Managing multiple clusters for container run-time environments

Christian Frank (#473088)
February 27, 2020

Wirtschaftsinformatik: IT-Infrastruktur
Dipl.-Wirt.-Inf (FH) Frank R. Becker M.A.
FOM - Hochschule für Oekonomie & Management
WS 2019

In this paper, we'll first have a look at container technologies and orchestration frameworks. After introducing the most popular orchestration framework, Kubernetes, we'll look at challenges for Enterprises to manage multiple Kubernetes clusters. For a possible solution, we'll look at Rancher from Rancher Inc., a solution that offers exciting features to manage multiple clusters. In closing, we'll look at future developments in container environments.

Contents

List of Figures

List of Abbreviations

AWS	Amazon Web Services
CI/CD	Continuous Integration / Continuous Deployment
CLI	Command-Line Interface
CNCF	Cloud Native Computing Foundation
ESX	Elastic Sky X (VMware)
IaaS	Infrastructure as a Service
IaC	Infrastructure as Code
IT	Information Technology
K3s	Kubernetes on the edge
K8s	Kubernetes
NIST	National Institute of Standards and Technology
PaaS	Platform as a Service
RBAC	Role-Based Access Control
REST	Representational State Transfer
S3	Simple Storage Service (Amazon)
SaaS	Software as a Service
YAML	YAML Ain't Markup Language

1 Introduction to Cloud Computing and Container Orchestration Frameworks

1.1 Enterprise IT

Ever since the start of the decade, Enterprise IT has undergone a massive transformation towards a utility-like service business. As of 2019, in Enterprise IT, Cloud Computing is now the new norm, according to Gartner Research,[1] and is expected to grow even further.[2]

In the early days, compute transformation was focused on virtualization, whereas cloud computing now focuses on containerization. Both technologies are quite old, virtualization started in the early 70s, pioneered by IBM, with VM/CMS; the first attempt at containerization was made with the implementation of chroot() for Unix System V in the late 70s.

Outside of the mainframe world, virtualization technologies were not widespread until VMware commoditized virtualization with ESX/ESXi in the early 00s. Around virtualization, a new ecosystem of self-service portals appeared, such as Microfocus' Cloud Service Automation.[3] Enterprise virtualization with a self-service portal is not cloud computing, though - for cloud computing to come to life, the first real open-source cloud operation system, OpenStack was needed, together with the arrival of the major public cloud providers (Amazon Web Services, Microsoft Azure, Google Cloud, Alibaba Cloud).

AWS started initially by providing excess compute capacity to its customers from its internal Amazon platforms, before turning into one of the biggest IT providers worldwide.

With all this raw compute power now available on tap, there is no real reason for Enterprise IT anymore to operate in-house data centers.

1.2 Container Run-time

During that time, Docker pioneered the first simple orchestration environment to run containers, on a single node.[4] Whereas virtualization is focused on virtual compute instances, containerization focuses on application delivery and deployment; with immutable images and a focus on automation, it targets the development of cloud-native

[1] See *Gartner (2019)*: Cloud computing is the new norm. [9]
[2] See *Gartner (2019)*: Gartner Forecasts Worldwide Public Cloud Revenue. [10]
[3] See *Micro Focus (2019)*: Cloud Service Automation. [18]
[4] See *Docker (2019)*: Enterprise Container Platform. [6]

applications.[5]

It is crucial to understand the difference: Virtualization was aimed at the infrastructure level, containerization, on the other hand, is aimed at (agile) application development and deployment. Docker, and especially Docker Images, addresses the age-old problem of "works on my machine" by providing a stable and easily deployable run-time environment. It also paves the way for full automation and Infrastructure as Code (IaC).

Docker evolved and spawned other container run-time environments, such as containerd and podman, but they mostly remained focused on executing on a single node. An orchestration framework is needed to orchestrate containers on more than one node.

1.3 Container Orchestration Frameworks

A key feature of cloud-native applications is horizontal scalability - to enable this, a container run-time environment will most likely consist of multiple nodes with shared network and storage interfaces. Horizontal scaling serves two purposes in cloud-native application deployment; it increases throughput and performance through parallel processing and provides for high-availability through redundancy.

Docker does not provide this out of the box, so that's where the various orchestration frameworks come in.

Here are the most popular in 2019:

- Docker Swarm
- Kubernetes
- Mesos DC/OS

After Mirantis acquired the Docker Enterprise business, it announced the end of life for Docker Swarm in 2021;[6] Mesos DC/OS never gathered a large following, so as of the time of writing, CNCF's Kubernetes remains as the only container orchestration framework with a sizable installed base and that is under active development.

