Ansible

Sander Kuusemets
sander.kuusemets@ut.ee
Veebruar 2026

Treeningu eesmärgid

Päev 1

Ansible baastarkused

Õpime kuidas Ansible töötab ja käitub, vaatame ja proovime järgi kuidas ehitada enda automaatikat.

Ansible edasijõudnud kontseptsioonid

Tutvume variantidega, mis muudavad Ansible kasutuse lihtsamaks keerulistes keskkondades.

Päev 2

Ansible kasutuse kogemused

Kuulame kogemusi ja soovitusi 10 aastasest Ansible kasutuse kogemusest kasvavas tiimis.

Ansible ökosüsteem

Vaatame, mis on maailmas veel Ansible jaoks ehitatud, mis võiks meie elu lihtsamaks teha.

Ansible treeningu plaan

Loenguvormis ja praktikumi vormis treening, kordamööda.

Sissejuhatus

Praktiline osa

Ansible projekt ja playbook

Praktiline osa

Ansible muutujad ja tingimused

Praktiline osa

... jms

Teisel päeval vaatame edasijõudnute asju, vabamas vormis, kasutades olemasolevat dokumentatsiooni ja reaalelulisi näiteid.

Ansible praktilised osad

Selleks, et proovida omal käel Ansible head-vead järgi, on igaühele jagatud ligipääs neljale virtuaalmasinale.

Kaks tükki neist, nimedega CentOS ja Debian, on tühjad Linux virtuaalmasinad, loodud olema liivakast kus testida mida Ansiblega teha saab Linux keskkonnas. Nendele saate ligi SSH'ga üle **root** kasutaja. Hiljem võtame kasutusele ka Windows masina, et proovida selle haldust.

Viimane masin, millele on muu kasutaja, on selleks kui ei saa enda masinast Ansiblet jooksutada.

Kõik praktilise osa juhendid on saadaval siin:

http://ansible.hashify.ee

SANDER KUUSEMETS

Töökogemus

2020 - praegu **Infrastruktuuri grupijuht** Tartu Ülikool, teadusarvutuste keskus

- Tehniline juht 15-liikmeselisele infrastruktuuri tiimile
- HPC klaster, pilveteenus, self-service, kubernetes

2018 - praegu **Süsteemihalduse + Kubernetese + DevOps ainete lektor** Tartu Ülikool

- Linux põhise süsteemihalduse kursuse loengud
- Tudengitele baas-IT teenuste kogemuse andmine

2014 - 2020 **Süsadmin** Tartu Ülikool

- Tartu Ülikooli infrastruktuuri ja teenuste haldamine
- Ansiblet hakkasin siin kasutama

Haridus

2018 - praegu **Informaatika magister**

2013 - 2018 **Arvutitehnika bakalareus**

Ansible sissejuhatus ja arhitektuur

01

Ansible - mis see on?

Ansible is an open source IT automation engine that automates provisioning, configuration management, application deployment, orchestration, and many other IT processes. It is free to use, and the project benefits from the experience and intelligence of its thousands of contributors.

- Ansible koduleht

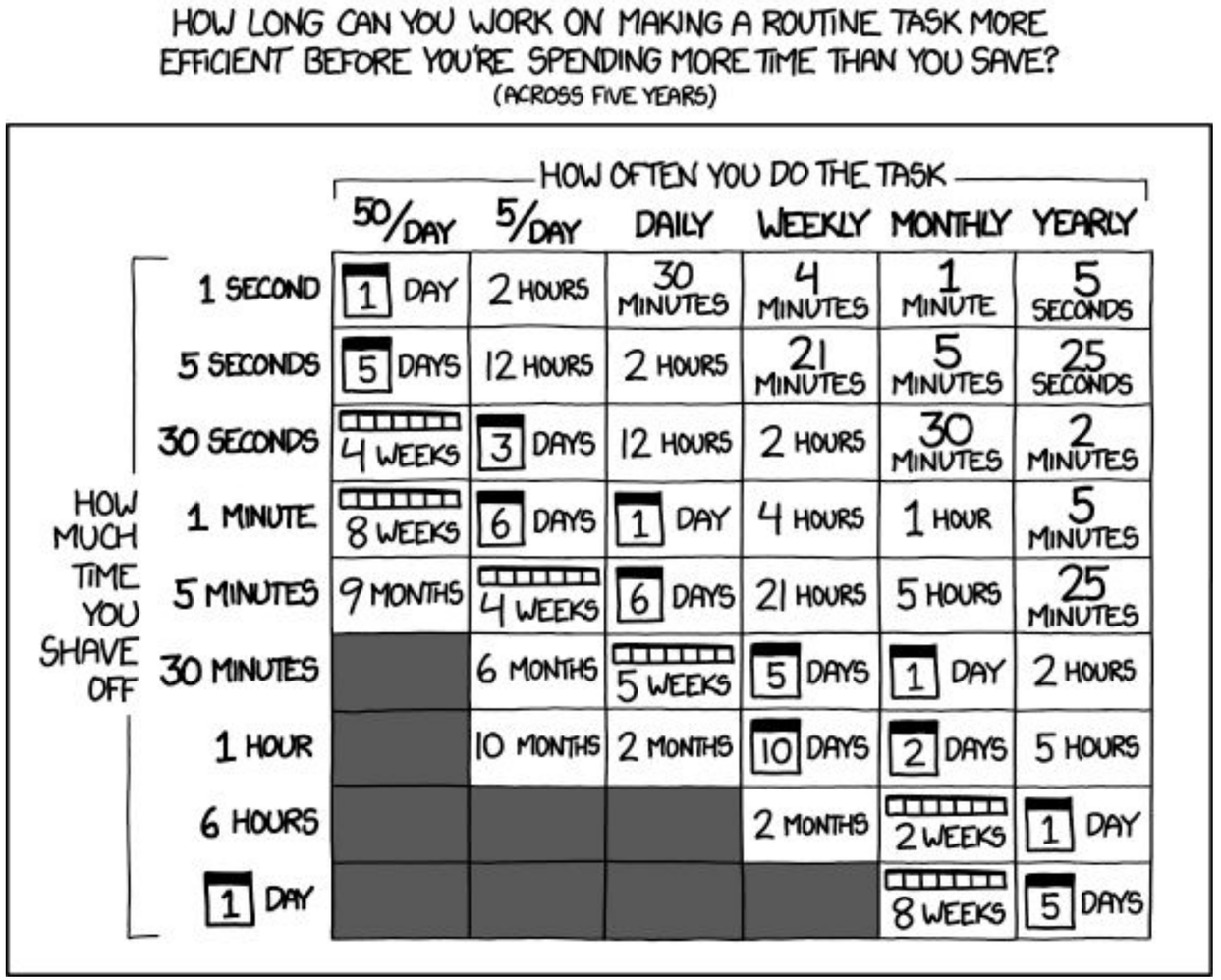
Ansible - milleks seda kasutatakse?

Meie vaatame Ansible-t põhiliselt just konfiguratsioonihalduse ja automaatika seisukohalt.

Ansible - miks seda kasutatakse?

Ansible-t kasutame põhiliselt selleks, et lihtsustada/kiirendada protsesse.

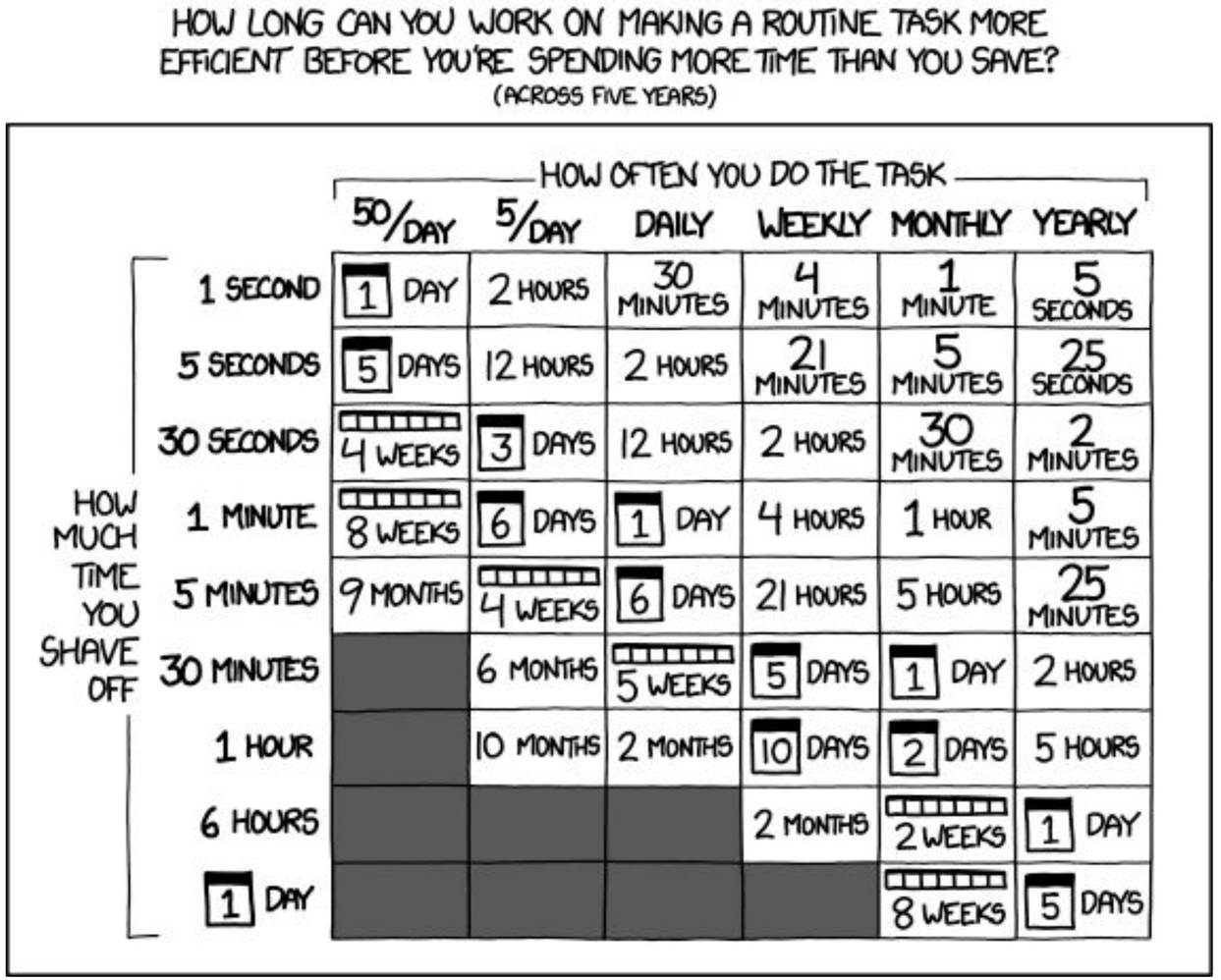
Ansible lubab meil OS/tarkvara kihi halduse ära automatiseerida.



