Public docker host setup

Maxim Zaslavsky

3/5/16

Docker host setup

Overview

this server will have:

- nginx on 80 (public)
- shipyard on 8080 (3100 locally from ssh config)
- docker registry ui (for v2) on 4100 (and 4100 locally from ssh config). because shipyard doesn't support v2 registry as of 10-16-15
- docker registry (v2) on 5000 (not port forwarded)
- backups live at `~/backups`
- a shared directory `~/data/shared` can be loaded into containers

Begin setup

```
# install zsh shell
sudo apt-get install -y zsh git-core
wget
    https://github.com/robbyrussell/oh-my-zsh/raw/master/tools/install.sh
    -O - | zsh
chsh -s `which zsh`

# install docker
# https://docs.docker.com/installation/ubuntulinux/
sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80
    --recv-keys 58118E89F3A912897C070ADBF76221572C52609D
echo deb https://apt.dockerproject.org/repo ubuntu-trusty main >
    /etc/apt/sources.list.d/docker.list
sudo apt-get update
sudo apt-get install -y linux-image-generic-lts-trusty
    linux-headers-generic-lts-trusty
sudo apt-get update
sudo apt-get install -y docker-engine

# set up docker
sudo docker run hello-world
sudo usermod -aG docker maxim # replace with your username
# logout
# log back in
docker run hello-world

# install nginx reverse proxy
sudo apt-get install nginx
sudo apt-get install dos2unix -y

# also, install the zsh docker plugin:
    https://github.com/robbyrussell/oh-my-zsh/wiki/Plugins#docker
# edit ~/.zshrc to include docker in plugins line, which can look
    like: plugins=(rails git ruby)


# set up some directories
mkdir ~/data
mkdir ~/images
echo 'docker stats $(docker ps -q)' > stats.sh
chmod +x *.sh


# set up docker registry in case we want to have it later
docker run -d -p 5000:5000 --restart=always --name registry registry;
# a UI for docker registry
docker run -d -e ENV_DOCKER_REGISTRY_HOST=localhost -e
    ENV_DOCKER_REGISTRY_PORT=5000 --restart=always --name
    registry-frontend -p 4100:80
    konradkleine/docker-registry-frontend:v2;
```

An important change to docker config for DNS settings

See http://stackoverflow.com/a/24991137/130164:

Specifically: uncomment `DOCKER_OPTS` line in `/etc/default/docker`. then restart docker using `sudo service docker restart`.

Now we have to set up Postfix mail server interactively:

```
# set up postfix
sudo apt-get install postfix
```

Choose "Internet site"

Add to `/etc/postfix/main.cf` to set up sendgrid relay:

```
# for sendgrid
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = static:SENDGRIDUSERNAME:SENDGRIDPASSWORD
smtp_sasl_security_options = noanonymous
smtp_tls_security_level = encrypt
header_size_limit = 4096000
relayhost = [smtp.sendgrid.net]:2525

default_transport = smtp

# http://www.binarytides.com/postfix-mail-forwarding-debian/
# replace with your server dns
virtual_alias_domains = myserverdns.cloudapp.net cloudapp.net
virtual_alias_maps = hash:/etc/postfix/virtual
```

also, comment out the previous relayhost line

also set in that file:

```
inet_interfaces = loopback-only
myorigin: myserverdns.cloudapp.net
```

then, set up aliases for mail delivery (replace with your favorite email):

```
echo "root:     maxim@maximz.com" >>> /etc/aliases
echo "maxim:    maxim@maximz.com" >>> /etc/aliases
```

some final steps:

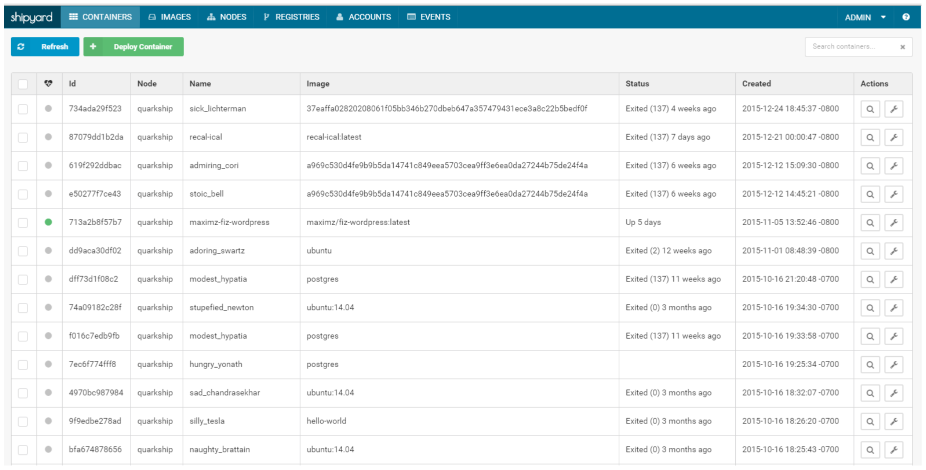
```
sudo echo "@myserverdns.cloudapp.net maxim@maximz.com" >
    /etc/postfix/virtual
sudo postmap /etc/postfix/virtual
sudo service postfix restart
sudo postfix reload
```

test it:

```
sudo apt-get install -y mailutils
echo "my message" | mail -s "test subject" maxim@maximz.com
```

check `/var/log/mail.log` for errors

Setup shipyard (docker dashboard)



