Deploying and Scaling Logstash

edit

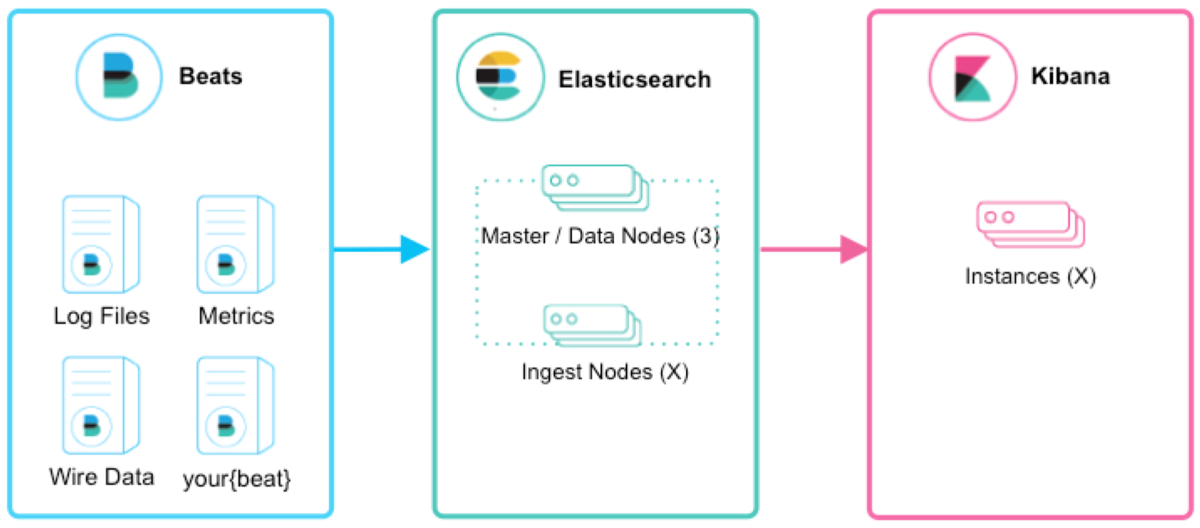
The Elastic Stack is used for tons of use cases, from operational log and metrics analytics, to enterprise and application search. Making sure your data gets scalably, durably, and securely transported to Elasticsearch is extremely important, especially for mission critical environments.

The goal of this document is to highlight the most common architecture patterns for Logstash and how to effectively scale as your deployment grows. The focus will be around the operational log, metrics, and security analytics use cases because they tend to require larger scale deployments. The deploying and scaling recommendations provided here may vary based on your own requirements.

Getting Started

edit

For first time users, if you simply want to tail a log file to grasp the power of the Elastic Stack, we recommend trying Filebeat Modules. Filebeat Modules enable you to quickly collect, parse, and index popular log types and view pre-built Kibana dashboards within minutes. Metricbeat Modules provide a similar experience, but with metrics data. In this context, Beats will ship data directly to Elasticsearch where Ingest Nodes will process and index your data.



Introducing Logstash

edit

What are the main benefits for integrating Logstash into your architecture?

- Scale through ingestion spikes - Logstash has an adaptive disk-based buffering system that will absorb incoming throughput, therefore mitigating backpressure
- Ingest from other data sources like databases, S3, or messaging queues
- Emit data to multiple destinations like S3, HDFS, or write to a file
- Compose more sophisticated processing pipelines with conditional dataflow logic

