z/OS V2.5 IBM Education Assistant

Solution Name: Ansible drives z/OSMF to do data set operation, USS file operation and console operation

Solution Element(s): z/OSMF Ansible

July 2021

Agenda

- Trademarks
- Objectives
- Overview
- Usage & Invocation
- Interactions & Dependencies
- Upgrade & Coexistence Considerations
- Installation & Configuration
- Summary
- Appendix

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.
- Additional Trademarks:
  - None

Objectives

- As a system operator, I wish to manipulate z/OS data sets via Ansible module, so that I can write and use Ansible playbooks to do operation works which needs read/write z/OS data sets.
- As a system operator, I wish to manipulate USS files via Ansible module, so that I can write and use Ansible playbooks to do operation works which needs read/write USS files.
- As a system administrator, I wish to issue console command via Ansible module, so that I can write and use Ansible playbooks to interact with z/OS consoles.

Overview

- Who (Audience)
  - System Admin, DevOps Operator
- What (Solution)
  - z/OSMF Ansible Collection supports to retrieve and save the content of a sequential data set or a member of a partitioned data set(PDS or PDSE) from the remote z/OS system and copy data from Ansible control node to a sequential data set or a member of partitioned data set(PDS or PDSE) on the remote z/OS system.
  - z/OSMF Ansible Collection supports to retrieve and save the content of a USS file from the remote z/OS system and copy data from Ansible control node to a USS file on the remote z/OS system.
  - z/OSMF Ansible Collection supports to issue a MVS command using a EMCS console and retrieve/save the response to Ansible control node.
- Wow (Benefit / Value, Need Addressed)
  - Ansible is the most important automation framework in hybrid-cloud environment, with above features/modules supplied by z/OSMF Ansible Collection, users can complete most of their daily z/OS maintain and operation work via Ansible framework. This gives users more flexibility to orchestrate their works and supplies unified experience of operation works between z/OS and open-platform systems.

Usage & Invocation – module: zmf_dataset_fetch

- Module zmf_dataset_fetch
  - Retrieve the contents of a sequential data set, or a member of a partitioned data set (PDS or PDSE) from the remote z/OS system.
  - Save the retrieved data set or member on Ansible control node.
  - Data set or member that already exists at *dataset_dest* will be overwritten if it is different than the *dataset_src*.

- Main parameters
  - dataset_src: Data set or the name of the PDS or PDSE member on the remote z/OS system to fetch.
  - dataset_dest: The local directory on control node where the data set or member should be saved to.
  - dataset_volser: The volume serial to identify the volume to be searched for an uncataloged data set or member.

Usage & Invocation – module: zmf_dataset_fetch

- Playbook example

```yaml
- name: sample of fetching data set from z/OS
  hosts: dataset
  gather_facts: no
  collections:
    - ibm.ibm_zos_zosmf
  vars_prompt:
    - name: zmf_user
      prompt: "Enter your zOSMF username (skip if zmf_crt and zmf_key are supplied)"
      private: no
    - name: zmf_password
      prompt: "Enter your zOSMF password (skip if zmf_crt and zmf_key are supplied)"
      private: yes
  tasks:
    - zmf_authenticate:
        zmf_host: "{{ zmf_host }}"
        zmf_port: "{{ zmf_port }}"
        zmf_user: "{{ zmf_user }}"
        zmf_password: "{{ zmf_password }}"
      register: result_auth
      delegate_to: localhost
    - zmf_dataset_fetch:
        zmf_credential: "{{ result_auth }}" # Authentication credentials returned by module zmf_authenticate
        dataset_src: "ZOSMF.ANSIBLE.LIB(MEMBER01)"
        dataset_dest: "/tmp/dataset_output"
        # dataset_volser: "VOL001" # The volume to be searched for an uncataloged data set or member
        # dataset_flat: false # Whether to override the default behavior of appending zmf_host to the destination. Default is false
        # dataset_data_type: "text" # Whether data conversion is to be performed on the returned data. Default is text (data conversion is performed)
        # dataset_encoding: # Which encodings the fetched data set should be converted from and to
        #   from: IBM-1047
        #   to: ISO8859-1
        # dataset_range: # A range that is used to retrieve records of the data set
        #   start: 0
        #   end: 499
        # dataset_search: # A series of parameters that are used to search the content of data set or member
        #   keyword: "Health Checker"
        #   insensitive: true
        #   maxreturnsize: 100
        # dataset_migrate_recall: "wait" # How a migrated data set is handled. Default is wait
        # dataset_checksum: "93822124D6E66E2213C64B0D10800224" # The checksum to be used to verify that the data set to be fetched is not changed since
      register: result
      delegate_to: localhost
    - debug: var=result
```

