docker-entry-point.sh

#!/bin/bash

set -Eeuo pipefail

if [ "${1:0:1}" = '-' ]; then

set -- mongod "$@"

fi

originalArgOne="$1"

# allow the container to be started with `--user`

# all mongo\* commands should be dropped to the correct user

if [[ "$originalArgOne" == mongo\* ]] && [ "$(id -u)" = '0' ]; then

if [ "$originalArgOne" = 'mongod' ]; then

find /data/configdb /data/db \! -user mongodb -exec chown mongodb '{}' +

fi

# make sure we can write to stdout and stderr as "mongodb"

# (for our "initdb" code later; see "--logpath" below)

chown --dereference mongodb "/proc/$$/fd/1" "/proc/$$/fd/2" || :

# ignore errors thanks to https://github.com/docker-library/mongo/issues/149

exec gosu mongodb "$BASH\_SOURCE" "$@"

fi

# you should use numactl to start your mongod instances, including the config servers, mongos instances, and any clients.

# https://docs.mongodb.com/manual/administration/production-notes/#configuring-numa-on-linux

if [[ "$originalArgOne" == mongo\* ]]; then

numa='numactl --interleave=all'

if $numa true &> /dev/null; then

set -- $numa "$@"

fi

fi

# usage: file\_env VAR [DEFAULT]

# ie: file\_env 'XYZ\_DB\_PASSWORD' 'example'

# (will allow for "$XYZ\_DB\_PASSWORD\_FILE" to fill in the value of

# "$XYZ\_DB\_PASSWORD" from a file, especially for Docker's secrets feature)

file\_env() {

local var="$1"

local fileVar="${var}\_FILE"

local def="${2:-}"

if [ "${!var:-}" ] && [ "${!fileVar:-}" ]; then

echo >&2 "error: both $var and $fileVar are set (but are exclusive)"

exit 1

fi

local val="$def"

if [ "${!var:-}" ]; then

val="${!var}"

elif [ "${!fileVar:-}" ]; then

val="$(< "${!fileVar}")"

fi

export "$var"="$val"

unset "$fileVar"

}

# see https://github.com/docker-library/mongo/issues/147 (mongod is picky about duplicated arguments)

\_mongod\_hack\_have\_arg() {

local checkArg="$1"; shift

local arg

for arg; do

case "$arg" in

"$checkArg"|"$checkArg"=\*)

return 0

;;

esac

done

return 1

}

# \_mongod\_hack\_get\_arg\_val '--some-arg' "$@"

\_mongod\_hack\_get\_arg\_val() {

local checkArg="$1"; shift

while [ "$#" -gt 0 ]; do

local arg="$1"; shift

case "$arg" in

"$checkArg")

echo "$1"

return 0

;;

"$checkArg"=\*)

echo "${arg#$checkArg=}"

return 0

;;

esac

done

return 1

}

declare -a mongodHackedArgs

# \_mongod\_hack\_ensure\_arg '--some-arg' "$@"

# set -- "${mongodHackedArgs[@]}"

\_mongod\_hack\_ensure\_arg() {

local ensureArg="$1"; shift

mongodHackedArgs=( "$@" )

if ! \_mongod\_hack\_have\_arg "$ensureArg" "$@"; then

mongodHackedArgs+=( "$ensureArg" )

fi

}

# \_mongod\_hack\_ensure\_no\_arg '--some-unwanted-arg' "$@"

# set -- "${mongodHackedArgs[@]}"

\_mongod\_hack\_ensure\_no\_arg() {

local ensureNoArg="$1"; shift

mongodHackedArgs=()

while [ "$#" -gt 0 ]; do

local arg="$1"; shift

if [ "$arg" = "$ensureNoArg" ]; then

continue

fi

mongodHackedArgs+=( "$arg" )

done

}

# \_mongod\_hack\_ensure\_no\_arg '--some-unwanted-arg' "$@"

# set -- "${mongodHackedArgs[@]}"

\_mongod\_hack\_ensure\_no\_arg\_val() {

local ensureNoArg="$1"; shift

mongodHackedArgs=()

while [ "$#" -gt 0 ]; do

local arg="$1"; shift

case "$arg" in

"$ensureNoArg")

shift # also skip the value

continue

;;

"$ensureNoArg"=\*)

