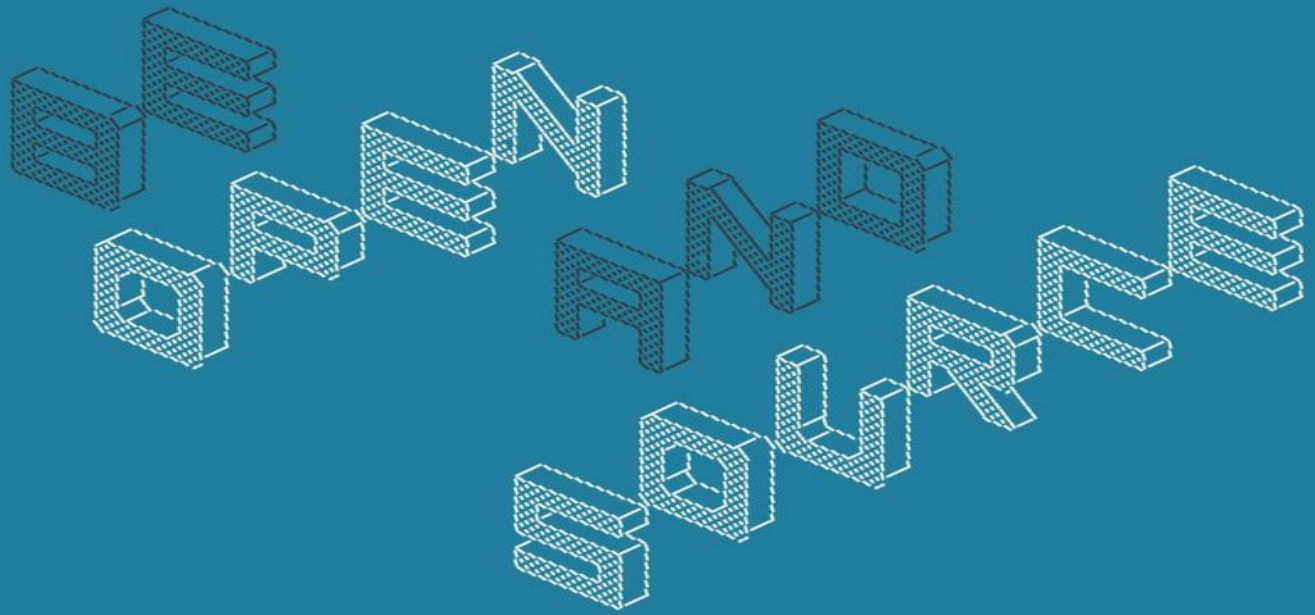
SLAC 2025

Secure Linux Administration Conference
vom 2.–4. Juni 2025 in Berlin



SLAC 2025

Adminsitriende

die auf Terminals starren.

Ansible Laufzeitoptimierung

SLAC 2025

Andreas Fritzsche

Systemadministrator

Danke an meine Freundin und Familie, die mir bei dem Vortrag sehr geholfen und mich unterstützt haben.
Die mir viele Aufgaben abgenommen haben.
Ohne sie gäbe es heute keinen Vortrag.

SLAC 2025

Das wohl bekannteste Optimierungsproblem ist das Auffinden eines Minimums oder Maximums einer differenzierbaren eindimensionalen Funktion $f(x)$, was in der Regel durch Berechnung der Nullstellen der ersten Ableitung $f'(x)$ gelingt. Ein einfaches Beispiel ist in nebenstehendem Bild aufgezeigt.

[https://de.wikipedia.org/wiki/Optimierungsproblem]

[https://gemini.google.com/ Flash 2.0]

SLAC 2025

Automatisierung ist wichtig, weil sie Prozesse effizienter, produktiver und kostengünstiger macht, menschliche Fehler reduziert und die Qualität verbessert. Sie ermöglicht es Unternehmen, sich auf wertschöpfende Tätigkeiten zu konzentrieren, anstatt Zeit mit repetitiven Aufgaben zu verschwenden.

[Google KI: Warum Automatisieren]

Effizienzsteigerung:

Durch Automatisierung werden Prozesse schneller und präziser abgewickelt.

Kosteneinsparungen:

Wiederholende Aufgaben, die manuell erledigt wurden, können nun mit Maschinen und Software erledigt werden, was zu Kosteneinsparungen führt.

Verbesserte Qualität:

Automatisierte Systeme sind weniger fehleranfällig, als menschliche Mitarbeiter, was die Qualität der Produkte und Dienstleistungen verbessert.

Freie Ressourcen:

Mitarbeiter können sich auf komplexere und kreativere Aufgaben konzentrieren, anstatt Zeit mit Routinearbeiten zu verbringen.

Flexibilität und Anpassungsfähigkeit:

Automatisierungssysteme können schnell an neue Bedürfnisse und Anforderungen angepasst werden.

Verbesserte Sicherheit:

In gefährlichen Umgebungen wie Bergbau oder Landwirtschaft kann Automatisierung das Unfallrisiko reduzieren.

Datensicherheit:

Automatisierte Systeme können dazu beitragen, Daten vor unbefugtem Zugriff und Verlust zu schützen.

Fachkräftemangel:

In Zeiten des Fachkräftemangels kann Automatisierung helfen, die Lücken im Personal zu schließen.

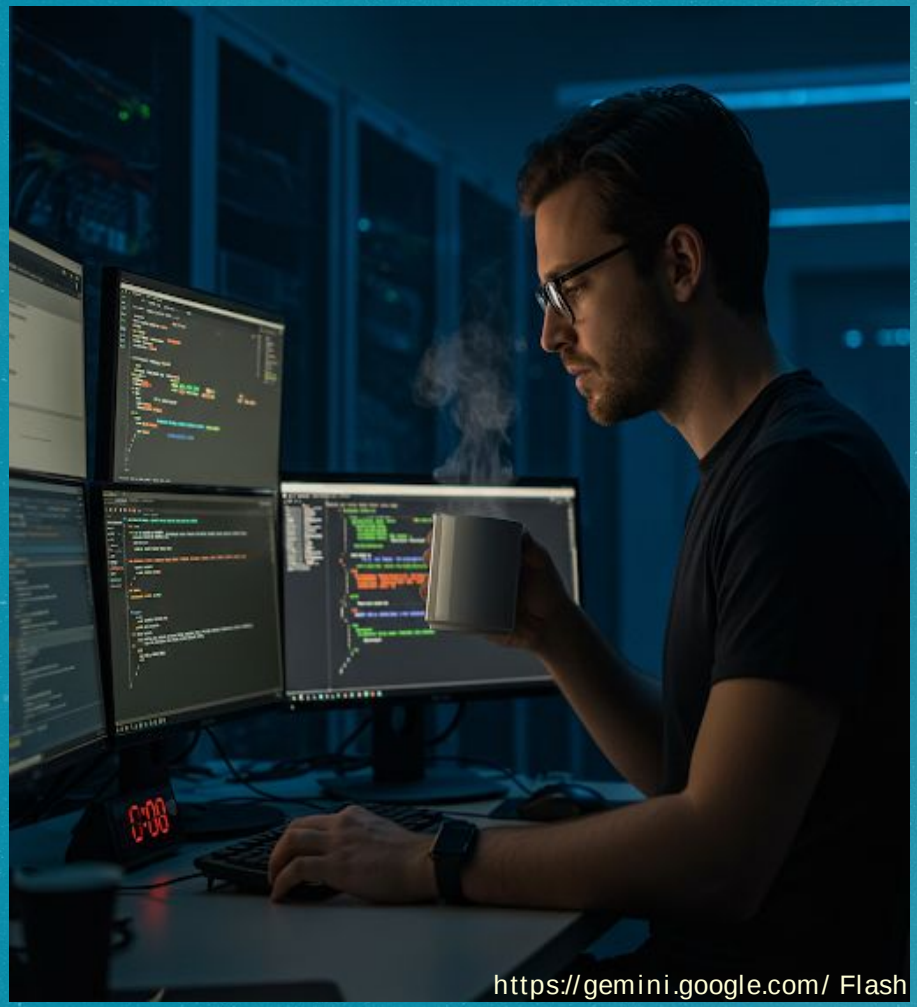
[Google KI: Warum Automatisieren]

SLAC 2025

Automatisierung ist wichtig,
weil sie Prozesse effizienter, produktiver und kostengünstiger mac
menschliche Fehler reduziert und die Qualität verbessert.
Sie ermöglicht es Unternehmen,
sich auf wertschöpfende Tätigkeiten zu konzentrieren,
anstatt Zeit mit repetitiven Aufgaben zu verschwenden.

[Google KI: Warum Automatisieren]

Das wohl bekannteste Optimierungsproblem ist das Auffinden eines Minimums
oder Ma
ist das Auffinden eines Minimums oder Maximums
in der Regel durch Berechnung der Nullstellen der ersten Ableitung $f'(x)$
gelingt. Ein einfaches Beispiel ist in nebenstehendem Bild aufgezeigt.



https://gemini.google.com/ Flash 2.0

Kaffee:

- psychotropes, koffeinhaltiges Getränk
- geröstete, gemahlene Kaffeebohnen + Wasser
- köstlich

Koffein:

- Erhöhung der Kontraktionskraft des Herzens/ Steigerung der Herzfrequenz
- Bronchialerweiterung
- Schwach harntreibende, Anregung der Peristaltik des Darms
- Leistungssteigernd (war mal auf Dopingliste)

SLAC 2025

ANSIBLE

[https://de.m.wikipedia.org/wiki/Datei:Ansible_logo.svg]

Was ist Ansible?

agentenlos

„Ansible ist agentenlos und kann daher Knoten managen, ohne dass darauf eine Software installiert werden muss.“

Verwendet SSH-Protokoll.

deklarativ/prozedural

kommt auf das Modul drauf an

SLAC 2025

ANSIBLE

[https://de.m.wikipedia.org/wiki/Datei:Ansible_logo.svg]

[https://en.wikipedia.org/wiki/Python_(programming_language)#/media/File:Python-logo-notext.svg]

Core & Module

Vorteil:

- Ökosystem
- Modulentwicklung

Nachteile:

- Python Abhängigkeit
- „Geschwindigkeit“

SLAC 2025

Schlüsselkomponenten

Managed Nodes

verwalteter Server oder Geräte

Inventorie(s)

Liste(n) von Manged Nodes

Hostfile

Enthält Hostangaben

Control Node

- Ansible installiert
- Ausführen von Task

[https://de.m.wikipedia.org/wiki/Datei:Ansible_logo.svg]