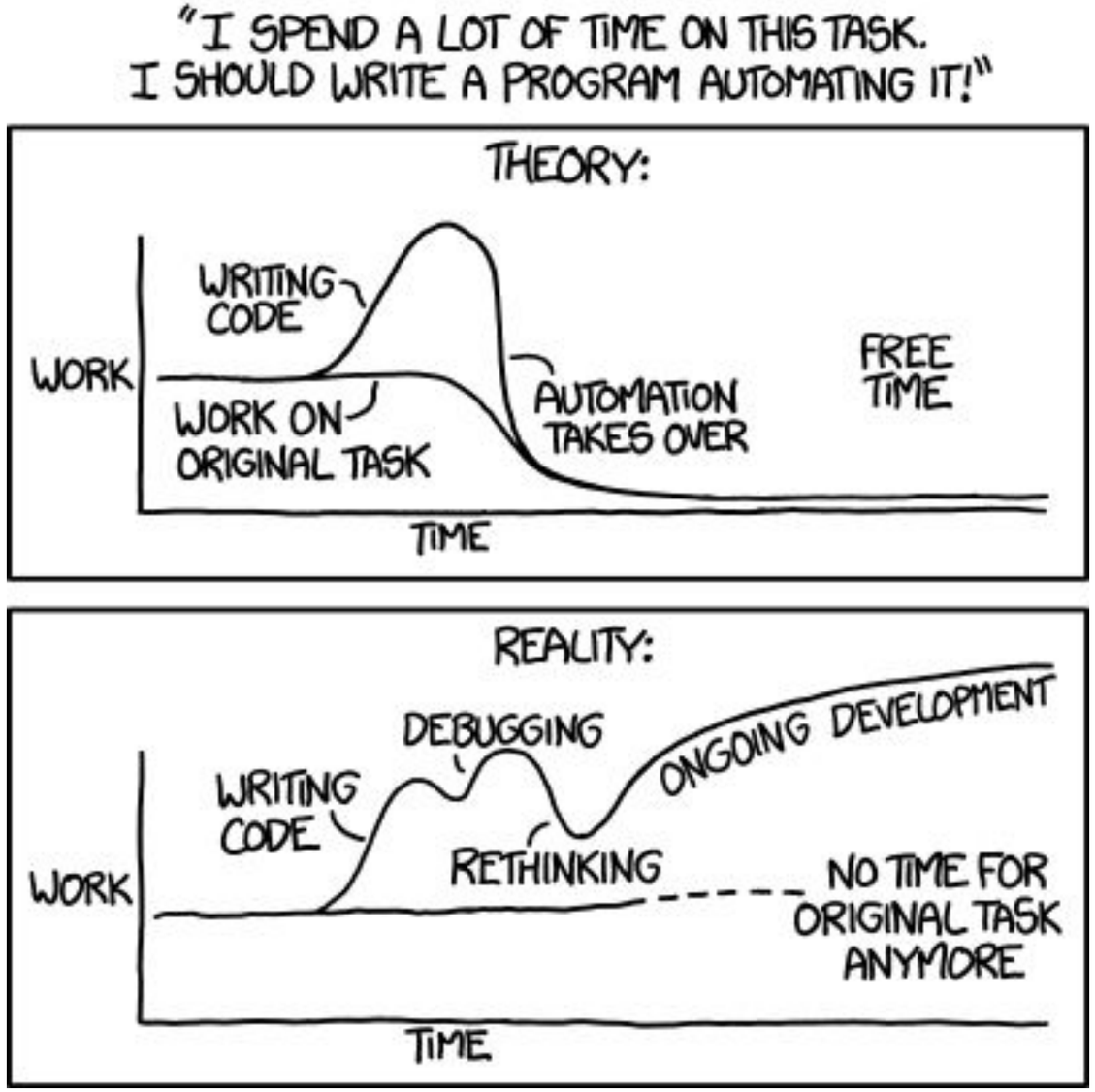
Ansible - miks seda kasutatakse?

Ansible-t kasutame põhiliselt selleks, et lihtsustada/kiirendada protsesse.

Ansible lubab meil OS/tarkvara kihi halduse ära automatiseerida.

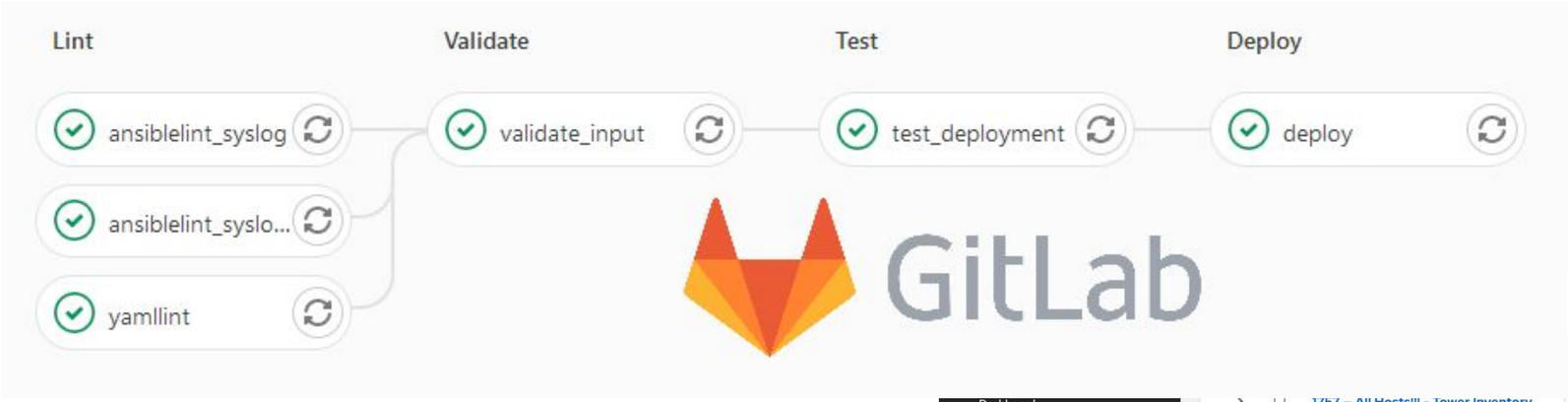


Sidenote

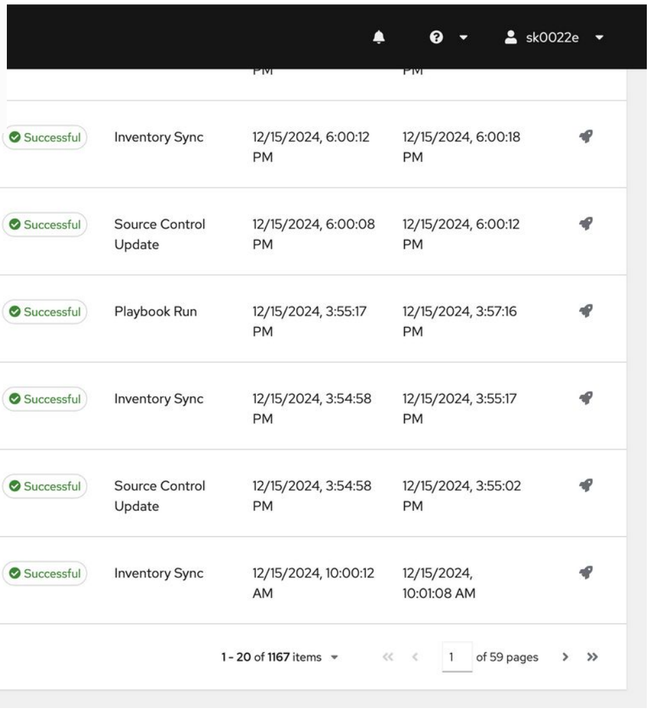

"I SPEND A LOT OF TIME ON THIS TASK.
I SHOULD WRITE A PROGRAM AUTOMATING IT!"
THEORY:
WORK
WRITING CODE
WORK ON ORIGINAL TASK
AUTOMATION TAKES OVER
FREE TIME
TIME
REALITY:
WORK
WRITING CODE
DEBUGGING
RETHINKING
ONGOING DEVELOPMENT
NO TIME FOR ORIGINAL TASK ANYMORE
TIME

Ansible - kuidas seda kasutatakse?

