// DIGITALE SCHULUNGEN MIT GITLAB CI / CD

@STDEVEL

CHRISTIAN STANKOWIC

> Senior System Engineer und Technical Leader bei SVA

> Linux, DevOps, Infrastructure as Code, Trainer

@OKIN

NIKO WENSELOWSKI

> System Engineer bei SVA

> kümmert sich um DevOps bei *SVA Operational Services*

> GitLab, OpenShift, Linux, Python, IaC

// AGENDA

> Motivation

> Tooling

> Automatisierte Infrastruktur

> Testing für Trainer und Teilnehmer

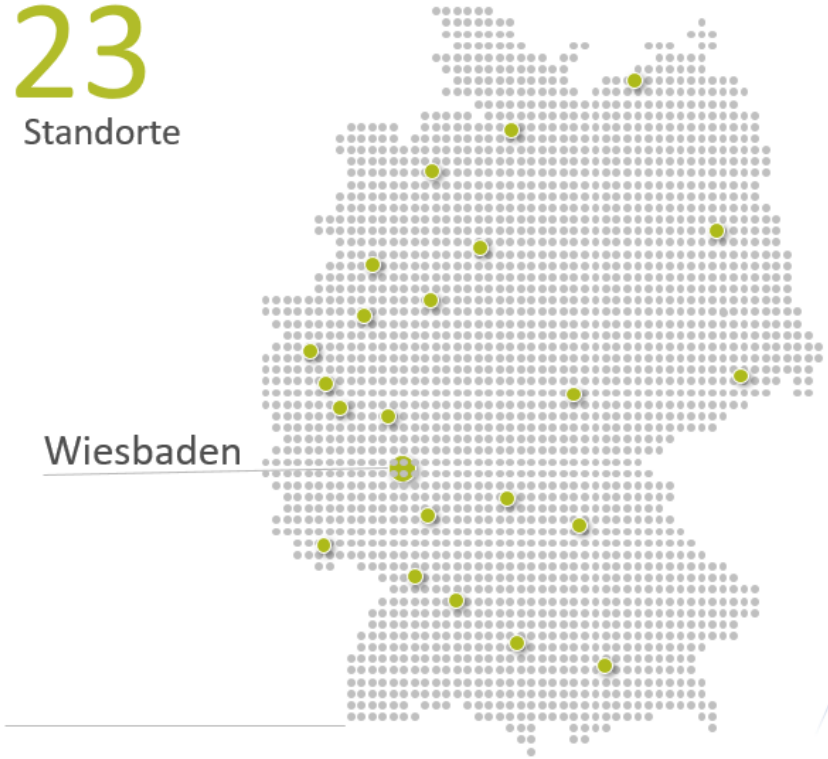
> GitLab CI

> Lessons Learned

> Fazit

UNTERNEHMEN SVA

23
Standorte



6 TOP
Branchen

- Automotive
- Retail
- Public
- Maschinen & Anlagenbau
- Finance & Insurance
- Telekommu-nikation

PORTFOLIO

Datacenter-Infrastruktur

- Storage- & Server-Systeme
- IP-Netzwerk-Infrastruktur
- Software Defined Data Center

Mainframe

- Services / Consulting
- zHosting
- zBusiness Services
- IBM System z Hardware
- Managed Services

End User Computing

- Application & Desktop (CAD/E) Virtualization
- Virtual Workspace & Mobility
- Application, Information & Device Management
- Unified Endpoint Management

Big Data & Analytics

- Big Data Plattformen
- IT Operations Analytics
- Business Analytics und IoT

SAP

- Architektur, Sizing & Beratung
- Installation, Konfiguration, Upgrades
- OS-/DB-Migrationen & Unicode-Konvert.
- Betriebsunterstützung, Managed Services

Business Continuity

- Archivierung
- Backup und Recovery
- Disaster Recovery
- High Availability
- Notfallplanung

Service Management

- Software Asset Management
- Enterprise Asset Management
- IT Service Management
- Enterprise Service Management

IT Security

- Information Security Consulting
- IT-Security-Architektur und -Integration
- Vulnerability Assessment und Management
- Security Operational Services
- Security Incident Response

SVA Produkte

- BVQ
- IDR
- Liberyse
- medPower

// MOTIVATION

MOTIVATION

> IT ist **kurzlebig**, immer **mehr** Infrastruktur in **kürzerer** Zeit

MOTIVATION

> IT ist **kurzlebig**, immer **mehr** Infrastruktur in **kürzerer** Zeit

> Während das Halbwertszeit von Fachwissen bei ca. 5 Jahren liegt, kommt IT-Fachwissen auf **knapp 1,5 Jahre***

