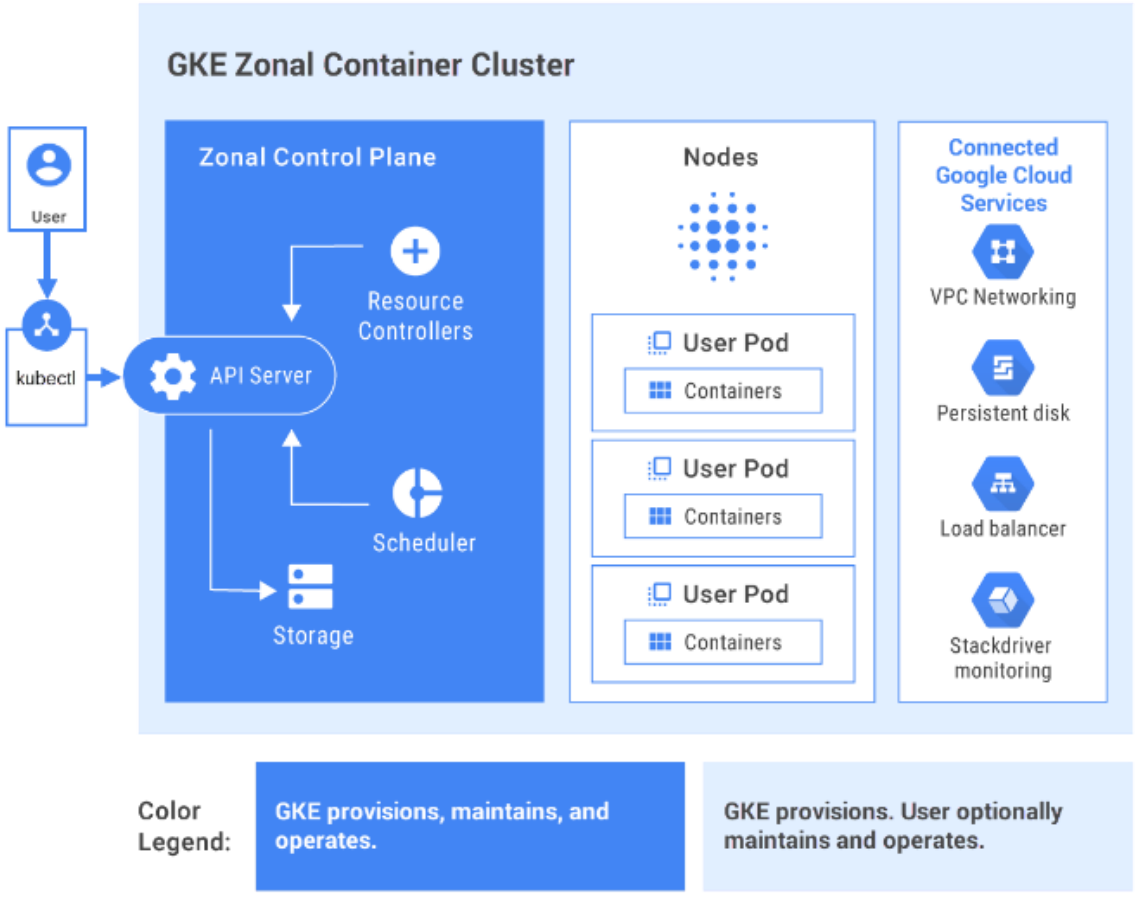
Cluster architecture

Clustering means that multiple servers are grouped together to achieve the same service. It can be regarded as a computer, a **cloud computing** platform, or through A software system centralizes the use of distributed deployment resources.

A *cluster* is the foundation of Google Kubernetes Engine (GKE): the Kubernetes objects that represent your containerized applications all run on top of a cluster.

In GKE, a cluster consists of at least one *control plane* and multiple worker machines called *nodes*. These control plane and node machines run the Kubernetes cluster orchestration system.

The following diagram provides an overview of the architecture for a zonal cluster in GKE:



Control plane

The *control plane* runs the control plane processes, including the Kubernetes API server, scheduler, and core resource controllers. The lifecycle of the control plane is managed by GKE when you create or delete a cluster. This includes upgrades to the Kubernetes version running on the control plane, which GKE performs automatically, or manually at your request if you prefer to upgrade earlier than the automatic schedule.

