Kubernetes Cluster Edge Reference Architecture with Omni and Talos Linux

Executive Summary

This document serves as a foundational reference for deploying **edge Kubernetes** clusters with **Omni** and **Talos Linux**, 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 edge Kubernetes, making infrastructures secure by default, easier to use, and more reliable to operate.

This edge architecture document provides a blueprint for deploying a standard 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, this is our recommended architecture.

Edge compute environments are defined by close locality to sources of data, or for data distribution, and architecturally it is a form of distributed computing. In these environments, there may be restrictions on compute, networking, and power. Earthly environmental conditions may also apply such as temperature and humidity. Specialized hardware and software may also be used to supplement data collection, data distribution, or varying restrictions and conditions.

Technology Overview

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

Talos Linux

Talos Linux is an open source Linux operating system (OS) purpose-built for Kubernetes and edge deployments, that operates entirely through an API, eliminating traditional SSH and shell access and 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 eliminates patching.
- **API-Only Management:** No SSH or shell access, reducing attack surfaces. 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, boot from an in-memory SquashFS file system, 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 collapsing the overly complex Kubernetes functions into a merged infrastructure with operating system management. Omni seamlessly delivers edge deployment with simplified remote management, centralized cluster provisioning, secure access, and supported maintenance.

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 Edge:** Unlike general-purpose solutions, Omni and Talos Linux were built from the ground up to operate Kubernetes securely and efficiently.
- **Zero-Trust Security Model:** Omni and Talos Linux ensure hardware and software integrity with central authentication, Trusted Boot, TPM 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 edge nodes to a central control plane, enabling seamless remote management with RBAC and access policies.
- **Simplified Deployment and Upgrades:** Effortless provision compute through booth media, connecting compute from any location running Talos Linux. Simple, automated upgrades for easy deployment and prevents configuration drift.
- **Minimal Footprint, Maximum Performance:** Omni and Talos Linux are lightweight, secure, and highly optimized for constrained environments.

Upstream Kubernetes

Talos Linux deploys upstream Kubernetes without API modifications and ensures 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 Kubernetes distribution under 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

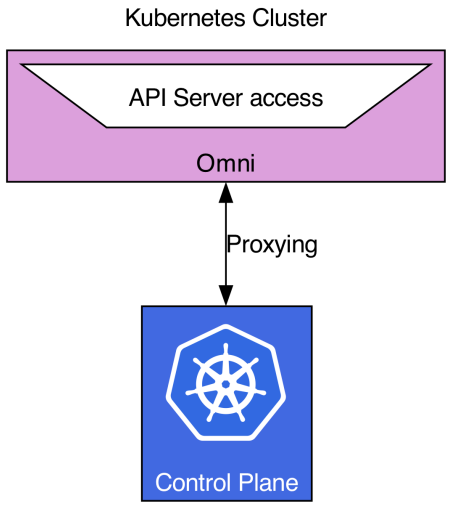
This document targets edge enablement for Kubernetes, leveraging Talos Linux as the operating system and Omni for remote management and access to leverage limited compute capacity and edge limitations.

Cluster Architecture

There are two suitable configurations for using Talos Linux and Omni:

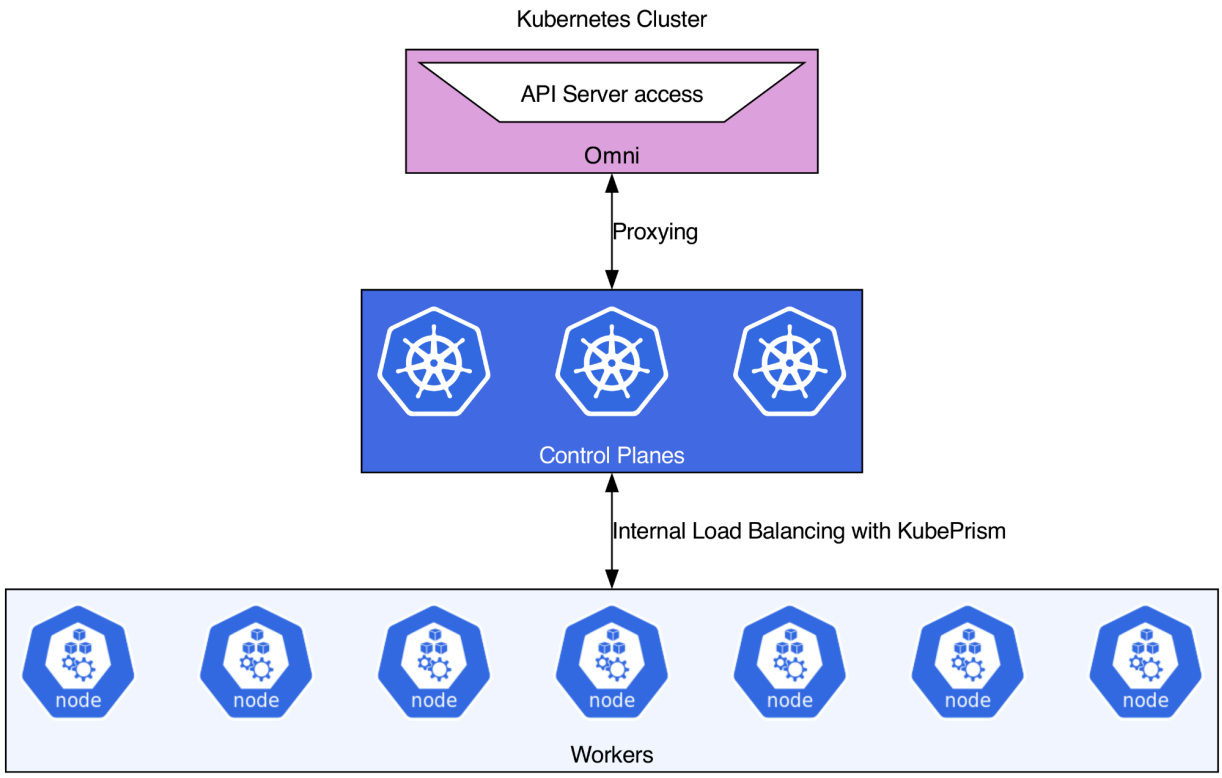
Single Node Configuration

- Control Plane Nodes: One node configured to allow the scheduling of workloads
- Configured without high availability



Multi-node control plane

- Control Plane Nodes: One or three Talos Linux nodes for the control plane. If only one control plane node is deployed, note that the control plane will not be highly available, although it can support multiple worker nodes for worker node fault tolerance. In the event of the failure of the control plane (which is more likely with a single control plane node), workers can continue to run their workloads temporarily but cannot schedule or manage new workloads.
- Worker Nodes: Scalable Talos Linux nodes sized for workload requirements.



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 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 disk storage if the cluster has fewer than 100 worker nodes
    - A minimum of 32 GB of memory, 8 cores, and 40GB of disk storage if the cluster has more than 100 worker nodes

Beyond this, sizing is dependent on the workload. 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.

For Highly Available Configurations:

- Use three control plane nodes.
    - While it is possible to scale beyond three control plane nodes, increasing the number of nodes beyond three can 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 as comes at the expense of performance.
- Because control planes run not only the Kubernetes API server components but also the etcd datastore, it is important to ensure they have a fast disk IO performance. We recommend locally attached NVMe drives.
- Because etcd writes must be replicated among 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 supported. This is because configurations using two control plane nodes lack etcd fault-tolerance, which can lead to instability and lack of high availability, while configurations using four or more provide no additional fault tolerance and increases the number of nodes that can fail. Kubernetes documentation also recommends at least three control plane nodes.

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 a strong reason 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 API Server load balancer for the endpoint is down or unavailable.

For Highly Available Configurations:

- 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.
- For a single-node configuration, control planes must allow workloads to be scheduled on them.
- When deploying with high availability, we recommend using Talos Linux Virtual (Shared) IP for Kubernetes control plane access. However, this isn't suitable for high external load on the Kubernetes API Server, as only one control plane node will receive requests at any given time, and it is not necessary for single-node clusters.

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 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:

- Clusters requiring 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 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
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

For Single-Node Deployments:

Disable KubeSpan. It is intended for connecting nodes in different networks together and is not useful in this configuration

For Highly Available Configurations:

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 recommended for clusters with more than 100 nodes.

KubeSpan uses Wireguard VPN tunneling to provide cluster meshing. It is not recommended for high-throughput applications such as storage replication and HPC due to packet encryption overhead, which reduces total throughput and 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.
- 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. For many use cases, Local Path Provisioner may be sufficient for a deployment given its lightweight profile.

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

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

Load Balancing

While load balancing is not required in single-node clusters, we recommend MetalLB or KubeVIP for most use cases to make exposing Services more straightforward.

With one of the above solutions, load balancing can be configured with either ARP or BGP.

Monitoring and Logging

We recommend Prometheus for small or few clusters and VictoriaMetrics for more complicated or large-scale 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. For more information, see the documentation.

Tuning

If performance is more important than minimizing power consumption, we recommend setting appropriate performance settings, as documented in the latest performance tuning documentation.

If desired, 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. 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. 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 deployment processor, such as intel-ucode or amd-ucode.

More information can be found at:

- Talos Extensions Documentation
- SideroLabs Extensions Repository

Talos Linux Overlays

Often in edge deployments, ARM devices or embedded hardware are utilized. In order to maximize or enable compatibility, the Talos Linux factory and imager provide overlays for firmware, bootloader, and device tree blobs.

Overlays include support for many devices, such as:

- Raspberry Pi

- Orange Pi
- Rock64
- Jetson Nano
- Pine64
- And many more

See official overlays here.

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 for all cases, careful consideration must be taken into account.

We recommend using TrustedBoot for edge deployments.

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.

- 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 edge 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.

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.

GitOps is incredibly important for remote deployments, where access to the cluster is only possible through GitOps.

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.

Device Plugins

Accessing specialized or specific hardware such as cameras, robotics equipment, or other USB and serial devices via Linux device filesystem on Talos Linux is managed in a non-privileged way using the generic device driver plugin, if not a specific plugin driver for Kubernetes. For details, see the documentation here.

NUT Client Extension

When deploying on the edge with a UPS and 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 more information, please see the following documentation.

Progressive migration from Virtual Machines

KubeVirt on Talos Linux is a proven, reliable way to migrate legacy workloads from Virtual Machines into Kubernetes for edge deployments to run alongside containerized applications, and is also capable of ensuring the trusted execution of processes within a virtualized 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. Import your existing virtualized workloads using tooling like Forklift from providers like VMware vSphere, OVA, oVirt, and OpenStack.

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.