Title: How to mount local volumes in docker machine

Post Body:

I am trying to use docker-machine with docker-compose. The file docker-compose.yml has definitions as follows:

```
web:   build: .   command: ./run_web.sh   volumes:     - .:/app   ports:     - '8000:8000'   links:     - db:db     - rabbitmq
```

When running `docker-compose up -d` all goes well until trying to execute the command and an error is produced:

> Cannot start container b58e2dfa503b696417c1c3f49e2714086d4e9999bd71915a53502cb6ef43936d: [8] System error: exec: './run_web.sh': stat ./run_web.sh: no such file or directory

Local volumes are not mounted to the remote machine. Whats the recommended strategy to mount the local volumes with the webapps' code?

Accepted Answer: None

Highest Rated Answer:

Docker-machine automounts the users directory... But sometimes that just isn't enough.

I don't know about docker 1.6, but in 1.8 you *CAN* add an additional mount to docker-machine

Add Virtual Machine Mount Point (part 1)

CLI: (Only works when machine is stopped)

```
VBoxManage sharedfolder add <machine name/id> --name <mount_name> --hostpath <host_dir> --automount
```

So an example in windows would be

```
/c/Program\ Files/Oracle/VirtualBox/VBoxManage.exe sharedfolder add default --name e --hostpath 'e:\' --automount
```

GUI: (does NOT require the machine be stopped)

1. Start 'Oracle VM VirtualBox Manager'
2. Right-Click `<machine name>` (default)
3. Settings...
4. Shared Folders
5. The Folder+ Icon on the Right (Add Share)
6. Folder Path: `<host dir>` (e:)
7. Folder Name: `<mount name>` (e)
8. Check on 'Auto-mount' and 'Make Permanent' (Read only if you want...) (The auto-mount is sort of pointless currently...)

Mounting in boot2docker (part 2)

Manually mount in boot2docker:

1. There are various ways to log in, use 'Show' in 'Oracle VM VirtualBox Manager', or ssh/putty into docker by IP address `docker-machine ip default`, etc...
2. `sudo mkdir -p <local_dir>`
3. `sudo mount -t vboxsf -o defaults,uid=`id -u docker`,gid=`id -g docker` <mount_name> <local_dir>`

But this is only good until you restart the machine, and then the mount is lost...

Adding an automount to boot2docker:

While logged into the machine

1. Edit/create (as root) `/mnt/sda1/var/lib/boot2docker/bootlocal.sh`, sda1 may be different for you...

   Add

   ```
   mkdir -p <local_dir> mount -t vboxsf -o defaults,uid=`id -u docker`,gid=`id -g docker` <mount_name> <local_dir>
   ```

With these changes, you should have a new mount point. This is one of the few files I could find that is called on boot and is persistent. Until there is a better solution, this should work.

Old method: **Less recommended**, but left as an alternative

- Edit (as root) `/mnt/sda1/var/lib/boot2docker/profile`, sda1 may be different for you...

  Add

  ```
  add_mount() {   if ! grep -q 'try_mount_share $1 $2' /etc/rc.d/automount-shares ; then     echo 'try_mount_share $1 $2' >> /
  ```

As a **last resort**, you can take the slightly more tedious alternative, and you can just modify the boot image.

- `git -c core.autocrlf=false clone https://github.com/boot2docker/boot2docker.git`
- `cd boot2docker`
- `git -c core.autocrlf=false checkout v1.8.1` #or your appropriate version
- Edit `rootfs/etc/rc.d/automount-shares`

  Add `try_mount_share <local_dir> <mount_name>` line right before fi at the end. For example

  ```
  try_mount_share /e e
  ```

  Just be sure not to set the to anything the os needs, like /bin, etc...

- `docker build -t boot2docker .` #This will take about an hour the first time :(
- `docker run --rm boot2docker > boot2docker.iso`
- Backup the old boot2docker.iso and copy your new one in its place, in ~/.docker/machine/machines/

This does work, it's just long and complicated

docker version 1.8.1, docker-machine version 0.4.0