

- 1 [Description](#)
- 2 [How it works?](#)
- 3 [Install Guide](#)
 - 3.1 [Ansible Acsenv](#)
 - 3.2 [VirtualBox 6.1](#)
 - 3.3 [Vagrant](#)
 - 3.4 [Docker \(not need for Virtual Machines only setup\)](#)
 - 3.5 [pyenv \(recommended\)](#)
- 4 [Setting up a Development Environment](#)
 - 4.1 [Vault Pass](#)
 - 4.2 [Virtual Machine](#)
 - 4.2.1 [build](#)
 - 4.2.2 [provision](#)
 - 4.2.3 [destroy](#)
 - 4.3 [Docker Container](#)
 - 4.4 [Appendix: Accessing you VM through ssh](#)
 - 4.4.1 [Accessing from host through private network](#)
 - 4.4.2 [Accessing from other Host in the Network through public IP](#)
 - 4.4.3 [Accessing using vagrant ssh](#)
- 5 [Appendix: Using RDP to connect to VM to display GNome Desktop](#)
 - 5.1 [Microsoft Remote Desktop](#)
 - 5.2 [Remmina Remote Client \(Linux\)](#)
- 6 [Appendix: Using VirtualBox GUI to display GNome Desktop](#)
- 7 [Appendix: How to connect to a VPN using OpenConnect](#)

Acsenv allows you to deploy acs development environments with ease. It can set up:

- one or more virtual machines, all of them connected to the same internal network
- one or more docker containers, all of them connected to same network

Virtual Machines requires more computers resources. At a minimum, each VM should be at least 4GB (4096) of memory, to allow for ACS to run. So be careful that the amount of VM resources matches the ones available on you local host. Though VM are slower to spin up and consumes your resources, they are great to provide isolation and are able to truly build up the final environment your application will run. Along with it, VM are able to display your graphical applications, so anything related to GUI a VM is your tool.

On the other hand, lightweight development environments can be created through Docker. That setup comes handy when you need to edit files and perform compilation from command line. Docker containers are great to run your TAT tests. Your development cycle can be speed up through the use of containers.

The following operating systems are supported:

- Mac OS X Catalina
- Linux Ubuntu

This tool has been tested in Ubuntu 20.04 and Mac OS Catalina 10.15.6

Often, when it is time to develop code for ACS, you end up trying to mixed up these elements:

- The ACS Distribution (acs_binary) with all the base system already compiled. You want to develop code on top of that release.
- Your git local repository (acs_repo)
- A working repository with patches (INTROOT) to be shared between nodes. (acs_working)

All these elements are not suitable to be stored inside the VM or Docker container, because of the space they use (Several GB) and that we want to recreate the environment several times during the development cycle. For that reason, that elements are kept inside the local host and they are shared with the environment through:

- NFS (network File Sharing) between the VM (guest) and the host
- Docker Volumes (between container and host)

Both VM and containers are provisioned through Ansible playbooks.

Ansible Acsenv


Ansible Acsenv can be cloned from:

https://bitbucket.sco.alma.cl/scm/~pburgos/ansible_acsenv.git

VirtualBox 6.1

Download Virtualbox from official site:

<https://www.virtualbox.org/wiki/Downloads>



VirtualBox

Download VirtualBox

Here you will find links to VirtualBox binaries and its source code.

VirtualBox binaries

By downloading, you agree to the terms and conditions of the res

If you're looking for the latest VirtualBox 6.0 packages, see [Virtu](#) in 6.1. Version 6.0 will remain supported until July 2020.

If you're looking for the latest VirtualBox 5.2 packages, see [Virtu](#) Version 5.2 will remain supported until July 2020.

VirtualBox 6.1.12 platform packages

- [Windows hosts](#)
- [OS X hosts](#)
- [Linux distributions](#)
- [Solaris hosts](#)

[About](#)
[Screenshots](#)
[Downloads](#)
[Documentation](#)
 [End-user docs](#)
 [Technical docs](#)
[Contribute](#)
[Community](#)


Vagrant

Vagrant allows you to script the Virtual Machines, so that all tasks related to create and destroy VM can be done through files and command line. Along with that, several tedious steps like setting up a NFS Share are greatly simplify using Vagrant.

Download Vagrant from the official site:

<https://www.vagrantup.com/downloads>

And follow the install instructions for your Operating System.



