

Contents

Logstash Introduction	28
The Power of Logstash	28
Logs and Metrics.....	29
The Web.....	29
Data Stores and Streams.....	30
Sensors and IoT.....	30
Easily Enrich Everything	30
Choose Your Stash	31
Getting Started with Logstash	32
Installing Logstash.....	33
Installing from a Downloaded Binary.....	33
Installing from Package Repositories.....	33
Docker.....	35
Stashing Your First Event	36
Parsing Logs with Logstash	37
Configuring Filebeat to Send Log Lines to Logstash.....	37
Configuring Logstash for Filebeat Input.....	38
Stitching Together Multiple Input and Output Plugins.....	49
How Logstash Works	53
Inputs	53
Filters	53
Outputs	53
Codecs	54
Execution Model	55
Setting Up and Running Logstash	56
Logstash Directory Layout.....	57
Directory Layout of .zip and .tar.gz Archives.....	57
Directory Layout of Debian and RPM Packages.....	57
Directory Layout of Docker Images.....	58
Logstash Configuration Files	60
Pipeline Configuration Files	60

Logstash 5.2 Configuration Guide

Settings Files	60
Running Logstash as a Service on Debian or RPM	61
Running Logstash by Using Systemd.....	61
Running Logstash by Using Upstart	61
Running Logstash by Using SysV	61
Running Logstash on Docker.....	61
Configuring Logstash for Docker.....	62
Logging Configuration	63
Settings File	64
Command-Line Flags	69
Logging	72
Log file location.....	72
Log4j 2 Configuration.....	72
Slowlog.....	72
Logging APIs	73
Persistent Queues	74
Limitations of Persistent Queues	74
How Persistent Queues Work	74
Configuring Persistent Queues	75
Handling Back Pressure	76
Controlling Durability	76
Disk Garbage Collection	77
Shutting Down Logstash	78
Stall Detection Example	78
Breaking changes	80
Changes in Logstash Core	80
Breaking Changes in Plugins	83
Ruby Filter and Custom Plugin Developers.....	85
Upgrading Logstash	87
Upgrading Using Package Managers	88
Upgrading Using a Direct Download	89
Upgrading Logstash to 5.0	90
When to Upgrade.....	90

Logstash 5.2 Configuration Guide

When Not to Upgrade.....	90
Configuring Logstash.....	91
Structure of a Config File	92
Plugin Configuration	92
Value Types	92
Array.....	93
Comments.....	95
Event Dependent Configuration	96
Using Environment Variables in the Configuration	102
Overview	102
Examples	102
Logstash Configuration Examples	105
Reloading the Config File	111
How Automatic Config Reloading Works.....	111
Managing Multiline Events	112
Examples of Multiline Codec Configuration.....	112
Glob Pattern Support	115
Deploying and Scaling Logstash.....	116
The Minimal Installation.....	116
Using Filters.....	116
Using Filebeat.....	117
Scaling to a Larger Elasticsearch Cluster.....	117
Managing Throughput Spikes with Message Queueing.....	118
Multiple Connections for Logstash High Availability	119
Scaling Logstash.....	121
Performance Tuning.....	122
Performance Troubleshooting Guide	123
Tuning and Profiling Logstash Performance	125
Monitoring APIs	128
Common Options	128
Node Info API	130
Pipeline Info	130
OS Info.....	130

JVM Info	131
Plugins Info API	132
Node Stats API	133
JVM Stats	133
Process Stats	134
Pipeline Stats	135
Reload Stats	136
OS Stats.....	137
Hot Threads API	138
Working with plugins	142
Listing plugins.....	142
Adding plugins to your deployment.....	142
Updating plugins	143
Removing plugins.....	143
Proxy Support.....	143
Generating Plugins.....	145
Offline Plugin Management	146
Building Offline Plugin Packs.....	146
Installing Offline Plugin Packs	147
Updating Offline Plugins	147
Building the Offline Package (Deprecated Procedure).....	147
WARNING: Deprecated in 5.2	147
Install or Update a local plugin (Deprecated Procedure).....	148
Managing Plugin Packs	148
Private Gem Repositories.....	150
Editing the Gemfile	150
Event API.....	152
Input plugins	155
beats.....	158
Synopsis.....	158
Details	160
cloudwatch.....	165
Synopsis.....	166

Logstash 5.2 Configuration Guide

Details	167
couchdb_changes	172
Synopsis.....	172
Details	173
drupal_dblog.....	178
Synopsis.....	178
Details	178
elasticsearch.....	181
Synopsis.....	182
Details	183
eventlog	188
Synopsis.....	188
Details	188
exec	191
Synopsis.....	191
Details	191
file	194
Reading from remote network volumes.....	194
Tracking of current position in watched files	194
File rotation.....	195
Synopsis.....	195
Details	196
ganglia	200
Synopsis.....	200
Details	200
gelf	203
Synopsis.....	203
Details	204
gemfire.....	207
Synopsis.....	207
Details	207
generator	211
Synopsis.....	211

Logstash 5.2 Configuration Guide

Details	211
github	215
Synopsis.....	215
Details	215
graphite.....	217
Synopsis.....	218
Details	219
heartbeat	223
Synopsis.....	223
Details	223
heroku	227
Synopsis.....	227
Details	227
http.....	231
Synopsis.....	231
Details	232
http_poller	236
Example.....	236
Synopsis.....	237
Details	238
imap	245
Synopsis.....	245
Details	246
irc	250
Synopsis.....	250
Details	251
jdbc.....	255
Drivers.....	255
Scheduling.....	255
State	255
Dealing With Large Result-sets	255
Usage:.....	256
Configuring SQL statement	256

Logstash 5.2 Configuration Guide

Predefined Parameters	256
Synopsis.....	257
Details	259
jmx.....	266
Synopsis.....	267
Details	268
kafka.....	271
Synopsis.....	272
Details	274
kinesis.....	284
Synopsis.....	284
Details	285
log4j.....	288
Synopsis.....	288
Details	289
lumberjack	292
Synopsis.....	292
Details	293
meetup.....	296
Synopsis.....	296
Details	296
pipe	299
Synopsis.....	299
Details	299
puppet_facter	302
Synopsis.....	302
Details	302
rabbitmq	306
Synopsis.....	307
Details	308
rackspace	316
Synopsis.....	316
Details	316

Logstash 5.2 Configuration Guide

redis	321
Synopsis.....	321
Details	322
relp	326
Synopsis.....	326
Details	327
rss	330
Synopsis.....	330
Details	330
s3.....	333
Synopsis.....	333
Details	335
salesforce.....	340
Example.....	340
Synopsis.....	341
Details	341
snmptrap.....	346
Synopsis.....	346
Details	347
sqlite.....	350
Synopsis.....	351
Details	351
sqs	354
Synopsis.....	355
Details	356
stdin	360
Synopsis.....	360
Details	360
stomp	363
Synopsis.....	363
Details	364
syslog.....	367
Synopsis.....	367

Logstash 5.2 Configuration Guide

Details	368
tcp	372
Synopsis.....	372
Details	373
twitter	377
Synopsis.....	377
Details	378
udp	383
Synopsis.....	383
Details	383
unix.....	387
Synopsis.....	387
Details	387
varnishlog.....	391
Synopsis.....	391
Details	391
websocket	394
Synopsis.....	394
Details	394
wmi	397
Synopsis.....	397
Details	398
xmpp	401
Synopsis.....	401
Details	401
zenoss.....	405
Synopsis.....	405
Details	406
zeromq	410
Synopsis.....	410
Details	410
Output plugins	414
boundary.....	417

Logstash 5.2 Configuration Guide

Synopsis.....	417
Details	418
circonus.....	421
Synopsis.....	421
Details	421
cloudwatch.....	424
Summary:.....	424
Details:	424
Synopsis.....	425
Details	426
csv	432
Synopsis.....	432
Details	433
datadog.....	436
Synopsis.....	436
Details	437
datadog_metrics	440
Synopsis.....	440
Details	441
elasticsearch.....	444
Template management for Elasticsearch 5.x.....	444
Retry Policy	444
Batch Sizes.....	445
DNS Caching	445
Synopsis.....	445
Details	447
email.....	459
Synopsis.....	459
Details	460
exec	465
Synopsis.....	465
Details	466
file	468

Logstash 5.2 Configuration Guide

Synopsis.....	468
Details	469
ganglia	472
Synopsis.....	472
Details	473
gelf	476
Synopsis.....	476
Details	477
google_bigquery	481
Synopsis.....	482
Details	483
google_cloud_storage.....	488
Synopsis.....	489
Details	489
graphite.....	493
Synopsis.....	493
Details	494
graphtastic	498
Synopsis.....	498
Details	499
hipchat	502
Synopsis.....	502
Details	503
http.....	506
Synopsis.....	506
Details	507
influxdb	514
Synopsis.....	514
Details	515
irc	520
Synopsis.....	520
Details	521
jira	524

Logstash 5.2 Configuration Guide

Synopsis.....	524
Details	525
juggernaut.....	528
Synopsis.....	528
Details	529
Synopsis.....	533
Details	535
librato.....	543
Synopsis.....	543
Details	543
loggly	547
Synopsis.....	547
Details	548
lumberjack	551
Synopsis.....	551
Details	551
metriccatcher.....	554
Synopsis.....	554
Details	555
mongodb.....	558
Synopsis.....	558
Details	558
nagios	561
Synopsis.....	561
Details	562
nagios_nsca.....	564
Synopsis.....	564
Details	565
newrelic.....	568
Synopsis.....	568
Details	569
opentsdb.....	572
Synopsis.....	572

Logstash 5.2 Configuration Guide

Details	572
pagerduty.....	575
Synopsis.....	575
Details	576
pipe	578
Synopsis.....	578
Details	578
rabbitmq	581
Synopsis.....	581
Details	582
rackspace	588
Synopsis.....	588
Details	588
redis	591
Synopsis.....	591
Details	592
redmine.....	597
Synopsis.....	597
Details	598
Synopsis.....	602
Details	602
riemann.....	606
Synopsis.....	606
Details	607
s3.....	610
Synopsis.....	611
Details	613
sns	619
Upgrading to 2.0.0	619
Synopsis.....	619
Details	620
solr_http.....	624
Synopsis.....	624

Logstash 5.2 Configuration Guide

Details	625
sqS	627
Batch Publishing.....	627
Synopsis.....	628
Details	629
statsd.....	633
Synopsis.....	634
Details	634
stdout.....	638
Synopsis.....	638
Details	639
stomp	641
Synopsis.....	641
Details	641
syslog.....	644
Synopsis.....	644
Details	645
tcp	650
Synopsis.....	650
Details	651
udp	654
Synopsis.....	654
Details	654
webhdFs.....	657
Dependencies.....	657
Operational Notes.....	657
Usage.....	657
Synopsis.....	658
Details	659
websocket	663
Synopsis.....	663
Details	663
xmpp	666

Logstash 5.2 Configuration Guide

Synopsis.....	666
Details	666
zabbix	669
Synopsis.....	669
Details	670
zeromq	673
Synopsis.....	673
Details	673
Filter plugins.....	676
aggregate	679
Example #1.....	679
Example #2 : no start event	680
Example #3 : no end event.....	681
Example #4 : no end event and tasks come one after the other.....	682
How it works	682
Use Cases	683
Synopsis.....	683
Details	684
alter	690
Synopsis.....	690
Details	691
anonymize.....	695
Synopsis.....	695
Details	695
cidr	700
Synopsis.....	700
Details	700
cipher	705
Synopsis.....	705
Details	706
clone.....	712
Synopsis.....	712
Details	712

Logstash 5.2 Configuration Guide

collate.....	716
Synopsis.....	716
Details	717
csv	720
Synopsis.....	720
Details	721
date	726
Synopsis.....	726
Details	727
de_dot.....	735
Synopsis.....	735
Details	736
dissect	740
Dissect or how to de-structure text.....	740
Synopsis.....	742
Details	743
dns.....	748
Synopsis.....	748
Details	749
drop.....	754
Synopsis.....	754
Details	755
elapsed.....	759
Synopsis.....	760
Details	761
elasticsearch.....	765
Synopsis.....	766
Details	767
environment	773
Synopsis.....	773
Details	774
extractnumbers.....	778
Synopsis.....	778

Logstash 5.2 Configuration Guide

Details	779
fingerprint	782
Synopsis.....	782
Details	782
geoip	787
Synopsis.....	787
Details	788
grok	794
Grok Basics	794
Regular Expressions	795
Custom Patterns.....	795
Synopsis.....	797
Details	797
i18n	804
Synopsis.....	804
Details	804
json.....	808
Synopsis.....	808
Details	809
json_encode.....	814
Synopsis.....	814
Details	815
kv.....	819
Synopsis.....	819
Details	820
metaevent.....	830
Synopsis.....	830
Details	830
metricize	834
Synopsis.....	834
Details	835
metrics	840
meter values	840

Logstash 5.2 Configuration Guide

timer values	840
Example: Computing event rate	841
Synopsis.....	842
Details	843
mutate.....	848
Synopsis.....	848
Details	849
oui	856
Synopsis.....	856
Details	856
prune	861
Synopsis.....	862
Details	862
punct	867
Synopsis.....	867
Details	867
range	871
Synopsis.....	871
Details	872
ruby	876
Synopsis.....	876
Details	877
sleep.....	881
Synopsis.....	881
Details	881
split.....	887
Synopsis.....	887
Details	888
syslog_pri	891
Synopsis.....	892
Details	893
throttle	897
Synopsis.....	899

Logstash 5.2 Configuration Guide

Details	900
tld	905
Synopsis.....	905
Details	905
translate.....	910
Synopsis.....	910
Details	911
truncate.....	918
Synopsis.....	918
Details	919
urldecode.....	923
Synopsis.....	923
Details	924
useragent.....	929
Synopsis.....	929
Details	930
uuid	935
Synopsis.....	935
Details	935
xml.....	939
Synopsis.....	939
Details	940
yaml.....	947
Synopsis.....	947
Details	947
zeromq	952
Synopsis.....	952
Details	953
Codec plugins.....	957
avro	959
Encoding.....	959
Decoding	959
Usage.....	959

Logstash 5.2 Configuration Guide

Synopsis.....	959
Details	960
cef.....	962
Synopsis.....	962
Details	963
cloudfront	967
Synopsis.....	967
Details	968
cloudtrail.....	970
Synopsis.....	970
Details	970
collectd.....	971
Synopsis.....	971
Details	971
compress_spooler.....	974
Synopsis.....	974
Details	974
dots	975
Synopsis.....	975
Details	975
edn	977
Synopsis.....	977
Details	977
edn_lines.....	979
Synopsis.....	979
Details	979
es_bulk.....	981
Synopsis.....	981
Details	981
fluent.....	983
Synopsis.....	983
Details	984
graphite.....	985

Logstash 5.2 Configuration Guide

Synopsis.....	985
Details	985
gzip_lines.....	988
Synopsis.....	988
Details	989
json.....	991
Synopsis.....	991
Details	992
json_lines	995
Synopsis.....	995
Details	996
line.....	999
Synopsis.....	999
Details	1000
msgpack	1003
Synopsis.....	1003
Details	1003
multiline.....	1005
Synopsis.....	1006
Details	1008
netflow.....	1012
Supported Netflow/IPFIX exporters.....	1012
Usage.....	1012
Synopsis.....	1013
Details	1013
nmap	1017
Synopsis.....	1017
Details	1017
oldlogstashjson	1019
Synopsis.....	1019
plain.....	1020
Synopsis.....	1020
Details	1021

protobuf.....	1024
Synopsis.....	1024
Details	1025
Synopsis.....	1027
Details	1027
s3_plain.....	1029
Synopsis.....	1029
sflow.....	1030
Synopsis.....	1030
Details	1031
Contributing to Logstash.....	1034
Adding plugins.....	1034
Extending Logstash core	1035
Get started	1036
Create a GitHub repo for your new plugin	1036
Use the plugin generator tool	1036
Copy the input code	1037
See what your plugin looks like	1038
Coding input plugins.....	1039
encoding	1039
require Statements.....	1039
Plugin Body.....	1039
Inline Documentation	1039
class Declaration.....	1040
config_name	1040
Configuration Parameters.....	1040
Plugin Methods.....	1041
register Method.....	1041
run Method.....	1042
Building the Plugin	1043
External dependencies	1043
Add a Gemfile.....	1044
Add a gemspec file	1044

Runtime & Development Dependencies	1045
Jar dependencies	1045
Add Tests	1046
Clone and test!	1046
Building and Testing	1047
Build	1047
Test installation	1047
Submitting your plugin to RubyGems.org and logstash-plugins	1048
Licensing	1048
Publishing to RubyGems.org	1048
Contributing your source code to logstash-plugins	1049
Benefits	1049
Acceptance Guidelines	1050
How to write a Logstash codec plugin	1051
Get started	1051
Create a GitHub repo for your new plugin	1051
Use the plugin generator tool	1051
Copy the codec code	1052
See what your plugin looks like	1053
Coding codec plugins	1054
encoding	1054
require Statements	1054
Plugin Body	1054
Inline Documentation	1054
class Declaration	1055
config_name	1055
Configuration Parameters	1055
Plugin Methods	1056
register Method	1056
decode Method	1057
encode Method	1057
Building the Plugin	1057
External dependencies	1057

Add a Gemfile	1058
Add a <code>gemspec</code> file	1058
Runtime & Development Dependencies	1059
Jar dependencies	1060
Add Tests	1060
Clone and test!	1060
Building and Testing	1061
Build	1061
Test installation	1061
Submitting your plugin to RubyGems.org and logstash-plugins	1063
Licensing	1063
Publishing to RubyGems.org	1063
Contributing your source code to logstash-plugins	1064
Benefits	1064
Acceptance Guidelines	1064
How to write a Logstash filter plugin	1065
Get started	1065
Create a GitHub repo for your new plugin	1065
Use the plugin generator tool	1065
Copy the filter code	1066
See what your plugin looks like	1067
Coding filter plugins	1068
<code>encoding</code>	1068
<code>require</code> Statements	1068
Plugin Body	1068
Inline Documentation	1068
<code>class</code> Declaration	1069
<code>config_name</code>	1069
Configuration Parameters	1069
Plugin Methods	1070
<code>register</code> Method	1070
<code>filter</code> Method	1070
Building the Plugin	1071

External dependencies	1071
Add a Gemfile	1072
Add a <code>gemspec</code> file	1072
Runtime & Development Dependencies	1073
Jar dependencies	1074
Add Tests	1074
Clone and test!	1074
Building and Testing	1075
Build	1075
Test installation	1075
Submitting your plugin to RubyGems.org and logstash-plugins	1077
Licensing	1077
Publishing to RubyGems.org	1077
Contributing your source code to logstash-plugins	1078
Benefits	1078
Acceptance Guidelines	1078
Get started	1079
Create a GitHub repo for your new plugin	1079
Use the plugin generator tool	1079
Copy the output code	1080
See what your plugin looks like	1081
Coding output plugins	1082
<code>encoding</code>	1082
<code>require</code> Statements	1082
Plugin Body	1082
Inline Documentation	1082
<code>class</code> Declaration	1083
<code>config_name</code>	1083
Configuration Parameters	1083
Plugin Methods	1084
<code>register</code> Method	1084
Building the Plugin	1085
External dependencies	1085

Add a Gemfile	1085
Add a gemspec file	1086
Runtime & Development Dependencies	1087
Jar dependencies	1087
Add Tests	1087
Clone and test!	1087
Building and Testing	1088
Build	1088
Test installation	1089
Submitting your plugin to RubyGems.org and logstash-plugins	1089
Licensing	1090
Publishing to RubyGems.org	1090
Contributing your source code to logstash-plugins	1091
Benefits	1091
Acceptance Guidelines	1091
Contributing a Patch to a Logstash Plugin	1092
Input Plugins	1092
Codec Plugins	1092
Filter Plugins.....	1093
Output Plugins	1093
Process	1093
Testing Methodologies	1094
Putting it all together	1097
Logstash Plugins Community Maintainer Guide.....	1102
Contribution Guidelines.....	1102
Document Goals.....	1102
Development Workflow.....	1102
Versioning Plugins.....	1105
Logging.....	1106
Contributor License Agreement (CLA) Guidance	1106
Need Help?.....	1106
Community Administration.....	1107
Submitting your plugin to RubyGems.org and the logstash-plugins repository.....	1108

Logstash 5.2 Configuration Guide

Licensing.....	1108
Publishing to RubyGems.org	1108
Contributing your source code to logstash-plugins	1109
Benefits	1109
Acceptance Guidelines.....	1109
Glossary of Terms	1110

Logstash Introduction

Logstash is an open source data collection engine with real-time pipelining capabilities. Logstash can dynamically unify data from disparate sources and normalize the data into destinations of your choice. Cleanse and democratize all your data for diverse advanced downstream analytics and visualization use cases.

While Logstash originally drove innovation in log collection, its capabilities extend well beyond that use case. Any type of event can be enriched and transformed with a broad array of input, filter, and output plugins, with many native codecs further simplifying the ingestion process. Logstash accelerates your insights by harnessing a greater volume and variety of data.

The Power of Logstash

The ingestion workhorse for Elasticsearch and more

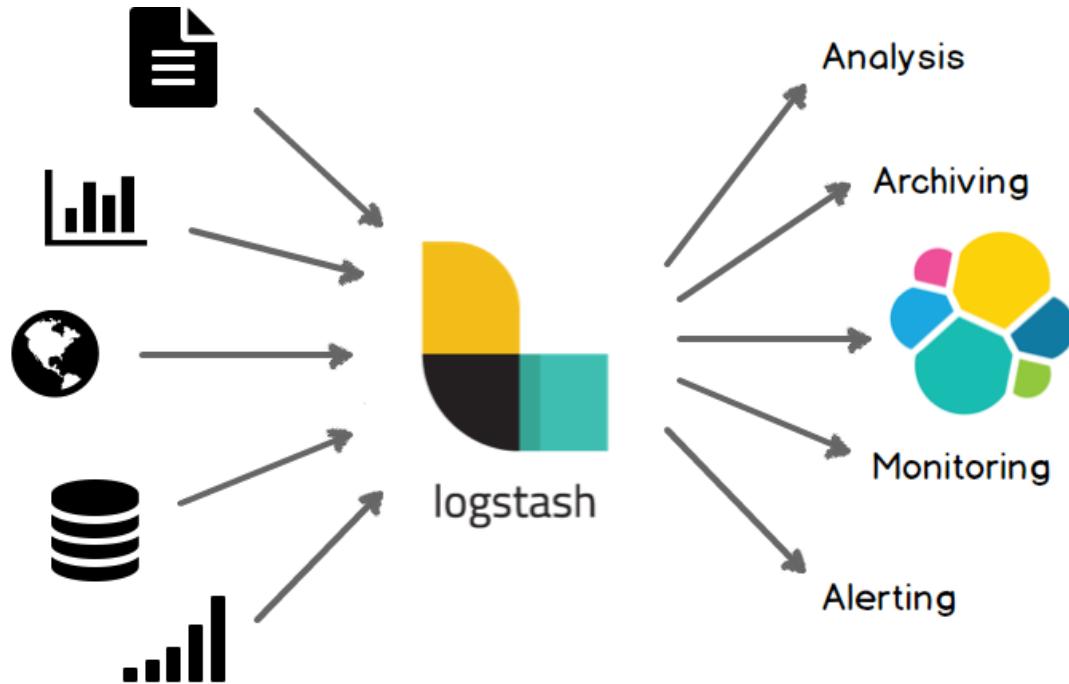
Horizontally scalable data processing pipeline with strong Elasticsearch and Kibana synergy

Pluggable pipeline architecture

Mix, match, and orchestrate different inputs, filters, and outputs to play in pipeline harmony

Community-extensible and developer-friendly plugin ecosystem

Over 200 plugins available, plus the flexibility of creating and contributing your own



Logstash Loves Data

Collect more, so you can know more. Logstash welcomes data of all shapes and sizes.

Logs and Metrics

Where it all started.

- Handle all types of logging data
 - Easily ingest a multitude of web logs like [Apache](#), and application logs like [log4j](#) for Java
 - Capture many other log formats like [syslog](#), [Windows event logs](#), networking and firewall logs, and more
- Enjoy complementary secure log forwarding capabilities with [Filebeat](#)
- Collect metrics from [Ganglia](#), [collectd](#), [NetFlow](#), [JMX](#), and many other infrastructure and application platforms over [TCP](#) and [UDP](#)

The Web

Unlock the World Wide Web.

- Transform [HTTP requests](#) into events
 - Consume from web service firehoses like [Twitter](#) for social sentiment analysis
 - Webhook support for GitHub, HipChat, JIRA, and countless other applications
 - Enables many [Watcher](#) alerting use cases
- Create events by polling [HTTP endpoints](#) on demand
 - Universally capture health, performance, metrics, and other types of data from web application interfaces
 - Perfect for scenarios where the control of polling is preferred over receiving

Data Stores and Streams

Discover more value from the data you already own.

- Better understand your data from any relational database or NoSQL store with a [JDBC](#) interface
- Unify diverse data streams from messaging queues like Apache [Kafka](#), [RabbitMQ](#), [Amazon SQS](#), and [ZeroMQ](#)

Sensors and IoT

Explore an expansive breadth of other data.

- In this age of technological advancement, the massive IoT world unleashes endless use cases through capturing and harnessing data from connected sensors.
- Logstash is the common event collection backbone for ingestion of data shipped from mobile devices to intelligent homes, connected vehicles, healthcare sensors, and many other industry specific applications.

Easily Enrich Everything

The better the data, the better the knowledge. Clean and transform your data during ingestion to gain near real-time insights immediately at index or output time. Logstash comes out-of-box with many aggregations and mutations along with pattern matching, geo mapping, and dynamic lookup capabilities.

- [Grok](#) is the bread and butter of Logstash filters and is used ubiquitously to derive structure out of unstructured data. Enjoy a wealth of integrated patterns aimed to help quickly resolve web, systems, networking, and other types of event formats.
- Expand your horizons by deciphering [geo coordinates](#) from IP addresses, normalizing [date](#) complexity, simplifying [key-value pairs](#) and [CSV](#) data, [anonymizing](#) sensitive information, and further enriching your data with [local lookups](#) or Elasticsearch [queries](#).
- Codecs are often used to ease the processing of common event structures like [JSON](#) and [multiline](#) events.

Choose Your Stash

Route your data where it matters most. Unlock various downstream analytical and operational use cases by storing, analyzing, and taking action on your data.

Analysis <ul style="list-style-type: none">• Elasticsearch• Data stores such as MongoDB and Riak	Archiving <ul style="list-style-type: none">• HDFS• S3• Google Cloud Storage
Monitoring <ul style="list-style-type: none">• Nagios• Ganglia• Zabbix• Graphite• Datadog• CloudWatch	Alerting <ul style="list-style-type: none">• Watcher with Elasticsearch• Email• Pagerduty• HipChat• IRC• SNS

Getting Started with Logstash

This section guides you through the process of installing Logstash and verifying that everything is running properly. After learning how to stash your first event, you go on to create a more advanced pipeline that takes Apache web logs as input, parses the logs, and writes the parsed data to an Elasticsearch cluster. Then you learn how to stitch together multiple input and output plugins to unify data from a variety of disparate sources.

This section includes the following topics:

- [Installing Logstash](#)
- [Stashing Your First Event](#)
- [Parsing Logs with Logstash](#)
- [Stitching Together Multiple Input and Output Plugins](#)

Installing Logstash

Logstash requires Java 8. Java 9 is not supported. Use the [official Oracle distribution](#) or an open-source distribution such as [OpenJDK](#).

To check your Java version, run the following command:

```
java -version
```

On systems with Java installed, this command produces output similar to the following:

```
java version "1.8.0_65"
Java(TM) SE Runtime Environment (build 1.8.0_65-b17)
Java HotSpot(TM) 64-Bit Server VM (build 25.65-b01, mixed mode)
```

Installing from a Downloaded Binary

Download the [Logstash installation file](#) that matches your host environment. Unpack the file. Do not install Logstash into a directory path that contains colon (:) characters.

On supported Linux operating systems, you can use a package manager to install Logstash.

Installing from Package Repositories

NOTE: We also have repositories available for APT and YUM based distributions. Note that we only provide binary packages, but no source packages, as the packages are created as part of the Logstash build.

We have split the Logstash package repositories by version into separate urls to avoid accidental upgrades across major versions. For all 5.x.y releases use 5.x as version number.

We use the PGP key [D88E42B4](#), Elastic's Signing Key, with fingerprint

```
4609 5ACC 8548 582C 1A26 99A9 D27D 666C D88E 42B4
```

to sign all our packages. It is available from <https://pgp.mit.edu>.

[APT](#)

Download and install the Public Signing Key:

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

Logstash 5.2 Configuration Guide

You may need to install the `apt-transport-https` package on Debian before proceeding:

```
sudo apt-get install apt-transport-https
```

Save the repository definition to `/etc/apt/sources.list.d/elastic-5.x.list`:

```
echo "deb https://artifacts.elastic.co/packages/5.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-5.x.list
```

WARNING!! Use the `echo` method described above to add the Logstash repository. Do not use `add-apt-repository` as it will add a `deb-src` entry as well, but we do not provide a source package. If you have added the `deb-src` entry, you will see an error like the following:

```
Unable to find expected entry 'main/source/Sources' in Release file (Wrong sources.list entry or malformed file)
```

Just delete the `deb-src` entry from the `/etc/apt/sources.list` file and the installation should work as expected.

Run `sudo apt-get update` and the repository is ready for use. You can install it with:

```
sudo apt-get update && sudo apt-get install logstash
```

See [Running Logstash](#) for details about managing Logstash as a system service.

[YUM](#)

Download and install the public signing key:

```
rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
```

Add the following in your `/etc/yum.repos.d/` directory in a file with a `.repo` suffix, for example `logstash.repo`

```
[logstash-5.x]
name=Elastic repository for 5.x packages
baseurl=https://artifacts.elastic.co/packages/5.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md
```

And your repository is ready for use. You can install it with:

```
sudo yum install logstash
```

WARNING!! The repositories do not work with older rpm based distributions that still use RPM v3, like CentOS5.

See the [Running Logstash](#) document for managing Logstash as a system service.

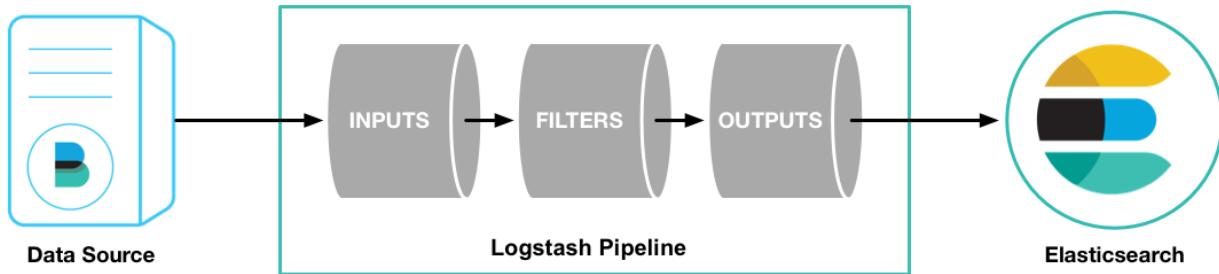
Docker

An image is available for running Logstash as a Docker container. It is available from the Elastic Docker registry. See [Running Logstash on Docker](#) for details on how to configure and run Logstash Docker containers.

Stashing Your First Event

First, let's test your Logstash installation by running the most basic *Logstash pipeline*.

A Logstash pipeline has two required elements, `input` and `output`, and one optional element, `filter`. The input plugins consume data from a source, the filter plugins modify the data as you specify, and the output plugins write the data to a destination.



To test your Logstash installation, run the most basic Logstash pipeline:

```
cd logstash-5.2.1
bin/logstash -e 'input { stdin {} } output { stdout {} }'
```

The `-e` flag enables you to specify a configuration directly from the command line. Specifying configurations at the command line lets you quickly test configurations without having to a file between iterations. The pipeline in the example takes input from the standard input, `stdin`, and moves that input to the standard output, `stdout`, in a structured format.

After starting Logstash, wait until you see "Pipeline main started" and then enter `hello world` at the command prompt:

```
hello world
2013-11-21T01:22:14.405+0000 0.0.0.0 hello world
```

Logstash adds timestamp and IP address information to the message. Exit Logstash by issuing a **CTRL-D** command in the shell where Logstash is running.

Congratulations! You've created and run a basic Logstash pipeline. Next, you learn how to create a more realistic pipeline.

Parsing Logs with Logstash

In [Stashing Your First Event](#), you created a basic Logstash pipeline to test your Logstash setup. In the real world, a Logstash pipeline is a bit more complex: it typically has one or more input, filter, and output plugins.

In this section, you create a Logstash pipeline that uses Filebeat to take Apache web logs as input, parses those logs to create specific, named fields from the logs, and writes the parsed data to an Elasticsearch cluster. Rather than defining the pipeline configuration at the command line, you'll define the pipeline in a config file.

To get started, go [here](#) to download the sample data set used in this example. Unpack the file.

Configuring Filebeat to Send Log Lines to Logstash

Before you create the Logstash pipeline, you'll configure Filebeat to send log lines to Logstash. The [Filebeat](#) client is a lightweight, resource-friendly tool that collects logs from files on the server and forwards these logs to your Logstash instance for processing. Filebeat is designed for reliability and low latency. Filebeat has a light resource footprint on the host machine, and the [Beats input](#) plugin minimizes the resource demands on the Logstash instance.

NOTE: In a typical use case, Filebeat runs on a separate machine from the machine running your Logstash instance. For the purposes of this tutorial, Logstash and Filebeat are running on the same machine.

The default Logstash installation includes the [Beats input](#) plugin. The Beats input plugin enables Logstash to receive events from the Elastic Beats framework, which means that any Beat written to work with the Beats framework, such as Packetbeat and Metricbeat, can also send event data to Logstash.

To install Filebeat on your data source machine, download the appropriate package from the Filebeat [product page](#). You can also refer to [Getting Started with Filebeat](#) in the Beats documentation for additional installation instructions.

After installing Filebeat, you need to configure it. Open the `filebeat.yml` file located in your Filebeat installation directory, and replace the contents with the following lines. Make sure `paths` points to the example Apache log file, `logstash-tutorial.log`, that you downloaded earlier:

```
filebeat.prospectors:  
- input_type: log  
  paths:
```

```
- /path/to/file/logstash-tutorial.log
output.logstash:
  hosts: ["localhost:5043"]
    Absolute path to the file or files that Filebeat processes.
```

Save your changes.

To keep the configuration simple, you won't specify TLS/SSL settings as you would in a real world scenario.

At the data source machine, run Filebeat with the following command:

```
sudo ./filebeat -e -c filebeat.yml -d "publish"
```

Filebeat will attempt to connect on port 5043. Until Logstash starts with an active Beats plugin, there won't be any answer on that port, so any messages you see regarding failure to connect on that port are normal for now.

Configuring Logstash for Filebeat Input

Next, you create a Logstash configuration pipeline that uses the Beats input plugin to receive events from Beats.

The following text represents the skeleton of a configuration pipeline:

```
# The # character at the beginning of a line indicates a comment. Use
# comments to describe your configuration.
input {
}
# The filter part of this file is commented out to indicate that it is
# optional.
# filter {
#
# }
output {
}
```

This skeleton is non-functional, because the input and output sections don't have any valid options defined.

To get started, copy and paste the skeleton configuration pipeline into a file named `first-pipeline.conf` in your home Logstash directory.

Next, configure your Logstash instance to use the Beats input plugin by adding the following lines to the `input` section of the `first-pipeline.conf` file:

```
beats {
```

Logstash 5.2 Configuration Guide

```
    port => "5043"
}
```

You'll configure Logstash to write to Elasticsearch later. For now, you can add the following line to the `output` section so that the output is printed to stdout when you run Logstash:

```
    stdout { codec => rubydebug }
```

When you're done, the contents of `first-pipeline.conf` should look like this:

```
input {
    beats {
        port => "5043"
    }
}
# The filter part of this file is commented out to indicate that it is
# optional.
# filter {
#
# }
output {
    stdout { codec => rubydebug }
}
```

To verify your configuration, run the following command:

```
bin/logstash -f first-pipeline.conf --config.test_and_exit
```

The `--config.test_and_exit` option parses your configuration file and reports any errors.

If the configuration file passes the configuration test, start Logstash with the following command:

```
bin/logstash -f first-pipeline.conf --config.reload.automatic
```

The `--config.reload.automatic` option enables automatic config reloading so that you don't have to stop and restart Logstash every time you modify the configuration file.

If your pipeline is working correctly, you should see a series of events like the following written to the console:

```
{
    "@timestamp" => 2016-10-11T20:54:06.733Z,
    "offset" => 325,
    "@version" => "1",
    "beat" => {
        "hostname" => "My-MacBook-Pro.local",
        "name" => "My-MacBook-Pro.local"
    },
    "input_type" => "log",
    "host" => "My-MacBook-Pro.local",
```

Logstash 5.2 Configuration Guide

```
    "source" => "/path/to/file/logstash-tutorial.log",
    "message" => "83.149.9.216 -- [04/Jan/2015:05:13:42 +0000] \"GET
/presentations/logstash-monitorama-2013/images/kibana-search.png HTTP/1.1\""
200 203023 \"http://semicomplete.com/presentations/logstash-monitorama-
2013/\" \"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36\"",
    "type" => "log",
    "tags" => [
      [0] "beats_input_codec_plain_applied"
    ]
}
...

```

Parsing Web Logs with the Grok Filter Plugin

Now you have a working pipeline that reads log lines from Filebeat. However you'll notice that the format of the log messages is not ideal. You want to parse the log messages to create specific, named fields from the logs. To do this, you'll use the `grok` filter plugin.

The `grok` filter plugin is one of several plugins that are available by default in Logstash. For details on how to manage Logstash plugins, see the [reference documentation](#) for the plugin manager.

The `grok` filter plugin enables you to parse the unstructured log data into something structured and queryable.

Because the `grok` filter plugin looks for patterns in the incoming log data, configuring the plugin requires you to make decisions about how to identify the patterns that are of interest to your use case. A representative line from the web server log sample looks like this:

```
83.149.9.216 -- [04/Jan/2015:05:13:42 +0000] "GET /presentations/logstash-
monitorama-2013/images/kibana-search.png
HTTP/1.1" 200 203023 "http://semicomplete.com/presentations/logstash-
monitorama-2013/" "Mozilla/5.0 (Macintosh; Intel
Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77
Safari/537.36"
```

The IP address at the beginning of the line is easy to identify, as is the timestamp in brackets. To parse the data, you can use the `%{COMBINEDAPACHELOG}` grok pattern, which structures lines from the Apache log using the following schema:

Information	Field Name
IP Address	clientip
User ID	ident
User Authentication	auth
timestamp	timestamp
HTTP Verb	verb
Request body	request

Logstash 5.2 Configuration Guide

HTTP Version	httpversion
HTTP Status Code	response
Bytes served	bytes
Referrer URL	referrer
User agent	agent

Edit the `first-pipeline.conf` file and replace the entire `filter` section with the following text:

```
filter {
    grok {
        match => { "message" => "%{COMBINEDAPACHELOG}" }
    }
}
```

When you're done, the contents of `first-pipeline.conf` should look like this:

```
input {
    beats {
        port => "5043"
    }
}
filter {
    grok {
        match => { "message" => "%{COMBINEDAPACHELOG}" }
    }
}
output {
    stdout { codec => rubydebug }
}
```

Save your changes. Because you've enabled automatic config reloading, you don't have to restart Logstash to pick up your changes. However, you do need to force Filebeat to read the log file from scratch. To do this, go to the terminal window where Filebeat is running and press **Ctrl+C** to shut down Filebeat. Then delete the Filebeat registry file. For example, run:

```
sudo rm data/registry
```

Since Filebeat stores the state of each file it harvests in the registry, deleting the registry file forces Filebeat to read all the files it's harvesting from scratch.

Next, restart Filebeat with the following command:

```
sudo ./filebeat -e -c filebeat.yml -d "publish"
```

After processing the log file with the grok pattern, the events will have the following JSON representation:

```
{
```

Logstash 5.2 Configuration Guide

```
"request" => "/presentations/logstash-monitorama-2013/images/kibana-search.png",
  "agent" => "\"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36\"",
  "offset" => 325,
  "auth" => "-",
  "ident" => "-",
  "input_type" => "log",
  "verb" => "GET",
  "source" => "/path/to/file/logstash-tutorial.log",
  "message" => "83.149.9.216 -- [04/Jan/2015:05:13:42 +0000] \"GET /presentations/logstash-monitorama-2013/images/kibana-search.png HTTP/1.1\" 200 203023 \"http://semicomplete.com/presentations/logstash-monitorama-2013/\" \"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36\"",
  "type" => "log",
  "tags" => [
    [0] "beats_input_codec_plain_applied"
  ],
  "referrer" => "\"http://semicomplete.com/presentations/logstash-monitorama-2013/\"",
  "@timestamp" => 2016-10-11T21:04:36.167Z,
  "response" => "200",
  "bytes" => "203023",
  "clientip" => "83.149.9.216",
  "@version" => "1",
  "beat" => {
    "hostname" => "My-MacBook-Pro.local",
    "name" => "My-MacBook-Pro.local"
  },
  "host" => "My-MacBook-Pro.local",
  "httpversion" => "1.1",
  "timestamp" => "04/Jan/2015:05:13:42 +0000"
}
```

Notice that the event includes the original message, but the log message is also broken down into specific fields.

Enhancing Your Data with the Geoip Filter Plugin

In addition to parsing log data for better searches, filter plugins can derive supplementary information from existing data. As an example, the [geoip](#) plugin looks up IP addresses, derives geographic location information from the addresses, and adds that location information to the logs.

Configure your Logstash instance to use the `geoip` filter plugin by adding the following lines to the `filter` section of the `first-pipeline.conf` file:

```
geoip {
  source => "clientip"
}
```

The `geoip` plugin configuration requires you to specify the name of the source field that contains the IP address to look up. In this example, the `clientip` field contains the IP address.

Since filters are evaluated in sequence, make sure that the `geoip` section is after the `grok` section of the configuration file and that both the `grok` and `geoip` sections are nested within the `filter` section.

When you're done, the contents of `first-pipeline.conf` should look like this:

```
input {
    beats {
        port => "5043"
    }
}

filter {
    grok {
        match => { "message" => "%{COMBINEDAPACHELOG}" }
    }
    geoip {
        source => "clientip"
    }
}

output {
    stdout { codec => rubydebug }
}
```

Save your changes. To force Filebeat to read the log file from scratch, as you did earlier, shut down Filebeat (press Ctrl+C), delete the registry file, and then restart Filebeat with the following command:

```
sudo ./filebeat -e -c filebeat.yml -d "publish"
```

Notice that the event now contains geographic location information:

```
{
    "request" => "/presentations/logstash-monitorama-2013/images/kibana-search.png",
    "agent" => "\"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36\"",
    "geoip" => {
        "timezone" => "Europe/Moscow",
        "ip" => "83.149.9.216",
        "latitude" => 55.7522,
        "continent_code" => "EU",
        "city_name" => "Moscow",
        "country_code2" => "RU",
        "country_name" => "Russia",
        "dma_code" => nil,
        "country_code3" => "RU",
        "region_name" => "Moscow",
        "location" => [
            [0] 37.6156,
            [1] 55.7522
        ]
    }
}
```

```
        ],
        "postal_code" => "101194",
        "longitude" => 37.6156,
        "region_code" => "MOW"
    },
    ...
}
```

[Indexing Your Data into Elasticsearch](#)

Now that the web logs are broken down into specific fields, the Logstash pipeline can index the data into an Elasticsearch cluster. Edit the `first-pipeline.conf` file and replace the entire `output` section with the following text:

```
output {
    elasticsearch {
        hosts => [ "localhost:9200" ]
    }
}
```

With this configuration, Logstash uses http protocol to connect to Elasticsearch. The above example assumes that Logstash and Elasticsearch are running on the same instance. You can specify a remote Elasticsearch instance by using the `hosts` configuration to specify something like `hosts => ["es-machine:9092"]`.

At this point, your `first-pipeline.conf` file has input, filter, and output sections properly configured, and looks something like this:

```
input {
    beats {
        port => "5043"
    }
}
filter {
    grok {
        match => { "message" => "%{COMBINEDAPACHELOG}" }
    }
    geoip {
        source => "clientip"
    }
}
output {
    elasticsearch {
        hosts => [ "localhost:9200" ]
    }
}
```

Save your changes. To force Filebeat to read the log file from scratch, as you did earlier, shut down Filebeat (press Ctrl+C), delete the registry file, and then restart Filebeat with the following command:

```
sudo ./filebeat -e -c filebeat.yml -d "publish"
```

Testing Your Pipeline

Now that the Logstash pipeline is configured to index the data into an Elasticsearch cluster, you can query Elasticsearch.

Try a test query to Elasticsearch based on the fields created by the `grok` filter plugin. Replace `$DATE` with the current date, in `YYYY.MM.DD` format:

```
curl -XGET 'localhost:9200/logstash-$DATE/_search?pretty&q=response=200'
```

The date used in the index name is based on UTC, not the timezone where Logstash is running. If the query returns `index_not_found_exception`, make sure that `logstash-$DATE` reflects the actual name of the index. To see a list of available indexes, use this query: `curl 'localhost:9200/_cat/indices?v'`.

You should get multiple hits back. For example:

```
{
  "took" : 21,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 98,
    "max_score" : 3.745223,
    "hits" : [
      {
        "_index" : "logstash-2016.10.11",
        "_type" : "log",
        "_id" : "AVe14gMiYMKU36o_eVsA",
        "_score" : 3.745223,
        "_source" : {
          "request" : "/presentations/logstash-monitorama-
2013/images/frontend-response-codes.png",
          "agent" : "\"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36\"",
          "geoip" : {
            "timezone" : "Europe/Moscow",
            "ip" : "83.149.9.216",
            "latitude" : 55.7522,
            "continent_code" : "EU",
            "city_name" : "Moscow",
            "country_code2" : "RU",
            "country_name" : "Russia",
            "dma_code" : null,
            "country_code3" : "RU",
            "region_name" : "Moscow",
            "location" : [
              {
                "lat" : 55.7522,
                "lon" : 37.6156
              }
            ]
          }
        }
      }
    ]
  }
}
```

Logstash 5.2 Configuration Guide

```
    37.6156,
    55.7522
  ],
  "postal_code" : "101194",
  "longitude" : 37.6156,
  "region_code" : "MOW"
},
"offset" : 2932,
"auth" : "-",
"ident" : "-",
"input_type" : "log",
"verb" : "GET",
"source" : "/path/to/file/logstash-tutorial.log",
"message" : "83.149.9.216 -- [04/Jan/2015:05:13:45 +0000] \"GET
/presentations/logstash-monitorama-2013/images/frontend-response-codes.png
HTTP/1.1\" 200 52878 \"http://semicomplete.com/presentations/logstash-
monitorama-2013\" \"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36\""",
  "type" : "log",
  "tags" : [
    "beats_input_codec_plain_applied"
  ],
  "referrer" : "\"http://semicomplete.com/presentations/logstash-
monitorama-2013\"",
  "@timestamp" : "2016-10-11T22:34:25.317Z",
  "response" : "200",
  "bytes" : "52878",
  "clientip" : "83.149.9.216",
  "@version" : "1",
  "beat" : {
    "hostname" : "My-MacBook-Pro.local",
    "name" : "My-MacBook-Pro.local"
  },
  "host" : "My-MacBook-Pro.local",
  "httpversion" : "1.1",
  "timestamp" : "04/Jan/2015:05:13:45 +0000"
}
},
...
}
```

Try another search for the geographic information derived from the IP address. Replace \$DATE with the current date, in YYYY.MM.DD format:

```
curl -XGET 'localhost:9200/logstash-
$DATE/_search?pretty&q=geoip.city_name=Buffalo'
```

A few log entries come from Buffalo, so the query produces the following response:

```
{
  "took" : 3,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
```

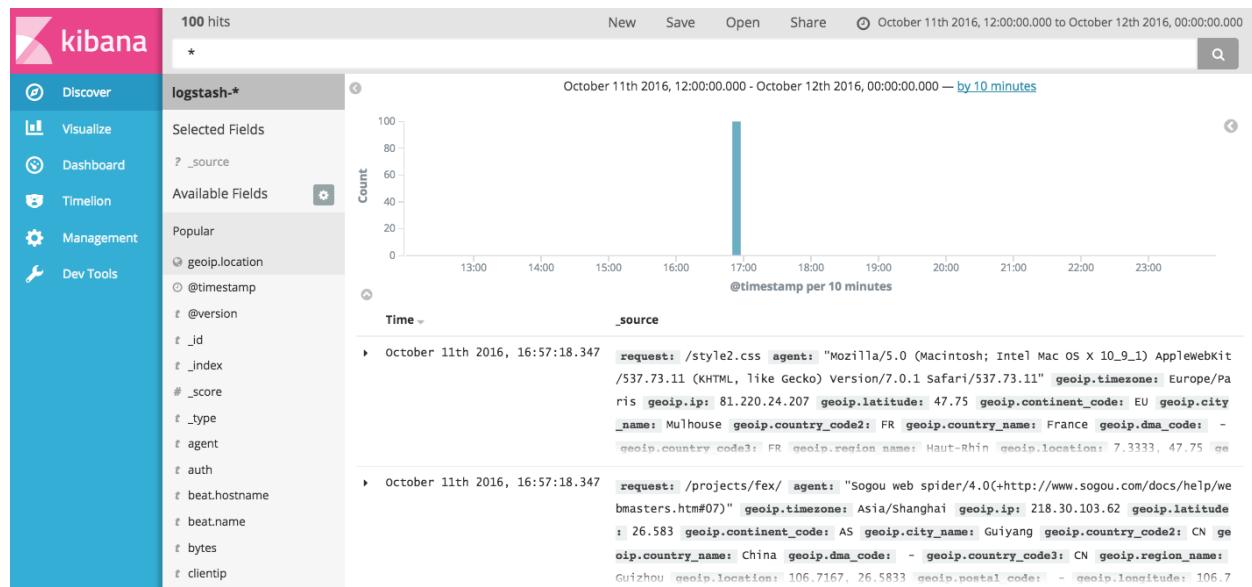
Logstash 5.2 Configuration Guide

```
"failed" : 0
},
"hits" : {
  "total" : 3,
  "max_score" : 2.6390574,
  "hits" : [
    {
      "_index" : "logstash-2016.10.11",
      "_type" : "log",
      "_id" : "AVe14gMjYMkU36o_eVtO",
      "_score" : 2.6390574,
      "_source" : {
        "request" : "/?flav=rss20",
        "agent" : "\"-\""",
        "geoip" : {
          "timezone" : "America/New_York",
          "ip" : "108.174.55.234",
          "latitude" : 42.9864,
          "continent_code" : "NA",
          "city_name" : "Buffalo",
          "country_code2" : "US",
          "country_name" : "United States",
          "dma_code" : 514,
          "country_code3" : "US",
          "region_name" : "New York",
          "location" : [
            -78.7279,
            42.9864
          ],
          "postal_code" : "14221",
          "longitude" : -78.7279,
          "region_code" : "NY"
        },
        "offset" : 21471,
        "auth" : "-",
        "ident" : "-",
        "input_type" : "log",
        "verb" : "GET",
        "source" : "/path/to/file/logstash-tutorial.log",
        "message" : "108.174.55.234 - - [04/Jan/2015:05:27:45 +0000] \"GET /?flav=rss20 HTTP/1.1\" 200 29941 \"-\" \"-\""",
        "type" : "log",
        "tags" : [
          "beats_input_codec_plain_applied"
        ],
        "referrer" : "\"-\""",
        "@timestamp" : "2016-10-11T22:34:25.318Z",
        "response" : "200",
        "bytes" : "29941",
        "clientip" : "108.174.55.234",
        "@version" : "1",
        "beat" : {
          "hostname" : "My-MacBook-Pro.local",
          "name" : "My-MacBook-Pro.local"
        },
        "host" : "My-MacBook-Pro.local",
        "httpversion" : "1.1",
        "version" : "5.2.0"
      }
    }
  ]
}
```

Logstash 5.2 Configuration Guide

```
    "timestamp" : "04/Jan/2015:05:27:45 +0000"
  }
},  
..
```

If you are using Kibana to visualize your data, you can also explore the Filebeat data in Kibana:



See the [Filebeat getting started docs](#) for info about loading the Kibana index pattern for Filebeat.

You've successfully created a pipeline that uses Filebeat to take Apache web logs as input, parses those logs to create specific, named fields from the logs, and writes the parsed data to an Elasticsearch cluster. Next, you learn how to create a pipeline that uses multiple input and output plugins.

Stitching Together Multiple Input and Output Plugins

The information you need to manage often comes from several disparate sources, and use cases can require multiple destinations for your data. Your Logstash pipeline can use multiple input and output plugins to handle these requirements.

In this section, you create a Logstash pipeline that takes input from a Twitter feed and the Filebeat client, then sends the information to an Elasticsearch cluster as well as writing the information directly to a file.

Reading from a Twitter Feed

To add a Twitter feed, you use the `twitter` input plugin. To configure the plugin, you need several pieces of information:

- A *consumer key*, which uniquely identifies your Twitter app.
- A *consumer secret*, which serves as the password for your Twitter app.
- One or more *keywords* to search in the incoming feed. The example shows using "cloud" as a keyword, but you can use whatever you want.
- An *oauth token*, which identifies the Twitter account using this app.
- An *oauth token secret*, which serves as the password of the Twitter account.

Visit <https://dev.twitter.com/apps> to set up a Twitter account and generate your consumer key and secret, as well as your access token and secret. See the docs for the `twitter` input plugin if you're not sure how to generate these keys.

Like you did earlier when you worked on [Parsing Logs with Logstash](#), create a config file (called `second-pipeline.conf`) that contains the skeleton of a configuration pipeline. If you want, you can reuse the file you created earlier, but make sure you pass in the correct config file name when you run Logstash.

Add the following lines to the `input` section of the `second-pipeline.conf` file, substituting your values for the placeholder values shown here:

```
twitter {
    consumer_key => "enter_your_consumer_key_here"
    consumer_secret => "enter_your_secret_here"
    keywords => ["cloud"]
    oauth_token => "enter_your_access_token_here"
    oauth_token_secret => "enter_your_access_token_secret_here"
}
```

Configuring Filebeat to Send Log Lines to Logstash

As you learned earlier in [Configuring Filebeat to Send Log Lines to Logstash](#), the Filebeat client is a lightweight, resource-friendly tool that collects logs from files on the server and forwards these logs to your Logstash instance for processing.

After installing Filebeat, you need to configure it. Open the `filebeat.yml` file located in your Filebeat installation directory, and replace the contents with the following lines. Make sure `paths` points to your syslog:

```
filebeat.prospectors:  
- input_type: log  
  paths:
```

```
    - /var/log/*.log  
fields:
```

```
  type: syslog
```

```
output.logstash:  
  hosts: ["localhost:5043"]
```

Absolute path to the file or files that Filebeat processes.

Adds a field called `type` with the value `syslog` to the event.

Save your changes.

To keep the configuration simple, you won't specify TLS/SSL settings as you would in a real world scenario.

Configure your Logstash instance to use the Filebeat input plugin by adding the following lines to the `input` section of the `second-pipeline.conf` file:

```
beats {  
  port => "5043"  
}
```

[Writing Logstash Data to a File](#)

You can configure your Logstash pipeline to write data directly to a file with the `file` output plugin.

Configure your Logstash instance to use the `file` output plugin by adding the following lines to the `output` section of the `second-pipeline.conf` file:

```
file {  
  path => "/path/to/target/file"  
}
```

[Writing to Multiple Elasticsearch Nodes](#)

Writing to multiple Elasticsearch nodes lightens the resource demands on a given Elasticsearch node, as well as providing redundant points of entry into the cluster when a particular node is unavailable.

Logstash 5.2 Configuration Guide

To configure your Logstash instance to write to multiple Elasticsearch nodes, the `output` section of the `second-pipeline.conf` file to read:

```
output {
    elasticsearch {
        hosts => ["IP Address 1:port1", "IP Address 2:port2", "IP Address 3"]
    }
}
```

Use the IP addresses of three non-master nodes in your Elasticsearch cluster in the host line. When the `hosts` parameter lists multiple IP addresses, Logstash load-balances requests across the list of addresses. Also note that the default port for Elasticsearch is 9200 and can be omitted in the configuration above.

[Testing the Pipeline](#)

At this point, your `second-pipeline.conf` file looks like this:

```
input {
    twitter {
        consumer_key => "enter_your_consumer_key_here"
        consumer_secret => "enter_your_secret_here"
        keywords => ["cloud"]
        oauth_token => "enter_your_access_token_here"
        oauth_token_secret => "enter_your_access_token_secret_here"
    }
    beats {
        port => "5043"
    }
}
output {
    elasticsearch {
        hosts => ["IP Address 1:port1", "IP Address 2:port2", "IP Address 3"]
    }
    file {
        path => "/path/to/target/file"
    }
}
```

Logstash is consuming data from the Twitter feed you configured, receiving data from Filebeat, and indexing this information to three nodes in an Elasticsearch cluster as well as writing to a file.

At the data source machine, run Filebeat with the following command:

```
sudo ./filebeat -e -c filebeat.yml -d "publish"
```

Filebeat will attempt to connect on port 5043. Until Logstash starts with an active Beats plugin, there won't be any answer on that port, so any messages you see regarding failure to connect on that port are normal for now.

To verify your configuration, run the following command:

```
bin/logstash -f second-pipeline.conf --config.test_and_exit
```

The `--config.test_and_exit` option parses your configuration file and reports any errors. When the configuration file passes the configuration test, start Logstash with the following command:

```
bin/logstash -f second-pipeline.conf
```

Use the `grep` utility to search in the target file to verify that information is present:

```
grep syslog /path/to/target/file
```

Run an Elasticsearch query to find the same information in the Elasticsearch cluster:

```
curl -XGET 'localhost:9200/logstash-$DATE/_search?pretty&q=fields.type:syslog'
```

Replace `$DATE` with the current date, in `YYYY.MM.DD` format.

To see data from the Twitter feed, try this query:

```
curl -XGET 'http://localhost:9200/logstash-$DATE/_search?pretty&q=client:iphone'
```

Again, remember to replace `$DATE` with the current date, in `YYYY.MM.DD` format.

How Logstash Works

The Logstash event processing pipeline has three stages: inputs → filters → outputs. Inputs generate events, filters modify them, and outputs ship them elsewhere. Inputs and outputs support codecs that enable you to encode or decode the data as it enters or exits the pipeline without having to use a separate filter.

Inputs

You use inputs to get data into Logstash. Some of the more commonly-used inputs are:

- **file**: reads from a file on the filesystem, much like the UNIX command `tail -f`
- **syslog**: listens on the well-known port 514 for syslog messages and parses according to the RFC3164 format
- **redis**: reads from a redis server, using both redis channels and redis lists. Redis is often used as a "broker" in a centralized Logstash installation, which queues Logstash events from remote Logstash "shippers".
- **beats**: processes events sent by [Filebeat](#).

For more information about the available inputs, see [Input Plugins](#).

Filters

Filters are intermediary processing devices in the Logstash pipeline. You can combine filters with conditionals to perform an action on an event if it meets certain criteria. Some useful filters include:

- **grok**: parse and structure arbitrary text. Grok is currently the best way in Logstash to parse unstructured log data into something structured and queryable. With 120 patterns built-in to Logstash, it's more than likely you'll find one that meets your needs!
- **mutate**: perform general transformations on event fields. You can rename, remove, replace, and modify fields in your events.
- **drop**: drop an event completely, for example, *debug* events.
- **clone**: make a copy of an event, possibly adding or removing fields.
- **geoip**: add information about geographical location of IP addresses (also displays amazing charts in Kibana!)

For more information about the available filters, see [Filter Plugins](#).

Outputs

Outputs are the final phase of the Logstash pipeline. An event can pass through multiple outputs, but once all output processing is complete, the event has finished its execution. Some commonly used outputs include:

- **elasticsearch**: send event data to Elasticsearch. If you're planning to save your data in an efficient, convenient, and easily queryable format... Elasticsearch is the way to go. Period. Yes, we're biased :)
- **file**: write event data to a file on disk.
- **graphite**: send event data to graphite, a popular open source tool for storing and graphing metrics. <http://graphite.readthedocs.io/en/latest/>
- **statsd**: send event data to statsd, a service that "listens for statistics, like counters and timers, sent over UDP and sends aggregates to one or more pluggable backend services". If you're already using statsd, this could be useful for you!

For more information about the available outputs, see [Output Plugins](#).

Codecs

Codecs are basically stream filters that can operate as part of an input or output. Codecs enable you to easily separate the transport of your messages from the serialization process. Popular codecs include `json`, `msgpack`, and `plain` (text).

- **json**: encode or decode data in the JSON format.
- **multiline**: merge multiple-line text events such as java exception and stacktrace messages into a single event.

For more information about the available codecs, see [Codec Plugins](#).

[Setting Up and Running Logstash »](#)

Execution Model

The Logstash event processing pipeline coordinates the execution of inputs, filters, and outputs.

Each input stage in the Logstash pipeline runs in its own thread. Inputs write events to a common Java [SynchronousQueue](#). This queue holds no events, instead transferring each pushed event to a free worker, blocking if all workers are busy. Each pipeline worker thread takes a batch of events off this queue, creating a buffer per worker, runs the batch of events through the configured filters, then runs the filtered events through any outputs. The size of the batch and number of pipeline worker threads are configurable (see [Tuning and Profiling Logstash Performance](#)).

By default, Logstash uses in-memory bounded queues between pipeline stages (input → filter and filter → output) to buffer events. If Logstash terminates unsafely, any events that are stored in memory will be lost. To prevent data loss, you can enable Logstash to persist in-flight events to disk. See [Persistent Queues](#) for more information.

Setting Up and Running Logstash

Before reading this section, see [Installing Logstash](#) for basic installation instructions to get you started.

This section includes additional information on how to set up and run Logstash, including:

- [Logstash Directory Layout](#)
- [Logstash Configuration Files](#)
- [Running Logstash as a Service on Debian or RPM](#)
- [Running Logstash on Docker](#)
- [Settings File](#)
- [Command-Line Flags](#)
- [Logging](#)
- [Persistent Queues](#)
- [Shutting Down Logstash](#)

Logstash Directory Layout

This section describes the default directory structure that is created when you unpack the Logstash installation packages.

Directory Layout of .zip and .tar.gz Archives

The .zip and .tar.gz packages are entirely self-contained. All files and directories are, by default, contained within the home directory — the directory created when unpacking the archive.

This is very convenient because you don't have to create any directories to start using Logstash, and uninstalling Logstash is as easy as removing the home directory. However, it is advisable to change the default locations of the config and the logs directories so that you do not delete important data later on.

Type	Description	Default Location	Setting
home	Home directory of the Logstash installation.	{extract.path}- Directory created by unpacking the archive	
bin	Binary scripts, including logstash to start Logstash and logstash-plugin to install plugins	{extract.path}/bin	
settings	Configuration files, including logstash.yml and jvm.options	{extract.path}/config	path.settings
logs	Log files	{extract.path}/logs	path.logs
plugins	Local, non Ruby-Gem plugin files. Each plugin is contained in a subdirectory. Recommended for development only.	{extract.path}/plugins	path.plugins

Directory Layout of Debian and RPM Packages

The Debian package and the RPM package each place config files, logs, and the settings files in the appropriate locations for the system:

Type	Description	Default Location	Setting
home	Home directory of the Logstash installation.	/usr/share/logstash	

Type	Description	Default Location	Setting
bin	Binary scripts including <code>logstash</code> to start Logstash and <code>logstash-plugin</code> to install plugins	/usr/share/logstash/bin	
settings	Configuration files, including <code>logstash.yml</code> , <code>jvm.options</code> , and <code>startup.options</code>	/etc/logstash	path.settings
conf	Logstash pipeline configuration files	/etc/logstash/conf.d	path.config
logs	Log files	/var/log/logstash	path.logs
plugins	Local, non Ruby-Gem plugin files. Each plugin is contained in a subdirectory. Recommended for development only.	/usr/share/logstash/plugins	path.plugins

Directory Layout of Docker Images

The Docker images are created from the `.tar.gz` packages, and follow a similar directory layout.

Type	Description	Default Location	Setting
home	Home directory of the Logstash installation.	/usr/share/logstash	
bin	Binary scripts, including <code>logstash</code> to start Logstash and <code>logstash-plugin</code> to install plugins	/usr/share/logstash/bin	
settings	Configuration files, including <code>logstash.yml</code> and <code>jvm.options</code>	/usr/share/logstash/config	path.settings
conf	Logstash pipeline configuration files	/usr/share/logstash/pipeline	path.config
plugins	Local, non Ruby-Gem plugin files. Each plugin is contained in a subdirectory. Recommended for development only.	/usr/share/logstash/plugins	path.plugins

NOTE: Logstash Docker containers do not create log files by default. They log to standard output.

Logstash Configuration Files

Logstash has two types of configuration files: *pipeline configuration files*, which define the Logstash processing pipeline, and *settings files*, which specify options that control Logstash startup and execution.

Pipeline Configuration Files

You create pipeline configuration files when you define the stages of your Logstash processing pipeline. On deb and rpm, you place the pipeline configuration files in the `/etc/logstash/conf.d` directory. Logstash tries to load all files in the `/etc/logstash/conf.d` directory, so don't store any non-config files or backup files in this directory.

See [Configuring Logstash](#) for more info.

Settings Files

The settings files are already defined in the Logstash installation. Logstash includes the following settings files:

`logstash.yml`

Contains Logstash configuration flags. You can set flags in this file instead of passing the flags at the command line. Any flags that you set at the command line override the corresponding settings in the `logstash.yml` file. See [Settings File](#) for more info.

`jvm.options`

Contains JVM configuration flags. Specify each flag on a separate line. You can also use this file to set the locale for Logstash.

`startup.options (Linux)`

Contains options used by the `system-install` script in `/usr/share/logstash/bin` to build the appropriate startup script for your system. When you install the Logstash package, the `system-install` script executes at the end of the installation process and uses the settings specified in `startup.options` to set options such as the user, group, service name, and service description. By default, Logstash services are installed under the user `logstash`. The `startup.options` file makes it easier for you to install multiple instances of the Logstash service. You can copy the file and change the values for specific settings. Note that the `startup.options` file is not read at startup. If you want to change the Logstash startup script (for example, to change the Logstash user or read from a different configuration path), you must re-run the `system-install` script (as root) to pass in the new settings.

Running Logstash as a Service on Debian or RPM

Logstash is not started automatically after installation. How to start and stop Logstash depends on whether your system uses systemd, upstart, or SysV.

Here are some common operating systems and versions, and the corresponding startup styles they use. This list is intended to be informative, not exhaustive.

Distribution	Service System
Ubuntu 16.04 and newer	systemd
Ubuntu 12.04 through 15.10	upstart
Debian 8 "jessie" and newer	systemd
Debian 7 "wheezy" and older	sysv
CentOS (and RHEL) 7 and newer	systemd
CentOS (and RHEL) 6	upstart

Running Logstash by Using Systemd

Distributions like Debian Jessie, Ubuntu 15.10+, and many of the SUSE derivatives use systemd and the `systemctl` command to start and stop services. Logstash places the systemd unit files in `/etc/systemd/system` for both deb and rpm. After installing the package, you can start up Logstash with:

```
sudo systemctl start logstash.service
```

Running Logstash by Using Upstart

For systems that use upstart, you can start Logstash with:

```
sudo initctl start logstash
```

The auto-generated configuration file for upstart systems is `/etc/init/logstash.conf`.

Running Logstash by Using SysV

For systems that use SysV, you can start Logstash with:

```
sudo /etc/init.d/logstash start
```

The auto-generated configuration file for SysV systems is `/etc/init.d/logstash`.

Running Logstash on Docker

Docker images for Logstash are available from the Elastic Docker registry.

Obtaining Logstash for Docker is as simple as issuing a `docker pull` command against the Elastic Docker registry.

The Docker image for Logstash 5.2.1 can be retrieved with the following command:

```
docker pull docker.elastic.co/logstash/logstash:5.2.1
```

Configuring Logstash for Docker

Logstash differentiates between two types of configuration: [Settings and Pipeline Configuration](#).

Pipeline Configuration

It is essential to place your pipeline configuration where it can be found by Logstash. By default, the container will look in `/usr/share/logstash/pipeline/` for pipeline configuration files.

In this example we use a bind-mounted volume to provide the configuration via the `docker run` command:

```
docker run --rm -it -v ~/pipeline:/usr/share/logstash/pipeline/ docker.elastic.co/logstash/logstash:5.2.1
```

Every file in the host directory `~/pipeline/` will then be parsed by Logstash as pipeline configuration.

If you don't provide configuration to Logstash, it will run with a minimal config that listens for messages from the [Beats input plugin](#) and echoes any that are received to `stdout`. In this case, the startup logs will be similar to the following:

```
Sending Logstash logs to /usr/share/logstash/logs which is now configured via log4j2.properties.  
[2016-10-26T05:11:34,992] [INFO ] [logstash.inputs.beats      ] Beats inputs:  
Starting input listener {:address=>"0.0.0.0:5044"}  
[2016-10-26T05:11:35,068] [INFO ] [logstash.pipeline          ] Starting pipeline  
{"id":>"main", "pipeline.workers":>4, "pipeline.batch.size":>125,  
"pipeline.batch.delay":>5, "pipeline.max_inflight":>500}  
[2016-10-26T05:11:35,078] [INFO ] [org.logstash.beats.Server] Starting server  
on port: 5044  
[2016-10-26T05:11:35,078] [INFO ] [logstash.pipeline          ] Pipeline main  
started  
[2016-10-26T05:11:35,105] [INFO ] [logstash.agent           ] Successfully  
started Logstash API endpoint {:port=>9600}
```

This is the default configuration for the image, defined in `/usr/share/logstash/pipeline/logstash.conf`. If this is the behaviour that you are observing, ensure that your pipeline configuration is being picked up correctly, and that you are replacing either `logstash.conf` or the entire pipeline directory.

Settings Files

Settings files can also be provided through bind-mounts. Logstash expects to find them at `/usr/share/logstash/config/`.

It's possible to provide an entire directory containing all needed files:

```
docker run --rm -it -v ~/settings:/usr/share/logstash/config/  
docker.elastic.co/logstash/logstash:5.2.1
```

Alternatively, a single file can be mounted:

```
docker run --rm -it -v  
~/settings/logstash.yml:/usr/share/logstash/config/logstash.yml  
docker.elastic.co/logstash/logstash:5.2.1
```

NOTE: Bind-mounted configuration files will retain the same permissions and ownership within the container that they have on the host system. Be sure to set permissions such that the files will be readable and, ideally, not writeable by the container's `logstash` user (UID 1000).

Custom Images

Bind-mounted configuration is not the only option, naturally. If you prefer the *Immutable Infrastructure* approach, you can prepare a custom image containing your configuration by using a Dockerfile like this one:

```
FROM docker.elastic.co/logstash/logstash:5.2.1  
RUN rm -f /usr/share/logstash/pipeline/logstash.conf  
ADD pipeline/ /usr/share/logstash/pipeline/  
ADD config/ /usr/share/logstash/config/
```

Be sure to replace or delete `logstash.conf` in your custom image, so that you don't retain the example config from the base image.

Logging Configuration

Under Docker, Logstash logs go to standard output by default. To change this behaviour, use any of the techniques above to replace the file at

`/usr/share/logstash/config/log4j2.properties`.

[Command-Line Flags »](#)

Settings File

You can set options in the Logstash settings file, `logstash.yml`, to control Logstash execution. For example, you can specify pipeline settings, the location of configuration files, logging options, and other settings. Most of the settings in the `logstash.yml` file are also available as [command-line flags](#) when you run Logstash. Any flags that you set at the command line override the corresponding settings in the `logstash.yml` file.

The `logstash.yml` file, which is written in [YAML](#), is located in `LOGSTASH_HOME/config`. You can specify settings in hierarchical form or use flat keys. For example, to use hierarchical form to set the pipeline batch size and batch delay, you specify:

```
pipeline:
  batch:
    size: 125
    delay: 5
```

To express the same values as flat keys, you specify:

```
pipeline.batch.size: 125
pipeline.batch.delay: 5
```

The `logstash.yml` file includes the following settings:

Setting	Description	Default value
<code>node.name</code>	A descriptive name for the node.	Machine's hostname
<code>path.data</code>	The directory that Logstash and its plugins use for any persistent needs.	<code>LOGSTASH_HOME/data</code>
<code>pipeline.workers</code>	The number of workers that will, in parallel, execute the filter and output stages of the pipeline. If you find that events are backing up, or that the CPU is not saturated, consider increasing this number to better utilize machine processing power.	Number of the host's CPU cores
<code>pipeline.output.workers</code>	The number of workers to use per output plugin instance.	1
<code>pipeline.batch.size</code>	The maximum number of events an individual worker thread will collect from inputs before attempting to execute its filters and	125

Setting	Description	Default value
	outputs. Larger batch sizes are generally more efficient, but come at the cost of increased memory overhead. You may have to increase the JVM heap size by setting the <code>LS_HEAP_SIZE</code> variable to effectively use the option.	
<code>pipeline.batch.delay</code>	When creating pipeline event batches, how long in milliseconds to wait before dispatching an undersized batch to filters and workers.	5
<code>pipeline.unsafe_shutdown</code>	When set to <code>true</code> , forces Logstash to exit during shutdown even if there are still inflight events in memory. By default, Logstash will refuse to quit until all received events have been pushed to the outputs. Enabling this option can lead to data loss during shutdown.	false
<code>path.config</code>	The path to the Logstash config for the main pipeline. If you specify a directory or wildcard, config files are read from the directory in alphabetical order.	Platform-specific. See Logstash Directory Layout .
<code>config.string</code>	A string that contains the pipeline configuration to use for the main pipeline. Use the same syntax as the config file.	None
<code>config.test_and_exit</code>	When set to <code>true</code> , checks that the configuration is valid and then exits. Note that grok patterns are not checked for correctness with this setting. Logstash can read multiple config files from a directory. If you combine this setting with <code>log.level: debug</code> , Logstash will log the combined config file, annotating each config block with the source file it came from.	false

Setting	Description	Default value
config.reload.automatic	When set to <code>true</code> , periodically checks if the configuration has changed and reloads the configuration whenever it is changed. This can also be triggered manually through the <code>SIGHUP</code> signal.	false
config.reload.interval	How often in seconds Logstash checks the config files for changes.	3
config.debug	When set to <code>true</code> , shows the fully compiled configuration as a debug log message. You must also set <code>log.level: debug</code> . WARNING: The log message will include any <code>password</code> options passed to plugin configs as plaintext, and may result in plaintext passwords appearing in your logs!	false
queue.type	The internal queuing model to use for event buffering. Specify <code>memory</code> for legacy in-memory based queuing, or <code>persisted</code> for disk-based ACKed queueing (persistent queues).	memory
path.queue	The directory path where the data files will be stored when persistent queues are enabled (<code>queue.type: persisted</code>).	path.data/queue
queue.page_capacity	The size of the page data files used when persistent queues are enabled (<code>queue.type: persisted</code>). The queue data consists of append-only data files separated into pages.	250mb
queue.max_events	The maximum number of unread events in the queue when persistent queues are enabled (<code>queue.type: persisted</code>).	0 (unlimited)
queue.max_bytes	The total capacity of the queue in number of bytes. Make sure the capacity of your disk drive is greater than the value you specify here. If both <code>queue.max_events</code>	1024mb (1g)

Setting	Description	Default value
	and <code>queue.max_bytes</code> are specified, Logstash uses whichever criteria is reached first.	
<code>queue.checkpoint.acks</code>	The maximum number of ACKed events before forcing a checkpoint when persistent queues are enabled (<code>queue.type: persisted</code>). Specify <code>queue.checkpoint.acks: 0</code> to set this value to unlimited.	1024
<code>queue.checkpoint.writes</code>	The maximum number of written events before forcing a checkpoint when persistent queues are enabled (<code>queue.type: persisted</code>). Specify <code>queue.checkpoint.writes: 0</code> to set this value to unlimited.	1024
<code>queue.checkpoint.interval</code>	The interval in milliseconds when a checkpoint is forced on the head page when persistent queues are enabled (<code>queue.type: persisted</code>). Specify <code>queue.checkpoint.interval: 0</code> for no periodic checkpoint.	1000
<code>http.host</code>	The bind address for the metrics REST endpoint.	"127.0.0.1"
<code>http.port</code>	The bind port for the metrics REST endpoint.	9600
<code>log.level</code>	The log level. Valid options are: <ul style="list-style-type: none"> • <code>fatal</code> • <code>error</code> • <code>warn</code> • <code>info</code> • <code>debug</code> • <code>trace</code> 	info
<code>log.format</code>	The log format. Set to <code>json</code> to log in JSON format, or <code>plain</code> to use <code>Object#.inspect</code> .	plain
<code>path.logs</code>	The directory where Logstash will write its log to.	'LOGSTASH_HOME/logs'

Setting	Description	Default value
path.plugins	Where to find custom plugins. You can specify this setting multiple times to include multiple paths. Plugins are expected to be in a specific directory hierarchy: PATH/logstash/TYPE/NAME.rb where TYPE is inputs, filters, outputs, or codecs, and NAME is the name of the plugin.	Platform-specific. See Logstash Directory Layout .

[Logging »](#)

Command-Line Flags

Logstash has the following flags. You can use the `--help` flag to display this information.

Instead of specifying options at the command line, we recommend that you control Logstash execution by specifying options in the Logstash [settings file](#). Using a settings file makes it easier for you to specify multiple options, and it provides you with a single, versionable file that you can use to start up Logstash consistently for each run.

Any flags that you set at the command line override the corresponding settings in the Logstash [settings file](#).

`--node.name NAME`

Specify the name of this Logstash instance. If no value is given it will default to the current hostname.

`-f, --path.config CONFIG_PATH`

Load the Logstash config from a specific file or directory. If a directory is given, all files in that directory will be concatenated in lexicographical order and then parsed as a single config file. Specifying this flag multiple times is not supported. If you specify this flag multiple times, Logstash uses the last occurrence (for example, `-f foo -f bar` is the same as `-f bar`).

You can specify wildcards ([globs](#)) and any matched files will be loaded in the order described above. For example, you can use the wildcard feature to load specific files by name:

```
bin/logstash --debug -f '/tmp/{one,two,three}'
```

With this command, Logstash concatenates three config files, `/tmp/one`, `/tmp/two`, and `/tmp/three`, and parses them into a single config.

`-e, --config.string CONFIG_STRING`

Use the given string as the configuration data. Same syntax as the config file. If no input is specified, then the following is used as the default input: `input { stdin { type => stdin } }` and if no output is specified, then the following is used as the default output: `output { stdout { codec => rubydebug } }`. If you wish to use both defaults, please use the empty string for the `-e` flag. The default is nil.

`-w, --pipeline.workers COUNT`

Sets the number of pipeline workers to run. This option sets the number of workers that will, in parallel, execute the filter and output stages of the pipeline. If you find that events are backing up, or that the CPU is not saturated, consider increasing this number to better utilize machine processing power. The default is the number of the host's CPU cores.

`-b, --pipeline.batch.size SIZE`

Size of batches the pipeline is to work in. This option defines the maximum number of events an individual worker thread will collect from inputs before attempting to execute its filters and outputs. The default is 125 events. Larger batch sizes are generally more efficient, but come at the cost of increased memory overhead. You may have to increase the JVM heap size by setting the `LS_HEAP_SIZE` variable to effectively use the option.

-u, --pipeline.batch.delay DELAY_IN_MS

When creating pipeline batches, how long to wait while polling for the next event. This option defines how long in milliseconds to wait while polling for the next event before dispatching an undersized batch to filters and workers. The default is 250ms.

--pipeline.unsafe_shutdown

Force Logstash to exit during shutdown even if there are still inflight events in memory. By default, Logstash will refuse to quit until all received events have been pushed to the outputs. Enabling this option can lead to data loss during shutdown.

--path.data PATH

This should point to a writable directory. Logstash will use this directory whenever it needs to store data. Plugins will also have access to this path. The default is the `data` directory under Logstash home.

-p, --path.plugins PATH

A path of where to find custom plugins. This flag can be given multiple times to include multiple paths. Plugins are expected to be in a specific directory hierarchy:

`PATH/logstash/TYPE/NAME.rb` where `TYPE` is `inputs`, `filters`, `outputs`, or `codecs`, and `NAME` is the name of the plugin.

-l, --path.logs PATH

Directory to write Logstash internal logs to.

--log.level LEVEL

Set the log level for Logstash. Possible values are:

- `fatal`: log very severe error messages that will usually be followed by the application aborting
- `error`: log errors
- `warn`: log warnings
- `info`: log verbose info (this is the default)
- `debug`: log debugging info (for developers)
- `trace`: log finer-grained messages beyond debugging info

--config.debug

Show the fully compiled configuration as a debug log message (you must also have `--log.level=debug` enabled). **WARNING:** The log message will include any `password` options passed to plugin configs as plaintext, and may result in plaintext passwords appearing in your logs!

-i, --interactive SHELL

Drop to shell instead of running as normal. Valid shells are "irb" and "pry".

-v, --version

Emit the version of Logstash and its friends, then exit.

-t, --config.test_and_exit

Check configuration for valid syntax and then exit. Note that grok patterns are not checked for correctness with this flag. Logstash can read multiple config files from a

directory. If you combine this flag with `--log.level=debug`, Logstash will log the combined config file, annotating each config block with the source file it came from.

-r, --config.reload.automatic

Monitor configuration changes and reload whenever the configuration is changed.

NOTE: Use SIGHUP to manually reload the config. The default is false.

--config.reload.interval RELOAD_INTERVAL

How frequently to poll the configuration location for changes, in seconds. The default is every 3 seconds.

--http.host HTTP_HOST

Web API binding host. This option specifies the bind address for the metrics REST endpoint. The default is "127.0.0.1".

--http.port HTTP_PORT

Web API http port. This option specifies the bind port for the metrics REST endpoint. The default is 9600-9700. This setting accepts a range of the format 9600-9700. Logstash will pick up the first available port.

--log.format FORMAT

Specify if Logstash should write its own logs in JSON form (one event per line) or in plain text (using Ruby's Object#inspect). The default is "plain".

--path.settings SETTINGS_DIR

Set the directory containing the `logstash.yml` [settings file](#) as well as the log4j logging configuration. This can also be set through the LS_SETTINGS_DIR environment variable. The default is the `config` directory under Logstash home.

-h, --help

Print help

Logging

Logstash emits internal logs during its operation, which are placed in `LS_HOME/logs` (or `/var/log/logstash` for DEB/RPM). The default logging level is `INFO`. Logstash's logging framework is based on [Log4j 2 framework](#), and much of its functionality is exposed directly to users.

When debugging problems, particularly problems with plugins, it can be helpful to increase the logging level to `DEBUG` to emit more verbose messages. Previously, you could only set a log level that applied to the entire Logstash product. Starting with 5.0, you can configure logging for a particular subsystem in Logstash. For example, if you are debugging issues with Elasticsearch Output, you can increase log levels just for that component. This way you can reduce noise due to excessive logging and focus on the problem area effectively.

Log file location

You can specify the log file location using `--path.logs` setting.

Log4j 2 Configuration

Logstash ships with a `log4j2.properties` file with out-of-the-box settings. You can modify this file directly to change the rotation policy, type, and other [log4j2 configuration](#). You must restart Logstash to apply any changes that you make to this file.

Slowlog

Slow-log for Logstash adds the ability to log when a specific event takes an abnormal amount of time to make its way through the pipeline. Just like the normal application log, you can find slow-logs in your `--path.logs` directory. Slowlog is configured in the `logstash.yml` settings file with the following options:

```
slowlog.threshold.warn (default: -1)
slowlog.threshold.info (default: -1)
slowlog.threshold.debug (default: -1)
slowlog.threshold.trace (default: -1)
```

By default, these values are set to `-1nanos` to represent an infinite threshold where no slowlog will be invoked. These `slowlog.threshold` fields are configured using a time-value format which enables a wide range of trigger intervals. The positive numeric ranges can be specified using the following time units: `nanos` (nanoseconds), `micros` (microseconds), `ms` (milliseconds), `s` (second), `m` (minute), `h` (hour), `d` (day).

Here is an example:

```
slowlog.threshold.warn: 2s
slowlog.threshold.info: 1s
```

```
slowlog.threshold.debug: 500ms
slowlog.threshold.trace: 100ms
```

In the above configuration, events that take longer than two seconds to be processed within a filter will be logged. The logs will include the full event and filter configuration that are responsible for the slowness.

Logging APIs

You could modify the `log4j2.properties` file and restart your Logstash, but that is both tedious and leads to unnecessary downtime. Instead, you can dynamically update logging levels through the logging API. These settings are effective immediately and do not need a restart. To update logging levels, take the subsystem/module you are interested in and prepend `logger.` to it. For example:

```
PUT /_node/logging
{
  "logger.logstash.outputs.elasticsearch" : "DEBUG"
}
```

While this setting is in effect, Logstash will begin to emit DEBUG-level logs for *all* the Elasticsearch outputs specified in your configuration. Please note this new setting is transient and will not survive a restart.

To retrieve a list of logging subsystems available at runtime, you can do a `GET` request to `_node/logging`

```
GET /_node/logging?pretty
```

Example response:

```
{
  ...
  "loggers" : {
    "logstash.registry" : "WARN",
    "logstash.instrument.periodicpoller.os" : "WARN",
    "logstash.instrument.collector" : "WARN",
    "logstash.runner" : "WARN",
    "logstash.inputs.stdin" : "WARN",
    "logstash.outputs.stdout" : "WARN",
    "logstash.agent" : "WARN",
    "logstash.api.service" : "WARN",
    "logstash.instrument.periodicpoller.jvm" : "WARN",
    "logstash.pipeline" : "WARN",
    "logstash.codecs.line" : "WARN"
  }
}
```

Persistent Queues

WARNING!! This functionality is in beta and is subject to change. Deployment in production is at your own risk.

By default, Logstash uses in-memory bounded queues between pipeline stages (inputs → pipeline workers) to buffer events. The size of these in-memory queues is fixed and not configurable. If Logstash experiences a temporary machine failure, the contents of the in-memory queue will be lost. Temporary machine failures are scenarios where Logstash or its host machine are terminated abnormally but are capable of being restarted.

In order to protect against data loss during abnormal termination, Logstash has a persistent queue feature which will store the message queue on disk. Persistent queues provide durability of data within Logstash.

Persistent queues are also useful for Logstash deployments that need large buffers. Instead of deploying and managing a message broker, such as Redis, RabbitMQ, or Apache Kafka, to facilitate a buffered publish-subscriber model, you can enable persistent queues to buffer events on disk and remove the message broker.

In summary, the two benefits of enabling persistent queues are as follows:

- Provides protection from in-flight message loss when the Logstash process is abnormally terminated.
- Absorbs bursts of events without needing an external buffering mechanism like Redis or Apache Kafka.

Limitations of Persistent Queues

The following are problems not solved by the persistent queue feature:

- Input plugins that do not use a request-response protocol cannot be protected from data loss. For example: tcp, udp, zeromq push+pull, and many other inputs do not have a mechanism to acknowledge receipt to the sender. Plugins such as beats and http, which **do** have acknowledgement capability, are well protected by this queue.
- It does not handle permanent machine failures such as disk corruption, disk failure, and machine loss. The data persisted to disk is not replicated.

How Persistent Queues Work

The queue sits between the input and filter stages in the same process:

input → queue → filter + output

When an input has events ready to process, it writes them to the queue. When the write to the queue is successful, the input can send an acknowledgement to its data source.

When processing events from the queue, Logstash acknowledges events as completed, within the queue, only after filters and outputs have completed. The queue keeps a record of events that have been processed by the pipeline. An event is recorded as processed (in this document, called "acknowledged" or "ACKed") if, and only if, the event has been processed completely by the Logstash pipeline.

What does acknowledged mean? This means the event has been handled by all configured filters and outputs. For example, if you have only one output, Elasticsearch, an event is ACKed when the Elasticsearch output has successfully sent this event to Elasticsearch.

During a normal shutdown (**CTRL+C** or **SIGTERM**), Logstash will stop reading from the queue and will finish processing the in-flight events being processed by the filters and outputs. Upon restart, Logstash will resume processing the events in the persistent queue as well as accepting new events from inputs.

If Logstash is abnormally terminated, any in-flight events will not have been ACKed and will be reprocessed by filters and outputs when Logstash is restarted. Logstash processes events in batches, so it is possible that for any given batch, some of that batch may have been successfully completed, but not recorded as ACKed, when an abnormal termination occurs.

For more details specific behaviors of queue writes and acknowledgement, see [Controlling Durability](#).

Configuring Persistent Queues

To configure persistent queues, you can specify the following options in the Logstash [settings file](#):

- `queue.type`: Specify `persisted` to enable persistent queues. By default, persistent queues are disabled (default: `queue.type: memory`).
- `path.queue`: The directory path where the data files will be stored. By default, the files are stored in `path.data/queue`.
- `queue.page_capacity`: The maximum size of a queue page in bytes. The queue data consists of append-only files called "pages". The default size is 250mb. Changing this value is unlikely to have performance benefits.
- `queue.max_events`: The maximum number of events that are allowed in the queue. The default is 0 (unlimited). This value is used internally for the Logstash test suite.
- `queue.max_bytes`: The total capacity of the queue in number of bytes. The default is 1024mb (1gb). Make sure the capacity of your disk drive is greater than the value you specify here.

If both `queue.max_events` and `queue.max_bytes` are specified, Logstash uses whichever criteria is reached first. See [Handling Back Pressure](#) for behavior when these queue limits are reached.

You can also specify options that control when the checkpoint file gets updated (`queue.checkpoint.acks`, `queue.checkpoint.writes`). See [Controlling Durability](#).

Example configuration:

```
queue.type: persisted  
queue.max_bytes: 4gb
```

Handling Back Pressure

When the queue is full, Logstash puts back pressure on the inputs to stall data flowing into Logstash. This mechanism helps Logstash control the rate of data flow at the input stage without overwhelming outputs like Elasticsearch.

Use `queue.max_bytes` setting to configure the total capacity of the queue on disk. The following example sets the total capacity of the queue to 8gb:

```
queue.type: persisted  
queue.max_bytes: 8gb
```

With these settings specified, Logstash will buffer events on disk until the size of the queue reaches 8gb. When the queue is full of unACKed events, and the size limit has been reached, Logstash will no longer accept new events.

Each input handles back pressure independently. For example, when the [beats](#) input encounters back pressure, it no longer accepts new connections and waits until the persistent queue has space to accept more events. After the filter and output stages finish processing existing events in the queue and ACKs them, Logstash automatically starts accepting new events.

Controlling Durability

Durability is a property of storage writes that ensures data will be available after it's written.

When the persistent queue feature is enabled, Logstash will store events on disk. Logstash commits to disk in a mechanism called checkpointing.

To discuss durability, we need to introduce a few details about how the persistent queue is implemented.

First, the queue itself is a set of pages. There are two kinds of pages: head pages and tail pages. The head page is where new events are written. There is only one head page. When the head page is of a certain size (see `queue.page_capacity`), it becomes a tail page, and a new head page is created. Tail pages are immutable, and the head page is append-only. Second, the queue

records details about itself (pages, acknowledgements, etc) in a separate file called a checkpoint file.

When recording a checkpoint, Logstash will:

- Call fsync on the head page.
- Atomically write to disk the current state of the queue.

The following settings are available to let you tune durability:

- `queue.checkpoint.writes`: Logstash will checkpoint after this many writes into the queue. Currently, one event counts as one write, but this may change in future releases.
- `queue.checkpoint.acks`: Logstash will checkpoint after this many events are acknowledged. This configuration controls the durability at the processing (filter + output) part of Logstash.

Disk writes have a resource cost. Tuning the above values higher or lower will trade durability for performance. For instance, if you want the strongest durability for all input events, you can set `queue.checkpoint.writes: 1`.

The process of checkpointing is atomic, which means any update to the file is saved if successful.

If Logstash is terminated, or if there is a hardware level failure, any data that is buffered in the persistent queue, but not yet checkpointed, is lost. To avoid this possibility, you can set `queue.checkpoint.writes: 1`, but keep in mind that this setting can severely impact performance.

Disk Garbage Collection

On disk, the queue is stored as a set of pages where each page is one file. Each page can be at most `queue.page_capacity` in size. Pages are deleted (garbage collected) after all events in that page have been ACKed. If an older page has at least one event that is not yet ACKed, that entire page will remain on disk until all events in that page are successfully processed. Each page containing unprocessed events will count against the `queue.max_bytes` byte size.

Shutting Down Logstash

When you attempt to shut down a running Logstash instance, Logstash performs several steps before it can safely shut down. It must:

- Stop all input, filter and output plugins
- Process all in-flight events
- Terminate the Logstash process

The following conditions affect the shutdown process:

- An input plugin receiving data at a slow pace.
- A slow filter, like a Ruby filter executing `sleep(10000)` or an Elasticsearch filter that is executing a very heavy query.
- A disconnected output plugin that is waiting to reconnect to flush in-flight events.

These situations make the duration and success of the shutdown process unpredictable.

Logstash has a stall detection mechanism that analyzes the behavior of the pipeline and plugins during shutdown. This mechanism produces periodic information about the count of inflight events in internal queues and a list of busy worker threads.

To enable Logstash to forcibly terminate in the case of a stalled shutdown, use the `--pipeline.unsafe_shutdown` flag when you start Logstash.

WARNING!! Unsafe shutdowns, force-kills of the Logstash process, or crashes of the Logstash process for any other reason may result in data loss (unless you've enabled Logstash to use [persistent queues](#)). Shut down Logstash safely whenever possible.

Stall Detection Example

In this example, slow filter execution prevents the pipeline from clean shutdown. By starting Logstash with the `--pipeline.unsafe_shutdown` flag, quitting with **Ctrl+C** results in an eventual shutdown that loses 20 events.

```
bin/logstash -e 'input { generator { } } filter { ruby { code => "sleep 10000" } } output { stdout { codec => dots } }' -w 1 --pipeline.unsafe_shutdown
Pipeline main started
^SIGINT received. Shutting down the agent. {:level=>:warn}
stopping pipeline {:id=>"main", :level=>:warn}
Received shutdown signal, but pipeline is still waiting for in-flight events to be processed. Sending another ^C will force quit Logstash, but this may cause
data loss. {:level=>:warn}
```

Logstash 5.2 Configuration Guide

```
{"inflight_count":>125, "stalling_thread_info":>{ ["LogStash::Filters::Ruby", {"code":>"sleep 10000"}]}=>[{"thread_id":>19, "name":>"[main]>worker0", "current_call":>"(ruby filter code):1:in `sleep'"}]} {:level=>:warn}  
The shutdown process appears to be stalled due to busy or blocked plugins.  
Check the logs for more information. {:level=>:error}  
{"inflight_count":>125, "stalling_thread_info":>{ ["LogStash::Filters::Ruby", {"code":>"sleep 10000"}]}=>[{"thread_id":>19, "name":>"[main]>worker0", "current_call":>"(ruby filter code):1:in `sleep'"}]} {:level=>:warn}  
{"inflight_count":>125, "stalling_thread_info":>{ ["LogStash::Filters::Ruby", {"code":>"sleep 10000"}]}=>[{"thread_id":>19, "name":>"[main]>worker0", "current_call":>"(ruby filter code):1:in `sleep'"}]} {:level=>:warn}  
Forcefully quitting logstash.. {:level=>:fatal}
```

When `--pipeline.unsafe_shutdown` isn't enabled, Logstash continues to run and produce these reports periodically.

Breaking changes

This section discusses the changes that you need to be aware of when migrating your application to Logstash 5.0 from the previous major release of Logstash (2.x).

Changes in Logstash Core

These changes can impact any instance of Logstash and are plugin agnostic, but only if you are using the features that are impacted.

Application Settings

IMPORTANT: Logstash 5.0 introduces a new way to [configure application settings](#) for Logstash through a `logstash.yml` file.

This file is typically located in `${LS_HOME} /config`, or `/etc/logstash` when installed via packages. Logstash will not be able to start without this file, so please make sure to pass in `--path.settings` if you are starting Logstash manually after installing it via a package (RPM, DEB).

```
bin/logstash --path.settings /path/to/logstash.yml
```

URL Changes for DEB/RPM Packages

The previous `packages.elastic.co` URL has been altered to `artifacts.elastic.co`. Ensure you update your repository files before running the upgrade process, or your operating system may not see the new packages.

Release Packages

When Logstash 5.0 is installed via DEB or RPM packages, it now uses `/usr/share/logstash` to install binaries. Previously it used to install in `/opt/logstash` directory. This change was done to make the user experience consistent with other products in the Elastic Stack.

	DEB	RPM
Logstash 2.x	<code>/opt/logstash</code>	<code>/opt/logstash</code>
Logstash 5.0	<code>/usr/share/logstash</code>	<code>/usr/share/logstash</code>

A complete directory layout is described in [Logstash Directory Layout](#). This will likely impact any scripts that you may have written to support installing or manipulating Logstash, such as via Puppet.

Default Logging Level

The default log severity level changed to `INFO` instead of `WARN` to match Elasticsearch. Existing logs (in core and plugins) were too noisy at the `INFO` level, so we audited our log messages and switched some of them to `DEBUG` level.

You can use the new `logstash.yml` file to configure the `log.level` setting or continue to pass the new `--log.level` command line flag.

```
bin/logstash --log.level warn
```

Plugin Manager Renamed

`bin/plugin` has been renamed to `bin/logstash-plugin`. This occurred in Logstash 2.3 and it was mainly prevent `PATH` being polluted when other components of the Elastic Stack are installed on the same machine. Also, this provides a foundation for future change which will allow Elastic Stack packs to be installed via this script.

Logstash 5.0 also adds a `remove` option, which is an alias for the now-deprecated `uninstall` option.

As with earlier releases, the updated script allows both online and offline plugin installation. For example, to install a plugin named “`my-plugin`”, it’s as simple as running:

```
bin/logstash-plugin install my-plugin
```

Similar to the package changes, this is likely to impact and scripts that have been written to follow Logstash installations.

Like earlier releases of Logstash, most plugins are bundled directly with Logstash, so no additional action is required while upgrading from earlier Logstash releases. However, if you are attempting to install a non-bundled plugin, then make sure that it supports Logstash 5.0 before upgrading!

Logstash with All Plugins Download

The Logstash All Plugins download option has been removed. For users previously using this option as a convenience for offline plugin management purposes (air-gapped environments), please see the [Offline Plugin Management](#) documentation page.

There were 17 plugins removed from 5.0 default bundle. These plugins can still be installed manually for use:

- `logstash-codec-oldlogstashjson`
- `logstash-input-eventlog`
- `logstash-input-log4j`
- `logstash-input-zeromq`

- logstash-filter-anonymize
- logstash-filter-checksum
- logstash-filter-multiline
- logstash-output-email
- logstash-output-exec
- logstash-output-ganglia
- logstash-output-gelf
- logstash-output-hipchat
- logstash-output-juggernaut
- logstash-output-lumberjack
- logstash-output-nagios_nsca
- logstash-output-opentsdb
- logstash-output-zeromq

Command Line Interface

Some CLI Options changed in Logstash 5.0. If you were using the “long form” of the [options](#), then this will impact the way that you launch Logstash. They were changed to match the `logstash.yml` format used to simplify future setup, as well as behave in the same way as other products in the Elastic Stack. For example, here’s two before-and-after examples. In Logstash 2.x, you may have run something:

```
bin/logstash --config my.conf --pipeline-workers 8
```

```
bin/logstash -f my.conf -w 8
```

Long form options `config` and `pipeline-workers` are used here.

Short form options `f` and `w` (aliases for the former) are used here.

But, in Logstash 5.0, this becomes:

```
bin/logstash --path.config my.conf --pipeline.workers 8
```

```
bin/logstash -f my.conf -w 8
```

Long form options are changed to reflect the new options.

Short form options are unchanged.

NOTE: None of the short form options have changed!

RSpec testing script

The `rspec` script is no longer bundled with Logstash release artifacts. This script has been used previously to run unit tests for validating Logstash configurations. While this was useful to some users, this mechanism assumed that Logstash users were familiar with the RSpec framework, which is a Ruby testing framework.

Breaking Changes in Plugins

Elasticsearch Output workers Setting Removed

Starting with Logstash 5.0, the `workers` setting in the Elasticsearch output plugin is no longer supported. Pipelines that specify this setting will no longer start up. You need to specify the `pipeline.workers` setting at the pipeline level instead. For more information about setting `pipeline.workers`, see [Settings File](#).

Elasticsearch Output Index Template

The index template for Elasticsearch 5.0 has been changed to reflect [Elasticsearch's mapping changes](#). Most importantly, the subfield for string multi-fields has changed from `.raw` to `.keyword` to match Elasticsearch's default behavior. The impact of this change to various user groups is detailed below:

- New Logstash 5.0 and Elasticsearch 5.0 users: Multi-fields (often called sub-fields) use `.keyword` from the outset. In Kibana, you can use `my_field.keyword` to perform aggregations against text-based fields, in the same way that it used to be `my_field.raw`.
- Existing users with custom templates: Using a custom template means that you control the template completely, and our template changes do not impact you.
- Existing users with default template: Logstash does not force you to upgrade templates if one already exists. If you intend to move to the new template and want to use `.keyword`, you will most likely want to reindex existing data so that it also uses the `.keyword` field, unless you are able to transition from `.raw` to `.keyword`. Elasticsearch's [reindexing API](#) can help move your data from using `.raw` subfields to `.keyword`, thereby avoiding any transition time. You *can* use a custom template to get both `.raw` and `.keyword` so that you can wait until all `.raw` data has stopped existing before transitioning to only using `.keyword`; this will waste some storage space and memory, but it does help users to avoid having to relearn operations.

Plugin Versions

Logstash is unique amongst the Elastic Stack with respect to its plugins. Unlike Elasticsearch and Kibana, which both require plugins to be targeted to a specific release, Logstash's plugin ecosystem provides more flexibility so that it can support outside ecosystems *within the same release*. Unfortunately, that flexibility can cause issues when handling upgrades.

Non-standard plugins must always be checked for compatibility, but some bundled plugins are upgraded in order to remain compatible with the tools or frameworks that they use for

communication. For example, the [Kafka Input](#) and [Kafka Output](#) plugins serve as a primary example of such compatibility changes. The latest version of the Kafka plugins is only compatible with Kafka 0.10, but as the compatibility matrices show: earlier plugin versions are required for earlier versions of Kafka (e.g., Kafka 0.9).

Automatic upgrades generally lead to improved features and support, but network layer changes like those above may make part of your architecture incompatible. You should always test your Logstash configurations in a test environment before deploying to production, which would catch these kinds of issues. If you do face such an issue, then you should also check the specific plugin's page to see how to get a compatible, older plugin version if necessary.

For example, if you upgrade to Logstash 5.0, but you want to run against Kafka 0.9, then you need to remove the bundled plugin(s) that only work with Kafka 0.10 and replace them:

```
bin/logstash-plugin remove logstash-input-kafka  
bin/logstash-plugin remove logstash-output-kafka  
bin/logstash-plugin install --version 4.0.0 logstash-input-kafka  
bin/logstash-plugin install --version 4.0.1 logstash-output-kafka
```

The version numbers were found by checking the compatibility matrix for the individual plugins.

[Kafka Input Configuration Changes](#)

As described in the section [above](#), the Kafka plugin has been updated to bring in new consumer features. In addition, to the plugin being incompatible with 0.8.x version of the Kafka broker, *most* of the config options have been changed to match the new consumer configurations from the Kafka Java consumer. Here's a list of important config options that have changed:

- `topic_id` is renamed to `topics` and accepts an array of topics to consume from.
- `zk_connect` has been dropped; you should use `bootstrap_servers`. There is no need for the consumer to go through ZooKeeper.
- `consumer_id` is renamed to `client_id`.

We recommend users of the Kafka plugin to check the documentation for the latest [config options](#).

[File Input](#)

The [File Input](#) SinceDB file is now saved at `<path.data>/plugins/inputs/file` location, where `path.data` is the path defined in the new `logstash.yml` file.

	Default <code>sincedb_path</code>
Logstash 2.x	<code>\$HOME/.sincedb*</code>
Logstash 5.0	<code><path.data>/plugins/inputs/file</code>

If you have manually specified `sincedb_path` as part of the configuration, this change will not affect you. If you are moving from Logstash 2.x to Logstash 5.0, and you would like to use the existing SinceDB file, then it must be copied over to `path.data` manually to use the save state (or the path needs to be changed to point to it).

[GeoIP Filter](#)

The GeoIP filter has been updated to use MaxMind's GeoIP2 database. Previous GeoIP version is now considered legacy by MaxMind. As a result of this, `.dat` version files are no longer supported, and only `.mddb` format is supported. The new database will not include ASN data in the basic free database file.

Previously, when the filter encountered an IP address for which there were no results in the database, the event would just pass through the filter without modification. It will now add a `_geoip_lookup_failure` tag to the event which will allow for some subsequent stage of the pipeline to identify those events and perform some other operation. To simply get the same behavior as the earlier versions, just add a filter conditional on that tag which then drops the tag from the event.

Ruby Filter and Custom Plugin Developers

With the migration to the new [Event API](#), we have changed how you can access internal data compared to previous release. The `event` object no longer returns a reference to the data. Instead, it returns a copy. This might change how you perform manipulation of your data, especially when working with nested hashes. When working with nested hashes, it's recommended that you use the [field reference syntax](#) instead of using multiple square brackets.

As part of this change, Logstash has introduced new Getter/Setter APIs for accessing information in the `event` object.

Examples:

Prior to Logstash 5.0, you may have used Ruby filters like so:

```
filter {
  ruby {
    code => "event['name'] = 'Logstash'"
  }
  ruby {
    code => "event['product']['version'] = event['major'] + '.' +
event['minor']"
  }
}
```

The above syntax, which uses the `event` object as a reference, is no longer supported in Logstash 5.0. Fortunately, the change to make it work is very simple:

```
filter {
```

Logstash 5.2 Configuration Guide

```
ruby {
  code => "event.set('name', 'Logstash')"
}
ruby {
  code => "event.set('[product][version]', event.get('major') + '.' +
event.get('minor'))"
}
}
```

NOTE: Moving from the old syntax to the new syntax, it can be easy to miss that `['product']['version']` became `'[product][version]'`. The quotes moved from inside of the square brackets to outside of the square brackets!

The [Event API](#) documentation describes the available syntax in great detail.

Upgrading Logstash

IMPORTANT: Before upgrading Logstash:

- Consult the [breaking changes](#) docs.
- Test upgrades in a development environment before upgrading your production cluster.

If you are installing Logstash with other components in the Elastic Stack, also see the [Elastic Stack installation and upgrade documentation](#).

See the following topics for information about upgrading Logstash:

- [Upgrading Using Package Managers](#)
- [Upgrading Using a Direct Download](#)
- [Upgrading Logstash to 5.0](#)

[Upgrading Using a Direct Download »](#)

Upgrading Using Package Managers

This procedure uses [package managers](#) to upgrade Logstash.

1. Shut down your Logstash pipeline, including any inputs that send events to Logstash.
2. Using the directions in the *Package Repositories* section, update your repository links to point to the 5.x repositories instead of the previous version.
3. Run the `apt-get upgrade logstash` or `yum update logstash` command as appropriate for your operating system.
4. Test your configuration file with the `logstash --config.test_and_exit -f <configuration-file>` command. Configuration options for some Logstash plugins have changed in the 5.x release.
5. Restart your Logstash pipeline after updating your configuration file.

[Upgrading Logstash to 5.0 »](#)

Upgrading Using a Direct Download

This procedure downloads the relevant Logstash binaries directly from Elastic.

1. Shut down your Logstash pipeline, including any inputs that send events to Logstash.
2. Download the [Logstash installation file](#) that matches your host environment.
3. Unpack the installation file into your Logstash directory.
4. Test your configuration file with the `logstash --config.test_and_exit -f <configuration-file>` command. Configuration options for some Logstash plugins have changed in the 5.x release.
5. Restart your Logstash pipeline after updating your configuration file.

Upgrading Logstash to 5.0

Before upgrading Logstash, remember to read the [breaking changes](#).

If you are installing Logstash with other components in the Elastic Stack, also see the [Elastic Stack installation and upgrade documentation](#).

When to Upgrade

Fresh installations can and should start with the same version across the Elastic Stack.

Elasticsearch 5.0 does not require Logstash 5.0. An Elasticsearch 5.0 cluster will happily receive data from a Logstash 2.x instance via the default HTTP communication layer. This provides some flexibility to decide when to upgrade Logstash relative to an Elasticsearch upgrade. It may or may not be convenient for you to upgrade them together, and it is not required to be done at the same time as long as Elasticsearch is upgraded first.

You should upgrade in a timely manner to get the performance improvements that come with Logstash 5.0, but do so in the way that makes the most sense for your environment.

When Not to Upgrade

If any Logstash plugin that you require is not compatible with Logstash 5.0, then you should wait until it is ready before upgrading.

Although we make great efforts to ensure compatibility, Logstash 5.0 is not completely backwards compatible. As noted in the Elastic Stack upgrade guide, Logstash 5.0 should not be upgraded before Elasticsearch 5.0. This is both practical and because some Logstash 5.0 plugins may attempt to use features of Elasticsearch 5.0 that did not exist in earlier versions. For example, if you attempt to send the 5.x template to a cluster before Elasticsearch 5.0, then it will not be able to use it and all indexing will fail likely fail. If you use your own, custom template with Logstash, then this issue can be ignored.

Note the Elasticsearch Output Index Template change in the *Breaking changes* documentation for further insight into this change and how it impacts operations.

Configuring Logstash

To configure Logstash, you create a config file that specifies which plugins you want to use and settings for each plugin. You can reference event fields in a configuration and use conditionals to process events when they meet certain criteria. When you run logstash, you use the `-f` to specify your config file.

Let's step through creating a simple config file and using it to run Logstash. Create a file named "logstash-simple.conf" and save it in the same directory as Logstash.

```
input { stdin { } }
output {
  elasticsearch { hosts => ["localhost:9200"] }
  stdout { codec => rubydebug }
}
```

Then, run logstash and specify the configuration file with the `-f` flag.

```
bin/logstash -f logstash-simple.conf
```

Et voilà! Logstash reads the specified configuration file and outputs to both Elasticsearch and stdout. Before we move on to some [more complex examples](#), let's take a closer look at what's in a config file.

Structure of a Config File

A Logstash config file has a separate section for each type of plugin you want to add to the event processing pipeline. For example:

```
# This is a comment. You should use comments to describe
# parts of your configuration.
input {
    ...
}

filter {
    ...
}

output {
    ...
}
```

Each section contains the configuration options for one or more plugins. If you specify multiple filters, they are applied in the order of their appearance in the configuration file.

Plugin Configuration

The configuration of a plugin consists of the plugin name followed by a block of settings for that plugin. For example, this input section configures two file inputs:

```
input {
    file {
        path => "/var/log/messages"
        type => "syslog"
    }

    file {
        path => "/var/log/apache/access.log"
        type => "apache"
    }
}
```

In this example, two settings are configured for each of the file inputs: *path* and *type*.

The settings you can configure vary according to the plugin type. For information about each plugin, see [Input Plugins](#), [Output Plugins](#), [Filter Plugins](#), and [Codec Plugins](#).

Value Types

A plugin can require that the value for a setting be a certain type, such as boolean, list, or hash. The following value types are supported.

Array

This type is now mostly deprecated in favor of using a standard type like `string` with the plugin defining the `:list => true` property for better type checking. It is still needed to handle lists of hashes or mixed types where type checking is not desired.

Example:

```
users => [ {id => 1, name => bob}, {id => 2, name => jane} ]
```

Lists

Not a type in and of itself, but a property types can have. This makes it possible to type check multiple values. Plugin authors can enable list checking by specifying `:list => true` when declaring an argument.

Example:

```
path => [ "/var/log/messages", "/var/log/*.log" ]
uris => [ "http://elastic.co", "http://example.net" ]
```

This example configures `path`, which is a `string` to be a list that contains an element for each of the three strings. It also will configure the `uris` parameter to be a list of URIs, failing if any of the URIs provided are not valid.

Boolean

A boolean must be either `true` or `false`. Note that the `true` and `false` keywords are not enclosed in quotes.

Example:

```
ssl_enable => true
```

Bytes

A bytes field is a string field that represents a valid unit of bytes. It is a convenient way to declare specific sizes in your plugin options. Both SI (k M G T P E Z Y) and Binary (Ki Mi Gi Ti Pi Ei Zi Yi) units are supported. Binary units are in base-1024 and SI units are in base-1000. This field is case-insensitive and accepts space between the value and the unit. If no unit is specified, the integer string represents the number of bytes.

Examples:

```
my_bytes => "1113"    # 1113 bytes
my_bytes => "10MiB"   # 10485760 bytes
my_bytes => "100kib"  # 102400 bytes
my_bytes => "180 mb"  # 180000000 bytes
```

Codec

A codec is the name of Logstash codec used to represent the data. Codecs can be used in both inputs and outputs.

Input codecs provide a convenient way to decode your data before it enters the input. Output codecs provide a convenient way to encode your data before it leaves the output. Using an input or output codec eliminates the need for a separate filter in your Logstash pipeline.

A list of available codecs can be found at the [Codec Plugins](#) page.

Example:

```
codec => "json"
```

Hash

A hash is a collection of key value pairs specified in the format "field1" => "value1". Note that multiple key value entries are separated by spaces rather than commas.

Example:

```
match => {
  "field1" => "value1"
  "field2" => "value2"
  ...
}
```

Number

Numbers must be valid numeric values (floating point or integer).

Example:

```
port => 33
```

Password

A password is a string with a single value that is not logged or printed.

Example:

```
my_password => "password"
```

URI

A URI can be anything from a full URL like *http://elastic.co/* to a simple identifier like *foobar*. If the URI contains a password such as *http://user:pass@example.net* the password portion of the URI will not be logged or printed.

Example:

```
my_uri => "http://foo:bar@example.net"  
Path
```

A path is a string that represents a valid operating system path.

Example:

```
my_path => "/tmp/logstash"  
String
```

A string must be a single character sequence. Note that string values are enclosed in quotes, either double or single. Literal quotes in the string need to be escaped with a backslash if they are of the same kind as the string delimiter, i.e. single quotes within a single-quoted string need to be escaped as well as double quotes within a double-quoted string.

Example:

```
name => "Hello world"  
name => 'It\'s a beautiful day'
```

Comments

Comments are the same as in perl, ruby, and python. A comment starts with a # character, and does not need to be at the beginning of a line. For example:

```
# this is a comment  
  
input { # comments can appear at the end of a line, too  
    # ...  
}
```

Event Dependent Configuration

The logstash agent is a processing pipeline with 3 stages: inputs → filters → outputs. Inputs generate events, filters modify them, outputs ship them elsewhere.

All events have properties. For example, an apache access log would have things like status code (200, 404), request path ("/", "index.html"), HTTP verb (GET, POST), client IP address, etc. Logstash calls these properties "fields."

Some of the configuration options in Logstash require the existence of fields in order to function. Because inputs generate events, there are no fields to evaluate within the input block—they do not exist yet!

Because of their dependency on events and fields, the following configuration options will only work within filter and output blocks.

IMPORTANT: Field references, sprintf format and conditionals, described below, will not work in an input block.

Field References

It is often useful to be able to refer to a field by name. To do this, you can use the Logstash field reference syntax.

The syntax to access a field is `[fieldname]`. If you are referring to a **top-level field**, you can omit the `[]` and simply use `fieldname`. To refer to a **nested field**, you specify the full path to that field: `[top-level field] [nested field]`.

