**Minik8s 验收报告**

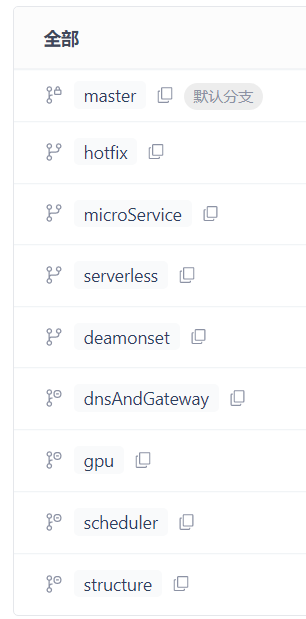
Minik8s 是一个类似于 Kubernetes 的迷你容器编排工具，能够在多机上对满足 CRI 接口的容器进行管理。支持容器生命周期管理、动态伸缩、自动扩容等基本功能，并且基于 Minik8s 实现了 Serverless 平台集成。

1. **成员分工及贡献比例**

|  |  |  |
| --- | --- | --- |
| 姓名 | 分工 | 贡献度 |
| 杨景凯（组长） | Service抽象，DNS与转发，Microservice | 34% |
| 彭逸帆 | Pod抽象，ReplicaSet抽象，多机部署与调度，控制面容错，CI/CD | 33% |
| 张世昊 | 动态伸缩，GPU应用，Serverless | 33% |

1. **项目开发过程简介**

* gitee仓库地址：<https://gitee.com/ethereal-O/k8s.git>
* 仓库共有9个分支：



* 所有分支可分为三类：

1. **master分支**是Minik8s的稳定版本，可以随时发布。 master分支不接受push，只接受hotfix分支的并入。并且只有master分支会触发CI/CD。
2. **hotfix分支**接受各个功能性分支的并入请求。由于各个功能在集成时可能出现新的问题，因此需要在hotfix分支上快速解决这些问题，待确认无误后会并入master分支。
3. 其余分支如serverless, gpu都是实现并测试某个具体功能的**功能性分支**，最终会并入hotfix分支。

* 以下是一个完整的工作流，以serverless功能的开发为例：

1. 以master分支为基线，创建serverless分支。
2. 在serverless分支上进行开发和单元测试。
3. 通过PR将serverless分支并入hotfix分支。
4. 拉取hotfix分支，检查serverless功能与其他功能的集成是否存在问题。
5. 如果有问题，在hotfix分支上进行修改。
6. 确认无误后，通过PR将hotfix分支并入master分支。
7. master分支成为新的基线。

