HackTalk, 07.06.2022

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

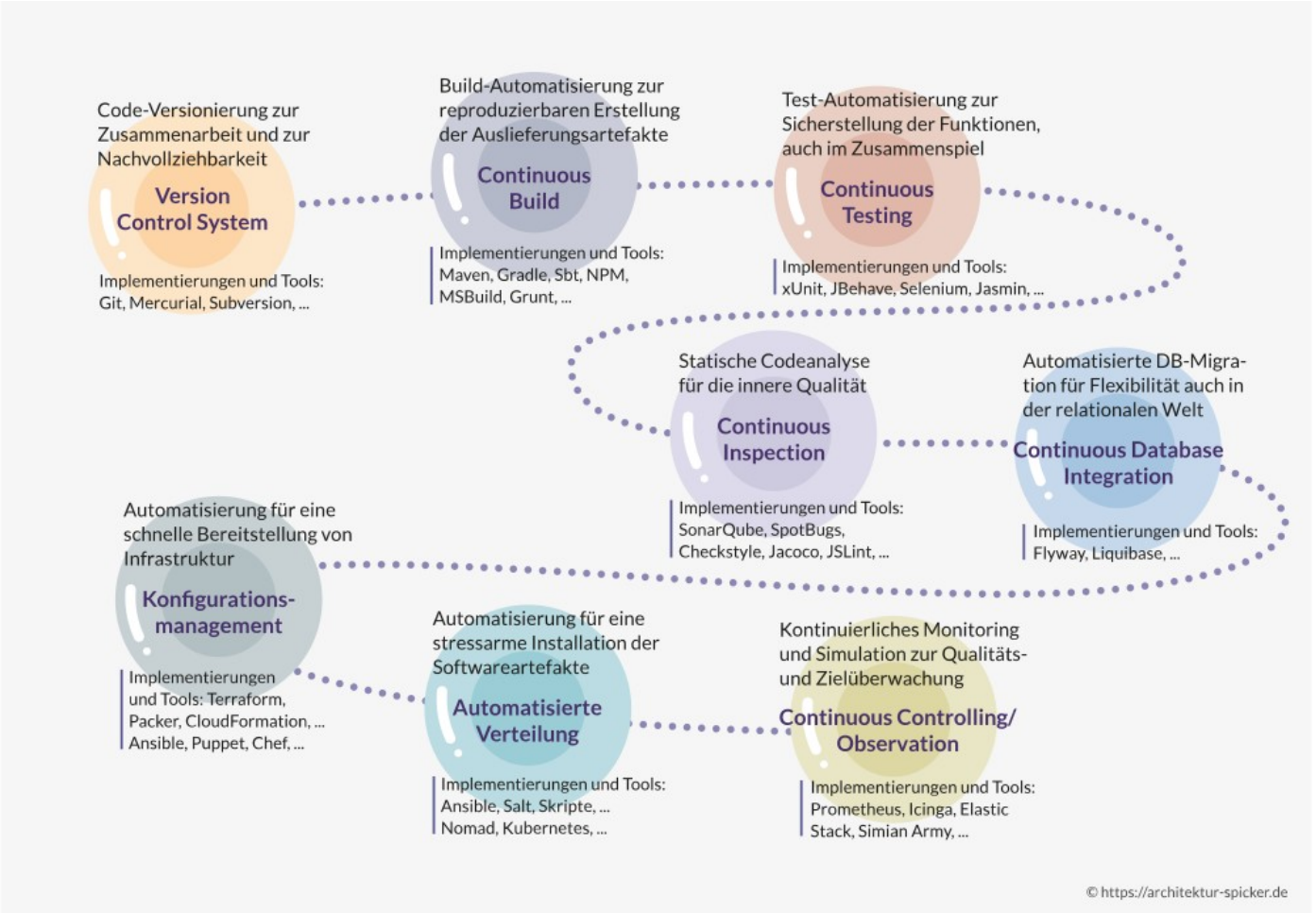
https://ready-for-review.dev

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

Mein Vater

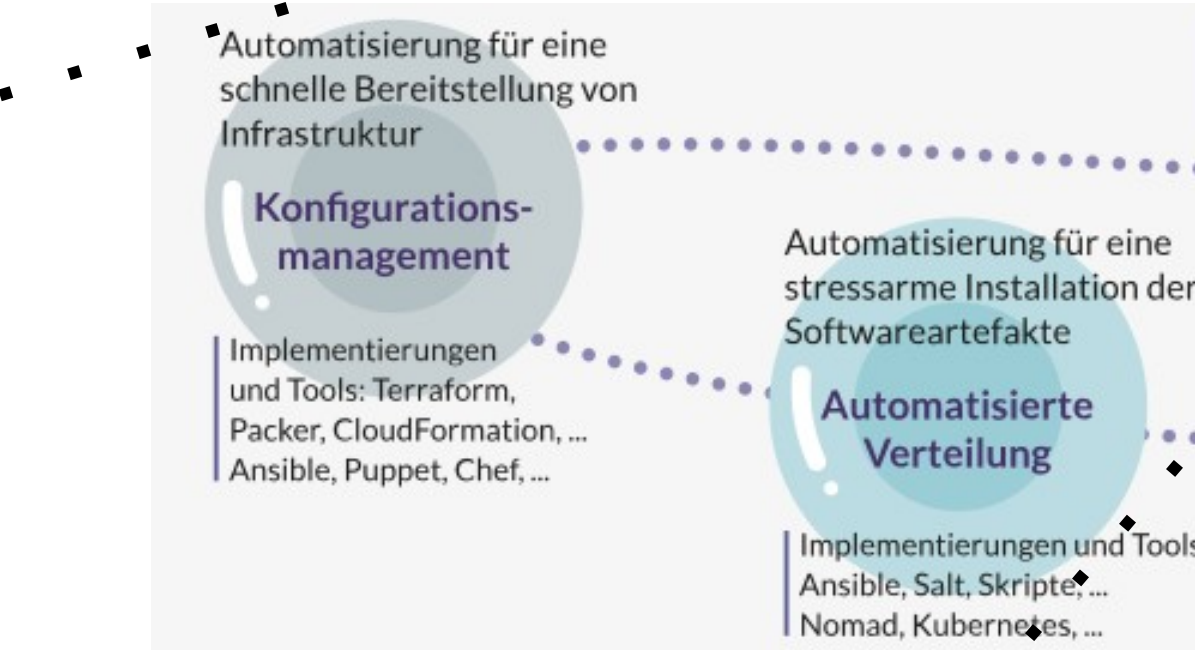
Was hat das mit Infrastructure As Code zu tun?

Continuous Delivery



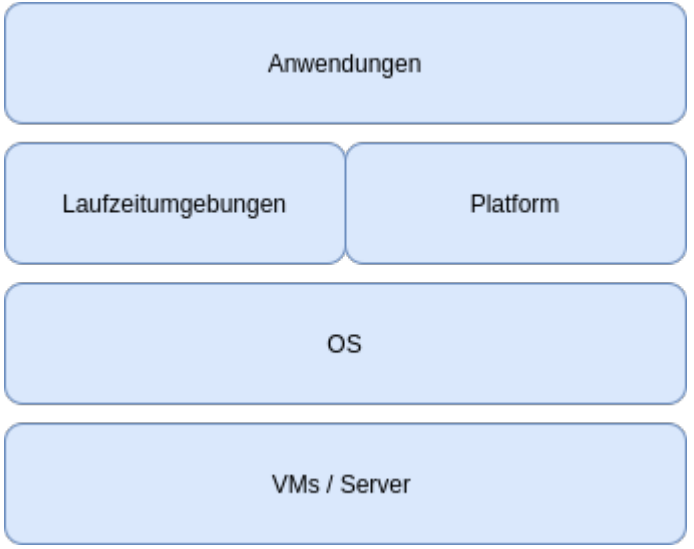
Aus: Architektur Spicker 7 – Continuous Delivery

