Use Built-in Tools to Monitor Kubernetes Applications

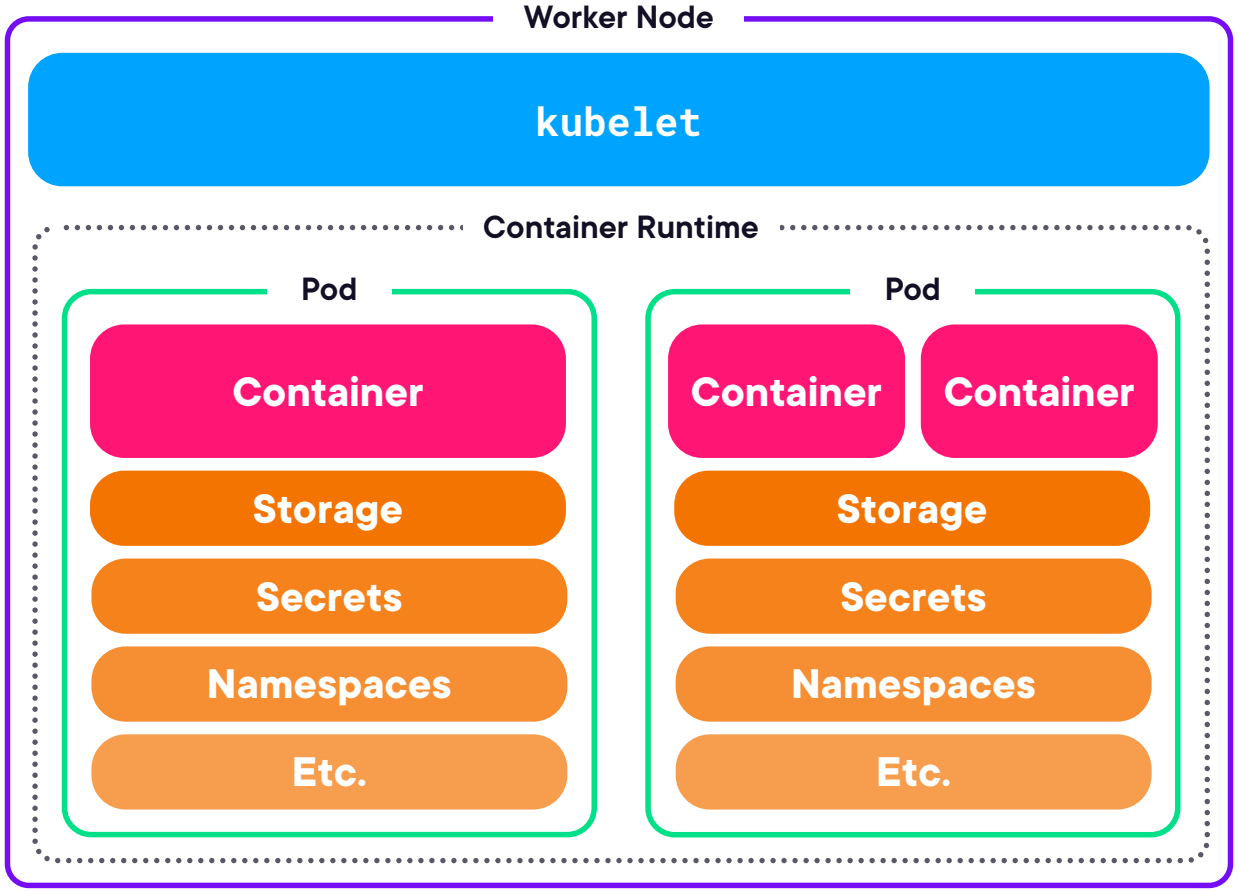
Elle Krout
Principal Course Author, Pluralsight