CI/CD



Tower/AWX



CLI

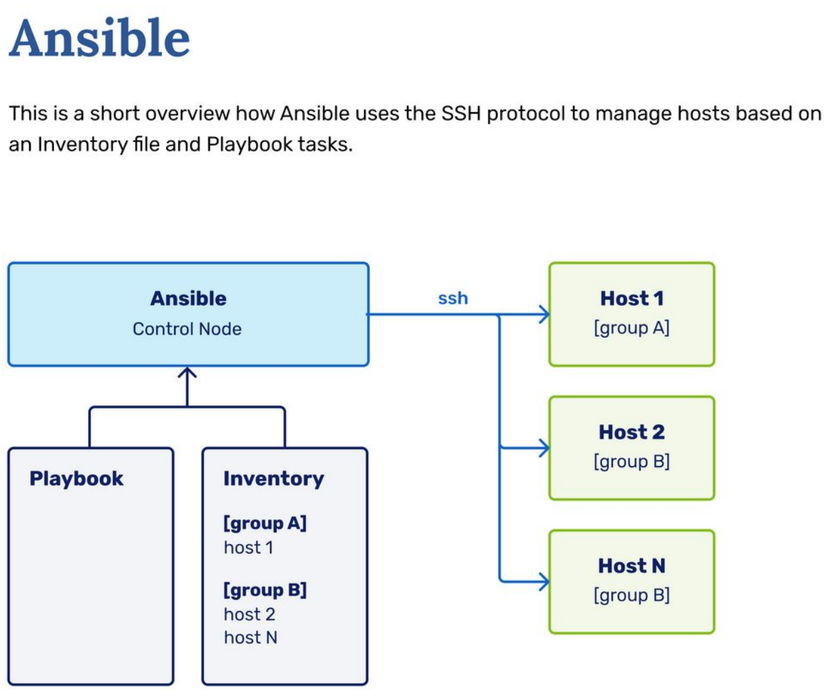
tags.yml

```
TASK [Make sure suricata_sniffing_interface is defined] ************************
ok: [delta-prism.hpc.ut.ee] => {
    "changed": false,
    "msg": "All assertions passed"
}
ok: [lossi-prism.hpc.ut.ee] => {
    "changed": false,
    "msg": "All assertions passed"
}

TASK [Install EPEL release and DNF plugins] ************************************
ok: [lossi-prism.hpc.ut.ee]
ok: [delta-prism.hpc.ut.ee]

TASK [Install Suricata] ********************************************************
ok: [delta-prism.hpc.ut.ee] => (item=suricata)
ok: [lossi-prism.hpc.ut.ee] => (item=suricata)
ok: [lossi-prism.hpc.ut.ee] => (item=ethtool)
ok: [delta-prism.hpc.ut.ee] => (item=ethtool)

TASK [Copy custom rules] *******************************************************
--- before: /etc/suricata/custom-rules/anomalies.rules
+++ after: /runner/project/templates/custom-rules/anomalies.rules
@@ -4,8 +4,8 @@
 # --- HTTP on non-standard port ---
 alert tcp any any -> any !80 (msg:"SURICATA CUSTOM ANOMALY HTTP on unusual port"; flow:to_server; app-layer-
protocol:http; threshold: type limit, track by_src, seconds 60, count 1; sid:9000001; rev:1;)


RUNNING HANDLER [restart suricata] *********************************************
changed: [lossi-prism.hpc.ut.ee]
changed: [delta-prism.hpc.ut.ee]

PLAY RECAP *********************************************************************
delta-prism.hpc.ut.ee      : ok=13   changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
lossi-prism.hpc.ut.ee      : ok=13   changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Ansible - arhitektuur

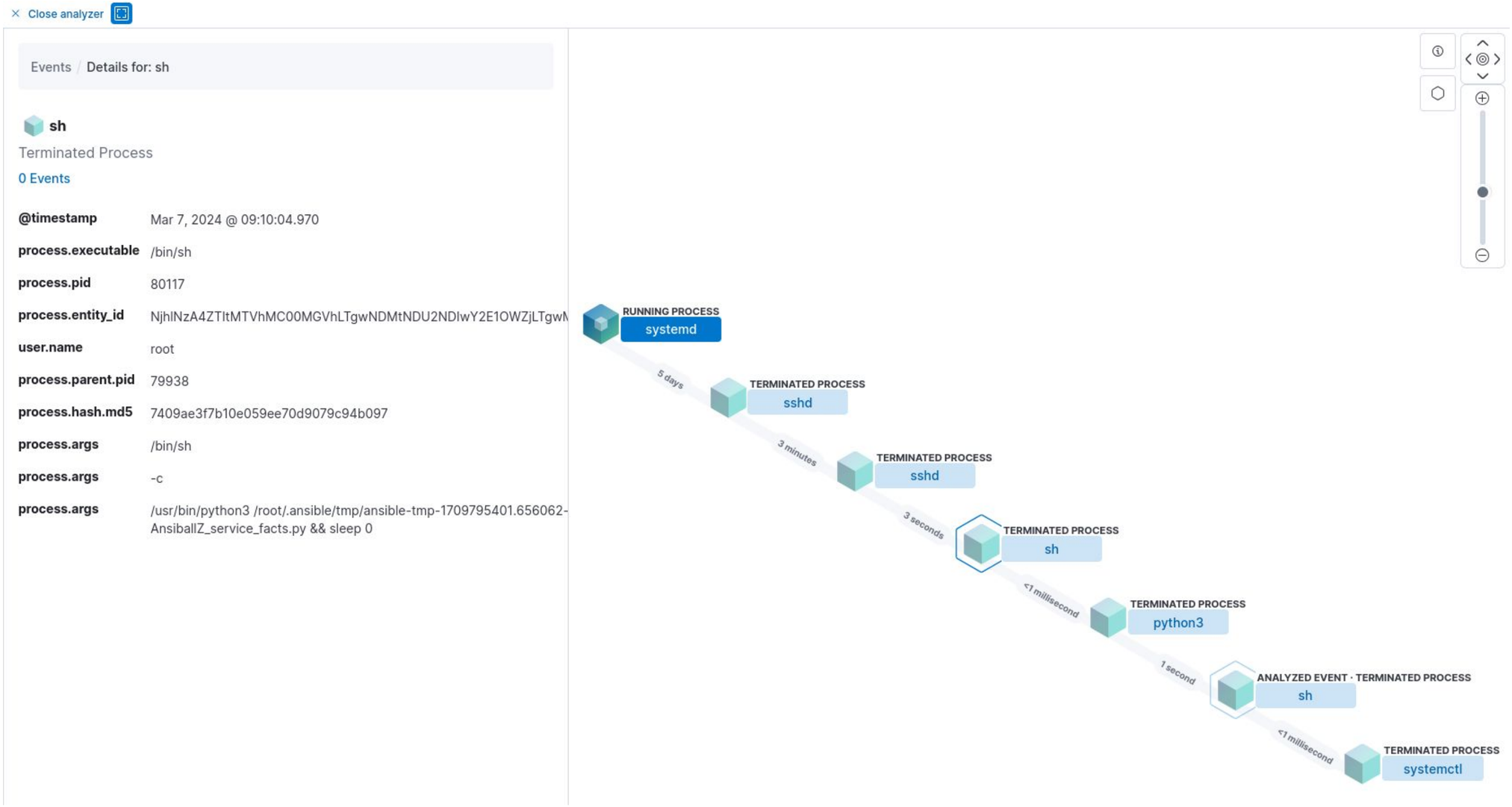
Ansible jooksutamiseks on vaja:

- *Controller/management node*
- Masinat mida hallata üritatakse (*managed node*)
- Installitud Ansiblet
- Käsku mida Ansible täitma peab



Üldiselt töötab Ansible üle SSH ühenduse kasutades Pythonit, aga on ka teisi variante.

Ansible - käskude käivitus



Ansible - omadused

Agentless

Erinevalt paljudest teistest sarnastest tehnoloogiatest, kasutab Ansible SSH-d ühendumiseks, samades õigustes mis operaator (inimene). See tähendab, et ei ole vaja klastrit, agenti, või muud veidrat usaldusprotsessi.

Python põhine

Ansible on kirjutatud Pythonis, ning üldiselt töötab nii, et kopeerib controller node-st Python scripti masinasse, ja paneb selle käima. Sellel on mitu head põhjust, aga ka paar tüütut tagajärge.

Kui ühendamiseks pole vaja SSH-d, siis saab ka ilma.

Push-based

Controller node alustab kõiki ühendusi, kaotades ära vajaduse tsentraalse serveri järele. Samuti on jooksutamise ajal võimalik aru saada kas, miks ja kunas midagi ei õnnestunud.

Käsk tähendab moodulit*

Iga käsk Ansible-le tõlgendatakse Pythoni scriptiks. Selline modulaarsus ja käskude erisus teeb Ansible funktsionaalsuse väga laiendatavaks, ja kasutab hästi ära Unix filosoofiat.

Ansible - moodulid

Ansible-l on suur kogus mooduleid vabalt kasutamiseks:

https://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html

Ning nende moodulite dokumentatsioon on IT kohta ebatavaliselt hea, koos näidetega.

Ansible - moodulite dokumentatsioon

Lisaks veebipõhisele dokumentatsioonile on ka üle käsurea lihtne leida dokumentatsiooni.

Loetleb kõik installeeritud moodulid:

ansible-doc -l

Näitab dokumentatsiooni kindla mooduli kohta:

ansible-doc <moodulinimi>

Ansible kasutab Pythonit

Kuna Ansible on vabavaraline ning kirjutatud Pythonis, on väga lihtne minna ja vaadata, kuidas mingi moodul on midagi kirjutatud tegema.

https://github.com/ansible/ansible/blob/devel/lib/ansible/modules/hostname.py

Ansible - ad-hoc

Lihtsaim meetod Ansible-t jooksutada on kasutada ad-hoc käskusid. Need on sarnased käsurea käskudele, aga kasutavad Ansiblet jooksutamiseks.

