Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

Welcome to the Couchbase Shell `cbsh` documentation! If you are new, you will want to start with the quickstart section and then proceed with the introduction. If you are already familiar with the shell, feel free to jump right into the couchbase commands.

Note that while the project is maintained by Couchbase, it is not covered under the EE support contract. We are providing community support through this bug tracker (https://github.com/couchbaselabs/couchbase-shell/issues).

1. Quickstart

1.1. Installation

The current latest version is **1.0.0**.

There are a couple ways you can get access to `cbsh` .

1.1.1. Binaries

Download our pre-built binaries for your platform of choice:

- Linux: cbsh-x86_64-unknown-linux-gnu.tar.gz
  (https://github.com/couchbaselabs/couchbase-shell/releases/download/v1.0.0/cbsh-x86_64-unknown-linux-gnu.tar.gz)

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Windows: cbsh-x86_64-pc-windows-msvc.zip
  (https://github.com/couchbaselabs/couchbase-shell/releases/download/v1.0.0/cbsh-x86_64-pc-windows-msvc.zip)

Once you've downloaded the `zip` file, extract it and switch into the just created directory. The following example shows it for mac, but it works very similar if you are on linux (just align the commands with the file you just downloaded):

```
$ unzip cbsh-x86_64-apple-darwin.zip
$ ls
cbsh LICENSE LICENSE_AGREEMENT  README.md
```

You can now run the `cbsh` binary:

```
❯ ./cbsh --version
The Couchbase Shell 1.0.0
```

> **TIP** If you are running a recent macOS release (i.e. 10.15.x), you'll likely see an error similar to **"cbsh" was blocked from use because it is not from an identified developer**. This is because our binaries are not yet signed. To run it nonetheless you need to either navigate to `System Preferences → Security & Privacy` and click `Allow Anyway`, or run `sudo xattr -r -d com.apple.quarantine $PWD/cbsh` inside your terminal. Next time you run the binary you'll get another prompt but then it should run fine.

1.1.2. Homebrew

If running on macOS you can install via the Homebrew (https://formulae.brew.sh/formula/couchbase-shell) formula:

```
$ brew install couchbase-shell
```

Then run as follows:

```
$ cbsh --version
1.0.0
```

1.2. Connecting to a Cluster

The first time that you run `./cbsh` you will receive a prompt asking if you'd like to create a config file. If you choose yes then the shell will provide you with a series of prompts to provide information about your default cluster. If you choose no then it will try to connect to `localhost` using the `Administrator` username and the `password` password.

You can modify this through CLI arguments (see `./cbsh -h` for more information).

Note: Unless you specify TLS settings then PLAIN authentication is used and your credentials are sent in plaintext.

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
le-tls
```

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
ult, credentials will sent
```

```
👤 Administrator 🏠 default
>
```

Once in the shell, you can start to execute commands (see the introduction section for more information). As a quick sanity check, list the nodes in the cluster:

```
> nodes
┌───┬─────────┬──────────────┬─────────┬──────────────────────┬──────────────────────┬──────
| # |  cluster  |    hostname    | status  |         services         |        version        |
├───┼─────────┼──────────────┼─────────┼──────────────────────┼──────────────────────┼──────
| 0 | dev.local | 127.0.0.1:8091 | healthy | search,indexing,kv,query | 8.0.0-1246-enterprise | x86_64
└───┴─────────┴──────────────┴─────────┴──────────────────────┴──────────────────────┴──────
```

To start experimenting with data operations load some sample data onto the cluster:

```
👤 Administrator 🏠 default
> buckets load-sample travel-sample
┌───┬─────────┬─────────────────┬─────────┐
| # | cluster |    sample     | status  |
├───┼─────────┼─────────────────┼─────────┤
| 0 | local   | travel-sample | success |
└───┴─────────┴─────────────────┴─────────┘
```

Now you can try running N1QL queries using the query command.

```
👤 Administrator 🏠 default
> query "SELECT * FROM `travel-sample`.inventory.airline LIMIT 1"
┌───┬──────────────────────────────┬─────────┐
| # |           airline            | cluster |
├───┼──────────────────────────────┼─────────┤
| 0 | ┌─────────┬──────────────┐ | local   |
|   | | id      | 10           | |         |
|   | | type    | airline      | |         |
|   | | name    | 40-Mile Air  | |         |
|   | | iata    | Q5           | |         |
|   | | icao    | MLA          | |         |
|   | | callsign | MILE-AIR     | |         |
|   | | country | United States | |         |
|   | └─────────┴──────────────┘ |         |
└───┴──────────────────────────────┴─────────┘
```

Or you can get documents by switching to the travel sample bucket with cb-env and using doc get (https://couchbase.sh/docs/#_doc_get):

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
                                                                                                  ┌─────┬─────────┐
                                                                                                  │rror │ cluster │
                                                                                                  ├─────┼─────────┤
| 0 | airline_10 | ┌─────────┬──────────────────┐ | 1722587332136927232 |     | local   |
|   |            | | id      | 10               | |                     |     |         |
|   |            | | type    | airline          | |                     |     |         |
|   |            | | name    | 40-Mile Air      | |                     |     |         |
|   |            | | iata    | Q5               | |                     |     |         |
|   |            | | icao    | MLA              | |                     |     |         |
|   |            | | callsign| MILE-AIR         | |                     |     |         |
|   |            | | country | United States    | |                     |     |         |
|   |            | └─────────┴──────────────────┘ |                     |     |         |
└───┴────────────┴──────────────────────────────┴─────────────────────┴─────┴─────────┘
```

1.3. The config dotfiles

Connecting to a single cluster through the command line is nice when you are starting out, but later on you will likely either connect to the same cluster all the time or even to a multitude of them. To help with this, you can create a config file to hold your cluster details that the shell will read on startup.

The config file must be called `config` and be placed in a `.cbsh` dot file either in your home directory or in the directory from which the shell is being run. If you want to change the path of the directory where the config file is held this can be done with the `config-dir` flag when the shell is run:

```
> ls ~/config_file
config
> ./cbsh --config-dir ~/config_file
```

Note that even when the path to the directory containing the config file is given using this flag, the file containing the cluster information must still be called `config` .

The downloaded zip contains an example already, but here is a small sample config to help you get started as well:

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
password = "password"

[[cluster]]
identifier = "remote"
connstr = "10.143.200.101"
default-bucket = "myapp"
username = "user"
password = "pass"
capella-organization = "my-org"

[[capella-organization]]
identifier = "my-org"
access-key = "get-your-own"
secret-key = "get-your-own"
default-project = "default"
```

This will register two clusters, one called `local` and one called `remote`. Now when you start the shell, it will connect to `local` automatically and you are all set. Changing between registered clusters is done using cb-env cluster.

The config also registers a Capella organization which is associated with the cluster "remote" using the "capella-organization" field, see cb-env organizations for more details. A config file can be created with just a capella-organization then the shell can be used to create and register a cluster for use.

Please check out the reference section on additional parameters you can set as well as how to move the credentials to a separate `credentials` file in case you want to share your config with other people and they do not use the same credentials.

2. Introduction

Couchbase Shell is fully featured, so it does not only contain commands related to Couchbase but is actually built on top of a general purpose shell called Nushell (https://www.nushell.sh/). This allows you to interact with the file system or any other command available on your machine, making it a great tool for both operational and development tasks on top of Couchbase.

The following introduction only touches on the basic concepts to make you productive quickly. We recommend also checking out the great Nushell documentation (https://www.nushell.sh/book) so you can get the most out of it.

2.1. Navigating the Shell

Commands take inputs and produce output in a structured manner, most often represented as tables. Note how both the generic `ls` command and the Couchbase-specific `buckets` command both produce a table as their output:

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
|  3 | Dockerfile          | file |     590 B | 2 months ago  |
|  4 | LICENSE             | file |  11.1 KiB | 2 months ago  |
|  5 | LICENSE_AGREEMENT   | file |  43.7 KiB | 2 months ago  |
|  6 | NOTICES             | file | 221.6 KiB | a week ago    |
|  7 | README.md           | file |   9.4 KiB | 2 months ago  |
|  8 | build               | dir  |      64 B | 4 weeks ago   |
|  9 | docs                | dir  |     608 B | 3 minutes ago |
| 10 | examples            | dir  |     192 B | 4 months ago  |
| 11 | jupyter             | dir  |     128 B | 4 months ago  |
| 12 | rust-toolchain.toml | file |     524 B | 2 weeks ago   |
| 13 | src                 | dir  |     416 B | a day ago     |
| 14 | target              | dir  |     352 B | 2 weeks ago   |
| 15 | tests               | dir  |     160 B | 4 months ago  |
```

```
> buckets
```

#	cluster	name	type	replicas	min_durability_level	ram_quota	flush_enable
0	default	beer-sample	couchbase	1	none	200.0 MiB	false
1	default	default	couchbase	2	none	100.0 MiB	true
2	default	travel-sample	couchbase	2	none	200.0 MiB	false

You can pipe the output into other commands, for example if you only want to see buckets that have `sample` in their name you can utilize the `where` command:

```
> buckets | where name =~ "sample"
```

#	cluster	name	type	replicas	min_durability_level	ram_quota	flush_enable
0	default	beer-sample	couchbase	1	none	200.0 MiB	false
1	default	travel-sample	couchbase	2	none	200.0 MiB	false

In a similar fashion you can turn this structured table into other output formats, for example JSON:

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)


```
    "min_durability_level": "none",
    "ram_quota": 209715200,
    "flush_enabled": false,
    "status": "",
    "cloud": false
  },
  {
    "cluster": "default",
    "name": "travel-sample",
    "type": "couchbase",
    "replicas": 2,
    "min_durability_level": "none",
    "ram_quota": 209715200,
    "flush_enabled": false,
    "status": "",
    "cloud": false
  }
]
```


Exactly this type of composition takes the unix philosophy of "do one thing well" and meshes it together with the idea of flexible structured pipelines. This allows to build powerful compositions that help you in your daily operations with Couchbase, both from a developer or operations point of view.

2.2. Getting Help

Other than using this documentation for help, each command can be called with `-h` or `--help` to get information about potential flags, arguments and subcommands. Also, some commands provide additional examples.

```
> buckets -h
Perform bucket management operations

Usage:
  > buckets {flags}

Subcommands:
  buckets config - Shows the bucket config (low level)
  buckets create - Creates a bucket
  buckets drop - Drops buckets through the HTTP API
  buckets flush - Flushes buckets through the HTTP API
  buckets get - Fetches buckets through the HTTP API
  buckets load-sample - Load a sample bucket
  buckets update - Updates a bucket

Flags:
  -h, --help - Display the help message for this command
  --clusters <String> - the clusters which should be contacted
```

Some commands (like the one above) only act as groupings for subcommands, like `from`, `to` or `doc`. Since they do not serve a purpose on their own, they will render their help output automatically:

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
    doc insert - Insert a document through the data service
    doc remove - Removes a document through the data service
    doc replace - Replace a document through the data service
    doc upsert - Upsert (insert or override) a document through the data service

  Flags:
    -h, --help - Display the help message for this command
```

2.3. The Prompt explained

Couchbase Shell uses a custom, two line prompt to show you exactly in what environment you are working in right now. Since you can connect to different clusters, switch buckets etc. it is important to know what is currently "active", as the active buckets/scope/collection will be what commands are run against. Here is a sample prompt that will greet you when starting the shell:

```
👤 Administrator 🏠 local in 🗄 travel-sample._default._default
>
```

It tells you that your user is `Administrator`, the current active cluster identifier is `local` and the active bucket is `travel-sample`. The current scope and collection have not been set, so they assume the `_default` value.

If you have an active scope or collection set then the prompt will also update to reflect that:

```
👤 Administrator 🏠 local in 🗄 travel-sample.inventory.landmark
>
```

If we ran a `doc get` it would fetch the doc from travel-sample.inventory.landmark. In the second line, your actual user prompt starts.

