



ElasticSearch

Techmaster

Elasticsearch security principles



- ❖ Run Elasticsearch with security enabled
- ❖ Run Elasticsearch with a dedicated non-root user
- ❖ Protect Elasticsearch from public internet traffic
- ❖ Implement role based access control

- + Configuring security
- + Updating node security certificates
- + User authentication
- + User authorization
- + Enable audit logging
- + Restricting connections with IP filtering
- + Securing clients and integrations
- + Operator privileges
- + Troubleshooting

Limitations

Configure security for the Elastic Stack

Elastic Security Layers

**Elasticsearch
Development**

Minimal security



**Elasticsearch
Production**

Basic security



Elastic Stack

Basic security

+

TLS for REST



Minimal security



- ❖ Set up Elasticsearch on your laptop
- ❖ This configuration prevents unauthorized access to your local cluster by setting up passwords for the built-in users
- ❖ You also configure password authentication for Kibana

Minimum

```
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch
    container_name: elasticsearch
    environment:
      - node.name=elasticsearch
      - cluster.name=datasearch
      - bootstrap.memory_lock=true
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
      - xpack.security.enabled=true
      - xpack.security.transport.ssl.enabled=true
      - cluster.initial_master_nodes=elasticsearch
    ulimits:
      memlock:
        soft: -1
        hard: -1
    ports:
      - "9200:9200"
    volumes:
      - esdata:/usr/share/elasticsearch/data

  kibana:
    image: docker.elastic.co/kibana/kibana:7.10.1
    ports:
      - "5601:5601"
    environment:
      ELASTICSEARCH_USERNAME: kibana_system
      ELASTICSEARCH_PASSWORD: t6FcBXUMKKzVOBmEfDYK
```

```
[root@ff705c4c8935 bin]# ./elasticsearch-setup-passwords auto
Initiating the setup of passwords for reserved users elastic,apm_system,kibana,kibana_system,logstash_system,beats_system,remote_monitoring_user.
The passwords will be randomly generated and printed to the console.
Please confirm that you would like to continue [y/N]y
```

```
Please confirm that you would like to continue [y/N]y
```

```
Changed password for user apm_system
PASSWORD apm_system = Z3ZqpEEEXKsbBZJyvogK
```

```
Changed password for user kibana_system
PASSWORD kibana_system = t6FcBXUMKKzVOBmEfDYK
```

```
Changed password for user kibana
PASSWORD kibana = t6FcBXUMKKzVOBmEfDYK
```

```
Changed password for user logstash_system
PASSWORD logstash_system = 8YfqMYuTXzr9FY5C1gCl
```

```
Changed password for user beats_system
PASSWORD beats_system = DJt0ezuInCWunRYUcvFs
```

```
Changed password for user remote_monitoring_user
PASSWORD remote_monitoring_user = 4niergz00z3iFpSx7L0r
```

```
Changed password for user elastic
PASSWORD elastic = ixCC0DaebKQfTTkfLnG5
```

```
[root@ff705c4c8935 bin]#
```

Basic security



- ❖ This scenario builds on the minimal security requirements by adding transport Layer Security (TLS) for communication between nodes
- ❖ This additional layer requires that nodes verify security certificates, which prevents unauthorized nodes from joining your Elasticsearch cluster
- ❖ Your external HTTP traffic between Elasticsearch and Kibana won't be encrypted, but internode communication will be secured

Basic security

Understanding transport contexts

Transport Layer Security (TLS) is the name of an industry standard protocol for applying security controls (such as encryption) to network communications. TLS is the modern name for what used to be called Secure Sockets Layer (SSL). The Elasticsearch documentation uses the terms TLS and SSL interchangeably.

Transport Protocol is the name of the protocol that Elasticsearch nodes use to communicate with one another. This name is specific to Elasticsearch and distinguishes the transport port (default `9300`) from the HTTP port (default `9200`). Nodes communicate with one another using the transport port, and REST clients communicate with Elasticsearch using the HTTP port.

