Дипломный практикум в Yandex.Cloud

(https://github.com/komarovma/diplnet)

Часть первая

Подготовка облачной инфраструктуры на базе облачного провайдера Яндекс.Облако

Сначала создадим облачный S3 bucket в созданном ЯО аккаунте, а также Yandex Container Registry. В первом блоке terraform будем хранить backend локально. После создания объектов в файле terraform.tfstate возьмём ключи доступа

```
// "version": 4,

// "terraform_version": "1.2.2",
// "serial": 23,
// "lineage": "ZZZZZZZZZZZZZZZZZZZZZZZZZ",
// "outputs": {
//   "access_key": {
//     "value": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
//     "type": "string",
//     "sensitive": true
//   },
//   "secret_key": {
//     "value": "YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY",
//     "type": "string",
//     "sensitive": true
//   }
```

https://github.com/komarovma/diplnet/tree/main/tf_s3_backend

Далее создадим основной модули terraform в отдельном каталоге. Пропишем облачный backend в фале main.tf

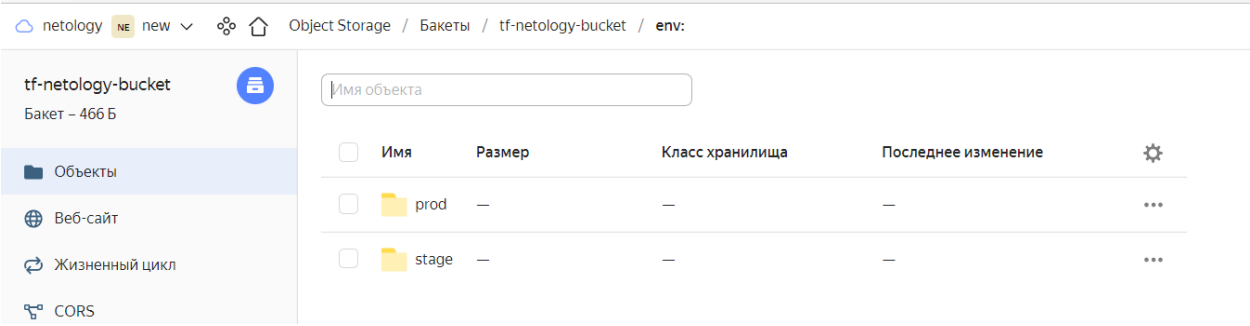
```
  backend "s3" {
    endpoint   = "storage.yandexcloud.net"
    bucket     = "tf-netology-bucket"
    region     = "ru-central1-a"
    key        = "diplom/terraform.tfstate"
    access_key = "XXXXXXXXXXXXXXXXX"
    secret_key = "YYYYYYYYYYYYYYYYYYYYYYYYYYY"
    skip_region_validation      = true
    skip_credentials_validation = true
```

Потом создадим workspace в terraform и зададим имя машины в зависимости от workspace

```
mike@HOMEDX79SR:~/diplom$ terraform.exe workspace list
  default
  prod
* stage
```

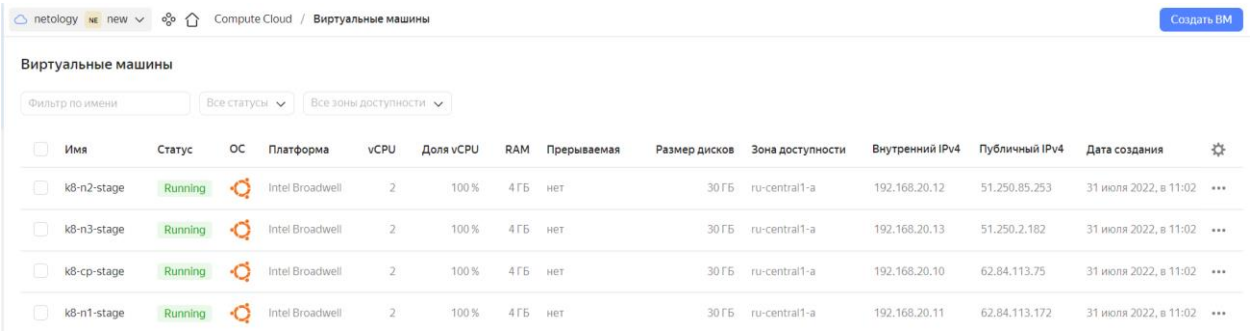
```
resource "yandex_compute_instance" "netology-vm-n2" {
  name        = "k8-n2-${terraform.workspace}"
  platform_id = "standard-v1"
  zone        = var.zone_id_a
```

