# Implementierung moderner IT-Anforderungen in Traffic Management System-Produkten

## Implementing modern IT requirements in Traffic Management System products

Kari Haapala

**Der Unterhalt eines Verkehrsmanagementsystems (Traffic Management System – TMS) oder eines zentralen Betriebsleitsystems (Centralized Traffic Control – CTC) unterscheidet sich nicht von dem anderer IT-Systeme. Die Systemeigentümer setzen unter anderem hohe Verfügbarkeit, einfache Upgrades, Betrieb ohne Ausfallzeiten, Selbstheilung, hohe Datensicherheit und flexible Skalierung voraus. Dieser Beitrag beschreibt, wie eben diese Standard-Plattformfunktionen sogar in eine Legacy-TMS-Anwendung implementiert werden können und welche Vorteile die modernen IT-Technologien für die Anwendungen bringen.**

### 1 Herausforderungen beim Management eines TMS

TMS oder CTC sind ein wesentlicher Bestandteil des Betriebs im Infrastrukturmanagement. Auch wenn solche Systeme normalerweise nicht direkt etwas mit der Sicherheit im Schienenverkehr zu tun haben, ist ihre Verfügbarkeit in geschäftlicher Hinsicht absolut unerlässlich. Sowohl geplante als auch ungeplante Ausfallzeiten haben Umsatzeinbußen, Reputationsschäden und zusätzliche Kosten zur Folge.

Bei jedem IT-System besteht inhärent die Gefahr von Cyberattacken. Früher wurden die Systeme abgesichert, indem man sie so weit wie möglich von jeglicher externer Kommunikation isolierte, um die Angriffspunkte zu verringern. In letzter Zeit besteht jedoch zunehmend der Bedarf, Öffnungen zuzulassen und einen ein- oder zweiseitigen externen Datentransfer zu ermöglichen, um beispielsweise die vorbeugende Wartung, Ferndiagnosen, die Erfassung von Geschäftskennzahlen oder andere Datenanalysen zu erleichtern. Daher kann man Sicherheitsbedrohungen nicht mehr nach dem Walled-Garden-Prinzip abwehren, sondern nur noch durch ein aktives Sicherheitsmanagement und regelmäßige Systemupdates.

In der allgemeinen IT-Branche ist dies bereits seit langem der Fall. Es wurden Praktiken und Technologien für ein aktives Systemmanagement geschaffen, um diese Herausforderungen zu bewältigen. Im Systemmanagement haben sich insbesondere folgende Grundsatzanforderungen herauskristallisiert:

- Null Ausfallzeiten: Jedes System bzw. jede Anwendung muss jederzeit aktualisiert werden können, ohne Ausfallzeiten in der Geschäftsanwendung zu verursachen. Ebenso müssen eventuelle Rollbacks ohne Ausfallzeit durchgeführt werden können, falls das Update nicht erfolgreich war. Diese Anforderung umfasst auch Hardware (HW)-Updates, die speziell im Schienenverkehr bei sehr langen Lebenszyklen wichtig sind.
- Hohe Verfügbarkeit: Die Systemarchitektur muss so aufgebaut sein, dass der Ausfall einer einzelnen Komponente die Verfügbarkeit der Geschäftsanwendung nicht beeinträchtigt.

**Maintaining a Traffic Management System (TMS) or a Centralised Traffic Control (CTC) system is no different to maintaining any other IT system. The system owners require high availability, easy upgrades, zero downtime, self-healing, a high level of information security and flexible scaling etc. This article describes how these very standard platform features can be implemented even for a legacy TMS application and what benefits modern IT technologies can bring to applications.**

### 1 The challenges in managing a TMS

TMS or CTC are a vital part of any infra manager's operations. Even though these systems are not normally directly related to railway safety, their availability is absolutely essential from a commercial point of view. Both planned and unplanned downtime causes losses of revenue, reputational damage and extra costs.

Any IT system is inherently vulnerable to cybersecurity threats. A traditional method of securing these systems has been to isolate them as far as possible from any external communications in order to reduce the attack footprint. However, it has recently become increasingly necessary to open up and allow one or two-way external information transfer in order to facilitate preventive maintenance, remote diagnostics, business Key performance Indicators (KPI) collection or other data analytics. Therefore, the security threats can no longer be managed with the walled garden approach, but via active security management policies and regular system updates.

This has already long been the case in the regular IT industry and active system management practices and technologies have been created to tackle these challenges. In particular, the following requirements have been set as clear system management principles:

- zero downtime: any system or application must be able to be updated at any time without any downtime for the business application. Similarly, roll-backs must also be able to be performed without any downtime, if the update has not been successful. This requirement also covers any hardware updates that are especially important for railways with very long lifecycles.
- high availability: the system must been designed so that a failure in a single component does not impact the availability of the business application.
- scalability: traffic management capacity requirements are usually well-known, but there may still be situations where some services may temporarily need an unknown amount of processing capacity (e.g. some system post-processing for an-

