Process automation

Jorge Antonio Aqué González
Data Engineering
Universidad Politécnica de Yucatán
Km. 4.5. Carretera Mérida — Tetiz
Tablaje Catastral 4448. CP 97357
Ucú, Yucatán. México
Email: st1809007@upy.edu.mx

Axel Gonzalez
Rich it
Pétalo El Reloj, Del. Coyoacán, CDMX
Email: agonzalez@richit.ai

Angel Arturo Pech Che
Universidad Politécnica de Yucatán
Km. 4.5. Carretera Mérida — Tetiz
Tablaje Catastral 4448. CP 97357
Ucú, Yucatán. México
Email: angel.pech@upy.edu.mx

Abstract

Process automation is a topic that is taking a lot of popularity these days, this project covers related topics about process automation. Process automation has great benefits for work as it helps us reduce time and expense, in the same way this entire project is focused on the DevOps methodology, a methodology that would be of great help to implement in companies since it helps both the programmer as to the work in progress. The next project consists of the application of process automation at a basic level where the basic skills to learn to use automation tools are shown.

Index Terms

Automation, DevOps, Ansible, Docker, Git, Nmtui, SSH, Connections, Bash, Yaml, Node, VPN.

Process automation

I. Introduction

DUring technological advancement, people have created tools to facilitate daily work, having benefits such as saving time and money, in the field of engineering there is a lot of information involved and at all times you have to be updated with all the tools of the market to be able to have a better performance.

Process automation is an optimal solution to replace manual processes to reduce labor costs, time and avoid human errors that normally can happen. The concept of process automation is a topic that is gaining popularity lately and many tools have been created in order to be the most popular among engineers with the help of the use of friendly interfaces, little system demand, and the intuitiveness of the system.

During the meetings with the project managers at Rich IT, the idea of learning new tools that may be useful for data engineering was proposed, therefore, they talked about researching the most used tools for data automation in order to choose some within from the catalog and implement some tasks with the tool.

By having good practices, we can improve the performance of workflows with good results like those mentioned before, therefore, it is necessary to learn from the easiest task to the most tedious to handle. It is very likely that during the following years more tools and strategies will be created to make the most of production time and it is also very likely that we will have to work automating some process, that is why it is important for engineers to learn this concept and how implement it since it is good practice to learn tools that we can use to have a better performance.

II. Objectives

The objective of this project is to have a small immersion on the topic of data automation in order to have the necessary bases to start using tools of this type, therefore, the purpose is to investigate the most used tools and a implementation of process automation that serve as a basis.

III. State of the Art

When we talk about process automation, a lot of manual jobs are included that can be replaced, such as data collection, software management, computer administration. One of the cases in which automation is most used in the management and administration of computers/nodes. Therefore, when we talk about managing computers/nodes we mean providing all the necessary software such as laptops, desktops, smartphones and tablets from a central location. In other words, we orchestrate all those nodes.

A. *What is orchestration?*

In principle, it must be borne in mind that automation is not the same as orchestration but they complement each other. Orchestration refers to the automated configuration, management, and coordination employed for either a computer system, applications, and / or services. Orchestration helps IT manage complex tasks and workflows more easily. Managing many systems at the same time is very difficult and more if we do it manually, that is why the orchestration is responsible for managing several servers / machines at the same time, it can take advantage of pre-written tasks and only manage what is necessary to the machine that does it. need.

1) Automation and Orchestration: As mentioned before, automation and orchestration are not the same but they are related together, automation refers to the automatic management of a single task in order to replace human management, on the other hand, orchestration refers to automatic management multi-tasking

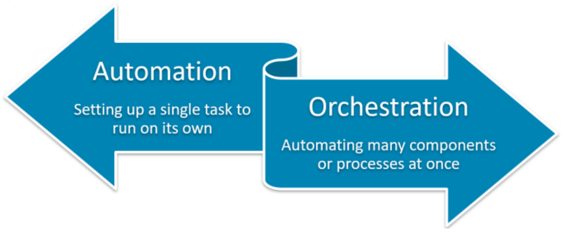

Automation
Setting up a single task to run on its own
Orchestration
Automating many components or processes at once

Fig. 1. Automation and Orchetration

2) The nodes: In the computer world there are several concepts for each particular thing, in this case when we talk about automation and orchestration everything is done from a central point (the provider) and everything necessary is distributed to multiple points (the receivers) so much a node is one of the elements that receives either software, configurations or simply administration.