Control plane and the Kubernetes API

The control plane is the unified endpoint for your cluster. All interactions with the cluster are done via Kubernetes API calls, and the control plane runs the **Kubernetes API Server** process to handle those requests. You can make Kubernetes API calls directly via HTTP/gRPC, or indirectly, by running commands from the Kubernetes command-line client (`kubectl`) or interacting with the UI in the Cloud Console.

The API server process is the hub for all communication for the cluster. All internal cluster processes (such as the cluster nodes, system and components, application controllers) all act as clients of the API server; the API server is the single "source of truth" for the entire cluster.

Control plane and node interaction

The control plane is responsible for deciding what runs on all of the cluster's nodes. This can include scheduling workloads, like containerized applications, and managing the workloads' lifecycle, scaling, and upgrades. The control plane also manages network and storage resources for those workloads.

The control plane and nodes also communicate using Kubernetes APIs.

Control plane interactions with the gcr.io container registry

When you create or update a cluster, container images for the Kubernetes software running on the control plane (and nodes) are pulled from the gcr.io container registry. An outage affecting the gcr.io registry may cause the following types of failures:

- Creating new clusters will fail during the outage.
- Upgrading clusters will fail during the outage.
- Disruptions to workloads may occur even without user intervention, depending on the specific nature and duration of the outage.

In the event of a zonal or regional outage of the gcr.io container registry, Google may redirect requests to a zone or region not affected by the outage.

To check the current status of Google Cloud services, go to the Google Cloud status dashboard.

Nodes

A cluster typically has one or more *nodes*, which are the worker machines that run your containerized applications and other workloads. The individual machines are Compute Engine VM instances that GKE creates on your behalf when you create a cluster.

Each node is managed from the control plane, which receives updates on each node's self-reported status. You can exercise some manual control over node lifecycle, or you can have GKE perform automatic repairs and automatic upgrades on your cluster's nodes.

A node runs the services necessary to support the Docker containers that make up your cluster's workloads. These include the Docker runtime and the Kubernetes node agent (`kubelet`) which communicates with the control plane and is responsible for starting and running Docker containers scheduled on that node.

In GKE, there are also a number of special containers that run as per-node agents to provide functionality such as log collection and intra-cluster network connectivity.

Node machine type

Each node is of a standard Compute Engine machine type. The default type is `e2-medium`. You can select a different machine type when you create a cluster.

Node OS images

Each node runs a specialized OS image for running your containers. You can specify which OS image your clusters and node pools use.

Minimum CPU platform

When you create a cluster or node pool, you can specify a baseline minimum CPU platform for its nodes. Choosing a specific CPU platform can be advantageous for

advanced or compute-intensive workloads. For more information, refer to Minimum CPU Platform.

Node allocatable resources

Some of a node's resources are required to run the GKE and Kubernetes node components necessary to make that node function as part of your cluster. As such, you may notice a disparity between your node's *total resources* (as specified in the machine type documentation) and the node's *allocatable resources* in GKE.

As larger machine types tend to run more containers (and by extension, more Pods), the amount of resources that GKE reserves for Kubernetes components scales upward for larger machines. Windows Server nodes also require more resources than a typical Linux node. The nodes need the extra resources to account for running the Windows OS and for the Windows Server components that can't run in containers.

You can make a request for resources for your Pods or limit their resource usage. To learn how to request or limit resource usage for Pods, refer to Managing Resources for Containers.

To inspect the node allocatable resources available in a cluster, run the following command:

```
kubectl describe node node-name | grep Allocatable -B 7 -A 6
```

where **node-name** is the name of the node to inspect.

The returned output contains `Capacity` and `Allocatable` fields with measurements for ephemeral storage, memory, and CPU.

Eviction threshold

To determine how much memory is available for Pods, you must also consider the eviction threshold. GKE reserves an additional 100 MiB of memory on each node for kubelet eviction.

