UMBC Cyber Dawgs - Advanced VMs

By: Zack Orndorff and Christian Beam

Outline

You're going to configure a VM to serve MySQL and Git on one machine.

Protip: Don't copy-paste from this document. Google Drive made the quotes curly quotes, and curly quotes and UNIX don't mix.

We recommend you have another Linux desktop VM to connect to this VM from, but it's possible to do from your host as well.

What's been done in the OVA

In the OVA, I've installed CentOS 7, created a user "umbccd", with password "umbccd", and downloaded everything you'll need.

If you get an error about USB in your VM, you will need to go to the VM settings, go to the "Ports" tab, and then "USB", and select USB 1.1.

If sudo doesn't work for you (which it won't, I forgot to set the permissions for that), you'll need to log in as root (password umbccd) and run "usermod -G wheel umbccd", at which point you can log back in and it'll work.

VM networking is fun. At the time I did the research for this lab, in order to have a VM access the local machine and the internet, you needed to have 2 adapters, one for each of those purposes. I arbitrarily put the NAT network (to access the internet) on the first adapter, and the host-only network (for accessing from your web browser) on the second adapter.

I used the default IPs that Virtualbox used to use, but it seems that default has changed, so you will need to fix the ips. To do that, observe what the IP looks like for the host-only adapter in `ipconfig` on your host and adjust the third octet following the instructions at the end.

If you don't have a host-only adapter, you will need to create one.

You can just yum install as normal, I've cached the packages for you.

For gogs, I've already downloaded that package, you can find it in /opt.

Install MySQL

1. Source: https://www.digitalocean.com/community/tutorials/how-to-install-mariadb-on-centos-7
2. Use Yum to install MariaDB
   a. `sudo yum install mariadb-server`
3. Start the server
   a. `sudo systemctl start mariadb`
4. Check its status
   a. `sudo systemctl status mariadb`
   b. It should say active (running), let us know if it says something else
5. Enable MariaDB to start on boot
   a. `sudo systemctl enable mariadb`
   b. You should see something like "Created symlink from …... to …..."
6. Secure the server
   a. We really should make you do this manually, but for time's sake we will allow you to use this security script provided by MariaDB (also we think it's best practice)
      i. `sudo mysql_secure_installation`
      ii. Hit enter when it asks for a root password (you haven't set it yet)
      iii. Hit y and provide a root password for mysql
      iv. Proceed to hit y for the rest of the options provided
7. Test the server
   a. `mysqladmin -u root -p version`
   b. This should output something like "mysqladmin Ver 9.0 Distrib 5.5.56-MariaDB, for Linux on x86_64..." and more

Install Gogs

Gogs is a Git server. It's like GitHub, only free, and you can host it yourself. Its competitors include Bitbucket and GitLab. But Gogs is easier to install, so we're doing that.

1. `yum install git`
   a. Install the Git version control software
2. `sudo useradd git`
   a. Not a special name, but it's used by convention.
3. `sudo -iu git`
   a. Equivalent to `sudo su - git`
4. `curl -LO https://dl.gogs.io/0.11.66/gogs_0.11.66_linux_amd64.tar.gz`
   a. I've downloaded it for you, you can just copy it from /opt
5. `tar xf gogs_0.11.66_linux_amd64.tar.gz`
   a. Older versions of tar would have required tar xzf, with the z standing for Gzip.

   b. Any recent version of GNU tar should work with just tar xf
   c. https://xkcd.com/1168/
6. `cd gogs`
7. `mysql -u root -p`
   a. *type password*
   b. `source scripts/mysql.sql`
      i. creates a database for gogs, it’s a simple script provided by them
   c. `CREATE USER ‘gogs’@’localhost’ IDENTIFIED BY ‘gogsisawesome’;`
      i. As a general rule, most database users are for applications, not for human users.
      ii. In production, you’d actually pick a decent password.
   d. `GRANT ALL PRIVILEGES ON gogs.* TO ‘gogs’@’localhost’;`
      i. You can give less permissions in theory, but it still has to be able to read everything...
   e. exit
8. `./gogs web`
   a. Run the application
9. Go to http://`192.168.56.25`:3000/ in a web browser
10. Configure Gogs:
    a. to use the database and user (NOT root) you just created for MySQL
    b. Change the Application URL to be http://`192.168.56.25`:3000/
    c. Under “Server and Other Services Settings”, enable offline mode
       i. You shouldn’t need this... but it’ll reduce problems if stuff breaks.
    d. Expand the menu to create an admin account and create one for yourself.
11. Play around
12. Now go Ctrl+c the Gogs program
13. Log out of the git user (type exit)
14. Let’s configure systemd to start Gogs automatically on boot
15. `sudo cp /home/git/gogs/scripts/systemd/gogs.service /etc/systemd/system`
    a. gogs.service is a **systemd unit file**, equivalent to an old-style init script
16. `sudo vi /etc/systemd/system/gogs.service`
    a. Change the After= line with all the databases to be just After=mariadb.service
    b. Don’t change the other After= lines
17. `sudo systemctl start gogs.service`
18. `sudo systemctl enable gogs.service`
19. Now go try to log into Gogs again in your web browser and see if it still works.

