**Elasticsearch优化及升级**

1. **Elasticsearch优化**
2. 内存优化**(系统内存的50%，最大32G)**

-Xmx8g -Xms8g

1. 禁止交换区和最大映射文件数（/etc/sysctl.conf）

vm.swappiness=1

net.core.somaxconn=65535

vm.max\_map\_count=262144

fs.file-max=518144

1. 内存锁定以及文件句柄优化（/etc/security/limits.conf）

user soft nofile 65536

user hard nofile 131072

user soft nproc 2048

user hard nproc 4096

user soft memlock unlimited

user hard memlock unlimited

1. 索引刷新时间

PUT /my\_logs

{

"settings": {

"refresh\_interval": "30s"

}

}

PUT /my\_index/\_settings

{

"index.translog.durability": "async",

"index.translog.sync\_interval": "5s"

}

1. 集群单播优化(elasticsearch.yml)

# ---------------------------------- Cluster -----------------------------------

cluster.name: elasticsearch

# ------------------------------------ Node ------------------------------------

node.name: elasticsearch-node1

# ----------------------------------- Paths ------------------------------------

# ----------------------------------- Memory -----------------------------------

bootstrap.memory\_lock: true

bootstrap.system\_call\_filter:false

# ---------------------------------- Network -----------------------------------

# network.host: 192.168.0.1

http.port: 9200

http.compression: true

transport.tcp.port: 9300

transport.tcp.compress: true

# --------------------------------- Discovery ----------------------------------

#discovery.zen.minimum\_master\_nodes: 3

discovery.zen.ping\_timeout: 10s

discovery.zen.fd.ping\_retries: 3

discovery.zen.fd.ping\_interval: 3s

discovery.zen.fd.ping\_timeout: 30s

# ---------------------------------- Gateway -----------------------------------

# gateway.recover\_after\_nodes: 3

# ---------------------------------- Various -----------------------------------

action.destructive\_requires\_name: true //是否禁用全局通配符操作

action.auto\_create\_index: false //是否自动创建索引

xpack.security.enabled: false //是否启用安全认证

# ---------------------------------- Indices -----------------------------------

indices.fielddata.cache.size: 30%

indices.query.bool.max\_clause\_count: 100000

index.number\_of\_shards: 5

index.number\_of\_replicas: 1

index.translog.sync\_interval: 30s

index.merge.scheduler.max\_thread\_count: 1

index.refresh\_interval:60s

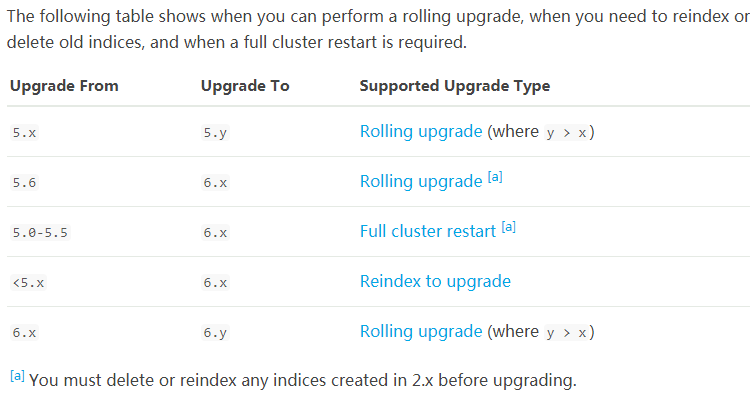
index.translog.flush\_threshold\_ops: 1000000

index.translog.durability: async

index.merge.policy.floor\_segment": 100mb

index.merge.scheduler.max\_thread\_count: 1

index.merge.policy.min\_merge\_size: 10mb



1. **Rolling upgrades**

A rolling upgrade allows an Elasticsearch cluster to be upgraded one node at a time so upgrading does not interrupt service. Running multiple versions of Elasticsearch in the same cluster beyond the duration of an upgrade is not supported, as shards cannot be replicated from upgraded nodes to nodes running the older version.

Rolling upgrades can be performed between minor versions. Elasticsearch 6.x supports rolling upgrades from **Elasticsearch 5.6**. Upgrading from earlier 5.x versions requires a [full cluster restart](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/restart-upgrade.html). You must [reindex to upgrade](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/reindex-upgrade.html) from versions prior to 5.x.

To perform a rolling upgrade:

1. **Disable shard allocation**.

When you shut down a node, the allocation process waits for one minute before starting to replicate the shards on that node to other nodes in the cluster, causing a lot of wasted I/O. You can avoid racing the clock by disabling allocation before shutting down the node:

PUT \_cluster/settings

{

"persistent": {

"cluster.routing.allocation.enable": "none"

}

}

1. **Stop non-essential indexing and perform a synced flush.** (Optional)

While you can continue indexing during the upgrade, shard recovery is much faster if you temporarily stop non-essential indexing and perform a [synced-flush](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/indices-synced-flush.html).