# value is already included

continue

;;

esac

mongodHackedArgs+=( "$arg" )

done

}

# \_mongod\_hack\_ensure\_arg\_val '--some-arg' 'some-val' "$@"

# set -- "${mongodHackedArgs[@]}"

\_mongod\_hack\_ensure\_arg\_val() {

local ensureArg="$1"; shift

local ensureVal="$1"; shift

\_mongod\_hack\_ensure\_no\_arg\_val "$ensureArg" "$@"

mongodHackedArgs+=( "$ensureArg" "$ensureVal" )

}

# \_js\_escape 'some "string" value'

\_js\_escape() {

jq --null-input --arg 'str' "$1" '$str'

}

jsonConfigFile="${TMPDIR:-/tmp}/docker-entrypoint-config.json"

tempConfigFile="${TMPDIR:-/tmp}/docker-entrypoint-temp-config.json"

\_parse\_config() {

if [ -s "$tempConfigFile" ]; then

return 0

fi

local configPath

if configPath="$(\_mongod\_hack\_get\_arg\_val --config "$@")"; then

# if --config is specified, parse it into a JSON file so we can remove a few problematic keys (especially SSL-related keys)

# see https://docs.mongodb.com/manual/reference/configuration-options/

mongo --norc --nodb --quiet --eval "load('/js-yaml.js'); printjson(jsyaml.load(cat($(\_js\_escape "$configPath"))))" > "$jsonConfigFile"

jq 'del(.systemLog, .processManagement, .net, .security)' "$jsonConfigFile" > "$tempConfigFile"

return 0

fi

return 1

}

dbPath=

\_dbPath() {

if [ -n "$dbPath" ]; then

echo "$dbPath"

return

fi

if ! dbPath="$(\_mongod\_hack\_get\_arg\_val --dbpath "$@")"; then

if \_parse\_config "$@"; then

dbPath="$(jq -r '.storage.dbPath // empty' "$jsonConfigFile")"

fi

fi

: "${dbPath:=/data/db}"

echo "$dbPath"

}

if [ "$originalArgOne" = 'mongod' ]; then

file\_env 'MONGO\_INITDB\_ROOT\_USERNAME'

file\_env 'MONGO\_INITDB\_ROOT\_PASSWORD'

# pre-check a few factors to see if it's even worth bothering with initdb

shouldPerformInitdb=

if [ "$MONGO\_INITDB\_ROOT\_USERNAME" ] && [ "$MONGO\_INITDB\_ROOT\_PASSWORD" ]; then

# if we have a username/password, let's set "--auth"

\_mongod\_hack\_ensure\_arg '--auth' "$@"

set -- "${mongodHackedArgs[@]}"

shouldPerformInitdb='true'

elif [ "$MONGO\_INITDB\_ROOT\_USERNAME" ] || [ "$MONGO\_INITDB\_ROOT\_PASSWORD" ]; then

cat >&2 <<-'EOF'

error: missing 'MONGO\_INITDB\_ROOT\_USERNAME' or 'MONGO\_INITDB\_ROOT\_PASSWORD'

both must be specified for a user to be created

EOF

exit 1

fi

if [ -z "$shouldPerformInitdb" ]; then

# if we've got any /docker-entrypoint-initdb.d/\* files to parse later, we should initdb

for f in /docker-entrypoint-initdb.d/\*; do

case "$f" in

\*.sh|\*.js) # this should match the set of files we check for below

shouldPerformInitdb="$f"

break

;;

esac

done

fi

# check for a few known paths (to determine whether we've already initialized and should thus skip our initdb scripts)

if [ -n "$shouldPerformInitdb" ]; then

dbPath="$(\_dbPath "$@")"

for path in \

"$dbPath/WiredTiger" \

"$dbPath/journal" \

"$dbPath/local.0" \

"$dbPath/storage.bson" \

; do

if [ -e "$path" ]; then

shouldPerformInitdb=

break

fi

done

fi

if [ -n "$shouldPerformInitdb" ]; then

mongodHackedArgs=( "$@" )

if \_parse\_config "$@"; then

\_mongod\_hack\_ensure\_arg\_val --config "$tempConfigFile" "${mongodHackedArgs[@]}"

fi

\_mongod\_hack\_ensure\_arg\_val --bind\_ip 127.0.0.1 "${mongodHackedArgs[@]}"

\_mongod\_hack\_ensure\_arg\_val --port 27017 "${mongodHackedArgs[@]}"

# remove "--auth" and "--replSet" for our initial startup (see https://docs.mongodb.com/manual/tutorial/enable-authentication/#start-mongodb-without-access-control)

