Tame the Chain:

Smart VM Dependency Management with Ansible

James Freeman

@jamesfreeman959
James Freeman

Who Am I?

- Sr Technical Account Manager at AWS
- Published author (Ansible)
- Reiki Master Teacher
- Autism Dad

Agenda

- Problem Statement
- How Ansible can help
- Solution 1 – Ansible, Jinja2, Systemd
- Solution 2 – Ansible, Custom Filter Plugin
- Lessons Learned

Problem Statement

- Home Lab relies on multiple VM’s running on Linux
- Startup and shutdown order matters
  - Start Firewall first
  - Start AD Server second (authentication, DNS)
  - Start NAS
  - Start Docker storage VM
  - Start all other VM’s

But...

This isn’t specifically an ordering problem - it’s a dependency problem

Problem Statement – the reality (v2)

- Home Lab relies on multiple VM's running on Linux
- Startup and shutdown order matters
  - All VM's are dependent on the firewall
  - AD Server depends on the firewall
  - NAS VM depends on AD being up (Samba)
  - Docker storage VM depends on firewall
  - Docker worker nodes are dependent on NAS and Docker Storage VM
  - Other VM's (e.g. AWX) can be started in any order after firewall comes up

Possible Solutions

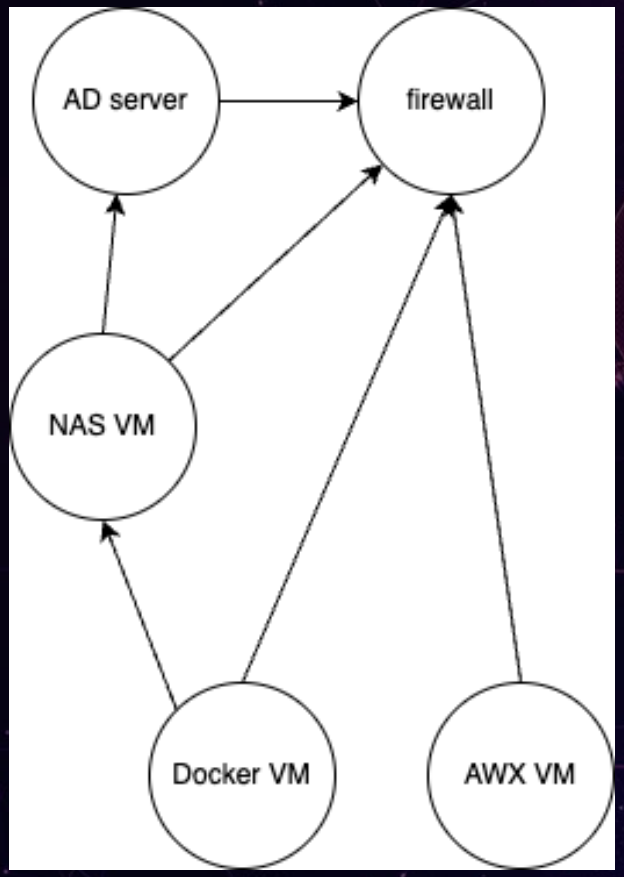
- Libvirt/QEMU autostart
  - No ordering possible
- Bash Script
  - Possible but doesn’t lend itself to dependency management
- Python Script
  - Better, but Ansible already has libvirt modules, wait_for
- Ansible
  - Has everything we need in one place

Specify the dependencies

```
$ cat host_vars/hypervisor.example.com.yml
---
vms:
  - name: firewall
    depends_on: []
  - name: adserver
    depends_on:
      - firewall
  - name: nas
    depends_on:
      - adserver
      - firewall
  - name: dockervm
    depends_on:
      - nas
      - firewall
  - name: awx
    depends_on:
      - firewall
```

Specify the dependencies

- As a graph…



Solution 1 – Ansible, Jinja2, Systemd

- Systemd has built in dependency logic
- Can specify startup and shutdown with the system
- Why not use it for the heavy lifting?
- Get Ansible to build the `.service` files from Jinja2 template

Jinja2 solution – part 1

```
[Unit]
Description=VM - {{ item.name }}
{% if item.depends_on %}
After=network.target {% for dep in item.depends_on %} {{ dep }}.service{% if
not loop.last %} {% endif %}{% endfor +%}
Requires={% for dep in item.depends_on %}{{ dep }}.service{% if not loop.last
%} {% endif %}{% endfor +%}
{% else %}
After=network.target
Requires=
{% endif %}
```

