

DZone (/) > Big Data Zone (/big-data-analytics-tutorials-tools-news) > Deploying Kafka With the ELK Stack

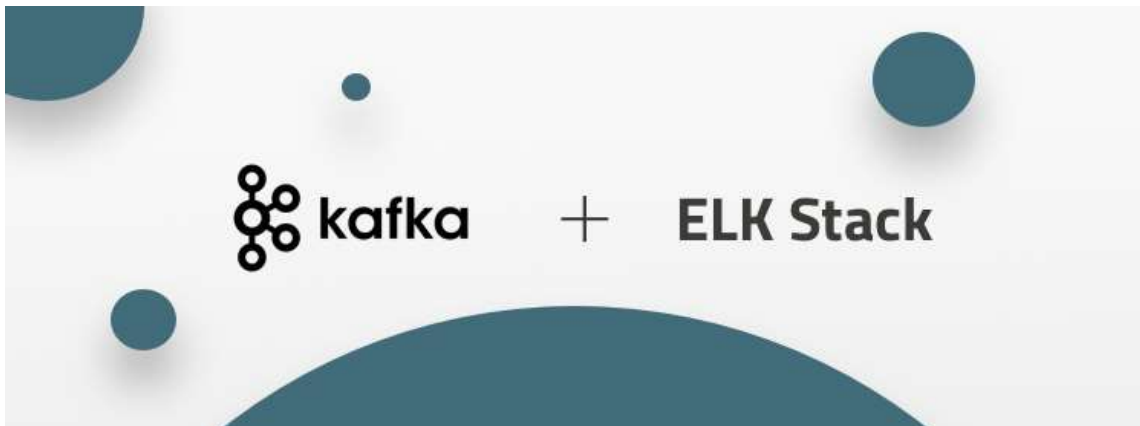
Deploying Kafka With the ELK Stack



by Daniel Berman (/users/2757213/proudboffin.html)  MVB · Jun. 07, 19 · Big Data Zone (/big-data-analytics-tutorials-tools-news) · Tutorial

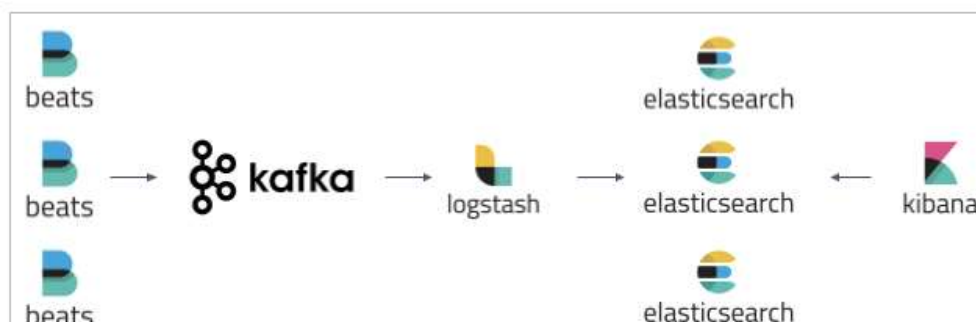
 Like (21)  Comment (4)  Save  Tweet

Setting up an observability platform across your teams? This technical cheat sheet covers what is needed to increase both speed to market and agile manage your infrastructure.



Following a production incident, and precisely when you need them the most, logs can suddenly surge and overwhelm your logging infrastructure. To protect Logstash and Elasticsearch against such data bursts, users deploy buffering mechanisms to act as message brokers.

Apache Kafka is the most common broker solution deployed together the ELK Stack. Usually, Kafka is deployed between the shipper and the indexer, acting as an entrypoint for the data being collected:



- Filebeat - collects logs and forwards them to a Kafka topic.
- Kafka - brokers the data flow and queues it.
- Logstash - aggregates the data from the Kafka topic, processes it and ships to Elasticsearch.
- Elasticsearch - indexes the data.
- Kibana - for analyzing the data.

My Environment

To perform the steps below, I set up a single Ubuntu 16.04 machine on AWS EC2 using local storage. In real-life scenarios you will probably have all these components running on separate machines.

