1. Building project JAR

* In Intellij, or any IDE of choice, navigate to the project folder in the built-in terminal
* Run a clean build on the project:
  + gradlew clean build(Gradle project), mvn clean install(Maven project)

1. Making Docker image

* Open a new terminal, cmd or powershell will do, and navigate to the project folder
* Make sure there is a Dockerfile in the folder
  + For references on how to write Dockerfile, visit https://docs.docker.com/engine/reference/builder/#usage
* Run the following command in the terminal window: docker build -t : .
  + Replace the name and version for the project and make sure to include the dot at the end to specify the location of Dockerfile
* To check if the docker image has been build properly, run the following command: docker images
  + Make sure the image shows up in the docker images list with the correct name and version

1. Building Kubernetes pod with the Docker image, injected with Istio sidecar

* Make sure Kubernetes is running(minikube start) and the docker environment is running(minikube docker-env | Invoke-Expression)
* Make sure Istio is installed onto the Kubernets cluster
  + For referencs on the istallation, visit https://istio.io/docs/setup/kubernetes/quick-start/
* Make a deployment.yaml file that will create a deployment and service using the image
  + For references on how to write the deployment.yaml file, visit https://www.baeldung.com/spring-boot-minikube
* If Istio sidecar injection is not enabled, manually inject the sidecar with the following command: istioctl kube-inject -f deployment.yaml -o deployment-istio.yaml
* Once Istion sidecar has been injected, deploy into Kubernetes with the following command: kubectl create -f deployment-istio.yaml
  + If the yaml file is updated, use the following command to also update the deployments in Kubernetes without redeploying:
    - kubectl apply -f deployment-istio.yaml
* To check if the deployment is successful, run the following: kubectl get pods
  + There should be 2 pods for each deployment, one is the actual pod and one is the Envoy proxy
* Also can use kubectl get services and kubectl get deployments to check the services has been set up properly

1. Setting up Ingress gateway and virtual service

* Make sure project-gateway.yaml file is present in the project folder
  + For references on gateway yaml file, check out the istio/samples/bookinfo/networking/bookinfo-gateway.yaml in the Istio installation folder
* Use kubectl create -f project-gateway.yaml to deploy the virtual service and gateway
* To check for gateway IP, run the following command: minikube ip
* To check for port, run the following command: kubectl get services -n istio-system istio-ingressgateway
  + Check the port that is being forwarded by the gateway to the project port
  + Can choose to modify the project port to one of the default forwarded port in ingress, ex. port 80 forwards to 31380 in Ingress gateway
* To test the gateway, visit the following url in a browser: http://:

# Error handling

Minikube Error: UNAVAILABLE:no healthy upstream

UNAVAILABLE:upstream connect error or disconnect/reset before headers

run: bash minikube update-context

to update the context. However, most of the time, these above 2 errors are cuased because the pod you are trying to access has not finished initialization. Just wait for a little bit will usually do the trick. If the service was not initialized, you would not even make it to this page. It will display a cannot connect or default 404 error page.

To Test if an cluster internal location can be accessed, one can run the following command.