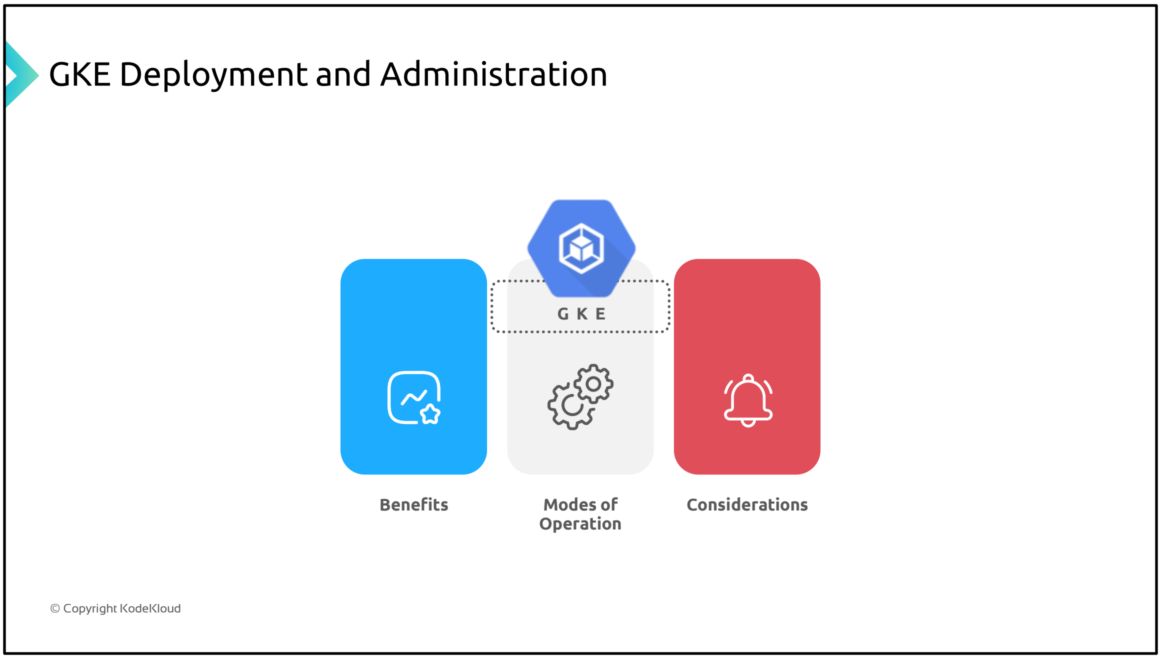
GKE Deployment and Administration



GKE Deployment and Administration

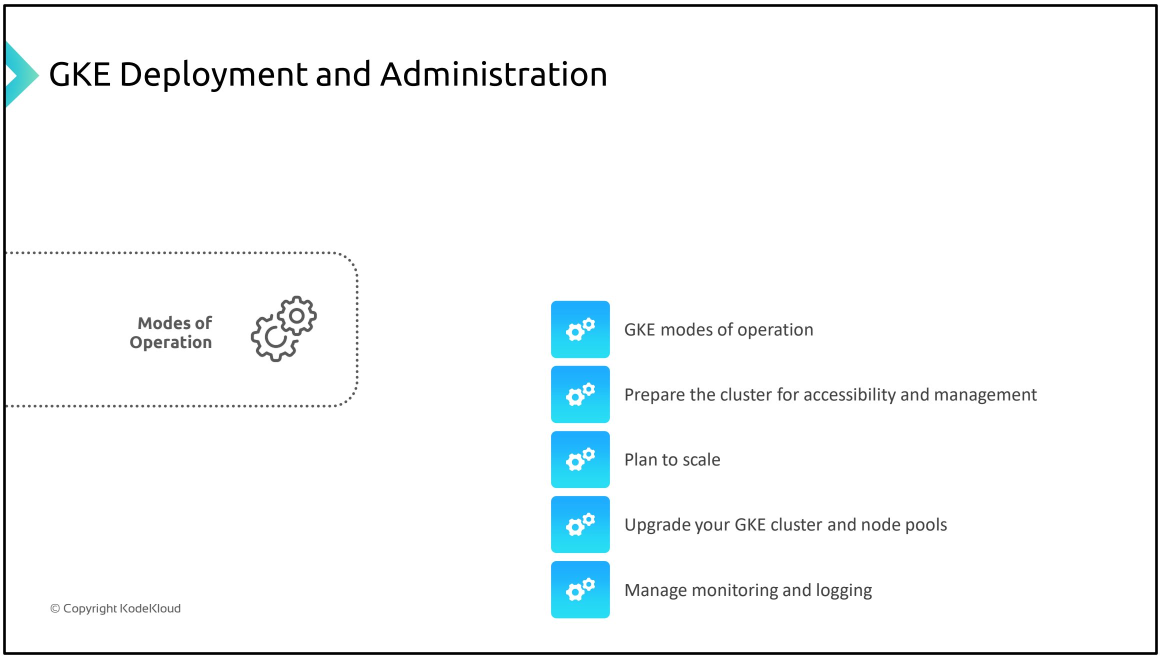
GKE

Benefits

Modes of Operation

Considerations

In this section, we will explore the different modes of operation that are available for a GKE cluster. We'll also discuss the benefits of deploying a GKE cluster in one or the other modes. We'll then talk about some considerations that we should be taking into account while choosing between the options that are available for deploying a GKE cluster, followed by a few scenarios discussing the suitability of the different GKE modes.



GKE Deployment and Administration

Modes of Operation

- GKE modes of operation
- Prepare the cluster for accessibility and management
- Plan to scale
- Upgrade your GKE cluster and node pools
- Manage monitoring and logging

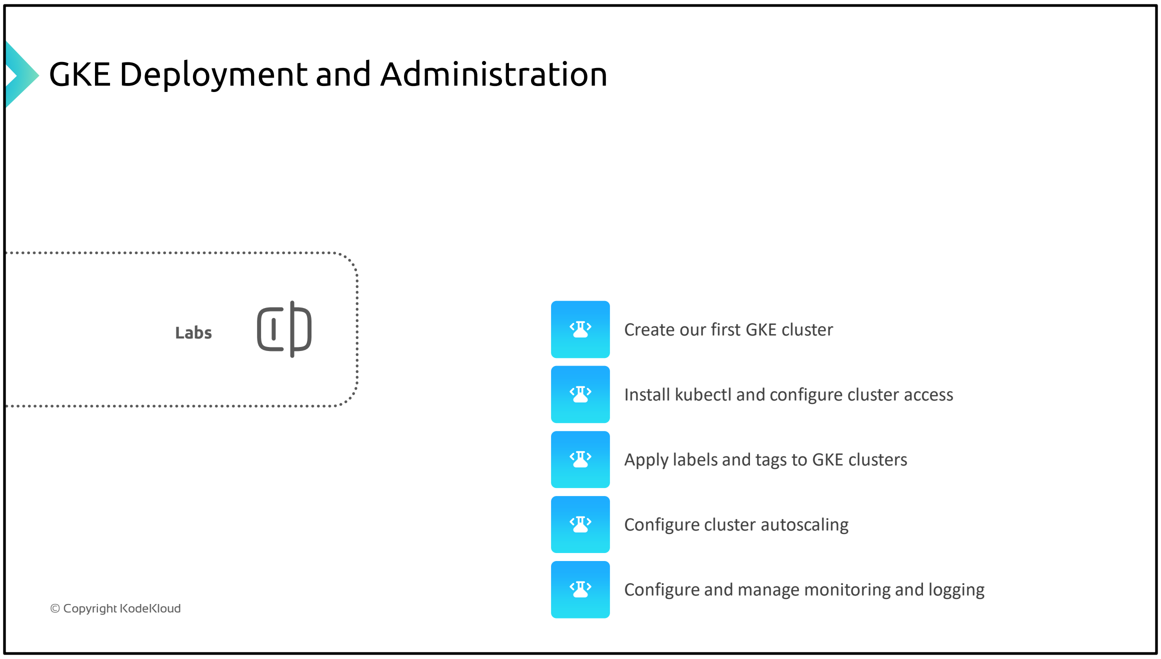
GKE modes of operation:

Preparing the cluster for accessibility and management

Planning to scale: Cluster autoscaling and node auto-provisioning

Upgrading your GKE cluster and node pools

Monitoring and Logging: Cloud Operations for GKE



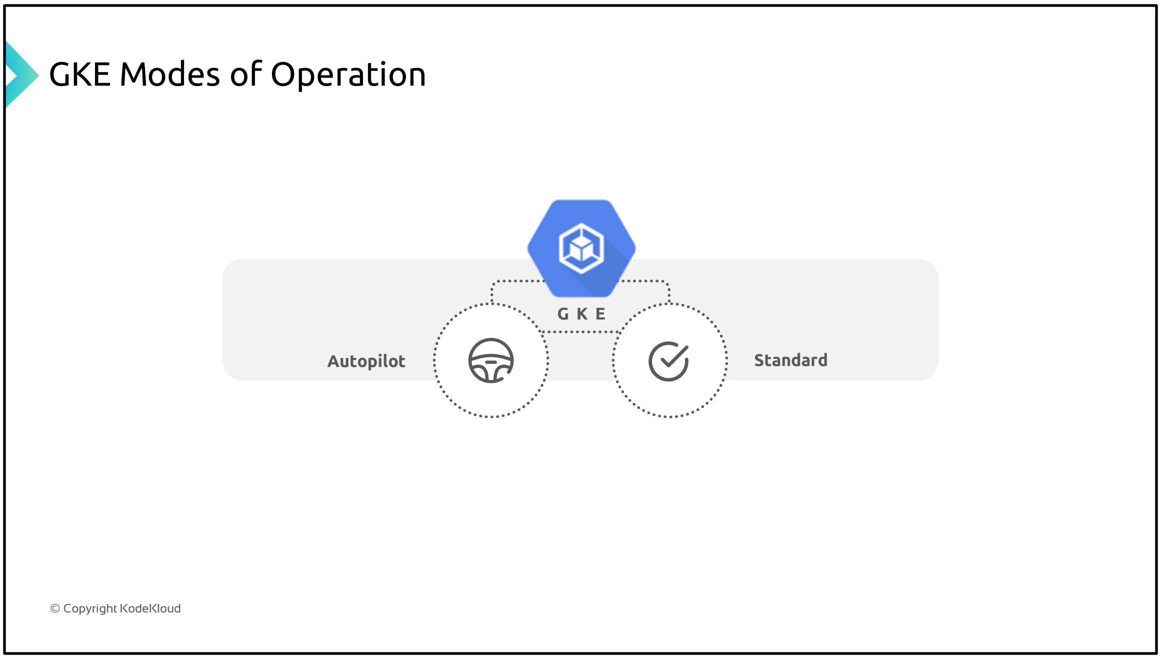
GKE Deployment and Administration

Labs

- Create our first GKE cluster
- Install kubectl and configure cluster access
- Apply labels and tags to GKE clusters
- Configure cluster autoscaling
- Configure and manage monitoring and logging

Lab: Creating our first GKE cluster
Lab: Installing kubectl and configuring cluster access
Lab: Applying labels and tags to GKE clusters
Lab: Configure cluster autoscaling Lab: Manually upgrade a GKE cluster or a node pool
Lab: Configuring and managing monitoring and logging for GKE

GKE Modes of Operation



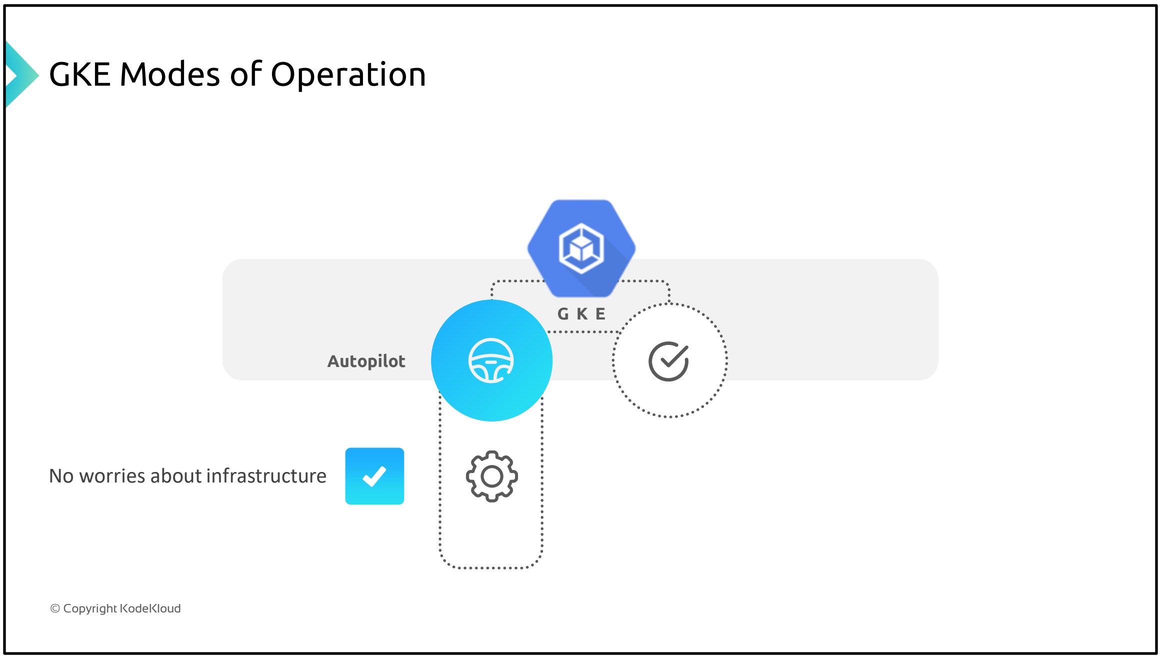
GKE Modes of Operation

GKE

Autopilot

Standard

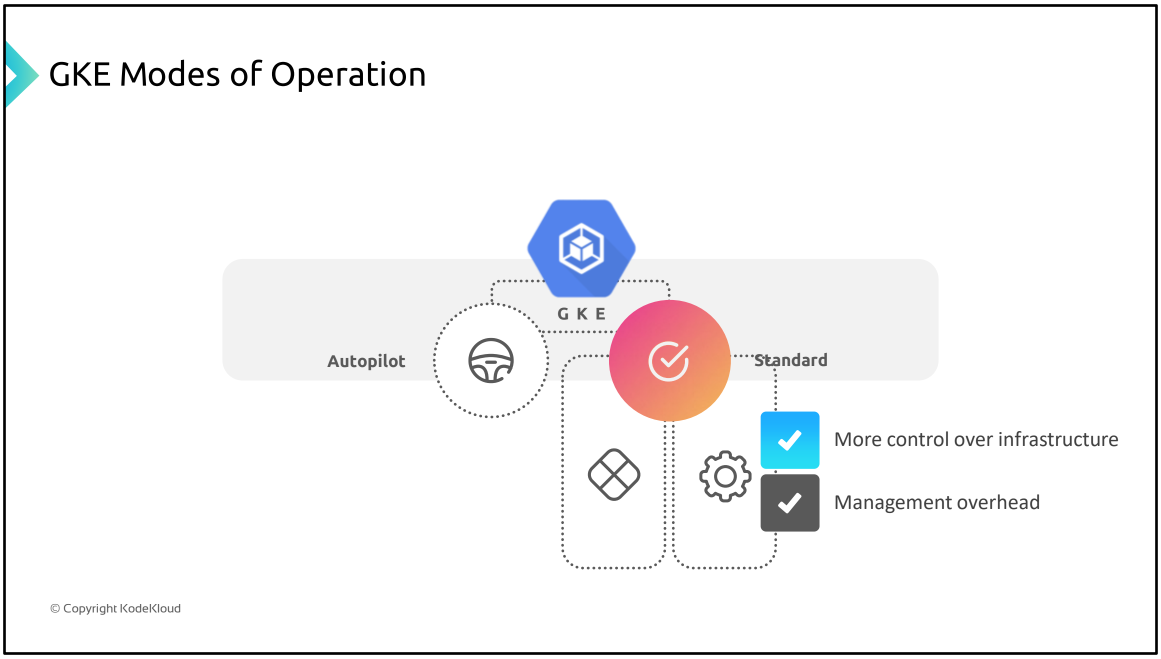
Google Kubernetes Engine (GKE) offers two modes of operation: Autopilot and Standard.



GKE Modes of Operation

No worries about infrastructure ☑

Autopilot is a fully managed mode that allows its users to focus on the applications without worrying about the underlying infrastructure.



GKE Modes of Operation

GKE

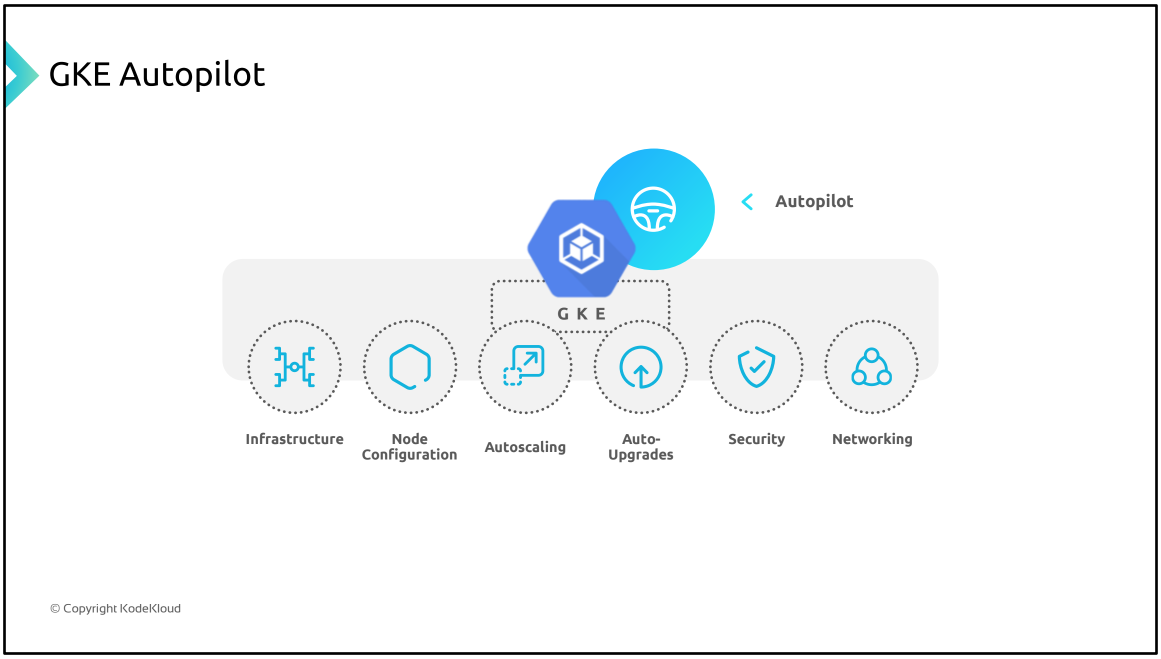
Autopilot

Standard

✓ More control over infrastructure

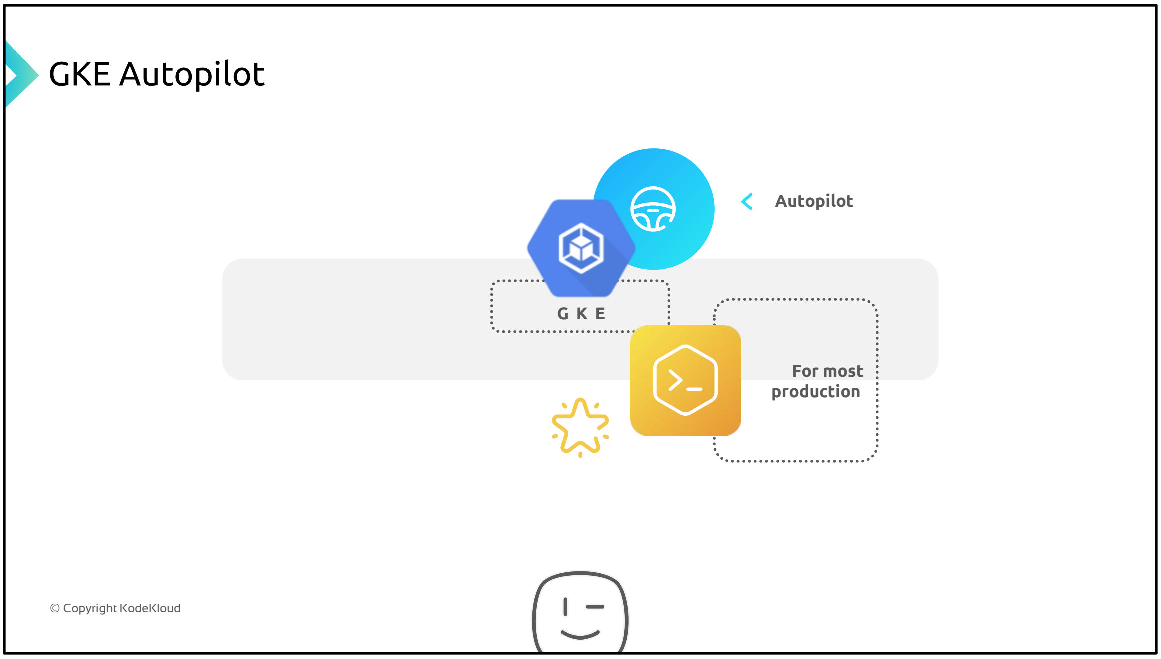
✓ Management overhead

Whereas Standard mode gives **more control over the infrastructure**, but it also **involves some management overhead**. Understanding the features and benefits of these modes helps in making an informed decision on which one of the two operation modes suits the requirements of a particular workload.

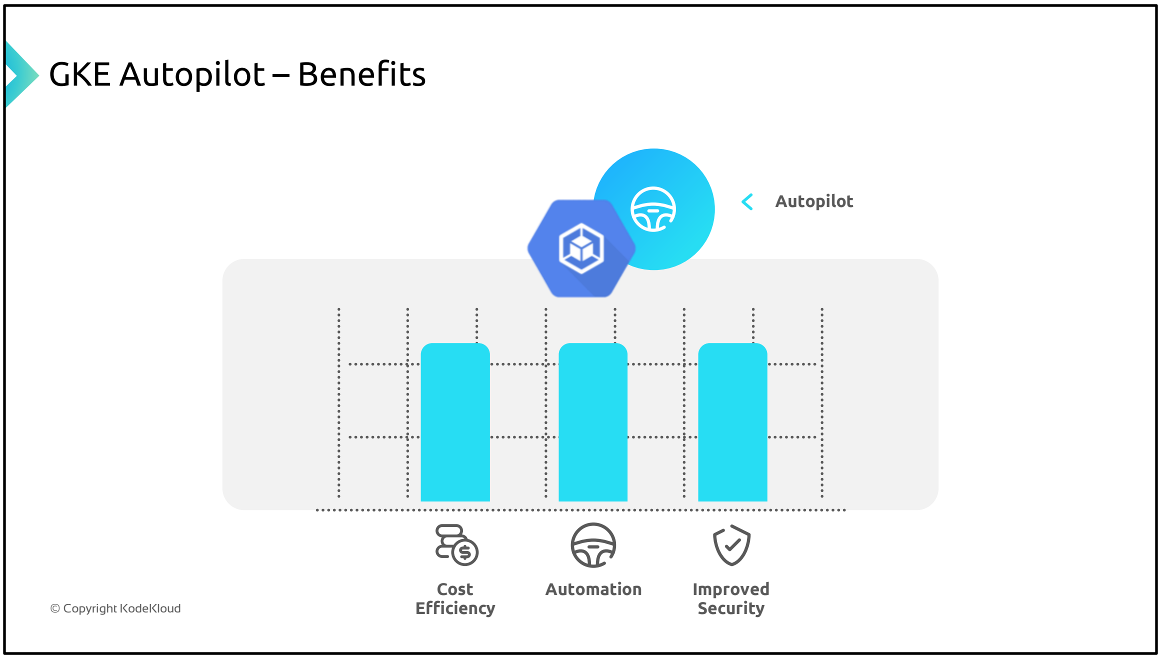


GKE Autopilot

With GKE Autopilot, Google manages the underlying infrastructure, including node configuration, autoscaling, auto-upgrades, baseline security configurations, and baseline networking configuration. The user simply deploys applications to GKE, and Google takes care of the rest.


GKE Autopilot

GKE Autopilot mode is also the recommended option in GKE for most production workloads unless there is a specific requirement to manage or configure the underlying infrastructure.

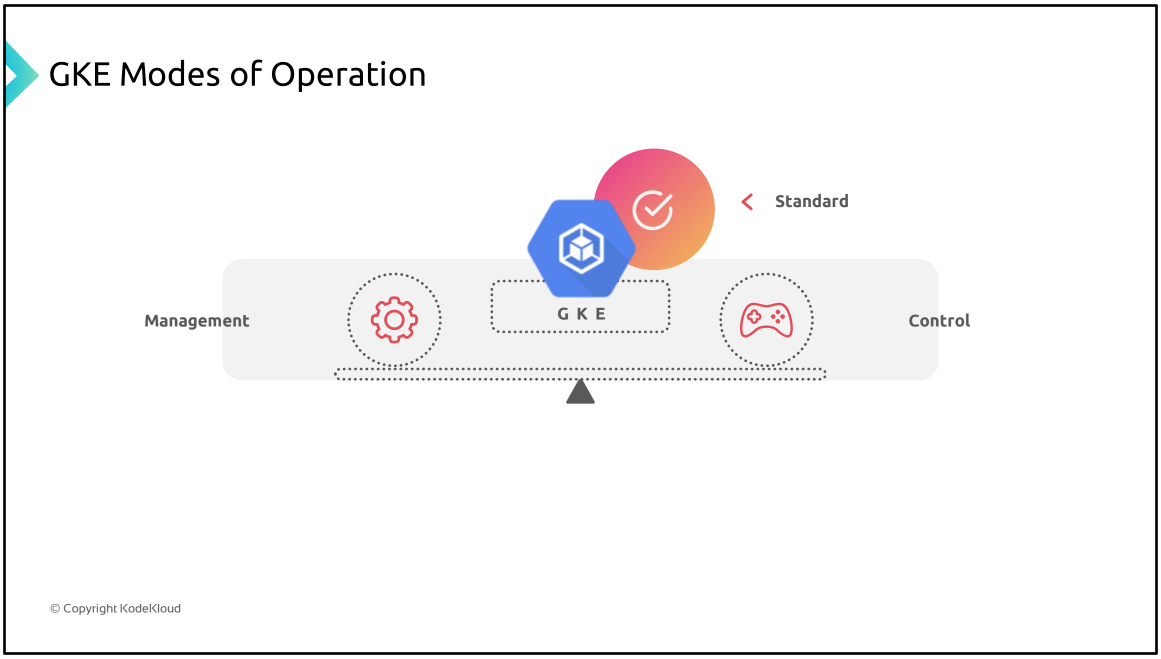


GKE Autopilot – Benefits

There are some key out-of-the-box features that we get with GKE Autopilot mode.

Cost efficiency: With Autopilot, a user is only charged for the compute resources that the workloads actually use while running. The fact that we don’t have to manage any of the node configurations while deploying a GKE cluster in autopilot mode, as Google manages all that aspect for us, automates this process of providing us with the required resources for running the workload. This means there is no need to worry about paying

for unused capacity on nodes, system pods, operating system costs, or unscheduled workloads.
Automation: Google manages the nodes or worker machines as they are called. This includes the creation of new nodes for applications workloads and configuring automatic upgrades and repairs for the existing nodes in a cluster. It also handles scaling of nodes and workloads based on traffic, ensuring optimal performance without manual intervention.
Improved security posture and reliability: Autopilot mode implements many Kubernetes best practices and security settings by default. Google also automatically applies security patches to the cluster nodes when they become available, ensuring the cluster stays protected with the latest security updates.



GKE Modes of Operation

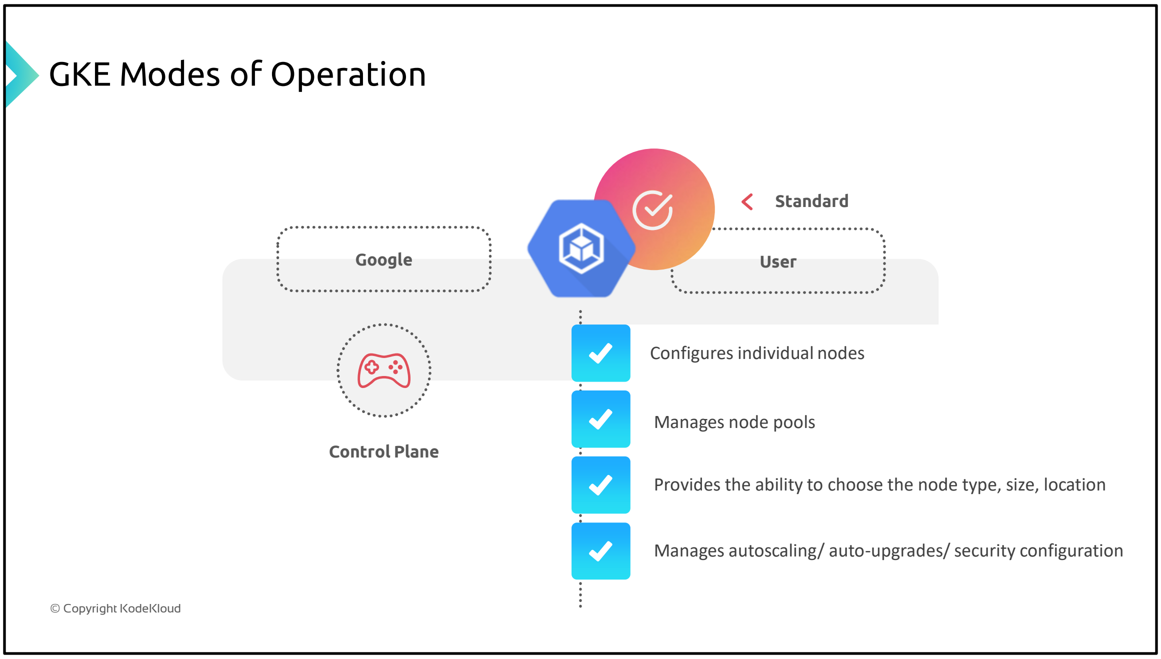
Standard

Management

GKE

Control

GKE Standard mode, on the other hand, adopts a hybrid approach maintaining a fine balance between the management of underlying infrastructure components and, at the same time, giving its users the capability to control how worker machines are configured, giving more control over the underlying infrastructure.