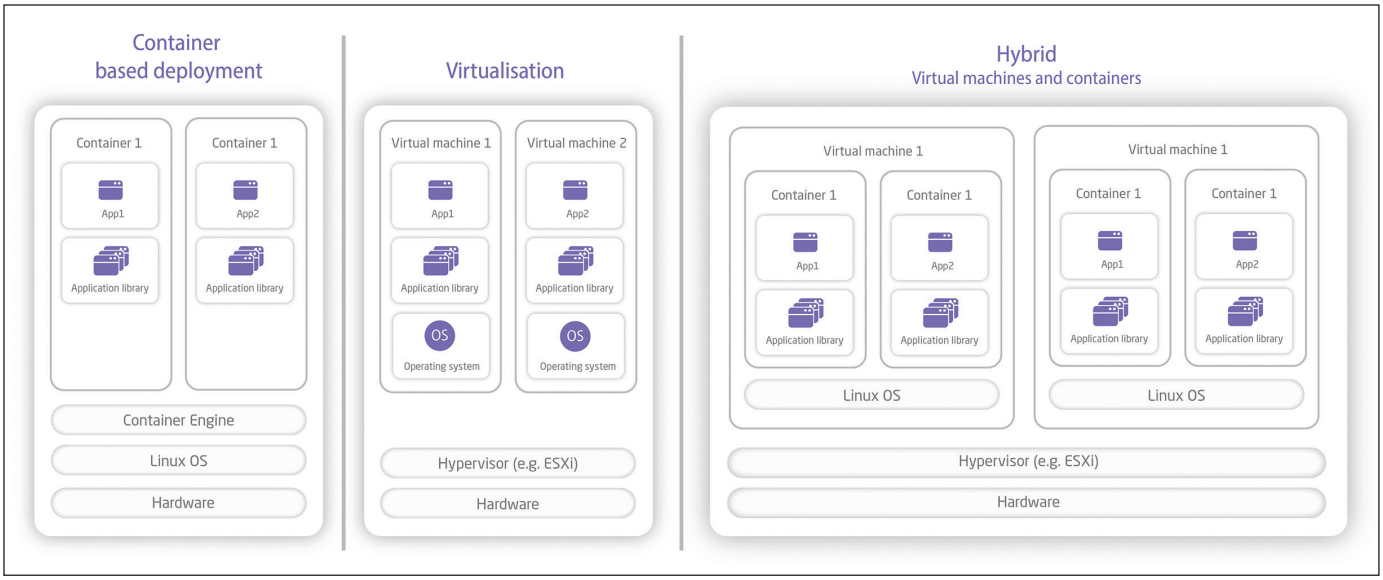


**Bild 1: Virtualisierung und Container**
Fig. 1: Virtualisation and containers

Quelle (alle Bilder) / *Source (all fig.)*: eigene Darstellung / *own illustartion*

- Skalierbarkeit: Normalerweise sind die Kapazitätsanforderungen im Verkehrsmanagement genau bekannt. Es kann jedoch immer Situationen geben, in denen bestimmte Dienste vorübergehend eine Verarbeitungskapazität unbekannten Umfangs erfordern (z. B. System-Postprocessing für Analysezwecke). In diesen Fällen muss das System automatisch hoch- und herunterskalieren können, um die Verarbeitungsanforderungen zu bewältigen.
- Betriebskontinuität: Das Verkehrsmanagement muss in jeder Situation funktionsfähig bleiben, auch im Fall lokaler Katastrophen wie Brände, Überschwemmungen und Erdbeben. Möglich wird dies durch eine sorgfältige Planung der Betriebskontinuität unter Einbeziehung georedundanter Architektur und Disaster-Recovery-Prozesse.
- Selbstheilung: Jedes IT-System hat irgendwelche Probleme. Software kann Programmierfehler haben, HW-Server können ausfallen. Die Fähigkeit zur Selbstheilung bedeutet, dass das System zum einen Diagnosen zur Problemerkennung vornehmen kann und zum anderen ein automatisches Verfahren vorhanden ist, um das System bei Auftreten von Problemen wieder betriebsfähig zu machen.
- Cybersicherheit: Nicht zuletzt muss das IT-System relevante Cybersicherheitsstandards erfüllen können. Aus operativer Sicht bedeutet dies, dass das System die technischen Voraussetzungen haben muss, regelmäßige Updates zu ermöglichen, um die erkannten Schwachstellen zu patchen. Normalerweise wird dies mit der geeigneten DevOps-Praxis samt einem Continuous-Delivery-Prozess erreicht.

## 2 Moderne IT-Technologien und wie sie beim IT-Management helfen

### 2.1 Virtualisierungstechnologie

Die Virtualisierung ist eine bekannte und bewährte Technologie, die im Lauf der letzten 20 Jahre in Standard-Serverumgebungen Einzug gehalten hat. Die Virtualisierung ermöglicht es, die HW-Kapazität auf mehrere virtuelle Maschinen aufzuteilen. Diese können jeweils unabhängig voneinander verschiedene Anwendungen in verschiedenen Betriebssystemen ausführen. Die größten

alytics). In those cases, the system must be able to be automatically scaled up and down in order to facilitate the processing requirements.
- business continuity: the Traffic Management must remain operational under any situation, including local catastrophes, such as fires, flooding and earthquakes. This can be achieved by means of a careful business continuity design, including geo-redundant architecture and disaster recovery procedures.
- self-healing: every IT system has its problems. Software can have bugs or hardware servers can break down. Self-healing capability means the system's ability to undertake diagnostics in order to detect these issues and to implement an automatic procedure to bring the system back to an operational state, if any such issues occur.
- cybersecurity: last but not least, the IT system must be able to comply with the relevant cybersecurity standards. From an operating standpoint, this means that the system must have the technical capability to allow regular updates to patch any detected vulnerabilities. This is usually achieved using the appropriate DevOps practice with a Continuous Delivery process.