Infrastructure As Code

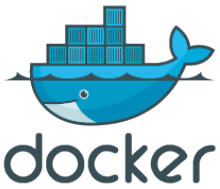


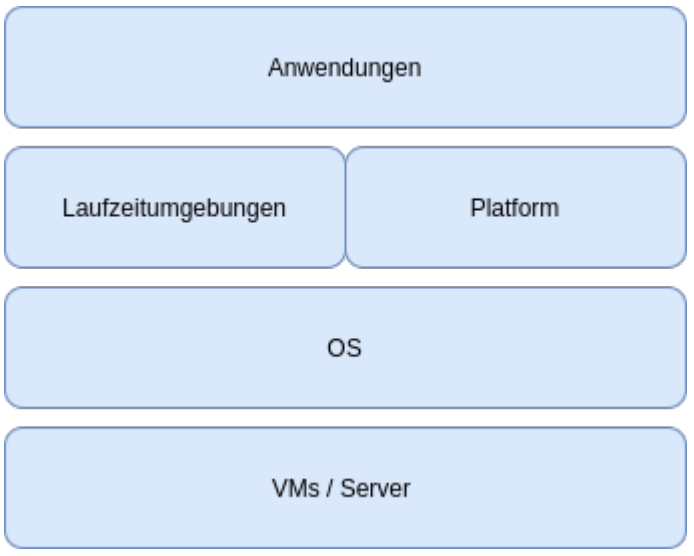
In der klassischen IT:
Ops-Tätigkeiten

Infrastruktur-Stack



Infrastructure as Code (IaC)





HashiCorp Terraform

```
resource "azurerm_linux_vi
  name                =
  location            =
  resource_group_name =
  network_interface_ids =
  size                =
  admin_username      =
  tags                =
    app = "hero"
}

os_d
  ca
  st
}

sour
  pu
  of
  sk
  ve
}

admi
  us
  public_key = file(
}
```

```
FROM adoptopenjdk:11-jre-hotspot as build
WORKDIR application
COPY *.jar application.jar
RUN java -Djarmode=layertools -jar applic

FROM gcr.io/distroless/java:11
WORKDIR application
EXPOSE 8080
COPY --from=builder application/dependenc
COPY --from=builder application/spring-bo
COPY --from=builder application/snapshot-
COPY --from=builder application/applicati
ENTRYPOINT ["java", "org.springframework.
```

