Since we are working under NDA, I have removed all variables, constants that define URLs of the targeted tested system. That's why HTTP request defaults configuration element is empty, and the user and script variables, related to the URLs are incomplete as well.

The set of scripts in the **d\_profile** folder imitates load for an application, that manages graphical contents. The application allows users to search, get, update, create, delete contents and to perform social activities such as commenting, managing comments. The graphical contents can be organized in collections. Some of the activities, like search, get contents, collection and information are available for anonymous users. Others, that connected with creation, update, deletion are available only for registered ones. There are three variants of script in the folder, the difference between them is little. **3dw\_profile\_400u\_lastUpdate.jmx** is the one, that is primarily used. All these scripts imitate above mentioned user activities. The primary usage of these scripts is to check before every release, that the tested application provides the same performance metrics. Releases occur every two weeks. Test is executed on the dedicated stack, that is close in its configuration to production environment. The production data is deployed on the load stack. The number of users selected for script, the rates of every API call are selected from the analysis of the real data. API calls are logically gathered into thread groups, according to use cases. All thread group run at the same time, not consecutively. In the most of thread groups the API calls are executed both for registered and anonymous users. The variables related to rates are set in the user defined variables area, so that we can tune them if it's necessary. Most of the API calls take data from csv files, that are numerous in the script. Since many csv files have size up to 30-40 Mb, I cut them to the reasonable size for this demo. Each csv file has real data, taken from the production db: id of entities, collections, users, activities, e.t,c. Logs had been analyzed before composing csv data files. For instance, for search calls, we modeled csv data files from real statistics: parameters passed in the request, length of searched data, e.t.c.. The architecture of the system requires to authenticate users in one domain, while using the ticket, that users get, if they are authorized, in the domain of the system under test. That's why there's bean shell script, that places cookie with ticket to the targeted domain. The configuration element, that is called servers URL is one, that keeps URLs of servers of the tested system. We need to do it this way, as we in fact have a few load stacks and since we perform load testing, using Blazemeter, we quickly change servers.csv file in the test scripts set to switch from one environment to another. It's quicker and eliminates possible error, than to change script every time. As I've mentioned we execute test from Blazemeter. The resource utilization data (CPU, Mem, e.t.c.) we collect from the DataDog, that is configured for our load stack.

The application, that is tested with the script in the **v\_profile** folder is application for management users media contents. The user of that application can add, update, manage, remove contents and perform some social activities, like commenting, e.t.c. The script architecture is similar to the previous one: each thread group represents a certain use case, data for API call parameters are taken from csv files. As this application is not in the production yet, we generate all data for testing, before load testing script is launched. While data are being generated, the csv files are created with parameters for script API calls.