Kubernetes Cluster Reference Architecture with Talos Linux

Executive Summary

This document serves as a foundational reference for deploying highly available **Kubernetes** clusters with **Talos Linux**, ensuring security, performance, and operational excellence.

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

This architecture document provides a blueprint for deploying a standard Kubernetes cluster using Talos Linux, outlining best practices for management, security, high availability, and disaster recovery to deliver a scalable and resilient platform for cloud native applications. While there are many options for deploying functions in Kubernetes, this is our recommended architecture.

Technology Overview

Talos Linux

Talos Linux is an open source Linux operating system (OS) purpose-built for Kubernetes, operates entirely through an API, eliminating traditional no SSH or shell access, thus 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 surfaces. Declarative API prevents configuration drift.

- **Built-in Security:** Inherent security through the implementation of Kernel Self Protection Project standards, SELinux, TPM support, disk encryption, SecureBoot, read-only root filesystem, boot from memory SquashFS file system, and modern cryptographic standards.
- **Lightweight and Optimized:** No package manager or traditional userland tools available or supported. All compute resources are available and dedicated to Kubernetes workloads. Separately Talos Linux only contains 20 unique binaries and only requires the use of extensions and overlays to add additional functionality or support.

Vanilla Upstream Kubernetes

Talos Linux deploys upstream Kubernetes without modifications and ensures full compatibility with the Kubernetes ecosystem. This provides:

- **Consistent Behavior:** It is a conformant Kubernetes distribution under the CNCF Kubernetes conformance program, meaning that Kubernetes is the same as every other distribution.
- **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.

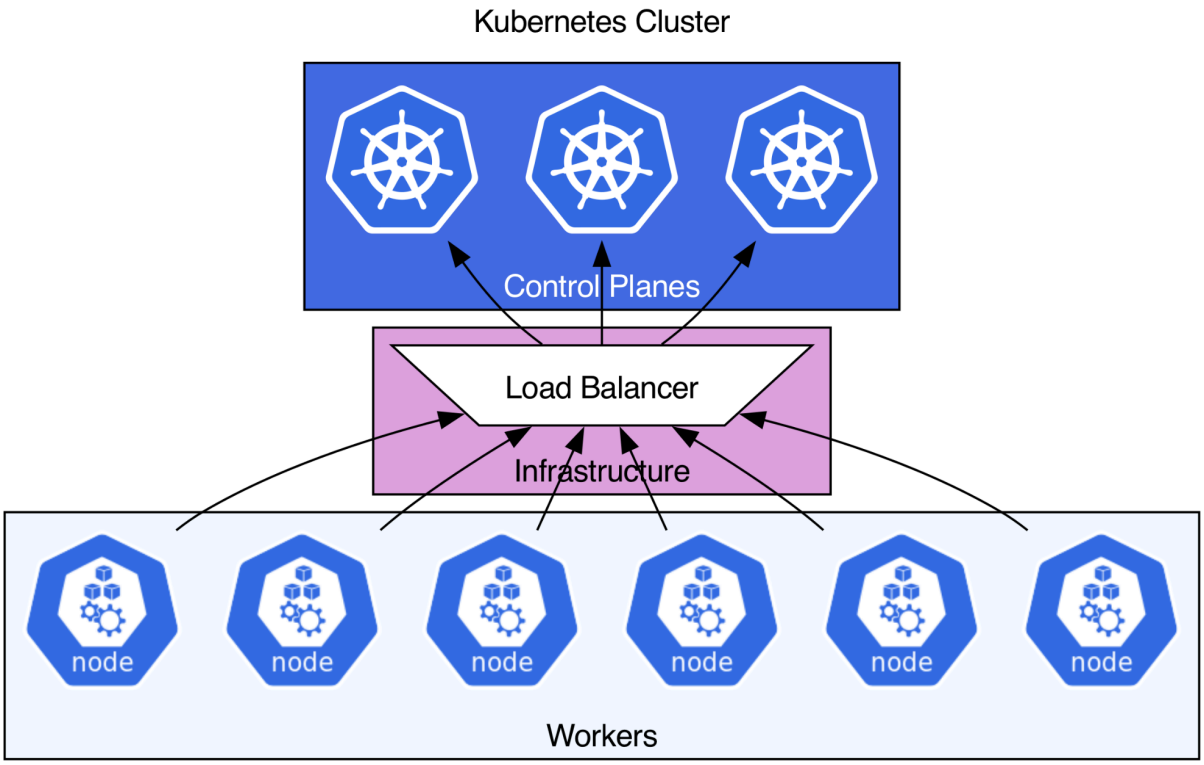
By running pure upstream Kubernetes, Talos Linux provides a reliable, community-aligned foundation for cloud native workloads.

Solution Overview

This reference architecture document targets high availability (HA) for Kubernetes, leveraging Talos Linux as the OS. Other architectures are possible with Talos Linux (including single node clusters, clusters that allow workload scheduling on control planes, and even clusters that span datacenters), but this document focuses on a standard HA cluster with dedicated control plane nodes.

Cluster Architecture

- Control Plane Nodes: Minimum and recommended number of three Talos-based nodes for HA.
- Worker Nodes: Scalable Talos-based nodes sized for workload requirements.
- Networking: Cilium, Flannel, or Calico based on deployment needs.
- Storage: CSI-compatible storage solutions such as Longhorn or Ceph.



