Kubernetes Cluster Data Center Reference Architecture with Omni and Talos Linux

Executive Summary

This document serves as a foundational reference for deploying **Kubernetes** clusters with **Omni** and **Talos Linux** in the **data center**, ensuring security, performance, and operational excellence.

Kubernetes is the leading container orchestration platform, enabling organizations to efficiently deploy, manage, and scale applications. Omni and Talos Linux bring simplicity and security to bare-metal and data center Kubernetes, making infrastructures secure by default, easier to use, and more reliable to operate.

This data center architecture document provides a blueprint for deploying a Kubernetes cluster using Omni and Talos Linux, outlining best practices for management, security, and disaster recovery to deliver a scalable and resilient platform for cloud native applications. While there are many options for deploying Kubernetes, we recommend the following architecture.

Data center environments are characterized by spaces containing hundreds or thousands of servers in racks, as well as networking, power redundancy, ventilation, cooling, and physical security. In these environments, it is vital to ensure high availability, as services and applications are business-critical. High availability is achieved through a strategic, horizontally scaled design for infrastructure and applications, enabling redundancy and fault-tolerance through replication and removing single points of failure.

Technology Overview

Omni and Talos Linux redefine data center operating systems by replacing the fragile, cobbled-together approach of remote management for traditional Linux distributions and Kubernetes deployments with an optimized and autonomous approach to data center Kubernetes.

Talos Linux

Talos Linux is an open source Linux operating system (OS) purpose-built for Kubernetes and data centers, that operates entirely through an API. Talos Linux eliminates the traditional SSH and shell access, and associated user management burden, thereby providing a highly secure and minimal operating system for Kubernetes clusters.

Key features include:

- **Immutable OS:** Prevents configuration drift and enhances security. Image-based updates simplify upgrades and eliminate patching.
- **API-Only Management:** No SSH or shell access, reducing attack surface. Declarative API prevents configuration drift, and imperative API endpoints provide on-demand information gathering and debugging.
- **Built-in Security:** Inherent security through the implementation of Kernel Self Protection Project standards (KSPP), SELinux, TPM support, disk encryption, SecureBoot, read-only root filesystem, and modern cryptographic standards.
- **Lightweight and Optimized:** Talos Linux is designed specifically to run Kubernetes. It comes with fewer than 50 binaries and no package manager or traditional userland tools included by default. It also provides system extensions and overlays to add optional drivers, services, and hardware support.

Omni

Omni makes managing your Kubernetes clusters predictable and reliable by abstracting away complex Kubernetes functions and replacing them with a simple management solution that provides complete oversight of your deployment. Omni seamlessly delivers data center deployment with simplified remote management, centralized cluster provisioning, secure access, and supported maintenance. Omni supports infrastructure providers to provision nodes on bare metal and Virtual Machines, such as with KubeVirt.

Key features include:

- **Observability:** Simplifies oversight and management through the collation of node service status, kernel logs, usage metrics, and configuration.
- **Purpose-Built for the Data Center:** Unlike general-purpose solutions, Omni and Talos Linux were built from the ground up, with no unnecessary binaries or functionality, allowing for the secure and efficient operation of Kubernetes.
- **Zero-Trust Security Model:** Omni and Talos Linux ensure hardware and software integrity with central authentication, Trusted Boot, TPM disk encryption, integrated KMS disk encryption, an immutable OS, and secure access through a private network with SideroLink, all with no local network firewall configuration required.
- **Automated Remote Operations:** Omni establishes a secure, encrypted tunnel that automatically connects nodes to a central control plane, enabling seamless remote management integrated with RBAC and access policies.
- **Simplified Deployment and Upgrades:** Effortlessly provision compute through boot media, connecting compute from any location. Simple, automated upgrades for easy deployment and preventing configuration drift.
- **Minimal Footprint, Maximum Performance:** Omni and Talos Linux are lightweight, secure, and highly optimized for constrained environments.
- **Automatic Dynamic Compute Management:** Utilize infrastructure with bare metal and Virtual Machine providers to automatically provision machines for use.

Upstream Kubernetes

Talos Linux deploys upstream Kubernetes without API modifications, ensuring full compatibility with the Kubernetes ecosystem. By running pure upstream Kubernetes, Talos Linux provides a reliable, community-aligned foundation for cloud native workloads. This provides:

- **Consistent Behavior:** Conforms to the CNCF Kubernetes conformance program, ensuring your deployment with Talos Linux is consistent with other Kubernetes distributions.
- **Maximum Compatibility:** Works seamlessly with all Kubernetes tooling, APIs, and extensions.

