Proprietary & Confidential

Google Cloud 實作工作坊：透過 GKE Autopilot 部署專屬於您的私人 AI 機器人服務

Denny Tsai
2024/10/24

Google Cloud

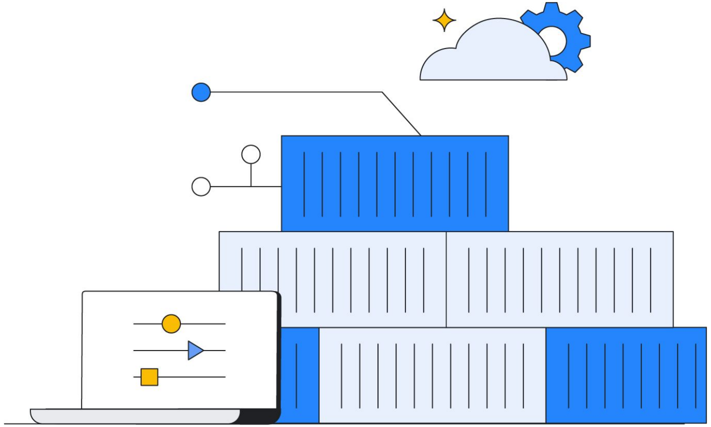
Material

https://dennygoog.gitlab.io/workshops/run-gemma-chatbot-on-gke-autopilot/

What is GKE Autopilot

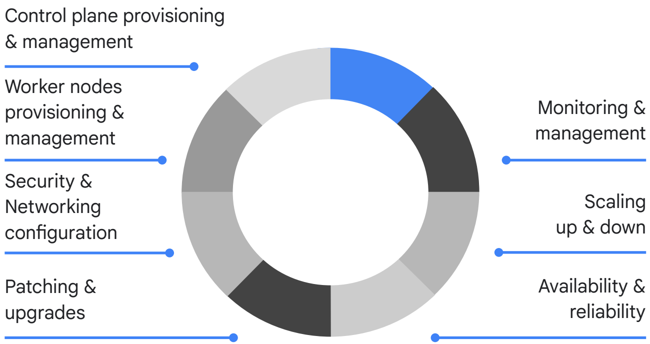
GKE Autopilot provides the most **fully automated, secure,** and **scalable** managed Kubernetes service based on **decades of experience** running containers at massive scale.

Focus on deploying **your workloads** and we'll take care of the rest.

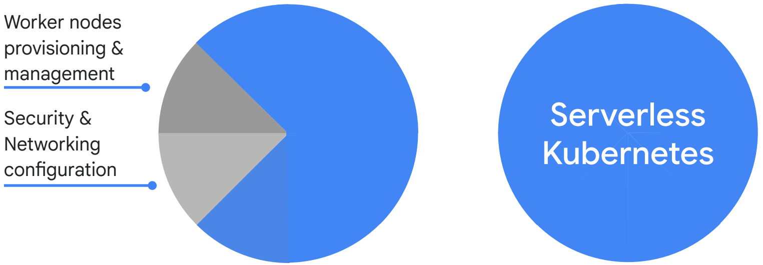


One GKE - two modes of operations

DIY Kubernetes Service


Control plane provisioning & management
Worker nodes provisioning & management
Security & Networking configuration
Patching & upgrades
Monitoring & management
Scaling up & down
Availability & reliability

Google Kubernetes Engine (GKE)


Worker nodes provisioning & management
Security & Networking configuration
Serverless Kubernetes

Standard mode
Managed Kubernetes with configuration flexibility

Autopilot mode
Optimized **Fully**-Managed Kubernetes, aka "Serverless Kubernetes"

