THE EVERGREEN PROJECT

LIBRARY SOFTWARE CONFERENCE

GROW YOUR OWN EVERGREEN! HACKING, AND TESTING ON YOUR VERY OWN TEST SYSTEM

1:00 PM – 2:00 PM EASTERN

MAY 26, 2021

CAPTIONING PROVIDED BY:

CAPTIONACCESS

contact@captionaccess.com

[www.captionaccess.com](http://www.captionaccess.com)

\* \* \* \* \*

This is being provided in a rough-draft format. Communication Access Realtime Translation (CART) is provided in order to facilitate communication accessibility and may not be a totally verbatim record of the proceedings

\* \* \* \*

>> GINA MONTI: We will get started with the session. Grow your own Evergreen! Exploring, hacking, and testing on your very own test system. Thank you so much, Michele, for coming in to present this. I will be moderating the session and before we get started I wanted to talk about our sponsors. So thank you, Evergreen Community Development Initiative for sponsor for Hopin and Mobius is sponsor captioning for this track so there will be a link I will post in the chat and that will be available to you. If you have any questions for Michele pop them in the chat and we will make sure she gets your questions. This is a recorded session so you will be able to watch post conference. Whenever you are ready you can go ahead and start, Michele.

>> MICHELE MORGAN: Thanks, Gina. I will start by sharing my screen. I hope you can see my first slide. Welcome to grow your own Evergreen. My name is Michele Morgan, I am the technical support analyst at NOBLE, the north of Boston Library exchange. In Massachusetts. We are a consortium of public and academic libraries. We have about 25 members.

Why do you need your own Evergreen? Well, maybe you don't really need one but you make it to the point where you feel like you need one like I did. You might want to do some test driving and learn about the system. There might be some functionality that you want to explore. To see if you can use it for your system. When a new release comes out you might want to take that for a test drive. My main interests were working on launchpad bugs, testing and confirming launchpad bugs, signing off on them, and eventually, hopefully, to fix some.

The Evergreen community often provides test servers for people to log into. There are community demo servers but you don't really have full access to those and you have to be mindful that you are sharing those with other people. You don't want to interfere with what other people are doing on them and you don't what they are doing to interfere with what you are interested in testing. One of the good reasons to grow your own system is it is yours. You can get under the hood, you can try things you wouldn't on other systems that you use regularly or have access to. You wouldn't want to just try things on the production system, not a good idea.

You can break it, it won't bother anybody else. And if you do break it you can't just throw it away and start over. And it is a great learning opportunity to get into the code come into the system, look around and play with stuff. So what skills do you need to build your system? Well, that depends on your goals. There is the NewDevs Page has been mentioned a few times during this conference and it is a great place to start. Here is the link to it on the slide. You can click on it from the presentation. It is a great place to look for tools for if you are inexperienced with working with servers in Evergreen. A shout out to Taryn McCanna who started the group and is the head of the group and she is very organized and puts together great programs and records great information off of this made.

So it is a good place if you are unfamiliar with some of the things we are talking about, start here and you will find some good information. A few specific skills which are good to have, being familiar with the command line interface is really important. Because a lot of work when you are building your system is on the command line interface so just being comfortable there is a good thing. To get some experience. Navigating Linux filesystems is a plus. Some knowledge of SQL, especially if you are going to get into really digging into the system to see how it works. A little knowledge of IP addresses and basic networking is a good thing. Many of the problems that I had along the way were because my wireless network was not always happy about giving me an IP address for my server. So I would lose connectivity. It is good to know how to recover from that. And knowledge of get mac for testing, signing off, and fixing launch pad bugs.

How do we build our system? We could do it the traditional way, find a machine, install an operating system on it, Ubuntu or Debian or then we could follow the installation instructions that are always on the Evergreen download page where the new releases are posted. So if you follow that link you get something like this. But don't read it too closely because that is not what we are going to do. What we are going to do instead is billed a virtual Ubuntu server on our desktop or laptop. For me it is the same machine that I use to do my daily work, it does not have to be.

