Unleashing Ansible with Cloud for Automation

Yashaswini J.
Department of Computer Science and Engineering
JSSSTU, Mysore, India
Dr. M. P. Pushpalatha
Professor and Head of the Department
Department. of CS&E
JSS ST&U, Mysuru.

Abstract: As the proliferation of cloud, Virtualization, and automation is increasing in today's information technology environment, the need arises to abide with this dominance. In this paper, our focus is on downloading the cloud builds from Jfrog artifactory using the URL provided and store it in a prescribed path locally. And, the downloaded file must be installed. To do this, we go with the configuration management tool- Ansible (DevOps tool) which reduces human- intervention when needed to be performed for more than one machine, which reduces a tedious repetitive task. And for the ease of use, we have built a GUI for the same.

Keywords: Configuration Management Tool (CMT), Playbooks, Infrastructure as Code (IaC), YAML.

Introduction: Earlier, if we want to perform any specific configuration management tasks, Ex: Installation of a software then, we were logging- in into that particular machine and we were checking if that specific software is already present in that machine and if not we were proceeding with the installation procedure. And if the software is already present, then we were going to check with the recent version of it. If it is upgraded with latest version then Yes, we are done; and if not, we were going with installation of the same.

And now consider a scenario where this procedure (Installation) must be done for hundreds or thousands of machines in an organization, then instead of doing it manually i.e., by entering into each and every system manually and check if or if not the machine is having a particular software or not, people started doing automation of this repetitive task.

To start with automation, we have many languages and tools which are already available respectively.

- Python Scripting
- Perl Scripting
- Ruby Scripting
- YAML Scripting
- Shell Scripting

And,

- Chef
- Puppet
- SaltStack
- Ansible

Among them we have chosen Ansible- which is one of the most efficient and popular CMT, and easy- to- understand, human- readable data- serialization language- YAML.

Ansible is a popular, open- source tool to automate (like, deploy apps, managing systems, and crush up complexity), accelerate (solve the problem once and share with everybody), collaborate (breaking down the silos and create an automation culture), integrate (automate the technology you are using) of what we call IaC. Ansible is heading towards unraveling the mystery of how exactly the work is done and also, it's doing a wonder just by turning tough tasks into simple playbooks.

YAML is an acronym for Yet Another Markup Language which is popularly known to be a data- serialization language, is often utilized mostly to create- the configuration files and works accordingly and in concurrence with any programming language, with minimal syntax and which can also be interpreted as a strict superset of JavaScript Object Notation (JSON).

Coming to a background, here in this paper we introduce a complete web- framework/ platform by using which we can download and install few of the files respectively using Ansible as a CMT tool to perform this job. The remnant of the paper, is structured as follows;

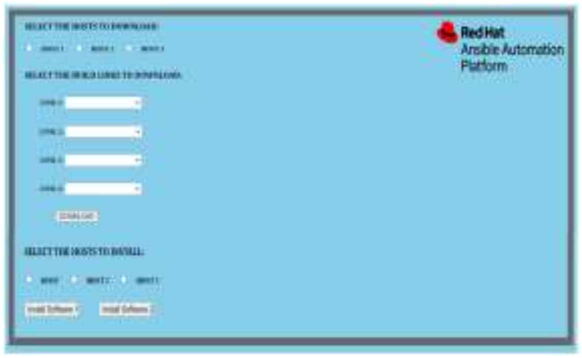
In Section I, we focus on the complete purpose of this entire paper.
In Section II, we discuss about the design of front- end in which we have used very basic HTML and CSS by which, the GUI looks user- friendly.
In Section III, we walk through the CGI scripting portion which is done via python.
In Section IV, we take a closer look at writing and executing Ansible- Playbook for required purpose.

I. Intended Scenario:

In the cloud environment, The Research and Development engineers must get some builds from sub- ordinate teams to get their work done; on all the machines which they are working upon. The idea was to downlink and place those build's files in a prescribed location locally on the host/ server. And also, to enthrone few of the executable files to install the software, but only for a selected host or also for a selected group of hosts/ machines, using Ansible as a management tool. To make trouble- free use of Ansible- playbook, we have designed a tranquil web- interface.

