

ArangoDB v1.0 **CHEAT SHEET**

Starting & accessing

<code>./arangod /path/to/my/db</code>	start server
<code>./arangod --console --log error /path/to/my/db</code>	start emergency console (do not use with a db which has a server attached to it!)
<code>http://localhost:8529/_admin</code>	access admin front end in browser
<code>./arangosh</code>	start ArangoDB shell

arangod frequently used options

<code>--log</code>	set log level: fatal, error, warning, info, debug, trace
<code>--server.http-port host:port</code>	set address and port for HTTP-Clients
<code>--daemon</code>	run as daemon/background process

Database methods in arangosh

<code>db.create(collection-name, properties)</code>	create collection (with properties)
<code>db.collection(collection-name collection-id)</code>	get collection
<code>db.collections()</code>	list all collections
<code>db.collection-name</code>	get a collection by name
<code>db.drop(collection-name collection-id)</code>	drop collection with indexes
<code>db.truncate(collection-name collection-id)</code>	remove collection, keep indexes

Collection methods in arangosh

<code>collection.drop()</code>	drop collection with indexes
<code>collection.truncate()</code>	remove documents, keep indexes
<code>collection.properties()</code>	get all document properties
<code>collection.properties(properties)</code>	change property
<code>collection.figures()</code>	get all collection figures
<code>collection.load()</code>	load collection into memory
<code>collection.unload()</code>	start to unload a collection
<code>collection.rename(new-name)</code>	rename collection to new-name

Document methods in arangosh

<code>collection.document(document)</code>	get document by identifier
<code>collection.save(data)</code>	create new document
<code>collection.replace(document, data)</code>	replace existing document
<code>collection.remove(document)</code>	remove document
<code>db.document(document document-handle)</code>	get document by identifier handle
<code>db.replace(document document-</code>	replace existing document

`handle,data)`

`db.remove(document)` remove document

Edges in arangosh

<code>edge-collection.save(from, to, document)</code>	save new edge
<code>edge-collection.edges(vertex)</code>	find edges from (outbound) to (inbound) vertex
<code>edge-collection.inEdges(vertices)</code>	find all edges ending in (inbound)
<code>edge-collection.outEdges(vertices)</code>	find all edges starting from (outbound)

Queries in arangosh

<code>collection.all()</code>	select all documents and return cursor
<code>collection.byExample(example)</code>	select all documents that matches the given <i>example</i>
<code>collection.firstExample(example)</code>	select the first document that matches the given <i>example</i>
<code>collection.range(attribute, left, right)</code>	select all documents with attribute \geq <i>left</i> and $<$ <i>right</i>
<code>collection.count()</code>	returns the number of living documents in the collection
<code>collection.toArray()</code>	convert the collection into an array of documents (not for production!)

Geo Queries in arangosh

<code>collection.near(latitude, longitude)</code>	get documents near the given coordinates
<code>collection.within(latitude, longitude, distance)</code>	get all documents within a radius of <i>distance</i> around the given coordinates
<code>collection.geo(location)</code>	the next near or within operator will use the specific geo-spatial index

Pagination in arangosh

<code>query.limit(number)</code>	limits a result to the first number documents
<code>query.skip(number)</code>	skips the first number documents

Sequential Access And Cursors in arangosh

<code>query.hasNext()</code>	returns true if the cursor still has documents
<code>query.next()</code>	advance cursor
<code>query.dispose()</code>	free resources associated with a cursor
<code>query.count()</code>	returns number of documents in the result set