Module defaults

If you frequently call the same module with the same arguments, it can be useful to define default arguments for that particular module using the `module_defaults` keyword.

Here is a basic example:

```
- hosts: localhost
  module_defaults:
    ansible.builtin.file:
      owner: root
      group: root
      mode: 0755
  tasks:
    - name: Create file1
      ansible.builtin.file:
        state: touch
        path: /tmp/file1

    - name: Create file2
      ansible.builtin.file:
        state: touch
        path: /tmp/file2

    - name: Create file3
      ansible.builtin.file:
        state: touch
        path: /tmp/file3
```

The `module_defaults` keyword can be used at the play, block, and task level. Any module arguments explicitly specified in a task will override any established default for that module argument.

```
- block:
    - name: Print a message
      ansible.builtin.debug:
        msg: "Different message"
  module_defaults:
    ansible.builtin.debug:
      msg: "Default message"
```

You can remove any previously established defaults for a module by specifying an empty dict.

```
- name: Create file1
  ansible.builtin.file:
    state: touch
    path: /tmp/file1
  module_defaults:
    file: {}
```

> **Note**
>
> Any module defaults set at the play level (and block/task level when using `include_role` or `import_role`) will apply to any roles used, which may cause unexpected behavior in the role.

Here are some more realistic use cases for this feature.

Interacting with an API that requires auth.

```
- hosts: localhost
  module_defaults:
    ansible.builtin.uri:
      force_basic_auth: true
      user: some_user
      password: some_password
  tasks:
    - name: Interact with a web service
      ansible.builtin.uri:
        url: http://some.api.host/v1/whatever1

    - name: Interact with a web service
      ansible.builtin.uri:
        url: http://some.api.host/v1/whatever2

    - name: Interact with a web service
      ansible.builtin.uri:
        url: http://some.api.host/v1/whatever3
```

Setting a default AWS region for specific EC2-related modules.

```
- hosts: localhost
  vars:
    my_region: us-west-2
  module_defaults:
    amazon.aws.ec2:
      region: '{{ my_region }}'
    community.aws.ec2_instance_info:
      region: '{{ my_region }}'
    amazon.aws.ec2_vpc_net_info:
      region: '{{ my_region }}'
```

Module defaults groups

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)playbooks_module_defaults.rst, line 105)

Unknown directive type "versionadded".

```
.. versionadded:: 2.7
```

Ansible 2.7 adds a preview-status feature to group together modules that share common sets of parameters. This makes it easier to author playbooks making heavy use of API-based modules such as cloud modules.

Group	Purpose	Ansible Version
aws	Amazon Web Services	2.7
azure	Azure	2.7
gcp	Google Cloud Platform	2.7
k8s	Kubernetes	2.8
os	OpenStack	2.8
acme	ACME	2.10
docker*	Docker	2.10
ovirt	oVirt	2.10
vmware	VMware	2.10

- The docker_stack module is not included in the `docker` defaults group.

Use the groups with `module_defaults` by prefixing the group name with `group/` - for example `group/aws`.

In a playbook, you can set module defaults for whole groups of modules, such as setting a common AWS region.

```
# example_play.yml
- hosts: localhost
  module_defaults:
    group/aws:
      region: us-west-2
  tasks:
  - name: Get info
    aws_s3_bucket_info:

  # now the region is shared between both info modules

  - name: Get info
    ec2_ami_info:
      filters:
        name: 'RHEL*7.5*'
```

In ansible-core 2.12, collections can define their own groups in the `meta/runtime.yml` file. `module_defaults` does not take the `collections` keyword into account, so the fully qualified group name must be used for new groups in `module_defaults`.

Here is an example `runtime.yml` file for a collection and a sample playbook using the group.

```
# collections/ansible_collections/ns/coll/meta/runtime.yml
action_groups:
  groupname:
    - module
    - another.collection.module
```

```
- hosts: localhost
  module_defaults:
    group/ns.coll.groupname:
      option_name: option_value
```

```
tasks:
  - ns.coll.module:
  - another.collection.module
```