Harris Linux server notes

These instructions are intended to get WebCTRL 8 running on a Ubuntu 20.04 server starting from scratch for Harris hosting. These reference instructions elsewhere but some of the steps are changed and noted here.

Linux users note; For security reasons ALC reccomends running WebCTRL under a standard user. The user can have sudo privilages but it must not be the actual root user. Secondly for XRDP compatibility that user must be secured with a password so that you can fill out the login details in remote desktop.

The rest of this document assumes user webctrl for the WebCTRL system service to run under.

Note; to swith users via CLI; su {username} eg; su webctrl

Per ALC instructions WebCTRL will be running in the home directory of this standard user and it will create cache files under that user file permissions. While it is possible to use "sudo" to execute SiteBuilder or the WebCTRL service DO NOT DO THAT! Running WebCTRL as sudo in the users folder will create cache files under sudo user, which are inaccessable to that user even though they are in that users home directory. If that happens the next time the server is started under the user normally it will not be able to access those files, which typically manifest as 4xx errors in web pages that won't load. IF this happens stop WebCTRL and clean out the contents of the cache located in ~/WebCTRL8.0/jspcache/org/apache/jsp using command sudo rm –r *

The next time the service is correctly started under the user account as non sudo the cache will be rebuilt with correct file permissions.

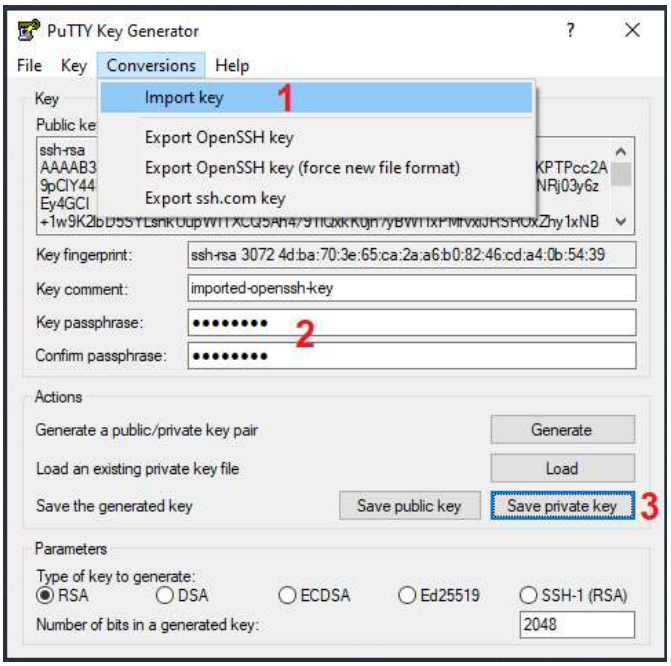
Background; having WebCTRL run under a non root account actually creates a problem because only privilaged accounts can open TCP/IP ports below 1024. That is why we use iptables to forward HTTP 80 -> 8080 and HTTPS 443 -> 8443.

While there is no IANA official alternate port number for HTTPS it is common practice in the industry to see HTTPS on 8443. There is a IANA alternat port number for HTTP mentioned at 8080 and several others.