I started the instance in the public subnet of a VPC and then set up a security group to enable access from anywhere using SSH and TCP 5601 (for Kibana). Finally, I added a new elastic IP address and associated it with the running instance.

The example logs used for the tutorial are Apache access logs.

Step 1: Installing Elasticsearch

We will start with installing the main component in the stack — Elasticsearch. Since version 7.x, Elasticsearch has been bundled with Java so we can jump right ahead with adding Elastic's signing key:

```
1 wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
```

For installing Elasticsearch on Debian, we also need to install the apt-transport-https package:

```
1 sudo apt-get update
2 sudo apt-get install apt-transport-https
```

Our next step is to add the repository definition to our system:

```
1 echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo
2 tee -a /etc/apt/sources.list.d/elastic-7.x.list
```

All that's left to do is to update your repositories and install Elasticsearch:

```
1 sudo apt-get update && sudo apt-get install elasticsearch
```

Before we bootstrap Elasticsearch, we need to apply some basic configurations using the Elasticsearch configuration file at: `/etc/elasticsearch/elasticsearch.yml` :

```
1 sudo su
2 vim /etc/elasticsearch/elasticsearch.yml
```

Since we are installing Elasticsearch on AWS, we will bind Elasticsearch to localhost. Also, we need to define the private IP of our EC2 instance as a master-eligible node:

```
1 network.host: "localhost"
```



```
2 http.port:9200
3 cluster.initial_master_nodes: ["<InstancePrivateIP"]
```



(/users/login.html)



(/search)

[REFCARDZ \(/refcardz\)](#) [TREND REPORTS \(/trendreports\)](#) [WEBINARS \(/webinars\)](#) [ZONES](#) ▾

Save the file and run Elasticsearch with:

```
1 sudo service elasticsearch start
```

To confirm that everything is working as expected, point curl to: *http://localhost:9200* , and you should see something like the following output (give Elasticsearch a minute or two before you start to worry about not seeing any response):

```
1 {
2   "name" : "ip-172-31-49-60",
3   "cluster_name" : "elasticsearch",
4   "cluster_uuid" : "yP0uMKA6QmCsXQon-rxawQ",
5   "version" : {
6     "number" : "7.0.0",
7     "build_flavor" : "default",
8     "build_type" : "deb",
9     "build_hash" : "b7e28a7",
10    "build_date" : "2019-04-05T22:55:32.697037Z",
11    "build_snapshot" : false,
12    "lucene_version" : "8.0.0",
13    "minimum_wire_compatibility_version" : "6.7.0",
14    "minimum_index_compatibility_version" : "6.0.0-beta1"
15  },
16   "tagline" : "You Know, for Search"
17 }
```

Step 2: Installing Logstash

Next up, the "L" in ELK — Logstash. Logstash will require us to install Java 8 which is fine because this is also a requirement for running Kafka:

```
1 sudo apt-get install default-jre
```

Verify java is installed:

```
1 java -version
2
3 openjdk version "1.8.0_191"
4 OpenJDK Runtime Environment (build 1.8.0_191-8u191-b12-2ubuntu0.16.04.1-b12)
5 OpenJDK 64-Bit Server VM (build 25.191-b12, mixed mode)
```

Since we already defined the repository in the system, all we have to do to install Logstash is run:


```
1 sudo apt-get install logstash
```

Next, we will configure a Logstash pipeline that pulls our logs from a Kafka topic, process these logs and ships them on to Elasticsearch for indexing.

Let's create a new config file:

```
1 sudo vim /etc/logstash/conf.d/apache.conf
```

Paste the following configurations:



[REFCARDZ \(/refcardz\)](#)
[TREND REPORTS \(/trendreports\)](#)
[WEBINARS \(/webinars\)](#)
[ZONES](#)

```

1 input {
2   kafka {
3     bootstrap_servers => "localhost:9092"
4     topics => ["combined-apache-log"]
5   }
6 }
7
8 filter {
9   grok {
10    match => { "message" => "%{COMBINEDAPACHELOG}" }
11  }
12  date {
13    match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
14  }
15  geoip {
16    source => "clientip"
17  }
18 }
19
20 output {
21   elasticsearch {
22     hosts => ["localhost:9200"]
23   }
24 }

```

As you can see, we're using the Logstash Kafka input plugin to define the Kafka host and the topic we want Logstash to pull from. We're applying some filtering to the logs and we're shipping the data to our local Elasticsearch instance.