		Id	Node	Name	Image	Status	Created	Actions
☐	●	734ada29f523	quarkship	sick_lichterman	37eaffa02820208061f05bb346b270dbeb647a357479431ece3a8c22b5bedf0f	Exited (137) 4 weeks ago	2015-12-24 18:45:37 -0800	
☐	●	87079dd1b2da	quarkship	recal-ical	recal-ical:latest	Exited (137) 7 days ago	2015-12-21 00:00:47 -0800	
☐	●	619f292ddbac	quarkship	admiring_cori	a969c530d4fe9b9b5da14741c849eea5703cea9ff3e6ea0da27244b75de24f4a	Exited (137) 6 weeks ago	2015-12-12 15:09:30 -0800	
☐	●	e50277f7ce43	quarkship	stoic_bell	a969c530d4fe9b9b5da14741c849eea5703cea9ff3e6ea0da27244b75de24f4a	Exited (137) 6 weeks ago	2015-12-12 14:45:21 -0800	
☐	🟢	713a2b8f57b7	quarkship	maximz-fiz-wordpress	maximz/fiz-wordpress:latest	Up 5 days	2015-11-05 13:52:46 -0800	
☐	●	dd9aca30df02	quarkship	adoring_swartz	ubuntu	Exited (2) 12 weeks ago	2015-11-01 08:48:39 -0800	
☐	●	dff73d1f08c2	quarkship	modest_hypatia	postgres	Exited (137) 11 weeks ago	2015-10-16 21:20:48 -0700	
☐	●	74a09182c28f	quarkship	stupefied_newton	ubuntu:14.04	Exited (0) 3 months ago	2015-10-16 19:34:30 -0700	
☐	●	f016c7edb9fb	quarkship	modest_hypatia	postgres	Exited (137) 11 weeks ago	2015-10-16 19:33:58 -0700	
☐	●	7ec6f774fff8	quarkship	hungry_yonath	postgres		2015-10-16 19:25:34 -0700	
☐	●	4970bc987984	quarkship	sad_chandrasekhar	ubuntu:14.04	Exited (0) 3 months ago	2015-10-16 18:32:07 -0700	
☐	●	9f9edbe278ad	quarkship	silly_tesla	hello-world	Exited (0) 3 months ago	2015-10-16 18:26:20 -0700	
☐	●	bfa674878656	quarkship	naughty_brattain	ubuntu:14.04	Exited (0) 3 months ago	2015-10-16 18:25:43 -0700	

Figure 1: Shipyard screenshot

```
curl -sSL https://shipyard-project.com/deploy | bash -s
```

Shipyard will run on 8080. So in your local ssh config, add something like: `LocalForward 3100 localhost:8080`. That way, you can access at `localhost:3100` when you SSH to your server.

Now, go to Shipyard and login with: `admin / shipyard`

to change the password using a shipyard cli running in a separate docker container:

```
docker run -it --rm --link shipyard-controller:shipyard
    shipyard/shipyard-cli
> echo "For login URL use: http://$SHIPYARD_PORT_8080_TCP_ADDR:8080"
> shipyard login
> > # paste in the the login url
> > admin / shipyard
> shipyard change-password # then follow prompt
> # Ctrl-D to exit docker container
```

autobackup setup (using maxim's script)

See script docs to understand how this script works. Currently calls an uploader script that uploads to windows azure. Easy to change to S3 – will do.

```
git clone https://github.com/maximz/autobackup.git
cd autobackup
echo 'placeholder' > .identity
nano /home/maxim/paths_to_backup.txt
mkdir $HOME"/files_backup/"
echo 'placeholder file -- where the snapshots live' >
    $HOME"/files_backup/.identity"
mkdir $HOME"/files_backup_tars/"
mkdir $HOME"/logs"

echo "export AZURE_STORAGE_ACCOUNT=storageaccountname" > .profile
echo "export AZURE_STORAGE_ACCESS_KEY='key'" > .profile

sudo visudo
# put in:
Defaults env_keep += "AZURE_STORAGE_ACCOUNT AZURE_STORAGE_ACCESS_KEY"

# install python and requirements
sudo apt-get update && sudo apt-get install -y python python-dev
    python-pip python-virtualenv; # if you don't have python and pip
    yet
sudo pip install python-dateutil;
sudo pip install azure;

cd autobackup
```

