Frequently Asked Questions

Why can't I set `listen_address` to listen on 0.0.0.0 (all my addresses)?

Cassandra is a gossip-based distributed system and `listen_address` is the address a node tells other nodes to reach it at. Telling other nodes "contact me on any of my addresses" is a bad idea; if different nodes in the cluster pick different addresses for you, Bad Things happen.

If you don't want to manually specify an IP to `listen_address` for each node in your cluster (understandable!), leave it blank and Cassandra will use `InetAddress.getLocalHost()` to pick an address. Then it's up to you or your ops team to make things resolve correctly (`/etc/hosts/` , dns, etc).

One exception to this process is JMX, which by default binds to 0.0.0.0 (Java bug 6425769).

See `256` and `43` for more gory details.

What ports does Cassandra use?

By default, Cassandra uses 7000 for cluster communication (7001 if SSL is enabled), 9042 for native protocol clients, and 7199 for JMX. The internode communication and native protocol ports are configurable in the `cassandra-yaml` . The JMX port is configurable in `cassandra-env.sh` (through JVM options). All ports are TCP.

What happens to existing data in my cluster when I add new nodes?

When a new nodes joins a cluster, it will automatically contact the other nodes in the cluster and copy the right data to itself. See `topology-changes` .

I delete data from Cassandra, but disk usage stays the same. What gives?

Data you write to Cassandra gets persisted to SSTables. Since SSTables are immutable, the data can't actually be removed when you perform a delete, instead, a marker (also called a "tombstone") is written to indicate the value's new status. Never fear though, on the first compaction that occurs between the data and the tombstone, the data will be expunged completely and the corresponding disk space recovered. See `compaction` for more detail.

Why does nodetool ring only show one entry, even though my nodes logged that they see each other joining the ring?

This happens when you have the same token assigned to each node. Don't do that.

Most often this bites people who deploy by installing Cassandra on a VM (especially when using the Debian package, which auto-starts Cassandra after installation, thus generating and saving a token), then cloning that VM to other nodes.

The easiest fix is to wipe the data and commitlog directories, thus making sure that each node will generate a random token on the next restart.

Can I change the replication factor (a a keyspace) on a live cluster?

Yes, but it will require running a full repair (or cleanup) to change the replica count of existing data:

- `Alter <alter-keyspace-statement>` the replication factor for desired keyspace (using cqlsh for instance).
- If you're reducing the replication factor, run `nodetool cleanup` on the cluster to remove surplus replicated data. Cleanup runs on a per-node basis.

- If you're increasing the replication factor, run `nodetool repair -full` to ensure data is replicated according to the new configuration. Repair runs on a per-replica set basis. This is an intensive process that may result in adverse cluster performance. It's highly recommended to do rolling repairs, as an attempt to repair the entire cluster at once will most likely swamp it. Note that you will need to run a full repair (`-full`) to make sure that already repaired sstables are not skipped. You should use `ConsistencyLevel.QUORUM` or `ALL` (depending on your existing replication factor) to make sure that a replica that actually has the data is consulted. Otherwise some clients potentially being told no data exists until repair is done.

Can I Store (large) BLOBs in Cassandra?

Cassandra isn't optimized for large file or BLOB storage and a single `blob` value is always read and send to the client entirely. As such, storing small blobs (less than single digit MB) should not be a problem, but it is advised to manually split large blobs into smaller chunks.

Please note in particular that by default, any value greater than 16MiB will be rejected by Cassandra due the `max_mutation_size` configuration of the `cassandra-yaml` file (which default to half of `commitlog_segment_size`, which itself default to 32MiB).

Nodetool says "Connection refused to host: 127.0.1.1" for any remote host. What gives?

Nodetool relies on JMX, which in turn relies on RMI, which in turn sets up its own listeners and connectors as needed on each end of the exchange. Normally all of this happens behind the scenes transparently, but incorrect name resolution for either the host connecting, or the one being connected to, can result in crossed wires and confusing exceptions.

If you are not using DNS, then make sure that your `/etc/hosts` files are accurate on both ends. If that fails, try setting the `-Djava.rmi.server.hostname=<public name>` JVM option near the bottom of `cassandra-env.sh` to an interface that you can reach from the remote machine.

Will batching my operations speed up my bulk load?

No. Using batches to load data will generally just add "spikes" of latency. Use asynchronous INSERTs instead, or use true `bulk-loading`.

An exception is batching updates to a single partition, which can be a Good Thing (as long as the size of a single batch stay reasonable). But never ever blindly batch everything!

On RHEL nodes are unable to join the ring

Check if SELinux is on; if it is, turn it off.

How do I unsubscribe from the email list?

Send an email to `user-unsubscribe@cassandra.apache.org`.

Why does top report that Cassandra is using a lot more memory than the Java heap max?

