Scaleway

01 - Foundations

?? 1. Les grandes familles

Besoin	Produit à retenir
VM classique	**Instances**
Serveur physique dédié	**Elastic Metal**
GPU / IA / rendu / encodage	**GPU Instances**
ARM / Ampere	**CAX**
Mac Apple Silicon	**Apple Silicon / Mac mini**
Kubernetes managé	**Kapsule**
Conteneur serverless	**Serverless Containers**
Fonction serverless	**Serverless Functions**
Fichiers / objets	**Object Storage**
Disque attaché à une VM	**Block Storage**
Archivage froid	**Glacier**
Base SQL	**Managed Databases**
Cache / sessions	**Redis**
Analytics / gros volumes	**ClickHouse**
Répartition du trafic	**Load Balancer**
IP publique portable	**Flexible IP**
Monitoring / observabilité	**Cockpit**
Identités / permissions	**IAM**
Messages pub/sub	**Topics & Events**
Messages point-to-point	**Queues**
Emails transactionnels	**Transactional Email**
IA générative par API	**Generative APIs**
Infrastructure as Code	**Terraform**

?? 2. Instances

Development Instances

➡ Développement / tests
➡ **vCPU partagés**

General Purpose

➡ Usage général
➡ C'est notamment là qu'on trouve **CAX**, avec processeurs **Ampere ARM**.

GPU Instances

➡ **IA**
➡ Data processing
➡ Rendering
➡ Video encoding

Elastic Metal

➡ **Bare Metal**
➡ Serveur physique, pas une VM.

Apple Silicon

➡ Mac mini Apple Silicon
➡ Connexion notamment **SSH + VNC**
➡ Attention au **minimum de facturation de 24 h**.

? 3. Storage

Object Storage

Imagine :

Bucket → Objects

Un bucket peut contenir énormément d'objets, pas seulement 100.

Les objets sont **privés par défaut**.

Versioning

Si tu modifies un objet avec le versioning activé :

```
photo.jpg
   ↓ modification
photo.jpg version 1
photo.jpg version 2
photo.jpg version 3
```

➡ Les anciennes versions sont conservées.
➡ Elles consomment donc du stockage → **coût supplémentaire**.

Site web dans Object Storage

Pour personnaliser le domaine :

CNAME

Block Storage

Pense :

> **Block Storage = disque**

Tu peux :

attach → attacher le volume à une Instance
detach → le détacher

Très différent d'Object Storage.

Glacier

Pense :

> **Glacier = archivage / stockage froid**

Et surtout :

 Pas de Multi-AZ redundancy comme le stockage Standard Multi-AZ.

?? 4. Databases

PostgreSQL / MySQL

➡ données structurées
➡ applications classiques

Redis

➡ **mémoire / vitesse**
➡ cache
➡ sessions
➡ données temporaires
➡ API

Question typique :

> "Best database for storing sessions?"

 Redis

ClickHouse

➡ Analytics
➡ Très gros volumes de données
➡ Data science / Business Intelligence

Et :

> **Data Warehouse = centralized repository for large sets of structured data**

?? 5. Kubernetes

Pour l'instant, retiens simplement :

Kapsule = Kubernetes managé par Scaleway

Le cluster est intégré à l'écosystème Scaleway.

Et surtout :

> **Terraform peut gérer un cluster Kapsule.**

On va approfondir ça juste après avec la certification **Scaleway Associate Kubernetes**. □□

? 6. Serverless

Serverless Functions

➡ Tu déploies une fonction
➡ Scaleway gère l'infrastructure
➡ Ça scale automatiquement.

Quand il n'y a plus d'activité :

```
requêtes
  ↓
instances
  ↓
0
```

Puis une nouvelle requête arrive.

Le temps nécessaire pour redémarrer :

Cold start

Point important

Une instance de Serverless Function ne traite pas plusieurs requêtes simultanément.

➡ Scaleway peut créer plusieurs instances pour gérer la charge.

Serverless Containers

➡ Tu fournis un **container**
➡ Scaleway gère l'infrastructure
➡ Possibilité d'utiliser des **encrypted environment variables** pour les informations sensibles.

? 7. Network

Load Balancer

➡ Distribue les requêtes entre plusieurs ressources.

Il peut être :

Public
Private

Flexible IP

C'est une IP :

public + portable + persistante

Tu peux :

```
Flexible IP
     ↓
Instance A
     ↓
detach
     ↓
Instance B
```

Elle est indépendante de la ressource.

⚠ Elle est liée à une **Availability Zone**.