Then we are going to install Evergreen on the VM, virtual machine, that we just built using scripts that are made to do all of the work, all of the hard work. Why do we want to use a virtual server? Because you can run it on your workstation, you do not need a special piece of equipment. We will talk a little bit about how special it needs to be. Low overhead. The virtual machine software, you install it on your system, you run it like any other program so in that sense there is low overhead. You can start it up when you need it; you can close it when you don't. One of the things I really like about it is that you can save the state of your server. If you are in the middle of something and you get to the end of the day and you need to close it down you can say freeze this just the way I had it, close it down, and when you bring it up the next day it would be in exactly that same state.

As it was when you left it. You can build multiple servers. I certainly had trouble having more than one open at the same time. Mostly because IP address conflicts, I think. But you can work on one, save that, have another one that you are doing another project on. So you can have multiples that exist at the same time but are not running at the same time. And virtual servers are disposable in a good way. The servers will come to the end of their life, especially when a new release comes out. That is a good time to throw it away and build a new one.

Another good time to throw it away and build a new one is when you break it. Which I have done many times. A note about the workstation that you build your virtual servers on. It needs to be pretty powerful. Currently, I have a laptop running Windows 10. It has and i7 processor, 32 gigabytes of RAM. Which is really nice. My workstation was provided by NOBLE and I got it sometime during this long year when who knows what happened to time but it was an upgrade from a laptop with an i3 processor and eight gigabytes of RAM. Which I was running virtual machines on but it was a little too much for it and moving into this more powerful laptop was like moving from a tiny studio apartment to a 4,000 square-foot luxury house, so it was really nice.

What will we need to build our machine? You need a platform for your virtual machine. I am using virtual box. There are others. You need an image of the Ubuntu server that you will be installing on your virtual box. And you will need a script to do all of the work of installing Evergreen. I am going to use the ansible script. I will talk about another script as we go along, which is also a possibility for you.

The first thing we want to do is install virtual box. The link is on this slide and you can't install -- you can get it for Windows comma, Linux, other operating systems should you choose the one that matches your operating system and install that. The script that we will be using to install is the ansible script, which was built and maintained by Bill Erickson, shout out to Bill Erickson. And other community members have contributed to it, too. It's on get help and the link is available on this slide.

This is a snapshot from the page and note that there are different scripts for different versions of the Ubuntu operating system so you need to make sure the script matches the operating system you are installing. And throughout this presentation I have screenshots that show different versions of the operating system because things change. When I first started everything was on Ubuntu 1804 but Ubuntu comes out the new version so the script gets updated to the new version.

So you just have to be sensitive to these versions that you are working with. I did want to show -- this is a snapshot from the readme for the ansible script. Step 1 is install Ubuntu 2104. Step 2 is when you pull down the ansible script using git and this short list of about five or six commands is all you need to execute on your virtual machine to install Evergreen. I also wanted to mention the docker script. I have not used the docker script much. I started testing it initially on my old laptop but something bad happened and I moved on to ansible.

My impression is that if you have less experience you might want to try docker. On the screen from docker -- made me think maybe docker isn't stable enough for what I wanted to do, which was test launchpad bugs and sign off on them. If you want to explore Evergreen, docker might be something good to try. The next piece we need is the Ubuntu server installation file, which you can get from the Ubuntu website. Down here is a link to 2004, which is the current version. For now. When you download that to your workstation you will get a file, and iso file, that you will use when we start up virtual box.

Let's grow and Evergreen Expo. I am not a system admin. I do have a lot of experience crawling around the Evergreen system. I take a lot of support calls, which prompts me to find out what is wrong so that we can file bugs and hopefully fix them. There was a lot of trial and error when I was developing this process. I must have tried building probably a dozen or so machines. Some of them worked; some of them didn't. And there may be better ways to do some of the things in this process but it works and it works for me but I am happy to learn easier ways to do things.

