OS PATCHING MIT ANSIBLE

29.10.2020

ROLLENSPIELE MIT WINDOWS UND LINUX

PHILIPP HOFFMANN

Systems Engineer
Automation Services

SPIRIT/21

Agenda

1. Scope & Szenario
2. Inventory Gruppenstruktur
3. Nested Playbooks (Applikationsintegration)
4. Linux und Windows unter einem Hut
5. Details zur Linux Patch Installation (SIGHUP)
6. Details zur Windows Patch Installation

Scope

- Clientseitiges Patching von Linux und Windows Systeme in heterogenen Landschaften
- Berücksichtigung von Cluster Szenarien
- Berücksichtigung von Applikations Start-Stopps
- Ausführung von beliebigem Aktionen im Wartungsfenster
- Erfahrungen weitergeben

Szenario (YMMV)

- Ausführung über Ansible Tower
- Inventory und Gruppen werden über Cloudforms erstellt
- Linux Repositories per Repo-File Config
- Windows Patches per WSUS
- Systeme auf Bare-Metal, VMware und AWS
- Service Provider Infrastruktur mit Kundenkontexten (Mandantenfähigkeit)

Inventory Gruppenstruktur

- Gruppe für Kunden
- Gruppen für Patchwave
- Gruppe für Subpatchwave → manuelle Serialisierung

Kunde	PatchWave	Subpatchwave
customer_a	wave1	subwave_a
customer_b	wave2	subwave_b
customer_c	wave3	subwave_c
customer_d	wave4	subwave_d
customer_e	wave5	

Inventory Gruppenkombinationen

```yaml
all:
  customer_a:
    hosts:
      server_a
      server_b
      server_c
      server_d
      server_e
      server_f
      server_g
      server_h
      server_i
      server_j
      server_k
      server_l
      server_m
  customer_b:
    hosts:
      server_n
      server_o
      server_p
```