For example, the following event has five top-level fields (`agent`, `ip`, `request`, `response`, `ua`) and three nested fields (`status`, `bytes`, `os`).

```
{
  "agent": "Mozilla/5.0 (compatible; MSIE 9.0)",
  "ip": "192.168.24.44",
  "request": "/index.html"
  "response": {
    "status": 200,
    "bytes": 52353
  },
  "ua": {
    "os": "Windows 7"
  }
}
```

To reference the `os` field, you specify `[ua][os]`. To reference a top-level field such as `request`, you can simply specify the field name.

sprintfformat

The field reference format is also used in what Logstash calls *sprintfformat*. This format enables you to refer to field values from within other strings. For example, the statsd output has an `increment` setting that enables you to keep a count of apache logs by status code:

```
output {  
    statsd {  
        increment => "apache.%{[response][status]}"  
    }  
}
```

Similarly, you can convert the timestamp in the `@timestamp` field into a string. Instead of specifying a field name inside the curly braces, use the `+FORMAT` syntax where `FORMAT` is a [time format](#).

For example, if you want to use the file output to write to logs based on the event's date and hour and the `type` field:

```
output {  
    file {  
        path => "/var/log/%{type}.%{+yyyy.MM.dd.HH}"  
    }  
}
```

Conditionals

Sometimes you only want to filter or output an event under certain conditions. For that, you can use a conditional.

Conditionals in Logstash look and act the same way they do in programming languages. Conditionals support `if`, `else if` and `else` statements and can be nested.

The conditional syntax is:

```
if EXPRESSION {  
    ...  
} else if EXPRESSION {  
    ...  
} else {  
    ...  
}
```

What's an expression? Comparison tests, boolean logic, and so on!

You can use the following comparison operators:

- **equality:** ==, !=, <, >, <=, >=
- **regexp:** =~, !~ (checks a pattern on the right against a string value on the left)
- **inclusion:** in, not in

The supported boolean operators are:

- and, or, nand, xor

The supported unary operators are:

- !

Expressions can be long and complex. Expressions can contain other expressions, you can negate expressions with !, and you can group them with parentheses (...).

For example, the following conditional uses the mutate filter to remove the field `secret` if the field `action` has a value of `login`:

```
filter {
  if [action] == "login" {
    mutate { remove_field => "secret" }
  }
}
```

You can specify multiple expressions in a single condition:

```
output {
  # Send production errors to pagerduty
  if [loglevel] == "ERROR" and [deployment] == "production" {
    pagerduty {
      ...
    }
  }
}
```

You can use the `in` operator to test whether a field contains a specific string, key, or (for lists) element:

```
filter {
  if [foo] in [foobar] {
    mutate { add_tag => "field in field" }
  }
  if [foo] in "foo" {
    mutate { add_tag => "field in string" }
  }
  if "hello" in [greeting] {
    mutate { add_tag => "string in field" }
  }
  if [foo] in ["hello", "world", "foo"] {
    mutate { add_tag => "field in list" }
  }
}
```

```

if [missing] in [alsomissing] {
    mutate { add_tag => "shouldnotexist" }
}
if !("foo" in ["hello", "world"]) {
    mutate { add_tag => "shouldexist" }
}
}

```

You use the `not in` conditional the same way. For example, you could use `not in` to only route events to Elasticsearch when `grok` is successful:

```

output {
    if "_grokparsefailure" not in [tags] {
        elasticsearch { ... }
    }
}

```

You can check for the existence of a specific field, but there's currently no way to differentiate between a field that doesn't exist versus a field that's simply false. The expression `if [foo]` returns `false` when:

- `[foo]` doesn't exist in the event,
- `[foo]` exists in the event, but is false, or
- `[foo]` exists in the event, but is null

For more complex examples, see [Using Conditionals](#).

The @metadata field

In Logstash 1.5 and later, there is a special field called `@metadata`. The contents of `@metadata` will not be part of any of your events at output time, which makes it great to use for conditionals, or extending and building event fields with field reference and `sprintf` formatting.

The following configuration file will yield events from STDIN. Whatever is typed will become the `message` field in the event. The `mutate` events in the filter block will add a few fields, some nested in the `@metadata` field.

```

input { stdin { } }

filter {
    mutate { add_field => { "show" => "This data will be in the output" } }
    mutate { add_field => { "[@metadata][test]" => "Hello" } }
    mutate { add_field => { "[@metadata][no_show]" => "This data will not be in
the output" } }
}

output {
    if [@metadata][test] == "Hello" {
        stdout { codec => rubydebug }
    }
}

```

Let's see what comes out:

```
$ bin/logstash -f ./test.conf
Pipeline main started
asdf
{
    "@timestamp" => 2016-06-30T02:42:51.496Z,
    "@version" => "1",
    "host" => "example.com",
    "show" => "This data will be in the output",
    "message" => "asdf"
}
```

The "asdf" typed in became the `message` field contents, and the conditional successfully evaluated the contents of the `test` field nested within the `@metadata` field. But the output did not show a field called `@metadata`, or its contents.

The `rubydebug` codec allows you to reveal the contents of the `@metadata` field if you add a config flag, `metadata => true`:

```
stdout { codec => rubydebug { metadata => true } }
```

Let's see what the output looks like with this change:

```
$ bin/logstash -f ./test.conf
Pipeline main started
asdf
{
    "@timestamp" => 2016-06-30T02:46:48.565Z,
    "@metadata" => {
        "test" => "Hello",
        "no_show" => "This data will not be in the output"
    },
    "@version" => "1",
    "host" => "example.com",
    "show" => "This data will be in the output",
    "message" => "asdf"
}
```

Now you can see the `@metadata` field and its sub-fields.

IMPORTANT: Only the `rubydebug` codec allows you to show the contents of the `@metadata` field.

Make use of the `@metadata` field any time you need a temporary field but do not want it to be in the final output.

Perhaps one of the most common use cases for this new field is with the `date` filter and having a temporary timestamp.

This configuration file has been simplified, but uses the timestamp format common to Apache and Nginx web servers. In the past, you'd have to delete the timestamp field yourself, after using it to overwrite the `@timestamp` field. With the `@metadata` field, this is no longer necessary:

```
input { stdin { } }

filter {
    grok { match => [ "message", "%{HTTPDATE:[@metadata][timestamp]}" ] }
    date { match => [ "[@metadata][timestamp]", "dd/MMM/yyyy:HH:mm:ss Z" ] }
}

output {
    stdout { codec => rubydebug }
}
```

Notice that this configuration puts the extracted date into the `[@metadata][timestamp]` field in the `grok` filter. Let's feed this configuration a sample date string and see what comes out:

```
$ bin/logstash -f ./test.conf
Pipeline main started
02/Mar/2014:15:36:43 +0100
{
    "@timestamp" => 2014-03-02T14:36:43.000Z,
    "@version" => "1",
    "host" => "example.com",
    "message" => "02/Mar/2014:15:36:43 +0100"
}
```

That's it! No extra fields in the output, and a cleaner config file because you do not have to delete a "timestamp" field after conversion in the `date` filter.

Another use case is the CouchDB Changes input plugin (See https://github.com/logstash-plugins/logstash-input-couchdb_changes). This plugin automatically captures CouchDB document field metadata into the `@metadata` field within the input plugin itself. When the events pass through to be indexed by Elasticsearch, the Elasticsearch output plugin allows you to specify the `action` (delete, update, insert, etc.) and the `document_id`, like this:

```
output {
    elasticsearch {
        action => "%{[@metadata][action]}"
        document_id => "%{[@metadata][_id]}"
        hosts => ["example.com"]
        index => "index_name"
        protocol => "http"
    }
}
```

Using Environment Variables in the Configuration

Overview

- You can set environment variable references in the configuration for Logstash plugins by using `${var}` .
- At Logstash startup, each reference will be replaced by the value of the environment variable.
- The replacement is case-sensitive.
- References to undefined variables raise a Logstash configuration error.
- You can give a default value by using the form `${var:default value}` . Logstash uses the default value if the environment variable is undefined.
- You can add environment variable references in any plugin option type : string, number, boolean, array, or hash.
- Environment variables are immutable. If you update the environment variable, you'll have to restart Logstash to pick up the updated value.

Examples

The following examples show you how to use environment variables to set the values of some commonly used configuration options.

Setting the TCP Port

Here's an example that uses an environment variable to set the TCP port:

```
input {
  tcp {
    port => "${TCP_PORT}"
  }
}
```

Now let's set the value of `TCP_PORT`:

```
export TCP_PORT=12345
```

At startup, Logstash uses the following configuration:

```
input {
  tcp {
    port => 12345
  }
}
```

If the `TCP_PORT` environment variable is not set, Logstash returns a configuration error.

You can fix this problem by specifying a default value:

```
input {
```

Logstash 5.2 Configuration Guide

```
tcp {
    port => "${TCP_PORT:54321}"
}
}
```

Now, instead of returning a configuration error if the variable is undefined, Logstash uses the default:

```
input {
    tcp {
        port => 54321
    }
}
```

If the environment variable is defined, Logstash uses the value specified for the variable instead of the default.

Setting the Value of a Tag

Here's an example that uses an environment variable to set the value of a tag:

```
filter {
    mutate {
        add_tag => [ "tag1", "${ENV_TAG}" ]
    }
}
```

Let's set the value of `ENV_TAG`:

```
export ENV_TAG="tag2"
```

At startup, Logstash uses the following configuration:

```
filter {
    mutate {
        add_tag => [ "tag1", "tag2" ]
    }
}
```

Setting a File Path

Here's an example that uses an environment variable to set the path to a log file:

```
filter {
    mutate {
        add_field => {
            "my_path" => "${HOME}/file.log"
        }
    }
}
```

Let's set the value of `HOME`:

Logstash 5.2 Configuration Guide

```
export HOME="/path"
```

At startup, Logstash uses the following configuration:

```
filter {
  mutate {
    add_field => {
      "my_path" => "/path/file.log"
    }
  }
}
```

Logstash Configuration Examples

The following examples illustrate how you can configure Logstash to filter events, process Apache logs and syslog messages, and use conditionals to control what events are processed by a filter or output.

Configuring Filters

Filters are an in-line processing mechanism that provide the flexibility to slice and dice your data to fit your needs. Let's take a look at some filters in action. The following configuration file sets up the `grok` and `date` filters.

```
input { stdin { } }

filter {
  grok {
    match => { "message" => "%{COMBINEDAPACHELOG}" }
  }
  date {
    match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
  }
}

output {
  elasticsearch { hosts => ["localhost:9200"] }
  stdout { codec => rubydebug }
}
```

Run Logstash with this configuration:

```
bin/logstash -f logstash-filter.conf
```

Now, paste the following line into your terminal so it will be processed by the `stdin` input:

```
127.0.0.1 -- [11/Dec/2013:00:01:45 -0800] "GET /xampp/status.php HTTP/1.1"
200 3891 "http://cadenza/xampp/navi.php" "Mozilla/5.0 (Macintosh; Intel Mac
OS X 10.9; rv:25.0) Gecko/20100101 Firefox/25.0"
```

You should see something returned to `stdout` that looks like this:

```
{
  "message" => "127.0.0.1 -- [11/Dec/2013:00:01:45 -0800] \"GET
/xampp/status.php HTTP/1.1\" 200 3891 \"http://cadenza/xampp/navi.php\"
\"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:25.0) Gecko/20100101
Firefox/25.0\"",
  "@timestamp" => "2013-12-11T08:01:45.000Z",
  "@version" => "1",
  "host" => "cadenza",
  "clientip" => "127.0.0.1",
  "ident" => "-",
  "auth" => "-"}
```

```

    "timestamp" => "11/Dec/2013:00:01:45 -0800",
    "verb" => "GET",
    "request" => "/xampp/status.php",
    "httpversion" => "1.1",
    "response" => "200",
    "bytes" => "3891",
    "referrer" => "\"http://cadenza/xampp/navi.php\"",
    "agent" => "\"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:25.0)
Gecko/20100101 Firefox/25.0\""
}

```

As you can see, Logstash (with help from the `grok` filter) was able to parse the log line (which happens to be in Apache "combined log" format) and break it up into many different discrete bits of information. This is extremely useful once you start querying and analyzing our log data. For example, you'll be able to easily run reports on HTTP response codes, IP addresses, referrers, and so on. There are quite a few grok patterns included with Logstash out-of-the-box, so it's quite likely if you need to parse a common log format, someone has already done the work for you. For more information, see the list of [Logstash grok patterns](#) on GitHub.

The other filter used in this example is the `date` filter. This filter parses out a timestamp and uses it as the timestamp for the event (regardless of when you're ingesting the log data). You'll notice that the `@timestamp` field in this example is set to December 11, 2013, even though Logstash is ingesting the event at some point afterwards. This is handy when backfilling logs. It gives you the ability to tell Logstash "use this value as the timestamp for this event".

[Processing Apache Logs](#)

Let's do something that's actually **useful**: process apache2 access log files! We are going to read the input from a file on the localhost, and use a [conditional](#) to process the event according to our needs. First, create a file called something like `logstash-apache.conf` with the following contents (you can change the log's file path to suit your needs):

```

input {
  file {
    path => "/tmp/access_log"
    start_position => "beginning"
  }
}

filter {
  if [path] =~ "access" {
    mutate { replace => { "type" => "apache_access" } }
    grok {
      match => { "message" => "%{COMBINEDAPACHELOG}" }
    }
  }
  date {
    match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
  }
}

output {

```

Logstash 5.2 Configuration Guide

```
elasticsearch {  
    hosts => ["localhost:9200"]  
}  
stdout { codec => rubydebug }  
}
```

Then, create the input file you configured above (in this example, "/tmp/access_log") with the following log entries (or use some from your own webserver):

```
71.141.244.242 - kurt [18/May/2011:01:48:10 -0700] "GET /admin HTTP/1.1" 301  
566 "-" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.3)  
Gecko/20100401 Firefox/3.6.3"  
134.39.72.245 - - [18/May/2011:12:40:18 -0700] "GET /favicon.ico HTTP/1.1"  
200 1189 "-" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0;  
.NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; InfoPath.2;  
.NET4.0C; .NET4.0E)"  
98.83.179.51 - - [18/May/2011:19:35:08 -0700] "GET /css/main.css HTTP/1.1"  
200 1837 "http://www.safesand.com/information.htm" "Mozilla/5.0 (Windows NT  
6.0; WOW64; rv:2.0.1) Gecko/20100101 Firefox/4.0.1"
```

Now, run Logstash with the -f flag to pass in the configuration file:

```
bin/logstash -f logstash-apache.conf
```

Now you should see your apache log data in Elasticsearch! Logstash opened and read the specified input file, processing each event it encountered. Any additional lines logged to this file will also be captured, processed by Logstash as events, and stored in Elasticsearch. As an added bonus, they are stashed with the field "type" set to "apache_access" (this is done by the type => "apache_access" line in the input configuration).

In this configuration, Logstash is only watching the apache access_log, but it's easy enough to watch both the access_log and the error_log (actually, any file matching *log), by changing one line in the above configuration:

```
input {  
    file {  
        path => "/tmp/*_log"  
    ...  
}
```

When you restart Logstash, it will process both the error and access logs. However, if you inspect your data (using elasticsearch-kopf, perhaps), you'll see that the access_log is broken up into discrete fields, but the error_log isn't. That's because we used a grok filter to match the standard combined apache log format and automatically split the data into separate fields. Wouldn't it be nice if we could control how a line was parsed, based on its format? Well, we can...

Note that Logstash did not reprocess the events that were already seen in the access_log file. When reading from a file, Logstash saves its position and only processes new lines as they are added. Neat!

Using Conditionals

You use conditionals to control what events are processed by a filter or output. For example, you could label each event according to which file it appeared in (access_log, error_log, and other random files that end with "log").

```
input {
  file {
    path => "/tmp/*_log"
  }
}

filter {
  if [path] =~ "access" {
    mutate { replace => { type => "apache_access" } }
    grok {
      match => { "message" => "%{COMBINEDAPACHELOG}" }
    }
    date {
      match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
    }
  } else if [path] =~ "error" {
    mutate { replace => { type => "apache_error" } }
  } else {
    mutate { replace => { type => "random_logs" } }
  }
}

output {
  elasticsearch { hosts => ["localhost:9200"] }
  stdout { codec => rubydebug }
}
```

This example labels all events using the `type` field, but doesn't actually parse the `error` or `random` files. There are so many types of error logs that how they should be labeled really depends on what logs you're working with.

Similarly, you can use conditionals to direct events to particular outputs. For example, you could:

- alert nagios of any apache events with status 5xx
- record any 4xx status to Elasticsearch
- record all status code hits via statsd

To tell nagios about any http event that has a 5xx status code, you first need to check the value of the `type` field. If it's apache, then you can check to see if the `status` field contains a 5xx error. If it is, send it to nagios. If it isn't a 5xx error, check to see if the `status` field contains a 4xx error. If so, send it to Elasticsearch. Finally, send all apache status codes to statsd no matter what the `status` field contains:

```
output {
```

Logstash 5.2 Configuration Guide

```
if [type] == "apache" {
  if [status] =~ /^5\d\d/ {
    nagios { ... }
  } else if [status] =~ /^4\d\d/ {
    elasticsearch { ... }
  }
  statsd { increment => "apache.%{status}" }
}
}
```

[Processing Syslog Messages](#)

Syslog is one of the most common use cases for Logstash, and one it handles exceedingly well (as long as the log lines conform roughly to RFC3164). Syslog is the de facto UNIX networked logging standard, sending messages from client machines to a local file, or to a centralized log server via rsyslog. For this example, you won't need a functioning syslog instance; we'll fake it from the command line so you can get a feel for what happens.

First, let's make a simple configuration file for Logstash + syslog, called *logstash-syslog.conf*.

```
input {
  tcp {
    port => 5000
    type => syslog
  }
  udp {
    port => 5000
    type => syslog
  }
}

filter {
  if [type] == "syslog" {
    grok {
      match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp}%
%{SYSLOGHOST:syslog_hostname}%
%{DATA:syslog_program}(?:\[ %{POSINT:syslog_pid}\])?:
%{GREEDYDATA:syslog_message}" }
      add_field => [ "received_at", "%{@timestamp}" ]
      add_field => [ "received_from", "%{host}" ]
    }
    date {
      match => [ "syslog_timestamp", "MMM d HH:mm:ss", "MMM dd HH:mm:ss" ]
    }
  }
}

output {
  elasticsearch { hosts => ["localhost:9200"] }
  stdout { codec => rubydebug }
}
```

Run Logstash with this new configuration:

```
bin/logstash -f logstash-syslog.conf
```

Normally, a client machine would connect to the Logstash instance on port 5000 and send its message. For this example, we'll just telnet to Logstash and enter a log line (similar to how we entered log lines into STDIN earlier). Open another shell window to interact with the Logstash syslog input and enter the following command:

```
telnet localhost 5000
```

Copy and paste the following lines as samples. (Feel free to try some of your own, but keep in mind they might not parse if the `grok` filter is not correct for your data).

```
Dec 23 12:11:43 louis postfix/smtpd[31499]: connect from
unknown[95.75.93.154]
Dec 23 14:42:56 louis named[16000]: client 199.48.164.7#64817: query (cache)
'amsterdamboothuren.com/MX/IN' denied
Dec 23 14:30:01 louis CRON[619]: (www-data) CMD (php
/usr/share/cacti/site/poller.php >/dev/null 2>/var/log/cacti/poller-
error.log)
Dec 22 18:28:06 louis rsyslogd: [origin software="rsyslogd" swVersion="4.2.0"
x-pid="2253" x-info="http://www.rsyslog.com"] rsyslogd was HUPed, type
'lightweight'.
```

Now you should see the output of Logstash in your original shell as it processes and parses messages!

```
{
    "message" => "Dec 23 14:30:01 louis CRON[619]: (www-data)
CMD (php /usr/share/cacti/site/poller.php >/dev/null 2>/var/log/cacti/poller-
error.log)",
    "@timestamp" => "2013-12-23T22:30:01.000Z",
    "@version" => "1",
    "type" => "syslog",
    "host" => "0:0:0:0:0:0:1:52617",
    "syslog_timestamp" => "Dec 23 14:30:01",
    "syslog_hostname" => "louis",
    "syslog_program" => "CRON",
    "syslog_pid" => "619",
    "syslog_message" => "(www-data) CMD (php
/usr/share/cacti/site/poller.php >/dev/null 2>/var/log/cacti/poller-
error.log)",
    "received_at" => "2013-12-23 22:49:22 UTC",
    "received_from" => "0:0:0:0:0:0:1:52617",
    "syslog_severity_code" => 5,
    "syslog_facility_code" => 1,
    "syslog_facility" => "user-level",
    "syslog_severity" => "notice"
}
```

Reloading the Config File

Starting with Logstash 2.3, you can set Logstash to detect and reload configuration changes automatically.

To enable automatic config reloading, start Logstash with the `--config.reload.automatic` (or `-r`) command-line option specified. For example:

```
bin/logstash -f apache.config --config.reload.automatic
```

NOTE: The `--config.reload.automatic` option is not available when you specify the `-e` flag to pass in configuration settings from the command-line.

By default, Logstash checks for configuration changes every 3 seconds. To change this interval, use the `--config.reload.interval <seconds>` option, where `seconds` specifies how often Logstash checks the config files for changes.

If Logstash is already running without auto-reload enabled, you can force Logstash to reload the config file and restart the pipeline by sending a SIGHUP (signal hangup) to the process running Logstash. For example:

```
kill -1 14175
```

Where 14175 is the ID of the process running Logstash.

How Automatic Config Reloading Works

When Logstash detects a change in a config file, it stops the current pipeline by stopping all inputs, and it attempts to create a new pipeline that uses the updated configuration. After validating the syntax of the new configuration, Logstash verifies that all inputs and outputs can be initialized (for example, that all required ports are open). If the checks are successful, Logstash swaps the existing pipeline with the new pipeline. If the checks fail, the old pipeline continues to function, and the errors are propagated to the console.

During automatic config reloading, the JVM is not restarted. The creating and swapping of pipelines all happens within the same process.

Changes to [grok](#) pattern files are also reloaded, but only when a change in the config file triggers a reload (or the pipeline is restarted).

Managing Multiline Events

Several use cases generate events that span multiple lines of text. In order to correctly handle these multiline events, Logstash needs to know how to tell which lines are part of a single event.

Multiline event processing is complex and relies on proper event ordering. The best way to guarantee ordered log processing is to implement the processing as early in the pipeline as possible. The preferred tool in the Logstash pipeline is the [multiline codec](#), which merges lines from a single input using a simple set of rules.

The most important aspects of configuring the multiline codec are the following:

- The `pattern` option specifies a regular expression. Lines that match the specified regular expression are considered either continuations of a previous line or the start of a new multiline event. You can use [grok](#) regular expression templates with this configuration option.
- The `what` option takes two values: `previous` or `next`. The `previous` value specifies that lines that match the value in the `pattern` option are part of the previous line. The `next` value specifies that lines that match the value in the `pattern` option are part of the following line.* The `negate` option applies the multiline codec to lines that *do not* match the regular expression specified in the `pattern` option.

See the full documentation for the [multiline codec](#) plugin for more information on configuration options.

Examples of Multiline Codec Configuration

The examples in this section cover the following use cases:

- Combining a Java stack trace into a single event
- Combining C-style line continuations into a single event
- Combining multiple lines from time-stamped events

[Java Stack Traces](#)

Java stack traces consist of multiple lines, with each line after the initial line beginning with whitespace, as in this example:

```
Exception in thread "main" java.lang.NullPointerException
        at com.example.myproject.Book.getTitle(Book.java:16)
        at com.example.myproject.Author.getBookTitles(Author.java:25)
        at com.example.myproject.Bootstrap.main(Bootstrap.java:14)
```

To consolidate these lines into a single event in Logstash, use the following configuration for the multiline codec:

```
input {
    stdin {
```

Logstash 5.2 Configuration Guide

```
    codec => multiline {
      pattern => "^\\s"
      what => "previous"
    }
  }
}
```

This configuration merges any line that begins with whitespace up to the previous line.

Line Continuations

Several programming languages use the \ character at the end of a line to denote that the line continues, as in this example:

```
printf ("%10.10ld  \t %10.10ld \t %s\
%f", w, x, y, z );
```

To consolidate these lines into a single event in Logstash, use the following configuration for the multiline codec:

```
input {
  stdin {
    codec => multiline {
      pattern => "\\$"
      what => "next"
    }
  }
}
```

This configuration merges any line that ends with the \ character with the following line.

Timestamps

Activity logs from services such as Elasticsearch typically begin with a timestamp, followed by information on the specific activity, as in this example:

```
[2015-08-24 11:49:14,389] [INFO ] [env] [Letha] using [1]
data paths, mounts [[/(/dev/disk1)]], net usable_space [34.5gb], net total_space [118.9gb], types [hfs]
```

To consolidate these lines into a single event in Logstash, use the following configuration for the multiline codec:

```
input {
  file {
    path => "/var/log/someapp.log"
    codec => multiline {
      pattern => "^%{TIMESTAMP_ISO8601} "
      negate => true
      what => previous
    }
  }
}
```

```
        }  
    }  
}
```

This configuration uses the `negate` option to specify that any line that does not begin with a timestamp belongs to the previous line.

Glob Pattern Support

Logstash supports the following patterns wherever glob patterns are allowed:

Match any file. You can also use an `*` to restrict other values in the glob. For example, `*.conf` matches all files that end in `.conf`. `*apache*` matches any files with `apache` in the name. This pattern does not match hidden files (dot files) on Unix-like operating systems. To match dot files, use a pattern like `{*, .*}`.

Match directories recursively.

?

Match any one character.

[set]

Match any one character in a set. For example, `[a-z]`. Also supports set negation (`[^a-z]`).

{p,q}

Match either literal `p` or literal `q`. The matching literal can be more than one character, and you can specify more than two literals. This pattern is the equivalent to using alternation with the vertical bar in regular expressions (`foo|bar`).

Escape the next metacharacter. This means that you cannot use a backslash in Windows as part of a glob. The pattern `c:\foo*` will not work, so use `foo*` instead.

Example Patterns

Here are some common examples of glob patterns:

`"/path/to/*.*.conf"`

Matches config files ending in `.conf` in the specified path.

`"/var/log/*.*.log"`

Matches log files ending in `.log` in the specified path.

`"/var/log/**/*.*.log"`

Matches log files ending in `.log` in subdirectories under the specified path.

`"/path/to/logs/{app1,app2,app3}/data.*.log"`

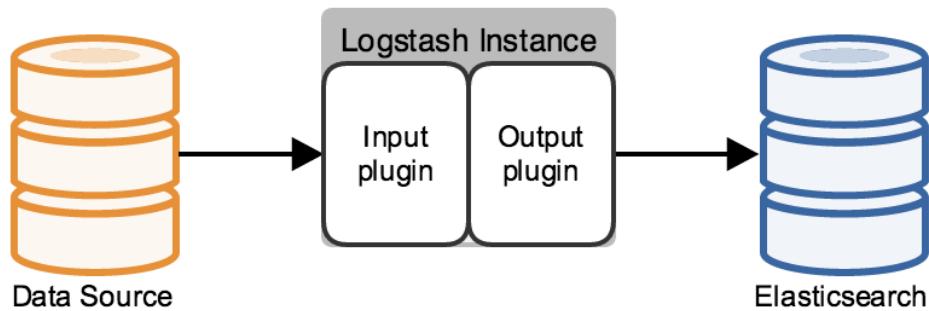
Matches app log files in the `app1`, `app2`, and `app3` subdirectories under the specified path.

Deploying and Scaling Logstash

As your use case for Logstash evolves, the preferred architecture at a given scale will change. This section discusses a range of Logstash architectures in increasing order of complexity, starting from a minimal installation and adding elements to the system. The example deployments in this section write to an Elasticsearch cluster, but Logstash can write to a large variety of [endpoints](#).

The Minimal Installation

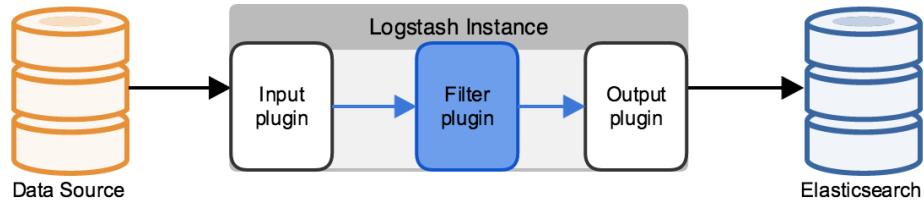
The minimal Logstash installation has one Logstash instance and one Elasticsearch instance. These instances are directly connected. Logstash uses an [input plugin](#) to ingest data and an Elasticsearch [output plugin](#) to index the data in Elasticsearch, following the Logstash [processing pipeline](#). A Logstash instance has a fixed pipeline constructed at startup, based on the instance's configuration file. You must specify an input plugin. Output defaults to `stdout`, and the filtering section of the pipeline, which is discussed in the next section, is optional.



Using Filters

Log data is typically unstructured, often contains extraneous information that isn't relevant to your use case, and sometimes is missing relevant information that can be derived from the log contents. You can use a [filter plugin](#) to parse the log into fields, remove unnecessary information, and derive additional information from the existing fields. For example, filters can derive geolocation information from an IP address and add that information to the logs, or parse and structure arbitrary text with the [grok](#) filter.

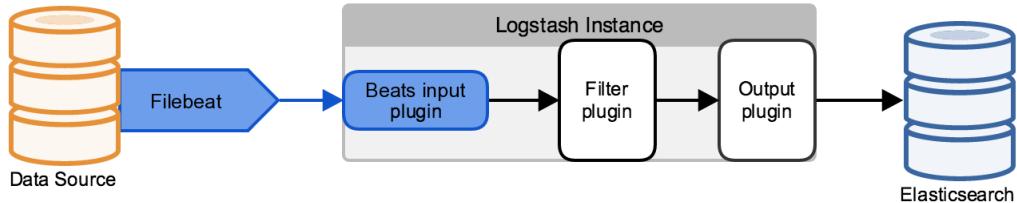
Adding a filter plugin can significantly affect performance, depending on the amount of computation the filter plugin performs, as well as on the volume of the logs being processed. The `grok` filter's regular expression computation is particularly resource-intensive. One way to address this increased demand for computing resources is to use parallel processing on multicore machines. Use the `-w` switch to set the number of execution threads for Logstash filtering tasks. For example the `bin/logstash -w 8` command uses eight different threads for filter processing.



Using Filebeat

[Filebeat](#) is a lightweight, resource-friendly tool written in Go that collects logs from files on the server and forwards these logs to other machines for processing. Filebeat uses the [Beats](#) protocol to communicate with a centralized Logstash instance. Configure the Logstash instances that receive Beats data to use the [Beats input plugin](#).

Filebeat uses the computing resources of the machine hosting the source data, and the Beats input plugin minimizes the resource demands on the Logstash instance, making this architecture attractive for use cases with resource constraints.



Scaling to a Larger Elasticsearch Cluster

Typically, Logstash does not communicate with a single Elasticsearch node, but with a cluster that comprises several nodes. By default, Logstash uses the HTTP protocol to move data into the cluster.

You can use the Elasticsearch HTTP REST APIs to index data into the Elasticsearch cluster. These APIs represent the indexed data in JSON. Using the REST APIs does not require the Java client classes or any additional JAR files and has no performance disadvantages compared to the transport or node protocols. You can secure communications that use the HTTP REST APIs by using [X-Pack Security](#), which supports SSL and HTTP basic authentication.

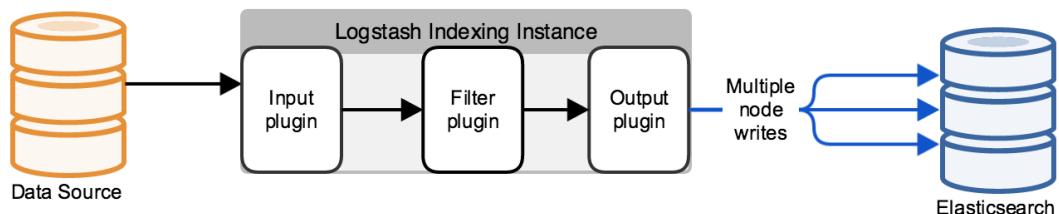
When you use the HTTP protocol, you can configure the Logstash Elasticsearch output plugin to automatically load-balance indexing requests across a specified set of hosts in the Elasticsearch cluster. Specifying multiple Elasticsearch nodes also provides high availability for the Elasticsearch cluster by routing traffic to active Elasticsearch nodes.

You can also use the Elasticsearch Java APIs to serialize the data into a binary representation, using the transport protocol. The transport protocol can sniff the endpoint of the request and select an arbitrary client or data node in the Elasticsearch cluster.

Using the HTTP or transport protocols keep your Logstash instances separate from the Elasticsearch cluster. The node protocol, by contrast, has the machine running the Logstash instance join the Elasticsearch cluster, running an Elasticsearch instance. The data that needs indexing propagates from this node to the rest of the cluster. Since the machine is part of the cluster, the cluster topology is available, making the node protocol a good fit for use cases that use a relatively small number of persistent connections.

You can also use a third-party hardware or software load balancer to handle connections between Logstash and external applications.

Make sure that your Logstash configuration does not connect directly to Elasticsearch dedicated [master nodes](#), which perform dedicated cluster management. Connect Logstash to client or data nodes to protect the stability of your Elasticsearch cluster.



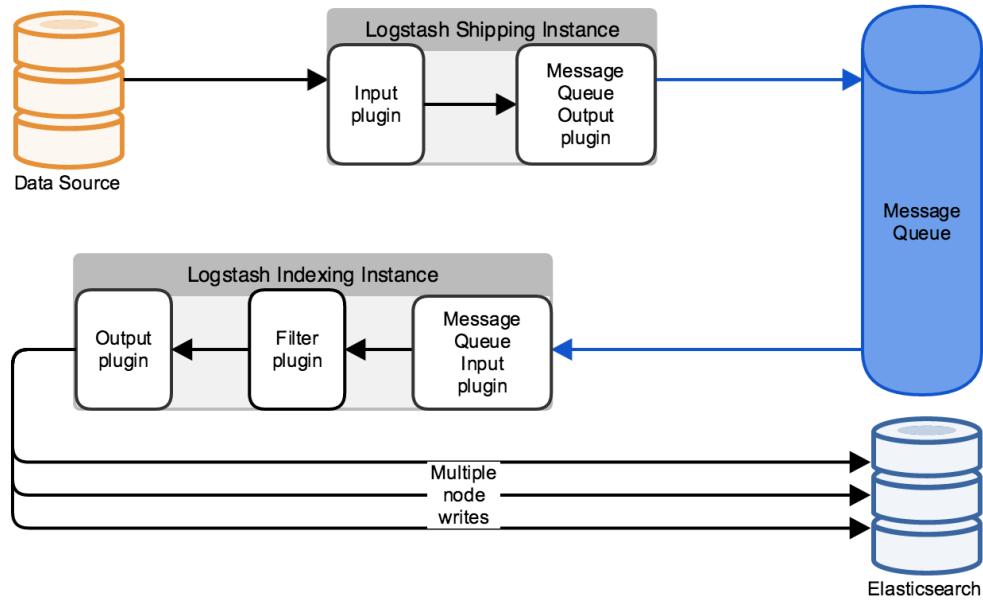
Managing Throughput Spikes with Message Queueing

When the data coming into a Logstash pipeline exceeds the Elasticsearch cluster's ability to ingest the data, you can use a message broker as a buffer. By default, Logstash throttles incoming events when indexer consumption rates fall below incoming data rates. Since this throttling can lead to events being buffered at the data source, preventing back pressure with message brokers becomes an important part of managing your deployment.

Adding a message broker to your Logstash deployment also provides a level of protection from data loss. When a Logstash instance that has consumed data from the message broker fails, the data can be replayed from the message broker to an active Logstash instance.

Several third-party message brokers exist, such as Redis, Kafka, or RabbitMQ. Logstash provides input and output plugins to integrate with several of these third-party message brokers. When your Logstash deployment has a message broker configured, Logstash functionally exists in two phases: shipping instances, which handles data ingestion and storage in the message

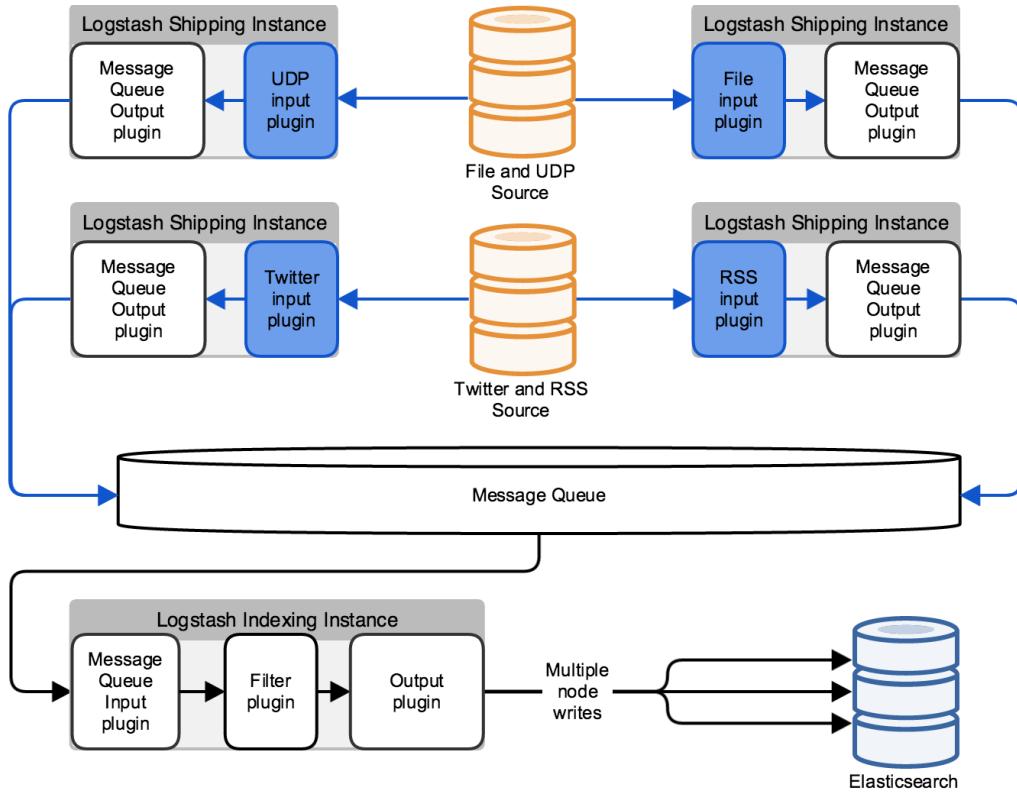
broker, and indexing instances, which retrieve the data from the message broker, apply any configured filtering, and write the filtered data to an Elasticsearch index.



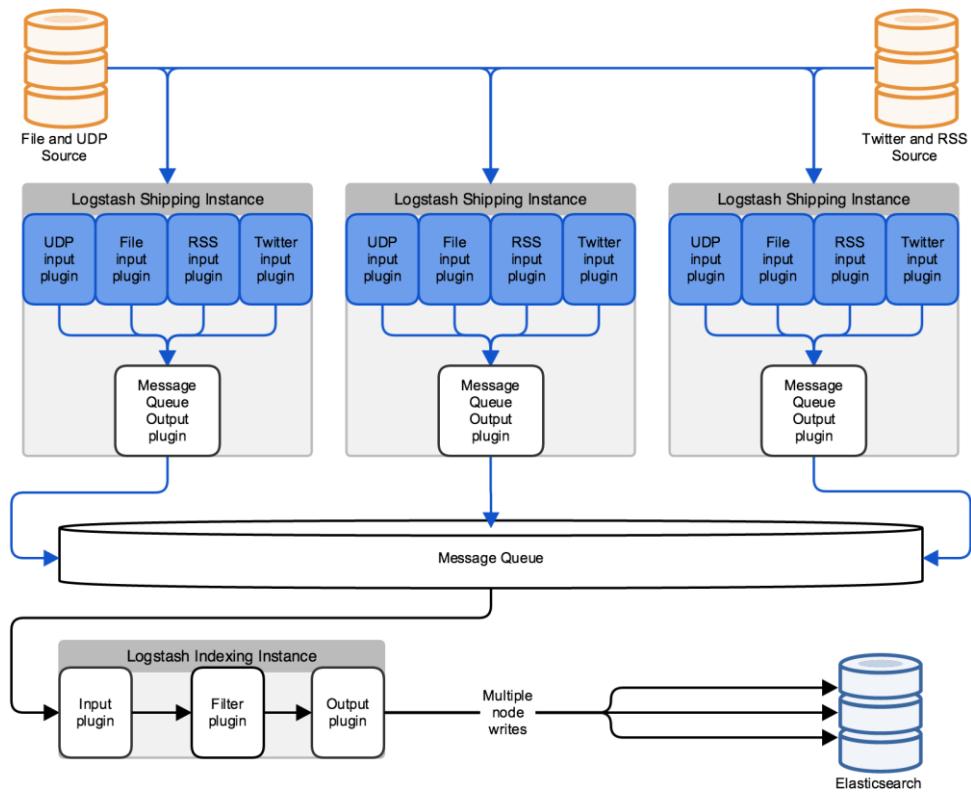
Multiple Connections for Logstash High Availability

To make your Logstash deployment more resilient to individual instance failures, you can set up a load balancer between your data source machines and the Logstash cluster. The load balancer handles the individual connections to the Logstash instances to ensure continuity of data ingestion and processing even when an individual instance is unavailable.

Logstash 5.2 Configuration Guide



The architecture in the previous diagram is unable to process input from a specific type, such as an RSS feed or a file, if the Logstash instance dedicated to that input type becomes unavailable. For more robust input processing, configure each Logstash instance for multiple inputs, as in the following diagram:



This architecture parallelizes the Logstash workload based on the inputs you configure. With more inputs, you can add more Logstash instances to scale horizontally. Separate parallel pipelines also increases the reliability of your stack by eliminating single points of failure.

Scaling Logstash

A mature Logstash deployment typically has the following pipeline:

- The *input* tier consumes data from the source, and consists of Logstash instances with the proper input plugins.
- The *message broker* serves as a buffer to hold ingested data and serve as failover protection.
- The *filter* tier applies parsing and other processing to the data consumed from the message broker.
- The *indexing* tier moves the processed data into Elasticsearch.

Any of these layers can be scaled by adding computing resources. Examine the performance of these components regularly as your use case evolves and add resources as needed. When Logstash routinely throttles incoming events, consider adding storage for your message broker. Alternately, increase the Elasticsearch cluster's rate of data consumption by adding more Logstash indexing instances.

Performance Tuning

This section includes the following information about tuning Logstash performance:

- [Performance Troubleshooting Guide](#)
- [Tuning and Profiling Logstash Performance](#)

Performance Troubleshooting Guide

You can use this troubleshooting guide to quickly diagnose and resolve Logstash performance problems. Advanced knowledge of pipeline internals is not required to understand this guide. However, the [pipeline documentation](#) is recommended reading if you want to go beyond this guide.

You may be tempted to jump ahead and change settings like `pipeline.workers` (`-w`) as a first attempt to improve performance. In our experience, changing this setting makes it more difficult to troubleshoot performance problems because you increase the number of variables in play. Instead, make one change at a time and measure the results. Starting at the end of this list is a sure-fire way to create a confusing situation.

Performance Checklist

1. Check the performance of input sources and output destinations:

- Logstash is only as fast as the services it connects to. Logstash can only consume and produce data as fast as its input and output destinations can!

2. Check system statistics:

- CPU
 - Note whether the CPU is being heavily used. On Linux/Unix, you can run `top -H` to see process statistics broken out by thread, as well as total CPU statistics.
 - If CPU usage is high, skip forward to the section about checking the JVM heap and then read the section about tuning Logstash worker settings.
- Memory
 - Be aware of the fact that Logstash runs on the Java VM. This means that Logstash will always use the maximum amount of memory you allocate to it.
 - Look for other applications that use large amounts of memory and may be causing Logstash to swap to disk. This can happen if the total memory used by applications exceeds physical memory.
- I/O Utilization
 - Monitor disk I/O to check for disk saturation.
 - Disk saturation can happen if you're using Logstash plugins (such as the file output) that may saturate your storage.
 - Disk saturation can also happen if you're encountering a lot of errors that force Logstash to generate large error logs.
 - On Linux, you can use iostat, dstat, or something similar to monitor disk I/O.
 - Monitor network I/O for network saturation.
 - Network saturation can happen if you're using inputs/outputs that perform a lot of network operations.
 - On Linux, you can use a tool like dstat or iftop to monitor your network.

3. Check the JVM heap:

- Often times CPU utilization can go through the roof if the heap size is too low, resulting in the JVM constantly garbage collecting.

- A quick way to check for this issue is to double the heap size and see if performance improves. Do not increase the heap size past the amount of physical memory. Leave at least 1GB free for the OS and other processes.
- You can make more accurate measurements of the JVM heap by using either the `jmap` command line utility distributed with Java or by using VisualVM. For more info, see [the section called “Profiling the Heap”](#).

4. Tune Logstash worker settings:

- Begin by scaling up the number of pipeline workers by using the `-w` flag. This will increase the number of threads available for filters and outputs. It is safe to scale this up to a multiple of CPU cores, if need be, as the threads can become idle on I/O.
- Each output can only be active in a single pipeline worker thread by default. You can increase this by changing the `workers` setting in the configuration block for each output. Never make this value larger than the number of pipeline workers.
- You may also tune the output batch size. For many outputs, such as the Elasticsearch output, this setting will correspond to the size of I/O operations. In the case of the Elasticsearch output, this setting corresponds to the batch size.

Tuning and Profiling Logstash Performance

The Logstash defaults are chosen to provide fast, safe performance for most users. However if you notice performance issues, you may need to modify some of the defaults. Logstash provides the following configurable options for tuning pipeline performance: `pipeline.workers`, `pipeline.batch.size`, and `pipeline.batch.delay`. For more information about setting these options, see [Settings File](#).

Make sure you've read the [Performance Troubleshooting Guide](#) before modifying these options.

- The `pipeline.workers` setting determines how many threads to run for filter and output processing. If you find that events are backing up, or that the CPU is not saturated, consider increasing the value of this parameter to make better use of available processing power. Good results can even be found increasing this number past the number of available processors as these threads may spend significant time in an I/O wait state when writing to external systems. Legal values for this parameter are positive integers.
- The `pipeline.batch.size` setting defines the maximum number of events an individual worker thread collects before attempting to execute filters and outputs. Larger batch sizes are generally more efficient, but increase memory overhead. Some hardware configurations require you to increase JVM heap size by setting the `LS_HEAP_SIZE` variable to avoid performance degradation with this option. Values of this parameter in excess of the optimum range cause performance degradation due to frequent garbage collection or JVM crashes related to out-of-memory exceptions. Output plugins can process each batch as a logical unit. The Elasticsearch output, for example, issues [bulk requests](#) for each batch received. Tuning the `pipeline.batch.size` setting adjusts the size of bulk requests sent to Elasticsearch.
- The `pipeline.batch.delay` setting rarely needs to be tuned. This setting adjusts the latency of the Logstash pipeline. Pipeline batch delay is the maximum amount of time in milliseconds that Logstash waits for new messages after receiving an event in the current pipeline worker thread. After this time elapses, Logstash begins to execute filters and outputs. The maximum time that Logstash waits between receiving an event and processing that event in a filter is the product of the `pipeline.delay` and `pipeline.batch.size` settings.

[Notes on Pipeline Configuration and Performance](#)

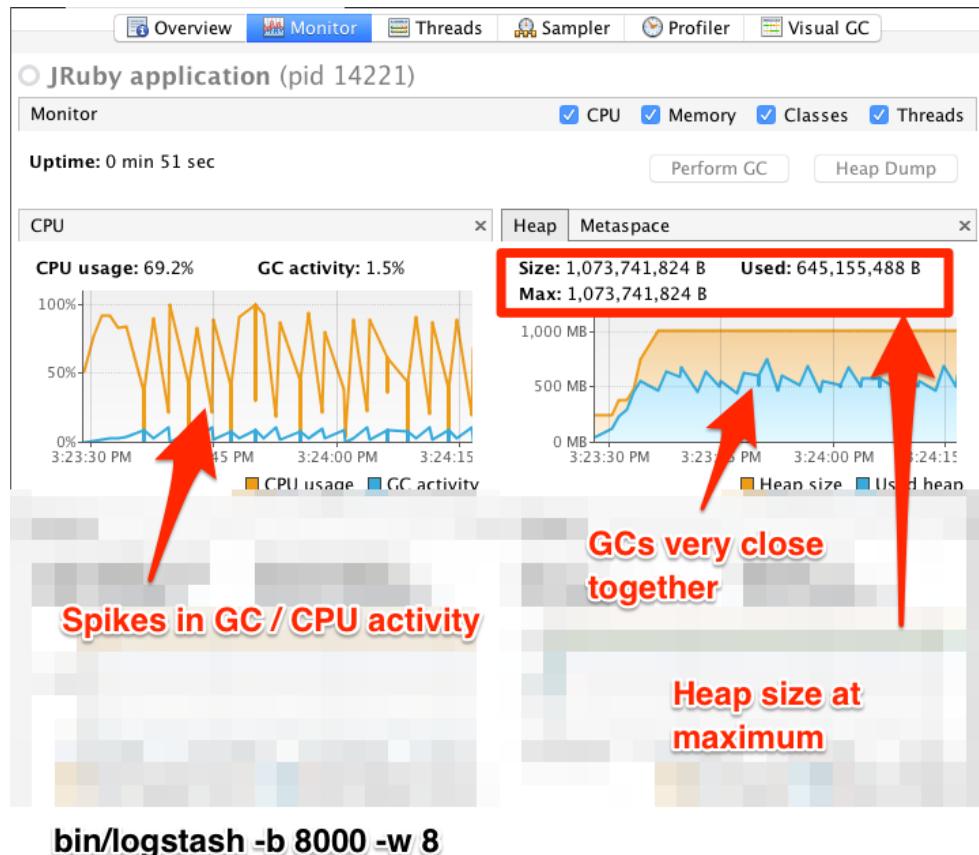
If you plan to modify the default pipeline settings, take into account the following suggestions:

- The total number of inflight events is determined by the product of the `pipeline.workers` and `pipeline.batch.size` settings. This product is referred to as the *inflight count*. Keep the value of the inflight count in mind as you adjust the `pipeline.workers` and `pipeline.batch.size` settings. Pipelines that intermittently receive large events at irregular intervals require sufficient memory to handle these spikes. Configure the `LS_HEAP_SIZE` variable accordingly.
- Measure each change to make sure it increases, rather than decreases, performance.
- Ensure that you leave enough memory available to cope with a sudden increase in event size. For example, an application that generates exceptions that are represented as large blobs of text.

- The number of workers may be set higher than the number of CPU cores since outputs often spend idle time in I/O wait conditions.
- Threads in Java have names and you can use the `jstack`, `top`, and the VisualVM graphical tools to figure out which resources a given thread uses.
- On Linux platforms, Logstash labels all the threads it can with something descriptive. For example, inputs show up as `[base]<inputname`, filter/output workers show up as `[base]>workerN`, where N is an integer. Where possible, other threads are also labeled to help you identify their purpose.

Profiling the Heap

When tuning Logstash you may have to adjust the heap size. You can use the [VisualVM](#) tool to profile the heap. The **Monitor** pane in particular is useful for checking whether your heap allocation is sufficient for the current workload. The screenshots below show sample **Monitor** panes. The first pane examines a Logstash instance configured with too many inflight events. The second pane examines a Logstash instance configured with an appropriate amount of inflight events. Note that the specific batch sizes used here are most likely not applicable to your specific workload, as the memory demands of Logstash vary in large part based on the type of messages you are sending.





In the first example we see that the CPU isn't being used very efficiently. In fact, the JVM is often times having to stop the VM for "full GCs". Full garbage collections are a common symptom of excessive memory pressure. This is visible in the spiky pattern on the CPU chart. In the more efficiently configured example, the GC graph pattern is more smooth, and the CPU is used in a more uniform manner. You can also see that there is ample headroom between the allocated heap size, and the maximum allowed, giving the JVM GC a lot of room to work with.

Examining the in-depth GC statistics with a tool similar to the excellent [VisualGC](#) plugin shows that the over-allocated VM spends very little time in the efficient Eden GC, compared to the time spent in the more resource-intensive Old Gen "Full" GCs.

NOTE: As long as the GC pattern is acceptable, heap sizes that occasionally increase to the maximum are acceptable. Such heap size spikes happen in response to a burst of large events passing through the pipeline. In general practice, maintain a gap between the used amount of heap memory and the maximum. This document is not a comprehensive guide to JVM GC tuning. Read the official [Oracle guide](#) for more information on the topic. We also recommend reading [Debugging Java Performance](#).

Monitoring APIs

WARNING!!! This functionality is experimental and may be changed or removed completely in a future release. Elastic will take a best effort approach to fix any issues, but experimental features are not subject to the support SLA of official GA features.

Logstash provides the following monitoring APIs to retrieve runtime metrics about Logstash:

- [Node Info API](#)
- [Plugins Info API](#)
- [Node Stats API](#)
- [Hot Threads API](#)

You can use the root resource to retrieve general information about the Logstash instance, including the host and version.

```
GET /
```

Example response:

```
{  
  "host": "skywalker",  
  "version": "5.2.1",  
  "http_address": "127.0.0.1:9600"  
}
```

NOTE: By default, the monitoring API attempts to bind to `tcp:9600`. If this port is already in use by another Logstash instance, you need to launch Logstash with the `--http.port` flag specified to bind to a different port. See [Command-Line Flags](#) for more information.

Common Options

The following options can be applied to all of the Logstash monitoring APIs.

Pretty Results

When appending `?pretty=true` to any request made, the JSON returned will be pretty formatted (use it for debugging only!).

Human-Readable Output

NOTE: For Logstash 5.2.1, the `human` option is supported for the [Hot Threads API](#) only. When you specify `human=true`, the results are returned in plain text instead of JSON format. The default is false.

Statistics are returned in a format suitable for humans (eg `"exists_time": "1h"` or `"size": "1kb"`) and for computers (eg `"exists_time_in_millis": 3600000` or `"size_in_bytes": 1024`). The human-readable values can be turned off by adding `?human=false` to the query string. This makes sense when the stats results are being consumed by a monitoring tool, rather than intended for human consumption. The default for the `human` flag is `false`.

Node Info API

NOTE: This functionality is experimental and may be changed or removed completely in a future release. Elastic will take a best effort approach to fix any issues, but experimental features are not subject to the support SLA of official GA features.

The node info API retrieves information about the node.

```
GET /_node/<types>
```

Where `<types>` is optional and specifies the types of node info you want to return.

You can limit the info that's returned by combining any of the following types in a comma-separated list:

- [`pipeline`](#) Gets pipeline-specific information and settings.
- [`os`](#) Gets node-level info about the OS.
- [`jvm`](#) Gets node-level JVM info, including info about threads.

See [Common Options](#) for a list of options that can be applied to all Logstash monitoring APIs.

Pipeline Info

The following request returns a JSON document that shows pipeline info, such as the number of workers, batch size, and batch delay:

```
GET /_node/pipeline
```

If you want to view additional information about the pipeline, such as stats for each configured input, filter, or output stage, see the [Pipeline Stats](#) section under the [Node Stats API](#).

Example response:

```
{  
  "pipeline": {  
    "workers": 8,  
    "batch_size": 125,  
    "batch_delay": 5,  
    "config_reload_automatic": true,  
    "config_reload_interval": 3  
  }  
}
```

OS Info

The following request returns a JSON document that shows the OS name, architecture, version, and available processors:

```
GET /_node/os
```

Example response:

```
{  
  "os": {  
    "name": "Mac OS X",  
    "arch": "x86_64",  
    "version": "10.12.1",  
    "available_processors": 8  
  }  
}
```

JVM Info

The following request returns a JSON document that shows node-level JVM stats, such as the JVM process id, version, VM info, memory usage, and info about garbage collectors:

```
GET /_node/jvm
```

Example response:

```
{  
  "jvm": {  
    "pid": 59616,  
    "version": "1.8.0_65",  
    "vm_name": "Java HotSpot(TM) 64-Bit Server VM",  
    "vm_version": "1.8.0_65",  
    "vm_vendor": "Oracle Corporation",  
    "start_timeInMillis": 1484251185878,  
    "mem": {  
      "heap_init_in_bytes": 268435456,  
      "heap_max_in_bytes": 1037959168,  
      "non_heap_init_in_bytes": 2555904,  
      "non_heap_max_in_bytes": 0  
    },  
    "gc_collectors": [  
      "ParNew",  
      "ConcurrentMarkSweep"  
    ]  
  }  
}
```

Plugins Info API

WARNING!!! This functionality is experimental and may be changed or removed completely in a future release. Elastic will take a best effort approach to fix any issues, but experimental features are not subject to the support SLA of official GA features.

The plugins info API gets information about all Logstash plugins that are currently installed. This API basically returns the output of running the `bin/logstash-plugin list --verbose` command.

```
GET /_node/plugins
```

See [Common Options](#) for a list of options that can be applied to all Logstash monitoring APIs.

The output is a JSON document.

Example response:

```
{
  "total": 92,
  "plugins": [
    {
      "name": "logstash-codec-cef",
      "version": "4.1.2"
    },
    {
      "name": "logstash-codec-collectd",
      "version": "3.0.3"
    },
    {
      "name": "logstash-codec-dots",
      "version": "3.0.2"
    },
    {
      "name": "logstash-codec-edn",
      "version": "3.0.2"
    },
    .
    .
    .
  ]
}
```

Node Stats API

WARNING!!! This functionality is experimental and may be changed or removed completely in a future release. Elastic will take a best effort approach to fix any issues, but experimental features are not subject to the support SLA of official GA features.

The node stats API retrieves runtime stats about Logstash.

```
GET /_node/stats/<types>
```

Where `<types>` is optional and specifies the types of stats you want to return.

By default, all stats are returned. You can limit the info that's returned by combining any of the following types in a comma-separated list:

- `jvm` Gets JVM stats, including stats about threads, memory usage, garbage collectors, and uptime.
- `process` Gets process stats, including stats about file descriptors, memory consumption, and CPU usage.
- `pipeline` Gets runtime stats about the Logstash pipeline.
- `reloads` Gets runtime stats about config reload successes and failures.
- `os` Gets runtime stats about cgroups when Logstash is running in a container.

See [Common Options](#) for a list of options that can be applied to all Logstash monitoring APIs.

JVM Stats

The following request returns a JSON document containing JVM stats:

```
GET /_node/stats/jvm
```

Example response:

```
{
  "jvm": {
    "threads": {
      "count": 35,
      "peak_count": 36
    },
    "mem": {
      "heap_used_in_bytes": 318691184,
      "heap_used_percent": 15,
      "heap_committed_in_bytes": 519045120,
      "heap_max_in_bytes": 2075918336,
      ...
    }
  }
}
```

Logstash 5.2 Configuration Guide

```
"non_heap_used_in_bytes": 189382304,  
"non_heap_committed_in_bytes": 200728576,  
"pools": {  
    "survivor": {  
        "peak_used_in_bytes": 8912896,  
        "used_in_bytes": 9538656,  
        "peak_max_in_bytes": 35782656,  
        "max_in_bytes": 71565312,  
        "committed_in_bytes": 17825792  
    },  
    "old": {  
        "peak_used_in_bytes": 106946320,  
        "used_in_bytes": 181913072,  
        "peak_max_in_bytes": 715849728,  
        "max_in_bytes": 1431699456,  
        "committed_in_bytes": 357957632  
    },  
    "young": {  
        "peak_used_in_bytes": 71630848,  
        "used_in_bytes": 127239456,  
        "peak_max_in_bytes": 286326784,  
        "max_in_bytes": 572653568,  
        "committed_in_bytes": 143261696  
    }  
},  
"gc": {  
    "collectors": {  
        "old": {  
            "collection_time_in_millis": 58,  
            "collection_count": 2  
        },  
        "young": {  
            "collection_time_in_millis": 338,  
            "collection_count": 26  
        }  
    },  
    "uptime_in_millis": 382701  
}
```

Process Stats

The following request returns a JSON document containing process stats:

```
GET /_node/stats/process
```

Example response:

```
{  
    "process": {  
        "open_file_descriptors": 164,  
        "peak_open_file_descriptors": 166,  
        "max_file_descriptors": 10240,  
        "mem": {
```

```
        "total_virtual_in_bytes": 5399474176
    },
    "cpu": {
        "total_in_millis": 72810537000,
        "percent": 0,
        "load_average": {
            "1m": 2.41943359375
        }
    }
}
```

Pipeline Stats

The following request returns a JSON document containing pipeline stats, including:

- the number of events that were input, filtered, or output by the pipeline
- stats for each configured filter or output stage
- info about config reload successes and failures (when [config reload](#) is enabled)
- info about the persistent queue (when [persistent queues](#) are enabled)

NOTE: Detailed pipeline stats for input plugins are not currently available, but will be available in a future release. For now, the node stats API returns an empty set array for inputs ("inputs": []).

```
GET /_node/stats/pipeline
```

Example response:

```
{
    "pipeline": {
        "events": {
            "duration_in_millis": 6304989,
            "in": 200,
            "filtered": 200,
            "out": 200
        },
        "plugins": {
            "inputs": [],
            "filters": [
                {
                    "id": "4e3d4bed6ba821ebb47f4752bb757b04a754d736-2",
                    "events": {
                        "duration_in_millis": 113,
                        "in": 200,
                        "out": 200
                    },
                    "matches": 200,
                    "patterns_per_field": {

```

```

        "message": 1
    },
    "name": "grok"
},
{
    "id": "4e3d4bed6ba821ebb47f4752bb757b04a754d736-3",
    "events": {
        "duration_in_millis": 526,
        "in": 200,
        "out": 200
    },
    "name": "geoip"
}
],
"outputs": [
{
    "id": "4e3d4bed6ba821ebb47f4752bb757b04a754d736-4",
    "events": {
        "duration_in_millis": 2312,
        "in": 200,
        "out": 200
    },
    "name": "stdout"
}
]
},
" reloads": {
    "last_error": null,
    "successes": 0,
    "last_success_timestamp": null,
    "last_failure_timestamp": null,
    "failures": 0
},
"queue": {
    "events": 26,
    "type": "persisted",
    "capacity": {
        "page_capacity_in_bytes": 262144000,
        "max_queue_size_in_bytes": 4294967296,
        "max_unread_events": 0
    },
    "data": {
        "path": "/path/to/data/queue",
        "free_space_in_bytes": 123027787776,
        "storage_type": "hfs"
    }
}
}
}
```

Reload Stats

The following request returns a JSON document that shows info about config reload successes and failures.

GET /_node/stats/reloads

Example response:

```
{  
  "reloads": {  
    "successes": 0,  
    "failures": 0  
  }  
}
```

OS Stats

When Logstash is running in a container, the following request returns a JSON document that contains cgroup information to give you a more accurate view of CPU load, including whether the container is being throttled.

GET /_node/stats/os

Example response:

```
{  
  "os" : {  
    "cgroup" : {  
      "cpuacct" : {  
        "control_group" : "/elastic1",  
        "usage_nanos" : 378477588075  
      },  
      "cpu" : {  
        "control_group" : "/elastic1",  
        "cfs_period_micros" : 1000000,  
        "cfs_quota_micros" : 800000,  
        "stat" : {  
          "number_of_elapsed_periods" : 4157,  
          "number_of_times_throttled" : 460,  
          "time_throttled_nanos" : 581617440755  
        }  
      }  
    }  
  }  
}
```

Hot Threads API

WARNING!!! This functionality is experimental and may be changed or removed completely in a future release. Elastic will take a best effort approach to fix any issues, but experimental features are not subject to the support SLA of official GA features.

The hot threads API gets the current hot threads for Logstash. A hot thread is a Java thread that has high CPU usage and executes for a longer than normal period of time.

```
GET /_node/hot_threads
```

The output is a JSON document that contains a breakdown of the top hot threads for Logstash.

Example response:

```
{
  "time": "2017-01-12T12:09:45-08:00",
  "busiest_threads": 3,
  "threads": [
    {
      "name": "Logstash::Runner",
      "percent_of_cpu_time": 1.07,
      "state": "timed_waiting",
      "traces": [
        "java.lang.Object.wait(Native Method)",
        "java.lang.Thread.join(Thread.java:1253)",

        "org.jruby.internal.runtime.NativeThread.join(NativeThread.java:75)",
        "org.jruby.RubyThread.join(RubyThread.java:697)",

        "org.jruby.RubyThread$INVOKER$i$0$1$join.call(RubyThread$INVOKER$i$0$1$join.gen)",

        "org.jruby.internal.runtime.methods.JavaMethod$JavaMethodN.call(JavaMethod.java:663)",

        "org.jruby.internal.runtime.methods.DynamicMethod.call(DynamicMethod.java:198)",

        "org.jruby.runtime.callsite.CachingCallSite.cacheAndCall(CachingCallSite.java:306)",

        "org.jruby.runtime.callsite.CachingCallSite.call(CachingCallSite.java:136)",
        "org.jruby.ast.CallNoArgNode.interpret(CallNoArgNode.java:60)"
      ],
    },
    {
  }
```

Logstash 5.2 Configuration Guide

```
"name": "[main]>worker7",
"percent_of_cpu_time": 0.71,
"state": "waiting",
"traces": [
    "sun.misc.Unsafe.park(Native Method)",
    "java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)",
    "java.util.concurrent.locks.AbstractQueuedSynchronizer.parkAndCheckInterrupt(
AbstractQueuedSynchronizer.java:836)",
    "java.util.concurrent.locks.AbstractQueuedSynchronizer.doAcquireInterruptibly(
AbstractQueuedSynchronizer.java:897)",
    "java.util.concurrent.locks.AbstractQueuedSynchronizer.acquireInterruptibly(A
bstractQueuedSynchronizer.java:1222)",
    "java.util.concurrent.locks.ReentrantLock.lockInterruptibly(ReentrantLock.jav
a:335)",
        "org.jruby.RubyThread.lockInterruptibly(RubyThread.java:1470)",
        "org.jruby.ext.thread.Mutex.lock(Mutex.java:91)",
        "org.jruby.ext.thread.Mutex.synchronize(Mutex.java:147)",
    "org.jruby.ext.thread.Mutex$INVOKER$i$0$0$synchronize.call(Mutex$INVOKER$i$0$0
$synchronize.gen)"
],
},
{
    "name": "[main]>worker3",
    "percent_of_cpu_time": 0.71,
    "state": "waiting",
    "traces": [
        "sun.misc.Unsafe.park(Native Method)",
        "java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)",
        "java.util.concurrent.locks.AbstractQueuedSynchronizer.parkAndCheckInterrupt(
AbstractQueuedSynchronizer.java:836)",
        "java.util.concurrent.locks.AbstractQueuedSynchronizer.doAcquireInterruptibly(
AbstractQueuedSynchronizer.java:897)",
        "java.util.concurrent.locks.AbstractQueuedSynchronizer.acquireInterruptibly(A
bstractQueuedSynchronizer.java:1222)",
        "java.util.concurrent.locks.ReentrantLock.lockInterruptibly(ReentrantLock.jav
a:335)",
            "org.jruby.RubyThread.lockInterruptibly(RubyThread.java:1470)",
            "org.jruby.ext.thread.Mutex.lock(Mutex.java:91)",
            "org.jruby.ext.thread.Mutex.synchronize(Mutex.java:147)",
    "org.jruby.ext.thread.Mutex$INVOKER$i$0$0$synchronize.call(Mutex$INVOKER$i$0$0
$synchronize.gen)"
]
}
]
```

}

The parameters allowed are:

threads	The number of hot threads to return. The default is 3.
human	If true, returns plain text instead of JSON format. The default is false.
ignore_idle_threads	If true, does not return idle threads. The default is true.

See [Common Options](#) for a list of options that can be applied to all Logstash monitoring APIs.

You can use the `?human` parameter to return the document in a human-readable format.

```
GET /_node/hot_threads?human=true
```

Example of a human-readable response:

```
::: {}
Hot threads at 2017-01-12T12:10:15-08:00, busiestThreads=3:
=====
===
1.02 % of cpu usage, state: timed_waiting, thread name: 'Logstash::Runner'
    java.lang.Object.wait(Native Method)
    java.lang.Thread.join(Thread.java:1253)
    org.jruby.internal.runtime.NativeThread.join(NativeThread.java:75)
    org.jruby.RubyThread.join(RubyThread.java:697)

org.jruby.RubyThread$INVOKER$i$0$1$join.call(RubyThread$INVOKER$i$0$1$join.ge
n)

org.jruby.internal.runtime.methods.JavaMethod$JavaMethodN.call(JavaMethod.jav
a:663)

org.jruby.internal.runtime.methods.DynamicMethod.call(DynamicMethod.java:198)

org.jruby.runtime.callsite.CachingCallSite.cacheAndCall(CachingCallSite.java:
306)

org.jruby.runtime.callsite.CachingCallSite.call(CachingCallSite.java:136)
    org.jruby.ast.CallNoArgNode.interpret(CallNoArgNode.java:60)
-----
===
0.71 % of cpu usage, state: waiting, thread name: '[main]>worker7'
    sun.misc.Unsafe.park(Native Method)
    java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)

java.util.concurrent.locks.AbstractQueuedSynchronizer.parkAndCheckInterrupt(A
bstractQueuedSynchronizer.java:836)

java.util.concurrent.locks.AbstractQueuedSynchronizer.doAcquireInterruptibly(
AbstractQueuedSynchronizer.java:897)
```

Logstash 5.2 Configuration Guide

```
java.util.concurrent.locks.AbstractQueuedSynchronizer.acquireInterruptibly(AbstractQueuedSynchronizer.java:1222)

java.util.concurrent.locks.ReentrantLock.lockInterruptibly(ReentrantLock.java:335)
    org.jruby.RubyThread.lockInterruptibly(RubyThread.java:1470)
    org.jruby.ext.thread.Mutex.lock(Mutex.java:91)
    org.jruby.ext.thread.Mutex.synchronize(Mutex.java:147)

org.jruby.ext.thread.Mutex$INVOKER$i$0$0$synchronize.call(Mutex$INVOKER$i$0$0$synchronize.gen)
-----
-----
0.71 % of cpu usage, state: timed_waiting, thread name: '[main]>worker3'
    sun.misc.Unsafe.park(Native Method)

java.util.concurrent.locks.LockSupport.parkNanos(LockSupport.java:215)

java.util.concurrent.SynchronousQueue$TransferStack.awaitFulfill(SynchronousQueue.java:460)

java.util.concurrent.SynchronousQueue$TransferStack.transfer(SynchronousQueue.java:362)
    java.util.concurrent.SynchronousQueue.poll(SynchronousQueue.java:941)
    sun.reflect.GeneratedMethodAccessor6.invoke(Unknown Source)

sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    java.lang.reflect.Method.invoke(Method.java:497)

org.jruby.javasupport.JavaMethod.invokeDirectWithExceptionHandling(JavaMethod.java:466)
    org.jruby.javasupport.JavaMethod.invokeDirect(JavaMethod.java:324)
```

Working with plugins

Logstash has a rich collection of input, filter, codec and output plugins. Plugins are available as self-contained packages called gems and hosted on RubyGems.org. The plugin manager accessed via `bin/logstash-plugin` script is used to manage the lifecycle of plugins in your Logstash deployment. You can install, remove and upgrade plugins using the Command Line Interface (CLI) invocations described below.

Listing plugins

Logstash release packages bundle common plugins so you can use them out of the box. To list the plugins currently available in your deployment:

```
bin/logstash-plugin list
```

```
bin/logstash-plugin list --verbose
```

```
bin/logstash-plugin list '*namefragment*'
```

```
bin/logstash-plugin list --group output
```

Will list all installed plugins

Will list installed plugins with version information

Will list all installed plugins containing a namefragment

Will list all installed plugins for a particular group (input, filter, codec, output)

Adding plugins to your deployment

The most common situation when dealing with plugin installation is when you have access to internet. Using this method, you will be able to retrieve plugins hosted on the public repository (RubyGems.org) and install on top of your Logstash installation.

```
bin/logstash-plugin install logstash-output-kafka
```

Once the plugin is successfully installed, you can start using it in your configuration file.

Advanced: Adding a locally built plugin

In some cases, you want to install plugins which have not yet been released and not hosted on RubyGems.org. Logstash provides you the option to install a locally built plugin which is packaged as a ruby gem. Using a file location:

```
bin/logstash-plugin install /path/to/logstash-output-kafka-1.0.0.gem
```

Advanced: Using --path.plugins

Using the Logstash `--path.plugins` flag, you can load a plugin source code located on your file system. Typically this is used by developers who are iterating on a custom plugin and want to test it before creating a ruby gem.

The path needs to be in a specific directory hierarchy: `PATH/logstash/TYPE/NAME.rb`, where `TYPE` is *inputs filters*, *outputs* or *codecs* and `NAME` is the name of the plugin.

```
# supposing the code is in /opt/shared/lib/logstash/inputs/my-custom-plugin-
code.rb
bin/logstash --path.plugins /opt/shared/lib
```

Updating plugins

Plugins have their own release cycle and are often released independent of Logstash's core release cycle. Using the update subcommand you can get the latest or update to a particular version of the plugin.

```
bin/logstash-plugin update
```

```
bin/logstash-plugin update logstash-output-kafka
```

will update all installed plugins

will update only this plugin

Removing plugins

If you need to remove plugins from your Logstash installation:

```
bin/logstash-plugin remove logstash-output-kafka
```

Proxy Support

The previous sections relied on Logstash being able to communicate with RubyGems.org. In certain environments, Forwarding Proxy is used to handle HTTP requests. Logstash Plugins can be installed and updated through a Proxy by setting the `HTTP_PROXY` environment variable:

```
export HTTP_PROXY=http://127.0.0.1:3128
```

```
bin/logstash-plugin install logstash-output-kafka
```

Once set, plugin commands `install`, `update` can be used through this proxy.

Generating Plugins

You can now create your own Logstash plugin in seconds! The generate subcommand of `bin/logstash-plugin` creates the foundation for a new Logstash plugin with templated files. It creates the correct directory structure, gemspec files, and dependencies so you can start adding custom code to process data with Logstash.

Example Usage

```
bin/logstash-plugin generate --type input --name xkcd --path  
~/ws/elastic/plugins
```

- `--type`: Type of plugin - input, filter, output, or codec
- `--name`: Name for the new plugin
- `--path`: Directory path where the new plugin structure will be created. If not specified, it will be created in the current directory.

Offline Plugin Management

The Logstash [plugin manager](#) provides support for preparing offline plugin packs that you can use to install Logstash plugins on systems that don't have Internet access.

This procedure requires a staging machine running Logstash that has access to a public or [private Rubygems](#) server. The staging machine downloads and packages all the files and dependencies required for offline installation.

NOTE: If you used offline plugin management prior to Logstash 5.2, you used the `pack` and `unpack` subcommands. Those subcommands are now deprecated, but the procedure for using them is still available in the documentation [here](#).

Building Offline Plugin Packs

An *offline plugin pack* is a compressed file that contains all the plugins your offline Logstash installation requires, along with the dependencies for those plugins.

To build an offline plugin pack:

1. Make sure all the plugins that you want to package are installed on the staging server and that the staging server can access the Internet.
2. Run the `bin/logstash-plugin prepare-offline-pack` subcommand to package the plugins and dependencies:

```
bin/logstash-plugin prepare-offline-pack --output OUTPUT [PLUGINS]
```

where:

- `OUTPUT` specifies the location where the compressed plugin pack will be written. The default location is `/LOGSTASH_HOME/logstash-offline-plugins-5.2.1.zip`.
- `[PLUGINS]` specifies one or more plugins that you want to include in the pack.

Examples:

```
bin/logstash-plugin prepare-offline-pack logstash-input-beats
```

```
bin/logstash-plugin prepare-offline-pack logstash-filter-*
bin/logstash-plugin prepare-offline-pack logstash-filter-* logstash-input-
beats
```

Packages the Beats input plugin and any dependencies.

Uses a wildcard to package all filter plugins and any dependencies.

Packages all filter plugins, the Beats input plugin, and any dependencies.

NOTE: Downloading all dependencies for the specified plugins may take some time, depending on the plugins listed.

Installing Offline Plugin Packs

To install an offline plugin pack:

1. Move the compressed bundle to the machine where you want to install the plugins.
2. Run the `bin/logstash-plugin install` subcommand to install the packaged plugins:

```
bin/logstash-plugin install file:///path/to/logstash-offline-plugins-5.2.1.zip
```

Where `path/to/logstash-offline-plugins-5.2.1.zip` is the path to the offline plugin pack.

Updating Offline Plugins

To update offline plugins, you update the plugins on the staging server and then use the same process that you followed to build and install the plugin pack:

1. On the staging server, run the `bin/logstash-plugin update` subcommand to update the plugins. See [the section called “Updating plugins”](#).
2. Create a new version of the plugin pack. See [the section called “Building Offline Plugin Packs”](#).
3. Install the new version of the plugin pack. See [the section called “Installing Offline Plugin Packs”](#).

Building the Offline Package (Deprecated Procedure)

WARNING: Deprecated in 5.2.

Starting with Logstash 5.2, the `pack` and `unpack` commands are deprecated and replaced by the `prepare-offline-pack` and `install` commands

Working with offline plugins requires you to create an *offline package*, which is a compressed file that contains all of the plugins your offline Logstash installation requires, along with the dependencies for those plugins.

1. Create the offline package with the `bin/logstash-plugin pack` subcommand.

When you run the `bin/logstash-plugin pack` subcommand, Logstash creates a compressed bundle that contains all of the currently installed plugins and the dependencies for those plugins. By default, the compressed bundle is a GZipped TAR file when you run the `bin/logstash-plugin pack` subcommand on a UNIX machine. By default, when you run the `bin/logstash-plugin pack` subcommand on a Windows machine, the compressed bundle is a ZIP file. See [Managing Plugin Packs](#) for details on changing these default behaviors.

NOTE: Downloading all dependencies for the specified plugins may take some time, depending on the plugins listed.

2. Move the compressed bundle to the offline machines that are the source for offline plugin installation, then use the `bin/logstash-plugin unpack` subcommand to make the packaged plugins available.

Install or Update a local plugin (Deprecated Procedure)

WARNING: Deprecated in 5.2.

To install or update a local plugin, use the `--local` option with the `install` and `update` commands, as in the following examples:

Example 1. Installing a local plugin

```
bin/logstash-plugin install --local logstash-input-jdbc
```

Example 2. Updating a local plugin

```
bin/logstash-plugin update --local logstash-input-jdbc
```

Example 3. Updating all local plugins in one command

```
bin/logstash-plugin update --local
```

Managing Plugin Packs

WARNING!!! Deprecated in 5.2.

The `pack` and `unpack` subcommands for `bin/logstash-plugin` take the following options:

- tgz Generate the offline package as a GZipped TAR file. The default behavior on UNIX systems.
- zip Generate the offline package as a ZIP file. The default behavior on Windows systems.
- [packname] --override Generates a new offline package that overwrites an existing offline with the specified name. [packname] --[no-]clean: Deletes offline packages matching the specified name.

Private Gem Repositories

The Logstash plugin manager connects to a Ruby gems repository to install and update Logstash plugins. By default, this repository is <http://rubygems.org>.

Some use cases are unable to use the default repository, as in the following examples:

- A firewall blocks access to the default repository.
- You are developing your own plugins locally.
- Airgap requirements on the local system.

When you use a custom gem repository, be sure to make plugin dependencies available.

Several open source projects enable you to run your own plugin server, among them:

- [Geminabox](#)
- [Gemirro](#)
- [Gemfury](#)
- [Artifactory](#)

Editing the Gemfile

The gemfile is a configuration file that specifies information required for plugin management. Each gem file has a `source` line that specifies a location for plugin content.

By default, the gemfile's `source` line reads:

```
# This is a Logstash generated Gemfile.
# If you modify this file manually all comments and formatting will be lost.

source "https://rubygems.org"
```

To change the source, the `source` line to contain your preferred source, as in the following example:

```
# This is a Logstash generated Gemfile.
# If you modify this file manually all comments and formatting will be lost.

source "https://my.private.repository"
```

After saving the new version of the gemfile, use [plugin management commands](#) normally.

The following links contain further material on setting up some commonly used repositories:

- [Geminabox](#)
- [Artifactory](#)
- Running a [rubygems mirror](#)

Event API

This section is targeted for plugin developers and users of Logstash's Ruby filter. Below we document recent changes (starting with version 5.0) in the way users have been accessing Logstash's event based data in custom plugins and in the Ruby filter. Note that [Event Dependent Configuration](#) data flow in Logstash's config files — using [the section called “Field References”](#) — is not affected by this change, and will continue to use existing syntax.

Event Object

Event is the main object that encapsulates data flow internally in Logstash and provides an API for the plugin developers to interact with the event's content. Typically, this API is used in plugins and in a Ruby filter to retrieve data and use it for transformations. Event object contains the original data sent to Logstash and any additional fields created during Logstash's filter stages.

In 5.0, we've re-implemented the Event class and its supporting classes in pure Java. Since Event is a critical component in data processing, a rewrite in Java improves performance and provides efficient serialization when storing data on disk. For the most part, this change aims at keeping backward compatibility and is transparent to the users. To this extent we've updated and published most of the plugins in Logstash's ecosystem to adhere to the new API changes. However, if you are maintaining a custom plugin, or have a Ruby filter, this change will affect you. The aim of this guide is to describe the new API and provide examples to migrate to the new changes.

Event API

Prior to version 5.0, developers could access and manipulate event data by directly using Ruby hash syntax. For example, `event[field] = foo`. While this is powerful, our goal is to abstract the internal implementation details and provide well-defined getter and setter APIs.

Get API

The getter is a read-only access of field-based data in an Event.

Syntax: `event.get(field)`

Returns: Value for this field or nil if the field does not exist. Returned values could be a string, numeric or timestamp scalar value.

`field` is a structured field sent to Logstash or created after the transformation process. `field` can also be a nested field reference such as `[field][bar]`.

Examples:

```
event.get("foo") # => "baz"
```

Logstash 5.2 Configuration Guide

```
event.get("[foo]") # => "zab"
event.get("[foo][bar]") # => 1
event.get("[foo][bar]") # => 1.0
event.get("[foo][bar]") # => [1, 2, 3]
event.get("[foo][bar]") # => {"a" => 1, "b" => 2}
event.get("[foo][bar]") # => {"a" => 1, "b" => 2, "c" => [1, 2]}
```

Accessing @metdata

```
event.get("@metadata") # => "baz"
```

Set API

This API can be used to mutate data in an Event.

Syntax: `event.set(field, value)`

Returns: The current Event after the mutation, which can be used for chainable calls.

Examples:

```
event.set("foo", "baz")
event.set("[foo]", "zab")
event.set("[foo][bar]", 1)
event.set("[foo][bar]", 1.0)
event.set("[foo][bar]", [1, 2, 3])
event.set("[foo][bar]", {"a" => 1, "b" => 2})
event.set("[foo][bar]", {"a" => 1, "b" => 2, "c" => [1, 2]})
event.set("@metadata", "baz")
```

Mutating a collection after setting it in the Event has an undefined behaviour and is not allowed.

```
h = {"a" => 1, "b" => 2, "c" => [1, 2]}
event.set("[foo][bar]", h)

h["c"] = [3, 4]
event.get("[foo][bar][c]") # => undefined
```

Suggested way of mutating collections:

```
h = {"a" => 1, "b" => 2, "c" => [1, 2]}
event.set("[foo][bar]", h)

h["c"] = [3, 4]
event.set("[foo][bar]", h)

# Alternatively,
event.set("[foo][bar][c]", [3, 4])
```

Ruby Filter

The [Ruby Filter](#) can be used to execute any ruby code and manipulate event data using the API described above. For example, using the new API:

Logstash 5.2 Configuration Guide

```
filter {
  ruby {
    code => 'event.set("lowercase_field", event.get("message").downcase)'
  }
}
```

This filter will lowercase the `message` field, and set it to a new field called `lowercase_field`

Input plugins

An input plugin enables a specific source of events to be read by Logstash.

The following input plugins are available below. For a list of Elastic supported plugins, please consult the [Support Matrix](#).

Plugin	Description	Github repository
beats	Receives events from the Elastic Beats framework	logstash-input-beats
cloudwatch	Pulls events from the Amazon Web Services CloudWatch API	logstash-input-cloudwatch
couchdb_changes	Streams events from CouchDB's _changes URI	logstash-input-couchdb_changes
drupal_dblog	Retrieves watchdog log events from Drupal installations with DBLog enabled	logstash-input-drupal_dblog
elasticsearch	Reads query results from an Elasticsearch cluster	logstash-input-elasticsearch
eventlog	Pulls events from the Windows Event Log	logstash-input-eventlog
exec	Captures the output of a shell command as an event	logstash-input-exec
file	Streams events from files	logstash-input-file
ganglia	Reads Ganglia packets over UDP	logstash-input-ganglia
gelf	Reads GELF-format messages from Graylog2 as events	logstash-input-gelf
gemfire	Pushes events to a GemFire region	logstash-input-gemfire
generator	Generates random log events for test purposes	logstash-input-generator
github	Reads events from a GitHub webhook	logstash-input-github
graphite	Reads metrics from the graphite tool	logstash-input-graphite
heartbeat	Generates heartbeat events for testing	logstash-input-heartbeat
heroku	Streams events from the logs of a Heroku app	logstash-input-heroku
http	Receives events over HTTP or HTTPS	logstash-input-http
http_poller	Decodes the output of an HTTP API into events	logstash-input-http_poller
imap	Reads mail from an IMAP server	logstash-input-imap
irc	Reads events from an IRC server	logstash-input-irc
jdbc	Creates events from JDBC data	logstash-input-jdbc
jmx	Retrieves metrics from remote Java applications over JMX	logstash-input-jmx

kafka	Reads events from a Kafka topic	logstash-input-kafka
kinesis	Receives events through an AWS Kinesis stream	logstash-input-kinesis
log4j	Reads events over a TCP socket from a Log4j SocketAppender object	logstash-input-log4j
lumberjack	Receives events using the Lumberjack protocol	logstash-input-lumberjack
meetup	Captures the output of command line tools as an event	logstash-input-meetup
pipe	Streams events from a long-running command pipe	logstash-input-pipe
puppet_facter	Receives facts from a Puppet server	logstash-input-puppet_facter
rabbitmq	Pulls events from a RabbitMQ exchange	logstash-input-rabbitmq
rackspace	Receives events from a Rackspace Cloud Queue service	logstash-input-rackspace
redis	Reads events from a Redis instance	logstash-input-redis
relop	Receives RELP events over a TCP socket	logstash-input-relop
rss	Captures the output of command line tools as an event	logstash-input-rss
s3	Streams events from files in a S3 bucket	logstash-input-s3
salesforce	Creates events based on a Salesforce SOQL query	logstash-input-salesforce
snmptrap	Creates events based on SNMP trap messages	logstash-input-snmptrap
sqlite	Creates events based on rows in an SQLite database	logstash-input-sqlite
sqsh	Pulls events from an Amazon Web Services Simple Queue Service queue	logstash-input-sqsh
stdin	Reads events from standard input	logstash-input-stdin
stomp	Creates events received with the STOMP protocol	logstash-input-stomp
syslog	Reads syslog messages as events	logstash-input-syslog
tcp	Reads events from a TCP socket	logstash-input-tcp
twitter	Reads events from the Twitter Streaming API	logstash-input-twitter
udp	Reads events over UDP	logstash-input-udp
unix	Reads events over a UNIX socket	logstash-input-unix
varnishlog	Reads from the varnish cache shared memory log	logstash-input-varnishlog
websocket	Reads events from a websocket	logstash-input-websocket

<u>wmi</u>	Creates events based on the results of a WMI query	<u>logstash-input-wmi</u>
<u>xmpp</u>	Receives events over the XMPP/Jabber protocol	<u>logstash-input-xmpp</u>
<u>zenoss</u>	Reads Zenoss events from the fanout exchange	<u>logstash-input-zenoss</u>
<u>zeromq</u>	Reads events from a ZeroMQ SUB socket	<u>logstash-input-zeromq</u>

beats

- Version: 3.1.12
- Released on: 2016-11-30
- [Changelog](#)

This input plugin enables Logstash to receive events from the [Elastic Beats](#) framework.

The following example shows how to configure Logstash to listen on port 5044 for incoming Beats connections and to index into Elasticsearch:

```
input {
  beats {
    port => 5044
  }
}

output {
  elasticsearch {
    hosts => "localhost:9200"
    manage_template => false
    index => "%{@metadata}[beat]-%{+YYYY.MM.dd}"
    document_type => "%{@metadata}[type]"
  }
}
```

NOTE: The Beats shipper automatically sets the `type` field on the event. You cannot override this setting in the Logstash config. If you specify a setting for the [`type`](#) config option in Logstash, it is ignored.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
beats {
  port => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
cipher_suites	array	No	java.lang.String[TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256] @44b63841
client_inactivity_timeout	number	No	60
codec	codec	No	"plain"
enable_metric	boolean	No	true
host	string	No	"0.0.0.0"
id	string	No	
include_codec_tag	boolean	No	true
port	number	Yes	
ssl	boolean	No	false
ssl_certificate	a valid filesystem path	No	
ssl_certificateAuthorities	array	No	[]
ssl_handshake_timeout	number	No	10000
ssl_key	a valid filesystem path	No	
ssl_key_passphrase	password	No	
ssl_verify_mode	string , one of ["none", "peer", "force_peer"]	No	"none"
tags	array	No	
tls_max_version	number	No	1.2
tls_min_version	number	No	1
type	string	No	

Details

`add_field`

- Value type is [hash](#)
- Default value is {}

Add a field to an event

`cipher_suites`

- Value type is [array](#)
- Default value is `java.lang.String[TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256]@44b63841`

The list of ciphers suite to use, listed by priorities.

`client_inactivity_timeout`

- Value type is [number](#)
- Default value is 60

Close Idle clients after X seconds of inactivity.

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

`congestion_threshold (DEPRECATED)`

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [number](#)
- Default value is 5

The number of seconds before we raise a timeout. This option is useful to control how much time to wait if something is blocking the pipeline.

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`host`

- Value type is [string](#)
- Default value is `"0.0.0.0"`

The IP address to listen on.

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
  stdout {  
    id => "my_plugin_id"  
  }  
}
```

`include_codec_tag`

- Value type is [boolean](#)
- Default value is `true`

`port`

- This is a required setting.
- Value type is [number](#)
- There is no default value for this setting.

The port to listen on.

`ssl`

- Value type is [boolean](#)
- Default value is `false`

Events are by default sent in plain text. You can enable encryption by setting `ssl` to true and configuring the `ssl_certificate` and `ssl_key` options.

ssl_certificate

- Value type is [path](#)
- There is no default value for this setting.

SSL certificate to use.

ssl_certificateAuthorities

- Value type is [array](#)
- Default value is []

Validate client certificates against these authorities. You can define multiple files or paths. All the certificates will be read and added to the trust store. You need to configure the `ssl_verify_mode` to `peer` or `force_peer` to enable the verification.

ssl_handshake_timeout

- Value type is [number](#)
- Default value is 10000

Time in milliseconds for an incomplete ssl handshake to timeout

ssl_key

- Value type is [path](#)
- There is no default value for this setting.

SSL key to use. NOTE: This key need to be in the PKCS8 format, you can convert it with [OpenSSL](#) for more information.

ssl_key_passphrase

- Value type is [password](#)
- There is no default value for this setting.

SSL key passphrase to use.

ssl_verify_mode

- Value can be any of: `none`, `peer`, `force_peer`
- Default value is "none"

By default the server doesn't do any client verification.

`peer` will make the server ask the client to provide a certificate. If the client provides a certificate, it will be validated.

`force_peer` will make the server ask the client to provide a certificate. If the client doesn't provide a certificate, the connection will be closed.

This option needs to be used with `ssl_certificateAuthorities` and a defined list of CAs.

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

target_field_for_codec (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [string](#)
- Default value is "message"

This is the default field to which the specified codec will be applied.

tls_max_version

- Value type is [number](#)
- Default value is 1.2

The maximum TLS version allowed for the encrypted connections. The value must be the one of the following: 1.0 for TLS 1.0, 1.1 for TLS 1.1, 1.2 for TLS 1.2

tls_min_version

- Value type is [number](#)
- Default value is 1

The minimum TLS version allowed for the encrypted connections. The value must be one of the following: 1.0 for TLS 1.0, 1.1 for TLS 1.1, 1.2 for TLS 1.2

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

cloudwatch

- Version: 2.0.0
- Released on: 2016-10-22
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
`bin/logstash-plugin install logstash-input-cloudwatch`.

Pull events from the Amazon Web Services CloudWatch API.

To use this plugin, you **must** have an AWS account, and the following policy

Typically, you should setup an IAM policy, create a user and apply the IAM policy to the user. A sample policy for EC2 metrics is as follows:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1444715676000",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:GetMetricStatistics",
        "cloudwatch>ListMetrics"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Stmt1444716576170",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": "*"
    }
  ]
}
```

See <http://aws.amazon.com/iam/> for more details on setting up AWS identities.

Configuration Example

```
input {
  cloudwatch {
    namespace => "AWS/EC2"
    metrics => [ "CPUUtilization" ]
    filters => { "tag:Group" => "API-Production" }
    region => "us-east-1"
  }
}
```

```

input {
  cloudwatch {
    namespace => "AWS/EBS"
    metrics => ["VolumeQueueLength"]
    filters => { "tag:Monitoring" => "Yes" }
    region => "us-east-1"
  }
}
input {
  cloudwatch {
    namespace => "AWS/RDS"
    metrics => ["CPUUtilization", "CPUCr Usage"]
    filters => { "EngineName" => "mysql" } # Only supports EngineName,
DatabaseClass and DBInstanceIdentifier
    region => "us-east-1"
  }
}

```

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```

cloudwatch {
  filters => ...
}

```

Available configuration options:

Setting	Input type	Required	Default value
access_key_id	string	No	
add_field	hash	No	{ }
aws_credentials_file	string	No	
codec	codec	No	"plain"
combined	boolean	No	false
enable_metric	boolean	No	true
filters	array	Yes	
id	string	No	
interval	number	No	900
metrics	array	No	["CPUUtilization", "DiskReadOps", "DiskWriteOps",

Setting	Input type	Required	Default value
			"NetworkIn", "NetworkOut"]
namespace	string	No	"AWS/EC2"
period	number	No	300
proxy_uri	string	No	
region	string , one of ["us-east-1", "us-west-1", "us-west-2", "eu-central-1", "eu-west-1", "ap-southeast-1", "ap-southeast-2", "ap-northeast-1", "ap-northeast-2", "sa-east-1", "us-gov-west-1", "cn-north-1", "ap-south-1"]	No	"us-east-1"
secret_access_key	string	No	
session_token	string	No	
statistics	array	No	["SampleCount", "Average", "Minimum", "Maximum", "Sum"]
tags	array	No	
type	string	No	
use_ssl	boolean	No	true

Details

`access_key_id`

- Value type is [string](#)
- There is no default value for this setting.

This plugin uses the AWS SDK and supports several ways to get credentials, which will be tried in this order:

1. Static configuration, using `access_key_id` and `secret_access_key` params in logstash plugin config
2. External credentials file specified by `aws_credentials_file`
3. Environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`
4. Environment variables `AMAZON_ACCESS_KEY_ID` and `AMAZON_SECRET_ACCESS_KEY`
5. IAM Instance Profile (available when running inside EC2)

`add_field`

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

`aws_credentials_file`

- Value type is [string](#)
- There is no default value for this setting.

Path to YAML file containing a hash of AWS credentials. This file will only be loaded if `access_key_id` and `secret_access_key` aren't set. The contents of the file should look like this:

```
:access_key_id: "12345"
:secret_access_key: "54321"
```

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

`combined`

- Value type is [boolean](#)
- Default value is `false`

Use this for namespaces that need to combine the dimensions like S3 and SNS.

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`filters`

- This is a required setting.
- Value type is [array](#)
- There is no default value for this setting.

Specify the filters to apply when fetching resources:

This needs to follow the AWS convention of specifying filters. Instances: { *instance-id* ⇒ *i-12344321* } Tags: { "tag:Environment" ⇒ "Production" } Volumes: { *attachment.status* ⇒ *attached* } Each namespace uniquely support certain dimensions. Please consult the documentation to ensure you're using valid filters.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

*interval*

- Value type is [number](#)
- Default value is 900

Set how frequently CloudWatch should be queried

The default, 900, means check every 15 minutes. Setting this value too low (generally less than 300) results in no metrics being returned from CloudWatch.

*metrics*

- Value type is [array](#)
- Default value is ["CPUUtilization", "DiskReadOps", "DiskWriteOps", "NetworkIn", "NetworkOut"]

Specify the metrics to fetch for the namespace. The defaults are AWS/EC2 specific. See <http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/aws-namespaces.html> for the available metrics for other namespaces.

*namespace*

- Value type is [string](#)
- Default value is "AWS/EC2"

If undefined, Logstash will complain, even if codec is unused. The service namespace of the metrics to fetch.

The default is for the EC2 service. See  
<http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/aws-namespaces.html> for valid values.

*period*

- Value type is [number](#)
- Default value is 300

Set the granularity of the returned datapoints.

Must be at least 60 seconds and in multiples of 60.

*proxy\_uri*

- Value type is [string](#)
- There is no default value for this setting.

URI to proxy server if required

*region*

- Value can be any of: us-east-1, us-west-1, us-west-2, eu-central-1, eu-west-1, ap-southeast-1, ap-southeast-2, ap-northeast-1, ap-northeast-2, sa-east-1, us-gov-west-1, cn-north-1, ap-south-1
- Default value is "us-east-1"

The AWS Region

*secret\_access\_key*

- Value type is [string](#)
- There is no default value for this setting.

The AWS Secret Access Key

*session\_token*

- Value type is [string](#)
- There is no default value for this setting.

The AWS Session token for temporary credential

*statistics*

- Value type is [array](#)
- Default value is [ "SampleCount", "Average", "Minimum", "Maximum", "Sum" ]

Specify the statistics to fetch for each namespace

*tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

*type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

*use\_ssl*

- Value type is [boolean](#)
- Default value is `true`

Should we require (true) or disable (false) using SSL for communicating with the AWS API? The AWS SDK for Ruby defaults to SSL so we preserve that

## couchdb\_changes

- Version: 3.1.0
- Released on: 2016-10-13
- [Changelog](#)

This CouchDB input allows you to automatically stream events from the CouchDB [changes](#) URI. Moreover, any "future" changes will automatically be streamed as well making it easy to synchronize your CouchDB data with any target destination

# Upsert and delete You can use event metadata to allow for document deletion. All non-delete operations are treated as upserts

# Starting at a Specific Sequence The CouchDB input stores the last sequence number value in location defined by `sequence_path`. You can use this fact to start or resume the stream at a particular sequence.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
couchdb_changes {
 db => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
always_reconnect	<a href="#">boolean</a>	No	true
ca_file	a valid filesystem path	No	
codec	<a href="#">codec</a>	No	"plain"
db	<a href="#">string</a>	Yes	
enable_metric	<a href="#">boolean</a>	No	true
heartbeat	<a href="#">number</a>	No	1000
host	<a href="#">string</a>	No	"localhost"
id	<a href="#">string</a>	No	
ignore_attachments	<a href="#">boolean</a>	No	true

Setting	Input type	Required	Default value
initial_sequence	<a href="#">number</a>	No	
keep_id	<a href="#">boolean</a>	No	false
keep_revision	<a href="#">boolean</a>	No	false
password	<a href="#">password</a>	No	nil
port	<a href="#">number</a>	No	5984
reconnect_delay	<a href="#">number</a>	No	10
secure	<a href="#">boolean</a>	No	false
sequence_path	<a href="#">string</a>	No	
tags	<a href="#">array</a>	No	
timeout	<a href="#">number</a>	No	
type	<a href="#">string</a>	No	
username	<a href="#">string</a>	No	nil

## Details

[\*add\\_field\*](#)

- Value type is [hash](#)
- Default value is {}

Add a field to an event

[\*always\\_reconnect\*](#)

- Value type is [boolean](#)
- Default value is true

Reconnect flag. When true, always try to reconnect after a failure

[\*ca\\_file\*](#)

- Value type is [path](#)
- There is no default value for this setting.

Path to a CA certificate file, used to validate certificates

[\*codec\*](#)

- Value type is [codec](#)

- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

### `db`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The CouchDB db to connect to. Required parameter.

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `heartbeat`

- Value type is [number](#)
- Default value is `1000`

Logstash connects to CouchDB's \_changes with feed=continuous. The heartbeat is how often (in milliseconds) Logstash will ping CouchDB to ensure the connection is maintained. Changing this setting is not recommended unless you know what you are doing.

### `host`

- Value type is [string](#)
- Default value is "`localhost`"

IP or hostname of your CouchDB instance

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
ignore_attachments
```

- Value type is [boolean](#)
- Default value is `true`

Future feature! Until implemented, changing this from the default will not do anything.

Ignore attachments associated with CouchDB documents.

*initial\_sequence*

- Value type is [number](#)
- There is no default value for this setting.

If unspecified, Logstash will attempt to read the last sequence number from the `sequence_path` file. If that is empty or non-existent, it will begin with 0 (the beginning).

If you specify this value, it is anticipated that you will only be doing so for an initial read under special circumstances and that you will unset this value afterwards.

*keep\_id*

- Value type is [boolean](#)
- Default value is `false`

Preserve the CouchDB document id "`_id`" value in the output.

*keep\_revision*

- Value type is [boolean](#)
- Default value is `false`

Preserve the CouchDB document revision "`_rev`" value in the output.

*password*

- Value type is [password](#)
- Default value is `nil`

Password, if authentication is needed to connect to CouchDB

*port*

- Value type is [number](#)
- Default value is 5984

Port of your CouchDB instance.

*reconnect\_delay*

- Value type is [number](#)
- Default value is 10

Reconnect delay: time between reconnect attempts, in seconds.

*secure*

- Value type is [boolean](#)
- Default value is `false`

Connect to CouchDB's `_changes` feed securely (via https) Default: `false` (via http)

*sequence\_path*

- Value type is [string](#)
- There is no default value for this setting.

File path where the last sequence number in the `_changes` stream is stored. If unset it will write to `$HOME/.couchdb_seq`

*tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

*timeout*

- Value type is [number](#)
- There is no default value for this setting.

Timeout: Number of milliseconds to wait for new data before terminating the connection. If a timeout is set it will disable the heartbeat configuration option.

*type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

*username*

- Value type is [string](#)
- Default value is `nil`

Username, if authentication is needed to connect to CouchDB

## drupal\_dblog

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-input-drupal_dblog`.

Retrieve watchdog log events from a Drupal installation with DBLog enabled. The events are pulled out directly from the database. The original events are not deleted, and on every consecutive run only new events are pulled.

The last watchdog event id that was processed is stored in the Drupal variable table with the name "logstash\_last\_wid". Delete this variable or set it to 0 if you want to re-import all events.

More info on DBLog: <http://drupal.org/documentation/modules/dblog>

### Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
drupal_dblog {
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
add_usernames	<a href="#">boolean</a>	No	false
bulksize	<a href="#">number</a>	No	5000
codec	<a href="#">codec</a>	No	"plain"
databases	<a href="#">hash</a>	No	
interval	<a href="#">number</a>	No	10
tags	<a href="#">array</a>	No	
type	<a href="#">string</a>	No	"watchdog"

### Details

### `add_field`

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

### `add_usernames`

- Value type is [boolean](#)
- Default value is `false`

By default, the event only contains the current user id as a field. If you whish to add the username as an additional field, set this to true.

### `bulksize`

- Value type is [number](#)
- Default value is 5000

The amount of log messages that should be fetched with each query. Bulk fetching is done to prevent querying huge data sets when lots of messages are in the database.

### `charset (DEPRECATED)`

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value can be any of: ASCII-8BIT, Big5, Big5-HKSCS, Big5-UAO, CP949, Emacs-Mule, EUC-JP, EUC-KR, EUC-TW, GB18030, GBK, ISO-8859-1, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9, ISO-8859-10, ISO-8859-11, ISO-8859-13, ISO-8859-14, ISO-8859-15, ISO-8859-16, KOI8-R, KOI8-U, Shift\_JIS, US-ASCII, UTF-8, UTF-16BE, UTF-16LE, UTF-32BE, UTF-32LE, Windows-1251, GB2312, IBM437, IBM737, IBM775, CP850, IBM852, CP852, IBM855, CP855, IBM857, IBM860, IBM861, IBM862, IBM863, IBM864, IBM865, IBM866, IBM869, Windows-1258, GB1988, macCentEuro, macCroatian, macCyrillic, macGreek, macIceland, macRoman, macRomania, macThai, macTurkish, macUkraine, CP950, CP951, stateless-ISO-2022-JP, eucJP-ms, CP51932, GB12345, ISO-2022-JP, ISO-2022-JP-2, CP50220, CP50221, Windows-1252, Windows-1250, Windows-1256, Windows-1253, Windows-1255, Windows-1254, TIS-620, Windows-874, Windows-1257, Windows-31J, MacJapanese, UTF-7, UTF8-MAC, UTF-16, UTF-32, UTF8-DoCoMo, SJIS-DoCoMo, UTF8-KDDI, SJIS-KDDI, ISO-2022-JP-KDDI, stateless-ISO-2022-JP-KDDI, UTF8-SoftBank, SJIS-SoftBank, BINARY, CP437, CP737, CP775, IBM850, CP857, CP860, CP861, CP862, CP863, CP864, CP865, CP866, CP869, CP1258, Big5-HKSCS:2008, eucJP, euc-jp-ms, eucKR, eucTW, EUC-CN, eucCN, CP936, ISO2022-JP, ISO2022-JP2, ISO8859-1, CP1252, ISO8859-2, CP1250, ISO8859-3, ISO8859-4, ISO8859-5, ISO8859-6, CP1256, ISO8859-7, CP1253, ISO8859-

## Logstash 5.2 Configuration Guide

8, CP1255, ISO8859-9, CP1254, ISO8859-10, ISO8859-11, CP874, ISO8859-13, CP1257, ISO8859-14, ISO8859-15, ISO8859-16, CP878, CP932, csWindows31J, SJIS, PCK, MacJapan, ASCII, ANSI\_X3.4-1968, 646, CP65000, CP65001, UTF-8-MAC, UTF-8-HFS, UCS-2BE, UCS-4BE, UCS-4LE, CP1251, external, locale

- There is no default value for this setting.

The character encoding used in this input. Examples include `UTF-8` and `cp1252`

This setting is useful if your log files are in `Latin-1` (aka `cp1252`) or in another character set other than `UTF-8`.

This only affects `plain` format logs since `json` is `UTF-8` already.

### `codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

### `databases`

- Value type is [hash](#)
- There is no default value for this setting.

Specify all drupal databases that you whish to import from. This can be as many as you whish. The format is a hash, with a unique site name as the key, and a databse url as the value.

Example: [ "site1", "mysql://user1:password@host1.com/databasename", "other\_site", "mysql://user2:password@otherhost.com/databasename", ... ]

### `debug (DEPRECATED)`

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [boolean](#)
- Default value is `false`

### `format (DEPRECATED)`

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value can be any of: `plain`, `json`, `json_event`, `msgpack_event`
- There is no default value for this setting.

The format of input data (`plain`, `json`, `json_event`)

*interval*

- Value type is [number](#)
- Default value is 10

Time between checks in minutes.

*message\_format (DEPRECATED)*

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [string](#)
- There is no default value for this setting.

If format is `json`, an event `sprintf` string to build what the display `@message` should be given (defaults to the raw JSON). `sprintf` format strings look like `%{fieldname}`

If format is `json_event`, ALL fields except for `@type` are expected to be present. Not receiving all fields will cause unexpected results.

*tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

*type*

- Value type is [string](#)
- Default value is "watchdog"

Label this input with a type. Types are used mainly for filter activation.

If you create an input with type "foobar", then only filters which also have type "foobar" will act on them.

The type is also stored as part of the event itself, so you can also use the type to search for in the web interface.

[eventlog »](#)

## elasticsearch

- Version: 4.0.2
- Released on: 2017-02-07
- [Changelog](#)

Read from an Elasticsearch cluster, based on search query results. This is useful for replaying test logs, reindexing, etc.

Example:

```
input {
 # Read all documents from Elasticsearch matching the given query
 elasticsearch {
 hosts => "localhost"
 query => '{ "query": { "match": { "statuscode": 200 } }, "sort": ["_doc"] }'
 }
}
```

This would create an Elasticsearch query with the following format:

```
curl 'http://localhost:9200/logstash-*/_search?&scroll=1m&size=1000' -d '{
 "query": {
 "match": {
 "statuscode": 200
 }
 },
 "sort": ["_doc"]
}'
```

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
elasticsearch {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">ca_file</a>	a valid filesystem path	No	
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">docinfo</a>	<a href="#">boolean</a>	No	false

Setting	Input type	Required	Default value
<a href="#">docinfo_fields</a>	<a href="#">array</a>	No	[ "_index", "_type", "_id" ]
<a href="#">docinfo_target</a>	<a href="#">string</a>	No	"@metadata"
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">hosts</a>	<a href="#">array</a>	No	
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">index</a>	<a href="#">string</a>	No	"logstash-*"
<a href="#">password</a>	<a href="#">password</a>	No	
<a href="#">query</a>	<a href="#">string</a>	No	"{ \"sort\": [ \"_doc\" ] }"
<a href="#">scroll</a>	<a href="#">string</a>	No	"1m"
<a href="#">size</a>	<a href="#">number</a>	No	1000
<a href="#">ssl</a>	<a href="#">boolean</a>	No	false
<a href="#">tags</a>	<a href="#">array</a>	No	
<a href="#">type</a>	<a href="#">string</a>	No	
<a href="#">user</a>	<a href="#">string</a>	No	

## Details

### [add\\_field](#)

- Value type is [hash](#)
- Default value is { }

Add a field to an event

### [ca\\_file](#)

- Value type is [path](#)
- There is no default value for this setting.

SSL Certificate Authority file in PEM encoded format, must also include any chain certificates as necessary

### [codec](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

### `docinfo`

- Value type is [boolean](#)
- Default value is `false`

If set, include Elasticsearch document information such as index, type, and the id in the event.

It might be important to note, with regards to metadata, that if you're ingesting documents with the intent to re-index them (or just update them) that the `action` option in the `elasticsearch` output wants to know how to handle those things. It can be dynamically assigned with a field added to the metadata.

### Example

```
input {
 elasticsearch {
 hosts => "es.production.mysite.org"
 index => "mydata-2018.09.*"
 query => "*"
 size => 500
 scroll => "5m"
 docinfo => true
 }
}
output {
 elasticsearch {
 index => "copy-of-production.%{@metadata}[_index]"
 index_type => "%{@metadata}[_type]"
 document_id => "%{@metadata}[_id]"
 }
}
```

### `docinfo_fields`

- Value type is [array](#)
- Default value is `["_index", "_type", "_id"]`

List of document metadata to move to the `docinfo_target` field To learn more about Elasticsearch metadata fields read

[http://www.elasticsearch.org/guide/en/elasticsearch/guide/current/\\_document\\_metadata.html](http://www.elasticsearch.org/guide/en/elasticsearch/guide/current/_document_metadata.html)

### `docinfo_target`

- Value type is [string](#)
- Default value is `"@metadata"`

Where to move the Elasticsearch document information by default we use the `@metadata` field.

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `hosts`

- Value type is [array](#)
- There is no default value for this setting.

List of elasticsearch hosts to use for querying. each host can be either IP, HOST, IP:port or HOST:port port defaults to 9200

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

### `index`

- Value type is [string](#)
- Default value is `"logstash-*"`

The index or alias to search.

### `password`

- Value type is [password](#)
- There is no default value for this setting.

Basic Auth - password

### `query`

- Value type is [string](#)

- Default value is "`{ \"sort\": [ \"_doc\" ] }`"

The query to be executed. Read the Elasticsearch query DSL documentation for more info <https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html>

### *scroll*

- Value type is [string](#)
- Default value is "1m"

This parameter controls the keepalive time in seconds of the scrolling request and initiates the scrolling process. The timeout applies per round trip (i.e. between the previous scroll request, to the next).

### *size*

- Value type is [number](#)
- Default value is 1000

This allows you to set the maximum number of hits returned per scroll.

### *ssl*

- Value type is [boolean](#)
- Default value is `false`

## SSL

### *tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

### *type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

[\*user\*](#)

- Value type is [string](#)
- There is no default value for this setting.

Basic Auth - username

## eventlog

This input will pull events from a [Windows Event Log](#). Note that Windows Event Logs are stored on disk in a binary format and are only accessible from the Win32 API. This means Logstash needs to be running as an agent on Windows servers where you wish to collect logs from, and will not be accessible across the network.

To collect Events from the System Event Log, use a config like:

```
input {
 eventlog {
 type => 'Win32-EventLog'
 logfile => 'System'
 }
}
```

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
eventlog { }
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">logfile</a>	<a href="#">string</a>	No	"Application"
<a href="#">interval</a>	<a href="#">number</a>	No	1000
<a href="#">tags</a>	<a href="#">array</a>	No	
<a href="#">type</a>	<a href="#">string</a>	No	

## Details

[add\\_field](#)

- Value type is [hash](#)

- Default value is { }

Add a field to an event

*codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

*logfile*

- Value type is [array](#)
- Default value is ["Application", "Security", "System"]

Event Log Name. System and Security may require that privileges are given to the user running Logstash. For more information, see:

<https://social.technet.microsoft.com/forums/windowsserver/en-US/d2f813db-6142-4b5b-8d86-253ebb740473/easy-way-to-read-security-log>

*interval*

- Value type is [number](#)
- Default value is 1000

How frequently should tail check for new event logs in milliseconds

*tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

*type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

## exec

- Version: 3.1.2
- Released on: 2016-09-15
- [Changelog](#)

Periodically run a shell command and capture the whole output as an event.

Notes:

- The command field of this event will be the command run.
- The message field of this event will be the entire stdout of the command.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
exec {
 command => ...
 interval => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
codec	<a href="#">codec</a>	No	"plain"
command	<a href="#">string</a>	Yes	
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	
interval	<a href="#">number</a>	Yes	
tags	<a href="#">array</a>	No	
type	<a href="#">string</a>	No	

## Details

### `add_field`

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

### `codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

### `command`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Command to run. For example, `uptime`

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

*interval*

- This is a required setting.
- Value type is [number](#)
- There is no default value for this setting.

Interval to run the command. Value is in seconds.

*tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

*type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

## file

- Version: 4.0.0
- Released on: 2016-10-17
- [Changelog](#)

Stream events from files, normally by tailing them in a manner similar to `tail -0F` but optionally reading them from the beginning.

By default, each event is assumed to be one line and a line is taken to be the text before a newline character. Normally, logging will add a newline to the end of each line written. If you would like to join multiple log lines into one event, you'll want to use the multiline codec or filter.

The plugin aims to track changing files and emit new content as it's appended to each file. It's not well-suited for reading a file from beginning to end and storing all of it in a single event (not even with the multiline codec or filter).

### Reading from remote network volumes

The file input is not tested on remote filesystems such as NFS, Samba, s3fs-fuse, etc. These remote filesystems typically have behaviors that are very different from local filesystems and are therefore unlikely to work correctly when used with the file input.

### Tracking of current position in watched files

The plugin keeps track of the current position in each file by recording it in a separate file named `sincedb`. This makes it possible to stop and restart Logstash and have it pick up where it left off without missing the lines that were added to the file while Logstash was stopped.

By default, the `sincedb` file is placed in the home directory of the user running Logstash with a filename based on the filename patterns being watched (i.e. the `path` option). Thus, changing the filename patterns will result in a new `sincedb` file being used and any existing current position state will be lost. If you change your patterns with any frequency it might make sense to explicitly choose a `sincedb` path with the `sincedb_path` option.

