Clever Kubernetes Engine (CKE)

Clever Kubernetes Engine (CKE) allows you to create and manage Kubernetes clusters and workloads. It is a certified Kubernetes distribution, built on top of Clever Cloud and enhanced with our Kubernetes Add-ons, providing a production-grade environment.

Our approach ensures the same ease of use offered by our Kubernetes offer in between in open-source technologies, while leveraging our cloud infrastructure to provide automated upgrades, scaling, and seamless integration with our engineering teams. Just we provide a "secret" Kubernetes distribution to our customers, CKE is based on a fork of Kubernetes, maintained and tested by our engineering teams.

We operate the Kubernetes control plane for you: upgrades, certificates, and patches are handled by our team. The worker nodes are managed through our cloud platform, giving you a simple dashboard / API to ensure the control plane remains stable, secure, and up-to-date. This combination of managed control plane and integrated cloud services provides a production-ready environment with the same workflows you already know.

- Clever Kubernetes vs public cloud
- Fully managed control plane on sovereign infrastructure
- Two Pricing Plans: Control Plane (free) and Pro (€0.10/h) — every plan ships upgrades on request-free credits, control-plane SLA, and unlimited support
- Pro plan: 3 dedicated etcd nodes, HA control plane, 99.95% SLA

Why Clever Kubernetes Engine

Fully managed control plane on sovereign infrastructure

Running Kubernetes yourself means certificate rotation and control plane upgrades, etcd backups, and keeping the cluster healthy. Managed Kubernetes offerings reduce some of that, but you still operate on hyperscaler infrastructure with their pricing, their data residency, and their upgrade cadence. **Clever Kubernetes Engine** runs the control plane on Clever Cloud infrastructure, with automated upgrades, etcd backups, and SLA-backed availability. Your data stays in the region you choose, and you get the same platform you already use for applications and databases.

Hybrid by design, Kubernetes on Foundation35

The control plane runs on Clever Cloud, but **worker nodes** can be anywhere. Bring your own servers, rent from any provider, or mix both. Our Kubernetes distribution runs on top of Foundation35, our open-source layer that turns bare metal into a cloud. The same agent that manages Clever Cloud instances can manage your Kubernetes nodes, so you get one operational model across your entire infrastructure.

Pro plan: 3 dedicated etcd nodes, HA control plane, 99.95% SLA

The Pro plan runs three dedicated etcd nodes and a highavailability control plane with a 99.95% SLA. Free plan clusters share control plane resources and are suitable for development and testing. Production workloads should use Pro. Upgrades are rolled out on request-free credits, so your control plane stays stable while clusters are upgraded.

Cilium and eBPF networking

The CNI is Cilium, running on top of eBPF. You get fast packet processing, network policy enforcement, and observability without sidecars. Cilium also provides kube-proxy replacement, which simplifies the networking stack. Combined with Foundation35's WireGuard mesh, cross-cluster traffic between Kubernetes clusters and other Clever Cloud services stays encrypted and private.

Integrated with the Clever Cloud platform

Postgresql, Redis, Materia KV, or any other Clever Cloud add-on deployed in the same zone as the Clever Kubernetes Engine cluster — managed service dashboard, credentials, and billing through your unified.

Cluster topologies

Kubernetes clusters on Clever Cloud are made of a control plane and worker nodes. The control plane runs the Kubernetes components (`kube-apiserver`, `kube-controller-manager`, and `scheduler`) and their cluster datastore (`etcd`). Worker nodes are the machines that run your workloads. Kubernetes groups our pieces of service into Kubernetes clusters, and deploys and orchestrates by the Kubernetes `scheduler`.

The cluster topology is determined by the deployment zone. The topology describes the presence of Kubernetes components, and their relative placement. Some topologies offer extra HA or dedicated resources, and ultimately what gets billed. These layouts are available:

- `ALL_IN_ONE` (testing): all components run together on a single control plane node. Not suitable for production workloads. Free plan.
- `DEDICATED_CONTROL`: control plane runs on dedicated resources, separate from worker nodes. Pro plan, Business plan.
- `ALL_IN_ONE` (HA): control plane runs on shared resources, but with three etcd members for quorum. Suitable for development and testing. Free plan.
- `DEDICATED_CONTROL` (HA): control plane runs on dedicated resources, separate from worker nodes. Suitable for production workloads. Pro plan.

