

|  |
| --- |
| ANSIBLE AUTOMATION  TOOL |

|  |  |  |
| --- | --- | --- |
| LEBANESE UNIVERSITY-Faculty of Science |  |  |
| COURSE: I3351 System Administration  Professor: Dr. Ahmad Fadlallah  Done by: Simon Zeidan |  |  |

TABLE OF CONTENTS

[Acknowlegements 4](#_Toc34862777)

[Abstract 5](#_Toc34862778)

[Section 1- Introduction 6](#_Toc34862779)

[Automation Concept 6](#_Toc34862780)

[Problem to Solve 6](#_Toc34862781)

[Ansible as Solution 7](#_Toc34862782)

[Report Structure 7](#_Toc34862783)

[Section 2-Ansible-Automation Tool 8](#_Toc34862784)

[Ansible Background 8](#_Toc34862785)

[Ansible Installation 9](#_Toc34862786)

[Ad-Hoc Ansible command 9](#_Toc34862787)

[Creating an inventory file 10](#_Toc34862788)

[Brief Explanation on YAML files 11](#_Toc34862789)

[Variables in YAML files 11](#_Toc34862790)

[Loop task 12](#_Toc34862791)

[Ansible Playbooks 12](#_Toc34862792)

[Execute Ansible Playbook 12](#_Toc34862793)

[Ansible Modules 12](#_Toc34862794)

[Ansible Roles 13](#_Toc34862795)

[Jinja2 Templating 14](#_Toc34862796)

[Section 3-Working With Ansible 15](#_Toc34862797)

[Introduction 15](#_Toc34862798)

[My Demo 15](#_Toc34862799)

[Section 4-Conclusion 17](#_Toc34862800)

[Summary 17](#_Toc34862801)

[Achievements and Problems faced 17](#_Toc34862802)

[Section 5-Appendix 18](#_Toc34862803)

[Working with Virtual Box 18](#_Toc34862804)

[Enabling SSH on the Virtual Machine 18](#_Toc34862805)

[Build a RPM to install ansible 18](#_Toc34862806)

[Change the port in the target machines 18](#_Toc34862807)

[Changing Configuration in ansible.cfg file 19](#_Toc34862808)

[Hosts File 19](#_Toc34862809)

[My Work 20](#_Toc34862810)

[My Work Output 21](#_Toc34862811)

[Section 6- Reference 23](#_Toc34862812)

[2. Useful sites 23](#_Toc34862813)

**TABLE OF Figures**

[Figure 1: A few groups of servers 8](#_Toc34862814)

[Figure 2: Installation of Ansible 9](#_Toc34862815)

[Figure 3: Ad-hoc running "pwd" command 10](#_Toc34862816)

[Figure 4: Example of inventory file 11](#_Toc34862817)

[Figure 5: Importing the roles to the main playbook 14](#_Toc34862818)

[Figure 6: Ansible\_Project Folder 16](#_Toc34862819)

[Figure 7: Installing Code::Blocks from another machine 16](#_Toc34862820)

[Figure 8: file "ssh\_config" to identify the ssh port used 18](#_Toc34862821)

[Figure 9: Configuration done on ansible.cfg file 19](#_Toc34862822)

[Figure 10: Content of folder "/etc/ansible" 20](#_Toc34862823)

[Figure 11: The Role folder backup2 contents 20](#_Toc34862824)

[Figure 12: A YAML file containing the tasks done on machine2 20](#_Toc34862825)

[Figure 13: Result after executing an Ad-hoc 21](#_Toc34862826)

[Figure 14: Executing "ls -al" command 21](#_Toc34862827)

[Figure 15: Ansible\_Project output part1 22](#_Toc34862828)

[Figure 16: Ansible\_Project output part2 22](#_Toc34862829)

# Acknowlegements

We thank Dr. Ahmad Fadlallah for his help throughout this semester while we were working on the project. His help has organized and cleared our thoughts not only on this project but also on every single project from now on.

# Abstract

As we all know that nowadays, there is a need to be fast and to reach each client either if we need to configure a system or an online application, for example twitter has 500 million tweet per day [3], so to manage this large number of data we need a helping automation tool that maintains that human errors will not occur. Throughout the following report I’ll be discussing the need of automation tools mainly Ansible and how can we work with it.

