So. Whichever you prefer, you can actually use going forward. And lastly, what I'm going to do is add my user to the Docker group so that this user can actually use Docker as expected without any restrictions. If we just SU to ourselves. We enter the password and again we can execute Docker, run it for interactive RM to cleanup Ubuntu's the image and bashes the command and you can see there we're in the container now as the host name has changed. And let's just do a courtesy CAT/EDC/OS dash release before we exit. Edit for your corresponding desktop, you should now have Docker installed. You should also have Docker Dash compose installed. If you installed Docker for desktop, you will automatically have this and if you followed the steps for Linux, we installed this manually. Join me in the next video where we're going to be installing. The Ansible lab. Hello and welcome to this video installing the Ansible lab. With this course we have a fantastic lab experience that will allow you to access Ansible and a number of guests either in your web browser or via SSH directly from your system. The environment for this course is hosted on GitHub. If you navigate to the following URL, you'll see the code and within this are detailed instructions for the setup. Given the variances in operating systems, I will be keeping this page up to date, so I encourage you to read through the entire README just in case there are issues that have been encountered since this video was published. We're going to go through an example setup. I'm using Mac OS X, but this process has been proven to work. On Mac OS X, Windows and Linux, there's some very slight variations for each OS, and I'll be cover. That in the video. So firstly, let's take a look at the repository. OK, so this is the repository as I actually mentioned. You can actually see here we have detailed instructions of how to actually set up the course lab. Some things I want to actually bring your attention to. So we need to have both. Docker and Docker compose install. If you went through the previous videos, you should have this set up already. I recommend that you download and configure this to the dive into Ansible lab folder within your user Home directory. If you happen to be on a Mac or a Linux system, or even a Windows system which has git installed, you should be able to just do a git clone of this repository direct into your user home direct.

Oh.

If you don't or you actually prefer to download the repository, you can do so at that URL listed there. This validation section is of particular importance and after you've cloned the repository or you've extracted it via the zip. Ohh please go through and actually check that all of your paths tally up with those listed for the corresponding operating system. You can then run the lab by running Docker compose app from either your command prompt or your terminal directly from the dive in to answer. Lab directory. If this goes well, you should then be able to browse in your browser to Localhost 1000 and you'll get the portal interface. This periodically in the future I will be updating the images as new versions of Ant support come out. If you wish to actually update those images, you can quite simply do a Docker compose poll. If you want to refresh the lab at any point, you can do a Docker compose RM and the next time you do an. It will actually refresh the lab as we're using volumes for the home directories. Your data that you actually have there should be person. OK, let's see this in action. I'm going to scroll back up to the top and because I'm on a Mac system, I'm going to use Git, so we go to code and I'm going to copy this, but if you wish to, you can do the download zip here and I'm going to go to my terminal. If I do a PWD, you can actually see I'm in. Slash users slash James and what I'm going to do is a Git clone on that repository. If you wish to, you can download the zip file and unzip this and you'll be in exactly the same situation. OK, great. So that has done that. Now we can change directory into dive into Ansible lab. OK. And what we should be able to do now is a Docker compose app. OK, brilliant. So that's working. And now if we actually go to our browser, we should be able to browse to Localhost 1000. OK, great. And then we've got the actual starting page and then go across to and support terminal. Click Ubuntu C. And at this point you should be able to log in as Ansible. With the password of password. Perfect. The course also provides the ability to be used for free from Google Cloud Shell. To do this, navigate to diveinto.com/P/playground. And click the button there. Now at this point you can actually choose to trust the repo or not. It's literally just some markdown files, but either way it will work as trusted or not trusted. By default, cloud Shell will be running in ephemeral mode, so the data will be lost as soon as the session is closed. This is fine if you want a quick throw away Ansible environment, but to have your changes saved, switch off a femoral mode. Now we can follow the tutorial on the right hand side. And if we follow through the remaining steps. And then we start the lab. This takes a moment to start the environment. We can actually click this helper here which would actually show us what icon to actually use. So instead I'm just going to go up to this icon at the top and from there we can click preview on Port 8080 and now you have the lab environment ready to go.

And.