Schlüsselkomponenten

Ad-hoc-Befehl

Befehl oder Modul

Nicht wiederverwendbar

Playbooks

YAML

Plays ≥ 1

Plugins

Programme auf Controler Node

Rollen

Sammlung von Playbooks, Variablen, Dateien

Module

„kleine“ Programme auf Manage Nodes

ANSIBLE

[https://de.m.wikipedia.org/wiki/Datei:Ansible_logo.svg]

SLAC 2025

Plugins

become

cache

callback

cliconf

connection

httpapi

inventory

lookup

strategy

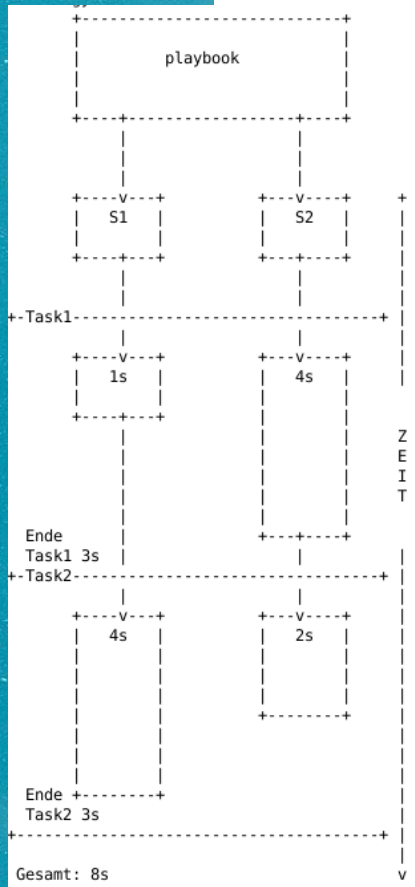
ANSIBLE

[https://de.m.wikipedia.org/wiki/Datei:Ansible_logo.svg]

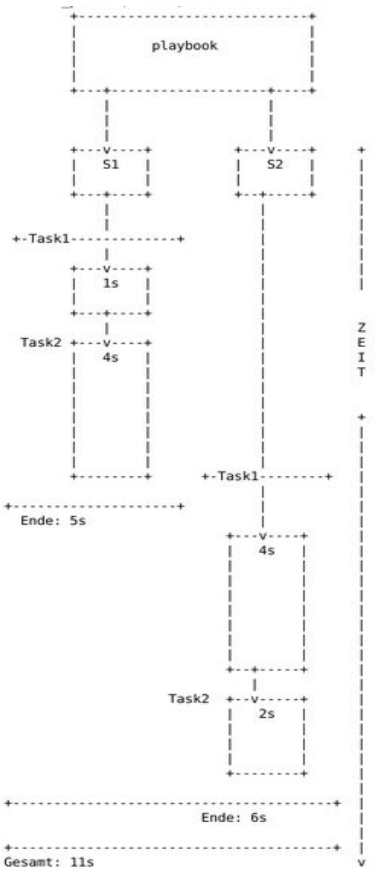
SLAC 2025

strategy

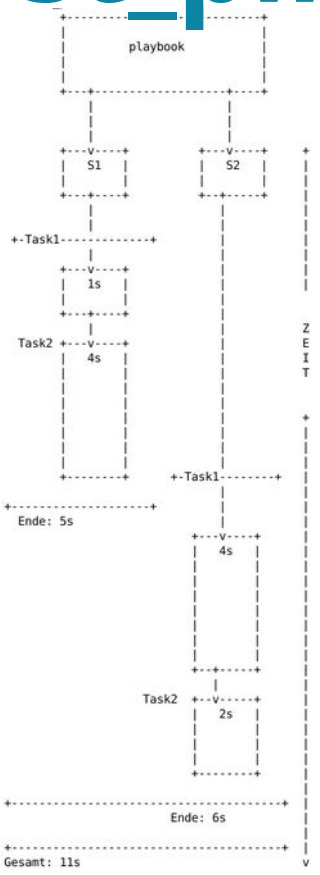
liniear



free

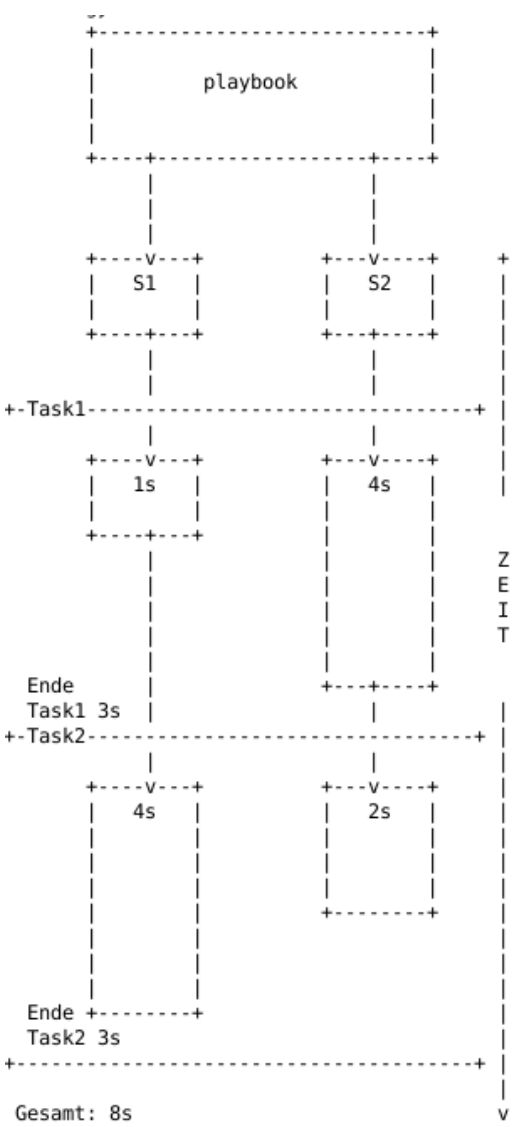


host_pinned



SLAC 2025

linear



Vorteil:

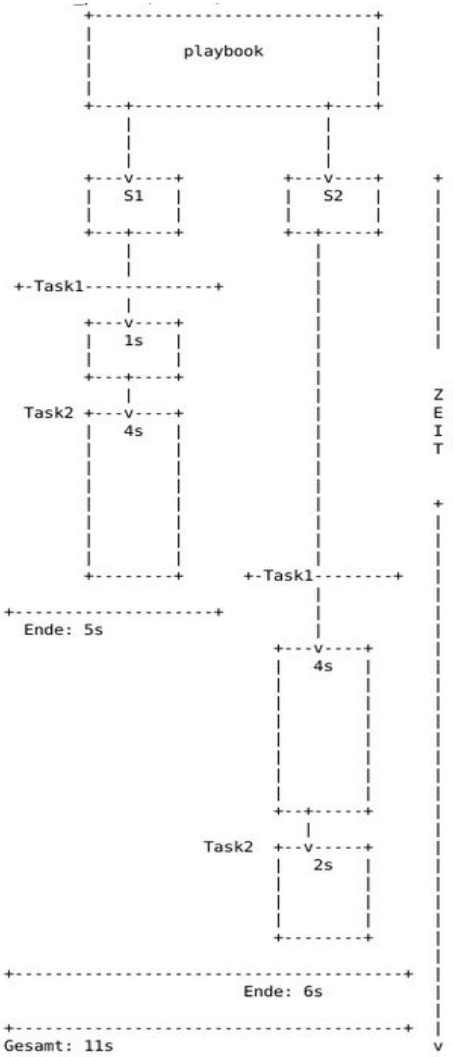
- Reihenfolge
- Debugging

Nachteile:

- Blockade durch langsame Hosts

SLAC 2025

free



Vorteil:

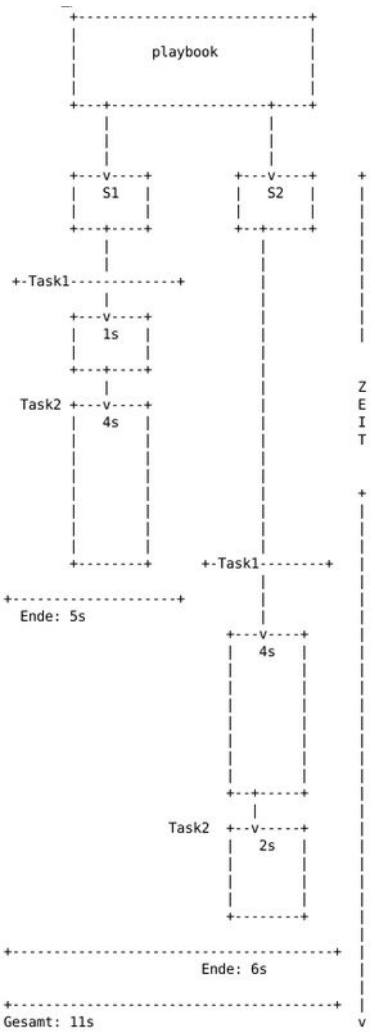
- Schneller bei unabhängigen Tasks
- Vielen Hosts

Nachteile:

- Ausgabe
- Debugging
- delegate to

SLAC 2025

host_pinned



Vorteil:

- wie free aber Hosts nacheinander

Nachteile:

- delegate to

SLAC 2025

Debug -v -vvv -vvvv

Vorteil:

- Troubleshooting

Nachteile:

- Hohe Last auf CN

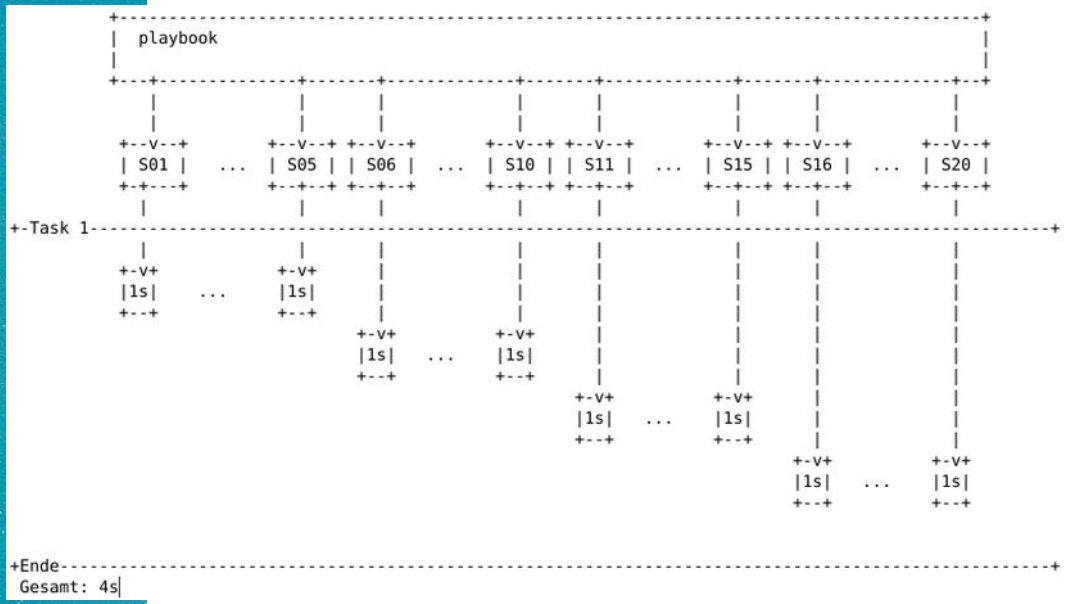
SLAC 2025

forks

5

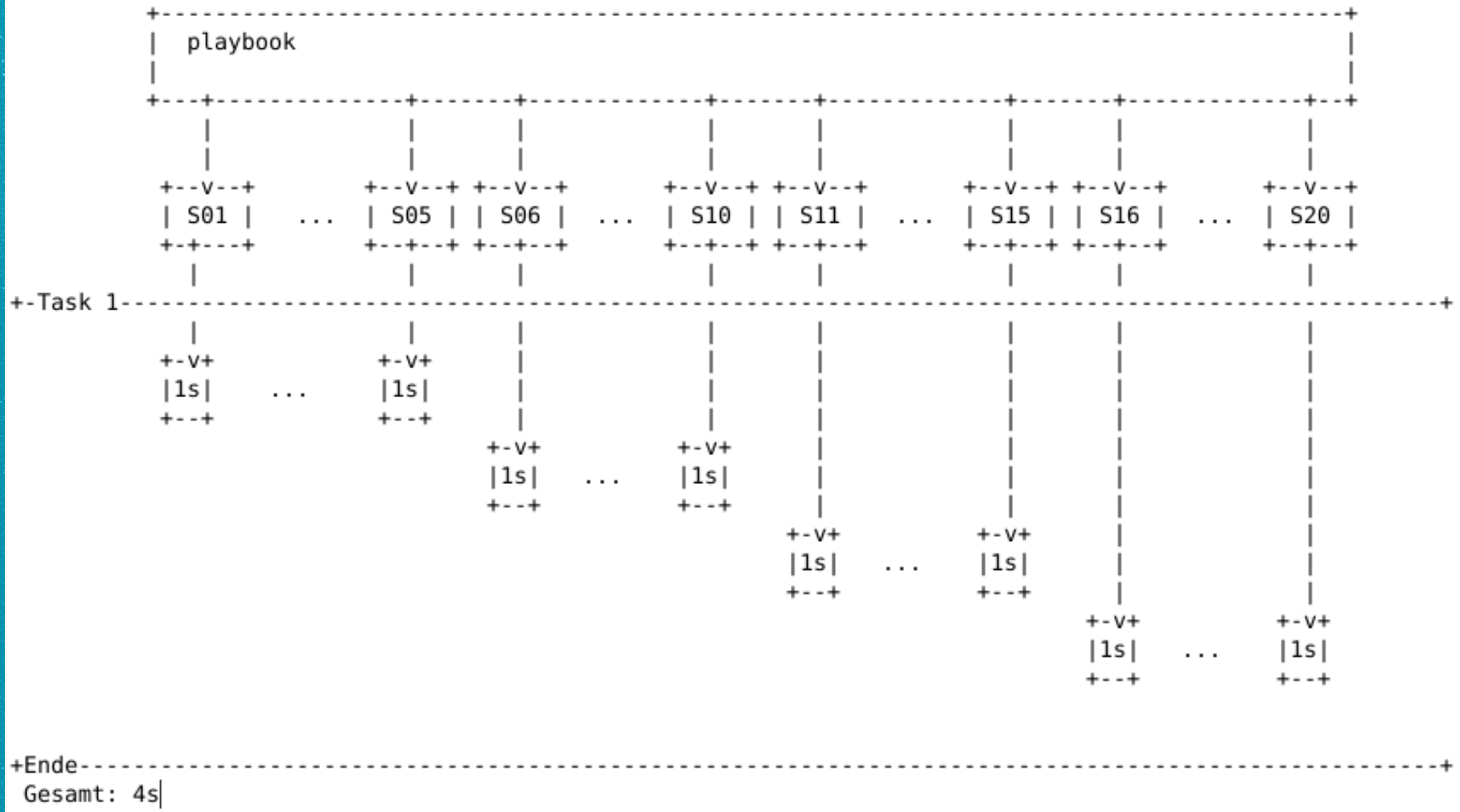
10

20



SLAC 2025

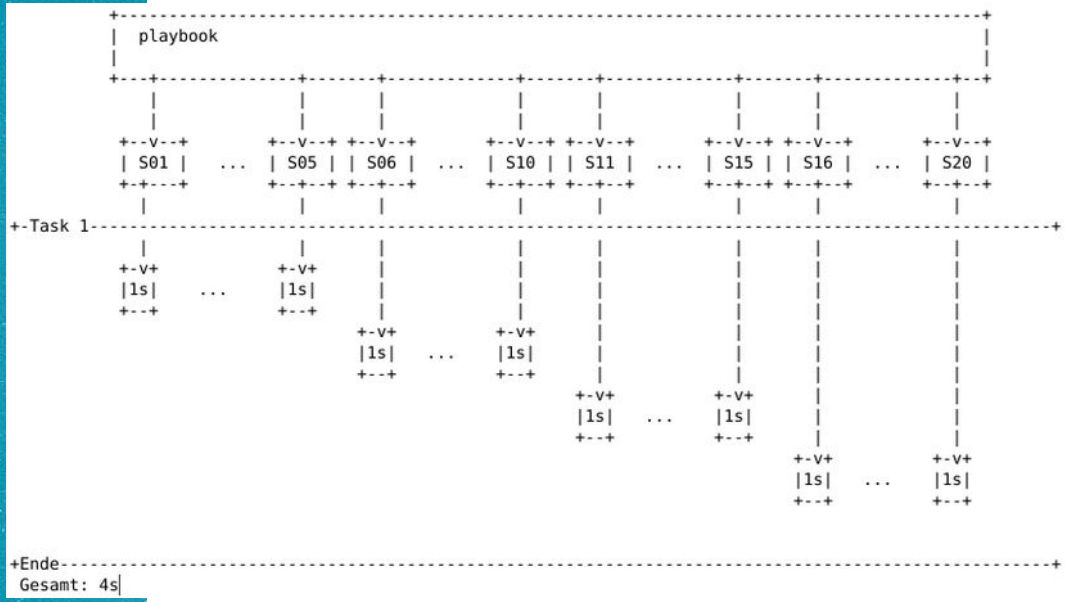
forks



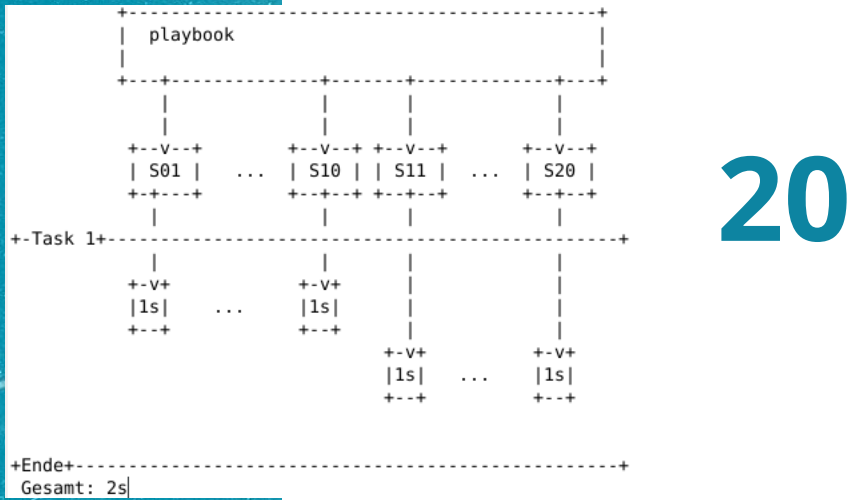
SLAC 2025

forks

5



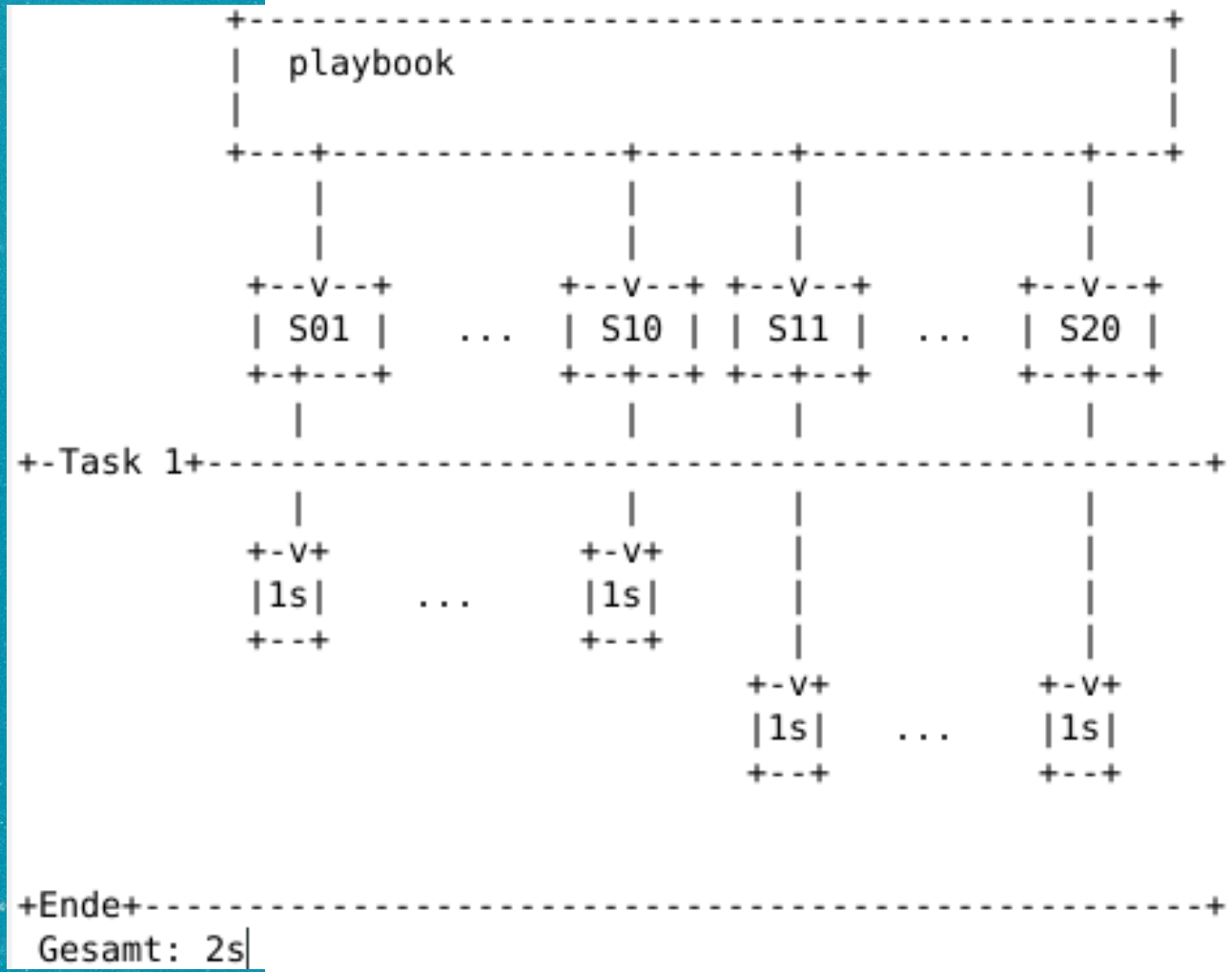
10



20

SLAC 2025

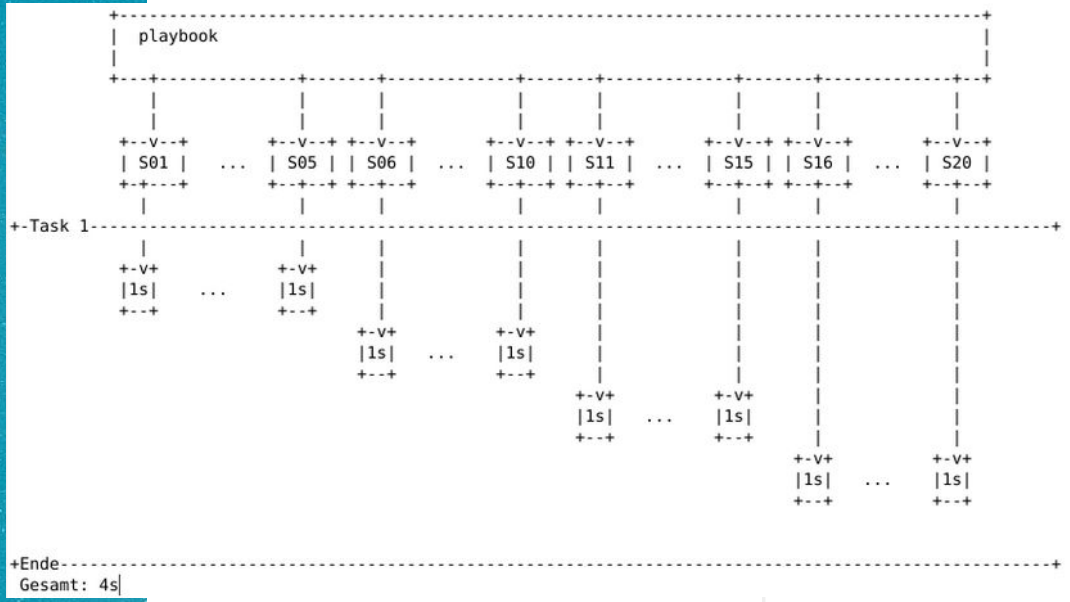
forks



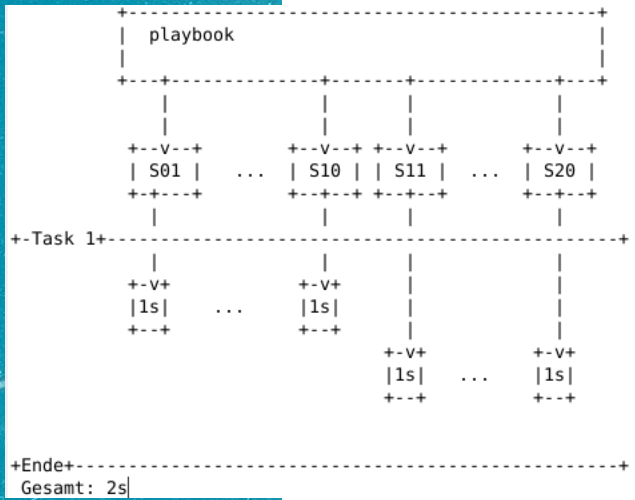
SLAC 2025

forks

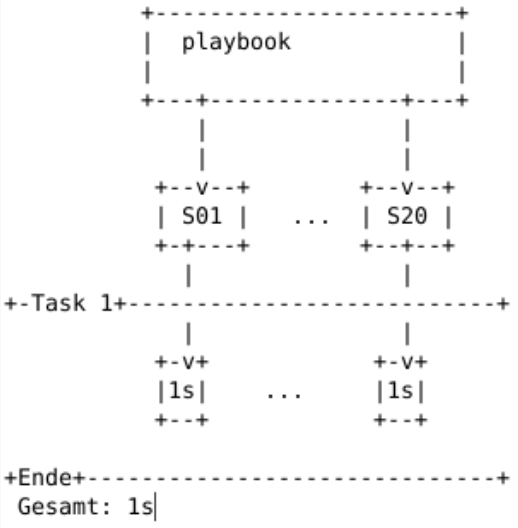
5



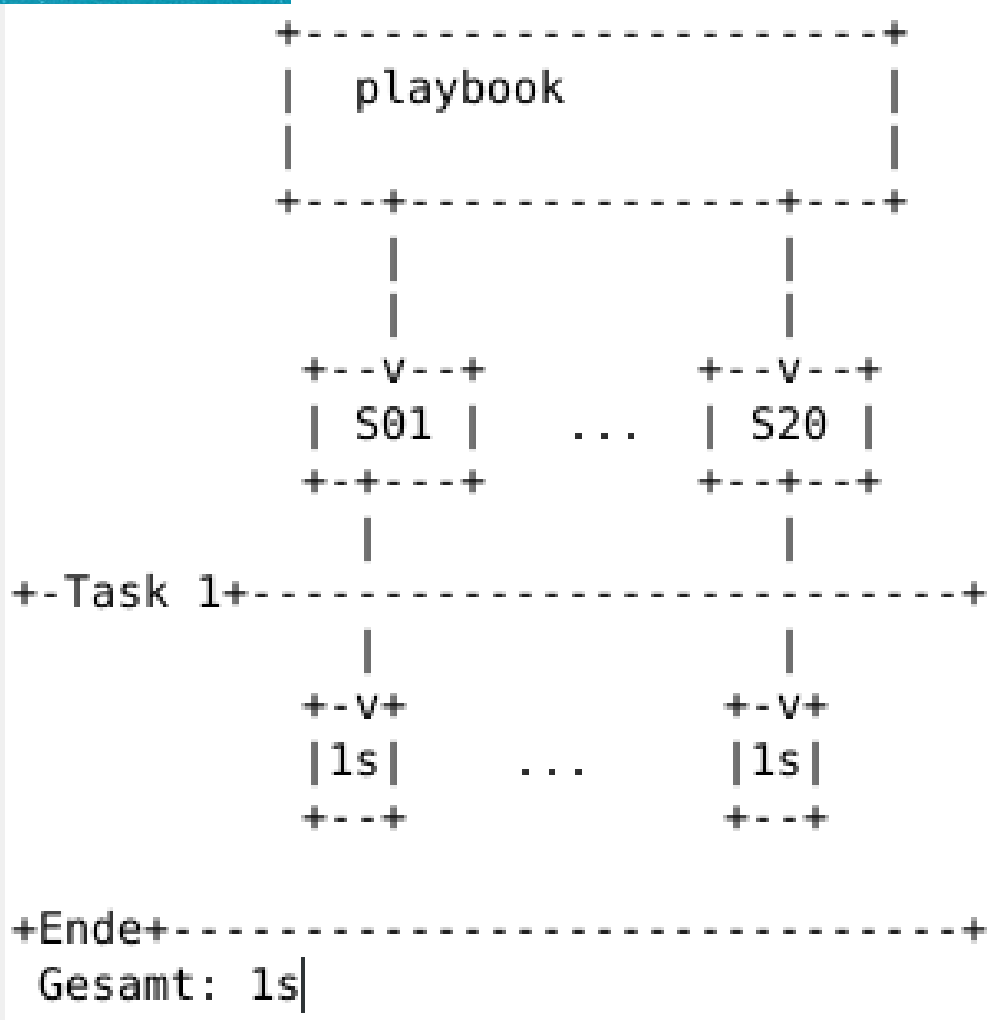
10



20



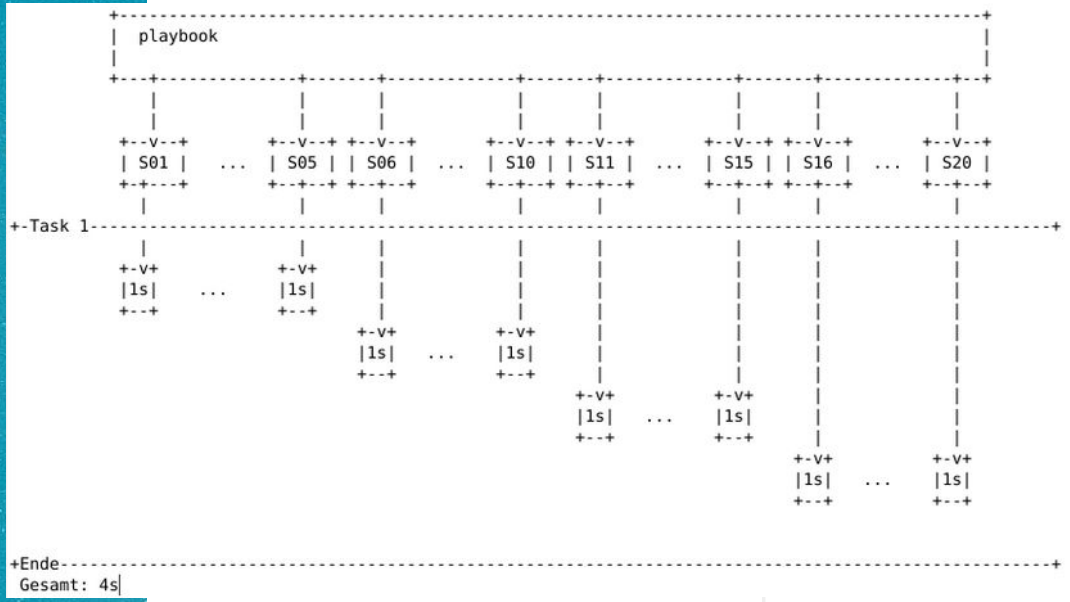
forks



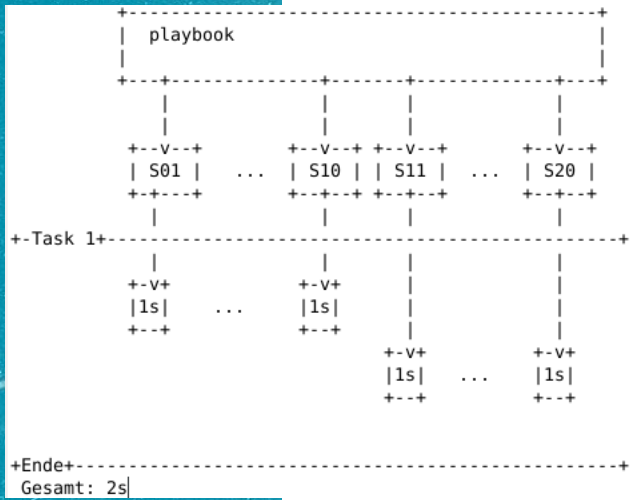
SLAC 2025

forks

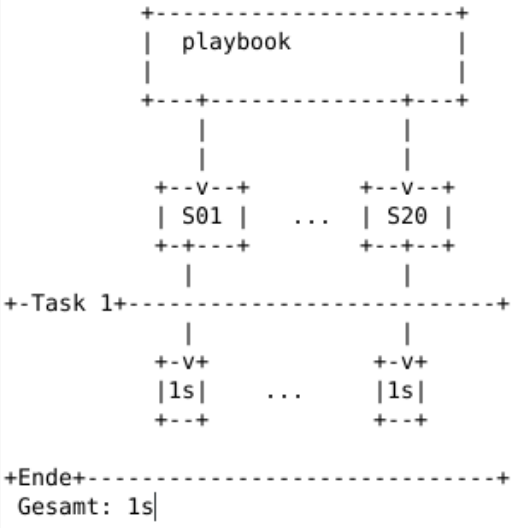
5



10



20



SLAC 2025

Messen

https://gemini.google.com/ Flash 2.0

...Achten wir auf die Zeit,
streckt sie sich,
achten wir nicht auf die Zeit,
vergeht sie ganz schnell...

[https://www.ndr.de/kultur/Zeit-vergeht-wie-im-Flug-oder-zieht-sich-ewig-warum-ist-das-so,zeit372.html]

SLAC 2025

time ansible-playbook play.yml

```
real    1m12.483s