После terraform init создался backend в облаке



Результат работы terraform.exe apply -lock=false в https://github.com/komarovma/diplnet/tree/main/tf_k8

```
yandex_vpc_network.netology-network-tf: Creating...
yandex_vpc_network.netology-network-tf: Creation complete after 1s [id=enptjq938j6ajki1mlcc]
yandex_vpc_subnet.private_c: Creating...
yandex_vpc_subnet.private_a: Creating...
yandex_vpc_subnet.private_b: Creating...
yandex_vpc_subnet.private_a: Creation complete after 1s [id=e9bdv00nj6dt1n7t6u0j]
yandex_compute_instance.netology-vm-cp: Creating...
yandex_compute_instance.netology-vm-n1: Creating...
yandex_compute_instance.netology-vm-n2: Creating...
yandex_compute_instance.netology-vm-n3: Creating...
yandex_vpc_subnet.private_b: Creation complete after 2s [id=e2lhgnqleh103ubiqlsm]
yandex_vpc_subnet.private_c: Creation complete after 3s [id=b0c8o9s6upcts0hscu4t]
yandex_compute_instance.netology-vm-cp: Still creating... [10s elapsed]
yandex_compute_instance.netology-vm-n1: Still creating... [10s elapsed]
yandex_compute_instance.netology-vm-n2: Still creating... [10s elapsed]
yandex_compute_instance.netology-vm-n3: Still creating... [10s elapsed]
yandex_compute_instance.netology-vm-cp: Still creating... [20s elapsed]
yandex_compute_instance.netology-vm-n1: Still creating... [20s elapsed]
yandex_compute_instance.netology-vm-n2: Still creating... [20s elapsed]
yandex_compute_instance.netology-vm-n3: Still creating... [20s elapsed]
yandex_compute_instance.netology-vm-n1: Creation complete after 24s [id=fhmovqovi1b506fea0ko]
yandex_compute_instance.netology-vm-cp: Creation complete after 25s [id=fhmh09pmejgonl80tn0v]
yandex_compute_instance.netology-vm-n3: Creation complete after 27s [id=fhmegpc6a851jgvo2rrg]
yandex_compute_instance.netology-vm-n2: Creation complete after 28s [id=fhma25tfpc19869kt0mb]

Apply complete! Resources: 8 added, 0 changed, 0 destroyed.

Outputs:

netology-vm-cp_external_ip = "62.84.113.75"
netology-vm-cp_internal_ip = "192.168.20.10"
netology-vm-n1_external_ip = "62.84.113.172"
netology-vm-n1_internal_ip = "192.168.20.11"
netology-vm-n2_external_ip = "51.250.85.253"
netology-vm-n2_internal_ip = "192.168.20.12"
netology-vm-n3_external_ip = "51.250.2.182"
netology-vm-n3_internal_ip = "192.168.20.13"
```



Виртуальная инфраструктура создана.

Часть вторая

Создание Kubernetes кластера

Клонировал репозиторий https://github.com/kubernetes-sigs/kubespray

Скопировал пример в свою директорию cp -rfp inventory/sample inventory/mycluster. Зашел в созданные виртуальные машины по SSH

Запустил билдер и подготовил inventory/mycluster/hosts.yaml

declare -a IPS=(62.84.113.75 62.84.113.172 51.250.85.253 51.250.2.182)

CONFIG_FILE=inventory/mycluster/hosts.yaml python3 contrib/inventory_builder/inventory.py ${IPS[@]}

Изменим hosts.yaml

```
all:
  hosts:
    cp1:
      ansible_host: 62.84.113.75
      ansible_user: ubuntu
    node1:
      ansible_host: 62.84.113.172
      ansible_user: ubuntu
    node2:
      ansible_host: 51.250.85.253
      ansible_user: ubuntu
    node3:
      ansible_host: 51.250.2.182
      ansible_user: ubuntu
  children:
    kube_control_plane:
```