# Section 1- Introduction

## Automation Concept

Since the beginning of networked computing, deploying and managing servers reliably and efficiently has been a challenge. Historically, system administrators have typically been walled off from the developers and users who interact with the systems they administer, and so they have had to manage servers by hand, installing software, changing configurations, and administering services on individual servers. As data centers grew, and hosted applications became more complex, administrators realized they couldn’t scale their manual systems management as fast as the applications they were enabling. That’s why server provisioning and configuration management tools came to flourish. Server virtualization brought large-scale infrastructure management to the fore, and the number of servers managed by one admin (or by a small team of admins), has grown by an order of magnitude. Instead of deploying, patching, and destroying every server by hand, admins now are expected to bring up new servers, either automatically or with minimal intervention. Large-scale IT deployments now may involve hundreds or thousands of servers; in many of the largest environments, server provisioning, configuration, and decommissioning are all entirely automated. [1]

## Problem to Solve

A system administrator who does everything manually wastes not only his time but the time of the employees in his company as well. Thus tasks that a system administrator performs repeatedly should be automated. One of the solutions, which some developers apply is through writing scripts and run them on each machine since automation through scripting save some time and prevents human error-related mistakes, but on the other hand, this method is time consumption since he will need to reach each machine alone and needs more coding skills as well as these scripts need regular maintenance especially if tasks are changing by time. In my opinion, there is a need for a specialized tool for automation since it frees up a system administrator’s time and performs these automated tasks faster and more efficiently. Last even if you decide to ignore the Ansible’s powerful features, you will be able to manage your servers much more efficiently after reading this

## Ansible as Solution

Ansible is powerful IT automation tool that can be quickly learn and it is simple enough to be learned by any one even the ones who are not skilled in IT domain in the present of Ansible tower which is a GUI interface of Ansible tool available on Ansible official website free and easy to download and use. Ansible helps us increase our productivity and efficiency by automating a number of repetitive tasks in our environment whether it is sizing, creating a new host, or virtual machines, configuration on these virtual machines, handling hundreds of servers, migration, deploying application, performing security. Ansible is a powerful automation tool that we can learn quickly, it is simple enough for everyone in IT, also powerful enough to automate even the most complex deployments. It can be used for provisioning, configuration management, continuous delivery, application deployment, security compliance. Finally, Ansible is simple, powerful and agentless.

## Report Structure

The report contains:

1. In Section 2:

* A brief explanation on ansible.
* Ansible installation.
* Explaining ansible inventory files, YAML files.
* Explanation on playbooks, roles, jinja2 templating.

1. In Section 3:

* You’ll see a sample on how to work with ansible.

1. In Section 4:

* A conclusion about this work.

1. In Section 5:

* More explanation and images illustrating how to configure your machine to work with ansible.
* My work through this project.

# Section 2-Ansible-Automation Tool

## Ansible Background

Ansible is an automation tool used to configure a number of machines or servers. It has been design to make configuration easy in almost every way, through its simple installation to its simple language where it is so close to the formal English language. Ansible uses ssh secured connection for more security thus there is a need to configure the ssh connection between the master and the target machine. After configuring this connection Ansible uses three types of files inventory files, playbook files, as well as files for role. All these file are used to add tasks and configuration to deploy tasks on the target machines.

“Ansible was released in 2012 by Michael DeHaan (@laserllama on Twitter), a developer who has been working with configuration management and infrastructure orchestration in one form or another for many years. Through his work with Puppet Labs and RedHat (where he worked on Cobbler, a configuration management tool and Func, a tool for communicating commands to remote servers), and some other projects¹⁰, he experienced the trials and tribulations of many different organizations and individual sysadmins on their quest to simplify and automate their infrastructure management operations.” [1]

