PS - Ansible challenge

[1. Overview 1](#_Toc73977505)

[2. Approach 2](#_Toc73977506)

[3. Environment preparation 3](#_Toc73977507)

[Cloudmachine Application 3](#_Toc73977508)

[Cloudmachine access 3](#_Toc73977509)

[4. Ansible controller host 4](#_Toc73977510)

[Inventory for managing hosts 4](#_Toc73977511)

[Ping from ansible 5](#_Toc73977512)

[5. AppD Controller 6](#_Toc73977513)

[TIP 6](#_Toc73977514)

[6. Machine-agents installation 7](#_Toc73977515)

[TIP 7](#_Toc73977516)

[7. Database agent installation 7](#_Toc73977517)

[8. Application component: Tomcat 7](#_Toc73977518)

[TIP 8](#_Toc73977519)

[9. Application component: JBOSS 9](#_Toc73977520)

[10. Application component: IIS application pools 10](#_Toc73977521)

[11. Application component: BDR 11](#_Toc73977522)

[Application 11](#_Toc73977523)

[Install agent with ansible 11](#_Toc73977524)

[TIP 11](#_Toc73977525)

[12. Application component: .net core on linux 12](#_Toc73977526)

[Application 12](#_Toc73977527)

[.NET Core agent installation 12](#_Toc73977528)

[Generating load 13](#_Toc73977529)

[TIP 14](#_Toc73977530)

[Conclusion 15](#_Toc73977531)

[LEARN ANSIBLE 15](#_Toc73977532)

## Overview

Imagine that you have a customer with huge multi technology business application. Moreover, that application tend to be maintained automatically. Automation as one of the CI/CD part chosen to be ansible based. Talking about technologies that application have, it’s based on:

.Net IIS application on windows

.Net core application on linux

Single java application

Java app on tomcat

Java app on jboss wildfly

Customer team don’t want to spend a time performing all jobs manually, so they want to make automation based on ansible roles.

To make sure that PS AppD make all things live, customer provides each technology based application component on development environment where you might proof your power.

## Approach

Approach is to use ansible.

Let me take a couple of moments and introduce you ansible,



From it’s own words:

Ansible is an IT automation tool. It can configure systems, deploy software, and orchestrate more advanced IT tasks.

Ansible’s main goals are simplicity and ease-of-use. It also has a strong focus on security and reliability, featuring a minimum of moving parts, usage of OpenSSH for transport (with other transports and pull modes as alternatives), and a YAML language.

Ansible manages machines in an agent-less manner from a control node.

Here are main concepts of Ansible <https://docs.ansible.com/ansible/latest/network/getting_started/basic_concepts.html>

Getting back to our main challenge, Appdynamics smart minds designed roles that available from ansible-galaxy appdynamics collection and could be used in your playbooks for automating deployment.

<https://www.ansible.com/overview/how-ansible-works>

Sometimes world pushing hard, and some of application components couldn’t be covered easily with automation because of different starting up way and complex settings.

## Environment preparation

### Cloudmachine Application

On the cloudmachine you may find template to start with:

Graphical user interface, text, application

Description automatically generated

Called ansible-automation-challenge-0.6

It contains 7 VMs for:

* Java app on tomcat
* Java app on jboss wildfly
* .Net on windows (standard onboarding lab)
* .Net core on linux (Roman Lahinousky netcore challenge)
* Single java application (BDR onboarding lab)
* AppDynamics controller
* Ansible controller host

Create your own application based on that template to participate the challenge.

Some of application components are exposed to interact with users, others represent API that could be pushed with a loadgenerator scripts, also you might see an application that generates load by itself, without any help.

Each application details would be covered in separate section.