In this paper, we'll have a look at one of the challenges posed by introducing Kubernetes as a container orchestration framework into Enterprise IT and showcase a possible solution.

[5] See *Wiggins, A. (2017)*: The Twelve-Factor App. [34]
[6] See *Mirantis (2019)*: What We Announced Today and Why it Matters. [20]

2 Kubernetes Terms and Concepts

2.1 Kubernetes and Cloud Computing

What exactly is Kubernetes? "Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community."[7]

Also, according to the National Institute of Standards and Technology (NIST), the key characteristics of cloud computing are:

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service[8]

Gartner similarly defines cloud computing as "Cloud computing is a style of computing in which scalable and elastic IT-enabled capabilities are delivered as a service using Internet technologies."[9]

Cloud computing is also the foundation for Continuous Integration / Continuous Deployment (CI/CD) in agile software development.

Sitting on top of the Docker container run-time, Kubernetes is the ideal tool to enable cloud computing and CI/CD on Public Cloud as well as for in-house data centers. It provides a rich set of API calls, aimed at application development and deployment, and is an easy tool to use for automated software development.

Kubernetes is widely adopted, not only in the commercial space but also in the military. "It's a flexible but universal development platform for software teams across the military and prevents vendor-lock in",[10] Nicolas Chaillan explained during his presentation on why the US Air Force chose Kubernetes for their F-16 fighter jets.

[7] *The Linux Foundation (2019)*: Production-Grade Container Orchestration. [32]
[8] See *Mell, P. (2011)*: The NIST Definition of Cloud Computing. [17]
[9] *Gartner (2019)*: Gartner Glossary - Cloud Computing. [11]
[10] *Chaillan, N. (2019)*: How the U.S. Air Force Deployed Kubernetes. [4]

2.2 Kubernetes Architecture

Generally speaking, a Kubernetes cluster consists of one or more logically grouped systems. These systems can be either physical nodes or better, be virtualized, and all of them run a supported container run-time, such as Docker.

A cluster consists of one or more Master nodes, the so-called Control Plane, and one or more worker nodes, the Execution Plane. Most commonly, there are three nodes in the control plane, and three or more nodes in the execution plane, sometimes with different sizing or capabilities. An odd number of nodes should be chosen for the control plane because it's a distributed system and needs to reach quorum on startup.

The control plane takes care of orchestrating the application deployments and maintains their state in a distributed database; the worker nodes execute the actual application containers as defined and scheduled by the control plane.

Kubernetes, in itself, is also a containerized application.

In addition to the compute nodes, Kubernetes also provides an overlay network that allows communication between the cluster nodes, invisible to the outside world, and ingress controllers for applications. Persistent storage, even though it's an anathema to stateless, cloud-native computing, is provided through persistent volumes and storage classes.

All administrative access to a Kubernetes cluster is through the master nodes and the Kubernetes API endpoints it provides. Kubernetes has a command-line interface tool, kubectl, to access the API; a detailed description can be found in the Kubernetes online documentation.[11] Besides the command line interface, most development pipelines, such as Jenkins, now come with native Kubernetes integration.[12]

2.3 Kubernetes Terms and Concepts

The key terms and concepts in Kubernetes are:

- Pod
- Replica Set
- Service
- Deployment

[11] See *The Linux Foundation (2019)*: Overview of kubectl. [30]
[12] See *Jenkins (2019)*: Kubernetes plugin for Jenkins. [16]

- Namespace

A Pod in Kubernetes is the smallest unit that can be deployed (ephemeral primitive). A Pod includes of one or more container images that are always scheduled together on the same node. Each Pod gets a unique IP address, and container images in a pod can speak to each other via localhost.

A Replica Set defines the desired scale and state of a group of pods. Replica sets are not utilized directly; however, the resource is essential as it is the basic building block for application deployments on Kubernetes. Replica sets can (when instructed to) scale up or down to the desired number of pods.

A Service defines a DNS name that refers to a group of pods; the service thus provides a consistent endpoint for that group of pods and aids service discovery. There are different types of services (nodePort, clusterIP, and load balancer); a more detailed explanation would exceed the scope of this paper.

A Deployment is the level of abstraction above replica sets; deployments create and update replica sets and allow to scale applications up or down quickly and perform rolling upgrades if needed. Deployments can also include volume claims for data persistence.

A Namespace is an organizational unit defined within Kubernetes that can group deployments and is used to grant access permissions.

