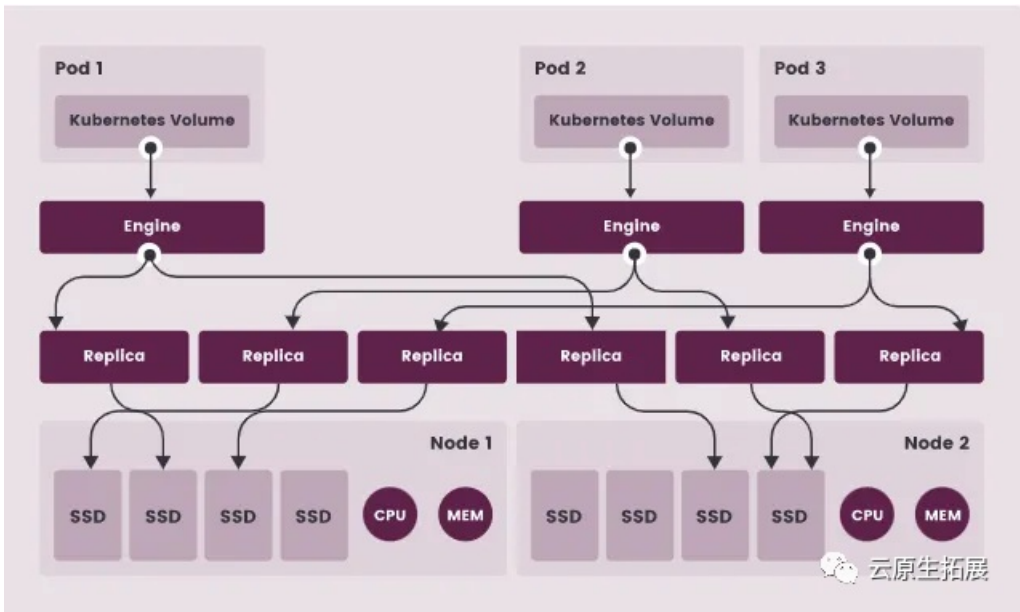


119Kubernetes 系列（一二）探索 Longhorn: Kubernetes 存储的游戏规则改变者

今天，我们将注意力转向 Kubernetes 的一个关键但往往具有挑战性的方面——存储。确保 Kubernetes 环境中持久且可靠的存储有时感觉像是一个复杂的难题。然而，有一个解决方案可以帮助我们将这些部分无缝地组合在一起—— **Longhorn**，并且可以在任何环境下运行。

Longhorn(<https://longhorn.io/kb/>) 是 Rancher Labs 的一个创新开源项目，为 Kubernetes 提供了可靠、轻量级且用户友好的分布式块存储系统。它正在改变游戏规则，让在 Kubernetes 环境中运行有状态应用程序变得更加简单，而无需遇到存储问题。

在这篇博文中，我们将深入了解 Longhorn 的世界，探索它的功能、架构以及为什么它可能非常适合您的 Kubernetes 存储需求。无论您是已经熟悉 Kubernetes 领域还是刚刚开始您的旅程，这份 Longhorn 综合指南都将为有效管理 Kubernetes 存储提供宝贵的见解。



Longhorn: 起源、架构和目的

Longhorn 诞生于 Rancher Labs，于 2014 年开始作为一个开源项目。目标很明确：简单、简化和民主化 Kubernetes 环境中持久存储卷的管理。它最初是 Rancher Labs 的内部项目，于 2020 年捐赠给云原生计算基金会（CNCF），巩固了其在 Kubernetes 生态系统中的地位。

什么是 Longhorn?

从本质上讲，Longhorn 是一个轻量级、可靠且易于使用的 Kubernetes 分布式块存储系统。本质上，它将商用硬件和云卷转变为可靠的分布式块存储解决方案，为 Kubernetes 提供软件定义存储（SDS）。

Longhorn 如何工作?

