FI-TS Finance Cloud
dynamisch. sicher. compliant.

Cloud Native - Benutzerhandbuch (v1.1.1)

S2 = zur Weitergabe an Kunden

55050 - Cloud Native

Ein Unternehmen der Finanz Informatik

finanz informatik
technologie service

Inhaltsverzeichnis

1 Überblick

IT-Verantwortliche in vielen Wirtschaftszweigen mussten erkennen, dass große monolitisch strukturierte Anwendungen im Betrieb oft unzureichend skalieren, schwierig zu warten sind und sich aufgrund massiver innerer Abhängigkeiten nur mit sehr hohem personellen, organisatorischen und materiellen Aufwand weiterentwickeln lassen. Angeregt durch agil arbeitende Entwicklercommunitys und neu etablierte Devops-Prozesse in der Industrie sowie die Fähigkeit von Linux, Workloads getrennt in leichtgewichtigen Containern abzuarbeiten, hat in den letzten Jahren ein Wandel hin zu containerisierten Entwicklungs-, Build- und Betriebsprozessen stattgefunden.

Im Gegensatz zu monolitisch aufgebauten Anwendungen zeichnen sich auf mehrere Container verteilte Applikationen durch ihre einfache Skalierbarkeit, leichte Wartbarkeit sowie eine lose Kopplung der einzelnen Funktionseinheiten auf, was deren Weiterentwicklung in der Praxis ungleich leichter gestaltet. Das wegen seiner Open-Source-Strategie zukunftssichere Kubernetes-Projekt ist unschwer als Quasi-Standard für den teilautomatisierten Betrieb komplexer Containerlandschaften in modernen Clouds auszumachen.

Mit den Managed Hosting Services der Produktfamilie **FI-TS Finance-Cloud Native** bietet FI-TS seinen Kunden die Möglichkeit, container-basierte Anwendungen auf einer hochmodernen dedizierten Kubernetes-Plattform zu entwickeln und produktiv zu betreiben. FI-TS-Kunden können des Weiteren aus einem wachsenden Produktset Cloud-native Services rund um die Themen Storage, Datenbanken und Data Streaming wählen. In Abgrenzung zu anderen Marktteilnehmern orientiert sich das Design der Cloudarchitektur in besonderem Maße auf die IT-Sicherheit und die Einhaltung der Compliance-Vorgaben von Banken und Versicherungen. FI-TS betreibt die Kundencluster zudem ausnahmslos in eigenen Rechenzentren in Deutschland. Alle Kubernetes-Cluster von FI-TS unterliegen einem strikten rollen-basierten Zugriffsrechte-System, in das auf Wunsch kundeneigene Berechtigungssysteme integrierbar sind.

Die Kubernetes-Lösung beinhaltet die komplette Hardware-Infrastruktur, die Software sowie den Support, die für das Ausführen von Containern und für deren Orchestrierung durch Kubernetes nötig sind. Jedes Kunden-Cluster wird in einem eigenen Netzsegment mit eigener dedizierter Firewall und eigener Routing-Instanz (VRF) bereitgestellt. Lokal angebundene NVMe-Flashdisks gewährleisten selbst bei stark ressourcenabhängigen Anwendungsfällen stets eine optimale Performance. Kubernetes-Cluster sind inhärent mit Mechanismen zum Schutz gegen Ausfälle ausgestattet. FI-TS berät darüber hinaus über die Möglichkeit, Cluster parallel an mehreren RZ-Standorten zu betreiben.

Die Bereitstellung, das Management und das Löschen von ganzen Kubernetes-Clustern erfolgt automatisiert über ein von FI-TS konzipiertes API. Das Bereitstellen zusätzlicher Clusterserver (sogenannter Worker-Nodes) erfolgt dynamisch anhand der im Kubernetes-Deployment vorgenommenen Ressourcen-Reservierungen. Anders als bei konventionellen Cloudarchitekturen muss der Kunde also nicht in einem vorgegebenen Verfahrensschritt explizit einen Server ordern, wenn er in seinem Cluster mehr Computing-Leistung benötigt.

Die Abrechnung der erbrachten Leistungen erfolgt dynamisch anhand von reservierten virtuellen CPUs, RAM und Storage für die Container des Kunden im erfassten Zeitraum. Bestandskunden benötigen deshalb für die Nutzung der Cloud-Native Plattform eine Vertragserweiterung, die die dynamische Verrechnung regelt.

2 Management der Cloud-Native-Umgebung

FI-TS bietet seine Dienstleistungen auf der Cloud-Native-Plattform in verschiedenen Leistungsschnitten an. Der Kunde darf für einzelne Umgebungen (zum Beispiel Test, Pre-Produktion und Produktion) passend zu seinem Bedarf verschiedene Schnitte zu wählen.

Technisch erfolgen der Bezug und die Verwaltung sämtlicher Cloud-Native-Produkte über APIs. Im ersten Schritt ist dies das von FI-TS übergreifend bereitgestellte Cloud-Native-API, aber auch über die APIs der jeweiligen Produkte lassen sich vertrags- oder service-relevante Änderungen durchführen.

2.1 Kubernetes-Cluster

2.1.1 Leistungsschnitt: Anwendungsbetrieb durch den Kunden

Im einfachsten Leistungsschnitt stellt FI-TS dem Kunden lediglich ein oder mehrere Kubernetes-Cluster bereit. In diesem Szenario dürfen und müssen die Kunden das Management ihrer Cluster selbstständig über das Cloud-API von FI-TS vornehmen. Hierzu gehören insbesondere:

- das Anlegen und Löschen der Cluster
- das Durchführen von Updates und Upgrades
- das Deployment von Anwendungen im Kubernetes-Cluster mit administrativen Rechten

2.1.2 Leistungsschnitt: Anwendungsbetrieb durch FI-TS

In der wertigeren Service-Variante übernehmen die FI-TS die Betriebsverantwortung für die Anwendungen im Kubernetes-Cluster. Die Spezialisten managen die Kubernetes-Cluster in umfassender Weise. Beispielsweise führen sie - wenn möglich - Tests auf einer Pre-Produktionsumgebung durch, bevor sie Updates in Produktivsysteme einspielen. Organisatorische und regulatorische Gründe machen es bei diesem Leistungsschnitt nötig, aus Perspektive des Kunden die Zugriffsrechte auf das Cloud-API der FI-TS und das Kubernetes-Cluster deutlich zu beschränken.

2.2 Cloud-Native-API

Das Anlegen und Verwalten von Cloud-Native-Produkten gelingen über ein REST-API beziehungsweise den zugehörigen Kommandozeilen-Tool `cloudctl`. Das API löst unter anderem die folgenden Aktionen aus:

- Verwalten von Projekten zur logischen Strukturierung von Umgebungen und Anwendungen
- Anlegen, Aktualisieren oder Löschen von Kubernetes-Clustern
- Abruf von Zugriffsdaten (Credentials) für die Kubernetes-Cluster
- Verwalten der reservierten IP-Adressen und Netze
- Abruf von Verbrauchsdaten

Der Zugriff auf die Cloud-Native-APIs erfolgt über das Internet. Für diesen Vorgang ist ein gültiger OpenID-Connect-Token von einem akkreditierten OpenID Connect Authentication Server[1] erforderlich.

2.2.1 Hierarchische Strukturierung der Cloud-Native-Einheiten

Für die Verwaltung der Cloud-Native-Plattform sind die folgenden Strukturierungseinheiten wichtig:

- **Tenant**: Mit Vertragsabschluss legt FI-TS einen Mandanten-Zugang für Cloud-Native an und ordnet diesem Zugang auch den Einzelvertrag für die Abrechnung der in Anspruch genommenen Cloud-Native-Ressourcen zu. (Als Spezialfall kann es sinnvoll sein, dass ein Kunde mehrere Cloud-Native-Einzelverträge schließt. Dann wird FI-TS jeden Vertrag als separaten **Tenant** behandeln.) Innerhalb des Tenant kann der Mandant mehrere **Projekte** anlegen.
- **Projekt**: Projekte dienen der logischen Gruppierung von mehreren Kubernetes-Clustern. Cluster, die zu einem Projekt gehören, können externe IP-Adressen gemeinsam verwenden (zum Beispiel für Loadbalancing oder Failover).
- **Cluster**: Die Cloud-Native-Architektur von FI-TS gestattet es, dass Kunden mehrere unabhängige Kubernetes-Cluster verwenden, um zum Beispiel Stages oder Sicherheitszonen voneinander zu trennen. Um dabei ein sehr hohes Maß an Sicherheit zu gewährleisten, richtet FI-TS Cluster auf dedizierter Server-Hardware (Worker-Nodes und Firewall) ein und trennt es auf Layer-3-Ebene mit Firewall-Systemen von anderen ab.

2.2.2 Reservierung von IP-Adressen

Kunden können für die Services in ihrem Kubernetes-Cluster über das Cloud-Native-API sowohl private[2] als auch öffentliche IP-Adressen reservieren. Wichtig ist, dass sie IP-Adressen beim Reservieren stets einem **Projekt** zuordnen und nur in Kubernetes-Clustern verwenden, die diesem Projekt zugeordnet sind.

[1] Ein OpenID Connect Gateway zur Kopplung vorhandener Verzeichnisdienste (namentlich Active-Directory und LDAP) kann FI-TS im Zuge einer Ersteinrichtung bereitstellen.

[2] Die Cloud-Native-Umgebung vergibt private IP-Adressen aus dem RFC6598-Bereich 100.64.0.0/10. Sie sind über den MPLS-Backbone der FI-TS aus den Bestandsnetzen oder über CredNet erreichbar.

2.2.3 Privilegierte Container

Grundsätzlich beschränkt Kubernetes nicht die Ausführung privilegierter Container.

Seit der Kubernetes-Version 1.23 lassen sich privilegierte Container auf Namespace-Ebene mithilfe des integrierten `PodSecurity Admission Controller` jedoch steuern. Der Default-Pod-Security-Standard für Namespaces kann der Kunde über `cloudctl cluster update <ID> --default-pod-security-standard` festgelegen.

Für Cluster mit Kubernetes Version 1.24 oder früher bestand und besteht noch die Möglichkeit, die veralteten PodSecurityPolcies (deprecated) zu verwenden. Diese Funktionalität kann der Kunde über den Befehl `cloudctl cluster update <ID> --allowprivileged=false` aktivieren.

FI-TS empfiehlt Kunden, Cluster mit Kubernetes ab Versionen 1.24 auf den integrierten PodSecurity Admission Controller anstelle von PodSecurityPolicies umzustellen.

2.2.4 Auditing

Für jedes Cluster kann ein Audit-Log über die Zugriffe auf die `kube-apiserver` API geführt werden und muss über `cloudctl cluster audit` konfiguriert werden. Das Audit-Log kann an unterschiedliche Endpunkte wie Splunk oder für den Test-Betrieb an das eigene Cluster weitergeleitet werden. Details hierzu finden sich in den folgenden Kapiteln. Mittels `cloudctl cluster audit --cluster-id=<id> policy --from-file <path>` kann die Audit-Policy gesetzt werden. Diese Policy beschreibt welche API-Aufrufe an die gewählten Endpunkte weitergeleitet werden.

Standardmäßig wird sichergestellt, dass die gewählten Kubernetes API-Aufrufe am gewählten Endpunkt ankommen bevor der Request ausgeführt wird. Dies kann mittels des `cloudctl cluster audit --cluster-id=<id> mode <mode>` umgestellt werden. Die folgenden Einstellungen stehen zur Auswahl:

- **`batch`**: sammelt Events und leitet diese asynchron in Batches weiter.
- **`blocking`**: blockiert API-Aufrufe bis die Verarbeitung des Events abgeschlossen oder fehlgeschlagen ist.
- **`blocking-strict`**: blockiert ebenfalls die API-Aufrufe. Wenn die Verarbeitung fehlschlägt, wird die Anfrage abgelehnt.

2.2.4.1 Splunk-Konfiguration

Das empfohlene Ziel für die Audit-Logs ist Splunk und kann mittels `cloudctl cluster audit --cluster-id=<id> splunk` konfiguriert werden. Hierfür muss das folgende Kommando mit den entsprechenden Zugangsdaten ausgeführt werden:

```bash
$ cloudctl cluster audit --cluster-id=<id> \
    splunk \
    --enabled \
    --ca <path-to-certificate-authority> \
    --host <the-HEC-host-of-splunk> \
    --index <the-target-index> \
    --port <the-port-of-splunk> \
    --token <the-token-for-splunk>
```

2.2.4.2 Cluster-Forwarding-Konfiguration

Als weiteres Ziel für die Audit-Logs kann das Cluster-Forwarding für den Test-Betrieb verwendet werden. Mittels `cloudctl cluster audit --cluster-id=<id> cluster-forwarding --enabled` können die Audit-Logs an das eigene Cluster weitergeleitet werden. Hierfür wird ein Deployment `audittailer` mit einem Pod im Namespace `audit` erstellt. Die Auit-Log-Meldungen werden von diesem Pod als Container-Log geschrieben und können mit einer beliebigen Kubernetes Logging-Lösung verarbeitet werden.

> ⚠ **Warning**
>
> Wir empfehlen Cluster-Forwarding **nicht** für die Verwendung im Produktions-Betrieb. Grund hierfür ist, dass Anwender, die Zugriff auf die Kubernetes API besitzen, versuchen können das Audit-Forwarding zu blockieren oder zu manipulieren. Um diesen Angriffsvektor auszuschließen, verwenden Sie stattdessen das Splunk Backend. Die Audit-Logs werden dann direkt zu Splunk weitergeleitet ohne den Umweg über das eigene Cluster zu nehmen. Außerdem ist das Feature nicht mit hochverfügbaren Kubernetes Control Planes (siehe Abschnitt 7.2.4) kompatibel.

Mittels `--enabled=false` kann das Cluster-Forwarding wieder deaktiviert werden. Das `audittailer` Deployment wird hierdurch ebenfalls entfernt.

2.3 Kubernetes API

Mit dem Start eines Kubernetes-Clusters entsteht für dieses Cluster zugleich ein API-Server. Das Kubernetes-API ist weitgehend standardisiert und öffentlich dokumentiert. Die

Kubernetes-Implementierung der FI-TS beinhaltet einige Erweiterungen, die den compliance-konformen Einsatz von Kubernetes unterstützen und das Verwalten vereinfachen:

2.3.1 Berechtigungsvergabe

FI-TS berechtigt einen Kubernetes-Cluster-User beim Zugriff auf API-Server auch anhand der Gruppenzuordnung, die sein OpenID-Connect-Token enthält. Hierzu existiert ein Namenskonzept, das die Rechtevergabe in Stufen zulässt (siehe auch IAM Roles Overview). Die Struktur ist so konzipiert, dass sie die Integration in vorhandene Berechtigungsmanagement-Systeme erleichtert.

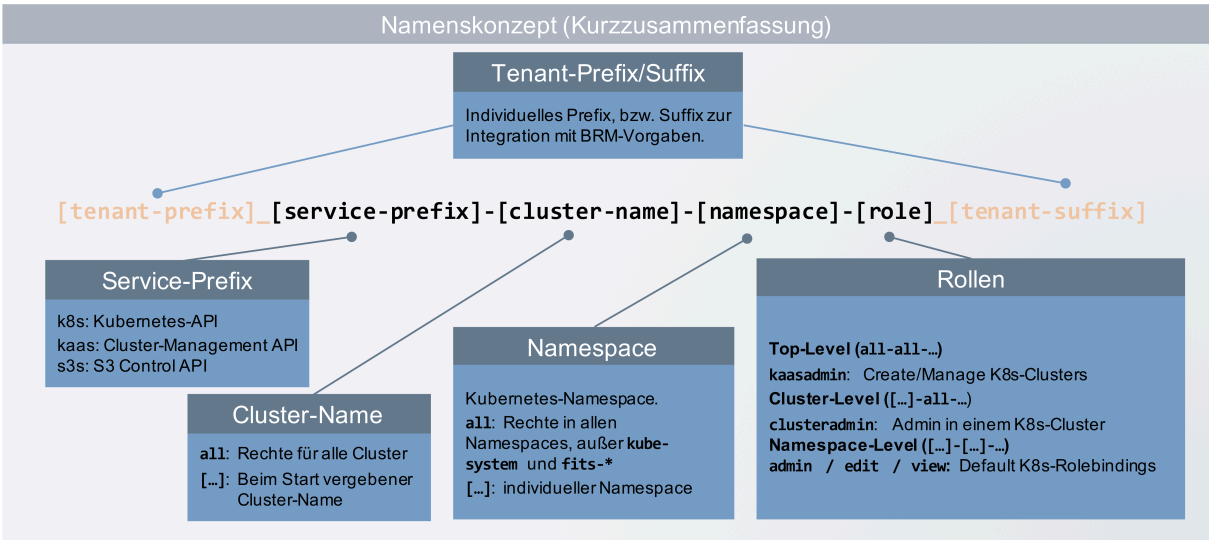

Abbildung 1: Namenskonzept zur Abstufung der Zugriffsrechte

Um auch Zugriffe auf Ebene der Kubernetes-Namespaces zu beschränken, bekommen FI-TS-Kubernetes-Cluster einen Authentication Webhook installiert, der die API-Zugriffe zusätzlich zu den RBAC-Mechanismen von Kubernetes reglementiert.

