组件工具

1 - 特性门控

本页详述了管理员可以在不同的 Kubernetes 组件上指定的各种特性门控。

关于特性各个阶段的说明，请参见特性阶段。

概述

特性门控是描述 Kubernetes 特性的一组键值对。你可以在 Kubernetes 的各个组件中使用 `--feature-gates` flag 来启用或禁用这些特性。

每个 Kubernetes 组件都支持启用或禁用与该组件相关的一组特性门控。 使用 `-h` 参数来查看所有组件支持的完整特性门控。 要为诸如 kubelet 之类的组件设置特性门控，请使用 `--feature-gates` 参数，并向其 传递一个特性设置键值对列表：

```
--feature-gates="...,GracefulNodeShutdown=true"
```

下表总结了在不同的 Kubernetes 组件上可以设置的特性门控。

- 引入特性或更改其发布阶段后，"开始（Since）" 列将包含 Kubernetes 版本。
- "结束（Until）" 列（如果不为空）包含最后一个 Kubernetes 版本，你仍可以在其中使用特性门控。
- 如果某个特性处于 Alpha 或 Beta 状态，你可以在 Alpha 和 Beta 特性门控表中找到该特性。
- 如果某个特性处于稳定状态，你可以在 已毕业和废弃特性门控表 中找到该特性的所有阶段。
- 已毕业和废弃特性门控表 还列出了废弃的和已被移除的特性。

Alpha 和 Beta 状态的特性门控

特性	默认值	状态	开始（Since）	结束（Until）
APIListChunking	false	Alpha	1.8	1.8
APIListChunking	true	Beta	1.9	
APIPriorityAndFairness	false	Alpha	1.17	1.19

特性	默认值	状态	开始 (Since)	结束 (Until)
APIPriorityAndFairness	true	Beta	1.20	
APIResponseCompression	false	Alpha	1.7	1.15
APIResponseCompression	true	Beta	1.16	
APIServerIdentity	false	Alpha	1.20	
APIServerTracing	false	Alpha	1.22	
AllowInsecureBackendProxy	true	Beta	1.17	
AnyVolumeDataSource	false	Alpha	1.18	
AppArmor	true	Beta	1.4	
ControllerManagerLeaderMigration	false	Alpha	1.21	
CPUManager	false	Alpha	1.8	1.9
CPUManager	true	Beta	1.10	
CPUManagerPolicyOptions	false	Alpha	1.22	
CSIInlineVolume	false	Alpha	1.15	1.15
CSIInlineVolume	true	Beta	1.16	-
CSIMigration	false	Alpha	1.14	1.16
CSIMigration	true	Beta	1.17	
CSIMigrationAWS	false	Alpha	1.14	
CSIMigrationAWS	false	Beta	1.17	
CSIMigrationAzureDisk	false	Alpha	1.15	1.18

特性	默认值	状态	开始 (Since)	结束 (Until)
CSIMigrationAzureDisk	false	Beta	1.19	
CSIMigrationAzureFile	false	Alpha	1.15	1.19
CSIMigrationAzureFile	false	Beta	1.21	
CSIMigrationGCE	false	Alpha	1.14	1.16
CSIMigrationGCE	false	Beta	1.17	
CSIMigrationOpenStack	false	Alpha	1.14	1.17
CSIMigrationOpenStack	true	Beta	1.18	
CSIMigrationvSphere	false	Beta	1.19	
CSIStorageCapacity	false	Alpha	1.19	1.20
CSIStorageCapacity	true	Beta	1.21	
CSIVolumeFSGroupPolicy	false	Alpha	1.19	1.19
CSIVolumeFSGroupPolicy	true	Beta	1.20	
CSIVolumeHealth	false	Alpha	1.21	
CSRDuration	true	Beta	1.22	
ConfigurableFSGroupPolicy	false	Alpha	1.18	1.19
ConfigurableFSGroupPolicy	true	Beta	1.20	
ControllerManagerLeaderMigration	false	Alpha	1.21	1.21
ControllerManagerLeaderMigration	true	Beta	1.22	
CustomCPUCFSQuotaPeriod	false	Alpha	1.12	

特性	默认值	状态	开始 (Since)	结束 (Until)
DaemonSetUpdateSurge	false	Alpha	1.21	1.21
DaemonSetUpdateSurge	true	Beta	1.22	
DefaultPodTopologySpread	false	Alpha	1.19	1.19
DefaultPodTopologySpread	true	Beta	1.20	
DelegateFSGroupToCSIDriver	false	Alpha	1.22	
DevicePlugins	false	Alpha	1.8	1.9
DevicePlugins	true	Beta	1.10	
DisableAcceleratorUsageMetrics	false	Alpha	1.19	1.19
DisableAcceleratorUsageMetrics	true	Beta	1.20	
DisableCloudProviders	false	Alpha	1.22	
DownwardAPIHugePages	false	Alpha	1.20	1.20
DownwardAPIHugePages	false	Beta	1.21	
EfficientWatchResumption	false	Alpha	1.20	1.20
EfficientWatchResumption	true	Beta	1.21	
EndpointSliceTerminatingCondition	false	Alpha	1.20	1.21
EndpointSliceTerminatingCondition	true	Beta	1.22	
EphemeralContainers	false	Alpha	1.16	
ExpandCSIVolumes	false	Alpha	1.14	1.15
ExpandCSIVolumes	true	Beta	1.16	

特性	默认值	状态	开始 (Since)	结束 (Until)
`ExpandedDNSConfig`	`false`	Alpha	1.22	
`ExpandInUsePersistentVolumes`	`false`	Alpha	1.11	1.14
`ExpandInUsePersistentVolumes`	`true`	Beta	1.15	
`ExpandPersistentVolumes`	`false`	Alpha	1.8	1.10
`ExpandPersistentVolumes`	`true`	Beta	1.11	
`ExperimentalHostUserNamespaceDefaulting`	`false`	Beta	1.5	
`GenericEphemeralVolume`	`false`	Alpha	1.19	1.20
`GenericEphemeralVolume`	`true`	Beta	1.21	
`GracefulNodeShutdown`	`false`	Alpha	1.20	1.20
`GracefulNodeShutdown`	`true`	Beta	1.21	
`HPAContainerMetrics`	`false`	Alpha	1.20	
`HPAScaleToZero`	`false`	Alpha	1.16	
`IndexedJob`	`false`	Alpha	1.21	1.21
`IndexedJob`	`true`	Beta	1.22	
`IngressClassNamespacedParams`	`false`	Alpha	1.21	1.21
`IngressClassNamespacedParams`	`true`	Beta	1.22	
`InTreePluginAWSUnregister`	`false`	Alpha	1.21	
`InTreePluginAzureDiskUnregister`	`false`	Alpha	1.21	
`InTreePluginAzureFileUnregister`	`false`	Alpha	1.21	

特性	默认值	状态	开始 (Since)	结束 (Until)
`InTreePluginGCEUnregister`	`false`	Alpha	1.21	
`InTreePluginOpenStackUnregister`	`false`	Alpha	1.21	
`InTreePluginvSphereUnregister`	`false`	Alpha	1.21	
`IPv6DualStack`	`false`	Alpha	1.15	1.20
`IPv6DualStack`	`true`	Beta	1.21	
`JobTrackingWithFinalizers`	`false`	Alpha	1.22	
`KubeletCredentialProviders`	`false`	Alpha	1.20	
`KubeletInUserNamespace`	`false`	Alpha	1.22	
`KubeletPodResourcesGetAllocatable`	`false`	Alpha	1.21	
`LocalStorageCapacityIsolation`	`false`	Alpha	1.7	1.9
`LocalStorageCapacityIsolation`	`true`	Beta	1.10	
`LocalStorageCapacityIsolationFSQuotaMonitoring`	`false`	Alpha	1.15	
`LogarithmicScaleDown`	`false`	Alpha	1.21	1.21
`LogarithmicScaleDown`	`true`	Beta	1.22	
`MemoryManager`	`false`	Alpha	1.21	1.21
`MemoryManager`	`true`	Beta	1.22	
`MemoryQoS`	`false`	Alpha	1.22	
`MixedProtocolLBService`	`false`	Alpha	1.20	
`NetworkPolicyEndPort`	`false`	Alpha	1.21	1.21

特性	默认值	状态	开始 (Since)	结束 (Until)
NetworkPolicyEndPort	true	Beta	1.22	
NodeSwap	false	Alpha	1.22	
NonPreemptingPriority	false	Alpha	1.15	1.18
NonPreemptingPriority	true	Beta	1.19	
PodDeletionCost	false	Alpha	1.21	1.21
PodDeletionCost	true	Beta	1.22	
PodAffinityNamespaceSelector	false	Alpha	1.21	1.21
PodAffinityNamespaceSelector	true	Beta	1.22	
PodOverhead	false	Alpha	1.16	1.17
PodOverhead	true	Beta	1.18	
PodSecurity	false	Alpha	1.22	
PreferNominatedNode	false	Alpha	1.21	1.21
PreferNominatedNode	true	Beta	1.22	
ProbeTerminationGracePeriod	false	Alpha	1.21	1.21
ProbeTerminationGracePeriod	false	Beta	1.22	
ProcMountType	false	Alpha	1.12	
ProxyTerminatingEndpoints	false	Alpha	1.22	
QOSReserved	false	Alpha	1.11	
ReadWriteOncePod	false	Alpha	1.22	

特性	默认值	状态	开始 (Since)	结束 (Until)
`RemainingItemCount`	`false`	Alpha	1.15	1.15
`RemainingItemCount`	`true`	Beta	1.16	
`RemoveSelfLink`	`false`	Alpha	1.16	1.19
`RemoveSelfLink`	`true`	Beta	1.20	
`RotateKubeletServerCertificate`	`false`	Alpha	1.7	1.11
`RotateKubeletServerCertificate`	`true`	Beta	1.12	
`SeccompDefault`	`false`	Alpha	1.22	
`ServiceInternalTrafficPolicy`	`false`	Alpha	1.21	1.21
`ServiceInternalTrafficPolicy`	`true`	Beta	1.22	
`ServiceLBNodePortControl`	`false`	Alpha	1.20	1.21
`ServiceLBNodePortControl`	`true`	Beta	1.22	
`ServiceLoadBalancerClass`	`false`	Alpha	1.21	1.21
`ServiceLoadBalancerClass`	`true`	Beta	1.22	
`SizeMemoryBackedVolumes`	`false`	Alpha	1.20	1.21
`SizeMemoryBackedVolumes`	`true`	Beta	1.22	
`StatefulSetMinReadySeconds`	`false`	Alpha	1.22	
`StorageVersionAPI`	`false`	Alpha	1.20	
`StorageVersionHash`	`false`	Alpha	1.14	1.14
`StorageVersionHash`	`true`	Beta	1.15	

特性	默认值	状态	开始 (Since)	结束 (Until)
`SuspendJob`	`false`	Alpha	1.21	1.21
`SuspendJob`	`true`	Beta	1.22	
`TTLAfterFinished`	`false`	Alpha	1.12	1.20
`TTLAfterFinished`	`true`	Beta	1.21	
`TopologyAwareHints`	`false`	Alpha	1.21	
`TopologyManager`	`false`	Alpha	1.16	1.17
`TopologyManager`	`true`	Beta	1.18	
`VolumeCapacityPriority`	`false`	Alpha	1.21	-
`WinDSR`	`false`	Alpha	1.14	
`WinOverlay`	`false`	Alpha	1.14	1.19
`WinOverlay`	`true`	Beta	1.20	
`WindowsHostProcessContainers`	`false`	Alpha	1.22	

已毕业和已废弃的特性门控

特性	默认值	状态	开始 (Since)	结束 (Until)
`Accelerators`	`false`	Alpha	1.6	1.10
`Accelerators`	-	Deprecated	1.11	-
`AdvancedAuditing`	`false`	Alpha	1.7	1.7
`AdvancedAuditing`	`true`	Beta	1.8	1.11
`AdvancedAuditing`	`true`	GA	1.12	-

特性	默认值	状态	开始 (Since)	结束 (Until)
`AffinityInAnnotations`	`false`	Alpha	1.6	1.7
`AffinityInAnnotations`	-	Deprecated	1.8	-
`AllowExtTrafficLocalEndpoints`	`false`	Beta	1.4	1.6
`AllowExtTrafficLocalEndpoints`	`true`	GA	1.7	-
`AttachVolumeLimit`	`false`	Alpha	1.11	1.11
`AttachVolumeLimit`	`true`	Beta	1.12	1.16
`AttachVolumeLimit`	`true`	GA	1.17	-
`BalanceAttachedNodeVolumes`	`false`	Alpha	1.11	1.21
`BalanceAttachedNodeVolumes`	`false`	Deprecated	1.22	
`BlockVolume`	`false`	Alpha	1.9	1.12
`BlockVolume`	`true`	Beta	1.13	1.17
`BlockVolume`	`true`	GA	1.18	-
`BoundServiceAccountTokenVolume`	`false`	Alpha	1.13	1.20
`BoundServiceAccountTokenVolume`	`true`	Beta	1.21	1.21
`BoundServiceAccountTokenVolume`	`true`	GA	1.22	-
`CRIContainerLogRotation`	`false`	Alpha	1.10	1.10
`CRIContainerLogRotation`	`true`	Beta	1.11	1.20
`CRIContainerLogRotation`	`true`	GA	1.21	-
`CSIBlockVolume`	`false`	Alpha	1.11	1.13
`CSIBlockVolume`	`true`	Beta	1.14	1.17
`CSIBlockVolume`	`true`	GA	1.18	-
`CSIDriverRegistry`	`false`	Alpha	1.12	1.13

特性	默认值	状态	开始 (Since)	结束 (Until)
`CSIDriverRegistry`	`true`	Beta	1.14	1.17
`CSIDriverRegistry`	`true`	GA	1.18	
`CSIMigrationAWSComplete`	`false`	Alpha	1.17	1.20
`CSIMigrationAWSComplete`	-	Deprecated	1.21	-
`CSIMigrationAzureDiskComplete`	`false`	Alpha	1.17	1.20
`CSIMigrationAzureDiskComplete`	-	Deprecated	1.21	-
`CSIMigrationAzureFileComplete`	`false`	Alpha	1.17	1.20
`CSIMigrationAzureFileComplete`	-	Deprecated	1.21	-
`CSIMigrationGCEComplete`	`false`	Alpha	1.17	1.20
`CSIMigrationGCEComplete`	-	Deprecated	1.21	-
`CSIMigrationOpenStackComplete`	`false`	Alpha	1.17	1.20
`CSIMigrationOpenStackComplete`	-	Deprecated	1.21	-
`CSIMigrationvSphereComplete`	`false`	Beta	1.19	1.21
`CSIMigrationvSphereComplete`	-	Deprecated	1.22	-
`CSINodeInfo`	`false`	Alpha	1.12	1.13
`CSINodeInfo`	`true`	Beta	1.14	1.16
`CSINodeInfo`	`true`	GA	1.17	
`CSIPersistentVolume`	`false`	Alpha	1.9	1.9
`CSIPersistentVolume`	`true`	Beta	1.10	1.12
`CSIPersistentVolume`	`true`	GA	1.13	-
`CSIServiceAccountToken`	`false`	Alpha	1.20	1.20

特性	默认值	状态	开始 (Since)	结束 (Until)
CSIServiceAccountToken	true	Beta	1.21	1.21
CSIServiceAccountToken	true	GA	1.22	
CronJobControllerV2	false	Alpha	1.20	1.20
CronJobControllerV2	true	Beta	1.21	1.21
CronJobControllerV2	true	GA	1.22	-
CustomPodDNS	false	Alpha	1.9	1.9
CustomPodDNS	true	Beta	1.10	1.13
CustomPodDNS	true	GA	1.14	-
CustomResourceDefaulting	false	Alpha	1.15	1.15
CustomResourceDefaulting	true	Beta	1.16	1.16
CustomResourceDefaulting	true	GA	1.17	-
CustomResourcePublishOpenAPI	false	Alpha	1.14	1.14
CustomResourcePublishOpenAPI	true	Beta	1.15	1.15
CustomResourcePublishOpenAPI	true	GA	1.16	-
CustomResourceSubresources	false	Alpha	1.10	1.10
CustomResourceSubresources	true	Beta	1.11	1.15
CustomResourceSubresources	true	GA	1.16	-
CustomResourceValidation	false	Alpha	1.8	1.8
CustomResourceValidation	true	Beta	1.9	1.15
CustomResourceValidation	true	GA	1.16	-
CustomResourceWebhookConversion	false	Alpha	1.13	1.14
CustomResourceWebhookConversion	true	Beta	1.15	1.15
CustomResourceWebhookConversion	true	GA	1.16	-

特性	默认值	状态	开始 (Since)	结束 (Until)
DryRun	false	Alpha	1.12	1.12
DryRun	true	Beta	1.13	1.18
DryRun	true	GA	1.19	-
DynamicAuditing	false	Alpha	1.13	1.18
DynamicAuditing	-	Deprecated	1.19	-
DynamicKubeletConfig	false	Alpha	1.4	1.10
DynamicKubeletConfig	true	Beta	1.11	1.21
DynamicKubeletConfig	false	Deprecated	1.22	-
DynamicProvisioningScheduling	false	Alpha	1.11	1.11
DynamicProvisioningScheduling	-	Deprecated	1.12	-
DynamicVolumeProvisioning	true	Alpha	1.3	1.7
DynamicVolumeProvisioning	true	GA	1.8	-
EnableAggregatedDiscoveryTimeout	true	Deprecated	1.16	-
EnableEquivalenceClassCache	false	Alpha	1.8	1.14
EnableEquivalenceClassCache	-	Deprecated	1.15	-
EndpointSlice	false	Alpha	1.16	1.16
EndpointSlice	false	Beta	1.17	1.17
EndpointSlice	true	Beta	1.18	1.20
EndpointSlice	true	GA	1.21	-
EndpointSliceNodeName	false	Alpha	1.20	1.20
EndpointSliceNodeName	true	GA	1.21	-

特性	默认值	状态	开始 (Since)	结束 (Until)
`EndpointSliceProxying`	`false`	Alpha	1.18	1.18
`EndpointSliceProxying`	`true`	Beta	1.19	1.21
`EndpointSliceProxying`	`true`	GA	1.22	-
`ExperimentalCriticalPodAnnotation`	`false`	Alpha	1.5	1.12
`ExperimentalCriticalPodAnnotation`	`false`	Deprecated	1.13	-
`EvenPodsSpread`	`false`	Alpha	1.16	1.17
`EvenPodsSpread`	`true`	Beta	1.18	1.18
`EvenPodsSpread`	`true`	GA	1.19	-
`ExecProbeTimeout`	`true`	GA	1.20	-
`ExternalPolicyForExternalIP`	`true`	GA	1.18	-
`GCERegionalPersistentDisk`	`true`	Beta	1.10	1.12
`GCERegionalPersistentDisk`	`true`	GA	1.13	-
`HugePageStorageMediumSize`	`false`	Alpha	1.18	1.18
`HugePageStorageMediumSize`	`true`	Beta	1.19	1.21
`HugePageStorageMediumSize`	`true`	GA	1.22	-
`HugePages`	`false`	Alpha	1.8	1.9
`HugePages`	`true`	Beta	1.10	1.13
`HugePages`	`true`	GA	1.14	-
`HugePageStorageMediumSize`	`false`	Alpha	1.18	1.18
`HugePageStorageMediumSize`	`true`	Beta	1.19	1.21
`HugePageStorageMediumSize`	`true`	GA	1.22	-
`HyperVContainer`	`false`	Alpha	1.10	1.19
`HyperVContainer`	`false`	Deprecated	1.20	-

特性	默认值	状态	开始 (Since)	结束 (Until)
`ImmutableEphemeralVolumes`	`false`	Alpha	1.18	1.18
`ImmutableEphemeralVolumes`	`true`	Beta	1.19	1.20
`ImmutableEphemeralVolumes`	`true`	GA	1.21	
`Initializers`	`false`	Alpha	1.7	1.13
`Initializers`	-	Deprecated	1.14	-
`KubeletConfigFile`	`false`	Alpha	1.8	1.9
`KubeletConfigFile`	-	Deprecated	1.10	-
`KubeletPluginsWatcher`	`false`	Alpha	1.11	1.11
`KubeletPluginsWatcher`	`true`	Beta	1.12	1.12
`KubeletPluginsWatcher`	`true`	GA	1.13	-
`KubeletPodResources`	`false`	Alpha	1.13	1.14
`KubeletPodResources`	`true`	Beta	1.15	
`KubeletPodResources`	`true`	GA	1.20	
`LegacyNodeRoleBehavior`	`false`	Alpha	1.16	1.18
`LegacyNodeRoleBehavior`	`true`	Beta	1.19	1.20
`LegacyNodeRoleBehavior`	`false`	GA	1.21	-
`MountContainers`	`false`	Alpha	1.9	1.16
`MountContainers`	`false`	Deprecated	1.17	-
`MountPropagation`	`false`	Alpha	1.8	1.9
`MountPropagation`	`true`	Beta	1.10	1.11
`MountPropagation`	`true`	GA	1.12	-
`NodeDisruptionExclusion`	`false`	Alpha	1.16	1.18

特性	默认值	状态	开始 (Since)	结束 (Until)
NodeDisruptionExclusion	true	Beta	1.19	1.20
NodeDisruptionExclusion	true	GA	1.21	-
NodeLease	false	Alpha	1.12	1.13
NodeLease	true	Beta	1.14	1.16
NodeLease	true	GA	1.17	-
NamespaceDefaultLabelName	true	Beta	1.21	1.21
NamespaceDefaultLabelName	true	GA	1.22	-
PVCProtection	false	Alpha	1.9	1.9
PVCProtection	-	Deprecated	1.10	-
PersistentLocalVolumes	false	Alpha	1.7	1.9
PersistentLocalVolumes	true	Beta	1.10	1.13
PersistentLocalVolumes	true	GA	1.14	-
PodDisruptionBudget	false	Alpha	1.3	1.4
PodDisruptionBudget	true	Beta	1.5	1.20
PodDisruptionBudget	true	GA	1.21	-
PodPriority	false	Alpha	1.8	1.10
PodPriority	true	Beta	1.11	1.13
PodPriority	true	GA	1.14	-
PodReadinessGates	false	Alpha	1.11	1.11
PodReadinessGates	true	Beta	1.12	1.13
PodReadinessGates	true	GA	1.14	-
PodShareProcessNamespace	false	Alpha	1.10	1.11
PodShareProcessNamespace	true	Beta	1.12	1.16
PodShareProcessNamespace	true	GA	1.17	-

特性	默认值	状态	开始 (Since)	结束 (Until)
RequestManagement	false	Alpha	1.15	1.16
RequestManagement	-	Derecated	1.17	-
ResourceLimitsPriorityFunction	false	Alpha	1.9	1.18
ResourceLimitsPriorityFunction	-	Deprecated	1.19	-
ResourceQuotaScopeSelectors	false	Alpha	1.11	1.11
ResourceQuotaScopeSelectors	true	Beta	1.12	1.16
ResourceQuotaScopeSelectors	true	GA	1.17	-
RootCAConfigMap	false	Alpha	1.13	1.19
RootCAConfigMap	true	Beta	1.20	1.20
RootCAConfigMap	true	GA	1.21	-
RotateKubeletClientCertificate	true	Beta	1.8	1.18
RotateKubeletClientCertificate	true	GA	1.19	-
RunAsGroup	true	Beta	1.14	1.20
RunAsGroup	true	GA	1.21	-
RuntimeClass	false	Alpha	1.12	1.13
RuntimeClass	true	Beta	1.14	1.19
RuntimeClass	true	GA	1.20	-
SCTPSupport	false	Alpha	1.12	1.18
SCTPSupport	true	Beta	1.19	1.19
SCTPSupport	true	GA	1.20	-
ScheduleDaemonSetPods	false	Alpha	1.11	1.11
ScheduleDaemonSetPods	true	Beta	1.12	1.16
ScheduleDaemonSetPods	true	GA	1.17	-

特性	默认值	状态	开始 (Since)	结束 (Until)
SelectorIndex	false	Alpha	1.18	1.18
SelectorIndex	true	Beta	1.19	1.19
SelectorIndex	true	GA	1.20	-
ServerSideApply	false	Alpha	1.14	1.15
ServerSideApply	true	Beta	1.16	1.21
ServerSideApply	true	GA	1.22	-
ServiceAccountIssuerDiscovery	false	Alpha	1.18	1.19
ServiceAccountIssuerDiscovery	true	Beta	1.20	1.20
ServiceAccountIssuerDiscovery	true	GA	1.21	-
ServiceAppProtocol	false	Alpha	1.18	1.18
ServiceAppProtocol	true	Beta	1.19	1.19
ServiceAppProtocol	true	GA	1.20	-
ServiceLoadBalancerFinalizer	false	Alpha	1.15	1.15
ServiceLoadBalancerFinalizer	true	Beta	1.16	1.16
ServiceLoadBalancerFinalizer	true	GA	1.17	-
ServiceNodeExclusion	false	Alpha	1.8	1.18
ServiceNodeExclusion	true	Beta	1.19	1.20
ServiceNodeExclusion	true	GA	1.21	-
ServiceTopology	false	Alpha	1.17	1.19
ServiceTopology	false	Deprecated	1.20	-
SetHostnameAsFQDN	false	Alpha	1.19	1.19
SetHostnameAsFQDN	true	Beta	1.20	1.21
SetHostnameAsFQDN	true	GA	1.22	-

特性	默认值	状态	开始 (Since)	结束 (Until)
`StartupProbe`	`false`	Alpha	1.16	1.17
`StartupProbe`	`true`	Beta	1.18	1.19
`StartupProbe`	`true`	GA	1.20	-
`StorageObjectInUseProtection`	`true`	Beta	1.10	1.10
`StorageObjectInUseProtection`	`true`	GA	1.11	-
`StreamingProxyRedirects`	`false`	Beta	1.5	1.5
`StreamingProxyRedirects`	`true`	Beta	1.6	1.17
`StreamingProxyRedirects`	`true`	Deprecated	1.18	1.21
`StreamingProxyRedirects`	`false`	Deprecated	1.22	-
`SupportIPVSProxyMode`	`false`	Alpha	1.8	1.8
`SupportIPVSProxyMode`	`false`	Beta	1.9	1.9
`SupportIPVSProxyMode`	`true`	Beta	1.10	1.10
`SupportIPVSProxyMode`	`true`	GA	1.11	-
`SupportNodePidsLimit`	`false`	Alpha	1.14	1.14
`SupportNodePidsLimit`	`true`	Beta	1.15	1.19
`SupportNodePidsLimit`	`true`	GA	1.20	-
`SupportPodPidsLimit`	`false`	Alpha	1.10	1.13
`SupportPodPidsLimit`	`true`	Beta	1.14	1.19
`SupportPodPidsLimit`	`true`	GA	1.20	-
`Sysctls`	`true`	Beta	1.11	1.20
`Sysctls`	`true`	GA	1.21	
`TaintBasedEvictions`	`false`	Alpha	1.6	1.12
`TaintBasedEvictions`	`true`	Beta	1.13	1.17
`TaintBasedEvictions`	`true`	GA	1.18	-

特性	默认值	状态	开始 (Since)	结束 (Until)
TaintNodesByCondition	false	Alpha	1.8	1.11
TaintNodesByCondition	true	Beta	1.12	1.16
TaintNodesByCondition	true	GA	1.17	-
TokenRequest	false	Alpha	1.10	1.11
TokenRequest	true	Beta	1.12	1.19
TokenRequest	true	GA	1.20	-
TokenRequestProjection	false	Alpha	1.11	1.11
TokenRequestProjection	true	Beta	1.12	1.19
TokenRequestProjection	true	GA	1.20	-
ValidateProxyRedirects	false	Alpha	1.12	1.13
ValidateProxyRedirects	true	Beta	1.14	1.21
ValidateProxyRedirects	true	Deprecated	1.22	-
VolumePVCDataSource	false	Alpha	1.15	1.15
VolumePVCDataSource	true	Beta	1.16	1.17
VolumePVCDataSource	true	GA	1.18	-
VolumeScheduling	false	Alpha	1.9	1.9
VolumeScheduling	true	Beta	1.10	1.12
VolumeScheduling	true	GA	1.13	-
VolumeSnapshotDataSource	false	Alpha	1.12	1.16
VolumeSnapshotDataSource	true	Beta	1.17	1.19
VolumeSnapshotDataSource	true	GA	1.20	-
VolumeSubpath	true	GA	1.10	-
VolumeSubpathEnvExpansion	false	Alpha	1.14	1.14
VolumeSubpathEnvExpansion	true	Beta	1.15	1.16

特性	默认值	状态	开始 (Since)	结束 (Until)
`VolumeSubpathEnvExpansion`	`true`	GA	1.17	-
`WarningHeaders`	`true`	Beta	1.19	1.21
`WarningHeaders`	`true`	GA	1.22	-
`WatchBookmark`	`false`	Alpha	1.15	1.15
`WatchBookmark`	`true`	Beta	1.16	1.16
`WatchBookmark`	`true`	GA	1.17	-
`WindowsEndpointSliceProxying`	`false`	Alpha	1.19	1.20
`WindowsEndpointSliceProxying`	`true`	Beta	1.21	1.21
`WindowsEndpointSliceProxying`	`true`	GA	1.22	-
`WindowsGMSA`	`false`	Alpha	1.14	1.15
`WindowsGMSA`	`true`	Beta	1.16	1.17
`WindowsGMSA`	`true`	GA	1.18	-
`WindowsHostProcessContainers`	`false`	Alpha	1.22	
`WindowsRunAsUserName`	`false`	Alpha	1.16	1.16
`WindowsRunAsUserName`	`true`	Beta	1.17	1.17
`WindowsRunAsUserName`	`true`	GA	1.18	-

使用特性

特性阶段

处于 *Alpha* 、*Beta* 、 *GA* 阶段的特性。

Alpha 特性代表：

- 默认禁用。
- 可能有错误，启用此特性可能会导致错误。
- 随时可能删除对此特性的支持，恕不另行通知。
- 在以后的软件版本中，API 可能会以不兼容的方式更改，恕不另行通知。
- 建议将其仅用于短期测试中，因为开启特性会增加错误的风险，并且缺乏长期支持。