Although the word *transport* appears in both contexts, they mean different things. It's possible to apply TLS to both the Elasticsearch transport port and the HTTP port. We know that these overlapping terms can be confusing, so to clarify, in this scenario we're applying TLS to the Elasticsearch transport port. In [the next scenario](#), we'll apply TLS to the Elasticsearch HTTP port.

Basic security



Visitors
computer

Hello, is your connection secure?

Sure, I'm sending my certificate now! 



Server



Visitors
computer

Great! Ready to establish a safe connection?

I will encrypt the path now and decrypt it when it's safe!



Server

Basic security

Identity Information and
Public Key of Mario Rossi

Name: *Mario Rossi*
Organization: *Wikimedia*
Address: *via*
Country: *United States*



Public Key
of
Mario Rossi

Certificate Authority
verifies the identity of Mario Rossi
and encrypts with its Private Key



Certificate of Mario Rossi

Name: *Mario Rossi*
Organization: *Wikimedia*
Address: *via*
Country: *United States*
Validity: *1997/07/01 - 2047/06/30*




Public Key
of
Mario Rossi

Digital Signature
of the Certificate Authority

Digitally Signed by
Certificate Authority

Generate the certificate authority

- 
1. On any single node, use the `elasticsearch-certutil` tool to generate a CA for your cluster.

```
./bin/elasticsearch-certutil ca
```

- a. When prompted, accept the default file name, which is `elastic-stack-ca.p12`. This file contains the public certificate for your CA and the private key used to sign certificates for each node.

Generate the certificate authority

2. On any single node, generate a certificate and private key for the nodes in your cluster. You include the `elastic-stack-ca.p12` output file that you generated in the previous step.

```
./bin/elasticsearch-certutil cert --ca elastic-stack-ca.p12
```

`--ca <ca_file>`

Name of the CA file used to sign your certificates. The default file name from the `elasticsearch-certutil` tool is `elastic-stack-ca.p12`.

- a. Enter the password for your CA, or press **Enter** if you did not configure one in the previous step.
- b. Create a password for the certificate and accept the default file name.

The output file is a keystore named `elastic-certificates.p12`. This file contains a node certificate, node key, and CA certificate.

Generate the certificate authority

1. Open the `$ES_PATH_CONF/elasticsearch.yml` file and make the following changes:

a. Add the `cluster.name` setting and enter a name for your cluster:

```
cluster.name: my-cluster
```

b. Add the `node.name` setting and enter a name for the node. The node name defaults to the hostname of the machine when Elasticsearch starts.

```
node.name: node-1
```

c. Add the following settings to enable internode communication and provide access to the node's certificate.

Because you are using the same `elastic-certificates.p12` file on every node in your cluster, set the verification mode to `certificate`:

```
xpack.security.transport.ssl.enabled: true
xpack.security.transport.ssl.verification_mode: certificate ❶
xpack.security.transport.ssl.client_authentication: required
xpack.security.transport.ssl.keystore.path: elastic-certificates.p12
xpack.security.transport.ssl.truststore.path: elastic-certificates.p12
```

Basic security



- ❖ This scenario builds on the minimal security requirements by adding transport Layer Security (TLS) for communication between nodes
- ❖ This additional layer requires that nodes verify security certificates, which prevents unauthorized nodes from joining your Elasticsearch cluster
- ❖ Your external HTTP traffic between Elasticsearch and Kibana won't be encrypted, but internode communication will be secured

Elastic Stack plus secured HTTPS traffic

Signing certificates

The first question that the `elasticsearch-certutil` tool prompts you with is whether you want to generate a Certificate Signing Request (CSR). Answer `n` if you want to sign your own certificates, or `y` if you want to sign certificates with a central CA.

Sign your own certificates

If you want to use the CA that you created when [Generating the certificate authority](#), answer `n` when asked if you want to generate a CSR. You then specify the location of your CA, which the tool uses to sign and generate a `.p12` certificate. The steps in this procedure follow this workflow.

Sign certificates with a central CA

If you work in an environment with a central security team, they can likely generate a certificate for you. Infrastructure within your organization might already be configured to trust an existing CA, so it may be easier for clients to connect to Elasticsearch if you use a CSR and send that request to the team that controls your CA. To use a central CA, answer `y` to the first question.