All these items above are defined in plain-text YAML files, which can be kept, together with the application source code, in the central version control system. It is important to note that, except for the actual cluster nodes, all infrastructure definitions (compute, network, storage) have thus moved into the realm and responsibility of the application development teams.

3 Management of Container Platforms in the Enterprise

3.1 Key requirements for Kubernetes cluster in Enterprise IT

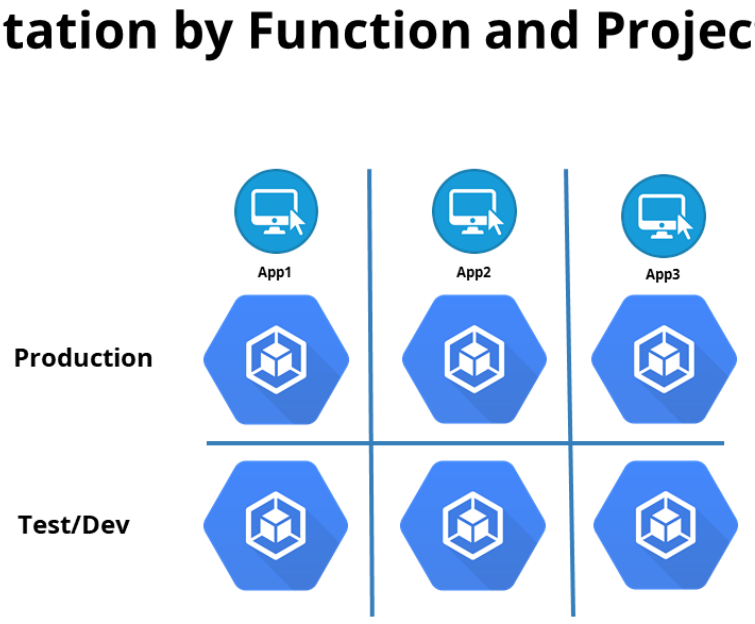
Installing the first Kubernetes cluster is not a big task anymore, especially on the three big public cloud providers, which all offer managed Kubernetes clusters with easy, one-click installations.

Designing the platform landscape, however, does need some architectural knowledge and should be planned well in advance.

Once the transition to cloud-native application development begins, the DevOps teams are in place, and application deployment with containers is introduced to IT production, Day Two operations and security become the primary concerns.

One of the crucial components of security when running containers in production, according to NIST, is the separation in between applications and systems.[13]

Figure 1: Cluster Separation



Segmentation of applications could be performed along the lines of function (Production, Development/Test), or between applications, or both. In single-cluster environments,

[13] See *Souppaya, M. (2017)*: Application Container Security Guide. [28]

separation could be achieved through networking or otherwise through the introduction of multiple clusters.

A key consideration when implementing separation is the so-called "Blast Radius", a term borrowed from the military, which depicts the amount of damage an explosive would cause. IT uses it to assess the damage a breach, data loss, or failure of a given IT system would cause to the whole operation, in technical and financial terms.

Defining application and data security classes is part of the preparation process when introducing containers to production, and would exceed the scope of this paper.

Kubernetes itself, at the time of writing, does not provide for hard tenancy. To entirely separate applications on all layers (compute, network, and storage), the use of multiple clusters is a good option. Many enterprises might thus end up with more than one Kubernetes cluster, sometimes with many more, which will, in turn, have a significant effect on operations.

We're observing a trend in Enterprise IT to move from traditional perimeter-based network security to more modern zero- or low-trust networks with identity-based security and temporary infrastructure. Having multiple, short-lived Kubernetes cluster is a likely scenario going forward.

3.2 The principle of least privilege

The second crucial component of security is user authentication and authorization; especially with the shift towards identity-based security, Enterprise IT needs to establish a trusted identity provider and link it to its container platform.

Many Enterprise IT already have some form of central user authentication through Microsoft Active Directory or any other Single-Sign-On provider. Kubernetes has built-in Role-Based Access Control and can restrict access to its resources based on the roles bound to a specific user or object.

Defining Roles and Responsibilities is part of the Role Engineering process during the introduction of containers, and would exceed the scope of this paper. It is, however, necessary in RBAC to fully embrace the principle of least privilege - only ever grant the minimum access rights required to fulfill the job without jeopardizing efficiency.

A comprehensive reference for RBAC is in David F. Ferraiolo's book of the same name, Role-Based Access Control[8].

3.3 Organizational considerations

The third crucial component of security is departmental organization - NIST recommends that an organization changes its operational culture and technical processes to embrace and support agile application development fully.