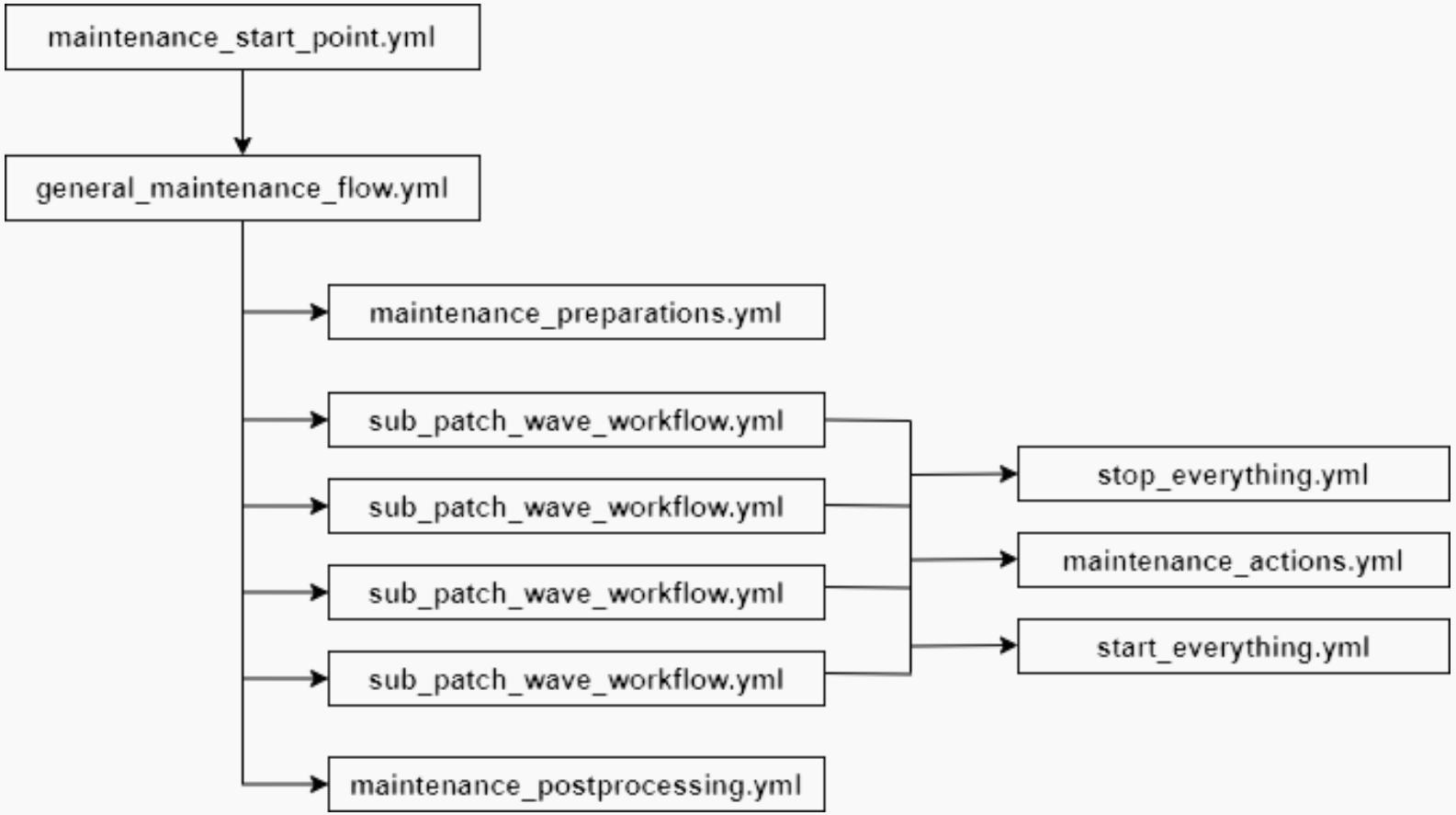
```yaml
all:
  wave1:
    hosts:
      server_a
      server_b
      server_c
      server_d
      server_e
      server_f
      server_n
  wave2:
    hosts:
      server_g
      server_h
      server_i
      server_o
  wave3:
    hosts:
      server_j
      server_k
      server_l
      server_m
      server_p
```

```yaml
all:
  subwave_a:
    hosts:
      server_a
      server_b
      server_c
      server_j
      server_n
      server_o
      server_p
  subwave_b:
    hosts:
      server_d
      server_e
      server_f
      server_k
  subwave_c:
    hosts:
      server_l
  subwave_d:
    hosts:
      server_m
```

Inventory Applikationsgruppen

```
all:
  tomcat:
    hosts:
      server_a
  atlassian:
    hosts:
      server_b
      server_j
```

Playbook Struktur Nested Playbooks



Minimum Variable Set

- execution_variant:
- maintenance_actions:
- customer:

Ebene 1: execution_variant

maintenance_start_point.yml | general_maintenance_flow.yml | sub_patch_wave_workflow.yml | maintenance_actions.yml

```yaml
- hosts: "{{ ansible_limit }}"
  tasks:
    - name: "Structured maintenance - standard mode"
      include_tasks: tasks/general_maintenance_flow.yml
      when: execution_variant == "standard"

    - name: "Structured maintenance - Only Windows 2016"
      include_tasks: tasks/general_maintenance_flow.yml
      when:
        - execution_variant == "only-win2016"
        - ansible_distribution in ["Microsoft Windows Server 2016 Datacenter", "Microsoft Windows Server 2016 Standard"]

    - name: "Structured maintenance - Only Windows without 2016"
      include_tasks: tasks/general_maintenance_flow.yml
      when:
        - execution_variant == "no-win2016"
        - ansible_distribution not in ["Microsoft Windows Server 2016 Datacenter", "Microsoft Windows Server 2016 Standard"]
        - ansible_os_family == "Windows"
```

