Quick Start For Using CLI Tools To Automate Zend Server (Single / Cluster)

Applies to

Zend Server v.6 and up
Linux Platforms (According to System Requirements)
zs-manage.sh - available in all versions
zs-client.sh - available starting from v.8, and also as a Github community project called ZendServerSDK

Note:
Zend Server Small Business Edition does not have all Web APIs enabled.

Introduction

Zend Server provides Web API infrastructure for most of its management features.
In addition to the Admin UI in browser, which is available on *http://hostname:10081* and *https://hostname:10082*, many tasks for initial launch, configuration, deployment, restart and so on can be performed from good old Command Line.

zs-manage.sh

Starting with **zs-manage.sh**, which resides under **/usr/local/zend/bin** (I recommend to place this quickly in your system / user profile **$PATH** for everyday use).
zs-manage.sh was introduced in Zend Server for automating environments, such as development servers and cloud systems.
When you execute it with "help" parameter, you will get the complete set of commands and options.

You can use **zs-manage.sh** for kick-starting a Zend Server installation, by performing a Bootstrap (Initial Launch) with parameters such as admin password (and developer password if you are on Zend Server edition which is NOT "Developer", where the admin user is basically a developer by nature), License order number and key, Accepting EULA (just say TRUE for yes) and more.
In response to successful bootstrap, you get as output a freshly generated Admin Web API key, which is also available in the Zend Server Admin UI under Administration -> Web API.

```
# zs-manage.sh bootstrap-single-server \

-o <ORDER_NUMBER> \

-l <LICENSE_KEY> \

-a TRUE \

-r <PRODUCTION_TRUE_FALSE> \

-e <ADMIN_EMAIL> \

-d <DEVELOPER_PASS> \

-p <ADMIN_PASS> > /tmp/admin.key
```

Once you have a bootstrapped (launched) Zend Server, you can keep the admin Web API key for performing additional manual / automated tasks from**zs-manage**. In the above command, I saved the output of the bootstrap command to a temporary key file, but it can also be kept for longer time / additional sessions with the following hint.

Hint:
The output of **zs-manage.sh bootstrap** action is a bit verbose so I clean it automatically when provisioning new Zend Server machines. The final admin key which I refer to below is "purified" HASH, and this can be achieved by the following one-liner, which also blocks it from being accessed from non-root users and scripts:

```
# echo -n `head -1 /tmp/admin.key` > /usr/local/zend/admin.key; chmod 600 /usr/local/zend/admin.key; rm /tmp/admin.key
```

A real world example now. This will store a memory limit directive for OPCache on PHP 5.5 (earlier version is called Zend Optimizer Plus), and restart PHP to take effect:

```
# zs-manage.sh store-directive -d "opcache.memory_consumption" -v "99" -N admin -K `cat /usr/local/zend/admin.key`

# zs-manage.sh restart-php -N admin -K `cat /usr/local/zend/admin.key`
```

To apply many configuration changes at once, you can manually edit the relevant set of ini files on the server, and once done, apply the changes to the configuration blueprint, as shown below. If the ini changes are not applied via "config-apply-changes" command, you will get conflict notification messages in the admin UI, and, to resolve the the conflicts, Zend Server will rewrite your modifications at a later time. Make sure your new directives/values are modified properly, so you don't get any errors or problems loading components (alas, logs are the admin's best friends).

```
# zs-manage.sh config-apply-changes -N admin -K `cat /usr/local/zend/admin.key`
```

Hint:
When you wish to manage a Zend Server Cluster instead of a single server, it works the same way.
However, your admin Web API key now comes from the cluster database and not from the local server - make sure you are not using a local server admin key in this case.
If you have several Zend Server machines you wish to manage remotely using zs-manage, you can wrap the **zs-manage.sh** commands and apply ZS URL and admin Web API key from some resource, according to the target Server / Cluster you wish to manage.

Sounds complicated to manage multiple servers... Now, you are probably asking yourself, can this be modernized a little bit? Or, Directly Manage multiple Zend Server endpoints in the CLI tool? Or, how to utilise more Web APIs which are provided by Zend Server backend?

OK. You asked for it, You got it!
This is where **zs-client.sh** comes in to the rescue.

zs-client.sh

Since version 8 of Zend Server, you no longer need to GIT CLONE the ZendServerSDK (unless you want bleeding edge or newer branch) from Github.
It is officially packaged under */usr/local/zend/bin* (or already in your system / profile **$PATH**, if you read the first part).
zs-manage.sh leverage a huge set of Web APIs from Zend Server and can be used on multiple Zend Server targets out of the box.

This is how the above example is accomplished with zs-client, with the additional option to add a new Zend Server target, and use instantly with all available Web APIs:

```
# zs-client.sh addTarget --target=<TARGET_NAME> --zskey=admin --zssecret=<HASH>
# zs-client.sh configurationStoreDirectives --directives="opcache.memory_consumption=128" --target=<TARGET_NAME>
```

You can review complete usage of zs-client.sh by running the command without parameters.

Output Format: By default, XML is returned as output from the Web API methods. You can also choose KV (Key->Value) and JSON, if it helps you parse or save the responses better. Add the command params for output as **--output-format=xml|json|kv**

Further Adventures, if you chose the Red Pill:

You can check the Zend-Patterns@Github repos to find out some nicely done integrations with Puppet, Chef, Vagrant, Docker and more Configuration Management systems. There are quite a lot of Z-Ray extensions there to get for your favorite PHP apps. The Integrations with CI / CM / CD systems are merely calling Zend Server CLI inside other systems flows, so if you're one of the lucky people who happens to own a DevOps consultancy shop (or dream about it while coding PHP in Zend Studio), and happen to own Zend Server as well, you can create some magic to save great deal of time while automatically provisioning all kinds of Zend Server installations with applications, libraries, configuration and more. Be sure to check our Continuous Delivery Blueprint on your next visit to zend.com, if you're interested (not hard to find :)

Links

Zend Server Web API Reference
Zend Server SDK on Zend-Patterns@Github
Zend Server Web API Module for Zend Framework 2
Zend Server Edition for Production
Zend Server Editions for Development