## 2 Modern IT technologies and how they can help in IT management

### 2.1 Virtualisation technology

Virtualisation is an old and stable technology that has been brought to standard server environments over the last 20 years. Virtualisation allows the hardware capacity to be split into several virtual machines that can independently run different applications under different operating systems. The biggest benefits from virtualisation technology relate to reducing the number of necessary physical servers. These benefits are detailed below:
Easier server management: virtualisation makes it very easy to adapt to changing server requirements, for example when deploying new applications. For example, creating new virtual servers or modifying parameters (CPU, RAM, HD) for existing ones can be undertaken simply using the management console.

Vorteile der Virtualisierungstechnologie hängen damit zusammen, dass die Anzahl der benötigten physischen Server reduziert wird. Unter den Vorteilen sind zu nennen:
Einfachere Serververwaltung: Durch die Virtualisierung ist es möglich, Anpassungen an die wechselnden Serveranforderungen vorzunehmen, etwa wenn neue Anwendungen eingeführt werden. So können beispielsweise neue virtuelle Server erstellt oder die Parameter (CPU, RAM, HD) der bestehenden Server einfach über die Managementkonsole modifiziert werden.
Niedrigere IT-Kosten: Die Partitionierung des physischen Servers in mehrere virtuelle Maschinen bedeutet geringere Ausgaben für diese Server.
Platzeinsparung: Es wird eine geringere Anzahl physischer Server benötigt, was den Platzbedarf verringert.
Energiekosten: Niedrigere Energiekosten sowohl für die Stromversorgung als auch für die Kühlung des Servers – weniger physische Server reduzieren den Stromverbrauch.
Hohe Verfügbarkeit, Disaster Recovery und Georedundanz: Die Virtualisierung bietet mehrere Optionen, was die Steuerung der Verfügbarkeitsanforderungen anbelangt. Auch wenn die Anwendungsworkloads eigentlich nicht auf Verfügbarkeit ausgelegt sind, kann die Virtualisierungsplattform diese Funktionen auf Infrastrukturebene unterstützen. Die Virtualisierungsplattform kann beispielsweise eine synchronisierte Kopie der virtuellen Maschine an einem anderen Standort erstellen. Diese kann übernehmen, falls der primäre Standort ausfällt. Die Virtualisierungsplattform kann die ausgefallene virtuelle Maschine auch automatisch neu starten, um den Dienst schnell wiederherzustellen.
Betriebskontinuität: Das Einrichten eines geeigneten Backup-Plans wird durch die Virtualisierung vereinfacht. Ein zentrales Backup-System erfordert normalerweise einen Backup-Agenten auf jedem Server, wenn es keine Virtualisierungslösung gibt. Dank der Virtualisierung können die Backups durch die Plattform selbst von außerhalb der virtuellen Maschine erstellt werden, ohne die Applikationslogik zu beeinträchtigen.
Die Virtualisierung ist heute bereits eine ausgereifte und gut erforschte Technologie, die in Unternehmen jeder Größe breite Anwendung findet.
Bei TMS können die meisten Teile des Systems virtualisiert werden: Backend-Server mit Virtualisierung im üblichen Sinn und

Lower IT costs: partitioning the physical server into several virtual machines means less money spent on those servers.
Space saving: a smaller number of physical servers is needed, so the footprint is reduced.
Energy costs: lower energy costs for both powering and cooling the server, because fewer physical servers consume less power.
High availability, disaster recovery and geo redundancy: virtualisation offers several options for managing availability requirements. Even if the application workloads have not originally been designed with availability in mind, the virtualisation platform can provide infrastructure level support for these features. For example, the virtualisation platform can create a synchronised copy of a virtual machine on a different site to be used if the primary site fails. The virtualisation platform can also automatically restart the failed virtual machine to recover the service quickly.
Business continuity: setting up a proper backup policy is simplified with virtualisation. A centralised backup system typically requires a backup agent to be deployed on each server if no virtualisation has been used. Virtualisation enables the backups to be taken from outside the virtual machine by the platform itself without interfering in the application logic.
Virtualisation is currently already a mature and well-understood technology and is widely used by different sized organisations.
TMS can utilise virtualisation for most parts of the system: backend servers with regular virtualisation and front-end workstations using desktop virtualisation technologies. Virtualisation can also solve or help with the other IT challenges shown above for TMS.

