neo4j (../../) Search Neo4j docs...

Table of Contents ⊞

5.5. Settings reference Enterprise Edition

This section lists the important settings related to running a Neo4j Causal Cluster.

Parameter	Explanation
`dbms.mode` (../../reference/configuration-settings/#config_dbms.mode)	This setting configures the operating mode of the database. For Causal Clustering, there are two possible modes: `CORE` or `READ_REPLICA`. **Example:** `dbms.mode=READ_REPLICA` will define this server as a Read Replica.
`causal_clustering.minimum_core_cluster_size_at_formation` (../../reference/configuration-settings/#config_causal_clustering.minimum_core_cluster_size_at_formation)	Minimum number of Core machines required to form a cluster. **Example:** `causal_clustering.minimum_core_cluster_size_at_formation=3` will specify that the cluster will form when at least three Core members have discovered each other.

Parameter	Explanation
`causal_clustering.minimum_core_cluster_size_at_runtime` (../../reference/configuration-settings/#config_causal_clustering.minimum_core_cluster_size_at_runtime)	The minimum size of the dynamically adjusted voting set (which only Core members may be a part of). Adjustments to the voting set happen automatically as the availability of Core members changes, due to explicit operations such as starting or stopping a member, or unintended issues such as network partitions. Please note that this dynamic scaling of the voting set is generally desirable, as under some circumstances it can increase the number of instance failures which may be tolerated. A majority of the voting set must be available before members are voted in or out. **Example:** `causal_clustering.minimum_core_cluster_size_at_runtime=3` will specify that the cluster should not try to dynamically adjust below three Core members in the voting set.

Parameter	Explanation
`causal_clustering.discovery_type (../../reference/configuration-settings/#config_causal_clustering.discovery_type)`	This setting specifies the strategy that the instance will use to determine the addresses for other instances in the cluster to contact for *bootstrapping*. Possible values are: `LIST`, `DNS`, `SRV`, and `K8S`. **`LIST`** Treat `causal_clustering.initial_discovery_members` as a list of addresses of Core Servers to contact for discovery. **`DNS`** Treat `causal_clustering.initial_discovery_members` as a domain name to resolve via DNS. Expect DNS resolution to provide A records with hostnames or IP addresses of Cores to contact for discovery, on the port specified by `causal_clustering.initial_discovery_members`. **`SRV`** Treat `causal_clustering.initial_discovery_members` as a domain name to resolve via DNS. Expect DNS resolution to provide SRV records with hostnames or IP addresses, and ports, of Cores to contact for discovery. **`K8S`** Access the Kubernetes list service API to derive addresses of Cores to contact for discovery. Requires `causal_clustering.kubernetes.label_selector` to be a Kubernetes label selector for Kubernetes services running a Core each and `causal_clustering.kubernetes.service_port_name` to be a service port name identifying the discovery port of Core services. The value of `causal_clustering.initial_discovery_members` is ignored for this option. The value of this setting determines how `causal_clustering.initial_discovery_members` is interpreted. Detailed information about discovery and discovery configuration options is given in Section 5.2.7.2, "Initial discovery of cluster members with DNS" (../setup-new-cluster/#causal-clustering-discovery-dns). **Example:** `causal_clustering.discovery_type=DNS` combined with `causal_clustering.initial_discovery_members=cluster01.example.com:5000` will fetch all DNS A records for *cluster01.example.com* and attempt to reach Neo4j instances listening on port 5000 for each A record's IP address.

Parameter	Explanation
`causal_clustering.initial_discovery_members` (../../reference/configuration-settings/#config_causal_clustering.initial_discovery_members)	The network addresses of an initial set of Core cluster members that are available to bootstrap this Core or Read Replica instance. In the default case, the initial discovery members are given as a comma-separated list of address/port pairs, and the default port for the discovery service is `:5000`. It is good practice to set this parameter to the same value on all Core Servers. It is good practice to set this parameter to the same value on all Core Servers. The behavior of this setting can be modified by configuring the setting `causal_clustering.discovery_type`. This is described in detail in Section 5.2.7.2, "Initial discovery of cluster members with DNS" (../setup-new-cluster/#causal-clustering-discovery-dns). **Example:** `causal_clustering.discovery_type=LIST` combined with `core01.example.com:5000,core02.example.com:5000,core03.example.com:5000` will attempt to reach Neo4j instances listening on *core01.example.com*, core01.example.com and core01.example.com; all on port 5000.
`causal_clustering.raft_advertised_address` (../../reference/configuration-settings/#config_causal_clustering.raft_advertised_address)	The address/port setting that specifies where the Neo4j instance advertises to other members of the cluster that it will listen for Raft messages within the Core cluster. **Example:** `causal_clustering.raft_advertised_address=192.168.33.20:7000` will listen for for cluster communication in the network interface bound to 192.168.33.20 on port 7000.
`causal_clustering.transaction_advertised_address` (../../reference/configuration-settings/#config_causal_clustering.transaction_advertised_address)	The address/port setting that specifies where the instance advertises where it will listen for requests for transactions in the transaction-shipping catchup protocol. **Example:** `causal_clustering.transaction_advertised_address=192.168.33.20:6001` will listen for transactions from cluster members on the network interface bound to 192.168.33.20 on port 6001.