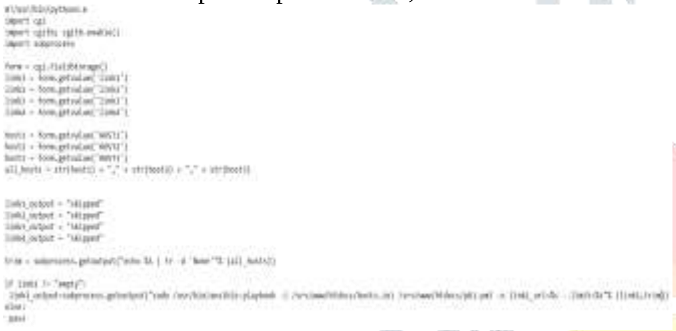
II. Front- end Development:

To build a user- friendly GUI, we have used option- buttons, check- boxes for seamless selection and easy usage and few submit buttons. The sample image of web- Interface looks like;

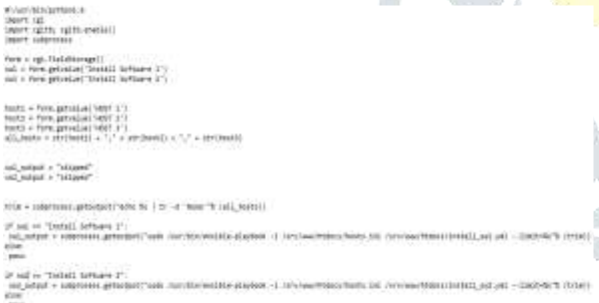


III. CGI code using python:

Here, what we are deliberated to do is, triggering the Ansible- Playbooks by taking the user's input from the GUI based environment. Sample script looks like;



```
#!/usr/bin/python3.6
import cgi
import cgitb; cgitb.enable()
import subprocess

form = cgi.FieldStorage()
link1 = form.getvalue('link1')
link2 = form.getvalue('link2')
link3 = form.getvalue('link3')
link4 = form.getvalue('link4')

host1 = form.getvalue('HOST1')
host2 = form.getvalue('HOST2')
host3 = form.getvalue('HOST3')
all_hosts = str(host1) + "," + str(host2) + "," + str(host3)


link1_output = "skipped"
link2_output = "skipped"
link3_output = "skipped"
link4_output = "skipped"

trim = subprocess.getoutput("echo %s | tr -d 'None'" % (all_hosts))

if link1 != "empty":
 link1_output=subprocess.getoutput("sudo /usr/bin/ansible-playbook -i /var/www/html/hosts.ini /var/www/html/p01.yml -e [illegible]" % ([illegible]))
else:
 pass
```



```
#!/usr/bin/python3.6
import cgi
import cgitb; cgitb.enable()
import subprocess

form = cgi.FieldStorage()
sw1 = form.getvalue("Install Software 1")
sw2 = form.getvalue("Install Software 2")

host1 = form.getvalue('HOST 1')
host2 = form.getvalue('HOST 2')
host3 = form.getvalue('HOST 3')
all_hosts = str(host1) + "," + str(host2) + "," + str(host3)

sw1_output = "skipped"
sw2_output = "skipped"

trim = subprocess.getoutput("echo %s | tr -d 'None'" % (all_hosts))

if sw1 == "Install Software 1":
 sw1_output = subprocess.getoutput("sudo /usr/bin/ansible-playbook -i /var/www/html/hosts.ini /var/www/html/install_sw1.yml --limit=%s" % (trim))
else:
 pass

if sw2 == "Install Software 2":
 sw2_output = subprocess.getoutput("sudo /usr/bin/ansible-playbook -i /var/www/html/hosts.ini /var/www/html/install_sw2.yml --limit=%s" % (trim))
else:
 pass
```