2.4. Loading Data into the Shell

If you want to import data into Couchbase, or just load it into the shell for further processing, there are different commands available to help you. Once data is loaded into the shell the simplest way to save it to the connected cluster is using the doc import command. Alternatively if you want more control over the format and key of the uploaded data you can upload it manually using one of the other Couchbase save commands: `doc upsert` and `doc insert`.

2.4.1. Manual import

The `open` command will look at file endings and try to decode it automatically. Imagine a file named `user.json` in your current directory.

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
> open user.json
╭────────┬─────────╮
│ name   │ Michael │
│ age    │ 32      │
│ height │ 180     │
╰────────┴─────────╯
```

As you can see, the `open` command automatically decoded the JSON document into the tabular format. If the filename would only be `user`, the import would look like this instead:

```
> open user
{
  "name": "Michael",
  "age": 32,
  "height": 180
}
```

If you are dealing with data that cannot be decoded automatically, you can use the various `from` subcommands to help with decoding. In our case we use `from json`:

```
> open user | from json
╭────────┬─────────╮
│ name   │ Michael │
│ age    │ 32      │
│ height │ 180     │
╰────────┴─────────╯
```

The `doc upsert` and `doc insert` commands require there to be only two fields/columns. By default, these two columns are named `id` and `content`, but these can be overridden with `--id-column` and `--content-column`. The id column will be used as the key for the document, while the content column will be the json stored using that key. To get `user.json` in a format that can be inserted into the connected Cluster we wrap (https://www.nushell.sh/commands/docs/wrap.html) the entire document into a content column and then insert (https://www.nushell.sh/commands/docs/insert.html) the id that we want to use:

```
> open user.json | wrap content | insert id $in.content.name
╭─────────┬──────────────────────╮
│         │ ╭────────┬─────────╮ │
│ content │ │ name   │ Michael │ │
│         │ │ age    │ 32      │ │
│         │ │ height │ 180     │ │
│         │ ╰────────┴─────────╯ │
│ id      │ Michael              │
╰─────────┴──────────────────────╯
```

There are many other approaches to achieving this same result. With our data in the correct format we can then upsert:

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

See the Importing data recipes for more examples.

2.5. Exporting Data from the Shell

The export counterparts to `open` and `from`, are `save` and `to`. You can use both commands to take tabular data from the shell and store it in files of the needed target format.

Like `open`, `save` will try to discern the format from the file ending. The following example will load a JSON file and then save it as CSV:

```
> cat user.json
{
  "name": "Michael",
  "age": 32,
  "height": 180
}
```

```
> open user.json | save user.csv
```

```
> cat user.csv
name,age,height
Michael,32,180
```

This example is dealing with only one row for simplicity, but you can save as many rows as you need in one file.

As a motivating example, the following snippet runs a N1QL query and stores the result as a csv file:

```
> query "select airportname,city,country from `travel-sample` where type = 'airport' limit 10" |
save output.csv
```

```
> cat output.csv
airportname,city,country
Calais Dunkerque,Calais,France
Peronne St Quentin,Peronne,France
Les Loges,Nangis,France
Couterne,Bagnole-de-l'orne,France
Bray,Albert,France
Le Touquet Paris Plage,Le Tourquet,France
Denain,Valenciennes,France
Glisy,Amiens,France
La Garenne,Agen,France
Cazaux,Cazaux,France
```

See the Exporting data recipes for more information.

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

Couchbase

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

ıost) active. The same is true for nmands against (this can be

The "cb-env" command will tell you which resources are currently active (you are also able to tell from the prompt):

```
> cb-env
┌──────────────────────┬──────────────┐
| username             | charlie      |
| display_name         | Charlie      |
| cluster              | capella      |
| bucket               | default      |
| scope                | inventory    |
| collection           | hotel        |
| cluster_type         | provisioned  |
| capella-organization | couchbase    |
| capella-project      | Charlie work |
| llm                  | Gemini       |
└──────────────────────┴──────────────┘
```

If you were to now run a command then we would be running it:

- As the user "charlie"
- Against the "capella" cluster
- Against the "default" bucket
- Against the "inventory" scope
- Against the "hotel" collection

Note that `display_name` is the name that appears in your shell prompt and is not used by commands.

You can also change the active resources with the cb-env command.

```
    cb-env bucket - Sets the active bucket based on its name
    cb-env capella-organization - Sets the active capella organization based on its identifier
    cb-env cluster - Sets the active cluster based on its identifier
    cb-env collection - Sets the active collection based on its name
    cb-env managed - Lists all clusters currently managed by couchbase shell
    cb-env project - Sets the active project based on its name
    cb-env register - Registers a cluster for use with the shell
    cb-env scope - Sets the active scope based on its name
    cb-env timeouts - Sets the active timeouts for operations
    cb-env unregister - Unregisters a cluster for use with the shell

  Flags:
    -h, --help - Display the help message for this command
    --capella - show default execution environment of capella
    --timeouts - show default execution environment for timeouts
```

3.1. cb-env managed

Lists all the clusters you have registered with the shell.

```
> cb-env managed
╭───┬────────┬─────────┬────────────┬────────────────┬──────────────────────╮
| # | active |  tls    | identifier |   username     | capella_organization |
├───┼────────┼─────────┼────────────┼────────────────┼──────────────────────┤
| 0 | true   | false | dev.local  | Administrator |                      |
| 1 | false  | true  | capella    | charlie       |                      |
╰───┴────────┴─────────┴────────────┴────────────────┴──────────────────────╯
```

3.2. cb-env cluster

Changes the active cluster. The change of cluster will be reflected in the output of `cb-env` and the prompt:

```
👤 Charlie 🏠 capella
> cb-env cluster local
👤 Charlie 🏠 local
> cb-env
╭──────────────┬─────────────────╮
| username     | charlie       |
| display_name | Charlie       |
| cluster      | local         |
| bucket       |               |
| scope        |               |
| collection   |               |
| cluster_type | other         |
╰──────────────┴─────────────────╯
```

Notice that when you are connected to a locally running (non-Capella) cluster then the capella specific fields (project and capella organization) will not appear.

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

ot registered:

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

and.

3.3. cb-env bucket/scope/collection

The `cb-env bucket`, `cb-env scope` and `cb-env collection` commands are used to set the active bucket/scope/collection respectively. When changing the bucket/scope/collection the change will be reflected in the prompt and the output of the cb-env command:

```
👤 Charlie 🏠 local
> cb-env bucket travel-sample
👤 Charlie 🏠 local in 🗄 travel-sample._default._default
> cb-env scope inventory
👤 Charlie 🏠 local in 🗄 travel-sample.inventory._default
> cb-env collection landmark
👤 Charlie 🏠 local in 🗄 travel-sample.inventory.landmark
>
```

Note that when we set an active bucket the scope and collection are set to the `_default` scope and collection.

When a "higher level" entity is changed the "lower level" entities are returned to this default setting:

```
👤 Charlie 🏠 local in 🗄 travel-sample.inventory.landmarks
> cb-env scope some_scope
👤 Charlie 🏠 local in 🗄 travel-sample.some_scope._default
> cb-env bucket some_bucket
👤 Charlie 🏠 local in 🗄 some_bucket._default._default
>
```

Setting the scope resets the collection and setting the bucket will reset both the scope and collection. It is important to remember that these commands do not validate the existence of resources when the environment is changed. Setting the active bucket/scope/collection to one that doesn't exist will only have an impact when you try and perform operations against it.

```
👤 Charlie 🏠 local
> cb-env bucket not-a-real-bucket
👤 Charlie 🏠 local in 🗄 not-a-real-bucket._default._default
> doc get test_doc
┌───┬────┬─────────┬─────┬──────────────────────────────────────────────────────────────────────────────────────────────────────┐
│ # │ id │ content │ cas │                                                error                                                 │
├───┼────┼─────────┼─────┼──────────────────────────────────────────────────────────────────────────────────────────────────────┤
│ 0 │    │         │   0 │ Failed to load cluster config: bucket 'not-a-real-bucket' could not be four
└───┴────┴─────────┴─────┴──────────────────────────────────────────────────────────────────────────────────────────────────────┘
```

Checking that the active environment is correctly set to resources that exist is a good first port of call if things aren't working as expected.

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

rn contain clusters. Although you

s required to perform management

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

g the `cb-env capella-organization` command:

```
> cb-env capella-organization my-org
```

Or if you switch to an active cluster that has a default Capella organization specified on registration then that org will become active. The logic is similar with the active project which can either be changed manually:

```
👤 Administrator 🏠 local in 🗄 default._default._default
> cb-env project my-project
```

Alternatively if the active organization has a default project defined on registration then that default project will become active. Similarly to changing active clusters, the `cb-env capella-organization` command will return an error if the named org has not been registered with the shell:

```
> cb-env capella-organization not-an-org
Error:   × Organization not registered
  help: Has the organization not-an-org been registered in the config file?
```

But the same is not true with projects, so if operations are unexpectedly failing against a Capella cluster, make sure your active project is correctly set.

3.5. cb-env register

```
> cb-env register --help
Registers a cluster for use with the shell

Usage:
  > cb-env register {flags} <identifier> <connstr> <username> <password>

Flags:
  -h, --help - Display the help message for this command
  --display_name <String> - the display name to use for the user when this cluster is active
  --default-bucket <String> - the default bucket to use with this cluster
  --default-scope <String> - the default scope to use with this cluster
  --default-collection <String> - the default collection to use with this cluster
  --tls-enabled <String> - whether or not to enable tls, defaults to true
  --tls-cert-path <String> - the path to the certificate to use with tls
  --tls-accept-all-certs <String> - whether or not to accept all certs with tls, defaults to true
  --save - whether or not to add the cluster to the .cbsh config file, defaults to false
  --capella-organization <String> - capella organization that this cluster belongs to

Parameters:
  identifier <string>: the identifier to use for this cluster
  connstr <string>: the connection string to use for this cluster
  username <string>: the username to use for this cluster
  password <string>: the password to use for this cluster
```

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

entry to the config file and restart.
ered for use. `cb-env register`
al parameters:

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

ıtor password

Once registered the new cluster can be seen in the output of `cb-env managed`, and can be set as the active cluster using cb-env cluster:

```
👤 Charlie 🏠 remote in ☁ default._default._default
> cb-env managed
┌───┬────────┬───────┬────────────┬───────────────┬──────────────────────┐
| # | active |  tls  | identifier |   username    | capella_organization |
├───┼────────┼───────┼────────────┼───────────────┼──────────────────────┤
| 0 | false  | false | local      | Administrator |                      |
| 1 | false  | true  | new-cluster | Administrator |                      |
| 2 | true   | true  | remote     | charlie       | my-org               |
└───┴────────┴───────┴────────────┴───────────────┴──────────────────────┘
```

Note that by default although tls is enabled the shell will accept all certs. This can be changed with the `--tls-accept-all-certs` flag.

3.6. cb-env llm

```
> cb-env llm --help
Sets the active llm based on its identifier

Usage:
  > cb-env llm <identifier>

Flags:
  -h, --help - Display the help message for this command

Parameters:
  identifier <string>: the identifier of the llm
```

This command sets the active llm (large language model), which will be used by the vector enrich-doc, vector enrich-text and ask commands.

To be set as active the llm must be specified in the config file. For example:

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
[[llm]]
identifier = "Bedrock-titan"
provider = "Bedrock"
embed_model = "amazon.titan-embed-text-v1"
chat_model = "amazon.titan-text-express-v1"

[[llm]]
identifier = "Gemini-pro"
provider = "Gemini"
embed_model = "text-embedding-004"
chat_model = "gemini-1.0-pro"
api_key = "get-your-own"
```

The currently supported providers are Gemini (Google), Bedrock (AWS) and OpenAI. Specifying values other than these for the provider will result in an error when starting the shell. Notice that the Bedrock entry does not have an API key, this is because it requires the user to configure an appropriate role using the AWS CLI (https://docs.aws.amazon.com/cli/v1/userguide/cli-configure-role.html).