Now put some backup scripts in `do_backups.sh`. Here's my current version of that file:

```
#!/bin/bash

# Makes docker backups
# Make sure not to have -it in these commands below.

# Here's an example for a wordpress container I host that exposes a
    couple of volumes.
# maximz-mywebsite-wordpress

docker run --rm --volumes-from maximz-mywebsite-wordpress -v
    ~/backups:/backup ubuntu:14.04 sh -c 'mkdir -p /backup/mywebsite
    && tar zcvf /backup/mywebsite/mysql.tar.gz /var/lib/mysql && tar
    zcvf /backup/mywebsite/www.tar.gz /usr/share/nginx/www && tar
    zcvf /backup/mywebsite/log.tar.gz /var/log && tar zcvf
    /backup/mywebsite/admin.tar.gz /admin';

# maximz-tracker
```

```
docker run --rm --volumes-from maximz-tracker -v ~/backups:/backup
    ubuntu:14.04
sh -c 'mkdir -p /backup/tracker && tar zcvf
    /backup/tracker/web.tar.gz /home/docker/code/app && tar zcvf
    /backup/tracker/log.tar.gz /var/log';
```

continue:

```
chmod +x do_backups.sh
chmod +x autobackup.sh
```

then modify autobackup.sh to have the following at the top:

```
echo "preparing for backup ... dumping docker data"
sudo ./do_backups.sh # the sudo is critical here
```

setup backup paths: `nano ~/paths_to_backup.txt`. put in:

```
/home/maxim/backups;docker backups
/home/maxim/images;dockerfiles
```

test it: `./autobackup.sh`

Add to crontab: `crontab -e`. Put in at top: `MAILTO=maxim@maximz.com`. Put in at bottom:

```
@daily . ~/.profile; cd ~/autobackup; sudo ./autobackup.sh
@reboot echo 'Reboot' | ~/slackpost.sh 2>&1 >/dev/null
@daily sh ~/diskspace.sh 2>&1 >/dev/null
0 * * * * sh ~/docker_events.sh 2>&1 >/dev/null
#* * * * * env > ~/cronenv
* * * * * sudo ~/keep_processes_up.sh
@hourly sh ~/patch_sys.sh 2>&1 >/dev/null
```

Some other scripts

Install slack poster

```
git clone https://github.com/course-hero/slacktee.git
sudo bash ./slacktee/install.sh # also launches interactive setup
sudo cp ~/.slacktee /etc/slacktee.conf
which slacktee.sh
echo 'hi' | slacktee.sh
echo 'hi2'  | slacktee.sh -a "danger" -e "Date and Time" "$(date)"
    -s "Host" "$(hostname)"
```

I chose the following settings (which got stored in `~/.slacktee`):

```
webhook_url="https://hooks.slack.com/services/MYWEBHOOK"
upload_token=""
tmp_dir="/tmp"
channel="webhooks"
username="slacktee"
icon="ghost"
attachment=""
```

patch_sys.sh

```
#!/bin/bash
sudo apt-get update -y        # Fetches the list of available updates
sudo apt-get upgrade -y       # Strictly upgrades the current
    packages
sudo apt-get dist-upgrade -y  # Installs updates (new ones)
```

keep_processes_up.sh

```
#!/bin/sh
# keep core services running
sudo ./keep_service_up.sh nginx
sudo ./keep_service_up.sh docker
sudo ./keep_service_up.sh monit
```

keep_service_up.sh

```
#!/bin/sh
# test if a service is up, else relaunch it
# run with sudo

service_name="$1" # store the argument

if P=$(pgrep $service_name)
then
    exit 0; #echo "$SERVICE is running, PID is $P"
else
    /etc/init.d/"$service_name" start > /dev/null; #echo "$SERVICE
        is not running"
fi
```

docker_events.sh

```
rm /tmp/dockerevents
timeout 1 docker events --since='1h' > /tmp/dockerevents
cat /tmp/dockerevents | ~/slackpost.sh
```

diskspace.sh

```sh
#!/bin/sh
ADMIN="maxim@maximz.com"
THRESHOLD=85

df -PkH | grep -vE '^Filesystem|tmpfs|cdrom|media' | awk '{ print $5
    " " $6 }' | while read output;
do
  usep=$(echo $output | awk '{ print $1}' | cut -d'%' -f1 )
  partition=$(echo $output | awk '{print $2}' )
  if [ $usep -ge $THRESHOLD ]; then
    echo "Running out of space \"$partition ($usep%)\" on
        $(hostname) as on $(date)" | ~/slackpost.sh |
    mail -s "Alert: Almost out of disk space $usep%" $ADMIN
  fi
done
```

slackpost.sh

```sh
/usr/local/bin/slacktee.sh -a "danger" -e "Date and Time" "$(date)"
    -s "Host" "$(hostname)"
```

monitalert.sh

```sh
#!/bin/sh
echo "$MONIT_SERVICE - $MONIT_DESCRIPTION" | /home/maxim/slackpost.sh
```

monit_docker_restart.sh

```sh
#!/bin/sh
/home/maxim/monitalert.sh;
docker restart $1 && echo "restarted $1" | /home/maxim/slackpost.sh;
```

usage.sh

```sh
free -m | awk 'NR==2{printf "Memory Usage: %s/%sMB (%.2f%%)\n",
    $3,$2,$3*100/$2
}'
df -h | awk '$NF=="/"{printf "Disk Usage: %d/%dGB (%s)\n", $3,$2,$5}'
top -bn1 | grep load | awk '{printf "CPU Load: %.2f\n", $(NF-2)}'
```

Make sure to run a `chmod +x *.sh`.

monit for monitoring

the goal: extra disk space monitoring, and automatic monitoring of docker container status. if it detects a non-200 status code, it restarts the container. notifications go into a slack channel.

config is in /etc/monit/monitrc. alerts call ~/monitalert.sh. Note that monit will not let you chain commands in an `exec`; the solution is to wrap them in a shell script and call that instead.

