# Продвинутые абстракции Kubernetes

> ! Задание нужно выполнять в нэймспэйсе default

Разверните в кластере сервер систему мониторинга Prometheus.

1.

Создайте в кластере ConfigMap со следующим содержимым:

```
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
```

2.

Создайте объекты для авторизации Prometheus сервера в Kubernetes-API.

2.1

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: prometheus
  namespace: default
```

2.2

```
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
```

```
metadata:
  name: prometheus
rules:
- apiGroups: [""]
  resources:
  - nodes
  verbs: ["get", "list", "watch"]
```

2.3

```
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: prometheus
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: prometheus
subjects:
- kind: ServiceAccount
  name: prometheus
  namespace: default
```

3.

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

3.1

В нем должнен быть описан порт 9090 TCP volumeClaimTemplate - ReadWriteOnce, 5Gi, подключенный по пути /prometheus Подключение конфигмапа с настройками выше по пути /etc/prometheus

3.2

Так же в этом стейтфулсете нужно объявить initContainer для изменения прав на вольюм data на 777. См пример из лекции 4: practice/4.resources-and-persistence/persistence/deployment.yaml

Не забудьте указать обязательное поле serviceName

3.3

Так же укажите поле serviceAccount: prometheus на одном уровне с containers, initContainers, volumes См пример с rabbitmq из материалов лекции.

4.

Создайте 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

5.

Проверьте что при обращении из браузера на белый IP вы видите открывшееся приложение Prometheus

6.

В этом же неймспэйсе создайте DaemonSet node-exporter как в примере к лекции: practice/7.advanced-abstractions/daemonset.yaml

7.

Откройте в браузере интерфейс Prometheus. Попробуйте открыть Status -> Targets Тут вы должны увидеть все ноды своего кластера, которые Prometheus смог определить и собирает с ним метрики.

8.

Так же можете попробовать на вкладке Graph выполнить запрос node_load1 - это минутный Load Average для каждой из нод в кластере.

1.

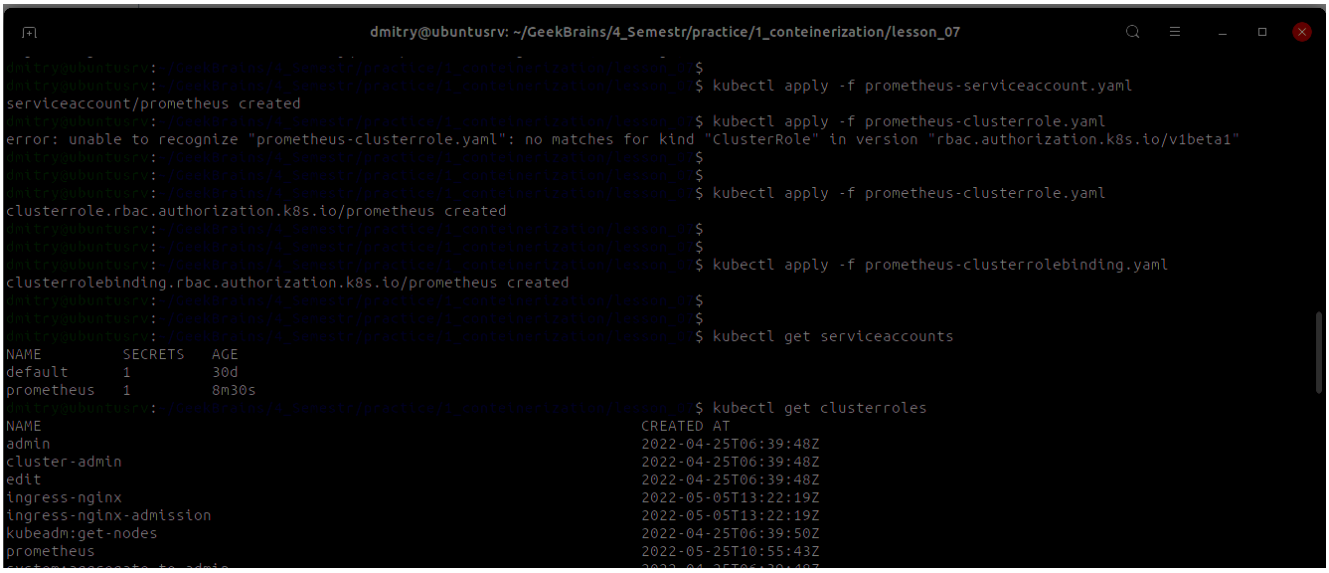
Создайте в кластере ConfigMap:

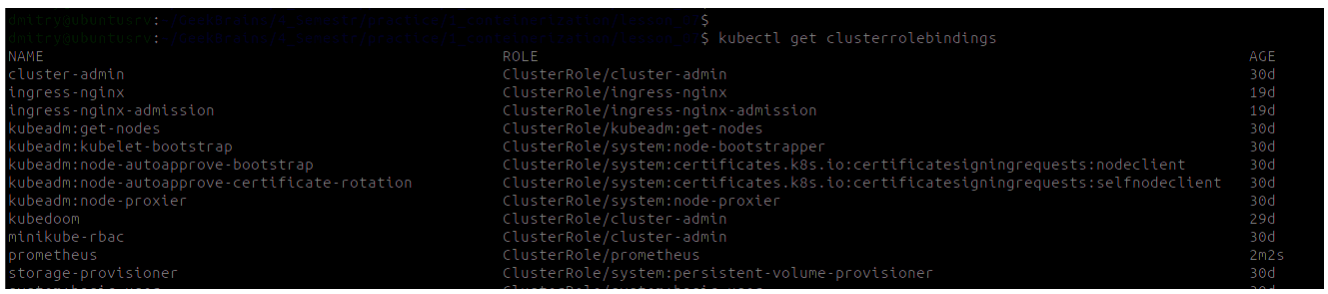
```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-configmap
data:
  prometheus.yml: |
    global:
      scrape_interval: 30s