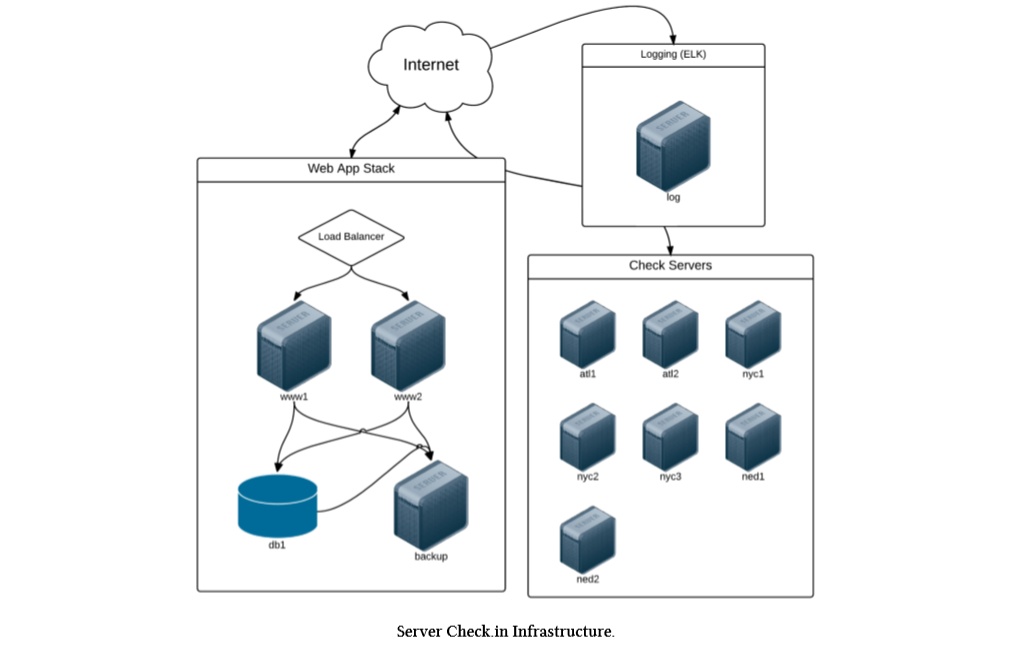


Figure 1: A few groups of servers

## Ansible Installation

There are different options to install Ansible on a Linux system remember Ansible can only be Linux as of now however windows machines can be a target for automation with Ansible so Ansible do support windows as a target server.

* Fedora/RHEL/CentOS:

Use the following command:

yum –y install ansible

* Debian/Ubuntu:

Use the following command:

apt-get install ansible

* Additional options:

Installation from a source on Git and build Rpm yourself.

For more information, see the Appendix section.

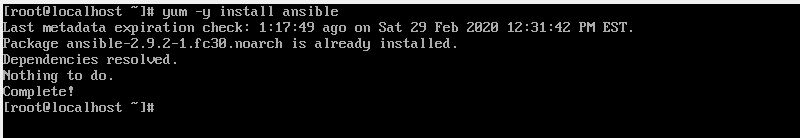


Figure 2: Installation of Ansible

## Ad-Hoc Ansible command

Ansible allows admins to run ad-hoc commands on one or hundreds of machines at the same time, using the ansible command, we can run a couple of commands (ping and free -m) on target servers that we added to our Ansible inventory file. This part will explore ad-hoc commands. Use ansible with the -a argument ‘hostname’ to run hostname against all the servers:

$ ansible all –i 10.0.2.10:3212, -m ping –u client1 –k

SSH password:

Ansible will run this command against all three of the servers, and return the results (if Ansible can’t reach one a server, it will show an error for that server, but continue running the command on the others). By default, Ansible will run your commands in parallel, using multiple process forks, so the command will complete more quickly. If you’re managing a few servers, this may not be much quicker than running the command serially, on one server after the other, but even managing 510 servers, you’ll notice a dramatic speedup if you use Ansible’s parallelism (which is enabled by default).

Run the same command over and over again, and it will always return results in the same order. It’s fairly rare that you will ever need to do this, but it’s much more frequent that you’ll want to increase the value (like -f 10, or -f 25… depending on how much your system and network connection can handle) to speed up the process of running commands on tens or hundreds of servers.

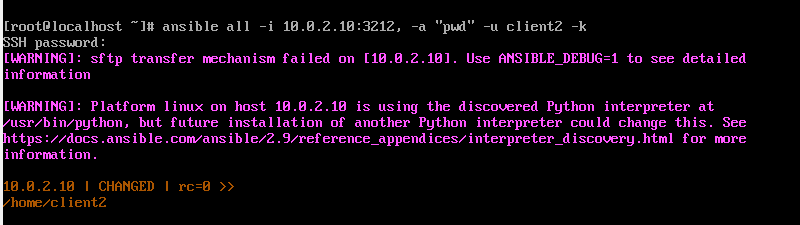


Figure 3: Ad-hoc running "pwd" command

## Creating an inventory file

Ansible can work with your multiple systems in the infrastructure virtual box at the same time. in order to work with multiple servers Ansible needs to establish connectivity to those servers this is done using ssh for Linux and power shell remoting for windows that’s what makes Ansible agentless. Agentless means that you don’t need to install any additional software on the target machines to be able to work with Ansible it uses a simple ssh connectivity. The major disadvantage of other automation tools is that you are required to configure an agent on the target system before you can invoice any kind of automation. The information of these target systems is stored in a file called inventory file if we don’t define an inventory file Ansible uses the default host file located in /etc/ansible/hosts.

An inventory file is in an ini format; it is simple number of servers listed one after the other. We can also group different servers together by defining it by the name of the group followed by the ip address of the host machine/s. We can define more than a group in a single inventory file. Also we can define a group of groups. These inventory files are used to target a group of servers to configure them at once. The form of inventory files allows us to define alias to each server define in these file and allows us to add parameters that helps in saving the information of the host machine, the type of the connection, port used, ssh password for connection. (see the appendix to know more about the ssh connection).