2.3.2 IP-basierte Einschränkung des Zugriffs auf den Kubernetes API Server

Standardmäßig ist der Kubernetes API Server auch von außerhalb der Cloud-Native Netze aus dem Internet erreichbar. Kunden haben die Möglichkeit, den Zugriff auf den Kubernetes API Server mit `cloudctl` auf bestimmte Netzbereiche einzuschränken (ACL Whitelisting):

```bash
cloudctl cluster update ... --enable-kube-apiserver-acl --kube-apiserver-acl-set-allowed-cidrs <CIDR>,<CIDR>
```

> ⚠ **Warning**
>
> Auch die Mitarbeiter von FI-TS haben nach der Aktivierung dieser Funktion keinen Zugriff mehr auf Ihren Kubernetes-API-Server und können Sie daher bei Problemen innerhalb Ihres Clusters nicht mehr unterstützen.

2.3.3 Technische Benutzer für Continuous Integration

CI-Pipelines steuern ihre Workflows häufig mit Hilfe von technischen Benutzern (Serviceaccounts). Beim Anlegen solcher nicht-interaktiver User ist darauf zu achten, den Accounts nur das funktional unabdingbare Set an Rechten einzuräumen.

Der erste Schritt im folgenden Beispiel legt einen `ServiceAccount` und entsprechende Rollen im Namespace `test` an und bindet die Rollen an den Serviceaccount.

`ci-service-account.yml`:

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ci-service-account
  namespace: test
```

`ci-role-and-rolebinding.yml`:

```yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: ci-role
  namespace: test
rules:
- apiGroups: ["*"]   # '*' nur für das Beispiel, Rechte einschränken!
  resources: ["*"]
  verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: ci-rolebinding
  namespace: test
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: ci-role
subjects:
- namespace: test
  kind: ServiceAccount
  name: ci-service-account
```

Serviceaccount & Role Binding anlegen:

```bash
kubectl create -f ci-service-account.yml
kubectl create -f ci-role-and-rolebinding.yml
```

Danach holt sich der Administrator das Token für den Serviceaccount und erzeugt eine `kubeconfig`:

```bash
kubectl get -n test $(kubectl get secrets -n test -o name | grep ci-service-account-token) -o json | jq -r '.data | .token' | base64 -d
```

2.3.4 Eine Kubeconfig erzeugen

Oft ist für ein Cluster bereits eine `kubeconfig` vorhanden, sodass dessen Administrator diese Datei kopieren und anpassen kann. Sind diese Kubeconfig und der passende Context gerade aktiv, extrahiert er den benötigten Kubeconfig-Ausschnitt:

```bash
kubectl config view --flatten --minify > ci-kubeconfig
```

Dann editiert der Cluster-Administrator den User, indem er die vorhandenen Werte durch das Token ersetzt:

```yaml
apiVersion: v1
kind: Config
clusters:
- cluster:
    certificate-authority-data: ...
    server: ...
  name: name-des-clusters
contexts:
- context:
    cluster: name-des-clusters
    user: ci-service-account
  name: ci-service-account-context
current-context: ci-service-account-context
users:
- name: ci-service-account
  user:
    token: <hier Token des service-accounts einfügen>
```

Wer die Kubeconfig benutzen will, gibt sie `kubectl` als Parameter oder in Form der Environment-Variable `KUBECONFIG` mit.

```bash
kubectl --kubeconfig ci-kubeconfig -n test create deployment nginx --image nginx
kubectl --kubeconfig ci-kubeconfig -n test get pods
```

2.3.5 Firewall, Network-Policies und `ClusterwideNetworkpolicy`

Um eine optimale Trennung der Kubernetes-Cluster von verschiedenen Kunden und Stages zu erreichen, stellt FI-TS für jedes Cluster eine eigene physische Firewall als Übergangspunkt zu anderen Netzsegmenten bereit. Diese verbietet in der Grundeinstellung alle eingehenden Zugriffe (Whitelist-Strategie) und erlaubt nur abgehende HTTPS-Verbindungen.

Ebenso wird Calico als CNI-Provider mit der Unterstützung von Network-Policies aktiviert, um eine Micro-Segmentierung der Netzkommunikation im Kubernetes-Cluster zu etablieren.

Network-Policies sind für die Beschränkung der Kommunikation zwischen Pods und Services im Cluster verantwortlich. Und erst durch Definition der entsprechenden

`ClusterwideNetworkpolicy` im Kubernetes-Deployment wird eine Kommunikation nach außen möglich. Ein spezieller Firewall-Controller setzt diese Policies in Regeln auf der vorgelagerten Firewall um.

Firewall-Freischaltungen

Damit der Firewall-Controller `ClusterwideNetworkPolicy`-Objekte als relevant erachtet, müssen diese zum Namespace `firewall` gehören.

Als Beispiel folgt hier eine `ClusterwideNetworkPolicy` mit einer Egress-Regel für NTP und einer Ingress-Regel für auf Port 8443 eingehende Pakete:

```yaml
---
apiVersion: metal-stack.io/v1
kind: ClusterwideNetworkPolicy
metadata:
  namespace: firewall
  name: clusterwidenetworkpolicy-sample
spec:
  egress:
  - to:
    - cidr: 1.1.0.0/24
      except:
      - 1.1.1.0/16
    - cidr: 8.8.8.8/32
    ports:
    - protocol: UDP
      port: 53
    - protocol: TCP
      port: 53
  ingress:
  - from:
    - cidr: 1.1.0.0/24
      except:
      - 1.1.1.0/16
    - cidr: 8.8.8.8/32
    ports:
    - protocol: TCP
      port: 8443
```

Eine umfassende und stets aktuelle Dokumentation dazu ist beim Metal-Stack-Projekt zu finden.

Services vom Typ `LoadBalancer` setzt der Firewall-Controller in Firewall-Regeln um. Diese können auch Einschränkungen und Freischaltungen anhand von Quell-IP-Netzen enthalten. Hier das Beispiel einer Service-Definition mit einer Whitelist erlaubter Quell-IP-Netze:

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: s1
  namespace: test-ns
spec:
  type: LoadBalancer
  loadBalancerIP: 212.37.83.1
  loadBalancerSourceRanges:
  - 192.168.0.0/24
  - 192.168.2.0/24
  ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 8063
```

Die Service-Definition erlaubt Port-80-Zugriffe zur IP 212.37.83.1 von den Adressbereichen 192.168.0.0/24 und 192.168.2.0/24. Die zugehörige Netfilter-Regel gestaltet sich folgendermaßen:

```bash
ip saddr { 192.168.0.0/24, 192.168.2.0/24 } ip daddr { 212.37.83.1 } tcp
dport { 80 } counter accept comment "accept traffic for k8s service test-
ns/s1"
```

Der Firewall-Controller spiegelt seine Informationen zu Drop-Vorgängen in den Cluster. Sie sind über den `droptailer`-Pod im `firewall`-Namespace einsehbar: `stern -n firewall drop`.

SNAT-Adressen der Firewall

Das Metal-API weist jeder Firewall bei deren Start pro Netzsegment eine dynamische IP-Adresse zu. Die Firewall fungiert in diesem Netzsegment fortan als Ein- und Austrittspunkt, realisiert hierbei Source-NATing und verbirgt somit den cluster-internen Traffic vor dessen Außenwelt.

Kunden, denen die mangelnde Vorhersagbarkeit der zugeteilten Firewall-IP Probleme bereitet, können zur Laufzeit des Firewall-Controllers eine oder mehrere zusätzliche, statische

Adressen zuweisen. Dazu muss der Cluster-Admin per `cloudctl ip allocate` eine statische IP-Adresse des Segmentes reservieren, die er dann bei `cloudctl cluster create` oder `... update` persistiert:

```bash
cloudctl ip allocate --name egress-ip-internet --network internet --project
<PROJECT-ID> --description "static ip used for egress traffic"
cloudctl cluster create ... --egress internet:<Internet-IP_1>
cloudctl cluster update ... --egress internet:<Internet-IP_1> --egress
internet:<Internet-IP_2> --egress mpls:<MPLS-IP>
```

Nebenbedingungen:

- Egress-Adressen lassen sich erst mit Firewall-Images seit dem Build-Datum 22.11.2020 beeinflussen
- Egress-Adressen müssen statisch sein und sind nicht für andere Zwecke verwendbar
- Wichtig: Wer Egress-Einstellungen ändert, muss stets **alle** Adressen **aller** Netzsegmente angeben
- Sind mehrere IP-Adressen pro Netzsegment konfiguriert (`--egress internet:Internet-IP_1 --egress internet:Internet-IP_2`), setzt die Firewall für jede davon SNAT um. Dieses Feature kommt Anwendungen entgegen, bei denen Port Exhaustion (keine freien Ports mehr) infolge vieler offengehaltener Verbindungen droht.

2.3.6 Cloud-Provider-Interface für die FI-TS Infrastruktur

Kubernetes steuert auch die Bereitstellung der für das Cluster notwendigen Hardware-Ressourcen. Ausschlaggebend sind die im Kubernetes-Deployment spezifizierten Ressourcen-Reservierungen[3] (`spec.containers[].resources.requests.cpu`, `spec.containers[].resources.requests.memory` oder Volume-Claims für die Storage-Klasse `csi-lvm`). Ein Autoscaler sorgt dafür, dass zusätzliche Worker-Nodes in das Kubernetes-Cluster aufgenommen werden, wenn die Summe der angeforderten Ressourcen die aktuell verfügbaren überschreitet.

2.3.7 Austausch defekter Nodes

Jeder Node im Kubernetes-Cluster der FCN läuft exklusiv auf einer dedizierten physischen Maschine. Es kann vorkommen, dass diese Nodes ausfallen, sei es aufgrund von defekter Hardware, Software-Fehlern (z.B. im Kernel, Containerd oder Kubelet) oder durch fehlerhafte Anwendungen im Cluster. In solchen Fällen versetzt Kubernetes den betroffenen Node nor-

[3] Die im Kubernetes ebenfalls mögliche Angabe von Limits für die Container-Ressourcen darf die Reservierung überschreiten und ist nicht abrechnungsrelevant. Die im Limit gesetzten Werte lösen aber keine Reaktion des Autoscalers aus. Es wird daher empfohlen, dass die Limits nur moderat über den Reservierungen liegen, um unvorhergesehene Performance-Probleme der Anwendungscontainer zu vermeiden.

malerweise in den Zustand „NotReady“ und verteilt die Workloads, sofern sie keine lokalen Volumes verwenden, umgehend auf andere Nodes im Cluster.

Die FCN ersetzt ausgefallene Nodes innerhalb einer konfigurierbaren Zeitspanne automatisch. Standardmäßig beträgt die Zeitspanne für den Austausch von defekten Nodes 7 Tage. Es wird empfohlen, diesen Wert auf einen niedrigeren Wert zu setzen, beispielsweise auf 24 Stunden. Kunden können dies über die Kommandozeile mit dem Befehl `cloudctl cluster update --healthtimeout 24h` konfigurieren.

Für den Fall, dass der Workernode ein Fehlverhalten zeigt, das Kubernetes nicht automatisch erkennt, oder wenn der Kunde den Austausch eines defekten Nodes aus anderen Gründen sofort durchführen möchten, besteht die Möglichkeit, dies manuell mit `kubectl` zu veranlassen. Dazu muss dem Node die folgende Annotation hinzugefügt bekommen:

```bash
kubectl annotate node <Node-Name> node.machine.sapcloud.io/trigger-deletion-by-mcm=true
```

Dadurch fügt Kubernetes dem Cluster einen neuen Node hinzu und fordert den angegebene Node zum „Drain“ auf. Die Workloads verteilen sich, sofern möglich, auf die anderen Nodes.

Zu beachten ist, dass beim Austausch von Nodes lokale Volumes (StorageClass `csi-lvm`) immer verloren gehen.

Falls eine Firewall offensichtlich oder vermutet defekt ist, lässt sich diese ebenfalls per `kubectl` ausgetauschen. Dazu muss ein Admin das entsprechende `fwmon`-Objekt mit der folgenden Annotation versehen:

```bash
kubectl annotate fwmon -n firewall <Fwmon-Name> firewall.metal-stack.io/roll-set=true
```

Dadurch bekommt der Cluster eine neue Firewall zugeordnet, und die vorhandene Firewall wird entfernt. Dies führt zu einer kurzzeitigen Unterbrechung der Verfügbarkeit des Clusters.

2.3.8 Local NVMe-Storage

Alle Workernodes im Kubernetes-Cluster verfügen über lokal installierte NVM-Express-SSDs. Damit steht jedem Node ein hochperformantes Speichermedium zur Verfügung, das Kubernetes über seinen **Persistent Volume**-Mechanismus (PV) an die Container anzubinden in der Lage ist.

Beim Lösungsdesign haben Kunden zu beachten, dass diese lokalen Volumes als solche nicht redundant ausgelegt sind. Bei einem Hardwaredefekt gehen die Daten auf dem einzelnen Workernode potenziell verloren, auf das Cluster bezogen bleiben die Daten regelmäßig jedoch erhalten, sobald sie eine dafür ausgelegte Anwendung auf andere Knoten repliziert. Wir raten daher grundsätzlich an Cloud Native Block-Storage (beschrieben in Abschnitt 2.3.9 zu verwenden).

Um den lokalen Node-Storage nutzbar zu machen, kann über die Cloud Native API ein von der FI-TS gewartetes CSI Plugin namens csi-driver-lvm im Cluster provisioniert werden. Hierbei ist zu beachten, dass dieses Plugin nicht (mehr) standardmäßig im Cluster bereitgestellt wird, sondern bei der Anlage des Clusters mit dem Flag `--enable-csi-driver-lvm` explizit aktiviert werden muss. Das Plugin kann auch noch nachträglich über das `cloudctl cluster update --enable-csi-driver-lvm` aktiviert oder deaktiviert werden.

Daraufhin werden weitere `StorageClasses` zum Cluster hinzugefügt, die in `PersistentVolumeClaims` (PVCs) verwendet werden können. Die Storage Classes basieren auf der Verwendung von Linux LVM. Hier eine Erläuterung zu den Storage Classes:

- `csi-driver-lvm-linear`: Daten werden sequenziell auf den zur Verfügung stehenden Speichermedien gespeichert.
- `csi-driver-lvm-mirror`: Daten werden gleichmäßig auf den zur Verfügung stehenden Speichermedien verteilt, was die Lese- und Schreibgeschwindigkeit erhöht. Dies bietet eine höhere Performance, aber der Ausfall eines Volumes führt zu Datenverlust.
- `csi-driver-lvm-striped`: Daten werden auf den zur Verfügung stehenden Speichermedien dupliziert, was die Datensicherheit erhöht. Die Ausfallsicherheit ist hier höher, allerdings steht in Summe weniger Speicherplatz zur Verfügung (50% der Kapazität).

