

JAX Remote, 05.05.2021

Infrastructure as Code - Muss man nicht
testen, Hauptsache es läuft

Sandra Parsick

@SandraParsick
mail@sandra-parsick.de

Wer bin ich?

- Sandra Parsick
- Freiberuflicher Softwareentwickler und Consultant im Java-Umfeld
- Schwerpunkte:
 - Java Enterprise Anwendungen
 - Agile Methoden
 - Software Craftmanship
 - Automatisierung von Entwicklungsprozessen
- Trainings
- Workshops

✉ mail@sandra-parsick.de

🐦 @SandraParsick

✕ xing.to/sparsick

📡 https://www.sandra-parsick.de

📻 https://ready-for-review.de

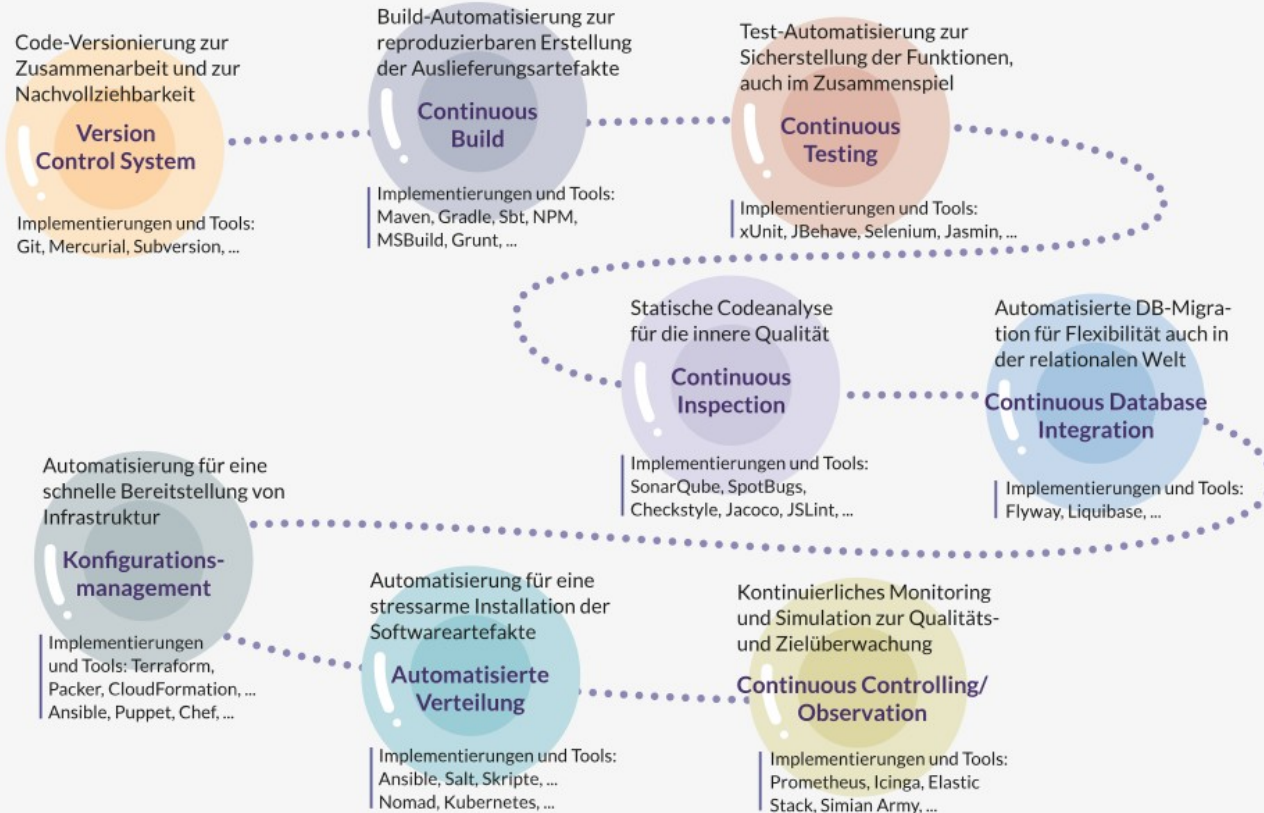


*„Du muss nicht jede Erfahrung selber machen, es
kommt dir günstiger, wenn du aus Fehler, die
Andere schon gemacht haben, lernst.“*

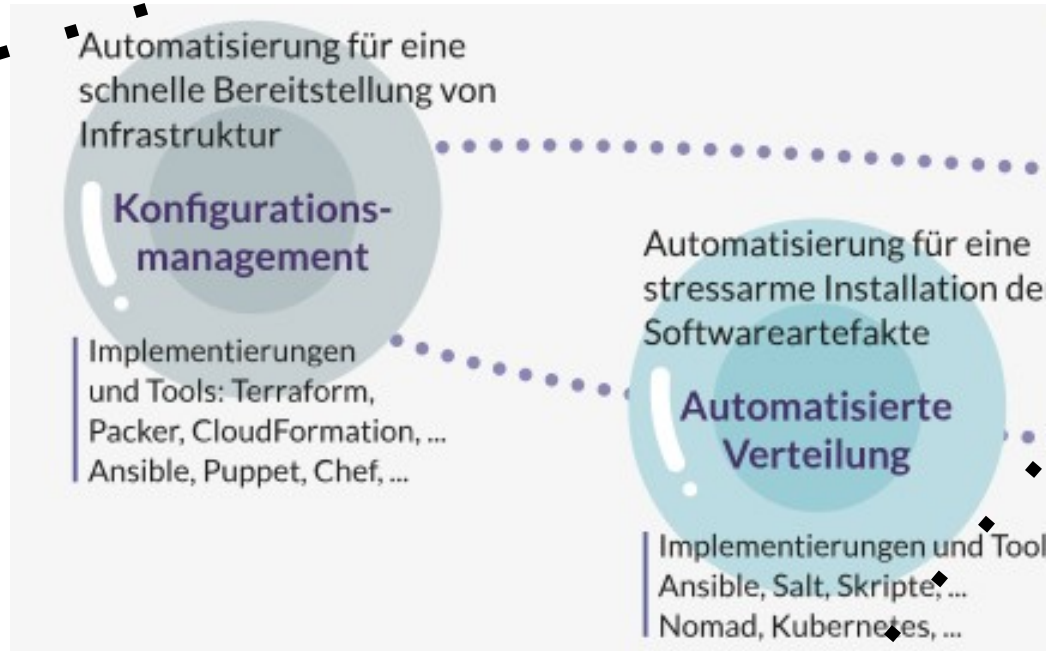
Mein Vater

Was hat das mit Infrastructure As Code zu tun?

Continuous Delivery

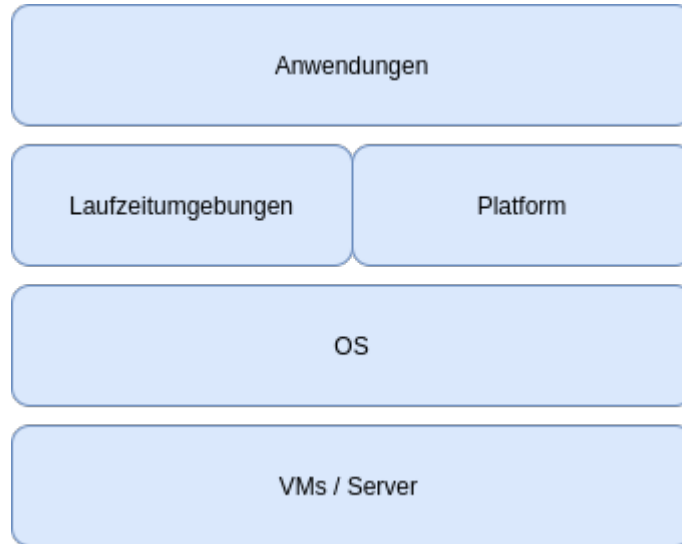


Infrastructure As Code

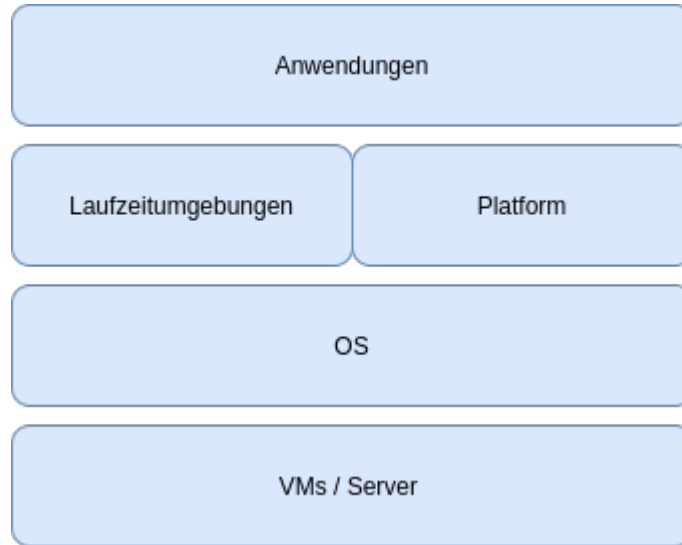
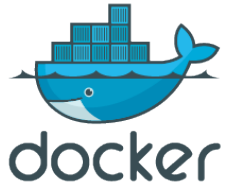