Playbook Struktur Nested Playbooks
Ebene 2: grobe Prozessschritte

maintenance_start_point.yml | general_maintenance_flow.yml | sub_patch_wave_workflow.yml | maintenance_actions.yml

```yaml
- include_tasks: maintenance_preparations.yml

- name: Sub-Patch-Wave-A
  include_tasks: sub_patch_wave_workflow.yml
  when: "'sub_patch_wave_a' in group_names"

- name: Sub-Patch-Wave-B
  include_tasks: sub_patch_wave_workflow.yml
  when: "'sub_patch_wave_b' in group_names"

- name: Sub-Patch-Wave-C
  include_tasks: sub_patch_wave_workflow.yml
  when: "'sub_patch_wave_c' in group_names"

- name: Sub-Patch-Wave-D
  include_tasks: sub_patch_wave_workflow.yml
  when: "'sub_patch_wave_d' in group_names"

- include_tasks: maintenance_postprocessing.yml
```

Playbook Struktur Nested Playbooks
Ebene 3: Subwave

maintenance_start_point.yml | general_maintenance_flow.yml | sub_patch_wave_workflow.yml | maintenance_actions.yml

```yaml
1
2  - name: Stop Everything
3    include_tasks: stop_everything.yml
4
5  - name: "Execute maintenance action(s)"
6    include_tasks: maintenance_actions.yml
7
8  - name: Start everything.yml
9    include_tasks: start_everything.yml
10
```

Nested Playbook

Ebene 4.1: Stop - Prozeduren

stop_everything.yml | stop_general.yml | stop_customerx.yml

```yaml
- name: check if customer exists
  local_action:
    module: stat
    path: "{{ playbook_dir }}/tasks/stop/stop_{{ customer }}.yml"
  register: stop_yml
  check_mode: no
  become: no # Ansible Tower security

- name: "Stop Apps for customer {{ customer }}"
  include_tasks: "stop/stop_{{ customer }}.yml"
  when: stop_yml.stat.exists

- name: "Stop DB + SAP"
  include_tasks: "stop/stop_general.yml"
```

stop_everything.yml | stop_general.yml | stop_customerx.yml | stop_

```yaml
- name: 'Stop Tomcat'
  command: ./shutdown.sh
  args:
    chdir: /var/lib/tomcat/bin/
  when: "'tomcat' in group_names"

- name: 'Stop Services'
  service:
    name: "{{ item }}"
    state: stopped
  with_items:
    - "postgresql-9.6"
    - "jira"
    - "confluence"
  when: "'atlassian' in group_names"
```

stop_everything.yml | stop_general.yml | stop_customerx.yml

```yaml
- name: '[General][DB2] Stop'
  include_role:
    name: dbms_db2_operations
  vars:
    - operation: stop

- name: '[General][Oracle] Stop'
  include_role:
    name: dbms_oracle_operations
  vars:
    - operation: shutdown

- name: '[General][SAP] Stop'
  include_role:
    name: sap_operations
  vars:
    - operation: stop
```

Playbook Struktur Nested Playbooks
Ebene 4.2: die Wartungsaktion

maintenance_start_point.yml | general_maintenance_flow.yml | sub_patch_wave_workflow.yml | maintenance_actions.yml

```yaml
- name: 'Patch everything'
  include_role:
    name: os_patching
  vars:
    - operation: install
    - auto_reboot: yes
  when: "'patching' in maintenance_actions"

- name: 'Reboot'
  win_reboot:
    reboot_timeout: 3600
  when:
    - ansible_os_family == "Windows"
    - "'reboot' in maintenance_actions"

- name: 'Reboot'
  reboot:
    reboot_timeout: 3600
  when:
    - ansible_os_family != "Windows"
    - "'reboot' in maintenance_actions"


- name: 'VMware Settings'
  include_role:
    name: vmware_settings.yml
  when: "vmware_settings" in maintenance_actions

```

Nested Playbook

Ebene 4.3: Start - Prozeduren

start_everything.yml | start_general.yml | start_customerx.yml

```yaml
- name: "Start DB + SAP"
  include_tasks: start/start_general.yml

- name: check if customer exists
  local_action:
    module: stat
    path: "{{ playbook_dir }}/start/start_{{ customer }}.yml"
  register: start_yml
  check_mode: no
  become: no # Ansible Tower security

- name: "Start Apps for customer {{ customer }}"
  include_tasks: "start/start_{{ customer }}.yml"
  when: start_yml.stat.exists