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: lvm-pvc
  namespace: default
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: csi-driver-lvm-linear
  resources:
    requests:
      storage: 50Mi
```

> ⚠ **Warning**
>
> Manche Workernodes verfügen nur über eine einzige lokale NVMe-Disk. In diesem Fall entstehen aus der Verwendung der Storage Classes `csi-driver-lvm-mirror` oder `csi-driver-lvm-striped` keine Vorteile. Erkundigen Sie sich ggf. vor dem Einsatz von lokalem Storage über die Workertypes, die für den Cluster verwendet werden sollen.

2.3.8.1 Migration von csi-lvm zu csi-driver-lvm

In der Vergangenheit wurde in jedem Cluster eine Storage Class `csi-lvm` zur Verfügung gestellt. Diese Installation wird auch bei bestehenden Clustern weiterhin fortgeführt. Allerdings implementiert das dazugehörige Plugin eine inzwischen veraltete Version der CSI Spec von Kubernetes. Es gibt inzwischen eine Nachfolgeversion mit dem Namen `csi-driver-lvm`, die neben der Möglichkeit `striped` und `mirror` Volumes zu konfigurieren auch die Vergrößerung von Volumes erlaubt.

Zum jetzigen Zeitpunkt existiert kein direkter Migrationspfad, um die existierenden Volumes auf das Management durch das neue CSI Plugin zu migrieren. Aktuell wäre eine Umstellung auf das neue CSI Plugin nur auf folgendem Weg möglich:

1. Entfernen aller PVs / PVCs, die mit der alten Storage Class `csi-lvm` (Provisioner `metal-stack.io/csi-lvm`) angelegt wurden.

1 Deaktivierung des alten CSI Plugins mit dem Kommando: `cloudctl cluster update <id> --enable-csi-lvm=false`. Dies entfernt auch die `csi-lvm` Storage Class aus dem Cluster.

1. Aktivierung von `csi-driver-lvm` über: `cloudctl cluster update <id> --enable-csi-driver-lvm`. Es erscheinen nun die in Sektion Abschnitt 2.3.8 beschriebenen Storage Classes.
   - Aus Komptabilitätsgründen wird wieder eine `csi-lvm` Storage Class erscheinen, damit bestehende Kubernetes Manifeste weiterhin funktionieren.
1. Ggf. können über vom Cluster Anwender eingesetzte Backup-Restore-Mechanismen die Daten bei der Neuanlage der PVs wiederhergestellt werden.

> ⚠ **Warning**
>
> Theoretisch ist es möglich, das alte CSI Plugin zu deaktivieren ohne die bestehenden PVs vorher zu löschen. In dem Fall laufen existierende Container zwar weiter, die dazugehörigen Volumes sind aber verwaist. Das neue CSI Plugin verwaltet diese Volumes nicht weiter. Spätestens nach einem Reboot des Nodes können diese Volumes nicht mehr

> gemounted werden. Das Löschen von verwaisten PVs gibt den Speicher auf den Platten nicht wieder frei.

Optional wäre eine eigene Installation von `csi-driver-lvm` durch den Cluster Anwender denkbar, um beide Plugin Versionen parallel zu betreiben. In dem Fall könnten Tools wie pv-migrate zur Migration der Daten eingesetzt werden. In jedem Fall sollte ein solcher Migrationsweg im Vorfeld gründlich erprobt werden.

2.3.9 Cloud Native Block-Storage

Neben dem lokalen Storage bietet FI-TS auch zentralen Blockstorage an jedem RZ-Standort an, der unabhängig von Workernodes persistent und über das Netz erreichbar ist. Die Performance ist vergleichbar mit der einer Workernode-lokalen Festplatte. FI-TS empfiehlt, Cloud Native Blockstorage zu benutzen, da Kubernetes-Clusterupdates, -migrationen und das Workernode-Scaling ungleich leichter von der Hand gehen als mit Local NVMe-Storage.

Ein CSI-Provider in Kubernetes bindet Volumes auf diesem Storage automatisch an einen Node, wenn ein Pod den jeweiligen **PersistentVolumeClaim** nutzt. Ein Persistent Volume ist aber jeweils nur an einem Node gleichzeitig nutzbar (**Access-Mode: RWO** - `ReadWriteOnce`).

Damit ein Kubernetes-Cluster den zentralen Blockstorage erreichen kann, muss es an das Storage-Netz der jeweiligen Cloud Native Partition angebunden sein:

- Fellbach (`fel-wps101`) – `59e0f510-41bf-425c-a062-424ae7dad630`
- Stuttgart (`stg-kkw701`) – `cd12be9f-74d9-4460-8b06-e13a51faee23`
- Nürnberg (`nbg-w8101`) – `282cd3b2-fe3f-47e8-b78e-d074aa9c20f5`
- Nürnberg 2 (`n2-tm1601`) – `0c3ac419-e7f8-49ae-a2ec-bcc6ade99602`

Die IDs der Storage-Netzwerke können Benutzer auch über das Kommando `cloudctl cluster inputs` einsehen.

Diese Netzanbindung erfolgt bei der Cluster-Erstellung normalerweise automatisch und passend zum Standort.

Altsysteme, bei denen das nicht der Fall ist, kann der Cluster-Admin über das Kommando `cloudctl cluster update <Cluster-UID> --external-networks <NetworkList>` nachkonfigurieren. In der `NetworkList` ergänzt er hierbei per Komma separiert die entsprechende Storage-Netz-UUID. Die bestehenden Netze lassen sich zuvor mittels `cloudctl cluster describe <ClusterID> -o json | jq .AdditionalNetworks` auslesen.

Ist das Storage-Netz vom Cluster aus erreichbar, steht eine zusätzliche Storage Class `partition-gold` bereit. Dies lässt sich zum Beispiel mit `kubectl get StorageClasses` überprüfen.

Zum Verwenden des Block-Storage muss ein `PersistentVolumeClaim` (PVC) der **Storage Class** `partition-gold` angelegt sein. Beispiel:

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: fcn-block-pvc
  namespace: default
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: partition-gold
  resources:
    requests:
      storage: 50Mi
```

Es besteht weiterhin die Möglichkeit, mit dem Befehl `cloudctl cluster update <Cluster-UID> --default-storage-class partition-gold` diese Storage Class automatisch bei allen neu angelegten PVCs zu verwenden.

2.3.9.1 Nutzung bestehender Block-Storage-Volumes

Blockstorage-Volumes bleiben über die Laufzeit der Kubernetes-Cluster hinaus bestehen. **Voraussetzung:** Die `ReclaimPolicy` des PV muss auf `Retain` gestellt sein. In der Grundeinstellung steht die `ReclaimPolicy` auf `Delete`, was bewirkt, dass Kubernetes das PV nach dem Löschen des zugehörigen Clusters ebenfalls löscht!

Innerhalb desselben **Projekts** und derselben **Partition** dürfen Kunden bereits mit Nutzdaten gefüllte und mit `Retain` parametrisierte Volumes in neuen Kubernetes-Clustern dagegen wieder anbinden. Hierzu können Sie mit `cloudctl volume manifest` eine Konfigurationsvorlage für das bestehende **PersistentVolume** erzeugen.

2.3.9.2 Snapshots

Kubernetes **ab 1.20.x** unterstützt nativ das Anfertigen von Volume-Snapshots, wenn der zugehörige CSI-Treiber dies zulässt. Die Vorgehensweise beschreibt die offiziellen Kubernetes-Dokumentation detailliert: Volume Snapshots.

Aus einem Snapshot können Kunden wieder ein Volume erzeugen, zum Beispiel, um davon ein Backup anzufertigen oder um einen Test mit den Orginaldaten durchzuführen. Wichtig zu wissen: Der Quell-Snapshot lässt sich erst dann wieder löschen, wenn alle davon abgeleiteten Volumes ebenfalls gelöscht wurden.

Die vorhanden Snapshots kann sich der Cluster-Admin mittels `cloudctl volume snapshot ls --project <Project-UID>` anzeigen lassen und sie mittels `cloudctl volume snapshot delete <Volume-UID>` löschen.

2.3.9.3 Optional: Clientseitige Volume-Verschlüsselung

Es gibt Kunden, die infolge regulatorischer Vorgaben oder aus eigenem Antrieb persistente Daten verschlüsselt ablegen wollen. FI-TS FCN Hosting biete solchen Kunden an, ihre Nutzdaten in Volumes auf Cloud Native Blockstorage kryptografisch geschützt abzulegen. Zum Einsatz kommt hierbei die aus dem Linux-Kernel bekannte Blockverschlüsselungstechnik LUKS2, deren Stärke und Implementierungsqualität in Fachkreisen anerkannt ist. Im konkreten Fall passiert die Ver- und Entschlüsselung als Client-side Encryption im Kubernetes-Cluster (ab Version 1.20.x!) des Kunden, genauer: auf dem Workernode, der das PV verwendet. Auf dem Ziel-Block-Storage-System ist das zugehörige Volume von Anfang an verschlüsselt, also schon vor dem Anlegen des Filesystems.

Dazu ist es notwendig, dass der Kunde zuerst mit dem Befehl `cloudctl cluster update <Cluster-UID> --encrypted-storage-classes` die entsprechende **Storage Class** im Cluster installiert.

Für die Nutzung der eigentlichen Verschlüsselung muss nun noch ein Secret im Namespace der Anwendung angelegt werden:

```yaml
---
apiVersion: v1
kind: Secret
metadata:
  name: storage-encryption-key
  namespace: default # Hier bitte den entsprechenden Namespace eingeben
stringData:
  host-encryption-passphrase: please-change-me # Natürlich ein besseres Passwort
  vergeben
type: Opaque
```

Das Cloud-API unterstützt den Kunden beim Anlegen des Secrets mittels `cloudctl volume encryption-secret-manifest --namespace <Namespace> --passphrase <Passphrase>`. Danach

kann er den PVC mit der Storage Class `partition-gold-encrypted` anlegen und zugleich verschlüsseln.

FI-TS weist darauf hin, dass bei Verlust des Secrets die gespeicherten Nutzdaten unwiederbringlich verloren sind, und empfieht darum, Secret-Kopien an geeigneter Stelle abzulegen. Der Einsatz von Verschlüsselung hat außerdem Auswirkung auf die Performance des Block Storages. So ist mit einer spürbaren Reduktion des Durchsatzes sowie einer erhöhten Latenz zu rechnen.

2.3.10 S3-kompatibles Object-Storage

In Containerumgebungen hat sich zum persistenten Speichern von Daten vielerlei Art sogenannter S3-Storage durchgesetzt. FI-TS betreibt auf hochmoderner Hardware für seine Cloud-Native-Kunden solche Speichersysteme als Managed Service, der innerhalb jeder Partition als ein AWS-S3-kompatibler Object Store zur Verfügung steht.

Falls noch nicht vorhanden, muss der Hauptbenutzer anfänglich mit `cloudctl project create` unter Nennung des Mandanten (englisch: Tenant) ein Projekt anlegen. Beispiel:

```bash
cloudctl project create --name "s3-test" --description "Mein S3-Test" --
tenant grossbank
```

Dann lässt sich mittels `cloudctl s3 create --id ... --project <Projekt-UID> --partition -p <Rechenzentrum>` der eigentliche Benutzer für S3 angelegen. Bei der ID kann man sich einen Namen für den Benuter aussuchen, solange dieser noch nicht vergeben ist. Das Rechenzentrum (`-p`), in dem die Daten gespeichert werden sollen, ist Nürnberg, Nürnberg 2, Fellbach oder Stuttgart. Hinweis: Es findet keine automatische Geo-Replizierung zwischen den RZs statt.

Um sich die verfügbaren Rechenzentren anzuzeigen gibt es folgenden Befehl:

```bash
cloudctl s3 partitions
```

Die Ausgabe des Befehls transportiert ein Schlüsselpaar (Accesskey, Secretkey). Diese sind dringend zu notieren, weil sie dem Benutzer als Credentials für seinen Zugriff auf S3 dienen werden. Auf diese Weise kann er mit jedem üblichen S3-Client Buckets anlegen sowie Objekte speichern, herunterladen und so weiter. Dafür empfiehlt sich beispielsweise das Kommandozeilen-Tool MiniO MC Client, das über Github ladbar ist:

```bash
wget https://dl.min.io/client/mc/release/linux-amd64/mc
chmod +x mc
./mc --help
```

Die folgenden Beispiele zeigen, wie sich Zugriffe auf Buckets beziehungsweise Objekte einrichten lassen. Via Bucket Policies gelingt es darüber hinaus, Rechte S3-kompatibel einzuschränken oder zu erweitern.

Beispiel 1: Zugriff auf bestimmte IP-Adressen einschränken:

```json
{
  "Statement": [
    {
      "Effect": "Deny",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "s3:ListBucket",
        "s3:DeleteObject",
        "s3:GetObject",
        "s3:DeleteBucket",
        "s3:PutObject"
      ],
      "Resource": ["arn:aws:s3:::mybucket","arn:aws:s3:::mybucket/*"]
      "Condition": {
        "NotIpAddress": {
          "aws:SourceIp": ["212.34.64.0/19","185.153.64.0/22"]
        }
      }
    }
  ]
}
```

Beispiel 2: Einem anderen S3-User Read-only-Zugriff auf ein Bucket gewähren:

```json
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": ["arn:aws:iam::Ft:user/cd4eac58-46a5-4a31-
        b59f-2ec207baa817_otheruser"]
      },
      "Action": [
        "s3:ListBucket",
        "s3:GetObject"
      ],
      "Resource": ["arn:aws:s3:::mybucket","arn:aws:s3:::mybucket/*"]
    }
  ]
}
```

(`Principal` setzt sich hier zusammen aus: `arn:aws:iam::<Tenant>:user/<Project-ID>_<User-ID>`)

Beispiel 3: Öffentlichen (Anonymous)-Zugriff auf Bucket-Inhalte gewähren:

```json
{
  "Statement":[
    {
      "Effect":"Allow",
      "Principal": "*",
      "Action":["s3:GetObject"],
      "Resource": ["arn:aws:s3:::mybucket","arn:aws:s3:::mybucket/*"]
    }
  ]
}
```

Ein Download passiert im einfachsten Fall über einen Browser. Die URL folgt dem Schema `<S3-Endpoint>/<Tenant>:<Bucket>/<File>`, zum Beispiel `https://s3.prod-01-fel-wps101.fits.cloud/Ft:mybucket/myfile.zip`.

2.4 Middleware und Datenbanken

Auf die Kubernetes-Plattform aufsetzend bietet FI-TS verschiedenen Middleware- und Datenbank-Produkte als (Managed) Services an. Kunden der Cloud-Native-Plattform können derzeit folgende Dienste buchen:

- PostgreSQL
- Elasticsearch
- Kafka

Ähnlich wie beim Kubernetes-Service übernimmt der Anwendungsbetrieb der FI-TS die Steuerung der Middleware- und Datenbank-Updates im Rahmen des Anwendungsbetriebs.

2.5 Cloud Platform Status Dashboard

Die allgemeine Verfügbarkeit der von der Cloud-Native-Plattform bereitgestellten Services kann über eine Webseite abgefragt werden. Die Seite enthält außerdem Hinweise zur Verwendung der Plattform, Zusatzinformationen zu Updates und Störungen sowie Download-Links zur aktuellsten Version des Benutzerhandbuchs.

Sie können das Dashboard unter folgender URL erreichen: https://status.fits.cloud/.

2.6 Beendigung

Die Cloud-Native Ressourcen werden anhand ihrer Nutzung verrechnet. Durch die Terminierung von Ressourcen über die API ist es möglich auch die anfallenden Kosten zu reduzieren. Es ist aber zu beachten, dass bei der Terminierung auch eventuell gespeicherte Daten unwiederbringlich gelöscht werden. Diese müssen gegebenenfalls vor dem Löschen exportiert werden. Hierzu sind verschiedene Methoden möglich. Im Folgenden werden einige Vorschläge für den Export und die Vorgehensweise zum Löschen der Daten beschrieben.

2.6.1 Export der Kubernetes-Konfiguration und -Volumes

Die im Kubernetes-Cluster vorhandene Konfiguration und die auf lokalen oder Blockstorage-Volumes gespeicherten Daten lassen sich über das Kubernetes-API exportieren. Hierfür existieren mehrere Werkzeuge. Eines der bekanntesten ist das Open-Source-Tool Velero. Dieses ist für Backups oder für die Migration von/zu einem anderen Kubernetes-Anbieter nutzbar.

Details zur Nutzung von Velero sind unter https://velero.io/docs/ dokumentiert.

2.6.2 Löschen von Kubernetes-Clustern

Kubernetes-Cluster können mittels `cloudctl cluster delete` (bzw. über die FI-TS Cloud API) gelöscht werden. Hierbei werden auch die Worker-Nodes des Kubernetes-Clusters freigegeben. Um die Daten der Kunden zu schützen, wird im Zuge der Node-Freigabe auch immer der Schlüssel der lokalen Festplatten (NVMe-Disks) verworfen. Damit gehen alle auf lokalen

Volumes gespeicherten Daten verloren. Auch FI-TS hat keine Möglichkeit, diese Daten wiederherzustellen.

Nicht gelöscht werden die Daten die aus dem Kubernetes-Cluster heraus auf *Cloud-Native Block-Storage* oder auf **S3-Buckets** gespeicherte Daten.

2.6.3 Löschen von Block-Storage Volumes

Blockstorage-Volumes bleiben auch über die Laufzeit des Kubernetes-Clusters hinaus erhalten. Diese können dann später an ein Kubernetes-Cluster (im selben Cloud-Native-Projekt) angebunden werden.

Sollen auch die Daten auf den Block-Storage Volume gelöscht werden, müssen diese explizit mittels `cloudctl volume delete` gelöscht werden. Eine Übersicht zu allen bestehenden Volumes ist mit `cloudctl volume list` abrufbar.

Ein Export der Volume-Daten ist nur über die Kubernetes-API möglich (siehe oben). Volumes, die nicht an ein Kubernetes-Cluster angebunden sind, müssen erst mit einem Cluster verknüpft werden, bevor diese (z.B. mit Velero) exportiert werden können.

2.6.4 S3 Object-Store

„Datentöpfe“ auf dem **S3-kompatiblen Object-Store** werden in sogenannten Buckets organisiert. Auf diese kann über das S3-API zugegriffen werden. Credentials für den API-Zugriff verwalten Anwender mittels `cloudctl s3`. Für den Zugriff über das S3-API existieren ebenfalls wieder diverse Open-Source Tools. Für den Export oder die Migration gut geeignet ist der Minio-Client. Nachdem die Credentials für den Zugriff konfiguriert wurden, können mittels `mc ls` Informationen zu den Buckets und den dort gespeichert Daten abgerufen werden. `mc mirror` überträgt die Daten aus dem S3-Bucket auf ein lokales Verzeichnis.