I did find that a lot of my issues when things failed were those IP address issues and I suddenly could not connect to my server or when it was building it could not get out to the Internet and pull down all of the files it needed. We are going to fire up virtual docs. When you first fire it up after installing it this panel over here will be empty because you have not created any machine yet. Your created machines will live here. These are your machine commands across the top. The first thing we are going to do is click on new to create a new machine and it will start prompting us for some things. We will need to give it a name. It will supply a folder to store the machine. The default is fine unless you want to change it.

Mostly the defaults on the screen are fine but there are some that you are going to need to change. For example, the memory size -- this is a slider, I usually slide it to around eight gigabytes. When I did smaller ones Evergreen did not run very well or at all. You do need to make sure that you don't give your virtual machine more memory than your actual physical machine has and you need to need leave enough memory for your physical machine to operate while your virtual server is running.

So since I have the luxury of 32 gigabytes giving my virtual machine 8 is no problem at all. We are going to create a virtual hard disk now and click on the create button at the bottom of the screen. That will give us another prompt. File location, I accept the default. Another thing, there is a slider here to give your hard disk plenty of space. I have run into problems why did not have enough space for Evergreen previously so I usually go up to about 40 gigabytes and I have not had a problem with that.

Again, don't give your virtual machine more than your extra machine can afford. Then we click create and we go back to this screen. One thing I have discovered that I needed to do to be able to connect to my machines was go into this setting tab and then go over to network and for the network adapter it defaults to NAT and when I change it to bridge adapter I had much better luck. So that is what I did. Then we go back to our virtual machine controller and we click on the start button.

It is going to ask us a couple more things. This is where we choose whatever iso file we downloaded from Ubuntu, the server iso file. This screen shot was from 1804. The previous screen shot was 20 so again, just be aware of your version numbers. When we start it is going to start giving us these command prompts in a terminal window. And to navigate through this window you can use the arrow keys, up and down arrow keys, a space will select the box if you need to check something and the enter key are usually -- press the enter key to move to the next screen. Most screens you will just be continuing to the next screen. There are a few screens -- there are a lot of screens because you have a lot of options when you install Ubuntu but we don't have to worry about many of them for this.

This screen make if you pause, and I apologize, it's hard to read but what it says is confirmed destructive action, which means it is going to format the disk space that you gave to it when you set up your virtual machine. It is not going to kill your actual machine. Everything in this relates to the virtual machine but it gives me prompts, do I really want to do this? Yes, I do. Once you do that you will give it some information. It wants your name, you need to name your server. It does not have to be the same name as you gave it an virtual box but I find it is less confusing if I do.

You need to pick a username and a password that you will use to log into Ubuntu when it is built. Then we arrow down to done and press enter. The other set up piece that I like to choose is to install the open SSH server so you put an X in that little box there. This is useful if you want to connect to your server with your favorite text editor or some other kind of development application that you like to use. You can do all of your work in this terminal that virtual box shows you but it gets a little hard on the eyes after a while so I always install that.

At some point a whole bunch of stuff will scroll across the screen. A lot of this process the virtual machine is very busy scrolling stuff across the screen but you can just sit back and watch it. Notice up here it says finished install and now we arrow down and we hit reboot. Again, I hope bunch of stuff will scroll across the screen and eventually we will get to a prompt that says log in. You can log into your -- using your username and password that you credit before and you get to a screen like this. We are now logged in.

One thing to notice on the screen, this server has gotten an IP address from your network so right here you can kind of read it. I typed it out over here. It's 10.0.2.15. For this machine. That might be different the next time you boot up your machine. I know you can probably give it a static IP address. But I don't -- I don't have as much knowledge as I would like to really do that [indiscernible by captioner] at this point you have built Ubuntu system. Are halfway there.