Longhorn 通过使用容器来创建存储卷并在 Kubernetes 集群中的多个节点之间同步复制它。它通过在不同节点和磁盘上智能地分布数据副本来确保存储在这些卷中的数据的高可用性。如果节点或磁盘发生故障，Longhorn 会自动将工作负载切换到另一个副本以保持可用性。

Longhorn 目标

Longhorn 的主要目标是尽可能轻松地管理 Kubernetes 工作负载的持久存储。在 Longhorn 等工具出现之前，管理 Kubernetes 中的有状态工作负载非常具有挑战性。开发人员和系统管理员通常不得不依赖外部存储系统、手动配置存储或仅限于特定的云提供商。

有了 Longhorn，这种情况就会改变。它允许用户直接从 Kubernetes 界面或通过 Kubernetes API 创建、管理和自动扩展持久卷。它还卷备份、快照和存储管理提供直观的界面——所有这些都是开箱即用的。

Longhorn 的设计与平台无关。这意味着它可以跨裸机、本地和基于云的 Kubernetes 安装运行，无论集群托管在何处，都能提供一致的体验。

在 Kubernetes 环境中使用 Longhorn 的特性和优点

Longhorn 提供了一组丰富的功能，旨在满足 Kubernetes 环境中的各种需求。这些功能使其成为处理 Kubernetes 中的持久存储、简化有状态应用程序的管理的绝佳选择。

1. 安装、操作方便

Longhorn 可以使用单个命令安装在任何 Kubernetes 集群上，因此入门非常简单。它提供了直观的图形用户界面，可简化管理任务，例如创建和附加卷、管理快照以及设置备份。

2. 轻便可靠

Longhorn 占用空间小，不需要任何额外的服务来运行，使其成为一个轻量级解决方案。它通过在集群中的不同节点之间同步复制卷来确保高可用性。如果某个节点或磁盘发生故障，Longhorn 会自动将数据重新复制到其他节点，从而维护数据安全。

3. 云原生和 Kubernetes 原生

作为云原生解决方案，Longhorn 可以在任何 Kubernetes 集群上运行，无论它是在本地、云端还是混合环境中。Longhorn 卷原生集成到 Kubernetes 中，这意味着您可以像管理任何其他 Kubernetes 资源一样管理它们。

4. 灾难恢复卷

Longhorn 支持跨集群容灾卷，在发生灾难时能够快速激活备用卷。此功能对于业务连续性至关重要，并满足企业应用程序的高可用性要求。

5. 增量快照和备份

Longhorn 允许您拍摄卷的增量快照，然后将其备份到辅助存储。此方法优化了存储使用并加快了备份和恢复操作。

6. ReadWriteMany (RWX) 卷支持

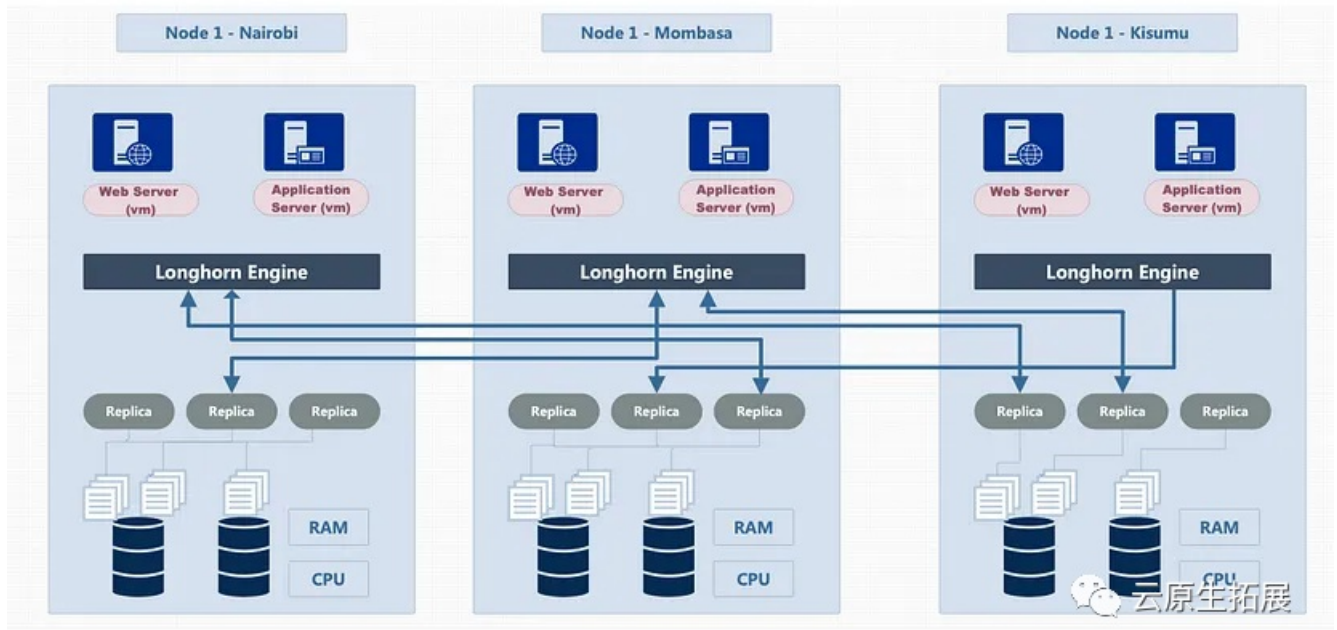
从 Longhorn 1.1.0 版本开始，Longhorn 支持 ReadWriteMany (RWX) 卷，允许多个节点同时读写一个卷，这是某些类型应用程序的关键要求。

