SourceCodes

Docker-compose.yml:

# Docker Compose file for ELK stack

# Build with:

# docker-compose build elk

# Run with:

# docker-compose up

elk:

  build: .

  ports:

    - "5601:5601"

    - "9200:9200"

    - "5044:5044"

Elasticsearch-init:

#!/bin/bash

#

# /etc/init.d/elasticsearch -- startup script for Elasticsearch

#

### BEGIN INIT INFO

# Provides:          elasticsearch

# Required-Start:    $network $remote\_fs $named

# Required-Stop:     $network $remote\_fs $named

# Default-Start:     2 3 4 5

# Default-Stop:      0 1 6

# Short-Description: Starts elasticsearch

# Description:       Starts elasticsearch using start-stop-daemon

### END INIT INFO

PATH=/bin:/usr/bin:/sbin:/usr/sbin

NAME=elasticsearch

DESC="Elasticsearch Server"

DEFAULT=/etc/default/$NAME

if [ `id -u` -ne 0 ]; then

    echo "You need root privileges to run this script"

    exit 1

fi

. /lib/lsb/init-functions

if [ -r /etc/default/rcS ]; then

        . /etc/default/rcS

fi

# The following variables can be overwritten in $DEFAULT

# Run Elasticsearch as this user ID and group ID

ES\_USER=elasticsearch

ES\_GROUP=elasticsearch

# Directory where the Elasticsearch binary distribution resides

ES\_HOME=

# Directory containing Java

ES\_JAVA\_HOME=$ES\_HOME/jdk

# Additional Java OPTS

#ES\_JAVA\_OPTS=

# Maximum number of open files

#MAX\_OPEN\_FILES=65536

# Maximum amount of locked memory

#MAX\_LOCKED\_MEMORY=

# Elasticsearch log directory

LOG\_DIR=/var/log/$NAME

# Elasticsearch data directory

DATA\_DIR=/var/lib/$NAME

# Elasticsearch configuration directory

ES\_PATH\_CONF=/etc/$NAME

# Maximum number of VMA (Virtual Memory Areas) a process can own

#MAX\_MAP\_COUNT=262144

# Elasticsearch PID file directory

PID\_DIR="/var/run/elasticsearch"

# End of variables that can be overwritten in $DEFAULT

# overwrite settings from default file

if [ -f "$DEFAULT" ]; then

    . "$DEFAULT"

fi

# Define other required variables

PID\_FILE="$PID\_DIR/$NAME.pid"

DAEMON=$ES\_HOME/bin/elasticsearch

DAEMON\_OPTS="-d -p $PID\_FILE -Epath.logs=$LOG\_DIR -Epath.data=$DATA\_DIR"

DAEMON\_ENV\_VARS="ES\_PATH\_CONF=$ES\_PATH\_CONF"

export ES\_JAVA\_OPTS

export JAVA\_HOME

export ES\_INCLUDE

export ES\_JVM\_OPTIONS

if [ ! -x "$DAEMON" ]; then

    echo "The elasticsearch startup script does not exists or it is not executable, tried: $DAEMON"

    exit 1

fi

checkJava() {

    if [ -x "$JAVA\_HOME/bin/java" ]; then

        JAVA="$JAVA\_HOME/bin/java"

    else

        JAVA=`which java`

    fi

    if [ ! -x "$JAVA" ]; then

        echo "Could not find any executable java binary. Please install java in your PATH or set JAVA\_HOME"

        exit 1

    fi

}

case "$1" in

  start)

        checkJava

        log\_daemon\_msg "Starting $DESC"

        pid=`pidofproc -p $PID\_FILE elasticsearch`

        if [ -n "$pid" ] ; then

            log\_begin\_msg "Already running."

            log\_end\_msg 0

            exit 0

        fi

        # Ensure that the PID\_DIR exists (it is cleaned at OS startup time)

        if [ -n "$PID\_DIR" ] && [ ! -e "$PID\_DIR" ]; then

            mkdir -p "$PID\_DIR" && chown "$ES\_USER":"$ES\_GROUP" "$PID\_DIR"

        fi

        if [ -n "$PID\_FILE" ] && [ ! -e "$PID\_FILE" ]; then

            touch "$PID\_FILE" && chown "$ES\_USER":"$ES\_GROUP" "$PID\_FILE"

        fi

        if [ -n "$MAX\_OPEN\_FILES" ]; then

            ulimit -n $MAX\_OPEN\_FILES

        fi

        if [ -n "$MAX\_LOCKED\_MEMORY" ]; then

            ulimit -l $MAX\_LOCKED\_MEMORY

        fi

        if [ -n "$MAX\_MAP\_COUNT" ]; then

            sysctl -q -w vm.max\_map\_count=$MAX\_MAP\_COUNT

        fi

        # Start Daemon

        start-stop-daemon -d $ES\_HOME --start --user "$ES\_USER" -c "$ES\_USER" --pidfile "$PID\_FILE" \

            --exec /usr/bin/env $DAEMON\_ENV\_VARS $DAEMON -- $DAEMON\_OPTS

        return=$?