? 8. IAM

IAM =

> **Identity and Access Management**

Il sert à gérer :

qui → peut faire quoi → sur quelles ressources

Organization

L'Organization regroupe les utilisateurs et ressources.

Owner

➡ propriétaire de l'Organization.

Members / Users

➡ utilisateurs auxquels on attribue des permissions.

Policies

➡ définissent les permissions.

API Keys

Point à retenir :

> **API keys inherit the permissions of their bearer.**

Donc une API key n'est pas magiquement plus puissante que l'utilisateur auquel elle appartient.

MFA

➡ authentification multi-facteurs.

? 9. Cockpit

Cockpit =

> **Observability**

Il permet de centraliser :

- métriques
- logs
- dashboards
- alertes

Et surtout :

 pré-built dashboards pour les produits compatibles.

Cockpit n'est pas limité aux produits Scaleway : il peut aussi recevoir des données externes.

? Facturation Cockpit

À retenir pour le QCM :

 consultation d'un dashboard → pas facturée
 création d'une alerte → pas facturée
⚠ rétention au-delà de la limite → facturée

? 10. Messaging

Deux concepts à ne surtout pas mélanger :

Topics & Events

◫ Publish / Subscribe

```
Publisher
    ↓
 Topic
 ↙   ↓   ↘
A    B    C
```

Plusieurs consommateurs peuvent recevoir le message.

Queues

◫ Point-to-point

```
Producer
   ↓
 Queue
   ↓
Consumer
```

➡ Un message est destiné à être traité par un consommateur.

◫ Mémo :

Topics = broadcast
Queues = file d'attente

? 11. Transactional Email

Workflow :

```
1. Create account
       ↓
2. SPF / DKIM / DMARC
       ↓
3. Integrate SMTP / API
       ↓
4. Send
       ↓
5. Monitor
```

Donc :

Create → Authenticate → Integrate → Send → Monitor

? 12. AI

GPU Instances

➡ GPU à la demande.

AI Supercomputers

➡ infrastructures GPU massives pour l'IA.

Generative APIs

➡ accès aux modèles génératifs via API.

Facturation :

> **Pay-as-you-go / tokens**

Donc :

plus de tokens → plus de consommation

?? 13. Terraform

Terraform = **Infrastructure as Code**

Au lieu de :

```
Console
→ clic
→ clic
→ clic
```

tu décris :

```
Terraform
→ configuration
→ plan
→ apply
```

Bonne pratique importante

Évite de mélanger :

création manuelle dans Console + gestion Terraform

Si une ressource existe déjà :

➡ **import dans Terraform state**
➡ puis gestion par Terraform.

? 14. Region vs Availability Zone

Très important pour les QCM :

Region

➡ zone géographique.

Exemple :

Paris

Availability Zone

➡ zone d'infrastructure isolée au sein d'une région.

Et attention à la phrase :

> "Each Region is composed of two Availability Zones."

 False

Il ne faut surtout pas mémoriser "1 région = 2 AZ".

? 15. Billing

À retenir :

Facturation mensuelle

La consommation des services est généralement regroupée dans une **facture mensuelle**.

Mais certains éléments peuvent avoir des modalités particulières.

Exemple :

Domain names et certains **Elastic Metal setup fees** peuvent être facturés lors de la commande.

? 16. Les associations à connaître PAR CŒUR

Si tu ne devais retenir que ça :

```
Kubernetes       → Kapsule
Sessions         → Redis
Analytics        → ClickHouse
Structured data  → Managed Database
Objects          → Bucket
Disk             → Block Storage
Archive          → Glacier
GPU / AI         → GPU Instances
ARM / Ampere     → CAX
Bare Metal       → Elastic Metal
Monitoring       → Cockpit
Permissions      → IAM
Public IP        → Flexible IP
Traffic          → Load Balancer
Pub/Sub          → Topics & Events
Point-to-point   → Queues
IaC              → Terraform
Email            → Transactional Email
LLM API          → Generative APIs
```

Et surtout :

> **Bucket = Object Storage**
> **Volume = Block Storage**
> **Database = données structurées**
> **Redis = vitesse / sessions**
> **Kapsule = Kubernetes**

02 - Kubernetes

?? 1. Les fondamentaux Kubernetes

Pod

- **Plus petite unité déployable de Kubernetes**
- Un Pod peut contenir **un ou plusieurs conteneurs**
- Les conteneurs d'un même Pod partagent notamment le réseau et peuvent partager des volumes.

Pod → conteneur(s) → Node

Node

- Machine sur laquelle les Pods s'exécutent.
- Un Node peut être une **VM ou une machine physique**.

