GitOps / DevOps / AppOps / SRE

Konzepte & Tools

Thomas Siwczak

Version 0.22.1, 29.01.2024

Inhaltsverzeichnis

1. Einstieg in DevOps . . . 1
2. Kaniko . . . 4
3. Packer.io . . . 6
4. Skopeo . . . 8
5. Terraform . . . 10
6. ArgoCD . . . 16
7. GitLab Pipelines . . . 19
8. Gitlab pipelines in advanced . . . 22
9. Jenkins . . . 25
10. Tekton . . . 27
11. Kubernetes . . . 29
12. k9s . . . 32
13. kURL . . . 34
14. Podman . . . 36
15. Trivy . . . 37
16. Asciidoctor . . . 41
17. Hugo . . . 51
18. Git . . . 53
19. Semantic Versioning . . . 63
20. RKE2 - Rancher . . . 64
21. RKE2 - Rancher . . . 66
22. HAProxy . . . 68
23. Consul . . . 71
24. Tmux . . . 72
25. Vagrant . . . 75
26. Gegenüberstellung: Ansible, Chef, Puppet und SaltStack . . . 76
27. Semantische Versionsbezeichnungen . . . 78
28. Sammlung nützlicher Befehle und Scripte . . . 80

1. Einstieg in DevOps

1.1. Einleitung

DevOps ist ein Begriff, der in der Softwareentwicklung immer mehr an Bedeutung gewinnt. Es handelt sich um eine Kultur, Methodik und einen Satz von Werkzeugen, die darauf abzielen, die Zusammenarbeit und Integration zwischen Entwicklungsteams (Dev) und Betriebsteams (Ops) zu verbessern. In diesem Blogpost werden wir uns näher mit dem Einstieg in DevOps beschäftigen und Schritte aufzeigen, wie Sie diese Praktiken erfolgreich implementieren können.

1.2. Was ist DevOps?

DevOps ist eine Philosophie, die auf der Idee basiert, dass Entwicklung und Betrieb nicht als getrennte Bereiche betrachtet werden sollten. Stattdessen sollten sie eng zusammenarbeiten, um Software effizienter bereitzustellen und kontinuierlich zu verbessern. DevOps fördert eine Kultur der Zusammenarbeit, Automatisierung und kontinuierlichen Überprüfung, um die Bereitstellung von Software schneller und stabiler zu machen.

1.3. Die Vorteile von DevOps

Die Einführung von DevOps bietet eine Vielzahl von Vorteilen für Organisationen. Einige der wichtigsten sind:

1. Schnellere Bereitstellung: DevOps ermöglicht es Teams, Software schneller zu entwickeln und bereitzustellen. Durch die Automatisierung von Prozessen und die enge Zusammenarbeit zwischen Entwicklung und Betrieb können Softwareänderungen schnell getestet und implementiert werden.
2. Höhere Qualität: Durch kontinuierliche Integration, automatisierte Tests und kontinuierliches Deployment können potenzielle Fehler frühzeitig erkannt und behoben werden. Dies führt zu einer insgesamt höheren Softwarequalität.
3. Bessere Zusammenarbeit: DevOps fördert eine Kultur der Zusammenarbeit und Kommunikation zwischen Entwicklung und Betrieb. Durch den gemeinsamen Fokus auf die Bereitstellung von hochwertiger Software können Konflikte minimiert und die Produktivität gesteigert werden.
4. Skalierbarkeit: DevOps ermöglicht es Organisationen, schnell auf sich ändernde Anforderungen und Kundenbedürfnisse zu reagieren. Durch die Automatisierung von Prozessen können Teams effizienter arbeiten und Softwarelösungen skalieren.

1.4. Der Einstieg in DevOps

Der Einstieg in DevOps erfordert eine schrittweise Herangehensweise und eine klare Strategie. Hier sind einige wichtige Schritte, die Sie berücksichtigen sollten:

1.4.1. Verständnis der DevOps-Prinzipien

Bevor Sie mit der Implementierung von DevOps beginnen, ist es wichtig, die zugrunde liegenden Prinzipien und Best Practices zu verstehen. Dazu gehören kontinuierliche Integration, kontinuierliches Deployment, Automatisierung, Testautomatisierung und kontinuierliches Monitoring. Informieren Sie sich über diese Konzepte und ihre Auswirkungen auf die Softwareentwicklung.

1.4.2. Schaffen Sie eine Kultur der Zusammenarbeit

DevOps erfordert eine enge Zusammenarbeit zwischen Entwicklungsteams, Betriebsteams und anderen relevanten Abteilungen. Schaffen Sie eine Kultur, die auf offener Kommunikation, Vertrauen und gemeinsamer Verantwortung basiert. Fördern Sie den Austausch von Wissen und Ideen zwischen den Teams und schaffen Sie Möglichkeiten für regelmäßige Meetings und Zusammenarbeit.

1.4.3. Automatisierung der Bereitstellung

Die Automatisierung spielt eine entscheidende Rolle in DevOps. Automatisieren Sie so viele Prozesse wie möglich, um die Effizienz und Geschwindigkeit der Bereitstellung von Software zu erhöhen. Automatisieren Sie beispielsweise den Build-Prozess, die Tests, das Deployment und das Monitoring. Verwenden Sie Werkzeuge wie Jenkins, Ansible oder Docker, um diese Aufgaben zu automatisieren.

1.4.4. Einsatz von Continuous Integration und Continuous Deployment

Die kontinuierliche Integration und das kontinuierliche Deployment sind Kernprinzipien von DevOps. Implementieren Sie eine Pipeline für die kontinuierliche Integration, um sicherzustellen, dass Codeänderungen regelmäßig und automatisch getestet werden. Verwenden Sie Tools wie Git, Jenkins und SonarQube, um den Prozess der kontinuierlichen Integration zu unterstützen. Für das kontinuierliche Deployment verwenden Sie Werkzeuge wie Kubernetes oder AWS Elastic Beanstalk, um die Software nahtlos und automatisch in die Produktionsumgebung zu übertragen.

1.4.5. Überwachung und Feedback

Eine kontinuierliche Überwachung ist unerlässlich, um die Leistung und Stabilität Ihrer Anwendungen zu gewährleisten. Implementieren Sie ein effektives Monitoring-System, das wichtige Metriken und Alarme erfasst. Verwenden Sie Werkzeuge wie Nagios, Grafana oder ELK Stack, um den Zustand Ihrer Anwendungen und Infrastruktur zu überwachen. Nutzen Sie das Feedback aus der Überwachung, um Verbesserungen vorzunehmen und Engpässe zu identifizieren.

1.4.6. Kontinuierliche Verbesserung

DevOps ist ein kontinuierlicher Prozess. Stellen Sie sicher, dass Sie regelmäßige Reviews und Retrospektiven durchführen, um den Fortschritt zu bewerten und Verbesserungspotenziale zu identifizieren. Nutzen Sie die gewonnenen Erkenntnisse, um den DevOps-Prozess weiter zu optimieren und effizienter zu gestalten.

1.5. Fazit

Der Einstieg in DevOps erfordert ein klares Verständnis der zugrunde liegenden Prinzipien und Best Practices sowie eine schrittweise Umsetzung. Durch die Schaffung einer Kultur der Zusammenarbeit, die Automatisierung von Prozessen und die kontinuierliche Integration und Bereitstellung können Organisationen die Vorteile von DevOps nutzen. Mit der richtigen Strategie und den geeigneten Tools können Sie Ihre Softwareentwicklung und Bereitstellung optimieren und qualitativ hochwertige Software effizienter liefern.

Beginnen Sie noch heute Ihren DevOps-Weg und erleben Sie die positiven Auswirkungen auf Ihre Organisation!

2. Kaniko

Einleitung, Anleitung und Beispiele

2.1. Einleitung

Kaniko ist ein Open-Source-Tool, entwickelt von Google, das zum Bauen von Docker-Images innerhalb eines Kubernetes-Clusters oder einer anderen Umgebung ohne Docker Daemon verwendet wird. Es ermöglicht eine sichere und effiziente Erstellung von Container-Images direkt aus dem Quellcode.

2.2. Wie verwendet man Kaniko?

Um Kaniko zu nutzen, benötigen Sie zunächst eine Kubernetes-Umgebung. Sie können dann ein `kaniko` Pod in Ihrem Cluster starten, der auf Ihren Dockerfile zeigt und ein Image in Ihrer gewünschten Registry erstellt.

2.2.1. Schritte zur Verwendung von Kaniko

1. Installieren Sie Kaniko in Ihrem Kubernetes-Cluster:

```
kubectl create -f
https://github.com/GoogleContainerTools/kaniko/blob/master/deploy/Dockerfile
```

1. Erstellen Sie eine geheime Datei für Ihre Registry:

```
kubectl create secret docker-registry regcred --docker-server=<your-registry-server>
--docker-username=<your-name> --docker-password=<your-pword> --docker-email=<your-
email>
```

1. Verwenden Sie einen `kaniko` Pod, um Ihr Image zu erstellen:

```
apiVersion: v1
kind: Pod
```

```
metadata:
  name: kaniko
spec:
  containers:
  - name: kaniko
    image: gcr.io/kaniko-project/executor:latest
    args: ["--dockerfile=/Dockerfile",
           "--context=dir://<your-source-code>",
           "--destination=<your-registry>/<your-image>:<your-tag>"]
    volumeMounts:
      - name: kaniko-secret
        mountPath: /secret
        readOnly: true
    env:
      - name: GOOGLE_APPLICATION_CREDENTIALS
        value: /secret/kaniko-secret.json
  restartPolicy: Never
  volumes:
    - name: kaniko-secret
      secret:
        secretName: kaniko-secret
```

2.3. Beispiele

Nun, da Sie eine Vorstellung davon haben, wie man Kaniko verwendet, finden Sie hier einige Anwendungsfälle:

1. **Erstellen eines Python-Images**: Wenn Sie einen Dockerfile haben, der auf ein Python-Image zeigt und Anforderungen aus einer `requirements.txt`-Datei installiert, können Sie Kaniko verwenden, um dieses Image effizient zu erstellen und es in Ihrer Registry bereitzustellen.
2. **Erstellen eines Java-Images**: Ähnlich wie beim Python-Beispiel können Sie einen Dockerfile verwenden, der auf ein Java-Image zeigt und Ihre `.jar`-Datei in das Image kopiert. Kaniko kann dann verwendet werden, um dieses Image zu erstellen und es in Ihrer Registry bereitzustellen.

2.4. Links / Cheatsheets

- https://link.medium.com/jkNWyPVkwub

3. Packer.io

3.1. Was ist Packer.io?

Packer.io ist eine kostenlose Open-Source-Tool zur Erstellung identischer Maschinenbilder für mehrere Plattformen aus einer einzigen Quellkonfiguration. Es wird von HashiCorp entwickelt, einem Unternehmen, das für die Entwicklung von Tools wie Vagrant, Terraform und Consul bekannt ist. Packer.io ist in der Programmiersprache Go geschrieben und kann auf mehreren Plattformen wie Linux, Windows und Mac OS X laufen.

3.2. Warum Packer.io verwenden?

Es gibt viele Gründe, warum Entwickler und Systemadministratoren Packer.io verwenden. Einige davon sind:

- **Konsistenz:** Mit Packer.io können Sie Maschinenbilder erstellen, die auf allen Ihren Servern gleich sind. Dies verringert das Risiko von Fehlern und Vereinfacht die Fehlersuche.
- **Zeitersparnis:** Mit Packer.io können Sie Maschinenbilder automatisch erstellen, ohne dass ein manueller Eingriff erforderlich ist. Das bedeutet, dass Sie weniger Zeit mit der Konfiguration von Servern verbringen und mehr Zeit für andere Aufgaben haben.
- **Plattformunabhängigkeit:** Packer.io unterstützt eine Vielzahl von Plattformen, darunter Amazon EC2, Google Cloud, Microsoft Azure, VMware, Docker und viele mehr. Sie können also dasselbe Tool verwenden, unabhängig davon, wo Ihre Server laufen.

3.3. Wie funktioniert Packer.io?

Packer.io verwendet Konfigurationsdateien, die in JSON geschrieben sind. In diesen Dateien definieren Sie, welche Art von Maschinenbild Sie erstellen möchten, welche Software darauf installiert sein soll und wie das Bild konfiguriert werden soll.

Sobald Sie Ihre Konfigurationsdatei erstellt haben, verwenden Sie das Befehlszeilen-Interface von

Packer.io, um das Maschinenbild zu erstellen. Packer.io führt dann eine Reihe von Schritten aus, die als "Provisioner" und "Post-Prozessoren" bezeichnet werden, um das Maschinenbild zu erstellen und zu konfigurieren.

3.4. Fazit

Packer.io ist ein leistungsfähiges Tool, das Ihnen hilft, konsistente und zuverlässige Serverumgebungen zu erstellen. Mit seiner Unterstützung für eine Vielzahl von Plattformen und seinem flexiblen Konfigurationssystem ist Packer.io ein unverzichtbares Tool für jeden, der mit Serverinfrastruktur arbeitet.

4. Skopeo

Arbeiten mit Remote-Images

Skopeo ist ein Befehlszeilen-Tool, das entwickelt wurde, um mit Container-Images und Image-Repositories zu interagieren. Es ermöglicht Benutzern, Images von Containerregistern herunterzuladen, Informationen über Images zu erhalten, Images zwischen Registern und lokalen Speichern zu verschieben und vieles mehr.

4.1. Hauptmerkmale von Skopeo

- **Breite Plattformunterstützung**: Skopeo unterstützt eine Vielzahl von Containern und Image-Speicher, einschließlich Docker, OpenShift und mehr.
- **Inspektion von Images**: Skopeo kann detaillierte Informationen über ein Image ohne dessen Herunterladen oder Ausführung liefern.
- **Kopieren und Synchronisieren von Images**: Skopeo kann Images zwischen verschiedenen Registern und lokalen Speichern kopieren und synchronisieren.

4.2. Skopeo installieren und verwenden

Abhängig von deinem Betriebssystem, kann Skopeo wie folgt installiert werden:

Ubuntu und andere Linux-Distributionen: `sudo apt-get install skopeo`

Fedora: `sudo dnf install skopeo`

Nach der Installation kannst du Skopeo verwenden, um mit Container-Images zu arbeiten. Hier sind einige grundlegende Befehle und Beispiele:

4.2.1. Images inspizieren

Um Informationen über ein Image zu erhalten, verwenden Sie den `inspect` Befehl. Zum Beispiel:

```
$ skopeo inspect docker://docker.io/fedora
```

4.2.2. Images kopieren

Um ein Image von einem Register zu einem anderen zu kopieren, verwenden Sie den `copy` Befehl. Zum Beispiel:

```
$ skopeo copy docker://myregistry.com/myimage:latest
docker://myotherregistry.com/myimage:latest
```

Bitte beachte, dass Skopeo verschiedene Authentifizierungsoptionen für den Zugriff auf private Register unterstützt. Weitere Informationen finden Sie in der Skopeo-Dokumentation.

Mit diesen grundlegenden Befehlen und Konzepten bist du in der Lage, effektiv mit Skopeo zu arbeiten und deine Arbeit mit Container-Images zu optimieren.

5. Terraform

Terraform ist ein Open-Source-Tool, entwickelt von HashiCorp, das dazu dient, Infrastruktur als Code (IaC) zu definieren und bereitzustellen. Es ermöglicht Benutzern, ihre gesamte Infrastruktur (einschließlich Netzwerk, Storage, Server usw.) in Code zu definieren, der in einer Versionskontrolle gespeichert werden kann. Dieser Code kann dann verwendet werden, um die Infrastruktur auf verschiedenen Plattformen zu erstellen und zu aktualisieren.

5.1. Welche Probleme werden damit gelöst?

Terraform löst eine Reihe von Problemen im Bereich Infrastrukturmanagement:

1. **Standardisierung und Wiederverwendbarkeit**: Durch das Schreiben von Infrastruktur als Code können Teams ihre Setup-Prozesse standardisieren und Codeblöcke wiederverwenden, was zu einer effizienteren und konsistenteren Bereitstellung führt.
2. **Multi-Cloud-Deployments**: Terraform unterstützt eine Vielzahl von Service-Providern und ermöglicht es Benutzern, ihre Infrastruktur über mehrere Cloud-Plattformen hinweg zu verwalten.
3. **Vereinfachte Änderungssteuerung**: Mit Terraform können Änderungen an der Infrastruktur vor der Anwendung visualisiert und überprüft werden, was das Risiko von Ausfällen reduziert.

