Deploy:

* Do a build on all the projects that are being deployed
* Write a DockerFile or Docker-compose file for each project and in a cmd prompt window, navigate to the folder of the project and run docker build –t <image\_name:version> .
* Docker will then locate and run the DockerFile to make the image with the jar file that was built earlier.
* To deploy the image in Kubernetes as a deployment and service, there are two options:
  + First method is to run the command kubectl run <deployment\_name> --image=<image\_name:version> --port=<port> and this will create the deployment along with a pod. Then to expose the deployment as a service, run command kubectl expose deployment <deployment\_name> --type=”type”.
    - There are different type of services. The default type is ClusterIP, meaning it is only exposed internally and cannot be accessed from outside the VM. There is also “NodePort” type which is to expose externally and can be accessed from local browser. (Method for accessing will be described later). There is also “LoadBalancer” type with if the service has multiple deployments (ex. Multiple versions), the load balancer service will balance the incoming traffic and direct it to a certain version, depending on the rule specified for traffic routing.
  + The other method is to make a deployment.yaml file with the necessary information to make a deployment and a service with the imaged created earlier.
    - For making the deployment (kind = deployment) and the service (kind = service), specify name and label so they can be identified by other services with the label.
    - For the deployment section, the number of replicas can be set if one wishes to deploy multiple instances of the app. Also need to specify the name and version of the image that is to be used, along with the port for this container.
    - For the service section. Specify the deployment that is being exposed with spec.selector.app=<deployment label>. Also include the port and protocol for accessing the service.
* To use all the features in Istio, the Kubernetes cluster must have all the istio services implemented. To do so, download the istio folder and navigate to its directory. Set the Istioctl in the bin folder as path variable. Then install the istio crds (custom resource definition) with kubectl apply –f install/kubernetes/helm/istio/templates/crds.yml. Then to install the istio without mTLS authentication, run kubectl apply –f install/kubernetes/istio-demo.yml. Now, all the features should be installed in the istio-system namespace (can be accessed by adding the tag “–n istio-system” after commands)
* When using ingress gateway in deploying with Istio, the ingress host is (minikube ip) and port is found in (kubectl describe service istio-ingressgateway) under the port with name http2. The secured host is under the port with name https.
* Go to INGRESS\_HOST:INGRESS\_PORT/productpage to access the application. (/productpage is endpoints specified in by the ingress gateway yaml file)
* Istioctl kube-inject –f <input\_filename> -o <output\_filename> takes the input file and inject istio sidecar envoy into the service. Kubectl create –f <outpu\_filename> then takes the file and create a service with it.