Dotnet Core Documentation

Contents

Setup Dotnet Core on CentOS

```
sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc

sudo sh -c 'echo -e "[packages-microsoft-com-prod]\nname=packages-microsoft-com-prod
\nbaseurl=https://packages.microsoft.com/yumrepos/microsoft-rhel7.3-
prod\nenabled=1\ngpgcheck=1\ngpgkey=https://packages.microsoft.com/keys/microsoft.asc" >
/etc/yum.repos.d/dotnetdev.repo'

sudo yum update

sudo yum install libunwind libicu

sudo yum install dotnet-sdk-2.0.0

export PATH=$PATH:$HOME/dotnet

dotnet --version
```

Setting up Dotnet Core Watch

- Have a directory on your machine which will include the dotnet core project files (where the shell script file will later point to for rebooting the service). (dotnetcore/office.starsupplies.com/Stairsupplies/Stairsupplies) -> This is the folder we need to copy
- Create a configuration file in the root of the created folder for settings and name it: "**stairsupplies.cfg**".

Configuration Settings

Each setting in the "**stairsupplies.cfg**" file should be in the following format on a new line: "**Setting_Name:Value_Here**". Empty lines will be ignored, along with lines starting with "**#**".

MySQL:

- mysql_username
- mysql_password
- mysql_server
- mysql_port
- mysql_db_name

Payeezy:

- gateway_id
- gateway_password
- key_id
- hmac_key
- payeezy_mode
    - "**live**" or "**test**" (use 'live' for submitting transactions live, or 'test' for using a demo account)

Quickbooks

- quickbooks_client_id
    - Go to "**My Apps**" on QB, select app, and click "**Keys**" to retrieve this value.
- quickbooks_client_secret
    - Go to "**My Apps**" on QB, select app, and click "**Keys**" to retrieve this value.
- quickbooks_redirect_uri
    - The redirect URI that is being used for the QB login. This URI must be set on QB as well --- under your "**App > Keys**".
- quickbooks_realm_id
    - This will be required when we use client credentials with OAuth, opposed to requiring a user login. You can find this value by going to your app's dashboard, followed by pressing the following key combination: "**CTRL + ALT + /**". This will open a dialog containing the "**Company ID**".
- quickbooks_mode
    - "**live**" or "**test**".
- In the AppCommon.cfc file, set the global variable: "**`Application.QUICKBOOKS_API_CLIENT_ID`**".

Other:

- log_path
    - The directory path to store all log files, ex: "**C:\StairSupplies**". A sub directory will be created underneath for each of the APIs, along with a directory for exceptions, and failures which will contain the text file logs.

Apache/Forwarding

*****By default, Apache should already be installed.**

Edit the following file:

/etc/httpd/conf/vhosts/office_stairsupplies.conf

Within the 'VirtualHost' tag, input the following:

```
ProxyPass "/ironbaluster/app1" "http://localhost:5000"
ProxyPassReverse "/ironbaluster/app1" "http://localhost:5000"
```

Setting up cron job

- Create a shell script file in a different directory (such as script.sh). Include the following code for the shell script: (/home/peakey/cron/script.sh)

```
#!/bin/bash

SERVICE_PATH="/home/peakey/dotnetcore";
LOG_PATH="/home/peakey/cron/test.txt";
OUTPUT=$(curl "http://localhost:5000/api/hubspot/")
RESPONSE="$?";

if [ $RESPONSE -eq 0 ]; then
        echo 'SUCCESSFULLLLLL';
        echo "$OUTPUT" > $LOG_PATH
elif [ $RESPONSE -eq 7 ]; then
        echo 'Oh no, the connection was refused!';
        cd $SERVICE_PATH
         pkkill dotnet
         dotnet restore
 dotnet watch run &
else
        echo 'Other error';
        echo "$?";
fi
```

- To create a cron job (scheduled task) for your script, navigate to **'/etc'**. Using **'ls'**, you should find a file titled **'cron.hourly'**. Running the following command will create/link to your shell script file:

```
sudo ln -s /home/peakey/cron/script.sh dotnet_watch
```

- To specify a time for the cron job, you can use the command **'crontab -e'**. For running it hourly, you can input the following to this file:

```
* * * * * /home/peakey/cron/script.sh
```