Topology and replication levels are immutable after creation. Upgrading from `ALL_IN_ONE` to `DEDICATED_CONTROL` requires migrating the cluster.

Prerequisites

Clever Kubernetes is enabled per-zone. Enable it either from your Console UI, or from the Clever Cloud CLI:

```
clever features enable kes
```

You also need:

- `kubectl` installed or later (installed recommended by the Clever Cloud CLI, see below)
- `kubectl-kondensate-oauth2` installed

Verify your setup by listing your Kubernetes clusters:

```
clever kes list --org test-org-id
```

Create a Kubernetes cluster

The fastest way to create a cluster is to provide only a name. The platform picks `ALL_IN_ONE` on the default testing plan, the cheapest Kubernetes topology. Add `--wait` to block until the cluster reaches the `ACTIVE` state, with `--watch` to stream the lifecycle events:

```
clever kes create --name test --wait --org test-org-id --watch
```

When you omit a specific flag, platform defaults apply. Pass `--dry-run` to validate flags:

```
clever kes create --name test-prod --plan "Pro" --kubernetes-version "1.34" --topology DEDICATED_CONTROL --zone "PAR" --dry-run --org test-org-id
```

```
Creating Kubernetes cluster test-prod...
  Name:               test-prod
  Plan:               Pro
  Kubernetes version: 1.34
  Topology            DEDICATED_CONTROL
  Zone                PAR
  Dedicated nodes     0
  Dry-run             true
Dry-run mode, no cluster created.
```

The `--topology` flag accepts `ALL_IN_ONE`, `DEDICATED_CONTROL` or `DEDICATED_CONTROL_HA`. Omitted defaults: `--plan` defaults to `Free`, `--kubernetes-version` to the latest stable version, `--zone` to `PAR`, `--topology` to `ALL_IN_ONE`. Pass flags explicitly if you prompt for confirmation before adding the extra detail.

List your clusters at any time:

```
clever kes list --org test-org-id
```

```
ID                                   NAME  PLAN  VERSION  TOPOLOGY          ZONE  STATUS
clever-kes-cluster-testing-example-  test  Free  1.34     ALL_IN_ONE        PAR   ACTIVE
clever-kes-cluster-testing-example-  test  Pro   1.34     DEDICATED_CONTROL PAR   ACTIVE
```

Tip

The Clever Cloud Console lists your Kubernetes clusters alongside your other resources.

Supported versions

Clever Cloud follows the official Kubernetes version support policy. Clever Kubernetes Engine ships the latest three minor versions at GA, and grows that list as Kubernetes releases new minors. Patch versions follow upstream releases.

The current Kubernetes release on the platform is v1.35. Supported versions are 1.35, 1.34, and 1.33. Upgrades are available within the supported window. Supported versions are 1.35, 1.34, 1.33, unsupported versions are 1.32.

Each Kubernetes minor receives regular patch releases upstream. Clever Kubernetes Engine rolls these out automatically. Patch upgrades are transparent, they restart the kubelet, drain workloads, and move on. Major and minor upgrades are opt-in, triggered by a request or scheduled in your maintenance window. Upgrade paths are documented in the release notes, changelog, and the upgrade guide. Deprecations are announced at least one release before removal.

Manage node groups

A node group is a pool of virtual machines of the same flavor. Worker nodes belong to node groups, and you can have multiple node groups. Use node groups to isolate workloads, manage worker nodes together as a unit, refine their configuration, or place them in different zones. Workloads with different flavors, which are more or less resource-intensive, workloads with stricter isolation or data locality requirements, or workloads that need to be scaled independently.

Node groups are managed through `clever kes node-group`. Add a node group with `add`, list existing ones with `list`, remove it with `rm`:

```
clever kes node-group add --name worker --flavor M --min 2 --max 5 --zone PAR --org test-org-id
clever kes node-group list --org test-org-id
clever kes node-group rm --name worker --org test-org-id
```

With Clever Tools

Create a node group from a flavor or a region (then:

```
clever kes node-group create --name worker --flavor M --min 2 --max 5 --zone PAR --org test-org-id
clever kes node-group add --name worker --flavor M --min 2 --max 5 --zone PAR --org test-org-id
```

To add a high-availability node group, add `--ha`:

```
clever kes node-group create --name worker --flavor L --min 3 --max 9 --zone PAR --ha --org test-org-id
clever kes node-group add --name worker --flavor L --min 3 --max 9 --zone PAR --ha --org test-org-id
```