« Nodes are only virtual machines » → **False**

Namespace

Permet de **séparer/organiser les ressources à l'intérieur d'un cluster**.

`Namespace → séparation logique des ressources`

Labels

Permettent notamment de **sélectionner des objets Kubernetes**.

```
kubectl get pods -l app=odoo
```

True

? 2. Les principaux objets Kubernetes

Objet	À retenir
Pod	Plus petite unité déployable
Deployment	Applications généralement stateless
ReplicaSet	Maintient un nombre donné de Pods
StatefulSet	Applications stateful
DaemonSet	Pods sur chaque Node sélectionné
Job	Tâche qui se termine
CronJob	Job planifié

ReplicaSet

Son rôle est de garantir qu'un nombre défini de réplicas de Pods fonctionne.

□□ `replicas: 3` → Kubernetes essaie de maintenir **3 Pods**.

StatefulSet

□ **StatefulSet = stateless apps** → False

□□ **StatefulSet → stateful**
□□ **Deployment → stateless**

?? 3. Kapsule

Kapsule = ?

□ **Managed Kubernetes**

Scaleway gère notamment le **Control Plane**.

 Kapsule = unmanaged Kubernetes → **False**

Qui fait quoi ?

Scaleway :

- Control Plane
- composants du Control Plane
- etcd / infrastructure du Control Plane

Utilisateur :

- Worker Nodes
- Application Pods
- applications déployées

Kapsule = Control Plane managé, workload utilisateur.

?? 4. Control Plane

Le Control Plane :

- gère/orchestre le cluster
- expose l'API Kubernetes
- planifie les Pods
- maintient l'état du cluster

Stockage de l'état du cluster

etcd

etcd = base de données clé-valeur qui conserve l'état/configuration du cluster.

Application Pods ?

 Le Control Plane est responsable de faire tourner les Application Pods → **False**

Les Pods tournent sur les **Worker Nodes**.

? 5. Worker Node

Les 3 composants demandés dans ton quiz :

1. **kubelet**
2. **CNI**
3. **Pods**

À retenir :

> **kubelet → gère les conteneurs**
> **CNI → réseau**
> **Pods → unités qui encapsulent les conteneurs**

?? 6. kubectl vs Scaleway CLI

`kubectl`

Outil Kubernetes permettant d'interagir avec le cluster :

```
kubectl get pods
kubectl get deployments
kubectl apply -f app.yaml
```

◫ Kubernetes cluster → kubectl

Scaleway CLI (`scw`)

Outil permettant de gérer les **ressources Scaleway depuis le terminal**.

◫ Scaleway infrastructure → `scw`

Le CLI peut notamment servir à créer :

- un cluster Kapsule

- un Private Network

? 7. Authentification Kapsule

Pour se connecter au cluster :

 kubeconfig

Puis :

```
kubectl ...
```

 Kapsule → kubeconfig → kubectl

? 8. Helm

Helm est le **package manager de Kubernetes**.

Kapsule supporte :

 Helm charts
 manifests YAML personnalisés

 « Helm is not supported » → **False**

? 9. Réseau Kapsule

Private Networking

Kapsule supporte les Private Networks.

 Yes, via VPCs

Configuration réseau par défaut

Pour ton quiz :

 Private Network with a /22 IP subnet

Et la question suivante confirmait :

 /22

 Kapsule → Private Network → /22

? 10. Autohealing

⚠ **Très important : le quiz t'a corrigé sur cette question.**

Quand Autohealing est activé :

 Unresponsive nodes are restarted or replaced after 15 minutes.

 Failed pods are automatically rescheduled on healthy nodes.

 Upgrade automatique de Kubernetes
 Reboot quotidien du Control Plane

À retenir

> **Node malade → Autohealing → 15 min → restart / replacement**

Et :

> **Pod défaillant → rescheduling sur un Node sain**

Attention

Cette affirmation est :

> "Autohealing can detect and replace nodes with excessive resource usage."

 False

Charge CPU/RAM élevée ≠ Node défaillant.

? 11. Autoscaling

Le **Cluster Autoscaling** ajuste le nombre de Nodes selon les besoins.

