**Kubernetes Node**

Node是Kubernetes中的工作机器，早先称为minion。Node可以是虚拟机或物理机，取决于所在的集群。每个Node包含有运行pod必须的服务，并被master组件管理。在Node上的服务包括container runtime，kubelet和kube-proxy，在架构设计文档中可以看到更多细节。

**Node Status（节点状态）**

节点状态包含以下信息：

**Addresses（地址）**

地址信息包含三个字段，字段值取决于云提供商或裸机配置

HostName，有Node内核决定，可以通过kubelet --hostname-override参数覆盖。

ExternalIP，从外部访问Node的IP地址。

InternalIP，仅能通过集群内部访问的IP地址。

**Conditions（状态）**

该字段描述了所有运行中的Node状态，其可能的值如下：

|  |  |
| --- | --- |
| Node Condition | 描述 |
| OutOfDisk | True，如果可用空间不足以添加新的pod，否则False |
| Ready | True，如果Node健康且能接收pod；  False，Node不健康且无法接收pod；  Unknown，Node Controller没有从上一个监管周期得到回复 |
| MemoryPressure | True，Node可用内存低，否则False |
| PIDPressure | True，Node中进程过多，否则False |
| DiskPressure | True，Node可用磁盘空间低，否则False |
| NetworkUnavailable | True，Node网络配置未正确配置，否则False |

若Ready状态下，保持Unknown或False超过了pod-eviction-timeout，所有Node上的pod将通过Node Controller计划进行删除。默认的超时时间是5分钟。

当Node无法访问时，apiserver无法与Node上的kubelet通信，删除pods的决定无法与kubelet通信，直到apiserver与其重新建立通信。在这期间，要删除的pod可能在其他Node上继续运行。

在1.5之前，Node Controller会强制删除这些pod，但在1.5以后，会在确认它们在集群中停止运行后才删除。可能看到pods会以Terminating或Unknown状态运行在无法访问的Node上。如果一个Node从集群离开，Kubernetes可能无法从底层设备上推断出来，这时管理员需要手动删除该Node。删除了Node对象后，所有运行在其上的pod会从apiserver中删除，并释放它们的名称。

在版本1.12中，TaintNodesByCondition功能被提升为beta版，因此节点生命周期控制器会自动创建表示条件的污点。 类似地，调度程序在考虑节点时忽略条件; 相反，它会查看Node的污点和Pod的容忍度。

现在，用户可以在旧的调度模型和更灵活的新调度模型之间进行选择。 根据旧型号，可以安排没有任何容忍度的Pod。 但是可以在该节点上安排容忍特定节点的污点的Pod。

警告：启用此功能会在观察到条件和创建污点之间产生一个小延迟。 此延迟通常小于一秒，但它可以增加成功安排但被kubelet拒绝的Pod的数量

**Capacity and Allocatable（容量与可分配）**

描述Node可用资源，CPU，内存，最大Pod数量。

容量块的字段声明Node资源总量，可分配块描述Node能使用的资源数量。

**Info（信息）**

描述Node的通用信息，比如内核版本，Kubernetes版本，Docker版本，操作系统名称，这些信息会通过kubelet收集。

**Management（管理）**

与pod和service不同，node不由Kubernetes创建，是由外部提供商创建，或存在于物理池或虚拟机。所以当Kubernetes创建Node时，它只是创建了一个对象代表Node。创建完成后，Kubernetes会检查Node是否合法。Kubernetes保留无效节点的对象，并不断检查它是否有效，必须显式删除Node对象才能停止此过程。

目前，存在三个组件与Kubernetes Node交互。

**Node Controller**

是Master组件，管理Node多个方面。

在Node生命周期中扮演多个角色，首先是当Node注册时分配CIDR块（需要CIDR分配功能开启）

第二是保持Node Controller中的Node列表与云提供商的可用机器列表保持一致。在云环境中运行时，只要节点不健康，节点控制器就会询问云提供商该节点的VM是否仍然可用。 如果不是，则节点控制器从其节点列表中删除该节点。

第三是监测节点的健康状况。 节点控制器负责在节点变得无法访问时将NodeStatus的NodeReady条件更新为ConditionUnknown（即节点控制器因某种原因停止接收心跳，例如由于节点关闭），然后逐出驱逐节点中的所有pod （如果节点仍然无法访问，则使用正常终止）。 （默认超时为40秒，开始报告ConditionUnknown，之后5米开始驱逐pod。）节点控制器每隔--node-monitor-period秒检查每个节点的状态。