> Neues Wissen sollte schnellstmöglich adaptiert werden

MOTIVATION

> IT ist **kurzlebig**, immer **mehr** Infrastruktur in **kürzerer** Zeit

> Während das Halbwertszeit von Fachwissen bei ca. 5 Jahren liegt, kommt IT-Fachwissen auf **knapp 1,5 Jahre***

> Neues Wissen sollte schnellstmöglich adaptiert werden

> Mitarbeiter sollen über **aktuelles Wissen** verfügen aber dennoch **profitabel** sein

> Die Auswahl an Werkzeugen ist (zu) groß

*Klassiker ausgenommen, Quelle: BIBB

MOTIVATION

> Corona-**Pandemie** macht **Vor-Ort**-Veranstaltungen **schwierig bis unmöglich**

MOTIVATION

> Corona-**Pandemie** macht **Vor-Ort**-Veranstaltungen **schwierig bis unmöglich**

> **Hands-On**-Übungen ermöglichen Teilnehmern anderes **Lernerlebnis**

> Schulungsumgebungen müssen **einheitlich installiert** sein

> Bereitstellung von Schulungsumgebungen oftmals mit Hürden behaftet

MOTIVATION

> Corona-**Pandemie** macht **Vor-Ort**-Veranstaltungen **schwierig bis unmöglich**

> **Hands-On**-Übungen ermöglichen Teilnehmern anderes **Lernerlebnis**

> Schulungsumgebungen müssen **einheitlich installiert** sein

> Bereitstellung von Schulungsumgebungen oftmals mit Hürden behaftet

> **Gemeinsame** Erarbeitung von Schulungsinhalten ermöglichen

> **Know-How** auf mehrere Trainer **verteilen**

// TOOLING

ANFORDERUNGEN

> Einfach zu benutzen, **geringe Hürde** für neue Trainer

> **Keine** besonderen Tools zur Verwendung/Erstellung

ANFORDERUNGEN

> Einfach zu benutzen, **geringe Hürde** für neue Trainer

> **Keine** besonderen Tools zur Verwendung/Erstellung

> Themes für Corporate Identity

> Definition im Klartext, keine proprietären Formate

ANFORDERUNGEN

> Einfach zu benutzen, **geringe Hürde** für neue Trainer

> **Keine** besonderen Tools zur Verwendung/Erstellung

> Themes für Corporate Identity

> Definition im Klartext, keine proprietären Formate

> muss **offline** ohne Server benutzbar sein

> Präsentatoransicht mit Timer

ANFORDERUNGEN

> Einfach zu benutzen, **geringe Hürde** für neue Trainer

> **Keine** besonderen Tools zur Verwendung/Erstellung

> Themes für Corporate Identity

> Definition im Klartext, keine proprietären Formate

> muss **offline** ohne Server benutzbar sein

> Präsentatoransicht mit Timer

> Syntax-**Highlightning** für verschiedene Sprachen

> **PDF**-Export

KLASSISCHE PRÄSENTATIONSPROGRAMME

- Benutzen bereits **vorhandener** Vorlagen mit
  - Microsoft PowerPoint
  - LibreOffice Impress
  - FreeOffice / SoftMaker Office Presentations

KLASSISCHE PRÄSENTATIONSPROGRAMME

- Benutzen bereits **vorhandener** Vorlagen mit
  - Microsoft PowerPoint
  - LibreOffice Impress
  - FreeOffice / SoftMaker Office Presentations
- **Einfache** Umsetzung, bekannter Workflow, Präsentatoransicht
- Änderungen in Versionskontrolle **intransparent** da Binärdatei*
- Große Probleme mit 200+ Folien und **hochauflösenden** Fotos

KLASSISCHE PRÄSENTATIONSPROGRAMME

> Benutzen bereits **vorhandener** Vorlagen mit
  > Microsoft PowerPoint
  > LibreOffice Impress
  > FreeOffice / SoftMaker Office Presentations