Usage & Invocation – module: zmf_dataset_copy

- Module zmf_dataset_copy
  - Copy data from Ansible control node to a sequential data set, or a member of a partitioned data set (PDS or PDSE) on the remote z/OS system.
  - If the target data set or member already exists, it can be overwritten. If the target PDS or PDSE member does not exist, it can be created.
  - If the target data set does not exist, it can be created based on *dataset_model* or the size of the source.

- Main parameters
  - dataset_src: The local path on control node of the data to be copied to the target data set or member.
  - dataset_dest: Data set or the name of the PDS or PDSE member on the remote z/OS system where the data should be copied to.
  - dataset_volser: The volume serial to identify the volume to be searched for an uncataloged data set or member.
  - dataset_content: The contents to be copied to the target data set or member. This variable is used instead of *dataset_src*.
  - dataset_model: When copying a local file to a non-existing PDS, PDSE or PS, specify a model data set to allocate the target data set.

Usage & Invocation – module: zmf_dataset_copy

- Playbook example

```yaml
- name: sample of copying data to a z/OS data set or member
  hosts: dataset
  gather_facts: no
  collections:
    - ibm.ibm_zos_zosmf
  vars_prompt:
    - name: zmf_user
      prompt: "Enter your zOSMF username (skip if zmf_crt and zmf_key are supplied)"
      private: no
    - name: zmf_password
      prompt: "Enter your zOSMF password (skip if zmf_crt and zmf_key are supplied)"
      private: yes
  tasks:
    - zmf_authenticate:
        zmf_host: "{{ zmf_host }}"
        zmf_port: "{{ zmf_port }}"
        zmf_user: "{{ zmf_user }}"
        zmf_password: "{{ zmf_password }}"
      register: result_auth
      delegate_to: localhost
    - zmf_dataset_copy:
        zmf_credential: "{{ result_auth }}" # Authentication credentials returned by module zmf_authenticate
        dataset_src: "/tmp/dataset_input/member01"
        # dataset_content: "Sample profile\nTZ=EST5EDT\n"
        dataset_dest: "ZOSMF.ANSIBLE.LIB(MEMBER01)"
        # dataset_volser: "VOL001" # The volume to be searched for an uncataloged data set or member
        # dataset_force: true # Whether the target data set must always be overwritten. Default is true
        # dataset_data_type: "text" # Whether data conversion is to be performed on the data to be copied. Default is text
        # dataset_encoding: # Which encodings the data to be copied should be converted from and to
        #   from: ISO8859-1
        #   to: IBM-1047
        # dataset_crlf: false # Whether each input text line is terminated with a carriage return line feed (CRLF) or a lin
        # dataset_diff: false # Whether the input consists of commands in the same format as produced by the z/OS UNIX 'dif
        # dataset_migrate_recall: "wait" # How a migrated data set is handled. Default is wait
        # dataset_checksum: "93822124D6E66E2213C64B0D10800224" # The checksum to be used to verify that the target data set
      register: result
      delegate_to: localhost
    - debug: var=result
```

Usage & Invocation – module: zmf_file_fetch

- Module zmf_file_fetch
  - Retrieve the contents of a z/OS UNIX System Services (USS) file from the remote z/OS system, and save them on Ansible control node.
  - USS file that already exists at *file_dest* will be overwritten if it is different than the *file_src*.