A different `sincedb_path` must be used for each input. Using the same path will cause issues. The read checkpoints for each input must be stored in a different path so the information does not override.

Sincedb files are text files with four columns:

1. The inode number (or equivalent).
2. The major device number of the file system (or equivalent).
3. The minor device number of the file system (or equivalent).
4. The current byte offset within the file.

On non-Windows systems you can obtain the inode number of a file with e.g. `ls -li`.

## File rotation

File rotation is detected and handled by this input, regardless of whether the file is rotated via a rename or a copy operation. To support programs that write to the rotated file for some time after the rotation has taken place, include both the original filename and the rotated filename (e.g. `/var/log/syslog` and `/var/log/syslog.1`) in the filename patterns to watch (the `path` option). Note that the rotated filename will be treated as a new file so if `start_position` is set to *beginning* the rotated file will be reprocessed.

With the default value of `start_position` (*end*) any messages written to the end of the file between the last read operation prior to the rotation and its reopening under the new name (an interval determined by the `stat_interval` and `discover_interval` options) will not get picked up.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
file {
 path => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>add_field</code>	<a href="#">hash</a>	No	{ }
<code>close_older</code>	<a href="#">number</a>	No	3600
<code>codec</code>	<a href="#">codec</a>	No	"plain"
<code>delimiter</code>	<a href="#">string</a>	No	"\n"
<code>discover_interval</code>	<a href="#">number</a>	No	15
<code>enable_metric</code>	<a href="#">boolean</a>	No	true
<code>exclude</code>	<a href="#">array</a>	No	
<code>id</code>	<a href="#">string</a>	No	
<code>ignore_older</code>	<a href="#">number</a>	No	
<code>max_open_files</code>	<a href="#">number</a>	No	
<code>path</code>	<a href="#">array</a>	Yes	

Setting	Input type	Required	Default value
sincedb_path	<a href="#">string</a>	No	
sincedb_write_interval	<a href="#">number</a>	No	15
start_position	<a href="#">string</a> , one of ["beginning", "end"]	No	"end"
stat_interval	<a href="#">number</a>	No	1
tags	<a href="#">array</a>	No	
type	<a href="#">string</a>	No	

## Details

*add\_field*

- Value type is [hash](#)
- Default value is {}

Add a field to an event

*close\_older*

- Value type is [number](#)
- Default value is 3600

The file input closes any files that were last read the specified timespan in seconds ago. This has different implications depending on if a file is being tailed or read. If tailing, and there is a large time gap in incoming data the file can be closed (allowing other files to be opened) but will be queued for reopening when new data is detected. If reading, the file will be closed after closed\_older seconds from when the last bytes were read. The default is 1 hour

*codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

*delimiter*

- Value type is [string](#)
- Default value is "\n"

set the new line delimiter, defaults to "\n"

### `discover_interval`

- Value type is [number](#)
- Default value is 15

How often (in seconds) we expand the filename patterns in the `path` option to discover new files to watch.

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `exclude`

- Value type is [array](#)
- There is no default value for this setting.

Exclusions (matched against the filename, not full path). Filename patterns are valid here, too. For example, if you have

```
path => "/var/log/*"
```

You might want to exclude gzipped files:

```
exclude => "*.gz"
```

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

### `ignore_older`

- Value type is [number](#)
- There is no default value for this setting.

When the file input discovers a file that was last modified before the specified timespan in seconds, the file is ignored. After it's discovery, if an ignored file is modified it is no longer ignored and any new data is read. By default, this option is disabled. Note this unit is in seconds.

*max\_open\_files*

- Value type is [number](#)
- There is no default value for this setting.

What is the maximum number of file\_handles that this input consumes at any one time. Use close\_older to close some files if you need to process more files than this number. This should not be set to the maximum the OS can do because file handles are needed for other LS plugins and OS processes. The default of 4095 is set in filewatch.

*path*

- This is a required setting.
- Value type is [array](#)
- There is no default value for this setting.

The path(s) to the file(s) to use as an input. You can use filename patterns here, such as /var/log/\*.log. If you use a pattern like /var/log/\*\*/\*.log, a recursive search of /var/log will be done for all \*.log files. Paths must be absolute and cannot be relative.

You may also configure multiple paths. See an example on the [Logstash configuration page](#).

*sincedb\_path*

- Value type is [string](#)
- There is no default value for this setting.

Path of the sincedb database file (keeps track of the current position of monitored log files) that will be written to disk. The default will write sincedb files to some path matching \$HOME/.sincedb\* NOTE: it must be a file path and not a directory path

*sincedb\_write\_interval*

- Value type is [number](#)
- Default value is 15

How often (in seconds) to write a since database with the current position of monitored log files.

*start\_position*

- Value can be any of: beginning, end
- Default value is "end"

Choose where Logstash starts initially reading files: at the beginning or at the end. The default behavior treats files like live streams and thus starts at the end. If you have old data you want to import, set this to *beginning*.

This option only modifies "first contact" situations where a file is new and not seen before, i.e. files that don't have a current position recorded in a since db file read by Logstash. If a file has already been seen before, this option has no effect and the position recorded in the since db file will be used.

`stat_interval`

- Value type is [number](#)
- Default value is 1

How often (in seconds) we stat files to see if they have been modified. Increasing this interval will decrease the number of system calls we make, but increase the time to detect new log lines.

`tags`

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

`type`

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

## ganglia

- Version: 3.1.0
- Released on: 2016-12-26
- [Changelog](#)

Read ganglia packets from the network via udp

### Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
ganglia {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">host</a>	<a href="#">string</a>	No	"0.0.0.0"
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">port</a>	<a href="#">number</a>	No	8649
<a href="#">tags</a>	<a href="#">array</a>	No	
<a href="#">type</a>	<a href="#">string</a>	No	

### Details

#### [add\\_field](#)

- Value type is [hash](#)
- Default value is { }

Add a field to an event

### `codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `host`

- Value type is [string](#)
- Default value is "0.0.0.0"

The address to listen on

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

### `port`

- Value type is [number](#)
- Default value is 8649

The port to listen on. Remember that ports less than 1024 (privileged ports) may require root to use.

*tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

*type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

## gelf

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

This input will read GELF messages as events over the network, making it a good choice if you already use Graylog2 today.

The main use case for this input is to leverage existing GELF logging libraries such as the GELF log4j appender. A library used by this plugin has a bug which prevents it parsing uncompressed data. If you use the log4j appender you need to configure it like this to force gzip even for small messages:

```
<Socket name="logstash" protocol="udp" host="logstash.example.com"
port="5001">
 <GelfLayout compressionType="GZIP" compressionThreshold="1" />
</Socket>
```

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
gelf {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">host</a>	<a href="#">string</a>	No	"0.0.0.0"
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">port</a>	<a href="#">number</a>	No	12201
<a href="#">remap</a>	<a href="#">boolean</a>	No	true
<a href="#">strip_leading_underscore</a>	<a href="#">boolean</a>	No	true
<a href="#">tags</a>	<a href="#">array</a>	No	

Setting	Input type	Required	Default value
<a href="#">type</a>	<a href="#">string</a>	No	

## Details

[`add\_field`](#)

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

[`codec`](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

[`enable\_metric`](#)

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

[`host`](#)

- Value type is [string](#)
- Default value is "0.0.0.0"

The IP address or hostname to listen on.

[`id`](#)

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when

you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
port
```

- Value type is [number](#)
- Default value is 12201

The port to listen on. Remember that ports less than 1024 (privileged ports) may require root to use.

*remap*

- Value type is [boolean](#)
- Default value is `true`

Whether or not to remap the GELF message fields to Logstash event fields or leave them intact.

Remapping converts the following GELF fields to Logstash equivalents:

- `full\_message` becomes `event["message"]`.
- If there is no `full\_message`, `short\_message` becomes `event["message"]`.

*strip\_leading\_underscore*

- Value type is [boolean](#)
- Default value is `true`

Whether or not to remove the leading `\_` in GELF fields or leave them in place. (Logstash < 1.2 did not remove them by default.). Note that GELF version 1.1 format now requires all non-standard fields to be added as an "additional" field, beginning with an underscore.

e.g. `\_foo` becomes `foo`

*tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

*type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

## gemfire

Push events to a GemFire region.

GemFire is an object database.

To use this plugin you need to add gemfire.jar to your CLASSPATH. Using format=json requires jackson.jar too; use of continuous queries requires antlr.jar.

Note: this plugin has only been tested with GemFire 7.0.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
gemfire {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">cache_name</a>	<a href="#">string</a>	No	"logstash"
<a href="#">cache_xml_file</a>	<a href="#">string</a>	No	nil
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">interest_regexp</a>	<a href="#">string</a>	No	".\*"
<a href="#">query</a>	<a href="#">string</a>	No	nil
<a href="#">region_name</a>	<a href="#">string</a>	No	"Logstash"
<a href="#">serialization</a>	<a href="#">string</a>	No	nil
<a href="#">tags</a>	<a href="#">array</a>	No	
<a href="#">threads</a>	<a href="#">number</a>	No	1
<a href="#">type</a>	<a href="#">string</a>	No	

## Details

### `add_field`

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

### `cache_name`

- Value type is [string](#)
- Default value is "logstash"

Your client cache name

### `cache_xml_file`

- Value type is [string](#)
- Default value is `nil`

The path to a GemFire client cache XML file.

Example:

```
<client-cache>
 <pool name="client-pool" subscription-enabled="true" subscription-
redundancy="1">
 <locator host="localhost" port="31331"/>
 </pool>
 <region name="Logstash">
 <region-attributes refid="CACHING_PROXY" pool-name="client-pool" >
 </region-attributes>
 </region>
 </client-cache>
<codec>
```

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

### `interest_regexp`

- Value type is [string](#)
- Default value is "./\*"

A regexp to use when registering interest for cache events. Ignored if a :query is specified.

*query*

- Value type is [string](#)
- Default value is `nil`

A query to run as a GemFire "continuous query"; if specified it takes precedence over `:interest_regexp` which will be ignore.

Important: use of continuous queries requires subscriptions to be enabled on the client pool.

*region\_name*

- Value type is [string](#)
- Default value is "Logstash"

The region name

*serialization*

- Value type is [string](#)
- Default value is `nil`

How the message is serialized in the cache. Can be one of "json" or "plain"; default is plain

*tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

*threads*

- Value type is [number](#)
- Default value is 1

*type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

## generator

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

Generate random log events.

The general intention of this is to test performance of plugins.

An event is generated first

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
generator {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">count</a>	<a href="#">number</a>	No	0
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">lines</a>	<a href="#">array</a>	No	
<a href="#">message</a>	<a href="#">string</a>	No	"Hello world!"
<a href="#">tags</a>	<a href="#">array</a>	No	
<a href="#">threads</a>	<a href="#">number</a>	No	1
<a href="#">type</a>	<a href="#">string</a>	No	

## Details

### `add_field`

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

### `codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

### `count`

- Value type is [number](#)
- Default value is 0

Set how many messages should be generated.

The default, 0, means generate an unlimited number of events.

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

### *lines*

- Value type is [array](#)
- There is no default value for this setting.

The lines to emit, in order. This option cannot be used with the *message* setting.

Example:

```
input {
 generator {
 lines => [
 "line 1",
 "line 2",
 "line 3"
]
 # Emit all lines 3 times.
 count => 3
 }
}
```

The above will emit line 1 then line 2 then line, then line 1, etc...

### *message*

- Value type is [string](#)
- Default value is "Hello world!"

The message string to use in the event.

If you set this to `stdin` then this plugin will read a single line from `stdin` and use that as the message string for every event.

Otherwise, this value will be used verbatim as the event message.

### *tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

### *threads*

- Value type is [number](#)
- Default value is 1

*type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

## github

- Version: 3.0.1
- Released on: 2016-07-14
- [Changelog](#)

This plugin does not ship with Logstash by default, but it is easy to install by running  
`bin/logstash-plugin install logstash-input-github`.

Read events from github webhooks

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
github {
 port => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
codec	<a href="#">codec</a>	No	"plain"
drop_invalid	<a href="#">boolean</a>	No	false
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	
ip	<a href="#">string</a>	No	"0.0.0.0"
port	<a href="#">number</a>	Yes	
secret_token	<a href="#">string</a>	No	
tags	<a href="#">array</a>	No	
type	<a href="#">string</a>	No	

## Details

#### `add_field`

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

#### `codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

#### `drop_invalid`

- Value type is [boolean](#)
- Default value is `false`

If Secret is defined, we drop the events that don't match. Otherwise, we'll just add an invalid tag

#### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

#### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique ID to the plugin instance, this ID is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

`ip`

- Value type is [string](#)

- Default value is "0.0.0.0"

The ip to listen on

port

- This is a required setting.
- Value type is [number](#)
- There is no default value for this setting.

The port to listen on

secret_token

- Value type is [string](#)
- There is no default value for this setting.

Your GitHub Secret Token for the webhook

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

graphite

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

Receive graphite metrics. This plugin understands the text-based graphite carbon protocol. Both `N` and `specific-timestamp` forms are supported, example:

```
mysql.slow_query.count 204 N
haproxy.live_backends 7 1364608909
```

`N` means `now` for a timestamp. This plugin also supports having the time specified in the metric payload:

For every metric received from a client, a single event will be emitted with the metric name as the field (like `mysql.slow_query.count`) and the metric value as the field's value.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
graphite {
    port => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>add_field</code>	hash	No	<code>{ }</code>
<code>codec</code>	codec	No	<code>"plain"</code>
<code>enable_metric</code>	boolean	No	<code>true</code>
<code>host</code>	string	No	<code>"0.0.0.0"</code>
<code>id</code>	string	No	
<code>mode</code>	string , one of <code>["server", "client"]</code>	No	<code>"server"</code>
<code>port</code>	number	Yes	
<code>proxy_protocol</code>	boolean	No	<code>false</code>
<code>ssl_cert</code>	a valid filesystem path	No	
<code>ssl_enable</code>	boolean	No	<code>false</code>
<code>ssl_extra_chain_certs</code>	array	No	<code>[]</code>

Setting	Input type	Required	Default value
ssl_key	a valid filesystem path	No	
ssl_key_passphrase	password	No	nil
ssl_verify	boolean	No	true
tags	array	No	
type	string	No	

Details

[`add_field`](#)

- Value type is [hash](#)
- Default value is {}

Add a field to an event

[`codec`](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

[`data_timeout \(DEPRECATED\)`](#)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [number](#)
- Default value is -1

[`enable_metric`](#)

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

[`host`](#)

- Value type is [string](#)

- Default value is "0.0.0.0"

When mode is `server`, the address to listen on. When mode is `client`, the address to connect to.

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
  stdout {  
    id => "my_plugin_id"  
  }  
}
```

`mode`

- Value can be any of: `server`, `client`
- Default value is "server"

Mode to operate in. `server` listens for client connections, `client` connects to a server.

`port`

- This is a required setting.
- Value type is [number](#)
- There is no default value for this setting.

When mode is `server`, the port to listen on. When mode is `client`, the port to connect to.

`proxy_protocol`

- Value type is [boolean](#)
- Default value is `false`

Proxy protocol support, only v1 is supported at this time

<http://www.haproxy.org/download/1.5/doc/proxy-protocol.txt>

`ssl_cacert (DEPRECATED)`

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [path](#)
- There is no default value for this setting.

The SSL CA certificate, chainfile or CA path. The system CA path is automatically included.

`ssl_cert`

- Value type is [path](#)
- There is no default value for this setting.

SSL certificate path

`ssl_enable`

- Value type is [boolean](#)
- Default value is `false`

Enable SSL (must be set for other `ssl_` options to take effect).

`ssl_extra_chain_certs`

- Value type is [array](#)
- Default value is `[]`

An Array of extra X509 certificates to be added to the certificate chain. Useful when the CA chain is not necessary in the system store.

`ssl_key`

- Value type is [path](#)
- There is no default value for this setting.

SSL key path

`ssl_key_passphrase`

- Value type is [password](#)
- Default value is `nil`

SSL key passphrase

`ssl_verify`

- Value type is [boolean](#)
- Default value is `true`

Verify the identity of the other end of the SSL connection against the CA. For input, sets the field `sslsubject` to that of the client certificate.

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

heartbeat

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

Generate heartbeat messages.

The general intention of this is to test the performance and availability of Logstash.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
heartbeat {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{}
codec	codec	No	"plain"
count	number	No	-1
enable_metric	boolean	No	true
id	string	No	
interval	number	No	60
message	string	No	"ok"
tags	array	No	
threads	number	No	1
type	string	No	

Details

[add_field](#)

- Value type is [hash](#)

- Default value is { }

Add a field to an event

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

`count`

- Value type is [number](#)
- Default value is -1

How many times to iterate. This is typically used only for testing purposes.

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
  stdout {
    id => "my_plugin_id"
  }
}
```

`interval`

- Value type is [number](#)
- Default value is 60

Set how frequently messages should be sent.

The default, `60`, means send a message every 60 seconds.

message

- Value type is [string](#)
- Default value is "`ok`"

The message string to use in the event.

If you set this to `epoch` then this plugin will use the current timestamp in unix timestamp (which is by definition, UTC). It will output this value into a field called `clock`

If you set this to `sequence` then this plugin will send a sequence of numbers beginning at 0 and incrementing each interval. It will output this value into a field called `clock`

Otherwise, this value will be used verbatim as the event message. It will output this value into a field called `message`

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

threads

- Value type is [number](#)
- Default value is `1`

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

heroku

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-input-heroku`.

Stream events from a heroku app's logs.

This will read events in a manner similar to how the `heroku logs -t` command fetches logs.

Recommended filters:

```
filter {
  grok {
    pattern => "^%{TIMESTAMP_ISO8601:timestamp}
%{WORD:component}\[%{WORD:process}(:\.%{INT:instance:int})?\]: 
%{DATA:message}$"
  }
  date { timestamp => ISO8601 }
}
```

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
heroku {
  app => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
app	string	Yes	
codec	codec	No	"plain"
tags	array	No	
type	string	No	

Details

`add_field`

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

`app`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The name of your heroku application. This is usually the first part of the domain name my-app-name.herokuapp.com

`charset (DEPRECATED)`

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value can be any of: ASCII-8BIT, Big5, Big5-HKSCS, Big5-UAO, CP949, Emacs-Mule, EUC-JP, EUC-KR, EUC-TW, GB18030, GBK, ISO-8859-1, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9, ISO-8859-10, ISO-8859-11, ISO-8859-13, ISO-8859-14, ISO-8859-15, ISO-8859-16, KOI8-R, KOI8-U, Shift_JIS, US-ASCII, UTF-8, UTF-16BE, UTF-16LE, UTF-32BE, UTF-32LE, Windows-1251, GB2312, IBM437, IBM737, IBM775, CP850, IBM852, CP852, IBM855, CP855, IBM857, IBM860, IBM861, IBM862, IBM863, IBM864, IBM865, IBM866, IBM869, Windows-1258, GB1988, macCentEuro, macCroatian, macCyrillic, macGreek, macIceland, macRoman, macRomania, macThai, macTurkish, macUkraine, CP950, CP951, stateless-ISO-2022-JP, eucJP-ms, CP51932, GB12345, ISO-2022-JP, ISO-2022-JP-2, CP50220, CP50221, Windows-1252, Windows-1250, Windows-1256, Windows-1253, Windows-1255, Windows-1254, TIS-620, Windows-874, Windows-1257, Windows-31J, MacJapanese, UTF-7, UTF8-MAC, UTF-16, UTF-32, UTF8-DoCoMo, SJIS-DoCoMo, UTF8-KDDI, SJIS-KDDI, ISO-2022-JP-KDDI, stateless-ISO-2022-JP-KDDI, UTF8-SoftBank, SJIS-SoftBank, BINARY, CP437, CP737, CP775, IBM850, CP857, CP860, CP861, CP862, CP863, CP864, CP865, CP866, CP869, CP1258, Big5-HKSCS:2008, eucJP, euc-jp-ms, euckR, eucTW, EUC-CN, eucCN, CP936, ISO2022-JP, ISO2022-JP2, ISO8859-1, CP1252, ISO8859-2, CP1250, ISO8859-3, ISO8859-4, ISO8859-5, ISO8859-6, CP1256, ISO8859-7, CP1253, ISO8859-8, CP1255, ISO8859-9, CP1254, ISO8859-10, ISO8859-11, CP874, ISO8859-13, CP1257, ISO8859-14, ISO8859-15, ISO8859-16, CP878, CP932, csWindows31J, SJIS, PCK, MacJapan, ASCII, ANSI_X3.4-1968, 646, CP65000, CP65001, UTF-8-MAC, UTF-8-HFS, UCS-2BE, UCS-4BE, UCS-4LE, CP1251, external, locale
- There is no default value for this setting.

The character encoding used in this input. Examples include `UTF-8` and `cp1252`

This setting is useful if your log files are in Latin-1 (aka cp1252) or in another character set other than UTF-8.

This only affects plain format logs since json is UTF-8 already.

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

debug (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [boolean](#)
- Default value is false

format (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value can be any of: plain, json, json_event, msgpack_event
- There is no default value for this setting.

The format of input data (plain, json, json_event)

message_format (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [string](#)
- There is no default value for this setting.

If format is json, an event sprintf string to build what the display @message should be given (defaults to the raw JSON). sprintf format strings look like %{fieldname}

If format is json_event, ALL fields except for @type are expected to be present. Not receiving all fields will cause unexpected results.

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

http

- Version: 3.0.3
- Released on: 2016-07-14
- [Changelog](#)

We keep the redefined method in a new http server class, this is because in other parts of logstash we might be using puma as webserver, for example in the sinatra part we need this method to actually return the REQUEST_PATH, so it can actually infer the right resource to use. Fixes <https://github.com/logstash-plugins/logstash-input-http/issues/51>

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
http {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
additional_codecs	hash	No	{"application/json"=>"json"}
codec	codec	No	"plain"
enable_metric	boolean	No	true
host	string	No	"0.0.0.0"
id	string	No	
keystore	a valid filesystem path	No	
keystore_password	password	No	
password	password	No	
port	number	No	8080
response_headers	hash	No	{"Content-Type"=>"text/plain"}
ssl	boolean	No	false
tags	array	No	

Setting	Input type	Required	Default value
threads	number	No	4
type	string	No	
user	string	No	
verify mode	string , one of ["none", "peer", "force_peer"]	No	"none"

Details

[add_field](#)

- Value type is [hash](#)
- Default value is { }

Add a field to an event

[additional_codecs](#)

- Value type is [hash](#)
- Default value is {"application/json"=>"json"}

Apply specific codecs for specific content types. The default codec will be applied only after this list is checked and no codec for the request's content-type is found

[codec](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

[enable_metric](#)

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

[host](#)

- Value type is [string](#)

- Default value is "0.0.0.0"

Codec used to decode the incoming data. This codec will be used as a fall-back if the content-type is not found in the "additional_codecs" hash. The host or ip to bind

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
  stdout {  
    id => "my_plugin_id"  
  }  
}  
keystore
```

- Value type is [path](#)
- There is no default value for this setting.

The JKS keystore to validate the client's certificates

keystore_password

- Value type is [password](#)
- There is no default value for this setting.

Set the truststore password

password

- Value type is [password](#)
- There is no default value for this setting.

Password for basic authorization

port

- Value type is [number](#)
- Default value is 8080

The TCP port to bind to

response_headers

- Value type is [hash](#)
- Default value is { "Content-Type"=>"text/plain"}

specify a custom set of response headers

ssl

- Value type is [boolean](#)
- Default value is `false`

SSL Configurations

Enable SSL

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

threads

- Value type is [number](#)
- Default value is 4

Maximum number of threads to use

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

user

- Value type is [string](#)
- There is no default value for this setting.

Username for basic authorization

verify_mode

- Value can be any of: `none`, `peer`, `force_peer`
- Default value is "`none`"

Set the client certificate verification method. Valid methods: `none`, `peer`, `force_peer`

http_poller

- Version: 3.1.1
- Released on: 2017-01-10
- [Changelog](#)

This Logstash input plugin allows you to call an HTTP API, decode the output of it into event(s), and send them on their merry way. The idea behind this plugin came from a need to read springboot metrics endpoint, instead of configuring jmx to monitor my java application memory/gc/ etc.

Example

Reads from a list of urls and decodes the body of the response with a codec. The config should look like this:

```
input {
  http_poller {
    urls => {
      test1 => "http://localhost:9200"
      test2 => {
        # Supports all options supported by ruby's Manticore HTTP client
        method => get
        url => "http://localhost:9200/_cluster/health"
        headers => {
          Accept => "application/json"
        }
        auth => {
          user => "AzureDiamond"
          password => "hunter2"
        }
      }
    }
    request_timeout => 60
    # Supports "cron", "every", "at" and "in" schedules by rufus scheduler
    schedule => { cron => "* * * * * UTC" }
    codec => "json"
    # A hash of request metadata info (timing, response headers, etc.) will
    be sent here
    metadata_target => "http_poller_metadata"
  }
}

output {
  stdout {
    codec => rubydebug
  }
}
```

Using the HTTP poller with custom a custom CA or self signed cert.

If you have a self signed cert you will need to convert your server's certificate to a valid# .jks or .p12 file. An easy way to do it is to run the following one-liner, substituting your server's URL for the placeholder MYURL and MYPORt.

```
openssl s_client -showcerts -connect MYURL:MYPORt </dev/null
2>/dev/null|openssl x509 -outform PEM > downloaded_cert.pem; keytool -import
-alias test -file downloaded_cert.pem -keystore downloaded_truststore.jks
```

The above snippet will create two files `downloaded_cert.pem` and `downloaded_truststore.jks`. You will be prompted to set a password for the `jks` file during this process. To configure logstash use a config like the one that follows.

```
http_poller {
    urls => {
        myurl => "https://myhostname:1234"
    }
    truststore => "/path/to/downloaded_truststore.jks"
    truststore_password => "mypassword"
    interval => 30
}
```

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
http_poller {
    urls => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
automatic_retries	number	No	1
cacert	a valid filesystem path	No	
client_cert	a valid filesystem path	No	
client_key	a valid filesystem path	No	
codec	codec	No	"plain"
connect_timeout	number	No	10
cookies	boolean	No	true

Setting	Input type	Required	Default value
enable_metric	boolean	No	true
follow_redirects	boolean	No	true
id	string	No	
keepalive	boolean	No	true
keystore	a valid filesystem path	No	
keystore_password	password	No	
keystore_type	string	No	"JKS"
metadata_target	string	No	"@metadata"
pool_max	number	No	50
pool_max_per_route	number	No	25
proxy	<<,>>	No	
request_timeout	number	No	60
retry_non_idempotent	boolean	No	false
schedule	hash	No	
socket_timeout	number	No	10
ssl_certificate_validation	boolean	No	true
tags	array	No	
target	string	No	
truststore	a valid filesystem path	No	
truststore_password	password	No	
truststore_type	string	No	"JKS"
type	string	No	
urls	hash	Yes	
validate_after_inactivity	number	No	200

Details

[add_field](#)

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

`automatic_retries`

- Value type is [number](#)
- Default value is 1

How many times should the client retry a failing URL. We highly recommend NOT setting this value to zero if keepalive is enabled. Some servers incorrectly end keepalives early requiring a retry! Note: if `retry_non_idempotent` is set only GET, HEAD, PUT, DELETE, OPTIONS, and TRACE requests will be retried.

`cacert`

- Value type is [path](#)
- There is no default value for this setting.

If you need to use a custom X.509 CA (.pem certs) specify the path to that here

`client_cert`

- Value type is [path](#)
- There is no default value for this setting.

If you'd like to use a client certificate (note, most people don't want this) set the path to the x509 cert here

`client_key`

- Value type is [path](#)
- There is no default value for this setting.

If you're using a client certificate specify the path to the encryption key here

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

`connect_timeout`

- Value type is [number](#)
- Default value is 10

Timeout (in seconds) to wait for a connection to be established. Default is 10s

cookies

- Value type is [boolean](#)
- Default value is `true`

Enable cookie support. With this enabled the client will persist cookies across requests as a normal web browser would. Enabled by default

enable_metric

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

follow_redirects

- Value type is [boolean](#)
- Default value is `true`

Should redirects be followed? Defaults to `true`

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
  stdout {  
    id => "my_plugin_id"  
  }  
}
```

interval (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [number](#)
- There is no default value for this setting.

How often (in seconds) the urls will be called DEPRECATED. Use `schedule` option instead. If both interval and schedule options are specified, interval option takes higher precedence

keepalive

- Value type is [boolean](#)
- Default value is `true`

Turn this on to enable HTTP keepalive support. We highly recommend setting `automatic_retries` to at least one with this to fix interactions with broken keepalive implementations.

keystore

- Value type is [path](#)
- There is no default value for this setting.

If you need to use a custom keystore (`.jks`) specify that here. This does not work with `.pem` keys!

keystore_password

- Value type is [password](#)
- There is no default value for this setting.

Specify the keystore password here. Note, most `.jks` files created with keytool require a password!

keystore_type

- Value type is [string](#)
- Default value is "JKS"

Specify the keystore type here. One of `JKS` or `PKCS12`. Default is `JKS`

metadata_target

- Value type is [string](#)
- Default value is "@metadata"

If you'd like to work with the request/response metadata. Set this value to the name of the field you'd like to store a nested hash of metadata.

pool_max

- Value type is [number](#)
- Default value is 50

Max number of concurrent connections. Defaults to 50

pool_max_per_route

- Value type is [number](#)
- Default value is 25

Max number of concurrent connections to a single host. Defaults to 25

proxy

- Value type is [string](#)
- There is no default value for this setting.

If you'd like to use an HTTP proxy . This supports multiple configuration syntaxes:

1. Proxy host in form: `http://proxy.org:1234`
2. Proxy host in form: `{host => "proxy.org", port => 80, scheme => 'http', user => 'username@host', password => 'password'}`
3. Proxy host in form: `{url => 'http://proxy.org:1234', user => 'username@host', password => 'password'}`

request_timeout

- Value type is [number](#)
- Default value is 60

Timeout (in seconds) for the entire request

retry_non_idempotent

- Value type is [boolean](#)
- Default value is `false`

If `automatic_retries` is enabled this will cause non-idempotent HTTP verbs (such as POST) to be retried.

schedule

- Value type is [hash](#)
- There is no default value for this setting.

Schedule of when to periodically poll from the urls Format: A hash with + key: "cron" | "every" | "in" | "at" + value: string Examples: a) { "every" => "1h" } b) { "cron" => "* * * * * UTC" } See: rufus/scheduler for details about different schedule options and value string format

socket_timeout

- Value type is [number](#)
- Default value is 10

Timeout (in seconds) to wait for data on the socket. Default is 10s

ssl_certificate_validation

- Value type is [boolean](#)
- Default value is `true`

Set this to false to disable SSL/TLS certificate validation Note: setting this to false is generally considered insecure!

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

target

- Value type is [string](#)
- There is no default value for this setting.

Define the target field for placing the received data. If this setting is omitted, the data will be stored at the root (top level) of the event.

truststore

- Value type is [path](#)
- There is no default value for this setting.

If you need to use a custom truststore (.jks) specify that here. This does not work with .pem certs!

truststore_password

- Value type is [password](#)
- There is no default value for this setting.

Specify the truststore password here. Note, most .jks files created with keytool require a password!

truststore_type

- Value type is [string](#)
- Default value is "JKS"

Specify the truststore type here. One of `JKS` or `PKCS12`. Default is `JKS`

`type`

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

`urls`

- This is a required setting.
- Value type is [hash](#)
- There is no default value for this setting.

A Hash of urls in this format : `"name" => "url"`. The name and the url will be passed in the outputted event

`validate_after_inactivity`

- Value type is [number](#)
- Default value is 200

How long to wait before checking if the connection is stale before executing a request on a connection using `keepalive`. # You may want to set this lower, possibly to 0 if you get connection errors regularly Quoting the Apache commons docs (this client is based Apache Commons):
Defines period of inactivity in milliseconds after which persistent connections must be re-validated prior to being leased to the consumer. Non-positive value passed to this method disables connection validation. This check helps detect connections that have become stale (half-closed) while kept inactive in the pool. See [these docs for more info](#)

imap

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

Read mails from IMAP server

Periodically scan an IMAP folder (`INBOX` by default) and move any read messages to the trash.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
imap {
    host => ...
    password => ...
    user => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
check_interval	number	No	300
codec	codec	No	"plain"
content_type	string	No	"text/plain"
delete	boolean	No	false
enable_metric	boolean	No	true
expunge	boolean	No	false
fetch_count	number	No	50
folder	string	No	"INBOX"
host	string	Yes	
id	string	No	
lowercase_headers	boolean	No	true
password	password	Yes	
port	number	No	

Setting	Input type	Required	Default value
secure	boolean	No	true
strip_attachments	boolean	No	false
tags	array	No	
type	string	No	
user	string	Yes	
verify_cert	boolean	No	true

Details

add_field

- Value type is [hash](#)
- Default value is {}

Add a field to an event

check_interval

- Value type is [number](#)
- Default value is 300

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

content_type

- Value type is [string](#)
- Default value is "text/plain"

For multipart messages, use the first part that has this content-type as the event message.

delete

- Value type is [boolean](#)
- Default value is false

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`expunge`

- Value type is [boolean](#)
- Default value is `false`

`fetch_count`

- Value type is [number](#)
- Default value is `50`

`folder`

- Value type is [string](#)
- Default value is `"INBOX"`

`host`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
  stdout {
    id => "my_plugin_id"
  }
}
```

`lowercase_headers`

- Value type is [boolean](#)
- Default value is `true`

password

- This is a required setting.
- Value type is [password](#)
- There is no default value for this setting.

port

- Value type is [number](#)
- There is no default value for this setting.

secure

- Value type is [boolean](#)
- Default value is `true`

strip_attachments

- Value type is [boolean](#)
- Default value is `false`

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

user

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

verify_cert

- Value type is [boolean](#)
- Default value is `true`

irc

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

Read events from an IRC Server.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
irc {
    channels => ...
    host => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
catch_all	boolean	No	false
channels	array	Yes	
codec	codec	No	"plain"
enable_metric	boolean	No	true
get_stats	boolean	No	false
host	string	Yes	
id	string	No	
nick	string	No	"logstash"
password	password	No	
port	number	No	6667
real	string	No	"logstash"
secure	boolean	No	false
stats_interval	number	No	5
tags	array	No	
type	string	No	

Setting	Input type	Required	Default value
user	string	No	"logstash"

Details

[*add_field*](#)

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

[*catch_all*](#)

- Value type is [boolean](#)
- Default value is `false`

Catch all IRC channel/user events not just channel messages

[*channels*](#)

- This is a required setting.
- Value type is [array](#)
- There is no default value for this setting.

Channels to join and read messages from.

These should be full channel names including the # symbol, such as "#logstash".

For passworded channels, add a space and the channel password, such as "#logstash password".

[*codec*](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

[*enable_metric*](#)

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

get_stats

- Value type is [boolean](#)
- Default value is `false`

Gather and send user counts for channels - this requires `catch_all` and will force it

host

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Host of the IRC Server to connect to.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
  stdout {  
    id => "my_plugin_id"  
  }  
}
```

nick

- Value type is [string](#)
- Default value is `"logstash"`

IRC Nickname

password

- Value type is [password](#)
- There is no default value for this setting.

IRC Server password

port

- Value type is [number](#)
- Default value is 6667

Port for the IRC Server

real

- Value type is [string](#)
- Default value is "logstash"

IRC Real name

secure

- Value type is [boolean](#)
- Default value is false

Set this to true to enable SSL.

stats_interval

- Value type is [number](#)
- Default value is 5

How often in minutes to get the user count stats

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

`user`

- Value type is [string](#)
- Default value is "logstash"

IRC Username

jdbc

- Version: 4.1.3
- Released on: 2016-10-20
- [Changelog](#)

This plugin was created as a way to ingest data in any database with a JDBC interface into Logstash. You can periodically schedule ingestion using a cron syntax (see `schedule` setting) or run the query one time to load data into Logstash. Each row in the resultset becomes a single event. Columns in the resultset are converted into fields in the event.

Drivers

This plugin does not come packaged with JDBC driver libraries. The desired jdbc driver library must be explicitly passed in to the plugin using the `jdbc_driver_library` configuration option.

Scheduling

Input from this plugin can be scheduled to run periodically according to a specific schedule. This scheduling syntax is powered by [rufus-scheduler](#). The syntax is cron-like with some extensions specific to Rufus (e.g. timezone support).

Examples:

* 5 * 1-3 *	will execute every minute of 5am every day of January through March.
0 * * * *	will execute on the 0th minute of every hour every day.
0 6 * * * America/Chicago	will execute at 6:00am (UTC/GMT -5) every day.

Further documentation describing this syntax can be found [here](#).

State

The plugin will persist the `sql_last_value` parameter in the form of a metadata file stored in the configured `last_run_metadata_path`. Upon query execution, this file will be updated with the current value of `sql_last_value`. Next time the pipeline starts up, this value will be updated by reading from the file. If `clean_run` is set to true, this value will be ignored and `sql_last_value` will be set to Jan 1, 1970, or 0 if `use_column_value` is true, as if no query has ever been executed.

Dealing With Large Result-sets

Many JDBC drivers use the `fetch_size` parameter to limit how many results are pre-fetched at a time from the cursor into the client's cache before retrieving more results from the result-set. This is configured in this plugin using the `jdbc_fetch_size` configuration option. No fetch size is set by default in this plugin, so the specific driver's default size will be used.

Usage:

Here is an example of setting up the plugin to fetch data from a MySQL database. First, we place the appropriate JDBC driver library in our current path (this can be placed anywhere on your filesystem). In this example, we connect to the `mydb` database using the user: `mysql` and wish to input all rows in the `songs` table that match a specific artist. The following examples demonstrates a possible Logstash configuration for this. The `schedule` option in this example will instruct the plugin to execute this input statement on the minute, every minute.

```
input {
  jdbc {
    jdbc_driver_library => "mysql-connector-java-5.1.36-bin.jar"
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    jdbc_connection_string => "jdbc:mysql://localhost:3306/mydb"
    jdbc_user => "mysql"
    parameters => { "favorite_artist" => "Beethoven" }
    schedule => "* * * * *"
    statement => "SELECT * from songs where artist = :favorite_artist"
  }
}
```

Configuring SQL statement

A sql statement is required for this input. This can be passed-in via a `statement` option in the form of a string, or read from a file (`statement_filepath`). File option is typically used when the SQL statement is large or cumbersome to supply in the config. The file option only supports one SQL statement. The plugin will only accept one of the options. It cannot read a statement from a file as well as from the `statement` configuration parameter.

Predefined Parameters

Some parameters are built-in and can be used from within your queries. Here is the list:

<code>sql_last_value</code>	The value used to calculate which rows to query. Before any query is run, this is set to Thursday, 1 January 1970, or 0 if <code>use_column_value</code> is true and <code>tracking_column</code> is set. It is updated accordingly after subsequent queries are run.
-----------------------------	---

Example:

```
input {
  jdbc {
```

```

        statement => "SELECT id, mycolumn1, mycolumn2 FROM my_table WHERE id >
:sql_last_value"
    use_column_value => true
    tracking_column => id
    # ... other configuration bits
}
}

```

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```

jdbc {
    jdbc_connection_string => ...
    jdbc_driver_class => ...
    jdbc_user => ...
}

```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
charset	string	No	
clean_run	boolean	No	false
codec	codec	No	"plain"
columns_charset	hash	No	{ }
connection_retry_attempts	number	No	1
connection_retry_attempts_wait_time	number	No	0.5
enable_metric	boolean	No	true
id	string	No	
jdbc_connection_string	string	Yes	
jdbc_default_timezone	string	No	
jdbc_driver_class	string	Yes	
jdbc_driver_library	string	No	
jdbc_fetch_size	number	No	
jdbc_page_size	number	No	100000

Setting	Input type	Required	Default value
jdbc_paging_enabled	boolean	No	false
jdbc_password	password	No	
jdbc_password_filepath	a valid filesystem path	No	
jdbc_pool_timeout	number	No	5
jdbc_user	string	Yes	
jdbc_validate_connection	boolean	No	false
jdbc_validation_timeout	number	No	3600
last_run_metadata_path	string	No	"/Users/suyog/.logstash_jdbc_last_run"
lowercase_column_names	boolean	No	true
parameters	hash	No	{ }
record_last_run	boolean	No	true
schedule	string	No	
sequel_opts	hash	No	{ }
sql_log_level	string , one of ["fatal", "error", "warn", "info", "debug"]	No	"info"
statement	string	No	
statement_filepath	a valid filesystem path	No	
tags	array	No	
tracking_column	string	No	
tracking_column_type	string , one of ["numeric", "timestamp"]	No	"numeric"
type	string	No	
use_column_value	boolean	No	false

Details

`add_field`

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

`charset`

- Value type is [string](#)
- There is no default value for this setting.

The character encoding of all columns, leave empty if the columns are already properly UTF-8 encoded. Specific columns charsets using `:columns_charset` can override this setting.

`clean_run`

- Value type is [boolean](#)
- Default value is `false`

Whether the previous run state should be preserved

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

`columns_charset`

- Value type is [hash](#)
- Default value is `{ }`

The character encoding for specific columns. This option will override the `:charset` option for the specified columns.

Example:

```
input {
  jdbc {
    ...
    columns_charset => { "column0" => "ISO-8859-1" }
  }
}
```

```
    ...
}
```

this will only convert column0 that has ISO-8859-1 as an original encoding.

`connection_retry_attempts`

- Value type is [number](#)
- Default value is 1

Maximum number of times to try connecting to database

`connection_retry_attempts_wait_time`

- Value type is [number](#)
- Default value is 0.5

Number of seconds to sleep between connection attempts

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

#### `jdbc_connection_string`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

JDBC connection string

### `jdbc_default_timezone`

- Value type is [string](#)
- There is no default value for this setting.

Timezone conversion. SQL does not allow for timezone data in timestamp fields. This plugin will automatically convert your SQL timestamp fields to Logstash timestamps, in relative UTC time in ISO8601 format.

Using this setting will manually assign a specified timezone offset, instead of using the timezone setting of the local machine. You must use a canonical timezone, **America/Denver**, for example.

### `jdbc_driver_class`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

JDBC driver class to load, for example, "org.apache.derby.jdbc.ClientDriver" NB per <https://github.com/logstash-plugins/logstash-input-jdbc/issues/43> if you are using the Oracle JDBC driver (ojdbc6.jar) the correct `jdbc_driver_class` is "Java::oracle.jdbc.driver.OracleDriver"

### `jdbc_driver_library`

- Value type is [string](#)
- There is no default value for this setting.

JDBC driver library path to third party driver library. In case of multiple libraries being required you can pass them separated by a comma.

If not provided, Plugin will look for the driver class in the Logstash Java classpath.

### `jdbc_fetch_size`

- Value type is [number](#)
- There is no default value for this setting.

JDBC fetch size. if not provided, respective driver's default will be used

### `jdbc_page_size`

- Value type is [number](#)
- Default value is 100000

JDBC page size

#### *jdbc\_paging\_enabled*

- Value type is [boolean](#)
- Default value is `false`

#### JDBC enable paging

This will cause a sql statement to be broken up into multiple queries. Each query will use limits and offsets to collectively retrieve the full result-set. The limit size is set with `jdbc_page_size`.

Be aware that ordering is not guaranteed between queries.

#### *jdbc\_password*

- Value type is [password](#)
- There is no default value for this setting.

#### JDBC password

#### *jdbc\_password\_filepath*

- Value type is [path](#)
- There is no default value for this setting.

#### JDBC password filename

#### *jdbc\_pool\_timeout*

- Value type is [number](#)
- Default value is 5

Connection pool configuration. The amount of seconds to wait to acquire a connection before raising a PoolTimeoutError (default 5)

#### *jdbc\_user*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

#### JDBC user

#### *jdbc\_validate\_connection*

- Value type is [boolean](#)
- Default value is `false`

Connection pool configuration. Validate connection before use.

*jdbc\_validation\_timeout*

- Value type is [number](#)
- Default value is 3600

Connection pool configuration. How often to validate a connection (in seconds)

*last\_run\_metadata\_path*

- Value type is [string](#)
- Default value is "/Users/suyog/.logstash\_jdbc\_last\_run"

Path to file with last run time

*lowercase\_column\_names*

- Value type is [boolean](#)
- Default value is true

Whether to force the lowercasing of identifier fields

*parameters*

- Value type is [hash](#)
- Default value is {}

Hash of query parameter, for example { "target\_id" => "321" }

*record\_last\_run*

- Value type is [boolean](#)
- Default value is true

Whether to save state or not in last\_run\_metadata\_path

*schedule*

- Value type is [string](#)
- There is no default value for this setting.

Schedule of when to periodically run statement, in Cron format for example: "\* \* \* \* \*" (execute query every minute, on the minute)

There is no schedule by default. If no schedule is given, then the statement is run exactly once.

*sequel\_opts*

- Value type is [hash](#)
- Default value is { }

General/Vendor-specific Sequel configuration options.

An example of an optional connection pool configuration max\_connections - The maximum number of connections the connection pool

examples of vendor-specific options can be found in this documentation page:

[https://github.com/jeremyevans/sequel/blob/master/doc/opening\\_databases.rdoc](https://github.com/jeremyevans/sequel/blob/master/doc/opening_databases.rdoc)

*sql\_log\_level*

- Value can be any of: fatal, error, warn, info, debug
- Default value is "info"

Log level at which to log SQL queries, the accepted values are the common ones fatal, error, warn, info and debug. The default value is info.

*statement*

- Value type is [string](#)
- There is no default value for this setting.

If undefined, Logstash will complain, even if codec is unused. Statement to execute

To use parameters, use named parameter syntax. For example:

```
"SELECT * FROM MYTABLE WHERE id = :target_id"
```

here, ":target\_id" is a named parameter. You can configure named parameters with the parameters setting.

*statement\_filepath*

- Value type is [path](#)
- There is no default value for this setting.

Path of file containing statement to execute

*tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

#### `tracking_column`

- Value type is [string](#)
- There is no default value for this setting.

If tracking column value rather than timestamp, the column whose value is to be tracked

#### `tracking_column_type`

- Value can be any of: numeric, timestamp
- Default value is "numeric"

Type of tracking column. Currently only "numeric" and "timestamp"

#### `type`

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

#### `use_column_value`

- Value type is [boolean](#)
- Default value is `false`

Use an incremental column value rather than a timestamp

## jmx

- Version: 3.0.1
- Released on: 2017-01-10
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
 bin/logstash-plugin install logstash-input-jmx.

This input plugin permits to retrieve metrics from remote Java applications using JMX. Every polling\_frequency, it scans a folder containing json configuration files describing JVMs to monitor with metrics to retrieve. Then a pool of threads will retrieve metrics and create events.

## The configuration:

In Logstash configuration, you must set the polling frequency, the number of thread used to poll metrics and a directory absolute path containing json files with the configuration per jvm of metrics to retrieve. Logstash input configuration example:

```
jmx {
 //Required
 path => "/apps/logstash_conf/jmxconf"
 //Optional, default 60s
 polling_frequency => 15
 type => "jmx"
 //Optional, default 4
 nb_thread => 4
}
```

Json JMX configuration example:

```
{
 //Required, JMX listening host/ip
 "host" : "192.168.1.2",
 //Required, JMX listening port
 "port" : 1335,
 //Optional, the username to connect to JMX
 "username" : "user",
 //Optional, the password to connect to JMX
 "password": "pass",
 //Optional, use this alias as a prefix in the metric name. If not set use
<host>_<port>
 "alias" : "test.homeserver.elasticsearch",
 //Required, list of JMX metrics to retrieve
 "queries" : [
 {
 //Required, the object name of Mbean to request
 "object_name" : "java.lang:type=Memory",
 //Optional, use this alias in the metrics value instead of the
 object_name
 "object_alias" : "Memory"
```

```

}, {
 "object_name" : "java.lang:type=Runtime",
 //Optional, set of attributes to retrieve. If not set retrieve
 //all metrics available on the configured object_name.
 "attributes" : ["Uptime", "StartTime"],
 "object_alias" : "Runtime"
}, {
 //object_name can be configured with * to retrieve all matching Mbeans
 "object_name" : "java.lang:type=GarbageCollector,name=*",
 "attributes" : ["CollectionCount", "CollectionTime"],
 //object_alias can be based on specific value from the object_name thanks
 to ${<varname>}.
 //In this case ${type} will be replaced by GarbageCollector...
 "object_alias" : "${type}.${name}"
}, {
 "object_name" : "java.nio:type=BufferPool,name=*",
 "object_alias" : "${type}.${name}"
}
}

```

Here are examples of generated events. When returned metrics value type is number/boolean it is stored in `metric_value_number` event field otherwise it is stored in `metric_value_string` event field.

```

{
 "@version" => "1",
 "@timestamp" => "2014-02-18T20:57:27.688Z",
 "host" => "192.168.1.2",
 "path" => "/apps/logstash_conf/jmxconf",
 "type" => "jmx",
 "metric_path" =>
"test.homeserver.elasticsearch.GarbageCollector.ParNew.CollectionCount",
 "metric_value_number" => 2212
}
{
 "@version" => "1",
 "@timestamp" => "2014-02-18T20:58:06.376Z",
 "host" => "localhost",
 "path" => "/apps/logstash_conf/jmxconf",
 "type" => "jmx",
 "metric_path" =>
"test.homeserver.elasticsearch.BufferPool.mapped.ObjectName",
 "metric_value_string" => "java.nio:type=BufferPool,name=mapped"
}

```

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
jmx {
 path => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
codec	<a href="#">codec</a>	No	"plain"
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	
nb_thread	<a href="#">number</a>	No	4
path	<a href="#">string</a>	Yes	
polling_frequency	<a href="#">number</a>	No	60
tags	<a href="#">array</a>	No	
type	<a href="#">string</a>	No	

## Details

### [add\\_field](#)

- Value type is [hash](#)
- Default value is { }

Add a field to an event

### [codec](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

### [enable\\_metric](#)

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### *id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique ID to the plugin instance, this ID is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

nb_thread

- Value type is [number](#)
- Default value is 4

Indicate number of thread launched to retrieve metrics

path

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Path where json conf files are stored

polling_frequency

- Value type is [number](#)
- Default value is 60

Indicate interval between two jmx metrics retrieval (in s)

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

kafka

- Version: 5.1.6
- Released on: 2017-01-25
- [Changelog](#)

This input will read events from a Kafka topic. It uses the 0.10 version of the consumer API provided by Kafka to read messages from the broker.

Here's a compatibility matrix that shows the Kafka client versions that are compatible with each combination of Logstash and the Kafka input plugin:

Kafka Client Version	Logstash Version	Plugin Version	Why?
0.8	2.0.0 - 2.x.x	<3.0.0	Legacy, 0.8 is still popular
0.9	2.0.0 - 2.3.x	3.x.x	Works with the old Ruby Event API (<code>event['product']['price'] = 10</code>)
0.9	2.4.x - 5.x.x	4.x.x	Works with the new getter/setter APIs (<code>event.set('product')[price]', 10)</code>)
0.10.0.x	2.4.x - 5.x.x	5.x.x	Not compatible with the ≤ 0.9 broker

We recommend that you use matching Kafka client and broker versions. During upgrades, you should upgrade brokers before clients because brokers target backwards compatibility. For example, the 0.9 broker is compatible with both the 0.8 consumer and 0.9 consumer APIs, but not the other way around.

This input supports connecting to Kafka over:

- SSL (requires plugin version 3.0.0 or later)
- Kerberos SASL (requires plugin version 5.1.0 or later)

By default security is disabled but can be turned on as needed.

The Logstash Kafka consumer handles group management and uses the default offset management strategy using Kafka topics.

Logstash instances by default form a single logical group to subscribe to Kafka topics. Each Logstash Kafka consumer can run multiple threads to increase read throughput. Alternatively, you could run multiple Logstash instances with the same `group_id` to spread the load across physical machines. Messages in a topic will be distributed to all Logstash instances with the same `group_id`.

Ideally you should have as many threads as the number of partitions for a perfect balance — more threads than partitions means that some threads will be idle.

For more information see <http://kafka.apache.org/documentation.html#theconsumer>

Kafka consumer configuration: <http://kafka.apache.org/documentation.html#consumerconfigs>

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
kafka {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>add_field</code>	hash	No	{ }
<code>auto_commit_interval_ms</code>	string	No	"5000"
<code>auto_offset_reset</code>	string	No	
<code>bootstrap_servers</code>	string	No	"localhost:9092"
<code>check_crcs</code>	string	No	
<code>client_id</code>	string	No	"logstash"
<code>codec</code>	codec	No	"plain"
<code>connections_max_idle_ms</code>	string	No	
<code>consumer_threads</code>	number	No	1
<code>decorate_events</code>	boolean	No	false
<code>enable_auto_commit</code>	string	No	"true"
<code>enable_metric</code>	boolean	No	true

Setting	Input type	Required	Default value
exclude_internal_topics	string	No	
fetch_max_wait_ms	string	No	
fetch_min_bytes	string	No	
group_id	string	No	"logstash"
heartbeat_interval_ms	string	No	
id	string	No	
jaas_path	a valid filesystem path	No	
kerberos_config	a valid filesystem path	No	
key_deserializer_class	string	No	"org.apache.kafka.common.serialization.StringDeserializer"
max_partition_fetch_bytes	string	No	
max_poll_records	string	No	
metadata_max_age_ms	string	No	
partition_assignment_strategy	string	No	
poll_timeout_ms	number	No	100
receive_buffer_bytes	string	No	
reconnect_backoff_ms	string	No	
request_timeout_ms	string	No	
retry_backoff_ms	string	No	
sasl_kerberos_service_name	string	No	
sasl_mechanism	string	No	"GSSAPI"
security_protocol	string , one of ["PLAINTEXT", "SSL", "SASL_PLAINTEXT", "SASL_SSL"]	No	"PLAINTEXT"
send_buffer_bytes	string	No	
session_timeout_ms	string	No	

Setting	Input type	Required	Default value
ssl_key_password	password	No	
ssl_keystore_location	a valid filesystem path	No	
ssl_keystore_password	password	No	
ssl_keystore_type	string	No	
ssl_truststore_location	a valid filesystem path	No	
ssl_truststore_password	password	No	
ssl_truststore_type	string	No	
tags	array	No	
topics	array	No	["logstash"]
topics_pattern	string	No	
type	string	No	
value_deserializer_class	string	No	"org.apache.kafka.common.serialization.StringDeserializer"

Details

[add_field](#)

- Value type is [hash](#)
- Default value is {}

Add a field to an event

[auto_commit_interval_ms](#)

- Value type is [string](#)
- Default value is "5000"

The frequency in milliseconds that the consumer offsets are committed to Kafka.

[auto_offset_reset](#)

- Value type is [string](#)

- There is no default value for this setting.

What to do when there is no initial offset in Kafka or if an offset is out of range:

- earliest: automatically reset the offset to the earliest offset
- latest: automatically reset the offset to the latest offset
- none: throw exception to the consumer if no previous offset is found for the consumer's group
- anything else: throw exception to the consumer.

bootstrap_servers

- Value type is [string](#)
- Default value is "localhost:9092"

A list of URLs to use for establishing the initial connection to the cluster. This list should be in the form of host1:port1,host2:port2 These urls are just used for the initial connection to discover the full cluster membership (which may change dynamically) so this list need not contain the full set of servers (you may want more than one, though, in case a server is down).

check_crcs

- Value type is [string](#)
- There is no default value for this setting.

Automatically check the CRC32 of the records consumed. This ensures no on-the-wire or on-disk corruption to the messages occurred. This check adds some overhead, so it may be disabled in cases seeking extreme performance.

client_id

- Value type is [string](#)
- Default value is "logstash"

The id string to pass to the server when making requests. The purpose of this is to be able to track the source of requests beyond just ip/port by allowing a logical application name to be included.

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

`connections_max_idle_ms`

- Value type is [string](#)
- There is no default value for this setting.

Close idle connections after the number of milliseconds specified by this config.

`consumer_threads`

- Value type is [number](#)
- Default value is 1

Ideally you should have as many threads as the number of partitions for a perfect balance — more threads than partitions means that some threads will be idle

`decorate_events`

- Value type is [boolean](#)
- Default value is `false`

Option to add Kafka metadata like topic, message size to the event. This will add a field named `kafka` to the logstash event containing the following attributes: `topic`: The topic this message is associated with `consumer_group`: The consumer group used to read in this event `partition`: The partition this message is associated with `offset`: The offset from the partition this message is associated with `key`: A ByteBuffer containing the message key

`enable_auto_commit`

- Value type is [string](#)
- Default value is `"true"`

If true, periodically commit to Kafka the offsets of messages already returned by the consumer. This committed offset will be used when the process fails as the position from which the consumption will begin.

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`exclude_internal_topics`

- Value type is [string](#)
- There is no default value for this setting.

Whether records from internal topics (such as offsets) should be exposed to the consumer. If set to true the only way to receive records from an internal topic is subscribing to it.

`fetch_max_wait_ms`

- Value type is [string](#)
- There is no default value for this setting.

The maximum amount of time the server will block before answering the fetch request if there isn't sufficient data to immediately satisfy `fetch_min_bytes`. This should be less than or equal to the timeout used in `poll_timeout_ms`

`fetch_min_bytes`

- Value type is [string](#)
- There is no default value for this setting.

The minimum amount of data the server should return for a fetch request. If insufficient data is available the request will wait for that much data to accumulate before answering the request.

`group_id`

- Value type is [string](#)
- Default value is "logstash"

The identifier of the group this consumer belongs to. Consumer group is a single logical subscriber that happens to be made up of multiple processors. Messages in a topic will be distributed to all Logstash instances with the same `group_id`

`heartbeat_interval_ms`

- Value type is [string](#)
- There is no default value for this setting.

The expected time between heartbeats to the consumer coordinator. Heartbeats are used to ensure that the consumer's session stays active and to facilitate rebalancing when new consumers join or leave the group. The value must be set lower than `session.timeout.ms`, but typically should be set no higher than 1/3 of that value. It can be adjusted even lower to control the expected time for normal rebalances.

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when

you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
  stdout {
    id => "my_plugin_id"
  }
}
jaas_path
```

- Value type is [path](#)
- There is no default value for this setting.

The Java Authentication and Authorization Service (JAAS) API supplies user authentication and authorization services for Kafka. This setting provides the path to the JAAS file. Sample JAAS file for Kafka client:

```
KafkaClient {
  com.sun.security.auth.module.Krb5LoginModule required
  useTicketCache=true
  renewTicket=true
  serviceName="kafka";
};
```

Please note that specifying `jaas_path` and `kerberos_config` in the config file will add these to the global JVM system properties. This means if you have multiple Kafka inputs, all of them would be sharing the same `jaas_path` and `kerberos_config`. If this is not desirable, you would have to run separate instances of Logstash on different JVM instances.

[`kerberos_config`](#)

- Value type is [path](#)
- There is no default value for this setting.

Optional path to kerberos config file. This is krb5.conf style as detailed in
https://web.mit.edu/kerberos/krb5-1.12/doc/admin/conf_files/krb5_conf.html

[`key_deserializer_class`](#)

- Value type is [string](#)
- Default value is `"org.apache.kafka.common.serialization.StringDeserializer"`

Java Class used to deserialize the record's key

[`max_partition_fetch_bytes`](#)

- Value type is [string](#)
- There is no default value for this setting.

The maximum amount of data per-partition the server will return. The maximum total memory used for a request will be <code>#partitions * max.partition.fetch.bytes</code>. This size must be at least as large as the maximum message size the server allows or else it is possible for the producer to send messages larger than the consumer can fetch. If that happens, the consumer can get stuck trying to fetch a large message on a certain partition.

max_poll_records

- Value type is [string](#)
- There is no default value for this setting.

The maximum number of records returned in a single call to poll().

metadata_max_age_ms

- Value type is [string](#)
- There is no default value for this setting.

The period of time in milliseconds after which we force a refresh of metadata even if we haven't seen any partition leadership changes to proactively discover any new brokers or partitions

partition_assignment_strategy

- Value type is [string](#)
- There is no default value for this setting.

The class name of the partition assignment strategy that the client will use to distribute partition ownership amongst consumer instances

poll_timeout_ms

- Value type is [number](#)
- Default value is 100

Time kafka consumer will wait to receive new messages from topics

receive_buffer_bytes

- Value type is [string](#)
- There is no default value for this setting.

The size of the TCP receive buffer (SO_RCVBUF) to use when reading data.

reconnect_backoff_ms

- Value type is [string](#)
- There is no default value for this setting.

The amount of time to wait before attempting to reconnect to a given host. This avoids repeatedly connecting to a host in a tight loop. This backoff applies to all requests sent by the consumer to the broker.

`request_timeout_ms`

- Value type is [string](#)
- There is no default value for this setting.

The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted.

`retry_backoff_ms`

- Value type is [string](#)
- There is no default value for this setting.

The amount of time to wait before attempting to retry a failed fetch request to a given topic partition. This avoids repeated fetching-and-failing in a tight loop.

`sasl_kerberos_service_name`

- Value type is [string](#)
- There is no default value for this setting.

The Kerberos principal name that Kafka broker runs as. This can be defined either in Kafka's JAAS config or in Kafka's config.

`sasl_mechanism`

- Value type is [string](#)
- Default value is "GSSAPI"

[SASL mechanism](#) used for client connections. This may be any mechanism for which a security provider is available. GSSAPI is the default mechanism.

`security_protocol`

- Value can be any of: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL
- Default value is "PLAINTEXT"

Security protocol to use, which can be either of
PLAINTEXT,SSL,SASL_PLAINTEXT,SASL_SSL

send_buffer_bytes

- Value type is [string](#)
- There is no default value for this setting.

The size of the TCP send buffer (SO_SNDBUF) to use when sending data

session_timeout_ms

- Value type is [string](#)
- There is no default value for this setting.

The timeout after which, if the `poll_timeout_ms` is not invoked, the consumer is marked dead and a rebalance operation is triggered for the group identified by `group_id`

ssl (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [boolean](#)
- Default value is `false`

Enable SSL/TLS secured communication to Kafka broker.

ssl_key_password

- Value type is [password](#)
- There is no default value for this setting.

The password of the private key in the key store file.

ssl_keystore_location

- Value type is [path](#)
- There is no default value for this setting.

If client authentication is required, this setting stores the keystore path.

ssl_keystore_password

- Value type is [password](#)
- There is no default value for this setting.

If client authentication is required, this setting stores the keystore password

ssl_keystore_type

- Value type is [string](#)
- There is no default value for this setting.

The keystore type.

ssl_truststore_location

- Value type is [path](#)
- There is no default value for this setting.

The JKS truststore path to validate the Kafka broker's certificate.

ssl_truststore_password

- Value type is [password](#)
- There is no default value for this setting.

The truststore password

ssl_truststore_type

- Value type is [string](#)
- There is no default value for this setting.

The truststore type.

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

topics

- Value type is [array](#)
- Default value is `["logstash"]`

A list of topics to subscribe to, defaults to `["logstash"]`.

topics_pattern

- Value type is [string](#)

- There is no default value for this setting.

A topic regex pattern to subscribe to. The topics configuration will be ignored when using this configuration.

`type`

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

`value_deserializer_class`

- Value type is [string](#)
- Default value is "`org.apache.kafka.common.serialization.StringDeserializer`"

Java Class used to deserialize the record's value

kinesis

- Version: 2.0.3
- Released on: 2016-12-15
- [Changelog](#)

This plugin does not ship with Logstash by default, but it is easy to install by running
`bin/logstash-plugin install logstash-input-kinesis.`

Receive events through an AWS Kinesis stream.

This input plugin uses the Java Kinesis Client Library underneath, so the documentation at
<https://github.com/awslabs/amazon-kinesis-client> will be useful.

AWS credentials can be specified either through environment variables, or an IAM instance role. The library uses a DynamoDB table for worker coordination, so you'll need to grant access to that as well as to the Kinesis stream. The DynamoDB table has the same name as the `application_name` configuration option, which defaults to "logstash".

The library can optionally also send worker statistics to CloudWatch.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
kinesis {
    kinesis_stream_name => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>add_field</code>	hash	No	<code>{ }</code>
<code>application_name</code>	string	No	<code>"logstash"</code>
<code>checkpoint_interval_seconds</code>	number	No	60
<code>codec</code>	codec	No	<code>"plain"</code>
<code>enable_metric</code>	boolean	No	<code>true</code>

Setting	Input type	Required	Default value
id	string	No	
kinesis_stream_name	string	Yes	
metrics	string , one of [nil, "cloudwatch"]	No	nil
region	string	No	"us-east-1"
tags	array	No	
type	string	No	

Details

[*add_field*](#)

- Value type is [hash](#)
- Default value is {}

Add a field to an event

[*application_name*](#)

- Value type is [string](#)
- Default value is "logstash"

The application name used for the dynamodb coordination table. Must be unique for this kinesis stream.

[*checkpoint_interval_seconds*](#)

- Value type is [number](#)
- Default value is 60

How many seconds between worker checkpoints to dynamodb.

[*codec*](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

enable_metric

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance, by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

### *kinesis\_stream\_name*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The kinesis stream name.

### *metrics*

- Value can be any of: `cloudwatch`
- Default value is `nil`

Worker metric tracking. By default this is disabled, set it to "cloudwatch" to enable the cloudwatch integration in the Kinesis Client Library.

### *region*

- Value type is [string](#)
- Default value is `"us-east-1"`

The AWS region for Kinesis, DynamoDB, and CloudWatch (if enabled)

### *tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

*type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

## log4j

- Version: 3.0.3
- Released on: 2016-08-11
- [Changelog](#)

Read events over a TCP socket from a Log4j SocketAppender. This plugin works only with log4j version 1.x.

Can either accept connections from clients or connect to a server, depending on `mode`. Depending on which `mode` is configured, you need a matching SocketAppender or a SocketHubAppender on the remote side.

One event is created per received log4j LoggingEvent with the following schema:

- `timestamp` ⇒ the number of milliseconds elapsed from 1/1/1970 until logging event was created.
- `path` ⇒ the name of the logger
- `priority` ⇒ the level of this event
- `logger_name` ⇒ the name of the logger
- `thread` ⇒ the thread name making the logging request
- `class` ⇒ the fully qualified class name of the caller making the logging request.
- `file` ⇒ the source file name and line number of the caller making the logging request in a colon-separated format "fileName:lineNumber".
- `method` ⇒ the method name of the caller making the logging request.
- `NDC` ⇒ the NDC string
- `stack_trace` ⇒ the multi-line stack-trace

Also if the original log4j LoggingEvent contains MDC hash entries, they will be merged in the event as fields.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
log4j {
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
codec	<a href="#">codec</a>	No	"plain"
enable_metric	<a href="#">boolean</a>	No	true
host	<a href="#">string</a>	No	"0.0.0.0"
id	<a href="#">string</a>	No	
mode	<a href="#">string</a> , one of ["server", "client"]	No	"server"
port	<a href="#">number</a>	No	4560
tags	<a href="#">array</a>	No	
type	<a href="#">string</a>	No	

## Details

### [`add\_field`](#)

- Value type is [hash](#)
- Default value is { }

Add a field to an event

### [`codec`](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

### [`enable\_metric`](#)

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### [`host`](#)

- Value type is [string](#)
- Default value is "0.0.0.0"

When mode is `server`, the address to listen on. When mode is `client`, the address to connect to.

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

### `mode`

- Value can be any of: `server`, `client`
- Default value is `"server"`

Mode to operate in. `server` listens for client connections, `client` connects to a server.

### `port`

- Value type is [number](#)
- Default value is `4560`

When mode is `server`, the port to listen on. When mode is `client`, the port to connect to.

### `tags`

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

### `type`

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

## lumberjack

- Version: 3.1.1
- Released on: 2016-07-14
- [Changelog](#)

Receive events using the lumberjack protocol.

This is mainly to receive events shipped with lumberjack[<http://github.com/jordansissel/lumberjack>], now represented primarily via the [Logstash-forwarder](#).

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
lumberjack {
 port => ...
 ssl_certificate => ...
 ssl_key => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">congestion_threshold</a>	<a href="#">number</a>	No	5
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">host</a>	<a href="#">string</a>	No	"0.0.0.0"
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">port</a>	<a href="#">number</a>	Yes	
<a href="#">ssl_certificate</a>	a valid filesystem path	Yes	
<a href="#">ssl_key</a>	a valid filesystem path	Yes	
<a href="#">ssl_key_passphrase</a>	<a href="#">password</a>	No	
<a href="#">tags</a>	<a href="#">array</a>	No	
<a href="#">type</a>	<a href="#">string</a>	No	

## Details

### `add_field`

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

### `codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

### `congestion_threshold`

- Value type is [number](#)
- Default value is 5

The number of seconds before we raise a timeout, this option is useful to control how much time to wait if something is blocking the pipeline.

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `host`

- Value type is [string](#)
- Default value is `"0.0.0.0"`

The IP address to listen on.

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
port
```

- This is a required setting.
- Value type is [number](#)
- There is no default value for this setting.

The port to listen on.

[\*ssl\\_certificate\*](#)

- This is a required setting.
- Value type is [path](#)
- There is no default value for this setting.

SSL certificate to use.

[\*ssl\\_key\*](#)

- This is a required setting.
- Value type is [path](#)
- There is no default value for this setting.

SSL key to use.

[\*ssl\\_key\\_passphrase\*](#)

- Value type is [password](#)
- There is no default value for this setting.

SSL key passphrase to use.

[\*tags\*](#)

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

*type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

## meetup

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-input-meetup`.

Run command line tools and capture the whole output as an event.

Notes:

- The `@source` of this event will be the command run.
- The `@message` of this event will be the entire stdout of the command as one event.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
meetup {
 interval => ...
 meetupkey => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
codec	<a href="#">codec</a>	No	"plain"
eventstatus	<a href="#">string</a>	No	"upcoming,past"
groupid	<a href="#">string</a>	No	
interval	<a href="#">number</a>	Yes	
meetupkey	<a href="#">string</a>	Yes	
tags	<a href="#">array</a>	No	
type	<a href="#">string</a>	No	
urlname	<a href="#">string</a>	No	
venueid	<a href="#">string</a>	No	

## Details

*add\_field*

- Value type is [hash](#)
- Default value is {}

Add a field to an event

*codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

*eventstatus*

- Value type is [string](#)
- Default value is "upcoming,past"

Event Status'

*groupid*

- Value type is [string](#)
- There is no default value for this setting.

The Group ID, multiple may be specified separated by commas Must have one of `urlname`, `venueid`, `groupid`

*interval*

- This is a required setting.
- Value type is [number](#)
- There is no default value for this setting.

Interval to run the command. Value is in seconds.

*meetupkey*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Meetup Key

*tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

*type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

*urlname*

- Value type is [string](#)
- There is no default value for this setting.

URLName - the URL name ie `ElasticSearch-Oklahoma-City` Must have one of `urlname`, `venue_id`, `group_id`

*venueid*

- Value type is [string](#)
- There is no default value for this setting.

The venue ID Must have one of `urlname`, `venue_id`, `group_id`

## pipe

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

Stream events from a long running command pipe.

By default, each event is assumed to be one line. If you want to join lines, you'll want to use the multiline filter.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
pipe {
 command => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">command</a>	<a href="#">string</a>	Yes	
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">tags</a>	<a href="#">array</a>	No	
<a href="#">type</a>	<a href="#">string</a>	No	

## Details

### [add\\_field](#)

- Value type is [hash](#)
- Default value is { }

## Add a field to an event

### `codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

### `command`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

**TODO(sissel):** This should switch to use the `line` codec by default once we switch away from doing `readline` Command to run and read events from, one line at a time.

Example:

```
command => "echo hello world"
enable_metric
```

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}

tags
```

- Value type is [array](#)

- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

#### `type`

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

## puppet\_facter

- Version: 3.0.0
- Released on: 2016-09-09
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
`bin/logstash-plugin install logstash-input-puppet_facter.`

Connects to a puppet server and requests facts

### Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
puppet_facter {
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
codec	<a href="#">codec</a>	No	"plain"
enable_metric	<a href="#">boolean</a>	No	true
environment	<a href="#">string</a>	No	"production"
host	<a href="#">string</a>	No	"0.0.0.0"
id	<a href="#">string</a>	No	
interval	<a href="#">number</a>	No	600
port	<a href="#">number</a>	No	8140
private_key	a valid filesystem path	No	
public_key	a valid filesystem path	No	
ssl	<a href="#">boolean</a>	No	true
tags	<a href="#">array</a>	No	
type	<a href="#">string</a>	No	

### Details

*add\_field*

- Value type is [hash](#)
- Default value is {}

Add a field to an event

*codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

*enable\_metric*

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

*environment*

- Value type is [string](#)
- Default value is "production"

*host*

- Value type is [string](#)
- Default value is "0.0.0.0"

*id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique ID to the plugin instance, this ID is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

interval

- Value type is [number](#)
- Default value is 600

port

- Value type is [number](#)
- Default value is 8140

private_key

- Value type is [path](#)
- There is no default value for this setting.

public_key

- Value type is [path](#)
- There is no default value for this setting.

ssl

- Value type is [boolean](#)
- Default value is `true`

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

rabbitmq

- Version: 5.2.2
- Released on: 2017-01-17
- [Changelog](#)

Pull events from a [RabbitMQ](#) queue.

The default settings will create an entirely transient queue and listen for all messages by default. If you need durability or any other advanced settings, please set the appropriate options

This plugin uses the [March Hare](#) library for interacting with the RabbitMQ server. Most configuration options map directly to standard RabbitMQ and AMQP concepts. The [AMQP 0-9-1 reference guide](#) and other parts of the RabbitMQ documentation are useful for deeper understanding.

The properties of messages received will be stored in the `[@metadata][rabbitmq_properties]` field if the `@metadata_enabled` setting is checked. Note that storing metadata may degrade performance. The following properties may be available (in most cases dependent on whether they were set by the sender):

- app-id
- cluster-id
- consumer-tag
- content-encoding
- content-type
- correlation-id
- delivery-mode
- exchange
- expiration
- message-id
- priority
- redeliver
- reply-to
- routing-key
- timestamp
- type
- user-id

For example, to get the RabbitMQ message's timestamp property into the Logstash event's `@timestamp` field, use the date filter to parse the `[@metadata][rabbitmq_properties][timestamp]` field:

```
filter {
  if [@metadata][rabbitmq_properties][timestamp] {
    date {
      match => ["[@metadata][rabbitmq_properties][timestamp]", "UNIX"]
```

```

        }
    }
}
```

Additionally, any message headers will be saved in the `[@metadata][rabbitmq_headers]` field.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
rabbitmq {
    host => ...
    subscription_retry_interval_seconds => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
ack	boolean	No	true
add_field	hash	No	{ }
arguments	array	No	{ }
auto_delete	boolean	No	false
automatic_recovery	boolean	No	true
codec	codec	No	"plain"
connect_retry_interval	number	No	1
connection_timeout	number	No	
durable	boolean	No	false
enable_metric	boolean	No	true
exchange	string	No	
exchange_type	string	No	
exclusive	boolean	No	false
heartbeat	number	No	
host	string	Yes	
id	string	No	
key	string	No	"logstash"

Setting	Input type	Required	Default value
metadata_enabled	boolean	No	false
passive	boolean	No	false
password	password	No	"guest"
port	number	No	5672
prefetch_count	number	No	256
queue	string	No	""
ssl	boolean	No	
ssl_certificate_password	string	No	
ssl_certificate_path	a valid filesystem path	No	
ssl_version	string	No	"TLSv1.2"
subscription_retry_interval_seconds	number	Yes	5
tags	array	No	
threads	number	No	1
type	string	No	
user	string	No	"guest"
vhost	string	No	"/"

Details

[ack](#)

- Value type is [boolean](#)
- Default value is `true`

Enable message acknowledgements. With acknowledgements messages fetched by Logstash but not yet sent into the Logstash pipeline will be requeued by the server if Logstash shuts down. Acknowledgements will however hurt the message throughput.

This will only send an ack back every `prefetch_count` messages. Working in batches provides a performance boost here.

[add_field](#)

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

arguments

- Value type is [array](#)
- Default value is {}

Extra queue arguments as an array. To make a RabbitMQ queue mirrored, use: { "x-ha-policy" => "all"}

auto_delete

- Value type is [boolean](#)
- Default value is false

Should the queue be deleted on the broker when the last consumer disconnects? Set this option to false if you want the queue to remain on the broker, queueing up messages until a consumer comes along to consume them.

automatic_recovery

- Value type is [boolean](#)
- Default value is true

Set this to automatically recover from a broken connection. You almost certainly don't want to override this!!!

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

connect_retry_interval

- Value type is [number](#)
- Default value is 1

Time in seconds to wait before retrying a connection

connection_timeout

- Value type is [number](#)
- There is no default value for this setting.

The default connection timeout in milliseconds. If not specified the timeout is infinite.

debug (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [boolean](#)
- Default value is `false`

durable

- Value type is [boolean](#)
- Default value is `false`

Is this queue durable? (aka; Should it survive a broker restart?)

enable_metric

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

exchange

- Value type is [string](#)
- There is no default value for this setting.

The name of the exchange to bind the queue to. Specify `exchange_type` as well to declare the exchange if it does not exist

exchange_type

- Value type is [string](#)
- There is no default value for this setting.

The type of the exchange to bind to. Specifying this will cause this plugin to declare the exchange if it does not exist.

exclusive

- Value type is [boolean](#)
- Default value is `false`

Is the queue exclusive? Exclusive queues can only be used by the connection that declared them and will be deleted when it is closed (e.g. due to a Logstash restart).

heartbeat

- Value type is [number](#)
- There is no default value for this setting.

Heartbeat delay in seconds. If unspecified no heartbeats will be sent

host

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

RabbitMQ server address(es) host can either be a single host, or a list of hosts i.e. host \Rightarrow "localhost" or host \Rightarrow ["host01", "host02"]

if multiple hosts are provided on the initial connection and any subsequent recovery attempts of the hosts is chosen at random and connected to. Note that only one host connection is active at a time.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
    stdout {  
        id => "my_plugin_id"  
    }  
}
```

key

- Value type is [string](#)
- Default value is "logstash"

The routing key to use when binding a queue to the exchange. This is only relevant for direct or topic exchanges.

- Routing keys are ignored on fanout exchanges.
- Wildcards are not valid on direct exchanges.

metadata_enabled

- Value type is [boolean](#)
- Default value is `false`

Enable the storage of message headers and properties in `@metadata`. This may impact performance

passive

- Value type is [boolean](#)
- Default value is `false`

If true the queue will be passively declared, meaning it must already exist on the server. To have Logstash create the queue if necessary leave this option as false. If actively declaring a queue that already exists, the queue options for this plugin (durable etc) must match those of the existing queue.

password

- Value type is [password](#)
- Default value is "guest"

RabbitMQ password

port

- Value type is [number](#)
- Default value is `5672`

RabbitMQ port to connect on

prefetch_count

- Value type is [number](#)
- Default value is `256`

Prefetch count. If acknowledgements are enabled with the `ack` option, specifies the number of outstanding unacknowledged messages allowed.

queue

- Value type is [string](#)
- Default value is `""`

The properties to extract from each message and store in a `@metadata` field.

Technically the exchange, redeliver, and routing-key properties belong to the envelope and not the message but we ignore that distinction here. However, we extract the headers separately via `get_headers` even though the header table technically is a message property.

Freezing all strings so that code modifying the event's `@metadata` field can't touch them.

If updating this list, remember to update the documentation above too. The default codec for this plugin is JSON. You can override this to suit your particular needs however. The name of the queue Logstash will consume events from. If left empty, a transient queue with a randomly chosen name will be created.

`ssl`

- Value type is [boolean](#)
- There is no default value for this setting.

Enable or disable SSL. Note that by default remote certificate verification is off. Specify `ssl_certificate_path` and `ssl_certificate_password` if you need certificate verification

`ssl_certificate_password`

- Value type is [string](#)
- There is no default value for this setting.

Password for the encrypted PKCS12 (.p12) certificate file specified in `ssl_certificate_path`

`ssl_certificate_path`

- Value type is [path](#)
- There is no default value for this setting.

Path to an SSL certificate in PKCS12 (.p12) format used for verifying the remote host

`ssl_version`

- Value type is [string](#)
- Default value is "TLSv1.2"

Version of the SSL protocol to use.

`subscription_retry_interval_seconds`

- This is a required setting.
- Value type is [number](#)
- Default value is 5

Amount of time in seconds to wait after a failed subscription request before retrying. Subscribes can fail if the server goes away and then comes back.

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

threads

- Value type is [number](#)
- Default value is 1

tls_certificate_password (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [string](#)
- There is no default value for this setting.

TLS certificate password

tls_certificate_path (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [path](#)
- There is no default value for this setting.

TLS certificate path

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

user

- Value type is [string](#)
- Default value is "guest"

RabbitMQ username

vhost

- Value type is [string](#)
- Default value is "/"

The vhost (virtual host) to use. If you don't know what this is, leave the default. With the exception of the default vhost (""/"), names of vhosts should not begin with a forward slash.

rackspace

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-input-rackspace`.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
rackspace {
    api_key => ...
    username => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
api_key	string	Yes	
claim	number	No	1
codec	codec	No	"plain"
queue	string	No	"logstash"
region	string	No	"dfw"
tags	array	No	
ttl	number	No	60
type	string	No	
username	string	Yes	

Details

`add_field`

- Value type is [hash](#)
- Default value is { }

Add a field to an event

api_key

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Rackspace Cloud API Key

charset (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value can be any of: ASCII-8BIT, Big5, Big5-HKSCS, Big5-UAO, CP949, Emacs-Mule, EUC-JP, EUC-KR, EUC-TW, GB18030, GBK, ISO-8859-1, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9, ISO-8859-10, ISO-8859-11, ISO-8859-13, ISO-8859-14, ISO-8859-15, ISO-8859-16, KOI8-R, KOI8-U, Shift_JIS, US-ASCII, UTF-8, UTF-16BE, UTF-16LE, UTF-32BE, UTF-32LE, Windows-1251, GB2312, IBM437, IBM737, IBM775, CP850, IBM852, CP852, IBM855, CP855, IBM857, IBM860, IBM861, IBM862, IBM863, IBM864, IBM865, IBM866, IBM869, Windows-1258, GB1988, macCentEuro, macCroatian, macCyrillic, macGreek, macIceland, macRoman, macRomania, macThai, macTurkish, macUkraine, CP950, CP951, stateless-ISO-2022-JP, eucJP-ms, CP51932, GB12345, ISO-2022-JP, ISO-2022-JP-2, CP50220, CP50221, Windows-1252, Windows-1250, Windows-1256, Windows-1253, Windows-1255, Windows-1254, TIS-620, Windows-874, Windows-1257, Windows-31J, MacJapanese, UTF-7, UTF8-MAC, UTF-16, UTF-32, UTF8-DoCoMo, SJIS-DoCoMo, UTF8-KDDI, SJIS-KDDI, ISO-2022-JP-KDDI, stateless-ISO-2022-JP-KDDI, UTF8-SoftBank, SJIS-SoftBank, BINARY, CP437, CP737, CP775, IBM850, CP857, CP860, CP861, CP862, CP863, CP864, CP865, CP866, CP869, CP1258, Big5-HKSCS:2008, eucJP, euc-jp-ms, eucKR, eucTW, EUC-CN, eucCN, CP936, ISO2022-JP, ISO2022-JP2, ISO8859-1, CP1252, ISO8859-2, CP1250, ISO8859-3, ISO8859-4, ISO8859-5, ISO8859-6, CP1256, ISO8859-7, CP1253, ISO8859-8, CP1255, ISO8859-9, CP1254, ISO8859-10, ISO8859-11, CP874, ISO8859-13, CP1257, ISO8859-14, ISO8859-15, ISO8859-16, CP878, CP932, csWindows31J, SJIS, PCK, MacJapan, ASCII, ANSI_X3.4-1968, 646, CP65000, CP65001, UTF-8-MAC, UTF-8-HFS, UCS-2BE, UCS-4BE, UCS-4LE, CP1251, external, locale
- There is no default value for this setting.

The character encoding used in this input. Examples include `UTF-8` and `cp1252`

This setting is useful if your log files are in Latin-1 (aka `cp1252`) or in another character set other than `UTF-8`.

This only affects `plain` format logs since `json` is `UTF-8` already.

claim

- Value type is [number](#)
- Default value is 1

number of messages to claim Min: 1, Max: 10

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

debug (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [boolean](#)
- Default value is `false`

format (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value can be any of: plain, json, json_event, msgpack_event
- There is no default value for this setting.

The format of input data (plain, json, json_event)

message_format (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [string](#)
- There is no default value for this setting.

If format is json, an event `sprintf` string to build what the display `@message` should be given (defaults to the raw JSON). `sprintf` format strings look like `%{fieldname}`

If format is json_event, ALL fields except for `@type` are expected to be present. Not receiving all fields will cause unexpected results.

queue

- Value type is [string](#)
- Default value is "logstash"

Rackspace Queue Name

region

- Value type is [string](#)
- Default value is "dfw"

Rackspace region ord, dfw, lon, syd, etc

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

ttl

- Value type is [number](#)
- Default value is 60

length of time to hold claim Min: 60

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

username

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Rackspace Cloud Username

redis

- Version: 3.1.2
- Released on: 2017-02-01
- [Changelog](#)

This input will read events from a Redis instance; it supports both Redis channels and lists. The list command (BLPOP) used by Logstash is supported in Redis v1.3.1+, and the channel commands used by Logstash are found in Redis v1.3.8+. While you may be able to make these Redis versions work, the best performance and stability will be found in more recent stable versions. Versions 2.6.0+ are recommended.

For more information about Redis, see <http://redis.io/>

`batch_count` note: If you use the `batch_count` setting, you **must** use a Redis version 2.6.0 or newer. Anything older does not support the operations used by batching.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
redis {
    data_type => ...
    key => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>add_field</code>	hash	No	{ }
<code>batch_count</code>	number	No	125
<code>codec</code>	codec	No	"plain"
<code>data_type</code>	string , one of ["list", "channel", "pattern_channel"]	Yes	
<code>db</code>	number	No	0
<code>enable_metric</code>	boolean	No	true
<code>host</code>	string	No	"127.0.0.1"
<code>id</code>	string	No	
<code>key</code>	string	Yes	

Setting	Input type	Required	Default value
password	password	No	
port	number	No	6379
tags	array	No	
threads	number	No	1
timeout	number	No	5
type	string	No	

Details

add_field

- Value type is [hash](#)
- Default value is {}

Add a field to an event

batch_count

- Value type is [number](#)
- Default value is 125

The number of events to return from Redis using EVAL.

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

data_type

- This is a required setting.
- Value can be any of: list, channel, pattern_channel
- There is no default value for this setting.

Specify either list or channel. If redis_type is list, then we will BLPOP the key. If redis_type is channel, then we will SUBSCRIBE to the key. If redis_type is pattern_channel, then we will PSUBSCRIBE to the key.

db

- Value type is [number](#)
- Default value is 0

The Redis database number.

enable_metric

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

host

- Value type is [string](#)
- Default value is "127.0.0.1"

The hostname of your Redis server.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
  stdout {
    id => "my_plugin_id"
  }
}
```

key

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The name of a Redis list or channel.

password

- Value type is [password](#)

- There is no default value for this setting.

Password to authenticate with. There is no authentication by default.

port

- Value type is [number](#)
- Default value is 6379

The port to connect on.

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

threads

- Value type is [number](#)
- Default value is 1

timeout

- Value type is [number](#)
- Default value is 5

Initial connection timeout in seconds.

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

relp

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-input-relp`.

Read RELP events over a TCP socket.

For more information about RELP, see <http://www.rsyslog.com/doc/imrelp.html>

This protocol implements application-level acknowledgements to help protect against message loss.

Message acks only function as far as messages being put into the queue for filters; anything lost after that point will not be retransmitted

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
relp {
    port => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
codec	codec	No	"plain"
host	string	No	"0.0.0.0"
port	number	Yes	
ssl_cacert	a valid filesystem path	No	
ssl_cert	a valid filesystem path	No	
ssl_enable	boolean	No	false
ssl_key	a valid filesystem path	No	
ssl_key_passphrase	password	No	nil
ssl_verify	boolean	No	true
tags	array	No	

Setting	Input type	Required	Default value
type	string	No	

Details

add_field

- Value type is [hash](#)
- Default value is {}

Add a field to an event

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

host

- Value type is [string](#)
- Default value is "0.0.0.0"

The address to listen on.

port

- This is a required setting.
- Value type is [number](#)
- There is no default value for this setting.

The port to listen on.

ssl_cacert

- Value type is [path](#)
- There is no default value for this setting.

The SSL CA certificate, chainfile or CA path. The system CA path is automatically included.

ssl_cert

- Value type is [path](#)
- There is no default value for this setting.

SSL certificate path

ssl_enable

- Value type is [boolean](#)
- Default value is `false`

Enable SSL (must be set for other `ssl_` options to take effect).

ssl_key

- Value type is [path](#)
- There is no default value for this setting.

SSL key path

ssl_key_passphrase

- Value type is [password](#)
- Default value is `nil`

SSL key passphrase

ssl_verify

- Value type is [boolean](#)
- Default value is `true`

Verify the identity of the other end of the SSL connection against the CA. For input, sets the field `sslsubject` to that of the client certificate.

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

rss

- Version: 3.0.1
- Released on: 2016-07-14
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
bin/logstash-plugin install logstash-input-rss.

Run command line tools and capture the whole output as an event.

Notes:

- The @source of this event will be the command run.
- The @message of this event will be the entire stdout of the command as one event.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
rss {
    interval => ...
    url => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
codec	codec	No	"plain"
enable_metric	boolean	No	true
id	string	No	
interval	number	Yes	
tags	array	No	
type	string	No	
url	string	Yes	

Details

`add_field`

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique ID to the plugin instance, this ID is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

#### `interval`

- This is a required setting.
- Value type is [number](#)
- There is no default value for this setting.

Interval to run the command. Value is in seconds.

### *tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

### *type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

### *url*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

RSS/Atom feed URL

## s3

- Version: 3.1.2
- Released on: 2017-01-24
- [Changelog](#)

Stream events from files from a S3 bucket.

Each line from each file generates an event. Files ending in `.gz` are handled as gzip'ed files.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
s3 {
 bucket => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
access_key_id	<a href="#">string</a>	No	
add_field	<a href="#">hash</a>	No	{ }
aws_credentials_file	<a href="#">string</a>	No	
backup_add_prefix	<a href="#">string</a>	No	nil
backup_to_bucket	<a href="#">string</a>	No	nil
backup_to_dir	<a href="#">string</a>	No	nil
bucket	<a href="#">string</a>	Yes	
codec	<a href="#">codec</a>	No	"plain"
delete	<a href="#">boolean</a>	No	false
enable_metric	<a href="#">boolean</a>	No	true
exclude_pattern	<a href="#">string</a>	No	nil
id	<a href="#">string</a>	No	
interval	<a href="#">number</a>	No	60

Setting	Input type	Required	Default value
prefix	<a href="#">string</a>	No	nil
proxy_uri	<a href="#">string</a>	No	
region	<a href="#">string</a> , one of [ "us-east-1", "us-west-1", "us-west-2", "eu-central-1", "eu-west-1", "ap-southeast-1", "ap-southeast-2", "ap-northeast-1", "ap-northeast-2", "sa-east-1", "us-gov-west-1", "cn-north-1", "ap-south-1"]	No	"us-east-1"
secret_access_key	<a href="#">string</a>	No	
session_token	<a href="#">string</a>	No	
sincedb_path	<a href="#">string</a>	No	nil
tags	<a href="#">array</a>	No	

Setting	Input type	Required	Default value
temporary_directory	<a href="#">string</a>	No	"/var/folders/_9/x4bq65rs6vd0rrjthct3zxjw0000gn/T/logstash"
type	<a href="#">string</a>	No	

## Details

### [access\\_key\\_id](#)

- Value type is [string](#)
- There is no default value for this setting.

This plugin uses the AWS SDK and supports several ways to get credentials, which will be tried in this order:

1. Static configuration, using `access_key_id` and `secret_access_key` params in logstash plugin config
2. External credentials file specified by `aws_credentials_file`
3. Environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`
4. Environment variables `AMAZON_ACCESS_KEY_ID` and `AMAZON_SECRET_ACCESS_KEY`
5. IAM Instance Profile (available when running inside EC2)

### [add\\_field](#)

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

### [aws\\_credentials\\_file](#)

- Value type is [string](#)
- There is no default value for this setting.

Path to YAML file containing a hash of AWS credentials. This file will only be loaded if `access_key_id` and `secret_access_key` aren't set. The contents of the file should look like this:

```
:access_key_id: "12345"
:secret_access_key: "54321"
```

### [backup\\_add\\_prefix](#)

- Value type is [string](#)

- Default value is `nil`

Append a prefix to the key (full path including file name in s3) after processing. If backing up to another (or the same) bucket, this effectively lets you choose a new *folder* to place the files in

`backup_to_bucket`

- Value type is [string](#)
- Default value is `nil`

Name of a S3 bucket to backup processed files to.

`backup_to_dir`

- Value type is [string](#)
- Default value is `nil`

Path of a local directory to backup processed files to.

`bucket`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The name of the S3 bucket.

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

`delete`

- Value type is [boolean](#)
- Default value is `false`

Whether to delete processed files from the original bucket.

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

*exclude\_pattern*

- Value type is [string](#)
- Default value is `nil`

Ruby style regexp of keys to exclude from the bucket

*id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

*interval*

- Value type is [number](#)
- Default value is 60

Interval to wait between to check the file list again after a run is finished. Value is in seconds.

*prefix*

- Value type is [string](#)
- Default value is `nil`

If specified, the prefix of filenames in the bucket must match (not a regexp)

*proxy\_uri*

- Value type is [string](#)
- There is no default value for this setting.

URI to proxy server if required

### *region*

- Value can be any of: us-east-1, us-west-1, us-west-2, eu-central-1, eu-west-1, ap-southeast-1, ap-southeast-2, ap-northeast-1, ap-northeast-2, sa-east-1, us-gov-west-1, cn-north-1, ap-south-1
- Default value is "us-east-1"

## The AWS Region

### *secret\_access\_key*

- Value type is [string](#)
- There is no default value for this setting.

## The AWS Secret Access Key

### *session\_token*

- Value type is [string](#)
- There is no default value for this setting.

## The AWS Session token for temporary credential

### *sincedb\_path*

- Value type is [string](#)
- Default value is nil

Where to write the since database (keeps track of the date the last handled file was added to S3). The default will write sincedb files to some path matching "\$HOME/.sincedb\*" Should be a path with filename not just a directory.

### *tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

### *temporary\_directory*

- Value type is [string](#)
- Default value is "/var/folders/\_9/x4bq65rs6vd0rrjthct3zxjw0000gn/T/logstash"

Set the directory where logstash will store the tmp files before processing them. default to the current OS temporary directory in linux /tmp/logstash

*type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

## salesforce

- Version: 3.0.0
- Released on: 2017-01-24
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
`bin/logstash-plugin install logstash-input-salesforce.`

This Logstash input plugin allows you to query Salesforce using SOQL and puts the results into Logstash, one row per event. You can configure it to pull entire sObjects or only specific fields.

NOTE: This input plugin will stop after all the results of the query are processed and will need to be re-run to fetch new results. It does not utilize the streaming API.

In order to use this plugin, you will need to create a new SFDC Application using oauth. More details can be found here:

[https://help.salesforce.com/apex/HTViewHelpDoc?id=connected\\_app\\_create.htm](https://help.salesforce.com/apex/HTViewHelpDoc?id=connected_app_create.htm)

You will also need a username, password, and security token for your salesforce instance. More details for generating a token can be found here:

[https://help.salesforce.com/apex/HTViewHelpDoc?id=user\\_security\\_token.htm](https://help.salesforce.com/apex/HTViewHelpDoc?id=user_security_token.htm)

In addition to specifying an sObject, you can also supply a list of API fields that will be used in the SOQL query.

## Example

This example prints all the Salesforce Opportunities to standard out

```
input {
 salesforce {
 client_id => 'OAUTH CLIENT ID FROM YOUR SFDC APP'
 client_secret => 'OAUTH CLIENT SECRET FROM YOUR SFDC APP'
 username => 'email@example.com'
 password => 'super-secret'
 security_token => 'SECURITY TOKEN FOR THIS USER'
 sfdc_object_name => 'Opportunity'
 }
}

output {
 stdout {
 codec => rubydebug
 }
}
```

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
salesforce {
 client_id => ...
 client_secret => ...
 password => ...
 security_token => ...
 sfdc_object_name => ...
 username => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
api_version	<a href="#">string</a>	No	
client_id	<a href="#">string</a>	Yes	
client_secret	<a href="#">string</a>	Yes	
codec	<a href="#">codec</a>	No	"plain"
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	
password	<a href="#">string</a>	Yes	
security_token	<a href="#">string</a>	Yes	
sfdc_fields	<a href="#">array</a>	No	[]
sfdc_filters	<a href="#">string</a>	No	""
sfdc_object_name	<a href="#">string</a>	Yes	
tags	<a href="#">array</a>	No	
to_underscores	<a href="#">boolean</a>	No	false
type	<a href="#">string</a>	No	
use_test_sandbox	<a href="#">boolean</a>	No	false
username	<a href="#">string</a>	Yes	

## Details

*add\_field*

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

*api\_version*

- Value type is [string](#)
- There is no default value for this setting.

By default, this uses the default Restforce API version. To override this, set this to something like "32.0" for example

*client\_id*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Consumer Key for authentication. You must set up a new SFDC connected app with oath to use this output. More information can be found here:

[https://help.salesforce.com/apex/HTViewHelpDoc?id=connected\\_app\\_create.htm](https://help.salesforce.com/apex/HTViewHelpDoc?id=connected_app_create.htm)

*client\_secret*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Consumer Secret from your oauth enabled connected app

*codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

*enable\_metric*

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### *id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

password

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The password used to login to sfdc

security_token

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The security token for this account. For more information about generating a security token, see:
https://help.salesforce.com/apex/HTViewHelpDoc?id=user_security_token.htm

sfdc_fields

- Value type is [array](#)
- Default value is `[]`

These are the field names to return in the Salesforce query If this is empty, all fields are returned.

sfdc_filters

- Value type is [string](#)
- Default value is `""`

These options will be added to the WHERE clause in the SOQL statement. Additional fields can be filtered on by adding `field1 = value1 AND field2 = value2 AND...`

`sfdc_object_name`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The name of the salesforce object you are creating or updating

`tags`

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

`to_underscores`

- Value type is [boolean](#)
- Default value is `false`

Setting this to true will convert SFDC's NamedFieldsc to *named_fieldsc*

`type`

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

`use_test_sandbox`

- Value type is [boolean](#)
- Default value is `false`

Set this to true to connect to a sandbox sfdc instance logging in through test.salesforce.com

username

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

A valid salesforce user name, usually your email address. Used for authentication and will be the user all objects are created or modified by

snmptrap

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

Read snmp trap messages as events

Resulting @message looks like :

```
#<SNMP::SNMPv1_Trap:0x6f1a7a4 @varbind_list=[#<SNMP::VarBind:0x2d7bcd8f
@value="teststring",
@name=[1.11.12.13.14.15]>], @timestamp=#<SNMP::TimeTicks:0x1af47e9d
@value=55>, @generic_trap=6,
@enterprise=[1.2.3.4.5.6], @source_ip="127.0.0.1",
@agent_addr=#<SNMP::IpAddress:0x29a4833e @value="\xC0\xC1\xC2\xC3">,
@specific_trap=99>
```

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
snmptrap {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{}
codec	codec	No	"plain"
community	array	No	"public"
enable_metric	boolean	No	true
host	string	No	"0.0.0.0"
id	string	No	
port	number	No	1062
tags	array	No	
type	string	No	
yamlmibdir	string	No	

Details

`add_field`

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

`community`

- Value type is [array](#)
- Default value is "public"

SNMP Community String to listen for.

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`host`

- Value type is [string](#)
- Default value is "0.0.0.0"

The address to listen on

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
  stdout {  
    id => "my_plugin_id"  
  }  
}  
port
```

- Value type is [number](#)
- Default value is 1062

The port to listen on. Remember that ports less than 1024 (privileged ports) may require root to use. hence the default of 1062.

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

yamlmibdir

- Value type is [string](#)
- There is no default value for this setting.

directory of YAML MIB maps (same format ruby-snmp uses)

sqlite

- Version: 3.0.0
- Released on: 2016-09-10
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
 bin/logstash-plugin install logstash-input-sqlite.

Read rows from an sqlite database.

This is most useful in cases where you are logging directly to a table. Any tables being watched must have an `id` column that is monotonically increasing.

All tables are read by default except:

- ones matching `sqlite_%` - these are internal/administrative tables for sqlite
- `since_table` - this is used by this plugin to track state.

Example

```
% sqlite /tmp/example.db
sqlite> CREATE TABLE weblogs (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    ip STRING,
    request STRING,
    response INTEGER);
sqlite> INSERT INTO weblogs (ip, request, response)
    VALUES ("1.2.3.4", "/index.html", 200);
```

Then with this logstash config:

```
input {
  sqlite {
    path => "/tmp/example.db"
    type => weblogs
  }
}

output {
  stdout {
    debug => true
  }
}
```

Sample output:

```
{
  "@source"      => "sqlite://sadness/tmp/x.db",
  "@tags"        => [],
  "@fields"      => {
```

```

"ip"      => "1.2.3.4",
"request" => "/index.html",
"response" => 200
},
"@timestamp" => "2013-05-29T06:16:30.850Z",
"@source_host" => "sadness",
"@source_path" => "/tmp/x.db",
"@message" => "",
"@type" => "foo"
}

```

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
sqlite {
    path => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
batch	number	No	5
codec	codec	No	"plain"
enable_metric	boolean	No	true
exclude_tables	array	No	[]
id	string	No	
path	string	Yes	
tags	array	No	
type	string	No	

Details

[*add_field*](#)

- Value type is [hash](#)
- Default value is { }

Add a field to an event

batch

- Value type is [number](#)
- Default value is 5

How many rows to fetch at a time from each SELECT call.

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

enable_metric

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

exclude_tables

- Value type is [array](#)
- Default value is []

Any tables to exclude by name. By default all tables are followed.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique ID to the plugin instance, this ID is used for tracking information for a specific configuration of the plugin.

``` output { stdout { id => "ABC" } } ```

If you don't explicitly set this variable Logstash will generate a unique name.

*path*

- This is a required setting.

- Value type is [string](#)
- There is no default value for this setting.

The path to the sqlite database file.

*tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

*type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

## sqs

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

Pull events from an Amazon Web Services Simple Queue Service (SQS) queue.

SQS is a simple, scalable queue system that is part of the Amazon Web Services suite of tools.

Although SQS is similar to other queuing systems like AMQP, it uses a custom API and requires that you have an AWS account. See <http://aws.amazon.com/sqs/> for more details on how SQS works, what the pricing schedule looks like and how to setup a queue.

To use this plugin, you **must**:

- Have an AWS account
- Setup an SQS queue
- Create an identity that has access to consume messages from the queue.

The "consumer" identity must have the following permissions on the queue:

- sqs:ChangeMessageVisibility
- sqs:ChangeMessageVisibilityBatch
- sqs:DeleteMessage
- sqs:DeleteMessageBatch
- sqs:GetQueueAttributes
- sqs:GetQueueUrl
- sqs>ListQueues
- sqs:ReceiveMessage

Typically, you should setup an IAM policy, create a user and apply the IAM policy to the user. A sample policy is as follows:

```
{
 "Statement": [
 {
 "Action": [
 "sqs:ChangeMessageVisibility",
 "sqs:ChangeMessageVisibilityBatch",
 "sqs:GetQueueAttributes",
 "sqs:GetQueueUrl",
 "sqs>ListQueues",
 "sqs:SendMessage",
 "sqs:SendMessageBatch"
],
 "Effect": "Allow",
 "Resource": [
 "arn:aws:sqs:us-east-1:123456789012:Logstash"
]
 }
]
}
```

```

]
 }
]
}
```

See <http://aws.amazon.com/iam/> for more details on setting up AWS identities.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
sqs {
 queue => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">access_key_id</a>	<a href="#">string</a>	No	
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">aws_credentials_file</a>	<a href="#">string</a>	No	
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">id_field</a>	<a href="#">string</a>	No	
<a href="#">md5_field</a>	<a href="#">string</a>	No	
<a href="#">polling_frequency</a>	<a href="#">number</a>	No	20
<a href="#">proxy_uri</a>	<a href="#">string</a>	No	
<a href="#">queue</a>	<a href="#">string</a>	Yes	
<a href="#">region</a>	<a href="#">string</a> , one of ["us-east-1", "us-west-1", "us-west-2", "eu-central-1", "eu-west-1", "ap-southeast-1", "ap-southeast-2", "ap-northeast-1", "ap-northeast-2", "sa-east-1", "us-gov-west-1", "cn-north-1", "ap-south-1"]	No	"us-east-1"
<a href="#">secret_access_key</a>	<a href="#">string</a>	No	
<a href="#">sent_timestamp_field</a>	<a href="#">string</a>	No	

Setting	Input type	Required	Default value
<a href="#">session_token</a>	<a href="#">string</a>	No	
<a href="#">tags</a>	<a href="#">array</a>	No	
<a href="#">threads</a>	<a href="#">number</a>	No	1
<a href="#">type</a>	<a href="#">string</a>	No	

## Details

### [access\\_key\\_id](#)

- Value type is [string](#)
- There is no default value for this setting.

This plugin uses the AWS SDK and supports several ways to get credentials, which will be tried in this order:

1. Static configuration, using `access_key_id` and `secret_access_key` params in logstash plugin config
2. External credentials file specified by `aws_credentials_file`
3. Environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`
4. Environment variables `AMAZON_ACCESS_KEY_ID` and `AMAZON_SECRET_ACCESS_KEY`
5. IAM Instance Profile (available when running inside EC2)

### [add\\_field](#)

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

### [aws\\_credentials\\_file](#)

- Value type is [string](#)
- There is no default value for this setting.

Path to YAML file containing a hash of AWS credentials. This file will only be loaded if `access_key_id` and `secret_access_key` aren't set. The contents of the file should look like this:

```
:access_key_id: "12345"
:secret_access_key: "54321"
```

### *codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

### *enable\_metric*

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### *id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

### *id\_field*

- Value type is [string](#)
- There is no default value for this setting.

Name of the event field in which to store the SQS message ID

### *md5\_field*

- Value type is [string](#)
- There is no default value for this setting.

Name of the event field in which to store the SQS message MD5 checksum

### *polling\_frequency*

- Value type is [number](#)

- Default value is 20

Polling frequency, default is 20 seconds

### *proxy\_uri*

- Value type is [string](#)
- There is no default value for this setting.

URI to proxy server if required

### *queue*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Name of the SQS Queue name to pull messages from. Note that this is just the name of the queue, not the URL or ARN.

### *region*

- Value can be any of: us-east-1, us-west-1, us-west-2, eu-central-1, eu-west-1, ap-southeast-1, ap-southeast-2, ap-northeast-1, ap-northeast-2, sa-east-1, us-gov-west-1, cn-north-1, ap-south-1
- Default value is "us-east-1"

The AWS Region

### *secret\_access\_key*

- Value type is [string](#)
- There is no default value for this setting.

The AWS Secret Access Key

### *sent\_timestamp\_field*

- Value type is [string](#)
- There is no default value for this setting.

Name of the event field in which to store the SQS message Sent Timestamp

### *session\_token*

- Value type is [string](#)
- There is no default value for this setting.

The AWS Session token for temporary credential

*tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

*threads*

- Value type is [number](#)
- Default value is 1

*type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

## stdin

- Version: 3.2.2
- Released on: 2017-01-17
- [Changelog](#)

Read events from standard input.

By default, each event is assumed to be one line. If you want to join lines, you'll want to use the multiline codec.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
stdin {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">tags</a>	<a href="#">array</a>	No	
<a href="#">type</a>	<a href="#">string</a>	No	

## Details

### [add\\_field](#)

- Value type is [hash](#)
- Default value is { }

Add a field to an event

### `codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

### `tags`

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

### `type`

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

## stomp

- Version: 3.0.2
- Released on: 2016-10-11
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
 bin/logstash-plugin install logstash-input-stomp.

Handle disconnects

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
stomp {
 destination => ...
 host => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
codec	<a href="#">codec</a>	No	"plain"
debug	<a href="#">boolean</a>	No	false
destination	<a href="#">string</a>	Yes	
enable_metric	<a href="#">boolean</a>	No	true
host	<a href="#">string</a>	Yes	"localhost"
id	<a href="#">string</a>	No	
password	<a href="#">password</a>	No	""
port	<a href="#">number</a>	No	61613
reconnect	<a href="#">boolean</a>	No	true
reconnect_interval	<a href="#">number</a>	No	30
tags	<a href="#">array</a>	No	
type	<a href="#">string</a>	No	
user	<a href="#">string</a>	No	""

Setting	Input type	Required	Default value
vhost	<a href="#">string</a>	No	nil

## Details

[\*add\\_field\*](#)

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

[\*codec\*](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

[\*debug\*](#)

- Value type is [boolean](#)
- Default value is `false`

Enable debugging output?

[\*destination\*](#)

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The destination to read events from.

Example: `/topic/logstash`

[\*enable\\_metric\*](#)

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### *host*

- This is a required setting.
- Value type is [string](#)
- Default value is "localhost"

The address of the STOMP server.

### *id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

password

- Value type is [password](#)
- Default value is ""

The password to authenticate with.

port

- Value type is [number](#)
- Default value is 61613

The port to connect to on your STOMP server.

reconnect

- Value type is [boolean](#)
- Default value is `true`

Auto reconnect

reconnect_interval

- Value type is [number](#)

- Default value is 30

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

user

- Value type is [string](#)
- Default value is ""

The username to authenticate with.

vhost

- Value type is [string](#)
- Default value is `nil`

The vhost to use

syslog

- Version: 3.2.0
- Released on: 2016-11-21
- [Changelog](#)

Read syslog messages as events over the network.

This input is a good choice if you already use syslog today. It is also a good choice if you want to receive logs from appliances and network devices where you cannot run your own log collector.

Of course, *syslog* is a very muddy term. This input only supports RFC3164 syslog with some small modifications. The date format is allowed to be RFC3164 style or ISO8601. Otherwise the rest of RFC3164 must be obeyed. If you do not use RFC3164, do not use this input.

For more information see the [RFC3164 page](#).

Note: This input will start listeners on both TCP and UDP.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
syslog {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
codec	codec	No	"plain"
enable_metric	boolean	No	true
facility_labels	array	No	["kernel", "user-level", "mail", "system", "security/authorization", "syslogd", "line printer", "network news", "UUCP", "clock", "security/authorization", "FTP", "NTP", "log audit", "log alert", "clock", "local0", "local1", "local2", "local3", "local4", "local5", "local6", "local7"]

Setting	Input type	Required	Default value
host	string	No	"0.0.0.0"
id	string	No	
locale	string	No	
port	number	No	514
proxy protocol	boolean	No	false
severity labels	array	No	["Emergency", "Alert", "Critical", "Error", "Warning", "Notice", "Informational", "Debug"]
tags	array	No	
timezone	string	No	
type	string	No	
use labels	boolean	No	true

Details

[*add_field*](#)

- Value type is [hash](#)
- Default value is {}

Add a field to an event

[*codec*](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

[*enable_metric*](#)

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

facility_labels

- Value type is [array](#)
- Default value is `["kernel", "user-level", "mail", "system", "security/authorization", "syslogd", "line printer", "network news", "UUCP", "clock", "security/authorization", "FTP", "NTP", "log audit", "log alert", "clock", "local0", "local1", "local2", "local3", "local4", "local5", "local6", "local7"]`

Labels for facility levels. These are defined in RFC3164.

host

- Value type is [string](#)
- Default value is `"0.0.0.0"`

The address to listen on.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
  stdout {  
    id => "my_plugin_id"  
  }  
}
```

locale

- Value type is [string](#)
- There is no default value for this setting.

Specify a locale to be used for date parsing using either IETF-BCP47 or POSIX language tag. Simple examples are `en,en-US` for BCP47 or `en_US` for POSIX. If not specified, the platform default will be used.

The locale is mostly necessary to be set for parsing month names (pattern with MMM) and weekday names (pattern with EEE).

port

- Value type is [number](#)

- Default value is 514

The port to listen on. Remember that ports less than 1024 (privileged ports) may require root to use.

proxy_protocol

- Value type is [boolean](#)
- Default value is `false`

Proxy protocol support, only v1 is supported at this time

<http://www.haproxy.org/download/1.5/doc/proxy-protocol.txt>

severity_labels

- Value type is [array](#)
- Default value is `["Emergency", "Alert", "Critical", "Error", "Warning", "Notice", "Informational", "Debug"]`

Labels for severity levels. These are defined in RFC3164.

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

timezone

- Value type is [string](#)
- There is no default value for this setting.

Specify a time zone canonical ID to be used for date parsing. The valid IDs are listed on the [Joda.org available time zones page](<http://joda-time.sourceforge.net/timezones.html>). This is useful in case the time zone cannot be extracted from the value, and is not the platform default. If this is not specified the platform default will be used. Canonical ID is good as it takes care of daylight saving time for you For example, `America/Los_Angeles` or `Europe/France` are valid IDs.

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

use_labels

- Value type is [boolean](#)
- Default value is `true`

Use label parsing for severity and facility levels.

tcp

- Version: 4.1.0
- Released on: 2016-11-21
- [Changelog](#)

Read events over a TCP socket.

Like stdin and file inputs, each event is assumed to be one line of text.