<https://www.elastic.co/guide/en/elasticsearch/reference/current/security-basic-setup-https.html>

User Authentication



Authentication identifies an individual. To gain access to restricted resources, a user must prove their identity, via passwords, credentials, or some other means (typically referred to as authentication tokens).

The Elastic Stack authenticates users by identifying the users behind the requests that hit the cluster and verifying that they are who they claim to be. The authentication process is handled by one or more authentication services called *realms*.

You can use the native support for managing and authenticating users, or integrate with external user management systems such as LDAP and Active Directory.

The Elastic Stack security features provide built-in realms such as `native`, `ldap`, `active_directory`, `pki`, `file`, `saml`, and `oidc`. If none of the built-in realms meet your needs, you can also build your own custom realm and plug it into the Elastic Stack.

Built-in users



The Elastic Stack security features provide built-in user credentials to help you get up and running. These users have a fixed set of privileges and cannot be authenticated until their passwords have been set. The `elastic` user can be used to [set all of the built-in user passwords](#).

`elastic`

A built-in [superuser](#).

`kibana_system`

The user Kibana uses to connect and communicate with Elasticsearch.

`logstash_system`

The user Logstash uses when storing monitoring information in Elasticsearch.

`beats_system`

The user the Beats use when storing monitoring information in Elasticsearch.

`apm_system`

The user the APM server uses when storing monitoring information in Elasticsearch.

Realms



The Elastic Stack security features authenticate users by using realms and one or more token-based authentication services.

A realm is used to resolve and authenticate users based on authentication tokens. The security features provide the following built-in realms