```

start_everything.yml | start_general.yml | start_customerx.yml

```yaml
- name: '[DB2] Start'
  include_role:
    name: dbms_db2_operations
  vars:
    - operation: start

- name: '[Oracle] Start'
  include_role:
    name: dbms_oracle_operations
  vars:
    - operation: startup

- name: '[SAP] Start'
  include_role:
    name: sap_operations
  vars:
    - operation: start

```

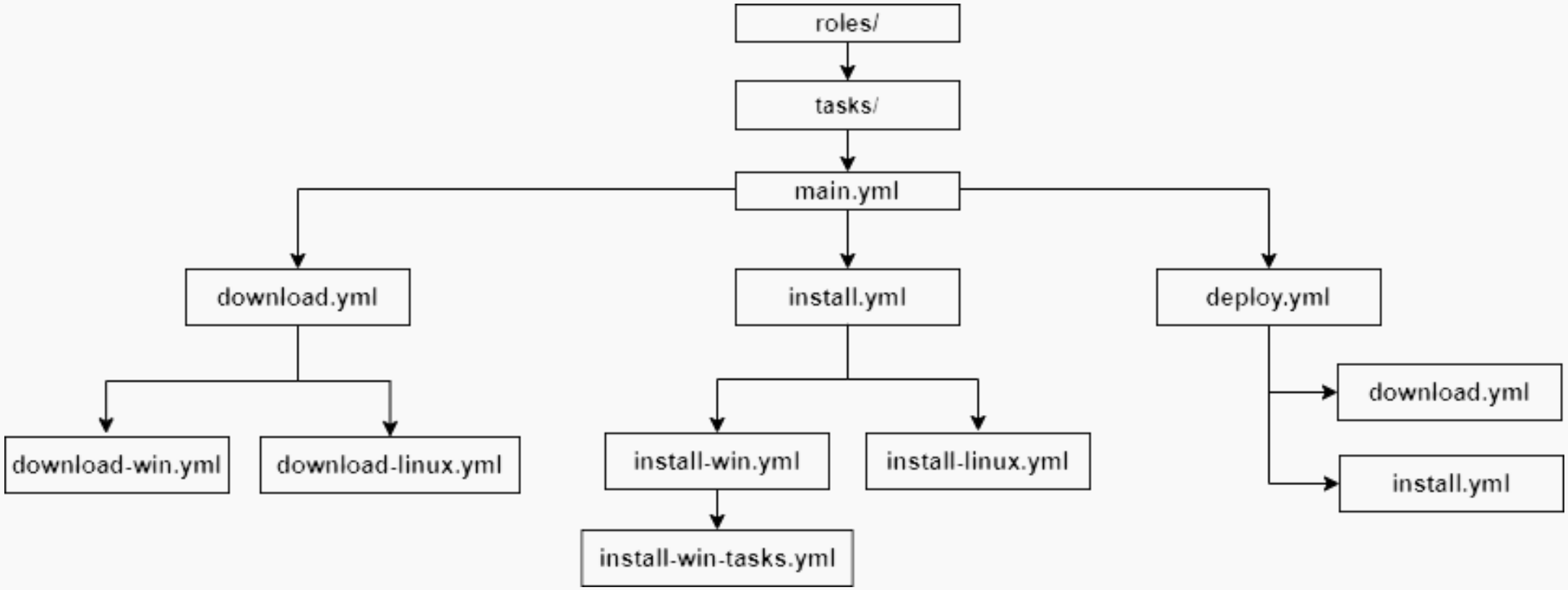
start_everything.yml | start_general.yml | start_customerx.yml

```yaml
- name: 'Start Tomcat'
  command: ./startup.sh
  args:
    chdir: /var/lib/tomcat/bin/
  when: "'tomcat' in group_names"

- name: 'Start Services'
  service:
    name: "{{ item }}"
    state: started
  with_items:
    - "postgresql-9.6"
    - "jira"
    - "confluence"
  when: "'atlassian' in group_names"
```