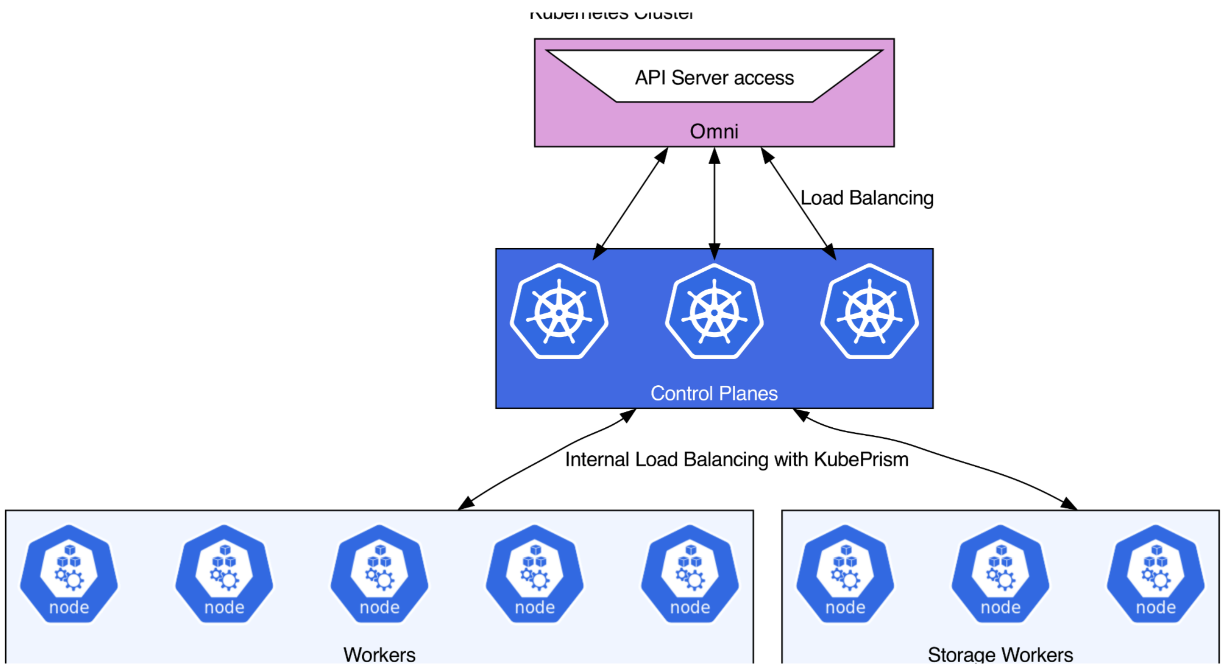
- **Security and Stability:** Avoids vendor lock-in and ensures regular updates and security patches from the Kubernetes community.
- **Predictable Upgrades:** Ensures smooth upgrades without proprietary patches that could introduce unexpected issues.

Solution Overview

This document targets data center enablement for Kubernetes, leveraging Talos Linux as the operating system and Omni for remote management and access.

Cluster Architecture

- Control Plane Nodes: Three virtual or bare metal Talos Linux nodes for the control plane to ensure high availability.
- Worker Nodes: Scalable Talos Linux nodes sized for workload requirements.
- Storage Nodes (optional): Three or more Talos Linux nodes sized for performance and storage requirements, containing HDD, SSD, NVMe, or equivalent drives for storing workload data. Increasing the number of storage nodes improves fault tolerance.



Control Plane Nodes

Control plane nodes host the critical components responsible for managing the cluster's state, coordinating nodes, and providing the API server. It is essential that they are secure, available, and performant.

Kubernetes uses etcd on control plane nodes as a distributed datastore that provides high availability, fault tolerance, and performance.

Best Practices for Provisioning Control Plane Nodes

- Control plane nodes should be sized appropriately for the workloads. In order to determine the best sizing, testing with actual workloads is recommended. A Kubernetes cluster that has few deployments, static node membership, and no additional namespaces will have very different requirements for the control plane nodes compared to a cluster that has a rapid rate of deployments, frequent worker node scaling, many namespaces, and applications that make expensive API requests, even if they have the same number of nodes.
- As a general guideline, control planes should have either:
  - A minimum of 8GB of memory, 4 cores, and 40GB of NVMe disk storage if the cluster has fewer than 100 worker nodes
  - A minimum of 32 GB of memory, 8 cores, and 40GB of NVMe disk storage if the cluster has more than 100 worker nodes

Beyond this, sizing is dependent on the workload and rate of change. We recommend gradually scaling up the workload on the cluster and monitoring the control plane nodes. If resource usage of either memory or CPU exceeds 60% capacity, then increase the CPU or memory resources available to the control plane nodes. This will ensure that the control planes have sufficient capacity to handle resource spikes without compromising the stability of the cluster.