Can either accept connections from clients or connect to a server, depending on `mode`.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
tcp {
    port => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
codec	codec	No	"plain"
enable_metric	boolean	No	true
host	string	No	"0.0.0.0"
id	string	No	
mode	string , one of ["server", "client"]	No	"server"
port	number	Yes	
proxy_protocol	boolean	No	false
ssl_cert	a valid filesystem path	No	
ssl_enable	boolean	No	false
ssl_extra_chain_certs	array	No	[]
ssl_key	a valid filesystem path	No	
ssl_key_passphrase	password	No	nil
ssl_verify	boolean	No	true

Setting	Input type	Required	Default value
tags	array	No	
type	string	No	

Details

add_field

- Value type is [hash](#)
- Default value is {}

Add a field to an event

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

data_timeout (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [number](#)
- Default value is -1

enable_metric

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

host

- Value type is [string](#)
- Default value is "0.0.0.0"

When mode is `server`, the address to listen on. When mode is `client`, the address to connect to.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
  stdout {
    id => "my_plugin_id"
  }
}
```

mode

- Value can be any of: `server`, `client`
- Default value is "`server`"

Mode to operate in. `server` listens for client connections, `client` connects to a server.

port

- This is a required setting.
- Value type is [number](#)
- There is no default value for this setting.

When mode is `server`, the port to listen on. When mode is `client`, the port to connect to.

proxy_protocol

- Value type is [boolean](#)
- Default value is `false`

Proxy protocol support, only v1 is supported at this time

<http://www.haproxy.org/download/1.5/doc/proxy-protocol.txt>

ssl_cacert (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [path](#)
- There is no default value for this setting.

The SSL CA certificate, chainfile or CA path. The system CA path is automatically included.

ssl_cert

- Value type is [path](#)
- There is no default value for this setting.

SSL certificate path

ssl_enable

- Value type is [boolean](#)
- Default value is `false`

Enable SSL (must be set for other `ssl_` options to take effect).

ssl_extra_chain_certs

- Value type is [array](#)
- Default value is `[]`

An Array of extra X509 certificates to be added to the certificate chain. Useful when the CA chain is not necessary in the system store.

ssl_key

- Value type is [path](#)
- There is no default value for this setting.

SSL key path

ssl_key_passphrase

- Value type is [password](#)
- Default value is `nil`

SSL key passphrase

ssl_verify

- Value type is [boolean](#)
- Default value is `true`

Verify the identity of the other end of the SSL connection against the CA. For input, sets the field `sslsubject` to that of the client certificate.

tags

- Value type is [array](#)

- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

`type`

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

twitter

- Version: 3.0.3
- Released on: 2016-10-13
- [Changelog](#)

Ingest events from the Twitter Streaming API.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
twitter {
    consumer_key => ...
    consumer_secret => ...
    oauth_token => ...
    oauth_token_secret => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
codec	codec	No	"plain"
consumer_key	string	Yes	
consumer_secret	password	Yes	
enable_metric	boolean	No	true
follows	array	No	
full_tweet	boolean	No	false
id	string	No	
ignore_re tweets	boolean	No	false
keywords	array	No	
languages	array	No	
locations	string	No	
oauth_token	string	Yes	
oauth_token_secret	password	Yes	
proxy_address	string	No	"127.0.0.1"

Setting	Input type	Required	Default value
proxy_port	number	No	3128
rate_limit_reset_in	number	No	300
tags	array	No	
type	string	No	
use_proxy	boolean	No	false
use_samples	boolean	No	false

Details

add_field

- Value type is [hash](#)
- Default value is {}

Add a field to an event

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

consumer_key

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Your Twitter App's consumer key

Don't know what this is? You need to create an "application" on Twitter, see this url:
<https://dev.twitter.com/apps/new>

consumer_secret

- This is a required setting.
- Value type is [password](#)
- There is no default value for this setting.

Your Twitter App's consumer secret

If you don't have one of these, you can create one by registering a new application with Twitter:
<https://dev.twitter.com/apps/new>

enable_metric

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

follows

- Value type is [array](#)
- There is no default value for this setting.

A comma separated list of user IDs, indicating the users to return statuses for in the Twitter stream. See <https://dev.twitter.com/streaming/overview/request-parameters#follow> for more details.

full_tweet

- Value type is [boolean](#)
- Default value is `false`

Record full tweet object as given to us by the Twitter Streaming API.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
  stdout {
    id => "my_plugin_id"
  }
}
```

ignore_re tweets

- Value type is [boolean](#)
- Default value is `false`

Lets you ignore the retweets coming out of the Twitter API. Default ⇒ false

keywords

- Value type is [array](#)
- There is no default value for this setting.

Any keywords to track in the Twitter stream. For multiple keywords, use the syntax ["foo", "bar"]. There's a logical OR between each keyword string listed and a logical AND between words separated by spaces per keyword string. See

<https://dev.twitter.com/streaming/overview/request-parameters#track> for more details.

The wildcard "*" option is not supported. To ingest a sample stream of all tweets, the use_samples option is recommended.

languages

- Value type is [array](#)
- There is no default value for this setting.

A list of BCP 47 language identifiers corresponding to any of the languages listed on Twitter's advanced search page will only return tweets that have been detected as being written in the specified languages.

locations

- Value type is [string](#)
- There is no default value for this setting.

A comma-separated list of longitude, latitude pairs specifying a set of bounding boxes to filter tweets by. See <https://dev.twitter.com/streaming/overview/request-parameters#locations> for more details.

oauth_token

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Your oauth token.

To get this, login to Twitter with whatever account you want, then visit
<https://dev.twitter.com/apps>

Click on your app (used with the consumer_key and consumer_secret settings) Then at the bottom of the page, click *Create my access token* which will create an oauth token and secret bound to your account and that application.

oauth_token_secret

- This is a required setting.
- Value type is [password](#)
- There is no default value for this setting.

Your oauth token secret.

To get this, login to Twitter with whatever account you want, then visit
<https://dev.twitter.com/apps>

Click on your app (used with the consumer_key and consumer_secret settings) Then at the bottom of the page, click *Create my access token* which will create an oauth token and secret bound to your account and that application.

proxy_address

- Value type is [string](#)
- Default value is "127.0.0.1"

Location of the proxy, by default the same machine as the one running this LS instance

proxy_port

- Value type is [number](#)
- Default value is 3128

Port where the proxy is listening, by default 3128 (squid)

rate_limit_reset_in

- Value type is [number](#)
- Default value is 300

Duration in seconds to wait before retrying a connection when twitter responds with a 429 TooManyRequests In some cases the *x-rate-limit-reset* header is not set in the response and <error>.rate_limit.reset_in is nil. If this occurs then we use the integer specified here. The default is 5 minutes.

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

use_proxy

- Value type is [boolean](#)
- Default value is `false`

When to use a proxy to handle the connections

use_samples

- Value type is [boolean](#)
- Default value is `false`

Returns a small random sample of all public statuses. The tweets returned by the default access level are the same, so if two different clients connect to this endpoint, they will see the same tweets. If set to true, the keywords, follows, locations, and languages options will be ignored. Default ⇒ false

udp

- Version: 3.1.0
- Released on: 2016-11-24
- [Changelog](#)

Read messages as events over the network via udp. The only required configuration item is `port`, which specifies the udp port logstash will listen on for event streams.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
udp {
    port => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>add_field</code>	hash	No	<code>{ }</code>
<code>buffer_size</code>	number	No	65536
<code>codec</code>	codec	No	"plain"
<code>enable_metric</code>	boolean	No	true
<code>host</code>	string	No	"0.0.0.0"
<code>id</code>	string	No	
<code>port</code>	number	Yes	
<code>queue_size</code>	number	No	2000
<code>receive_buffer_bytes</code>	number	No	
<code>tags</code>	array	No	
<code>type</code>	string	No	
<code>workers</code>	number	No	2

Details

add_field

- Value type is [hash](#)
- Default value is {}

Add a field to an event

buffer_size

- Value type is [number](#)
- Default value is 65536

The maximum packet size to read from the network

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

enable_metric

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

host

- Value type is [string](#)
- Default value is "0.0.0.0"

The address which logstash will listen on.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
  stdout {
    id => "my_plugin_id"
  }
}
```

port

- This is a required setting.
- Value type is [number](#)
- There is no default value for this setting.

The port which logstash will listen on. Remember that ports less than 1024 (privileged ports) may require root or elevated privileges to use.

queue_size

- Value type is [number](#)
- Default value is 2000

This is the number of unprocessed UDP packets you can hold in memory before packets will start dropping.

receive_buffer_bytes

- Value type is [number](#)
- There is no default value for this setting.

The socket receive buffer size in bytes. If option is not set, the operating system default is used. The operating system will use the max allowed value if `receive_buffer_bytes` is larger than allowed. Consult your operating system documentation if you need to increase this max allowed value.

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

workers

- Value type is [number](#)
- Default value is 2

Number of threads processing packets

unix

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

Read events over a UNIX socket.

Like `stdin` and `file` inputs, each event is assumed to be one line of text.

Can either accept connections from clients or connect to a server, depending on `mode`.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
unix {
    path => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>add_field</code>	hash	No	{ }
<code>codec</code>	codec	No	"plain"
<code>data_timeout</code>	number	No	-1
<code>enable_metric</code>	boolean	No	true
<code>force_unlink</code>	boolean	No	false
<code>id</code>	string	No	
<code>mode</code>	string , one of ["server", "client"]	No	"server"
<code>path</code>	string	Yes	
<code>tags</code>	array	No	
<code>type</code>	string	No	

Details

`add_field`

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

`data_timeout`

- Value type is [number](#)
- Default value is -1

The *read* timeout in seconds. If a particular connection is idle for more than this timeout period, we will assume it is dead and close it.

If you never want to timeout, use -1.

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`force_unlink`

- Value type is [boolean](#)
- Default value is `false`

Remove socket file in case of EADDRINUSE failure

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when

you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
    stdout {  
        id => "my_plugin_id"  
    }  
}  
mode
```

- Value can be any of: `server`, `client`
- Default value is "`server`"

Mode to operate in. `server` listens for client connections, `client` connects to a server.

path

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

When mode is `server`, the path to listen on. When mode is `client`, the path to connect to.

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

varnishlog

- Version: 3.0.0
- Released on: 2016-09-10
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
`bin/logstash-plugin install logstash-input-varnishlog`.

Read from varnish cache's shared memory log

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
varnishlog {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
codec	codec	No	"plain"
enable_metric	boolean	No	true
id	string	No	
tags	array	No	
threads	number	No	1
type	string	No	

Details

`add_field`

- Value type is [hash](#)
- Default value is { }

Add a field to an event

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

enable_metric

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

### *tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

### *threads*

- Value type is [number](#)
- Default value is 1

### *type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

## websocket

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-input-websocket`.

Read events over the websocket protocol.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
websocket {
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
codec	<a href="#">codec</a>	No	"json"
mode	<a href="#">string</a> , one of ["server", "client"]	No	"client"
tags	<a href="#">array</a>	No	
type	<a href="#">string</a>	No	
url	<a href="#">string</a>	No	"0.0.0.0"

## Details

### [`add\_field`](#)

- Value type is [hash](#)
- Default value is { }

Add a field to an event

### [`codec`](#)

- Value type is [codec](#)
- Default value is "json"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

### *mode*

- Value can be any of: `server`, `client`
- Default value is "`client`"

Operate as a client or a server.

Client mode causes this plugin to connect as a websocket client to the URL given. It expects to receive events as websocket messages.

(NOT IMPLEMENTED YET) Server mode causes this plugin to listen on the given URL for websocket clients. It expects to receive events as websocket messages from these clients.

### *tags*

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

### *type*

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

### *url*

- Value type is [string](#)
- Default value is "`0.0.0.0`"

The url to connect to or serve from



## wmi

- Version: 3.0.0
- Released on: 2016-09-10
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
bin/logstash-plugin install logstash-input-wmi.

Collect data from WMI query

This is useful for collecting performance metrics and other data which is accessible via WMI on a Windows host

Example:

```
input {
 wmi {
 query => "select * from Win32_Process"
 interval => 10
 }
 wmi {
 query => "select PercentProcessorTime from
Win32_PerfFormattedData_PerfOS_Processor where name = '_Total'"
 }
 wmi { # Connect to a remote host
 query => "select * from Win32_Process"
 host => "MyRemoteHost"
 user => "mydomain\myuser"
 password => "Password"
 }
}
```

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
wmi {
 query => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
codec	<a href="#">codec</a>	No	"plain"
enable_metric	<a href="#">boolean</a>	No	true
host	<a href="#">string</a>	No	"localhost"
id	<a href="#">string</a>	No	
interval	<a href="#">number</a>	No	10
namespace	<a href="#">string</a>	No	"root\\cimv2"
password	<a href="#">password</a>	No	
query	<a href="#">string</a>	Yes	
tags	<a href="#">array</a>	No	
type	<a href="#">string</a>	No	
user	<a href="#">string</a>	No	

## Details

### [add\\_field](#)

- Value type is [hash](#)
- Default value is { }

Add a field to an event

### [codec](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

### [enable\\_metric](#)

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

*host*

- Value type is [string](#)
- Default value is "localhost"

Host to connect to ( Defaults to localhost )

*id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

interval

- Value type is [number](#)
- Default value is 10

Polling interval

namespace

- Value type is [string](#)
- Default value is "root\\cimv2"

Namespace when doing remote connections

password

- Value type is [password](#)
- There is no default value for this setting.

Password when doing remote connections

query

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

WMI query

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

user

- Value type is [string](#)
- There is no default value for this setting.

Username when doing remote connections

xmpp

- Version: 3.1.1
- Released on: 2016-07-14
- [Changelog](#)

load the MUC Client anyway, its mocked in testing This input allows you to receive events over XMPP/Jabber.

This plugin can be used for accepting events from humans or applications XMPP, or you can use it for PubSub or general message passing for logstash to logstash.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
xmpp {
    password => ...
    user => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
codec	codec	No	"plain"
enable_metric	boolean	No	true
host	string	No	
id	string	No	
password	password	Yes	
rooms	array	No	
tags	array	No	
type	string	No	
user	string	Yes	

Details

`add_field`

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`host`

- Value type is [string](#)
- There is no default value for this setting.

The xmpp server to connect to. This is optional. If you omit this setting, the host on the user/identity is used. (`foo.com` for `user@foo.com`)

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
  stdout {
    id => "my_plugin_id"
  }
}
```

password

- This is a required setting.
- Value type is [password](#)
- There is no default value for this setting.

The xmpp password for the user/identity.

rooms

- Value type is [array](#)
- There is no default value for this setting.

if muc/multi-user-chat required, give the name of the room that you want to join:
room@conference.domain/nick

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

user

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The user or resource ID, like `foo@example.com`.

zenoss

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-input-zenoss`.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
zenoss {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
ack	boolean	No	true
add_field	hash	No	{ }
arguments	array	No	{ }
auto_delete	boolean	No	false
automatic_recovery	boolean	No	true
codec	codec	No	"plain"
connect_retry_interval	number	No	1
connection_timeout	number	No	
durable	boolean	No	false
exchange	string	No	"zenoss.zenevents"
exclusive	boolean	No	false
heartbeat	number	No	
host	string	No	"localhost"
key	string	No	"zenoss.zenevent.#"
passive	boolean	No	false
password	password	No	"zenoss"
port	number	No	5672
prefetch_count	number	No	256
queue	string	No	""

Setting	Input type	Required	Default value
ssl	boolean	No	false
tags	array	No	
threads	number	No	1
type	string	No	
user	string	No	"zenoss"
verify_ssl	boolean	No	false
vhost	string	No	"/zenoss"

Details

ack

- Value type is [boolean](#)
- Default value is `true`

add_field

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

arguments

- Value type is [array](#)
- Default value is `{ }`

auto_delete

- Value type is [boolean](#)
- Default value is `false`

automatic_recovery

- Value type is [boolean](#)
- Default value is `true`

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

connect_retry_interval

- Value type is [number](#)
- Default value is 1

connection_timeout

- Value type is [number](#)
- There is no default value for this setting.

debug (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [boolean](#)
- Default value is false

durable

- Value type is [boolean](#)
- Default value is false

exchange

- Value type is [string](#)
- Default value is "zenoss.zenevents"

The name of the exchange to bind the queue. This is analogous to the *rabbitmq output* [config name](..outputs/rabbitmq)

exclusive

- Value type is [boolean](#)
- Default value is false

heartbeat

- Value type is [number](#)
- There is no default value for this setting.

host

- Value type is [string](#)
- Default value is "localhost"

Your rabbitmq server address

key

- Value type is [string](#)
- Default value is "zenoss.zenevent.#"

The routing key to use. This is only valid for direct or fanout exchanges

- Routing keys are ignored on topic exchanges.
- Wildcards are not valid on direct exchanges.

passive

- Value type is [boolean](#)
- Default value is `false`

password

- Value type is [password](#)
- Default value is "zenoss"

Your rabbitmq password

port

- Value type is [number](#)
- Default value is 5672

prefetch_count

- Value type is [number](#)
- Default value is 256

queue

- Value type is [string](#)
- Default value is ""

ssl

- Value type is [boolean](#)
- Default value is `false`

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

threads

- Value type is [number](#)
- Default value is 1

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

user

- Value type is [string](#)
- Default value is "zenoss"

Your rabbitmq username

verify_ssl

- Value type is [boolean](#)
- Default value is `false`

vhost

- Value type is [string](#)
- Default value is "/zenoss"

The vhost to use. If you don't know what this is, leave the default.

zeromq

NOTE: This is a community-maintained plugin!

Read events over a 0MQ SUB socket.

You need to have the 0mq 2.1.x library installed to be able to use this input plugin.

The default settings will create a subscriber binding to `tcp://127.0.0.1:2120` waiting for connecting publishers.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
zeromq {
    topology => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
address	array	No	["tcp://*:2120"]
codec	codec	No	"json"
mode	string , one of ["server", "client"]	No	"server"
sender	string	No	
sockopt	hash	No	
tags	array	No	
topic	array	No	
topology	string , one of ["pushpull", "pubsub", "pair"]	Yes	
type	string	No	

Details

`add_field`

- Value type is [hash](#)
- Default value is `{ }`

Add a field to an event

`address`

- Value type is [array](#)
- Default value is `["tcp://*:2120"]`

0mq socket address to connect or bind Please note that `inproc://` will not work with logstash as each we use a context per thread. By default, inputs bind/listen and outputs connect

`codec`

- Value type is [codec](#)
- Default value is "json"

The codec used for input data. Input codecs are a convenient method for decoding your data before it enters the input, without needing a separate filter in your Logstash pipeline.

`mode`

- Value can be any of: `server`, `client`
- Default value is "server"

mode server mode binds/listens client mode connects

`sender`

- Value type is [string](#)
- There is no default value for this setting.

sender overrides the sender to set the source of the event default is `zmq+topology://type/`

`sockopt`

- Value type is [hash](#)
- There is no default value for this setting.

0mq socket options This exposes `zmq_setsockopt` for advanced tuning see <http://api.zeromq.org/2-1:zmq-setsockopt> for details

This is where you would set values like:

Logstash 5.2 Configuration Guide

- `ZMQ::HWM` - high water mark
- `ZMQ::IDENTITY` - named queues
- `ZMQ::SWAP_SIZE` - space for disk overflow

example: `sockopt => ["ZMQ::HWM", 50, "ZMQ::IDENTITY", "my_named_queue"]`

tags

- Value type is [array](#)
- There is no default value for this setting.

Add any number of arbitrary tags to your event.

This can help with processing later.

topic

- Value type is [array](#)
- There is no default value for this setting.

0mq topic This is used for the pubsub topology only On inputs, this allows you to filter messages by topic On outputs, this allows you to tag a message for routing NOTE: ZeroMQ does subscriber side filtering. NOTE: All topics have an implicit wildcard at the end You can specify multiple topics here

topology

- This is a required setting.
- Value can be any of: `pushpull`, `pubsub`, `pair`
- There is no default value for this setting.

0mq topology The default logstash topologies work as follows:

- `pushpull` - inputs are pull, outputs are push
- `pubsub` - inputs are subscribers, outputs are publishers
- `pair` - inputs are clients, inputs are servers

If the predefined topology flows don't work for you, you can change the `mode` setting TODO (luis) add req/rep MAYBE TODO (luis) add router/dealer

type

- Value type is [string](#)
- There is no default value for this setting.

Add a `type` field to all events handled by this input.

Types are used mainly for filter activation.

The type is stored as part of the event itself, so you can also use the type to search for it in Kibana.

If you try to set a type on an event that already has one (for example when you send an event from a shipper to an indexer) then a new input will not override the existing type. A type set at the shipper stays with that event for its life even when sent to another Logstash server.

Output plugins

An output plugin sends event data to a particular destination. Outputs are the final stage in the event pipeline.

The following output plugins are available below. For a list of Elastic supported plugins, please consult the [Support Matrix](#).

Plugin	Description	Github repository
boundary	Sends annotations to Boundary based on Logstash events	logstash-output-boundary
circonus	Sends annotations to Circonus based on Logstash events	logstash-output-circonus
cloudwatch	Aggregates and sends metric data to AWS CloudWatch	logstash-output-cloudwatch
csv	Writes events to disk in a delimited format	logstash-output-csv
datadog	Sends events to DataDogHQ based on Logstash events	logstash-output-datadog
datadog_metrics	Sends metrics to DataDogHQ based on Logstash events	logstash-output-datadog_metrics
elasticsearch	Stores logs in Elasticsearch	logstash-output-elasticsearch
email	Sends email to a specified address when output is received	logstash-output-email
exec	Runs a command for a matching event	logstash-output-exec
file	Writes events to files on disk	logstash-output-file
ganglia	Writes metrics to Ganglia's gmond	logstash-output-ganglia
gelf	Generates GELF formatted output for Graylog2	logstash-output-gelf
google_bigquery	Writes events to Google BigQuery	logstash-output-google_bigquery
google_cloud_storage	Writes events to Google Cloud Storage	logstash-output-google_cloud_storage
graphite	Writes metrics to Graphite	logstash-output-graphite
graphtastic	Sends metric data on Windows	logstash-output-graphtastic
hipchat	Writes events to HipChat	logstash-output-hipchat
http	Sends events to a generic HTTP or HTTPS endpoint	logstash-output-http
influxdb	Writes metrics to InfluxDB	logstash-output-influxdb

irc	Writes events to IRC	logstash-output-irc
jira	Writes strutured JSON events to JIRA	logstash-output-jira
juggernaut	Pushes messages to the Juggernaut websockets server	logstash-output-juggernaut
kafka	Writes events to a Kafka topic	logstash-output-kafka
librato	Sends metrics, annotations, and alerts to Librato based on Logstash events	logstash-output-librato
loggly	Ships logs to Loggly	logstash-output-loggly
lumberjack	Sends events using the <code>lumberjack</code> protocol	logstash-output-lumberjack
metriccatcher	Writes metrics to MetricCatcher	logstash-output-metriccatcher
mongodb	Writes events to MongoDB	logstash-output-mongodb
nagios	Sends passive check results to Nagios	logstash-output-nagios
nagios_nsca	Sends passive check results to Nagios using the NSCA protocol	logstash-output-nagios_nsca
newrelic	Sends logstash events to New Relic Insights as custom events	logstash-output-newrelic
opentsdb	Writes metrics to OpenTSDB	logstash-output-opentsdb
pagerduty	Sends notifications based on preconfigured services and escalation policies	logstash-output-pagerduty
pipe	Pipes events to another program's standard input	logstash-output-pipe
rabbitmq	Pushes events to a RabbitMQ exchange	logstash-output-rabbitmq
rackspace	Sends events to a Rackspace Cloud Queue service	logstash-output-rackspace
redis	Sends events to a Redis queue using the <code>RPUSH</code> command	logstash-output-redis
redmine	Creates tickets using the Redmine API	logstash-output-redmine
riak	Writes events to the Riak distributed key/value store	logstash-output-riak
riemann	Sends metrics to Riemann	logstash-output-riemann
s3	Sends Logstash events to the Amazon Simple Storage Service	logstash-output-s3
sns	Sends events to Amazon's Simple Notification Service	logstash-output-sns
solr_http	Stores and indexes logs in Solr	logstash-output-solr_http

<u>sqs</u>	Pushes events to an Amazon Web Services Simple Queue Service queue	<u>logstash-output-sqs</u>
<u>statsd</u>	Sends metrics using the <code>statsd</code> network daemon	<u>logstash-output-statsd</u>
<u>stdout</u>	Prints events to the standard output	<u>logstash-output-stdout</u>
<u>stomp</u>	Writes events using the STOMP protocol	<u>logstash-output-stomp</u>
<u>syslog</u>	Sends events to a <code>syslog</code> server	<u>logstash-output-syslog</u>
<u>tcp</u>	Writes events over a TCP socket	<u>logstash-output-tcp</u>
<u>udp</u>	Sends events over UDP	<u>logstash-output-udp</u>
<u>webhdfs</u>	Sends Logstash events to HDFS using the <code>webhdfs</code> REST API	<u>logstash-output-webhdfs</u>
<u>websocket</u>	Publishes messages to a websocket	<u>logstash-output-websocket</u>
<u>xmpp</u>	Posts events over XMPP	<u>logstash-output-xmpp</u>
<u>zabbix</u>	Sends events to a Zabbix server	<u>logstash-output-zabbix</u>
<u>zeromq</u>	Writes events to a ZeroMQ PUB socket	<u>logstash-output-zeromq</u>

boundary

- Version: 3.0.0
- Released on: 2016-09-09
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
`bin/logstash-plugin install logstash-output-boundary`.

This output lets you send annotations to Boundary based on Logstash events

Note that since Logstash maintains no state these will be one-shot events

By default the start and stop time will be the event timestamp

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
boundary {
    api_key => ...
    org_id => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
api_key	string	Yes	
auto	boolean	No	false
bsubtype	string	No	
btags	array	No	
btype	string	No	
codec	codec	No	"plain"
enable_metric	boolean	No	true
end_time	string	No	
id	string	No	
org_id	string	Yes	
start_time	string	No	

Setting	Input type	Required	Default value
workers	<>, >>	No	1

Details

[api_key](#)

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Your Boundary API key

[auto](#)

- Value type is [boolean](#)
- Default value is `false`

Auto If set to true, logstash will try to pull boundary fields out of the event. Any field explicitly set by config options will override these. `['type', 'subtype', 'creation_time', 'end_time', 'links', 'tags', 'loc']`

[bsubtype](#)

- Value type is [string](#)
- There is no default value for this setting.

Sub-Type

[btags](#)

- Value type is [array](#)
- There is no default value for this setting.

Tags Set any custom tags for this event Default are the Logstash tags if any

[btype](#)

- Value type is [string](#)
- There is no default value for this setting.

Type

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`end_time`

- Value type is [string](#)
- There is no default value for this setting.

End time Override the stop time Note that Boundary requires this to be seconds since epoch If overriding, it is your responsibility to type this correctly By default this is set to `event[@"@timestamp"].to_i`

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

### `org_id`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Your Boundary Org ID

*start\_time*

- Value type is [string](#)
- There is no default value for this setting.

Start time Override the start time Note that Boundary requires this to be seconds since epoch If overriding, it is your responsibility to type this correctly By default this is set to event["@timestamp"].to\_i

*workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

## circonus

- Version: 3.0.0
- Released on: 2016-09-09
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
 bin/logstash-plugin install logstash-output-circonus.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
circonus {
 annotation => ...
 api_token => ...
 app_name => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
annotation	<a href="#">hash</a>	Yes	{ }
api_token	<a href="#">string</a>	Yes	
app_name	<a href="#">string</a>	Yes	
codec	<a href="#">codec</a>	No	"plain"
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	
workers	<<, >>	No	1

## Details

### *annotation*

- This is a required setting.
- Value type is [hash](#)
- Default value is { }

**Annotations** Registers an annotation with Circonus. The only required field is `title` and `description`. `start` and `stop` will be set to the event timestamp. You can add any other optional annotation values as well. All values will be passed through `event.strftime`.

Example:

```
["title": "Logstash event", "description": "Logstash event for %{host}"]
or
["title": "Logstash event", "description": "Logstash event for %{host}",
"parent_id", "1"]
api_token
```

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

This output lets you send annotations to Circonus based on Logstash events

Your Circonus API Token

[app\\_name](#)

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Your Circonus App name This will be passed through `event.strftime` so variables are allowed here:

Example: `app_name => "%{myappname}"`

[codec](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

[enable\\_metric](#)

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

*id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

workers

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

cloudwatch

- Version: 3.0.3
- Released on: 2016-09-15
- [Changelog](#)

This output lets you aggregate and send metric data to AWS CloudWatch

Summary:

This plugin is intended to be used on a logstash indexer agent (but that is not the only way, see below.) In the intended scenario, one cloudwatch output plugin is configured, on the logstash indexer node, with just AWS API credentials, and possibly a region and/or a namespace. The output looks for fields present in events, and when it finds them, it uses them to calculate aggregate statistics. If the `metricname` option is set in this output, then any events which pass through it will be aggregated & sent to CloudWatch, but that is not recommended. The intended use is to NOT set the `metricname` option here, and instead to add a `CW_metricname` field (and other fields) to only the events you want sent to CloudWatch.

When events pass through this output they are queued for background aggregation and sending, which happens every minute by default. The queue has a maximum size, and when it is full aggregated statistics will be sent to CloudWatch ahead of schedule. Whenever this happens a warning message is written to logstash's log. If you see this you should increase the `queue_size` configuration option to avoid the extra API calls. The queue is emptied every time we send data to CloudWatch.

Note: when logstash is stopped the queue is destroyed before it can be processed. This is a known limitation of logstash and will hopefully be addressed in a future version.

Details:

There are two ways to configure this plugin, and they can be used in combination: event fields & per-output defaults

Event Field configuration... You add fields to your events in inputs & filters and this output reads those fields to aggregate events. The names of the fields read are configurable via the `field_*` options.

Per-output defaults... You set universal defaults in this output plugin's configuration, and if an event does not have a field for that option then the default is used.

Notice, the event fields take precedence over the per-output defaults.

At a minimum events must have a "metric name" to be sent to CloudWatch. This can be achieved either by providing a default here OR by adding a `CW_metricname` field. By default, if

no other configuration is provided besides a metric name, then events will be counted (Unit: Count, Value: 1) by their metric name (either a default or from their `CW_metricname` field)

Other fields which can be added to events to modify the behavior of this plugin are, `CW_namespace`, `CW_unit`, `CW_value`, and `CW_dimensions`. All of these field names are configurable in this output. You can also set per-output defaults for any of them. See below for details.

Read more about [AWS CloudWatch](#), and the specific of API endpoint this output uses, [PutMetricData](#)

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
cloudwatch {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
access_key_id	string	No	
aws_credentials_file	string	No	
batch_size	number	No	20
codec	codec	No	"plain"
dimensions	hash	No	
enable_metric	boolean	No	true
field_dimensions	string	No	"CW_dimensions"
field_metricname	string	No	"CW_metricname"
field_namespace	string	No	"CW_namespace"
field_unit	string	No	"CW_unit"
field_value	string	No	"CW_value"
id	string	No	
metricname	string	No	
namespace	string	No	"Logstash"
proxy_uri	string	No	

Setting	Input type	Required	Default value
queue_size	number	No	10000
region	string , one of ["us-east-1", "us-west-1", "us-west-2", "eu-central-1", "eu-west-1", "ap-southeast-1", "ap-southeast-2", "ap-northeast-1", "ap-northeast-2", "sa-east-1", "us-gov-west-1", "cn-north-1", "ap-south-1"]	No	"us-east-1"
secret access key	string	No	
session token	string	No	
timeframe	string	No	"1m"
unit	string , one of ["Seconds", "Microseconds", "Milliseconds", "Bytes", "Kilobytes", "Megabytes", "Gigabytes", "Terabytes", "Bits", "Kilobits", "Megabits", "Gigabits", "Terabits", "Percent", "Count", "Bytes/Second", "Kilobytes/Second", "Megabytes/Second", "Gigabytes/Second", "Terabytes/Second", "Bits/Second", "Kilobits/Second", "Megabits/Second", "Gigabits/Second", "Terabits/Second", "Count/Second", "None"]	No	"Count"
value	string	No	"1"
workers	<<, >>	No	1

Details

[access_key_id](#)

- Value type is [string](#)
- There is no default value for this setting.

This plugin uses the AWS SDK and supports several ways to get credentials, which will be tried in this order:

1. Static configuration, using `access_key_id` and `secret_access_key` params in logstash plugin config

2. External credentials file specified by `aws_credentials_file`
3. Environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`
4. Environment variables `AMAZON_ACCESS_KEY_ID` and `AMAZON_SECRET_ACCESS_KEY`
5. IAM Instance Profile (available when running inside EC2)

`aws_credentials_file`

- Value type is [string](#)
- There is no default value for this setting.

Path to YAML file containing a hash of AWS credentials. This file will only be loaded if `access_key_id` and `secret_access_key` aren't set. The contents of the file should look like this:

```
:access_key_id: "12345"  
:secret_access_key: "54321"
```

`batch_size`

- Value type is [number](#)
- Default value is 20

How many data points can be given in one call to the CloudWatch API

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

`dimensions`

- Value type is [hash](#)
- There is no default value for this setting.

The default dimensions [name, value, ...] to use for events which do not have a `cw_dimensions` field

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

field_dimensions

- Value type is [string](#)
- Default value is "CW_dimensions"

The name of the field used to set the dimensions on an event metric. The field named here, if present in an event, must have an array of one or more key & value pairs, for example...

```
add_field => [ "CW_dimensions", "Environment", "CW_dimensions", "prod" ] or,  
equivalently... add_field => [ "CW_dimensions", "Environment" ] add_field => [  
"CW_dimensions", "prod" ]
```

field_metricname

- Value type is [string](#)
- Default value is "CW_metricname"

The name of the field used to set the metric name on an event. The author of this plugin recommends adding this field to events in inputs & filters rather than using the per-output default setting so that one output plugin on your logstash indexer can serve all events (which of course had fields set on your logstash shippers.)

field_namespace

- Value type is [string](#)
- Default value is "CW_namespace"

The name of the field used to set a different namespace per event. Note: Only one namespace can be sent to CloudWatch per API call so setting different namespaces will increase the number of API calls and those cost money.

field_unit

- Value type is [string](#)
- Default value is "CW_unit"

The name of the field used to set the unit on an event metric

field_value

- Value type is [string](#)
- Default value is "CW_value"

The name of the field used to set the value (float) on an event metric

id

- Value type is [string](#)

- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
  stdout {
    id => "my_plugin_id"
  }
}
metricname
```

- Value type is [string](#)
- There is no default value for this setting.

The default metric name to use for events which do not have a `CW_metricname` field. Beware: If this is provided then all events which pass through this output will be aggregated and sent to CloudWatch, so use this carefully. Furthermore, when providing this option, you will probably want to also restrict events from passing through this output using event type, tag, and field matching

`namespace`

- Value type is [string](#)
- Default value is "Logstash"

The default namespace to use for events which do not have a `CW_namespace` field

`proxy_uri`

- Value type is [string](#)
- There is no default value for this setting.

URI to proxy server if required

`queue_size`

- Value type is [number](#)
- Default value is 10000

How many events to queue before forcing a call to the CloudWatch API ahead of `timeframe` schedule Set this to the number of events-per-timeframe you will be sending to CloudWatch to avoid extra API calls

region

- Value can be any of: us-east-1, us-west-1, us-west-2, eu-central-1, eu-west-1, ap-southeast-1, ap-southeast-2, ap-northeast-1, ap-northeast-2, sa-east-1, us-gov-west-1, cn-north-1, ap-south-1
- Default value is "us-east-1"

The AWS Region

secret_access_key

- Value type is [string](#)
- There is no default value for this setting.

The AWS Secret Access Key

session_token

- Value type is [string](#)
- There is no default value for this setting.

The AWS Session token for temporary credential

timeframe

- Value type is [string](#)
- Default value is "1m"

Constants aggregate_key members Units How often to send data to CloudWatch This does not affect the event timestamps, events will always have their actual timestamp (to-the-minute) sent to CloudWatch.

We only call the API if there is data to send.

See the Rufus Scheduler docs for an [explanation of allowed values](#)

unit

- Value can be any of: Seconds, Microseconds, Milliseconds, Bytes, Kilobytes, Megabytes, Gigabytes, Terabytes, Bits, Kilobits, Megabits, Gigabits, Terabits, Percent, Count, Bytes/Second, Kilobytes/Second, Megabytes/Second, Gigabytes/Second, Terabytes/Second, Bits/Second, Kilobits/Second, Megabits/Second, Gigabits/Second, Terabits/Second, Count/Second, None
- Default value is "Count"

The default unit to use for events which do not have a `cw_unit` field If you set this option you should probably set the "value" option along with it

value

- Value type is [string](#)
- Default value is "1"

The default value to use for events which do not have a `cw_value` field If provided, this must be a string which can be converted to a float, for example... "1", "2.34", ".5", and "0.67" If you set this option you should probably set the `unit` option along with it

workers

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

CSV

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

CSV output.

Write events to disk in CSV or other delimited format Based on the file output, many config values are shared Uses the Ruby csv library internally

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
csv {
    fields => ...
    path => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	codec	No	"plain"
create_if_deleted	boolean	No	true
csv_options	hash	No	{ }
dir_mode	number	No	-1
enable_metric	boolean	No	true
fields	array	Yes	
file_mode	number	No	-1
filename_failure	string	No	"_filepath_failures"
flush_interval	number	No	2
gzip	boolean	No	false
id	string	No	
path	string	Yes	
spreadsheet_safe	boolean	No	true
workers	<>	No	1

Details

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

`create_if_deleted`

- Value type is [boolean](#)
- Default value is `true`

If the configured file is deleted, but an event is handled by the plugin, the plugin will recreate the file. Default \Rightarrow true

`csv_options`

- Value type is [hash](#)
- Default value is `{}`

Options for CSV output. This is passed directly to the Ruby stdlib `to_csv` function. Full documentation is available on the [Ruby CSV documentation page](#). A typical use case would be to use alternative column or row separators eg: `csv_options => {"col_sep" => "\t", "row_sep" => "\r\n"}` gives tab separated data with windows line endings

`dir_mode`

- Value type is [number](#)
- Default value is `-1`

Dir access mode to use. Note that due to the bug in jruby system umask is ignored on linux: <https://github.com/jruby/jruby/issues/3426> Setting it to `-1` uses default OS value. Example: `"dir_mode" => 0750`

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

fields

- This is a required setting.
- Value type is [array](#)
- There is no default value for this setting.

The field names from the event that should be written to the CSV file. Fields are written to the CSV in the same order as the array. If a field does not exist on the event, an empty string will be written. Supports field reference syntax eg: `fields => ["field1", "[nested] [field]"]`.

file_mode

- Value type is [number](#)
- Default value is -1

File access mode to use. Note that due to the bug in jruby system umask is ignored on linux:

<https://github.com/jruby/jruby/issues/3426> Setting it to -1 uses default OS value. Example:

`"file_mode" => 0640`

filename_failure

- Value type is [string](#)
- Default value is `"_filepath_failures"`

If the generated path is invalid, the events will be saved into this file and inside the defined path.

flush_interval

- Value type is [number](#)
- Default value is 2

Flush interval (in seconds) for flushing writes to log files. 0 will flush on every message.

gzip

- Value type is [boolean](#)
- Default value is `false`

Gzip the output stream before writing to disk.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when

you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
  stdout {  
    id => "my_plugin_id"  
  }  
}  
path
```

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The path to the file to write. Event fields can be used here, like

`/var/log/logstash/%{host}/%{application}` One may also utilize the path option for date-based log rotation via the joda time format. This will use the event timestamp. E.g.: `path => "./test-%{+YYYY-MM-dd}.txt"` to create `./test-2013-05-29.txt`

If you use an absolute path you cannot start with a dynamic string. E.g: `/%{myfield}/`, `/test-%{myfield}/` are not valid paths

```
spreadsheet_safe
```

- Value type is [boolean](#)
- Default value is `true`

Option to not escape/munge string values. Please note turning off this option may not make the values safe in your spreadsheet application

```
workers
```

- Value type is [string](#)
- Default value is `1`

TODO remove this in Logstash 6.0 when we no longer support the `:legacy` type This is hacky, but it can only be herne

datadog

- Version: 3.0.0
- Released on: 2016-09-09
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
 bin/logstash-plugin install logstash-output-datadog.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
datadog {
    api_key => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
alert_type	string , one of ["info", "error", "warning", "success"]	No	
api_key	string	Yes	
codec	codec	No	"plain"
date_happened	string	No	
dd_tags	array	No	
enable_metric	boolean	No	true
id	string	No	
priority	string , one of ["normal", "low"]	No	
source_type_name	string , one of ["nagios", "hudson", "jenkins", "user", "my apps", "feed", "chef", "puppet", "git", "bitbucket", "fabric", "capistrano"]	No	"my apps"
text	string	No	"%{message}"
title	string	No	"Logstash event for %{host}"
workers	<<,>>	No	1

Details

`alert_type`

- Value can be any of: `info`, `error`, `warning`, `success`
- There is no default value for this setting.

Alert type

`api_key`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

This output lets you send events (for now, soon metrics) to DataDogHQ based on Logstash events

Note that since Logstash maintains no state these will be one-shot events

Your DatadogHQ API key

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

`date_happened`

- Value type is [string](#)
- There is no default value for this setting.

Date Happened

`dd_tags`

- Value type is [array](#)
- There is no default value for this setting.

Tags Set any custom tags for this event Default are the Logstash tags if any

enable_metric

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

### *priority*

- Value can be any of: `normal`, `low`
- There is no default value for this setting.

## Priority

### *source\_type\_name*

- Value can be any of: `nagios`, `hudson`, `jenkins`, `user`, `my apps`, `feed`, `chef`, `puppet`, `git`, `bitbucket`, `fabric`, `capistrano`
- Default value is "`my apps`"

## Source type name

### *text*

- Value type is [string](#)
- Default value is "`%{message}`"

## Text

### *title*

- Value type is [string](#)
- Default value is "`Logstash event for %{host}`"

## Title

*workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

## datadog\_metrics

- Version: 3.0.0
- Released on: 2016-09-09
- [Changelog](#)

This plugin does not ship with Logstash by default, but it is easy to install by running  
`bin/logstash-plugin install logstash-output-datadog_metrics.`

This output lets you send metrics to DataDogHQ based on Logstash events. Default `queue_size` and `timeframe` are low in order to provide near realtime alerting. If you do not use Datadog for alerting, consider raising these thresholds.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
datadog_metrics {
 api_key => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
api_key	<a href="#">string</a>	Yes	
codec	<a href="#">codec</a>	No	"plain"
dd_tags	<a href="#">array</a>	No	
device	<a href="#">string</a>	No	"#{metric_device}"
enable_metric	<a href="#">boolean</a>	No	true
host	<a href="#">string</a>	No	"#{host}"
id	<a href="#">string</a>	No	
metric_name	<a href="#">string</a>	No	"#{metric_name}"
metric_type	<a href="#">string</a> , one of ["gauge", "counter", "%{metric_type}"]	No	"#{metric_type}"
metric_value	<>	No	"#{metric_value}"
queue_size	<a href="#">number</a>	No	10
timeframe	<a href="#">number</a>	No	10

Setting	Input type	Required	Default value
workers	<>, >>	No	1

## Details

### `api_key`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Your DatadogHQ API key. <https://app.datadoghq.com/account/settings#api>

### `codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### `dd_tags`

- Value type is [array](#)
- There is no default value for this setting.

Set any custom tags for this event, default are the Logstash tags if any.

### `device`

- Value type is [string](#)
- Default value is "%{metric\_device}"

The name of the device that produced the metric.

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `host`

- Value type is [string](#)
- Default value is "`%{host}`"

The name of the host that produced the metric.

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

`metric_name`

- Value type is [string](#)
- Default value is "`%{metric_name}`"

The name of the time series.

`metric_type`

- Value can be any of: `gauge`, `counter`, `%{metric_type}`
- Default value is "`%{metric_type}`"

The type of the metric.

`metric_value`

- Value type is [string](#)
- Default value is "`%{metric_value}`"

The value.

`queue_size`

- Value type is [number](#)
- Default value is 10

How many events to queue before flushing to Datadog prior to schedule set in `@timeframe`

timeframe

- Value type is [number](#)
- Default value is 10

How often (in seconds) to flush queued events to Datadog

workers

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

elasticsearch

- Version: 6.2.6
- Released on: 2017-02-07
- [Changelog](#)

This plugin is the recommended method of storing logs in Elasticsearch. If you plan on using the Kibana web interface, you'll want to use this output.

This output only speaks the HTTP protocol. HTTP is the preferred protocol for interacting with Elasticsearch as of Logstash 2.0. We strongly encourage the use of HTTP over the node protocol for a number of reasons. HTTP is only marginally slower, yet far easier to administer and work with. When using the HTTP protocol one may upgrade Elasticsearch versions without having to upgrade Logstash in lock-step.

You can learn more about Elasticsearch at <https://www.elastic.co/products/elasticsearch>

Template management for Elasticsearch 5.x

Index template for this version (Logstash 5.0) has been changed to reflect Elasticsearch's mapping changes in version 5.0. Most importantly, the subfield for string multi-fields has changed from `.raw` to `.keyword` to match ES default behavior.

- Users installing ES 5.x and LS 5.x ** This change will not affect you and you will continue to use the ES defaults.
- Users upgrading from LS 2.x to LS 5.x with ES 5.x ** LS will not force upgrade the template, if `logstash` template already exists. This means you will still use `.raw` for sub-fields coming from 2.x. If you choose to use the new template, you will have to reindex your data after the new template is installed.

Retry Policy

The retry policy has changed significantly in the 2.2.0 release. This plugin uses the Elasticsearch bulk API to optimize its imports into Elasticsearch. These requests may experience either partial or total failures.

The following errors are retried infinitely:

- Network errors (inability to connect)
- 429 (Too many requests) and
- 503 (Service unavailable) errors

NOTE: 409 exceptions are no longer retried. Please set a higher `retry_on_conflict` value if you experience 409 exceptions. It is more performant for Elasticsearch to retry these exceptions than this plugin.

Batch Sizes

This plugin attempts to send batches of events as a single request. However, if a request exceeds 20MB we will break it up until multiple batch requests. If a single document exceeds 20MB it will be sent as a single request.

DNS Caching

This plugin uses the JVM to lookup DNS entries and is subject to the value of [`networkaddress.cache.ttl`](#), a global setting for the JVM.

As an example, to set your DNS TTL to 1 second you would set the `LS_JAVA_OPTS` environment variable to `-Dnetworkaddress.cache.ttl=1`.

Keep in mind that a connection with keepalive enabled will not reevaluate its DNS value while the keepalive is in effect.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
elasticsearch {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>absolute_healthcheck_path</code>	<code>boolean</code>	No	<code>false</code>
<code>action</code>	<code>string</code>	No	<code>"index"</code>
<code>cacert</code>	a valid filesystem path	No	
<code>codec</code>	<code>codec</code>	No	<code>"plain"</code>
<code>doc_as_upsert</code>	<code>boolean</code>	No	<code>false</code>
<code>document_id</code>	<code>string</code>	No	
<code>document_type</code>	<code>string</code>	No	

Setting	Input type	Required	Default value
<code>enable_metric</code>	boolean	No	true
<code>failure_type logging whitelist</code>	array	No	[]
<code>flush_size</code>	number	No	
<code>healthcheck_path</code>	string	No	"/"
<code>hosts</code>	uri	No	[//127.0.0.1]
<code>id</code>	string	No	
<code>idle_flush_time</code>	number	No	1
<code>index</code>	string	No	"logstash-%{+YYYY.MM.dd}"
<code>keystore</code>	a valid filesystem path	No	
<code>keystore_password</code>	password	No	
<code>manage_template</code>	boolean	No	true
<code>parameters</code>	hash	No	
<code>parent</code>	string	No	nil
<code>password</code>	password	No	
<code>path</code>	string	No	
<code>pipeline</code>	string	No	nil
<code>pool_max</code>	number	No	1000
<code>pool_max_per_route</code>	number	No	100
<code>proxy</code>	uri	No	
<code>resurrect_delay</code>	number	No	5
<code>retry_initial_interval</code>	number	No	2
<code>retry_max_interval</code>	number	No	64
<code>retry_on_conflict</code>	number	No	1
<code>routing</code>	string	No	
<code>script</code>	string	No	""
<code>script_lang</code>	string	No	"painless"
<code>script_type</code>	string , one of ["inline", "indexed", "file"]	No	["inline"]
<code>script_var_name</code>	string	No	"event"
<code>scripted_upsert</code>	boolean	No	false
<code>sniffing</code>	boolean	No	false
<code>sniffing_delay</code>	number	No	5
<code>ssl</code>	boolean	No	

Setting	Input type	Required	Default value
ssl_certificate_verification	boolean	No	true
template	a valid filesystem path	No	
template_name	string	No	"logstash"
template_overwrite	boolean	No	false
timeout	number	No	60
truststore	a valid filesystem path	No	
truststore_password	password	No	
upsert	string	No	""
user	string	No	
validate_after_inactivity	number	No	10000
version	string	No	
version_type	string , one of ["internal", "external", "external_gt", "external_gte", "force"]	No	
workers	<<,>>	No	1

Details

[absolute_healthcheck_path](#)

- Value type is [boolean](#)
- Default value is false

When a `healthcheck_path` config is provided, this additional flag can be used to specify whether the `healthcheck_path` is appended to the existing path (default) or is treated as the absolute URL path.

For example, if hosts url is "http://localhost:9200/es" and `healthcheck_path` is "/health", the health check url will be:

- with `absolute_healthcheck_path: true`: "http://localhost:9200/es/health"
- with `absolute_healthcheck_path: false`: "http://localhost:9200/health"

[action](#)

- Value type is [string](#)

- Default value is "index"

Protocol agnostic (i.e. non-http, non-java specific) configs go here Protocol agnostic methods
The Elasticsearch action to perform. Valid actions are:

- index: indexes a document (an event from Logstash).
- delete: deletes a document by id (An id is required for this action)
- create: indexes a document, fails if a document by that id already exists in the index.
- update: updates a document by id. Update has a special case where you can upsert — update a document if not already present. See the `upsert` option. NOTE: This does not work and is not supported in Elasticsearch 1.x. Please upgrade to ES 2.x or greater to use this feature with Logstash!
- A sprintf style string to change the action based on the content of the event. The value `%{[foo]}` would use the foo field for the action

For more details on actions, check out the [Elasticsearch bulk API documentation](#)

cacert

- Value type is [path](#)
- There is no default value for this setting.

The .cer or .pem file to validate the server's certificate

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

doc_as_upsert

- Value type is [boolean](#)
- Default value is `false`

Enable `doc_as_upsert` for update mode. Create a new document with source if `document_id` doesn't exist in Elasticsearch

document_id

- Value type is [string](#)
- There is no default value for this setting.

The document ID for the index. Useful for overwriting existing entries in Elasticsearch with the same ID.

`document_type`

- Value type is [string](#)
- There is no default value for this setting.

The document type to write events to. Generally you should try to write only similar events to the same *type*. String expansion `%{foo}` works here. Unless you set `document_type`, the event *type* will be used if it exists otherwise the document type will be assigned the value of *logs*

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`failure_type_logging_whitelist`

- Value type is [array](#)
- Default value is `[]`

Set the Elasticsearch errors in the whitelist that you don't want to log. A useful example is when you want to skip all 409 errors which are `document_already_exists_exception`.

`flush_size`

- Value type is [number](#)
- There is no default value for this setting.

This plugin uses the bulk index API for improved indexing performance. This setting defines the maximum sized bulk request Logstash will make. You may want to increase this to be in line with your pipeline's batch size. If you specify a number larger than the batch size of your pipeline it will have no effect, save for the case where a filter increases the size of an inflight batch by outputting events.

`healthcheck_path`

- Value type is [string](#)
- Default value is `"/"`

When a backend is marked down a HEAD request will be sent to this path in the background to see if it has come back again before it is once again eligible to service requests. If you have custom firewall rules you may need to change this NOTE: any query parameters present in the URL or `query_params` config option will be removed

`hosts`

- Value type is [uri](#)
- Default value is `[//127.0.0.1]`

Sets the host(s) of the remote instance. If given an array it will load balance requests across the hosts specified in the `hosts` parameter. Remember the `http` protocol uses the [http](#) address (eg. 9200, not 9300). `"127.0.0.1" ["127.0.0.1:9200", "127.0.0.2:9200"]`
`["http://127.0.0.1"] ["https://127.0.0.1:9200"]`
`["https://127.0.0.1:9200/mypath"]` (If using a proxy on a subpath) It is important to exclude [dedicated master nodes](#) from the `hosts` list to prevent LS from sending bulk requests to the master nodes. So this parameter should only reference either data or client nodes in Elasticsearch.

Any special characters present in the URLs here MUST be URL escaped! This means # should be put in as %23 for instance.

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
    stdout {  
        id => "my_plugin_id"  
    }  
}
```

`idle_flush_time`

- Value type is [number](#)
- Default value is 1

The amount of time since last flush before a flush is forced.

This setting helps ensure slow event rates don't get stuck in Logstash. For example, if your `flush_size` is 100, and you have received 10 events, and it has been more than `idle_flush_time` seconds since the last flush, Logstash will flush those 10 events automatically.

This helps keep both fast and slow log streams moving along in near-real-time.

index

- Value type is [string](#)
- Default value is "logstash-%{+YYYY.MM.dd}"

The index to write events to. This can be dynamic using the `%{foo}` syntax. The default value will partition your indices by day so you can more easily delete old data or only search specific date ranges. Indexes may not contain uppercase characters. For weekly indexes ISO 8601 format is recommended, eg. `logstash-%{+xxxx.ww}`. LS uses Joda to format the index pattern from event timestamp. Joda formats are defined [here](#).

keystore

- Value type is [path](#)
- There is no default value for this setting.

The keystore used to present a certificate to the server. It can be either .jks or .p12

keystore_password

- Value type is [password](#)
- There is no default value for this setting.

Set the truststore password

manage_template

- Value type is [boolean](#)
- Default value is `true`

From Logstash 1.3 onwards, a template is applied to Elasticsearch during Logstash's startup if one with the name `template_name` does not already exist. By default, the contents of this template is the default template for `logstash-%{+YYYY.MM.dd}` which always matches indices based on the pattern `logstash-*`. Should you require support for other index names, or would like to change the mappings in the template in general, a custom template can be specified by setting `template` to the path of a template file.

Setting `manage_template` to false disables this feature. If you require more control over template creation, (e.g. creating indices dynamically based on field names) you should set `manage_template` to false and use the REST API to apply your templates manually.

parameters

- Value type is [hash](#)
- There is no default value for this setting.

Pass a set of key value pairs as the URL query string. This query string is added to every host listed in the `hosts` configuration. If the `hosts` list contains urls that already have query strings, the one specified here will be appended.

`parent`

- Value type is [string](#)
- Default value is `nil`

For child documents, ID of the associated parent. This can be dynamic using the `%{foo}` syntax.

`password`

- Value type is [password](#)
- There is no default value for this setting.

Password to authenticate to a secure Elasticsearch cluster

`path`

- Value type is [string](#)
- There is no default value for this setting.

HTTP Path at which the Elasticsearch server lives. Use this if you must run Elasticsearch behind a proxy that remaps the root path for the Elasticsearch HTTP API lives. Note that if you use paths as components of URLs in the `hosts` field you may not also set this field. That will raise an error at startup

`pipeline`

- Value type is [string](#)
- Default value is `nil`

Set which ingest pipeline you wish to execute for an event. You can also use event dependent configuration here like `pipeline => "%{INGEST_PIPELINE}"`

`pool_max`

- Value type is [number](#)
- Default value is 1000

While the output tries to reuse connections efficiently we have a maximum. This sets the maximum number of open connections the output will create. Setting this too low may mean frequently closing / opening connections which is bad.

`pool_max_per_route`

- Value type is [number](#)
- Default value is 100

While the output tries to reuse connections efficiently we have a maximum per endpoint. This sets the maximum number of open connections per endpoint the output will create. Setting this too low may mean frequently closing / opening connections which is bad.

`proxy`

- Value type is [uri](#)
- There is no default value for this setting.

Set the address of a forward HTTP proxy. This used to accept hashes as arguments but now only accepts arguments of the URI type to prevent leaking credentials.

`resurrect_delay`

- Value type is [number](#)
- Default value is 5

How frequently, in seconds, to wait between resurrection attempts. Resurrection is the process by which backend endpoints marked *down* are checked to see if they have come back to life

`retry_initial_interval`

- Value type is [number](#)
- Default value is 2

Set initial interval in seconds between bulk retries. Doubled on each retry up to `retry_max_interval`

`retry_max_interval`

- Value type is [number](#)
- Default value is 64

Set max interval in seconds between bulk retries.

`retry_on_conflict`

- Value type is [number](#)
- Default value is 1

The number of times Elasticsearch should internally retry an update/upserted document See the [partial updates](#) for more info

routing

- Value type is [string](#)
- There is no default value for this setting.

A routing override to be applied to all processed events. This can be dynamic using the `%{foo}` syntax.

script

- Value type is [string](#)
- Default value is ""

Set script name for scripted update mode

script_lang

- Value type is [string](#)
- Default value is "painless"

Set the language of the used script. If not set, this defaults to painless in ES 5.0

script_type

- Value can be any of: `inline`, `indexed`, `file`
- Default value is `["inline"]`

Define the type of script referenced by "script" variable
`inline` : "script" contains inline script
`indexed` : "script" contains the name of script directly indexed in elasticsearch file
`file` : "script" contains the name of script stored in elasticseach's config directory

script_var_name

- Value type is [string](#)
- Default value is "event"

Set variable name passed to script (scripted update)

scripted_upsert

- Value type is [boolean](#)
- Default value is `false`

if enabled, script is in charge of creating non-existent document (scripted update)

sniffing

- Value type is [boolean](#)
- Default value is `false`

This setting asks Elasticsearch for the list of all cluster nodes and adds them to the hosts list.

Note: This will return ALL nodes with HTTP enabled (including master nodes!). If you use this with master nodes, you probably want to disable HTTP on them by setting `http.enabled` to `false` in their `elasticsearch.yml`. You can either use the `sniffing` option or manually enter multiple Elasticsearch hosts using the `hosts` parameter.

sniffing_delay

- Value type is [number](#)
- Default value is `5`

How long to wait, in seconds, between sniffing attempts

ssl

- Value type is [boolean](#)
- There is no default value for this setting.

Enable SSL/TLS secured communication to Elasticsearch cluster. Leaving this unspecified will use whatever scheme is specified in the URLs listed in `hosts`. If no explicit protocol is specified plain HTTP will be used. If SSL is explicitly disabled here the plugin will refuse to start if an HTTPS URL is given in `hosts`

ssl_certificate_verification

- Value type is [boolean](#)
- Default value is `true`

Option to validate the server's certificate. Disabling this severely compromises security. For more information on disabling certificate verification please read

https://www.cs.utexas.edu/~shmat/shmat_ccs12.pdf

template

- Value type is [path](#)
- There is no default value for this setting.

You can set the path to your own template here, if you so desire. If not set, the included template will be used.

`template_name`

- Value type is [string](#)
- Default value is "logstash"

This configuration option defines how the template is named inside Elasticsearch. Note that if you have used the template management features and subsequently change this, you will need to prune the old template manually, e.g.

```
curl -XDELETE <http://localhost:9200/_template/OldTemplateName?pretty>
```

where `OldTemplateName` is whatever the former setting was.

`template_overwrite`

- Value type is [boolean](#)
- Default value is `false`

The `template_overwrite` option will always overwrite the indicated template in Elasticsearch with either the one indicated by `template` or the included one. This option is set to `false` by default. If you always want to stay up to date with the template provided by Logstash, this option could be very useful to you. Likewise, if you have your own template file managed by puppet, for example, and you wanted to be able to update it regularly, this option could help there as well.

Please note that if you are using your own customized version of the Logstash template (`logstash`), setting this to `true` will make Logstash to overwrite the "logstash" template (i.e. removing all customized settings)

`timeout`

- Value type is [number](#)
- Default value is `60`

Set the timeout, in seconds, for network operations and requests sent Elasticsearch. If a timeout occurs, the request will be retried.

`truststore`

- Value type is [path](#)
- There is no default value for this setting.

The JKS truststore to validate the server's certificate. Use either `:truststore` or `:cacert`

`truststore_password`

- Value type is [password](#)
- There is no default value for this setting.

Set the truststore password

`upsert`

- Value type is [string](#)
- Default value is ""

Set upsert content for update mode.s Create a new document with this parameter as json string if `document_id` doesn't exists

`user`

- Value type is [string](#)
- There is no default value for this setting.

Username to authenticate to a secure Elasticsearch cluster

`validate_after_inactivity`

- Value type is [number](#)
- Default value is 10000

How long to wait before checking if the connection is stale before executing a request on a connection using keepalive. You may want to set this lower, if you get connection errors regularly Quoting the Apache commons docs (this client is based Apache Commons): *Defines period of inactivity in milliseconds after which persistent connections must be re-validated prior to being leased to the consumer. Non-positive value passed to this method disables connection validation. This check helps detect connections that have become stale (half-closed) while kept inactive in the pool. See [these docs for more info](#)*

`version`

- Value type is [string](#)
- There is no default value for this setting.

The version to use for indexing. Use sprintf syntax like `%{my_version}` to use a field value here. See <https://www.elastic.co/blog/elasticsearch-versioning-support>.

`version_type`

- Value can be any of: `internal`, `external`, `external_gt`, `external_gte`, `force`
- There is no default value for this setting.

The `version_type` to use for indexing. See <https://www.elastic.co/blog/elasticsearch-versioning-support>. See also https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-index_.html#_version_types

workers

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

email

- Version: 4.0.3
- Released on: 2016-08-04
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
bin/logstash-plugin install logstash-output-email.

Send email when an output is received. Alternatively, you may include or exclude the email output execution using conditionals.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
email {
    to => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
address	string	No	"localhost"
attachments	array	No	[]
authentication	string	No	
body	string	No	""
cc	string	No	
codec	codec	No	"plain"
contenttype	string	No	"text/html; charset=UTF-8"
debug	boolean	No	false
domain	string	No	"localhost"
enable_metric	boolean	No	true
from	string	No	"logstash.alert@nowhere.com"
htmlbody	string	No	""
id	string	No	
password	string	No	

Setting	Input type	Required	Default value
port	number	No	25
replyto	string	No	
subject	string	No	""
to	string	Yes	
use_tls	boolean	No	false
username	string	No	
via	string	No	"smtp"
workers	<>	No	1

Details

address

- Value type is [string](#)
- Default value is "localhost"

The address used to connect to the mail server

attachments

- Value type is [array](#)
- Default value is []

Attachments - specify the name(s) and location(s) of the files.

authentication

- Value type is [string](#)
- There is no default value for this setting.

Authentication method used when identifying with the server

body

- Value type is [string](#)
- Default value is ""

Body for the email - plain text only.

`cc`

- Value type is [string](#)
- There is no default value for this setting.

The fully-qualified email address(es) to include as cc: address(es).

This field also accepts a comma-separated string of addresses, for example: "me@host.com, you@host.com"

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

`contenttype`

- Value type is [string](#)
- Default value is "text/html; charset=UTF-8"

`contenttype` : for multipart messages, set the content-type and/or charset of the HTML part.
NOTE: this may not be functional (KH)

`debug`

- Value type is [boolean](#)
- Default value is `false`

Run the mail relay in debug mode

`domain`

- Value type is [string](#)
- Default value is "localhost"

Domain used to send the email messages

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

from

- Value type is [string](#)
- Default value is "logstash.alert@nowhere.com"

The fully-qualified email address for the From: field in the email.

htmlbody

- Value type is [string](#)
- Default value is ""

HTML Body for the email, which may contain HTML markup.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique ID to the plugin instance, this ID is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

### *password*

- Value type is [string](#)
- There is no default value for this setting.

Password to authenticate with the server

### *port*

- Value type is [number](#)
- Default value is 25

Port used to communicate with the mail server

### *replyto*

- Value type is [string](#)
- There is no default value for this setting.

The fully qualified email address for the Reply-To: field.

*subject*

- Value type is [string](#)
- Default value is ""

Subject: for the email.

*to*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The fully-qualified email address to send the email to.

This field also accepts a comma-separated string of addresses, for example: "me@host.com, you@host.com"

You can also use dynamic fields from the event with the `%{fieldname}` syntax.

*use\_tls*

- Value type is [boolean](#)
- Default value is `false`

Enables TLS when communicating with the server

*username*

- Value type is [string](#)
- There is no default value for this setting.

Username to authenticate with the server

*via*

- Value type is [string](#)
- Default value is "smtp"

How Logstash should send the email, either via SMTP or by invoking sendmail.

*workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

## exec

- Version: 3.1.0
- Released on: 2016-12-06
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
`bin/logstash-plugin install logstash-output-exec`.

The exec output will run a command for each event received. Ruby's `system()` function will be used, i.e. the command string will be passed to a shell. You can use `%{name}` and other dynamic strings in the command to pass select fields from the event to the child process. Example:

```
output {
 if [type] == "abuse" {
 exec {
 command => "iptables -A INPUT -s %{clientip} -j DROP"
 }
 }
}
```

If you want it non-blocking you should use `&` or `detach` or other such techniques. There is no timeout for the commands being run so misbehaving commands could otherwise stall the Logstash pipeline indefinitely.

Exercise great caution with `%{name}` field placeholders. The contents of the field will be included verbatim without any sanitization, i.e. any shell metacharacters from the field values will be passed straight to the shell.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
exec {
 command => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	<a href="#">codec</a>	No	"plain"
command	<a href="#">string</a>	Yes	
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	
quiet	<a href="#">boolean</a>	No	false
workers	<<, >>	No	1

## Details

### *codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### *command*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Command line to execute via subprocess. Use `detach` or `screen` to make it non blocking. This value can include `%{name}` and other dynamic strings.

### *enable\_metric*

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### *id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique ID to the plugin instance, this ID is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

quiet

- Value type is [boolean](#)
- Default value is `false`

display the result of the command to the terminal

workers

- Value type is [string](#)
- Default value is `1`

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

file

- Version: 4.0.1
- Released on: 2016-09-15
- [Changelog](#)

This output writes events to files on disk. You can use fields from the event as parts of the filename and/or path.

By default, this output writes one event per line in **json** format. You can customise the line format using the `line` codec like

```
output {
  file {
    path => ...
    codec => line { format => "custom format: %{message}" }
  }
}
```

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
file {
  path => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	codec	No	"plain"
create_if_deleted	boolean	No	true
dir_mode	number	No	-1
enable_metric	boolean	No	true
file_mode	number	No	-1
filename_failure	string	No	"_filepath_failures"
flush_interval	number	No	2
gzip	boolean	No	false
id	string	No	

Setting	Input type	Required	Default value
path	string	Yes	
workers	<<,>>	No	1

Details

[codec](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

[create_if_deleted](#)

- Value type is [boolean](#)
- Default value is `true`

If the configured file is deleted, but an event is handled by the plugin, the plugin will recreate the file. Default \Rightarrow true

[dir_mode](#)

- Value type is [number](#)
- Default value is -1

Dir access mode to use. Note that due to the bug in jruby system umask is ignored on linux:

<https://github.com/jruby/jruby/issues/3426> Setting it to -1 uses default OS value. Example:

`"dir_mode" => 0750`

[enable_metric](#)

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

[file_mode](#)

- Value type is [number](#)
- Default value is -1

File access mode to use. Note that due to the bug in jruby system umask is ignored on linux:
<https://github.com/jruby/jruby/issues/3426> Setting it to -1 uses default OS value. Example:
"file_mode" => 0640

filename_failure

- Value type is [string](#)
- Default value is "_filepath_failures"

If the generated path is invalid, the events will be saved into this file and inside the defined path.

flush_interval

- Value type is [number](#)
- Default value is 2

Flush interval (in seconds) for flushing writes to log files. 0 will flush on every message.

gzip

- Value type is [boolean](#)
- Default value is false

Gzip the output stream before writing to disk.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
  stdout {
    id => "my_plugin_id"
  }
}
```

path

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The path to the file to write. Event fields can be used here, like

`/var/log/logstash/%{host}/%{application}` One may also utilize the path option for date-

based log rotation via the joda time format. This will use the event timestamp. E.g.: path => "./test-%{+YYYY-MM-dd}.txt" to create ./test-2013-05-29.txt

If you use an absolute path you cannot start with a dynamic string. E.g: %{myfield}/, /test-%{myfield}/ are not valid paths

workers

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

ganglia

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
`bin/logstash-plugin install logstash-output-ganglia`.

This output allows you to pull metrics from your logs and ship them to ganglia's gmond. This is heavily based on the graphite output.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
ganglia {
    metric => ...
    value => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	codec	No	"plain"
enable_metric	boolean	No	true
group	string	No	""
host	string	No	"localhost"
id	string	No	
lifetime	number	No	300
max_interval	number	No	60
metric	string	Yes	
metric_type	string , one of ["string", "int8", "uint8", "int16", "uint16", "int32", "uint32", "float", "double"]	No	"uint8"
port	number	No	8649
slope	string , one of ["zero", "positive", "negative", "both", "unspecified"]	No	"both"

Setting	Input type	Required	Default value
units	string	No	""
value	string	Yes	
workers	<>, >>	No	1

Details

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

enable_metric

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

group

- Value type is [string](#)
- Default value is ""

Metric group

host

- Value type is [string](#)
- Default value is "localhost"

The address of the ganglia server.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

#### *lifetime*

- Value type is [number](#)
- Default value is 300

Lifetime in seconds of this metric

#### *max\_interval*

- Value type is [number](#)
- Default value is 60

Maximum time in seconds between gmetric calls for this metric.

#### *metric*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The metric to use. This supports dynamic strings like `%{host}`

#### *metric\_type*

- Value can be any of: `string, int8, uint8, int16, uint16, int32, uint32, float, double`
- Default value is "uint8"

The type of value for this metric.

#### *port*

- Value type is [number](#)
- Default value is 8649

The port to connect on your ganglia server.

#### *slope*

- Value can be any of: `zero, positive, negative, both, unspecified`
- Default value is "both"

Metric slope, represents metric behavior

*units*

- Value type is [string](#)
- Default value is ""

Gmetric units for metric, such as "kb/sec" or "ms" or whatever unit this metric uses.

*value*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The value to use. This supports dynamic strings like %{bytes} It will be coerced to a floating point value. Values which cannot be coerced will zero (0)

*workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

## gelf

- Version: 3.1.2
- Released on: 2016-11-14
- [Changelog](#)

This plugin does not ship with Logstash by default, but it is easy to install by running  
`bin/logstash-plugin install logstash-output-gelf`.

This output generates messages in GELF format. This is most useful if you want to use Logstash to output events to Graylog2.

More information at [The Graylog2 GELF specs page](#)

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
gelf {
 host => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
chunksize	<a href="#">number</a>	No	1420
codec	<a href="#">codec</a>	No	"plain"
custom_fields	<a href="#">hash</a>	No	{ }
enable_metric	<a href="#">boolean</a>	No	true
full_message	<a href="#">string</a>	No	"%{message}"
host	<a href="#">string</a>	Yes	
id	<a href="#">string</a>	No	
ignore_metadata	<a href="#">array</a>	No	["@timestamp", "@version", "severity", "host", "source_host", "source_path", "short_message"]
level	<a href="#">array</a>	No	[ "%{severity}" , "INFO" ]
port	<a href="#">number</a>	No	12201

<b>Setting</b>	<b>Input type</b>	<b>Required</b>	<b>Default value</b>
sender	<a href="#">string</a>	No	"%{host}"
ship_metadata	<a href="#">boolean</a>	No	true
ship_tags	<a href="#">boolean</a>	No	true
short_message	<a href="#">string</a>	No	"short_message"
workers	<>	No	1

## Details

### *chunksize*

- Value type is [number](#)
- Default value is 1420

The GELF chunksize. You usually don't need to change this.

### *codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### *custom\_fields*

- Value type is [hash](#)
- Default value is {}

The GELF custom field mappings. GELF supports arbitrary attributes as custom fields. This exposes that. Exclude the \_ portion of the field name e.g. `custom_fields => ['foo_field', 'some_value']` sets `_foo_field=some_value`.

### *enable\_metric*

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `full_message`

- Value type is [string](#)
- Default value is "`%{message}`"

The GELF full message. Dynamic values like `%{foo}` are permitted here.

### `host`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Graylog2 server IP address or hostname.

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

`ignore_metadata`

- Value type is [array](#)
- Default value is `["@timestamp", "@version", "severity", "host", "source_host", "source_path", "short_message"]`

Ignore these fields when `ship_metadata` is set. Typically this lists the fields used in dynamic values for GELF fields.

`level`

- Value type is [array](#)
- Default value is `["%{severity}", "INFO"]`

The GELF message level. Dynamic values like `%{level}` are permitted here; useful if you want to parse the *log level* from an event and use that as the GELF level/severity.

Values here can be integers [0..7] inclusive or any of "debug", "info", "warn", "error", "fatal" (case insensitive). Single-character versions of these are also valid, "d", "i", "w", "e", "f", "u". The

following additional severity_labels from Logstash's syslog_pri filter are accepted: "emergency", "alert", "critical", "warning", "notice", and "informational".

port

- Value type is [number](#)
- Default value is 12201

Graylog2 server port number.

sender

- Value type is [string](#)
- Default value is "%{host}"

Allow overriding of the GELF `sender` field. This is useful if you want to use something other than the event's source host as the "sender" of an event. A common case for this is using the application name instead of the hostname.

ship_metadata

- Value type is [boolean](#)
- Default value is `true`

Should Logstash ship metadata within event object? This will cause Logstash to ship any fields in the event (such as those created by grok) in the GELF messages. These will be sent as underscored "additional fields".

ship_tags

- Value type is [boolean](#)
- Default value is `true`

Ship tags within events. This will cause Logstash to ship the tags of an event as the field `_tags`.

short_message

- Value type is [string](#)
- Default value is "short_message"

The GELF short message field name. If the field does not exist or is empty, the event message is taken instead.

workers

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

google_bigquery

- Version: 3.0.1
- Released on: 2016-09-13
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
bin/logstash-plugin install logstash-output-google_bigquery.

Author: Rodrigo De Castro <rdc@google.com> Date: 2013-09-20

Copyright 2013 Google Inc.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. Summary: plugin to upload log events to Google BigQuery (BQ), rolling files based on the date pattern provided as a configuration setting. Events are written to files locally and, once file is closed, this plugin uploads it to the configured BigQuery dataset.

VERY IMPORTANT: 1 - To make good use of BigQuery, your log events should be parsed and structured. Consider using grok to parse your events into fields that can be uploaded to BQ. 2 - You must configure your plugin so it gets events with the same structure, so the BigQuery schema suits them. In case you want to upload log events with different structures, you can utilize multiple configuration blocks, separating different log events with Logstash conditionals. More details on Logstash conditionals can be found here:

<http://logstash.net/docs/1.2.1/configuration#conditionals>

For more info on Google BigQuery, please go to: <https://developers.google.com/bigquery/>

In order to use this plugin, a Google service account must be used. For more information, please refer to: https://developers.google.com/storage/docs/authentication#service_accounts

Recommendations: a - Experiment with the settings depending on how much log data you generate, your needs to see "fresh" data, and how much data you could lose in the event of crash. For instance, if you want to see recent data in BQ quickly, you could configure the plugin to upload data every minute or so (provided you have enough log events to justify that). Note also, that if uploads are too frequent, there is no guarantee that they will be imported in the same order, so later data may be available before earlier data. b - BigQuery charges for storage and for queries, depending on how much data it reads to perform a query. These are other aspects to

consider when considering the date pattern which will be used to create new tables and also how to compose the queries when using BQ. For more info on BigQuery Pricing, please access:
<https://developers.google.com/bigquery/pricing>

USAGE: This is an example of logstash config:

```
output { google_bigquery { project_id => "folkloric-guru-278" (required) dataset => "logs" (required) csv_schema => "path:STRING,status:INTEGER,score:FLOAT" (required*) key_path => "/path/to/privatekey.p12" (required) key_password => "notasecret" (optional) service_account => "1234@developer.gserviceaccount.com" (required) temp_directory => "/tmp/logstash-bq" (optional) temp_file_prefix => "logstash_bq" (optional) date_pattern => "%Y-%m-%dT%H:00" (optional) flush_interval_secs => 2 (optional) uploader_interval_secs => 60 (optional) deleter_interval_secs => 60 (optional) } }
```

- Specify either a csv_schema or a json_schema.

Improvements TODO list: - Refactor common code between Google BQ and GCS plugins. - Turn Google API code into a Plugin Mixin (like AwsConfig). - There's no recover method, so if logstash/plugin crashes, files may not be uploaded to BQ.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
google_bigquery {
  dataset => ...
  key_path => ...
  project_id => ...
  service_account => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	codec	No	"plain"
csv_schema	string	No	nil
dataset	string	Yes	
date_pattern	string	No	"%Y-%m-%dT%H:00"
deleter_interval_secs	number	No	60
enable_metric	boolean	No	true

Setting	Input type	Required	Default value
flush_interval_secs	number	No	2
id	string	No	
ignore_unknown_values	boolean	No	false
json_schema	hash	No	nil
key_password	string	No	"notasecret"
key_path	string	Yes	
project_id	string	Yes	
service_account	string	Yes	
table_prefix	string	No	"logstash"
table_separator	string	No	"_"
temp_directory	string	No	""
temp_file_prefix	string	No	"logstash_bq"
uploader_interval_secs	number	No	60
workers	<>	No	1

Details

[codec](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

[csv_schema](#)

- Value type is [string](#)
- Default value is nil

Schema for log data. It must follow this format: <field1-name>:<field1-type>,<field2-name>:<field2-type>,... Example: path:STRING,status:INTEGER,score:FLOAT

[dataset](#)

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

BigQuery dataset to which these events will be added to.

date_pattern

- Value type is [string](#)
- Default value is "%Y-%m-%dT%H:00"

Time pattern for BigQuery table, defaults to hourly tables. Must Time.strptime patterns:
www.ruby-doc.org/core-2.0/Time.html#method-i-strftime

deleter_interval_secs

- Value type is [number](#)
- Default value is 60

Deleter interval when checking if upload jobs are done for file deletion. This only affects how long files are on the hard disk after the job is done.

enable_metric

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

flush_interval_secs

- Value type is [number](#)
- Default value is 2

Flush interval in seconds for flushing writes to log files. 0 will flush on every message.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

*ignore\_unknown\_values*

- Value type is [boolean](#)
- Default value is `false`

Indicates if BigQuery should allow extra values that are not represented in the table schema. If true, the extra values are ignored. If false, records with extra columns are treated as bad records, and if there are too many bad records, an invalid error is returned in the job result. The default value is false.

*json\_schema*

- Value type is [hash](#)
- Default value is `nil`

Schema for log data, as a hash. Example: `json_schema => { fields => [ { name => "timestamp" type => "TIMESTAMP" }, { name => "host" type => "STRING" }, { name => "message" type => "STRING" } ] }`

*key\_password*

- Value type is [string](#)
- Default value is `"notasecret"`

Private key password for service account private key.

*key\_path*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Path to private key file for Google Service Account.

*project\_id*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Google Cloud Project ID (number, not Project Name!).

*service\_account*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Service account to access Google APIs.

*table\_prefix*

- Value type is [string](#)
- Default value is "logstash"

BigQuery table ID prefix to be used when creating new tables for log data. Table name will be <table\_prefix><table\_separator><date>

*table\_separator*

- Value type is [string](#)
- Default value is "\_"

BigQuery table separator to be added between the table\_prefix and the date suffix.

*temp\_directory*

- Value type is [string](#)
- Default value is ""

Directory where temporary files are stored. Defaults to /tmp/logstash-bq-<random-suffix>

*temp\_file\_prefix*

- Value type is [string](#)
- Default value is "logstash\_bq"

Temporary local file prefix. Log file will follow the format: <prefix>\_hostname\_date.part?.log

*uploader\_interval\_secs*

- Value type is [number](#)
- Default value is 60

Uploader interval when uploading new files to BigQuery. Adjust time based on your time pattern (for example, for hourly files, this interval can be around one hour).

*workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here



## google\_cloud\_storage

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-output-google_cloud_storage`.

Author: Rodrigo De Castro <[rdc@google.com](mailto:rdc@google.com)> Date: 2013-09-20

Copyright 2013 Google Inc.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. Summary: plugin to upload log events to Google Cloud Storage (GCS), rolling files based on the date pattern provided as a configuration setting. Events are written to files locally and, once file is closed, this plugin uploads it to the configured bucket.

For more info on Google Cloud Storage, please go to: <https://cloud.google.com/products/cloud-storage>

In order to use this plugin, a Google service account must be used. For more information, please refer to: [https://developers.google.com/storage/docs/authentication#service\\_accounts](https://developers.google.com/storage/docs/authentication#service_accounts)

Recommendation: experiment with the settings depending on how much log data you generate, so the uploader can keep up with the generated logs. Using gzip output can be a good option to reduce network traffic when uploading the log files and in terms of storage costs as well.

USAGE: This is an example of logstash config:

```
output { google_cloud_storage { bucket => "my_bucket" (required) key_path => "/path/to/privatekey.p12" (required) key_password => "notasecret" (optional) service_account => "1234@developer.gserviceaccount.com" (required) temp_directory => "/tmp/logstash-gcs" (optional) log_file_prefix => "logstash_gcs" (optional) max_file_size_kbytes => 1024 (optional) output_format => "plain" (optional) date_pattern => "%Y-%m-%dT%H:00" (optional) flush_interval_secs => 2 (optional) gzip => false (optional) uploader_interval_secs => 60 (optional) } }
```

Improvements TODO list: - Support logstash event variables to determine filename. - Turn Google API code into a Plugin Mixin (like AwsConfig). - There's no recover method, so if logstash/plugin crashes, files may not be uploaded to GCS. - Allow user to configure file name. -

Allow parallel uploads for heavier loads (+ connection configuration if exposed by Ruby API client)

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
google_cloud_storage {
 bucket => ...
 key_path => ...
 service_account => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">bucket</a>	<a href="#">string</a>	Yes	
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">date pattern</a>	<a href="#">string</a>	No	"%Y-%m-%dT%H:00"
<a href="#">flush interval secs</a>	<a href="#">number</a>	No	2
<a href="#">gzip</a>	<a href="#">boolean</a>	No	false
<a href="#">key password</a>	<a href="#">string</a>	No	"notasecret"
<a href="#">key path</a>	<a href="#">string</a>	Yes	
<a href="#">log file prefix</a>	<a href="#">string</a>	No	"logstash_gcs"
<a href="#">max file size kbytes</a>	<a href="#">number</a>	No	10000
<a href="#">output format</a>	<a href="#">string</a> , one of ["json", "plain"]	No	"plain"
<a href="#">service account</a>	<a href="#">string</a>	Yes	
<a href="#">temp directory</a>	<a href="#">string</a>	No	""
<a href="#">uploader interval secs</a>	<a href="#">number</a>	No	60
<a href="#">workers</a>	<a href="#">number</a>	No	1

## Details

*bucket*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

GCS bucket name, without "gs://" or any other prefix.

*codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

*date\_pattern*

- Value type is [string](#)
- Default value is "%Y-%m-%dT%H:00"

Time pattern for log file, defaults to hourly files. Must Time.strptime patterns: [www.ruby-doc.org/core-2.0/Time.html#method-i-strftime](http://www.ruby-doc.org/core-2.0/Time.html#method-i-strftime)

*flush\_interval\_secs*

- Value type is [number](#)
- Default value is 2

Flush interval in seconds for flushing writes to log files. 0 will flush on every message.

*gzip*

- Value type is [boolean](#)
- Default value is false

Gzip output stream when writing events to log files.

*key\_password*

- Value type is [string](#)
- Default value is "notasecret"

GCS private key password.

*key\_path*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

GCS path to private key file.

*log\_file\_prefix*

- Value type is [string](#)
- Default value is "logstash\_gcs"

Log file prefix. Log file will follow the format: <prefix>\_hostname\_date<.part?>.log

*max\_file\_size\_kbytes*

- Value type is [number](#)
- Default value is 10000

Sets max file size in kbytes. 0 disable max file check.

*output\_format*

- Value can be any of: json, plain
- Default value is "plain"

The event format you want to store in files. Defaults to plain text.

*service\_account*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

GCS service account.

*temp\_directory*

- Value type is [string](#)
- Default value is ""

Directory where temporary files are stored. Defaults to /tmp/logstash-gcs-<random-suffix>

*uploader\_interval\_secs*

- Value type is [number](#)

- Default value is 60

Uploader interval when uploading new files to GCS. Adjust time based on your time pattern (for example, for hourly files, this interval can be around one hour).

*workers*

- Value type is [number](#)
- Default value is 1

The number of workers to use for this output. Note that this setting may not be useful for all outputs.

## graphite

- Version: 3.1.1
- Released on: 2016-07-14
- [Changelog](#)

This output allows you to pull metrics from your logs and ship them to Graphite. Graphite is an open source tool for storing and graphing metrics.

An example use case: Some applications emit aggregated stats in the logs every 10 seconds. Using the grok filter and this output, it is possible to capture the metric values from the logs and emit them to Graphite.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
graphite {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">exclude_metrics</a>	<a href="#">array</a>	No	[ "%{[^}]+}" ]
<a href="#">fields_are_metrics</a>	<a href="#">boolean</a>	No	false
<a href="#">host</a>	<a href="#">string</a>	No	"localhost"
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">include_metrics</a>	<a href="#">array</a>	No	[ ".*" ]
<a href="#">metrics</a>	<a href="#">hash</a>	No	{ }
<a href="#">metrics_format</a>	<a href="#">string</a>	No	"*"
<a href="#">nested_object_separator</a>	<a href="#">string</a>	No	". "
<a href="#">port</a>	<a href="#">number</a>	No	2003
<a href="#">reconnect_interval</a>	<a href="#">number</a>	No	2
<a href="#">resend_on_failure</a>	<a href="#">boolean</a>	No	false

Setting	Input type	Required	Default value
<a href="#">timestamp_field</a>	<a href="#">string</a>	No	"@timestamp"
<a href="#">workers</a>	<>	No	1

## Details

### `codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `exclude_metrics`

- Value type is [array](#)
- Default value is `[ "%{ [^} ]+" ]`

### `fields_are_metrics`

- Value type is [boolean](#)
- Default value is `false`

An array indicating that these event fields should be treated as metrics and will be sent verbatim to Graphite. You may use either `fields_are_metrics` or `metrics`, but not both.

### `host`

- Value type is [string](#)
- Default value is "localhost"

The hostname or IP address of the Graphite server.

### *id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
include_metrics
```

- Value type is [array](#)
- Default value is `[ "./*"]`

Include only regex matched metric names.

### *metrics*

- Value type is [hash](#)
- Default value is `{ }`

for metric names and also for values. This is a hash field with key being the metric name, value being the metric value. Example:

```
metrics => { "%{host}/uptime" => "%{uptime_1m}" }
```

The value will be coerced to a floating point value. Values which cannot be coerced will be set to zero (0). You may use either `metrics` or `fields_are_metrics`, but not both.

### *metrics\_format*

- Value type is [string](#)
- Default value is `"*"`

Defines the format of the metric string. The placeholder `*` will be replaced with the name of the actual metric.

```
metrics_format => "foo.bar.*.sum"
```

NOTE: If no `metrics_format` is defined, the name of the metric will be used as fallback.

*nested\_object\_separator*

- Value type is [string](#)
- Default value is ". "

When hashes are passed in as values they are broken out into a dotted notation For instance if you configure this plugin with # [source,ruby] metrics ⇒ "mymetrics"

and "mymetrics" is a nested hash of  $\{a \Rightarrow 1, b \Rightarrow \{c \Rightarrow 2\}\}$  this plugin will generate two metrics:  $a \Rightarrow 1$ , and  $b.c \Rightarrow 2$ . If you've specified a *metrics\_format* it will respect that, but you still may want control over the separator within these nested key names. This config setting changes the separator from the . default.

*port*

- Value type is [number](#)
- Default value is 2003

The port to connect to on the Graphite server.

*reconnect\_interval*

- Value type is [number](#)
- Default value is 2

Interval between reconnect attempts to Carbon.

*resend\_on\_failure*

- Value type is [boolean](#)
- Default value is false

Should metrics be resent on failure?

*timestamp\_field*

- Value type is [string](#)
- Default value is "@timestamp"

Use this field for the timestamp instead of @*timestamp* which is the default. Useful when backfilling or just getting more accurate data into graphite since you probably have a cache layer in front of Logstash.

*workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

## graphtastic

- Version: 3.0.0
- Released on: 2016-09-09
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
 bin/logstash-plugin install logstash-output-graphtastic.

A plugin for a newly developed Java/Spring Metrics application I didn't really want to code this project but I couldn't find a respectable alternative that would also run on any Windows machine - which is the problem and why I am not going with Graphite and statsd. This application provides multiple integration options so as to make its use under your network requirements possible. This includes a REST option that is always enabled for your use in case you want to write a small script to send the occasional metric data.

Find GraphTastic here : <https://github.com/NickPadilla/GraphTastic>

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
graphtastic {
}
```

Available configuration options:

Setting	Input type	Required	Default value
batch_number	<a href="#">number</a>	No	60
codec	<a href="#">codec</a>	No	"plain"
context	<a href="#">string</a>	No	"graphtastic"
enable_metric	<a href="#">boolean</a>	No	true
error_file	<a href="#">string</a>	No	""
host	<a href="#">string</a>	No	"127.0.0.1"
id	<a href="#">string</a>	No	
integration	<a href="#">string</a> , one of ["udp", "tcp", "rmi", "rest"]	No	"udp"
metrics	<a href="#">hash</a>	No	{ }
port	<a href="#">number</a>	No	

Setting	Input type	Required	Default value
retries	<a href="#">number</a>	No	1
workers	<<,>>	No	1

## Details

### *batch\_number*

- Value type is [number](#)
- Default value is 60

the number of metrics to send to GraphTastic at one time. 60 seems to be the perfect amount for UDP, with default packet size.

### *codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### *context*

- Value type is [string](#)
- Default value is "graphtastic"

if using rest as your end point you need to also provide the application url it defaults to localhost/graphtastic. You can customize the application url by changing the name of the .war file. There are other ways to change the application context, but they vary depending on the Application Server in use. Please consult your application server documentation for more on application contexts.

### *enable\_metric*

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### *error\_file*

- Value type is [string](#)
- Default value is ""

setting allows you to specify where we save errored transactions this makes the most sense at this point - will need to decide on how we reintegrate these error metrics NOT IMPLEMENTED!

### *host*

- Value type is [string](#)
- Default value is "127.0.0.1"

host for the graphmatic server - defaults to 127.0.0.1

### *id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique ID to the plugin instance, this ID is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

integration

- Value can be any of: udp, tcp, rmi, rest
- Default value is "udp"

options are udp(fastest - default) - rmi(faster) - rest(fast) - tcp(don't use TCP yet - some problems - errors out on linux)

metrics

- Value type is [hash](#)
- Default value is {}

metrics hash - you will provide a name for your metric and the metric data as key value pairs. so for example:

```
metrics => { "Response" => "%{response}" }
```

example for the logstash config

```
metrics => [ "Response", "%{response}" ]
```

you can also use the dynamic fields for the key value as well as the actual value

port

- Value type is [number](#)
- There is no default value for this setting.

port for the graphtastic instance - defaults to 1199 for RMI, 1299 for TCP, 1399 for UDP, and 8080 for REST

retries

- Value type is [number](#)
- Default value is 1

number of attempted retry after send error - currently only way to integrate errored transactions - should try and save to a file or later consumption either by graphtastic utility or by this program after connectivity is ensured to be established.

workers

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

hipchat

- Version: 4.0.2
- Released on: 2016-07-14
- [Changelog](#)

This plugin does not ship with Logstash by default, but it is easy to install by running
`bin/logstash-plugin install logstash-output-hipchat`.

This output allows you to write events to [HipChat](#).

Make sure your API token have the appropriate permissions and support sending messages.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
hipchat {
    room_id => ...
    token => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	codec	No	"plain"
color	string	No	"yellow"
enable_metric	boolean	No	true
format	string	No	"%{message}"
from	string	No	"logstash"
host	string	No	
id	string	No	
message_format	string , one of ["html", "text"]	No	"html"
room_id	string	Yes	
token	string	Yes	
trigger_notify	boolean	No	false
workers	<<, >>	No	1

Details

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

`color`

- Value type is [string](#)
- Default value is "yellow"

Background color for message. HipChat currently supports one of "yellow", "red", "green", "purple", "gray", or "random". (default: yellow), support fieldref

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`format`

- Value type is [string](#)
- Default value is "`%{message}`"

Message format to send, event tokens are usable here.

`from`

- Value type is [string](#)
- Default value is "logstash"

The name the message will appear be sent from, you can use fieldref

`host`

- Value type is [string](#)
- There is no default value for this setting.

HipChat host to use

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

*message\_format*

- Value can be any of: `html`, `text`
- Default value is "`html`"

Specify Message Format

*room\_id*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The ID or name of the room, support fieldref

*token*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The HipChat authentication token.

*trigger\_notify*

- Value type is [boolean](#)
- Default value is `false`

Whether or not this message should trigger a notification for people in the room.

*workers*

- Value type is [string](#)

- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

## http

- Version: 3.1.1
- Released on: 2016-07-14
- [Changelog](#)

We count outstanding requests with this queue. This queue tracks the requests to create backpressure. When this queue is empty no new requests may be sent, tokens must be added back by the client on success.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
http {
 http_method => ...
 url => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">automatic_retries</a>	<a href="#">number</a>	No	1
<a href="#">cacert</a>	a valid filesystem path	No	
<a href="#">client_cert</a>	a valid filesystem path	No	
<a href="#">client_key</a>	a valid filesystem path	No	
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">connect_timeout</a>	<a href="#">number</a>	No	10
<a href="#">content_type</a>	<a href="#">string</a>	No	
<a href="#">cookies</a>	<a href="#">boolean</a>	No	true
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">follow_redirects</a>	<a href="#">boolean</a>	No	true
<a href="#">format</a>	<a href="#">string</a> , one of ["json", "form", "message"]	No	"json"
<a href="#">headers</a>	<a href="#">hash</a>	No	

Setting	Input type	Required	Default value
<a href="#">http_method</a>	<a href="#">string</a> , one of ["put", "post", "patch", "delete", "get", "head"]	Yes	
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">keepalive</a>	<a href="#">boolean</a>	No	true
<a href="#">keystore</a>	a valid filesystem path	No	
<a href="#">keystore_password</a>	<a href="#">password</a>	No	
<a href="#">keystore_type</a>	<a href="#">string</a>	No	"JKS"
<a href="#">mapping</a>	<a href="#">hash</a>	No	
<a href="#">message</a>	<a href="#">string</a>	No	
<a href="#">pool_max</a>	<a href="#">number</a>	No	50
<a href="#">pool_max_per_route</a>	<a href="#">number</a>	No	25
<a href="#">proxy</a>	<>	No	
<a href="#">request_timeout</a>	<a href="#">number</a>	No	60
<a href="#">retry_non_idempotent</a>	<a href="#">boolean</a>	No	false
<a href="#">socket_timeout</a>	<a href="#">number</a>	No	10
<a href="#">ssl_certificate_validation</a>	<a href="#">boolean</a>	No	true
<a href="#">truststore</a>	a valid filesystem path	No	
<a href="#">truststore_password</a>	<a href="#">password</a>	No	
<a href="#">truststore_type</a>	<a href="#">string</a>	No	"JKS"
<a href="#">url</a>	<a href="#">string</a>	Yes	
<a href="#">validate_after_inactivity</a>	<a href="#">number</a>	No	200
<a href="#">workers</a>	<>	No	1

## Details

[automatic\\_retries](#)

- Value type is [number](#)
- Default value is 1

How many times should the client retry a failing URL. We highly recommend NOT setting this value to zero if keepalive is enabled. Some servers incorrectly end keepalives early requiring a retry! Note: if `retry_non_idempotent` is set only GET, HEAD, PUT, DELETE, OPTIONS, and TRACE requests will be retried.

*cacert*

- Value type is [path](#)
- There is no default value for this setting.

If you need to use a custom X.509 CA (.pem certs) specify the path to that here

*client\_cert*

- Value type is [path](#)
- There is no default value for this setting.

If you'd like to use a client certificate (note, most people don't want this) set the path to the x509 cert here

*client\_key*

- Value type is [path](#)
- There is no default value for this setting.

If you're using a client certificate specify the path to the encryption key here

*codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

*connect\_timeout*

- Value type is [number](#)
- Default value is 10

Timeout (in seconds) to wait for a connection to be established. Default is 10s

*content\_type*

- Value type is [string](#)
- There is no default value for this setting.

Content type

If not specified, this defaults to the following:

- if format is "json", "application/json"

- if format is "form", "application/x-www-form-urlencoded"

*cookies*

- Value type is [boolean](#)
- Default value is `true`

Enable cookie support. With this enabled the client will persist cookies across requests as a normal web browser would. Enabled by default

*enable\_metric*

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

*follow\_redirects*

- Value type is [boolean](#)
- Default value is `true`

Should redirects be followed? Defaults to `true`

*format*

- Value can be any of: `json`, `form`, `message`
- Default value is "`json`"

Set the format of the http body.

If form, then the body will be the mapping (or whole event) converted into a query parameter string, e.g. `foo=bar&baz=fizz...`

If message, then the body will be the result of formatting the event according to message

Otherwise, the event is sent as json.

*headers*

- Value type is [hash](#)
- There is no default value for this setting.

Custom headers to use format is `headers => ["X-My-Header", "%{host}"]`

### `http_method`

- This is a required setting.
- Value can be any of: put, post, patch, delete, get, head
- There is no default value for this setting.

The HTTP Verb. One of "put", "post", "patch", "delete", "get", "head"

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}

keepalive
```

- Value type is [boolean](#)
- Default value is `true`

Turn this on to enable HTTP keepalive support. We highly recommend setting `automatic_retries` to at least one with this to fix interactions with broken keepalive implementations.

### `keystore`

- Value type is [path](#)
- There is no default value for this setting.

If you need to use a custom keystore (`.jks`) specify that here. This does not work with `.pem` keys!

### `keystore_password`

- Value type is [password](#)
- There is no default value for this setting.

Specify the keystore password here. Note, most `.jks` files created with keytool require a password!

### *keystore\_type*

- Value type is [string](#)
- Default value is "JKS"

Specify the keystore type here. One of JKS or PKCS12. Default is JKS

### *mapping*

- Value type is [hash](#)
- There is no default value for this setting.

This lets you choose the structure and parts of the event that are sent.

For example:

```
mapping => {"foo", "%{host}", "bar", "%{type}"}
```

### *message*

- Value type is [string](#)
- There is no default value for this setting.

### *pool\_max*

- Value type is [number](#)
- Default value is 50

Max number of concurrent connections. Defaults to 50

### *pool\_max\_per\_route*

- Value type is [number](#)
- Default value is 25

Max number of concurrent connections to a single host. Defaults to 25

### *proxy*

- Value type is [string](#)
- There is no default value for this setting.

If you'd like to use an HTTP proxy . This supports multiple configuration syntaxes:

1. Proxy host in form: `http://proxy.org:1234`
2. Proxy host in form: `{host => "proxy.org", port => 80, scheme => 'http', user => 'username@host', password => 'password'}`
3. Proxy host in form: `{url => 'http://proxy.org:1234', user => 'username@host', password => 'password'}`

*request\_timeout*

- Value type is [number](#)
- Default value is 60

Timeout (in seconds) for the entire request

*retry\_non\_idempotent*

- Value type is [boolean](#)
- Default value is `false`

If `automatic_retries` is enabled this will cause non-idempotent HTTP verbs (such as POST) to be retried.

*socket\_timeout*

- Value type is [number](#)
- Default value is 10

Timeout (in seconds) to wait for data on the socket. Default is 10s

*ssl\_certificate\_validation*

- Value type is [boolean](#)
- Default value is `true`

Set this to false to disable SSL/TLS certificate validation Note: setting this to false is generally considered insecure!

*truststore*

- Value type is [path](#)
- There is no default value for this setting.

If you need to use a custom truststore (.jks) specify that here. This does not work with .pem certs!

*truststore\_password*

- Value type is [password](#)
- There is no default value for this setting.

Specify the truststore password here. Note, most .jks files created with keytool require a password!

`truststore_type`

- Value type is [string](#)
- Default value is "JKS"

Specify the truststore type here. One of `JKS` or `PKCS12`. Default is `JKS`

`url`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

This output lets you send events to a generic HTTP(S) endpoint

This output will execute up to `pool_max` requests in parallel for performance. Consider this when tuning this plugin for performance.

Additionally, note that when parallel execution is used strict ordering of events is not guaranteed!

Beware, this gem does not yet support codecs. Please use the `format` option for now. URL to use

`validate_after_inactivity`

- Value type is [number](#)
- Default value is 200

How long to wait before checking if the connection is stale before executing a request on a connection using `keepalive`. # You may want to set this lower, possibly to 0 if you get connection errors regularly Quoting the Apache commons docs (this client is based Apache Commons): *Defines period of inactivity in milliseconds after which persistent connections must be re-validated prior to being leased to the consumer. Non-positive value passed to this method disables connection validation. This check helps detect connections that have become stale (half-closed) while kept inactive in the pool. See [these docs for more info](#)*

`workers`

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the `:legacy` type This is hacky, but it can only be here

## influxdb

- Version: 4.0.0
- Released on: 2016-09-09
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
`bin/logstash-plugin install logstash-output-influxdb`.

This output lets you output Metrics to InfluxDB (>= 0.9.0-rc31)

The configuration here attempts to be as friendly as possible and minimize the need for multiple definitions to write to multiple measurements and still be efficient

the InfluxDB API let's you do some semblance of bulk operation per http call but each call is database-specific

You can learn more at [InfluxDB homepage](#)

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
influxdb {
 data_points => ...
 host => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
allow_time_override	<a href="#">boolean</a>	No	false
codec	<a href="#">codec</a>	No	"plain"
coerce_values	<a href="#">hash</a>	No	{ }
data_points	<a href="#">hash</a>	Yes	{ }
db	<a href="#">string</a>	No	"statistics"
enable_metric	<a href="#">boolean</a>	No	true
exclude_fields	<a href="#">array</a>	No	[ "@timestamp", "@version",

Setting	Input type	Required	Default value
			"sequence", "message", "type"]
flush_size	<a href="#">number</a>	No	100
host	<a href="#">string</a>	Yes	
id	<a href="#">string</a>	No	
idle_flush_time	<a href="#">number</a>	No	1
measurement	<a href="#">string</a>	No	"logstash"
password	<a href="#">password</a>	No	nil
port	<a href="#">number</a>	No	8086
retention_policy	<a href="#">string</a>	No	"default"
send_as_tags	<a href="#">array</a>	No	["host"]
ssl	<a href="#">boolean</a>	No	false
time_precision	<a href="#">string</a> , one of [ "n", "u", "ms", "s", "m", "h" ]	No	"ms"
use_event_fields_for_data_points	<a href="#">boolean</a>	No	false
user	<a href="#">string</a>	No	nil
workers	<>	No	1

## Details

`allow_time_override`

- Value type is [boolean](#)
- Default value is `false`

Allow the override of the `time` column in the event?

By default any column with a name of `time` will be ignored and the time will be determined by the value of `@timestamp`.

Setting this to `true` allows you to explicitly set the `time` column yourself

Note: `time` must be an epoch value in either seconds, milliseconds or microseconds

### `codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### `coerce_values`

- Value type is [hash](#)
- Default value is {}

Allow value coercion

this will attempt to convert data point values to the appropriate type before posting otherwise sprintf-filtered numeric values could get sent as strings format is `{'column_name' => 'datatype'}`

currently supported datatypes are `integer` and `float`

### `data_points`

- This is a required setting.
- Value type is [hash](#)
- Default value is {}

Hash of key/value pairs representing data points to send to the named database Example:  
`{'column1' => 'value1', 'column2' => 'value2'}`

Events for the same measurement will be batched together where possible Both keys and values support sprintf formatting

### `db`

- Value type is [string](#)
- Default value is "statistics"

The database to write - supports sprintf formatting

### `enable_metric`

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

*exclude\_fields*

- Value type is [array](#)
- Default value is ["@timestamp", "@version", "sequence", "message", "type"]

An array containing the names of fields from the event to exclude from the data points

Events, in general, contain keys "@version" and "@timestamp". Other plugins may add others that you'll want to exclude (such as "command" from the exec plugin).

This only applies when `use_event_fields_for_data_points` is true.

*flush\_size*

- Value type is [number](#)
- Default value is 100

This setting controls how many events will be buffered before sending a batch of events. Note that these are only batched for the same measurement

*host*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The hostname or IP address to reach your InfluxDB instance

*id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

idle_flush_time

- Value type is [number](#)
- Default value is 1

The amount of time since last flush before a flush is forced.

This setting helps ensure slow event rates don't get stuck in Logstash. For example, if your `flush_size` is 100, and you have received 10 events, and it has been more than `idle_flush_time` seconds since the last flush, logstash will flush those 10 events automatically.

This helps keep both fast and slow log streams moving along in near-real-time.

measurement

- Value type is [string](#)
- Default value is "logstash"

Measurement name - supports sprintf formatting

password

- Value type is [password](#)
- Default value is nil

The password for the user who access to the named database

port

- Value type is [number](#)
- Default value is 8086

The port for InfluxDB

retention_policy

- Value type is [string](#)
- Default value is "default"

The retention policy to use

send_as_tags

- Value type is [array](#)
- Default value is ["host"]

An array containing the names of fields to send to Influxdb as tags instead of fields. Influxdb 0.9 convention is that values that do not change every request should be considered metadata and given as tags.

ssl

- Value type is [boolean](#)
- Default value is false

Enable SSL/TLS secured communication to InfluxDB

time_precision

- Value can be any of: n, u, ms, s, m, h
- Default value is "ms"

Set the level of precision of `time`

only useful when overriding the time value

use_event_fields_for_data_points

- Value type is [boolean](#)
- Default value is `false`

Automatically use fields from the event as the data points sent to Influxdb

user

- Value type is [string](#)
- Default value is `nil`

The user who has access to the named database

workers

- Value type is [string](#)
- Default value is `1`

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

irc

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

Write events to IRC

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
irc {
    channels => ...
    host => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
channels	array	Yes	
codec	codec	No	"plain"
enable_metric	boolean	No	true
format	string	No	"%{message}"
host	string	Yes	
id	string	No	
messages_per_second	number	No	0.5
nick	string	No	"logstash"
password	password	No	
port	number	No	6667
post_string	string	No	
pre_string	string	No	
real	string	No	"logstash"
secure	boolean	No	false
user	string	No	"logstash"
workers	<>	No	1

Details

channels

- This is a required setting.
- Value type is [array](#)
- There is no default value for this setting.

Channels to broadcast to.

These should be full channel names including the # symbol, such as "#logstash".

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

enable_metric

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

format

- Value type is [string](#)
- Default value is "%{message}"

Message format to send, event tokens are usable here

host

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Address of the host to connect to

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
  stdout {
    id => "my_plugin_id"
  }
}
messages_per_second
```

- Value type is [number](#)
- Default value is `0.5`

Limit the rate of messages sent to IRC in messages per second.

nick

- Value type is [string](#)
- Default value is `"logstash"`

IRC Nickname

password

- Value type is [password](#)
- There is no default value for this setting.

IRC server password

port

- Value type is [number](#)
- Default value is `6667`

Port on host to connect to.

post_string

- Value type is [string](#)
- There is no default value for this setting.

Static string after event

pre_string

- Value type is [string](#)
- There is no default value for this setting.

Static string before event

real

- Value type is [string](#)
- Default value is "logstash"

IRC Real name

secure

- Value type is [boolean](#)
- Default value is `false`

Set this to true to enable SSL.

user

- Value type is [string](#)
- Default value is "logstash"

IRC Username

workers

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

jira

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-output-jira`.

Origin <https://groups.google.com/forum/#!msg/logstash-users/exgrB4iQ-mw/R34apku5nXsJ> and <https://botbot.me/freenode/logstash/msg/4169496/> via <https://gist.github.com/electrical/4660061e8fff11cdcf37#file-jira-rb> Uses jiralicious as the bridge to JIRA By Martin Cleaver, Blended Perspectives with a lot of help from *electrical* in #logstash

This is so is most useful so you can use logstash to parse and structure your logs and ship structured, json events to JIRA

To use this, you'll need to ensure your JIRA instance allows REST calls

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
jira {
    issuetypeid => ...
    password => ...
    priority => ...
    projectid => ...
    summary => ...
    username => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
assignee	string	No	
codec	codec	No	"plain"
host	string	No	
issuetypeid	string	Yes	
password	string	Yes	
priority	string	Yes	
projectid	string	Yes	
reporter	string	No	

Setting	Input type	Required	Default value
summary	string	Yes	
username	string	Yes	
workers	number	No	1

Details

assignee

- Value type is [string](#)
- There is no default value for this setting.

JIRA Reporter

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

exclude_tags (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [array](#)
- Default value is []

Only handle events without any of these tags. Optional.

host

- Value type is [string](#)
- There is no default value for this setting.

The hostname to send logs to. This should target your JIRA server and has to have the REST interface enabled

issuetypeid

- This is a required setting.
- Value type is [string](#)

- There is no default value for this setting.

JIRA Issuetype number

password

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

priority

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

JIRA Priority

projectid

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Javalicious has no proxy support JIRA Project number

reporter

- Value type is [string](#)
- There is no default value for this setting.

JIRA Reporter

summary

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

JIRA Summary

tags (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [array](#)
- Default value is []

Only handle events with all of these tags. Optional.

type (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [string](#)
- Default value is ""

The type to act on. If a type is given, then this output will only act on messages with the same type. See any input plugin's `type` attribute for more. Optional.

username

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

workers

- Value type is [number](#)
- Default value is 1

The number of workers to use for this output. Note that this setting may not be useful for all outputs.

juggernaut

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

This plugin does not ship with Logstash by default, but it is easy to install by running
 bin/logstash-plugin install logstash-output-juggernaut.

Push messages to the juggernaut websockets server:

- <https://github.com/maccman/juggernaut>

Wraps Websockets and supports other methods (including xhr longpolling) This is basically, just an extension of the redis output (Juggernaut pulls messages from redis). But it pushes messages to a particular channel and formats the messages in the way juggernaut expects.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
juggernaut {
    channels => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
channels	array	Yes	
codec	codec	No	"plain"
db	number	No	0
enable_metric	boolean	No	true
host	string	No	"127.0.0.1"
id	string	No	
message_format	string	No	
password	password	No	
port	number	No	6379
timeout	number	No	5

Setting	Input type	Required	Default value
workers	<<,>>	No	1

Details

channels

- This is a required setting.
- Value type is [array](#)
- There is no default value for this setting.

List of channels to which to publish. Dynamic names are valid here, for example `logstash-%{type}`.

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

db

- Value type is [number](#)
- Default value is 0

The redis database number.

enable_metric

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

host

- Value type is [string](#)
- Default value is "127.0.0.1"

The hostname of the redis server to which juggernaut is listening.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

### *message\_format*

- Value type is [string](#)
- There is no default value for this setting.

How should the message be formatted before pushing to the websocket.

### *password*

- Value type is [password](#)
- There is no default value for this setting.

Password to authenticate with. There is no authentication by default.

### *port*

- Value type is [number](#)
- Default value is 6379

The port to connect on.

### *timeout*

- Value type is [number](#)
- Default value is 5

Redis initial connection timeout in seconds.

### *workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here



[librato »](#)

- Version: 5.1.5
- Released on: 2017-01-25
- [Changelog](#)

Write events to a Kafka topic. This uses the Kafka Producer API to write messages to a topic on the broker.

Here's a compatibility matrix that shows the Kafka client versions that are compatible with each combination of Logstash and the Kafka output plugin:

Kafka Client Version	Logstash Version	Plugin Version	Why?
0.8	2.0.0 - 2.x.x	<3.0.0	Legacy, 0.8 is still popular
0.9	2.0.0 - 2.3.x	3.x.x	Works with the old Ruby Event API (event['product']['price'] = 10)
0.9	2.4.x - 5.x.x	4.x.x	Works with the new getter/setter APIs (event.set('product')[price], 10))
0.10.0.x	2.4.x - 5.x.x	5.x.x	Not compatible with the <= 0.9 broker

We recommended that you use matching Kafka client and broker versions. During upgrades, you should upgrade brokers before clients because brokers target backwards compatibility. For example, the 0.9 broker is compatible with both the 0.8 consumer and 0.9 consumer APIs, but not the other way around.

This output supports connecting to Kafka over:

- SSL (requires plugin version 3.0.0 or later)
- Kerberos SASL (requires plugin version 5.1.0 or later)

By default security is disabled but can be turned on as needed.

The only required configuration is the `topic_id`. The default codec is plain, so events will be persisted on the broker in json format. If you select a codec of plain, Logstash will encode your messages with not only the message but also with a timestamp and hostname. If you do not want anything but your message passing through, you should make the output configuration something like:

```
output {
```

```

kafka {
 codec => plain {
 format => "%{message}"
 }
 topic_id => "mytopic"
}
}

```

For more information see  
<http://kafka.apache.org/documentation.html#theproducer>

Kafka producer configuration: <http://kafka.apache.org/documentation.html#newproducerconfigs>

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```

kafka {
 topic_id => ...
}

```

Available configuration options:

Setting	Input type	Required	Default value
acks	<a href="#">string</a> , one of ["0", "1", "all"]	No	"1"
batch_size	<a href="#">number</a>	No	16384
bootstrap_servers	<a href="#">string</a>	No	"localhost:9092"
buffer_memory	<a href="#">number</a>	No	33554432
client_id	<a href="#">string</a>	No	
codec	<a href="#">codec</a>	No	"plain"
compression_type	<a href="#">string</a> , one of ["none", "gzip", "snappy", "lz4"]	No	"none"
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	

<b>Setting</b>	<b>Input type</b>	<b>Required</b>	<b>Default value</b>
jaas_path	a valid filesystem path	No	
kerberos_config	a valid filesystem path	No	
key_serializer	<a href="#">string</a>	No	"org.apache.kafka.common.serialization.StringSerializer"
linger_ms	<a href="#">number</a>	No	0
max_request_size	<a href="#">number</a>	No	1048576
message_key	<a href="#">string</a>	No	
metadata_fetch_timeout_ms	<a href="#">number</a>	No	60000
metadata_max_age_ms	<a href="#">number</a>	No	300000
receive_buffer_bytes	<a href="#">number</a>	No	32768
reconnect_backoff_ms	<a href="#">number</a>	No	10
request_timeout_ms	<a href="#">string</a>	No	
retries	<a href="#">number</a>	No	0
retry_backoff_ms	<a href="#">number</a>	No	100
sasl_kerberos_service_name	<a href="#">string</a>	No	
sasl_mechanism	<a href="#">string</a>	No	"GSSAPI"
security_protocol	<a href="#">string</a> , one of [ "PLAINTEXT", "SSL", "SASL_PLAINTEXT", "SASL_SSL" ]	No	"PLAINTEXT"
send_buffer_bytes	<a href="#">number</a>	No	131072
ssl_key_password	<a href="#">password</a>	No	
ssl_keystore_location	a valid filesystem path	No	
ssl_keystore_password	<a href="#">password</a>	No	
ssl_keystore_type	<a href="#">string</a>	No	

Setting	Input type	Required	Default value
ssl_truststore_location	a valid filesystem path	No	
ssl_truststore_password	<a href="#">password</a>	No	
ssl_truststore_type	<a href="#">string</a>	No	
topic_id	<a href="#">string</a>	Yes	
value_serializer	<a href="#">string</a>	No	"org.apache.kafka.common.serialization.StringSerializer"
workers	<>	No	1

## Details

### [\*acks\*](#)

- Value can be any of: 0, 1, all
- Default value is "1"

The number of acknowledgments the producer requires the leader to have received before considering a request complete.

acks=0, the producer will not wait for any acknowledgment from the server at all. acks=1, This will mean the leader will write the record to its local log but will respond without awaiting full acknowledgement from all followers. acks=all, This means the leader will wait for the full set of in-sync replicas to acknowledge the record.

### [\*batch\\_size\*](#)

- Value type is [number](#)
- Default value is 16384

The producer will attempt to batch records together into fewer requests whenever multiple records are being sent to the same partition. This helps performance on both the client and the server. This configuration controls the default batch size in bytes.

### [\*block\\_on\\_buffer\\_full \(DEPRECATED\)\*](#)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.

- Value type is [boolean](#)
- Default value is `true`

When our memory buffer is exhausted we must either stop accepting new records (block) or throw errors. By default this setting is true and we block, however in some scenarios blocking is not desirable and it is better to immediately give an error.

### *bootstrap\_servers*

- Value type is [string](#)
- Default value is `"localhost:9092"`

This is for bootstrapping and the producer will only use it for getting metadata (topics, partitions and replicas). The socket connections for sending the actual data will be established based on the broker information returned in the metadata. The format is `host1:port1,host2:port2`, and the list can be a subset of brokers or a VIP pointing to a subset of brokers.

### *buffer\_memory*

- Value type is [number](#)
- Default value is `33554432`

The total bytes of memory the producer can use to buffer records waiting to be sent to the server.

### *client\_id*

- Value type is [string](#)
- There is no default value for this setting.

The id string to pass to the server when making requests. The purpose of this is to be able to track the source of requests beyond just ip/port by allowing a logical application name to be included with the request

### *codec*

- Value type is [codec](#)
- Default value is `"plain"`

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### *compression\_type*

- Value can be any of: `none, gzip, snappy, lz4`
- Default value is `"none"`

The compression type for all data generated by the producer. The default is none (i.e. no compression). Valid values are none, gzip, or snappy.

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

### `jaas_path`

- Value type is [path](#)
- There is no default value for this setting.