Monitoring in Kubernetes

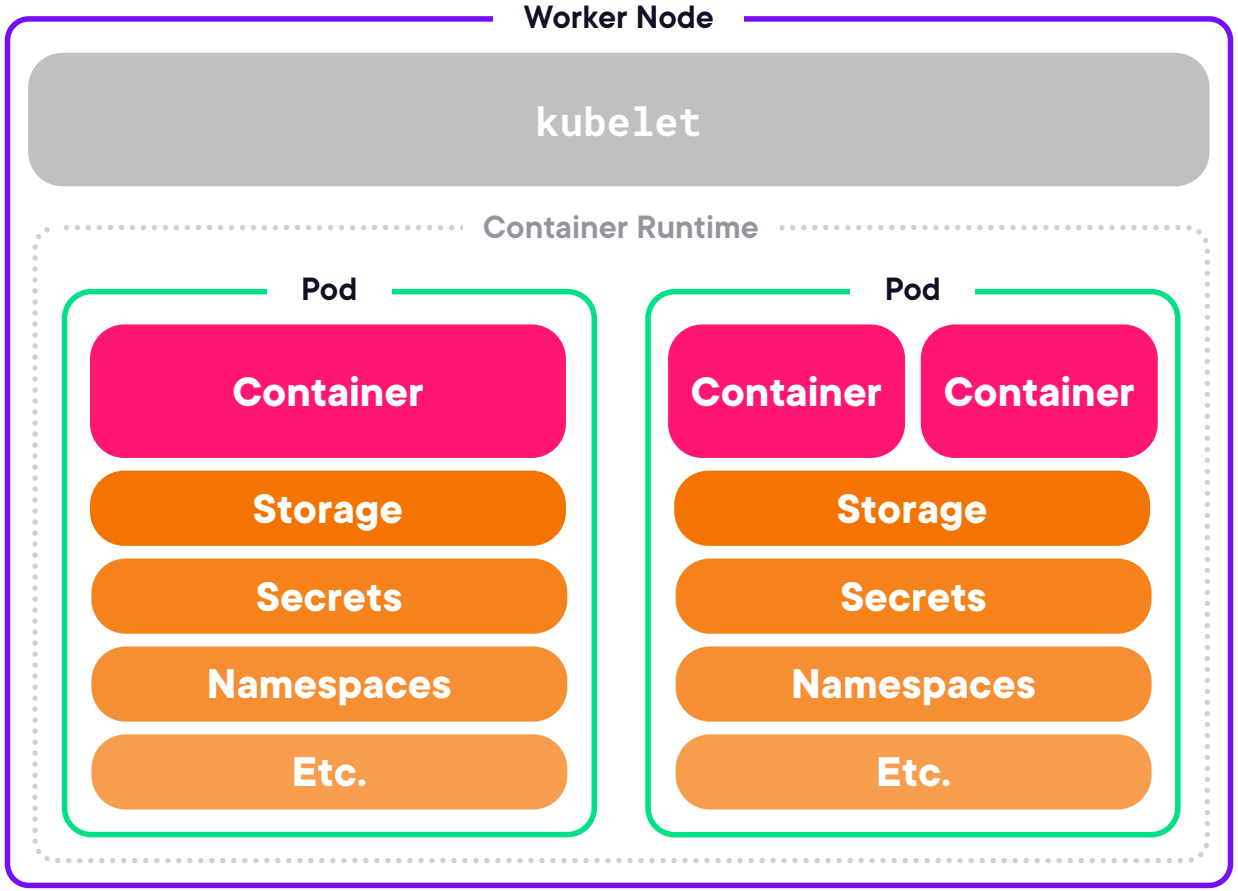
Monitoring Benefits

- Operational Insights
- Performance Tuning
- Troubleshooting Assistance
- Scaling Support
- Data-driven Decisions

What Should You Monitor?



What Should You Monitor?



CLI-only monitoring

No outside monitoring solutions

Metrics Server

Metrics API

Metrics Server is a Kubernetes-provided in-cluster data aggregator that collects resource metrics and exposes them to the API server through the use of the Metrics API.