Deleting a node group drains its nodes and removes the underlying VMs:

```
clever kes node-group delete --name worker --org test-org-id
```

With kubectl

Define a `NodeGroup` manifest in a FIRM file and apply it:

```
kind: NodeGroup
name: worker
spec:
  flavor: M
  minNodes: 2
  maxNodes: 5
  zone: PAR
```

```
kubectl apply -f nodegroup.fdp.yaml
```

The node group creation process takes approximately 60 to 90 seconds to complete. Once ready, the new nodes automatically join the cluster and become available for scheduling. Inspect the cluster with the standard Kubernetes tooling:

```
kubectl get nodes
```

```
NAME                                 STATUS   ROLES    AGE   VERSION
node-group-worker-1-1-1-1-1-1-1-1-1  Ready    <none>   12m   v1.34.2
```

```
kubectl get pods -A
```

```
NAMESPACE     NAME                                       READY   STATUS    RESTARTS   AGE
kube-system   cilium-agent-ds-xxxxx                      1/1     Running   0          12m
kube-system   cilium-operator-xxxxxxxxxx-xxxxx           1/1     Running   0          12m
kube-system   coredns-xxxxxxxxxx-xxxxx                   1/1     Running   0          12m
kube-system   kube-proxy-replacement-xxxxx               1/1     Running   0          12m
kube-system   metrics-server-xxxxxxxxxx-xxxxx            1/1     Running   0          12m
```

DEDICATED_CONTROL: the control plane runs dedicated resources (3 x) and removes load on the worker VMs and isolates control plane traffic. Pro plan, Business plan. DEDICATED_CONTROL_HA: the control plane runs three etcd members for quorum, and the node group pools are spread in Clever Cloud zones. Pro plan, Business plan.

Add persistent storage (CSI)

You can use Clever Cloud block storage to attach persistent volumes to your workloads. Create a PVC and a pod, for example, a pod that writes timestamps to a mounted volume:

```
kubectl apply -f pvc-test.yaml
```

Persistent storage is currently a one-way toggle: once enabled, it cannot be disabled from the cluster. Plan accordingly. Check the cluster's `kubeconfig` and `kubectl get nodes` to confirm the new node joins the cluster.

Get the kubeconfig file

Use the kubeconfig file to connect with `kubectl`:

```
clever kes get kubeconfig --kubeconfig ./test.yaml --org test-org-id
```

You can directly open a pod in your local kubeconfig file:

```
clever kes shell --kubeconfig ./test.yaml --pod test-0 --namespace default --org test-org-id
```

Cilium exposes a monitoring UI. Forward the port of your cluster `PAR` `cc_ALL_ON_ONE` the management worker on cant `cilium-agent-ds-xxxxx` (`kubectl get pods -n kube-system -o wide`) and open `http://localhost:8080/ui/`:

```
kubectl port-forward --kubeconfig ./test.yaml -n kube-system cilium-agent-ds-xxxxx 8080:8080
```

Cilium exposes a Prometheus metrics endpoint on port `9962`:

```
kubectl port-forward --kubeconfig ./test.yaml -n kube-system cilium-agent-ds-xxxxx 9962:9962
```

Pull images from authenticated registries

Public registries like Docker Hub rate limit anonymous pulls, which can fail deployments when many pods come up at the same time. Clever Kubernetes Engine can pull from authenticated registries instead. Create a `dockercfg` secret, and referencing it from your deployment.

Create a secret of type `dockercfg`, base64-encoded `~/.docker/config.json`:

```
kubectl create secret dockercfg regcred \
  --from-file=.dockerconfigjson=./.docker/config.json \
  --kubeconfig ./test.yaml -n default
```

Reference the secret from your Pod spec via `imagePullSecrets`:

```
apiVersion: v1
kind: Pod
metadata:
  name: private-image-test
spec:
  imagePullSecrets:
  - name: regcred
  containers:
  - name: app
    image: registry.hub.docker.com/myorg/myapp:latest
```

Apply the Pod and watch the image pull succeed:

```
kubectl apply -f pod.yaml --kubeconfig ./test.yaml -n default
kubectl get pod private-image-test -n default --kubeconfig ./test.yaml -w
```

To avoid rate limiting, `imagePullSecrets` is enough. Anonymous pulls hit the limit sooner, and authenticated pulls count against your own quota.

