Kubernetes Deployment Options for On-Prem Clusters

Lincoln Bryant, Robert W. Gardner, Fengping Hu, David Jordan
Enrico Fermi Institute, University of Chicago
Ryan P. Taylor
University of Victoria

June 2024

1   Introduction

First released by Google over a decade ago, Kubernetes [1] is an open-source service orchestration platform built to address the challenges of running containerized workloads at scale. Kubernetes is an open successor to an internal container orchestration system, Borg [2], that manages the application lifecycle of many of Google's most popular products today. This shift to an open ecosystem is facilitated by a vendor-neutral oversight and support organization, the Cloud Native Computing Foundation (CNCF) [3], a part of the Linux Foundation. The CNCF hosts core pieces of the Kubernetes development infrastructure, promulgates the cloud-native computing definition and technical vision through a Technical Oversight Comittee [4], and organizes industry-wide conferences (such as KubeCon and Cloud Native Con). This has led to a thriving landscape of cloud native software products covering application definition and development, orchestration and management, runtime and provisioning products [5].

Kubernetes is now used, maintained, and extended by hundreds of companies world-wide with thousands of contributors and end-users. It also is growing in popularity in the field of high energy physics computing, with an increasing number of infrastructures managed by the platform [6–12]. An un-opinionated and flexible architecture has allowed Kubernetes to become a ubiquitous, vendor-neutral platform for running a variety of workloads. In cloud computing, this has allowed vendors to tailor their Kubernetes offerings in a way that best fits hyperscale infrastructure while abstracting away the details of vendor-specific storage, networking and instancing from users. This is also a significant advantage for developers because the de-facto standardized abstractions provided by Kubernetes help avoid vendor lock-in and contribute significantly to a flourishing ecosystem.

However, system administrators who plan to deploy their own cluster on bare metal or otherwise on-premise may find the flexible and pluggable design of Kubernetes to be challenging. To deploy a functioning cluster into production, administrators must choose appropriate tools to provide essential networking, storage, security and monitoring among other items. To ease the difficulty of constructing a production Kubernetes cluster, a number of opinionated approaches have coalesced in the broader community. Indeed, Kubernetes is often likened to a "kernel of a distributed operating system". To extend this metaphor, efforts to provide fully integrated Kubernetes deployments should perhaps be seen as Kubernetes distributions, much in the same way as Debian or Red Hat are distributions in the Linux environment.

2   Deployment of Kubernetes Distributions

The following sections describe three different Kubernetes distributions, including their deployment method and any additional steps needed to bring a cluster into production. This note covers kubeadm [13] and Kubespray [14], Red Hat's OKD [15] and its commercially supported downstream version OpenShift, as well as SUSE's Rancher [16] which allows for deployment via the light-weight K3S [17] distribution as well as RKE2 [18], a distribution focused on the US Federal Government sector.

2.1   *kubeadm*

Following the metaphor of Linux distributions, the *kubeadm* tool shipped by the Kubernetes authors is most similar to a distribution that ships only a minimal base system with no "batteries included" to allow for maximum flexibility. While not completely manual, *kubeadm* is designed to quickly instantiate the minimum viable Kubernetes cluster. The authors intend [19] for *kubeadm* to be an un-opinionated building block used by higher-level tools to construct clusters. One such high-level tool, Kubespray, will be examined later in this article.

2.1.1   Control plane initialization

When creating a Kubernetes cluster via *kubeadm*, the Kubernetes control plane is initialized in the following way:

- The *kubelet* daemon is started on the control plane host(s).

- Certificates are issued for intra-cluster communication.

- The distributed key-value store *etcd* is instantiated to hold cluster state.

- The Kubernetes API service is started as a container on the control plane host(s).

Once the control plane has been initialized, the administrator may use the *kubeadm* tooling to add additional workers to the cluster. However before the cluster can be utilized, it is essential that the administrator first pick and deploy an appropriate networking plugin [20]. In our community, the Calico [21] plugin appears to be the most popular, given ease of setup/administration as well as a broad experience base.