GKE Modes of Operation

Standard

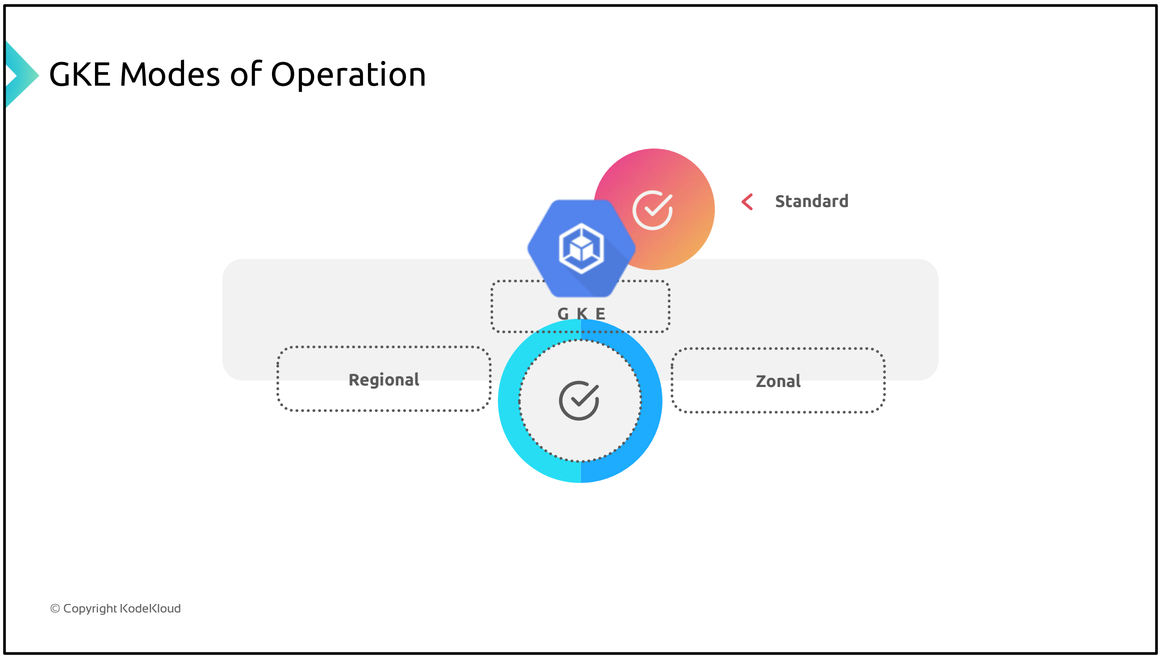
Google

User

Control Plane

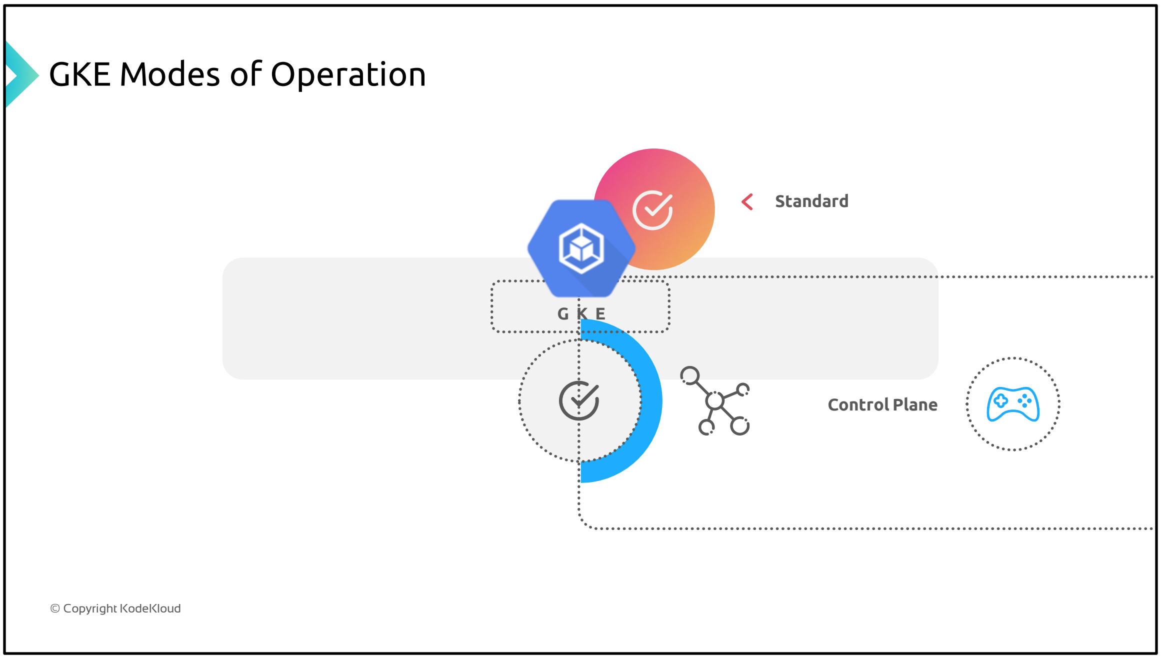
- ✓ Configures individual nodes
- ✓ Manages node pools
- ✓ Provides the ability to choose the node type, size, location
- ✓ Manages autoscaling/ auto-upgrades/ security configuration

While Google still manages the control plane, users are responsible for configuring the individual nodes and managing groups of nodes called node pools. A GKE cluster in Standard mode provides that ability to choose the node type, size, and location of the worker machines. The user can also manage the autoscaling, auto-upgrades, and security configuration of the cluster.



GKE Modes of Operation

GKE Standard mode supports two types of clusters: zonal and regional.



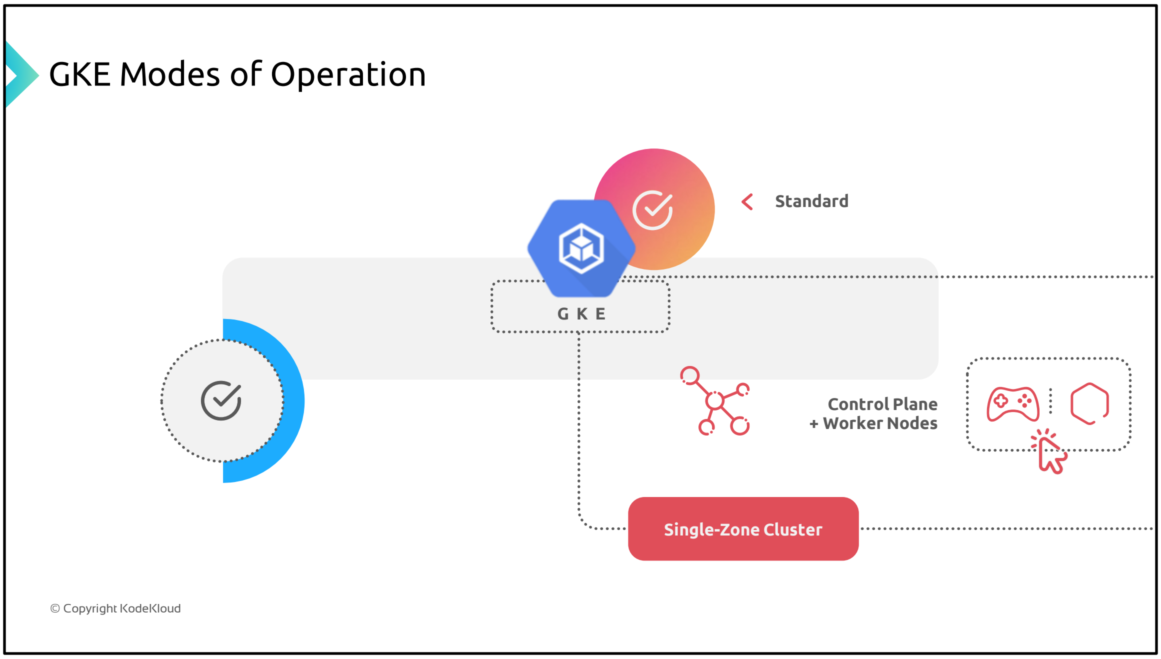
GKE Modes of Operation

Standard

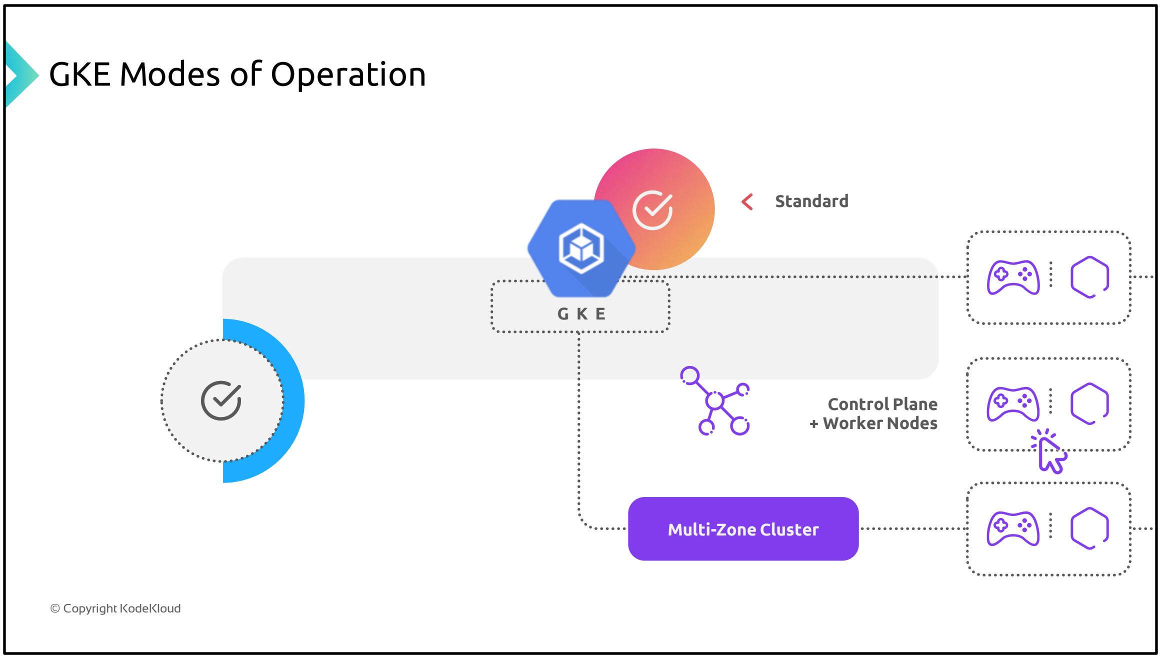
GKE

Control Plane

Zonal clusters are deployed in a single zone, which means it has only one control plane running in a single zone. So, if that particular zone has an outage, cluster resources will also become unavailable. However, the worker nodes for the zonal cluster can be distributed in a single zone or multiple zones.

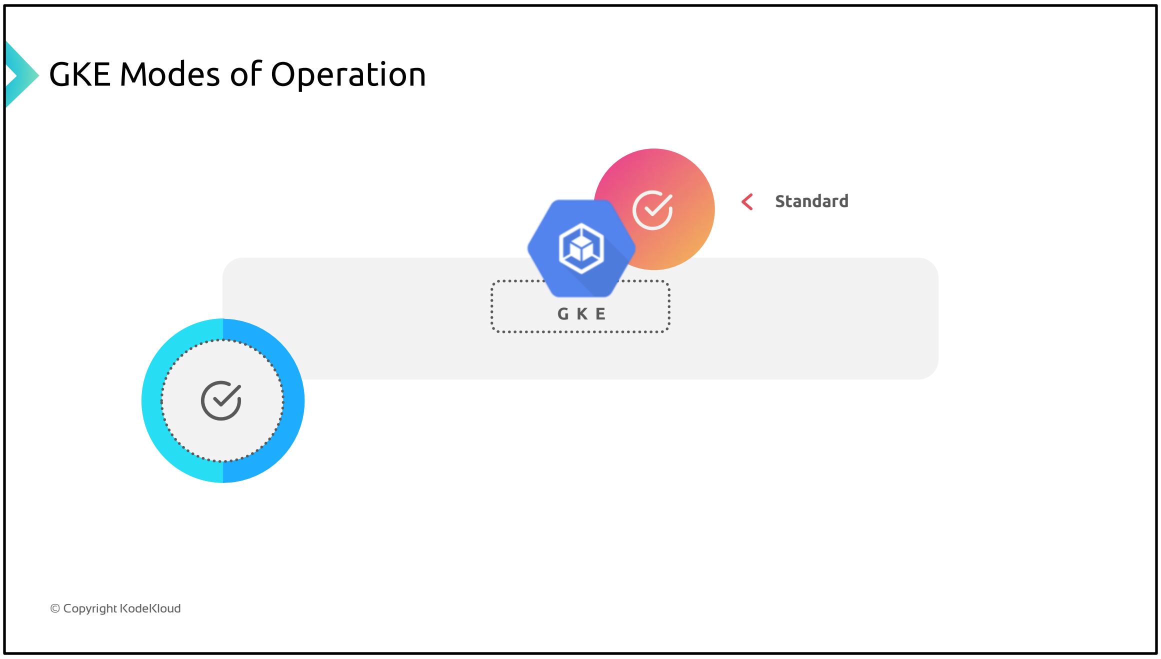


So, in a **single-zone cluster**, there will **only be one control plane running** in a zone managing the worker nodes in the same zone.



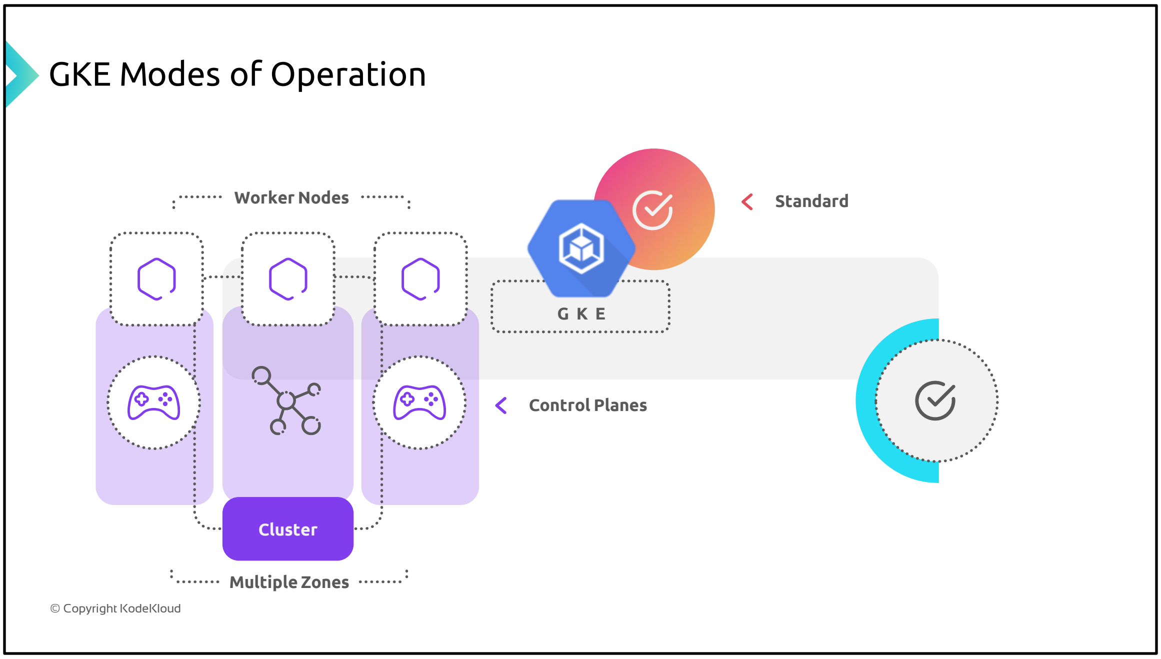
GKE Modes of Operation

In a **multi-zonal cluster**, there is still **a single replica of the control plane running in a single zone managing worker nodes running in multiple zones**. A multi-zonal cluster is resilient to zonal failures. During an upgrade of the cluster or an outage of the zone where the control plane runs, workloads still run. However, the cluster, its nodes, and its workloads cannot be configured until the control plane is available.



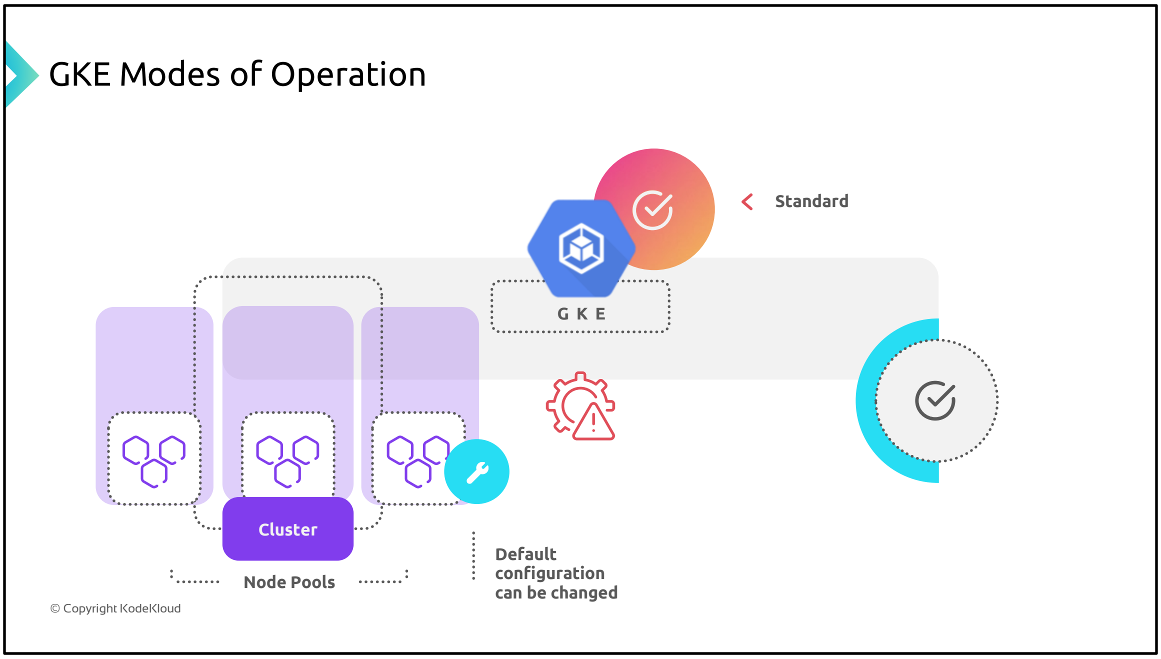
GKE Modes of Operation

Moving to regional clusters…



GKE Modes of Operation

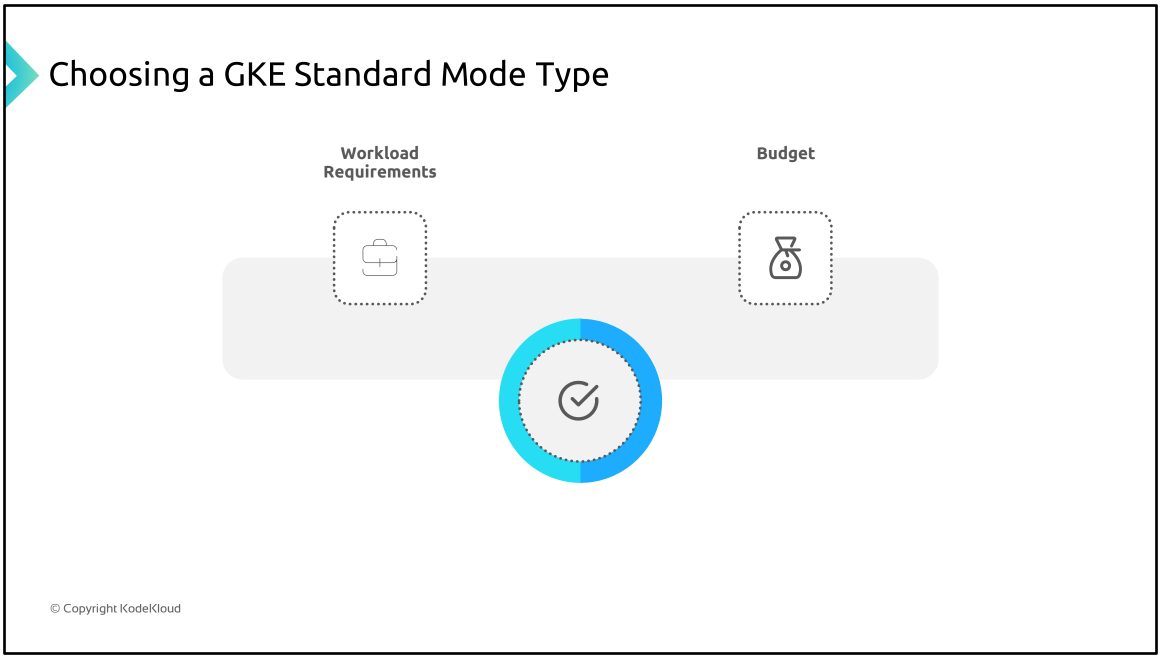
Regional clusters are deployed across multiple zones in a region with multiple replicas of the control plane. Similar to Zonal clusters, worker nodes in a regional cluster can run in multiple zones or a single zone depending on the configured node locations.



GKE Modes of Operation

By default, a regional GKE cluster replicates each node pool across three zones of the control plane's region. When the cluster is created or when a new node pool is added to an existing cluster, this default configuration can be changed by specifying the zone(s) in which the cluster's nodes run.

An important point to **Note here is that**: All the zones specified for the configuration of nodes or node pools must be within the same region as the control plane.

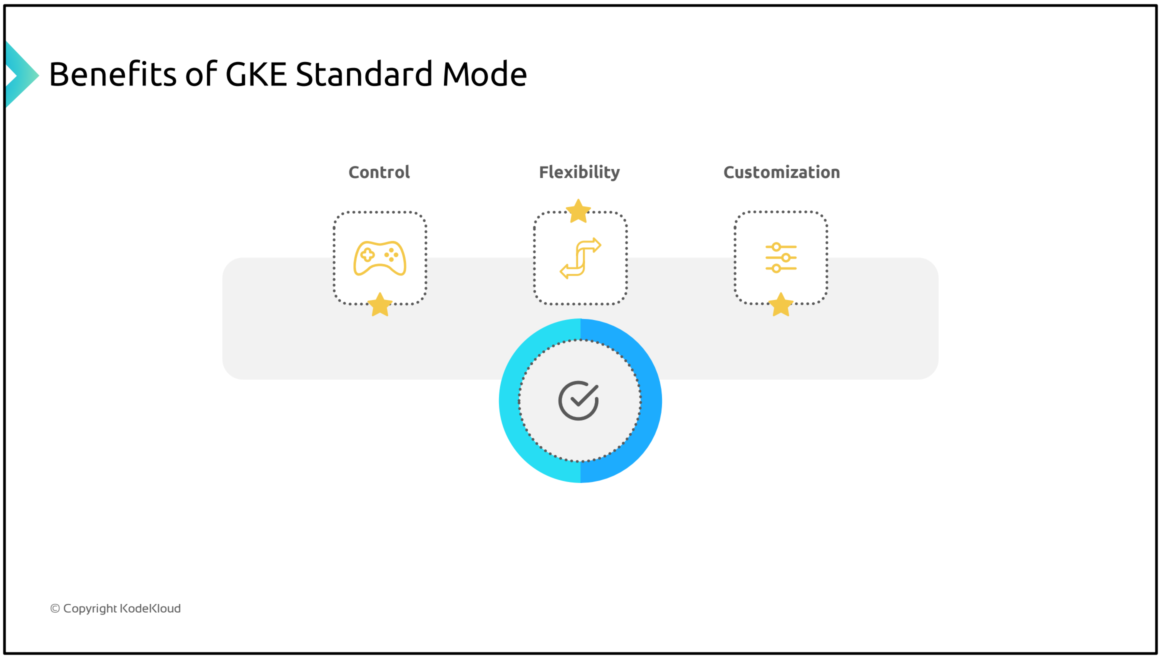


Choosing a GKE Standard Mode Type

There are a few factors that could be considered when choosing between zonal and regional GKE cluster, such as:

Workload requirements: Some workloads require high availability across zones, while others do not. Regional clusters provide better fault tolerance and are recommended for applications needing higher availability than zonal clusters.

The next thing to consider is **Budget**: Zonal clusters are less expensive than regional clusters because of the lesser number of resources needed for control plane.



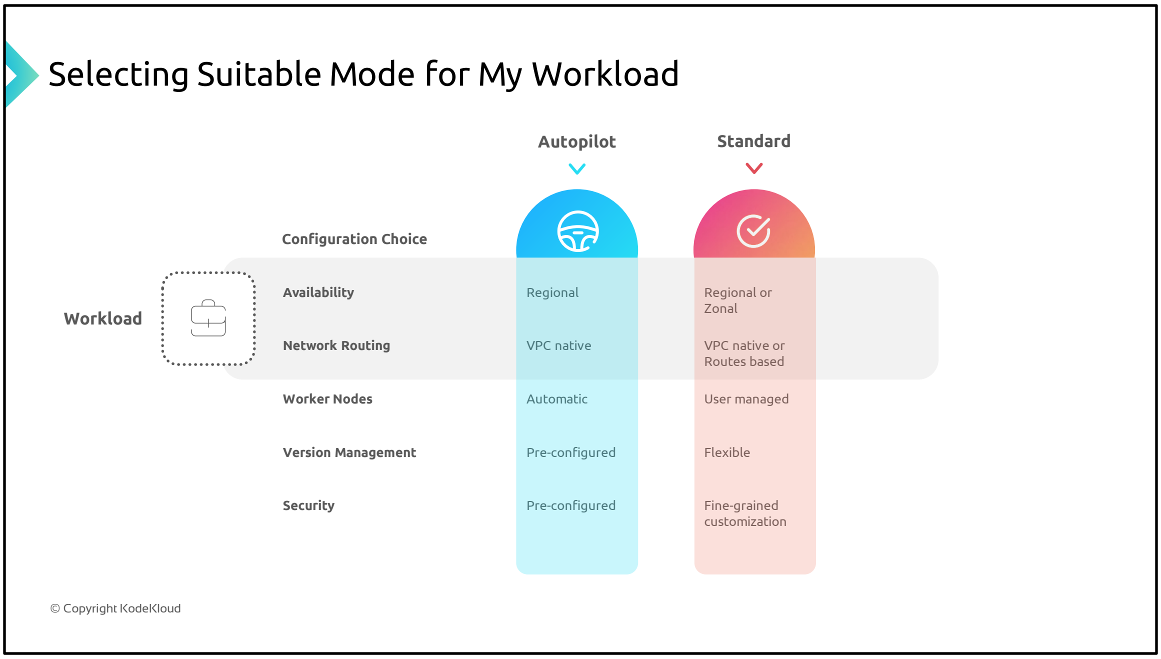
Benefits of GKE Standard Mode

Control
Flexibility
Customization

Some key advantages of using a GKE cluster in Standard mode are:

Control: As a user, you have full control over the underlying infrastructure of your cluster.
Flexibility: GKE Standard mode offers the options to choose the node type, size, and location of your nodes to meet application workload-specific needs. It offers both zonal and regional clusters, providing more flexibility and control over the Kubernetes deployments.
Customization: GKE cluster in standard mode also

allows customization of the security configuration of the cluster to meet specific business requirements.



Selecting Suitable Mode for My Workload

Workload

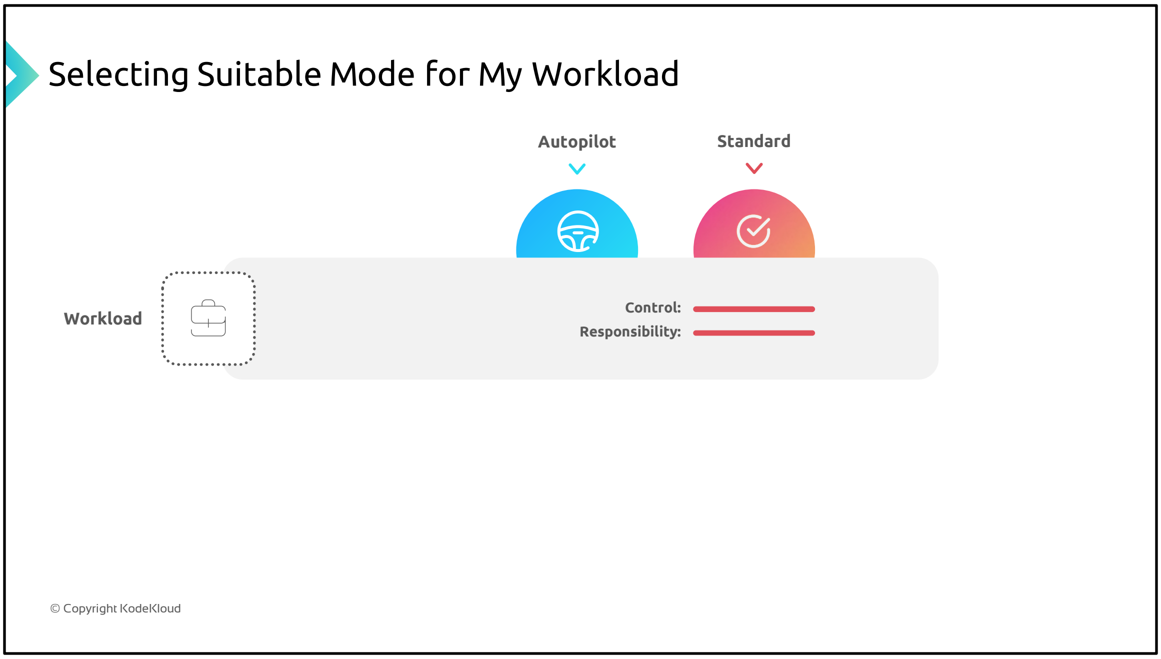
Configuration Choice	Autopilot	Standard
Availability	Regional	Regional or Zonal
Network Routing	VPC native	VPC native or Routes based
Worker Nodes	Automatic	User managed
Version Management	Pre-configured	Flexible
Security	Pre-configured	Fine-grained customization

Now that's a lot of information to digest for the different types of GKE clusters, their mode of operations and when would it make more sense to choose a particular mode for GKE cluster over the other. So, to make it simpler for you to understand, here are some key considerations to take into the account while choosing between a Standard or Autopilot mode in GKE.

In Autopilot mode, most of the choices are pre-configured by GKE for the cluster deployment. It provides a managed experience, optimized configurations, and automation, making it suitable for most production workloads.