Beta 特性代表：

- 默认启用。
- 该特性已经经过良好测试。启用该特性是安全的。
- 尽管详细信息可能会更改，但不会放弃对整体特性的支持。

- 对象的架构或语义可能会在随后的 Beta 或稳定版本中以不兼容的方式更改。当发生这种情况时，我们将提供迁移到下一版本的说明。此特性可能需要删除、编辑和重新创建 API 对象。编辑过程可能需要慎重操作，因为这可能会导致依赖该特性的应用程序停机。
- 推荐仅用于非关键业务用途，因为在后续版本中可能会发生不兼容的更改。如果你具有多个可以独立升级的，则可以放宽此限制。

> **说明：**
> 请试用 *Beta* 特性并提供相关反馈！ 一旦特性结束 Beta 状态，我们就不太可能再对特性进行大幅修改。

General Availability (GA) 特性也称为 *稳定* 特性，*GA* 特性代表着：

- 此特性会一直启用；你不能禁用它。
- 不再需要相应的特性门控。
- 对于许多后续版本，特性的稳定版本将出现在发行的软件中。

特性门控列表

每个特性门控均用于启用或禁用某个特定的特性：

- `APIListChunking`：启用 API 客户端以块的形式从 API 服务器检索（“LIST” 或 “GET”）资源。
- `APIPriorityAndFairness`：在每个服务器上启用优先级和公平性来管理请求并发。（由 `RequestManagement` 重命名而来）
- `APIResponseCompression`：压缩 “LIST” 或 “GET” 请求的 API 响应。
- `APIServerIdentity`：为集群中的每个 API 服务器赋予一个 ID。
- `APIServerTracing`：为集群中的每个 API 服务器添加对分布式跟踪的支持。
- `Accelerators`：使用 Docker 时启用 Nvidia GPU 支持。
- `AdvancedAuditing`：启用高级审计功能。
- `AffinityInAnnotations`：启用 Pod 亲和或反亲和。
- `AllowExtTrafficLocalEndpoints`：启用服务用于将外部请求路由到节点本地终端。
- `AllowInsecureBackendProxy`：允许用户在执行 Pod 日志访问请求时跳过 TLS 验证。
- `AnyVolumeDataSource`：允许使用任何自定义的资源来做作为 PVC 中的 `DataSource`.
- `AppArmor`：使用 Docker 时，在 Linux 节点上启用基于 AppArmor 机制的强制访问控制。 请参见 AppArmor 教程 获取详细信息。
- `AttachVolumeLimit`：启用卷插件用于报告可连接到节点的卷数限制。有关更多详细信息，请参阅 动态卷限制。
- `BalanceAttachedNodeVolumes`：在进行平衡资源分配的调度时，考虑节点上的卷数。 调度器在决策时会优先考虑 CPU、内存利用率和卷数更近的节点。
- `BlockVolume`：在 Pod 中启用原始块设备的定义和使用。有关更多详细信息，请参见 原始块卷支持。
- `BoundServiceAccountTokenVolume`：迁移 ServiceAccount 卷以使用由 ServiceAccountTokenVolumeProjection 组成的投射卷。集群管理员可以使用 `serviceaccount_stale_tokens_total` 度量值来监控依赖于扩展令牌的负载。 如果没有这种类型的负载，你可以在启动 `kube-apiserver` 时添加 `--service-account-extend-token-expiration=false` 参数关闭扩展令牌。查看 绑定服务账号令牌 获取更多详细信息。
- `ControllerManagerLeaderMigration`：为 kube-controller-manager 和 cloud-controller-manager 启用 Leader 迁移，它允许集群管理者在没有停机的高可用集群环境下，实时 把 kube-controller-manager 迁移迁移到外部的 controller-manager (例如 cloud-controller-manager) 中。
- `CPUManager`：启用容器级别的 CPU 亲和性支持，有关更多详细信息，请参见 CPU 管理策略。
- `CRIContainerLogRotation`：为 CRI 容器运行时启用容器日志轮换。日志文件的默认最大大小为 10MB，缺省情况下，一个容器允许的最大日志文件数为5。这些值可以在kubelet配置中配置。 更多细节请参见 日志架构。
- `CPUManagerPolicyOptions`：允许微调 CPU 管理策略。

- `CSIBlockVolume`：启用外部 CSI 卷驱动程序用于支持块存储。有关更多详细信息，请参见 csi 原始块卷支持。
- `CSIDriverRegistry`：在 csi.storage.k8s.io 中启用与 CSIDriver API 对象有关的所有逻辑。
- `CSIInlineVolume`：为 Pod 启用 CSI 内联卷支持。
- `CSIMigration`：确保封装和转换逻辑能够将卷操作从内嵌插件路由到相应的预安装 CSI 插件。
- `CSIMigrationAWS`：确保填充和转换逻辑能够将卷操作从 AWS-EBS 内嵌插件路由到 EBS CSI 插件。 如果节点未安装和配置 EBS CSI 插件，则支持回退到内嵌 EBS 插件。 这需要启用 CSIMigration 特性标志。
- `CSIMigrationAWSComplete`：停止在 kubelet 和卷控制器中注册 EBS 内嵌插件， 并启用 shims 和转换逻辑将卷操作从AWS-EBS 内嵌插件路由到 EBS CSI 插件。 这需要启用 CSIMigration 和 CSIMigrationAWS 特性标志，并在集群中的所有节点上安装和配置 EBS CSI 插件。该特性标志已被废弃，取而代之的是 `InTreePluginAWSUnregister` ，这会阻止注册 EBS 内嵌插件。
- `CSIMigrationAzureDisk`：确保填充和转换逻辑能够将卷操作从 Azure 磁盘内嵌插件路由到 Azure 磁盘 CSI 插件。如果节点未安装和配置 AzureDisk CSI 插件， 支持回退到内建 AzureDisk 插件。这需要启用 CSIMigration 特性标志。
- `CSIMigrationAzureDiskComplete`：停止在 kubelet 和卷控制器中注册 Azure 磁盘内嵌插件，并启用 shims 和转换逻辑以将卷操作从 Azure 磁盘内嵌插件路由到 AzureDisk CSI 插件。 这需要启用 CSIMigration 和 CSIMigrationAzureDisk 特性标志， 并在集群中的所有节点上安装和配置 AzureDisk CSI 插件。该特性标志已被废弃，取而代之的是 能防止注册内嵌 AzureDisk 插件的 `InTreePluginAzureDiskUnregister` 特性标志。
- `CSIMigrationAzureFile`：确保封装和转换逻辑能够将卷操作从 Azure 文件内嵌插件路由到 Azure 文件 CSI 插件。如果节点未安装和配置 AzureFile CSI 插件， 支持回退到内嵌 AzureFile 插件。这需要启用 CSIMigration 特性标志。
- `CSIMigrationAzureFileComplete`：停止在 kubelet 和卷控制器中注册 Azure-File 内嵌插件，并启用 shims 和转换逻辑以将卷操作从 Azure-File 内嵌插件路由到 AzureFile CSI 插件。 这需要启用 CSIMigration 和 CSIMigrationAzureFile 特性标志， 并在集群中的所有节点上安装和配置 AzureFile CSI 插件。该特性标志已被废弃，取而代之的是 能防止注册内嵌 AzureDisk 插件的 `InTreePluginAzureFileUnregister` 特性标志。
- `CSIMigrationGCE`：启用 shims 和转换逻辑，将卷操作从 GCE-PD 内嵌插件路由到 PD CSI 插件。如果节点未安装和配置 PD CSI 插件，支持回退到内嵌 GCE 插件。 这需要启用 CSIMigration 特性标志。
- `CSIMigrationGCEComplete`：停止在 kubelet 和卷控制器中注册 GCE-PD 内嵌插件， 并启用 shims 和转换逻辑以将卷操作从 GCE-PD 内嵌插件路由到 PD CSI 插件。 这需要启用 CSIMigration 和 CSIMigrationGCE 特性标志，并在集群中的所有节点上 安装和配置 PD CSI 插件。该特性标志已被废弃，取而代之的是 能防止注册内嵌 GCE PD 插件的 `InTreePluginGCEUnregister` 特性标志。
- `CSIMigrationOpenStack`：确保填充和转换逻辑能够将卷操作从 Cinder 内嵌插件路由到 Cinder CSI 插件。如果节点未安装和配置 Cinder CSI 插件，支持回退到内嵌 Cinder 插件。这需要启用 CSIMigration 特性标志。
- `CSIMigrationOpenStackComplete`：停止在 kubelet 和卷控制器中注册 Cinder 内嵌插件， 并启用 shims 和转换逻辑将卷操作从 Cinder 内嵌插件路由到 Cinder CSI 插件。 这需要启用 CSIMigration 和 CSIMigrationOpenStack 特性标志，并在集群中的所有节点上 安装和配置 Cinder CSI 插件。该特性标志已被废弃，取而代之的是 能防止注册内嵌 openstack cinder 插件的 `InTreePluginOpenStackUnregister` 特性标志。
- `CSIMigrationvSphere`：允许封装和转换逻辑将卷操作从 vSphere 内嵌插件路由到 vSphere CSI 插件。如果节点未安装和配置 vSphere CSI 插件，则支持回退到 vSphere 内嵌插件。这需要启用 CSIMigration 特性标志。
- `CSIMigrationvSphereComplete`：停止在 kubelet 和卷控制器中注册 vSphere 内嵌插件， 并启用 shims 和转换逻辑以将卷操作从 vSphere 内嵌插件路由到 vSphere CSI 插件。 这需要启用 CSIMigration 和 CSIMigrationvSphere 特性标志，并在集群中的所有节点上 安装和配置 vSphere CSI 插件。该特性标志已被废弃，取而代之的是 能防止注册内嵌 vsphere 插件的 `InTreePluginvSphereUnregister` 特性标志。
- `CSINodeInfo`：在 csi.storage.k8s.io 中启用与 CSINodeInfo API 对象有关的所有逻辑。
- `CSIPersistentVolume`：启用发现和挂载通过 CSI（容器存储接口） 兼容卷插件配置的卷。

- `CSIServiceAccountToken`：允许 CSI 驱动接收挂载卷目标 Pods 的服务账户令牌。 参阅令牌请求（Token Requests）。
- `CSIStorageCapacity`：使 CSI 驱动程序可以发布存储容量信息，并使 Kubernetes 调度程序在调度 Pod 时使用该信息。参见 存储容量。 详情请参见 `csi` 卷类型。
- `CSIVolumeFSGroupPolicy`：允许 CSIDrivers 使用 `fsGroupPolicy` 字段. 该字段能控制由 CSIDriver 创建的卷在挂载这些卷时是否支持卷所有权和权限修改。
- `CSIVolumeHealth`：启用对节点上的 CSI volume 运行状况监控的支持
- `CSRDuration`：允许客户端来通过请求 Kubernetes CSR API 签署的证书的持续时间。
- `ConfigurableFSGroupPolicy`： 在 Pod 中挂载卷时，允许用户为 fsGroup 配置卷访问权限和属主变更策略。请参见 为 Pod 配置卷访问权限和属主变更策略。
- `ControllerManagerLeaderMigration`：为 `kube-controller-manager` 和 `cloud-controller-manager` 开启 leader 迁移功能。
- `CronJobControllerV2`： 使用 CronJob 控制器的一种替代实现。否则，系统会选择同一控制器的 v1 版本。
- `CustomCPUCFSQuotaPeriod`： 使节点能够更改 kubelet 配置. 中的 `cpuCFSQuotaPeriod`。
- `CustomPodDNS`： 允许使用 Pod 的 `dnsConfig` 属性自定义其 DNS 设置。 更多详细信息，请参见 Pod 的 DNS 配置。
- `CustomResourceDefaulting`： 为 CRD 启用在其 OpenAPI v3 验证模式中提供默认值的支持。
- `CustomResourcePublishOpenAPI`： 启用 CRD OpenAPI 规范的发布。
- `CustomResourceSubresources`： 对于用 CustomResourceDefinition 创建的资源启用其 `/status` 和 `/scale` 子资源。
- `CustomResourceValidation`： 对于用 CustomResourceDefinition 创建的资源启用基于模式的验证。
- `CustomResourceWebhookConversion`： 对于用 CustomResourceDefinition 创建的资源启用基于 Webhook 的转换。
- `DaemonSetUpdateSurge`：使 DaemonSet 工作负载在每个节点的更新期间保持可用性。
- `DefaultPodTopologySpread`：启用 `PodTopologySpread` 调度插件来完成 默认的调度传播.
- `DelegateFSGroupToCSIDriver`：如果 CSI 驱动程序支持，则通过 NodeStageVolume 和 NodePublishVolume CSI 调用传递 `fsGroup` ，将应用 `fsGroup` 从 Pod 的 `securityContext` 的角色委托给驱动。
- `DevicePlugins`： 在节点上启用基于 设备插件的 资源制备。
- `DisableAcceleratorUsageMetrics`： 禁用 kubelet 收集加速器指标.
- `DisableCloudProviders`：禁用 `kube-apiserver`， `kube-controller-manager` 和 `kubelet` 组件的 `--cloud-provider` 标志相关 的所有功能。
- `DownwardAPIHugePages`： 允许在 下行（Downward）API 中使用巨页信息。
- `DryRun`： 启用在服务器端对请求进行 彩排（Dry Run）， 以便测试验证、合并和修改，同时避免提交更改。
- `DynamicAuditing`： 在 v1.19 版本前用于启用动态审计。
- `DynamicKubeletConfig`： 启用 kubelet 的动态配置。请参阅 重新配置 kubelet。
- `DynamicProvisioningScheduling`： 扩展默认调度器以了解卷拓扑并处理 PV 配置。 此特性已在 v1.12 中完全被 `VolumeScheduling` 特性取代。
- `DynamicVolumeProvisioning`： 启用持久化卷到 Pod 的 动态预配置。
- `EfficientWatchResumption`： 允许从存储发起的 bookmark（进度通知）事件被 通知到用户。此特性仅适用于 watch 操作。
- `EnableAggregatedDiscoveryTimeout`： 对聚集的发现调用启用五秒钟超时设置。
- `EnableEquivalenceClassCache`： 调度 Pod 时，使 scheduler 缓存节点的等效项。
- `EndpointSlice`： 启用 EndpointSlice 以实现可扩缩性和可扩展性更好的网络端点。 参阅启用 EndpointSlice。
- `EndpointSliceNodeName`： 允许使用 EndpointSlice 的 `nodeName` 字段。
- `EndpointSliceProxying`： 启用此特性门控时，Linux 上运行的 kube-proxy 会使用 EndpointSlices 而不是 Endpoints 作为其主要数据源，从而使得可扩缩性和性能 提升成为可能。参阅 启用 EndpointSlice。
- `EndpointSliceTerminatingCondition`： 允许使用 EndpointSlice 的 `terminating` 和 `serving` 状况字段。

- `EphemeralContainers`：启用添加 临时容器 到正在运行的 Pod 的特性。
- `EvenPodsSpread`：使 Pod 能够在拓扑域之间平衡调度。请参阅 Pod 拓扑扩展约束。
- `ExecProbeTimeout`：确保 kubelet 会遵从 exec 探针的超时值设置。 此特性门控的主要目的是方便你处理现有的、依赖于已被修复的缺陷的工作负载； 该缺陷导致 Kubernetes 会忽略 exec 探针的超时值设置。 参阅就绪态探针.
- `ExpandCSIVolumes`：启用扩展 CSI 卷。
- `ExpandedDNSConfig`：在 kubelet 和 kube-apiserver 上启用后， 允许更多的 DNS 搜索域和搜索域列表。 参阅 扩展 DNS 配置.
- `ExpandInUsePersistentVolumes`：启用扩充使用中的 PVC 的尺寸。请查阅 调整使用中的 PersistentVolumeClaim 的大小。
- `ExpandPersistentVolumes`：允许扩充持久卷。请查阅 扩展持久卷申领。
- `ExperimentalCriticalPodAnnotation`：启用将特定 Pod 注解为 *critical* 的方式，用于 确保其被调度。 从 v1.13 开始已弃用此特性，转而使用 Pod 优先级和抢占功能。
- `ExperimentalHostUserNamespaceDefaulting`：启用主机默认的用户名字空间。 这适用于使用其他主机名字空间、主机安装的容器，或具有特权或使用特定的非名字空间功能 （例如 MKNODE、SYS_MODULE 等）的容器。 如果在 Docker 守护程序中启用了用户名字空间重新映射，则启用此选项。
- `ExternalPolicyForExternalIP`： 修复 ExternalPolicyForExternalIP 没有应用于 Service ExternalIPs 的 bug。
- `GCERegionalPersistentDisk`：在 GCE 上启用带地理区域信息的 PD 特性。
- `GenericEphemeralVolume`：启用支持临时的内联卷，这些卷支持普通卷 （可以由第三方存储供应商提供、存储容量跟踪、从快照还原等等）的所有功能。请参见 临时卷。
- `GracefulNodeShutdown`：在 kubelet 中启用体面地关闭节点的支持。 在系统关闭时，kubelet 会尝试监测该事件并体面地终止节点上运行的 Pods。参阅 体面地关闭节点 以了解更多细节。
- `HPAContainerMetrics`：允许 `HorizontalPodAutoscaler` 基于目标 Pods 中各容器 的度量值来执行扩缩操作。
- `HPAScaleToZero`：使用自定义指标或外部指标时，可将 `HorizontalPodAutoscaler` 资源的 `minReplicas` 设置为 0。
- `HugePages`：启用分配和使用预分配的 巨页资源。
- `HugePageStorageMediumSize`：启用支持多种大小的预分配 巨页资源。
- `HyperVContainer`：为 Windows 容器启用 Hyper-V 隔离。
- `ImmutableEphemeralVolumes`：允许将各个 Secret 和 ConfigMap 标记为不可变更的， 以提高安全性和性能。
- `InTreePluginAWSUnregister`：在 kubelet 和 卷控制器上关闭注册 aws-ebs 内嵌插件。
- `InTreePluginAzureDiskUnregister`：在 kubelet 和 卷控制器上关闭注册 azuredisk 内嵌插件。
- `InTreePluginAzureFileUnregister`：在 kubelet 和 卷控制器上关闭注册 azurefile 内嵌插件。
- `InTreePluginGCEUnregister`：在 kubelet 和 卷控制器上关闭注册 gce-pd 内嵌插件。
- `InTreePluginOpenStackUnregister`：在 kubelet 和 卷控制器上关闭注册 OpenStack cinder 内嵌插件。
- `InTreePluginvSphereUnregister`：在 kubelet 和 卷控制器上关闭注册 vSphere 内嵌插件。
- `IndexedJob`：允许 Job 控制器按每个完成的索引去管理 Pod 完成。
- `IngressClassNamespacedParams`：允许引用命名空间范围的参数引用 `IngressClass` 资源。该特性增加了两个字段 —— `Scope` 和 `Namespace` 到 `IngressClass.spec.parameters`。
- `Initializers`： 使用 Initializers 准入插件允许异步协调对象创建。
- `IPv6DualStack`：启用 双协议栈 以支持 IPv6。
- `JobTrackingWithFinalizers`：启用跟踪 Job 完成情况，而不是永远从集群剩余 pod 来获取信息判断完成情况。Job 控制器使 用 Pod finalizers 和 Job 状态中的一个字段来跟踪已完成的 Pod 以计算完成。
- `KubeletConfigFile`：启用从使用配置文件指定的文件中加载 kubelet 配置。 有关更多详细信息，请参见 通过配置文件设置 kubelet 参数。
- `KubeletCredentialProviders`：允许使用 kubelet exec 凭据提供程序来设置 镜像拉取凭据。
- `KubeletInUserNamespace`：支持在 user namespace 里运行 kubelet 。 请参见 使用非 Root 用户来运行 Kubernetes 节点组件.
- `KubeletPluginsWatcher`：启用基于探针的插件监视应用程序，使 kubelet 能够发现 类似 CSI 卷驱动程序这类插件。

- `KubeletPodResources`：启用 kubelet 上 Pod 资源 GRPC 端点。更多详细信息，请参见 支持设备监控。
- `KubeletPodResourcesGetAllocatable`：启用 kubelet 的 pod 资源 的 `GetAllocatableResources` 功能。 该 API 增强了[资源分配报告]（/zh/docs/concepts/extend-kubernetes/compute-storage-net/device-plugins/#monitoring-device-plugin-resources）包含有关可分配资源的信息，使客户端能够正确跟踪节点上的可用计算资源。
- `LegacyNodeRoleBehavior`：禁用此门控时，服务负载均衡器中和节点干扰中的原先行为 会忽略 `node-role.kubernetes.io/master` 标签，使用 `NodeDisruptionExclusion` 和 `ServiceNodeExclusion` 对应特性所提供的标签。
- `LocalStorageCapacityIsolation`：允许使用 本地临时存储 以及 emptyDir 卷 的 `sizeLimit` 属性。
- `LocalStorageCapacityIsolationFSQuotaMonitoring`：如果 本地临时存储 启用了 `LocalStorageCapacityIsolation`，并且 emptyDir 卷 的后备文件系统支持项目配额，并且启用了这些配额，将使用项目配额来监视 emptyDir 卷的存储消耗 而不是遍历文件系统，以此获得更好的性能和准确性。
- `LogarithmicScaleDown`：启用 Pod 的半随机（semi-random）选择，控制器将根据 Pod 时间戳的对数桶按比例缩小去驱逐 Pod。
- `MemoryManager`：允许基于 NUMA 拓扑为容器设置内存亲和性。
- `MemoryQoS`：使用 cgroup v2 内存控制器在 pod / 容器上启用内存保护和使用限制。
- `MixedProtocolLBService`：允许在同一 `LoadBalancer` 类型的 Service 实例中使用不同 的协议。
- `MountContainers`：允许使用主机上的工具容器作为卷挂载程序。
- `MountPropagation`：启用将一个容器安装的共享卷共享到其他容器或 Pod。 更多详细信息，请参见挂载传播。
- `NamespaceDefaultLabelName`：配置 API 服务器以在所有名字空间上设置一个不可变的 label `kubernetes.io/metadata.name`，也包括名字空间。
- `NodeDisruptionExclusion`：启用节点标签 `node.kubernetes.io/exclude-disruption`， 以防止在可用区发生故障期间驱逐节点。
- `NodeLease`：启用新的 Lease（租期）API 以报告节点心跳，可用作节点运行状况信号。
- `NodeSwap`：启用 kubelet 为节点上的 Kubernetes 工作负载分配交换内存的能力。 必须将 `KubeletConfiguration.failSwapOn` 设置为 false 的情况下才能使用。 更多详细信息，请参见 交换内存。
- `NonPreemptingPriority`：为 PriorityClass 和 Pod 启用 `preemptionPolicy` 选项。
- `PVCProtection`：启用防止仍被某 Pod 使用的 PVC 被删除的特性。
- `PodDeletionCost`：启用 Pod 删除成本 功能。 该功能使用户可以影响 ReplicaSet 的降序顺序。
- `PersistentLocalVolumes`：允许在 Pod 中使用 `local（本地）` 卷类型。 如果请求 `local` 卷，则必须指定 Pod 亲和性属性。
- `PodDisruptionBudget`：启用 PodDisruptionBudget 特性。
- `PodAffinityNamespaceSelector`：启用 Pod 亲和性名称空间选择器 和 CrossNamespacePodAffinity 资源配额功能。
- `PodOverhead`：启用 PodOverhead 特性以考虑 Pod 开销。
- `PodPriority`：根据 优先级 启用 Pod 的调度和抢占。
- `PodReadinessGates`：启用 `podReadinessGate` 字段的设置以扩展 Pod 准备状态评估。 有关更多详细信息，请参见 Pod 就绪状态判别。
- `PodSecurity`：开启 `PodSecurity` 准入控制插件。
- `PodShareProcessNamespace`：在 Pod 中启用 `shareProcessNamespace` 的设置， 以便在 Pod 中运行的容器之间共享同一进程名字空间。更多详细信息，请参见 在 Pod 中的容器间共享同一进程名字空间。
- `PreferNominatedNode`：这个标志告诉调度器在循环遍历集群中的所有其他节点 之前，是否首先检查指定的节点。
- `ProbeTerminationGracePeriod`：在 Pod 上 启用 设置探测器级别 `terminationGracePeriodSeconds`。 有关更多信息，请参见 enhancement proposal。
- `ProcMountType`：允许容器通过设置 SecurityContext 的 `procMount` 字段来控制 对 proc 文件系统的挂载方式。

- `ProxyTerminatingEndpoints`：当 `ExternalTrafficPolicy=Local` 时， 允许 kube-proxy 来处理终止过程中的端点。
- `QOSReserved`： 允许在 QoS 级别进行资源预留，以防止处于较低 QoS 级别的 Pod 突发进入处于较高 QoS 级别的请求资源（目前仅适用于内存）。
- `ReadWriteOncePod`：允许使用 `ReadWriteOncePod` 访问模式的 PersistentVolume。
- `RemainingItemCount`： 允许 API 服务器在 分块列表请求 的响应中显示剩余条目的个数。
- `RemoveSelfLink`： 将 ObjectMeta 和 ListMeta 中的 `selfLink` 字段废弃并删除。
- `RequestManagement`： 允许在每个 API 服务器上通过优先级和公平性管理请求并发性。 自 1.17 以来已被 `APIPriorityAndFairness` 弃用。
- `ResourceLimitsPriorityFunction`： 启用某调度器优先级函数， 该函数将最低得分 1 指派给至少满足输入 Pod 的 CPU 和内存限制之一的节点， 目的是打破得分相同的节点之间的关联。
- `ResourceQuotaScopeSelectors`： 启用资源配额范围选择器。
- `RootCAConfigMap`： 配置 `kube-controller-manager`，使之发布一个名为 `kube-root-ca.crt` 的 ConfigMap，到 所有名字空间中。该 ConfigMap 包含用来验证与 kube-apiserver 之间连接的 CA 证书包。参阅 绑定服务账户令牌 以了解更多细节。
- `RotateKubeletClientCertificate`： 在 kubelet 上启用客户端 TLS 证书的轮换。 更多详细信息，请参见 kubelet 配置。
- `RotateKubeletServerCertificate`： 在 kubelet 上启用服务器 TLS 证书的轮换。 更多详细信息，请参见 kubelet 配置。
- `RunAsGroup`： 启用对容器初始化过程中设置的主要组 ID 的控制。
- `RuntimeClass`： 启用 RuntimeClass 特性用于选择容器运行时配置。
- `ScheduleDaemonSetPods`： 启用 DaemonSet Pods 由默认调度程序而不是 DaemonSet 控制器进行调度。
- `SCTPSupport`： 在 Pod、Service、Endpoints、NetworkPolicy 定义中 允许将 *SCTP* 用作 `protocol` 值。
- `SeccompDefault`：允许将所有工作负载的默认 seccomp 配置文件为 `RuntimeDefault`。seccomp 配置在 Pod 或者容器的 `securityContext` 字段中指定。
- `SelectorIndex`：允许在 API 服务器 watch 的缓存中基于标签和字段的索引来加速 list 的操作。
- `ServerSideApply`： 在 API 服务器上启用 服务器端应用（SSA）。
- `ServiceAccountIssuerDiscovery`： 在 API 服务器中为服务帐户颁发者启用 OIDC 发现端点（颁发者和 JWKS URL）。详情参见 为 Pod 配置服务账户 。
- `ServiceAppProtocol`： 为 Service 和 Endpoints 启用 `appProtocol` 字段。
- `ServiceInternalTrafficPolicy`： 为服务启用 `internalTrafficPolicy` 字段。
- `ServiceLBNodePortControl`： 为服务启用 `allocateLoadBalancerNodePorts` 字段。
  `ServiceLoadBalancerClass`：为服务启用 `loadBalancerClass` 字段。 有关更多信息，请参见 负载均衡器类的定义 implementation。
- `ServiceLoadBalancerFinalizer`： 为服务负载均衡启用终结器（finalizers）保护。
- `ServiceNodeExclusion`： 启用从云提供商创建的负载均衡中排除节点。 如果节点标记有 `node.kubernetes.io/exclude-from-external-load-balancers`， 标签，则可以排除该节点。
- `ServiceTopology`： 启用服务拓扑可以让一个服务基于集群的节点拓扑进行流量路由。 有关更多详细信息，请参见 服务拓扑。
- `SetHostnameAsFQDN`： 启用将全限定域名（FQDN）设置为 Pod 主机名的功能。 请参见为 Pod 设置 `setHostnameAsFQDN` 字段。
- `SizeMemoryBackedVolumes`： 允许 kubelet 检查基于内存制备的卷的尺寸约束 （目前主要针对 `emptyDir` 卷）。
- `StartupProbe`： 在 kubelet 中启用 启动探针。
- `StatefulSetMinReadySeconds`：允许 StatefulSet 控制器采纳 `minReadySeconds` 设置。
- `StorageObjectInUseProtection`： 如果仍在使用 PersistentVolume 或 PersistentVolumeClaim 对象，则将其删除操作推迟。
- `StorageVersionAPI`：启用 存储版本 API。
- `StorageVersionHash`： 允许 API 服务器在版本发现中公开存储版本的哈希值。
- `StreamingProxyRedirects`： 指示 API 服务器拦截（并跟踪）后端（kubelet） 的重定向以处理流请求。 流请求的例子包括 `exec`、 `attach` 和 `port-forward` 请求。

- `SupportIPVSProxyMode`：启用使用 IPVS 提供内服务负载平衡。更多详细信息，请参见 服务代理。
- `SupportNodePidsLimit`：启用支持，限制节点上的 PID 用量。 `--system-reserved` 和 `--kube-reserved` 中的参数 `pid=<数值>` 可以分别用来 设定为整个系统所预留的进程 ID 个数和为 Kubernetes 系统守护进程预留的进程 ID 个数。
- `SupportPodPidsLimit`：启用支持限制 Pod 中的进程 PID。
- `SuspendJob`： 启用支持以暂停和恢复作业。 更多详细信息，请参见 Jobs 文档。
- `Sysctls`：允许为每个 Pod 设置的名字空间内核参数（sysctls）。 更多详细信息，请参见 sysctls。

- `TTLAfterFinished`：资源完成执行后，允许 TTL 控制器清理资源。
- `TaintBasedEvictions`：根据节点上的污点和 Pod 上的容忍度启用从节点驱逐 Pod 的特性。更多详细信息可参见污点和容忍度。
- `TaintNodesByCondition`：根据节点状况 启用自动为节点标记污点。
- `TokenRequest`：在服务帐户资源上启用 `TokenRequest` 端点。
- `TokenRequestProjection`：启用通过 `projected` 卷 将服务帐户令牌注入到 Pod 中的特性。
- `TopologyAwareHints`： 在 EndpointSlices 中启用基于拓扑提示的拓扑感知路由。 更多详细信息可参见Topology Aware Hints
- `TopologyManager`：启用一种机制来协调 Kubernetes 不同组件的细粒度硬件资源分配。 详见控制节点上的拓扑管理策略。

- `ValidateProxyRedirects`： 这个标志控制 API 服务器是否应该验证只跟随到相同的主机的重定向。 仅在启用 `StreamingProxyRedirects` 标志时被使用。
- `VolumeCapacityPriority`：基于可用 PV 容量的拓扑，启用对不同节点的优先级支持。
- `VolumePVCDataSource`：启用对将现有 PVC 指定数据源的支持。
- `VolumeScheduling`：启用卷拓扑感知调度，并使 PersistentVolumeClaim（PVC） 绑定能够了解调度决策；当与 PersistentLocalVolumes 特性门控一起使用时， 还允许使用 `local` 卷类型。
- `VolumeSnapshotDataSource`：启用卷快照数据源支持。
- `VolumeSubpath`： 允许在容器中挂载卷的子路径。

- `VolumeSubpathEnvExpansion`：启用 `subPathExpr` 字段用于将环境变量在 `subPath` 中展开。
- `WarningHeaders`：允许在 API 响应中发送警告头部。
- `WatchBookmark`：启用对 watch 操作中 bookmark 事件的支持。
- `WinDSR`：允许 kube-proxy 为 Windows 创建 DSR 负载均衡。
- `WinOverlay`：允许 kube-proxy 在 Windows 的覆盖网络模式下运行。
- `WindowsEndpointSliceProxying`：当启用时，运行在 Windows 上的 kube-proxy 将使用 EndpointSlices 而不是 Endpoints 作为主要数据源，从而实现可伸缩性和并改进性能。 详情请参见启用端点切片.
- `WindowsGMSA`：允许将 GMSA 凭据规范从 Pod 传递到容器运行时。
- `WindowsHostProcessContainers`：启用对 Windows HostProcess 容器的支持。
- `WindowsRunAsUserName`： 提供使用非默认用户在 Windows 容器中运行应用程序的支持。 详情请参见 配置 RunAsUserName。

接下来

- Kubernetes 的弃用策略 介绍了项目针对已移除特性和组件的处理方法。

2 - kubelet

简介

kubelet 是在每个 Node 节点上运行的主要 “节点代理”。它可以使用以下之一向 apiserver 注册：主机名（hostname）；覆盖主机名的参数；某云驱动的特定逻辑。

kubelet 是基于 PodSpec 来工作的。每个 PodSpec 是一个描述 Pod 的 YAML 或 JSON 对象。kubelet 接受通过各种机制（主要是通过 apiserver）提供的一组 PodSpec，并确保这些 PodSpec 中描述的容器处于运行状态且运行状况良好。 kubelet 不管理不是由 Kubernetes 创建的容器。

除了来自 apiserver 的 PodSpec 之外，还可以通过以下三种方式将容器清单（manifest）提供给 kubelet。

文件（File）：利用命令行参数传递路径。kubelet 周期性地监视此路径下的文件是否有更新。 监视周期默认为 20s，且可通过参数进行配置。

HTTP 端点（HTTP endpoint）：利用命令行参数指定 HTTP 端点。 此端点的监视周期默认为 20 秒，也可以使用参数进行配置。

HTTP 服务器（HTTP server）：kubelet 还可以侦听 HTTP 并响应简单的 API （目前没有完整规范）来提交新的清单。

```
kubelet [flags]
```

选项

--add-dir-header

设置为 true 表示将文件目录添加到日志消息的头部

--address ip　默认值：0.0.0.0

kubelet 用来提供服务的 IP 地址（设置为 `0.0.0.0` 表示使用所有 IPv4 接口， 设置为 `::` 表示使用所有 IPv6 接口）。已弃用：应在 `--config` 所给的 配置文件中进行设置。 (进一步了解)

--allowed-unsafe-sysctls strings

用逗号分隔的字符串序列设置允许使用的非安全的 sysctls 或 sysctl 模式（以 `*` 结尾）。 使用此参数时风险自担。已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解).