```
$ ansible -m shell -a "cat /etc/os-release" all

193.40.11.110 | CHANGED | rc=0 >>
PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"
NAME="Debian GNU/Linux"
VERSION_ID="12"
VERSION="12 (bookworm)"
VERSION_CODENAME=bookworm
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
```

Need on vahepeal kasulikud, et mitme masina peal korraga tegeleda.

Ansible - ad-hoc formaat

Aga ad-hoc on ideaalne meetod, et aru saada,
kuidas Ansible käsud töötavad:

ansible -m <moodul> -a “parameeter=argument” <grupp>

Näiteks kasutaja lisamine:

ansible -m user -a “name=sander groups=wheel” webservers

Ansible - inventar (inventory)

Selleks, et Ansible teaks mis seadmeid ta haldama peaks, on vajalik ka inventar (kutsutakse ka inventory või *hosts*).

Vaikimisi asukoht: **/etc/ansible/hosts**
Aga on võimalik, ja soovituslik, muuta.

Toetab kahte formaati - YAML või INI.

Seadmeid saab kirjeldada IP aadressi või DNS nimega.

Ansible - inventari näited

INI

```
[proxy]
website.example.com

[webservers]
192.168.101.[100:200]

[webservers:vars]
ntp_server=ntp.example.com

[databases]
192.168.100.15 role="master"
192.168.100.16 role="replica"
```

YAML

```
all:
  children:
    proxy:
      hosts:
        website.example.com:

    webservers:
      hosts:
        192.168.101.[100:200]:
      vars:
        ntp_server: ntp.example.com

    databases:
      hosts:
        192.168.100.15:
          role: "master"
        192.168.100.16:
          role: "replica"
```

01

Praktiline osa

Praktiline osa 1 - “Algus ja ad-hoc”

01 Masinate ligipääs

02 Ansible paigaldamine

03 Esimesed Ansible käsud

Kui valmis, võib aega pausi jaoks ära kasutada

Ansible projekt ja “playbook”

02

Ansible - ad-hoc

Ad-hoc käsud on harva kasulikud, näiteks kui on vaja suurt kogust masinaid restartida.

```
$ ansible -m reboot webservers

193.40.11.124 | CHANGED => {
    "changed": true,
    "elapsed": 20,
    "rebooted": true
}
```

Tavaliselt kasutatakse playbook-e, mis on grupeering erinevatest käskudest.

```yaml
- hosts: webservers
  user: ansible
  become: true
  become_user: root

  tasks:
    - name: Lisa kasutaja
      user:
        name: user
        state: present
        groups: wheel

    - name: Paigalda nginx
      package:
        name: nginx
        state: latest
```

Ansible - playbook

Playbook on YAML-is kirjutatud kavand (ingl. k. blueprint) mis kirjeldab, mis tegevusi jooksutada milliste seadmete pihta.

Playbookis on kirjas kõik info Ansible käivitusest - mis moodulid, mis konfiguratsioonis, mis seadmete kohta käivad.

Ansible - projekt

Playbook käib käsikäes muude failidega, näiteks inventar. Aga võib olla veel faile.

Näiteks konfifaili masinasse lisades peab see kuskil Ansible jaoks kättesaadav olema *management* seadme peal.

```
production                # inventory file for production servers
staging                   # inventory file for staging environment

files/
templates/

group_vars/
   group1.yml             # here we assign variables to particular groups
   group2.yml
host_vars/
   hostname1.yml          # here we assign variables to particular systems
   hostname2.yml

library/                  # if any custom modules, put them here (optional)
module_utils/             # if any custom module_utils to support modules, put them here (optional)
filter_plugins/           # if any custom filter plugins, put them here (optional)

site.yml                  # master playbook
webservers.yml            # playbook for webserver tier
dbservers.yml             # playbook for dbserver tier

roles/
    common/               # this hierarchy represents a "role"
        tasks/            #
            main.yml      #  <-- tasks file can include smaller files if warranted
        handlers/         #
            main.yml      #  <-- handlers file
        templates/        #  <-- files for use with the template resource
            ntp.conf.j2   #  <------- templates end in .j2
        files/            #
            bar.txt       #  <-- files for use with the copy resource
            foo.sh        #  <-- script files for use with the script resource
        vars/             #
            main.yml      #  <-- variables associated with this role
        defaults/         #
            main.yml      #  <-- default lower priority variables for this role
        meta/             #
            main.yml      #  <-- role dependencies
        library/          # roles can also include custom modules
        module_utils/     # roles can also include custom module_utils
        lookup_plugins/   # or other types of plugins, like lookup in this case

    webtier/              # same kind of structure as "common" was above, done for the webtier role
    monitoring/           # ""
    fooapp/               # ""
```

Ansible - projektipõhine inventar

Kuigi teatud juhtudel kasutab Ansible süsteemset inventari, on, vähemalt alguses, soovituslik kasutada igas projektis enda oma.

Inventar kipub kasvama päris suureks ja keeruliseks. Hoides ainult projektiga seotud masinad inventaris, vähendab see segadust märkimisväärselt.

Ansible - playbook-i jooksutamine

Kui playbook on valmis, võib selle käima panna.

Playbooke käivitatakse üle käsurea:

ansible-playbook <playbook> -i hosts -k

```
PLAY [all]
*********************************************************************

TASK [Gathering Facts]
*********************************************************************
ok: [193.40.11.124]

TASK [Lisa kasutaja]
*********************************************************************
changed: [193.40.11.124]

TASK [Paigalda nginx]
*********************************************************************
ok: [193.40.11.124]

PLAY RECAP
*********************************************************************

193.40.11.124 : ok=3    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

Ansible - playbook-i käskude staatused

Staatus	Põhjendus	Tagajärg
ok	Käsk läks ilusti läbi, muuta ei olnud vaja midagi.	Playbook jätkab tööd.
changed	Käsk läks läbi, süsteemis tehti muudatusi.	Tehti muudatusi, playbook jätkab tööd.
unreachable	Ei saadud seadmele ligi, et käsku jooksutada. Tavaliselt juhtub esimese asjana.	Ei saadud seadmele ligi, playbooki ei pandud käima.
failed	Käsku üritati teha, aga midagi ebaõnnestus.	Käsk ebaõnnestus, enamasti lõpetab playbook töö.
skipped	Mingi tingimislause alusel käsk jäeti selle seadme peal vahele.	Playbook jätkab tööd.

Ansible - idempotentsus

```
PLAY [all] ********************************************************************

TASK [Gathering Facts] ********************************************************
ok: [193.40.11.124]

TASK [Lisa kasutaja] **********************************************************
changed: [193.40.11.124]

TASK [Paigalda nginx] *********************************************************
changed: [193.40.11.124]

PLAY RECAP ********************************************************************
193.40.11.124              : ok=3    changed=2
kipped=0    rescued=0    ignored=0
```

Esimene käivitus

Teine käivitus

```
PLAY [all] ********************************************************************

TASK [Gathering Facts] ********************************************************
ok: [193.40.11.124]

TASK [Lisa kasutaja] **********************************************************
ok: [193.40.11.124]

TASK [Paigalda nginx] *********************************************************
ok: [193.40.11.124]

PLAY RECAP ********************************************************************
193.40.11.124              : ok=3    changed=0    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

Ansible - projekti failid

```
production                # inventory file for production servers
staging                   # inventory file for staging environment

files/
templates/

group_vars/
   group1.yml             # here we assign variables to particular groups
   group2.yml
host_vars/
   hostname1.yml          # here we assign variables to particular systems
   hostname2.yml

library/                  # if any custom modules, put them here (optional)
module_utils/             # if any custom module_utils to support modules, put them here (optional)
filter_plugins/           # if any custom filter plugins, put them here (optional)

site.yml                  # master playbook
webservers.yml            # playbook for webserver tier
dbservers.yml             # playbook for dbserver tier

roles/
    common/               # this hierarchy represents a "role"
        tasks/            #
            main.yml      #  <-- tasks file can include smaller files if warranted
        handlers/         #
            main.yml      #  <-- handlers file
        templates/        #  <-- files for use with the template resource
            ntp.conf.j2   #  <------- templates end in .j2
        files/            #
            bar.txt       #  <-- files for use with the copy resource
            foo.sh        #  <-- script files for use with the script resource
        vars/             #
            main.yml      #  <-- variables associated with this role
        defaults/         #
            main.yml      #  <-- default lower priority variables for this role
        meta/             #
            main.yml      #  <-- role dependencies
        library/          # roles can also include custom modules
        module_utils/     # roles can also include custom module_utils
        lookup_plugins/   # or other types of plugins, like lookup in this case

    webtier/              # same kind of structure as "common" was above, done for the webtier role
    monitoring/           # ""
    fooapp/               # ""
```

Paljudel kaustadel Ansible projektis on mõte mingi mooduli pärast.

Näiteks **files** kaust on selleks, et hoida projekti ja selle masinatega seotud artefakte.

Ansible moodul, mis tegeleb failide liigutamisega, on tavaliselt **copy**.

Ansible - “copy” moodul

Copy moodul on selleks, et kopeerida Ansible controller masinast fail hallatavasse seadmesse.

Faile mida kopeerida otsitakse kõigepealt projekti **files** kaustast, kui ei ole tervet teekonda antud.

```
- name: Kopeeri fail
  copy:
    src: fail.txt
    dest: /opt/kaust/fail.txt
    force: true
```

Ansible - “copy” moodul

Copy moodulile võib anda ka "content" argumendi, kui tahta vältida faili kasutamist. Siis pannakse content otse faili.