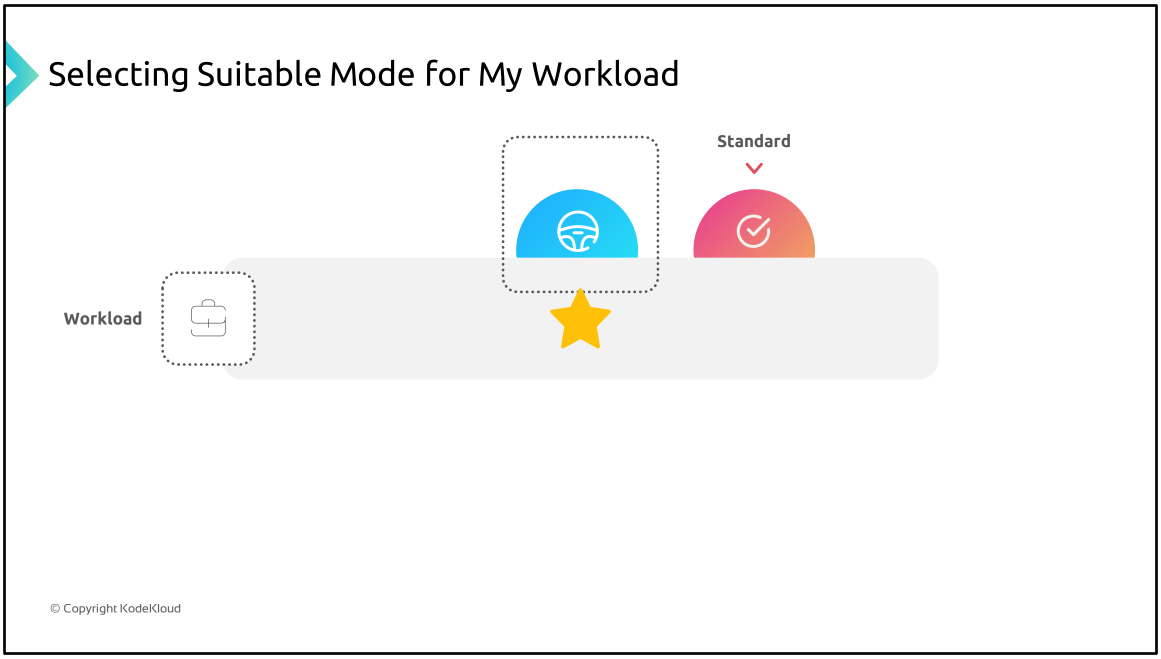
Whereas Standard mode offers more control and flexibility for advanced configurations and customization. For example:

- Specific configuration control for your cluster and nodes
- Fine-grained security customization
- Direct control over workload scheduling
- Flexibility in scaling and resource management
- Granular control over GKE version management



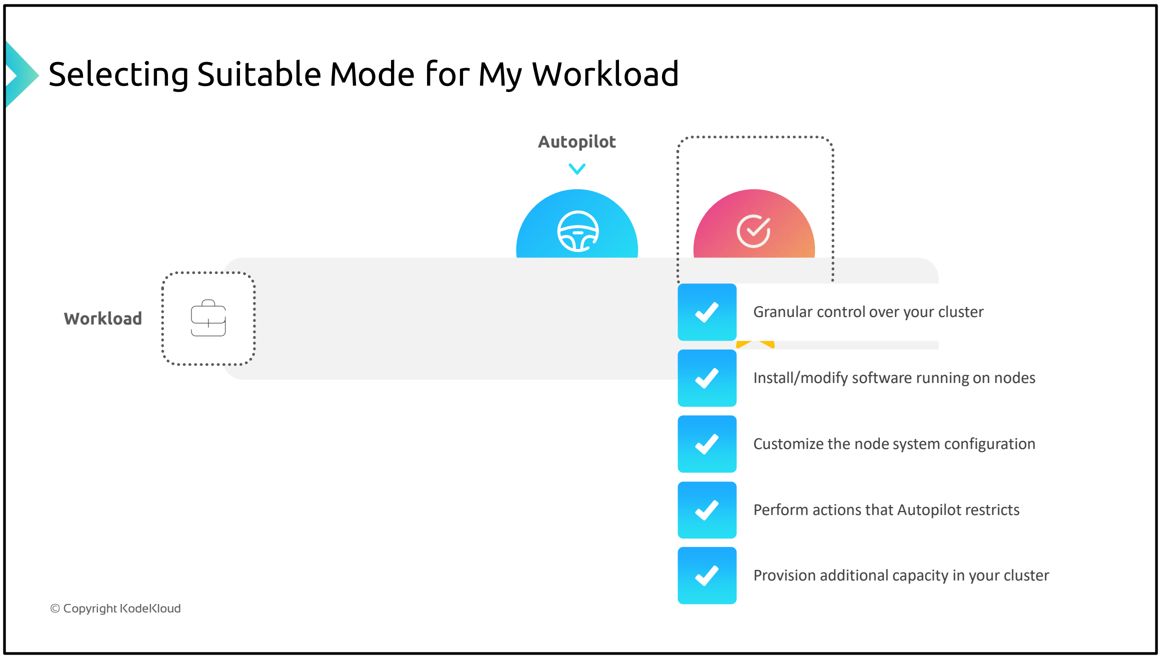
Selecting Suitable Mode for My Workload

However, it's important to note that with increased control comes increased responsibility for managing and optimizing your cluster.



Selecting Suitable Mode for My Workload

If you do not need that extra control or flexibility of Standard mode, then you should use Autopilot. Autopilot is a good choice for most workloads, and it is the most cost-effective way to run Kubernetes on Google Cloud.



Selecting Suitable Mode for My Workload

Autopilot
Workload
✓ Granular control over your cluster
✓ Install/modify software running on nodes
✓ Customize the node system configuration
✓ Perform actions that Autopilot restricts
✓ Provision additional capacity in your cluster

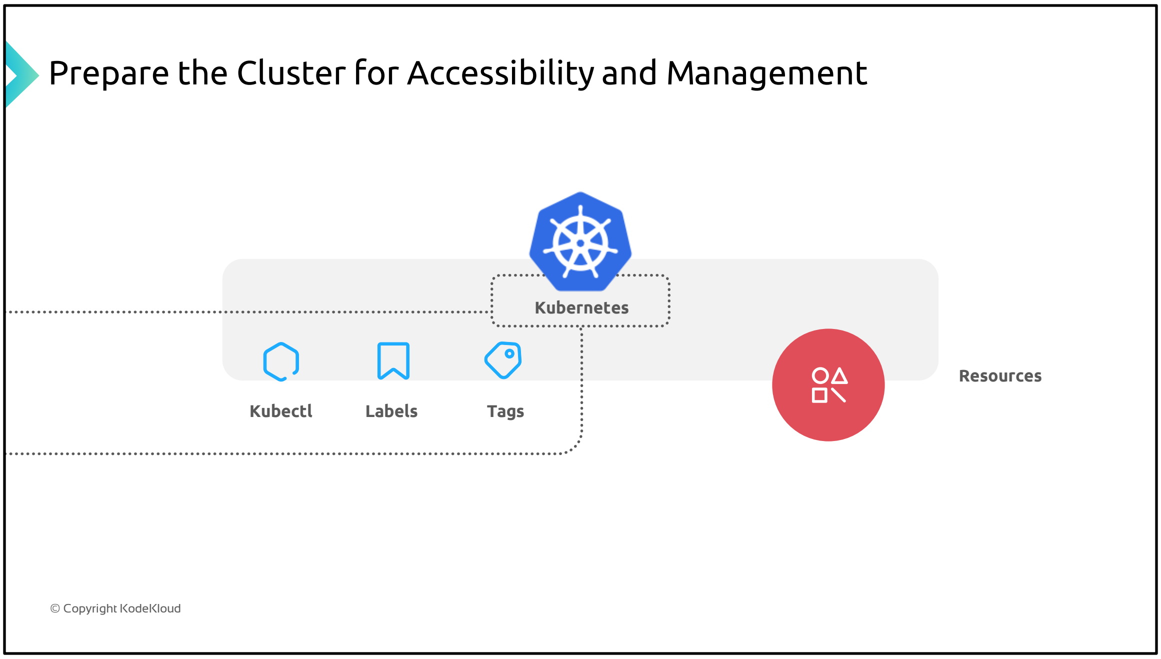
Some scenarios where you might want to consider standard mode are:

- More granular control over your cluster and node configuration is required. This includes the ability to directly connect to your nodes using SSH.
- The ability to install or modify software running on the nodes themselves is needed, such as changing the node operating system.
- You want to customize the node system configuration, such as by setting Linux sysctls.
- You want to perform actions that are not

recommended while deploying workloads to a GKE cluster and are restricted on Autopilot clusters, for example, running workloads in GKE-managed namespaces such as kube-system.

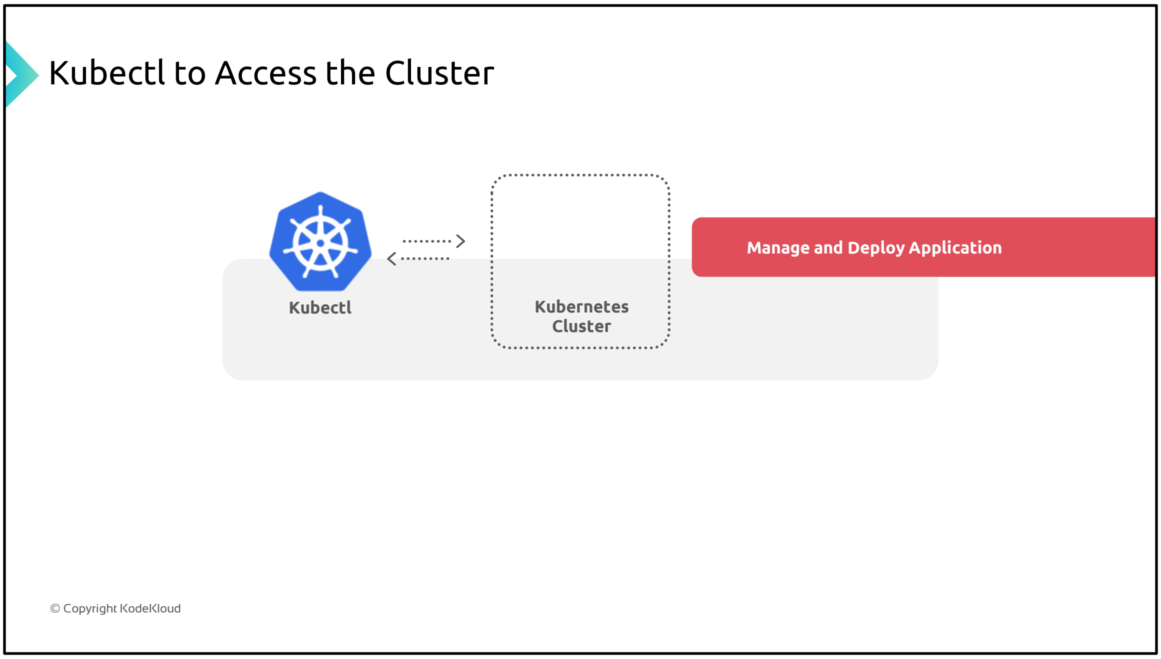
- There is a need to provision additional unused capacity in a cluster.

Prepare the Cluster for Accessibility and Management



Prepare the Cluster for Accessibility and Management

When working with GKE clusters, there are a few features that we use to ensure the cluster accessibility and efficient management. By leveraging kubectl, labels, and tags, administrators can efficiently manage, organize, and categorize resources within the cluster, enabling seamless navigation, efficient troubleshooting, and effective resource allocation.



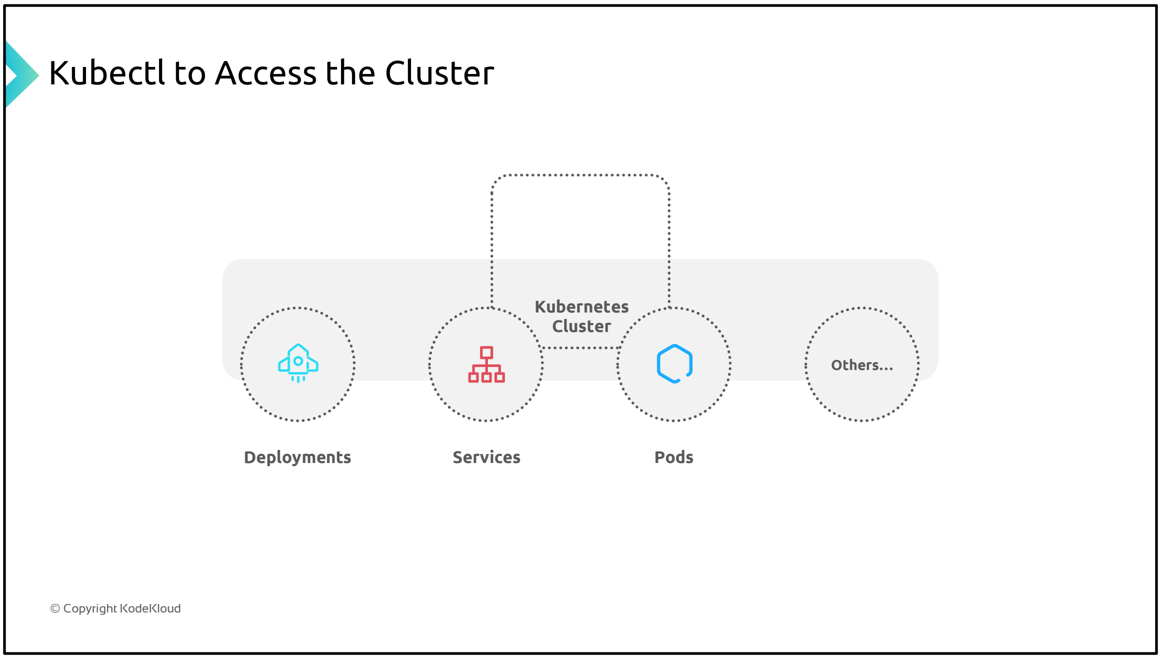
Kubectl to Access the Cluster

Kubectl

Kubernetes Cluster

Manage and Deploy Application

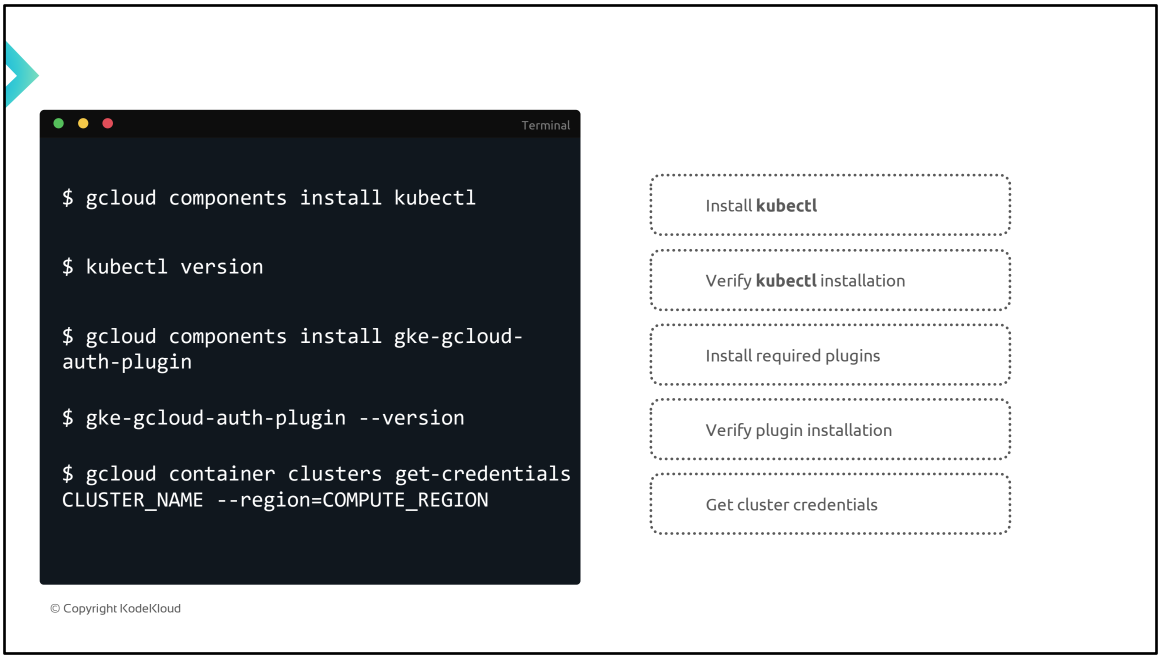
Kubectl is a command-line tool used to interact with any Kubernetes cluster. It is the primary interface for managing and deploying applications to a Kubernetes cluster.



Kubectl to Access the Cluster

With kubectl, a number of tasks can be performed such as creation and management of:

deployments,
services,
pods, and
other Kubernetes resources.



```
$ gcloud components install kubectl

$ kubectl version

$ gcloud components install gke-gcloud-
auth-plugin

$ gke-gcloud-auth-plugin --version

$ gcloud container clusters get-credentials
CLUSTER_NAME --region=COMPUTE_REGION
```

Install **kubectl**

Verify **kubectl** installation

Install required plugins

Verify plugin installation

Get cluster credentials

The steps required to have kubectl set up and be able to manage a GKE cluster from a machine are:

1. Install kubectl: Download and install the appropriate version of kubectl for your operating system. You can install `kubectl` using the Google Cloud CLI or an external package manager such as `apt` or `yum.` In this example, we used google cloud CLI.

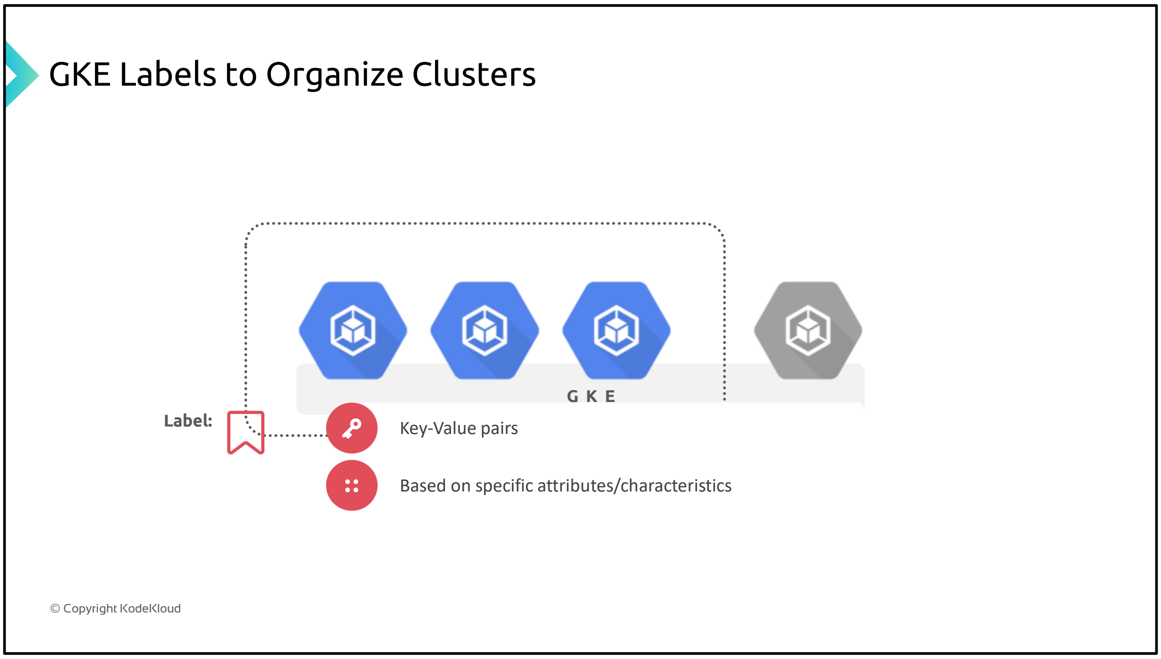
2. Verify that kubectl has been successfully installed by running the kubectl version.

3. kubectl and other Kubernetes clients require an authentication plugin, called **gke-gcloud-auth-**

plugin, which provides authentication tokens to communicate with GKE clusters. This plugin must be installed to use kubectl to interact with GKE. Run the command gcloud components install gke-gcloud-auth-plugin to install this plugin.

4. The installation of the plugin can be verified by running gke-gcloud-auth-plugin –version.
5. Lastly, run gcloud container clusters get-credentials CLUSTER_NAME, followed by region to retrieve the necessary credentials for accessing your GKE cluster with kubectl.

Please note that the installation of kubectl and gke-gcloud-auth-plugin is not required if you are using Google Cloud shell to access the cluster. All these components are preinstalled when an instance of cloud shell is fired up.

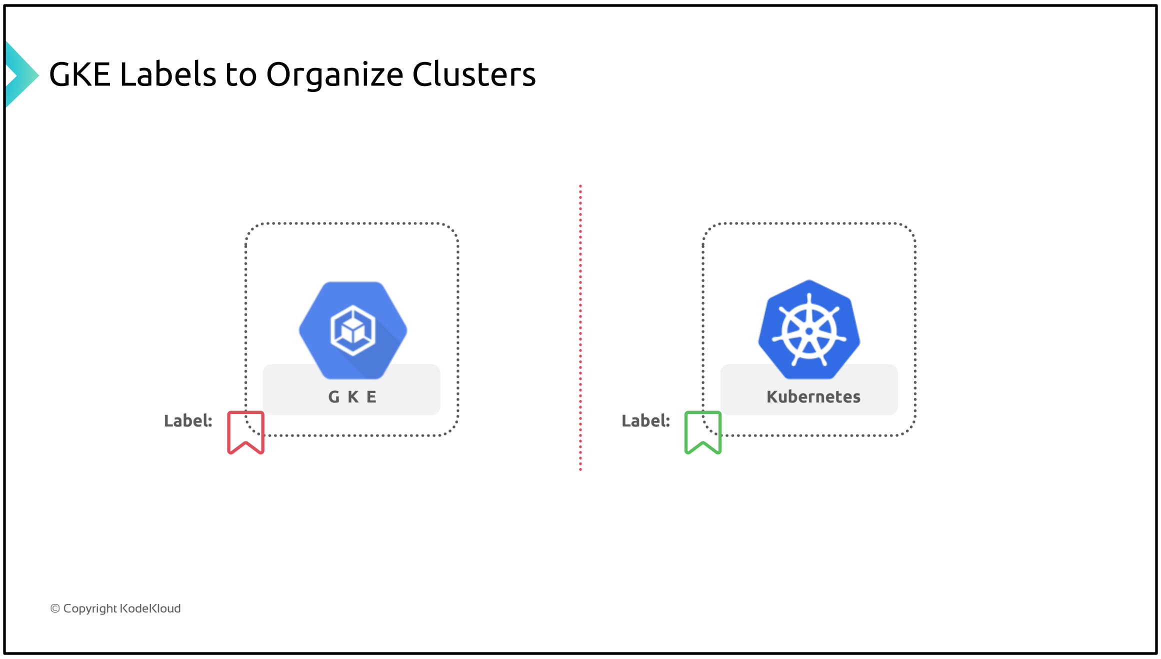


GKE Labels to Organize Clusters

Key-Value pairs

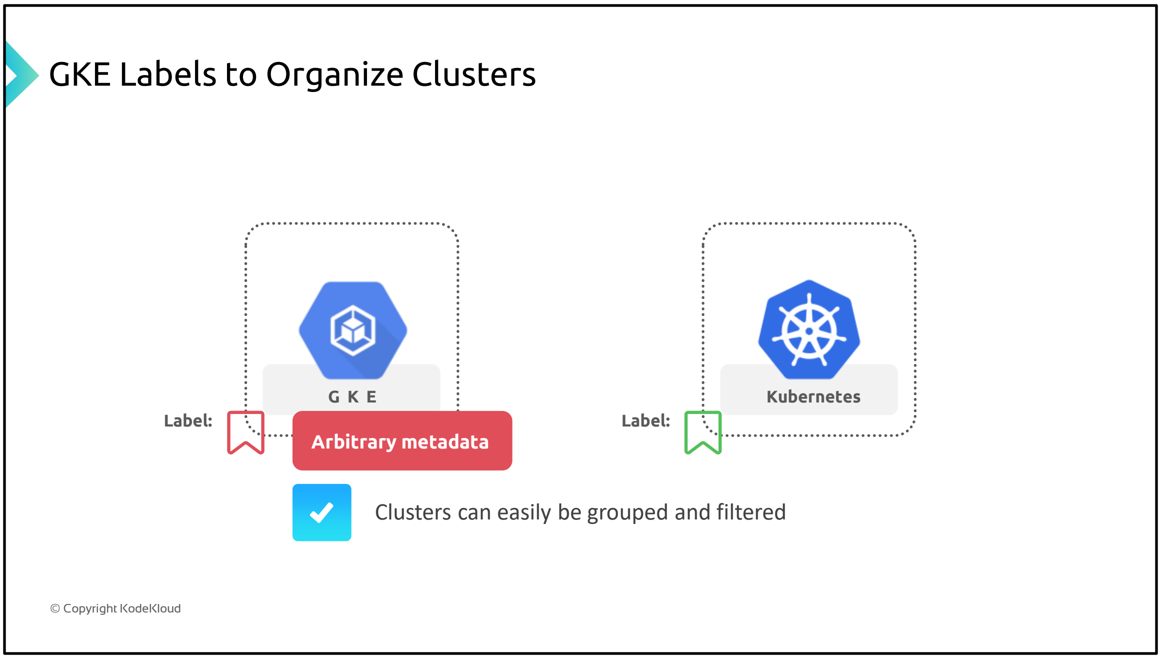
Based on specific attributes/characteristics

GKE cluster labels are key-value pairs that are assigned to GKE clusters to organize and categorize them based on specific attributes or characteristics.



GKE Labels to Organize Clusters

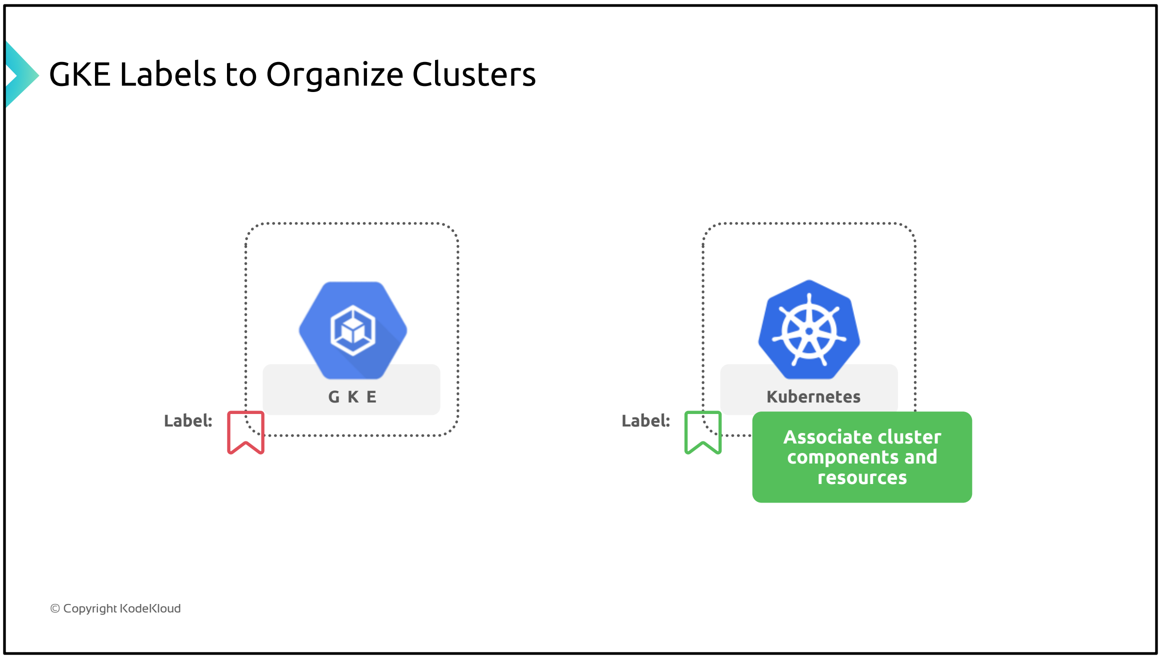
An important note is that the GKE cluster and node pool labels are distinct from labels in Kubernetes. The two labeling systems work independently and do not inherit or share labels.



GKE Labels to Organize Clusters

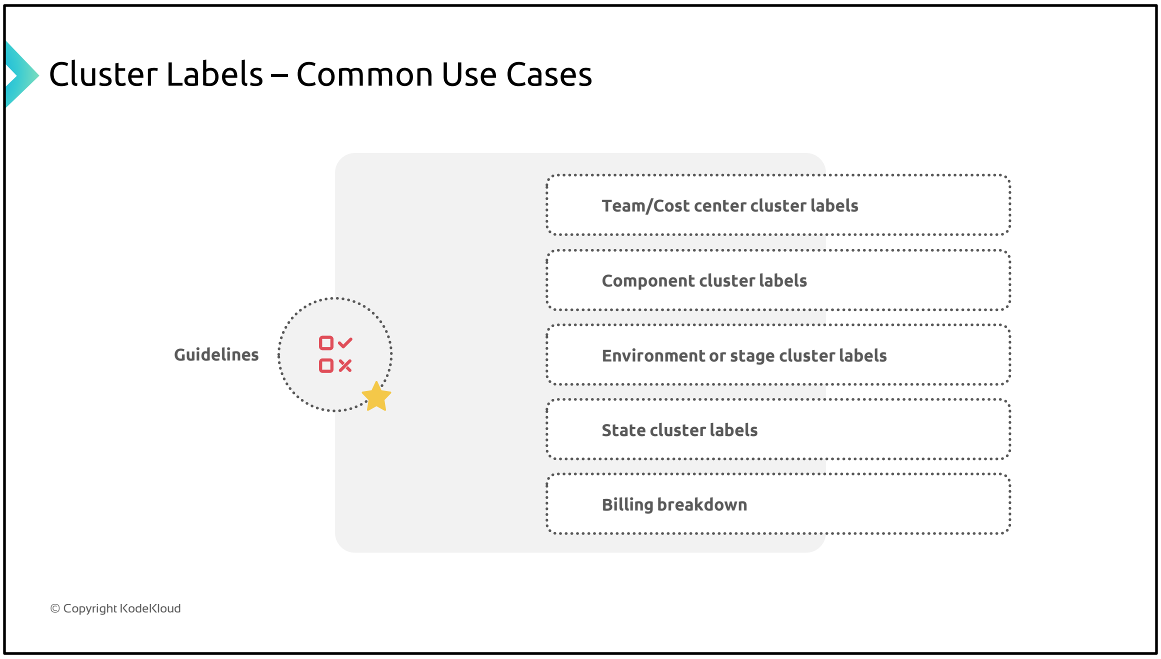
✓ Clusters can easily be grouped and filtered

GKE cluster and node pool labels are arbitrary metadata attached to the resources that are used to track usage and billing information. By applying labels, the clusters can easily be grouped and filtered for purposes such as management, monitoring, or resource allocation.



GKE Labels to Organize Clusters

Whereas the labels in a Kubernetes cluster are used by the system to associate cluster components and resources such as pods and nodes with one another and manage resource lifecycles.

