

Thomas Fahle

Perl in the Cloud - OpenShift Express by Red Hat

OpenShift by Red Hat soll die Entwicklung von Open-Source-Anwendungen für die Cloud vereinfachen. Diese *Platform as a Service* stellt eine Infrastruktur für verschiedene Programmiersprachen, Web-Frameworks und Open-Source-Anwendungen zur Verfügung.

Red Hat unterscheidet zwischen den Produktvarianten Express, Flex und Power. Die kostenlose Express Variante, für die nur eine Registrierung erforderlich ist, erlaubt u.a. die Verwendung der dynamischen Programmiersprache Perl in Version 5.10.1, der Datenbanken MySQL in Version 5.1, SQLite in Version 3, MongoDB in Version 2 und (ganz wichtig) die Installation von CPAN-Modulen. Die bekannten und beliebten Perl-Web-Frameworks *Dancer*, *Mojolicious* und *Catalyst* können verwendet werden.

Schritt für Schritt

Dieser Beitrag geht zuerst Schritt für Schritt durch die Installation, Initialisierung und Konfiguration der Clientprogramme von OpenShift Express. Danach wird ein betont einfaches, aber nützliches Perl-Programm erstellt und gezeigt, wie dieses in die Cloud ausgeliefert wird (Deployment). Im letzten Abschnitt werden noch ein paar kleine Tricks gezeigt, die den Umgang mit einer App vereinfachen.

Registrierung

Zur Verwendung von OpenShift Express ist eine Registrierung mit einer gültigen E-Mail-Adresse erforderlich. Alternativ kann auch ein bestehender Account des Red Hat Network (RHN) verwendet werden.

Installation der Client-Tools

OpenShift Express wird über eine Sammlung von Kommandozeilen-Tools verwaltet. Die Client-Tools sind in Ruby geschrieben und laufen auf Mac OSX, Linux und Windows (Cygwin). Zum Deployment der Apps wird Git verwendet. Die Kommunikation zwischen Client und OpenShift ist per SSH verschlüsselt.

Installation unter Red Hat/CentOS/Fedora

Für die Client-Tools bietet RedHat eine eigenes YUM-Repository an, das wie folgt installiert wird.

```
# wget https://openshift.redhat.com/app/  
repo/openshift.repo  
# mv openshift.repo /etc/yum.repos.d
```

Damit alle Paketabhängigkeiten aufgelöst werden, musste ich auch die EPEL- und RPMForge-Repositories hinzufügen. Nun können die Client-Tools installiert werden:

```
# yum install rhc
```

Die Clientprogramme befinden sich nun im Ordner `/usr/bin/` und sind damit direkt aufrufbar, da sie im Pfad enthalten sind.

Installation unter Ubuntu/Debian/SUSE

Für Ubuntu/Debian/SUSE werden Ruby in Version 1.8 einschließlich Entwicklerpaketen, Rubygems, ein passendes OpenSSL und Git benötigt. Nach der Installation der Pakete können die Client-Tools als `rubygems` installiert werden.

Beispielinstallation unter Ubuntu 10.04 LTS

```
sudo apt-get install ruby ruby-dev  
sudo apt-get install libopenssl-ruby  
sudo apt-get install rubygems  
sudo apt-get install git-core  
  
sudo gem install rhc
```



Die Clientprogramme befinden sich nun im Ordner `/var/lib/gems/1.8/bin/` und sind nur über die Angabe des kompletten Pfades erreichbar.

Abhilfe schafft hier eine Änderung der Umgebungsvariablen PATH:

```
export PATH="/var/lib/gems/1.8/bin:$PATH"
```

Diese Änderung lässt sich auch in `~/.bashrc` dauerhaft ablegen.

Initialisierung - Namensraum festlegen

Alle Apps eines Users werden in einen eigenen Namensraum (Domain) installiert. Apps sind dann nach dem Schema `http://$app-$domain.rhcloud.com` öffentlich erreichbar. Dazu gleich mehr.

Das Kommando `rhc-create-domain` erzeugt einen neuen Namensraum, die Konfigurationsdatei `express.conf` und SSH-Schlüssel `libra_id_rsa` zur Git-Authentifizierung. Über die Option `-n` wird der Namensraum festgelegt, die Option `-l` nimmt den OpenShift-Benutzernamen entgegen.

Weitere Optionen können über den Schalter `-h` angezeigt oder der manpage zu `rhc-create-domain` entnommen werden.

```
$ rhc-create-domain -n yourdomain \  
  -l user@example.com  
Password: <user password>  
  
Generating Openshift Express ssh key to  
  /home/UserName/.ssh/libra_id_rsa  
Generating public/private RSA key pair.  
Created directory '/home/UserName/.ssh'.  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in  
  /home/UserName/.ssh/libra_id_rsa.  
Your public key has been saved in  
  /home/UserName/.ssh/libra_id_rsa.pub.  
.  
.  
Contacting https://openshift.redhat.com  
Adding rhlogin to  
  /home/UserName/.openshift/express.conf  
Creation successful  
  
You may now create an application.  
Please make note of your local config file  
in /home/UserName/.openshift/express.conf  
which has been created and populated for  
you.
```

Jetzt noch Git (minimal) konfigurieren:

```
$ git config  
  --global user.name "Your Name"  
$ git config  
  --global user.email you@example.com
```

Dann kann endlich die erste App erstellt werden.