- ❖ Native
- ❖ Ldap
- ❖ Active directory
- ❖ pki
- ❖ file
- ❖ saml

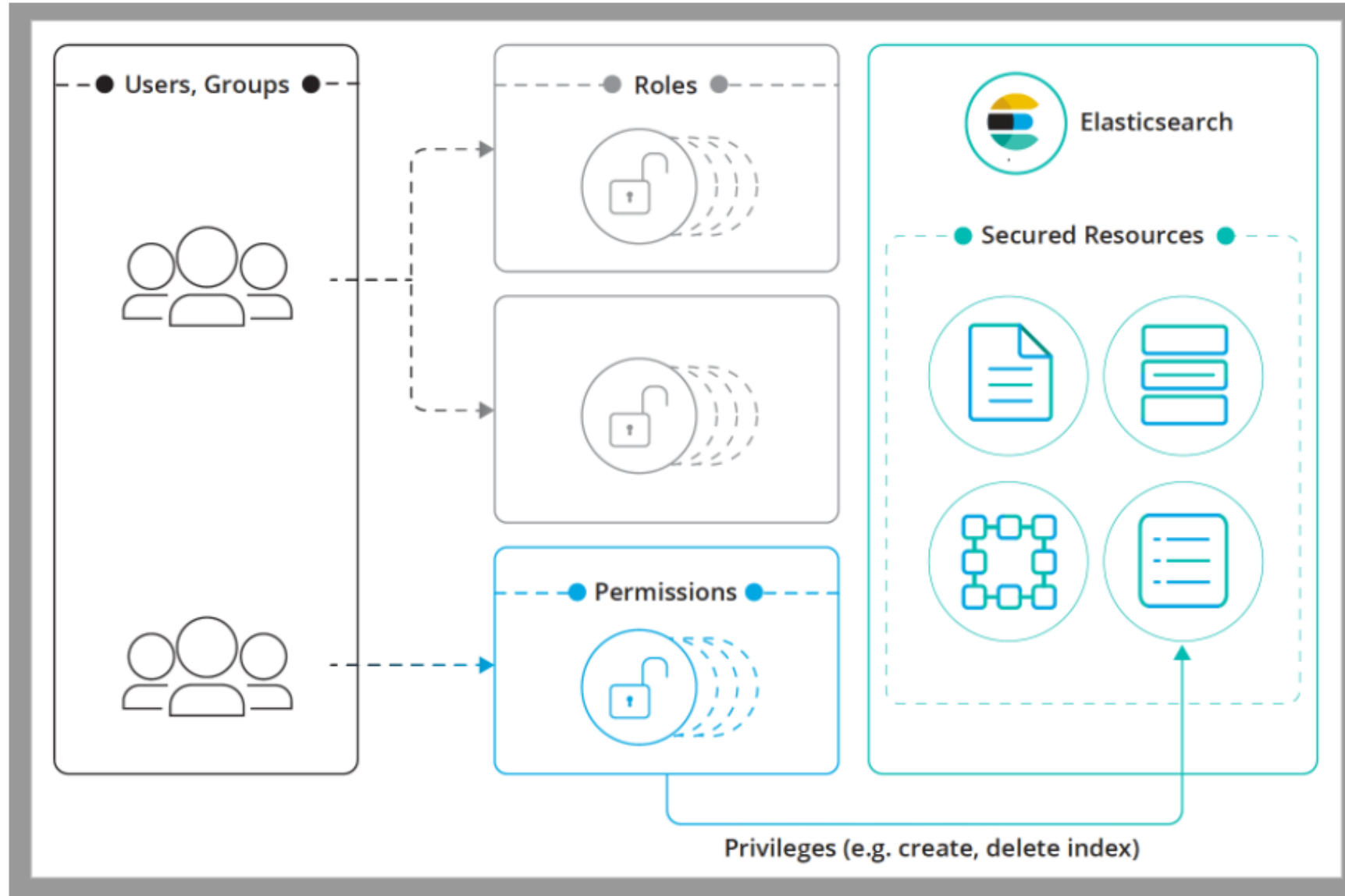
User authorization



The Elastic Stack security features add *authorization*, which is the process of determining whether the user behind an incoming request is allowed to execute the request.

This process takes place after the user is successfully identified and [authenticated](#).

Role based access control



Audit logging



You can log security-related events such as authentication failures and refused connections to monitor your cluster for suspicious activity (including data access authorization and user security configuration changes).

Audit logging also provides forensic evidence in the event of an attack.

To enable audit logging:

1. Set `xpack.security.audit.enabled` to `true` in `elasticsearch.yml`.
2. Restart Elasticsearch.

Bài toán



Require

- ❖ 1000 Tps
- ❖ Msg size: 2kB-5kB
- ❖ 1 ngày 30GB
- ❖ 30 ngày

Sizing

- ❖ 6 server
- ❖ Intel(R) Xeon(R) Bronze 3206R CPU @ 1.90GHz Socket(2)x8
- ❖ 180GB RAM
- ❖ 1TB
- ❖ 1 card 4 port

Restricting connections with IP filtering

You can apply IP filtering to application clients, node clients, or transport clients, in addition to other nodes that are attempting to join the cluster.

If a node's IP address is on the blacklist, the Elasticsearch security features allow the connection to Elasticsearch but it is be dropped immediately and no requests are processed.

```
xpack.security.transport.filter.allow: "192.168.0.1"  
xpack.security.transport.filter.deny: "192.168.0.0/24"
```

The `_all` keyword can be used to deny all connections that are not explicitly allowed.

```
xpack.security.transport.filter.allow: [ "192.168.0.1", "192.168.0.2", "192.168.0.3"  
xpack.security.transport.filter.deny: _all
```

Requirement for resiliency

| Category | Requirement |
|--------------------------------|--|
| Backup | <ul style="list-style-type: none">• The system must be recoverable after a loss of the stored data. To do this, the system must be able to perform both automatic full backups and incremental backups of the data sets.• The system must implement appropriate mechanisms that enable automated, complete and consistent recovery of the entire database.• The system must be able to be fully restored within two hours of a system failure at the latest in order to ensure system availability. |
| Fault Tolerance and management | <ul style="list-style-type: none">• The system must ensure that all stored data is fully and consistently preserved in the event of an error or failure.• The system must ensure that after a system failure, the data set of the platform can be fully restored to the level before the fault.• The system must ensure that no inconsistent data sets occur after the break or after a system function failure.• Appropriate testing and validation procedures must be integrated for the interface data (content data) that can check syntactic correctness (for available XML and JSON schemas).• A rejection of interface data should only take place in the event of massive technical defects or in the event of security risks. All exam results are to be logged |