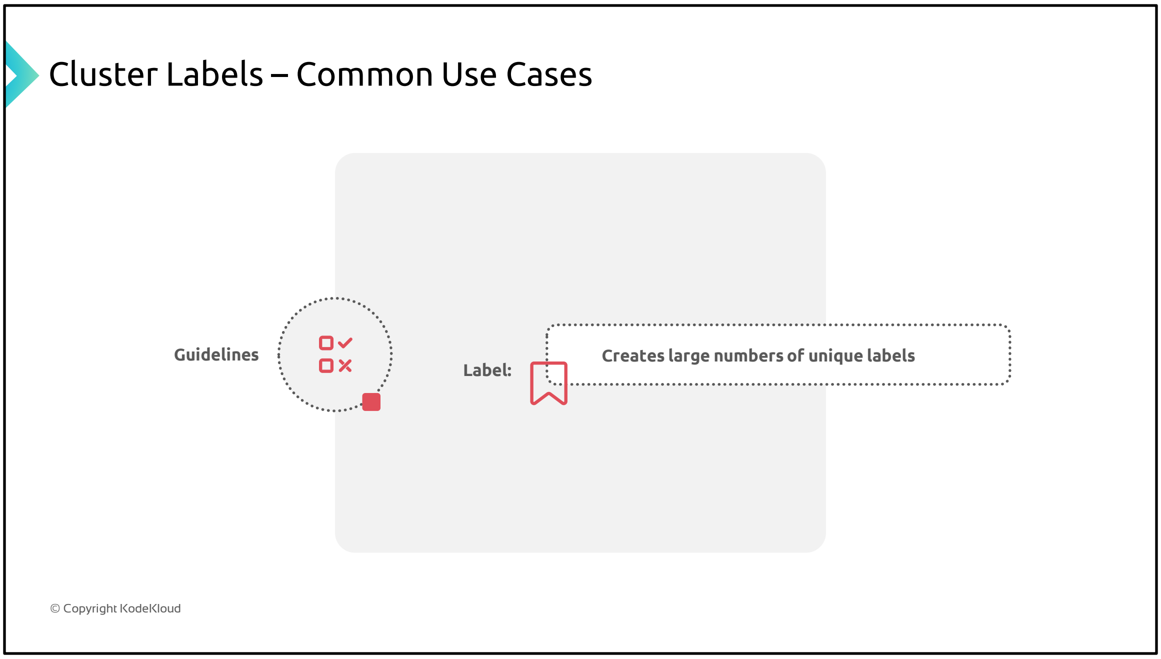


Cluster Labels – Common Use Cases

There are a few recommendations and guidelines for how and when the labels should be used. Here are some common use cases for cluster labels:

- **Team or cost center cluster labels:** Add labels based on a team or cost center to distinguish the clusters owned by different teams for example, using separate labels with key:value pair as team:research and team:analytics for two different clusters; this type of label can be used for cost accounting or budgeting;
- **Component cluster labels:** For example, component:redis, or component:ingest;

- **Environment or stage cluster labels:** For example, environment:production and environment:test;
- **State cluster labels:** For example, state:active, state:readytodelete; and, lastly,
- **Billing breakdown: …?**

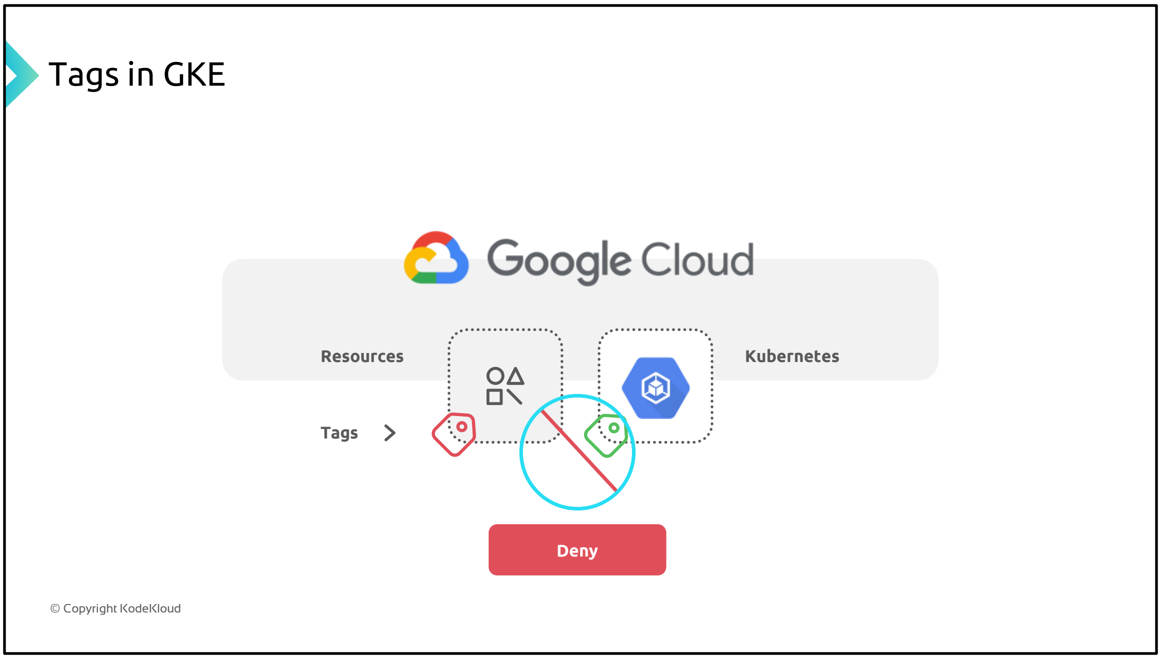


Cluster Labels – Common Use Cases

Guidelines

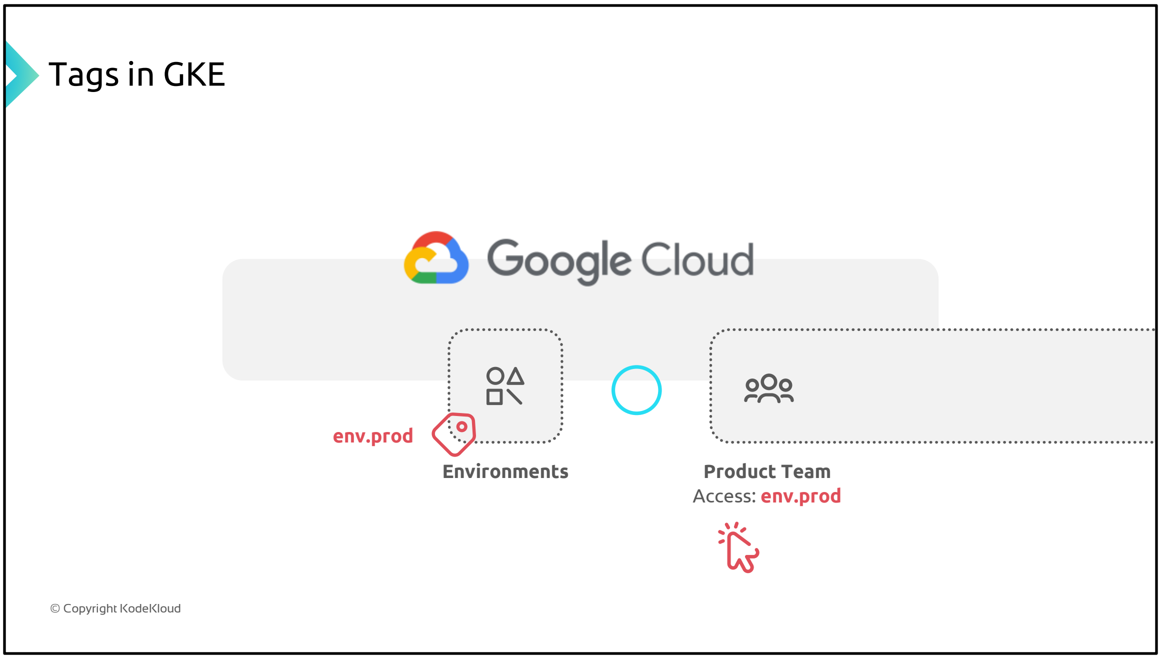
Label: Creates large numbers of unique labels

It’s not recommended to create large numbers of unique labels, such as for timestamps or individual values for every API call.



Tags in GKE

Now, Tags are also key-value pairs. Tags can be attached to any Google Cloud resource, including GKE clusters. Tags are used to conditionally allow or deny access to resources using policies based on whether a resource has a specific tag.



Tags in GKE

Google Cloud

env.prod

Environments

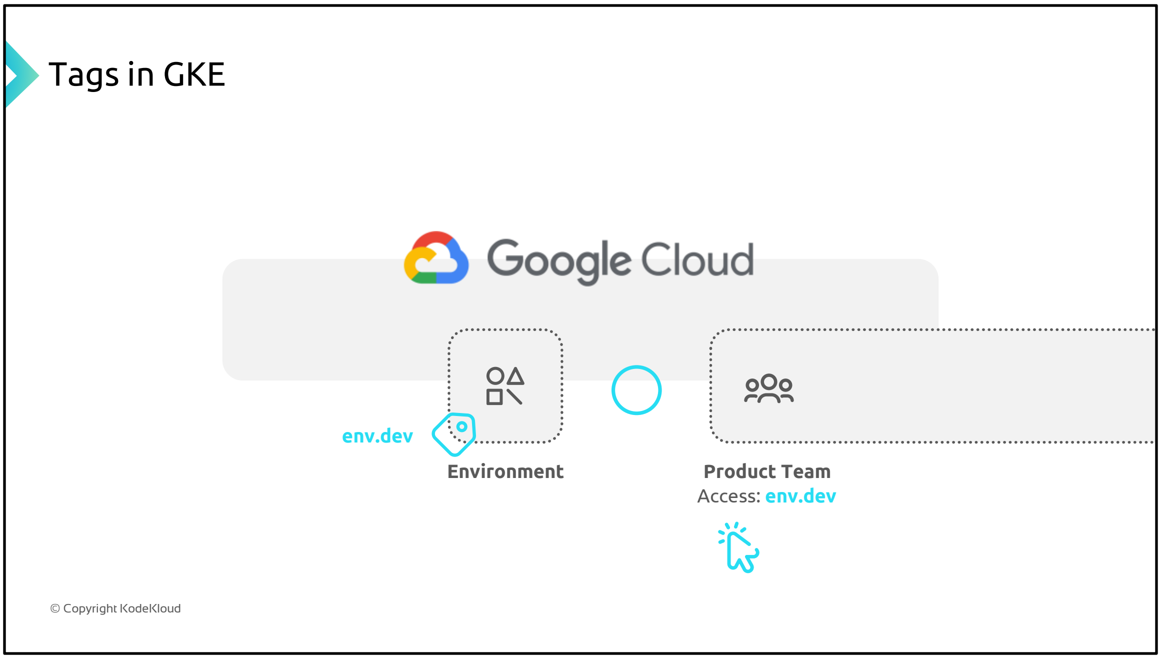
Product Team

Access: env.prod

One of the main use cases is implementing access controls and Tags to conditionally grant Identity and Access Management (IAM) roles based on whether a cluster has that specific tag or not.

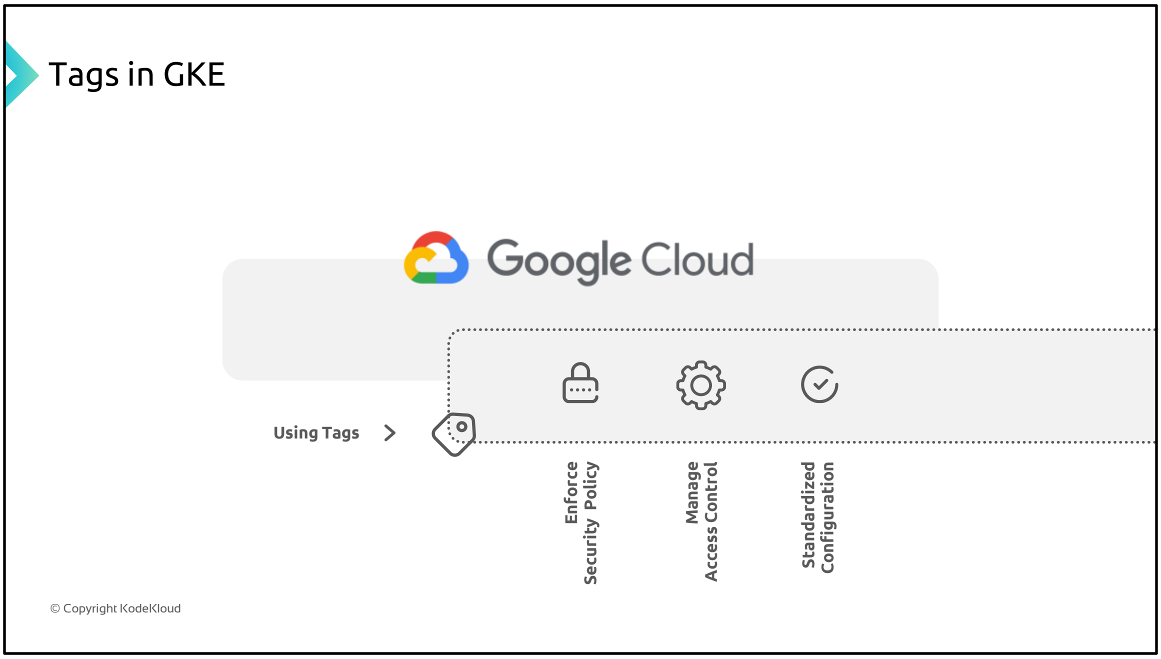
For example, you can create a tag called env to represent the environment for a cluster with the values prod and dev.

Then a policy can be created that grants users in the prod team access to clusters that have the env:prod tag



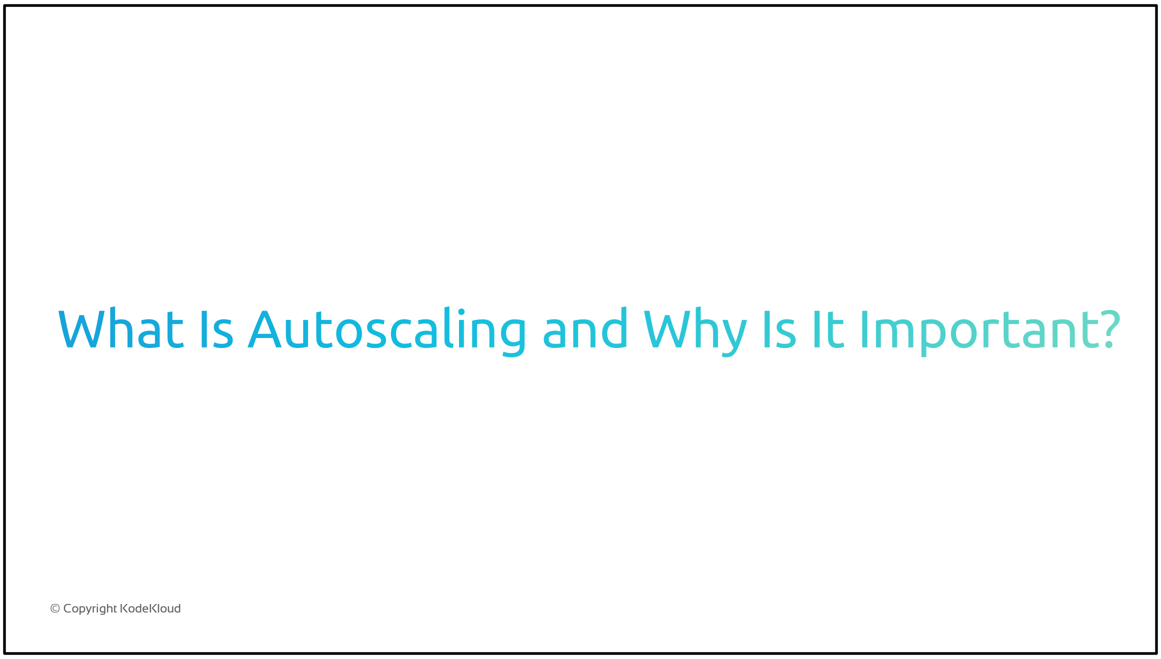
Tags in GKE

….and users in the dev team access to clusters that have the env:dev tag.

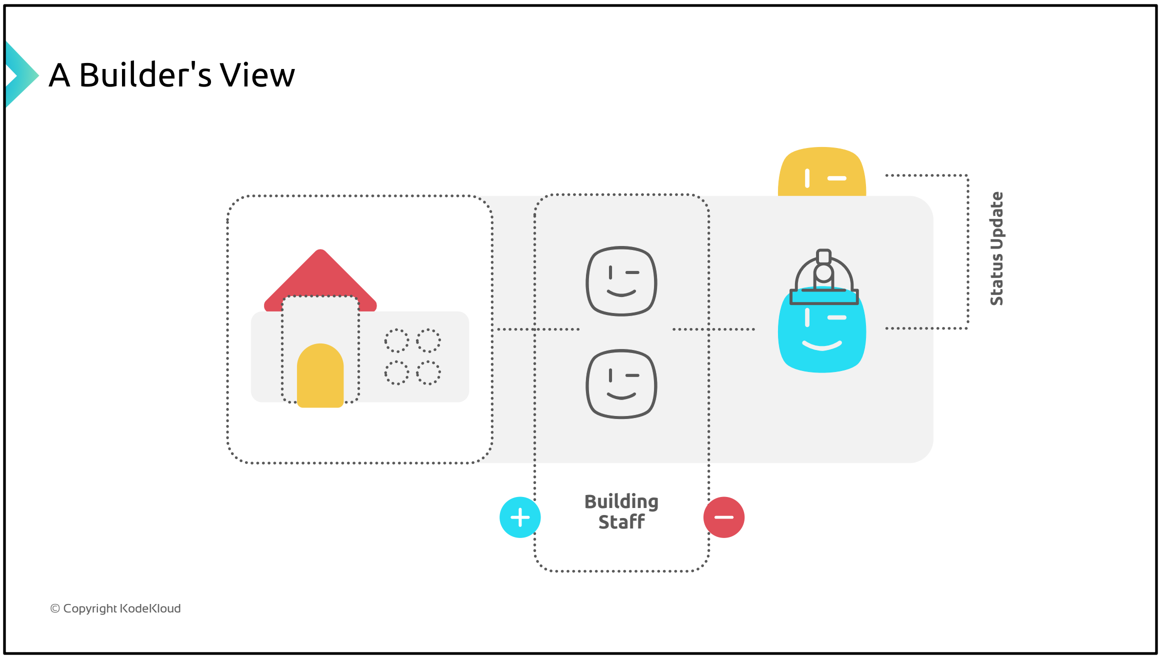


Tags in GKE

Using tags effectively allows us to enforce consistent security policies, easily manage access control, and apply standardized configurations to the GKE clusters.



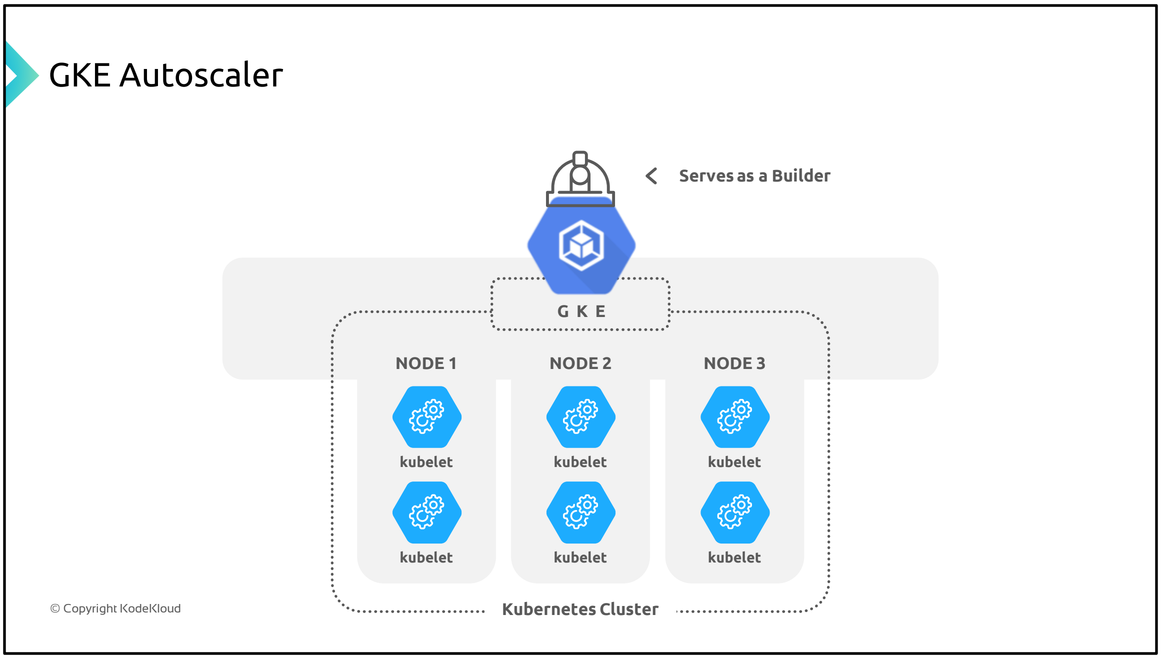
What Is Autoscaling and Why Is It Important?



A Builder's View

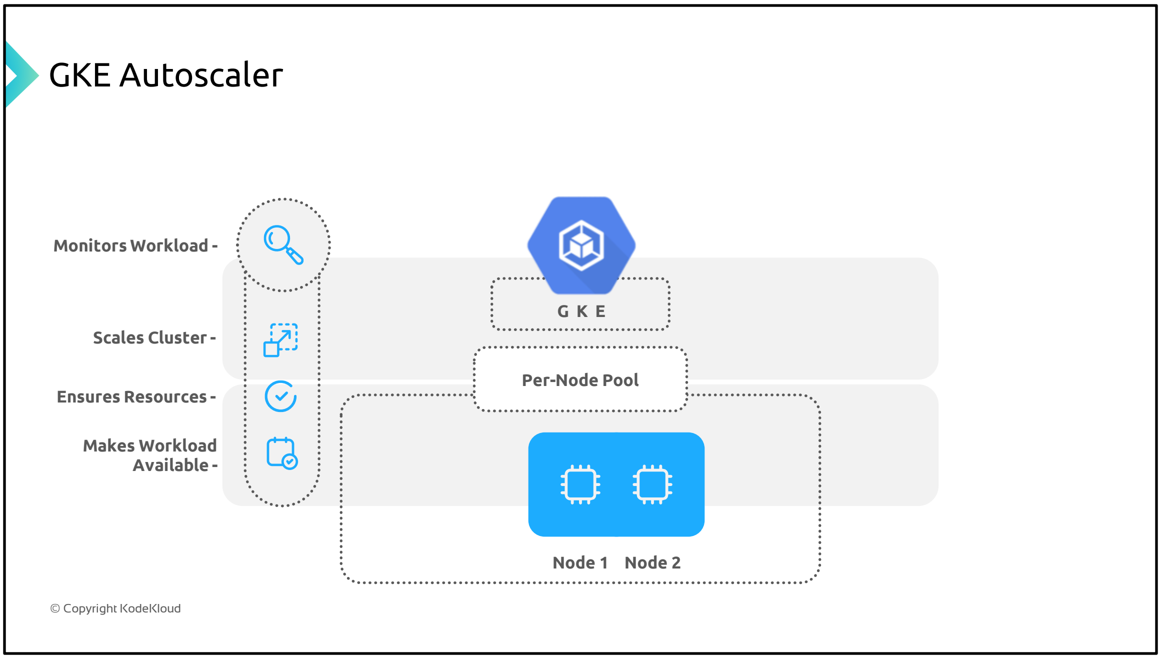
Let’s again consider the builder who we hired to construct a house. He now has a team of workers who are building the house. He has a plan that outlines how to build the house. As he is building the house, he needs to make sure that he has enough workers to complete the house on time. If he doesn't have enough workers, the house will not be completed on time. On the other hand, If he has too many workers, he will be wasting money.

As the project progresses, the number of tradies needed may vary based on the workload. Sometimes you may require more carpenters or electricians, and at other times, fewer. The builder plays a crucial role in dynamically adjusting the workforce to match the current demand, ensuring efficient resource utilization.



GKE Autoscaler

In the world of Google Kubernetes Engine (GKE), the GKE Autoscaler serves as the builder for our Kubernetes cluster. It **monitors the workload** on the cluster and automatically **adds or removes nodes** as needed to keep the workload running smoothly. It **dynamically scales** the cluster up or down, **ensuring** that we have the right amount of resources available to run your containerized applications optimally. This helps **in increasing the availability** of the workloads when you need it, while controlling costs.



GKE Autoscaler

Monitors Workload -

Scales Cluster -

Ensures Resources -

Makes Workload Available -

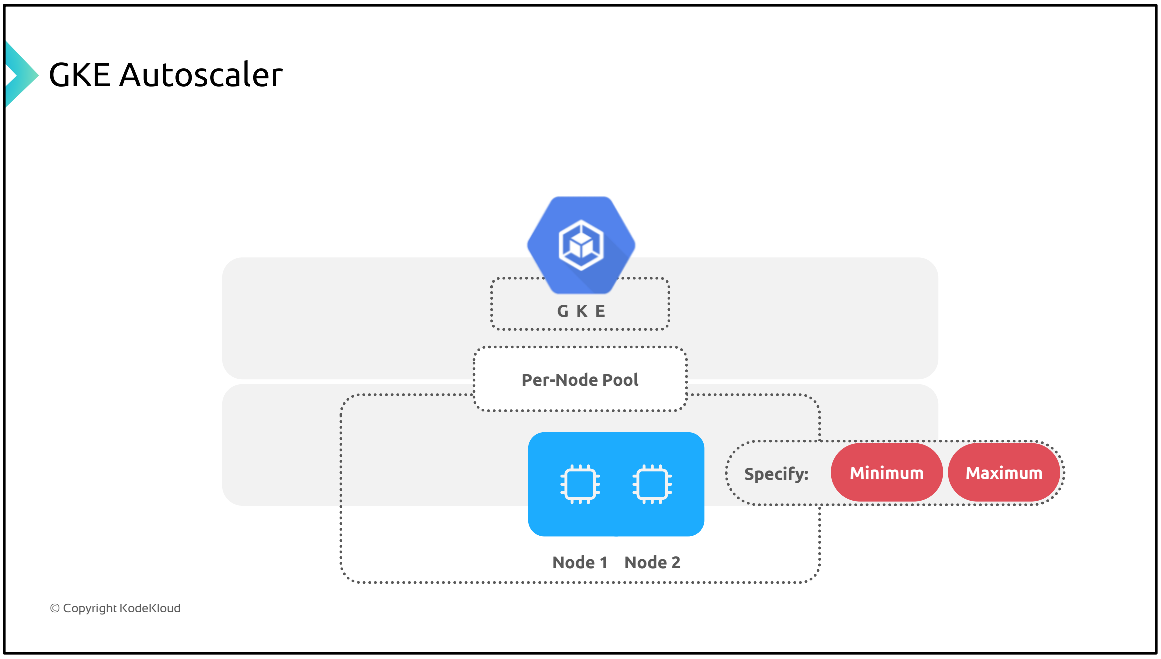
GKE

Per-Node Pool

Node 1

Node 2

In the world of Google Kubernetes Engine (GKE), the GKE Autoscaler serves as the builder for our Kubernetes cluster. It **monitors the workload** on the cluster and automatically **adds or removes nodes** as needed to keep the workload running smoothly. It **dynamically scales** the cluster up or down, **ensuring** that we have the right amount of resources available to run your containerized applications optimally. This helps **in increasing the availability** of the workloads when you its needed, while controlling costs at the same time.



GKE Autoscaler

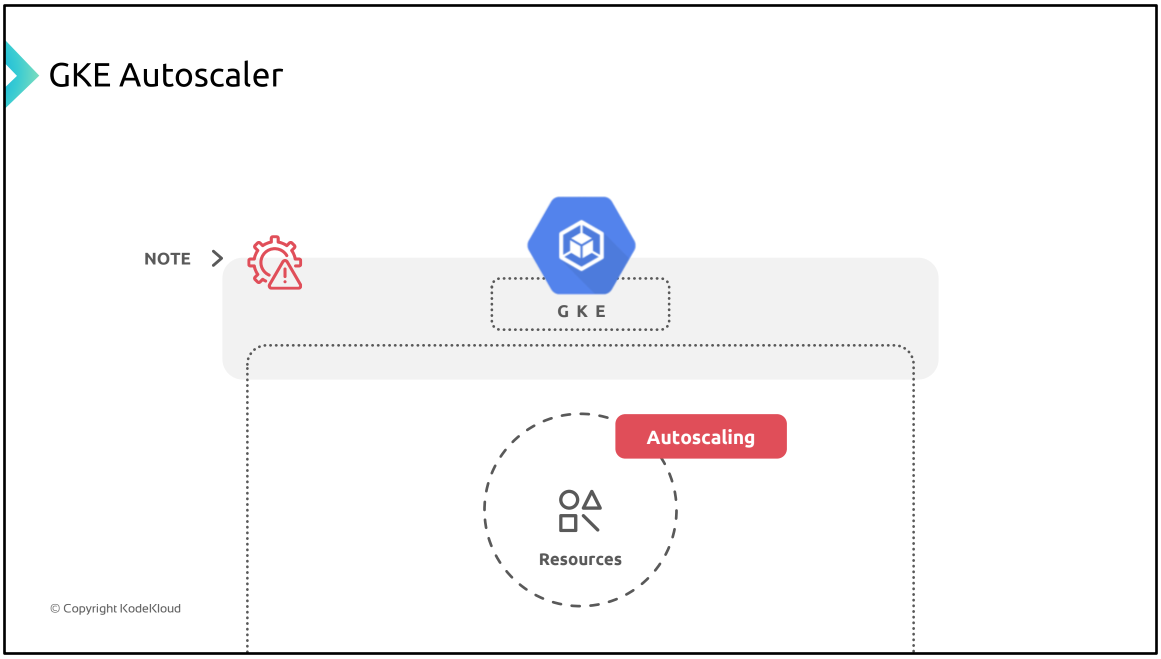
G K E

Per-Node Pool

Specify: Minimum Maximum

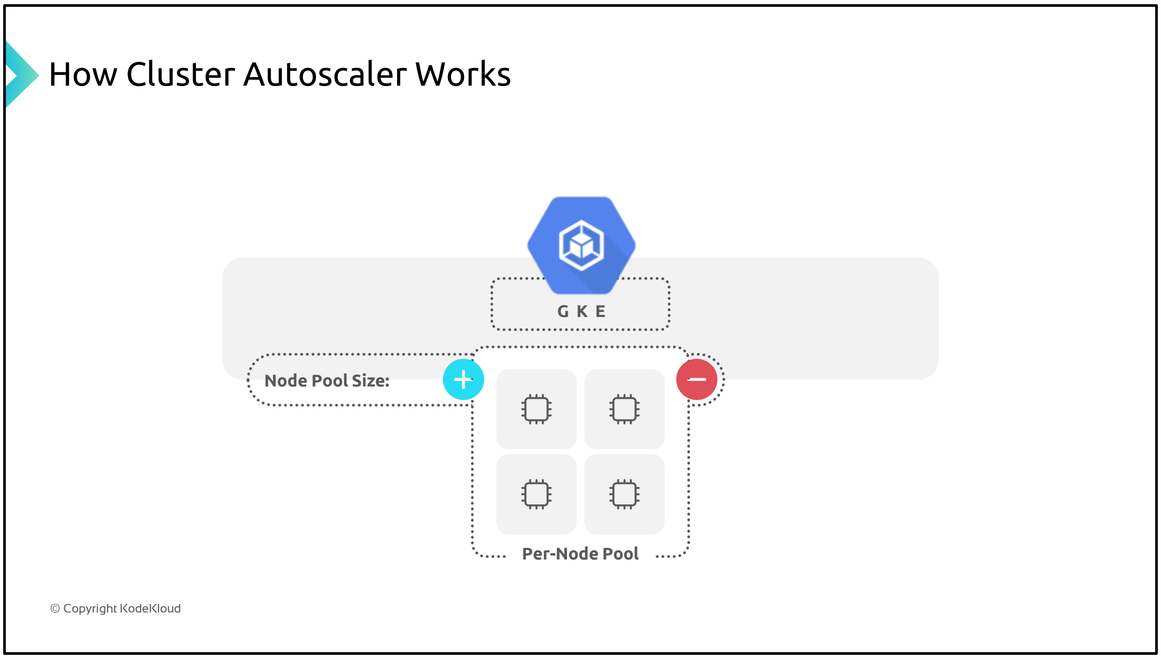
Node 1 Node 2

It's not required to manually add or remove nodes or over-provision the node pools of a GKE cluster. Instead, a minimum and maximum size for the node pool is specified in the cluster configuration, and the rest is automatic.



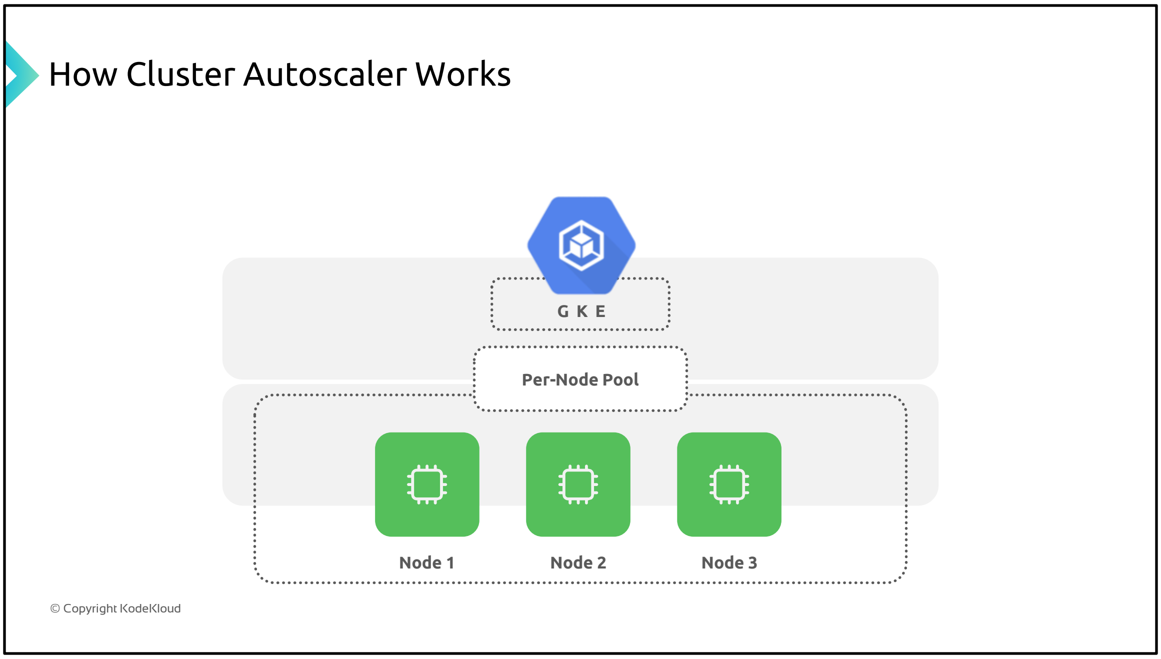
GKE Autoscaler

Note: In some scenarios, where workloads are not designed to tolerate potential disruption, if resources are deleted or moved when autoscaling a cluster, the workloads might experience transient disruption. For example, if the workload consists of a controller with a single replica, that replica's pod might be rescheduled onto a different node if its current node is deleted. To increase a workload's tolerance to interruption, consider deploying workloads using a controller with multiple replicas.