We recommend:

- Three control plane nodes.
  - While it is possible to scale beyond three control plane nodes, doing so will cause replication overhead, as all writes must be replicated to all etcd members. This can degrade performance.
  - A control plane of three nodes can only survive one node failure while maintaining quorum. Thus, it is imperative that effective monitoring is employed to trigger alerts on any failures of control plane nodes.

Increasing the control plane count beyond three nodes can add fault tolerance but comes at the expense of performance.

- Locally attached NVMe drives for control plane storage, as not only the Kubernetes API server components, but also the etcd datastore, need fast disk IO performance.
- Because etcd writes must be replicated amongst all control plane nodes, high-performance/low-latency networking between control plane nodes is essential. We recommend control plane nodes be located within different racks of the same network within the same datacenter to support fault tolerance. Network latency between nodes should be less than 15ms RTT (Round Trip Time).

Note: There are workloads where scaling beyond three control plane nodes can be beneficial for performance, as some reads can target any replica, but for most workloads, adding more etcd nodes decreases practical performance.

Note: Configurations using an even number of control planes are not advised. This is because configurations using two control plane nodes lack etcd fault-tolerance, which can lead to instability and lack of availability, while configurations using four or more provide no additional fault tolerance and increase the likelihood of a node failing (simply by increasing the number of nodes that can fail).

Best Practices for Configuring Control Plane Nodes

These practices are the default configuration for Talos Linux, regardless of whether or not you are using Omni. We mention them to ensure that they are not overridden in deployment unless you have strong reasons to do so.

- Ensure that KubePrism is enabled for all nodes in the cluster. This ensures all worker and control plane processes can access the Kubernetes API, even if the external load balancer for the API Server endpoint is down or unavailable.
- Do not allow workloads to be scheduled on control plane nodes. Doing so can expose the control plane to instability caused by workloads consuming resources unexpectedly and starving the control plane processes. It also reduces security by potentially allowing workload pods to take advantage of vulnerabilities and access secrets on the control plane. By default, Talos Linux taints control plane nodes so they cannot run workload pods.

Networking and CNI

By default, Talos Linux will install the Flannel CNI. Flannel is an appropriate choice for many enterprises as it provides:

- Simple and easy configuration, with less complexity than alternative CNIs, and therefore fewer issues and easier troubleshooting.
- Lower CPU and memory resource consumption compared to Cilium.

Cilium is a supported option and is selected by many enterprises with the following requirements:

- High throughput or low latency, delivered by eBPF-powered packet processing.
- Fine-grained network and security policies that are not possible in Flannel.

It is advised against changing a CNI once it has been deployed. Although possible, changing the CNI can cause major disruption, especially without careful attention to detail. As Talos Linux will install Flannel by default, in order to install Cilium or change the default Flannel configuration, it is necessary to override the machine config to specify that no CNI should be initially installed. To do so, please use the following:

yaml

```yaml
cluster:
  network:
    cni:
      name: none
```

It is then recommended to deploy Cilium using a suitable method from the documentation.

Best Practices for Networking and CNI

KubeSpan provides transparent network encryption between all nodes in a cluster-wide mesh VPN network. KubeSpan is well suited for use cases that involve bursting from one network (e.g. bare metal) to another (e.g. a cloud provider) for extra capacity. Because KubeSpan currently creates a full mesh network, it is not generally recommended for clusters with more than 100 nodes.

KubeSpan uses WireGuard encryption to provide cluster meshing. It is not recommended for high-throughput applications such as storage replication and HPC due to the packet encryption overhead, which reduces total throughput. It is best for general cluster networking.

