# Longhorn开发分析

<https://longhorn.io/blog/longhorn-v1.1.0/>

介绍了新特性， 我后面测试一下。

Longhorn-UI

前端Node.JS 21979行

|  |  |  |  |
| --- | --- | --- | --- |
| 组件 | 功能 | 开发语言 | 代码行数(go不含vendor) |
| longhorn-engine | 存储引擎 | Go | 20184 |
| Longhorn-UI | 前端UI | Node.JS | 21979 |
| Longhorn-Manager | Longhorn管理 | Go | 55701 |
| Longhorn-Instance-Manager | Volume编排 | Go | 3559 |
| longhorn-share-manager | Volume nfs共享 | Go | 664 |

|  |  |  |  |
| --- | --- | --- | --- |
| 组件 | 功能 | 开发语言 | 代码行数 |
| ceph | 存储集群 | CPP/C | 635192 |
| rook | K8s driver | Go | 124844 |

Longhorn应用场景： 用于Paas ， 容器私有云（云管和租户），内部云云管；

Ceph应用场景：PaaS+Iaas

Ceph性能 <https://blog.csdn.net/NewTyun/article/details/114362204>

# Longhorn Cleanup

<https://github.com/longhorn/longhorn-manager>

Longhorn CRD has finalizers in them, so user should delete the volumes and related resource first, give manager a chance to clean up after them.

To prevent damage to the Kubernetes cluster, we recommend deleting all Kubernetes workloads using Longhorn volumes (PersistentVolume, PersistentVolumeClaim, StorageClass, Deployment, StatefulSet, DaemonSet, etc).

1. Create the uninstallation job to cleanly purge CRDs from the system and wait for success:

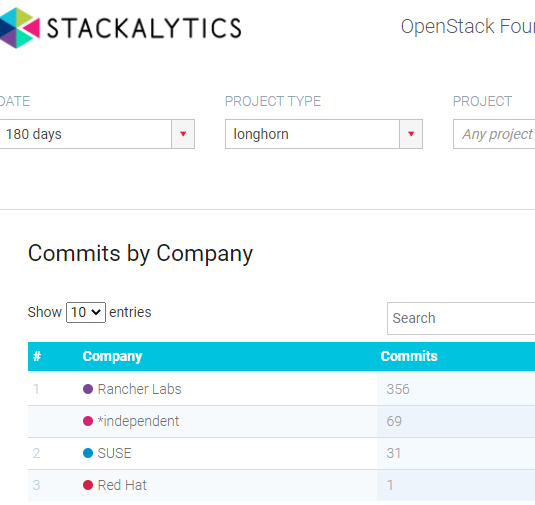
kubectl create -f deploy/uninstall/uninstall.yaml

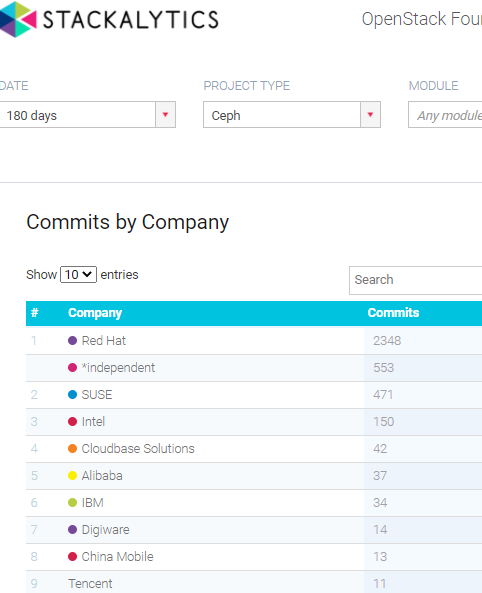
kubectl get job/longhorn-uninstall -w

# 开源社区动态

尽在掌握啊

<https://www.stackalytics.com/>





# Find排除某个目录

# 查找src目录下的vue, js和less文件总行数(递归)

find ./src/ -type f -regex ".\*.\(vue\|js\|less\)$" | xargs cat | wc -l

# 或者这个, 效果一样

find ./src/ -name "\*.js" -o -name "\*.vue" -o -name "\*.less" | xargs cat | wc -l

find . -name "\*.go" -o -name "\*.sh" -o -name "\*.c" | xargs cat | wc -l

This is the only one that worked for me.

find / -name MyFile ! -path '\*/Directory/\*'

Searching for "MyFile" excluding "Directory". Give emphasis to the stars \* .

find . -path ./tests -prune -false -o -name "\*.c" -o -name "\*.go" -o -name "\*.sh" | xargs cat | wc -l

# 认证系统

<https://zhuanlan.zhihu.com/p/65640809>

GitHub oauth 2.0

前端 Node.js

后端Golang

## 开源的权限登录系统

<https://blog.csdn.net/lovexiaotaozi/article/details/104156072>

# **权限验证框架Shiro**

<https://blog.csdn.net/xiaokang123456kao/article/details/72910806>

<https://blog.csdn.net/qq_18404993/article/details/106035472>

<https://blog.csdn.net/w54dxs/article/details/72890395>

# **OpenStack Keystone 总结**

Keystone（OpenStack Identity Service）是 OpenStack 框架中负责管理身份验证、服务访问规则和服务令牌功能的组件。用户访问资源需要验证用户的身份与权限，服务执行操作也需要进行权限检测，这些都需要通过 Keystone 来处理。Keystone 类似一个服务总线， 或者说是整个 Openstack 框架的注册表，OpenStack 服务通过 Keystone 来注册其 Endpoint（服务访问的URL），任何服务之间的相互调用，都需要先经过 Keystone 的身份验证，获得目标服务的 Endpoint ，然后再调用。

Keystone 的主要功能如下：

管理用户及其权限；

维护 OpenStack 服务的 Endpoint；

Authentication（认证）和 Authorization（鉴权）。

————————————————

版权声明：本文为CSDN博主「dylloveyou」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：<https://blog.csdn.net/dylloveyou/article/details/80329732>

## Node.JS

# **从零开始nodejs系列文章-nodejs到底能干什么**

<https://blog.csdn.net/weixin_39214481/article/details/82023696>