App-Gerüst erzeugen

Das Kommando `rhc-create-app` erzeugt das Gerüst der neuen App. Über die Option `-a` wird der Name der Applikation angegeben. Die Option `-t` legt den Typ der Applikation, hier `perl-5.10`, fest. Typen werden in der Dokumentation auch gerne als `cartridge` bezeichnet.

```
$ rhc-create-app -a X1 -t perl-5.10  
Password:  
  
Attempting to create remote  
application space: X1  
Now your new domain name is being  
propagated worldwide  
(this might take a minute)...  
Pulling new repo down  
...  
Confirming application 'X1' is available  
Attempt # 1  
  
Success! Your application 'X1' is now  
published here:  
  
  http://X1-perlhowto.rhcloud.com/  
...  
  
To make changes to 'X1', commit to X1/.  
Successfully created application: X1
```

Die neu erstellte Applikation mit dem Namen `X1` innerhalb des von mir gewählten Namensraumes `perlhowto` ist sofort unter der URL `http://x1-perlhowto.rhcloud.com/` erreichbar. Alle Applikationen sind auch über SSL (https) erreichbar.

Wer lieber seine eigene Domain verwenden möchte und über einen eigenen Nameserver verfügt, kann einen DNS-Alias (CNAME) einrichten. Eine Anleitung befindet sich in den OpenShift Express FAQs.



Orientierung im Gelände

Die neu erstellte Applikation befindet sich im Verzeichnis `$app`, hier `X1`.

```
$ tree X1
X1
|-- deplist.txt
|-- libs
|-- misc
|-- perl
|   |-- health_check.pl
|   |-- index.pl
|-- README
```

Das Verzeichnis `perl` ist die `DocumentRoot` der Webapp. Alle hier abgelegten Dateien sind öffentlich. Die Datei `index.pl` dient als `DirectoryIndex`.

Die Datei `deplist.txt` nimmt eine Liste der zu installierenden CPAN-Module auf, pro Zeile ein Modul ohne Versionsnummer.

Beispiel:

```
YAML
Dancer
Plack::Handler::Apache2
```

Die hier angeführten Module werden auf dem Cloud-Server mittels `cpanm` installiert.

Das Verzeichnis `misc` ist nicht öffentlich und kann für eigene Zwecke genutzt werden.

Das ebenfalls nicht öffentliche Verzeichnis `libs` dient als Speicherort für eigene Perl-Module.

Der Sinn und Zweck der Datei `health_check.pl` ist mir nicht ganz klar geworden, das Programm kann aber zum Monitoring eingesetzt werden.

Weiterhin gibt es noch ein verstecktes Verzeichnis `.openshift` zur Steuerung des Build-Prozesses. Dazu später mehr.

Das erste Programm: Umgebungsvariablen

Einige Konfigurationseinstellungen, z.B. für Datenbanken, sind als Umgebungsvariablen abgelegt.

Daher erstellen wir als erstes einfaches Beispiel keine Hallo-Welt-App, sondern eine nützliche App, welche die Umgebungsvariablen anzeigt.

Das Programm wird unter dem Namen `printenv.pl` im Ordner `perl` abgelegt.

```
#!/usr/bin/perl
use strict;
use warnings;

# printenv -- demo CGI program which
# just prints its environment
print "Content-type: text/plain\n\n";

foreach my $key ( sort( keys(%ENV) ) ) {
    my $val = $ENV{$key};
    $val =~ s|\n|\\n|g;
    $val =~ s|"|\\"|g;
    print qq~$key = $val\n~;
}
exit();
```

Zur Veröffentlichung (Deployment) der App verwendet OpenShift Git. Sobald die Datei hinzugefügt und committed wurde, kann diese per `git push` in die Cloud ausgeliefert werden.

```
$ git add printenv.pl
$ git commit -m 'Umgebungsvariablen App'
$ git push

...
remote: Stopping application...
remote: Waiting for stop to finish
remote: Done
...
```

Das Programm ist nun unter der URL `http://x1-perlhowto.rhc-loud.com/printenv.pl` erreichbar.

Hinweis: Alle Daten innerhalb des Git-Repositories werden dabei auf dem OpenShift Express Server zunächst gelöscht und dann neu eingespielt.

Hinweis: Da Umgebungsvariablen auch sensible Daten enthalten können, sollte dieses Programm nicht auf dem Cloud-Server verbleiben.