- Main parameters
  - file_dest: The local directory on control node where the USS file should be saved to.
  - file_src: USS file on the remote z/OS system to fetch. This variable must consist of a fully qualified path and file name.

Usage & Invocation – module: zmf_file_fetch

- Playbook example

```yaml
- name: sample of fetching USS file from z/OS
  hosts: file
  gather_facts: no
  collections:
    - ibm.ibm_zos_zosmf
  vars_prompt:
    - name: zmf_user
      prompt: "Enter your zOSMF username (skip if zmf_crt and zmf_key are supplied)"
      private: no
    - name: zmf_password
      prompt: "Enter your zOSMF password (skip if zmf_crt and zmf_key are supplied)"
      private: yes
  tasks:
    - zmf_authenticate:
        zmf_host: "{{ zmf_host }}"
        zmf_port: "{{ zmf_port }}"
        zmf_user: "{{ zmf_user }}"
        zmf_password: "{{ zmf_password }}"
      register: result_auth
      delegate_to: localhost
    - zmf_file_fetch:
        zmf_credential: "{{ result_auth }}" # Authentication credentials returned by module zmf_authenticate
        file_src: "/etc/profile"
        file_dest: "/tmp/file_output"
        # file_flat: false # Whether to override the default behavior of appending hostname/path/to/file to the destination.
        # file_data_type: "text" # Whether data conversion is to be performed on the returned data. Default is text (data co
        # file_encoding: # Which encodings the fetched USS file should be converted from and to
        #   from: IBM-1047
        #   to: ISO8859-1
        # file_range: # A range that is used to retrieve the USS file
        #   start: 0
        #   end: 499
        # file_search: # A series of parameters that are used to search the USS file
        #   keyword: "Health Checker"
        #   insensitive: true
        #   maxreturnsize: 100
        # file_checksum: "93822124D6E66E2213C64B0D10800224" # The checksum to be used to verify that the USS file to be fetc
      register: result
      delegate_to: localhost
    - debug: var=result
```

Usage & Invocation – module: zmf_file_copy

- Module zmf_file_copy
  - Copy data from Ansible control node to a z/OS UNIX System Services (USS) file on the remote z/OS system.
  - If the target USS file already exists, it can be overwritten. If the target USS file does not exist, it can be created with mode 644.

- Main parameters
  - file_dest: USS file on the remote z/OS system where the data should be copied to.
  - file_src: The local path on control node of the data to be copied to the target USS file.
  - file_content: The contents to be copied to the target USS file. This variable is used instead of *file_src*.

Usage & Invocation – module: zmf_file_copy

- Playbook example

```yaml
- name: sample of copying data to a z/OS USS file
  hosts: file
  gather_facts: no
  collections:
    - ibm.ibm_zos_zosmf
  vars_prompt:
    - name: zmf_user
      prompt: "Enter your zOSMF username (skip if zmf_crt and zmf_key are supplied)"
      private: no
    - name: zmf_password
      prompt: "Enter your zOSMF password (skip if zmf_crt and zmf_key are supplied)"
      private: yes
  tasks:
    - zmf_authenticate:
        zmf_host: "{{ zmf_host }}"
        zmf_port: "{{ zmf_port }}"
        zmf_user: "{{ zmf_user }}"
        zmf_password: "{{ zmf_password }}"
      register: result_auth
      delegate_to: localhost
    - zmf_file_copy:
        zmf_credential: "{{ result_auth }}" # Authentication credentials returned by module zmf_authenticate
        file_src: "/tmp/file_input/profile"
        # file_content: "Sample profile\nTZ=EST5EDT\n"
        file_dest: "/etc/profile"
        # file_force: true # Whether the target USS file must always be overwritten. Default is true
        # file_data_type: "text" # Whether data conversion is to be performed on the data to be copied. Default is text
        # file_encoding: # Which encodings the data to be copied should be converted from and to
        #   from: ISO8859-1
        #   to: IBM-1047
        # file_crlf: false # Whether each input text line is terminated with a carriage return line feed (CRLF) or a lin
        # file_diff: false # Whether the input consists of commands in the same format as produced by the z/OS UNIX 'dif
        # file_checksum: "93822124D6E66E2213C64B0D10800224" # The checksum to be used to verify that the target USS file
      register: result
      delegate_to: localhost
    - debug: var=result
```