А также all.yml

```
## External LB example config
## apiserver_loadbalancer_domain_name: "elb.some.domain"
loadbalancer_apiserver:
  address: 62.84.113.75
  port: 6443
```

Запускаем **ansible-playbook -i inventory/mycluster/hosts.yaml --become --become-user=root cluster.yml** в директории с клоном kuberspray

```
Sunday 31 July 2022  11:31:21 +0300 (0:00:00.095)       0:13:28.579 ***********
===============================================================================
kubernetes/preinstall : Install packages requirements ------------------------------------------------------------ 33.15s
kubernetes/kubeadm : Join to cluster ----------------------------------------------------------------------------- 31.10s
kubernetes/control-plane : kubeadm | Initialize first master ---------------------------------------------------- 27.59s
download : download_file | Validate mirrors ---------------------------------------------------------------------- 24.68s
kubernetes/preinstall : Preinstall | wait for the apiserver to be running ---------------------------------------- 14.23s
download : download_container | Download image if required ------------------------------------------------------- 13.52s
kubernetes-apps/ansible : Kubernetes Apps | Start Resources ------------------------------------------------------ 12.49s
network_plugin/calico : Wait for calico kubeconfig to be created ------------------------------------------------- 11.87s
kubernetes/preinstall : Update package management cache (APT) ---------------------------------------------------- 10.64s
kubernetes-apps/ansible : Kubernetes Apps | Lay Down CoreDNS templates ------------------------------------------- 10.37s
download : download_container | Download image if required ------------------------------------------------------- 9.84s
download : download_container | Download image if required ------------------------------------------------------- 9.40s
kubernetes/control-plane : Master | wait for kube-scheduler ------------------------------------------------------ 8.10s
download : download_container | Download image if required ------------------------------------------------------- 7.85s
network_plugin/calico : Start Calico resources ------------------------------------------------------------------- 6.69s
download : download_container | Download image if required ------------------------------------------------------- 6.45s
network_plugin/calico : Calico | Create calico manifests --------------------------------------------------------- 5.96s
container-engine/containerd : containerd | Unpack containerd archive -------------------------------------------- 5.81s
etcd : reload etcd ----------------------------------------------------------------------------------------------- 5.75s
download : download_container | Download image if required ------------------------------------------------------- 5.65s
```

После окончания запускаем копирование файла конфигурации
ssh ubuntu@62.84.113.75"sudo cat /etc/kubernetes/admin.conf " > /home/mike/.kube/config
Меняем имя сервера на IP адрес в /home/mike/.kube/config

```
    server: https://62.84.113.75:6443
  name: cluster.local
```

И проверяем работы кластера

```
mike@HOMEDX79SR:~/diplom/k8$ kubectl get pods --all-namespaces
NAMESPACE     NAME                                       READY   STATUS    RESTARTS      AGE
kube-system   calico-kube-controllers-58dfb4874f-smjjj   1/1     Running   0             15m
kube-system   calico-node-kdr7c                          1/1     Running   0             16m
kube-system   calico-node-mrlq6                          1/1     Running   0             16m
kube-system   calico-node-r96vx                          1/1     Running   0             16m
kube-system   calico-node-xv4xf                          1/1     Running   0             16m
kube-system   coredns-76b4fb4578-4j6zz                   1/1     Running   0             15m
kube-system   coredns-76b4fb4578-8mhsq                   1/1     Running   0             15m
kube-system   dns-autoscaler-7979fb6659-89s6l            1/1     Running   0             15m
kube-system   kube-apiserver-cp1                         1/1     Running   1             18m
kube-system   kube-controller-manager-cp1                1/1     Running   2 (14m ago)   18m
kube-system   kube-proxy-744xh                           1/1     Running   0             16m
kube-system   kube-proxy-rsjgf                           1/1     Running   0             16m
kube-system   kube-proxy-shw6h                           1/1     Running   0             17m
kube-system   kube-proxy-v26xj                           1/1     Running   0             16m
kube-system   kube-scheduler-cp1                         1/1     Running   2 (14m ago)   18m
kube-system   nodelocaldns-8hxwm                         1/1     Running   0             15m
kube-system   nodelocaldns-p5t7f                         1/1     Running   0             15m
kube-system   nodelocaldns-qxf9c                         1/1     Running   0             15m
kube-system   nodelocaldns-vzt2c                         1/1     Running   0             15m
mike@HOMEDX79SR:~/diplom/k8$ kubectl get nodes
NAME    STATUS   ROLES                  AGE   VERSION
cp1     Ready    control-plane,master   18m   v1.23.6
node1   Ready    <none>                 17m   v1.23.6
node2   Ready    <none>                 17m   v1.23.6
node3   Ready    <none>                 17m   v1.23.6
mike@HOMEDX79SR:~/diplom/k8$
```