```
- name: Kopeeri fail
  copy:
    content: "Lorem ipsum."
    dest: /opt/kaust/fail.txt
    force: true
```

02

Praktiline osa

Praktiline osa 2 - Playbook

04 Projekti struktuuri loomine

05 Playbooki ehitamine

06 Esimese playbooki jooksutamine

07 Masinasse veebilehe üles seadmine

Kui valmis, võib aega pausi jaoks ära kasutada

Ansible muutujad ja tingimused

03

Ansible - muutujad

Siiamaani on meie playbookid olnud väga staatilised - kõik väärtused on otse playbooki sisse kirjutatud.

Teatud juhtudel on kasulik, kui me saaks hoopis kasutada muutujaid kas playbookide sees, või isegi failide sees, mida playbook haldab.

See on kasulik süsteemide erinevuste haldamiseks. Lihtne näide domeeninimi, nginx kaust.

```yaml
- hosts: webservers
  user: ansible
  become: true
  become_user: root

  tasks:
    - name: Lisa kasutaja
      user:
        name: user
        state: present
        groups: wheel

    - name: Paigalda nginx
      package:
        name: nginx
        state: latest
```

Ansible - muutujad

Muutujad Ansible-s on samaväärsed muutujatega teistes programmeerimiskeeltes.

Nende kasutuseks tuleb kõigepealt need kuidagi kuhugi defineerida.

Seejärel saab neid kasutada kas playbooki sees, failide sees mida Ansible haldab, või isegi teiste muutujate sees.

Ansible - muutujate defineerimine

Muutujate defineerimise süntaks on võrdlemisi lihtne:

Inventari failis (INI):

muutuja=väärtus

Muutujate failides:

muutuja: väärtus

Ansible - muutujate defineerimine

Muutujat saab defineerida ka mooduliga, playbooki ajal:

```
- name: Defineeri muutuja
  set_fact:
    muutuja: "väärtus"
```

Seda saab järgnevates käskudes ilusti kasutada, nagu ka teisi muutujaid.

Ansible - muutujate defineerimise kohad

Koht	Failiformaat	Eesmärk
inventarifail	INI või YAML	Sobilik masina põhiste muutujate jaoks, mis on tugevalt seotud masina käitumisega.
host_vars/<hostinimi>.yml	YAML	Sobilik üleüldisteks masina põhisteks muutujateks.
group_vars/<grupinimi>.yml	YAML	Üleüldised muutujad mida anda tervele seadmete grupile.
set_facts moodul	YAML	Muutujad mis pannakse dünaamiliselt paika playbooki jooksul.
rolli vars/main.yml	YAML	Muutujad mis on ühe rolli kohta, rolliülesed.
rolli defaults/main.yml	YAML	Vaikimisi väärtuste andmiseks rolli sees muutujatele.

Ansible - muutujate definitsiooni järjekord

https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html#understanding-variable-precedence

Ansible - muutujate kasutamine

Muutujat saab kasutada erinevalt. Üks variantidest on otse playbooki käsu sees:

```yaml
- hosts: all
  user: root
  become: true
  become_user: root

  vars:
    kasutaja: "sander"

  tasks:
    - name: Tee kasutaja
      user:
        name: "{{ kasutaja }}"
```

Ansible - “template” moodul

Teine variant on **template** moodul, mis töötab sarnaselt **copy** mooduliga, aga lubab ka panna muutujaid faili sisse, mis siis ära renderdatakse.

Faile mida kopeerida otsitakse kõigepealt projekti **template** kaustast, kui ei ole tervet teekonda antud.

```
$ cat templates/templiit.txt.j2

Muutuja väärtus: {{ kasutaja }}

---

- name: Renderda fail
  template:
    src: templiit.txt.j2
    dest: /tmp/example.txt

---

$ ansible -m shell -a "cat /tmp/example.txt" all

193.40.11.124 | CHANGED | rc=0 >>
Muutuja väärtus: sander
```

Ansible - “copy” moodul

Copy moodulile võib anda ka "content" argumendi, siis pannakse content otse faili.

See lubab ka muutujate kasutamist.

```
- name: Renderda fail
  copy:
    content: "Muutuja väärtus: {{ kasutaja }}"
    dest: /tmp/example.txt
```

Ansible - tingimuslaused

Ansible toetab ka tinglikult käskude jooksutamist muutujate alusel.

Käsk käivitatakse ainult juhul, kui **when** tingimus on tõene.

Vastasel juhul jäetakse vahele sinise **SKIPPED** staatusega.

```yaml
- hosts: all
  user: root
  become: true
  become_user: root

  vars:
    roll: "veebiserver"

  tasks:
    - name: Renderda fail
      package:
        name: nginx
        state: present
      when: roll == "veebiserver"

    - name: Renderda fail
      package:
        name: postgresql
        state: present
      when: roll == "andmebaas"
```

Ansible - faktid

Tingimuslaused on eriti kasulikud koos faktidega.

Faktid on info, mida Ansible kogub automaatselt masinate kohta, kohe kui playbook masina peal käima läheb.

Seda saab ka ise välja kutsuda, kasutades **setup** moodulit:

ansible -m setup all

Ansible - faktide väljund

Faktides on väga palju kasulikku infot, näiteks:

- Masina Linux distributioon
- Masina operatsioonisüsteem
- Arhitektuur
- Võrguaadressid

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

Ansible - faktide väljund

Kui mitu masinat jooksevad sama playbooki all, pääseb ühe masina Ansible kontekstist ka teise masina muutujatele ligi.

```
- hosts: all
  user: root
  tasks:
    - name: Print Ubuntu machine info
      debug:
        msg: "Mina: {{ ansible_distribution }}
Teine: {{ hostvars['193.40.11.110']
['ansible_distribution'] }}"
      when: ansible_distribution == "CentOS"

---

ok: [193.40.11.124] => {
    "msg": "Mina: CentOS  Teine: Debian"
}
```

Ansible - maagilised muutujad

On terve hulk muutujaid, mida Ansible kasutab, et näidata enda sisemist seisu. Neid nimesid üldiselt oma muutujate nimedeks ei saa kasutada.

```
- hosts: all
  user: root
  tasks:
    - name: Groups muutuja
      debug:
        msg: "{{ groups }}"

---

ok: [193.40.11.110] => {
    "msg": {
        "all": [
            "193.40.11.124",
            "193.40.11.110"
        ],
        "centos": [
            "193.40.11.124"
        ],
        "ubuntu": [
            "193.40.11.110"
        ],
        "ungrouped": []
    }
}
```

Ansible - maagilised muutujad

Õnneks sellised muutujad enamasti algavad sõnadega **ansible_**

Aga mitte alati

https://docs.ansible.com/ansible/latest/reference_appendices/special_variables.html

Ansible - käitumuslikud muutujad

On ka olemas terve hulk käitumuslikke muutujaid, millega saab kontrollida kuidas Ansible käitub.

Näiteks **ansible_password** muutujaga saab ette anda parooli, ilma et peaks seda iga kord küsima.

```
[centos]
193.40.11.124 ansible_password="parool"
```

https://docs.ansible.com/ansible/latest/inventory_guide/intro_inventory.html#connecting-to-hosts-behavioral-inventory-parameters

03

Praktiline osa

Praktiline osa 3 - muutujad

08 Maagiliste muutujate kasutamine

09 Ise muutuja defineerimine ja kasutamine

10 Setup moodul

11 Tingimuslaused muutujate alusel

Kui valmis, võib aega pausi jaoks ära kasutada

Ansible - boonus - “register”

Ansible-l on olemas ka **register** argument, mis sobib iga käsu järgi.

See salvestab käsu väljundi ja muu info muutujasse.

Mida salvestatakse, saab vaadata mooduli dokumentatsioonist.

```
- hosts: controller
  user: root

  tasks:
    - name: Lisa kasutaja
      user:
        name: user
        state: present
        groups: wheel
      register: output

    - debug: var=output

--

ok: [ansible.hashify.ee] => {
    "output": {
        "changed": true,
        "comment": "",
        "create_home": true,
        "failed": false,
        "group": 1047,
        "groups": "wheel",
        "home": "/home/user",
        "name": "user",
        "shell": "/bin/bash",
        "state": "present",
        "system": false,
        "uid": 1047
    }
}
```

Ansible rollid, impordid, handlerid

04

Ansible - handlers

Meie siiamaani ehitatud playbook rikub idempotentsuse reeglit.

Igal Ansible käivitusel restarditakse nginx-i, olenemata kas midagi muutus.

Ansible - handlers

```yaml
- hosts: all
  user: root

  tasks:
  - name: Add SSHD config
    copy:
      content: "PasswordAuthentication yes"
      dest: /etc/ssh/sshd_config.d/10-password.conf
    notify: restart sshd

  handlers:
  - name: restart sshd
    service:
      name: sshd
      state: restarted
```

Tegevusteks, mida tuleb teha ainult korra, ja ainult kui miski muutus muudes task-ides, on olemas handlerid.

Ansible - rollid

Playbookid üksinda lubavad kõik asjad ära lahendada, aga niimoodi need kasvavad päris pikaks.

Samuti uue playbooki tegemiseks, mis täidab sarnaseid ülesandeid, tähendab ühekaupa moodulite üle kopeerimist.

DRY?

Ansible - rollid

Selle probleemi lahendamiseks on meil rollid - eraldatud tükid Ansible-t, mis on mõeldud lahendama kindlat probleemi

```
roles/
    nginx/               # this hierarchy represents a "role"
        tasks/            #
            main.yml      #  <-- tasks file can include smaller files if warranted
        handlers/         #
            main.yml      #  <-- handlers file
        templates/        #  <-- files for use with the template resource
            ntp.conf.j2   #  <------- templates end in .j2
        files/            #
            bar.txt       #  <-- files for use with the copy resource
            foo.sh        #  <-- script files for use with the script resource
        vars/             #
            main.yml      #  <-- variables associated with this role
        defaults/         #
            main.yml      #  <-- default lower priority variables for this role
        meta/             #
            main.yml      #  <-- role dependencies
        library/          # roles can also include custom modules
        module_utils/     # roles can also include custom module_utils
        lookup_plugins/   # or other types of plugins, like lookup in this case

    webtier/              # same kind of structure as "common" was above, done for the webtier role
    monitoring/           # ""
    fooapp/               # ""
```

Kuna nad on eraldatud kaustas, on neid ka väga lihtne jagada.

Ansible - rollid - tasks

Iga erinev osa rollist saab alguse **main.yml** failist.

Näiteks **tasks**, mis oli enne eraldi blokk playbookis, on nüüd kirjas tasks/main.yml failis.