Save the file.

Step 3: Installing Kibana

Let's move on to the next component in the ELK Stack - Kibana. As before, we will use a simple apt command to install Kibana:

```
1 sudo apt-get install kibana
```

We will then open up the Kibana configuration file at: `/etc/kibana/kibana.yml`, and make sure we have the correct configurations defined:

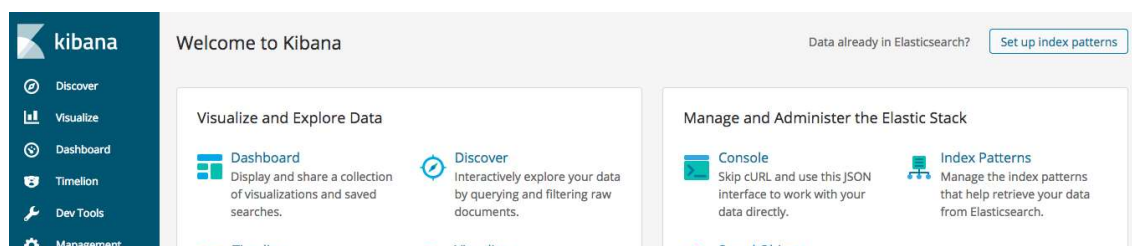
```
1 server.port: 5601
2 elasticsearch.url: "http://localhost:9200"
```

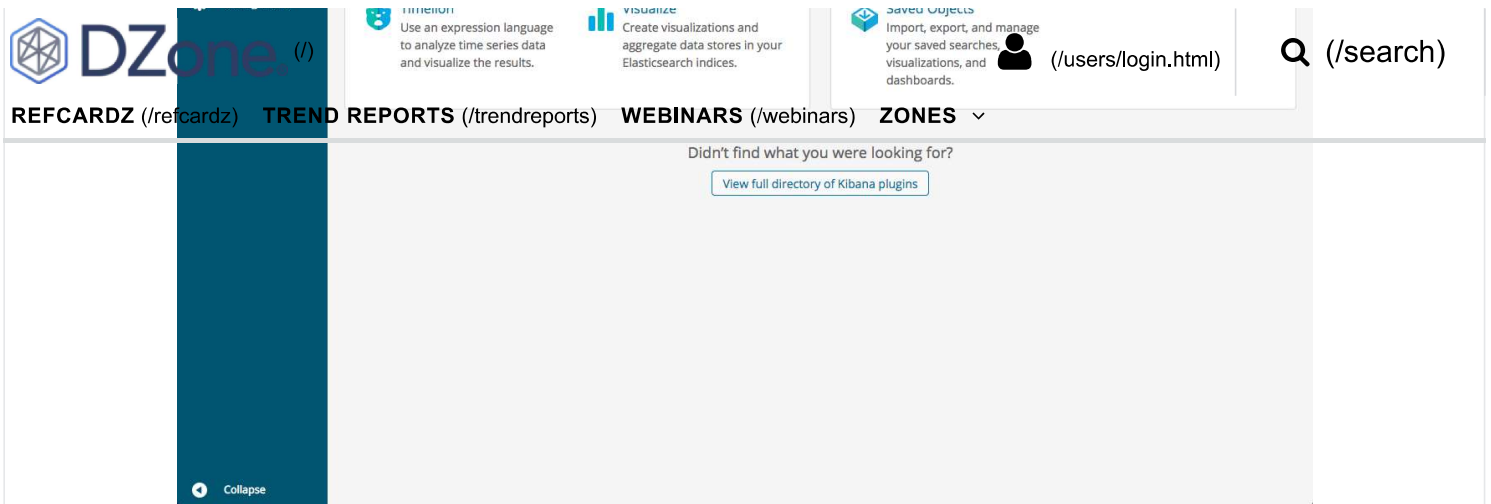
These specific configurations tell Kibana which Elasticsearch to connect to and which port to use.

Now, we can start Kibana with:

```
1 sudo service kibana start
```

Open up Kibana in your browser with: `http://localhost:5601` (`http://localhost:5601`). You will be presented with the Kibana home page.