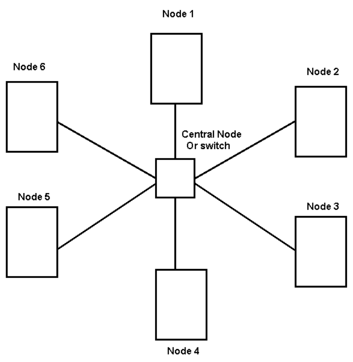


Fig. 2. Nodes

B. DevOps

During this project, the idea of learning new tools that contribute to data engineering was discussed, therefore in a work network there is the concept ”DevOps” which consists of unifying two departments of work, Development and Operations in order to avoid mistakes and have benefits such as delivery time, streamlining of work and ideas and saving money forming an endless cycle. Therefore the automation of processes is part of the cycle process which enters into Operations

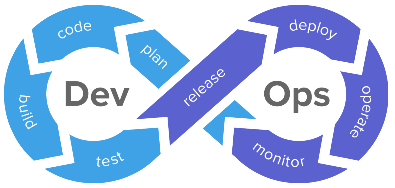


Fig. 3. DevOps cycle

1) Plan: Where every project begins, every work is born with a plan granting positions and functionalities to members and in the same way laying a basis for a minimum functionality for the project.

2) Code: Phase in which the idea begins to be tangible, This phase is where it is built, whether it is writing new code, redesigning infrastructure, automating processes, defining tests or implementing security.

3) Build: In other words, the project is completely finished.

4) Test: This phase is important since before delivering any code it needs to be tested against various problems in order to find out and make sure that the code is really good and that it can be competent.

5) Release: Once tested, the project can be made known as something functional.

6) Deploy: Can be delivered to clients as a functional project.

7) Operate and monitoring: Once the code is launched, it is essential to follow up. To do this, it is necessary to carry out an automated reading of parameters that allow us to warn of unforeseen failures.

C. Automation and Docker Containers

During this project, the idea of combining an automation tool with Docker was addressed in order to automate something very simple such as the deployment of a docker container. Together with the automation tool it is possible to deploy docker containers with a simple task, being something that saves us a lot of time since we can have that task saved. A Docker container allows us to transport some type of software(image) in a type of virtual ”box” together with everything necessary to be able to execute the application correctly, so we can execute said application on any machine with the only requirement of having Docker installed. Docker containers have many benefits and it is a very famous software among engineers since it facilitates the use of machine resources. Some benefits are:

- Containers are lighter (since they work directly on the kernel) than virtual machines.
- It is not necessary to install one operating system per container.
- Less use of machine resources.
- Greater quantity of containers per physical equipment.
- Better portability.

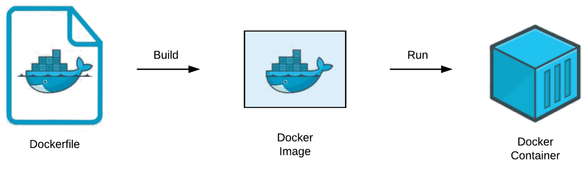


Fig. 4. Docker Process

IV. Methods and Tools

As an initial step after 4 weeks of the process automation project, during the first weeks I focused on researching fundamental concepts of automation and searching for tools used for this purpose. During this phase I used several websites and official pages of automation tools like Pupper, Vagrant, Cheff, Saltstack, Docker, GitHub among others, in the same way for the research I made use of small automation courses and searches on Github as part of my support for my project and official documentation, for the following weeks I used several tools that helped me to prepare my project such as:

- Language
  - Bash (CentOs)
- Bash Tools (Basics)
  - yum
  - vim
  - cd
- Software
  - Ansible
  - Apache
  - Docker
  - Github
- Query Language
  - YAML (Ansible)
  - Docker query language
  - Git Bash
- Connection software or protocol
  - VPN (OpenVPN)
  - SSH

