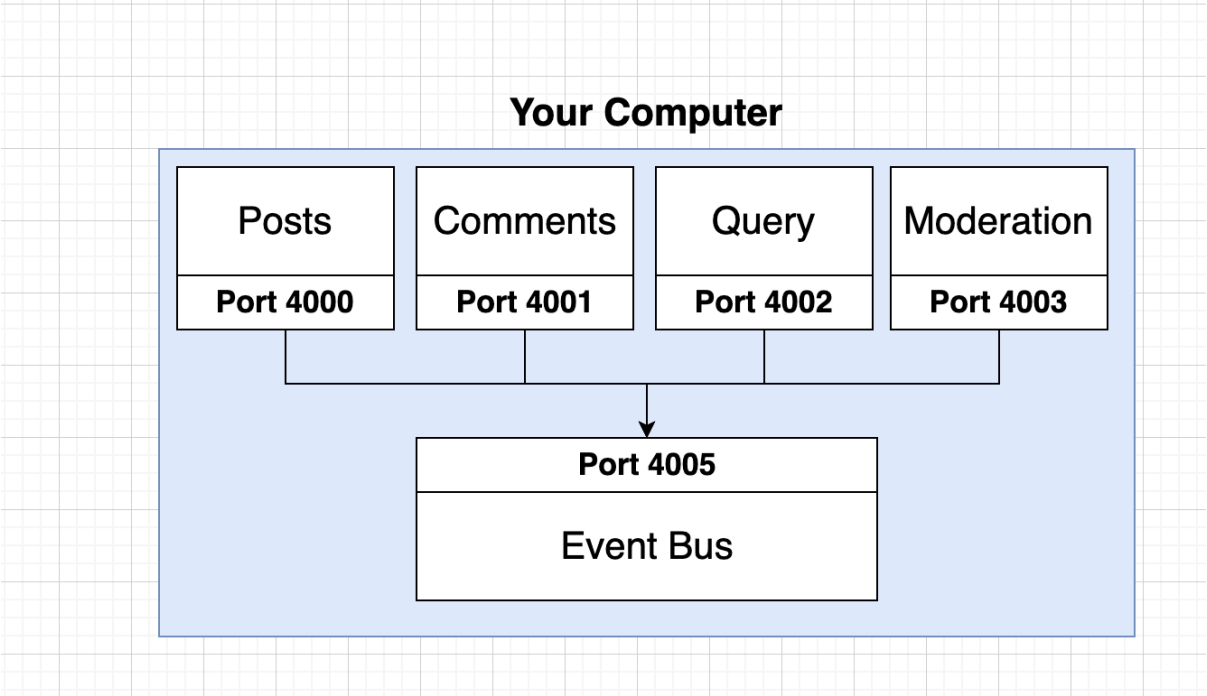
INFO4057: Architecture Logiciel
2023-2024

TP: Kubernetes et Orchestration des conteneurs

Scénario



Après avoir utilisé Docker-compose pour orchestrer l'exécution de vos conteneurs. Vous vous rendez compte que docker-compose est limité pour faire évoluer l'infrastructure de votre application microservice.

- Vous aimeriez mettre à jour votre application sans arrêt.
- Automatisez les processus de création et de déploiement de vos containers et suivre l'état de votre application.

Consignes:

1- Proposer deux architectures utilisant kubernetes pour votre application.

3 - Expliquer le fonctionnement des différents services dans l'infrastructure kubernetes.

2 - Créer les différentes ressources (pods, déploiements, services) de votre application.

4- Automatiser le déploiement de votre application avec l'outil Scaffold.