The `embed-model` field is the model that will be used to generate embeddings by the vector enrich-doc and vector enrich-text commands. While the `chat-model` is the model that will be used to answer questions with the ask command. These models can be any that the provider's API supports, and should be provided in the format given in the provider's API docs.

The api-keys can also be given separately in the credentials file, for example:

```
[[llm]]
identifier = "Gemini-pro"
api_key = "get-your-own"
```

The identifier must be the same as the entry in the config file for this to work.

The active llm can be checked using the cb-env command:

```
👤 Charlie 🏠 remote in ☁ default._default._default
> cb-env
┌──────────────────────┬────────────────┐
| username             | Administrator  |
| display_name         | Charlie        |
| cluster              | remote         |
| bucket               | default        |
| scope                | _default       |
| collection           | _default       |
| cluster_type         | provisioned    |
| capella-organization | my-org         |
| llm                  | Bedrock-titan  |
└──────────────────────┴────────────────┘
```

When the active llm is changed using `cb-env llm` this will be reflected in the output of `cb-env`:

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
| bucket               | default      |
| scope                | _default     |
| collection           | _default     |
| cluster_type         | provisioned  |
| capella-organization | my-org       |
| llm                  | Gemini-pro   |
```

3.7. Per command execution environments

On many commands you will notice a set of flags which allow you to override the active execution environment. Different commands support different flags, depending on the command you can expect to see any of:

- `--clusters`
- `--bucket`
- `--scope`
- `--collection`

3.7.1. The `--clusters` flag

The argument for this flag is an identifier combined with a regular expression. So imagine you have three clusters setup with the following names:

```
> cb-env managed | get identifier
╭───┬──────────────────╮
│ 0 │ prod-us-east     │
│ 1 │ prod-us-west     │
│ 2 │ prod-eu-center   │
╰───┴──────────────────╯
```

If you wanted to run a command against all clusters in `prod-us`, you could use `--clusters prod-us.*`, e.g.

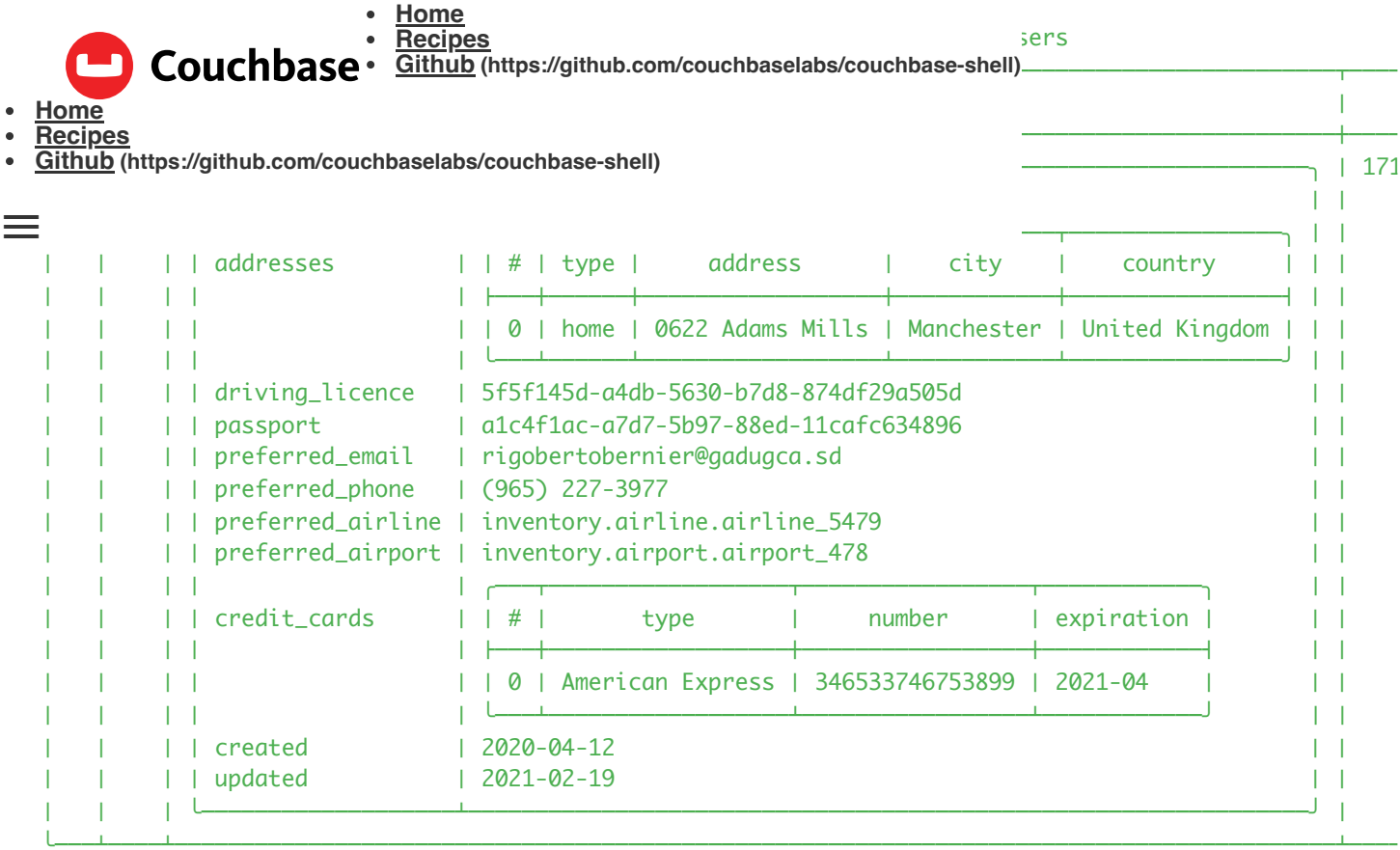
```
> buckets --clusters prod-us.*
╭───┬──────────────┬──────────────┬────────────┬──────────┬──────────────────────┬────────────┬─────────
│ # │   cluster    │     name     │    type    │ replicas │ min_durability_level │ ram_quota  │ flush_e
├───┼──────────────┼──────────────┼────────────┼──────────┼──────────────────────┼────────────┼─────────
│ 0 │ prod-us-east │ default      │ couchbase  │        1 │ none                 │ 200.0 MiB  │ false
│ 1 │ prod-us-west │ default      │ couchbase  │        1 │ none                 │ 200.0 MiB  │ false
│ 2 │ prod-us-west │ travel-sample│ couchbase  │        1 │ none                 │ 200.0 MiB  │ false
╰───┴──────────────┴──────────────┴────────────┴──────────┴──────────────────────┴────────────┴─────────
```

In the background this gets passed to a regex engine, so you can go a little crazy with it if needed.

3.7.2. The `--bucket`, `--scope`, `--collection` flags

These flags are a little different to the `--clusters` flag, they are not regular expressions and can only be used to define a single name each. Unlike `--clusters` the name provided to these flags does not have to be already known to Couchbase Shell, they can refer to any bucket, scope, and collection that exist within your active cluster or defined cluster(s). For example:

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)



4. Couchbase Commands

The following sections discuss the individual Couchbase specific commands in greater detail. Remember, you can always mix and match them with built-in other shell commands as well as executable programs from your environment.

4.1. buckets

The `buckets` commands are used to perform bucket management operations. They can be particularly powerful combined with the --clusters (https://couchbase.sh/docs/#_the_clusters_flag) flag.

4.1.1. `buckets`

Lists all the buckets from your active cluster:

```
Charlie dev.local
> buckets
```

#	cluster	name	type	replicas	min_durability_level	ram_quota	flush_enable
0	dev.local	beer-sample	couchbase	2	none	412.0 MiB	false
1	dev.local	default	couchbase	0	none	512.0 MiB	false
2	dev.local	memd	memcached	0	none	100.0 MiB	false

Check this recipe (https://couchbase.sh/docs/recipes/#_moving_data_between_clusters) to see how `cbsh` can be used to copy these buckets to another cluster.

4.1.2. `buckets config`

Retrieves the full config for the named bucket:

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
| bucketType             | membase
| collectionsManifestUid | 21
| compressionMode        | passive
| conflictResolutionType | seqno
| controllers            | {record 4 fields}
| ddocs                  | {record 1 field}
| durabilityMinLevel     | none
| evictionPolicy         | valueOnly
| localRandomKeyUri      | /pools/default/buckets/default/localRandomKey
| maxTTL                 | 0
| name                   | default
| nodeLocator            | vbucket
| nodes                  | [table 1 row]
| numVBuckets            | 64
| pitrEnabled            | false
| pitrGranularity        | 600
| pitrMaxHistoryAge      | 86400
| quota                  | {record 2 fields}
| replicaIndex           | false
| replicaNumber          | 0
| stats                  | {record 3 fields}
| storageBackend         | couchstore
| streamingUri           | /pools/default/bucketsStreaming/default?bucket_uuid=0ef162c33e14b163630f046
| threadsNumber          | 3
| uri                    | /pools/default/buckets/default?bucket_uuid=0ef162c33e14b163630f04639b347937
| uuid                   | 0ef162c33e14b163630f04639b347937
| vBucketServerMap       | {record 4 fields}
```

If you are unsure what you would use this for, you probably don't need it.

4.1.3. buckets create

Creates a bucket with the specified RAM quota on the active cluster:

```
👤 Charlie 🏠 local
> buckets create default 256
👤 Charlie 🏠 local
> buckets
┌───┬─────────┬─────────┬───────────┬──────────┬──────────────────────┬───────────┬───────────────┬────
| # | cluster |  name   |   type    | replicas | min_durability_level | ram_quota | flush_enabled | cl
├───┼─────────┼─────────┼───────────┼──────────┼──────────────────────┼───────────┼───────────────┼────
| 0 | local   | default | couchbase |        1 | none                 | 256.0 MiB | false         | fc
└───┴─────────┴─────────┴───────────┴──────────┴──────────────────────┴───────────┴───────────────┴────
```

Check this recipe (https://couchbase.sh/docs/recipes.html#_managing_multiple_clusters) to see how `cbsh` can help find a cluster for your bucket.

4.1.4. buckets drop

Drops the named bucket:

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
──────────┬──────────────────────┬──
am_quota | flush_enabled | cl
──────────┼──────────────────────┼──
56.0 MiB | false          | fa
──────────┴──────────────────────┴──
```

```
👤 Charlie 🏠 local
> buckets drop default
👤 Charlie 🏠 local
> buckets
╭─────────────╮
│ empty list │
╰─────────────╯
```

4.1.5. buckets flush

Deletes all docs from the named buckets:

```
👤 Charlie 🏠 local
> query "SELECT * FROM `travel-sample`" | length
31591
👤 Charlie 🏠 local
> buckets flush travel-sample
👤 Charlie 🏠 local
> query "SELECT * FROM `travel-sample`" | length
0
```

Not following what's going on, check the query (https://couchbase.sh/docs/#_query) and length (https://www.nushell.sh/commands/docs/length.html) documentation.

4.1.6. buckets get

Gets the named bucket:

```
> buckets get travel-sample
╭───┬─────────┬──────────────┬───────────┬──────────┬──────────────────────┬────────────┬──────────────
│ # │ cluster │     name     │   type    │ replicas │ min_durability_level │ ram_quota  │ flush_enable
├───┼─────────┼──────────────┼───────────┼──────────┼──────────────────────┼────────────┼──────────────
│ 0 │ local   │ travel-sample │ couchbase │        1 │ none                 │ 200.0 MiB  │ true
╰───┴─────────┴──────────────┴───────────┴──────────┴──────────────────────┴────────────┴──────────────
```

4.1.7. buckets load-sample

Loads the named sample bucket (https://docs.couchbase.com/server/current/manage/manage-settings/install-sample-buckets.html) onto the active cluster:

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
👤 Charlie 🏠 local
> buckets
┌───┬─────────┬──────────────┬───────────┬──────────┬──────────────────────┬────────────┬──────────────
| # | cluster |     name     |   type    | replicas | min_durability_level | ram_quota | flush_enable
├───┼─────────┼──────────────┼───────────┼──────────┼──────────────────────┼────────────┼──────────────
| 0 | local   | travel-sample | couchbase |        1 | none                 | 200.0 MiB | false
└───┴─────────┴──────────────┴───────────┴──────────┴──────────────────────┴────────────┴──────────────
```

4.1.8. buckets update

Updates the settings of an existing bucket:

```
👤 Charlie 🏠 local
> buckets
┌───┬─────────┬──────────────┬───────────┬──────────┬──────────────────────┬────────────┬──────────────
| # | cluster |     name     |   type    | replicas | min_durability_level | ram_quota | flush_enable
├───┼─────────┼──────────────┼───────────┼──────────┼──────────────────────┼────────────┼──────────────
| 0 | local   | travel-sample | couchbase |        1 | none                 | 200.0 MiB | false
└───┴─────────┴──────────────┴───────────┴──────────┴──────────────────────┴────────────┴──────────────
👤 Charlie 🏠 local
> buckets update travel-sample --flush true --replicas 2 --expiry 100
👤 Charlie 🏠 local
> buckets
┌───┬─────────┬──────────────┬───────────┬──────────┬──────────────────────┬────────────┬──────────────
| # | cluster |     name     |   type    | replicas | min_durability_level | ram_quota | flush_enable
├───┼─────────┼──────────────┼───────────┼──────────┼──────────────────────┼────────────┼──────────────
| 0 | local   | travel-sample | couchbase |        2 | none                 | 200.0 MiB | true
└───┴─────────┴──────────────┴───────────┴──────────┴──────────────────────┴────────────┴──────────────
```

4.2. scopes

The `scopes` commands are used to manage scopes.