# https://github.com/docker-library/mongo/issues/211

\_mongod\_hack\_ensure\_no\_arg --auth "${mongodHackedArgs[@]}"

if [ "$MONGO\_INITDB\_ROOT\_USERNAME" ] && [ "$MONGO\_INITDB\_ROOT\_PASSWORD" ]; then

\_mongod\_hack\_ensure\_no\_arg\_val --replSet "${mongodHackedArgs[@]}"

fi

sslMode="$(\_mongod\_hack\_have\_arg '--sslPEMKeyFile' "$@" && echo 'allowSSL' || echo 'disabled')" # "BadValue: need sslPEMKeyFile when SSL is enabled" vs "BadValue: need to enable SSL via the sslMode flag when using SSL configuration parameters"

\_mongod\_hack\_ensure\_arg\_val --sslMode "$sslMode" "${mongodHackedArgs[@]}"

if stat "/proc/$$/fd/1" > /dev/null && [ -w "/proc/$$/fd/1" ]; then

# https://github.com/mongodb/mongo/blob/38c0eb538d0fd390c6cb9ce9ae9894153f6e8ef5/src/mongo/db/initialize\_server\_global\_state.cpp#L237-L251

# https://github.com/docker-library/mongo/issues/164#issuecomment-293965668

\_mongod\_hack\_ensure\_arg\_val --logpath "/proc/$$/fd/1" "${mongodHackedArgs[@]}"

else

initdbLogPath="$(\_dbPath "$@")/docker-initdb.log"

echo >&2 "warning: initdb logs cannot write to '/proc/$$/fd/1', so they are in '$initdbLogPath' instead"

\_mongod\_hack\_ensure\_arg\_val --logpath "$initdbLogPath" "${mongodHackedArgs[@]}"

fi

\_mongod\_hack\_ensure\_arg --logappend "${mongodHackedArgs[@]}"

pidfile="${TMPDIR:-/tmp}/docker-entrypoint-temp-mongod.pid"

rm -f "$pidfile"

\_mongod\_hack\_ensure\_arg\_val --pidfilepath "$pidfile" "${mongodHackedArgs[@]}"

"${mongodHackedArgs[@]}" --fork

mongo=( mongo --host 127.0.0.1 --port 27017 --quiet )

# check to see that our "mongod" actually did start up (catches "--help", "--version", MongoDB 3.2 being silly, slow prealloc, etc)

# https://jira.mongodb.org/browse/SERVER-16292

tries=30

while true; do

if ! { [ -s "$pidfile" ] && ps "$(< "$pidfile")" &> /dev/null; }; then

# bail ASAP if "mongod" isn't even running

echo >&2

echo >&2 "error: $originalArgOne does not appear to have stayed running -- perhaps it had an error?"

echo >&2

exit 1

fi

if "${mongo[@]}" 'admin' --eval 'quit(0)' &> /dev/null; then

# success!

break

fi

(( tries-- ))

if [ "$tries" -le 0 ]; then

echo >&2

echo >&2 "error: $originalArgOne does not appear to have accepted connections quickly enough -- perhaps it had an error?"

echo >&2

exit 1

fi

sleep 1

done

if [ "$MONGO\_INITDB\_ROOT\_USERNAME" ] && [ "$MONGO\_INITDB\_ROOT\_PASSWORD" ]; then

rootAuthDatabase='admin'

"${mongo[@]}" "$rootAuthDatabase" <<-EOJS

db.createUser({

user: $(\_js\_escape "$MONGO\_INITDB\_ROOT\_USERNAME"),

pwd: $(\_js\_escape "$MONGO\_INITDB\_ROOT\_PASSWORD"),

roles: [ { role: 'root', db: $(\_js\_escape "$rootAuthDatabase") } ]

})

EOJS

fi

export MONGO\_INITDB\_DATABASE="${MONGO\_INITDB\_DATABASE:-test}"

echo

for f in /docker-entrypoint-initdb.d/\*; do

case "$f" in

\*.sh) echo "$0: running $f"; . "$f" ;;

\*.js) echo "$0: running $f"; "${mongo[@]}" "$MONGO\_INITDB\_DATABASE" "$f"; echo ;;

\*) echo "$0: ignoring $f" ;;

esac

echo

done

"${mongodHackedArgs[@]}" --shutdown

rm -f "$pidfile"

echo

echo 'MongoDB init process complete; ready for start up.'

echo

fi

unset "${!MONGO\_INITDB\_@}"

fi

rm -f "$jsonConfigFile" "$tempConfigFile"

exec "$@"