A. Language

1) What is CentOs?: During this project the language used was Bash in CentOs (Community Enterprise Operating System). According with Red Hat, CentOS is a particular distribution of the Linux operating system. It is very popular for its benefits over other similar languages for its stability, consistency, easy to use administration and direct replication, this version of the open source operating system was created as a decision of Red Hat Enterprise Linux (RHEL). CentOs was something that could not be changed since the servers used were handling this distribution. Some benefits of CentOs are:

- Stability: Linux runs only stable and basic scheduled versions, reducing the risk of system crashes.
- Speed: CentOs has the ability to process tasks faster and more efficiently than many other similar Linux distributions.
- Security: The CentOS operating system is less prone to attack, which does not mean that it is impossible. This is ranked among the best in terms of security.
- Backup and Support: CentOs has the full support of RedHat, and in addition to its engineers and a large community of developers who always keep it safe and updated.

Fig. 5. CentOs

2) Bash: The only and main language used during this project was Bash. According to GNU Bash it is a command interpreter that executes, one by one, the instructions entered by the user or contained in a script and returns the results. It acts as an interface between the Linux kernel and text-mode users or programs. It incorporates numerous programming utilities and improvements over sh, its predecessor shell. Because it is a tool developed by GNU, it is usually used by default in current distros.

B. Bash tools

1) yum: It is a software manager package. It is useful for installing, updating and removing packages along with their dependencies on RPM-based Linux distributions (redhat, fedora, CentOs).

2) vim: It is an editor to create or edit a text file.There are two modes in vim. One is the command mode and another is the insert mode. In the command mode, user can move around the file, delete text, etc. In the insert mode, user can insert text.

3) cd: It is a command used in DOS and UNIX operating systems to change the working directory. It can be used both on a command line and in a script, either the UNIX shell or a DOS BAT.

C. Software

1) Ansible: Ansible was the main software to be able to develop this project, after several days of research it was concluded that Ansible would be a good tool for process automation, thus competing with popular tools such as Puppet or Chef. According to Red Hat Ansible is a free software platform for configuring and managing computers. It combines multi-node installation, ad hoc task executions, and configuration management. Additionally, Ansible is categorized as an orchestration tool. For effective task, Ansible generates fully editable action branches that operate simultaneously, allowing an operation to be executed in a short time and with excellent management throughout the process. Each package of orders or operations is known as a ”playbook”, which is easily transferable to any member of the team and has the power to be interpreted without problems.

- Requirements: Ansible can run on any operating system, such as Windows, Linux, Ubuntu or iOS. It can also be installed in any cloud or operating framework of a system. The only essential requirement for the tool to work is to have Python 2.4 or later.
- Advantages:
  - Techniques:
    - ∗ Allows flexible use of APIs, modules and plugins.
    - ∗ It does not require a root user or software installation on the machines it manages.
    - ∗ It has a ssh authentication method with parallel keys.
    - ∗ It allows to configure complex tasks using YAML language (Playbooks).
    - ∗ It offers the implementation of a fleet of remote servers, to which software can be installed and controlled remotely.
  - Qualitative:
    - ∗ Allows disaster recovery.
    - ∗ Offers higher productivity.
    - ∗ Improve the relationship with the work team.
    - ∗ It allows you to locate errors and perfect a procedure.
    - ∗ It is adapted according to the needs of the project.

2) Apache: During the first tests with Ansible, Apache Web Service was used. Apache HTTP Server is free and open source web server software for Unix platforms. It allows website owners to serve content on the web, hence the name ”web server”.

3) Docker: Docker is an open source software in which we can deploy applications as something portable which is named images, Docker is a software that is self-sufficient since everything is contained in a ”container” o ”instance” that can be run on the cloud or locally. Docker has many benefits for both the user and the machine, such as:

- Docker makes distribution easy.
- Docker deployment is very fast.
- The layering system in Docker is very optimal.
- Docker takes full advantage of the server.

4) GitHub: Currently GitHub is a very popular system among programmers since it is a project management and code version control system, as well as a social network platform designed for developers. Like other platforms, GitHub has several advantages such as:

- You can work in the offline
- You can display your work
- GitHub provides you with constant information on the activities around a repository.