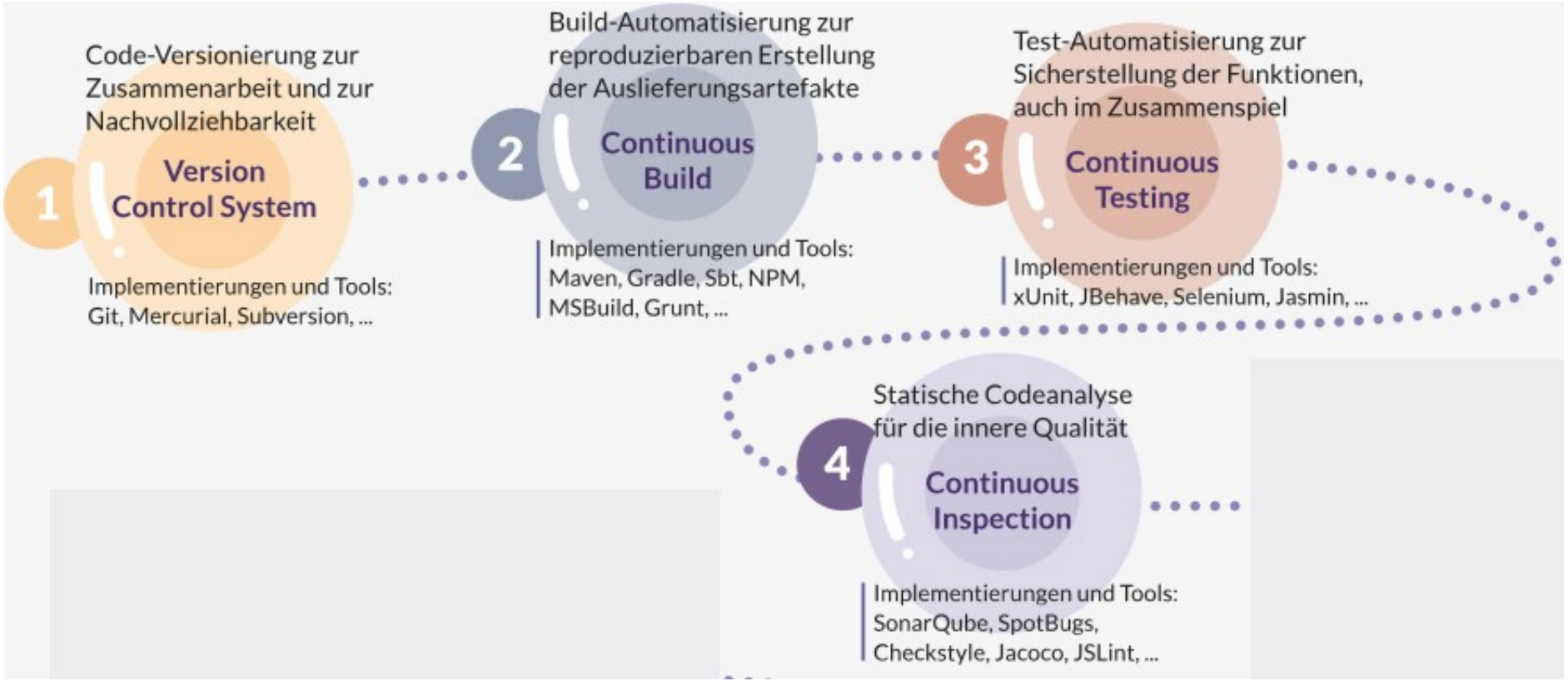
```
- hosts: application_serve
  vars:
    tomcat_version: 9.0.27
    tomcat_base_name: apac

  tasks:
    - name: install java
      apt:
        name: openjdk-8-jd
        become: yes
        become_method: sudo
```

```
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?


Code-Versionierung zur Zusammenarbeit und zur Nachvollziehbarkeit
1
Version Control System
Implementierungen und Tools: Git, Mercurial, Subversion, ...
Build-Automatisierung zur reproduzierbaren Erstellung der Auslieferungsartefakte
2
Continuous Build
Implementierungen und Tools: Maven, Gradle, Sbt, NPM, MSBuild, Grunt, ...
Test-Automatisierung zur Sicherstellung der Funktionen, auch im Zusammenspiel
3
Continuous Testing
Implementierungen und Tools: xUnit, JBehave, Selenium, Jasmin, ...
Statische Codeanalyse für die innere Qualität
4
Continuous Inspection
Implementierungen und Tools: SonarQube, SpotBugs, Checkstyle, Jacoco, JSLint, ...

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

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.

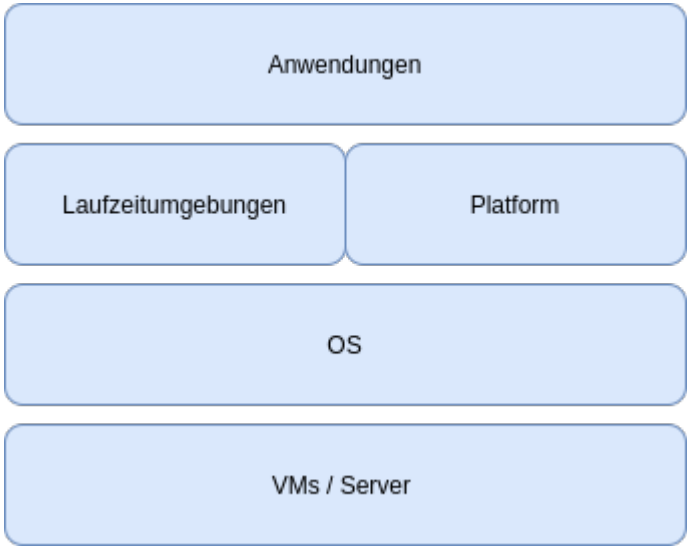
BEST PRACTICES

- 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).

Arten von Repositories



- Skripte, die Anwendung verteilen und konfigurieren → nah an der Anwendung
- Skripte, die Platformen / Systeme konfigurieren
- Skripte, die virtuelle Hardware bereitstellen
- Konfigurationsdateien

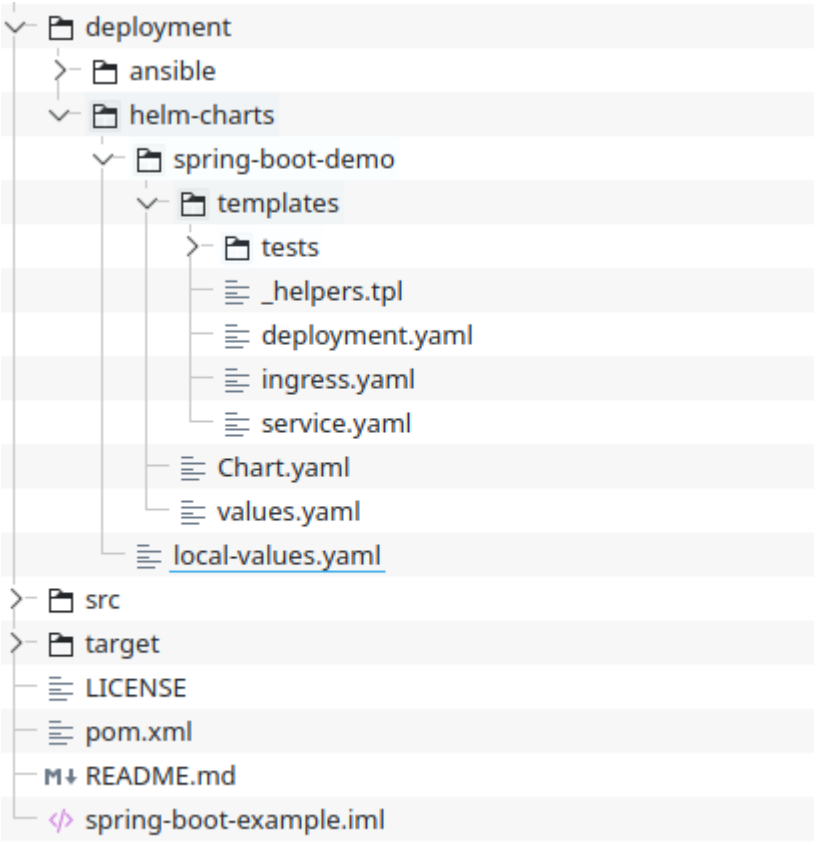
Beispiel: K8s

- Skripte, die den Cluster allgemein konfigurieren
- Skripte, die Anwendung auf den Cluster deployen
- Konfigurationsdateien

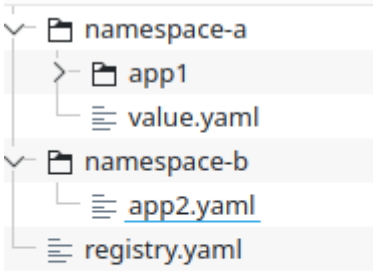
Beispiel: K8s

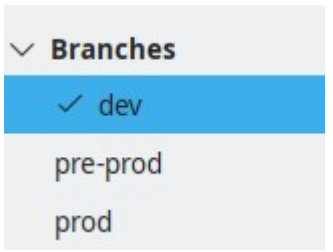
- Eigenes Repo für die Skripte, die den Cluster allgemein konfigurieren
- Deployment Skripte, in das Repo, wo auch die Source Code der Anwendung liegt
- Eigenes Repo für die Konfiguration

Beispiel: K8s



Beispiel: K8s






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 *.yml'
                }
            }
        }
        stage('Ansible Playbook run with tests') {
            steps {
                dir('samples') {
                    sh 'vagrant up'
                    sh 'ansible-playbook -i inventories/test setup-tomcat.yml'
                    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

3 Continuous Testing

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

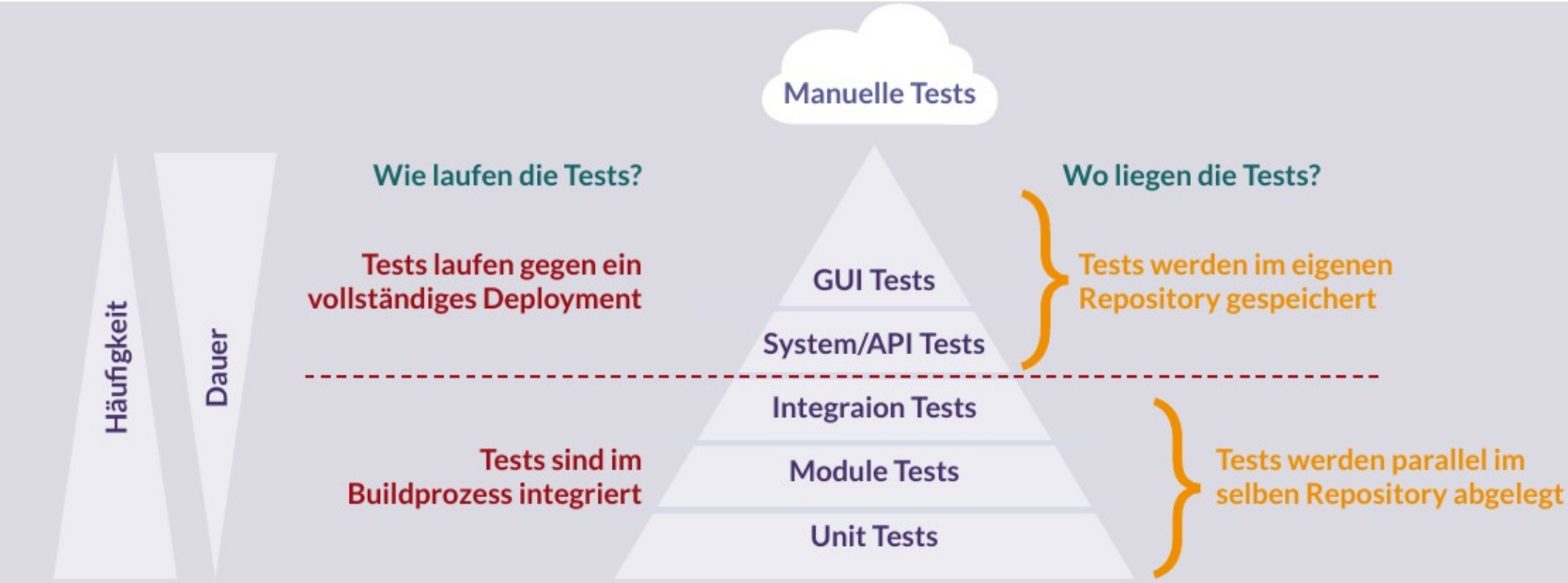
BEST PRACTICES

- 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.

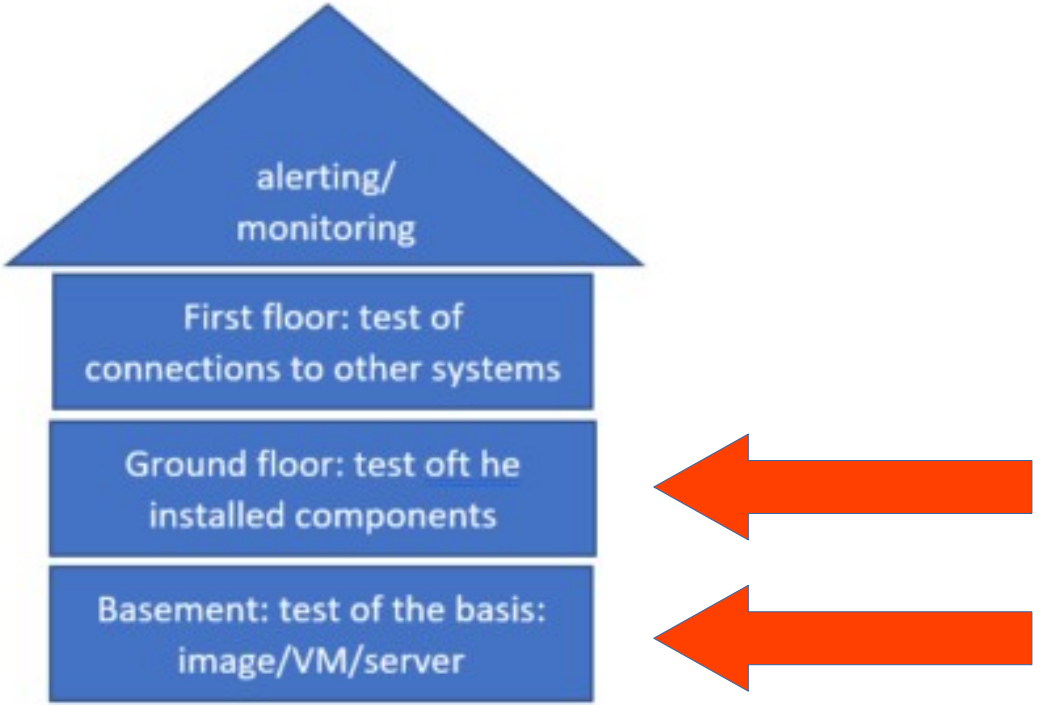
SO ARBEITEN EURE TEAMS

- 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



Disclaimer: Resultat aus einer Session auf dem QS Barcamp 2020

Continuous Testing für Ops