Scaling Ingest

edit

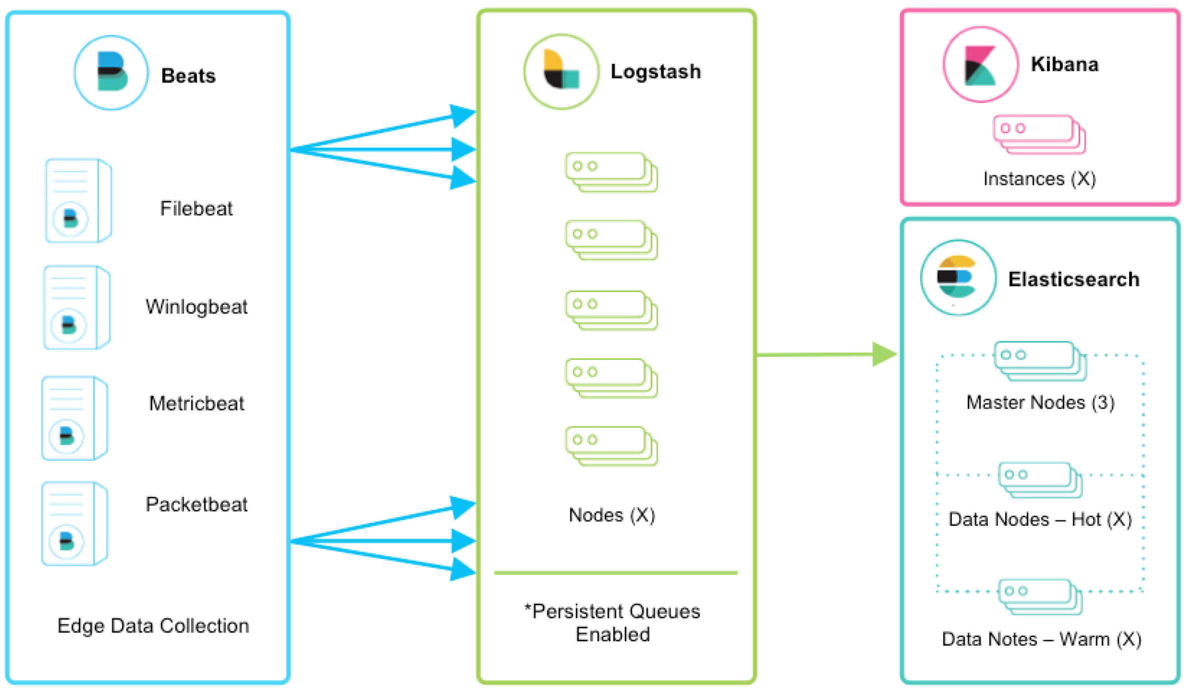
Beats and Logstash make ingest awesome. Together, they provide a comprehensive solution that is scalable and resilient. What can you expect?

- Horizontal scalability, high availability, and variable load handling
- Message durability with at-least-once delivery guarantees
- End-to-end secure transport with authentication and wire encryption

Beats and Logstash

edit

Beats run across thousands of edge host servers, collecting, tailing, and shipping logs to Logstash. Logstash serves as the centralized streaming engine for data unification and enrichment. The Beats input plugin exposes a secure, acknowledgement-based endpoint for Beats to send data to Logstash.



NOTE Enabling persistent queues is strongly recommended, and these architecture characteristics assume that they are enabled. We encourage you to review the Persistent Queues documentation for feature benefits and more details on resiliency.

Scalability

edit

Logstash is horizontally scalable and can form groups of nodes running the same pipeline. Logstash's adaptive buffering capabilities will facilitate smooth streaming even through variable throughput loads. If the Logstash layer becomes an ingestion bottleneck, simply add more nodes to scale out. Here are a few general recommendations:

- Beats should load balance across a group of Logstash nodes.
- A minimum of two Logstash nodes are recommended for high availability.
- It's common to deploy just one Beats input per Logstash node, but multiple Beats inputs can also be deployed per Logstash node to expose independent endpoints for different data sources.

Resiliency

edit

When using Filebeat or Winlogbeat for log collection within this ingest flow, **at-least-once delivery** is guaranteed. Both the communication protocols, from Filebeat or Winlogbeat to Logstash, and from Logstash to Elasticsearch, are synchronous and support acknowledgements. The other Beats don't yet have support for acknowledgements.

Logstash persistent queues provide protection across node failures. For disk-level resiliency in Logstash, it's important to ensure disk redundancy. For on-premise deployments, it's recommended that you configure RAID. When running in the cloud or a containerized environment, it's recommended that you use persistent disks with replication strategies that reflect your data SLAs.

NOTE Make sure `queue.checkpoint.writes: 1` is set for at-least-once guarantees. For more details, see the persistent queue durability documentation.

Processing

edit

Logstash will commonly extract fields with grok or dissect, augment geographical info, and can further enrich events with file, database, or Elasticsearch lookup datasets. Be aware that processing complexity can affect overall throughput and CPU utilization. Make sure to check out the other available filter plugins.

Secure Transport

edit

Enterprise-grade security is available across the entire delivery chain.

- Wire encryption is recommended for both the transport from Beats to Logstash and from Logstash to Elasticsearch.
- There's a wealth of security options when communicating with Elasticsearch including basic authentication, TLS, PKI, LDAP, AD, and other custom realms. To enable Elasticsearch security, see Secure a cluster.