Vor dem Löschen des Buckets müssen Anwender die Objekte der dort gespeicherten Daten löschen. In Anlehnung an die Vorgehensweise unter Unix geschieht dies mit `mc rm ...`. Zum endgültigen Löschen der Buckets dient das Kommando `mc rb`. (Hinweis: Wer beim Upload der Daten Server-Side-Encryption verwendet, muss erforderlicherweise den Encrytion-Key auch beim Löschen mit angeben - siehe `mc`-Dokumentation).

3 Patchmanagement und Lifecycle

3.1 Patchmanagement Cluster

Die Verantwortlichkeiten für das Patchmanagement der Software-Komponenten der Kubernetes-Cluster sind vom vereinbarten Leistungsschnitt abhängig.

3.1.1 Kubernetes-Update/-Upgrade

FI-TS beobachtet und bewertet laufend das Entwicklungsgeschehen beim Kubernetes-Projekt, um Kunden stets eine sichere und moderne Ablaufumgebung für deren Container-Workloads bereitzustellen. Als Ergebnis der Expertise bereitet FI-TS Kubernetes-Update-Sets auf und unterscheidet dabei zwischen:

- **Patch-Release-Updates**, zum Beispiel von Version 1.16.1 auf 1.16.2: Solche kleinen Versionsupdates, die neben funktionsbeeinträchtigenden Bugs auch erkannte Sicherheitslücken beheben, spielt der FI-TS-Cluster-Manager (nach ausgiebigen Tests in einem FI-TS-eigenen Cluster) automatisch in die Kundenumgebung ein. Der Update-Vorgang erfolgt für die Anwendungscontainer in der Regel unterbrechungsfrei, es sei denn, die besitzen eine direkte Abhängigkeit zu Kubernetes-Services, beispielsweise weil sie laufend auf den Kubernetes-API-Server zugreifen.
- **Minor-/Major-Updates**, zum Beispiel von Version 1.16._x_ auf 1.17._x_: Solche funktionserweiternden Updates stellt FI-TS zwar bereit, installiert sie aber nicht automatisch. Es obliegt somit dem Kunden oder dem Anwendungsbetrieb, das Updates im Rahmen des Change-Prozesses in die Kubernetes-Cluster einzuspielen. Hierzu eignet sich das Kommando `cloudctl cluster update`. Der Cluster-Manager prozessiert dann ein Rolling Update, das die Kubernetes-Version aktualisiert. Typisch für das Szenario ist, dass die Anwendungscontainer neu starten - gegebenenfalls auch auf anderen Worker-Nodes als bisher.

Das Produkt *FI-TS Finance Cloud Native Kubernetes* beinhaltet den Softwaresupport für die aktuelle und die beiden vorhergehenden Minor- und Major-Versionen.

3.1.2 Betriebssystem-Update/-Upgrade für Nodes

Auch für die Betriebssysteme der Nodes stellt FI-TS regelmäßig aktualisierte Betriebssystem-Images bereit, spielt diese jedoch nicht automatisiert ein. Vielmehr ist vorgesehen, neue Images im Zuge des Cluster-Updates (`cloudctl cluster update`) zu verteilen. Der Erneuerungsvorgang passiert im Cluster als Rolling Update, das Worker-Nodes zeitlich nacheinander aktualisiert.

Der eingesetzte Kubernetes Cluster-Manager führt hierzu einen *Drain* auf dem entsprechenden Node durch und verteilt die auf dem Knoten laufenden Container auf andere Worker, wo sie neu starten. Anschließend bekommt ein neuer Knoten ein aktuelles Betriebssystem-Image aufgespielt und ein neu erstelltes Local-Persistent-Volume. Nach einem Reboot wird der neue Node ins Cluster integriert.

Das Auswechseln der Local-Persistent-Volumes hat zur Folge, dass Kubernetes-Pods, die auf ein Local-Persistent-Volume zugreifen und sich damit an ihren Knoten gebunden haben, im Update-Zeitfenster nicht zur Verfügung stehen.

3.1.3 Release-Reporting und Software-Lifecycle für Kubernetes-Cluster

Cloud-Native-Kunden der FI-TS können sich mit Hilfe von `cloudctl cluster issues` einen Überblick über die cluster-weit eingesetzten Softwareversionen verschaffen, insbesondere für Kubernetes, die Betriebssysteme der Worker-Nodes sowie Komponenten, die über eine eigene Firmware verfügen. Anhand der Analyse ergeben sich Empfehlungen Updates durchzuführen. FI-TS kategorisiert hierbei nach:

- **Stable**: Die aktuell empfohlene Version. Diese hat FI-TS in einer Testumgebung überprüft und benutzt sie per Default, wenn der Neustart eines Kubernetes-Clusters nötig ist.
- **Deprecated**: Eine gegenüber der Stable-Version veraltete, aber noch mit Support unterlegte Variante. FI-TS empfiehlt, die Software so bald wie möglich einem Minor- oder Major-Update zu unterziehen. Dem erzeugten Bericht kann der Kunde entnehmen, wann der Support abläuft.
- **Unsupported**: Eine veraltete Version, für die kein Support mehr besteht. FI-TS rät nachdrücklich davon ab, Cluster mit unsupporteter Software zu betreiben. Solange keine Sicherheitslücken bekannt sind, die aus der Ferne ausnutzbar sind (Remote Exploits), wird FI-TS solche Systeme nach dem „Best Effort“-Prinzip weiter betrieben, das vereinbarte SLA aber bis zur Heilung des Versions-Mangels aussetzen. Bei Bekanntwerden eines kritischen Remote Exploits behält sich die FI-TS vor, das Cluster im Rahmen eines Notfall-Changes umgehend zu isolieren.

Des Weiteren informiert FI-TS Kunden per E-Mail über wichtige Statusänderungen, beispielsweise die bevorstehende Löschung eines Evaluierungs-Clusters.

Wenn zwei oder mehr Cluster einen Verbund zur Realisierung einer Notfallvorsorge bilden, empfiehlt FI-TS die Aktualisierung in kurzer Folge durchzuführen. Eine technische Notwendigkeit hierfür besteht zwar nicht, kann im Einzelfall aber ein inkonsistentes Verhalten der Anwendungen vermeiden.

Trägt FI-TS vereinbarungsgemäß die Verantwortung für den Anwendungsbetrieb, wird die Fachabteilung für den Anwendungsbetrieb die zugehörigen Lifecycle-Reports auswerten, erforderliche Updates mit dem Kunden abstimmen und im Rahmen der Changes durchführen.

3.2 Patchmanagement für Middleware und Datenbanken

Auf die Kubernetes-Plattform aufsetzend bietet FI-TS verschiedenen Middleware- und Datenbank-Produkte als (Managed) Services an. Der Umfang des zugehörigen Patchmanagements unterscheidet sich je nach gebuchtem Leistungsschnitt.

Analog zur Kubernetes-Strategie stellt FI-TS diese Produkte entweder in einem gehosteten Service-Cluster bereit oder - wenn auch den Anwendungsbetrieb vereinbart ist - richtet die Dienste im eigenen Kubernetes-Cluster ein.

3.2.1 Updates/Upgrades

Für die Durchführung von Updates und Upgrades stehen für den Anwendungsbetrieb beziehungsweise den Kunden, abhängig vom Automatisierungsgrad, zwei Varianten zur Verfügung:

1. **Automatisiertes Update über das Cloud-API**: FI-TS wird für Produkte, die sie bei mehreren Kunden einsetzt, spezielle Funktionen in das Cloud-API der FI-TS integrieren, über die sich das Produkt im Lebenszyklus (Installation, Updates, Löschen, etc.) verwalten lässt.

2. **Updates durch den FI-TS-Betrieb**: Solange kein API für ein Produkt zur Verfügung steht, können Kunden Updates oder sonstige Änderungen über Aufträge im Rahmen des Change-Prozesses an die Gruppe `FITS55052-Cloud-Native-Services` stellen.

3.2.2 Release-Reporting und Software-Lifecycle

Analog zu den Kubernetes-Services liefert die Cloud-Native-Reportingfunktion auch die Release-Informationen für die von FI-TS gepflegten Middleware- und Datenbank-Container. Die Kategorien (Stable, Deprecated, Unsupported) und Regelungen bezüglich der SLAs gelten hier in gleicher Weise.

4 Netzanbindung

Die Cloud-Native-Umgebungen (Partitionen) von FI-TS verfügen über Verbindungen zum Internet- und, auf Kundenwunsch, zum MPLS-Backbone. Über den MPLS-Backbone sind die Bestandsnetze in den FI-TS-Rechenzentren, Verbundnetze der Sparkassen-Organisation und das Crednet erreichbar. Verbindungen zwischen zwei Partitionen laufen ebenfalls über diese Backbone-Netze.

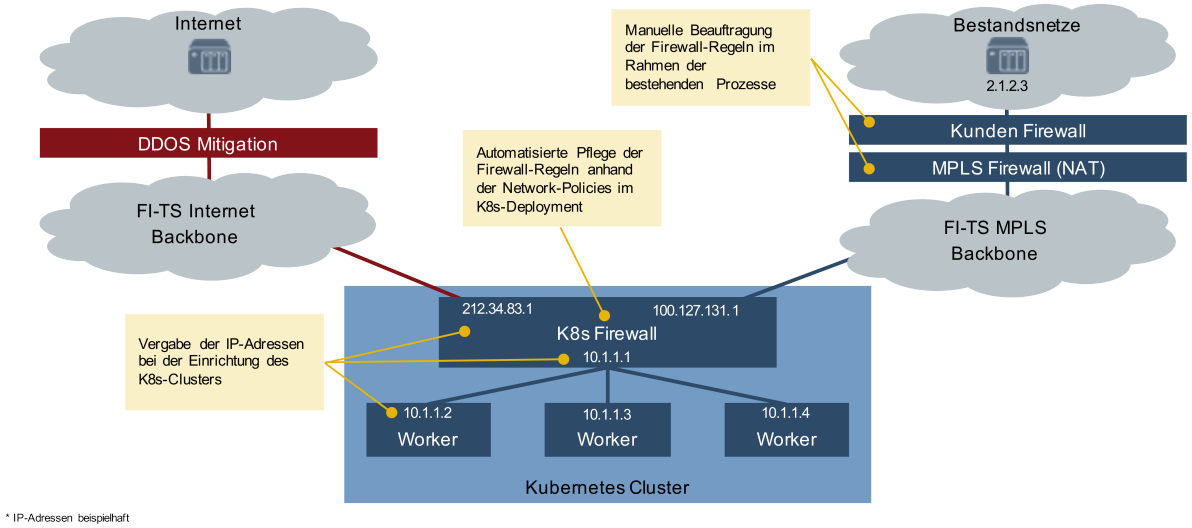

Abbildung 2: Überblick zur Cloud-Native Netzanbindung

Jedes über das Cloud-Native-API gestartete Kubernetes-Cluster bekommt eine eigene physische Firewall zugewiesen, die den Verkehr in und aus dem Kubernetes-Cluster reglementiert. Die Regeln für die Firewall werden automatisch aus den ausgerollten `ClusterwideNetworkpolicy`-Sets generiert (siehe Policies-Kapitel).

Jede Firewall erhält bei ihrem Start eine öffentliche Internet-IP-Adresse und auf Wunsch auch eine RFC6598-Adresse (das ist der Bereich 100.64.0.0/10) für die Kommunikation über den MPLS-Backbone. Außerdem ist es in beiden Bereichen möglich, weitere IP-Adressen für Services zu beziehen, die aus dem Internet beziehungsweise aus den MPLS-Bereichen erreichbar sind. Für die Pods und die Worker in den Kubernetes-Clustern kommen RFC1918-Adressen aus dem Bereich 10.0.0.0/8 zur Anwendung.

4.1 Internet-Verbindungen

Bei den abgehenden Verbindungen aus dem Bereich 10.0.0.0/8 verbirgt das Cluster die Quell-Adressen durch NATing (Masquerading der Firewall) hinter der jeweiligen Firewall-IP.

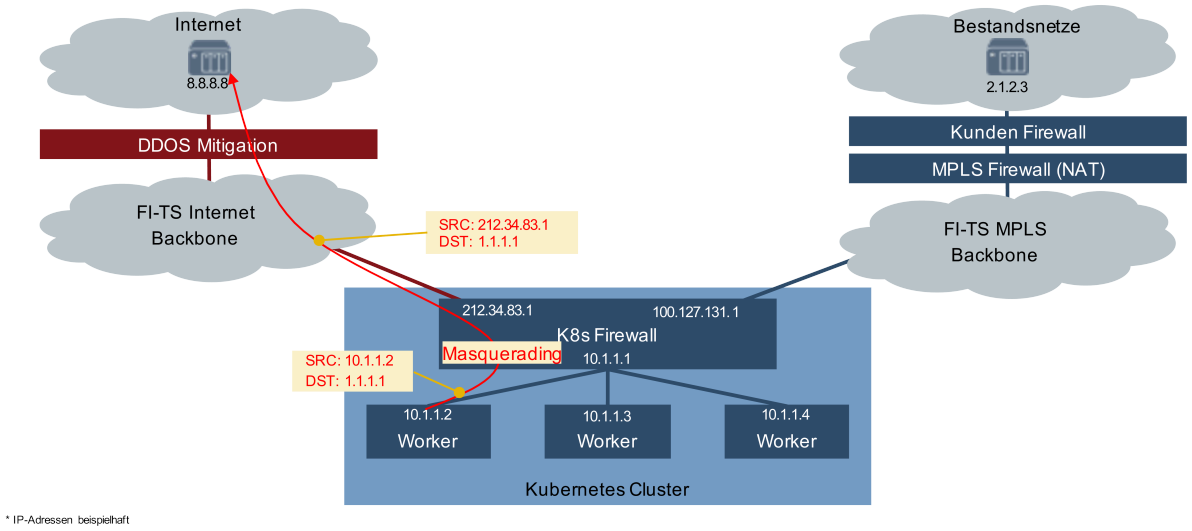

Abbildung 3: Ausgehende Internet-Verbindungen

Soll umgekehrt ein Service des Clusters für User aus dem Internet erreichbar sein, muss er dafür eine extra Internet-IP zugeordnet bekommen. Dies geschieht automatisch, falls das Kubernetes-Deployment einen Loadbalancer anfordert und dabei keine Adresse spezifiziert. Wer stattdessen eine feste IP-Adresse vergeben will, muss diese vorab über das Cloud-Native-API reservieren und im Kubernetes-Deployment angeben. Um nicht jedem einzelnen Service eine separate IP zuteilen zu müssen, empfiehlt sich einen Kubernetes-Ingres-Controller[4] einzusetzen. Dieser filtert Verbindungen auf Applikationsebene und leitet sie an die zuständigen Backend-Services weiter.

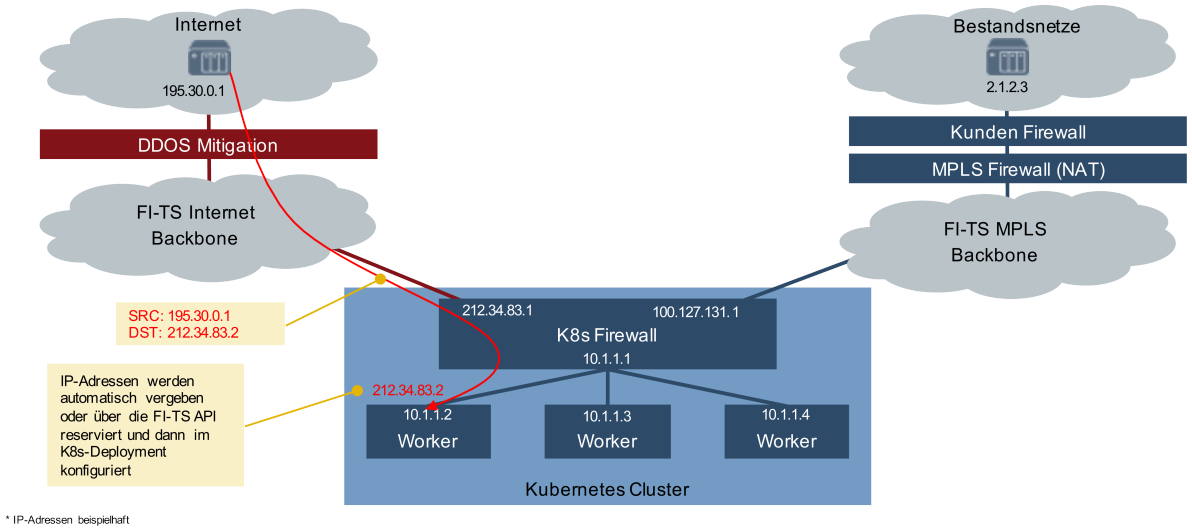

Abbildung 4: Eingehende Verbindungen aus dem Internet

[4] Auf Kundenwunsch stellt FI-TS ein Deployment-Template für einen professionell gewarteten *Nginx Ingress Controller* bereit.