> **Einfache** Umsetzung, bekannter Workflow, Präsentatoransicht

> Änderungen in Versionskontrolle **intransparent** da Binärdatei*

> Große Probleme mit 200+ Folien und **hochauflösenden** Fotos

> *Das kannste schon so machen, aber...*

*PPTX = zip + XML

TEXTSATZSYSTEME

> Verwendung von **TeX** bzw. **LaTeX**

> Sehr **mächtig** und vielseitig

TEXTSATZSYSTEME

> Verwendung von **TeX** bzw. **LaTeX**

> Sehr **mächtig** und vielseitig

> Viele Präsentationsvorlagen online verfügbar

> vollständige TeX-Umgebung benötigt (*4 GB+*)

TEXTSATZSYSTEME

> Verwendung von **TeX** bzw. **LaTeX**

> Sehr **mächtig** und vielseitig

> Viele Präsentationsvorlagen online verfügbar

> vollständige TeX-Umgebung benötigt (*4 GB+*)

> spezieller Viewer für Präsentatoransicht benötigt*

> steile Lernkurve, definitiv **ungeeignet** für Einsteiger

*z.B. pdfpc, BeamerPresenter

TEXTSATZSYSTEME

> Verwendung von **TeX** bzw. **LaTeX**

> Sehr **mächtig** und vielseitig

> Viele Präsentationsvorlagen online verfügbar

> vollständige TeX-Umgebung benötigt (*4 GB+*)

> spezieller Viewer für Präsentatoransicht benötigt*

> steile Lernkurve, definitiv **ungeeignet** für Einsteiger

> *Wie mache ich mein Leben spannender?*

*z.B. pdfpc, BeamerPresenter

WEB-FRAMEWORKS

Zu den gängigsten Frameworks* zählen u.a.:

> Prezi

> reveal.js, Hacker Slides

> flowtime.js, deck.js

> DZSlides

> remark

> marp

* neues Framework in 3, 2,...

REMARK

> HTML-Präsentator auf Markdown-Basis

> **simpler** Aufbau, einfach anpassbar

REMARK

> HTML-Präsentator auf Markdown-Basis

> **simpler** Aufbau, einfach anpassbar

> Präsentatoransicht mit Timer

> Syntax-Highlightning für verschiedene Sprachen

REMARK

> HTML-Präsentator auf Markdown-Basis

> **simpler** Aufbau, einfach anpassbar

> Präsentatoransicht mit Timer

> Syntax-Highlightning für verschiedene Sprachen

> PDF-Export via DeckTape

> **geringe Einarbeitungszeit** durch Markdown

REMARK

```
# Digitale Schulungen mit GitLab CI / CD
---
### remark

- HTML-Präsentator auf Markdown-Basis
- **simpler** Aufbau, einfach anpassbar
--

- Präsentatoransicht mit Timer
- Syntax-Highlightning für verschiedene Sprachen
--

- PDF-Export via [DeckTape](https://github.com/astefanutti/decktape)
- **geringe Einarbeitungszeit** durch Markdown
```

Zwei Folien dieser Präsentation

ZUSAMMENARBEIT

> Versionierung der Ausgangsdateien in **git**-Repository

ZUSAMMENARBEIT

> Versionierung der Ausgangsdateien in **git**-Repository

> Werkzeug der Wahl: **GitLab**

> Ermöglicht **Kollaboration** ohne große Hürden

> *GitLab Web IDE* ermöglicht *Markdown Live Preview*

ZUSAMMENARBEIT

> Versionierung der Ausgangsdateien in **git**-Repository

> Werkzeug der Wahl: **GitLab**

> Ermöglicht **Kollaboration** ohne große Hürden

> *GitLab Web IDE* ermöglicht *Markdown Live Preview*

> Mittels **CI** automatisierte Erstellung fertiger Präsentationen

> Arbeitsorganisation durch Issues & Labels

// AUTOMATISIERTE INFRASTRUKTUR

AUTOMATISIERTE INFRASTRUKTUR

- Schulungen bestehen i.d.R. aus einer **Testumgebung**
  - verschiedene VMs
  - IP-Adressen, Netzwerke
  - vorinstalliertes **Betriebssystem**
  - **vorkonfigurierte** Applikationen

AUTOMATISIERTE INFRASTRUKTUR

- Schulungen bestehen i.d.R. aus einer **Testumgebung**
  - verschiedene VMs
  - IP-Adressen, Netzwerke
  - vorinstalliertes **Betriebssystem**
  - **vorkonfigurierte** Applikationen
- Teilnehmer erwarten **sofort** loslegen zu können
- Spätere Experimente sollen jedoch auch möglich sein