Zum Bau und zur Auslieferung der Applikation werden die Programme im Ordner `.openshift/action_hooks/` ausgeführt. Um sich die Umgebungsvariablen anzeigen zu lassen, genügt es in die Datei `build` die Anweisung `export` einzufügen. Dann werden die Umgebungsvariablen bei jedem `push` angezeigt.

```
$ cat .openshift/action_hooks/build

#!/bin/bash
# This is a simple build script and will
# be executed on your CI system if
# available. Otherwise it will execute
# while your application is stopped
# before the deploy step. This script
# gets executed directly, so it
# could be python, php, ruby, etc.
export
```

Eigene Umgebungsvariablen können derzeit nicht gesetzt werden.

Logbuch-Dateien

Das Kommando `rhc-tail-files` ermöglicht den Zugriff auf die Logbuch-Dateien auf dem Cloud-Server.

```
$ rhc-tail-files -a X1
Password:

Attempting to tail files: X1/logs/*
Use ctl + c to stop

==> X1/logs/error_log-... <==
[Date/Time] [notice] ...

==> X1/logs/access_log-... <==
xx.xxx.xxx.IP - - [Date/Time] \
"GET /printenv.pl HTTP/1.0" 200 2323 "-" "
```

Der Zugriff auf die `error_log` Dateien erleichtert das Debuggen erheblich.

Snapshots

Das Kommando `rhc-snapshot` erstellt einen Snapshot der Applikation und liefert diesen als gezippte `tar`-Datei zurück:

```
$ rhc-snapshot -a X1
Password:

Pulling down a snapshot to X1.tar.gz

Stopping application...
Waiting for stop to finish
Done
Creating and sending tar.gz
Starting application...
Done
```

Wenn man die Datei `X1.tar.gz` auspackt, sieht man B<alle> Verzeichnisse und Dateien der Applikation:

```
$ tree
.
|-- git
`-- X1
    |-- ci
    |-- conf
    |   |-- magic -> /etc/httpd/conf/magic
    |-- conf.d
    |-- data
    |-- logs
    |-- modules -> /usr/lib64/httpd/modules
    |-- perl5lib
    |-- repo -> runtime/repo
    |-- run
    |-- runtime
    |   |-- repo
    |       |-- deplist.txt
    |       |-- libs
    |       |-- misc
    |       |-- perl
    |           |-- health_check.pl
    |           |-- index.pl
    |           |-- printenv.pl
    |       |-- README
    |-- tmp
```

Die Ausgabe der Verzeichnisauflistung habe ich für diesen Artikel deutlich gekürzt.

Persistent Storage

Wie oben bereits erwähnt werden bei der Auslieferung per `git push` alle Dateien, die sich innerhalb des Git-Repositories befinden, auf dem Cloud-Server gelöscht und neu eingespielt. Persistente Daten, z.B. SQLite Dateien, müssen daher ausserhalb des Git-Repositories auf dem Server aufbewahrt werden. Dazu stellt OpenShift Express den Ordner `data` zur Verfügung.

Der Pfad zum Ordner `data` kann aus der Umgebungsvariablen `OPENSIFT_DATA_DIR` ermittelt werden.



```
#!/usr/bin/perl
use strict;
use warnings;

print "Content-type: text/plain\n\n";

my $data_dir = $ENV{OPENSIFT_DATA_DIR};
my $file = 'test.txt';

open (OUT, ">" , "$data_dir/$file" )
    or die $!;

for ( 1 .. 10 ) {
    print OUT "$_\n";
}
close(OUT) or die $!;
print "Okay. Datei $file angelegt.";
exit();
```

Die im Ordner *data* gespeicherten Daten lassen sich per *rhc-snapshot* vom Cloud-Server holen.

```
$ tree
.
|-- data
|   |-- test.txt
```

Weitere Cartridges

Applikationen lassen sich mit dem Kommando *rhc-ctl-app* anpassen. Der Parameter *-L* listet alle verfügbaren Ergänzungen auf:

```
$ rhc-ctl-app -a X1 -L

List of supported embedded cartridges:

Obtaining list of cartridges ...
jenkins-client-1.4, mongodb-2.0,
rockmongo-1.1, metrics-0.1, mysql-5.1,
phpmyadmin-3.4
```

Cartridges werden in eine bestehende App eingebettet (-e).

```
$rhc-ctl-app -a X1 -e add-mysql-5.1
Password:

RESULT:

Mysql 5.1 database added.
Please make note of these credentials:

    Root User: admin
    Root Password: .....
    Database Name: X1

Connection URL: mysql://127.1.21.1:3306/

You can manage your new Mysql database by
also embedding phpmyadmin-3.4.
```

Dieses Beispiel erzeugt eine MySQL-Datenbank ohne Tabellen, diese müssen durch ein eigenes `I<create tables>` Programm erzeugt werden.

END }

OpenShift Express ist eigentlich einfach zu benutzen. Dieser Artikel enthält sehr wenig Perl-Code, aber den kann der Leser ja selber schreiben.