    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
prometheus-configmap.yaml (END)
```

2.

Создайте объекты для авторизации Prometheus сервера в Kubernetes-API ( RBAC resources

The **rbac.authorization.k8s.io/v1beta1** API version of ClusterRole, ClusterRoleBinding, Role, and RoleBinding is no longer served as of v1.22. ):



```
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$ kubectl apply -f prometheus-pv.yaml
persistentvolume/my-prometheus created
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$ kubectl get pv
NAME            CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM              STORAGECLASS   REASON   AGE
my-prometheus   5Gi        RWO            Retain           Available                      local                   10s
my-rabbit       10Gi       RWO            Retain           Bound       default/files-db   local                   27d
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$
```

3.

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

```
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$ kubectl apply -f prometheus-configmap.yaml
configmap/prometheus-configmap created
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$ kubectl get configmaps
NAME                  DATA   AGE
kube-root-ca.crt      1      30d
prometheus-configmap  1      17s
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$
```

```
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$ kubectl apply -f prometheus-statefullset.yaml
statefulset.apps/prometheus created
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$ kubectl get statefulset
NAME         READY   AGE
prometheus   0/1     41s
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$ kubectl get pv
NAME            CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                       STORAGECLASS   REASON   AGE
my-prometheus   5Gi        RWO            Retain           Bound    default/data-prometheus-0   local                   40m
my-rabbit       10Gi       RWO            Retain           Bound    default/files-db            local                   27d
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$ kubectl get pvc
NAME                STATUS   VOLUME          CAPACITY   ACCESS MODES   STORAGECLASS   AGE
data-prometheus-0   Bound    my-prometheus   5Gi        RWO            local          75s
files-db            Bound    my-rabbit       10Gi       RWO            local          27d
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$
```