Storage

While there are many different Kubernetes storage options, and Talos Linux will work with most of them, we generally recommend:

- Longhorn, which is simple to manage, prioritizes data redundancy, exports local filesystem storage as CSI, has configurable cluster PVC backups, and is suitable for general Kubernetes workloads.
- Rook+Ceph for more complex use cases. It scales infinitely but is significantly more complex, requires careful tuning for performance vs. resilience, and is not well-suited for small clusters. If running Rook+Ceph within the workload cluster, we recommend dedicated storage nodes to isolate storage performance impacts from application workloads. For example, operations such as rebalancing among storage nodes can significantly impact network, memory, and CPU.
- OpenEBS Mayastor is a multi-mode storage provider exporting both local filesystem storage and drives as CSI, provides replicated storage for redundancy, has enhanced NVMe support with NVMe-TCP, features low-latency, and is great for general Kubernetes workloads.
- Local Path Provisioner, which provides simple but less resilient storage, linking nodes to PersistentVolumeClaims using Filesystem mode. For ephemeral node-bound tasks, Local Path Provisioner is suitable given its lightweight profile. This provider is particularly useful with ephemeral storage like EphemeralVolumes or KubeVirt CDI.

Alternatively, storage outside of the cluster can be consumed within a cluster, using providers such as:

- nfs-subdir-external-provisioner
- csi-driver-iscsi

Best Practices for Configuring Storage Nodes

We recommend the following:

- Locate worker nodes running a storage provider close to each other, such as on the same switch, to ensure high throughput.
- Do not depend on KubeSpan for storage provider connectivity, due to the performance overhead of the network layer encryption.
- Avoid scheduling regular workloads or control plane components on storage nodes in order to avoid potential disruption. Use node labels or taints and tolerations for application configuration to ensure this. Upon defining workers in Omni cluster templates, set storage node-specific labels (i.e, *node-role.kubernetes.io/storage=''*), allowing for node storage workloads (e.g. Longhorn, Rook+Ceph, or OpenEBS) to be schedulable separately from regular workers. See here.
- For high performance, use drives such as SSDs and NVMe drives.
- Configure Talos Linux as described in the guides specific to each storage provider. Some providers require kernel args (e.g. hugepages), kernel modules (e.g. nvme_tcp), Talos Linux extensions (for features like enabling specific filesystem support), etc in order to provide the best performance.

Load Balancing

We recommend MetalLB, KubeVIP, or Cilium CNI for exposing Services. These solutions provide Layer 2 announcement and/or particularly BGP routing, enabling Kubernetes Service Load Balancing and making Services running inside the cluster accessible to local or global networks, as configured.

Monitoring and Logging

We recommend VictoriaMetrics for data center deployments.

We recommend Grafana for observability and VictoriaLogs or Loki logs for logging.

It should be noted that systems running Talos Linux are compatible with most monitoring solutions. The above are simply monitoring systems that are good choices for Kubernetes infrastructure that we and our customers have had success with.

We recommend configuring Talos Linux to send system logs to a logging server. See the documentation.

Tuning

We recommend setting appropriate performance settings as documented here.

If desired, the Talos Linux dashboard can be disabled, saving ~50MB of RAM. See the following kernel parameter `talos.dashboard.disabled=1`.

Talos Linux Extensions

Talos Linux extensions provide additional functionality beyond the base OS. They are recommended for:

- Situations where the hardware running Talos Linux requires specific firmware, drivers, or services (i.e., CPU drivers, GPU drivers, and NIC firmware).
- Security-related features (e.g. gVisor and Kata-Containers).
- External integrations such as storage or network options (e.g. DRDB, iSCSI, Btrfs, Cloudflared).

To install an extension, a custom Talos Linux installer image must be built with the desired extensions included. Such an image can be produced through the hosted service at factory.talos.dev or from Omni. Aside from configuration, once deployed, the Talos Linux OS cannot be modified without the image being replaced via an upgrade to the custom-built image. Omni automates this process, safely rolling out new image versions as desired.

It is recommended to install microcode extensions for the target bare metal deployment processor, such as intel-ucode or amd-ucode.

More information can be found at:

- Talos Extensions Documentation
- SideroLabs Extensions Repository

Security Considerations

Talos Linux is already highly secure with a default install, but there are additional capabilities that can be enabled. Note that there is overhead with additional security features, whether in operational complexity or node performance.

Omni contains SideroLink, which automatically creates a firewall to the Talos API and eliminates the need to apply local firewall rules to secure the Talos API before the node is provisioned.

TrustedBoot with SecureBoot and Disk Encryption

Talos Linux supports TrustedBoot in order to protect against boot-level malware attacks. Talos Linux implements a fully signed execution path from firmware to userspace. When used in conjunction with disk encryption backed by hardware TPM or Omni's KMS, this provides very strong protection and guarantees that only trusted Talos Linux can run, even when an attacker has physical access to the server.