5.2. How to use it

Terraform verwendet eine eigene Domain Specific Language (DSL) namens HashiCorp Configuration Language (HCL), um Infrastruktur als Code zu definieren. Terraform-Prozesse werden im Allgemeinen in vier Schritten durchgeführt:

- Schreiben Sie den Infrastrukturcode in HCL und speichern Sie ihn in `.tf`-Dateien.
- Führen Sie `terraform init` aus, um das Terraform-Projekt zu initialisieren und die benötigten Provider-Plugins herunterzuladen.
- Führen Sie `terraform plan` aus, um die Änderungen zu sehen, die auf der Infrastruktur vorgenommen werden.
- Führen Sie `terraform apply` aus, um die Änderungen anzuwenden.

5.3. Beispielcodes

Ein einfacher Terraform-Code zum Erstellen einer AWS EC2-Instanz könnte folgendermaßen aussehen:

```
provider "aws" {
```

```
  region = "us-west-2"
}

resource "aws_instance" "example" {
  ami           = "ami-0c94855ba95c574c8"
  instance_type = "t2.micro"

  tags = {
    Name = "example-instance"
  }
}
```

Nach dem Schreiben dieses Codes in einer `.tf`-Datei würden Sie `terraform init`, `terraform plan` und `terraform apply` in Ihrer Befehlszeile ausführen, um die Instanz zu erstellen.

5.4. Terraform Komponenten

5.4.1. Terraform Core

Terraform Core ist die primäre Komponente von Terraform und verantwortlich für das Lesen und Interpretieren der Terraform-Konfigurationen (in `.tf` Dateien), Erstellen und Verwalten des Zustands der Ressourcen, und Aufrufen von entsprechenden Anbietern, um diese Ressourcen zu erstellen und zu ändern.

5.4.2. Terraform CLI (Command Line Interface)

Terraform CLI ist das primäre Benutzerinterface für Terraform. Es bietet Befehle zum Verwalten und Interagieren mit Terraform-Konfigurationen, Zustand und Modulen.

5.4.3. Terraform Provider

Provider sind Plugins, die von Terraform genutzt werden, um mit verschiedenen Diensten zu interagieren. Sie definieren und bieten Ressourcen an, die in Terraform-Konfigurationen erstellt und verwaltet werden können. Einige Beispiele für Provider sind AWS, Google Cloud, Azure, usw.

5.4.4. Terraform Modules

Module sind selbstständige Pakete von Terraform-Konfigurationen, die als Einheiten wiederverwendet werden können. Sie können Ressourcen enthalten, Variablen definieren und Ausgaben produzieren.

5.4.5. Terraform State

Der Terraform-Zustand ist eine wichtige Komponente, die Terraform verwendet, um den aktuellen Zustand der in den Terraform-Konfigurationen definierten Ressourcen zu verfolgen.

5.4.6. Terraform Backends

Backends sind Komponenten, die zum Speichern des Terraform-Zustands und zur Durchführung von Operationen verwendet werden. Sie ermöglichen Funktionen wie Zustandsspeicherung, Zustandsverriegelung und Umgebungssteuerung.

5.5. Erstellung eines Terraform-Moduls

Zuerst, erstellen Sie ein neues Verzeichnis, das Ihr Modul enthalten wird. Zum Beispiel, `my_module`.

```
$ mkdir my_module
```

Als nächstes, erstellen Sie eine Terraform-Konfigurationsdatei innerhalb dieses Verzeichnisses. Nennen wir sie `main.tf`.

```
variable "image_id" {
  description = "Die ID des AMI"
}

variable "availability_zone_names" {
  description = "Eine Liste der Verfügbarkeitszonen"
  type        = list(string)
}

resource "aws_instance" "example" {
  ami           = var.image_id
  instance_type = "t2.micro"

  availability_zone = var.availability_zone_names[0]
}

output "instance_public_ip" {
  value = aws_instance.example.public_ip
}
```

5.5.1. Benutzen eines Terraform-Moduls

Jetzt können wir dieses Modul in unserer Haupt-Terraform-Konfiguration verwenden. Hier ist ein Beispiel, wie das aussehen könnte:

```
module "example_module" {
  source = "./my_module"

  image_id = "ami-abc123"
  availability_zone_names = ["us-west-2a", "us-west-2b"]
}
```

Jetzt können Sie `terraform init` und `terraform apply` ausführen, um das Modul in Aktion zu sehen.

5.6. Datei auf einen Server kopieren mit Terraform

Um eine Datei auf einen Server zu kopieren, kannst du den "file" oder den "template_file" Provider von Terraform verwenden. Du musst den Inhalt der Datei bereitstellen und die Datei in deiner Terraform-Konfiguration erstellen.

Hier ist ein Beispiel, wie du eine Datei in Terraform erstellst:

Beispiel: Datei mit dem Inhalt "Hallo, Welt!"

```
resource "null_resource" "example" {
  provisioner "file" {
    content     = "Hallo, Welt!"
    destination = "/pfad/zu/deiner/datei.txt"

    connection {
      type        = "ssh"
      user        = "username"
      password    = "password"
      host        = self.public_ip
    }
  }
}
```

In diesem Beispiel wird eine Datei mit dem Inhalt "Hallo, Welt!" an den angegebenen Pfad auf dem Server kopiert.

> Arbeiten mit Passwörtern in Klartext ist ein Sicherheitsrisiko. Du solltest sichere Methoden zum Umgang mit Passwörtern verwenden, wie z.B. das Einlesen aus sicheren Speichern oder die Verwendung von SSH-Schlüsseln anstelle von Passwörtern.

Zusätzlich muss das Ziel-Server Terraform unterstützen und SSH-Zugriff ermöglichen. Du solltest sicherstellen, dass der Pfad zur Datei auf dem Zielserver existiert und schreibbar ist.

Falls du eine existierende Datei kopieren möchtest, kannst du die `source` Eigenschaft anstelle von `content` verwenden. Zum Beispiel:

Beispiel mit bereits vorhandener Datei

```
resource "null_resource" "example" {
  provisioner "file" {
    source      = "/pfad/zu/lokal/datei.txt"
    destination = "/pfad/zu/ziel/datei.txt"

    connection {
      type        = "ssh"
```

```
      user        = "username"
      password    = "password"
      host        = self.public_ip
    }
  }
}
```

Hierbei wird eine lokale Datei auf deinem Rechner an den angegebenen Pfad auf dem Server kopiert.

Ein weiteres Beispiel unter Verwendung von einem SSH-Key

```
resource "null_resource" "example" {
  provisioner "file" {
    content     = "Hallo, Welt!"
    destination = "/pfad/zu/deiner/datei.txt"

    connection {
      type        = "ssh"
      user        = "username"
      private_key = file("~/.ssh/id_rsa")
      host        = self.public_ip
    }
  }
}
```

Das Terraform-Verhalten kann sich abhängig von der spezifischen Serverkonfiguration und den verwendeten Terraform-Provisionern ändern. Dieses Beispiel könnte angepasst werden müssen, um in deiner spezifischen Umgebung zu funktionieren.

5.7. Vor- und Nachteile von Terraform

Wie jedes Tool hat auch Terraform seine Vor- und Nachteile:

Vorteile

- **Provider-übergreifend**: Terraform unterstützt eine Vielzahl von Providern, sowohl Cloud als auch On-Premises.
- **Immutable Infrastructure**: Terraform erstellt und verwaltet Ressourcen auf eine Weise, die Änderungen an der bestehenden Infrastruktur minimiert.
- **Einfach zu lernen**: HCL ist eine recht einfache und lesbare Sprache.

Nachteile

- **Fehler können schwerwiegend sein**: Ein Fehler in Ihrem Terraform-Code kann zu großen Problemen in Ihrer Infrastruktur führen.

- **Komplexität bei großen Setups**: Während Terraform bei kleineren Projekten einfach zu verwenden ist, kann es bei großen und komplexen Setups schwierig sein, den Überblick zu behalten.
- **Fehlende Unterstützung für bestimmte Ressourcen**: Während Terraform viele Provider unterstützt, gibt es immer noch Ressourcen und Dienste, die nicht unterstützt werden.
- **Keine Multiuser/Platform unterstützung**: Per default werden die "state" - Files lokal abgelegt. Dadurch ist es nicht einfach so möglich, Änderungen von einem anderen Rechner oder anderem User durchzuführen. Hierzu müssen "backends" für die Provider definiert werden. Mögliche Backends sind Cloud-Speicher bei: Amazon, Google, Azure oder ein Cloud-Dienst von HashiCorp (Terraform). Eine weitere Alternative ist die Nutzung von GitLab als backend - mehr dazu im nächsten Kapitel.

5.8. Zusammenfassung

Abschließend ist Terraform ein leistungsstarkes Tool zur Verwaltung Ihrer Infrastruktur als Code. Mit seiner Fähigkeit, eine breite Palette von Anbietern zu unterstützen und den Infrastrukturprozess zu standardisieren, ist es ein unverzichtbares Tool in der modernen DevOps-Werkzeugkette. Es ist jedoch wichtig, sorgfältig mit Terraform umzugehen, um mögliche Fehler zu vermeiden, die Auswirkungen auf die Produktionsinfrastruktur haben könnten.

5.9. Links / Cheatsheet

- Introducing Terramate — An Orchestrator and Code Generator for Terraform

5.10. GitLab als Terraform Backend

coming soon

- A complete overview of **GitLab managed terraform state**
- How to run **terraform script** using **GitLab CI/CD**?

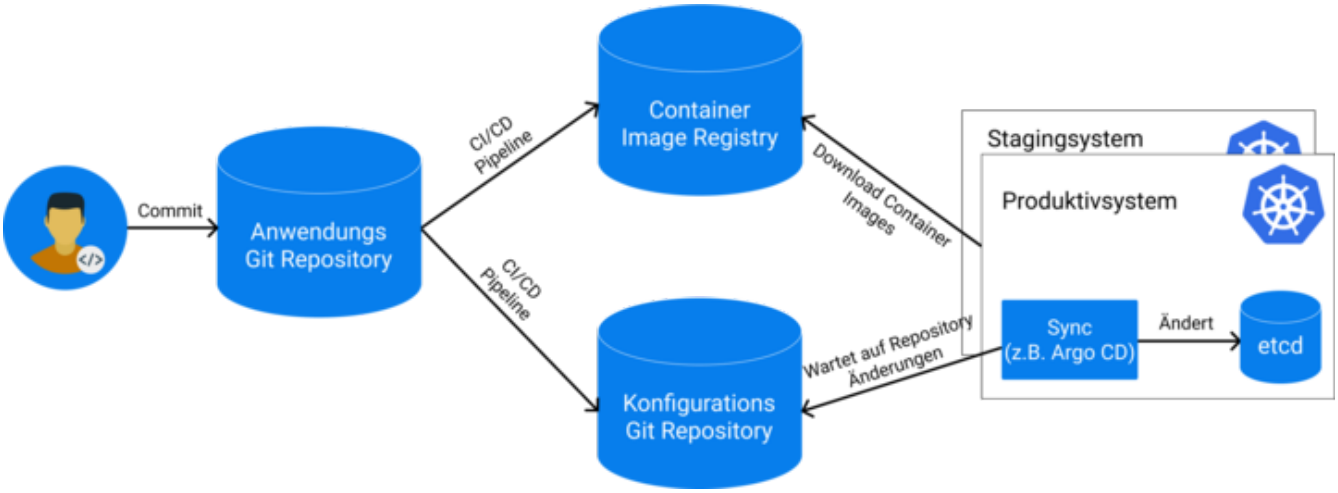
6. ArgoCD

6.1. Welches Problem wird mit ArgoCD gelöst?

Anwendungsdefinitionen, Konfigurationen und Umgebungen sollten deklarativ und versioniert sein.

Die Anwendungsbereitstellung und das Lebenszyklusmanagement sollten automatisiert, überprüfbar und leicht verständlich sein.

6.2. Überblick



6.3. Wie arbeitet ArgoCD?

- Deklarativ - beschreibt Zielbild, nicht den Weg
- Arbeitet mit Kubernetes und OpenShift
- Kann folgende Quellen verarbeiten:
  - kubernetes manifests
  - Helm Charts
  - Kustomize resources

6.4. Ways to interact with ArgoCD

- Web GUI
- ArgoCD CLI
- Kubernetes Manifest files

6.5. Installation von ArgoCD

```
kubectl create namespace argocd
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-
cd/stable/manifests/install.yaml
```

Initiales Admin Password abfragen

```
kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath
="{.data.password}" | base64 -d; echo
```

6.6. Webhook

- https://argo-cd.readthedocs.io/en/stable/operator-manual/webhook/

6.7. Sync-Waves

tbd.

6.8. Bootstrap - Project

- https://github.com/argoproj/argocd-example-apps/tree/master/apps
- Automatically create multiple applications in Argo CD

6.9. High Availability

High Availability installation is recommended for production use. This bundle includes the same components but tuned for high availability and resiliency.

- `ha/install.yaml` - the same as install.yaml but with multiple replicas for supported components.
- `ha/namespace-install.yaml` - the same as namespace-install.yaml but with multiple replicas for supported components.

6.10. Manage multible Cluster

6.11. Workflow Hardening

- Practical Argo Workflows Hardening

6.12. Disaster Recovery

- https://argo-cd.readthedocs.io/en/stable/operator-manual/disaster_recovery/

6.13. Bonus

- ArgoCD Custom Plugins - Creating a Custom Plugin to Process OpenShift Templates
- https://medium.com/@geoffrey.muselli/argocd-multi-cluster-helm-charts-installation-in-mono-repo-0a406ff7c578

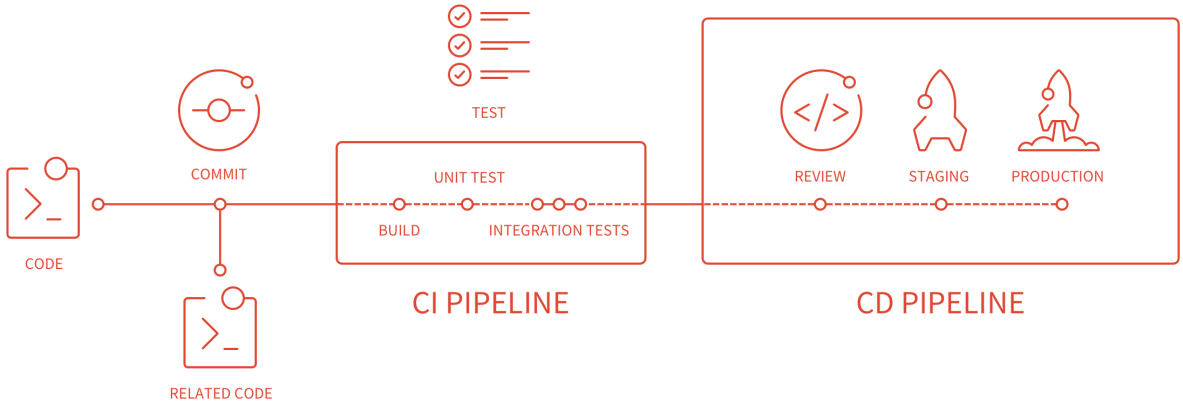
6.14. ArgoCD und DevOps

https://codefresh.io/blog/using-argo-cd-and-kustomize-for-configmap-rollouts

6.15. Links / Cheatsheet

- Solving ArgoCD Secret Management with the argocd-vault-plugin
- External HTTPS SSO Callback Ingress
- How to Deploy Argo CD Dashboard over Nginx Ingress-Controller

7. GitLab Pipelines



- vordefinierter Aktionen, die bei Events starten
- Beispiel: bei Commit startet Testaktion

7.1. Gitlab Runner

- Der Gitlab-CI-Server, auch Koordinator genannt, führt selbst keine Builds aus, sonder delegiert dies an sogenannte Runner.
- Ein Runner ist ein Prozess, der auf einem beliebigen Rechner laufen kann, und den Koordinator pollt, um anstehende Jobs abzuholen und zu bearbeiten.
- Der Runner kann direkt installiert sein oder als Container gestartet werden.
- Gitlab Runner in Docker

7.2. Pipeline - Get started

- GitLab CI/CD Configfile `.gitlab-ci.yml`

```
stages:
  - build
  - test

build-code-job:
  stage: build
  script:
    - ruby -v
    - rake

test-code-job1:
  stage: test
  script:
    - echo "If the files are built successfully, test:"
```

```
    - rake test1
```

7.3. Pipeline - Stages

Die "Stages" - Liste gruppiert die Jobs und definiert die Reihenfolge der Ausführung.

```
stages:
  - build
  - test
  - deploy
```

7.4. Pipeline - Jobs

Jobs sind der fundamentale Bestandteil von Pipelines

- in Jobs wird definiert, was ausgeführt werden soll
- müssen mindestens das Element "script" enthalten
- können beliebige Namen haben
- sind in der Anzahl nicht begrenzt
- max execution time = **60min** per job

7.5. Pipeline - Job example

```
node-lint:
  image: $NODE_BASE_IMAGE
  stage: test
  script:
    - mkdir output
    - cd app
    - npm install --silent
    - npx eslint ./ --fix -f html -o ../output/lint-report.html
  artifacts:
    paths:
      - output/lint-report.html
```

7.6. Links to follow

https://hilfe.uni-paderborn.de/GitLab_-_CI/CD

8. Gitlab pipelines in advanced

Um eine GitLab-Pipeline zu erstellen, die mit mehreren Projekten arbeitet, können Sie die folgenden Schritte befolgen:

8.1. Projektübergreifende Pipelines definieren

Sie können in GitLab projektübergreifende Pipelines definieren, indem Sie die `trigger`-Anweisung in Ihrer `.gitlab-ci.yml`-Datei verwenden. Diese ermöglicht es einem Projekt, eine Pipeline in einem anderen Projekt auszulösen.

