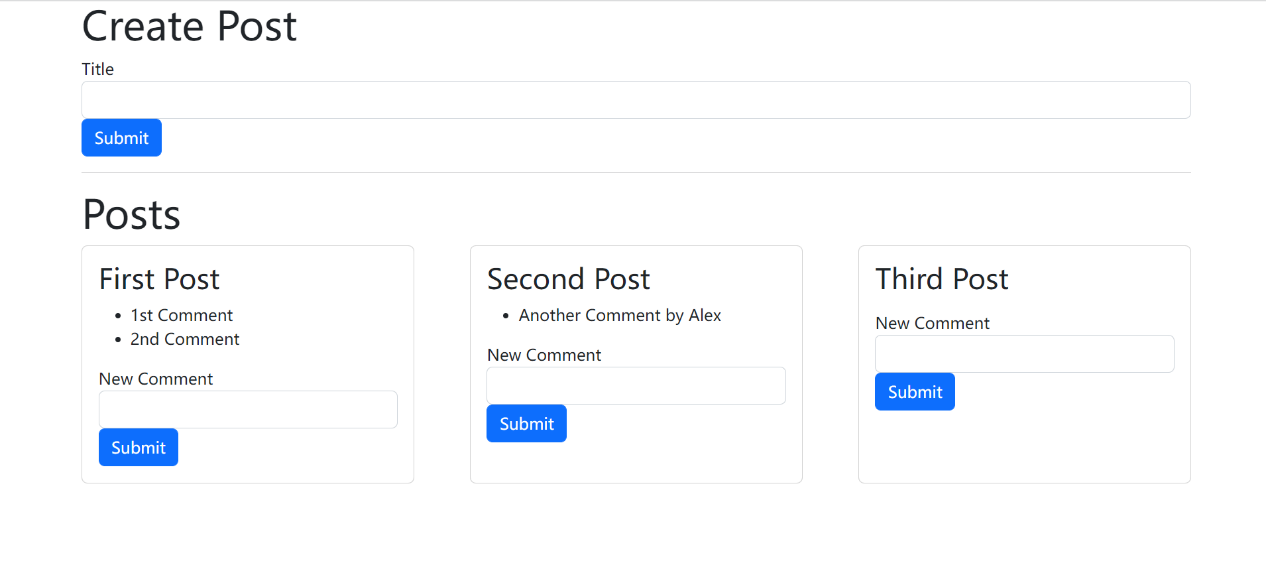
This basic blog app is a react application with a microservice architecture, using an asynchronized communication. Below is a mockup of this app:



It’s my very first attempt to practice microservices architecture, Docker and Kubernetes and so these will be the main focus. The project will include:

* Building docker images and push to online repo
* Building and checking Kubernetes pods, deployments, and services
* Concept of asynchronized service communication using events and event-bus
* Automatic update of changes to microservices and Kubernetes objects with nodemon and scaffold.
* A basic use of useState react hooks

We will explicitly ignore and defer the following features/aspects of a good app to my next practice project:

* Use of External databases. Each microservice will instead use in-memory database in the form of a dict or list within the source code
* Use of production-grade Event-bus. A basic home-made one is written.
* Concurrent event handling
* Deploying to cloud providers such as AWS, GC, or Azure
* Automation tests. Postman and Kubernetes NodePort were used to quickly test the app
* Typescript (use of class, type inference and etc.)
* Automatic refresh the page to present the content created/updated

Microservices Available:

**Posts**: allows user to create a new post

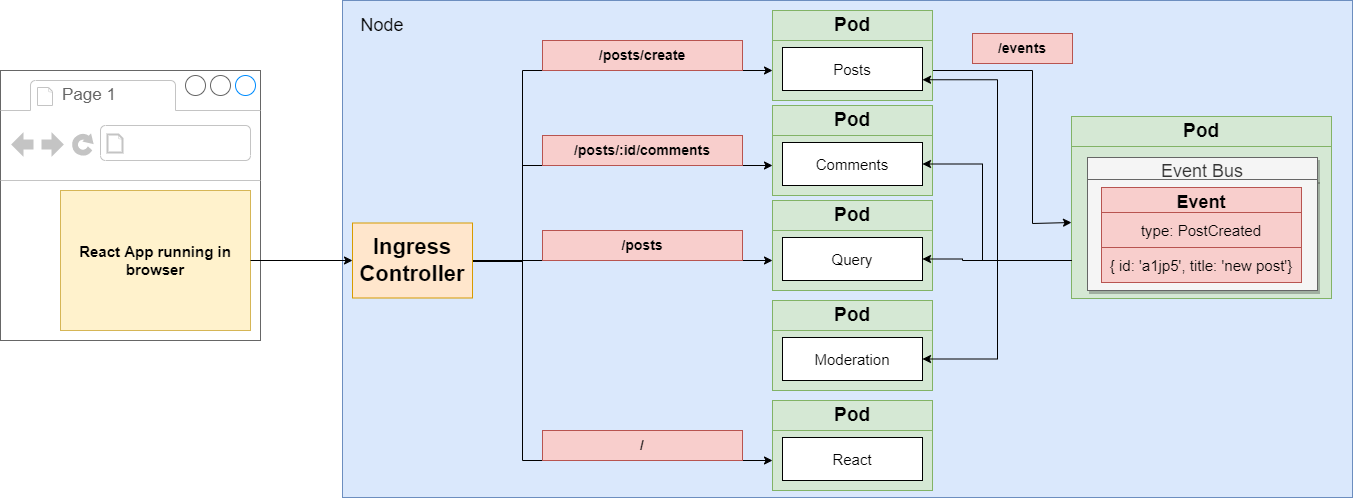
**Comments**: allow user to add comment to an existing post

**Query**: contain posts and comments for react-client to request

**Moderation**: filters restricted keywords in comment

**Event-Bus**: retrieves event emitted by other services and broadcast to all running services

Below is a schematic interaction between these services within the Kubernetes cluster.



Getting started:

With Kubernetes installed, user can go to the infra/k8s directory and run “kubctl apply -f . “ to start all Kubernetes objects in this project. After that the user can head to the local machine’s port 3000 to see the entry react page.