Микросервисная архитектура и контейнеризация

Урок 7. Продвинутые абстракции

https://github.com/adterskov/geekbrains-conteinerization/tree/master/homework/7.advanced-abstractions

Меняем ns на default:

```
kubectl config set-context --current --namespace=default
```

Создаем ConfigMap **ConfigMap.yaml**:

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: default
data:
  prometheus.yml: |
    global:
      scrape_interval: 30s

    scrape_configs:
      - job_name: 'prometheus'
        static_configs:
        - targets: ['localhost:9090']

      - job_name: 'kubernetes-nodes'
        kubernetes_sd_configs:
        - role: node
        relabel_configs:
        - source_labels: [__address__]
          regex: (.+):(.+)
          target_label: __address__
          replacement: ${1}:9101
```

Создаем объекты для авторизации Prometheus сервера в Kubernetes-API **PrometheusAuthObj.yaml**:

```yaml
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: prometheus
  namespace: default
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: prometheus
rules:
- apiGroups: [""]
  resources:
  - nodes
  verbs: ["get", "list", "watch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: prometheus
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: prometheus
subjects:
- kind: ServiceAccount
  name: prometheus
  namespace: default
```

Создаем StatefulSet для Prometheus сервера из образа prom/prometheus:v2.19.2 с одной репликой **StatefulSet.yaml**:

```yaml
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: prometheus
  namespace: default
spec:
  serviceName: prometheus
  replicas: 1
  selector:
    matchLabels:
      app: prometheus
  template:
    metadata:
      labels:
        app: prometheus
    spec:
      serviceAccount: prometheus
      terminationGracePeriodSeconds: 10
      initContainers:
      - image: busybox
        name: mount-permissions-fix
        command: [ "sh", "-c", "chmod 777 /prometheus" ]
        volumeMounts:
          - name: data
            mountPath: /prometheus
      containers:
      - name: prometheus-k8s
        image: prom/prometheus:v2.19.2
        ports:
          - protocol: TCP
            containerPort: 9090
        imagePullPolicy: IfNotPresent
        volumeMounts:
          - name: config
            mountPath: /etc/prometheus
          - name: data
            mountPath: /prometheus
      volumes:
        - name: config
          configMap:
            name: prometheus-config
  volumeClaimTemplates:
    - metadata:
        name: data
      spec:
        accessModes: ["ReadWriteOnce"]
        resources:
```

```
          requests:
            storage: 5Gi
        storageClassName: csi-ceph-hdd-dp1
```

Создаем service и ingress для этого стейтфулсета, так чтобы запросы с любым доменом на белый IP вашего сервиса nginx-ingress-controller (тот что в нэймспэйсе ingress-nginx с типом LoadBalancer) шли на приложение:

Ingress.yaml

```
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: prometheus
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  rules:
  - http:
      paths:
      - path: "/"
        pathType: Prefix
        backend:
          service:
            name: prometheus
            port:
              number: 80icePort: 80
```

Service.yaml

```
---
kind: Service
apiVersion: v1
metadata:
  name: prometheus
  labels:
    app: prometheus
spec:
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9090
  selector:
    app: prometheus
```

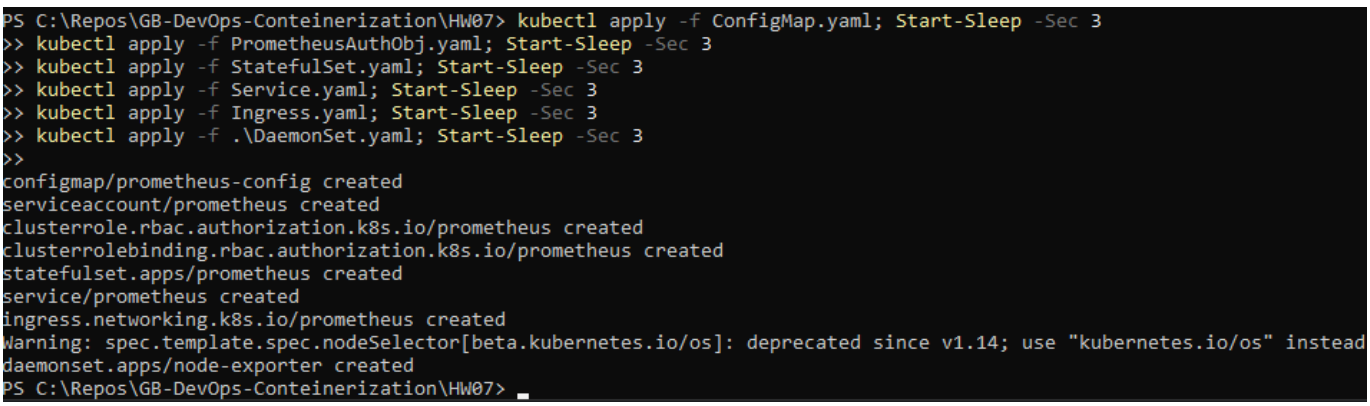
Создаем DaemonSet node-exporter **DaemonSet.yaml**:

```yaml
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  labels:
    app: node-exporter
  name: node-exporter
spec:
  updateStrategy:
    rollingUpdate:
      maxUnavailable: 1
    type: RollingUpdate
  selector:
    matchLabels:
      app: node-exporter
  template:
    metadata:
      labels:
        app: node-exporter
    spec:
      containers:
      - args:
        - --web.listen-address=0.0.0.0:9101
        - --path.procfs=/host/proc
        - --path.sysfs=/host/sys
        - --collector.filesystem.ignored-mount-
points=^/(dev|proc|sys|var/lib/docker/.+)($|/)
        - --collector.filesystem.ignored-fs-
types=^(autofs|binfmt_misc|cgroup|configfs|debugfs|devpts|devtmpfs|fusectl|hugetlbfs|m
queue|overlay|proc|procfs|pstore|rpc_pipefs|securityfs|sysfs|tracefs)$
        image: quay.io/prometheus/node-exporter:v0.16.0
        imagePullPolicy: IfNotPresent
        name: node-exporter
        volumeMounts:
        - mountPath: /host/proc
          name: proc
        - mountPath: /host/sys
          name: sys
        - mountPath: /host/root
          name: root
          readOnly: true
      hostNetwork: true
      hostPID: true
      tolerations:
        - effect: NoSchedule
          operator: Exists
      nodeSelector:
        beta.kubernetes.io/os: linux
```

```
      volumes:
      - hostPath:
          path: /proc
          type: ""
        name: proc
      - hostPath:
          path: /sys
          type: ""
        name: sys
      - hostPath:
          path: /
          type: ""
        name: root
```

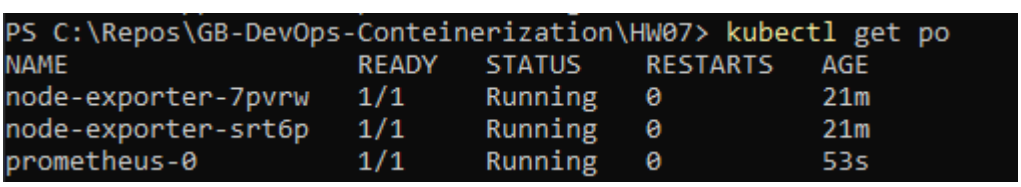
Разворачиваем созданые конфигурации:

```
kubectl apply -f ConfigMap.yaml
kubectl apply -f PrometheusAuthObj.yaml
kubectl apply -f StatefulSet.yaml
kubectl apply -f Service.yaml
kubectl apply -f Ingress.yaml
kubectl apply -f DaemonSet.yaml
```



Проверяем POD'ы и находим внешний адрес:

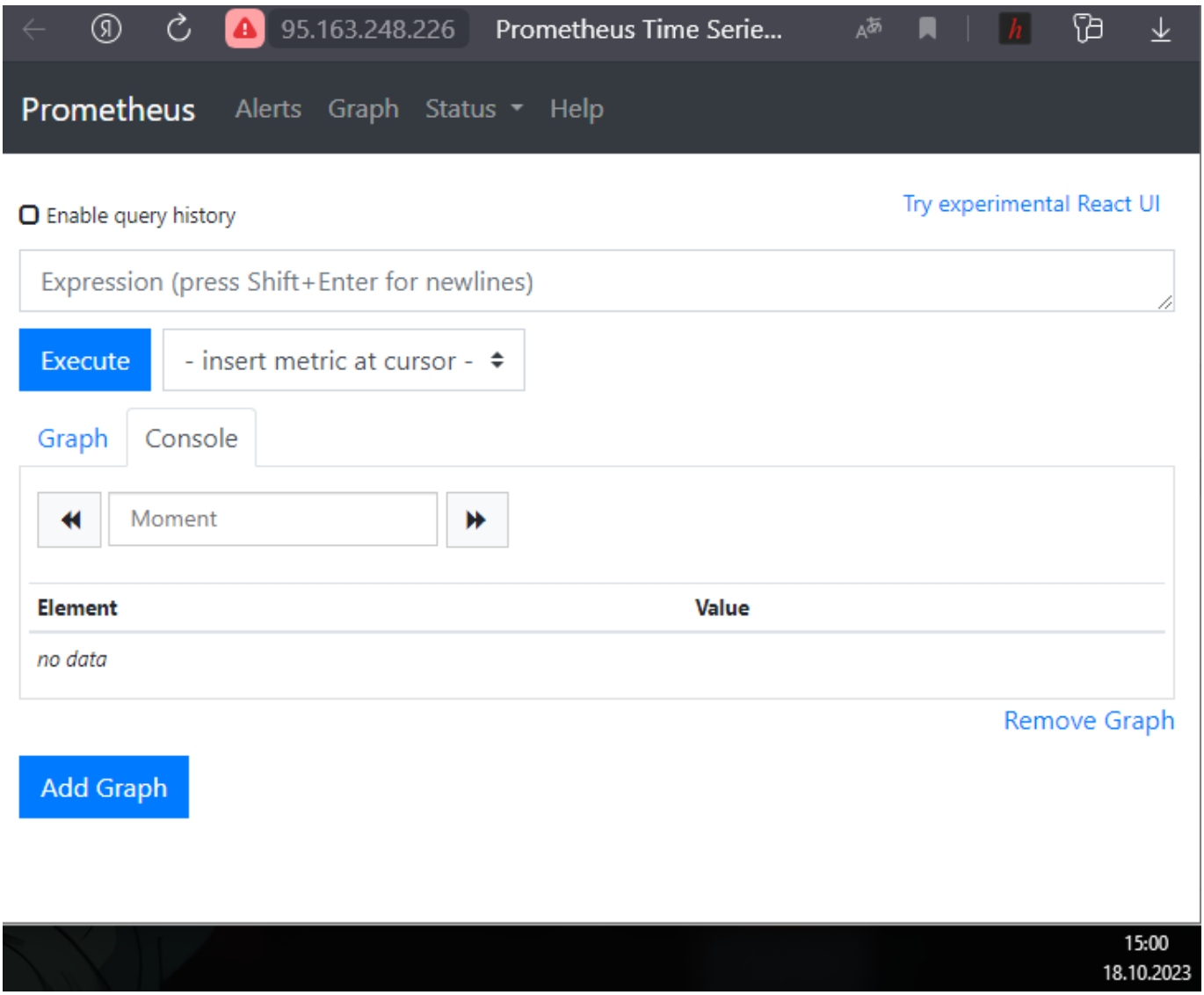
```
kubectl get po
```



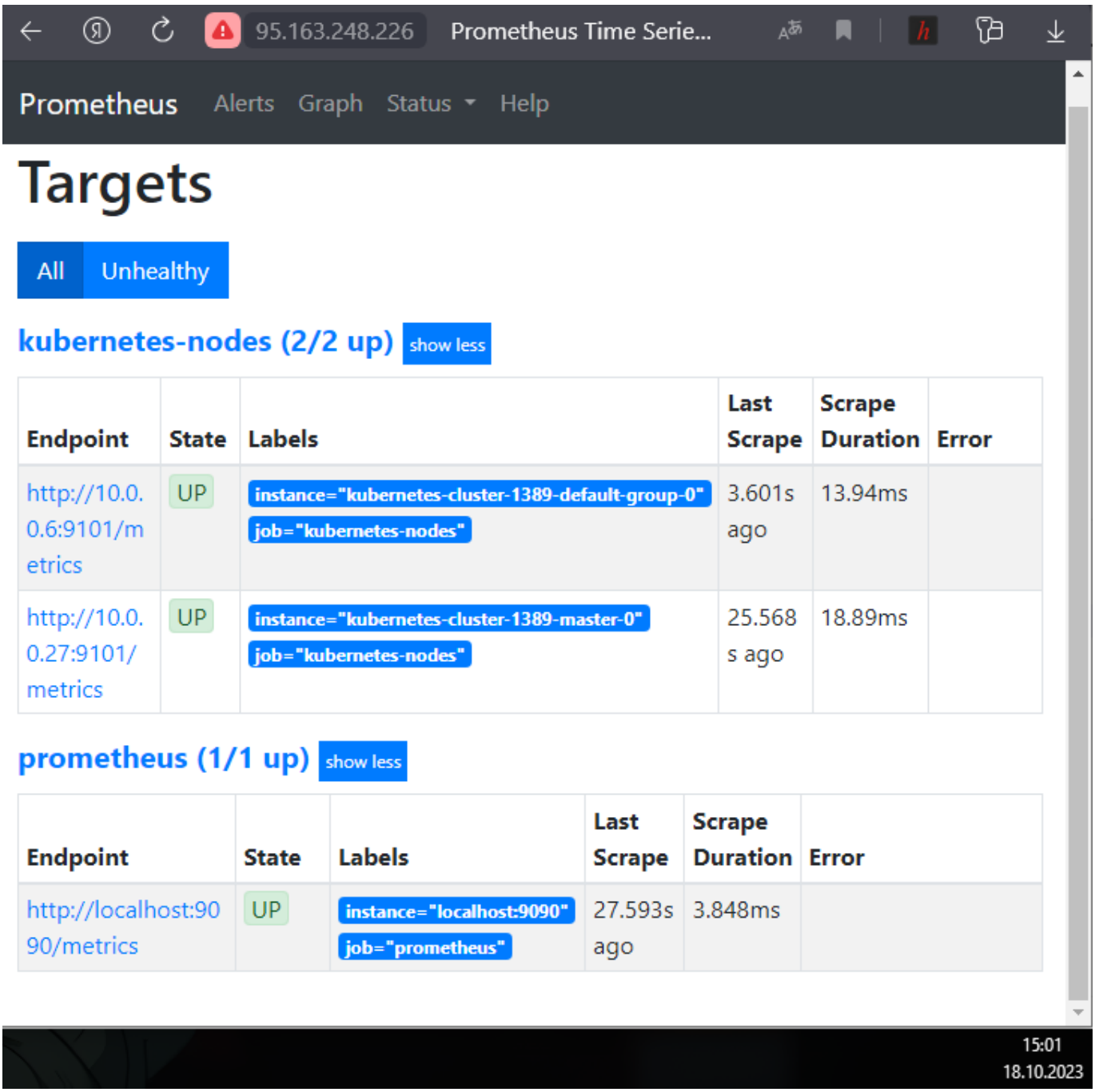
```
kubectl get svc -A
```