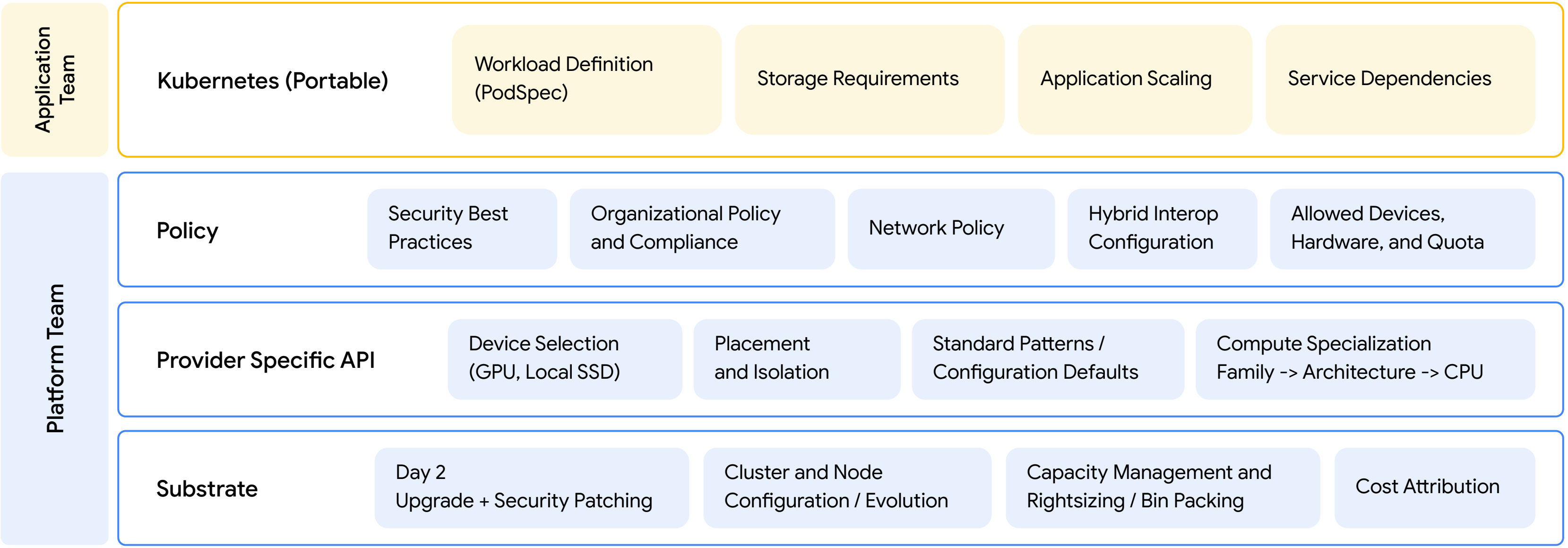
GKE Autopilot is a **mode of operation** in GKE
Mode of operation = level of control over a GKE cluster

Layers of a Kubernetes Platform

Application Team	Kubernetes (Portable)	Workload configuration using portable K8s API resources.
Platform Team	Policy	Policy controls managed by the platform team to meet security, compliance, reliability, and cost-efficiency requirements.
	Provider Specific API	The API, provided by the platform team, that allows more specialized workloads to consume provider-specific hardware and capabilities.
	Substrate	Provider-specific glue that translates application configuration to infrastructure constructs. Essential for operation, requires effort to maintain.

Layers of a Kubernetes Platform

To accommodate all but the simplest workloads, platform teams must also provide a layer of translation to expose provider specific capabilities necessary to fit advanced workload requirements.



Application Team

Kubernetes (Portable)
- Workload Definition (PodSpec)
- Storage Requirements
- Application Scaling
- Service Dependencies

Platform Team

Policy
- Security Best Practices
- Organizational Policy and Compliance
- Network Policy
- Hybrid Interop Configuration
- Allowed Devices, Hardware, and Quota