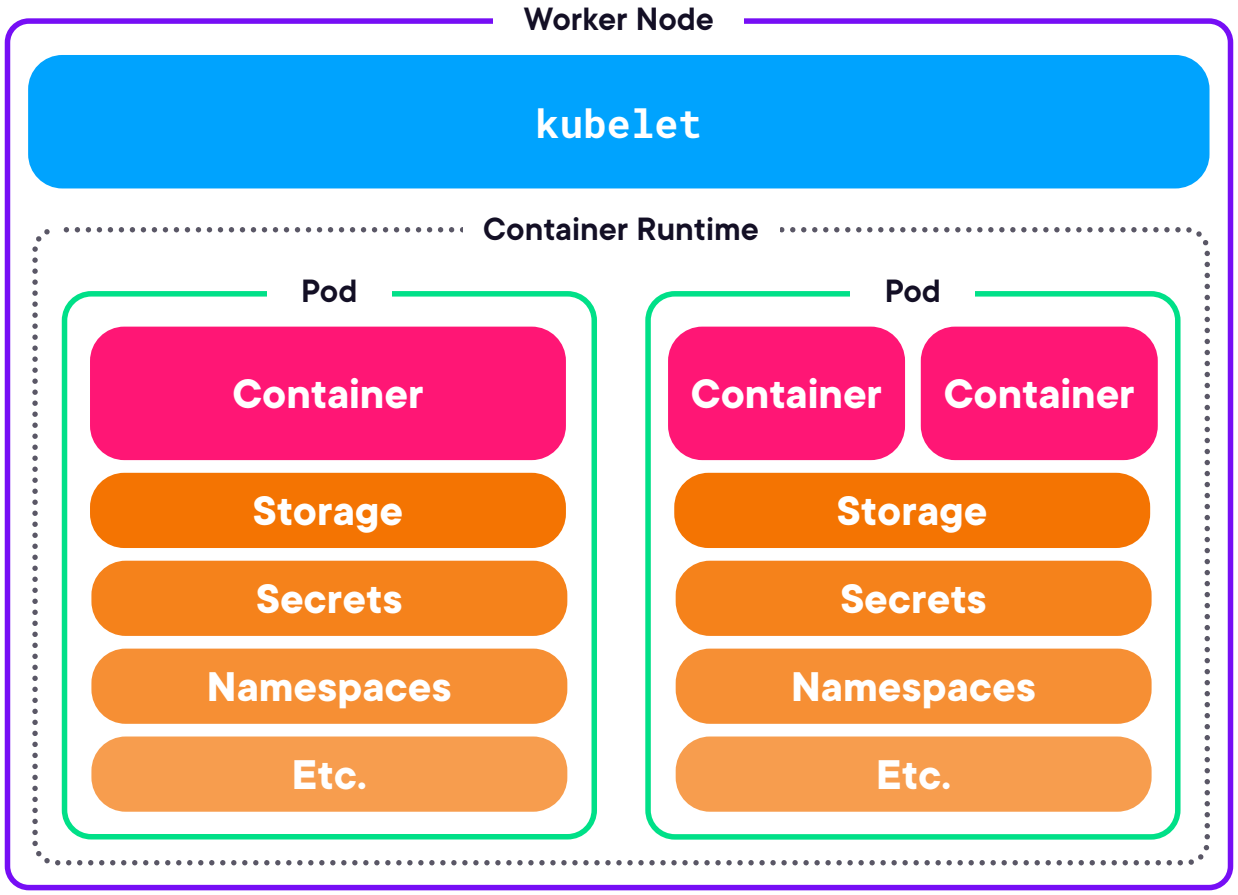
Use `/metrics` endpoint for sending metrics to other services.

Metrics Server

Not installed by default, but will be on the exam clusters

View resource-level metrics

- CPU
- Memory


Worker Node
kubelet
Container Runtime
Pod
Container
Storage
Secrets
Namespaces
Etc.
Pod
Container
Container
Storage
Secrets
Namespaces
Etc.

Demo: Installing Metrics Server

Metrics Server Prerequisites

The `kube-apiserver` must enable an aggregation layer

Metrics Server Prerequisites

Nodes must have Webhook authentication and authorization enabled

Metrics Server Prerequisites

The `kubelet` certificate needs to be signed by cluster Certificate Authority; alternatively, you can disable certificate validation by passing adding `--kubelet-insecure-tls` to the Metrics Server startup command in the manifest

Metrics Server Prerequisites

The container runtime must implement a container metrics RPCs (or have cAdvisor support)

Metrics Server Prerequisites

The network should support allow the control plane node to reach the Metrics Server on port 10250

Metrics server also needs to be able to speak to kubelet on all nodes

Accessing Kubernetes Metrics Data

kubectl top

View Metrics Server Data

```
> kubectl top nodes

NAME          CPU(cores)   CPU(%)   MEMORY(bytes)   MEMORY(%)
controller    217m         10%      1638Mi          43%
worker1       46m          2%       896Mi           23%
```

View Metrics Server Data

```
> kubectl top pods

NAME                          CPU(cores)   MEMORY(bytes)
web-app-7b99994d65-lwcmx      1m           5Mi
web-app-7b99994d65-st9rx      1m           9Mi
web-app-7b99994d65-zhbqd      1m           12Mi
```