install monit:

```
# sudo apt-get install monit # old version, don't install

# get monit 5.15 linux-x64 targz from internet

tar zxvf monit-5.15-linux-x64.tar.gz
cd monit-5.15/
sudo service monit stop
sudo cp bin/monit /usr/bin/monit
sudo chmod 0700 /etc/monit/monitrc
sudo ln -s /etc/monit/monitrc /etc/monitrc
sudo rm /etc/init.d/monit
sudo wget https://gist.githubusercontent.com/rahul286/9975061/raw/1aa107e62ecaaa2dacfdb61a12f13efb6f15 -P /etc/init.d/
sudo chmod u+x /etc/init.d/monit
sudo monit -t
sudo service monit start
sudo monit reload
```

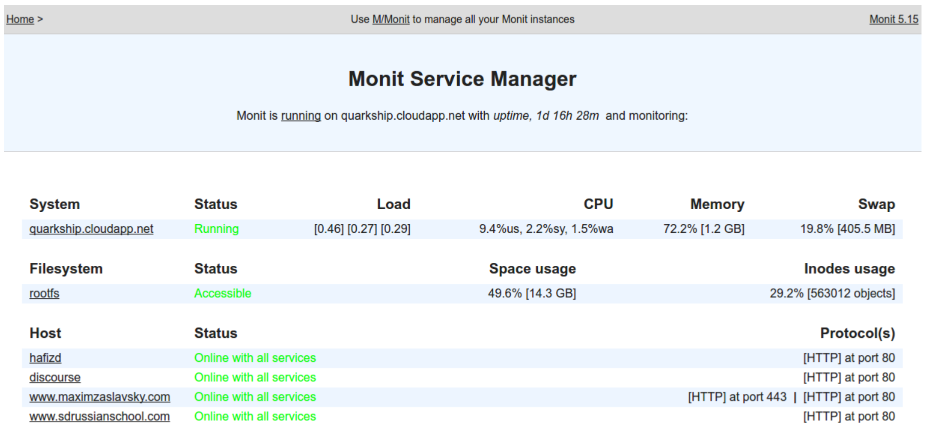
verify control file with: `sudo monit -t`. launch with `sudo monit`. reload with `sudo monit reload`. monit config examples: https://mmonit.com/monit/

monit control panel tunnelled to localhost:2812 through ssh config. the control panel looks like this:

debug monit exec like this: `then exec "/bin/bash -c '/home/maxim/monitalert.sh &>/tmp/myscript.out'"`. another way to debug: `/var/log/monit.log`

my `/etc/monitrc` (see end specifically):

```
###############################################################################
## Monit control file
###############################################################################
##
## Comments begin with a '#' and extend through the end of the line. Keywords
```


Home > Use M/Monit to manage all your Monit instances Monit 5.15

Monit Service Manager

Monit is running on quarkship.cloudapp.net with *uptime, 1d 16h 28m* and monitoring:

System	Status	Load	CPU	Memory	Swap
quarkship.cloudapp.net	Running	[0.46] [0.27] [0.29]	9.4%us, 2.2%sy, 1.5%wa	72.2% [1.2 GB]	19.8% [405.5 MB]

Filesystem	Status	Space usage	Inodes usage
rootfs	Accessible	49.6% [14.3 GB]	29.2% [563012 objects]

Host	Status	Protocol(s)
hafizd	Online with all services	[HTTP] at port 80
discourse	Online with all services	[HTTP] at port 80
www.maximzaslavsky.com	Online with all services	[HTTP] at port 443 | [HTTP] at port 80
www.sdrussianschool.com	Online with all services	[HTTP] at port 80