Il peut :

 augmenter le nombre de Nodes lorsque la charge augmente
 réduire le nombre de Nodes lorsque la demande diminue
 contribuer à réduire les coûts

⚠ Le quiz considère également comme bénéfice :

 Ensures high availability by distributing pods across nodes

Mais :

 Autoscaling ≠ upgrade Kubernetes

Activation

 Via the Console or API

?? 12. Kubernetes upgrades

Pour Kapsule :

 Upgrade via **Console / API / CLI**
 Control Plane upgraded before Worker Nodes
 Les upgrades *in-place* minimisent les interruptions

 Downgrade vers une ancienne version

Politique de versions

Le quiz attend :

Supports at least the latest patch versions of the last three minor releases

3 dernières versions mineures → dernières versions patch

? 13. Facturation Kapsule

Réponse du quiz :

Based on the resources allocated to nodes, with no cost for the mutualized control plane

Donc :

Control Plane mutualisé → gratuit

Tu paies notamment les ressources utilisées :

- Worker Nodes
- Block Storage
- Load Balancers
- etc.

? 14. Block Storage

Block Storage sert notamment à fournir du **stockage persistant** aux applications.

Très utile pour Kubernetes lorsqu'un Pod doit conserver ses données.

Snapshot

Un **snapshot** est une **copie à un instant donné d'un volume Block Storage**.

Il peut servir notamment à :

- sauvegarder l'état d'un volume
- restaurer/créer un volume à partir de cette copie

?? 15. Odoo

Pour le TP Odoo de ta formation :

Vérifier le Deployment

```
kubectl get deployments odoo
```

Credentials PostgreSQL

Le quiz indique :

 Odoo récupère les credentials PostgreSQL via des **environment variables provenant d'un Kubernetes Secret**.

Et pour le backend Node.js :

 Même principe : credentials DB via **Kubernetes Secret → environment variables**.

PVC Odoo

La StorageClass attendue dans le cours :

 `scw-bssd`

⚠ Il existe des évolutions dans les StorageClasses Scaleway, mais pour **ce TP/quiz**, retiens `scw-bssd`.

? 16. Kapsule + services Scaleway

Kapsule est intégré à l'écosystème Scaleway.

Compatible notamment avec :

 Load Balancers
 Block Storage
 Private Networks / VPC
 autres services Scaleway

 Kapsule n'est pas isolé du reste de l'écosystème.

? 17. Kapsule / Kosmos

Kosmos

Kosmos permet notamment d'intégrer des Nodes provenant de :

 AWS
 GCP

 Kosmos = Kubernetes multi-cloud

Kapsule ? Kosmos

 On ne peut pas simplement transformer un cluster Kapsule existant en Kosmos avec `SetClusterType`.

 Kapsule et Kosmos = types de clusters distincts.

?? 18. NAT

Le **Dynamic NAT** peut masquer les adresses IP privées internes lorsqu'elles communiquent avec Internet.

 Peut contribuer à réduire la surface d'attaque.

 Dynamic NAT → private IP → public IP

? 19. VPC

Création/configuration VPC — ordre exact du quiz

⚠ Celui-ci nous a piégés deux fois □□

L'ordre **validé par le quiz** est :

1. **Choose the region**
2. **Enter a name and optional tags**
3. **Create Private Networks inside the VPC**
4. **Attach the previously created Resources – Kubernetes clusters and other resources**
5. **Select the Routing type within your VPC**

□□ **Region → Name → Private Networks → Resources → Routing**

?? 20. Création d'un cluster Kapsule standard

Autre ordre à connaître par cœur :

1. **Choose the cluster type**
2. **Choose the region**
3. **Choose a control plane offer**
4. **Configure a Private Network**
5. **Review configuration and click "Create cluster"**

□□ **Cluster type → Region → Control Plane → Private Network → Create**

? 21. Suppression d'un cluster

Avec :

```
with-additional-resources=true
```

La suppression du cluster peut également supprimer les **ressources associées**, notamment les Load Balancers concernés.

 True

? 22. Monitoring Kapsule

Les réponses correctes du quiz :

 Kubernetes Dashboard
 Prometheus and Grafana (deployed by the user)
 Scaleway Cockpit

 Scaleway Load Balancer metrics comme outil de monitoring du cluster

Dashboard + Prometheus/Grafana + Cockpit

? 23. Dedicated Control Plane

Le Dedicated Control Plane est destiné notamment aux :

 heavy workloads and business-critical applications

Dedicated Control Plane = isolation / exigences élevées / workloads critiques

---

?? 24. La question piège "ContainerSet"

Tu as eu cette question :

> “ Which of the following is a valid Kubernetes object?

Le quiz a considéré :

 ContainerSet

et a marqué :

 DaemonSet

⚠ **Attention : c'est très probablement spécifique au contenu/quiz de cette formation ou une anomalie**, car **DaemonSet est bien un objet Kubernetes natif**.