### 2.2 Container technology

In the last ten years, the emergence of container technology has revolutionised the IT industry and has become the model for how to deploy applications. A discussion of the details of this technology falls beyond the scope of this article, but those who are interested can find a good source of information here [4], for example.
The biggest benefits offered by this technology relate to how the applications can be managed on the hardware: how the deployment, installation and upgrades are performed and how to achieve hardware independency for better cost control.

| | Legacy-System | Containerbasierte Architektur |
|---|---|---|
| Art der Anwendungsbereitstellung | Unterschiedlich: manuelles Kopieren, Skripterstellung, maßgeschneiderte Tools usw. Bereitstellung erfordert in der Regel viel Zeit. | Kontinuierliche Anwendungsbereitstellung. Bereitstellung erfolgt in Sekunden / Minuten, nicht in Stunden oder Tagen. |
| Upgrade-Methode | Unterschiedlich: manuelles Kopieren, Skripterstellung. Während des Upgrades ist die Anwendung normalerweise nicht verfügbar. | Automatische, fortlaufende Updates der Anwendungscontainer ohne Ausfallzeiten. |
| Logistik der Anwendungsbereitstellung | Manuelles Kopieren und Verteilen der Applikation auf Systeme vor Ort. | Automatische Kopie aus dem zentralen Repository. |
| Komplexität der Bereitstellung | Komplex / Stunden | Sehr einfach / Minuten |
| HW- und OS-Abhängigkeiten | Viele. | Keine. Container laufen nativ in verschiedenen Linux-Umgebungen. Windows-Umgebungen unterstützen Container mit WSL-Technologie. |

**Tab.1: Vorteile der Containertechnologie**

| | Legacy system | Container based architecture |
|---|---|---|
| Application deployment method | It varies: manual copying, scripting, bespoke tools etc. Deployment typically requires a long time. | Consistent deployment method. Deployment completed within seconds / minutes, not hours or days. |
| Upgrade method | It varies: manual copying, scripting. Upgrades typically require application downtime. | Automatic rolling updates of the application containers with no downtime. |
| Application deployment logistics | Manual copying and distribution of the application to on-site systems. | Automatic copying from a central repository. |
| Deployment complexity | Complex / hours | Very simple / minutes |
| Hardware and OS Dependency | Many | None. Containers can run natively in different Linux environments. Windows environments support containers with WSL technology. |

Tab. 1: The benefits of using container technology

Front-End-Workstations, die Virtualisierungstechnologien für den Desktop verwenden. Die Virtualisierung kann in einem TMS auch die anderen genannten IT-Herausforderungen lösen oder zur Lösung beitragen.

### 2.2 Containertechnologie

In den letzten zehn Jahren hat das Aufkommen der Containertechnologie die IT-Branche revolutioniert und wurde zum Vorbild dafür, wie Applikationen bereitgestellt werden. Eine detaillierte Diskussion dieser Technologie würde den Rahmen dieses Beitrags sprengen, aber Interessierte finden zum Beispiel unter [4] eine gute Informationsquelle.

Die größten Vorteile dieser Technologie liegen darin, wie die Anwendungen auf der HW verwaltet werden können, wie Bereitstellung, Installation und Upgrades durchgeführt werden und wie im Interesse einer besseren Kostenkontrolle HW-Unabhängigkeit erreicht wird.

Diese Vorteile sind in Tab. 1 aufgeführt:

Container können mit Virtualisierungslösungen kombiniert werden, um bei der Ausführung der Anwendung eine noch höhere Flexibilität zu erreichen. Dieses Setup macht es möglich, die Zuständigkeit für die Wartung der verschiedenen Systembereiche effizient aufzuteilen. HW: HW-Anbieter, Virtualisierungsumgebung: IT-Abteilung, Anwendung: CTC / TMS-Anbieter.

Obwohl es auf den ersten Blick eine gewisse Überschneidung zwischen Virtualisierung und Containern gibt, überwiegt die flexible-

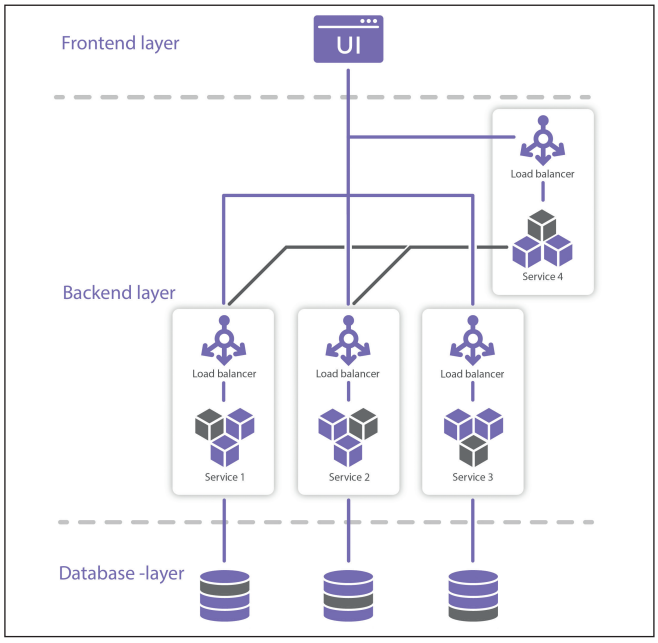


**Bild 2: Beispiel für eine Microservice-Architektur; jede Microservice-Instanz läuft in einem eigenen Container unter einem Load Balancer.**
Fig. 2: An example of a microservice architecture; each microservice instance is run in its own container, under a load balancer.



re Systemverwaltung den Aufwand beim Betrieb beider Technologien. Zu Beginn können die Legacy-Anwendungen problemlos virtualisierte Plattformen nutzen, doch sobald die Anwendungen nach und nach modernisiert werden, ist die Nutzung eines Hybridmodells aus Containern und Virtualisierung besser.