7. 主动监控和警报

Longhorn 通过与流行的开源监控解决方案 Prometheus 集成来提供主动监控和警报。

8. 内置数据局部性和亲和性功能

Longhorn 支持数据局部性，尝试将数据和工作负载保留在同一节点中，以提高性能。它还支持用于高级卷调度的卷和节点关联/反关联策略



Longhorn 架构和操作

Longhorn 的架构设计为轻量级，并与 Kubernetes 完全集成。它由一组组件组成，这些组件协同工作，为您的 Kubernetes 工作负载提供可靠的分布式块存储解决方案。以下是关键组件的概述：

Longhorn Manager

Longhorn Manager 在 Kubernetes 集群中的每个节点上运行。它负责编排其他组件、处理调度、检测节点故障以及维护集群的状态。

Longhorn engine

Longhorn 引擎是数据平面组件，负责向存储后端读取和写入数据。每个 Longhorn 卷都有一个相应的 Longhorn 引擎实例。该引擎捕获快照、创建备份并跨不同节点复制数据以实现高可用性。

Longhorn UI

Longhorn UI 是一个图形用户界面，允许用户轻松管理卷、拍摄快照、设置备份和监控存储系统的状态。

Longhorn CSI（容器存储接口）驱动程序

CSI 驱动程序允许 Kubernetes 使用 Longhorn 作为其存储提供者。它将 Kubernetes 卷操作转换为相应的 Longhorn 操作，例如创建卷、将卷附加到节点或从节点分离以及将卷挂载到 pod 或从 pod 卸载卷。

Longhorn 实例管理器

有两种类型的实例管理器，一种用于管理 Longhorn Engine 实例，另一种用于管理 Longhorn Replica 实例。集群中的每个节点为每种类型运行一个实例管理器 Pod。

Longhorn Replicas

这些是 Longhorn engine 在不同节点上同步创建的数据副本，以实现冗余和高可用性。

Longhorn 如何运作？

在 Longhorn 中，当您在 Kubernetes 中创建持久卷声明 (PVC) 时，Longhorn CSI 驱动程序会将此请求传达给 Longhorn Manager。然后，管理器配置一个新的 Longhorn 卷和相应的 Longhorn 引擎。

引擎存储卷的实际数据并管理到分布在不同节点上的 Longhorn 副本的复制。即使节点发生故障，此复制也可确保您的数据保持安全且可用。

此外，当您拍摄快照时，引擎会创建数据的时间点副本，而不会影响正在运行的工作负载。这些快照可用于增量备份，并且可以存储在 AWS S3 或 NFS 服务器等辅助存储中。

总而言之，Longhorn 的架构旨在与 Kubernetes 无缝运行，重点是简单性、可靠性和易用性。该系统确保您的数据始终安全、可用，并且可以直接从 Kubernetes 界面或通过 Kubernetes API 轻松管理。

安装 Longhorn

先决条件

在安装 Longhorn 之前，请确保您的 Kubernetes 集群满足以下要求：

- Kubernetes v1.14 或更高版本
- 至少 1 个工作节点
- 已安装 Helm 3（如果您打算使用 Helm 进行安装）
- Open-iscsi 安装在所有节点上（在大多数云 Kubernetes 服务中，如 GKE、EKS、AKS，这是预安装的）