Allocatable memory and CPU resources

Allocatable resources are calculated in the following way:

```
Allocatable = Capacity - Reserved - Eviction Threshold
```

For memory resources, GKE reserves the following:

- 255 MiB of memory for machines with less than 1 GB of memory
- 25% of the first 4GB of memory
- 20% of the next 4GB of memory (up to 8GB)
- 10% of the next 8GB of memory (up to 16GB)
- 6% of the next 112GB of memory (up to 128GB)
- 2% of any memory above 128GB

Note: Prior to 1.12.0, machines with less than 1GB of memory are exempt from memory reservations.

For CPU resources, GKE reserves the following:

- 6% of the first core
- 1% of the next core (up to 2 cores)
- 0.5% of the next 2 cores (up to 4 cores)
- 0.25% of any cores above 4 cores

The following table shows the amount of allocatable memory and CPU resources that are available for scheduling your cluster's Linux workloads for each standard node machine type.

Note: GKE reserves approximately 1.5 times more resources on Windows Server nodes, so the allocatable values are lower than the Linux values listed in the table.

Machine type	Memory capacity (GB)	Allocatable memory (GB)	CPU capacity (cores)	Allocatable CPU (cores)
`c2-standard-4`	16	13.3	4	3.92
`c2-standard-8`	32	28.6	8	7.91
`c2-standard-16`	64	58.7	16	15.89

c2-standard-30	120	111.2	30	29.89
c2-standard-60	240	228.4	60	59.85
e2-micro	1	0.62	2	0.94[1]
e2-small	2	1.35	2	0.94[1]
e2-medium (default)	4	2.76	2	0.94[1]
e2-standard-2	8	6.1	2	1.93
e2-standard-4	16	13.3	4	3.92
e2-standard-8	32	28.6	8	7.91
e2-standard-16	64	58.7	16	15.89
e2-highmem-2	16	13.3	2	1.93

e2-highmem-4	32	28.6	4	3.92
e2-highmem-8	64	58.7	8	7.91
e2-highmem-16	128	118.6	16	15.89
e2-highcpu-2	2	1.5	2	1.93
e2-highcpu-4	4	2.9	4	3.92
e2-highcpu-8	8	6.1	8	7.91
e2-highcpu-16	16	13.3	16	15.89
g1-small	1.7	1.2	1	0.94
m1-megamem-96	1433.6	1414.7	96	95.69

m1-ultramem-40	961	942.1	40	39.85
m1-ultramem-80	1922	1903.1	80	79.77
m1-ultramem-160	3844	3825.1	160	159.69
m1-ultramem-208	5888	5869.1	208	207.69
m1-ultramem-416	11776	11757.1	416	415.69
n1-standard-1	3.75	2.7	1	0.94
n1-standard-2	7.5	5.7	2	1.93
n1-standard-4	15	12.3	4	3.92
n1-standard-8	30	26.6	8	7.91

n1-standard-16	60	54.7	16	15.89
n1-standard-32	120	111.2	32	31.85
n1-standard-64	240	228.4	64	63.77
n1-standard-96	360	346.4	96	95.69
n1-highmem-2	13	10.7	2	1.93
n1-highmem-4	26	22.8	4	3.92
n1-highmem-8	52	47.2	8	7.91
n1-highmem-16	104	96.0	16	15.89
n1-highmem-32	208	197.4	32	31.85

n1-highmem-64	416	400.8	64	63.77
n1-highmem-96	624	605.1	96	95.69
n1-highcpu-2	1.8	1.3	2	1.93
n1-highcpu-4	3.6	2.6	4	3.92
n1-highcpu-8	7.2	5.5	8	7.91
n1-highcpu-16	14.4	11.9	16	15.89
n1-highcpu-32	28.8	25.3	32	31.85
n1-highcpu-64	57.6	52.5	64	63.77
n1-highcpu-96	86.4	79.6	96	95.69

n2-standard-2	8	6.1	2	1.93
n2-standard-4	16	13.3	4	3.92
n2-standard-8	32	28.6	8	7.91
n2-standard-16	64	58.7	16	15.89
n2-standard-32	128	118.6	32	31.85
n2-standard-48	192	182.6	48	47.85
n2-standard-64	256	244.4	64	63.77
n2-standard-80	320	308.4	80	79.77
n2-highmem-2	16	13.3	2	1.93