4.2.1. scopes

Lists all of the scopes in the active bucket:

```
👤 Charlie 🏠 local in 🗄 travel-sample._default._default
> scopes
┌───┬──────────────────┬─────────┐
| # |      scope       | cluster |
├───┼──────────────────┼─────────┤
| 0 | inventory        | local   |
| 1 | tenant_agent_00  | local   |
| 2 | tenant_agent_01  | local   |
| 3 | tenant_agent_02  | local   |
| 4 | tenant_agent_03  | local   |
| 5 | tenant_agent_04  | local   |
| 6 | _default         | local   |
└───┴──────────────────┴─────────┘
```

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
| # |  scope   | cluster |
├───┼──────────┼─────────┤
| 0 | _default | local   |
└───┴──────────┴─────────┘
👤 Charlie 🏠 local in 🗄 default._default._default
> scopes create new-scope
👤 Charlie 🏠 local in 🗄 default._default._default
> scopes
┌───┬────────────┬─────────┐
| # |   scope    | cluster |
├───┼────────────┼─────────┤
| 0 | new-scope  | local   |
| 1 | _default   | local   |
└───┴────────────┴─────────┘
```

4.2.3. scopes drop

Deletes the scope matching the given name:

```
👤 Charlie 🏠 local in 🗄 default._default._default
> scopes
┌───┬────────────┬─────────┐
| # |   scope    | cluster |
├───┼────────────┼─────────┤
| 0 | new-scope  | local   |
| 1 | _default   | local   |
└───┴────────────┴─────────┘
👤 Charlie 🏠 local in 🗄 default._default._default
> scopes drop new-scope
👤 Charlie 🏠 local in 🗄 default._default._default
> scopes
┌───┬──────────┬─────────┐
| # |  scope   | cluster |
├───┼──────────┼─────────┤
| 0 | _default | local   |
└───┴──────────┴─────────┘
```

4.3. collections

The `collections` commands are used to manage collections.

4.3.1. collections

Lists all of the collections in the active scope:

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
| 2 | airport    | inherited  | local   |
| 3 | airline    | inherited  | local   |
| 4 | route      | inherited  | local   |
╰───┴────────────┴────────────┴─────────╯
```

4.3.2. collections create

Create a collection with the name supplied:

```
👤 Charlie 🏠 local in 🗄 my-bucket.my-scope._default
> collections
╭─────────────╮
│ empty list  │
╰─────────────╯
👤 Charlie 🏠 local in 🗄 my-bucket.my-scope._default
> collections create new-collection
👤 Charlie 🏠 local in 🗄 my-bucket.my-scope._default
> collections
╭───┬────────────────┬────────────┬─────────╮
│ # │   collection   │ max_expiry │ cluster │
├───┼────────────────┼────────────┼─────────┤
│ 0 │ new-collection │ inherited  │ local   │
╰───┴────────────────┴────────────┴─────────╯
```

4.3.3. collections drop

Drop the collection matching the name given:

```
👤 Charlie 🏠 local in 🗄 default.my-scope._default
> collections
╭───┬────────────────┬────────────┬─────────╮
│ # │   collection   │ max_expiry │ cluster │
├───┼────────────────┼────────────┼─────────┤
│ 0 │ new-collection │ inherited  │ local   │
╰───┴────────────────┴────────────┴─────────╯
👤 Charlie 🏠 local in 🗄 default.my-scope._default
> collections drop new-collection
👤 Charlie 🏠 local in 🗄 default.my-scope._default
> collections
╭─────────────╮
│ empty list  │
╰─────────────╯
```

4.4. doc

The `doc` commands are for managing the documents stored in the registered clusters.

4.4.1. doc get

Gets a doc from the active cluster, bucket, scope and collection:

- **Home**
- **Recipes**
- **Github** (https://github.com/couchbaselabs/couchbase-shell)


```
                                                                        ror | cluster |
                                                                            | local   |
                                                                            |         |
|    |            | | type     | airline       | |                    |       |         |
|    |            | | name     | 40-Mile Air   | |                    |       |         |
|    |            | | iata     | Q5            | |                    |       |         |
|    |            | | icao     | MLA           | |                    |       |         |
|    |            | | callsign | MILE-AIR      | |                    |       |         |
|    |            | | country  | United States | |                    |       |         |
|    |            | └──────────┴───────────────┘ |                    |       |         |
└────┴────────────┴──────────────────────────────┴────────────────────┴───────┴─────────┘
```

The contents of the document are returned in the `content` column alongside the document metadata (id, cas, error and cluster).

`doc get` can get multiple documents at once using an input stream:

```
👤 Charlie 🏠 local in 🗄 travel-sample._default._default
> [airline_10 airline_10748 airline_137] | wrap id | doc get
┌───┬──────────────┬────────────────────────────┬─────────────────────┬───────┬──────────────┐
| # |      id      |          content           |         cas         | error |   cluster    |
├───┼──────────────┼────────────────────────────┼─────────────────────┼───────┼──────────────┤
| 0 | airline_10   | ┌──────────┬───────────────┐ | 1712321628975398912 |       | prod-us-west |
|   |              | | id       | 10            | |                     |       |              |
|   |              | | type     | airline       | |                     |       |              |
|   |              | | name     | 40-Mile Air   | |                     |       |              |
|   |              | | iata     | Q5            | |                     |       |              |
|   |              | | icao     | MLA           | |                     |       |              |
|   |              | | callsign | MILE-AIR      | |                     |       |              |
|   |              | | country  | United States | |                     |       |              |
|   |              | └──────────┴───────────────┘ |                     |       |              |
| 1 | airline_137  | ┌──────────┬────────────┐    | 1712321633323712512 |       | prod-us-west |
|   |              | | id       | 137        |    |                     |       |              |
|   |              | | type     | airline    |    |                     |       |              |
|   |              | | name     | Air France |    |                     |       |              |
|   |              | | iata     | AF         |    |                     |       |              |
|   |              | | icao     | AFR        |    |                     |       |              |
|   |              | | callsign | AIRFRANS   |    |                     |       |              |
|   |              | | country  | France     |    |                     |       |              |
|   |              | └──────────┴────────────┘    |                     |       |              |
| 2 | airline_10748 | ┌──────────┬───────────────┐ | 1712321631323947008 |       | prod-us-west |
|   |              | | id       | 10748         | |                     |       |              |
|   |              | | type     | airline       | |                     |       |              |
|   |              | | name     | Locair        | |                     |       |              |
|   |              | | iata     | ZQ            | |                     |       |              |
|   |              | | icao     | LOC           | |                     |       |              |
|   |              | | callsign | LOCAIR        | |                     |       |              |
|   |              | | country  | United States | |                     |       |              |
|   |              | └──────────┴───────────────┘ |                     |       |              |
└───┴──────────────┴────────────────────────────┴─────────────────────┴───────┴──────────────┘
```

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

put column using the `--id-`

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

supported by the from (https://www.nushell.sh/commands/docs/from.html) command.

```
👤 Charlie 🏠 local in 🗄 default._default._default
> cat user.json
{
  "id": 123,
  "name": "Michael",
  "age": 32,
  "height": 180
}
👤 Charlie 🏠 local in 🗄 default._default._default
> doc import user.json
╭───┬───────────┬─────────┬────────┬──────────┬─────────╮
│ # │ processed │ success │ failed │ failures │ cluster │
├───┼───────────┼─────────┼────────┼──────────┼─────────┤
│ 0 │         1 │       1 │      0 │          │ local   │
╰───┴───────────┴─────────┴────────┴──────────┴─────────╯
👤 Charlie 🏠 local in 🗄 default._default._default
> doc get 123
╭───┬─────┬──────────────────────┬──────────────────┬───────┬─────────╮
│ # │ id  │       content        │       cas        │ error │ cluster │
├───┼─────┼──────────────────────┼──────────────────┼───────┼─────────┤
│ 0 │ 123 │ ╭────────┬─────────╮ │ 1726063078042501120 │       │ local   │
│   │     │ │ id     │ 123     │ │                  │       │         │
│   │     │ │ name   │ Michael │ │                  │       │         │
│   │     │ │ age    │ 32      │ │                  │       │         │
│   │     │ │ height │ 180     │ │                  │       │         │
│   │     │ ╰────────┴─────────╯ │                  │       │         │
╰───┴─────┴──────────────────────┴──────────────────┴───────┴─────────╯
```

`doc import` will use the `id` field in the source document as the document key by default, but this behaviour can be changed with the `--id-column` flag.

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
| # | processed | success | failed | failures | cluster |
├───┼───────────┼─────────┼────────┼──────────┼─────────┤
| 0 |         1 |       1 |      0 |          | local   |
└───┴───────────┴─────────┴────────┴──────────┴─────────┘
```

```
👤 Charlie 🏠 local in 🗄 default._default._default
> doc get Michael
```

```
┌───┬─────────┬──────────────────────┬──────────────────────┬───────┬─────────┐
| # |   id    |       content        |         cas          | error | cluster |
├───┼─────────┼──────────────────────┼──────────────────────┼───────┼─────────┤
| 0 | Michael | ┌────────┬─────────┐ | 1726063937243250688 |       | local   |
|   |         | | name   | Michael | |                      |       |         |
|   |         | | age    | 32      | |                      |       |         |
|   |         | | height | 180     | |                      |       |         |
|   |         | └────────┴─────────┘ |                      |       |         |
└───┴─────────┴──────────────────────┴──────────────────────┴───────┴─────────┘
```

> **TIP**
>
> look at the many different import formats `from` supports, including csv, xml, yaml and even sqlite. With this simple tool at hand you are able to load many different data formats quickly and import them into Couchbase!

4.4.3. `doc insert`

Inserts document into the active bucket/scope/collection:

```
👤 Charlie 🏠 remote in ☁ default._default._default
> open user.json
```

```
┌────────┬─────────┐
| name   | Michael |
| age    | 32      |
| height | 180     |
└────────┴─────────┘
```

```
👤 Charlie 🏠 remote in ☁ default._default._default
> open user.json | wrap content | insert id $in.content.name | doc insert
```

```
┌───┬───────────┬─────────┬────────┬──────────┬─────────┐
| # | processed | success | failed | failures | cluster |
├───┼───────────┼─────────┼────────┼──────────┼─────────┤
| 0 |         1 |       1 |      0 |          | remote  |
└───┴───────────┴─────────┴────────┴──────────┴─────────┘
```

And if a document already exists in the active collection with this key then the command will fail:

```
👤 Charlie 🏠 remote in ☁ default._default._default
> open user.json | wrap content | insert id $in.content.name | doc insert
```

```
┌───┬───────────┬─────────┬────────┬──────────────────┬─────────┐
| # | processed | success | failed |     failures     | cluster |
├───┼───────────┼─────────┼────────┼──────────────────┼─────────┤
| 0 |         1 |       0 |      1 | Key already exists | remote  |
└───┴───────────┴─────────┴────────┴──────────────────┴─────────┘
```

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

doc insert command.

les.

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

e the following stored:

```
👤 Charlie 🏠 remote in ☁ default._default._default
> doc get Michael
┌───┬─────────┬──────────────────────┬──────────────────┬───────┬─────────┐
│ # │   id    │       content        │       cas        │ error │ cluster │
├───┼─────────┼──────────────────────┼──────────────────┼───────┼─────────┤
│ 0 │ Michael │ ┌────────┬─────────┐ │ 1726821299020365824 │       │ remote  │
│   │         │ │ name   │ Michael │ │                  │       │         │
│   │         │ │ age    │ 32      │ │                  │       │         │
│   │         │ │ height │ 180     │ │                  │       │         │
│   │         │ └────────┴─────────┘ │                  │       │         │
└───┴─────────┴──────────────────────┴──────────────────┴───────┴─────────┘
```

We would remove it as follows:

```
👤 Charlie 🏠 remote in ☁ default._default._default
> doc remove Michael
┌───┬───────────┬─────────┬────────┬──────────┬─────────┐
│ # │ processed │ success │ failed │ failures │ cluster │
├───┼───────────┼─────────┼────────┼──────────┼─────────┤
│ 0 │         1 │       1 │      0 │          │ remote  │
└───┴───────────┴─────────┴────────┴──────────┴─────────┘
```

If a document matching the key cannot be found, then an error is returned:

```
👤 Charlie 🏠 remote in ☁ default._default._default
> doc remove Michael
┌───┬───────────┬─────────┬────────┬──────────────┬─────────┐
│ # │ processed │ success │ failed │   failures   │ cluster │
├───┼───────────┼─────────┼────────┼──────────────┼─────────┤
│ 0 │         1 │       0 │      1 │ Key not found │ remote  │
└───┴───────────┴─────────┴────────┴──────────────┴─────────┘
```

You can also remove multiple documents at once with an input stream:

```
👤 Charlie 🏠 remote in ☁ travel-sample._default._default
> [airline_10 airline_10748 airline_137] | wrap id | doc remove
┌───┬───────────┬─────────┬────────┬──────────┬─────────┐
│ # │ processed │ success │ failed │ failures │ cluster │
├───┼───────────┼─────────┼────────┼──────────┼─────────┤
│ 0 │         3 │       3 │      0 │          │ remote  │
└───┴───────────┴─────────┴────────┴──────────┴─────────┘
```

4.4.5. doc replace

Replaces the document in Couchbase matching the key id of the new one, if there is no document matching the `id` then an error is returned. Say we have the following document stored in the connected cluster:

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
|   |         | | age    | 32      | |                    |       |          |
|   |         | | height | 180     | |                    |       |          |
|   |         | └────────┴─────────┘ |                    |       |          |
└───┴─────────┴─────────────────────┴────────────────────┴───────┴──────────┘
```

And we have an updated version stored locally:

```
👤 Charlie 🏠 remote in ☁ default._default._default
> open user.json
┌────────┬─────────┐
| name   | Michael |
| age    | 80      |
| height | 110     |
└────────┴─────────┘
```

Then we can replace the first with the second using `doc replace`:

```
👤 Charlie 🏠 remote in ☁ default._default._default
> open user.json | wrap content | insert id $in.content.name | doc replace
┌───┬───────────┬─────────┬────────┬──────────┬─────────┐
| # | processed | success | failed | failures | cluster |
├───┼───────────┼─────────┼────────┼──────────┼─────────┤
| 0 |         1 |       1 |      0 |          | remote  |
└───┴───────────┴─────────┴────────┴──────────┴─────────┘
👤 Charlie 🏠 remote in ☁ default._default._default
> doc get Michael
┌───┬─────────┬─────────────────────┬──────────────────────┬───────┬─────────┐
| # |   id    |       content       |         cas          | error | cluster |
├───┼─────────┼─────────────────────┼──────────────────────┼───────┼─────────┤
| 0 | Michael | ┌────────┬────────┐ | 1726821910540648448  |       | remote  |
|   |         | | name   | Michael| |                      |       |         |
|   |         | | age    | 80     | |                      |       |         |
|   |         | | height | 110    | |                      |       |         |
|   |         | └────────┴────────┘ |                      |       |         |
└───┴─────────┴─────────────────────┴──────────────────────┴───────┴─────────┘
```

See the manual import section for an explanation of the formatting we do before piping to the `doc replace` command. If there was no document with the `id` Michael, then the replace would fail:

```
👤 Charlie 🏠 remote in ☁ default._default._default
> open user.json | wrap content | insert id $in.content.name | doc replace
┌───┬───────────┬─────────┬────────┬──────────────┬─────────┐
| # | processed | success | failed |   failures   | cluster |
├───┼───────────┼─────────┼────────┼──────────────┼─────────┤
| 0 |         1 |       0 |      1 | Key not found | remote  |
└───┴───────────┴─────────┴────────┴──────────────┴─────────┘
```

Similarly to `doc insert`, `doc replace` can be used to replace multiple documents at once, see importing data for examples.

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

e is an existing doc in the cluster behaves the same as doc insert:

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
╭───┬─────────┬─────────┬─────┬─────────────────┬─────────╮
│ # │   id    │ content │ cas │      error      │ cluster │
├───┼─────────┼─────────┼─────┼─────────────────┼─────────┤
│ 0 │ Michael │         │   0 │ Key not found   │ remote  │
╰───┴─────────┴─────────┴─────┴─────────────────┴─────────╯
👤 Charlie 🏠 remote in ☁ default._default._default
> open user.json
╭────────┬─────────╮
│ name   │ Michael │
│ age    │ 32      │
│ height │ 180     │
╰────────┴─────────╯
👤 Charlie 🏠 remote in ☁ default._default._default
> open user.json | wrap content | insert id $in.content.name | doc upsert
╭───┬───────────┬─────────┬────────┬──────────┬─────────╮
│ # │ processed │ success │ failed │ failures │ cluster │
├───┼───────────┼─────────┼────────┼──────────┼─────────┤
│ 0 │         1 │       1 │      0 │          │ remote  │
╰───┴───────────┴─────────┴────────┴──────────┴─────────╯
👤 Charlie 🏠 remote in ☁ default._default._default
> doc get Michael
╭───┬─────────┬─────────────────────┬──────────────────┬───────┬─────────╮
│ # │   id    │       content       │       cas        │ error │ cluster │
├───┼─────────┼─────────────────────┼──────────────────┼───────┼─────────┤
│ 0 │ Michael │ ╭────────┬────────╮ │ 1726822249315041280 │       │ remote  │
│   │         │ │ name   │ Michael│ │                  │       │         │
│   │         │ │ age    │ 32     │ │                  │       │         │
│   │         │ │ height │ 180    │ │                  │       │         │
│   │         │ ╰────────┴────────╯ │                  │       │         │
╰───┴─────────┴─────────────────────┴──────────────────┴───────┴─────────╯
```

See the manual import section for an explanation of the formatting we do before piping to the `doc upsert` command.

Or if there is an existing document with the same `id` then `upsert` will behave the same as a `replace` :

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
|   |         | | age    | 32      | |                    |       |          |
|   |         | | height | 180     | |                    |       |          |
|   |         | └────────┴─────────┘ |                    |       |          |
└───┴─────────┴─────────────────────┴────────────────────┴───────┴──────────┘
👤 Charlie 🏠 remote in ☁ default._default._default
> open user.json
┌────────┬─────────┐
| name   | Michael |
| age    | 80      |
| height | 110     |
└────────┴─────────┘
👤 Charlie 🏠 remote in ☁ default._default._default
> open user.json | wrap content | insert id $in.content.name | doc upsert
┌───┬────────────┬─────────┬────────┬──────────┬─────────┐
| # | processed | success | failed | failures | cluster |
├───┼────────────┼─────────┼────────┼──────────┼─────────┤
| 0 |          1 |       1 |      0 |          | remote  |
└───┴────────────┴─────────┴────────┴──────────┴─────────┘
👤 Charlie 🏠 remote in ☁ default._default._default
> doc get Michael
┌───┬─────────┬─────────────────────┬──────────────────┬───────┬─────────┐
| # |   id    |       content       |       cas        | error | cluster |
├───┼─────────┼─────────────────────┼──────────────────┼───────┼─────────┤
| 0 | Michael | ┌────────┬─────────┐ | 1726822368839598080 |       | remote  |
|   |         | | name   | Michael | |                  |       |         |
|   |         | | age    | 80      | |                  |       |         |
|   |         | | height | 110     | |                  |       |         |
|   |         | └────────┴─────────┘ |                  |       |         |
└───┴─────────┴─────────────────────┴──────────────────┴───────┴─────────┘
```

4.5. nodes

The `nodes` command allows you to list all the nodes of the cluster you are currently connected to.

```
> nodes
┌───┬──────────────┬──────────────────────┬─────────┬──────────────────────┬──────────────────────┐
| # |   cluster    |       hostname       | status  |       services       |       version        |
├───┼──────────────┼──────────────────────┼─────────┼──────────────────────┼──────────────────────┤
| 0 | prod-us-west | 192.168.107.128:8091 | healthy | search,indexing,kv,query | 7.6.2-3505-enterprise |
| 1 | prod-us-west | 192.168.107.129:8091 | healthy | search,indexing,kv,query | 7.6.2-3505-enterprise |
| 2 | prod-us-west | 192.168.107.130:8091 | healthy | search,indexing,kv,query | 7.6.2-3505-enterprise |
└───┴──────────────┴──────────────────────┴─────────┴──────────────────────┴──────────────────────┘
```

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

Couchbase

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)


```
:he data service
```

```
Flags:
  -h, --help - Display the help message for this command
  --id-column <String> - the name of the id column if used with an input stream
  --bucket <String> - the name of the bucket
  --scope <String> - the name of the scope
  --collection <String> - the name of the collection
  --clusters <String> - the clusters which should be contacted
  --batch-size <Number> - the maximum number of items to batch send at a time
  -e, --halt-on-error - halt on any errors

Parameters:
  path <any>: the path(s) to be fetched from the documents
  id <string>: the document id (optional)
```

It can be used to retrieve a field from a single document:

```
👤 Administrator 🏠 cluster in 🗄 travel-sample._default._default
> subdoc get address landmark_10019
╭───┬────────────────┬──────────────────────────┬──────────────────┬─────────┬─────────╮
│ # │       id       │         content          │       cas        │ error   │ cluster │
├───┼────────────────┼──────────────────────────┼──────────────────┼─────────┼─────────┤
│ 0 │ landmark_10019 │ Prince Arthur Road, ME4 4UG │ 1722410659053961216 │         │ local   │
╰───┴────────────────┴──────────────────────────┴──────────────────┴─────────┴─────────╯
```

Or similarly to the `doc` commands a stream of ids can be provided:

```
👤 Administrator 🏠 cluster in 🗄 travel-sample._default._default
> [landmark_10019 landmark_10020] | subdoc get address
╭───┬────────────────┬──────────────────────────┬──────────────────┬─────────┬─────────╮
│ # │       id       │         content          │       cas        │ error   │ cluster │
├───┼────────────────┼──────────────────────────┼──────────────────┼─────────┼─────────┤
│ 0 │ landmark_10019 │ Prince Arthur Road, ME4 4UG │ 1722410659053961216 │         │ local   │
│ 1 │ landmark_10020 │ 4 High Street, ME7 1BB   │ 1722410654151999488 │         │ local   │
╰───┴────────────────┴──────────────────────────┴──────────────────┴─────────┴─────────╯
```

The path parameter can be a list, allowing retrieval of multiple fields in one or more docs:

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)


```
──────────────────┬─────────┬─────
 cas              │ error │
──────────────────┼─────────┼─────
─────┤
│ 0 │ landmark_10019 │ ┌─────────┬──────────────────────────┐ │ 1722410659053961216 │         │
local    │
│   │                │ │ name    │ Royal Engineers Museum   │ │                     │         │
│
│   │                │ │ address │ Prince Arthur Road, ME4 4UG │ │                  │         │
│
│   │                │ └─────────┴──────────────────────────┘ │                     │         │
│
│ 1 │ landmark_10020 │ ┌─────────┬──────────────────────┐     │ 1722410654151999488 │         │
local    │
│   │                │ │ name    │ Hollywood Bowl       │     │                     │         │
│
│   │                │ │ address │ 4 High Street, ME7 1BB │   │                     │         │
│
│   │                │ └─────────┴──────────────────────┘     │                     │         │
│
└───┴────────────────┴────────────────────────────────────────┴─────────────────────┴─────────┴─────
─────┘
```

4.7. query

The query commands can be used to explore/create indexes and execute queries.

