

## Homework\_Lesson40\_Report

Задание:

1. Выберите один из способов развёртывания кластера (GCP, AWS, Azure, Hardway+VM on any cloud) и выполните его..
2. Изучите основные конфигурационные файлы и настройки, необходимые для настройки кластера Kubernetes.
3. Спланируйте мероприятия по обеспечению безопасности и масштабируемости кластера Kubernetes и запишите свой план в виде небольшого отчёта.
4. Создайте несколько (1-10) контейнеризированных приложений, которые вы хотите развернуть внутри кластера Kubernetes.
5. Разверните приложения внутри вашего кластера Kubernetes, используя манифесты Kubernetes, которые описывают ресурсы, такие как Pods, Services и Deployments.
6. Разработайте манифесты YAML для создания нескольких Namespaces, Pods и подходящего типа Controller для вашего приложения.
7. Namespace можете определять самостоятельно, например: rand, finance, blue, gree, dev, prod, qa...
8. Разработайте манифесты YAML для создания каждого типа Controller и запустите их на кластере Kubernetes.
9. Запустите приложение и убедитесь, что все ресурсы правильно созданы и функционируют.
10. Все созданные манифесты должны быть загружены в ваш репозиторий для проверки"

Выполнение:

1. Выберите один из способов развёртывания кластера.

Кластер будем использовать на GSP

Поднимать будем через terraform.

## 2. Изучите основные конфигурационные файлы и настройки, необходимые для настройки кластера Kubernetes.

При настройке Kubernetes-кластера важно ознакомиться со следующими файлами и настройками:

**kubeconfig** (~/.kube/config) – файл конфигурации для kubectl.

Физически находится в домашней директории пользователя (~/.kube/).

Используется для хранения информации о контексте кластеров и аутентификационных данных.

```
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUVMRENDQXBTZ0F3SUJBZ0.....
  server: https://34.88.31.69
  name: gke_glossy-precinct-450612-g6_europe-north1-a_my-gke-cluster
contexts:
- context:
  cluster: gke_glossy-precinct-450612-g6_europe-north1-a_my-gke-cluster
  user: gke_glossy-precinct-450612-g6_europe-north1-a_my-gke-cluster
  name: gke_glossy-precinct-450612-g6_europe-north1-a_my-gke-cluster
  current-context: gke_glossy-precinct-450612-g6_europe-north1-a_my-gke-cluster
kind: Config
preferences: { }
users:
- name: gke_glossy-precinct-450612-g6_europe-north1-a_my-gke-cluster
  user:
  exec:
    apiVersion: client.authentication.k8s.io/v1beta1
    command: gke-gcloud-auth-plugin.exe
    installHint: Install gke-gcloud-auth-plugin for use with kubectl by following
      https://cloud.google.com/kubernetes-engine/docs/how-to/cluster-access-for-kubectl#install_plugin
    provideClusterInfo: true
```

Манифесты Kubernetes (YAML-файлы):

**Pod:** минимальная единица в Kubernetes.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
  - name: nginx
    image: nginx:latest
```

**Deployment YAML-файлы** – используются для описания развертывания приложений.

Хранятся в проекте и могут находиться в любой директории (например, `~/kubernetes/deployments/`).

Запускаются командой:

```
kubectl apply -f deployment.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
        - name: my-container
          image: my-image:latest
          ports:
            - containerPort: 80
```

**Service YAML-файлы** – управляют сетевыми настройками приложений.

Описывают, как сервисы взаимодействуют друг с другом.

Пример местоположения: `~/kubernetes/services/`.

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

**ConfigMap и Secret** – управляют конфиденциальными данными и настройками.

ConfigMap хранится в etcd, но создается через YAML-файлы или команду:

```
kubectl create configmap my-config --from-file=config.properties
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-config
data:
  config.json: |
    {
      "key": "value"
    }
```

**RBAC (Role-Based Access Control)** – управление доступом к ресурсам.

Определяется через ClusterRole, Role, RoleBinding и ClusterRoleBinding.

Физически файлы могут храниться в ~/kubernetes/rbac/.

Определяется через ClusterRole, Role, RoleBinding и ClusterRoleBinding.

Физически файлы могут храниться в ~/kubernetes/rbac/.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

**Network Policies** – контроль сетевого взаимодействия между подами.

Хранятся в виде YAML-файлов и применяются через `kubectl apply`.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-app-traffic
spec:
  podSelector:
    matchLabels:
      app: my-app
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: another-app
    ports:
    - protocol: TCP
      port: 80
```

3. Спланируйте мероприятия по обеспечению безопасности и масштабируемости кластера Kubernetes и запишите свой план в виде небольшого отчёта.