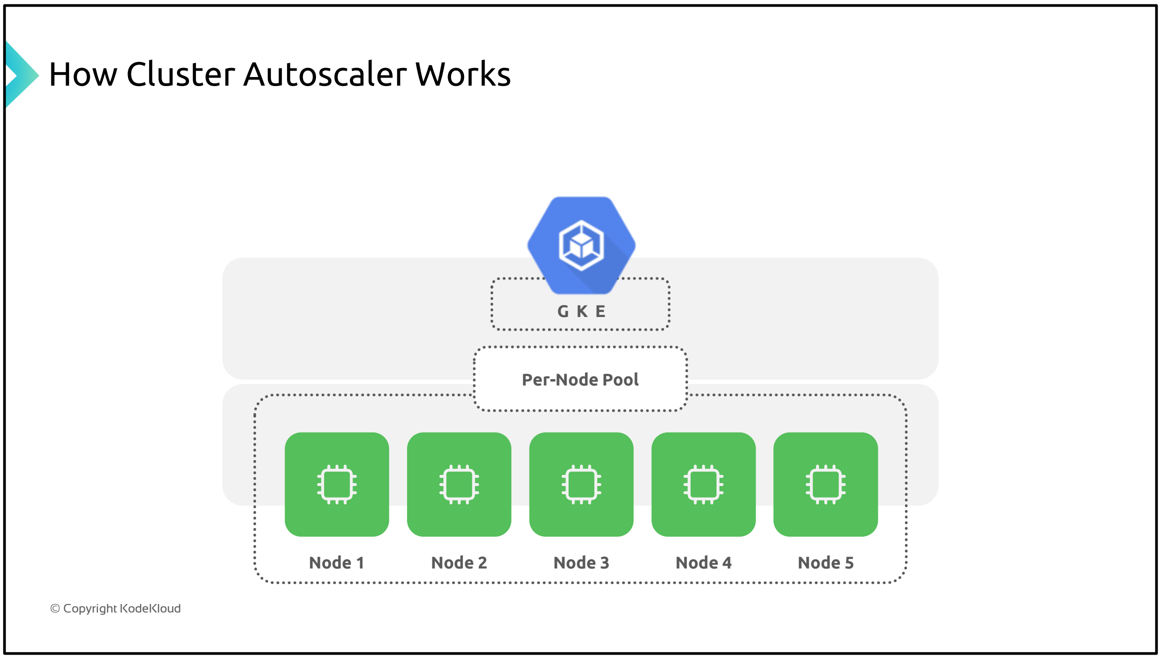
How Cluster Autoscaler Works

GKE autoscaler works on a per-node pool basis. That means when configuring a node pool with cluster autoscaler, we specify a minimum and maximum number of nodes for the node pool and is referred to as size of the node pool.



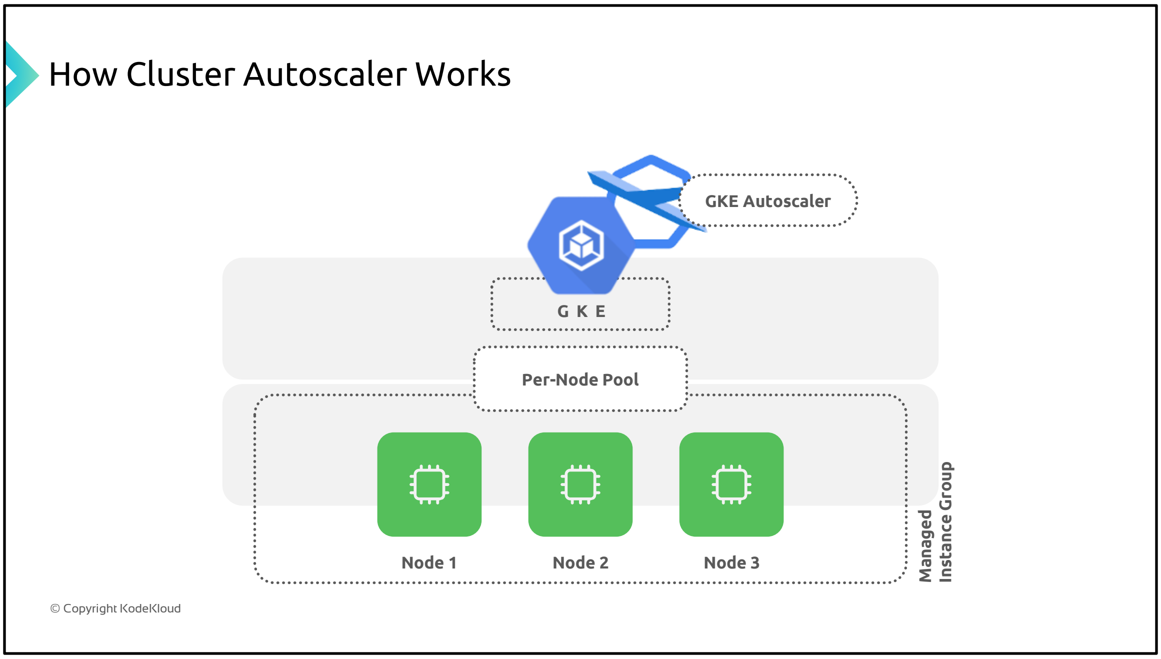
How Cluster Autoscaler Works

Based on the size of the node pool defined, …



How Cluster Autoscaler Works

Cluster autoscaler then increases…

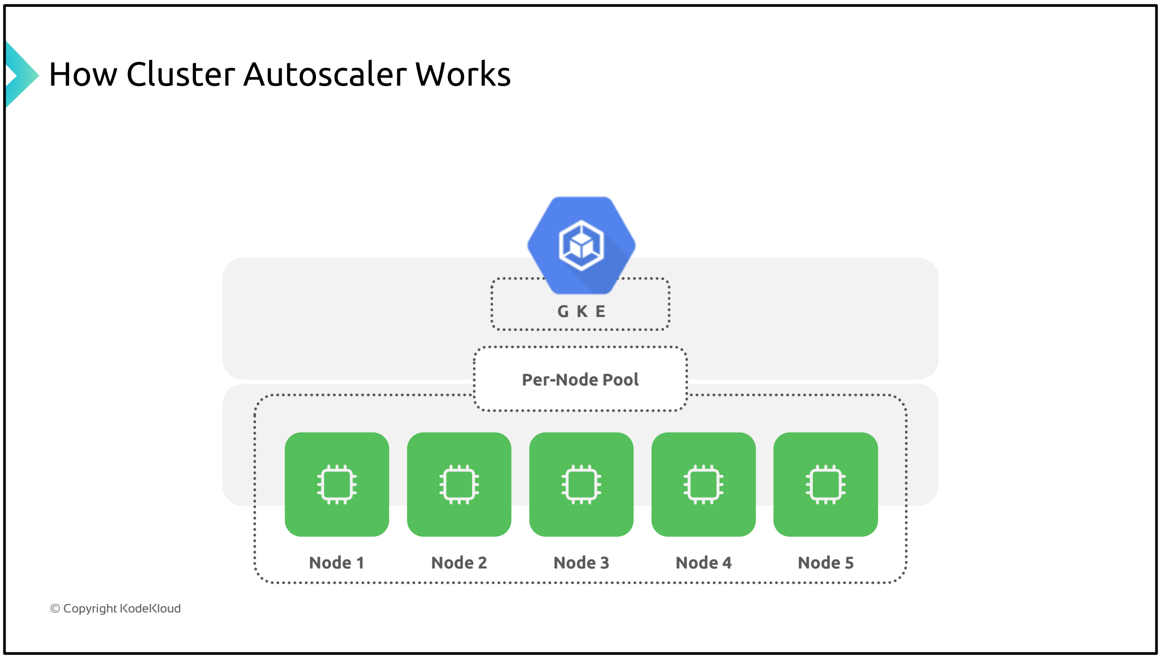


How Cluster Autoscaler Works

© Copyright KodeKloud

… or decreases the size of the node pool automatically by adding or removing worker nodes inside the node pool. These nodes are a set of virtual machine (VM) instances in the underlying Compute Engine Managed Instance Group (MIG) for the node pool.

The scaling decisions are made by autoscaler based on the resource requests of pods running on that node pool's nodes, not the actual resource utilization of the nodes. The status of pods and nodes is periodically checked by autoscaler to take necessary actions.



How Cluster Autoscaler Works

G K E

Per-Node Pool

Node 1

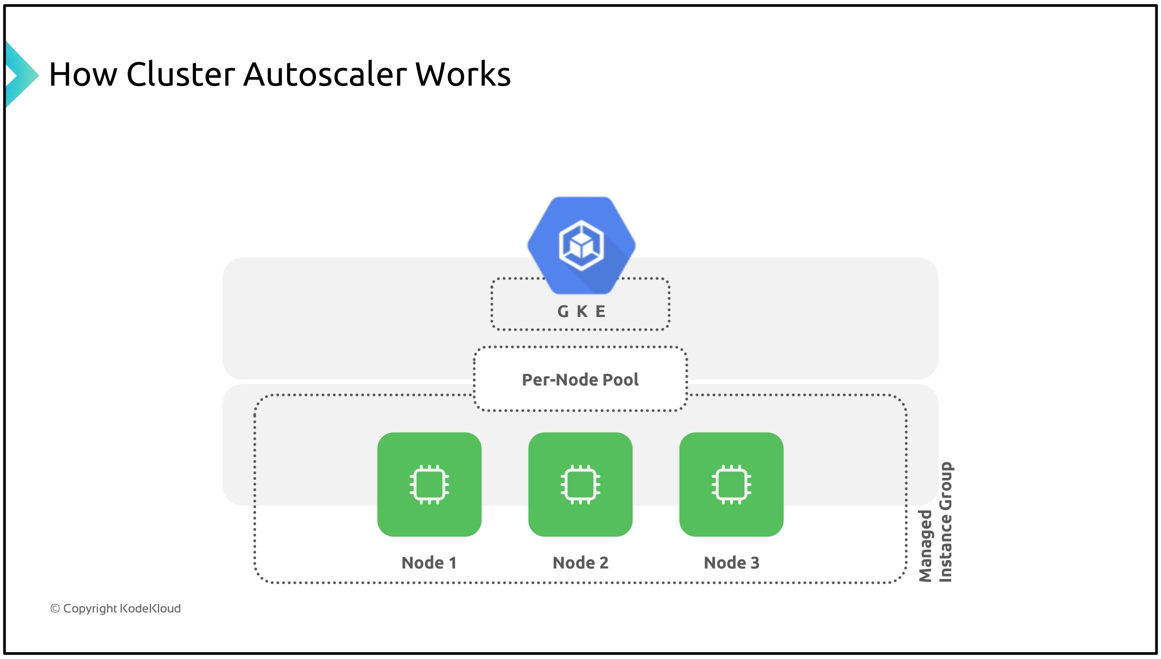
Node 2

Node 3

Node 4

Node 5

If pods are unschedulable because there are not enough nodes in the node pool cluster, autoscaler adds nodes inside the node pool, up to the maximum size of the node pool that was configured.



How Cluster Autoscaler Works

G K E

Per-Node Pool

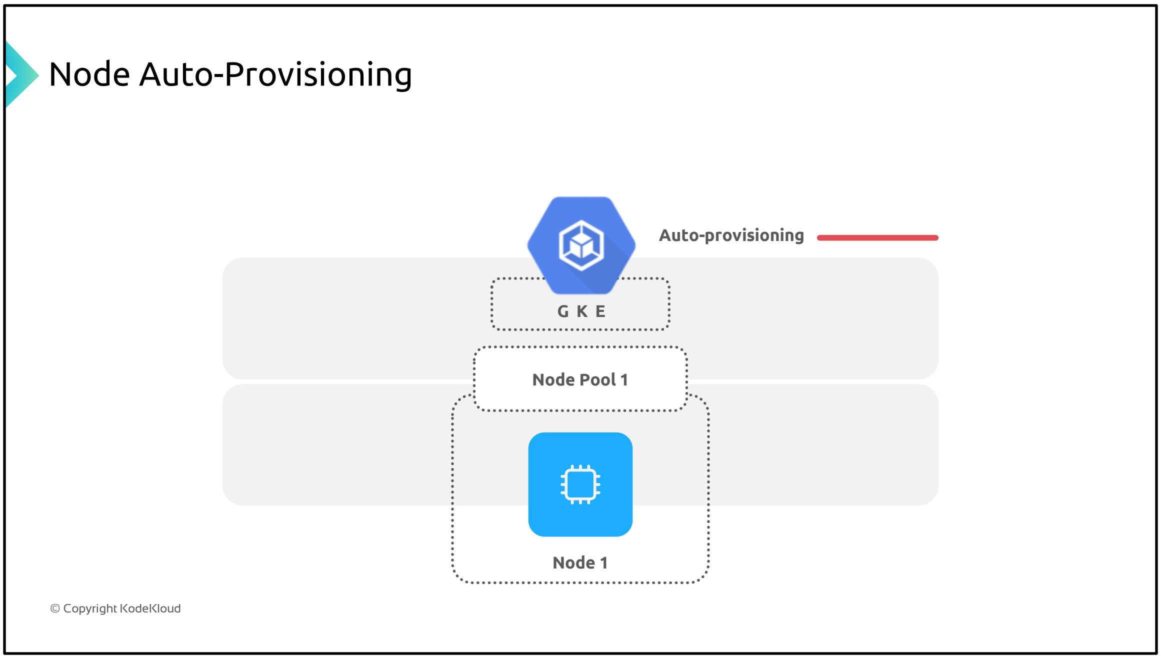
Node 1

Node 2

Node 3

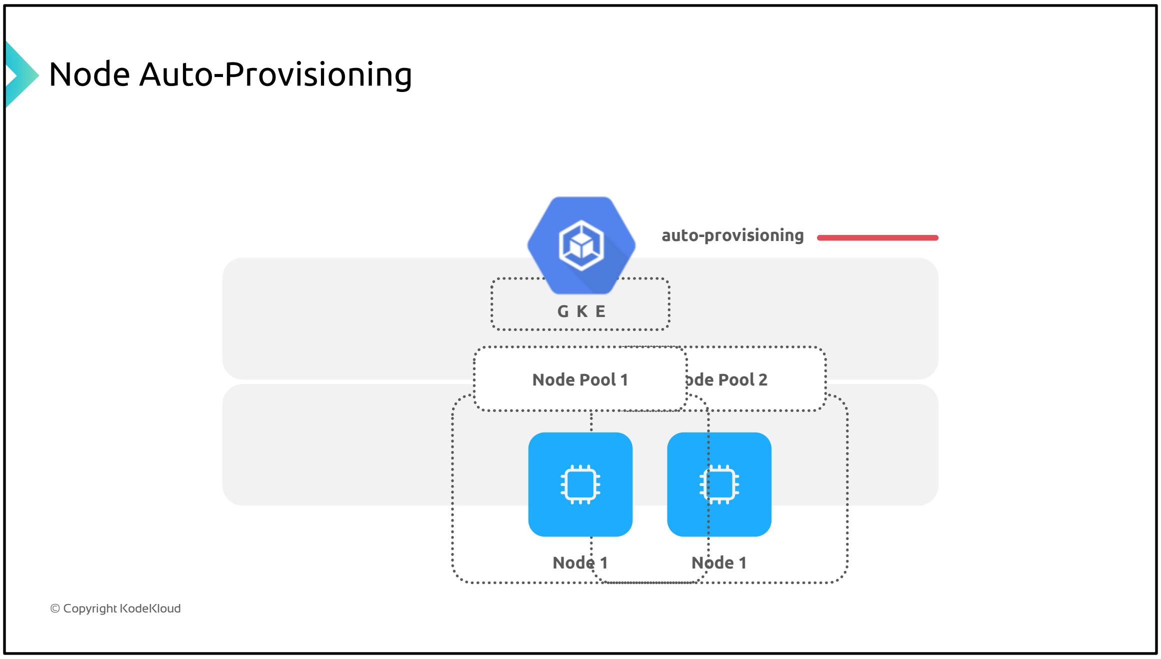
Managed
Instance Group

However, if all pods could be scheduled with fewer nodes in the node pool, autoscaler removes nodes from the node pool, bringing down the node numbers to the minimum size of the node pool.



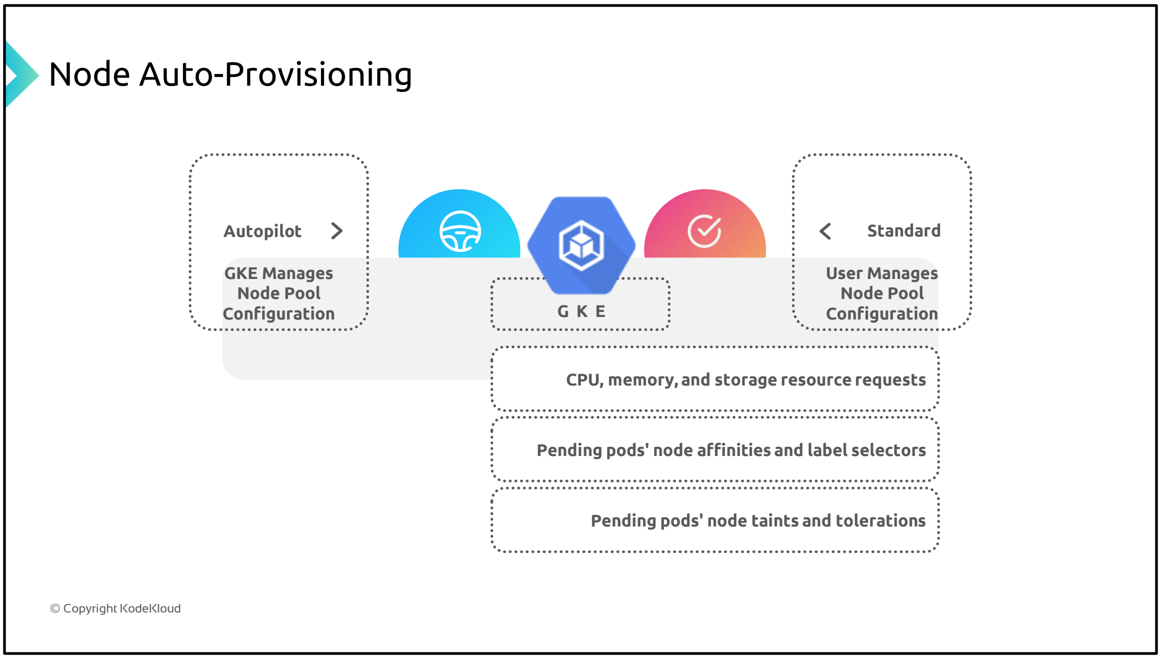
Node Auto-Provisioning

Now, as we discussed, cluster autoscaler scales the number of nodes on a per-node pool basis, which means that nodes are added or removed in the cluster inside a node pool based on the minimum and maximum number of nodes that is set during node pool configuration. However, with node auto-provisioning enabled, ...



Node Auto-Provisioning

...GKE dynamically manages the creation and deletion of node pools for the cluster, allowing the cluster to scale based on the workload demands and specifications of unschedulable pods.



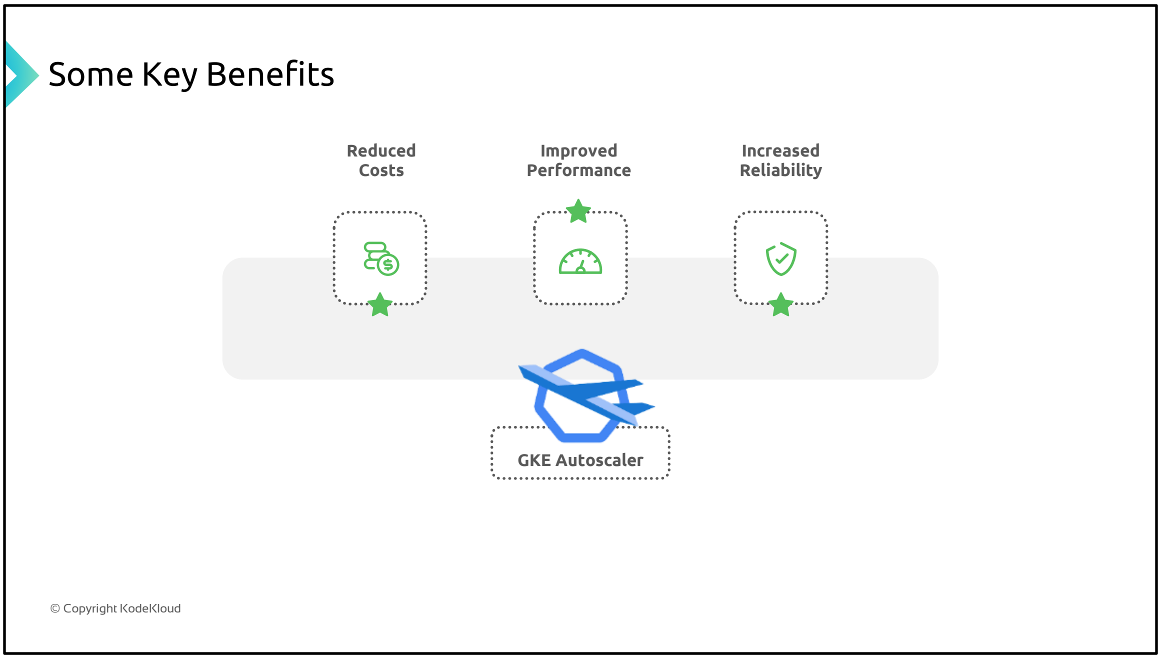
Node Auto-Provisioning

Autopilot
GKE Manages Node Pool Configuration
G K E
Standard
User Manages Node Pool Configuration
CPU, memory, and storage resource requests
Pending pods' node affinities and label selectors
Pending pods' node taints and tolerations

With Autopilot clusters, we don't need to manually provision nodes or manage node pools because node pools are automatically provisioned and managed through node auto-provisioning for us by GKE cluster. However, in Standard Google Kubernetes Engine (GKE) clusters, node auto-provisioning needs to be configured by us. Without configuring node auto-provisioning in a Standard mode, GKE adds or removes nodes only from user-created node pools.

With node auto-provisioning, new node pools are created and deleted automatically based on the following information:

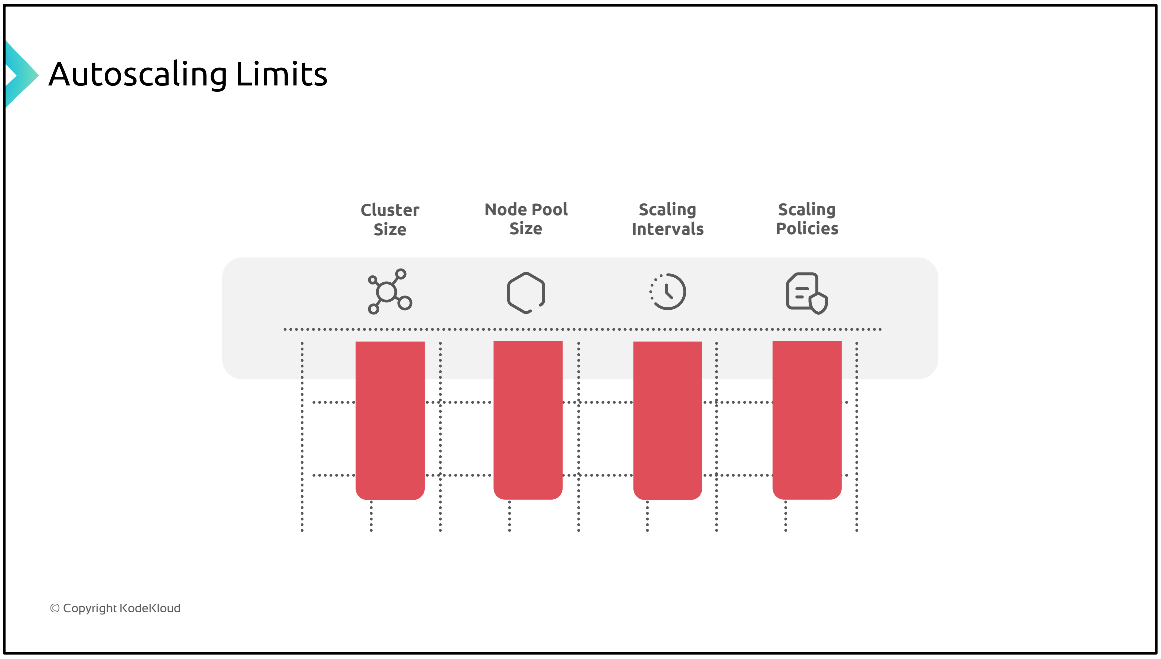
- CPU, memory, and storage resource requests
- Pending pods' node affinities and label selectors
- Pending pods' node taints and tolerations



Some Key Benefits

Some key benefits of GKE autoscaler:

- Reduced costs: The GKE autoscaler can help you to reduce costs by automatically scaling your cluster up or down based on demand. This can help you to avoid overprovisioning, which can lead to wasted resources.
- Improved performance: The GKE autoscaler can help you to improve performance by ensuring that your cluster has the right number of nodes to meet the demands of your workload. This can help to reduce latency and improve application responsiveness.
- Increased reliability: The GKE autoscaler can help you to increase reliability by automatically scaling your cluster up or down in response to failures. This can help to ensure that your applications are always available.

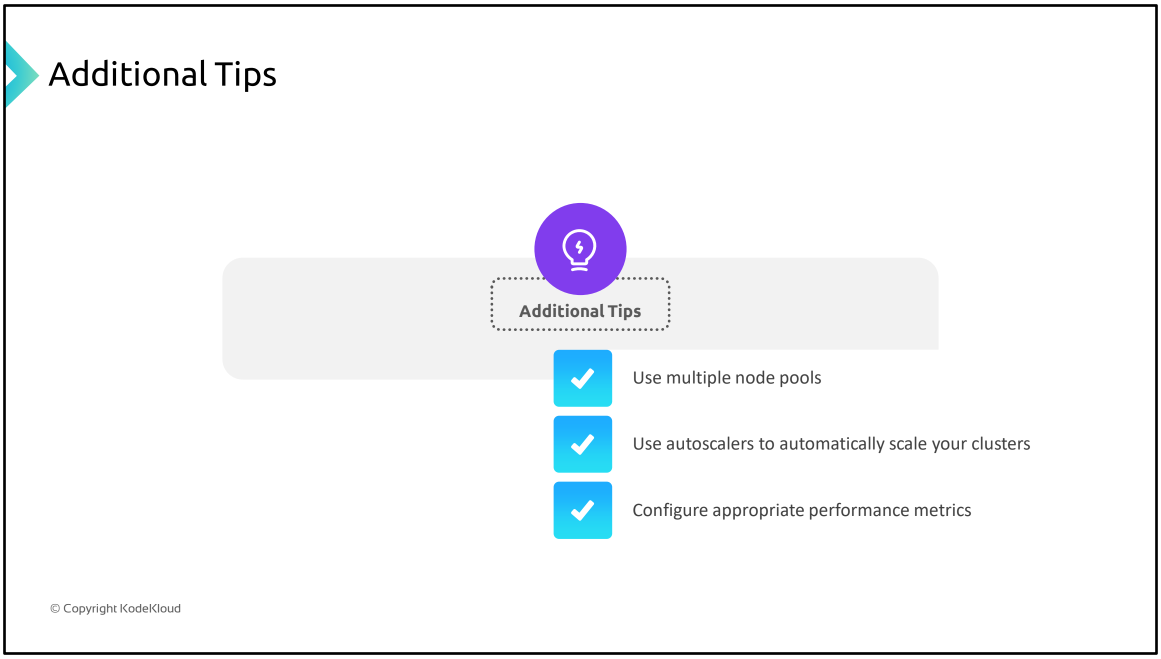


Autoscaling Limits

There are a few limitations that you should also be aware of while working with cluster autoscaling and node auto-provisioning:

- **Cluster size:** The maximum size of a GKE cluster is 15,000 nodes and 150,000 pods. However, there are other factors that can limit the size of your cluster, such as the amount of available resources in your project.
- **Node pool size**: The minimum and maximum number of nodes in a node pool is determined by the node machine type and the number of zones in which the node pool is deployed. For example, a node pool with a single zone and a machine type of n1-standard-1 can have a minimum of 1 and a maximum of 10 nodes.
- **Scaling intervals**: The cluster autoscaler scales your cluster at intervals of 15 seconds. This means that it can take up to 15 seconds for the cluster autoscaler to react to a change in load.
- **Scaling policies:** The cluster autoscaler uses a variety of policies to determine when to scale your cluster. These policies are based on factors such as the number of pods in your cluster, the CPU and memory utilization of your pods, and the availability of resources in your project.

It is important to be aware of these limitations when designing your GKE clusters. If you need to scale your cluster to a large size, you may need to use multiple node pools or autoscalers. You may also need to adjust the scaling policies to meet your specific needs.