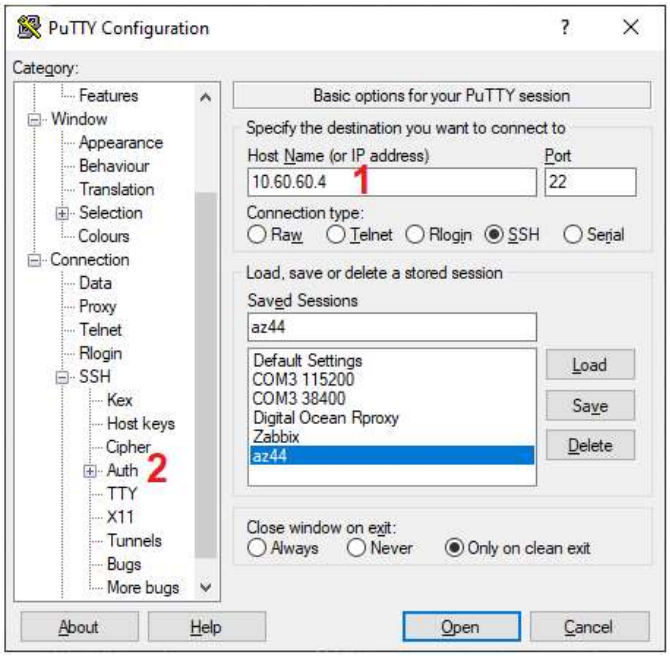
Tools Required

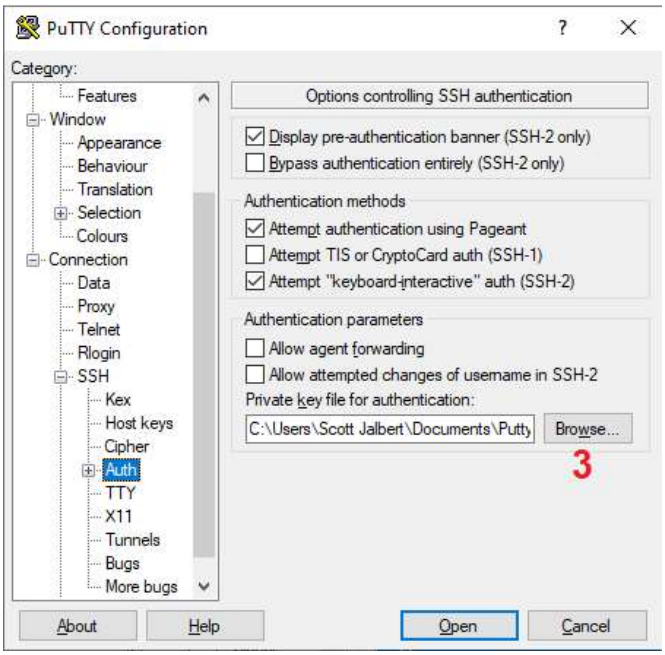
Tools; The full install of Putty for SSH command line interface access.
https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

Use PuTTYgen to convert the provided pem file to a ppk file PuTTY can use. Conversions, Import Key and select the pem file. Enter a password you make up in the passphrase. Save private ppk file in my documents where Putty can reach it.



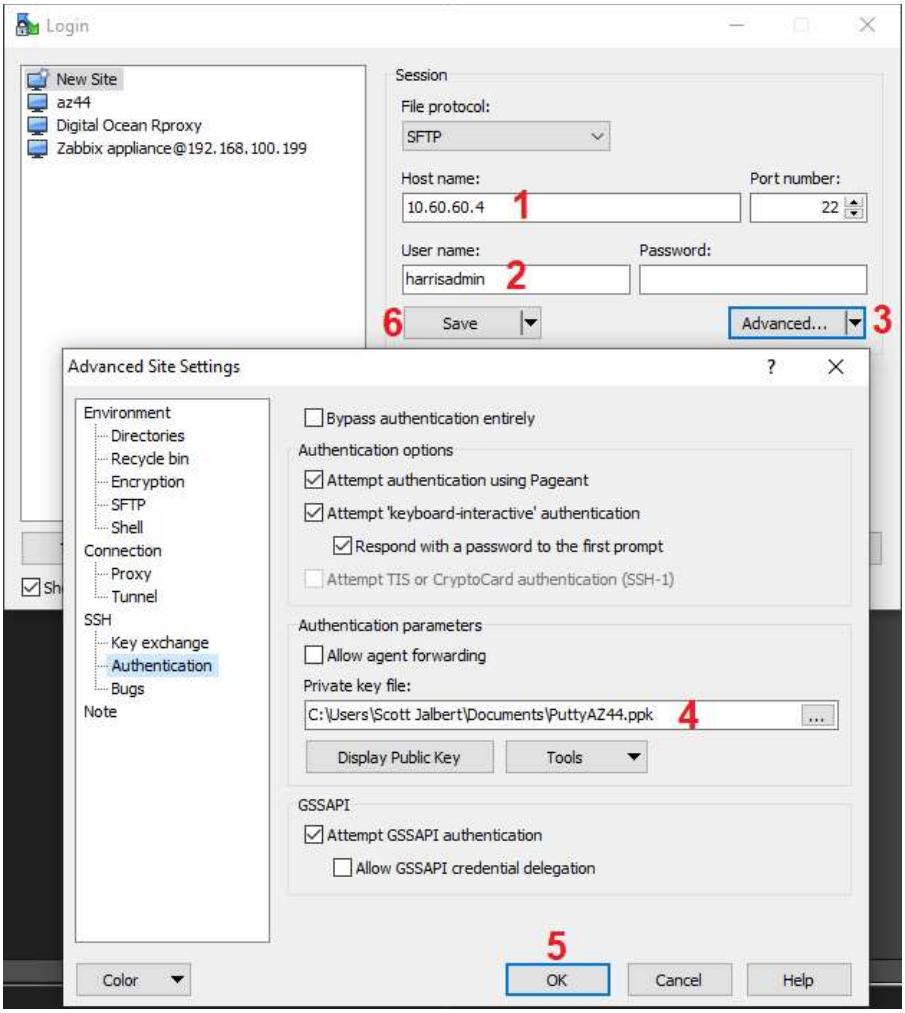
To start the SSH session enter the IP address and click Auth under SSH in Category. Browse to your private key and click Open. You will first be prompted for the username, this is the linux username, the passphrase is what you made up when converting the PPK file.