### 2.3 Container-Orchestrierung

Während die Containertechnologie die Möglichkeit bietet, eine Anwendung unabhängig von HW- und OS (Operating System, Betriebssystem) -Einschränkungen auszuführen, ermöglicht es die Container-Orchestrierung, einen Server-Cluster als eine Einheit zu behandeln. Seit 2014 erfreut sich die von Google entwickelte Orchestrierungstechnologie Kubernetes großer Beliebtheit als Laufzeitumgebung für Anwendungen. Obwohl Kubernetes sowohl lokale als auch Cloud-Bereitstellungsmodelle unterstützt, wird es oft als „das Betriebssystem für die Cloud" bezeichnet.
Bei der Orchestrierung werden die Anwendungen nicht mehr auf einem einzelnen, spezifisch dafür vorgesehenen Server bzw. einer virtuellen Maschine bereitgestellt, sondern auf dem Cluster insgesamt. Der Orchestrator entscheidet aus der Liste der berechtigten Serverressourcen, wo die Applikation ausgeführt werden soll.
Wenn die Applikation oder der Server ausfällt, stellt der Orchestrator den Dienst um. Das involviert möglicherweise die Verlagerung auf einen anderen Server, wodurch die Selbstheilungsfunktion gewährleistet ist. In der Regel verfügt ein Orchestrator auch über Funktionen zum laufenden Aktualisieren der Anwendung, sodass die Auswirkungen auf den Service auf Kundenseite minimal sind. Im besten Fall können die Updates ganz ohne Servicebeeinträchtigung durchgeführt werden. Diese Funktion wird oft als Zero-Downtime-Update bezeichnet.

## 3 Die Implementierung orchestrierter Container im Verkehrsmanagement

Es gibt viele Wege, die IT-Herausforderungen eines TMS zu lösen, doch wie zuvor aufgezeigt bietet die Nutzung einer modernen, orchestrierten Containerarchitektur die größten IT-Vorteile. Tatsächlich nutzten im Jahr 2020 bereits 83 % der Unternehmen Kubernetes in der Produktion (CNCF-Umfrage, [2]). Die meisten Anwendungsanbieter greifen bei sämtlichen neuen Applikationen bevorzugt darauf zurück. Eignet es sich aber auch für Legacy-Anwendungen? Müssen bei der Auswahl der richtigen Technologie irgendwelche Anwendungsbeschränkungen berücksichtigt werden? Ja, müssen sie, und ja, in vielen Fällen kann auch ein TMS-Altsystem die orchestrierten Container unterstützen.

### 3.1 Schritt für Schritt zu orchestrierten Containern im Verkehrsmanagement

Das Erstellen einer containerisierten Verkehrsmanagement-Applikation ist eine relativ alltägliche Aufgabe im IT-Architekturdesign, wenn die Anwendung von Grund auf neu erstellt wird. Container passen sehr gut zu einer modernen Microservice-Architektur mit Frontend (UI) auf Web-Technologie-Basis. Die Microservice-Architektur ist eine Variante der bekannten serviceorientierten Architektur (SOA). Dabei wird die Applikation als Sammlung intern kohärenter, aber lose gekoppelter Dienste organisiert. Jeder Dienst hat seine eigene, eng umrissene Aufgabe und eine klar definierte Service-API (Application Programming Interface, Programmierschnittstelle), die von anderen Diensten verwendet wird. Beispiele für solche Microservices sind z. B. Dienste wie Zugstrecken, Fahrpläne oder Gleisbelegungen. Die API-Technologie ermöglicht es,

These benefits are summarised in tab. 1:
The use of containers can be combined with virtualisation to achieve even more flexibility for running the application. This kind of setup allows the effective distribution of responsibility for maintaining the different areas of the system: the hardware – the hardware vendor, the virtualisation environment – the IT Department, the application – the CTC / TMS vendor.
Although there seems to be some overlap between virtualisation and the containers, the system management flexibility outweighs the burden of running both. Initially, the legacy applications could easily utilise a virtualised platform, but they will be able to utilise the hybrid model with containers and virtualisation better once the applications have been modernised.

### 2.3 Container orchestration

While container technology provides the ability to run the application independently of any hardware and operating system (OS) constraints, container orchestration technology will enable a cluster of servers to be treated as a single entity. The Google-originated orchestration technology known as Kubernetes has become immensely popular for providing a runtime environment for applications since 2014. Even though it supports both on-premises and cloud deployment models, it is often referred as "the operating system for the cloud"".
With orchestration, the applications are no longer deployed on a single dedicated server or virtual machine, but on a cluster as a whole. The orchestrator decides where to run the application from the list of eligible server resources.
If the application or server fails, the orchestrator will reconcile the service, which may involve relocating it to a different server, whereby the self-healing capability can be guaranteed. Orchestrators typically also have features for updating the application in a rolling manner, which means there is minimal impact on customer service when this is being performed. In the best case scenario, updates can be performed without any service impact at all. This feature is often called a zero-downtime update policy.

## 3 Implementing orchestrated containers for traffic management

Solving IT challenges for TMS can be achieved in many different ways, but utilising the modern orchestrated container-based architecture will provide the biggest IT benefits as discussed above. In fact, 83% of companies were already using Kubernetes in production in 2020 (a CFCF survey, [2]). Most application vendors recommend this for any new application. However, can it also be applied to legacy applications? Are there any application constraints to be considered when picking the right technology? There definitely are and yes, legacy TMS can also support orchestrated containers in many cases.