Requirement for resiliency

| | |
|--------------------------------|--|
| High Availability | <ul style="list-style-type: none">• The system and its operation should be designed for a statefully, highly available operation.• The system should be available without interruption in the sense of a 24x7 (24 hours operation, 7 days per week, 365 days per year) use and secured by corresponding SLAs.• The system must achieve a proven availability of all system functions of at least 99.5% in the annual calendar year.• If the mobility data platform is only available in subareas, but these are a prerequisite for the usability of the system by data providers and users, the system shall be deemed to be unavailable.• The system must achieve a proven availability of the data broker of at least 99.9% in the annual calendar year.• The system must ensure that the data broker is available again within 30 minutes in the event of a failure.• The system must allow operation without a regular war window. Basic functions of the system, in particular the interfaces to data providers and data controllers, must be able to remain available during maintenance work, the execution of batch processes and data backups.• The system is designed to support software patches, upgrades, and version changes during operation, for example, by gradually modifying individual clusters. |
| Protection against cyberattack | <ul style="list-style-type: none">• A backup solution as a contingency to cyberattack.• Data needs to be backed up |
| Scalability | <ul style="list-style-type: none">• Scalable to support all stages of the implementation, including development, test, and production.• Sufficient scalability to support delivery of high volume of vaccines to large scale of users |
| Disaster Recovery | <ul style="list-style-type: none">• Provide resiliency to disasters impacting the hosting facility and must provide the ability for hosting to be transferred to an alternative data center. |

Requirement for resiliency of ES



There is a limit to how small a resilient cluster can be. All Elasticsearch clusters require:

- One [elected master node](#) node
- At least one node for each [role](#).
- At least one copy of every [shard](#).

A resilient cluster requires redundancy for every required cluster component. This means a resilient cluster must have:

- At least three master-eligible nodes
- At least two nodes of each role
- At least two copies of each shard (one primary and one or more replicas, unless the index is a [searchable snapshot index](#))

Requirement for small clusters

One-node clusters



If your cluster consists of one node, that single node must do everything. To accommodate this, Elasticsearch assigns nodes every role by default.

A single node cluster is not resilient. If the node fails, the cluster will stop working. Because there are no replicas in a one-node cluster, you cannot store your data redundantly. However, by default at least one replica is required for a [green cluster health status](#). To ensure your cluster can report a [green](#) status, override the default by setting `index.number_of_replicas` to `0` on every index.

Two-node clusters



If you have two nodes, we recommend they both be data nodes. You should also ensure every shard is stored redundantly on both nodes by setting `index.number_of_replicas` to `1` on every index that is not a [searchable snapshot index](#). This is the default behaviour but may be overridden by an [index template](#). [Auto-expand replicas](#) can also achieve the same thing, but it's not necessary to use this feature in such a small cluster.

Two-node clusters with a tiebreaker

Requirement for small clusters

Three-node clusters



If you have three nodes, we recommend they all be [data nodes](#) and every index that is not a [searchable snapshot index](#) should have at least one replica. Nodes are data nodes by default. You may prefer for some indices to have two replicas so that each node has a copy of each shard in those indices. You should also configure each node to be [master-eligible](#) so that any two of them can hold a master election without needing to communicate with the third node. Nodes are master-eligible by default. This cluster will be resilient to the loss of any single node.

Clusters with more than three nodes



Once your cluster grows to more than three nodes, you can start to specialise these nodes according to their responsibilities, allowing you to scale their resources independently as needed. You can have as many [data nodes](#), [ingest nodes](#), [machine learning nodes](#), etc. as needed to support your workload. As your cluster grows larger, we recommend using dedicated nodes for each role. This allows you to independently scale resources for each task.