Tools; WinSCP for transferring files; https://winscp.net/eng/download.php

To connect open a new site, fill in the IP and linux user name. Then use the drop down arrow on advanced to select advanced. In the popup window select Authentication and load the ppk file you previously made with puttygen. Save that connection. When you open this connection you will will be prompted for your password you previously made up with puttygen.



Commissioning a new linux server

Putty SSH into the server and run updates

sudo apt update

sudo apt upgrade

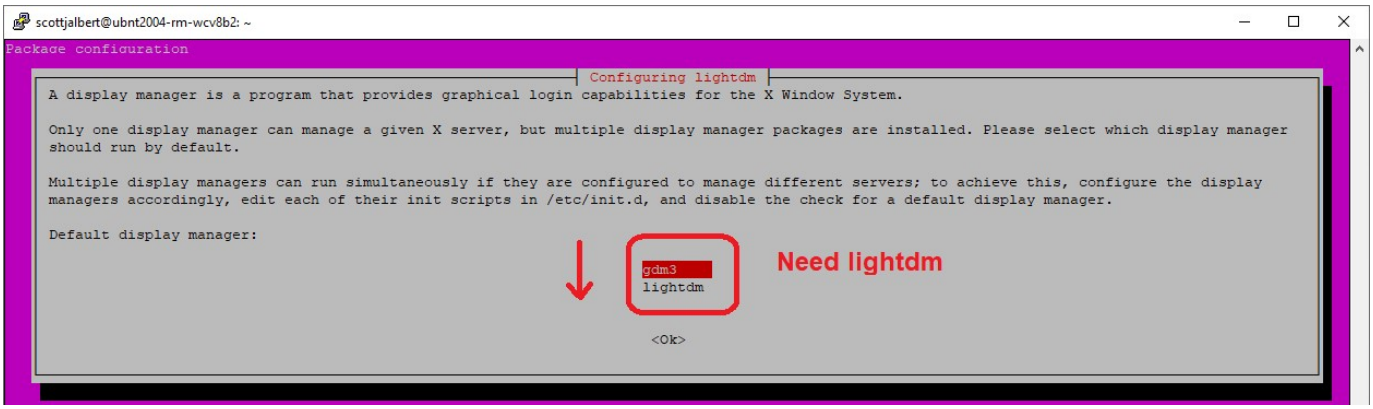
Install Ubuntu Desktop with Xfce for GUI remote desktop access. https://idroot.us/install-xrdp-server-ubuntu-20-04/

