**Automagically update git repositories on your VMs**

It sure would be nice not to have to manually update your VMs. After following this tutorial, your Azure VM will automatically update itself (by running git pull) immediately after you push to your GitHub repository.

**Step 1: An update.php script**

First, create a new file in top-level directory of your GitHub repository named update.php which contains the following code:

<?php $result = exec("git pull"); ?>

Push this code to GitHub. It should be available at [https://[FQDN]/[your-GH-repo]/update.php](about:blank)

It won’t work until you finish this tutorial.

You will need to manually run git pull one last time on your VM to get the update script on it.

**Step 2: Add a GitHub webhook**

Using your web browser, navigate to your repository on GitHub. Once on that page, click on the Settings button. On the left-hand column of the settings page, you will see an option for Webhooks. Click it.

A webhook is a RESTful method for GitHub to notify external services that something has changed in your repository. It does so by sending a POST request to an address you specify every time you push to your repository. Click the button that says Add webhook. You will be prompted to log into GitHub.

On the add webhook page, you will need to change the Payload URL. The Payload URL is [https://[FQDN]/[your-GH-repo]/update.php](about:blank)

The Enable SSL verification button should be enabled. But if not, enable it. And we only need to trigger the webhook on push events, which should be the default. Finally, make sure the Active checkbox is checked.

Once you’ve done everything, click the Add webhook button.

**Step 3: Setting up sudo access for Apache**

Now, we need to be able to have Apache run the git command. To do so, we will need to add apache to our sudoers file. This can be done by running the command sudo visudo

You want to scroll down until you see this line:

%sudo ALL=(ALL:ALL) ALL

Immediately underneath that line, add this line:

www-data ALL=NOPASSWD:/usr/bin/git

This added line says, in effect, “for the www-data user, on this host, permit the use of the /usr/bin/git binary without the need for a password.”

Save and quit.

**Step 4: Adding SSH keys**

You will need to add SSH keys in the appropriate location for Apache to find them when running git.

If your SSH keys have a password on them, you will need to make a new set of SSH keys just for Apache. Follow previous tutorials for making SSH keys and do not put a password on these new keys.

You will need to create a .ssh directory for Apache:

$ cd /var/www

$ sudo mkdir .ssh

Next, you will need to add the private key for your GitHub repo to that directory:

$ cd .ssh

$ sudo cp ~/.ssh/id\_ed25519 .

$ sudo cp ~/.ssh/id\_ed25519.pub .

$ sudo chmod 600 id\_ed25519

**Step 5: Changing ownership of /var/www**

At the moment, the /var/www directory is probably owned by root. We need it to be owned by www-data (the Apache user). We can do so like this:

$ cd /var

$ sudo chown -R www-data:www-data www

**Step 6: Final setup**

We need to do all the first-run git things. Run these commands:

$ cd /var/www/html/<your-GH-repo>

$ sudo -u www-data git pull

**Step 7: Test**

On GitHub Desktop, add a new file named test.txt to your lab8 directory. It can have any contents in it. Push that new file to your GitHub repository.

Then, on your VM, cd to your /var/www/html/[your-GH-repo]/lab8 directory and run the ls command. You should see your new file in there. If you do, then you are all set.

**Step 7: Secure**

If you would like some added security with your webhook, follow GitHub’s instructions here: <https://docs.github.com/en/developers/webhooks-and-events/webhooks/securing-your-webhooks>