Configuration: Main

Objectives

To understand the configuration of:

- The main server process.
- Network communication.

Configuration File

The main Aerospike configuration file is located at: `/etc/aerospike/aerospike.conf`. Items covered in this section are in BLUE.

- service (required)
- network (required)
- namespace (at least 1 required)

Each context will look like this:

```
service {
        ...
}
```

Server Process

This section covers the behavior of the high level database process.

Topics covered:

- Linux user/group running the process
- Location of the PID (Process ID)
- Transaction settings for storage

Linux User/Group

Description	Controls the Linux username/group that runs the Aerospike database.
Context location	service
Config parameters (defaults)	user (root) group (root)
Notes	If you set the username/group to a non-root user, you must make sure that the following are writable by the user/group you select: - the log file (`/var/log/aerospike/aerospike.log` by default) - the persistence file (if using RAM + disk for persistence) - any Flash/SSD devices you are using - the PID file
Change dynamically	No
Best practices	Most customers run the daemon as root. You must be careful if you are changing users on an already running database. The major issue is permissions to files/SSDs. Be sure to test thoroughly when doing so.

Transaction Settings for Storage

Description	Sets configuration for how queues and threads read from storage
Context location	service
Config parameters (defaults)	service-threads (4) transaction-queues (4) transaction-threads-per-queue (4)
Notes	Changes to the behavior vary greatly. We strongly recommend sticking to the settings in the “Best practices” section below.
Change dynamically	No
Best practices	The service-threads and transaction-queues should be set to the number of cores on the server. This includes hyper-threaded cores if the server is hyper-threaded. You should set transaction-threads-per-queue to “3”.

Server Process Example Config

For the server process here are examples of the configuration for a standard production environment for an SSD cluster.

```
service {
    user root
    group root
    pidfile /var/run/aerospike/asd.pid
    service-threads 24
    transaction-queues 24
    transaction-threads-per-queue 3
    ...
}
```

The Network

Networking is crucial to the function of any distributed system.

Topics covered:

- File descriptor limit (connection limit)
- The main database service
- Cluster formation (heartbeats)
- The fabric (inter-node communication)

Maximum Number of File Descriptors

Description	This is the maximum number of Linux file descriptors that the server will be able to set. This is not the just the number of open files, but also the maximum number of connections.
Context location	service
Config parameters (defaults)	proto-fd-max (15000) proto-fd-idle-ms (600000)
Note	There is also a maximum value that is set by the operating system. The Aerospike installer normally sets the OS maximum at 100,000. The proto-fd-max variable is limited by this number. The proto-fd-idle-ms sets the timeout for transactions
Change dynamically	Yes
Best practices	For production use, this should be set at 15,000. It may be set as low as 1,000 for development work. Sometimes when using certain client languages this, should be set at much higher such as 30,000 or even higher. The proto-fd-idle-ms should normally be used when you will be using a client with many short-lived connections, such as PHP. Then set this to 10,000. When not set with these languages, performance will suffer.

The maximum number of filehandles controls the maximum number of connections. In many cases the default is 1024, which is often very small for a database.

Aerospike recommends that proto-fd-max be set at 15,000 or higher, if using languages with many short lived connections such as PHP. Under these circumstances you should also set the proto-fd-idle-ms to 10000 to minimize the number of outstanding connections.

Main Database Service

Description	This is the configuration for the main database service. This is the port that applications will use to connect to this node.
Context location	network:service
Config parameters (defaults)	address access-address port
Notes	address: the is the IP address that the service will listen on. You may also specify "any" access-address: for servers with multiple IP addresses, this is the one it will share with the other nodes to use. This should match the address that the client applications will use. port: cannot be blank, standard value is 3000
Change dynamically	No
Best practices	Normally, you will want to set the following: address any access-address [IP address used by applications] port 3000 It is important that every node (even the first) point to some other node that will be in the cluster. This allows you to restart the first server as well.

The main database service is how your application will query the database.

The setting that people may need to set is for the access-address. While the service can listen to any IP address, the access-address is the one that the node will specify as its IP address for connections. This is normally done when the server has multiple IP addresses and you wish to only use 1 as the service.

Cluster Formation

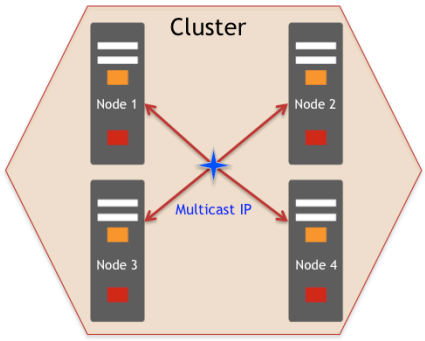
There are 2 different ways that a cluster can form. One is to use multicast connections, the other is to use mesh (or unicast).

The basic way this operates is that each node must send heartbeats that can be heard by other nodes. When enough of the heartbeats from one server have been missed by the others, it will be removed from the cluster.

You must choose one and only one mode for each cluster.

Cluster Formation
Heartbeat - Multicast

- When starting a multicast cluster, you start with isolated nodes (4 in this example).
- Each node will send a heartbeat to a multicast IP address, so all the nodes will know of each other.
- The cluster will form with the list of nodes. This map is also stored in each client, so they will know where to go for any given record. One of the nodes will create the partition map and will distribute it to the rest of the nodes in the cluster.



Automatic multicast gossip protocol for node discovery.
Paxos consensus algorithm determines nodes in cluster.
Ordered list of nodes determines data location.
Data partitions balanced for minimal data motion.
Vote initiated and terminated in 100 milliseconds.

