Overview of Kubernetes Addons

An Overview of Kubernetes Addons and Their Role in My Cluster.

---

1. What Are Kubernetes Addons

Kubernetes addons are a collection of utilities that make life easier for cluster operators. They often include networking plugins (CNI), metrics servers, and other tools to enhance cluster functionality. Most cloud providers ship clusters with these essentials by default.

For example, EKS provide the followings:

- vpc-cni
- coredns
- metrics-server

And a real production environment usually requires more than the defaults: components for ingress, observability, storage, and automation are often added to support day-to-day workloads.

2. Core Infrastructure

Traefik and **Traefik CRDs** serve as the main ingress controller in the cluster. Instead of using traditional Ingress objects, I adopt the experimental Kubernetes Gateway API, which provides a more expressive and extensible model for routing. This setup allows both L7 HTTP traffic and raw TCP streams to be aggregated under the same `LoadBalancer` service — a single entry point managed by AWS Load Balancer Controller, which automatically provisions the corresponding AWS NLB resources.

For observability, I use kube-prometheus-stack, a comprehensive monitoring suite combining Prometheus, Alertmanager, and Grafana. It's a straightforward, production-ready solution for cluster operations, offering metrics, alerting, and dashboards out of the box.

When bootstrapping a new cluster, my FluxCD reconciles prometheus first, before any workload or operator depending on CRDs such as `ServiceMonitor`. This ensures that custom monitoring resources are recognized and registered properly during subsequent deployments.

3. Operational Utilities

A common pitfall when starting out is treating the CNCF landscape like a supermarket: grabbing every shiny graduated project and asking 'do I need this?'

In practice, most of them end up unused. After several iterations, only a few utilities remain truly valuable in daily operations. One such tool is Headlamp a lightweight Kubernetes dashboard that integrates smoothly with RBAC.

4. GitOps Controller

FluxCD is the backbone of my operational workflow and usually the first component I deploy after the pod network is up. It continuously reconciles manifests from the Git repository, ensuring the cluster state always matches the declared configuration.

Here's an example layout from my repo:

```
.
|-- README.md
|-- base
|   `-- route
|       |-- httpRoute
|       `-- ssh
|-- clusters
|   `-- master
|       |-- flux-system
|       `-- overlays.yaml
|-- helm
|   |-- aws-load-balancer-controller
|   |   |-- helmRelease.yaml
|   |   `-- kustomization.yaml
|   |-- cert-manager
|   |   |-- README.md
|   |   |-- helmRelease.yaml
|   |   `-- kustomization.yaml
|   |-- cloudnative-pg
|   |   |-- helmRelease.yaml
|   |   `-- kustomization.yaml
|   |-- csi-driver-nfs
|   |   |-- helmRelease.yaml
|   |   `-- kustomization.yaml
|   |-- gitlab
|   |   |-- helmRelease.yaml
|   |   `-- kustomization.yaml
|   |-- headlamp
|   |   |-- helmRelease.yaml
|   |   `-- kustomization.yaml
|   |-- kube-prometheus-stack
|   |   |-- README.md
|   |   |-- helmRelease.yaml
|   |   `-- kustomization.yaml
|   |-- traefik
|   |   |-- helmRelease.yaml
|   |   `-- kustomization.yaml
|   `-- traefik-crds
```

```
|       |-- helmRelease.yaml
|       `-- kustomization.yaml
|-- helm-repo
|   |-- cnpg
|   |   |-- helmRepository.yaml
|   |   `-- kustomization.yaml
|   |-- csi-driver-nfs
|   |   |-- helmRepository.yaml
|   |   `-- kustomization.yaml
|   |-- eks
|   |   |-- helmRepository.yaml
|   |   `-- kustomization.yaml
|   |-- headlamp
|   |   |-- helmRepository.yaml
|   |   `-- kustomization.yaml
|   |-- jetstack
|   |   |-- helmRepository.yaml
|   |   `-- kustomization.yaml
|   |-- prometheus-community
|   |   |-- helmRepository.yaml
|   |   `-- kustomization.yaml
|   `-- traefik
|       |-- helmRepository.yaml
|       `-- kustomization.yaml
`-- overlays
    |-- master-infra
    |   |-- headlamp
    |   |-- kustomization.yaml
    |   `-- route
    |-- master-infra-cert
    |   |-- cert
    |   `-- kustomization.yaml
    |-- master-infra-pre
    |   |-- cert-manager
    |   |-- csi-driver-nfs
    |   |-- kustomization.yaml
    |   |-- namespace.yaml
    |   |-- traefik
    |   `-- traefik-crds
    |-- master-kube
    |   |-- aws-load-balancer-controller
    |   `-- kustomization.yaml
    |-- master-mon
    |   |-- kustomization.yaml
    |   `-- route
```

```
|-- master-mon-pre
|   |-- kube-prometheus-stack
|   |-- kustomization.yaml
|   `-- namespace.yaml
|-- master-pgsql
|   |-- kustomization.yaml
|   `-- pgsql-cluster
`-- master-pgsql-pre
    |-- cloudnative-pg
    |-- kustomization.yaml
    `-- namespace.yaml
```

Another advantage of GitOps is that my colleagues won't fear taking over my responsibilities once I leave the company.

Everything is declarative and versioned. Anyone with access to the repository can understand how the cluster is built simply by reading through the git commits.