```
roles/
    nginx/               # this hierarchy represents a "role"
        tasks/            #
            main.yml      #  <-- tasks file can include smaller files if warranted
        handlers/         #
            main.yml      #  <-- handlers file
        templates/        #  <-- files for use with the template resource
            ntp.conf.j2   #  <------- templates end in .j2
        files/            #
            bar.txt       #  <-- files for use with the copy resource
            foo.sh        #  <-- script files for use with the script resource
        vars/             #
            main.yml      #  <-- variables associated with this role
        defaults/         #
            main.yml      #  <-- default lower priority variables for this role
        meta/             #
            main.yml      #  <-- role dependencies
        library/          # roles can also include custom modules
        module_utils/     # roles can also include custom module_utils
        lookup_plugins/   # or other types of plugins, like lookup in this case

    webtier/              # same kind of structure as "common" was above, done for the webtier role
    monitoring/           # ""
    fooapp/               # ""
```

Ansible - rollid - muutujate defineerimise kohad

Koht	Failiformaat	Eesmärk
inventarifail	INI või YAML	Sobilik masina põhiste muutujate jaoks, mis on tugevalt seotud masina käitumisega.
host_vars/<hostinimi>.yml	YAML	Sobilik üleüldisteks masina põhisteks muutujateks.
group_vars/<grupinimi>.yml	YAML	Üleüldised muutujad mida anda tervele seadmete grupile.
set_facts moodul	YAML	Muutujad mis pannakse dünaamiliselt paika playbooki jooksul.
rolli vars/main.yml	YAML	Muutujad mis on ühe rolli kohta, rolliülesed.
rolli defaults/main.yml	YAML	Vaikimisi väärtuste andmiseks rolli sees muutujatele.

Ansible - rollid - kasutamine

```
$ ls -d roles/nginx/

roles/nginx/
---
- hosts: all
  user: root

  roles:
    - nginx
```

Ansible - rollid - muutujate propagatsioon

```
$ cat hosts

[centos]
193.40.11.124 var_inventory="olemas"
---
$ cat group_vars/all.yml

var_group_vars: "olemas"
---
$ cat vars.yml

- hosts: all
  user: root

  vars:
    var_playbook: "olemas"

  roles:
    - variables
---
$ cat roles/variables/vars/main.yml

var_roll: "olemas"
---
ok: [193.40.11.124] => (item=var_inventory - olemas) => {
    "msg": "var_inventory - olemas"
}
ok: [193.40.11.124] => (item=var_group_vars - olemas) => {
    "msg": "var_group_vars - olemas"
}
ok: [193.40.11.124] => (item=var_playbook - olemas) => {
    "msg": "var_playbook - olemas"
}
ok: [193.40.11.124] => (item=var_roll - olemas) => {
    "msg": "var_roll - olemas"
}
```

04

Praktiline osa

Praktiline 4 - handlerid ja rollid

12 Kasutame handlereid

13 Teeme Nginx osast rolli

Kui valmis, võib aega pausi jaoks ära kasutada

Ansible silumine (debug)

05

Ansible - silumine

Nagu IT-s ikka, asjad ei kipu minema meiemoodi. Tuleb tegeleda silumisega (debug), et leida vigu ja lahendada probleeme.

Ansible-l on erinevaid variante kuidas seda saavutada, kolm esimest on:

- –syntax-check
- –diff
- –check

Ansible - silumine - syntax check

```
$ ansible-playbook --syntax-check changes.yml

playbook: changes.yml
$ sed -i 's/tasks/taskss/g' changes.yml
$ ansible-playbook --syntax-check changes.yml
[ERROR]: 'taskss' is not a valid attribute for a Play
Origin: /home/sk0022e/Work/Trainings/ansible2025/exercises/ex5/changes.yml:4:3

2   user: root
3
4   taskss:
    ^ column 3
```

Syntax check on lihtne kontroll, et kogu playbook süntaksi poolest sobib.

Kui on viga, antakse teada veateatega.

Sarnane asi toimub playbooki käima pannes.

Ansible - silumine - check

Check lippu kasutades saab kontrollida, kas seda playbooki jooksutades miski muutuks.

Töötab enamuste* moodulitega.

Käitub nagu “dry run” - mooduleid päriselt ei käivitata.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/user_module.html#attributes

```
ansible-playbook changes.yml --limit ansible.hashify.ee -i hosts --check

PLAY [all] *********************************************************
*********************************************************

TASK [Make file change] ********************************************
*********************************************************
changed: [ansible.hashify.ee]

TASK [Change permissions] ******************************************
*********************************************************
ok: [ansible.hashify.ee]

PLAY RECAP *********************************************************
*********************************************************
ansible.hashify.ee         : ok=2    changed=1    unreachable=0
    failed=0    skipped=0    rescued=0    ignored=0
```

Ansible - silumine - diff

Diff võtit kasutades playbook jookseb normaalselt, aga faili muudatuste korral näidatakse rohkem infot selle kohta, mis muutus.

NB! –check ja –diff saab koos kasutada

```
ansible-playbook changes.yml --limit ansible.hashify.ee -i hosts --diff

PLAY [all] *********************************************************
*****************

TASK [Make file change] ********************************************
*****************
--- before: /tmp/test
+++ after: /tmp/test
@@ -1 +1 @@
-Lammas
\ No newline at end of file
+Kala
\ No newline at end of file

changed: [ansible.hashify.ee]

TASK [Change permissions] ******************************************
*****************
--- before
+++ after
@@ -1,6 +1,6 @@
 {
-    "group": 989,
-    "mode": "0600",
-    "owner": 992,
+    "group": 0,
+    "mode": "0700",
+    "owner": 0,
     "path": "/tmp/test"
 }

changed: [ansible.hashify.ee]
```

Ansible selektiivne käivitamine

06

Ansible - selektiivne käivitamine

Testides, või piisavalt suure playbooki / piisavalt paljude seadmetega tekib olukordi, kus ei taha kogu playbooki kõigi masinate pihta jooksutada.

Selleks on Ansible-l featuurid sildid (tags) ja limiidid (limits).

Ansible - selektiivne käivitamine - sildid

tags.yml

```yaml
- hosts: centos
  user: root

  tasks:
    - name: task1
      debug: msg="Task tagged 'one'"
      tags: one

    - name: task2
      debug: msg="Task tagged 'two'"
      tags: two

    - name: task3
      debug: msg="Task tagged 'always'"
      tags: always
```

Siltidega saab ära sildistada erinevad taskid, ja siis hiljem neid kasutada ainult nende taskide jooksutamiseks.

tags.yml

```
$ ansible-playbook tags.yml --tags one

PLAY [centos] ************************************

TASK [task1] *************************************
ok: [ansible.hashify.ee] => {
    "msg": "Task tagged 'one'"
}

TASK [task3] *************************************
ok: [ansible.hashify.ee] => {
    "msg": "Task tagged 'always'"
}
```

Ansible - selektiivne käivitamine - limiidid

Limiidid lubavad kontrollida, millise seadme pihta playbook / ad-hoc task jookseb.

Limiidid on kas grupi või seadme *inventory name* põhised.

```
$ ansible-playbook limits.yml

PLAY [My play targeting all hosts] *********************************************************************

TASK [debug] *******************************************************************************************
ok: [ansible.hashify.ee] => {
    "msg": "Running on all hosts"
}
ok: [dbserver-01] => {
    "msg": "Running on all hosts"
}
ok: [webserver-01] => {
    "msg": "Running on all hosts"
}

PLAY [Targeting multiple groups called dbservers and webservers] ***************************************

TASK [debug] *******************************************************************************************
ok: [dbserver-01] => {
    "msg": "Running on all dbservers and webserver"
}
ok: [webserver-01] => {
    "msg": "Running on all dbservers and webserver"
}

PLAY [Targeting only one host called dbserver-01] ******************************************************

TASK [debug] *******************************************************************************************
ok: [dbserver-01] => {
    "msg": "Running on one hosts"
}

PLAY [Running on the Ansible server] *******************************************************************

TASK [debug] *******************************************************************************************
ok: [localhost] => {
    "msg": "Running on localhost"
}

PLAY RECAP *********************************************************************************************
ansible.hashify.ee         : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
dbserver-01                : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
localhost                  : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
webserver-01               : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Ansible - selektiivne käivitamine - limiidid

Ansible - selektiivne käivitamine - limiidid

This table lists common patterns for targeting inventory hosts and groups.