```
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$ kubectl apply -f prometheus-statefullset.yaml
statefulset.apps/prometheus created
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$ kubectl get statefulset
NAME         READY   AGE
prometheus   0/1     11s
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$ kubectl get pod
NAME                          READY   STATUS    RESTARTS        AGE
postgresql-65ddd4b575-cs7vz   1/1     Running   5 (5m51s ago)   28d
prometheus-0                  0/1     Running   0               18s
redmine-8bb48fcd4-jbxjh       1/1     Running   6 (5m51s ago)   20d
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$ kubectl logs prometheus-0
level=info ts=2022-05-26T12:03:38.045Z caller=main.go:302 msg="No time or size retention was set so using the default time retention" duration=15d
level=info ts=2022-05-26T12:03:38.046Z caller=main.go:337 msg="Starting Prometheus" version="(version=2.19.2, branch=HEAD, revision=c448ada63d83002e9c1d2c9f84e09f55a61f0ff7)"
level=info ts=2022-05-26T12:03:38.046Z caller=main.go:338 build_context="(go=go1.14.4, user=root@dd72efe1549d, date=20200626-09:02:20)"
level=info ts=2022-05-26T12:03:38.046Z caller=main.go:339 host_details="(Linux 4.19.202 #1 SMP Tue Feb 8 19:13:02 UTC 2022 x86_64 prometheus-0 (none))"
level=info ts=2022-05-26T12:03:38.046Z caller=main.go:340 fd_limits="(soft=1048576, hard=1048576)"
level=info ts=2022-05-26T12:03:38.047Z caller=main.go:341 vm_limits="(soft=unlimited, hard=unlimited)"
level=info ts=2022-05-26T12:03:38.050Z caller=main.go:678 msg="Starting TSDB ..."
level=info ts=2022-05-26T12:03:38.050Z caller=web.go:524 component=web msg="Start listening for connections" address=0.0.0.0:9090
level=info ts=2022-05-26T12:03:38.060Z caller=head.go:645 component=tsdb msg="Replaying WAL and on-disk memory mappable chunks if any, this may take a while"
level=info ts=2022-05-26T12:03:38.068Z caller=head.go:706 component=tsdb msg="WAL segment loaded" segment=0 maxSegment=1
level=info ts=2022-05-26T12:03:38.068Z caller=head.go:706 component=tsdb msg="WAL segment loaded" segment=1 maxSegment=1
level=info ts=2022-05-26T12:03:38.069Z caller=head.go:709 component=tsdb msg="WAL replay completed" duration=8.126154ms
level=info ts=2022-05-26T12:03:38.071Z caller=main.go:694 fs_type=TMPFS_MAGIC
level=info ts=2022-05-26T12:03:38.071Z caller=main.go:695 msg="TSDB started"
level=info ts=2022-05-26T12:03:38.071Z caller=main.go:799 msg="Loading configuration file" filename=/etc/prometheus/prometheus.yml
level=info ts=2022-05-26T12:03:38.073Z caller=kubernetes.go:253 component="discovery manager scrape" discovery=k8s msg="Using pod service account via in-cluster config"
level=info ts=2022-05-26T12:03:38.076Z caller=main.go:827 msg="Completed loading of configuration file" filename=/etc/prometheus/prometheus.yml
level=info ts=2022-05-26T12:03:38.076Z caller=main.go:646 msg="Server is ready to receive web requests."
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$ kubectl get pod
NAME                          READY   STATUS    RESTARTS       AGE
postgresql-65ddd4b575-cs7vz   1/1     Running   5 (9m5s ago)   28d
prometheus-0                  1/1     Running   0              3m32s
redmine-8bb48fcd4-jbxjh       1/1     Running   6 (9m5s ago)   20d
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$ kubectl get statefulset
NAME         READY   AGE
prometheus   1/1     4m44s
dmitrygubuntusrv:~/GeekBrains/4_Semestr/practice/1_contetnerization/lesson_07$
```

4.

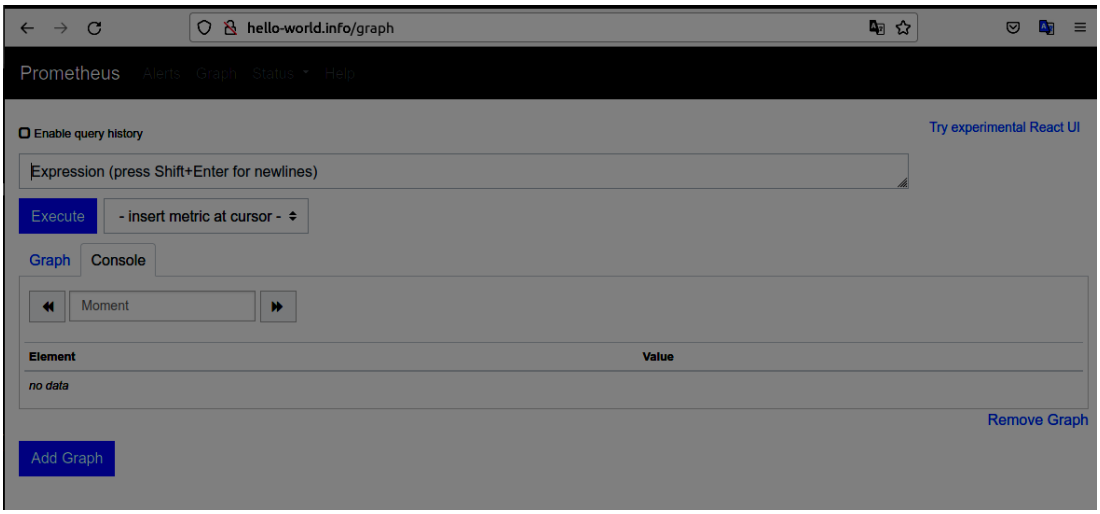
Создайте service и ingress для этого стейтфулсета, так чтобы запросы с любым доменом на белый IP вашего сервиса nginx-ingress-controller (тот что в нэймспэйсе ingress-nginx с типом LoadBalancer) шли на приложение