### Cloudmachine access

Connect to proper vpn to access VMs

Cicso using anyconnect to access cloudmachine hosts:

<https://confluence.corp.appdynamics.com/display/NETENG/Cisco+AnyConnect+for+ves-appd>

Just add one of following urls in your anyconnect to access application template.

|  |  |  |
| --- | --- | --- |
| URL: | Location: | AnyConnect menu Option: |
| [sjc-cci-vpn-cluster-appd.cisco.com/ssl](https://sjc-cci-vpn-cluster-appd.cisco.com/ssl) | San Jose | ~ CCI AppDynamics - San Jose Duo |
| [bgl-cci-vpn-cluster-appd.cisco.com/ssl](https://bgl-cci-vpn-cluster-appd.cisco.com/ssl) | Bangalore | ~ CCI AppDynamics - Bangalore Duo |
| [Aer-cci-vpn-cluster-appd.cisco.com/ssl](https://aer-cci-vpn-cluster-appd.cisco.com/ssl) | Almere | ~ CCI AppDynamics - Almere Duo |

To access machines use generic-vm private key that designed for public use on cloud machine templates.

## Ansible controller host

As far as main approach is to use ansible, your initial and final point would be here, on ansible controller.

Everything is prepared on that host to start help you automate.

[centos@ip-10-97-10-167 ~]$ ansible --version

ansible [core 2.11.1]

config file = None

configured module search path = ['/home/centos/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']

ansible python module location = /usr/local/lib/python3.6/site-packages/ansible

ansible collection location = /home/centos/.ansible/collections:/usr/share/ansible/collections

executable location = /usr/local/bin/ansible

python version = 3.6.8 (default, Aug 24 2020, 17:57:11) [GCC 8.3.1 20191121 (Red Hat 8.3.1-5)]

jinja version = 2.10.1

libyaml = True

[centos@ip-10-97-10-167 ~]$ ll ansible-challenge/

total 0

[centos@ip-10-97-10-167 ~]$

### Inventory for managing hosts

First of all you need to determine some managed hosts for ansible in inventory file.

What you need is to create inventory file somewhere. It could it be /etc/ansible/hosts or it could be placed just in challenge dir:, like ~/ansible-challenge/hosts. Then you can reference this file by adding to any ansible command -i <path\_to\_inventory>/hosts

It’s allowed to be YAML-based inventory, so as INI-one.

<https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html#organizing-host-and-group-variables>

<https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html#intro-inventory>

Great reference is on the docs where

ansible\_host is very useful parameter for storing in inventory.

You may find that for some of your hosts are non-standard, with different users or systems.

That could be also managed with ansible.

* ansible\_user for ubuntu-based host would required.
* Ansible communicates with Windows with ntlm using winrm module <https://docs.ansible.com/ansible/latest/collections/ansible/builtin/winrm_connection.html>

### Ping from ansible

Once you have any host with it’s own name in your inventory file you may check basic communication between ansible controller and managed host

[centos@ip-10-97-10-167 ansible-challenge]$ ansible tomcat -m ping -i hosts

tomcat | SUCCESS => {

"ansible\_facts": {

"discovered\_interpreter\_python": "/usr/bin/python3"

},

"changed": false,

"ping": "pong"

}

here I mentioned tomcat because my inventory file contains short name for tomcat host:

...

tomcat ansible\_host=...

...

For pinging windows ansible is using other ansible module: win\_ping

<https://docs.ansible.com/ansible/2.5/modules/win_ping_module.html?highlight=win_ping>

<https://docs.ansible.com/ansible/2.5/user_guide/windows_winrm.html?highlight=winrm#ntlm>

[centos@ip-10-97-10-167 ansible-challenge]$ ansible windows -m win\_ping -i hosts

Additional settings for win\_ping required:

connection type,

connection type transport,

and port required for windows managing

port is default for ntlm.

Next step here is to install AppDynamics ansible galaxy collection:

<https://github.com/Appdynamics/appdynamics-ansible>

ansible-galaxy collection install appdynamics.agents

## AppD Controller

Login to enterprise console admin@appd

Login to controller appd@appd

Make sure that your controller is working and license is up to date.

Apply license (generate your own if outdated).

Grab the controller communication variables.