In der klassischen IT:
Ops-Tätigkeiten

Infrastruktur-Stack



Infrastructure as Code (IaC)




```
hosts: application_server
```

```
vars:
```

```
tomcat_version: 9.0.27
```

```
tomcat_base_name: apache-tomcat-{{ tomcat_version }}
```

```
tasks:
```

```
- name: install java
```

```
apt:
```

```
name: openjdk-8-jdk
```

```
state: present
```

```
update_cache: yes
```

```
become: yes
```

```
become_method: sudo
```

```
- name: Setup Group tomcat
```

```
group:
```

```
name: tomcat
```

```
become: yes
```

```
become_method: sudo
```

```
- name: Setup User tomcat
```

```
user:
```

```
name: tomcat
```

```
group: tomcat
```

```
become: yes
```

```
become_method: sudo
```



```
FROM adoptopenjdk
```

```
WORKDIR applicat
```

```
COPY *.jar appli
```

```
RUN java -Djarmo
```

```
FROM gcr.io/dist
```

```
WORKDIR applicat
```

```
EXPOSE 8080
```

```
COPY --from=buil
```

```
COPY --from=buil
```

```
COPY --from=buil
```

```
COPY --from=buil
```

```
ENTRYPOINT ["jav
```

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
```

```
name: {{ include "spring-boot-demo.fullname" . }}
```

```
namespace: {{ include "spring-boot-demo.namespaceName" . }}
```

```
labels:
```

```
{{- include "spring-boot-demo.labels" . | nindent 4 }}
```

```
spec:
```

```
{{- if not false }}
```

```
replicas: 1
```

```
{{- end }}
```

```
selector:
```

```
matchLabels:
```

```
{{- include "spring-boot-demo.selectorLabels" . | nindent 6 }}
```

```
template:
```

```
metadata:
```

```
annotations:
```

```
seccomp.security.alpha.kubernetes.io/pod: runtime/default
```

```
labels:
```

```
{{- include "spring-boot-demo.selectorLabels" . | nindent 8 }}
```

```
spec:
```

```
{{- with .Values.imagePullSecrets }}
```

```
imagePullSecrets:
```

```
{{- toYaml . | nindent 8 }}
```

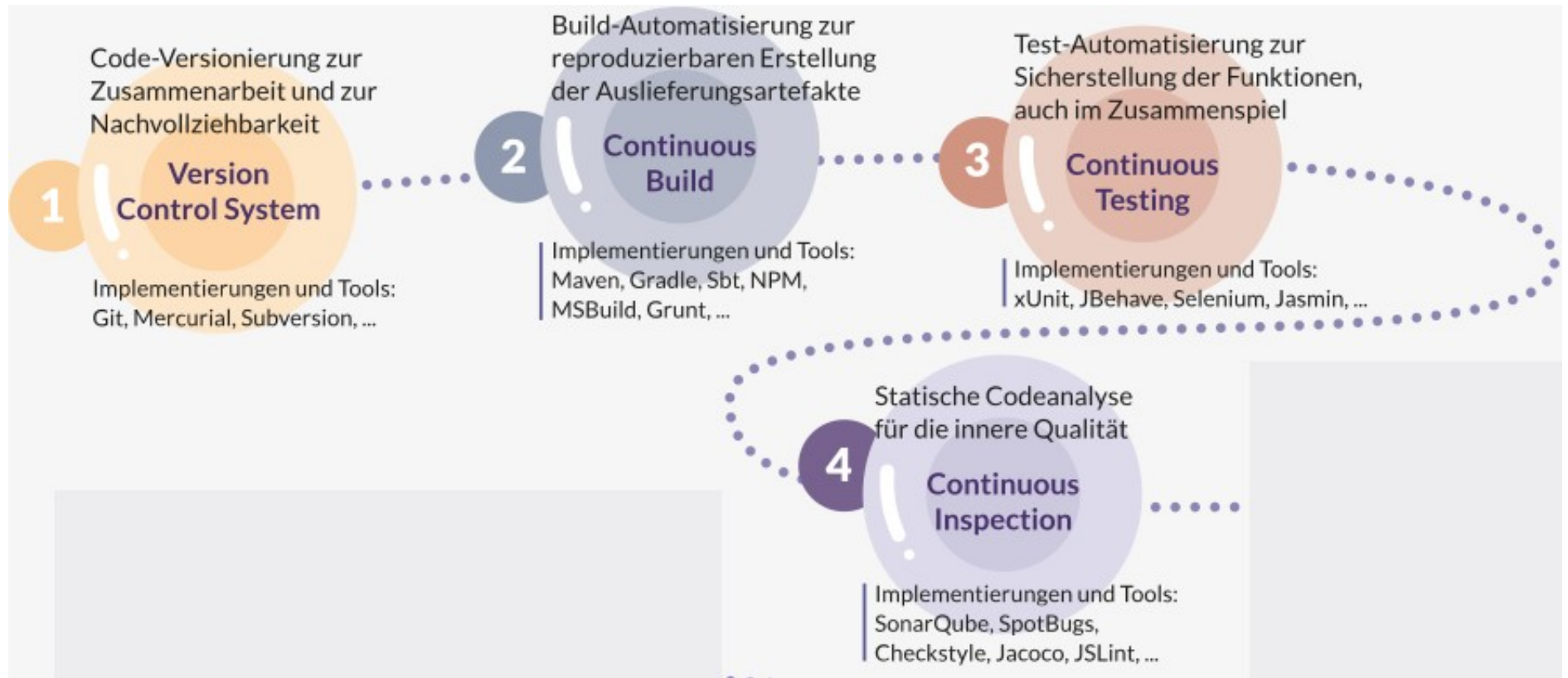
```
{{- end }}
```



Konsequenz für die Ops-Abteilung

- Ops brauchen ein Grundverständnis für das Programmieren (Algorithmen und Datenstrukturen)
- Ops stoßen auf ähnliche Probleme, die Devs auch hatten und für die sie Lösungen gefunden haben
- Ops brauchen auch einen **Entwicklungsprozess**

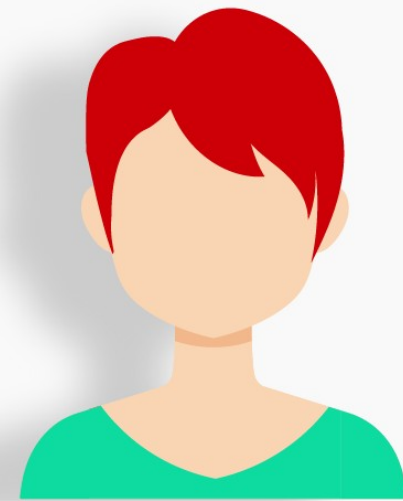
Wie sieht der Entwicklungsprozess
bei den Devs aus,
wenn sie nur mit ihren Code zu tun haben?



Was können die Ops aus diesem Prozess für sich übernehmen?



IaC Skripte liegen
direkt auf dem Server.



Sie funktionieren
gerade nicht.



Keine Ahnung,
wer sie geändert hat

Version Control System

Code-Versionierung zur
Zusammenarbeit und zur
Nachvollziehbarkeit

Version
Control System

1

Implementierungen und Tools:
Git, Mercurial, Subversion, ...

1

Versionskontrollsystem


Ein Versionskontrollsystem (VCS) dient als Basis jeder CI- (und später CD-) Umgebung. Mit ihm kehrt ein Team bei Bedarf (z.B. im Fehlerfall) zuverlässig auf einen früheren, funktionierenden Stand zurück.

BEST PRACTICES

- Das Projekt im VCS besitzt eine konsistente Ordnerstruktur.
- Source Code ist an einer zentralen/offiziellen Stelle abgelegt.
- Zum Source Code gehören: Build-Skripte, Programm-Code, Konfigurationen (zu Beginn, später im Konfigurationsmanagement), Deployment-Skripte, Datenbank-Skripte

SO ARBEITEN EURE TEAMS

- Entwickler checken Code häufig ein.
- Teams entscheiden sich für einen VCS Workflow und passen ihre Arbeitsweise daran an.
- Jedes Team ist für seinen Quelltext gemeinsam verantwortlich (Collective Ownership).

 master ▾

 3 branches  0 tags

Go to file

Add file ▾

 Code ▾

Gitpod



sparsick update fro architektur-punsch

756a7ec on 17 Dec 2020  5 commits



samples

update fro architektur-punsch

2 months ago



slides

update fro architektur-punsch

2 months ago



.gitignore

init ansible scripts

6 months ago



LICENSE

Initial commit

6 months ago



README.md

update fro architektur-punsch

2 months ago

 master ▾

Gitpod

🔗 Commits on Feb 8, 2021

add terratest for helm charts testing

 **sparsick** committed 10 minutes ago



fc80cef



add docker sample

 **sparsick** committed 1 hour ago



9b01334



add helm charts sample

 **sparsick** committed 2 hours ago



69b9a1b



mv ansible scripts

 **sparsick** committed 2 hours ago



9350287



🔗 Commits on Dec 17, 2020

update fro architektur-punsch

 **sparsick** committed on 17 Dec 2020



756a7ec



🔗 Commits on Sep 4, 2020

add slides

 **sparsick** committed on 4 Sep 2020



abf5f76




🔗 Commits on Aug 23, 2020



Die Skripte in VCS
laufen nicht durch.



Bei mir aber.



Letzten Mal hatten sie
sogar
Syntaxfehler.

Continuous Build



2 Continuous Build

Der erste Automatisierungsschritt etabliert einen kontinuierlichen Build-Lauf. Jede Änderung im VCS zieht das Bauen nach sich. So bleibt die Software durchgängig mindestens kompilierfähig.



BEST PRACTICES

- Keine manuellen Schritte – Builds laufen automatisiert ab.
- Das Abhängigkeitsmanagement erfolgt über das Buildwerkzeug.
- Das Resultat eines Build sind fertige, umgebungsunabhängige Deployment-Artefakte.
- Alleinige Wahrheit über den Zustand des Builds ist eine dedizierte Build Integration Maschine (CI-Server) – kein „It works on my machine.“

SO ARBEITEN EURE TEAMS

- Sie fixen einen fehlgeschlagenen Build auf dem Server unverzüglich.
- Entwickler lassen den Build auch auf ihrem lokalen Rechner laufen.

Continuous Build für Ops

- IaC Skripte können lokal ausgeführt werden.
- Beispiel:
 - Vagrant + Virtualbox / Docker 
 - minikube 

Demo Vagrant + Virtualbox

Demo Minikube

Alternativen zu Minikube

- k3s
- k3d
- kind
- microk8s
- k0s

Continuous Build für Ops

- Lokale Arbeitsrechner der Ops müssen dafür geeignet sein.
 - Zu oft müssen Ops wie auch Devs Office-Rechner benutzen, die nicht genügend Performance haben.
 - Falsche OS
 - Keine Admin-Rechte auf den Arbeitsrechner

**YES, I'M LOOKING
ON YOU, MANAGEMENT**



Continuous Build für Ops

- IaC Repository wird von CI Server auf Änderungen überwacht.

Pipeline IaC CI Pipeline




[Recent Changes](#)

Stage View

		Declarative: Checkout SCM	Ansible Lint Check	Ansible Playbook run with tests	Declarative: Post Actions
Average stage times:		267ms	1s	29s	2s
#5	Aug 23 17:12 No Changes	184ms	1s	23s failed	2s
#4	Aug 23 17:10 1 commit	245ms	1s	1min 1s	2s
#3	Aug 23 17:09 No Changes				
#2	Aug 23 16:57 No Changes	373ms	1s	2s failed	2s
#1	Aug 23 16:56 No Changes				

