at computing
The one-stop-Linux-shop

Men&Mice Suite Ansible Integration

Ton Kersten

Table of Contents

Chapter 1. Ansible setup for Men&Mice Suite

With the Ansible setup for the Men&Mice suite you can manage a Men&Mice installation through Ansible. The Ansible modules and plugins connect to the Men&Mice Suite API and perform all needed actions.

The modules and plugins need to be installed on the Ansible control node, often called the Ansible Master and Ansible needs to be configured so that the modules and plugins can be found by Ansible.

1.1. Installation

Installing the Ansible modules and plugins is a straight forward process. Copy the Ansible modules and plugins to a directory on the Ansible control node, let us assume `/tmp/mandm`.

1.1.1. Ansible modules

The Ansible modules can than be placed in a number of directories, depending on your installation and requirements.

1. `/usr/share/ansible/plugins/modules/` System wide installation, modules available to all users
2. `~/.ansible/plugins/modules/` Modules available only to the current user, as the modules are installed in the users home-directory
3. `/etc/ansible/library/` Local installation. As most Ansible installations use the `/etc/ansible` directory as the Ansible top-directory (as this is the default in an Ansible installation), this is probably the best installation option. When installing the modules in this directory, the Ansible `library` path needs to be set in the `/etc/ansible/ansible.cfg` file, pointing to the module directory.

```
library = /etc/ansible/library
```

After installing the Ansible modules a check can be run to determine if the modules are installed correctly, by running the command:

```
ansible-doc -l | grep '^mm_'
```

This should produce a list with all the Men&Mice Suite Ansible modules.

1.1.2. Ansible lookup plugins

The set of Ansible modules consists of multiple sets (`lookup` and `inventory`) and these

should be installed in their own directories

The `lookup` plugins can be installed in:

1. `/usr/share/ansible/plugins/lookup` System wide installation, modules available to all users
2. `~/.ansible/plugins/lookup` Plugins available only to the current user, as the plugins are installed in the users home-directory
3. `/etc/ansible/plugins/lookup` Local installation. As most Ansible installations use the `/etc/ansible` directory as the Ansible top-directory (as this is the default in an Ansible installation)this is probably the best installation option. When installing the lookup plugins in this directory, the Ansible `lookup` path needs to be set in the `/etc/ansible/ansible.cfg` file, pointing to the lookup plugin directory.

```
lookup_plugins =
/usr/share/ansible/plugins/lookup:/etc/ansible/plugins/lookup
```

To check if the modules are installed correctly and are available to Ansible, issue the command:

```
ansible-doc -t lookup -l
```

Which should produce a list with all the Men&Mice Suite Ansible lookup plugins.

1.1.3. Ansible inventory plugins

The set of Ansible modules consists of multiple sets (`lookup` and `inventory`) and these should be installed in their own directories

The `inventory` plugins can be installed in:

1. `/usr/share/ansible/plugins/inventory` System wide installation, modules available to all users
2. `~/.ansible/plugins/inventory` Plugins available only to the current user, as the plugins are installed in the users home-directory
3. `/etc/ansible/plugins/inventory` Local installation. As most Ansible installations use the `/etc/ansible` directory as the Ansible top-directory (as this is the default in an Ansible installation) this is probably the best installation option. When installing the inventory plugins in this directory, the Ansible `lookup` path needs to be set in the `/etc/ansible/ansible.cfg` file, pointing to the lookup plugin directory.

```
inventory_plugins =
/usr/share/ansible/plugins/inventory:/etc/ansible/plugins/inventory
```


To check if the modules are installed correctly and are available to Ansible, issue the command:

```
ansible-doc -t inventory -l
```

Which should produce a list with all the Men&Mice Suite Ansible inventory plugins.

The `mm_inventory` plugin also needs some extra configuration, read the `README_inventory.md` for more information.

1.2. API user

As the Ansible modules and plugins connect to a Men&Mice Suite installation, a connection between Ansible and the Men&Mice Suite needs to be made.