The Java Authentication and Authorization Service (JAAS) API supplies user authentication and authorization services for Kafka. This setting provides the path to the JAAS file. Sample JAAS file for Kafka client:

```
KafkaClient {
 com.sun.security.auth.module.Krb5LoginModule required
 useTicketCache=true
 renewTicket=true
 serviceName="kafka";
};
```

Please note that specifying `jaas_path` and `kerberos_config` in the config file will add these to the global JVM system properties. This means if you have multiple Kafka inputs, all of them would be sharing the same `jaas_path` and `kerberos_config`. If this is not desirable, you would have to run separate instances of Logstash on different JVM instances.

*kerberos\_config*

- Value type is [path](#)
- There is no default value for this setting.

Optional path to kerberos config file. This is krb5.conf style as detailed in  
[https://web.mit.edu/kerberos/krb5-1.12/doc/admin/conf\\_files/krb5\\_conf.html](https://web.mit.edu/kerberos/krb5-1.12/doc/admin/conf_files/krb5_conf.html)

*key\_serializer*

- Value type is [string](#)
- Default value is "org.apache.kafka.common.serialization.StringSerializer"

Serializer class for the key of the message

*linger\_ms*

- Value type is [number](#)
- Default value is 0

The producer groups together any records that arrive in between request transmissions into a single batched request. Normally this occurs only under load when records arrive faster than they can be sent out. However in some circumstances the client may want to reduce the number of requests even under moderate load. This setting accomplishes this by adding a small amount of artificial delay—that is, rather than immediately sending out a record the producer will wait for up to the given delay to allow other records to be sent so that the sends can be batched together.

*max\_request\_size*

- Value type is [number](#)
- Default value is 1048576

The maximum size of a request

*message\_key*

- Value type is [string](#)
- There is no default value for this setting.

The key for the message

*metadata\_fetch\_timeout\_ms*

- Value type is [number](#)
- Default value is 60000

the timeout setting for initial metadata request to fetch topic metadata.

### `metadata_max_age_ms`

- Value type is [number](#)
- Default value is 300000

the max time in milliseconds before a metadata refresh is forced.

### `receive_buffer_bytes`

- Value type is [number](#)
- Default value is 32768

The size of the TCP receive buffer to use when reading data

### `reconnect_backoff_ms`

- Value type is [number](#)
- Default value is 10

The amount of time to wait before attempting to reconnect to a given host when a connection fails.

### `request_timeout_ms`

- Value type is [string](#)
- There is no default value for this setting.

The configuration controls the maximum amount of time the client will wait for the response of a request. If the response is not received before the timeout elapses the client will resend the request if necessary or fail the request if retries are exhausted.

### `retries`

- Value type is [number](#)
- Default value is 0

Setting a value greater than zero will cause the client to resend any record whose send fails with a potentially transient error.

### `retry_backoff_ms`

- Value type is [number](#)
- Default value is 100

The amount of time to wait before attempting to retry a failed produce request to a given topic partition.

*sasl\_kerberos\_service\_name*

- Value type is [string](#)
- There is no default value for this setting.

The Kerberos principal name that Kafka broker runs as. This can be defined either in Kafka's JAAS config or in Kafka's config.

*sasl\_mechanism*

- Value type is [string](#)
- Default value is "GSSAPI"

[SASL mechanism](#) used for client connections. This may be any mechanism for which a security provider is available. GSSAPI is the default mechanism.

*security\_protocol*

- Value can be any of: PLAINTEXT, SSL, SASL\_PLAINTEXT, SASL\_SSL
- Default value is "PLAINTEXT"

Security protocol to use, which can be either of  
PLAINTEXT,SSL,SASL\_PLAINTEXT,SASL\_SSL

*send\_buffer\_bytes*

- Value type is [number](#)
- Default value is 131072

The size of the TCP send buffer to use when sending data.

*ssl (DEPRECATED)*

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [boolean](#)
- Default value is `false`

Enable SSL/TLS secured communication to Kafka broker.

*ssl\_key\_password*

- Value type is [password](#)
- There is no default value for this setting.

The password of the private key in the key store file.

*ssl\_keystore\_location*

- Value type is [path](#)
- There is no default value for this setting.

If client authentication is required, this setting stores the keystore path.

*ssl\_keystore\_password*

- Value type is [password](#)
- There is no default value for this setting.

If client authentication is required, this setting stores the keystore password

*ssl\_keystore\_type*

- Value type is [string](#)
- There is no default value for this setting.

The keystore type.

*ssl\_truststore\_location*

- Value type is [path](#)
- There is no default value for this setting.

The JKS truststore path to validate the Kafka broker's certificate.

*ssl\_truststore\_password*

- Value type is [password](#)
- There is no default value for this setting.

The truststore password

*ssl\_truststore\_type*

- Value type is [string](#)
- There is no default value for this setting.

The truststore type.

*timeout\_ms (DEPRECATED)*

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [number](#)

- Default value is 30000

The configuration controls the maximum amount of time the server will wait for acknowledgments from followers to meet the acknowledgment requirements the producer has specified with the acks configuration. If the requested number of acknowledgments are not met when the timeout elapses an error will be returned. This timeout is measured on the server side and does not include the network latency of the request.

*topic\_id*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The topic to produce messages to

*value\_serializer*

- Value type is [string](#)
- Default value is "org.apache.kafka.common.serialization.StringSerializer"

Serializer class for the value of the message

*workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

## librato

- Version: 3.0.0
- Released on: 2016-09-09
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
`bin/logstash-plugin install logstash-output-librato.`

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
librato {
 account_id => ...
 api_token => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
account_id	<a href="#">string</a>	Yes	
annotation	<a href="#">hash</a>	No	{ }
api_token	<a href="#">string</a>	Yes	
batch_size	<a href="#">string</a>	No	"10"
codec	<a href="#">codec</a>	No	"plain"
counter	<a href="#">hash</a>	No	{ }
enable_metric	<a href="#">boolean</a>	No	true
gauge	<a href="#">hash</a>	No	{ }
id	<a href="#">string</a>	No	
workers	<code>&lt;&lt;, &gt;&gt;</code>	No	1

## Details

### *account\_id*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

This output lets you send metrics, annotations and alerts to Librato based on Logstash events

This is VERY experimental and inefficient right now. Your Librato account usually an email address

### *annotation*

- Value type is [hash](#)
- Default value is `{ }`

Annotations Registers an annotation with Librato. The only required field is `title` and `name`. `start_time` and `end_time` will be set to `event["@timestamp"].to_i`. You can add any other optional annotation values as well. All values will be passed through `event.strftime`

Example:

```
{
 "title" => "Logstash event on %{host}"
 "name" => "logstash_stream"
}
or
{
 "title" => "Logstash event"
 "description" => "%{message}"
 "name" => "logstash_stream"
}
api_token
```

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Your Librato API Token

### *batch\_size*

- Value type is [string](#)
- Default value is "10"

Batch size Number of events to batch up before sending to Librato.

### *codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### *counter*

- Value type is [hash](#)
- Default value is {}

Counters Send data to Librato as a counter

Example:

```
{
 "value" => "1"
 "source" => "%{host}"
 "name" => "messages_received"
}
```

Additionally, you can override the `measure_time` for the event. Must be a unix timestamp:

```
{
 "value" => "1"
 "source" => "%{host}"
 "name" => "messages_received"
 "measure_time" => "%{my_unixtime_field}"
}
Default is to use the event's timestamp
enable_metric
```

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### *gauge*

- Value type is [hash](#)
- Default value is {}

Gauges Send data to Librato as a gauge

Example:

```
{
 "value" => "%{bytes_received}"
 "source" => "%{host}"
 "name" => "apache_bytes"
}
Additionally, you can override the `measure_time` for the event. Must be a
unix timestamp:
{
 "value" => "%{bytes_received}"
 "source" => "%{host}"
 "name" => "apache_bytes"
 "measure_time" => "%{my_unixtime_field}"
}
Default is to use the event's timestamp
```

*id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

workers

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

loggly

- Version: 3.0.0
- Released on: 2016-09-09
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
bin/logstash-plugin install logstash-output-loggly.

TODO(sissel): Move to something that performs better than net/http Ugly monkey patch to get around <http://jira.codehaus.org/browse/JRUBY-5529>

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
loggly {
    key => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
can_retry	boolean	No	true
codec	codec	No	"plain"
enable_metric	boolean	No	true
host	string	No	"logs-01.loggly.com"
id	string	No	
key	string	Yes	
proto	string	No	"http"
proxy_host	string	No	
proxy_password	password	No	""
proxy_port	number	No	
proxy_user	string	No	
retry_count	number	No	5
tag	string	No	"logstash"
workers	<<, >>	No	1

Details

`can_retry`

- Value type is [boolean](#)
- Default value is `true`

Can Retry. Setting this value true helps user to send multiple retry attempts if the first request fails

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`host`

- Value type is [string](#)
- Default value is "logs-01.loggly.com"

The hostname to send logs to. This should target the loggly http input server which is usually "logs-01.loggly.com" (Gen2 account). See Loggly HTTP endpoint documentation at <https://www.loggly.com/docs/http-endpoint/>

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique ID to the plugin instance, this ID is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

### `key`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The loggly http input key to send to. This is usually visible in the Loggly *Inputs* page as something like this:

```
https://logs-01.loggly.net/inputs/abcdef12-3456-7890-abcd-ef0123456789
^^^^^^^^^^^^^^^^^
/ \-----> key <-----
```

You can use `%{foo}` field lookups here if you need to pull the api key from the event. This is mainly aimed at multitenant hosting providers who want to offer shipping a customer's logs to that customer's loggly account.

### `proto`

- Value type is [string](#)
- Default value is "http"

Should the log action be sent over https instead of plain http

### `proxy_host`

- Value type is [string](#)
- There is no default value for this setting.

### Proxy Host

### `proxy_password`

- Value type is [password](#)
- Default value is ""

### Proxy Password

### `proxy_port`

- Value type is [number](#)
- There is no default value for this setting.

### Proxy Port

### *proxy\_user*

- Value type is [string](#)
- There is no default value for this setting.

## Proxy Username

### *retry\_count*

- Value type is [number](#)
- Default value is 5

Retry count. It may be possible that the request may timeout due to slow Internet connection if such condition appears, retry\_count helps in retrying request for multiple times. It will try to submit request until retry\_count and then halt

### *tag*

- Value type is [string](#)
- Default value is "logstash"

Loggly Tag Tag helps you to find your logs in the Loggly dashboard easily. You can make a search in Loggly using tag as "tag:logstash-contrib" or the tag set by you in the config file.

Helpful for leveraging Loggly source groups. <https://www.loggly.com/docs/source-groups/>

### *workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type. This is hacky, but it can only be here.

## **lumberjack**

- Version: 3.1.2
- Released on: 2016-09-15
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
`bin/logstash-plugin install logstash-output-lumberjack.`

### **Synopsis**

This plugin supports the following configuration options:

Required configuration options:

```
lumberjack {
 hosts => ...
 port => ...
 ssl_certificate => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	<a href="#">codec</a>	No	"plain"
enable_metric	<a href="#">boolean</a>	No	true
flush_size	<a href="#">number</a>	No	1024
hosts	<a href="#">array</a>	Yes	
id	<a href="#">string</a>	No	
idle_flush_time	<a href="#">number</a>	No	1
port	<a href="#">number</a>	Yes	
ssl_certificate	a valid filesystem path	Yes	
workers	<<, >>	No	1

### **Details**

### *codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### *enable\_metric*

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### *flush\_size*

- Value type is [number](#)
- Default value is `1024`

To make efficient calls to the lumberjack output we are buffering events locally. if the number of events exceed the number the declared `flush_size` we will send them to the logstash server.

### *hosts*

- This is a required setting.
- Value type is [array](#)
- There is no default value for this setting.

list of addresses lumberjack can send to

### *id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

idle_flush_time

- Value type is [number](#)

- Default value is 1

The amount of time since last flush before a flush is forced.

This setting helps ensure slow event rates don't get stuck in Logstash. For example, if your `flush_size` is 100, and you have received 10 events, and it has been more than `idle_flush_time` seconds since the last flush, Logstash will flush those 10 events automatically.

This helps keep both fast and slow log streams moving along in near-real-time.

port

- This is a required setting.
- Value type is [number](#)
- There is no default value for this setting.

the port to connect to

ssl_certificate

- This is a required setting.
- Value type is [path](#)
- There is no default value for this setting.

ssl certificate to use

workers

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

metriccatcher

- Version: 3.0.0
- Released on: 2016-09-10
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
`bin/logstash-plugin install logstash-output-metriccatcher`.

This output ships metrics to MetricCatcher, allowing you to utilize Coda Hale's Metrics.

More info on MetricCatcher: <https://github.com/clearspring/MetricCatcher>

At Clearspring, we use it to count the response codes from Apache logs:

```
metriccatcher {
  host => "localhost"
  port => "1420"
  type => "apache-access"
  fields => [ "response" ]
  meter => {
    "%{host}.apache.response.%{response}" => "1"
  }
}
```

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
metriccatcher { }
```

Available configuration options:

Setting	Input type	Required	Default value
biased	hash	No	
codec	codec	No	"plain"
counter	hash	No	
enable_metric	boolean	No	true
gauge	hash	No	
host	string	No	"localhost"

Setting	Input type	Required	Default value
id	string	No	
meter	hash	No	
port	number	No	1420
timer	hash	No	
uniform	hash	No	
workers	<code><<, >></code>	No	1

Details

biased

- Value type is [hash](#)
- There is no default value for this setting.

The metrics to send. This supports dynamic strings like `%{host}` for metric names and also for values. This is a hash field with key of the metric name, value of the metric value.

The value will be coerced to a floating point value. Values which cannot be coerced will zero (0)

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

counter

- Value type is [hash](#)
- There is no default value for this setting.

The metrics to send. This supports dynamic strings like `%{host}` for metric names and also for values. This is a hash field with key of the metric name, value of the metric value. Example:

```
counter => { "%{host}.apache.hits.%{response} => "1" }
```

The value will be coerced to a floating point value. Values which cannot be coerced will zero (0)

enable_metric

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

gauge

- Value type is [hash](#)
- There is no default value for this setting.

The metrics to send. This supports dynamic strings like `%{host}` for metric names and also for values. This is a hash field with key of the metric name, value of the metric value.

The value will be coerced to a floating point value. Values which cannot be coerced will zero (0)

host

- Value type is [string](#)
- Default value is "localhost"

The address of the MetricCatcher

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique ID to the plugin instance, this ID is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

*meter*

- Value type is [hash](#)
- There is no default value for this setting.

The metrics to send. This supports dynamic strings like `%{host}` for metric names and also for values. This is a hash field with key of the metric name, value of the metric value.

The value will be coerced to a floating point value. Values which cannot be coerced will zero (0)

*port*

- Value type is [number](#)
- Default value is 1420

The port to connect on your MetricCatcher

*timer*

- Value type is [hash](#)
- There is no default value for this setting.

for metric names and also for values. This is a hash field with key of the metric name, value of the metric value. Example:

```
timer => { "%{host}.apache.response_time => "%{response_time}" }
```

The value will be coerced to a floating point value. Values which cannot be coerced will zero (0)

*uniform*

- Value type is [hash](#)
- There is no default value for this setting.

The metrics to send. This supports dynamic strings like %{host} for metric names and also for values. This is a hash field with key of the metric name, value of the metric value.

The value will be coerced to a floating point value. Values which cannot be coerced will zero (0)

*workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

## mongodb

- Version: 3.0.0
- Released on: 2017-01-16
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
 bin/logstash-plugin install logstash-output-mongodb.

```
require_relative "bson/logstash_event"
```

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
mongodb {
 collection => ...
 database => ...
 uri => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	<a href="#">codec</a>	No	"plain"
collection	<a href="#">string</a>	Yes	
database	<a href="#">string</a>	Yes	
enable_metric	<a href="#">boolean</a>	No	true
generateId	<a href="#">boolean</a>	No	false
id	<a href="#">string</a>	No	
isodate	<a href="#">boolean</a>	No	false
retry_delay	<a href="#">number</a>	No	3
uri	<a href="#">string</a>	Yes	
workers	<<, >>	No	1

## Details

### *codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### *collection*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The collection to use. This value can use `%{foo}` values to dynamically select a collection based on data in the event.

### *database*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The database to use

### *enable\_metric*

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### *generateId*

- Value type is [boolean](#)
- Default value is `false`

If true, an "`_id`" field will be added to the document before insertion. The "`_id`" field will use the timestamp of the event and overwrite an existing "`_id`" field in the event.

### *id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

isodate

- Value type is [boolean](#)
- Default value is `false`

If true, store the `@timestamp` field in mongodb as an `ISODate` type instead of an `ISO8601` string. For more information about this, see <http://www.mongodb.org/display/DOCS/Dates>

retry_delay

- Value type is [number](#)
- Default value is `3`

Number of seconds to wait after failure before retrying

uri

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

a MongoDB URI to connect to See <http://docs.mongodb.org/manual/reference/connection-string/>

workers

- Value type is [string](#)
- Default value is `1`

TODO remove this in Logstash 6.0 when we no longer support the `:legacy` type This is hacky, but it can only be here

nagios

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

The Nagios output is used for sending passive check results to Nagios via the Nagios command file. This output currently supports Nagios 3.

For this output to work, your event *must* have the following Logstash event fields:

- `nagios_host`
- `nagios_service`

These Logstash event fields are supported, but optional:

- `nagios_annotation`
- `nagios_level` (*overrides* `nagios_level` configuration option)

There are two configuration options:

- `commandfile` - The location of the Nagios external command file. Defaults to `/var/lib/nagios3/rw/nagios.cmd`
- `nagios_level` - Specifies the level of the check to be sent. Defaults to CRITICAL and can be overridden by setting the "nagios_level" field to one of "OK", "WARNING", "CRITICAL", or "UNKNOWN"
- `output{`
- `if [message] =~ /(error|ERROR|CRITICAL)/ {`
- `nagios {`
- `# your config here`
- `}`
- `}`
- }

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
nagios {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	codec	No	"plain"
commandfile	<<,>>	No	"/var/lib/nagios3/rw/nagios.cmd"
enable_metric	boolean	No	true
id	string	No	
nagios_level	string , one of ["0", "1", "2", "3"]	No	"2"
workers	<<,>>	No	1

Details

[codec](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

[commandfile](#)

- Value type is [string](#)
- Default value is "/var/lib/nagios3/rw/nagios.cmd"

The full path to your Nagios command file.

[enable_metric](#)

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

[id](#)

- Value type is [string](#)
- There is no default value for this setting.

Add a unique ID to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when

you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
  stdout {
    id => "my_plugin_id"
  }
}

nagios_level
```

- Value can be any of: 0, 1, 2, 3
- Default value is "2"

The Nagios check level. Should be one of 0=OK, 1=WARNING, 2=CRITICAL, 3=UNKNOWN. Defaults to 2 - CRITICAL.

workers

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

nagios_nsca

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
`bin/logstash-plugin install logstash-output-nagios_nsca.`

The `nagios_nsca` output is used for sending passive check results to Nagios through the NSCA protocol.

This is useful if your Nagios server is not the same as the source host from where you want to send logs or alerts. If you only have one server, this output is probably overkill # for you, take a look at the `nagios` output instead.

Here is a sample config using the `nagios_nsca` output:

```
output {
  nagios_nsca {
    # specify the hostname or ip of your nagios server
    host => "nagios.example.com"
    # specify the port to connect to
    port => 5667
  }
}
```

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
nagios_nsca {
  nagios_status => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	codec	No	"plain"
enable_metric	boolean	No	true
host	string	No	"localhost"

Setting	Input type	Required	Default value
id	string	No	
message_format	string	No	"#{@timestamp} %{host}: %{message}"
nagios_host	string	No	"%{host}"
nagios_service	string	No	"LOGSTASH"
nagios_status	string	Yes	
port	number	No	5667
send_nsca_bin	string	No	/usr/sbin/send_nsca
send_nsca_config	a valid filesystem path	No	
workers	<<,>>	No	1

Details

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

enable_metric

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

host

- Value type is [string](#)
- Default value is "localhost"

The nagios host or IP to send logs to. It should have a NSCA daemon running.

id

- Value type is [string](#)

- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

### `message_format`

- Value type is [string](#)
- Default value is "`%{@timestamp} %{host}: %{message}`"

The format to use when writing events to nagios. This value supports any string and can include `%{name}` and other dynamic strings.

### `nagios_host`

- Value type is [string](#)
- Default value is "`%{host}`"

The nagios `host` you want to submit a passive check result to. This parameter accepts interpolation, e.g. you can use `@source_host` or other logstash internal variables.

### `nagios_service`

- Value type is [string](#)
- Default value is "`LOGSTASH`"

The nagios `service` you want to submit a passive check result to. This parameter accepts interpolation, e.g. you can use `@source_host` or other logstash internal variables.

### `nagios_status`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The status to send to nagios. Should be 0 = OK, 1 = WARNING, 2 = CRITICAL, 3 = UNKNOWN

### `port`

- Value type is [number](#)
- Default value is 5667

The port where the NSCA daemon on the nagios host listens.

`send_nsca_bin`

- Value type is [string](#)
- Default value is "/usr/sbin/send\_nsca"

The path to the *send\_nsca* binary on the local host.

`send_nsca_config`

- Value type is [path](#)
- There is no default value for this setting.

The path to the *send\_nsca config* file on the local host. Leave blank if you don't want to provide a config file.

`workers`

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

## newrelic

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-output-newrelic`.

This output sends logstash events to New Relic Insights as custom events.

You can learn more about New Relic Insights here: <https://docs.newrelic.com/docs/insights/new-relic-insights/understanding-insights/new-relic-insights>

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
newrelic {
 account_id => ...
 insert_key => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
account_id	<a href="#">string</a>	Yes	
batch	<a href="#">boolean</a>	No	true
batch_events	<a href="#">number</a>	No	10
batch_timeout	<a href="#">number</a>	No	5
codec	<a href="#">codec</a>	No	"plain"
enable_metric	<a href="#">boolean</a>	No	true
event_type	<a href="#">string</a>	No	"LogstashEvent"
id	<a href="#">string</a>	No	
insert_key	<a href="#">string</a>	Yes	
proto	<a href="#">string</a>	No	"https"
proxy_host	<a href="#">string</a>	No	
proxy_password	<a href="#">password</a>	No	""
proxy_port	<a href="#">number</a>	No	80
proxy_user	<a href="#">string</a>	No	

Setting	Input type	Required	Default value
workers	<<, >>	No	1

## Details

### *account\_id*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Your New Relic account ID. This is the 5 or 6-digit number found in the URL when you are logged into New Relic: [account\\_id](#)/...

### *batch*

- Value type is [boolean](#)
- Default value is `true`

Batch Processing - all optional This plugin uses the New Relic Insights REST API to send data. To make efficient REST API calls, we will buffer a certain number of events before flushing that out to Insights.

### *batch\_events*

- Value type is [number](#)
- Default value is 10

This setting controls how many events will be buffered before sending a batch of events.

### *batch\_timeout*

- Value type is [number](#)
- Default value is 5

This setting controls how long the output will wait before sending a batch of events, should the minimum specified in `batch_events` not be met yet.

### *codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`event_type`

- Value type is [string](#)
- Default value is `"LogstashEvent"`

The name for your event type. Use alphanumeric characters only. If left out, your events will be stored under "logstashEvent".

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

`insert_key`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Your Insights Insert Key. You will need to generate one if you haven't already, as described here: <https://docs.newrelic.com/docs/insights/new-relic-insights/adding-querying-data/inserting-custom-events-insights-api#register>

`proto`

- Value type is [string](#)
- Default value is `"https"`

Should the log events be sent to Insights over https instead of plain http (typically yes).

proxy_host

- Value type is [string](#)
- There is no default value for this setting.

Proxy info - all optional If using a proxy, only proxy_host is required.

proxy_password

- Value type is [password](#)
- Default value is ""

Proxy_password should be left out if connecting to your proxy unauthenticated.

proxy_port

- Value type is [number](#)
- Default value is 80

Proxy_port will default to port 80 if left out.

proxy_user

- Value type is [string](#)
- There is no default value for this setting.

Proxy_user should be left out if connecting to your proxy unauthenticated.

workers

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

opentsdb

- Version: 3.1.1
- Released on: 2016-07-14
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
`bin/logstash-plugin install logstash-output-opentsdb`.

This output allows you to pull metrics from your logs and ship them to opentsdb. Opentsdb is an open source tool for storing and graphing metrics.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
opentsdb {
    metrics => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	codec	No	"plain"
enable_metric	boolean	No	true
host	string	No	"localhost"
id	string	No	
metrics	array	Yes	
port	number	No	4242
workers	<code><<, >></code>	No	1

Details

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`host`

- Value type is [string](#)
- Default value is "localhost"

The address of the opentsdb server.

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

### `metrics`

- This is a required setting.
- Value type is [array](#)
- There is no default value for this setting.

for metric names and also for values. This is an array field with key of the metric name, value of the metric value, and multiple tag,values . Example:

```
[
 "%{host}/uptime",
 %{uptime_1m} ",
 "hostname" ,
 "%{host}
 "anotherhostname" ,
 "%{host}
]
```

The value will be coerced to a floating point value. Values which cannot be coerced will zero (0)

*port*

- Value type is [number](#)
- Default value is 4242

The port to connect on your graphite server.

*workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

## pagerduty

- Version: 3.0.3
- Released on: 2016-09-15
- [Changelog](#)

The PagerDuty output will send notifications based on pre-configured services and escalation policies. Logstash can send "trigger", "acknowledge" and "resolve" event types. In addition, you may configure custom descriptions and event details. The only required field is the PagerDuty "Service API Key", which can be found on the service's web page on pagerduty.com. In the default case, the description and event details will be populated by Logstash, using `message`, `timestamp` and `host` data.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
pagerduty {
 service_key => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">description</a>	<a href="#">string</a>	No	"Logstash event for %{host}"
<a href="#">details</a>	<a href="#">hash</a>	No	{"timestamp"=>"%{@timestamp}", "message"=>"%{message}"}
<a href="#">enable_metrics</a>	<a href="#">boolean</a>	No	true
<a href="#">event_type</a>	<a href="#">string</a> , one of ["trigger", "acknowledge" , "resolve"]	No	"trigger"
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">incident_key</a>	<a href="#">string</a>	No	"logstash/%{host}/%{type}"
<a href="#">pdurl</a>	<a href="#">string</a>	No	"https://events.pagerduty.com/generic/2010-04-15/create_event.json"
<a href="#">service_key</a>	<a href="#">string</a>	Yes	

Setting	Input type	Required	Default value
<a href="#"><u>workers</u></a>	<>,>>	No	1

## Details

### *codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### *description*

- Value type is [string](#)
- Default value is "Logstash event for %{host}"

Custom description

### *details*

- Value type is [hash](#)
- Default value is {"timestamp"=>"%{@timestamp}", "message"=>"%{message}"}

The event details. These might be data from the Logstash event fields you wish to include. Tags are automatically included if detected so there is no need to explicitly add them here.

### *enable\_metric*

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### *event\_type*

- Value can be any of: trigger, acknowledge, resolve
- Default value is "trigger"

Event type

### *id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

### *incident\_key*

- Value type is [string](#)
- Default value is "`logstash/%{host}/%{type}`"

The service key to use. You'll need to set this up in PagerDuty beforehand.

### *pdurl*

- Value type is [string](#)
- Default value is "`https://events.pagerduty.com/generic/2010-04-15/create_event.json`"

PagerDuty API URL. You shouldn't need to change this, but is included to allow for flexibility should PagerDuty iterate the API and Logstash hasn't been updated yet.

### *service\_key*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The PagerDuty Service API Key

### *workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type. This is hacky, but it can only be here.

## pipe

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

Pipe output.

Pipe events to stdin of another program. You can use fields from the event as parts of the command. **WARNING:** This feature can cause logstash to fork off multiple children if you are not careful with per-event commandline.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
pipe {
 command => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">command</a>	<a href="#">string</a>	Yes	
<a href="#">enable metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">message format</a>	<a href="#">string</a>	No	
<a href="#">ttl</a>	<a href="#">number</a>	No	10
<a href="#">workers</a>	<<, >>	No	1

## Details

[codec](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### `command`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Command line to launch and pipe to

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

### `message_format`

- Value type is [string](#)
- There is no default value for this setting.

The format to use when writing events to the pipe. This value supports any string and can include `%{name}` and other dynamic strings.

If this setting is omitted, the full json representation of the event will be written as a single line.

### `ttl`

- Value type is [number](#)
- Default value is 10

Close pipe that hasn't been used for TTL seconds. -1 or 0 means never close.

*workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

## rabbitmq

- Version: 4.0.6
- Released on: 2017-01-17
- [Changelog](#)

encoding: UTF-8 Push events to a RabbitMQ exchange. Requires RabbitMQ 2.x or later version (3.x is recommended).

Relevant links:

- [RabbitMQ](#)
- [March Hare](#)

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
rabbitmq {
 exchange => ...
 exchange_type => ...
 host => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">arguments</a>	<a href="#">array</a>	No	{ }
<a href="#">automatic_recovery</a>	<a href="#">boolean</a>	No	true
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">connect_retry_interval</a>	<a href="#">number</a>	No	1
<a href="#">connection_timeout</a>	<a href="#">number</a>	No	
<a href="#">durable</a>	<a href="#">boolean</a>	No	true
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">exchange</a>	<a href="#">string</a>	Yes	
<a href="#">exchange_type</a>	<a href="#">string</a> , one of ["fanout", "direct", "topic", "x-consistent-hash", "x-modulus-hash"]	Yes	

Setting	Input type	Required	Default value
<a href="#">heartbeat</a>	<a href="#">number</a>	No	
<a href="#">host</a>	<a href="#">string</a>	Yes	
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">key</a>	<a href="#">string</a>	No	"logstash"
<a href="#">passive</a>	<a href="#">boolean</a>	No	false
<a href="#">password</a>	<a href="#">password</a>	No	"guest"
<a href="#">persistent</a>	<a href="#">boolean</a>	No	true
<a href="#">port</a>	<a href="#">number</a>	No	5672
<a href="#">ssl</a>	<a href="#">boolean</a>	No	
<a href="#">ssl_certificate_password</a>	<a href="#">string</a>	No	
<a href="#">ssl_certificate_path</a>	a valid filesystem path	No	
<a href="#">ssl_version</a>	<a href="#">string</a>	No	"TLSv1.2"
<a href="#">user</a>	<a href="#">string</a>	No	"guest"
<a href="#">vhost</a>	<a href="#">string</a>	No	"/"
<a href="#">workers</a>	<<,>>	No	1

## Details

### *arguments*

- Value type is [array](#)
- Default value is {}

Extra queue arguments as an array. To make a RabbitMQ queue mirrored, use: { "x-ha-policy" => "all" }

### *automatic\_recovery*

- Value type is [boolean](#)
- Default value is true

Set this to automatically recover from a broken connection. You almost certainly don't want to override this!!!

### *codec*

- Value type is [codec](#)

- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### `connect_retry_interval`

- Value type is [number](#)
- Default value is 1

Time in seconds to wait before retrying a connection

### `connection_timeout`

- Value type is [number](#)
- There is no default value for this setting.

The default connection timeout in milliseconds. If not specified the timeout is infinite.

### `debug (DEPRECATED)`

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [boolean](#)
- Default value is `false`

Enable or disable logging

### `durable`

- Value type is [boolean](#)
- Default value is `true`

Is this exchange durable? (aka; Should it survive a broker restart?)

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `exchange`

- This is a required setting.
- Value type is [string](#)

- There is no default value for this setting.

The name of the exchange

*exchange\_type*

- This is a required setting.
- Value can be any of: `fanout`, `direct`, `topic`, `x-consistent-hash`, `x-modulus-hash`
- There is no default value for this setting.

The exchange type (fanout, topic, direct)

*heartbeat*

- Value type is [number](#)
- There is no default value for this setting.

Heartbeat delay in seconds. If unspecified no heartbeats will be sent

*host*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

RabbitMQ server address(es) host can either be a single host, or a list of hosts i.e. host  $\Rightarrow$  "localhost" or host  $\Rightarrow$  ["host01", "host02"]

if multiple hosts are provided on the initial connection and any subsequent recovery attempts of the hosts is chosen at random and connected to. Note that only one host connection is active at a time.

*id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

### *key*

- Value type is [string](#)
- Default value is "logstash"

The default codec for this plugin is JSON. You can override this to suit your particular needs however. Key to route to by default. Defaults to *logstash*

- Routing keys are ignored on fanout exchanges.

### *passive*

- Value type is [boolean](#)
- Default value is `false`

Passive queue creation? Useful for checking queue existance without modifying server state

### *password*

- Value type is [password](#)
- Default value is "guest"

RabbitMQ password

### *persistent*

- Value type is [boolean](#)
- Default value is `true`

Should RabbitMQ persist messages to disk?

### *port*

- Value type is [number](#)
- Default value is 5672

RabbitMQ port to connect on

### *ssl*

- Value type is [boolean](#)
- There is no default value for this setting.

Enable or disable SSL. Note that by default remote certificate verification is off. Specify `ssl_certificate_path` and `ssl_certificate_password` if you need certificate verification

*ssl\_certificate\_password*

- Value type is [string](#)
- There is no default value for this setting.

Password for the encrypted PKCS12 (.p12) certificate file specified in `ssl_certificate_path`

*ssl\_certificate\_path*

- Value type is [path](#)
- There is no default value for this setting.

Path to an SSL certificate in PKCS12 (.p12) format used for verifying the remote host

*ssl\_version*

- Value type is [string](#)
- Default value is "TLSv1.2"

Version of the SSL protocol to use.

*tls\_certificate\_password (DEPRECATED)*

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [string](#)
- There is no default value for this setting.

TLS certificate password

*tls\_certificate\_path (DEPRECATED)*

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [path](#)
- There is no default value for this setting.

TLS certificate path

*user*

- Value type is [string](#)
- Default value is "guest"

RabbitMQ username

*vhost*

- Value type is [string](#)
- Default value is "/"

The vhost (virtual host) to use. If you don't know what this is, leave the default. With the exception of the default vhost ("/"), names of vhosts should not begin with a forward slash.

*workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

## rackspace

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-output-rackspace`.

### Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
rackspace {
 api_key => ...
 username => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
api_key	<a href="#">string</a>	Yes	
codec	<a href="#">codec</a>	No	"json"
queue	<a href="#">string</a>	No	"logstash"
region	<a href="#">string</a>	No	"dfw"
ttl	<a href="#">number</a>	No	360
username	<a href="#">string</a>	Yes	
workers	<a href="#">number</a>	No	1

### Details

#### *api\_key*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Rackspace Cloud API Key

### `codec`

- Value type is [codec](#)
- Default value is "json"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### `exclude_tags (DEPRECATED)`

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [array](#)
- Default value is []

Only handle events without any of these tags. Optional.

### `queue`

- Value type is [string](#)
- Default value is "logstash"

Rackspace Queue Name

### `region`

- Value type is [string](#)
- Default value is "dfw"

Rackspace region ord, dfw, lon, syd, etc

### `tags (DEPRECATED)`

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [array](#)
- Default value is []

Only handle events with all of these tags. Optional.

### `ttl`

- Value type is [number](#)
- Default value is 360

time for item to live in queue

*type* (*DEPRECATED*)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [string](#)
- Default value is ""

The type to act on. If a type is given, then this output will only act on messages with the same type. See any input plugin's `type` attribute for more. Optional.

*username*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Rackspace Cloud Username

*workers*

- Value type is [number](#)
- Default value is 1

The number of workers to use for this output. Note that this setting may not be useful for all outputs.

## redis

- Version: 3.0.3
- Released on: 2016-10-13
- [Changelog](#)

This output will send events to a Redis queue using RPUSH. The RPUSH command is supported in Redis v0.0.7+. Using PUBLISH to a channel requires at least v1.3.8+. While you may be able to make these Redis versions work, the best performance and stability will be found in more recent stable versions. Versions 2.6.0+ are recommended.

For more information, see [the Redis homepage](#)

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
redis {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">batch</a>	<a href="#">boolean</a>	No	false
<a href="#">batch_events</a>	<a href="#">number</a>	No	50
<a href="#">batch_timeout</a>	<a href="#">number</a>	No	5
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">congestion_interval</a>	<a href="#">number</a>	No	1
<a href="#">congestion_threshold</a>	<a href="#">number</a>	No	0
<a href="#">data_type</a>	<a href="#">string</a> , one of ["list", "channel"]	No	
<a href="#">db</a>	<a href="#">number</a>	No	0
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">host</a>	<a href="#">array</a>	No	["127.0.0.1"]
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">key</a>	<a href="#">string</a>	No	
<a href="#">password</a>	<a href="#">password</a>	No	

Setting	Input type	Required	Default value
<a href="#">port</a>	<a href="#">number</a>	No	6379
<a href="#">reconnect interval</a>	<a href="#">number</a>	No	1
<a href="#">shuffle hosts</a>	<a href="#">boolean</a>	No	true
<a href="#">timeout</a>	<a href="#">number</a>	No	5
<a href="#">workers</a>	<>	No	1

## Details

### *batch*

- Value type is [boolean](#)
- Default value is false

Set to true if you want Redis to batch up values and send 1 RPUSH command instead of one command per value to push on the list. Note that this only works with `data_type="list"` mode right now.

If true, we send an RPUSH every "batch\_events" events or "batch\_timeout" seconds (whichever comes first). Only supported for `data_type` is "list".

### *batch\_events*

- Value type is [number](#)
- Default value is 50

If batch is set to true, the number of events we queue up for an RPUSH.

### *batch\_timeout*

- Value type is [number](#)
- Default value is 5

If batch is set to true, the maximum amount of time between RPUSH commands when there are pending events to flush.

### *codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

*congestion\_interval*

- Value type is [number](#)
- Default value is 1

How often to check for congestion. Default is one second. Zero means to check on every event.

*congestion\_threshold*

- Value type is [number](#)
- Default value is 0

In case Redis `data_type` is `list` and has more than `@congestion_threshold` items, block until someone consumes them and reduces congestion, otherwise if there are no consumers Redis will run out of memory, unless it was configured with OOM protection. But even with OOM protection, a single Redis list can block all other users of Redis, until Redis CPU consumption reaches the max allowed RAM size. A default value of 0 means that this limit is disabled. Only supported for `list` Redis `data_type`.

*data\_type*

- Value can be any of: `list`, `channel`
- There is no default value for this setting.

Either `list` or `channel`. If `redis_type` is `list`, then we will set `RPUSH` to `key`. If `redis_type` is `channel`, then we will `PUBLISH` to `key`.

*db*

- Value type is [number](#)
- Default value is 0

The Redis database number.

*enable\_metric*

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### *host*

- Value type is [array](#)
- Default value is `["127.0.0.1"]`

The hostname(s) of your Redis server(s). Ports may be specified on any hostname, which will override the global port config. If the hosts list is an array, Logstash will pick one random host to connect to, if that host is disconnected it will then pick another.

For example:

```
"127.0.0.1"
["127.0.0.1", "127.0.0.2"]
["127.0.0.1:6380", "127.0.0.1"]
id
```

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `id` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
key
```

- Value type is [string](#)
- There is no default value for this setting.

The name of a Redis list or channel. Dynamic names are valid here, for example `logstash-%{type}`.

### *name (DEPRECATED)*

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [string](#)
- Default value is `"default"`

Name is used for logging in case there are multiple instances.

### *password*

- Value type is [password](#)
- There is no default value for this setting.

Password to authenticate with. There is no authentication by default.

### *port*

- Value type is [number](#)
- Default value is 6379

The default port to connect on. Can be overridden on any hostname.

### *queue (DEPRECATED)*

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [string](#)
- There is no default value for this setting.

The name of the Redis queue (we'll use RPUSH on this). Dynamic names are valid here, for example `logstash-%{type}`

### *reconnect\_interval*

- Value type is [number](#)
- Default value is 1

Interval for reconnecting to failed Redis connections

### *shuffle\_hosts*

- Value type is [boolean](#)
- Default value is `true`

Shuffle the host list during Logstash startup.

### *timeout*

- Value type is [number](#)
- Default value is 5

Redis initial connection timeout in seconds.

*workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

## redmine

- Version: 3.0.0
- Released on: 2016-09-10
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
`bin/logstash-plugin install logstash-output-redmine.`

The redmine output is used to create a ticket via the API redmine.

It send a POST request in a JSON format and use TOKEN authentication

— Exemple of use --

```
output {
 redmine {
 url => "http://redmineserver.tld"
 token => 'token'
 project_id => 200
 tracker_id => 1
 status_id => 3
 priority_id => 2
 subject => "Error ... detected"
 }
}
```

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
redmine {
 priority_id => ...
 project_id => ...
 status_id => ...
 token => ...
 tracker_id => ...
 url => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>assigned_to_id</code>	<a href="#">number</a>	No	<code>nil</code>

Setting	Input type	Required	Default value
categorie_id	<a href="#">number</a>	No	nil
codec	<a href="#">codec</a>	No	"plain"
description	<a href="#">string</a>	No	"%{message}"
enable_metric	<a href="#">boolean</a>	No	true
fixed_version_id	<a href="#">number</a>	No	nil
id	<a href="#">string</a>	No	
parent_issue_id	<a href="#">number</a>	No	nil
priority_id	<a href="#">number</a>	Yes	
project_id	<a href="#">number</a>	Yes	
ssl	<a href="#">boolean</a>	No	false
status_id	<a href="#">number</a>	Yes	
subject	<a href="#">string</a>	No	"%{host}"
token	<a href="#">string</a>	Yes	
tracker_id	<a href="#">number</a>	Yes	
url	<a href="#">string</a>	Yes	
workers	<<,>>	No	1

## Details

*assigned\_to\_id*

- Value type is [number](#)
- Default value is [nil](#)

redmine issue assigned\_to not required for post\_issue

*categorie\_id*

- Value type is [number](#)
- Default value is [nil](#)

not required for post\_issue

*codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

*description*

- Value type is [string](#)
- Default value is "%{message}"

redmine issue description required

*enable\_metric*

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

*fixed\_version\_id*

- Value type is [number](#)
- Default value is `nil`

redmine issue fixed\_version\_id

*id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique ID to the plugin instance, this ID is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

parent_issue_id

- Value type is [number](#)
- Default value is `nil`

redmine issue parent_issue_id not required for post_issue

priority_id

- This is a required setting.

- Value type is [number](#)
- There is no default value for this setting.

redmine issue priority_id required

project_id

- This is a required setting.
- Value type is [number](#)
- There is no default value for this setting.

redmine issue projet_id required

ssl

- Value type is [boolean](#)
- Default value is `false`

status_id

- This is a required setting.
- Value type is [number](#)
- There is no default value for this setting.

redmine issue status_id required

subject

- Value type is [string](#)
- Default value is "`%{host}`"

redmine issue subject required

token

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

redmine token user used for authentication

tracker_id

- This is a required setting.
- Value type is [number](#)
- There is no default value for this setting.

redmine issue tracker_id required

url

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

host of redmine app value format : *http://urlofredmine.tld* - Not add /issues at end

workers

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

riak

- Version: 3.0.0
- Released on: 2016-09-10
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
bin/logstash-plugin install logstash-output-riak.

Riak is a distributed k/v store from Basho. It's based on the Dynamo model.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
riak {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
bucket	array	No	["logstash-%{+YYYY.MM.dd}"]
bucket_props	hash	No	
codec	codec	No	"plain"
enable_metric	boolean	No	true
enable_search	boolean	No	false
enable_ssl	boolean	No	false
id	string	No	
indices	array	No	
key_name	string	No	
nodes	hash	No	{"localhost"=>"8098"}
proto	string , one of ["http", "pb"]	No	"http"
ssl_opts	hash	No	
workers	<<,>>	No	1

Details

bucket

- Value type is [array](#)
- Default value is `["logstash-%{+YYYY.MM.dd}"]`

The bucket name to write events to Expansion is supported here as values are passed through event.sprintf Multiple buckets can be specified here but any bucket-specific settings defined apply to ALL the buckets.

bucket_props

- Value type is [hash](#)
- There is no default value for this setting.

Bucket properties (NYI) Logstash hash of properties for the bucket i.e.

```
bucket_props => {
    "r" => "one"
    "w" => "one"
    "dw", "one
}
or
bucket_props => { "n_val" => "3" }
Properties will be passed as-is
```

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

enable_metric

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

enable_search

- Value type is [boolean](#)
- Default value is `false`

Search Enable search on the bucket defined above

`enable_ssl`

- Value type is [boolean](#)
- Default value is `false`

SSL Enable SSL

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

### `indices`

- Value type is [array](#)
- There is no default value for this setting.

Indices Array of fields to add 2i on e.g.

```
`indices => ["source_host", "type"]
Off by default as not everyone runs leveldb
key_name
```

- Value type is [string](#)
- There is no default value for this setting.

The event key name variables are valid here.

Choose this carefully. Best to let riak decide.

### `nodes`

- Value type is [hash](#)
- Default value is `{"localhost"=>"8098"}`

The nodes of your Riak cluster This can be a single host or a Logstash hash of node/port pairs e.g

```
{
 "node1" => "8098"
 "node2" => "8098"
}
```

*proto*

- Value can be any of: http, pb
- Default value is "http"

The protocol to use HTTP or ProtoBuf Applies to ALL backends listed above No mix and match

*ssl\_opts*

- Value type is [hash](#)
- There is no default value for this setting.

SSL Options Options for SSL connections Only applied if SSL is enabled Logstash hash that maps to the riak-client options here: <https://github.com/basho/riak-ruby-client/wiki/Connecting-to-Riak> You'll likely want something like this:

```
ssl_opts => {
 "pem" => "/etc/riak.pem"
 "ca_path" => "/usr/share/certificates"
}
```

Per the riak client docs, the above sample options will turn on SSL VERIFY\_PEER

*workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

## riemann

- Version: 3.0.0
- Released on: 2016-11-11
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
`bin/logstash-plugin install logstash-output-riemann`.

Riemann is a network event stream processing system.

While Riemann is very similar conceptually to Logstash, it has much more in terms of being a monitoring system replacement.

Riemann is used in Logstash much like statsd or other metric-related outputs

You can learn about Riemann here:

- <http://riemann.io/> You can see the author talk about it here:
- <http://vimeo.com/38377415>

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
riemann {
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	<a href="#">codec</a>	No	"plain"
debug	<a href="#">boolean</a>	No	false
enable_metric	<a href="#">boolean</a>	No	true
host	<a href="#">string</a>	No	"localhost"
id	<a href="#">string</a>	No	
map_fields	<a href="#">boolean</a>	No	false
port	<a href="#">number</a>	No	5555

Setting	Input type	Required	Default value
protocol	<a href="#">string</a> , one of ["tcp", "udp"]	No	"tcp"
riemann_event	<a href="#">hash</a>	No	
sender	<a href="#">string</a>	No	"%{host}"
workers	<>, >>	No	1

## Details

### *codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### *debug*

- Value type is [boolean](#)
- Default value is `false`

Enable debugging output?

### *enable\_metric*

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### *host*

- Value type is [string](#)
- Default value is "localhost"

The address of the Riemann server.

### *id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

map_fields

- Value type is [boolean](#)
- Default value is `false`

If set to true automatically map all logstash defined fields to riemann event fields. All nested logstash fields will be mapped to riemann fields containing all parent keys separated by dots and the deepest value.

As an example, the logstash event:

```
{  
    "@timestamp": "2013-12-10T14:36:26.151+0000",  
    "@version": 1,  
    "message": "log message",  
    "host": "host.domain.com",  
    "nested_field": {  
        "key": "value"  
    }  
}  
Is mapped to this riemann event:  
{  
    :time 1386686186,  
    :host host.domain.com,  
    :message log message,  
    :nested_field.key value  
}
```

It can be used in conjunction with or independent of the `riemann_event` option. When used with the `riemann_event` any duplicate keys receive their value from `riemann_event` instead of the logstash event itself.

port

- Value type is [number](#)
- Default value is `5555`

The port to connect to on your Riemann server.

protocol

- Value can be any of: `tcp`, `udp`
- Default value is `"tcp"`

The protocol to use UDP is non-blocking TCP is blocking

Logstash's default output behaviour is to never lose events As such, we use tcp as default here

riemann_event

- Value type is [hash](#)
- There is no default value for this setting.

A Hash to set Riemann event fields (<http://riemann.io/concepts.html>).

The following event fields are supported: `description`, `state`, `metric`, `ttl`, `service`

Tags found on the Logstash event will automatically be added to the Riemann event.

Any other field set here will be passed to Riemann as an event attribute.

Example:

```
riemann {  
    riemann_event => {  
        "metric"  => "%{metric}"  
        "service" => "%{service}"  
    }  
}
```

`metric` and `ttl` values will be coerced to a floating point value. Values which cannot be coerced will zero (0.0).

`description`, by default, will be set to the event message but can be overridden here.

sender

- Value type is [string](#)
- Default value is "`%{host}`"

The name of the sender. This sets the `host` value in the Riemann event

workers

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the `:legacy` type This is hacky, but it can only be herne

s3

- Version: 4.0.5
- Released on: 2017-01-18
- [Changelog](#)

INFORMATION:

This plugin batches and uploads logstash events into Amazon Simple Storage Service (Amazon S3).

Requirements: * Amazon S3 Bucket and S3 Access Permissions (Typically access_key_id and secret_access_key) * S3 PutObject permission

S3 outputs create temporary files into the OS' temporary directory, you can specify where to save them using the `temporary_directory` option.

S3 output files have the following format

`ls.s3.ip-10-228-27-95.2013-04-18T10.00.tag_hello.part0.txt`

ls.s3	indicate logstash plugin s3
ip-10-228-27-95	indicates the ip of your machine.
2013-04-18T10.00	represents the time whenever you specify <code>time_file</code> .
tag_hello	this indicates the event's tag.
part0	this means if you indicate <code>size_file</code> then it will generate more parts if you <code>file.size > size_file</code> . When a file is full it will be pushed to the bucket and then deleted from the temporary directory. If a file is empty, it is simply deleted. Empty files will not be pushed

Crash Recovery: * This plugin will recover and upload temporary log files after crash/abnormal termination when using `restore` set to true

Usage: This is an example of logstash config:

```
output {
  s3{
    access_key_id => "crazy_key"           (required)
    secret_access_key => "monkey_access_key" (required)
    region => "eu-west-1"                  (optional, default = "us-east-1")
    bucket => "your_bucket"                (required)
    size_file => 2048                      (optional) - Bytes
    time_file => 5                         (optional) - Minutes
  }
}
```

```

        format => "plain"                      (optional)
        canned_acl => "private"                 (optional. Options are
"private", "public_read", "public_read_write", "authenticated_read". Defaults
to "private" )
    }

```

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
s3 {
    bucket => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
access_key_id	string	No	
aws_credentials_file	string	No	
bucket	string	Yes	
canned_acl	string , one of ["private", "public_read", " ", "public_read_write", "authenticated_read"]	No	"private"
codec	codec	No	"plain"
enable_metric	boolean	No	true
encoding	string , one of ["none", "gzip"]	No	"none"
id	string	No	
prefix	string	No	""
proxy_uri	string	No	
region	string , one of ["us-east-1", "us-west-1",	No	"us-east-1"

Setting	Input type	Required	Default value
	"us-west-2", "eu-central-1", "eu-west-1", "ap-southeast-1", "ap-southeast-2", "ap-northeast-1", "ap-northeast-2", "sa-east-1", "us-gov-west-1", "cn-north-1", "ap-south-1"]		
restore	boolean	No	true
rotation_strategy	string , one of ["size_and_time", "size", "time"]	No	"size_and_time"
secret_access_key	string	No	
server_side_encryption	boolean	No	false
server_side_encryption_algorithm	string , one of ["AES256", "aws:kms"]	No	"AES256"
session_token	string	No	
signature_version	string , one of ["v2", "v4"]	No	
size_file	number	No	5242880
ssekms_key_id	string	No	
storage_class	string , one of ["STANDARD", "REDUCED_REDUNDANCY", "STANDARD_IA"]	No	"STANDARD"
tags	array	No	[]
temporary_directory	string	No	"/var/folders/_9/x4bq65rs6vd0rrjthct3zxjw0000gn/T/logstash"

Setting	Input type	Required	Default value
time_file	number	No	15
upload_queue_size	number	No	4
upload_workers_count	number	No	4
validate_credentials_on_root_bucket	boolean	No	true
workers	<<,>>	No	1

Details

[access_key_id](#)

- Value type is [string](#)
- There is no default value for this setting.

This plugin uses the AWS SDK and supports several ways to get credentials, which will be tried in this order:

1. Static configuration, using `access_key_id` and `secret_access_key` params in logstash plugin config
2. External credentials file specified by `aws_credentials_file`
3. Environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`
4. Environment variables `AMAZON_ACCESS_KEY_ID` and `AMAZON_SECRET_ACCESS_KEY`
5. IAM Instance Profile (available when running inside EC2)

[aws_credentials_file](#)

- Value type is [string](#)
- There is no default value for this setting.

Path to YAML file containing a hash of AWS credentials. This file will only be loaded if `access_key_id` and `secret_access_key` aren't set. The contents of the file should look like this:

```
:access_key_id: "12345"
:secret_access_key: "54321"
```

[bucket](#)

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

S3 bucket

`canned_acl`

- Value can be any of: `private`, `public_read`, `public_read_write`, `authenticated_read`
- Default value is "private"

The S3 canned ACL to use when putting the file. Defaults to "private".

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`encoding`

- Value can be any of: `none`, `gzip`
- Default value is "none"

Specify the content encoding. Supports ("gzip"). Defaults to "none"

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
  stdout {
    id => "my_plugin_id"
  }
}
```

prefix

- Value type is [string](#)
- Default value is ""

Specify a prefix to the uploaded filename, this can simulate directories on S3. Prefix does not require leading slash. This option support string interpolation, be warned this can created a lot of temporary local files.

proxy_uri

- Value type is [string](#)
- There is no default value for this setting.

URI to proxy server if required

region

- Value can be any of: us-east-1, us-west-1, us-west-2, eu-central-1, eu-west-1, ap-southeast-1, ap-southeast-2, ap-northeast-1, ap-northeast-2, sa-east-1, us-gov-west-1, cn-north-1, ap-south-1
- Default value is "us-east-1"

The AWS Region

restore

- Value type is [boolean](#)
- Default value is true

rotation_strategy

- Value can be any of: size_and_time, size, time
- Default value is "size_and_time"

Define the strategy to use to decide when we need to rotate the file and push it to S3, The default strategy is to check for both size and time, the first one to match will rotate the file.

secret_access_key

- Value type is [string](#)
- There is no default value for this setting.

The AWS Secret Access Key

server_side_encryption

- Value type is [boolean](#)
- Default value is `false`

Specifies whether or not to use S3's server side encryption. Defaults to no encryption.

server_side_encryption_algorithm

- Value can be any of: `AES256`, `aws:kms`
- Default value is "`AES256`"

Specifies what type of encryption to use when SSE is enabled.

session_token

- Value type is [string](#)
- There is no default value for this setting.

The AWS Session token for temporary credential

signature_version

- Value can be any of: `v2`, `v4`
- There is no default value for this setting.

The version of the S3 signature hash to use. Normally uses the internal client default, can be explicitly specified here

size_file

- Value type is [number](#)
- Default value is `5242880`

Set the size of file in bytes, this means that files on bucket when have dimension > file_size, they are stored in two or more file. If you have tags then it will generate a specific size file for every tags

ssekms_key_id

- Value type is [string](#)
- There is no default value for this setting.

The key to use when specified along with `server_side_encryption` ⇒ `aws:kms`. If `server_side_encryption` ⇒ `aws:kms` is set but this is not default KMS key is used.
<http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingKMSEncryption.html>

storage_class

- Value can be any of: STANDARD, REDUCED_REDUNDANCY, STANDARD_IA
- Default value is "STANDARD"

Specifies what S3 storage class to use when uploading the file. More information about the different storage classes can be found:

<http://docs.aws.amazon.com/AmazonS3/latest/dev/storage-class-intro.html> Defaults to STANDARD.

tags

- Value type is [array](#)
- Default value is []

Define tags to be appended to the file on the S3 bucket.

Example: tags ⇒ ["elasticsearch", "logstash", "kibana"]

Will generate this file: "ls.s3.logstash.local.2015-01-01T00.00.tag_elasticsearch.logstash.kibana.part0.txt"

temporary_directory

- Value type is [string](#)
- Default value is "/var/folders/_9/x4bq65rs6vd0rrjthct3zxjw0000gn/T/logstash"

Set the directory where logstash will store the tmp files before sending it to S3 default to the current OS temporary directory in linux /tmp/logstash

time_file

- Value type is [number](#)
- Default value is 15

Set the time, in MINUTES, to close the current sub_time_section of bucket. If you define file_size you have a number of files in consideration of the section and the current tag. 0 stay all time on listerner, beware if you specific 0 and size_file 0, because you will not put the file on bucket, for now the only thing this plugin can do is to put the file when logstash restart.

upload_queue_size

- Value type is [number](#)
- Default value is 4

Number of items we can keep in the local queue before uploading them

upload_workers_count

- Value type is [number](#)
- Default value is 4

Specify how many workers to use to upload the files to S3

validate_credentials_on_root_bucket

- Value type is [boolean](#)
- Default value is true

The common use case is to define permission on the root bucket and give Logstash full access to write its logs. In some circumstances you need finer grained permission on subfolder, this allow you to disable the check at startup.

workers

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

sns

- Version: 4.0.2
- Released on: 2016-07-14
- [Changelog](#)

SNS output.

Send events to Amazon's Simple Notification Service, a hosted pub/sub framework. It supports various subscription types, including email, HTTP/S, SMS, and SQS.

For further documentation about the service see:

<http://docs.amazonwebservices.com/sns/latest/api/>

This plugin looks for the following fields on events it receives:

- `sns` - If no ARN is found in the configuration file, this will be used as the ARN to publish.
- `sns_subject` - The subject line that should be used. will be truncated to 100 characters. If `sns_subject` is set to a non-string value a JSON version of that value will be saved.
- `sns_message` - Optional string of message to be sent. If this is set to a non-string value it will be encoded with the specified `codec`. If this is not set the entire event will be encoded with the `codec`. with the @message truncated so that the length of the JSON fits in 32768 bytes.

Upgrading to 2.0.0

This plugin used to have a `format` option for controlling the encoding of messages prior to being sent to SNS. This plugin now uses the logstash standard `codec` option for encoding instead. If you want the same *plain* format as the v0/1 codec (`format => "plain"`) use `codec => "s3_plain"`.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
sns {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
access_key_id	string	No	
arn	string	No	
aws_credentials_file	string	No	
codec	codec	No	"plain"
enable_metric	boolean	No	true
id	string	No	
proxy_uri	string	No	
publish_boot_message_arn	string	No	
region	string, one of ["us-east-1", "us-west-1", "us-west-2", "eu-central-1", "eu-west-1", "ap-southeast-1", "ap-southeast-2", "ap-northeast-1", "ap-northeast-2", "sa-east-1", "us-gov-west-1", "cn-north-1", "ap-south-1"]	No	"us-east-1"
secret_access_key	string	No	
session_token	string	No	
workers	<<,>>	No	1

Details

[access_key_id](#)

- Value type is [string](#)
- There is no default value for this setting.

This plugin uses the AWS SDK and supports several ways to get credentials, which will be tried in this order:

1. Static configuration, using `access_key_id` and `secret_access_key` params in logstash plugin config
2. External credentials file specified by `aws_credentials_file`
3. Environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`
4. Environment variables `AMAZON_ACCESS_KEY_ID` and `AMAZON_SECRET_ACCESS_KEY`
5. IAM Instance Profile (available when running inside EC2)

[arn](#)

- Value type is [string](#)

- There is no default value for this setting.

Optional ARN to send messages to. If you do not set this you must include the `sns` field in your events to set the ARN on a per-message basis!

`aws_credentials_file`

- Value type is [string](#)
- There is no default value for this setting.

Path to YAML file containing a hash of AWS credentials. This file will only be loaded if `access_key_id` and `secret_access_key` aren't set. The contents of the file should look like this:

```
:access_key_id: "12345"  
:secret_access_key: "54321"
```

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
  stdout {  
    id => "my_plugin_id"  
  }  
}
```

proxy_uri

- Value type is [string](#)
- There is no default value for this setting.

URI to proxy server if required

publish_boot_message_arn

- Value type is [string](#)
- There is no default value for this setting.

When an ARN for an SNS topic is specified here, the message "Logstash successfully booted" will be sent to it when this plugin is registered.

Example: arn:aws:sns:us-east-1:770975001275:logstash-testing

region

- Value can be any of: us-east-1, us-west-1, us-west-2, eu-central-1, eu-west-1, ap-southeast-1, ap-southeast-2, ap-northeast-1, ap-northeast-2, sa-east-1, us-gov-west-1, cn-north-1, ap-south-1
- Default value is "us-east-1"

The AWS Region

secret_access_key

- Value type is [string](#)
- There is no default value for this setting.

The AWS Secret Access Key

session_token

- Value type is [string](#)
- There is no default value for this setting.

The AWS Session token for temporary credential

workers

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

solr_http

- Version: 3.0.1
- Released on: 2016-07-14
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
`bin/logstash-plugin install logstash-output-solr_http.`

This output lets you index&store your logs in Solr. If you want to get started quickly you should use version 4.4 or above in schemaless mode, which will try and guess your fields automatically. To turn that on, you can use the example included in the Solr archive:

```
tar zxf solr-4.4.0.tgz
cd example
mv solr solr_ #back up the existing sample conf
cp -r example-schemaless/solr/ . #put the schemaless conf in place
java -jar start.jar #start Solr
```

You can learn more at [the Solr home page](#)

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
solr_http {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	codec	No	"plain"
document_id	string	No	nil
enable_metric	boolean	No	true
flush_size	number	No	100
id	string	No	
idle_flush_time	number	No	1
solr_url	string	No	"http://localhost:8983/solr"
workers	<<, >>	No	1

Details

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

`document_id`

- Value type is [string](#)
- Default value is `nil`

Solr document ID for events. You'd typically have a variable here, like

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`flush_size`

- Value type is [number](#)
- Default value is 100

Number of events to queue up before writing to Solr

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

*idle\_flush\_time*

- Value type is [number](#)
- Default value is 1

Amount of time since the last flush before a flush is done even if the number of buffered events is smaller than flush\_size

*solr\_url*

- Value type is [string](#)
- Default value is "http://localhost:8983/solr"

URL used to connect to Solr

*workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

## sqS

- Version: 4.0.0
- Released on: 2017-01-10
- [Changelog](#)

Forcibly load all modules marked to be lazily loaded.

It is recommended that this is called prior to launching threads. See <https://aws.amazon.com/blogs/developer/threading-with-the-aws-sdk-for-ruby/>. Push events to an Amazon Web Services (AWS) Simple Queue Service (SQS) queue.

SQS is a simple, scalable queue system that is part of the Amazon Web Services suite of tools. Although SQS is similar to other queuing systems such as Advanced Message Queuing Protocol (AMQP), it uses a custom API and requires that you have an AWS account. See <http://aws.amazon.com/sqs/> for more details on how SQS works, what the pricing schedule looks like and how to setup a queue.

The "consumer" identity must have the following permissions on the queue:

- sqs:GetQueueUrl
- sqs:SendMessage
- sqs:SendMessageBatch

Typically, you should setup an IAM policy, create a user and apply the IAM policy to the user. See <http://aws.amazon.com/iam/> for more details on setting up AWS identities. A sample policy is as follows:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "sns:Publish"
],
 "Resource": "arn:aws:sns:us-east-1:123456789012:my-sns-topic"
 }
]
}
```

## Batch Publishing

This output publishes messages to SQS in batches in order to optimize event throughput and increase performance. This is done using the [SendMessageBatch](<http://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference>) API.

[ce/API\\_SendMessageBatch.html](#)) API. When publishing messages to SQS in batches, the following service limits must be respected (see [Limits in Amazon SQS](<http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/limits-messages.html>)):

- The maximum allowed individual message size is 256KiB.
- The maximum total payload size (i.e. the sum of the sizes of all individual messages within a batch) is also 256KiB.

This plugin will dynamically adjust the size of the batch published to SQS in order to ensure that the total payload size does not exceed 256KiB.

This output cannot currently handle messages larger than 256KiB. Any single message exceeding this size will be dropped.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
sqsh {
 queue => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
access_key_id	<a href="#">string</a>	No	
aws_credentials_file	<a href="#">string</a>	No	
batch_events	<a href="#">number</a>	No	10
codec	<a href="#">codec</a>	No	"plain"
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	
message_max_size	<a href="#">bytes</a>	No	"256KiB"
proxy_uri	<a href="#">string</a>	No	
queue	<a href="#">string</a>	Yes	

Setting	Input type	Required	Default value
region	<a href="#">string</a> , one of ["us-east-1", "us-west-1", "us-west-2", "eu-central-1", "eu-west-1", "ap-southeast-1", "ap-southeast-2", "ap-northeast-1", "ap-northeast-2", "sa-east-1", "us-gov-west-1", "cn-north-1", "ap-south-1"]	No	"us-east-1"
secret_access_key	<a href="#">string</a>	No	
session_token	<a href="#">string</a>	No	
workers	<>, >>	No	1

## Details

### [access\\_key\\_id](#)

- Value type is [string](#)
- There is no default value for this setting.

This plugin uses the AWS SDK and supports several ways to get credentials, which will be tried in this order:

1. Static configuration, using `access_key_id` and `secret_access_key` params in logstash plugin config
2. External credentials file specified by `aws_credentials_file`
3. Environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`
4. Environment variables `AMAZON_ACCESS_KEY_ID` and `AMAZON_SECRET_ACCESS_KEY`
5. IAM Instance Profile (available when running inside EC2)

### [aws\\_credentials\\_file](#)

- Value type is [string](#)
- There is no default value for this setting.

Path to YAML file containing a hash of AWS credentials. This file will only be loaded if `access_key_id` and `secret_access_key` aren't set. The contents of the file should look like this:

```
:access_key_id: "12345"
:secret_access_key: "54321"
```

### *batch (DEPRECATED)*

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [boolean](#)
- Default value is `true`

Set to `true` to send messages to SQS in batches (with the `SendMessageBatch` API) or `false` to send messages to SQS individually (with the `SendMessage` API). The size of the batch is configurable via `batch_events`.

### *batch\_events*

- Value type is [number](#)
- Default value is `10`

The number of events to be sent in each batch. Set this to `1` to disable the batch sending of messages.

### *batch\_timeout (DEPRECATED)*

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [number](#)
- There is no default value for this setting.

### *codec*

- Value type is [codec](#)
- Default value is `"plain"`

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### *enable\_metric*

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### *id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}

message_max_size
```

- Value type is [bytes](#)
- Default value is "256KiB"

The maximum number of bytes for any message sent to SQS. Messages exceeding this size will be dropped. See

<http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/limits-messages.html>.

*proxy\_uri*

- Value type is [string](#)
- There is no default value for this setting.

URI to proxy server if required

*queue*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The name of the target SQS queue. Note that this is just the name of the queue, not the URL or ARN.

*region*

- Value can be any of: us-east-1, us-west-1, us-west-2, eu-central-1, eu-west-1, ap-southeast-1, ap-southeast-2, ap-northeast-1, ap-northeast-2, sa-east-1, us-gov-west-1, cn-north-1, ap-south-1
- Default value is "us-east-1"

The AWS Region

*secret\_access\_key*

- Value type is [string](#)

- There is no default value for this setting.

### The AWS Secret Access Key

*session\_token*

- Value type is [string](#)
- There is no default value for this setting.

### The AWS Session token for temporary credential

*workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

## statsd

- Version: 3.1.1
- Released on: 2016-07-14
- [Changelog](#)

statsd is a network daemon for aggregating statistics, such as counters and timers, and shipping over UDP to backend services, such as Graphite or Datadog. The general idea is that you send metrics to statsd and every few seconds it will emit the aggregated values to the backend.

Example aggregates are sums, average and maximum values, their standard deviation, etc. This plugin makes it easy to send such metrics based on data in Logstash events.

You can learn about statsd here:

- [Etsy blog post announcing statsd](#)
- [statsd on github](#)

Typical examples of how this can be used with Logstash include counting HTTP hits by response code, summing the total number of bytes of traffic served, and tracking the 50th and 95th percentile of the processing time of requests.

Each metric emitted to statsd has a dot-separated path, a type, and a value. The metric path is built from the `namespace` and `sender` options together with the metric name that's picked up depending on the type of metric. All in all, the metric path will follow this pattern:

```
namespace.sender.metric
```

With regards to this plugin, the default namespace is "logstash", the default sender is the `host` field, and the metric name depends on what is set as the metric name in the `increment`, `decrement`, `timing`, `count`, `set` or `gauge` options. In metric paths, colons (":"), pipes ("|") and at signs ("@") are reserved and will be replaced by underscores ("\_").

Example:

```
output {
 statsd {
 host => "statsd.example.org"
 count => {
 "http.bytes" => "%{bytes}"
 }
 }
}
```

If run on a host named hal9000 the configuration above will send the following metric to statsd if the current event has 123 in its `bytes` field:

```
logstash.hal9000.http.bytes:123|c
```

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
statsd {
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	<a href="#">codec</a>	No	"plain"
count	<a href="#">hash</a>	No	{ }
decrement	<a href="#">array</a>	No	[]
enable_metric	<a href="#">boolean</a>	No	true
gauge	<a href="#">hash</a>	No	{ }
host	<a href="#">string</a>	No	"localhost"
id	<a href="#">string</a>	No	
increment	<a href="#">array</a>	No	[]
namespace	<a href="#">string</a>	No	"logstash"
port	<a href="#">number</a>	No	8125
sample_rate	<a href="#">number</a>	No	1
sender	<a href="#">string</a>	No	"%{host}"
set	<a href="#">hash</a>	No	{ }
timing	<a href="#">hash</a>	No	{ }
workers	<>,>>	No	1

## Details

### [codec](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### *count*

- Value type is [hash](#)
- Default value is {}

A count metric. `metric_name => count` as hash. `%{fieldname}` substitutions are allowed in the metric names.

### *decrement*

- Value type is [array](#)
- Default value is []

A decrement metric. Metric names as array. `%{fieldname}` substitutions are allowed in the metric names.

### *enable\_metric*

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### *gauge*

- Value type is [hash](#)
- Default value is {}

A gauge metric. `metric_name => gauge` as hash. `%{fieldname}` substitutions are allowed in the metric names.

### *host*

- Value type is [string](#)
- Default value is "localhost"

The hostname or IP address of the statsd server.

### *id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique ID to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when

you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
increment
```

- Value type is [array](#)
- Default value is []

An increment metric. Metric names as array. `%{fieldname}` substitutions are allowed in the metric names.

*namespace*

- Value type is [string](#)
- Default value is "logstash"

The statsd namespace to use for this metric. `%{fieldname}` substitutions are allowed.

*port*

- Value type is [number](#)
- Default value is 8125

The port to connect to on your statsd server.

*sample\_rate*

- Value type is [number](#)
- Default value is 1

The sample rate for the metric.

*sender*

- Value type is [string](#)
- Default value is "%{host}"

The name of the sender. Dots will be replaced with underscores. `%{fieldname}` substitutions are allowed.

*set*

- Value type is [hash](#)

- Default value is {}

A set metric.`metric_name => "string"` to append as hash. `%{fieldname}` substitutions are allowed in the metric names.

*timing*

- Value type is [hash](#)
- Default value is {}

A timing metric.`metric_name => duration` as hash. `%{fieldname}` substitutions are allowed in the metric names.

*workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

## stdout

- Version: 3.1.0
- Released on: 2016-08-22
- [Changelog](#)

A simple output which prints to the STDOUT of the shell running Logstash. This output can be quite convenient when debugging plugin configurations, by allowing instant access to the event data after it has passed through the inputs and filters.

For example, the following output configuration, in conjunction with the Logstash -e command-line flag, will allow you to see the results of your event pipeline for quick iteration.

```
output {
 stdout { }
}
```

Useful codecs include:

`rubydebug`: outputs event data using the ruby "awesome\_print" [library](#)

```
output {
 stdout { codec => rubydebug }
}
```

`json`: outputs event data in structured JSON format

```
output {
 stdout { codec => json }
}
```

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
stdout { }
```

Available configuration options:

Setting	Input type	Required	Default value
codec	<a href="#">codec</a>	No	"plain"

Setting	Input type	Required	Default value
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	
workers	<code>&lt;&lt;, &gt;&gt;</code>	No	1

## Details

### `codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

### `workers`

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

## stomp

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-output-stomp`.

### Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
stomp {
 destination => ...
 host => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	<a href="#">codec</a>	No	"plain"
debug	<a href="#">boolean</a>	No	false
destination	<a href="#">string</a>	Yes	
host	<a href="#">string</a>	Yes	
password	<a href="#">password</a>	No	""
port	<a href="#">number</a>	No	61613
user	<a href="#">string</a>	No	""
vhost	<a href="#">string</a>	No	nil
workers	<a href="#">number</a>	No	1

### Details

#### `codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

*debug*

- Value type is [boolean](#)
- Default value is `false`

Enable debugging output?

*destination*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The destination to read events from. Supports string expansion, meaning `%{foo}` values will expand to the field value.

Example: `"/topic/logstash"`

*host*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The address of the STOMP server.

*password*

- Value type is [password](#)
- Default value is `""`

The password to authenticate with.

*port*

- Value type is [number](#)
- Default value is `61613`

The port to connect to on your STOMP server.

*user*

- Value type is [string](#)
- Default value is `""`

The username to authenticate with.

*vhost*

- Value type is [string](#)
- Default value is `nil`

The vhost to use

*workers*

- Value type is [number](#)
- Default value is `1`

The number of workers to use for this output. Note that this setting may not be useful for all outputs.

## syslog

- Version: 3.0.1
- Released on: 2016-07-14
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
`bin/logstash-plugin install logstash-output-syslog`.

Send events to a syslog server.

You can send messages compliant with RFC3164 or RFC5424 using either UDP or TCP as the transport protocol.

By default the contents of the `message` field will be shipped as the free-form message text part of the emitted syslog message. If your messages don't have a `message` field or if you for some other reason want to change the emitted message, modify the `message` configuration option.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
syslog {
 host => ...
 port => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
appname	<a href="#">string</a>	No	"LOGSTASH"
codec	<a href="#">codec</a>	No	"plain"
enable_metric	<a href="#">boolean</a>	No	true
facility	<a href="#">string</a>	No	"user-level"
host	<a href="#">string</a>	Yes	
id	<a href="#">string</a>	No	
message	<a href="#">string</a>	No	"%{message}"
msgid	<a href="#">string</a>	No	"-"

Setting	Input type	Required	Default value
port	<a href="#">number</a>	Yes	
priority	<a href="#">string</a>	No	"%{syslog_pri}"
procid	<a href="#">string</a>	No	"-"
protocol	<a href="#">string</a> , one of ["tcp", "udp", "ssl-tcp"]	No	"udp"
reconnect_interval	<a href="#">number</a>	No	1
rfc	<a href="#">string</a> , one of ["rfc3164", "rfc5424"]	No	"rfc3164"
severity	<a href="#">string</a>	No	"notice"
sourcehost	<a href="#">string</a>	No	"%{host}"
ssl_cacert	a valid filesystem path	No	
ssl_cert	a valid filesystem path	No	
ssl_key	a valid filesystem path	No	
ssl_key_passphrase	<a href="#">password</a>	No	nil
ssl_verify	<a href="#">boolean</a>	No	false
use_labels	<a href="#">boolean</a>	No	true
workers	<<,>>	No	1

## Details

### [appname](#)

- Value type is [string](#)
- Default value is "LOGSTASH"

application name for syslog message. The new value can include `%{foo}` strings to help you build a new value from other parts of the event.

### [codec](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### [enable\\_metric](#)

- Value type is [boolean](#)

- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

*facility*

- Value type is [string](#)
- Default value is "user-level"

facility label for syslog message default fallback to user-level as in rfc3164 The new value can include `%{foo}` strings to help you build a new value from other parts of the event.

*host*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

syslog server address to connect to

*id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

``` output { stdout { id => "ABC" } } ```

If you don't explicitly set this variable Logstash will generate a unique name.

message

- Value type is [string](#)
- Default value is "`%{message}`"

message text to log. The new value can include `%{foo}` strings to help you build a new value from other parts of the event.

msgid

- Value type is [string](#)
- Default value is "-"

Logstash 5.2 Configuration Guide

message id for syslog message. The new value can include `%{foo}` strings to help you build a new value from other parts of the event.

`port`

- This is a required setting.
- Value type is [number](#)
- There is no default value for this setting.

syslog server port to connect to

`priority`

- Value type is [string](#)
- Default value is "`%{syslog_pri}`"

syslog priority The new value can include `%{foo}` strings to help you build a new value from other parts of the event.

`procid`

- Value type is [string](#)
- Default value is "`-`"

process id for syslog message. The new value can include `%{foo}` strings to help you build a new value from other parts of the event.

`protocol`

- Value can be any of: `tcp`, `udp`, `ssl-tcp`
- Default value is "`udp`"

syslog server protocol. you can choose between udp, tcp and ssl/tls over tcp

`reconnect_interval`

- Value type is [number](#)
- Default value is `1`

when connection fails, retry interval in sec.

`rfc`

- Value can be any of: `rfc3164`, `rfc5424`
- Default value is "`rfc3164`"

syslog message format: you can choose between rfc3164 or rfc5424

severity

- Value type is [string](#)
- Default value is "notice"

severity label for syslog message default fallback to notice as in rfc3164 The new value can include `%{foo}` strings to help you build a new value from other parts of the event.

sourcehost

- Value type is [string](#)
- Default value is "`%{host}`"

source host for syslog message. The new value can include `%{foo}` strings to help you build a new value from other parts of the event.

ssl_cacert

- Value type is [path](#)
- There is no default value for this setting.

The SSL CA certificate, chainfile or CA path. The system CA path is automatically included.

ssl_cert

- Value type is [path](#)
- There is no default value for this setting.

SSL certificate path

ssl_key

- Value type is [path](#)
- There is no default value for this setting.

SSL key path

ssl_key_passphrase

- Value type is [password](#)
- Default value is `nil`

SSL key passphrase

ssl_verify

- Value type is [boolean](#)

- Default value is `false`

Verify the identity of the other end of the SSL connection against the CA.

use_labels

- Value type is [boolean](#)
- Default value is `true`

use label parsing for severity and facility levels use priority field if set to false

workers

- Value type is [string](#)
- Default value is `1`

TODO remove this in Logstash 6.0 when we no longer support the `:legacy` type This is hacky, but it can only be herne

tcp

- Version: 4.0.0
- Released on: 2016-09-13
- [Changelog](#)

Write events over a TCP socket.

Each event json is separated by a newline.

Can either accept connections from clients or connect to a server, depending on `mode`.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
tcp {
    host => ...
    port => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	codec	No	"plain"
enable_metric	boolean	No	true
host	string	Yes	
id	string	No	
mode	string , one of ["server", "client"]	No	"client"
port	number	Yes	
reconnect_interval	number	No	10
ssl_cacert	a valid filesystem path	No	
ssl_cert	a valid filesystem path	No	
ssl_enable	boolean	No	false
ssl_key	a valid filesystem path	No	
ssl_key_passphrase	password	No	nil
ssl_verify	boolean	No	false

Setting	Input type	Required	Default value
workers	<<,>>	No	1

Details

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

enable_metric

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

host

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

When mode is `server`, the address to listen on. When mode is `client`, the address to connect to.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
  stdout {
    id => "my_plugin_id"
  }
}
```

message_format (DEPRECATED)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [string](#)
- There is no default value for this setting.

The format to use when writing events to the file. This value supports any string and can include `%{name}` and other dynamic strings.

If this setting is omitted, the full json representation of the event will be written as a single line.

mode

- Value can be any of: `server`, `client`
- Default value is "client"

Mode to operate in. `server` listens for client connections, `client` connects to a server.

port

- This is a required setting.
- Value type is [number](#)
- There is no default value for this setting.

When mode is `server`, the port to listen on. When mode is `client`, the port to connect to.

reconnect_interval

- Value type is [number](#)
- Default value is 10

When connect failed,retry interval in sec.

ssl_cacert

- Value type is [path](#)
- There is no default value for this setting.

The SSL CA certificate, chainfile or CA path. The system CA path is automatically included.

ssl_cert

- Value type is [path](#)
- There is no default value for this setting.

SSL certificate path

ssl_enable

- Value type is [boolean](#)
- Default value is `false`

Enable SSL (must be set for other `ssl_` options to take effect).

ssl_key

- Value type is [path](#)
- There is no default value for this setting.

SSL key path

ssl_key_passphrase

- Value type is [password](#)
- Default value is `nil`

SSL key passphrase

ssl_verify

- Value type is [boolean](#)
- Default value is `false`

Verify the identity of the other end of the SSL connection against the CA. For input, sets the field `sslsubject` to that of the client certificate.

workers

- Value type is [string](#)
- Default value is `1`

TODO remove this in Logstash 6.0 when we no longer support the `:legacy` type This is hacky, but it can only be here

udp

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

Send events over UDP

Keep in mind that UDP will lose messages.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
udp {
    host => ...
    port => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	codec	No	"plain"
enable_metric	boolean	No	true
host	string	Yes	
id	string	No	
port	number	Yes	
workers	<<, >>	No	1

Details

[codec](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`host`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The address to send messages to

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
    stdout {  
        id => "my_plugin_id"  
    }  
}
```

`port`

- This is a required setting.
- Value type is [number](#)
- There is no default value for this setting.

The port to send messages on

`workers`

- Value type is [string](#)
- Default value is `1`

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

webhdfs

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

This plugin sends Logstash events into files in HDFS via the [webhdfs](#) REST API.

Dependencies

This plugin has no dependency on jars from hadoop, thus reducing configuration and compatibility problems. It uses the webhdfs gem from Kazuki Ohta and TAGOMORI Satoshi (@see: <https://github.com/kzk/webhdfs>). Optional dependencies are zlib and snappy gem if you use the compression functionality.

Operational Notes

If you get an error like:

```
Max write retries reached. Exception: initialize: name or service not known
{:level=>:error}
```

make sure that the hostname of your namenode is resolvable on the host running Logstash. When creating/appending to a file, webhdfs sometime sends a 307 TEMPORARY_REDIRECT with the HOSTNAME of the machine its running on.

Usage

This is an example of Logstash config:

```
input {
    ...
}

filter {
    ...
}

output {
    webhdfs {
        host => "127.0.0.1"                      # (required)
        port => 50070                                # (optional, default: 50070)
        path => "/user/logstash/dt=%{+YYYY-MM-dd}/logstash-%{+HH}.log"  #
(required)
        user => "hue"                               # (required)
    }
}
```

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
webhdfs {
    host => ...
    path => ...
    user => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	codec	No	"plain"
compression	string , one of ["none", "snappy", "gzip"]	No	"none"
enable metric	boolean	No	true
flush size	number	No	500
host	string	Yes	
id	string	No	
idle flush time	number	No	1
open timeout	number	No	30
path	string	Yes	
port	number	No	50070
read timeout	number	No	30
retry interval	number	No	0.5
retry known errors	boolean	No	true
retry times	number	No	5
single file per thread	boolean	No	false
snappy bufsize	number	No	32768
snappy format	string , one of ["stream", "file"]	No	"stream"
standby host	string	No	false
standby port	number	No	50070
use httpfs	boolean	No	false
user	string	Yes	
workers	<<,>>	No	1

Details

`codec`

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

`compression`

- Value can be any of: `none`, `snappy`, `gzip`
- Default value is "`none`"

Compress output. One of [`none`, `snappy`, `gzip`]

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`flush_size`

- Value type is [number](#)
- Default value is 500

Sending data to webhdfs if event count is above, even if `store_interval_in_secs` is not reached.

`host`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The server name for webhdfs/httpfs connections.

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
  stdout {  
    id => "my_plugin_id"  
  }  
}  
idle_flush_time
```

- Value type is [number](#)
- Default value is 1

Sending data to webhdfs in x seconds intervals.

open_timeout

- Value type is [number](#)
- Default value is 30

WebHdfs open timeout, default 30s.

path

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The path to the file to write to. Event fields can be used here, as well as date fields in the joda time format, e.g.: /user/logstash/dt=%{+YYYY-MM-dd}/{@source_host}-%{+HH}.log

port

- Value type is [number](#)
- Default value is 50070

The server port for webhdfs/httpfs connections.

read_timeout

- Value type is [number](#)
- Default value is 30

The WebHdfs read timeout, default 30s.

retry_interval

- Value type is [number](#)
- Default value is 0.5

How long should we wait between retries.

retry_known_errors

- Value type is [boolean](#)
- Default value is true

Retry some known webhdfs errors. These may be caused by race conditions when appending to same file, etc.

retry_times

- Value type is [number](#)
- Default value is 5

How many times should we retry. If retry_times is exceeded, an error will be logged and the event will be discarded.

single_file_per_thread

- Value type is [boolean](#)
- Default value is false

Avoid appending to same file in multiple threads. This solves some problems with multiple logstash output threads and locked file leases in webhdfs. If this option is set to true, `%{@metadata}[thread_id]` needs to be used in path config setting.

snappy_bufsize

- Value type is [number](#)
- Default value is 32768

Set snappy chunksize. Only necessary for stream format. Defaults to 32k. Max is 65536 @see http://code.google.com/p/snappy/source/browse/trunk/framing_format.txt

snappy_format

- Value can be any of: stream, file
- Default value is "stream"

Set snappy format. One of "stream", "file". Set to stream to be hive compatible.

standby_host

- Value type is [string](#)
- Default value is `false`

Standby namenode for ha hdfs.

standby_port

- Value type is [number](#)
- Default value is `50070`

Standby namenode port for ha hdfs.

use_https

- Value type is [boolean](#)
- Default value is `false`

Use https mode if set to true, else webhdfs.

user

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The Username for webhdfs.

workers

- Value type is [string](#)
- Default value is `1`

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

websocket

- Version: 3.0.1
- Released on: 2016-07-14
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
`bin/logstash-plugin install logstash-output-websocket`.

This output runs a websocket server and publishes any messages to all connected websocket clients.

You can connect to it with `ws://<host>:<port>/`

If no clients are connected, any messages received are ignored.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
websocket {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	codec	No	"plain"
enable_metric	boolean	No	true
host	string	No	"0.0.0.0"
id	string	No	
port	number	No	3232
workers	<code><<, >></code>	No	1

Details

codec

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

enable_metric

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

host

- Value type is [string](#)
- Default value is "0.0.0.0"

The address to serve websocket data from

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

### *port*

- Value type is [number](#)
- Default value is 3232

The port to serve websocket data from

### *workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be herne

## xmpp

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

This output allows you ship events over XMPP/Jabber.

This plugin can be used for posting events to humans over XMPP, or you can use it for PubSub or general message passing for logstash to logstash.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
xmpp {
 message => ...
 password => ...
 user => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">codec</a>	<a href="#">codec</a>	No	"plain"
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">host</a>	<a href="#">string</a>	No	
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">message</a>	<a href="#">string</a>	Yes	
<a href="#">password</a>	<a href="#">password</a>	Yes	
<a href="#">rooms</a>	<a href="#">array</a>	No	
<a href="#">user</a>	<a href="#">string</a>	Yes	
<a href="#">users</a>	<a href="#">array</a>	No	
<a href="#">workers</a>	<code>&lt;&lt;, &gt;&gt;</code>	No	1

## Details

### *codec*

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### *enable\_metric*

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### *host*

- Value type is [string](#)
- There is no default value for this setting.

The xmpp server to connect to. This is optional. If you omit this setting, the host on the user/identity is used. (foo.com for [user@foo.com](#))

### *id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

### *message*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The message to send. This supports dynamic strings like `%{host}`

*password*

- This is a required setting.
- Value type is [password](#)
- There is no default value for this setting.

The xmpp password for the user/identity.

*rooms*

- Value type is [array](#)
- There is no default value for this setting.

if muc/multi-user-chat required, give the name of the room that you want to join:  
[room@conference.domain/nick](#)

*user*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The user or resource ID, like [foo@example.com](#).

*users*

- Value type is [array](#)
- There is no default value for this setting.

The users to send messages to

*workers*

- Value type is [string](#)
- Default value is 1

TODO remove this in Logstash 6.0 when we no longer support the :legacy type This is hacky, but it can only be here

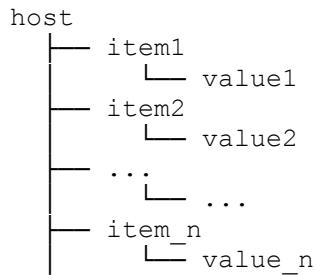
## zabbix

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-output-zabbix`.

The Zabbix output is used to send item data (key/value pairs) to a Zabbix server. The event `@timestamp` will automatically be associated with the Zabbix item data.

The Zabbix Sender protocol is described at

[https://www.zabbix.org/wiki/Docs/protocols/zabbix\\_sender/2.0](https://www.zabbix.org/wiki/Docs/protocols/zabbix_sender/2.0) Zabbix uses a kind of nested key/value store.



Each "host" is an identifier, and each item is associated with that host. Items are typed on the Zabbix side. You can send numbers as strings and Zabbix will Do The Right Thing.

In the Zabbix UI, ensure that your hostname matches the value referenced by `zabbix_host`. Create the item with the key as it appears in the field referenced by `zabbix_key`. In the item configuration window, ensure that the type dropdown is set to Zabbix Trapper. Also be sure to set the type of information that Zabbix should expect for this item.

This plugin does not currently send in batches. While it is possible to do so, this is not supported. Be careful not to flood your Zabbix server with too many events per second.

This plugin will log a warning if a necessary field is missing. It will not attempt to resend if Zabbix is down, but will log an error message.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
zabbix {
 zabbix_host => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
codec	<a href="#">codec</a>	No	"plain"
multi_value	<a href="#">array</a>	No	
timeout	<a href="#">number</a>	No	1
workers	<a href="#">number</a>	No	1
zabbix_host	<a href="#">string</a>	Yes	
zabbix_key	<a href="#">string</a>	No	
zabbix_server_host	<a href="#">string</a>	No	"localhost"
zabbix_server_port	<a href="#">number</a>	No	10051
zabbix_value	<a href="#">string</a>	No	"message"

## Details

### [codec](#)

- Value type is [codec](#)
- Default value is "plain"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### [multi\\_value](#)

- Value type is [array](#)
- There is no default value for this setting.

Use the `multi_value` directive to send multiple key/value pairs. This can be thought of as an array, like:

```
[zabbix_key1, zabbix_value1, zabbix_key2, zabbix_value2, ... zabbix_keyN,
zabbix_valueN]
```

...where `zabbix_key1` is an instance of `zabbix_key`, and `zabbix_value1` is an instance of `zabbix_value`. If the field referenced by any `zabbix_key` or `zabbix_value` does not exist, that entry will be ignored.

This directive cannot be used in conjunction with the single-value directives `zabbix_key` and `zabbix_value`.

`timeout`

- Value type is [number](#)
- Default value is 1

The number of seconds to wait before giving up on a connection to the Zabbix server. This number should be very small, otherwise delays in delivery of other outputs could result.

`workers`

- Value type is [number](#)
- Default value is 1

The number of workers to use for this output. Note that this setting may not be useful for all outputs.

`zabbix_host`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The field name which holds the Zabbix host name. This can be a sub-field of the @metadata field.

`zabbix_key`

- Value type is [string](#)
- There is no default value for this setting.

A single field name which holds the value you intend to use as the Zabbix item key. This can be a sub-field of the @metadata field. This directive will be ignored if using `multi_value`

`zabbix_server_host`

- Value type is [string](#)
- Default value is "localhost"

The IP or resolvable hostname where the Zabbix server is running

`zabbix_server_port`

- Value type is [number](#)
- Default value is 10051

The port on which the Zabbix server is running

`zabbix_value`

- Value type is [string](#)
- Default value is "message"

The field name which holds the value you want to send. This directive will be ignored if using `multi_value`

## zeromq

NOTE: This is a community-maintained plugin!

Write events to a 0MQ PUB socket.

You need to have the 0mq 2.1.x library installed to be able to use this output plugin.

The default settings will create a publisher connecting to a subscriber bound to `tcp://127.0.0.1:2120`

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
zeromq {
 topology => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">address</a>	<a href="#">array</a>	No	<code>["tcp://127.0.0.1:2120"]</code>
<a href="#">codec</a>	<a href="#">codec</a>	No	<code>"json"</code>
<a href="#">mode</a>	<a href="#">string</a> , one of <code>["server", "client"]</code>	No	<code>"client"</code>
<a href="#">sockopt</a>	<a href="#">hash</a>	No	
<a href="#">topic</a>	<a href="#">string</a>	No	<code>""</code>
<a href="#">topology</a>	<a href="#">string</a> , one of <code>["pushpull", "pubsub", "pair"]</code>	Yes	
<a href="#">workers</a>	<a href="#">number</a>	No	<code>1</code>

## Details

### `address`

- Value type is [array](#)
- Default value is `["tcp://127.0.0.1:2120"]`

0mq socket address to connect or bind. Please note that `inproc://` will not work with logstash. For each we use a context per thread. By default, inputs bind/listen and outputs connect.

### `codec`

- Value type is [codec](#)
- Default value is "json"

The codec used for output data. Output codecs are a convenient method for encoding your data before it leaves the output, without needing a separate filter in your Logstash pipeline.

### `mode`

- Value can be any of: `server`, `client`
- Default value is "client"

Server mode binds/listens. Client mode connects.

### `sockopt`

- Value type is [hash](#)
- There is no default value for this setting.

This exposes `zmq_setsockopt` for advanced tuning. See <http://api.zeromq.org/2-1:zmq-setsockopt> for details.

This is where you would set values like:

- `ZMQ::HWM` - high water mark
- `ZMQ::IDENTITY` - named queues
- `ZMQ::SWAP_SIZE` - space for disk overflow

Example:

```
sockopt => {
 "ZMQ::HWM" => 50
 "ZMQ::IDENTITY" => "my_named_queue"
}
```

### `topic`

- Value type is [string](#)
- Default value is ""

This is used for the *pubsub* topology only. On inputs, this allows you to filter messages by topic. On outputs, this allows you to tag a message for routing. NOTE: ZeroMQ does subscriber-side filtering NOTE: Topic is evaluated with `event.strftime` so macros are valid here.

*topology*

- This is a required setting.
- Value can be any of: pushpull, pubsub, pair
- There is no default value for this setting.

The default logstash topologies work as follows:

- pushpull - inputs are pull, outputs are push
- pubsub - inputs are subscribers, outputs are publishers
- pair - inputs are clients, inputs are servers

If the predefined topology flows don't work for you, you can change the *mode* setting

*workers*

- Value type is [number](#)
- Default value is 1

The number of workers to use for this output. Note that this setting may not be useful for all outputs.

# Filter plugins

A filter plugin performs intermediary processing on an event. Filters are often applied conditionally depending on the characteristics of the event.

The following filter plugins are available below. For a list of Elastic supported plugins, please consult the [Support Matrix](#).

Plugin	Description	Github repository
<a href="#">aggregate</a>	Aggregates information from several events originating with a single task	<a href="#">logstash-filter-aggregate</a>
<a href="#">alter</a>	Performs general alterations to fields that the <code>mutate</code> filter does not handle	<a href="#">logstash-filter-alter</a>
<a href="#">anonymize</a>	Replaces field values with a consistent hash	<a href="#">logstash-filter-anonymize</a>
<a href="#">cidr</a>	Checks IP addresses against a list of network blocks	<a href="#">logstash-filter-cidr</a>
<a href="#">cipher</a>	Applies or removes a cipher to an event	<a href="#">logstash-filter-cipher</a>
<a href="#">clone</a>	Duplicates events	<a href="#">logstash-filter-clone</a>
<a href="#">collate</a>	Collates events by time or count	<a href="#">logstash-filter-collate</a>
<a href="#">csv</a>	Parses comma-separated value data into individual fields	<a href="#">logstash-filter-csv</a>
<a href="#">date</a>	Parses dates from fields to use as the Logstash timestamp for an event	<a href="#">logstash-filter-date</a>
<a href="#">de_dot</a>	Computationally expensive filter that removes dots from a field name	<a href="#">logstash-filter-de_dot</a>
<a href="#">dissect</a>	Extracts unstructured event data into fields using delimiters	<a href="#">logstash-filter-dissect</a>
<a href="#">dns</a>	Performs a standard or reverse DNS lookup	<a href="#">logstash-filter-dns</a>
<a href="#">drop</a>	Drops all events	<a href="#">logstash-filter-drop</a>
<a href="#">elapsed</a>	Calculates the elapsed time between a pair of events	<a href="#">logstash-filter-elapsed</a>
<a href="#">elasticsearch</a>	Copies fields from previous log events in Elasticsearch to current events	<a href="#">logstash-filter-elasticsearch</a>
<a href="#">environment</a>	Stores environment variables as metadata sub-fields	<a href="#">logstash-filter-environment</a>
<a href="#">extractnumbers</a>	Extracts numbers from a string	<a href="#">logstash-filter-extractnumbers</a>
<a href="#">fingerprint</a>	Fingerprints fields by replacing values with a consistent hash	<a href="#">logstash-filter-fingerprint</a>

<a href="#">geoip</a>	Adds geographical information about an IP address	<a href="#">logstash-filter-geoip</a>
<a href="#">grok</a>	Parses unstructured event data into fields	<a href="#">logstash-filter-grok</a>
<a href="#">i18n</a>	Removes special characters from a field	<a href="#">logstash-filter-i18n</a>
<a href="#">json</a>	Parses JSON events	<a href="#">logstash-filter-json</a>
<a href="#">json_encode</a>	Serializes a field to JSON	<a href="#">logstash-filter-json_encode</a>
<a href="#">kv</a>	Parses key-value pairs	<a href="#">logstash-filter-kv</a>
<a href="#">metaevent</a>	Adds arbitrary fields to an event	<a href="#">logstash-filter-metaevent</a>
<a href="#">metricize</a>	Takes complex events containing a number of metrics and splits these up into multiple events, each holding a single metric	<a href="#">logstash-filter-metricize</a>
<a href="#">metrics</a>	Aggregates metrics	<a href="#">logstash-filter-metrics</a>
<a href="#">mutate</a>	Performs mutations on fields	<a href="#">logstash-filter-mutate</a>
<a href="#">oui</a>	Parse OUI data from MAC addresses	<a href="#">logstash-filter-oui</a>
<a href="#">prune</a>	Prunes event data based on a list of fields to blacklist or whitelist	<a href="#">logstash-filter-prune</a>
<a href="#">punct</a>	Strips all non-punctuation content from a field	<a href="#">logstash-filter-punct</a>
<a href="#">range</a>	Checks that specified fields stay within given size or length limits	<a href="#">logstash-filter-range</a>
<a href="#">ruby</a>	Executes arbitrary Ruby code	<a href="#">logstash-filter-ruby</a>
<a href="#">sleep</a>	Sleeps for a specified time span	<a href="#">logstash-filter-sleep</a>
<a href="#">split</a>	Splits multi-line messages into distinct events	<a href="#">logstash-filter-split</a>
<a href="#">syslog_pri</a>	Parses the PRI (priority) field of a syslog message	<a href="#">logstash-filter-syslog_pri</a>
<a href="#">throttle</a>	Throttles the number of events	<a href="#">logstash-filter-throttle</a>
<a href="#">tld</a>	Replaces the contents of the default message field with whatever you specify in the configuration	<a href="#">logstash-filter-tld</a>
<a href="#">translate</a>	Replaces field contents based on a hash or YAML file	<a href="#">logstash-filter-translate</a>
<a href="#">truncate</a>	Truncates fields longer than a given length.	<a href="#">logstash-filter-truncate</a>
<a href="#">urldecode</a>	Decodes URL-encoded fields	<a href="#">logstash-filter-urldecode</a>
<a href="#">useragent</a>	Parses user agent strings into fields	<a href="#">logstash-filter-useragent</a>

<a href="#">uuid</a>	Adds a UUID to events	<a href="#">logstash-filter-uuid</a>
<a href="#">xml</a>	Parses XML into fields	<a href="#">logstash-filter-xml</a>
<a href="#">yaml</a>	Takes an existing field that contains YAML and expands it into an actual data structure within the Logstash event	<a href="#">logstash-filter-yaml</a>
<a href="#">zeromq</a>	Sends an event to ZeroMQ	<a href="#">logstash-filter-zeromq</a>

## aggregate

- Version: 2.5.1
- Released on: 2017-01-08
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
bin/logstash-plugin install logstash-filter-aggregate.

The aim of this filter is to aggregate information available among several events (typically log lines) belonging to a same task, and finally push aggregated information into final task event.

You should be very careful to set logstash filter workers to 1 (-w 1 flag) for this filter to work correctly otherwise events may be processed out of sequence and unexpected results will occur.

### Example #1

- with these given logs :

```
INFO - 12345 - TASK_START - start
INFO - 12345 - SQL - sqlQuery1 - 12
INFO - 12345 - SQL - sqlQuery2 - 34
INFO - 12345 - TASK_END - end
```

- you can aggregate "sql duration" for the whole task with this configuration :

```
filter {
 grok {
 match => ["message", "%{LOGLEVEL:loglevel} - %{NOTSPACE:taskid} - %{NOTSPACE:logger} - %{WORD:label}(- %{INT:duration:int})?"]
 }

 if [logger] == "TASK_START" {
 aggregate {
 task_id => "%{taskid}"
 code => "map['sql_duration'] = 0"
 map_action => "create"
 }
 }

 if [logger] == "SQL" {
 aggregate {
 task_id => "%{taskid}"
 code => "map['sql_duration'] += event.get('duration')"
 map_action => "update"
 }
 }

 if [logger] == "TASK_END" {
 aggregate {
 task_id => "%{taskid}"
 }
 }
}
```

```

 code => "event.set('sql_duration', map['sql_duration'])"
 map_action => "update"
 end_of_task => true
 timeout => 120
 }
}
}

```

- the final event then looks like :

```
{
 "message" => "INFO - 12345 - TASK_END - end message",
 "sql_duration" => 46
}
```

the field `sql_duration` is added and contains the sum of all sql queries durations.

## Example #2 : no start event

- If you have the same logs than example #1, but without a start log :

```
INFO - 12345 - SQL - sqlQuery1 - 12
INFO - 12345 - SQL - sqlQuery2 - 34
INFO - 12345 - TASK_END - end
```

- you can also aggregate "sql duration" with a slightly different configuration :

```

filter {
 grok {
 match => ["message", "%{LOGLEVEL:loglevel} - %{NOTSPACE:taskid} - %{NOTSPACE:logger} - %{WORD:label}(- %{INT:duration:int})?"]
 }

 if [logger] == "SQL" {
 aggregate {
 task_id => "%{taskid}"
 code => "map['sql_duration'] ||= 0 ; map['sql_duration'] += event.get('duration')"
 }
 }

 if [logger] == "TASK_END" {
 aggregate {
 task_id => "%{taskid}"
 code => "event.set('sql_duration', map['sql_duration'])"
 end_of_task => true
 timeout => 120
 }
 }
}

```

- the final event is exactly the same than example #1

- the key point is the "`| |=`" ruby operator. It allows to initialize `sql_duration` map entry to 0 only if this map entry is not already initialized

### Example #3 : no end event

Third use case: You have no specific end event.

A typical case is aggregating or tracking user behaviour. We can track a user by its ID through the events, however once the user stops interacting, the events stop coming in. There is no specific event indicating the end of the user's interaction.

In this case, we can enable the option `push_map_as_event_on_timeout` to enable pushing the aggregation map as a new event when a timeout occurs. In addition, we can enable `timeout_code` to execute code on the populated timeout event. We can also add `timeout_task_id_field` so we can correlate the task\_id, which in this case would be the user's ID.

- Given these logs:

```
INFO - 12345 - Clicked One
INFO - 12345 - Clicked Two
INFO - 12345 - Clicked Three
```

- You can aggregate the amount of clicks the user did like this:

```
filter {
 grok {
 match => ["message", "%{LOGLEVEL:loglevel} - %{NOTSPACE:user_id} - %{GREEDYDATA:msg_text}"]
 }

 aggregate {
 task_id => "%{user_id}"
 code => "map['clicks'] ||= 0; map['clicks'] += 1;"
 push_map_as_event_on_timeout => true
 timeout_task_id_field => "user_id"
 timeout => 600 # 10 minutes timeout
 timeout_tags => ['_aggregatetimerout']
 timeout_code => "event.set('several_clicks', event.get('clicks') > 1)"
 }
}
```

- After ten minutes, this will yield an event like:

```
{
 "user_id": "12345",
 "clicks": 3,
 "several_clicks": true,
 "tags": [
 "_aggregatetimerout"
]
}
```

## Example #4 : no end event and tasks come one after the other

Fourth use case : like example #3, you have no specific end event, but also, tasks come one after the other. That is to say : tasks are not interleaved. All task1 events come, then all task2 events come, ... In that case, you don't want to wait task timeout to flush aggregation map. \* A typical case is aggregating results from jdbc input plugin. \* Given that you have this SQL query :

SELECT country\_name, town\_name FROM town \* Using jdbc input plugin, you get these 3 events from :

```
{ "country_name": "France", "town_name": "Paris" }
{ "country_name": "France", "town_name": "Marseille" }
{ "country_name": "USA", "town_name": "New-York" }
```

- And you would like these 2 result events to push them into elasticsearch :

```
{ "country_name": "France", "town_name": ["Paris", "Marseille"] }
{ "country_name": "USA", "town_name": ["New-York"] }
```

- You can do that using `push_previous_map_as_event` aggregate plugin option :

```
filter {
 aggregate {
 task_id => "%{country_name}"
 code => "
 map['town_name'] ||= []
 event.to_hash.each do |key,value|
 map[key] = value unless map.has_key?(key)
 map[key] << value if map[key].is_a?(Array) and
!value.is_a?(Array)
 end
 "
 push_previous_map_as_event => true
 timeout => 5
 timeout_tags => ['aggregated']
 }
 if "aggregated" not in [tags] {
 drop {}
 }
}
```

- The key point is that each time aggregate plugin detects a new `country_name`, it pushes previous aggregate map as a new logstash event (with `aggregated` tag), and then creates a new empty map for the next country
- When 5s timeout comes, the last aggregate map is pushed as a new event
- Finally, initial events (which are not aggregated) are dropped because useless

## How it works

- the filter needs a "task\_id" to correlate events (log lines) of a same task
- at the task beginning, filter creates a map, attached to task\_id

- for each event, you can execute code using `event` and `map` (for instance, copy an event field to map)
- in the final event, you can execute a last code (for instance, add map data to final event)
- after the final event, the map attached to task is deleted
- in one filter configuration, it is recommended to define a timeout option to protect the feature against unterminated tasks. It tells the filter to delete expired maps
- if no timeout is defined, by default, all maps older than 1800 seconds are automatically deleted
- all timeout options have to be defined in only one aggregate filter per task\_id pattern. Timeout options are : `timeout`, `timeout_code`, `push_map_as_event_on_timeout`, `push_previous_map_as_event`, `timeout_task_id_field`, `timeout_tags`
- if `code` execution raises an exception, the error is logged and event is tagged `_aggregateexception`

## Use Cases

- extract some cool metrics from task logs and push them into task final log event (like in example #1 and #2)
- extract error information in any task log line, and push it in final task event (to get a final event with all error information if any)
- extract all back-end calls as a list, and push this list in final task event (to get a task profile)
- extract all http headers logged in several lines to push this list in final task event (complete http request info)
- for every back-end call, collect call details available on several lines, analyse it and finally tag final back-end call log line (error, timeout, business-warning, ...)
- Finally, task id can be any correlation id matching your need : it can be a session id, a file path, ...

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
aggregate {
 code => ...
 task_id => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>add_field</code>	<a href="#">hash</a>	No	{ }
<code>add_tag</code>	<a href="#">array</a>	No	[ ]
<code>aggregate_maps_path</code>	<a href="#">string</a>	No	

Setting	Input type	Required	Default value
code	<a href="#">string</a>	Yes	
enable_metric	<a href="#">boolean</a>	No	true
end_of_task	<a href="#">boolean</a>	No	false
id	<a href="#">string</a>	No	
map_action	<a href="#">string</a>	No	"create_or_update"
periodic_flush	<a href="#">boolean</a>	No	false
push_map_as_event_on_timeout	<a href="#">boolean</a>	No	false
push_previous_map_as_event	<a href="#">boolean</a>	No	false
remove_field	<a href="#">array</a>	No	[]
remove_tag	<a href="#">array</a>	No	[]
task_id	<a href="#">string</a>	Yes	
timeout	<a href="#">number</a>	No	
timeout_code	<a href="#">string</a>	No	
timeout_tags	<a href="#">array</a>	No	[]
timeout_task_id_field	<a href="#">string</a>	No	

## Details

### [add\\_field](#)

- Value type is [hash](#)
- Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 aggregate {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
 # You can also add multiple fields at once:
 filter {
 aggregate {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
 }
}
```

```

 }
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

#### `add_tag`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```

filter {
 aggregate {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 aggregate {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

#### `aggregate_maps_path`

- Value type is [string](#)
- There is no default value for this setting.

The path to file where aggregate maps are stored when logstash stops and are loaded from when logstash starts.

If not defined, aggregate maps will not be stored at logstash stop and will be lost. Must be defined in only one aggregate filter (as aggregate maps are global).

Example value : `"/path/to/.aggregate_maps"`

#### `code`

- This is a required setting.

- Value type is [string](#)
- There is no default value for this setting.

The code to execute to update map, using current event.

Or on the contrary, the code to execute to update event, using current map.

You will have a *map* variable and an *event* variable available (that is the event itself).

Example value : "map['sql\_duration'] += event.get('duration')"

[enable\\_metric](#)

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

[end\\_of\\_task](#)

- Value type is [boolean](#)
- Default value is `false`

Tell the filter that task is ended, and therefore, to delete aggregate map after code execution.

[id](#)

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

``` output { stdout { id => "ABC" } } ```

If you don't explicitly set this variable Logstash will generate a unique name.

[map_action](#)

- Value type is [string](#)
- Default value is `"create_or_update"`

Tell the filter what to do with aggregate map.

`create`: create the map, and execute the code only if map wasn't created before

`update`: doesn't create the map, and execute the code only if map was created before

`create_or_update`: create the map if it wasn't created before, execute the code in all cases

`periodic_flush`

- Value type is [boolean](#)
- Default value is `false`

Call the filter flush method at regular interval. Optional.

`push_map_as_event_on_timeout`

- Value type is [boolean](#)
- Default value is `false`

When this option is enabled, each time a task timeout is detected, it pushes task aggregation map as a new logstash event. This enables to detect and process task timeouts in logstash, but also to manage tasks that have no explicit end event.

`push_previous_map_as_event`

- Value type is [boolean](#)
- Default value is `false`

When this option is enabled, each time aggregate plugin detects a new task id, it pushes previous aggregate map as a new logstash event, and then creates a new empty map for the next task.

this option works fine only if tasks come one after the other. It means : all task1 events, then all task2 events, etc...

`remove_field`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
  aggregate {
    remove_field => [ "foo_{somefield}" ]
  }
}
# You can also remove multiple fields at once:
filter {
```

```
aggregate {  
    remove_field => [ "foo_{somefield}", "my_extraneous_field" ]  
}  
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

- Value type is [array](#)
- Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {  
    aggregate {  
        remove_tag => [ "foo_{somefield}" ]  
    }  
}  
# You can also remove multiple tags at once:  
filter {  
    aggregate {  
        remove_tag => [ "foo_{somefield}", "sad_unwanted_tag" ]  
    }  
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

`task_id`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The expression defining task ID to correlate logs.

This value must uniquely identify the task in the system.

`timeout`

- Value type is [number](#)
- There is no default value for this setting.

The amount of seconds after a task "end event" can be considered lost.

When timeout occurs for a task, The task "map" is evicted.

Timeout can be defined for each "task_id" pattern.

If no timeout is defined, default timeout will be applied : 1800 seconds.

`timeout_code`

- Value type is [string](#)
- There is no default value for this setting.

The code to execute to complete timeout generated event, when `push_map_as_event_on_timeout` or `push_previous_map_as_event` is set to true. The code block will have access to the newly generated timeout event that is pre-populated with the aggregation map.

If `timeout_task_id_field` is set, the event is also populated with the task_id value

Example value: "event.set('state', 'timeout')"

`timeout_tags`

- Value type is [array](#)
- Default value is []

Defines tags to add when a timeout event is generated and yield

`timeout_task_id_field`

- Value type is [string](#)
- There is no default value for this setting.

This option indicates the timeout generated event's field for the "task_id" value. The task id will then be set into the timeout event. This can help correlate which tasks have been timed out.

This field has no default value and will not be set on the event if not configured.

Example:

If the task_id is "12345" and this field is set to "my_id", the generated timeout event will contain 'my_id' key with '12345' value.

alter

- Version: 3.0.0
- Released on: 2016-11-16
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
`bin/logstash-plugin install logstash-filter-alter.`

The alter filter allows you to do general alterations to fields that are not included in the normal mutate filter.

The functionality provided by this plugin is likely to be merged into the *mutate* filter in future versions.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
alter { }
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
add_tag	array	No	[]
coalesce	array	No	
condrewrite	array	No	
condrewriteother	array	No	
enable_metric	boolean	No	true
id	string	No	
periodic_flush	boolean	No	false
remove_field	array	No	[]
remove_tag	array	No	[]

Details

`add_field`

- Value type is [hash](#)
- Default value is `{ }`

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
  alter {
    add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
  }
}
# You can also add multiple fields at once:
filter {
  alter {
    add_field => {
      "foo_%{somefield}" => "Hello world, from %{host}"
      "new_field" => "new_static_value"
    }
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

`add_tag`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  alter {
    add_tag => [ "foo_%{somefield}" ]
  }
}
# You can also add multiple tags at once:
filter {
```

```
alter {
    add_tag => [ "foo_{somefield}", "taggedy_tag"]
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`coalesce`

- Value type is [array](#)
- There is no default value for this setting.

Sets the value of `field_name` to the first nonnull expression among its arguments.

Example:

```
filter {
    alter {
        coalesce => [
            "field_name", "value1", "value2", "value3", ...
        ]
    }
}
```

`condrewrite`

- Value type is [array](#)
- There is no default value for this setting.

Change the content of the field to the specified value if the actual content is equal to the expected one.

Example:

```
filter {
    alter {
        condrewrite => [
            "field_name", "expected_value", "new_value",
            "field_name2", "expected_value2", "new_value2",
            ...
        ]
    }
}
```

`condrewriteother`

- Value type is [array](#)
- There is no default value for this setting.

Change the content of the field to the specified value if the content of another field is equal to the expected one.

Example:

```
filter {
    alter {
        condrewriteother => [
            "field_name", "expected_value", "field_name_to_change", "value",
            "field_name2", "expected_value2", "field_name_to_change2", "value2",
            ...
        ]
    }
}
enable_metric
```

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

*periodic\_flush*

- Value type is [boolean](#)
- Default value is `false`

Call the filter flush method at regular interval. Optional.

*remove\_field*

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 alter {
 remove_field => ["foo_%{somefield}"]
 }
}
```

```
}
```

# You can also remove multiple fields at once:

```
filter {
 alter {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

### `remove_tag`

- Value type is [array](#)
- Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 alter {
 remove_tag => ["foo_{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 alter {
 remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

## anonymize

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
`bin/logstash-plugin install logstash-filter-anonymize.`

Anonymize fields using by replacing values with a consistent hash.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
anonymize {
 algorithm => ...
 fields => ...
 key => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
add_tag	<a href="#">array</a>	No	[]
algorithm	<a href="#">string</a> , one of ["SHA1", "SHA256", "SHA384", "SHA512", "MD5", "MURMUR3", "IPV4_NETWORK"]	Yes	"SHA1"
enable_metric	<a href="#">boolean</a>	No	true
fields	<a href="#">array</a>	Yes	
id	<a href="#">string</a>	No	
key	<a href="#">string</a>	Yes	
periodic_flush	<a href="#">boolean</a>	No	false
remove_field	<a href="#">array</a>	No	[]
remove_tag	<a href="#">array</a>	No	[]

## Details

[add\\_field](#)

- Value type is [hash](#)
- Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 anonymize {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 anonymize {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

[add\\_tag](#)

- Value type is [array](#)
- Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 anonymize {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 anonymize {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

}

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

### *algorithm*

- This is a required setting.
- Value can be any of: SHA1, SHA256, SHA384, SHA512, MD5, MURMUR3, IPV4\_NETWORK
- Default value is "SHA1"

digest/hash type

### *enable\_metric*

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### *fields*

- This is a required setting.
- Value type is [array](#)
- There is no default value for this setting.

The fields to be anonymized

### *id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique ID to the plugin instance, this ID is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

key

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Hashing key When using MURMUR3 the key is ignored but must still be set. When using IPV4_NETWORK key is the subnet prefix length

periodic_flush

- Value type is [boolean](#)
- Default value is `false`

Call the filter flush method at regular interval. Optional.

remove_field

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
    anonymize {
        remove_field => [ "foo_{somefield}" ]
    }
}
# You can also remove multiple fields at once:
filter {
    anonymize {
        remove_field => [ "foo_{somefield}", "my_extraneous_field" ]
    }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

remove_tag

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
    anonymize {
        remove_tag => [ "foo_{somefield}" ]
    }
}
# You can also remove multiple tags at once:
filter {
```

Logstash 5.2 Configuration Guide

```
anonymize {  
    remove_tag => [ "foo_{somefield}", "sad_unwanted_tag"]  
}  
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

cidr

- Version: 3.0.0
- Released on: 2016-08-24
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
`bin/logstash-plugin install logstash-filter-cidr`.

The CIDR filter is for checking IP addresses in events against a list of network blocks that might contain it. Multiple addresses can be checked against multiple networks, any match succeeds. Upon success additional tags and/or fields can be added to the event.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
cidr {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
add_tag	array	No	[]
address	array	No	[]
enable_metric	boolean	No	true
id	string	No	
network	array	No	[]
periodic_flush	boolean	No	false
remove_field	array	No	[]
remove_tag	array	No	[]

Details

`add_field`

- Value type is [hash](#)
- Default value is `{ }`

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
  cidr {
    add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
  }
}
# You can also add multiple fields at once:
filter {
  cidr {
    add_field => {
      "foo_%{somefield}" => "Hello world, from %{host}"
      "new_field" => "new_static_value"
    }
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

`add_tag`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  cidr {
    add_tag => [ "foo_%{somefield}" ]
  }
}
# You can also add multiple tags at once:
filter {
  cidr {
    add_tag => [ "foo_%{somefield}", "taggedy_tag" ]
  }
}
```

Logstash 5.2 Configuration Guide

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`address`

- Value type is [array](#)
- Default value is `[]`

The IP address(es) to check with. Example:

```
filter {
  cidr {
    add_tag => [ "testnet" ]
    address => [ "%{src_ip}", "%{dst_ip}" ]
    network => [ "192.0.2.0/24" ]
  }
}
enable_metric
```

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

### `network`

- Value type is [array](#)
- Default value is `[]`

The IP network(s) to check against. Example:

```
filter {
 cidr {
 add_tag => ["linklocal"]
 address => ["%{clientip}"]
 network => ["169.254.0.0/16", "fe80::/64"]
 }
}
```

```
 }
}

periodic_flush
```

- Value type is [boolean](#)
- Default value is `false`

Call the filter flush method at regular interval. Optional.