Some of the Inventory parameters:

* ansible\_connection = {-ssh/winrm/localhost}
* ansible-host = {server1.company.com,10.0.2.10}
* ansible\_port = {-22/5986}
* ansible\_user = {-root/administrator}
* ansible\_ssh\_pass = {Password}



Figure 4: Example of inventory file

## Brief Explanation on YAML files

Ansible playbooks are text files or configuration files that are written in a format called YAML. All ansible playbooks are written in YAML. YAML files contains data which are written in formal English, the data is in form key-value pair form separated by a colon. We have other ways of writing these data that extends the key-value form and depends on it by using Array/List form or dictionary form. The Array/List form is an ordered collection of elements related to the same key whereas dictionary/map form is an unordered collection of set of properties grouped together which describes a certain key. A dash must be added before each element of a List while there is no need for a dash in representing a data in a dictionary form. But both forms use the key-value form we illustrate before.

## Variables in YAML files

Just like other scripting or programming language the variables are used to store values that differ with items. For example, we are trying to perform the same operation on 100 servers we only need a single playbook for all hundred servers however it’s the variables that store information about the host name, user name or password that are used to connect to each server. We can also define a variable in a playbook and place its values in another file or in the same file.

## Loop task

The term “with-items” is a looping directive which execute the same task multiple number of times each time it runs it stores the value of the item in the item variable so you can simply replace the given host name for example with {{ item\_value }} make sure to put double braces.

## Ansible Playbooks

Ansible playbooks are ansible orchestration language it is in playbooks where we define what we want ansible to do it is a set of instructions provided to ansible so it can be run on the target machine. It is similar to running a service of a command on different servers in a sequence and restarting these servers in a particular order or it could be hard as deploying hundreds of virtual machines in public and private infrastructure setup their network, cluster configuration, setup monitors.

Playbooks are written in a YAML format a playbook is a single YAML file containing a set of plays. A play defines a set of activities or tasks to be run on a single or a group of host. A task is a single action to be performed on a host.

Example of tasks:

Execute a command

* Run a script
* Install a package
* Shutdown / Restart

## Execute Ansible Playbook

Once you successfully build the ansible playbook you can run it by executing the ansible-playbook command and specify the name of the playbook file which will be .yml extension

## Ansible Modules

Ansible Modules are categories into various groups based on their functionality there are 3387 modules available once we downloaded ansible automation tool. Modules like some command scripts, and programs are basically python programs that perform specific actions if you don’t find the module that satisfies your requirements you can develop your own module. You simply need to write a python programs and place it in the module directory on your serve. The module must be written in a particular format.

## Ansible Roles

In this part we will explain more about ansible roles that will be used in the next section where we go through our own example on ansible tool.

If you have worked with any other language or scripts, you probably know that you could write scripts or programs in a single file or what is more preferred is to modulate it into packages: packages, modules, classes, functions. That way our code becomes more organized and useable and easy to read and share with others. Our folder is built in ansible with the help of roles. We don’t have a code here but here we have: inventory files, variables, playbooks. When we have too many things to automate our playbooks, inventory files and variables are going to get bigger and difficult to manage so unlike what we did before writing a large playbook will not be ideal in that case that’s where include statements and roles come into play.

In order to modulate it we simply cut the large playbook into smaller files that address different use cases then finally we have a master playbook that includes these files.

Now our master playbook is simply few lines long we have more separated playbook files that are smaller and easier to maintain and they can be reused for different use cases.

Previously we have been creating inventory files and defining variables and writing playbooks as we thought it fits our needs. The standard way that ansible recommends is by writing your project in a role folder.

Ansible roles define a structure for our project and standards on how files and folders are organized in this project. In this example we have an ansible project that contains an inventory.txt that contains the information of the target machines and a setup-application.yml file which is a single large playbook to do certain tasks. With roles we will reorganize the playbook and variables, we explained about them before, by creating a new folder called Role within which we create another folder with the name of the role in our case here it is called webservers. In this case webservers folder contain additional folders: files, template, tasks, handler, vars, defaults, meta each of these folders contains files associated for those purposes for example the tasks folder will contain yml file containing list of tasks and the vars folder contain var files defined with variables declaration. So we move the variables into a file in the var folder and we move the task into file in the tasks folder. Now in our master playbook we can simple say assign the following roles to the server.

Ex:

- name: Copy files

host: target

roles:

- webservers

The role is webserver which is the role we created in the roles folder one of the advantages of using roles is that we don’t have to worry about important tasks and variable like we did before.

Ansible automatically import all tasks from the tasks folder and all variables from vars folder as well as supporting files from the other folder.



Figure 5: Importing the roles to the main playbook

## Jinja2 Templating

Templating is a process of generating dynamic content or expressions in fact we already use templating in some of playbooks especially when we used the variable item and we define a loop to replace this variable. The {{ }} is a jinja2 template if we google jinja2 online we find that it is a fulfill templating language used for python, so it is not something specific to ansible but we can use it in our playbooks to make our work easier. Finally, jinja2 is a general template engine used for various purposes.