Donc pour **ce questionnaire précis**, si la même question revient :

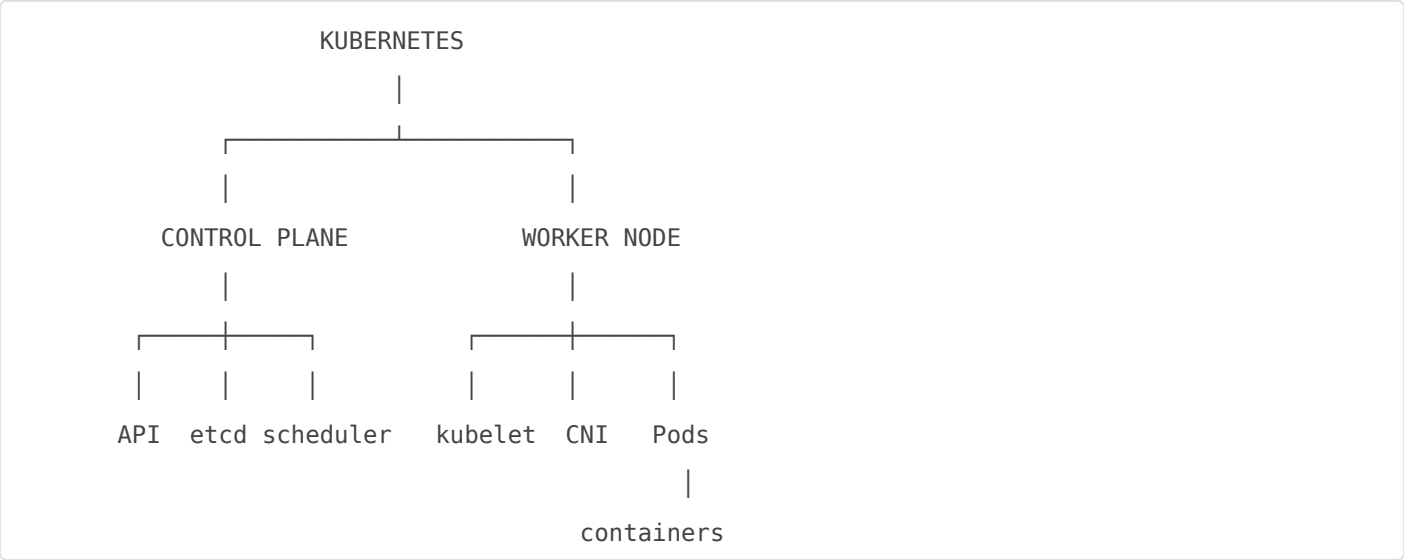
 réponds comme le quiz l'attend : ContainerSet.

Mais pour ta connaissance Kubernetes réelle :

> **DaemonSet est bel et bien un objet Kubernetes valide.**

? LA CARTE MENTALE À APPRENDRE

Si tu ne dois retenir qu'une seule page :



```
Kapsule
  ↓
Managed Kubernetes
  ↓
Scaleway manages Control Plane
  ↓
User manages workloads / worker nodes
```

```
Pod       → smallest deployable unit
Deployment → stateless
StatefulSet → stateful
ReplicaSet → maintains Pod replicas
DaemonSet → Pod on selected nodes
Namespace → separates resources
Labels → select objects
Helm → package manager
kubectl → Kubernetes CLI
scw → Scaleway CLI
kubeconfig → cluster authentication
```

```
Kapsule networking
        ↓
Private Network
        ↓
/22
```

```
Autohealing
   ↓
unresponsive Node
   ↓
15 minutes
   ↓
restart / replace
```

```
failed Pod
   ↓
rescheduled on healthy Node
```

```
Autoscaling
   ↓
increase Nodes when needed
decrease Nodes when demand drops
   ↓
cost optimization
```

```
Kapsule upgrades
   ↓
Console / API / CLI
   ↓
Control Plane first
   ↓
Workers
   ↓
NO downgrade
   ↓
3 latest minor releases
```

? Les 15 choses que je mémoriserais en priorité

1. **Kapsule = Managed Kubernetes**
2. **Control Plane = Scaleway**
3. **Pods = Worker Nodes**
4. **Pod = smallest deployable unit**
5. **Pod peut contenir plusieurs containers**
6. **Deployment = stateless**
7. **StatefulSet = stateful**
8. **ReplicaSet = nombre de replicas**
9. **kubectl = Kubernetes**
10. **scw = Scaleway**
11. **kubeconfig = authentification cluster**

12. **Kapsule Private Network = /22**
13. **Autohealing = 15 min + failed Pods rescheduled**
14. **Autoscaling = ajuste les Nodes**
15. **Upgrade = Control Plane → Workers, jamais downgrade**