```
pipeline {
    agent any
    stages {
        stage('Ansible Lint Check') {
            steps {
                dir('samples') {
                    sh 'ansible-lint *.yaml'
                }
            }
        }
        stage('Ansible Playbook run with tests') {
            steps {
                dir('samples') {
                    sh 'vagrant up'
                    sh 'ansible-playbook -i inventories/test setup-tomcat.yaml'
                    sh 'py.test --connection=ansible --ansible-inventory inventory/test -v tests/*.py'
                }
            }
        }
    }
}


post {
    always {
        sh 'vagrant destroy -f'
    }
}
}
```



Die Skripte in VCS
funktionieren schon wieder
nicht wie erwartet.



Kann nicht sein.



Ups, dann ist wohl
beim Refactoring
was kaputt gegangen.

Continuous Testing

Test-Automatisierung zur
Sicherstellung der Funktionen,
auch im Zusammenspiel

3

Continuous
Testing

Implementierungen und Tools:
xUnit, JBehave, Selenium, Jasmin, ...

3

Continuous Testing

Automatisierte und regelmäßige Tests auf mehreren Ebenen verkürzen die Feedbackzeiten zum Zustand des Systems – über das bloße Bauen hinaus.

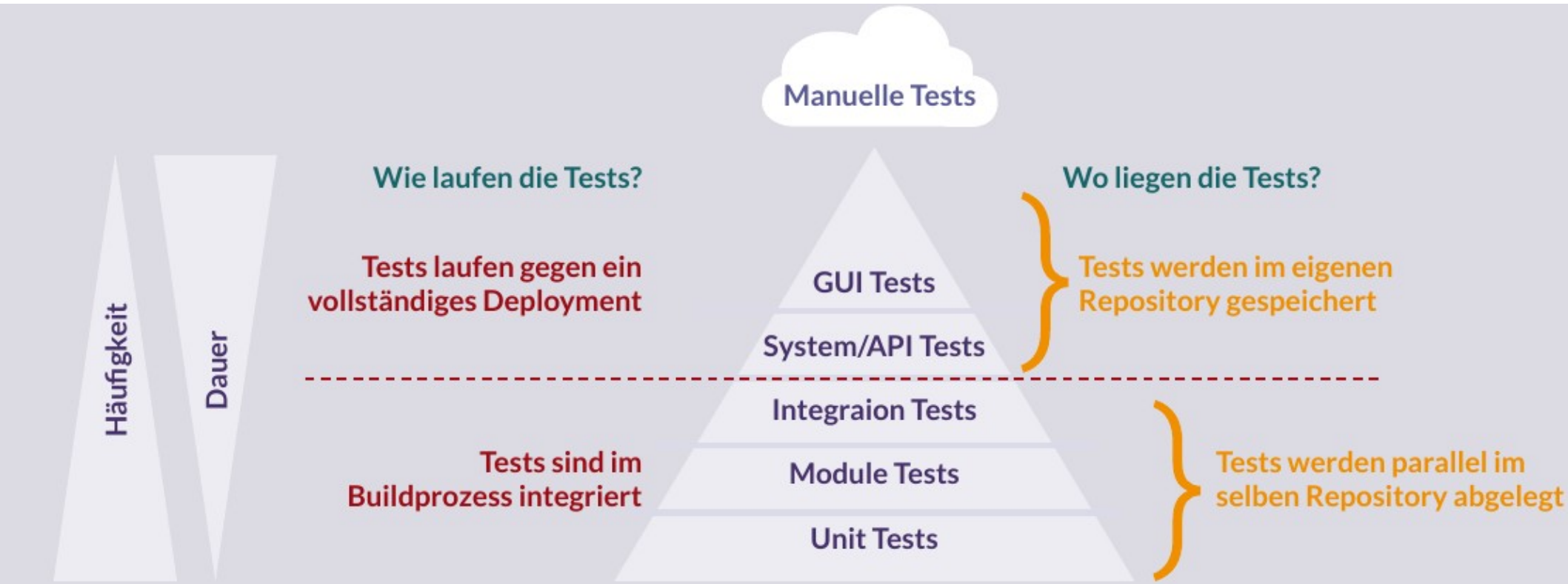
BEST PRACTICES

- Tests sind automatisiert, wiederholbar und laufen voneinander unabhängig.
- Tests liegen entweder in eigenen Repositories oder parallel zum Source Code.
- Tests sind kategorisiert (siehe Testpyramide).
- Für frühes Feedback laufen schnelle Tests zuerst.

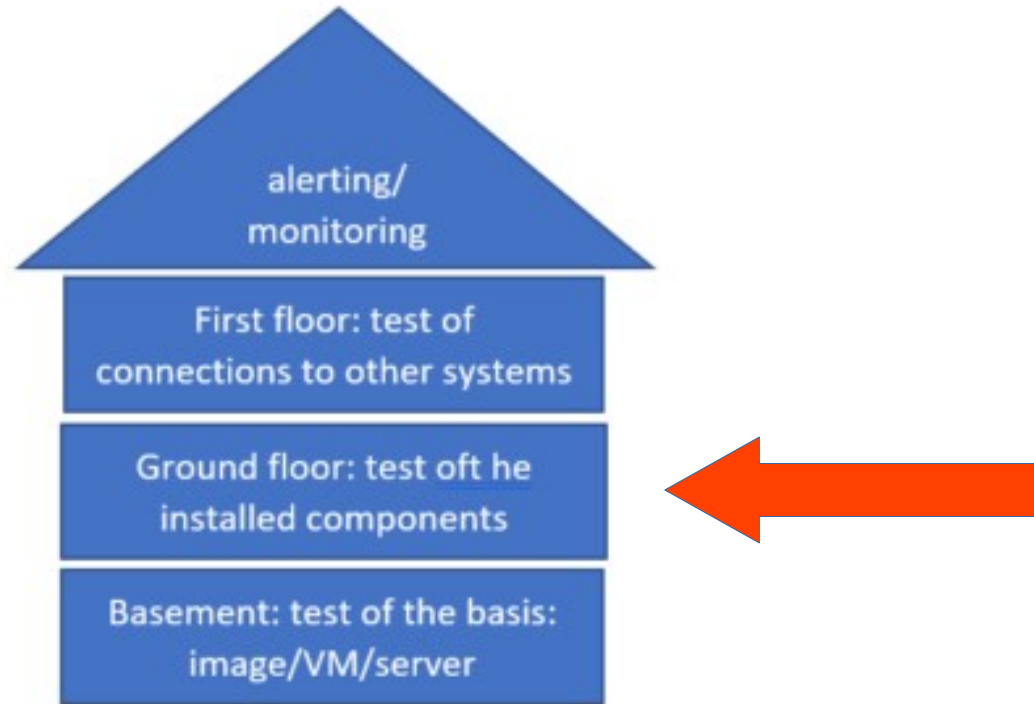
SO ARBEITEN EURE TEAMS

- Entwickler lassen die Tests auch auf ihren lokalen Rechnern laufen.
- Sie schreiben automatisierte Entwicklertests und integrieren sie in den Build.
- Teammitglieder checken keine Tests ein, die fehlschlagen.
- Die Teams dokumentieren auftretende Softwarefehler anhand von Tests.

Testpyramide



Infrastrukturtest-Haus





Continuous Testing für Ops