According to ansible .NET agent role example <https://github.com/Appdynamics/appdynamics-ansible>, you would need next controller parameters:

# Your controller details

controller\_account\_access\_key: "123456" # Please add this to your Vault

controller\_global\_analytics\_account\_name: "customer1\_GUID" # Please add this to your Vault

controller\_host\_name: "XXXXXXXXX"

controller\_account\_name: "customer1" # Please add this to your Vault

enable\_ssl: "true"

controller\_port: "443"

enable\_proxy: "true" #use quotes please

proxy\_host: "10.0.1.3"

proxy\_port: "80"

These variables might become common for all your agents, and it might be helpful to create a variable file and include it later, like it mentioned in other example:

---

- hosts: all

tasks:

- name: Include variables for the controller settings

include\_vars: vars/controller.yaml

So now you’re able to create variables yaml file for future use in playbooks

### TIP

take a look on ansible roles in github and search for example.

## Machine-agents installation

Example from galaxy-collection is pretty straight-forward

<https://github.com/Appdynamics/appdynamics-ansible#machine-agent>

Don’t be shy with copy-paste it to your own ansible playbook yaml file, include your controller variables configuration and run it.

ansible-playbook machine-lin.yaml -i hosts

After couple of minutes you would be able to get performance metrics on the Servers tab of controller.

### TIP

If you want to get verbose output on what’s happening behind the scenes while executing playbook, just add -v, -vv, or -vvv

ansible-playbook machine-lin.yaml -i hosts -vv

## Database agent installation

Simple role, [https://github.com/Appdynamics/appdynamics-ansible#db-agent](https://github.com/Appdynamics/appdynamics-ansible" \l "db-agent) allows you to install DB agent.

I recommend you to install it on AppDynamics controller itself and make sure that it connects to AppD controller.

You might be able to check it on Admin page with DB agents tab.

## Application component: Tomcat

Based on role from galaxy collection

<https://github.com/Appdynamics/appdynamics-ansible#instrument-apache-tomcat>

Don’t forget apply variables from the var file

One word about setenv.sh – it’s a Catalina script which is started with initial java process. Your environment might not have it. Don’t worry, role would create it for you. Only you need to take really care – proper path to setenv.sh and a proper app user which is running tomcat

# ... it’s just a part of playbook, not a whole one!

- include\_role:

name: appdynamics.agents.instrument\_tomcat

vars:

# instrument tomcat:

tomcat\_service: tomcat

application\_name: "xxxx"

tier\_name: "xxxx"

node\_name: "xxxx"

app\_user: WHAT\_USER

restart\_app: yes

tomcat\_config: /PATH/TO/CATALINA/setenv.sh

After successful agent registration you might generate traffic on tomcat application page:

http://<tomcat-host>:8080/examples/

http://<tomcat-host>:8080/examples/websocket/index.xhtml

Text

Description automatically generated

As an additional playground, you’re able to log in management console of tomcat:

Management console http://<tomcat-host>:8080/manager

admin@AppDynamics!257

or tomcat@AppDynamics!257

### TIP

If AppD agents won’t change, they wouldn’t restart the tomcat according to ansible handlers definition. <https://docs.ansible.com/ansible/latest/user_guide/playbooks_handlers.html>

You’re able to do it on your ownin couple ways:

1. Create additional task with restarting tomcat service. In that case each time playbook would run, it would restart tomcat. Probably it’s not so good for prod, uh?
2. Manually trigger restarting tomcat through ansible CLI with system module: ansible tomcat -i hosts -m systemd -a "name=tomcat state=restarted" -b. You need just adjust tomcat to name of your tomcat server name from inventory file.
3. Change something in agent settings, to trigger restarting handler.

