Playbook Improvements

Use Modules

Whenever possible, use modules. Ansible modules are written in Python and run faster than shell commands.
Ansible cannot see the status of the shell command. It always shows "modified" by default which reduces or prevents the playbook idempotency.

Fact Gathering

In a job template the fact gathering only takes a few seconds but in a workflow template this can easily add up to minutes if every job template collects the facts again.

To avoid this you should ensure to collect facts only if you need them in your playbook.

It is also possible to collect only certain facts with a task during the playbook run. This can be achieved with the setup module.

https://docs.ansible.com/ansible/latest/collections/ansible/builtin/setup_module.html

```
- name: Collect only distribution facts
  ansible.builtin.setup:
    gather_subset:
      - '!all'
      - distribution
```

Async Tasks

By default Ansible waits for a task to complete before closing the network connection. For tasks which take some time this can slowdown the whole playbook.

If the following tasks have no dependency on the current tasks you can use async with an polling interval.

Ansible will then start the execution of the task and check the progress of it by polling the state in the defined interval.

Even if the task is not complete it will continue with the next tasks.

```
- name: Initiate custom snapshot
  shell:
    "my_long_running_command"
  async: 120 # maximum allowed time (seconds) to finish the task
  poll: 05 # polling interval (seconds)
```

Execution Strategies

https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html

linear

Ansible has different execution strategies, by default it uses a strategy which is called "linear".
This strategy executes a task and wait until the task has finished on all hosts of the play.
Which means the entire task is only as fast as it is on the slowest host.

This can create a big bottleneck if one host has network issues or in under heavy load and needs way longer to execute the tasks.

free

Another execution strategy is called "free", this can get used If there are no dependencies between the hosts. Then it is not required to wait for the slowest host and Ansible can finish faster hosts first.

If the playbook has finished on one of the hosts it will free one fork and be able to start on the next one without the need to wait for the slower hosts.

```
- hosts: myservers
  strategy: free
  tasks:
    ...
```