Agile software development requires an agile organizational structure and self-organized teams to support the agile mantra "You build it, you run it" (Werner Vogels).[14]

Without such changes, from experience, the likelihood of failing the transformation to container-based, cloud-native application development is quite high.

3.4 Day Two operations with multiple Kubernetai

Once all preparations have been completed, the organizational structure is about to change, and multiple clusters are installed, there are a lot of configuration items and actions that need to be synchronized across all clusters.

Central user authentication and authorization is critical. Authentication can be performed through an identity provider, but roles and responsibilities need to be defined for each cluster, and users need to be assigned the roles for authorization. There's only one hierarchy level on Kubernetes; roles can be bound to user objects and become active in one or more namespaces.

Kubernetes provides two types of security policies, one for pod security and one for network security. Pod security policies, as the name implies, govern security-relevant aspects of pod specification,[15] whereas network policies govern the allowed communication between groups of pods and the outside world.[16]

As time goes on, supporting applications will need to be updated, and Kubernetes itself might need to be updated unless a public cloud provider manages Kubernetes.

To effectively control a Kubernetes cluster and aid in troubleshooting, logging and monitoring should be configured and enabled.

Kubernetes stores its configuration and metadata information in a distributed database. This information state needs to be regularly backed up, and in case of issues, be restored.

To provide for data persistence, volumes and shared file systems need to be created for various Kubernetes clusters.

---

[14] *Orban, S. (2015)*: Enterprise DevOps: Why You Should Run What You Build. [21]
[15] See *The Linux Foundation (2019)*: Pod Security Policies. [31]
[16] See *The Linux Foundation (2019)*: Network Policies. [29]

For software development and automatic deployment, it makes sense to connect the development pipeline(s) to the respective clusters, have a central image registry and establish central governance for deployment workflows.

For micro-service observability, distributed tracing and network control, a service mesh could be installed - a quite popular choice would be Istio.[17]

All of the tasks above can be performed against individual clusters with native Kubernetes tools, but once there are a couple of clusters to manage, this will get tedious and quite difficult to synchronize.

It will even get more complicated once IaC enters the picture, and Kubernetes clusters are starting to become ephemeral with the run-time environment recreated fresh with each new application deployment, entirely relying on automation to implement governance and policies.

3.5 Possible solutions

To address all these issues, CNCF is working on Kubernetes federation and multi-cluster controlling, but the development still in its infancy and not production-ready yet.[18]

The three big cloud providers also provide solutions to manage multiple clusters: Google Cloud has Anthos,[19] Microsoft Azure has Azure Arc in Preview,[20] and AWS is working on AWS Outposts.[21]

A popular open-source solution for this problem is Rancher from Rancher Labs. In the next chapter, we'll have a look at whether Rancher could provide a solution to Kubernetes multi-cluster management and how it would address the issues outlined above.

[17] See *Istio Authors (2019)*: Istio - Connect, secure, control, and observe services. [15]
[18] See *sig-multicluster (2019)*: Kubernetes Cluster Federation. [27]
[19] See *Google (2019)*: Anthos - Bringing the cloud to you. [12]
[20] See *Microsoft (2019)*: Bring Azure services and management to any infrastructure. [19]
[21] See *AWS (2019)*: AWS Outposts. [2]

4 Using Rancher as an Enterprise Container Management Platform

4.1 Rancher overview

What is Rancher? According to the Rancher Labs website, it is "[...] a complete software stack for teams adopting containers. It addresses the operational and security challenges of managing multiple Kubernetes clusters, while providing DevOps teams with integrated tools for running containerized workloads" [22]

Rancher provides a management platform to centrally manage multiple Kubernetes clusters in Enterprise IT, all from a user-friendly GUI. Rancher also offers integration tools for application development and robust enterprise-grade features for security and governance. For operations, Rancher provides integrated solutions for logging, monitoring, and auditing, amongst other features.

Gartner recognized Rancher Labs in 2019 as a Firestarter in Gartner's annual technology award program.[23]

Rancher Labs offers many more open-source projects besides Rancher. There's RIO, a toolchain to streamline application development and deployment, and K3s, the Kubernetes distribution for edge computing. Rancher Labs funds itself solely through revenues from support, training, and consulting; they are not following a freemium or core approach, as many other open-source businesses do.