- In case there is more than one person working on the same project, GitHub allows you to keep track of all the versions
- Thanks to being cross-platform, it can be used to create local repositories on all operating systems: Windows, Linux or Mac.
- It is a completely free and unlimited tool for public projects. Similarly, there is a type of paid version in which all your work becomes private

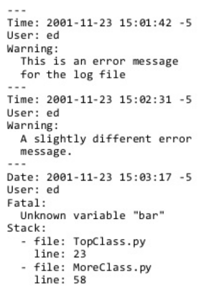
GitHub was used in this project as a delivery system for progress on the project, in the same way it worked as a system to make code queries for the project.

Fig. 6. Software used

D. Query Language

During the project several language formats were used for each software, they are similar to bash but have a touch of their own software.

1) YAML: The project consisted for the most part using the Ansible syntax, this format is saved in several files called playbooks with which we can run to provide some task to a server. YAML is a format for storing data objects with a tree structure. Its acronym stands for YAML Ain't Markup Language. In the same way, it can be called YML.



```
---
Time: 2001-11-23 15:01:42 -5
User: ed
Warning:
  This is an error message
  for the log file
---
Time: 2001-11-23 15:02:31 -5
User: ed
Warning:
  A slightly different error
  message.
---
Date: 2001-11-23 15:03:17 -5
User: ed
Fatal:
  Unknown variable "bar"
Stack:
  - file: TopClass.py
    line: 23
  - file: MoreClass.py
    line: 58
```

Fig. 7. YAML example

E. Docker Query Langueage and Git Bash

In the same way, the Docker and Git Bash formats were used to carry out the project, simple commands to create files. Git bash allows us to control git commands to simplify the actions we want to do.

F. Connection software or protocol

1) VPN (OpenVPN): In order to carry out this project and be able to do tests, the Rich IT company lent me two of their available servers with which I could enter and use at any time. The way in which I could connect to those servers was through a VPN connection (Virtual Private Network). The VPN connection allows us to create something similar to a local network but without the need to be in the same place, it does so through the internet with which we can have access to the company's data that are available to the user. In this case, for the project we use the ”OpenVPN” software. In other words, a VPN connection allows us to:

- Access remote company services or equipment.
- Connect two local networks of the same company using their internet channels.
- Can be used on various platforms, be it Windows, Linux, Android, iOS.
- Allows a person to work from home.

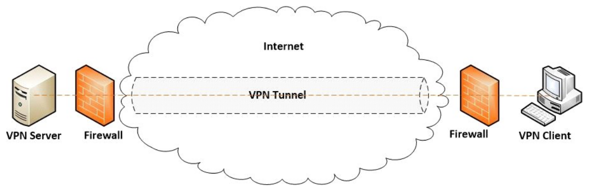


Fig. 8. VPN Diagram

2) SSH protocol: SSH or Secure Shell, is a remote administration protocol that allows users to control and modify their remote servers over the Internet through an authentication mechanism. Provides a mechanism to authenticate a remote user, transfer input from the client to the host, and relay the output back to the client. It was by this means that I was able to connect to the servers that the company lent me to carry out the project. The SSH format is shown as follows:

- ssh user @ host

When we put this command the ”ssh” tells the system to open a Shell connection which has to be encrypted and secure. The ”user” section indicates the user with which it will be entered, when the desired server is entered, the person who has entered will be shown. Finally the ”host” tells us the server we want to enter, it can be an IP address like: 192.168.15.11 or a domain like: www.texto.com. Once all this has been entered, the server will ask us for the password to be able to have access with which if it is correct we will immediately realize that we are already inside the server since it shows us our user.