8.2. GitLab CI/CD-Konfigurationsdatei erstellen

In jedem beteiligten Projekt benötigen Sie eine `.gitlab-ci.yml`-Datei, die die Pipeline definiert. Diese Datei legt die Jobs und Stufen fest, die in der Pipeline ausgeführt werden sollen.

8.3. Trigger einrichten

In der `.gitlab-ci.yml` des auslösenden Projekts verwenden Sie die `trigger`-Anweisung, um die Pipeline eines anderen Projekts zu starten. Sie können beispielsweise angeben, welche spezifische Pipeline eines anderen Projekts gestartet werden soll und unter welchen Bedingungen dies geschehen soll.

8.4. Abhängigkeiten zwischen Projekten verwalten

Wenn Ihre Projekte voneinander abhängig sind, z. B. wenn ein Projekt ein Artefakt erstellt, das von einem anderen Projekt verwendet wird, müssen Sie diese Abhängigkeiten in Ihren Pipelines entsprechend verwalten. Dies kann durch das Übergeben von Artefakten zwischen Pipelines oder durch die Verwendung von gemeinsamen Speicherorten wie einem Artefakt-Repository erfolgen.

8.5. Zugriffsrechte konfigurieren

Stellen Sie sicher, dass die Projekte die erforderlichen Berechtigungen haben, um Pipelines in anderen Projekten auszulösen. Dies kann Zugriffstoken oder spezielle Berechtigungskonfigurationen umfassen.

8.6. Pipeline-Status überwachen und debuggen

Nachdem Sie die Pipelines eingerichtet haben, sollten Sie den Fortschritt überwachen und eventuelle Probleme debuggen. GitLab bietet eine visuelle Darstellung des Pipeline-Status, sowie detaillierte Logs für jeden Job.

Beachten Sie, dass die genaue Konfiguration von Ihrem spezifischen Anwendungsfall und der Struktur Ihrer Projekte abhängt. Die GitLab-Dokumentation bietet detaillierte Anleitungen und Beispiele, die Ihnen helfen können, Ihre Pipeline-Konfiguration zu optimieren.

8.7. Beispiel für projektübergreifende Pipelines

Hier ist ein einfaches Beispiel, wie man eine GitLab-CI-Pipeline konfigurieren kann, die mit mehreren Projekten arbeitet:

8.7.1. Projekt A: Hauptprojekt

Stellen Sie sich vor, Projekt A ist Ihr Hauptprojekt, das eine Pipeline in Projekt B auslöst.

gitlab-ci.yml in Projekt A

```yaml
stages:
  - build
  - trigger

build_job:
  stage: build
  script:
    - echo "Building Project A..."

trigger_project_b:
  stage: trigger
  script:
    - echo "Triggering pipeline in Project B..."
  trigger:
    project: your-group/project-b
    branch: master
```

In diesem Beispiel gibt es zwei Stufen: `build` und `trigger`. Der `build_job` führt einen einfachen Befehl aus (z.B. den Build-Prozess), und `trigger_project_b` löst eine Pipeline im Projekt B aus.

8.7.2. Projekt B: Abhängiges Projekt

Projekt B könnte ein abhängiges Projekt sein, das durch Projekt A ausgelöst wird.

gitlab-ci.yml in Projekt B

```yaml
stages:
  - test

test_job:
  stage: test
  script:
    - echo "Testing Project B..."
```

In Projekt B gibt es eine einfache Pipeline mit einer Teststufe. Diese Pipeline wird ausgelöst, sobald die `trigger_project_b`-Stufe in Projekt A erfolgreich abgeschlossen ist.

8.7.3. Hinweise

- Ersetzen Sie `your-group/project-b` mit dem tatsächlichen Pfad Ihres Projekts B in GitLab.
- Stellen Sie sicher, dass für das Projekt, das die Pipeline eines anderen Projekts auslöst, die entsprechenden Zugriffsrechte eingerichtet sind. Eventuell benötigen Sie ein [CI/CD-Token](https://docs.gitlab.com/ee/ci/triggers/#adding-a-new-trigger).
- Dieses Beispiel ist grundlegend. Je nach Anforderungen Ihres Projekts können Sie komplexere Pipelines mit weiteren Stufen, Jobs und Bedingungen einrichten.

Diese Konfiguration ermöglicht eine einfache Interaktion zwischen zwei Projekten, wobei das eine Projekt (Projekt A) eine Aktion in einem anderen Projekt (Projekt B) auslöst.

9. Jenkins

Ein Überblick

9.1. Einführung

Jenkins ist ein freies Open Source Automatisierungstool, das hauptsächlich mit Java entwickelt wurde. Es dient zur kontinuierlichen Integration und kontinuierlichen Bereitstellung (CI/CD) von Projekten.

9.2. Hauptmerkmale von Jenkins

Jenkins bietet zahlreiche Funktionen, die seine Verwendung für DevOps und CI/CD-Prozesse attraktiv machen:

- Einfache Installation: Jenkins ist ein selbstständiges Java-Programm, das direkt aus dem Paket auf jedem System ausgeführt werden kann, auf dem Java installiert ist.
- Plugin-Ökosystem: Es gibt Tausende von Jenkins-Plugins, die Integrationen mit fast jedem Tool im DevOps-Lebenszyklus ermöglichen.
- Skalierbarkeit: Jenkins kann horizontal und vertikal skaliert werden, um den Anforderungen großer und komplexer Projekte gerecht zu werden.
- Pipeline als Code: Jenkins ermöglicht es Benutzern, ihre CI/CD-Pipeline als Code zu definieren. Dies verbessert die Wartbarkeit und Versionskontrolle der Pipeline.

9.3. Jenkins-Architektur

Jenkins folgt einer Master-Slave-Architektur, um den Arbeitslast auszugleichen:

- Master: Der Master koordiniert die Builds und verteilt die Aufgaben an die Slaves. Es speichert auch Konfigurationsdetails und stellt die Benutzeroberfläche und die API bereit.

- Slaves: Die Slaves führen die Aufgaben aus, die ihnen vom Master zugewiesen wurden. Sie können auf verschiedenen Betriebssystemen laufen und unterschiedliche Hardwarekonfigurationen haben, je nach den Anforderungen der Builds.

9.4. Fazit

Jenkins ist ein äußerst vielseitiges und leistungsfähiges Tool für die Automatisierung von DevOps-Aufgaben. Sein großes Plugin-Ökosystem und seine Skalierbarkeit machen es zu einer hervorragenden Wahl für Teams jeder Größe.

9.5. Links / Cheatsheet

- Creating CI/CD Pipeline with Jenkins

10. Tekton

Eine Einführung, Anwendung und Analyse

10.1. Einführung

Tekton ist ein leistungsstarkes und flexibles Open-Source-Framework für die Erstellung von Continuous Integration und Continuous Delivery (CI/CD) Systemen. Entwickelt, um Kubernetes nativ zu sein, stellt Tekton eine Reihe von Kubernetes Custom Resource Definitions (CRDs) zur Verfügung, um Pipelines zu erstellen, die passend zu den modernen softwareentwicklungspraktiken sind.

10.2. Welche Probleme werden damit gelöst?

Tekton löst eine Reihe von Herausforderungen im Bereich der Softwarelieferung:

- **Kompatibilität**: Da Tekton auf Kubernetes basiert, kann es auf jeder Plattform eingesetzt werden, die Kubernetes unterstützt.
- **Anpassbarkeit**: Tekton ist hochgradig anpassbar und kann sich leicht an verschiedene CI/CD-Workflows anpassen.
- **Wiederverwendbarkeit**: Tekton-Aufgaben sind modular und wiederverwendbar, was bedeutet, dass Teams einmal erstellten Code in mehreren Pipelines verwenden können.

10.3. Wie benutzt man Tekton?

Die Verwendung von Tekton beinhaltet im Wesentlichen das Definieren und Ausführen von `Tasks` und `Pipelines`. Hier ist ein einfacher Ablauf zur Einrichtung eines Tekton-Workflows:

1. Installieren Sie Tekton auf Ihrem Kubernetes-Cluster.
2. Definieren Sie einen `Task`, der eine bestimmte Aufgabe ausführt.
3. Definieren Sie eine `Pipeline`, die mehrere `Tasks` verbindet.
4. Erstellen Sie einen `PipelineRun`, um die `Pipeline` auszuführen.

10.4. Beispielcodes

Hier ist ein einfacher Tekton Task, der ein Docker-Image erstellt:

```
apiVersion: tekton.dev/v1beta1
kind: Task
metadata:
  name: build-docker-image
spec:
  steps:
  - name: build-and-push
    image: docker:17.12.0-ce
    command: ["docker", "build", "-t", "my-image", "."]
```

10.5. Vor- und Nachteile

Wie jedes Tool hat auch Tekton seine Vor- und Nachteile:

10.5.1. Vorteile

- **Flexibilität**: Tekton bietet eine hohe Flexibilität bei der Gestaltung von CI/CD-Workflows.
- **Wiederverwendbarkeit**: Aufgaben in Tekton sind wiederverwendbar, was zur Effizienz der Pipelines beiträgt.

10.5.2. Nachteile

- **Komplexität**: Tekton kann für Einsteiger komplex sein, besonders wenn man nicht mit Kubernetes vertraut ist.
- **Fehlende Benutzeroberfläche**: Tekton selbst hat keine Benutzeroberfläche, obwohl es Drittanbieter-Optionen gibt.

Zusammenfassend lässt sich sagen, dass Tekton ein mächtiges Werkzeug für CI/CD-Pipelines ist, das sich durch seine Flexibilität und Wiederverwendbarkeit auszeichnet. Es hat jedoch eine steile Lernkurve und erfordert eine gute Kenntnis von Kubernetes.

10.6. Links

- Tekton CI/CD review

11. Kubernetes

Kubernetes, ein weitverbreitetes System zur Orchestrierung von Containeranwendungen, besteht aus verschiedenen Komponenten, die gemeinsam eine robuste und skalierbare Plattform bilden. Hier sind die wichtigsten Komponenten:

11.1. Master-Knoten (Master Node)

Er steuert den Kubernetes-Cluster und besteht aus mehreren Teilen:

- **API-Server** (kube-apiserver):
  Dient als Frontend für das Kubernetes-Steuerungsebenen-Netzwerk.
- **Etcd**:
  Eine konsistente und hochverfügbare Schlüsselwert-Datenbank, die als Kubernetes' Backing-Store für alle Clusterdaten genutzt wird.
- **Scheduler** (kube-scheduler):
  Entscheidet, auf welchem Knoten neu erstellte Container platziert werden.
- **Controller-Manager** (kube-controller-manager):
  Verwaltet die Controller, die den Zustand des Clusters überwachen und bei Bedarf Änderungen vornehmen.

11.2. Arbeitsknoten (Worker Nodes):

Diese Knoten führen die Containeranwendungen aus. Sie enthalten:

- **Kubelet**: Eine Agent-Anwendung, die sicherstellt, dass die Container in einem Pod laufen.
- **Kube-Proxy**: Ein Netzwerk-Proxy, der die Kubernetes-Netzwerkdienste auf dem Arbeitsknoten verwaltet.
- **Container-Runtime**: Die Software, die für das Ausführen von Containern verantwortlich ist (z.B. Docker, containerd).

11.3. Pods

Die kleinste Einheit, die in Kubernetes erstellt und verwaltet wird. Ein Pod ist eine Gruppe von einem oder mehreren Containern, die Ressourcen wie Netzwerk und Speicherplatz teilen.

11.4. Deployment und ReplicaSets

Diese Komponenten ermöglichen es Ihnen, den gewünschten Zustand Ihrer Anwendung zu definieren und Kubernetes kümmert sich um dessen Einhaltung.

11.5. Services

Eine Abstraktion, die einen logischen Satz von Pods definiert und eine Policy, um auf sie

zuzugreifen.

11.6. Namespaces

Erlauben die Unterteilung von Ressourcen in verschiedenen virtuellen Clustern im selben physischen Cluster.

11.7. ConfigMaps und Secrets

Für die Speicherung von Konfigurationsdaten und sensiblen Informationen, die von Pods genutzt werden können.

Diese Komponenten arbeiten zusammen, um eine hochverfügbare, skalierbare und flexible Umgebung für das Ausführen von containerisierten Anwendungen zu bieten. Kubernetes' Architektur ermöglicht es, Anwendungen effizient und zuverlässig zu verwalten, zu skalieren und zu verteilen.

11.8. Ingress / Egress

In Kubernetes und im Bereich der Netzwerkkommunikation beziehen sich die Begriffe "Ingress" und "Egress" auf den Datenverkehr, der in das Netzwerk eintritt oder es verlässt. Hier sind die Hauptunterschiede:

11.8.1. Ingress:

- **Bedeutung**:
  Ingress bezieht sich auf den eingehenden Netzwerkverkehr. In einem Kubernetes-Kontext bezeichnet es oft die Regeln und Mechanismen, die den Zugriff von außen auf Dienste innerhalb des Kubernetes-Clusters steuern.
- **Verwendung in Kubernetes**:
  In Kubernetes ist ein Ingress eine API-Ressource, die den Zugriff auf HTTP- und HTTPS-Routen von außerhalb des Clusters zu den Services innerhalb des Clusters steuert. Es ermöglicht Ihnen, Zugriffsregeln zu definieren, Hostnamen oder URL-Pfade auf bestimmte Services abzubilden und sogar SSL/TLS-Zertifikate für diese Endpunkte zu handhaben.
- **Beispiel**:
  Ein Ingress könnte konfiguriert werden, um Anfragen an meine-website.example.com an einen spezifischen Service in Ihrem Kubernetes-Cluster weiterzuleiten.