```
def test_openjdk_is_installed(host):
    openjdk = host.package("openjdk-8-jdk")
    assert openjdk.is_installed

def test_tomcat_catalina_script_exist(host):
    assert host.file("/opt/tomcat/bin/catalina.sh").exists
```



Continuous Testing für Ops



testinfra

```
~/dev/workspace/iac-qa-talk/samples(master x) py.test --connection=ansible --ansi
ble-inventory inventory/test -v tests/*.py -vv
===== test session starts =====
platform linux -- Python 3.8.2, pytest-5.4.3, py-1.9.0, pluggy-0.13.1 -- /usr/bin
/python3
cachedir: .pytest_cache
rootdir: /home/sparsick/dev/workspace/iac-qa-talk/samples
plugins: testinfra-5.2.1
collected 2 items

tests/test_tomcat.py::test_openjdk_is_installed[ansible://192.168.33.10] PASSED [
 50%]
tests/test_tomcat.py::test_tomcat_catalina_script_exist[ansible://192.168.33.10]
PASSED [100%]

===== 2 passed in 3.91s =====
```



Continuous Testing für Ops

- ContainerStructureTests

```
schemaVersion: 2.0.0

fileExistenceTests:
  - name: 'application'
    path: '/application/'
    shouldExist: true

metadataTest:
  exposedPorts: ["8080"]
  workdir: "/application"
```



Continuous Testing für Ops

```
~/dev/workspace/iac-qa-talk/samples/docker(master x)  
container-structure-test test --image sparsick/spring  
-boot-demo:latest --config spring-boot-test.yaml
```

```
===== Test file: spring-boot-test.yaml =====
```

```
=== RUN: File Existence Test: application
```

```
--- PASS
```

```
duration: 0s
```

```
=== RUN: Metadata Test
```

```
--- PASS
```

```
duration: 0s
```

```
===== RESULTS =====
```

```
Passes: 2
```

```
Failures: 0
```

```
Duration: 0s
```

```
Total tests: 2
```

```
PASS
```



Continuous Testing für Ops



Terratest

```
package test

import ...

func TestPodDeploysContainerImageHelmTemplateEngine(t *testing.T) {
    // Path to the helm chart we will test
    helmChartPath := "../spring-boot-demo"

    options := &helm.Options{
        ValuesFiles: []string{"../local-values.yaml"},
        ExtraArgs: map[string][]string{"upgrade": {"-i"}},
    }

    helm.Upgrade(t, options, helmChartPath, releaseName: "spring-boot-demo-instance")

    status, _ := http_helper.HttpGet(t, url: "http://spring-boot-demo.local/hero", tlsConfig: nil)

    assert.Equal(t, expected: 200, status)
}
```



Continuous Testing für Ops



Terratest

