Title: How do I create a (dockerized) Elasticsearch index using a python script running in a docker container?

Post Body:

I'm trying to index a containerized Elasticsearch db using the Python client https://github.com/elastic/elasticsearch-py called from a script (running in a container too).

By looking at existing pieces of code, it seems that `docker-compose` is a useful tool to use for my purpose. My dir structure is

```
docker-compose.yml indexer/ - Dockerfile - indexer.py - requirements.txt elasticsearch/ - Dockerfile
```

My `docker-compose.yml` reads

```
version: '3'  services:   elasticsearch:     build: elasticsearch/     ports:        - 9200:9200     networks:       - deploy_
```

`indexer.py` reads

```
from elasticsearch import Elasticsearch from elasticsearch.helpers import bulk      es = Elasticsearch(hosts=[{"host":'elastic
```

The Dockerfile for the elasticsearch service is

```
FROM docker.elastic.co/elasticsearch/elasticsearch-oss:6.1.3 EXPOSE 9200 EXPOSE 9300
```

and that for the indexer is

```
FROM python:3.6-slim WORKDIR /app ADD . /app RUN pip install -r requirements.txt ENTRYPOINT [ "python" ] CMD [ "indexer.py" ]
```

with `requirements.txt` containing only `elasticsearch` to be downloaded with pip.

Running with `docker-compose run indexer` gives me the error message at https://pastebin.com/6U8maxGX (`ConnectionRefusedError: [Errno 111] Connection refused`). elasticsearch is up as far as I can see with `curl -XGET 'http://localhost:9200/'` or by running `docker ps -a`.

How can I modify my `docker-compose.yml` or `indexer.py` to solve the problem?

P.S. A (working) version (informed by the answers below) of the code can be found here, for completeness' sake: https://github.com/davidefiocco/dockerized-elasticsearch-indexer.

Accepted Answer:

The issue is a synchronisation bug: `elasticsearch` hasn't fully started when `indexer` tries to connect to it. You'll have to add some retry logic which makes sure that `elasticsearch` is up and running before you try to run queries against it. Something like running `es.ping()` in a loop until it succeeds with an exponential backoff should do the trick.

UPDATE: The Docker `HEALTHCHECK` instruction can be used to achieve a similar result (i.e. make sure that `elasticsearch` is up and running before trying to run queries against it).

Highest Rated Answer:

Making more explicit @Mihai_Todor update, we could use `HEALTHCHECK` (docker 1.12+), for instance with a command like:

```
curl -fsSL 'http://$(hostname --ip-address):9200/_cat/health?h=status' | grep -E '^green'
```

To answer this question using using `HEALTHCHECK`:

```
FROM python:3.6-slim  WORKDIR /app ADD . /app RUN pip install -r requirements.txt  HEALTHCHECK CMD curl -fsSL 'http://$(hostna
```