Additional Tips

Additional Tips

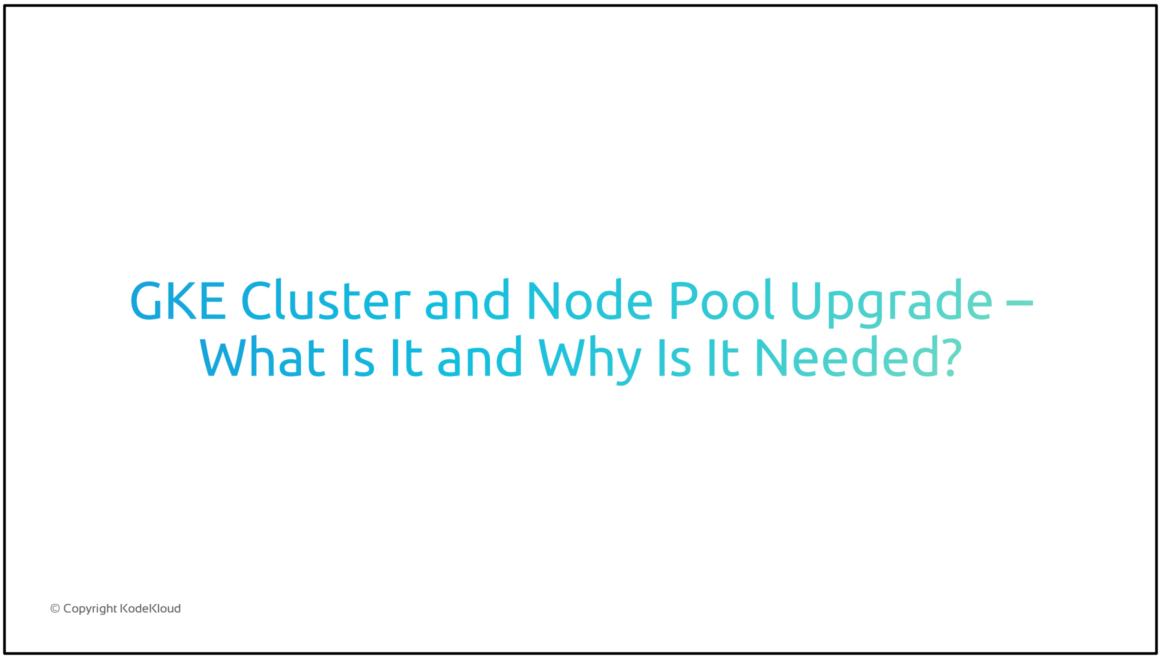
✓ Use multiple node pools

✓ Use autoscalers to automatically scale your clusters

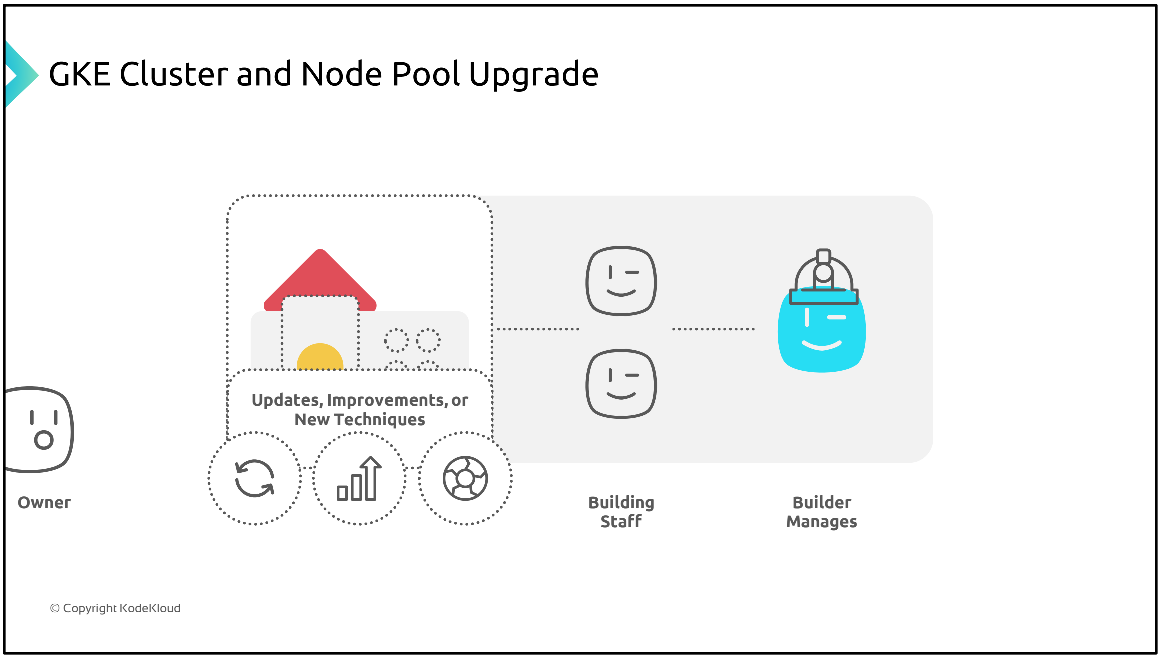
✓ Configure appropriate performance metrics

Here are some additional tips for scaling your GKE clusters:

- Use multiple node pools to spread your workloads across multiple machines in a production environment. This can help to improve performance and reliability.
- Use autoscalers to automatically scale your clusters up and down based on demand. This can help you to save money and resources.
- Configure appropriate performance metrics to monitor your clusters closely to ensure that they are performing as expected. This will help you to identify any potential issues before they cause problems.



GKE Cluster and Node Pool Upgrade – What Is It and Why Is It Needed?



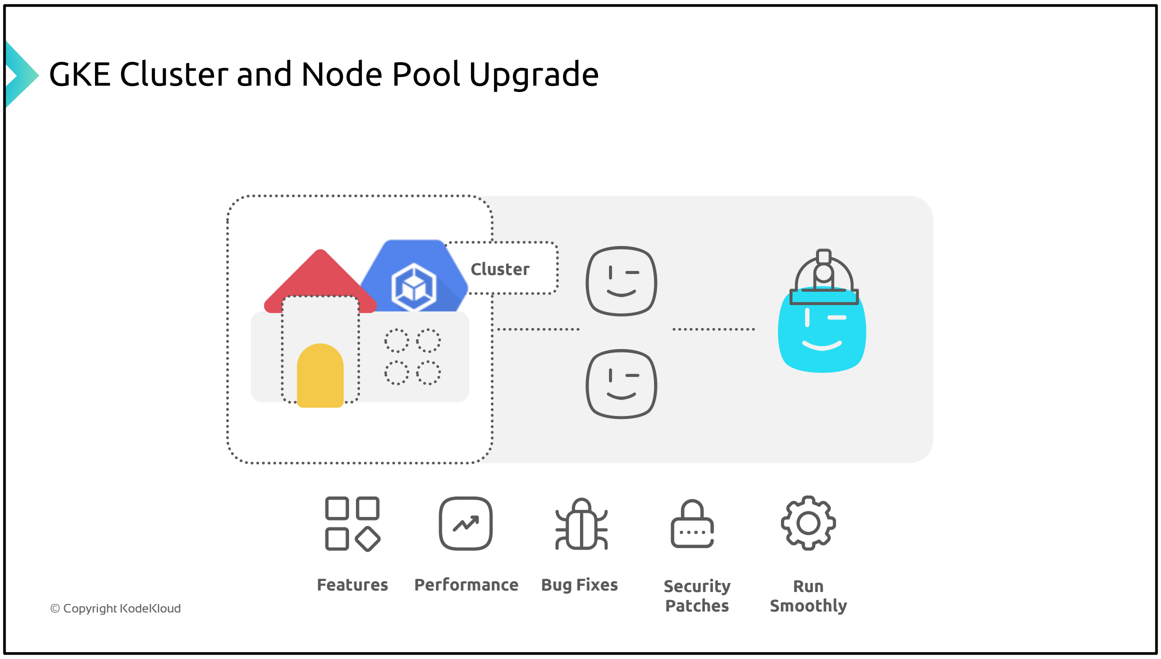
GKE Cluster and Node Pool Upgrade

In the context of the builder analogy, let's explore how upgrades work in Google Kubernetes Engine (GKE) clusters and node pools.

Let’s imagine as the construction of the house progresses, there are updates, improvements, or new techniques that become available. Things like security cameras, alarm systems, and so on. As the owner of the house, you want to ensure that these updates are incorporated into the construction process to enhance the quality and functionality of the house.

https://kodekloud.com/

https://kodekloud.com/courses/gke-google-kubernetes-engine/



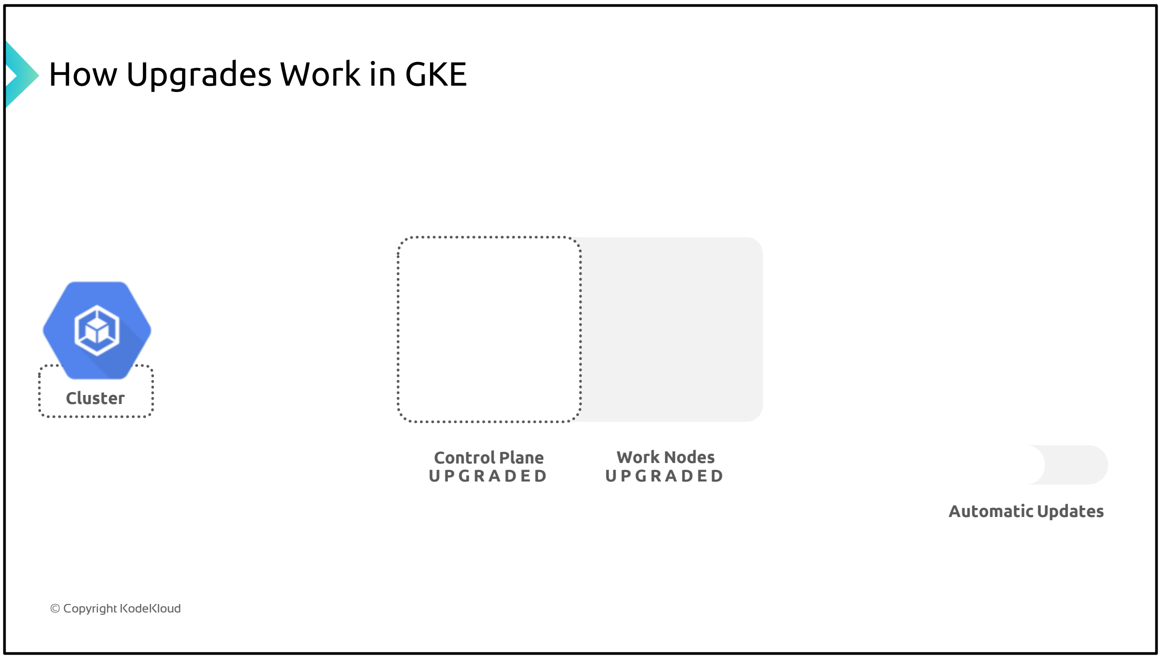
GKE Cluster and Node Pool Upgrade

Similarly, in GKE, a cluster is a representation of the house, while a node pool can be considered like a specific group of rooms within the house.

By incorporating upgrades into your GKE clusters and node pools, you can benefit from the latest **features**, **performance** improvements, **bug** fixes, and **security** patches provided by the Google Kubernetes Engine ecosystem. GKE's automated upgrade processes minimize disruptions, ensuring that your applications continue to run **smoothly** while maintaining high availability.

https://kodekloud.com/

https://kodekloud.com/courses/gke-google-kubernetes-engine/



How Upgrades Work in GKE

Cluster

Control Plane
UPGRADED

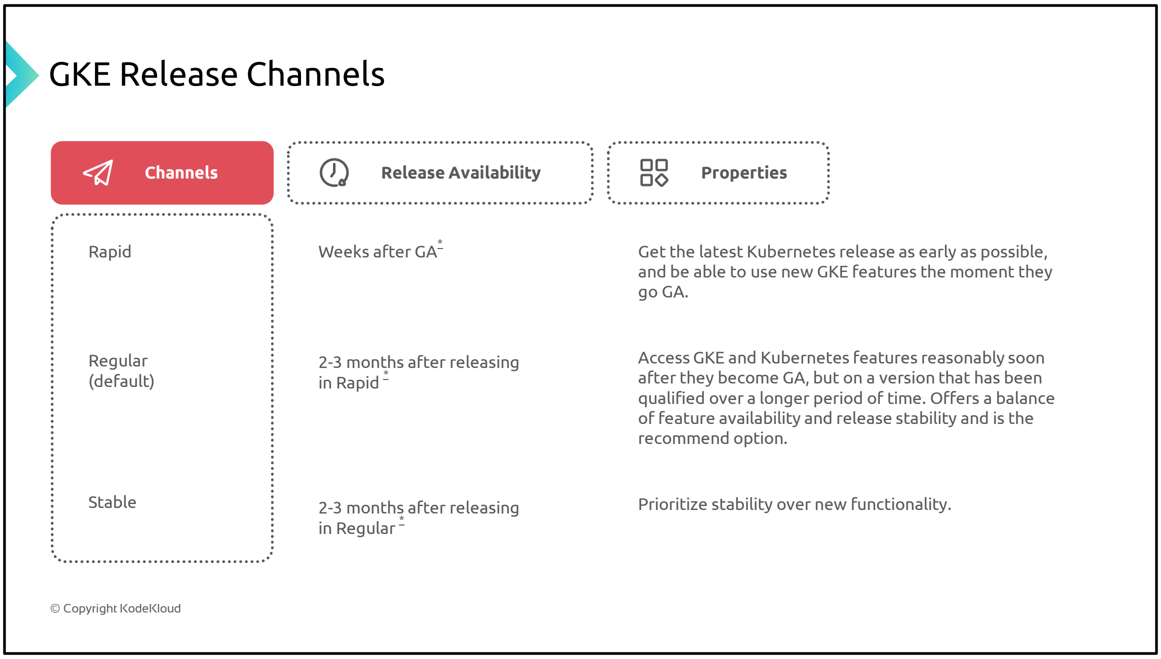
Work Nodes
UPGRADED

Automatic Updates

Upgrade in a GKE cluster happens in two phases; first, the control plane is upgraded and then the worker nodes are upgraded to match the version of the control plane.

By default, automatic upgrades are enabled for Google Kubernetes Engine (GKE) clusters and for GKE Standard node pools. However, GKE allows you to manually request an upgrade or downgrade for the control plane or nodes of a GKE cluster.

Now before we take a deeper look at the upgrade process for a GLE cluster, let’s first understand what a release channel is.



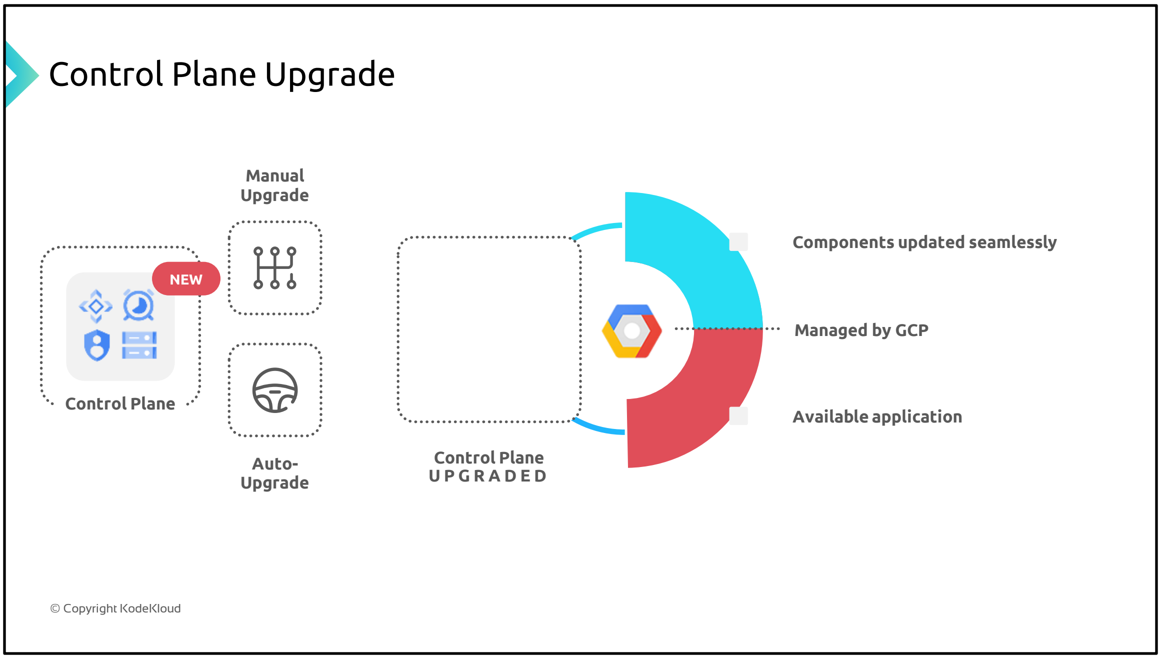
GKE Release Channels

Channels	Release Availability	Properties
Rapid	Weeks after GA*	Get the latest Kubernetes release as early as possible, and be able to use new GKE features the moment they go GA.
Regular (default)	2-3 months after releasing in Rapid*	Access GKE and Kubernetes features reasonably soon after they become GA, but on a version that has been qualified over a longer period of time. Offers a balance of feature availability and release stability and is the recommend option.
Stable	2-3 months after releasing in Regular*	Prioritize stability over new functionality.

Release channels implement Google Kubernetes Engine (GKE) best practices for versioning and upgrading a GKE cluster. There are three variants of release channel as show here:

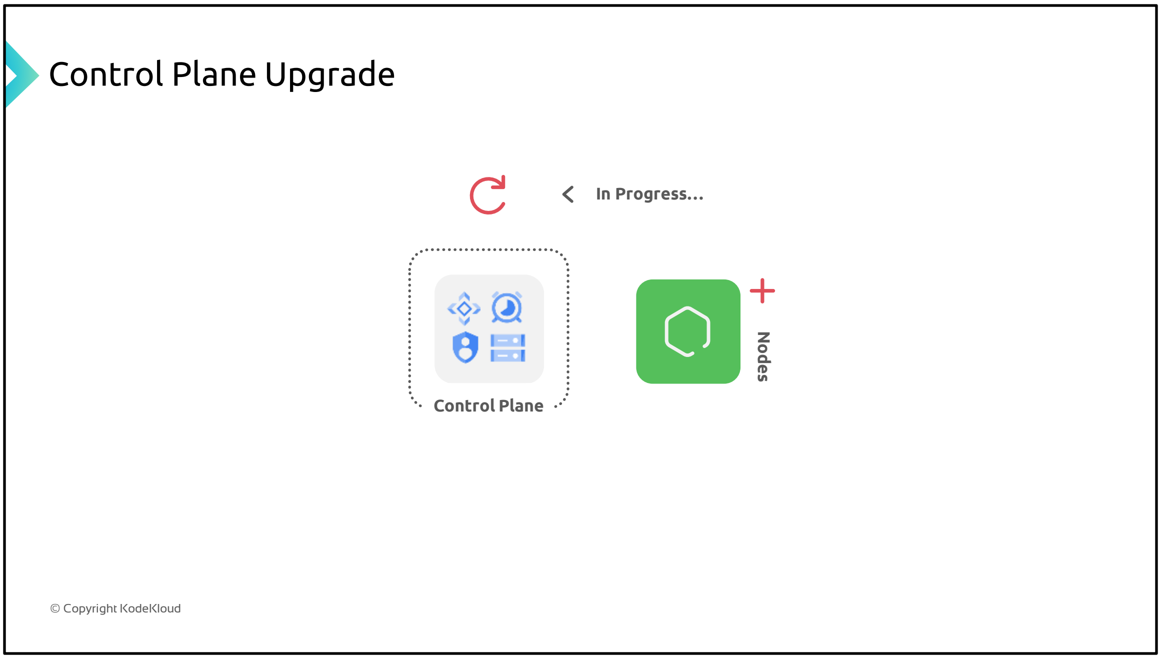
To learn more about release

channels, refer to the documentation : https://cloud.google.com/kubernetes-engine/docs/concepts/release-channels



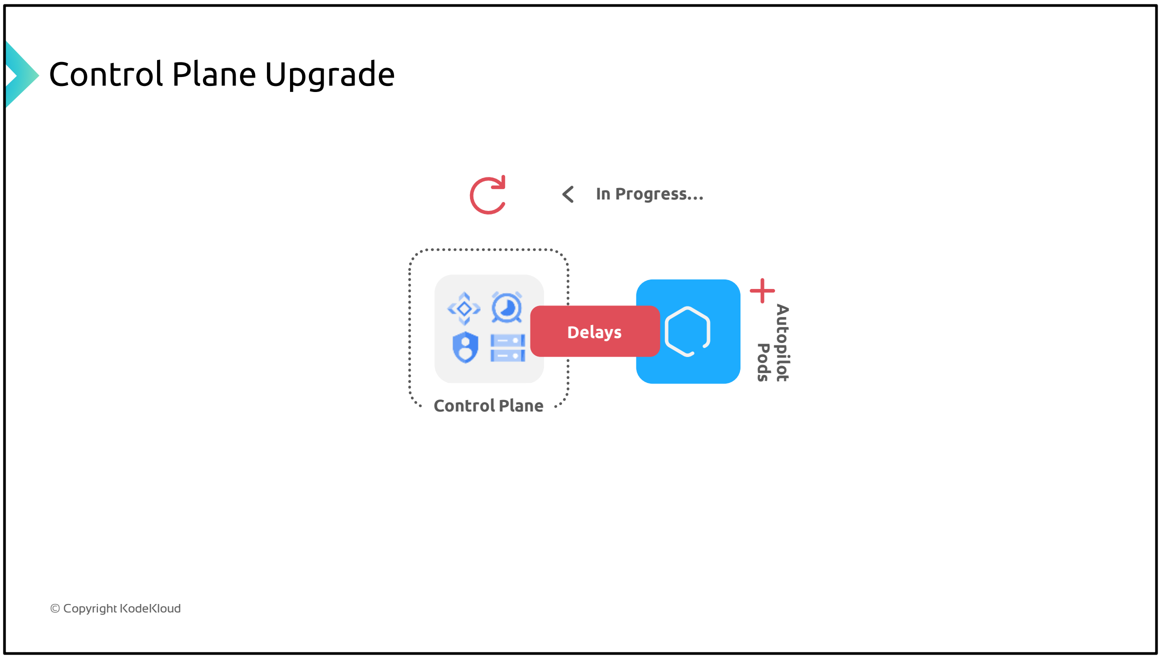
Control Plane Upgrade

When a new version of Kubernetes becomes available, you can either schedule a control plane upgrade manually or if auto-upgrade is enabled, GCP schedules an upgrade for the control plane of your cluster. During the upgrade process, GKE ensures that the control plane components are updated seamlessly, without affecting the availability of your applications. The control plane upgrade process is managed by GCP, and you typically don't need to take any manual actions.



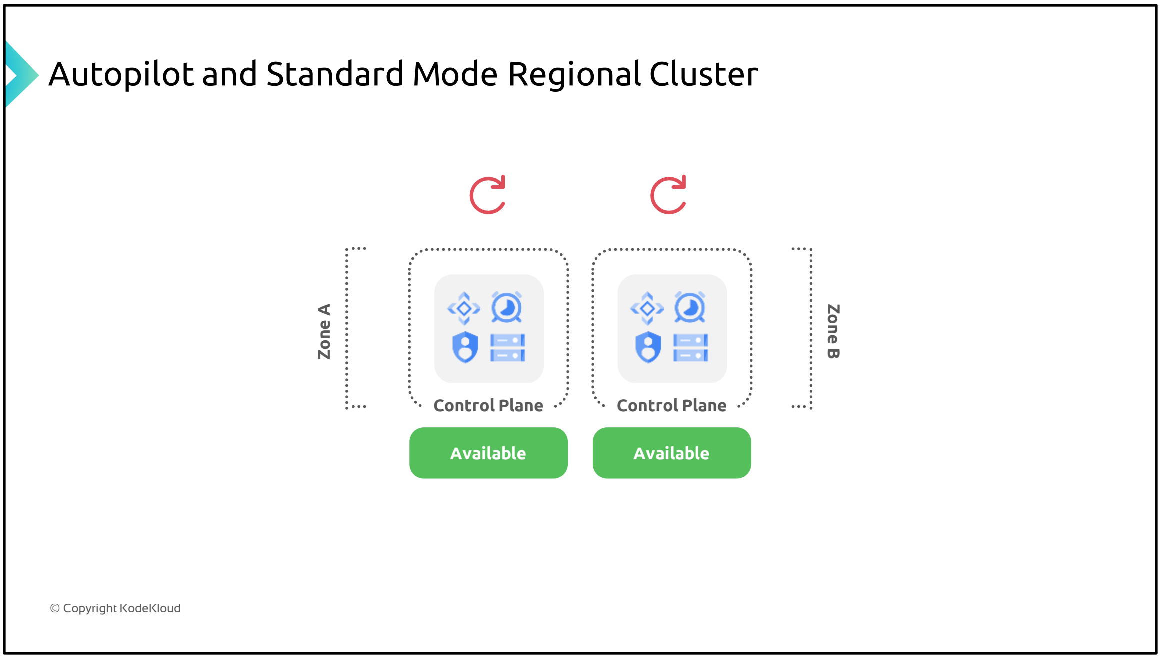
Control Plane Upgrade

Note: GKE can't create new nodes when a control plane upgrade is in progress in both Autopilot and Standard. If you deploy Autopilot Pods that require new node types while a control plane upgrade is in progress, you might experience delays until the control plane upgrade completes.



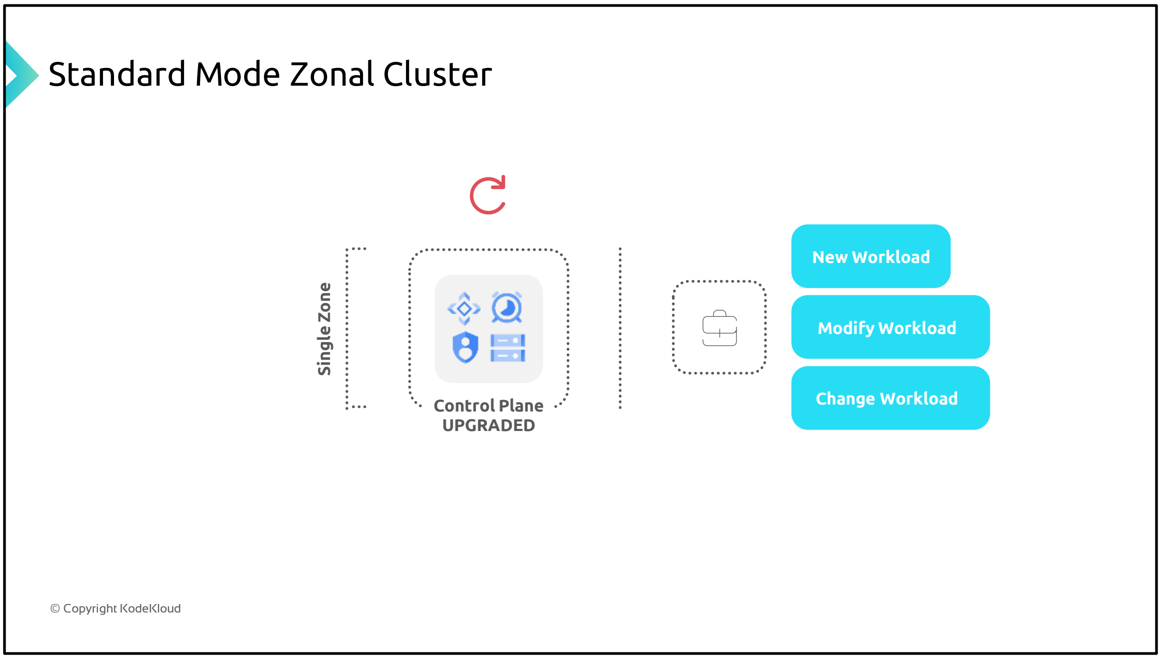
Control Plane Upgrade

Note: GKE can't create new nodes when a control plane upgrade is in progress in both Autopilot and Standard. If you deploy Autopilot Pods that require new node types while a control plane upgrade is in progress, you might experience delays until the control plane upgrade completes.



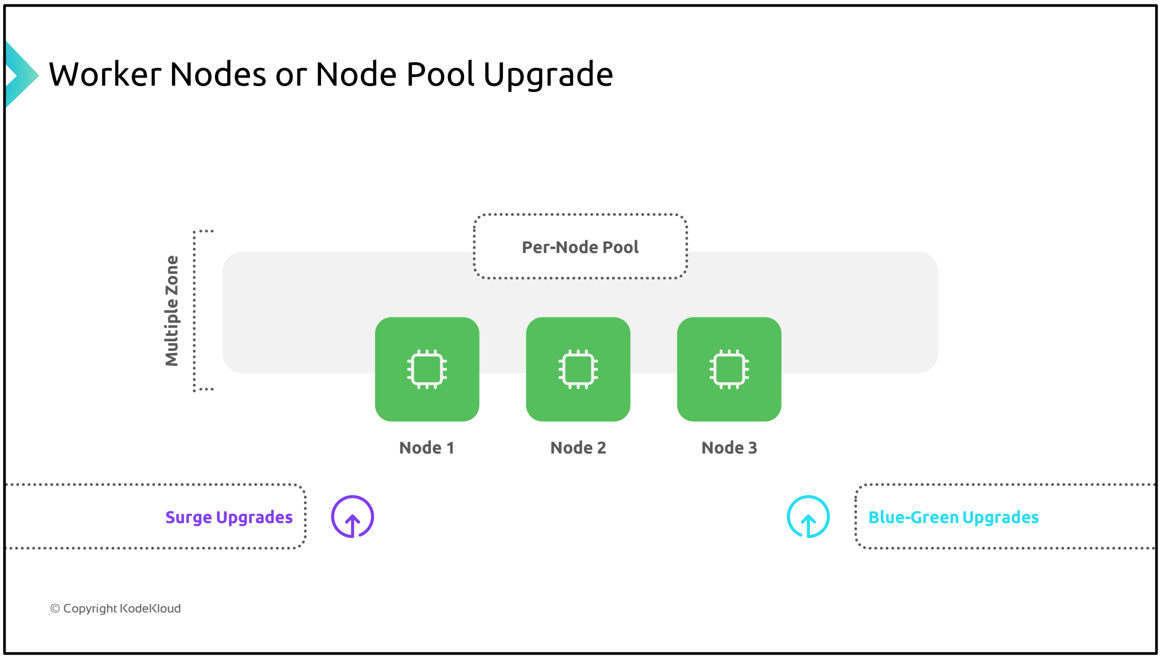
Autopilot and Standard Mode Regional Cluster

All Autopilot clusters are regional clusters and have multiple replicas of the control plane similar to Standard mode regional clusters. During the upgrade process, only one replica is upgraded at a time, in an undefined order. This ensures that the cluster remains highly available during automatic upgrades. Each control plane replica is only unavailable while the upgrade is in progress.



Standard Mode Zonal Cluster

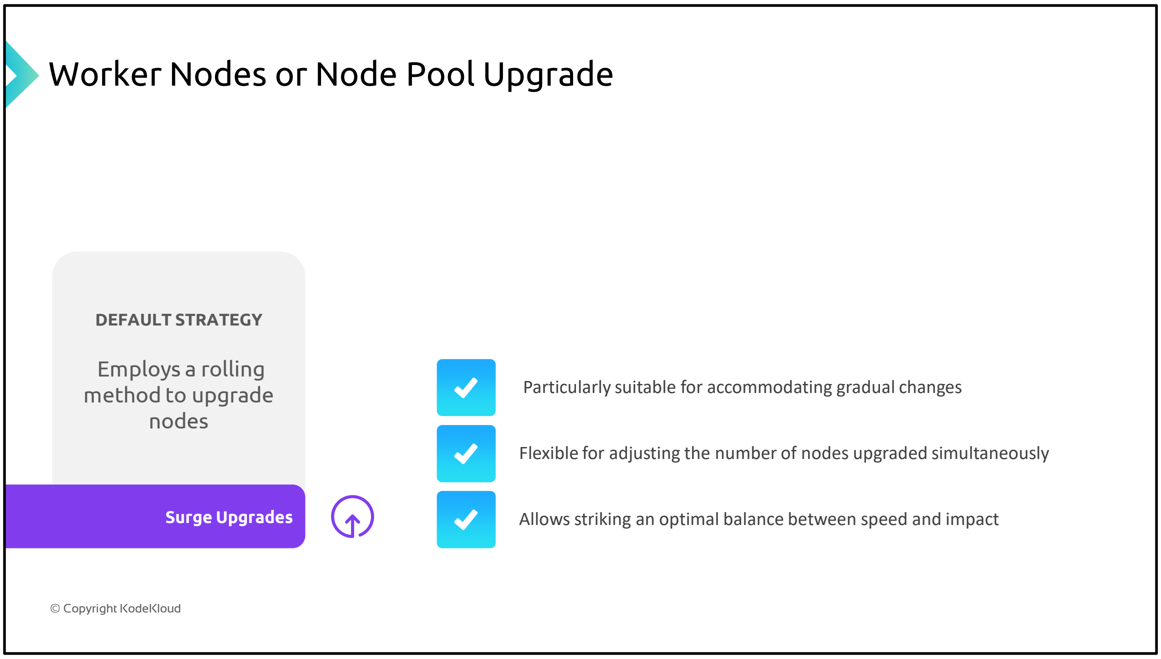
GKE Zonal clusters have only a single control plane. During the upgrade, the workloads running on the cluster continue to run, but new workloads cannot be deployed. You are also not able modify existing workloads, or make other changes to the cluster's configuration until the upgrade is complete.