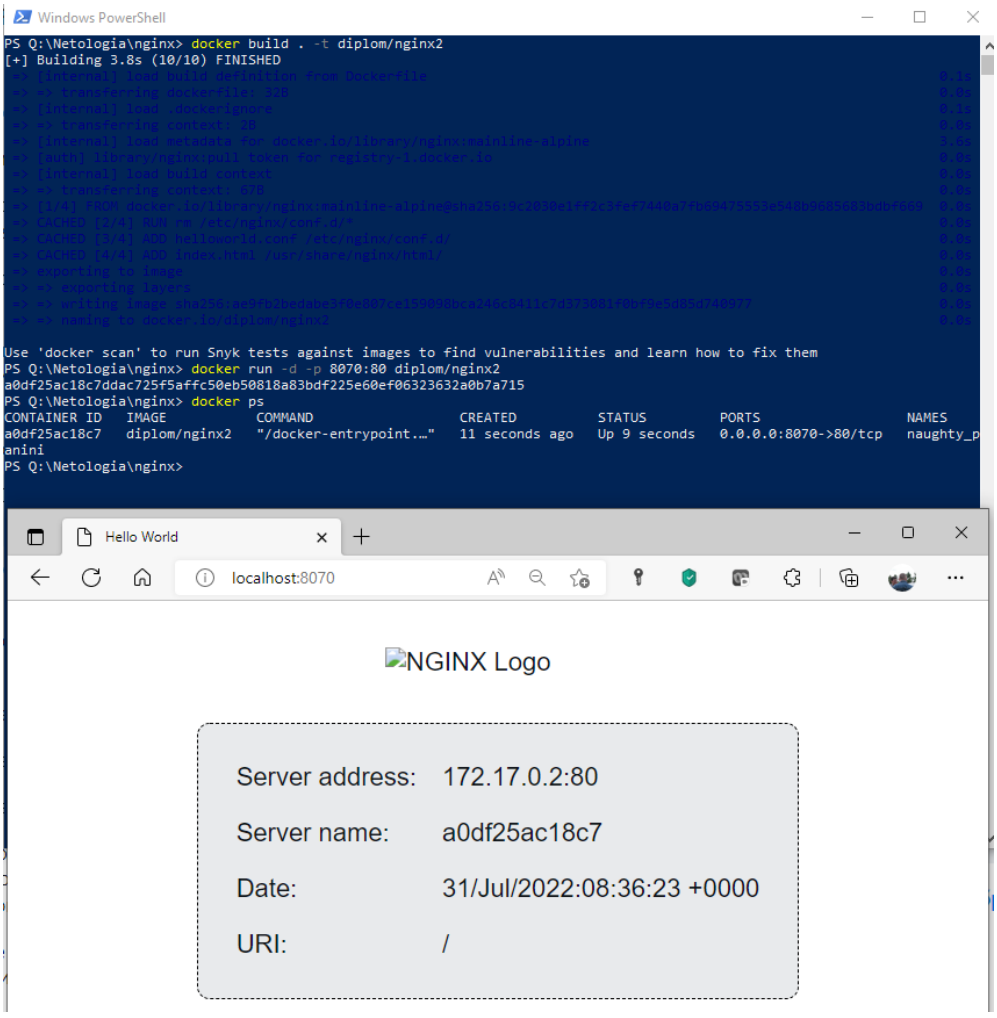
Часть третья

Создание тестового приложения.