Install Nginx

We're setting Nginx up as a reverse proxy for Gogs. This means that Nginx will be directly interacting with users, and it'll pass traffic to Gogs as necessary. This means you can use multiple server blocks/virtual hosts and share port 80 across different domain names.

1. `sudo yum install epel-release`
    a. CentOS only includes certain packages in the default repos, carrying certain support guarantees.
    b. Other packages are in EPEL, IIRC maintained by Fedora
    c. Still pretty good support
2. `sudo yum install nginx`
    a. You'll have to okay EPEL's signing key since this will be your first package coming from there
3. `sudo systemctl start nginx`
4. `sudo systemctl enable nginx`
5. Now go to http://192.168.56.25 in a web browser and see if it works.
    a. Should be the nginx default page
6. `sudo vi /etc/nginx/nginx.conf`
7. Edit the server block to contain settings like these:

```
server {
     listen 80;
     server_name 192.168.56.25;

     location / {
          proxy_pass http://192.168.56.25:3000/;
     }
}
```

What's a server block you ask? Each server corresponds to a "virtual host" in Apache terms.

8. `sudo setsebool -P httpd_can_network_connect true`
    a. Tell SELinux that reverse proxying is okay
    b. The -P is for permanent, IE for after a reboot
9. `sudo systemctl restart nginx`
10. Try going to http://192.168.56.25/
    a. It should work now
11. At this point, if we were doing firewalls, we would block off all access to port 3000 from the outside.

Test your configuration

1. Create a Gogs repo
2. Clone it to your VM

3. Make some changes
4. Push them back. Check them out in the Web interface

Extra stuff:

This stuff has already been done in the ova, but I'm leaving it here for completeness.

Install CentOS 7

1. You can download the ISO from here: https://www.centos.org/download/
    a. I recommend the Minimal ISO
    b. Fast mirror is: http://mirror.umd.edu/centos/7/isos/x86_64/CentOS-7-x86_64-Minimal-1708.iso
        i. We have a fast connection to most other universities, and College Park is especially fast
    c. SHA256 sum should be: `bba314624956961a2ea31dd460cd860a77911c1e0a56e4820a12b9c5dad363f5`
    d. Yes, verify it, it doesn't take that long.
2. Create a new VM with these settings:
    a. Red Hat (64-bit)
        i. CentOS is based on Red Hat
    b. 1024 MB RAM (you can probably get by with 512 though)
    c. 10 GB Disk
3. Configure the network in the VM (do it now, it'll save you frustration)
    a. Go to VM settings
    b. Adapter 1 -> NAT
    c. Adapter 2 -> Host-only
        i. If you are on a Linux host (If you are not sure, you aren't), you will need to configure a host-only network. Go to File > Preferences, then go to the Network tab, then the Host-only tab of that and create one.
4. Start up the installer
5. Configure "Installation Destination"
    a. Just click Done, the defaults are fine for what we're doing
6. Configure "Network and Host Name
    a. Click each network adapter and turn on the switch
    b. Click done
    c. This saves you having to manually enable the adapters later. It's easier, I promise.
    d. Change the hostname to "advlab.prac.umbccd.net"
        i. You can pick your own domain, but it is best practice to use one you actually own.

            ii. We own that one :)
            iii. Just pick one if you want, this isn't production
      e. Click Done.
7. Click "Begin Installation"
8. Set a root password, and create a user for yourself.
   a. Make yourself an administrator, that way sudo works. Otherwise you'll have to manually add yourself to the group.
9. Wait for it to finish
10. When it finishes, click the button to restart the VM into your new install.
11. When the machine boots, log in as your user.

Set a static IP address (only if you have VMWare, I've set it for VirtualBox)

As a general rule, you want your DNS servers to have a static IP address, otherwise you don't have a server to contact in order to find out what their address is.
You also want servers in general to have a static IP...

1. `sudo vi /etc/sysconfig/network-scripts/ifcfg-enp0s8`
2. Set `ONBOOT=yes` if not already set
   a. (otherwise you'd need to start it manually on every reboot)
3. Change `BOOTPROTO=dhcp` to `BOOTPROTO=static`
4. Add the following:
   a. `IPADDR=192.168.56.25`
   b. `NETMASK=255.255.255.0`
   c. `GATEWAY=192.168.56.1`
   d. `DEFROUTE=no`
      i. This keeps it from trying to route your internet traffic over the host-only adapter.
5. `sudo systemctl restart network`
6. Run `ip addr` or `ip a` to check changes
   a. These are the newer version of `ifconfig`, if you've heard of it