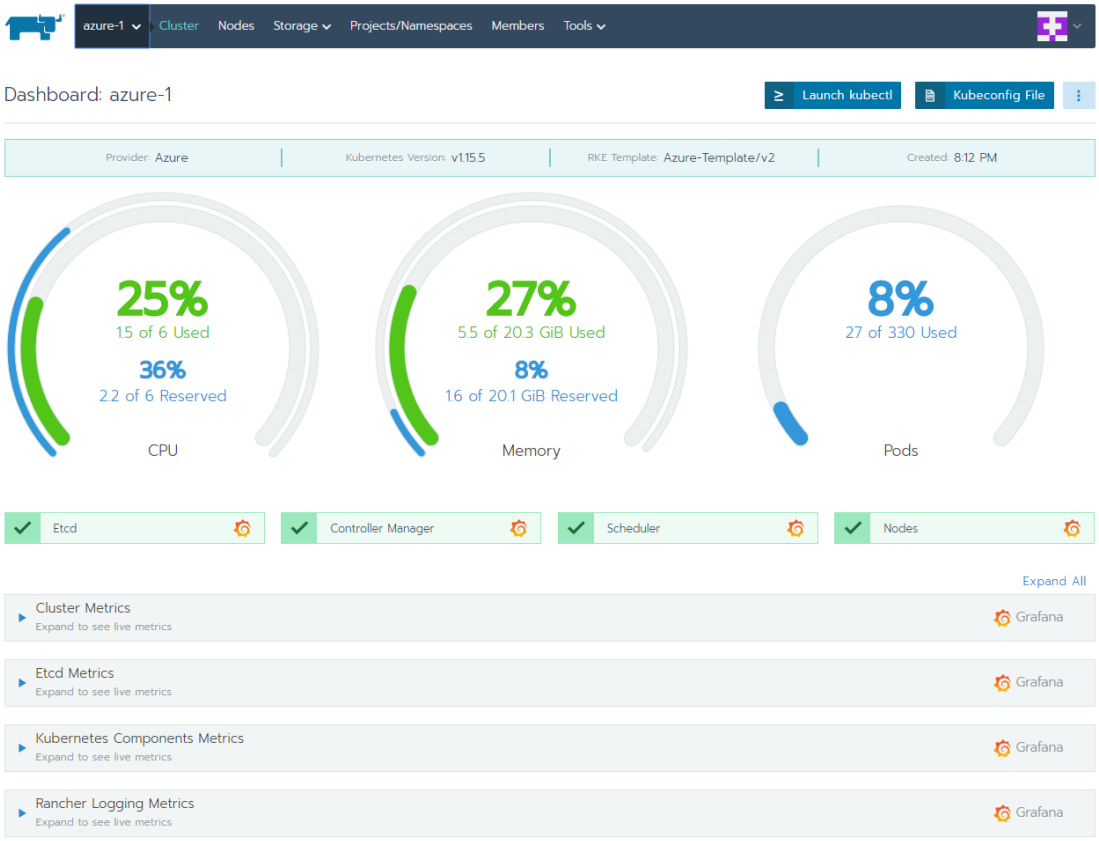
4.2 Rancher GUI

The central element of Rancher is the GUI provided by the Rancher server itself. It offers one-stop access for all administrative tasks necessary on Kubernetes clusters, from installation, upgrading, and decommissioning to hardening. The GUI allows all application administration, and also offers easy access to troubleshooting information. For developers, it provides user authentication and pipeline connection.

Let's look at an example. The Rancher Dashboard for an active Kubernetes cluster looks like this:

[22] *Rancher Labs (2019)*: Run Kubernetes Everywhere. [25]
[23] See *Rancher Labs (2019)*: Rancher Labs Recognized by 451 Research as a '451 Firestarter'. [24]

Figure 2: Rancher Dashboard



This cluster was created on Azure and has cluster monitoring enabled. At a glance we can see that CPU utilization is at 25%, memory utilization is at 27%, and 27 out of 330 possible pods are running; we can also see the Kubernetes version installed (v1.15.5), the cluster template used and the date and time the cluster was created.

The cluster dashboard gives an overview of the overall health of the cluster, including the event stream - a similar, more detailed panel is available for all deployed applications. This information is available for all clusters in a consistent manner, regardless of whether they're hosted on a public cloud provider or in-house. Access to the dashboard is based on user permissions, of course.

From the dashboard, all administrative tasks can be performed by drilling down on the individual components that make up the cluster and are present in the GUI, nodes, volumes, and applications.

4.3 Rancher Architecture

Rancher is, similar to Kubernetes, itself a containerized application and can be installed from a single image to a single Docker host. Such a single-node installation is ideal for testbeds or local Rancher installations on a laptop, for example. A single node installation does not provide any redundancy in case of failure.