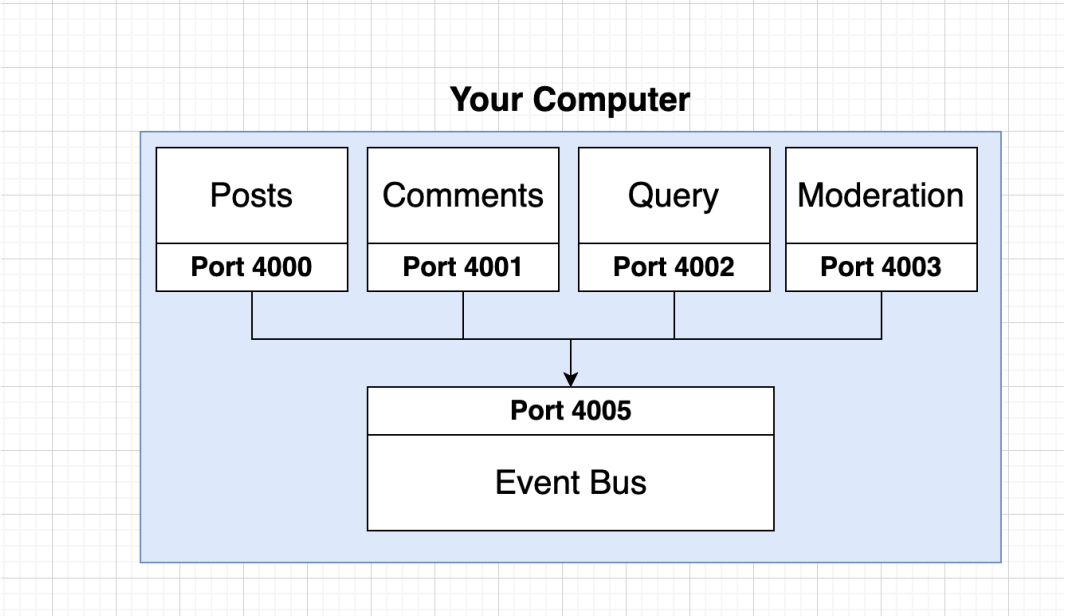
Consignes: Tp en groupe de 5.

Livrables:

- Un document qui présente votre processus de résolution de problèmes est envoyé à l'adresse. *moafembe@gmail.com*
- Le lien github de la solution .

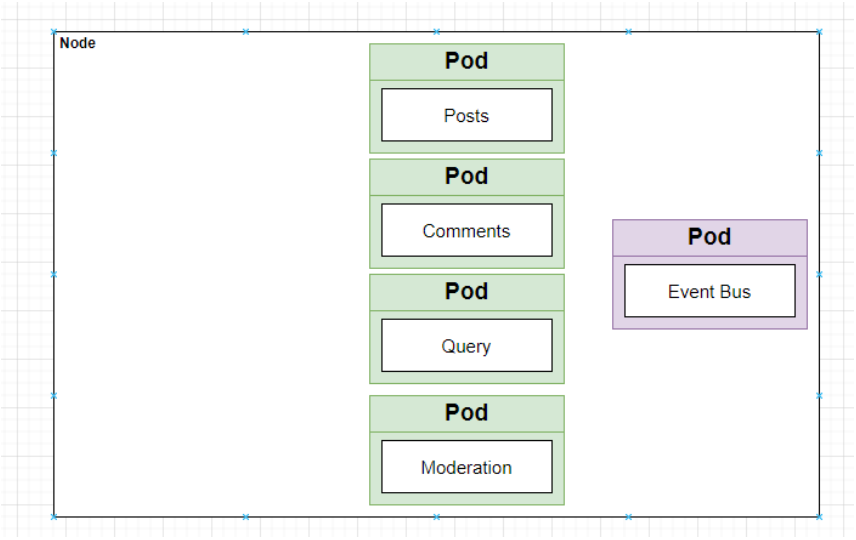
lien repo: https://github.com/atemengue/tp-blog

Orientation



L’orchestration des conteneurs de l’application microservice se fait en 3 niveaux. Les pods peuvent communiquer en utilisant les services nodeport, clusterIp et loadbalancer.

Niveau 1: Communication des services dans l’infrastructure



Le premier niveau consiste à faire communiquer les services de l’application dans le cluster kubernetes. Il est important de noter que tous les pods à l'intérieur du Node ne communiquent pas ensemble tant qu'une ressource services n’est pas définie pour établir la communication. Vous avez le choix entre l’utilisation des nodeport ou

des clusterip pour créer la communication. Utilisez les schéma ci-dessous pour faire fonctionner votre application. Vous expliquerez dans votre rapport pourquoi les nodeport ne sont pas une bonne solution.