However, it is good practice to limit the number of master-eligible nodes in the cluster to three. Master nodes do not scale like other node types since the cluster always elects just one of them as the master of the cluster. If there are too many master-eligible nodes then master elections

Summary



The cluster will be resilient to the loss of any node as long as:

- The [cluster health status](#) is `green`.
- There are at least two data nodes.
- Every index that is not a [searchable snapshot index](#) has at least one replica of each shard, in addition to the primary.
- The cluster has at least three master-eligible nodes, as long as at least two of these nodes are not voting-only master-eligible nodes.
- Clients are configured to send their requests to more than one node or are configured to use a load balancer that balances the requests across an appropriate set of nodes. The [Elastic Cloud](#) service provides such a load balancer.


Summary



The cluster will be resilient to the loss of any zone as long as:

- The [cluster health status](#) is `green`.
- There are at least two zones containing data nodes.
- Every index that is not a [searchable snapshot index](#) has at least one replica of each shard, in addition to the primary.
- Shard allocation awareness is configured to avoid concentrating all copies of a shard within a single zone.
- The cluster has at least three master-eligible nodes. At least two of these nodes are not voting-only master-eligible nodes, and they are spread evenly across at least three zones.
- Clients are configured to send their requests to nodes in more than one zone or are configured to use a load balancer that balances the requests across an appropriate set of nodes. The [Elastic Cloud](#) service provides such a load balancer.

Requirement for larger clusters



It's not unusual for nodes to share common infrastructure, such as network interconnects or a power supply. If so, you should plan for the failure of this infrastructure and ensure that such a failure would not affect too many of your nodes. It is common practice to group all the nodes sharing some infrastructure into *zones* and to plan for the failure of any whole zone at once.

Elasticsearch expects node-to-node connections to be reliable, have low latency, and have adequate bandwidth. Many Elasticsearch tasks require multiple round-trips between nodes. A slow or unreliable interconnect may have a significant effect on the performance and stability of your cluster.

Two-zone clusters



If you have two zones, you should have a different number of master-eligible nodes in each zone so that the zone with more nodes will contain a majority of them and will be able to survive the loss of the other zone. For instance, if you have three master-eligible nodes then you may put all of them in one zone or you may put two in one zone and the third in the other zone. You should not place an equal number of master-eligible nodes in each zone. If you place the same number of master-eligible nodes in each zone, neither zone has a majority of its own. Therefore, the cluster may not survive the loss of either zone.

Multi cluster Architecture



Use cross-cluster replication to construct several multi-cluster architectures within the Elastic Stack:

- [Disaster recovery](#) in case a primary cluster fails, with a secondary cluster serving as a hot backup
- [Data locality](#) to maintain multiple copies of the dataset close to the application servers (and users), and reduce costly latency
- [Centralized reporting](#) for minimizing network traffic and latency in querying multiple geo-distributed Elasticsearch clusters, or for preventing search load from interfering with indexing by offloading search to a secondary cluster