sudo apt install ubuntu-desktop

sudo apt update

sudo apt install xubuntu-desktop

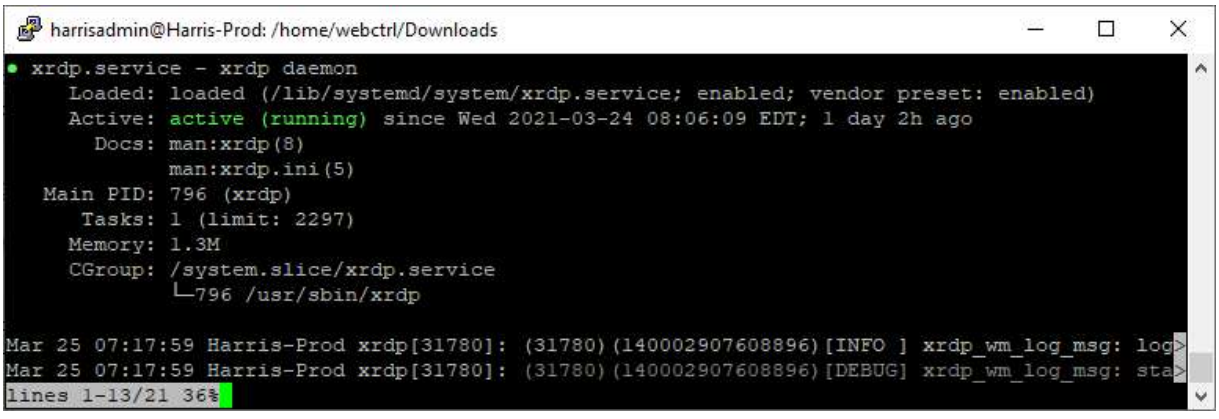
During install you will be asked to select a display manager, select lightdm



sudo apt install xrdp

sudo ufw allow 3389

sudo systemctl status xrdp Status should be running/green



Create the WebCTRL user account with sudo and document it in password keeper

adduser webctrl (follow prompts)

usermod –aG sudo webctrl

Make a note of the user group “groups <username>”; groups webctrl

Use WinSCP and upload the webctrl installer.sh, license file, certkeys and patches to ~/. That is a sym link for full path /home/harrisadmin. Then use sudo to move those to the webctrl download folder. WinSCP itself can't escilate to sudo to write straight to that directory because it belongs to another user.

```
sudo mv ~/*.* /home/webctrl/Downloads
```

```
sudo mv ~/* /home/webctrl/Downloads
```
(this moves certkeys, which doesn't have a file extension)

Use windows remote desktop and connect to the linux servers private IP address. Login with the webctrl credentials you created.

In RDP GUI desktop open a terminal session and go to your downloads folder.

```
cd ~/Downloads
```
full path behind the symlink is /home/webctrl/Downloads

Run the WebCTRL installer using script interpreter shebang. This will bring up the GUI linux WebCTRL installer.

```
/bin/sh WebCTRL_8.0_linux64_setup.sh
```

Default install directory for WebCTRL is in the user home folder ~/WebCTRL8.0 or full path /home/webctrl/WebCTRL8.0, keep that default. The rest of the GUI installer steps will be similar to windows.

Move the certkeys file you uploaded to the web server, you should receive a ovewrite warning

```
mv certkeys ~/WebCTRL8.0/webserver/keystores
```

To open SiteBuilder, change to the install directory;

```
cd ~/WebCTRL8.0
```

Launch the SiteBuilder script;

```
./SiteBuilder
```

GUI SiteBuilder should launch and function the same as the windows version. Change these settings;

- Web server on HTTPS ONLY
- HTTPS port is 8443
- Keystore password is in password keeper
- TLS 1.3 and 1.2
- Java VM maximum 3072 MB (assuming 4GB server memory)

At this point WebCTRL is installed and functional, and we will install the system service later.

Extra software and firewall

Install 7-Zip to work with zip files;

```
sudo apt install p7zip p7zip-rar p7zip-full –y
```

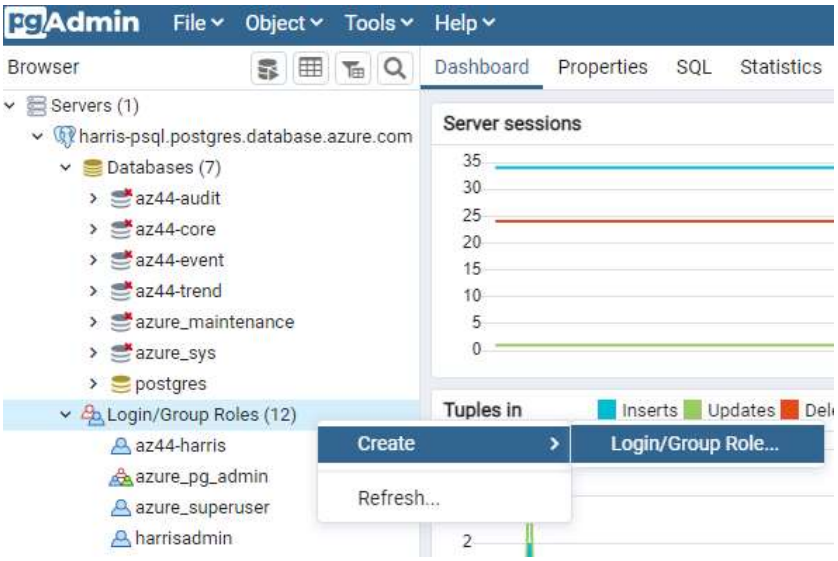
Check firewall and add rules

```
sudo ufw status            -inactive by default
sudo ufw show added    -list everything, LOOK FOR SSH and add it if it isn't there
sudo allow ssh   -keeps from locking yourself out of terminal
sudo ufw allow 80/tcp   -privilaged HTTP port (alternate command sudo ufw allow http)
sudo ufw allow 8080/tcp          -non-privilaged HTTP port
sudo ufw allow 443/tcp -privilaged HTTPS port (alternate command sudo ufw allow https)
sudo ufw allow 8443/tcp          -non-privilaged HTTPS port
sudo ufw allow 3389      -Remote Desktop
sudo ufw allow 1194/tcp          -OpenVPN for BMS
sudo ufw enable
sudo ufw status
```