### 3.1 Steps towards orchestrated containers for Traffic Management

Creating a containerised Traffic Management application is a fairly normal IT architecture design task, when building an application from scratch. Containers fit into modern microservice-based architectures very well with frontends (UI) built with web-technology. Microservice architecture is a variant of the well-known Service-Oriented Architecture (SOA), and it arranges the application as a collection of internally coherent, but loosely coupled services. Each service has its own narrow task and well-defined service API (Application Programming Inter-

den Dienst auf mehreren Instanzen und auf jedem Server innerhalb der Laufzeitplattform auszuführen.
Da jeder Microservice in einem eigenen Container bereitgestellt wird, entscheidet die Orchestrierungsplattform über den Service-Discovery-Mechanismus, wo der Container ausgeführt wird und wie der Service erreicht wird.
Da die Containertechnologie mehrere offensichtliche Vorteile bietet, sollte auch bei einer älteren Verkehrsmanagement-Anwendung (Legacy TMS) die Portierung auf eine Containerplattform erwogen werden. Allerdings eignet sich die Containertechnologie nicht in jedem Fall, etwa bei einer Desktop-Anwendung mit grafischer Benutzeroberfläche [1]. Außerdem muss man bei Anwendungen, die auf einem Windows-API basieren und daher das Windows-Betriebssystem erfordern, eine spezifische Windows-Containertechnologie verwenden, die sich von der Linux-Containertechnologie unterscheidet. In den meisten anderen Fällen ist jedoch mit einem kompetenten Team bei einer Legacy-Anwendung eine Lift-and-Shift-Containerisierung möglich. Erforderliche Kernkompetenzen des Teams:

- Genaue Kenntnisse der Anwendungsarchitektur
- Kompetenz in der Bereitstellungsarchitektur
- Kompetenz in der Container- und Orchestrierungstechnologie
- DevOps-Design-Kompetenz
- Testkompetenz

Die Portierung der Anwendung in einen Container umfasst die folgenden übergeordneten Schritte (die Details gehen über den Rahmen dieses Beitrags hinaus):

- Jeder separat laufende Anwendungsprozess sollte im Hinblick auf seine Laufzeitanforderungen analysiert werden: Bibliotheken, Kommunikationsmatrix, Ressourcenverbrauch (CPU, RAM, HD), Bedarf an dynamischen Daten, Sonstiges. Container können Ressourcenobergrenzen für diese Grenzbereiche festlegen, was einen zusätzlichen Sicherheitsfaktor im Systemdesign darstellt.
- Containerisierung. Die Anwendung wird containerisiert, indem die Containerbeschreibung (z. B. Dockerfile) erstellt wird. Sie gibt über das ausgewählte Container-Basis-Image die Schritte vor, wie die Applikation bereitgestellt wird.

face) that is utilised by the other services. Examples of such microservices could include the train route service, the timetable service or the track possession service. API technology allows the service to run on multiple instances and on any server within the runtime platform.
Since each microservice is deployed within its own container, the orchestration platform will decide where to run the container and how to reach the service by means of the service discovery mechanism.
Since there are several obvious benefits to container technology, even porting a legacy Traffic Management application to a container platform should be considered. However, container technology cannot be used in every case, e.g. for a desktop GUI application [1]. In addition, an application that specifically requires the Windows OS, because it is based on Windows API, will require the use of specific Windows container technology which is different to Linux containers. However, lift-and-shift containerisation is possible for most other cases of legacy applications, provided you have a competent team. The core team skills include:

- a detailed knowledge of application architecture
- deployment architecture skills
- container and orchestration technology skills
- DevOps design skills
- testing skills

Porting the application to a container involves the following high-level steps (the details fall beyond the scope of this document):

- each separately running application process should be analysed in order to understand its run-time requirements: libraries, communication matrix, resource (CPU, RAM, HD) footprint, need for dynamic data, etc. Containers can create resource caps for those limits, which is an additional safety factor in the system design.
- containerisation: applications are containerised by creating a container description (e.g. a Dockerfile) that provides the steps showing how to deploy the application over the selected container base image.

- Erstellung von Laufzeit-Deskriptoren. Je nach Fall kann die Anwendung direkt als orchestrierter Container (Kubernetes) oder als verwalteter Container (z. B. mit Podman oder Docker Compose) ausgeführt werden. Die Laufzeit-Deskriptoren definieren die Verwaltungsrichtlinien für die containerisierte Anwendung, z. B. was im Fehlerfall zu tun ist, wie der Fehlerzustand definiert wird oder spezifische Anforderungen an die Laufzeitumgebung und welche Kommunikation die Anwendung erfordert.

Nach der Containerisierung der Anwendung ist ein vollständiger Testzyklus, einschließlich funktionaler und nicht-funktionaler Tests, erforderlich, um sicherzustellen, dass während des Prozesses keine Probleme auftreten.