For production installations, Rancher can be installed on a Kubernetes cluster, using Kubernetes' redundancy mechanisms for high-availability and resiliency. It could either be co-hosted on an existing cluster or better, on a small, separate infrastructure cluster. It is good practice in IT to keep administrative tools on separate infrastructure from the administered infrastructure, and thus the installation on a different cluster is the most widespread.

In addition to the GUI, the Rancher server also provides a central Kubernetes API endpoint, which acts as an intermediary between the users and the actual Kubernetes clusters.

Rancher too has a command-line interface tool, rancher, to access the API; a detailed description can be found in the Rancher online documentation.[24]

4.4 Authentication and Authorization

Acting as an intermediary, Rancher can centrally authenticate user logins, and API calls through a variety of identity providers, most commonly with Microsoft's Active Directory. Once the user is authenticated, Rancher introduces a new level of hierarchy to Kubernetes, the Project.

Projects on a cluster can include Kubernetes namespaces, workloads or applications and serve as a logical group for teams or applications.

Rancher includes a variety of predefined roles, that can be easily expanded to match any organizational needs.

These roles, and the permissions that come with the roles, can then be assigned to one or many users or groups, on cluster or project/application level, providing fine-grained central role-based access control for all connected Kubernetes clusters.

This authentication and authorization mechanism works for the GUI, the Rancher CLI, or any other Rest-API Call to the Rancher server.

[24] See *Rancher Labs (2019)*: Using the Rancher Command Line Interface. [26]

4.5 Security Policies

Kubernetes provides Pod Security Policies and Network Policies, at the cluster level. The Rancher GUI allows to manage these policies from a central interface and apply them uniformly to all managed clusters; managing these policies from a graphical interface is also much less error-prone than writing YAML files and distributing them manually.

In addition to policies, Rancher provides templates for cluster and node creation that can help to ensure organizational compliance and common security standards. For cluster hardening, Rancher provides a detailed, easy to follow hardening guide.[25]

4.6 Cluster Management

From these templates, the installation of a new cluster through the GUI is quite simple and accomplished with a few clicks. Rancher provides cluster drivers for managed Kubernetes on the three major public cloud providers (Google GKE, Amazon EKS, and Microsoft Azure AKS) and node drivers for the same, plus several infrastructure providers, including VMware.

As of the latest Rancher version (2.3.3), the VMware node driver now supports dynamic provisioning of worker nodes so that fully flexible and breathing clusters can be implemented on an in-house ESXi cluster farm; previously, this feature was limited to public cloud infrastructure.

Clusters provisioned and installed through Rancher can also be upgraded through the Rancher GUI with a single click. Once a new version has been selected, Rancher will perform a rolling upgrade of all master and worker nodes, without application downtime.

For application management, Rancher interacts with the popular package manager Helm.[26] Helm is also used as the base installation mechanism for Rancher's integrated application catalogs, which can be extended.

Helm, directly or through the catalogs, provides version control and upgrades for application deployments.

[25] See *Rancher Labs (2019)*: Hardening Guide - Rancher v2.3.x. [22]
[26] See *CNCF (2019)*: Helm 3 - The package manager for Kubernetes. [5]

4.7 Logging and Monitoring

From the GUI, Logging can be defined for each cluster. Several log drivers are included out-of-the-box, from the venerable Syslog to the more modern Elasticsearch;[27] Rancher will fully install and configure log forwarders and make sure that the log records are complete.

If so desired, Rancher can install and configure a Prometheus monitoring instance with pre-configured Grafana dashboards on each cluster;[28] [29] the Grafana dashboards for cluster and application monitoring are all accessible from the Rancher GUI.

4.8 Backup and restore

Configuration and state of a Kubernetes cluster are stored in a distributed key-value store. For clusters provisioned by Rancher, Rancher can provide regularly scheduled backups of this datastore, either locally or to an S3-compliant object store; Rancher also offers the functionality to restore such snapshots from the GUI.

4.9 Persistent volumes

For persistent data storage, Kubernetes can provide volume mounts to application pods via the deployment definition. The available classes of volumes are dependent on the underlying infrastructure provider and the installed storage classes.

Rancher offers the creation and management of persistent volumes from the GUI.

Rancher does not offer backup and restore services for these volumes, though.

4.10 CI/CD

For application development, Rancher can install and configure a complete CI/CD pipeline from the GUI on a given cluster; the pipeline is based on Jenkins and can connect to all major source-code revision control systems.

Good software development practice calls for credentials to be stored apart from source code. Kubernetes offers secrets as objects for this purpose and has a built-in secret store. Rancher also has an integrated secret store, with significantly higher encryption and improved access control.