Step 4: Installing Filebeat

As mentioned above, we will be using Filebeat to collect the log files and forward them to Kafka.

To install Filebeat, we will use:

```
1 sudo apt-get install filebeat
```

Let's open the Filebeat configuration file at:

```
1 /etc/filebeat/filebeat.yml
2
3 sudo vim /etc/filebeat/filebeat.yml
```

Enter the following configurations:

```
1 filebeat.inputs:
2 - type: log
3   enabled: true
4   paths:
5     - /var/log/apache2/access.log
6
7 output.kafka:
8   codec.format:
9     string: '%{[@timestamp]} %{[message]}'
10  hosts: ["localhost:9092"]
11  topic: apache
12  partition.round_robin:
13    reachable_only: false
14  required_acks: 1
15  compression: gzip
16  max_message_bytes: 1000000
```

In the input section, we are telling Filebeat what logs to collect — Apache access logs. In the output section, we are telling Filebeat to forward the data to our local Kafka server and the relevant topic (to be installed in the next step).

Note the use of the *codec.format* directive — this is to make sure the message and timestamp fields are extracted correctly. Otherwise, the lines are sent in JSON to Kafka.

Save the file.

Step 4: Installing Kafka

Our last and final installation involves setting up Apache Kafka — our message broker

```
1 sudo apt-get install zookeeperd
```

Next, let's download and extract Kafka:

```
1 wget http://apache.mivzakim.net/kafka/2.2.0/kafka_2.12-2.2.0.tgz
2 tar -xvzf kafka_2.12-2.2.0.tgz
3 sudo cp -r kafka_2.12-2.2.0 /opt/kafka
```

We are now ready to run Kafka, which we will do with this script:

```
1 sudo /opt/kafka/bin/kafka-server-start.sh
2 /opt/kafka/config/server.properties
```

You should begin to see some INFO messages in your console:

```
1 [2019-04-22 11:48:16,489] INFO Registered
2 kafka:type=kafka.Log4jController MBean
3 (kafka.utils.Log4jControllerRegistration$)
4 [2019-04-22 11:48:18,589] INFO starting (kafka.server.KafkaServer)
```

Next, we're going to create a topic for our Apache logs:

```
1 bin/kafka-topics.sh --create --zookeeper localhost:2181
2 --replication-factor 1 --partitions 1 --topic apache
3
4 Created topic apache.
```

We are all set to start the pipeline.

Step 5: Starting the Data Pipeline

Now that we have all the pieces in place, it's time to start all the components in charge of running the data pipeline.

First, we'll start Filebeat:

```
1 sudo service filebeat start
```

Then, Logstash:


```
1 sudo service logstash start
```

It will take a few minutes for the pipeline to start streaming logs. To see Kafka in action, enter the following command in a separate tab:

```
1 /opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server
2 localhost:9092 --topic apache --from-beginning
```

If your Apache web server is indeed serving requests, you should begin to see the messages being forwarded to the Kafka topic by Filebeat in your console:

```
1 2019-04-23T13:50:01.559Z 89.138.90.236 - - [23/Apr/2019:13:50:00 +0000]
2 "GET /wp-content/themes/WordPress-4.7.5/images/avatars/avatar-default.png" 200 436 "-" "Mozilla/5.0"
```



GET /mysite.html HTTP/1.1 200 420 - Mozilla/5.0
 [Macintosh; Intel Mac OS X 10_13_6] AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/73.0.3683.86 Safari/537.36"

2019-04-23T13:51:36.581Z 89.138.90.236 - -
 [23/Apr/2019:13:51:34 +0000] GET /mysite.html HTTP/1.1 200 427

REFCARDZ (/refcardz) TRENDREPORTS (/trendreports) WEBINARS (/webinars) ZONES ▾

7 "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36
 8 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36"

To make sure Logstash is aggregating the data and shipping it into Elasticsearch, use:

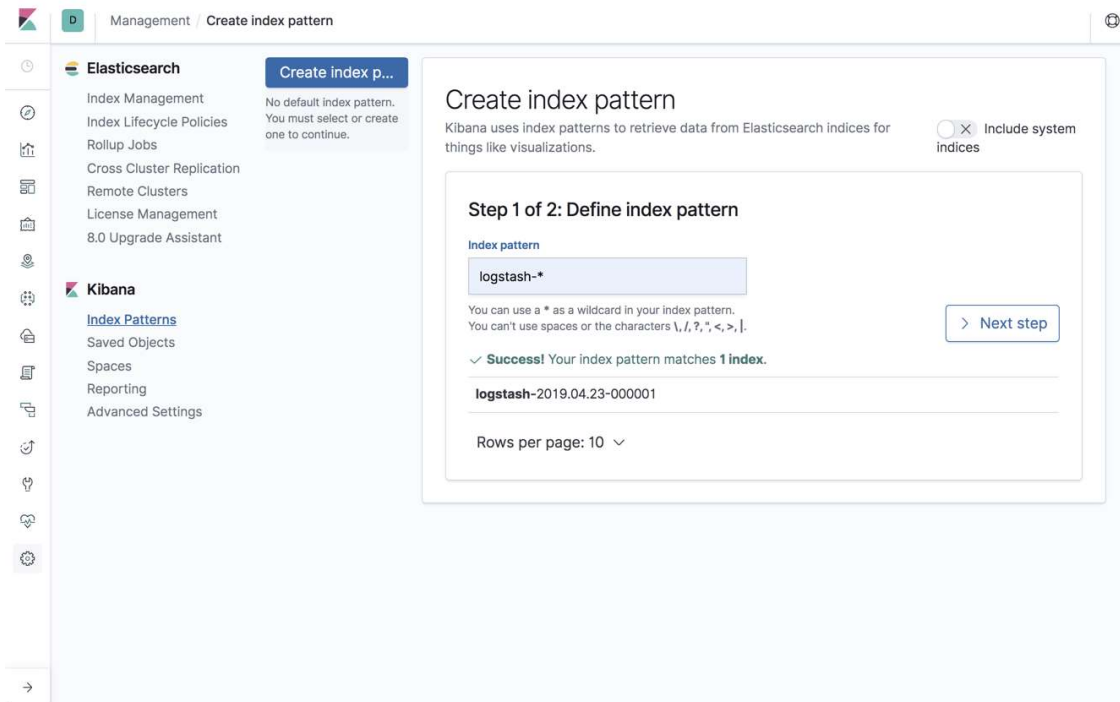
```
1 curl -X GET "localhost:9200/_cat/indices?v"
```

If all is working as expected, you should see a logstash-* index listed:

	health	status	index	uuid	pri	rep	docs.count	docs.deleted	store.size	pri.store.size
1	green	open	.kibana_task_manager	zmMH6yy8Q6yg2jJHxq3MFA	1	0	2	0	45.4kb	45.4kb
2	yellow	open	logstash-2019.04.23-000001	rBx5r_gIS3W2dTxxHzGJVvQ	1	1	9	0	69.4kb	69.4kb
3	green	open	.kibana_1	rv5f8uHnQTCGe8YrcKAw1Q	1	0	5	0		

If you don't see this index, it's time to do some debugging I'm afraid. Check out this blog post for some tips on debugging Logstash (<https://logz.io/blog/debug-logstash/>).

All we have to do now is define the index pattern in Kibana to begin analysis. This is done under **Management** → **Kibana Index Patterns**.



Management Create index pattern

Elasticsearch Create index p...

Index Management
 Index Lifecycle Policies
 Rollup Jobs
 Cross Cluster Replication
 Remote Clusters
 License Management
 8.0 Upgrade Assistant

No default index pattern.
 You must select or create one to continue.

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 \, /, ?, *, <, >, |.