SecureBoot does complicate OS installation and upgrades, and in all cases, careful consideration must be taken into account. Please refer to the hardware manufacturer's documentation, especially for devices like graphics cards.

Ingress Firewall

The Talos Linux Ingress Firewall provides an additional security layer by controlling inbound network traffic at the OS level. The Ingress Firewall defaults to "allow" for all traffic. When using Talos Linux through Omni, the Talos Linux API is only accessible through the SideroLink tunnel. All other access is blocked by Ingress Firewall rules. For a production setup, we recommend blocking all traffic not explicitly permitted and only allowing traffic explicitly to/from classes of machines, such as control plane nodes and workers. This may be appropriate where:

- Kubernetes workloads are exposed to the internet or untrusted devices.
- Regulatory compliance demands OS level ingress controls and security frameworks beyond Kubernetes-native NetworkPolicies.
- Zero-trust environments are needed to complement Kubernetes RBAC and service mesh policies by filtering traffic at the host level.

For an example of recommended rules, see the documentation.

OS and Kubernetes Authentication

The Kubernetes API Server supports configuring the use of external auth providers, meaning restriction based on authenticated and authorized users and identities for cluster access. It is recommended to adopt such a mechanism for production clusters. Any OAuth or OIDC provider is supported (such as Auth0), with varying levels of supporting configuration and authentication proxies, simplifying compliance and eliminating the risk of an employee leaving the enterprise with admin-level kubeconfig access for a cluster.

PodSecurity Standards

Talos Linux defaults to the baseline PodSecurity Standards enabled. This means that workloads cannot run with privileged security contexts, such as root user, node host network access, hostPath, and privileged.

Some components required in data center environments may require privileged access to engage with hardware devices, network, and local storage. We recommend placing privileged workloads in their own namespace to limit privileged Pod SecurityContext. For more information, see here.

Certificates and Secrets

Talos Linux uses secrets, including cluster CA and system CA, in the machine configuration to declare the cluster.

Omni completely manages cluster secrets, ensuring they are obscured and never revealed.

Cluster Upgrades

Omni simplifies cluster upgrades. Whether using the web interface or cluster templates, Omni safely reconciles cluster changes such as Talos Linux and Kubernetes versions, configuration patches, and node management declaratively by utilizing Talos Linux and Kubernetes APIs and health checks.

As cluster configuration templates are declarative and contain no secrets, administrators can utilize Git to keep cluster changes version-controlled.
Omni carefully handles all upgrade paths between older and latest versions, ensuring migration to the latest versions is at a suitable pace for organizational needs.

Since Talos Linux is image-based and uses an A/B boot system, upgrades between release versions of the OS will either succeed and boot into the new release or fail and revert back to the previous version, ensuring broken states never occur. Talos Linux upgrades are initiated on a per-node basis. The upgrade process will cordon the node in Kubernetes, then reboot into the new image. It is up to the user to ensure all nodes in a cluster are upgraded consistently and follow the recommended upgrade paths. When using Omni, Talos Linux upgrades are orchestrated on a cluster level, not a node level. Omni handles a wider range of upgrade paths, and any upgrade allowed in the UI is supported.

Kubernetes upgrades are managed separately from Talos Linux upgrades. During a Kubernetes upgrade, Kubernetes is upgraded by replacing components in the running cluster with new versions and migrating any related resources. When running in a highly available configuration, Kubernetes upgrades are non-disruptive and do not require a reboot.

We recommend tuning UpdateStrategy for Worker nodes if there are specific requirements for update rollout, see here.

Cluster Reproducibility

Deploying Kubernetes with Talos Linux starts with configuration. The machine configuration is fully declarative, enabling users to describe everything about the machine and its cluster components. Clusters can be reliably restored by applying the configuration files for each machine to the same number and types of machines, with the appropriate per-machine patches and restoring etcd, making redeployments and disaster recovery straightforward and consistent. Depending on the particular deployment, more steps may be required.

Omni further simplifies the configuration process by creating declarative configurations of clusters as a whole, not just a machine, allowing the creation, templating, modification, and recreation of clusters with declarative cluster templates that handle all cluster, machine, secret, and patch management, while never exposing the cluster secrets.

Application Management

We recommend ArgoCD for declarative GitOps-based management of applications, where one or more git repositories act as the source of truth for what is deployed on one or more clusters.

We recommend using Omni cluster templates for the initial configuration and deployment of ArgoCD (as demonstrated here) and then using Argo to manage the applications.

