Distributed Computing Accounting (dca) Usage Document

Introduction

dca is an accounting RESTful API that keeps track of job processing hours used by users and charges the respective tenant using Elasticsearch, a NoSQL document-based database. It is meant to be used in conjunction with a backend transactions system and optionally a web-based interface for viewing transactions and bills. dca can be interacted with through commands and outputs a json string containing the results.

Its data follows the hierarchy of having individual tenants that oversee a number of projects, each of which has a list of users that work on said project. Each tenant has their own budget which can then be distributed amongst their projects to be used by the users. dca has the ability to track the usage of budget on a per-user basis, and create a bill for the administrator that overviews how much each user spent each day. On top of having a basic budget system, dca also has a credit system that allows an administrator to dispense credit to tenants who may have underused their budget one month and require more the next. To maintain security, dca has integrated Elasticsearch’s privilege and role system into its own. dca is quite flexible and is simple to set up.

Setup

To use dca, Elasticsearch 5 (and consequently Java 1.7.0+) should have already been installed and properly configured. From here, Python 3+, Elasticsearch-py, and Elasticsearch XPack of the same version as ES should also be installed. Assuming an internet connection is available, the associated commands are:

yum install python-pip

pip install elasticsearch==5

%ESHOME%/bin/elasticsearch-plugin install x-pack –batch

Replace %ESHOME% with the installation path of elasticsearch. By default, the path is /usr/share/elasticsearch.

The default password of the ES superuser (name: elastic) is changeme. To change this password, run the command

curl -XPUT -u elastic 'localhost:9200/\_xpack/security/user/elastic/\_password' -d '{

"password" : "newpassword"

}'

Next, create a directory and download the dca files from <https://github.com/Logikable/dca>. From this directory, run ./dca\_setup in order to configure dca. Most likely, when the setup process is first run, you will be prompted to create the directory ~/.dca. After creating the directory, run the setup file once more. This time, a config file will be created and you will be prompted to fill out the relevant information. The location of the config file is ~/.dca/config. As of now, the necessary information are a set of ES credentials (username and password) and a comma-separated list of ES hosts. By default, it is recommended to use the ES superuser.

Once this information has been entered into the config file, **./dca\_setup should be run one last time** to create the necessary ES indices and mappings. It should not be run again unless the entire dca ES database is meant to be reset.

dca is now properly set up!

Commands

dca comes with a variety of features for your accounting needs. Along with basic commands to add, disable, or modify tenants, projects, and users, it allows for an easily customizable rate for which tenants will be charged at. **Disclaimer:** **arguments should be kept case consistent. Failure to do so may cause errors.**

All of the supported commands are listed below:

Tenant related commands – adding, disabling, adding credit, adding balance:

dca tenant add --tenant=<name> [--credit=<amount>]

dca tenant disable --tenant=<name> [-y]

dca tenant modify --tenant=<name> --credit=<amount>

dca tenant payment --tenant=<name> --payment=<amount>

Project related commands – adding, disabling, and moving budget around between projects and the tenant:

dca project add --project=<name> --tenant=<name> [--balance=<amount>] [--credit=<amount>]

dca project disable --project=<name> [-y]

dca project movebudget --from=<name> --to=<name> [--balance=<amount>] [--credit=<amount>] --type=p2p|t2p|p2t

User related commands – adding and deleting users:

dca user add --project=<name> --user=<name>

dca user delete --project=<name> --user=<name>

Listing information about a specified tenant, project, user, or any combination of the three:

dca list [--tenant=<name>] [--project=<name>] [--user=<name>]

Rate related commands – setting and getting the rate:

dca rate set --rate=<rate>

dca rate get

Transaction related commands – reserving part of a project’s budget and charging the project:

dca transaction reservebudget --project=<name> --estimate=<time>

dca transaction charge --project=<name> --user=<name> --estimate=<time> --jobtime=<time> --start=<time>

Generating a bill:

dca bill generate --project=<name> --time\_period=last\_day|last\_week|last\_month|<date>,<date>

There are restrictions and guidelines to what the argument values can be. All restrictions are listed below.

* By default, all of the outputs will be pretty printed (human readable format). In order to receive a mini output (flat, no indentation), use the optional argument -m or --mini.
* Tenant, project, and usernames must be alphanumeric and less than (or exactly) 32 characters in length. **Project names must be unique.**
* Deleting tenants and projects does not actually delete them. It merely disables them (which requires a confirmation). This is so that bills can still be generated
* Balance, credit, and payments amounts must be a non-negative numeric value.
* The type argument under project movebudget determines whether the from and to destinations are projects (p) or tenants (t).
* Transaction estimates and jobtimes are measured in seconds, and must be an integer value.
* The start time must be in the format YYYY-MM-DD HH:mm:ss, or a Unix epoch value.
* time\_period (if a default one is not chosen) must be two dates in the format YYYY-MM-DD.