4.2 Interne Verbindungen

Durch die (optionale) Anbindung an den MPLS-Backbone können Pods interne Systeme oder Hosts von Partner-Instituten leicht erreichen. Beim Lösungsdesign ist hierbei zu beachten, dass viele Bestandsnetze IPv4-Adressen verwenden, die auch im Internet vergeben sind. Um das Routing konfliktfrei zu gestalten, müssen solche lokalen Netzadressen durch ein weiteres, per MPLS-Firewall zwischengeschaltetes NATing aus dem Bereich 100.64.0.0/10 verborgen werden. Administrativ ist hierfür über einen vorgegebenen Genehmigungsprozess für Bestands-Firewallsysteme eine Freischaltung inklusive NATing zu beantragen.

Für abgehende Verbindungen zu internen Netzen führt die Kubernetes-Firewall ebenfalls ein NATing (Masquerading) durch und verwendet die beim Start zugewiesene RFC6598-Adresse.

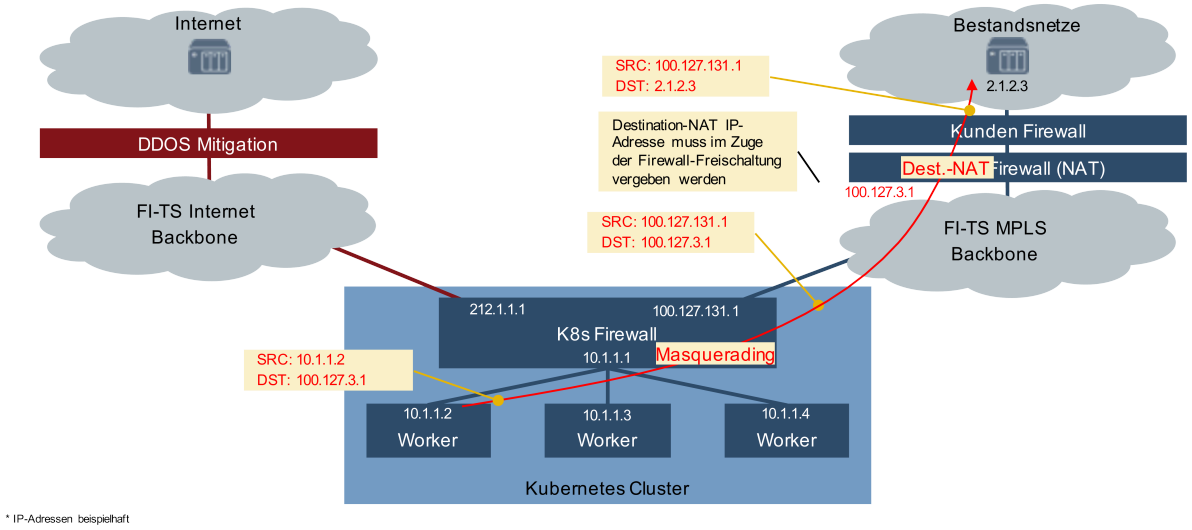

Abbildung 5: Abgehende Verbindungen zu internen Netzen

Analog dazu werden eingehende Verbindungen aus internen Netzen abgewickelt. Der Mechanismus zur Vergabe der Service-IPs entspricht dem des Internet-Bereichs, nur eben mit RFC6598-Adressen.

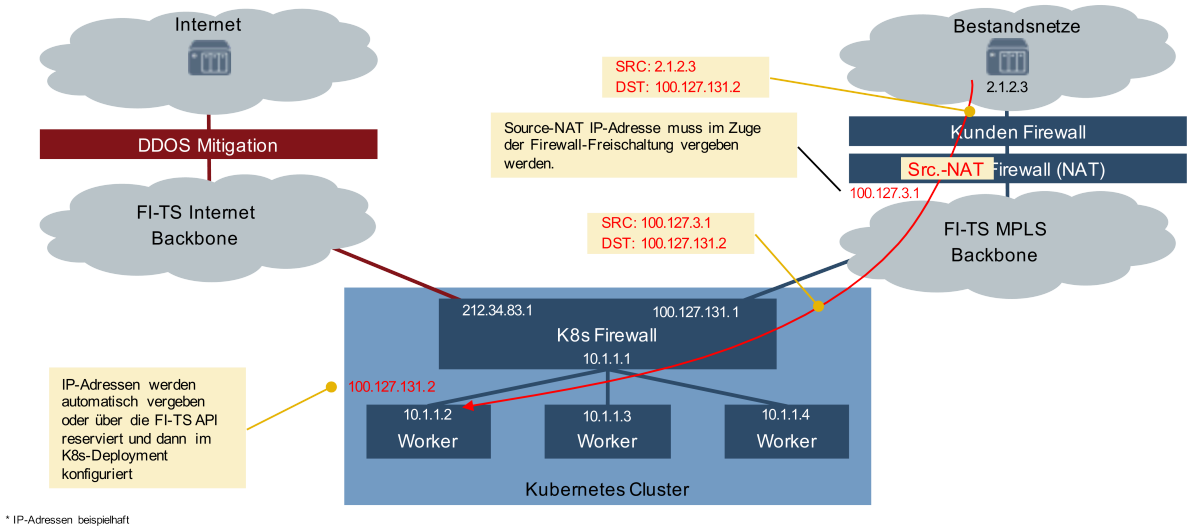

Abbildung 6: Eingehende Verbindungen aus internen Netzen

> **Warning**
>
> Das Cloud-Native-MPLS-Netz ist eine geteilte Ressource und eine gemeinsame Routing-Domain über die gesamte Plattform! Sollen Loadbalancer-Services eines Clusters aus anderen Clustern, auch anderer Mandanten, nicht erreichbar sein, muss dies der Kunde durch passende `loadBalancerSourceRanges` sicherstellen. Das genaue Vorgehen ist weiter oben im Management-Unterkapitel „Firewall-Freischaltungen“ beschrieben.

Der IP-Adressen vergeben und Netze verwalten will, greift auf das Cloud-Native-API beziehungsweise das Kommandozeilentool `cloudctl` zurück (siehe Cluster-Verwaltung).

> **Tip**
>
> Anmerkung: Das Masquerading der Kubernetes-Firewall verhindert es derzeit, an Bestandsfirewalls für einzelne (Quell-)Pods des Kubernetes-Clusters wirksam freizuschalten. Außerdem besteht eine Einschränkung auf maximal 65.000 Verbindungen zu einem Ziel im Bestandsnetz. FI-TS sieht NAT daher als Übergangslösung und beabsichtigt einen höherwertigen Gateway-Service bereitzustellen, sobald sie dessen Entwicklung abgeschlossen hat.

5 Produktoption Isolierte Kubernetes-Cluster

Der typische Einsatzzweck von Kubernetes ist es, Anwendungen dynamisch zu hosten, deren Anwender aus dem Internet zugreifen. Mindestens ebenso typisch ist es, dass Deployments ihre Containerimages ungehindert aus öffentlichen Repositories beziehen. Administratoren eines Kubernetes-Cluster haben zudem stets die Möglichkeit, Pods beliebige Wege in das Internet und aus dem Internet zu ermöglichen.

Einige Kunden von FCN möchten dagegen einige oder alle ihre Cluster gegen Zugriffe aus dem Internet und ins Internet nachhaltig absichern. Regulatorische Vorgaben können ein Motiv dafür bilden oder die Vorsorge gegen den Fall, dass Angreifer einzelne Anwendungen kompromittieren und anschließend Schadcode aus dem Internet nachladen. Ein in diesem Zusammenhang häufig geäußerter Kundenwunsch ist, dass Anwendungsadmins nicht mehr befähigt sein sollten, im Cluster Internetzugänge zu konfigurieren.

Um diese Anforderungen an die Absicherung von FCN-Kundenclustern gegen das Internet nachzukommen, hat FI-TS eine zweistufige Produktoption für FCN Kubernetes Hosting entwickelt, die Kunden befähigt sogenannte Isolierte Cluster anlegen. Der Schwerpunkt der Implementierung setzt eine logische Trennung in der Netzwerkarchitektur in Richtung Internet durch, die weder Anwendungen noch Anwendungsadministratoren überwinden können.

5.1 Clustervarianten in der FCN

Mit der Einführung von Isolierten Clustern muss der Kunde beim Anlegen des Cluster entscheiden, welche Form der Isolation er für seine Applikation benötigt. Es gibt seither drei Ausprägungen:

- `Internet Access Baseline`: Diese Variante ist der langjährige FCN-Standard und reglementiert weder den Internetzugriff noch die Art und Weise, wie ein Cluster Container beziehen kann.
- `Internet Access Restricted`: Bei einem mit dieser Option angelegten Cluster ist der Zugang zum Internet ein- und ausgehend anfänglich gesperrt. Der Clusteradmin darf diese Sperre granular rückgängig machen. Funktionell erforderliche Container-Images sind erreichbar, eigene Container-Images erfordern in der Praxis eine eigene Registry.
- `Internet Access Forbidden`: Mit dieser Option angelegte Cluster bleibt der Zugang zum Internet ein- und ausgehend verwehrt. Funktionell erforderliche Containerimages sind erreichbar, Kundenimages erfordern eine eigene Registry abseits des Internet.

5.2 Unbedingt erforderliche Containerimages

Um einen Kubernetes-Workernode zu erzeugen, ist es in der FCN notwendig, mehrere Containerimages zu pullen - die wichtigsten sind:

- Kubelet: Controller, der die Workload verwaltet
- CNI: Das Container Network Interface erstellt und verwaltet das Netzwerk für die Pods und Services
- CSI: Das Container Storage Interface erzeugt und verwaltet PVCs
- CoreDNS: Domainname-Service für Container
- MetalLB: Servicetype-`LoadBalancer`-Implementierung
- Node-Exporter und Metrics-Server: Monitoring des Workernode
- Mehrere Metal-Stack-Addons, zum Beispiel zum Anzeigen der Firewall- und Auditing-Events.

Für `Internet Access Restricted`- und `Forbidden`-Cluster haben die FCN-Entwickler folgende Architekturänderungen vorgenommen:

- Da öffentliche Registrys nicht mehr erreichbar sind, greifen Isolierte Cluster in eine private FCN-Containerregistry zu, die alle betriebsnotwendigen Containerimages gespiegelt vorhält - darüber hinaus aber keine weiteren. Der Zugriff auf diese Containerregistry ist nur aus den FCN-Kubernetes-Clustern möglich.
- Die Konfiguration des `containerd` auf jedem Workernode ist so modifiziert, dass er nur Containerimages aus der privaten Containerregistry pullen darf.

5.3 Containerimages für eigene Applikationen

Um die Containerimages eigener Applikationen in einem Cluster mit Isolation `Forbidden` deployen zu können, ist es unumgänglich, eine eigene Registry bereitzustellen. Die Kundenregistry muss innerhalb der für das Cluster erlaubten Netzwerke erreichbar sein. Außerdem muss die IP-Adresse der Registry im öffentlichen DNS auflösbar sein, und die CA des Workernode muss das TLS-Zertifikat der Registry als gültig erkennen.

In einem Cluster mit Isolation `Restricted` lassen sich Containerimages aus dem Internet theoretisch pullen, wenn der Clusteradmin eine CWNP anlegt (siehe unten), die auf eine externe oder öffentliche Registry zeigt. Allerdings laufen die meisten öffentlichen Containerregistrys innerhalb von Content-Delivery-Netzwerken, die einen Host unter hunderten IP-Adressen wechselnd exponieren. Dies verträgt sich in der Praxis mit einfachen CWNP nicht, zum Beispiel mit einer einzigen konfigurierten IP-Adresse für Docker.io. In der Konsequenz bedarf es für ein `Restricted`-Cluster auch einer dedizierten privaten Registry.

5.4 Netzwerkarchitektur

Für Isolierte Cluster pflegt FCN eine Liste von erlaubten Netzwerkbereichen. Sie enthält einige RFC-Netzwerke für den ein- und ausgehenden Verkehr sowie die FCN-Internetnetzwerke für den ausgehenden Verkehr. Kubernetes-Nutzer und -Administratoren sind nicht befähigt, diese Whitelist zu modifizieren.

Den Zugriff aus dem Cluster steuert eine eigene Custom Ressource `ClusterWideNetworkPolicy` (CWNP). FCN-Cluster bekommen bei der Anlage bereits einige CWNPs konfiguriert, die sich zwischen `Baseline`, `Forbidden` und `Restricted` unterscheiden.

`Baseline`-CWNPs:

Rule-Name	Ziel	Zweck
allow-to-http	0.0.0.0/0	Egress via HTTP
allow-to-https	0.0.0.0/0	Egress via HTTPS
allow-to-apiserver	IP des Kubernetes-API-Servers in der Controlplane	API-Server-Kommunikation für Kubelet und andere Controller
allow-to-dns	IP eines öffentlichen DNS-Servers	DNS-Auflösung der Kubernetes-Workernodes und -Containers
allow-to-ntp	IP eines öffentlichen NTP-Servers	Zeitsynchronisierung
allow-to-storage	Network des Containerstorage	Persistent Volumes per CNI-Treiber
allow-to-vpn	VPN-Endpoint-IP der Controlplane	Zum Verbinden von API-Server zum Kubelet für Containerlogs und Container-Exec

`Forbidden`- und `Restricted`-CWNPs:

Rule-Name	Ziel	Zweck
allow-to-apiserver	IP des Kubernetes-API-Servers in der Controlplane	API-Server-Kommunikation für Kubelet und andere Controller
allow-to-dns	IP eines öffentlichen DNS-Servers	DNS-Auflösung der Kubernetes-Workernodes und -Containers
allow-to-ntp	IP eines öffentlichen NTP-Servers	Zeitsynchronisierung
allow-to-registry	IP der privaten Registry	Pulling unbedingt nötiger Containerimages
allow-to-storage	Network des Containerstorage	Persistent Volumes per CNI-Treiber

allow-to-vpn	VPN-Endpoint-IP der Controlplane	Zum Verbinden von API-Server zum Kubelet für Containerlogs und Container-Exec

All diese CWNPs managt der (https://github.com/metal-stack/gardener-extension-provider-metal)[Gardener Extension Provider Metal], und dieser entfernt jede manuelle modifizierte Regel umgehend.

`Forbidden`-Cluster validieren CWNPs und Servicetype `LoadBalancer` gegen die Liste der erlaubten Netzwerke und setzen sie bei Verstoß nicht um. Im Unterschied dazu darf der Admin eines `Internet Access Restricted`-Clusters eigene CWNPs und Servicetype `LoadBalancer` anlegen, die außerhalb der erlaubten Netzwerke liegen.

Da DNS- und NTP-Requests von Isolierten Clustern nicht in Richtung Internet laufen dürfen, befriedigen extra dafür vorgehaltene FCN-Infrastrukturkomponenten solche Anfragen. Die Clusterbetriebssysteme setzen diese Vorgabe in geeigneter Weise durch.