Intro Docs VMWare Community

20.2k

Download


Download Vagrant

[MAC OS X](#) [WINDOWS](#) [LINUX](#) [DEBIAN](#) [CENTOS](#)

These are the available downloads for the latest version of Vagrant (2.2.9). Please download the proper package for your operating system and architecture.

» [Download VMWare Utility](#)

Continue learning with step by step tutorials at [HashiCorp Learn](#).



Mac OS X
64-bit ▾

Download

Current Version: 2.2.9
[Download Older Versions of Vagrant](#)

[SHA256 checksum \(2.2.9\)](#) [Changelog](#)

Docker (not need for Virtual Machines only setup)

pyenv (recommended)

pyenv is a tool that allows you to isolate python and their packages, so that you keep you deployment clean and reproduceable. It's likely your already use this tool, so my recomendation is to create an environment for python and activate it before running pip, the command that follows this paragraph.

The python version used was 3.5.7

Python required packages

Install python requirement with pip

```
pip install -r requirements.txt
```

Create .acsenv in your home folder

```
mkdir $HOME/.acsenv/
```

create yml file boxes_config.yml. An example can be downloaded [boxes_config.yml](#)

```
---
memory: 4096
cpus: 2
gui: false
boxes:
  - name: acs01
    ip_private: 10.10.10.10
    ip_public: 192.168.1.155
  - name: acs02
    ip_private: 10.10.10.11
    ip_public: 192.168.1.156
```

Warning: acs needs at least 4GB (4096MB) of memory.

Vault Pass

As you know, push passwords in clear code to a repository is a bad practice, even for environments intended for development. Sensitive information is encrypted using Ansible Vault. To allow ansible to decrypt the information you just need to add a file to the home of ansible_env folder. Example:



```
-->pburgos@rigel ~/.ansible_acsenv on master !2 ?1 > echo "password" > .vault_pass
```

```
cd <ansible_acsenv>
echo "Password" >> .vault_pass
```

Ask your instructor for the real password to be use here.

Virtual Machine

The cycle for a dev environment with virtual machine: build provisiondestroy

build

Let's follow an example. Let's say that:

- git repository is located at /home/pburgos/Development/Repositories/bitbucket/almasw
- acs binary is located at /home/pburgos/.acsenv/versions/COMMON-2020JUN/dist
- acs_working is located at /home/pburgos/.acsenv/versions/COMMON-2020JUN/working

vagrant up

```
vagrant --acs_repo=/home/pburgos/Development/Repositories/bitbucket/almasw --acs_binary=/home/pburgos/.acsenv/versions/COMMON-2020JUN/dist --acs_working=/home/pburgos/.acsenv/versions/COMMON-2020JUN/working up acs01
```

This command will retrieve Centos7 box, and configure the mount points as nfs shares. It will ask for the administrator password to setup the NFS.

If a public_ip was specified for the machine in boxes_config.yml, then the user will be prompted to select the network interface to bridge to:

```

vagrant --acs_repo=/Users/pburgos/Development/Repositories/bitbucket/almasw --acs_binary=/Users/pburgos/.acsenv
/versions/COMMON-2020JUN/dist --acs_working=/Users/pburgos/.acsenv/versions/COMMON-2020JUN/working up acs01
Bringing machine 'acs01' up with 'virtualbox' provider...
==> acs01: Importing base box 'centos/7'...
==> acs01: Matching MAC address for NAT networking...
==> acs01: Setting the name of the VM: ansible_acsenv_acs01_1595780860590_42357
==> acs01: Clearing any previously set network interfaces...
==> acs01: Available bridged network interfaces:
1) en3: Thunderbolt Ethernet
2) en0: Wi-Fi (AirPort)
3) en1: Thunderbolt 1
4) en2: Thunderbolt 2
5) bridge0
6) awdl0
==> acs01: When choosing an interface, it is usually the one that is
==> acs01: being used to connect to the internet.
==> acs01:
    acs01: Which interface should the network bridge to? 1
==> acs01: Preparing network interfaces based on configuration...
    acs01: Adapter 1: nat
    acs01: Adapter 2: hostonly
    acs01: Adapter 3: bridged
==> acs01: Forwarding ports...
    acs01: 22 (guest) => 2222 (host) (adapter 1)
==> acs01: Running 'pre-boot' VM customizations...
==> acs01: Booting VM...
==> acs01: Waiting for machine to boot. This may take a few minutes...
    acs01: SSH address: 127.0.0.1:2222
    acs01: SSH username: vagrant
    acs01: SSH auth method: private key
    acs01:
    acs01: Vagrant insecure key detected. Vagrant will automatically replace
    acs01: this with a newly generated keypair for better security.
    acs01:
    acs01: Inserting generated public key within guest...
    acs01: Removing insecure key from the guest if it's present...
    acs01: Key inserted! Disconnecting and reconnecting using new SSH key...
==> acs01: Machine booted and ready!
==> acs01: Checking for guest additions in VM...
    acs01: No guest additions were detected on the base box for this VM! Guest
    acs01: additions are required for forwarded ports, shared folders, host only
    acs01: networking, and more. If SSH fails on this machine, please install
    acs01: the guest additions and repackaging the box to continue.
    acs01:
    acs01: This is not an error message; everything may continue to work properly,
    acs01: in which case you may ignore this message.
==> acs01: Setting hostname...
==> acs01: Configuring and enabling network interfaces...
==> acs01: Rsyncing folder: /Users/pburgos/ansible_acsenv/ => /vagrant
==> acs01: Exporting NFS shared folders...
==> acs01: Preparing to edit /etc/exports. Administrator privileges will be required...
The nfsd service does not appear to be running.
Starting the nfsd service
==> acs01: Mounting NFS shared folders...