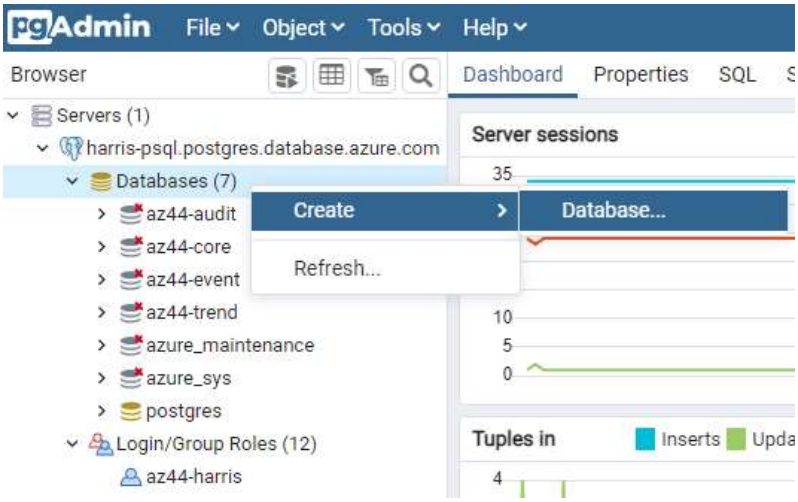
Job file conversion; All Harris Azure Linux servers use PostgreSQL hosted by Azure. It is not in a VM, it is part of the Azure environment. The DB can be administred remotely via pgAdmin, which itself is installable on windows. The "Bastian" server AZ00 has pgAdmin installed. Connection details for PostgreSQL server are in xxxxx site provided by xxxxx.

Detailed steps are provided in ALC doc "Operating system and database selection v8.0.pdf" but the major steps are;

1. Add a new PostgreSQL user by right clicking the login section

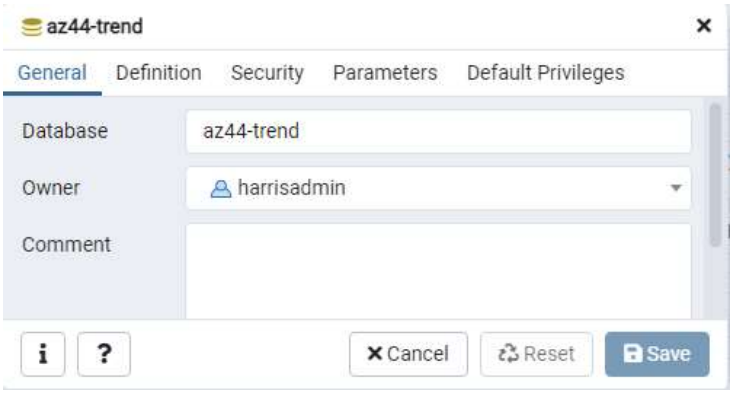


2. Enter tab info, leave all default except;
    a. General; username is previxed with server number, eg; az45-examplecustomer
    b. Definition; unique password, save these credentials in password keeper
    c. Priviliges; can login yes
        i. Save to close the window
3. Add a new database by right clicking Databases