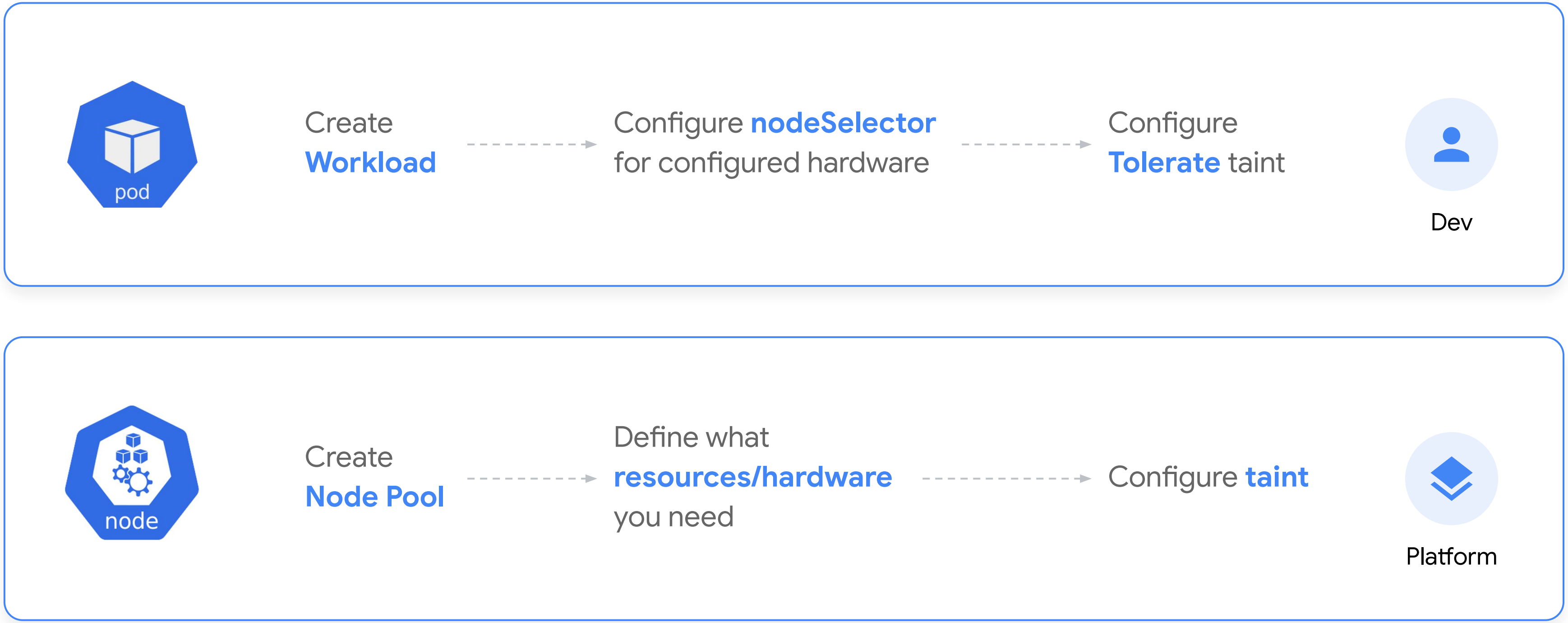
Provider Specific API
- Device Selection (GPU, Local SSD)
- Placement and Isolation
- Standard Patterns / Configuration Defaults
- Compute Specialization Family -> Architecture -> CPU

Substrate
- Day 2 Upgrade + Security Patching
- Cluster and Node Configuration / Evolution
- Capacity Management and Rightsizing / Bin Packing
- Cost Attribution