# Section 3-Working With Ansible

## Introduction

Setting up a complex infrastructure that spans across public and private cloud and hundreds of virtual machines using ansible. With ansible you can configure virtual machines on public clouds like amazon and private clouds with a private cloud setup like VMware base infrastructure and moving to configuring and setting up applications on those machines.

We have seen how to install ansible on a system but we can really use one or two additional systems to get a better understanding on how it works in the real world. There are multiple of ways to setup setting environments by setting up a few of VMs in your lab or using a virtualization software (Virtual box, VMware) to setup some virtual machines or use something like Docker.

Finally, we need a control system so install ansible on a control machine and have some hosts machines or target machines setup on the same network. Let’s start.

## My Demo

Before practicing ansible playbooks and roles so steps must be done. Our target is to setup multiple VMs one which will act as an ansible controller in my case it is the fedora machine and two other will act as target servers that ansible control machine will target to make configuration changes on them. In this demo I’ll be using Oracle virtual box (For more information on how to configure the machines see the Appendix section).

First to start working with ansible you need to run “ansible-galaxy init Ansible\_Project”, it will automatically create a folder with all the needed folders and files as we present before in the role section my work starts by writing the tasks that must be done in the tasks folder and by then we must edit the file main.yml in the tasks directory by importing the tasks so we add “- import\_tasks: backup1.yml” in this file. After that we run the command:

“ansible-playbook setup\_application –i inventory.txt –become\_exe=sudo” (For more information, on my work see the Appendix section).

Finally, the tasks done by this folder is copying a file form the control machine called information.txt to the target machines. Also these folders contain the task of installing a group of libraries which lead to installing CodeBlocks program on all target machines. (For more information, see the Appendix section).