Schéma 1: NodePort

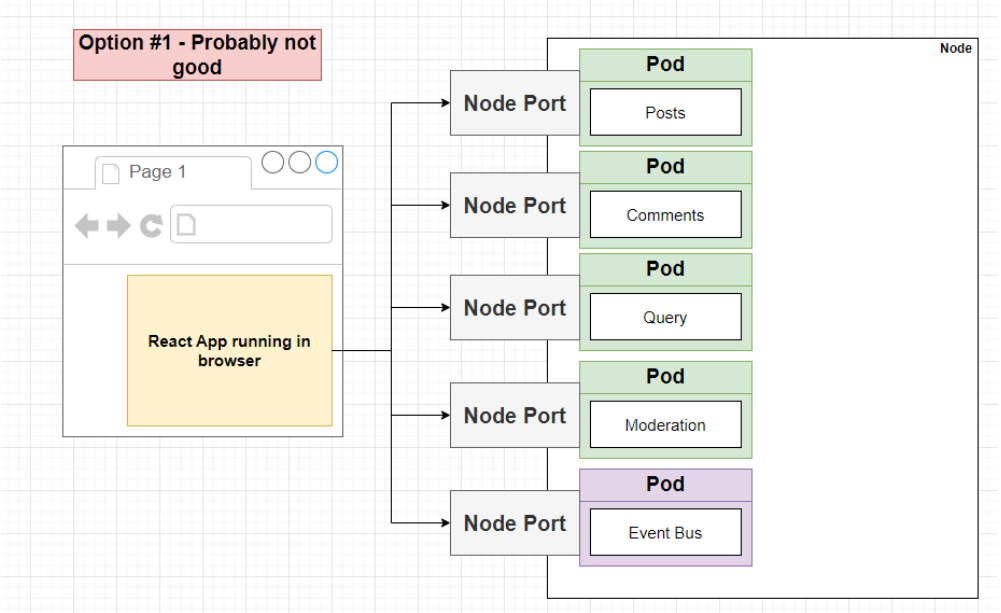
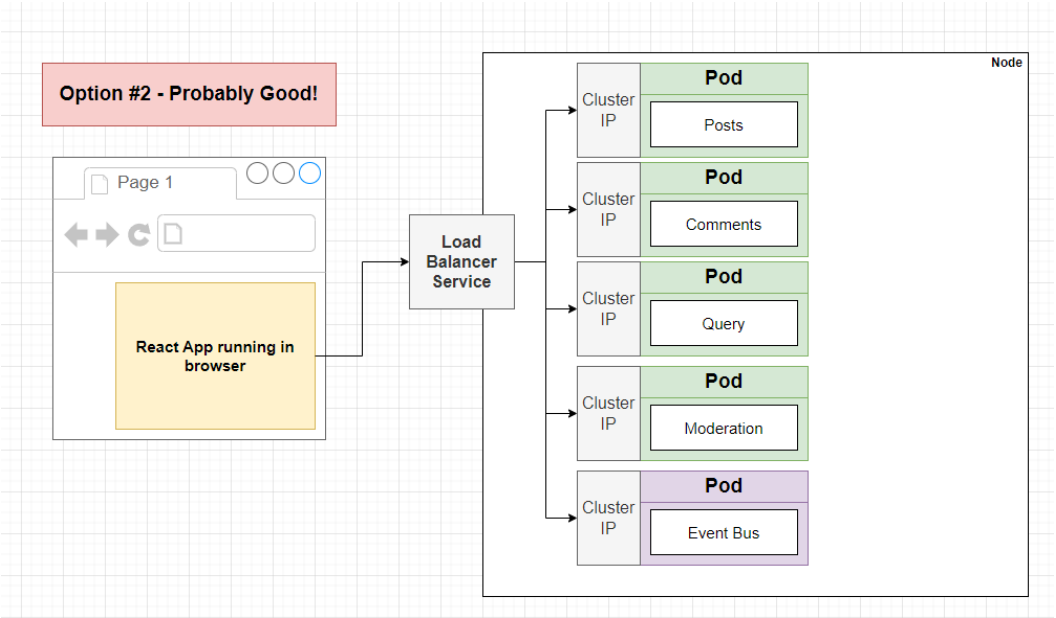