```

provision

To provision our new created VM we will use ansible playbooks. It does not matter how many machines are active at a given time, ansible will provision the machines that are running at a given time. Be careful if you use Virtual Machines in your local hosts for other purposes. The following command will try to provision all machines in running state.

```
ansible-playbook -i vagrant.py vagrant_provision.yml
```

destroy

You can destroy you VM to create a new one. These can happed when a different acs_binaries is needed.

Let's say I want to destroy acs01. First I verified the status of that machine

```
vagrant status
Current machine states:

acs01                running (virtualbox)
acs02                not created (virtualbox)
```

To destroy acs01 VM we issue:

```
vagrant destroy acs01
acs01: Are you sure you want to destroy the 'acs01' VM? [y/N] y
==> acs01: Forcing shutdown of VM...
==> acs01: Destroying VM and associated drives...
==> acs01: Pruning invalid NFS exports. Administrator privileges will be required...
Password:
```

Let's verified it was destroyed:

```
vagrant status acs01
Current machine states:

acs01                not created (virtualbox)
```

Great!

Docker Container

To be added soon.

Appendix: Accessing you VM through ssh

There are several ways to access your VM box through ssh. Let's review the alternatives:

Accessing from host through private network

Let's say we want to access to acs01. From boxes_config.yml file we review earlier, the private network IP for the host is 10.10.10.10. Then:

```
ssh almagr@10.10.10.10
The authenticity of host '10.10.10.10 (10.10.10.10)' can't be established.
ECDSA key fingerprint is SHA256:1BQ60Rqfa7clxxdytYKpXoLHqJKL8izw48maYxzCsBw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.10.10' (ECDSA) to the list of known hosts.
almagr@10.10.10.10's password:
Last login: Sun Jul 26 22:55:45 2020 from 192.168.1.5
[almagr@acs01 ~]$
```

Accessing from other Host in the Network through public IP

Let say we want to access from other computer in the network with ssh. From boxes_config.yml the public ip address is 192.168.1.155. Then:

```
ssh almagr@192.168.1.155
almagr@192.168.1.155's password:
Last login: Sun Jul 26 23:11:08 2020 from 192.168.1.5
[almagr@acs01 ~]$
```

Accessing using vagrant ssh

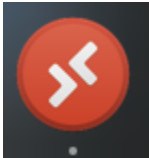
Vagrant comes with their own tool to access a box through ssh. From the ansible_acsenv folder execute:

```
vagrant ssh acs01
Last login: Sun Jul 26 21:09:53 2020 from 10.0.2.2
/usr/bin/xauth: file /home/vagrant/.Xauthority does not exist
[vagrant@acs01 ~]$
```

The VM comes preconfigured with RDP so that we can have access to Gnome Desktop to grab a Graphical Interface. Yo can use:

- Microsoft Remote Desktop for Mac OSX
- Remmina Client for Linux

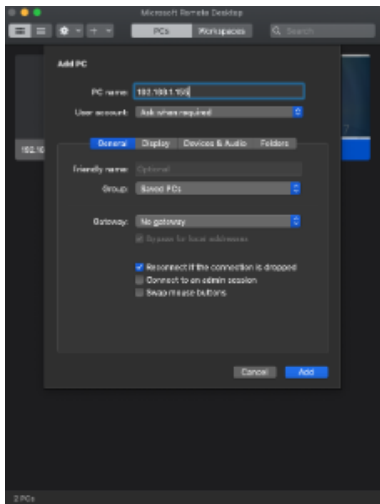
Microsoft Remote Desktop



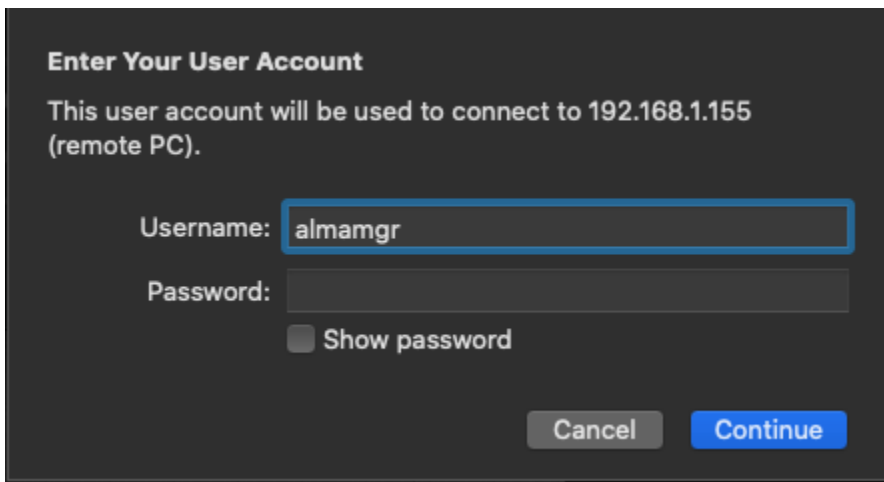
Microsoft Remote Desktop is available for free from the App Store



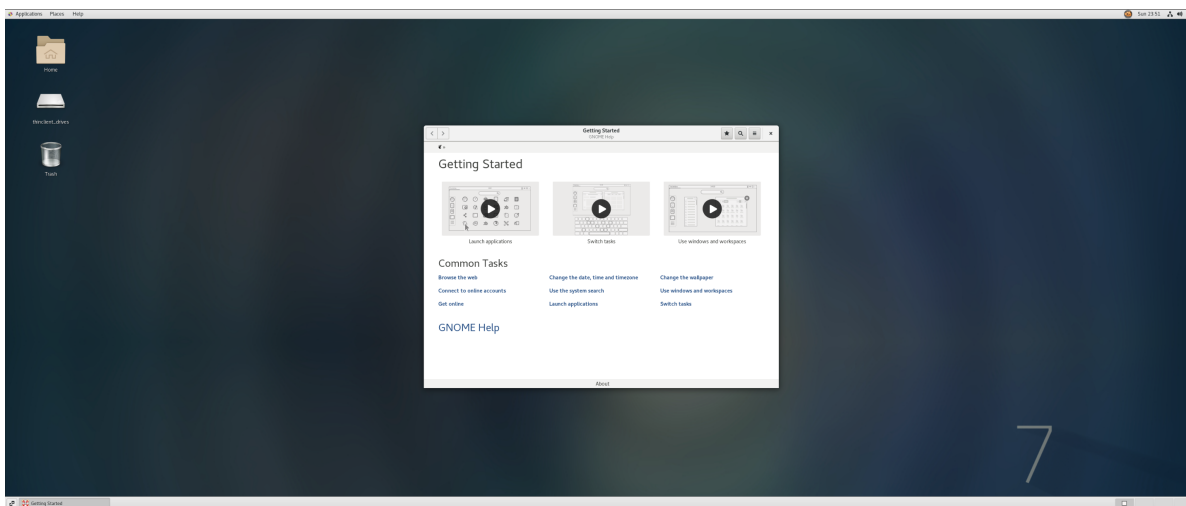
Yo can add your guest using the ip address for your box:



Once added, you can double click the desktop icon. A pop-up asking for password will appear:

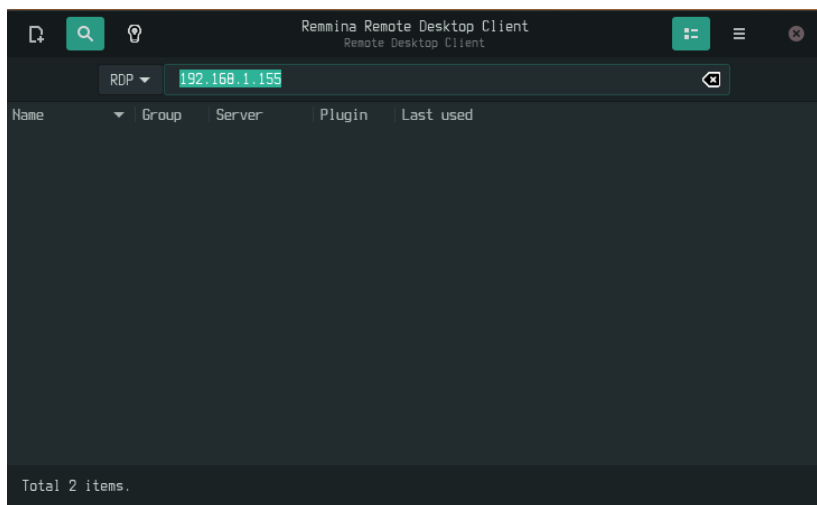


After entering your credentials you should have access to your desktop.

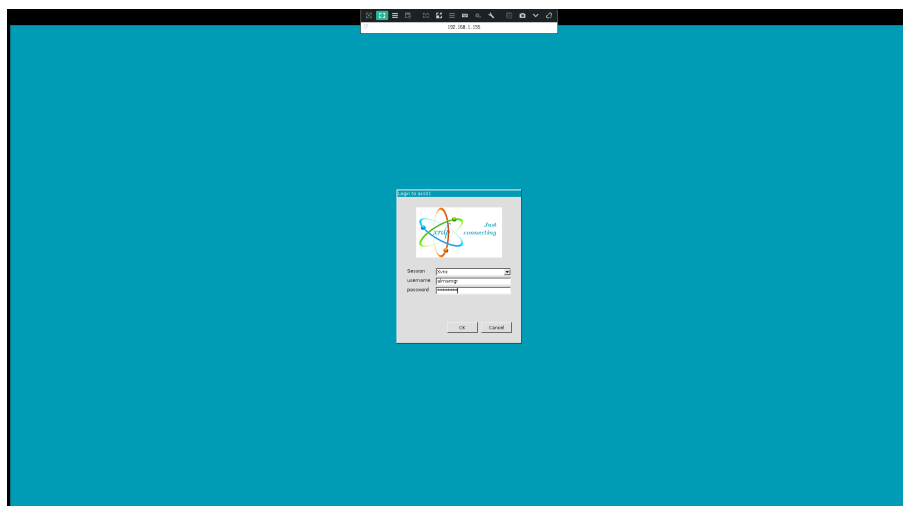


Remmina Remote Client (Linux)

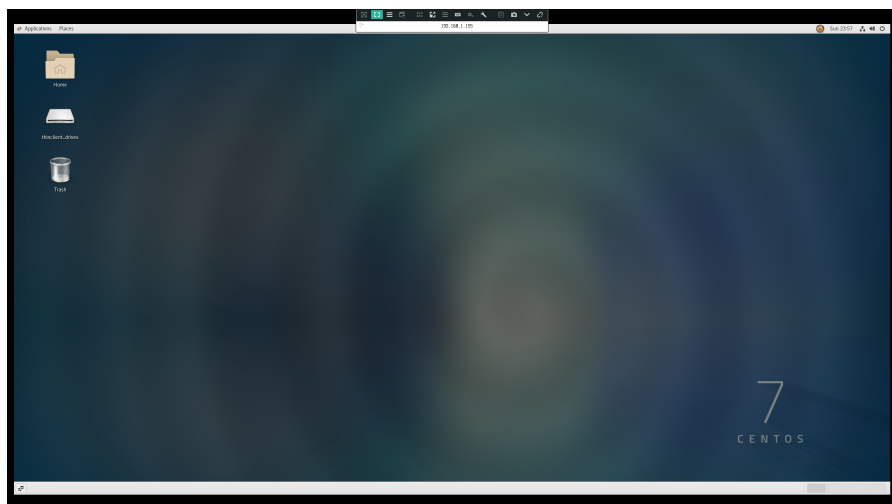
To access from remmina, select RDP as the protocol to be used, as depicted in the next image:



After connect, enter your credentials as depicted next:



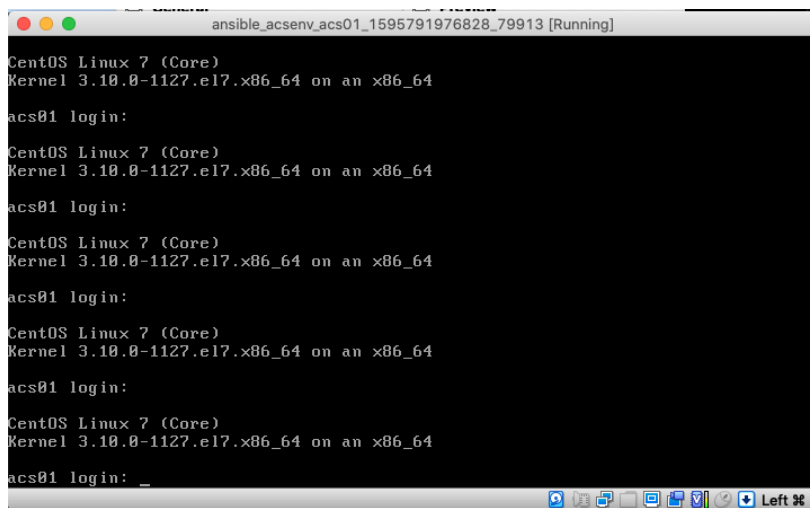
Great!...you should have access to your desktop.



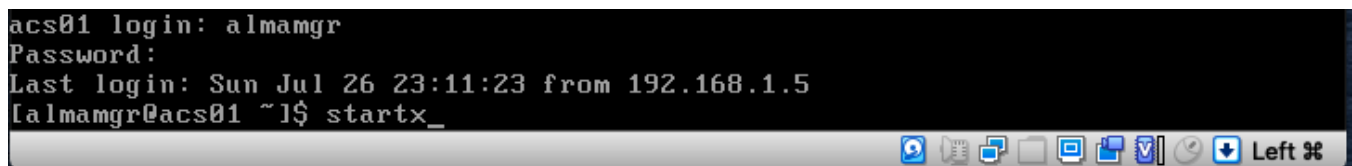
VirtualBox gui can be enable inside boxes_config.yml file. Just change the gui property to true. Example:

```
---
memory: 4096
cpus: 2
gui: true
boxes:
  - name: acs01
    ip_private: 10.10.10.10
    ip_public: 192.168.1.155
  - name: acs02
    ip_private: 10.10.10.11
    ip_public: 192.168.1.156
```

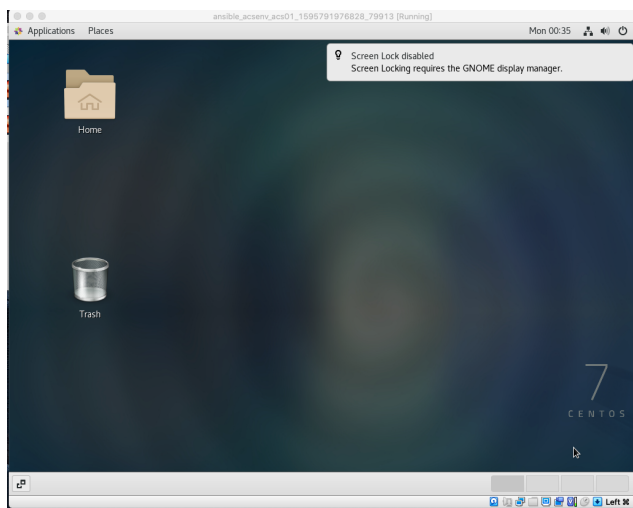
With that you can

A terminal window titled 'ansible_acsenv_acs01_1595791976828_79913 [Running]' showing the output of an Ansible playbook. The output displays the system information for 'CentOS Linux 7 (Core)' and 'Kernel 3.10.0-1127.el7.x86_64 on an x86_64' for the 'acs01' host. The prompt 'acs01 login:' is shown multiple times, indicating the playbook is running on that host. The terminal window has a standard Linux desktop environment with a taskbar at the bottom.

Once the credentials are entered just run startx as depicted:

A terminal window showing the login process for 'acs01'. The prompt 'acs01 login:' is followed by the username 'almamgr'. The prompt 'Password:' is shown, and then the login is successful. The output shows 'Last login: Sun Jul 26 23:11:23 from 192.168.1.5' and the prompt '[almamgr@acs01 ~]\$. The user then enters the command 'startx_'. The terminal window has a standard Linux desktop environment with a taskbar at the bottom.

After that you'll have your GUI environment:



From inside the VM you can connect to a VPN using openconnect, which comes already installed in your VM. To connect against, let's say, <vpn_server> open a new terminal and execute:

```
sudo openconnect <vpn_server>
```

You'll be prompted to accept a certificate. Answer yes and enter your credentials when requested.