[remove\\_field](#)

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 cidr {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 cidr {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

[remove\\_tag](#)

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 cidr {
 remove_tag => ["foo_{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 cidr {
```

```
remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
}
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

## cipher

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-filter-cipher`.

This filter parses a source and apply a cipher or decipher before storing it in the target.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
cipher {
 algorithm => ...
 mode => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
add_tag	<a href="#">array</a>	No	[]
algorithm	<a href="#">string</a>	Yes	
base64	<a href="#">boolean</a>	No	true
cipher_padding	<a href="#">string</a>	No	
iv_random_length	<a href="#">number</a>	No	
key	<a href="#">string</a>	No	
key_pad	<code>&lt;&lt;, &gt;&gt;</code>	No	"\u0000"
key_size	<a href="#">number</a>	No	16
max_cipher_reuse	<a href="#">number</a>	No	1
mode	<a href="#">string</a>	Yes	
periodic_flush	<a href="#">boolean</a>	No	false
remove_field	<a href="#">array</a>	No	[]
remove_tag	<a href="#">array</a>	No	[]
source	<a href="#">string</a>	No	"message"
target	<a href="#">string</a>	No	"message"

## Details

### `add_field`

- Value type is [hash](#)
- Default value is `{ }`

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 cipher {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 cipher {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

### `add_tag`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 cipher {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
```

```
cipher {
 add_tag => ["foo_{somefield}", "taggedy_tag"]
}
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

### *algorithm*

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The cipher algorithm

A list of supported algorithms can be obtained by

```
puts OpenSSL::Cipher.ciphers
base64
```

- Value type is [boolean](#)
- Default value is `true`

Do we have to perform a `base64` decode or encode?

If we are decrypting, `base64` decode will be done before. If we are encrypting, `base64` will be done after.

### *cipher\_padding*

- Value type is [string](#)
- There is no default value for this setting.

Cipher padding to use. Enables or disables padding.

By default encryption operations are padded using standard block padding and the padding is checked and removed when decrypting. If the pad parameter is zero then no padding is performed, the total amount of data encrypted or decrypted must then be a multiple of the block size or an error will occur.

See `EVP_CIPHER_CTX_set_padding` for further information.

We are using Openssl jRuby which uses default padding to PKCS5Padding If you want to change it, set this parameter. If you want to disable it, Set this parameter to 0

```
filter { cipher { cipher_padding => 0 } }
```

*iv* (*DEPRECATED*)

- DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.
- Value type is [string](#)
- There is no default value for this setting.

The initialization vector to use (statically hard-coded). For a random IV see the `iv_random_length` property

If `iv_random_length` is set, it takes precedence over any value set for "iv"

The cipher modes CBC, CFB, OFB and CTR all need an "initialization vector", or short, IV. ECB mode is the only mode that does not require an IV, but there is almost no legitimate use case for this mode because of the fact that it does not sufficiently hide plaintext patterns.

For AES algorithms set this to a 16 byte string.

```
filter { cipher { iv => "1234567890123456" } }
```

Deprecated: Please use `iv_random_length` instead

*iv\_random\_length*

- Value type is [number](#)
- There is no default value for this setting.

Force an random IV to be used per encryption invocation and specify the length of the random IV that will be generated via:

```
OpenSSL::Random.random_bytes(int_length)
```

If `iv_random_length` is set, it takes precedence over any value set for "iv"

Enabling this will force the plugin to generate a unique random IV for each encryption call. This random IV will be prepended to the encrypted result bytes and then base64 encoded. On decryption "`iv_random_length`" must also be set to utilize this feature. Random IV's are better than statically hardcoded IVs

For AES algorithms you can set this to a 16

```
filter { cipher { iv_random_length => 16 } }
```

`key`

- Value type is [string](#)
- There is no default value for this setting.

The key to use

If you encounter an error message at runtime containing the following:

"java.security.InvalidKeyException: Illegal key size: possibly you need to install Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files for your JRE"

Please read the following: <https://github.com/jruby/jruby/wiki/UnlimitedStrengthCrypto>

`key_pad`

```
 Value type is <>string, string>>
* Default value is `"\u0000"`
```

The character used to pad the key

`key_size`

- Value type is [number](#)
- Default value is 16

The key size to pad

It depends of the cipher algorithm. If your key doesn't need padding, don't set this parameter

Example, for AES-128, we must have 16 char long key. AES-256 = 32 chars

```
filter { cipher { key_size => 16 }
max_cipher_reuse
```

- Value type is [number](#)
- Default value is 1

If this is set the internal Cipher instance will be re-used up to @max\_cipher\_reuse times before being reset() and re-created from scratch. This is an option for efficiency where lots of data is being encrypted and decrypted using this filter. This lets the filter avoid creating new Cipher instances over and over for each encrypt/decrypt operation.

This is optional, the default is no re-use of the Cipher instance and max\_cipher\_reuse = 1 by default

## Logstash 5.2 Configuration Guide

```
filter { cipher { max_cipher_reuse => 1000 } }
mode
```

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

Encrypting or decrypting some data

Valid values are encrypt or decrypt

```
periodic_flush
```

- Value type is [boolean](#)
- Default value is false

Call the filter flush method at regular interval. Optional.

```
remove_field
```

- Value type is [array](#)
- Default value is []

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 cipher {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 cipher {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name foo\_hello if it is present. The second example would remove an additional, non-dynamic field.

```
remove_tag
```

- Value type is [array](#)
- Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the %{field} syntax.

Example:

```
filter {
 cipher {
 remove_tag => ["foo_%{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 cipher {
 remove_tag => ["foo_%{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag foo\_hello if it is present. The second example would remove a sad, unwanted tag as well.

*source*

- Value type is [string](#)
- Default value is "message"

The field to perform filter

Example, to use the @message field (default) :

```
filter { cipher { source => "message" } }
```

- Value type is [string](#)
- Default value is "message"

The name of the container to put the result

Example, to place the result into crypt :

```
filter { cipher { target => "crypt" } }
```

## clone

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

The clone filter is for duplicating events. A clone will be made for each type in the clone list. The original event is left unchanged.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
clone {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">add_tag</a>	<a href="#">array</a>	No	[]
<a href="#">clones</a>	<a href="#">array</a>	No	[]
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">periodic_flush</a>	<a href="#">boolean</a>	No	false
<a href="#">remove_field</a>	<a href="#">array</a>	No	[]
<a href="#">remove_tag</a>	<a href="#">array</a>	No	[]

## Details

### [add\\_field](#)

- Value type is [hash](#)
- Default value is { }

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 clone {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 clone {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

#### add\_tag

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 clone {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 clone {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

### `clones`

- Value type is [array](#)
- Default value is []

A new clone will be created with the given type for each type in this list.

### `enable_metric`

- Value type is [boolean](#)
- Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique ID to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

### `periodic_flush`

- Value type is [boolean](#)
- Default value is false

Call the filter flush method at regular interval. Optional.

### `remove_field`

- Value type is [array](#)
- Default value is []

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 clone {
 remove_field => ["foo_{somefield}"]
 }
}
```

```
You can also remove multiple fields at once:
filter {
 clone {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name foo\_hello if it is present. The second example would remove an additional, non-dynamic field.

### `remove_tag`

- Value type is [array](#)
- Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the %{field} syntax.

Example:

```
filter {
 clone {
 remove_tag => ["foo_{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 clone {
 remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag foo\_hello if it is present. The second example would remove a sad, unwanted tag as well.

## collate

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-filter-collate`.

Collate events by time or count.

The original goal of this filter was to merge the logs from different sources by the time of log, for example, in real-time log collection, logs can be collated by amount of 3000 logs or can be collated in 30 seconds.

The config looks like this:

```
filter {
 collate {
 size => 3000
 interval => "30s"
 order => "ascending"
 }
}
```

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
collate {
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
add_tag	<a href="#">array</a>	No	[]
count	<a href="#">number</a>	No	1000
interval	<a href="#">string</a>	No	"1m"
order	<a href="#">string</a> , one of ["ascending", "descending"]	No	"ascending"
periodic_flush	<a href="#">boolean</a>	No	false
remove_field	<a href="#">array</a>	No	[]
remove_tag	<a href="#">array</a>	No	[]

## Details

### `add_field`

- Value type is [hash](#)
- Default value is `{ }`

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 collate {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 collate {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

### `add_tag`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 collate {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
```

## Logstash 5.2 Configuration Guide

```
collate {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
}
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

### `count`

- Value type is [number](#)
- Default value is 1000

How many logs should be collated.

### `interval`

- Value type is [string](#)
- Default value is "1m"

The `interval` is the time window which how long the logs should be collated. (default `1m`)

### `order`

- Value can be any of: `ascending`, `descending`
- Default value is "`ascending`"

The `order` collated events should appear in.

### `periodic_flush`

- Value type is [boolean](#)
- Default value is `false`

Call the filter flush method at regular interval. Optional.

### `remove_field`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 collate {
 remove_field => ["foo_%{somefield}"]
 }
}
You can also remove multiple fields at once:
```

```
filter {
 collate {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

### `remove_tag`

- Value type is [array](#)
- Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 collate {
 remove_tag => ["foo_{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 collate {
 remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

## CSV

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

The CSV filter takes an event field containing CSV data, parses it, and stores it as individual fields (can optionally specify the names). This filter can also parse data with any separator, not just commas.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
csv {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">add_tag</a>	<a href="#">array</a>	No	[]
<a href="#">autogenerate_column_names</a>	<a href="#">boolean</a>	No	true
<a href="#">columns</a>	<a href="#">array</a>	No	[]
<a href="#">convert</a>	<a href="#">hash</a>	No	{ }
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">periodic_flush</a>	<a href="#">boolean</a>	No	false
<a href="#">quote_char</a>	<a href="#">string</a>	No	"\""
<a href="#">remove_field</a>	<a href="#">array</a>	No	[]
<a href="#">remove_tag</a>	<a href="#">array</a>	No	[]
<a href="#">separator</a>	<a href="#">string</a>	No	", "
<a href="#">skip_empty_columns</a>	<a href="#">boolean</a>	No	false
<a href="#">source</a>	<a href="#">string</a>	No	"message"
<a href="#">target</a>	<a href="#">string</a>	No	

## Details

### `add_field`

- Value type is [hash](#)
- Default value is `{ }`

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 csv {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 csv {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

### `add_tag`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 csv {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
```

```
csv {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
}
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

### `autogenerate_column_names`

- Value type is [boolean](#)
- Default value is `true`

Define whether column names should autogenerated or not. Defaults to true. If set to false, columns not having a header specified will not be parsed.

### `columns`

- Value type is [array](#)
- Default value is `[]`

Define a list of column names (in the order they appear in the CSV, as if it were a header line). If `columns` is not configured, or there are not enough columns specified, the default column names are `"column1"`, `"column2"`, etc. In the case that there are more columns in the data than specified in this column list, extra columns will be auto-numbered: (e.g. `"user_defined_1"`, `"user_defined_2"`, `"column3"`, `"column4"`, etc.)

### `convert`

- Value type is [hash](#)
- Default value is `{ }`

Define a set of datatype conversions to be applied to columns. Possible conversions are `integer`, `float`, `date`, `date_time`, `boolean`

### # Example:

```
filter {
 csv {
 convert => { "column1" => "integer", "column2" => "boolean" }
 }
}
```

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
periodic_flush
```

- Value type is [boolean](#)
- Default value is `false`

Call the filter flush method at regular interval. Optional.

### `quote_char`

- Value type is [string](#)
- Default value is `"\""`

Define the character used to quote CSV fields. If this is not specified the default is a double quote `".`. Optional.

### `remove_field`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 csv {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 csv {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
```

```
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

#### `remove_tag`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 csv {
 remove_tag => ["foo_%{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 csv {
 remove_tag => ["foo_%{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

#### `separator`

- Value type is [string](#)
- Default value is `,`

Define the column separator value. If this is not specified, the default is a comma `,`. Optional.

#### `skip_empty_columns`

- Value type is [boolean](#)
- Default value is `false`

Define whether empty columns should be skipped. Defaults to false. If set to true, columns containing no value will not get set.

*source*

- Value type is [string](#)
- Default value is "message"

The CSV data in the value of the `source` field will be expanded into a data structure.

*target*

- Value type is [string](#)
- There is no default value for this setting.

Define target field for placing the data. Defaults to writing to the root of the event.

## date

- Version: 3.1.3
- Released on: 2017-02-07
- [Changelog](#)

The date filter is used for parsing dates from fields, and then using that date or timestamp as the logstash timestamp for the event.

For example, syslog events usually have timestamps like this:

```
"Apr 17 09:32:01"
```

You would use the date format `MMM dd HH:mm:ss` to parse this.

The date filter is especially important for sorting events and for backfilling old data. If you don't get the date correct in your event, then searching for them later will likely sort out of order.

In the absence of this filter, logstash will choose a timestamp based on the first time it sees the event (at input time), if the timestamp is not already set in the event. For example, with file input, the timestamp is set to the time of each read.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
date {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">add_tag</a>	<a href="#">array</a>	No	[]
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">locale</a>	<a href="#">string</a>	No	

Setting	Input type	Required	Default value
<a href="#">match</a>	<a href="#">array</a>	No	[]
<a href="#">periodic flush</a>	<a href="#">boolean</a>	No	false
<a href="#">remove field</a>	<a href="#">array</a>	No	[]
<a href="#">remove tag</a>	<a href="#">array</a>	No	[]
<a href="#">tag on failure</a>	<a href="#">array</a>	No	[ "_dateparsefailure" ]
<a href="#">target</a>	<a href="#">string</a>	No	"@timestamp"
<a href="#">timezone</a>	<a href="#">string</a>	No	

## Details

### [add\\_field](#)

- Value type is [hash](#)
- Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the %{field}.

Example:

```
filter {
 date {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 date {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

### [add\\_tag](#)

- Value type is [array](#)

- Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 date {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 date {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

#### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

#### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

#### `locale`

- Value type is [string](#)
- There is no default value for this setting.

Specify a locale to be used for date parsing using either IETF-BCP47 or POSIX language tag. Simple examples are `en,en-US` for BCP47 or `en_US` for POSIX.

The locale is mostly necessary to be set for parsing month names (pattern with `MMM`) and weekday names (pattern with `EEE`).

If not specified, the platform default will be used but for non-english platform default an english parser will also be used as a fallback mechanism.

#### `match`

- Value type is [array](#)
- Default value is `[]`

An array with field name first, and format patterns following, `[ field, formats... ]`

If your time field has multiple possible formats, you can do this:

```
match => ["logdate", "MMM dd yyyy HH:mm:ss",
 "MMM d yyyy HH:mm:ss", "ISO8601"]
```

The above will match a syslog (rfc3164) or iso8601 timestamp.

There are a few special exceptions. The following format literals exist to help you save time and ensure correctness of date parsing.

- ISO8601 - should parse any valid ISO8601 timestamp, such as `2011-04-19T03:44:01.103Z`
- UNIX - will parse **float** or **int** value expressing unix time in seconds since epoch like `1326149001.132` as well as `1326149001`
- UNIX\_MS - will parse **int** value expressing unix time in milliseconds since epoch like `1366125117000`
- TAI64N - will parse tai64n time values

For example, if you have a field `logdate`, with a value that looks like `Aug 13 2010 00:03:44`, you would use this configuration:

```
filter {
 date {
 match => ["logdate", "MMM dd yyyy HH:mm:ss"]
 }
}
```

If your field is nested in your structure, you can use the nested syntax `[foo] [bar]` to match its value. For more information, please refer to [the section called “Field References”](#)

## More details on the syntax

The syntax used for parsing date and time text uses letters to indicate the kind of time value (month, minute, etc), and a repetition of letters to indicate the form of that value (2-digit month, full month name, etc).

Here's what you can use to parse dates and times:

y year

YYYY

full year number. Example: 2015.

yy

two-digit year. Example: 15 for the year 2015.

M month of the year

M

minimal-digit month. Example: 1 for January and 12 for December.

MM

two-digit month. zero-padded if needed. Example: 01 for January and 12 for December

MMM

abbreviated month text. Example: Jan for January. Note: The language used depends on your locale. See the `locale` setting for how to change the language.

MMMM

full month text, Example: January. Note: The language used depends on your locale.

d day of the month

d

minimal-digit day. Example: 1 for the 1st of the month.

dd

two-digit day, zero-padded if needed. Example: 01 for the 1st of the month.

H hour of the day (24-hour clock)

H

minimal-digit hour. Example: 0 for midnight.

HH

two-digit hour, zero-padded if needed. Example: 00 for midnight.

m minutes of the hour (60 minutes per hour)

m

minimal-digit minutes. Example: 0.

mm

two-digit minutes, zero-padded if needed. Example: 00.

s seconds of the minute (60 seconds per minute)

s

minimal-digit seconds. Example: 0.

ss

two-digit seconds, zero-padded if needed. Example: 00.

S fraction of a second **Maximum precision is milliseconds (sss). Beyond that, zeroes are appended.**

S

tenths of a second. Example: 0 for a subsecond value 012

SS

hundredths of a second. Example: 01 for a subsecond value 01

SSS

thousandths of a second. Example: 012 for a subsecond value 012

Z time zone offset or identity

Z

Timezone offset structured as HHmm (hour and minutes offset from Zulu/UTC). Example: -0700.

ZZ

Timezone offset structured as HH:mm (colon in between hour and minute offsets). Example: -07:00.

ZZZ

Timezone identity. Example: America/Los\_Angeles. Note: Valid IDs are listed on the [Joda.org available time zones page](#).

**z** time zone names. **Time zone names (z) cannot be parsed.**

**w** week of the year

**w**

minimal-digit week. Example: 1.

**ww**

two-digit week, zero-padded if needed. Example: 01.

**D** day of the year

**e** day of the week (number)

**E** day of the week (text)

**E, EE, EEE**

Abbreviated day of the week. Example: Mon, Tue, Wed, Thu, Fri, Sat, Sun. Note: The actual language of this will depend on your locale.

**EEEE**

The full text day of the week. Example: Monday, Tuesday, ... Note: The actual language of this will depend on your locale.

For non-formatting syntax, you'll need to put single-quote characters around the value. For example, if you were parsing ISO8601 time, "2015-01-01T01:12:23" that little "T" isn't a valid time format, and you want to say "literally, a T", your format would be this: "yyyy-MM-dd'T'HH:mm:ss"

Other less common date units, such as era (G), century (C), am/pm (a), and # more, can be learned about on the [joda-time documentation](#).

#### *periodic\_flush*

- Value type is [boolean](#)
- Default value is `false`

Call the filter flush method at regular interval. Optional.

#### *remove\_field*

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 date {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 date {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name foo\_hello if it is present. The second example would remove an additional, non-dynamic field.

#### [remove\\_tag](#)

- Value type is [array](#)
- Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the %{field} syntax.

Example:

```
filter {
 date {
 remove_tag => ["foo_{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 date {
 remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag foo\_hello if it is present. The second example would remove a sad, unwanted tag as well.

#### [tag\\_on\\_failure](#)

- Value type is [array](#)
- Default value is ["\_dateparsefailure"]

Append values to the tags field when there has been no successful match

*target*

- Value type is [string](#)
- Default value is "@timestamp"

Store the matching timestamp into the given target field. If not provided, default to updating the @timestamp field of the event.

*timezone*

- Value type is [string](#)
- There is no default value for this setting.

Specify a time zone canonical ID to be used for date parsing. The valid IDs are listed on the [Joda.org available time zones page](#). This is useful in case the time zone cannot be extracted from the value, and is not the platform default. If this is not specified the platform default will be used. Canonical ID is good as it takes care of daylight saving time for you For example, America/Los\_Angeles or Europe/Paris are valid IDs. This field can be dynamic and include parts of the event using the %{field} syntax

## de\_dot

- Version: 1.0.0
- Released on: 2016-10-26
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
`bin/logstash-plugin install logstash-filter-de_dot.`

This filter *appears* to rename fields by replacing . characters with a different separator. In reality, it's a somewhat expensive filter that has to copy the source field contents to a new destination field (whose name no longer contains dots), and then remove the corresponding source field.

It should only be used if no other options are available.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
de_dot { }
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
add_tag	<a href="#">array</a>	No	[]
enable_metric	<a href="#">boolean</a>	No	true
fields	<a href="#">array</a>	No	
id	<a href="#">string</a>	No	
nested	<a href="#">boolean</a>	No	false
periodic_flush	<a href="#">boolean</a>	No	false
remove_field	<a href="#">array</a>	No	[]
remove_tag	<a href="#">array</a>	No	[]
separator	<a href="#">string</a>	No	"_"

## Details

### `add_field`

- Value type is [hash](#)
- Default value is `{ }`

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 de_dot {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 de_dot {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

### `add_tag`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 de_dot {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
```

```
de_dot {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
}
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `fields`

- Value type is [array](#)
- There is no default value for this setting.

The `fields` array should contain a list of known fields to act on. If undefined, all top-level fields will be checked. Sub-fields must be manually specified in the array. For example:

`["field.suffix", "[foo] [bar.suffix]"]` will result in "field\_suffix" and nested or sub field `["foo"]["bar_suffix"]`

This is an expensive operation.

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

`nested`

- Value type is [boolean](#)
- Default value is `false`

If `nested` is *true*, then create sub-fields instead of replacing dots with a different separator.

`periodic_flush`

- Value type is [boolean](#)
- Default value is `false`

Call the filter flush method at regular interval. Optional.

`remove_field`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
  de_dot {
    remove_field => [ "foo_{somefield}" ]
  }
}
# You can also remove multiple fields at once:
filter {
  de_dot {
    remove_field => [ "foo_{somefield}", "my_extraneous_field" ]
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  de_dot {
    remove_tag => [ "foo_{somefield}" ]
  }
}
# You can also remove multiple tags at once:
filter {
  de_dot {
```

```
        remove_tag => [ "foo_%{somefield}", "sad_unwanted_tag"]
    }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

[separator](#)

- Value type is [string](#)
- Default value is "`_`"

Replace dots with this value.

dissect

- Version: 1.0.8
- Released on: 2016-10-22
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
`bin/logstash-plugin install logstash-filter-dissect`.

Dissect or how to de-structure text

The Dissect filter is a kind of split operation. Unlike a regular split operation where one delimiter is applied to the whole string, this operation applies a set of delimiters # to a string value. Dissect does not use regular expressions and is very fast. However, if the structure of your text varies from line to line then Grok is more suitable. There is a hybrid case where Dissect can be used to de-structure the section of the line that is reliably repeated and then Grok can be used on the remaining field values with # more regex predictability and less overall work to do.

A set of fields and delimiters is called a **dissection**.

The dissection is described using a set of `%{ }` sections:

```
%{a} - %{b} - %{c}
```

A **field** is the text from `%` to `}` inclusive.

A **delimiter** is the text between `}` and `%` characters.

delimiters can't contain these `}{%` characters.

The config might look like this:

```
filter {
  dissect {
    mapping => {
      "message" => "%{ts} %{+ts} %{+ts} %{src} %{ } %{prog} [%{pid}]: %{msg}"
    }
  }
}
```

When dissecting a string from left to right, text is captured upto the first delimiter - this captured text is stored in the first field. This is repeated for each field/# delimiter pair thereafter until the last delimiter is reached, then **the remaining text is stored in the last field**.

The Key: The key is the text between the `%{` and `}`, exclusive of the `?`, `+`, `&` prefixes and the ordinal suffix. `%{?aaa}` - key is `aaa` `%{+bbb/3}` - key is `bbb` `%{&ccc}` - key is `ccc`

Normal field notation: The found value is added to the Event using the key. `%{some_field}` - a normal field has no prefix or suffix

Skip field notation: The found value is stored internally but not added to the Event. The key, if supplied, is prefixed with a `?`.

`%{ }` is an empty skip field.

`%{?foo}` is a named skip field.

Append field notation: The value is appended to another value or stored if its the first field seen. The key is prefixed with a `+`. The final value is stored in the Event using the key.

The delimiter found before the field is appended with the value. If no delimiter is found before the field, a single space character is used.

`%{+some_field}` is an append field. `%{+some_field/2}` is an append field with an order modifier.

An order modifier, `/digits`, allows one to reorder the append sequence. e.g. for a text of `1 2 3 go`, this `%{+a/2} %{+a/1} %{+a/4} %{+a/3}` will build a key/value of `a => 2 1 go 3` Append fields without an order modifier will append in declared order. e.g. for a text of `1 2 3 go`, this `%{a} %{b} %{+a}` will build two key/values of `a => 1 3 go, b => 2`

Indirect field notation: The found value is added to the Event using the found value of another field as the key. The key is prefixed with a `&`. `%{&some_field}` - an indirect field where the key is indirectly sourced from the value of `some_field`. e.g. for a text of `error: some_error, some_description`, this `error: %{?err}, %{&err}` will build a key/value of `some_error => description`.

for append and indirect field the key can refer to a field that already exists in the event before dissection.

use a Skip field if you do not want the indirection key/value stored.

e.g. for a text of `google: 77.98`, this `%{?a}: %{&a}` will build a key/value of `google => 77.98`.

append and indirect cannot be combined and will fail validation. `%{+&something}` - will add a value to the `&something` key, probably not the intended outcome. `%{&+something}` will add a value to the `+something` key, again probably unintended.

Delimiter repetition: In the source text if a field has variable width padded with delimiters, the padding will be ignored. e.g. for texts of:

```
00000043 ViewReceiver I  
000000b3 Peer I
```

with a dissection of `%{a} %{b} %{c}`; the padding is ignored, `event.get([c]) -> "I"`

You probably want to use this filter inside an `if` block. This ensures that the event contains a field value with a suitable structure for the dissection.

For example...

```
filter {  
  if [type] == "syslog" or "syslog" in [tags] {  
    dissect {  
      mapping => {  
        "message" => "%{ts} %{+ts} %{+ts} %{src} %{prog} [%{pid}]: %{msg}"  
      }  
    }  
  }  
}
```

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
dissect {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
add_tag	array	No	[]
convert_datatype	hash	No	{ }
enable_metric	boolean	No	true
id	string	No	
mapping	hash	No	{ }
periodic_flush	boolean	No	false
remove_field	array	No	[]
remove_tag	array	No	[]
tag_on_failure	array	No	["_dissectfailure"]

Details

[add_field](#)

- Value type is [hash](#)
- Default value is { }

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
  dissect {
    add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
  }
  # You can also add multiple fields at once:
  filter {
    dissect {
      add_field => {
        "foo_%{somefield}" => "Hello world, from %{host}"
        "new_field" => "new_static_value"
      }
    }
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

`add_tag`

- Value type is [array](#)
- Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  dissect {
    add_tag => [ "foo_%{somefield}" ]
  }
}
# You can also add multiple tags at once:
filter {
  dissect {
    add_tag => [ "foo_%{somefield}", "taggedy_tag" ]
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`convert_datatype`

- Value type is [hash](#)
- Default value is {}

With this setting `int` and `float` datatype conversions can be specified. These will be done after all `mapping` dissections have taken place. Feel free to use this setting on its own without a `mapping` section.

For example

```
filter {
  dissect {
    convert_datatype => {
      cpu => "float"
      code => "int"
    }
  }
}
```

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

### *mapping*

- Value type is [hash](#)
- Default value is `{}`

A hash of dissections of `field => value` A later dissection can be done on values from a previous dissection or they can be independent.

For example

```
filter {
 dissect {
 mapping => {
 "message" => "%{field1} %{field2} %{description}"
 "description" => "%{field3} %{field4} %{field5}"
 }
 }
}
```

This is useful if you want to keep the field `description` but also dissect it some more.

### *periodic\_flush*

- Value type is [boolean](#)
- Default value is `false`

Call the filter flush method at regular interval. Optional.

### *remove\_field*

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 dissect {
 remove_field => ["foo_%{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 dissect {
 remove_field => ["foo_%{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name foo\_hello if it is present. The second example would remove an additional, non-dynamic field.

#### [remove\\_tag](#)

- Value type is [array](#)
- Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the %{field} syntax.

Example:

```
filter {
 dissect {
 remove_tag => ["foo_%{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 dissect {
 remove_tag => ["foo_%{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag foo\_hello if it is present. The second example would remove a sad, unwanted tag as well.

#### [tag\\_on\\_failure](#)

- Value type is [array](#)
- Default value is ["\_dissectfailure"]

Append values to the tags field when dissection fails



## dns

- Version: 3.0.3
- Released on: 2016-07-14
- [Changelog](#)

The DNS filter performs a lookup (either an A record/CNAME record lookup or a reverse lookup at the PTR record) on records specified under the `reverse` arrays or respectively under the `resolve` arrays.

The config should look like this:

```
filter {
 dns {
 reverse => ["source_host", "field_with_address"]
 resolve => ["field_with_fqdn"]
 action => "replace"
 }
}
```

This filter, like all filters, only processes 1 event at a time, so the use of this plugin can significantly slow down your pipeline's throughput if you have a high latency network. By way of example, if each DNS lookup takes 2 milliseconds, the maximum throughput you can achieve with a single filter worker is 500 events per second (1000 milliseconds / 2 milliseconds).

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
dns { }
```

Available configuration options:

Setting	Input type	Required	Default value
action	<a href="#">string</a> , one of ["append", "replace"]	No	"append"
add_field	<a href="#">hash</a>	No	{ }
add_tag	<a href="#">array</a>	No	[]
enable_metric	<a href="#">boolean</a>	No	true
failed_cache_size	<a href="#">number</a>	No	0

Setting	Input type	Required	Default value
failed_cache_ttl	<a href="#">number</a>	No	5
hit_cache_size	<a href="#">number</a>	No	0
hit_cache_ttl	<a href="#">number</a>	No	60
hostsfile	<a href="#">array</a>	No	
id	<a href="#">string</a>	No	
max_retries	<a href="#">number</a>	No	2
nameserver	<a href="#">array</a>	No	
periodic_flush	<a href="#">boolean</a>	No	false
remove_field	<a href="#">array</a>	No	[]
remove_tag	<a href="#">array</a>	No	[]
resolve	<a href="#">array</a>	No	
reverse	<a href="#">array</a>	No	
timeout	<a href="#">number</a>	No	0.5

## Details

### *action*

- Value can be any of: append, replace
- Default value is "append"

Determine what action to do: append or replace the values in the fields specified under `reverse` and `resolve`.

### *add\_field*

- Value type is [hash](#)
- Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 dns {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
```

## Logstash 5.2 Configuration Guide

```
You can also add multiple fields at once:
filter {
 dns {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

### `add_tag`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 dns {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 dns {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `failed_cache_size`

- Value type is [number](#)

- Default value is 0

cache size for failed requests

*failed\_cache\_ttl*

- Value type is [number](#)
- Default value is 5

how long to cache failed requests (in seconds)

*hit\_cache\_size*

- Value type is [number](#)
- Default value is 0

set the size of cache for successful requests

*hit\_cache\_ttl*

- Value type is [number](#)
- Default value is 60

how long to cache successful requests (in seconds)

*hostsfile*

- Value type is [array](#)
- There is no default value for this setting.

Use custom hosts file(s). For example: `["/var/db/my_custom_hosts"]`

*id*

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

### *max\_retries*

- Value type is [number](#)
- Default value is 2

number of times to retry a failed resolve/reverse

### *nameserver*

- Value type is [array](#)
- There is no default value for this setting.

Use custom nameserver(s). For example: ["8.8.8.8", "8.8.4.4"]

### *periodic\_flush*

- Value type is [boolean](#)
- Default value is false

Call the filter flush method at regular interval. Optional.

### *remove\_field*

- Value type is [array](#)
- Default value is []

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 dns {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 dns {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name foo\_hello if it is present. The second example would remove an additional, non-dynamic field.

### *remove\_tag*

- Value type is [array](#)
- Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 dns {
 remove_tag => ["foo_%{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 dns {
 remove_tag => ["foo_%{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

#### `resolve`

- Value type is [array](#)
- There is no default value for this setting.

Forward resolve one or more fields.

#### `reverse`

- Value type is [array](#)
- There is no default value for this setting.

TODO(sissel): The timeout limitation does seem to be fixed in here: [#](http://redmine.ruby-lang.org/issues/5100) but isn't currently in JRuby. TODO(sissel): make action required? This was always the intent, but it due to a typo it was never enforced. Thus the default behavior in past versions was append by accident. Reverse resolve one or more fields.

#### `timeout`

- Value type is [number](#)
- Default value is 0.5

`resolv` calls will be wrapped in a timeout instance

## drop

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

Drop filter.

Drops everything that gets to this filter.

This is best used in combination with conditionals, for example:

```
filter {
 if [loglevel] == "debug" {
 drop { }
 }
}
```

The above will only pass events to the drop filter if the loglevel field is `debug`. This will cause all events matching to be dropped.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
drop { }
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">add_tag</a>	<a href="#">array</a>	No	[]
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">percentage</a>	<a href="#">number</a>	No	100
<a href="#">periodic_flush</a>	<a href="#">boolean</a>	No	false
<a href="#">remove_field</a>	<a href="#">array</a>	No	[]
<a href="#">remove_tag</a>	<a href="#">array</a>	No	[]

## Details

### `add_field`

- Value type is [hash](#)
- Default value is `{ }`

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 drop {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 drop {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

### `add_tag`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 drop {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
```

```
drop {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
}
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
percentage
```

- Value type is [number](#)
- Default value is `100`

Drop all the events within a pre-configured percentage.

This is useful if you just need a percentage but not the whole.

Example, to only drop around 40% of the events that have the field `loglevel` with value "debug".

```
filter {
 if [loglevel] == "debug" {
 drop {
 percentage => 40
 }
 }
}
```

*periodic\_flush*

- Value type is [boolean](#)
- Default value is `false`

Call the filter flush method at regular interval. Optional.

*remove\_field*

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 drop {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 drop {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

*remove\_tag*

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 drop {
 remove_tag => ["foo_{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 drop {
 remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

## elapsed

- Version: 4.0.0
- Released on: 2016-11-29
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
`bin/logstash-plugin install logstash-filter-elapsed`.

### elapsed filter

This filter tracks a pair of start/end events and calculates the elapsed time between them. The elapsed filter tracks a pair of start/end events and uses their timestamps to calculate the elapsed time between them.

The filter has been developed to track the execution time of processes and other long tasks.

The configuration looks like this:

```
filter {
 elapsed {
 start_tag => "start event tag"
 end_tag => "end event tag"
 unique_id_field => "id field name"
 timeout => seconds
 new_event_on_match => true/false
 }
}
```

The events managed by this filter must have some particular properties. The event describing the start of the task (the "start event") must contain a tag equal to `start_tag`. On the other side, the event describing the end of the task (the "end event") must contain a tag equal to `end_tag`. Both these two kinds of event need to own an ID field which identify uniquely that particular task. The name of this field is stored in `unique_id_field`.

You can use a Grok filter to prepare the events for the elapsed filter. An example of configuration can be:

```
filter {
 grok {
 match => { "message" => "%{TIMESTAMP_ISO8601} START id: (?<task_id>.*)" }
 add_tag => ["taskStarted"]
 }
 grok {
 match => { "message" => "%{TIMESTAMP_ISO8601} END id: (?<task_id>.*)" }
 add_tag => ["taskTerminated"]
 }
 elapsed {
 start_tag => "taskStarted"
 end_tag => "taskTerminated"
```

```

 unique_id_field => "task_id"
 }
}

```

The elapsed filter collects all the "start events". If two, or more, "start events" have the same ID, only the first one is recorded, the others are discarded.

When an "end event" matching a previously collected "start event" is received, there is a match. The configuration property `new_event_on_match` tells where to insert the elapsed information: they can be added to the "end event" or a new "match event" can be created. Both events store the following information:

- the tags `elapsed` and `elapsed_match`
- the field `elapsed_time` with the difference, in seconds, between the two events timestamps
- an ID filed with the task ID
- the field `elapsed_timestamp_start` with the timestamp of the start event

If the "end event" does not arrive before "timeout" seconds, the "start event" is discarded and an "expired event" is generated. This event contains:

- the tags `elapsed` and `elapsed_expired_error`
- a field called `elapsed_time` with the age, in seconds, of the "start event"
- an ID filed with the task ID
- the field `elapsed_timestamp_start` with the timestamp of the "start event"

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```

elapsed {
 end_tag => ...
 start_tag => ...
 unique_id_field => ...
}

```

Available configuration options:

Setting	Input type	Required	Default value
<code>add_field</code>	<a href="#">hash</a>	No	{ }
<code>add_tag</code>	<a href="#">array</a>	No	[]
<code>enable_metric</code>	<a href="#">boolean</a>	No	true

Setting	Input type	Required	Default value
end_tag	<a href="#">string</a>	Yes	
id	<a href="#">string</a>	No	
new_event_on_match	<a href="#">boolean</a>	No	false
periodic_flush	<a href="#">boolean</a>	No	false
remove_field	<a href="#">array</a>	No	[]
remove_tag	<a href="#">array</a>	No	[]
start_tag	<a href="#">string</a>	Yes	
timeout	<a href="#">number</a>	No	1800
unique_id_field	<a href="#">string</a>	Yes	

## Details

### [add\\_field](#)

- Value type is [hash](#)
- Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 elapsed {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 elapsed {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

### `add_tag`

- Value type is [array](#)
- Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 elapsed {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 elapsed {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `end_tag`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The name of the tag identifying the "end event"

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique ID to the plugin instance, this ID is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

new_event_on_match

- Value type is [boolean](#)
- Default value is `false`

This property manage what to do when an "end event" matches a "start event". If it's set to `false` (default value), the elapsed information are added to the "end event"; if it's set to `true` a new "match event" is created.

periodic_flush

- Value type is [boolean](#)
- Default value is `false`

Call the filter flush method at regular interval. Optional.

remove_field

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
    elapsed {
        remove_field => [ "foo_{somefield}" ]
    }
}
# You can also remove multiple fields at once:
filter {
    elapsed {
        remove_field => [ "foo_{somefield}", "my_extraneous_field" ]
    }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

remove_tag

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
    elapsed {
        remove_tag => [ "foo_%{somefield}" ]
    }
}
# You can also remove multiple tags at once:
filter {
    elapsed {
        remove_tag => [ "foo_%{somefield}", "sad_unwanted_tag" ]
    }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

`start_tag`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The name of the tag identifying the "start event"

`timeout`

- Value type is [number](#)
- Default value is 1800

The amount of seconds after an "end event" can be considered lost. The corresponding "start event" is discarded and an "expired event" is generated. The default value is 30 minutes (1800 seconds).

`unique_id_field`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The name of the field containing the task ID. This value must uniquely identify the task in the system, otherwise it's impossible to match the couple of events.

elasticsearch

- Version: 3.1.0
- Released on: 2016-12-16
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
 bin/logstash-plugin install logstash-filter-elasticsearch.

Search Elasticsearch for a previous log event and copy some fields from it into the current event. Below are two complete examples of how this filter might be used.

The first example uses the legacy *query* parameter where the user is limited to an Elasticsearch query_string. Whenever logstash receives an "end" event, it uses this elasticsearch filter to find the matching "start" event based on some operation identifier. Then it copies the @timestamp field from the "start" event into a new field on the "end" event. Finally, using a combination of the "date" filter and the "ruby" filter, we calculate the time duration in hours between the two events.

```
if [type] == "end" {
  elasticsearch {
    hosts => ["es-server"]
    query => "type:start AND operation:%{[opid]}"
    fields => { "@timestamp" => "started" }
  }

  date {
    match => "[started]", "ISO8601"
    target => "[started]"
  }

  ruby {
    code => "event['duration_hrs'] = (event['@timestamp'] - event['started']) / 3600 rescue nil"
  }
}
```

The example below reproduces the above example but utilises the query_template. This query_template represents a full Elasticsearch query DSL and supports the standard Logstash field substitution syntax. The example below issues the same query as the first example but uses the template shown.

```
if [type] == "end" {
  elasticsearch {
    hosts => ["es-server"]
    query_template => "template.json"
  }

  date {
    match => "[started]", "ISO8601"
    target => "[started]"
  }
```

```

        }

    ruby {
        code => "event['duration_hrs'] = (event['@timestamp'] -
event['started']) / 3600 rescue nil"
    }
}

```

template.json:

```
{
    "query": {
        "query_string": {
            "query": "type:start AND operation:#{@pid}"
        }
    },
    "_source": ["@timestamp", "started"]
}
```

As illustrated above, through the use of 'opid', fields from the Logstash events can be referenced within the template.

The template will be populated per event prior to being used to query Elasticsearch.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
elasticsearch {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
add_tag	array	No	[]
ca_file	a valid filesystem path	No	
enable_metric	boolean	No	true
enable_sort	boolean	No	true
fields	array	No	{ }
hosts	array	No	["localhost:9200"]

Setting	Input type	Required	Default value
id	string	No	
index	string	No	""
password	password	No	
periodic_flush	boolean	No	false
query	string	No	
query_template	string	No	
remove_field	array	No	[]
remove_tag	array	No	[]
result_size	number	No	1
sort	string	No	"@timestamp:desc"
ssl	boolean	No	false
tag_on_failure	array	No	["_elasticsearch_lookup_failure"]
user	string	No	

Details

`add_field`

- Value type is [hash](#)
- Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
  elasticsearch {
    add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
  }
}
# You can also add multiple fields at once:
filter {
  elasticsearch {
    add_field => {
      "foo_%{somefield}" => "Hello world, from %{host}"
      "new_field" => "new_static_value"
    }
  }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

`add_tag`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  elasticsearch {
    add_tag => [ "foo_%{somefield}" ]
  }
}
# You can also add multiple tags at once:
filter {
  elasticsearch {
    add_tag => [ "foo_%{somefield}", "taggedy_tag" ]
  }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`ca_file`

- Value type is [path](#)
- There is no default value for this setting.

SSL Certificate Authority file

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`enable_sort`

- Value type is [boolean](#)
- Default value is `true`

Whether results should be sorted or not

fields

- Value type is [array](#)
- Default value is `{ }`

Array of fields to copy from old event (found via elasticsearch) into new event

hosts

- Value type is [array](#)
- Default value is `["localhost:9200"]`

List of elasticsearch hosts to use for querying.

id

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

*index*

- Value type is [string](#)
- Default value is `""`

Comma-delimited list of index names to search; use `_all` or empty string to perform the operation on all indices

*password*

- Value type is [password](#)
- There is no default value for this setting.

Basic Auth - password

*periodic\_flush*

- Value type is [boolean](#)
- Default value is `false`

Call the filter flush method at regular interval. Optional.

*query*

- Value type is [string](#)
- There is no default value for this setting.

Elasticsearch query string. Read the Elasticsearch query string documentation. for more info at: <https://www.elastic.co/guide/en/elasticsearch/reference/master/query-dsl-query-string-query.html#query-string-syntax>

*query\_template*

- Value type is [string](#)
- There is no default value for this setting.

File path to elasticseach query in DSL format. Read the Elasticsearch query documentation for more info at: <https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html>

*remove\_field*

- Value type is [array](#)
- Default value is []

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 elasticsearch {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 elasticsearch {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

*remove\_tag*

- Value type is [array](#)
- Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 elasticsearch {
 remove_tag => ["foo_{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 elasticsearch {
 remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag foo\_hello if it is present. The second example would remove a sad, unwanted tag as well.

*result\_size*

- Value type is [number](#)
- Default value is 1

How many results to return

*sort*

- Value type is [string](#)
- Default value is "@timestamp:desc"

Comma-delimited list of <field>:<direction> pairs that define the sort order

*ssl*

- Value type is [boolean](#)
- Default value is false

SSL

*tag\_on\_failure*

- Value type is [array](#)
- Default value is ["\_elasticsearch\_lookup\_failure"]

Tags the event on failure to look up geo information. This can be used in later analysis.

*user*

- Value type is [string](#)
- There is no default value for this setting.

Basic Auth - username

## environment

- Version: 3.0.0
- Released on: 2016-11-17
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-filter-environment`.

This filter stores environment variables as subfields in the `@metadata` field. You can then use these values in other parts of the pipeline.

Adding environment variables is as easy as: `filter { environment { add_metadata_from_env => { "field_name" => "ENV_VAR_NAME" } } }`

Accessing stored environment variables is now done through the `@metadata` field:

```
["@metadata"] ["field_name"]
```

This would reference field `field_name`, which in the above example references the `ENV_VAR_NAME` environment variable.

Previous versions of this plugin put the environment variables as fields at the root level of the event. Current versions make use of the `@metadata` field, as outlined. You have to change `add_field_from_env` in the older versions to `add_metadata_from_env` in the newer version.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
environment {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>add_field</code>	<a href="#">hash</a>	No	<code>{ }</code>

Setting	Input type	Required	Default value
add_metadata_from_env	<a href="#">hash</a>	No	{ }
add_tag	<a href="#">array</a>	No	[]
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	
periodic_flush	<a href="#">boolean</a>	No	false
remove_field	<a href="#">array</a>	No	[]
remove_tag	<a href="#">array</a>	No	[]

## Details

### [add\\_field](#)

- Value type is [hash](#)
- Default value is { }

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 environment {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 environment {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

### [add\\_metadata\\_from\\_env](#)

- Value type is [hash](#)

- Default value is {}

Specify a hash of field names and the environment variable name with the value you want imported into Logstash. For example:

```
add_metadata_from_env => { "field_name" => "ENV_VAR_NAME" }
```

or

```
add_metadata_from_env => {
 "field1" => "ENV1"
 "field2" => "ENV2"
 # "field_n" => "ENV_n"
}
add_tag
```

- Value type is [array](#)
- Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 environment {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 environment {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

[enable\\_metric](#)

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

[id](#)

- Value type is [string](#)

- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

`periodic_flush`

- Value type is [boolean](#)
- Default value is `false`

Call the filter flush method at regular interval. Optional.

`remove_field`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
  environment {
    remove_field => [ "foo_{somefield}" ]
  }
}
# You can also remove multiple fields at once:
filter {
  environment {
    remove_field => [ "foo_{somefield}", "my_extraneous_field" ]
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

Logstash 5.2 Configuration Guide

```
filter {
  environment {
    remove_tag => [ "foo_{somefield}" ]
  }
}
# You can also remove multiple tags at once:
filter {
  environment {
    remove_tag => [ "foo_{somefield}", "sad_unwanted_tag" ]
  }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

extractnumbers

- Version: 3.0.0
- Released on: 2016-12-07
- [Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
`bin/logstash-plugin install logstash-filter-extractnumbers.`

This filter automatically extracts all numbers found inside a string

This is useful when you have lines that don't match a grok pattern or use json but you still need to extract numbers.

Each numbers is returned in a `@fields.intX` or `@fields.floatX` field where X indicates the position in the string.

The fields produced by this filter are extra useful used in combination with kibana number plotting features.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
extractnumbers {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
add_tag	array	No	[]
enable_metric	boolean	No	true
id	string	No	
periodic_flush	boolean	No	false
remove_field	array	No	[]
remove_tag	array	No	[]
source	string	No	"message"

Details

`add_field`

- Value type is [hash](#)
- Default value is `{ }`

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
    extractnumbers {
        add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
    }
}
# You can also add multiple fields at once:
filter {
    extractnumbers {
        add_field => {
            "foo_%{somefield}" => "Hello world, from %{host}"
            "new_field" => "new_static_value"
        }
    }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

`add_tag`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
    extractnumbers {
        add_tag => [ "foo_%{somefield}" ]
    }
}
# You can also add multiple tags at once:
filter {
```

Logstash 5.2 Configuration Guide

```
extractnumbers {  
    add_tag => [ "foo_%{somefield}", "taggedy_tag"]  
}  
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

### `periodic_flush`

- Value type is [boolean](#)
- Default value is `false`

Call the filter flush method at regular interval. Optional.

### `remove_field`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 extractnumbers {
 remove_field => ["foo_%{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
```

```
extractnumbers {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
}
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

### `remove_tag`

- Value type is [array](#)
- Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 extractnumbers {
 remove_tag => ["foo_{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 extractnumbers {
 remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

### `source`

- Value type is [string](#)
- Default value is "message"

The source field for the data. By default is message.

## fingerprint

- Version: 3.0.2
- Released on: 2016-07-14
- [Changelog](#)

Fingerprint fields using by replacing values with a consistent hash.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
fingerprint {
 method => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">add_tag</a>	<a href="#">array</a>	No	[]
<a href="#">base64encode</a>	<a href="#">boolean</a>	No	false
<a href="#">concatenate_sources</a>	<a href="#">boolean</a>	No	false
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">key</a>	<a href="#">string</a>	No	
<a href="#">method</a>	<a href="#">string</a> , one of ["SHA1", "SHA256", "SHA384", "SHA512", "MD5", "MURMUR3", "IPV4_NETWORK", "UUID", "PUNCTUATION"]	Yes	"SHA1"
<a href="#">periodic_flush</a>	<a href="#">boolean</a>	No	false
<a href="#">remove_field</a>	<a href="#">array</a>	No	[]
<a href="#">remove_tag</a>	<a href="#">array</a>	No	[]
<a href="#">source</a>	<a href="#">array</a>	No	"message"
<a href="#">target</a>	<a href="#">string</a>	No	"fingerprint"

## Details

[add\\_field](#)

- Value type is [hash](#)
- Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 fingerprint {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 fingerprint {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

[add\\_tag](#)

- Value type is [array](#)
- Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 fingerprint {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 fingerprint {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

}

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

### `base64encode`

- Value type is [boolean](#)
- Default value is `false`

When set to `true`, `SHA1`, `SHA256`, `SHA384`, `SHA512` and `MD5` fingerprint methods will be returned base64 encoded rather than hex encoded.

### `concatenate_sources`

- Value type is [boolean](#)
- Default value is `false`

When set to `true`, we concatenate the values of all fields into 1 string like the old checksum filter.

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

### `key`

- Value type is [string](#)
- There is no default value for this setting.

When used with `IPV4_NETWORK` method fill in the subnet prefix length Not required for `MURMUR3` or `UUID` methods With other methods fill in the `HMAC` key

### `method`

- This is a required setting.
- Value can be any of: `SHA1`, `SHA256`, `SHA384`, `SHA512`, `MD5`, `MURMUR3`, `IPV4_NETWORK`, `UUID`, `PUNCTUATION`
- Default value is "SHA1"

### Fingerprint method

#### `periodic_flush`

- Value type is [boolean](#)
- Default value is `false`

Call the filter flush method at regular interval. Optional.

#### `remove_field`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 fingerprint {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 fingerprint {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

#### `remove_tag`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 fingerprint {
 remove_tag => ["foo_%{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 fingerprint {
 remove_tag => ["foo_%{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag foo\_hello if it is present. The second example would remove a sad, unwanted tag as well.

*source*

- Value type is [array](#)
- Default value is "message"

Source field(s)

*target*

- Value type is [string](#)
- Default value is "fingerprint"

Target field. will overwrite current value of a field if it exists.

## geoip

- Version: 4.0.4
- Released on: 2016-11-30
- [Changelog](#)

The GeoIP filter adds information about the geographical location of IP addresses, based on data from the Maxmind GeoLite2 database.

A `[geoip][location]` field is created if the GeoIP lookup returns a latitude and longitude. The field is stored in [GeoJSON](#) format. Additionally, the default Elasticsearch template provided with the `elasticsearch` [output](#) maps the `[geoip][location]` field to an [Elasticsearch geo\\_point](#).

As this field is a `geo_point` *and* it is still valid GeoJSON, you get the awesomeness of Elasticsearch's geospatial query, facet and filter functions and the flexibility of having GeoJSON for all other applications (like Kibana's map visualization).

Note: This product includes GeoLite2 data created by MaxMind, available from <http://www.maxmind.com>. This database is licensed under [Creative Commons Attribution-ShareAlike 4.0 International License](#)

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
geoip {
 source => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>add_field</code>	<a href="#">hash</a>	No	{ }
<code>add_tag</code>	<a href="#">array</a>	No	[]
<code>cache_size</code>	<a href="#">number</a>	No	1000
<code>database</code>	a valid filesystem path	No	

Setting	Input type	Required	Default value
enable_metric	<a href="#">boolean</a>	No	true
fields	<a href="#">array</a>	No	[ "city_name", "continent_code", "country_code2", "country_code3", "country_name", "dma_code", "ip", "latitude", "longitude", "postal_code", "region_name", "region_code", "timezone", "location" ]
id	<a href="#">string</a>	No	
lru_cache_size	<a href="#">number</a>	No	1000
periodic_flush	<a href="#">boolean</a>	No	false
remove_field	<a href="#">array</a>	No	[ ]
remove_tag	<a href="#">array</a>	No	[ ]
source	<a href="#">string</a>	Yes	
tag_on_failure	<a href="#">array</a>	No	[ "_geoip_lookup_failure" ]
target	<a href="#">string</a>	No	"geoip"

## Details

### `add_field`

- Value type is [hash](#)
- Default value is `{ }`

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 geoip {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 geoip {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

#### `add_tag`

- Value type is [array](#)
- Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 geoip {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 geoip {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

#### `cache_size`

- Value type is [number](#)
- Default value is 1000

GeoIP lookup is surprisingly expensive. This filter uses an cache to take advantage of the fact that IPs agents are often found adjacent to one another in log files and rarely have a random distribution. The higher you set this the more likely an item is to be in the cache and the faster this filter will run. However, if you set this too high you can use more memory than desired. Since the Geoip API upgraded to v2, there is not any eviction policy so far, if cache is full, no more record can be added. Experiment with different values for this option to find the best performance for your dataset.

This MUST be set to a value > 0. There is really no reason to not want this behavior, the overhead is minimal and the speed gains are large.

It is important to note that this config value is global to the `geoip_type`. That is to say all instances of the `geoip` filter of the same `geoip_type` share the same cache. The last declared cache size will *win*. The reason for this is that there would be no benefit to having multiple caches for

different instances at different points in the pipeline, that would just increase the number of cache misses and waste memory.

### `database`

- Value type is [path](#)
- There is no default value for this setting.

The path to the GeoLite2 database file which Logstash should use. Only City database is supported by now.

If not specified, this will default to the GeoLite2 City database that ships with Logstash.

### `enable_metric`

- Value type is [boolean](#)
- Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `fields`

- Value type is [array](#)
- Default value is `["city_name", "continent_code", "country_code2", "country_code3", "country_name", "dma_code", "ip", "latitude", "longitude", "postal_code", "region_name", "region_code", "timezone", "location"]`

An array of geoip fields to be included in the event.

Possible fields depend on the database type. By default, all geoip fields are included in the event.

For the built-in GeoLite2 City database, the following are available: `city_name`, `continent_code`, `country_code2`, `country_code3`, `country_name`, `dma_code`, `ip`, `latitude`, `longitude`, `postal_code`, `region_name` and `timezone`.

### `id`

- Value type is [string](#)
- There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}

lru_cache_size
```

- Value type is [number](#)
- Default value is 1000

GeoIP lookup is surprisingly expensive. This filter uses an LRU cache to take advantage of the fact that IPs agents are often found adjacent to one another in log files and rarely have a random distribution. The higher you set this the more likely an item is to be in the cache and the faster this filter will run. However, if you set this too high you can use more memory than desired.

Experiment with different values for this option to find the best performance for your dataset.

This MUST be set to a value  $> 0$ . There is really no reason to not want this behavior, the overhead is minimal and the speed gains are large.

It is important to note that this config value is global to the geoip\_type. That is to say all instances of the geoip filter of the same geoip\_type share the same cache. The last declared cache size will *win*. The reason for this is that there would be no benefit to having multiple caches for different instances at different points in the pipeline, that would just increase the number of cache misses and waste memory.

[periodic\\_flush](#)

- Value type is [boolean](#)
- Default value is `false`

Call the filter flush method at regular interval. Optional.

[remove\\_field](#)

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 geoip {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 geoip {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
```

```
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

### `remove_tag`

- Value type is [array](#)
- Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 geoip {
 remove_tag => ["foo_%{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 geoip {
 remove_tag => ["foo_%{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

### `source`

- This is a required setting.
- Value type is [string](#)
- There is no default value for this setting.

The field containing the IP address or hostname to map via geoip. If this field is an array, only the first value will be used.

### `tag_on_failure`

- Value type is [array](#)
- Default value is `[_geoip_lookup_failure]`

Tags the event on failure to look up geo information. This can be used in later analysis.

*target*

- Value type is [string](#)
- Default value is "geoip"

Specify the field into which Logstash should store the geoip data. This can be useful, for example, if you have `src_ip` and `dst_ip` fields and would like the GeoIP information of both IPs.

If you save the data to a target field other than `geoip` and want to use the `geo_point` related functions in Elasticsearch, you need to alter the template provided with the Elasticsearch output and configure the output to use the new template.

Even if you don't use the `geo_point` mapping, the `[target] [location]` field is still valid GeoJSON.

## grok

Version: 3.3.1

Released on: 2016-12-26

[Changelog](#)

Parse arbitrary text and structure it.

Grok is currently the best way in logstash to parse crappy unstructured log data into something structured and queryable.

This tool is perfect for syslog logs, apache and other webserver logs, mysql logs, and in general, any log format that is generally written for humans and not computer consumption.

Logstash ships with about 120 patterns by default. You can find them here:

<https://github.com/logstash-plugins/logstash-patterns-core/tree/master/patterns>. You can add your own trivially. (See the `patterns_dir` setting)

If you need help building patterns to match your logs, you will find the <http://grokdebug.herokuapp.com> and <http://grokconstructor.appspot.com/> applications quite useful!

### Grok Basics

Grok works by combining text patterns into something that matches your logs.

The syntax for a grok pattern is `%{SYNTAX:SEMANTIC}`

The `SYNTAX` is the name of the pattern that will match your text. For example, `3.44` will be matched by the `NUMBER` pattern and `55.3.244.1` will be matched by the `IP` pattern. The syntax is how you match.

The `SEMANTIC` is the identifier you give to the piece of text being matched. For example, `3.44` could be the duration of an event, so you could call it simply `duration`. Further, a string `55.3.244.1` might identify the `client` making a request.

For the above example, your grok filter would look something like this:

```
%{NUMBER:duration} %{IP:client}
```

Optionally you can add a data type conversion to your grok pattern. By default all semantics are saved as strings. If you wish to convert a semantic's data type, for example change a string to an integer then suffix it with the target data type. For example `%{NUMBER:num:int}` which converts

the `num` semantic from a string to an integer. Currently the only supported conversions are `int` and `float`.

**Examples:** With that idea of a syntax and semantic, we can pull out useful fields from a sample log like this fictional http request log:

```
55.3.244.1 GET /index.html 15824 0.043
```

The pattern for this could be:

```
%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes}
%{NUMBER:duration}
```

A more realistic example, let's read these logs from a file:

```
input {
 file {
 path => "/var/log/http.log"
 }
}
filter {
 grok {
 match => { "message" => "%{IP:client} %{WORD:method}
%{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration}" }
 }
}
```

After the grok filter, the event will have a few extra fields in it:

```
client: 55.3.244.1
method: GET
request: /index.html
bytes: 15824
duration: 0.043
```

## Regular Expressions

Grok sits on top of regular expressions, so any regular expressions are valid in grok as well. The regular expression library is Oniguruma, and you can see the full supported regexp syntax [on the Oniguruma site](#).

## Custom Patterns

Sometimes logstash doesn't have a pattern you need. For this, you have a few options.

First, you can use the Oniguruma syntax for named capture which will let you match a piece of text and save it as a field:

```
(?<field_name>the pattern here)
```

For example, postfix logs have a `queue_id` that is an 10 or 11-character hexadecimal value. I can capture that easily like this:

```
(?<queue_id>[0-9A-F]{10,11})
```

Alternately, you can create a custom patterns file.

Create a directory called `patterns` with a file in it called `extra` (the file name doesn't matter, but name it meaningfully for yourself)

In that file, write the pattern you need as the pattern name, a space, then the regexp for that pattern.

For example, doing the postfix queue id example as above:

```
contents of ./patterns/postfix:
POSTFIX_QUEUEID [0-9A-F]{10,11}
```

Then use the `patterns_dir` setting in this plugin to tell logstash where your custom patterns directory is. Here's a full example with a sample log:

```
Jan 1 06:25:43 mailserver14 postfix/cleanup[21403]: BEF25A72965: message-
id=<20130101142543.5828399CCAF@mailserver14.example.com>
filter {
 grok {
 patterns_dir => ["./patterns"]
 match => { "message" => "%{SYSLOGBASE} %{POSTFIX_QUEUEID:queue_id}:
%{GREEDYDATA:syslog_message}" }
 }
}
```

The above will match and result in the following fields:

```
timestamp: Jan 1 06:25:43

logsource: mailserver14

program: postfix/cleanup

pid: 21403

queue_id: BEF25A72965

syslog_message: message-
id=<20130101142543.5828399CCAF@mailserver14.example.com>
```

The `timestamp`, `logsource`, `program`, and `pid` fields come from the `SYSLOGBASE` pattern which itself is defined by other patterns.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
grok {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>add_field</code>	<a href="#">hash</a>	No	{ }
<code>add_tag</code>	<a href="#">array</a>	No	[]
<code>break_on_match</code>	<a href="#">boolean</a>	No	true
<code>enable_metric</code>	<a href="#">boolean</a>	No	true
<code>id</code>	<a href="#">string</a>	No	
<code>keep_empty_captures</code>	<a href="#">boolean</a>	No	false
<code>match</code>	<a href="#">hash</a>	No	{ }
<code>namedcaptures_only</code>	<a href="#">boolean</a>	No	true
<code>overwrite</code>	<a href="#">array</a>	No	[]
<code>patterns_dir</code>	<a href="#">array</a>	No	[]
<code>patterns_files_glob</code>	<a href="#">string</a>	No	"*"
<code>periodic_flush</code>	<a href="#">boolean</a>	No	false
<code>remove_field</code>	<a href="#">array</a>	No	[]
<code>remove_tag</code>	<a href="#">array</a>	No	[]
<code>tag_on_failure</code>	<a href="#">array</a>	No	[ "_grokparsefailure" ]
<code>tag_on_timeout</code>	<a href="#">string</a>	No	"_groktimeout"
<code>timeout_millis</code>	<a href="#">number</a>	No	30000

## Details

### `add_field`

Value type is [hash](#)

Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 grok {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
 # You can also add multiple fields at once:
 filter {
 grok {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

### `add_tag`

Value type is [array](#)

Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 grok {
 add_tag => ["foo_%{somefield}"]
 }
 # You can also add multiple tags at once:
 filter {
 grok {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
 }
}
```

}

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`break_on_match`

Value type is [boolean](#)

Default value is `true`

Break on first match. The first successful match by grok will result in the filter being finished. If you want grok to try all patterns (maybe you are parsing different things), then set this to false.

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
keep_empty_captures
```

Value type is [boolean](#)

Default value is `false`

If `true`, keep empty captures as event fields.

*match*

Value type is [hash](#)

Default value is {}

A hash of matches of field ⇒ value

For example:

```
filter {
 grok { match => { "message" => "Duration: %{NUMBER:duration}" } }
}
```

If you need to match multiple patterns against a single field, the value can be an array of patterns

```
filter {
 grok { match => { "message" => ["Duration: %{NUMBER:duration}", "Speed: %{NUMBER:speed}"] } }
}
named_captures_only
```

Value type is [boolean](#)

Default value is true

If true, only store named captures from grok.

*overwrite*

Value type is [array](#)

Default value is []

The fields to overwrite.

This allows you to overwrite a value in a field that already exists.

For example, if you have a syslog line in the message field, you can overwrite the message field with part of the match like so:

```
filter {
 grok {
 match => { "message" => "%{SYSLOGBASE} %{DATA:message}" }
 overwrite => ["message"]
 }
}
```

In this case, a line like `May 29 16:37:11 sadness logger: hello world` will be parsed and `hello world` will overwrite the original message.

### `patterns_dir`

Value type is [array](#)

Default value is `[]`

Logstash ships by default with a bunch of patterns, so you don't necessarily need to define this yourself unless you are adding additional patterns. You can point to multiple pattern directories using this setting. Note that Grok will read all files in the directory matching the `patterns_files_glob` and assume it's a pattern file (including any tilde backup files).

```
patterns_dir => ["/opt/logstash/patterns", "/opt/logstash/extr_patterns"]
```

Pattern files are plain text with format:

```
NAME PATTERN
```

For example:

```
NUMBER \d+
```

The patterns are loaded when the pipeline is created.

### `patterns_files_glob`

Value type is [string](#)

Default value is `"*"`

Glob pattern, used to select the pattern files in the directories specified by `patterns_dir`

### `periodic_flush`

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

### `remove_field`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 grok {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 grok {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name foo\_hello if it is present. The second example would remove an additional, non-dynamic field.

[remove\\_tag](#)

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the %{field} syntax.

Example:

```
filter {
 grok {
 remove_tag => ["foo_{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 grok {
 remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag foo\_hello if it is present. The second example would remove a sad, unwanted tag as well.

[tag\\_on\\_failure](#)

Value type is [array](#)

Default value is ["\_grokparsefailure"]

Append values to the `tags` field when there has been no successful match

`tag_on_timeout`

Value type is [string](#)

Default value is "`_groktimeout`"

Tag to apply if a grok regexp times out.

`timeout_millis`

Value type is [number](#)

Default value is 30000

Attempt to terminate regexps after this amount of time. This applies per pattern if multiple patterns are applied. This will never timeout early, but may take a little longer to timeout. Actual timeout is approximate based on a 250ms quantization. Set to 0 to disable timeouts

# i18n

This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-filter-i18n`.

The i18n filter allows you to remove special characters from a field

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
i18n {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>add_field</code>	<a href="#">hash</a>	No	<code>{ }</code>
<code>add_tag</code>	<a href="#">array</a>	No	<code>[]</code>
<code>periodic_flush</code>	<a href="#">boolean</a>	No	<code>false</code>
<code>remove_field</code>	<a href="#">array</a>	No	<code>[]</code>
<code>remove_tag</code>	<a href="#">array</a>	No	<code>[]</code>
<code>transliterate</code>	<a href="#">array</a>	No	

## Details

### `add_field`

Value type is [hash](#)

Default value is `{ }`

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 i18n {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 i18n {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

`add_tag`

Value type is [array](#)

Default value is [ ]

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 i18n {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 i18n {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`periodic_flush`

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

`remove_field`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 i18n {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 i18n {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 i18n {
 remove_tag => ["foo_{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 i18n {
 remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag foo\_hello if it is present. The second example would remove a sad, unwanted tag as well.

*transliterate*

Value type is [array](#)

There is no default value for this setting.

Replaces non-ASCII characters with an ASCII approximation, or if none exists, a replacement character which defaults to ?

Example:

```
filter {
 i18n {
 transliterate => ["field1", "field2"]
 }
}
```

# json

Version: 3.0.2

Released on: 2016-07-14

[Changelog](#)

This is a JSON parsing filter. It takes an existing field which contains JSON and expands it into an actual data structure within the Logstash event.

By default it will place the parsed JSON in the root (top level) of the Logstash event, but this filter can be configured to place the JSON into any arbitrary event field, using the `target` configuration.

This plugin has a few fallback scenario when something bad happen during the parsing of the event. If the JSON parsing fails on the data, the event will be untouched and it will be tagged with a `_jsonparsefailure` then you can use conditionals to clean the data. You can configured this tag with then `tag_on_failure` option.

If the parsed data contains a `@timestamp` field, we will try to use it for the event's `@timestamp`, if the parsing fails, the field will be renamed to `_@timestamp` and the event will be tagged with a `_timestampparsefailure`.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
json {
 source => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
add_tag	<a href="#">array</a>	No	[]
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	

Setting	Input type	Required	Default value
periodic_flush	<a href="#">boolean</a>	No	false
remove_field	<a href="#">array</a>	No	[]
remove_tag	<a href="#">array</a>	No	[]
skip_on_invalid_json	<a href="#">boolean</a>	No	false
source	<a href="#">string</a>	Yes	
tag_on_failure	<a href="#">array</a>	No	[ "_jsonparsefailure" ]
target	<a href="#">string</a>	No	

## Details

[`add\_field`](#)

Value type is [hash](#)

Default value is { }

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 json {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 json {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

### `add_tag`

Value type is [array](#)

Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 json {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 json {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

### `enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

### `id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
```

```
}
```

*periodic\_flush*

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

*remove\_field*

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 json {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 json {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

*remove\_tag*

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 json {
 remove_tag => ["foo_{somefield}"]
 }
}
```

## Logstash 5.2 Configuration Guide

```
You can also remove multiple tags at once:
filter {
 json {
 remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag foo\_hello if it is present. The second example would remove a sad, unwanted tag as well.

*skip\_on\_invalid\_json*

Value type is [boolean](#)

Default value is `false`

Allow to skip filter on invalid json (allows to handle json and non-json data without warnings)

*source*

This is a required setting.

Value type is [string](#)

There is no default value for this setting.

The configuration for the JSON filter:

```
source => source_field
```

For example, if you have JSON data in the `message` field:

```
filter {
 json {
 source => "message"
 }
}
```

The above would parse the json from the `message` field

*tag\_on\_failure*

Value type is [array](#)

Default value is `[ "_jsonparsefailure" ]`

Append values to the `tags` field when there has been no successful match

*target*

Value type is [string](#)

There is no default value for this setting.

Define the target field for placing the parsed data. If this setting is omitted, the JSON data will be stored at the root (top level) of the event.

For example, if you want the data to be put in the `doc` field:

```
filter {
 json {
 target => "doc"
 }
}
```

JSON in the value of the `source` field will be expanded into a data structure in the `target` field.

NOTE: if the `target` field already exists, it will be overwritten!

# json\_encode

Version: 3.0.0

Released on: 2016-11-13

[Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-filter-json_encode`.

JSON encode filter. Takes a field and serializes it into JSON

If no target is specified, the source field is overwritten with the JSON text.

For example, if you have a field named `foo`, and you want to store the JSON encoded string in `bar`, do this:

```
filter {
 json_encode {
 source => "foo"
 target => "bar"
 }
}
```

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
json_encode {
 source => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
add_tag	<a href="#">array</a>	No	[]
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	

Setting	Input type	Required	Default value
periodic_flush	<a href="#">boolean</a>	No	false
remove_field	<a href="#">array</a>	No	[]
remove_tag	<a href="#">array</a>	No	[]
source	<a href="#">string</a>	Yes	
target	<a href="#">string</a>	No	

## Details

[\*add\\_field\*](#)

Value type is [hash](#)

Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the %{field}.

Example:

```
filter {
 json_encode {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
 # You can also add multiple fields at once:
 filter {
 json_encode {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

[\*add\\_tag\*](#)

Value type is [array](#)

Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 json_encode {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 json_encode {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

`periodic_flush`

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

`remove_field`

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
  json_encode {
    remove_field => [ "foo_{somefield}" ]
  }
}
# You can also remove multiple fields at once:
filter {
  json_encode {
    remove_field => [ "foo_{somefield}", "my_extraneous_field" ]
  }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  json_encode {
    remove_tag => [ "foo_{somefield}" ]
  }
}
# You can also remove multiple tags at once:
filter {
  json_encode {
    remove_tag => [ "foo_{somefield}", "sad_unwanted_tag" ]
  }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

source

This is a required setting.

Value type is [string](#)

There is no default value for this setting.

The field to convert to JSON.

target

Value type is [string](#)

There is no default value for this setting.

The field to write the JSON into. If not specified, the source field will be overwritten.

kv

Version: 3.1.1

Released on: 2016-07-14

[Changelog](#)

This filter helps automatically parse messages (or specific event fields) which are of the `foo=bar` variety.

For example, if you have a log message which contains `ip=1.2.3.4 error=REFUSED`, you can parse those automatically by configuring:

```
filter {
  kv { }
}
```

The above will result in a message of `ip=1.2.3.4 error=REFUSED` having the fields:

```
ip: 1.2.3.4
error: REFUSED
```

This is great for postfix, iptables, and other types of logs that tend towards `key=value` syntax.

You can configure any arbitrary strings to split your data on, in case your data is not structured using `=` signs and whitespace. For example, this filter can also be used to parse query parameters like `foo=bar&baz=fizz` by setting the `field_split` parameter to `&`.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
kv { }
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
add_tag	array	No	[]
allow_duplicate_values	boolean	No	true
default_keys	hash	No	{ }
enable_metric	boolean	No	true
exclude_keys	array	No	[]
field_split	string	No	" "
id	string	No	
include_brackets	boolean	No	true
include_keys	array	No	[]
periodic_flush	boolean	No	false
prefix	string	No	""
recursive	boolean	No	false
remove_field	array	No	[]
remove_tag	array	No	[]
source	string	No	"message"
target	string	No	
transform_key	string , one of ["lowercase", "uppercase", "capitalize"]	No	
transform_value	string , one of ["lowercase", "uppercase", "capitalize"]	No	
trim	string	No	
trimkey	string	No	
value_split	string	No	"="

Details

[add_field](#)

Value type is [hash](#)

Default value is { }

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
  kv {
    add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
  }
}
# You can also add multiple fields at once:
filter {
  kv {
    add_field => {
      "foo_%{somefield}" => "Hello world, from %{host}"
      "new_field" => "new_static_value"
    }
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

`add_tag`

Value type is [array](#)

Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  kv {
    add_tag => [ "foo_%{somefield}" ]
  }
}
# You can also add multiple tags at once:
filter {
  kv {
    add_tag => [ "foo_%{somefield}", "taggedy_tag" ]
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`allow_duplicate_values`

Value type is [boolean](#)

Default value is `true`

A bool option for removing duplicate key/value pairs. When set to false, only one unique key/value pair will be preserved.

For example, consider a source like `from=me from=me`. `[from]` will map to an Array with two elements: `["me", "me"]`. To only keep unique key/value pairs, you could use this configuration:

```
filter {
  kv {
    allow_duplicate_values => false
  }
}
default_keys
```

Value type is [hash](#)

Default value is `{ }`

A hash specifying the default keys and their values which should be added to the event in case these keys do not exist in the source field being parsed.

```
filter {
  kv {
    default_keys => [ "from", "logstash@example.com",
                        "to", "default@dev.null" ]
  }
}
enable_metric
```

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`exclude_keys`

Value type is [array](#)

Default value is `[]`

An array specifying the parsed keys which should not be added to the event. By default no keys will be excluded.

For example, consider a source like `Hey, from=<abc>, to=def foo=bar`. To exclude `from` and `to`, but retain the `foo` key, you could use this configuration:

```
filter {
  kv {
    exclude_keys => [ "from", "to" ]
  }
}
field_split
```

Value type is [string](#)

Default value is " "

A string of characters to use as delimiters for parsing out key-value pairs.

These characters form a regex character class and thus you must escape special regex characters like `[` or `]` using `\`.

Example with URL Query Strings

For example, to split out the args from a url query string such as

`?pin=12345~0&d=123&e=foo@bar.com&oq=bobo&ss=12345`:

```
filter {
  kv {
    field_split => "&?"
  }
}
```

The above splits on both `&` and `?` characters, giving you the following fields:

`pin: 12345~0`

`d: 123`

`e: foo@bar.com`

`oq: bobo`

`ss: 12345`

id

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
  stdout {  
    id => "my_plugin_id"  
  }  
}  
include brackets
```

Value type is [boolean](#)

Default value is `true`

A boolean specifying whether to treat square brackets, angle brackets, and parentheses as value "wrappers" that should be removed from the value.

```
filter {  
  kv {  
    include_brackets => true  
  }  
}
```

For example, the result of this line: `bracketsone=(hello world) bracketstwo=[hello world] bracketsthree=<hello world>`

will be:

`bracketsone: hello world`

`bracketstwo: hello world`

`bracketsthree: hello world`

instead of:

`bracketsone: (hello`

`bracketstwo: [hello`

`bracketsthree: <hello`

include_keys

Value type is [array](#)

Default value is `[]`

An array specifying the parsed keys which should be added to the event. By default all keys will be added.

For example, consider a source like `Hey, from=<abc>, to=def foo=bar`. To include `from` and `to`, but exclude the `foo` key, you could use this configuration:

```
filter {
  kv {
    include_keys => [ "from", "to" ]
  }
}  
periodic_flush
```

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

prefix

Value type is [string](#)

Default value is `""`

A string to prepend to all of the extracted keys.

For example, to prepend `arg_` to all keys:

```
filter { kv { prefix => "arg_" } }
recursive
```

Value type is [boolean](#)

Default value is `false`

A boolean specifying whether to drill down into values and recursively get more key-value pairs from it. The extra key-value pairs will be stored as subkeys of the root key.

Default is not to recursive values.

```
filter {
  kv {
    recursive => "true"
  }
}
```

`remove_field`

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
  kv {
    remove_field => [ "foo_{somefield}" ]
  }
}
# You can also remove multiple fields at once:
filter {
  kv {
    remove_field => [ "foo_{somefield}", "my_extraneous_field" ]
  }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  kv {
    remove_tag => [ "foo_{somefield}" ]
  }
}
# You can also remove multiple tags at once:
filter {
  kv {
    remove_tag => [ "foo_{somefield}", "sad_unwanted_tag" ]
  }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

source

Value type is [string](#)

Default value is "message"

The field to perform key=value searching on

For example, to process the not_the_message field:

```
filter { kv { source => "not_the_message" } }
```

target

Value type is [string](#)

There is no default value for this setting.

The name of the container to put all of the key-value pairs into.

If this setting is omitted, fields will be written to the root of the event, as individual fields.

For example, to place all keys into the event field kv:

```
filter { kv { target => "kv" } }
```

Value can be any of: lowercase, uppercase, capitalize

There is no default value for this setting.

Transform keys to lower case, upper case or capitals.

For example, to lowercase all keys:

```
filter {
  kv {
    transform_key => "lowercase"
  }
}
```

transform_value

Value can be any of: lowercase, uppercase, capitalize

There is no default value for this setting.

Transform values to lower case, upper case or capitals.

For example, to capitalize all values:

```
filter {
  kv {
    transform_value => "capitalize"
  }
}
trim
```

Value type is [string](#)

There is no default value for this setting.

Constants used for transform check A string of characters to trim from the value. This is useful if your values are wrapped in brackets or are terminated with commas (like postfix logs).

These characters form a regex character class and thus you must escape special regex characters like [or] using \.

For example, to strip <, >, [,] and , characters from values:

```
filter {
  kv {
    trim => "<>\\[\\],"
  }
}
trimkey
```

Value type is [string](#)

There is no default value for this setting.

A string of characters to trim from the key. This is useful if your keys are wrapped in brackets or start with space.

These characters form a regex character class and thus you must escape special regex characters like [or] using \.

For example, to strip < > [] and , characters from keys:

```
filter {
  kv {
    trimkey => "<>\\[\\],"
  }
}
value_split
```

Value type is [string](#)

Default value is "="

A non-empty string of characters to use as delimiters for identifying key-value relations.

These characters form a regex character class and thus you must escape special regex characters like [or] using \.

For example, to identify key-values such as key1:value1 key2:value2:

```
filter { kv { value_split => ":" } }
```

metaevent

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-filter-metaevent`.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
metaevent {
    followed_by_tags => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
add_tag	array	No	[]
followed_by_tags	array	Yes	
period	number	No	5
periodic_flush	boolean	No	false
remove_field	array	No	[]
remove_tag	array	No	[]

Details

[`add_field`](#)

Value type is [hash](#)

Default value is { }

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
    metaevent {
        add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
    }
}
# You can also add multiple fields at once:
filter {
    metaevent {
        add_field => {
            "foo_%{somefield}" => "Hello world, from %{host}"
            "new_field" => "new_static_value"
        }
    }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

[`add_tag`](#)

Value type is [array](#)

Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
    metaevent {
        add_tag => [ "foo_%{somefield}" ]
    }
}
# You can also add multiple tags at once:
filter {
    metaevent {
        add_tag => [ "foo_%{somefield}", "taggedy_tag" ]
    }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

[`followed_by_tags`](#)

This is a required setting.

Value type is [array](#)

There is no default value for this setting.

syntax: followed_by_tags => ["tag", "tag"]

period

Value type is [number](#)

Default value is 5

syntax: period => 60

periodic_flush

Value type is [boolean](#)

Default value is false

Call the filter flush method at regular interval. Optional.

remove_field

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
  metaevent {
    remove_field => [ "foo_{somefield}" ]
  }
}
# You can also remove multiple fields at once:
filter {
  metaevent {
    remove_field => [ "foo_{somefield}", "my_extraneous_field" ]
  }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name foo_hello if it is present. The second example would remove an additional, non-dynamic field.

remove_tag

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
    metaevent {
        remove_tag => [ "foo_%{somefield}" ]
    }
}
# You can also remove multiple tags at once:
filter {
    metaevent {
        remove_tag => [ "foo_%{somefield}", "sad_unwanted_tag" ]
    }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

metricize

Version: 3.0.0

Released on: 2016-09-09

[Changelog](#)

This plugin does not ship with Logstash by default, but it is easy to install by running
bin/logstash-plugin install logstash-filter-metricize.

The metricize filter takes complex events containing a number of metrics and splits these up into multiple events, each holding a single metric.

Example:

Assume the following filter configuration:

```
filter {
    metricize {
        metrics => [ "metric1", "metric2" ]
    }
}
```

Assuming the following event is passed in:

```
{
    type => "type A"
    metric1 => "value1"
    metric2 => "value2"
}
This will result in the following 2 events being generated in addition to the original event:
{
    type => "type A"
    metric => "metric1"
    value => "value1"                                {
                                                    type => "type A"
                                                    metric => "metric2"
                                                    value => "value2"
}
}
```

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
metricize {
    metrics => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
add_tag	array	No	[]
drop_original_event	boolean	No	false
enable_metric	boolean	No	true
id	string	No	
metric_field_name	string	No	"metric"
metrics	array	Yes	
periodic_flush	boolean	No	false
remove_field	array	No	[]
remove_tag	array	No	[]
value_field_name	string	No	"value"

Details

[add_field](#)

Value type is [hash](#)

Default value is { }

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
    metricize {
        add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
    }
}
# You can also add multiple fields at once:
filter {
    metricize {
        add_field => {
            "foo_%{somefield}" => "Hello world, from %{host}"
            "new_field" => "new_static_value"
        }
    }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

`add_tag`

Value type is [array](#)

Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
    metricize {
        add_tag => [ "foo_%{somefield}" ]
    }
}
# You can also add multiple tags at once:
filter {
    metricize {
        add_tag => [ "foo_%{somefield}", "taggedy_tag" ]
    }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`drop_original_event`

Value type is [boolean](#)

Default value is `false`

Flag indicating whether the original event should be dropped or not.

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

id

Value type is [string](#)

There is no default value for this setting.

Add a unique ID to the plugin instance, this ID is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

*metric\_field\_name*

Value type is [string](#)

Default value is "metric"

Name of the field the metric name will be written to.

*metrics*

This is a required setting.

Value type is [array](#)

There is no default value for this setting.

A new metrics event will be created for each metric field in this list. All fields in this list will be removed from generated events.

*periodic\_flush*

Value type is [boolean](#)

Default value is false

Call the filter flush method at regular interval. Optional.

*remove\_field*

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary fields from this event. Example:

## Logstash 5.2 Configuration Guide

```
filter {
 metricize {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 metricize {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name foo\_hello if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the %{field} syntax.

Example:

```
filter {
 metricize {
 remove_tag => ["foo_{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 metricize {
 remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag foo\_hello if it is present. The second example would remove a sad, unwanted tag as well.

`value_field_name`

Value type is [string](#)

Default value is "value"

Name of the field the metric value will be written to.



# metrics

Version: 4.0.2

Released on: 2016-07-14

[Changelog](#)

The metrics filter is useful for aggregating metrics.

**IMPORTANT!!!** Elasticsearch 2.0 no longer allows field names with dots. Version 3.0 of the metrics filter plugin changes behavior to use nested fields rather than dotted notation to avoid colliding with versions of Elasticsearch 2.0+. Please note the changes in the documentation (underscores and sub-fields used).

For example, if you have a field `response` that is a http response code, and you want to count each kind of response, you can do this:

```
filter {
 metrics {
 meter => ["http_%{response}"]
 add_tag => "metric"
 }
}
```

Metrics are flushed every 5 seconds by default or according to `flush_interval`. Metrics appear as new events in the event stream and go through any filters that occur after as well as outputs.

In general, you will want to add a tag to your metrics and have an output explicitly look for that tag.

The event that is flushed will include every `meter` and `timer` metric in the following way:

## **meter values**

For a `meter => "something"` you will receive the following fields:

"[thing][count]" - the total count of events

"[thing][rate\_1m]" - the per-second event rate in a 1-minute sliding window

"[thing][rate\_5m]" - the per-second event rate in a 5-minute sliding window

"[thing][rate\_15m]" - the per-second event rate in a 15-minute sliding window

## **timer values**

For a `timer => [ "thing", "%{duration}" ]` you will receive the following fields:

- "`[thing][count]`" - the total count of events
- "`[thing][rate_1m]`" - the per-second event rate in a 1-minute sliding window
- "`[thing][rate_5m]`" - the per-second event rate in a 5-minute sliding window
- "`[thing][rate_15m]`" - the per-second event rate in a 15-minute sliding window
- "`[thing][min]`" - the minimum value seen for this metric
- "`[thing][max]`" - the maximum value seen for this metric
- "`[thing][stddev]`" - the standard deviation for this metric
- "`[thing][mean]`" - the mean for this metric
- "`[thing][pXX]`" - the XXth percentile for this metric (see percentiles)

The default lengths of the event rate window (1, 5, and 15 minutes) can be configured with the `rates` option.

## Example: Computing event rate

For a simple example, let's track how many events per second are running through logstash:

```
input {
 generator {
 type => "generated"
 }
}

filter {
 if [type] == "generated" {
 metrics {
 meter => "events"
 add_tag => "metric"
 }
 }
}

output {
 # only emit events with the 'metric' tag
 if "metric" in [tags] {
 stdout {
 codec => line {
 format => "rate: %{[events][rate_1m]}"
 }
 }
 }
}
```

}

Running the above:

```
% bin/logstash -f example.conf
rate: 23721.983566819246
rate: 24811.395722536377
rate: 25875.892745934525
rate: 26836.42375967113
```

We see the output includes our events' 1-minute rate.

In the real world, you would emit this to graphite or another metrics store, like so:

```
output {
 graphite {
 metrics => ["events.rate_1m", "%{[events][rate_1m]}"]
 }
}
```

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
metrics { }
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
add_tag	<a href="#">array</a>	No	[]
clear_interval	<a href="#">number</a>	No	-1
enable_metric	<a href="#">boolean</a>	No	true
flush_interval	<a href="#">number</a>	No	5
id	<a href="#">string</a>	No	
ignore_older_than	<a href="#">number</a>	No	0
meter	<a href="#">array</a>	No	[]
percentiles	<a href="#">array</a>	No	[1, 5, 10, 90, 95, 99, 100]
periodic_flush	<a href="#">boolean</a>	No	false

Setting	Input type	Required	Default value
rates	<a href="#">array</a>	No	[1, 5, 15]
remove_field	<a href="#">array</a>	No	[]
remove_tag	<a href="#">array</a>	No	[]
timer	<a href="#">hash</a>	No	{ }

## Details

[`add\_field`](#)

Value type is [hash](#)

Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 metrics {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
 # You can also add multiple fields at once:
 filter {
 metrics {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

[`add\_tag`](#)

Value type is [array](#)

Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 metrics {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 metrics {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`clear_interval`

Value type is [number](#)

Default value is `-1`

The clear interval, when all counter are reset.

If set to `-1`, the default value, the metrics will never be cleared. Otherwise, should be a multiple of 5s.

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`flush_interval`

Value type is [number](#)

Default value is `5`

The flush interval, when the metrics event is created. Must be a multiple of 5s.

### *id*

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
ignore_older_than
```

Value type is [number](#)

Default value is 0

Don't track events that have `@timestamp` older than some number of seconds.

This is useful if you want to only include events that are near real-time in your metrics.

For example, to only count events that are within 10 seconds of real-time, you would do this:

```
filter {
 metrics {
 meter => ["hits"]
 ignore_older_than => 10
 }
}
meter
```

Value type is [array](#)

Default value is []

**syntax:** `meter => [ "name of metric", "name of metric" ]`

### *percentiles*

Value type is [array](#)

Default value is [1, 5, 10, 90, 95, 99, 100]

The percentiles that should be measured and emitted for timer values.

`periodic_flush`

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

`rates`

Value type is [array](#)

Default value is `[1, 5, 15]`

The rates that should be measured, in minutes. Possible values are 1, 5, and 15.

`remove_field`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 metrics {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 metrics {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 metrics {
 remove_tag => ["foo_%{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 metrics {
 remove_tag => ["foo_%{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

`timer`

Value type is [hash](#)

Default value is `{ }`

**syntax:** `timer => [ "name of metric", "%{time_value}" ]`

## mutate

Version: 3.1.3

Released on: 2016-09-29

[Changelog](#)

The mutate filter allows you to perform general mutations on fields. You can rename, remove, replace, and modify fields in your events.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
mutate {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">add_tag</a>	<a href="#">array</a>	No	[]
<a href="#">convert</a>	<a href="#">hash</a>	No	
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">gsub</a>	<a href="#">array</a>	No	
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">join</a>	<a href="#">hash</a>	No	
<a href="#">lowercase</a>	<a href="#">array</a>	No	
<a href="#">merge</a>	<a href="#">hash</a>	No	
<a href="#">periodic_flush</a>	<a href="#">boolean</a>	No	false
<a href="#">remove_field</a>	<a href="#">array</a>	No	[]
<a href="#">remove_tag</a>	<a href="#">array</a>	No	[]
<a href="#">rename</a>	<a href="#">hash</a>	No	
<a href="#">replace</a>	<a href="#">hash</a>	No	
<a href="#">split</a>	<a href="#">hash</a>	No	

Setting	Input type	Required	Default value
<a href="#">strip</a>	<a href="#">array</a>	No	
<a href="#">update</a>	<a href="#">hash</a>	No	
<a href="#">uppercase</a>	<a href="#">array</a>	No	

## Details

[`add\_field`](#)

Value type is [hash](#)

Default value is `{ }`

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 mutate {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 mutate {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

[`add\_tag`](#)

Value type is [array](#)

Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 mutate {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 mutate {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

#### `convert`

Value type is [hash](#)

There is no default value for this setting.

Convert a field's value to a different type, like turning a string to an integer. If the field value is an array, all members will be converted. If the field is a hash, no action will be taken.

If the conversion type is `boolean`, the acceptable values are:

**True:** `true`, `t`, `yes`, `y`, and `1`

**False:** `false`, `f`, `no`, `n`, and `0`

If a value other than these is provided, it will pass straight through and log a warning message.

Valid conversion targets are: `integer`, `float`, `string`, and `boolean`.

Example:

```
filter {
 mutate {
 convert => { "fieldname" => "integer" }
 }
}
enable_metric
```

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

*gsub*

Value type is [array](#)

There is no default value for this setting.

Convert a string field by applying a regular expression and a replacement. If the field is not a string, no action will be taken.

This configuration takes an array consisting of 3 elements per field/substitution.

Be aware of escaping any backslash in the config file.

Example:

```
filter {
 mutate {
 gsub => [
 # replace all forward slashes with underscore
 "fieldname", "/", "_",
 # replace backslashes, question marks, hashes, and minuses
 # with a dot "."
 "fieldname2", "[\\?#-]", "."
]
 }
}
```

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

Value type is [hash](#)

There is no default value for this setting.

Join an array with a separator character. Does nothing on non-array fields.

Example:

```
filter {
 mutate {
 join => { "fieldname" => ",", }
 }
}
lowercase
```

Value type is [array](#)

There is no default value for this setting.

Convert a string to its lowercase equivalent.

Example:

```
filter {
 mutate {
 lowercase => ["fieldname"]
 }
}
merge
```

Value type is [hash](#)

There is no default value for this setting.

Merge two fields of arrays or hashes. String fields will be automatically be converted into an array, so:

```
`array` + `string` will work
`string` + `string` will result in an 2 entry array in `dest_field`
`array` and `hash` will not work
```

Example:

```
filter {
 mutate {
 merge => { "dest_field" => "added_field" }
 }
}
periodic_flush
```

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

`remove_field`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 mutate {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 mutate {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 mutate {
 remove_tag => ["foo_{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 mutate {
 remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag foo\_hello if it is present. The second example would remove a sad, unwanted tag as well.

### `rename`

Value type is [hash](#)

There is no default value for this setting.

Rename one or more fields.

Example:

```
filter {
 mutate {
 # Renames the 'HOSTORIP' field to 'client_ip'
 rename => { "HOSTORIP" => "client_ip" }
 }
}
replace
```

Value type is [hash](#)

There is no default value for this setting.

Replace a field with a new value. The new value can include %{foo} strings to help you build a new value from other parts of the event.

Example:

```
filter {
 mutate {
 replace => { "message" => "%{source_host}: My new message" }
 }
}
split
```

Value type is [hash](#)

There is no default value for this setting.

Split a field to an array using a separator character. Only works on string fields.

Example:

```
filter {
 mutate {
 split => { "fieldname" => "," }
 }
}
```

```
}
```

*strip*

Value type is [array](#)

There is no default value for this setting.

Strip whitespace from field. NOTE: this only works on leading and trailing whitespace.

Example:

```
filter {
 mutate {
 strip => ["field1", "field2"]
 }
}
```

*update*

Value type is [hash](#)

There is no default value for this setting.

Update an existing field with a new value. If the field does not exist, then no action will be taken.

Example:

```
filter {
 mutate {
 update => { "sample" => "My new message" }
 }
}
```

*uppercase*

Value type is [array](#)

There is no default value for this setting.

Convert a string to its uppercase equivalent.

Example:

```
filter {
 mutate {
 uppercase => ["fieldname"]
 }
}
```

# oui

Version: 3.0.0

Released on: 2016-11-28

[Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-filter-oui`.

Logstash filter to parse OUI data from MAC addresses

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
oui {
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
add_tag	<a href="#">array</a>	No	[]
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	
periodic_flush	<a href="#">boolean</a>	No	false
remove_field	<a href="#">array</a>	No	[]
remove_tag	<a href="#">array</a>	No	[]
source	<a href="#">string</a>	No	"message"
target	<a href="#">string</a>	No	"oui"

## Details

### `add_field`

Value type is [hash](#)

Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 oui {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 oui {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

### `add_tag`

Value type is [array](#)

Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 oui {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 oui {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

}

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

``` output { stdout { id => "ABC" } } ```

If you don't explicitly set this variable Logstash will generate a unique name.

`periodic_flush`

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

`remove_field`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
  oui {
    remove_field => [ "foo_{somefield}" ]
  }
}
```

Logstash 5.2 Configuration Guide

```
# You can also remove multiple fields at once:  
filter {  
  oui {  
    remove_field => [ "foo_{somefield}", "my_extraneous_field" ]  
  }  
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name foo_hello if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the %{field} syntax.

Example:

```
filter {  
  oui {  
    remove_tag => [ "foo_{somefield}" ]  
  }  
}  
# You can also remove multiple tags at once:  
filter {  
  oui {  
    remove_tag => [ "foo_{somefield}", "sad_unwanted_tag" ]  
  }  
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag foo_hello if it is present. The second example would remove a sad, unwanted tag as well.

`source`

Value type is [string](#)

Default value is "message"

Setting the config_name here is required. This is how you configure this filter from your Logstash config.

```
filter { example { message => "My message..." } }
```

The source field to parse

target

Value type is [string](#)

Default value is "oui"

The target field to place all the data

prune

Version: 3.0.0

Released on: 2017-01-04

[Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-filter-prune`.

The prune filter is for removing fields from events based on whitelists or blacklist of field names or their values (names and values can also be regular expressions).

This can e.g. be useful if you have a [json](#) or [kv](#) filter that creates a number of fields with names that you don't necessarily know the names of beforehand, and you only want to keep a subset of them.

Usage help: To specify a exact field name or value use the regular expression syntax

`^some_name_or_value$`. Example usage: Input data { "msg": "hello world", "msg_short": "hw" }

```
filter {
  prune {
    whitelist_names => [ "msg" ]
  }
}
```

Allows both `msg` and `msg_short` through.

While:

```
filter {
  prune {
    whitelist_names => [ "^msg$" ]
  }
}
```

Allows only `msg` through.

Logstash stores an event's `tags` as a field which is subject to pruning. Remember to `whitelist_names => ["^tags$"]` to maintain `tags` after pruning or use `blacklist_values => ["^tag_name$"]` to eliminate a specific tag.

This filter currently only support operations on top-level fields, i.e. whitelisting and blacklisting of subfields based on name or value does not work.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
prune {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
add_tag	array	No	[]
blacklist_names	array	No	["%{[^}]+}"]
blacklist_values	hash	No	{ }
enable_metric	boolean	No	true
id	string	No	
interpolate	boolean	No	false
periodic_flush	boolean	No	false
remove_field	array	No	[]
remove_tag	array	No	[]
whitelist_names	array	No	[]
whitelist_values	hash	No	{ }

Details

[*add_field*](#)

Value type is [hash](#)

Default value is { }

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

Logstash 5.2 Configuration Guide

```
filter {
  prune {
    add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
  }
}
# You can also add multiple fields at once:
filter {
  prune {
    add_field => {
      "foo_%{somefield}" => "Hello world, from %{host}"
      "new_field" => "new_static_value"
    }
  }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

[`add_tag`](#)

Value type is [array](#)

Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  prune {
    add_tag => [ "foo_%{somefield}" ]
  }
}
# You can also add multiple tags at once:
filter {
  prune {
    add_tag => [ "foo_%{somefield}", "taggedy_tag" ]
  }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

[`blacklist_names`](#)

Value type is [array](#)

Default value is `["%{[^}]+}"]`

Exclude fields whose names match specified regexps, by default exclude unresolved `%{field}` strings.

```
filter {
  prune {
    blacklist_names => [ "method", "(referrer|status)", "${some}_field" ]
  }
}
blacklist_values
```

Value type is [hash](#)

Default value is {}

Exclude specified fields if their values match one of the supplied regular expressions. In case field values are arrays, each array item is matched against the regular expressions and matching array items will be excluded.

```
filter {
  prune {
    blacklist_values => [ "uripath", "/index.php",
                           "method", "(HEAD|OPTIONS)",
                           "status", "^[^2]" ]
  }
}
enable_metric
```

Value type is [boolean](#)

Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

id

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

*interpolate*

Value type is [boolean](#)

Default value is `false`

Trigger whether configuration fields and values should be interpolated for dynamic values (when resolving `%{some_field}`). Probably adds some performance overhead. Defaults to false.

*periodic\_flush*

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

*remove\_field*

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 prune {
 remove_field => ["foo_%{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 prune {
 remove_field => ["foo_%{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

*remove\_tag*

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 prune {
 remove_tag => ["foo_%{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 prune {
 remove_tag => ["foo_%{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag foo\_hello if it is present. The second example would remove a sad, unwanted tag as well.

#### *whitelist\_names*

Value type is [array](#)

Default value is []

Include only fields only if their names match specified regexps, default to empty list which means include everything.

```
filter {
 prune {
 whitelist_names => ["method", "(referrer|status)", "${some}_field"]
 }
}
```

#### *whitelist\_values*

Value type is [hash](#)

Default value is {}

Include specified fields only if their values match one of the supplied regular expressions. In case field values are arrays, each array item is matched against the regular expressions and only matching array items will be included.

```
filter {
 prune {
 whitelist_values => ["uripath", "/index.php",
 "method", "(GET|POST)",
 "status", "^[^2]"]
 }
}
```

## punct

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-filter-punct`.

Strip everything but punctuation from a field and store the remainder in the a separate field. This is often used for fingerprinting log events.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
punct {
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
add_tag	<a href="#">array</a>	No	[]
periodic_flush	<a href="#">boolean</a>	No	false
remove_field	<a href="#">array</a>	No	[]
remove_tag	<a href="#">array</a>	No	[]
source	<a href="#">string</a>	No	"message"
target	<a href="#">string</a>	No	"punct"

## Details

`add_field`

Value type is [hash](#)

Default value is { }

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 punct {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 punct {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

[`add\_tag`](#)

Value type is [array](#)

Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 punct {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 punct {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

[`periodic\_flush`](#)

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

`remove_field`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 punct {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 punct {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 punct {
 remove_tag => ["foo_{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 punct {
 remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

*source*

Value type is [string](#)

Default value is "message"

The field reference to use for punctuation stripping

*target*

Value type is [string](#)

Default value is "punct"

The field to store the result.

## range

Version: 3.0.0

Released on: 2016-09-08

[Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-filter-range`.

This filter is used to check that certain fields are within expected size/length ranges. Supported types are numbers and strings. Numbers are checked to be within numeric value range. Strings are checked to be within string length range. More than one range can be specified for same fieldname, actions will be applied incrementally. When field value is within a specified range an action will be taken. Supported actions are drop event, add tag, or add field with specified value.

Example use cases are for histogram-like tagging of events or for finding anomaly values in fields or too big events that should be dropped.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
range {
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
add_tag	<a href="#">array</a>	No	[]
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	
negate	<a href="#">boolean</a>	No	false
periodic_flush	<a href="#">boolean</a>	No	false
ranges	<a href="#">array</a>	No	[]
remove_field	<a href="#">array</a>	No	[]

Setting	Input type	Required	Default value
remove_tag	<a href="#">array</a>	No	[ ]

## Details

[\*add\\_field\*](#)

Value type is [hash](#)

Default value is { }

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 range {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 range {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

[\*add\\_tag\*](#)

Value type is [array](#)

Default value is [ ]

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 range {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 range {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

*enable\_metric*

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

*id*

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

negate

Value type is [boolean](#)

Default value is `false`

Negate the range match logic, events should be outsize of the specified range to match.

periodic_flush

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

`ranges`

Value type is [array](#)

Default value is `[]`

An array of field, min, max, action tuples. Example:

```
filter {
    range {
        ranges => [ "message", 0, 10, "tag:short",
                    "message", 11, 100, "tag:medium",
                    "message", 101, 1000, "tag:long",
                    "message", 1001, 1e1000, "drop",
                    "duration", 0, 100, "field:latency:fast",
                    "duration", 101, 200, "field:latency:normal",
                    "duration", 201, 1000, "field:latency:slow",
                    "duration", 1001, 1e1000, "field:latency:outlier",
                    "requests", 0, 10, "tag:too_few_{host}_requests" ]
    }
}
```

Supported actions are drop tag or field with specified value. Added tag names and field names and field values can have `%{dynamic}` values.

`remove_field`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
    range {
        remove_field => [ "foo_{somefield}" ]
    }
}
# You can also remove multiple fields at once:
filter {
    range {
        remove_field => [ "foo_{somefield}", "my_extraneous_field" ]
    }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
    range {
        remove_tag => [ "foo_%{somefield}" ]
    }
}
# You can also remove multiple tags at once:
filter {
    range {
        remove_tag => [ "foo_%{somefield}", "sad_unwanted_tag" ]
    }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

ruby

Version: 3.0.2

Released on: 2016-07-14

[Changelog](#)

Execute ruby code.

For example, to cancel 90% of events, you can do this:

```
filter {
  ruby {
    # Cancel 90% of events
    code => "event.cancel if rand <= 0.90"
  }
}
```

If you need to create additional events, it cannot be done as in other filters where you would use `yield`, you must use a specific syntax `new_event_block.call(event)` like in this example duplicating the input event

```
filter {
  ruby {
    code => "new_event_block.call(event.clone)"
  }
}
```

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
ruby {
  code => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }

Setting	Input type	Required	Default value
add_tag	array	No	[]
code	string	Yes	
enable_metric	boolean	No	true
id	string	No	
init	string	No	
periodic_flush	boolean	No	false
remove_field	array	No	[]
remove_tag	array	No	[]

Details

[add_field](#)

Value type is [hash](#)

Default value is { }

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
  ruby {
    add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
  }
  # You can also add multiple fields at once:
  filter {
    ruby {
      add_field => {
        "foo_%{somefield}" => "Hello world, from %{host}"
        "new_field" => "new_static_value"
      }
    }
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

`add_tag`

Value type is [array](#)

Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  ruby {
    add_tag => [ "foo_%{somefield}" ]
  }
}
# You can also add multiple tags at once:
filter {
  ruby {
    add_tag => [ "foo_%{somefield}", "taggedy_tag" ]
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`code`

This is a required setting.

Value type is [string](#)

There is no default value for this setting.

The code to execute for every event. You will have an `event` variable available that is the event itself.

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
  stdout {  
    id => "my_plugin_id"  
  }  
}  
init
```

Value type is [string](#)

There is no default value for this setting.

Any code to execute at logstash startup-time

```
periodic_flush
```

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

```
remove_field
```

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {  
  ruby {  
    remove_field => [ "foo_{somefield}" ]  
  }  
}  
# You can also remove multiple fields at once:  
filter {  
  ruby {  
    remove_field => [ "foo_{somefield}", "my_extraneous_field" ]  
  }  
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  ruby {
    remove_tag => [ "foo_%{somefield}" ]
  }
}
# You can also remove multiple tags at once:
filter {
  ruby {
    remove_tag => [ "foo_%{somefield}", "sad_unwanted_tag" ]
  }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

sleep

Version: 3.0.3

Released on: 2016-12-26

[Changelog](#)

Sleep a given amount of time. This will cause logstash to stall for the given amount of time. This is useful for rate limiting, etc.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
sleep {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
add_tag	array	No	[]
enable_metric	boolean	No	true
every	string	No	1
id	string	No	
periodic_flush	boolean	No	false
remove_field	array	No	[]
remove_tag	array	No	[]
replay	boolean	No	false
time	string	No	

Details

`add_field`

Value type is [hash](#)

Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
  sleep {
    add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
  }
}
# You can also add multiple fields at once:
filter {
  sleep {
    add_field => {
      "foo_%{somefield}" => "Hello world, from %{host}"
      "new_field" => "new_static_value"
    }
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

`add_tag`

Value type is [array](#)

Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  sleep {
    add_tag => [ "foo_%{somefield}" ]
  }
}
# You can also add multiple tags at once:
filter {
  sleep {
    add_tag => [ "foo_%{somefield}", "taggedy_tag" ]
  }
}
```

}

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`every`

Value type is [string](#)

Default value is `1`

Sleep on every N'th. This option is ignored in replay mode.

Example:

```
filter {
    sleep {
        time => "1"      # Sleep 1 second
        every => 10      # on every 10th event
    }
}
id
```

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
    stdout {
        id => "my_plugin_id"
    }
}
periodic_flush
```

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

`remove_field`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
  sleep {
    remove_field => [ "foo_{somefield}" ]
  }
}
# You can also remove multiple fields at once:
filter {
  sleep {
    remove_field => [ "foo_{somefield}", "my_extraneous_field" ]
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  sleep {
    remove_tag => [ "foo_{somefield}" ]
  }
}
# You can also remove multiple tags at once:
filter {
  sleep {
    remove_tag => [ "foo_{somefield}", "sad_unwanted_tag" ]
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

`replay`

Value type is [boolean](#)

Default value is `false`

Enable replay mode.

Replay mode tries to sleep based on timestamps in each event.

The amount of time to sleep is computed by subtracting the previous event's timestamp from the current event's timestamp. This helps you replay events in the same timeline as original.

If you specify a `time` setting as well, this filter will use the `time` value as a speed modifier. For example, a `time` value of 2 will replay at double speed, while a value of 0.25 will replay at 1/4th speed.

For example:

```
filter {
    sleep {
        time => 2
        replay => true
    }
}
```

The above will sleep in such a way that it will perform replay 2-times faster than the original time speed.

`time`

Value type is [string](#)

There is no default value for this setting.

The length of time to sleep, in seconds, for every event.

This can be a number (eg, 0.5), or a string (eg, `%{foo}`) The second form (string with a field value) is useful if you have an attribute of your event that you want to use to indicate the amount of time to sleep.

Example:

```
filter {
    sleep {
```

Logstash 5.2 Configuration Guide

```
# Sleep 1 second for every event.  
time => "1"  
}  
}
```

split

Version: 3.1.1

Released on: 2016-07-14

[Changelog](#)

The split filter clones an event by splitting one of its fields and placing each value resulting from the split into a clone of the original event. The field being split can either be a string or an array.

An example use case of this filter is for taking output from the [exec input plugin](#) which emits one event for the whole output of a command and splitting that output by newline - making each line an event.

Split filter can also be used to split array fields in events into individual events. A very common pattern in JSON & XML is to make use of lists to group data together.

For example, a json structure like this:

```
{
  field1: ...,
  results: [
    { result ... },
    { result ... },
    { result ... },
    ...
  ]
}
```

The split filter can be used on the above data to create separate events for each value of `results` field

```
filter {
  split {
    field => "results"
  }
}
```

The end result of each split is a complete copy of the event with only the current split section of the given field changed.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
split {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
add_tag	array	No	[]
enable_metric	boolean	No	true
field	string	No	"message"
id	string	No	
periodic_flush	boolean	No	false
remove_field	array	No	[]
remove_tag	array	No	[]
target	string	No	
terminator	string	No	"\n"

Details

[add_field](#)

Value type is [hash](#)

Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the %{field}.

Example:

```
filter {  
    split {  
        add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }  
    }  
}  
# You can also add multiple fields at once:  
filter {  
    split {  
        add_field => {  
            "foo_%{somefield}" => "Hello world, from %{host}"  
            "new_field" => "new_static_value"  
        }  
    }  
}
```

}

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

`add_tag`

Value type is [array](#)

Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
    split {
        add_tag => [ "foo_%{somefield}" ]
    }
}
# You can also add multiple tags at once:
filter {
    split {
        add_tag => [ "foo_%{somefield}", "taggedy_tag" ]
    }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`enable_metric`

Value type is [boolean](#)

Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`field`

Value type is [string](#)

Default value is "message"

The field which value is split by the terminator. Can be a multiline message or the ID of an array. Nested arrays are referenced like: "[object_id][array_id]"

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
  stdout {
    id => "my_plugin_id"
  }
}
periodic_flush
```

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

`remove_field`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
  split {
    remove_field => [ "foo_{somefield}" ]
  }
}
# You can also remove multiple fields at once:
filter {
  split {
    remove_field => [ "foo_{somefield}", "my_extraneous_field" ]
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

remove_tag

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  split {
    remove_tag => [ "foo_%{somefield}" ]
  }
}
# You can also remove multiple tags at once:
filter {
  split {
    remove_tag => [ "foo_%{somefield}", "sad_unwanted_tag" ]
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

target

Value type is [string](#)

There is no default value for this setting.

The field within the new event which the value is split into. If not set, the target field defaults to split field name.

terminator

Value type is [string](#)

Default value is "\n"

The string to split on. This is usually a line terminator, but can be any string. If you are splitting a JSON array into multiple events, you can ignore this field.

syslog_pri

Version: 3.0.2

Released on: 2016-07-14

[Changelog](#)

Filter plugin for logstash to parse the `PRI` field from the front of a Syslog (RFC3164) message. If no priority is set, it will default to 13 (per RFC).

This filter is based on the original `syslog.rb` code shipped with logstash.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
syslog_pri { }
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
add_tag	array	No	[]
enable_metric	boolean	No	true
facility_labels	array	No	["kernel", "user-level", "mail", "daemon", "security/authorization", "syslogd", "line printer", "network news", "uucp", "clock", "security/authorization", "ftp", "ntp", "log audit", "log alert", "clock", "local0", "local1", "local2", "local3", "local4", "local5", "local6", "local7"]
id	string	No	
periodic_flush	boolean	No	false
remove_field	array	No	[]
remove_tag	array	No	[]
severity_labels	array	No	["emergency", "alert", "critical", "error", "warning", "notice", "informational", "debug"]
syslog_pri_field_name	string	No	"syslog_pri"
use_labels	boolean	No	true

Details

[add_field](#)

Value type is [hash](#)

Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
  syslog_pri {
    add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
  }
  # You can also add multiple fields at once:
  filter {
    syslog_pri {
      add_field => {
        "foo_%{somefield}" => "Hello world, from %{host}"
        "new_field" => "new_static_value"
      }
    }
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

[add_tag](#)

Value type is [array](#)

Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  syslog_pri {
    add_tag => [ "foo_%{somefield}" ]
  }
}
```

```
# You can also add multiple tags at once:
filter {
  syslog_pri {
    add_tag => [ "foo_%{somefield}", "taggedy_tag" ]
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`facility_labels`

Value type is [array](#)

Default value is `["kernel", "user-level", "mail", "daemon", "security/authorization", "syslogd", "line printer", "network news", "uucp", "clock", "security/authorization", "ftp", "ntp", "log audit", "log alert", "clock", "local0", "local1", "local2", "local3", "local4", "local5", "local6", "local7"]`

Labels for facility levels. This comes from RFC3164.

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
  stdout {
    id => "my_plugin_id"
  }
}
```

periodic_flush

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

remove_field

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
  syslog_pri {
    remove_field => [ "foo_{somefield}" ]
  }
}
# You can also remove multiple fields at once:
filter {
  syslog_pri {
    remove_field => [ "foo_{somefield}", "my_extraneous_field" ]
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

remove_tag

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  syslog_pri {
    remove_tag => [ "foo_{somefield}" ]
  }
}
# You can also remove multiple tags at once:
filter {
```

Logstash 5.2 Configuration Guide

```
    syslog_pri {  
        remove_tag => [ "foo_{somefield}", "sad_unwanted_tag"]  
    }  
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

severity_labels

Value type is [array](#)

Default value is ["emergency", "alert", "critical", "error", "warning", "notice", "informational", "debug"]

Labels for severity levels. This comes from RFC3164.

syslog_pri_field_name

Value type is [string](#)

Default value is "syslog_pri"

Name of field which passes in the extracted PRI part of the syslog message

use_labels

Value type is [boolean](#)

Default value is `true`

set the status to experimental/beta/stable Add human-readable names after parsing severity and facility from PRI

throttle

Version: 4.0.1

Released on: 2016-09-17

[Changelog](#)

The throttle filter is for throttling the number of events. The filter is configured with a lower bound, the "before_count", and upper bound, the "after_count", and a period of time. All events passing through the filter will be counted based on their key and the event timestamp. As long as the count is less than the "before_count" or greater than the "after_count", the event will be "throttled" which means the filter will be considered successful and any tags or fields will be added (or removed).