11.8.2. Egress:

- **Bedeutung**:
  Egress bezieht sich auf den ausgehenden Netzwerkverkehr, also den Datenverkehr, der von Ihrem Netzwerk (z.B. einem Kubernetes-Cluster) zu einem externen Ziel fließt.
- **Verwendung in Kubernetes**:
  In Kubernetes kontrollieren Egress-Regeln, wie der ausgehende Verkehr von den Pods in einem Cluster zu externen Diensten geleitet wird. Dies kann wichtig sein, um die Netzwerksicherheit

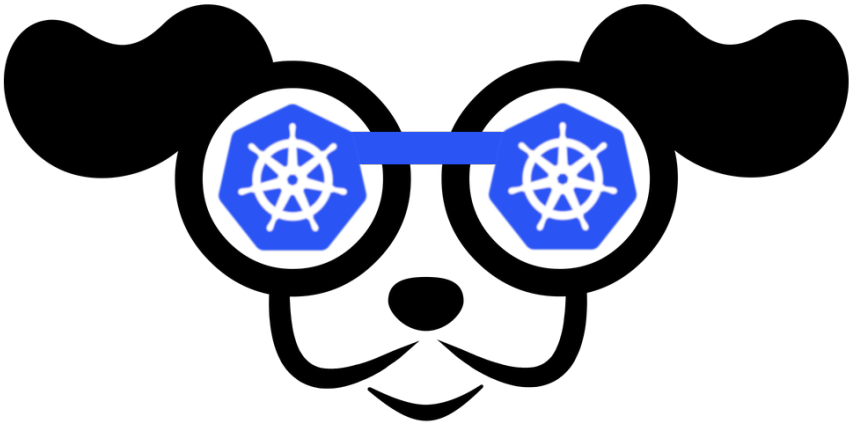
zu gewährleisten oder um zu kontrollieren, wie Ressourcen außerhalb des Clusters genutzt werden.

- **Beispiel**:
  Egress-Regeln könnten festlegen, dass bestimmte Pods Zugriff auf eine externe Datenbank oder eine API im Internet haben, während anderer Verkehr blockiert wird.

In der Praxis sind Ingress- und Egress-Kontrollen wesentliche Bestandteile des Netzwerkmanagements und der Sicherheit in Kubernetes, da sie detailliert steuern, wie der Datenverkehr in und aus dem Cluster fließt.

12. k9s

Kubernetes CLI To Manage Your Clusters



K9s is a terminal based UI to interact with your Kubernetes clusters. The aim of this project is to make it easier to navigate, observe and manage your deployed applications in the wild. K9s continually watches Kubernetes for changes and offers subsequent commands to interact with your observed resources.

12.1. Installation

K9s is available on Linux, macOS and Windows platforms.

Binaries for Linux, Windows and Mac are available as tarballs in the release page.

12.2. Features

12.2.1. Information At Your Finger Tips!

- Tracks in real-time activities of resources running in your Kubernetes cluster.

12.2.2. Standard or CRD?

- Handles both Kubernetes standard resources as well as custom resource definitions.

12.2.3. Cluster Metrics

- Tracks real-time metrics associates with resources such as pods, containers and nodes.

12.2.4. Power Users Welcome!

- Provides standard cluster management commands such as logs, scaling, port-forwards, restarts...
- Define your own command shortcuts for quick navigation via command aliases and hotkeys.
- Plugin support to extend K9s to create your very own cluster commands.

- Powerful filtering mode to allow user to drill down and view workload related resources.

12.2.5. Error Zoom

- Drill down directly to what's wrong with your cluster's resources.

12.2.6. Skinnable and Customizable

- Define your very own look and feel via K9s skins.
- Customize/Arrange which columns to display on a per resource basis.

12.2.7. Narrow or Wide?

- Provides toggles to view minimal or full resource definitions

12.2.8. MultiResources Views

- Provides for an overview of your cluster resources via Pulses and XRay views.

12.2.9. We've got your RBAC!

- Supports for viewing RBAC rules such as cluster/roles and their associated bindings.
- Reverse lookup to asserts what a user/group or ServiceAccount can do on your clusters.

12.2.10. Built-in Benchmarking

- You can benchmark your HTTP services/pods directly from K9s to see how your application fare and adjust your resources request/limit accordingly.

12.2.11. Resource Graph Traversals

- K9s provides for easy traversal of Kubernetes resources and their associated resources.

12.3. Links

- https://k9scli.io

13. kURL

Open Source Kubernetes Installer

[kurl logo@2x] | *https://kurl.sh/kurl_logo@2x.png*

"kURL - Kubernetes Installer" ist ein Tool, das zur Vereinfachung der Installation und Bereitstellung von Kubernetes-Clustern entwickelt wurde. Es bietet eine automatisierte Methode, um Kubernetes auf verschiedenen Plattformen einzurichten.

Die Verwendung von kURL erfolgt in mehreren Schritten:

13.1. Vorbereitung der Infrastruktur

Stellen Sie sicher, dass die Infrastruktur für Ihren Kubernetes-Cluster bereit ist. Dies umfasst das Einrichten von Servern oder virtuellen Maschinen, auf denen Kubernetes installiert werden soll.

13.2. Herunterladen von kURL

Laden Sie das kURL-Installationsprogramm herunter. Dieses Programm enthält die erforderlichen Skripte und Konfigurationsdateien, um die Kubernetes-Installation durchzuführen.

Zusätzlich besteht die Möglichkeit, nicht nur die Scripte, sondern auch alle benötigten Resourcen, in eine tar-File herunter zu laden. Damit ist dann auch eine einfache Installetion in einer Airgap-Umgebung möglich.

13.3. Konfiguration

Passen Sie die Konfigurationsdateien an Ihre spezifischen Anforderungen an. Dies umfasst die Festlegung von Netzwerkeinstellungen, Authentifizierungsoptionen, Speicheroptionen usw.

13.4. Installation

Führen Sie das kURL-Installationsprogramm aus und geben Sie die angepassten Konfigurationsdateien an. Das Installationsprogramm führt dann den Prozess der Kubernetes-Installation durch, einschließlich der Installation von Docker, der Einrichtung des Kubernetes-Master-Knotens und der Bereitstellung der Worker-Knoten.

13.5. Überprüfung

Nach Abschluss der Installation können Sie den Status des Kubernetes-Clusters überprüfen, um sicherzustellen, dass alles korrekt eingerichtet wurde. Dies umfasst die Überprüfung der Verfügbarkeit der Kubernetes-API und das Testen der Kommunikation zwischen den Clusterknoten.

13.6. Fazit

KURL ist ein flexibles Tool, das auf verschiedene Szenarien und Plattformen zugeschnitten werden kann. Es ermöglicht eine schnelle und effiziente Installation von Kubernetes-Clustern mit einem standardisierten Ansatz. Nicht zuletzt bietet es eine einfache Möglichkeit für Installationen in Airgap-Umgebungen.

13.7. Links

- https://kurl.sh

14. Podman

14.1. Einleitung

Podman ist ein Tool zur Containerverwaltung, das von Red Hat entwickelt wurde und als Alternative zu Docker dient.

14.2. Vorteile gegenüber Docker

- Daemon-los
- Verbesserte Sicherheit durch Root-losen Betrieb
- Aufgeteilte Befehle für verschiedene Aufgaben

14.3. Installation

14.3.1. macOS

- Installation über Homebrew: `brew install podman`
- VM-Initialisierung: `podman machine init` und `podman machine start`

14.3.2. Linux

- Ubuntu: `sudo apt-get -y install podman`
- Fedora: `sudo dnf -y install podman`

14.4. Häufig verwendete Befehle

- `podman pull`: Image herunterladen
- `podman run`: Befehl in neuem Container ausführen
- `podman ps`: Container auflisten
- `podman exec`: Prozess in laufendem Container ausführen
- `podman stop`: Container stoppen

14.5. Links

- https://podman.io

15. Trivy

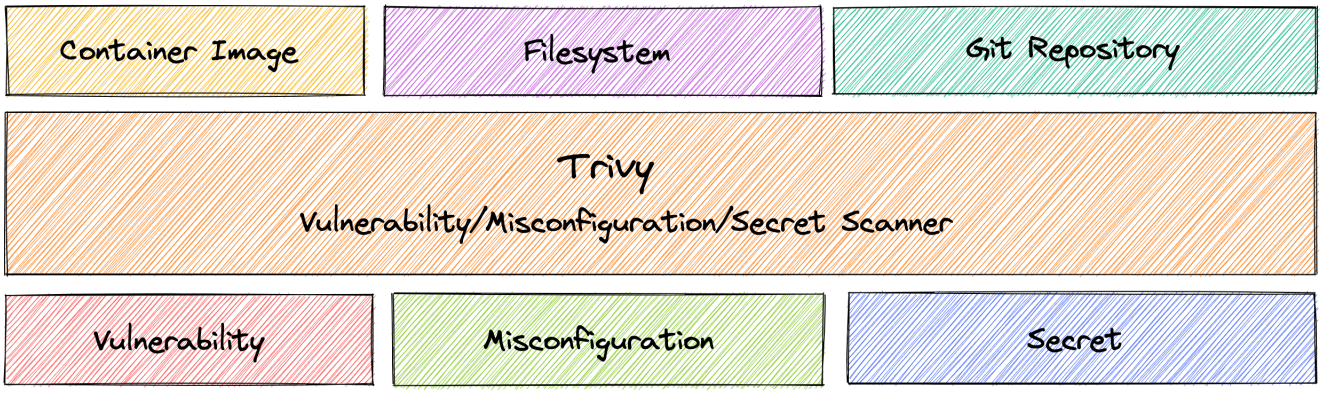
Vulnerability Scanner

Im stetig wachsenden digitalen Zeitalter spielen Sicherheit und Datenschutz eine immer wichtigere Rolle. Ein kritischer Aspekt davon ist die Aufdeckung und Behebung von Sicherheitslücken in Software, auch als Vulnerabilities bekannt. Ein Tool, das sich dabei als besonders nützlich erweist, ist der Trivy-Scanner.

15.1. Was ist Trivy?

Trivy, entwickelt von Aqua Security, ist ein umfassender und einfach zu bedienender Vulnerability-Scanner für Container und andere Artefakte. Es wurde mit dem Fokus auf Komfort und Effizienz entwickelt, ohne dabei auf Präzision und Zuverlässigkeit zu verzichten.

Trivy ist leicht zu installieren und kann sowohl auf der Kommandozeile als auch in der CI/CD-Pipeline genutzt werden. Es hat eine umfangreiche Abdeckung von Betriebssystemen und Sprachpaketen und liefert genaue Ergebnisse, indem es sowohl Betriebssystem- als auch Sprachspezifische Schwachstellen aufdeckt.



15.2. Warum Trivy?

Das Besondere an Trivy ist seine einfache Handhabung. Es erfordert keine aufwendige Konfiguration und ist daher besonders benutzerfreundlich. Die Benutzer müssen nur den Namen des Containers oder des Repositories angeben, und Trivy kümmert sich um den Rest.

Darüber hinaus besticht Trivy durch seine geringe False-Positive-Rate. Dies ist von entscheidender Bedeutung, da ein übermäßig hoher False-Positive-Rate die Effektivität eines Vulnerability-Scanners erheblich einschränken kann. Durch die Verwendung eines umfassenden Schwachstellendatensatzes und einer genauen Matching-Logik kann Trivy eine genaue und effiziente Analyse bieten.

15.3. Codebeispiele

Hier sind ein paar Beispiele, wie Sie den Trivy-Scanner in Ihren Code integrieren können.

15.3.1. Scannen eines Docker-Images

Das einfachste Beispiel ist das Scannen eines Docker-Images. Hier ist ein Befehl, um ein Image zu scannen:

```
trivy image [Optionen] ImageName
```

Beispiel:

```
trivy image python:3.7-alpine
```

15.3.2. Scannen eines Dateisystems

Sie können auch ein bestimmtes Dateisystem mit Trivy scannen:

```
trivy fs /pfad/zum/dateisystem
```

15.3.3. Integration in eine CI/CD Pipeline

Trivy kann auch in CI/CD-Pipelines integriert werden. Hier ist ein einfaches Beispiel für die Integration in eine GitHub Actions Pipeline:

```
name: CI

on:
  push:
    branches: [ master ]

jobs:
  trivy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Run Trivy vulnerability scanner
        uses: aquasecurity/trivy-action@master
        with:
          image-ref: 'python:3.7-alpine'
          format: 'template'
          template: '@/contrib/sarif.tpl'
          output: 'trivy-results.sarif'

      - name: Upload Trivy scan results to GitHub Security tab
        uses: github/codeql-action/upload-sarif@v1
        with:
          sarif_file: 'trivy-results.sarif'
```

15.4. Zusammenfassung

Zusammenfassend lässt sich sagen, dass Trivy eine hervorragende Wahl für Entwickler und Sicherheitsteams ist, die einen gründlichen und zuverlässigen Vulnerability-Scanner benötigen. Mit seiner einfachen Handhabung, genauen Ergebnissen und weitreichenden Abdeckung ist Trivy ein unverzichtbares Tool in der modernen Softwareentwicklung.

Es ist klar, dass die Bedeutung von Vulnerability Scanning in der heutigen Welt nicht genug betont werden kann. Und mit Tools wie Trivy wird diese Aufgabe um einiges einfacher und effektiver. Mit kontinuierlichen Updates und Verbesserungen bleibt Trivy auch weiterhin ein Vorreiter auf dem Gebiet der Sicherheit im Bereich Softwareentwicklung.

Indem wir die Risiken erkennen und aktiv angehen, können wir sicherstellen, dass unsere digitalen Lösungen sicher bleiben und weiterhin Vertrauen und Zuverlässigkeit bieten. Dabei ist Trivy ein wichtiger Verbündeter.

15.5. Links

- https://aquasecurity.github.io/trivy/dev/

16. Asciidoctor

AsciiDoc ist eine vereinfachte Auszeichnungssprache, die dazu dient, Texte in verschiedenen Dokumentenformaten zu veröffentlichen.

AsciiDoc hat den Vorteil, leicht erlernbar zu sein und auch unverarbeitet (als Quelltext) gut lesbar zu sein.

16.1. Welches Problem wird damit gelöst?

- docx, pdf nicht in GIT / Versionskontrolle verwaltbar
- Erstellung von Dokument in verschiedenen Formaten ist aufwendig
- Ergebnis nicht immer korrekt reproduzierbar
- Anpassungen benötigen oft spezielle Software (Office, Visio, usw.)