* 项目采用的软件测试方法：

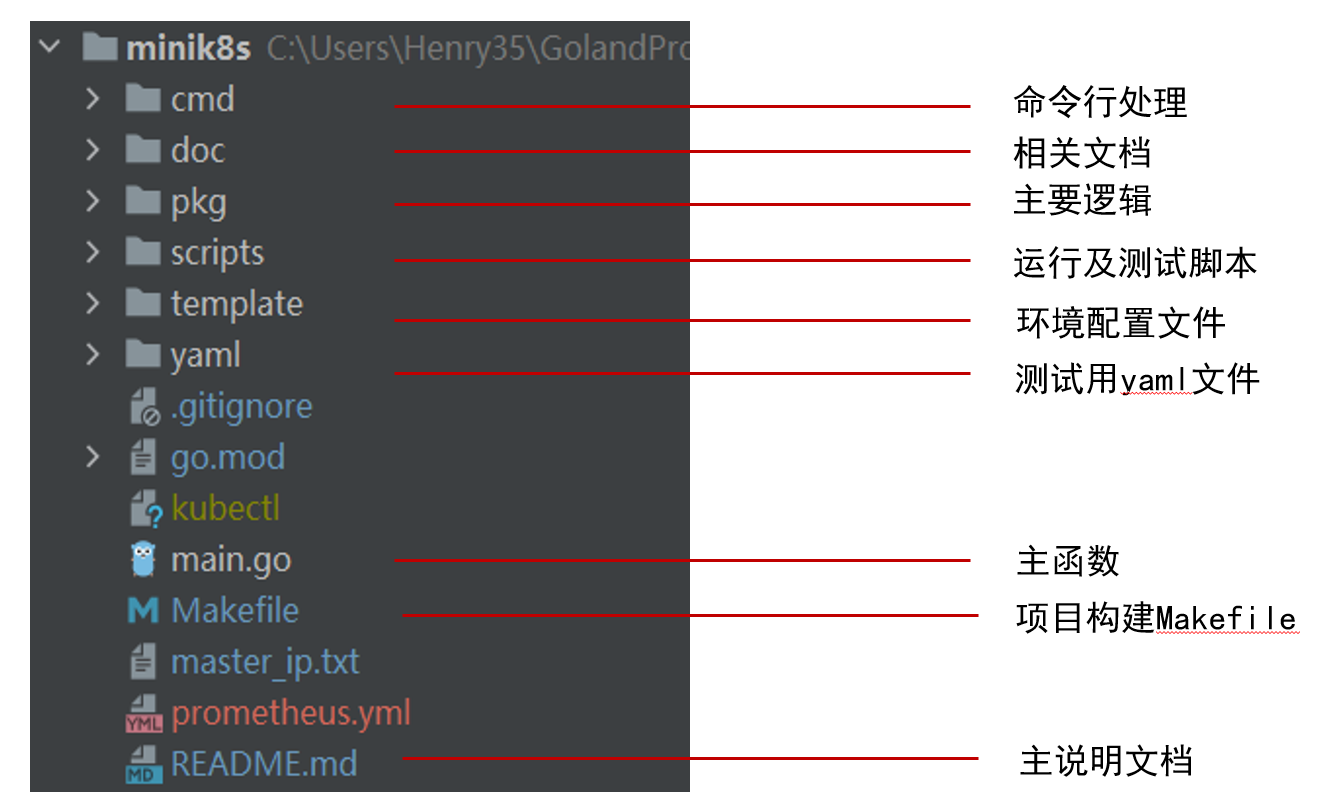
1. **代码走查**：在每周组会中，开发某一功能的组员需要向其余两位组员逐条讲解代码并描述功能的设计、框架、实现思路。其余两位组员对其中的细节和可能存在的问题进行质询，从而保证软件质量。
2. **同行审查**：在分支合并的过程中，开发某一功能的组员需要指派另一名组员作为审查人员和测试人员，从而对功能的实现和缺陷进行进一步的测试和检查，提高软件质量。
3. **回归测试**：在hotfix分支上，开发某一功能的组员需要运行以往的所有测试脚本，以保证各个功能在集成时没有出现新的问题。

