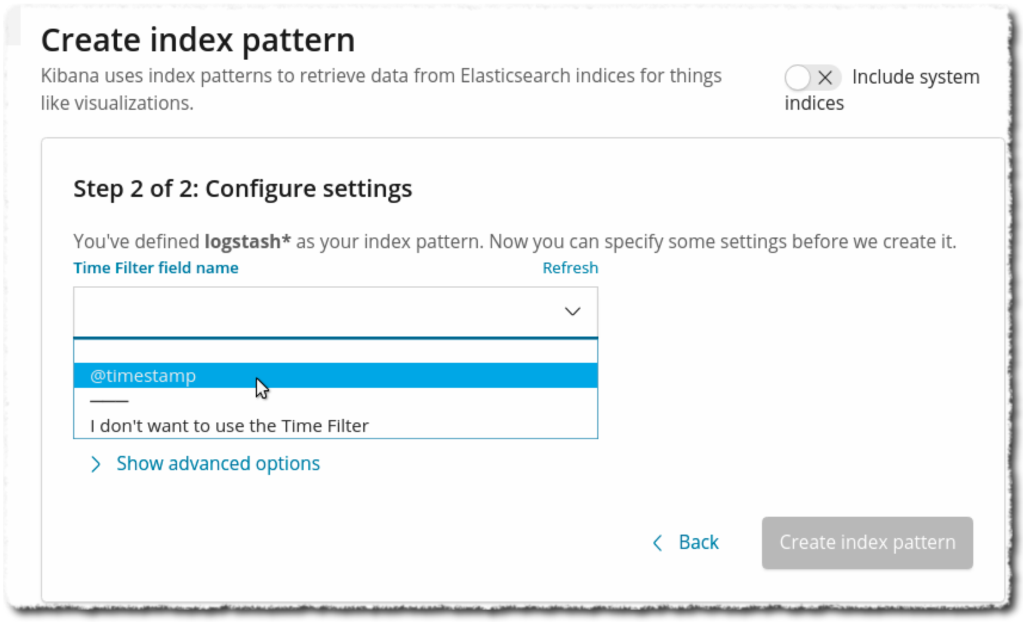
Next step

Success! Your index pattern matches 1 index.

logstash-

Rows per page: 10

On **Step 2 of 2** choose **@timestamp**



Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

Include system indices

Step 2 of 2: Configure settings

You've defined **logstash*** as your index pattern. Now you can specify some settings before we create it.

Time Filter field name Refresh

@timestamp

I don't want to use the Time Filter

Show advanced options

Back Create index pattern

Click on **Create index pattern** to complete the wizard.

After this click again on **Discover** in the sidebar menu. You should see your logs!

Now you might want to do things like:

- make requests to the world front-end (http://10.20.1.2 (http://10.20.1.2))
- restart the docker container (via ssh access to 10.20.1.4 and 10.20.1.5)

You should see messages appearing

This concludes this lab. Feel free to try the extra(s)

Feel free to add our worlddb postgres database to this infrastructure using the tailing of files.