### 3.2 Erfahrungen aus der realen Welt

Mipro hat sich entschieden, sein TMS komplett auf Microservice- und Containerarchitektur zu migrieren. Die Legacy-Teile der Anwendung wurden bereits in Container portiert. Alle neu entwickelten Teile werden von Anfang an mit containerisierten Microservices implementiert. Die wichtigsten Ergebnisse dieses Prozesses sind im Folgenden aufgeführt:

- Das Verschieben der Legacy-Anwendung in einen Container ist eine gute Gelegenheit, gleich andere Schlüsseltechnologien auf neuere Versionen zu aktualisieren. Bei Mipro haben wir bei allen wichtigen Drittanbieterkomponenten Updates auf die neueste Version vorgenommen, um den vollen Testzyklus zu nutzen, was bei einem Projekt wie der Containerisierung ohnehin nötig ist.
- Bei der Migration ist ein ordnungsgemäßer Funktionstest unerlässlich. Um sicherzustellen, dass keine Regressionsprobleme auftreten, ist eine Investition in automatisierte Funktionstests erforderlich. Die Erstellung einer containerisierten Anwendung erfordert viele Iterationen, daher müssen bei jedem Durchlauf kontinuierliche Regressionstests erfolgen.
- Für die Entwicklung von Fähigkeiten und Kompetenzen waren einige Investitionen erforderlich. Der Paradigmenwechsel, den die Containertechnologie darstellt, muss von den Schlüsselteams richtig verstanden und angenommen werden. Dafür waren sowohl externe Schulungen als auch der Einsatz interner Change Agents erforderlich.
- Sofern nicht bereits vorhanden, sollte eine geeignete DevOps-Infrastruktur mit den entsprechenden Praktiken eingeführt werden. Automatisierte Build-Prozesse und Continuous Deployment (CD) beschleunigen den Entwicklungszyklus und ermöglichen automatisierte Tests für alle Software-Builds. Außerdem sollte ein Pfad für die Softwarebereitstellung von der DevOps-Umgebung (Registry) zur Kundenumgebung erstellt werden.
- Das Geschäftsmodell für die Wartung der bestehenden Kundenimplementierungen während der Entwicklung eines neuen Systems ist sorgfältig zu planen. Das Modell sollte eine Möglichkeit für die Wartung über den Produktlebenszyklus einschließlich Änderungen der Systemarchitektur enthalten. Wenn das Modell nicht flexibel mit dem Kunden abgestimmt werden kann, können die Vorteile der neuen Technologie nicht genutzt werden.
- Ein Cloud-natives Ökosystem bietet vorgefertigte Lösungen für viele Standardprobleme wie Ablaufverfolgung und Logging, Anwendungsüberwachung und sichere Kommunikation. Diese sollten nach Möglichkeit genutzt werden, um den größtmöglichen Vorteil aus der neuen Technologie zu ziehen.

### 3.3 Mögliche Probleme bei der Containerisierung

Wie jede andere Technologie hat auch die Containerisierung Grenzen und eignet sich nicht in jedem Fall. Zum Beispiel sind oft die Details der Umgebungsanforderungen, z. B. die Anwendungskommunikationsmatrix oder Disk-Nutzungsmuster bei Legacy-Anwen-

- the creation of run-time descriptors. The application can be run directly as a container, using orchestration (Kubernetes) or as a managed container (e.g. with the Podman or Docker Compose tools) depending on the case. The run-time descriptors define the management policy for the containerised application, e.g. state what to do in the case of a failure, how to define a failed state or the specific requirements for the run-time environment and what communication the application requires.

A full testing cycle, including functional and non-functional tests, is necessary once the application has been containerised in order to ensure that no issues have been created during the process.

### 3.2 Real world experience

Mipro has decided to migrate its TMS completely into microservice and container architecture. The legacy parts of the application have already been ported to containers and all the newly developed parts are implemented with containerised microservices from the start. Some of the key findings from the process have been listed below:

- moving the legacy application to a container creates an opportunity to update other key technologies to newer versions at the same time. At Mipro, we have updated all the main 3rd party components to their latest versions in order to utilise the full testing cycle that was needed in any case for a project like containerisation.
- proper functional testing is essential for the migration. It is necessary to invest in Functional Test Automation in order to ensure that no regression issues occur. Building a containerised application requires many iterations, so continuous regression testing for each iteration is a must.
- some investment in skills and competence development was necessary. The paradigm change for container technology needs to be properly understood and embraced by the key teams. Therefore, both external training and the use of internal change agents are necessary.
- proper DevOps infrastructure and practices should be set up, unless they are already in place. Practices such as automated builds and continuous deployment (CD) will speed up the development cycle and will enable automated testing to be run against all the software builds. In addition, the software delivery path from the DevOps environment (registry) to the customer environment should also be built.
- the commercial model for how to maintain the existing customer deployments, while developing a new system, has to be carefully planned. The model should include a way of performing product lifecycle maintenance, including system architectural changes. The benefits from the new technology cannot be received, unless the model can be creatively agreed with customer.
- a cloud-native ecosystem provides ready-made solutions for many standard problems, such as tracing and logging, application monitoring and secure communication. They should be used wherever possible to achieve the highest benefits from the new technology.