User/role support

Although dca does support custom ES roles and users, a RESTful API for creating, modifying, and deleting roles and users has not been created. This section acts as a guide to creating your own roles and users. **Disclaimer: it is the administrator’s responsibility to ensure that roles are properly configured. dca will not function if a necessary role was not provided to the user.**

From here on, all commands should be run in the ES installation folder (/usr/share/elasticsearch by default).

The first step to setting up permissions for ES is to create a user. The command is

bin/x-pack/users useradd <username>

The guidelines that ES provides for creating a username are as follows: A username must be at least 1 character and no longer than 30 characters. The first character must be a letter (a-z or A-Z) or an underscore (\_). Subsequent characters can be letters, underscores (\_), digits (0-9), or any of the following symbols @, -, . or $.

You will then be prompted for a password. Alternatively, if command line history security is not an issue, a password can be provided in the original command using a -p <password> argument.

To add a role, a curl command can be run. Existing roles can be modified by running the same command, and the changes will be made. Here is a sample command that is annotated with the details of what each field does:

curl -XPOST -u USERNAME 'localhost:9200/\_xpack/security/role/ROLENAME?pretty' -d'

{

"indices":[{

"names": ["dca"],

"privileges": ["all"],

"field\_security" : {

"grant" : ["category", "@timestamp", "message"]

},

"query": "{\"match\": {\"category\": \"click\"}}"

}]}'

* The USERNAME field specified in the command itself (not in the body) is the user that will be running this command. It can be any user that has permissions to update or write to \_xpack/security/role/, but since this is your first user/role, chances are the only user with those permissions you have right now is the SU elastic. Use that username for now.
* The ROLENAME field is the name of the role you wish to create.
* The names field is a list of the names of the indices that you want these privileges apply to. For example, to give a role the ability to write to the tenant index, this field would be [“tenant”].
* The privileges field represents the privileges that this role will have in the specified indices. See below for a comprehensive list of privileges.
* Field\_security allows you to grant or deny privileges to specific fields within documents in the specified indices.
* Query allows this role to have the aforementioned privileges only when this query is satisified. In this case, this role can only make changes to documents in the index “dca” that have a field “category” with the value “click”.

Once you have created the roles you need, the roles can be then added to the users created earlier using the command:

bin/x-pack/users roles <username> -a <comma-separate list of roles> -r <comma-separated list of roles>

The -a roles will be added to the user specified by the username, and the -r roles will be removed.

For further reading, here are a few links that go into more detail about users, roles, and permissions:

<https://www.elastic.co/guide/en/x-pack/current/file-realm.html>

<https://www.elastic.co/guide/en/x-pack/current/defining-roles.html>

<https://www.elastic.co/guide/en/x-pack/current/security-privileges.html>

Database Structure

In order to properly create roles, you will find that the structure and mappings of the database need to be understood. The details will be listed here, and can also be found in the dca\_setup file as well.

dca consists of 6 separate indices, specifically, tenant, project, transaction, payment, rate, and log. Their names should be self-explanatory. Using each of the commands requires the following permissions (**note: all commands except those that involve transactions or listing/billing require being able to index to log**):

* tenant add: searching tenants
* tenant disable: searching tenants, getting projects, updating projects, updating tenants
* tenant modify: searching tenants, searching projects, updating tenants
* tenant payment: searching tenants, updating tenants, indexing payments
* project add: searching tenants, searching projects, getting rates, update tenants, index projects
* project disable: searching projects, updating projects
* project movebudget: searching tenants, searching projects, updating projects
* user add: searching projects, updating projects
* user delete: searching projects, updating projects
* list: searching tenants, getting projects, searching projects
* rate set: update rates, updating projects
* rate get: get rates
* transaction reservebudget: searching projects, updating projects
* transaction charge: searching projects, updating projects, updating tenants, indexing transactions
* bill generate: searching projects, getting tenants, searching transactions, searching payments

Below you can find their fields and field types:

/tenant/tenant/

{

"name": text,

"balance": float,

"credit": float,

"projects": [text],

"d": boolean

}

/project/project/

{

"tenant": text,

"project": text,

"balance": float,

"credit": float,

"requested": float,

"rate": float,

"users": [text],

"d": boolean

}

/transaction/transaction/

{

"project": text,

"user": text,

"start": date,

"end": date,

"runtime": integer,

"cost": float

}

/payment/payment/

{

"tenant": text,

"date": date,

"payment": float

}

/rate/rate/

{

"rate": float

}

/log/log/

{

"category": text,

"action": text,

"details": text,

"date": date

}

Future Plans

As is the case with a lot of software, dca has room for improvement. In the future, dca will see the full integration of an identification-permissions system, with infrastructure for a hierarchy of roles that can only change those below it. We hope to provide a layer of abstraction for the user to simplify the process of creating roles, as Elasticsearch’s current system is improperly documented and heavyweight.