安装步骤

使用 kubectl

首先，从 Longhorn 发布页面下载 Longhorn YAML 文件。例如，如果您要安装 Longhorn 1.1.0，您将使用：

```
https://raw.githubusercontent.com/longhorn/longhorn/v1.1.0/deploy/longhorn.yaml
```

然后，使用 `kubectl` 应用 YAML 文件：

```
kubectl apply -f longhorn.yaml
```

使用 Helm

首先，添加 Longhorn Helm 存储库：

```
helm repo add longhorn https://charts.longhorn.io
```

然后，更新您的 Helm 存储库：

```
helm repo update
```

创建 longhorn 将安装到的命名空间

```
kubectl create namespace longhorn-system
```

最后，安装 Longhorn：

```
helm install longhorn longhorn/longhorn --namespace longhorn-system
```

```
onai@ians-MacBook-Air ~ % helm install longhorn longhorn/longhorn --namespace longhorn-system
```

```
NAME: longhorn
LAST DEPLOYED: Wed May 24 08:48:59 2023
NAMESPACE: longhorn-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Longhorn is now installed on the cluster!
```

```
Please wait a few minutes for other Longhorn components such as CSI deployments, Engine Images, and Instance Managers to be initialized.
```

云原生拓展


现在 Longhorn 已安装，我们可以查看正在运行的 Pod

```
kubectl get pods -n longhorn-system
```

```
onai@ians-MacBook-Air ~ % kubectl get pods -n longhorn-system
```

NAME	READY	STATUS	RESTARTS	AGE
longhorn-admission-webhook-65669f9957-4mjdk	1/1	Running	0	2m52s
longhorn-admission-webhook-65669f9957-xpw4h	1/1	Running	0	2m51s
longhorn-conversion-webhook-5c75577484-jbtgx	1/1	Running	0	2m52s
longhorn-conversion-webhook-5c75577484-v2mhb	1/1	Running	0	2m51s
longhorn-driver-deployer-9d85f7675-cggcj	0/1	Init:0/1	0	2m52s
longhorn-manager-qc67f	0/1	CrashLoopBackOff	2 (18s ago)	2m52s
longhorn-recovery-backend-c4c77d7d9-qtzxp	1/1	Running	0	2m51s
longhorn-recovery-backend-c4c77d7d9-rw96s	1/1	Running	0	2m52s
longhorn-ui-5d8569577-fn5qf	1/1	Running	0	2m52s
longhorn-ui-5d8569577-n5kbb	1/1	Running	0	2m51s

```
onai@ians-MacBook-Air ~ %
```



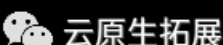
接下来我们可以查看服务并且能够查看longhorn ui

```
kubectl get svc -n longhorn-system
```

```
onai@ians-MacBook-Air ~ % kubectl get svc -n longhorn-system
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
longhorn-admission-webhook	ClusterIP	10.105.136.182	<none>	9443/TCP	4m28s
longhorn-backend	ClusterIP	10.110.153.141	<none>	9500/TCP	4m28s
longhorn-conversion-webhook	ClusterIP	10.109.148.154	<none>	9443/TCP	4m28s
longhorn-engine-manager	ClusterIP	None	<none>	<none>	4m28s
longhorn-frontend	ClusterIP	10.97.173.209	<none>	80/TCP	4m28s
longhorn-recovery-backend	ClusterIP	10.111.249.150	<none>	9600/TCP	4m28s
longhorn-replica-manager	ClusterIP	None	<none>	<none>	4m28s