The plugin is thread-safe and properly tracks past events.

For example, if you wanted to throttle events so you only receive an event after 2 occurrences and you get no more than 3 in 10 minutes, you would use the configuration:

```
period => 600
max_age => 1200
before_count => 3
after_count => 5
```

Which would result in:

```
event 1 - throttled (successful filter, period start)
event 2 - throttled (successful filter)
event 3 - not throttled
event 4 - not throttled
event 5 - not throttled
event 6 - throttled (successful filter)
event 7 - throttled (successful filter)
event x - throttled (successful filter)
period end
event 1 - throttled (successful filter, period start)
event 2 - throttled (successful filter)
event 3 - not throttled
event 4 - not throttled
event 5 - not throttled
event 6 - throttled (successful filter)
...

```

Another example is if you wanted to throttle events so you only receive 1 event per hour, you would use the configuration:

```
period => 3600
max_age => 7200
before_count => -1
```

```
after_count => 1
```

Which would result in:

```
event 1 - not throttled (period start)
event 2 - throttled (successful filter)
event 3 - throttled (successful filter)
event 4 - throttled (successful filter)
event x - throttled (successful filter)
period end
event 1 - not throttled (period start)
event 2 - throttled (successful filter)
event 3 - throttled (successful filter)
event 4 - throttled (successful filter)
...
...
```

A common use case would be to use the throttle filter to throttle events before 3 and after 5 while using multiple fields for the key and then use the drop filter to remove throttled events. This configuration might appear as:

```
filter {
  throttle {
    before_count => 3
    after_count => 5
    period => 3600
    max_age => 7200
    key => "%{host}%{message}"
    add_tag => "throttled"
  }
  if "throttled" in [tags] {
    drop {}
  }
}
```

Another case would be to store all events, but only email non-throttled events so the op's inbox isn't flooded with emails in the event of a system error. This configuration might appear as:

```
filter {
  throttle {
    before_count => 3
    after_count => 5
    period => 3600
    max_age => 7200
    key => "%{message}"
    add_tag => "throttled"
  }
}
output {
  if "throttled" not in [tags] {
    email {
      from => "logstash@mycompany.com"
      subject => "Production System Alert"
      to => "ops@mycompany.com"
      via => "sendmail"
    }
  }
}
```

```

        body => "Alert on %{host} from path %{path}:\n\n%{message}"
        options => { "location" => "/usr/sbin/sendmail" }
    }
}
elasticsearch_http {
    host => "localhost"
    port => "19200"
}
}

```

When an event is received, the event key is stored in a `key_cache`. The key references a `timeslot_cache`. The event is allocated to a timeslot (created dynamically) based on the timestamp of the event. The timeslot counter is incremented. When the next event is received (same key), within the same "period", it is allocated to the same timeslot. The timeslot counter is incremented once again.

The timeslot expires if the maximum age has been exceeded. The age is calculated based on the latest event timestamp and the `max_age` configuration option.

---[::: . DESIGN .:::]---

```

- [key_cache] --- [timeslot_cache] -- ||| @created: 1439839636 || @latest:
1439839836 | [a.b.c] => ----- | [1439839636] => 1 || [1439839736] => 3 ||
[1439839836] => 2 | -----
+
+--- [timeslot_cache] --+
| @created: eeeeeeeeeee |
| @latest: lllllllllll |
[x.y.z]  => +-----+
| [0000000060] => x   |
| [0000000120] => y   |
| | [.....] => N   |
+-----+ +-----+

```

Frank de Jong (@frapex) Mike Pilone (@mikepilone)

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
throttle {
    key => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
add_tag	array	No	[]
after_count	number	No	-1
before_count	number	No	-1
enable_metric	boolean	No	true
id	string	No	
key	string	Yes	
max_age	number	No	3600
max_counters	number	No	100000
period	string	No	"60"
periodic_flush	boolean	No	true
remove_field	array	No	[]
remove_tag	array	No	[]

Details

[add_field](#)

Value type is [hash](#)

Default value is { }

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
  throttle {
    add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
  }
}
# You can also add multiple fields at once:
filter {
  throttle {
    add_field => {
      "foo_%{somefield}" => "Hello world, from %{host}"
      "new_field" => "new_static_value"
    }
}
```

}

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

`add_tag`

Value type is [array](#)

Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  throttle {
    add_tag => [ "foo_%{somefield}" ]
  }
}
# You can also add multiple tags at once:
filter {
  throttle {
    add_tag => [ "foo_%{somefield}", "taggedy_tag" ]
  }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`after_count`

Value type is [number](#)

Default value is -1

Events greater than this count will be throttled. Setting this value to -1, the default, will cause no events to be throttled based on the upper bound.

`before_count`

Value type is [number](#)

Default value is -1

Events less than this count will be throttled. Setting this value to -1, the default, will cause no events to be throttled based on the lower bound.

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
  stdout {  
    id => "my_plugin_id"  
  }  
}
```

`key`

This is a required setting.

Value type is [string](#)

There is no default value for this setting.

The key used to identify events. Events with the same key are grouped together. Field substitutions are allowed, so you can combine multiple fields.

`max_age`

Value type is [number](#)

Default value is 3600

The maximum age of a timeslot. Higher values allow better tracking of an asynchronous flow of events, but require more memory. As a rule of thumb you should set this value to at least twice the period. Or set this value to period + maximum time offset between unordered events with the same key. Values below the specified period give unexpected results if unordered events are processed simultaneously.

max_counters

Value type is [number](#)

Default value is 100000

The maximum number of counters to store before decreasing the maximum age of a timeslot. Setting this value to -1 will prevent an upper bound with no constraint on the number of counters. This configuration value should only be used as a memory control mechanism and can cause early counter expiration if the value is reached. It is recommended to leave the default value and ensure that your key is selected such that it limits the number of counters required (i.e. don't use UUID as the key).

period

Value type is [string](#)

Default value is "60"

The period in seconds after the first occurrence of an event until a new timeslot is created. This period is tracked per unique key and per timeslot. Field substitutions are allowed in this value. This allows you to specify that certain kinds of events throttle for a specific period of time.

periodic_flush

Value type is [boolean](#)

Default value is true

The name to use in configuration files. The memory control mechanism automatically adjusts the maximum age of a timeslot based on the maximum number of counters. Call the filter flush method at regular interval. It is used by the memory control mechanism. Set to false if you like your VM to go (B)OOM.

remove_field

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
  throttle {
    remove_field => [ "foo_{somefield}" ]
  }
}
# You can also remove multiple fields at once:
```

```
filter {
  throttle {
    remove_field => [ "foo_{somefield}", "my_extraneous_field" ]
  }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  throttle {
    remove_tag => [ "foo_{somefield}" ]
  }
}
# You can also remove multiple tags at once:
filter {
  throttle {
    remove_tag => [ "foo_{somefield}", "sad_unwanted_tag" ]
  }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

tld

Version: 3.0.0

Released on: 2016-10-22

[Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-filter-tld`.

This example filter will replace the contents of the default message field with whatever you specify in the configuration.

It is only intended to be used as an example.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
tld {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
add_tag	array	No	[]
enable_metric	boolean	No	true
id	string	No	
periodic_flush	boolean	No	false
remove_field	array	No	[]
remove_tag	array	No	[]
source	string	No	"message"
target	string	No	"tld"

Details

`add_field`

Value type is [hash](#)

Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
  tld {
    add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
  }
}
# You can also add multiple fields at once:
filter {
  tld {
    add_field => {
      "foo_%{somefield}" => "Hello world, from %{host}"
      "new_field" => "new_static_value"
    }
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

`add_tag`

Value type is [array](#)

Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  tld {
    add_tag => [ "foo_%{somefield}" ]
  }
}
# You can also add multiple tags at once:
filter {
```

Logstash 5.2 Configuration Guide

```
tld {  
    add_tag => [ "foo_{somefield}", "taggedy_tag"]  
}  
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

`periodic_flush`

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

`remove_field`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 tld {
```

## Logstash 5.2 Configuration Guide

```
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 tld {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name foo\_hello if it is present. The second example would remove an additional, non-dynamic field.

### remove\_tag

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the %{field} syntax.

Example:

```
filter {
 tld {
 remove_tag => ["foo_{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 tld {
 remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag foo\_hello if it is present. The second example would remove a sad, unwanted tag as well.

### source

Value type is [string](#)

Default value is "message"

Setting the config\_name here is required. This is how you configure this filter from your Logstash config.

```
filter { example { message => "My message..." } }
```

The source field to parse

*target*

Value type is [string](#)

Default value is "tld"

The target field to place all the data

## translate

Version: 3.0.1

Released on: 2016-12-26

[Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
bin/logstash-plugin install logstash-filter-translate.

A general search and replace tool that uses a configured hash and/or a file to determine replacement values. Currently supported are YAML, JSON, and CSV files.

The dictionary entries can be specified in one of two ways: First, the `dictionary` configuration item may contain a hash representing the mapping. Second, an external file (readable by logstash) may be specified in the `dictionary_path` configuration item. These two methods may not be used in conjunction; it will produce an error.

Operationally, if the event field specified in the `field` configuration matches the EXACT contents of a dictionary entry key (or matches a regex if `regex` configuration item has been enabled), the field's value will be substituted with the matched key's value from the dictionary.

By default, the translate filter will replace the contents of the matching event field (in-place). However, by using the `destination` configuration item, you may also specify a target event field to populate with the new translated value.

Alternatively, for simple string search and replacements for just a few values you might consider using the `gsub` function of the mutate filter.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
translate {
 field => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
add_tag	<a href="#">array</a>	No	[]
destination	<a href="#">string</a>	No	"translation"
dictionary	<a href="#">hash</a>	No	{ }
dictionary_path	a valid filesystem path	No	
enable_metric	<a href="#">boolean</a>	No	true
exact	<a href="#">boolean</a>	No	true
fallback	<a href="#">string</a>	No	
field	<a href="#">string</a>	Yes	
id	<a href="#">string</a>	No	
override	<a href="#">boolean</a>	No	false
periodic_flush	<a href="#">boolean</a>	No	false
refresh_interval	<a href="#">number</a>	No	300
regex	<a href="#">boolean</a>	No	false
remove_field	<a href="#">array</a>	No	[]
remove_tag	<a href="#">array</a>	No	[]

## Details

### [add\\_field](#)

Value type is [hash](#)

Default value is { }

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 translate {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 translate {
 add_field => {
```

```
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
}
}
}
```

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

### *add\_tag*

Value type is [array](#)

Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 translate {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 translate {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

### *destination*

Value type is [string](#)

Default value is "translation"

The destination field you wish to populate with the translated code. The default is a field named `translation`. Set this to the same value as source if you want to do a substitution, in this case filter will allways succeed. This will clobber the old value of the source field!

### *dictionary*

Value type is [hash](#)

Default value is { }

The dictionary to use for translation, when specified in the logstash filter configuration item (i.e. do not use the `@dictionary_path` file).

Example:

```
filter {
 translate {
 dictionary => ["100", "Continue",
 "101", "Switching Protocols",
 "merci", "thank you",
 "old version", "new version"]
 }
}
```

It is an error to specify both `dictionary` and `dictionary_path`.

*dictionary\_path*

Value type is [path](#)

There is no default value for this setting.

The full path of the external dictionary file. The format of the table should be a standard YAML, JSON, or CSV. Make sure you specify any integer-based keys in quotes. For example, the YAML file should look something like this:

```
"100": Continue
"101": Switching Protocols
merci: gracias
old version: new version
```

it is an error to specify both `dictionary` and `dictionary_path`.

The currently supported formats are YAML, JSON, and CSV. Format selection is based on the file extension: `json` for JSON, `yaml` or `yml` for YAML, and `csv` for CSV. The JSON format only supports simple key/value, unnested objects. The CSV format expects exactly two columns, with the first serving as the original text, and the second column as the replacement.

*enable\_metric*

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

*exact*

Value type is [boolean](#)

Default value is `true`

When `exact => true`, the translate filter will populate the destination field with the exact contents of the dictionary value. When `exact => false`, the filter will populate the destination field with the result of any existing destination field's data, with the translated value substituted in-place.

For example, consider this simple `translation.yml`, configured to check the `data` field:

```
foo: bar
```

If logstash receives an event with the `data` field set to `foo`, and `exact => true`, the destination field will be populated with the string `bar`. If `exact => false`, and logstash receives the same event, the destination field will be also set to `bar`. However, if logstash receives an event with the `data` field set to `foofing`, the destination field will be set to `barfing`.

Set both `exact => true` AND `regex => `true`` if you would like to match using dictionary keys as regular expressions. A large dictionary could be expensive to match in this case.

*fallback*

Value type is [string](#)

There is no default value for this setting.

In case no translation occurs in the event (no matches), this will add a default translation string, which will always populate `field`, if the match failed.

For example, if we have configured `fallback => "no match"`, using this dictionary:

```
foo: bar
```

Then, if logstash received an event with the field `foo` set to `bar`, the destination field would be set to `bar`. However, if logstash received an event with `foo` set to `nope`, then the destination field would still be populated, but with the value of `no match`. This configuration can be dynamic and include parts of the event using the `%{field}` syntax.

*field*

This is a required setting.

Value type is [string](#)

There is no default value for this setting.

The name of the logstash event field containing the value to be compared for a match by the translate filter (e.g. `message`, `host`, `response_code`).

If this field is an array, only the first value will be used.

*id*

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

override

Value type is [boolean](#)

Default value is `false`

If the destination (or target) field already exists, this configuration item specifies whether the filter should skip translation (default) or overwrite the target field value with the new translation value.

periodic_flush

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

refresh_interval

Value type is [number](#)

Default value is 300

When using a dictionary file, this setting will indicate how frequently (in seconds) logstash will check the dictionary file for updates.

`regex`

Value type is [boolean](#)

Default value is `false`

If you'd like to treat dictionary keys as regular expressions, set `exact => true`. Note: this is activated only when `exact => true`.

`remove_field`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
  translate {
    remove_field => [ "foo_%{somefield}" ]
  }
}
# You can also remove multiple fields at once:
filter {
  translate {
    remove_field => [ "foo_%{somefield}", "my_extraneous_field" ]
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  translate {
```

Logstash 5.2 Configuration Guide

```
        remove_tag => [ "foo_%{somefield}" ]
    }
}
# You can also remove multiple tags at once:
filter {
    translate {
        remove_tag => [ "foo_%{somefield}", "sad_unwanted_tag" ]
    }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

truncate

Version: 1.0.0

Released on: 2016-11-29

[Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-filter-truncate`.

Allows you to truncate fields longer than a given length.

This truncates on bytes values, not character count. In practice, this should mean that the truncated length is somewhere between `length_bytes` and `length_bytes - 6` (UTF-8 supports up to 6-byte characters).

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
truncate {
    length_bytes => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>add_field</code>	hash	No	<code>{ }</code>
<code>add_tag</code>	array	No	<code>[]</code>
<code>enable_metric</code>	boolean	No	<code>true</code>
<code>fields</code>	string	No	
<code>id</code>	string	No	
<code>length_bytes</code>	number	Yes	
<code>periodic_flush</code>	boolean	No	<code>false</code>
<code>remove_field</code>	array	No	<code>[]</code>

Setting	Input type	Required	Default value
remove_tag	array	No	[]

Details

[*add_field*](#)

Value type is [hash](#)

Default value is { }

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
  truncate {
    add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
  }
  # You can also add multiple fields at once:
  filter {
    truncate {
      add_field => {
        "foo_%{somefield}" => "Hello world, from %{host}"
        "new_field" => "new_static_value"
      }
    }
  }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

[*add_tag*](#)

Value type is [array](#)

Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
  truncate {
    add_tag => [ "foo_%{somefield}" ]
  }
}
# You can also add multiple tags at once:
filter {
  truncate {
    add_tag => [ "foo_%{somefield}", "taggedy_tag" ]
  }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

enable_metric

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

fields

Value type is [string](#)

There is no default value for this setting.

A list of fieldrefs to truncate if they are too long.

If not specified, the default behavior will be to attempt truncation on all strings in the event. This default behavior could be computationally expensive, so if you know exactly which fields you wish to truncate, it is advised that you be specific and configure the fields you want truncated.

Special behaviors for non-string fields:

Numbers: No action

Array: this plugin will attempt truncation on all elements of that array.

Hash: truncate will try all values of the hash (recursively, if this hash contains other hashes).

id

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

`length_bytes`

This is a required setting.

Value type is [number](#)

There is no default value for this setting.

Fields over this length will be truncated to this length.

Truncation happens from the end of the text (the start will be kept).

As an example, if you set `length_bytes => 10` and a field contains "hello world, how are you?", then this field will be truncated and have this value: "hello worl"

`periodic_flush`

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

`remove_field`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 truncate {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 truncate {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

Value type is [array](#)

Default value is [ ]

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 truncate {
 remove_tag => ["foo_%{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 truncate {
 remove_tag => ["foo_%{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

## urldecode

Version: 3.0.2

Released on: 2016-07-14

[Changelog](#)

The urldecode filter is for decoding fields that are urlencoded.

### Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
urldecode {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">add_tag</a>	<a href="#">array</a>	No	[]
<a href="#">all_fields</a>	<a href="#">boolean</a>	No	false
<a href="#">charset</a>	<a href="#">string</a> , one of ["ASCII-8BIT", "UTF-8", "US-ASCII", "Big5", "Big5-HKSCS", "Big5-UAO", "CP949", "Emacs-Mule", "EUC-JP", "EUC-KR", "EUC-TW", "GB2312", "GB18030", "GBK", "ISO-8859-1", "ISO-8859-2", "ISO-8859-3", "ISO-8859-4", "ISO-8859-5", "ISO-8859-6", "ISO-8859-7", "ISO-8859-8", "ISO-8859-9", "ISO-8859-10", "ISO-8859-11", "ISO-8859-13", "ISO-8859-14", "ISO-8859-15", "ISO-8859-16", "KOI8-R", "KOI8-U", "Shift_JIS", "UTF-16BE", "UTF-16LE", "UTF-32BE", "UTF-32LE", "Windows-31J", "Windows-1250", "Windows-1251", "Windows-1252", "IBM437", "IBM737", "IBM775", "CP850", "IBM852", "CP852", "IBM855", "CP855", "IBM857", "IBM860", "IBM861", "IBM862", "IBM863", "IBM864", "IBM865", "IBM866", "IBM869", "Windows-1258", "GB1988", "macCentEuro", "macCroatian", "macCyrillic", "macGreek",	No	"UTF-8"

Setting	Input type	Required	Default value
	"macIceland", "macRoman", "macRomania", "macThai", "macTurkish", "macUkraine", "CP950", "CP951", "IBM037", "stateless-ISO-2022-JP", "eucJP-ms", "CP51932", "EUC-JIS-2004", "GB12345", "ISO-2022-JP", "ISO-2022-JP-2", "CP50220", "CP50221", "Windows-1256", "Windows-1253", "Windows-1255", "Windows-1254", "TIS-620", "Windows-874", "Windows-1257", "MacJapanese", "UTF-7", "UTF8-MAC", "UTF-16", "UTF-32", "UTF8-DoCoMo", "SJIS-DoCoMo", "UTF8-KDDI", "SJIS-KDDI", "ISO-2022-JP-KDDI", "stateless-ISO-2022-JP-KDDI", "UTF8-SoftBank", "SJIS-SoftBank", "BINARY", "CP437", "CP737", "CP775", "IBM850", "CP857", "CP860", "CP861", "CP862", "CP863", "CP864", "CP865", "CP866", "CP869", "CP1258", "Big5-HKSCS:2008", "ebcdic-cp-us", "eucJP", "euc-jp-ms", "EUC-JISX0213", "eucKR", "eucTW", "EUC-CN", "eucCN", "CP936", "ISO2022-JP", "ISO2022-JP2", "ISO8859-1", "ISO8859-2", "ISO8859-3", "ISO8859-4", "ISO8859-5", "ISO8859-6", "CP1256", "ISO8859-7", "CP1253", "ISO8859-8", "CP1255", "ISO8859-9", "CP1254", "ISO8859-10", "ISO8859-11", "CP874", "ISO8859-13", "CP1257", "ISO8859-14", "ISO8859-15", "ISO8859-16", "CP878", "MacJapan", "ASCII", "ANSI_X3.4-1968", "646", "CP65000", "CP65001", "UTF-8-MAC", "UTF-8-HFS", "UCS-2BE", "UCS-4BE", "UCS-4LE", "CP932", "csWindows31J", "SJIS", "PCK", "CP1250", "CP1251", "CP1252", "external", "locale"]		
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">field</a>	<a href="#">string</a>	No	"message"
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">periodic_flush</a>	<a href="#">boolean</a>	No	false
<a href="#">remove_field</a>	<a href="#">array</a>	No	[]
<a href="#">remove_tag</a>	<a href="#">array</a>	No	[]

## Details

[add\\_field](#)

Value type is [hash](#)

Default value is { }

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 urldecode {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 urldecode {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

[`add\_tag`](#)

Value type is [array](#)

Default value is [ ]

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 urldecode {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 urldecode {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`all_fields`

Value type is [boolean](#)

Default value is `false`

Urldecode all fields

`charset`

Value can be any of: ASCII-8BIT, UTF-8, US-ASCII, Big5, Big5-HKSCS, Big5-UAO, CP949, Emacs-Mule, EUC-JP, EUC-KR, EUC-TW, GB2312, GB18030, GBK, ISO-8859-1, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9, ISO-8859-10, ISO-8859-11, ISO-8859-13, ISO-8859-14, ISO-8859-15, ISO-8859-16, KOI8-R, KOI8-U, Shift\_JIS, UTF-16BE, UTF-16LE, UTF-32BE, UTF-32LE, Windows-31J, Windows-1250, Windows-1251, Windows-1252, IBM437, IBM737, IBM775, CP850, IBM852, CP852, IBM855, CP855, IBM857, IBM860, IBM861, IBM862, IBM863, IBM864, IBM865, IBM866, IBM869, Windows-1258, GB1988, macCentEuro, macCroatian, macCyrillic, macGreek, macIceland, macRoman, macRomania, macThai, macTurkish, macUkraine, CP950, CP951, IBM037, stateless-ISO-2022-JP, eucJP-ms, CP51932, EUC-JIS-2004, GB12345, ISO-2022-JP, ISO-2022-JP-2, CP50220, CP50221, Windows-1256, Windows-1253, Windows-1255, Windows-1254, TIS-620, Windows-874, Windows-1257, MacJapanese, UTF-7, UTF8-MAC, UTF-16, UTF-32, UTF8-DoCoMo, SJIS-DoCoMo, UTF8-KDDI, SJIS-KDDI, ISO-2022-JP-KDDI, stateless-ISO-2022-JP-KDDI, UTF8-SoftBank, SJIS-SoftBank, BINARY, CP437, CP737, CP775, IBM850, CP857, CP860, CP861, CP862, CP863, CP864, CP865, CP866, CP869, CP1258, Big5-HKSCS:2008, ebcdic-cp-us, eucJP, euc-jp-ms, EUC-JISX0213, eucKR, eucTW, EUC-CN, eucCN, CP936, ISO2022-JP, ISO2022-JP2, ISO8859-1, ISO8859-2, ISO8859-3, ISO8859-4, ISO8859-5, ISO8859-6, CP1256, ISO8859-7, CP1253, ISO8859-8, CP1255, ISO8859-9, CP1254, ISO8859-10, ISO8859-11, CP874, ISO8859-13, CP1257, ISO8859-14, ISO8859-15, ISO8859-16, CP878, MacJapan, ASCII, ANSI\_X3.4-1968, 646, CP65000, CP65001, UTF-8-MAC, UTF-8-HFS, UCS-2BE, UCS-4BE, UCS-4LE, CP932, csWindows31J, SJIS, PCK, CP1250, CP1251, CP1252, external, locale

Default value is "UTF-8"

The character encoding used in this filter. Examples include `UTF-8` and `cp1252`

This setting is useful if your url decoded string are in Latin-1 (aka `cp1252`) or in another character set other than `UTF-8`.

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

*field*

Value type is [string](#)

Default value is "message"

The field which value is urldecoded

*id*

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
periodic_flush
```

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

*remove\_field*

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 urldecode {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
```

```
filter {
 urldecode {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 urldecode {
 remove_tag => ["foo_{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 urldecode {
 remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

## useragent

Version: 3.0.3

Released on: 2016-09-15

[Changelog](#)

Parse user agent strings into structured data based on BrowserScope data

UserAgent filter, adds information about user agent like family, operating system, version, and device

Logstash releases ship with the regexes.yaml database made available from ua-parser with an Apache 2.0 license. For more details on ua-parser, see <https://github.com/tobie/ua-parser/>.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
useragent {
 source => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">add_tag</a>	<a href="#">array</a>	No	[]
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">lru_cache_size</a>	<a href="#">number</a>	No	1000
<a href="#">periodic_flush</a>	<a href="#">boolean</a>	No	false
<a href="#">prefix</a>	<a href="#">string</a>	No	""
<a href="#">regexes</a>	<a href="#">string</a>	No	
<a href="#">remove_field</a>	<a href="#">array</a>	No	[]
<a href="#">remove_tag</a>	<a href="#">array</a>	No	[]
<a href="#">source</a>	<a href="#">string</a>	Yes	

Setting	Input type	Required	Default value
<a href="#">target</a>	<a href="#">string</a>	No	

## Details

[add\\_field](#)

Value type is [hash](#)

Default value is { }

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 useragent {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
 # You can also add multiple fields at once:
 filter {
 useragent {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

[add\\_tag](#)

Value type is [array](#)

Default value is [ ]

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 useragent {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 useragent {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

[`enable\_metric`](#)

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

[`id`](#)

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

[`lru\_cache\_size`](#)

Value type is [number](#)

Default value is `1000`

UA parsing is surprisingly expensive. This filter uses an LRU cache to take advantage of the fact that user agents are often found adjacent to one another in log files and rarely have a random

distribution. The higher you set this the more likely an item is to be in the cache and the faster this filter will run. However, if you set this too high you can use more memory than desired.

Experiment with different values for this option to find the best performance for your dataset.

This MUST be set to a value  $> 0$ . There is really no reason to not want this behavior, the overhead is minimal and the speed gains are large.

It is important to note that this config value is global. That is to say all instances of the user agent filter share the same cache. The last declared cache size will *win*. The reason for this is that there would be no benefit to having multiple caches for different instances at different points in the pipeline, that would just increase the number of cache misses and waste memory.

*periodic\_flush*

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

*prefix*

Value type is [string](#)

Default value is `""`

A string to prepend to all of the extracted keys

*regexes*

Value type is [string](#)

There is no default value for this setting.

`regexes.yaml` file to use

If not specified, this will default to the `regexes.yaml` that ships with logstash.

You can find the latest version of this here: <https://github.com/ua-parser/uap-core/blob/master/regexes.yaml>

*remove\_field*

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 useragent {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 useragent {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name foo\_hello if it is present. The second example would remove an additional, non-dynamic field.

[remove\\_tag](#)

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the %{field} syntax.

Example:

```
filter {
 useragent {
 remove_tag => ["foo_{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 useragent {
 remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag foo\_hello if it is present. The second example would remove a sad, unwanted tag as well.

[source](#)

This is a required setting.

Value type is [string](#)

There is no default value for this setting.

The field containing the user agent string. If this field is an array, only the first value will be used.

*target*

Value type is [string](#)

There is no default value for this setting.

The name of the field to assign user agent data into.

If not specified user agent data will be stored in the root of the event.

# uuid

Version: 3.0.2

Released on: 2016-07-14

[Changelog](#)

The `uuid` filter allows you to add a `_UUID` field to messages. This is useful to be able to control the `_id` messages are indexed into Elasticsearch with, so that you can insert duplicate messages (i.e. the same message multiple times without creating duplicates) - for log pipeline reliability

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
uuid {
 target => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">add_tag</a>	<a href="#">array</a>	No	[]
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">overwrite</a>	<a href="#">boolean</a>	No	false
<a href="#">periodic_flush</a>	<a href="#">boolean</a>	No	false
<a href="#">remove_field</a>	<a href="#">array</a>	No	[]
<a href="#">remove_tag</a>	<a href="#">array</a>	No	[]
<a href="#">target</a>	<a href="#">string</a>	Yes	

## Details

### `add_field`

Value type is [hash](#)

Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 uuid {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
filter {
 uuid {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

### `add_tag`

Value type is [array](#)

Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 uuid {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 uuid {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

}

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
overwrite
```

Value type is [boolean](#)

Default value is `false`

If the value in the field currently (if any) should be overridden by the generated UUID. Defaults to `false` (i.e. if the field is present, with ANY value, it won't be overridden)

Example:

```
filter {
 uuid {
 target => "@uuid"
 overwrite => true
 }
}
```

*periodic\_flush*

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

*remove\_field*

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 uuid {
 remove_field => ["foo_{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 uuid {
 remove_field => ["foo_{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

*remove\_tag*

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 uuid {
 remove_tag => ["foo_{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
```

```
 uuid {
 remove_tag => ["foo_{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field "`somefield`" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

### `target`

This is a required setting.

Value type is [string](#)

There is no default value for this setting.

Add a UUID to a field.

Example:

```
filter {
 uuid {
 target => "@uuid"
 }
}
```

## xml

Version: 4.0.2

Released on: 2016-10-31

[Changelog](#)

XML filter. Takes a field that contains XML and expands it into an actual datastructure.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
xml {
 source => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">add_field</a>	<a href="#">hash</a>	No	{ }
<a href="#">add_tag</a>	<a href="#">array</a>	No	[]
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">force_array</a>	<a href="#">boolean</a>	No	true
<a href="#">force_content</a>	<a href="#">boolean</a>	No	false
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">namespaces</a>	<a href="#">hash</a>	No	{ }
<a href="#">periodic_flush</a>	<a href="#">boolean</a>	No	false
<a href="#">remove_field</a>	<a href="#">array</a>	No	[]
<a href="#">remove_namespaces</a>	<a href="#">boolean</a>	No	false
<a href="#">remove_tag</a>	<a href="#">array</a>	No	[]
<a href="#">source</a>	<a href="#">string</a>	Yes	
<a href="#">store_xml</a>	<a href="#">boolean</a>	No	true
<a href="#">suppress_empty</a>	<a href="#">boolean</a>	No	true
<a href="#">target</a>	<a href="#">string</a>	No	
<a href="#">xpath</a>	<a href="#">hash</a>	No	{ }

## Details

[add\\_field](#)

Value type is [hash](#)

Default value is { }

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 xml {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
}
You can also add multiple fields at once:
```

```
filter {
 xml {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

### `add_tag`

Value type is [array](#)

Default value is `[]`

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 xml {
 add_tag => ["foo_%{somefield}"]
 }
}
You can also add multiple tags at once:
filter {
 xml {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

### `enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

*force\_array*

Value type is [boolean](#)

Default value is `true`

By default the filter will force single elements to be arrays. Setting this to false will prevent storing single elements in arrays.

*force\_content*

Value type is [boolean](#)

Default value is `false`

By default the filter will expand attributes differently from content inside of tags. This option allows you to force text content and attributes to always parse to a hash value.

*id*

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
namespaces
```

Value type is [hash](#)

Default value is `{ }`

By default only namespaces declarations on the root element are considered. This allows to configure all namespace declarations to parse the XML document.

Example:

```
filter {
 xml {
 namespaces => {
```

## Logstash 5.2 Configuration Guide

```
"xsl" => "http://www.w3.org/1999/XSL/Transform"
"xhtml" => http://www.w3.org/1999/xhtml"
}
}
}
periodic_flush
```

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

```
remove_field
```

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
 xml {
 remove_field => ["foo_%{somefield}"]
 }
}
You can also remove multiple fields at once:
filter {
 xml {
 remove_field => ["foo_%{somefield}", "my_extraneous_field"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

```
remove_namespaces
```

Value type is [boolean](#)

Default value is `false`

Remove all namespaces from all nodes in the document. Of course, if the document had nodes with the same names but different namespaces, they will now be ambiguous.

```
remove_tag
```

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 xml {
 remove_tag => ["foo_%{somefield}"]
 }
}
You can also remove multiple tags at once:
filter {
 xml {
 remove_tag => ["foo_%{somefield}", "sad_unwanted_tag"]
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

`source`

This is a required setting.

Value type is [string](#)

There is no default value for this setting.

Config for xml to hash is:

`source => source_field`

For example, if you have the whole XML document in your `message` field:

```
filter {
 xml {
 source => "message"
 }
}
```

The above would parse the XML from the `message` field.

`store_xml`

Value type is [boolean](#)

Default value is `true`

By default the filter will store the whole parsed XML in the destination field as described above. Setting this to false will prevent that.

`suppress_empty`

Value type is [boolean](#)

Default value is `true`

By default, output nothing if the element is empty. If set to `false`, empty element will result in an empty hash object.

`target`

Value type is [string](#)

There is no default value for this setting.

Define target for placing the data

For example if you want the data to be put in the `doc` field:

```
filter {
 xml {
 target => "doc"
 }
}
```

XML in the value of the source field will be expanded into a datastructure in the `target` field. Note: if the `target` field already exists, it will be overridden. Required if `store_xml` is true (which is the default).

`xpath`

Value type is [hash](#)

Default value is `{ }`

`xpath` will additionally select string values (non-strings will be converted to strings with Ruby's `to_s` function) from parsed XML (using each source field defined using the method above) and place those values in the destination fields. Configuration:

```
xpath => ["xpath-syntax", "destination-field"]
```

Values returned by XPath parsing from `xpath-syntax` will be put in the destination field. Multiple values returned will be pushed onto the destination field as an array. As such, multiple matches across multiple source fields will produce duplicate entries in the field.

More on XPath: [http://www.w3schools.com/xml/xml\\_xpath.asp](http://www.w3schools.com/xml/xml_xpath.asp)

The XPath functions are particularly powerful: [http://www.w3schools.com/xsl/xsl\\_functions.asp](http://www.w3schools.com/xsl/xsl_functions.asp)

## yaml

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-filter-yaml`.

This is a YAML parsing filter. It takes an existing field which contains YAML and expands it into an actual data structure within the Logstash event.

By default it will place the parsed YAML in the root (top level) of the Logstash event, but this filter can be configured to place the YAML into any arbitrary event field, using the `target` configuration.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
yaml {
 source => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	<a href="#">hash</a>	No	{ }
add_tag	<a href="#">array</a>	No	[]
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	
periodic_flush	<a href="#">boolean</a>	No	false
remove_field	<a href="#">array</a>	No	[]
remove_tag	<a href="#">array</a>	No	[]
source	<a href="#">string</a>	Yes	
target	<a href="#">string</a>	No	

## Details

### `add_field`

Value type is [hash](#)

Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the `%{field}`.

Example:

```
filter {
 yaml {
 add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
 }
 # You can also add multiple fields at once:
 filter {
 yaml {
 add_field => {
 "foo_%{somefield}" => "Hello world, from %{host}"
 "new_field" => "new_static_value"
 }
 }
 }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add field `foo_hello` if it is present, with the value above and the `%{host}` piece replaced with that value from the event. The second example would also add a hardcoded field.

### `add_tag`

Value type is [array](#)

Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
 yaml {
 add_tag => ["foo_%{somefield}"]
 }
 # You can also add multiple tags at once:
 filter {
 yaml {
 add_tag => ["foo_%{somefield}", "taggedy_tag"]
 }
 }
}
```

}

If the event has field "somefield" == "hello" this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

``` output { stdout { id => "ABC" } } ```

If you don't explicitly set this variable Logstash will generate a unique name.

`periodic_flush`

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

`remove_field`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
  yaml {
    remove_field => [ "foo_#{@somefield}" ]
  }
}
```

Logstash 5.2 Configuration Guide

```
# You can also remove multiple fields at once:  
filter {  
  yaml {  
    remove_field => [ "foo_{somefield}", "my_extraneous_field" ]  
  }  
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the field with name foo_hello if it is present. The second example would remove an additional, non-dynamic field.

[remove_tag](#)

Value type is [array](#)

Default value is []

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the %{field} syntax.

Example:

```
filter {  
  yaml {  
    remove_tag => [ "foo_{somefield}" ]  
  }  
}  
# You can also remove multiple tags at once:  
filter {  
  yaml {  
    remove_tag => [ "foo_{somefield}", "sad_unwanted_tag" ]  
  }  
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag foo_hello if it is present. The second example would remove a sad, unwanted tag as well.

[source](#)

This is a required setting.

Value type is [string](#)

There is no default value for this setting.

The configuration for the YAML filter:

```
source => source_field
```

For example, if you have YAML data in the @message field:

```
filter {
  yaml {
    source => "message"
  }
}
```

The above would parse the yaml from the @message field

target

Value type is [string](#)

There is no default value for this setting.

Define the target field for placing the parsed data. If this setting is omitted, the YAML data will be stored at the root (top level) of the event.

For example, if you want the data to be put in the `doc` field:

```
filter {
  yaml {
    target => "doc"
  }
}
```

YAML in the value of the `source` field will be expanded into a data structure in the `target` field.

NOTE: if the `target` field already exists, it will be overwritten!

zeromq

This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-filter-zeromq`.

ZeroMQ filter. This is the best way to send an event externally for filtering. It works much like an exec filter would by sending the event "offsite" for processing and waiting for a response.

The protocol here is: * REQ sent with JSON-serialized logstash event * REP read expected to be the full JSON *filtered* event * - if reply read is an empty string, it will cancel the event.

Note that this is a limited subset of the zeromq functionality in inputs and outputs. The only topology that makes sense here is: REQ/REP. bunde

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
zeromq {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
add_field	hash	No	{ }
add_tag	array	No	[]
add_tag_on_timeout	string	No	"zeromqtimeout"
address	string	No	"tcp://127.0.0.1:2121"
field	string	No	
mode	string , one of ["server", "client"]	No	"client"
periodic_flush	boolean	No	false
remove_field	array	No	[]

Setting	Input type	Required	Default value
remove_tag	array	No	[]
retries	number	No	3
sockopt	hash	No	
timeout	number	No	500

Details

[add_field](#)

Value type is [hash](#)

Default value is {}

If this filter is successful, add any arbitrary fields to this event. Field names can be dynamic and include parts of the event using the %{field}.

Example:

```
filter {
    zeromq {
        add_field => { "foo_%{somefield}" => "Hello world, from %{host}" }
    }
    # You can also add multiple fields at once:
    filter {
        zeromq {
            add_field => {
                "foo_%{somefield}" => "Hello world, from %{host}"
                "new_field" => "new_static_value"
            }
        }
    }
}
```

If the event has field "somefield" == "hello" this filter, on success, would add field foo_hello if it is present, with the value above and the %{host} piece replaced with that value from the event. The second example would also add a hardcoded field.

[add_tag](#)

Value type is [array](#)

Default value is []

If this filter is successful, add arbitrary tags to the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
    zeromq {
        add_tag => [ "foo_%{somefield}" ]
    }
}
# You can also add multiple tags at once:
filter {
    zeromq {
        add_tag => [ "foo_%{somefield}", "taggedy_tag" ]
    }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would add a tag `foo_hello` (and the second example would of course add a `taggedy_tag` tag).

`add_tag_on_timeout`

Value type is [string](#)

Default value is `"zeromqtimeout"`

tag to add if zeromq timeout expires before getting back an answer. If set to `""` then no tag will be added.

`address`

Value type is [string](#)

Default value is `"tcp://127.0.0.1:2121"`

0mq socket address to connect or bind Please note that `inproc://` will not work with logstash as we use a context per thread By default, filters connect

`field`

Value type is [string](#)

There is no default value for this setting.

The field to send off-site for processing If this is unset, the whole event will be sent

`mode`

Value can be any of: `server, client`

Default value is "client"

0mq mode server mode binds/listens client mode connects

`periodic_flush`

Value type is [boolean](#)

Default value is `false`

Call the filter flush method at regular interval. Optional.

`remove_field`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary fields from this event. Example:

```
filter {
    zeromq {
        remove_field => [ "foo_{somefield}" ]
    }
}
# You can also remove multiple fields at once:
filter {
    zeromq {
        remove_field => [ "foo_{somefield}", "my_extraneous_field" ]
    }
}
```

If the event has field `"somefield" == "hello"` this filter, on success, would remove the field with name `foo_hello` if it is present. The second example would remove an additional, non-dynamic field.

`remove_tag`

Value type is [array](#)

Default value is `[]`

If this filter is successful, remove arbitrary tags from the event. Tags can be dynamic and include parts of the event using the `%{field}` syntax.

Example:

```
filter {
    zeromq {
```

Logstash 5.2 Configuration Guide

```
        remove_tag => [ "foo_%{somefield}" ]
    }
}
# You can also remove multiple tags at once:
filter {
    zeromq {
        remove_tag => [ "foo_%{somefield}", "sad_unwanted_tag" ]
    }
}
```

If the event has field "somefield" == "hello" this filter, on success, would remove the tag `foo_hello` if it is present. The second example would remove a sad, unwanted tag as well.

retries

Value type is [number](#)

Default value is 3

number of retries, used for both sending and receiving messages. for sending, retries should return instantly. for receiving, the total blocking time is up to retries X timeout, which by default is 3 X 500 = 1500ms

sockopt

Value type is [hash](#)

There is no default value for this setting.

0mq socket options This exposes `zmq_setsockopt` for advanced tuning see <http://api.zeromq.org/2-1:zmq-setsockopt> for details

This is where you would set values like: ZMQ::HWM - high water mark ZMQ::IDENTITY - named queues ZMQ::SWAP_SIZE - space for disk overflow ZMQ::SUBSCRIBE - topic filters for pubsub

example: `sockopt => ["ZMQ::HWM", 50, "ZMQ::IDENTITY", "my_named_queue"]`

timeout

Value type is [number](#)

Default value is 500

timeout in milliseconds on which to wait for a reply.

Codec plugins

A codec plugin changes the data representation of an event. Codecs are essentially stream filters that can operate as part of an input or output.

The following codec plugins are available below. For a list of Elastic supported plugins, please consult the [Support Matrix](#).

Plugin	Description	Github repository
avro	Reads serialized Avro records as Logstash events	logstash-codec-avro
cef	Reads the ArcSight Common Event Format (CEF).	logstash-codec-cef
cloudfront	Reads AWS CloudFront reports	logstash-codec-cloudfront
cloudtrail	Reads AWS Cloudtrail events	logstash-codec-cloudtrail
collectd	Reads events from the <code>collectd</code> binary protocol using UDP.	logstash-codec-collectd
compress_spooler	Compresses events into spooled batches	logstash-codec-compress_spooler
dots	Sends 1 dot per event to <code>stdout</code> for performance tracking	logstash-codec-dots
edn	Reads EDN format data	logstash-codec-edn
edn_lines	Reads newline-delimited EDN format data	logstash-codec-edn_lines
es_bulk	Reads the Elasticsearch bulk format into separate events, along with metadata	logstash-codec-es_bulk
fluent	Reads the <code>fluentd</code> msgpack schema	logstash-codec-fluent
graphite	Reads graphite formatted lines	logstash-codec-graphite
gzip_lines	Reads <code>gzip</code> encoded content	logstash-codec-gzip_lines
json	Reads JSON formatted content, creating one event per element in a JSON array	logstash-codec-json
json_lines	Reads newline-delimited JSON	logstash-codec-json_lines
line	Reads line-oriented text data	logstash-codec-line
msgpack		logstash-codec-msgpack
multiline	Merges multiline messages into a single event	logstash-codec-multiline
netflow	Reads Netflow v5 and Netflow v9 data	logstash-codec-netflow
nmap	Reads Nmap data in XML format	logstash-codec-nmap
oldlogstashjson	Reads Logstash JSON in the schema used by Logstash versions earlier than 1.2.0	logstash-codec-oldlogstashjson

plain	Reads plaintext with no delimiting between events	logstash-codec-plain
protobuf	Converts protobuf encoded messages into logstash events	logstash-codec-protobuf
rubydebug	Applies the Ruby Awesome Print library to Logstash events	logstash-codec-rubydebug
s3_plain	Provides backwards compatibility with earlier versions of S3 Output	logstash-codec-s3_plain
sflow	Decodes sflow v5 flows	logstash-codec-sflow

[« zeromq](#) [avro »](#)

avro

Version: 3.0.0

Released on: 2016-09-08

[Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
bin/logstash-plugin install logstash-codec-avro.

Read serialized Avro records as Logstash events

This plugin is used to serialize Logstash events as Avro datums, as well as deserializing Avro datums into Logstash events.

Encoding

This codec is for serializing individual Logstash events as Avro datums that are Avro binary blobs. It does not encode Logstash events into an Avro file.

Decoding

This codec is for deserializing individual Avro records. It is not for reading Avro files. Avro files have a unique format that must be handled upon input.

Usage

Example usage with Kafka input.

```
input {
  kafka {
    codec => avro {
      schema_uri => "/tmp/schema.avsc"
    }
  }
  filter {
    ...
  }
  output {
    ...
  }
}
```

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
avro {  
    schema_uri => ...  
}
```

Available configuration options:

Setting	Input type	Required	Default value
enable_metric	boolean	No	true
id	string	No	
schema_uri	string	Yes	

Details

enable_metric

Value type is [boolean](#)

Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

id

Value type is [string](#)

There is no default value for this setting.

Add a unique ID to the plugin instance, this ID is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

*schema\_uri*

This is a required setting.

Value type is [string](#)

There is no default value for this setting.

schema path to fetch the schema from. This can be a *http* or *file* scheme URI example:

**http** - `http://example.com/schema.avsc`

**file** - `/path/to/schema.avsc`

## cef

Version: 4.1.2

Released on: 2016-12-26

[Changelog](#)

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running  
bin/logstash-plugin install logstash-codec-cef.

Implementation of a Logstash codec for the ArcSight Common Event Format (CEF) Based on Revision 20 of Implementing ArcSight CEF, dated from June 05, 2013

<https://protect724.hp.com/servlet/JiveServlet/downloadBody/1072-102-6-4697/CommonEventFormat.pdf>

If this codec receives a payload from an input that is not a valid CEF message, then it will produce an event with the *message* field and a *\_cefparsfailure* tag.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
cef {
 }
}
```

Available configuration options:

Setting	Input type	Required	Default value
delimiter	<a href="#">string</a>	No	
enable_metric	<a href="#">boolean</a>	No	true
fields	<a href="#">array</a>	No	[]
id	<a href="#">string</a>	No	
name	<a href="#">string</a>	No	"Logstash"
product	<a href="#">string</a>	No	"Logstash"
severity	<a href="#">string</a>	No	"6"

Setting	Input type	Required	Default value
signature	<a href="#">string</a>	No	"Logstash"
vendor	<a href="#">string</a>	No	"Elasticsearch"
version	<a href="#">string</a>	No	"1.0"

## Details

*delimiter*

Value type is [string](#)

There is no default value for this setting.

If your input puts a delimiter between each CEF event, you'll want to set this to be that delimiter.

For example, with the TCP input, you probably want to put this:

```
input {
 tcp {
 codec => cef { delimiter => "\r\n" }
 # ...
 }
}
```

This setting allows the following character sequences to have special meaning:

\r (backslash "r") - means carriage return (ASCII 0x0D)

\n (backslash "n") - means newline (ASCII 0x0A)

*deprecated\_v1\_fields (DEPRECATED)*

DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.

Value type is [boolean](#)

There is no default value for this setting.

Set this flag if you want to have both v1 and v2 fields indexed at the same time. Note that this option will increase the index size and data stored in outputs like Elasticsearch. This option is available to ease transition to new schema

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`fields`

Value type is [array](#)

Default value is `[]`

Fields to be included in CEF extension part as key/value pairs

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
name
```

Value type is [string](#)

Default value is `"Logstash"`

Name field in CEF header. The new value can include `%{foo}` strings to help you build a new value from other parts of the event.

`product`

Value type is [string](#)

Default value is `"Logstash"`

Device product field in CEF header. The new value can include `%{foo}` strings to help you build a new value from other parts of the event.

*sev* (*DEPRECATED*)

DEPRECATED WARNING: This configuration item is deprecated and may not be available in future versions.

Value type is [string](#)

There is no default value for this setting.

Deprecated severity field for CEF header. The new value can include `%{foo}` strings to help you build a new value from other parts of the event.

This field is used only if :severity is unchanged set to the default value.

Defined as field of type string to allow sprintf. The value will be validated to be an integer in the range from 0 to 10 (including). All invalid values will be mapped to the default of 6.

*severity*

Value type is [string](#)

Default value is "6"

Severity field in CEF header. The new value can include `%{foo}` strings to help you build a new value from other parts of the event.

Defined as field of type string to allow sprintf. The value will be validated to be an integer in the range from 0 to 10 (including). All invalid values will be mapped to the default of 6.

*signature*

Value type is [string](#)

Default value is "Logstash"

Signature ID field in CEF header. The new value can include `%{foo}` strings to help you build a new value from other parts of the event.

*vendor*

Value type is [string](#)

Default value is "Elasticsearch"

Device vendor field in CEF header. The new value can include `%{foo}` strings to help you build a new value from other parts of the event.

`version`

Value type is [string](#)

Default value is "1.0"

Device version field in CEF header. The new value can include `%{foo}` strings to help you build a new value from other parts of the event.

## cloudfront

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-codec-cloudfront`.

This codec will read cloudfront encoded content

### Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
cloudfront {
}
```

Available configuration options:

Setting	Input type	Required	Default value
charset	<code>string</code> , one of ["ASCII-8BIT", "Big5", "Big5-HKSCS", "Big5- UAO", "CP949", "Emacs-Mule", "EUC-JP", "EUC-KR", "EUC- TW", "GB18030", "GBK", "ISO-8859-1", "ISO-8859-2", "ISO-8859-3", "ISO-8859-4", "ISO-8859-5", "ISO-8859- 6", "ISO-8859-7", "ISO-8859-8", "ISO-8859-9", "ISO- 8859-10", "ISO-8859-11", "ISO-8859-13", "ISO-8859-14", "ISO-8859-15", "ISO-8859-16", "KOI8-R", "KOI8-U", "Shift_JIS", "US-ASCII", "UTF-8", "UTF-16BE", "UTF- 16LE", "UTF-32BE", "UTF-32LE", "Windows-1251", "GB2312", "Windows-31J", "IBM437", "IBM737", "IBM775", "CP850", "IBM852", "CP852", "IBM855", "CP855", "IBM857", "IBM860", "IBM861", "IBM862", "IBM863", "IBM864", "IBM865", "IBM866", "IBM869", "Windows- 1258", "GB1988", "macCentEuro", "macCroatian", "macCyrillic", "macGreek", "macIceland", "macRoman", "macRomania", "macThai", "macTurkish", "macUkraine", "CP950", "CP951", "stateless-ISO-2022-JP", "eucJP-ms", "CP51932", "EUC-JIS-2004", "GB12345", "ISO-2022-JP", "ISO-2022-JP-2", "CP50220", "CP50221", "Windows-1252", "Windows-1250", "Windows-1256", "Windows-1253", "Windows-1255", "Windows-1254", "TIS-620", "Windows- 874", "Windows-1257", "MacJapanese", "UTF-7", "UTF8- MAC", "UTF-16", "UTF-32", "UTF8-DoCoMo", "SJIS- DoCoMo", "UTF8-KDDI", "SJIS-KDDI", "ISO-2022-JP-KDDI", "stateless-ISO-2022-JP-KDDI", "UTF8-SoftBank", "SJIS- SoftBank", "BINARY", "CP437", "CP737", "CP775", "IBM850", "CP857", "CP860", "CP861", "CP862", "CP863",	No	"UTF-8"

Setting	Input type	Required	Default value
	"CP864", "CP865", "CP866", "CP869", "CP1258", "Big5-HKSCS:2008", "eucJP", "euc-jp-ms", "EUC-JISX0213", "eucKR", "euctW", "EUC-CN", "eucCN", "CP936", "ISO2022-JP", "ISO2022-JP2", "ISO8859-1", "CP1252", "ISO8859-2", "CP1250", "ISO8859-3", "ISO8859-4", "ISO8859-5", "ISO8859-6", "CP1256", "ISO8859-7", "CP1253", "ISO8859-8", "CP1255", "ISO8859-9", "CP1254", "ISO8859-10", "ISO8859-11", "CP874", "ISO8859-13", "CP1257", "ISO8859-14", "ISO8859-15", "ISO8859-16", "CP878", "MacJapan", "ASCII", "ANSI_X3.4-1968", "646", "CP65000", "CP65001", "UTF-8-MAC", "UTF-8-HFS", "UCS-2BE", "UCS-4BE", "UCS-4LE", "CP932", "csWindows31J", "SJIS", "PCK", "CP1251", "external", "locale"]		

## Details

### charset

Value can be any of: ASCII-8BIT, Big5, Big5-HKSCS, Big5-UAO, CP949, Emacs-Mule, EUC-JP, EUC-KR, EUC-TW, GB18030, GBK, ISO-8859-1, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9, ISO-8859-10, ISO-8859-11, ISO-8859-13, ISO-8859-14, ISO-8859-15, ISO-8859-16, KOI8-R, KOI8-U, Shift\_JIS, US-ASCII, UTF-8, UTF-16BE, UTF-16LE, UTF-32BE, UTF-32LE, Windows-1251, GB2312, Windows-31J, IBM437, IBM737, IBM775, CP850, IBM852, CP852, IBM855, CP855, IBM857, IBM860, IBM861, IBM862, IBM863, IBM864, IBM865, IBM866, IBM869, Windows-1258, GB1988, macCentEuro, macCroatian, macCyrillic, macGreek, macIceland, macRoman, macRomania, macThai, macTurkish, macUkraine, CP950, CP951, stateless-ISO-2022-JP, eucJP-ms, CP51932, EUC-JIS-2004, GB12345, ISO-2022-JP, ISO-2022-JP-2, CP50220, CP50221, Windows-1252, Windows-1250, Windows-1256, Windows-1253, Windows-1255, Windows-1254, TIS-620, Windows-874, Windows-1257, MacJapanese, UTF-7, UTF8-MAC, UTF-16, UTF-32, UTF8-DoCoMo, SJIS-DoCoMo, UTF8-KDDI, SJIS-KDDI, ISO-2022-JP-KDDI, stateless-ISO-2022-JP-KDDI, UTF8-SoftBank, SJIS-SoftBank, BINARY, CP437, CP737, CP775, IBM850, CP857, CP860, CP861, CP862, CP863, CP864, CP865, CP866, CP869, CP1258, Big5-HKSCS:2008, eucJP, euc-jp-ms, EUC-JISX0213, eucKR, euctW, EUC-CN, eucCN, CP936, ISO2022-JP, ISO2022-JP2, ISO8859-1, CP1252, ISO8859-2, CP1250, ISO8859-3, ISO8859-4, ISO8859-5, ISO8859-6, CP1256, ISO8859-7, CP1253, ISO8859-8, CP1255, ISO8859-9, CP1254, ISO8859-10, ISO8859-11, CP874, ISO8859-13, CP1257, ISO8859-14, ISO8859-15, ISO8859-16, CP878, MacJapan, ASCII, ANSI\_X3.4-1968, 646, CP65000, CP65001, UTF-8-MAC, UTF-8-HFS, UCS-2BE, UCS-4BE, UCS-4LE, CP932, csWindows31J, SJIS, PCK, CP1251, external, locale

Default value is "UTF-8"

The character encoding used in this codec. Examples include "UTF-8" and "CP1252"

JSON requires valid UTF-8 strings, but in some cases, software that emits JSON does so in another encoding (nxlog, for example). In weird cases like this, you can set the charset setting to the actual encoding of the text and logstash will convert it for you.

For nxlog users, you'll want to set this to "CP1252"

## cloudtrail

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-codec-cloudtrail`.

### Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
cloudtrail {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>spool_size</code>	<a href="#">number</a>	No	50

### Details

`spool_size`

Value type is [number](#)

Default value is 50

## collectd

Version: 3.0.3

Released on: 2016-12-26

[Changelog](#)

encoding utf-8

### Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
collectd {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">authfile</a>	<a href="#">string</a>	No	
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">nan_handling</a>	<a href="#">string</a> , one of ["change_value", "warn", "drop"]	No	"change_value"
<a href="#">nan_tag</a>	<a href="#">string</a>	No	"_collectdNaN"
<a href="#">nan_value</a>	<a href="#">number</a>	No	0
<a href="#">prune_intervals</a>	<a href="#">boolean</a>	No	true
<a href="#">security_level</a>	<a href="#">string</a> , one of ["None", "Sign", "Encrypt"]	No	"None"
<a href="#">typesdb</a>	<a href="#">array</a>	No	

### Details

*authfile*

Value type is [string](#)

There is no default value for this setting.

Path to the authentication file. This file should have the same format as the [AuthFile](#) in collectd. You only need to set this option if the `security_level` is set to `Sign` or `Encrypt`

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

`nan_handling`

Value can be any of: `change_value`, `warn`, `drop`

Default value is "`change_value`"

What to do when a value in the event is `NaN` (Not a Number)

`change_value` (default): Change the `NaN` to the value of the `nan_value` option and add `nan_tag` as a tag

`warn`: Change the `NaN` to the value of the `nan_value` option, print a warning to the log and add `nan_tag` as a tag

`drop`: Drop the event containing the `NaN` (this only drops the single event, not the whole packet)

*nan\_tag*

Value type is [string](#)

Default value is "\_collectdNaN"

The tag to add to the event if a 'NaN' value was found Set this to an empty string ('') if you don't want to tag

*nan\_value*

Value type is [number](#)

Default value is 0

Only relevant when `nan_handeling` is set to `change_value` Change NaN to this configured value

*prune\_intervals*

Value type is [boolean](#)

Default value is `true`

Prune interval records. Defaults to `true`.

*security\_level*

Value can be any of: `None`, `Sign`, `Encrypt`

Default value is "`None`"

Security Level. Default is `None`. This setting mirrors the setting from the collectd [Network plugin](#)

*typesdb*

Value type is [array](#)

There is no default value for this setting.

File path(s) to collectd `types.db` to use. The last matching pattern wins if you have identical pattern names in multiple files. If no `types.db` is provided the included `types.db` will be used (currently 5.4.0).

## compress\_spooler

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-codec-compress_spooler`.

### Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
compress_spooler {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">compress_level</a>	<a href="#">number</a>	No	6
<a href="#">spool_size</a>	<a href="#">number</a>	No	50

### Details

[compress\\_level](#)

Value type is [number](#)

Default value is 6

[spool\\_size](#)

Value type is [number](#)

Default value is 50

## dots

Version: 3.0.2

Released on: 2016-07-14

[Changelog](#)

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
dots {
 }
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	

## Details

[enable\\_metric](#)

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

[id](#)

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

## edn

Version: 3.0.2

Released on: 2016-07-14

[Changelog](#)

### Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
edn {
 }
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	

### Details

[enable\\_metric](#)

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

[id](#)

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

# edn\_lines

Version: 3.0.2

Released on: 2016-07-14

[Changelog](#)

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
edn_lines {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	

## Details

[enable\\_metric](#)

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

[id](#)

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

## es\_bulk

Version: 3.0.3

Released on: 2016-11-17

[Changelog](#)

This codec will decode the [Elasticsearch bulk format](#) into individual events, plus metadata into the @metadata field.

Encoding is not supported at this time as the Elasticsearch output submits Logstash events in bulk format.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
es_bulk {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	

## Details

[enable\\_metric](#)

Value type is [boolean](#)

Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

*id*

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

## fluent

Version: 3.0.2

Released on: 2016-07-14

[Changelog](#)

This codec handles fluentd's msgpack schema.

For example, you can receive logs from `fluent-logger-ruby` with:

```
input {
 tcp {
 codec => fluent
 port => 4000
 }
}
```

And from your ruby code in your own application:

```
logger = Fluent::Logger::FluentLogger.new(nil, :host => "example.log", :port => 4000)
logger.post("some_tag", { "your" => "data", "here" => "yay!" })
```

Notes:

the fluent uses a second-precision time for events, so you will never see subsecond precision on events processed by this codec.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
fluent { }
```

Available configuration options:

Setting	Input type	Required	Default value
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	

## Details

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

# graphite

Version: 3.0.2

Released on: 2016-07-14

[Changelog](#)

This codec will encode and decode Graphite formated lines.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
graphite {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">exclude_metrics</a>	<a href="#">array</a>	No	[ "%{[^}]+}" ]
<a href="#">fields_are_metrics</a>	<a href="#">boolean</a>	No	false
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">include_metrics</a>	<a href="#">array</a>	No	[ ".*" ]
<a href="#">metrics</a>	<a href="#">hash</a>	No	{ }
<a href="#">metrics_format</a>	<a href="#">string</a>	No	"*"

## Details

[enable\\_metric](#)

Value type is [boolean](#)

Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

*exclude\_metrics*

Value type is [array](#)

Default value is `["%{[^}]+"}"]`

*fields\_are\_metrics*

Value type is [boolean](#)

Default value is `false`

Indicate that the event @fields should be treated as metrics and will be sent as is to graphite

*id*

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
include_metrics
```

Value type is [array](#)

Default value is `".*"`

Include only regex matched metric names

*metrics*

Value type is [hash](#)

Default value is `{}`

The metric(s) to use. This supports dynamic strings like `%{host}` for metric names and also for values. This is a hash field with key of the metric name, value of the metric value. Example:

```
["%{host}/uptime", "%{uptime_1m}"]
```

The value will be coerced to a floating point value. Values which cannot be coerced will zero (0)

*metrics\_format*

Value type is [string](#)

Default value is "\*"

Defines format of the metric string. The placeholder `*` will be replaced with the name of the actual metric.

```
metrics_format => "foo.bar.*.sum"
```

NOTE:If no metrics\_format is defined the name of the metric will be used as fallback.

## gzip\_lines

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-codec-gzip_lines`.

This codec will read gzip encoded content

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
gzip_lines {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">charset</a>	<code>string</code> , one of ["ASCII-8BIT", "Big5", "Big5-HKSCS", "Big5- UAO", "CP949", "Emacs-Mule", "EUC-JP", "EUC-KR", "EUC-TW", "GB18030", "GBK", "ISO-8859-1", "ISO-8859-2", "ISO-8859-3", "ISO-8859-4", "ISO-8859-5", "ISO-8859- 6", "ISO-8859-7", "ISO-8859-8", "ISO-8859-9", "ISO- 8859-10", "ISO-8859-11", "ISO-8859-13", "ISO-8859-14", "ISO-8859-15", "ISO-8859-16", "KOI8-R", "KOI8-U", "Shift_JIS", "US-ASCII", "UTF-8", "UTF-16BE", "UTF- 16LE", "UTF-32BE", "UTF-32LE", "Windows-1251", "GB2312", "Windows-31J", "IBM437", "IBM737", "IBM775", "CP850", "IBM852", "CP852", "IBM855", "CP855", "IBM857", "IBM860", "IBM861", "IBM862", "IBM863", "IBM864", "IBM865", "IBM866", "IBM869", "Windows- 1258", "GB1988", "macCentEuro", "macCroatian", "macCyrillic", "macGreek", "macIceland", "macRoman", "macRomania", "macThai", "macTurkish", "macUkraine", "CP950", "CP951", "stateless-ISO-2022-JP", "eucJP-ms", "CP51932", "EUC-JIS-2004", "GB12345", "ISO-2022-JP", "ISO-2022-JP-2", "CP50220", "CP50221", "Windows-1252", "Windows-1250", "Windows-1256", "Windows-1253", "Windows-1255", "Windows-1254", "TIS-620", "Windows- 874", "Windows-1257", "MacJapanese", "UTF-7", "UTF8- MAC", "UTF-16", "UTF-32", "UTF8-DoCoMo", "SJIS- DoCoMo", "UTF8-KDDI", "SJIS-KDDI", "ISO-2022-JP-KDDI", "stateless-ISO-2022-JP-KDDI", "UTF8-SoftBank", "SJIS- SoftBank", "BINARY", "CP437", "CP737", "CP775", "IBM850", "CP857", "CP860", "CP861", "CP862", "CP863",	No	"UTF-8"

Setting	Input type	Required	Default value
	"CP864", "CP865", "CP866", "CP869", "CP1258", "Big5-HKSCS:2008", "eucJP", "euc-jp-ms", "EUC-JISX0213", "eucKR", "eucTW", "EUC-CN", "eucCN", "CP936", "ISO2022-JP", "ISO2022-JP2", "ISO8859-1", "CP1252", "ISO8859-2", "CP1250", "ISO8859-3", "ISO8859-4", "ISO8859-5", "ISO8859-6", "CP1256", "ISO8859-7", "CP1253", "ISO8859-8", "CP1255", "ISO8859-9", "CP1254", "ISO8859-10", "ISO8859-11", "CP874", "ISO8859-13", "CP1257", "ISO8859-14", "ISO8859-15", "ISO8859-16", "CP878", "MacJapan", "ASCII", "ANSI_X3.4-1968", "646", "CP65000", "CP65001", "UTF-8-MAC", "UTF-8-HFS", "UCS-2BE", "UCS-4BE", "UCS-4LE", "CP932", "csWindows31J", "SJIS", "PCK", "CP1251", "external", "locale"]		

## Details

### charset

Value can be any of: ASCII-8BIT, Big5, Big5-HKSCS, Big5-UAO, CP949, Emacs-Mule, EUC-JP, EUC-KR, EUC-TW, GB18030, GBK, ISO-8859-1, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9, ISO-8859-10, ISO-8859-11, ISO-8859-13, ISO-8859-14, ISO-8859-15, ISO-8859-16, KOI8-R, KOI8-U, Shift\_JIS, US-ASCII, UTF-8, UTF-16BE, UTF-16LE, UTF-32BE, UTF-32LE, Windows-1251, GB2312, Windows-31J, IBM437, IBM737, IBM775, CP850, IBM852, CP852, IBM855, CP855, IBM857, IBM860, IBM861, IBM862, IBM863, IBM864, IBM865, IBM866, IBM869, Windows-1258, GB1988, macCentEuro, macCroatian, macCyrillic, macGreek, macIceland, macRoman, macRomania, macThai, macTurkish, macUkraine, CP950, CP951, stateless-ISO-2022-JP, eucJP-ms, CP51932, EUC-JIS-2004, GB12345, ISO-2022-JP, ISO-2022-JP-2, CP50220, CP50221, Windows-1252, Windows-1250, Windows-1256, Windows-1253, Windows-1255, Windows-1254, TIS-620, Windows-874, Windows-1257, MacJapanese, UTF-7, UTF8-MAC, UTF-16, UTF-32, UTF8-DoCoMo, SJIS-DoCoMo, UTF8-KDDI, SJIS-KDDI, ISO-2022-JP-KDDI, stateless-ISO-2022-JP-KDDI, UTF8-SoftBank, SJIS-SoftBank, BINARY, CP437, CP737, CP775, IBM850, CP857, CP860, CP861, CP862, CP863, CP864, CP865, CP866, CP869, CP1258, Big5-HKSCS:2008, eucJP, euc-jp-ms, EUC-JISX0213, eucKR, eucTW, EUC-CN, eucCN, CP936, ISO2022-JP, ISO2022-JP2, ISO8859-1, CP1252, ISO8859-2, CP1250, ISO8859-3, ISO8859-4, ISO8859-5, ISO8859-6, CP1256, ISO8859-7, CP1253, ISO8859-8, CP1255, ISO8859-9, CP1254, ISO8859-10, ISO8859-11, CP874, ISO8859-13, CP1257, ISO8859-14, ISO8859-15, ISO8859-16, CP878, MacJapan, ASCII, ANSI\_X3.4-1968, 646, CP65000, CP65001, UTF-8-MAC, UTF-8-HFS, UCS-2BE, UCS-4BE, UCS-4LE, CP932, csWindows31J, SJIS, PCK, CP1251, external, locale

Default value is "UTF-8"

The character encoding used in this codec. Examples include "UTF-8" and "CP1252"

JSON requires valid UTF-8 strings, but in some cases, software that emits JSON does so in another encoding (nxlog, for example). In weird cases like this, you can set the charset setting to the actual encoding of the text and logstash will convert it for you.

For nxlog users, you'll want to set this to "CP1252"

# json

Version: 3.0.2

Released on: 2016-07-14

[Changelog](#)

This codec may be used to decode (via inputs) and encode (via outputs) full JSON messages. If the data being sent is a JSON array at its root multiple events will be created (one per element).

If you are streaming JSON messages delimited by `\n` then see the `json_lines` codec.

Encoding will result in a compact JSON representation (no line terminators or indentation)

If this codec receives a payload from an input that is not valid JSON, then it will fall back to plain text and add a tag `_jsonparsefailure`. Upon a JSON failure, the payload will be stored in the `message` field.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
json {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">charset</a>	<code>string</code> , one of ["ASCII-8BIT", "UTF-8", "US-ASCII", "Big5", "Big5-HKSCS", "Big5-UAO", "CP949", "Emacs-Mule", "EUC-JP", "EUC-KR", "EUC-TW", "GB2312", "GB18030", "GBK", "ISO-8859-1", "ISO-8859-2", "ISO-8859-3", "ISO-8859-4", "ISO-8859-5", "ISO-8859-6", "ISO-8859-7", "ISO-8859-8", "ISO-8859-9", "ISO-8859-10", "ISO-8859-11", "ISO-8859-13", "ISO-8859-14", "ISO-8859-15", "ISO-8859-16", "KOI8-R", "KOI8-U", "Shift_JIS", "UTF-16BE", "UTF-16LE", "UTF-32BE", "UTF-32LE", "Windows-31J", "Windows-1250", "Windows-1251", "Windows-1252", "IBM437", "IBM737", "IBM775", "CP850", "IBM852", "CP852", "IBM855", "CP855"]	No	"UTF-8"

Setting	Input type	Required	Default value
	"IBM857", "IBM860", "IBM861", "IBM862", "IBM863", "IBM864", "IBM865", "IBM866", "IBM869", "Windows-1258", "GB1988", "macCentEuro", "macCroatian", "macCyrillic", "macGreek", "macIceland", "macRoman", "macRomania", "macThai", "macTurkish", "macUkraine", "CP950", "CP951", "IBM037", "stateless-ISO-2022-JP", "eucJP-ms", "CP51932", "EUC-JIS-2004", "GB12345", "ISO-2022-JP", "ISO-2022-JP-2", "CP50220", "CP50221", "Windows-1256", "Windows-1253", "Windows-1255", "Windows-1254", "TIS-620", "Windows-874", "Windows-1257", "MacJapanese", "UTF-7", "UTF8-MAC", "UTF-16", "UTF-32", "UTF8-DoCoMo", "SJIS-DoCoMo", "UTF8-KDDI", "SJIS-KDDI", "ISO-2022-JP-KDDI", "stateless-ISO-2022-JP-KDDI", "UTF8-SoftBank", "SJIS-SoftBank", "BINARY", "CP437", "CP737", "CP775", "IBM850", "CP857", "CP860", "CP861", "CP862", "CP863", "CP864", "CP865", "CP866", "CP869", "CP1258", "Big5-HKSCS:2008", "ebcdic-cp-us", "eucJP", "euc-jp-ms", "EUC-JISX0213", "eucKR", "eucTW", "EUC-CN", "eucCN", "CP936", "ISO2022-JP", "ISO2022-JP2", "ISO8859-1", "ISO8859-2", "ISO8859-3", "ISO8859-4", "ISO8859-5", "ISO8859-6", "CP1256", "ISO8859-7", "CP1253", "ISO8859-8", "CP1255", "ISO8859-9", "CP1254", "ISO8859-10", "ISO8859-11", "CP874", "ISO8859-13", "CP1257", "ISO8859-14", "ISO8859-15", "ISO8859-16", "CP878", "MacJapan", "ASCII", "ANSI_X3.4-1968", "646", "CP65000", "CP65001", "UTF-8-MAC", "UTF-8-HFS", "UCS-2BE", "UCS-4BE", "UCS-4LE", "CP932", "csWindows31J", "SJIS", "PCK", "CP1250", "CP1251", "CP1252", "external", "locale"]		
<a href="#"><u>enable_metric</u></a>	<a href="#"><u>boolean</u></a>	No	true
<a href="#"><u>id</u></a>	<a href="#"><u>string</u></a>	No	

## Details

[charset](#)

Value can be any of: ASCII-8BIT, UTF-8, US-ASCII, Big5, Big5-HKSCS, Big5-UAO, CP949, Emacs-Mule, EUC-JP, EUC-KR, EUC-TW, GB2312, GB18030, GBK, ISO-8859-1, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9, ISO-8859-10, ISO-8859-11, ISO-8859-13, ISO-8859-14, ISO-8859-15, ISO-8859-16, KOI8-R, KOI8-U, Shift\_JIS, UTF-16BE, UTF-16LE, UTF-32BE, UTF-32LE,

Windows-31J, Windows-1250, Windows-1251, Windows-1252, IBM437, IBM737, IBM775, CP850, IBM852, CP852, IBM855, CP855, IBM857, IBM860, IBM861, IBM862, IBM863, IBM864, IBM865, IBM866, IBM869, Windows-1258, GB1988, macCentEuro, macCroatian, macCyrillic, macGreek, macIceland, macRoman, macRomania, macThai, macTurkish, macUkraine, CP950, CP951, IBM037, stateless-ISO-2022-JP, eucJP-ms, CP51932, EUC-JIS-2004, GB12345, ISO-2022-JP, ISO-2022-JP-2, CP50220, CP50221, Windows-1256, Windows-1253, Windows-1255, Windows-1254, TIS-620, Windows-874, Windows-1257, MacJapanese, UTF-7, UTF8-MAC, UTF-16, UTF-32, UTF8-DoCoMo, SJIS-DoCoMo, UTF8-KDDI, SJIS-KDDI, ISO-2022-JP-KDDI, stateless-ISO-2022-JP-KDDI, UTF8-SoftBank, SJIS-SoftBank, BINARY, CP437, CP737, CP775, IBM850, CP857, CP860, CP861, CP862, CP863, CP864, CP865, CP866, CP869, CP1258, Big5-HKSCS:2008, ebcdic-cp-us, eucJP, euc-jp-ms, EUC-JISX0213, eucKR, eucTW, EUC-CN, eucCN, CP936, ISO2022-JP, ISO2022-JP2, ISO8859-1, ISO8859-2, ISO8859-3, ISO8859-4, ISO8859-5, ISO8859-6, ISO8859-7, CP1253, ISO8859-8, CP1255, ISO8859-9, CP1254, ISO8859-10, ISO8859-11, CP874, ISO8859-13, CP1257, ISO8859-14, ISO8859-15, ISO8859-16, CP878, MacJapan, ASCII, ANSI\_X3.4-1968, 646, CP65000, CP65001, UTF-8-MAC, UTF-8-HFS, UCS-2BE, UCS-4BE, UCS-4LE, CP932, csWindows31J, SJIS, PCK, CP1250, CP1251, CP1252, external, locale

Default value is "UTF-8"

The character encoding used in this codec. Examples include "UTF-8" and "CP1252".

JSON requires valid UTF-8 strings, but in some cases, software that emits JSON does so in another encoding (nxlog, for example). In weird cases like this, you can set the `charset` setting to the actual encoding of the text and Logstash will convert it for you.

For nxlog users, you may to set this to "CP1252".

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when

you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

# json\_lines

Version: 3.0.2

Released on: 2016-07-14

[Changelog](#)

This codec will decode streamed JSON that is newline delimited. Encoding will emit a single JSON string ending in a @delimiter NOTE: Do not use this codec if your source input is line-oriented JSON, for example, redis or file inputs. Rather, use the json codec. More info: This codec is expecting to receive a stream (string) of newline terminated lines. The file input will produce a line string without a newline. Therefore this codec cannot work with line oriented inputs.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
json_lines {
}
```

Available configuration options:

Setting	Input type	Required	Default value
charset	<a href="#">string</a> , one of ["ASCII-8BIT", "UTF-8", "US-ASCII", "Big5", "Big5-HKSCS", "Big5-UAO", "CP949", "Emacs-Mule", "EUC-JP", "EUC-KR", "EUC-TW", "GB2312", "GB18030", "GBK", "ISO-8859-1", "ISO-8859-2", "ISO-8859-3", "ISO-8859-4", "ISO-8859-5", "ISO-8859-6", "ISO-8859-7", "ISO-8859-8", "ISO-8859-9", "ISO-8859-10", "ISO-8859-11", "ISO-8859-13", "ISO-8859-14", "ISO-8859-15", "ISO-8859-16", "KOI8-R", "KOI8-U", "Shift_JIS", "UTF-16BE", "UTF-16LE", "UTF-32BE", "UTF-32LE", "Windows-31J", "Windows-1250", "Windows-1251", "Windows-1252", "IBM437", "IBM737", "IBM775", "CP850", "IBM852", "CP852", "IBM855", "CP855", "IBM857", "IBM860", "IBM861", "IBM862", "IBM863", "IBM864", "IBM865", "IBM866", "IBM869", "Windows-1258", "GB1988", "macCentEuro", "macCroatian", "macCyrillic", "macGreek", "macIceland", "macRoman",	No	"UTF-8"

Setting	Input type	Required	Default value
	"macRomania", "macThai", "macTurkish", "macUkraine", "CP950", "CP951", "IBM037", "stateless-ISO-2022-JP", "eucJP-ms", "CP51932", "EUC-JIS-2004", "GB12345", "ISO-2022-JP", "ISO-2022-JP-2", "CP50220", "CP50221", "Windows-1256", "Windows-1253", "Windows-1255", "Windows-1254", "TIS-620", "Windows-874", "Windows-1257", "MacJapanese", "UTF-7", "UTF8-MAC", "UTF-16", "UTF-32", "UTF8-DoCoMo", "SJIS-DoCoMo", "UTF8-KDDI", "SJIS-KDDI", "ISO-2022-JP-KDDI", "stateless-ISO-2022-JP-KDDI", "UTF8-SoftBank", "SJIS-SoftBank", "BINARY", "CP437", "CP737", "CP775", "IBM850", "CP857", "CP860", "CP861", "CP862", "CP863", "CP864", "CP865", "CP866", "CP869", "CP1258", "Big5-HKSCS:2008", "ebcdic-cp-us", "eucJP", "euc-jp-ms", "EUC-JISX0213", "eucKR", "eucTW", "EUC-CN", "eucCN", "CP936", "ISO2022-JP", "ISO2022-JP2", "ISO8859-1", "ISO8859-2", "ISO8859-3", "ISO8859-4", "ISO8859-5", "ISO8859-6", "CP1256", "ISO8859-7", "CP1253", "ISO8859-8", "CP1255", "ISO8859-9", "CP1254", "ISO8859-10", "ISO8859-11", "CP874", "ISO8859-13", "CP1257", "ISO8859-14", "ISO8859-15", "ISO8859-16", "CP878", "MacJapan", "ASCII", "ANSI_X3.4-1968", "646", "CP65000", "CP65001", "UTF-8-MAC", "UTF-8-HFS", "UCS-2BE", "UCS-4BE", "UCS-4LE", "CP932", "csWindows31J", "SJIS", "PCK", "CP1250", "CP1251", "CP1252", "external", "locale"]		
delimiter	<a href="#">string</a>	No	"\n"
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	

## Details

[charset](#)

Value can be any of: ASCII-8BIT, UTF-8, US-ASCII, Big5, Big5-HKSCS, Big5-UAO, CP949, Emacs-Mule, EUC-JP, EUC-KR, EUC-TW, GB2312, GB18030, GBK, ISO-8859-1, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9, ISO-8859-10, ISO-8859-11, ISO-8859-13, ISO-8859-14, ISO-8859-15, ISO-8859-16, KOI8-R, KOI8-U, Shift\_JIS, UTF-16BE, UTF-16LE, UTF-32BE, UTF-32LE, Windows-31J, Windows-1250, Windows-1251, Windows-1252, IBM437, IBM737, IBM775, CP850, IBM852, CP852, IBM855, CP855, IBM857, IBM860, IBM861, IBM862, IBM863, IBM864, IBM865, IBM866, IBM869, Windows-1258, GB1988, macCentEuro, macCroatian,

## Logstash 5.2 Configuration Guide

```
macCyrillic, macGreek, macIceland, macRoman, macRomania, macThai, macTurkish,
macUkraine, CP950, CP951, IBM037, stateless-ISO-2022-JP, eucJP-ms, CP51932,
EUC-JIS-2004, GB12345, ISO-2022-JP, ISO-2022-JP-2, CP50220, CP50221, Windows-
1256, Windows-1253, Windows-1255, Windows-1254, TIS-620, Windows-874, Windows-
1257, MacJapanese, UTF-7, UTF8-MAC, UTF-16, UTF-32, UTF8-DoCoMo, SJIS-DoCoMo,
UTF8-KDDI, SJIS-KDDI, ISO-2022-JP-KDDI, stateless-ISO-2022-JP-KDDI, UTF8-
SoftBank, SJIS-SoftBank, BINARY, CP437, CP737, CP775, IBM850, CP857, CP860, CP861,
CP862, CP863, CP864, CP865, CP866, CP869, CP1258, Big5-HKSCS:2008, ebcdic-cp-us,
eucJP, euc-jp-ms, EUC-JISX0213, eucKR, eucTW, EUC-CN, eucCN, CP936, ISO2022-JP,
ISO2022-JP2, ISO8859-1, ISO8859-2, ISO8859-3, ISO8859-4, ISO8859-5, ISO8859-6,
CP1256, ISO8859-7, CP1253, ISO8859-8, CP1255, ISO8859-9, CP1254, ISO8859-10,
ISO8859-11, CP874, ISO8859-13, CP1257, ISO8859-14, ISO8859-15, ISO8859-16,
CP878, MacJapan, ASCII, ANSI_X3.4-1968, 646, CP65000, CP65001, UTF-8-MAC, UTF-8-
HFS, UCS-2BE, UCS-4BE, UCS-4LE, CP932, csWindows31J, SJIS, PCK, CP1250, CP1251,
CP1252, external, locale
```

Default value is "UTF-8"

The character encoding used in this codec. Examples include `UTF-8` and `CP1252`

JSON requires valid `UTF-8` strings, but in some cases, software that emits JSON does so in another encoding (nxlog, for example). In weird cases like this, you can set the `charset` setting to the actual encoding of the text and logstash will convert it for you.

For nxlog users, you'll want to set this to `CP1252`

`delimiter`

Value type is [string](#)

Default value is "\n"

Change the delimiter that separates lines

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

# line

Version: 3.0.2

Released on: 2016-07-14

[Changelog](#)

Line-oriented text data.

Decoding behavior: Only whole line events will be emitted.

Encoding behavior: Each event will be emitted with a trailing newline.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
line {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">charset</a>	<code>string</code> , one of ["ASCII-8BIT", "UTF-8", "US-ASCII", "Big5", "Big5-HKSCS", "Big5-UAO", "CP949", "Emacs-Mule", "EUC-JP", "EUC-KR", "EUC-TW", "GB2312", "GB18030", "GBK", "ISO-8859-1", "ISO-8859-2", "ISO-8859-3", "ISO-8859-4", "ISO-8859-5", "ISO-8859-6", "ISO-8859-7", "ISO-8859-8", "ISO-8859-9", "ISO-8859-10", "ISO-8859-11", "ISO-8859-13", "ISO-8859-14", "ISO-8859-15", "ISO-8859-16", "KOI8-R", "KOI8-U", "Shift_JIS", "UTF-16BE", "UTF-16LE", "UTF-32BE", "UTF-32LE", "Windows-31J", "Windows-1250", "Windows-1251", "Windows-1252", "IBM437", "IBM737", "IBM775", "CP850", "IBM852", "CP852", "IBM855", "CP855", "IBM857", "IBM860", "IBM861", "IBM862", "IBM863", "IBM864", "IBM865", "IBM866", "IBM869", "Windows-1258", "GB1988", "macCentEuro", "macCroatian", "macCyrillic", "macGreek", "macIceland", "macRoman", "macRomania", "macThai", "macTurkish", ]	No	"UTF-8"

Setting	Input type	Required	Default value
	"macUkraine", "CP950", "CP951", "IBM037", "stateless-ISO-2022-JP", "eucJP-ms", "CP51932", "EUC-JIS-2004", "GB12345", "ISO-2022-JP", "ISO-2022-JP-2", "CP50220", "CP50221", "Windows-1256", "Windows-1253", "Windows-1255", "Windows-1254", "TIS-620", "Windows-874", "Windows-1257", "MacJapanese", "UTF-7", "UTF8-MAC", "UTF-16", "UTF-32", "UTF8-DoCoMo", "SJIS-DoCoMo", "UTF8-KDDI", "SJIS-KDDI", "ISO-2022-JP-KDDI", "stateless-ISO-2022-JP-KDDI", "UTF8-SoftBank", "SJIS-SoftBank", "BINARY", "CP437", "CP737", "CP775", "IBM850", "CP857", "CP860", "CP861", "CP862", "CP863", "CP864", "CP865", "CP866", "CP869", "CP1258", "Big5-HKSCS:2008", "ebcdic-cp-us", "eucJP", "euc-jp-ms", "EUC-JISX0213", "eucKR", "eucTW", "EUC-CN", "eucCN", "CP936", "ISO2022-JP", "ISO2022-JP2", "ISO8859-1", "ISO8859-2", "ISO8859-3", "ISO8859-4", "ISO8859-5", "ISO8859-6", "CP1256", "ISO8859-7", "CP1253", "ISO8859-8", "CP1255", "ISO8859-9", "CP1254", "ISO8859-10", "ISO8859-11", "CP874", "ISO8859-13", "CP1257", "ISO8859-14", "ISO8859-15", "ISO8859-16", "CP878", "MacJapan", "ASCII", "ANSI_X3.4-1968", "646", "CP65000", "CP65001", "UTF-8-MAC", "UTF-8-HFS", "UCS-2BE", "UCS-4BE", "UCS-4LE", "CP932", "csWindows31J", "SJIS", "PCK", "CP1250", "CP1251", "CP1252", "external", "locale"]		
<a href="#">delimiter</a>	<a href="#">string</a>	No	"\n"
<a href="#">enable metric</a>	<a href="#">boolean</a>	No	true
<a href="#">format</a>	<a href="#">string</a>	No	
<a href="#">id</a>	<a href="#">string</a>	No	

## Details

### [charset](#)

**Value can be any of:** ASCII-8BIT, UTF-8, US-ASCII, Big5, Big5-HKSCS, Big5-UAO, CP949, Emacs-Mule, EUC-JP, EUC-KR, EUC-TW, GB2312, GB18030, GBK, ISO-8859-1, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9, ISO-8859-10, ISO-8859-11, ISO-8859-13, ISO-8859-14, ISO-8859-15, ISO-8859-16, KOI8-R, KOI8-U, Shift\_JIS, UTF-16BE, UTF-16LE, UTF-32BE, UTF-32LE, Windows-31J, Windows-1250, Windows-1251, Windows-1252, IBM437, IBM737, IBM775, CP850, IBM852, CP852, IBM855, CP855, IBM857, IBM860, IBM861, IBM862, IBM863, IBM864, IBM865, IBM866, IBM869, Windows-1258, GB1988, macCentEuro, macCroatian,

## Logstash 5.2 Configuration Guide

macCyrillic, macGreek, macIceland, macRoman, macRomania, macThai, macTurkish, macUkraine, CP950, CP951, IBM037, stateless-ISO-2022-JP, eucJP-ms, CP51932, EUC-JIS-2004, GB12345, ISO-2022-JP, ISO-2022-JP-2, CP50220, CP50221, Windows-1256, Windows-1253, Windows-1255, Windows-1254, TIS-620, Windows-874, Windows-1257, MacJapanese, UTF-7, UTF8-MAC, UTF-16, UTF-32, UTF8-DoCoMo, SJIS-DoCoMo, UTF8-KDDI, SJIS-KDDI, ISO-2022-JP-KDDI, stateless-ISO-2022-JP-KDDI, UTF8-SoftBank, SJIS-SoftBank, BINARY, CP437, CP737, CP775, IBM850, CP857, CP860, CP861, CP862, CP863, CP864, CP865, CP866, CP869, CP1258, Big5-HKSCS:2008, ebcdic-cp-us, eucJP, euc-jp-ms, EUC-JISX0213, eucKR, eucTW, EUC-CN, eucCN, CP936, ISO2022-JP, ISO2022-JP2, ISO8859-1, ISO8859-2, ISO8859-3, ISO8859-4, ISO8859-5, ISO8859-6, CP1256, ISO8859-7, CP1253, ISO8859-8, CP1255, ISO8859-9, CP1254, ISO8859-10, ISO8859-11, CP874, ISO8859-13, CP1257, ISO8859-14, ISO8859-15, ISO8859-16, CP878, MacJapan, ASCII, ANSI\_X3.4-1968, 646, CP65000, CP65001, UTF-8-MAC, UTF-8-HFS, UCS-2BE, UCS-4BE, UCS-4LE, CP932, csWindows31J, SJIS, PCK, CP1250, CP1251, CP1252, external, locale

Default value is "UTF-8"

The character encoding used in this input. Examples include `UTF-8` and `cp1252`

This setting is useful if your log files are in Latin-1 (aka `cp1252`) or in another character set other than `UTF-8`.

This only affects "plain" format logs since json is `UTF-8` already.

`delimiter`

Value type is [string](#)

Default value is `\n`

Change the delimiter that separates lines

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`format`

Value type is [string](#)

There is no default value for this setting.

Set the desired text format for encoding.

*id*

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

## msgpack

Version: 3.0.2

Released on: 2016-07-14

[Changelog](#)

Msgpack does not care about UTF-8 Treat as plain text and try to do the best we can with it?

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
msgpack {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">format</a>	<a href="#">string</a>	No	nil
<a href="#">id</a>	<a href="#">string</a>	No	

## Details

[enable\\_metric](#)

Value type is [boolean](#)

Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

[format](#)

Value type is [string](#)

Default value is `nil`

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

## multiline

Version: 3.0.3

Released on: 2016-09-15

[Changelog](#)

The multiline codec will collapse multiline messages and merge them into a single event.

The original goal of this codec was to allow joining of multiline messages from files into a single event. For example, joining Java exception and stacktrace messages into a single event.

The config looks like this:

```
input {
 stdin {
 codec => multiline {
 pattern => "pattern, a regexp"
 negate => "true" or "false"
 what => "previous" or "next"
 }
 }
}
```

The `pattern` should match what you believe to be an indicator that the field is part of a multi-line event.

The `what` must be `previous` or `next` and indicates the relation to the multi-line event.

The `negate` can be `true` or `false` (defaults to `false`). If `true`, a message not matching the pattern will constitute a match of the multiline filter and the `what` will be applied. (vice-versa is also true)

For example, Java stack traces are multiline and usually have the message starting at the far-left, with each subsequent line indented. Do this:

```
input {
 stdin {
 codec => multiline {
 pattern => "^\\s"
 what => "previous"
 }
 }
}
```

This says that any line starting with whitespace belongs to the previous line.

Another example is to merge lines not starting with a date up to the previous line..

```
input {
 file {
 path => "/var/log/someapp.log"
 codec => multiline {
 # Grok pattern names are valid! :)
 pattern => "^%{TIMESTAMP_ISO8601} "
 negate => true
 what => previous
 }
 }
}
```

This says that any line not starting with a timestamp should be merged with the previous line.

One more common example is C line continuations (backslash). Here's how to do that:

```
filter {
 multiline {
 type => "somefiletype"
 pattern => "\\$"
 what => "next"
 }
}
```

This says that any line ending with a backslash should be combined with the following line.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
multiline {
 pattern => ...
 what => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#"><u>auto_flush_interval</u></a>	<a href="#"><u>number</u></a>	No	
<a href="#"><u>charset</u></a>	<a href="#"><u>string</u></a> , one of ["ASCII-8BIT", "UTF-8", "US-ASCII", "Big5", "Big5-HKSCS", "Big5-UAO", "CP949", "Emacs-Mule", "EUC-JP", "EUC-KR", "EUC-TW",	No	"UTF-8"

Setting	Input type	Required	Default value
	"GB2312", "GB18030", "GBK", "ISO-8859-1", "ISO-8859-2", "ISO-8859-3", "ISO-8859-4", "ISO-8859-5", "ISO-8859-6", "ISO-8859-7", "ISO-8859-8", "ISO-8859-9", "ISO-8859-10", "ISO-8859-11", "ISO-8859-13", "ISO-8859-14", "ISO-8859-15", "ISO-8859-16", "KOI8-R", "KOI8-U", "Shift_JIS", "UTF-16BE", "UTF-16LE", "UTF-32BE", "UTF-32LE", "Windows-31J", "Windows-1250", "Windows-1251", "Windows-1252", "IBM437", "IBM737", "IBM775", "CP850", "IBM852", "CP852", "IBM855", "CP855", "IBM857", "IBM860", "IBM861", "IBM862", "IBM863", "IBM864", "IBM865", "IBM866", "IBM869", "Windows-1258", "GB1988", "macCentEuro", "macCroatian", "macCyrillic", "macGreek", "macIceland", "macRoman", "macRomania", "macThai", "macTurkish", "macUkraine", "CP950", "CP951", "IBM037", "stateless-ISO-2022-JP", "eucJP-ms", "CP51932", "EUC-JIS-2004", "GB12345", "ISO-2022-JP", "ISO-2022-JP-2", "CP50220", "CP50221", "Windows-1256", "Windows-1253", "Windows-1255", "Windows-1254", "TIS-620", "Windows-874", "Windows-1257", "MacJapanese", "UTF-7", "UTF8-MAC", "UTF-16", "UTF-32", "UTF8-DoCoMo", "SJIS-DoCoMo", "UTF8-KDDI", "SJIS-KDDI", "ISO-2022-JP-KDDI", "stateless-ISO-2022-JP-KDDI", "UTF8-SoftBank", "SJIS-SoftBank", "BINARY", "CP437", "CP737", "CP775", "IBM850", "CP857", "CP860", "CP861", "CP862", "CP863", "CP864", "CP865", "CP866", "CP869", "CP1258", "Big5-HKSCS:2008", "ebcdic-cp-us", "eucJP", "euc-jp-ms", "EUC-JISX0213", "eucKR", "eucTW", "EUC-CN", "eucCN", "CP936", "ISO2022-JP", "ISO2022-JP2", "ISO8859-1", "ISO8859-2", "ISO8859-3", "ISO8859-4", "ISO8859-5", "ISO8859-6", "CP1256", "ISO8859-7", "CP1253", "ISO8859-8", "CP1255", "ISO8859-9", "CP1254", "ISO8859-10", "ISO8859-11", "CP874", "ISO8859-13", "CP1257", "ISO8859-14", "ISO8859-15", "ISO8859-16", "CP878", "MacJapan", "ASCII", "ANSI_X3.4-1968", "646", "CP65000", "CP65001", "UTF-8-MAC", "UTF-8-HFS", "UCS-2BE", "UCS-4BE",		

Setting	Input type	Required	Default value
	"UCS-4LE", "CP932", "csWindows31J", "SJIS", "PCK", "CP1250", "CP1251", "CP1252", "external", "locale"]		
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">max_bytes</a>	<a href="#">bytes</a>	No	"10 MiB"
<a href="#">max_lines</a>	<a href="#">number</a>	No	500
<a href="#">multiline_tag</a>	<a href="#">string</a>	No	"multiline"
<a href="#">negate</a>	<a href="#">boolean</a>	No	false
<a href="#">pattern</a>	<a href="#">string</a>	Yes	
<a href="#">patterns_dir</a>	<a href="#">array</a>	No	[]
<a href="#">what</a>	<a href="#">string</a> , one of ["previous", "next"]	Yes	

## Details

[auto\\_flush\\_interval](#)

Value type is [number](#)

There is no default value for this setting.

The accumulation of multiple lines will be converted to an event when either a matching new line is seen or there has been no new data appended for this time `auto_flush_interval`. No default. If unset, no `auto_flush`. Units: seconds

[charset](#)

Value can be any of: ASCII-8BIT, UTF-8, US-ASCII, Big5, Big5-HKSCS, Big5-UAO, CP949, Emacs-Mule, EUC-JP, EUC-KR, EUC-TW, GB2312, GB18030, GBK, ISO-8859-1, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9, ISO-8859-10, ISO-8859-11, ISO-8859-13, ISO-8859-14, ISO-8859-15, ISO-8859-16, KOI8-R, KOI8-U, Shift\_JIS, UTF-16BE, UTF-16LE, UTF-32BE, UTF-32LE, Windows-31J, Windows-1250, Windows-1251, Windows-1252, IBM437, IBM737, IBM775, CP850, IBM852, CP852, IBM855, CP855, IBM857, IBM860, IBM861, IBM862, IBM863, IBM864, IBM865, IBM866, IBM869, Windows-1258, GB1988, macCentEuro, macCroatian, macCyrillic, macGreek, macIceland, macRoman, macRomania, macThai, macTurkish, macUkraine, CP950, CP951, IBM037, stateless-ISO-2022-JP, eucJP-ms, CP51932, EUC-JIS-2004, GB12345, ISO-2022-JP, ISO-2022-JP-2, CP50220, CP50221, Windows-1256, Windows-1253, Windows-1255, Windows-1254, TIS-620, Windows-874, Windows-1257, MacJapanese, UTF-7, UTF8-MAC, UTF-16, UTF-32, UTF8-DoCoMo, SJIS-DoCoMo,

## Logstash 5.2 Configuration Guide

UTF8-KDDI, SJIS-KDDI, ISO-2022-JP-KDDI, stateless-ISO-2022-JP-KDDI, UTF8-SoftBank, SJIS-SoftBank, BINARY, CP437, CP737, CP775, IBM850, CP857, CP860, CP861, CP862, CP863, CP864, CP865, CP866, CP869, CP1258, Big5-HKSCS:2008, ebcdic-cp-us, eucJP, euc-jp-ms, EUC-JISX0213, eucKR, eucTW, EUC-CN, eucCN, CP936, ISO2022-JP, ISO2022-JP2, ISO8859-1, ISO8859-2, ISO8859-3, ISO8859-4, ISO8859-5, ISO8859-6, CP1256, ISO8859-7, CP1253, ISO8859-8, CP1255, ISO8859-9, CP1254, ISO8859-10, ISO8859-11, CP874, ISO8859-13, CP1257, ISO8859-14, ISO8859-15, ISO8859-16, CP878, MacJapan, ASCII, ANSI\_X3.4-1968, 646, CP65000, CP65001, UTF-8-MAC, UTF-8-HFS, UCS-2BE, UCS-4BE, UCS-4LE, CP932, csWindows31J, SJIS, PCK, CP1250, CP1251, CP1252, external, locale

Default value is "UTF-8"

The character encoding used in this input. Examples include `UTF-8` and `cp1252`

This setting is useful if your log files are in Latin-1 (aka `cp1252`) or in another character set other than `UTF-8`.

This only affects "plain" format logs since JSON is `UTF-8` already.

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

`max_bytes`

Value type is [bytes](#)

Default value is "10 MiB"

The accumulation of events can make logstash exit with an out of memory error if event boundaries are not correctly defined. This settings make sure to flush multiline events after reaching a number of bytes, it is used in combination max\_lines.

`max_lines`

Value type is [number](#)

Default value is 500

The accumulation of events can make logstash exit with an out of memory error if event boundaries are not correctly defined. This settings make sure to flush multiline events after reaching a number of lines, it is used in combination max\_bytes.

`multiline_tag`

Value type is [string](#)

Default value is "multiline"

Tag multiline events with a given tag. This tag will only be added to events that actually have multiple lines in them.

`negate`

Value type is [boolean](#)

Default value is `false`

Negate the regexp pattern (*if not matched*).

`pattern`

This is a required setting.

Value type is [string](#)

There is no default value for this setting.

The regular expression to match.

`patterns_dir`

Value type is [array](#)

Default value is [ ]

Logstash ships by default with a bunch of patterns, so you don't necessarily need to define this yourself unless you are adding additional patterns.

Pattern files are plain text with format:

NAME PATTERN

For example:

NUMBER \d+  
*what*

This is a required setting.

Value can be any of: previous, next

There is no default value for this setting.

If the pattern matched, does event belong to the next or previous event?

# netflow

Version: 3.2.2

Released on: 2016-12-25

[Changelog](#)

The "netflow" codec is used for decoding Netflow v5/v9/v10 (IPFIX) flows.

## Supported Netflow/IPFIX exporters

The following Netflow/IPFIX exporters are known to work with the most recent version of the netflow codec:

Netflow exporter	v5	v9	IPFIX	Remarks
Softflowd	y	y	y	IPFIX supported in <a href="https://github.com/djmdjm/softflowd">https://github.com/djmdjm/softflowd</a>
nProbe	y	y	y	
ipt_NETFLOW	y	y	y	
Cisco ASA		y		
Cisco IOS 12.x		y		
fprobe	y			
Juniper MX80	y			SW > 12.3R8
OpenBSD pflow	y	n	y	<a href="http://man.openbsd.org/OpenBSD-current/man4/pflow.4">http://man.openbsd.org/OpenBSD-current/man4/pflow.4</a>
Mikrotik 6.35.4	y		n	<a href="http://wiki.mikrotik.com/wiki/Manual:IP/Traffic_Flow">http://wiki.mikrotik.com/wiki/Manual:IP/Traffic_Flow</a>
Ubiquiti Edgerouter X		y		With MPLS labels
Citrix Netscaler			y	Still some unknown fields, labeled netscalerUnknown<id>

## Usage

Example Logstash configuration:

```
input {
 udp {
 host => localhost
 port => 2055
 codec => netflow {
 versions => [5, 9]
 }
 type => netflow
 }
 udp {
 host => localhost
 }
}
```

```
port => 4739
codec => netflow {
 versions => [10]
 target => ipfix
}
type => ipfix
}
tcp {
 host => localhost
 port => 4739
 codec => netflow {
 versions => [10]
 target => ipfix
 }
 type => ipfix
}
}
```

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
netflow {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">cache_save_path</a>	a valid filesystem path	No	
<a href="#">cache_ttl</a>	<a href="#">number</a>	No	4000
<a href="#">enable_metric</a>	<a href="#">boolean</a>	No	true
<a href="#">id</a>	<a href="#">string</a>	No	
<a href="#">include_flowset_id</a>	<a href="#">boolean</a>	No	false
<a href="#">ipfix_definitions</a>	a valid filesystem path	No	
<a href="#">netflow_definitions</a>	a valid filesystem path	No	
<a href="#">target</a>	<a href="#">string</a>	No	"netflow"
<a href="#">versions</a>	<a href="#">array</a>	No	[5, 9, 10]

## Details

`cache_save_path`

Value type is [path](#)

There is no default value for this setting.

Where to save the template cache This helps speed up processing when restarting logstash (So you don't have to await the arrival of templates) cache will save as path/netflow\_templates.cache and/or path/ipfix\_templates.cache

`cache_ttl`

Value type is [number](#)

Default value is 4000

Netflow v9/v10 template cache TTL (minutes)

`enable_metric`

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`id`

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
include_flowset_id
```

Value type is [boolean](#)

Default value is `false`

Only makes sense for ipfix, v9 already includes this Setting to true will include the flowset\_id in events Allows you to work with sequences, for instance with the aggregate filter

### *ipfix\_definitions*

Value type is [path](#)

There is no default value for this setting.

Override YAML file containing IPFIX field definitions

Very similar to the Netflow version except there is a top level Private Enterprise Number (PEN) key added:

```
pen:
 id:
 - :uintN or :ip4_addr or :ip6_addr or :mac_addr or :string
 - :name
 id:
 - :skip
```

There is an implicit PEN 0 for the standard fields.

See <https://github.com/logstash-plugins/logstash-codec-netflow/blob/master/lib/logstash/codecs/netflow/ipfix.yaml> for the base set.

### *netflow\_definitions*

Value type is [path](#)

There is no default value for this setting.

Override YAML file containing Netflow field definitions

Each Netflow field is defined like so:

```

id:
- default length in bytes
- :name
id:
- :uintN or :ip4_addr or :ip6_addr or :mac_addr or :string
- :name
id:
- :skip
```

See <https://github.com/logstash-plugins/logstash-codec-netflow/blob/master/lib/logstash/codecs/netflow/netflow.yaml> for the base set.

*target*

Value type is [string](#)

Default value is "netflow"

Specify into what field you want the Netflow data.

*versions*

Value type is [array](#)

Default value is [ 5, 9, 10 ]

Specify which Netflow versions you will accept.

## nmap

This codec is used to parse [nmap](#) output data which is serialized in XML format. Nmap ("Network Mapper") is a free and open source utility for network discovery and security auditing. For more information on nmap, see <https://nmap.org/>.

Note: This codec can only be used for decoding data.

Event types are listed below

`nmap_scan_metadata`: An object containing top level information about the scan, including how many hosts were up, and how many were down. Useful for the case where you need to check if a DNS based hostname does not resolve, where both those numbers will be zero. `nmap_host`: One event is created per host. The full data covering an individual host, including open ports and traceroute information as a nested structure. `nmap_port`: One event is created per host/port. This duplicates data already in `nmap_host`: This was put in for the case where you want to model ports as separate documents in Elasticsearch (which Kibana prefers). `nmap_traceroute_link`: One of these is output per traceroute *connection*, with a `from` and a `to` object describing each hop. Note that traceroute hop data is not always correct due to the fact that each tracing ICMP packet may take a different route. Also very useful for Kibana visualizations.

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
nmap {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>emit_hosts</code>	<a href="#">boolean</a>	No	true
<code>emit_ports</code>	<a href="#">boolean</a>	No	true
<code>emit_scan_metadata</code>	<a href="#">boolean</a>	No	true
<code>emit_traceroute_links</code>	<a href="#">boolean</a>	No	true

## Details

`emit_hosts`

Value type is [boolean](#)

Default value is `true`

Emit all host data as a nested document (including ports + traceroutes) with the type *nmap\_fullscan*

`emit_ports`

Value type is [boolean](#)

Default value is `true`

Emit each port as a separate document with type *nmap\_port*

`emit_scan_metadata`

Value type is [boolean](#)

Default value is `true`

Emit scan metadata

`emit_traceroute_links`

Value type is [boolean](#)

Default value is `true`

Emit each hop\_tuple of the traceroute with type *nmap\_traceroute\_link*

## **oldlogstashjson**

### **Synopsis**

This plugin has no configuration options.

Complete configuration example:

```
oldlogstashjson {
}
```

# plain

Version: 3.0.2

Released on: 2016-07-14

[Changelog](#)

The "plain" codec is for plain text with no delimiting between events.

This is mainly useful on inputs and outputs that already have a defined framing in their transport protocol (such as zeromq, rabbitmq, redis, etc)

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
plain {
}
```

Available configuration options:

Setting	Input type	Required	Default value
<a href="#">charset</a>	<code>string</code> , one of ["ASCII-8BIT", "UTF-8", "US-ASCII", "Big5", "Big5-HKSCS", "Big5-UAO", "CP949", "Emacs-Mule", "EUC-JP", "EUC-KR", "EUC-TW", "GB2312", "GB18030", "GBK", "ISO-8859-1", "ISO-8859-2", "ISO-8859-3", "ISO-8859-4", "ISO-8859-5", "ISO-8859-6", "ISO-8859-7", "ISO-8859-8", "ISO-8859-9", "ISO-8859-10", "ISO-8859-11", "ISO-8859-13", "ISO-8859-14", "ISO-8859-15", "ISO-8859-16", "KOI8-R", "KOI8-U", "Shift_JIS", "UTF-16BE", "UTF-16LE", "UTF-32BE", "UTF-32LE", "Windows-31J", "Windows-1250", "Windows-1251", "Windows-1252", "IBM437", "IBM737", "IBM775", "CP850", "IBM852", "CP852", "IBM855", "CP855", "IBM857", "IBM860", "IBM861", "IBM862", "IBM863", "IBM864", "IBM865", "IBM866", "IBM869", "Windows-1258", "GB1988", "macCentEuro", "macCroatian", "macCyrillic", "macGreek", "macIceland", "macRoman", "macRomania", "macThai", "macTurkish", "macUkraine", "CP950", "CP951", "IBM037",	No	"UTF-8"

Setting	Input type	Required	Default value
	"stateless-ISO-2022-JP", "eucJP-ms", "CP51932", "EUC-JIS-2004", "GB12345", "ISO-2022-JP", "ISO-2022-JP-2", "CP50220", "CP50221", "Windows-1256", "Windows-1253", "Windows-1255", "Windows-1254", "TIS-620", "Windows-874", "Windows-1257", "MacJapanese", "UTF-7", "UTF8-MAC", "UTF-16", "UTF-32", "UTF8-DoCoMo", "SJIS-DoCoMo", "UTF8-KDDI", "SJIS-KDDI", "ISO-2022-JP-KDDI", "stateless-ISO-2022-JP-KDDI", "UTF8-SoftBank", "SJIS-SoftBank", "BINARY", "CP437", "CP737", "CP775", "IBM850", "CP857", "CP860", "CP861", "CP862", "CP863", "CP864", "CP865", "CP866", "CP869", "CP1258", "Big5-HKSCS:2008", "ebcdic-cp-us", "eucJP", "euc-jp-ms", "EUC-JISX0213", "eucKR", "eucTW", "EUC-CN", "eucCN", "CP936", "ISO2022-JP", "ISO2022-JP2", "ISO8859-1", "ISO8859-2", "ISO8859-3", "ISO8859-4", "ISO8859-5", "ISO8859-6", "CP1256", "ISO8859-7", "CP1253", "ISO8859-8", "CP1255", "ISO8859-9", "CP1254", "ISO8859-10", "ISO8859-11", "CP874", "ISO8859-13", "CP1257", "ISO8859-14", "ISO8859-15", "ISO8859-16", "CP878", "MacJapan", "ASCII", "ANSI_X3.4-1968", "646", "CP65000", "CP65001", "UTF-8-MAC", "UTF-8-HFS", "UCS-2BE", "UCS-4BE", "UCS-4LE", "CP932", "csWindows31J", "SJIS", "PCK", "CP1250", "CP1251", "CP1252", "external", "locale"]		
<a href="#">enable metric</a>	<a href="#">boolean</a>	No	true
<a href="#">format</a>	<a href="#">string</a>	No	
<a href="#">id</a>	<a href="#">string</a>	No	

## Details

[charset](#)

**Value can be any of:** ASCII-8BIT, UTF-8, US-ASCII, Big5, Big5-HKSCS, Big5-UAO, CP949, Emacs-Mule, EUC-JP, EUC-KR, EUC-TW, GB2312, GB18030, GBK, ISO-8859-1, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9, ISO-8859-10, ISO-8859-11, ISO-8859-13, ISO-8859-14, ISO-8859-15, ISO-8859-16, KOI8-R, KOI8-U, Shift\_JIS, UTF-16BE, UTF-16LE, UTF-32BE, UTF-32LE, Windows-31J, Windows-1250, Windows-1251, Windows-1252, IBM437, IBM737, IBM775, CP850, IBM852, CP852, IBM855, CP855, IBM857, IBM860, IBM861, IBM862, IBM863, IBM864, IBM865, IBM866, IBM869, Windows-1258, GB1988, macCentEuro, macCroatian, macCyrillic, macGreek, macIceland, macRoman, macRomania, macThai, macTurkish, macUkraine, CP950, CP951, IBM037, stateless-ISO-2022-JP, eucJP-ms, CP51932,

## Logstash 5.2 Configuration Guide

EUC-JIS-2004, GB12345, ISO-2022-JP, ISO-2022-JP-2, CP50220, CP50221, Windows-1256, Windows-1253, Windows-1255, Windows-1254, TIS-620, Windows-874, Windows-1257, MacJapanese, UTF-7, UTF8-MAC, UTF-16, UTF-32, UTF8-DoCoMo, SJIS-DoCoMo, UTF8-KDDI, SJIS-KDDI, ISO-2022-JP-KDDI, stateless-ISO-2022-JP-KDDI, UTF8-SoftBank, SJIS-SoftBank, BINARY, CP437, CP737, CP775, IBM850, CP857, CP860, CP861, CP862, CP863, CP864, CP865, CP866, CP869, CP1258, Big5-HKSCS:2008, ebcdic-cp-us, eucJP, euc-jp-ms, EUC-JISX0213, eucKR, eucTW, EUC-CN, eucCN, CP936, ISO2022-JP, ISO2022-JP2, ISO8859-1, ISO8859-2, ISO8859-3, ISO8859-4, ISO8859-5, ISO8859-6, CP1256, ISO8859-7, CP1253, ISO8859-8, CP1255, ISO8859-9, CP1254, ISO8859-10, ISO8859-11, CP874, ISO8859-13, CP1257, ISO8859-14, ISO8859-15, ISO8859-16, CP878, MacJapan, ASCII, ANSI\_X3.4-1968, 646, CP65000, CP65001, UTF-8-MAC, UTF-8-HFS, UCS-2BE, UCS-4BE, UCS-4LE, CP932, csWindows31J, SJIS, PCK, CP1250, CP1251, CP1252, external, locale

**Default value is "UTF-8"**

The character encoding used in this input. Examples include `UTF-8` and `cp1252`

This setting is useful if your log files are in Latin-1 (aka `cp1252`) or in another character set other than `UTF-8`.

This only affects "plain" format logs since json is `UTF-8` already.

`enable_metric`

**Value type is [boolean](#)**

**Default value is `true`**

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

`format`

**Value type is [string](#)**

There is no default value for this setting.

Set the message you which to emit for each event. This supports `sprintf` strings.

This setting only affects outputs (encoding of events).

`id`

**Value type is [string](#)**

There is no default value for this setting.

Add a unique `ID` to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {
 stdout {
 id => "my_plugin_id"
 }
}
```

## protobuf

Version: 1.0.0

Released on: 2016-12-04

[Changelog](#)

This plugin does not ship with Logstash by default, but it is easy to install by running  
`bin/logstash-plugin install logstash-codec-protobuf`.

This codec converts protobuf encoded messages into logstash events and vice versa.

Requires the protobuf definitions as ruby files. You can create those using the [ruby-protoc compiler](<https://github.com/codekitchen/ruby-protocol-buffers>).

