Article
Dmitry Maslennikov · Mar 13, 2017 6m read

Containerization Caché

In this article, I am going to give some examples to get your own docker image with InterSystems Caché/Ensemble.

Let ' s start from the beginning, from DockerfileDockerfile is a plaintext configuration file which is used to build a docker image.

I would recommend using centos as a core distributive for the image because InterSystems supports RedHat, and Centos is the most compatible distributive.

```
FROM centos:6
```

You can add your name as an author of this file.

```
MAINTAINER Dmitry Maslennikov <mrdaimor@gmail.com>
```

In the first step, we should install some dependencies, and configure operating systems, as I configured TimeZone here. These dependencies needs for the installation process, and for Caché itself.

```
# update OS + dependencies & run Caché silent instal
RUN yum -y update \
 && yum -y install which tar hostname net-tools wget \
 && yum -y clean all \
 && ln -sf /etc/locatime /usr/share/zoneinfo/Europe/Prague
```

Let ' s define the folder where we will store installation distributive.

```
ENV TMP_INSTALL_DIR=/tmp/distrib
```

Let's set up some arguments with default values. These arguments can be changed during the build process.

```
ARG password="Qwerty@12"
ARG cache=ensemble-2016.2.1.803.0
```

Then we should define some environment variables for silent installation.

```
ENV ISC_PACKAGE_INSTANCENAME="ENSEMBLE" \
    ISC_PACKAGE_INSTALLDIR="/opt/ensemble/" \
    ISC_PACKAGE_UNICODE="Y" \
    ISC_PACKAGE_CLIENT_COMPONENTS="" \
    ISC_PACKAGE_INITIAL_SECURITY="Normal" \
    ISC_PACKAGE_USER_PASSWORD=${password}
```

I decided to set security to the normal level, and I should set some password.

You can look at the documentation to find more options.

```
WORKDIR ${TMP_INSTALL_DIR}
```

Working directory would be used as a current directory for the next commands. If the directory does not exist, it would be created.

```
COPY cache.key $ISC_PACKAGE_INSTALLDIR/mgr/
```

You can include license key file if you are not going to publish this image in public repositories.

Now we should get the installation distributive, and there are several ways to do it:

- Download manually and place this file near to Dockerfile and use this line.

```
ADD $cache-lnxrhx64.tar.gz .
```

This command will copy and extract distributive to our working directory.

- Download file directly from the WRC.

```
RUN wget -qO /dev/null --keep-session-cookies --save-cookies /dev/stdout --post-data=
"UserName=$WRC_USERNAME&Password=$WRC_PASSWORD" 'https://login.intersystems.com/login
/SSO.UI.Login.cls?referrer=https%253A//wrc.intersystems.com/wrc/login.csp' \
 | wget -O - --load-cookies /dev/stdin "https://wrc.intersystems.com/wrc/WRC.StreamSe
rver.cls?FILE=/wrc/distrib/$cache-lnxrhx64.tar.gz" \
 | tar xvfzC - .
```

In this case, we should pass login password for the WRC. And you can add this lines in this file above.

```
ARG WRC_USERNAME=”username”
ARG WRC_PASSWORD=”password”
```

But you should know that in this case, login/password could be extracted from the image. So, it is not the secure way.

- And the preferable way, publish this file on internal FTP/HTTP server in the company.

```
RUN wget -O - "ftp://ftp.company.com/cache/$cache-lnxrhx64.tar.gz" \
 | tar xvfzC - .
```

Now we are ready to install.

```
RUN ./$cache-lnxrhx64/cinstall_silent
```

Once installation is being completed shutdown the instance.

```
RUN ccontrol stop $ISC_PACKAGE_INSTANCENAME quietly
```

But it is not over. We should have some control process In a Docker image and this task could be done by

ccontainermain project made by Luca Ravazzolo. So, download it directly from the github repository.

```
# Caché container main process PID 1 (https://github.com/zrml/ccontainermain)
RUN curl -L https://github.com/zrml/ccontainermain/raw/master/distrib/linux/ccontaine
rmain -o /ccontainermain \
 && chmod +x /ccontainermain
```