Usage & Invocation – module: zmf_console_command

- Module zmf_console_command
  - Issue MVS command by using a system console through z/OS console RESTful services.
  - Retrieve command response and define success condition based on specified keywords in the command response or broadcast messages.
  - Save the command response on Ansible control node.

- Main parameters
  - console_cmd: Specifies the command to issue.
  - console_cmdresponse_keyword: Specifies a keyword that you want to detect in the command response. Case is not significant.
  - Console_name: Name of the EMCS console that is used to issue the command.
  - Console_system: Nickname of the target z/OS system in the same sysplex that the command is routed to

Usage & Invocation – module: zmf_console_command

- Playbook example

```yaml
- name: sample of issuing MVS command by using a system console
  hosts: console
  gather_facts: no
  collections:
    - ibm.ibm_zos_zosmf
  vars_prompt:
   - name: zmf_user
      prompt: "Enter your zOSMF username (skip if zmf_crt and zmf_key are supplied)"
      private: no
    - name: zmf_password
      prompt: "Enter your zOSMF password (skip if zmf_crt and zmf_key are supplied)"
      private: yes
  tasks:
    - zmf_authenticate:
        zmf_host: "{{ zmf_host }}"
        zmf_port: "{{ zmf_port }}"
        zmf_user: "{{ zmf_user }}"
        zmf_password: "{{ zmf_password }}"
      register: result_auth
      delegate_to: localhost
    - zmf_console_command:
        zmf_credential: "{{ result_auth }}" # Authentication credentials returned by module zmf_authenticate
        console_cmd: "start pegasus"
        console_system: "{{ inventory_hostname }}"
        # console_cmdresponse_keyword: "SLP registration initiated" # The keyword that you want to detect in the command response. The mo
        # console_cmdresponse_reg: false # Whether console_cmdresponse_keyword represents a regular expression. Default is false
        # console_broadcastmsg_keyword: "started CIM server" # The keyword that you want to detect in broadcast messages. The module will
        # console_broadcastmsg_reg: false # Whether console_broadcastmsg_keyword represents a regular expression. Default is false
        # console_broadcastmsg_detect_timeout: 30 # How long, in seconds, the console attempts to detect the value of console_broadcastms
        # console_cmdresponse_retrieve_times: 1 # How many times the console attempts to retrieve the command response. Default is 1
        console_save_output_localpath: "/tmp/cmd_output" # The local path on control node where the command response will be saved to
      register: result
      delegate_to: localhost
    - debug: var=result
```

Interactions & Dependencies

- Software Dependencies
  - z/OSMF
    - z/OS data set and files REST interface
    - z/OS console services
- Hardware Dependencies
  - None
- Exploiters
  - None

Upgrade & Coexistence Considerations

- To exploit this solution, all systems in the Plex must be at the new z/OS level:
  - No
- List any toleration/coexistence APARs/PTFs.
  - N/A
- List anything that doesn't work the same anymore.
  - N/A
- Upgrade involves only those actions required to make the new system behave as the old one did.
  - N/A
- Coexistence applies to lower level systems which coexist (share resources) with latest z/OS systems.
  - N/A

Installation & Configuration

- z/OSMF Ansible collection can be installed from Ansible Galaxy: https://galaxy.ansible.com/ibm/ibm_zos_zos

Summary

- The following z/OS V2R5 item has been explained:

  Ansible drives z/OSMF to do data set operation, USS file operation and console operation

Appendix

- Website:https://ibm.github.io/ibm_zos_zosmf/index.html