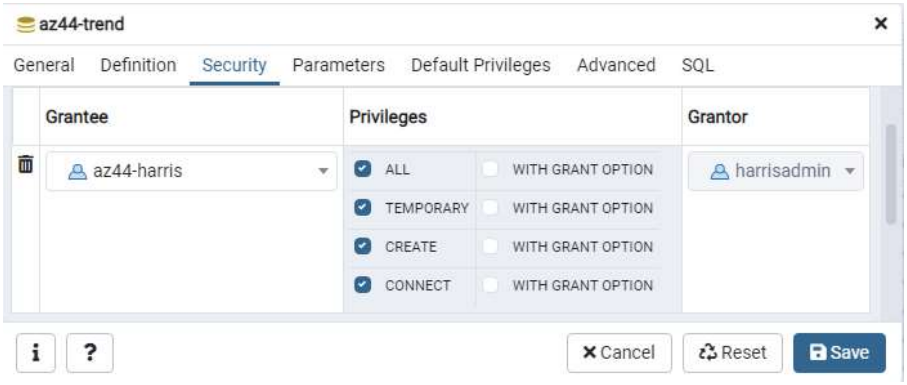
4. Enter tab info, leaving all default except;
    a. Database name, prefixed with server number; az45-main (audit, alarms, trends)

b. Database Owner is the user you logged into the PostgreSQL server under, harrisadmin



c. Security tab click the + sign to get a new row

d. On the Security tab fill in;
   i. Grantee; the user you created for this server
   ii. Privileges (click empty area); All
   iii. Grantor; The harrisadmin account you logged into PostgreSQL with



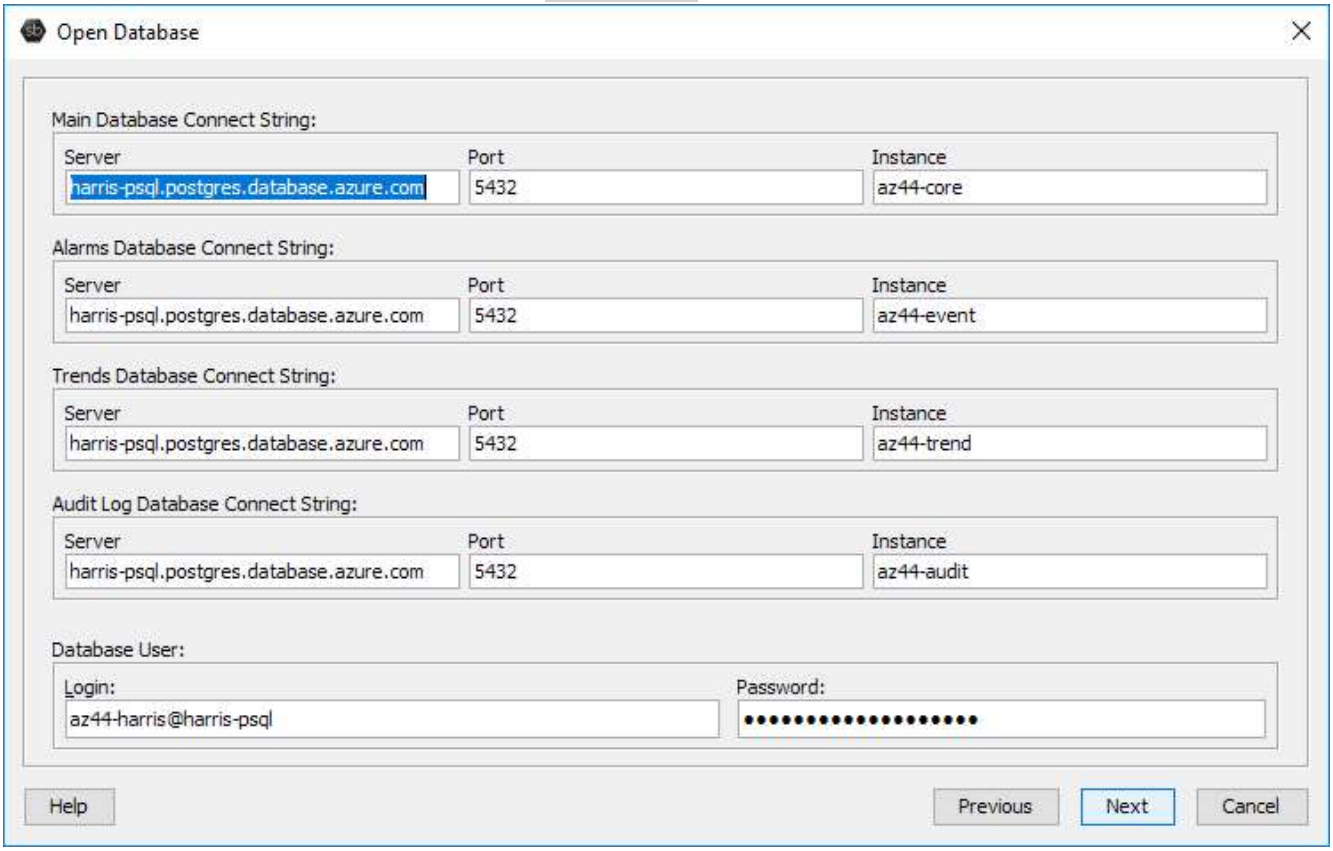
e. Save and Repeat step 4 for audit, alarms, trends databases.