```
~/dev/workspace/iac-qa-talk/samples/helm-charts/test(master x) go test . -v
=== RUN   TestPodDeploysContainerImageHelmTemplateEngine
TestPodDeploysContainerImageHelmTemplateEngine 2021-02-08T21:44:35+01:00 logger.go:66: Running command helm with args
[upgrade -i -f /home/sparsick/dev/workspace/iac-qa-talk/samples/helm-charts/local-values.yaml --install spring-boot-de
mo-instance /home/sparsick/dev/workspace/iac-qa-talk/samples/helm-charts/spring-boot-demo]
TestPodDeploysContainerImageHelmTemplateEngine 2021-02-08T21:44:35+01:00 logger.go:66: Release "spring-boot-demo-insta
nce" has been upgraded. Happy Helming!
TestPodDeploysContainerImageHelmTemplateEngine 2021-02-08T21:44:35+01:00 logger.go:66: NAME: spring-boot-demo-instance
TestPodDeploysContainerImageHelmTemplateEngine 2021-02-08T21:44:35+01:00 logger.go:66: LAST DEPLOYED: Mon Feb  8 21:44
:35 2021
TestPodDeploysContainerImageHelmTemplateEngine 2021-02-08T21:44:35+01:00 logger.go:66: NAMESPACE: default
TestPodDeploysContainerImageHelmTemplateEngine 2021-02-08T21:44:35+01:00 logger.go:66: STATUS: deployed
TestPodDeploysContainerImageHelmTemplateEngine 2021-02-08T21:44:35+01:00 logger.go:66: REVISION: 5
TestPodDeploysContainerImageHelmTemplateEngine 2021-02-08T21:44:35+01:00 http_helper.go:32: Making an HTTP GET call to
URL http://spring-boot-demo.local/hero
--- PASS: TestPodDeploysContainerImageHelmTemplateEngine (0.67s)
PASS
ok      test      (cached)
~/dev/workspace/iac-qa-talk/samples/helm-charts/test(master x) █
```

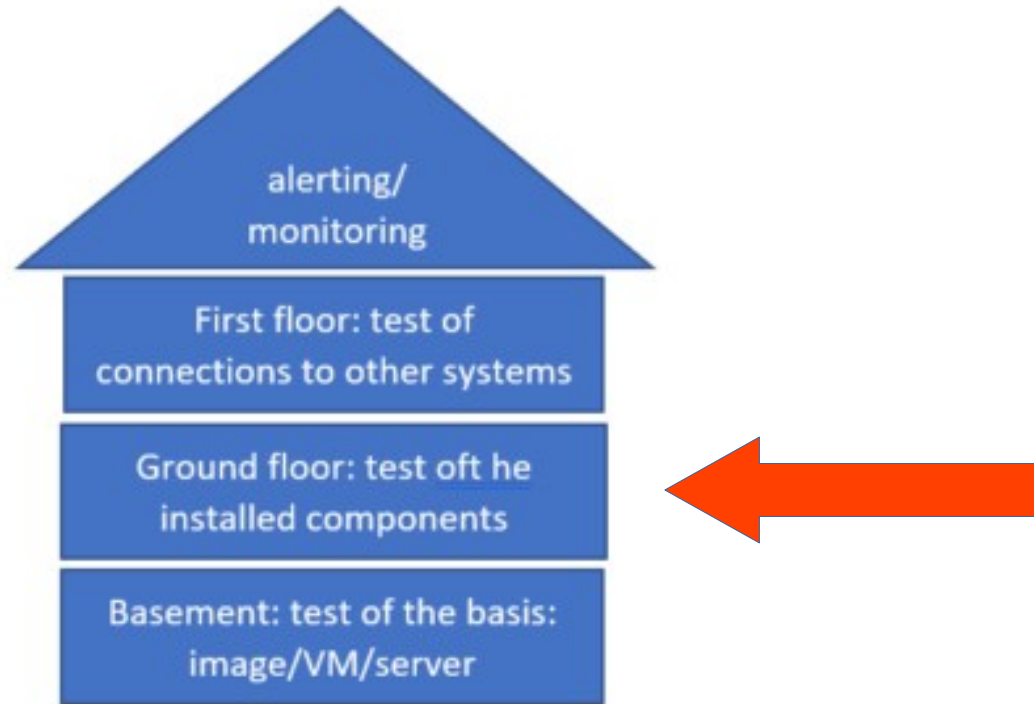
Continuous Testing für Ops


Weitere Testwerkzeuge



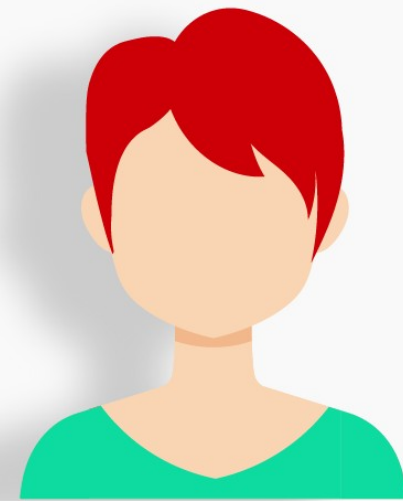
Goss

Infrastrukturtest-Haus






Die Skripte in VCS
folgen schon wieder
nicht unseren Style-Guide.

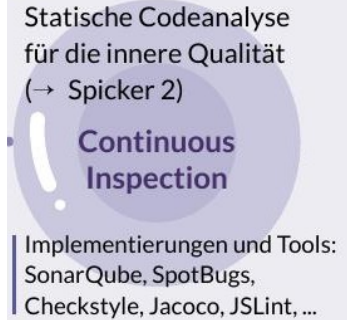


Wir haben eine
Style-Guide?



Ja, Good Practices werden
auch nicht eingehalten.

Continuous Inspection



- Automatisierte statische Codeanalyse

Best Practices

- Regelsatz orientiert sich an den Best Practices aus der Community
- Team einigt sich auf ein Regelsatz

So arbeitet das Team

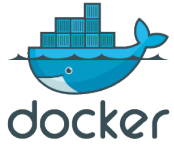
- Codeanalyse ist automatisiert und in den Build integriert.
- Verstöße gegen den Regelsatz werden sofort behoben.



Continuous Inspection für Ops

```
~/dev/workspace/iac-qa-talk/samples(master x) ansible-lint *.yaml
[502] All tasks should be named
setup-tomcat.yaml:34
Task/Handler: file name=/opt mode=511 owner=tomcat group=tomcat __line__=35 __file__=setup-tomcat.yaml

[502] All tasks should be named
setup-tomcat.yaml:63
Task/Handler: find paths=/opt/{{ tomcat_base_name }}/bin patterns=*.sh
```



Continuous Inspection für Ops

```
~/dev/workspace/iac-qa-talk/samples/docker(master x) hadolint Dockerfile
Dockerfile:2 DL3000 error: Use absolute WORKDIR
Dockerfile:7 DL3000 error: Use absolute WORKDIR
```



Continuous Inspection für Ops

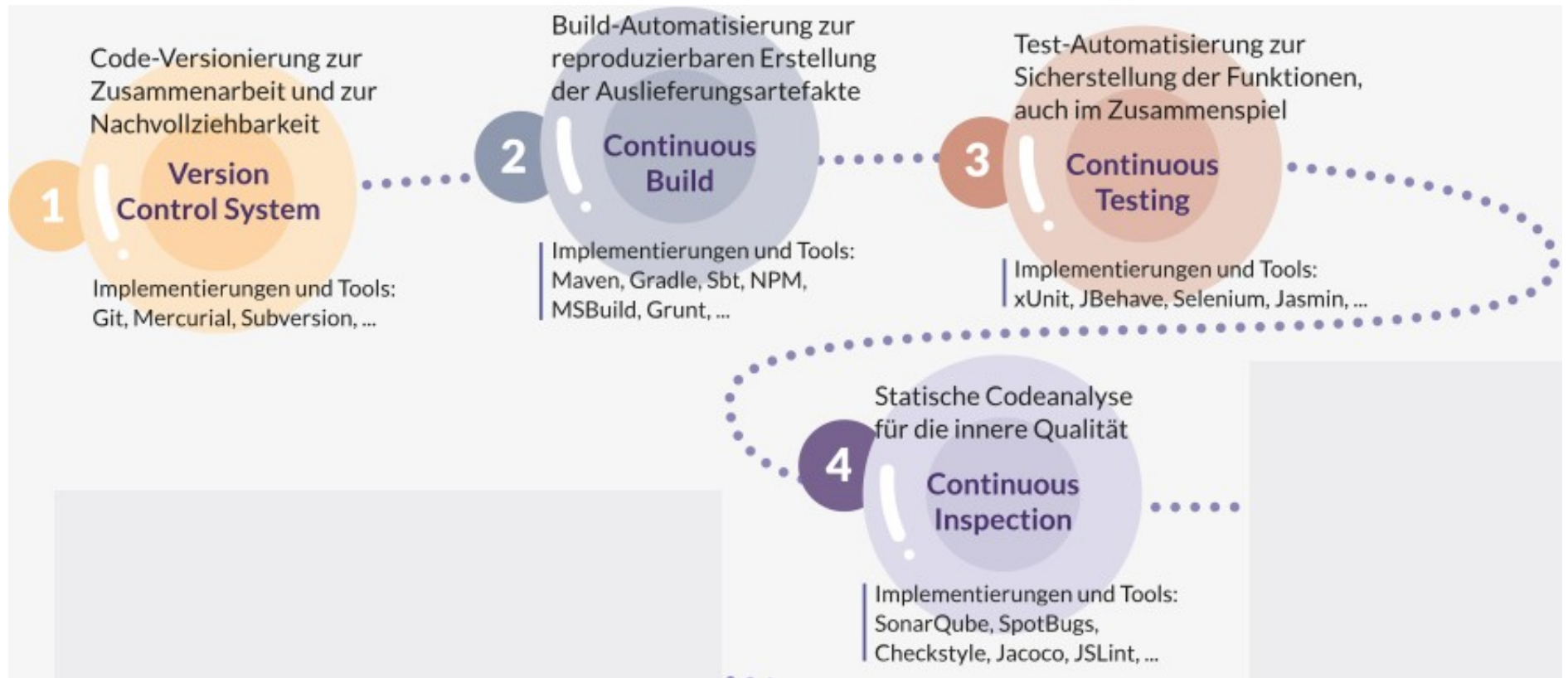
```
~/dev/workspace/iac-qa-talk/samples/helm-charts(master x)
helm lint spring-boot-demo -f local-values.yaml
==> Linting spring-boot-demo
[INFO] Chart.yaml: icon is recommended