### 3.3 Possible issues with containerisation

Like any technology, containerisation technology has its limitations and it does not suit every case. For example, the environment requirement details, such as the application communication matrix or disk use patterns for legacy applications or 3rd party applications, may not be available which means that containerisation may be very difficult. In addition, even though

dungen oder Drittanbieteranwendungen, nicht verfügbar, was die Containerisierung sehr erschweren kann. Ein weiterer Punkt: Obwohl zustandsbehaftete Anwendungen (z. B. Datenbanken oder Nachrichtenwarteschlangen) containerisiert werden können, ist es bei ihnen unter Umständen schwieriger, IT-Vorteile zu erzielen, als bei zustandslosen Anwendungen (z. B. UI). Beispielsweise muss bei ihren Zero-Downtime-Updates die Anwendungslogik selbst diese Funktion unterstützen – Containerisierung hilft dabei nicht.
Bei der Migration einer Anwendung in eine Containerarchitektur sollte der Aufbau entsprechender technischer Fähigkeiten innerhalb des Teams im Vordergrund stehen. Sowohl das Team, das die Umgebung verwaltet, als auch dasjenige, das die Anwendung erstellt, muss in der Lage sein, die Grenzen der Technologie und der Plattformen richtig zu verstehen.
Da die Containertechnologie noch recht jung ist, entwickelt sie sich kontinuierlich weiter. Daher ist es sehr wichtig, einen Technologieanbieter zu beauftragen, der Schienenverkehrs-Applikationen über ihren langen Lebenszyklus hinweg unterstützen kann.

### 3.4 Wie steht es mit der Benutzeroberfläche?

Die Containertechnologie eignet sich am besten für die Backend- und Serverseite der Architektur. Ob sie auf der Client-Seite (Benutzeroberflächen) anwendbar ist, hängt von der Architektur der Lösung ab. Wenn die Anwendung vollständig webbasiert ist und die Client-Desktops nur einen Webbrowser als Benutzeroberfläche haben, ist es kein Problem. Für die Client-Seite werden keine Container benötigt. Bei einer Thick-Client-Architektur (z. B. einer Desktop-Anwendung) kann der Client-Teil der Anwendung jedoch möglicherweise nicht in Container migriert werden. Bei solchen Fällen ist die Desktop-Virtualisierung (VDI), z. B. mittels Technologien wie Citrix oder VMware Horizon, ein besserer Lösungsansatz. Auch wenn das System nicht alle Vorteile der Containerisierung nutzen kann, profitiert es doch von einigen Vorteilen einer zentralen IT, z. B. einfache HW-Updates.

## 4 Fazit

Die Auswahl der richtigen Architektur und Technologie für das TMS ist entscheidend, um sicherzustellen, dass Standard- und normale IT-Anforderungen optimal erfüllt werden. Viele schwer implementierbare Anforderungen, insbesondere nicht-funktionale Anforderungen (NFR) wie Cybersicherheit und Betriebskontinuität, werden durch den Einsatz moderner IT-Technologien auch für den Schienenverkehr deutlich einfacher und beherrschbarer. Die Nutzung dieser Technologien erfordert im Betrieb jedoch spezifische IT-Kenntnisse. An anderer Stelle wurde gesagt, dass Container wie ein Haustier seien. Wer nicht bereit ist, sich richtig darum zu kümmern, sollte keines haben. Daher ist es für eine erfolgreiche Digitalisierung des TMS unerlässlich, in die Kompetenzentwicklung eines modernen IT-Managements zu investieren. ■

stateful applications (e.g. databases or message queues) can be containerised, many IT benefits for those applications can be more difficult to obtain than for stateless applications (e.g. the UI). For example, the zero-downtime updates for these applications require the application logic itself to support that feature – containerisation does not help with this.
When migrating an application to a container architecture, sufficient focus should be placed on building up the relevant technical skills in the team. Both the team that is managing the environment and the team that is building the application have to be able to correctly understand the limits of the technology and platforms.
Container technology is undergoing continuous development, because it is still fairly young. It is therefore very important to use a technology vendor that can support the long lifecycles of railway applications.

### 3.4 What about user interfaces?

Container technology works best for back-ends and the server side of the architecture. Applicability to the client side (the user interfaces) depends on how the solution has been designed. If the application is fully web-based and the client desktops only have a web browser for the UI, there is no issue. No containers are needed for the client side. However, in the case of a thick client architecture (e.g. a desktop application), the client part of the application may not able to be feasibly containerised. A better way to manage these cases is via desktop virtualisation (VDI), e.g. using technologies such as Citrix or VMware Horizon. Even though the system will not enjoy the full benefits of containerisation, some of the IT benefits of using centralised IT can still be achieved, such as easy hardware updates.

## 4 Conclusions

Selecting the right architecture and technology for TMS is essential in order to ensure that standard and normal IT requirements are optimally met. Many hard to implement requirements, especially non-functional requirements (NFR), such as cybersecurity and business continuity also become much easier and more manageable for railways with the use of modern IT technologies. However, utilising these technologies requires specific IT skills during operations. It has been said that containers are like pets; if you do not plan to take good care of them, you should not keep them. Therefore, investing in skills development for modern IT management is necessary for the successful digitalisation of TMS. ■



**LITERATUR | LITERATURE**

[1] https://www.freecodecamp.org/news/7-cases-when-not-to-use-docker/
[2] https://www.cncf.io/wp-content/uploads/2020/11/CNCF_Survey_Report_2020.pdf
[3] https://community.connection.com/rethinking-virtualisation-containers-in-the-data-center/
[4] https://www.techradar.com/news/what-is-container-technology

**AUTOR | AUTHOR**

**Kari Haapala, M. Sc.**
Chief Technology Officer
Mipro Oy
Anschrift / *Address*: Kunnanmäki 9, FI-50600 Mikkeli
E-Mail: kari.haapala@mipro.fi