Figure 2: monit control panel

```
## are case insensitive. All path's MUST BE FULLY QUALIFIED, starting with '/'.
##
## Below you will find examples of some frequently used statements. For
## information about the control file and a complete list of statements and
## options, please have a look in the Monit manual.
##
##
###############################################################################
## Global section
###############################################################################
##
## Start Monit in the background (run as a daemon):
#
   set daemon 120            # check services at 2-minute intervals
   with start delay 240     # optional: delay the first check by 4-minutes (by
#                            # default Monit check immediately after Monit start)
#
#
## Set syslog logging with the 'daemon' facility. If the FACILITY option is
## omitted, Monit will use 'user' facility by default. If you want to log to
## a standalone log file instead, specify the full path to the log
```

```
    file
#
# set logfile syslog facility log_daemon
  set logfile /var/log/monit.log
#
#
## Set the location of the Monit id file which stores the unique id
    for the
## Monit instance. The id is generated and stored on first Monit
    start. By
## default the file is placed in $HOME/.monit.id.
#
# set idfile /var/.monit.id
  set idfile /var/lib/monit/id
#
## Set the location of the Monit state file which saves monitoring
    states
## on each cycle. By default the file is placed in
    $HOME/.monit.state. If
## the state file is stored on a persistent filesystem, Monit will
    recover
## the monitoring state across reboots. If it is on temporary
    filesystem, the
## state will be lost on reboot which may be convenient in some
    situations.
#
  set statefile /var/lib/monit/state
#
## Set the list of mail servers for alert delivery. Multiple servers
    may be
## specified using a comma separator. If the first mail server
    fails, Monit
# will use the second mail server in the list and so on. By default
    Monit uses
# port 25 - it is possible to override this with the PORT option.
#
# set mailserver mail.bar.baz,               # primary mailserver
#                backup.bar.baz port 10025,  # backup mailserver on
    port 10025
#                localhost                   # fallback relay
#
#
## By default Monit will drop alert events if no mail servers are
    available.
## If you want to keep the alerts for later delivery retry, you can
    use the
```

```
## EVENTQUEUE statement. The base directory where undelivered alerts will be
## stored is specified by the BASEDIR option. You can limit the maximal queue
## size using the SLOTS option (if omitted, the queue is limited by space
## available in the back end filesystem).
#
  set eventqueue
      basedir /var/lib/monit/events # set the base directory where events will be stored
      slots 100                     # optionally limit the queue size
#
#
## Send status and events to M/Monit (for more informations about M/Monit
## see http://mmonit.com/). By default Monit registers credentials with
## M/Monit so M/Monit can smoothly communicate back to Monit and you don't
## have to register Monit credentials manually in M/Monit. It is possible to
## disable credential registration using the commented out option below.
## Though, if safety is a concern we recommend instead using https when
## communicating with M/Monit and send credentials encrypted.
#
# set mmonit http://monit:monit@192.168.1.10:8080/collector
#     # and register without credentials     # Don't register credentials
#
#
## Monit by default uses the following format for alerts if the the mail-format
## statement is missing::
## --8<--
## set mail-format {
##      from: monit@$HOST
##   subject: monit alert --  $EVENT $SERVICE
##   message: $EVENT Service $SERVICE
##                 Date:        $DATE
##                 Action:      $ACTION
##                 Host:        $HOST
##                 Description: $DESCRIPTION
##
##            Your faithful employee,
##            Monit
## }
## --8<--
##
## You can override this message format or parts of it, such as subject
## or sender using the MAIL-FORMAT statement. Macros such as $DATE, etc.
## are expanded at runtime. For example, to override the sender, use:
#
# set mail-format { from: monit@foo.bar }
#
#
## You can set alert recipients whom will receive alerts if/when a
## service defined in this file has errors. Alerts may be restricted on
## events by using a filter as in the second example below.
#
# set alert sysadm@foo.bar                       # receive all alerts
## Do not alert when Monit start,stop or perform a user initiated action
# set alert manager@foo.bar not on { instance, action }
#
#
## Monit has an embedded web server which can be used to view status of
## services monitored and manage services from a web interface. See the
## Monit Wiki if you want to enable SSL for the web server.
#
# set httpd port 2812 and
#    use address localhost  # only accept connection from localhost
#    allow localhost        # allow localhost to connect to the server and
#    allow admin:monit      # require user 'admin' with password 'monit'
#    allow @monit           # allow users of group 'monit' to connect (rw)
#    allow @users readonly  # allow users of group 'users' to connect readonly
#
###############################################################################
## Services
###############################################################################
##
```

```
## Check general system resources such as load average, cpu and
    memory
## usage. Each test specifies a resource, conditions and the action
    to be
## performed should a test fail.
#
#  check system myhost.mydomain.tld
#    if loadavg (1min) > 4 then alert
#    if loadavg (5min) > 2 then alert
#    if memory usage > 75% then alert
#    if swap usage > 25% then alert
#    if cpu usage (user) > 70% then alert
#    if cpu usage (system) > 30% then alert
#    if cpu usage (wait) > 20% then alert
#
#
## Check if a file exists, checksum, permissions, uid and gid. In
    addition
## to alert recipients in the global section, customized alert can
    be sent to
## additional recipients by specifying a local alert handler. The
    service may
## be grouped using the GROUP option. More than one group can be
    specified by
## repeating the 'group name' statement.
#
#  check file apache_bin with path /usr/local/apache/bin/httpd
#    if failed checksum and
#       expect the sum 8f7f419955cefa0b33a2ba316cba3659 then
    unmonitor
#    if failed permission 755 then unmonitor
#    if failed uid root then unmonitor
#    if failed gid root then unmonitor
#    alert security@foo.bar on {
#           checksum, permission, uid, gid, unmonitor
#        } with the mail-format { subject: Alarm! }
#    group server
#
#
## Check that a process is running, in this case Apache, and that it
    respond
## to HTTP and HTTPS requests. Check its resource usage such as cpu
    and memory,
## and number of children. If the process is not running, Monit will
    restart
## it by default. In case the service is restarted very often and the
## problem remains, it is possible to disable monitoring using the
    TIMEOUT
## statement. This service depends on another service (apache_bin)
    which
## is defined above.
#
#  check process apache with pidfile /usr/local/apache/logs/httpd.pid
#    start program = "/etc/init.d/httpd start" with timeout 60
    seconds
#    stop program  = "/etc/init.d/httpd stop"
#    if cpu > 60% for 2 cycles then alert
#    if cpu > 80% for 5 cycles then restart
#    if totalmem > 200.0 MB for 5 cycles then restart
#    if children > 250 then restart
#    if loadavg(5min) greater than 10 for 8 cycles then stop
#    if failed host www.tildeslash.com port 80 protocol http
#       and request "/somefile.html"
#       then restart
#    if failed port 443 type tcpssl protocol http
#       with timeout 15 seconds
#       then restart
#    if 3 restarts within 5 cycles then timeout
#    depends on apache_bin
#    group server
#
#
## Check filesystem permissions, uid, gid, space and inode usage.
    Other services,
## such as databases, may depend on this resource and an
    automatically graceful
## stop may be cascaded to them before the filesystem will become
    full and data
## lost.
#
#  check filesystem datafs with path /dev/sdb1
#    start program  = "/bin/mount /data"
#    stop program  = "/bin/umount /data"
#    if failed permission 660 then unmonitor
#    if failed uid root then unmonitor
#    if failed gid disk then unmonitor
#    if space usage > 80% for 5 times within 15 cycles then alert
#    if space usage > 99% then stop
#    if inode usage > 30000 then alert
#    if inode usage > 99% then stop
#    group server
#
```

```
#
## Check a file's timestamp. In this example, we test if a file is
    older
## than 15 minutes and assume something is wrong if its not updated.
    Also,
## if the file size exceed a given limit, execute a script
#
#  check file database with path /data/mydatabase.db
#    if failed permission 700 then alert
#    if failed uid data then alert
#    if failed gid data then alert
#    if timestamp > 15 minutes then alert
#    if size > 100 MB then exec "/my/cleanup/script" as uid dba and
    gid dba
#
#
## Check directory permission, uid and gid.  An event is triggered
    if the
## directory does not belong to the user with uid 0 and gid 0.  In
    addition,
## the permissions have to match the octal description of 755 (see
    chmod(1)).
#
#  check directory bin with path /bin
#    if failed permission 755 then unmonitor
#    if failed uid 0 then unmonitor
#    if failed gid 0 then unmonitor
#
#
## Check a remote host availability by issuing a ping test and check
    the
## content of a response from a web server. Up to three pings are
    sent and
## connection to a port and an application level network check is
    performed.
#
#  check host myserver with address 192.168.1.1
#    if failed icmp type echo count 3 with timeout 3 seconds then
    alert
#    if failed port 3306 protocol mysql with timeout 15 seconds then
    alert
#    if failed url http://user:password@www.foo.bar:8080/?querystring
#       and content == 'action="j_security_check"'
#       then alert
#
#
```

```
###############################################################################
## Includes
###############################################################################
##
## It is possible to include additional configuration parts from
    other files or
## directories.
#
   include /etc/monit/conf.d/*
#


 check system dnsname.cloudapp.net
    if loadavg (5min) > 16 for 15 cycles then exec
        "/home/maxim/monitalert.sh"
    if memory usage > 85% then exec "/home/maxim/monitalert.sh"
    if swap usage > 40% then exec "/home/maxim/monitalert.sh"

# how to monitor a local container
# note that this hits localhost:80 with host header mywebsite.com
check host HOSTNAME with address localhost
        if failed port 80 protocol http with http headers [Host:
            "www.DOMAINNAME.com:80"] and timeout 10 seconds for 3
            times within 5 cycles then exec
            "/home/maxim/monit_docker_restart.sh CONTAINER-NAME"

# might as well use this monit to monitor some other things
check host www.externaldomain.com with address www.externaldomain.com
      if failed port 80 protocol http timeout 10 seconds for 3 times
          within 5 cycles then exec "/home/maxim/monitalert.sh"
     if failed port 443 protocol https timeout 10 seconds for 3
         times within 5 cycles then exec "/home/maxim/monitalert.sh"

check filesystem rootfs with path /
       if space usage > 95% then exec "/home/maxim/monitalert.sh"



set httpd port 2812
        use address 127.0.0.1
        allow admin:MYADMINPASSWORDHERE
```