```
onai@ians-MacBook-Air ~ %
```




接下来我们要查看 longhorn ui。我将通过端口转发：

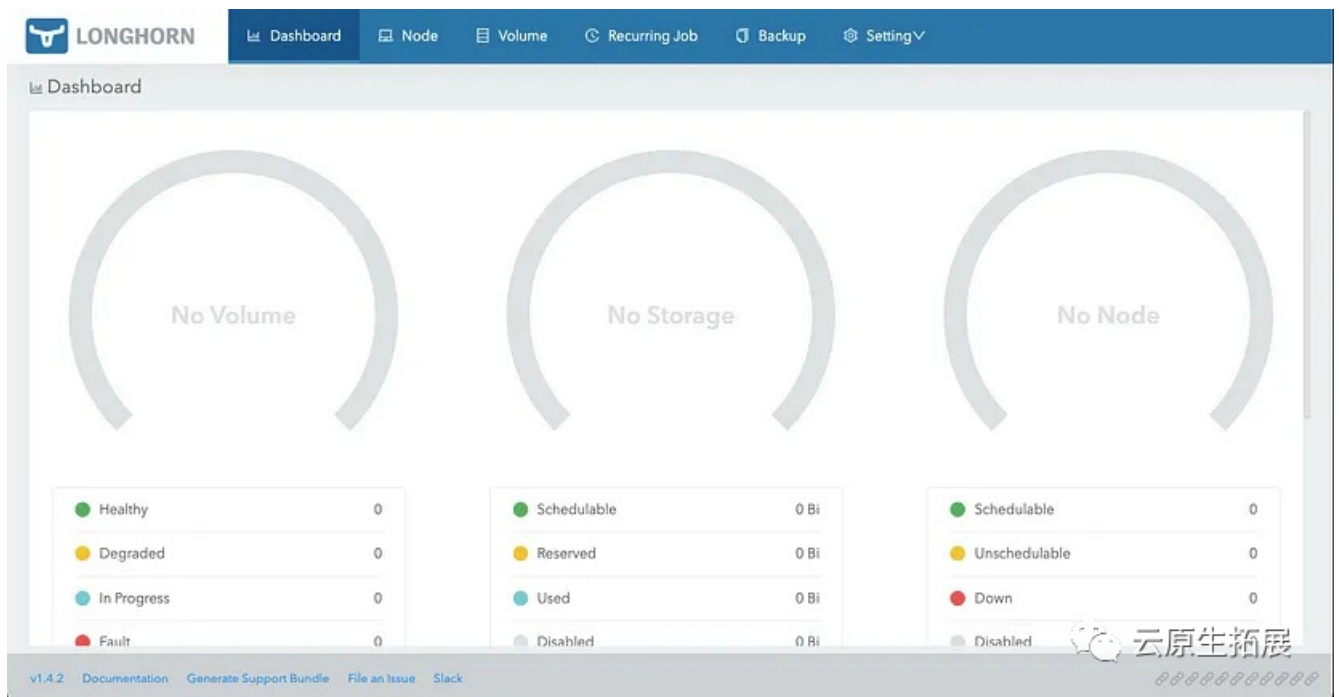
```
kubectl port-forward -n longhorn-system svc/longhorn-frontend 8080:80
```