Worker Nodes or Node Pool Upgrade

By default, nodes within a node pool are also upgraded one at a time, in an undefined order. If a node pool spread across multiple zones, upgrades take place zone by zone.
With GKE node pool upgrades, there are two configurable, built-in upgrade strategies that we can choose from and tune the upgrade process based on our cluster environment's needs.

- Surge Upgrades
- Blue Green upgrades



Worker Nodes or Node Pool Upgrade

DEFAULT STRATEGY

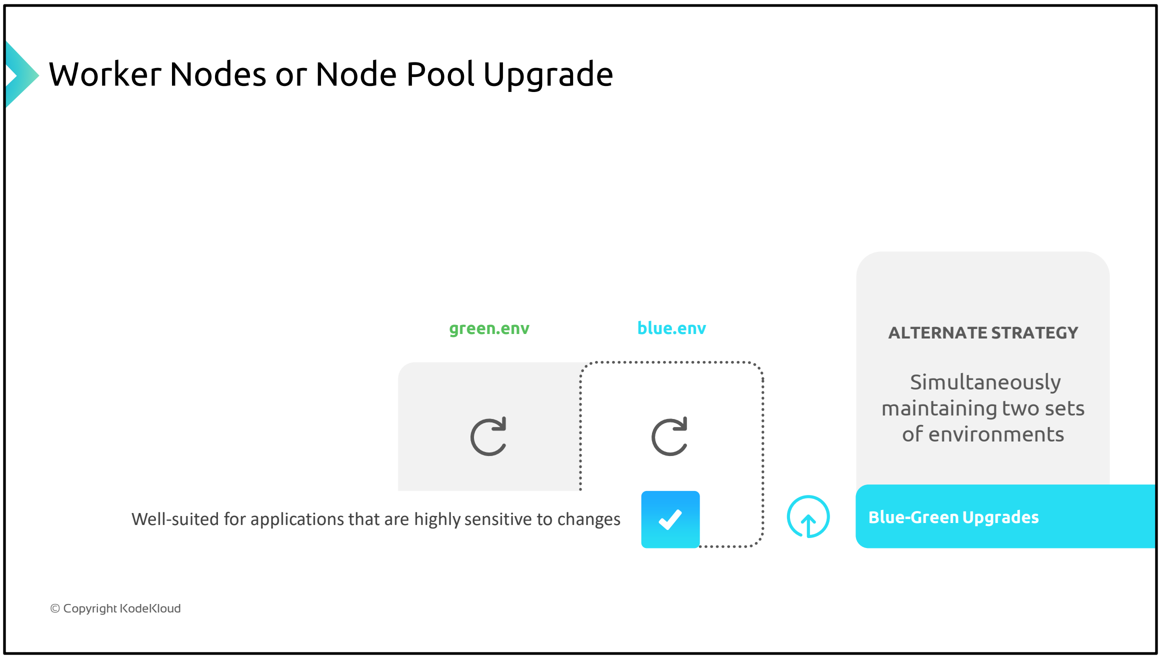
Employs a rolling method to upgrade nodes

Surge Upgrades

- ✓ Particularly suitable for accommodating gradual changes
- ✓ Flexible for adjusting the number of nodes upgraded simultaneously
- ✓ Allows striking an optimal balance between speed and impact

Surge upgrades

The default strategy for node pool upgrades is the surge upgrade strategy, which employs a rolling method to upgrade nodes. This strategy is particularly suitable for applications that can accommodate gradual and non-disruptive changes. With configurable settings, you have the flexibility to adjust the number of nodes upgraded simultaneously and the level of disruption caused by the upgrades. This allows us to strike an optimal balance between the speed of upgrades and the impact on your environment, aligning with environment specific requirements.



Worker Nodes or Node Pool Upgrade

ALTERNATE STRATEGY

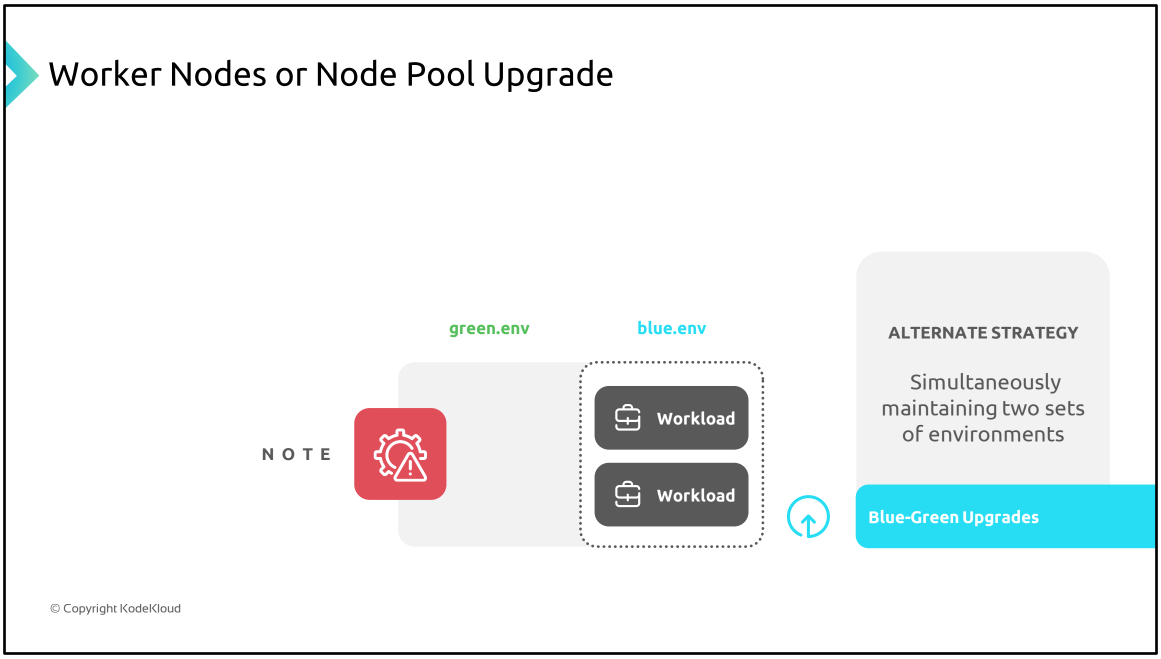
Simultaneously maintaining two sets of environments

Well-suited for applications that are highly sensitive to changes

Blue-Green Upgrades

Blue-green upgrades

The alternative approach is blue-green upgrades and involves simultaneously maintaining two sets of environments: the original environment called ("blue") and the new environment known as ("green"). This approach is designed to facilitate easy rollback if necessary. Blue-green upgrades are more resource intensive but are **well-suited for applications that are highly sensitive to changes**.



Worker Nodes or Node Pool Upgrade

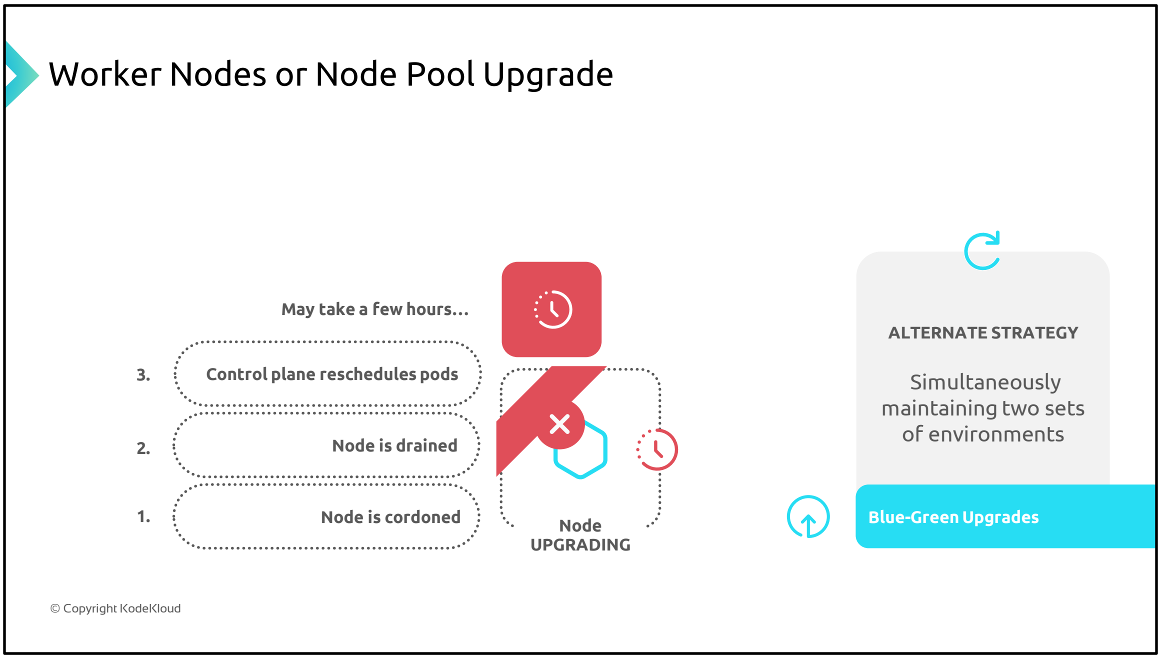
ALTERNATE STRATEGY

Simultaneously maintaining two sets of environments

Blue-Green Upgrades

In this strategy, workloads are gradually migrated from the original "blue" environment to the new "green" environment. If required, the workloads can be swiftly rolled back to the existing "blue" environment, ensuring flexibility and minimizing disruptions.

Please note that during a node pool upgrade, we cannot make changes to the cluster configuration unless the upgrade is cancelled. Also, Cluster Autoscaler scale-up events may still occur during a node pool upgrade. For a multi-zone node pool, a node may be created running the older node version if the zone's nodes have not yet been upgraded.



Worker Nodes or Node Pool Upgrade

May take a few hours...

3. Control plane reschedules pods

2. Node is drained

1. Node is cordoned

Node
UPGRADING

ALTERNATE STRATEGY

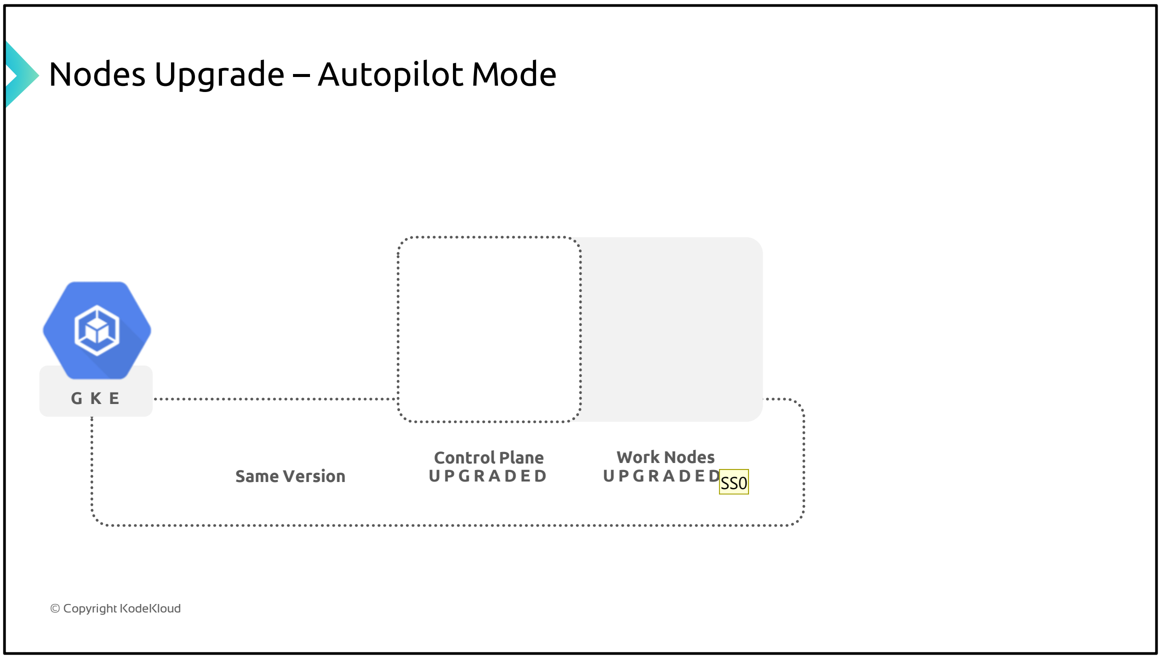
Simultaneously maintaining two sets of environments

Blue-Green Upgrades

To upgrade a node, GKE removes pods from the node so that it can be upgraded. So, when a node is being upgraded:

1. The node is cordoned so that Kubernetes does not schedule new pods on it.
2. The node is then drained, meaning that the pods are removed.
3. The control plane reschedules pods managed by controllers onto other nodes. Pods that cannot be rescheduled stay in the Pending phase until they can be rescheduled.

The node pool upgrade process may take up to a few hours depending on the upgrade strategy, the number of nodes, and their workload configurations.



Nodes Upgrade – Autopilot Mode

GKE

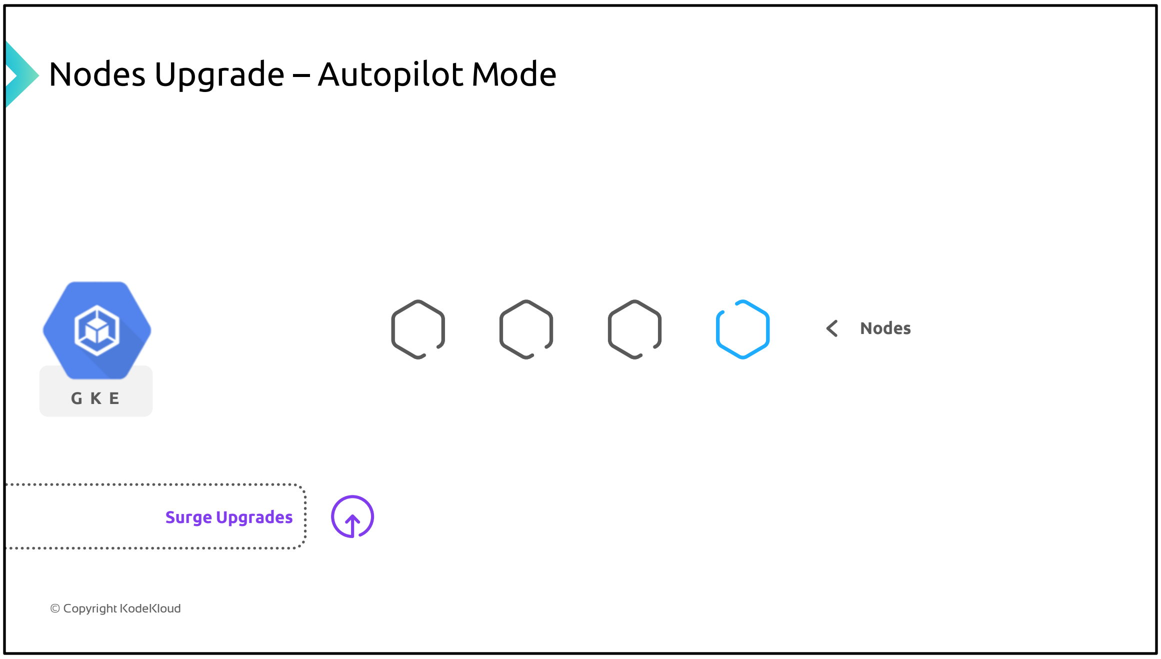
Same Version

Control Plane
UPGRADED

Work Nodes
UPGRADED

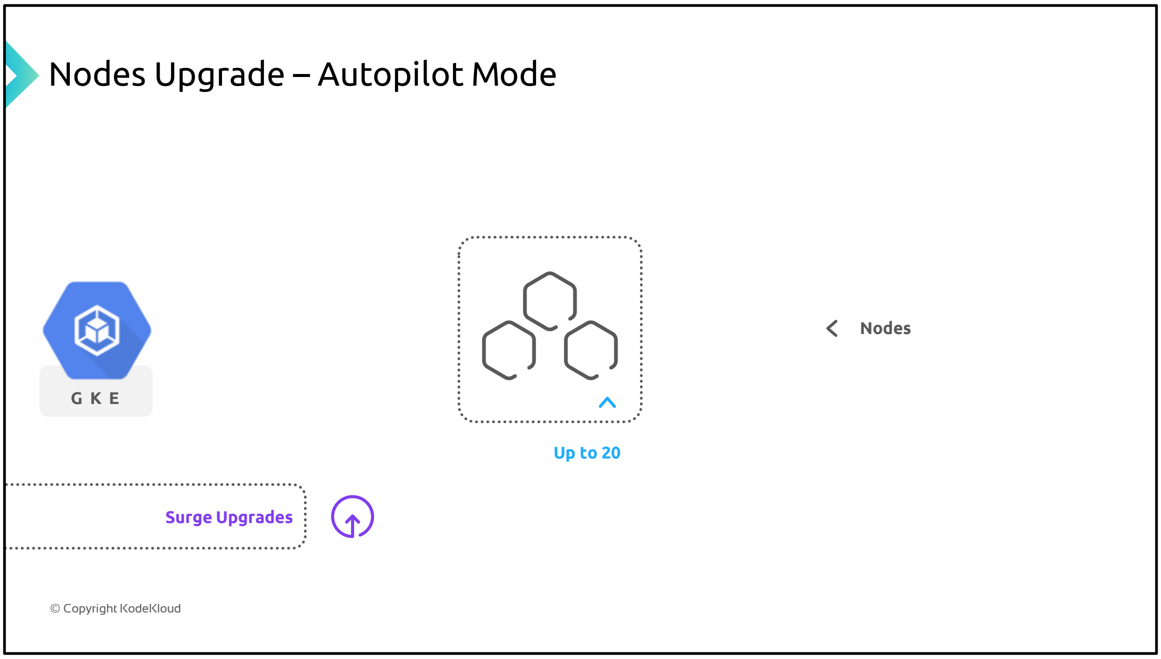
SS0

In Autopilot mode, Google Kubernetes Engine (GKE) provides automatic node upgrades to ensure that both the control plane and nodes are running on the same GKE version. After the control plane is upgraded, GKE upgrades the nodes to the same GKE version.



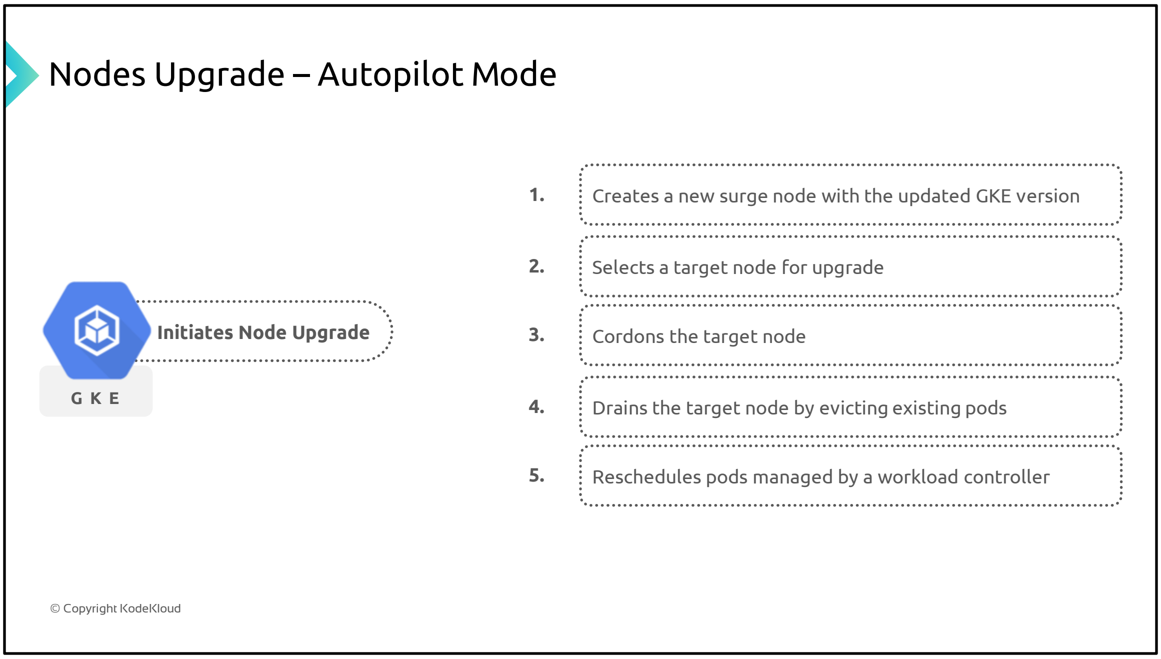
Nodes Upgrade – Autopilot Mode

To facilitate efficient upgrades, GKE groups nodes with similar characteristics together. Using surge upgrades, GKE can upgrade up to 20 nodes in a group simultaneously, with the actual number varying to maintain high availability.



Nodes Upgrade – Autopilot Mode

To facilitate efficient upgrades, GKE groups nodes with similar characteristics together. Using surge upgrades, GKE can upgrade up to 20 nodes in a group simultaneously, with the actual number varying to maintain high availability.



Nodes Upgrade – Autopilot Mode

GKE

Initiates Node Upgrade

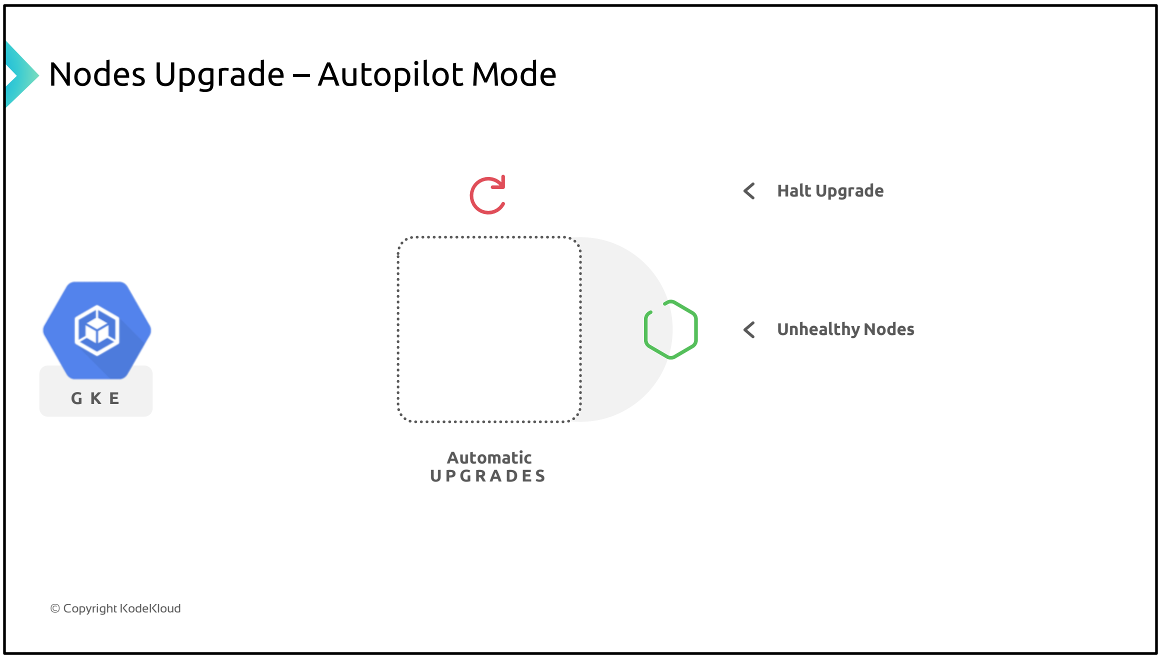
1. Creates a new surge node with the updated GKE version
2. Selects a target node for upgrade
3. Cordons the target node
4. Drains the target node by evicting existing pods
5. Reschedules pods managed by a workload controller

When GKE initiates a node upgrade, it follows a series of steps.

- First, it creates a new surge node with the updated GKE version and waits for it to register with the control plane.
- Next, it selects a target node for upgrade,
- Cordons the target node to prevent new pods from being scheduled on it, and
- Then drains the target node by evicting existing pods.
- GKE then reschedules pods managed by a workload controller onto other available nodes.

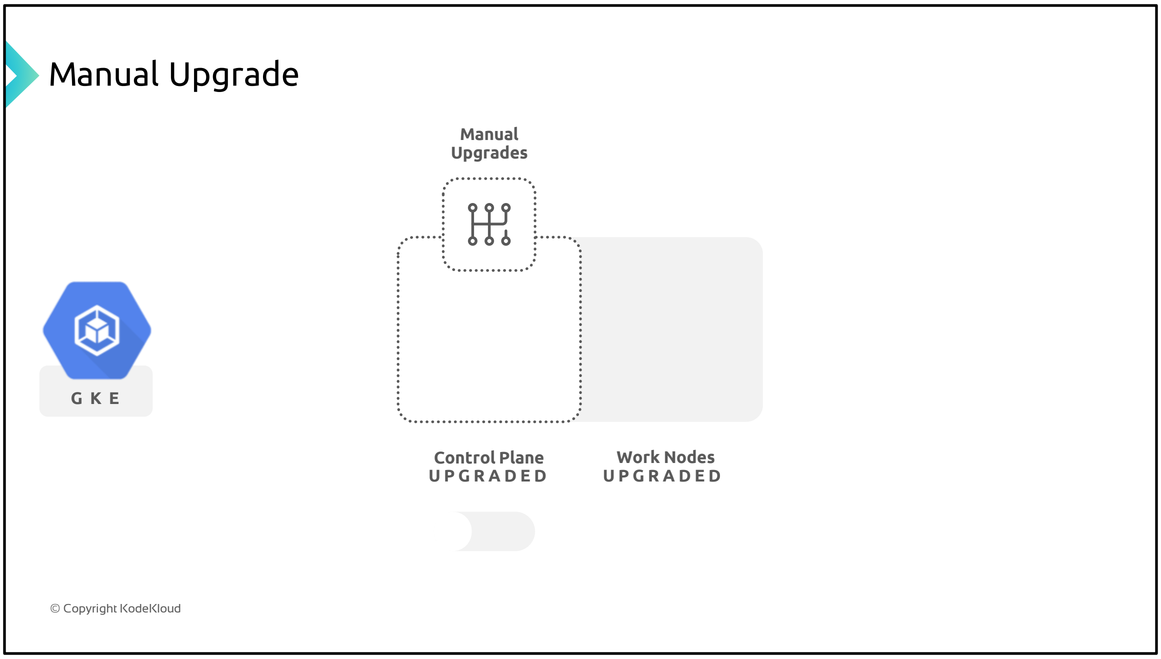
However, if a pod cannot be immediately rescheduled, it remains in a PENDING state until GKE can find a suitable replacement node.

- It's important to note that if the target node contains static pods, those pods will not be rescheduled and will be deleted along with the target node.



Nodes Upgrade – Autopilot Mode

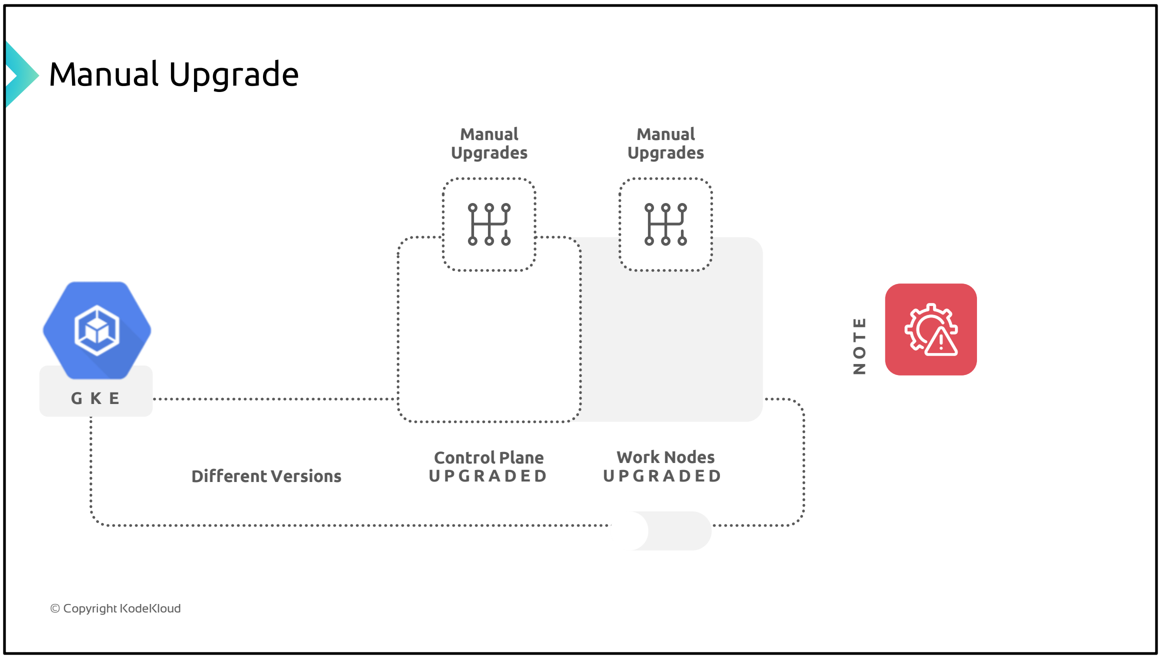
In cases where a significant number of automatic upgrades to a particular GKE version result in unhealthy nodes across the GKE fleet, GKE temporarily halts upgrades to that version to investigate the issue and ensure the stability of the cluster.



Manual Upgrade

By default, automatic upgrades are enabled for Google Kubernetes Engine (GKE) clusters and for GKE Standard node pools. However, we can also manually request an upgrade or downgrade for the control plane or nodes of a GKE cluster as follows:

•**Autopilot: Only the control plane version can be manually upgraded:** The GKE version of an Autopilot cluster control plane can be manually configured. If the control plane version that the cluster is being upgraded to is the default version in the release channel, GKE automatically upgrades its nodes to match the control plane version. However, If the control plane version is not the default version in the release channel, Autopilot waits until the version becomes the default before upgrading your nodes.



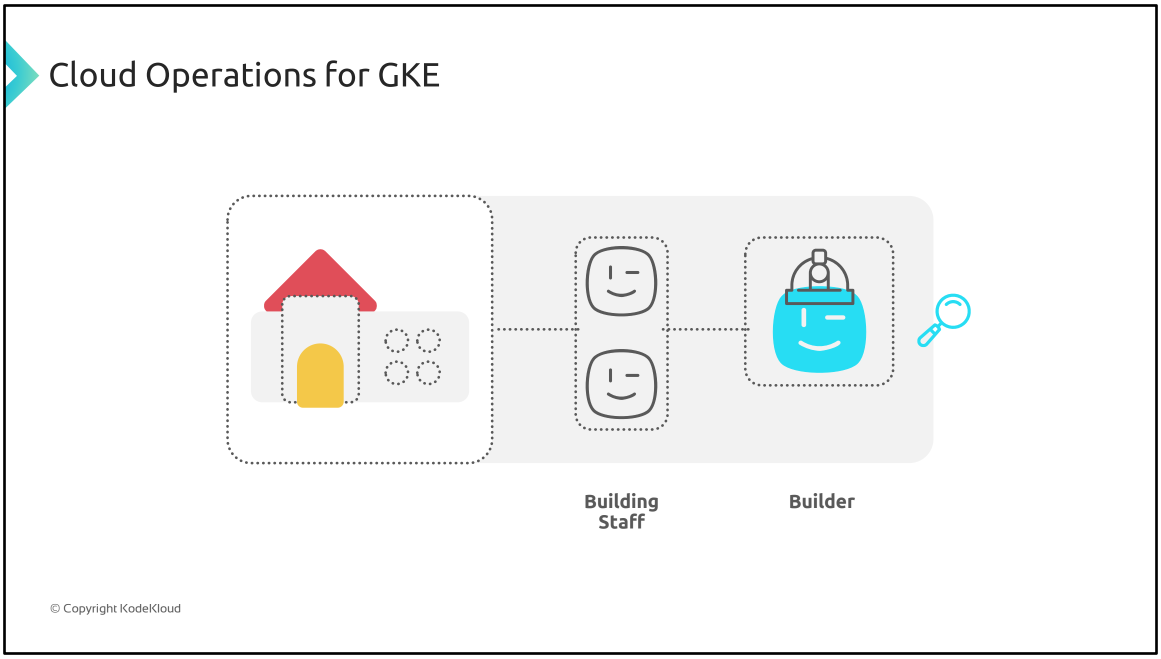
Manual Upgrade

© Copyright KodeKloud

Standard: Upgrade the control plane version and the node pool version: In a Standard cluster, we can upgrade both control plane and node pool versions manually. Node auto-upgrade is enabled by default. Although it is not recommended, the node auto-upgrade can be disabled if needed. If you opt out of node auto-upgrades, it becomes your responsibility to ensure that the cluster's nodes run a version compatible with the cluster’s control plane version, and that the version adheres to the Kubernetes version skew support policy. A cluster's node pools can be no more than two minor versions behind the control plane version, to maintain compatibility with the cluster API.