在1.13之前的Kubernetes版本中，NodeStatus是节点的心跳。 从Kubernetes 1.13开始，节点租用功能作为alpha功能引入（NodeLease，KEP-0009）。 启用节点租用功能后，每个节点在kube-node-lease命名空间中都有一个关联的Lease对象，该节点由节点定期更新，NodeStatus和节点租约都被视为来自节点的心跳。 节点租约经常更新，而NodeStatus仅在有一些更改或经过足够时间时从节点报告为主节点（默认值为1分钟，超过默认超时40秒的无法访问节点）。 由于节点租约比NodeStatus轻得多，因此从可伸缩性和性能角度来看，此功能使节点心跳显着降低。

在Kubernetes 1.4中，我们更新了节点控制器的逻辑，以便在大量节点到达主站时遇到问题时更好地处理案例（例如，因为主站有网络问题）。 从1.4开始，节点控制器在决定pod驱逐时查看集群中所有节点的状态。

在大多数情况下，节点控制器将驱逐率限制为每秒 - 节点驱逐率（默认值0.1），这意味着它不会每10秒从多个节点驱逐pod。

当给定可用区中的节点变得不健康时，节点逐出行为会发生变化。 节点控制器同时检查区域中节点的百分比是否不健康（NodeReady条件是ConditionUnknown或ConditionFalse）。 如果不健康节点的比例至少为--unhealthy-zone-threshold（默认值为0.55），那么驱逐率会降低：如果群集很小（即小于或等于--large-cluster-size-threshold节点） - 默认为50）然后停止驱逐，否则驱逐率降低到每秒二次节点驱逐率（默认值0.01）。 每个可用区域实施这些策略的原因是，一个可用区域可能从主服务器分区而其他可用区域保持连接。 如果您的群集未跨越多个云提供商可用区域，则只有一个可用区域（整个群集）。

在可用区域之间传播节点的一个关键原因是，当整个区域出现故障时，工作负载可以转移到健康区域。因此，如果区域中的所有节点都不健康，则节点控制器以正常速率驱逐 - 节点驱逐率。角落情况是所有区域完全不健康（即群集中没有健康的节点）。在这种情况下，节点控制器假定主连接存在一些问题，并在某些连接恢复之前停止所有驱逐。

从Kubernetes 1.6开始，NodeController还负责驱逐在具有NoExecute taints的节点上运行的pod，当pod不能容忍taints时。此外，作为默认禁用的alpha功能，NodeController负责添加与节点无法访问或未就绪等节点问题相对应的污点。有关NoExecute taints和alpha功能的详细信息，请参阅此文档。

从版本1.8开始，节点控制器可以负责创建表示节点条件的污点。这是1.8版的alpha功能。

**Self-Registration of Nodes**

当kubelet标志--register-node为true（默认值）时，kubelet将尝试向API服务器注册自己。 这是大多数发行版使用的首选模式。

对于自行注册，可以使用以下选项启动kubelet：

--kubeconfig：凭证的路径，以向apiserver验证自身。

--cloud-provider：如何与云提供商交谈以阅读有关自身的元数据。

--register-node：自动注册API服务器。

--register-with-taints：使用给定的taints列表注册节点（逗号分隔<key> = <value>：<effect>）。 如果register-node为false，则为no-op。  
--node-ip：Node的IP地址。

--node-lables：在群集中注册节点时要添加的标签

--node-status-update-frequency：指定kubelet将节点状态发布到master的频率。

**Manual Node Administration**

集群管理员可以创建和修改节点对象。

如果管理员希望手动创建节点对象，请设置kubelet标志--register-node = false。

管理员可以修改节点资源（无论--register-node的设置如何）。 修改包括在节点上设置标签并将其标记为不可调度。

节点上的标签可以与pod上的节点选择器结合使用以控制调度，例如， 将pod限制为仅有资格在节点的子集上运行。

将节点标记为不可调度可防止将新pod调度到该节点，但不会影响节点上的任何现有pod。

**Node capacity**

节点的容量（cpus的数量和内存量）是节点对象的一部分。 通常，节点在创建节点对象时注册自己并报告其容量。 如果您正在进行手动节点管理，则需要在添加节点时设置节点容量。

Kubernetes调度程序确保节点上的所有pod都有足够的资源。 它检查节点上容器请求的总和不大于节点容量。 它包括由kubelet启动的所有容器，但不包括由容器运行时直接启动的容器，也不包括在容器外部运行的任何进程。