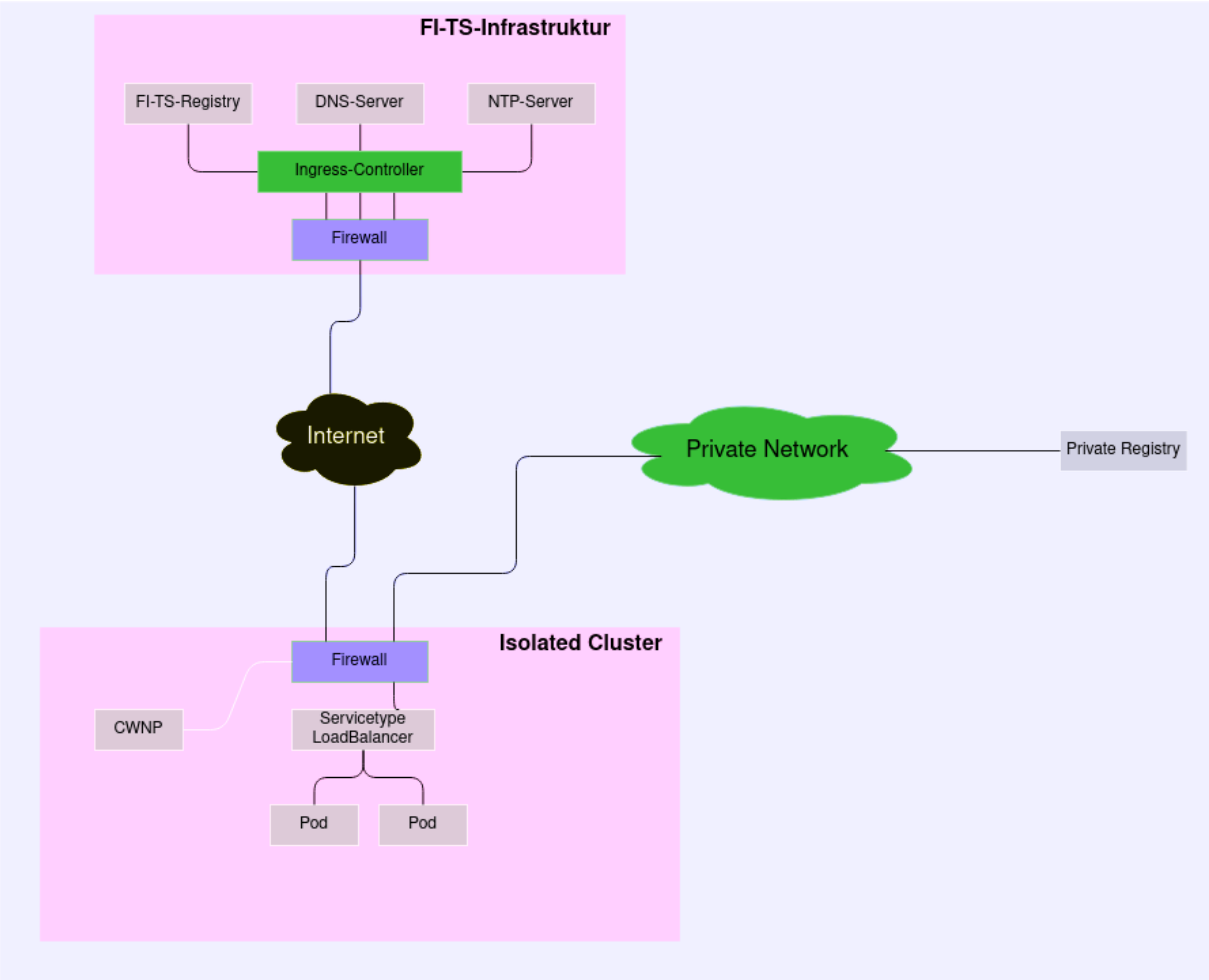

Abbildung 7: Clusterkommunikation

5.5 Praktische Handhabung

5.5.1 Internet Access Restricted

Die Kubernetes-Version 1.27 und folgende eröffnet FCN-Kunden die Möglichkeit, `Internet Access Restricted`-Cluster anzulegen. Es ist technisch ausgeschlossen, `Internet Access Baseline`- oder `Forbidden`-Cluster auf diesen Isolationslevel zu updaten. Die Syntax ist folgende:

```bash
> cloudctl cluster create --name <Name_des_Cluster> --project <Project-ID> --partition <Clusterstandort> --network-isolation restricted [...]

WARNING: You are going to create a cluster that has no default internet access with the following consequences:
- pulling images is only possible from private registries you provide, these registries must be resolvable from the public dns and must be secured with a trusted TLS certificate
- you can create cluster wide network policies to the outside world without restrictions
- pulling container images from registries requires to create a corresponding CWNP to these registries
- It is not possible to change this cluster back to "baseline" after creation
Are you sure? (y/n)
```

Kunden können nun die Liste der erlaubten Netzwerke mit `cloudctl cluster describe` abfragen:

```yaml
networkaccessrestrictions:
    fra-equ01:
        allowednetworks:
            egress:
                - 212.34.83.0/27
                - 100.64.0.0/10
                - 10.0.0.0/8
            ingress:
                - 100.64.0.0/10
        maskedregistries:
            - docker.io
            - quay.io
            - docker.lightbitslabs.com
            - eu.gcr.io
            - ghcr.io
            - registry.k8s.io
            - r.metal-stack.io
[...]
```

5.5.2 Internet Access Forbidden

Die Kubernetes-Version 1.27 und folgende eröffnet FCN-Kunden die Möglichkeit, `Internet Access Forbidden`-Cluster anzulegen. Es ist technisch ausgeschlossen, `Internet Access Baseline`- oder `Restricted`-Cluster auf diesen Isolationslevel zu updaten. Die Syntax ist folgende:

```bash
> cloudctl cluster create --name <Name_des_Cluster> --project <Project-ID> --partition <Clusterstandort> --network-isolation forbidden [...]

WARNING: You are going to create a cluster which has no internet access with the following consequences:
- pulling images is only possible from private registries you provide, these registries must be resolvable from the public dns, their IP must be located in one of the allowed networks (see cluster inputs), and must be secured with a trusted TLS certificate
- service type loadbalancer can only be created in networks which are specified in the allowed networks (see cluster inputs)
- cluster wide network policies can only be created in certain network ranges which are specified in the allowed networks (see cluster inputs)
- It is not possible to change this cluster back to "baseline" after creation
Are you sure? (y/n)
```

CWNPs und Servicetype `LoadBalancer` für ausgehenden Datenverkehr validiert der Cluster gegen die Liste der erlaubten Netzwerke setzt sie bei Verstoß nicht um:

```bash
> kubectl get clusterwidenetworkpolicies.metal-stack.io
NAME                  STATUS      MESSAGE
allow-to-apiserver    deployed
allow-to-dns          deployed
allow-to-ntp          deployed
allow-to-registry     deployed
allow-to-storage      deployed
allow-to-vpn          deployed
allow-to-google       ignored     ingress/egress does not match allowed networks
```

Zudem erzeugt Kubernetes einen Event:

```bash
> kubectl get events
5s          Warning   ForbiddenCIDR          clusterwidenetworkpolicy/allow-to-google    address:"8.8.8.8/32" is outside of the allowed network range:"10.0.0.0/8,100.64.0.0/10,212.34.83.0/27", ignoring
```

Eingehenden Netzwerkverkehr gestattet ein `Internet Access Forbidden`-Cluster ebenfalls nur von den erlaubten Netzwerken. Dazu ist es erforderlich, die `loadbalancerIP` mit einer vorher als `static` reservierten IP-Adresse aus einem erlaubten Netzwerke zu konfigurieren:

```yaml
apiVersion: v1
kind: Service
spec:
  type: LoadBalancer
  loadBalancerIP: 10.1.1.1 # ip from the internal network
```

Wer es versäumt, die `loadBalancerIP` anzugeben, bekommt einen Event ausgestellt, der beschreibt, warum dies so ist:

```bash
> kubectl get events
8s          Warning   AllocationFailed          service/example-service   Failed
to allocate IP for "default/example-service": no available IPs
3s          Warning   SyncLoadBalancerFailed    service/example-service   Error
syncing load balancer: failed to ensure load balancer: no default network for ip
acquisition specified, acquire an ip for your cluster's project and specify it
directly in "spec.loadBalancerIP"
```

Die External-IP im Service verharrt derweil auf Pending:

```bash
> kubectl get svc
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP    PORT(S)
AGE
example-service   LoadBalancer   10.244.75.171   <pending>      443:32179/TCP   4s
```

5.6 Service

Der Serviceumfang, den FI-TS zu den Isolierten Kubernetes-Clustern leistet, unterscheidet sich im Grundsatz nicht zu dem für `Internet Access Baseline` zu erbringenden. Kunden müssen gleichwohl beachten, dass das Personal von FI-TS ebenfalls nicht dazu in der Lage ist, die Clusterisolation zu deaktivieren. Dies ist nicht als Nachteil zu betrachten, sondern ist Folge des nachhaltigen Schutzes.

Sollte der Kunde jedoch Applikationen oder den Cluster selbst durch Fehlkonfiguration außer Betrieb setzen, die nur durch Assets aus dem Internet behebbar wäre, kann FI-TS dies nicht im Zuge des Incidentprozesesses gewährleisten. Der Ausweg besteht darin, dass der Kunde (s)eine frühere Konfiguration wiederherstellt. FI-TS behält sich in solchen Havariefällen vor, die SLA-Messung auszusetzen.

5.7 Kosten

Für die Verwendung der Cluster- und Netzwerk-Isolation entstehen erhöhte Betriebs- und Betreuungskosten, die Kunden als Produktoption zusätzlich je Cluster für die Dauer der Clusternutzung in Rechnung gestellt bekommen. Die bezogenen Verbräuche lassen sich mittels `cloudctl billing` abrufen.

```bash
> cloudctl billing product-option

TENANT  FROM                                TO
PROJECTID                                   PROJECTNAME              OPTION
CLUSTERID                                   CLUSTERNAME     LIFETIME
fits    2024-01-31T23:00:00.000Z
2024-02-05T09:58:35.508Z        cd4eac58-46a5-4a31-b59f-2ec207baa817    mwen
PRODUCT_OPTION_CLUSTER_ISOLATION        3f7348d4-89bf-4edb-b554-d4abc303364c
mwentest        5m
```

Wer für die Ausgabe das Yaml-Format wählt, bekommt Details zu Gesicht:

```yaml
---
clusterid: 3f7348d4-89bf-4edb-b554-d4abc303364c
clustername: mwentest
contract: "906000002"
debtorid: ""
id: PRODUCT_OPTION_CLUSTER_ISOLATION
lifetime: 300000000000
projectid: cd4eac58-46a5-4a31-b59f-2ec207baa817
projectname: mwen
tenant: fits
tenantname: FI-TS
annotations:
    - restricted # "Internet Access Forbidden"-Cluster, andernfalls "forbidden"
```

6 Servertypen

Für Kubernetes Cluster stehen an jedem Standort unterschiedliche Servertypen zur Auswahl, die als Kubernetes Worker Nodes verwendet werden können.

Das Spezifizieren des Servertyps für den Kubernetes Cluster ist optional und kann mittels `cloudctl cluster create --machinetype <type>` erfolgen. Da die Worker Nodes des Clusters automatisch skalieren können, ist eine explizite Angabe des Servertyps für den Anwender in der Regel nicht notwendig. Die Wahl eines bestimmten Servertyps ist nur dann notwendig, wenn die geplante Workload besondere Anforderungen an die Hardware stellt oder keine ausreichende Anzahl von Servern an einem Standort von einem speziellen Typ mehr zur Verfügung steht.

Die verfügbaren Servertypen können mit dem Befehl `cloudctl cluster inputs` abgefragt werden. Um das Ergebnis auf eine spezielle Partition zu filtern, kann auch `cloudctl cluster inputs --partition <id>` verwendet werden.

Servertyp	Empfehlung	vCPUs	Memory	stg-kkw701	fel-wps101	nbg-w8101
`c1-xlarge-x86`	Compute-Intensive	24	96G	X	X	X
`m1-large-x86`	Memory-Intensive	24	192G	X	X	X
`m1-xlarge-x86`	Memory-Intensive	24	384G	X	X	X
`n1-medium-x86`	Network-focussed	8	32G	X	X	X
`g1-medium-x86`	Graphic-Intensive	32	256G		X	
`s1-large-x86`	Storage-Intensive	8	196G	X	X	X
`s2-xlarge-x86`	Storage-Intensive	8	196G	X	X	X

Tabelle 1: Übersicht über die verfügbaren Servertypen.

Bestimmte Maschinentypen führen zu einem Kostenaufschlag, der zusätzlich zu den monatlichen Kosten für den Kubernetes Cluster berechnet wird. Dazu zählen die Maschinentypen `m1-xlarge-x86` und `g1-medium-x86`.

> **Tip**
>
> Mit dem Einsatz von Worker Groups (beschrieben in einem nachfolgenden Kapitel) ist es außerdem möglich Maschinengrößen zu mischen und die Last entsprechend des Maschinentyps aufzuteilen. Auch für Update-Szenarien können Worker Groups nützlich sein.

6.1 Server mit GPUs

GPU-Nodes können momentan über zwei Wege verwendet werden:

1. Spezifizieren des Maschinentyps „g1-medium-x86“ bei der Cluster-Anlage über `cloudctl cluster create --machinetype g1-medium-x86 --machineimage nvidia-550.0`
2. Einbindung einer zweiten Worker-Group bei einem existierenden Cluster. Hierbei ist zu beachten, dass zusätzliche Worker Groups momentan nur vom Provider eingerichtet werden können. Zum Schedulen von Workload auf Nodes einer weiteren Workergruppe können Node Selectors verwendet werden.

6.1.1 Einrichtung

Um GPU-Nodes im Kubernetes Cluster verwenden zu können, müssen nach der Provisionierung der Worker zusätzliche Installationsschritte durchgeführt werden. Zwar sind die passenden Grafikkarten-Treiber bereits vorinstalliert und der Containerd Shim bereits eingerichtet, jedoch ist es zusätzlich notwendig den nvidia-operator im Cluster zu deployen, um die GPUs in Kubernetes nutzbar zu machen.

Die Installation des Operators kann beispielweise über `helm` erfolgen. Wir haben die Funktionalität mit folgender Parametrisierung geprüft:

```bash
helm repo add nvidia https://helm.ngc.nvidia.com/nvidia
helm repo update

kubectl create ns gpu-operator
kubectl label --overwrite ns gpu-operator pod-security.kubernetes.io/
enforce=privileged

helm install --wait \
  --generate-name \
  --namespace gpu-operator \
  --create-namespace \
    nvidia/gpu-operator \
    --set driver.enabled=false \
    --set toolkit.enabled=true
```

Nach der Installation sollte `kubectl describe node` für den GPU-Worker eine entsprechende Kapazität an GPU-Kernen ausweisen:

```
...
Capacity:
  cpu:                64
  ephemeral-storage:  100205640Ki
  hugepages-1Gi:      0
  hugepages-2Mi:      0
  memory:             263802860Ki
  nvidia.com/gpu:     1
  pods:               510
...
```

> ⚠ **Warning**
>
> Bei der beschriebenen Installation kann nur ein einzelner Pod auf eine GPU zugreifen. Falls hingegen mehrere Pods gemeinsam auf dieselbe GPU zugreifen können sollen, muss der Operator entsprechend konfiguriert werden.
>
> Um dies zu erreichen gibt es mehrere Ansätze, die auf der offiziellen Webseite von NVIDIA nachgeschlagen werden können:
>
> - https://developer.nvidia.com/blog/improving-gpu-utilization-in-kubernetes
> - https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/gpu-operator-mig.html

- https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/gpu-sharing.html

7 Ratgeber und Best Practices

Seit der Einführung der Cloud-Native-Plattform hat die FI-TS viele praktische Erfahrungen im Umgang mit Kubernetes und den zugehörigen Services gesammelt. Dieses Kapitel beinhaltet wertvolle Ratgeber und Best Practices für die Verwendung der Plattform, die viel unnötigen Frust und Diskussion ersparen können. Anwender sollten sich mit den vorgestellten Punkten vertraut machen und sich bemühen diese umzusetzen, um Day-2 Betriebsaufwände zu minimieren und unnötige Service-Ausfälle zu vermeiden.

7.1 Release Notes und Changes

Die FI-TS entwickelt die Cloud-Native-Plattform kontinuierlich weiter, um auf schnellstem Weg Bug- und Vulnerability-Fixes, Verbesserung für Performance und Stabilität sowie neue Funktionalitäten und Software-Versionen für Anwender zur Verfügung zu stellen.

Releases werden während eines geplanten Change Windows eingespielt. Changes erfolgen in der Regel mindestens ein Mal pro Woche. Primär aktualisieren wir während eines Changes unsere eigene Plattform-Infrastruktur, wobei als Nebenprodukt Änderungen an den von der FI-TS gemanagten Komponenten aus den Clustern der Anwender erfolgen können[5]. Updates an diesen Komponenten werden im selbst einstellbaren Maintenance Timewindow eines Clusters (siehe Abschnitt 7.5) automatisiert ausgerollt. Hiervon ausgenommen sind Cluster, die nicht mit dem Purpose `production` versehen wurden. Diese werden bereits innerhalb des Change Windows vom Betriebsteam der FI-TS geupdated (siehe Abschnitt 7.2.3).

Neben Updates in den geplanten Change Windows, gibt es bestimmte Komponenten bei denen der Zeitpunkt der Durchführung vom Anwender selbst bestimmt wird. Dies kann aus der folgenden Tabelle entnommen werden:

Komponente	**Update-Zeitpunkt**
Cloud-Native-API	Im Change Window
Kubernetes Control Plane und gemanageten Komponenten	Im Cluster Maintenance Time Window[6]
Firewall-Controller Version	Im Change Window

[5] z.B. ETCD, Kubernetes API Server / Controller Manager / Scheduler oder im Cluster-befindliche Komponenten wie kubelet, CoreDNS, CNI, CSI, … (alles, was in einem neu erstellten Cluster zu finden ist).

[6] Hierbei ausgenommen sind Cluster, die nicht mit dem Purpose `production` versehen sind, siehe auch Abschnitt 7.2.3.

[7] Mit Hilfe des Auto-Update Features können Kubernetes Patch Versionen selbstständig ausgerollt werden, sobald neue Versionen verfügbar werden. Mehr Informationen dazu Abschnitt 7.5 dieses Dokuments. Des Weiteren kann nicht ausgeschlossen, dass zu einem gegebenen Zeitpunkt alte Kubernetes Versionen durch

Kubernetes Version	frei bestimmbar[7]
Betriebssystem-Images Version	frei bestimmbar[8]

Die FI-TS ist bemüht die Releases der Cloud-Native-Plattform so transparent wie möglich zu gestalten. Alle Änderungen können in Github Releases nachgelesen werden:

- https://github.com/fi-ts/releases
- https://github.com/metal-stack/releases

Des Weiteren werden Changes und neue Funktionalitäten über unsere Status-Seite https://status.fits.cloud/ kommuniziert.