AUTOMATISIERTE INFRASTRUKTUR

> Testumgebungen kosten Geld

> Definition von **Verfügbarkeit** oftmals nicht eindeutig

AUTOMATISIERTE INFRASTRUKTUR

> Testumgebungen kosten Geld

> Definition von **Verfügbarkeit** oftmals nicht eindeutig

> **Interne** Schulungen auf eigener Infrastruktur

> Zugriff für **externe** Kunden nicht möglich und erwünscht

AUTOMATISIERTE INFRASTRUKTUR

> Testumgebungen kosten Geld

> Definition von **Verfügbarkeit** oftmals nicht eindeutig

> **Interne** Schulungen auf eigener Infrastruktur

> Zugriff für **externe** Kunden nicht möglich und erwünscht

> Schulungsumgebungen mittelfristig für Teilnehmer aufrecht erhalten ist **unpraktikabel** und teuer

> Wo das Ganze hosten?

AUTOMATISIERTE INFRASTRUKTUR

> Testumgebungen kosten Geld

> Definition von **Verfügbarkeit** oftmals nicht eindeutig

> **Interne** Schulungen auf eigener Infrastruktur

> Zugriff für **externe** Kunden nicht möglich und erwünscht

> Schulungsumgebungen mittelfristig für Teilnehmer aufrecht erhalten ist **unpraktikabel** und teuer

> Wo das Ganze hosten?

> **Lösung**: $Cloud + Terraform

INFRASTRUCTURE AS CODE

> Schulungsumgebung wird **automatisiert** erstellt

> tatsächliche Plattform ist **irrelevant**
  > lokale Testumgebung
  > Cloud-Anbieter
  > Teilnehmer-Notebook

INFRASTRUCTURE AS CODE

- Schulungsumgebung wird **automatisiert** erstellt
- tatsächliche Plattform ist **irrelevant**
  - lokale Testumgebung
  - Cloud-Anbieter
  - Teilnehmer-Notebook
- Definition des **Soll-Zustands** der benötigten Resourcen
  - VMs, Netzwerke, OS, Applikation,...
- **Ansible** kümmert sich um die Implementation

HASHICORP TERRAFORM

> Terraform verwaltet komplexe (*produktive*) Infrastruktur

> Sinnvoll für mehrere unterschiedliche Plattformen
  > z.B. mehrere On-Premise Hypervisor + Cloud

HASHICORP TERRAFORM

> Terraform verwaltet komplexe (*produktive*) Infrastruktur

> Sinnvoll für mehrere unterschiedliche Plattformen
  > z.B. mehrere On-Premise Hypervisor + Cloud

> Infrastruktur-Definition in **HCL** (*HashiCorp Configuration Language*) oder JSON

> Sehr große Anzahl verfügbarer Integrationen*

HASHICORP TERRAFORM

> Terraform verwaltet komplexe (*produktive*) Infrastruktur

> Sinnvoll für mehrere unterschiedliche Plattformen
  > z.B. mehrere On-Premise Hypervisor + Cloud

> Infrastruktur-Definition in **HCL** (*HashiCorp Configuration Language*) oder JSON

> Sehr große Anzahl verfügbarer Integrationen*

> Integration in gängiges Configuration Management

> Prinzip: **D.R.Y** (*Don't repeat yourself*)

*ca. 100 Erweiterungen

HASHICORP TERRAFORM

> **Cloud**: AWS, Azure, GCE, vCloud, DigitalOcean, Hetzner, 1&1, Nutanix, OpenStack

> **Infrastruktur**: Docker, k8s, Helm, Rancher, Vault, Chef, Cobbler

> **Netzwerk**: Cisco, FortiOS, Palo Alto, F5 BIG-IP, Power/UltraDNS, DNSimple

> **Datenbanken**: MySQL, PostgreSQL, InfluxDB, Grafana

> **Monitoring**: Icinga2, Datadog, Runscope

> **Community**: Actice Directory, Elasticsearch, GitHub, GitLab, Infoblox, JIRA, Proxmox, oVirt

> ...