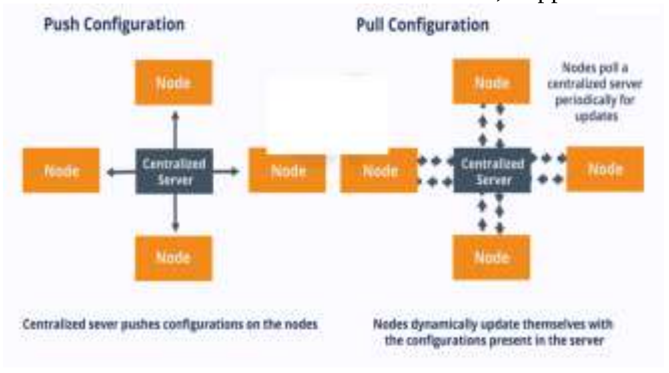
Necessary modules found:

- ✓ import cgi: CGI (Common Gateway Interface) scripts are supported by this module. This module defines a set of tools for use by Python-based CGI programmes. An HTTP server runs a CGI script, which is commonly used to process user input submitted via HTML or an element.
- ✓ FieldStorage(): to access form data that has been submitted. Use the encoding keyword parameter set to the value of the document's encoding if the form contains non-ASCII characters.
- ✓ subprocess(): The Python subprocess module is used to execute new apps or programmes by generating new processes from Python code. You can use the subprocess module to create new processes, connect to their input/output/error pipes, and get their return codes.

IV. Ansible- heart and soul of this context:

i. Configuration Management Tool are of basically 2 types:

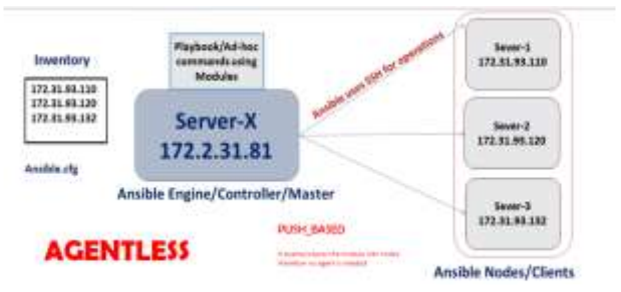
- Push based CMT: Ansible, SaltStack.
- Pull based CMT: Chef, Puppet


Push Configuration
Pull Configuration
Node
Centralized Server
Nodes poll a centralized server periodically for updates
Centralized sever pushes configurations on the nodes
Nodes dynamically update themselves with the configurations present in the server

ii. Why we have chosen Ansible:

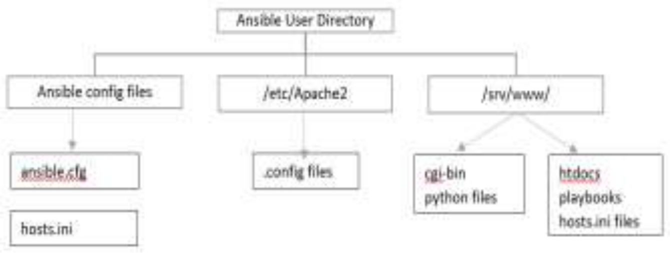
- Agentless: Information is transferred to or gathered from computers without the use of proprietary agents in an agentless design. Ansible uses SSH to push changes from a single source to numerous distant resources, making it "agentless."
- Push- Based: The primary server (where the configuration data is kept) pushes the configuration to the node in this sort of configuration management solution. As a result, communication is initiated by the main server rather than the nodes. This implies that each node may or may not have an agent/client installed.
- Idempotency: Idempotent operations, such as multiplication by zero, can be applied numerous times without altering the result beyond the first application.
- Reusable: One of the objectives we established for our deployment automation project was that every script created may be used for two purposes: deploying a new version of an application and provisioning new servers with that application.
- Less time – More work.

iii. How does Ansible works?