You also might need the task that allows application with agent write logs. By default java agent installed under appdynamics user and group. So you need to add tomcat user (which is running tomcat) to appdynamics group:

<https://docs.ansible.com/ansible/latest/collections/ansible/builtin/user_module.html>

## Application component: JBOSS

Based on role from galaxy collection

<https://github.com/Appdynamics/appdynamics-ansible#instrument-jbosswildfly>

Applying variables from the var file is a good approach instead of applying them from the ansible playbook

Instrumenting JBOSS with appdynamics roles don’t forget about user under which JBOSS is running. You’re able to figure it out right on the host.

# ... it’s just a part of playbook, not a whole one!

- include\_role:

name: appdynamics.agents.instrument\_jboss

vars:

# instrument jboss:

application\_name: "xxx"

tier\_name: "xxxx"

jboss\_service: wildfly

app\_user: WHICH\_USER\_RUNNING\_JBOSS

restart\_app: yes

jboss\_config: /PATH/TO/START/JBOSS/standalone.sh

Once you’ve done an instrumentation. You might apply a load to application:

JBOSS application is available on http://<jboss-host>:8080/ping/

Graphical user interface, text, application, email

Description automatically generated

Management console is also available on JBOSS for user

admin@Admin!2

## Application component: IIS application pools

Based on role from galaxy collection

<https://github.com/Appdynamics/appdynamics-ansible#net-agent>

Working as expected except one small point. IIS running application pools with .net version 4.0, so consider it while setting up playbook.

runtime\_reinstrumentation: "true" # Runtime reinstrumentation works for .NET Framework 4.5.2 and greater.

Also, you may consider including restart\_app field to automatically pick up application pool processes with .net agent.

By default, restarting IIS pools is disabled because of maintenance issues. No one from IIS admins would thank you for auto restarting their apps.

Perform that action wisely.

To perform load on application pool, there is a powershell script stored on the host.

C:\Lab\GenerateLoad.ps1

You may decide on your own way – run it from ansible, or by the hand.

## Application component: BDR

Java agent role [https://github.com/Appdynamics/appdynamics-ansible#java-agent](https://github.com/Appdynamics/appdynamics-ansible" \l "java-agent)

designed for delivering java agent to the host and configure default controller.xml configuration file with controller communication and application – tier – node naming.

Don’t forget that main approach is to deliver every change you need to do on the host with ansible.

In that challenge part you required to design additional tasks in your playbook that would help application start with proper settings.

### Application

Application is BDR onboarding lab with three components running and communication each other.

Three of them are java based. You’re able to manage application using /home/centos/BDR/start\_BDR.sh

/home/centos/BDR/stop\_BDR.sh

This application runs the load by itself, no loadgenerator required.

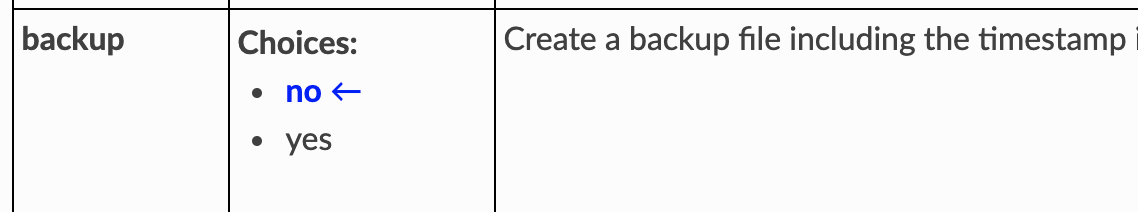
### Install agent with ansible

You might consider that additional setup for starting app required.

To achieve it, look on lineinfile ansible module.

<https://docs.ansible.com/ansible/2.5/modules/lineinfile_module.html?highlight=lineinfile>

It would allow you to rewrite any file with conditions. For that module quite important to use backup option, to not ruin startup script:



by default, it’s not enabled.

### TIP

Ensure to add the -javaagent argument before the -jar argument.

You also might need the task that allows application with agent write logs. By default java agent installed under appdynamics user and group. So you need to add centos user (which is running BDR app) to appdynamics group:

<https://docs.ansible.com/ansible/latest/collections/ansible/builtin/user_module.html>

You might dig into concept of ansible playbooks running with handlers. Notifying handler after task might trigger the handler itself.