2.1.2   Declarative deployments with Kubespray

The *kubeadm* tool is sufficient for building and managing clusters, but often times a higher-level tool is useful for programmatic management of clusters. Building on the popular configuration management tool Ansible [22], the Kubespray tool abstracts many of the deployment steps for a *kubeadm*-based cluster on bare metal or virtual machines. With Kubespray, administrators can write a declarative YAML configuration to deploy a highly-available Kubernetes across a variety of Linux distributions with a sensible default configuration while retaining significant leeway in tools and plugin choices. Kubespray deployment is especially convenient in that it is designed for idempotence, such that repeated operations of the Kubespray "playbook" will result in the same cluster deployment, including resuming an interrupted deployment. Beyond installation, Kubespray offers convenient tooling for upgrading clusters between major versions of Kubernetes.

One potential drawback of Kubespray is that the installation mechanism is push-based, relying on making outbound SSH connections to nodes in order to configure and manage them. The SSH-based push model may also be unsuitable for situations that require entire nodes to be dynamically added or removed from the cluster.

2.2   OpenShift/OKD

Red Hat's OKD is another downstream, fully compliant Kubernetes distribution that integrates a number of value-added components not found in the product published by the CNCF. Red Hat also sells a supported edition, *OpenShift*, wherein they provide support for 4 minor releases of OpenShift, each having a 4 month release cycle, giving users at least 16 months of support for a given release [23]. Red Hat has positioned OpenShift/OKD as an Enterprise solution for Kubernetes deployments with built-in features including: multi-tenancy, CI/CD, container registry, monitoring and log aggregation, metering, virtualization for traditional applications, and so on [24]. For the purposes of this article, we will refer to these two related distributions together as simply OKD.

2.3   CoreOS and Ignition

Unlike other Kubernetes distributions, OKD can manage the entire lifecycle of a node in the cluster down to the baseboard management controller (via the Redfish API [25], an industry standard specification for RESTful management of servers). The OKD authors provide four ways to install OKD on bare-metal [26], including an automated installer using Redfish, a web-based assistant built on top of the automated installer, an agent-based installer designed for restrictive or air-gapped networks, and a "full control" installation recipe that allows administrators to manage the cluster installation themselves.

The OKD Kubernetes distribution is tightly integrated with Fedora CoreOS [27], a container-focused operating system that implements many of the Cloud Native Computing principles including immutability and declarative configuration at a low-level. Fedora CoreOS, like other CoreOS derivatives including Flatcar Linux [28], includes a relatively new server provisioning and initial configuration software *Ignition* [29]. Ignition uses a declarative configuration format to manage disks, filesystems, and services in CoreOS. OKD uses Ignition to create immutable server configurations, such that operating system updates are treated as an atomic process, with the server booting into an entirely new OS image loaded from the network after PXE.

By treating servers in a Kubernetes cluster as ephemeral objects much in the same way as Kubernetes treats containers in pods, the OKD distribution makes it easy to scale to integrate additional (or remove deprecated/repurposed) hardware as needed. However, because OKD assumes a high level of control over the server provisioning environment, it may be challenging in some instances to integrate it with existing site provisioning infrastructure.

2.4   Rancher

Developed by SUSE, Ranchers occupies a space between the CNCF's upstream Kubernetes release and the highly-opinionated Openshift/OKD distribution. Rancher offers many features similar to OKD, including a web dashboard, options for deploying in especially security-conscious environments (such as air-gapped clusters), prescriptive choices of plugins including Helm and the Nginx Ingress Controller, etc. Like Red Hat, SUSE offers support contracts and licenses for Rancher that guarantee a certain level of Enterprise support for Kubernetes clusters. Rancher offers two versions of Kubernetes for different deployment targets: K3S for single-node clusters and edge networks, and RKE2 for a larger clusters and those that need strong security guarantees.

2.4.1 K3S

Designed to be fully compliant with upstream Kubernetes, K3S [17] is an opinionated Kubernetes distribution, built by the Rancher authors, that provides a base installation as well as a networking plugin (Flannel), DNS solution (CoreDNS), Ingress controller (Traefik), and other needed tools for a production instance. One unique feature of K3S is that it allows the administrator to use a traditional SQL database (e.g. SQLite, Postgres, MariaDB) instead of the embedded *etcd* key-value database used by other Kubernetes distributions. However, K3S is explicitly not a fork of upstream Kubernetes and seeks to maintain compatibility with the core functionality while removing in-tree cloud and storage providers to shrink the final binary size.