--alsologtostderr

设置为 true 表示将日志输出到文件的同时输出到 stderr

--anonymous-auth　默认值：true

设置为 true 表示 kubelet 服务器可以接受匿名请求。未被任何认证组件拒绝的请求将被视为匿名请求。 匿名请求的用户名为 `system:anonymous` ，用户组为 `system:unauthenticated` 。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--authentication-token-webhook

使用 `TokenReview` API 对持有者令牌进行身份认证。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--authentication-token-webhook-cache-ttl duration　默认值： `2m0s`

对 Webhook 令牌认证组件所返回的响应的缓存时间。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--authorization-mode string

kubelet 服务器的鉴权模式。可选值包括： `AlwaysAllow` 、 `Webhook` 。 `Webhook` 模式使用 `SubjectAccessReview` API 鉴权。 当 `--config` 参数未被设置时，默认值为 `AlwaysAllow` ，当使用了 `--config` 时，默认值为 `Webhook` 。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--authorization-webhook-cache-authorized-ttl duration 默认值： `5m0s`

对 Webhook 认证组件所返回的 “Authorized（已授权）” 应答的缓存时间。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--authorization-webhook-cache-unauthorized-ttl duration 默认值： `30s`

对 Webhook 认证组件所返回的 “Unauthorized（未授权）” 应答的缓存时间。 `--config` 时，默认值为 `Webhook` 。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--azure-container-registry-config string

包含 Azure 容器镜像库配置信息的文件的路径。

--bootstrap-kubeconfig string

某 kubeconfig 文件的路径，该文件将用于获取 kubelet 的客户端证书。 如果 `--kubeconfig` 所指定的文件不存在，则使用引导所用 kubeconfig 从 API 服务器请求客户端证书。成功后，将引用生成的客户端证书和密钥的 kubeconfig 写入 --kubeconfig 所指定的路径。客户端证书和密钥文件将存储在 `--cert-dir` 所指的目录。

--cert-dir string 默认值： `/var/lib/kubelet/pki`

TLS 证书所在的目录。如果设置了 `--tls-cert-file` 和 `--tls-private-key-file` ，则此标志将被忽略。

--cgroup-driver string 默认值： `cgroupfs`

kubelet 用来操作本机 cgroup 时使用的驱动程序。支持的选项包括 `cgroupfs` 和 `systemd` 。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解) /td>

--cgroup-root string 默认值： `""`

可选的选项，为 Pod 设置根 cgroup。容器运行时会尽可能使用此配置。 默认值 `""` 意味着将使用容器运行时的默认设置。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--cgroups-per-qos 默认值： `true`

启用创建 QoS cgroup 层次结构。此值为 true 时 kubelet 为 QoS 和 Pod 创建顶级的 cgroup。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--chaos-chance float

如果此值大于 0.0，则引入随机客户端错误和延迟。用于测试。 已启用：将在未来版本中移除。

--client-ca-file string

如果设置了此参数，则使用对应文件中机构之一检查请求中所携带的客户端证书。 若客户端证书通过身份认证，则其对应身份为其证书中所设置的 CommonName。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--cloud-config string

云驱动配置文件的路径。空字符串表示没有配置文件。 已弃用：将在 1.23 版本中移除，以便于从 kubelet 中去除云驱动代码。

--cloud-provider string

云服务的提供者。设置为空字符串表示在没有云驱动的情况下运行。 如果设置了此标志，则云驱动负责确定节点的名称（参考云提供商文档以确定是否以及如何使用主机名）。 已弃用：将在 1.23 版本中移除，以便于从 kubelet 中去除云驱动代码。

--cluster-dns strings

DNS 服务器的 IP 地址，以逗号分隔。此标志值用于 Pod 中设置了 “`dnsPolicy=ClusterFirst`”时为容器提供 DNS 服务。注意：列表中出现的所有 DNS 服务器必须包含相同的记录组， 否则集群中的名称解析可能无法正常工作。至于名称解析过程中会牵涉到哪些 DNS 服务器， 这一点无法保证。 `--config` 时，默认值为 `Webhook` 。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--cluster-domain string

集群的域名。如果设置了此值，kubelet 除了将主机的搜索域配置到所有容器之外，还会为其 配置所搜这里指定的域名。 `--config` 时，默认值为 `Webhook` 。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--cni-bin-dir string 默认值： `/opt/cni/bin`

<警告：alpha 特性> 此值为以逗号分隔的完整路径列表。 kubelet 将在所指定路径中搜索 CNI 插件的可执行文件。 仅当容器运行环境设置为 `docker` 时，此特定于 docker 的参数才有效。

--cni-cache-dir string 默认值： `/var/lib/cni/cache`

<警告：alpha 特性> 此值为一个目录的全路径名。CNI 将在其中缓存文件。 仅当容器运行环境设置为 `docker` 时，此特定于 docker 的参数才有效。

--cni-conf-dir string 默认值： `/etc/cni/net.d`

<警告：alpha 特性> 此值为某目录的全路径名。kubelet 将在其中搜索 CNI 配置文件。 仅当容器运行环境设置为 `docker` 时，此特定于 docker 的参数才有效。

--config string

kubelet 将从此标志所指的文件中加载其初始配置。此路径可以是绝对路径，也可以是相对路径。 相对路径按 kubelet 的当前工作目录起计。省略此参数时 kubelet 会使用内置的默认配置值。 命令行参数会覆盖此文件中的配置。

--container-log-max-files int32 默认值：5

设置容器的日志文件个数上限。此值必须不小于 2。 此标志只能与 `--container-runtime=remote` 标志一起使用。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--container-log-max-size string 默认值： `10Mi`

设置容器日志文件在轮换生成新文件时之前的最大值（例如， `10Mi` ）。 此标志只能与 `--container-runtime=remote` 标志一起使用。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--container-runtime string 默认值： `docker`

要使用的容器运行时。目前支持 `docker` 、 `remote` 。

--container-runtime-endpoint string 默认值： `unix:///var/run/dockershim.sock`

[实验性特性] 远程运行时服务的端点。目前支持 Linux 系统上的 UNIX 套接字和 Windows 系统上的 npipe 和 TCP 端点。例如： `unix:///var/run/dockershim.sock` 、 `npipe:////./pipe/dockershim` 。

--contention-profiling

当启用了性能分析时，启用锁竞争分析。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--cpu-cfs-quota 默认值： `true`

为设置了 CPU 限制的容器启用 CPU CFS 配额保障。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--cpu-cfs-quota-period duration 默认值： `100ms`

设置 CPU CFS 配额周期 `cpu.cfs_period_us` 。默认使用 Linux 内核所设置的默认值 。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--cpu-manager-policy string 默认值： `none`

要使用的 CPU 管理器策略。可选值包括： `none` 和 `static` 。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--cpu-manager-reconcile-period duration 默认值： `10s`

<警告：alpha 特性> 设置 CPU 管理器的调和时间。例如： `10s` 或者 `1m` 。 如果未设置，默认使用节点状态更新频率。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--docker-endpoint string 默认值： `unix:///var/run/docker.sock`

使用这里的端点与 docker 端点通信。 仅当容器运行环境设置为 `docker` 时，此特定于 docker 的参数才有效。

--dynamic-config-dir string

kubelet 使用此目录来保存所下载的配置，跟踪配置运行状况。 如果目录不存在，则 kubelet 创建该目录。此路径可以是绝对路径，也可以是相对路径。 相对路径从 kubelet 的当前工作目录计算。 设置此参数将启用动态 kubelet 配置。必须启用 `DynamicKubeletConfig` 特性门控之后才能设置此标志；由于此特性为 beta 阶段，对应的特性门控当前默认为 `true` 。

--enable-controller-attach-detach 默认值： `true`

启用 Attach/Detach 控制器来挂接和摘除调度到该节点的卷，同时禁用 kubelet 执行挂接和摘除操作。

--enable-debugging-handlers Default: `true`

启用服务器上用于日志收集和在本地运行容器和命令的端点。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--enable-server

启用 kubelet 服务器。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--enforce-node-allocatable strings Default: `pods`

用逗号分隔的列表，包含由 kubelet 强制执行的节点可分配资源级别。 可选配置为：`none`、`pods`、`system-reserved` 和 `kube-reserved`。 在设置 `system-reserved` 和 `kube-reserved` 这两个值时，同时要求设置 `--system-reserved-cgroup` 和 `--kube-reserved-cgroup` 这两个参数。 如果设置为 `none`，则不需要设置其他参数。 参考相关文档。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--event-burst int32　默认值：10

事件记录的个数的突发峰值上限，在遵从 `--event-qps` 阈值约束的前提下 临时允许事件记录达到此数目。仅在 `--event-qps` 大于 0 时使用。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--event-qps int32　Default: 5

设置大于 0 的值表示限制每秒可生成的事件数量。设置为 0 表示不限制。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--eviction-hard string　默认值：`imagefs.available<15%,memory.available<100Mi,nodefs.available<10%`

触发 Pod 驱逐操作的一组硬性门限（例如：`memory.available<1Gi` （内存可用值小于 1G））设置。在 Linux 节点上，默认值还包括 `nodefs.inodesFree<5%`。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--eviction-max-pod-grace-period int32

响应满足软性驱逐阈值（Soft Eviction Threshold）而终止 Pod 时使用的最长宽限期（以秒为单位）。 如果设置为负数，则遵循 Pod 的指定值。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--eviction-minimum-reclaim mapStringString

当某资源压力过大时，kubelet 将执行 Pod 驱逐操作。 此参数设置软性驱逐操作需要回收的资源的最小数量（例如：`imagefs.available=2Gi`）。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--eviction-pressure-transition-period duration　默认值：`5m0s`

kubelet 在驱逐压力状况解除之前的最长等待时间。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--eviction-soft mapStringString

设置一组驱逐阈值（例如：`memory.available<1.5Gi`）。 如果在相应的宽限期内达到该阈值，则会触发 Pod 驱逐操作。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--eviction-soft-grace-period mapStringString

设置一组驱逐宽限期（例如，`memory.available=1m30s`），对应于触发软性 Pod 驱逐操作之前软性驱逐阈值所需持续的时间长短。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--exit-on-lock-contention

设置为 true 表示当发生锁文件竞争时 kubelet 可以退出。

--experimental-allocatable-ignore-eviction　默认值：`false`

设置为 `true` 表示在计算节点可分配资源数量时忽略硬性逐出阈值设置。 参考 相关文档。 已启用：将在 1.23 版本中移除。

--experimental-bootstrap-kubeconfig string

已弃用：应使用 `--bootstrap-kubeconfig` 标志

--experimental-check-node-capabilities-before-mount

[实验性特性] 设置为 `true` 表示 kubelet 在进行挂载卷操作之前要 在本节点上检查所需的组件（如可执行文件等）是否存在。 已弃用：将在 1.23 版本中移除，以便使用 CSI。

--experimental-kernel-memcg-notification

设置为 true 表示 kubelet 将会集成内核的 memcg 通知机制而不是使用轮询机制来 判断是否达到了内存驱逐阈值。 此标志将在 1.23 版本移除。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--experimental-log-sanitization

[试验性功能] 启用此标志之后，kubelet 会避免将标记为敏感的字段（密码、密钥、令牌等）写入日志中。运行时的日志清理可能会带来相当的计算开销，因此不应该在 产品环境中启用。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--experimental-mounter-path string　默认值： `mount`

[实验性特性] 卷挂载器（mounter）的可执行文件的路径。设置为空表示使用默认挂载器 `mount` 。 已弃用：将在 1.23 版本移除以支持 CSI。

--fail-swap-on　默认值： `true`

设置为 true 表示如果主机启用了交换分区，kubelet 将直接失败。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--feature-gates mapStringBool

用于 alpha 实验性质的特性开关组，每个开关以 key=value 形式表示。当前可用开关包括：
APIListChunking=true|false (BETA - 默认值为 true)
APIPriorityAndFairness=true|false (BETA - 默认值为 true)
APIResponseCompression=true|false (BETA - 默认值为 true)
APIServerIdentity=true|false (ALPHA - 默认值为 false)
AllAlpha=true|false (ALPHA - 默认值为 false)
AllBeta=true|false (BETA - 默认值为 false)
AllowInsecureBackendProxy=true|false (BETA - 默认值为 true)
AnyVolumeDataSource=true|false (ALPHA - 默认值为 false)
AppArmor=true|false (BETA - 默认值为 true)
BalanceAttachedNodeVolumes=true|false (ALPHA - 默认值为 false)
BoundServiceAccountTokenVolume=true|false (ALPHA - 默认值为 false)
CPUManager=true|false (BETA - 默认值为 true)
CSIInlineVolume=true|false (BETA - 默认值为 true)
CSIMigration=true|false (BETA - 默认值为 true)
CSIMigrationAWS=true|false (BETA - 默认值为 false)
CSIMigrationAWSComplete=true|false (ALPHA - 默认值为 false)
CSIMigrationAzureDisk=true|false (BETA - 默认值为 false)
CSIMigrationAzureDiskComplete=true|false (ALPHA - 默认值为 false)
CSIMigrationAzureFile=true|false (ALPHA - 默认值为 false)
CSIMigrationAzureFileComplete=true|false (ALPHA - 默认值为 false)
CSIMigrationGCE=true|false (BETA - 默认值为 false)
CSIMigrationGCEComplete=true|false (ALPHA - 默认值为 false)
CSIMigrationOpenStack=true|false (BETA - 默认值为 false)
CSIMigrationOpenStackComplete=true|false (ALPHA - 默认值为 false)
CSIMigrationvSphere=true|false (BETA - 默认值为 false)
CSIMigrationvSphereComplete=true|false (BETA - 默认值为 false)
CSIServiceAccountToken=true|false (ALPHA - 默认值为 false)
CSIStorageCapacity=true|false (ALPHA - 默认值为 false)
CSIVolumeFSGroupPolicy=true|false (BETA - 默认值为 true)
ConfigurableFSGroupPolicy=true|false (BETA - 默认值为 true)
CronJobControllerV2=true|false (ALPHA - 默认值为 false)
CustomCPUCFSQuotaPeriod=true|false (ALPHA - 默认值为 false)
DefaultPodTopologySpread=true|false (BETA - 默认值为 true)
DevicePlugins=true|false (BETA - 默认值为 true)
DisableAcceleratorUsageMetrics=true|false (BETA - 默认值为 true)
DownwardAPIHugePages=true|false (ALPHA - 默认值为 false)

DynamicKubeletConfig=true|false (BETA - 默认值为 true)
EfficientWatchResumption=true|false (ALPHA - 默认值为 false)
EndpointSlice=true|false (BETA - 默认值为 true)
EndpointSliceNodeName=true|false (ALPHA - 默认值为 false)
EndpointSliceProxying=true|false (BETA - 默认值为 true)
EndpointSliceTerminatingCondition=true|false (ALPHA - 默认值为 false)
EphemeralContainers=true|false (ALPHA - 默认值为 false)
ExpandCSIVolumes=true|false (BETA - 默认值为 true)
ExpandInUsePersistentVolumes=true|false (BETA - 默认值为 true)
ExpandPersistentVolumes=true|false (BETA - 默认值为 true)
ExperimentalHostUserNamespaceDefaulting=true|false (BETA - 默认值为 false)
GenericEphemeralVolume=true|false (ALPHA - 默认值为 false)
GracefulNodeShutdown=true|false (ALPHA - 默认值为 false)
HPAContainerMetrics=true|false (ALPHA - 默认值为 false)
HPAScaleToZero=true|false (ALPHA - 默认值为 false)
HugePageStorageMediumSize=true|false (BETA - 默认值为 true)
IPv6DualStack=true|false (ALPHA - 默认值为 false)
ImmutableEphemeralVolumes=true|false (BETA - 默认值为 true)
KubeletCredentialProviders=true|false (ALPHA - 默认值为 false)
KubeletPodResources=true|false (BETA - 默认值为 true)
LegacyNodeRoleBehavior=true|false (BETA - 默认值为 true)
LocalStorageCapacityIsolation=true|false (BETA - 默认值为 true)
LocalStorageCapacityIsolationFSQuotaMonitoring=true|false (ALPHA - 默认值为 false)
MixedProtocolLBService=true|false (ALPHA - 默认值为 false)
NodeDisruptionExclusion=true|false (BETA - 默认值为 true)
NonPreemptingPriority=true|false (BETA - 默认值为 true)
PodDisruptionBudget=true|false (BETA - 默认值为 true)
PodOverhead=true|false (BETA - 默认值为 true)
ProcMountType=true|false (ALPHA - 默认值为 false)
QOSReserved=true|false (ALPHA - 默认值为 false)
RemainingItemCount=true|false (BETA - 默认值为 true)
RemoveSelfLink=true|false (BETA - 默认值为 true)
RootCAConfigMap=true|false (BETA - 默认值为 true)
RotateKubeletServerCertificate=true|false (BETA - 默认值为 true)
RunAsGroup=true|false (BETA - 默认值为 true)
ServerSideApply=true|false (BETA - 默认值为 true)
ServiceAccountIssuerDiscovery=true|false (BETA - 默认值为 true)
ServiceLBNodePortControl=true|false (ALPHA - 默认值为 false)
ServiceNodeExclusion=true|false (BETA - 默认值为 true)
ServiceTopology=true|false (ALPHA - 默认值为 false)
SetHostnameAsFQDN=true|false (BETA - 默认值为 true)
SizeMemoryBackedVolumes=true|false (ALPHA - 默认值为 false)
StorageVersionAPI=true|false (ALPHA - 默认值为 false)
StorageVersionHash=true|false (BETA - 默认值为 true)
Sysctls=true|false (BETA - 默认值为 true)
TTLAfterFinished=true|false (ALPHA - 默认值为 false)
TopologyManager=true|false (BETA - 默认值为 true)
ValidateProxyRedirects=true|false (BETA - 默认值为 true)
WarningHeaders=true|false (BETA - 默认值为 true)
WinDSR=true|false (ALPHA - 默认值为 false)
WinOverlay=true|false (BETA - 默认值为 true)
WindowsEndpointSliceProxying=true|false (ALPHA - 默认值为 false)
已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--file-check-frequency duration 默认值： `20s`

检查配置文件中新数据的时间间隔。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--hairpin-mode string 默认值： `promiscuous-bridge`

设置 kubelet 执行发夹模式（hairpin）网络地址转译的方式。 该模式允许后端端点对其自身服务的访问能够再次经由负载均衡转发回自身。 可选项包括“ `promiscuous-bridge` ”、“ `hairpin-veth` ”和“ `none` ”。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--healthz-bind-address ip 默认值： `127.0.0.1`

用于运行 healthz 服务器的 IP 地址（设置为 `0.0.0.0` 表示使用所有 IPv4 接口， 设置为 `::` 表示使用所有 IPv6 接口。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--healthz-port int32　默认值：10248

本地 healthz 端点使用的端口（设置为 0 表示禁用）。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

-h, --help

kubelet 操作的帮助命令

--hostname-override string

如果为非空，将使用此字符串而不是实际的主机名作为节点标识。如果设置了 `--cloud-provider`，则云驱动将确定节点的名称 （请查阅云服务商文档以确定是否以及如何使用主机名）。

--housekeeping-interval duration　默认值： `10s`

清理容器操作的时间间隔。

--http-check-frequency duration　默认值： `20s`

HTTP 服务以获取新数据的时间间隔。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--image-credential-provider-bin-dir string

指向凭据提供组件可执行文件所在目录的路径。

--image-credential-provider-config string

指向凭据提供插件配置文件所在目录的路径。

--image-gc-high-threshold int32　默认值：85

镜像垃圾回收上限。磁盘使用空间达到该百分比时，镜像垃圾回收将持续工作。 值必须在 [0，100] 范围内。要禁用镜像垃圾回收，请设置为 100。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--image-gc-low-threshold int32　默认值：80

镜像垃圾回收下限。磁盘使用空间在达到该百分比之前，镜像垃圾回收操作不会运行。 值必须在 [0，100] 范围内，并且不得大于 `--image-gc-high-threshold` 的值。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--image-pull-progress-deadline duration　默认值： `1m0s`

如果在该参数值所设置的期限之前没有拉取镜像的进展，镜像拉取操作将被取消。 仅当容器运行环境设置为 `docker` 时，此特定于 docker 的参数才有效。

--image-service-endpoint string

[实验性特性] 远程镜像服务的端点。若未设定则默认情况下使用 `--container-runtime-endpoint` 的值。目前支持的类型包括在 Linux 系统上的 UNIX 套接字端点和 Windows 系统上的 npipe 和 TCP 端点。 例如： `unix:///var/run/dockershim.sock` 、 `npipe:////./pipe/dockershim` 。

--iptables-drop-bit int32　默认值：15

标记数据包将被丢弃的 fwmark 位设置。必须在 [0，31] 范围内。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--iptables-masquerade-bit int32　默认值：14

标记数据包将进行 SNAT 的 fwmark 空间位设置。必须在 [0，31] 范围内。 请将此参数与 `kube-proxy` 中的相应参数匹配。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--keep-terminated-pod-volumes

设置为 true 表示 Pod 终止后仍然保留之前挂载过的卷，常用于调试与卷有关的问题。 已弃用：将未来版本中移除。

--kernel-memcg-notification

若启用，则 kubelet 将与内核中的 memcg 通知机制集成，不再使用轮询的方式来判定 是否 Pod 达到内存驱逐阈值。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--kube-api-burst int32　默认值：10

每秒发送到 apiserver 的突发请求数量上限。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--kube-api-content-type string　默认值： `application/vnd.kubernetes.protobuf`

发送到 apiserver 的请求的内容类型。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--kube-api-qps int32　默认值：5

与 apiserver 通信的每秒查询个数（QPS）。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--kube-reserved mapStringString　默认值：<None>

kubernetes 系统预留的资源配置，以一组 资源名称=资源数量 格式表示。 （例如： `cpu=200m,memory=500Mi,ephemeral-storage=1Gi,pid='100'` ）。 当前支持 `cpu` 、 `memory` 和用于根文件系统的 `ephemeral-storage` 。 请参阅相关文档获取更多信息。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--kube-reserved-cgroup string　默认值： `""`

给出某个顶层 cgroup 绝对名称，该 cgroup 用于管理通过标志 `--kube-reserved` 为 kubernetes 组件所预留的计算资源。例如： `"/kube-reserved"` 。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--kubeconfig string

kubeconfig 配置文件的路径，指定如何连接到 API 服务器。 提供 `--kubeconfig` 将启用 API 服务器模式，而省略 `--kubeconfig` 将启用独立模式。

--kubelet-cgroups string

用于创建和运行 kubelet 的 cgroup 的绝对名称。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--lock-file string

<警告：alpha 特性> kubelet 使用的锁文件的路径。

--log-backtrace-at traceLocation　默认值： `:0`

形式为 `<file>:<N>` 。 当日志逻辑执行到命中 <file> 的第 <N> 行时，转储调用堆栈。

--log-dir string

如果此值为非空，则在所指定的目录中写入日志文件。

--log-file string

如果此值非空，使用所给字符串作为日志文件名。

--log-file-max-size uint　默认值：1800

设置日志文件的最大值。单位为兆字节（M）。如果值为 0，则表示文件大小无限制。

--log-flush-frequency duration　默认值： `5s`

两次日志刷新之间的最大秒数（默认值为 5s）。

--logging-format string　默认值： `"text"`

设置日志文件格式。可以设置的格式有： `"text"` 、 `"json"` 。 非默认的格式不会使用以下标志的配置： `--add-dir-header` ， `--alsologtostderr` ， `--log-backtrace-at` ， `--log-dir` ， `--log-file` ， `--log-file-max-size` ， `--logtostderr` ， `--skip-headers` ， `--skip-log-headers` ， `--stderrthreshold` ， `--log-flush-frequency` 。 非默认选项的其它值都应视为 Alpha 特性，将来出现更改时不会额外警告。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--logtostderr　默认值： `true`

日志输出到 stderr 而不是文件。

--make-iptables-util-chains　默认值： `true`

设置为 true 表示 kubelet 将确保 `iptables` 规则在主机上存在。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--manifest-url string

用于访问要运行的其他 Pod 规范的 URL。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--manifest-url-header string

取值为由 HTTP 头部组成的逗号分隔列表，在访问 `--manifest-url` 所给出的 URL 时使用。 名称相同的多个头部将按所列的顺序添加。该参数可以多次使用。例如： `--manifest-url-header 'a:hello,b:again,c:world' --manifest-url-header 'b:beautiful'` 。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--master-service-namespace string　默认值： `default`

kubelet 向 Pod 注入 Kubernetes 主控服务信息时使用的命名空间。 已弃用：此标志将在未来的版本中删除。

--max-open-files int　默认值：1000000

kubelet 进程可以打开的最大文件数量。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--max-pods int32　默认值：110

此 kubelet 能运行的 Pod 最大数量。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--maximum-dead-containers int32　默认值：-1

设置全局可保留的已停止容器实例个数上限。 每个实例会占用一些磁盘空间。要禁用，请设置为负数。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--maximum-dead-containers-per-container int32　默认值：1

每个已停止容器可以保留的的最大实例数量。每个容器占用一些磁盘空间。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--minimum-container-ttl-duration duration

已结束的容器在被垃圾回收清理之前的最少存活时间。 例如： `300ms` 、 `10s` 或者 `2h45m` 。 已弃用：请改用 `--eviction-hard` 或者 `--eviction-soft` 。 此标志将在未来的版本中删除。

--minimum-image-ttl-duration duration　默认值： `2m0s`

不再使用的镜像在被垃圾回收清理之前的最少存活时间。 例如： `300ms` 、 `10s` 或者 `2h45m` 。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--network-plugin string

<警告：alpha 特性> 设置 kubelet/Pod 生命周期中各种事件调用的网络插件的名称。 仅当容器运行环境设置为 `docker` 时，此特定于 docker 的参数才有效。