user    0m8.535s
sys     0m2.225s
```

Vorteil:

- Simpel
- Standardtool

Nachteile:

- Nur kompletter Lauf

SLAC 2025

Ansible Callback Plugins

profile_tasks profile_roles timer

```
===============================================================================
websv ------------------------------------------------------------ 251.00s
common ----------------------------------------------------------- 84.80s
setup ------------------------------------------------------------ 1.87s
include_role ----------------------------------------------------- 1.80s
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
total ----------------------------------------------------------- 339.47s
===============================================================================
Gathering Facts -------------------------------------------------- 1.42s
websv : richte cronjob ein --------------------------------------- 1.31s
common : bilde leere liste mit gruppen --------------------------- 0.94s
common : zeige liste an ------------------------------------------ 0.74s
websv : entpacke skript ------------------------------------------ 0.68s
common : remove user_home directory ------------------------------ 0.63s
common : change user user group ---------------------------------- 0.63s
common : creating groups ----------------------------------------- 0.61s
common : copy change/date to /etc/motd --------------------------- 0.56s
common : print vars for uninstall -------------------------------- 0.55s
common : make user_home -> _user_home ---------------------------- 0.49s
websv : cleanJboss_loop delete rename change app websv directory -------- 0.49s
websv : do stat of WEBSV_LOG_HOME's old -------------------------- 0.44s
```

Vorteil:

- Teil von Ansible-core
- EinzelnE Komponenten

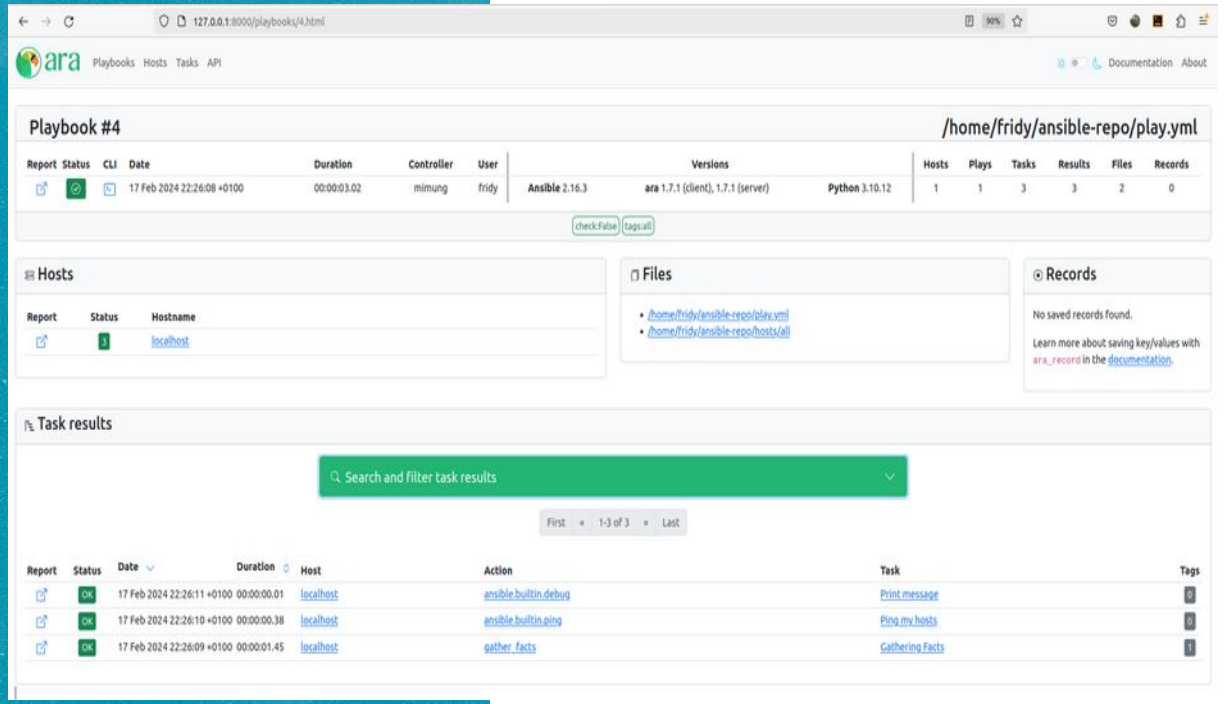
Nachteile:

- „Leserlichkeit“

SLAC 2025

Ansible ARA

https://ara.recordsansible.org/



Vorteil:

- Mehr als nur messen
- „logging“

Nachteile:

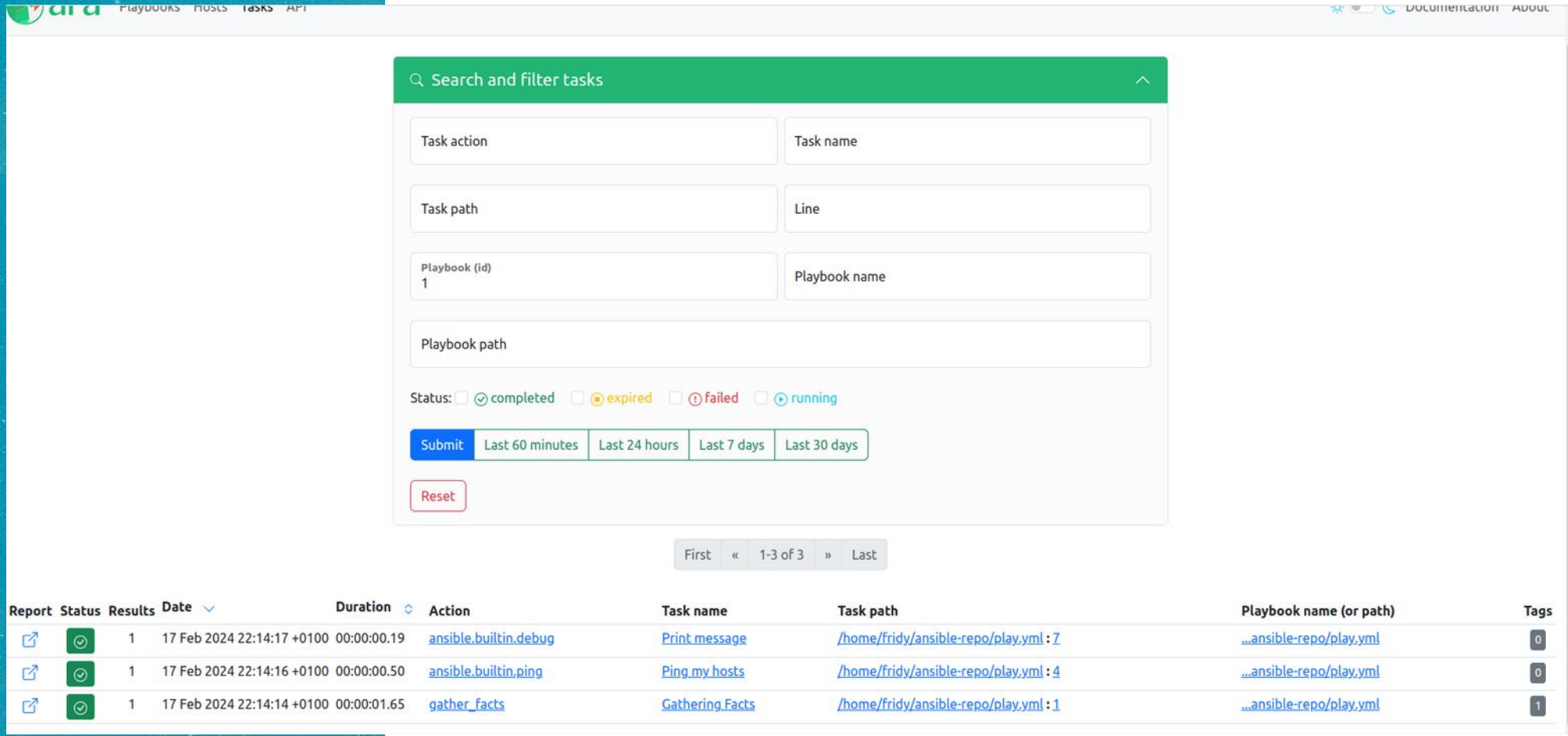
- Extra installation
- Extra Datenbank

Relevant:

- ansible-pull

SLAC 2025

Ansible ARA



SLAC 2025

Ansible AWX /(AAP)

https://github.com/ansible/awx

Vorteil:

- Sehr mächtig
- Parallelisierung

Nachteile:

- Komplexes Setup
- Ressourcenhungrig

SLAC 2025

POLEMARCH

https://polemarch.org/

Vorteil:

- Job History
- Clusterfähig(parallelisieren)

Nachteile:

- Extra Installation
- Weniger verbreitet

SLAC 2025

SEMAPHORE

https://semaphoreui.com

Vorteil:

- Runner
- Support
- Nicht nur für Ansible

Nachteile:

- Extra Installation
- Weniger verbreitet

SLAC 2025

Ansible Entwicklung

SLAC 2025

OS + Ansibleversion

Betriebssystem	Aktuell unterstützte Hauptversionen	Ansible Core Version
Red Hat Enterprise Linux (RHEL)	8;9	2.9-2.16;2.9-2.14
SUSE Linux Enterprise Server (SLES)	15	2.9
Debian	11; 12; 13	2.9;2.14;2.19
Ubuntu	22.04 LTS; 24.04 LTS; 24.10; 25.04	2.12; 2.16;2.16;2.18
FreeBSD	13.x, 14.x	-2.18, -2.18
NetBSD	9.x, 10.x	-2.18;-2.18
OpenIndiana	Hipster)	2.15
DragonFlyBSD	6.x, 7.x	3.9 - 3.10 3.11 - 3.12

SLAC 2025

OS + Pythonversion

Betriebssystem	Aktuell unterstützte Hauptversionen	Typische Python 3 Version(en) (Standard)	Python 2 Verfügbarkeit / Empfehlung
Red Hat Enterprise Linux (RHEL)	8;9	3.8 & 3.9; 3.9 - 3.11	-/-
SUSE Linux Enterprise Server (SLES)	15	3.9 & 3.10	-/-
Debian	11; 12; 13	3.9; 3.11; 3.12	-/-
Ubuntu	22.04 LTS; 24.04 LTS; 24.10; 25.04	3.10; 3.12; 3.12; 3.13	-/-
FreeBSD	13.x, 14.x	3.9 - 3.12 , 3.11 - 3.12	+/-
NetBSD	9.x, 10.x	3.8, - 3.10 3.9, -3.12	+
OpenIndiana	Hipster)	3.9 bis 3.11	+/-
DragonFlyBSD	6.x, 7.x	3.9 - 3.10 3.11 - 3.12	+/-

SLAC 2025

Optimieren

Fact Gathering

gather_facts:

- false (extrem schnell)
- true langsam

gather_subset

- nur benötigte Teile
  /wissen, was man braucht

```
========================          ========================
Gathering Facts -- 7.66s          debug ------------ 1.88s
debug ------------ 1.70s

real    0m20.125s                 real    1m12.483s
user    0m15.076s                 user    0m8.535s
sys     0m5.225s                  sys     0m2.225s
```

cache

Schnell aus dem Cache
json
+ einfach
- skaliert schlecht, nur ein CN
REDIS
+ Schnell, skaliert gut
- eigene Instanz

SLAC 2025

Playbook

```
- hosts: all
  gather_facts: false
  tasks:
    - name: Xauth
      ansible.builtin.dnf:
        name: xorg-x11-xauth
        state: latest
    - name: File
      ansible.builtin.file:
        src: '{{ item.src }}'
        dest: '{{ item.dest }}'
        state: link
      loop: {{ symlinks }}
    - name: Shell
      ansible.builtin.shell: ./sshtest.sh
      register: shelltest

    - name: Debug
      ansible.builtin.debug:
        msg: "Shelltest: {{ shelltest }}"
```

SLAC 2025

Optimieren

forks

20

```
forks 20
==============================
dnf ------------------- 21.05s
ansible.builtin.file --  3.50s
shell ----------------  2.42s
debug ----------------  1.86s
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
total ----------------- 28.83s
```

60

```
forks 60
==============================
dnf ------------------ 10.34s
ansible.builtin.file -  6.36s
debug ----------------  6.16s
shell ----------------  2.50s
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
total ---------------- 25.36s
```

„Playbook“

```
- hosts: all
  gather_facts: false
  tasks:
    - name: task 1
      ansible.builtin.shell: /bin/sleep 1
      register: result
    - name: task 1 result
      ansible.builtin.debug:
        msg: "Task 1 Script result: {{ result }}"
    - name: task 2
      ansible.builtin.shell: /bin/sleep 1
      register: result
    - name: task 2 result
      ansible.builtin.debug:
        msg: "Task 2 Script result: {{ result }}"
    - name: task 3
      ansible.builtin.shell: /bin/sleep 1
      register: result
    - name: task 3 result
      ansible.builtin.debug:
        msg: "Task 3 Script result: {{ result }}"
    - name: task 4
      ansible.builtin.shell: /bin/sleep 1
      register: result
    - name: task 4 result
      ansible.builtin.debug:
        msg: "Task 4 Script result: {{ result }}"
    - name: task 5
      ansible.builtin.shell: /bin/sleep 1
      register: result
    - name: task 5 result
      ansible.builtin.debug:
        msg: "Task 5 Script result: {{ result }}"
    - name: task 6
      ansible.builtin.shell: /bin/sleep 1
      register: result
    - name: task 6 result
      ansible.builtin.debug:
        msg: "Task 6 Script result: {{ result }}"
    - name: task 7
      ansible.builtin.shell: /bin/sleep 1
      register: result
    - name: task 7 result
      ansible.builtin.debug:
        msg: "Task 8 Script result: {{ result }}"
    - name: task 8
      ansible.builtin.shell: /bin/sleep 1
      register: result
    - name: task 8 result
      ansible.builtin.debug:
        msg: "Task 8 Script result: {{ result }}"
    - name: task 9
      ansible.builtin.shell: /bin/sleep 1
      register: result
    - name: task 9 result
      ansible.builtin.debug:
        msg: "Task 9 Script result: {{ result }}"
    - name: task 10
      ansible.builtin.shell: /bin/sleep 1
      register: result
    - name: task 10 result
      ansible.builtin.debug:
        msg: "Task 10 Script result: {{ result }}"
```

SLAC 2025

strategie forks: 25

10 Hosts

```
free                                 linear
==================================== ====================================
task 10 result ---------- 1.36s      task 7 ------------------ 3.36s
task 1 ------------------ 1.34s      task 8 ------------------ 1.45s
task 2 ------------------ 1.19s      task 1 ------------------ 1.43s
task 3 ------------------ 1.11s      task 10 ----------------- 1.42s
task 8 ------------------ 1.09s      task 9 ------------------ 1.35s
task 10 ----------------- 1.08s      task 6 ------------------ 1.35s
task 7 ------------------ 1.07s      task 5 ------------------ 1.34s
task 9 ------------------ 1.05s      task 4 ------------------ 1.34s
task 5 ------------------ 1.05s      task 2 ------------------ 1.33s
task 4 ------------------ 1.03s      task 3 ------------------ 1.32s
task 6 ------------------ 1.03s      task 10 result ---------- 0.20s
task 5 result ----------- 0.27s      task 3 result ----------- 0.15s
task 8 result ----------- 0.24s      task 5 result ----------- 0.13s
task 4 result ----------- 0.23s      task 2 result ----------- 0.12s
task 6 result ----------- 0.22s      task 4 result ----------- 0.12s
task 3 result ----------- 0.22s      task 9 result ----------- 0.12s
task 9 result ----------- 0.21s      task 7 result ----------- 0.11s
task 7 result ----------- 0.19s      task 8 result ----------- 0.11s
task 2 result ----------- 0.17s      task 6 result ----------- 0.11s
task 1 result ----------- 0.14s      task 1 result ----------- 0.10s
==================================== ====================================
ansible.builtin.shell -- 11.03s      ansible.builtin.shell -- 15.70s
ansible.builtin.debug --- 3.27s      ansible.builtin.debug --- 1.26s
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
total ------------------ 14.29s      total ------------------ 16.96s

real    0m14,922s                    real    0m17,597s
user    0m8,585s                     user    0m5,577s
sys     0m3,303s                     sys     0m2,967s
```

SLAC 2025

strategie forks: 25

50 Hosts

```
free                                  liniear
=================================== ===================================
task 10 ----------------- 2.99s task 9 ------------------- 4.80s
task 1 ------------------ 2.65s task 6 ------------------- 4.78s
task 10 result ---------- 2.48s task 5 ------------------- 4.71s
task 6 ------------------ 2.41s task 7 ------------------- 4.68s
task 8 ------------------ 2.14s task 1 ------------------- 3.90s
task 3 ------------------ 2.02s task 8 ------------------- 3.83s
task 7 ------------------ 1.91s task 2 ------------------- 3.79s
task 4 ------------------ 1.90s task 10 ------------------ 3.11s
task 4 result ----------- 1.59s task 3 ------------------- 2.87s
task 9 ------------------ 1.48s task 4 ------------------- 2.85s
task 8 result ----------- 1.21s task 10 result ---------- 0.89s
task 6 result ----------- 1.15s task 3 result ----------- 0.56s
task 1 result ----------- 1.09s task 1 result ----------- 0.54s
task 5 ------------------ 0.99s task 8 result ----------- 0.53s
task 2 ------------------ 0.93s task 2 result ----------- 0.53s
task 2 result ----------- 0.93s task 6 result ----------- 0.53s
task 3 result ----------- 0.84s task 7 result ----------- 0.52s
task 5 result ----------- 0.79s task 9 result ----------- 0.52s
task 9 result ----------- 0.78s task 4 result ----------- 0.52s
task 7 result ----------- 0.71s task 5 result ----------- 0.51s
=================================== ===================================
ansible.builtin.shell -- 19.38s ansible.builtin.shell -- 39.31s
ansible.builtin.debug -- 11.54s ansible.builtin.debug --- 5.65s
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
total ------------------- 30.92s total ------------------- 44.96s

real    0m31,779s                 real    0m45,701s
user    0m26,062s                 user    0m22,244s
sys     0m16,760s                 sys     0m14,709s
```