1.2.1. API user for Men&Mice Suite

In the Men&Mice Suite a user needs to be defined that has all rights in the Men&Mice Suite (`administrator`) so it is able to perform all needed tasks.

1.2.2. API Provider in Ansible

For the Ansible modules and plugins to function correctly a *provider* has to be defined. This provider consist of a `user`, `password` and connection url (`mmurl`) and this provider needs to be defined in the Ansible setup, either through Ansible Tower/AWX or in the Ansible directory.

As the modules and plugins can be used by all systems under Ansible control, it is advised to define the API provider for the `all` group. Create a file `all` in the `/etc/ansible/group_vars` directory, or the `/etc/ansible/inventory/group_vars` directory (if your inventory is a directory instead of a file) which contains something similar to:

```
---
provider:
  mmurl: http://mmsuite.example.net
  user: apiuser
  password: apipasswd
```

Where the `apipasswd` should be encrypted with `ansible-vault` to prevent plain passwords in the Ansible tree. An example to achieve this is


```
printf "apipasswd"              | \
    ansible-vault               \
        encrypt_string          \
        --stdin-name="password"
```

Which results in:

```
password: !vault |
  $ANSIBLE_VAULT;1.1;AES256

34646538383265336162666535313938333861323465333661376436656637643434316266666430

6139656636383537336365313165663861383165616632330a3332303566623361613934396666431

353931306565653131383835656261373536393162393937353066633763626138616231356656634

6332393063643531390a343661373263656132363737326666396132373461323631613034356565
  6138
```

The defined provider can be used in Ansible playbooks like:

```
- name: Claim IP address
  mm_claimip:
    state: present
    ipaddress: 172.16.12.14
    provider: "{{ provider }}"
  delegate_to: localhost
```

The reason for the `delegate_to: localhost` option, is that all commands can be performed on the Ansible control node. So, it is possible to protect the Men&Mice Suite API to only accept commands from the Ansible control node and not from everywhere. This can also be achieved by creating a playbook that has a connection with `localhost` and is specific for the interaction with the Men&Mice Suite.


```
---
- name: host connection example
  hosts: localhost
  connection: local
  become: false

  tasks:
    - name: Claim IP address
      mm_claimip:
        state: present
        ipaddress: 172.16.12.14
        provider: "{{ provider }}"
```

1.3. Ansible configuration example

Beneath the is an example Ansible configuration file (`ansible.cfg`) with the assumption that all Men&Mice plugins and modules are installed in the `/etc/ansible` directory.

```
# ==================================================
[defaults]
remote_tmp              = $HOME/.ansible/tmp
inventory               = inventory
pattern                 = *
forks                   = 5
poll_interval           = 15
ask_pass                = False
remote_port             = 22
remote_user             = ansible
gathering               = implicit
host_key_checking       = False
interpreter_python      = auto_silent
force_valid_group_names = true
retry_files_enabled     = False
library                 = /etc/ansible/library
action_plugins          = /usr/share/ansible_plugins/action_plugins
callback_plugins        = /etc/ansible/plugins/callback_plugins
connection_plugins      = /usr/share/ansible_plugins/connection_plugins
filter_plugins          = /usr/share/ansible_plugins/filter_plugins
inventory_plugins       =
/usr/share/ansible_plugins/inventory_plugins:/etc/ansible/plugins/invent
ory
lookup_plugins          =
/usr/share/ansible_plugins/lookup_plugins:/etc/ansible/plugins/lookup
vars_plugins            = /usr/share/ansible_plugins/vars_plugins
callback_whitelist      = minimal, dense, oneline
stdout_callback         = default

[inventory]
enable_plugins   = mm_inventory, host_list, auto
cache            = no
cache_plugin     = pickle
cache_prefix     = mm_inv
cache_timeout    = 60
cache_connection = /tmp/mm_inventory_cache

[privilege_escalation]
become          = False
become_method   = sudo
become_user     = root
become_ask_pass = False
```


Chapter 2. Ansible mm_freeip plugin

This Men&Mice FreeIP lookup plugin finds one or more free IP addresses in a certain network, defined in the Men&Mice suite.

