
AEROSPIKE
IN-MEMORY + NOSQL + ACID
AEROSPIKE DEPLOYMENT
CONFIGURE - MAIN
Aerospike aer . o . spike [air-oh- spahyk]
noun, 1. tip of a rocket that enhances speed and stability

## Objectives

To understand the configuration of:

- The main server process
- Network communication

## Configuration File

There are 7 major contexts in an Aerospike configuration file. Items covered in this section are in BLUE.

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

Each context will look like this:

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

## Aerospike Configuration File Notes

The main Aerospike configuration file contains all the configuration variables for a node.

- Located at `/etc/aerospike/aerospike.conf` on each node.
- NOT centrally managed by Aerospike.
- Most variables can be changed dynamically while the Aerospike node is up.
- If you wish for changes to the file to be persistent, you must edit the configuration file manually.
- You may choose to use shorthand (K, M, G) to represent large numbers. For example 4 gigabytes can be represented as 4G, which is mathematically 4*1024*1024*1024.

## Special Note

These training slides move from topic to topic. While this generally corresponds to a location (context) in the configuration file, this is not always true.

Parameters that are most commonly problematic are denoted in RED. Pay special attention to these, since the ramifications of improperly setting these variables may take months to show up or be difficult to fix once set.

## Server Process

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

Topics covered:

- Linux user/group running the process
- Single replica limit
- 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)<br>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:<br>- the log file (`/var/log/aerospike/aerospike.log` by default)<br>- the persistence file (if using RAM + disk for persistence)<br>- any Flash/SSD devices you are using<br>- the PID file |
| Change dynamically | No |
| Best practices | Most customers run the daemon as root.<br><br>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. |

## Single Replica Limit

| | |
|---|---|
| Description | Sets the limit at which the cluster will no longer maintain a replica of the data. This is done as a safety measure so administrators may choose between |
| Context location | service |
| Config parameters (defaults) | paxos-single-replica-limit (1) |
| Notes | If the cluster size is less than or equal to this value, keep only a single copy of all data in the cluster. |
| Change dynamically | No |
| Best practices | There is no single best practice. This depends on what the administrator believes is the best choice. If you believe that evicting data and poorer performance is acceptable, set this at a level consistent with what you believe is a worst (but possible) case of node loss. If you would prefer to maintain performance, but are willing to live with possible loss of data, keep this at 1. |

## Transaction Settings for Storage

| | |
|---|---|
| Description | Sets configuration for how queues and threads read from storage |
| Context location | service |
| Config parameters (defaults) | transaction-queues (4)<br>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 | You should set both to “4” if using only DRAM or DRAM + persistence namespaces. Set both to “8” if using any Flash/SSD namespaces. |

