CouchDB Cheatsheet

A comprehensive guide to Apache CouchDB, covering basic concepts, querying, administration, and common tasks.

Core Concepts

Basic Definitions

Document	A JSON document that is the basic unit of data in CouchDB. It has a unique `_id` and `_rev`.
Database	A collection of documents. Each CouchDB instance can host multiple databases.
View	A function (written in JavaScript or another language) that transforms documents into a queryable index. Uses MapReduce.
MapReduce	A programming model for processing large datasets with a map function that transforms data and a reduce function that aggregates the mapped results.
Replication	The process of synchronizing databases between CouchDB instances, allowing for distributed data storage and offline access.
Conflicts	Occur when the same document is updated concurrently on different nodes during replication. CouchDB resolves conflicts by choosing a winning revision and storing the conflicting revisions as a history.

Document Structure

A CouchDB document is a JSON object with special fields:

- `_id`: Unique identifier for the document (string).
- `_rev`: Revision token (string), used for optimistic concurrency control. Updated with each modification.
- Other user-defined fields: Contain the actual data.

Example:

```
{
  "_id": "doc1",
  "_rev": "1-
6484e3cf6594867333363a7b539a0a1b",
  "name": "John Doe",
  "age": 30,
  "city": "New York"
}
```

Key Concepts Illustrated

Imagine a database of books.

- **Document:** Each book is a document with fields like title, author, and ISBN.
- **Database:** All book documents are stored in a database named `books`.
- **View:** A view could be created to list books by author or to find books published in a specific year.
- **Replication:** Replicating the `books` database to multiple servers ensures data availability and allows users in different locations to access the book information.

Querying with Views

Creating Views

Views are defined within design documents. A design document is a special document whose `_id` starts with `_design/`.

Example:

```
{
  "_id": "_design/books",
  "views": {
    "by_author": {
      "map": "function (doc) { if (doc.author)
{ emit(doc.author, doc.title); } }"
    }
  }
}
```

This creates a view named `by_author` inside the `books` design document. The map function emits the author as the key and the book title as the value.

Querying Views

To query a view, use the following URL:

```
GET
/<database>/_design/<design_doc>/_view/<view_n
ame>
```

Example:

```
GET /books/_design/books/_view/by_author?
key="John Doe"
```

This will return all books by John Doe.

Common Query Parameters:

- `key`: Match documents with a specific key.
- `startkey`, `endkey`: Define a key range.
- `limit`: Limit the number of results.
- `skip`: Skip the first N results.
- `descending`: Return results in descending order.

MapReduce Functions

Map Function	Processes each document and emits key-value pairs. The `emit(key, value)` function is used to create index entries.
Reduce Function	Aggregates the results of the map function. It takes keys, values, and a `rereduce` flag as input.
Example Map Function	`function (doc) { if (doc.type === 'comment') { emit(doc.post_id, 1); } }`
Example Reduce Function	`function (keys, values, rereduce) { return sum(values); }`

CouchDB API

Database Operations

Create Database	`PUT /<database>`
Get Database Info	`GET /<database>`
Delete Database	`DELETE /<database>`
List All Databases	`GET /_all_dbs`

Document Operations

Create Document	`POST /<database>`
	`Content-Type: application/json`
	
	`{ ... }`

Get Document	`GET /<database>/<document_id>`

Update Document	`PUT /<database>/<document_id>`
	`Content-Type: application/json`
	
	`{ ... }`

Delete Document	`DELETE /<database>/<document_id>?rev=<revision>`

Bulk Operations

Bulk operations allow you to perform multiple document operations in a single request, improving performance.

```
POST /<database>/_bulk_docs
Content-Type: application/json

{
  "docs": [
    { ... },
    { ... }
  ]
}
```

Administration and Maintenance

Configuration

CouchDB is configured through a configuration file (`local.ini`) or via the API.

Key Configuration Sections:

- `[couchdb]` : Core CouchDB settings.
- `[httpd]` : HTTP server settings.
- `[log]` : Logging settings.
- `[replicator]` : Replication settings.
- `[query_server_config]` : Javascript query server settings.

Example Setting:

```
[httpd]
port = 5984
```

Replication

CouchDB supports continuous and one-time replication. Replication can be configured using the `_replicator` database or via the command line.

Example Configuration Document in `_replicator`:

```
{
  "_id": "replication_job_1",
  "source": "http://source-couchdb:5984/source_db",
  "target": "http://target-couchdb:5984/target_db",
  "continuous": true
}
```

Compaction

Compaction removes unused data and optimizes database storage. It is triggered automatically, but can also be initiated manually.

To compact a database:

```
POST /<database>/_compact
```

To compact a design document (and its views):

```
POST /<database>/_compact/<design_document>
```

Security

Authentication	CouchDB supports various authentication methods, including Basic Authentication and Cookie Authentication.
Authorization	Access control is managed through roles. Users can be assigned roles to grant them specific permissions.
Admin Party	A mode where all requests are processed as an administrator. It is recommended to disable the admin party in production environments.