GKE Autopilot | Accelerator for Platform Teams

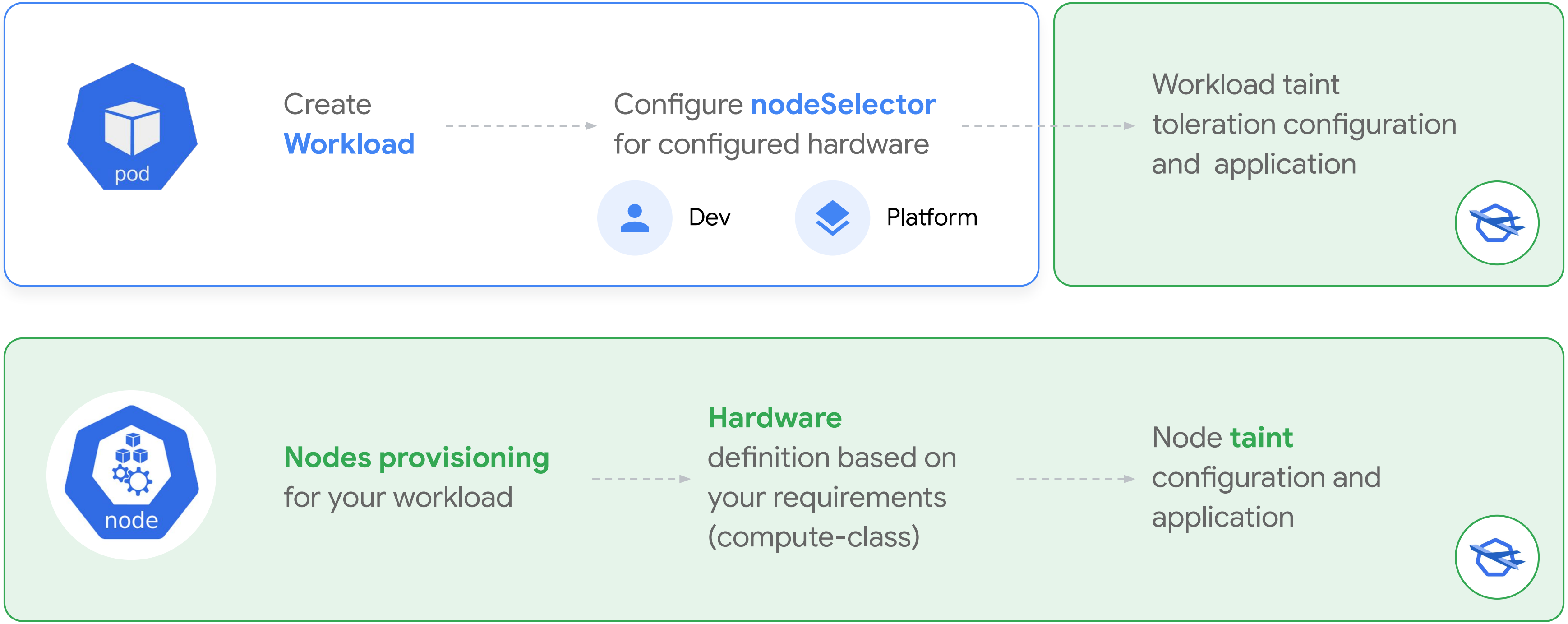
Application Team	Kubernetes (Portable)	Continue to manage workloads the way you're used to. Autopilot retains the full flexibility and power of the Kubernetes API and community.
Platform Team	Policy	Best practice configurations out-of-the box, with full freedom to extend with any policies that are important to your business.
	Provider Specific API	Autopilot provides a standardized API that makes it easy to utilize provider specific capabilities. The customization you need, less boilerplate.
	Substrate	Autopilot manages the substrate, taking full responsibility for day-2 infrastructure ops handling capacity, patching, and upgrade coordination.

Node selection with traditional managed Kubernetes



pod
Create **Workload**
Configure **nodeSelector** for configured hardware
Configure **Tolerate** taint
Dev
node
Create **Node Pool**
Define what **resources/hardware** you need
Configure **taint**
Platform

Configure node selection with GKE Autopilot



pod

Create **Workload**

Configure **nodeSelector** for configured hardware

Dev

Platform

Workload taint toleration configuration and application

node

Nodes provisioning for your workload

Hardware definition based on your requirements (compute-class)

Node **taint** configuration and application

Compute Class | Selecting specific hardware classes

Faster time to market. GKE Autopilot compute class let you set specific hardware requirements for **individual workloads.**

General-Purpose

Best price/ performance for x86

Great default choice for most compute

- Web serving / API
- Microservices
- Dev environments

Series: **E family** (Default)

Balanced

Consistent performance

Wide range of VM shapes (high Mem/ CPU)

Very flexible and stable

- Web serving / APIs
- Microservices
- Stateful Apps (DB / Cache
- Media/Streaming
- Back office Apps

Series: **N2/ N2D**

Scale-out

Best price/performance for high throughput workloads

x86 / ARM

- Scaled-out
- Web serving / API
- Microservices
- Data log processing
- Media transcoding
- Large-scale Java applications