16.2. Wie löst Aciidoc diese Probleme?

- Asciidoc Dokumente sind in Rohform lesbar
- Doc as code
- Presentation as code
- Keine spezielle Software notwendig
- Ausgabe in verschiedene Formate möglich
- Formatierung über Templates

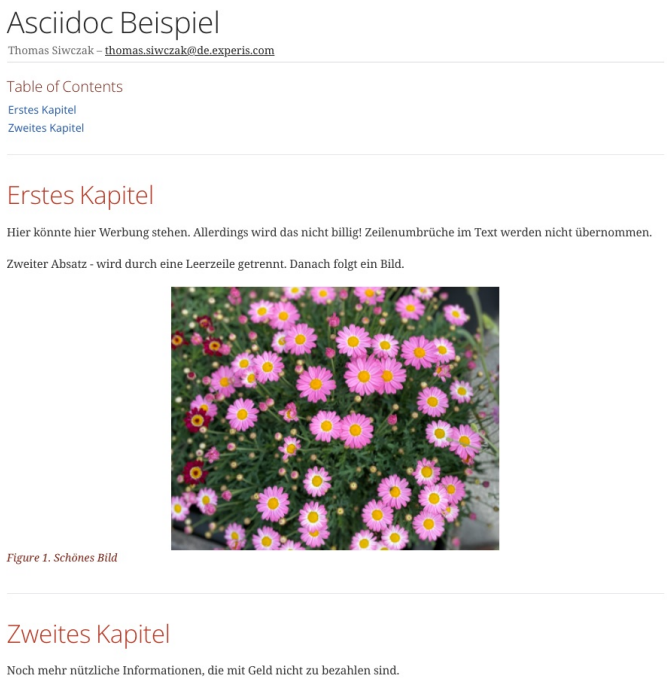
16.3. Asciidoc Code - Example

```
= Asciidoc Beispiel
:Author: Thomas Siwczak
:Email: thomas.siwczak@de.experis.com
:Date: 11.05.2022
:Revision: 1.2.3
:data-uri: true
:toc: // Inhaltsverzeichnis
```

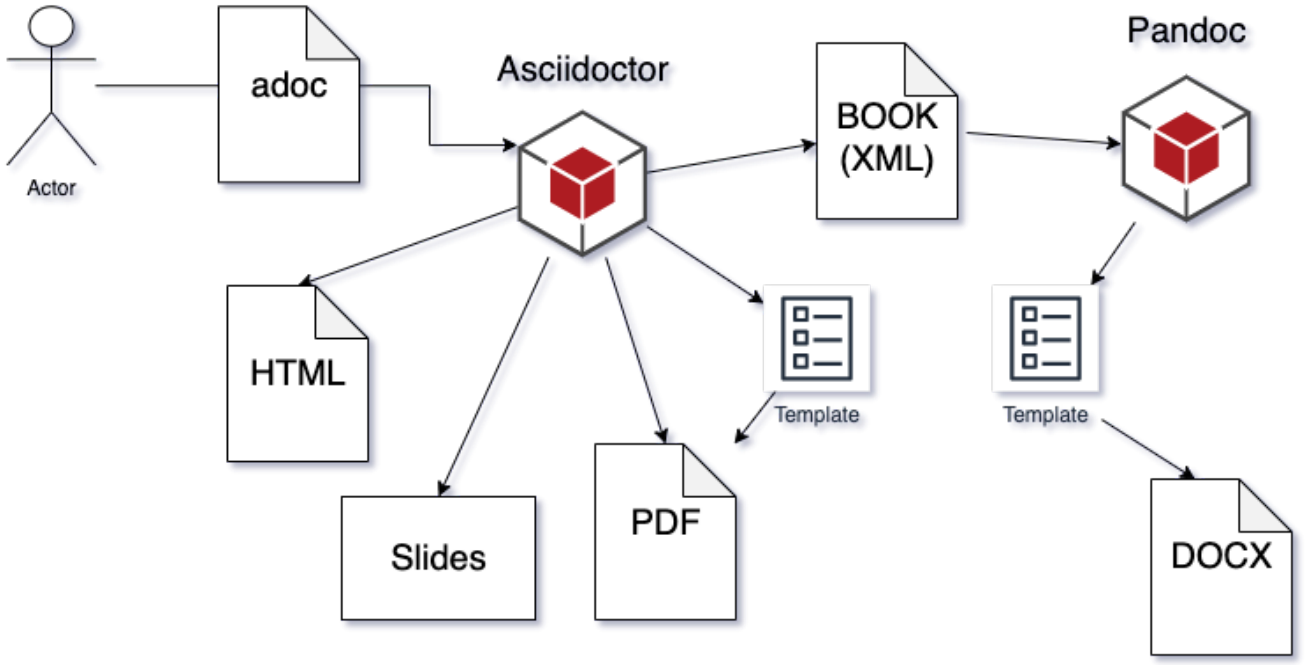
```
 8
 9 == Erstes Kapitel
10
11 Hier könnte hier Werbung stehen. Allerdings wird das
12 nicht billig! Zeilenumbrüche im Text werden nicht übernommen.
13
14 Zweiter Absatz - wird durch eine Leerzeile getrennt.
15 Danach folgt ein Bild.
16
17 .Schönes Bild
18 image::../images/nice-pic.jpg[width=50%,align="center"]
19
20 == Zweites Kapitel
21
22 Noch mehr nützliche Informationen, die mit Geld
23 nicht zu bezahlen sind.
```

HTML-Version dieses Asciidoc Beispiels

16.4. Ergebnis als Html Output



16.5. Funktionsübersicht



16.6. Headlines

```
== Level 1 / Kapitel
```

Level 1 / Kapitel

```
=== Level 2
```

Level 2

```
==== Level 3
```

Level 3

```
===== Level 4
```

Level 4

16.7. Paragraphs

```
1 Ein erster Absatz mit nicht ganz so viel Text.
2 *Auch hier könnte ihre Werbung stehen!*
3
4 Der zweite Absatz wird durch eine Leerzeile
```

```
5 getrennt und dadurch automatisch ein neue Absatz erzeugt.
```

Ein erster Absatz mit nicht ganz so viel Text. **Auch hier könnte ihre Werbung stehen!**

Der zweite Absatz wird durch eine Leerzeile getrennt und dadurch automatisch ein neue Absatz erzeugt.

16.8. Formatierung

```
*Fett*
```

Fett

```
_Kursiv_
```

Kursiv

```
+Monospace+
```

Monospace

```
Einen Zeilenumbruch +
erzwingen
```

Einen Zeilenumbruch
erzwingen

16.9. Images

```
1 image::images/nice-pic.jpg[]
2 // oder mit optionalen Attributen
3 image::images/nice-pic.jpg[width=50%, algin="center"]
```

16.10. Listen

```
1 // unordered list
2 * First
3 ** sub first
4 ** sub secound
5 *** Sub Sub
6 * Second
7 * Thirt
```

```
1 // ordered list
2 . First
3 .. sub first
4 .. sub secound
5 ... Sub Sub
6 . Second
7 . Thirt
```

16.11. Listen Ergebnis

Unordered list

- First
  - sub first
  - sub secound
    - Sub Sub
- Second
- Thirt

Ordered list

1. First

    a. sub first
    b. sub secound
        i. Sub Sub
2. Second
3. Thirt

16.12. Tabellen

A table with a title

```
|===
|Column 1, header row |Column 2, header row
|Cell in column 1, row 2
|Cell in column 2, row 2
|===
```

Column 1, header row	Column 2, header row
Cell in column 1, row 2	Cell in column 2, row 2

16.13. Tabellen - Best Practice

```
[%header,format=csv]
|===
Artist,Track,Genre
Baauer,Harlem Shake,Hip Hop
The Lumineers,Ho Hey,Folk Rock
|===
```

Artist	Track	Genre
Baauer	Harlem Shake	Hip Hop
The Lumineers	Ho Hey	Folk Rock

16.14. Sourcecode

```
[source, java, linenums]
----
class Simple{
    public static void main(String args[]){
     System.out.println("Hello Java");
    }
}
----
```

```
class Simple{
    public static void main(String args[]){
        System.out.println("Hello Java");
    }
}
```

16.15. Inhaltsverzeichnis

Im Dokumenten-Header :toc: angeben, damit wird automatisch ein Inhaltsverzeichnis erstellt.

```
= Dokument mit Inhaltsverzeichnis
Thomas Siwczak <thomas.siwczak@de.experis.com>

:toc:  // normal

:toc: left  // In HTML Ausgabe Inhaltsverzeichnis links

:toc: macro  // Damit kann das Verzeichnis frei platziert werden

// some Text

toc::[]  // Platzierung des Verzeichnis
```

16.16. Diagram Source

```
// Diagram MSD30
[ditaa, "diag-msd30", title="MSD30", caption="Abbildung {counter:abb1}: ",align="center", debug=false]
....
/------------+-------------------------------------------------------+ +-------------------------------\
|            |                                                 ZD    | |  dzK                          |
+------------/                                                       | |                               |
|      /-------------------------------\             /-------------\     | |                               |
|      |                               |             |  cDDE       |     | |                               |
|      |   /--------912--------\       \----1093-----+ ProvService |     | |                               |
|      |   |                   |                     |             |<-------------------------------\      |
|      |   |                   v                  /->|             |<--------------------\         |      |
|      v   v             /---------------\         |  |             |     | |             |         |      |
|  /-----------\         |  cFFE         |         |  \-------------/     | |             |         |      |
|  | cEFF      |         |               +---1088--/                      | |       /-----+---+-----\   |      |
|  |           |         |    SDK 2.0    |                                | |       | Router |cADE |   |      |
|  |   HPESM   |         |               |<---------------------865-------------+->|  cADE  |     |   |      |
|  |           |         | Ansible Tower |                                | |    |  +--------/     |   |      |
|  |           |         |               |<------------------905-----------------\ |  |              |   |      |
|  \-----------/         |               |                                | |  | \->|  KonSl       |   |      |
|                        |               |<---------------907-----------\ | |  |    |              |   |      |
|                        |               |                              | | |  |    \--------------/   |      |
|                        |               +-------------16----------\     | | |  |                       |      |
|                        |               |                         |    | | |  |    /--------------\   |      |
|                        \---------------/                         |    | | |  |    |    cADE      |   |      |
|                                                                  |    | | |  |    |  KonAu       |   |      |
|                                                                  |    | | |  \--->|              |   |      |
|                                                                  |    | | |       \--------------/   |      |
|                                                                  |    | | |                          |      |
|                                                                  |    | | |       /--------------\   |      |
|                                                                  |    | | |       |    cADE      |   |      |
|                                                                  |    | | |       |  KonMa       |   |      |
|                                                                  |    | | \------>|              |   |      |
|                                                                  |    | |         \--------------/   |      |
|                                                                  |    | |              /-------------/      |
|                                                                  |    | |              |                    |
|                                                                  |    | |         /----+---+-----\          |
|                                                                  |    | |         | Router |cADE |          |
|                                                                  |   \-------------->|  cADE  |     |          |
|                                                                  |    | |         +--------/     |          |
|                                                                  |    | |         |              |          |
|                                                                  |    | |         |  MFZ         |          |
|                                                                  |    | |         |              |          |
|                                                                  |    | |         \--------------/          |
|                                                                  |    | |                                   |
\------------------------------------------------------------------+ +-------------------------------/
....
```

16.17. Diagram - Output

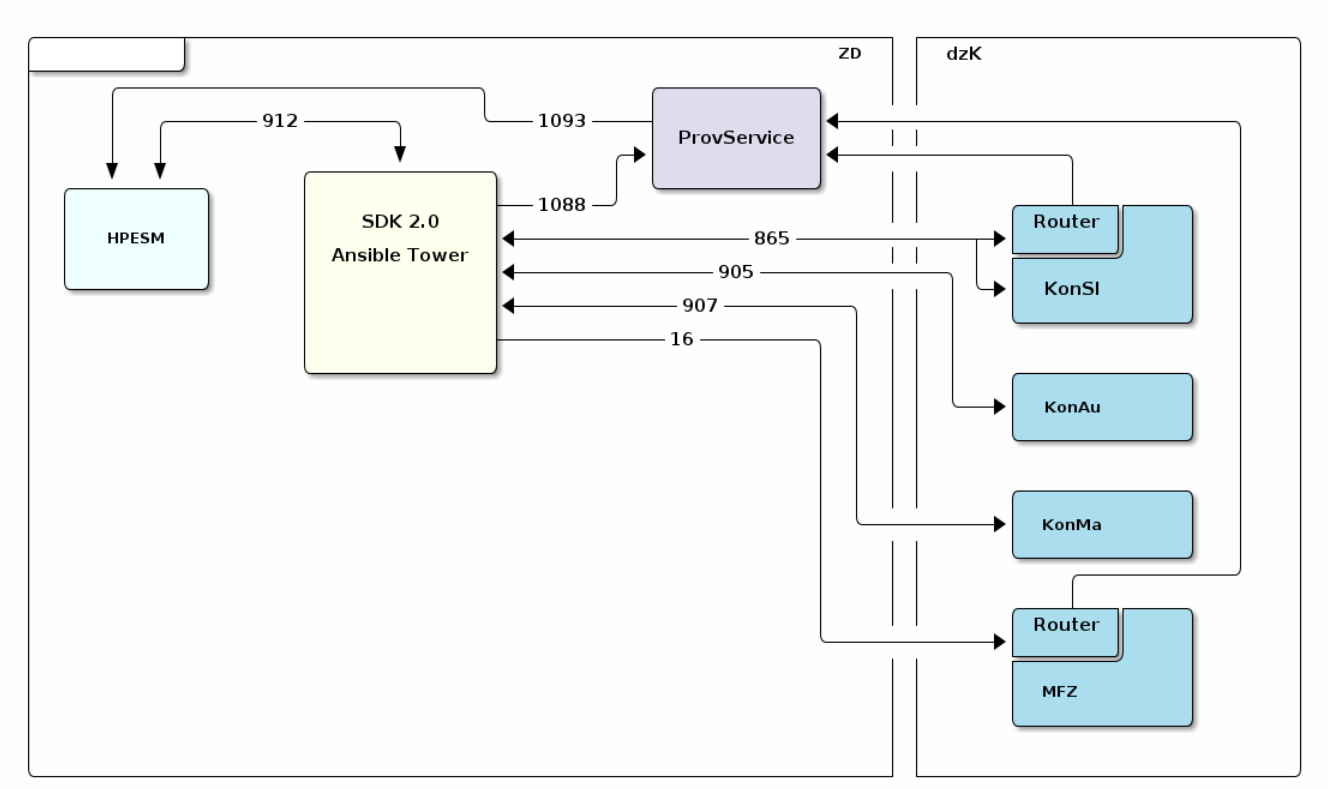


Abbildung 1: MSD30

16.18. Output Format

- html
- pdf
- xml / docbook
- reveal.js / Slides
- mit Pandoc weitere Formate:
  - docx
  - odt
  - uvm.

16.19. Best Practice

folgt in Kürze ...

- Captions
- includes
- Tabellen
- Diagrams

- Templates
- uvm.

16.20. Captions

16.21. Best Practice - Includes

Mittels includes lassen sich komplette Source Code Dateien oder Config Files extern in die Dokumentation einbinden (z.B. yaml files) Dadurch wird bei der Aktualisierung des Sourcecode automatisch auch immer die Dokumentation mit aktualisiert.

(adoc, Code, meta-data, uws.)

16.22. Best Practice - Diagrams

- ditaa
- plantuml
- draw.io ?

16.23. Best Practice - Tabellen mit csv

16.24. Best Practice - Templates

16.25. Pandoc

16.26. Dokumente generieren

Einfaches Beispiel

```
asciidoctor handbuch.adoc
```

Beispiel mit Diagram-Addon

```
asciidoctor handbuch.adoc -r asciidoctor-diagram
```

Generierung von PDF

```
asciidoctor-pdf handbuch.adoc
// oder
asciidoctor-pdf handbuch.adoc -r asciidoctor-diagram  -o output/handbuch.pdf
```

16.27. Dokumente generieren mit Docker

```
// asciidoctor
docker run --rm -v $(pwd):/documents/ asciidoctor/docker-asciidoctor asciidoctor
index.adoc

// asciidoctor-pdf
docker run --rm -v $(pwd):/documents/ asciidoctor/docker-asciidoctor asciidoctor-pdf
index.adoc

// asciidoctor mit diagram
docker run --rm -v $(pwd):/documents/ asciidoctor/docker-asciidoctor asciidoctor -r
asciidoctor-diagram index.adoc
```

Aktuelles Verzeichnis als Volume angeben:
`-v $(pwd):/documents/`

16.28. Dokumente generieren mit Podman

```
podman run --rm -v $(pwd):/documents/ docker.io/asciidoctor/docker-asciidoctor
asciidoctor-pdf index.adoc
```

Imagename = `docker.io/asciidoctor/docker-asciidoctor`

16.29. Links / Cheatsheet

- Asciidoctor Doku
- Quick Reference
- Cheatsheet
- Ascciidoc Diagram online editor https://asciiflow.com/
- Convert Markdown to Asciidoc

17. Hugo

[hugo logo wide] | */images/logos/hugo-logo-wide.svg*

17.1. Einführung

Hugo ist ein statischer Site-Generator, der in Go geschrieben wurde. Er ist bekannt für seine Geschwindigkeit und Flexibilität. Im Gegensatz zu dynamischen Web-Content-Management-Systemen, die Serverressourcen benötigen, generiert Hugo die gesamte Website in HTML, CSS und JavaScript vor dem Hochladen auf den Server.

17.2. Hauptmerkmale von Hugo

Die Hauptmerkmale von Hugo beinhalten:

- Schnelligkeit: Hugo ist bekannt als der schnellste Website-Generator auf dem Markt. Er kann Tausende von Seiten in Sekundenbruchteilen generieren.
- Go Templates: Hugo verwendet Go's eingebaute Template-Bibliothek für die Erstellung von Website-Templates.
- Markdown Unterstützung: Hugo unterstützt Markdown für Inhalte, was es einfach macht, Inhalte zu erstellen und zu formatieren.
- Anpassungsfähigkeit: Hugo kann einfach angepasst werden, um eine Vielzahl von Website-Typen zu erstellen, einschließlich Blogs, Dokumentation, Portfolio-Sites und mehr.