// TESTING FÜR TRAINER UND TEILNEHMER

TESTING

> Schulungen bestehen aus Aufgaben um Interaktivität zu fördern

> Teilnehmer sollen **selbstständig** Richtigkeit überprüfen

TESTING

> Schulungen bestehen aus Aufgaben um Interaktivität zu fördern
    > Teilnehmer sollen **selbstständig** Richtigkeit überprüfen

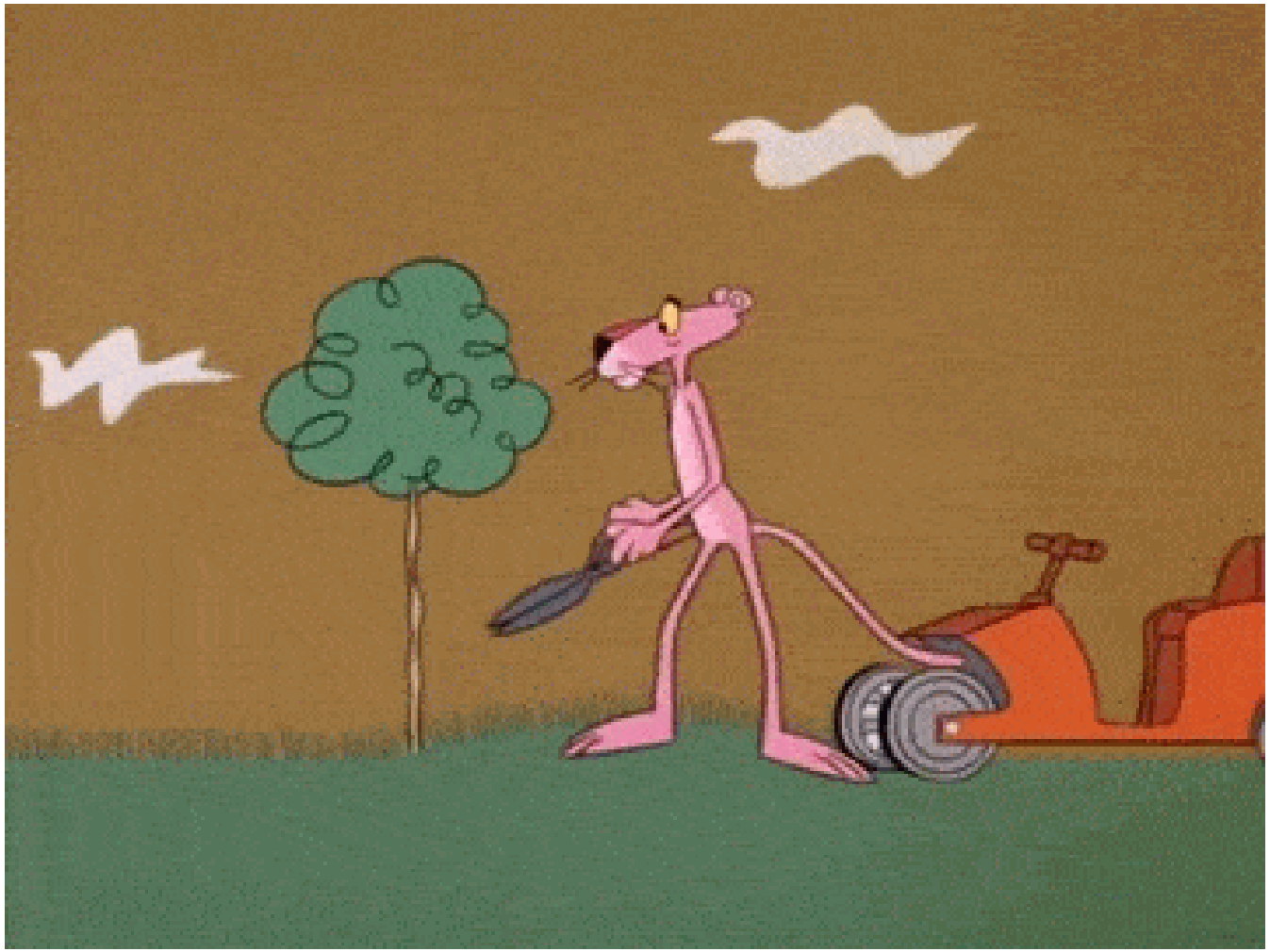
> Cloud-Schulungsumgebungen **müssen** getestet werden
    > Wartungsarbeiten des Anbieters, neue Features/**Fehler**

TESTING

> Schulungen bestehen aus Aufgaben um Interaktivität zu fördern
    > Teilnehmer sollen **selbstständig** Richtigkeit überprüfen

> Cloud-Schulungsumgebungen **müssen** getestet werden
    > Wartungsarbeiten des Anbieters, neue Features/**Fehler**

> Trainer fügt neue Inhalte und **Musterlösungen** hinzu
    > **Syntax**-Fehler sollen vermieden werden



FRAMEWORK

Denkbare Frameworks:

> Deployment:
  > WIP: `terratest` für Terraform-Deployment

FRAMEWORK

Denkbare Frameworks:

- Deployment:
  - WIP: `terratest` für Terraform-Deployment
- Betriebssystem / Anwendung
  - Ansible `assert`
  - Testinfra

FRAMEWORK

Denkbare Frameworks:

- Deployment:
  - WIP: `terratest` für Terraform-Deployment
- Betriebssystem / Anwendung
  - Ansible `assert`
  - Testinfra
- Musterlösungen / Aufgaben
  - Goss
  - Commander

TESTINFRA

```
def test_software(host):
    """
    Check whether required software packages are installed
    """
    packages = [
        "vim-common",
        "bind-utils",
        "epel-release",
        "ansible"
    ]
    for pkg in packages:
        _pkg = host.package(pkg)
        assert _pkg.is_installed
```

Unittest überprüft installierte Software-Pakete

COMMANDER

> In Go geschriebenes Testing-Tool

> **Unabhängig** von verwenderter Skript-/Programmiersprache

COMMANDER

- In Go geschriebenes Testing-Tool
- **Unabhängig** von verwenderter Skript-/Programmiersprache
- Tests werden in **YAML** definiert
- Unterstützt verschiedene Ziele
  - localhost
  - SSH-Hosts
  - Docker-Container

COMMANDER

- In Go geschriebenes Testing-Tool
- **Unabhängig** von verwenderter Skript-/Programmiersprache
- Tests werden in **YAML** definiert
- Unterstützt verschiedene Ziele
  - localhost
  - SSH-Hosts
  - Docker-Container
- schnell zu schreiben und zu verstehen

COMMANDER: BEISPIEL

```
tests:
  lab01:
    command: git status
    stdout: "is up to date"
    exit-code: 0
```

`keen.yml` - Definition eines Tests

```
$ commander test keen.yml
Starting test file keen.yaml...

✓ lab01

Duration: 0.002s
Count: 1, Failed: 0
```

Ausführen des Tests

LINTING

> Mithilfe eines **Lint**-Tools (*englisch für Fussel*) kann eine **statische Code-Analyse** durchgeführt werden

LINTING

> Mithilfe eines **Lint**-Tools (*englisch für Fussel*) kann eine **statische Code-Analyse** durchgeführt werden

> Tool überprüft Code auf **Syntax**- und **Design-Fehler**, z.B.:

LINTING

> Mithilfe eines **Lint**-Tools (*englisch für Fussel*) kann eine **statische Code-Analyse** durchgeführt werden

> Tool überprüft Code auf **Syntax**- und **Design-Fehler**, z.B.:
  > veraltete Formate und Funktionen
  > schlechter Stil
  > fehlerhafte Aufrufe

LINTING

> Mithilfe eines **Lint**-Tools (*englisch für Fussel*) kann eine **statische Code-Analyse** durchgeführt werden

> Tool überprüft Code auf **Syntax**- und **Design-Fehler**, z.B.:
  > veraltete Formate und Funktionen
  > schlechter Stil
  > fehlerhafte Aufrufe

> Die ersten Linter entstanden 1979 um Schwächen des C-Compilers auszugleichen

LINTING

> Mithilfe eines **Lint**-Tools (*englisch für Fussel*) kann eine **statische Code-Analyse** durchgeführt werden

> Tool überprüft Code auf **Syntax**- und **Design-Fehler**, z.B.:
  > veraltete Formate und Funktionen
  > schlechter Stil
  > fehlerhafte Aufrufe

> Die ersten Linter entstanden 1979 um Schwächen des C-Compilers auszugleichen

> **Resultat**: optimierter und fehlerfreier Code

LINTING AS A SERVICE

Für Schulungsinhalte relevant:

- `markdownlint` für Markdown-Code (sic!)
- `ansible-lint` für Playbooks/Rollen
- `yamllint` für YAML-Format
- `flake8` für `testinfra`-Tests
- `pylint` für Python-Code
- `TFLint` für Terraform-Code
- `rubocop` für Ruby-/Vagrant-Code

LINTING: SCOPE

> Ziel soll es sein den Trainer auf **Best Practices** hinzuweisen

> **Fehler** brechen die Pipeline, Warnungen nicht

> Nicht jeder strebt perfekten Code an (*leider*)