Series: **T2/T2D**

Accelerators

Accelerators

GPU/ TPU

GPU Sharing

- AI workloads
- Inference at large scale
- Small to medium Machine Learning
- Batch

Series: **T4 / A100 / L4 / H100**

Compute Class | Requesting compute classes

```
apiVersion: v1
kind: Pod
metadata:
 name: nginx
 labels:
   pod: nginx-pod
spec:
 nodeSelector:
   cloud.google.com/compute-class: Scale-Out
 containers:
   - image: nginx
     name: nginx-container
```

Compute Class | Requesting architecture (ARM)

```
apiVersion: v1
kind: Pod
metadata:
 name: nginx
 labels:
   pod: nginx-pod
spec:
 nodeSelector:
   cloud.google.com/compute-class: Scale-Out
   kubernetes.io/arch: arm64
 containers:
   - image: nginx
     name: nginx-container
```

Compute Class | Requesting spot pods

```
apiVersion: v1
kind: Pod
metadata:
 name: nginx
 labels:
   pod: nginx-pod
spec:
 nodeSelector:
   cloud.google.com/compute-class: Scale-Out
   kubernetes.io/arch: arm64
   cloud.google.com/gke-spot: "true"
 containers:
   - image: nginx
     name: nginx-container
```

Compute Class | Requesting GPU

```
apiVersion: v1
kind: Pod
metadata:
 name: tensorflow
 labels:
   pod: tensorflow-pod
spec:
 nodeSelector:
   cloud.google.com/compute-class: "Accelerator"
   cloud.google.com/gke-accelerator: nvidia-tesla-a100
 containers:
   - image: tensorflow/tensorflow:latest-gpu-jupyter
     name: tensorflow-a100
     resources:
       requests:
         nvidia.com/gpu: "1"
```

Compute Class | Define and use your own classes

Advanced node config options, including fall-back priorities with reconciliation abstracted to a single node selector in the workload

Node selection prioritization

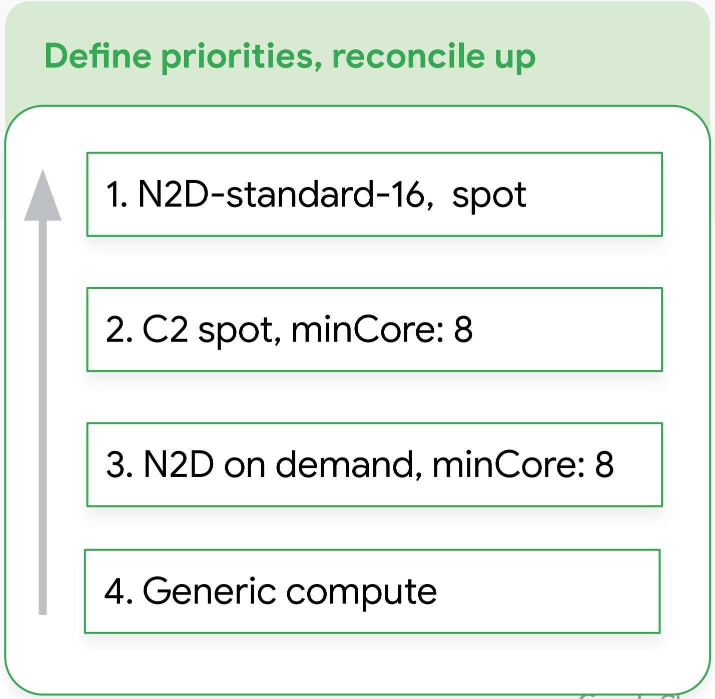
- Fall-back priorities for nodes
- **Spot** priorities with fall-backs
- Define by instance characteristics (machine type/family, size)
- Scaling profiles
- GPU/TPU support
- Named GCE **reservations**
- Node system configuration

Active reconciliation to top priorities

- Reconcile workloads to top priorities
- Subject to TTL, PDB, etc

Default classes

- Override Autopilot default class per namespace
- Even without nodeSelectors, workloads get desired node config