1 chart(s) linted, 0 chart(s) failed
~/dev/workspace/iac-qa-talk/samples/helm-charts(master x)
█
```

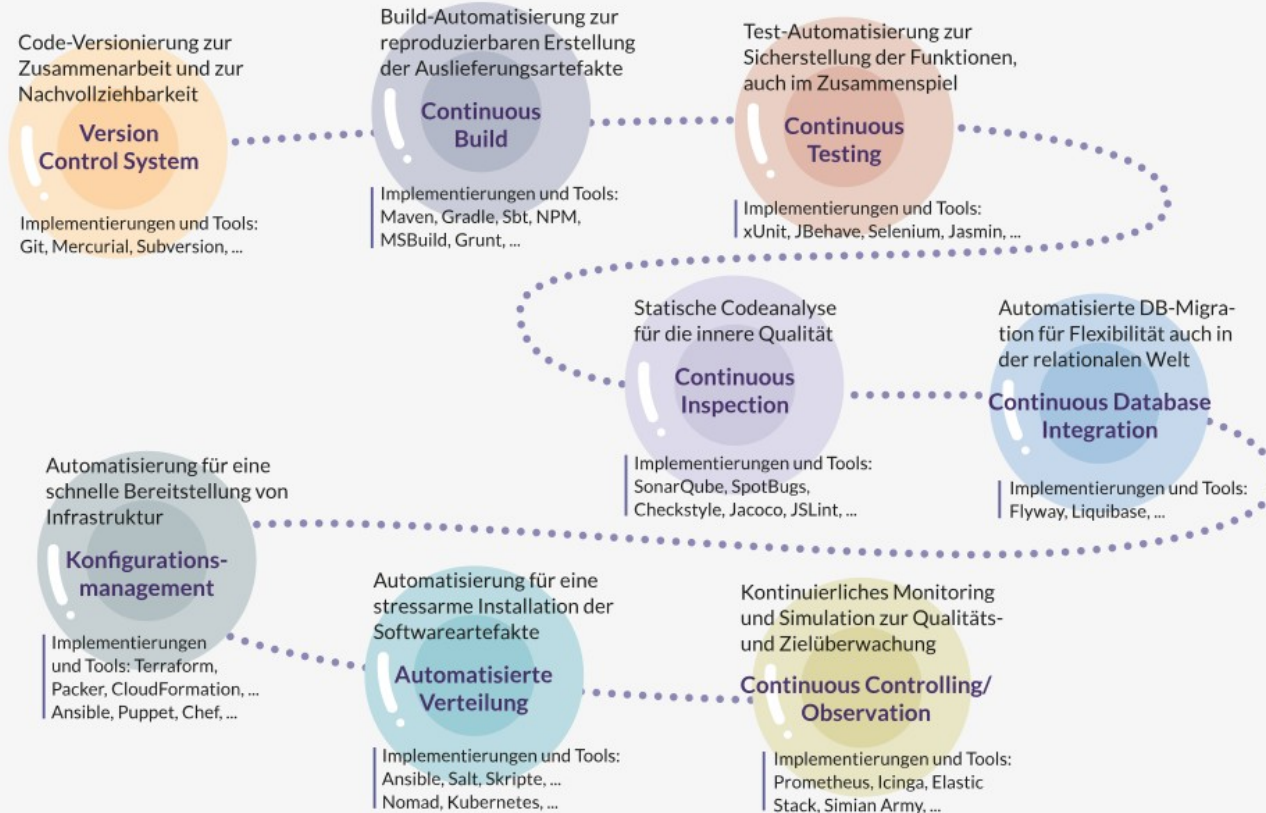
Continuous Inspection für Ops

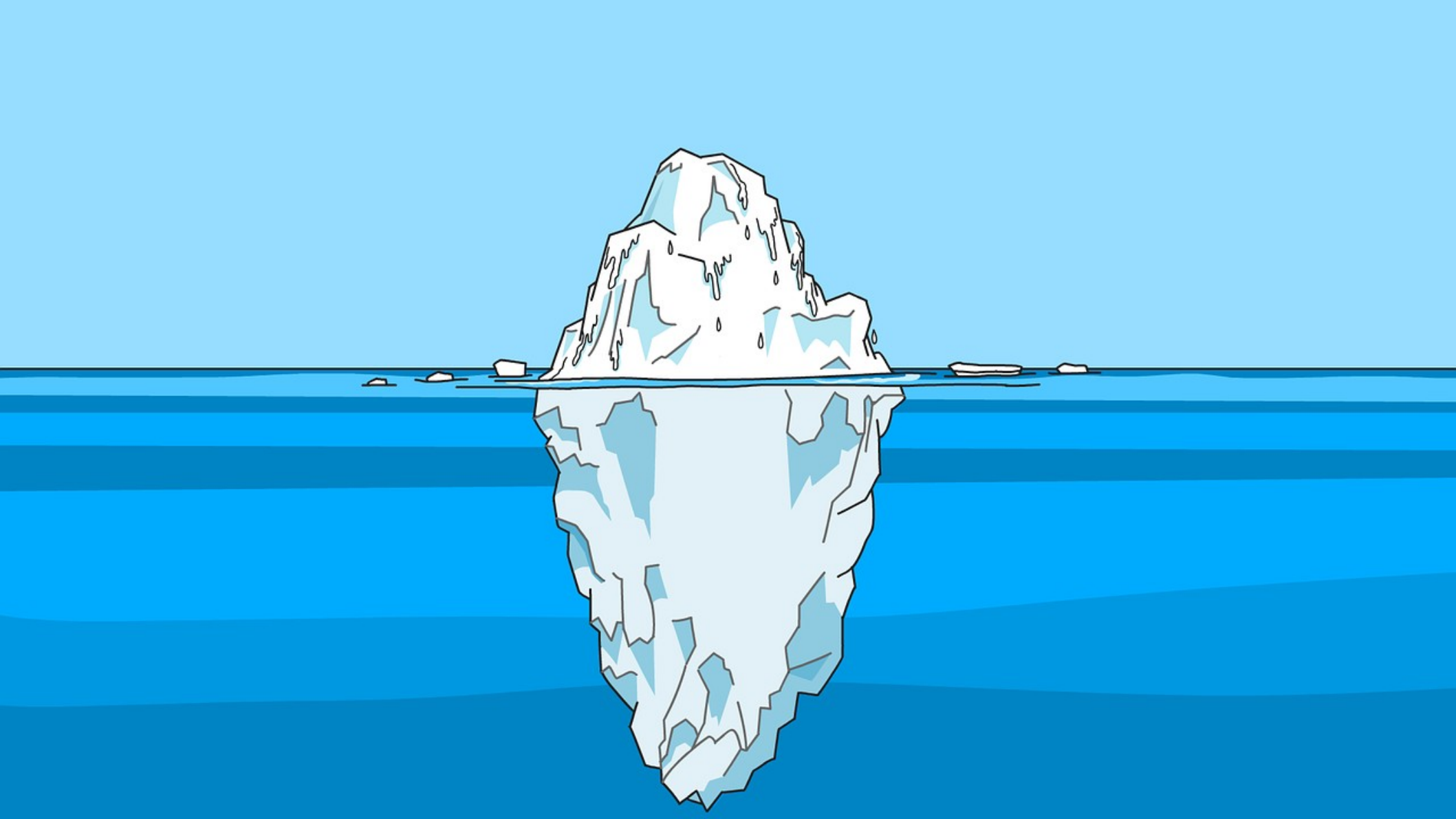
Weitere Linter

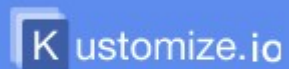
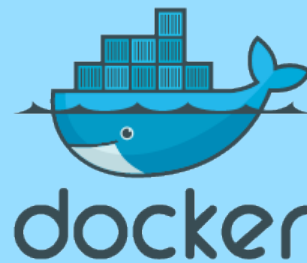
- Docker
 - dockerfile_lint
 - dockerlint
- Kubernetes
 - yamllint
 - kube-score
 - kubelint
- Ansible
 - yamllint



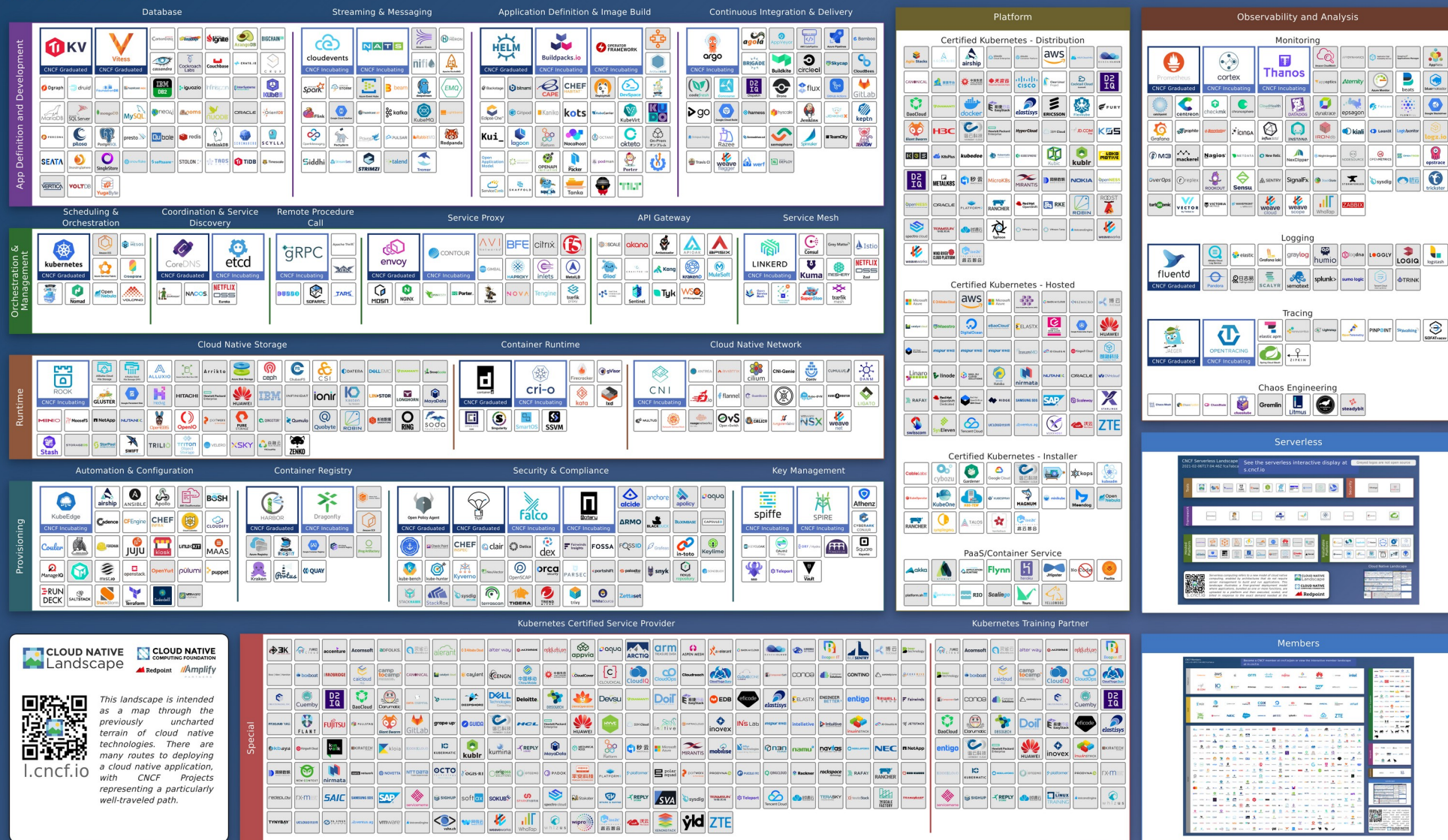
Continuous Delivery





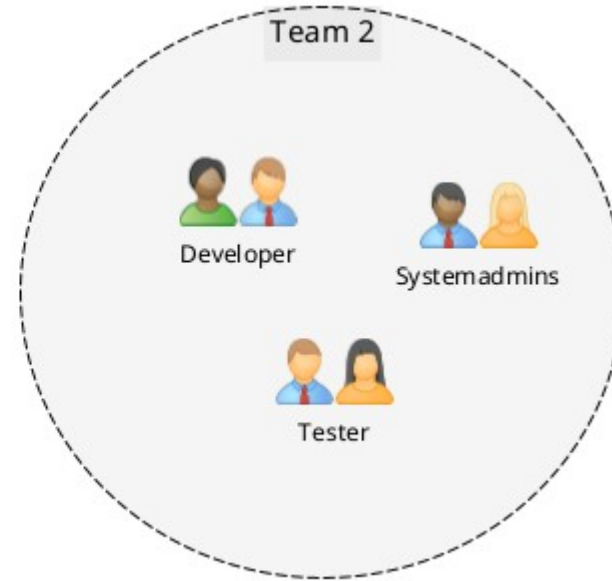
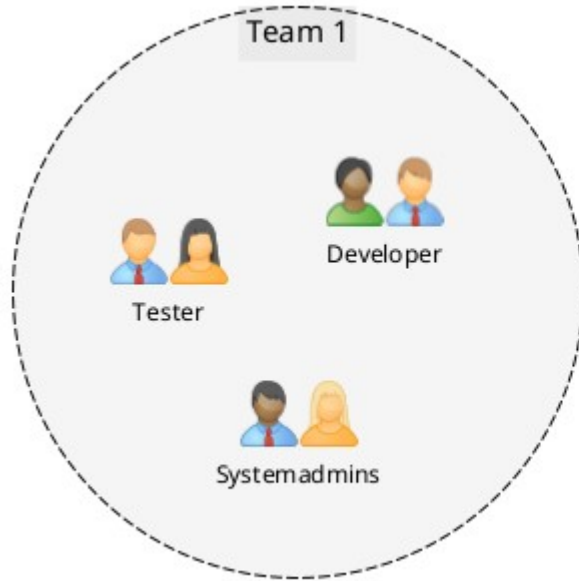


HashiCorp
Terraform

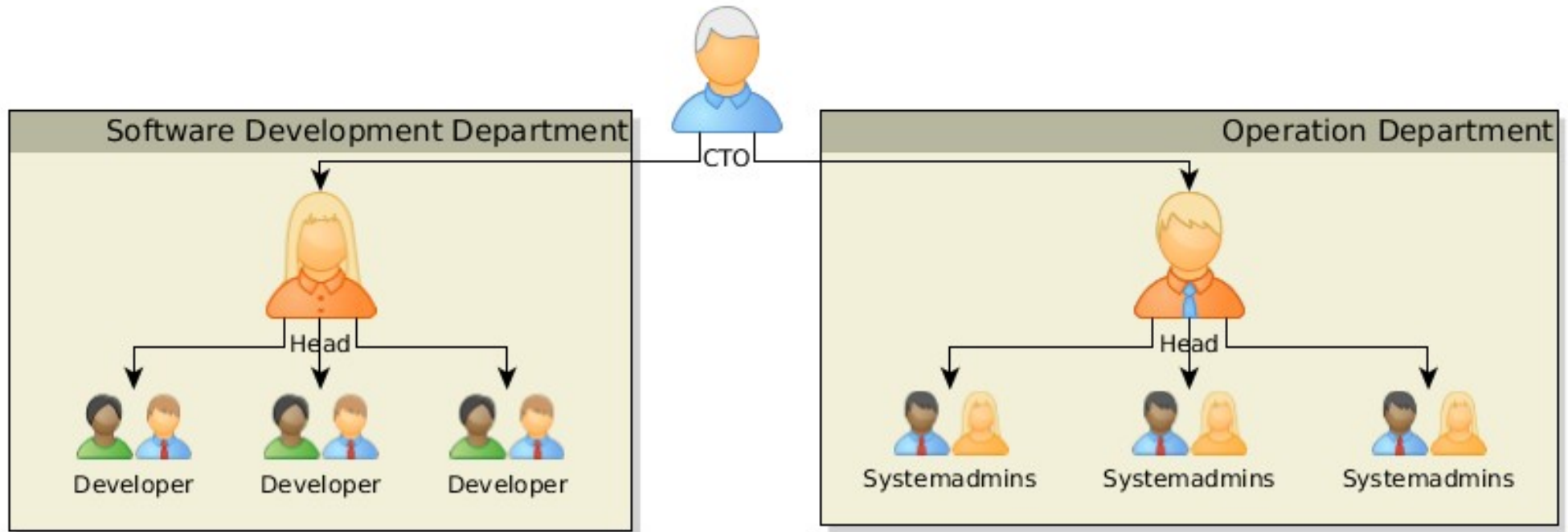


Diese technische Lösung und Überlegungen bringen nichts, wenn sich die Arbeitsweise der Organisation nicht ändert.

Idealbild



(Oft noch) Realität



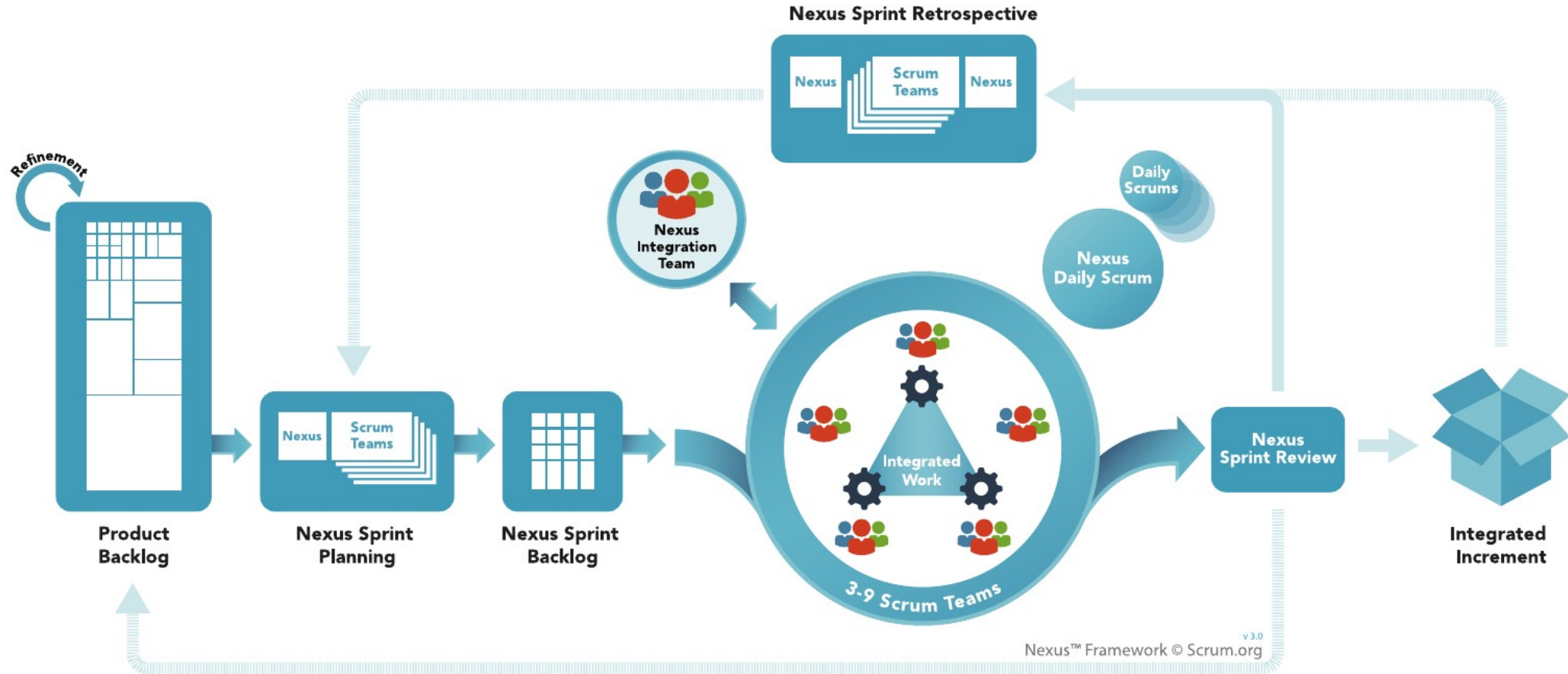
Herausforderungen Ops-Abteilung

- Ops brauchen ein Grundverständnis für das Programmieren (Algorithmen und Datenstrukturen)
- Ops stoßen auf ähnliche Probleme, die Devs auch hatten und für die sie Lösungen gefunden haben
- Ops brauchen auch einen Entwicklungsprozess

Evolution statt Revolution

- Schritt für Schritt Annäherung
- Beispiele:
 - Gemeinsame Repositories für Devs und Ops und mit Pull Requests und Reviews arbeiten
 - Developer und Operation pairen Aufgaben zusammen

NEXUS™ FRAMEWORK



BUSINESS AGILITY

MEASURE & GROW



Enterprise Solution Delivery

- Apply Lean system engineering to build really big systems
- Coordinate and align the full supply chain
- Continually evolve live systems

Lean System and
Solution EngineeringCoordinating Trains
and SuppliersContinually Evolve
Live Systems

Lean Portfolio Management

- Align strategy, funding, and execution