I mentioned before it's the end of the day and you want to savor your success for the end of the day. If you hit the X in the upper right-hand corner of that terminal window, that virtual box gives you, you will get this box and you can power off the machine, send a shutdown signal, or save the machine state and I almost always save the machine state unless I am doing something that requires a restart. So I save the machine state so it will be exactly where I left at the next time I open it up.

Let's move on to installing Evergreen. Just a note about users on your system. We already talked about this user, it is the one you created when you built your VM. The root user is the all-powerful user. You will need to do some -- execute some commands as that user. And the OpenSRF user, we don't have that yet. That will be created as part of the Evergreen installation process but once we run that that is the user that owns Evergreen so many commands you have to run need to be run by that user.

This is a cheat sheet for changing users on your virtual box. When you log into your server you are logged in as your Ubuntu user. You have to get to the root user to get to the OpenSRF user. You could probably -- not probably, definitely, you could assign the OpenSRF user password that you know that you would log in directly. But knowing this process is useful. So you execute this su do SU root command. You enter your password and that will get you to the root user. SU-OpenSRF will get you directly to the OpenSRF user. If you exit from the OpenSRF user you will go back to root and if you exit route you go back to your Ubuntu user.

From the ansible readme file these are pretty much the commands that you need to install. You just type them in, you wait until you get your command prompt back, sometimes it takes a long time. But eventually you will get there. This last command right here is what actually invokes the script, which is programmed to do all the tasks that will install your Evergreen. Here is a couple snapshots of the ansible script running. Again, it will take a while, it will keep scrolling and it will test things but eventually you will get your command prompt back and that means it is done.

If you take your IP address that you recorded before and go in your browser to HTTP://that address/HD4/staff//you should get a log in. Notice I have a different IP address showing on this machine. I was on another network for the screen shot, that's why it has a different address. The default password for your system, for your test system is admin demo 123, you might be familiar with that password. From systems that are built for bug squashing day. So that one gets by default.

If I log in -- I have to register my workstation but I will eventually get the Evergreen screen and you are in. You can also connect to the OPAC just with http:// and the IP address. I built a new machine right after 37 came out and it is the Boostrap OPAC which is the default in 37 and my bootstrap looked weird so I asked IRC and it turns out there is a little missing command in the ansible script for the new Boostrap OPAC and thanks to Jeff Davis these two simple commands fixed it. So our Evergreen system has the stock database. Also known as the concerto database. There is information on what is available for logins, users, some information on the records that are in that database at this wiki link.

So definitely it is useful to bookmark that link because if you have to test something and you need a user or you need to know what has parts or what has serials, that is a good place to go to find out what to look for. Congratulations, you now have an Evergreen system. Before I go on are there any questions? I am not sure how we are doing for time? I guess we're doing pretty good.

>> GINA MONTI: We are at 1:30. We have 20 minutes. Plenty of time for any discussion. Just as a reminder this is a recorded session so our documents and links and so forth I am sure will also be shared.

>> MICHELE MORGAN: I have a few more things to show so I will continue on. Feel free to ask questions as they come in. I see there was the question about docker on Windows and that is what I did try, installing docker on Windows and I am going to blame my old laptop that I did not get it to work. So give docker a try if you want. I like to do a few extra things beyond the plain install. One of the things I like to add is [indiscernible by captioner] which is a GUI interface to the Evergreen database. I crawl around the database all of the time. I think it's a great weight to learn the system.

It's great that the community has test systems available and Taryn does a great job organizing test systems for bugs squashing day but you can't get to the database in most systems and you can't log into those systems to see what is going on the code. I always install PHPPG admin and I wrote myself a cheat sheet. All of these cheat sheets, pretty much all of these cheat sheets I got from Google. How do I install PHPPG admin here's how I do it. So the commands are here. And you browse to that IP address with PHPPG admin on the end and the default username and password for your Evergreen database is Evergreen.