--network-plugin-mtu int32

<警告：alpha 特性> 传递给网络插件的 MTU 值，将覆盖默认值。 设置为 0 则使用默认的 MTU 1460。仅当容器运行环境设置为 `docker` 时， 此特定于 docker 的参数才有效。

--node-ip string

节点的 IP 地址。如果设置，kubelet 将使用该 IP 地址作为节点的 IP 地址。

--node-labels mapStringString

<警告：alpha 特性> kubelet 在集群中注册本节点时设置的标签。标签以 `key=value` 的格式表示，多个标签以逗号分隔。名字空间 `kubernetes.io` 中的标签必须以 `kubelet.kubernetes.io` 或 `node.kubernetes.io` 为前缀， 或者在以下明确允许范围内： `beta.kubernetes.io/arch` , `beta.kubernetes.io/instance-type` , `beta.kubernetes.io/os` , `failure-domain.beta.kubernetes.io/region` , `failure-domain.beta.kubernetes.io/zone` , `kubernetes.io/arch` , `kubernetes.io/hostname` , `kubernetes.io/os` , `node.kubernetes.io/instance-type` , `topology.kubernetes.io/region` , `topology.kubernetes.io/zone` 。

--node-status-max-images int32　默认值：50

在 `node.status.images` 中可以报告的最大镜像数量。如果指定为 -1，则不设上限。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--node-status-update-frequency duration　默认值： `10s`

指定 kubelet 向主控节点汇报节点状态的时间间隔。注意：更改此常量时请务必谨慎， 它必须与节点控制器中的 `nodeMonitorGracePeriod` 一起使用。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--non-masquerade-cidr string　默认值： `10.0.0.0/8`

kubelet 向该 IP 段之外的 IP 地址发送的流量将使用 IP 伪装技术。 设置为 `0.0.0.0/0` 则不使用伪装。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--one-output

如果设置此标志为 `true`，则仅将日志写入其原来的严重性级别中，而不是同时将其写入更低严重性级别中。

--oom-score-adj int32　默认值：-999

kubelet 进程的 oom-score-adj 参数值。有效范围为 `[-1000，1000]`。已弃用：应在 `--config` 所给的配置文件中进行设置。（进一步了解）

--pod-cidr string

用于给 Pod 分配 IP 地址的 CIDR 地址池，仅在独立运行模式下使用。在集群模式下，CIDR 设置是从主服务器获取的。对于 IPv6，分配的 IP 的最大数量为 65536。已弃用：应在 `--config` 所给的配置文件中进行设置。（进一步了解）

--pod-infra-container-image string　默认值：`k8s.gcr.io/pause:3.2`

所指定的镜像不会被镜像垃圾收集器删除。当容器运行环境设置为 `docker` 时，各个 Pod 中的所有容器都会 使用此镜像中的网络和 IPC 名字空间。其他 CRI 实现有自己的配置来设置此镜像。

--pod-manifest-path string

设置包含要运行的静态 Pod 的文件的路径，或单个静态 Pod 文件的路径。以点（`.`）开头的文件将被忽略。已弃用：应在 `--config` 所给的配置文件中进行设置。（进一步了解）

--pod-max-pids int　默认值：-1

设置每个 Pod 中的最大进程数目。如果为 -1，则 kubelet 使用节点可分配的 PID 容量作为默认值。已弃用：应在 `--config` 所给的配置文件中进行设置。（进一步了解）

--pods-per-core int32

kubelet 在每个处理器核上可运行的 Pod 数量。此 kubelet 上的 Pod 总数不能超过 `--max-pods` 标志值。因此，如果此计算结果导致在 kubelet 上允许更多数量的 Pod，则使用 `--max-pods` 值。值为 0 表示不作限制。已弃用：应在 `--config` 所给的配置文件中进行设置。（进一步了解）

--port int32　默认值：10250

kubelet 服务监听的本机端口号。已弃用：应在 `--config` 所给的配置文件中进行设置。（进一步了解）

--protect-kernel-defaults

设置 kubelet 的默认内核调整行为。如果已设置该参数，当任何内核可调参数与 kubelet 默认值不同时，kubelet 都会出错。已弃用：应在 `--config` 所给的配置文件中进行设置。（进一步了解）

--provider-id string

设置主机数据库（即，云驱动）中用来标识节点的唯一标识。已弃用：应在 `--config` 所给的配置文件中进行设置。（进一步了解）

--qos-reserved mapStringString

<警告：alpha 特性> 设置在指定的 QoS 级别预留的 Pod 资源请求，以一组 `"资源名称=百分比"` 的形式进行设置，例如 `memory=50%`。当前仅支持内存（memory）。要求启用 `QOSReserved` 特性门控。已弃用：应在 `--config` 所给的配置文件中进行设置。（进一步了解）

--read-only-port int32　默认值：10255

kubelet 可以在没有身份验证/鉴权的情况下提供只读服务的端口（设置为 0 表示禁用）。已弃用：应在 `--config` 所给的配置文件中进行设置。（进一步了解）

--really-crash-for-testing

设置为 true 表示发生失效时立即崩溃。仅用于测试。 已弃用：将在未来版本中移除。

--redirect-container-streaming

启用容器流数据重定向。如果设置为 false，则 kubelet 将在 apiserver 和容器运行时 之间转发容器流数据；如果设置为 true，则 kubelet 将返回指向 apiserver 的 HTTP 重定向信息，而 apiserver 将直接访问容器运行时。代理方法更安全，但会带来一些开销。 重定向方法性能更高，但安全性较低，因为 apiserver 和容器运行时之间的连接可能未通过身份验证。
已弃用：容器流数据重定向会在 v1.20 中从 kubelet 中移除，此标志会在 v1.22 中移除。 相关信息可参见改进说明。

--register-node　默认值： `true`

将本节点注册到 API 服务器。如果未提供 `--kubeconfig` 标志设置， 则此参数无关紧要，因为 kubelet 将没有要注册的 API 服务器。

--register-schedulable　默认值：true

注册本节点为可调度的节点。当 `--register-node` 标志为 false 时此设置无效。 已弃用：此参数将在未来的版本中删除。

--register-with-taints mapStringString

设置本节点的污点标记，格式为 `<key>=<value>:<effect>` ， 以逗号分隔。当 `--register-node` 为 false 时此标志无效。 已弃用：将在未来版本中移除。

--registry-burst int32　默认值：10

设置突发性镜像拉取的个数上限，在不超过 `--registration-qps` 设置值的前提下 暂时允许此参数所给的镜像拉取个数。仅在 `--registry-qps` 大于 0 时使用。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--registry-qps int32　Default: 5

如此值大于 0，可用来限制镜像仓库的 QPS 上限。设置为 0，表示不受限制。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--reserved-cpus string

用逗号分隔的一组 CPU 或 CPU 范围列表，给出为系统和 Kubernetes 保留使用的 CPU。 此列表所给出的设置优先于通过 `--system-reserved` 和 `--kube-reskube-reserved` 所保留的 CPU 个数配置。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--resolv-conf string　默认值： `/etc/resolv.conf`

名字解析服务的配置文件名，用作容器 DNS 解析配置的基础。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--root-dir string　默认值： `/var/lib/kubelet`

设置用于管理 kubelet 文件的根目录（例如挂载卷的相关文件等）。

--rotate-certificates

<警告：Beta 特性> 设置当客户端证书即将过期时 kubelet 自动从 `kube-apiserver` 请求新的证书进行轮换。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--rotate-server-certificates

当 kubelet 的服务证书即将过期时自动从 kube-apiserver 请求新的证书进行轮换。 要求启用 `RotateKubeletServerCertificate` 特性门控，以及对提交的 `CertificateSigningRequest` 对象进行批复（Approve）操作。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--runonce

设置为 true 表示从本地清单或远程 URL 创建完 Pod 后立即退出 kubelet 进程。 与 `--enable-server` 标志互斥。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--runtime-cgroups string

设置用于创建和运行容器运行时的 cgroup 的绝对名称。

--runtime-request-timeout duration　默认值： `2m0s`

设置除了长时间运行的请求（包括 `pull` 、 `logs` 、 `exec` 和 `attach` 等操作）之外的其他运行时请求的超时时间。 到达超时时间时，请求会被取消，抛出一个错误并会等待重试。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--seccomp-profile-root string　默认值： `/var/lib/kubelet/seccomp`

<警告：alpha 特性> seccomp 配置文件目录。 已弃用：将在 1.23 版本中移除，以使用 `<root-dir>/seccomp` 目录。

--serialize-image-pulls　默认值： `true`

逐一拉取镜像。建议 *不要* 在 docker 守护进程版本低于 1.9 或启用了 Aufs 存储后端的节点上 更改默认值。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--skip-headers

设置为 true 时在日志消息中去掉标头前缀。

--skip-log-headers

设置为 true，打开日志文件时去掉标头。

--stderrthreshold int　默认值：2

设置严重程度达到或超过此阈值的日志输出到标准错误输出。

--streaming-connection-idle-timeout duration　默认值： `4h0m0s`

设置流连接在自动关闭之前可以空闲的最长时间。0 表示没有超时限制。 例如： `5m` 。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--sync-frequency duration　默认值： `1m0s`

在运行中的容器与其配置之间执行同步操作的最长时间间隔。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--system-cgroups /

此标志值为一个 cgroup 的绝对名称，用于所有尚未放置在根目录下某 cgroup 内的非内核进程。 空值表示不指定 cgroup。回滚该参数需要重启机器。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--system-reserved mapStringString　默认值：无

系统预留的资源配置，以一组 资源名称=资源数量 的格式表示， （例如：`cpu=200m,memory=500Mi,ephemeral-storage=1Gi,pid='100'`）。 目前仅支持 `cpu` 和 `memory` 的设置。 更多细节可参考 相关文档。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--system-reserved-cgroup string　默认值： `""`

此标志给出一个顶层 cgroup 绝对名称，该 cgroup 用于管理非 kubernetes 组件， 这些组件的计算资源通过 `--system-reserved` 标志进行预留。 例如 `"/system-reserved"`。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--tls-cert-file string

包含 x509 证书的文件路径，用于 HTTPS 认证。 如果有中间证书，则中间证书要串接在在服务器证书之后。 如果未提供 `--tls-cert-file` 和 `--tls-private-key-file`， kubelet 会为公开地址生成自签名证书和密钥，并将其保存到通过 `--cert-dir` 指定的目录中。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--tls-cipher-suites string

服务器端加密算法列表，以逗号分隔。如果不设置，则使用 Go 语言加密包的默认算法列表。
可选加密算法包括：
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305,TLS_ECDHE_ECDSA_WITH_RC4_128_SHA,TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305,TLS_ECDHE_RSA_WITH_RC4_128_SHA,TLS_RSA_WITH_3DES_EDE_CBC_SHA,TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_128_CBC_SHA256,TLS_RSA_WITH_AES_128_GCM_SHA256,TLS_RSA_WITH_AES_256_CBC_SHA,TLS_RSA_WITH_AES_256_GCM_SHA384,TLS_RSA_WITH_RC4_128_SHA
已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--tls-min-version string

设置支持的最小 TLS 版本号，可选的版本号包括： `VersionTLS10` 、 `VersionTLS11` 、 `VersionTLS12` 和 `VersionTLS13` 。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--tls-private-key-file string

包含与 `--tls-cert-file` 对应的 x509 私钥文件路径。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--topology-manager-policy string　默认值： `none`

设置拓扑管理策略（Topology Manager policy）。可选值包括： `none` 、 `best-effort` 、 `restricted` 和 `single-numa-node` 。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

--topology-manager-scope string　默认值： `container`

拓扑提示信息使用范围。拓扑管理器从提示提供者（Hints Providers）处收集提示信息， 并将其应用到所定义的范围以确保 Pod 准入。 可选值包括： `container` （默认）、 `pod` 。 已弃用：应在 `--config` 所给的配置文件中进行设置。 （进一步了解）

-v, --v Level

设置 kubelet 日志级别详细程度的数值。

--version version[=true]

打印 kubelet 版本信息并退出。

--vmodule moduleSpec

以逗号分隔的 `pattern=N` 设置列表，用于文件过滤的日志记录

--volume-plugin-dir string　默认值：`/usr/libexec/kubernetes/kubelet-plugins/volume/exec/`

用来搜索第三方存储卷插件的目录。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

--volume-stats-agg-period duration　默认值：`1m0s`

指定 kubelet 计算和缓存所有 Pod 和卷的磁盘用量总值的时间间隔。要禁用磁盘用量计算，请设置为 0。 已弃用：应在 `--config` 所给的配置文件中进行设置。 (进一步了解)

3 - kube-apiserver

简介

Kubernetes API 服务器验证并配置 API 对象的数据， 这些对象包括 pods、services、replicationcontrollers 等。 API 服务器为 REST 操作提供服务，并为集群的共享状态提供前端， 所有其他组件都通过该前端进行交互。

```
kube-apiserver [flags]
```

选项

--add-dir-header

如果为 true，则将文件目录添加到日志消息的标题中

--admission-control-config-file string

包含准入控制配置的文件。

--advertise-address string

向集群成员通知 apiserver 消息的 IP 地址。 这个地址必须能够被集群中其他成员访问。 如果 IP 地址为空，将会使用 --bind-address， 如果未指定 --bind-address，将会使用主机的默认接口地址。

--allow-metric-labels stringToString　默认值：[]

允许使用的指标标签到指标值的映射列表。键的格式为 <MetricName>,<LabelName>. 值的格式为 <allowed_value>,<allowed_value>...。 例如： `metric1,label1='v1,v2,v3', metric1,label2='v1,v2,v3' metric2,label1='v1,v2,v3'` 。

--allow-privileged

如果为 true, 将允许特权容器。[默认值=false]

--alsologtostderr

在向文件输出日志的同时，也将日志写到标准输出。

--anonymous-auth　默认值：true

启用到 API 服务器的安全端口的匿名请求。 未被其他认证方法拒绝的请求被当做匿名请求。匿名请求的用户名为 `system:anonymous` ， 用户组名为 system:unauthenticated。

--api-audiences strings

API 的标识符。 服务帐户令牌验证者将验证针对 API 使用的令牌是否已绑定到这些受众中的至少一个。 如果配置了 `--service-account-issuer` 标志，但未配置此标志， 则此字段默认为包含发布者 URL 的单个元素列表。

--apiserver-count int　默认值：1

集群中运行的 API 服务器数量，必须为正数。 （在启用 --endpoint-reconciler-type=master-count 时使用。）

--audit-log-batch-buffer-size int　默认值：10000

批处理和写入之前用于存储事件的缓冲区大小。 仅在批处理模式下使用。

--audit-log-batch-max-size int　默认值：1

每个批次的最大大小。仅在批处理模式下使用。

--audit-log-batch-max-wait duration

强制写入尚未达到最大大小的批次之前要等待的时间。 仅在批处理模式下使用。

--audit-log-batch-throttle-burst int

如果之前未使用 ThrottleQPS，则为同时发送的最大请求数。 仅在批处理模式下使用。

--audit-log-batch-throttle-enable

是否启用了批量限制。仅在批处理模式下使用。

--audit-log-batch-throttle-qps float

每秒的最大平均批次数。仅在批处理模式下使用。

--audit-log-compress

若设置了此标志，则被轮换的日志文件会使用 gzip 压缩。

--audit-log-format string　默认值："json"

所保存的审计格式。 "legacy" 表示每行一个事件的文本格式。"json" 表示结构化的 JSON 格式。 已知格式为 legacy，json。

--audit-log-maxage int

根据文件名中编码的时间戳保留旧审计日志文件的最大天数。

--audit-log-maxbackup int

要保留的旧的审计日志文件个数上限。

--audit-log-maxsize int

轮换之前，审计日志文件的最大大小（以兆字节为单位）。

--audit-log-mode string　默认值："blocking"

用来发送审计事件的策略。 阻塞（blocking）表示发送事件应阻止服务器响应。 批处理（batch）会导致后端异步缓冲和写入事件。 已知的模式是批处理（batch），阻塞（blocking），严格阻塞（blocking-strict）。

--audit-log-path string

如果设置，则所有到达 API 服务器的请求都将记录到该文件中。 "-" 表示标准输出。

--audit-log-truncate-enabled

是否启用事件和批次截断。

--audit-log-truncate-max-batch-size int　默认值：10485760

发送到下层后端的每批次的最大数据量。 实际的序列化大小可能会增加数百个字节。 如果一个批次超出此限制，则将其分成几个较小的批次。

--audit-log-truncate-max-event-size int　默认值：102400

发送到下层后端的每批次的最大数据量。 如果事件的大小大于此数字，则将删除第一个请求和响应； 如果这样做没有减小足够大的程度，则将丢弃事件。

--audit-log-version string　默认值："audit.k8s.io/v1"

用于对写入日志的审计事件执行序列化的 API 组和版本。

--audit-policy-file string

定义审计策略配置的文件的路径。

--audit-webhook-batch-buffer-size int　默认值：10000

划分批次和写入之前用于存储事件的缓冲区大小。 仅在批处理模式下使用。

--audit-webhook-batch-max-size int　默认值：400

批次的最大大小。 仅在批处理模式下使用。

--audit-webhook-batch-max-wait duration　默认值：30s

强制写入尚未达到最大大小的批处理之前要等待的时间。 仅在批处理模式下使用。

--audit-webhook-batch-throttle-burst int　默认值：15

如果之前未使用 ThrottleQPS，同时发送的最大请求数。 仅在批处理模式下使用。

--audit-webhook-batch-throttle-enable　默认值：true

是否启用了批量限制。仅在批处理模式下使用。

--audit-webhook-batch-throttle-qps float32　默认值：10

每秒的最大平均批次数。仅在批处理模式下使用。

--audit-webhook-config-file string

定义审计 webhook 配置的 kubeconfig 格式文件的路径。

--audit-webhook-initial-backoff duration　默认值：10s

重试第一个失败的请求之前要等待的时间。

--audit-webhook-mode string　默认值："batch"

发送审计事件的策略。 阻止（Blocking）表示发送事件应阻止服务器响应。 批处理（Batch）导致后端异步缓冲和写入事件。 已知的模式是批处理（batch），阻塞（blocking），严格阻塞（blocking-strict）。

--audit-webhook-truncate-enabled

是否启用事件和批处理截断。

--audit-webhook-truncate-max-batch-size int　默认值：10485760

发送到下层后端的批次的最大数据量。 实际的序列化大小可能会增加数百个字节。 如果一个批次超出此限制，则将其分成几个较小的批次。

--audit-webhook-truncate-max-event-size int　默认值：102400

发送到下层后端的批次的最大数据量。 如果事件的大小大于此数字，则将删除第一个请求和响应； 如果事件和事件的大小没有减小到一定幅度，则将丢弃事件。

--audit-webhook-version string　默认值："audit.k8s.io/v1"

用于序列化写入 Webhook 的审计事件的 API 组和版本。

--authentication-token-webhook-cache-ttl duration　2m0s

对来自 Webhook 令牌身份验证器的响应的缓存时间。

--authentication-token-webhook-config-file string

包含 Webhook 配置的 kubeconfig 格式文件，用于进行令牌认证。 API 服务器将查询远程服务，以对持有者令牌进行身份验证。

--authentication-token-webhook-version string　默认值："v1beta1"

与 Webhook 之间交换 authentication.k8s.io TokenReview 时使用的 API 版本。

--authorization-mode stringSlice　默认值："AlwaysAllow"

在安全端口上进行鉴权的插件的顺序列表。 逗号分隔的列表：AlwaysAllow、AlwaysDeny、ABAC、Webhook、RBAC、Node。

--authorization-policy-file string

包含鉴权策略的文件，其内容为分行 JSON 格式， 在安全端口上与 --authorization-mode=ABAC 一起使用。

--authorization-webhook-cache-authorized-ttl duration　默认值：5m0s

对来自 Webhook 鉴权组件的 “授权（authorized）” 响应的缓存时间。

--authorization-webhook-cache-unauthorized-ttl duration　默认值：30s

对来自 Webhook 鉴权模块的 “未授权（unauthorized）” 响应的缓存时间。

--authorization-webhook-config-file string

包含 Webhook 配置的文件，其格式为 kubeconfig， 与 --authorization-mode=Webhook 一起使用。 API 服务器将查询远程服务，以对 API 服务器的安全端口的访问执行鉴权。

--authorization-webhook-version string　默认值："v1beta1"

与 Webhook 之间交换 authorization.k8s.io SubjectAccessReview 时使用的 API 版本。

--azure-container-registry-config string

包含 Azure 容器仓库配置信息的文件的路径。

--bind-address string　默认值："0.0.0.0"

用来监听 `--secure-port` 端口的 IP 地址。 集群的其余部分以及 CLI/web 客户端必须可以访问所关联的接口。 如果为空白或未指定地址（0.0.0.0 或 ::），则将使用所有接口。

--cert-dir string　默认值："/var/run/kubernetes"

TLS 证书所在的目录。 如果提供了 `--tls-cert-file` 和 `--tls-private-key-file` 标志值，则将忽略此标志。

--client-ca-file string

如果已设置，则使用与客户端证书的 CommonName 对应的标识对任何出示由 client-ca 文件中的授权机构之一签名的客户端证书的请求进行身份验证。

--cloud-config string

云厂商配置文件的路径。空字符串表示无配置文件。

--cloud-provider string

云服务提供商。空字符串表示没有云厂商。

--cloud-provider-gce-l7lb-src-cidrs cidrs　默认值："130.211.0.0/22,35.191.0.0/16"

在 GCE 防火墙中打开 CIDR，以进行第 7 层负载均衡流量代理和健康状况检查。

--contention-profiling

如果启用了性能分析，则启用锁争用性能分析。

--cors-allowed-origins strings

CORS 允许的来源清单，以逗号分隔。 允许的来源可以是支持子域匹配的正则表达式。 如果此列表为空，则不会启用 CORS。

--default-not-ready-toleration-seconds int　默认值：300

对污点 NotReady:NoExecute 的容忍时长（以秒计）。 默认情况下这一容忍度会被添加到尚未具有此容忍度的每个 pod 中。

--default-unreachable-toleration-seconds int　默认值：300

对污点 Unreachable:NoExecute 的容忍时长（以秒计） 默认情况下这一容忍度会被添加到尚未具有此容忍度的每个 pod 中。

--default-watch-cache-size int　默认值：100

默认监听（watch）缓存大小。 如果为零，则将为没有设置默认监视大小的资源禁用监视缓存。

--delete-collection-workers int　默认值： 1

为 DeleteCollection 调用而产生的工作线程数。 这些用于加速名字空间清理。

--disable-admission-plugins strings

尽管位于默认启用的插件列表中（NamespaceLifecycle、LimitRanger、ServiceAccount、TaintNodesByCondition、PodSecurity、Priority、DefaultTolerationSeconds、DefaultStorageClass、StorageObjectInUseProtection、PersistentVolumeClaimResize、RuntimeClass、CertificateApproval、CertificateSigning、CertificateSubjectRestriction、DefaultIngressClass、MutatingAdmissionWebhook、ValidatingAdmissionWebhook、ResourceQuota）仍须被禁用的插件。
取值为逗号分隔的准入插件列表：AlwaysAdmit、AlwaysDeny、AlwaysPullImages、CertificateApproval、CertificateSigning、CertificateSubjectRestriction、DefaultIngressClass、DefaultStorageClass、DefaultTolerationSeconds、DenyServiceExternalIPs、EventRateLimit、ExtendedResourceToleration、ImagePolicyWebhook、LimitPodHardAntiAffinityTopology、LimitRanger、MutatingAdmissionWebhook、NamespaceAutoProvision、NamespaceExists、NamespaceLifecycle、NodeRestriction、OwnerReferencesPermissionEnforcement、PersistentVolumeClaimResize、PersistentVolumeLabel、PodNodeSelector、PodSecurity、PodSecurityPolicy、PodTolerationRestriction、Priority、ResourceQuota、RuntimeClass、SecurityContextDeny、ServiceAccount、StorageObjectInUseProtection、TaintNodesByCondition、ValidatingAdmissionWebhook。
该标志中插件的顺序无关紧要。

--disabled-metrics strings

此标志为行为不正确的度量指标提供一种处理方案。 你必须提供完全限定的指标名称才能将其禁止。 声明：禁用度量值的行为优先于显示已隐藏的度量值。

--egress-selector-config-file string

带有 API 服务器出站选择器配置的文件。

--enable-admission-plugins stringSlice

除了默认启用的插件（NamespaceLifecycle、LimitRanger、ServiceAccount、TaintNodesByCondition、PodSecurity、Priority、DefaultTolerationSeconds、DefaultStorageClass、StorageObjectInUseProtection、PersistentVolumeClaimResize、RuntimeClass、CertificateApproval、CertificateSigning、CertificateSubjectRestriction、DefaultIngressClass、MutatingAdmissionWebhook、ValidatingAdmissionWebhook、ResourceQuota）之外要启用的插件
取值为逗号分隔的准入插件列表：AlwaysAdmit、AlwaysDeny、AlwaysPullImages、CertificateApproval、CertificateSigning、CertificateSubjectRestriction、DefaultIngressClass、DefaultStorageClass、DefaultTolerationSeconds、DenyServiceExternalIPs、EventRateLimit、ExtendedResourceToleration、ImagePolicyWebhook、LimitPodHardAntiAffinityTopology、LimitRanger、MutatingAdmissionWebhook、NamespaceAutoProvision、NamespaceExists、NamespaceLifecycle、NodeRestriction、OwnerReferencesPermissionEnforcement、PersistentVolumeClaimResize、PersistentVolumeLabel、PodNodeSelector、PodSecurity、PodSecurityPolicy、PodTolerationRestriction、Priority、ResourceQuota、RuntimeClass、SecurityContextDeny、ServiceAccount、StorageObjectInUseProtection、TaintNodesByCondition、ValidatingAdmissionWebhook
该标志中插件的顺序无关紧要。

--enable-aggregator-routing

允许聚合器将请求路由到端点 IP 而非集群 IP。

--enable-bootstrap-token-auth

启用以允许将 "kube-system" 名字空间中类型为 "bootstrap.kubernetes.io/token" 的 Secret 用于 TLS 引导身份验证。

--enable-garbage-collector　默认值：true

启用通用垃圾收集器。必须与 kube-controller-manager 的相应标志同步。

--enable-priority-and-fairness　默认值：true

如果为 true 且启用了 `APIPriorityAndFairness` 特性门控， 请使用增强的处理程序替换 max-in-flight 处理程序， 以便根据优先级和公平性完成排队和调度。

--encryption-provider-config string

包含加密提供程序配置信息的文件，用在 etcd 中所存储的 Secret 上。

--endpoint-reconciler-type string　默认值："lease"

使用端点协调器（ `master-count` 、 `lease` 或 `none` ）。

--etcd-cafile string

用于保护 etcd 通信的 SSL 证书颁发机构文件。

--etcd-certfile string

用于保护 etcd 通信的 SSL 证书文件。

--etcd-compaction-interval duration　默认值：5m0s

压缩请求的间隔。 如果为0，则禁用来自 API 服务器的压缩请求。

--etcd-count-metric-poll-period duration　默认值：1m0s

针对每种类型的资源数量轮询 etcd 的频率。 0 值表示禁用度量值收集。

--etcd-db-metric-poll-interval duration　默认值：30s

轮询 etcd 和更新度量值的请求间隔。0 值表示禁用度量值收集。

--etcd-healthcheck-timeout duration　检查 etcd 健康状况时使用的超时时长。

--etcd-keyfile string

用于保护 etcd 通信的 SSL 密钥文件。

--etcd-prefix string　默认值："/registry"

要在 etcd 中所有资源路径之前添加的前缀。

--etcd-servers strings

要连接的 etcd 服务器列表（ `scheme://ip:port` ），以逗号分隔。

--etcd-servers-overrides strings

etcd 服务器针对每个资源的重载设置，以逗号分隔。 单个替代格式：组/资源#服务器（group/resource#servers）， 其中服务器是 URL，以分号分隔。

--event-ttl duration　默认值：1h0m0s

事件的保留时长。

--experimental-logging-sanitization

[试验性功能] 启用此标志时，被标记为敏感的字段（密码、密钥、令牌）都不会被日志输出。
运行时的日志清理可能会引入相当程度的计算开销，因此不应该在产品环境中启用。

--external-hostname string

为此主机生成外部化 UR L时要使用的主机名（例如 Swagger API 文档或 OpenID 发现）。

--feature-gates <逗号分隔的 'key=True|False' 键值对>

一组 key=value 对，用来描述测试性/试验性功能的特性门控。可选项有：
APIListChunking=true|false (BETA - 默认值=true)
APIPriorityAndFairness=true|false (BETA - 默认值=true)
APIResponseCompression=true|false (BETA - 默认值=true)
APIServerIdentity=true|false (ALPHA - 默认值=false)
APIServerTracing=true|false (ALPHA - 默认值=false)
AllAlpha=true|false (ALPHA - 默认值=false)
AllBeta=true|false (BETA - 默认值=false)
AnyVolumeDataSource=true|false (ALPHA - 默认值=false)
AppArmor=true|false (BETA - 默认值=true)
CPUManager=true|false (BETA - 默认值=true)
CPUManagerPolicyOptions=true|false (ALPHA - 默认值=false)
CSIInlineVolume=true|false (BETA - 默认值=true)
CSIMigration=true|false (BETA - 默认值=true)
CSIMigrationAWS=true|false (BETA - 默认值=false)
CSIMigrationAzureDisk=true|false (BETA - 默认值=false)
CSIMigrationAzureFile=true|false (BETA - 默认值=false)
CSIMigrationGCE=true|false (BETA - 默认值=false)