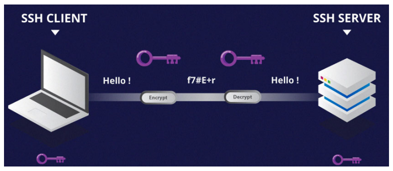


Fig. 9. SSH Diagram

V. Development

A. Stage 1

The work was divided into 3 stages, the first stage was carried out during the first week in which I began to investigate the process automation tools, and the DevOps methodology since this project was related to that concept. It was agreed that during this project there would be two meetings per week to show progress and assign tasks. During this stage I showed a small presentation about the DevOps methodology and automation tools, it was concluded that the Ansible tool would be the best option to learn since it is intuitive, fast and effective, in the same way it is the same software that the company was experimenting for its application. I base myself on several official pages such as Chef, Puppet and some similar.

1) The connections: In the same way, as the first steps, they gave me two servers with which I connected through a VPN connection, the application was provided by them (OpenVPN), here they gave me the password and some permissions, as the second step was to connect to the servers , with which they provided me with the usernames and password of two servers in which one would be to make code and the other would be to implement the code, the connection to these servers was through the SSH protocol that I have explained previously. These two servers ”Slave” and ”Master” are with which I was going to carry out my project.

B. Stage 2

This section was the application since it was where I started using Ansible. They gave me a series of tasks with which to address the basics of project automation, which were:

- Install Ansible (Master).
- Configure Ansible (Master).
- Install Apacher web (slave).
- Server restart with Ansible..
- Server shutdown with Ansible.
- Restart Apache web with Ansible.
- Turning on Apache web with Ansible.
- Shutting down Apache web with Ansible.

1) Ansible (Master): The servers made use of the Linux distribution ”CentOs” with which I was not completely familiar but it was easy to approach this distribution, in the same way they provided me documentation about CentOs. The first part was to install Ansible which was simple: ”yum install ansible” and as the next step just check that the version of ansible is the most recent.

2) Configure Ansible (Master): This step was one of the longest and most difficult in my opinion since I did not have a notion about Ansible, therefore I made use of the official Ansible documentation to find out how to configure it. In this step I had to connect the two servers through ansible, so I could provide all the necessary tasks from the ”Master server to the” Slave ”server. Ansible has a format which is called” Playbook ”with what I was going to work throughout the project Once well researched, Ansible has a section called ”Host” in which we have to put the machines that we will manage, to be able to manage them you have to enter them with a certain format, currently you can connect through of the SSH protocol or by entering ip direction like:

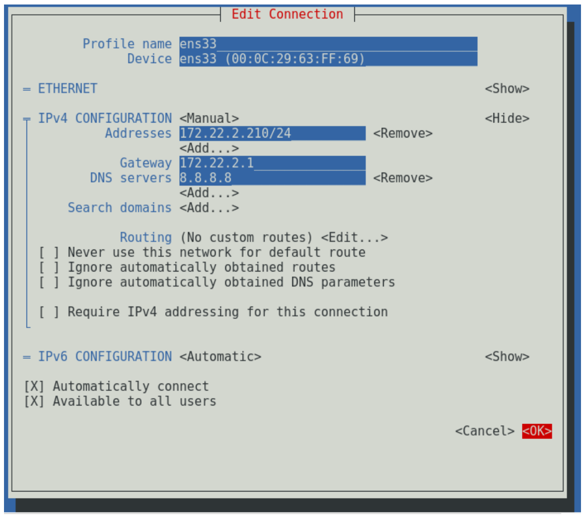
[Slave] – Group

192.168.15. – Ip direction

During this process it took me several days since I could not connect to the first since I tried to connect through the SSH protocol but for some reason Ansible did not allow it, therefore I used the ip method to connect, to make sure that the servers had communication I used the ”ping pong” method for sa to get used to using Ansible. Once the server is connected through ansible we can manage the entire server through tasks in ”playbook” files, the only thing to do is start Ansible and create a playbook like: test.yml and to run said playbook we simply start ansible and we warn you that we will run a playbook: ansible-playbook test.yml. Once all this is done, what remains is to create the different tasks that we want the server to do.

3) Apache web: To install Apache it can be done in two ways, one is from the same server using the install command or it can be done from Ansible created a playbook. The format for shutting down, restarting or starting the Apache service is similar. In order to create the yml files, we must take into account the distribution of the server that we are going to administer since the commands vary. Once having this knowledge we can provide any type of compatible software to any server that we have within our server list, we can assign specific servers so that they can be administered.

