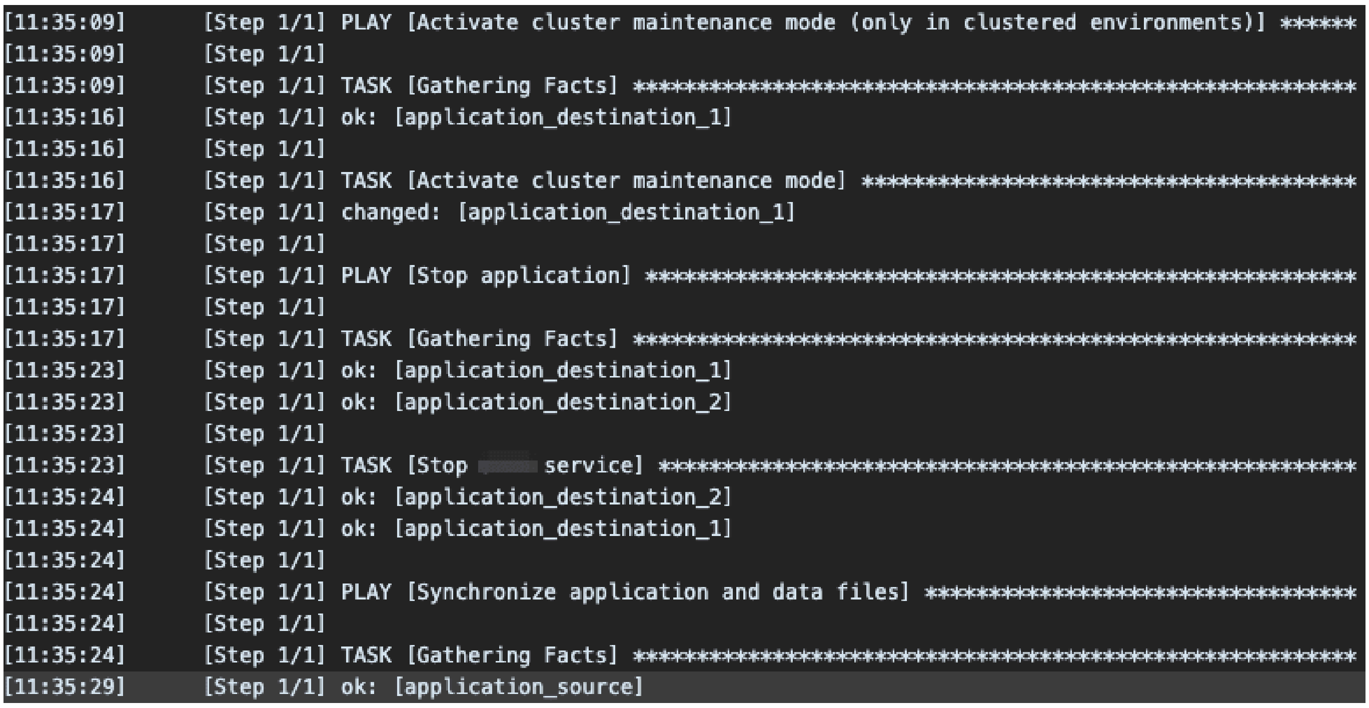
We're using some good amount of **Ansible** playbooks in my company and we use **TeamCity** as our main CI service.

On TeamCity, our playbook output looks like this:



```
[11:35:09]      [Step 1/1] PLAY [Activate cluster maintenance mode (only in clustered environments)] ******
[11:35:09]      [Step 1/1]
[11:35:09]      [Step 1/1] TASK [Gathering Facts] *********************************************************
[11:35:16]      [Step 1/1] ok: [application_destination_1]
[11:35:16]      [Step 1/1]
[11:35:16]      [Step 1/1] TASK [Activate cluster maintenance mode] ***************************************
[11:35:17]      [Step 1/1] changed: [application_destination_1]
[11:35:17]      [Step 1/1]
[11:35:17]      [Step 1/1] PLAY [Stop application] ********************************************************
[11:35:17]      [Step 1/1]
[11:35:17]      [Step 1/1] TASK [Gathering Facts] *********************************************************
[11:35:23]      [Step 1/1] ok: [application_destination_1]
[11:35:23]      [Step 1/1] ok: [application_destination_2]
[11:35:23]      [Step 1/1]
[11:35:23]      [Step 1/1] TASK [Stop [redacted] service] *******************************************************
[11:35:24]      [Step 1/1] ok: [application_destination_2]
[11:35:24]      [Step 1/1] ok: [application_destination_1]
[11:35:24]      [Step 1/1]
[11:35:24]      [Step 1/1] PLAY [Synchronize application and data files] **********************************
[11:35:24]      [Step 1/1]
[11:35:24]      [Step 1/1] TASK [Gathering Facts] *********************************************************
[11:35:29]      [Step 1/1] ok: [application_source]
```

□□□

I don't know about you, but I quite dislike the Ansible playbook default output. It's quite packed and big and it's hard to check where Ansible is currently working on.

TeamCity has a very nice feature called **Build Script Interaction**, where you format your stdout according to specific standards so that it can be interpreted by TeamCity.

This way you can open and close blocks, set variables, mark errors, etc.

Did I say „open and close blocks"? Yes I did and this would make the perfect feature for Ansible playbooks, right?

Luckily, Ansible can be extended by plugins and the Ansible output is handled by so called **callback plugins**. Callback plugins react to certain events (check out the class definition of **CallbackBase** for a list of available methods to override) and some of those events regard the start of plays or tasks.

So I extended the original default output plugin and override the specific methods for plays and tasks.

And with this plugin in place, the logs now look like this:

□□□

All nicely structured and TeamCity can even calculate the times of plays and tasks!

Check it out on GitHub:

dodevops / **ansible-teamcity-callback**

An Ansible callback plugin to output suitable for TeamCity

Ansible Callback Plugin for Teamcity

Introduction

This **Ansible** **callback plugin** formats the output of an Ansible playbook, so that it can be better interpreted by TeamCity.

Why?

Because Ansible logs in Teamcity look like this:

With this plugin in place, they look like this:

All plays and tasks are nicely put into their own blocks allowing TeamCity to collapse them and calculate the different times.

Usage

You have different options of using this plugin:

- In your playbook repository
  - Create a folder named `callback_plugins` directly where your playbook lives
  - Download the **file teamcity.py** and place it in inside the `callback_plugins` directory
- Somewhere else
  - **Download** or **clone** the plugin, unzip it and place the folder somewhere accessible (i.e. use the **Teamcity tools feature** to distribute the plugin to all agents)
  - Use the following environment variable to tell Ansible where to find your plugin: `export ANSIBLE_CALLBACK_PLUGINS=`

Set the following environment…

View on GitHub

Originally published to ***dev.to***