CSIMigrationOpenStack=true|false (BETA - 默认值=true)
CSIMigrationvSphere=true|false (BETA - 默认值=false)
CSIStorageCapacity=true|false (BETA - 默认值=true)
CSIVolumeFSGroupPolicy=true|false (BETA - 默认值=true)
CSIVolumeHealth=true|false (ALPHA - 默认值=false)
CSRDuration=true|false (BETA - 默认值=true)
ConfigurableFSGroupPolicy=true|false (BETA - 默认值=true)
ControllerManagerLeaderMigration=true|false (BETA - 默认值=true)
CustomCPUCFSQuotaPeriod=true|false (ALPHA - 默认值=false)
DaemonSetUpdateSurge=true|false (BETA - 默认值=true)
默认值PodTopologySpread=true|false (BETA - 默认值=true)
DelegateFSGroupToCSIDriver=true|false (ALPHA - 默认值=false)
DevicePlugins=true|false (BETA - 默认值=true)
DisableAcceleratorUsageMetrics=true|false (BETA - 默认值=true)
DisableCloudProviders=true|false (ALPHA - 默认值=false)
DownwardAPIHugePages=true|false (BETA - 默认值=false)
EfficientWatchResumption=true|false (BETA - 默认值=true)
EndpointSliceTerminatingCondition=true|false (BETA - 默认值=true)
EphemeralContainers=true|false (ALPHA - 默认值=false)
ExpandCSIVolumes=true|false (BETA - 默认值=true)
ExpandInUsePersistentVolumes=true|false (BETA - 默认值=true)
ExpandPersistentVolumes=true|false (BETA - 默认值=true)
ExpandedDNSConfig=true|false (ALPHA - 默认值=false)
ExperimentalHostUserNamespace默认值ing=true|false (BETA - 默认值=false)
GenericEphemeralVolume=true|false (BETA - 默认值=true)
GracefulNodeShutdown=true|false (BETA - 默认值=true)
HPAContainerMetrics=true|false (ALPHA - 默认值=false)
HPAScaleToZero=true|false (ALPHA - 默认值=false)
IPv6DualStack=true|false (BETA - 默认值=true)
InTreePluginAWSUnregister=true|false (ALPHA - 默认值=false)
InTreePluginAzureDiskUnregister=true|false (ALPHA - 默认值=false)
InTreePluginAzureFileUnregister=true|false (ALPHA - 默认值=false)
InTreePluginGCEUnregister=true|false (ALPHA - 默认值=false)
InTreePluginOpenStackUnregister=true|false (ALPHA - 默认值=false)
InTreePluginvSphereUnregister=true|false (ALPHA - 默认值=false)
IndexedJob=true|false (BETA - 默认值=true)
IngressClassNamespacedParams=true|false (BETA - 默认值=true)
JobTrackingWithFinalizers=true|false (ALPHA - 默认值=false)
KubeletCredentialProviders=true|false (ALPHA - 默认值=false)
KubeletInUserNamespace=true|false (ALPHA - 默认值=false)
KubeletPodResources=true|false (BETA - 默认值=true)
KubeletPodResourcesGetAllocatable=true|false (ALPHA - 默认值=false)
LocalStorageCapacityIsolation=true|false (BETA - 默认值=true)
LocalStorageCapacityIsolationFSQuotaMonitoring=true|false (ALPHA - 默认值=false)
LogarithmicScaleDown=true|false (BETA - 默认值=true)
MemoryManager=true|false (BETA - 默认值=true)
MemoryQoS=true|false (ALPHA - 默认值=false)
MixedProtocolLBService=true|false (ALPHA - 默认值=false)
NetworkPolicyEndPort=true|false (BETA - 默认值=true)
NodeSwap=true|false (ALPHA - 默认值=false)
NonPreemptingPriority=true|false (BETA - 默认值=true)
PodAffinityNamespaceSelector=true|false (BETA - 默认值=true)
PodDeletionCost=true|false (BETA - 默认值=true)
PodOverhead=true|false (BETA - 默认值=true)
PodSecurity=true|false (ALPHA - 默认值=false)
PreferNominatedNode=true|false (BETA - 默认值=true)
ProbeTerminationGracePeriod=true|false (BETA - 默认值=false)
ProcMountType=true|false (ALPHA - 默认值=false)
ProxyTerminatingEndpoints=true|false (ALPHA - 默认值=false)
QOSReserved=true|false (ALPHA - 默认值=false)
ReadWriteOncePod=true|false (ALPHA - 默认值=false)
RemainingItemCount=true|false (BETA - 默认值=true)
RemoveSelfLink=true|false (BETA - 默认值=true)
RotateKubeletServerCertificate=true|false (BETA - 默认值=true)
Seccomp默认值=true|false (ALPHA - 默认值=false)
ServiceInternalTrafficPolicy=true|false (BETA - 默认值=true)
ServiceLBNodePortControl=true|false (BETA - 默认值=true)
ServiceLoadBalancerClass=true|false (BETA - 默认值=true)
SizeMemoryBackedVolumes=true|false (BETA - 默认值=true)
StatefulSetMinReadySeconds=true|false (ALPHA - 默认值=false)
StorageVersionAPI=true|false (ALPHA - 默认值=false)
StorageVersionHash=true|false (BETA - 默认值=true)

SuspendJob=true|false (BETA - 默认值=true)
TTLAfterFinished=true|false (BETA - 默认值=true)
TopologyAwareHints=true|false (ALPHA - 默认值=false)
TopologyManager=true|false (BETA - 默认值=true)
VolumeCapacityPriority=true|false (ALPHA - 默认值=false)
WinDSR=true|false (ALPHA - 默认值=false)
WinOverlay=true|false (BETA - 默认值=true)
WindowsHostProcessContainers=true|false (ALPHA - 默认值=false)

--goaway-chance float

为防止 HTTP/2 客户端卡在单个 API 服务器上，可启用随机关闭连接（GOAWAY）。 客户端的其他运行中请求将不会受到影响，并且客户端将重新连接， 可能会在再次通过负载平衡器后登陆到其他 API 服务器上。 此参数设置将发送 GOAWAY 的请求的比例。 具有单个 API 服务器或不使用负载平衡器的群集不应启用此功能。 最小值为0（关闭），最大值为 .02（1/50 请求）； 建议使用 .001（1/1000）。

-h, --help

kube-apiserver 的帮助命令

--http2-max-streams-per-connection int

服务器为客户端提供的 HTTP/2 连接中最大流数的限制。 零表示使用 GoLang 的默认值。

--identity-lease-duration-seconds int 默认值：3600

kube-apiserver 租约时长（按秒计），必须是正数。 （当 APIServerIdentity 特性门控被启用时使用此标志值）

--identity-lease-renew-interval-seconds int 默认值：10

kube-apiserver 对其租约进行续期的时间间隔（按秒计），必须是正数。 （当 APIServerIdentity 特性门控被启用时使用此标志值）

--kubelet-certificate-authority string

证书颁发机构的证书文件的路径。

--kubelet-client-certificate string

TLS 的客户端证书文件的路径。

--kubelet-client-key string

TLS 客户端密钥文件的路径。

--kubelet-preferred-address-types strings 默认值：Hostname,InternalDNS,InternalIP,ExternalDNS,ExternalIP

用于 kubelet 连接的首选 NodeAddressTypes 列表。

--kubelet-timeout duration 默认值：5s

kubelet 操作超时时间。

--kubernetes-service-node-port int

如果非零，那么 Kubernetes 主服务（由 apiserver 创建/维护）将是 NodePort 类型， 使用它作为端口的值。 如果为零，则 Kubernetes 主服务将为 ClusterIP 类型。

--lease-reuse-duration-seconds int 默认值：60

每个租约被重用的时长。 如果此值比较低，可以避免大量对象重用此租约。 注意，如果此值过小，可能导致存储层出现性能问题。

--livez-grace-period duration

此选项代表 API 服务器完成启动序列并生效所需的最长时间。 从 API 服务器的启动时间到这段时间为止， `/livez` 将假定未完成的启动后钩子将成功完成，因此返回 true。

--log-backtrace-at traceLocation 默认值：`:0`

当日志机制执行到'文件 :N'时，生成堆栈跟踪。

--log-dir string

如果为非空，则在此目录中写入日志文件。

--log-file string

如果为非空，使用此值作为日志文件。

--log-file-max-size uint 默认值：1800

定义日志文件可以增长到的最大大小。单位为兆字节。 如果值为 0，则最大文件大小为无限制。

--log-flush-frequency duration 默认值：5s

两次日志刷新之间的最大秒数

--logging-format string 默认值："text"

设置日志格式。允许的格式："text"。
非默认格式不支持以下标志： `--add-dir-header` 、 `--alsologtostderr` 、 `--log-backtrace-at` 、 `--log-dir` 、 `--log-file` 、 `--log-file-max-size` 、 `--logtostderr` 、 `--one-output` 、 `-skip-headers` 、 `-skip-log-headers` 、 `--stderrthreshold` 、 `-vmodule` 和 `--log-flush-frequency` 。
当前非默认选择为 alpha，会随时更改而不会发出警告。

--logtostderr 默认值：true

在标准错误而不是文件中输出日志记录。

--master-service-namespace string 默认值："default"

已废弃：应该从其中将 Kubernetes 主服务注入到 Pod 中的名字空间。

--max-connection-bytes-per-sec int

如果不为零，则将每个用户连接限制为该数（字节数/秒）。 当前仅适用于长时间运行的请求。

--max-mutating-requests-inflight int 默认值：200

如果 --enable-priority-and-fairness 为 true，那么此值和 --max-requests-inflight 的和将确定服务器的总并发限制（必须是正数）。 否则，该值限制进行中变更类型请求的最大个数，零表示无限制。

--max-requests-inflight int 默认值：400

如果 --enable-priority-and-fairness 为 true，那么此值和 --max-mutating-requests-inflight 的和将确定服务器的总并发限制（必须是正数）。 否则，该值限制进行中非变更类型请求的最大个数，零表示无限制。

--min-request-timeout int 默认值：1800

可选字段，表示处理程序在请求超时前，必须保持其处于打开状态的最小秒数。 当前只对监听（Watch）请求的处理程序有效，它基于这个值选择一个随机数作为连接超时值， 以达到分散负载的目的。

--oidc-ca-file string

如果设置该值，将会使用 oidc-ca-file 中的机构之一对 OpenID 服务的证书进行验证， 否则将会使用主机的根 CA 对其进行验证。

--oidc-client-id string

OpenID 连接客户端的要使用的客户 ID，如果设置了 oidc-issuer-url，则必须设置这个值。

--oidc-groups-claim string

如果提供该值，这个自定义 OpenID 连接声明将被用来设定用户组。 该声明值需要是一个字符串或字符串数组。 此标志为实验性的，请查阅身份认证相关文档进一步了解详细信息。

--oidc-groups-prefix string

如果提供了此值，则所有组都将以该值作为前缀，以防止与其他身份认证策略冲突。

--oidc-issuer-url string

OpenID 颁发者 URL，只接受 HTTPS 方案。 如果设置该值，它将被用于验证 OIDC JSON Web Token(JWT)。

--oidc-required-claim <逗号分隔的 'key=value' 键值对列表>

描述 ID 令牌中必需声明的键值对。 如果设置此值，则会验证 ID 令牌中存在与该声明匹配的值。 重复此标志以指定多个声明。

--oidc-signing-algs strings　默认值：RS256

允许的 JOSE 非对称签名算法的逗号分隔列表。 若 JWT 所带的 "alg" 标头值不在列表中，则该 JWT 将被拒绝。 取值依据 RFC 7518 https://tools.ietf.org/html/rfc7518#section-3.1 定义。

--oidc-username-claim string　默认值："sub"

要用作用户名的 OpenID 声明。 请注意，除默认声明（"sub"）以外的其他声明不能保证是唯一且不可变的。 此标志是实验性的，请参阅身份认证文档以获取更多详细信息。

--oidc-username-prefix string

如果提供，则所有用户名都将以该值作为前缀。 如果未提供，则除 "email" 之外的用户名声明都会添加颁发者 URL 作为前缀，以避免冲突。 要略过添加前缀处理，请设置值为 "-"。

--one-output

此标志为真时，日志只会被写入到其原生的严重性级别中（而不是同时写到所有较低 严重性级别中）。

--permit-address-sharing　默认值：false

若此标志为 true，则使用 SO_REUSEADDR 来绑定端口。 这样设置可以同时绑定到用通配符表示的类似 0.0.0.0 这种 IP 地址， 以及特定的 IP 地址。也可以避免等待内核释放 TIME_WAIT 状态的套接字。

--permit-port-sharing　默认值：false

如果为 true，则在绑定端口时将使用 SO_REUSEPORT， 这样多个实例可以绑定到同一地址和端口上。

--profiling　默认值：true

通过 Web 接口 `host:port/debug/pprof/` 启用性能分析。

--proxy-client-cert-file string

当必须调用外部程序以处理请求时，用于证明聚合器或者 kube-apiserver 的身份的客户端证书。 包括代理转发到用户 api-server 的请求和调用 Webhook 准入控制插件的请求。 Kubernetes 期望此证书包含来自于 --requestheader-client-ca-file 标志中所给 CA 的签名。 该 CA 在 kube-system 命名空间的 "extension-apiserver-authentication" ConfigMap 中公开。 从 kube-aggregator 收到调用的组件应该使用该 CA 进行各自的双向 TLS 验证。

--proxy-client-key-file string

当必须调用外部程序来处理请求时，用来证明聚合器或者 kube-apiserver 的身份的客户端私钥。 这包括代理转发给用户 api-server 的请求和调用 Webhook 准入控制插件的请求。

--request-timeout duration　默认值：1m0s

可选字段，指示处理程序在超时之前必须保持打开请求的持续时间。 这是请求的默认请求超时，但对于特定类型的请求，可能会被 `--min-request-timeout` 等标志覆盖。

--requestheader-allowed-names strings

此值为客户端证书通用名称（Common Name）的列表；表中所列的表项可以用来提供用户名， 方式是使用 `--requestheader-username-headers` 所指定的头部。 如果为空，能够通过 `--requestheader-client-ca-file` 中机构 认证的客户端证书都是被允许的。

--requestheader-client-ca-file string

在信任请求头中以 `--requestheader-username-headers` 指示的用户名之前， 用于验证接入请求中客户端证书的根证书包。 警告：一般不要假定传入请求已被授权。

--requestheader-extra-headers-prefix strings

用于查验请求头部的前缀列表。建议使用 `X-Remote-Extra-` 。

--requestheader-group-headers strings

用于查验用户组的请求头部列表。建议使用 `X-Remote-Group` 。

--requestheader-username-headers strings

用于查验用户名的请求头头列表。建议使用 `X-Remote-User` 。

--runtime-config <逗号分隔的 'key=value' 对列表>

一组启用或禁用内置 API 的键值对。支持的选项包括：
v1=true|false（针对核心 API 组）
<group>/<version>=true|false（针对特定 API 组和版本，例如：apps/v1=true）
api/all=true|false 控制所有 API 版本
api/ga=true|false 控制所有 v[0-9]+ API 版本
api/beta=true|false 控制所有 v[0-9]+beta[0-9]+ API 版本
api/alpha=true|false 控制所有 v[0-9]+alpha[0-9]+ API 版本
api/legacy 已弃用，并将在以后的版本中删除

--secure-port int　默认值：6443

带身份验证和鉴权机制的 HTTPS 服务端口。 不能用 0 关闭。

--service-account-extend-token-expiration　默认值：true

在生成令牌时，启用投射服务帐户到期时间扩展， 这有助于从旧版令牌安全地过渡到绑定的服务帐户令牌功能。 如果启用此标志，则准入插件注入的令牌的过期时间将延长至 1 年，以防止过渡期间发生意外故障， 并忽略 service-account-max-token-expiration 的值。

--service-account-issuer strings

服务帐号令牌颁发者的标识符。 颁发者将在已办法令牌的 "iss" 声明中检查此标识符。 此值为字符串或 URI。 如果根据 OpenID Discovery 1.0 规范检查此选项不是有效的 URI，则即使特性门控设置为 true， ServiceAccountIssuerDiscovery 功能也将保持禁用状态。 强烈建议该值符合 OpenID 规范：https://openid.net/specs/openid-connect-discovery-1_0.html。 实践中，这意味着 service-account-issuer 取值必须是 HTTPS URL。 还强烈建议此 URL 能够在 {service-account-issuer}/.well-known/openid-configuration 处提供 OpenID 发现文档。 当此值被多次指定时，第一次的值用于生成令牌，所有的值用于确定接受哪些发行人。

--service-account-jwks-uri string

覆盖 `/.well-known/openid-configuration` 提供的发现文档中 JSON Web 密钥集的 URI。 如果发现文档和密钥集是通过 API 服务器外部 （而非自动检测到或被外部主机名覆盖）之外的 URL 提供给依赖方的，则此标志很有用。 仅在启用 ServiceAccountIssuerDiscovery 特性门控的情况下有效。

--service-account-key-file strings

包含 PEM 编码的 x509 RSA 或 ECDSA 私钥或公钥的文件，用于验证 ServiceAccount 令牌。 指定的文件可以包含多个键，并且可以使用不同的文件多次指定标志。 如果未指定，则使用 `--tls-private-key-file` 。 提供 `--service-account-signing-key` 时必须指定。

--service-account-lookup　默认值：true

如果为 true，则在身份认证时验证 etcd 中是否存在 ServiceAccount 令牌。

--service-account-max-token-expiration duration

服务帐户令牌发布者创建的令牌的最长有效期。 如果请求有效期大于此值的有效令牌请求，将使用此值的有效期颁发令牌。

--service-account-signing-key-file string

包含服务帐户令牌颁发者当前私钥的文件的路径。 颁发者将使用此私钥签署所颁发的 ID 令牌。

--service-cluster-ip-range string

CIDR 表示的 IP 范围用来为服务分配集群 IP。 此地址不得与指定给节点或 Pod 的任何 IP 范围重叠。

--service-node-port-range <形式为 'N1-N2' 的字符串>　默认值：30000-32767

保留给具有 NodePort 可见性的服务的端口范围。 例如："30000-32767"。范围的两端都包括在内。

--show-hidden-metrics-for-version string

你要显示隐藏指标的先前版本。仅先前的次要版本有意义，不允许其他值。 格式为 <major>.<minor>，例如："1.16"。 这种格式的目的是确保你有机会注意到下一个版本是否隐藏了其他指标， 而不是在此之后将它们从发行版中永久删除时感到惊讶。

--shutdown-delay-duration duration

延迟终止时间。在此期间，服务器将继续正常处理请求。 端点 /healthz 和 /livez 将返回成功，但是 /readyz 立即返回失败。 在此延迟过去之后，将开始正常终止。 这可用于允许负载平衡器停止向该服务器发送流量。

--skip-headers

如果为 true，日志消息中避免标题前缀。

--skip-log-headers

如果为 true，则在打开日志文件时避免标题。

--stderrthreshold int　默认值：2

将达到或超过此阈值的日志写到标准错误输出

--storage-backend string

持久化存储后端。选项："etcd3"（默认）。

--storage-media-type string　默认值："application/vnd.kubernetes.protobuf"

用于在存储中存储对象的媒体类型。 某些资源或存储后端可能仅支持特定的媒体类型，并且将忽略此设置。

--strict-transport-security-directives strings

为 HSTS 所设置的指令列表，用逗号分隔。 如果此列表为空，则不会添加 HSTS 指令。 例如： 'max-age=31536000,includeSubDomains,preload'

--tls-cert-file string

包含用于 HTTPS 的默认 x509 证书的文件。（CA 证书（如果有）在服务器证书之后并置）。 如果启用了 HTTPS 服务，并且未提供 `--tls-cert-file` 和 `--tls-private-key-file` ， 为公共地址生成一个自签名证书和密钥，并将其保存到 `--cert-dir` 指定的目录中。

--tls-cipher-suites strings

服务器的密码套件的列表，以逗号分隔。如果省略，将使用默认的 Go 密码套件。
首选值： TLS_AES_128_GCM_SHA256、TLS_AES_256_GCM_SHA384、
TLS_CHACHA20_POLY1305_SHA256、TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA、
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256、
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA、
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384、
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305、
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256、
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA、
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA、
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256、
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA、
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384、
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305、
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256、
TLS_RSA_WITH_3DES_EDE_CBC_SHA、TLS_RSA_WITH_AES_128_CBC_SHA、
TLS_RSA_WITH_AES_128_GCM_SHA256、 TLS_RSA_WITH_AES_256_CBC_SHA,
TLS_RSA_WITH_AES_256_GCM_SHA384. 不安全的值有：
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256、
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA、
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256、TLS_ECDHE_RSA_WITH_RC4_128_SHA、
TLS_RSA_WITH_AES_128_CBC_SHA256、TLS_RSA_WITH_RC4_128_SHA。

--tls-min-version string

支持的最低 TLS 版本。可能的值：VersionTLS10，VersionTLS11，VersionTLS12，VersionTLS13

--tls-private-key-file string

包含匹配 `--tls-cert-file` 的 x509 证书私钥的文件。

--tls-sni-cert-key string　默认值：[]

一对 x509 证书和私钥文件路径，（可选）后缀为全限定域名的域名模式列表，可以使用带有通配符的前缀。 域模式也允许使用 IP 地址，但仅当 apiserver 对客户端请求的IP地址具有可见性时，才应使用 IP。 如果未提供域模式，则提取证书的名称。 非通配符匹配优先于通配符匹配，显式域模式优先于提取出的名称。 对于多个密钥/证书对，请多次使用 `--tls-sni-cert-key` 。 示例："example.crt,example.key" 或 "foo.crt,foo.key:*.foo.com,foo.com"。

--token-auth-file string

如果设置该值，这个文件将被用于通过令牌认证来保护 API 服务的安全端口。

--tracing-config-file string

包含 API 服务器跟踪配置的文件。

-v, --v int

日志级别详细程度的数字。

--version version[=true]

打印版本信息并退出

--vmodule <用逗号分隔的多个 'pattern=N' 配置字符串>

以逗号分隔的 `pattern=N` 设置列表，用于文件过滤的日志记录。

--watch-cache　默认值：true

在 API 服务器中启用监视缓存。

--watch-cache-sizes strings

某些资源（Pods、Nodes 等）的监视缓存大小设置，以逗号分隔。 每个资源对应的设置格式： `resource[.group]#size` ，其中 `resource` 为小写复数（无版本）， 对于 apiVersion v1（旧版核心 API）的资源要省略 `group` ， 对其它资源要给出 `group` ； `size` 为一个数字 。 启用 `watch-cache` 时，此功能生效。 某些资源（ `replicationcontrollers` 、 `endpoints` 、 `nodes` 、 `pods` 、 `services` 、 `apiservices.apiregistration.k8s.io` ） 具有通过启发式设置的系统默认值，其他资源默认为 `default-watch-cache-size` 。

4 - kube-controller-manager

简介

Kubernetes 控制器管理器是一个守护进程，内嵌随 Kubernetes 一起发布的核心控制回路。 在机器人和自动化的应用中，控制回路是一个永不休止的循环，用于调节系统状态。 在 Kubernetes 中，每个控制器是一个控制回路，通过 API 服务器监视集群的共享状态， 并尝试进行更改以将当前状态转为期望状态。 目前，Kubernetes 自带的控制器例子包括副本控制器、节点控制器、命名空间控制器和服务账号控制器等。

```
kube-controller-manager [flags]
```

选项

--add-dir-header

若为 true，将文件目录添加到日志消息的头部。

--allocate-node-cidrs

基于云驱动来为 Pod 分配和设置子网掩码。

--allow-metric-labels stringToString　默认值：""

从度量值标签到准许值列表的映射。键名的格式为<MetricName>,<LabelName>。 准许值的格式为<allowed_value>,<allowed_value>...。 例如， `metric1,label1='v1,v2,v3', metric1,label2='v1,v2,v3' metric2,label='v1,v2,v3'` 。

--alsologtostderr

在向文件输出日志的同时，也将日志写到标准输出。

--attach-detach-reconcile-sync-period duration　默认值：1m0s

协调器（reconciler）在相邻两次对存储卷进行挂载和解除挂载操作之间的等待时间。 此时长必须长于 1 秒钟。此值设置为大于默认值时，可能导致存储卷无法与 Pods 匹配。

--authentication-kubeconfig string

此标志值为一个 kubeconfig 文件的路径名。该文件中包含与某 Kubernetes “核心” 服务器相关的信息，并支持足够的权限以创建 tokenreviews.authentication.k8s.io。 此选项是可选的。如果设置为空值，所有令牌请求都会被认作匿名请求， Kubernetes 也不再在集群中查找客户端的 CA 证书信息。

--authentication-skip-lookup

此值为 false 时，通过 authentication-kubeconfig 参数所指定的文件会被用来 检索集群中缺失的身份认证配置信息。

--authentication-token-webhook-cache-ttl duration　默认值：10s

对 Webhook 令牌认证设施返回结果的缓存时长。

--authentication-tolerate-lookup-failure

此值为 true 时，即使无法从集群中检索到缺失的身份认证配置信息也无大碍。 需要注意的是，这样设置可能导致所有请求都被视作匿名请求。

--authorization-always-allow-paths strings　默认值："/healthz,/readyz,/livez"

鉴权过程中会忽略的一个 HTTP 路径列表。 换言之，控制器管理器会对列表中路径的访问进行授权，并且无须征得 Kubernetes “核心” 服务器同意。

--authorization-kubeconfig string

包含 Kubernetes “核心” 服务器信息的 kubeconfig 文件路径， 所包含信息具有创建 subjectaccessreviews.authorization.k8s.io 的足够权限。 此参数是可选的。如果配置为空字符串，未被鉴权模块所忽略的请求都会被禁止。

--authorization-webhook-cache-authorized-ttl duration　默认值：10s

对 Webhook 形式鉴权组件所返回的“已授权（Authorized）”响应的缓存时长。

--authorization-webhook-cache-unauthorized-ttl duration　默认值：10s

对 Webhook 形式鉴权组件所返回的“未授权（Unauthorized）”响应的缓存时长。

--azure-container-registry-config string

指向包含 Azure 容器仓库配置信息的文件的路径名。

--bind-address ip　默认值：0.0.0.0

针对 `--secure-port` 端口上请求执行监听操作的 IP 地址。 所对应的网络接口必须从集群中其它位置可访问（含命令行及 Web 客户端）。 如果此值为空或者设定为非特定地址（`0.0.0.0` 或 `::`）， 意味着所有网络接口都在监听范围。

--cert-dir string

TLS 证书所在的目录。如果提供了 `--tls-cert-file` 和 `--tls-private-key-file`，此标志会被忽略。

--cidr-allocator-type string　默认值："RangeAllocator"

要使用的 CIDR 分配器类型。

--client-ca-file string

如果设置了此标志，对于所有能够提供客户端证书的请求，若该证书由 `--client-ca-file` 中所给机构之一签署，则该请求会被 成功认证为客户端证书中 CommonName 所标识的实体。

--cloud-config string

云驱动程序配置文件的路径。空字符串表示没有配置文件。

--cloud-provider string

云服务的提供者。空字符串表示没有对应的提供者（驱动）。

--cluster-cidr string

集群中 Pods 的 CIDR 范围。要求 `--allocate-node-cidrs` 标志为 true。

--cluster-name string　默认值："kubernetes"

集群实例的前缀。

--cluster-signing-cert-file string

包含 PEM 编码格式的 X509 CA 证书的文件名。该证书用来发放集群范围的证书。 如果设置了此标志，则不能指定更具体的 `--cluster-signing-*` 标志。

--cluster-signing-duration duration　默认值：8760h0m0s

所签名证书的有效期限。每个 CSR 可以通过设置 spec.expirationSeconds 来请求更短的证书。

--cluster-signing-key-file string

包含 PEM 编码的 RSA 或 ECDSA 私钥的文件名。该私钥用来对集群范围证书签名。 若指定了此选项，则不可再设置 `--cluster-signing-*` 参数。

--cluster-signing-kube-apiserver-client-cert-file string

包含 PEM 编码的 X509 CA 证书的文件名， 该证书用于为 kubernetes.io/kube-apiserver-client 签署者颁发证书。 如果指定，则不得设置 `--cluster-signing-{cert,key}-file` 。

--cluster-signing-kube-apiserver-client-key-file string

包含 PEM 编码的 RSA 或 ECDSA 私钥的文件名， 该私钥用于为 kubernetes.io/kube-apiserver-client 签署者签名证书。 如果指定，则不得设置 `--cluster-signing-{cert,key}-file` 。

--cluster-signing-kubelet-client-cert-file string

包含 PEM 编码的 X509 CA 证书的文件名， 该证书用于为 kubernetes.io/kube-apiserver-client-kubelet 签署者颁发证书。 如果指定，则不得设置 `--cluster-signing-{cert,key}-file` 。

--cluster-signing-kubelet-client-key-file string

包含 PEM 编码的 RSA 或 ECDSA 私钥的文件名， 该私钥用于为 kubernetes.io/kube-apiserver-client-kubelet 签署者签名证书。 如果指定，则不得设置 `--cluster-signing-{cert,key}-file` 。

--cluster-signing-kubelet-serving-cert-file string

包含 PEM 编码的 X509 CA 证书的文件名， 该证书用于为 kubernetes.io/kubelet-serving 签署者颁发证书。 如果指定，则不得设置 --cluster-signing-{cert,key}-file。

--cluster-signing-kubelet-serving-key-file string

包含 PEM 编码的 RSA或ECDSA 私钥的文件名， 该私钥用于对 kubernetes.io/kubelet-serving 签署者的证书进行签名。 如果指定，则不得设置 `--cluster-signing-{cert,key}-file` 。

--cluster-signing-legacy-unknown-cert-file string

包含 PEM 编码的 X509 CA 证书的文件名， 用于为 kubernetes.io/legacy-unknown 签署者颁发证书。 如果指定，则不得设置 `--cluster-signing-{cert,key}-file` 。

--cluster-signing-legacy-unknown-key-file string

包含 PEM 编码的 RSA 或 ECDSA 私钥的文件名， 用于为 kubernetes.io/legacy-unknown 签署者签名证书。 如果指定，则不得设置 `--cluster-signing-{cert,key}-file` 。