<https://docs.ansible.com/ansible/latest/user_guide/playbooks_handlers.html>

## Application component: .net core on linux

Based on dotnetcore role from galaxy collection <https://github.com/Appdynamics/appdynamics-ansible/tree/master/roles/dotnetcore>

.net core role designed for delivering .netcore libraries to the host and configure default AppDynamics.json configuration file with blocks about controller communication, application – tier – node naming and full agent feature.

Don’t forget that main approach is to deliver every change you need to do on the host with ansible.

In that challenge part you required to design additional tasks in your playbook that would help application start with proper settings.

### Application

Application is RestaurantGuide API layer, which contains of two components:

Gateway,

OrderFullfillment.

Both are .net for linux based. You’re able to manage components using /home/centos/startAll.sh

/home/centos/killAll.sh

Application logs stored on each folder separately:

/home/centos/Gateway/applogs.txt

/home/centos/Orderfulfilment/applogs.txt

### .NET Core agent installation

<https://docs.appdynamics.com/21.6/en/application-monitoring/install-app-server-agents/net-agent/net-agent-for-linux/install-the-net-agent-for-linux>

To install and run agent properly you would need to specify environment variables with at least 3 variables CORECLR\_\*.

Also you might rewrite configuration file properties with env variables.

Talking a bit more about env variables, they might be applied to the process when it starts.

### Generating load

As far as application is API based, we might load it internally with swagger, or with any POST sender you have.

https://<netcore-host>:6001/swagger

It’s starting up with self-signed certificate, so cert error might occure.

Try out at first create couple Guests.

Graphical user interface, text, application

Description automatically generated

To combine and communicate with both application components you might decide to trigger registration visit with restaurantid: 1 (that one exists in DB)

Graphical user interface, text, application

Description automatically generated

### TIP

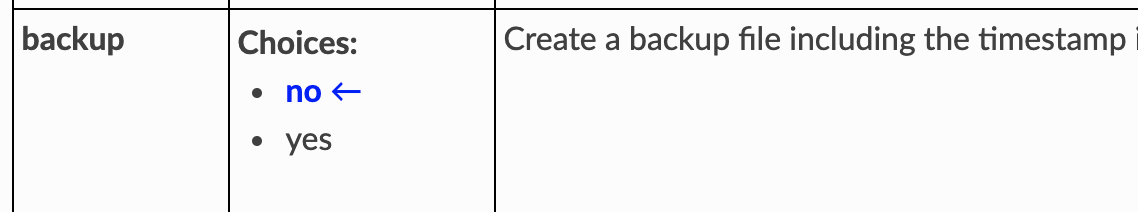
Take a look on starting app script with attention.

Piece of advice here is to start .net applications with env variables. That could require modifying start up scripts.

I used blockinfile ansible module to achieve it. Pretty sure, you are smart enough to repeat it, or even more, design your own way.

<https://docs.ansible.com/ansible/latest/collections/ansible/builtin/blockinfile_module.html>

For that module quite important to use backup option, to not ruin startup script:



by default, it’s not enabled.

You might dig into concept of ansible playbooks running with handlers. Notifying handler after task might trigger the handler itself.

<https://docs.ansible.com/ansible/latest/user_guide/playbooks_handlers.html>

If your agent didn’t appear on controller side while all logs looks fine, try latest version.

## Conclusion

Hope that you have picture similar to that on your controller:

Graphical user interface, application

Description automatically generated

## LEARN ANSIBLE

If you like the ansible and want to know more, there are two excellent courses on udemy.

For beginners

<https://cisco.udemy.com/course/learn-ansible/learn/lecture/7133354#overview>

Advanced ansible

[https://cisco.udemy.com/course/learn-ansible-advanced/learn/lecture/7687716](https://cisco.udemy.com/course/learn-ansible-advanced/learn/lecture/7687716%20%201.5)