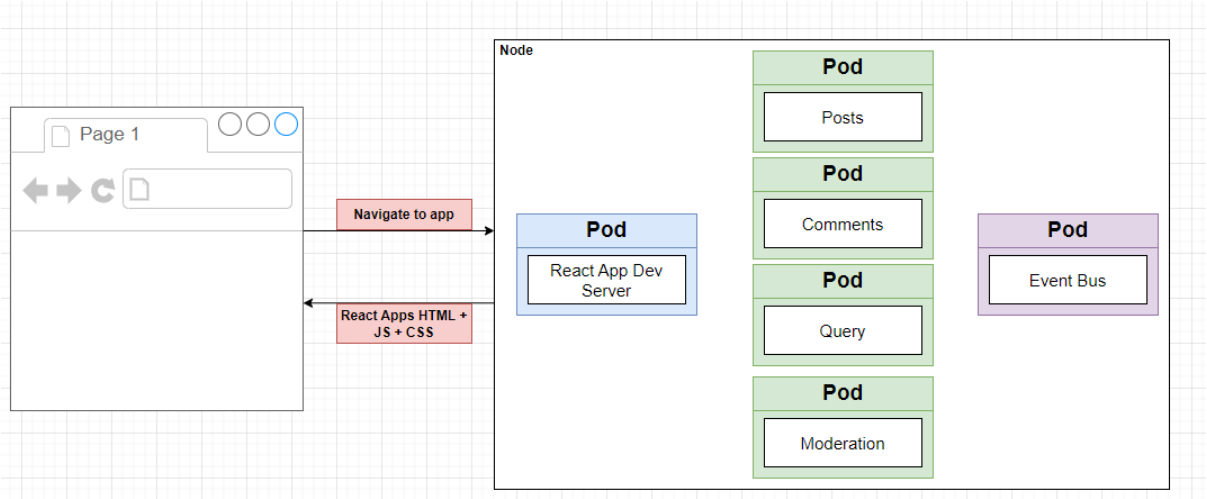
Schéma 2: ClusterIp



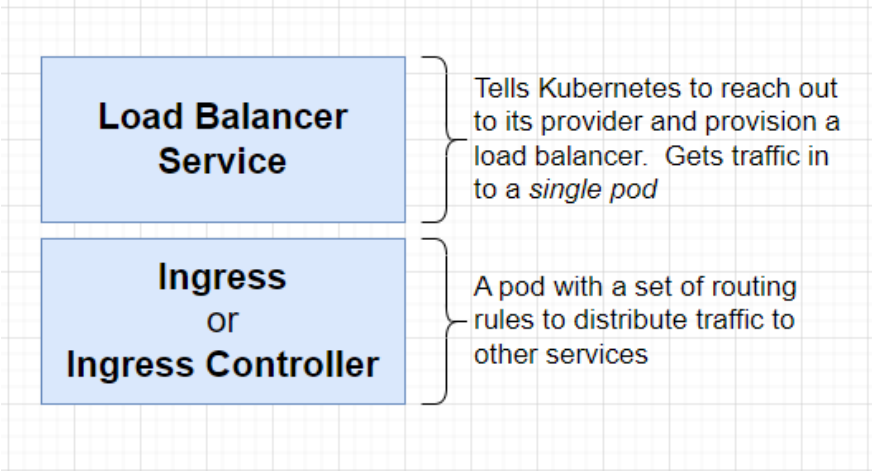
Niveau 2: Front-React

l’application front-end react peut être insérée dans le cluster kubernetes et fonctionner avec les autres services ou alors rester à l'extérieur du cluster.

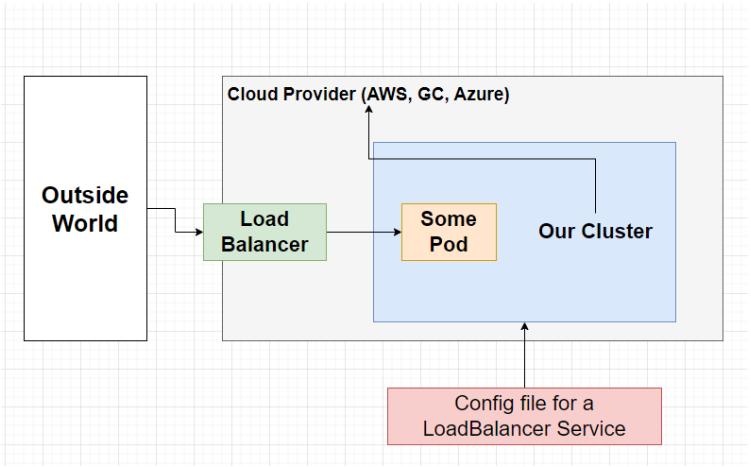
- Si elle est insérée dans le cluster, il faut trouver un moyen de la faire communiquer avec les autres pods. Vous allez faire face à un problème pour les requêtes, car le pod votre application react ne renvoie que le code html, css et javascript lorsqu’il est appelé par le navigateur.