        if [ $return -eq 0 ]; then

            i=0

            timeout=10

            # Wait for the process to be properly started before exiting

            until { kill -0 `cat "$PID\_FILE"`; } >/dev/null 2>&1

            do

                sleep 1

                i=$(($i + 1))

                if [ $i -gt $timeout ]; then

                    log\_end\_msg 1

                    exit 1

                fi

                done

        fi

        log\_end\_msg $return

        exit $return

        ;;

  stop)

        log\_daemon\_msg "Stopping $DESC"

        if [ -f "$PID\_FILE" ]; then

            start-stop-daemon --stop --pidfile "$PID\_FILE" \

                --user "$ES\_USER" \

                --quiet \

                --retry TERM/60/KILL/5 > /dev/null

            if [ $? -eq 1 ]; then

                log\_progress\_msg "$DESC is not running but pid file exists, cleaning up"

            elif [ $? -eq 3 ]; then

                PID="`cat $PID\_FILE`"

                log\_failure\_msg "Failed to stop $DESC (pid $PID)"

                exit 1

            fi

            rm -f "$PID\_FILE"

        else

            log\_progress\_msg "(not running)"

        fi

        log\_end\_msg 0

        ;;

  status)

        status\_of\_proc -p $PID\_FILE elasticsearch elasticsearch && exit 0 || exit $?

        ;;

  restart|force-reload)

        if [ -f "$PID\_FILE" ]; then

            $0 stop

        fi

        $0 start

        ;;

  \*)

        log\_success\_msg "Usage: $0 {start|stop|restart|force-reload|status}"

        exit 1

        ;;

esac

exit 0

kibana-init:

#!/bin/sh

#

# /etc/init.d/kibana5\_init -- startup script for kibana5

# bsmith@the408.com 2015-02-20; used elasticsearch init script as template

# https://github.com/akabdog/scripts/edit/master/kibana4\_init

# spujadas 2015-04-09; updated to run as non-root user

#

### BEGIN INIT INFO

# Provides:          kibana5\_init

# Required-Start:    $network $remote\_fs $named

# Required-Stop:     $network $remote\_fs $named

# Default-Start:     2 3 4 5

# Default-Stop:      0 1 6

# Short-Description: Starts kibana5\_init

# Description:       Starts kibana5\_init using start-stop-daemon

### END INIT INFO

KIBANA\_HOME=

KIBANA\_BIN=${KIBANA\_HOME}/bin

NAME=kibana5

PID\_FILE=/var/run/$NAME.pid

PATH=/bin:/usr/bin:/sbin:/usr/sbin:$KIBANA\_BIN

DAEMON=$KIBANA\_BIN/kibana

KIBANA\_USER=kibana

KIBANA\_GROUP=kibana

KIBANA\_LOG\_DIR=/var/log/kibana

KIBANA\_LOG\_FILE="${KIBANA\_LOG\_DIR}/$NAME.log"

DAEMON\_OPTS="-l ${KIBANA\_LOG\_FILE}"

DESC="Kibana5"

NODE\_OPTIONS="--max-old-space-size=4096"

if [ $(id -u) -ne 0 ]; then

  echo "You need root privileges to run this script"

  exit 1

fi

. /lib/lsb/init-functions

if [ -r /etc/default/rcS ]; then

  . /etc/default/rcS

fi

case "$1" in

  start)

    log\_daemon\_msg "Starting $DESC"

    pid=$(pidofproc -p $PID\_FILE kibana)

    if [ -n "$pid" ]; then

      log\_begin\_msg "Already running."

      log\_end\_msg 0

      exit 0

    fi

    touch ${KIBANA\_LOG\_FILE}

    chown ${KIBANA\_USER}:${KIBANA\_GROUP} ${KIBANA\_LOG\_FILE}

    # Start Daemon

    NODE\_OPTIONS="$NODE\_OPTIONS" start-stop-daemon --start --user $KIBANA\_USER -c $KIBANA\_USER \

      --group $KIBANA\_GROUP --pidfile "$PID\_FILE" --make-pidfile \

      --background --exec $DAEMON -- $DAEMON\_OPTS

    log\_end\_msg $?

    ;;

  stop)

    log\_daemon\_msg "Stopping $DESC"

    if [ -f "$PID\_FILE" ]; then

      start-stop-daemon --stop --pidfile "$PID\_FILE" \

        --retry TERM/60/KILL/5 >/dev/null

      if [ $? -eq 1 ]; then

        log\_progress\_msg "$DESC is not running but pid file exists, cleaning up"

      elif [ $? -eq 3 ]; then

        PID="$(cat $PID\_FILE)"

        log\_failure\_msg "Failed to stop $DESC (pid $PID)"

        exit 1

      fi

      rm -f "$PID\_FILE"

    else

      log\_progress\_msg "(not running)"

    fi

    log\_end\_msg 0

    ;;

  status)

    status\_of\_proc -p $PID\_FILE kibana kibana && exit 0 || exit $?