my nginx setup

`/etc/nginx/nginx.conf`:

```
user www-data;
worker_processes 4;
pid /run/nginx.pid;

events {
    worker_connections 768;
    # multi_accept on;
}

http {

    ##
    # Basic Settings
    ##

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;
    # server_tokens off;

    # server_names_hash_bucket_size 64;
    # server_name_in_redirect off;

    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    ##
    # Logging Settings
    ##

    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    ##
    # Gzip Settings
    ##

    gzip on;
    gzip_disable "msie6";

    # gzip_vary on;
    # gzip_proxied any;
    # gzip_comp_level 6;
    # gzip_buffers 16 8k;
    # gzip_http_version 1.1;
    # gzip_types text/plain text/css application/json
        application/x-javascript text/xml application/xml
        application/xml+rss text/javascript;

    ##
    # nginx-naxsi config
    ##
    # Uncomment it if you installed nginx-naxsi
    ##

    #include /etc/nginx/naxsi_core.rules;

    ##
    # nginx-passenger config
    ##
    # Uncomment it if you installed nginx-passenger
    ##

    #passenger_root /usr;
    #passenger_ruby /usr/bin/ruby;

    ##
    # Virtual Host Configs
    ##

    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
}


#mail {
#   # See sample authentication script at:
#   # http://wiki.nginx.org/ImapAuthenticateWithApachePhpScript
#
#   # auth_http localhost/auth.php;
#   # pop3_capabilities "TOP" "USER";
#   # imap_capabilities "IMAP4rev1" "UIDPLUS";
#
#   server {
#       listen     localhost:110;
#       protocol   pop3;
#       proxy      on;
#   }
#
#   server {
#       listen     localhost:143;
#       protocol   imap;
#       proxy      on;
#   }
#}
```

set up git

- Add an ssh github key.
- `git config --global push.default simple`
- `git config --global core.autocrlf input`
- `git config --global user.name "Git Username"`
- `git config --global user.email gitemail@domain.com`

my final local `.ssh/config`

tunneling for:

- shiypard
- docker registry ui
- monit

```
Host myserver
  User maxim
  HostName myserverdns.cloudapp.net
  LocalForward 3100 localhost:8080
  LocalForward 4100 localhost:4100
  LocalForward 2812 localhost:2812
```

TODOS

TODO:

- update autobackup to upload to s3 and not to azure; just make it call s3cmd:

```
sudo apt-get install s3cmd
s3cmd --configure
s3cmd ls
s3cmd put backupdir/* s3://bucketname/backups/
```

- script that checks s3 backup bucket and confirms that new files being added
- logrotate setup
- munin, which will look something like this:

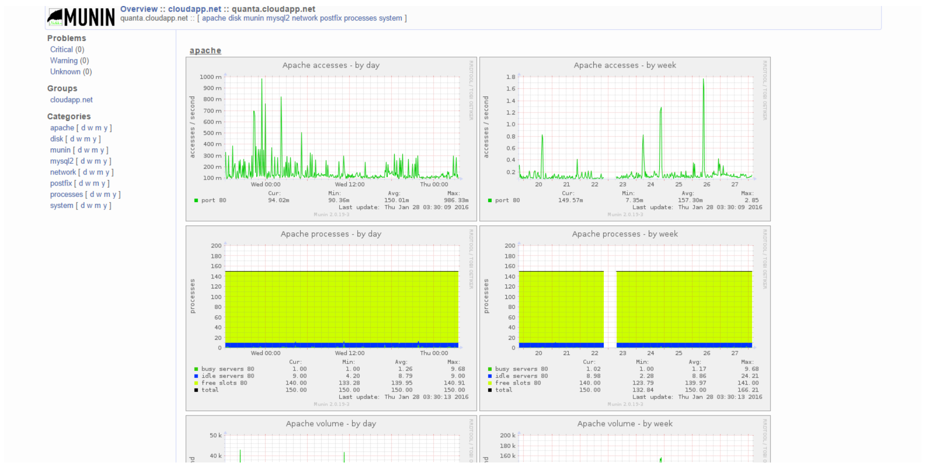


Figure 3: munin

Docker runbook

How to do all sorts of stuff with a container.

I'm going to use a wordpress container I host for a friend as an example. Note that this container goes a bit against docker philosophy because it includes two functionalities – web and db – as opposed to only one. But at least will demonstrate the main docker tasks.

Example nginx reverse proxy:

This example lives in `/etc/nginx/sites-available/mywebsite.conf` and is symlinked into `/etc/nginx/sites-enabled/`

```
upstream mywebsite.localhost {
    server 127.0.0.1:3100;
}

server {

    #listen 80;

    gzip_types text/plain text/css application/json
        application/x-javascript
               text/xml application/xml application/xml+rss
                   text/javascript;

    server_name .mywebsite.com;

    location / {
        proxy_pass http://mywebsite.localhost;
        include /etc/nginx/proxy_params;
    }
}
```