Тестовое в приложение находиться в отдельно репозитории https://github.com/komarovma/nginx . Основа приложения Dockerfile

```
FROM nginx:mainline-alpine
RUN rm /etc/nginx/conf.d/*
ADD helloworld.conf /etc/nginx/conf.d/
ADD index.html /usr/share/nginx/html/
```

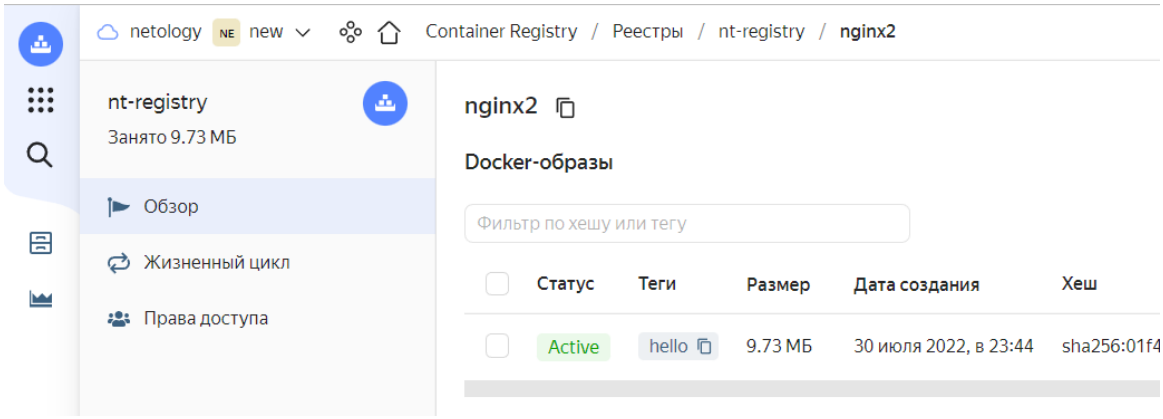
А также 2 файла конфигурации и веб страницы. Клонируем репозиторий собираем образ. Ставим метку и пушим образ в Yandex Container Registry/



docker tag diplom/nginx2 cr.yandex/crpXXXXXXXXXXX/nginx2:hello

docker push cr.yandex/crpXXXXXXXXXXX/nginx2:hello

В результате получаем



Часть четвертая

Подготовка системы мониторинга.

Сначала включим ingress в kubespray по следующему пути

/inventory/mycluster/group_vars/k8s_cluster/addons.yml

```
# Nginx ingress controller deployment
ingress_nginx_enabled: true
```

И заново применим конфиг

ansible-playbook -i inventory/mycluster/hosts.yaml --become --become-user=root cluster.yml

Проверим результат

kubectl get ns

NAME STATUS AGE

default Active 116m

ingress-nginx Active 50m

kube-node-lease Active 116m

kube-public Active 116m

kube-system Active 117m

Появилась новая namespace

Переходим на контроль ноду.

На контроль ноде ставим Helm и добавляем репозитории

curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3

chmod 700 get_helm.sh

./get_helm.sh

helm repo add stable https://charts.helm.sh/stable

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

root@cp1:~# **helm search repo prometheus-community**

NAME CHART VERSION APP VERSION DESCRIPTION

prometheus-community/alertmanager 0.19.0 v0.23.0 The Alertmanager handles alerts sent by client ...

prometheus-community/kube-prometheus-stack 39.12.0 0.58.0 kube-prometheus-stack collects Kubernetes manif...

Создаем namespace для мониторинга

root@cp1:~# kubectl create namespace monitoring

namespace/prometheus created

Устанавливаем туда kube-prometheus-stack

root@cp1:~# **helm install prometheus-community/kube-prometheus-stack --generate-name --namespace monitoring**

NAME: kube-prometheus-stack-1662903542

LAST DEPLOYED: Sun Sep 11 13:39:06 2022

NAMESPACE: prometheus

STATUS: deployed

REVISION: 1

NOTES:

kube-prometheus-stack has been installed. Check its status by running:

kubectl --namespace prometheus get pods -l "release=kube-prometheus-stack-1662903542"

Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanager and Prometheus instances using the Operator.