[27] See *Elasticsearch (2019)*: The heart of the Elastic Stack. [7]
[28] See *The Linux Foundation (2019)*: From metrics to insight. [33]
[29] See *Grafana Labs (2019)*: The open observability platform. [13]

It is also good practice to separate code and configuration. Kubernetes provides config map objects for this purpose, and Rancher allows the creation and configuration of config maps from the GUI.

4.11 Service Mesh integration

Installation and configuration of a service mesh can be a daunting task. With Rancher 2.3, the automated installation of a service mesh based on Istio, with automatic injection, is now included in the GUI.

Istio is quite resource-hungry, though, and should only be deployed on clusters with ample spare capacity.

4.12 Infrastructure as Code

All major cloud providers have their infrastructure scripting tools, but there's a declarative tool that's available for all infrastructure platforms, in-house or public, Terraform by HashiCorp.[30].

As of Rancher 2.3, Rancher now has a Terraform provider, and cluster creation and decommissioning can easily be performed from a Terraform plan as part of a move of IT to IaC.[31]

[30] See *HashiCorp (2019)*: Deliver infrastructure as code with Terraform. [14]
[31] See *Rancher Labs (2019)*: Introducing the Rancher 2 Terraform Provider. [23]

5 Summary and recommendations

As recommended by NIST, most Enterprise IT will most likely end up with more than one Kubernetes cluster in production, either permanently as part of the infrastructure or ephemerally as part of application deployment.

We have seen that Rancher currently provides an excellent tool for Enterprise IT to manage multiple Kubernetes clusters and offers all the necessary capabilities to operate Kubernetes successfully in production. Rancher also provides easy-to-use mechanisms for governance, security, and compliance.

There are other options, though. Google Cloud is enhancing Anthos, AWS has AWS Outposts, and Microsoft Azure has Azure Arc (in preview). These tools aim at extending the management capabilities of the respective cloud provider to other public cloud platforms and in-house data centers, and to provide management of multiple Kubernetes clusters.

For the time being, however, Rancher is the only open-source, provider-independent solution and the recommended choice.

There are upcoming changes on the horizon that threaten infrastructure itself: With the advent of serverless functions and Event-Driven Architecture, new patterns emerge in cloud-native application design that have the potential to render virtualization and containerization obsolete. The leading contenders in this space are AWS Lambda, Google Cloud Run, Google Cloud Functions, and Microsoft Functions.

Also, AWS and Cloudflare are developing new container run-time environments and micro-VMs; the most significant announcement at the end of 2019 was the introduction of Kubernetes on Fargate by AWS.[32]

There are a lot of developments coming in 2020 with uncertain outcomes, however, what we can still say for sure is that software will eat the world.[33]

[32] See *AWS (2019)*: Run Serverless Kubernetes Pods Using Amazon EKS.[3]
[33] See *Andreessen, M. (2011)*: Why Software Is Eating the World.[1]

References

[1] M. Andreessen. (2011) Why software is eating the world. [Access 2020-01-05]. [Online]. Available: https://a16z.com/2011/08/20/why-software-is-eating-the-world/

[2] AWS. (2019) Aws outposts. [Access 2020-01-05]. [Online]. Available: https://aws.amazon.com/outposts/

[3] AWS. (2019) Run serverless kubernetes pods using amazon eks. [Access 2020-01-05]. [Online]. Available: https://aws.amazon.com/about-aws/whats-new/2019/12/run-serverless-kubernetes-pods-using-amazon-eks-and-aws-fargate/

[4] N. Chaillan and T. Krazit. (2019) How the u.s. air force deployed kubernetes and istio on an f-16 in 45 days. [Access 2020-01-05]. [Online]. Available: https://thenewstack.io/how-the-u-s-air-force-deployed-kubernetes-and-istio-on-an-f-16-in-45-days/

[5] CNCF. (2019) Helm 3: The package manager for kubernetes. [Access 2020-01-05]. [Online]. Available: https://helm.sh/

[6] Docker. (2019) Enterprise container platform. [Access 2020-01-05]. [Online]. Available: https://www.docker.com/

[7] Elasticsearch B.V. (2019) Elasticsearch - the heart of the elastic stack. [Access 2020-01-05]. [Online]. Available: https://www.elastic.co/products/elasticsearch

[8] D. Ferraiolo, R. Kuhn, and R. Chandramouli, *Role-Based Access Control*, 2nd ed. Artech Print on Demand, 2007.