Parameter	Explanation
`causal_clustering.discovery_listen_address` (../../reference/configuration-settings/#config_causal_clustering.discovery_listen_address)	The address/port setting for use by the discovery protocol. This is the setting which will be included in the setting `causal_clustering.initial_discovery_members` which are set in the configuration of the other members of the cluster. **Example:** `causal_clustering.discovery_listen_address=0.0.0.0:5001` will listen for cluster membership communication on any network interface at port 5001.
`causal_clustering.raft_listen_address` (../../reference/configuration-settings/#config_causal_clustering.raft_listen_address)	The address/port setting that specifies which network interface and port the Neo4j instance will bind to for cluster communication. This setting must be set in coordination with the address this instance advertises it will listen at in the setting `causal_clustering.raft_advertised_address`. **Example:** `causal_clustering.raft_listen_address=0.0.0.0:7000` will listen for cluster communication on any network interface at port 7000.
`causal_clustering.transaction_listen_address` (../../reference/configuration-settings/#config_causal_clustering.transaction_listen_address)	The address/port setting that specifies which network interface and port the Neo4j instance will bind to for cluster communication. This setting must be set in coordination with the address this instance advertises it will listen at in the setting `causal_clustering.transaction_advertised_address`. **Example:** `causal_clustering.transaction_listen_address=0.0.0.0:6001` will listen for cluster communication on any network interface at port 7000.
`causal_clustering.refuse_to_be_leader` (../../reference/configuration-settings/#config_causal_clustering.refuse_to_be_leader)	Prevents the current instance from volunteering to become Raft leader if set to `true`. Defaults to `false`, and should only be used in exceptional circumstances when advised by Neo4j Professional Services. **Example:** `causal_clustering.refuse_to_be_leader=false`
`causal_clustering.cluster_allow_reads_on_followers` (../../reference/configuration-settings/#config_causal_clustering.cluster_allow_reads_on_followers)	Defaults to `true` so that followers are available for read-only queries in a typical heterogeneous setup. Note: if there are no Read Replicas in the cluster, followers are made available for read, regardless the value of this setting. **Example:** `causal_clustering.cluster_allow_reads_on_followers=true`

Parameter	Explanation
`causal_clustering.store_copy_max_retry_time_per_request (../../reference/configuration-settings/#config_causal_clustering.store_copy_max_retry_time_per_request)`	Condition for when store copy should eventually fail. A request is allowed to retry for any amount of attempts as long as the configured time has not been met. For very large stores or other reason that might make transferring of files slow this could be increased. **Example:** `causal_clustering.store_copy_max_retry_time_per_request=60min`

5.5.1. Multi-data center settings

Parameter	Explanation
`causal_clustering.multi_dc_license (../../reference/configuration-settings/#config_causal_clustering.multi_dc_license)`	Enables multi-data center features. Requires appropriate licensing. **Example:** `causal_clustering.multi_dc_license=true` will enable the multi-data center features.
`causal_clustering.server_groups (../../reference/configuration-settings/#config_causal_clustering.server_groups)`	A list of group names for the server used when configuring load balancing and replication policies. **Example:** `causal_clustering.server_groups=us,us-east` will add the current instance to the groups `us` and `us-east`.
`causal_clustering.upstream_selection_strategy (../../reference/configuration-settings/#config_causal_clustering.upstream_selection_strategy)`	An ordered list in descending preference of the strategy which Read Replicas use to choose upstream database server from which to pull transactional updates. **Example:** `causal_clustering.upstream_selection_strategy=connect-randomly-within-server-group,typically-connect-to-random-read-replica` will configure the behavior so that the Read Replica will first try to connect to any other instance in the group(s) specified in `causal_clustering.server_groups`. Should we fail to find any live instances in those groups, then we will connect to a random Read Replica. A value of `user-defined` will enable custom strategy definitions using the setting `causal_clustering.user_defined_upstream_strategy`.

Parameter	Explanation
`causal_clustering.user_defined_upstream_strategy` (../../reference/configuration-settings/#config_causal_clustering.user_defined_upstream_strategy)	Defines the configuration of upstream dependencies. Can only be used if `causal_clustering.upstream_selection_strategy` is set to `user-defined`. **Example:** `causal_clustering.user_defined_upstream_strategy=groups(north2); groups(north); halt()` will look for servers in the `north2`. If none are available it will look in the `north` server group. Finally, if we cannot resolve any servers in any of the previous groups, then rule chain will be stopped via `halt()`.
`causal_clustering.load_balancing.plugin` (../../reference/configuration-settings/#config_causal_clustering.load_balancing.plugin)	The load balancing plugin to use. One pre-defined plugin named `server_policies` is available by default. **Example:** `causal_clustering.load_balancing.plugin=server_policies` will enable custom policy definitions.
`causal_clustering.load_balancing.config.server_policies.<policy-name>`	Defines a custom policy under the name `<policy-name>`. Note that load balancing policies are cluster-global configurations and should be defined the exact same way on all core machines. **Example:** `causal_clustering.load_balancing.config.server_policies.north1_only=groups(north1)→min(2); halt();` will define a load balancing policy named `north1_only`. Queries are only sent to servers in the `north1` server group, provided there are two of them available. If there are less than two servers in `north1` then the chain is halted.