Для обеспечения безопасности и масштабируемости сделаем слеует проделать следующие шаги:

- Настроим Network Policies для ограничения взаимодействия подов. Это поможет изолировать поды и минимизировать риски несанкционированного доступа. Например, можно настроить политики так, чтобы поды из одного namespace-а могли общаться только с определенными сервисами, а внешний трафик был строго ограничен.
- Используем RBAC, чтобы ограничить доступ пользователей и сервисов. Мы можем создать отдельные роли для админов, разработчиков и сервисов (ServiceAccounts). Также стоит привязать их к конкретным ресурсам (например, только чтение ConfigMaps или запуск подов) и namespace-ам, чтобы избежать избыточных прав.
- Настроим Horizontal Pod Autoscaler (HPA) для управления нагрузкой. Чтобы он работал эффективно, нужно настроить метрики, например, CPU. Так же нужно подумать о минимальном и максимальном количестве подов, чтобы избежать перерасхода ресурсов
- Включить Node Autoscaler, чтобы добавлять и удалять узлы по необходимости. Cluster Autoscaler будет автоматически подстраивать количество узлов под нагрузку.
- Настроим CI/CD с помощью Jenkins для быстрого обновления приложений.

Создадим namespace red и blue. И прикрепим к ним 2 наших приложения.

```
PS D:\kubernetes\kubernetes> kubectl get ns
NAME                STATUS  AGE
blue                Active  85m
default             Active  25h
gke-managed-cim     Active  25h
gke-managed-system  Active  25h
gke-managed-volumepopulator  Active  25h
gmp-public          Active  25h
gmp-system          Active  25h
kube-node-lease     Active  25h
kube-public         Active  25h
kube-system         Active  25h
red                 Active  85m
PS D:\kubernetes\kubernetes>
```

Наши контейнеры и сервис привязанные к namespace blue.

```
PS D:\kubernetes\kubernetes> kubectl get pods -n blue
NAME                                READY   STATUS    RESTARTS   AGE
my-aap-7c4fb47496-rtfn8            1/1     Running   0           70m
my-aap-7c4fb47496-z5zb2            1/1     Running   0           70m
PS D:\kubernetes\kubernetes> kubectl get svc -n blue
NAME      TYPE          CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
my-aap    LoadBalancer 34.118.235.208 34.88.22.45   5000:30980/TCP   70m
PS D:\kubernetes\kubernetes>
```

Наши контейнеры и сервис, привязанные к namespace red.

```
PS D:\kubernetes\kubernetes> kubectl get svc -n red
NAME             TYPE          CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
http-aap-service LoadBalancer 34.118.232.126 35.228.106.45 5050:31179/TCP   48m
PS D:\kubernetes\kubernetes> kubectl get pods -n red
NAME                                READY   STATUS    RESTARTS   AGE
http-aap-ff9dbf875-kqm69           1/1     Running   0           48m
http-aap-ff9dbf875-vcnv4           1/1     Running   0           48m
PS D:\kubernetes\kubernetes>
```

Проверим доступность наших приложений.

The screenshot shows two browser windows. The top window displays the 'Главная' (Main) page of a web application. It features a 'Добавить раздел' (Add section) form with a text input for 'Название раздела' (Section name) and a green 'Добавить' (Add) button. Below this is a 'Список разделов' (List of sections) table with two entries: 'Windows' and 'Linux'. Each entry has a 'Новое название' (New name) input field, a blue 'Редактировать' (Edit) button, and a red 'Удалить' (Delete) button. The bottom window displays the 'Information' page, which shows user details: 'Levchenko Alexey Viktorovich', 'Group: DOS24-onl', 'Topic: webserver', and 'IP Address: 192.168.1.210:8080'. At the bottom of the 'Information' page, there are three identical images of a red truck, followed by the text 'Created by Levchenko Alexey'.

Как видим приложения наши доступны и работают.