```
onai@ians-MacBook-Air ~ % kubectl port-forward -n longhorn-system svc/longhorn-frontend 8080:80
```

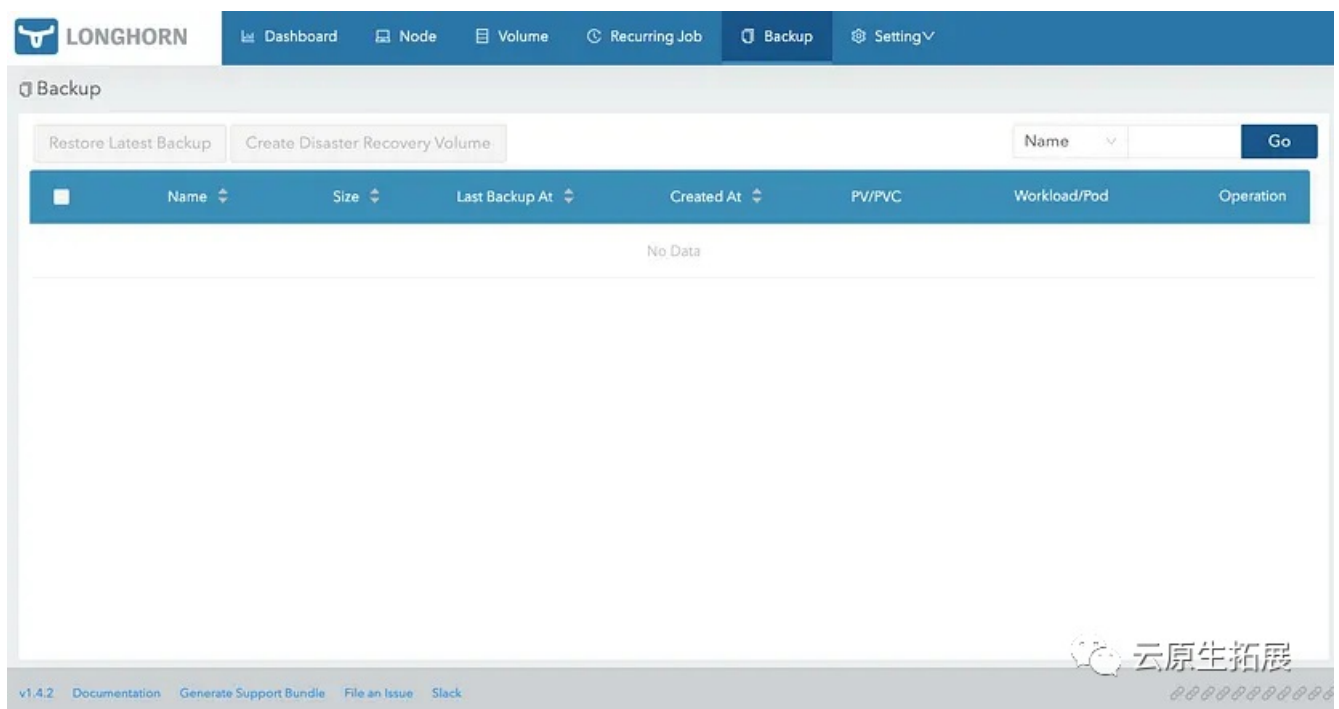
```
Forwarding from 127.0.0.1:8080 -> 8000
Forwarding from [::1]:8080 -> 8000
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
```



获取IP地址并查看UI



从仪表板中，我们可以看到不同的方面。在顶部栏中，您可以单击备份并能够查看已完成的备份或恢复到以前的备份。



在 Longhorn 中创建卷

您可以从 UI 创建卷，也可以通过 Kubernetes 持久卷声明 (PVC) 创建卷

1. 单击页面顶部的 **Volume** 选项卡。
2. 单击 **Create** 按钮。这将打开 Create Volume 页面。
3. 填写所需的详细信息，例如 Name、Size、Number of Replicas 等。然后单击 Create

* Name:

* Size: Gi ▼

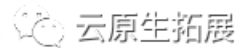
* Number of Replicas: ✓

* Frontend: Block Device ▼ ✓

Data Locality: disabled ▼ ✓

Access Mode: ReadWriteOnce ▼ ✓

Backing Image: ▼



通过 Kubernetes 持久卷声明 (PVC):

您还可以通过在 Kubernetes 中创建 PVC 来创建 Longhorn 卷。以下是如何执行此操作的示例:

创建 `PersistentVolumeClaim` YAML 文件。例如, 您可以将其命名为 `longhorn-volume-pvc.yaml` :

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: longhorn-vol-claim
spec:
  storageClassName: longhorn
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
```

```
kubectl apply -f longhorn-volume-pvc.yaml
```

```
onai@ians-MacBook-Air Longhorn % kubectl apply -f longhorn-volume-pvc.yaml
persistentvolumeclaim/longhorn-vol-claim created
onai@ians-MacBook-Air Longhorn %
```



您现在可以从仪表板查看 PVC 并使用它, 您可以创建备份并恢复它们。

当我们结束对 Longhorn 的讨论时，可以清楚地看到这种云原生、轻量级且易于使用的存储系统为 Kubernetes 环境带来的许多优势。从简单的安装到与云无关的性质，再到其高可用性和强大的数据保护功能，Longhorn 旨在简化 Kubernetes 工作负载的存储管理。

借助 Longhorn，用户可以更加专注于构建应用程序，而不必担心底层存储层。我们强调的功能（例如增量快照和备份、对 RWX 卷的支持以及跨集群灾难恢复）只是 Longhorn 所能提供的功能的一小部分。更不用说其直观的用户界面，使您的存储资源的管理和故障排除变得轻而易举。

话虽这么说，我相信 Longhorn 是一个强大的存储解决方案，可以满足小规模和企业级 Kubernetes 部署。如果您还没有尝试过，我鼓励您尝试一下并探索它如何简化您的存储管理。

请记住，与任何技术一样，理解它的最好方法就是亲自实践。安装 Longhorn，创建一些卷，看看它如何改善您的 Kubernetes 体验。