```
$ kubectl get services
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
database     ClusterIP   10.109.76.238   <none>        5432/TCP         21d
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP          32d
prometheus   ClusterIP   None            <none>        9090/TCP         20h
redmine      NodePort    10.98.50.123    <none>        3000:30016/TCP   21d
srv-redmine  NodePort    10.107.144.92   <none>        3000:30015/TCP   21d
$ kubectl describe services prometheus
Name:              prometheus
Namespace:         default
Labels:            app=prometheus
Annotations:       <none>
Selector:          app=prometheus
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                None
IPs:               None
Port:              prometheus  9090/TCP
TargetPort:        9090/TCP
Endpoints:         172.17.0.6:9090
Session Affinity:  None
Events:            <none>
$
```

```
$ kubectl get ingress
NAME                 CLASS   HOSTS             ADDRESS          PORTS   AGE
prometheus-ingress   nginx   hello-world.info  192.168.59.100   80      15h
$
$ kubectl describe ingress prometheus-ingress
Name:             prometheus-ingress
Labels:           <none>
Namespace:        default
Address:          192.168.59.100
Default backend:  default-http-backend:80 (<error: endpoints "default-http-backend" not found>)
Rules:
  Host              Path  Backends
  ----              ----  --------
  hello-world.info
                    /   prometheus:9090 (172.17.0.6:9090)
Annotations:        nginx.ingress.kubernetes.io/rewrite-target: /
Events:
  Type    Reason  Age                  From                      Message
  ----    ------  ----                 ----                      -------
  Normal  Sync    15h (x2 over 15h)    nginx-ingress-controller  Scheduled for sync
  Normal  Sync    3m9s (x3 over 4m10s) nginx-ingress-controller  Scheduled for sync
$
```

5.

Проверьте что при обращении из браузера на белый IP вы видите открывшееся приложение Prometheus



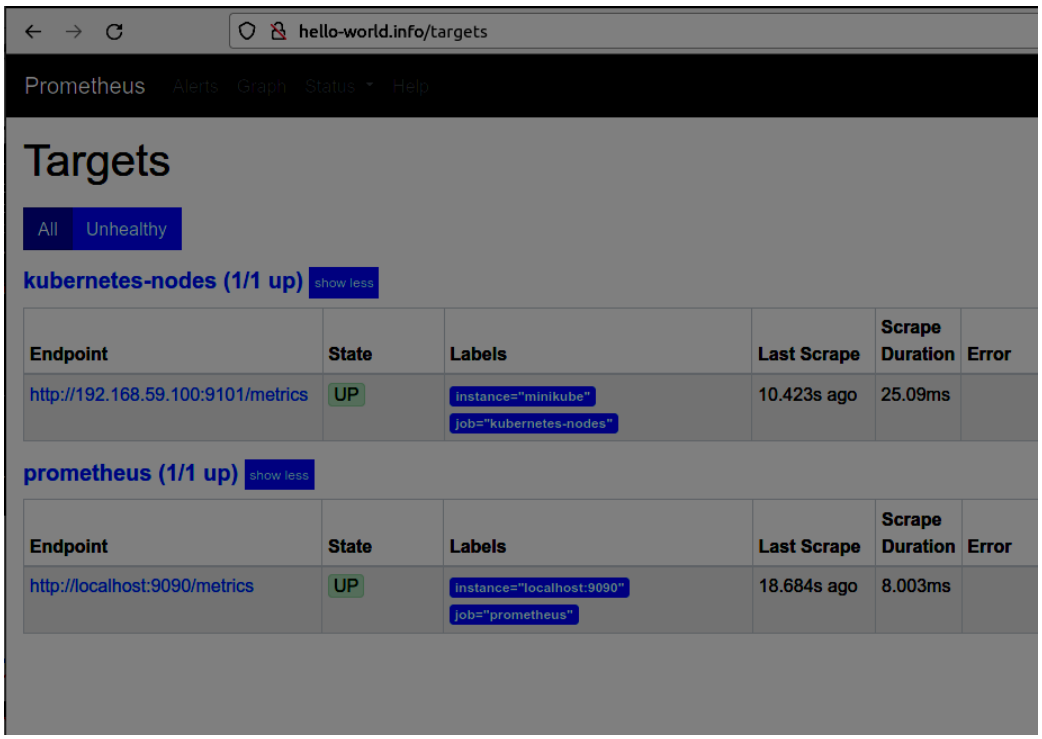
6.

В этом же неймспэйсе создайте DaemonSet node-exporter как в примере к лекции: practice/7.advanced-abstractions/daemonset.yaml