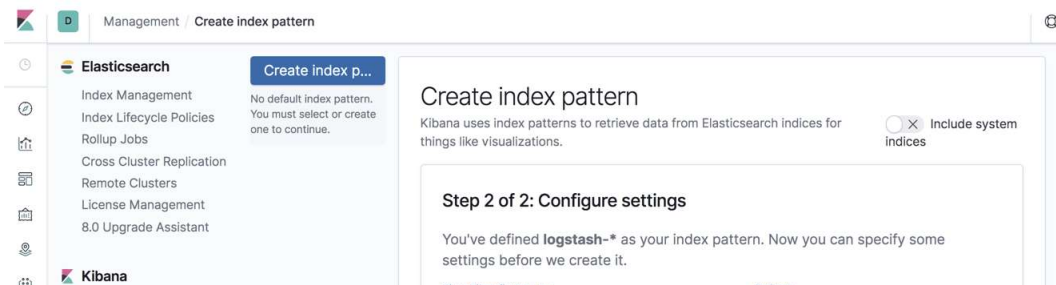
> Next step

✓ Success! Your index pattern matches 1 index.

logstash-2019.04.23-000001

Rows per page: 10 ▾

Kibana will identify the index, so simply define it in the relevant field and continue on to the next step of selecting the timestamp field:



Management Create index pattern

Elasticsearch Create index p...

Index Management
 Index Lifecycle Policies
 Rollup Jobs
 Cross Cluster Replication
 Remote Clusters
 License Management
 8.0 Upgrade Assistant

No default index pattern.
 You must select or create one to continue.

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

Default

Once you create the index pattern, you'll see a list of all the parsed and mapped fields:


Open the Discover page to begin analyzing your data!



Resilient data pipelines are a must in any production-grade ELK deployments. Logs are crucial for piecing together events and they are needed most in emergencies. We cannot afford to have our logging infrastructure fail at the exact point in time when it is most required. Kafka, and similar brokers, play a huge part in buffering the data flow so Logstash and Elasticsearch don't cave under the pressure of a sudden burst.

The example above is a basic setup of course. Production deployments will include multiple Kafka instances, a much larger amount of data and much more complicated pipelines. This will involve a considerable amount of engineering time and resources, something to take into consideration. The instructions here will help you understand how to get started though. I also recommend taking a look at our article explaining how to log Kafka itself (<https://logz.io/blog/kafka-logging/>).

[Kafka](#) [Elasticsearch](#) [Data \(Computing\)](#) [Kibana](#) [Pipeline \(Software\)](#)

Published at DZone with permission of Daniel Berman, DZone MVB. [See the original article here.](#)  (<https://logz.io/blog/deploying-kafka-with-elk/>)

Opinions expressed by DZone contributors are their own.

Popular on DZone

- [Creating Your First Cloud-Agnostic Serverless Application with Java \(/articles/creating-your-first-cloud-agnostic-serverless-appl?fromrel=true\)](/articles/creating-your-first-cloud-agnostic-serverless-appl?fromrel=true)
- [When Should You Go For Microservice Architecture? \(/articles/when-should-you-go-for-microservices-architecture?fromrel=true\)](/articles/when-should-you-go-for-microservices-architecture?fromrel=true)
- [The Benefits of Containerization \(/articles/the-benefits-of-containerization?fromrel=true\)](/articles/the-benefits-of-containerization?fromrel=true)
- [Architecture of Kubernetes \(/articles/architecture-of-kubernetes?fromrel=true\)](/articles/architecture-of-kubernetes?fromrel=true)

Big Data Partner Resources

ABOUT US

[About DZone \(/pages/about\)](/pages/about)

[Send feedback \(mailto:support@dzone.com\)](mailto:support@dzone.com)

[Careers \(https://careers.dzone.com/\)](https://careers.dzone.com/)

[Sitemap \(/sitemap\)](/sitemap)

ADVERTISE

[Advertise with DZone \(https://advertise.dzone.com\)](https://advertise.dzone.com)

CONTRIBUTE ON DZONE

[Article Submission Guidelines \(/articles/dzones-article-submission-guidelines\)](/articles/dzones-article-submission-guidelines)

[MVB Program \(/pages/mvb\)](/pages/mvb)

[Become a Contributor \(/pages/contribute\)](/pages/contribute)

[Visit the Writers' Zone \(/writers-zone\)](/writers-zone)




600 Park Offices Drive

Suite 300

Durham, NC 27709

support@dzone.com (mailto:support@dzone.com)

+1 (919) 678-0300 (tel:+19196780300)

Let's be friends:    

(/pages/help/https://www.dzone.com/dzonecompany/dzone/)

DZone.com is powered by  (https://devada.com/answerhub/)