Another thing I like to do, especially since I want to get to the point where I can pull down patches from the working Evergreen repository so I can test launchpad buses to set up git. Setting up git for the first time is a little bit involved. I wrote a couple commands here, the git config to tell who I am in my email address but there's a lot of good information on the new depths page regarding how to set up git and getting -- you need to center git keys to the git admins so they can give you permission to post your branches or your sign offs to the working repository for testing launchpad bugs.

I do have a cheat sheet for adding the working repository. When you use the ansible script to install your system part of it it adds the Evergreen git repository, which is where the [indiscernible by captioner] code lives to your virtual server. When you install that you are installing a master system, which is the up to date code for Evergreen. When releases are cut like three got seven is a snapshot of the Evergreen code add to the point where the releases cut. A few patches may have been added and most likely were added since a releases cut.

So when you build your system it is grabbing a snapshot of master at the time you build it. So it might be a little bit ahead of the last release but that is not a bad thing. You just need to be aware of that. So your server already knows about the Evergreen repository. What it doesn't know about is the working repository that developers push their workbook fixers or community members push their working branches to. When they propose a patch for a bug.

So these commands will add to the working repository so that you have access to the Evergreen repository and also the working repository so they can pull down those patches to test. One thing I like to do after I get the system set up the way I wanted, I don't touch it. I don't touch that original system that I just built. I immediately make one or more clones. Making clones is easier than building systems so I leave my system that I built just the way it is and I make one or more clones that I can mess up. To my heart’s content. And I can go back, always go back to that system I just built and make another clone. It just saves a lot of time.

Clone is an exact duplicate of the system you built. You always have that pristine system that you can go back to and recline. If I want to make eight clone in virtual box it looks like I cut off the menu on the screen shot, but there is a machine option if I click on that I get an option to clone. I have my machine is selected here, it was a blue two oh nine that I just built. So I click on clone and it asks me -- I can give it a new name by default it's going to say clone of whatever I cloned.

And I don't know -- I always choose this generate new mac address for all network adapters. I don't know if I have to do that but it worked when I did it so I keep doing it. If I go next it's will ask me what type of clone I want and I always do a full clone because my goal is to leave the system that I just built exactly as it was. If I did a link clone any changes I make to the clone will also affect my original system, which I don't want to do. When I say clone we get the little sheep who is making -- that is my clone, I guess. This doesn't take very long at all. It's maybe takes a little more time then starting and stopping your virtual machines but it really takes definitely under a minute, probably under 30 seconds.

When it is done we have another machine that is all built, already to go, it is a clone of the one I built. Initially. I have a few more cheat sheets to share. I like cheat sheets. When I have eight $that means you are and OpenSRF user and when I have the # it means you are the root user. I mentioned before I've had trouble with IP addresses on my servers. So I find I can use this IF config command and it will show me the IP address that my server has. I can force my server to get a new IP address with these two commands. This TH client commands. I have a cheat sheet for restarting Evergreen. Sometimes when you pulled on a patch depending on how involved the patches if it touches a pro module you may have to pull down your patch, get it on your system using git, put into place one way or the other and restart Evergreen so that you are running Evergreen you will see the changes in that [indiscernible by captioner] module.

Angular files. If you are just starting out with this angular files are probably a good place to start because you don't have to stop and start Evergreen to see changes in angular files. You don't have to rebuild your system to see changes in angular files. You just need these commands, which will -- this first one will navigate to the place where the angular files live. The second one, the NG build--prod will compile the squads and then the third one will take those compiled files and copy them into the place where the running Evergreen system looks for them.

You don't even really have to do a hard refresh on your browser for the Evergreen files. Hard refresh his are always a good idea but usually just [indiscernible by captioner] let you see this changes. My favorite cheat on my server is the up arrow command. Some of those commands are pretty complicated and if you have typed them in once they will be in your history so you can always use your up arrow to find a command that you have done before and reexecute it.