Description	Pattern(s)	Targets
All hosts	all (or *)	
One host	host1	
Multiple hosts	host1:host2 (or host1,host2)	
One group	webservers	
Multiple groups	webservers:dbservers	all hosts in webservers plus all hosts in dbservers
Excluding groups	webservers:!atlanta	all hosts in webservers except those in atlanta
Intersection of groups	webservers:&staging	any hosts in webservers that are also in staging

https://docs.ansible.com/projects/ansible/latest/inventory_guide/intro_patterns.html#common-patterns

Praktiline 5 + 6 - silumine, selektiivne käivitamine

12 Check mode

13 Diff mode

14 Syntax check

15 Tags

16 Limits

Kui valmis, võib aega pausi jaoks ära kasutada

Ansible Galaxy

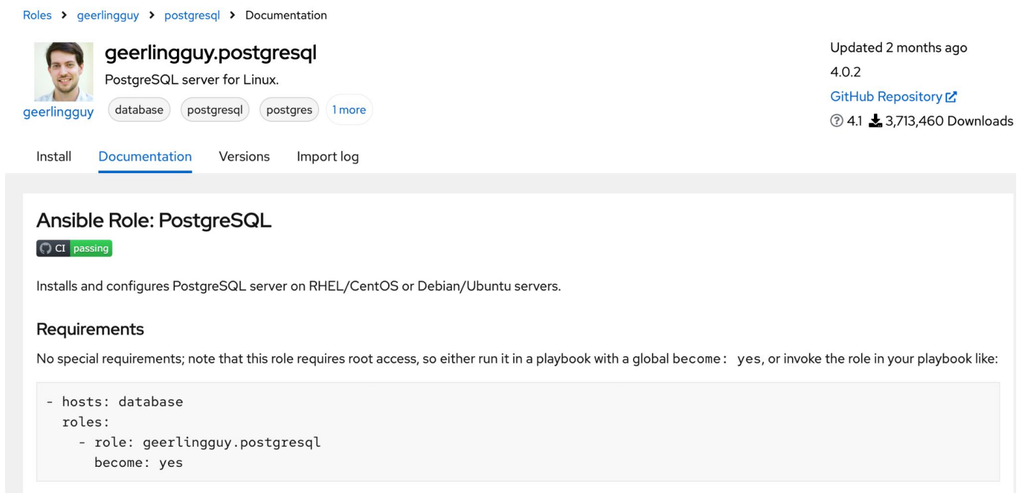
07

Ansible Galaxy - rollid ja kollektsioonid

Siiamaani oleme kasutanud ainult Ansible sisse ehitatud mooduleid (nn "Builtin").

Tänapäevane Ansible on aga modulaarne – enamik funktsionaalsust (AWS, Azure, Kubernetes, isegi osa Linuxi haldusest) asub eraldiseisvates kollektsioonides (Collections).

Samuti ei ole mõtet "jalgratast leiutada". Kui on vaja seadistada keeruline teenus (nt PostgreSQL klastris), on tõenäoline, et keegi on selleks juba kvaliteetse rolli kirjutanud.



Ansible - Galaxy

Selleks, et rolle ja kollektsioone hõlpsasti kasutada, on olemas teenus nimega Ansible Galaxy.

https://galaxy.ansible.com/ui/

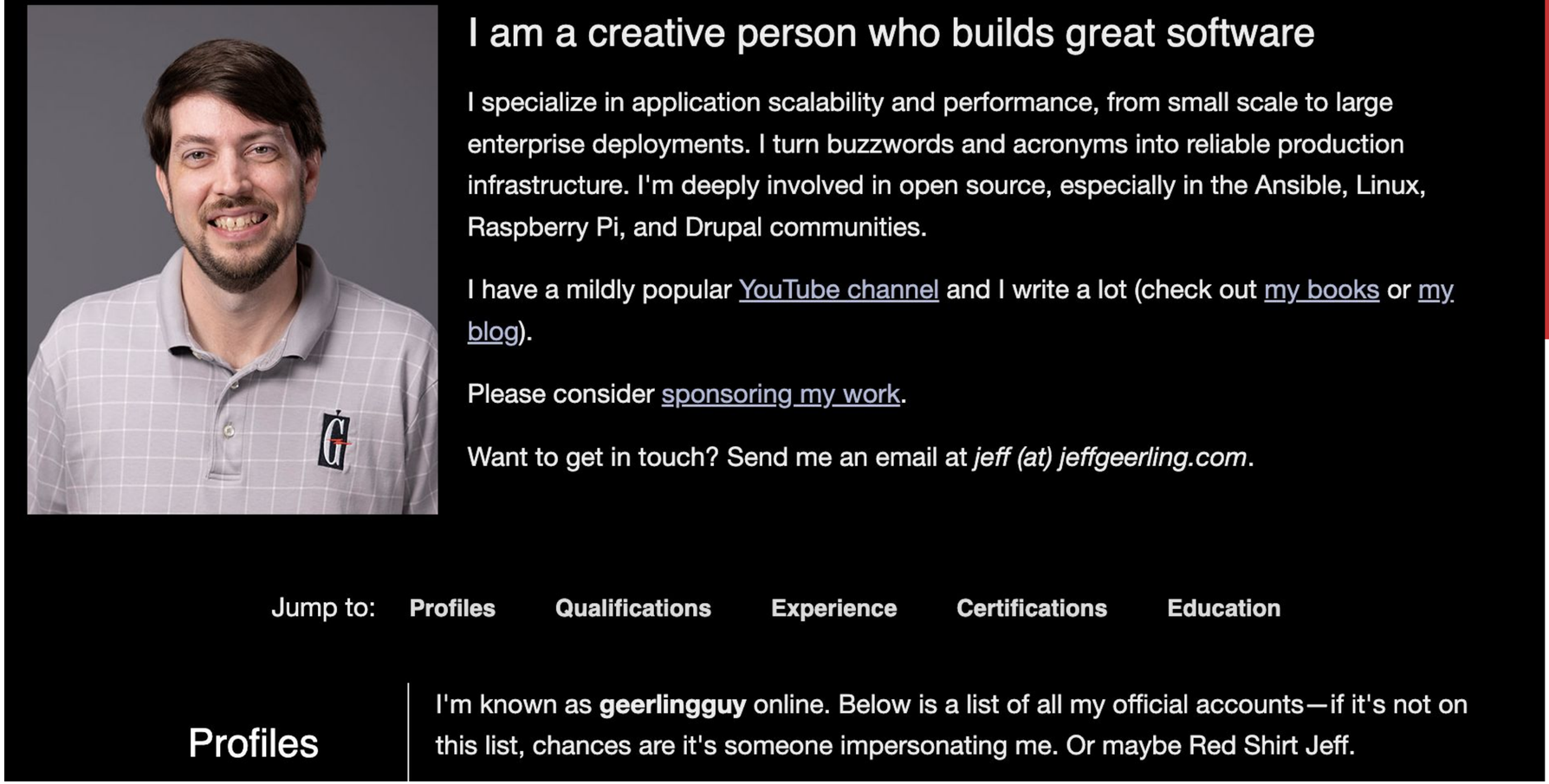
See on repositoorium kõigile Ansible rollidele ja kollektsioonidele, tehes otsingu väga lihtsaks.

Sidenote - Geerlingguy

See mees on Ansible kommuunis legendi staatuses.

Ta on enda peale võtnud mõndade rollide kirjutamise, ja tema rollid on tihti kõige paremad kindlat asja tegevad.

Kui näete tema tehtud rolli, siis seda võib usaldada.



https://www.jeffgeerling.com/about

Ansible - rollid (collections)

Kasutades **ansible-galaxy** käsureatööriista, saab installida Galaxy-st rolle.

Need rollid installitakse, vaikimisi, ~/.ansible/roles kausta.

Kui see on installitud, saab oma playbookides seda väga lihtsasti kasutada.

tags.yml

```
[training-17@ansible ~]$ ansible-galaxy role install geerlingguy.java
Starting galaxy role install process
- downloading role 'java', owned by geerlingguy
- downloading role from https://github.com/geerlingguy/ansible-role-
java/archive/2.5.2.tar.gz
- extracting geerlingguy.java to /home/training-
17/.ansible/roles/geerlingguy.java
- geerlingguy.java (2.5.2) was installed successfully

[training-17@ansible ~]$ ansible-galaxy role list
# /home/training-17/.ansible/roles
- geerlingguy.nginx, 3.3.0
- geerlingguy.java, 2.5.2

[training-17@ansible ~]$ ls -h .ansible/roles
geerlingguy.java  geerlingguy.nginx
```

tags.yml

```
- hosts: centos,debian
  user: root
  roles:
    - role: geerlingguy.java
      become: yes
```

Ansible - kollektsioonid

```
- name: Ensure an XFS filesystem on data volume (default /dev/sdb)
  community.general.filesystem:
    dev: "{{ data_storage_device | default('/dev/sdb') }}"
    fstype: xfs
    force: false
    resizefs: true
```

Tegelikult isegi need moodulid, mida me siiani oleme kasutanud, on osa kollektsioonist **ansible.builtin**

Tänapäeval soovitatakse kõigi moodulite pihta viidata FQCN formaadis, näiteks **ansible.builtin.user**

Lisaks rollidele, saab Ansible-ssse laadida ka kollektsioone (collections) - “pakke”, milles võib olla sees igasugu asju.

Mooduleid kollektsioonidest peab kasutama nende FQCN-iga.

Praktiline 7 - Ansible Galaxy

17 Oma rolli initsialiseerimine

18 Galaxy rolli install ja kasutus

19 Galaxy kollektsiooni install ja kasutus

Kui valmis, võib aega pausi jaoks ära kasutada

Ansible - requirements

requirements.txt

```
---
collections:
  - name: community.general
    version: 8.2.0
  - name: community.crypto
    version: 2.16.0

roles:
  - name: geerlingguy.nginx
    version: 3.1.1
  - name: geerlingguy.redis
    version: 1.8.0
```

Kasutades ansible-galaxy tööriista, installitakse asjad Ansible projektist eraldi.

Selleks, et dokumenteerida mida projekt vajab, tuleks see kuskile kirja panna.

ansible-galaxy toetab selleks requirements.txt faili (nagu python), mida saab siis installida kasutades käsku:

```
ansible-galaxy install -r requirements.yml
```

Ansible - DevSec hardening

https://galaxy.ansible.com/ui/repo/published/devsec/hardening/docs/

Rollide/collectionite kasutus

Molecule

Ansible - Windows ja Võrguseadmed

08

Ansible - Windows

Ansible suudab, teatud mööndustega, hallata Windowsi masinaid.