--concurrent-deployment-syncs int32　默认值：5

可以并发同步的 Deployment 对象个数。数值越大意味着对 Deployment 的响应越及时，同时也意味着更大的 CPU（和网络带宽）压力。

--concurrent-endpoint-syncs int32　默认值：5

可以并发执行的 Endpoints 同步操作个数。数值越大意味着更快的 Endpoints 更新操作， 同时也意味着更大的 CPU （和网络）压力。

--concurrent-gc-syncs int32　默认值：20

可以并发同步的垃圾收集工作线程个数。

--concurrent-namespace-syncs int32　默认值：10

可以并发同步的 Namespace 对象个数。较大的数值意味着更快的名字空间终结操作， 不过也意味着更多的 CPU （和网络）占用。

--concurrent-rc-syncs int32　默认值：5

可以并发同步的副本控制器对象个数。较大的数值意味着更快的副本管理操作， 不过也意味着更多的 CPU （和网络）占用。

--concurrent-replicaset-syncs int32　默认值：5

可以并发同步的 ReplicaSet 个数。数值越大意味着副本管理的响应速度越快， 同时也意味着更多的 CPU （和网络）占用。

--concurrent-resource-quota-syncs int32　默认值：5

可以并发同步的 ResourceQuota 对象个数。数值越大，配额管理的响应速度越快， 不过对 CPU （和网络）的占用也越高。

--concurrent-service-endpoint-syncs int32　默认值：5

可以并发执行的服务端点同步操作个数。数值越大，端点片段（Endpoint Slice） 的更新速度越快，不过对 CPU （和网络）的占用也越高。默认值为 5。

--concurrent-service-syncs int32　默认值：1

可以并发同步的 Service 对象个数。数值越大，服务管理的响应速度越快， 不过对 CPU （和网络）的占用也越高。

--concurrent-serviceaccount-token-syncs int32　默认值：5

可以并发同步的服务账号令牌对象个数。数值越大，令牌生成的速度越快， 不过对 CPU （和网络）的占用也越高。

--concurrent-statefulset-syncs int32　默认值：5

可以并发同步的 StatefulSet 对象个数。数值越大，StatefulSet 管理的响应速度越快， 不过对 CPU （和网络）的占用也越高。

--concurrent-ttl-after-finished-syncs int32　默认值：5

可以并发同步的 TTL-after-finished 控制器线程个数。

--configure-cloud-routes　默认值：true

决定是否由 `--allocate-node-cidrs` 所分配的 CIDR 要通过云驱动程序来配置。

--contention-profiling

在启用了性能分析（profiling）时，也启用锁竞争情况分析。

--controller-start-interval duration

在两次启动控制器管理器之间的时间间隔。

--controllers strings　默认值：[*]

要启用的控制器列表。 `\*` 表示启用所有默认启用的控制器； `foo` 启用名为 foo 的控制器； `-foo` 表示禁用名为 foo 的控制器。
控制器的全集：attachdetach、bootstrapsigner、cloud-node-lifecycle、clusterrole-aggregation、cronjob、csrapproving、csrcleaner、csrsigning、daemonset、deployment、disruption、endpoint、endpointslice、endpointslicemirroring、ephemeral-volume、garbagecollector、horizontalpodautoscaling、job、namespace、nodeipam、nodelifecycle、persistentvolume-binder、persistentvolume-expander、podgc、pv-protection、pvc-protection、replicaset、replicationcontroller、resourcequota、root-ca-cert-publisher、route、service、serviceaccount、serviceaccount-token、statefulset、tokencleaner、ttl、ttl-after-finished
默认禁用的控制器有：bootstrapsigner 和 tokencleaner。

--deployment-controller-sync-period duration　默认值：30s

Deployment 资源的同步周期。

--disable-attach-detach-reconcile-sync

禁用卷挂接/解挂调节器的同步。禁用此同步可能导致卷存储与 Pod 之间出现错位。 请小心使用。

--disabled-metrics strings

此标志提供对行为异常的度量值的防控措施。你必须提供度量值的 完全限定名称才能将其禁用。**声明**：禁用度量值的操作比显示隐藏度量值 的操作优先级高。

--enable-dynamic-provisioning　默认值：true

在环境允许的情况下启用动态卷制备。

--enable-garbage-collector　默认值：true

启用通用垃圾收集器。必须与 kube-apiserver 中对应的标志一致。

--enable-hostpath-provisioner

在没有云驱动程序的情况下，启用 HostPath 持久卷的制备。 此参数便于对卷供应功能进行开发和测试。HostPath 卷的制备并非受支持的功能特性， 在多节点的集群中也无法工作，因此除了开发和测试环境中不应使用。

--enable-leader-migration

此标志决定是否启用控制器领导者迁移。

--enable-taint-manager　默认值：true

警告：此为Beta 阶段特性。设置为 true 时会启用 NoExecute 污点， 并在所有标记了此污点的节点上逐出所有无法忍受该污点的 Pods。

--endpoint-updates-batch-period duration

端点（Endpoint）批量更新周期时长。对 Pods 变更的处理会被延迟， 以便将其与即将到来的更新操作合并，从而减少端点更新操作次数。 较大的数值意味着端点更新的迟滞时间会增长，也意味着所生成的端点版本个数会变少。

--endpointslice-updates-batch-period duration

端点片段（Endpoint Slice）批量更新周期时长。对 Pods 变更的处理会被延迟， 以便将其与即将到来的更新操作合并，从而减少端点更新操作次数。 较大的数值意味着端点更新的迟滞时间会增长，也意味着所生成的端点版本个数会变少。

--experimental-logging-sanitization

[试验性功能] 当启用此标志时，被标记为敏感的字段（密码、密钥、令牌）不会被日志输出。
运行时的日志清理操作可能会引入相当程度的计算开销，因此不应在生产环境中启用。

--external-cloud-volume-plugin string

当云驱动程序设置为 external 时要使用的插件名称。此字符串可以为空。 只能在云驱动程序为 external 时设置。目前用来保证节点控制器和卷控制器能够 在三种云驱动上正常工作。

--feature-gates <逗号分隔的 'key=True|False' 对列表>

一组 key=value 对，用来描述测试性/试验性功能的特性门控（Feature Gate）。可选项有：
APIListChunking=true|false (BETA - 默认值=true)
APIPriorityAndFairness=true|false (BETA - 默认值=true)
APIResponseCompression=true|false (BETA - 默认值=true)
APIServerIdentity=true|false (ALPHA - 默认值=false)
APIServerTracing=true|false (ALPHA - 默认值=false)
AllAlpha=true|false (ALPHA - 默认值=false)
AllBeta=true|false (BETA - 默认值=false)
AnyVolumeDataSource=true|false (ALPHA - 默认值=false)
AppArmor=true|false (BETA - 默认值=true)
CPUManager=true|false (BETA - 默认值=true)
CPUManagerPolicyOptions=true|false (ALPHA - 默认值=false)
CSIInlineVolume=true|false (BETA - 默认值=true)
CSIMigration=true|false (BETA - 默认值=true)
CSIMigrationAWS=true|false (BETA - 默认值=false)
CSIMigrationAzureDisk=true|false (BETA - 默认值=false)
CSIMigrationAzureFile=true|false (BETA - 默认值=false)
CSIMigrationGCE=true|false (BETA - 默认值=false)
CSIMigrationOpenStack=true|false (BETA - 默认值=true)
CSIMigrationvSphere=true|false (BETA - 默认值=false)
CSIStorageCapacity=true|false (BETA - 默认值=true)
CSIVolumeFSGroupPolicy=true|false (BETA - 默认值=true)
CSIVolumeHealth=true|false (ALPHA - 默认值=false)
CSRDuration=true|false (BETA - 默认值=true)
ConfigurableFSGroupPolicy=true|false (BETA - 默认值=true)
ControllerManagerLeaderMigration=true|false (BETA - 默认值=true)
CustomCPUCFSQuotaPeriod=true|false (ALPHA - 默认值=false)
DaemonSetUpdateSurge=true|false (BETA - 默认值=true)
默认值PodTopologySpread=true|false (BETA - 默认值=true)
DelegateFSGroupToCSIDriver=true|false (ALPHA - 默认值=false)
DevicePlugins=true|false (BETA - 默认值=true)
DisableAcceleratorUsageMetrics=true|false (BETA - 默认值=true)
DisableCloudProviders=true|false (ALPHA - 默认值=false)
DownwardAPIHugePages=true|false (BETA - 默认值=false)
EfficientWatchResumption=true|false (BETA - 默认值=true)
EndpointSliceTerminatingCondition=true|false (BETA - 默认值=true)
EphemeralContainers=true|false (ALPHA - 默认值=false)
ExpandCSIVolumes=true|false (BETA - 默认值=true)
ExpandInUsePersistentVolumes=true|false (BETA - 默认值=true)
ExpandPersistentVolumes=true|false (BETA - 默认值=true)
ExpandedDNSConfig=true|false (ALPHA - 默认值=false)
ExperimentalHostUserNamespace默认值ing=true|false (BETA - 默认值=false)
GenericEphemeralVolume=true|false (BETA - 默认值=true)
GracefulNodeShutdown=true|false (BETA - 默认值=true)
HPAContainerMetrics=true|false (ALPHA - 默认值=false)
HPAScaleToZero=true|false (ALPHA - 默认值=false)
IPv6DualStack=true|false (BETA - 默认值=true)
InTreePluginAWSUnregister=true|false (ALPHA - 默认值=false)
InTreePluginAzureDiskUnregister=true|false (ALPHA - 默认值=false)
InTreePluginAzureFileUnregister=true|false (ALPHA - 默认值=false)
InTreePluginGCEUnregister=true|false (ALPHA - 默认值=false)
InTreePluginOpenStackUnregister=true|false (ALPHA - 默认值=false)
InTreePluginvSphereUnregister=true|false (ALPHA - 默认值=false)
IndexedJob=true|false (BETA - 默认值=true)
IngressClassNamespacedParams=true|false (BETA - 默认值=true)
JobTrackingWithFinalizers=true|false (ALPHA - 默认值=false)
KubeletCredentialProviders=true|false (ALPHA - 默认值=false)

KubeletInUserNamespace=true|false (ALPHA - 默认值=false)
KubeletPodResources=true|false (BETA - 默认值=true)
KubeletPodResourcesGetAllocatable=true|false (ALPHA - 默认值=false)
LocalStorageCapacityIsolation=true|false (BETA - 默认值=true)
LocalStorageCapacityIsolationFSQuotaMonitoring=true|false (ALPHA - 默认值=false)
LogarithmicScaleDown=true|false (BETA - 默认值=true)
MemoryManager=true|false (BETA - 默认值=true)
MemoryQoS=true|false (ALPHA - 默认值=false)
MixedProtocolLBService=true|false (ALPHA - 默认值=false)
NetworkPolicyEndPort=true|false (BETA - 默认值=true)
NodeSwap=true|false (ALPHA - 默认值=false)
NonPreemptingPriority=true|false (BETA - 默认值=true)
PodAffinityNamespaceSelector=true|false (BETA - 默认值=true)
PodDeletionCost=true|false (BETA - 默认值=true)
PodOverhead=true|false (BETA - 默认值=true)
PodSecurity=true|false (ALPHA - 默认值=false)
PreferNominatedNode=true|false (BETA - 默认值=true)
ProbeTerminationGracePeriod=true|false (BETA - 默认值=false)
ProcMountType=true|false (ALPHA - 默认值=false)
ProxyTerminatingEndpoints=true|false (ALPHA - 默认值=false)
QOSReserved=true|false (ALPHA - 默认值=false)
ReadWriteOncePod=true|false (ALPHA - 默认值=false)
RemainingItemCount=true|false (BETA - 默认值=true)
RemoveSelfLink=true|false (BETA - 默认值=true)
RotateKubeletServerCertificate=true|false (BETA - 默认值=true)
Seccomp默认值=true|false (ALPHA - 默认值=false)
ServiceInternalTrafficPolicy=true|false (BETA - 默认值=true)
ServiceLBNodePortControl=true|false (BETA - 默认值=true)
ServiceLoadBalancerClass=true|false (BETA - 默认值=true)
SizeMemoryBackedVolumes=true|false (BETA - 默认值=true)
StatefulSetMinReadySeconds=true|false (ALPHA - 默认值=false)
StorageVersionAPI=true|false (ALPHA - 默认值=false)
StorageVersionHash=true|false (BETA - 默认值=true)
SuspendJob=true|false (BETA - 默认值=true)
TTLAfterFinished=true|false (BETA - 默认值=true)
TopologyAwareHints=true|false (ALPHA - 默认值=false)
TopologyManager=true|false (BETA - 默认值=true)
VolumeCapacityPriority=true|false (ALPHA - 默认值=false)
WinDSR=true|false (ALPHA - 默认值=false)
WinOverlay=true|false (BETA - 默认值=true)
WindowsHostProcessContainers=true|false (ALPHA - 默认值=false)

--flex-volume-plugin-dir string　默认值："/usr/libexec/kubernetes/kubelet-plugins/volume/exec/"

FlexVolume 插件要搜索第三方卷插件的目录路径全名。

-h, --help

kube-controller-manager 的帮助信息。

--horizontal-pod-autoscaler-cpu-initialization-period duration　默认值：5m0s

Pod 启动之后可以忽略 CPU 采样值的时长。

--horizontal-pod-autoscaler-downscale-stabilization duration　默认值：5m0s

自动扩缩程序的回溯时长。自动扩缩器不会基于在给定的时长内所建议的规模 对负载执行规模缩小的操作。

--horizontal-pod-autoscaler-initial-readiness-delay duration　默认值：30s

Pod 启动之后，在此值所给定的时长内，就绪状态的变化都不会作为初始的就绪状态。

--horizontal-pod-autoscaler-sync-period duration　默认值：15s

水平 Pod 扩缩器对 Pods 数目执行同步操作的周期。

--horizontal-pod-autoscaler-tolerance float　默认值：0.1

此值为目标值与实际值的比值与 1.0 的差值。只有超过此标志所设的阈值时， HPA 才会考虑执行缩放操作。

--http2-max-streams-per-connection int

服务器为客户端所设置的 HTTP/2 连接中流式连接个数上限。 此值为 0 表示采用 Go 语言库所设置的默认值。

--kube-api-burst int32　默认值：30

与 Kubernetes API 服务器通信时突发峰值请求个数上限。

--kube-api-content-type string　默认值："application/vnd.kubernetes.protobuf"

向 API 服务器发送请求时使用的内容类型（Content-Type）。

--kube-api-qps float32　默认值：20

与 API 服务器通信时每秒请求数（QPS）限制。

--kubeconfig string

指向 kubeconfig 文件的路径。该文件中包含主控节点位置以及鉴权凭据信息。

--large-cluster-size-threshold int32　默认值：50

节点控制器在执行 Pod 逐出操作逻辑时，基于此标志所设置的节点个数阈值来判断 所在集群是否为大规模集群。当集群规模小于等于此规模时， `--secondary-node-eviction-rate` 会被隐式重设为 0。

--leader-elect　默认值：true

在执行主循环之前，启动领导选举（Leader Election）客户端，并尝试获得领导者身份。 在运行多副本组件时启用此标志有助于提高可用性。

--leader-elect-lease-duration duration　默认值：15s

对于未获得领导者身份的节点，在探测到领导者身份需要更迭时需要等待 此标志所设置的时长，才能尝试去获得曾经是领导者但尚未续约的席位。 本质上，这个时长也是现有领导者节点在被其他候选节点替代之前可以停止 的最长时长。只有集群启用了领导者选举机制时，此标志才起作用。

--leader-elect-renew-deadline duration　默认值：10s

当前执行领导者角色的节点在被停止履行领导职责之前可多次尝试续约领导者身份； 此标志给出相邻两次尝试之间的间歇时长。 此值必须小于或等于租期时长（Lease Duration）。 仅在集群启用了领导者选举时有效。

--leader-elect-resource-lock string　默认值："leases"

在领导者选举期间用于锁定的资源对象的类型。 支持的选项为 "endpoints"、"configmaps"、"leases"、"endpointsleases" 和 "configmapsleases"。

--leader-elect-resource-name string　默认值："kube-controller-manager"

在领导者选举期间，用来执行锁操作的资源对象名称。

--leader-elect-resource-namespace string　默认值："kube-system"

在领导者选举期间，用来执行锁操作的资源对象的名字空间。

--leader-elect-retry-period duration　默认值：2s

尝试获得领导者身份时，客户端在相邻两次尝试之间要等待的时长。 此标志仅在启用了领导者选举的集群中起作用。

--leader-migration-config string

控制器领导者迁移所用的配置文件路径。 此值为空意味着使用控制器管理器的默认配置。 配置文件应该是 `controllermanager.config.k8s.io` 组、 `v1alpha1` 版本的 `LeaderMigrationConfiguration` 结构。

--log-backtrace-at traceLocation　默认值： `:0`

当执行到 `file:N` 所给的文件和代码行时，日志机制会生成一个调用栈快照。

--log-dir string

此标志为非空字符串时，日志文件会写入到所给的目录中。

--log-file string

此标志为非空字符串时，意味着日志会写入到所给的文件中。

--log-file-max-size uint　默认值：1800

定义日志文件大小的上限。单位是兆字节（MB）。 若此值为 0，则不对日志文件尺寸进行约束。

--log-flush-frequency duration　默认值：5s

将内存中日志数据清除到日志文件中时，相邻两次清除操作之间最大间隔秒数。

--logging-format string　默认值："text"

设置日志格式。允许的格式："text"。
非默认格式不支持以下标志： `--add-dir-header` 、 `--alsologtostderr` 》、 `--log-backtrace-at` 、 `--log-dir` 、 `--log-file` 、 `--log-file-max-size` 、 `--logtostderr` 、 `--one-output` 、 `--skip-headers` 、 `--skip-log-headers` 、 `--stderrthreshold` 、 `--vmodule` 、 `--log-flush-frequency` 。
当前非默认选项为 Alpha 阶段，如有更改，恕不另行通知。

--logtostderr　默认值：true

将日志写出到标准错误输出（stderr）而不是写入到日志文件。

--master string

Kubernetes API 服务器的地址。此值会覆盖 kubeconfig 文件中所给的地址。

--max-endpoints-per-slice int32　默认值：100

每个 EndpointSlice 中可以添加的端点个数上限。每个片段中端点个数越多， 得到的片段个数越少，但是片段的规模会变得更大。默认值为 100。

--min-resync-period duration　默认值：12h0m0s

自省程序的重新同步时隔下限。实际时隔长度会在 `min-resync-period` 和 `2 * min-resync-period` 之间。

--mirroring-concurrent-service-endpoint-syncs int32　默认值：5

EndpointSliceMirroring 控制器将同时执行的服务端点同步操作数。 较大的数量 = 更快的端点切片更新，但 CPU（和网络）负载更多。 默认为 5。

--mirroring-endpointslice-updates-batch-period duration

EndpointSlice 的长度更新了 EndpointSliceMirroring 控制器的批处理周期。 EndpointSlice 更改的处理将延迟此持续时间， 以使它们与潜在的即将进行的更新结合在一起，并减少 EndpointSlice 更新的总数。 较大的数量 = 较高的端点编程延迟，但是生成的端点修订版本数量较少

--mirroring-max-endpoints-per-subset int32　默认值：1000

EndpointSliceMirroring 控制器将添加到 EndpointSlice 的最大端点数。 每个分片的端点越多，端点分片越少，但资源越大。

--namespace-sync-period duration　默认值：5m0s

对名字空间对象进行同步的周期。

--node-cidr-mask-size int32

集群中节点 CIDR 的掩码长度。对 IPv4 而言默认为 24；对 IPv6 而言默认为 64。

--node-cidr-mask-size-ipv4 int32

在双堆栈（同时支持 IPv4 和 IPv6）的集群中，节点 IPV4 CIDR 掩码长度。默认为 24。

--node-cidr-mask-size-ipv6 int32

在双堆栈（同时支持 IPv4 和 IPv6）的集群中，节点 IPv6 CIDR 掩码长度。默认为 64。

--node-eviction-rate float32　默认值：0.1

当某区域变得不健康，节点失效时，每秒钟可以从此标志所设定的节点 个数上删除 Pods。 请参阅 `--unhealthy-zone-threshold` 以了解“健康”的判定标准。这里的区域（zone）在集群并不跨多个区域时 指的是整个集群。

--node-monitor-grace-period duration　默认值：40s

在将一个 Node 标记为不健康之前允许其无响应的时长上限。 必须比 kubelet 的 nodeStatusUpdateFrequency 大 N 倍； 这里 N 指的是 kubelet 发送节点状态的重试次数。

--node-monitor-period duration　默认值：5s

节点控制器对节点状态进行同步的重复周期。

--node-startup-grace-period duration　默认值：1m0s

在节点启动期间，节点可以处于无响应状态； 但超出此标志所设置的时长仍然无响应则该节点被标记为不健康。

--one-output

如果此标志为 true，则仅将日志写入其自身的严重性级别（而不是同时写入更低的严重性级别中）。

--permit-address-sharing

如果此标志为 true，则在绑定端口时使用 `SO_REUSEADDR` 。 这就意味着可以同时绑定到 `0.0.0.0` 和特定的 IP 地址， 并且避免等待内核释放处于 `TIME_WAITE` 状态的套接字。

--permit-port-sharing

如果为 true，则在绑定端口时将使用 `SO_REUSEPORT`， 这允许多个实例在同一地址和端口上进行绑定。

--pod-eviction-timeout duration　默认值：5m0s

在失效的节点上删除 Pods 时为其预留的宽限期。

--profiling　默认值：true

通过位于 `host:port/debug/pprof/` 的 Web 接口启用性能分析。

--pv-recycler-increment-timeout-nfs int32　默认值：30

NFS 清洗 Pod 在清洗用过的卷时，根据此标志所设置的秒数，为每清洗 1 GiB 数据 增加对应超时时长，作为 activeDeadlineSeconds。

--pv-recycler-minimum-timeout-hostpath int32　默认值：60

对于 HostPath 回收器 Pod，设置其 activeDeadlineSeconds 参数下限。 此参数仅用于开发和测试目的，不适合在多节点集群中使用。

--pv-recycler-minimum-timeout-nfs int32　默认值：300

NFS 回收器 Pod 要使用的 activeDeadlineSeconds 参数下限。

--pv-recycler-pod-template-filepath-hostpath string

对 HostPath 持久卷进行回收利用时，用作模版的 Pod 定义文件所在路径。 此标志仅用于开发和测试目的，不适合多节点集群中使用。

--pv-recycler-pod-template-filepath-nfs string

对 NFS 卷执行回收利用时，用作模版的 Pod 定义文件所在路径。

--pv-recycler-timeout-increment-hostpath int32　默认值：30

HostPath 清洗器 Pod 在清洗对应类型持久卷时，为每 GiB 数据增加此标志所设置的秒数，作为其 activeDeadlineSeconds 参数。此标志仅用于开发和测试环境，不适合多节点集群环境。

--pvclaimbinder-sync-period duration　默认值：15s

持久卷（PV）和持久卷申领（PVC）对象的同步周期。

--requestheader-allowed-names strings

标志值是客户端证书中的 Common Names 列表。其中所列的名称可以通过 `--requestheader-username-headers` 所设置的 HTTP 头部来提供用户名。 如果此标志值为空表，则被 `--requestheader-client-ca-file` 中机构所验证过的所有客户端证书都是允许的。

--requestheader-client-ca-file string

根证书包文件名。在信任通过 `--requestheader-username-headers` 所指定的任何用户名之前，要使用这里的证书来检查请求中的客户证书。 警告：一般不要依赖对请求所作的鉴权结果。

--requestheader-extra-headers-prefix strings　默认值： `"x-remote-extra-"`

要插入的请求头部前缀。建议使用 `X-Remote-Exra-` 。

--requestheader-group-headers strings　默认值： `"x-remote-group"`

用来检查用户组名的请求头部名称列表。建议使用 `X-Remote-Group` 。

--requestheader-username-headers strings　默认值： `"x-remote-user"`

用来检查用户名的请求头部名称列表。建议使用 `X-Remote-User` 。

--resource-quota-sync-period duration　默认值：5m0s

对系统中配额用量信息进行同步的周期。

--root-ca-file string

如果此标志非空，则在服务账号的令牌 Secret 中会包含此根证书机构。 所指定标志值必须是一个合法的 PEM 编码的 CA 证书包。

--route-reconciliation-period duration　默认值：10s

对云驱动为节点所创建的路由信息进行调解的周期。

--secondary-node-eviction-rate float32　默认值：0.01

当区域不健康，节点失效时，每秒钟从此标志所给的节点个数上删除 Pods。 参见 `--unhealthy-zone-threshold` 以了解“健康与否”的判定标准。 在只有一个区域的集群中，区域指的是整个集群。如果集群规模小于 `--large-cluster-size-threshold` 所设置的节点个数时， 此值被隐式地重设为 0。

--secure-port int　默认值：10257

在此端口上提供 HTTPS 身份认证和鉴权操作。若此标志值为 0，则不提供 HTTPS 服务。

--service-account-private-key-file string

包含 PEM 编码的 RSA 或 ECDSA 私钥数据的文件名，这些私钥用来对服务账号令牌签名。

--service-cluster-ip-range string

集群中 Service 对象的 CIDR 范围。要求 `--allocate-node-cidrs` 标志为 true。

--show-hidden-metrics-for-version string

你希望展示隐藏度量值的上一个版本。只有上一个次版本号有意义，其他值都是不允许的。字符串格式为 "<major>.<minor>"。例如：“1.16"。 此格式的目的是确保你能够有机会注意到下一个版本隐藏了一些额外的度量值， 而不是在更新版本中某些度量值被彻底删除时措手不及。

--skip-headers

若此标志为 true，则在日志消息中避免写入头部前缀信息。

--skip-log-headers

若此标志为 true，则在写入日志文件时避免写入头部信息。

--stderrthreshold severity　默认值：2

等于或大于此阈值的日志信息会被写入到标准错误输出（stderr）。

--terminated-pod-gc-threshold int32　默认值：12500

在已终止 Pods 垃圾收集器删除已终止 Pods 之前，可以保留的已删除 Pods 的个数上限。若此值小于等于 0，则相当于禁止垃圾回收已终止的 Pods。

--tls-cert-file string

包含 HTTPS 所用的默认 X509 证书的文件。如果有 CA 证书，会被串接在服务器证书之后。若启用了 HTTPS 服务且 `--tls-cert-file` 和 `--tls-private-key-file` 标志未设置，则为节点的公开地址生成自签名的证书和密钥，并保存到 `--cert-dir` 所给的目录中。

--tls-cipher-suites strings

供服务器使用的加密包的逗号分隔列表。若忽略此标志，则使用 Go 语言默认的加密包。
可选值包括：TLS_AES_128_GCM_SHA256、TLS_AES_256_GCM_SHA384、
TLS_CHACHA20_POLY1305_SHA256、TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA、
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256、
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA、
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384、
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305、
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256、
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA、
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA、
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256、
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA、
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384、
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305、
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256、
TLS_RSA_WITH_3DES_EDE_CBC_SHA、TLS_RSA_WITH_AES_128_CBC_SHA、
TLS_RSA_WITH_AES_128_GCM_SHA256、TLS_RSA_WITH_AES_256_CBC_SHA、
TLS_RSA_WITH_AES_256_GCM_SHA384.
不安全的值: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256、
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA、
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256、TLS_ECDHE_RSA_WITH_RC4_128_SHA、
TLS_RSA_WITH_AES_128_CBC_SHA256、TLS_RSA_WITH_RC4_128_SHA

--tls-min-version string

可支持的最低 TLS 版本。可选值包括： “VersionTLS10”、“VersionTLS11”、
“VersionTLS12”、“VersionTLS13”。

--tls-private-key-file string

包含与 `--tls-cert-file` 对应的默认 X509 私钥的文件。

--tls-sni-cert-key namedCertKey　默认值：[]

X509 证书和私钥文件路径的耦对。作为可选项，可以添加域名模式的列表， 其中每个域名模式都是可以带通配片段前缀的全限定域名（FQDN）。 域名模式也可以使用 IP 地址字符串，不过只有 API 服务器在所给 IP 地址上 对客户端可见时才可以使用 IP 地址。在未提供域名模式时，从证书中提取域名。 如果有非通配方式的匹配，则优先于通配方式的匹配；显式的域名模式优先于提取的域名。 当存在多个密钥/证书耦对时，可以多次使用 `--tls-sni-cert-key` 标志。 例如： `example.crt,example.key` 或 `foo.crt,foo.key:\*.foo.com,foo.com` 。

--unhealthy-zone-threshold float32　默认值：0.55

仅当给定区域中处于非就绪状态的节点（最少 3 个）的占比高于此值时， 才将该区域视为不健康。

--use-service-account-credentials

当此标志为 true 时，为每个控制器单独使用服务账号凭据。

-v, --v int

日志级别详细程度取值。

--version version[=true]

打印版本信息之后退出。

--vmodule <逗号分隔的 'pattern=N' 配置值>

由逗号分隔的列表，每一项都是 pattern=N 格式，用来执行根据文件过滤的日志行为。

--volume-host-allow-local-loopback　默认值：true

此标志为 false 时，禁止本地回路 IP 地址和 `--volume-host-cidr-denylist` 中所指定的 CIDR 范围。

--volume-host-cidr-denylist strings

用逗号分隔的一个 CIDR 范围列表，禁止使用这些地址上的卷插件。

5 - kube-proxy

简介

Kubernetes 网络代理在每个节点上运行。网络代理反映了每个节点上 Kubernetes API 中定义的服务，并且可以执行简单的 TCP、UDP 和 SCTP 流转发，或者在一组后端进行 循环 TCP、UDP 和 SCTP 转发。 当前可通过 Docker-links-compatible 环境变量找到服务集群 IP 和端口， 这些环境变量指定了服务代理打开的端口。 有一个可选的插件，可以为这些集群 IP 提供集群 DNS。 用户必须使用 apiserver API 创建服务才能配置代理。

```
kube-proxy [flags]
```

选项

--add-dir-header

若此标志为 true，则将文件目录添加到日志消息的头部。

--alsologtostderr

将日志输出到文件时也输出到标准错误输出（stderr）。

--azure-container-registry-config string

包含 Azure 容器仓库配置信息的文件的路径。

--bind-address 0.0.0.0　默认值：0.0.0.0

代理服务器要使用的 IP 地址（设置为 '0.0.0.0' 表示要使用所有 IPv4 接口； 设置为 '::' 表示使用所有 IPv6 接口）。

--bind-address-hard-fail

若此标志为 true，kube-proxy 会将无法绑定端口的失败操作视为致命错误并退出。

--boot-id-file string　默认值："/proc/sys/kernel/random/boot_id"