Jinja2 solution – part 2

```
{% set dependent_vms = vms | selectattr('depends_on', 'defined') |
selectattr('depends_on', 'contains', item.name) | map(attribute='name') | list
%}
{% if dependent_vms %}
Before={% for dep in dependent_vms %}{{ dep }}.service{% if not loop.last %} {%
endif %}{% endfor +%}
{% endif %}
PartOf=vm-group.target

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/bin/bash -c 'virsh domstate {{ item.name }} | grep -q running ||
virsh start {{ item.name }}'
ExecStop=/usr/bin/virsh shutdown {{ item.name }}

[Install]
WantedBy=vm-group.target
```

Things I learned from Solution 1

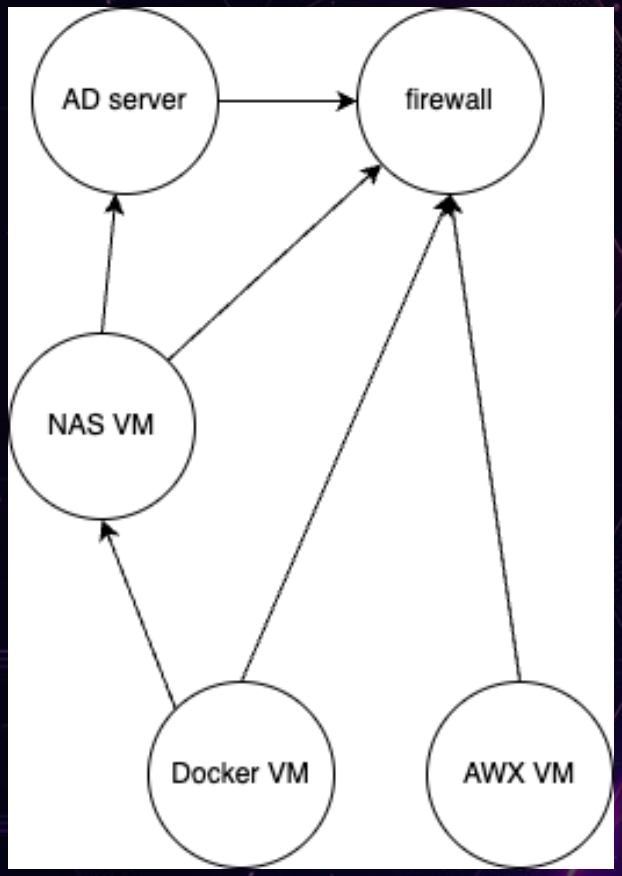
- Systemd startup and shutdown ordering is complex
- Multiple directives specify similar but subtly different things
- For example:
  - `Requires=` - Units that are required to be started before this one
  - `After=` - Units that must be started before this one if multiple exist at the same level
- Shutdown sequencing didn't work for me
  - ChatGPT didn't know either
- `virsh [start|destroy]` are asynchronous to VM state

Solution 2 – Custom filter plugin

- Remember I said ChatGPT didn't know?
- This solution was inspired by a hallucination
- ChatGPT (and Claude and Gemini) all said use something like:
  - `community.general.graph_topological_sort`
- No such collection or plugin
- BUT…
  - You can ask GenAI to write what it just hallucinated!

Remember the dependencies?

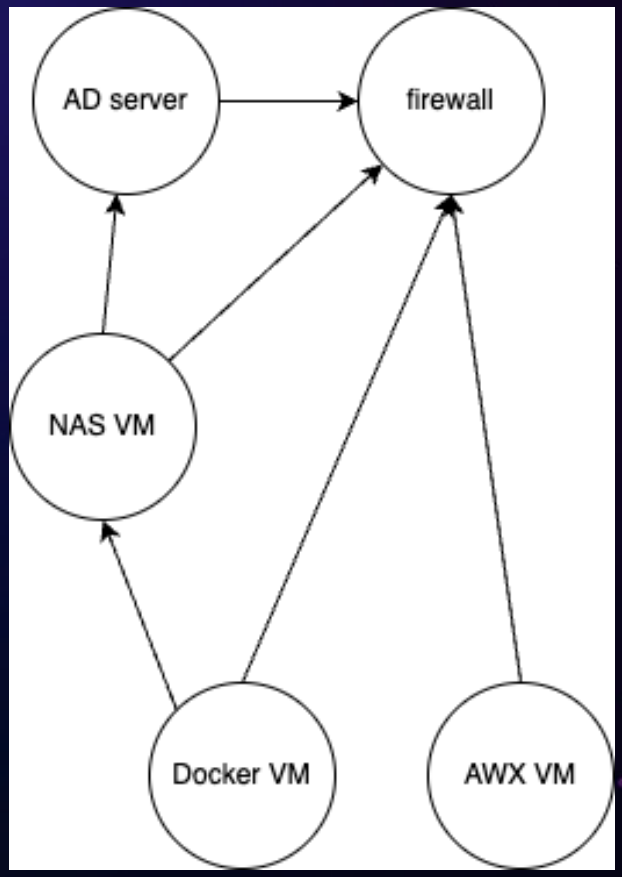
- They are a graph…
- We need them in some sort of linear structure
- ChatGPT says use "`community.general.graph_topological_sort`"
- No such collection exists!
- But it can write it for us…