SLAC 2025

strategie forks: 25

58 Hosts

```
free                                 linear
==================================== ====================================
task 10 result ---------- 3.27s      task 3 ------------------ 4.96s
task 1 ------------------ 2.84s      task 4 ------------------ 4.94s
task 10 ----------------- 2.47s      task 2 ------------------ 4.89s
task 3 ------------------ 2.44s      task 9 ------------------ 4.87s
task 2 ------------------ 2.05s      task 1 ------------------ 4.08s
task 9 ------------------ 2.01s      task 7 ------------------ 3.98s
task 6 ------------------ 1.96s      task 10 ----------------- 3.96s
task 5 ------------------ 1.95s      task 5 ------------------ 3.95s
task 4 ------------------ 1.73s      task 8 ------------------ 3.95s
task 8 ------------------ 1.72s      task 6 ------------------ 3.94s
task 7 ------------------ 1.70s      task 10 result ---------- 1.14s
task 3 result ----------- 1.27s      task 9 result ----------- 0.74s
task 5 result ----------- 1.12s      task 8 result ----------- 0.72s
task 6 result ----------- 1.12s      task 5 result ----------- 0.72s
task 8 result ----------- 1.11s      task 6 result ----------- 0.71s
task 9 result ----------- 1.06s      task 2 result ----------- 0.71s
task 7 result ----------- 1.04s      task 7 result ----------- 0.67s
task 4 result ----------- 1.03s      task 4 result ----------- 0.67s
task 2 result ----------- 1.00s      task 3 result ----------- 0.64s
task 1 result ----------- 0.82s      task 1 result ----------- 0.53s
==================================== ====================================
ansible.builtin.shell -- 20.85s      ansible.builtin.shell -- 43.51s
ansible.builtin.debug -- 12.82s      ansible.builtin.debug --- 7.26s
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
total ------------------ 33.66s      total ------------------ 50.76s

real    0m34,584s                    real    0m51,631s
user    0m29,571s                    user    0m29,129s
sys     0m19,780s                    sys     0m19,106s
```

SLAC 2025

Optimieren

ssh Verbindung

pipeling

+weniger ssh Overhead
- sudo/tty

ControlMaster

+ mehrere simultane Sessions
- Stabilität

KeepAlive

+ weniger Verbindungsaufbauten
- Ressourcen

SLAC 2025

```
ControlMaster=auto -o ControlPersist=60s
pipline off
===============================================
task 1 ------------------------------------ 3.68s
task 10 ----------------------------------- 3.59s
task 7 ------------------------------------ 3.56s
task 8 ------------------------------------ 3.55s
task 2 ------------------------------------ 3.55s
task 5 ------------------------------------ 3.52s
task 6 ------------------------------------ 3.51s
task 9 ------------------------------------ 3.50s
task 3 ------------------------------------ 2.19s
task 4 ------------------------------------ 1.69s
task 10 result ---------------------------- 0.19s
task 5 result ----------------------------- 0.14s
task 4 result ----------------------------- 0.12s
task 2 result ----------------------------- 0.12s
task 3 result ----------------------------- 0.12s
task 1 result ----------------------------- 0.11s
task 9 result ----------------------------- 0.11s
task 8 result ----------------------------- 0.11s
task 7 result ----------------------------- 0.11s
task 6 result ----------------------------- 0.10s
===============================================
ansible.builtin.shell ------------------- 32.34s
ansible.builtin.debug ------------------- 1.23s
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
total ------------------------------------ 33.57s
```

```
ControlMaster=auto -o ControlPersist=60s
pipelining on
===============================================
task 8 ------------------------------------ 2.10s
task 7 ------------------------------------ 1.48s
task 1 ------------------------------------ 1.47s
task 10 ----------------------------------- 1.43s
task 2 ------------------------------------ 1.41s
task 6 ------------------------------------ 1.39s
task 5 ------------------------------------ 1.39s
task 3 ------------------------------------ 1.38s
task 9 ------------------------------------ 1.37s
task 4 ------------------------------------ 1.37s
task 10 result ---------------------------- 0.18s
task 4 result ----------------------------- 0.15s
task 2 result ----------------------------- 0.14s
task 5 result ----------------------------- 0.13s
task 6 result ----------------------------- 0.13s
task 3 result ----------------------------- 0.12s
task 7 result ----------------------------- 0.12s
task 9 result ----------------------------- 0.11s
task 8 result ----------------------------- 0.11s
task 1 result ----------------------------- 0.11s
===============================================
ansible.builtin.shell ------------------- 14.79s
ansible.builtin.debug ------------------- 1.29s
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
total ------------------------------------ 16.09s
```

```
#ControlMaster=auto -o ControlPersist=60s
pipline off
================================================
task 1 ------------------------------------ 3.80s
task 9 ------------------------------------ 3.66s
task 8 ------------------------------------ 3.65s
task 3 ------------------------------------ 3.63s
task 2 ------------------------------------ 3.63s
task 4 ------------------------------------ 3.63s
task 5 ------------------------------------ 3.61s
task 10 ----------------------------------- 3.52s
task 6 ------------------------------------ 2.51s
task 7 ------------------------------------ 1.67s
task 10 result ---------------------------- 0.23s
task 8 result ----------------------------- 0.15s
task 4 result ----------------------------- 0.14s
task 6 result ----------------------------- 0.14s
task 2 result ----------------------------- 0.13s
task 9 result ----------------------------- 0.13s
task 3 result ----------------------------- 0.13s
task 5 result ----------------------------- 0.13s
task 7 result ----------------------------- 0.12s
task 1 result ----------------------------- 0.11s
================================================
ansible.builtin.shell -------------------- 33.32s
ansible.builtin.debug --------------------- 1.40s
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
total ------------------------------------- 34.72s
```

```
#ControlMaster=auto -o ControlPersist=60s
pipelining on
================================================
task 6 ------------------------------------ 2.28s
task 4 ------------------------------------ 2.22s
task 9 ------------------------------------ 1.91s
task 1 ------------------------------------ 1.61s
task 8 ------------------------------------ 1.45s
task 2 ------------------------------------ 1.41s
task 3 ------------------------------------ 1.40s
task 7 ------------------------------------ 1.37s
task 10 ----------------------------------- 1.36s
task 5 ------------------------------------ 1.33s
task 10 result ---------------------------- 0.21s
task 6 result ----------------------------- 0.14s
task 7 result ----------------------------- 0.13s
task 9 result ----------------------------- 0.13s
task 5 result ----------------------------- 0.13s
task 2 result ----------------------------- 0.12s
task 4 result ----------------------------- 0.12s
task 3 result ----------------------------- 0.12s
task 1 result ----------------------------- 0.11s
task 8 result ----------------------------- 0.11s
================================================
ansible.builtin.shell -------------------- 16.33s
ansible.builtin.debug --------------------- 1.32s
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
total ------------------------------------- 17.66s
```

SLAC 2025

#ControlMaster=auto -o ControlPersist=60s
pipline off

```
task 1 ---------- 3.80s
task 9 ---------- 3.66s
task 8 ---------- 3.65s
task 3 ---------- 3.63s
task 2 ---------- 3.63s
task 4 ---------- 3.63s
task 5 ---------- 3.61s
task 10 ---------- 3.52s
task 6 ---------- 2.51s
task 7 ---------- 1.67s
task 10 result ---------- 0.23s
task 8 result ---------- 0.15s
task 4 result ---------- 0.14s
task 6 result ---------- 0.14s
task 2 result ---------- 0.13s
task 9 result ---------- 0.13s
task 3 result ---------- 0.13s
task 5 result ---------- 0.13s
task 7 result ---------- 0.12s
task 1 result ---------- 0.11s
ansible.builtin.shell ---------- 33.32s
ansible.builtin.debug ---------- 1.40s
total ---------- 34.72s
```

ControlMaster=auto -o ControlPersist=60s
pipline off

```
task 1 ---------- 3.68s
task 10 ---------- 3.59s
task 7 ---------- 3.56s
task 8 ---------- 3.55s
task 2 ---------- 3.55s
task 5 ---------- 3.52s
task 6 ---------- 3.51s
task 9 ---------- 3.50s
task 3 ---------- 2.19s
task 4 ---------- 1.69s
task 10 result ---------- 0.19s
task 5 result ---------- 0.14s
task 4 result ---------- 0.12s
task 2 result ---------- 0.12s
task 3 result ---------- 0.12s
task 1 result ---------- 0.11s
task 9 result ---------- 0.11s
task 8 result ---------- 0.11s
task 7 result ---------- 0.11s
task 6 result ---------- 0.10s
ansible.builtin.shell ---------- 32.34s
ansible.builtin.debug ---------- 1.23s
total ---------- 33.57s
```

#ControlMaster=auto -o ControlPersist=60s
pipelining on

```
task 6 ---------- 2.28s
task 4 ---------- 2.22s
task 9 ---------- 1.91s
task 1 ---------- 1.61s
task 8 ---------- 1.45s
task 2 ---------- 1.41s
task 3 ---------- 1.40s
task 7 ---------- 1.37s
task 10 ---------- 1.36s
task 5 ---------- 1.33s
task 10 result ---------- 0.21s
task 6 result ---------- 0.14s
task 7 result ---------- 0.13s
task 9 result ---------- 0.13s
task 5 result ---------- 0.13s
task 2 result ---------- 0.12s
task 4 result ---------- 0.12s
task 3 result ---------- 0.12s
task 1 result ---------- 0.11s
task 8 result ---------- 0.11s
ansible.builtin.shell ---------- 16.33s
ansible.builtin.debug ---------- 1.32s
total ---------- 17.66s
```

ControlMaster=auto -o ControlPersist=60s
pipelining on

```
task 8 ---------- 2.10s
task 7 ---------- 1.48s
task 1 ---------- 1.47s
task 10 ---------- 1.43s
task 2 ---------- 1.41s
task 6 ---------- 1.39s
task 5 ---------- 1.39s
task 3 ---------- 1.38s
task 9 ---------- 1.37s
task 4 ---------- 1.37s
task 10 result ---------- 0.18s
task 4 result ---------- 0.15s
task 2 result ---------- 0.14s
task 5 result ---------- 0.13s
task 6 result ---------- 0.13s
task 3 result ---------- 0.12s
task 7 result ---------- 0.12s
task 9 result ---------- 0.11s
task 8 result ---------- 0.11s
task 1 result ---------- 0.11s
ansible.builtin.shell ---------- 14.79s
ansible.builtin.debug ---------- 1.29s
total ---------- 16.09s
```