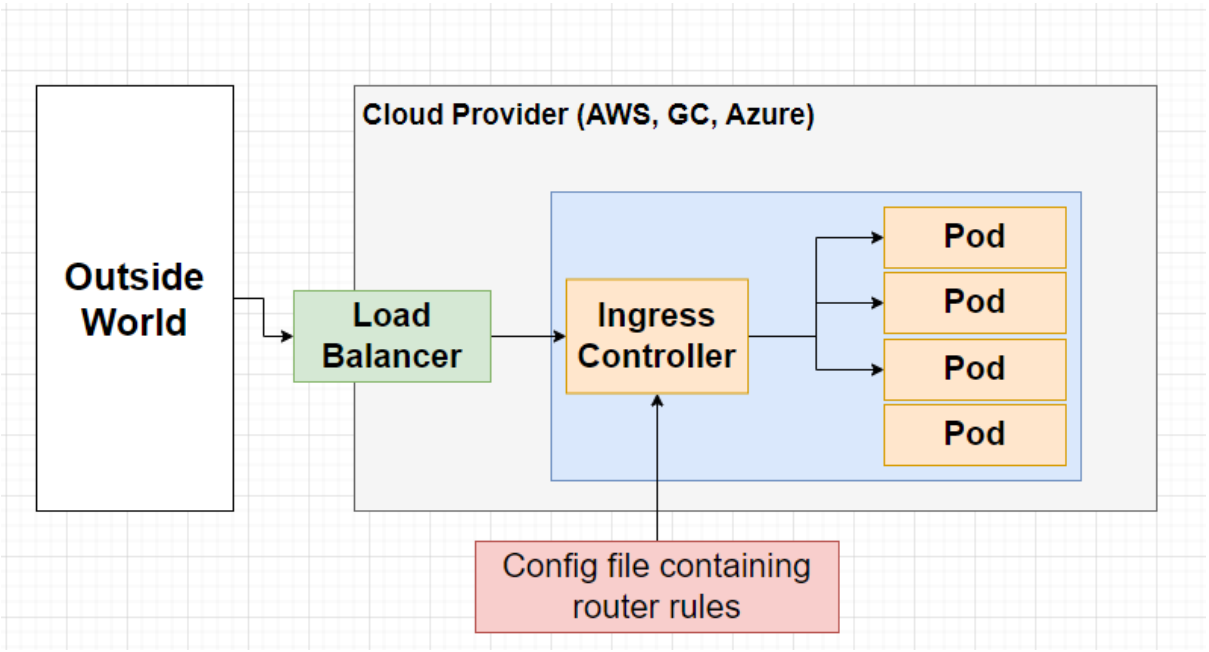


- **La meilleure solution.**



Vous avez la possibilité d’utiliser le load-balancer ou le ingress-controller pour faire communiquer vos services. Les services cloud provider disposent des load balancer qui permet d’entrer dans l'infrastructure cloud.





Pour ce projet nous allons utiliser ***ingress-nginx*** qui permettra de créer un load balancer et un ingress-controller pour nous.

NB: Suivre les instructions suivantes.

Si vous avez installé ***Nginx-ingress,*** veuillez supprimer les configuration et installer ***Ingress-Nginx*** a la place. Les deux bibliothèques sont différentes.

Veuillez vérifier que vous installez bien ***Ingress Nginx*** et non Nginx Ingress, qui est une bibliothèque totalement différente et incompatible.

Note - Les étudiants sous Windows doivent utiliser Docker Desktop avec WSL2 et non Minikube. Les instructions relatives à Minikube fournies ci-dessous s'adressent uniquement aux utilisateurs de Linux.

Installation - Docker Desktop (macOS et Windows WSL2)

https://kubernetes.github.io/ingress-nginx/deploy/#quick-start

Installation - Mini Kube (Linux)

Installation - Docker Desktop (macOS and Windows WSL2)

Quick start ¶

If you have Helm, you can deploy the ingress controller with the following command:

```
helm upgrade --install ingress-nginx ingress-nginx \
  --repo https://kubernetes.github.io/ingress-nginx \
  --namespace ingress-nginx --create-namespace
```

It will install the controller in the `ingress-nginx` namespace, creating that namespace if it doesn't already exist.