The reason this phase took me 2 weeks is because I had zero knowledge about Ansible. In the same way, there were some problems with the servers during this time since the connections were lost. In this process I used ”nmtui” from CentOs 7 which uses ncurses and allows us to easily configure the connection types from the terminal and without additional dependencies. It offers a graphical, text-based interface for the user to make these modifications. This was the method I used to be able to reconnect with the servers (Modifying some parameters of the servers)



```
                         Edit Connection

      Profile name ens33
            Device ens33 (00:0C:29:63:FF:69)

═ ETHERNET                                                <Show>

╤ IPv4 CONFIGURATION <Manual>                             <Hide>
│          Addresses 172.22.2.210/24           <Remove>
│                    <Add...>
│            Gateway 172.22.2.1
│        DNS servers 8.8.8.8                   <Remove>
│                    <Add...>
│     Search domains <Add...>
│
│            Routing (No custom routes) <Edit...>
│ [ ] Never use this network for default route
│ [ ] Ignore automatically obtained routes
│ [ ] Ignore automatically obtained DNS parameters
│
│ [ ] Require IPv4 addressing for this connection
└

═ IPv6 CONFIGURATION <Automatic>                          <Show>

[X] Automatically connect
[X] Available to all users

                                                  <Cancel> <OK>
```

Fig. 10. nmtui interface

C. Stage 3

This process was a bit more complicated since the objective was to automate software such as Docker and GitHub to deploy applications, in this phase the tasks were:

- Playbooks Administration
- Playbook example with Git.
- Example with Docker

1) Playbook Administration: The concept of managing playbooks refers to creating a playbook that reuses old playbooks, that way it is not necessary to retype all the commands from scratch. For this step it is recommended to always have a folder of playbooks to simply add the old playbooks.

2) Docker and Git: The Docker and Git section was in order to manage the software from Ansible by executing some type of command to run a container or make a request for Git. This process was the most difficult.

During the process, I was able to install Docker from Ansible and in the same way in Docker I created a container with an application inside (Wordpress) in order to deploy that container from Ansible.

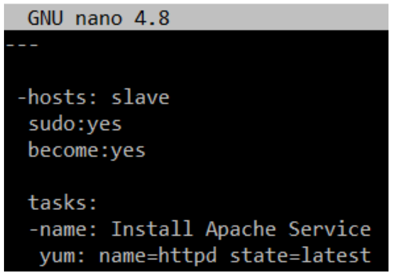
With Git I had to install Git Bash to be able to catch up and together with Ansible I did a git clone, git commit, git checkout and git push which did not finish in its entirety for the time that was short.

VI. Results

The results were enough since they were several tests.

1) Stage 1: During the first stage, I acquired knowledge on the subject and in the same way the advantages of the Ansible tool over others, it was something more theoretical.

2) Stage 2: As I mentioned earlier, this stage was a bit difficult since I didn't have any knowledge but I got good results. In the first image you can see how to install Apache web from ansible. In the same way, to avoid errors, a good practice is to assign the ”sudo” mode at once and the ”become” means that it accepts everything necessary.

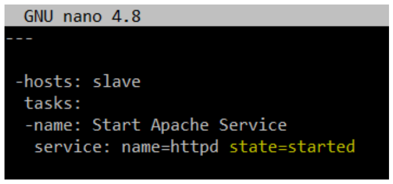


```
 GNU nano 4.8
---

-hosts: slave
 sudo:yes
 become:yes

 tasks:
 -name: Install Apache Service
  yum: name=httpd state=latest
```

Fig. 11. Install Apache web With Ansible

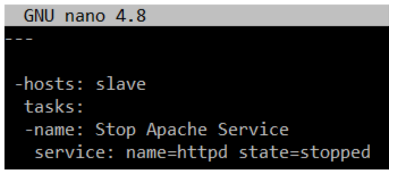
The following images is the yml file (playbook) to be able to start the Apache service, in the same way to be able to shut down and restart the service is in a similar way, the only thing that needs to be modified for each playbook is the ”status” like, ”started” , ”stopped”, ”restarted”, those are the states to be able to modify the Apache service. The dashes shown at the beginning ”- - -” means that we are starting a playbook.