## 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
    paxos-single-replica-limit 1
    pidfile /var/run/aerospike/asd.pid
    transaction-queues 8
    transaction-threads-per-queue 8
    ...
}
```

## 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)<br>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.<br>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.<br>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. |

# 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<br>access-address<br>port<br>reuse-address |
| Notes | address: the is the IP address that the service will listen on. You may also specify “any”<br>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.<br>port: cannot be blank, standard value is 3000<br>reuse-address: sets whether or not to reuse the addresses when the service comes back up. No value is required, but can be true or false. |
| Change dynamically | No |
| Best practices | Normally, you will want to set the following:<br>address any<br>access-address [IP address used by applications]<br>port 3000<br>reuse-address true<br>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. |

## 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).

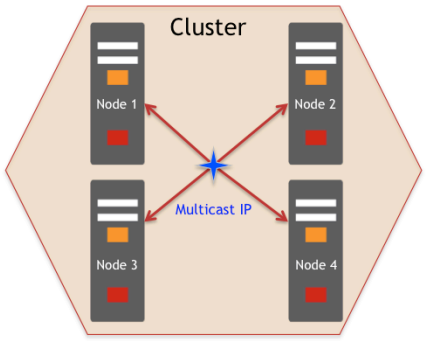
The basic way this operates is that each node must send heartbeats that can be heard by other nodes. When enough of the heartbeats from one server have been missed by the others, it will be removed from the cluster.

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

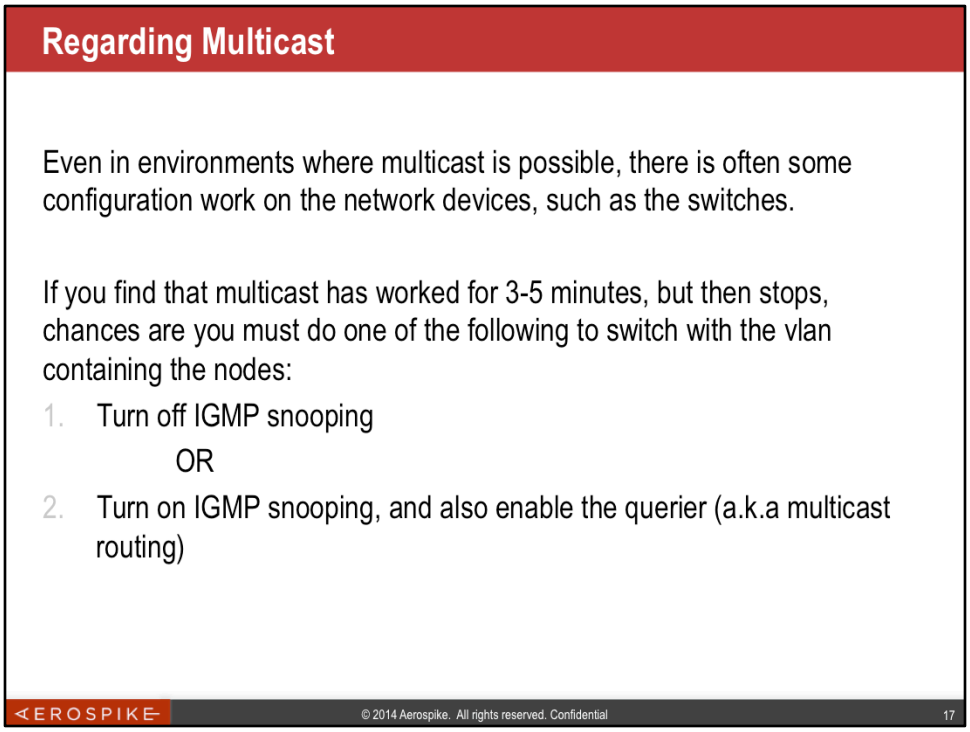
Ordered list of nodes determines data location

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<br>address<br>port<br>interval (150)<br>timeout (10) |
| Notes | Mode must be multicast to use this mechanism.<br>There is no default port, but is 9918 is standard.<br>interval is in milliseconds.<br>timeout is the number of missed heartbeats, before the node is declared dead. |
| Change dynamically | interval –yes<br>timeout – yes<br>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.<br>See following for note on multicast* |

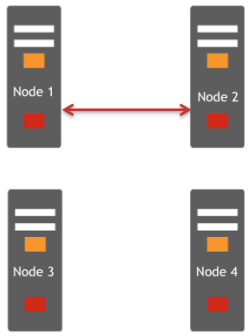



When checking for cluster stability make sure that you wait at least 5 minutes to see if the network will intrude.

## Cluster Formation

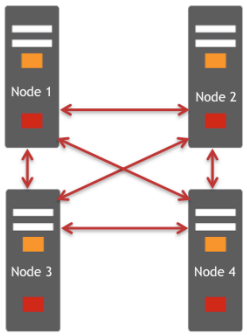
### Heartbeat – Mesh (unicast)

- 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<br>port<br>mesh-address<br>mesh-port<br>interval (150)<br>timeout (10) |
| Notes | Mode must be mesh to use this mechanism<br>The standard port is 3002, this is the address used by this node<br>mesh-address and mesh-port are the IP address and port used by the next node.<br>interval and timeout are as in Multicast. |
| Change dynamically | interval –yes<br>timeout – yes<br>others - no |
| Best practices | Aerospike has found that this mechanism works in production with up to 20 nodes.<br>For most production uses, use an interval of “150” and a timeout of “15”. For cloud environments, use “250” and “25”.<br>Note that most cloud environments like Amazon EC2 do not allow multicast. |

# Fabric

| | |
|---|---|
| Description | The fabric controls intra-cluster communication between nodes. |
| Context location | network:fabric |
| Config parameters (defaults) | address<br>port |
| Notes | The address should be the IP address that the fabric should respond on (you may also use “any”)<br>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 address is the IP address of another node in the cluster
                mesh-address 10.100.1.214
                mesh-port 3002
                interval 150
                timeout 10
        }

        fabric {
                port 3001
        }

        info {
                port 3003
        }
}
```

## Summary

What we have covered:

- The main server process
- Network communication