Dockerfile

```
FROM ubuntu:14.04
#MAINTAINER Eugene Ware <eugene@noblesamurai.com>
MAINTAINER Maxim Zaslavsky <maxim@maximz.com>

# Keep upstart from complaining
RUN dpkg-divert --local --rename --add /sbin/initctl
RUN ln -sf /bin/true /sbin/initctl

# Let the conatiner know that there is no tty
ENV DEBIAN_FRONTEND noninteractive

RUN apt-get update
RUN apt-get -y upgrade

# Basic Requirements
RUN apt-get -y install mysql-server mysql-client nginx php5-fpm
    php5-mysql php-apc pwgen python-setuptools curl git unzip

# Wordpress Requirements
RUN apt-get -y install php5-curl php5-gd php5-intl php-pear
    php5-imagick php5-imap php5-mcrypt php5-memcache php5-ming
    php5-ps php5-pspell php5-recode php5-sqlite php5-tidy
    php5-xmlrpc php5-xsl sendmail

# mysql config
RUN sed -i -e"s/^bind-address\s*=\s*127.0.0.1/bind-address =
    0.0.0.0/" /etc/mysql/my.cnf

# nginx config
```

```
RUN sed -i -e"s/keepalive_timeout\s*65/keepalive_timeout 2/"
    /etc/nginx/nginx.conf
RUN sed -i -e"s/keepalive_timeout 2/keepalive_timeout
    2;\n\tclient_max_body_size 100m/" /etc/nginx/nginx.conf
RUN echo "daemon off;" >> /etc/nginx/nginx.conf

# php-fpm config
RUN sed -i -e "s/;cgi.fix_pathinfo=1/cgi.fix_pathinfo=0/g"
    /etc/php5/fpm/php.ini
RUN sed -i -e "s/upload_max_filesize\s*=\s*2M/upload_max_filesize =
    100M/g" /etc/php5/fpm/php.ini
RUN sed -i -e "s/post_max_size\s*=\s*8M/post_max_size = 100M/g"
    /etc/php5/fpm/php.ini
RUN sed -i -e "s/;daemonize\s*=\s*yes/daemonize = no/g"
    /etc/php5/fpm/php-fpm.conf
RUN sed -i -e
    "s/;catch_workers_output\s*=\s*yes/catch_workers_output = yes/g"
    /etc/php5/fpm/pool.d/www.conf
RUN find /etc/php5/cli/conf.d/ -name "*.ini" -exec sed -i -re
    's/^(\s*)#(.*)/\1;\2/g' {} \;

# nginx site conf
ADD ./nginx-site.conf /etc/nginx/sites-available/default

# Supervisor Config
RUN /usr/bin/easy_install supervisor
RUN /usr/bin/easy_install supervisor-stdout
ADD ./supervisord.conf /etc/supervisord.conf

# Install Wordpress
ADD https://wordpress.org/latest.tar.gz
    /usr/share/nginx/latest.tar.gz
RUN cd /usr/share/nginx/ && tar xvf latest.tar.gz && rm latest.tar.gz
RUN mv /usr/share/nginx/html/5* /usr/share/nginx/wordpress
RUN rm -rf /usr/share/nginx/www
RUN mv /usr/share/nginx/wordpress /usr/share/nginx/www
RUN chown -R www-data:www-data /usr/share/nginx/www

# Wordpress Initialization and Startup Script
ADD ./start.sh /start.sh
RUN chmod 755 /start.sh

RUN mkdir /admin
RUN mkdir /hostshared


```

```
# private expose
EXPOSE 3306
EXPOSE 80

# volume for mysql database and wordpress install and for log files
VOLUME ["/var/lib/mysql", "/usr/share/nginx/www", "/var/log",
    "/admin", "/hostshared"]

CMD ["/bin/bash", "/start.sh"]
```

Build image (in **`~/images/mywebsite-wordpress/build.sh`**)

```
docker build -t maximz/mywebsite-wordpress . ;
docker tag -f maximz/mywebsite-wordpress
    maximz/mywebsite-wordpress:latest ;
```

Run:

```
docker run -p 3100:80 --name="maximz-mywebsite-wordpress" -v
~/data/shared:/hostshared --restart=always -d maximz/mywebsite-wordpress:latest
```

Attach

```
docker exec -it maximz-mywebsite-wordpress bash
```

How to migrate an old wordpress once attached

```
docker exec -it maximz-mywebsite-wordpress bash
> export MP=$(cat /admin/mysql-root-pw.txt);
> echo $MP;
> mysql -u root --password=$MP < /hostshared/mywebsite.sql # this is
    where old sql is. restores to database "wordpress"
> # then unzip any web content in /usr/share/nginx/www, which is the
    wordpress root
> # use
    http://codex.wordpress.org/User:MichaelH/Orphaned_Plugins_needing_Adoption/Emergency
    if you need to change admin password in wordpress
```

the point is that you can create container from image and then migrate things into it

backup, manual

```
docker run --rm --volumes-from maximz-mywebsite-wordpress -v
    ~/backups:/backup -it ubuntu:14.04 bash
> mkdir -p /backup/mywebsite
> tar zcvf /backup/mywebsite/mysql.tar.gz /var/lib/mysql
> tar zcvf /backup/mywebsite/www.tar.gz /usr/share/nginx/www
> tar zcvf /backup/mywebsite/log.tar.gz /var/log
> tar zcvf /backup/mywebsite/admin.tar.gz /admin
```

backup, one liner

see autobackup section earlier; the oneliner is in `do_backups.sh`

restore

```
# launch a new container as above, then stop it.

docker run --rm -it --volumes-from maximz-mywebsite-wordpress -v
    ~/backups:/backup ubuntu:14.04 bash
> tar zxvf /backup/mywebsite/mysql.tar.gz
> tar zxvf /backup/mywebsite/www.tar.gz
> tar zxvf /backup/mywebsite/admin.tar.gz
# we don't restore the logs
```

transfer data between containers

```
docker run --volumes-from maximz-mywebsite-wordpress --name
    temporarydatastore -it ubuntu:14.04 echo 'created';
# remove old container
docker run -p 3100:80 --name="maximz-mywebsite-wordpress" -d
    --volumes-from temporarydatastore
    maximz/mywebsite-wordpress:latest;
```

Docker tips and tricks

DNS errors while building?

Try: `docker build --no-cache -t maximz.myimagename .`

How to build with app content in root directory or elsewhere

http://stackoverflow.com/a/34392052/130164. (Lets you have dockerfiles in a subdirectory)

Alternatively/additionally, we can push a subdirectory to heroku: http://stackoverflow.com/questions/7539382
can-i-deploy-push-only-a-subdirectory-of-my-git-repo-to-heroku

Tail multiple files for docker logs

```
tail -f file1 | sed 's/^/file1: /' &
tail -f file2 | sed 's/^/file2: /'
```