Loading job file into Azure

From here if your job file is Derby you can upload the backup zip to the Linux server, we will handle the conversion to PostgreSQL in Linux.

If the job file is MySQL and small you could convert it to derby and send that to the Linux server, but if it is big I would suggest using the Bastian AZ00 to feed the MySQL contents into the Azure PostgreSQL server. The AZ00 server has Harris MySQL loaded on it, use that to mount the MySQL database in your job file and open it in SiteBuilder 8. Contact Donnie or Wade to get this machine started up, it may be shut down and unloaded to save money.

Either way loading the job file into PostgreSQL will be very similar once it is opened in SiteBuilder.

1. Open the job file in SiteBuilder and select File, manage database.
2. Chose Replicate (does not effect current system).
3. You will be asked for a new job file name, use the form; v80_azpg_customer_name.
4. SQL connect strings;
    a. Server is the full server name you originally used to open in pgAdmin
    b. Port is 5432
    c. Instance is the name of the database you created for this server
    d. Login name and password are the ones you created for this server in pgAdmin. The login name needs to be fully qualified with suffix of the DB server; user@server
    e. 

    f. Next screen should be a summary of the info you entered on the previous step. Any errors in connecting will be called out here in red. Use the previous button to go back and make changes and try again. Selecting next will progress to the next screen and show conversion process.
    g. When done if there are no errors you will have a split set of WebCTRL job files. The database will be in Azure and the static files will be in the v80_azpg_customer_name job file folder. Together they make up the job files but there is no one directory you can zip up to make a backup. The Azure infrastructure handles redundancy and backups.

    h. Cleanup; during the file copy process the MySQL DB contents will be carried over to the PostgreSQL job file folder and takes up a LOT of space; delete the mysql_data folder from the azpg job file folder.
5. From this point if you created the v80_azpg_customer_name job file in Windows zip it and use WinSCP to upload it to ~/ under the harrisadmin user.
    a. Using the harrisadmin account in SSH via putty; Unpack the ZIP file; 7z x -o"ext" v80_azpg_customer_name.zip Will extract contents of zip file with full paths inside new "ext" folder.
    b. Move that folder into webroot; sudo mv ~/test/ext/v80_azpg_customer_name ~/WebCTRL8.0/webroot
6. Open the v80_azpg_customer_name job file in Linux SiteBuilder
    a. Log into the GUI desktop via Windows Remote Desktop app
    b. Open terminal and chagne to the WebCTRL app directory; cd ~/WebCTRL8.0
    c. Launch GUI SiteBuilder; ./SiteBuilder
    d. Set your v80_azpg_customer_name job file as default before you open it. Should be no errors in opening the job file.
    e. Close SiteBuilder.
7. Run the WebCTRL server as GUI application ./'WebCTRL Server' The GUI application will open and concurrent SiteBuilder will be available. Observe and correct any errors listed in GUI window. More details will be available in the text log files located in ~/WebCTRL8.0/logs
8. In a browser on your computer test the following;
    a. https://serverip:8443 -if this fails check your SiteBuilder logs to see if WebCTRL started
    b. Don't bother with other ports or http-https yet, those aren't enabled yet.