用来检查 Boot-ID 的文件名，用逗号隔开。 第一个存在的文件会被使用。

--cleanup

如果为 true，清理 iptables 和 ipvs 规则并退出。

--cluster-cidr string

集群中 Pod 的 CIDR 范围。配置后，将从该范围之外发送到服务集群 IP 的流量被伪装，从 Pod 发送到外部 LoadBalancer IP 的流量将被重定向 到相应的集群 IP。

--config string

配置文件的路径。

--config-sync-period duration　默认值：15m0s

来自 apiserver 的配置的刷新频率。必须大于 0。

--conntrack-max-per-core int32　默认值：32768

每个 CPU 核跟踪的最大 NAT 连接数（0 表示保留当前限制并忽略 conntrack-min 设置）。

--conntrack-min int32　默认值：131072

无论 `conntrack-max-per-core` 多少，要分配的 conntrack 条目的最小数量（将 `conntrack-max-per-core` 设置为 0 即可 保持当前的限制）。

--conntrack-tcp-timeout-close-wait duration　默认值：1h0m0s

处于 `CLOSE_WAIT` 状态的 TCP 连接的 NAT 超时。

--conntrack-tcp-timeout-established duration　默认值：24h0m0s

已建立的 TCP 连接的空闲超时（0 保持当前设置）。

--detect-local-mode LocalMode

用于检测本地流量的模式。

--feature-gates <逗号分隔的 'key=True|False' 对’>

一组键=值（key=value）对，描述了 alpha/experimental 的特征。可选项有：
APIListChunking=true|false (BETA - 默认值=true)
APIPriorityAndFairness=true|false (BETA - 默认值=true)
APIResponseCompression=true|false (BETA - 默认值=true)
APIServerIdentity=true|false (ALPHA - 默认值=false)
APIServerTracing=true|false (ALPHA - 默认值=false)
AllAlpha=true|false (ALPHA - 默认值=false)
AllBeta=true|false (BETA - 默认值=false)
AnyVolumeDataSource=true|false (ALPHA - 默认值=false)
AppArmor=true|false (BETA - 默认值=true)
CPUManager=true|false (BETA - 默认值=true)
CPUManagerPolicyOptions=true|false (ALPHA - 默认值=false)
CSIInlineVolume=true|false (BETA - 默认值=true)
CSIMigration=true|false (BETA - 默认值=true)
CSIMigrationAWS=true|false (BETA - 默认值=false)
CSIMigrationAzureDisk=true|false (BETA - 默认值=false)
CSIMigrationAzureFile=true|false (BETA - 默认值=false)
CSIMigrationGCE=true|false (BETA - 默认值=false)
CSIMigrationOpenStack=true|false (BETA - 默认值=true)
CSIMigrationvSphere=true|false (BETA - 默认值=false)
CSIStorageCapacity=true|false (BETA - 默认值=true)
CSIVolumeFSGroupPolicy=true|false (BETA - 默认值=true)
CSIVolumeHealth=true|false (ALPHA - 默认值=false)
CSRDuration=true|false (BETA - 默认值=true)
ConfigurableFSGroupPolicy=true|false (BETA - 默认值=true)
ControllerManagerLeaderMigration=true|false (BETA - 默认值=true)
CustomCPUCFSQuotaPeriod=true|false (ALPHA - 默认值=false)
DaemonSetUpdateSurge=true|false (BETA - 默认值=true)
DefaultPodTopologySpread=true|false (BETA - 默认值=true)
DelegateFSGroupToCSIDriver=true|false (ALPHA - 默认值=false)
DevicePlugins=true|false (BETA - 默认值=true)
DisableAcceleratorUsageMetrics=true|false (BETA - 默认值=true)
DisableCloudProviders=true|false (ALPHA - 默认值=false)
DownwardAPIHugePages=true|false (BETA - 默认值=false)
EfficientWatchResumption=true|false (BETA - 默认值=true)
EndpointSliceTerminatingCondition=true|false (BETA - 默认值=true)

EphemeralContainers=true|false (ALPHA - 默认值=false)
ExpandCSIVolumes=true|false (BETA - 默认值=true)
ExpandInUsePersistentVolumes=true|false (BETA - 默认值=true)
ExpandPersistentVolumes=true|false (BETA - 默认值=true)
ExpandedDNSConfig=true|false (ALPHA - 默认值=false)
ExperimentalHostUserNamespaceDefaulting=true|false (BETA - 默认值=false)
GenericEphemeralVolume=true|false (BETA - 默认值=true)
GracefulNodeShutdown=true|false (BETA - 默认值=true)
HPAContainerMetrics=true|false (ALPHA - 默认值=false)
HPAScaleToZero=true|false (ALPHA - 默认值=false)
IPv6DualStack=true|false (BETA - 默认值=true)
InTreePluginAWSUnregister=true|false (ALPHA - 默认值=false)
InTreePluginAzureDiskUnregister=true|false (ALPHA - 默认值=false)
InTreePluginAzureFileUnregister=true|false (ALPHA - 默认值=false)
InTreePluginGCEUnregister=true|false (ALPHA - 默认值=false)
InTreePluginOpenStackUnregister=true|false (ALPHA - 默认值=false)
InTreePluginvSphereUnregister=true|false (ALPHA - 默认值=false)
IndexedJob=true|false (BETA - 默认值=true)
IngressClassNamespacedParams=true|false (BETA - 默认值=true)
JobTrackingWithFinalizers=true|false (ALPHA - 默认值=false)
KubeletCredentialProviders=true|false (ALPHA - 默认值=false)
KubeletInUserNamespace=true|false (ALPHA - 默认值=false)
KubeletPodResources=true|false (BETA - 默认值=true)
KubeletPodResourcesGetAllocatable=true|false (ALPHA - 默认值=false)
LocalStorageCapacityIsolation=true|false (BETA - 默认值=true)
LocalStorageCapacityIsolationFSQuotaMonitoring=true|false (ALPHA - 默认值=false)
LogarithmicScaleDown=true|false (BETA - 默认值=true)
MemoryManager=true|false (BETA - 默认值=true)
MemoryQoS=true|false (ALPHA - 默认值=false)
MixedProtocolLBService=true|false (ALPHA - 默认值=false)
NetworkPolicyEndPort=true|false (BETA - 默认值=true)
NodeSwap=true|false (ALPHA - 默认值=false)
NonPreemptingPriority=true|false (BETA - 默认值=true)
PodAffinityNamespaceSelector=true|false (BETA - 默认值=true)
PodDeletionCost=true|false (BETA - 默认值=true)
PodOverhead=true|false (BETA - 默认值=true)
PodSecurity=true|false (ALPHA - 默认值=false)
PreferNominatedNode=true|false (BETA - 默认值=true)
ProbeTerminationGracePeriod=true|false (BETA - 默认值=false)
ProcMountType=true|false (ALPHA - 默认值=false)
ProxyTerminatingEndpoints=true|false (ALPHA - 默认值=false)
QOSReserved=true|false (ALPHA - 默认值=false)
ReadWriteOncePod=true|false (ALPHA - 默认值=false)
RemainingItemCount=true|false (BETA - 默认值=true)
RemoveSelfLink=true|false (BETA - 默认值=true)
RotateKubeletServerCertificate=true|false (BETA - 默认值=true)
SeccompDefault=true|false (ALPHA - 默认值=false)
ServiceInternalTrafficPolicy=true|false (BETA - 默认值=true)
ServiceLBNodePortControl=true|false (BETA - 默认值=true)
ServiceLoadBalancerClass=true|false (BETA - 默认值=true)
SizeMemoryBackedVolumes=true|false (BETA - 默认值=true)
StatefulSetMinReadySeconds=true|false (ALPHA - 默认值=false)
StorageVersionAPI=true|false (ALPHA - 默认值=false)
StorageVersionHash=true|false (BETA - 默认值=true)
SuspendJob=true|false (BETA - 默认值=true)
TTLAfterFinished=true|false (BETA - 默认值=true)
TopologyAwareHints=true|false (ALPHA - 默认值=false)
TopologyManager=true|false (BETA - 默认值=true)
VolumeCapacityPriority=true|false (ALPHA - 默认值=false)
WinDSR=true|false (ALPHA - 默认值=false)
WinOverlay=true|false (BETA - 默认值=true)
WindowsHostProcessContainers=true|false (ALPHA - 默认值=false)

--healthz-bind-address 0.0.0.0　默认值：`0.0.0.0:10256`

服务健康状态检查的 IP 地址和端口（设置为 '0.0.0.0:10256' 表示使用所有 IPv4 接口，设置为 '[::]:10256' 表示使用所有 IPv6 接口）； 设置为空则禁用。

-h, --help

---

kube-proxy 操作的帮助命令。

--hostname-override string

如果非空，将使用此字符串而不是实际的主机名作为标识。

--iptables-masquerade-bit int32　默认值：14

在使用纯 iptables 代理时，用来设置 fwmark 空间的 bit，标记需要 SNAT 的数据包。必须在 [0,31] 范围内。

--iptables-min-sync-period duration　默认值：1s

iptables 规则可以随着端点和服务的更改而刷新的最小间隔（例如 '5s'、'1m'、'2h22m'）。

--iptables-sync-period duration　默认值：30s

--ipvs-exclude-cidrs strings

逗号分隔的 CIDR 列表，ipvs 代理在清理 IPVS 规则时不会此列表中的地址范围。

--ipvs-min-sync-period duration

ipvs 规则可以随着端点和服务的更改而刷新的最小间隔（例如 '5s'、'1m'、'2h22m'）。

--ipvs-scheduler string

代理模式为 ipvs 时所选的 ipvs 调度器类型。

--ipvs-strict-arp

通过将 `arp_ignore` 设置为 1 并将 `arp_announce` 设置为 2 启用严格的 ARP。

--ipvs-sync-period duration　默认值：30s

刷新 ipvs 规则的最大间隔（例如 '5s'、'1m'、'2h22m'）。必须大于 0。

--ipvs-tcp-timeout duration

空闲 IPVS TCP 连接的超时时间，0 保持连接（例如 '5s'、'1m'、'2h22m'）。

--ipvs-tcpfin-timeout duration

收到 FIN 数据包后，IPVS TCP 连接的超时，0 保持当前设置不变。（例如 '5s'、'1m'、'2h22m'）。

--ipvs-udp-timeout duration

IPVS UDP 数据包的超时，0 保持当前设置不变。（例如 '5s'、'1m'、'2h22m'）。

--kube-api-burst int32　默认值：10

与 kubernetes apiserver 通信的突发数量。

--kube-api-content-type string　默认值："application/vnd.kubernetes.protobuf"

发送到 apiserver 的请求的内容类型。

--kube-api-qps float32　默认值：5

与 kubernetes apiserver 交互时使用的 QPS。

--kubeconfig string

包含鉴权信息的 kubeconfig 文件的路径（主控节点位置由 master 标志设置）。

--log-backtrace-at <形式为 'file:N' 的字符串>　Default: :0

当日志逻辑执行到文件 file 的第 N 行时，输出调用堆栈跟踪。

--log-dir string

若此标志费控，则将日志文件写入到此标志所给的目录下。

--log-file string

若此标志非空，则该字符串作为日志文件名。

--log-file-max-size uint　默认值：1800

定义日志文件可增长到的最大尺寸。单位是兆字节（MB）。 如果此值为 0，则最大文件大小无限制。

--log-flush-frequency duration　默认值：5s

两次日志刷新之间的最大秒数。

--machine-id-file string　默认值："/etc/machine-id,/var/lib/dbus/machine-id"

用来检查 Machine-ID 的文件列表，用逗号分隔。 使用找到的第一个文件。

--masquerade-all

如果使用纯 iptables 代理，则对通过服务集群 IP 发送的所有流量 进行 SNAT（通常不需要）。

--master string

Kubernetes API 服务器的地址（覆盖 kubeconfig 中的相关值）。

--metrics-bind-address ipport　默认值： `127.0.0.1:10249`

metrics 服务器要使用的 IP 地址和端口 （设置为 '0.0.0.0:10249' 则使用所有 IPv4 接口，设置为 '[::]:10249' 则使用所有 IPv6 接口） 设置为空则禁用。

--nodeport-addresses strings

一个字符串值，指定用于 NodePort 服务的地址。 值可以是有效的 IP 块（例如 1.2.3.0/24, 1.2.3.4/32）。 默认的空字符串切片（[]）表示使用所有本地地址。

--one-output

若此标志为 true，则仅将日志写入到其原本的严重性级别之下 （而不是将其写入到所有更低严重性级别中）。

--oom-score-adj int32　默认值：-999

kube-proxy 进程中的 oom-score-adj 值，必须在 [-1000,1000] 范围内。

--profiling

如果为 true，则通过 Web 接口 `/debug/pprof` 启用性能分析。

--proxy-mode string

使用哪种代理模式：'userspace'（较旧）或 'iptables'（较快）或 'ipvs'。 如果为空，使用最佳可用代理（当前为 iptables）。 如果选择了 iptables 代理（无论是否为显式设置），但系统的内核或 iptables 版本较低，总是会回退到 userspace 代理。

--proxy-port-range port-range

可以用来代理服务流量的主机端口范围（包括'起始端口-结束端口'、 '单个端口'、'起始端口+偏移'几种形式）。 如果未指定或者设置为 0（或 0-0），则随机选择端口。

--show-hidden-metrics-for-version string

要显示隐藏指标的先前版本。 仅先前的次要版本有意义，不允许其他值。 格式为 <major>.<minor>，例如：'1.16'。 这种格式的目的是确保你有机会注意到下一个发行版是否隐藏了其他指标， 而不是在之后将其永久删除时感到惊讶。

--skip-headers

若此标志为 true，则避免在日志消息中包含头部前缀。

--skip-log-headers

如果此标志为 true，则避免在打开日志文件时使用头部。

--stderrthreshold int　默认值：2

如果日志消息处于或者高于此阈值所设置的级别，则将其输出到标准错误输出（stderr）。

--udp-timeout duration　默认值：250ms

空闲 UDP 连接将保持打开的时长（例如 '250ms'，'2s'）。必须大于 0。 仅适用于 proxy-mode=userspace。

-v, --v int

用来设置日志详细程度的数值。

--version version[=true]

打印版本信息并退出。

--vmodule <逗号分隔的 'pattern=N' 设置’>

用逗号分隔的列表，其中每一项为 'pattern=N' 格式。 用来支持基于文件过滤的日志机制。

--write-config-to string

如果设置，将默认配置信息写入此文件并退出。

6 - kube-scheduler

简介

Kubernetes 调度器是一个控制面进程，负责将 Pods 指派到节点上。 调度器基于约束和可用资源为调度队列中每个 Pod 确定其可合法放置的节点。 调度器之后对所有合法的节点进行排序，将 Pod 绑定到一个合适的节点。 在同一个集群中可以使用多个不同的调度器；kube-scheduler 是其参考实现。 参阅调度 以获得关于调度和 kube-scheduler 组件的更多信息。

```
kube-scheduler [flags]
```

选项

--add-dir-header

如果为 true，则将文件目录添加到日志消息的头部

--address string　默认值："0.0.0.0"

已弃用： 要监听 --port 端口的 IP 地址（将其设置为 0.0.0.0 或者 :: 用于监听所有接口和 IP 族）。 请参阅 --bind-address。 如果在 --config 中指定了一个配置文件，这个参数将被忽略。

--algorithm-provider string

已弃用： 要使用的调度算法驱动，此标志设置组件配置框架的默认插件。 可选值：ClusterAutoscalerProvider | DefaultProvider

--allow-metric-labels stringToString　默认值： []

这个键值映射表设置 度量标签 所允许设置的值。 其中键的格式是 <MetricName>,<LabelName>。 值的格式是 <allowed_value>,<allowed_value>。 例如：metric1,label1='v1,v2,v3', metric1,label2='v1,v2,v3' metric2,label1='v1,v2,v3'。

--alsologtostderr

日志记录到标准错误以及文件

--authentication-kubeconfig string

指向具有足够权限以创建 `tokenaccessreviews.authentication.k8s.io` 的 Kubernetes 核心服务器的 kubeconfig 文件。 这是可选的。如果为空，则所有令牌请求均被视为匿名请求，并且不会在集群中查找任何客户端 CA。

--authentication-skip-lookup

如果为 false，则 authentication-kubeconfig 将用于从集群中查找缺少的身份验证配置。

--authentication-token-webhook-cache-ttl duration　默认值：10s

缓存来自 Webhook 令牌身份验证器的响应的持续时间。

--authentication-tolerate-lookup-failure　默认值：true

如果为 true，则无法从集群中查找缺少的身份验证配置是致命的。 请注意，这可能导致身份验证将所有请求视为匿名。

--authorization-always-allow-paths strings　默认值："/healthz,/readyz,/livez"

在授权过程中跳过的 HTTP 路径列表，即在不联系 'core' kubernetes 服务器的情况下被授权的 HTTP 路径。

--authorization-kubeconfig string

指向具有足够权限以创建 subjectaccessreviews.authorization.k8s.io 的 Kubernetes 核心服务器的 kubeconfig 文件。这是可选的。 如果为空，则所有未被鉴权机制略过的请求都会被禁止。

--authorization-webhook-cache-authorized-ttl duration　默认值：10s

缓存来自 Webhook 授权者的 'authorized' 响应的持续时间。

--authorization-webhook-cache-unauthorized-ttl duration　默认值：10s

缓存来自 Webhook 授权者的 'unauthorized' 响应的持续时间。

--azure-container-registry-config string

包含 Azure 容器仓库配置信息的文件的路径。

--bind-address string　默认值：0.0.0.0

监听 --secure-port 端口的 IP 地址。 集群的其余部分以及 CLI/ Web 客户端必须可以访问关联的接口。 如果为空，将使用所有接口（0.0.0.0 表示使用所有 IPv4 接口，"::" 表示使用所有 IPv6 接口）。 如果为空或未指定地址 (0.0.0.0 或 ::)，所有接口将被使用。

--cert-dir string

TLS 证书所在的目录。如果提供了--tls-cert-file 和 --tls private-key-file， 则将忽略此参数。

--client-ca-file string

如果已设置，由 client-ca-file 中的证书机构签名的客户端证书的任何请求都将使用 与客户端证书的 CommonName 对应的身份进行身份验证。

--config string

配置文件的路径。以下标志会覆盖此文件中的值：
--algorithm-provider
--policy-config-file
--policy-configmap
--policy-configmap-namespace

--contention-profiling　默认值: true

已弃用: 如果启用了性能分析，则启用锁竞争分析。 如果 --config 指定了一个配置文件，那么这个参数将被忽略。

--disabled-metrics strings

这个标志提供了一个规避不良指标的选项。你必须提供完整的指标名称才能禁用它。 免责声明：禁用指标的优先级比显示隐藏的指标更高。

--experimental-logging-sanitization

[试验性功能] 当启用此标志时，标记为敏感的字段（密码、密钥、令牌）等不会被日志 输出。
运行时的日志清理操作可能引入相当程度的计算开销，因此不应在生产环境中启用。

--feature-gates <逗号分隔的 'key=True|False' 对>

一组 key=value 对，描述了 alpha/experimental 特征开关。选项包括：
A set of key=value pairs that describe feature gates for alpha/experimental features. Options are:

APIListChunking=true|false (BETA - 默认值=true)
APIPriorityAndFairness=true|false (BETA - 默认值=true)
APIResponseCompression=true|false (BETA - 默认值=true)
APIServerIdentity=true|false (ALPHA - 默认值=false)
AllAlpha=true|false (ALPHA - 默认值=false)
AllBeta=true|false (BETA - 默认值=false)
AnyVolumeDataSource=true|false (ALPHA - 默认值=false)
AppArmor=true|false (BETA - 默认值=true)
BalanceAttachedNodeVolumes=true|false (ALPHA - 默认值=false)
BoundServiceAccountTokenVolume=true|false (BETA - 默认值=true)
CPUManager=true|false (BETA - 默认值=true)
CSIInlineVolume=true|false (BETA - 默认值=true)
CSIMigration=true|false (BETA - 默认值=true)
CSIMigrationAWS=true|false (BETA - 默认值=false)
CSIMigrationAzureDisk=true|false (BETA - 默认值=false)
CSIMigrationAzureFile=true|false (BETA - 默认值=false)
CSIMigrationGCE=true|false (BETA - 默认值=false)
CSIMigrationOpenStack=true|false (BETA - 默认值=true)
CSIMigrationvSphere=true|false (BETA - 默认值=false)
CSIMigrationvSphereComplete=true|false (BETA - 默认值=false)
CSIServiceAccountToken=true|false (BETA - 默认值=true)
CSIStorageCapacity=true|false (BETA - 默认值=true)
CSIVolumeFSGroupPolicy=true|false (BETA - 默认值=true)
CSIVolumeHealth=true|false (ALPHA - 默认值=false)
ConfigurableFSGroupPolicy=true|false (BETA - 默认值=true)
ControllerManagerLeaderMigration=true|false (ALPHA - 默认值=false)
CronJobControllerV2=true|false (BETA - 默认值=true)
CustomCPUCFSQuotaPeriod=true|false (ALPHA - 默认值=false)
DaemonSetUpdateSurge=true|false (ALPHA - 默认值=false)
DefaultPodTopologySpread=true|false (BETA - 默认值=true)
DevicePlugins=true|false (BETA - 默认值=true)
DisableAcceleratorUsageMetrics=true|false (BETA - 默认值=true)
DownwardAPIHugePages=true|false (BETA - 默认值=false)
DynamicKubeletConfig=true|false (BETA - 默认值=true)
EfficientWatchResumption=true|false (BETA - 默认值=true)
EndpointSliceProxying=true|false (BETA - 默认值=true)
EndpointSliceTerminatingCondition=true|false (ALPHA - 默认值=false)
EphemeralContainers=true|false (ALPHA - 默认值=false)
ExpandCSIVolumes=true|false (BETA - 默认值=true)
ExpandInUsePersistentVolumes=true|false (BETA - 默认值=true)
ExpandPersistentVolumes=true|false (BETA - 默认值=true)
ExperimentalHostUserNamespaceDefaulting=true|false (BETA - 默认值=false)
GenericEphemeralVolume=true|false (BETA - 默认值=true)
GracefulNodeShutdown=true|false (BETA - 默认值=true)
HPAContainerMetrics=true|false (ALPHA - 默认值=false)
HPAScaleToZero=true|false (ALPHA - 默认值=false)
HugePageStorageMediumSize=true|false (BETA - 默认值=true)
IPv6DualStack=true|false (BETA - 默认值=true)
InTreePluginAWSUnregister=true|false (ALPHA - 默认值=false)
InTreePluginAzureDiskUnregister=true|false (ALPHA - 默认值=false)
InTreePluginAzureFileUnregister=true|false (ALPHA - 默认值=false)
InTreePluginGCEUnregister=true|false (ALPHA - 默认值=false)
InTreePluginOpenStackUnregister=true|false (ALPHA - 默认值=false)
InTreePluginvSphereUnregister=true|false (ALPHA - 默认值=false)
IndexedJob=true|false (ALPHA - 默认值=false)
IngressClassNamespacedParams=true|false (ALPHA - 默认值=false)
KubeletCredentialProviders=true|false (ALPHA - 默认值=false)
KubeletPodResources=true|false (BETA - 默认值=true)
KubeletPodResourcesGetAllocatable=true|false (ALPHA - 默认值=false)
LocalStorageCapacityIsolation=true|false (BETA - 默认值=true)
LocalStorageCapacityIsolationFSQuotaMonitoring=true|false (ALPHA - 默认值=false)
LogarithmicScaleDown=true|false (ALPHA - 默认值=false)
MemoryManager=true|false (ALPHA - 默认值=false)
MixedProtocolLBService=true|false (ALPHA - 默认值=false)
NamespaceDefaultLabelName=true|false (BETA - 默认值=true)
NetworkPolicyEndPort=true|false (ALPHA - 默认值=false)
NonPreemptingPriority=true|false (BETA - 默认值=true)
PodAffinityNamespaceSelector=true|false (ALPHA - 默认值=false)
PodDeletionCost=true|false (ALPHA - 默认值=false)
PodOverhead=true|false (BETA - 默认值=true)
PreferNominatedNode=true|false (ALPHA - 默认值=false)
ProbeTerminationGracePeriod=true|false (ALPHA - 默认值=false)

ProcMountType=true|false (ALPHA - 默认值=false)
QOSReserved=true|false (ALPHA - 默认值=false)
RemainingItemCount=true|false (BETA - 默认值=true)
RemoveSelfLink=true|false (BETA - 默认值=true)
RotateKubeletServerCertificate=true|false (BETA - 默认值=true)
ServerSideApply=true|false (BETA - 默认值=true)
ServiceInternalTrafficPolicy=true|false (ALPHA - 默认值=false)
ServiceLBNodePortControl=true|false (ALPHA - 默认值=false)
ServiceLoadBalancerClass=true|false (ALPHA - 默认值=false)
ServiceTopology=true|false (ALPHA - 默认值=false)
SetHostnameAsFQDN=true|false (BETA - 默认值=true)
SizeMemoryBackedVolumes=true|false (ALPHA - 默认值=false)
StorageVersionAPI=true|false (ALPHA - 默认值=false)
StorageVersionHash=true|false (BETA - 默认值=true)
SuspendJob=true|false (ALPHA - 默认值=false)
TTLAfterFinished=true|false (BETA - 默认值=true)
TopologyAwareHints=true|false (ALPHA - 默认值=false)
TopologyManager=true|false (BETA - 默认值=true)
ValidateProxyRedirects=true|false (BETA - 默认值=true)
VolumeCapacityPriority=true|false (ALPHA - 默认值=false)
WarningHeaders=true|false (BETA - 默认值=true)
WinDSR=true|false (ALPHA - 默认值=false)
WinOverlay=true|false (BETA - 默认值=true)
WindowsEndpointSliceProxying=true|false (BETA - 默认值=true)

--hard-pod-affinity-symmetric-weight int32　默认值：1

已弃用: RequiredDuringScheduling 亲和性是不对称的，但是存在与每个 RequiredDuringScheduling 关联性规则相对应的隐式 PreferredDuringScheduling 关联性规则。 `--hard-pod-affinity-symmetric-weight` 代表隐式 PreferredDuringScheduling 关联性规则的权重。权重必须在 0-100 范围内。 如果 --config 指定了一个配置文件，那么这个参数将被忽略。

-h, --help

kube-scheduler 帮助命令

--http2-max-streams-per-connection int

服务器为客户端提供的 HTTP/2 连接最大限制。零表示使用 Golang 的默认值。

--kube-api-burst int32　默认值：100

已弃用: 与 kubernetes API 通信时使用的突发请求个数限值。 如果 --config 指定了一个配置文件，那么这个参数将被忽略。

--kube-api-content-type string　默认值："application/vnd.kubernetes.protobuf"

已弃用: 发送到 API 服务器的请求的内容类型。 如果 --config 指定了一个配置文件，那么这个参数将被忽略。

--kube-api-qps float32　默认值：50

已弃用: 与 kubernetes apiserver 通信时要使用的 QPS 如果 --config 指定了一个配置文件，那么这个参数将被忽略。

--kubeconfig string

已弃用: 包含鉴权和主节点位置信息的 kubeconfig 文件的路径。 如果 --config 指定了一个配置文件，那么这个参数将被忽略。

--leader-elect　默认值：true

在执行主循环之前，开始领导者选举并选出领导者。 使用多副本来实现高可用性时，可启用此标志。

--leader-elect-lease-duration duration　默认值：15s

非领导者候选人在观察到领导者更新后将等待直到试图获得领导但未更新的领导者职位的等待时间。 这实际上是领导者在被另一位候选人替代之前可以停止的最大持续时间。 该情况仅在启用了领导者选举的情况下才适用。

--leader-elect-renew-deadline duration　默认值：10s

领导者尝试在停止领导之前更新领导职位的间隔时间。该时间必须小于或等于租赁期限。 仅在启用了领导者选举的情况下才适用。

--leader-elect-resource-lock string　默认值："leases"

在领导者选举期间用于锁定的资源对象的类型。支持的选项是 `endpoints`、`configmaps`、`leases`、`endpointleases` 和 `configmapsleases`。

--leader-elect-resource-name string　默认值："kube-scheduler"

在领导者选举期间用于锁定的资源对象的名称。

--leader-elect-resource-namespace string　默认值："kube-system"

在领导者选举期间用于锁定的资源对象的命名空间。

--leader-elect-retry-period duration　默认值：2s

客户应在尝试获取和更新领导之间等待的时间。仅在启用了领导者选举的情况下才适用。

--lock-object-name string　默认值："kube-scheduler"

已弃用: 定义锁对象的名称。将被删除以便使用 `--leader-elect-resource-name` 。 如果 --config 指定了一个配置文件，那么这个参数将被忽略。

--lock-object-namespace string　默认值："kube-system"

已弃用: 定义锁对象的命名空间。将被删除以便使用 `leader-elect-resource-namespace` 。 如果 --config 指定了一个配置文件，那么这个参数将被忽略。

--log-backtrace-at <a string in the form 'file:N'>　默认值: 0

当记录命中行文件 `file` 的第 `N` 行时输出堆栈跟踪。

--log-dir string

如果为非空，则在此目录中写入日志文件。

--log-file string

如果为非空，则使用此文件作为日志文件。

--log-file-max-size uint　默认值：1800

定义日志文件可以增长到的最大值。单位为兆字节。 如果值为 0，则最大文件大小为无限制。

--log-flush-frequency duration　默认值：5s

两次日志刷新之间的最大秒数。

--logging-format string　默认值：“text”

设置日志格式。可选格式：“json”，“text”。
采用非默认格式时，以下标识不会生效： --add-dir-header, --alsologtostderr, --log-backtrace-at, --log-dir, --log-file, --log-file-max-size, --logtostderr, --one-output, --skip-headers, --skip-log-headers, --stderrthreshold, --vmodule, --log-flush-frequency.
非默认选项目前处于 Alpha 阶段，有可能会出现变更且无事先警告。

--logtostderr 默认值：true

日志记录到标准错误输出而不是文件。

--master string

Kubernetes API 服务器的地址（覆盖 kubeconfig 中的任何值）。

--one-output

若此标志为 true，则日志仅写入其自身的严重性级别，而不会写入所有较低严重性级别。

--permit-address-sharing

如果为 true，在绑定端口时将使用 SO_REUSEADDR。 这将允许同时绑定诸如 0.0.0.0 这类通配符 IP和特定 IP， 并且它避免等待内核释放处于 TIME_WAIT 状态的套接字。 默认值：false

--permit-port-sharing

如果此标志为 true，在绑定端口时会使用 SO_REUSEPORT，从而允许不止一个 实例绑定到同一地址和端口。 默认值：false

--policy-config-file string

已弃用：包含调度器策略配置的文件。 当策略 ConfigMap 为提供时，或者 `--use-legacy-policy-config=true` 时使用此文件。 注意：当此标志与插件配置一起使用时，调度器会失败。

--policy-configmap string

已弃用: 包含调度器策略配置的 ConfigMap 对象的名称。 如果 --use-legacy-policy-config=false，则它必须在调度器初始化之前存在于 系统命名空间中。配置数据必须对应 'data' 映射中键名为 'policy.cfg' 的元素的值。 注意：如果与插件配置一起使用，调度器会失败。

--policy-configmap-namespace string 默认值："kube-system"

已弃用: 策略 ConfigMap 所在的名字空间。如果未提供或为空，则将使用 kube-system 名字空间。 注意：如果与插件配置一起使用，调度器会失败。

--port int 默认值：10251

已弃用： 在没有身份验证和鉴权的情况下不安全地为 HTTP 服务的端口。 如果为 0，则根本不提供 HTTP。请参见 --secure-port。 如果 --config 指定了一个配置文件，这个参数将被忽略。

--profiling 默认值： true

已弃用: 通过 Web 界面主机启用配置文件： `host:port/debug/pprof/` 。 如果 --config 指定了一个配置文件，这个参数将被忽略。

--requestheader-allowed-names stringSlice

客户端证书通用名称列表，允许在 `--requestheader-username-headers` 指定的头部中提供用户名。如果为空，则允许任何由 `--requestheader-client-ca-file` 中证书机构验证的客户端证书。

--requestheader-client-ca-file string

在信任 `--requestheader-username-headers` 指定的头部中的用户名之前 用于验证传入请求上的客户端证书的根证书包。 警告：通常不应假定传入请求已经完成鉴权。

--requestheader-extra-headers-prefix strings　默认值: "x-remote-extra-"

要检查请求头部前缀列表。建议使用 `X-Remote-Extra-` 。

--requestheader-group-headers strings　默认值: "x-remote-group"

用于检查组的请求头部列表。建议使用 `X-Remote-Group` 。

--requestheader-username-headers strings　默认值: "x-remote-user"

用于检查用户名的请求头部列表。 `X-Remote-User` 很常用。

--scheduler-name string　默认值："default-scheduler"

已弃用: 调度器名称，用于根据 Pod 的 “spec.schedulerName” 选择此 调度器将处理的 Pod。 如果 --config 指定了一个配置文件，那么这个参数将被忽略

--secure-port int　默认值：10259

通过身份验证和授权为 HTTPS 服务的端口。如果为 0，则根本不提供 HTTPS。

--show-hidden-metrics-for-version string

你希望显式隐藏指标的老版本号。只有较早的此版本号有意义，其它值都是不允许的。 格式为 <主版本>.<此版本>，例如：'1.16'。 此格式的目的是确保你有机会注意到是否下一个发行版本中隐藏了一些额外的指标， 而不是当某些指标在该版本之后被彻底移除时感到震惊。

--skip-headers

如果为 true，日志消息中不再写入头部前缀。

--skip-log-headers

如果为 true，则在打开日志文件时忽略其头部。

--stderrthreshold int　默认值：2

达到或超过此阈值的日志会被写入到标准错误输出。

--tls-cert-file string

包含默认的 HTTPS x509 证书的文件。（CA证书（如果有）在服务器证书之后并置）。 如果启用了 HTTPS 服务，并且未提供 `--tls-cert-file` 和 `--tls-private-key-file` ，则会为公共地址生成一个自签名证书和密钥， 并将其保存到 `--cert-dir` 指定的目录中。

--tls-cipher-suites strings

服务器的密码套件列表，以逗号分隔。如果省略，将使用默认的 Go 密码套件。 优先考虑的值： TLS_AES_128_GCM_SHA256, TLS_AES_256_GCM_SHA384, TLS_CHACHA20_POLY1305_SHA256, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305, TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256, TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305, TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256, TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_GCM_SHA384.
不安全的值： TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_ECDSA_WITH_RC4_128_SHA, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_RSA_WITH_RC4_128_SHA, TLS_RSA_WITH_AES_128_CBC_SHA256, TLS_RSA_WITH_RC4_128_SHA.