Define priorities, reconcile up

1. N2D-standard-16, spot

2. C2 spot, minCore: 8

3. N2D on demand, minCore: 8

4. Generic compute

Compute Class | Define and use your own classes

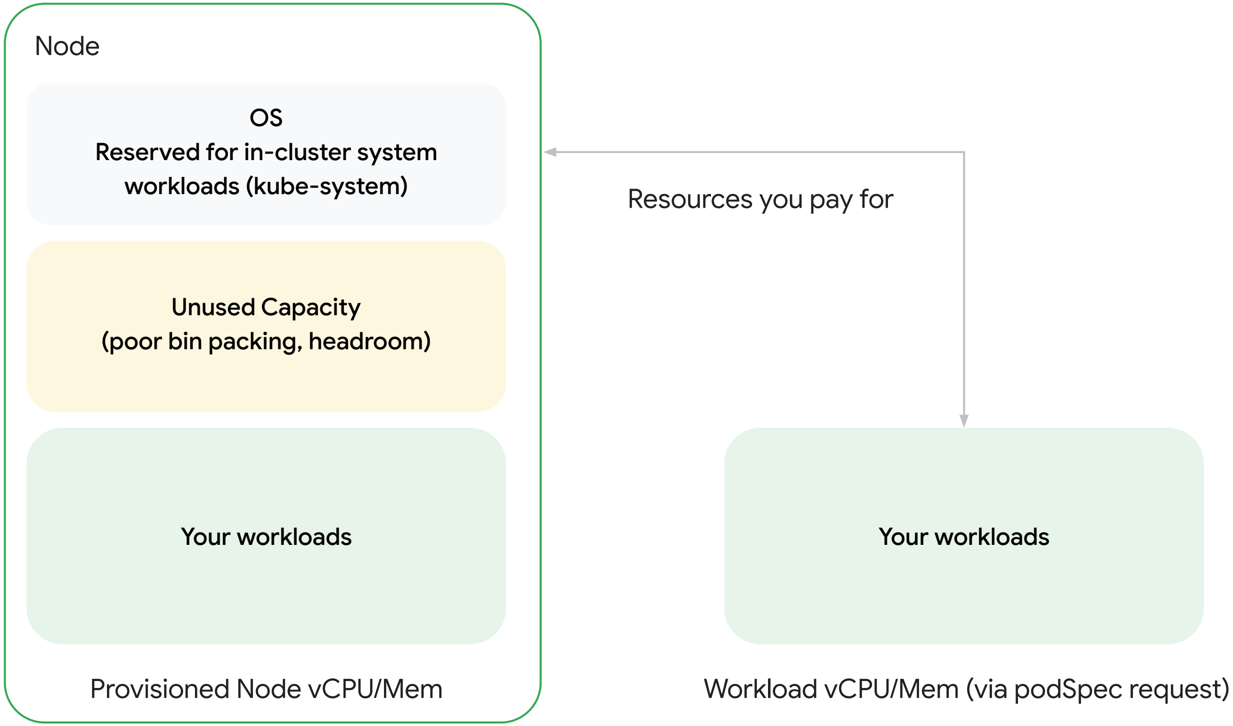
```
apiVersion: autoscaling.gke.io/v1alpha1
kind: ComputeClass
metadata:
 name: custom-config
spec:
activeMigration:
       optimizeRulePriority : true
nodePoolAutoCreation:
       enabled              : true

priorities:
-      machineType    : n2d-standard-16
       spot           : true

-      family         : c2
       spot           : true
       minCores       : 8

-      family         : n2d
       spot           : false
       minCores       : 8
```

```
apiVersion: v1
kind: Pod
metadata:
 name: nginx
 labels:
   pod: nginx-pod
spec:
 nodeSelector:
   cloud.google.com/compute-class: custom-config
 containers:
   - image: nginx
     name: nginx-container
```

Traditional Managed Kubernetes Node-based pricing

Autopilot pod-based pricing

Cost optimized



Node

OS
Reserved for in-cluster system workloads (kube-system)

Unused Capacity
(poor bin packing, headroom)

Your workloads

Provisioned Node vCPU/Mem

Resources you pay for

Your workloads

Workload vCPU/Mem (via podSpec request)