4.7.1. query

Takes a N1QL statement and executes it against the active cluster.

```
👤 Charlie 🏠 local in 🗄 travel-sample._default._default
> query "SELECT meta().id FROM `travel-sample`.inventory.landmark WHERE country = 'France'"
╭─────┬────────────────┬─────────╮
│   # │       id       │ cluster │
├─────┼────────────────┼─────────┤
│   0 │ landmark_10061 │ local   │
│ ... │      ...       │   ...   │
│ 387 │ landmark_9838  │ local   │
╰─────┴────────────────┴─────────╯
```

The query gets all the IDs of the docs where the country is France, then `cbsh` appends the cluster column to the results.

Named parameters are supported through the `--params` flag when the argument is a json object:

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
ountry = $country" --params {
```

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
| 387 | landmark_9838  | local   |
╰─────┴────────────────┴─────────╯
```

Multiple named parameters can be used at once, note there is no need to separate them with commas:

```
👤 Charlie 🏠 local in 🗄 travel-sample._default._default
> query "SELECT airline FROM `travel-sample`.inventory.route WHERE sourceairport = $aval AND distance
╭───┬─────────┬─────────╮
│ # │ airline │ cluster │
├───┼─────────┼─────────┤
│ 0 │ B6      │ local   │
│ 1 │ EK      │ local   │
│ 2 │ SV      │ local   │
╰───┴─────────┴─────────╯
```

The wildcard character '%' can be used the same as inside of the query statement. The following finds any IDs that match the regex 'hotel1002*'.

```
👤 Charlie 🏠 local in 🗄 travel-sample._default._default
> query "SELECT meta().id FROM `travel-sample`.inventory.hotel WHERE meta().id LIKE $pattern" --params
╭───┬──────────────┬─────────╮
│ # │      id      │ cluster │
├───┼──────────────┼─────────┤
│ 0 │ hotel_10025  │ local   │
│ 1 │ hotel_10026  │ local   │
╰───┴──────────────┴─────────╯
```

Positional parameters are also supported when the `--params` flag takes a list (https://www.nushell.sh/book/working_with_lists.html) argument:

```
👤 Charlie 🏠 local
> query "SELECT airline FROM `travel-sample`.inventory.route WHERE sourceairport = $1 AND distance > $
╭───┬─────────┬─────────╮
│ # │ airline │ cluster │
├───┼─────────┼─────────┤
│ 0 │ B6      │ local   │
│ 1 │ EK      │ local   │
│ 2 │ SV      │ local   │
╰───┴─────────┴─────────╯
```

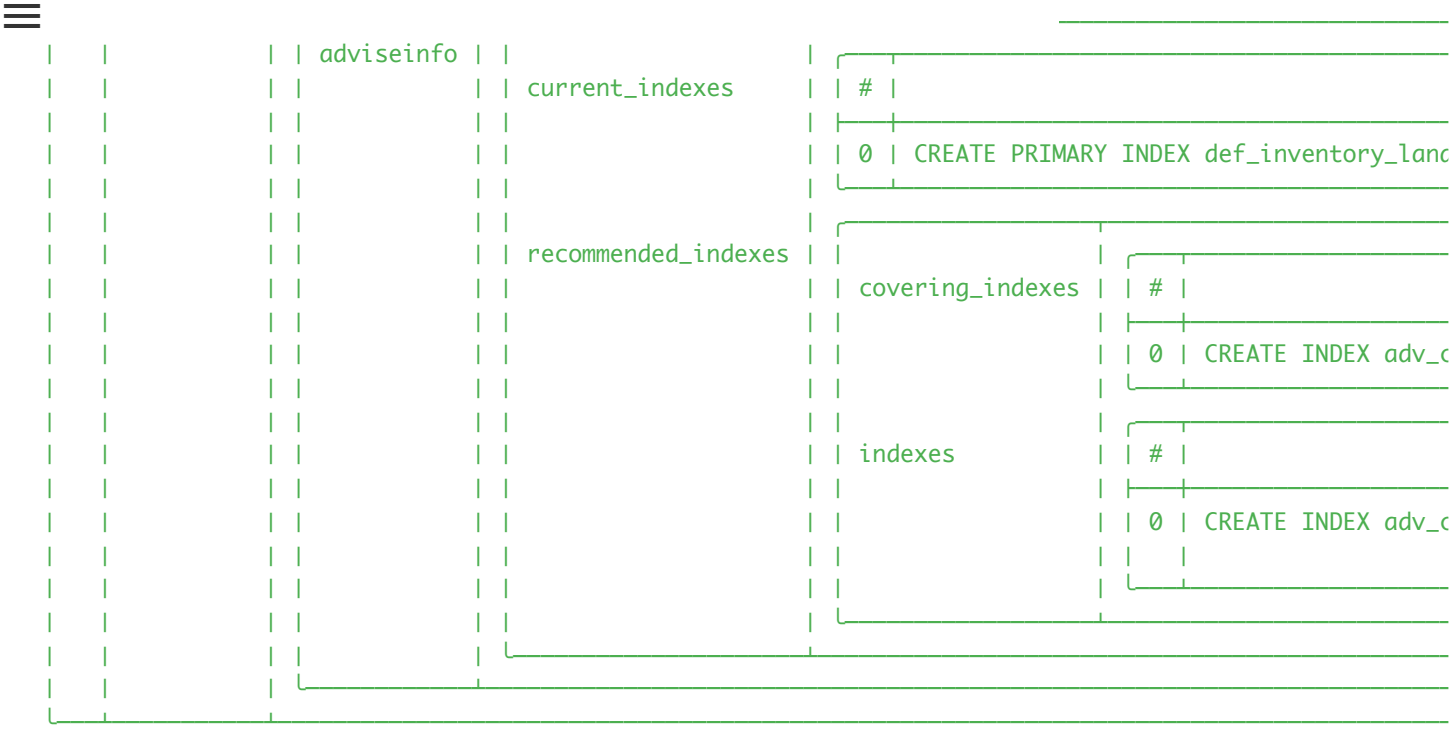
4.7.2. `query advise`

Helps you to learn about the indexes that your queries are using, and what indexes you could create to make them faster. For example we can take the first query from the `query` examples:

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
WHERE country = 'France'"
```

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)



This shows our query statement uses the Primary index `def_inventory_landmark_primary` and also recommends a covering index that we can create using `query` :

```
> query "CREATE INDEX adv_country ON `default`:`travel-sample`.`inventory`.`landmark`(`country`)"
```

4.7.3. `query indexes`

Lists all of the query indexes. If we had just created the recommended index at the end of the `query advise` section, the nushell command `find` can be used in conjunction with `query indexes` to check it was successfully created.

```
> query indexes | find adv_country
╭──┬──────────────┬───────────┬─────────────┬──────────┬─────────────┬─────────┬──────────┬─
│ # │    bucket    │ condition │  index_key  │ keyspace │    name     │ primary │  scope   │ s
├──┼──────────────┼───────────┼─────────────┼──────────┼─────────────┼─────────┼──────────┼─
│ 0 │ travel-sample │           │ ╭──┬─────╮ │ landmark │ adv_country │ false   │ inventory │ c
│   │              │           │ │ 0 │ `country` │ │          │             │         │           │
│   │              │           │ ╰──┴─────╯ │          │             │         │           │
╰──┴──────────────┴───────────┴─────────────┴──────────┴─────────────┴─────────┴──────────┴─
```

And if we want to check the definition this can be done using the `--definitions` flag:

```
> query indexes --definitions | find adv_country
╭──┬──────────────┬──────────┬────────────┬─────────────┬─────────┬──────────────┬──────────┬─
│ # │    bucket    │  scope   │ collection │    name     │ status │ storage_mode │ replicas │
├──┼──────────────┼──────────┼────────────┼─────────────┼─────────┼──────────────┼──────────┼─
│ 0 │ travel-sample │ inventory │ landmark   │ adv_country │ Ready  │ plasma       │        0 │ CREATE
╰──┴──────────────┴──────────┴────────────┴─────────────┴─────────┴──────────────┴──────────┴─
```

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

how `query indexes` can be

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

ıta. Check the recipes for how the

ipes.html#_similarity_search) or simple RAG (https://couchbase.sh/docs/recipes.html#_simple_rag).

4.8.1. `vector search`

Performs a vector search against a named vector index. Requires a vector to be used as the source of the search which can be supplied in a couple of formats.

For example imagine we have the following document stored in our cluster:

```
> doc get landmark_10019
╭───┬────────────────┬──────────────────┬──────────────────────────────────────────────────────────────────────────────╮
│ # │       id       │                  │                                                                              │
├───┼────────────────┼──────────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ 0 │ landmark_10019 │ ╭──────────────┬─┴──────────────────────────────────────────────────────────────────────────╮ │
│   │                │ │ title        │ Gillingham (Kent)                                                          │ │
│   │                │ │ name         │ Royal Engineers Museum                                                     │ │
│   │                │ │ alt          │                                                                            │ │
│   │                │ │ address      │ Prince Arthur Road, ME4 4UG                                                │ │
│   │                │ │ directions   │                                                                            │ │
│   │                │ │ phone        │ +44 1634 822839                                                            │ │
│   │                │ │ tollfree     │                                                                            │ │
│   │                │ │ email        │                                                                            │ │
│   │                │ │ url          │ http://www.remuseum.org.uk                                                 │ │
│   │                │ │ hours        │ Tues - Fri 9.00am to 5.00pm, Sat - Sun 11.30am - 5.00pm                    │ │
│   │                │ │ image        │                                                                            │ │
│   │                │ │ price        │                                                                            │ │
│   │                │ │ content      │ Adult - £6.99 for an Adult ticket that allows you to come bac              │ │
│   │                │ │              │  history of the British Empire. A quite extensive collection               │ │
│   │                │ │              │ collection of tank mounted bridges etc can be seen for free.               │ │
│   │                │ │              │ ticket.                                                                    │ │
│   │                │ │              │ ╭──────────┬─────────────────────────╮                                     │ │
│   │                │ │ geo          │ │ lat      │ 51.39                   │                                     │ │
│   │                │ │              │ │ lon      │ 0.54                    │                                     │ │
│   │                │ │              │ │ accuracy │ RANGE_INTERPOLATED      │                                     │ │
│   │                │ │              │ ╰──────────┴─────────────────────────╯                                     │ │
│   │                │ │ activity     │ see                                                                        │ │
│   │                │ │ type         │ landmark                                                                   │ │
│   │                │ │ id           │ 10019                                                                      │ │
│   │                │ │ country      │ United Kingdom                                                             │ │
│   │                │ │ city         │ Gillingham                                                                 │ │
│   │                │ │ state        │                                                                            │ │
│   │                │ │              │ ╭──────┬─────────╮                                                         │ │
│   │                │ │ contentVector│ │    0 │  0.02   │                                                         │ │
│   │                │ │              │ │  ... │   ...   │                                                         │ │
│   │                │ │              │ │ 1023 │ -0.00   │                                                         │ │
│   │                │ │              │ ╰──────┴─────────╯                                                         │ │
│   │                │ ╰──────────────┴────────────────────────────────────────────────────────────────────────────╯ │
╰───┴────────────────┴──────────────────────────────────────────────────────────────────────────────────────────────╯
```

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

ut to a vector search. Due to the

ushell.sh/commands/docs/select.html)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
╭───┬──────────────────────╮
│ # │  contentVector       │
├───┼──────────────────────┤
│ 0 │ ╭──────┬─────────╮   │
│   │ │    0 │  0.03   │   │
│   │ │ ...  │  ...    │   │
│   │ │ 1023 │ -0.00   │   │
│   │ ╰──────┴─────────╯   │
╰───┴──────────────────────╯
```