    ;;

  restart|force-reload)

    if [ -f "$PID\_FILE" ]; then

      $0 stop

      sleep 1

    fi

    $0 start

    ;;

  \*)

    log\_success\_msg "Usage: $0 {start|stop|restart|force-reload|status}"

    exit 1

    ;;

esac

exit 0

logstash-init:

#!/bin/sh

# Init script for logstash

# Maintained by Elasticsearch

# Generated by pleaserun.

# Implemented based on LSB Core 3.1:

#   \* Sections: 20.2, 20.3

# spujadas 2015-05-21; emptied LS\_HOME (updated by Dockerfile)

#

### BEGIN INIT INFO

# Provides:          logstash

# Required-Start:    $remote\_fs $syslog

# Required-Stop:     $remote\_fs $syslog

# Default-Start:     2 3 4 5

# Default-Stop:      0 1 6

# Short-Description:

# Description:        Starts Logstash as a daemon.

### END INIT INFO

PATH=/sbin:/usr/sbin:/bin:/usr/bin

export PATH

if [ $(id -u) -ne 0 ]; then

   echo "You need root privileges to run this script"

   exit 1

fi

name=logstash

pidfile="/var/run/$name.pid"

LS\_USER=logstash

LS\_GROUP=logstash

LS\_HOME=

LS\_HEAP\_SIZE="500m"

LS\_JAVA\_OPTS="-Djava.io.tmpdir=${LS\_HOME}"

LS\_LOG\_DIR=/var/log/logstash

LS\_LOG\_FILE="${LS\_LOG\_DIR}/${name}-plain.log"

LS\_OPEN\_FILES=16384

LS\_NICE=19

LS\_OPTS=

[ -r /etc/default/$name ] && . /etc/default/$name

[ -r /etc/sysconfig/$name ] && . /etc/sysconfig/$name

program=/opt/logstash/bin/logstash

args="--path.logs ${LS\_LOG\_DIR} ${LS\_OPTS}"

start() {

  HOME=${LS\_HOME}

  ## removing/updating next lines as overriding JAVA\_OPTS prevents Logstash

  ## from starting

  #JAVA\_OPTS=${LS\_JAVA\_OPTS}

  #export PATH HOME JAVA\_OPTS LS\_HEAP\_SIZE LS\_JAVA\_OPTS LS\_USE\_GC\_LOGGING

  export PATH HOME LS\_HEAP\_SIZE LS\_JAVA\_OPTS LS\_USE\_GC\_LOGGING

  touch ${LS\_LOG\_FILE}

  chown ${LS\_USER}:${LS\_GROUP} ${LS\_LOG\_FILE}

  # set ulimit as (root, presumably) first, before we drop privileges

  ulimit -n ${LS\_OPEN\_FILES}

  # Run the program!

  nice -n ${LS\_NICE} chroot --userspec $LS\_USER:$LS\_GROUP / sh -c "

    cd $LS\_HOME

    ulimit -n ${LS\_OPEN\_FILES}

    exec \"$program\" $args

  " > "${LS\_LOG\_DIR}/$name.stdout" 2> "${LS\_LOG\_DIR}/$name.err" &

  # Generate the pidfile from here. If we instead made the forked process

  # generate it there will be a race condition between the pidfile writing

  # and a process possibly asking for status.

  echo $! > $pidfile

  echo "$name started."

  return 0

}

stop() {

  # Try a few times to kill TERM the program

  if status; then

    pid=$(cat "$pidfile")

    echo "Killing $name (pid $pid) with SIGTERM"

    kill -TERM $pid

    # Wait for it to exit.

    for i in 1 2 3 4 5; do

      echo "Waiting for $name (pid $pid) to die..."

      status || break

      sleep 1

    done

    if status; then

      echo "$name stop failed; still running."

    else

      echo "$name stopped."

      rm -f $pidfile

    fi

  fi

}

status() {

  if [ -f "$pidfile" ] ; then

    pid=$(cat "$pidfile")

    if kill -0 $pid > /dev/null 2> /dev/null; then

      # process by this pid is running.

      # It may not be our pid, but that's what you get with just pidfiles.

      # TODO(sissel): Check if this process seems to be the same as the one we

      # expect. It'd be nice to use flock here, but flock uses fork, not exec,

      # so it makes it quite awkward to use in this case.

      return 0

    else

      return 2 # program is dead but pid file exists

    fi

  else

    return 3 # program is not running

  fi

}

force\_stop() {

  if status; then

    stop

    status && kill -KILL $(cat "$pidfile")

    rm -f $pidfile

  fi

}

case "$1" in

  start)

    status

    code=$?

    if [ $code -eq 0 ]; then

      echo "$name is already running"

    else

      start

      code=$?

    fi

    exit $code

    ;;

  stop) stop ;;

  force-stop) force\_stop ;;

  status)

    status

    code=$?

    if [ $code -eq 0 ]; then

      echo "$name is running"

    else

      echo "$name is not running"

    fi

    exit $code

    ;;

  restart) stop && start ;;

  \*)

    echo "Usage: $SCRIPTNAME {start|stop|force-stop|status|restart}" >&2

    exit 3

    ;;

esac

exit $?