n2-highmem-4	32	28.6	4	3.92
n2-highmem-8	64	58.7	8	7.91
n2-highmem-16	128	118.6	16	15.89
n2-highmem-32	256	244.4	32	31.85
n2-highmem-48	384	370.4	48	47.85
n2-highmem-64	512	496.8	64	63.77
n2-highmem-80	640	621.1	80	79.77
n2-highcpu-2	2	1.5	2	1.93
n2-highcpu-4	4	2.9	4	3.92

n2-highcpu-8	8	6.1	8	7.91
n2-highcpu-16	16	13.3	16	15.89
n2-highcpu-32	32	28.6	32	31.85
n2-highcpu-48	48	44.6	48	47.85
n2-highcpu-64	64	58.7	64	63.77
n2-highcpu-80	80	74.7	80	79.77
n2d-standard-2	8	6.1	2	1.93
n2d-standard-4	16	13.3	4	3.92
n2d-standard-8	32	28.6	8	7.91

n2d-standard-16	64	58.7	16	15.89
n2d-standard-32	128	118.6	32	31.85
n2d-standard-48	192	182.6	48	47.85
n2d-standard-64	256	244.4	64	63.77
n2d-standard-80	320	308.4	80	79.77
n2d-standard-96	384	370.4	96	95.69
n2d-standard-128	512	496.8	128	127.69
n2d-standard-224	896	877.1	224	223.69
n2d-highmem-2	16	13.3	2	1.93

n2d-highmem-4	32	28.6	4	3.92
n2d-highmem-8	64	58.7	8	7.91
n2d-highmem-16	128	118.6	16	15.89
n2d-highmem-32	256	244.4	32	31.85
n2d-highmem-48	384	370.4	48	47.85
n2d-highmem-64	512	496.8	64	63.77
n2d-highmem-80	640	621.1	80	79.77
n2d-highmem-96	780	761.1	96	95.69
n2d-highcpu-2	2	1.5	2	1.93

n2d-highcpu-4	4	2.9	4	3.92
n2d-highcpu-8	8	6.1	8	7.91
n2d-highcpu-16	16	13.3	16	15.89
n2d-highcpu-32	32	28.6	32	31.85
n2d-highcpu-48	48	44.6	48	47.85
n2d-highcpu-64	64	58.7	64	63.77
n2d-highcpu-80	80	74.7	80	79.77
n2d-highcpu-96	96	89.2	96	95.69
n2d-highcpu-128	128	118.6	128	127.69

n2d-highcpu-224	224	213.4	224	223.69

1GKE has decided to reduce the allocatable CPU resources available to schedule user workloads (known as the node allocatable resources) on e2-micro, e2-small, and e2-medium machine types. For more details, see the GKE release notes.

Note: The values listed for allocatable resources do not account for the resources used by `kube-system` pods, the amount of which varies with each Kubernetes release. These system pods generally occupy an additional 0.4 vCPU and 400 MiB memory on each node (values are approximate). It is recommended that you directly inspect your cluster if you require an exact accounting of usable resources on each node.**Note:** `f1-micro` machines are not supported, because they do not have sufficient memory to run GKE.

Allocatable local ephemeral storage resources

Beta

This feature is covered by the Pre-GA Offerings Terms of the Google Cloud Platform Terms of Service. Pre-GA features may have limited support, and changes to pre-GA features may not be compatible with other pre-GA versions. For more information, see the launch stage descriptions.

Beginning in GKE version 1.10, you can manage your local ephemeral storage resources as you do your CPU and memory resources. To learn how to make your Pods specify ephemeral storage requests and limits and to see how they are acted on, see Local ephemeral storage in the Kubernetes documentation.

GKE typically configures its nodes with a single file system and periodic scanning. A portion of the boot disk that depends on its size is reserved for system use.

Ephemeral storage can use up to the capacity of the boot disk, except for a portion which is reserved for system use and an eviction threshold. The total reserved size depends on the boot disk size according to the following formula:

```
10% * BOOT-DISK-CAPACITY + Min(50% * BOOT-DISK-CAPACITY, 6K + 35% * BOOT-DISK-CAPACITY, 100G)
```

For an approximate representation of the amount of allocatable ephemeral storage available as boot disk capacity increases, see the following graph:

Rate and review