While it is possible to run larger sites on K3S, it seems especially suitable for sites with constrained available resources, as it is optimized for deployment on edge networks. One downside of K3S is that the default SQLite-based database is not particularly suitable for high-availability deployments.

2.4.2 RKE2 / Rancher Government

The Rancher developers have also developed a Kubernetes distribution specifically targeted for compliance with the U.S. Government's "Federal Information Processing Standard" (FIPS): RKE2, also known as Rancher Government [18]. Compared to K3S, RKE2 has a stronger security posture and is DISA STIG-certified [30] for environments with especially demanding security requirements. RKE2 more closely tracks the upstream CNCF Kubernetes releases than K3S, supports high availability with an odd-numbered quorum of control plane nodes, and uses the typical etcd database while more tightly bundling various disparate Kubernetes plugins and software.

3 Comparison of features

While we cover just three popular Kubernetes deployment methods observed in High Energy Physics computing, there are a significant number of other Kubernetes integrators working in the Cloud Native Computing space [31], including the many downstream Cloud providers that offer proprietary solutions for popular Kubernetes features. Indeed, it is evident that there is no "one size fits all" solution for Kubernetes. To help sites decide on a distribution that is most appropriate for them, we will directly compare Kubespray using the official Kubernetes deployment tool kubeadm, OKD/OpenShift, and Rancher/K3S/RKE2 under a few different lenses.

3.1 Provisioning and deployment

Table 1 draws comparisons of the initial cluster deployment strategy. This is a key differentiator between Kubernetes distributions because different products have various scopes for managing the underlying operating system and server hardware. Note that OpenShift / OKD requires more nodes than other distributions for initial cluster bootstrapping. This is because a High Availability configuration is not considered optional under the OKD model, and it therefore does not support a single node control plane configuration. Sites should also be aware that Kubespray and OKD both require a node external to the cluster for the initial cluster provisioning. For Kubespray clusters, it would be wise to keep available a machine that can SSH to all nodes in the cluster for ongoing maintenance and operation. In the case of OKD, the documentation notes [32] that the provisioning machine can be removed after cluster initialization.

	Kubespray	OpenShift/OKD	Rancher
Minimum number of nodes	2 (SSH bootstrap + Kubernetes control plane)	4 (Provisioner + 3 Kubernetes control planes)	1 (Kubernetes control plane)
Deployment mechanism	Ansible playbook via SSH	Web-assisted installer, commandline automated installer, agent-based installer for restricted networks, manual installation	Commandline utility
High availability	Optional	Required	Optional
Linux distribution supported	Several, c.f. [14]	Fedora CoreOS	Several, c.f. [17]
Networking plugin	Several, c.f. [33]	OVN-Kubernetes (default), OpenShift SDN [34]	Flannel (K3S), Several c.f. [35] (RKE2)

Table 1: Kubernetes Cluster Deployment mechanisms and requirements

3.2 Essential additional tools and operators

After a cluster has been deployed, there are a number of additional plugins and integrations that a cluster administrator may need to deploy before the cluster is production-ready. The choice of which additional features to apply to a cluster is a key differentiator spanning the gamut of bare metal and cloud deployment options.

- **Persistent Storage**: If no persistent storage has been defined, then only ephemeral or stateless services can run. For certain types of caches or web services this may be appropriate, but in general some persistent storage is essential for databases, user data, etc. Sites typically use networked Rook [36] with Ceph storage or local storage devices for this purpose with various trade-offs. Red Hat's Openshift Data Foundation product is based on Rook and NooBaa (a multi-cloud object gateway), while Rancher's Longhorn [37] product delivers replicated block storage to services.

- **Load Balancer**: Services running in Kubernetes may have internally visible *ClusterIPs* or may be presented externally on an ephemeral port via *NodePorts*. For services which require a dedicated public IP on a standard port (e.g., web servers), a Load Balancer service is needed for this function. Some examples of general purpose Kubernetes load balancer software includes MetalLB [38] or PureLB [39].