LINTING: SCOPE

> Ziel soll es sein den Trainer auf **Best Practices** hinzuweisen
> **Fehler** brechen die Pipeline, Warnungen nicht
> Nicht jeder strebt perfekten Code an (*leider*)
> Mustervorlage ist fehlerfrei
> Dokumentation erklärt typische Fehler
> VScode-/VScodium-**Plugins** für Linting verfügbar
  > vscode-markdownlint
  > vscode-yaml
  > vscode-ruby

README.md | 02-platforms.md ● | 01-motivation.md | 03-tooling.md | 04-infrastructure.md | 05-testing.md | 06-pipeline.md | 07-

02-platforms.md > ...

```
1  class: center, middle
2  MD018/no-missing-space-atx: No space after hash on atx style heading markdownlint(MD018)
3  Peek Problem (Alt+F8)  Quick Fix... (Ctrl+.)
4  #Meeting-Plattformen
5
6  ---
7
8  ## Wozu eine Meeting-Plattform?
9
10 - Spätestens seit Corona **essentiell** um Termine wahrzunehmen
11 - Auch ein guter **Anlass** um Schulungskonzept zu überdenken:
12 --
13
14   - Reisekosten und **Zeitersparnis**
15   - Logistische Planung (*Räume, Verpflegung*)
16   - Schulungsmethoden (*Tools, Testumgebung*)
17 --
18
```

// GITLAB CI

AUTOMATISIERUNG MIT PIPELINES

PIPELINE

> Pipeline **kombiniert** drei vorher unabhängige **Teilschritte**
  > Erstellen von Schulungsunterlagen
  > Testing (*Slides, Labs*)
  > Infrastruktur bereitstellen (*und testen*)
> MVP* erfolgt, weitere Schritte in Arbeit

*Minimum Viable Product

PDF-EXPORT

> Mit DeckTape existiert ein Werkzeug zum *Drucken* von HTML-Präsentationen

> Unterstützt zahlreiche Frameworks, u.a. *remark*

> Geschrieben in JavaScript, Installation via `npm`

PDF-EXPORT

> Mit DeckTape existiert ein Werkzeug zum *Drucken* von HTML-Präsentationen

> Unterstützt zahlreiche Frameworks, u.a. *remark*

> Geschrieben in JavaScript, Installation via `npm`

> Glücklicherweise gibt es ein vorgefertigtes Docker-Image

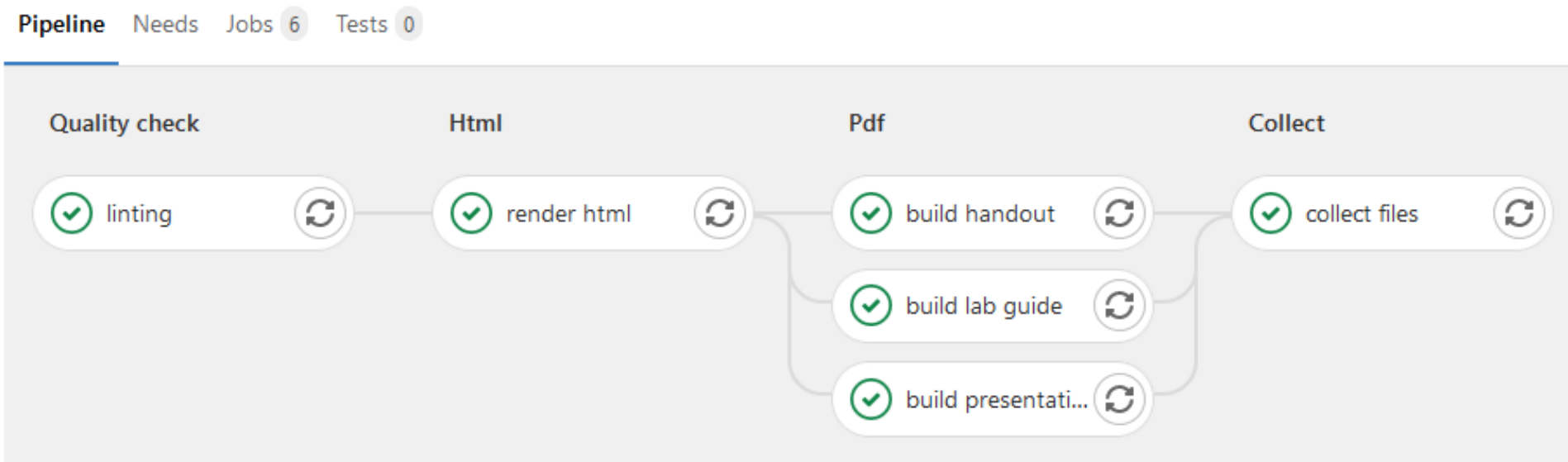
> einfache Integration in CI-/CD-Pipeline möglich