Keep in mind that it is not good practice to have only one playbook with a status and modify it according to our needs,



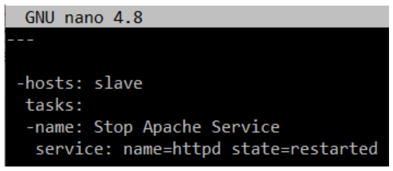
```
 GNU nano 4.8
---

-hosts: slave
 tasks:
 -name: Start Apache Service
  service: name=httpd state=started
```

Fig. 12. Start Apache Service (playbook)



```
 GNU nano 4.8
---

-hosts: slave
 tasks:
 -name: Stop Apache Service
  service: name=httpd state=stopped
```

Fig. 13. Stop Apache Service (playbook)



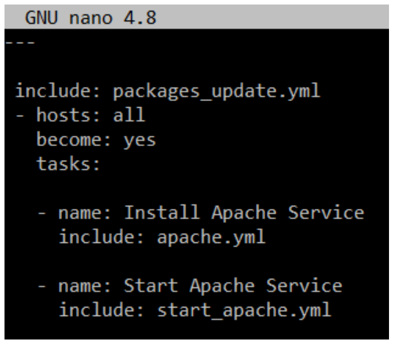
```
 GNU nano 4.8
---

-hosts: slave
 tasks:
 -name: Stop Apache Service
  service: name=httpd state=restarted
```

Fig. 14. Restart Apache Service (playbook)

you always have to have a playbook for each option even if the task is small.

A. Stage 3

During this part it was proposed that it be something more experimental, since it was a little more complicated on the part of Docker and Git. In the same way I could perform the administrations of playbooks, it refers to reusing old playbooks. During this process use the apache installation, before installing it update the CentOs system and finally when everything is ready the Apache service would start thanks to another playbook.

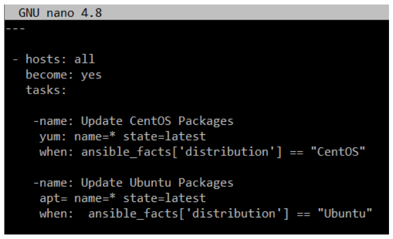


```
 GNU nano 4.8
---

include: packages_update.yml
- hosts: all
  become: yes
  tasks:

  - name: Install Apache Service
    include: apache.yml

  - name: Start Apache Service
    include: start_apache.yml
```

Fig. 15. Playbooks Administration

In this image (image 16) you can see that there is a conditional ”when”, this playbook is made to work in two distributions, both for CentOS and Ubuntu, when the playbook is run the commands will be read and depending on the distribution the update will be applied, the rest will be ignored.

The ”include” command adds the called playbooks, therefore if a playbook does not exist, Ansible will flag an error.



```
 GNU nano 4.8
---

 - hosts: all
   become: yes
   tasks:

    -name: Update CentOS Packages
     yum: name=* state=latest
     when: ansible_facts['distribution'] == "CentOS"

    -name: Update Ubuntu Packages
     apt= name=* state=latest
     when:  ansible_facts['distribution'] == "Ubuntu"
```

Fig. 16. Packages Update

As for the playbook that can be noticed, ”apache.yml”, ”start-apache.yml” are the same playbooks as the images above (image 11, image 12) only that they have a modification, which is the ”when” conditional.

It is good practice to put conditionals in the playbooks since if we have several types of distributions we may have some type of error or in the same way we would have to create several types of playbooks for each distribution.

B. Clarifications

On the part of Docker and Git the advances were very little due to the time and complexity. An agreement was reached to implement these software later on as it was a little more advanced than what was learned. The most advanced was with Docker since I created the containers with an image of an application and in the same way I could deploy it but I did not get to implement it together with Ansible.

These were the results of the final project which was very tedious as there is not much information from Ansible other than its documentation. It is worth mentioning the results of the project were the playbooks (all work in their entirety) and their application on the ”slave” server.

