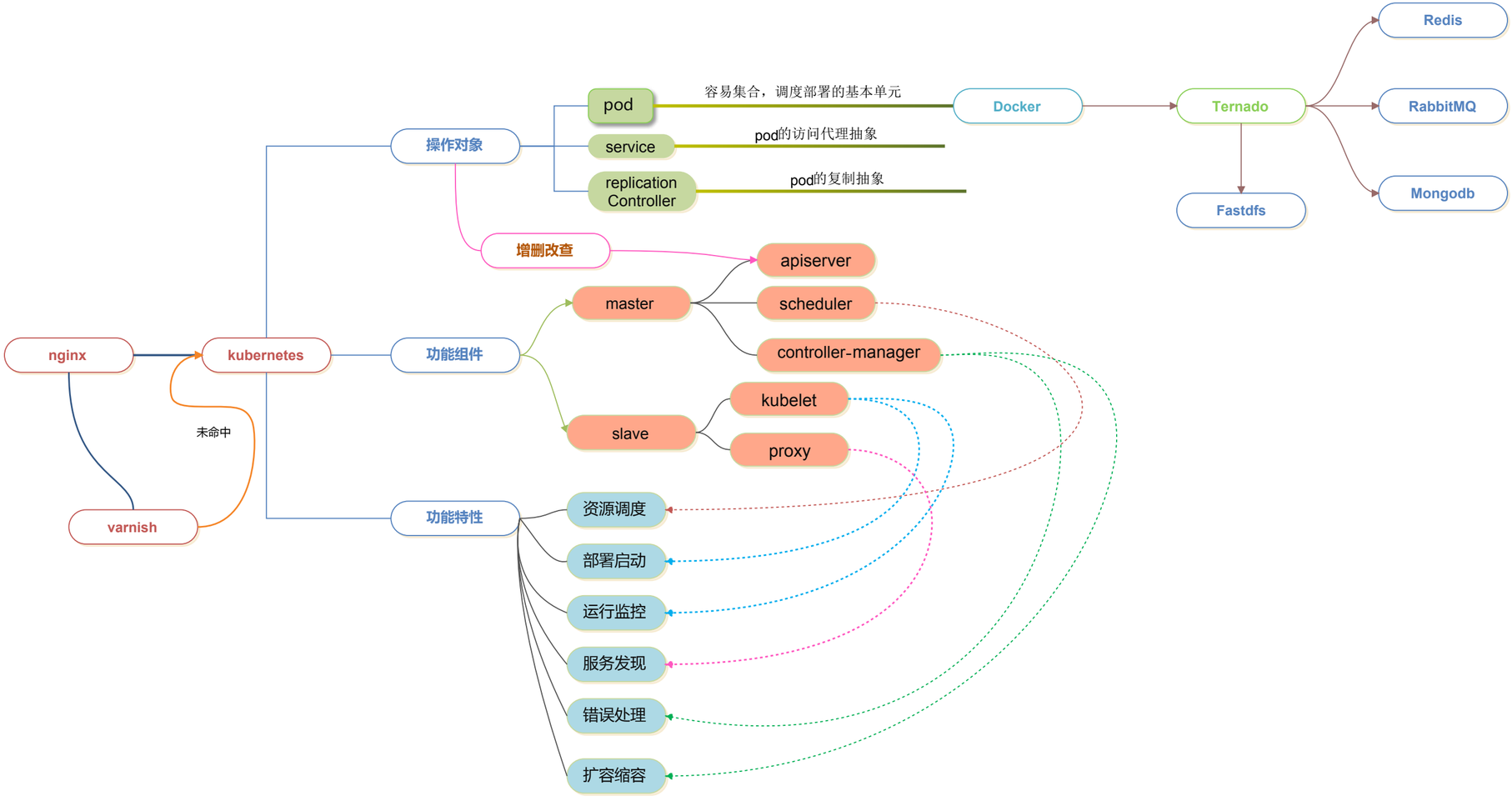

Redis
RabbitMQ
Mongodb
Fastdfs
Ternado
Docker
容易集合，调度部署的基本单元
pod
pod的访问代理抽象
service
pod的复制抽象
replication Controller
操作对象
增删改查
apiserver
master
scheduler
controller-manager
nginx
kubernetes
功能组件
kubelet
slave
proxy
未命中
varnish
功能特性
资源调度
部署启动
运行监控
服务发现
错误处理
扩容缩容