Install WebCTRL Linux Service and starting/stopping

1. Ensure you are user webctrl first
2. Get into the WebCTRL app directory; cd ~/WebCTRL8.0
3. Add the service; sudo "./WebCTRL Service" add
4. Starting the service; ./'WebCTRL service' start
5. Stopping; ./'WebCTRL Service' stop

ALC Linux iptables known problem 3/23/2021 case ID 892-27246;

If you follow the ALC linux install instructions for WebCTRL including setting up IPTables AND turn on HTTP->HTTPS redirect in SiteBuilder you wind up with external HTTP(80) being redirected to HTTPS(4430) which will be blocked at the external Azure firewall and the page won't load. Expectation was to have redirect pointed to HTTPS(443) and let IPTables handle the 443->4430 portion. That is why in the previous section we selected HTTPS only for SiteBuilder and we are now only going to make an iptable entry only for the 443-8443 redirect.

1. Review current iptables settings; sudo iptables --list -t nat
2. Install iptables app
    a. sudo apt-get update
    b. sudo apt-get install iptables (if you are looking at the ALC instructions, do NOT do the persistent yet)
3. Redirect standard HTTPS port 443 to the selected alternate port (ex: 8443)
    a. sudo iptables -t nat -I PREROUTING -p tcp --dport 443 -j REDIRECT --to-ports 4430
    b. sudo iptables -t nat -I OUTPUT -p tcp -o lo --dport 443 -j REDIRECT --to-ports 4430
4. The above tables are temporary and will go away after a reboot, TEST them first. In a browser on your computer test this address;
    a. https://serverip -if this fails check your iptables rules, also; is the WebCTRL Server application or service running?
5. If that worked make that permnant;
    a. sudo apt-get install iptables-persistent
6. The last step installed the persistent service and imported your temporary tables so iptables will recover after a reboot. The iptables files are saved at /etc/iptables/rules.v4 and /etc/iptables/rules.v6.

Workaround with nginx

Ideally in the previous section we would also enter a simple iptable rule to redirect TCP 80 to 8080, we can't currently do that due to the HTTP -> HTTPS redirect in WebCTRL being broken. To take the place of that feature we can use nginx web server to do a very simple 301 redirect for 80 to 443.

1. sudo apt update
2. sudo apt install nginx -nginx app will install, start, and add a service automatically
3. Back up existing config file; sudo mv /etc/nginx/nginx.conf ~/
4. Delete existing config file; sudo rm /etc/nginx/nginx.conf
5. Make new nginx config in nano text editor; sudo nano /etc/nginx/nginx.conf
6. Paste in the following contents;

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

events {
      worker_connections 768;
}

http {

   server {
     listen        8080;
     return        301 https://$host$request_uri;
   }

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

7. Use ctrl – x to exit nano, saying Y to save changes.
8. Test the config syntax; sudo nginx –t -should return syntax ok
9. Restart nginx to use the new config; sudo systemctl restart nginx
10. See if the service is running with no errors; sudo systemctl status nginx Should be green active running
11. In a browser on your computer test this address;
    a. http://serverip -if this fails to open or redirect you to HTTPS check to see if nginx is running and config loaded
        i. Sudo systemctl status nginx and check config with sudo nginx -t
        i. Also check; sudo ufw status
        ii. Also check to see if WebCTRL is running
12. System logs for nginx are stored in; /var/log/nginx/*.log
    a. Sudo tail -40 /var/log/nginx/error.log -show last 40 lines of the nginx error log
    b. Sudo tail -40 /var/log/nginx/access.log -show last 40 lines of the nginx access log
        i. 301 is a permnant redirect so you should only see a browser hit the access log at the start of a new session. Some browsers may even remember a site was https and when you type in the http URL they might go straight to https without even trying http 80 first.

DNS changes

In EasyDNS for harrisisi.com domain make two DNS A records pointing to the public IP address of the linux;

1. az##.harrisisi.com -this is for the VPN clients to connect to, and if we lock out the customer due to nonpayment service can use this address to get into the server just in case.
2. customername.harrisisi.com -this is the one we give out to customers and document on the tech site.
3. In a browser both az##.harrisisi.com and customername.harrisisi.com entered plainly should be automatically upgraded to HTTPS with a valid SSL certificate with no page warnings. If these don't open but everything worked in the previous steps with the servers private IP have IT check to make sure the Azure firewall is open and forwarding TCP 80 & 443. If you get a certificate warning you need to change to the *.harrisisi.com wildcard certkeys file.

OpenVPN install and generate keys, new seperate doc. Modify existing windows doc.