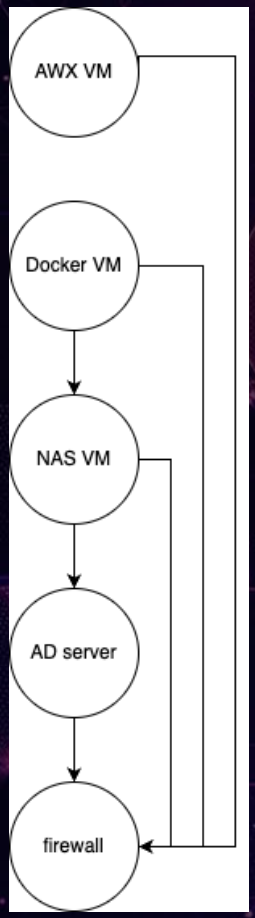
What is a Graph Topological Sort?

- Simply, it sorts a graph into a simple array where each node appears before all the nodes it points at
  - https://www.interviewcake.com/concept/java/topological-sort
- Does NOT work with cyclic graphs
  - But cyclic dependencies would break things anyway
- So we want…

Graph Topological Sort output



Becomes...



Filter code

```
from ansible.errors import AnsibleFilterError

def graph_topology_sort(graph):
    sorted_items = [] # List to hold sorted nodes
    visited = {} # Tracks visitation status of each node
    def visit(node):
        if node in visited:
                if visited[node] == 'temporary’:
                        # A node visited temporarily again means a cycle
                        raise AnsibleFilterError(f"Cyclic dependency detected: {node}")
                return # Already permanently visited; nothing to do
        visited[node] = 'temporary' # Mark node as temporarily visited
        # Recursively visit dependencies first
        for dependency in graph.get(node, []):
                visit(dependency)
        visited[node] = 'permanent' # Mark node as permanently visited
        sorted_items.append(node) # Append node after its dependencies
    for node in graph:
        visit(node)
    return sorted_items # Return list of nodes sorted correctly
class FilterModule(object):
    def filters(self):
        return {'graph_topology_sort': graph_topology_sort}
```

Extending the example

- Add `wait_for` functionality

```yaml
---
vms:
  - name: firewall
    depends_on: []
    waitfor_port: 22
    waitfor_addr: firewall.example.com
  - name: adserver
    depends_on:
      - firewall
    waitfor_port: 3389
    waitfor_addr: adserver.example.com
```

Setting up the play

```yaml
vars:
  vms_dict: "{{ dict(vms | map(attribute='name') | zip(vms)) }}"
  vm_graph: >-
    {{
      dict(vms | map(attribute='name') |
      zip(vms | map(attribute='depends_on')))
    }}

tasks:
  - name: Display VM dependency graph
    ansible.builtin.debug:
      var: vm_graph

  - name: Compute sorted VM list using custom filter
    ansible.builtin.set_fact:
      sorted_vms: "{{ vm_graph | graph_topology_sort }}"
```

Starting the VM’s (you can’ t loop over a block)

```yaml
- name: Start and Wait for VMs
  ansible.builtin.include_tasks:
    file: start-wait-vm.yml
  loop: "{{ sorted_vms }}"
  loop_control:
    loop_var: vmname
  when: vmname in vms | map(attribute='name') | list
```

Start each VM and wait for it to come up

```yaml
- name: Start the VM using the appropriate libvirt module
  community.libvirt.virt:
    name: "{{ vms_dict[vmname].name }}"
    state: running

- name: Wait for VM to become available only if waitfor_addr and
waitfor_port are defined
  ansible.builtin.wait_for:
    host: "{{ vms_dict[vmname].waitfor_addr }}"
    port: "{{ vms_dict[vmname].waitfor_port }}"
    timeout: 300
  when: vms_dict[vmname].waitfor_addr is defined and
vms_dict[vmname].waitfor_port is defined
```

Stopping the VM's (spot the difference!)

```yaml
- name: Start and Wait for VMs
  ansible.builtin.include_tasks:
    file: stop-wait-vm.yml
  loop: "{{ sorted_vms | reverse }}"
  loop_control:
    loop_var: vmname
  when: vmname in vms | map(attribute='name') | list
```

Stop each VM and wait for it to shut down

```yaml
- name: Stop the VM using the appropriate libvirt module
  community.libvirt.virt:
    name: "{{ vms_dict[vmname].name }}"
    state: stopped
```