2.1. Usage

When using the Men&Mice FreeIP plugin something needs to be taken into account. When running an Ansible lookup plugin, this lookup action takes place every time the variable is referenced. So it will not be possible to claim an IP address for further reference, this way. This has to do with the way Ansible works. A solution for this is to assign all collected IP addresses to an Ansible fact, but here you need to make sure the factname is not used over multiple hosts.

Example usage:

```
---
- name: Men&Mice FreeIP test play
  hosts: localhost
  connection: local
  become: false

  vars:
    provider:
      mmurl: http://mmsuite.example.net
      user: apiuser
      password: apipassword
    network: examplenet

  tasks:
    - name: Set free IP addresses as a fact
      set_fact:
        freeips: "{{ query('mm_freeip',
                           provider,
                           network,
                           multi=15,
                           claim=60,
                           startaddress='192.168.63.100',
                           excludedhcp=True,
                           ping=True)
                  }}"

    - name: Get the free IP address and show info
      debug:
        msg:
          - "Free IPs          : {{ freeips }}"
          - "Queried network   : {{ network }}"
          - "Ansible version   : {{ ansible_version.full }}"
          - "Python version    : {{ ansible_facts['python_version'] }}"
          - "Python executable : {{
ansible_facts['python']['executable'] }}"

    - name: Loop over IP addresses
      debug:
        msg:
          - "Next free IP      : {{ item }}"
      loop: "{{ freeips }}"
```

```
# ansible-playbook mmtest.yml

PLAY [Men&Mice FreeIP test play] ************************************

TASK [Gathering Facts] **********************************************
ok: [localhost]

TASK [Set free IP addresses as a fact] ******************************
ok: [localhost]

TASK [Get the free IP address and show info] ************************
ok: [localhost] => {
    "msg": [
        "Free IPs          : ['192.168.63.203', '192.168.63.204']",
        "Queried network   : nononet",
        "Ansible version   : 2.9.7",
        "Python version    : 3.6.8",
        "Python executable : /usr/libexec/platform-python"
    ]
}

TASK [Loop over IP addresses] ***************************************
ok: [localhost] => (item=192.168.63.203) => {
    "msg": [
        "Next free IP      : 192.168.63.203"
    ]
}
ok: [localhost] => (item=192.168.63.204) => {
    "msg": [
        "Next free IP      : 192.168.63.204"
    ]
}

PLAY RECAP **********************************************************
localhost : ok=4  changed=0  unreachable=0  failed=0  skipped=0
rescued=0  ignored=0
```


Chapter 3. Ansible mm_inventory plugin

This plugin generates the inventory from the Men&Mice Suite. It supports reading configuration from both a YAML configuration file and environment variables. If reading from the YAML file, the filename must end with `mm_inventory.(yml|yaml)`, the path in the command would be `/path/to/mm_inventory.(yml|yaml)`. If some arguments in the configuration file are missing, this plugin will try to fill in the missing arguments by reading from environment variables. If reading configurations from environment variables, the path in the command must be `@mm_inventory`.

Valid configuration filenames are:

- `mm_inventory`
- `mmsuite`
- `mandm`
- `menandmice`
- `mandmsuite`
- `mm_suite`
- `mandm_suite`

3.1. Options

There are two sets of configuration options, the options for the inventory plugin to function correctly and for Ansible to know how to use the plugin.

3.1.1. Plugin configuration

The `mm_inventory` plugin is configured through a configuration file, named `mm_inventory.yml` and the options are:

- plugin: Name of the plugin (`mm_inventory`)
- host: Men&Mice Suite to connect to (`http://mmsuite.example.net`)
- user: UserID to connect with (`apiuser`)
- password: The password to connect with (`apipasswd`)
- filters: Filter on custom properties, can be more than 1 and should be a list. If multiple filters are given, they act as an ``and'' function
- ranges: What IP ranges to examine (`172.16.17.0/24`) Multiple ranges can be given, they act as an ``or'' function

When both *ranges* and *filters* are supplied that will result in an ``and'' function.