Multi-tenancy

When running in the data center, multi-tenancy enables sharing resources amongst teams or customers on bare metal compute and leading to higher resource usage efficiency. In order to share resources without impacting security or other workloads, some considerations must be taken into account, such as API access controls and runtime isolation. See the following:

- **Shared clusters with Omni:** By using user accounts with Omni, cluster access is able to be provided for users to clusters, ensuring that access is only granted for specific users and no hardcoded credentials are ever used and clusters can be shared between teams.
- **ArgoCD:** Teams are able to utilize GitOps through ArgoCD to approve deployments and changes, ensuring cohesiveness to multi-tenancy.
- **RuntimeClasses:** By using a Talos Linux-supported RuntimeClass such as gVisor or Kata-Containers, with system extensions managed through Omni, teams are able to trust that their application execution is protected and secure.
- **KubeVirt + Omni:** Use Omni and KubeVirt Infrastructure Provider to provision virtualized Kubernetes clusters to partition bare metal cluster resources and create workload clusters for application administrators to consume. Virtual machines provide tighter security for workloads, allowing for Talos Linux machines to be provisioned on a bare metal cluster inside KubeVirt, thereby isolating a tenant's workloads.

Multi-Cluster

Given multiple clusters in multiple data centers, it is often important to bridge cluster networking, enabling replication and failover of global services like databases and load balancers. We recommend applying multi-cluster techniques in data center environments with Omni-managed clusters to ensure high availability. Consider the following:

- **Istio Multi-Cluster**: Securely route Kubernetes Services between multiple clusters using either multi-primary or primary-remote configuration.
- **Cilium ClusterMesh:** Securely route Pod IPs and Kubernetes Services discovery across multiple clusters.
- **ArgoCD:** Orchestrate workloads across multiple clusters.

Additional Software

The Kubernetes ecosystem contains many options for other functions that may or may not be desired for any particular deployment. Additional Kubernetes software can be used to address certain needs, such as container security and compliance, namespace controls, policy compliance and governance, CI/CD tooling, secret management, and more.

This reference architecture document does not express opinions on these functions. However, because Talos Linux deploys vanilla upstream Kubernetes, such clusters are compatible with virtually any of the options enterprises may be using for these functions. For enterprises that would like specific recommendations for their use cases, either Sidero Labs or one of our consulting partners can be engaged for consultation.

NUT Client Extension

When deploying in the data center, a UPS is used to ensure the integrity of stateful data and hardware longevity. The NUT client extension allows for Talos Linux shutdown to be triggered by compatible hardware when reserve power is nearly depleted.

See the following documentation:

- Sidero Labs Extensions - nut-client
- Network UPS Tools - nut

GPU drivers

NVIDIA GPUs are supported on Talos Linux through the NVIDIA driver extensions. Open or closed driver variants are both available with Talos Linux extensions. To enable it, the kernel modules and a sysctl must be configured.

For other accelerators, see here. For more information, please see the following documentation.

Virtual Machines

KubeVirt on Talos Linux is a proven, reliable way to migrate legacy workloads from Virtual Machines into Kubernetes for application deployments to run alongside containerized applications, and is also capable of ensuring the trusted execution of processes within a virtualized environment. Virtual Machines provide tighter isolation than regular containers and are suitable for running untrusted workloads or applications with tight restrictions on their environment.

With KubeVirt, Virtual Machines can talk to Pods, and vice versa, and can also be exposed like regular Pods through Services, Ingress, Gateway-API, and more. Retaining virtual machines and running them on Kubernetes is a helpful way to balance the needs of your organization and to ease into the process of modernizing. To effectively run Virtual Machines in KubeVirt, it is best to have:

- A CSI provider that supports LiveMigration, such as Longhorn
- A CNI plugin to provide attachment of multiple network interfaces, such as KubeOVN or Multus, which wraps an existing CNI (e.g. Flannel, Cilium)

With its declarative configuration, workloads running through KubeVirt can be managed through GitOps like ArgoCD. Running Virtual Machines on Talos Linux through KubeVirt is suitable for bare metal, datacenter, and edge deployments.

Further Configuration

There are a myriad of choices available when setting up Kubernetes clusters. While common best practices work in some scenarios, they may not be suitable for others.

Sidero Professional Services can work with you to architect a configuration tailored to your requirements.

Contact us

If you have questions or want to discuss how to get started with Omni and Talos Linux for Kubernetes at the edge, contact us.