- **Application routing via Ingress**: The ingress controller operates at the application level to proxy internet-facing traffic (typically HTTP/HTTPS, but encapsulating and proxying generic TCP workloads is possible) to back-end services running in a cluster. Typically Nginx [40] or Traefik [41] are used as ingress controllers. OKD's ingress controller is based on HAProxy, while Rancher uses Nginx.

- **Certificate Management**: Applications that are secured via TLS typically need a certificate issued that is recognized by standard Certificate Authorities. Typically tools like *cert-manager* [42] are used to automatically issue certificates through free authorities like Let's Encrypt [43] using the ACME protocol [44].

- **DNS Management**: Services running through Kubernetes will typically need DNS records created and assigned to internet-facing IPs in order to be usable in production. While it is possible to create these records manually, it is often cumbersome, especially if wildcard

records are not appropriate for a given application. Instead, administrators will frequently use tools like ExternalDNS [45] to programmatically manage the entire lifecycle of a DNS record including creation, update and deletion.

3.3 Commonly used extras

While not strictly necessary for production operations, many administrators add the following additional tools to their clusters:

- **Safe secret storage**: In the default configuration [46], secrets in Kubernetes are somewhat misleadingly named as they are stored in plaintext at rest in the database and are merely base64 encoded for transport. Encryption at rest requires additional tooling [47] that is typically not enabled by default. Additionally, secrets that are stored in an external repository can be managed via tools like Sealed Secrets [48] or SOPS [49].

- **Monitoring and Alerting**: Kubernetes does not have any significant tooling for monitoring and alerting included in the base installation. Many administrators choose to use a tool like Prometheus [50] combined with Grafana [51] to monitor and build alerts for their clusters. Tightly integrated distributions such as OKD and Rancher include Prometheus and Grafana by default.

- **GitOps**: Tools such as FluxCD [52], ArgoCD [53] allow Kubernetes objects to be managed via Git source code repositories for continuous delivery deployment patterns. While the base Kubernetes installation does not include a GitOps tool, OKD specifically offers OpenShift GitOps integration [54] based on ArgoCD. Rancher meanwhile offers its own tool for GitOps, Fleet [55].

- **Web Interface**: Minimalist installation methods like *kubeadm* and Kubespray do not include a web interface by default, however the Kubernetes Dashboard [56] can be added after installation. Tools like Rancher and OKD include built-in custom web interfaces.

From Table 2, it is clear that many of the integrations commonly needed in production environments are not included with minimal deployments. However, this is not to say that they are not possible to install. Instead, these integrations simply do not come pre-installed to maximize flexibility. Some integrations, e.g. GitOps, are considered non-essential but come with optional add-on components such as Red Hat's OpenShift GitOps or SUSE's Fleet for OKD and Rancher, respectively.

4 Conclusions

The possible configuration space for Kubernetes deployments is immense, driven by its widespread popularity and adoption in many disciplines and enterprises, and its continuous evolution. This diversity of options means there is no one-size-fits-all solution. Each deployment method offers advantages and trade-offs. Site administrators will want to carefully weigh specific requirements and constraints against local requirements and restrictions. In our field, obviously security and resource sharing considerations may weigh heavily in the choice of deployment method and distribution. We also encourage developers to take note of these differences, or more restrictive environments in some cases, when creating containerized applications and services for Kubernetes platforms.

	Kube-spray	OpenShift/OKD	Rancher
Persistent Storage	Not included	OpenShift Data Foundation [57]	Longhorn [37]
Load Balancer	Not included	MetalLB [34]	Optional [58]
HTTPS Certificate Management	Not included	(Optional) cert-manager Operator [59]	Not included
Ingress Controller	Not included	HAProxy-based [60]	Nginx-based [61]
External DNS Management	Not included	(Optional) External DNS Operator [62]	Not included
Secret Encryption	Not by default	Not by default [63]	Yes, via AES-CBC [64]
GitOps	Not included	(Optional) OpenShift GitOps (ArgoCD)	Fleet
Monitoring & Alerting	Not included	Prometheus and Grafana (built-in) [65]	Prometheus and Grafana (built-in) [66]
Web Interface	Not included	Yes	Yes

Table 2: Additional Integrations for Production Environments

5 Acknowledgements

In preparing this note, we utilized ChatGPT [67] in an early draft, following the principles established in *Nature 613, 612 (2023)* [68]. Our methods included: 1) using the language model's re-write feature to enhance text clarity; 2) employing the reference look-up feature to identify optimal literature sources; 3) updating and verifying these sources for accuracy, and ensuring they aligned with our own judgments based on several years of practical deployment experience; and 4) engaging in contextual dialogs by posing specific questions to develop narratives about tool set adoption and breadth activities in the cloud-native landscape. In addition to adhering to the principles adopted by the editorial board of *Nature*, we recognize transparency opinions noted in [69] and ethical warnings highlighted in [70].