Example:

```
filters:
  - location: home
  - owner: tonk
ranges:
  - 192.168.4.0/24
  - 172.16.17.0/24
```

Would result in an inventory for all host that have the `location: home` *and* `owner: tonk` custom properties set *and* are either a member of the `192.168.4.0/24` *or* `172.16.17.0/24` range.

An example of the `mm_inventory.yml` file:

```
plugin: mm_inventory
host: "http://mmsuite.example.net"
user: apiuser
password: apipasswd
filters:
  - location: London
ranges:
  - 172.16.17.0/24
```

Environment variables:

The `mm_inventory` plugin can also be configured through environment variables

```
export MM_HOST=YOUR_MM_HOST_ADDRESS
export MM_USER=YOUR_MM_USER
export MM_PASSWORD=YOUR_MM_PASSWORD
export MM_FILTERS=YOUR_MM_FILTERS
export MM_RANGES=YOUR_MM_RANGES
```

When reading configuration from the environment, the inventory path must always be `@mm_inventory`.

```
ansible-inventory -i @mm_inventory --list
```

3.1.2. Ansible configuration

Ansible needs to know about the `mm_inventory` plugin and also has some extra configuration options. First the `mm_inventory` plugin needs to be enabled, so Ansible can

use it. This is done in the `[inventory]` section in the `ansible.cfg` file.

```
[inventory]
enable_plugins   = mm_inventory, host_list, auto
cache            = yes
cache_plugin     = jsonfile
cache_prefix     = mm_inv
cache_timeout    = 3600
cache_connection = /tmp/mm_inventory_cache
```

With the following meaning:

- `cache`: Switch caching on and off
- `cache_plugin`: Which caching plugin to use
  - `jsonfile`
  - `yaml`
  - `pickle`
  - ...
- `cache_prefix`: User defined prefix to use when creating the cache files
- `cache_connection`: Path in which the cache plugin will save the cache files
- `cache_timeout`: Timeout for the cache

Now the inventory plugin can be used with Ansible, like:

```
ansible-inventory -i /path/to/mm_inventory.yml --list
```

Or set the `mm_inventory.yml` as the Ansible inventory in the `ansible.cfg` file.

```
inventory = mm_inventory.yml
```


Chapter 4. Ansible mm_ipinfo plugin

This Men&Mice IPInfo lookup plugin finds a lot of info about a specified IP address, defined in the Men&Mice suite.

4.1. Usage

The `mm_ipinfo` plugin delivers a complete set of information about an IP address, as it is delivered by the Men&Mice Suite API.

Example usage:

```
- name: Get all info for this IP address
  debug:
    var: ipinfo
  vars:
    ipinfo: "{{ query('mm_ipinfo', provider, '172.16.17.2') |
to_nice_json }}"
```

With output like (output shortened):

```
ok: [localhost] => {
    "ipinfo": {
        "addrRef": "IPAMRecords/11",
        "address": "172.16.17.2",
        "claimed": false,
        "customProperties": {
            "location": "At the attic"
        },
    }
}
```

4.2. Ansible modules

4.3. mm_claimip

Claim IP addresses in DHCP in the Men&Mice Suite

4.3.1. Options

- `customproperties`: Custom properties for the IP address. These properties must already exist. See also [mm_props]

- `ipaddress`: (required) The IP address(es) to work on.
- `provider`: (required) Definition of the Men&Mice suite API provider.
- `state`: The state of the claim. (`absent`, `present`)

4.3.2. Examples

```
- name: Claim IP address
  mm_claimip:
    state: present
    ipaddress: 172.16.12.14
    provider:
      mmurl: http://mmsuite.example.net
      user: apiuser
      password: apipasswd
  delegate_to: localhost

- name: Release claim on IP addresses
  mm_claimip:
    state: present
    ipaddress:
      - 172.16.12.14
      - 172.16.12.15
      - 172.16.12.16
    provider:
      mmurl: http://mmsuite.example.net
      user: apiuser
      password: apipasswd
  delegate_to: localhost
```