Disaster recovery and high availability



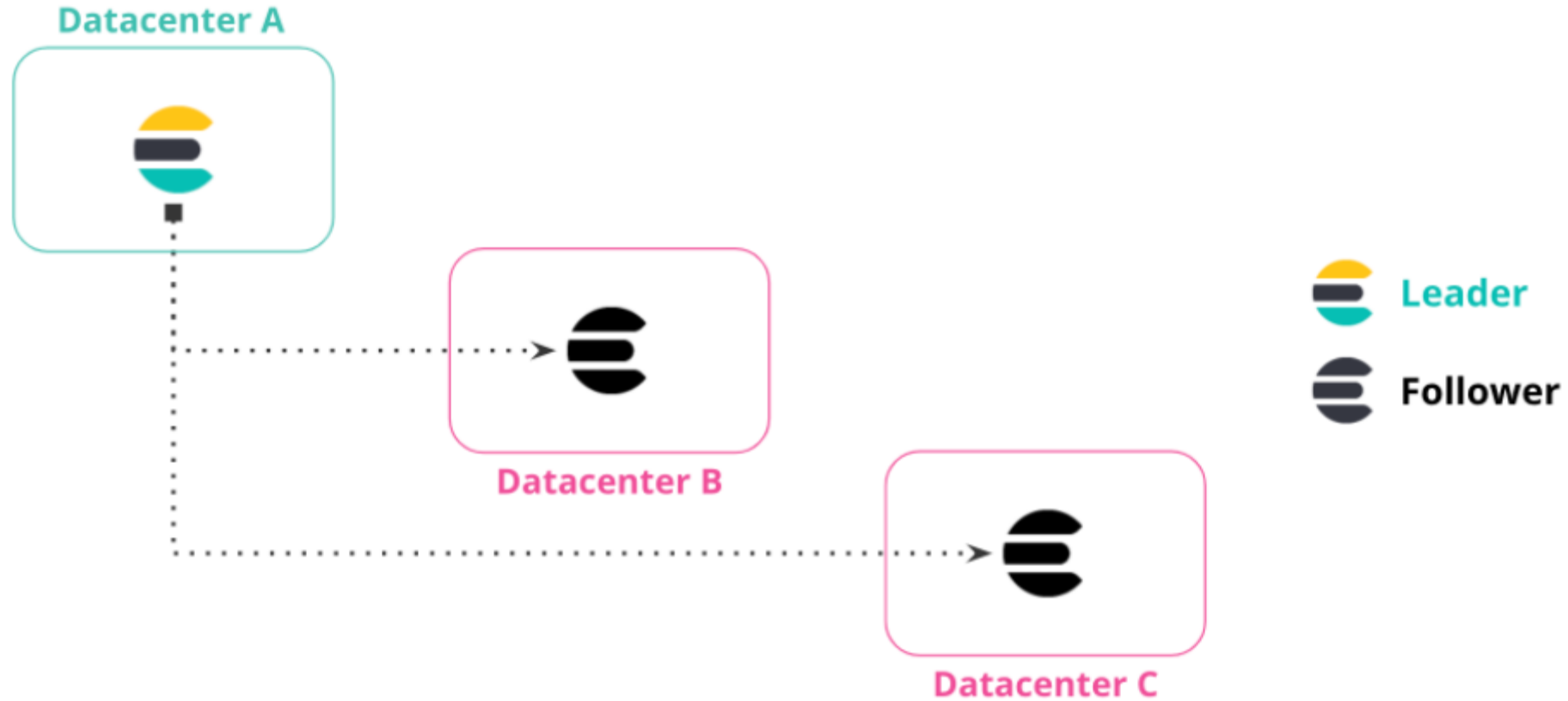
Disaster recovery provides your mission-critical applications with the tolerance to withstand datacenter or region outages. This use case is the most common deployment of cross-cluster replication. You can configure clusters in different architectures to support disaster recovery and high availability:

- [Single disaster recovery datacenter](#)
- [Multiple disaster recovery datacenters](#)
- [Chained replication](#)
- [Bi-directional replication](#)