PDF-EXPORT

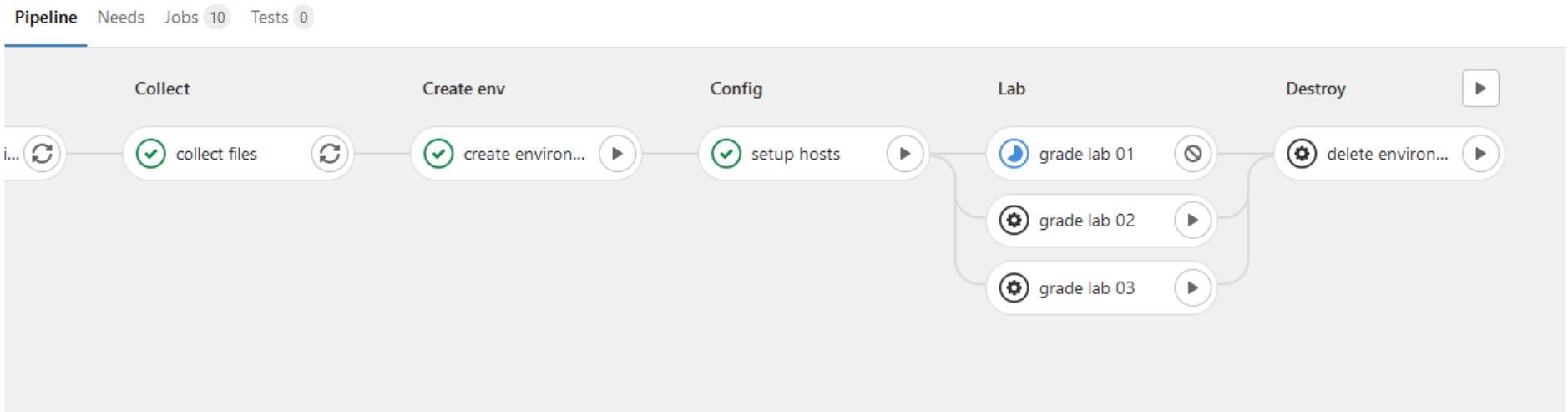
```
build presentation:
  stage: pdf
  image:
    name: astefanutti/decktape
    entrypoint: [""]
  script:
   - node /decktape/decktape.js --chrome-path chromium-browser --c
  artifacts:
    paths:
      - presentation.pdf
```

`.gitlab-ci.yml` - Teil der CI-/CD-Konfiguration für PDF-Export

PIPELINE: IST-ZUSTAND



PIPELINE: SOLL-ZUSTAND



// LESSONS LEARNED

LESSONS LEARNED

> **Zusammenarbeit** in **GitLab** funktioniert sehr gut

> **Aufteilen der Präsentation** in viele Dateien sinnvoll

> **Markdown für alle** Skill-Level nutzbar

> Fokus wird auf **Inhalte** gelenkt anstatt auf Design

LESSONS LEARNED

> **Zusammenarbeit** in **GitLab** funktioniert sehr gut

> **Aufteilen der Präsentation** in viele Dateien sinnvoll

> **Markdown für alle** Skill-Level nutzbar

> Fokus wird auf **Inhalte** gelenkt anstatt auf Design

> Wiederverwendung von **Pipeline-Templates** ermöglicht schnellen Start

> Verwendung unterschiedlicher Werkzeuge in GitLab CI sehr mächtig

LESSONS LEARNED

> Nur bedingt geeignet für **Bild-lastige** Präsentationen

> Rendern großer Präsentationen **dauert**

> Bilder-Scaling nicht sehr intuitiv

> *marp* bietet bessere Integration in VSCode (u.A. live Preview)

// FAZIT

FAZIT

> remark eignet sich **hervorragend** als Präsentationswerkzeug

> GitLab macht Kollaboration einfach

> Automatisierung mit GitLab sehr mächtig

> In Kombination mit einer Meeting-Plattform und Testumgebung auch für Schulungen geeignet

> MVP wird bereits genutzt, Feedback **positiv**

CONTRIBUTE!

Ein Schulungsbeispiel findet sich auf GitHub:

github.com/svalabs/training-as-code

Feedback ist jederzeit willkommen!

FRAGEN?

DANKE FÜR DIE AUFMERKSAMKEIT