--tls-min-version string

支持的最低 TLS 版本。可能的值：VersionTLS10, VersionTLS11, VersionTLS12, VersionTLS13

--tls-private-key-file string

包含与 --tls-cert-file 匹配的默认 x509 私钥的文件。

--tls-sni-cert-key string

一对 x509 证书和私钥文件路径，也可以包含由全限定域名构成的域名模式列表作为后缀，并可能带有前缀的通配符段。域名匹配还允许是 IP 地址， 但是只有当 apiserver 对客户端请求的 IP 地址可见时，才能使用 IP。 如果未提供域名匹配模式，则提取证书名称。 非通配符匹配优先于通配符匹配，显式域名匹配优先于提取而来的名称。 若有多个密钥/证书对，可多次使用 `--tls-sni-cert-key` 。 例子: "example.crt,example.key" 或者 "foo.crt,foo.key:*.foo.com,foo.com"。

--use-legacy-policy-config

已弃用：设置为 true 时，调度程序将忽略策略 ConfigMap 并使用策略配置文件。 注意：当此标志与插件配置一起使用时，调度器会失败。

-v, --v int

设置日志级别详细程度的数字

--version version[=true]

打印版本信息并退出。

--vmodule <逗号分隔的 ‘模式=N’ 配置列表>

以逗号分隔的 ‘模式=N’ 设置列表，用于文件过滤的日志记录。

--write-config-to string

如果已设置，将配置值写入此文件并退出。

7 - Kubelet 认证/鉴权

概述

kubelet 的 HTTPS 端点公开了 API， 这些 API 可以访问敏感度不同的数据， 并允许你在节点上和容器内以不同级别的权限执行操作。

本文档介绍了如何对 kubelet 的 HTTPS 端点的访问进行认证和鉴权。

Kubelet 身份认证

默认情况下，未被已配置的其他身份认证方法拒绝的对 kubelet 的 HTTPS 端点的请求会被视为匿名请求， 并被赋予 `system:anonymous` 用户名和 `system:unauthenticated` 组。

要禁用匿名访问并向未经身份认证的请求发送 `401 Unauthorized` 响应，请执行以下操作：

- 带 `--anonymous-auth=false` 标志启动 kubelet

要对 kubelet 的 HTTPS 端点启用 X509 客户端证书认证：

- 带 `--client-ca-file` 标志启动 kubelet，提供一个 CA 证书包以供验证客户端证书
- 带 `--kubelet-client-certificate` 和 `--kubelet-client-key` 标志启动 apiserver
- 有关更多详细信息，请参见 apiserver 身份验证文档

要启用 API 持有者令牌（包括服务帐户令牌）以对 kubelet 的 HTTPS 端点进行身份验证，请执行以下操作：

- 确保在 API 服务器中启用了 `authentication.k8s.io/v1beta1` API 组
- 带 `--authentication-token-webhook` 和 `--kubeconfig` 标志启动 kubelet
- kubelet 调用已配置的 API 服务器上的 `TokenReview` API，以根据持有者令牌确定用户信息

Kubelet 鉴权

任何成功通过身份验证的请求（包括匿名请求）之后都会被鉴权。 默认的鉴权模式为 `AlwaysAllow`，它允许所有请求。

细分对 kubelet API 的访问权限可能有多种原因：

- 启用了匿名身份验证，但是应限制匿名用户调用 kubelet API 的能力
- 启用了持有者令牌认证，但应限制任意 API 用户（如服务帐户）调用 kubelet API 的能力
- 启用了客户端证书身份验证，但仅应允许已配置的 CA 签名的某些客户端证书使用 kubelet API

要细分对 kubelet API 的访问权限，请将鉴权委派给 API 服务器：

- 确保在 API 服务器中启用了 `authorization.k8s.io/v1beta1` API 组
- 带 `--authorization-mode=Webhook` 和 `--kubeconfig` 标志启动 kubelet
- kubelet 调用已配置的 API 服务器上的 `SubjectAccessReview` API， 以确定每个请求是否得到鉴权

kubelet 使用与 apiserver 相同的 请求属性 方法对 API 请求执行鉴权。

请求的动词根据传入请求的 HTTP 动词确定：

HTTP 动词	请求动词
POST	create
GET, HEAD	get

HTTP 动词	请求动词
PUT	update
PATCH	patch
DELETE	delete

资源和子资源是根据传入请求的路径确定的：

Kubelet API	资源	子资源
/stats/*	nodes	stats
/metrics/*	nodes	metrics
/logs/*	nodes	log
/spec/*	nodes	spec
其它所有	nodes	proxy

名字空间和 API 组属性始终是空字符串， 资源名称始终是 kubelet 的 `Node` API 对象的名称。

在此模式下运行时，请确保传递给 apiserver 的由 `--kubelet-client-certificate` 和 `--kubelet-client-key` 标志标识的用户具有以下属性的鉴权：

- verb=*, resource=nodes, subresource=proxy
- verb=*, resource=nodes, subresource=stats
- verb=*, resource=nodes, subresource=log
- verb=*, resource=nodes, subresource=spec
- verb=*, resource=nodes, subresource=metrics

8 - TLS 启动引导

在一个 Kubernetes 集群中，工作节点上的组件（kubelet 和 kube-proxy）需要与 Kubernetes 主控组件通信，尤其是 kube-apiserver。 为了确保通信本身是私密的、不被干扰，并且确保集群的每个组件都在与另一个 可信的组件通信，我们强烈建议使用节点上的客户端 TLS 证书。

启动引导这些组件的正常过程，尤其是需要证书来与 kube-apiserver 安全通信的 工作节点，可能会是一个具有挑战性的过程，因为这一过程通常不受 Kubernetes 控制， 需要不少额外工作。 这也使得初始化或者扩缩一个集群的操作变得具有挑战性。

为了简化这一过程，从 1.4 版本开始，Kubernetes 引入了一个证书请求和签名 API 以便简化此过程。该提案可在 这里看到。

本文档描述节点初始化的过程，如何为 kubelet 配置 TLS 客户端证书启动引导， 以及其背后的工作原理。

初始化过程

当工作节点启动时，kubelet 执行以下操作：

1. 寻找自己的 `kubeconfig` 文件
2. 检索 API 服务器的 URL 和凭据，通常是来自 `kubeconfig` 文件中的 TLS 密钥和已签名证书
3. 尝试使用这些凭据来与 API 服务器通信

假定 kube-apiserver 成功地认证了 kubelet 的凭据数据，它会将 kubelet 视为 一个合法的节点并开始将 Pods 分派给该节点。

注意，签名的过程依赖于：

- `kubeconfig` 中包含密钥和本地主机的证书
- 证书被 kube-apiserver 所信任的一个证书机构（CA）所签名

负责部署和管理集群的人有以下责任：

1. 创建 CA 密钥和证书
2. 将 CA 证书发布到 kube-apiserver 运行所在的主控节点上
3. 为每个 kubelet 创建密钥和证书；强烈建议为每个 kubelet 使用独一无二的、 CN 取值与众不同的密钥和证书
4. 使用 CA 密钥对 kubelet 证书签名
5. 将 kubelet 密钥和签名的证书发布到 kubelet 运行所在的特定节点上

本文中描述的 TLS 启动引导过程有意简化甚至完全自动化上述过程，尤其是 第三步之后的操作，因为这些步骤是初始化或者扩缩集群时最常见的操作。

启动引导初始化

在启动引导初始化过程中，会发生以下事情：

1. kubelet 启动
2. kubelet 看到自己 *没有* 对应的 `kubeconfig` 文件
3. kubelet 搜索并发现 `bootstrap-kubeconfig` 文件
4. kubelet 读取该启动引导文件，从中获得 API 服务器的 URL 和用途有限的 一个“令牌（Token）”
5. kubelet 建立与 API 服务器的连接，使用上述令牌执行身份认证
6. kubelet 现在拥有受限制的凭据来创建和取回证书签名请求（CSR）
7. kubelet 为自己创建一个 CSR，并将其 signerName 设置为 `kubernetes.io/kube-apiserver-client-kubelet`
8. CSR 被以如下两种方式之一批复：

- 如果配置了，kube-controller-manager 会自动批复该 CSR
- 如果配置了，一个外部进程，或者是人，使用 Kubernetes API 或者使用 `kubectl` 来批复该 CSR

9. kubelet 所需要的证书被创建
10. 证书被发放给 kubelet
11. kubelet 取回该证书
12. kubelet 创建一个合适的 `kubeconfig`，其中包含密钥和已签名的证书
13. kubelet 开始正常操作
14. 可选地，如果配置了，kubelet 在证书接近于过期时自动请求更新证书
15. 更新的证书被批复并发放；取决于配置，这一过程可能是自动的或者手动完成

本文的其余部分描述配置 TLS 启动引导的必要步骤及其局限性。

配置

要配置 TLS 启动引导及可选的自动批复，你必须配置以下组件的选项：

- kube-apiserver
- kube-controller-manager
- kubelet
- 集群内的资源：`ClusterRoleBinding` 以及可能需要的 `ClusterRole`

此外，你需要有 Kubernetes 证书机构（Certificate Authority，CA）。

证书机构

就像在没有启动引导的情况下，你会需要证书机构（CA）密钥和证书。 这些数据会被用来对 kubelet 证书进行签名。 如前所述，将证书机构密钥和证书发布到主控节点是你的责任。

就本文而言，我们假定这些数据被发布到主控节点上的 `/var/lib/kubernetes/ca.pem`（证书）和 `/var/lib/kubernetes/ca-key.pem`（密钥）文件中。 我们将这两个文件称作“Kubernetes CA 证书和密钥”。 所有 Kubernetes 组件（kubelet、kube-apiserver、kube-controller-manager）都使用这些凭据，并假定这里的密钥和证书都是 PEM 编码的。

kube-apiserver 配置

启用 TLS 启动引导对 kube-apiserver 有若干需求：

- 能够识别对客户端证书进行签名的 CA
- 能够对启动引导的 kubelet 执行身份认证，并将其置入 `system:bootstrappers` 组
- 能够对启动引导的 kubelet 执行鉴权操作，允许其创建证书签名请求（CSR）

识别客户证书

对于所有客户端证书的认证操作而言，这是很常见的。 如果还没有设置，要为 kube-apiserver 命令添加 `--client-ca-file=FILENAME` 标志来启用客户端证书认证，在标志中引用一个包含用来签名的证书的证书机构包， 例如：`--client-ca-file=/var/lib/kubernetes/ca.pem`。

初始启动引导认证

为了让启动引导的 kubelet 能够连接到 kube-apiserver 并请求证书， 它必须首先在服务器上认证自身身份。你可以使用任何一种能够对 kubelet 执行身份认证的 身份认证组件。

尽管所有身份认证策略都可以用来对 kubelet 的初始启动凭据来执行认证， 出于容易准备的因素，建议使用如下两个身份认证组件：

1. 启动引导令牌（Bootstrap Token）
2. 令牌认证文件

启动引导令牌是一种对 kubelet 进行身份认证的方法，相对简单且容易管理， 且不需要在启动 kube-apiserver 时设置额外的标志。 启动引导令牌从 Kubernetes 1.12 开始是一种 **Beta** 功能特性。

无论选择哪种方法，这里的需求是 kubelet 能够被身份认证为某个具有如下权限的用户：

1. 创建和读取 CSR
2. 在启用了自动批复时，能够在请求节点客户端证书时得到自动批复

使用启动引导令牌执行身份认证的 kubelet 会被认证为 `system:bootstrappers` 组中的用户。这是使用启动引导令牌的一种标准方法。

随着这个功能特性的逐渐成熟，你需要确保令牌绑定到某基于角色的的访问控制（RBAC） 策略上，从而严格限制请求（使用启动引导令牌） 仅限于客户端申请提供证书。当 RBAC 被配置启用时，可以将令牌限制到某个组，从而 提高灵活性。例如，你可以在准备节点期间禁止某特定启动引导组的访问。

启动引导令牌

启动引导令牌的细节在这里 详述。启动引导令牌在 Kubernetes 集群中存储为 Secret 对象，被发放给各个 kubelet。 你可以在整个集群中使用同一个令牌，也可以为每个节点发放单独的令牌。

这一过程有两个方面：

1. 基于令牌 ID、机密数据和范畴信息创建 Kubernetes Secret
2. 将令牌发放给 kubelet

从 kubelet 的角度，所有令牌看起来都很像，没有特别的含义。 从 kube-apiserver 服务器的角度，启动引导令牌是很特殊的。 根据其 `type`、 `namespace` 和 `name`，kube-apiserver 能够将认作特殊的令牌， 并授予携带该令牌的任何人特殊的启动引导权限，换言之，将其视为 `system:bootstrappers` 组的成员。这就满足了 TLS 启动引导的基本需求。

关于创建 Secret 的进一步细节可访问这里。

如果你希望使用启动引导令牌，你必须在 kube-apiserver 上使用下面的标志启用之：

```
--enable-bootstrap-token-auth=true
```

令牌认证文件

kube-apiserver 能够将令牌视作身份认证依据。 这些令牌可以是任意数据，但必须表示为基于某安全的随机数生成器而得到的 至少 128 位混沌数据。这里的随机数生成器可以是现代 Linux 系统上的 `/dev/urandom`。生成令牌的方式有很多种。例如：

```
head -c 16 /dev/urandom | od -An -t x | tr -d ' '
```

上面的命令会生成类似于 `02b50b05283e98dd0fd71db496ef01e8` 这样的令牌。

令牌文件看起来是下面的例子这样，其中前面三个值可以是任何值，用引号括起来 的组名称则只能用例子中给的值。

```
02b50b05283e98dd0fd71db496ef01e8,kubelet-bootstrap,10001,"system:bootstrappers"
```

向 kube-apiserver 添加 `--token-auth-file=FILENAME` 标志（或许这要对 systemd 的单元文件作修改）以启用令牌文件。参见 这里 的文档以了解进一步的细节。

授权 kubelet 创建 CSR

现在启动引导节点被身份认证为 `system:bootstrapping` 组的成员，它需要被 *授权* 创建证书签名请求（CSR）并在证书被签名之后将其取回。 幸运的是，Kubernetes 提供了一个 `ClusterRole`，其中精确地封装了这些许可， `system:node-bootstrapper`。

为了实现这一点，你只需要创建 `ClusterRoleBinding`，将 `system:bootstrappers` 组绑定到集群角色 `system:node-bootstrapper`。

```yaml
# 允许启动引导节点创建 CSR
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: create-csrs-for-bootstrapping
subjects:
- kind: Group
  name: system:bootstrappers
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: system:node-bootstrapper
  apiGroup: rbac.authorization.k8s.io
```

kube-controller-manager 配置

API 服务器从 kubelet 收到证书请求并对这些请求执行身份认证，但真正负责发放 签名证书的是控制器管理器。

控制器管理器通过一个证书发放的控制回路来执行此操作。该操作的执行方式是使用磁盘上 的文件用 cfssl 本地签名组件 来完成。目前，所发放的所有证书都有一年的有效期，并设定了默认的一组密钥用法。

为了让控制器管理器对证书签名，它需要：

- 能够访问你之前所创建并分发的“Kubernetes CA 密钥和证书”
- 启用 CSR 签名

访问密钥和证书

如前所述，你需要创建一个 Kubernetes CA 密钥和证书，并将其发布到主控节点。 这些数据会被控制器管理器来对 kubelet 证书进行签名。

由于这些被签名的证书反过来会被 kubelet 用来在 kube-apiserver 执行普通的 kubelet 身份认证，很重要的一点是为控制器管理器所提供的 CA 也被 kube-apiserver 信任用来执行身份认证。CA 密钥和证书是通过 kube-apiserver 的标志 `--client-ca-file=FILENAME`（例如，`--client-ca-file=/var/lib/kubernetes/ca.pem`）， 来设定的，正如 kube-apiserver 配置节所述。

要将 Kubernetes CA 密钥和证书提供给 kube-controller-manager，可使用以下标志：

```
--cluster-signing-cert-file="/etc/path/to/kubernetes/ca/ca.crt" --cluster-signing-key-fi
```

例如：

```
--cluster-signing-cert-file="/var/lib/kubernetes/ca.pem" --cluster-signing-key-file="/va
```

所签名的证书的合法期限可以通过下面的标志来配置：

```
--cluster-signing-duration
```

批复

为了对 CSR 进行批复，你需要告诉控制器管理器批复这些 CSR 是可接受的。 这是通过将 RBAC 访问权限授予正确的组来实现的。

许可权限有两组：

- `nodeclient`：如果节点在为节点创建新的证书，则该节点还没有证书。该节点 使用前文所列的令牌之一来执行身份认证，因此是组 `system:bootstrappers` 组 的成员。
- `selfnodeclient`：如果节点在对证书执行续期操作，则该节点已经拥有一个证书。 节点持续使用现有的证书将自己认证为 `system:nodes` 组的成员。

要允许 kubelet 请求并接收新的证书，可以创建一个 `ClusterRoleBinding` 将启动 引导节点所处的组 `system:bootstrappers` 绑定到为其赋予访问权限的 `ClusterRole` `system:certificates.k8s.io:certificatesigningrequests:nodeclient`：

```yaml
# 批复 "system:bootstrappers" 组的所有 CSR
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: auto-approve-csrs-for-group
subjects:
- kind: Group
  name: system:bootstrappers
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: system:certificates.k8s.io:certificatesigningrequests:nodeclient
  apiGroup: rbac.authorization.k8s.io
```

要允许 kubelet 对其客户端证书执行续期操作，可以创建一个 `ClusterRoleBinding` 将正常工作的节点所处的组 `system:nodes` 绑定到为其授予访问许可的 `ClusterRole` `system:certificates.k8s.io:certificatesigningrequests:selfnodeclient`：

```yaml
# 批复 "system:nodes" 组的 CSR 续约请求
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: auto-approve-renewals-for-nodes
subjects:
- kind: Group
  name: system:nodes
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: system:certificates.k8s.io:certificatesigningrequests:selfnodeclient
  apiGroup: rbac.authorization.k8s.io
```

作为 kube-controller-manager 的一部分的 `csrapproving` 控制器是自动被启用的。 该控制器使用 `SubjectAccessReview` API 来确定是否某给定用户被授权请求 CSR，之后基于鉴权结果执行批复操作。 为了避免与其它批复组件发生冲突，内置的批复组件不会显式地拒绝任何 CSRs。 该组件仅是忽略未被授权的请求。 控制器也会作为垃圾收集的一部分清除已过期的证书。

kubelet 配置

最后，当主控节点被正确配置并且所有必要的身份认证和鉴权机制都就绪时， 我们可以配置 kubelet。

kubelet 需要以下配置来执行启动引导：

- 一个用来存储所生成的密钥和证书的路径（可选，可以使用默认配置）
- 一个用来指向尚不存在的 `kubeconfig` 文件的路径；kubelet 会将启动引导 配置文件放到这个位置

- 一个指向启动引导 `kubeconfig` 文件的路径，用来提供 API 服务器的 URL 和启动引导凭据，例如，启动引导令牌
- 可选的：轮换证书的指令

启动引导 `kubeconfig` 文件应该放在一个 kubelet 可访问的路径下，例如 `/var/lib/kubelet/bootstrap-kubeconfig`。

其格式与普通的 `kubeconfig` 文件完全相同。实例文件可能看起来像这样：

```
apiVersion: v1
kind: Config
clusters:
- cluster:
    certificate-authority: /var/lib/kubernetes/ca.pem
    server: https://my.server.example.com:6443
  name: bootstrap
contexts:
- context:
    cluster: bootstrap
    user: kubelet-bootstrap
  name: bootstrap
current-context: bootstrap
preferences: {}
users:
- name: kubelet-bootstrap
  user:
    token: 07401b.f395accd246ae52d
```

需要额外注意的一些因素有：

- `certificate-authority`：指向 CA 文件的路径，用来对 kube-apiserver 所出示 的服务器证书进行验证
- `server`： 用来访问 kube-apiserver 的 URL
- `token`：要使用的令牌

令牌的格式并不重要，只要它与 kube-apiserver 的期望匹配即可。 在上面的例子中，我们使用的是启动引导令牌。 如前所述，任何合法的身份认证方法都可以使用，不限于令牌。

因为启动引导 `kubeconfig` 文件是一个标准的 `kubeconfig` 文件，你可以使用 `kubectl` 来生成该文件。要生成上面的示例文件：

```
kubectl config --kubeconfig=/var/lib/kubelet/bootstrap-kubeconfig set-cluster bootstrap -
kubectl config --kubeconfig=/var/lib/kubelet/bootstrap-kubeconfig set-credentials kubelet
kubectl config --kubeconfig=/var/lib/kubelet/bootstrap-kubeconfig set-context bootstrap -
kubectl config --kubeconfig=/var/lib/kubelet/bootstrap-kubeconfig use-context bootstrap
```

要指示 kubelet 使用启动引导 `kubeconfig` 文件，可以使用下面的 kubelet 标志：

```
--bootstrap-kubeconfig="/var/lib/kubelet/bootstrap-kubeconfig" --kubeconfig="/var/lib/kub
```

在启动 kubelet 时，如果 `--kubeconfig` 标志所指定的文件并不存在，会使用通过标志 `--bootstrap-kubeconfig` 所指定的启动引导 kubeconfig 配置来向 API 服务器请求 客户端证书。在证书请求被批复并被 kubelet 收回时，一个引用所生成的密钥和所获得 证书的 kubeconfig 文件会被写入到通过 `--kubeconfig` 所指定的文件路径下。 证书和密钥文件会被放到 `--cert-dir` 所指定的目录中。

客户和服务证书

前文所述的内容都与 kubelet *客户端* 证书相关，尤其是 kubelet 用来向 kube-apiserver 认证自身身份的证书。

kubelet 也可以使用 *服务（Serving）* 证书。kubelet 自身向外提供一个 HTTPS 末端，包含若干功能特性。要保证这些末端的安全性，kubelet 可以执行以下操作 之一：

- 使用通过 `--tls-private-key-file` 和 `--tls-cert-file` 所设置的密钥和证书
- 如果没有提供密钥和证书，则创建自签名的密钥和证书
- 通过 CSR API 从集群服务器请求服务证书

TLS 启动引导所提供的客户端证书默认被签名为仅用于 `client auth`（客户端认证）， 因此不能作为提供服务的证书，或者 `server auth`。

不过，你可以启用服务器证书，至少可以部分地通过证书轮换来实现这点。

证书轮换

Kubernetes v1.8 和更高版本的 kubelet 实现了对客户端证书与/或服务证书进行轮换 这一 Beta 特性。这一特性通过 kubelet 对应的 `RotateKubeletClientCertificate` 和 `RotateKubeletServerCertificate` 特性门控标志来控制，并且是默认启用的。

`RotateKubeletClientCertificate` 会导致 kubelet 在其现有凭据即将过期时通过 创建新的 CSR 来轮换其客户端证书。要启用此功能特性，可将下面的标志传递给 kubelet：

```
--rotate-certificates
```

`RotateKubeletServerCertificate` 会让 kubelet 在启动引导其客户端凭据之后请求 一个服务证书 **且** 对该服务证书执行轮换操作。要启用此功能特性，将下面的标志 传递给 kubelet：

```
--rotate-server-certificates
```

说明：
Kubernetes 核心中所实现的 CSR 批复控制器出于 安全原因 并不会自动批复节点的 *服务* 证书。 要使用 `RotateKubeletServerCertificate` 功能特性，集群运维人员需要运行一个 定制的控制器或者手动批复服务证书的请求。

对 kubelet 服务证书的批复过程因集群部署而异，通常应该仅批复如下 CSR：

1. 由节点发出的请求（确保 `spec.username` 字段形式为 `system:node:<nodeName>` 且 `spec.groups` 包含 `system:nodes`）
2. 请求中包含服务证书用法（确保 `spec.usages` 中包含 `server auth`，可选地也可 包含 `digital signature` 和 `key encipherment`，且不包含其它用法）
3. 仅包含隶属于请求节点的 IP 和 DNS 的 `subjectAltNames`，没有 URI 和 Email 形式的 `subjectAltNames`（解析 `spec.request` 中的 x509 证书签名请求可以 检查 `subjectAltNames`）

其它身份认证组件

本文所描述的所有 TLS 启动引导内容都与 kubelet 相关。不过，其它组件也可能需要 直接与 kube-apiserver 直接通信。容易想到的是 kube-proxy，同样隶属于 Kubernetes 的控制面并且运行在所有节点之上，不过也可能包含一些其它负责 监控或者联网的组件。

与 kubelet 类似，这些其它组件也需要一种向 kube-apiserver 认证身份的方法。 你可以用几种方法来生成这类凭据：

- 较老的方式：和 kubelet 在 TLS 启动引导之前所做的一样，用类似的方式 创建和分发证书
- DaemonSet：由于 kubelet 自身被加载到所有节点之上，并且有足够能力来启动基本服务，你可以运行将 kube-proxy 和其它特定节点的服务作为 `kube-system` 名字空间中的 DaemonSet 来执行，而不是独立的进程。由于 DaemonSet 位于集群内部，你可以为其 指派一个合适的服务账户，使之具有适当的访问权限来完成其使命。这也许是配置此类 服务的最简单的方法。

kubectl 批复

CSRs 可以在控制器管理其内置的批复工作流之外被批复。

签名控制器并不会立即对所有证书请求执行签名操作。相反，它会等待这些请求被某 具有适当特权的用户标记为 “Approved（已批准）”状态。 这一流程有意允许由外部批复控制器来自动执行的批复，或者由控制器管理器内置的 批复控制器来自动批复。 不过，集群管理员也可以使用 `kubectl` 来手动批准证书请求。 管理员可以通过 `kubectl get csr` 来列举所有的 CSR，使用 `kubectl descsribe csr <name>` 来描述某个 CSR 的细节。 管理员可以使用 `kubectl certificate approve <name` 来批准某 CSR，或者 `kubectl certificate deny <name>` 来拒绝某 CSR。