Monitoring

edit

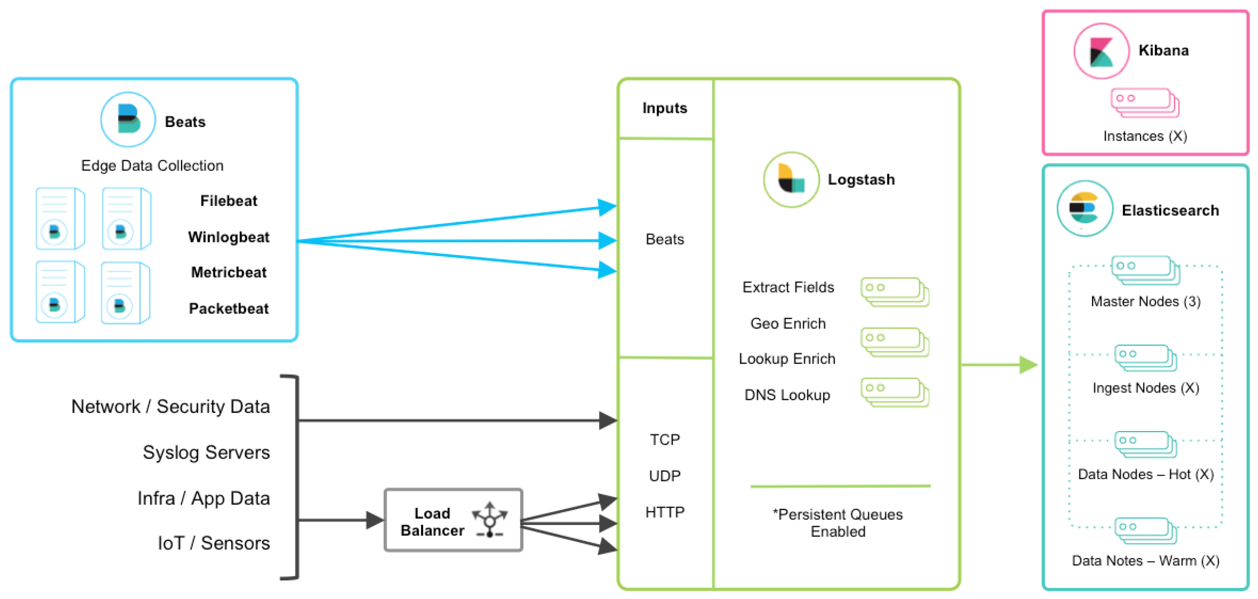
When running Logstash 5.2 or greater, the Monitoring UI provides deep visibility into your deployment metrics, helping observe performance and alleviate bottlenecks as you scale. Monitoring is an X-Pack feature under the Basic License and is therefore **free to use**. To get started, see Monitoring Logstash.

If external monitoring is preferred, there are Monitoring APIs that return point-in-time metrics snapshots.

Adding Other Popular Sources

edit

Users may have other mechanisms of collecting logging data, and it's easy to integrate and centralize them into the Elastic Stack. Let's walk through a few scenarios:



TCP, UDP, and HTTP Protocols

edit

The TCP, UDP, and HTTP protocols are common ways to feed data into Logstash. Logstash can expose endpoint listeners with the respective TCP, UDP, and HTTP input plugins. The data sources enumerated below are typically ingested through one of these three protocols.

NOTE The TCP protocol does not support application-level acknowledgements, so connectivity issues may result in data loss.

For high availability scenarios, a third-party hardware or software load balancer, like HAProxy, should be added to fan out traffic to a group of Logstash nodes.

Network and Security Data

edit

Although Beats may already satisfy your data ingest use case, network and security datasets come in a variety of forms. Let's touch on a few other ingestion points.

- Network wire data - collect and analyze network traffic with Packetbeat.

- Netflow v5/v9/v10 - Logstash understands data from Netflow/IPFIX exporters with the Netflow codec.
- Nmap - Logstash accepts and parses Nmap XML data with the Nmap codec.
- SNMP trap - Logstash has a native SNMP trap input.
- CEF - Logstash accepts and parses CEF data from systems like Arcsight SmartConnectors with the CEF codec. See this blog series for more details.

Centralized Syslog Servers

edit

Existing syslog server technologies like rsyslog and syslog-ng generally send syslog over to Logstash TCP or UDP endpoints for extraction, processing, and persistence. If the data format conforms to RFC3164, it can be fed directly to the Logstash syslog input.

Infrastructure & Application Data and IoT

edit

Infrastructure and application metrics can be collected with Metricbeat, but applications can also send webhooks to a Logstash HTTP input or have metrics polled from an HTTP endpoint with the HTTP poller input plugin.