[9] Gartner. (2019) Cloud computing is the new norm. [Access 2020-01-05]. [Online]. Available: https://www.gartner.com/en/information-technology/insights/cloud-strategy

[10] Gartner. (2019) Gartner forecasts worldwide public cloud revenue to grow 17.5 percent in 2019. [Access 2020-01-05]. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2019-04-02-gartner-forecasts-worldwide-public-cloud-revenue-to-g

[11] Gartner. (2019) Gartner glossary - cloud computing. [Access 2020-01-05]. [Online]. Available: https://www.gartner.com/en/information-technology/glossary/cloud-computing

[12] Google. (2019) Anthos - bringing the cloud to you. [Access 2020-01-05]. [Online]. Available: https://cloud.google.com/anthos/

[13] Grafana Labs. (2019) The open observability platform. [Access 2020-01-05]. [Online]. Available: https://grafana.com/

[14] HashiCorp. (2019) Deliver infrastructure as code with terraform. [Access 2020-01-05]. [Online]. Available: https://www.terraform.io/

[15] Istio Authors. (2019) Istio - connect, secure, control, and observe services. [Access 2020-01-05]. [Online]. Available: https://istio.io/

[16] Jenkins. (2019) Kubernetes plugin for jenkins. [Access 2020-01-05]. [Online]. Available: https://github.com/jenkinsci/kubernetes-plugin

[17] P. Mell and T. Grance. (2011) The nist definition of cloud computing. [Access 2020-01-05]. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-145

[18] Micro Focus. (2019) Cloud service automation - cloud management platform. [Access 2020-01-05]. [Online]. Available: https://www.microfocus.com/en-us/products/cloud-service-automation/overview

[19] Microsoft. (2019) Bring azure services and management to any infrastructure. [Access 2020-01-05]. [Online]. Available: https://azure.microsoft.com/en-us/services/azure-arc/

[20] Mirantis. (2019) What we announced today and why it matters. [Access 2020-01-05]. [Online]. Available: https://www.mirantis.com/blog/mirantis-acquires-docker-enterprise-platform-business/

[21] S. Orban. (2015) Enterprise devops: Why you should run what you build. [Access 2020-01-05]. [Online]. Available: https://aws.amazon.com/blogs/enterprise-strategy/enterprise-devops-why-you-should-run-what-you-build/

[22] Rancher Labs. (2019) Hardening guide - rancher v2.3.x. [Access 2020-01-05]. [Online]. Available: https://rancher.com/docs/rancher/v2.x/en/security/hardening-2.3/

[23] Rancher Labs. (2019) Introducing the rancher 2 terraform provider. [Access 2020-01-05]. [Online]. Available: https://rancher.com/blog/2019/rancher-2-terraform-provider/

[24] Rancher Labs. (2019) Rancher labs recognized by 451 research as a '451

firestarter'. [Access 2020-01-05]. [Online]. Available: https://rancher.com/press/2019-09-26-451-research-firestarter-press-release/

[25] Rancher Labs. (2019) Run kubernetes everywhere. [Access 2020-01-05]. [Online]. Available: https://rancher.com/

[26] Rancher Labs. (2019) Using the rancher command line interface. [Access 2020-01-05]. [Online]. Available: https://rancher.com/docs/rancher/v2.x/en/cli/

[27] sig multicluster. (2019) Kubernetes cluster federation. [Access 2020-01-05]. [Online]. Available: https://github.com/kubernetes-sigs/kubefed

[28] M. Souppaya, J. Morello, and K. Scarfone. (2017) Application container security guide. [Access 2020-01-05]. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-190

[29] The Linux Foundation. (2019) Network policies. [Access 2020-01-05]. [Online]. Available: https://kubernetes.io/docs/concepts/services-networking/network-policies/

[30] The Linux Foundation. (2019) Overview of kubectl. [Access 2020-01-05]. [Online]. Available: https://kubernetes.io/docs/reference/kubectl/overview/

[31] The Linux Foundation. (2019) Pod security policies. [Access 2020-01-05]. [Online]. Available: https://kubernetes.io/docs/concepts/policy/pod-security-policy/

[32] The Linux Foundation. (2019) Production-grade container orchestration. [Access 2020-01-05]. [Online]. Available: https://kubernetes.io/

[33] The Linux Foundation. (2019) Prometheus - from metrics to insight. [Access 2020-01-05]. [Online]. Available: https://prometheus.io/

[34] A. Wiggins. (2017) The twelve factor-app. [Access 2020-01-05]. [Online]. Available: https://12factor.net/