Die Qualität jedes unserer Releases wird durch umfangreiche Integrationstests sichergestellt. Die Integrationstests decken alle Bereiche der Cloud Native Plattform ab und umfassen metal-stack, Cloud-Native-API, Gardener, Lightbits und S3. Außerdem wird für jedes Release sichergestellt, dass die bereitgestellten Cluster die CNCF Conformance Tests bestehen.

> **Tip**
>
> Updates sind essentiell, um die Plattform am Leben zu erhalten. Bitte unterstützen Sie uns bei diesem Vorhaben! DevOps und kontinuierliche Integrationen sind eng mit der Idee von Kubernetes verwoben und diese Werte sollen in der Cloud-Native-Plattform gelebt werden. Machen Sie sich der Notwendigkeit bewusst, dass die durch uns gemanageten Komponenten bei einem Update neugestartet werden müssen. Kubernetes ist so konstruiert, dass diese Änderungen unterbrechungsfrei durchgeführt werden können. Die folgenden Sektionen enthalten alle Informationen, die Sie benötigen, damit Updates keinerlei Einfluss auf die Workload ihrer Anwendung haben.

7.2 Überlegungen vor der Anlage eines neuen Clusters

Die von der Plattform bereitgestellten Cluster wachsen – wie üblich bei einer Cloud – dynamisch mit Ihrem Ressourcenbedarf. Dennoch sollten vor der Anlage eines Clusters bestimmte Rahmenbedingungen betrachtet werden, um die Wahrscheinlichkeit einer aufwendigen Migration einer Anwendungsarchitektur auf einen neuen Cluster zu minimieren.

ein Zwangsupdate aktualisiert werden müssen. In diesem Fall werden die Cluster-Owner im Vorfeld von der FI-TS informiert. Durch regelmäßige Durchführung von Cluster-Updates kann dieses Problem vermieden werden.

[8] Dies ist sowohl für Machine als auch Firewall Images möglich, aber standardmäßig deaktiviert. Auch hierzu gibt es mehr Informationen in Abschnitt 7.5.

Bei der Cloud Native Plattform wird grundsätzlich davon ausgegangen, dass Ressourcen in der Umgebung dynamisch allokiert werden. Worker Nodes haben demnach keinen besonders langen Lebenszyklus und sollten nicht als statische Ressourcen betrachtet werden. Sie sollen in einem Cluster regelmäßig ausgetauscht, erweitert oder reduziert werden können. Da es sich bei den bereitgestellten Worker Nodes um physische Server („bare metal server“) handelt, sollte man sich ebenfalls nicht auf das beständige Vorhandensein verlassen, da über lange Sicht mit Hardware-Ausfällen zu rechnen ist. Für die Aufrechterhaltung der Services im Cluster sorgt Kubernetes. Über unsere Infrastruktur können innerhalb weniger Minuten neue Server provisioniert werden, sodass keine Abstriche durch den Verzicht auf Virtualisierung in unserer Plattform bestehen.

Abhängig von der Art der Anwendung, können Workloads CPU- oder Memory-intensiv sein. Für diese Zwecke stellen wir an jedem Standort unterschiedlich ausgestattete physische Server für ihre Cluster zur Verfügung. Die verfügbaren Servertypen werden in Abschnitt 6 beschrieben. Bitte beachten Sie, dass ggf. nicht alle Servertypen in allen Partition verfügbar sind.

Nutzen Sie für kritische Anwendungen mehrere Cluster mit unterschiedlichen Serverstandorten, um Ausfallszenarien von Rechenzentren bestmöglich abzudecken. Wie Sie Services über mehrere Standorte hinweg verfügbar machen können, wird später in diesem Dokument beschrieben.

7.2.1 Maximale Clustergröße and Anzahl von Pods pro Worker Node

Standardmäßig können Cluster 16 Worker Nodes umfassen auf denen jeweils 510 Pods gescheduled werden können.

Bei ressourcenintensiven Deployments, kann es ratsam sein die Maximalanzahl der Pods herunterzusetzen, um die maximale Clustergröße zu erhöhen. Diese Konfiguration kann bei der Anlage eines Clusters über das Flag `--max-pods-per-node` in `cloudctl cluster create` konfiguriert werden.

Den Zusammenhang zwischen maximaler Podanzahl pro Worker Node und Anzahl der möglichen Cluster Worker Nodes können folgender Tabelle entnommen werden:

Max Pods	Max Nodes
> 510	nicht ratsam
<= 510	16
<= 254	32
<= 126	64

< 100	nicht ratsam

Planen Sie eine Vollauslastung der Clustergröße mit ressourcenintensiven Anwendungen, wäre beispielweise eine Clusteranlage mit `cloudctl cluster create --max-pods-per-node 110` sinnvoll.

Es ist generell ratsam nicht die Grenze der maximalen Worker Nodes für ein Cluster zu erreichen. Hierzu sollen zwei wichtige Flags erwähnt werden, die beim Erstellen und Updaten eines Clusters zur Verfügung stehen:

- `--maxsurge`: Die maximale Anzahl der Worker die bei einem Rollen der Worker Nodes (z.B. durch ein Update) gleichzeitig zum Cluster hinzugefügt werden können.
- `--maxunavailable`: Die maximale Anzahl der Worker die bei einem Rollen der Worker Nodes (z.B. durch ein Update) gleichzeitig zum Cluster gleichzeitig entfernt werden können.

Wir empfehlen die Standardeinstellung von `--maxsurge 1` und `--maxunavailable 0`, um ein unterbrechungsfreies Rollen von Worker Nodes zu ermöglichen. Grund dafür ist, dass ein Cluster an der Grenze der maximalen Worker Nodes kein Worker Rolling mehr durchführen kann außer indem `--maxunavailable` erhöht wird. In diesem Fall muss der Cluster die Last des entfernten Worker Nodes auffangen, was zu Beeinträchtigungen der Stabilität des Clusters führen kann.

7.2.2 Einsatz von Cloud Storage

Sofern bei der Erstellung des Clusters das `--external-networks` nicht gesetzt wurde, wird jeder Cluster standardmäßig im partitionseigenen Storage Netzwerk hinzugefügt aus dem über die Storage Classes des Typs `csi.lightbitslabs.com` persistenter Storage über das NVMe/TCP Protokoll bezogen werden kann.

Die lokalen Festplatten von Worker Nodes und die damit verbunde Storage Class `csi-lvm` sollte in der Regel nicht für langfristigen Storage verwendet werden. Der Ausfall eines Worker Nodes würde zu Datenverlust führen und das Rolling Update behindern.

7.2.3 Cluster Purposes

Jedes Cluster kann vom Anwender über ein Feld mit dem Namen „Purpose“ für unterschiedliche Verwendungs-Zwecke gekennzeichnet werden. Die Konfiguration geschieht über cloudctl mit dem Flag `--purpose` entweder direkt bei der Cluster-Erstellung oder beim Cluster-Update Kommando. Der Wert kann zu jedem Zeitpunkt wieder vom Anwender verändert werden. Die zur Verfügung stehenden Werte für „Purpose“ sollen hier kurz erläutert werden:

- `evaluation`: Ein mit diesem Zweck gekennzeichneter Cluster impliziert, dass in diesem Cluster keine produktive Workload betrieben wird und dieser zu Evaluationsgründen angelegt wurde. Cluster mit diesem Purpose qualifizieren sich früher für neue Plattform-Features und werden innerhalb unserer Change Windows unmittelbar geupdated. Dadurch können Anwender und das Betriebsteam der FI-TS mögliche Probleme, die durch ein Update entstehen könnten, in diesen Clustern bereits feststellen noch bevor diese auf Cluster mit produktiver Workload angwendet werden. Allerdings werden diese Cluster nicht von der SLA gedeckt.
- `development`: Für diesen Purpose gelten die gleichen Eigenschaften wie für `evaluation`.
- `production`: Ein mit diesem Zweck gekennzeichneter Cluster impliziert, dass in diesem Cluster produktive Workload betrieben wird. Dieser Cluster wird bei Plattform-Updates nicht unmittelbar während des Change Windows geupdated, sondern erst im vom Nutzer einstellbaren Maintenance Timewindow (siehe Abschnitt 7.5). Neue Plattform-Features werden hier vom Betriebsteam der FI-TS tendenziell später ausgerollt, nachdem diese eine Weile auf Clustern mit nicht productionsrelevanter Workload erprobt wurden.
- `infrastructure`: Dieser Zweck steht nur Clustern der FI-TS zur Verfügung und dient zur Kennzeichnung von Clustern, die Basis-Funktionalität der Plattform bereistellen. Er kann von Anwendern nicht verwendet werden.

> **Tip**
>
> Sie helfen dem Betriebsteam der FI-TS, wenn Sie Cluster mit nicht produktionsrelevanter Workload über ihren Cluster-Purpose entsprechend kennzeichnen. Je größer die Gesamtmenge dieser Cluster ist, desto kleiner wird die Eintrittswahrscheinlichkeit, dass beim Ausrollen eines Updates für produktive Cluster ein unvorhersehbares Problem auftritt.

7.2.4 Hochverfügbare Kubernetes Control Plane

Cluster werden fortan standardmäßig mit dem HA Control Plane Feature Gate provisioniert. Das heißt, dass die von der Cloud Native verwalteten Komponenten der Kubernetes Control Plane in einer hochverfügbaren Konfiguration bereitgestellt werden.

Dies bietet dem Anwender einige Vorteile: Der ETCD des Clusters wird anstatt als Standalone-Installation in einer Cluster-Konfiguration bereitgestellt, wodurch Mechanismen wie automatische Skalierung dieser Komponente zu keiner Nichtverfügbarkeit der Kubernetes API führt. Außerdem wirken sich Plattform-Updates, die ggf. Updates an den Komponenten der Control Plane vornehmen, nicht auf die Verfügbarkeit der Kubernetes API aus.

Alte Cluster, die vor der Einführung von HA Control Planes erstellt wurden, sollten mit Hilfe des Kommandos `cloudctl cluster update <id> --high-availability-control-plane` auf die hochverfügbare Kubernetes Control Plane migriert werden.

> **Tip**
>
> Bei der Umstellung auf HA Control Planes ist zu beachten, dass das für Produktion **nicht** empfohlene Cluster-Forwarding Backend des Kubernetes API Server Auditing inkompatibel mit dem HA Control Plane Feature Gate ist. Das Thema Auditing wird in Abschnitt 2.2.4 genauer beschrieben und empfiehlt Splunk als Audit-Backend für Cluster mit produktiver Workload einzusetzen.

7.2.5 Cluster-Autoscaling

Da die Kosten abhängig von den angeforderten Ressourcen im Cluster berechnet werden, werden Cluster standardmäßig mit einem Node-Autoscaler provisioniert. Dieser erkennt automatisch einen erhöhten Ressourcenbedarf im Cluster und fügt dementsprechend neue Woker Nodes hinzu. Verringert sich der Ressourcenbedarf, so kann der Autoscaler auch Worker Nodes aus dem Cluster entfernen.

Durch dieses Feature können hohe Lastspitzen aufgefangen und horizontal skaliert werden ohne dass über lange Prozesse neue Ressourcen zur Verfügung gestellt werden müssen. Allerdings erfordert die entstehende Dynamik auch die Berücksichtigung von speziellen Konfigurationen in Kubernetes. Hierzu gehören:

- Die Einrichtung von `PodDisruptionBudgets`, die bei der Entfernung eines Nodes durch den Cluster-Autoscaler dafür sorgen, dass ein neuer Pod auf einem existierenden Node untergebracht werden kann bevor der alte gelöscht wird
- Keine Verwendung von `PersistentVolumes`, die eine Affinität auf einen einzigen Worker-Node haben (sprich: Verwendung der Storage Classes des Type `csi.lightbitslabs.com` anstatt `csi-lvm`)

> **Tip**
>
> Damit der Autoscaler richtig funktioniert, ist es sehr wichtig, dass die Pod Requests adäquat gesetzt werden. Lesen Sie hierzu das Kapitel Resource Management for Pods and Containers in der offiziellen Kubernetes Dokumentation. Monitoren Sie die Verbräuche der Pods mit der Einrichtung eines Cluster Monitorings und der von uns bereitgestellten Metrics API.

7.2.6 kubectl cp und kubectl exec

`kubectl cp` und `kubectl exec` sind hilfreiche Kommandos, um ein Kubernetes Cluster zu debuggen. Die Kommandos sollten allerdings nicht im produktiven Einsatz verwendet werden. Grund hierfür ist, dass diese Kommandos über den Kubernetes API Server geproxied werden und dann über das Kubelet in die Container Engine weitergeleitet werden. Die Komponenten dieser Chain unterliegen einer SLA Zeit und stehen bei Updates ggf. kurzzeitig nicht zur Verfügung. Die Kubernetes API Server sind selbst-skalierend deployed, und können auch ohne Ankündigung gerollt werden, um den Ressourcenbedarf anzupassen. Auch in diesem Fall bricht die Verbindung der Kommandos ab. Der Traffic fließt außerdem nicht über die Cluster-Firewall ihres Clusters und die Verbindung ist auf Grund der Komplexität verhältnismäßig langsam und fragil.

Anstatt `kubectl cp` zu verwenden, können Sie für kleine Dateien Kubernetes Ressourcen wie `ConfigMaps` oder `Secrets` verwenden und diese über CI ins Cluster deployen. Für große Dateien verwenden Sie den S3-kompatiblen Object Storage. Der Traffic fließt dann über die stabile Netzwerkverbindung ihrer Cluster Firewall, welche höchstens von einem durch Sie vorgenommenen Firewall Update unterbrochen werden kann.

Statt `kubectl exec` verwenden Sie ggf. Kubernetes Jobs für One-Time Executions oder fest installierte Sidecars, die Sie ebenfalls über CI definieren können.

7.2.7 IP Adressen

Im Folgenden finden sich einige Hinweise zum Bezug und dem Lebenszyklus von IP-Adressen der Cloud Native Plattform.

7.2.7.1 Flüchtige IP Adressen

Bestimmte IP Adressen sind flüchtig (engl. *ephemeral*) und werden bei der Löschung eines Clusters automatisch wieder freigegeben. Es besteht keine Garantie, dass diese IP Adressen wieder für dasselbe Projekt allokiert werden können. Daher sollten flüchtige IP Adressen vermutlich nicht für Network Policies oder Freischaltungen verwendet werden, die dauerhaft geplant sind.

Eine flüchtige IP Adresse kann mit `cloudctl ip static` in eine statische IP Adresse umgewandelt werden. Mit `cloudctl ip list` kann man eine Übersicht über alle verwendeten IPs erhalten.

7.2.7.1.1 Adresse des Kubernetes API Servers

Bei Updates oder Migrationen von Control Planes kann sich die IP Adresses einer Kubernetes API verändern. Verlassen kann man sich nur auf den DNS-Namen der API, der beispielsweise in der Cluster Kubeconfig zu finden ist.

> ⚠ **Warning**
>
> Bitte verwenden Sie aus diesem Grund die IP Adresse des API Servers in keinen wichtigen Regeln von Firewalls, etc.

7.2.7.2 Adresse der Cluster Firewall

Bei einem Firewall-Update kann sich die Internet IP einer Firewall verändern. Standardmäßig ist diese IP Adresse die Source Adresse für ausgehenden Traffic aus ihrem Cluster, weil ausgehender Traffic von der Firewall mit SNAT verschleiert wird.

Wenn Sie auf eine fixe Source IP-Adresse für ihr Cluster angewiesen sind, konfigurieren Sie eine statische Egress-IP für das Cluster über `cloudctl cluster update --egress ...`. Diese kann den Lebenszyklus eines Clusters überleben und Probleme wie Port Exhaustion können durch die Verwendung weiterer Egress-IPs verhindert werden.

7.2.7.3 Service Type Load Balancer Adressen

Adressen, die durch die Erstellung eines Services vom Typ Load Balancer im Kubernetes Cluster bezogen werden (insofern das `LoadBalancerIP` Feld nicht explizit gesetzt wird), sind standardmäßig flüchtig und werden beim Löschen eines Clusters wieder freigegeben.

Wenn Service-Adressen über den Cluster Lebenszyklus hinaus erhalten bleiben sollen, müssen diese über `cloudctl ip static` in eine statische Adresse umgewandelt werden.

> 💡 **Tip**
>
> Bitte beachten Sie, dass Sie selbst für das Aufräumen von statischen IP Adressen zuständig sind und auch für nicht verwendete IP Adresse Kosten anfallen.

7.2.7.4 Adresse der Cloud Native API

Verwenden Sie für die Konfiguration von `cloudctl` nur die URL https://api.fits.cloud/cloud.