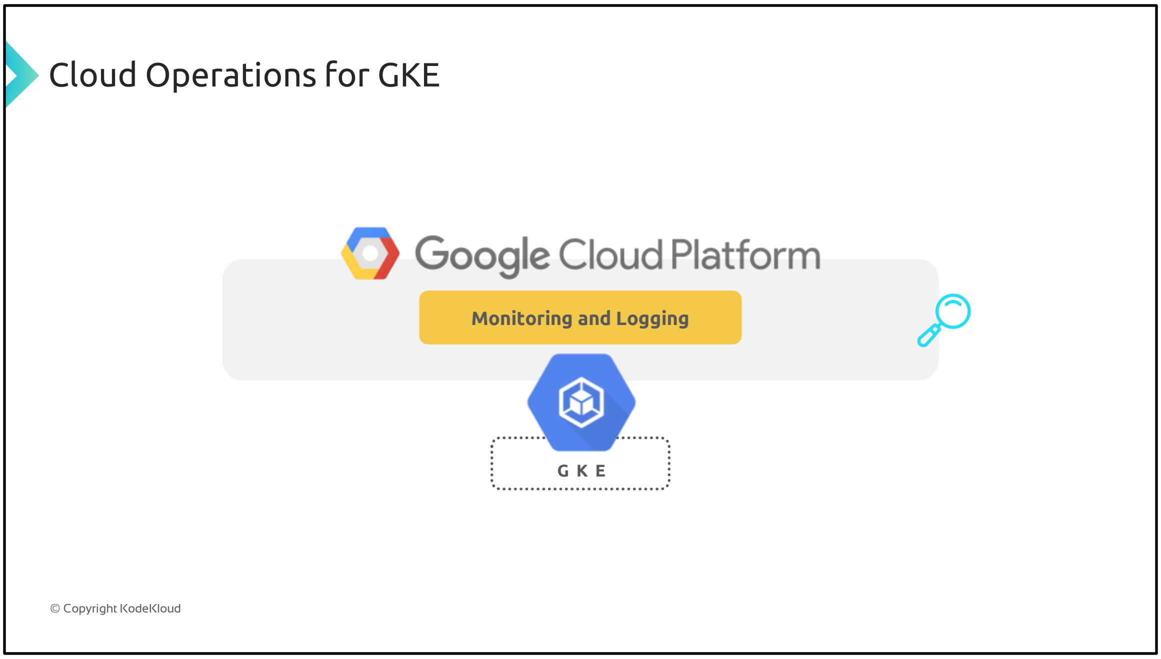
Note that a cluster's control plane and nodes do not necessarily run the same version at all times and the two versions can differ from time to time. Control planes are upgraded on a regular basis, and unlike node pool upgrades in standard clusters, it cannot be disabled. However, you can apply maintenance windows and exclusions to temporarily suspend upgrades for control planes and nodes.

Cloud Operations for GKE



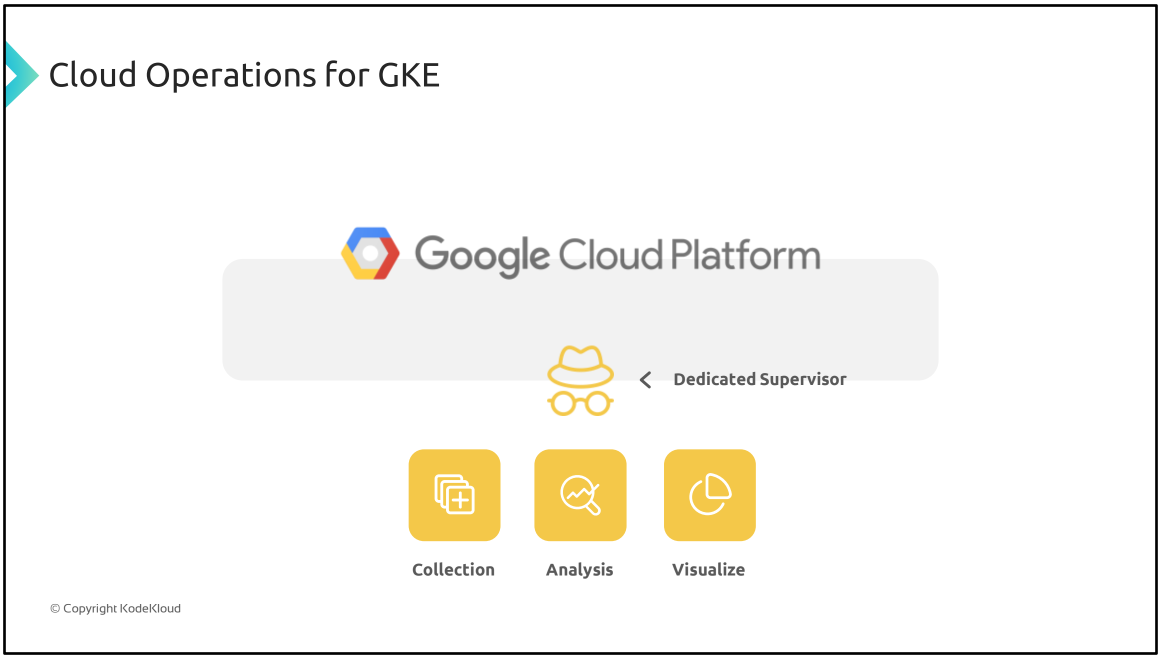
Cloud Operations for GKE

Let’s try to understand Cloud operation for GKE using our builder analogy. During the construction project, each tradie has their specific role, but to ensure the success of the project, the builder needs a way to monitor their progress and identify any issues that may arise.



Cloud Operations for GKE

This is where monitoring and logging come into play in the context of Cloud Operations Suite for Google Kubernetes Engine (GKE).



Cloud Operations for GKE

Google Cloud Platform

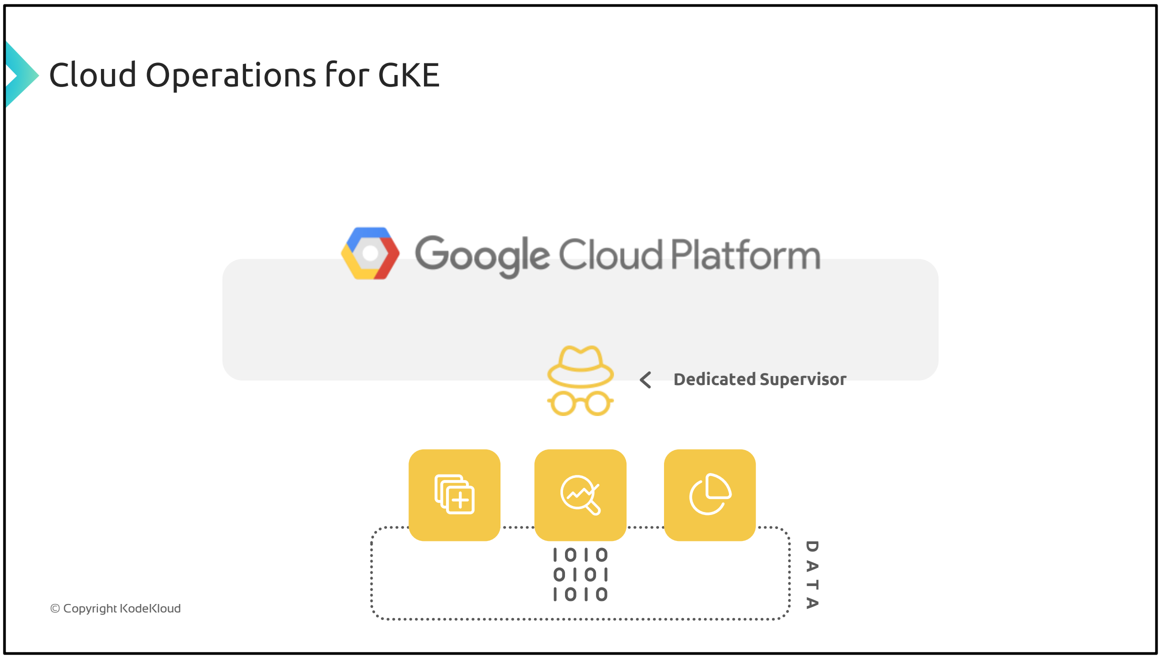
Dedicated Supervisor

Collection

Analysis

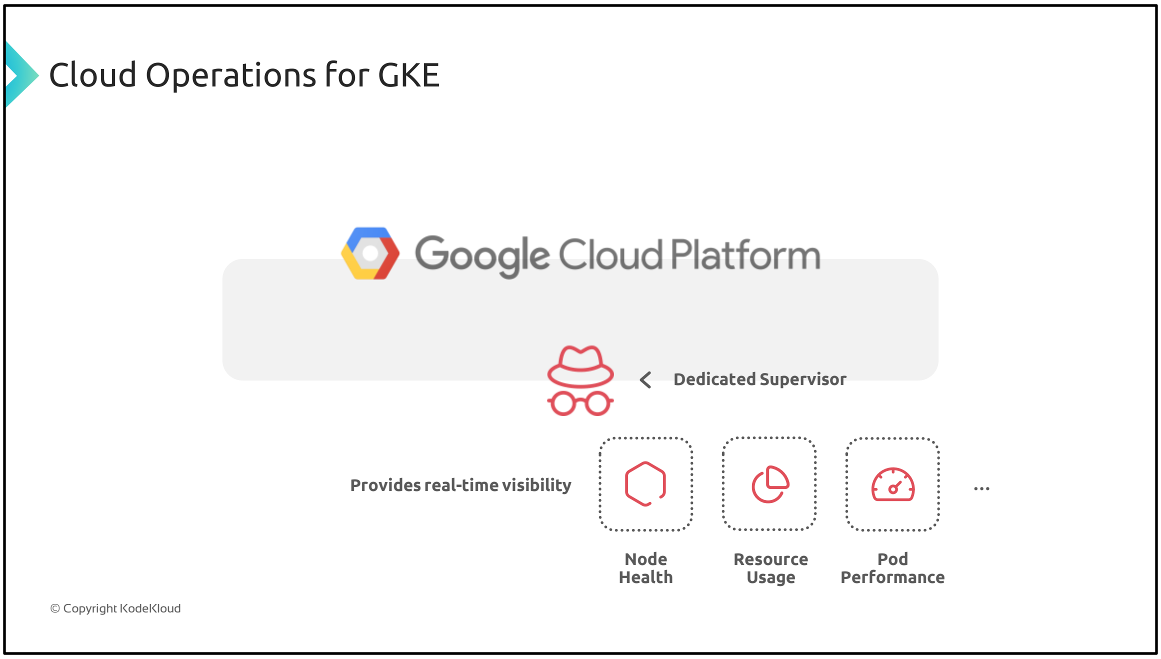
Visualize

Monitoring in Cloud Operations Suite for GKE is like having a dedicated supervisor overseeing the construction site. It involves the collection, analysis, and visualization of data related to your GKE cluster's performance, health, and resource utilization. Monitoring allows you to gain insights into the overall system behavior and identify any anomalies or potential problems.



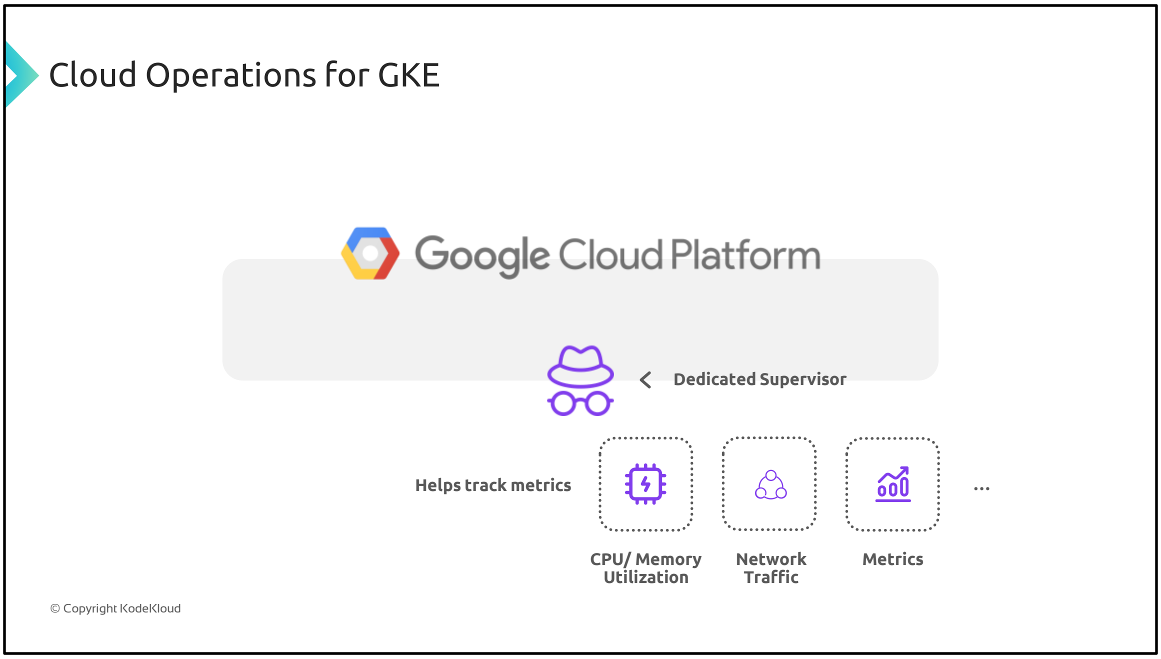
Cloud Operations for GKE

Monitoring in Cloud Operations Suite for GKE is like having a dedicated supervisor overseeing the construction site. It involves the collection, analysis, and visualization of data related to your GKE cluster's performance, health, and resource utilization. Monitoring allows you to gain insights into the overall system behavior and identify any anomalies or potential problems.



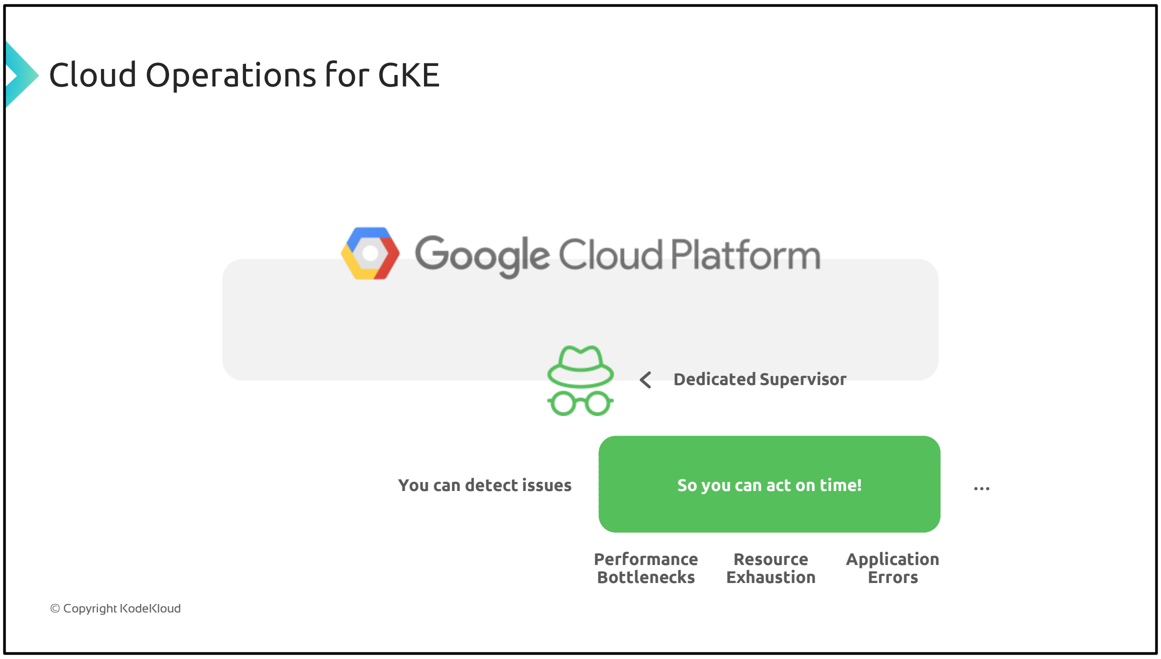
Cloud Operations for GKE

Similar to how the builder supervisor keeps an eye on tradies, GKE monitoring provides **real-time visibility** into various aspects of your cluster, including node health, resource usage, pod performance, and network connectivity.



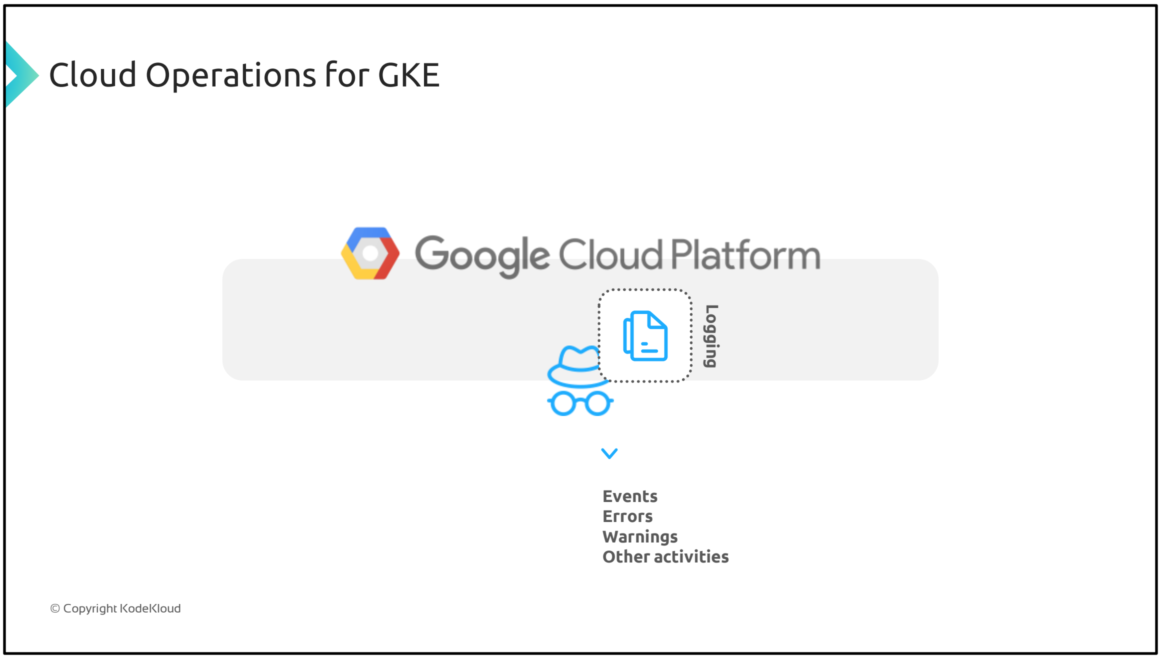
Cloud Operations for GKE

It helps you track metrics such as CPU and memory utilization, network traffic, and application-level metrics.



Cloud Operations for GKE

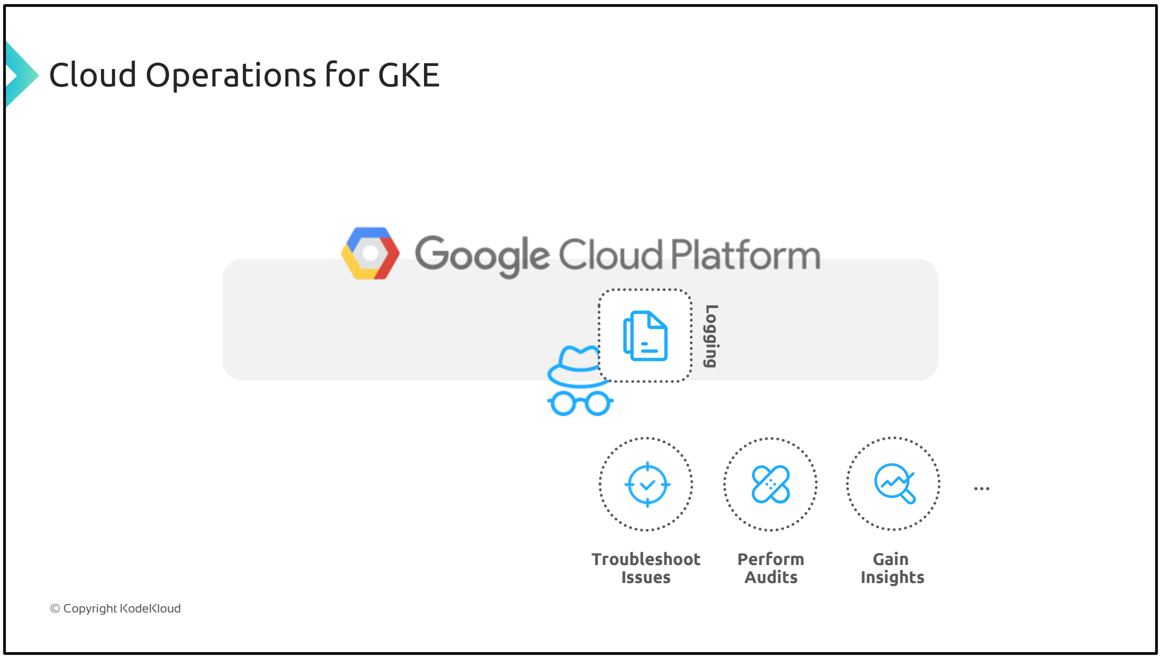
By monitoring these metrics, you can proactively detect issues like performance bottlenecks, resource exhaustion, or application errors, allowing you to take timely action before they impact your applications and users.



Cloud Operations for GKE

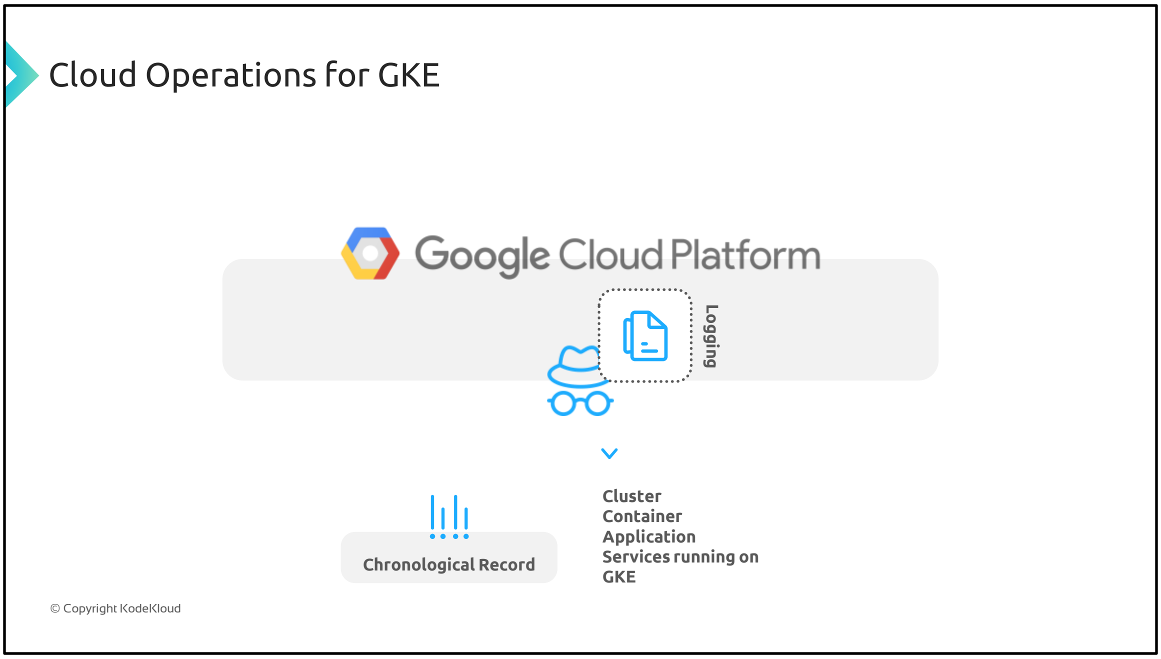
Events
Errors
Warnings
Other activities

Logging, on the other hand, is like keeping a detailed record of all activities happening on the construction site. In Cloud Operations Suite for GKE, logging involves capturing and storing relevant log data generated by your cluster and applications. Logs contain valuable information about events, errors, warnings, and other activities within the cluster,...



Cloud Operations for GKE

...allowing you to troubleshoot issues, perform audits, and gain insights into system behavior.

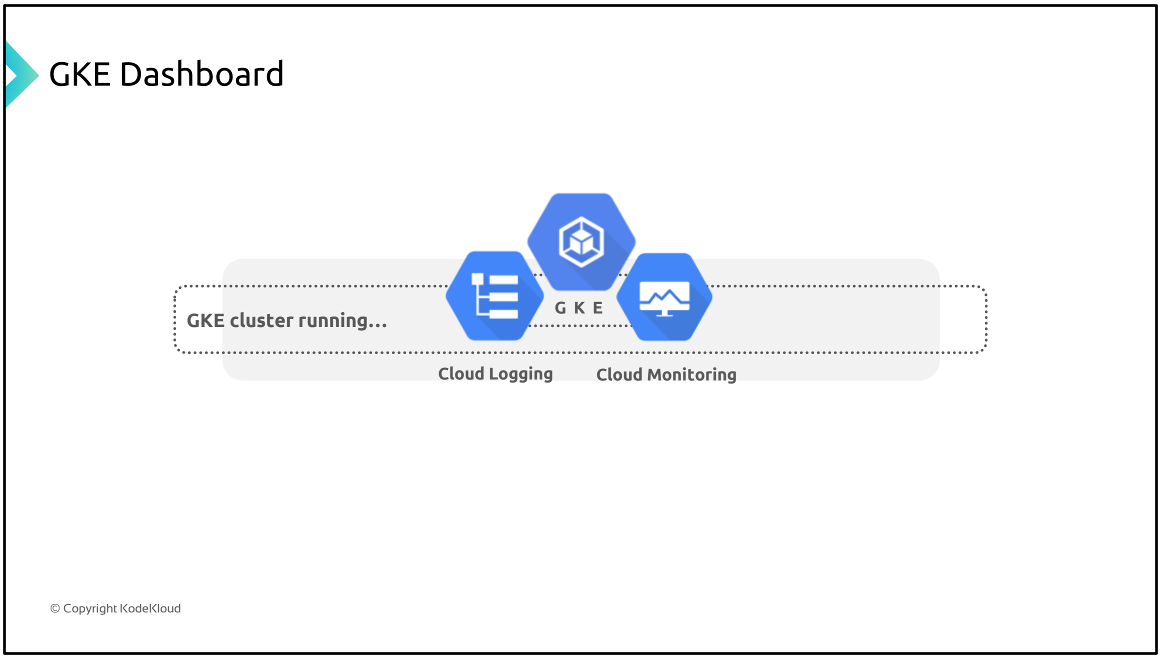


Cloud Operations for GKE

Cluster
Container
Application
Services running on
GKE

Chronological Record

Logs can be generated by various components, including the cluster itself, containers, applications, and services running on GKE. They provide a chronological record of events, helping you understand the sequence of actions and identify the root cause of any issues.



GKE Dashboard

GKE cluster running...

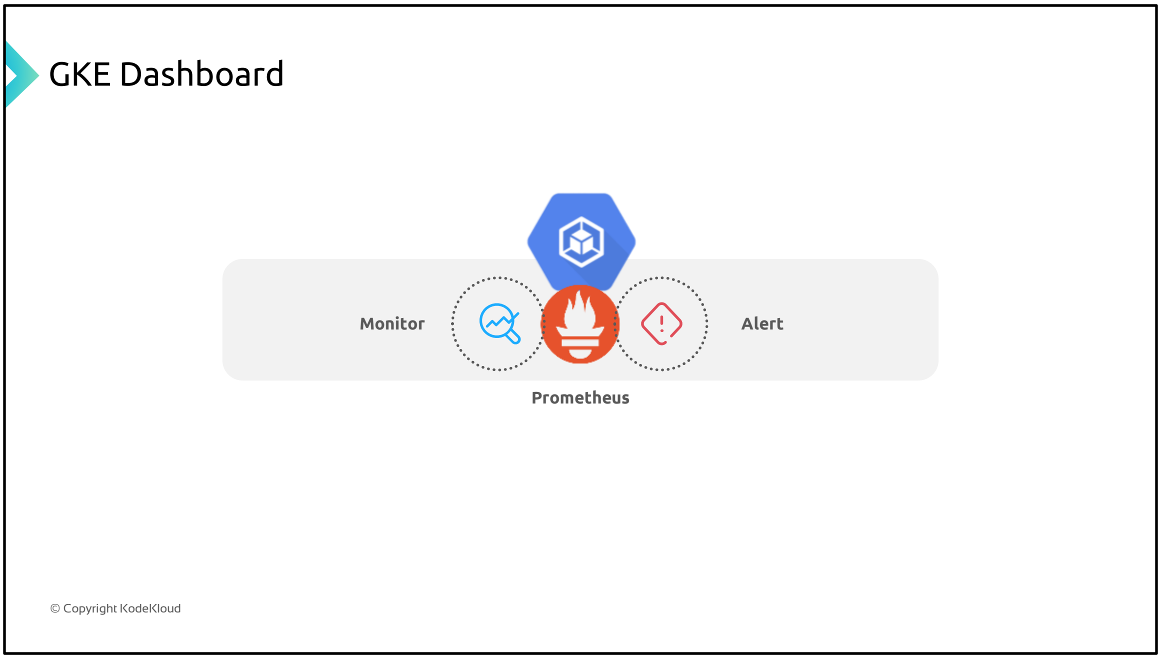
G K E

Cloud Logging

Cloud Monitoring

Google Kubernetes Engine (GKE) includes integration with Cloud Logging and Cloud Monitoring. When a GKE cluster is created and running on Google Cloud, Cloud Logging and Cloud Monitoring are enabled by default and provide observability specifically tailored for Kubernetes.

You can control which logs and metrics are sent from your GKE cluster to Cloud Logging and Cloud Monitoring.



GKE Dashboard

Google Kubernetes Engine (GKE) also provides integration with Google Cloud Managed Service for Prometheus. It can be manually enabled if needed and allows monitoring and alerting on the workloads, using Prometheus, without having to manually manage and operate Prometheus at scale. See Configuring Cloud Operations for GKE for installation and configuration instructions.