Cluster Formation - Multicast

Description	This section controls how the cluster will be formed from individual nodes.
Context location	network:heartbeat
Config parameters (defaults)	mode multicast address port interval (150) timeout (10)
Notes	Mode must be multicast to use this mechanism. There is no default port, but 9918 is standard. interval is in milliseconds. timeout is the number of missed heartbeats, before the node is declared dead.
Change dynamically	interval –yes timeout – yes others - no
Best practices	For most production uses, use an interval of “150” and a timeout of “15”. For cloud environments, use “250” and “25”. However, note that most cloud environments like Amazon EC2 do not allow multicast. See following for note on multicast*

Regarding Multicast

Even in environments where multicast is possible, there is often some configuration work on the network devices, such as the switches.

If you find that multicast has worked for 3-5 minutes, but then stops, chances are you must do one of the following to switch with the vlan containing the nodes:

1. Turn off IGMP snooping

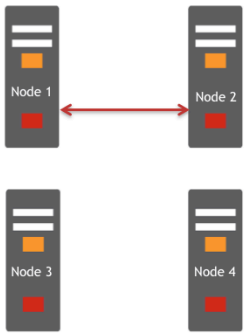
   OR

2. Turn on IGMP snooping, and also enable the querier (a.k.a multicast routing)

When checking for cluster stability make sure that you wait at least 5 minutes to see if the network will intrude.

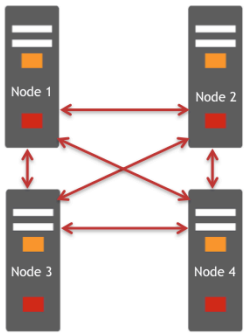
Cluster Formation
Heartbeat – Mesh (unicast)

- In the event that multicast is not possible, you can elect to use the mesh. This uses standard unicast. In this case you will need to bring up a single node first.
- As you bring up additional nodes, each one will be configured to communicate with a node that is already a part of the cluster (usually the first one) and share heartbeats with it.



Cluster Formation
Heartbeat – Mesh (unicast)

- In the event that multicast is not possible, you can elect to use the mesh. This uses standard unicast. In this case you will need to bring up a single node first.
- As you bring up additional nodes, each one will be configured to communicate with a node that is already a part of the cluster (usually the first one) and share heartbeats with it.



Cluster Formation – Mesh (Unicast)

Description	This section controls how the cluster will be formed from individual nodes.
Context location	network:heartbeat
Config parameters (defaults)	mode mesh address (any) port (3002) mesh-seed-address-port interval (150) timeout (10)
Notes	Mode must be mesh to use this mechanism The standard port is 3002, address is any this is the address on which the host will listen to heartbeat sent by the other nodes. mesh-seed-address-port is the IP address and port used by another node. You may specify multiple values with multiple lines. The interval and timeout are as in Multicast. Versions prior to 3.3.19 must use: mesh-address and mesh-port to specify a single other node.
Change dynamically	interval / timeout -yes others - no
Best practices	For most production uses, use an interval of “150” and a timeout of “10”. Note that most cloud environments like Amazon EC2 do not allow multicast.

You may specify multiple seed nodes in the event that one or more of the nodes is down.

Fabric

Description	The fabric controls intra-cluster communication between nodes.
Context location	network:fabric
Config parameters (defaults)	address port
Notes	The address should be the IP address that the fabric should respond on (you may also use “any”) The port is required and normally set to 3001
Change dynamically	No
Best practices	It is possible to configure the fabric to communicate on a different network device from the service.

Network Example Config (1 of 3)

For the connections variables, both configuration variables default to good values and can even be left unset in the file. You should only set them if:

- If your node is in a test environment and the node hardware is low-level, set `proto-fd-max` to 1000.
- If your clients have short lived connections (such as for PHP) you may want to apply the following:
  - `proto-fd-max 100000`
  - `proto-fd-idle-ms 10000`

```
service
    ...
    proto-fd-max 15000
    proto-fd-idle-ms 600000
    ...
}
```

Network Example Config (2 of 3)

If using multicast for heartbeats on IP address 239.1.99.222 and if you wish for your clients to access this node on the IP address 10.100.1.215, your config file may look like this:

```
network {
        service {
                address any
                port 3000
# If this server has multiple IP addresses, answer on this one (access-address)
                access-address 10.100.1.215
                reuse-address
        }

        heartbeat {
                mode multicast
# This address is the multicast IP address used by all the servers in the cluster
                address 239.1.99.222
                port 9918
                interval 150
                timeout 10
        }

        fabric {
                port 3001
        }

        info {
                port 3003
        }
}
```

Network Example Config (3 of 3)

If using mesh (unicast) for heartbeats. The IP address 10.100.1.215, your config file may look like this:

```
network {
        service {
                address any
                port 3000
# If this server has multiple IP addresses, answer on this one (access-address)
                access-address 10.100.1.215
                reuse-address
        }

        heartbeat {
                mode mesh
                port 3002
# The mesh-seed-address-port is the IP address/port of another node in the cluster
                mesh-seed-address-port 10.100.1.101 3002
                mesh-seed-address-port 10.100.1.102 3002
                mesh-seed-address-port 10.100.1.103 3002
                interval 150
                timeout 10
        }

        fabric {
                port 3001
        }

        info {
                port 3003
        }
}
```

Summary

What we have covered:

- The main server process.
- Network communication.