```go
package test

import (…)

func TestTerraformAwsHelloWorldExample(t *testing.T) {
  t.Parallel()

  // Construct the terraform options with default retryable errors to handle the most common
  // retryable errors in terraform testing.
  terraformOptions := terraform.WithDefaultRetryableErrors(t, &terraform.Options{
    // The path to where our Terraform code is located
    TerraformDir: "../azure-demo",
  })

  // At the end of the test, run `terraform destroy` to clean up any resources that were created.
  defer terraform.Destroy(t, terraformOptions)

  // Run `terraform init` and `terraform apply`. Fail the test if there are any errors.
  terraform.InitAndApply(t, terraformOptions)

  // Run `terraform output` to get the IP of the instance
  publicIp := terraform.Output(t, terraformOptions, "public_ip_address")

  assert.NotNil(t, publicIp)
}
```

Continuous Testing für Ops

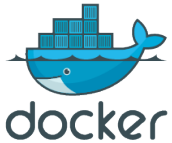

```
~/dev/workspace/iac-qa-talk/samples/terraform/test(master ✗) go test . -v
=== RUN   TestTerraformAwsHelloWorldExample
=== PAUSE TestTerraformAwsHelloWorldExample
=== CONT  TestTerraformAwsHelloWorldExample
TestTerraformAwsHelloWorldExample 2021-06-08T19:34:57+02:00 retry.go:91: terraform [init -upgrade=false]
TestTerraformAwsHelloWorldExample 2021-06-08T19:34:57+02:00 logger.go:66: Running command terraform with args [init -upgrade=false]
TestTerraformAwsHelloWorldExample 2021-06-08T19:34:57+02:00 logger.go:66:
TestTerraformAwsHelloWorldExample 2021-06-08T19:34:57+02:00 logger.go:66: Initializing the backend...
TestTerraformAwsHelloWorldExample 2021-06-08T19:34:57+02:00 logger.go:66:
TestTerraformAwsHelloWorldExample 2021-06-08T19:34:57+02:00 logger.go:66: Initializing provider plugins...
TestTerraformAwsHelloWorldExample 2021-06-08T19:34:57+02:00 logger.go:66: - Finding hashicorp/azurerm versions matching "2.59.0"...
TestTerraformAwsHelloWorldExample 2021-06-08T19:34:57+02:00 logger.go:66: - Installing hashicorp/azurerm v2.59.0...
TestTerraformAwsHelloWorldExample 2021-06-08T19:35:03+02:00 logger.go:66: - Installed hashicorp/azurerm v2.59.0 (signed by HashiCorp)
TestTerraformAwsHelloWorldExample 2021-06-08T19:35:03+02:00 logger.go:66:
TestTerraformAwsHelloWorldExample 2021-06-08T19:35:03+02:00 logger.go:66: Terraform has created a lock file .terraform.lock.hcl to record the provider
TestTerraformAwsHelloWorldExample 2021-06-08T19:35:03+02:00 logger.go:66: selections it made above. Include this file in your version control repository
TestTerraformAwsHelloWorldExample 2021-06-08T19:35:03+02:00 logger.go:66: so that Terraform can guarantee to make the same selections by default when
TestTerraformAwsHelloWorldExample 2021-06-08T19:35:03+02:00 logger.go:66: you run "terraform init" in the future.
TestTerraformAwsHelloWorldExample 2021-06-08T19:35:03+02:00 logger.go:66:
TestTerraformAwsHelloWorldExample 2021-06-08T19:35:03+02:00 logger.go:66: Terraform has been successfully initialized!
TestTerraformAwsHelloWorldExample 2021-06-08T19:35:03+02:00 logger.go:66:
TestTerraformAwsHelloWorldExample 2021-06-08T19:35:03+02:00 logger.go:66: You may now begin working with Terraform. Try running "terraform plan" to see

TestTerraformAwsHelloWorldExample 2021-06-08T19:35:03+02:00 logger.go:66: any changes that are required for your infrastructure. All Terraform commands

TestTerraformAwsHelloWorldExample 2021-06-08T19:35:03+02:00 logger.go:66: should now work.
```

Continuous Testing für Ops


```python
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


```
~/dev/workspace/iac-qa-talk/samples(master ✗) 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

```yaml
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 ✗)
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

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: "{{ include "spring-boot-demo.fullname" . }}-test-connection"
  labels:
    {{- include "spring-boot-demo.labels" . | nindent 4 }}
  annotations:
    "helm.sh/hook": test
spec:
  containers:
    - name: wget
      image: busybox
      command: ['wget']
      args: ['{{ include "spring-boot-demo.fullname" . }}:{{ .Values.service.port }}/actuator/health']
  restartPolicy: Never
```

Continuous Testing für Ops

```
~/dev/workspace/iac-qa-talk/samples/helm-charts (master ✗)
 helm test spring-boot-demo-instance
NAME: spring-boot-demo-instance
LAST DEPLOYED: Fri Mar 18 15:28:13 2022
NAMESPACE: default
STATUS: deployed
REVISION: 2
TEST SUITE:     spring-boot-demo-instance-spring-boot-demo
-helm-chart-test-connection
Last Started:   Fri Mar 18 15:36:49 2022
Last Completed: Fri Mar 18 15:36:52 2022
Phase:          Succeeded
```

Continuous Testing für Ops

Terratest