Metrics Server does not provide historical data

View Metrics Server Data

```
> kubectl top pods -n kube-system
```

```
NAME                                  CPU(cores)   MEMORY(bytes)
coredns-674b8bbfcf-kwpkd              3m           41Mi
coredns-674b8bbfcf-nbr4w              2m           41Mi
etcd-controller                       26m          79Mi
kube-apiserver-controller             64m          272Mi
kube-controller-manager-controller    25m          124Mi
kube-proxy-rhz2v                      1m           77Mi
kube-proxy-sr4sw                      1m           76Mi
kube-scheduler-controller             12m          80Mi
metrics-server-5b5786cf87-4p4wj       5m           66Mi
```

View Metrics Server Data

```
> kubectl top pods --all-namespaces
```

NAMESPACE	NAME	CPU(cores)	MEMORY(bytes)
default	web-app-7b99994d65-lwcmx	1m	5Mi
default	web-app-7b99994d65-st9rx	1m	9Mi
default	web-app-7b99994d65-zhbqd	1m	12Mi
kube-flannel	kube-flannel-ds-6bddn	14m	54Mi
kube-flannel	kube-flannel-ds-9tzzq	18m	54Mi
kube-system	coredns-674b8bbfcf-kwpkd	2m	41Mi
kube-system	coredns-674b8bbfcf-nbr4w	3m	41Mi
kube-system	etcd-controller	29m	80Mi
kube-system	kube-apiserver-controller	51m	272Mi
kube-system	kube-controller-manager-controller	26m	124Mi
kube-system	kube-proxy-rhz2v	1m	77Mi

...

View Raw Metrics Data

```
> kubectl get --raw /api/v1/nodes/<NODE_NAME>/proxy/metrics/resource
```

```
# HELP container_cpu_usage_seconds_total [STABLE] Cumulative cpu time consumed by the
container in core-seconds
# TYPE container_cpu_usage_seconds_total counter
container_cpu_usage_seconds_total{container="kube-flannel",namespace="kube-
flannel",pod="kube-flannel-ds-6bddn"} 8.782792 1750962620660
container_cpu_usage_seconds_total{container="kube-proxy",namespace="kube-system",pod="kube-
proxy-sr4sw"} 1.187455 1750962623858
container_cpu_usage_seconds_total{container="nginx",namespace="default",pod="web-
app-7b99994d65-lwcmx"} 0.151779 1750962627564
container_cpu_usage_seconds_total{container="nginx",namespace="default",pod="web-
app-7b99994d65-st9rx"} 0.187888 1750962622829
# HELP container_memory_working_set_bytes [STABLE] Current working set of the container in
bytes
# TYPE container_memory_working_set_bytes gauge
...
```

Demo: Working with Kubernetes Metrics Server

Exam Scenario

Create a web application deployment using the `nginx` image; name the deployment `nginx`, create five replicas, and expose port 80

Check the resource usage for the deployment pods and the related worker nodes

Generate some traffic using `curl` against the `nginx` deployment

Review the resource usage once more

View the raw metrics data for one of the affected nodes. Save it in the file `workernode.txt`

Kubernetes Rollout Monitoring

View Live Rollout Data

```
> kubectl rollout status deployment <DEPLOYMENT_NAME>
```

```
kubectl rollout status deployment <DEPLOYMENT_NAME>
Waiting for deployment "deployment-name" rollout to finish: 0 of 3 updated replicas
are available...
Waiting for deployment "deployment-name" rollout to finish: 1 of 3 updated replicas
are available...
Waiting for deployment "deployment-name" rollout to finish: 2 of 3 updated replicas
are available...
deployment "deployment-name" successfully rolled out
```

View Historical Rollouts

```
> kubectl rollout history deployment <DEPLOYMENT_NAME>

REVISION  CHANGE-CAUSE
1         <none>
2         kubectl set image deployment <deployment-name> busybox=busybox:latest --
record=true
```

View Historical Rollouts

```
> kubectl rollout history deployment <DEPLOYMENT_NAME>

REVISION  CHANGE-CAUSE
1         <none>
2         kubectl set image deployment <deployment-name> busybox=busybox:latest --
record=true
```

Demo: Monitoring Kubernetes Rollouts

https://github.com/pluralsight-cloud/application-observability-maintenance-ckad

Exam Scenario

Deploy the `web-app` manifest and view the rollout deployment live

Upgrade the deploy to use the latest version of the Nginx image. Watch the deployment, then confirm by viewing the deployment history