SLAC 2025

Optimieren

ssh Verbindung

pipeling

+weniger ssh Overhead
- sudo/tty

ControlMaster

+weniger ssh Overhead
+ loops
+ kurze Aufgaben
- Stabilität

KeepAlive

+ weniger Verbindungsaufbauten
- Ressourcen

SLAC 2025

Playbook

```
- hosts: all
  gather_facts: false
  tasks:
    - name: Xauth
      ansible.builtin.dnf:
        name: xorg-x11-xauth
        state: latest
    - name: File
      ansible.builtin.file:
        src: '{{ item.src }}'
        dest: '{{ item.dest }}'
        state: link
      loop: {{ symlinks }}
    - name: Shell
      ansible.builtin.shell: ./sshtest.sh
      register: shelltest

    - name: Debug
      ansible.builtin.debug:
        msg: "Shelltest: {{ shelltest }}"
```

SLAC 2025

Mit ohne Mitogen

60 Forks

```
Mitogen linear   free
================ ================
dnf ----- 120.06s dnf ----- 165.00s
debug --- 39.09s  debug --- 60.54s
file ---- 22.03s  file ---- 43.88s
shell --- 21.78s  shell --- 43.02s
 ~~~~~~~~~~~~~~~~ ~~~~~~~~~~~~~~~~

total --- 202.96s total --- 312.44s
ohne      linear free
================ ================
dnf ----- 100.98s dnf ----- 126.73s
debug --- 36.75s  debug --- 61.03s
file ---- 99.05s  file ---- 104.95s
shell --- 100.56s shell --- 110.64s
 ~~~~~~~~~~~~~~~~ ~~~~~~~~~~~~~~~~

total --- 337.35s total --- 403.36s
```

SLAC 2025

Optimieren

Playbooks

Vermeide unötige Wiederholungen

Variablen

Verwende loops

Conditional-Statements

SLAC 2025

Optimieren

Playbooks

delegete_to + run_once

Verwendung von handler

Kein Debug wenn nicht wirklich gebraucht

Module statt command

SLAC 2025

Optimieren

Playbooks

throttle

Copy statt template

“rescue“+“always“

async

Langlaufende Task in Hintergrund

SLAC 2025

asyncron

```
- hosts: all
  gather_facts: false
  tasks:
    - name: task 1 - background task
      ansible.builtin.command: /bin/sleep 5
      async: 10
      poll: 0
      register: async_task_result

    - name: task 2
      ansible.builtin.debug:
        msg: task 2

    - name: task 3
      ansible.builtin.debug:
        msg: task 3

    - name: Task 4 - wait for async task
      ansible.builtin.async_status:
        jid: "{{ async_task_result.ansible_job_id }}"
      until: job_result.finished
      register: job_result
      retries: 10
      delay: 2

    - name: task 5
      ansible.builtin.debug:
        msg: task 4
```

```
=========================================
Task 4 - wait for async task ------ 7.16s
task 1 - background task ---------- 0.82s
task 5 ---------------------------- 0.18s
task 2 ---------------------------- 0.11s
task 3 ---------------------------- 0.10s
=========================================
ansible.builtin.async_status ------ 7.16s
ansible.builtin.command ----------- 0.82s
ansible.builtin.debug ------------- 0.39s
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
total ----------------------------- 8.37s

real    0m8,956s
user    0m2,577s
sys     0m1,071s
```

SLAC 2025

fire and forget

```
pol 2                                  pool 0
=================================      =================================
task 3 ------------------- 17.42s      task 1 ------------------- 2.91s
task 9 ------------------- 16.94s      task 3 ------------------- 2.70s
task 2 ------------------- 16.93s      task 7 ------------------- 2.67s
```

```
"msg": "Task 10 Script result:
{'started': 1, 'finished': 1, 'stdout': '', 'stderr': '', 'stdout_lines': [], 'stderr_lines': [],
 'ansible_job_id': '656541655692.3401700', 'results_file': '*/656541655692.3401700',
 'changed': True, 'rc': 0, 'cmd': '/bin/sleep 12',
'start': '2024-03-14 10:39:55.574215',
'end': '2024-03-14 10:40:07.579631',
'delta': '0:00:12.005416', 'msg': '', 'failed': False}"

"msg": "Task 10 Script result:
{'failed': 0, 'started': 1, 'finished': 0, 'ansible_job_id': '625777812121.3237193',
'results_file': '*/625777812121.3237193', 'changed': True}"
```

```
task 3 result ------------ 0.11s      task 6 result ------------ 0.11s
task 5 result ------------ 0.11s      task 8 result ------------ 0.11s
=================================      =================================
ansible.builtin.shell --- 161.68s      ansible.builtin.shell ---- 15.55s
ansible.builtin.debug ----- 1.28s      ansible.builtin.debug ----- 1.23s
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
total -------------------- 162.96s      total -------------------- 16.78s
```

SLAC 2025

Optimieren

Module + Plugin

Modul Entwickeln

Wenn nötig mit C /Go

Plugin Entwickeln

Spliten eine Variable

sid: s-c1-major-minor-l-k1-e-v01

Service

Cluster 1

Major

Minor

Leipzig

Kunde 1

Entwicklung

Suchen einer Variable

connection-s-major-minor-l-k1-e-status

connection-s-major-minor-l-k1-e

connection-s-major-minor-l-k1-status

...

connection:

SLAC 2025

Optimieren

Paralelisieren

ansible-paralell

Gleichzeitig mehrere playbooks
(+-) Auslastung CM
- extra Install

ansible-pull

Ansible Installation auf Manage Node

SLAC 2025

ansible-pull

MN läuft Ansible selber

+ geringer Laste auf CM
+ weniger Netzwerk CM<→MN
+ skaliert gut

- zusätzliche Last auf MN
- debug
- Übersicht
- mehr Netzwerk Reproserver ↔️ MN

SLAC 2025

Optimieren

Paralelisieren

ansible-paralell

Gleichzeitig mehre playbooks
(+-) auslastung CM
- extra Install

ansible-pull

Ansible Installation auf Manage Node

eigene Idee

+ simpel
- noch nicht ausgereift

SLAC 2025

eigene Idee Script

[https://gitlab.com/1fridy/admins_die_auf_terminals_starren_ansible_laufzeitoptimierung]

ansible-playbook -i 60 test_mit3tasks.yml

```
ansible-playbook  -i 60 test_mit3tasks.yml
==============================
dnf ------------------- 10.32s
ansible.builtin.file --  4.40s
shell ----------------  1.63s
debug ----------------  1.40s
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
total ----------------- 17.74s

real    0m19,026s
user    0m9,651s
sys     0m5,999s
```

ansible-playbook -limit all[0:19] -i 60 test_mit3tasks.yml &

ansible-playbook -limit all[20:39] -i 60 test_mit3tasks.yml &

ansible-playbook -limit all[40:59] -i 60 test_mit3tasks.yml &

```
==> split_20_hosts_1_1 <==     ==> split_20_hosts_2_1 <==     ==> split_20_hosts_3_1 <==
============================   ============================   ============================
dnf ------------------- 5.48s  dnf ------------------- 5.94s  dnf ------------------- 5.78s
shell ----------------- 1.36s  ansible.builtin.file -- 1.22s  ansible.builtin.file -- 1.37s
ansible.builtin.file -- 1.28s  shell ----------------- 1.00s  shell ----------------- 1.06s
debug ----------------- 1.24s  debug ----------------- 0.85s  debug ----------------- 0.83s
~~~~~~~~~~~~~~~~~~~~~~~~~~~~   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
total ----------------- 9.36s  total ----------------- 9.01s  total ----------------- 9.04s

real    0m10,013s
user    0m9,481s
sys     0m2,989s
```

SLAC 2025

parallel.sh ansible-playbook -i 60 test_mit3tasks.yml

eigene Idee Script

[https://gitlab.com/1fridy/admins_die_auf_terminals_starren_ansible_laufzeitoptimierung]

```
#!/usr/bin/env bash
anzahl_paralleler_aufrufe=3
if [[ $* = '' ]]
then
  echo "nichts übergeben"
  exit 1
fi
ubergabe=$*
echo "Übergabe $ubergabe"
enthalt_ansibleplaybook=$( egrep -c '^ansible-playbook' <<< $ubergabe )
if [[ $enthalt_ansibleplaybook -eq 0 ]]
then
  echo "kein ansible-playbook enthalten"
  exit 3
fi
enthalt_limits=$( grep -c '\--limit' <<< $ubergabe)
list_hosts=$( sed 's/ansible-playbook/ansible-playbook --list-hosts/ '<<< $ubergabe)
anzahlhosts=$( $list_hosts | awk '/hosts/{ sub("\\(", "", $2);sub("\\):", "", $2);print $2 }')
if [[ -z $anzahlhosts ]]
then
  echo "keine Hostanzahl bestimmbar"
  exit 2
fi

weite=$( echo "$anzahlhosts/$anzahl_paralleler_aufrufe" | bc )
#echo "weite: $weite"
if [[ -z $weite ]]
then
  echo "keine Weite bestimmbar"
  exit 4
fi
#echo "Übergabe $ubergabe anzahl: $anzahlhosts"
if [[ $enthalt_limits -eq 0 ]]
then
  if [[ $weite -eq 0 ]]; then runmax=$[$anzahlhosts-1]; weite=1; else runmax=$[$anzahl_paralleler_aufrufe-1]; fi
  echo "runmax $runmax"
  for ((i=0 ; i<$runmax ; i++))
  do
    startwert=$[$i * $weite ]
    endwert=$[ ( ( ($i+1) * $weite) - 1 ) ]
    line=$( sed 's/ansible-playbook/ansible-playbook --limit all:['$startwert':'$endwert']/; s/$/&/' <<< $ubergabe  )
    echo $line
    $line&
  done
  startwert=$[$i*$weite]
  endwert=$[$anzahlhosts -1]
  line=$( sed 's/ansible-playbook/ansible-playbook --limit all:['$startwert':'$endwert']/; s/$/&/' <<< $ubergabe  )
  echo $line
```

SLAC 2025

Optimieren

Sonstiges

Logfile

Aufbau Inventory

```
all:
  children:
    uk:
      children:
        england:
          children:
            london:
              children:
                prod:
        wales:
          children:
            test
prod:
  children:
    awx:
test:
  children:
    selfhosted:
selfhosted:
  host:
    s1:
awx:
  host:
    a1:
    a2:
```

SLAC 2025

Optimieren

Zusammenfassung

Viele kleine Möglichkeiten.

Seine Systeme und Anforderungen Kennen.

* Engpässe erkennen.
* Testen
* Parallelisieren
  → Es muss nicht immer AWX sein SLAC 2026 ;-)
* Zum Schluss Eigenentwicklung
  → Jetzt entwickle ich für Ansible SLAC 2026 ;-)


Fragen?

SLAC 2025

https://gitlab.com/1fridy/
admins_die_auf_terminals_starren_ansible_laufzeitoptimierung

SLAC 2025

Thank you

denn

Zeit ist Geld