17.3. Hugo's Architektur

Hugo verwendet eine einfache Verzeichnisstruktur, die es dem Benutzer ermöglicht, die Struktur und das Design seiner Website intuitiv zu verstehen. Die Hauptkomponenten sind:

- Content-Verzeichnis: Hier speichert der Benutzer seine Inhaltsdateien. Jede Datei wird zu einer Seite auf der Website.
- Layout-Verzeichnis: Hier werden die HTML-Templates gespeichert, die definieren, wie die Website aussieht.
- Static-Verzeichnis: Hier werden alle statischen Ressourcen wie Bilder, CSS- und JavaScript-Dateien gespeichert.

17.4. Fazit

Hugo ist ein leistungsfähiges Tool für die Erstellung von Websites. Seine Geschwindigkeit, Flexibilität und einfache Anpassung machen es zu einer ausgezeichneten Wahl für Entwickler aller Erfahrungsstufen.

17.5. Links

- Hugo
- Docker for Hugo
- https://medium.com/@wabimantoro/create-and-deploy-website-for-free-with-hugo-8765485b0c39
- Usage with asciidoc

18. Git

18.1. Was ist Git?

Git ist eine Sammlung von Dienstprogrammen in der Kommandozeile, die Änderungen in Dateien verfolgen und aufzeichnen (meistens Quellcode, aber du kannst alle möglichen Dateien wie Textdateien und sogar Bild-Dateien "tracken".

Durch diese Funktionalität kannst du alte Versionen deines Projekts wiederherstellen, miteinander vergleichen, analysieren, Änderungen zusammenführen (mergen) und vieles mehr.

18.2. Die Vorteile der Versionsverwaltung mit Git

- verteiltes System zur Codeverwaltung
- Snapshots des aktuellen Zustands eines Codes
- Effiziente und intelligente Zusammenarbeit im Team
- Es erzeugt Zweige, die mehrere Arbeitsströme von verschiedenen Entwicklern, unabhängig voneinander, festhalten. Diese Zweige können zu einer einzigen Code-Datei zusammengeführt werden.
- Änderungen am Code sind nachvollziehbar, wer ihn geändert hat, wann er geändert wurde und welche Versionen es vorher gab.
- Es ist betriebssystem- und sprachunabhängig. Jeder Entwickler kann von jedem System und mit jeder Sprache an Git arbeiten.
- Git ist natürlich nicht die einzige Versionsverwaltung. Andere Versionsverwaltungssysteme sind CVS, Bazaar und SNV.

18.3. Funktionsweise

18.4. Repo Arten

Normales Git Repo

Repository (History) und Arbeitsverzeichnis für Änderungen

Bare / mirror

nur Repository und kein Arbeitsverzeichnis

zur zentralen Ablage / pull & push möglich

18.5. Häufig genutzte Commands

Es folgt eine Auswahl der häufig genutzten Befehle

init, clone, config, status, log, add, commit, fetch, pull, push, stash, branch, remote

18.6. git init / git clone

Dieser Befehl (**git init**) initialisiert ein neues lokales Git Repository. Der Repo-Name wird direkt nach dem Befehl hinzugefügt.

```
git init <myrepo>
```

Mit diesem Befehl (**git clone**) können wir den Quellcode aus einem entfernten Repository auf einen lokalen Rechner herunterladen. Er erstellt eine Kopie dieses Repos auf dem lokalen Rechner.

```
git clone <URL>
```

Es ist auch möglich lokale Repos aus dem Dateisystem zu clonen, muss nicht zwingend von einem Server erfolgen.

18.7. git config (email, name)

Dieser Befehl erlaubt es dir, git mitzuteilen, wer du bist. Du kannst deinen Namen und deine E-Mail hinzufügen.

```
git config --global user.name [firstname lastname]
```

set a name that is identifiable for credit when review version history

```
git config --global user.email [valid-email]
```

set an email address that will be associated with each history marker

```
git config --global color.ui auto
```

set automatic command line coloring for Git for easy reviewing

18.8. git status

Dieser Befehl zeigt den Status eines Branches an. Die Verwendung dieses Befehls sagt uns, ob es ungetrackte, staged oder unstaged Dateien gibt. Es lässt uns wissen, ob es Dateien zum Committen, Pushen oder Pullen gibt und ob ein Branch aktuell ist.

```
git status
```

```
% git status
Auf Branch dev
Ihr Branch ist auf demselben Stand wie 'origin/dev'.

Änderungen, die nicht zum Commit vorgemerkt sind:
  (benutzen Sie "git add <Datei>...", um die Änderungen zum Commit vorzumerken)
  (benutzen Sie "git restore <Datei>...", um die Änderungen im Arbeitsverzeichnis zu verwerfen)
        geändert:       doku/incl/git.adoc

keine Änderungen zum Commit vorgemerkt (benutzen Sie "git add" und/oder "git commit -a")
```

Figure 1. Beispiel - git status

18.9. git diff

diff of what is changed but not staged

```
git diff
```

diff of what is staged but not yet committed

```
git diff --staged
```

```
% git diff
diff --git a/doku/incl/git.adoc b/doku/incl/git.adoc
index 6857a29..71ea2d2 100644
--- a/doku/incl/git.adoc
+++ b/doku/incl/git.adoc
@@ -83,6 +83,13 @@ image::images/git-status.png[align="center"]

 == git diff

+diff of what is changed but not staged
+
+ git diff
+
+.Beispiel git diff
+image::images/git-diff.png[]
+
 == git log

 == git add
```

Figure 2. Beispiel - git diff

18.10. git log

Listet all Commits für den aktuellen Branch auf.

```
git log
```

um die Anzahl beschränken kann man einen Parameter angeben, z.B. für 3 "-3" und für einen einzeilige Anzeige wird der Parameter "--pretty=oneline" benutzt.

```
git log -10 --pretty=oneline
```

Um einen bestimmt Datei zu verfolgen:

```
git log --follow [file]
```

18.11. git add

Hiermit wird eine Datei bereitgestellt, damit sie an das Repository übergeben werden kann. Sie können auf drei Arten bereitstellen.

- **git add *** fügt alle Dateien, Ordner und Unterordner in einem Verzeichnis hinzu, mit Ausnahme von Dateien, die mit einem Punkt wie .gitignore beginnen.
- **git add <filename>** fügt nur die Datei hinzu, die mit dem Dateinamen angegeben wurde.
- **git add .** fügt alle Dateien, Ordner und Unterordner in einem Verzeichnis hinzu, einschließlich der Dateien, die mit einem Punkt wie .gitignore beginnen.

```
git add <filename>
```

```
// for example
git add hallo.txt
```

18.12. git commit

Dieser Befehl speichert Snapshots der Arbeitsversion eines Projekts. Er tut dies, indem er alle Dateien in das Repository überträgt. Du kannst commit nur verwenden, nachdem du die Dateien mit git add ins Repository gestellt hast.

Commits werden normalerweise mit einer Commit-Nachricht hinzugefügt.

```
git commit -m "[descriptive message]"
```

Alternativ kann auch der Parameter -m weg gelassen werden, dann öffnet sich im Anschluss der Standart Editor, wo dann die Commit Message bearbeitet / eingegeben werden muss.

Die Commit-Message sollte einer gewissen Struktur entsprechen, mehr dazu im Abschnitt: Commit Messages

18.13. git show (*new*)

Zeigt die letzt commit meaasge an oder eine bestimmte, wenn man einen commit hash angibt

```
git show -s
```

```
git show -s b907a23e9cdf08f04c009863140c3460bb0ff748
```

```
% git show b907a23e9cdf08f04c009863140c3460bb0ff748 -s
commit b907a23e9cdf08f04c009863140c3460bb0ff748
Author: Martin Fischer <martin.fischer@de.experis.com>
Date:   Thu Jul 7 17:32:02 2022 +0200

    ADD: Added interactive git demo URL
```

Figure 3. Beispiel - git show

18.14. git fetch / git pull

fetch - fetch down all the branches from that Git remote

```
git fetch
```

git pull holt und lädt Inhalte von einem entfernten Repo herunter und aktualisiert das lokale Repo mit den heruntergeladenen Inhalten.

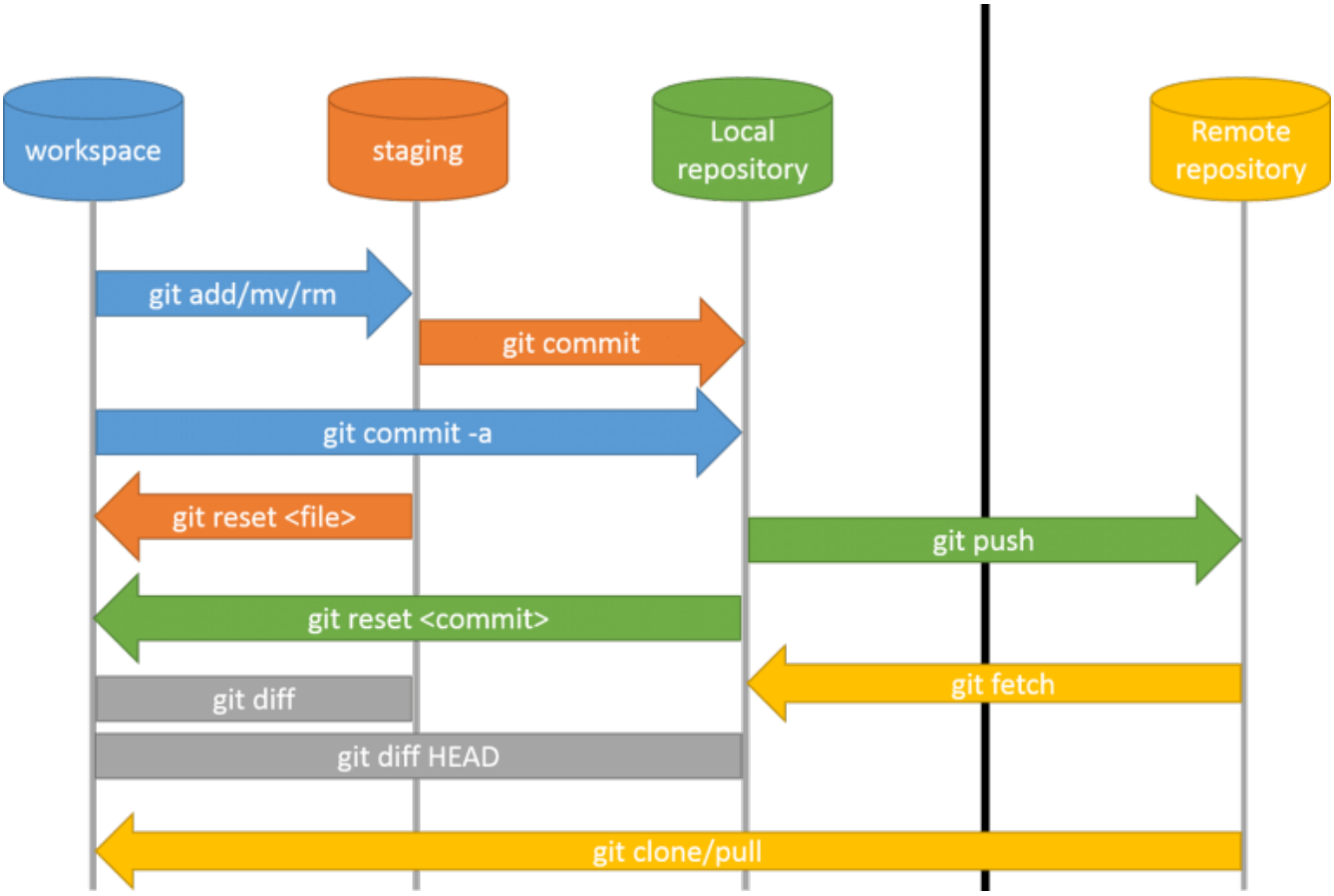
```
git pull
```

18.15. git push

Dieser Befehl pusht übertragene Änderungen aus einem lokalen Branch in ein anderes Repository.

```
git push [alias] [branch]
// for example
git push origin dev
```

18.16. git - stages



18.17. git stash (list, show, drop, pop / apply)

Stash local changes

```
git stash
```

List stashed changes

```
git stash list
```

Show stashed changes

```
git stash show
```

Remove stashed changes

```
git stash drop [<stash>]
```

Create branch from stashed changes and delete(!) stash

```
git stash branch <branchname> [stash>]
```

Remove single stashed state from stash list and apply it on current working tree

```
git stash pop [<stash>]
```

Apply stash to current working tree

```
git stash apply [<apply>]
```

Remove all stash entries

```
git stash clear
```

Saving temp work without stash

```
// hack hack hack
git switch -c my_wip
git commit -a -m "WIP"
git switch master
edit emergency fix
git commit -a -m "Fix in a hurry"
git switch my_wip
git reset --soft HEAD^
// continue hacking
```

Saving work with stashing

```
// hack hack hack
git stash
edit emergency fix
git commit -a -m "Fix in a hurry"
git stash pop
// continue hacking
```

18.18. branches

Show local branches

```
git branch
```

Show all branches

```
git branch -a
```

Create new branch

```
git branch <my_branch>
git checkout <my_branch>
```

Alternativer Shortcut

```
git checkout -b <my_branch>
```

Unterschiede zwischen Branches ermitteln / anzeiegn

```
git diff branchB...branchA
```

18.19. git remote

Origin - default remote

show details of remote settings

```
git remote show origin
```

get the remote repo-url

```
git remote get-url origin
```

set a new url for remote repo

```
git remote set-url origin
```

Weitere Remotes sind mittels

```
git remote add <name> <url>
```

definierbar.

18.20. Git stages

18.21. Commit Messages

Commit Messages sollten mit einer gewissen Stuktur erstellt werden: ADD:, CHG:, DEL: der Message voranstellen, Kurzbeschreibung in einer Zeile. Ausführliche Beschreibung der Änderung (Was, in welche[r|n] Datei[en], ggfs. Change-Nr, in weiteren Zeilen

18.22. .gitignore file

18.23. Best Practice

Does	Don't does
Name & Email setzen	git push --force
gute commit messages	bad code pushen
Branches nutzen	in master / main pushen
max 1 Funktion in commit	viele Changes in 1 Commit packen

18.24. Git-Submodule

Ein Git-Submodul ermöglicht es Ihnen, ein Git-Repository als Unterverzeichnis in einem anderen Git-Repository zu haben. Dies ist nützlich, wenn Sie Code wiederverwenden möchten, der in einem anderen Repository gepflegt wird. Hier ist ein einfacher Leitfaden, um ein Git-Submodul hinzuzufügen und es zu nutzen:

18.24.1. Git-Submodul hinzufügen

Um ein Submodul hinzuzufügen, verwenden Sie den `git submodule add` Befehl, gefolgt von der URL des Repositories, das Sie als Submodul hinzufügen möchten. Zum Beispiel:

```
1 git submodule add https://github.com/beispiel/repo.git
```

Dieser Befehl erzeugt ein neues Unterverzeichnis in Ihrem aktuellen Repository, klont das andere Repository hinein und setzt es auf den aktuellen Commit fest.

18.24.2. Git-Submodul aktualisieren

Nachdem das Submodul hinzugefügt wurde, ist es auf den Commit festgesetzt, der zu der Zeit aktuell war. Wenn Sie das Submodul auf den neuesten Stand bringen möchten, müssen Sie das Unterverzeichnis des Submoduls wechseln und `git pull` ausführen.

```
1 cd repo
2 git pull origin main
```

18.24.3. Git-Submodul entfernen

Das Entfernen eines Submoduls erfordert ein paar Schritte mehr:

- Entfernen Sie das Submodul aus der `.gitmodules` Datei.
- Entfernen Sie das Submodul aus der `.git/config` Datei.
- Führen Sie `git rm --cached path_to_submodule` aus (keinen abschließenden Schrägstrich).
- Führen Sie `rm -rf .git/modules/path_to_submodule` aus.
- Commit und löschen Sie das nun unverfolgte Submodul-Verzeichnis.

```
1 git rm --cached repo
2 rm -rf .git/modules/repo
3 rm -rf repo
```

Bitte ersetzen Sie `repo` durch den Namen Ihres Submoduls.