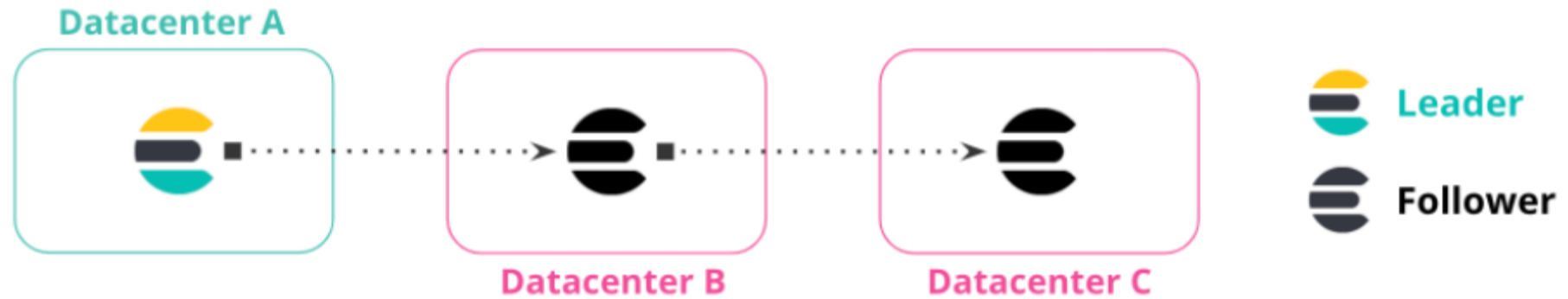
Single disaster recovery datacenter



Multi disaster recovery datacenter



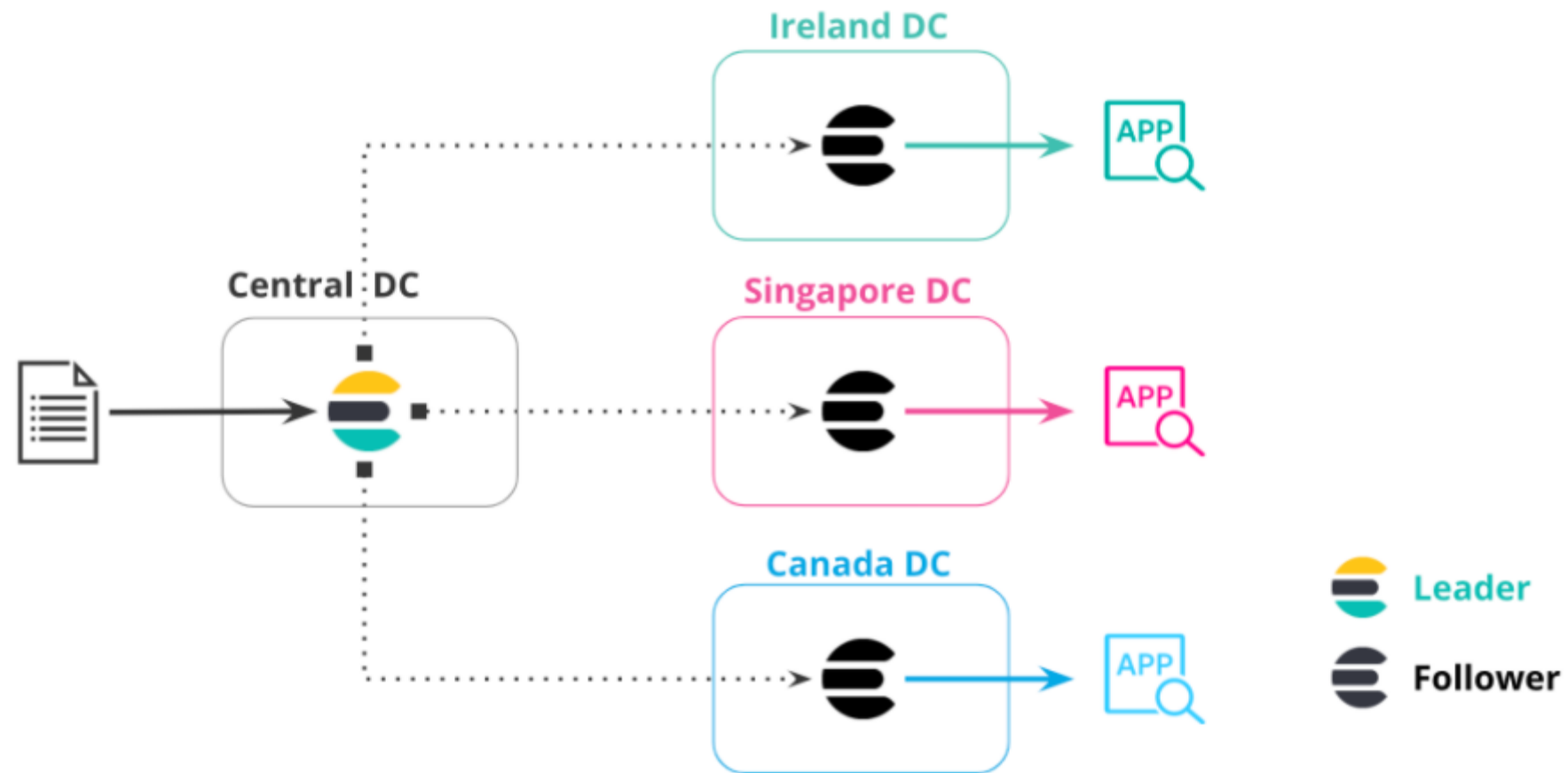
Chained replication



Bi-directional replication



Data locality



Centralized reporting