```
dmitrygubuntusrv:~/GeekBrains/4 Semestr/practice/3_containerization/lesson_07$ kubectl apply -f daemonset.yaml
Warning: spec.template.spec.nodeSelector[beta.kubernetes.io/os]: deprecated since v1.14; use "kubernetes.io/os" instead
daemonset.apps/node-exporter created
dmitrygubuntusrv:~/GeekBrains/4 Semestr/practice/3_containerization/lesson_07$
dmitrygubuntusrv:~/GeekBrains/4 Semestr/practice/3_containerization/lesson_07$ kubectl get daemonset
NAME            DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR               AGE
node-exporter   1         1         1       1            1           beta.kubernetes.io/os=linux   50m
dmitrygubuntusrv:~/GeekBrains/4 Semestr/practice/3_containerization/lesson_07$
dmitrygubuntusrv:~/GeekBrains/4 Semestr/practice/3_containerization/lesson_07$
dmitrygubuntusrv:~/GeekBrains/4 Semestr/practice/3_containerization/lesson_07$ kubectl describe daemonset node-exporter
Name:           node-exporter
Selector:       app=node-exporter
Node-Selector:  beta.kubernetes.io/os=linux
Labels:         app=node-exporter
Annotations:    deprecated.daemonset.template.generation: 1
Desired Number of Nodes Scheduled: 1
Current Number of Nodes Scheduled: 1
Number of Nodes Scheduled with Up-to-date Pods: 1
Number of Nodes Scheduled with Available Pods: 1
Number of Nodes Misscheduled: 0
Pods Status:  1 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=node-exporter
  Containers:
   node-exporter:
    Image:      quay.io/prometheus/node-exporter:v0.16.0
    Port:       <none>
    Host Port:  <none>
    Args:
      --web.listen-address=0.0.0.0:9101
      --path.procfs=/host/proc
      --path.sysfs=/host/sys
```

7.

Откройте в браузере интерфейс Prometheus. Попробуйте открыть Status -> Targets Тут вы должны увидеть все ноды своего кластера, которые Prometheus смог определить и собирает с ним метрики.



8.

Так же можете попробовать на вкладке Graph выполнить запрос node_load1 - это минутный Load Average для каждой из нод в кластере.