18.25. Nützliche Commandos / Befehle

Ermittelt die höchste Version / höchsten git tag

```
1 // tags aus remote repos abrufen
2 git fetch --tags
3
4 // tags auflisten | sortieren | letzten anzeigen
5 git tag | sort -V | tail -n 1
```

18.26. Links / Cheatsheet

Git - Installation & Dokumentation

Git Book

https://education.github.com/git-cheat-sheet-education.pdf

https://www.atlassian.com/git/tutorials/why-git

Interactive git branch demo

19. Semantic Versioning

19.1. Zusammenfassung

Auf Grundlage einer Versionsnummer von MAJOR.MINOR.PATCH werden die einzelnen Elemente folgendermaßen erhöht:

1. **MAJOR** wird erhöht, wenn API-inkompatible Änderungen veröffentlicht werden,
2. **MINOR** wird erhöht, wenn neue Funktionalitäten, die kompatibel zur bisherigen API sind, veröffentlicht werden, und
3. **PATCH** wird erhöht, wenn die Änderungen ausschließlich API-kompatible Bugfixes umfassen.

Außerdem sind Bezeichner für Vorveröffentlichungen und Build-Metadaten als Erweiterungen zum MAJOR.MINOR.PATCH-Format verfügbar.

19.2. Links / weitere Infos

- Semantic Versioning 2.0.0
- Conventional Commits
- Commitizen Tool

20. RKE2 - Rancher

RKE2 steht für "Rancher Kubernetes Engine 2", und es handelt sich dabei um eine Kubernetes-Distribution, die von Rancher Labs entwickelt wurde. Kubernetes ist ein Open-Source-Container-Orchestrierungs-Framework, das dazu dient, Container-Anwendungen in skalierbaren, hochverfügbaren Clustern zu verwalten. RKE2 ist eine spezielle Implementierung von Kubernetes, die einige besondere Merkmale und Vorteile bietet:

1. **Einfache Bereitstellung**: RKE2 wurde entwickelt, um die Bereitstellung von Kubernetes-Clustern zu vereinfachen. Es bietet ein benutzerfreundliches Installations- und Konfigurationsverfahren, das auch für Einsteiger zugänglich ist.
2. **Sicherheit**: RKE2 setzt auf Sicherheit und konzentriert sich auf die Minimierung von Angriffsflächen. Es verwendet standardmäßig den Containerd-Container-Manager anstelle von Docker und bietet Funktionen wie SELinux und AppArmor zur weiteren Verbesserung der Sicherheit.
3. **High Availability**: RKE2 unterstützt die Einrichtung von hochverfügbaren Kubernetes-Clustern. Dies bedeutet, dass Ihr Cluster weiterhin funktionieren kann, selbst wenn einzelne Komponenten oder Knoten ausfallen.
4. **Automatisierung**: RKE2 enthält Funktionen zur Automatisierung von Aufgaben wie Updates und Upgrades, was die Wartung und Verwaltung Ihres Kubernetes-Clusters erleichtert.
5. **Kubernetes-Kompatibilität**: RKE2 bleibt eng mit der Kubernetes-Community und dem Kubernetes-Ökosystem verbunden und ist daher mit vielen Kubernetes-Tools und -Ressourcen kompatibel.
6. **Modularität**: RKE2 verwendet eine modulare Architektur, die es Ihnen ermöglicht, Komponenten und Erweiterungen nach Bedarf hinzuzufügen oder zu entfernen.
7. **Unterstützung für verschiedene Plattformen**: RKE2 kann auf verschiedenen Betriebssystemen und Infrastrukturplattformen, einschließlich Bare-Metal-Servern, virtuellen Maschinen und Cloud-Diensten, ausgeführt werden.

Die Vorteile von RKE2 machen es zu einer attraktiven Option für Unternehmen und Entwickler, die Kubernetes in ihren Anwendungen und Diensten verwenden möchten, da es die Einrichtung, Verwaltung und Sicherheit von Kubernetes-Clustern erleichtert. Beachten Sie jedoch, dass sich die Technologie und Features von RKE2 im Laufe der Zeit weiterentwickeln können, sodass es ratsam ist, die neuesten Informationen und Dokumentationen zu überprüfen, um die aktuellen Funktionen und Best Practices zu verstehen.

20.1. Install

20.1.1. ARM64

```
1 mkdir /root/rke2-artifacts && cd /root/rke2-artifacts
2 wget http://bit.ly/3GQOxhd rke2-images.linux-arm64.tar.gz
3 wget https://github.com/rancher/rke2/releases/download/v1.27.3+rke2r1/rke2.linux-
  arm64.tar.gz
```

```
wget https://github.com/rancher/rke2/releases/download/v1.27.3+rke2r1/sha256sum-arm64.txt
curl -sfL https://get.rke2.io --output install.sh

INSTALL_RKE2_TYPE=agent INSTALL_RKE2_ARTIFACT_PATH=/root/rke2-artifacts sh install.sh

systemctl enable rke2-server
systemctl start rke2-server
```

Kubectl

```
curl -LO https://dl.k8s.io/release/v1.28.4/bin/linux/arm64/kubectl
install kubectl /usr/local/bin
rm kubectl -f
```

k9s

```
curl -LO https://github.com/derailed/k9s/releases/download/v0.29.1/k9s_Linux_arm64.tar.gz

tar xvzf k9s_Linux_arm64.tar.gz k9s ①

install k9s /usr/local/bin

rm -f k9s k9s_Linux_arm64.tar.gz ②
```

① Unpack only k9s binary from archive

② Cleanup

20.2. Create Aliase

```
alias kcn='kubectl config set-context --current --namespace'
alias k='kubectl'
alias kpo='kubectl get po -A'
alias kepo='kubectl get po -A | grep -Ev "Running|Completed"'
```

20.3. Links

- Offizelle Doku

21. RKE2 - Rancher

RKE2 steht für "Rancher Kubernetes Engine 2", und es handelt sich dabei um eine Kubernetes-Distribution, die von Rancher Labs entwickelt wurde. Kubernetes ist ein Open-Source-Container-Orchestrierungs-Framework, das dazu dient, Container-Anwendungen in skalierbaren, hochverfügbaren Clustern zu verwalten. RKE2 ist eine spezielle Implementierung von Kubernetes, die einige besondere Merkmale und Vorteile bietet:

1. **Einfache Bereitstellung**: RKE2 wurde entwickelt, um die Bereitstellung von Kubernetes-Clustern zu vereinfachen. Es bietet ein benutzerfreundliches Installations- und Konfigurationsverfahren, das auch für Einsteiger zugänglich ist.
2. **Sicherheit**: RKE2 setzt auf Sicherheit und konzentriert sich auf die Minimierung von Angriffsflächen. Es verwendet standardmäßig den Containerd-Container-Manager anstelle von Docker und bietet Funktionen wie SELinux und AppArmor zur weiteren Verbesserung der Sicherheit.
3. **High Availability**: RKE2 unterstützt die Einrichtung von hochverfügbaren Kubernetes-Clustern. Dies bedeutet, dass Ihr Cluster weiterhin funktionieren kann, selbst wenn einzelne Komponenten oder Knoten ausfallen.
4. **Automatisierung**: RKE2 enthält Funktionen zur Automatisierung von Aufgaben wie Updates und Upgrades, was die Wartung und Verwaltung Ihres Kubernetes-Clusters erleichtert.
5. **Kubernetes-Kompatibilität**: RKE2 bleibt eng mit der Kubernetes-Community und dem Kubernetes-Ökosystem verbunden und ist daher mit vielen Kubernetes-Tools und -Ressourcen kompatibel.
6. **Modularität**: RKE2 verwendet eine modulare Architektur, die es Ihnen ermöglicht, Komponenten und Erweiterungen nach Bedarf hinzuzufügen oder zu entfernen.
7. **Unterstützung für verschiedene Plattformen**: RKE2 kann auf verschiedenen Betriebssystemen und Infrastrukturplattformen, einschließlich Bare-Metal-Servern, virtuellen Maschinen und Cloud-Diensten, ausgeführt werden.

Die Vorteile von RKE2 machen es zu einer attraktiven Option für Unternehmen und Entwickler, die Kubernetes in ihren Anwendungen und Diensten verwenden möchten, da es die Einrichtung, Verwaltung und Sicherheit von Kubernetes-Clustern erleichtert. Beachten Sie jedoch, dass sich die Technologie und Features von RKE2 im Laufe der Zeit weiterentwickeln können, sodass es ratsam ist, die neuesten Informationen und Dokumentationen zu überprüfen, um die aktuellen Funktionen und Best Practices zu verstehen.

21.1. Install

21.1.1. ARM64

```
mkdir /root/rke2-artifacts && cd /root/rke2-artifacts/
wget https://github.com/rancher/rke2/releases/download/v1.27.3%2Brke2r1/rke2-
images.linux-arm64.tar.gz
wget https://github.com/rancher/rke2/releases/download/v1.27.3%2Brke2r1/rke2.linux-
```

```
arm64.tar.gz
wget https://github.com/rancher/rke2/releases/download/v1.27.3%2Brke2r1/sha256sum-
arm64.txt
curl -sfL https://get.rke2.io --output install.sh
INSTALL_RKE2_TYPE=agent INSTALL_RKE2_ARTIFACT_PATH=/root/rke2-artifacts sh install.sh

systemctl enable rke2-server
systemctl start rke2-server
```

Kubectl

```
curl -LO https://dl.k8s.io/release/v1.28.4/bin/linux/arm64/kubectl
install kubectl /usr/local/bin
rm kubectl -f
```

k9s

```
curl -LO
https://github.com/derailed/k9s/releases/download/v0.29.1/k9s_Linux_arm64.tar.gz
tar xvzf k9s_Linux_arm64.tar.gz
install k9s /usr/local/bin
rm -f k9s k9s_Linux_arm64.tar.gz
```

21.2. Create Aliase

```
alias kcn='kubectl config set-context --current --namespace '
alias k='kubectl'
alias kpo='kubectl get po -A'
alias kepo='kubectl get po -A | grep -v "Running|Complited"'
```

21.3. Links

- Offizelle Doku

22. HAProxy

Vor- und Nachteile, Installation & Konfiguration

HAProxy ist ein beliebter Load Balancer und Proxy-Server, der sich besonders durch seine Leistung und Zuverlässigkeit auszeichnet. Im Vergleich zu anderen Load Balancern bietet HAProxy sowohl Vorteile als auch Nachteile. Hier ist ein Überblick:

22.1. Vorteile von HAProxy

1. **Hohe Leistung und Zuverlässigkeit**: HAProxy ist bekannt für seine hohe Durchsatzkapazität und geringe Latenz, was es ideal für hochverfügbare Umgebungen macht.
2. **Flexibilität in der Konfiguration**: HAProxy bietet eine sehr detaillierte und flexible Konfiguration, die es ermöglicht, Verkehr sehr präzise zu steuern und zu manipulieren.
3. **Unterstützung für HTTP und TCP**: Es kann sowohl als HTTP-Load-Balancer als auch als TCP/UDP-Load-Balancer verwendet werden, was es vielseitig einsetzbar macht.
4. **Gesundheitsprüfungen und Failover**: HAProxy bietet fortschrittliche Gesundheitsprüfungen und Failover-Mechanismen, um die Verfügbarkeit der Dienste zu gewährleisten.
5. **Open Source und Gemeinschaftsunterstützung**: Als Open-Source-Tool hat es eine starke Community, die ständig zur Weiterentwicklung des Tools beiträgt.
6. **SSL/TLS-Unterstützung**: Es unterstützt SSL/TLS-Terminierung, was die Sicherheit verbessert.

22.2. Nachteile von HAProxy

1. **Komplexität in der Konfiguration**: Die detaillierte Konfiguration kann für neue Benutzer überwältigend sein und erfordert ein gewisses Maß an technischem Verständnis.
2. **Fehlende GUI**: Im Gegensatz zu einigen anderen Lösungen bietet HAProxy keine grafische Benutzeroberfläche, was die Konfiguration und das Management erschwert.
3. **Eingeschränkter Support für Websockets**: Obwohl HAProxy Websockets unterstützt, kann es im Vergleich zu spezialisierten Lösungen Limitierungen geben.
4. **Keine native Cloud-Integration**: Im Gegensatz zu Cloud-nativen Lösungen wie AWS Elastic Load Balancing oder Azure Load Balancer bietet HAProxy keine direkte Integration mit Cloud-Diensten.

22.3. Vergleich mit Anderen Load Balancern

- **Nginx**: Nginx ist ebenfalls ein sehr beliebter Load Balancer und Webserver. Im Vergleich zu HAProxy bietet Nginx eine einfachere Konfiguration und eine bessere Integration in Webserver-Funktionalitäten, ist aber in einigen High-Performance-Szenarien möglicherweise nicht so leistungsfähig wie HAProxy.
- **AWS Elastic Load Balancing (ELB)**: ELB ist eine Cloud-native Lösung, die eine nahtlose Integration in AWS-Dienste bietet. Während ELB eine einfache Konfiguration und automatische Skalierung bietet, fehlt ihm die Flexibilität und detaillierte Konfigurierbarkeit von HAProxy.

- **F5 Big-IP**: Big-IP ist eine kommerzielle Lösung, die neben Load Balancing auch Funktionen für Anwendungssicherheit und Performance-Management bietet. Im Vergleich zu HAProxy bietet Big-IP mehr Enterprise-Funktionen, ist aber auch kostenintensiver.

Jeder Load Balancer hat seine Stärken und Schwächen, und die Wahl hängt von den spezifischen Anforderungen Ihrer Infrastruktur, Ihrem Budget und Ihren technischen Fähigkeiten ab. HAProxy ist eine ausgezeichnete Wahl für Szenarien, in denen hohe Leistung, Zuverlässigkeit und detaillierte Verkehrskontrolle erforderlich sind.

22.4. Installation

1. **Systemaktualisierung**:
   - Debian-basierte Systeme:
     `sudo apt-get update`
2. **HAProxy installieren**:
   - Debian-basierte Systeme:
     `sudo apt-get install haproxy`
   - Für andere Linux-Distributionen wie CentOS den entsprechenden Paketmanager verwenden.

22.5. Grundlegende Konfiguration

1. **Konfigurationsdatei bearbeiten**:
   - Die Standardkonfigurationsdatei befindet sich unter `/etc/haproxy/haproxy.cfg`.
   - Öffnen Sie die Datei mit einem Texteditor, z.B. `sudo nano /etc/haproxy/haproxy.cfg`.
2. **Einfachen Load-Balancer konfigurieren**:
   - Fügen Sie Abschnitte für `defaults`, `frontend`, und `backend` hinzu.
   - Im `frontend`-Abschnitt definieren Sie den Port und leiten den Verkehr an `backend` weiter.
   - Im `backend`-Abschnitt definieren Sie die Server für den Lastausgleich.

22.6. Beispielkonfiguration

```
defaults
    mode http
    timeout connect 5000ms
    timeout client 50000ms
    timeout server 50000ms

frontend http_front
    bind *:80
    default_backend http_back

backend http_back
```

```
    balance roundrobin
    server server1 192.168.0.1:80 check
    server server2 192.168.0.2:80 check
```

22.7. Nach der Konfiguration

1. **Konfiguration überprüfen**:
   - `sudo haproxy -c -V -f /etc/haproxy/haproxy.cfg`
2. **HAProxy neu starten**:
   - `sudo systemctl restart haproxy`
3. **Status überprüfen**:
   - `sudo systemctl status haproxy`

22.8. Zusätzliche Schritte

- **Sicherheitsaspekte**: Konfigurieren Sie Ihre Firewall entsprechend.
- **Erweiterte Konfiguration**: Für fortgeschrittene Einstellungen konsultieren Sie die HAProxy-Dokumentation.

Wichtig: Diese Anleitung ist grundlegend. Für spezifische Anforderungen und Architekturen sollten Sie sich weitergehend informieren und die offizielle Dokumentation konsultieren.