References

[1] Kubernetes — production-grade container orchestration. `https://kubernetes.io/`, 2018.

[2] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. Borg, omega, and kubernetes. *ACM Queue*, 14:70–93, 2016.

[3] CNCF. Cloud native computing foundation (cncf). `https://www.cncf.io/`. Accessed: 2024-06-14.

[4] CNCF Technical Oversight Committee. Cncf technical oversight committee. `https://www.cncf.io/people/technical-oversight-committee/`, 2024. Accessed: 2024-06-20.

[5] CNCF. Cncf cloud native landscape. `https://landscape.cncf.io/`, 2024. Accessed: 2024-06-20.

[6] Lincoln Bryant. Second k8s-hep meetup. `https://indico.cern.ch/event/968726/overview`, 2020. Accessed: 2024-06-20.

[7] Fernando Harald Barreiro Megino et al. Using Kubernetes as an ATLAS computing site. *EPJ Web Conf.*, 245:07025, 2020.

[8] Ryan Paul Taylor, Jeffrey Ryan Albert, and Fernando Harald Barreiro Megino. A grid site reimagined: Building a fully cloud-native ATLAS Tier 2 on Kubernetes. *EPJ Web Conf.*, 295:07001, 2024.

[9] Fernando Barreiro Megino, Kaushik De, Johannes Elmsheuser, Alexei Klimentov, Mario Lassnig, Miles Euell, Nikolai Hartmann, Tadashi Maeno, Verena Martinez Outschoorn, Jay Ajitbhai Sandesara, and Dustin Sell. Operational experience and r&d results using the google cloud for high energy physics in the atlas experiment. `https://arxiv.org/abs/2403.15873`, 2024.

[10] R. Rocha. Kubernetes, magnum, openshift. `https://indico.cern.ch/event/1264168/?view=standard_nu` 2024. Accessed: 2024-06-20.

[11] Sam Albin, Garhan Attebury, Kenneth Bloom, Brian Bockelman, Carl Lundstedt, Oksana Shadura, and John Thiltges. Coffea-Casa: Building composable analysis facilities for the HL-LHC. *EPJ Web Conf.*, 295:07009, 2024.

[12] Tommaso Tedeschi, Vincenzo Eduardo Padulano, Daniele Spiga, Diego Ciangottini, Mirco Tracolli, Enric Tejedor Saavedra, Enrico Guiraud, and Massimo Biasotto. Prototyping a root-based distributed analysis workflow for hl-lhc: The cms use case. *Computer Physics Communications*, 295:108965, 2024.

[13] Kubernetes Project. kubeadm. `https://kubernetes.io/docs/reference/setup-tools/kubeadm/`, 2024. Accessed: 2024-06-20.

[14] Kubespray Contributors. Kubespray: Kubernetes deployment with ansible. `https://github.com/kubernetes-sigs/kubespray`. Accessed: 2024-06-14.

[15] OKD Project. Okd - the origin community distribution of kubernetes. `https://www.okd.io/`, 2024. Accessed: 2024-06-20.

[16] Rancher Labs. Rancher: Open source enterprise kubernetes management. `https://rancher.com/`, 2024. Accessed: 2024-06-20.

[17] Rancher Labs. K3s: Lightweight kubernetes. `https://k3s.io/`, 2024. Accessed: 2024-06-20.

[18] Rancher Government Solutions. Rke2: The next generation kubernetes distribution. `https://ranchergovernment.com/products/rke2`, 2024. Accessed: 2024-06-20.

[19] Kubernetes Documentation. Creating a cluster with kubeadm. `https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubea` 2024. Accessed: 2024-06-21.