Clean up the temporary folder.

```
RUN rm -rf $TMP_INSTALL_DIR
```

> In case if your docker daemon uses overlay driver for storage, we should add this workaround to prevent starting Cache with error <PROTECT>.
>
> ```
> # Workaround for an overlayfs bug which prevents Cache from starting with <PROT
> ECT> errors
> COPY ccontrol-wrapper.sh /usr/bin/
> RUN cd /usr/bin \
>  && rm ccontrol \
>  && mv ccontrol-wrapper.sh ccontrol \
>  && chmod 555 ccontrol
> ```
>
> Where ccontrol-wrapper.sh, should contain
>
> ```
> #!/bin/bash
>
> # Work around a werid overlayfs bug where files don't open properly if they hav
> en't been
> # touched first - see the yum-ovl plugin for a similar workaround
> if [ "${1,,}" == "start" ]; then
>     find $ISC_PACKAGE_INSTALLDIR -name CACHE.DAT -exec touch {} \;
> fi
>
> /usr/local/etc/cachesys/ccontrol $@
> ```
>
> You can use this command to check which driver is using docker.
>
> ```
> docker info --format '{{.Driver}}'
> ```

Here we say that our image exposes two standard for Caché ports 57772 for web and 1972 for binary connections.

```
EXPOSE 57772 1972
```

And finally we should say how to execute our container.

```
ENTRYPOINT ["/ccontainermain", "-cconsole", "-i", "ensemble"]
```

In the end our file should look like this:

```
FROM centos:6

MAINTAINER Dmitry Maslennikov <Dmitry.Maslennikov@csystem.cz>

# update OS + dependencies & run Caché silent instal
RUN yum -y update \
 && yum -y install which tar hostname net-tools wget \
 && yum -y clean all \
 && ln -sf /etc/locatime /usr/share/zoneinfo/Europe/Prague

ARG password="Qwerty@12"
ARG cache=ensemble-2016.2.1.803.0

ENV TMP_INSTALL_DIR=/tmp/distrib

# vars for Caché silent install
ENV ISC_PACKAGE_INSTANCENAME="ENSEMBLE" \
    ISC_PACKAGE_INSTALLDIR="/opt/ensemble/" \
    ISC_PACKAGE_UNICODE="Y" \
    ISC_PACKAGE_CLIENT_COMPONENTS="" \
    ISC_PACKAGE_INITIAL_SECURITY="Normal" \
    ISC_PACKAGE_USER_PASSWORD=${password}

# set-up and install Caché from distrib_tmp dir
WORKDIR ${TMP_INSTALL_DIR}

ADD $cache-lnxrhx64.tar.gz .

# cache distributive
RUN ./$cache-lnxrhx64/cinstall_silent \
 && ccontrol stop $ISC_PACKAGE_INSTANCENAME quietly \
# Caché container main process PID 1 (https://github.com/zrml/ccontainermain)
 && curl -L https://github.com/daimor/ccontainermain/raw/master/distrib/linux/ccontai
nermain -o /ccontainermain \
 && chmod +x /ccontainermain \
 && rm -rf $TMP_INSTALL_DIR

WORKDIR ${ISC_PACKAGE_INSTALLDIR}

# TCP sockets that can be accessed if user wants to (see 'docker run -p' flag)
EXPOSE 57772 1972

ENTRYPOINT ["/ccontainermain", "-cconsole", "-i", "ensemble"]
```

Now we are ready to build this image. Execute the following command in the same folder where you've placed our Dockerfile, execute command

```
docker build -t ensemble-simple .
```

You will see all process of building an image, since downloading source image, to installation Ensemble.