POST \_flush/synced

When you perform a synced flush, check the response to make sure there are no failures. Synced flush operations that fail due to pending indexing operations are listed in the response body, although the request itself still returns a 200 OK status. If there are failures, reissue the request.

1. **Shut down a single node**.
   * If you are running Elasticsearch with systemd:

sudo systemctl stop elasticsearch.service

* + If you are running Elasticsearch with SysV init:

sudo -i service elasticsearch stop

* + If you are running Elasticsearch as a daemon:

kill $(cat pid)

1. **Upgrade the node you shut down.**

To upgrade using a [Debian](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/deb.html) or [RPM](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/rpm.html) package:

* + Use rpm or dpkg to install the new package. All files are installed in the appropriate location for the operating system and Elasticsearch config files are not overwritten.

To upgrade using a zip or compressed tarball:

* 1. Extract the zip or tarball to a *new* directory. This is critical if you are not using external config and data directories.
  2. Set the ES\_PATH\_CONF environment variable to specify the location of your external config directory and jvm.options file. If you are not using an external configdirectory, copy your old configuration over to the new installation.
  3. Set path.data in config/elasticsearch.yml to point to your external data directory. If you are not using an external data directory, copy your old data directory over to the new installation.
  4. Set path.logs in config/elasticsearch.yml to point to the location where you want to store your logs. If you do not specify this setting, logs are stored in the directory you extracted the archive to.

|  |
| --- |
| When you extract the zip or tarball packages, the elasticsearch-n.n.ndirectory contains the Elasticsearh config, data, logs and plugins directories.  We recommend moving these directories out of the Elasticsearch directory so that there is no chance of deleting them when you upgrade Elasticsearch. To specify the new locations, use the ES\_PATH\_CONF environment variable and the path.data and path.logs settings. For more information, see [Important Elasticsearch configuration](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/important-settings.html).  The [Debian](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/deb.html) and [RPM](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/rpm.html) packages place these directories in the appropriate place for each operating system. In production, we recommend installing using the deb or rpm package. |

1. **Upgrade any plugins.**

Use the elasticsearch-plugin script to install the upgraded version of each installed Elasticsearch plugin. All plugins must be upgraded when you upgrade a node.

1. **Start the upgraded node.**

Start the newly-upgraded node and confirm that it joins the cluster by checking the log file or by submitting a \_cat/nodes request:

GET \_cat/nodes

1. **Reenable shard allocation.**

Because [transient settings take precedence over persistent settings](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/cluster-update-settings.html#_precedence_of_settings), this overrides the persistent setting used to disable shard allocation in the first step. If you don’t explicitly reenable shard allocation after a full cluster restart, the persistent setting is used and shard allocation remains disabled.

Once the node has joined the cluster, reenable shard allocation to start using the node:

PUT \_cluster/settings

{

"transient": {

"cluster.routing.allocation.enable": "all"

}

}

1. **Wait for the node to recover.**

Before upgrading the next node, wait for the cluster to finish shard allocation. You can check progress by submitting a [\_cat/health](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/cat-health.html) request:

GET \_cat/health

Wait for the status column to switch from yellow to green. Once the node is green, all primary and replica shards have been allocated.

|  |
| --- |
| During a rolling upgrade, primary shards assigned to a node running the new version cannot have their replicas assigned to a node with the old version. The new version might have a different data format that is not understood by the old version.  If it is not possible to assign the replica shards to another node (there is only one upgraded node in the cluster), the replica shards remain unassigned and status stays yellow.  In this case, you can proceed once there are no initializing or relocating shards (check the init and relo columns).  As soon as another node is upgraded, the replicas can be assigned and the status will change to green. |

Shards that were not [sync-flushed](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/indices-synced-flush.html) might take longer to recover. You can monitor the recovery status of individual shards by submitting a [\_cat/recovery](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/cat-recovery.html) request:

GET \_cat/recovery

If you stopped indexing, it is safe to resume indexing as soon as recovery completes.

1. **Repeat**

When the node has recovered and the cluster is stable, repeat these steps for each node that needs to be updated.

|  |
| --- |
| During a rolling upgrade, the cluster continues to operate normally. However, any new functionality is disabled or operates in a backward compatible mode until all nodes in the cluster are upgraded. New functionality becomes operational once the upgrade is complete and all nodes are running the new version. Once that has happened, there’s no way to return to operating in a backward compatible mode. Nodes running the previous major version will not be allowed to join the fully-updated cluster.  In the unlikely case of a network malfunction during the upgrade process that isolates all remaining old nodes from the cluster, you must take the old nodes offline and upgrade them to enable them to join the cluster. |

1. **Full cluster restart upgrade**

A full cluster restart upgrade requires that you shut all nodes in the cluster down, upgrade them, and restart the cluster. A full cluster restart was required when upgrading to major versions prior to 6.x. Elasticsearch 6.x supports [rolling upgrades](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/rolling-upgrades.html) from **Elasticsearch 5.6**. Upgrading to 6.x from earlier versions requires a full cluster restart. See the [Upgrade paths table](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/setup-upgrade.html#upgrade-paths) to verify the type of upgrade you need to perform.