4.4. mm_dhcp

Manage DHCP reservations on the Men&Mice Suite

4.4.1. Options

- `ddnshost`: The dynamic DNS host to place the entry in.
- `deleteunspecified`: Clear properties that are not explicitly set.
- `filename`: Filename to place the entry in.
- `ipaddress`: (required) The IP address(es) to make a reservation on. When the IP address is changed a new reservation is made. It is not allowed to make reservations in DHCP blocks.
- `macaddress`: (required) MAC address for the IP address.
- `name`: (required) Name of the reservation

- `nextserver`: Next server as DHCP option (bootp).
- `provider`: (required) Definition of the Men&Mice suite API provider.
- `servername`: Server to place the entry in.
- `state`: The state of the reservation. (`absent`, `present`)

4.4.2. Examples

```
- name: Add a reservation for an IP address
  mm_dhcp:
    state: present
    name: myreservation
    ipaddress: 172.16.17.8
    macaddress: 44:55:66:77:88:99
    provider:
      mmurl: http://mmsuite.example.net
      user: apiuser
      password: apipasswd
  delegate_to: localhost
```

4.5. mm_dnsrecord

Manage DNS records in the Men&Mice Suite

4.5.1. Options

- `aging`: The aging timestamp of dynamic records in AD integrated zones. Hours since January 1, 1601, UTC. Providing a non-zero value creates a dynamic record.
- `comment`: Comment string for the record. Note that only records in static DNS zones can have a comment string
- `data`: (required) The record data in a tab-separated list.
- `dnszone`: (required) The DNS zone where the action should take place.
- `enabled`: True if the record is enabled. If the record is disabled the value is false
- `name`: (required) The name of the DNS record. Can either be partially or fully qualified.
- `provider`: (required) Definition of the Men&Mice suite API provider.
- `rrtype`: Resource Record Type for the IP address.
- `state`: The state of the properties. (`absent`, `present`)
- `ttl`: The Time-To-Live of the DNS record.

4.5.2. Examples

```
- name: Set DNS record in zone for a defined name
  mm_dnsrecord:
    state: present
    name: beatles
    data: 172.16.17.2
    rrtype: A
    dnszone: example.net.
    provider:
      mmurl: http://mmsuite.example.net
      user: apiuser
      password: apipasswd
  delegate_to: localhost

- name: Set PTR record in zone for a defined name
  mm_dnsrecord:
    state: present
    name: "2.17.16.172.in-addr.arpa."
    data: beatles.example.net.
    rrtype: PTR
    dnszone: "17.16.172.in-addr.arpa."
    provider:
      mmurl: http://mmsuite.example.net
      user: apiuser
      password: apipasswd
  delegate_to: localhost
```

4.6. mm_ipprops

Set properties on an IP address in the Men&Mice Suite

4.6.1. Options

- `deleteunspecified`: Clear properties that are not explicitly set.
- `ipaddress`: (required) The IP address(es) to work on.
- `properties`: (required) Custom properties for the IP address. These properties must already be defined.
- `provider`: (required) Definition of the Men&Mice suite API provider.
- `state`: Property present or not. (`absent`, `present`)

4.6.2. Examples

```
- name: Set properties on IP
  mm_ipprops:
    state: present
    ipaddress: 172.16.12.14
    properties:
      claimed: false
      location: London
    provider:
      mmurl: http://mmsuite.example.net
      user: apiuser
      password: apipasswd
  delegate_to: localhost
```

4.7. mm_props

Manage custom properties in the Men&Mice Suite

4.7.1. Options

- `cloudtags`: Associated cloud tags.
- `defaultvalue`: Default value of the property.
- `dest`: (required) The section where to define the custom property.
- `listitems`: The items in the selection list.
- `mandatory`: Is the property mandatory.
- `multiline`: Is the property multiline.
- `name`: (required) Name of the property.
- `proptype`: Type of the property. These are not the types as described in the API, but the types as you can see them in the Men&Mice Management Console.
- `provider`: (required) Definition of the Men&Mice suite API provider.
- `readonly`: Is the property read only.
- `state`: The state of the properties or properties. (`absent`, `present`)
- `system`: Is the property system defined.
- `updateexisting`: Should objects be updated with the new values. Only valid when updating a property, otherwise ignored.