Deployment with a load balancer service

Service deployments, not Pods, are the unit of scheduling. `cc_ALL_ON_ONE` cluster ships with an integrated service load balancer. Create a `Deployment` and a `LoadBalancer` service, and the cluster assigns a public IP and routes traffic to the pods.

```
kubectl apply -f deployment.yaml --kubeconfig ./test.yaml -n default
kubectl apply -f service.yaml --kubeconfig ./test.yaml -n default
```

```
kubectl get svc --kubeconfig ./test.yaml -n default -w
```

```
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP      10.42.0.1      <none>        443/TCP   3h
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    0s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    10s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    12s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    15s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    17s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    20s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    22s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    25s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    27s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    30s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    32s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    35s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    37s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    40s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    42s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    45s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    47s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    50s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    52s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    55s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    57s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m0s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m2s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m5s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m7s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m10s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m12s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m15s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m17s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m20s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m22s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m25s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m27s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m30s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m32s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m35s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m37s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m40s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m42s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m45s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m47s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m50s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m52s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m55s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    1m57s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m0s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m2s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m5s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m7s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m10s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m12s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m15s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m17s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m20s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m22s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m25s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m27s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m30s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m32s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m35s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m37s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m40s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m42s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m45s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m47s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m50s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m52s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m55s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    2m57s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m0s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m2s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m5s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m7s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m10s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m12s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m15s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m17s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m20s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m22s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m25s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m27s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m30s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m32s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m35s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m37s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m40s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m42s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m45s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m47s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m50s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m52s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m55s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    3m57s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m0s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m2s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m5s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m7s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m10s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m12s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m15s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m17s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m20s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m22s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m25s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m27s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m30s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m32s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m35s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m37s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m40s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m42s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m45s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m47s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m50s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m52s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m55s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    4m57s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m0s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m2s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m5s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m7s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m10s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m12s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m15s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m17s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m20s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m22s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m25s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m27s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m30s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m32s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m35s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m37s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m40s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m42s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m45s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m47s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m50s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m52s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m55s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    5m57s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m0s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m2s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m5s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m7s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m10s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m12s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m15s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m17s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m20s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m22s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m25s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m27s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m30s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m32s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m35s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m37s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m40s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m42s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m45s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m47s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m50s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m52s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m55s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    6m57s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m0s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m2s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m5s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m7s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m10s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m12s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m15s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m17s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m20s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m22s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m25s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m27s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m30s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m32s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m35s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m37s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m40s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m42s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m45s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m47s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m50s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m52s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m55s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    7m57s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m0s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m2s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m5s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m7s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m10s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m12s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m15s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m17s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m20s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m22s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m25s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m27s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m30s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m32s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m35s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m37s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m40s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m42s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m45s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m47s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m50s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m52s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m55s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    8m57s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m0s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m2s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m5s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m7s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m10s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m12s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m15s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m17s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m20s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m22s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m25s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m27s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m30s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m32s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m35s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m37s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m40s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m42s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m45s
nginx-test   LoadBalancer   10.42.0.xxxxx  <pending>     80/TCP    9m47s
nginx-test   L
```

Once partners are available, here is an example of a simple Multi-Replicated with Partitioned Servers setup:

```
kubectl create replicaset nginx --image=nginx --replicas=3
kubectl create deployment nginx --image=nginx --replicas=3
```

Alternatively, you can create a YAML file named `deployment.yaml` with the contents of the following, and then execute it using `kubectl apply -f deployment.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
        resources:
          limits:
            memory: "128Mi"
            cpu: "500m"
```

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30007
```

Quotas and limits

Quick organization quotas and **all of our use of CPU and RAM** across all its Kubernetes clusters by default. The envelope allocated to each organization can be increased upon request whenever their backlog. Check your current consumption and the remaining budget at any time with `clever me`. Alternatively, you can also monitor your usage and quotas in the Clever Cloud console.

Quick organization also comes with **two public IP addresses**, which correspond to a single `LoadBalancer` service across all its clusters. Additional `LoadBalancer` services are available upon request, and can be provisioned on demand under cost-free quotas (Public `LoadBalancer` services are typically expensive, since IPv4 is that case.