This can then be piped directly into `vector search` :

```
> doc get landmark_10019 | flatten | select contentVector | vector search landmark-content-index conte
╭───┬──────────────────┬──────────────────────────────────────────────┬─────────╮
│ # │        id        │                    score                     │ cluster │
├───┼──────────────────┼──────────────────────────────────────────────┼─────────┤
│ 0 │ landmark_10019   │ 340282350000000000000000000000000000000      │ local   │
│ 1 │ landmark_28965   │ 1.0286634                                    │ local   │
│ 2 │ landmark_3547    │ 1.0150017                                    │ local   │
╰───┴──────────────────┴──────────────────────────────────────────────┴─────────╯
```

We specify the vector index to be queried (landmark-content-index) as well as the name of the field on which the vector index was created (contentVector). See vector create-index for further explanation.

Subdoc get can also be used to fetch the source vector:

```
> subdoc get contentVector landmark_10019 | select content | vector search landmark-content-index cont
╭───┬──────────────────┬──────────────────────────────────────────────┬─────────╮
│ # │        id        │                    score                     │ cluster │
├───┼──────────────────┼──────────────────────────────────────────────┼─────────┤
│ 0 │ landmark_10019   │ 340282350000000000000000000000000000000      │ local   │
│ 1 │ landmark_28965   │ 1.0286634                                    │ local   │
│ 2 │ landmark_3547    │ 1.0150017                                    │ local   │
╰───┴──────────────────┴──────────────────────────────────────────────┴─────────╯
```

Here we don't need to flatten and specify the field since the content of the `sub doc` output already only holds the `contentVector` field.

Another format that is accepted is the output of vector enrich-text :

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
rk-content-index
```

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
| 1 | landmark_20732 | 0.7003826  | local   |
| 2 | landmark_21682 | 0.6987926  | local   |
└───┴────────────────┴────────────┴─────────┘
```

Note the `--dimension` flag used here, the vector provided as the source of the search must match the dimension of the vectors on which the index was created else you will get no results.

Finally if you just have a plain vector that is a list of floats this can be passed in as a final positional parameter:

```
> let vector = [0.1 0.2 0.3 0.4]
> vector search vector-index fieldName $vector
```

The vector here only has a dimension of 4 which is unrealistically low, but it is used for illustrative purposes.

The output of the search is a table containing the IDs of the `n` (specified by the `--neighbours` flag) documents with the most similar vector in the field named. The similarity of vectors is calculated using the metric specified on index creation.

To see the contents of the documents identified in the table simply pipe the output of vector search into a doc get (https://couchbase.sh/docs/#_doc_get) command. If only interested in a particular field then the subdoc get command will also work in the same way:

```
👤 Charlie 🏠 local in ☁ default._default._default
> vector search landmark-content-index contentVector $vector | subdoc get address
┌───┬────────────────┬──────────────────────────────────────┬──────────────────┬───────┬─────────
─┐
| # |       id       |               content                |         cas      | error | cluster
|
├───┼────────────────┼──────────────────────────────────────┼──────────────────┼───────┼─────────
─┤
| 0 | landmark_11956 | Hornchurch Road, Hornchurch, RM11 1JU | 1722871671832641536 |       | local
|
| 1 | landmark_25284 | 4040 Twiggs St                       | 1722871685304025088 |       | local
|
| 2 | landmark_7744  | Grandstand Road                      | 1722871709487202304 |       | local
|
└───┴────────────────┴──────────────────────────────────────┴──────────────────┴───────┴─────────
─┘
```

The environment is important when using the `vector search` command as the fully qualified index name is constructed from the active bucket and scope. The above command will be trying to use the index `default._default.landmark-content-index` . If the index was created against a different bucket/scope then you will get an index not found error.

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
    └────
  help: Unexpected status code: 400, body: {"error":"rest_auth: preparePerms, err: index not found","r
  ...
```

If you want to use an index that was created against a different bucket/scope to those that are active, this can be done with the `--bucket` and `--scope` flags.

```
👤 Charlie 🏠 local in 🗄 travel-sample.inventory._default
> vector search landmark-content-index contentVector $vector --bucket default --scope _default
╭───┬─────────────────┬──────────────────────────────────────────────┬─────────╮
│ # │       id        │                    score                     │ cluster │
├───┼─────────────────┼──────────────────────────────────────────────┼─────────┤
│ 0 │ landmark_10019  │ 340282350000000000000000000000000000000000 │ local   │
│ 1 │ landmark_16379  │ 1.0082568                                    │ local   │
│ 2 │ landmark_33857  │ 0.9897698                                    │ local   │
╰───┴─────────────────┴──────────────────────────────────────────────┴─────────╯
```

4.8.2. `vector create-index`

Creates a vector index against the active bucket/scope or the bucket/scope specified with the corresponding flags. For example the following will create an index named new-vector-index over the field fieldName in documents in the bucket `default` and the scope `_default`:

```
👤 Charlie 🏠 local in ☁ default._default._default
> vector create-index new-vector-index fieldName 1024
```

Any documents located elsewhere will not be indexed, and a vector search performed with this index will not find them. If the documents to be indexed were stored elsewhere then the environment would need to be changed with `cb-env bucket/scope` or the bucket and scope could be passed in with the appropriate flags.

The field parameter is the name of the field in the documents that contains the vector, and the dimension (length of the vector) parameter must match that of the vectors contained in the named field. Imagine there are documents with the following structure in the bucket `users` and scope `embeddedUsers`:

```
{
    "name": "my-name",
    "e-mail": "shell-user@couchbase.com",
    "description": "A rich textual description of the user.",
    "descriptionEmbedding": [0.0178287, 0.123098, -0.0189239, 1.109238],
}
```

In this example the descriptionEmbedding is the result of using a large language model to generate an embedding from the description field. The dimension of the vector (4) is unrealistically low but it's been chosen for illustrative purposes. To create an index over such documents in order to find users with similar descriptions the command would be:

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
                                                                                ‹et users --scope
  embeddedUsers
```

In the first example the create command is run from within the bucket and scope that the documents are stored in. Whereas in the latter the environment values need to be overwritten with the `--bucket` and `--scope` flags to correctly create the index.

4.8.3. `vector enrich-doc`

Enriches an existing json document by generating an embedding using the active llm from a named field and storing it in a new field in the same document.

For example take one of the documents from the travel sample data set:

```
> doc get landmark_10019
╭───┬─────────────────┬──────────────┬──────────────────────────────────────────────────────────────────────────────╮
│ # │       id        │              │                                                                              │
├───┼─────────────────┼──────────────┼──────────────────────────────────────────────────────────────────────────────┤
│ 0 │ landmark_10019  │ ╭────────────┬─────────────────────────────────────────────────────────────────────────╮ │
│   │                 │ │ title      │ Gillingham (Kent)                                                       │ │
│   │                 │ │ name       │ Royal Engineers Museum                                                  │ │
│   │                 │ │ alt        │                                                                         │ │
│   │                 │ │ address    │ Prince Arthur Road, ME4 4UG                                             │ │
│   │                 │ │ directions │                                                                         │ │
│   │                 │ │ phone      │ +44 1634 822839                                                         │ │
│   │                 │ │ tollfree   │                                                                         │ │
│   │                 │ │ email      │                                                                         │ │
│   │                 │ │ url        │ http://www.remuseum.org.uk                                              │ │
│   │                 │ │ hours      │ Tues - Fri 9.00am to 5.00pm, Sat - Sun 11.30am - 5.00pm                 │ │
│   │                 │ │ image      │                                                                         │ │
│   │                 │ │ price      │                                                                         │ │
│   │                 │ │ content    │ Adult - £6.99 for an Adult ticket that allows you to come back f        │ │
│   │                 │ │            │  of the British Empire. A quite extensive collection that takes         │ │
│   │                 │ │            │ tank mounted bridges etc can be seen for free. There is also an         │ │
│   │                 │ │            │ ╭──────────┬──────────────────────╮                                     │ │
│   │                 │ │ geo        │ │ lat      │ 51.39                │                                     │ │
│   │                 │ │            │ │ lon      │ 0.54                 │                                     │ │
│   │                 │ │            │ │ accuracy │ RANGE_INTERPOLATED   │                                     │ │
│   │                 │ │            │ ╰──────────┴──────────────────────╯                                     │ │
│   │                 │ │ activity   │ see                                                                     │ │
│   │                 │ │ type       │ landmark                                                                │ │
│   │                 │ │ id         │ 10019                                                                   │ │
│   │                 │ │ country    │ United Kingdom                                                          │ │
│   │                 │ │ city       │ Gillingham                                                              │ │
│   │                 │ │ state      │                                                                         │ │
│   │                 │ ╰────────────┴─────────────────────────────────────────────────────────────────────────╯ │
╰───┴─────────────────┴──────────────┴──────────────────────────────────────────────────────────────────────────────╯
```

We can use the content field to generate an embedding, and the result will be a new document that is the same as the original, with a new field containing the vector that is the result of the embedding generation:

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
|   |                | | name          | Royal Engineers Museum
|   |                | | alt           |
|   |                | | address       | Prince Arthur Road, ME4 4UG
|   |                | | directions    |
|   |                | | phone         | +44 1634 822839
|   |                | | tollfree      |
|   |                | | email         |
|   |                | | url           | http://www.remuseum.org.uk
|   |                | | hours         | Tues - Fri 9.00am to 5.00pm, Sat - Sun 11.30am - 5.00pm
|   |                | | image         |
|   |                | | price         |
|   |                | | content       | Adult - £6.99 for an Adult ticket that allows you to come bac
|   |                | |               | history of the British Empire. A quite extensive collection t
|   |                | |               | collection of tank mounted bridges etc can be seen for free.
|   |                | |               | ┌───────────┬──────────────────────┐
|   |                | | geo           | | lat       | 51.39                |
|   |                | |               | | lon       | 0.54                 |
|   |                | |               | | accuracy  | RANGE_INTERPOLATED   |
|   |                | |               | └───────────┴──────────────────────┘
|   |                | | activity      | see
|   |                | | type          | landmark
|   |                | | id            | 10019
|   |                | | country       | United Kingdom
|   |                | | city          | Gillingham
|   |                | | state         |
|   |                | |               | ┌──────┬─────────┐
|   |                | | contentVector | |    0 |  0.02 |
|   |                | |               | |  ... |   ... |
|   |                | |               | | 1535 | -0.01 |
|   |                | |               | └──────┴─────────┘
|   |                | └───────────────┴──────────────────────────────────────────────────────────────────────────────────────────────────
└───┴────────────────┴
```

The resulting document is the same as the original, but with a new field `contentVector` which contains the result of embedding the content field with the active llm. The name of the field that the embedding will be written to will default to the name of the original field with "Vector" appended. This default behavior can be overwritten with the `vectorField` flag. The resulting document is formatted with an id and content column which allows it to be piped into a `doc upsert` command to store it in the connected couchbase cluster.