```go
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)
}
```

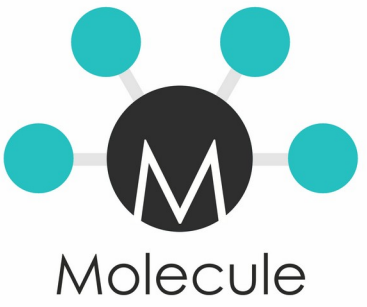
Continuous Testing für Ops

Terratest

```
~/dev/workspace/iac-qa-talk/samples/helm-charts/test(master ✗) 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 ✗)
```

Continuous Testing für Ops

Weitere Testwerkzeuge



Goss


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 im den Build integriert.
- Verstoße gegen den Regelsatz werden sofort behoben.

Continuous Inspection für Ops

```
~/dev/workspace/iac-qa-talk/samples/terraform/azure-demo(master x) tflint --loglevel=info .
20:25:09 config.go:105: [INFO] Load config: .tflint.hcl
20:25:09 loader.go:57: [INFO] Initialize new loader
20:25:09 loader.go:82: [INFO] Load configurations under .
20:25:09 loader.go:90: [INFO] Module inspection is disabled. Building a root module without children...
20:25:09 loader.go:170: [INFO] Load values files
20:25:09 runner.go:50: [INFO] Initialize new runner for root
20:25:09 discovery.go:54: [INFO] Plugin `azurerm` found
20:25:09 provider.go:62: [INFO] Prepare rules
20:25:09 provider.go:90: [INFO]   3 default rules enabled
```

Continuous Inspection für Ops

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

[502] All tasks should be named
setup-tomcat.yml: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
```

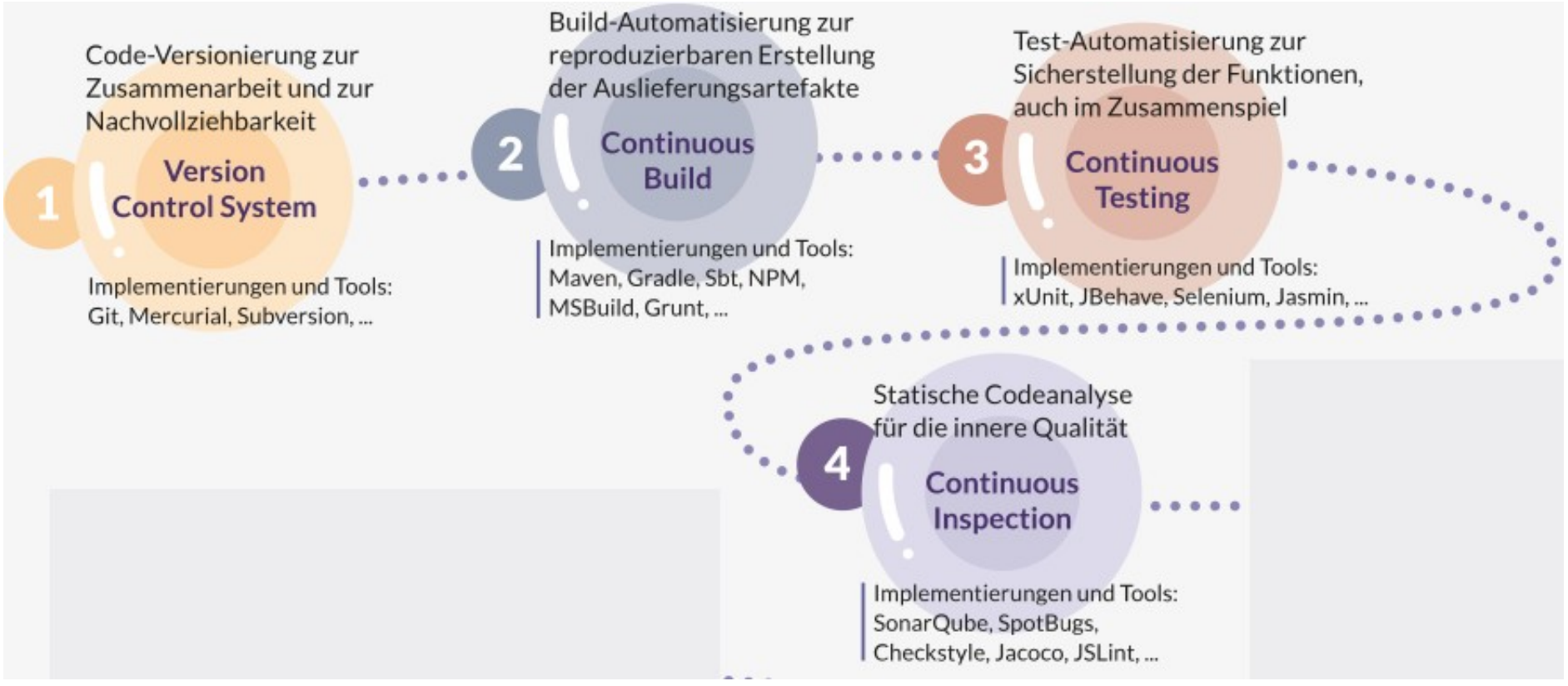
Continuous Inspection für Ops

```
~/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)
```