* CI/CD的实现方式：

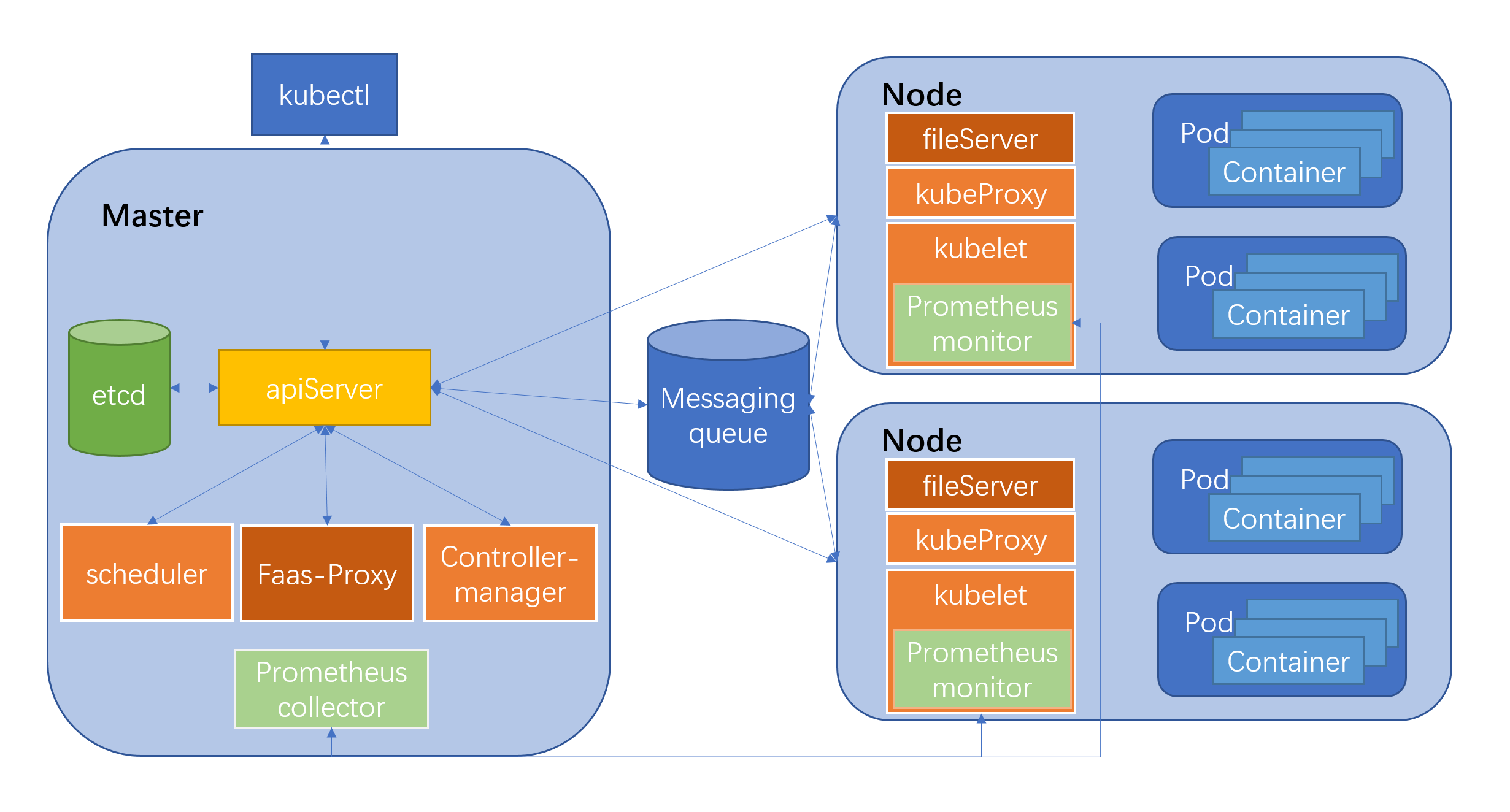
…

1. **项目整体架构与软件栈**

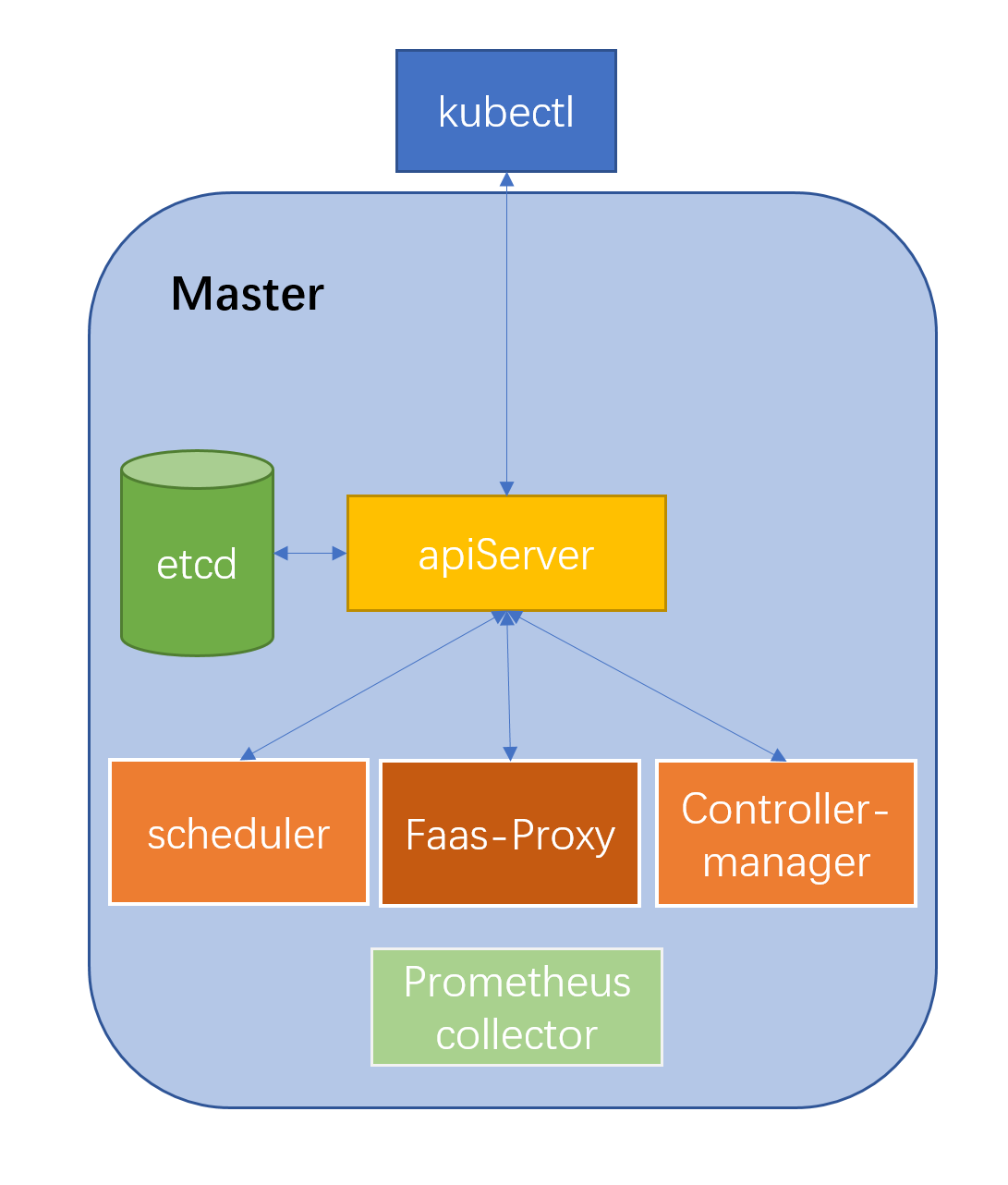
* 项目的文件结构：



* 项目的整体架构：

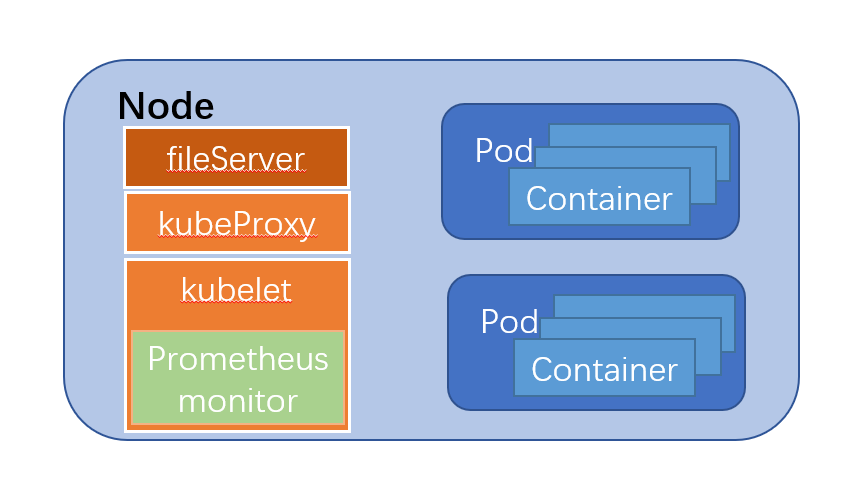


* Master节点上的组件及其作用：



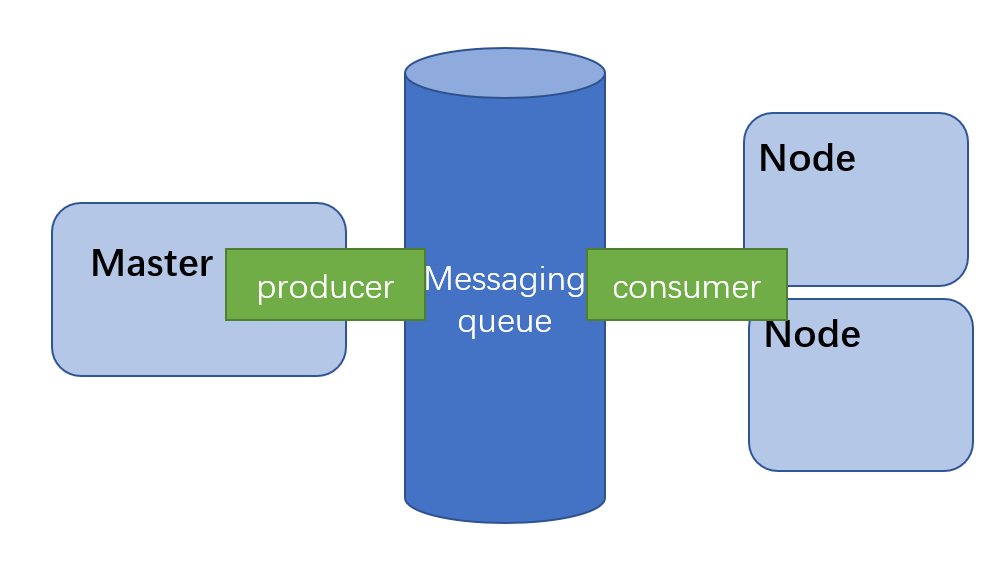
1. etcd：存储集群的所有需要持久化的数据。
2. apiServer：集群操作和CRUD的入口，采用现有echo框架，支持restful服务。
3. scheduler：负责集群资源调度，当前支持的策略有round-robin和根据Node资源以及Pod反亲和性调度。