```
> doc get landmark_10019 | vector enrich-doc content | doc upsert
Embedding batch 1/1
┌───┬───────────┬─────────┬────────┬──────────┬─────────┐
| # | processed | success | failed | failures | cluster |
├───┼───────────┼─────────┼────────┼──────────┼─────────┤
| 0 |         1 |       1 |      0 |          | local   |
└───┴───────────┴─────────┴────────┴──────────┴─────────┘
```

`vector enrich-doc` can enrich more than one document at a time. To repeat what we have done above on all the landmark documents we can get all the docs using a query and pipe the result directly into the `enrich-doc` command:

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
' | vector enrich-doc
```

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

ment id as well as the contents,
the same id as the original. If there
elect this and specify the field with

the `--id-column` flag. For example to use the "name" field as the id of the resulting documents do:

```
> query "SELECT name, * FROM `travel-sample` WHERE type = 'landmark'" | vector enrich-doc content --ic
```

4.8.4. `vector enrich-text`

Generates an embedding from the input text using the active_llm and outputs a json document containing the source text and the embedding.

It can be used directly on strings, which can be passed in as a positional parameter:

```
> vector enrich-text "some string" --dimension 5
Embedding batch 1/1
╭───┬─────────────────┬──────────────────────────────────╮
│ # │       id        │             content              │
├───┼─────────────────┼──────────────────────────────────┤
│ 0 │ vector-c9e02c   │ ╭──────────┬─────────────────╮   │
│   │                 │ │ text     │ some string     │   │
│   │                 │ │          │ ╭────┬────────╮ │   │
│   │                 │ │ vector   │ │ 0  │   0.31 │ │   │
│   │                 │ │          │ │ 1  │  -0.61 │ │   │
│   │                 │ │          │ │ 2  │  -0.21 │ │   │
│   │                 │ │          │ │ 3  │  -0.59 │ │   │
│   │                 │ │          │ │ 4  │  -0.38 │ │   │
│   │                 │ │          │ ╰────┴────────╯ │   │
│   │                 │ ╰──────────┴─────────────────╯   │
╰───┴─────────────────┴──────────────────────────────────╯
```

Or piped directly in:

```
> "some string" | vector enrich-text --dimension 5
Embedding batch 1/1
╭───┬─────────────────┬──────────────────────────────────╮
│ # │       id        │             content              │
├───┼─────────────────┼──────────────────────────────────┤
│ 0 │ vector-a269a7   │ ╭──────────┬─────────────────╮   │
│   │                 │ │ text     │ some string     │   │
│   │                 │ │          │ ╭────┬────────╮ │   │
│   │                 │ │ vector   │ │ 0  │   0.31 │ │   │
│   │                 │ │          │ │ 1  │  -0.61 │ │   │
│   │                 │ │          │ │ 2  │  -0.21 │ │   │
│   │                 │ │          │ │ 3  │  -0.59 │ │   │
│   │                 │ │          │ │ 4  │  -0.38 │ │   │
│   │                 │ │          │ ╰────┴────────╯ │   │
│   │                 │ ╰──────────┴─────────────────╯   │
╰───┴─────────────────┴──────────────────────────────────╯
```

Note that the dimension used here (5) is unrealistically low, but has just been chosen for illustrative purposes.

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

upsert it into the cluster:

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
| 0 |         1 |       1 |      0 |          | local   |
└---┴-----------┴---------┴--------┴----------┴---------┘
```

Or the output can be piped directly into `vector search` to find indexed docs with a similar vector. This makes `vector enrich-text` a very easy way to experiment with similarity search using strings. See vector search for an example of this.

`vector enrich-text` can also be used on larger chunks of text to produce multiple `vector docs` at once. For example say you have a text file called `some-text.txt`:

```
> open some-text.txt | vector enrich-text | doc upsert
Embedding batch 1/1
┌---┬-----------┬---------┬--------┬----------┬---------┐
| # | processed | success | failed | failures | cluster |
├---┼-----------┼---------┼--------┼----------┼---------┤
| 0 |        92 |      92 |      0 |          | local   |
└---┴-----------┴---------┴--------┴----------┴---------┘
```

When used on larger amounts of text `vector enrich-text` will split it into chunks of length 1024 by default, the length of the chunks can be changed with the `--chunks` flag. In this example the contents of `some-text.txt` was split into 92 chunks, resulting in 92 `vector docs` that were then upserted into the connected cluster.

Finally if you have a set of files within a directory that you want to generate `vector docs` from then you can list the files in the current directory with `ls` and pipe this list into `vector enrich-text`:

```
> ls | vector enrich-text | doc upsert
Embedding batch 1/1
┌---┬-----------┬---------┬--------┬----------┬---------┐
| # | processed | success | failed | failures | cluster |
├---┼-----------┼---------┼--------┼----------┼---------┤
| 0 |       278 |     278 |      0 |          | local   |
└---┴-----------┴---------┴--------┴----------┴---------┘
```

Here `vector enrich-text` will read each file, chunk the contents, retrieve the embeddings then generate the `vector docs`. Check this recipe (https://couchbase.sh/docs/recipes.html#_simple_rag) to see how `vector enrich-text` enables the implementation of simple Retrieval Augmented Generation.

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
Flags:
  -h, --help - Display the help message for this command
  --model <String> - the chat model to ask the question

Parameters:
  question <string>: the question to be asked
  context <any>: table of strings used as context for the question (optional)
```

`ask` can be used to ask a question of the active large language model. It can be used to answer a standalone question:

```
> ask "What is my favourite color"
I'm sorry, but as an assistant, I don't have access to personal information about you, such as your
favorite color. If you'd like to share it with me, I'd be happy to remember it for future reference!
```

Or context can be provided for the question as a list of strings. Either as a positional parameter:

```
> ask "What is my favourite color" ["My favourite color is blue"]
Your favourite color is blue.
```

This context can also be piped into the command which allows us to ask questions about docs in our database:

```
👤 Charlie 🏠 local in 🗄 travel-sample._default._default
> subdoc get content landmark_10019 | select content | ask "How much does it cost to go here?"
The admission price for an adult ticket is £6.99, which allows you to come back for further visits
within a year. There are also tickets available for children and concessionary rates. Additionally,
there is a collection of tank-mounted bridges and other outside attractions that can be seen for
free. Special event weekends are included in the cost of the annual ticket.
```

Since the context can be a list we can also use `ask` to summarize the results from various documents:

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
ask "what activities can I
```

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

```
e, you can explore various
ering. Activities may
```

```
1. Viewing exhibits on military equipment, vehicles, weapons, and uniforms.
2. Learning about the history of the British Empire and its impact on world events.
3. Participating in guided tours to gain more in-depth knowledge.
4. Attending special event weekends with themed activities and demonstrations.
5. Exploring the outside collection of tank-mounted bridges and other military hardware.

At the new restaurant, you can enjoy the following activities:

1. Dining on a menu that features burgers and ribs.
2. Appreciating the Hollywood-style decor and ambiance.
3. Socializing with friends or family in a lively atmosphere.
4. Trying out new dishes or drinks from the restaurant's menu.
5. Relaxing and unwinding in a cozy setting after a day of exploring or sightseeing.
```

The answering of questions with supplied context can be used to easily implement simple RAG.

4.10. `version`

The `version` command lists the version of the Couchbase shell.

```
> version
╭─────────┬─────────╮
│ version │ 0.92.0  │
╰─────────┴─────────╯
```

5. Reference

5.1. Config File Format

The `~/.cbsh/config` file with examples:

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

TOML

```
default-scope = "my-scope"
default-collection = "my-collection"
# The following can be part of the config or credentials
username = "Administrator"
password = "password"
# TLS defaults to on
# tls-enabled = false
# tls-cert-path = "/path/to/cert" # either accept all certs or provide a cert path
# tls-accept-all-certs = true
# tls-validate-hostnames = false

# User display name is optional and is used to display a different name to the username in the
prompt itself.
# This can be useful if the username that you are provided is a long randomly generated string or
similar.
# user-display-name = "Charlie"

# Timeouts broadly apply to the operations that you would expect them to.
# That is:
# * data: commands using the kv service such as `doc`
# * query: `query` commands
# * analytics: `analytics` commands
# * search: `search` commands
# * management: commands that perform management level operations, such as `users`, `bucket`,
`health` etc...
data-timeout = "10s"
query-timeout = "75s"
analytics-timeout = "75s"
search-timeout = "1m 15s"
management-timeout = "75s"
```

5.2. Credentials File Format

The optional `~/.cbsh/credentials` file with examples:

TOML

```
# Allows us to evolve in the future without breaking old config files
version = 1

[[cluster]]
identifier = "default"
username = "Administrator"
password = "password"
# TLS defaults to on, accepting all certs
# tls-enabled = true
# tls-cert-path = "/path/to/cert" # either accept all certs or provide a cert path
# tls-accept-all-certs = true
# tls-validate-hostnames = false
```

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Added support for bucket creation/management on Capella clusters
- Added support for loading sample data onto Capella clusters
- Added support for sub doc reads
- Added integration with large language models from Google, AWS and OpenAI
- Added support for named and positional query parameters
- Support document ids being ints for `doc` commands
- Support no expiry for collections
- Unset scope/collection when higher level entity changed
- Supports creation and management of Capella Clusters
- Added support for credential creation against Capella clusters
- Added support for creation of allowed CIDRs against Capella clusters
- Take username and password securely when using `cb-env register`
- Added support for managing Capella projects
- Added support for listing Capella organizations
- Enable use of `$nu` inside scripts and commands passed to cbsh

6.2. 0.75.2 - 2023-04-10

- Updated macOS build to not have any dependency on OpenSSL.
- Fixed the release workflow so that Linux release tarballs contain the `cbsh` binary.

6.3. 0.75.1 - 2023-04-13

This release contains a number of breaking changes, which are explicitly called out below. As our versioning continues to track the underlying Nushell minor version this has required breaking changes in a patch version.

- Updated config file to rename to (will continue to work).
- **Breaking** Updated config file to rename `hostnames` to `connstr` and changed the format to be a string.
- Added support, and detection, for different "cluster types"; Capella and Other. This allows us to modify behavior based on cluster type.
- **Breaking** Renamed `clusters health` to `health`.
- **Breaking** Renamed other `clusters …` commands to `database …`
- Replaced references to cluster with database.
- **Breaking** Removed support for whoami
- Added support for username aliases - added `display_name` to config.
- Trust the system store and Capella root CA when no certificate set.

Couchbase
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)
- Home
- Recipes
- Github (https://github.com/couchbaselabs/couchbase-shell)

- Fast fail `buckets` commands when used with Capella.
- Updated where config files are automatically written to.
- Fixed issue with config.nu file on Windows.
  - Pulled all beta and alpha release versions and updated version numbering
  - Couchbase Shell versions will now map to the Nushell version being used
  - Bundle Capella root CA to allow seamlessly connecting over TLS
  - Automatically detect when `query_context` should be sent
  - Update when SRV lookups are performed
  - Statically link OpenSSL
  - Various logging and error enhancements
  - Remove support for Capella InVpc
  - Renamed `clusters managed` to `cb-env managed`
  - Renamed `clusters register/unregister` to `cb-env register/unregister`
  - Expose CIDR in result of `clusters`
  - Fetch collection id over memcached rather than http

6.4. 0.75.0 - 2023-02-09

- Nushell pinned to 0.75
- Pulled all beta and alpha release versions and updated version numbering
- Couchbase Shell versions will now map to the Nushell version being used
- Bundle Capella root CA to allow seamlessly connecting over TLS
- Automatically detect when `query_context` should be sent
- Update when SRV lookups are performed
- Statically link OpenSSL
- Various logging and error enhancements
- Remove support for Capella InVpc
- Renamed `clusters managed` to `cb-env managed`
- Renamed `clusters register/unregister` to `cb-env register/unregister`
- Expose CIDR in result of `clusters`
- Fetch collection id over memcached rather than http

Last updated 2024-10-02 13:38:55 UTC