**Описание / план**

* Собираются метрики и строятся дашборды для двух методов сервиса Product Service - Health и GetAllProducts.
* Метод Health возвращает ответ сразу, метод GetAllProducts возвращает продукты из БД, поэтому latency второго в 2-3 раза больше, чем для первого.
* Для сбора / публикации метрик использовал ASP.NET Core экспортер - <https://github.com/prometheus-net/prometheus-net>.
* Для генерации нагрузки используется утилита Apache Benchmark (AB).  
  Сначала в цикле отправлял по 100 запросов в 10 потоков, для срабатывания алёртов нагрузку резко увеличил.

ab.exe -n 5000 -c 100 http://arch.homework/products/health

ab.exe -n 5000 -c 100 http://arch.homework/products

AB взята из скачанного пакета установки Apache.

* В hosts для arch.homework прописан IP, полученный через minikube ip.
* Не получалось вызывать ошибки, увеличивая нагрузку (видимо, из-за ограниченных возможностей ab), поэтому в Product Service зашил логику, чтобы запросы, приходящие в момент времени с количеством секунд, кратным 5ти, сразу бы генерировали 500 Internal Server Error.
* Prometheus и Graphana, а также & NGINX Ingress, установлен в namespaces monitoring, Product Service – в namespace otusapp. Встроенный в Kubernetes addon ingress предварительно отключён (так как для него нет встроенного сервис монитора для сбора и предоставления метрик).  
    
  kubectl create namespace monitoring  
  kubectl config set-context --current --namespace=monitoring

helm install prom google/prometheus-operator -f prometheus.yaml –atomic  
helm install nginx google/nginx-ingress -f nginx-ingress.yaml –atomic  
  
kubectl create namespace otusapp  
kubectl config set-context --current --namespace= otusapp

helm install app app-chart

* Для возможности открытия Prometheus GUI и Grafana GUI проброшены порты на локальный хост:

kubectl port-forward service/prom-prometheus-operator-prometheus 9090

kubectl port-forward service/prom-grafana 9000:80

* GUI открываем соответственно как <http://localhost:9090> и <http://loclahost:9000>.
* Для проведения более интересных экспериментов с метриками и дашбордами Product Service задеплоен на два инстанса / пода.
* Два дашборда сделаны по заданию (по методам сервиса и с ingress контроллера).  
  Третий, более подробный (с более подробной информацией по инстансам и методам), – больше для себя, хотелось поэкспериментировать с вариантами представления информации.
* В дашборде, отслеживающим метрики с ingress контроллера, настроены два алёрта:
  + если Error Rate превышает 30 ошибок в секунду;
  + если Latency (99й процентиль) превышает 250 миллисекунд.
* Были сложности с настройкой SMTP в Grafana, поэтому отправку алёртов настроил на специально созданный канал в Slack (через webhook).