В итоге

root@cp1:/home/ubuntu# **kubectl --namespace monitoring get svc**

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S)

AGE

alertmanager-operated ClusterIP None <none>
9093/TCP,9094/TCP,9094/UDP 120m

grafana ClusterIP 10.233.52.126 <none> 80/TCP

51m

```
kube-prometheus-stack-1663-alertmanager              ClusterIP   10.233.58.121   <none>      9093/TCP
      120m
kube-prometheus-stack-1663-operator                  ClusterIP   10.233.41.216   <none>      443/TCP
      120m
kube-prometheus-stack-1663-prometheus                ClusterIP   10.233.39.76    <none>      9090/TCP
      120m
kube-prometheus-stack-1663488968-grafana             ClusterIP   10.233.17.54    <none>      80/TCP
      120m
kube-prometheus-stack-1663488968-kube-state-metrics  ClusterIP   10.233.46.139   <none>
8080/TCP
      120m
kube-prometheus-stack-1663488968-prometheus-node-exporter   ClusterIP   10.233.20.238   <none>
9100/TCP
      120m
prometheus-operated                                  ClusterIP   None            <none>      9090/TCP
```

Далее создаем конфигурационный файл для Ingress, например mon.yaml

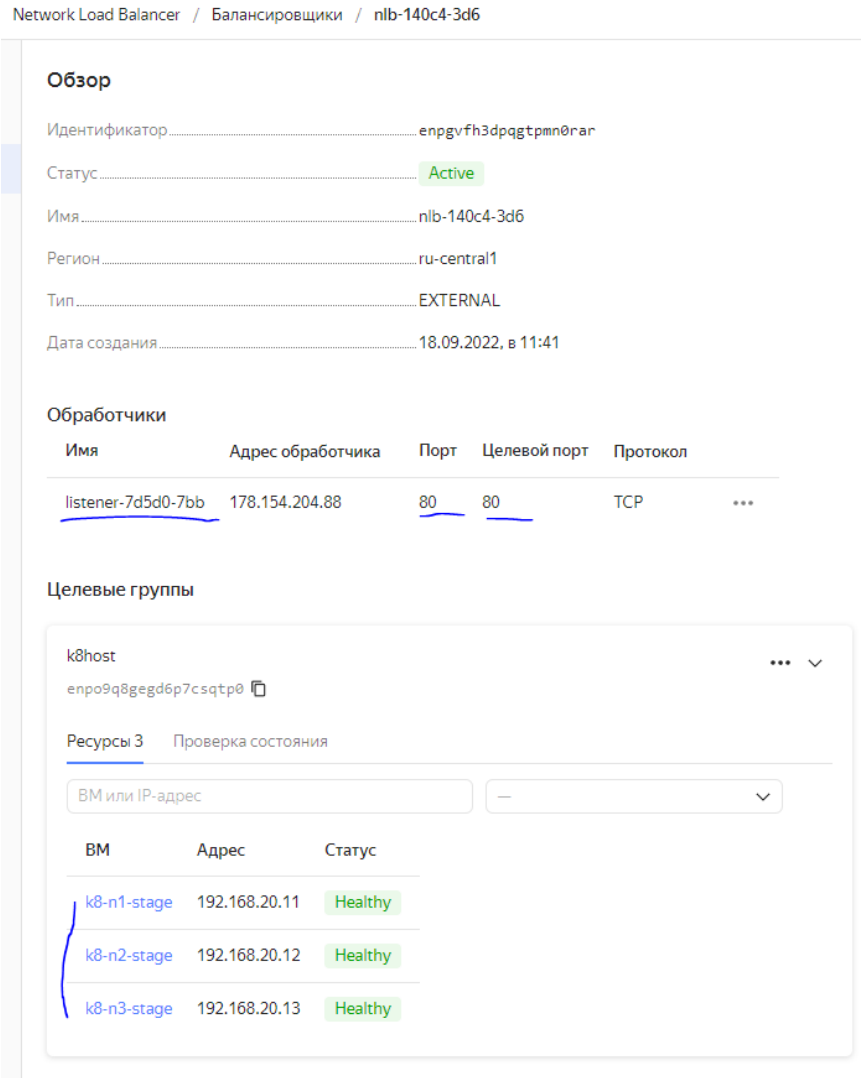
```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: grafana
  namespace: monitoring
spec:
  rules:
    - host: gr.akop.pw
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: grafana
                port:
                  name: web

---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: prometheus
  namespace: monitoring
spec:
```

```yaml
rules:
  - host: pr.akop.pw
    http:
      paths:
        - path: /
          pathType: Prefix
          backend:
            service:
              name: kube-prometheus-stack-1663-prometheus
              port:
                name: http-web
```