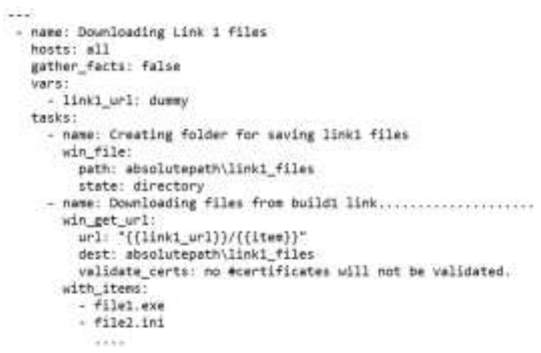
The above diagram depicts the architecture of ansible tool: There will be an Ansible- Engine from where we can control and configure hundreds and thousands of machines simultaneously. And there are hosts at the right side called Ansible- nodes (here there are 3 and we can take any numbers of nodes as we want). The connection between Engine and the Nodes can be established by SSH (in Linux) and WinRM/ OpenSSH (in Windows). And we write a code (here it's called as Playbook) on Ansible- Engine and it is copied or pushed onto the Ansible- Nodes and therefore it is called as push-based configuration management tool. Agentless because we need not to install any extra software on node's side. There are two important files to note here; ansible.cfg (where all the info related to ansible- nodes resides) and inventory (where the IPs or FQDN of the ansible- nodes resides). The working: mean to say whenever we run Ad- hoc commands or playbook, how they get executed on Ansible- managed- nodes. Note: Only Linux like systems supports installation of Ansible and can act as Ansible- Engine. And to be ansible- nodes any OS is fine. (RHEL, Linux, Ubuntu, Windows or even the combinations)

iv. Structure of Ansible- user directory:



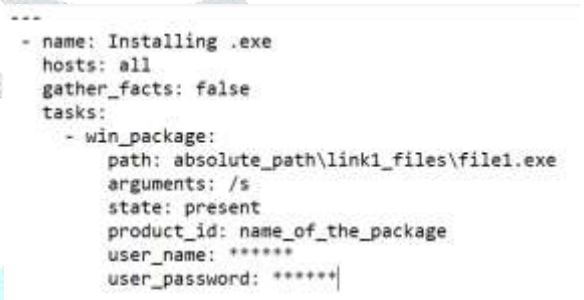
v. Playbook snippets for actual tasks:

Download files from JFrog artifacory :



```
---
 - name: Downloading Link 1 files
   hosts: all
   gather_facts: false
   vars:
     - link1_url: dummy
   tasks:
     - name: Creating folder for saving link1 files
       win_file:
         path: absolutepath\link1_files
         state: directory
     - name: Downloading files from build1 link....................
       win_get_url:
         url: "{{link1_url}}/{{item}}"
         dest: absolutepath\link1_files
         validate_certs: no #certificates will not be validated.
       with_items:
         - file1.exe
         - file2.ini
           ....
```

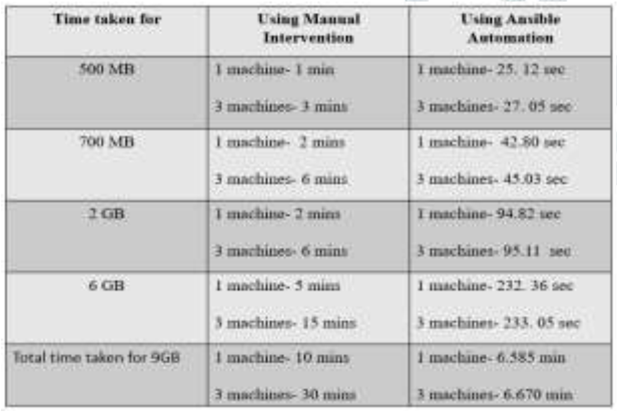
Installation of .exe files which is already downloaded and saved locally:



```
---
 - name: Installing .exe
   hosts: all
   gather_facts: false
   tasks:
     - win_package:
         path: absolute_path\link1_files\file1.exe
         arguments: /s
         state: present
         product_id: name_of_the_package
         user_name: ******
         user_password: ******
```