To perform a full cluster restart upgrade:

1. **Disable shard allocation.**

When you shut down a node, the allocation process waits for one minute before starting to replicate the shards on that node to other nodes in the cluster, causing a lot of wasted I/O. You can avoid racing the clock by disabling allocation before shutting down the node:

PUT \_cluster/settings

{

"persistent": {

"cluster.routing.allocation.enable": "none"

}

}

1. **Stop indexing and perform a synced flush.**

Performing a [synced-flush](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/indices-synced-flush.html) speeds up shard recovery.

POST \_flush/synced

When you perform a synced flush, check the response to make sure there are no failures. Synced flush operations that fail due to pending indexing operations are listed in the response body, although the request itself still returns a 200 OK status. If there are failures, reissue the request.

1. **Shutdown all nodes.**
   * If you are running Elasticsearch with systemd:

sudo systemctl stop elasticsearch.service

* + If you are running Elasticsearch with SysV init:

sudo -i service elasticsearch stop

* + If you are running Elasticsearch as a daemon:

kill $(cat pid)

1. **Upgrade all nodes.**

To upgrade using a [Debian](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/deb.html) or [RPM](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/rpm.html) package:

* + Use rpm or dpkg to install the new package. All files are installed in the appropriate location for the operating system and Elasticsearch config files are not overwritten.

To upgrade using a zip or compressed tarball:

* + Extract the zip or tarball to a *new* directory. This is critical if you are not using external config and data directories.
  + Set the ES\_PATH\_CONF environment variable to specify the location of your external config directory and jvm.options file. If you are not using an external configdirectory, copy your old configuration over to the new installation.
  + Set path.data in config/elasticsearch.yml to point to your external data directory. If you are not using an external data directory, copy your old data directory over to the new installation.
  + Set path.logs in config/elasticsearch.yml to point to the location where you want to store your logs. If you do not specify this setting, logs are stored in the directory you extracted the archive to.

|  |
| --- |
| When you extract the zip or tarball packages, the elasticsearch-n.n.ndirectory contains the Elasticsearh config, data, logs and plugins directories.  We recommend moving these directories out of the Elasticsearch directory so that there is no chance of deleting them when you upgrade Elasticsearch. To specify the new locations, use the ES\_PATH\_CONF environment variable and the path.data and path.logs settings. For more information, see [Important Elasticsearch configuration](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/important-settings.html).  The [Debian](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/deb.html) and [RPM](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/rpm.html) packages place these directories in the appropriate place for each operating system. In production, we recommend installing using the deb or rpm package. |

1. **Upgrade any plugins.**

Use the elasticsearch-plugin script to install the upgraded version of each installed Elasticsearch plugin. All plugins must be upgraded when you upgrade a node.

1. **Start each upgraded node.**

If you have dedicated master nodes, start them first and wait for them to form a cluster and elect a master before proceeding with your data nodes. You can check progress by looking at the logs.

As soon as the [minimum number of master-eligible nodes](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/modules-discovery-zen.html#master-election) have discovered each other, they form a cluster and elect a master. At that point, you can use [\_cat/health](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/cat-health.html) and[\_cat/nodes](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/cat-nodes.html) to monitor nodes joining the cluster:

GET \_cat/health

GET \_cat/nodes

The status column returned by \_cat/health shows the health of each node in the cluster: red, yellow, or green.

1. **Wait for all nodes to join the cluster and report a status of yellow.**

When a node joins the cluster, it begins to recover any primary shards that are stored locally. The [\_cat/health](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/cat-health.html) API initially reports a status of red, indicating that not all primary shards have been allocated.

Once a node recovers its local shards, the cluster status switches to yellow, indicating that all primary shards have been recovered, but not all replica shards are allocated. This is to be expected because you have not yet reenabled allocation. Delaying the allocation of replicas until all nodes are yellow allows the master to allocate replicas to nodes that already have local shard copies.

1. **Reenable allocation.**

When all nodes have joined the cluster and recovered their primary shards, reenable allocation.

PUT \_cluster/settings

{

"transient": {

"cluster.routing.allocation.enable": "all"

}

}

|  |
| --- |
| Because [transient settings take precedence over persistent settings](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/cluster-update-settings.html#_precedence_of_settings), this overrides the persistent setting used to disable shard allocation in the first step. If you don’t explicitly reenable shard allocation after a full cluster restart, the persistent setting is used and shard allocation remains disabled. |

Once allocation is reenabled, the cluster starts allocating replica shards to the data nodes. At this point it is safe to resume indexing and searching, but your cluster will recover more quickly if you can wait until all primary and replica shards have been successfully allocated and the status of all nodes is green.

You can monitor progress with the [\_cat/health](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/cat-health.html) and [\_cat/recovery](https://www.elastic.co/guide/en/elasticsearch/reference/6.0/cat-recovery.html) APIs:

GET \_cat/health

GET \_cat/recovery