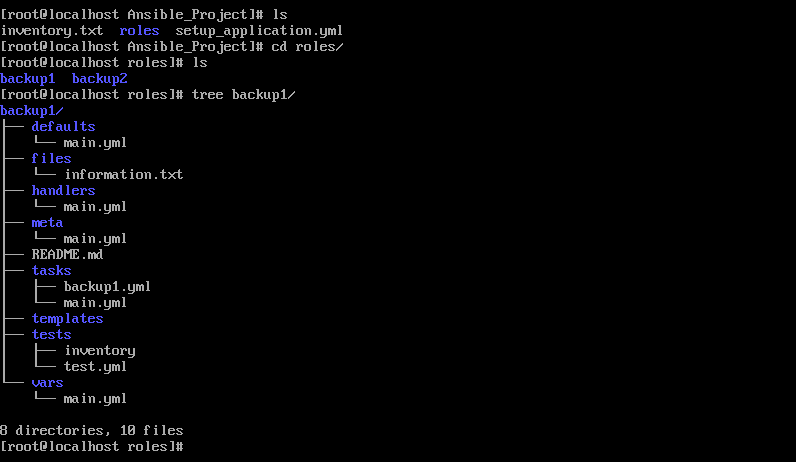


Figure 6: Ansible\_Project Folder

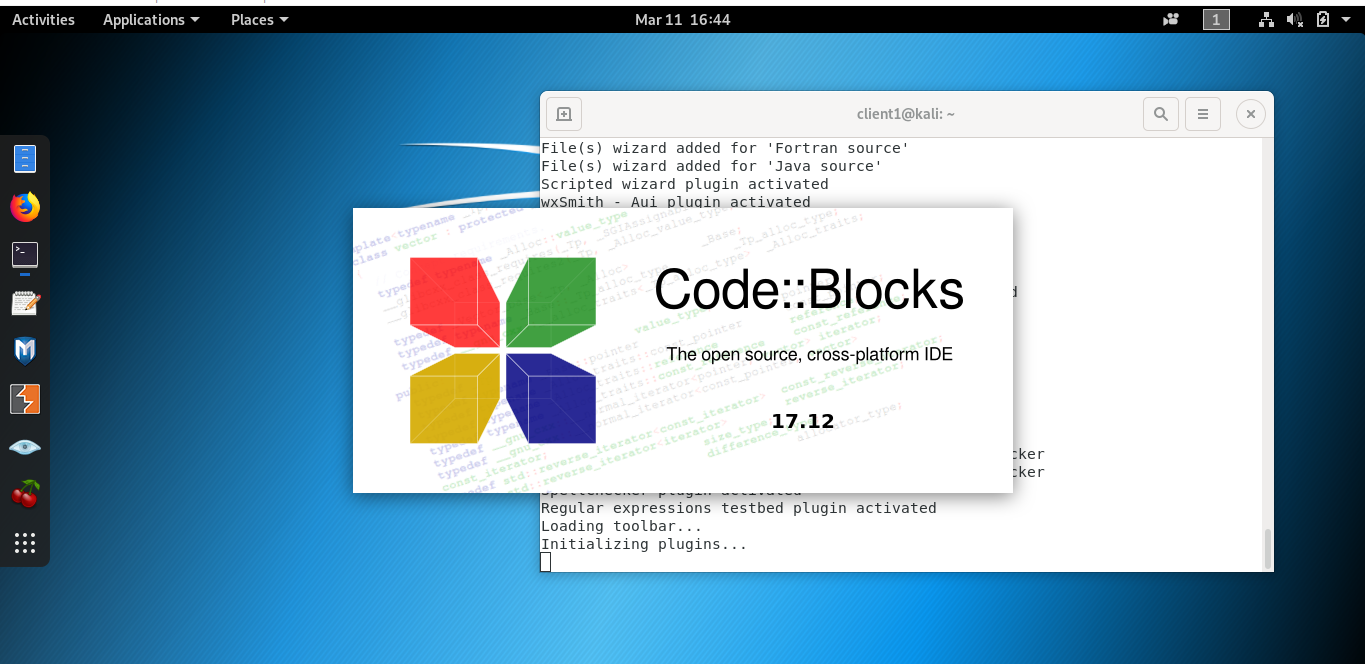


Figure 7: Installing Code::Blocks from another machine

# Section 4-Conclusion

## Summary

As a conclusion, on my work I hope it was useful in explaining the installation and the usage of ansible automation tool. Through this report we discussed about the importance of automation and we explained how to setup and install ansible. We talked about YAML files and jinja2 templating. Moreover, we learn how to develop playbooks and what are loops and variables used for. We also specifically discussed more details on roles and how to develop our own customized role. As an addition information, these roles can be shared on a website <https://galaxy.ansible.com> where you can also benefit from others’ playbooks. For more information visit the site <https://doc.ansible.com> all ansible documentation are found on this website.

## Achievements and Problems faced

Once I decided to take automation and specifically ansible tool as my project in the I3351 course it seemed to be an easy task at first but then as I started working on this tool I realized that it is so easy to be installed on any Linux machine. But the hard work comes after, since ansible is built with python libraries and concepts, so you must know a little about this language. Beside that ansible mainly uses YAML files which is also another language that you must know how to write your code in it with good syntax avoiding errors, since YAML language is so sensitive thus an additional space will destroy all your work.

I learned many things through this project since there is a need to go through configuration files change them without messing with the system, so there is a need always for a backup file for these configuration files. And I learn how to write an ansible role not only benefit from prewritten roles and playbooks found on the ansible documentation site or ansible.galaxy.com site. At last, by working on this project I realized how important each information we took in class which was for me a great reference for example when changing the privilege of a file to be accessed by the user I created (since some tasks are only done by the root). To solve this problem, I had to create a new user since we can’t access any system using ssh connection by root as a user. For that I created new users called client1 and client2 on two machines and I added them to the group root beside adding them to sudo users. This task wouldn’t be easy if I didn’t take the system administration course since writing well commands makes tasks easier for example changing the privilege of all files in a folder is not easy if it is done file by file rather than using the command “ls | xargs sudo chmod g=+rwx”.

# Section 5-Appendix

## Working with Virtual Box

While creating VMs in Virtual box if only 64 bit virtual machines are listed, kindly check on the following:

1. Check if Virtualization is enabled on your system. For this you must access the BIOS of your laptop and enable Virtualization if it’s not already enabled.
2. Check if Hyper-V is enabled on your system. If it is, ensure that it is disabled as two virtualization technologies cannot co-exist

## Enabling SSH on the Virtual Machine

If you need SSH enabled on the system, follow the below steps:

1. Ensure the /etc/apt/sources.list file has been updated as per above
2. Run the command: **apt-get update**
3. Run the command: **apt-get install openssh-server**
4. Run the command: **service sshd start**

**Remark:** Once you have a problem in executing your playbook or your Ad-hoc command, must go to the target machine and restart the ssh connection.

## Build a RPM to install ansible

You can also build an RPM yourself. From the root of a checkout or tar ball, use the make rpm command to build an RPM you can distribute and install.