Info

This command is *idempotent*:

- if the ingress controller is not installed, it will install it,
- if the ingress controller is already installed, it will upgrade it.

If you don't have Helm or if you prefer to use a YAML manifest, you can run the following command instead:

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/contro
```

https://kubernetes.github.io/ingress-nginx/deploy/#quick-start

Installation - Mini Kube (Linux)

Environment-specific instructions ¶

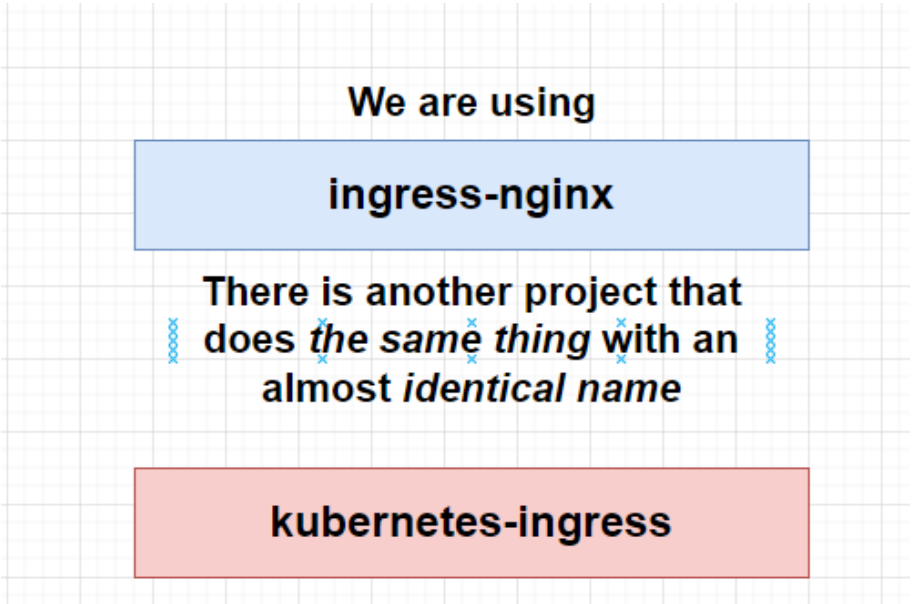
Local development clusters

minikube

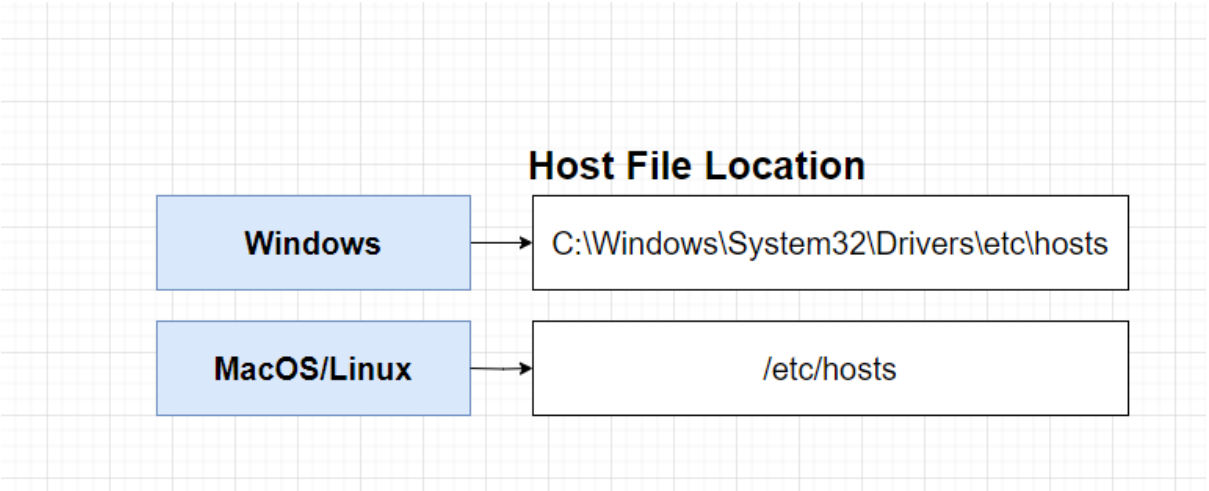
The ingress controller can be installed through minikube's addons system:

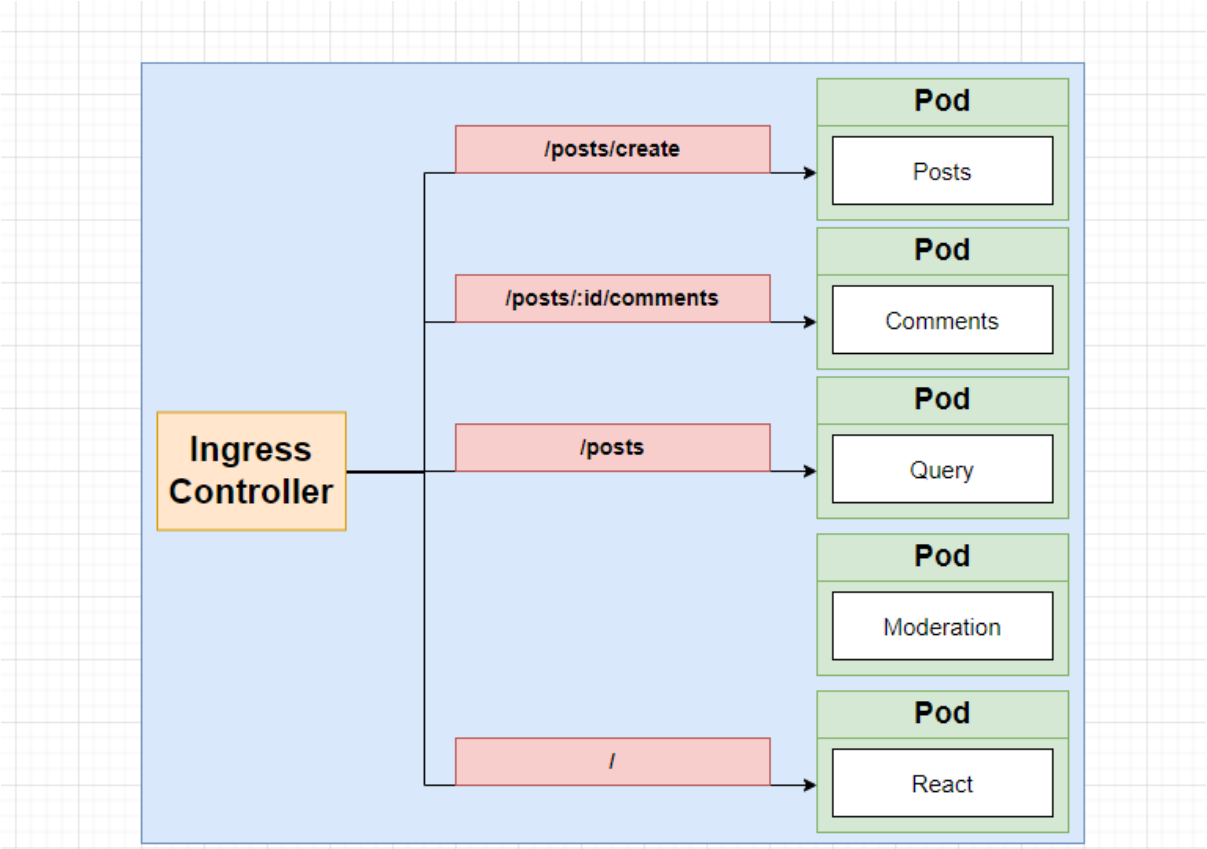
```
minikube addons enable ingress
```

https://kubernetes.github.io/ingress-nginx/deploy/#minikube



- Ajouter un host local pour votre configuration ingress-nginx.





Niveau 3: Automatiser la création de votre infrastructures en utilisant scaffold. (Expliquer la configuration du fichier skaffold.yaml)

NB: Utiliser la programme **orientation** du depot github et consulter le fichier dockerfiler du client pour corriger les erreurs de compatibilite de l'application react (creact-react-app) dans un cluster kubernetes

lien repo: https://github.com/atemengue/tp-blog