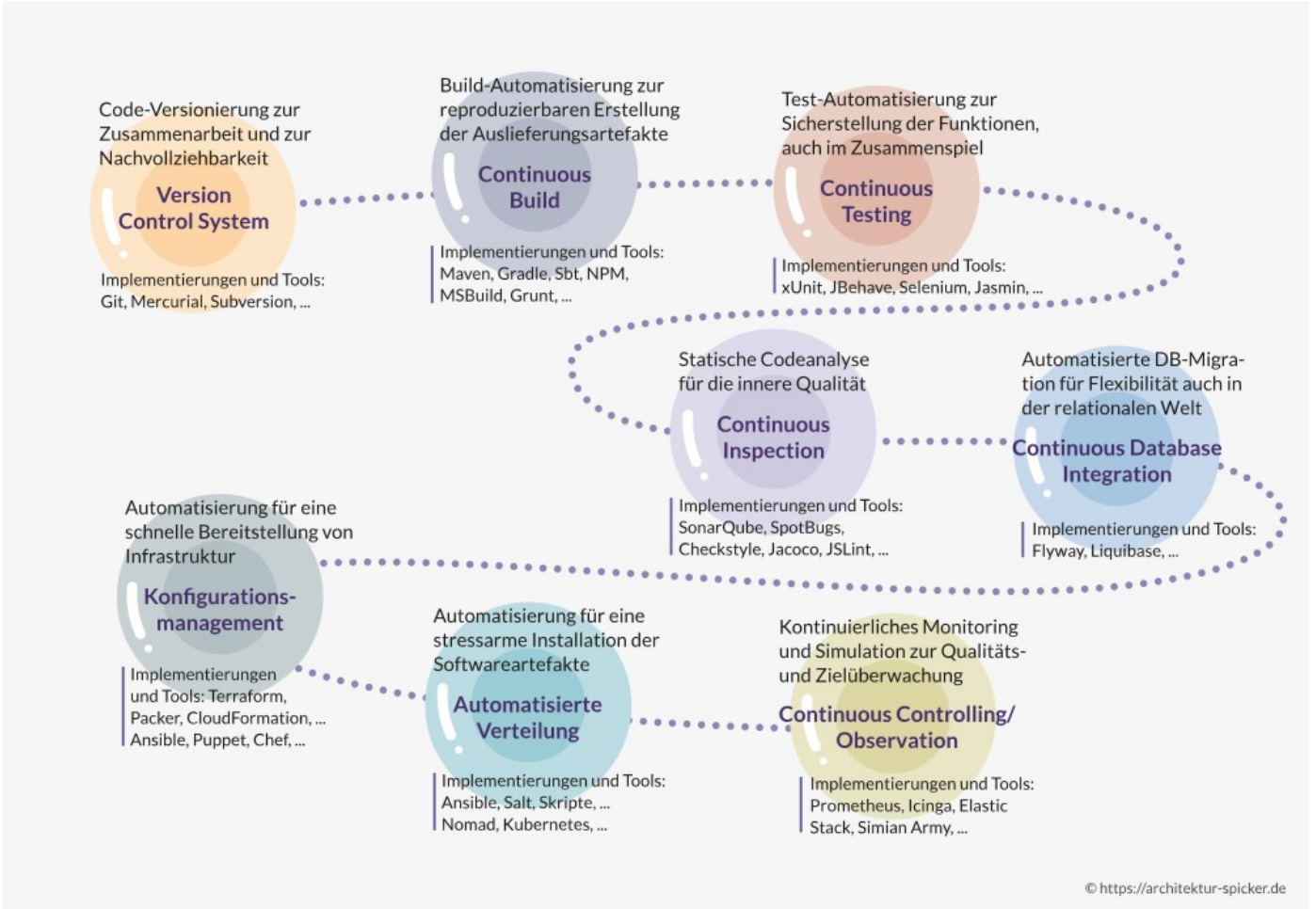
Continuous Inspection für Ops

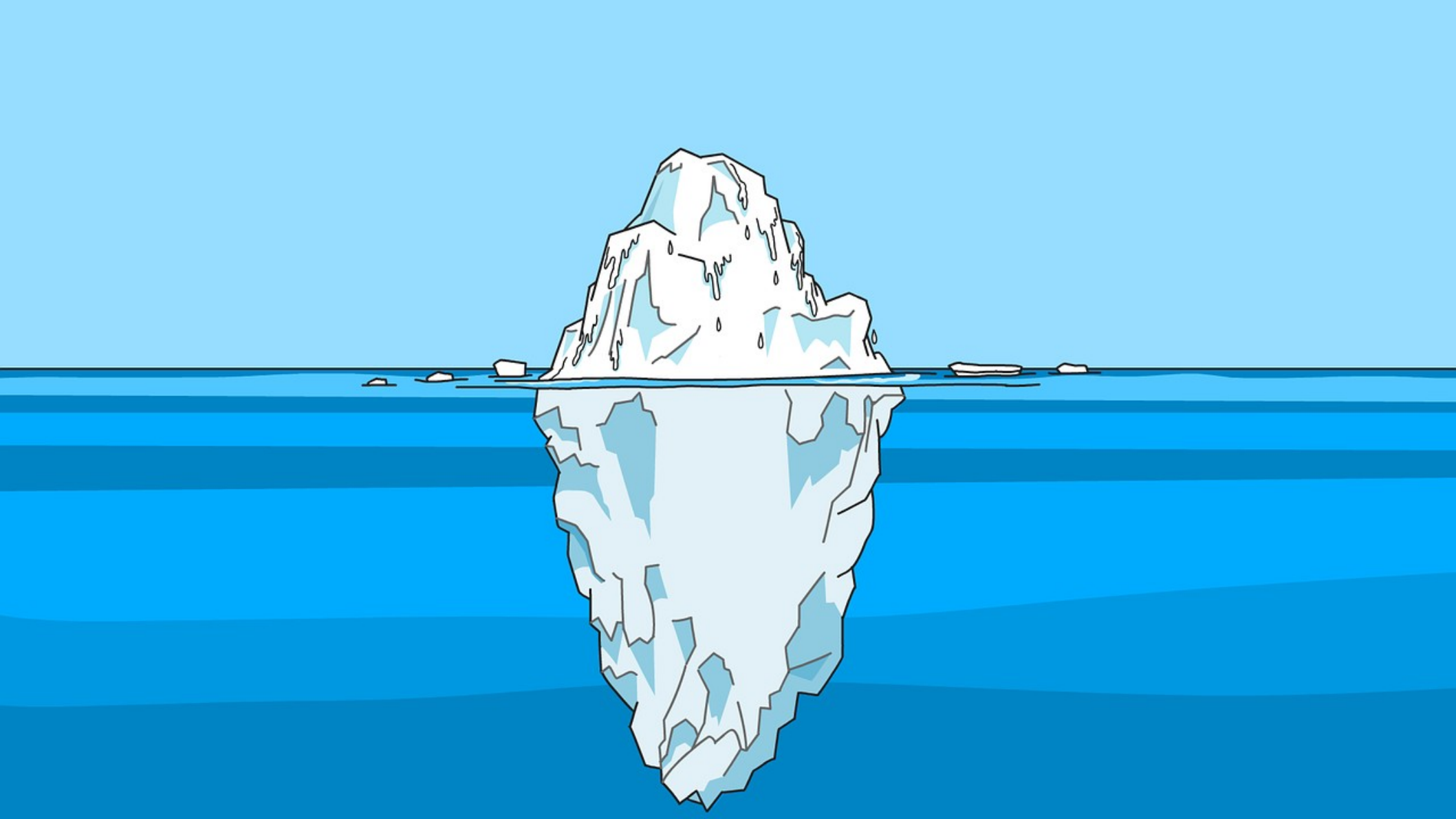
Weitere Linter

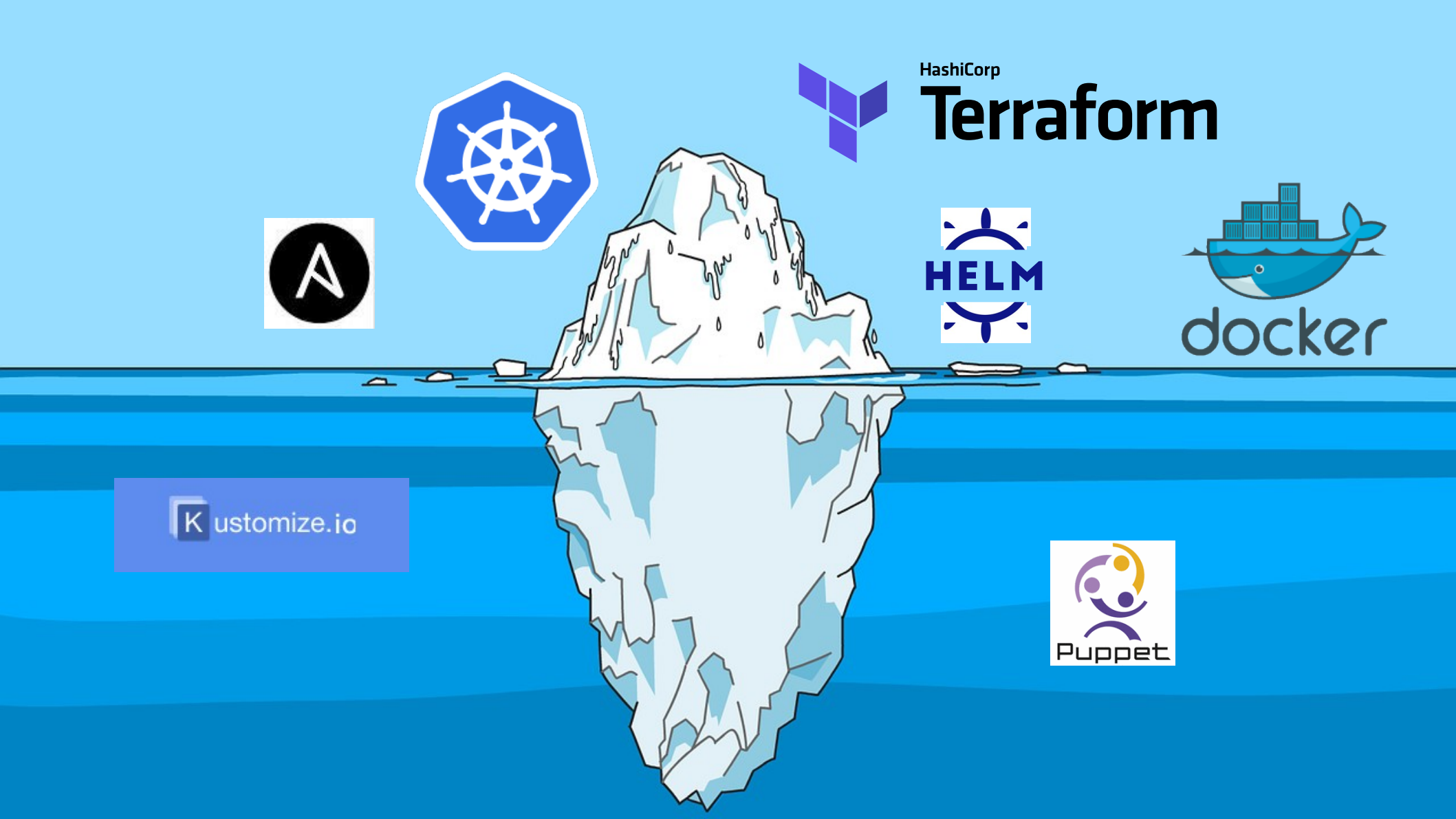
- Docker
  - dockerfile_lint
  - dockerlint
- Kubernetes
  - yamlint
  - kube-score
  - kubelint
- Ansible
  - yamlint