The following shows a usage example for decoding events from a kafka stream:

```
kafka
{
 zk_connect => "127.0.0.1"
 topic_id => "your_topic_goes_here"
 codec => protobuf
 {
 class_name => "Animal::Unicorn"
 include_path => ['/path/to/protobuf/definitions/UnicornProtobuf.pb.rb']
 }
}
```

## Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
protobuf {
 class_name => ...
 include_path => ...
}
```

Available configuration options:

Setting	Input type	Required	Default value
<code>class_name</code>	<a href="#">string</a>	Yes	

Setting	Input type	Required	Default value
enable_metric	<a href="#">boolean</a>	No	true
id	<a href="#">string</a>	No	
include_path	<a href="#">array</a>	Yes	

## Details

*class\_name*

This is a required setting.

Value type is [string](#)

There is no default value for this setting.

Name of the class to decode. If your protobuf definition contains modules, prepend them to the class name with double colons like so:

```
class_name => "Foods::Dairy::Cheese"
```

This corresponds to a protobuf definition starting as follows:

```
module Foods
 module Dairy
 class Cheese
 # here are your field definitions.
```

If your class references other definitions: you only have to add the main class here.

*enable\_metric*

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

*id*

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

`include_path`

This is a required setting.

Value type is [array](#)

There is no default value for this setting.

List of absolute pathes to files with protobuf definitions. When using more than one file, make sure to arrange the files in reverse order of dependency so that each class is loaded before it is referred to by another.

Example: a class *Cheese* referencing another protobuf class *Milk*

```
module Foods
  module Dairy
    class Cheese
      set_fully_qualified_name "Foods.Dairy.Cheese"
      optional ::Foods::Cheese::Milk, :milk, 1
      optional :int64, :unique_id, 2
      # here be more field definitions
```

would be configured as

```
include_path =>
['/path/to/protobuf/definitions/Milk.pb.rb', '/path/to/protobuf/definitions/Cheese.pb.rb']
```

When using the codec in an output plugin: * make sure to include all the desired fields in the protobuf definition, including timestamp. Remove fields that are not part of the protobuf definition from the event by using the mutate filter. * the `@` symbol is currently not supported in field names when loading the protobuf definitions for encoding. Make sure to call the timestamp field "timestamp" instead of "@timestamp" in the protobuf file. Logstash event fields will be stripped of the leading `@` before conversion.

rubydebug

Version: 3.0.2

Released on: 2016-07-14

[Changelog](#)

The rubydebug codec will output your Logstash event data using the Ruby Awesome Print library.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
rubydebug {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
enable_metric	boolean	No	true
id	string	No	
metadata	boolean	No	false

Details

[enable_metric](#)

Value type is [boolean](#)

Default value is true

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

id

Value type is [string](#)

There is no default value for this setting.

Add a unique ID to the plugin configuration. If no ID is specified, Logstash will generate one. It is strongly recommended to set this ID in your configuration. This is particularly useful when you have two or more plugins of the same type, for example, if you have 2 grok filters. Adding a named ID in this case will help in monitoring Logstash when using the monitoring APIs.

```
output {  
  stdout {  
    id => "my_plugin_id"  
  }  
}  
metadata
```

Value type is [boolean](#)

Default value is `false`

Should the event's metadata be included?

s3_plain

NOTE: This is a community-maintained plugin! It does not ship with Logstash by default, but it is easy to install by running `bin/logstash-plugin install logstash-codec-s3_plain`.

The "s3_plain" codec is used for backward compatibility with previous version of the S3 Output

Synopsis

This plugin has no configuration options.

Complete configuration example:

```
s3_plain {  
}
```

sflow

Version: 2.0.0

Released on: 2016-11-02

[Changelog](#)

Compatible: 5.1.1.1, 5.0.0, 2.4.1, 2.4.0, 2.3.4

NOTE: This plugin does not ship with Logstash by default, but it is easy to install by running
bin/logstash-plugin install logstash-codec-sflow.

The "sflow" codec is for decoding sflow v5 flows.

Synopsis

This plugin supports the following configuration options:

Required configuration options:

```
sflow {  
}
```

Available configuration options:

Setting	Input type	Required	Default value
enable_metric	boolean	No	true
id	string	No	
interface_cache_size	number	No	1000
interface_cache_ttl	number	No	3600
optional_removed_field	array	No	<ul style="list-style-type: none"> "sflow_version", "header_size", "ip_header_length", "ip_dscp", "ip_ecn", "ip_total_length", "ip_identification", "ip_flags", "ip_fragment_offset", "ip_ttl", "ip_checksum", "ip_options", "tcp_seq_number", "tcp_ack_number", "tcp_header_length", "tcp_reserved", "tcp_is_nonce", "tcp_is_cwr", "tcp_is_ecn_echo", "tcp_is_urgent", "tcp_is_ack", "tcp_is_push",

Setting	Input type	Required	Default value
			"tcp_is_reset", "tcp_is_syn", "tcp_is_fin", "tcp_window_size", "tcp_checksum", "tcp_urgent_pointer", "tcp_options", "vlan_cfi", "sequence_number", "flow_sequence_number", "vlan_type", "udp_length", "udp_checksum"]
snmp_community	string	No	"public"
snmp_interface	boolean	No	false
versions	array	No	[5]

Details

[*enable_metric*](#)

Value type is [boolean](#)

Default value is `true`

Disable or enable metric logging for this specific plugin instance by default we record all the metrics we can, but you can disable metrics collection for a specific plugin.

[*id*](#)

Value type is [string](#)

There is no default value for this setting.

Add a unique `ID` to the plugin instance, this `ID` is used for tracking information for a specific configuration of the plugin.

```
``` output { stdout { id => "ABC" } } ```
```

If you don't explicitly set this variable Logstash will generate a unique name.

[\*interface\\_cache\\_size\*](#)

Value type is [number](#)

Default value is `1000`

Specify the max number of element in the interface resolution local cache (only if snmp\_interface true)

*interface\_cache\_ttl*

Value type is [number](#)

Default value is 3600

Specify the duration for each element in the interface resolution local cache (only if snmp\_interface true)

*optional\_removed\_field*

Value type is [array](#)

Default value is ["sflow\_version", "header\_size", "ip\_header\_length", "ip\_dscp", "ip\_ecn", "ip\_total\_length", "ip\_identification", "ip\_flags", "ip\_fragment\_offset", "ip\_ttl", "ip\_checksum", "ip\_options", "tcp\_seq\_number", "tcp\_ack\_number", "tcp\_header\_length", "tcp\_reserved", "tcp\_is\_nonce", "tcp\_is\_cwr", "tcp\_is\_ecn\_echo", "tcp\_is\_urgent", "tcp\_is\_ack", "tcp\_is\_push", "tcp\_is\_reset", "tcp\_is\_syn", "tcp\_is\_fin", "tcp\_window\_size", "tcp\_checksum", "tcp\_urgent\_pointer", "tcp\_options", "vlan\_cfi", "sequence\_number", "flow\_sequence\_number", "vlan\_type", "udp\_length", "udp\_checksum"]

Specify which sflow fields must not be send in the event

*snmp\_community*

Value type is [string](#)

Default value is "public"

Specify if codec must perform SNMP call so agent\_ip for interface resolution.

*snmp\_interface*

Value type is [boolean](#)

Default value is false

Specify if codec must perform SNMP call so agent\_ip for interface resolution.

*versions*

Value type is [array](#)

Default value is [ 5 ]

Specify which Sflow versions you will accept.

# Contributing to Logstash

Before version 1.5, Logstash included all plugins in each release. This made it easy to make use of any plugin, but it complicated plugin development—a new release of Logstash became necessary if a plugin needed patching. Since version 1.5, all plugins are independent of the Logstash core. Now you can add your own input, codec, filter, or output plugins to Logstash much more easily!

## Adding plugins

Since plugins can now be developed and deployed independently of the Logstash core, there are documents which guide you through the process of coding and deploying your own plugins:

[Generating a New Plugin](#)

[How to write a Logstash input plugin](#)

[How to write a Logstash codec plugin](#)

[How to write a Logstash filter plugin](#)

[How to write a Logstash output plugin](#)

[Contributing a Patch to a Logstash Plugin](#)

[Community Maintainer's Guide](#)

[Submitting a Plugin](#)

### *Plugin Shutdown APIs*

Starting in Logstash 2.0, we changed how input plugins shut down to increase shutdown reliability. There are three methods for plugin shutdown: `stop`, `stop?`, and `close`.

Call the `stop` method from outside the plugin thread. This method signals the plugin to stop.

The `stop?` method returns `true` when the `stop` method has already been called for that plugin.

The `close` method performs final bookkeeping and cleanup after the plugin's `run` method and the plugin's thread both exit. The `close` method is a new name for the method known as `teardown` in previous versions of Logstash.

The `shutdown`, `finished`, `finished?`, `running?`, and `terminating?` methods are redundant and no longer present in the Plugin Base class.

Sample code for the plugin shutdown APIs is [available](#).

## Extending Logstash core

We also welcome contributions and bug fixes to the Logstash core feature set.

Please read through our [contribution](#) guide, and the Logstash [readme](#) document.

# How to write a Logstash input plugin

To develop a new input for Logstash, you build a self-contained Ruby gem whose source code lives in its own GitHub repository. The Ruby gem can then be hosted and shared on RubyGems.org. You can use the example input implementation as a starting point. (If you’re unfamiliar with Ruby, you can find an excellent quickstart guide at <https://www.ruby-lang.org/en/documentation/quickstart/>.)

## Get started

Let’s step through creating an input plugin using the [example input plugin](#).

### Create a GitHub repo for your new plugin

Each Logstash plugin lives in its own GitHub repository. To create a new repository for your plugin:

Log in to GitHub.

Click the **Repositories** tab. You’ll see a list of other repositories you’ve forked or contributed to.

Click the green **New** button in the upper right.

Specify the following settings for your new repo:

**Repository name** — a unique name of the form `logstash-input-pluginname`.

**Public or Private** — your choice, but the repository must be Public if you want to submit it as an official plugin.

**Initialize this repository with a README** — enables you to immediately clone the repository to your computer.

Click **Create Repository**.

## Use the plugin generator tool

You can now create your own Logstash plugin in seconds! The `generate` subcommand of `bin/logstash-plugin` creates the foundation for a new Logstash plugin with templated files. It creates the correct directory structure, gemspec files, and dependencies so you can start adding custom code to process data with Logstash.

For more information, see [Generating Plugins](#)

## Copy the input code

Alternatively, you can use the examples repo we host on github.com

**Clone your plugin.** Replace `GITUSERNAME` with your github username, and `MYPLUGINNAME` with your plugin name.

```
git clone https://github.com/GITUSERNAME/logstash-input-MYPLUGINNAME.git
```

alternately, via ssh: `git clone git@github.com:GITUSERNAME/logstash-input-MYPLUGINNAME.git`

```
cd logstash-input-MYPLUGINNAME
```

You don't want to include the example .git directory or its contents, so delete it before you copy the example.

```
cd /tmp
```

```
git clone https://github.com/logstash-plugins/logstash-input-example.git
```

```
cd logstash-input-example
```

```
rm -rf .git
```

```
cp -R * /path/to/logstash-input-mypluginname/
```

**Rename the following files to match the name of your plugin.**

```
logstash-input-example.gemspec
```

```
example.rb
```

```
example_spec.rb
```

```
cd /path/to/logstash-input-mypluginname
mv logstash-input-example.gemspec logstash-input-
```

```
mypluginname.gemspec
```

```
mv lib/logstash/inputs/example.rb
```

```
lib/logstash/inputs/mypluginname.rb
```

```
mv spec/inputs/example_spec.rb spec/inputs/mypluginname_spec.rb
```

Your file structure should look like this:

```
$ tree logstash-input-mypluginname
└── Gemfile
└── LICENSE
└── README.md
```

```
└── Rakefile
└── lib
 └── logstash
 └── inputs
 └── mypluginname.rb
└── logstash-input-mypluginname.gemspec
└── spec
 └── inputs
 └── mypluginname_spec.rb
```

For more information about the Ruby gem file structure and an excellent walkthrough of the Ruby gem creation process, see <http://timelessrepo.com/making-ruby-gems>

## See what your plugin looks like

Before we dive into the details, open up the plugin file in your favorite text or and take a look.

```
encoding: utf-8
require "logstash/inputs/base"
require "logstash/namespace"
require "stud/interval"
require "socket" # for Socket.gethostname

Add any asciidoc formatted documentation here
Generate a repeating message.
#
This plugin is intended only as an example.

class LogStash::Inputs::Example < LogStash::Inputs::Base
 config_name "example"

 # If undefined, Logstash will complain, even if codec is unused.
 default :codec, "plain"

 # The message string to use in the event.
 config :message, :validate => :string, :default => "Hello World!"

 # Set how frequently messages should be sent.
 #
 # The default, `1`, means send a message every second.
 config :interval, :validate => :number, :default => 1

 public
 def register
 @host = Socket.gethostname
 end # def register

 def run(queue)
 Stud.interval(@interval) do
 event = LogStash::Event.new("message" => @message, "host" => @host)
 decorate(event)
 queue << event
 end # loop
 end # def run
```

```
end # class Logstash::Inputs::Example
```

## Coding input plugins

Now let's take a line-by-line look at the example plugin.

### **encoding**

It seems like a small thing, but remember to specify the encoding at the beginning of your plugin code:

```
encoding: utf-8
```

Logstash depends on things being in UTF-8, so we put this here to tell the Ruby interpreter that we're going to be using the UTF-8 encoding.

### **require Statements**

Logstash input plugins require parent classes defined in `logstash/inputs/base` and `logstash/namespac`e:

```
require "logstash/inputs/base"
require "logstash/namespac
```

Of course, the plugin you build may depend on other code, or even gems. Just put them here along with these Logstash dependencies.

## Plugin Body

Let's go through the various elements of the plugin itself.

## Inline Documentation

Logstash provides infrastructure to automatically generate documentation for plugins. We use the asciidoc format to write documentation so *any* comments in the source code will be first converted into asciidoc and then into html.

All plugin documentation is then rendered and placed in [the Logstash section of the Elasticsearch Guide](#).

The inline documentation can include code blocks and config examples! To include Ruby code, use the asciidoc `[source, ruby]` directive:

```
Using hashes:
[source, ruby]

match => {
```

```
"field1" => "value1"
"field2" => "value2"
...
}

```

In the rendered HTML document, this block would look like:

Using hashes:

```
match => {
 "field1" => "value1"
 "field2" => "value2"
 ...
}
```

Tip: For more asciidoc formatting tips, see the excellent reference at  
<https://github.com/elastic/docs#asciidoc-guide>

## class Declaration

The input plugin class should be a subclass of `LogStash::Inputs::Base`:

```
class LogStash::Inputs::Example < LogStash::Inputs::Base
```

The class name should closely mirror the plugin name, for example:

```
LogStash::Inputs::Example

config_name

 config_name "example"
```

This is the name your plugin will call inside the input configuration block.

If you set `config_name "example"` in your plugin code, the corresponding Logstash configuration block would need to look like this:

```
input {
 example {...}
}
```

## Configuration Parameters

```
config :variable_name, :validate => :variable_type, :default => "Default
value", :required => boolean, :deprecated => boolean, :obsolete => string
```

The configuration, or `config` section allows you to define as many (or as few) parameters as are needed to enable Logstash to process events.

There are several configuration attributes:

- `:validate` - allows you to enforce passing a particular data type to Logstash for this configuration option, such as `:string`, `:password`, `:boolean`, `:number`, `:array`, `:hash`, `:path` (a file-system path), `uri` (starting in 5.0.0), `:codec` (since 1.2.0), `:bytes` (starting in 1.5.0). Note that this also works as a coercion in that if I specify "true" for boolean (even though technically a string), it will become a valid boolean in the config. This coercion works for the `:number` type as well where "1.2" becomes a float and "22" is an integer.
- `:default` - lets you specify a default value for a parameter
- `:required` - whether or not this parameter is mandatory (a Boolean `true` or `false`)
- `:list` - whether or not this value should be a list of values. Will typecheck the list members, and convert scalars to one element lists. Note that this mostly obviates the array type, though if you need lists of complex objects that will be more suitable.
- `:deprecated` - informational (also a Boolean `true` or `false`)
- `:obsolete` - used to declare that a given setting has been removed and is no longer functioning. The idea is to provide an informed upgrade path to users who are still using a now-removed setting.

## Plugin Methods

Logstash inputs must implement two main methods: `register` and `run`.

### `register` Method

```
public
def register
end # def register
```

The Logstash `register` method is like an `initialize` method. It was originally created to enforce having `super` called, preventing headaches for newbies. (Note: It may go away in favor of `initialize`, in conjunction with some enforced testing to ensure `super` is called.)

`public` means the method can be called anywhere, not just within the class. This is the default behavior for methods in Ruby, but it is specified explicitly here anyway.

You can also assign instance variables here (variables prepended by `@`). Configuration variables are now in scope as instance variables, like `@message`

## run Method

The example input plugin has the following `run` Method:

```
def run(queue)
 Stud.interval(@interval) do
 event = LogStash::Event.new("message" => @message, "host" => @host)
 decorate(event)
 queue << event
 end # loop
end # def run
```

The `run` method is where a stream of data from an input becomes an event.

The stream can be plain or generated as with the [heartbeat](#) input plugin. In these cases, though no codec is used, [a default codec](#) must be set in the code to avoid errors.

Here's another example `run` method:

```
def run(queue)
 while true
 begin
 # Based on some testing, there is no way to interrupt an IO.sysread
 nor
 # IO.select call in JRuby.
 data = $stdin.sysread(16384)
 @codec.decode(data) do |event|
 decorate(event)
 event.set("host", @host) if !event.include?("host")
 queue << event
 end
 rescue IOError, EOFError, LogStash::ShutdownSignal
 # stdin closed or a requested shutdown
 break
 end
 end # while true
 finished
end # def run
```

In this example, the `data` is being sent to the codec defined in the configuration block to `decode` the data stream and return an event.

In both examples, the resulting `event` is passed to the `decorate` method:

```
decorate(event)
```

This applies any tags you might have set in the input configuration block. For example, `tags => ["tag1", "tag2"]`.

Also in both examples, the `event`, after being "decorated," is appended to the queue:

```
queue << event
```

This inserts the event into the pipeline.

Tip: Because input plugins can range from simple to complex, it is helpful to see more examples of how they have been created:

[syslog](#)

[zeromq](#)

[stdin](#)

[tcp](#)

There are many more examples in the [logstash-plugin github repository](#).

## Building the Plugin

At this point in the process you have coded your plugin and are ready to build a Ruby Gem from it. The following steps will help you complete the process.

## External dependencies

A `require` statement in Ruby is used to include necessary code. In some cases your plugin may require additional files. For example, the `collectd` plugin [uses](#) the `types.db` file provided by `collectd`. In the main directory of your plugin, a file called `vendor.json` is where these files are described.

The `vendor.json` file contains an array of JSON objects, each describing a file dependency. This example comes from the [collectd](#) codec plugin:

```
[{
 "sha1": "a90fe6cc53b76b7bdd56dc57950d90787cb9c96e",
 "url": "http://collectd.org/files/collectd-5.4.0.tar.gz",
 "files": ["/src/types.db"]
}
```

`sha1` is the `sha1` signature used to verify the integrity of the file referenced by `url`.

`url` is the address from where Logstash will download the file.

`files` is an optional array of files to extract from the downloaded file. Note that while tar archives can use absolute or relative paths, treat them as absolute in this array. If `files` is not present, all files will be uncompressed and extracted into the `vendor` directory.

Another example of the `vendor.json` file is the [geoip filter](#)

The process used to download these dependencies is to call `rake vendor`. This will be discussed further in the testing section of this document.

Another kind of external dependency is on jar files. This will be described in the "Add a `gemspec` file" section.

## Add a Gemfile

Gemfiles allow Ruby's Bundler to maintain the dependencies for your plugin. Currently, all we'll need is the Logstash gem, for testing, but if you require other gems, you should add them in here.

Tip: See [Bundler's Gemfile page](#) for more details.

```
source 'https://rubygems.org'
gemspec
gem "logstash", :github => "elastic/logstash", :branch => "5.2"
```

## Add a `gemspec` file

Gemspecs define the Ruby gem which will be built and contain your plugin.

Tip: More information can be found on the [Rubygems Specification page](#).

```
Gem::Specification.new do |s|
 s.name = 'logstash-input-example'
 s.version = '0.1.0'
 s.licenses = ['Apache License (2.0)']
 s.summary = "This input does x, y, z in Logstash"
 s.description = "This gem is a logstash plugin required to be installed on top of the Logstash core pipeline using $LS_HOME/bin/logstash-plugin install gemname. This gem is not a stand-alone program"
 s.authors = ["Elastic"]
 s.email = 'info@elastic.co'
 s.homepage = "http://www.elastic.co/guide/en/logstash/current/index.html"
 s.require_paths = ["lib"]

 # Files
 s.files =
Dir['lib/**/*', 'spec/**/*', 'vendor/**/*', '*.gemspec', '*.md', 'CONTRIBUTORS', 'Gemfile', 'LICENSE', 'NOTICE.TXT']
 # Tests
 s.test_files = s.files.grep(%r{^(test|spec|features)/})>

 # Special flag to let us know this is actually a logstash plugin
 s.metadata = { "logstash_plugin" => "true", "logstash_group" => "input" }

 # Gem dependencies
 s.add_runtime_dependency "logstash-core-plugin-api", ">= 1.60", "<= 2.99"
 s.add_development_dependency 'logstash-devutils'
end
```

It is appropriate to change these values to fit your plugin. In particular, `s.name` and `s.summary` shoud reflect your plugin's name and behavior.

`s.licenses` and `s.version` are also important and will come into play when you are ready to publish your plugin.

Logstash and all its plugins are licensed under [Apache License, version 2 \("ALv2"\)](#). If you make your plugin publicly available via [RubyGems.org](#), please make sure to have this line in your gemspec:

```
s.licenses = ['Apache License (2.0)']
```

The gem version, designated by `s.version`, helps track changes to plugins over time. You should use [semver versioning](#) strategy for version numbers.

## Runtime & Development Dependencies

At the bottom of the `gemspec` file is a section with a comment: `Gem dependencies`. This is where any other needed gems must be mentioned. If a gem is necessary for your plugin to function, it is a runtime dependency. If a gem are only used for testing, then it would be a development dependency.

NOTE: You can also have versioning requirements for your dependencies—including other Logstash plugins:

```
Gem dependencies
s.add_runtime_dependency "logstash-core-plugin-api", ">= 1.60", "<= 2.99"
s.add_development_dependency 'logstash-devutils'
```

This gemspec has a runtime dependency on the `logstash-core-plugin-api` and requires that it have a version number greater than or equal to version 1.60 and less than or equal to 2.99.

IMPORTANT!!! All plugins have a runtime dependency on the `logstash-core-plugin-api` gem, and a development dependency on `logstash-devutils`.

## Jar dependencies

In some cases, such as the [Elasticsearch output plugin](#), your code may depend on a jar file. In cases such as this, the dependency is added in the gemspec file in this manner:

```
Jar dependencies
s.requirements << "jar 'org.elasticsearch:elasticsearch', '5.0.0'"
s.add_runtime_dependency 'jar-dependencies'
```

With these both defined, the install process will search for the required jar file at <http://mvnrepository.com> and download the specified version.

## Add Tests

Logstash loves tests. Lots of tests. If you're using your new input plugin in a production environment, you'll want to have some tests to ensure you are not breaking any existing functionality.

NOTE: A full exposition on RSpec is outside the scope of this document. Learn more about RSpec at <http://rspec.info>

For help learning about tests and testing, look in the `spec/inputs/` directory of several other similar plugins.

### Clone and test!

Now let's start with a fresh clone of the plugin, build it and run the tests.

**Clone your plugin into a temporary location** Replace `GITUSERNAME` with your github username, and `MYPLUGINNAME` with your plugin name.

```
git clone https://github.com/GITUSERNAME/logstash-input-MYPLUGINNAME.git
alternately, via ssh: git clone git@github.com:GITUSERNAME/logstash-input-MYPLUGINNAME.git
cd logstash-input-MYPLUGINNAME
```

Then, you'll need to install your plugins dependencies with bundler:

```
bundle install
```

**IMPORTANT:** If your plugin has an external file dependency described in `vendor.json`, you must download that dependency before running or testing. You can do this by running:

```
rake vendor
```

And finally, run the tests:

```
bundle exec rspec
```

You should see a success message, which looks something like this:

```
Finished in 0.034 seconds
1 example, 0 failures
```

Hooray! You're almost there! (Unless you saw failures... you should fix those first).

## Building and Testing

Now you're ready to build your (well-tested) plugin into a Ruby gem.

### Build

You already have all the necessary ingredients, so let's go ahead and run the build command:

```
gem build logstash-input-example.gemspec
```

That's it! Your gem should be built and be in the same path with the name

```
logstash-input-mypluginname-0.1.0.gem
```

The `s.version` number from your `gemspec` file will provide the gem version, in this case, `0.1.0`.

### Test installation

You should test install your plugin into a clean installation of Logstash. Download the latest version from the [Logstash downloads page](#).

Untar and cd in to the directory:

```
curl -O https://download.elastic.co/logstash/logstash/logstash-
 5.2.1.tar.gz
tar xzvf logstash-5.2.1.tar.gz
cd logstash-5.2.1
```

Using the plugin tool, we can install the gem we just built.

Replace `/my/logstash/plugins` with the correct path to the gem for your environment, and `0.1.0` with the correct version number from the `gemspec` file.

```
bin/logstash-plugin install /my/logstash/plugins/logstash-input-
example/logstash-input-example-0.1.0.gem
```

After running this, you should see feedback from Logstash that it was successfully installed:

```
validating /my/logstash/plugins/logstash-input-example/logstash-
 input-example-0.1.0.gem >= 0
Valid logstash plugin. Continuing...
Successfully installed 'logstash-input-example' with version
'0.1.0'
```

Tip: You can also use the Logstash plugin tool to determine which plugins are currently available:

```
bin/logstash-plugin list
```

Depending on what you have installed, you might see a short or long list of plugins: inputs, codecs, filters and outputs.

Now try running Logstash with a simple configuration passed in via the command-line, using the `-e` flag.

NOTE: Your results will depend on what your input plugin is designed to do.

```
bin/logstash -e 'input { example{} } output { stdout { codec => rubydebug } }'
```

The example input plugin will send the contents of `message` (with a default message of "Hello World!") every second.

```
{
 "message" => "Hello World!",
 "@version" => "1",
 "@timestamp" => "2015-01-27T19:17:18.932Z",
 "host" => "cadenza"
}
```

Feel free to experiment and test this by changing the `message` and `interval` parameters:

```
bin/logstash -e 'input { example{ message => "A different message" interval => 5 } } output { stdout { codec => rubydebug } }'
```

Congratulations! You've built, deployed and successfully run a Logstash input.

## Submitting your plugin to [RubyGems.org](#) and [logstash-plugins](#)

Logstash uses [RubyGems.org](#) as its repository for all plugin artifacts. Once you have developed your new plugin, you can make it available to Logstash users by simply publishing it to RubyGems.org.

## Licensing

Logstash and all its plugins are licensed under [Apache License, version 2 \("ALv2"\)](#). If you make your plugin publicly available via [RubyGems.org](#), please make sure to have this line in your gemspec:

```
s.licenses = ['Apache License (2.0)']
```

## Publishing to [RubyGems.org](#)

To begin, you'll need an account on RubyGems.org

[Sign-up for a RubyGems account.](#)

After creating an account, [obtain](#) an API key from RubyGems.org. By default, RubyGems uses the file `~/.gem/credentials` to store your API key. These credentials will be used to publish the gem. Replace `username` and `password` with the credentials you created at RubyGems.org:

```
curl -u username:password https://rubygems.org/api/v1/api_key.yaml >
~/.gem/credentials
chmod 0600 ~/.gem/credentials
```

Before proceeding, make sure you have the right version in your `gemspec` file and commit your changes.

```
s.version = '0.1.0'
```

To publish version 0.1.0 of your new logstash gem:

```
bundle install
bundle exec rake vendor
bundle exec rspec
bundle exec rake publish_gem
```

Note: Executing `rake publish_gem`:

Reads the version from the `gemspec` file (`s.version = '0.1.0'`)

Checks in your local repository if a tag exists for that version. If the tag already exists, it aborts the process. Otherwise, it creates a new version tag in your local repository.

Builds the gem

Publishes the gem to RubyGems.org

That's it! Your plugin is published! Logstash users can now install your plugin by running:

```
bin/logstash-plugin install logstash-input-mypluginname
```

## Contributing your source code to [logstash-plugins](#)

It is not required to contribute your source code to [logstash-plugins](#) github organization, but we always welcome new plugins!

## Benefits

Some of the many benefits of having your plugin in the `logstash-plugins` repository are:

**Discovery** Your plugin will appear in the [Logstash Reference](#), where Logstash users look first for plugins and documentation.

**Documentation** Your plugin documentation will automatically be added to the [Logstash Reference](#).

**Testing** With our testing infrastructure, your plugin will be continuously tested against current and future releases of Logstash. As a result, users will have the assurance that if incompatibilities arise, they will be quickly discovered and corrected.

## Acceptance Guidelines

**Code Review** Your plugin must be reviewed by members of the community for coherence, quality, readability, stability and security.

**Tests** Your plugin must contain tests to be accepted. These tests are also subject to code review for scope and completeness. It's ok if you don't know how to write tests — we will guide you. We are working on publishing a guide to creating tests for Logstash which will make it easier. In the meantime, you can refer to <http://betterspecs.org/> for examples.

To begin migrating your plugin to logstash-plugins, simply create a new [issue](#) in the Logstash repository. When the acceptance guidelines are completed, we will facilitate the move to the logstash-plugins organization using the recommended [github process](#).

# How to write a Logstash codec plugin

To develop a new codec for Logstash, you build a self-contained Ruby gem whose source code lives in its own GitHub repository. The Ruby gem can then be hosted and shared on RubyGems.org. You can use the example codec implementation as a starting point. (If you're unfamiliar with Ruby, you can find an excellent quickstart guide at <https://www.ruby-lang.org/en/documentation/quickstart/>.)

## Get started

Let's step through creating a codec plugin using the [example codec plugin](#).

### Create a GitHub repo for your new plugin

Each Logstash plugin lives in its own GitHub repository. To create a new repository for your plugin:

Log in to GitHub.

Click the **Repositories** tab. You'll see a list of other repositories you've forked or contributed to.

Click the green **New** button in the upper right.

Specify the following settings for your new repo:

**Repository name** — a unique name of the form `logstash-codec-pluginname`.

**Public or Private** — your choice, but the repository must be Public if you want to submit it as an official plugin.

**Initialize this repository with a README** — enables you to immediately clone the repository to your computer.

Click **Create Repository**.

## Use the plugin generator tool

You can now create your own Logstash plugin in seconds! The `generate` subcommand of `bin/logstash-plugin` creates the foundation for a new Logstash plugin with templated files. It creates the correct directory structure, gemspec files, and dependencies so you can start adding custom code to process data with Logstash.

For more information, see [Generating Plugins](#)

## Copy the codec code

Alternatively, you can use the examples repo we host on github.com

**Clone your plugin.** Replace `GITUSERNAME` with your github username, and `MYPLUGINNAME` with your plugin name.

```
git clone https://github.com/GITUSERNAME/logstash-codec-MYPLUGINNAME.git
```

```
alternately, via ssh: git clone git@github.com:GITUSERNAME/logstash-codec-MYPLUGINNAME.git
```

```
cd logstash-codec-MYPLUGINNAME
```

You don't want to include the example .git directory or its contents, so delete it before you copy the example.

```
cd /tmp
```

```
git clone https://github.com/logstash-plugins/logstash-codec-example.git
```

```
cd logstash-codec-example
```

```
rm -rf .git
```

```
cp -R * /path/to/logstash-codec-mypluginname/
```

**Rename the following files to match the name of your plugin.**

```
logstash-codec-example.gemspec
```

```
example.rb
```

```
example_spec.rb
```

```
cd /path/to/logstash-codec-mypluginname
mv logstash-codec-example.gemspec logstash-codec-mypluginname.gemspec
mv lib/logstash/codecs/example.rb
lib/logstash/codecs/mypluginname.rb
mv spec/codecs/example_spec.rb spec/codecs/mypluginname_spec.rb
```

Your file structure should look like this:

```
$ tree logstash-codec-mypluginname
└── Gemfile
└── LICENSE
└── README.md
```

```
└── Rakefile
└── lib
 └── logstash
 └── codecs
 └── mypluginname.rb
└── logstash-codec-mypluginname.gemspec
└── spec
 └── codecs
 └── mypluginname_spec.rb
```

For more information about the Ruby gem file structure and an excellent walkthrough of the Ruby gem creation process, see <http://timelessrepo.com/making-ruby-gems>

## See what your plugin looks like

Before we dive into the details, open up the plugin file in your favorite text or and take a look.

```
encoding: utf-8
require "logstash/codecs/base"
require "logstash/codecs/line"

Add any asciidoc formatted documentation here
class LogStash::Codecs::Example < LogStash::Codecs::Base

 # This example codec will append a string to the message field
 # of an event, either in the decoding or encoding methods
 #
 # This is only intended to be used as an example.
 #
 # input {
 # stdin { codec => example }
 # }
 #
 # or
 #
 # output {
 # stdout { codec => example }
 # }
 config_name "example"

 # Append a string to the message
 config :append, :validate => :string, :default => ', Hello World!'

 public
 def register
 @lines = LogStash::Codecs::Line.new
 @lines.charset = "UTF-8"
 end

 public
 def decode(data)
 @lines.decode(data) do |line|
 replace = { "message" => line["message"].to_s + @append }
 yield LogStash::Event.new(replace)
 end
 end
```

```
end # def decode

public
def encode(event)
 @on_event.call(event, event.get("message").to_s + @append + NL)
end # def encode

end # class Logstash::Codecs::Example
```

## Coding codec plugins

Now let's take a line-by-line look at the example plugin.

### **encoding**

It seems like a small thing, but remember to specify the encoding at the beginning of your plugin code:

```
encoding: utf-8
```

Logstash depends on things being in UTF-8, so we put this here to tell the Ruby interpreter that we're going to be using the UTF-8 encoding.

### **require Statements**

Logstash codec plugins require parent classes defined in `logstash/codecs/base` and `logstash/namespac`:

```
require "logstash/codecs/base"
require "logstash/namespac"
```

Of course, the plugin you build may depend on other code, or even gems. Just put them here along with these Logstash dependencies.

## Plugin Body

Let's go through the various elements of the plugin itself.

## Inline Documentation

Logstash provides infrastructure to automatically generate documentation for plugins. We use the asciidoc format to write documentation so *any* comments in the source code will be first converted into asciidoc and then into html.

All plugin documentation is then rendered and placed in [the Logstash section of the Elasticsearch Guide](#).

The inline documentation can include code blocks and config examples! To include Ruby code, use the asciidoc [source, ruby] directive:

```
Using hashes:
[source, ruby]

match => {
"field1" => "value1"
"field2" => "value2"
...
}

```

In the rendered HTML document, this block would look like:

Using hashes:

```
match => {
 "field1" => "value1"
 "field2" => "value2"
 ...
}
```

Tip: For more asciidoc formatting tips, see the excellent reference at <https://github.com/elastic/docs#asciidoc-guide>

## class Declaration

The codec plugin class should be a subclass of `LogStash::Codecs::Base`:

```
class Logstash::Codecs::Example < Logstash::Codecs::Base
```

The class name should closely mirror the plugin name, for example:

```
Logstash::Codecs::Example
```

```
config_name
```

```
 config_name "example"
```

This is the name your plugin will call inside the codec configuration block.

If you set `config_name "example"` in your plugin code, the corresponding Logstash configuration block would need to look like this:

## Configuration Parameters

```
config :variable_name, :validate => :variable_type, :default => "Default
value", :required => boolean, :deprecated => boolean, :obsolete => string
```

The configuration, or `config` section allows you to define as many (or as few) parameters as are needed to enable Logstash to process events.

There are several configuration attributes:

- `:validate` - allows you to enforce passing a particular data type to Logstash for this configuration option, such as `:string`, `:password`, `:boolean`, `:number`, `:array`, `:hash`, `:path` (a file-system path), `uri` (starting in 5.0.0), `:codec` (since 1.2.0), `:bytes` (starting in 1.5.0). Note that this also works as a coercion in that if I specify "true" for boolean (even though technically a string), it will become a valid boolean in the config. This coercion works for the `:number` type as well where "1.2" becomes a float and "22" is an integer.
- `:default` - lets you specify a default value for a parameter
- `:required` - whether or not this parameter is mandatory (a Boolean `true` or `false`)
- `:list` - whether or not this value should be a list of values. Will typecheck the list members, and convert scalars to one element lists. Note that this mostly obviates the array type, though if you need lists of complex objects that will be more suitable.
- `:deprecated` - informational (also a Boolean `true` or `false`)
- `:obsolete` - used to declare that a given setting has been removed and is no longer functioning. The idea is to provide an informed upgrade path to users who are still using a now-removed setting.

## Plugin Methods

Logstash codecs must implement the `register` method, and the `decode` method or the `encode` method (or both).

### `register` Method

```
public
def register
end # def register
```

The Logstash `register` method is like an `initialize` method. It was originally created to enforce having `super` called, preventing headaches for newbies. (Note: It may go away in favor of `initialize`, in conjunction with some enforced testing to ensure `super` is called.)

`public` means the method can be called anywhere, not just within the class. This is the default behavior for methods in Ruby, but it is specified explicitly here anyway.

You can also assign instance variables here (variables prepended by @). Configuration variables are now in scope as instance variables, like @message

## decode Method

```
public
def decode(data)
 @lines.decode(data) do |line|
 replace = { "message" => line["message"].to_s + @append }
 yield LogStash::Event.new(replace)
 end
end # def decode
```

The codec's decode method is where data coming in from an input is transformed into an event. There are complex examples like the [collectd](#) codec, and simpler examples like the [spool](#) codec.

There must be a `yield` statement as part of the `decode` method which will return decoded events to the pipeline.

## encode Method

```
public
def encode(event)
 @on_event.call(event, event.get("message").to_s + @append + NL)
end # def encode
```

The `encode` method takes an event and serializes it (*encodes*) into another format. Good examples of `encode` methods include the simple [plain](#) codec, the slightly more involved [msgpack](#) codec, and even an [avro](#) codec.

In most cases, your `encode` method should have an `@on_event.call()` statement. This call will output data per event in the described way.

## Building the Plugin

At this point in the process you have coded your plugin and are ready to build a Ruby Gem from it. The following steps will help you complete the process.

## External dependencies

A `require` statement in Ruby is used to include necessary code. In some cases your plugin may require additional files. For example, the collectd plugin [uses](#) the `types.db` file provided by collectd. In the main directory of your plugin, a file called `vendor.json` is where these files are described.

The `vendor.json` file contains an array of JSON objects, each describing a file dependency. This example comes from the [collectd](#) codec plugin:

```
[{
 "sha1": "a90fe6cc53b76b7bdd56dc57950d90787cb9c96e",
 "url": "http://collectd.org/files/collectd-5.4.0.tar.gz",
 "files": ["/src/types.db"]
}]
```

`sha1` is the `sha1` signature used to verify the integrity of the file referenced by `url`.

`url` is the address from where Logstash will download the file.

`files` is an optional array of files to extract from the downloaded file. Note that while tar archives can use absolute or relative paths, treat them as absolute in this array. If `files` is not present, all files will be uncompressed and extracted into the vendor directory.

Another example of the `vendor.json` file is the [geoip filter](#)

The process used to download these dependencies is to call `rake vendor`. This will be discussed further in the testing section of this document.

Another kind of external dependency is on jar files. This will be described in the "Add a `gemspec` file" section.

## Add a Gemfile

Gemfiles allow Ruby's Bundler to maintain the dependencies for your plugin. Currently, all we'll need is the Logstash gem, for testing, but if you require other gems, you should add them in here.

Tip: See [Bundler's Gemfile page](#) for more details.

```
source 'https://rubygems.org'
gemspec
gem "logstash", :github => "elastic/logstash", :branch => "5.2"
```

## Add a `gemspec` file

Gemspecs define the Ruby gem which will be built and contain your plugin.

Tip: More information can be found on the [Rubygems Specification page](#).

```
Gem::Specification.new do |s|
 s.name = 'logstash-codec-example'
 s.version = '0.1.0'
 s.licenses = ['Apache License (2.0)']
 s.summary = "This codec does x, y, z in Logstash"
 s.description = "This gem is a logstash plugin required to be installed on top of the Logstash core pipeline using $LS_HOME/bin/logstash-plugin install gemname. This gem is not a stand-alone program"
 s.authors = ["Elastic"]
```

```
s.email = 'info@elastic.co'
s.homepage = "http://www.elastic.co/guide/en/logstash/current/index.html"
s.require_paths = ["lib"]

Files
s.files =
Dir['lib/**/*','spec/**/*','vendor/**/*','*.gemspec','*.md','CONTRIBUTORS','Gemfile','LICENSE','NOTICE.TXT']
Tests
s.test_files = s.files.grep(%r{^(test|spec|features)/})

Special flag to let us know this is actually a logstash plugin
s.metadata = { "logstash_plugin" => "true", "logstash_group" => "codec" }

Gem dependencies
s.add_runtime_dependency "logstash-core-plugin-api", ">= 1.60", "<= 2.99"
s.add_development_dependency 'logstash-devutils'
end
```

It is appropriate to change these values to fit your plugin. In particular, `s.name` and `s.summary` shoud reflect your plugin's name and behavior.

`s.licenses` and `s.version` are also important and will come into play when you are ready to publish your plugin.

Logstash and all its plugins are licensed under [Apache License, version 2 \("ALv2"\)](#). If you make your plugin publicly available via [RubyGems.org](#), please make sure to have this line in your gemspec:

```
s.licenses = ['Apache License (2.0)']
```

The gem version, designated by `s.version`, helps track changes to plugins over time. You should use [semver versioning](#) strategy for version numbers.

## Runtime & Development Dependencies

At the bottom of the `gemspec` file is a section with a comment: `Gem dependencies`. This is where any other needed gems must be mentioned. If a gem is necessary for your plugin to function, it is a runtime dependency. If a gem are only used for testing, then it would be a development dependency.

You can also have versioning requirements for your dependencies—including other Logstash plugins:

```
Gem dependencies
s.add_runtime_dependency "logstash-core-plugin-api", ">= 1.60", "<= 2.99"
s.add_development_dependency 'logstash-devutils'
```

This gemspec has a runtime dependency on the logstash-core-plugin-api and requires that it have a version number greater than or equal to version 1.60 and less than or equal to version 2.99.

**IMPORTANT:** All plugins have a runtime dependency on the `logstash-core-plugin-api` gem, and a development dependency on `logstash-devutils`.

## Jar dependencies

In some cases, such as the [Elasticsearch output plugin](#), your code may depend on a jar file. In cases such as this, the dependency is added in the gemspec file in this manner:

```
Jar dependencies
s.requirements << "jar 'org.elasticsearch:elasticsearch', '5.0.0'"
s.add_runtime_dependency 'jar-dependencies'
```

With these both defined, the install process will search for the required jar file at <http://mvnrepository.com> and download the specified version.

## Add Tests

Logstash loves tests. Lots of tests. If you're using your new codec plugin in a production environment, you'll want to have some tests to ensure you are not breaking any existing functionality.

NOTE: A full exposition on RSpec is outside the scope of this document. Learn more about RSpec at <http://rspec.info>

For help learning about tests and testing, look in the `spec/codecs/` directory of several other similar plugins.

## Clone and test!

Now let's start with a fresh clone of the plugin, build it and run the tests.

**Clone your plugin into a temporary location** Replace `GITUSERNAME` with your github username, and `MYPLUGINNAME` with your plugin name.

```
git clone https://github.com/GITUSERNAME/logstash-codec-
MYPLUGINNAME.git
```

```
alternately, via ssh: git clone git@github.com:GITUSERNAME/logstash-
codec-MYPLUGINNAME.git
```

```
cd logstash-codec-MYPLUGINNAME
```

Then, you'll need to install your plugins dependencies with bundler:

```
bundle install
```

**IMPORTANT:** If your plugin has an external file dependency described in `vendor.json`, you must download that dependency before running or testing. You can do this by running:

```
rake vendor
```

And finally, run the tests:

```
bundle exec rspec
```

You should see a success message, which looks something like this:

```
Finished in 0.034 seconds
1 example, 0 failures
```

Hooray! You're almost there! (Unless you saw failures... you should fix those first).

## Building and Testing

Now you're ready to build your (well-tested) plugin into a Ruby gem.

### Build

You already have all the necessary ingredients, so let's go ahead and run the build command:

```
gem build logstash-codec-example.gemspec
```

That's it! Your gem should be built and be in the same path with the name

```
logstash-codec-mypluginname-0.1.0.gem
```

The `s.version` number from your `gemspec` file will provide the gem version, in this case, `0.1.0`.

### Test installation

You should test install your plugin into a clean installation of Logstash. Download the latest version from the [Logstash downloads page](#).

Untar and cd in to the directory:

```
curl -O https://download.elastic.co/logstash/logstash/logstash-
 5.2.1.tar.gz
tar xzvf logstash-5.2.1.tar.gz
cd logstash-5.2.1
```

Using the plugin tool, we can install the gem we just built.

Replace `/my/logstash/plugins` with the correct path to the gem for your environment, and `0.1.0` with the correct version number from the `gemspec` file.

```
bin/logstash-plugin install /my/logstash/plugins/logstash-codec-example/logstash-codec-example-0.1.0.gem
```

After running this, you should see feedback from Logstash that it was successfully installed:

```
validating /my/logstash/plugins/logstash-codec-example/logstash-codec-example-0.1.0.gem >= 0
Valid logstash plugin. Continuing...
Successfully installed 'logstash-codec-example' with version
'0.1.0'
```

Tip: You can also use the Logstash plugin tool to determine which plugins are currently available:

```
bin/logstash-plugin list
```

Depending on what you have installed, you might see a short or long list of plugins: inputs, codecs, filters and outputs.

Now try running Logstash with a simple configuration passed in via the command-line, using the `-e` flag.

NOTE: Your results will depend on what your codec plugin is designed to do.

```
bin/logstash -e 'input { stdin{ codec => example{}} } output {stdout { codec => rubydebug }}'
```

The example codec plugin will append the contents of `append` (which by default appends ", Hello World!")

After starting Logstash, type something, for example "Random output string". The resulting output message field contents should be, "Random output string, Hello World!":

```
Random output string
{
 "message" => "Random output string, Hello World!",
 "@version" => "1",
 "@timestamp" => "2015-01-27T19:17:18.932Z",
 "host" => "cadenza"
}
```

Feel free to experiment and test this by changing the `append` parameter:

```
bin/logstash -e 'input { stdin{ codec => example{ append => ", I am appending
this! } } output {stdout { codec => rubydebug }}}'
```

Congratulations! You've built, deployed and successfully run a Logstash codec.

## Submitting your plugin to [RubyGems.org](#) and [logstash-plugins](#)

Logstash uses [RubyGems.org](#) as its repository for all plugin artifacts. Once you have developed your new plugin, you can make it available to Logstash users by simply publishing it to RubyGems.org.

### Licensing

Logstash and all its plugins are licensed under [Apache License, version 2 \("ALv2"\)](#). If you make your plugin publicly available via [RubyGems.org](#), please make sure to have this line in your gemspec:

```
s.licenses = ['Apache License (2.0)']
```

### Publishing to [RubyGems.org](#)

To begin, you'll need an account on RubyGems.org

[Sign-up for a RubyGems account.](#)

After creating an account, [obtain](#) an API key from RubyGems.org. By default, RubyGems uses the file `~/.gem/credentials` to store your API key. These credentials will be used to publish the gem. Replace `username` and `password` with the credentials you created at RubyGems.org:

```
curl -u username:password https://rubygems.org/api/v1/api_key.yaml >
~/.gem/credentials
chmod 0600 ~/.gem/credentials
```

Before proceeding, make sure you have the right version in your gemspec file and commit your changes.

```
s.version = '0.1.0'
```

To publish version 0.1.0 of your new logstash gem:

```
bundle install
bundle exec rake vendor
bundle exec rspec
bundle exec rake publish_gem
```

NOTE: Executing `rake publish_gem`:

Reads the version from the gemspec file (`s.version = '0.1.0'`)

Checks in your local repository if a tag exists for that version. If the tag already exists, it aborts the process. Otherwise, it creates a new version tag in your local repository.

Builds the gem

Publishes the gem to RubyGems.org

That's it! Your plugin is published! Logstash users can now install your plugin by running:

```
bin/logstash-plugin install logstash-codec-mypluginname
```

## Contributing your source code to [logstash-plugins](#)

It is not required to contribute your source code to [logstash-plugins](#) github organization, but we always welcome new plugins!

## Benefits

Some of the many benefits of having your plugin in the logstash-plugins repository are:

**Discovery** Your plugin will appear in the [Logstash Reference](#), where Logstash users look first for plugins and documentation.

**Documentation** Your plugin documentation will automatically be added to the [Logstash Reference](#).

**Testing** With our testing infrastructure, your plugin will be continuously tested against current and future releases of Logstash. As a result, users will have the assurance that if incompatibilities arise, they will be quickly discovered and corrected.

## Acceptance Guidelines

**Code Review** Your plugin must be reviewed by members of the community for coherence, quality, readability, stability and security.

**Tests** Your plugin must contain tests to be accepted. These tests are also subject to code review for scope and completeness. It's ok if you don't know how to write tests — we will guide you. We are working on publishing a guide to creating tests for Logstash which will make it easier. In the meantime, you can refer to <http://betterspecs.org/> for examples.

To begin migrating your plugin to logstash-plugins, simply create a new [issue](#) in the Logstash repository. When the acceptance guidelines are completed, we will facilitate the move to the logstash-plugins organization using the recommended [github process](#).

# How to write a Logstash filter plugin

To develop a new filter for Logstash, you build a self-contained Ruby gem whose source code lives in its own GitHub repository. The Ruby gem can then be hosted and shared on RubyGems.org. You can use the example filter implementation as a starting point. (If you're unfamiliar with Ruby, you can find an excellent quickstart guide at <https://www.ruby-lang.org/en/documentation/quickstart/>.)

## Get started

Let's step through creating a filter plugin using the [example filter plugin](#).

### Create a GitHub repo for your new plugin

Each Logstash plugin lives in its own GitHub repository. To create a new repository for your plugin:

Log in to GitHub.

Click the **Repositories** tab. You'll see a list of other repositories you've forked or contributed to.

Click the green **New** button in the upper right.

Specify the following settings for your new repo:

**Repository name** — a unique name of the form `logstash-filter-pluginname`.

**Public or Private** — your choice, but the repository must be Public if you want to submit it as an official plugin.

**Initialize this repository with a README** — enables you to immediately clone the repository to your computer.

Click **Create Repository**.

## Use the plugin generator tool

You can now create your own Logstash plugin in seconds! The `generate` subcommand of `bin/logstash-plugin` creates the foundation for a new Logstash plugin with templated files. It creates the correct directory structure, gemspec files, and dependencies so you can start adding custom code to process data with Logstash.

For more information, see [Generating Plugins](#)

## Copy the filter code

Alternatively, you can use the examples repo we host on github.com

**Clone your plugin.** Replace `GITUSERNAME` with your github username, and `MYPLUGINNAME` with your plugin name.

```
git clone https://github.com/GITUSERNAME/logstash-filter-MYPLUGINNAME.git
```

```
alternately, via ssh: git clone git@github.com:GITUSERNAME/logstash-filter-MYPLUGINNAME.git
```

```
cd logstash-filter-MYPLUGINNAME
```

You don't want to include the example .git directory or its contents, so delete it before you copy the example.

```
cd /tmp
```

```
git clone https://github.com/logstash-plugins/logstash-filter-example.git
```

```
cd logstash-filter-example
```

```
rm -rf .git
```

```
cp -R * /path/to/logstash-filter-mypluginname/
```

**Rename the following files to match the name of your plugin.**

```
logstash-filter-example.gemspec
```

```
example.rb
```

```
example_spec.rb
```

```
cd /path/to/logstash-filter-mypluginname
```

```
mv logstash-filter-example.gemspec logstash-filter-mypluginname.gemspec
```

```
mv lib/logstash/filters/example.rb
```

```
lib/logstash/filters/mypluginname.rb
```

```
mv spec/filters/example_spec.rb spec/filters/mypluginname_spec.rb
```

Your file structure should look like this:

```
$ tree logstash-filter-mypluginname
└── Gemfile
└── LICENSE
└── README.md
```

```
└── Rakefile
└── lib
 └── logstash
 └── filters
 └── mypluginname.rb
└── logstash-filter-mypluginname.gemspec
└── spec
 └── filters
 └── mypluginname_spec.rb
```

For more information about the Ruby gem file structure and an excellent walkthrough of the Ruby gem creation process, see <http://timelesrepo.com/making-ruby-gems>

### See what your plugin looks like

Before we dive into the details, open up the plugin file in your favorite text or and take a look.

```
encoding: utf-8
require "logstash/filters/base"
require "logstash/namespacedoc"

Add any asciidoc formatted documentation here
This example filter will replace the contents of the default
message field with whatever you specify in the configuration.
#
It is only intended to be used as an example.
class LogStash::Filters::Example < LogStash::Filters::Base

 # Setting the config_name here is required. This is how you
 # configure this filter from your Logstash config.
 #
 # filter {
 # example { message => "My message..." }
 # }
 config_name "example"

 # Replace the message with this value.
 config :message, :validate => :string, :default => "Hello World!"

public
def register
 # Add instance variables
end # def register

public
def filter(event)

 if @message
 # Replace the event message with our message as configured in the
 # config file.
 event.set("message", @message)
 end

 # filter_matched should go in the last line of our successful code
```

```
 filter_matched(event)
end # def filter

end # class Logstash::Filters::Example
```

## Coding filter plugins

Now let's take a line-by-line look at the example plugin.

### **encoding**

It seems like a small thing, but remember to specify the encoding at the beginning of your plugin code:

```
encoding: utf-8
```

Logstash depends on things being in UTF-8, so we put this here to tell the Ruby interpreter that we're going to be using the UTF-8 encoding.

### **require Statements**

Logstash filter plugins require parent classes defined in `logstash/filters/base` and `logstash/namespac`:

```
require "logstash/filters/base"
require "logstash/namespac"
```

Of course, the plugin you build may depend on other code, or even gems. Just put them here along with these Logstash dependencies.

## Plugin Body

Let's go through the various elements of the plugin itself.

### Inline Documentation

Logstash provides infrastructure to automatically generate documentation for plugins. We use the asciidoc format to write documentation so *any* comments in the source code will be first converted into asciidoc and then into html.

All plugin documentation is then rendered and placed in [the Logstash section of the Elasticsearch Guide](#).

The inline documentation can include code blocks and config examples! To include Ruby code, use the asciidoc `[source, ruby]` directive:

```
Using hashes:
```

```
[source,ruby]

match => {
"field1" => "value1"
"field2" => "value2"
...
}
```

In the rendered HTML document, this block would look like:

Using hashes:

```
match => {
 "field1" => "value1"
 "field2" => "value2"
 ...
}
```

Tip: For more asciidoc formatting tips, see the excellent reference at  
<https://github.com/elastic/docs#asciidoc-guide>

### **class Declaration**

The filter plugin class should be a subclass of `LogStash::Filters::Base`:

```
class LogStash::Filters::Example < LogStash::Filters::Base
```

The class name should closely mirror the plugin name, for example:

```
LogStash::Filters::Example

config_name

 config_name "example"
```

This is the name your plugin will call inside the filter configuration block.

If you set `config_name "example"` in your plugin code, the corresponding Logstash configuration block would need to look like this:

### **Configuration Parameters**

```
 config :variable_name, :validate => :variable_type, :default => "Default
value", :required => boolean, :deprecated => boolean, :obsolete => string
```

The configuration, or `config` section allows you to define as many (or as few) parameters as are needed to enable Logstash to process events.

There are several configuration attributes:

```
:validate - allows you to enforce passing a particular data type to Logstash for this configuration option, such as :string, :password, :boolean, :number, :array, :hash, :path (a file-system path), uri (starting in 5.0.0), :codec (since 1.2.0), :bytes (starting in 1.5.0). Note that this also works as a coercion in that if I specify "true" for boolean (even though technically a string), it will become a valid boolean in the config. This coercion works for the :number type as well where "1.2" becomes a float and "22" is an integer.

:default - lets you specify a default value for a parameter

:required - whether or not this parameter is mandatory (a Boolean true or false)

:list - whether or not this value should be a list of values. Will typecheck the list members, and convert scalars to one element lists. Note that this mostly obviates the array type, though if you need lists of complex objects that will be more suitable.

:deprecated - informational (also a Boolean true or false)

:obsolete - used to declare that a given setting has been removed and is no longer functioning. The idea is to provide an informed upgrade path to users who are still using a now-removed setting.
```

## Plugin Methods

Logstash filters must implement the `register` and `filter` methods.

### `register` Method

```
public
def register
end # def register
```

The Logstash `register` method is like an `initialize` method. It was originally created to enforce having `super` called, preventing headaches for newbies. (Note: It may go away in favor of `initialize`, in conjunction with some enforced testing to ensure `super` is called.)

`public` means the method can be called anywhere, not just within the class. This is the default behavior for methods in Ruby, but it is specified explicitly here anyway.

You can also assign instance variables here (variables prepended by `@`). Configuration variables are now in scope as instance variables, like `@message`

### `filter` Method

```
public
def filter(event)

 if @message
 # Replace the event message with our message as configured in the
 # config file.
 event.set("message", @message)
 end

 # filter_matched should go in the last line of our successful code
 filter_matched(event)
end # def filter
```

The plugin's `filter` method is where the actual filtering work takes place! Inside the `filter` method you can refer to the event data using the `Event` object. Event is the main object that encapsulates data flow internally in Logstash and provides an [API](#) for the plugin developers to interact with the event's content.

The `filter` method should also handle any [event dependent configuration](#) by explicitly calling the `sprintf` method available in `Event` class. For example:

```
field_foo = event.strftime(field)
```

Note that configuration variables are now in scope as instance variables, like `@message`

```
filter_matched(event)
```

Calling the `filter_matched` method upon successful execution of the plugin will ensure that any fields or tags added through the Logstash configuration for this filter will be handled correctly. For example, any `add_field`, `remove_field`, `add_tag` and/or `remove_tag` actions will be performed at this time.

Event methods such as `event.cancel` are now available to control the workflow of the event being processed.

## Building the Plugin

At this point in the process you have coded your plugin and are ready to build a Ruby Gem from it. The following steps will help you complete the process.

## External dependencies

A `require` statement in Ruby is used to include necessary code. In some cases your plugin may require additional files. For example, the `collectd` plugin [uses](#) the `types.db` file provided by `collectd`. In the main directory of your plugin, a file called `vendor.json` is where these files are described.

The `vendor.json` file contains an array of JSON objects, each describing a file dependency. This example comes from the [collectd](#) codec plugin:

```
[{
 "sha1": "a90fe6cc53b76b7bdd56dc57950d90787cb9c96e",
 "url": "http://collectd.org/files/collectd-5.4.0.tar.gz",
 "files": ["/src/types.db"]
}]
```

`sha1` is the `sha1` signature used to verify the integrity of the file referenced by `url`.

`url` is the address from where Logstash will download the file.

`files` is an optional array of files to extract from the downloaded file. Note that while tar archives can use absolute or relative paths, treat them as absolute in this array. If `files` is not present, all files will be uncompressed and extracted into the vendor directory.

Another example of the `vendor.json` file is the [geoip filter](#)

The process used to download these dependencies is to call `rake vendor`. This will be discussed further in the testing section of this document.

Another kind of external dependency is on jar files. This will be described in the "Add a `gemspec` file" section.

## Add a Gemfile

Gemfiles allow Ruby's Bundler to maintain the dependencies for your plugin. Currently, all we'll need is the Logstash gem, for testing, but if you require other gems, you should add them in here.

Tip: See [Bundler's Gemfile page](#) for more details.

```
source 'https://rubygems.org'
gemspec
gem "logstash", :github => "elastic/logstash", :branch => "5.2"
```

## Add a `gemspec` file

Gemspecs define the Ruby gem which will be built and contain your plugin.

Tip: More information can be found on the [Rubygems Specification page](#).

```
Gem::Specification.new do |s|
 s.name = 'logstash-filter-example'
 s.version = '0.1.0'
 s.licenses = ['Apache License (2.0)']
 s.summary = "This filter does x, y, z in Logstash"
```

```
s.description = "This gem is a logstash plugin required to be installed on
top of the Logstash core pipeline using $LS_HOME/bin/logstash-plugin install
gemname. This gem is not a stand-alone program"
s.authors = ["Elastic"]
s.email = 'info@elastic.co'
s.homepage = "http://www.elastic.co/guide/en/logstash/current/index.html"
s.require_paths = ["lib"]

Files
s.files =
Dir['lib/**/*', 'spec/**/*', 'vendor/**/*', '*.gemspec', '*.md', 'CONTRIBUTORS', 'G
emfile', 'LICENSE', 'NOTICE.TXT']
Tests
s.test_files = s.files.grep(%r{^(test|spec|features)/})

Special flag to let us know this is actually a logstash plugin
s.metadata = { "logstash_plugin" => "true", "logstash_group" => "filter" }

Gem dependencies
s.add_runtime_dependency "logstash-core-plugin-api", ">= 1.60", "<= 2.99"
s.add_development_dependency 'logstash-devutils'
end
```

It is appropriate to change these values to fit your plugin. In particular, `s.name` and `s.summary` shoud reflect your plugin's name and behavior.

`s.licenses` and `s.version` are also important and will come into play when you are ready to publish your plugin.

Logstash and all its plugins are licensed under [Apache License, version 2 \("ALv2"\)](#). If you make your plugin publicly available via [RubyGems.org](#), please make sure to have this line in your `gemspec`:

```
s.licenses = ['Apache License (2.0)']
```

The gem version, designated by `s.version`, helps track changes to plugins over time. You should use [semver versioning](#) strategy for version numbers.

## Runtime & Development Dependencies

At the bottom of the `gemspec` file is a section with a comment: `Gem dependencies`. This is where any other needed gems must be mentioned. If a gem is necessary for your plugin to function, it is a runtime dependency. If a gem are only used for testing, then it would be a development dependency.

NOTE: You can also have versioning requirements for your dependencies—including other Logstash plugins:

```
Gem dependencies
s.add_runtime_dependency "logstash-core-plugin-api", ">= 1.60", "<= 2.99"
s.add_development_dependency 'logstash-devutils'
```

This gemspec has a runtime dependency on the logstash-core-plugin-api and requires that it have a version number greater than or equal to version 1.60 and less than or equal to version 2.99.

**IMPORTANT!!!** All plugins have a runtime dependency on the `logstash-core-plugin-api` gem, and a development dependency on `logstash-devutils`.

## Jar dependencies

In some cases, such as the [Elasticsearch output plugin](#), your code may depend on a jar file. In cases such as this, the dependency is added in the gemspec file in this manner:

```
Jar dependencies
s.requirements << "jar 'org.elasticsearch:elasticsearch', '5.0.0'"
s.add_runtime_dependency 'jar-dependencies'
```

With these both defined, the install process will search for the required jar file at <http://mvnrepository.com> and download the specified version.

## Add Tests

Logstash loves tests. Lots of tests. If you're using your new filter plugin in a production environment, you'll want to have some tests to ensure you are not breaking any existing functionality.

NOTE: A full exposition on RSpec is outside the scope of this document. Learn more about RSpec at <http://rspec.info>

For help learning about tests and testing, look in the `spec/filters/` directory of several other similar plugins.

## Clone and test!

Now let's start with a fresh clone of the plugin, build it and run the tests.

**Clone your plugin into a temporary location** Replace `GITUSERNAME` with your github username, and `MYPLUGINNAME` with your plugin name.

```
git clone https://github.com/GITUSERNAME/logstash-filter-
MYPLUGINNAME.git
```

```
alternately, via ssh: git clone git@github.com:GITUSERNAME/logstash-
filter-MYPLUGINNAME.git
```

```
cd logstash-filter-MYPLUGINNAME
```

Then, you'll need to install your plugins dependencies with bundler:

```
bundle install
```

**IMPORTANT!!!** If your plugin has an external file dependency described in `vendor.json`, you must download that dependency before running or testing. You can do this by running:

```
rake vendor
```

And finally, run the tests:

```
bundle exec rspec
```

You should see a success message, which looks something like this:

```
Finished in 0.034 seconds
1 example, 0 failures
```

Hooray! You're almost there! (Unless you saw failures... you should fix those first).

## Building and Testing

Now you're ready to build your (well-tested) plugin into a Ruby gem.

### Build

You already have all the necessary ingredients, so let's go ahead and run the build command:

```
gem build logstash-filter-example.gemspec
```

That's it! Your gem should be built and be in the same path with the name

```
logstash-filter-mypluginname-0.1.0.gem
```

The `s.version` number from your `gemspec` file will provide the gem version, in this case, `0.1.0`.

### Test installation

You should test install your plugin into a clean installation of Logstash. Download the latest version from the [Logstash downloads page](#).

Untar and cd in to the directory:

```
curl -O https://download.elastic.co/logstash/logstash/logstash-
5.2.1.tar.gz
tar xzvf logstash-5.2.1.tar.gz
cd logstash-5.2.1
```

Using the plugin tool, we can install the gem we just built.

Replace `/my/logstash/plugins` with the correct path to the gem for your environment, and `0.1.0` with the correct version number from the `gemspec` file.

```
bin/logstash-plugin install /my/logstash/plugins/logstash-filter-example/logstash-filter-example-0.1.0.gem
```

After running this, you should see feedback from Logstash that it was successfully installed:

```
validating /my/logstash/plugins/logstash-filter-example/logstash-filter-example-0.1.0.gem >= 0
Valid logstash plugin. Continuing...
Successfully installed 'logstash-filter-example' with version
'0.1.0'
```

Tip: You can also use the Logstash plugin tool to determine which plugins are currently available:

```
bin/logstash-plugin list
```

Depending on what you have installed, you might see a short or long list of plugins: inputs, codecs, filters and outputs.

Now try running Logstash with a simple configuration passed in via the command-line, using the `-e` flag.

NOTE: Your results will depend on what your filter plugin is designed to do.

```
bin/logstash -e 'input { stdin{} } filter { example {} } output { stdout {
codec => rubydebug }}'
```

Test your filter by sending input through `stdin` and output (after filtering) through `stdout` with the `rubydebug` codec, which enhances readability.

In the case of the example filter plugin, any text you send will be replaced by the contents of the `message` configuration parameter, the default value being "Hello World!":

```
Testing 1, 2, 3
{
 "message" => "Hello World!",
 "@version" => "1",
 "@timestamp" => "2015-01-27T19:17:18.932Z",
 "host" => "cadenza"
}
```

Feel free to experiment and test this by changing the `message` parameter:

```
bin/logstash -e 'input { stdin{} } filter { example { message => "This is a
new message!" } } output { stdout { codec => rubydebug }}'
```

Congratulations! You've built, deployed and successfully run a Logstash filter.

## Submitting your plugin to [RubyGems.org](#) and [logstash-plugins](#)

Logstash uses [RubyGems.org](#) as its repository for all plugin artifacts. Once you have developed your new plugin, you can make it available to Logstash users by simply publishing it to RubyGems.org.

### Licensing

Logstash and all its plugins are licensed under [Apache License, version 2 \("ALv2"\)](#). If you make your plugin publicly available via [RubyGems.org](#), please make sure to have this line in your gemspec:

```
s.licenses = ['Apache License (2.0)']
```

### Publishing to [RubyGems.org](#)

To begin, you'll need an account on RubyGems.org

[Sign-up for a RubyGems account.](#)

After creating an account, [obtain](#) an API key from RubyGems.org. By default, RubyGems uses the file `~/.gem/credentials` to store your API key. These credentials will be used to publish the gem. Replace `username` and `password` with the credentials you created at RubyGems.org:

```
curl -u username:password https://rubygems.org/api/v1/api_key.yaml >
~/.gem/credentials
chmod 0600 ~/.gem/credentials
```

Before proceeding, make sure you have the right version in your gemspec file and commit your changes.

```
s.version = '0.1.0'
```

To publish version 0.1.0 of your new logstash gem:

```
bundle install
bundle exec rake vendor
bundle exec rspec
bundle exec rake publish_gem
```

NOTE: Executing `rake publish_gem`:

Reads the version from the gemspec file (`s.version = '0.1.0'`)

Checks in your local repository if a tag exists for that version. If the tag already exists, it aborts the process. Otherwise, it creates a new version tag in your local repository.

Builds the gem

Publishes the gem to RubyGems.org

That's it! Your plugin is published! Logstash users can now install your plugin by running:

```
bin/logstash-plugin install logstash-filter-mypluginname
```

## Contributing your source code to [logstash-plugins](#)

It is not required to contribute your source code to [logstash-plugins](#) github organization, but we always welcome new plugins!

## Benefits

Some of the many benefits of having your plugin in the logstash-plugins repository are:

**Discovery** Your plugin will appear in the [Logstash Reference](#), where Logstash users look first for plugins and documentation.

**Documentation** Your plugin documentation will automatically be added to the [Logstash Reference](#).

**Testing** With our testing infrastructure, your plugin will be continuously tested against current and future releases of Logstash. As a result, users will have the assurance that if incompatibilities arise, they will be quickly discovered and corrected.

## Acceptance Guidelines

**Code Review** Your plugin must be reviewed by members of the community for coherence, quality, readability, stability and security.

**Tests** Your plugin must contain tests to be accepted. These tests are also subject to code review for scope and completeness. It's ok if you don't know how to write tests — we will guide you. We are working on publishing a guide to creating tests for Logstash which will make it easier. In the meantime, you can refer to <http://betterspecs.org/> for examples.

To begin migrating your plugin to logstash-plugins, simply create a new [issue](#) in the Logstash repository. When the acceptance guidelines are completed, we will facilitate the move to the logstash-plugins organization using the recommended [github process](#).

# How to write a Logstash output plugin

To develop a new output for Logstash, you build a self-contained Ruby gem whose source code lives in its own GitHub repository. The Ruby gem can then be hosted and shared on RubyGems.org. You can use the example output implementation as a starting point. (If you’re unfamiliar with Ruby, you can find an excellent quickstart guide at <https://www.ruby-lang.org/en/documentation/quickstart/>.)

## Get started

Let’s step through creating an output plugin using the [example output plugin](#).

### Create a GitHub repo for your new plugin

Each Logstash plugin lives in its own GitHub repository. To create a new repository for your plugin:

Log in to GitHub.

Click the **Repositories** tab. You’ll see a list of other repositories you’ve forked or contributed to.

Click the green **New** button in the upper right.

Specify the following settings for your new repo:

**Repository name** — a unique name of the form `logstash-output-pluginname`.

**Public or Private** — your choice, but the repository must be Public if you want to submit it as an official plugin.

**Initialize this repository with a README** — enables you to immediately clone the repository to your computer.

Click **Create Repository**.

## Use the plugin generator tool

You can now create your own Logstash plugin in seconds! The `generate` subcommand of `bin/logstash-plugin` creates the foundation for a new Logstash plugin with templated files. It creates the correct directory structure, gemspec files, and dependencies so you can start adding custom code to process data with Logstash.

For more information, see [Generating Plugins](#)

## Copy the output code

Alternatively, you can use the examples repo we host on github.com

**Clone your plugin.** Replace `GITUSERNAME` with your github username, and `MYPLUGINNAME` with your plugin name.

```
git clone https://github.com/GITUSERNAME/logstash-output-MYPLUGINNAME.git
```

```
alternately, via ssh: git clone git@github.com:GITUSERNAME/logstash-output-MYPLUGINNAME.git
```

```
cd logstash-output-MYPLUGINNAME
```

You don't want to include the example .git directory or its contents, so delete it before you copy the example.

```
cd /tmp
```

```
git clone https://github.com/logstash-plugins/logstash-output-example.git
```

```
cd logstash-output-example
```

```
rm -rf .git
```

```
cp -R * /path/to/logstash-output-mypluginname/
```

**Rename the following files to match the name of your plugin.**

```
logstash-output-example.gemspec
```

```
example.rb
```

```
example_spec.rb
```

```
cd /path/to/logstash-output-mypluginname
mv logstash-output-example.gemspec logstash-output-
mypluginname.gemspec
mv lib/logstash/outputs/example.rb
lib/logstash/outputs/mypluginname.rb
mv spec/outputs/example_spec.rb spec/outputs/mypluginname_spec.rb
```

Your file structure should look like this:

```
$ tree logstash-output-mypluginname
└── Gemfile
└── LICENSE
└── README.md
```

```
└── Rakefile
└── lib
 └── logstash
 └── outputs
 └── mypluginname.rb
└── logstash-output-mypluginname.gemspec
└── spec
 └── outputs
 └── mypluginname_spec.rb
```

For more information about the Ruby gem file structure and an excellent walkthrough of the Ruby gem creation process, see <http://timelesrepo.com/making-ruby-gems>

### See what your plugin looks like

Before we dive into the details, open up the plugin file in your favorite text or and take a look.

```
encoding: utf-8
require "logstash/outputs/base"
require "logstash/namespace"

Add any asciidoc formatted documentation here
An example output that does nothing.
class LogStash::Outputs::Example < LogStash::Outputs::Base
 config_name "example"

 # This sets the concurrency behavior of this plugin. By default it is
 :legacy, which was the standard
 # way concurrency worked before Logstash 2.4
 #
 # You should explicitly set it to either :single or :shared as :legacy will
 be removed in Logstash 6.0
 #
 # When configured as :single a single instance of the Output will be shared
 among the
 # pipeline worker threads. Access to the
 `#multi_receive/#multi_receive_encoded/#receive` method will be synchronized
 # i.e. only one thread will be active at a time making threadsafety much
 simpler.
 #
 # You can set this to :shared if your output is threadsafe. This will
 maximize
 # concurrency but you will need to make appropriate uses of mutexes in
 `#multi_receive/#receive`.
 #
 # Only the `#multi_receive/#multi_receive_encoded` methods need to actually
 be threadsafe, the other methods
 # will only be executed in a single thread
 concurrency :single

 public
 def register
 end # def register

 public
```

```
Takes an array of events
Must be threadsafe if `concurrency :shared` is set
def multi_receive(events)
end # def multi_receive
end # class Logstash::Outputs::Example
```

## Coding output plugins

Now let's take a line-by-line look at the example plugin.

### **encoding**

It seems like a small thing, but remember to specify the encoding at the beginning of your plugin code:

```
encoding: utf-8
```

Logstash depends on things being in UTF-8, so we put this here to tell the Ruby interpreter that we're going to be using the UTF-8 encoding.

### **require Statements**

Logstash output plugins require parent classes defined in `logstash/outputs/base` and `logstash/namespac`:

```
require "logstash/outputs/base"
require "logstash/namespac"
```

Of course, the plugin you build may depend on other code, or even gems. Just put them here along with these Logstash dependencies.

## Plugin Body

Let's go through the various elements of the plugin itself.

### **Inline Documentation**

Logstash provides infrastructure to automatically generate documentation for plugins. We use the asciidoc format to write documentation so *any* comments in the source code will be first converted into asciidoc and then into html.

All plugin documentation is then rendered and placed in [the Logstash section of the Elasticsearch Guide](#).

The inline documentation can include code blocks and config examples! To include Ruby code, use the asciidoc `[source, ruby]` directive:

```
Using hashes:
[source,ruby]

match => {
"field1" => "value1"
"field2" => "value2"
...
}

```

In the rendered HTML document, this block would look like:

Using hashes:

```
match => {
 "field1" => "value1"
 "field2" => "value2"
 ...
}
```

Tip: For more asciidoc formatting tips, see the excellent reference at  
<https://github.com/elastic/docs#asciidoc-guide>

### **class Declaration**

The output plugin class should be a subclass of `LogStash::Outputs::Base`:

```
class LogStash::Outputs::Example < LogStash::Outputs::Base
```

The class name should closely mirror the plugin name, for example:

```
LogStash::Outputs::Example
config_name
 config_name "example"
```

This is the name your plugin will call inside the output configuration block.

If you set `config_name "example"` in your plugin code, the corresponding Logstash configuration block would need to look like this:

### **Configuration Parameters**

```
config :variable_name, :validate => :variable_type, :default => "Default
value", :required => boolean, :deprecated => boolean, :obsolete => string
```

The configuration or `config` section allows you to define as many (or as few) parameters as are needed to enable Logstash to process events.

There are several configuration attributes:

```
:validate - allows you to enforce passing a particular data type to Logstash for this configuration option, such as :string, :password, :boolean, :number, :array, :hash, :path (a file-system path), uri (starting in 5.0.0), :codec (since 1.2.0), :bytes (starting in 1.5.0). Note that this also works as a coercion in that if I specify "true" for boolean (even though technically a string), it will become a valid boolean in the config. This coercion works for the :number type as well where "1.2" becomes a float and "22" is an integer.

:default - lets you specify a default value for a parameter

:required - whether or not this parameter is mandatory (a Boolean true or false)

:list - whether or not this value should be a list of values. Will typecheck the list members, and convert scalars to one element lists. Note that this mostly obviates the array type, though if you need lists of complex objects that will be more suitable.

:deprecated - informational (also a Boolean true or false)

:obsolete - used to declare that a given setting has been removed and is no longer functioning. The idea is to provide an informed upgrade path to users who are still using a now-removed setting.
```

## Plugin Methods

Logstash outputs must implement the `register` and `multi_receive` methods.

### `register` Method

```
public
def register
end # def register
```

The Logstash `register` method is like an `initialize` method. It was originally created to enforce having `super` called, preventing headaches for newbies. (Note: It may go away in favor of `initialize`, in conjunction with some enforced testing to ensure `super` is called.)

`public` means the method can be called anywhere, not just within the class. This is the default behavior for methods in Ruby, but it is specified explicitly here anyway.

You can also assign instance variables here (variables prepended by `@`). Configuration variables are now in scope as instance variables, like `@message`

## Building the Plugin

At this point in the process you have coded your plugin and are ready to build a Ruby Gem from it. The following steps will help you complete the process.

### External dependencies

A `require` statement in Ruby is used to include necessary code. In some cases your plugin may require additional files. For example, the collectd plugin [uses](#) the `types.db` file provided by collectd. In the main directory of your plugin, a file called `vendor.json` is where these files are described.

The `vendor.json` file contains an array of JSON objects, each describing a file dependency. This example comes from the [collectd](#) codec plugin:

```
[{
 "sha1": "a90fe6cc53b76b7bdd56dc57950d90787cb9c96e",
 "url": "http://collectd.org/files/collectd-5.4.0.tar.gz",
 "files": ["/src/types.db"]
}]
```

`sha1` is the `sha1` signature used to verify the integrity of the file referenced by `url`.

`url` is the address from where Logstash will download the file.

`files` is an optional array of files to extract from the downloaded file. Note that while tar archives can use absolute or relative paths, treat them as absolute in this array. If `files` is not present, all files will be uncompressed and extracted into the `vendor` directory.

Another example of the `vendor.json` file is the [geoip filter](#)

The process used to download these dependencies is to call `rake vendor`. This will be discussed further in the testing section of this document.

Another kind of external dependency is on jar files. This will be described in the "Add a `gemspec` file" section.

### Add a Gemfile

Gemfiles allow Ruby's Bundler to maintain the dependencies for your plugin. Currently, all we'll need is the Logstash gem, for testing, but if you require other gems, you should add them in here.

Tip: See [Bundler's Gemfile page](#) for more details.

```
source 'https://rubygems.org'
gemspec
```

```
gem "logstash", :github => "elastic/logstash", :branch => "5.2"
```

### Add a `gemspec` file

Gemspecs define the Ruby gem which will be built and contain your plugin.

Tip: More information can be found on the [Rubygems Specification page](#).

```
Gem::Specification.new do |s|
 s.name = 'logstash-output-example'
 s.version = '0.1.0'
 s.licenses = ['Apache License (2.0)']
 s.summary = "This output does x, y, z in Logstash"
 s.description = "This gem is a logstash plugin required to be installed on top of the Logstash core pipeline using $LS_HOME/bin/logstash-plugin install gemname. This gem is not a stand-alone program"
 s.authors = ["Elastic"]
 s.email = 'info@elastic.co'
 s.homepage = "http://www.elastic.co/guide/en/logstash/current/index.html"
 s.require_paths = ["lib"]

 # Files
 s.files =
 Dir['lib/**/*', 'spec/**/*', 'vendor/**/*', '*.gemspec', '*.md', 'CONTRIBUTORS', 'Gemfile', 'LICENSE', 'NOTICE.TXT']
 # Tests
 s.test_files = s.files.grep(%r{^(test|spec|features)/})

 # Special flag to let us know this is actually a logstash plugin
 s.metadata = { "logstash_plugin" => "true", "logstash_group" => "output" }

 # Gem dependencies
 s.add_runtime_dependency "logstash-core-plugin-api", ">= 1.60", "<= 2.99"
 s.add_development_dependency 'logstash-devutils'
end
```

It is appropriate to change these values to fit your plugin. In particular, `s.name` and `s.summary` shoud reflect your plugin's name and behavior.

`s.licenses` and `s.version` are also important and will come into play when you are ready to publish your plugin.

Logstash and all its plugins are licensed under [Apache License, version 2 \("ALv2"\)](#). If you make your plugin publicly available via [RubyGems.org](#), please make sure to have this line in your gemspec:

```
s.licenses = ['Apache License (2.0)']
```

The gem version, designated by `s.version`, helps track changes to plugins over time. You should use [semver versioning](#) strategy for version numbers.

## Runtime & Development Dependencies

At the bottom of the `gemspec` file is a section with a comment: `Gem dependencies`. This is where any other needed gems must be mentioned. If a gem is necessary for your plugin to function, it is a runtime dependency. If a gem are only used for testing, then it would be a development dependency.

NOTE: You can also have versioning requirements for your dependencies—including other Logstash plugins:

```
Gem dependencies
s.add_runtime_dependency "logstash-core-plugin-api", ">= 1.60", "<= 2.99"
s.add_development_dependency 'logstash-devutils'
```

This `gemspec` has a runtime dependency on the `logstash-core-plugin-api` and requires that it have a version number greater than or equal to version 1.60 and less than or equal to version 2.99.

**IMPORTANT!!!** All plugins have a runtime dependency on the `logstash-core-plugin-api` gem, and a development dependency on `logstash-devutils`.

## Jar dependencies

In some cases, such as the [Elasticsearch output plugin](#), your code may depend on a jar file. In cases such as this, the dependency is added in the `gemspec` file in this manner:

```
Jar dependencies
s.requirements << "jar 'org.elasticsearch:elasticsearch', '5.0.0'"
s.add_runtime_dependency 'jar-dependencies'
```

With these both defined, the install process will search for the required jar file at <http://mvnrepository.com> and download the specified version.

## Add Tests

Logstash loves tests. Lots of tests. If you're using your new output plugin in a production environment, you'll want to have some tests to ensure you are not breaking any existing functionality.

NOTE: A full exposition on RSpec is outside the scope of this document. Learn more about RSpec at <http://rspec.info>

For help learning about tests and testing, look in the `spec/outputs/` directory of several other similar plugins.

## Clone and test!

Now let's start with a fresh clone of the plugin, build it and run the tests.

**Clone your plugin into a temporary location** Replace `GITUSERNAME` with your github username, and `MYPLUGINNAME` with your plugin name.

```
git clone https://github.com/GITUSERNAME/logstash-output-MYPLUGINNAME.git
```

```
alternately, via ssh: git clone git@github.com:GITUSERNAME/logstash-output-MYPLUGINNAME.git
```

```
cd logstash-output-MYPLUGINNAME
```

Then, you'll need to install your plugins dependencies with bundler:

```
bundle install
```

**IMPORTANT!!!** If your plugin has an external file dependency described in `vendor.json`, you must download that dependency before running or testing. You can do this by running:

```
rake vendor
```

And finally, run the tests:

```
bundle exec rspec
```

You should see a success message, which looks something like this:

```
Finished in 0.034 seconds
1 example, 0 failures
```

Hooray! You're almost there! (Unless you saw failures... you should fix those first).

## Building and Testing

Now you're ready to build your (well-tested) plugin into a Ruby gem.

### Build

You already have all the necessary ingredients, so let's go ahead and run the build command:

```
gem build logstash-output-example.gemspec
```

That's it! Your gem should be built and be in the same path with the name

```
logstash-output-mypluginname-0.1.0.gem
```

The `s.version` number from your `gemspec` file will provide the gem version, in this case, `0.1.0`.

## Test installation

You should test install your plugin into a clean installation of Logstash. Download the latest version from the [Logstash downloads page](#).

Untar and cd in to the directory:

```
curl -O https://download.elastic.co/logstash/logstash/logstash-
5.2.1.tar.gz
tar xzvf logstash-5.2.1.tar.gz
cd logstash-5.2.1
```

Using the plugin tool, we can install the gem we just built.

Replace `/my/logstash/plugins` with the correct path to the gem for your environment, and `0.1.0` with the correct version number from the `gemspec` file.

```
bin/logstash-plugin install /my/logstash/plugins/logstash-output-
example/logstash-output-example-0.1.0.gem
```

After running this, you should see feedback from Logstash that it was successfully installed:

```
validating /my/logstash/plugins/logstash-output-example/logstash-
output-example-0.1.0.gem >= 0
Valid logstash plugin. Continuing...
Successfully installed 'logstash-output-example' with version
'0.1.0'
```

Tip: You can also use the Logstash plugin tool to determine which plugins are currently available:

```
bin/logstash-plugin list
```

Depending on what you have installed, you might see a short or long list of plugins: inputs, codecs, filters and outputs.

Now try running Logstash with a simple configuration passed in via the command-line, using the `-e` flag.

NOTE: Your results will depend on what your output plugin is designed to do.

Congratulations! You've built, deployed and successfully run a Logstash output.

**Submitting your plugin to [RubyGems.org](#) and [logstash-plugins](#)**

Logstash uses [RubyGems.org](#) as its repository for all plugin artifacts. Once you have developed your new plugin, you can make it available to Logstash users by simply publishing it to RubyGems.org.

## Licensing

Logstash and all its plugins are licensed under [Apache License, version 2 \("ALv2"\)](#). If you make your plugin publicly available via [RubyGems.org](#), please make sure to have this line in your gemspec:

```
s.licenses = ['Apache License (2.0)']
```

## Publishing to [RubyGems.org](#)

To begin, you'll need an account on RubyGems.org

[Sign-up for a RubyGems account.](#)

After creating an account, [obtain](#) an API key from RubyGems.org. By default, RubyGems uses the file `~/.gem/credentials` to store your API key. These credentials will be used to publish the gem. Replace `username` and `password` with the credentials you created at RubyGems.org:

```
curl -u username:password https://rubygems.org/api/v1/api_key.yaml >
~/.gem/credentials
chmod 0600 ~/.gem/credentials
```

Before proceeding, make sure you have the right version in your gemspec file and commit your changes.

```
s.version = '0.1.0'
```

To publish version 0.1.0 of your new logstash gem:

```
bundle install
bundle exec rake vendor
bundle exec rspec
bundle exec rake publish_gem
```

**NOTE: Executing `rake publish_gem`:**

Reads the version from the gemspec file (`s.version = '0.1.0'`)

Checks in your local repository if a tag exists for that version. If the tag already exists, it aborts the process. Otherwise, it creates a new version tag in your local repository.

Builds the gem

Publishes the gem to RubyGems.org

That's it! Your plugin is published! Logstash users can now install your plugin by running:

```
bin/logstash-plugin install logstash-output-mypluginname
```

## Contributing your source code to [logstash-plugins](#)

It is not required to contribute your source code to [logstash-plugins](#) github organization, but we always welcome new plugins!

## Benefits

Some of the many benefits of having your plugin in the logstash-plugins repository are:

**Discovery** Your plugin will appear in the [Logstash Reference](#), where Logstash users look first for plugins and documentation.

**Documentation** Your plugin documentation will automatically be added to the [Logstash Reference](#).

**Testing** With our testing infrastructure, your plugin will be continuously tested against current and future releases of Logstash. As a result, users will have the assurance that if incompatibilities arise, they will be quickly discovered and corrected.

## Acceptance Guidelines

**Code Review** Your plugin must be reviewed by members of the community for coherence, quality, readability, stability and security.

**Tests** Your plugin must contain tests to be accepted. These tests are also subject to code review for scope and completeness. It's ok if you don't know how to write tests — we will guide you. We are working on publishing a guide to creating tests for Logstash which will make it easier. In the meantime, you can refer to <http://betterspecs.org/> for examples.

To begin migrating your plugin to logstash-plugins, simply create a new [issue](#) in the Logstash repository. When the acceptance guidelines are completed, we will facilitate the move to the logstash-plugins organization using the recommended [github process](#).

## Contributing a Patch to a Logstash Plugin

This section discusses the information you need to know to successfully contribute a patch to a Logstash plugin.

Each plugin defines its own configuration options. These control the behaviour of the plugin to some degree. Configuration option definitions commonly include:

- Data validation

- The default value

- Any required flags

Plugins are subclasses of a Logstash base class. A plugin's base class defines common configuration and methods.

### Input Plugins

Input plugins ingest data from an external source. Input plugins are always associated with a codec. An input plugin always has an associated codec plugin. Input and codec plugins operate in conjunction to create a Logstash event and add that event to the processing queue. An input codec is a subclass of the `LogStash::Inputs::Base` class.

**Table 1. Input API**

<code>#register() -&gt; nil</code>	Required. This API sets up resources for the plugin, typically the connection to the external source.
<code>#run(queue) -&gt; nil</code>	Required. This API fetches or listens for source data, typically looping until stopped. Must handle errors inside the loop. Pushes any created events to the queue object specified in the method argument. Some inputs may receive batched data to minimize the external call overhead.
<code>#stop() -&gt; nil</code>	Optional. Stops external connections and cleans up.

### Codec Plugins

Codec plugins decode input data that has a specific structure, such as JSON input data. A codec plugin is a subclass of `LogStash::Codecs::Base`.

**Table 2. Codec API**

<code>#register() -&gt; nil</code>	Identical to the API of the same name for input plugins.
------------------------------------	----------------------------------------------------------

<code>#decode(data) {  event  block} -&gt; nil</code>	Must be implemented. Used to create an Event from the raw data given in the method argument. Must handle errors. The caller must provide a Ruby block. The block is called with the created Event.
<code>#encode(event) -&gt; nil</code>	Required. Used to create a structured data object from the given Event. May handle errors. This method calls a block that was previously stored as @on_event with two arguments: the original event and the data object.

## Filter Plugins

A mechanism to change, mutate or merge one or more Events. A filter plugin is a subclass of the `LogStash::Filters::Base` class.

**Table 3. Filter API**

<code>#register() -&gt; nil</code>	Identical to the API of the same name for input plugins.
<code>#filter(event) -&gt; nil</code>	Required. May handle errors. Used to apply a mutation function to the given event.

## Output Plugins

A mechanism to send an event to an external destination. This process may require serialization. An output plugin is a subclass of the `LogStash::Outputs::Base` class.

**Table 4. Output API**

<code>#register() -&gt; nil</code>	Identical to the API of the same name for input plugins.
<code>#receive(event) -&gt; nil</code>	Required. Must handle errors. Used to prepare the given event for transmission to the external destination. Some outputs may buffer the prepared events to batch transmit to the destination.

## Process

A bug or feature is identified. An issue is created in the plugin repository. A patch is created and a pull request (PR) is submitted. After review and possible rework the PR is merged and the plugin is published.

The [Community Maintainer Guide](#) explains, in more detail, the process of getting a patch accepted, merged and published. The Community Maintainer Guide also details the roles that contributors and maintainers are expected to perform.

## Testing Methodologies

### *Test Driven Development*

Test Driven Development, colloquially known as TDD, describes a methodology for using tests to guide evolution of source code. For our purposes, we are only going to use a part of it, that is, before writing the fix - we create tests that illustrate the bug by failing. We stop when we have written enough code to make the tests pass and submit the fix and tests as a patch. It is not necessary to write the tests before the fix, but it is very easy to write a passing test afterwards that may not actually verify that the fault is really fixed especially if the fault can be triggered via multiple execution paths or varying input data.

### *The RSpec Framework*

Logstash uses Rspec, a Ruby testing framework, to define and run the test suite. What follows is a summary of various sources.

#### Rspec Example

```

1 # encoding: utf-8
2 require "logstash/devutils/rspec/spec_helper"
3 require "logstash/plugin"
4
5 describe "outputs/riemann" do
6 describe "#register" do
7 let(:output) do
8 LogStash::Plugin.lookup("output", "riemann").new(configuration)
9 end
10
11 context "when no protocol is specified" do
12 let(:configuration) { Hash.new }
13
14 it "the method completes without error" do
15 expect {output.register}.not_to raise_error
16 end
17 end
18
19 context "when a bad protocol is specified" do
20 let(:configuration) { {"protocol" => "fake"} }
21
22 it "the method fails with error" do
23 expect {output.register}.to raise_error
24 end
25 end
26
27 context "when the tcp protocol is specified" do
28 let(:configuration) { {"protocol" => "tcp"} }

```

```

29
30 it "the method completes without error" do
31 expect {output.register}.not_to raise_error
32 end
33 end
35
36 describe "#receive" do
37 let(:output) do
38 LogStash::Plugin.lookup("output", "riemann").new(configuration)
39 end
40
41 context "when operating normally" do
42 let(:configuration) { Hash.new }
43 let(:event) do
44 data = {"message"=>"hello", "@version"=>"1",
45 "@timestamp"=>"2015-06-03T23:34:54.076Z",
46 "host"=>"vagrant-ubuntu-trusty-64"}
47 LogStash::Event.new(data)
48 end
49
50 before(:example) do
51 output.register
52 end
53
54 it "should accept the event" do
55 expect { output.receive event }.not_to raise_error
56 end
57 end
58 end
59 end

```

### Describe blocks (lines 5, 6 and 36 in Example 1).

```
describe(string){block} -> nil
describe(Class){block} -> nil
```

With RSpec, we are always describing the plugin method behavior. The describe block is added in logical sections and can accept either an existing class name or a string. The string used in line 5 is the plugin name. Line 6 is the register method, line 36 is the receive method. It is a RSpec convention to prefix instance methods with one hash and class methods with one dot.

### Context blocks (lines 11, 19, 27 and 41).

```
context(string){block} -> nil
```

In RSpec, context blocks define sections that group tests by a variation. The string should start with the word `when` and then detail the variation. See line 11. The tests in the content block should only be for that variation.

### Let blocks (lines 7, 12, 20, 28, 37, 42 and 43).

```
let(symbol){block} -> nil
```

In RSpec, `let` blocks define resources for use in the test blocks. These resources are reinitialized for every test block. They are available as method calls inside the test block. Define `let` blocks in `describe` and `context` blocks, which scope the `let` block and any other nested blocks. You can use other `let` methods defined later within the `let` block body. See lines 7-9, which define the output resource and use the configuration method, defined with different variations in lines 12, 20 and 28.

### Before blocks (line 50).

```
before(symbol){block} -> nil - symbol is one of :suite, :context, :example,
but :all and :each are synonyms for :suite and :example respectively.
```

In RSpec, `before` blocks are used to further set up any resources that would have been initialized in a `let` block. You cannot define `let` blocks inside `before` blocks.

You can also define `after` blocks, which are typically used to clean up any setup activity performed by a `before` block.

### It blocks (lines 14, 22, 30 and 54).

```
it(string){block} -> nil
```

In RSpec, `it` blocks set the expectations that verify the behavior of the tested code. The string should not start with `it` or `should`, but needs to express the outcome of the expectation. When put together the texts from the enclosing `describe`, `context` and `it` blocks should form a fairly readable sentence, as in lines 5, 6, 11 and 14:

```
outputs/riemann
#register when no protocol is specified the method completes without error
```

Readable code like this make the goals of tests easy to understand.

### Expect method (lines 15, 23, 31, 55).

```
expect(object){block} -> nil
```

In RSpec, the `expect` method verifies a statement that compares an actual result to an expected result. The `expect` method is usually paired with a call to the `to` or `not_to` methods. Use the block form when expecting errors or observing for changes. The `to` or `not_to` methods require a `matcher` object that encapsulates the expected value. The argument form of the `expect` method encapsulates the actual value. When put together the whole line tests the actual against the expected value.

### Matcher methods (lines 15, 23, 31, 55).

```
raise_error(error class|nil) -> matcher instance
be(object) -> matcher instance
```

```
eq(object) -> matcher instance
eql(object) -> matcher instance
 for more see http://www.relishapp.com/rspec/rspec-expectations/docs/built-in-matchers
```

In RSpec, a matcher is an object generated by the equivalent method call (be, eq) that will be used to evaluate the expected against the actual values.

## Putting it all together

This example fixes an [issue](#) in the ZeroMQ output plugin. The issue does not require knowledge of ZeroMQ.

The activities in this example have the following prerequisites:

A minimal knowledge of Git and Github. See the [Github boot camp](#).

A text or.

A JRuby [runtime environment](#). The `chruby` tool manages Ruby versions.

JRuby 1.7.22 or later.

The `bundler` and `rake` gems installed.

ZeroMQ [installed](#).

In Github, fork the ZeroMQ [output plugin repository](#).

On your local machine, [clone](#) the fork to a known folder such as `logstash/`.

Open the following files in a text or:

```
logstash-output-zeromq/lib/logstash/outputs/zeromq.rb
logstash-output-zeromq/lib/logstash/util/zeromq.rb
logstash-output-zeromq/spec/outputs/zeromq_spec.rb
```

According to the issue, log output in server mode must indicate `bound`. Furthermore, the test file contains no tests.

Line 21 of `util/zeromq.rb` reads `@logger.info("0mq: #{server? ? 'connected' : 'bound'}", :address => address)`

In the text or, set the file encoding and require `zeromq.rb` for the file `zeromq_spec.rb` by adding the following lines:

```
encoding: utf-8
require "logstash/outputs/zeromq"
require "logstash/devutils/rspec/spec_helper"
```

The desired error message should read:

```
LogStash::Outputs::ZeroMQ when in server mode a 'bound' info line is
logged
```

To properly generate this message, add a `describe` block with the fully qualified class name as the argument, a context block, and an `it` block.

```
describe LogStash::Outputs::ZeroMQ do
 context "when in server mode" do
 it "a 'bound' info line is logged" do
 end
 end
 end
```

To add the missing test, use an instance of the ZeroMQ output and a substitute logger. This example uses an RSpec feature called *test doubles* as the substitute logger.

Add the following lines to `zeromq_spec.rb`, after `describe LogStash::Outputs::ZeroMQ do` and before `context "when in server mode" do:`

```
let(:output) { described_class.new("mode" => "server", "topology" =>
"pushpull")
let(:tracer) { double("logger") }
```

Add the body to the `it` block. Add the following five lines after the line `context "when in server mode" do:`

```
allow(tracer).to receive(:debug)

output.logger = logger
expect(tracer).to receive(:info).with("0mq: bound",
{:address=>"tcp://127.0.0.1:2120"})

output.register

output.do_close
Allow the double to receive debug method calls.
```

Make the output use the test double.

Set an expectation on the test to receive an `info` method call.

Call `register` on the output.

Call `do_close` on the output so the test does not hang.

At the end of the modifications, the relevant code section reads:

```
encoding: utf-8
require "logstash/outputs/zeromq"
require "logstash/devutils/rspec/spec_helper"

describe Logstash::Outputs::ZeroMQ do
 let(:output) { described_class.new("mode" => "server", "topology" =>
"pushpull") }
 let(:tracer) { double("logger") }

 context "when in server mode" do
 it "a 'bound' info line is logged" do
 allow(tracer).to receive(:debug)
 output.logger = tracer
 expect(tracer).to receive(:info).with("0mq: bound",
{:address=>"tcp://127.0.0.1:2120"})
 output.register
 output.do_close
 end
 end
end
```

To run this test:

Open a terminal window

Mavigate to the cloned plugin folder

The first time you run the test, run the command `bundle install`

Run the command `bundle exec rspec`

Assuming all prerequisites were installed correctly, the test fails with output similar to:

```
Using Accessor#strict_set for specs
Run options: exclude {:redis=>true, :socket=>true, :performance=>true,
:couchdb=>true, :elasticsearch=>true,
:elasticsearch_secure=>true, :export_cypher=>true, :integration=>true,
:windows=>true}

Logstash::Outputs::ZeroMQ
when in server mode
```

## Logstash 5.2 Configuration Guide

```
a 'bound' info line is logged (FAILED - 1)
```

Failures:

```
1) LogStash::Outputs::ZeroMQ when in server mode a 'bound' info line is
logged
```

```
Failure/Error: output.register
Double "logger" received :info with unexpected arguments
 expected: ("0mq: bound", {:address=>"tcp://127.0.0.1:2120"})
 got: ("0mq: connected", {:address=>"tcp://127.0.0.1:2120"})
./lib/logstash/util/zeromq.rb:21:in `setup'
./lib/logstash/outputs/zeromq.rb:92:in `register'
./lib/logstash/outputs/zeromq.rb:91:in `register'
./spec/outputs/zeromq_spec.rb:13:in `(root)'
/Users/guy/.gem/jruby/1.9.3/gems/rspec-wait-
0.0.7/lib/rspec/wait.rb:46:in `(root)'
```

```
Finished in 0.133 seconds (files took 1.28 seconds to load)
1 example, 1 failure
```

Failed examples:

```
rspec ./spec/outputs/zeromq_spec.rb:10 # LogStash::Outputs::ZeroMQ when in
server mode a 'bound' info line is logged
```

Randomized with seed 2568

To correct the error, open the `util/zeromq.rb` file in your text or and swap the positions of the words `connected` and `bound` on line 21. Line 21 now reads:

```
@logger.info("0mq: #{server? ? 'bound' : 'connected'}", :address => address)
```

Run the test again with the `bundle exec rspec` command.

The test passes with output similar to:

```
Using Accessor#strict_set for specs
Run options: exclude { :redis=>true, :socket=>true, :performance=>true,
:couchdb=>true, :elasticsearch=>true, :elasticsearch_secure=>true,
:export_cypher=>true, :integration=>true, :windows=>true}
```

```
LogStash::Outputs::ZeroMQ
when in server mode
 a 'bound' info line is logged
```

```
Finished in 0.114 seconds (files took 1.22 seconds to load)
1 example, 0 failures
```

Randomized with seed 45887

[Commit](#) the changes to git and Github.

Your pull request is visible from the [Pull Requests](#) section of the original Github repository. The plugin maintainers review your work, suggest changes if necessary, and merge and publish a new version of the plugin.

# Logstash Plugins Community Maintainer Guide

This document, to be read by new Maintainers, should explain their responsibilities. It was inspired by the [C4](#) document from the ZeroMQ project. This document is subject to change and suggestions through Pull Requests and issues are strongly encouraged.

## Contribution Guidelines

For general guidance around contributing to Logstash Plugins, see the [\*Contributing to Logstash\*](#) section.

## Document Goals

To help make the Logstash plugins community participation easy with positive feedback.

To increase diversity.

To reduce code review, merge and release dependencies on the core team by providing support and tools to the Community and Maintainers.

To support the natural life cycle of a plugin.

To codify the roles and responsibilities of: Maintainers and Contributors with specific focus on patch testing, code review, merging and release.

## Development Workflow

All Issues and Pull Requests must be tracked using the Github issue tracker.

The plugin uses the [Apache 2.0 license](#). Maintainers should check whether a patch introduces code which has an incompatible license. Patch ownership and copyright is defined in the Elastic [Contributor License Agreement](#) (CLA).

### [Terminology](#)

A "Contributor" is a role a person assumes when providing a patch. Contributors will not have commit access to the repository. They need to sign the Elastic [Contributor License Agreement](#) before a patch can be reviewed. Contributors can add themselves to the plugin Contributor list.

A "Maintainer" is a role a person assumes when maintaining a plugin and keeping it healthy, including triaging issues, and reviewing and merging patches.

#### *Patch Requirements*

A patch is a minimal and accurate answer to exactly one identified and agreed upon problem. It must conform to the [code style guidelines](#) and must include RSpec tests that verify the fitness of the solution.

A patch will be automatically tested by a CI system that will report on the Pull Request status.

A patch CLA will be automatically verified and reported on the Pull Request status.

A patch commit message has a single short (less than 50 character) first line summarizing the change, a blank second line, and any additional lines as necessary for change explanation and rationale.

A patch is mergeable when it satisfies the above requirements and has been reviewed positively by at least one other person.

#### *Development Process*

A user will log an issue on the issue tracker describing the problem they face or observe with as much detail as possible.

To work on an issue, a Contributor forks the plugin repository and then works on their forked repository and submits a patch by creating a pull request back to the plugin.

Maintainers must not merge patches where the author has not signed the CLA.

Before a patch can be accepted it should be reviewed. Maintainers should merge accepted patches without delay.

Maintainers should not merge their own patches except in exceptional cases, such as non-responsiveness from other Maintainers or core team for an extended period (more than 2 weeks).

Reviewer's comments should not be based on personal preferences.

The Maintainers should label Issues and Pull Requests.

Maintainers should involve the core team if help is needed to reach consensus.

Review non-source changes such as documentation in the same way as source code changes.

#### *Branch Management*

The plugin has a master branch that always holds the latest in-progress version and should always build. Topic branches should kept to the minimum.

### *Changelog Management*

Every plugin should have a changelog (CHANGELOG.md). If not, please create one. When changes are made to a plugin, make sure to include a changelog entry under the respective version to document the change. The changelog should be easily understood from a user point of view. As we iterate and release plugins rapidly, users use the changelog as a mechanism for deciding whether to update.

Changes that are not user facing should be tagged as `internal`. For example:

`internal: Refactored specs for better testing`

`config: Default timeout configuration changed from 10s to 5s`

[Detailed format of CHANGELOG.md](#)

Sharing a similar format of CHANGELOG.md in plugins ease readability for users. Please see following annotated example and see a concrete example in [logstash-filter-date](#).

```
1.0.x //
- change description //
- tag: change description //
- tag1,tag2: change description //
- tag: Multi-line description //
 must be indented and can use
 additional markdown syntax
//
1.0.0 //
[...]
```

Latest version is the first line of CHANGELOG.md

Each version identifier should be a level-2 header using `##`

One change description is described as a list item using a dash - aligned under the version identifier

One change can be tagged by a word and suffixed by :. Common tags are `bugfix`, `feature`, `doc`, `test` or `internal`.

One change can have multiple tags separated by a comma and suffixed by :

A multi-line change description must be properly indented

Please take care to **separate versions with an empty line**

Previous version identifier

#### *Continuous Integration*

Plugins are setup with automated continuous integration (CI) environments and there should be a corresponding badge on each Github page. If it's missing, please contact the Logstash core team.

Every Pull Request opened automatically triggers a CI run. To conduct a manual run, comment "Jenkins, please test this." on the Pull Request.

## Versioning Plugins

Logstash core and its plugins have separate product development lifecycles. Hence the versioning and release strategy for the core and plugins do not have to be aligned. In fact, this was one of our goals during the great separation of plugins work in Logstash 1.5.

At times, there will be changes in core API in Logstash, which will require mass update of plugins to reflect the changes in core. However, this does not happen frequently.

For plugins, we would like to adhere to a versioning and release strategy that can better inform our users, about any breaking changes to the Logstash configuration formats and functionality.

Plugin releases follows a three-placed numbering scheme X.Y.Z. where X denotes a major release version which may break compatibility with existing configuration or functionality. Y denotes releases which includes features which are backward compatible. Z denotes releases which includes bug fixes and patches.

#### *Changing the version*

Version can be changed in the Gemspec, which needs to be associated with a changelog entry. Following this, we can publish the gem to RubyGem.org manually. At this point only the core developers can publish a gem.

#### *Labeling*

Labeling is a critical aspect of maintaining plugins. All issues in GitHub should be labeled correctly so it can:

Provide good feedback to users/developers

Help prioritize changes

Be used in release notes

Most labels are self explanatory, but here's a quick recap of few important labels:

`bug`: Labels an issue as an unintentional defect

`needs details`: If a the issue reporter has incomplete details, please ask them for more info and label as `needs details`.

`missing cla`: Contributor License Agreement is missing and patch cannot be accepted without it

`adopt me`: Ask for help from the community to take over this issue

`enhancement`: New feature, not a bug fix

`needs tests`: Patch has no tests, and cannot be accepted without unit/integration tests

`docs`: Documentation related issue/PR

## Logging

Although it's important not to bog down performance with excessive logging, debug level logs can be immensely helpful when diagnosing and troubleshooting issues with Logstash. Please remember to liberally add debug logs wherever it makes sense as users will be forever gracious.

```
@logger.debug("Logstash loves debug logs!", :actions => actions)
```

## Contributor License Agreement (CLA) Guidance

### 1. Why is a [CLA](#) required?

We ask this of all Contributors in order to assure our users of the origin and continuing existence of the code. We are not asking Contributors to assign copyright to us, but to give us the right to distribute a Contributor's code without restriction.

### 2. Please make sure the CLA is signed by every Contributor prior to reviewing PRs and commits.

Contributors only need to sign the CLA once and should sign with the same email as used in Github. If a Contributor signs the CLA after a PR is submitted, they can refresh the automated CLA checker by pushing another comment on the PR after 5 minutes of signing.

## Need Help?

Ping @logstash-core on Github to get the attention of the Logstash core team.

## **Community Administration**

The core team is there to support the plugin Maintainers and overall ecosystem.

Maintainers should propose Contributors to become a Maintainer.

Contributors and Maintainers should follow the Elastic Community [Code of Conduct](#). The core team should block or ban "bad actors".

# Submitting your plugin to RubyGems.org and the logstash-plugins repository

Logstash uses [RubyGems.org](#) as its repository for all plugin artifacts. Once you have developed your new plugin, you can make it available to Logstash users by simply publishing it to RubyGems.org.

## Licensing

Logstash and all its plugins are licensed under [Apache License, version 2 \("ALv2"\)](#). If you make your plugin publicly available via [RubyGems.org](#), please make sure to have this line in your gemspec:

```
s.licenses = ['Apache License (2.0)']
```

## Publishing to [RubyGems.org](#)

To begin, you'll need an account on RubyGems.org

[Sign-up for a RubyGems account.](#)

After creating an account, [obtain](#) an API key from RubyGems.org. By default, RubyGems uses the file `~/.gem/credentials` to store your API key. These credentials will be used to publish the gem. Replace `username` and `password` with the credentials you created at RubyGems.org:

```
curl -u username:password https://rubygems.org/api/v1/api_key.yaml >
~/.gem/credentials
chmod 0600 ~/.gem/credentials
```

Before proceeding, make sure you have the right version in your gemspec file and commit your changes.

```
s.version = '0.1.0'
```

To publish version 0.1.0 of your new logstash gem:

```
bundle install
bundle exec rake vendor
bundle exec rspec
bundle exec rake publish_gem
```

Executing `rake publish_gem`:

Reads the version from the gemspec file (`s.version = '0.1.0'`)

Checks in your local repository if a tag exists for that version. If the tag already exists, it aborts the process. Otherwise, it creates a new version tag in your local repository.

Builds the gem

Publishes the gem to RubyGems.org

That's it! Your plugin is published! Logstash users can now install your plugin by running:

```
bin/plugin install logstash-output-mypluginname
```

## Contributing your source code to [logstash-plugins](#)

It is not required to contribute your source code to [logstash-plugins](#) github organization, but we always welcome new plugins!

## Benefits

Some of the many benefits of having your plugin in the logstash-plugins repository are:

**Discovery** Your plugin will appear in the [Logstash Reference](#), where Logstash users look first for plugins and documentation.

**Documentation** Your plugin documentation will automatically be added to the [Logstash Reference](#).

**Testing** With our testing infrastructure, your plugin will be continuously tested against current and future releases of Logstash. As a result, users will have the assurance that if incompatibilities arise, they will be quickly discovered and corrected.

## Acceptance Guidelines

**Code Review** Your plugin must be reviewed by members of the community for coherence, quality, readability, stability and security.

**Tests** Your plugin must contain tests to be accepted. These tests are also subject to code review for scope and completeness. It's ok if you don't know how to write tests — we will guide you. We are working on publishing a guide to creating tests for Logstash which will make it easier. In the meantime, you can refer to <http://betterspecs.org/> for examples.

To begin migrating your plugin to logstash-plugins, simply create a new [issue](#) in the Logstash repository. When the acceptance guidelines are completed, we will facilitate the move to the logstash-plugins organization using the recommended [github process](#).

# Glossary of Terms

## @metadata

A special field for storing content that you don't want to include in output [events](#). For example, the `@metadata` field is useful for creating transient fields for use in [conditional statements](#).

## codec plugin

A Logstash [plugin](#) that changes the data representation of an [event](#). Codecs are essentially stream filters that can operate as part of an input or output. Codecs enable you to separate the transport of messages from the serialization process. Popular codecs include json, msgpack, and plain (text).

## conditional

A control flow that executes certain actions based on whether a statement (also called a condition) is true or false. Logstash supports `if`, `else if`, and `else` statements. You can use conditional statements to apply filters and send events to a specific output based on conditions that you specify.

## event

A single unit of information, containing a timestamp plus additional data. An event arrives via an input, and is subsequently parsed, timestamped, and passed through the Logstash [pipeline](#).

## field

An [event](#) property. For example, each event in an apache access log has properties, such as a status code (200, 404), request path ("/", "index.html"), HTTP verb (GET, POST), client IP address, and so on. Logstash uses the term "fields" to refer to these properties.

## field reference

A reference to an event [field](#). This reference may appear in an output block or filter block in the Logstash config file. Field references are typically wrapped in square `[]` brackets, for example `[fieldname]`. If you are referring to a top-level field, you can omit the `[]` and simply use the field name. To refer to a nested field, you specify the full path to that field: `[top-level field] [nested field]`.

## filter plugin

A Logstash [plugin](#) that performs intermediary processing on an [event](#). Typically, filters act upon event data after it has been ingested via inputs, by mutating, enriching, and/or modifying the data according to configuration rules. Filters are often applied conditionally depending on the characteristics of the event. Popular filter plugins include grok, mutate, drop, clone, and geoip. Filter stages are optional.

## gem

A self-contained package of code that's hosted on [RubyGems.org](#). Logstash [plugins](#) are packaged as Ruby Gems. You can use the Logstash [plugin manager](#) to manage Logstash gems.

## hot thread

A Java thread that has high CPU usage and executes for a longer than normal period of time.

## input plugin

A Logstash [plugin](#) that reads [event](#) data from a specific source. Input plugins are the first stage in the Logstash event processing [pipeline](#). Popular input plugins include file, syslog, redis, and beats.

indexer

A Logstash instance that is tasked with interfacing with an Elasticsearch cluster in order to index [event](#) data.

message broker

Also referred to as a *message buffer* or *message queue*, a message broker is external software (such as Redis, Kafka, or RabbitMQ) that stores messages from the Logstash shipper instance as an intermediate store, waiting to be processed by the Logstash indexer instance.

output plugin

A Logstash [plugin](#) that writes [event](#) data to a specific destination. Outputs are the final stage in the event [pipeline](#). Popular output plugins include elasticsearch, file, graphite, and statsd.

pipeline

A term used to describe the flow of [events](#) through the Logstash workflow. A pipeline typically consists of a series of input, filter, and output stages. [Input](#) stages get data from a source and generate events, [filter](#) stages, which are optional, modify the event data, and [output](#) stages write the data to a destination. Inputs and outputs support [codecs](#) that enable you to encode or decode the data as it enters or exits the pipeline without having to use a separate filter.

plugin

A self-contained software package that implements one of the stages in the Logstash event processing [pipeline](#). The list of available plugins includes [input plugins](#), [output plugins](#), [codec plugins](#), and [filter plugins](#). The plugins are implemented as Ruby [gems](#) and hosted on [RubyGems.org](#). You define the stages of an event processing [pipeline](#) by configuring plugins.

plugin manager

Accessed via the `bin/logstash-plugin` script, the plugin manager enables you to manage the lifecycle of [plugins](#) in your Logstash deployment. You can install, remove, and upgrade plugins by using the plugin manager Command Line Interface (CLI).

shipper

An instance of Logstash that send events to another instance of Logstash, or some other application.

worker

The filter thread model used by Logstash, where each worker receives an [event](#) and applies all filters, in order, before emitting the event to the output queue. This allows scalability across CPUs because many filters are CPU intensive.