- Optimize operations across the portfolio
- Lightweight governance empowers decentralized decision-making



Agile Product Delivery

- The customer is the center of your product strategy
- Develop on cadence and release on demand
- Continuously explore, integrate, deploy, and innovate

Customer Centricity
& Design ThinkingDevelop on Cadence
Release on DemandDevOps and the Continuous
Delivery Pipeline

Organizational Agility

- Create an enterprise-wide, Lean-Agile mindset
- Lean out business operations
- Respond quickly to opportunities and threats

Lean-thinking People
and Agile TeamsLean Business
Operations

Strategy Agility

Continuous Learning Culture

- Everyone in the organization learns and grows together
- Exploration and creativity are part of the organization's DNA
- Continuously improving solutions, services, and processes is everyone's responsibility

Learning
OrganizationInnovation
CultureRelentless
Improvement

Team And Technical Agility

- High-performing, cross-functional, Agile teams
- Business and technical teams build business solutions
- Quality business solutions delight customers



Agile Teams

Teams of
Agile TeamsBuilt-In
Quality

Lean-Agile Leadership

- Inspire others by modeling desired behaviors
- Align mindset, words, and actions to Lean-Agile values and principles
- Actively lead the change and guide others to the new way of working



Leading by Example



Mindset & Principles



Leading Change

SAFe® for Lean Enterprises 5.0

Select Configuration

Overview

Essential SAFe

Large Solution SAFe

Portfolio SAFe

Full SAFe

Business Agility

Measure & Grow

Organizational Agility



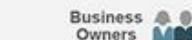
Lean Portfolio Management



Enterprise Solution Delivery