Linux und Windows unter einem Hut

mit Ansible Rollen



1. Ebene: ‘operation‘ Entscheidung

main.yml | install.yml | install-linux.yml | install-win.yml | install-win-tasks.yml

```yaml
# ************************************************************************
# * Manage OS Updates
# ************************************************************************
---
- name: "Listing parameters"
  fail:
    msg: "Please set operation parameter"
  when: operation == 'noop'

- name: Download Updates
  include_tasks: download.yml
  when: operation == 'download'

- name: Install Updates
  include_tasks: install.yml
  when: operation == 'install'

- name: Deploy (Download & Install) Updates
  include_tasks: deploy.yml
  when: operation == 'deploy'

```

2. Ebene: Betriebssystem Entscheidung

main.yml | install.yml | install-linux.yml | install-win.yml | install-win-tasks.yml

```yaml
# ************************************************************************
# * Install OS Updates
# ************************************************************************
---
- name: Include windows installation tasks
  include_tasks: install-win.yml
  when: ansible_os_family == "Windows"

- name: Include linux installation tasks
  include_tasks: install-linux.yml
  when: ansible_os_family in ['RedHat', 'Debian', 'Suse']
```

3.1 Ebene: Betriebssystemspezifische Aktionen

main.yml | install.yml | install-linux.yml | install-win.yml | install-win-tasks.yml | install-win.yml

```yaml
# ****************************************************************
# * Install available Linux Updates for major distributions (For Linux only)
# ****************************************************************
---

- name: YUM - upgrade all packages # noqa 403
  yum:
    name: '*'
    state: latest
  register: yum_result
  when: ansible_pkg_mgr == "yum"
  retries: 3
  until: yum_result is not failed

- name: YUM - Clean Archive # noqa 303 305
  shell: yum clean all
  when: ansible_pkg_mgr == "yum"

- name: APT - Upgrade all packages to the latest version # noqa 403
  apt:
    name: "*"
    state: latest
    update_cache: yes
  register: apt_result
  when: ansible_pkg_mgr == "apt"
  retries: 3
  until: apt_result is not failed
```

3.2 Ebene: Betriebssystemspezifische Aktionen

main.yml | install.yml | install-linux.yml | install-win.yml | install-win-tasks.yml

```yaml
- name: Reboot before patch installation
  win_reboot:
    reboot_timeout: 3600
  when: pre_reboot

- name: Include windows installation sub tasks
  include_tasks: install-win-tasks.yml
  loop: "{{ range(0, max_reboots)|list }}"

```

4. Ebene: „Block“-Loop

main.yml | install.yml | install-linux.yml | install-win.yml | install-win-tasks.yml

```yaml
- name: "Execute Script - Install Windows Updates"
  script: files/{{ script_ps_install_windowsupdates }}
  register: update_install_result
  retries: 10
  delay: 20
  until:
    - update_install_result is succeeded
    - update_install_result is defined

- name: "Execute Script - Test Pending Reboot"
  script: files/{{ module_pending_reboot }}
  register: pending_reboot
  retries: 3
  delay: 20
  until:
    - pending_reboot is succeeded
    - pending_reboot is defined

- name: trim whitespaces for reboot var
  set_fact:
    reboot_required: "{{ pending_reboot.stdout | trim | default(False) }}"
  when: not ansible_check_mode

- name: Reboot
  win_reboot:
    reboot_timeout: 3600
  when: auto_reboot and reboot_required
```

Linux Patch Installation

- dnf
- yum
- zypper
- apt

Linux SIGHUP

- Entgegen dem aktuellen Bash Standardverhalten werden alle Child-Prozesse beim Beenden von einem Ansible Task beendet. Alle Applikationen die über simple Shell Befehle gestartet werden, werden sehr schnell wieder beendet.
- SO-Artikel über Bash SIGHUP Verhalten
- SO-Artikel zu Ansible SIGHUP Verhalten (alt und unbeantwortet)
- Diskussion zu SIGHUP im Ansible Github

Linux SIGHUP Statement

jborean93 commented on 25 Sep 2018 — Contributor

Is This A Bug?