[20] Project Calico. *The Kubernetes Networking Guide.* Project Calico, 2021. Available for free download from the Calico website.

[21] Project calico documentation. `https://docs.projectcalico.org/`. Accessed: 2024-06-14.

[22] Red Hat, Inc. Ansible: Simple it automation. `https://www.ansible.com/`. Accessed: 2024-06-14.

[23] Red Hat. Openshift container platform life cycle policy. `https://access.redhat.com/support/policy/updates/openshift`, 2024. Accessed: 2024-06-20.

[24] Red Hat. Red hat openshift and kubernetes. `https://www.redhat.com/en/technologies/cloud-computing` 2024. Accessed: 2024-06-20.

[25] DMTF. Redfish: Scalable platform management. `https://www.dmtf.org/standards/redfish`, 2024. Accessed: 2024-06-20.

[26] OKD Project. Deploying okd on bare metal. `https://docs.okd.io/latest/installing/installing_bare_` 2024. Accessed: 2024-06-20.

[27] Fedora Project. Fedora coreos. `https://getfedora.org/coreos/`, 2024. Accessed: 2024-06-20.

[28] Flatcar Linux. Flatcar linux: A friendly fork of coreos container linux. `https://www.flatcar.org/`, 2024. Accessed: 2024-06-20.

[29] CoreOS. Ignition: The next generation machine provisioning. `https://coreos.com/ignition/`, 2024. Accessed: 2024-06-20.

[30] Defense Information Systems Agency (DISA). Disa releases the rancher government solutions rke2 security technical implementation guide. `https://tinyurl.com/2faedy3n`, 2024. Accessed: 2024-06-20.

[31] Nubenetes. Kubernetes comparison matrix table. `https://nubenetes.com/matrix-table/`, 2024. Accessed: 2024-06-20.

[32] OKD Project. Installing a cluster on bare metal with the openshift installer. `https://docs.okd.io/latest/installing/installing_bare_metal_ipi/ipi-install-prerequisites.` 2024. Accessed: 2024-06-20.

[33] Container Networking Interface (CNI). Cni: The container network interface. `https://github.com/containernetworking/cni`, 2024. Accessed: 2024-06-21.

[34] Red Hat Documentation. About networking in openshift container platform 4.15. `https://docs.openshift.com/container-platform/4.15/networking/about-networking.html`, 2024. Accessed: 2024-06-21.

[35] RKE2 Documentation. Basic network options. `https://docs.rke2.io/networking/basic_network_options`, 2024. Accessed: 2024-06-21.

[36] Rook Project Contributors. Rook: Cloud-native storage orchestration for kubernetes. `https://rook.io`, 2024. Accessed: 2024-06-17.

[37] Longhorn Documentation. Longhorn: Cloud-native distributed block storage for kubernetes. `https://longhorn.io/`, 2024. Accessed: 2024-06-21.

[38] Metallb, a network load-balancer implementation for kubernetes using standard routing protocols. `https://metallb.universe.tf/`. Accessed: 2024-06-16.

[39] PureLB Authors. Purelb: Service load balancer for kubernetes. `https://github.com/purelb/purelb`. Accessed: 2024-06-14.

[40] NGINX, Inc. Nginx: High performance load balancer, web server, & reverse proxy. `https://www.f5.com/products/nginx`. Accessed: 2024-06-14.

[41] Traefik Labs. Traefik: The cloud native edge router. `https://traefik.io/traefik/`. Accessed: 2024-06-14.

[42] cert-manager Authors. cert-manager: Automatically provision and manage tls certificates in kubernetes. `https://github.com/cert-manager/cert-manager`. Accessed: 2024-06-14.

[43] Let's Encrypt. Let's encrypt: Free ssl/tls certificates. `https://letsencrypt.org/`. Accessed: 2024-06-14.

[44] Internet Security Research Group. Automatic certificate management environment (acme). `https://datatracker.ietf.org/doc/html/rfc8555`, 2019. Accessed: 2024-06-20.

[45] Kubernetes SIGs. Externaldns: Publicly accessible dns records managed automatically. `https://github.com/kubernetes-sigs/external-dns`. Accessed: 2024-06-14.

