Kubernetes is an open-source platform for automating deployment, scaling, and operations of application containers across clusters of hosts, providing container-centric infrastructure. It was originally designed by Google. It has many features which are especially useful for applications running in production, like service naming and discovery, load balancing, application health checking, horizontal auto-scaling, and rolling updates. There are several important concepts around Kubernetes we should know before going into the sample.

**Pod** – This is the basic unit in Kubernetes. It can consist of one or more containers that are guaranteed to be co-located on the host machine and share the same resources. All containers deployed inside pod can see other containers via localhost. Each pod has a unique IP address within the cluster

**Service** – This is a set of pods that work together. By default, a service is exposed inside a cluster but it can also be exposed onto an external IP address outside your cluster. We can expose it using one of four available behaviors: ClusterIP, NodePort, LoadBalancer and ExternalName.

**Replication Controller** – This is a specific type of Kubernetes controller. It handles replication and scaling by running a specified number of copies of a pod across the cluster. It is also responsible for pod replacement if the underlying node fails.

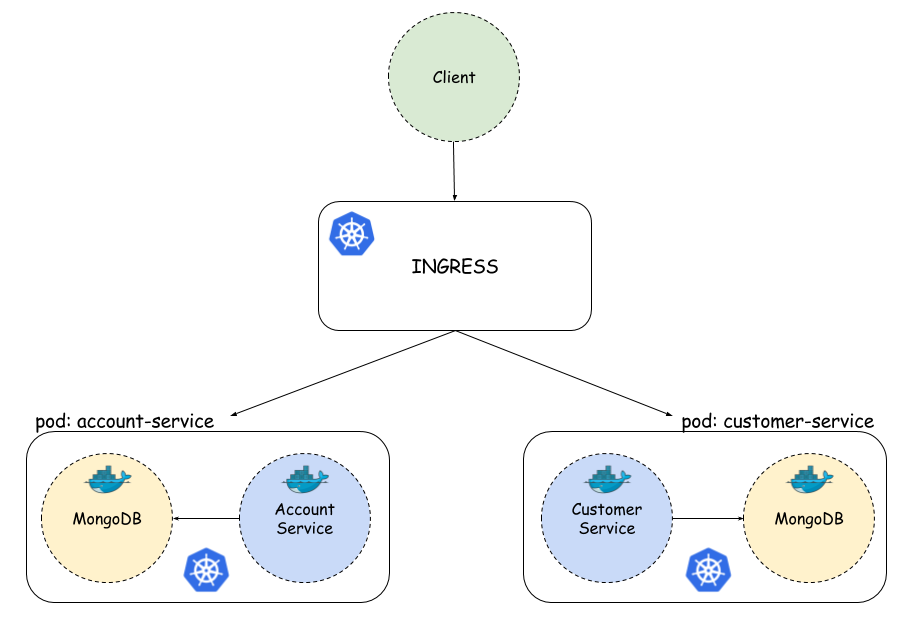
**Minikube**

The configuration of highly available Kubernetes cluster is not an easy task to perform. Fortunately, there is a tool that makes it easy to run Kubernetes locally – Minikube. It can run a single-node cluster inside a VM, which is really important for developers who want to try it out. The beginning is really easy. For examples on Windows, you have to download *minikube.exe* and *kubectl.exe* and add them to the PATH environment variable. Then you can start it from the command line using the minikube start command and use almost all of Kubernetes features available by calling the kubectl command.  An alternative for the command line option is Kubernetes Dashboard. It can be launched by calling the minikube dashboard command. We can create, update, or delete deployment from the UI dashboard, and also list and view a configuration of all pods, services, ingresses, replication controllers, etc.

Gateway (Zuul) and discovery (Eureka) for Spring Boot services are not required, because such mechanisms available on Kubernetes out of the box.

Here’s a picture illustrating the architecture of the presented solution. Each microservice’s pod consists of two containers: first with microservice application, and second with Mongo database. Account and customer microservices have their own database where all data is stored. Each pod is exposed as a service and can by searched by name on Kubernetes.

We also configure **Kubernetes Ingress**, which acts as a gateway for our microservices.



Create the docker images

1. Login to azure container registry

docker login -u syndemoregistry -p "5//b=aG3Oo+/+ZBQhQSXCc=FEHA/Fv9w" syndemoregistry.azurecr.io

1. Build the images for customer-service and account-image

Docker build -t syndemoregistry.azurecr.io/customer-service .

Docker build -t syndemoregistry.azurecr.io/account-service .

1. Upload the images to azure CR

Docker push syndemoregistry.azurecr.io/account-service

Docker push syndemoregistry.azurecr.io/customer-service

Create kubectl secret to access the private registry (syndemoregistry.azurecr.io)