VII. Conclusion

The tools for data automation are something that today are becoming more and more popular, at first my knowledge about these tools was null and I had to learn everything possible to carry out this project. I researched and learned a lot of concepts that I was not aware of. As a final result of this project, I understood the importance of the DevOps methodology. DevOps is something that all companies should have especially companies dedicated to software since it is a way to be resourceful and dynamic at work as well as bringing great benefits to work performance. In the same way, I believe that it is necessary for all companies to have some automated actions since it saves a lot of time and money. Not many people know about automation tools and it is something that engineers should learn to use from their studies. The results of this project were excellent in my opinion. Something very interesting that I noticed during this project is that there is not much information about the application of automation tools, on the contrary, it is very scarce and in many occupations the only information that exists is the documentation. It is worth mentioning that during the project, several talks were also taken about Big Data, Data Governance, Automation and others. To finish, I just emphasize that it is important to learn more tools that help us for our career.

Appendix A
The DevOps Tools

This is a diagram showing the different tools that are used for the DevOps cycle. It is important to learn about any of these tools and, in the same way, to know which tool suits us best for each phase of DevOps.

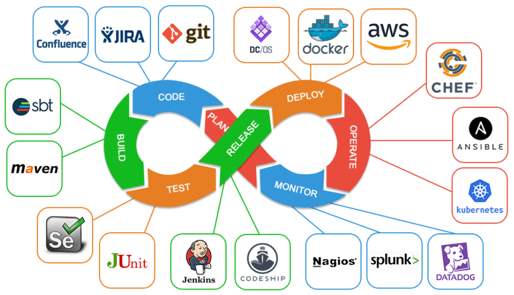


Fig. 17. DevOps Tools

Acknowledgment

I would like to thank Rich It for allowing me to do this project together with them since I am very enthusiastic about learning new things that have to do with my career in the same way, teacher Juan Vazquez already helped us contact the Rich It companies and he the one who helped us prepare the documents. In the same way, thanks to the members of Rich IT, Axel Gonzalez, Francisco Garcia, Daniel Molinari and Marko Douda since they were the ones who guided and motivated me during the stays. It was a pleasant experience and I hope it will be repeated. Lastly, to my partner Danilo Cuevas since thanks to him I was able to be in this project.y

References

[1] Red Hat, *Ansible Documentation*, Docs.ansible.com, 2020. [Online]. Available: https://docs.ansible.com/ansible/latest/index.html. [Accessed: 25- Aug- 2020]

[2] Red Hat, *AUTOMATION*, redhat.com, 2020. [Online]. Available:https://www.redhat.com/es/topics/automation/whats-it-automation [Accessed: 25 Aug 2020]

[3] GNU, *GNU Bash*, 2015,[Online]. Available:https://www.gnu.org/software/bash/ [Accessed: 25 Aug 2020]

[4] Docker, *docker images*, docs.docker.com, 2013, [Online]. Available:https://docs.docker.com/engine/reference/commandline/images/ [Accessed: 25 Aug 2020]

[5] AWS, *What is DevOps?*,aws.amazon.com, 2020, [Online]. Available:https://aws.amazon.com/es/devops/what-is-devops/[Accessed: 25 Aug 2020]

[6] Git, *Git Documentation*,git-scm.com, 2017 [Online.] Available:https://git-scm.com/docs/[Accessed: 25 Aug 2020]

[7] Julio Sanz, *Network Manager Text User Interface (NMTUI): configurando interfaces de red en Red Hat y derivadas sin despeinarte*, El array de Jota, 11-Nov-2018. [Online]. Available: https://www.elarraydejota.com/network-manager-text-user-interface-nmtui-configurando-interfaces-de-red-en-red-hat-y-derivadas-sin-despeinarte/. [Accessed: 25-Aug-2020].

[8] Docker *What is a Container?*, Docker.com , 2019. [Online]. Available: https://www.docker.com/resources/what-container. [Accessed: 25-Aug-2020].

[9] R. H. Ansible, *Ansible and Docker* 2015. [Online]. Available: https://www.ansible.com/integrations/containers/docker. [Accessed: 25-Aug-2020].

[10] Centos, *CentOS Documentation Home* CentOS.org, 2018. [Online]. Available: https://docs.centos.org/en-US/docs/. [Accessed: 25-Aug-2020].