Running in Google cloud shell. I hope you enjoyed this. Please join me in the next video configuring SSH connectivity between hosts. In our previous video we configured and set up our lamp. We're now going to configure SSH connectivity between hosts and table is an agentless architecture and therefore for connectivity to take place between Ansible and our targets, A trusted relationship is required for automated. Passwordless connectivity. What happens when we run SSH between a client and a server? If we look at the example here, I show that we're logged in as the Ansible user and I've entered the command SSH Ubuntu One, IE open a secure shell from Ubuntu EC2 Ubuntu 1. And we press return. The following will happen. An SSH session is requested. Both the server Ubuntu, Dash C and Ubuntu One exchange the SSH protocol versions that they support. If they're compatible, they agree, otherwise the connection will be stopped at this point. Then a key exchange takes place with the cryptographic primitives that each sides. Ports after they've agreed on the cryptographic primitives they negotiate a session key using the Diffie Hellman algorithm. It's at this point where some of you who've used SSH before will see this familiar message where it's asking us to verify the target fingerprints. We accept this. And this is stored to the clients known host file. Now their session is in place, new keys are exchanged between the client and the server and going forward both sides use the new keys. Finally, we have an encrypted session and we're ready for authentication, and if you were to try this, we'd see a request for the password from the Ubuntu 1 system. Watch the answer for Ubuntu icons carefully. On this video I've given both the client Ubuntu C and the server Ubuntu 1A glow to indicate the connectivity is established as we go through this video, when the icons glow in the same way, the server and client have completed the steps we've outlined and have established a secure connection. Note at this point they have not authenticated, they've just established a secure channel. We take a look and do this in our lab and we enter here LOCALHOST 1000 and from here we're going to choose the Ansible terminal. From there select Ubuntu etc. And then we enter the username of Ansible. And the password of password. We're now in the terminal of Ubuntu C Let's try opening an SSH connection to Ubuntu 1. We accept the fingerprint. This will write this fingerprint to Ubuntu C's known hosts file. We now have a secure channel and we're being asked for the password, so we're at the authentication stage. If you enter the password, which is quite simply password. We're now logged in to Ubuntu 1 and if you type exit here, you'll go back to the answer for host. Let's take a look at the known hosts file. SSH information is kept in a hidden directory called dot SSH. The dot means that you won't see this with a standard LS, but. If you were to list with all information, you would see this. We can then CD to change directory to this location and if we do an LS here to list. Falls. We can see how known hosts fall. We can then use the Unix command cat to show the contents of this file. If you're new to Unix, it's called cat as it also provides the functionality to concatenate files. We're just using it to show files here. This at the moment doesn't appear to be too friendly and we can actually see two entries even though we've only accepted one host key at this point. The reason being when we accepted the fingerprint, it captured the fingerprint for both the host name and the IP address. We can prove this is the case by generating the fingerprint ourselves with the SSH Keygen command. If we do this firstly for the host name. We can see that this entry tallies up with the first entry. If we ping Ubuntu one to quickly get the IP address. And press control C to exit. We can now run the same command against the IP address. And near the second entry tallies up with the second line in the known hosts file. If we were to delete the known host file, then attempt to reconnect again, this would once again ask us to verify the fingerprint. We'll just accept the fingerprint and then press central C and now we exit from here. We've got a good understanding of SSH connectivity, so let's have a look at how this relates to Ansible, where we would like Ansible to be able to connect and manage our systems without the use of passwords and support uses SSH. So to resolve this we'll be implementing this at the SSH layer. What we just did with the password is in an equivalent of the following. Our secure channel is established. We then enter the password and this is sent securely through the channel to our target Ubuntu 1 and once it's validated. We have connectivity. But we don't want to be entering a password like this every time, So what we do is use SSH to create public and private keys and then those keys can be used for access. We're going to talk through the theory, and then we'll do this in our answerable lab for all of our hosts on our Ansible control. Host Ubuntu. C We'll be creating a key pair comprising of a public and private key on our target hosts within our dot SSH directory. We can utilize what is known as the authorized keys file, allowing us to add the public keys of our trusted hosts. With our secure SSH channels established, note the globe between our entities when authenticating the private key of Ubuntu C will be you. And is verified with the public key on Ubuntu 1-2 and three, as these tally up our connection can now be established. Hence we have green arrows. Let's put this into practice on our hosts. Let's log in again as Ansible and we're going to use the SSH. Keygen command. We can accept all of the defaults. We can now take a look at the generated files, both of which will be in the SSH directory. Firstly, let's take a look at the public key. You can see by the end that by default it's added the username of our host, which in our case is Ubuntu C This is the key we can share with others. The private key has the same name but without the dot pub on the end. We could technically copy the contents of the public key to the authorized keys file on a remote system. However, if we were to do this manually, we'd also need to make sure that the permissions of the authorized key file and the SSH directory are. Correct. Otherwise switch will reject it. There's a much more efficient way to do this. This is with the tool bundled with SSH known as SSH copy ID. Let's use this to copy our public key to the remote system. The format is the username in our case. Ansible at the remote system. We entered the password of password. And this has copied our file if we wish to, we could now SSH to the remote host listed using our private key. Therefore no password prompt. For now, we'll just exit from this and go back to our answerable host. Your notes we didn't get asked for the fingerprint as we've already accepted this, but if we were to try another host. We enter yes to accept the fingerprint and we enter the password and this is now set up for our second host Ubuntu 2. We want to copy our public key to all of our target systems as both the answerable and root user. We could do this manually and enter yes for the finger. Print and type the password in each time for all of our hosts, or we could quickly automate this given that this is a course on Ansible and efficiency, let's automate to address the requirement for having to enter a password. We'll install the program SSH pass. This is a Ubuntu system, so first. Let's update APT using sudo and if you're not aware of pseudo, this means super user do. We enter our standard password, which is quite simply password as this account is already configured with pseudo access. Now their APT is updated. We can install SSHD pass. Brilliant. Let's create a password file called password dot TXT with our password in. We'll echo the password and redirect the output to a few. Let's do a series of four loops for the variations we need. We have two users we wish to target Ansible and Root 2 operating systems, Ubuntu and CentOS, and three instances Ubuntu 1-2 and three and CentOS. 1-2 and three. So first of all, we'll do both of our users. We'll do both of our operating systems.