Plugin Index

These are the plugins in the ansible.windows collection:

Modules

- async_status module – Obtain status of asynchronous task
- setup module – Gathers facts about remote hosts
- slurp module – Slurps a file from remote nodes
- win_acl module – Set file/directory/registry/certificate permissions for a system user or group
- win_acl_inheritance module – Change ACL inheritance
- win_audit_policy_system module – Used to make changes to the system wide Audit Policy
- win_audit_rule module – Adds an audit rule to files, folders, or registry keys
- win_auto_logon module – Adds or Sets auto logon registry keys.
- win_certificate_info module – Get information on certificates from a Windows Certificate Store
- win_certificate_store module – Manages the certificate store
- win_command module – Executes a command on a remote Windows node
- win_computer_description module – Set windows description, owner and organization
- win_copy module – Copies files to remote locations on windows hosts
- win_credential module – Manages Windows Credentials in the Credential Manager
- win_dhcp_lease module – Manage Windows Server DHCP Leases
- win_dns_client module – Configures DNS lookup on Windows hosts
- win_dns_record module – Manage Windows Server DNS records
- win_dns_zone module – Manage Windows Server DNS Zones
- win_dsc module – Invokes a PowerShell DSC configuration
- win_environment module – Modify environment variables on windows hosts

Töötab sarnaselt nagu UNIX-iga, aga kasutab hoopis Powershell skripte, mitte Pythonit.

Kuna Windowsil ei ole samu operatsioonisüsteemi APIsid Pythoni jaoks, siis on loodud Windowsile oma versioonid tüüpilistest käskudest.

Ansible - Windowsiga ühendamine

Windowsi maailm on kirju, seetõttu on ka Ansible Windowsi maailmas kirju.

Windowsiga ühendamiseks on 3 erinevat võimalikku protokolli:

- winrm
- psrp
- SSH (!?)

Autentimiseks on ka erinevaid variante: Basic, NTLM, Kerberos, CredSSP

Iga meetod vajab oma konfi, porte ja Pythoni librasid. Näiteks winrm-iga ühendudes:

pip install pywinrm

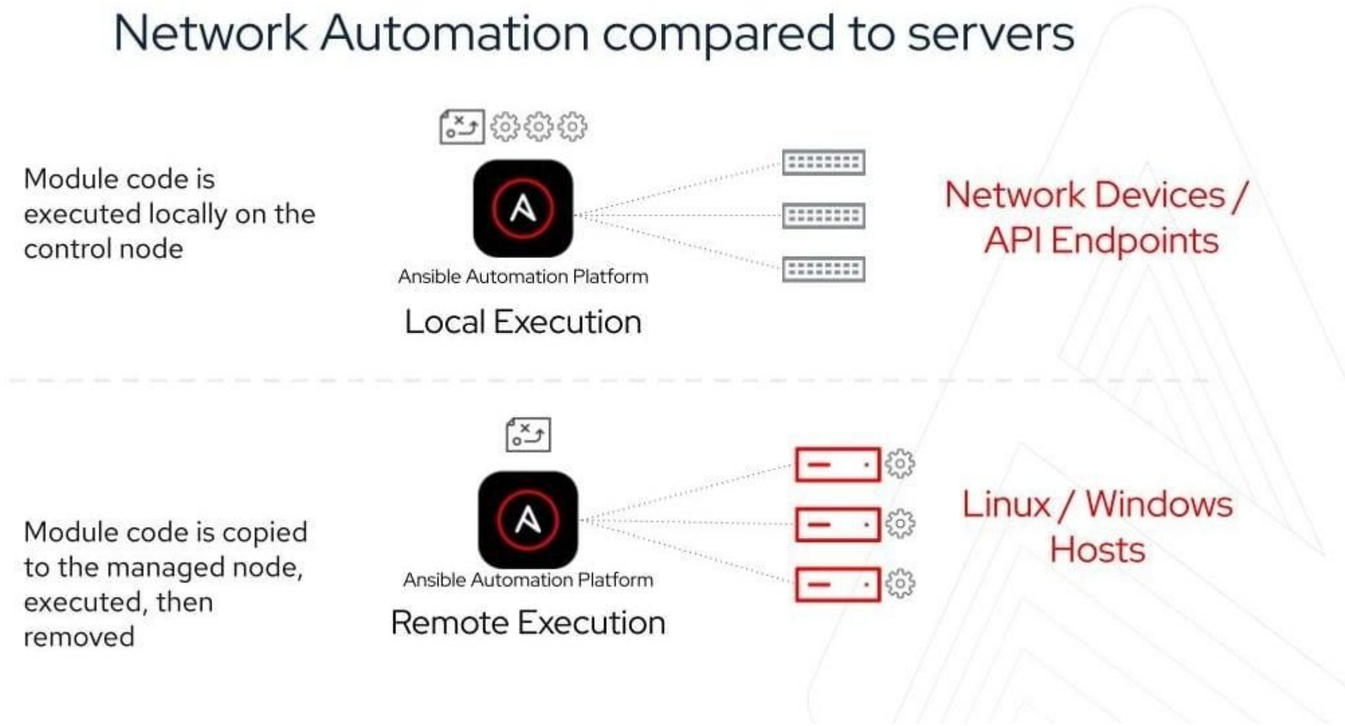
hosts

```
ansible_connection=winrm
ansible_winrm_port=5985
ansible_winrm_server_cert_validation=ignore
ansible_username=Administrator
ansible_password=<pass>
```

Ansible - Windows halduse dokumentatsioon

https://docs.ansible.com/projects/ansible/latest/os_guide/intro_windows.html

Ansible - võrguseadmete haldus



https://docs.ansible.com/projects/ansible/2.9/modules/list_of_network_modules.html#network-modules

Ansible - Lookupid, filtrid, saladused

09

Ansible - saladused

Best-practice arvestades, ei tohiks olla ükski saladus loetav Ansible projekti sees.

Neid tuleks võimalikult palju mitte kasutada, ära peita, või isegi kuskilt mujalt sisse lugeda.

Linux puhul paroolide kasutuse asemel näiteks SSH võtmed.

Muud paroolid kas ära krüpteerida, leida meetod kuidas neid Ansible-st väljaspoolt kasutada, või teha enterprise turvat.

Ansible - lookupid

Ansible-l on väga hea tööriist, kuidas “väljastpoolt” infot sisse lugeda (tüüpiliselt muutujasse).

Selleks kasutatakse lookup pluginaid.

Defineerides lookupi, jookseb see alati controller node peal.

Süntaks: "{{ lookup('<plugin>', '<arg1>’, ‘<arg2>) }}"

playbook.yml

```yaml
vars:
  # Read the contents of a public key from the controller
  my_ssh_key: "{{ lookup('file', '/home/user/.ssh/id_rsa.pub') }}"

  # Get a secret from an environment variable
  db_password: "{{ lookup('env', 'POSTGRES_PASSWORD') }}"
```

Kõrvalepõige - Ansible pluginad

Ansible ise on suhteliselt kerge raamistik - enamus muud on “plugin”. Neid saab lihtsamini installida, hallata, ise arendada, jms.

<projekt>/ansible.cfg

```
[defaults]

callbacks_enabled = profile_tasks
```

Näited primaarsetest plugina tüüpidest: https://docs.ansible.com/projects/ansible/latest/plugins/plugins.html

Ka moodul on tegelikult “plugin”. Et näha, mis pluginad juba installitud on:

```
ansible-doc -t <type> -l
```

Ansible - filtrid

<projekt>/playbook.yml

```yaml
vars:
  # Safety: Use 80 if 'http_port' is not defined
  port: "{{ http_port | default(80) }}"

  # Hashing: Turn a cleartext string into a Linux shadow hash
  shadow_hash: "{{ 'my_secret' | password_hash('sha512') }}"

  # Chaining: Join a list into a string, then capitalize it
  # ['a','b'] -> "a, b" -> "A, B"
  message: "{{ ['a', 'b'] | join(', ') | upper }}"

  # Three-way-merge: [ [ 1, "a", "d" ], [ 2, "b", "e" ], [ 3, "c", "f" ] ]
  insanity: "{{ [1,2,3] | zip(['a','b','c'], ['d','e','f']) }}"
```

Filtrid on äärmiselt võimsad jinja2 konstruktsioonid, et muutujaid vastavalt vajadusele mudida. Süntaks:

{{ muutuja | filter1 | filter2 }}

Filtreid on palju erinevaid, ja nendega saab teha igasugu lollusi.

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_filters.html

Ansible - Ansible Vault

Ansible üks kõige lihtsam variant paroole turvaliselt hoida on need ära krüpteerida parooliga.

Krüpteeritud faili võib panna reposse, sest see on loetamatu.

Kasutamiseks peab playbookile andma asukoha (var failina) ja vaulti parooli.

Ansible küsib vaulti parooli –ask-vault-pass lipuga.

vault/passwords.yml

```
$ANSIBLE_VAULT;1.1;AES256
32626430373063353234613432346261383363656461
353261303932343138356133383737373836
32666631643765643835636631383938316261343635
36360a376365396463306139333832366365
643161393235363764303336666164393561623865 31
656563393930626630323966353030333638
373832616263306537 0a6235663862326166643232 31
393163646166333663383637663139323363
3661
```

playbook.yml

```
- name: Demo using vaulted variables
  hosts: all
  vars_files:
    - vault/passwords.yml

  tasks:
    - name: Show vaulted secrets
      debug:
        msg:
          - "another_secret: {{ muutuja }}"
```

Ansible - Must Maagia

#FF0000