To change default password or build of cache

```
docker build --build-arg password=SuperSecretPassword -t ensemble-simple .
```

```
docker build --build-arg cache=ensemble-2016.2.1.803.1 -t ensemble-simple .
```

And we are ready to run this image, with commad

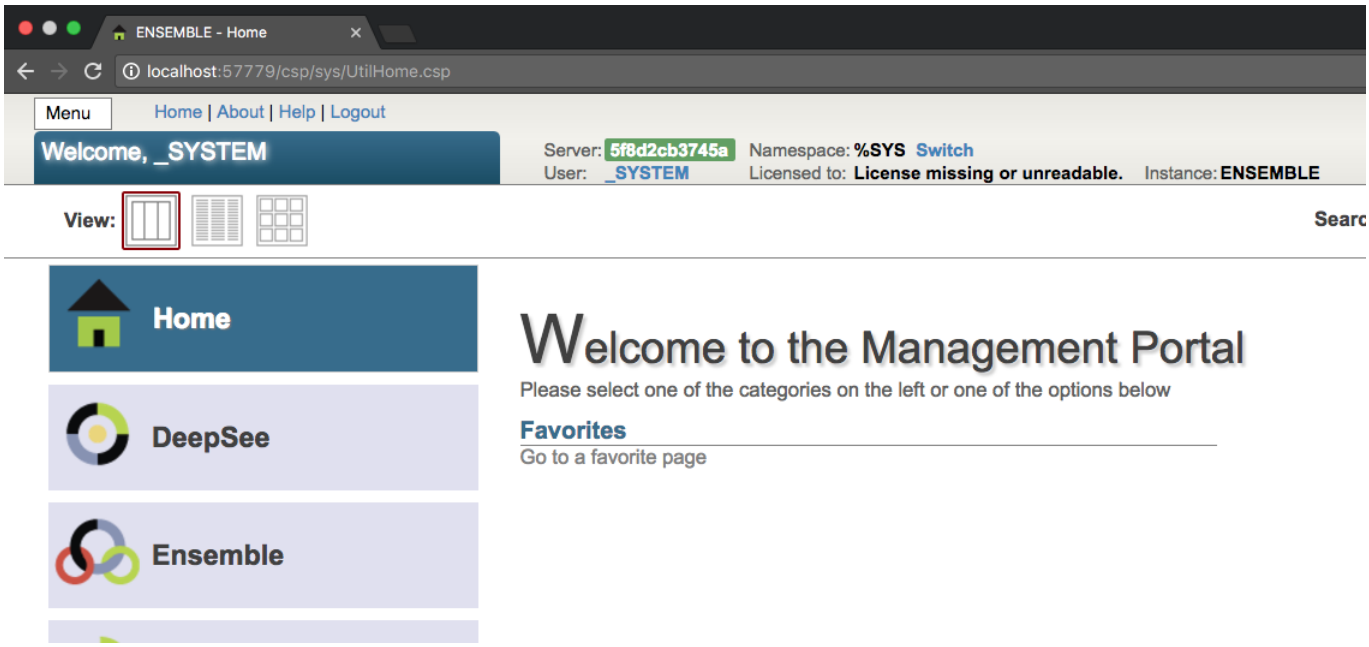
```
docker run -d -p 57779:57772 -p 1979:1972 ensemble-simple
```

Here 57779 and 1979, are the ports which you can use to access to inside our container.

```
docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS                                            NAMES
5f8d2cb3745a        ensemble-simple     "/ccontainermain -..."   18 seconds ago
Up 17 seconds       0.0.0.0:1979->1972/tcp, 0.0.0.0:57779->57772/tcp   keen_carson
```

This command shows all running containers with some details.

You can now open http://localhost:57779/csp/sys/UtilHome.csp. Our new system running and available.



Sources can be found here on GitHub.

UPD: Next part of this article have already available here.

#Cloud #Containerization #DevOps #Docker #System Administration #Caché

Source URL:https://community.intersystems.com/post/containerization-cach%C3%A9