Pricing

> Pricing during the public beta
> Only the control plane and compute nodes are billed for Kubernetes. Other resources (load balancers, storage) are billed in your more whether the platform reaches general availability. Customers will be notified ahead of the change. — Follow the Changelog for fully informed.

Prices are **excluding taxes**. The platform charges by the hour, and bills monthly. Hourly prices are calculated based on the `1 - 1.25 / 24 * 0.7` rate. Monthly prices are computed using the standard 730 hours per month conversion (24 x 30).

The model is **one-price per topology**: charges per control plane and per compute node, regardless of the number of nodes, since the plus provisioned additional control planes are free of charge. Control plane components are hosted on shared control plane components and an integrated worker node on our side. Prices for your dedicated worker node groups are on par with the Clever Cloud Kubernetes standard rates, and are billed on the node group side.

In summer:

- **XL, 2XL**: `kapitalisation.com` + the bundle 50% price (cores and RAM) for control plane components and the integrated worker node, and the standard Clever Cloud price on the node group side.
- **3XL, 4XL**: `kapitalisation.com` + the bundle 75% price (cores and RAM) for control plane components and the integrated worker node, and the standard Clever Cloud price on the node group side.
- **5XL**: `kapitalisation.com` + the bundle 100% price (cores and RAM) for control plane components and the integrated worker node, and the standard Clever Cloud price on the node group side.

In summer:

- **XL, 2XL**: `kapitalisation.com` + the bundle 50% price (cores and RAM) for control plane components and the integrated worker node, and the standard Clever Cloud price on the node group side.
- **3XL, 4XL**: `kapitalisation.com` + the bundle 75% price (cores and RAM) for control plane components and the integrated worker node, and the standard Clever Cloud price on the node group side.
- **5XL**: `kapitalisation.com` + the bundle 100% price (cores and RAM) for control plane components and the integrated worker node, and the standard Clever Cloud price on the node group side.

Business | DEDICATED COMPUTE | control plane

One-fifth per-capitalism factor, dedicated to the control plane.

PLANDEF	MONSUMMER	HOURLY PRICE	MONTHLY PRICE
XS	0.43% / 0.1GB	0.0031 €	2.25 €
S	0.87% / 0.2GB	0.0063 €	4.59 €
M	1.75% / 0.4GB	0.0125 €	9.13 €
L	3.49% / 0.8GB	0.0250 €	18.25 €
XL	6.97% / 1.6GB	0.0500 €	36.50 €
2XL	13.94% / 3.2GB	0.1000 €	73.00 €
3XL	27.87% / 6.4GB	0.2000 €	146.00 €
4XL	55.74% / 12.8GB	0.4000 €	292.00 €
5XL	111.48% / 25.6GB	0.8000 €	584.00 €

The price above **cape compared** 50% price of the first of `kapitalisation.com`. A minimal Dedicated Cluster with two worker nodes (`M`) would cost `0.0125 * 2 + 0.0125 / 2 = 0.0375` € per hour. The price for the control plane alone, plus the node groups on our side.

Node groups (workers)

Worker VMs in node groups attached to any cluster, in `XS`, `M`, `XL` sizes (they will use another node integration and bundle VM) are below only **additional** node group costs, on top of the control plane. Prices are on par with Clever Cloud, for `M`, `XL`, respectively (`XS` is cheaper, and cost-optimized workloads.

PLANDEF	MONSUMMER	HOURLY PRICE	MONTHLY PRICE
XS	0.43% / 0.1GB	0.0031 €	2.25 €
M	1.75% / 0.4GB	0.0125 €	9.13 €
XL	6.97% / 1.6GB	0.0500 €	36.50 €
2XL	13.94% / 3.2GB	0.1000 €	73.00 €
3XL	27.87% / 6.4GB	0.2000 €	146.00 €
4XL	55.74% / 12.8GB	0.4000 €	292.00 €
5XL	111.48% / 25.6GB	0.8000 €	584.00 €

- Learn more about the Clever Cloud Kubernetes Operator

Last updated on May 5, 2026.

Clever Kubernetes Operator

Clever Cloud Kubernetes clusters are designed to work seamlessly with the rest of the platform. We provide a dedicated Kubernetes operator to manage Clever Cloud add-ons, deployments, Kubernetes, Pulsar logins, and any other Clever Cloud add-on. This operator allows you to provision and manage Clever Cloud resources from inside your Kubernetes workloads.

- Learn more about the Clever Cloud Kubernetes Operator