Code-Versionierung zur Zusammenarbeit und zur Nachvollziehbarkeit
1
Version Control System
Implementierungen und Tools: Git, Mercurial, Subversion, ...
Build-Automatisierung zur reproduzierbaren Erstellung der Auslieferungsartefakte
2
Continuous Build
Implementierungen und Tools: Maven, Gradle, Sbt, NPM, MSBuild, Grunt, ...
Test-Automatisierung zur Sicherstellung der Funktionen, auch im Zusammenspiel
3
Continuous Testing
Implementierungen und Tools: xUnit, JBehave, Selenium, Jasmin, ...
Statische Codeanalyse für die innere Qualität
4
Continuous Inspection
Implementierungen und Tools: SonarQube, SpotBugs, Checkstyle, Jacoco, JSLint, ...

Continuous Delivery






HashiCorp
Terraform
HELM
docker
Kustomize.io
Puppet

CNCF Cloud Native Landscape
2021-02-06T17:04:46Z fca7abca

Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io

Greyed logos are not open source

App Definition and Development

Database

CNCF Graduated: KV, Vitess

Streaming & Messaging

CNCF Incubating: cloudevents, NATS

Application Definition & Image Build

CNCF Graduated: HELM

CNCF Incubating: Buildpacks.io, Operator Framework

Continuous Integration & Delivery

CNCF Incubating: argo

Orchestration & Management

Scheduling & Orchestration

CNCF Graduated: kubernetes

Coordination & Service Discovery

CNCF Graduated: CoreDNS

CNCF Incubating: etcd

Remote Procedure Call

CNCF Incubating: gRPC

Service Proxy

CNCF Graduated: envoy

CNCF Incubating: CONTOUR

API Gateway

CNCF Incubating: Ambassador, APISIX, Kong, Kuma, MuleSoft

Service Mesh

CNCF Incubating: LINKERD, Kuma

Runtime

Cloud Native Storage

CNCF Incubating: ROOK

Container Runtime

CNCF Graduated: containerd

CNCF Incubating: cri-o

Cloud Native Network

CNCF Incubating: CNI, Cilium, Flannel, OVS, Weave Net

Provisioning

Automation & Configuration

CNCF Incubating: KubeEdge

Container Registry

CNCF Graduated: Harbor

CNCF Incubating: Dragonfly

Security & Compliance

CNCF Graduated: Open Policy Agent

CNCF Incubating: Falco, Notary

Key Management

CNCF Incubating: SPIFFE, SPIRE

Platform

Certified Kubernetes - Distribution

Certified Kubernetes - Hosted

Certified Kubernetes - Installer

PaaS/Container Service

Observability and Analysis

Monitoring

CNCF Graduated: Prometheus

CNCF Incubating: cortex, Thanos

Logging

CNCF Graduated: fluentd

Tracing

CNCF Graduated: Jaeger

CNCF Incubating: OpenTracing

Chaos Engineering

Serverless

Members

Kubernetes Certified Service Provider

Kubernetes Training Partner

Special

CLOUD NATIVE Landscape

CLOUD NATIVE COMPUTING FOUNDATION

Redpoint Amplify Partners

l.cncf.io

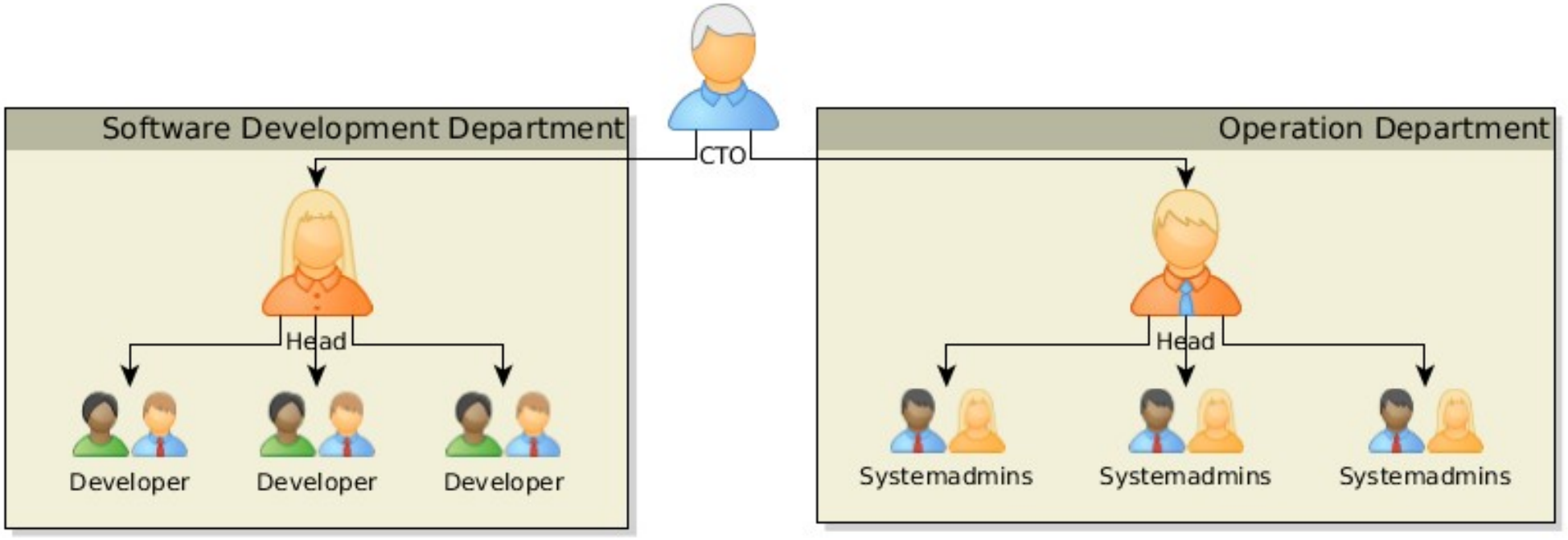
This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.