[46] Kubernetes Project. Secrets. `https://kubernetes.io/docs/concepts/configuration/secret/`, 2024. Accessed: 2024-06-20.

[47] Kubernetes Documentation. Encrypting secret data at rest. `https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/`, 2024. Accessed: 2024-06-21.

[48] Bitnami. Sealed secrets. `https://github.com/bitnami-labs/sealed-secrets`. Accessed: 2024-06-14.

[49] Mozilla. Sops: Secrets operations. `https://github.com/mozilla/sops`. Accessed: 2024-06-14.

[50] Prometheus Authors. Prometheus: Monitoring system & time series database. `https://prometheus.io/`. Accessed: 2024-06-14.

[51] Grafana Labs. Grafana: The open observability platform. `https://grafana.com/`. Accessed: 2024-06-14.

[52] Flux CD Documentation. Flux: The gitops family of continuous delivery tools for kubernetes. `https://fluxcd.io/`, 2024. Accessed: 2024-06-21.

[53] Argo cd - declarative gitops cd for kubernetes. `https://argo-cd.readthedocs.io/`. Accessed: 2024-06-14.

[54] Red Hat Documentation. Openshift gitops. `https://www.redhat.com/en/technologies/cloud-computing/`, 2024. Accessed: 2024-06-21.

[55] Rancher Labs. Fleet: Manage large fleets of kubernetes clusters. `https://github.com/rancher/fleet`, 2024. Accessed: 2024-06-21.

[56] Kubernetes Documentation. Web ui (dashboard) for kubernetes. `https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/`, 2024. Accessed: 2024-06-21.

[57] Red Hat Documentation. Openshift data foundation. `https://www.redhat.com/en/technologies/cloud-computing/openshift-data-foundation`, 2024. Accessed: 2024-06-21.

[58] Rancher Labs Documentation. Layer 4 and layer 7 load balancing. `https://ranchermanager.docs.rancher.com/how-to-guides/new-user-guides/kubernetes-resources`, 2024. Accessed: 2024-06-21.

[59] Red Hat Documentation. Cert-manager operator issuer (acme). `https://docs.openshift.com/container-platform/4.15/security/cert_manager_operator/cert-man`, 2024. Accessed: 2024-06-21.

[60] OKD Documentation. Ingress operator. `https://docs.okd.io/4.15/networking/ingress-operator.html`, 2024. Accessed: 2024-06-21.

[61] Rancher Labs Documentation. Load balancer and ingress controller setup. `https://ranchermanager.docs.rancher.com/how-to-guides/new-user-guides/kubernetes-resources`, 2024. Accessed: 2024-06-21.

[62] OKD Documentation. Understanding external dns operator. `https://docs.okd.io/latest/networking/external_dns_operator/understanding-external-dns-ope`, 2024. Accessed: 2024-06-21.

[63] OKD Documentation. Encrypting etcd. `https://docs.okd.io/4.15/security/encrypting-etcd.html`, 2024. Accessed: 2024-06-21.

[64] RKE2 Documentation. Secrets encryption. `https://docs.rke2.io/security/secrets_encryption`, 2024. Accessed: 2024-06-21.

[65] OKD Documentation. Monitoring with prometheus using the operator sdk. `https://docs.okd.io/latest/operators/operator_sdk/osdk-monitoring-prometheus.html#osdk-mon`, 2024. Accessed: 2024-06-21.

[66] Rancher Labs Documentation. Monitoring and alerting in rancher. `https://ranchermanager.docs.rancher.com/integrations-in-rancher/monitoring-and-alerting`, 2024. Accessed: 2024-06-21.

[67] OpenAI. Chatgpt: A large language model. `https://www.openai.com/chatgpt`, 2024. Accessed: 2024-06-20.

[68] Editorial Board. Tools such as chatgpt threaten transparent science; here are our ground rules for their use. *Nature*, 613:612, 2023.

[69] Andrea Gaggioli. Ethics: disclose use of ai in scientific manuscripts. *Nature*, 614:416–418, 2023.

[70] N. Dehouche. Plagiarism in the age of massive generative pre-trained transformers (gpt-3). *Ethics in Science and Environmental Politics*, 21:17–23, 2021.