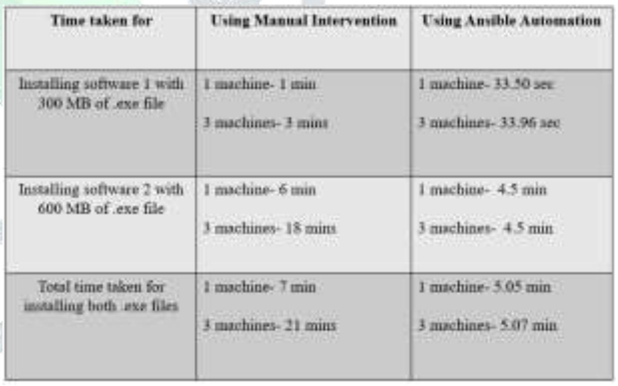
vi. Time taken charts:

Time for download:



Time taken for	Using Manual Intervention	Using Ansible Automation
500 MB	1 machine- 1 min 3 machines- 3 mins	1 machine- 25. 12 sec 3 machines- 27. 05 sec
700 MB	1 machine- 2 mins 3 machines- 6 mins	1 machine- 42.80 sec 3 machines- 45.03 sec
2 GB	1 machine- 2 mins 3 machines- 6 mins	1 machine- 94.82 sec 3 machines- 95.11 sec
6 GB	1 machine- 5 mins 3 machines- 15 mins	1 machine- 232. 36 sec 3 machines- 233. 05 sec
Total time taken for 9GB	1 machine- 10 mins 3 machines- 30 mins	1 machine- 6.585 min 3 machines- 6.670 min

Time for installation:



Time taken for	Using Manual Intervention	Using Ansible Automation
Installing software 1 with 300 MB of .exe file	1 machine- 1 min 3 machines- 3 mins	1 machine- 33.50 sec 3 machines- 33.96 sec
Installing software 2 with 600 MB of .exe file	1 machine- 6 min 3 machines- 18 mins	1 machine- 4.5 min 3 machines- 4.5 min
Total time taken for installing both .exe files	1 machine- 7 min 3 machines- 21 mins	1 machine- 5.05 min 3 machines- 5.07 min

Conclusion:

Finally, we can say that Ansible is a straightforward and powerful tool for configuration management and automation. Ansible, on the other hand, is a newcomer to the market and must contend with established competitors.
On the other hand, the rising interest in Ansible because of its adoption by well-known organizations like as NASA has the potential to turn the tables. Ansible's many features, including as provisioning, orchestration, application deployment, and security and compliance, demonstrate its versatility. Ansible's skills can be efficiently translated into a comprehensive DevOps tool.

References:

1. https://docs.ansible.com/
2. https://docs.ansible.com/ansible/latest/collections/ansible/windows/win_copy_module.html
3. https://www.middlewareinventory.com/blog/ansible-get_url-examples-how-to-download-file-from-url/
4. https://www.tutorialspoint.com/yaml/yaml_comments.htm
5. https://www.middlewareinventory.com/blog/ansible-get_url-examples-how-to-download-file-from-url/
6. https://www.suse.com/search/?s=get_url#gsc.tab=0&gsc.q=get_url&gsc.page=1
7. https://docs.ansible.com/ansible/latest/collections/ansible/builtin/items_lookup.html
8. https://www.suse.com/search/?s=get_url#gsc.tab=0&gsc.q=get_url&gsc.page=1
9. https://stackoverflow.com/questions/61948417/how-to-measure-and-display-time-taken-for-tasks-when-running-ansible-playbook
10. https://stackoverflow.com/questions/36696952/copy-multiple-files-with-ansible
11. https://www.danielhall.me/blog/ansible-pattern-downloading-artifacts/
12. https://docs.ansible.com/ansible/latest/collections/ansible/windows/win_package_module.html
13. https://docs.ansible.com/ansible/latest/collections/community/windows/win_psexec_module.html
14. https://www.tecmint.com/zypper-commands-to-manage-suse-linux-package-management/
15. https://forums.opensuse.org/showthread.php/455686-Where-is-the-trash-folder