7.3 Cluster- und Node-Updates mit Zero Downtime

Wir möchten dazu motivieren, Cluster häufig zu updaten (im Idealfall mit Auto-Updates), damit sichergestellt wird, dass Security Fixes auf schnellstem Wege ausgerollt werden. In dieser Sektion werden die Maßnahmen beschrieben, die angewendet werden können, damit diese Prozedur planbar und ohne Ausfälle durchgeführt werden kann.

7.3.1 Reduktion der Kopplung zu gemanageten Komponenten

Um einen stabilen Betrieb der Anwendung in ihrem Cluster zu gewährleisten, ist es notwendig die SLA-Zeiten der von uns gemangeten Komponenten zu berücksichtigen, da wir diese Zeiten nutzen, um Updates zur Verfügung zu stellen. Als Konsequenz sollte man sich als Anwender nicht zu stark an diese Komponenten koppeln. Insbesondere sehen wir vor:

- dass die Kubernetes API Server bei einem Update oder Change zeitweilig nicht zur Verfügung stehen[9],
- dass Komponenten auf dem Worker Node wie Kubelet oder Container Runtime Engine bei einem Update oder Change neugestartet werden können,
- dass von uns verwaltete Komponenten innerhalb des Kubernetes Clusters (z.B. CoreDNS, MetalLB…) neugestartet werden können.[10]

7.3.2 Simulation von Updateszenarien

Man kann sich auf die erwartbaren Szenarien von Updates einstellen, indem man diese in einer Test-Stage simuliert und dabei die Verfügbarkeit der Anwendung von außen überwacht. Sollte das Durchspielen der folgenden Szenarien keine Auswirkungen auf die Service-Erreichbarkeit haben, ist ein reguläres Plattform-Update unterbrechungsfrei.

7.3.2.1 kubelet und Container Runtime

Um wie bei einem Plattform-Update den Neustart des Kubelets und der Container Runtime auszulösen, kann folgende Node Annotationen verwendet werden:

```bash
kubectl annotate node <your-node> worker.gardener.cloud/restart-systemd-services=kubelet.service,containerd.service
```

Beim Setzen dieser Annotation führt eine Anwendung auf dem Worker Node einen Restart des kubelets und der Container Runtime durch. Beim Neustart der Kubelet Komponente

[9] Bei Clustern, die mit dem HA Control Plane Feature Gate konfiguriert sind (siehe Abschnitt 7.2.4), reduziert sich die potentielle Nichtverfügbarkeit des Kubernetes API Servers bei Plattform-Updates erheblich. Wir empfehlen ältere Cluster zu HA Control Planes zu migrieren, falls dies noch nicht geschehen sein sollte.

[10] Diese Komponenten werden immer durch ein Rolling Update aktualisiert, sodass durch Spreading von Pods auf mehrerer Worker Nodes das Update unterbrechungsfrei stattfinden kann.

markiert der kube-controller-manager den Node kurzzeitig mit dem Status `NodeNotReady`, was dazu führt, dass der Node für einen kurzen Zeitpunkt aus den `Services` entfernt wird, wodurch die Paketweiterleitung zu den auf diesen Nodes gescheduleten Pods unterbrochen wird. Dieser Zustand dauert in der Regel weniger als eine Sekunde, führt aber dazu, dass stehende Verbindungen von Komponenten auf dem Node sowohl innerhalb des Clusters als auch von außen unterbrochen werden.

> **Tip**
>
> Auf dieses Verhalten von Kubernetes können wir keinen Einfluss nehmen. Es ist daher wichtig, dass verwendete Software, die stehende Verbindungen zu einem Pod aufbaut, einen selbstständigen Wiederaufbau einer Verbindung implementiert.

7.3.2.2 MetalLB

Die Netzwerkinfrastruktur der Cloud Native Plattform basiert auf dem BGP Protokoll. Für eine Internet IP Adresse, die über einen Service Type Load Balancer bezogen wird, übernimmt die von uns verwaltete Komponente MetalLB das Route Announcement für die bezogene IP-Adresse über ein Peering mit dem auf Node befindlichen FRR Daemon.

Beim Rollen des Metallb Speaker `DaemonSets` im `metallb-system` Namespace wird für den Zeitraum des Neustart des Pods das Route Announcement kurzzeitig zurückgezogen.

> **Tip**
>
> Das Rollen des Speaker `DeamonSets` geschieht nicht häufig. Dennoch ist es ratsam, dass eine hochverfügbare, über einen Service Type Load Balancer exponierte Anwendung über zwei Nodes verteilt wird. Das sogenannte *Spreading* über die Nodes kann mit Hilfe von Kubernetes über Pod „anti-affinity“ realisiert werden. Hierzu empfehlen wir das Kapitel Assigning Pods to Nodes in der Kubernetes Dokumentation.

Das Rolling Update des `DeamonSets` kann mit dem folgenden Befehl ausgelöst werden:

```bash
kubectl rollout restart ds -n metallb-system speaker
```

7.4 Nutzung von Worker Groups

Der Einsatz von Worker Groups kann z.B. für folgende Anwendungsfälle nützlich sein:

- Separation von Workload innerhalb des Clusters auf dedizierte Maschinen (z.B. dedizierten Nodes für CI-Runner bei Gitlab-Deployments)
- Kubernetes Updates bei der Nodes manuell migriert werden müssen

Ein Cluster wird standardmäßig mit einer einzigen Worker Group `group-0` angelegt. Innerhalb dieser Worker Group sind alle Nodes identisch, d.h. sie verwenden denselben Servertyp und das konfigurierte Betriebssystem Image. Nodes desselben Pools können über das gemeinsame Node Label `worker.gardener.cloud/pool` selektiert werden.

Werte für den Cluster-Autoscaler wie `--minsize`, `--maxsize`, `--maxsurge`, `--maxunavailable` beziehen sich auf eine einzelne Worker Group und können für jede Worker Group individuell konfiguriert werden. Die aktuelle Worker-Gruppen-Konfiguration kann jederzeit über den Befehl `cloudctl cluster describe <id> -o yaml --no-machines` abgerufen werden.

Nach der Anlage des Clusters können weitere Worker Groups hinzugefügt oder entfernt werden. Derzeit muss allerdings immer mindestens eine Worker Group vorhanden sein. Das Hinzufügen einer Worker Group über `cloudctl` geschieht beispielsweise über `cloudctl cluster update <id> --workergroup ci-runners --minsize 1--maxsize 2 --machineimage debian-12.0 --machinetype m1-xlarge-x86`.

Da Kubernetes die Verwendung von Kubelets mit einer bis zu zwei Minor-Versionen kleineren API Version unterstützt (näheres dazu in der Kubernetes Dokumentation unter Version Skew Policy), können Worker Groups mit `--workerversion` unabhängig von der Version des Kube API Servers auf eine ggf. niederigere Kubernetes gepinnt werden. Für Kubernetes Updates kann dies vorteilhaft sein:

- Statt die einzelne Worker Group zwei Mal rollen zu müssen, braucht man durch Version Pinning nur ein einziges Mal die Worker Group rollen
- Es besteht kein Zeitdruck bei der Verlagerung der Workload auf die neuen Nodes

> **Tip**
>
> Da zusätzliche Worker Groups potentiell gegen das bedarfsgesteuerte Abrechnungsmodell arbeitet, fallen für Worker Groups zusätzliche Kosten an.

7.4.1 Upgrade von Kubernetes mit Hilfe von Worker Groups

Für das Beispiel nehmen wir einen Cluster mit Kubernetes Version `1.25.11` mit einer einzelnen Worker Group (`group-0`) an, dass auf Kubernetes Version `1.27.8` aktualisiert werden soll.

1. Falls die bestehende Worker Group noch auf keine spezifische Version gepinnt wurde, wird die Gruppe nun auf die aktuelle Kubernetes Version gepinnt. Das Vorgehen verhindert, dass die Worker Nodes gerollt werden, wenn die Version des Kubenertes API Servers erhöht wird: `cloudctl cluster update <id> --workergroup group-0 --workerversion 1.25.11`
2. Jetzt kann das Cluster zwei Mal geupdated werden. Selbstverständlich sollte nach jedem Update geprüft werden, dass die Anwendung im Cluster noch einwandfrei funktioniert. Im Vorfeld muss sichergestellt werden, dass keine veralteten Kubernetes API Ressourcen mehr verwendet werden. Das Kommando für das Update lautet: `cloudctl cluster update <id> --version 1.26.13`, anschließend `cloudctl cluster update <id> --version 1.27.8`.
3. Nun wird eine neue Worker Group hinzugefügt, z.B. durch `cloudctl cluster update <id> --workergroup group-1 --machineimage debian-12.0 --machinetype c1-xlarge-x86 --workerversion 1.22.13 --minsize 1 --maxsize 5`
4. Die Workload der alten Worker Gruppe kann nun auf die neue Worker Gruppe verschoben werden. Dies geschieht üblicherweise mit den Kommandos `kubectl cordon` und `kubectl drain`.
5. Nachdem die alten Worker Nodes frei geworden sind, kann die alte Worker Gruppe im Anschluss an die erfolgreich Migration entfernt werden: `cloudctl cluster update <id> --workergroup group-0 --remove-workergroup`

7.5 Auto-Updates

Für folgende Komponenten bieten wird eine Auto-Update Funktionalität an:

- Kubernetes Patch Version
- Worker Node Betriebssystem Image
- Firewall Betriebssystem Image

Sobald auf der Finance Cloud Native Plattform innerhalb eines Changes neue Versionen der vorgestellten Komponenten zur Verfügung gestellt werden, können diese automatisch aktualisiert werden ohne dass der Update-Prozess manuell angestoßen werden muss.

Auto-Updates finden im konfigurierten Wartungszeitfenster (*Maintenance Time Window*) des Clusters statt. Standardmäßig ist dies von 23 Uhr bis 0 Uhr voreingestellt. Das Wartungszeitfenster kann auch über die Flags `--maintenance-begin` und `--maintenance-end` modifiziert werden. Eine Cluster Maintenance kann außerdem über `cloudctl cluster reconcile --operation maintain` erzwungen werden.

Die Funktionalität kann einzeln über die `cloudctl cluster update --autoupdate-*` Befehle pro Cluster konfiguriert werden.

Wir empfehlen Auto-Updates uneingeschränkt für Test Umgebungen, um diese stets auf dem neuesten Stand zu halten. Für Produktionsumgebungen kann es geschickter sein, die Updates manuell anzustoßen, was allerdings mit einem höheren Wartungsaufwand verbunden ist.

7.5.1 Kubernetes Patch Version

Ein Update der Kubernetes Patch Version (z.B. `1.27.7` auf `1.27.8`, nicht Minor-Version wie z.B. `1.26.11` auf `1.27.8`) findet in-place statt. D.h. dass die Worker Nodes des Clusters nicht ausgetauscht werden. Stattdessen wird nur die neue Version des API Server ausgerollt und auf den Worker Nodes die neue Kubelet Version installiert (mit allen beschriebenen Implikationen aus Abschnitt 7.3.2.1).

7.5.2 Machine Image Updates

Bei einer Aktualisierung des Worker Node Betriebssystem Images werden die Worker Nodes des Clusters entsprechend der Worker Group Konfiguration gerollt. Für diese Art von Update müssen die in Abschnitt 7.3 beschriebenen Vorbereitungen getroffen werden, damit es nicht zu Service-Unterbrechungen kommt.

7.5.3 Firewall Image Updates

Bei einer Aktualisierung des Firewall Betriebssystem Images wird zunächst eine neue Firewall parallel zur existierenden Firewall bereitgestellt. Sobald die neue Firewall sich konfiguriert hat, übernimmt diese das Traffic Routing und die alte Firewall wird entfernt. Bei der Übernahme des Traffics von der alten auf die neue Firewall werden stehende Verbindungen unterbrochen. Auch kann der Paketfluss für einige Sekunden unterbrochen werden.

7.6 Geo-redundante Software Deployments

Dieser Abschnitt ist noch nicht fertiggestellt.

7.7 Q&A

Frage: Ich habe einen Controller, der davon abhängig ist, dass die Kubernetes API immer zu 100% zur Verfügung steht. Wie kann man damit umgehen?

Antwort: Der Controller muss die Abwesenheit der Kubernetes API tolerieren können. Erhöhen Sie ggf. Timeouts auf die maximale SLA-Zeit. Ist dies nicht möglich, kann man durch das Deployment eines eigenen ETCDs und Kubernetes API Server im eigenen Cluster die Kopplung auflösen und Resistenz gegenüber Plattform-Updates werden.

Frage: Kann ich einen Überblick über die Kubernetes Control Plane erhalten?

Jeder Kubernetes Cluster verfügt über eine separate, read-only Grafana Instanz, die ein Anwender einsehen darf. Die URL zum Dashboard und die dazugehörigen Anmeldedaten können über `cloudctl cluster monitoring-secret` abgerufen werden.

Das Dashboard liefert einen Überblick über die von uns gemanageten Control Plane Komponenten und ist nicht modifizierbar. In Problemfällen kann es sehr nützlich sein, beispielsweise falls durch eigens ausgerollte Controller oder Webhooks die API Server des eigenen Kubernetes Clusters destabilisiert werden. Kontrollieren Sie in Fällen, in denen der API Server außerhalb eines Change Windows nicht wie gewohnt reagiert die Dashboards für den Cluster.

7.8 Weitere Hilfestellung

Gerne stellen wir Ihnen ein Beratungskontingent zur Verfügung in dem wir Ihnen helfen können die vorgestellten Best Practices für ihr Anwendungsdeployment umzusetzen.

8 Änderungshistorie

2023-02-24	‚Kleinert, Jan‘	Fix in der Kapitel-Hierarchie
2023-02-24	‚Kleinert, Jan‘	Weiterer Fix in Kapitel-Hierarchie
2023-02-24	‚Reiger, Michael‘	Resolve „MPLS service sourceRanges“
2023-04-25	‚Christian Brunner‘	Aufnahme der Dokumentenklassifizierung in die Fußzeile
2023-05-12	‚Wennrich, Markus‘	add instructions how to replace defect nodes/firewalls
2023-05-30	‚Stefan Majer‘	Fix some typos
2023-06-06	‚Fensterer, Markus‘	DMZ-Setup
2023-07-31	‚Schwerthelm, Gerrit‘	Add storage network for n2-tm1601.
2023-08-30	‚Markus Wennrich‘	PSP are deprecated
2023-08-30	‚Markus Wennrich‘	add –default-pod-security-standard
2023-08-30	‚Kleinert, Jan‘	Stilverbesserung
2023-09-12	‚Anastasiia Orlova‘	Dokumentation wurde aktualisiert und mit Lightbits erweitert.
2024-03-20	‚Majer, Stefan‘	Isolated Clusters
2024-04-04	‚Kleinert, Jan‘	Update 05-Isolierte-Cluster.md (2 Typos).
2024-04-16	‚Peter, Mike‘	Merge branch ‚master‘ into ‚update-docs‘
2024-06-11	‚Majer, Stefan‘	Migrate to typst
2024-06-12	‚Schwerthelm, Gerrit‘	Sektion über die Verwendung von GPU Worker Nodes.
2024-06-12	‚Stefan Majer‘	Keine Zeilennummern in Quelltexten
2024-06-13	‚Brunner, Christian‘	Erstellung eines deutschsprachigen Berechtigungskonzepts
2024-06-14	‚Schwerthelm, Gerrit‘	Manage dependency versions in a single place.
2024-06-14	‚Wennrich, Markus‘	add hint to kube-apiserver-acl cloudctl commands
2024-06-14	‚Schwerthelm, Gerrit‘	Fix some typos.
2024-06-17	‚Schwerthelm, Gerrit‘	Add section for best practices.
2024-06-19	‚Schwerthelm, Gerrit‘	Korrektur Struktur Best Practices.
2024-07-05	‚Gerrit‘	Korrektur „physikalisch“ zu „physisch“.
2024-08-14	‚Knabel, Valentin‘	feat(audit): upgrade to latest cloudctl version 32
2024-08-20	‚Gerrit‘	Hervorhebung weshalb Audit-Log Cluster-Forwarding nicht für Produktion verwendet werden sollte.
2024-08-28	‚Gerrit‘	Anpassung Worker-Node Annotation seit Gardener Node Agent.
2024-09-03	‚Majer, Stefan‘	Merge branch ‚audit-logs-why-not-cluster-forwarding‘ into ‚master‘

2024-10-02	‚Pilz, Alexander‘	Berichtigung S3 Kommando
2025-02-11	‚Schwerthelm, Gerrit‘	Anpassungen zur neuen Rollout Strategie von Plattform Updates.
2025-02-21	‚Majer, Stefan‘	Typst v0.13
2025-02-27	‚Markus Wennrich‘	Update GPU operator installation instructions to enable toolkit
2025-02-27	‚Markus Wennrich‘	keep spacing
2025-03-10	‚Schwerthelm, Gerrit‘	Erklärung von HA Control Planes.
2025-03-10	‚Schwerthelm, Gerrit‘	Update zu lokalem NVMe Storage von Workernodes.