и применяем его

kubectl apply -f .

Идем в Yandex Cloud и создаем балансировщик который указывает на наши ноды (созданные ранее виртуальные машины)



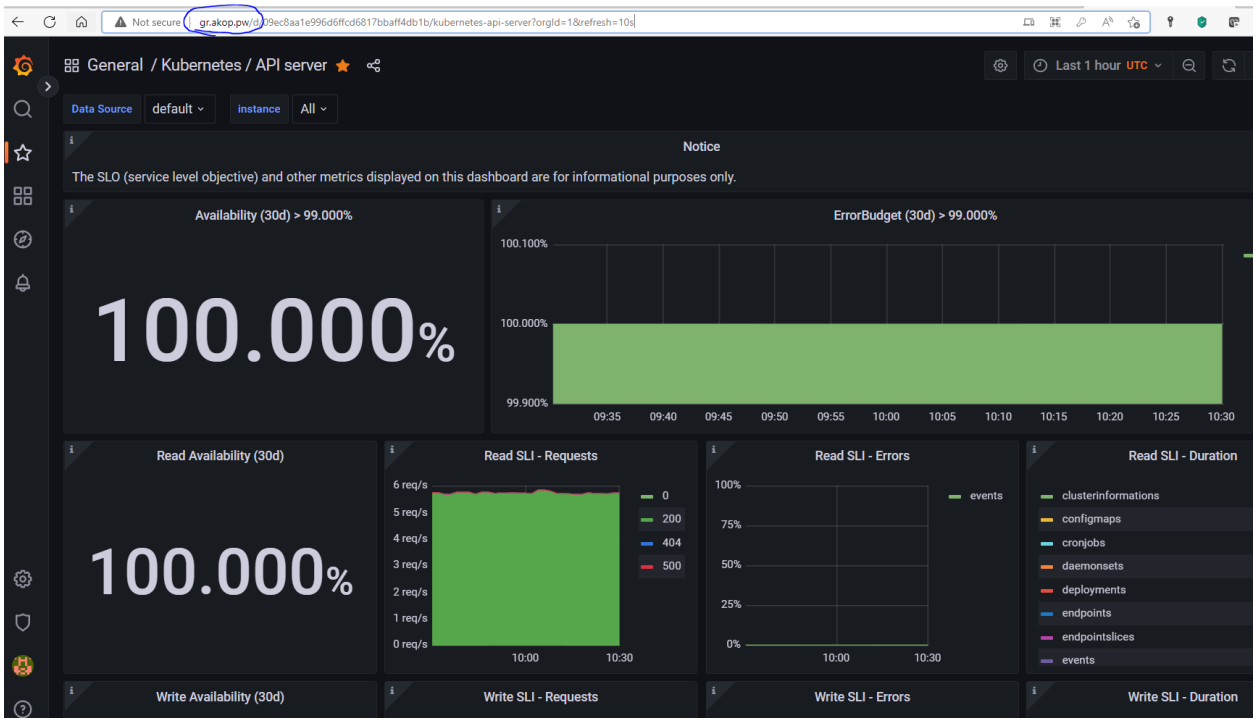
Идем файл host на компьютере, с которого будем ходить в мониторинг и добавляем записи

178.154.204.88 gr.akop.pw

178.154.204.88 pr.akop.pw

Где 178.154.204.88 адрес балансировщика

Заходим в Grafana



Заходим в Prometheus