23. Consul

Consul ist ein Dienstnetzwerk-Tool, das von HashiCorp entwickelt wurde, dem Unternehmen hinter Vagrant und Packer. Es bietet eine vollständige Plattform für die Entdeckung von Diensten, die Konfigurationsverwaltung und die Segmentierung in verteilten Anwendungen und Diensten.

23.1. Kernfunktionen von Consul

Consul bietet viele Funktionen, die dazu beitragen, die Herausforderungen des Betriebs von verteilten Systemen zu bewältigen. Dazu gehören:

- **Dienstentdeckung:** Anwendungen können Consul verwenden, um andere Dienste im Netzwerk mit einem DNS- oder HTTP-Interface zu entdecken.
- **Gesundheitsüberprüfungen:** Consul kann die Verfügbarkeit von Diensten überwachen und Anwendungen die Anforderungen an gesunde oder ungesunde Instanzen weiterleiten.
- **Key/Value-Speicher:** Ein flexibler Key/Value-Speicher ermöglicht die dynamische Konfiguration, das Feature-Flagging, die Koordination, die Führungswahl und vieles mehr.
- **Sichere Dienstekommunikation:** Automatische TLS-Verschlüsselung und Identitätsbasierte Autorisierung für Dienste.

23.2. Wie man Consul einsetzt

Consul ist sehr flexibel und kann in einer Vielzahl von Umgebungen und Anwendungsfällen eingesetzt werden. Hier sind einige gängige Einsatzmöglichkeiten:

- **Mikroservicenetze:** Consul kann als zentrales Dienstverzeichnis für ein Netzwerk von Mikroservices verwendet werden, um die Dienstentdeckung und -segmentierung zu vereinfachen.
- **Multicloud- und Plattformübergreifende Bereitstellungen:** Mit Consul können Sie Dienste über verschiedene Cloud-Plattformen und -Bereitstellungsumgebungen hinweg koordinieren.
- **Konfigurationsverwaltung:** Sie können Consul verwenden, um Konfigurationsdaten für Ihre Anwendungen zu speichern und abzurufen.

Um Consul zu installieren und zu verwenden, laden Sie es von der offiziellen HashiCorp-Website herunter und folgen Sie den Anleitungen in der Dokumentation. Ein typisches Consul-Setup könnte mehrere Consul-Server enthalten, die in verschiedenen Teilen Ihres Netzwerks laufen, um Dienste zu entdecken und zu überwachen.

24. Tmux

Terminal-Multiplexer-Tool

Tmux, kurz für 'Terminal Multiplexer', ist ein wertvolles Werkzeug für jeden, der viel Zeit in der Befehlszeile verbringt. Es ermöglicht den Benutzern, mehrere Terminal-Sitzungen in einem einzigen Fenster zu verwalten, und bietet eine Reihe von Funktionen, die die Produktivität erhöhen und den Workflow optimieren.

24.1. Hauptmerkmale von Tmux

- **Sitzungsmanagement**: Tmux ermöglicht es Benutzern, Sitzungen zu erstellen, zu trennen, anzuhängen und zwischen ihnen zu wechseln, was es ideal für das gleichzeitige Arbeiten an mehreren Aufgaben oder Projekten macht.
- **Fenster und Bereiche**: Innerhalb einer Tmux-Sitzung können Benutzer mehrere Fenster öffnen und jedes Fenster in mehrere Bereiche unterteilen. Dies erleichtert die Navigation und das Multitasking.
- **Anpassbarkeit**: Tmux ist hochgradig anpassbar und erlaubt den Benutzern, Schlüsselbindungen zu ändern und das Aussehen des Interfaces anzupassen.

Mit Tmux kannst du deinen Befehlszeilen-Workflow erheblich verbessern, ob du nun ein Entwickler bist, der mehrere Codebasen verwaltet, oder ein Systemadministrator, der verschiedene Server überwacht.

24.2. Installation

Abhängig von deinem Betriebssystem, kann Tmux wie folgt installiert werden:

Ubuntu und andere Linux-Distributionen: `sudo apt-get install tmux`

Mac OS X: `brew install tmux`

Windows: Unter Windows empfehle ich die Verwendung von WSL (Windows Subsystem for Linux) und dann den gleichen Befehl wie bei Ubuntu.

24.3. Erste Schritte

Nach der Installation kannst du eine neue Tmux-Sitzung starten, indem du `tmux` in die Kommandozeile eingibst. Du solltest nun eine neue Tmux-Sitzung sehen mit einer Statusleiste am unteren Rand.

24.4. Sitzungen, Fenster und Bereiche

In Tmux gibt es Konzepte wie Sitzungen, Fenster und Bereiche.

Sitzung: Eine Sitzung ist eine unabhängige Arbeitsumgebung mit einer eigenen Gruppe von

Fenstern.

Fenster: Ein Fenster nimmt den gesamten Bildschirm ein und kann mehrere Bereiche enthalten.

Bereiche: Ein Fenster kann in mehrere Bereiche unterteilt werden.

24.5. Grundlegende Befehle

Um Befehle an Tmux zu senden, verwendest du den Tmux-Befehlspräfix, der standardmäßig `Ctrl-b` ist, gefolgt von einem anderen Schlüssel. Hier sind einige grundlegende Befehle:

`Ctrl-b "`
: Teilt das aktuelle Fenster horizontal.

`Ctrl-b %`
: Teilt das aktuelle Fenster vertikal.

`Ctrl-b o`
: Wechselt den Fokus zwischen Bereichen.

`Ctrl-b c`
: Erstellt ein neues Fenster.

`Ctrl-b n`
: Wechselt zum nächsten Fenster.

`Ctrl-b l`
: Wechselt zum letzten Fenster.

`Ctrl-b d`
: Trennt die aktuelle Sitzung (diese läuft weiter im Hintergrund).

24.6. Sitzungsmanagement

Tmux ermöglicht es dir, Sitzungen zu verwalten, die im Hintergrund laufen können. Hier sind einige Befehle dazu:

`tmux new -s mysession`
: Erstellt eine neue Sitzung namens "mysession".

`tmux attach -t mysession`
: Hängt sich an eine vorhandene Sitzung namens "mysession" an.

`tmux switch -t mysession`
: Wechselt zu einer vorhandenen Sitzung namens "mysession".

`tmux list-sessions`
: Listet alle aktiven Sitzungen auf.

Bitte beachte, dass die Tastenkombinationen und Befehle konfigurierbar sind und durch die tmux-Konfigurationsdatei (normalerweise `~/.tmux.conf`) geändert werden können.

Mit diesen grundlegenden Befehlen und Konzepten bist du in der Lage, effektiv mit Tmux zu arbeiten und kannst deinen Workflow verbessern. Es gibt natürlich noch viele weitere Befehle und Möglichkeiten zur Anpassung, die du erkunden kannst, wenn du mit Tmux vertrauter bist.

24.7. Links / Cheatsheet

- https://gist.github.com/MohamedAlaa/2961058

25. Vagrant

Vagrant ist ein Open-Source-Tool, das von HashiCorp entwickelt wurde, um die Erstellung und Verwaltung von virtuellen Maschinen-Umgebungen zu vereinfachen. Vagrant ist plattformunabhängig und unterstützt eine Vielzahl von Betriebssystemen wie Linux, Windows und Mac. Darüber hinaus unterstützt es auch eine Vielzahl von Virtualisierungsplattformen, auch Provider genannt, wie VirtualBox, VMware, Hyper-V und mehr.

25.1. Vorteile von Vagrant

- **Einfache Verwendung:** Vagrant bietet eine einfache Befehlszeilenschnittstelle zur Verwaltung von virtuellen Maschinen. Mit einem einzigen Befehl können Sie eine VM starten, stoppen, löschen oder neu starten.
- **Reproduzierbarkeit:** Mit Vagrant können Sie eine "Vagrantfile" -Konfigurationsdatei erstellen, die die Anforderungen Ihrer VM definiert. Dies stellt sicher, dass jeder, der das Vagrantfile hat, genau die gleiche VM-Umgebung erstellen kann.
- **Integration:** Vagrant integriert sich nahtlos mit bestehenden Konfigurationsverwaltungstools wie Chef, Puppet, Ansible und anderen, um die Konfiguration und Verwaltung von VMs zu vereinfachen.

25.2. Wie man Vagrant einsetzt

Um Vagrant zu verwenden, folgen Sie den folgenden grundlegenden Schritten:

1. **Installieren Sie Vagrant:** Laden Sie Vagrant von der offiziellen Website herunter und installieren Sie es auf Ihrem System.
2. **Erstellen Sie eine Vagrantfile:** Eine Vagrantfile ist eine Konfigurationsdatei, die Vagrant sagt, welche Art von Maschine und Ressourcen Sie benötigen, und wie diese konfiguriert werden sollen.
3. **Starten Sie die VM:** Verwenden Sie das 'vagrant up'-Kommando, um die VM zu starten. Vagrant wird die VM entsprechend Ihrer Vagrantfile erstellen und konfigurieren.
4. **Verbinden Sie sich mit der VM:** Verwenden Sie das 'vagrant ssh'-Kommando, um sich mit Ihrer VM zu verbinden und darauf zu arbeiten.
5. **Beenden und Löschen Sie die VM:** Wenn Sie mit Ihrer VM fertig sind, können Sie das 'vagrant halt'-Kommando verwenden, um sie zu stoppen, und das 'vagrant destroy'-Kommando, um sie zu löschen.

Vagrant bietet eine effiziente und flexible Möglichkeit, mit virtuellen Maschinen zu arbeiten. Ob Sie eine isolierte Entwicklungsumgebung benötigen oder eine komplexe VM-Infrastruktur verwalten, Vagrant kann Ihnen dabei helfen.

26. Gegenüberstellung: Ansible, Chef, Puppet und SaltStack

Die Gegenüberstellung von Ansible, Chef, Puppet und SaltStack offenbart unterschiedliche Merkmale und Vorteile, die auf verschiedene organisatorische Bedürfnisse im Bereich der Konfigurationsverwaltung zugeschnitten sind:

26.1. Ansible:

- **Typ**: Überwiegend agentenlos (unterstützt auch agentenbasiert).
- **Hauptmerkmale**: Automatisiert Cloud-Ökosysteme, Anwendungen, Netzwerke, Container, Sicherheit.
- **Kompatibilität**: Funktioniert mit vielen Linux-Versionen, macOS, FreeBSD, Solaris; basiert auf Python.
- **Ansatz**: Verwendet eine Kombination aus prozeduraler und deklarativer Sprache; zielt auf gewünschte Zustände ab.
- **Vorteil**: Vereinfacht IT-Bereitstellungen mit menschenlesbaren Datenbeschreibungen und Modulen für Automatisierung.

26.2. Chef Infrastructure Management:

- **Typ**: Agentenbasiert.
- **Hauptmerkmale**: Automatisiert über Cloud, physische und virtuelle Ökosysteme; unterstützt AIOps.
- **Kompatibilität**: Unterstützt verschiedene Unix-, Linux- und Windows-Versionen.
- **Ansatz**: Verwendet Ruby; betont Skalierbarkeit und präventive Tests für Änderungen.
- **Vorteil**: Gut für Umgebungen, in denen Sicherheit entscheidend ist; ermöglicht autonome Knoten.

26.3. Puppet Enterprise:

- **Typ**: Agentenlos.
- **Hauptmerkmale**: Betont Service-Stabilität und -Zuverlässigkeit; verringert Änderungsfehlerraten.
- **Kompatibilität**: Breite Unterstützung für moderne Betriebssysteme.
- **Ansatz**: Verwendet Infrastruktur als Code mit Zustandsdurchsetzung.
- **Vorteil**: Vereinfacht das Management und ermöglicht eine effizientere Handhabung von mehr Ressourcen.

26.4. SaltStack:

- **Typ**: Bietet sowohl agentenbasierte (Minions oder Proxy-Agenten) als auch agentenlose (SSH/WinRM) Möglichkeiten.
- **Hauptmerkmale**: Ereignisgesteuerte Automatisierung; bewältigt komplexe Szenarien wie mehrstufige Patch-Vorgänge.
- **Kompatibilität**: Unterstützt Windows, verschiedene Linux-Distributionen und Unix.
- **Ansatz**: Python-basiert mit YAML-Unterstützung; kombiniert imperativen und deklarativen Ausführungsansatz.
- **Vorteil**: Passt sich gut an skalierende Umgebungen mit komplexen Anforderungen an.

26.5. Fazit

Das richtige Werkzeug auswählen: Die Entscheidung dreht sich nicht darum, das beste Werkzeug auf dem Markt zu finden, sondern das passendste für die spezifischen Bedürfnisse einer Organisation. Zu berücksichtigende Faktoren sind: - **Agent vs. Agentenlos**: Ansible und Puppet sind agentenlos; Chef verwendet Agenten, und SaltStack bietet beides. - **OS-Kompatibilität**: Dies kann ein entscheidender Faktor sein, abhängig von der Infrastruktur der Organisation. - **Teamkompetenz**: Die Vertrautheit mit den zugrunde liegenden Programmiersprachen (z. B. Python für Ansible, Ruby für Chef) ist wichtig. - **Spezifische organisatorische Anforderungen**: Jedes Werkzeug hat einzigartige Stärken, die für verschiedene Umgebungen und Bedürfnisse geeignet sind.

Letztendlich werden die Stärken und Kompetenzen des IT-Teams maßgeblich beeinflussen, welches Konfigurationsverwaltungsprodukt am geeignetsten ist.

27. Semantische Versionsbezeichnungen

Um semantische Versionsbezeichnung in einer GitLab-Pipeline zu realisieren, können Sie die folgenden Schritte befolgen:

1. Generieren Sie eine neue Versionsnummer basierend auf den Anforderungen der semantischen Versionsbezeichnung. Hierzu können Sie ein Skript verwenden, das in Ihrem Projekt nach spezifischen Commit-Messages sucht, um zu bestimmen, ob es sich um eine Major-, Minor- oder Patch-Änderung handelt.
2. In Ihrem `.gitlab-ci.yml`-Datei, definieren Sie einen Job, der das Skript ausführt und die neue Versionsnummer generiert. Speichern Sie diese Nummer als CI/CD-Variable für die nachfolgenden Jobs.
3. Verwenden Sie diese Versionsnummer in Ihren nachfolgenden Jobs - z.B. beim Bauen, Testen und Bereitstellen Ihrer Anwendung.
4. Schließlich, in einem separaten Job, erstellen Sie ein neues Git-Tag mit dieser Versionsnummer und pushen es zurück in Ihr Repository.

27.1. Beispiel

Hier ist ein einfaches Beispiel für `.gitlab-ci.yml`:

```
stages:
  - versioning
  - build
  - deploy

versioning:
  stage: versioning
  script:
    - VERSION=$(./generate-version.sh) # Verwenden Sie Ihr eigenes Skript zur
Generierung der Version
    - echo "VERSION=$VERSION" >> build.env

build:
  stage: build
  script:
    - source build.env
    - echo "Building version $VERSION"
    # Fügen Sie hier Ihren Build-Code ein

deploy:
  stage: deploy
  script:
    - source build.env
    - echo "Deploying version $VERSION"
    # Fügen Sie hier Ihren Deployment-Code ein
  after_script:
```

```
    - git tag $VERSION
    - git push origin $VERSION
```

In diesem Beispiel verwendet `generate-version.sh` Ihr eigenes Skript zur Generierung der Version. Sie können es so anpassen, dass es die Anforderungen der semantischen Versionsbezeichnung erfüllt.

Bitte beachten Sie, dass Sie geeignete Berechtigungen benötigen, um Tags zu Ihrem Repository hinzuzufügen. Auch kann das Script je nach den spezifischen Anforderungen Ihres Projekts variieren.

27.2. Links

- https://semantic-release.gitbook.io/semantic-release/#highlights
- https://gitlab.com/gitlab-org/gitlab/-/issues/16290

28. Sammlung nützlicher Befehle und Scripte

28.1. git

Abfrage und Sortierung von git-tags (grep -v "-" → blendet rc aus):

```
git tag --sort=-v:refname | grep -v "-" | head -n 3
```

Ausgabe:

```
v16.6.1
v16.6.0
v16.5.3
```

28.2. kubectl

change namespace

```
kubectl config set-context --current --namespace=xxx
```