Cours de virtualisation avancée: *Kubernetes*, suite

1 *Rolling Updates*

Permet des mises à jour d'images sans temps de coupure

```
---
...
spec:
  ...
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 0
  ...
```

- `maxUnavailable`: défini le nombre maximal de *replicas* indisponibles
- `maxSurge`: défini le nombre maximal de *replicas* à mettre à jour en même temps

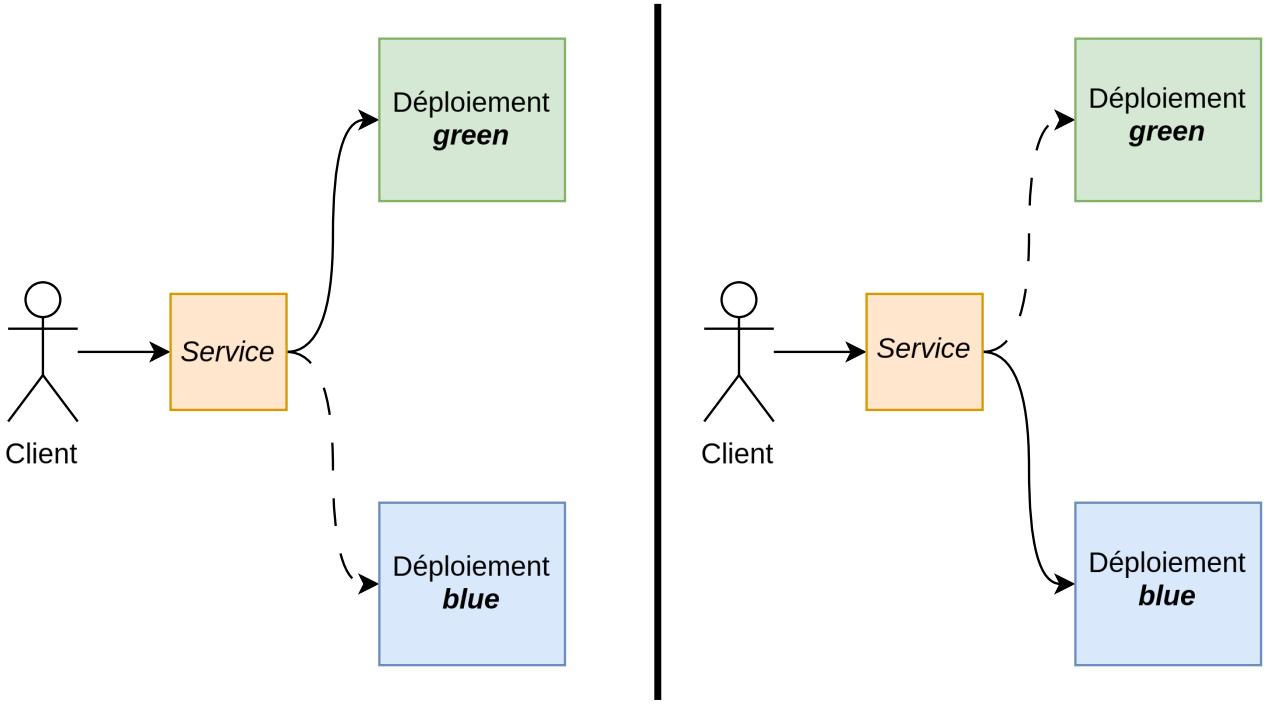
Ensuite, mettre à jour l'image avec:

```
kubectl set image deployment/<nom déploiement> <nom conteneur>=<nouvelle image:tag>
```

But:

- Mettre à jour l'application sans *downtime*

2 Déploiement *blue / green*

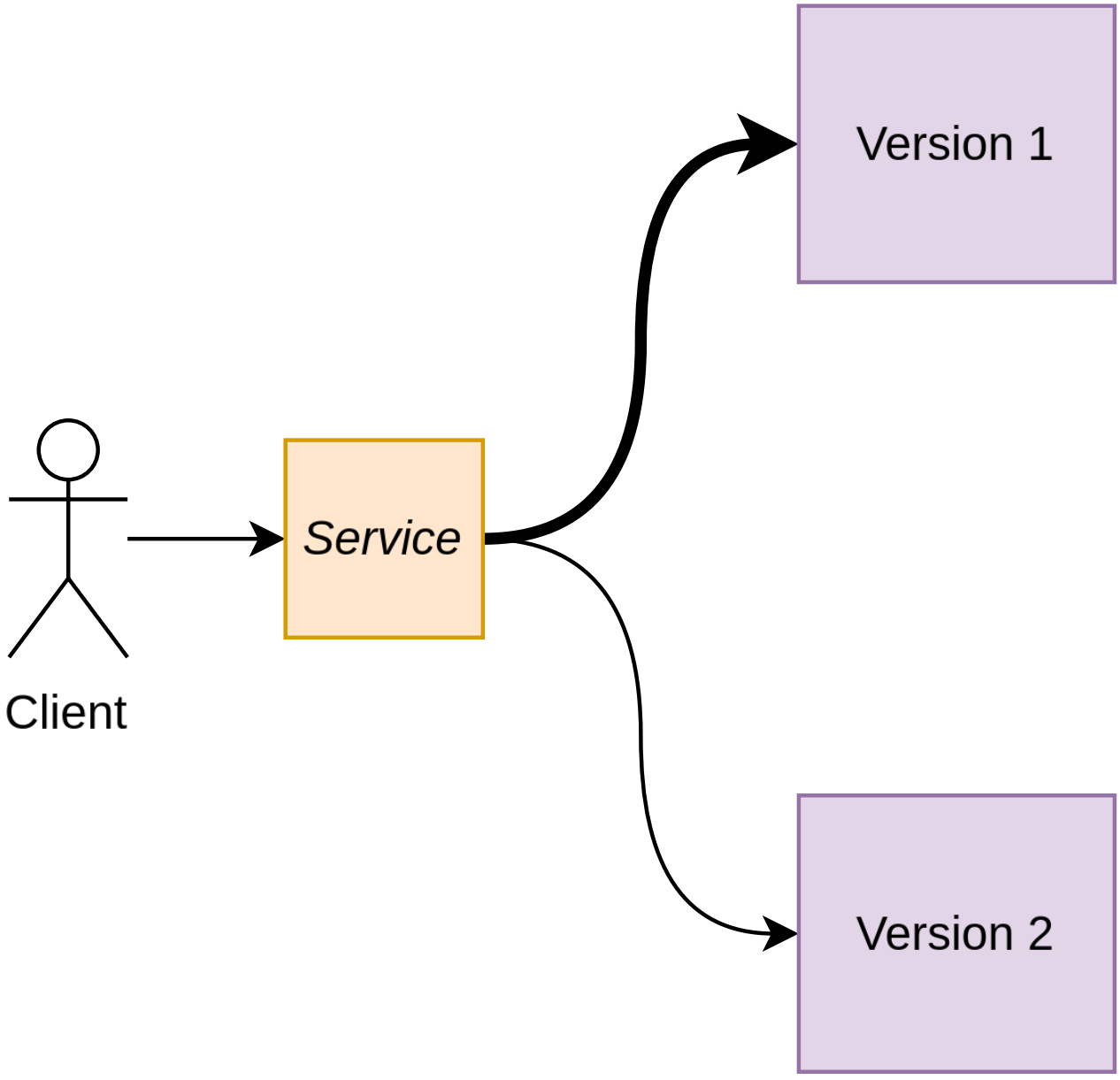


- On déploie la nouvelle version en parallèle de l'ancienne
- Une fois que tout est déployé on redirige le trafic vers le nouveau déploiement

But:

- Tester le déploiement de la nouvelle version avant de la basculer en prod

3 Déploiement *canary*



- On déploie la nouvelle version en parallèle de l'ancienne **avec moins de *replicas***
- Automatiquement une petite partie des clients tomberont sur la nouvelle version

But:

- Faire tester à quelques utilisateurs aléatoires la nouvelle version

4 Mise à l'échelle (*scaling*)

- Nécessaire quand la charge augmente
- Par exemple pour suivre l'augmentation du trafic de notre application

4.1 Verticale (*vertical scaling*)

4.1.1 Fonctionnement

- On augmente la puissance des nœuds
- Augmente les capacité de l'application existante
- L'application n'a pas besoin d'être capable de gérer la réplication

4.1.2 Inconvénients

- Coûts pas toujours proportionnels à la puissance des nœuds
- Si nœud déjà au max: comment faire ?

5 Aller plus loin