$ git clone https://github.com/ansible/ansible.git

$ cd ./ansible

$ make rpm

$ sudo rpm -Uvh ./rpm-build/ansible-\*.noarch.rpm

## Change the port in the target machines

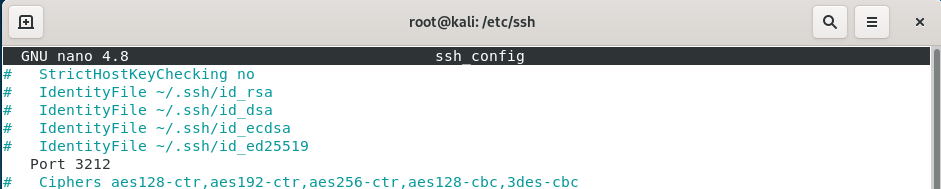
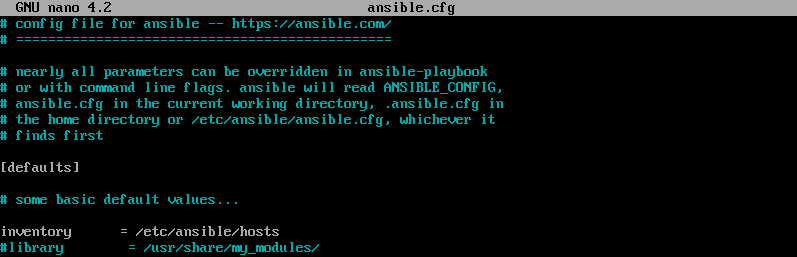
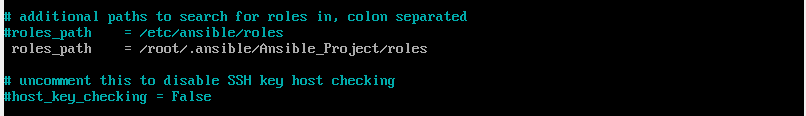


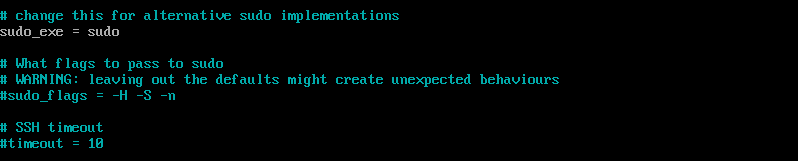
Figure 8: file "ssh\_config" to identify the ssh port used

## Changing Configuration in ansible.cfg file

Once you download ansible you need to change the configurations in the file ansible.cfg located in /etc/ansible folder. You just need to uncomment the following statements in the following figures.







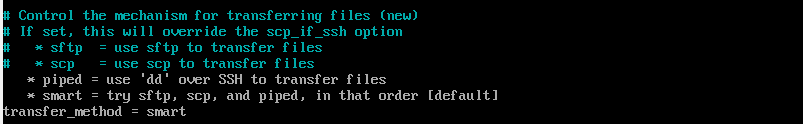
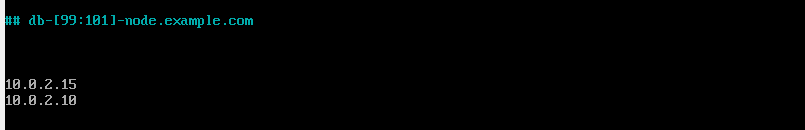


Figure 9: Configuration done on ansible.cfg file

## Hosts File

You must add the ip address of the target machines you are working with in a hosts file.



## My Work

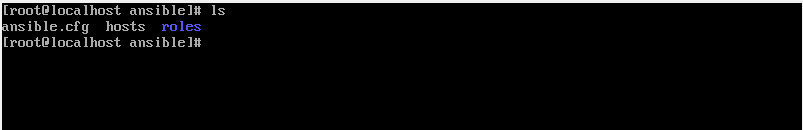


Figure 10: Content of folder "/etc/ansible"