Time examples:

The rest of the line `wget -O - -q -t 1` basically tells the server to request a url, so that the server executes the cron script

Examples

When	Setting
Every 1 minute	`* * * * *`
Every 15 minutes	`*/15 * * * *`
Every 30 minutes	`*/30 * * * *`
Every 1 hour	`0 * * * *`
Every 6 hours	`0 */6 * * *`
Every 12 hours	`0 */12 * * *`
Once a day	`4 0 * * *`
Once a week	`4 0 * * 0`
Once a month	`4 0 1 * *`

Here is a diagram of the general crontab syntax, for illustration:

```
# +---------------- minute (0 - 59)
# |  +------------- hour (0 - 23)
# |  |  +---------- day of month (1 - 31)
# |  |  |  +------- month (1 - 12)
# |  |  |  |  +---- day of week (0 - 6) (Sunday=0)
# |  |  |  |  |
  *  *  *  *  *  command to be executed
```

Thus, the cron command example above means "ping http://www.example.com/cron.php at the zero minute on every hour of every day of every month of every day of the week."

- Finally, navigate to **'/etc/init.d'**. Run the following command as before to create/link to your shell script file:

```
sudo ln -s /home/peakey/cron/script.sh dotnet_watch
```

Startup Dotnet Services from Shellscript

Requirements:

1. Create a directory under the root titled '**nonexistent**'. This will be required for the Nuget configuration, when the shellscript runs the '**dotnet restore**' command.
    a. Be sure to set the user/group permission on this directory to '**nobody:nogroup**' (or whichever one you are using). Example: '***chown nobody:nogroup /nonexistent***'. If not, an error will be received with '**Permission Denied**'.
2. In the dotnet directory (where the source code is), navigate to '**bin/Debug/netcoreapp2.0**'. Set the user/group permission to '**nobody:nogroup**' for all files under this directory. Failure to do so will result in a '**Permission Denied**' error.
3. If you encounter an error like '**Unable to obtain lock file access on '/tmp/NuGetScratch/lock**', navigate to the '**/tmp/NuGetScratch/lock**' directory from the root. Either remove all files under this folder, or set the user/group permission to '**nobody:nogroup**'.

Debugging:

- Use the command '**ps ax | grep dotnet**' to view the running processes for dotnet.
- If you need to kill all dotnet processes, use the command '**killall dotnet**'.
- If everything is set/working properly, there should be roughly 4 dotnet services running after executing the shellscript. The running services should look something like:
    - Dotnet (always runs)
    - Dotnet exec
    - Dotnet watch run
    - Exec –depsfile

Shellscript Example:

```
#!/bin/bash

DOTNET_PROCESS_COUNT=$(ps -C dotnet | wc -l)
SERVICE_PATH="/ironbaluster.dev1/office/html/dotnetcore";
SERVICES_RUNNING_FILE="/ironbaluster.dev1/office/html/dotnetcore/dotnet_service
s_running.txt";
#LOG_PATH="/ironbaluster.dev1/office/html/dotnetcore/test.txt";


# There will always be 1 running
if [ $DOTNET_PROCESS_COUNT -gt 1 ]; then
        echo "Dotnet processes running.";

#       exit 1
else
        echo "Dotnet processes not found, restarting service...";


        cd $SERVICE_PATH
        echo "Changed to dotnetcore path";

        killall dotnet
        #ps ax | grep dotnet | awk '{print $1}' | xargs kill -9 $1
        echo "Stopped dotnet processes";

        export DOTNET_SKIP_FIRST_TIME_EXPERIENCE=true
        export DOTNET_CLI_TELEMETRY_OPTOUT=1

        rm -f $SERVICES_RUNNING_FILE # If exists, delete file created from
dotnet (which is created after services are running)
        dotnet restore
        echo "Started dotnet restore";

        echo "Preparing to execute dotnet watch run command...";
        nohup dotnet watch run > /dev/null 2>&1 &

        while true; do
                if [ -f $SERVICES_RUNNING_FILE ]; then
                        # Services are running
                        echo "STARTED";
                        break
                else
                        # Services are not running yet
                        sleep 2
                fi
        done

        exit
fi
```