4. Controller-manager：负责维护集群的实际状态与预期状态一致。
5. Faas-Proxy：负责接受并转发serverless的请求，支持流量统计，缓存。
6. Prometheus collector：通过Prometheus采集Node的监控数据。

* Worker节点上的组件及其作用：



1. kubelet：负责维护Pod和容器的生命周期，负责Node的启动，维护与删除，同时也通过内置的Prometheus monitor 向Master节点提供Node和Pod的资源使用情况。
2. kubeProxy：负责提供集群内部的服务发现和负载均衡，目前提供三套服务方案：基于nginx的反向代理与负载均衡，基于iptables的请求转发，基于sidecar的拦截分发。
3. fileServer：负责节点之间的文件传输，包括GPU配置文件，CUDA文件，函数文件等。

* 节点间的消息传递机制——消息队列：



Producer运行在Master节点上，负责上传数据。Consumer运行在Worker节点上，负责接收数据。不同节点共享topic，但拥有独立的channel。

* 项目使用的软件栈与开源组件：

1. 数据存储：etcd
2. 消息队列：nsq
3. Web框架：echo
4. 容器运行时：Docker
5. 指标监控：Prometheus
6. CNI插件：weave和flannel
7. **项目的所有功能、使用方法和实现方式**
8. 多机部署与调度
9. Pod抽象
10. Service抽象
11. ReplicaSet抽象
12. 动态伸缩
13. DNS与转发
14. 控制面容错
15. GPU应用
16. Serverless
17. Microservice