Figure 11: The Role folder backup2 contents

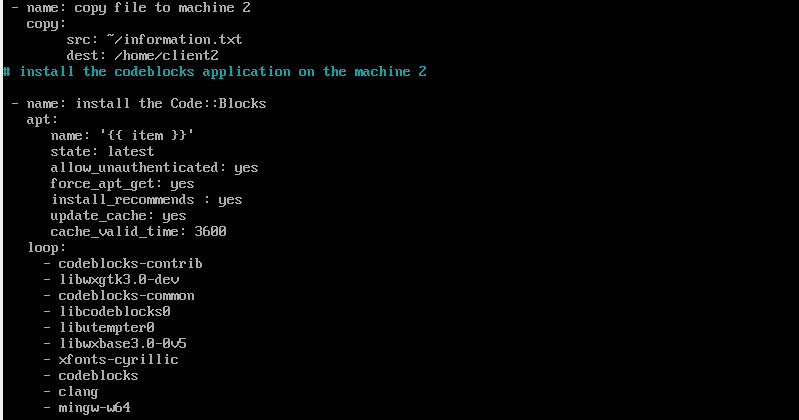


Figure 12: A YAML file containing the tasks done on machine2

## My Work Output

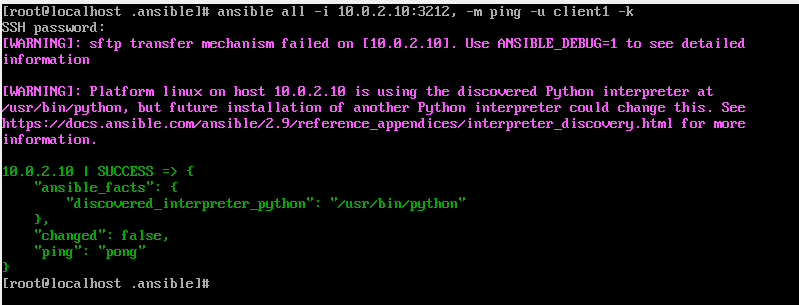


Figure 13: Result after executing an Ad-hoc

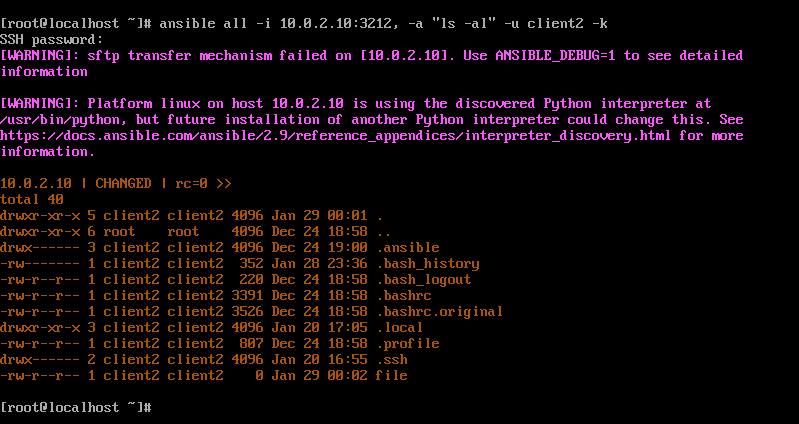


Figure 14: Executing "ls -al" command

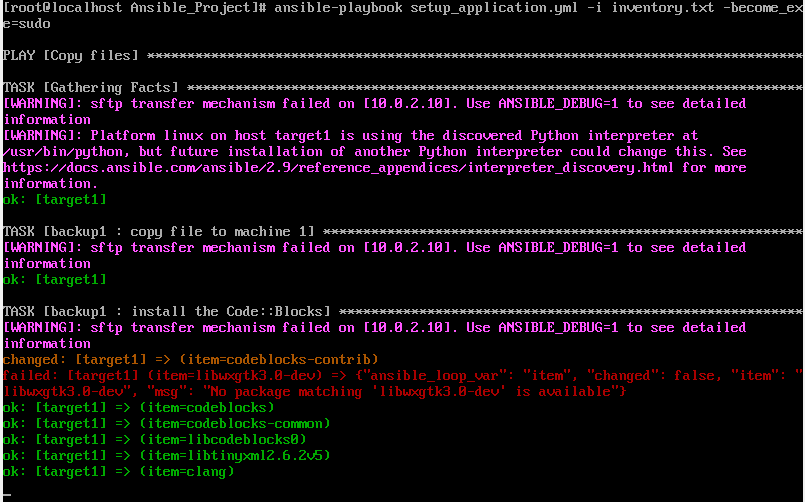


Figure 15: Ansible\_Project output part1

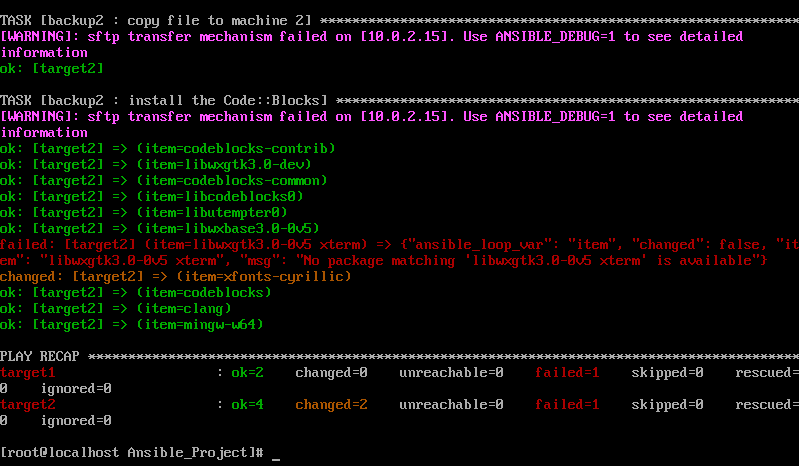


Figure 16: Ansible\_Project output part2

# Section 6- Reference

1. Ansible for DevOps

Server and configuration management for humans

Jeff Geerling

This book is for sale at http://leanpub.com/ansible-for-devops

This version was published on 2015-07-24

ISBN 978-0-9863934-0-2

## Useful sites

Ansible documentation:

<https://docs.ansible.com/>

Ansible Galaxy is Ansible’s official hub for sharing Ansible content:

<https://galaxy.ansible.com/>

1. <https://business.twitter.com/>