My goal for all of this is to be able to test bugs and sign off on them. I just wanted to go over the process that I used to do that. The first thing I do is I start my VM. I navigate to clone OpenSRF Evergreen as the OpenSRF user. And that is the directory that git knows about or that knows about git so if I'm doing git commands I have to be there. I usually do a git pool which will go to the Evergreen repository and pull down the latest master code.

That is not always totally necessary but I always do it as a matter of course was always up to date. The next thing I do is a git fetch working, which will pull down the working repository which has all of the branches in the launchpad bugs that you want to test. If I want to test a branch I will create a new git branch, I usually using this git check out minus b command. I usually use a command that is the bug number, that just works for me. And if I put the origin/master at the end that means the branch that I am creating is tracking the master system.

If what you are doing takes a little time, like a week or more, or if it's especially active with code being committed to master you can reconcile, you can know is reconcile with your branch with the current version of the master system if you track it. That is probably not a very good explanation but it is the hand waving one. The next step is to find your bug, you need to do a cherry pit. The minus s is your sign-up. You don't know they will sign off at this point on the bug but if you do you have a signoff there and you don't have to worry about it.

Then you need the first seven characters from the big long committed stream. To pull that patch into your git branch. The next thing you need to do is make sure that the patch that you pulled down gets to the place where you are running Evergreen systems using it. Depending on how complicated the bug is it could involve something simple like copy the patch from one place to another on the server.

We talked about the angular files already. It could be just around the commands and compiled the angular files and copy those into place. If it's a very complicated patch you might have to rebuild Evergreen. My system manager Martha Driscoll gave me steps to do that once and I did it once but I don't have a cheat sheet for that here. Sometimes you will need to restart Evergreen. I had a cheat sheet earlier. It so is a good idea to restart Evergreen to make sure that it is current. So then you can log into your system and test the patch.

When you test patches you really want to test them. You need to consider permissions, library settings, global flags, what the staff sees, what the patron sees, and different workflows. If it's a simple bug like this column is labeled wrong, that is a simple thing to test. Just keep in mind that there can be a lot involved in testing. So if everything looks good you can use the git push command to push your working branch with your signoff up to the working repository and then you'd update the launchpad bug and, hopefully, at a signoff tag -- you will at a signoff tag and hopefully very soon a computer will review it and decide whether it is appropriate to merge it to master.

So that is the process. And that is pretty much all I have so I am happy to look at questions.

>> GINA MONTI: We had one that seemed to be answered by the chat but I figured I might ask it anyways. You put your keys on your VM 2?

>> MICHELE MORGAN: Idea. Before I started VM's [indiscernible by captioner] it's not run your system. I do copy my keys from git batch to my virtual server so I don't have to ask the git admins to constantly give my new case permission. I didn't have a cheat sheet for that. It's a little bit involved but that's why didn't include here. But yes, I do do that.

>> GINA MONTI: Are there any other questions for Michele? There's a few more minutes before we go on to break. Feel free to throw them a check. Or as always, you can go into the open discussion forums during the break, too. To continue the conversation there.

A lot of thanks from people, thanking you for giving your time to us today.

>> MICHELE MORGAN: Thank you. I'm usually in IRC or wherever IRC is when to move to. It's a great resource being in IRC because you can just ask a question and somebody will answer it.

>> GINA MONTI: I will give a testimony to that. I have been wanting to set up and test a machine as well so I will definitely be taking your notes. With that, thanks, everybody, for joining us and we do have a break until 2:30 in which there be lighting talks in track one afterwards at 3:00 we will resume the regular schedule for the sessions. Again, this is a recorded session. I'm assuming you're going to also provide the slides as well as a document?

>> MICHELE MORGAN: Yes, I will definitely do that.

>>: GINA MONTI: Awesome. We will keep you all abreast by that by email. Thanks again so much.

[ end of meeting ]