Cassandra uses Memory Mapped Files (mmap) internally. That is, we use the operating system's virtual memory system to map a number of on-disk files into the Cassandra process' address space. This will "use" virtual memory; i.e. address space, and will be reported by tools like top accordingly, but on 64 bit systems virtual address space is effectively unlimited so you should not worry about that.

What matters from the perspective of "memory use" in the sense as it is normally meant, is the amount of data allocated on brk() or mmap'd /dev/zero, which represent real memory used. The key issue is that for a mmap'd file, there is never a need to retain the data resident in physical memory. Thus, whatever you do keep resident in physical memory is essentially just there as a cache, in the same way as normal I/O will cause the kernel page cache to retain data that you read/write.

The difference between normal I/O and mmap() is that in the mmap() case the memory is actually mapped to the process, thus affecting the virtual size as reported by top. The main argument for using mmap() instead of standard I/O is the fact that reading entails just touching memory - in the case of the memory being resident, you just read it - you don't even take a page fault (so no overhead in entering the kernel and doing a semi-context switch). This is covered in more detail here.

What are seeds?

Seeds are used during startup to discover the cluster.

If you configure your nodes to refer some node as seed, nodes in your ring tend to send Gossip message to seeds more often (also see the `section on gossip <gossip>`) than to non-seeds. In other words, seeds are worked as hubs of Gossip network. With seeds, each node can detect status changes of other nodes quickly.

Seeds are also referred by new nodes on bootstrap to learn other nodes in ring. When you add a new node to ring, you need to specify at least one live seed to contact. Once a node join the ring, it learns about the other nodes, so it doesn't need seed on subsequent boot.

You can make a seed a node at any time. There is nothing special about seed nodes. If you list the node in seed list it is a seed

Seeds do not auto bootstrap (i.e. if a node has itself in its seed list it will not automatically transfer data to itself) If you want a node to do that, bootstrap it first and then add it to seeds later. If you have no data (new install) you do not have to worry about bootstrap at all.

Recommended usage of seeds:

- pick two (or more) nodes per data center as seed nodes.
- sync the seed list to all your nodes

Does single seed mean single point of failure?

The ring can operate or boot without a seed; however, you will not be able to add new nodes to the cluster. It is recommended to configure multiple seeds in production system.

Why can't I call jmx method X on jconsole?

Some of JMX operations use array argument and as jconsole doesn't support array argument, those operations can't be called with jconsole (the buttons are inactive for them). You need to write a JMX client to call such operations or need array-capable JMX monitoring tool.

Why do I see "… messages dropped …" in the logs?

This is a symptom of load shedding — Cassandra defending itself against more requests than it can handle.

Internode messages which are received by a node, but do not get not to be processed within their proper timeout (see `read_request_timeout`, `write_request_timeout`, … in the `cassandra-yaml`), are dropped rather than processed (since the as the coordinator node will no longer be waiting for a response).

For writes, this means that the mutation was not applied to all replicas it was sent to. The inconsistency will be repaired by read repair, hints or a manual repair. The write operation may also have timeouted as a result.

For reads, this means a read request may not have completed.

Load shedding is part of the Cassandra architecture, if this is a persistent issue it is generally a sign of an overloaded node or cluster.

Cassandra dies with `java.lang.OutOfMemoryError: Map failed`

If Cassandra is dying **specifically** with the "Map failed" message, it means the OS is denying java the ability to lock more memory. In linux, this typically means memlock is limited. Check `/proc/<pid of cassandra>/limits` to verify this and raise it (eg, via ulimit in bash). You may also need to increase `vm.max_map_count.` Note that the debian package handles this for you automatically.

What happens if two updates are made with the same timestamp?

Updates must be commutative, since they may arrive in different orders on different replicas. As long as Cassandra has a deterministic way to pick the winner (in a timestamp tie), the one selected is as valid as any other, and the specifics should be treated as an implementation detail. That said, in the case of a timestamp tie, Cassandra follows two rules: first, deletes take precedence over inserts/updates. Second, if there are two updates, the one with the lexically larger value is selected.

Why bootstrapping a new node fails with a "Stream failed" error?

Two main possibilities:

> the GC may be creating long pauses disrupting the streaming process
>
> compactions happening in the background hold streaming long enough that the TCP connection fails

In the first case, regular GC tuning advices apply. In the second case, you need to set TCP keepalive to a lower value (default is very high on Linux). Try to just run the following:

```
$ sudo /sbin/sysctl -w net.ipv4.tcp_keepalive_time=60 net.ipv4.tcp_keepalive_intvl=60
net.ipv4.tcp_keepalive_probes=5
```

To make those settings permanent, add them to your `/etc/sysctl.conf` file.

Note: GCE's firewall will always interrupt TCP connections that are inactive for more than 10 min. Running the above command is highly recommended in that environment.