Hi!

Thanks very much for your submission to Ansible. It sincerely means a lot to us.

We're not sure this is a bug, and we don't mean for this to be confrontational. Let's explain what we're thinking:

- Each task is run over SSH and are based on the terminal created for that session
- Once the parent process is finished, the terminal is closed which then means any other processes associated with that terminal are also killed
- This is standard practice for SSH sessions and the only way to avoid this is to have the process detach itself from the terminal
- You can use async or a nohup command to achieve this if you want
- But, you are probably better off running this as a proper service which gives you a lot more benefits than what you can get with async or a nohup command.

As such, we're going to close this ticket. However, we're open to being corrected, should you wish to discuss. You can stop by one of our two mailing lists
to talk about this and we might be persuaded otherwise.

- https://groups.google.com/forum/#!forum/ansible-project - for user questions, tips, and tricks
- https://groups.google.com/forum/#!forum/ansible-devel - for strategy, future planning, and questions about writing code

Comments on closed tickets aren't something we monitor, so if you do disagree with this, a mailing list thread is probably appropriate.

Thank you once again for this and your interest in Ansible!

1 1

Windows Patch Installation

- Ansible Module für Windows Updates

- "Microsoft.Update.Session"

  Ausführliches Beispiel zur Verwendung mit VBS

  Thread mit Code-Beispiel zur Verwendung mit Powershell

Windows Patch Installation

Minimal-Beispiel Powershell Script

```powershell
$ErrorActionPreference = "Stop"

4 references
function Install-WindowsUpdates {
    $result = @{}
    $updateSession = new-object -com "Microsoft.Update.Session"
    $criteria = "IsInstalled=0"
    $updates = $updateSession.CreateupdateSearcher().Search($criteria).Updates

    if ($updates.Count -gt 0) {
        $downloader = $updateSession.CreateUpdateDownloader()
        $downloader.Updates = $updates
        $downloader.Download()
        $installer = $updateSession.CreateUpdateInstaller()
        $updatesToInstall = New-object -com "Microsoft.Update.UpdateColl"
        $updates | foreach-Object {$updatesToInstall.Add($_)} | out-null
        $installer.Updates = $updatesToInstall
        $installationResult = $installer.Install()
      if ($installationResult -ne $null) {
        $result['UpdateCount'] = $updates.Count
      }
    } else {
        $result['UpdateCount'] = 0
    }
    $result | ConvertTo-Json
}

Install-WindowsUpdates
```

Windows Patching Privilege Escalation

- Zum Installieren von Patches benötigt man die höchsten Privilegien

```
---
hosts: my_hosts
gather_facts: yes
tasks:
  - name: Windows OS Patching
    # this module does not exist
    install_patches:
      action: install
    become: yes
    become_user: SYSTEM
    become_method: run_as
```

WinRM Stabilität

- WinRM läuft nicht so stabil, wie man sich das wünschen würde.

```
# One way to ensure the system is reliable just after a reboot, is to set WinRM to a delayed startup
- name: Ensure WinRM starts when the system has settled and is ready to work reliably
  ansible.windows.win_service:
    name: WinRM
    start_mode: delayed
```

```
1  - name: "Execute Script - Install Windows Updates"
2    script: files/{{ script_ps_install_windowsupdates }}
3    register: update_install_result
4    retries: 10
5    delay: 20
6    until: update_install_result is succeeded
7
```

Windows Multi-Installation / Multi-Reboot

multi-reboot-1-install-win.yml | multi-reboot-2-install-win-tasks.yml

```yaml
- name: Include windows installation sub tasks
  include_tasks: multi-reboot-2-install-win-tasks.yml
  loop: "{{ range(0, 4)|list }}"

```

Weitere Themen

- GitHub Link: https://github.com/phospi/ansible_anwenderforum_2020
- Laufzeitbegrenzung
- Umgang mit fehlschlagenden Reboots
- Anzahl paralleler Tasks (Forks)
- Package Reports