For applications that log with log4j2, it's recommended to use the SocketAppender to send JSON to the Logstash TCP input. Alternatively, log4j2 can also log to a file for collection with FIlebeat. Usage of the log4j1 SocketAppender is not recommended.

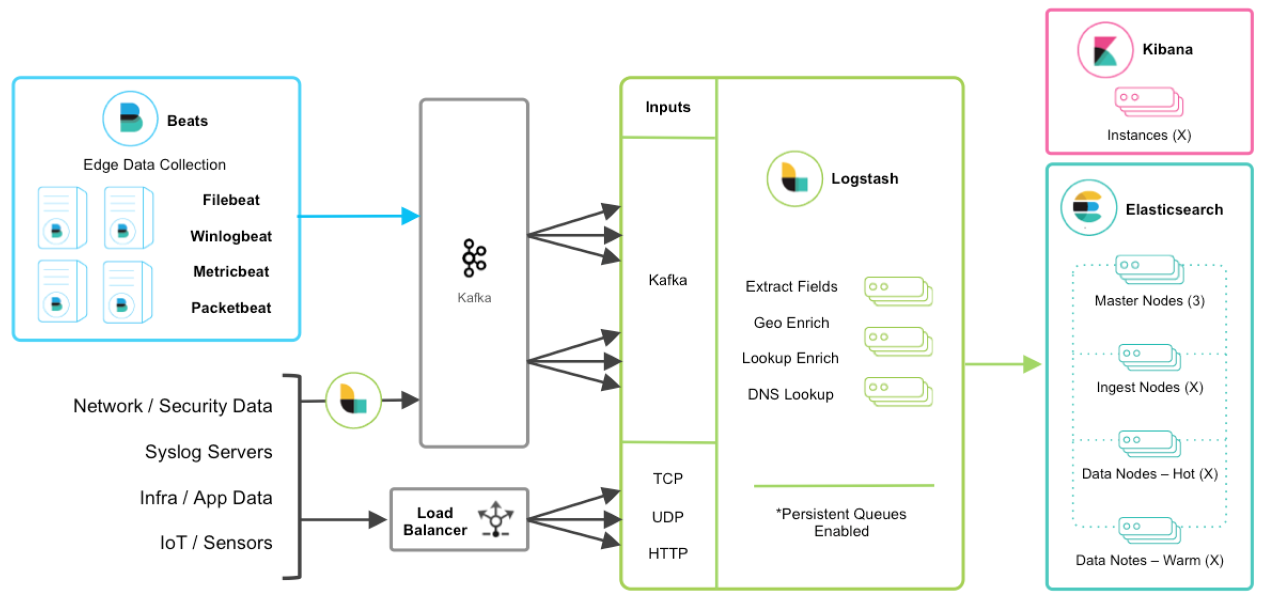
IoT devices like Raspberry Pis, smartphones, and connected vehicles often send telemetry data through one of these protocols.

Integrating with Messaging Queues

edit

If you are leveraging message queuing technologies as part of your existing infrastructure, getting that data into the Elastic Stack is easy. For existing users who are utilizing an external queuing layer like Redis or RabbitMQ just for data buffering with Logstash, it's recommended to use Logstash persistent queues instead of an external queuing layer. This will help with overall ease of management by removing an unnecessary layer of complexity in your ingest architecture.

For users who want to integrate data from existing Kafka deployments or require the underlying usage of ephemeral storage, Kafka can serve as a data hub where Beats can persist to and Logstash nodes can consume from.



The other TCP, UDP, and HTTP sources can persist to Kafka with Logstash as a conduit to achieve high availability in lieu of a load balancer. A group of Logstash nodes can then consume from topics with the Kafka input to further transform and enrich the data in transit.

Resiliency and Recovery

edit

When Logstash consumes from Kafka, persistent queues should be enabled and will add transport resiliency to mitigate the need for reprocessing during Logstash node failures. In this context, it's recommended to use the default persistent queue disk allocation size `queue.max_bytes: 1GB`.

If Kafka is configured to retain data for an extended period of time, data can be reprocessed from Kafka in the case of disaster recovery and reconciliation.

Other Messaging Queue Integrations

edit

Although an additional queuing layer is not required, Logstash can consume from a myriad of other message queuing technologies like RabbitMQ and Redis. It also supports ingestion from hosted queuing services like Pub/Sub, Kinesis, and SQS.

« Enriching Data with Lookups

Performance Tuning »