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

Idealbild


Team 1
Developer
Tester
Systemadmins
Team 2
Developer
Systemadmins
Tester

(Oft noch) Realität



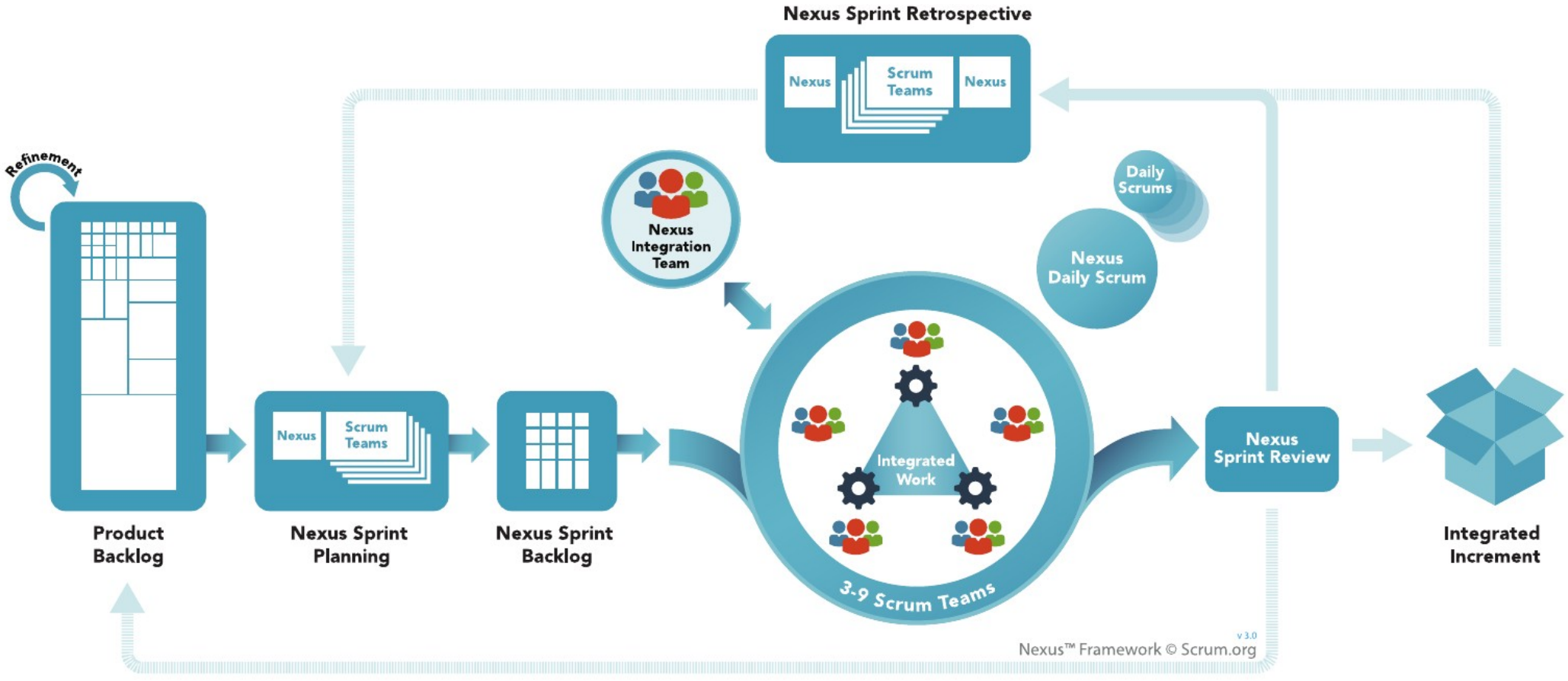
Herausforderungen Ops-Abteilung

- 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



SAFe® for Lean Enterprises 5.0

Select Configuration

Overview | Essential SAFe | Large Solution SAFe | Portfolio SAFe | Full SAFe

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 Engineering

Coordinating Trains and Suppliers

Continually 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 Thinking

Develop on Cadence Release on Demand

DevOps 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 Teams

Lean Business Operations

Strategy Agility

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 Teams

Built-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

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 Organization

Innovation Culture

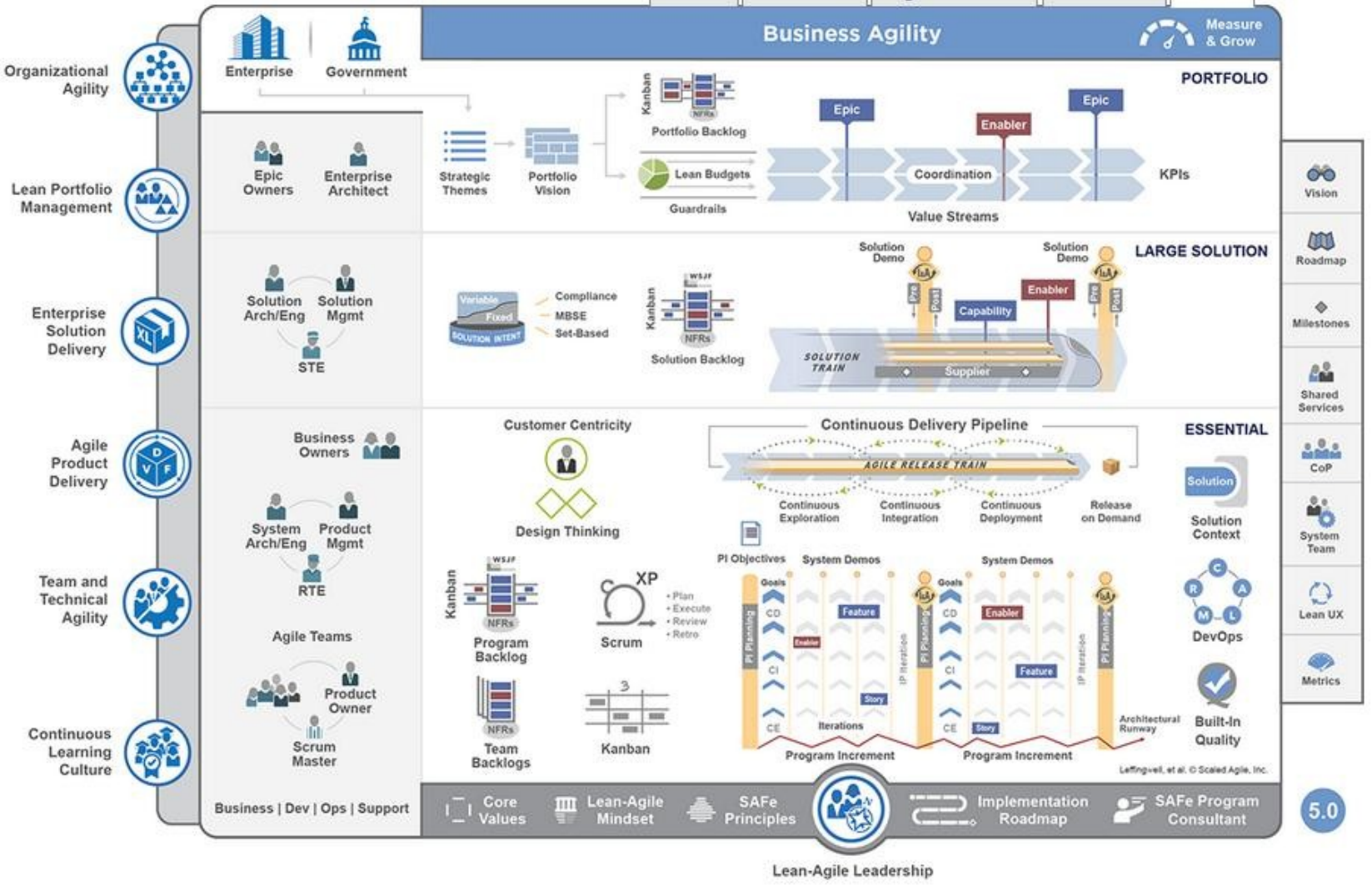
Relentless Improvement

SAFe® PROVIDED BY SCALED AGILE®

SAFe® for Lean Enterprises 5.0

Select Configuration

Overview | Essential SAFe | Large Solution SAFe | Portfolio SAFe | Full SAFe



Reality-Check

„Du musst 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 vernachlässig
    - 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/