Kubernetes以RESTFul形式开放接口，用户可操作的REST对象有三个：

pod：是Kubernetes最基本的部署调度单元，可以包含container，逻辑上表示某种应用的一个实例。比如一个web站点应用由前端、后端及数据库构建而成，这三个组件将运行在各自的容器中，那么我们可以创建包含三个container的pod。

service：是pod的路由代理抽象，用于解决pod之间的服务发现问题。因为pod的运行状态可动态变化(比如切换机器了、缩容过程中被终止了等)，所以访问端不能以写死IP的方式去访问该pod提供的服务。service的引入旨在保证pod的动态变化对访问端透明，访问端只需要知道service的地址，由service来提供代理。

replicationController：是pod的复制抽象，用于解决pod的扩容缩容问题。通常，分布式应用为了性能或高可用性的考虑，需要复制多份资源，并且根据负载情况动态伸缩。通过replicationController，我们可以指定一个应用需要几份复制，Kubernetes将为每份复制创建一个pod，并且保证实际运行pod数量总是与该复制数量相等(例如，当前某个pod宕机时，自动创建新的pod来替换)。

master运行三个组件：

apiserver：作为kubernetes系统的入口，封装了核心对象的增删改查操作，以RESTFul接口方式提供给外部客户和内部组件调用。它维护的REST对象将持久化到etcd（一个分布式强一致性的key/value存储）。

scheduler：负责集群的资源调度，为新建的pod分配机器。这部分工作分出来变成一个组件，意味着可以很方便地替换成其他的调度器。

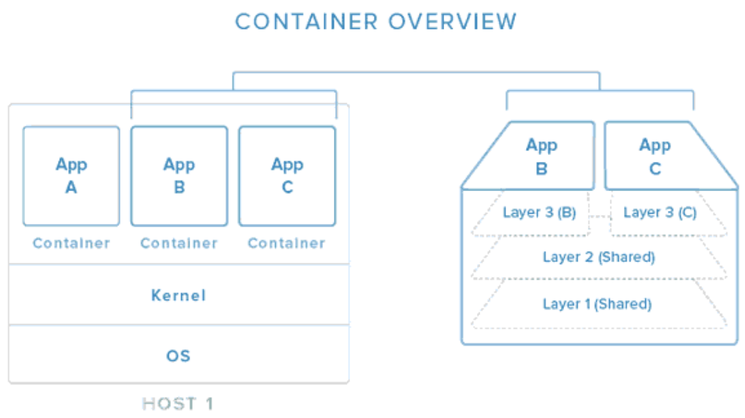
controller-manager：负责执行各种控制器，目前有两类：

- endpoint-controller ：定期关联service和pod(关联信息由endpoint对象维护)，保证service到pod的映射总是最新的。
- replication-controller ：定期关联replicationController和pod，保证replicationController定义的复制数量与实际运行pod的数量总是一致的。

slave(称作minion)运行两个组件：

kubelet：负责管控docker容器，如启动/停止、监控运行状态等。它会定期从etcd获取分配到本机的pod，并根据pod信息启动或停止相应的容器。同时，它也会接收apiserver的HTTP请求，汇报pod的运行状态。

proxy：负责为pod提供代理。它会定期从etcd获取所有的service，并根据service信息创建代理。当某个客户pod要访问其他pod时，访问请求会经过本机proxy做转发。


CONTAINER OVERVIEW
App A
App B
App C
Container
Container
Container
Kernel
OS
HOST 1
App B
App C
Layer 3 (B)
Layer 3 (C)
Layer 2 (Shared)
Layer 1 (Shared)

Varnish 处理 HTTP 请求的过程大致分为如下几个步骤:

Receive 状态（vcl_recv）： 也就是请求处理的入口状态，根据 VCL 规则判断该请求应该 pass（vcl_pass）或是 pipe（vcl_pipe），还是进入 lookup（本地查询）。

Lookup 状态： 进入该状态后，会在 hash 表中查找数据，若找到，则进入 hit（vcl_hit）状态，否则进入 miss（vcl_miss）状态。

Pass（vcl_pass）状态: 在此状态下，会直接进入后端请求，即进入 fetch（vcl_fetch）状态

Fetch（vcl_fetch）状态: 在 fetch 状态下，对请求进行后端获取，发送请求，获得数据，并根据设置进行本地存储。

Deliver（vcl_deliver）状态： 将获取到的数据发给客户端，然后完成本次请求。