Control Plane Nodes

Control plane nodes host the critical components responsible for managing the cluster's state, coordinating nodes, and providing the API server for interaction. As such it is essential that they are secured, 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

- 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 noted comes at the expense of performance.
    - *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*.
- 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 (for fault tolerance) of the same network within the same datacenter. Network latency between nodes should be less than 15ms RTT (Round Trip Time).
- Control plane nodes should be sized appropriately for the workloads. Unfortunately, what is “appropriate” often cannot be specified without testing on the actual workloads. 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.
- Some general guidelines are that control planes should have either:
    - A minimum of 8GB of memory and 4 cores

    - 32 GB of memory and 8 cores if the cluster has 100 worker nodes and a relatively static workload

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 resource 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.

Best Practices for Configuring Control Plane Nodes

These practices are the default configuration on Talos Linux. We mention them to ensure that they are not overridden in deployment.

- 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. Note that by default, Talos Linux taints control plane nodes so they cannot run workload pods.
- Use an external load balancer that distributes requests to all healthy control plane nodes for the Kubernetes API server access. Do not use the built-in Talos Linux Virtual IP functionality for clusters with high external load on the Kubernetes API, as only one control plane node will receive requests at any time.
- 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 endpoint is down or unavailable.

Networking and CNI

By default, Talos Linux will install the Flannel CNI.

Flannel is an appropriate choice for many enterprises:

- Simple and easy to configure, with less complexity (which results in less issues and easier troubleshooting)
- Lower CPU and memory resource consumption compared to Cilium

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

- Large clusters or 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 after it's deployed. Although possible, changing CNI can cause major disruption especially without delicate attention to detail. Talos Linux will install Flannel by default. In order to install Cilium it is necessary to override the machine config to specify that no CNI should be initially installed, see the following:

Create a `patch.yaml` file with the following contents:

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

It is then recommended to deploy Cilium using one of these four documented methods.

Storage

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

- Longhorn for simpler use cases. It is simple to manage, prioritizes data redundancy, exports local filesystem storage as CSI, has configurable cluster PVC backups, and is great for general Kubernetes workloads.
- Rook+Ceph for more complex use cases. It scales infinitely but is significantly more complex, pools full local disks and exports as CSI, 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 the workloads (for example, when rebalancing among storage nodes, network, memory and CPU can all be significantly impacted.).

Load Balancing

In IaaS (Infrastructure as a Service) environments (AWS, GCP, Azure, etc), we recommend use of the cloud provider's native load balancing services.

In bare metal environments, we recommend MetalLB or KubeVIP for most use cases.

Monitoring and Logging

We recommend Prometheus for small or few clusters and VictoriaMetrics for more complicated or large-scale deployments.

We recommend Grafana and Loki for observability and logging.

It should be noted that systems running Talos Linux are compatible with monitoring from 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. It does not preclude the use of other systems.

Tuning

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

Talos Linux Extensions

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

- Situations where the hardware running Talos 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 bundled in it. Such an image can suitably be produced either through the hosted service at factory.talos.dev. Meaning that aside from configuration, the Talos OS deployed cannot be modified without the image being replaced via an upgrade to the custom built image.

More information can be found at:

- Talos Extensions Documentation
- SideroLabs Extensions Repository

Security Considerations

Talos Linux is 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.

SecureBoot and Disk Encryption with TPM support

Talos Linux supports SecureBoot 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 the TPM based disk encryption that Talos Linux supports, this provides very strong protection and guarantees that only trusted Talos Linux can run, even given an attacker with physical access to the server.

SecureBoot does complicate OS installation and upgrades, so is not recommended for all cases and requires specially taking it into account.

Ingress Firewall

The Talos 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. 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.

KubeSpan

KubeSpan provides transparent wire-level network encryption between all nodes in a cluster and simplifies network management by joining a private and secure 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 greater than 100 nodes.

OS and Kubernetes Authentication

The Kubernetes APIServer supports configuring the use of an external authorization provider, this meaning for only authenticated and authorized users to access a given cluster. It is highly recommended to adopt such a mechanism for production clusters. Any OAuth or OIDC provider is supported, albeit with varying levels of supporting configuration and authentication proxies. This eliminates the risk of an employee leaving the enterprise with admin-level kubeconfig access for a cluster, and simplified compliance.

PodSecurity Standards

By default and since Kubernetes v1.23, Talos Linux has baseline PodSecurity Standards enabled. Meaning that workloads cannot run with privileged SecurityContexts, such as: root user, node host network access, hostPath, and privileged.

For more information, see here.

Cluster Upgrades

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 one, never causing a broken state.

Similarly, Kubernetes upgrades are managed separately to Talos upgrades, where a version of Kubernetes is upgraded via replacing components in the running cluster with new versions and migrating any related resources.

Cluster Reproducibility

Bringing up a Kubernetes cluster with Talos Linux starts with configuration. As the cluster configuration is declarative, it describes everything. Given some existing configuration, a cluster can simply be redeployed.

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.

Further Configuration

There are ever expansive 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 that is tailored to your deployment.

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.

Progressive migration from Virtual Machines

KubeVirt on Talos Linux is a proven reliable way to migrate legacy workloads from Virtual Machines into Kubernetes to run alongside containerized applications. With KubeVirt, Virtual Machines are able to 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 modernizing. Import your existing virtualized workloads using tooling like Forklift from providers like VMware vSphere, OVA, oVirt, and OpenStack.

To effectively running Virtual Machines in KubeVirt, it is best to have the following:

- a CSI provider which 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 are able to be managed through GitOps like ArgoCD. Running Virtual Machines on Talos Linux through KubeVirt is suitable for bare metal, datacenter, and edge deployments.

Contact us

We would love to hear from you, how you are using Talos Linux, and to support your deployments.

Please find contact details here at siderolabs.com/contact.