(More) Things I learned

- VSCode+Copilot+`ansible-lint` is a time-saver
- Hallucinations can lead to solutions
  - Just because it was hallucinated, doesn't mean it can't be built
- GenAI should be a tool in your arsenal (it probably is)
  - Just be aware of its limitations
  - (it said my Systemd code would work…)
- `pipx` is a great way to install Ansible
- `ansible-lint` lives in the `ansible-dev-tools` package

Next Steps

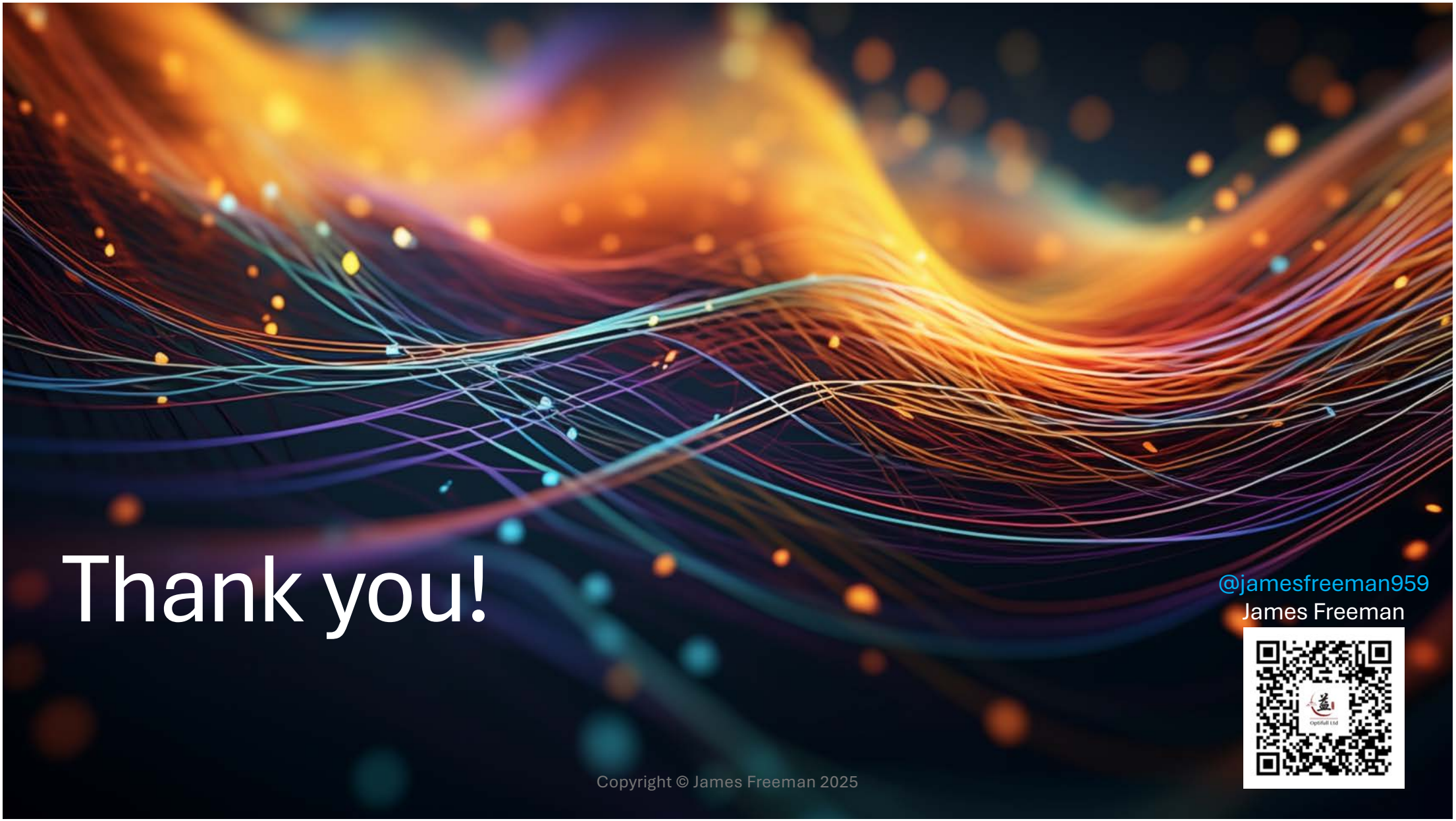
- Ideas for further development
  - Add `graph_topology_sort` to a collection and submit to Ansible Galaxy
  - Learn the intricacies of Systemd startup/shutdown order
    - Plus add equivalent “wait for” function to the startup code
  - Turn this whole example into a collection?

Demo code

- https://github.com/jamesfreeman959/ansible-vm-dependency-ordering

Feedback (optional)

Questions?



Thank you!

@jamesfreeman959
James Freeman