Agile Product Delivery



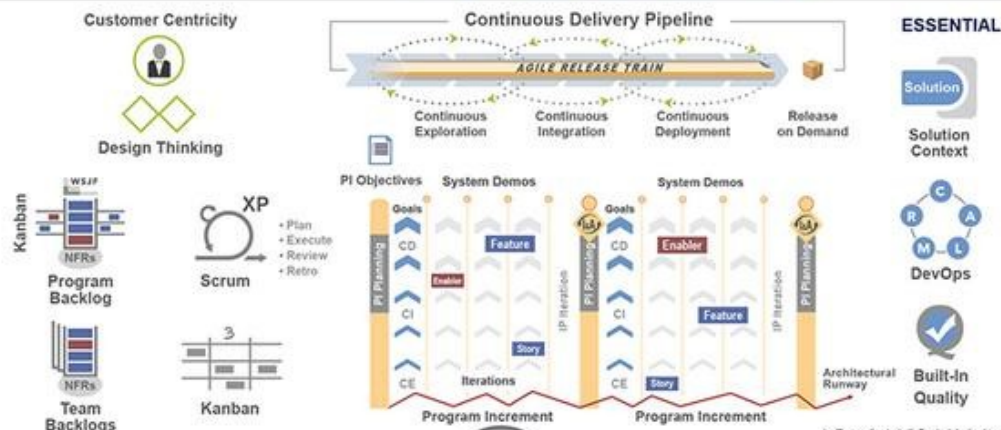
Team and Technical Agility



Continuous Learning Culture



Business | Dev | Ops | Support



Core Values

Lean-Agile Mindset

SAFe Principles



Implementation Roadmap

SAFe Program Consultant

5.0

Lean-Agile Leadership

Vision

Roadmap

Milestones

Shared Services

CoP

System Team

Lean UX

Metrics

Lettingwell, et al. © Scaled Agile, Inc.

Reality-Check

„Du muss nicht jede Erfahrung selber machen, es kommt dir günstiger, wenn du aus Fehler, die Andere schon gemacht haben, lernst.“

Mein Vater

Reality-Check

- Low Hanging Fruits
 - VCS benutzen
 - Lokale Entwicklungsumgebungen bereitstellen
 - Bei Änderungen im VCS immer die Skripte im CI-Server ausführen

Reality-Check

- Könnte Low Hanging Fruits sein
 - Einsatz von Linter
 - ähnliches Schicksal wie Sonarqube

Reality-Check

- Steile Lernkurve
 - Tests schreiben
 - wird oft vernachlässigt
 - Déjà-vu
 - Gründe:
 - Zeitdruck
 - Viel zu lernen (auch bei DEV Unterstützung)
 - neue Programmiersprachen (GO, Python, Ruby)
 - Programmieren an sich
 - Ähnliche Diskussion wie in der SWE: „Für diese einfachen Skripte brauchen wir keinen Tests“

Fazit

Fragen?

mail@sandra-parsick.de

@SandraParsick

<https://github.com/sparsick/iac-qa-talk>

Literatur

- Architektur Spicker 7 – Continuous Delivery
<https://www.sandra-parsick.de/publication/architektur-spicker-cd/>
- Scaling Scrum with Nexus
<https://www.scrum.org/resources/scaling-scrum>
- Scaled Agile Framework SAFe
<https://www.scaledagileframework.com/>
- Infrastructure as Code: Dynamic Systems for the Cloud Age von Kief Morris, O'Reilly
<https://www.oreilly.com/library/view/infrastructure-as-code/9781098114664/>