```
PS C:\Repos\GB-DevOps-Conteinerization\HW07> kubectl get svc -A
NAMESPACE              NAME                                 TYPE           CLUSTER-IP       EXTERNAL-IP
(S)                      AGE
default                kubernetes                           ClusterIP      10.254.0.1       <none>
TCP                      58m
default                prometheus                           ClusterIP      10.254.255.55    <none>
CP                       70s
ingress-nginx          ingress-nginx-controller             LoadBalancer   10.254.69.68     95.163.248.226.nip.io
0420/TCP,443:30097/TCP   33m
```

Проверяем:



Откроем в браузере интерфейс Prometheus, Status -> Targets Видим все ноды своего кластера, которые Prometheus смог определить и собирает с них метрики:



Так же смотрим на вкладке Graph запрос node_load1:


95.163.248.226
Prometheus Time Series Collection and Processi...
Prometheus
Alerts
Graph
Status
Help
1h
Until
Res. (s)
stacked
0.5
0.4
0.3
0.2
0.1
0
11:15
11:30
11:45
12:00
node_load1{instance="kubernetes-cluster-1389-master-0",job="kubernetes-nodes"}
node_load1{instance="kubernetes-cluster-1389-default-group-0",job="kubernetes-nodes"}
95.163.248.226
15:03
18.10.2023