4.7.2. Examples

```
- name: Set deinition for custom properties
  mm_props:
    name: location
    state: present
    proptype: text
    dest: zone
    provider:
      mmurl: http://mmsuite.example.net
      user: apiuser
      password: apipasswd
  delegate_to: localhost
```

4.8. mm_user

Manage user accounts and user properties on the Men&Mice Suite

4.8.1. Options

- `authentication_type`: Authentication type to use. e.g. Internal, AD. Required if `state=present`.
- `descr`: Description of the user.
- `email`: The users email address.
- `groups`: Make the user a member of these groups.
- `name`: (required) Name of the user to create, remove or modify.
- `password`: Users password (plaintext). Required if `state=present`.
- `provider`: (required) Definition of the Men&Mice suite API provider.
- `roles`: Make the user a member of these roles.
- `state`: Should the users account exist or not. (`absent`, `present`)

4.8.2. Examples


```
- name: Add the user 'johnd' as an admin
    mm_user:
    username: johnd
    password: password
    full_name: John Doe
    state: present
    authentication_type: internal
    roles:
        - Administrators (built-in)
        - DNS Administrators (built-in)
        - DHCP Administrators (built-in)
        - IPAM Administrators (built-in)
        - User Administrators (built-in)
        - Approvers (built-in)
        - Requesters (built-in)
    provider:
      mmurl: http://mmsuite.example.net
      user: apiuser
      password: apipasswd
  delegate_to: localhost

- name: Remove user 'johnd'
  mm_user:
    username: johnd
    state: absent
    provider:
      mmurl: http://mmsuite.example.net
      user: apiuser
      password: apipasswd
  delegate_to: localhost
```

4.9. mm_zone

Manage DNS zones in the Men&Mice Suite

4.9.1. Options

- `adintegrated`: True if the zone is Active Directory integrated.
- `adpartition`: The AD partition if the zone is Active Directory integrated.
- `adreplicationtype`: Type of the AD replication.
- `authority`: Name of the DNS server that contains the zone or the string `[Active Directory]` if the zone is integrated in the Active Directory.
- `customproperties`: Custom properties for the zone. These properties must already

exist. See also [mm_props].

- `dnssecsigned`: True if the zone is a DNSSEC signed zone.
- `dynamic`: Dynamic DNS zone.
- `kskids`: A comma separated string of IDs of KSKs, starting with active keys, then inactive keys in parenthesis
- `masters`: The IP addresses of the master servers if the new zone is not a master zone.
- `name`: (required) Name of the zone.
- `nameserver`: (required) Nameserver to define the zone on.
- `provider`: (required) Definition of the Men&Mice suite API provider.
- `servtype`: Type of the master server.
- `state`: The state of the zone. (`absent`, `present`)
- `zskids`: A comma separated string of IDs of ZSKs, starting with active keys, then inactive keys in parenthesis

4.9.2. Examples

```
- name: Create a new zone
  mm_zone:
    state: present
    name: example.com
    nameserver: ns1.example.com
    authority: mmsuite.example.net
    customproperties:
      location: Reykjavik
    provider:
      mmurl: http://mmsuite.example.net
      user: apiuser
      password: apipasswd
  delegate_to: localhost

- name: Release a zone
  mm_zone:
    state: absent
    name: example.com
    nameserver: ns1.example.com
    authority: mmsuite.example.net
    provider:
      mmurl: http://mmsuite.example.net
      user: apiuser
      password: apipasswd
  delegate_to: localhost
```

