**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**УНИВЕРСИТЕТ ИТМО**

**Факультет информационных технологий и программирования**

**Теория систем и системный анализ**

**Лабораторная работа №3**

Оценка структурной сложности информационной системы

**Выполнили студенты группы № М3300**:  
Балашов С.А.

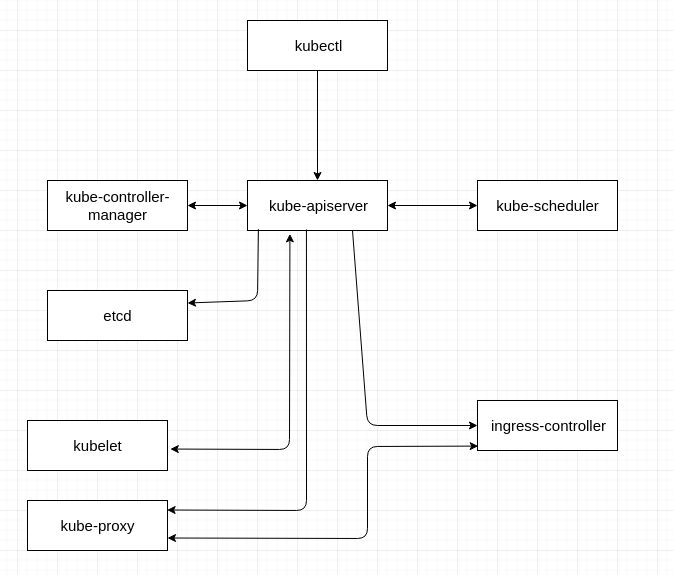
Гринин В.Н.

Сметанина Д.Д.

**САНКТ-ПЕТЕРБУРГ**

**2019**

**Выделенные модули в соответствии с заданием**

****

**Задание 1.** Для каждого модуля определить степень внутренней связности по предложенной классификации.

Сила связности для модулей

* kubectl - коммуникативная связность (CC=7);
* kube-controller-manager - временная связность (CC=3);
* kube-apiserver - информационная связность (CC=9);
* kube-scheduler - функциональная связность (CC=10);
* etcd - функциональная связность (CC=10);
* ingress-controller - коммуникативная связность (CC=7);
* kubelet - информационная связность (CC=9);
* kube-proxy - информационная связность (CC=9).

**Задание 2.** Для каждой связи определить тип сцепления по предложенной классификации.

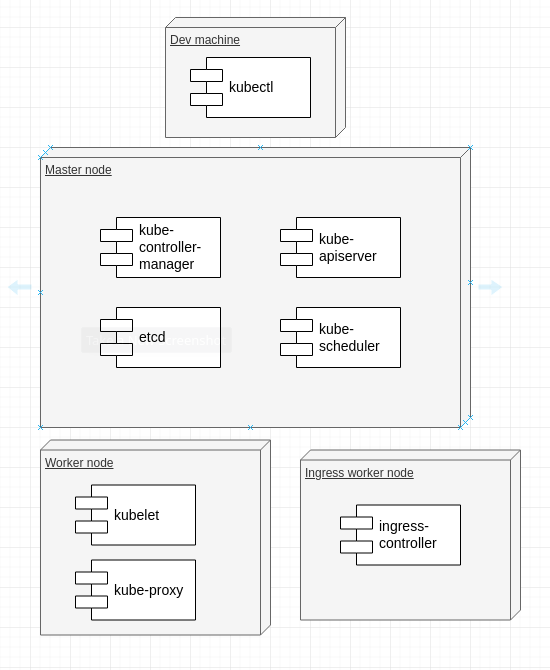
* kubectl - kube-apiserver - сцепление по образцу (СЦ=3);
* kube-controller-manager - kube-apiserver - сцепление по управлению (СЦ=4);
* kube-scheduler - kube-apiserver - сцепление по управлению (СЦ=4);
* etcd - kube-apiserver - сцепление по образцу (СЦ=3);
* kubelet - kube-apiserver - сцепление по управлению (СЦ=4);
* kube-proxy - kube-apiserver - сцепление по управлению (СЦ=4);
* ingress-controller - kube-apiserver - сцепление по образцу (СЦ=3);
* kube-proxy - ingress-controller - сцепление по образцу (СЦ=3);

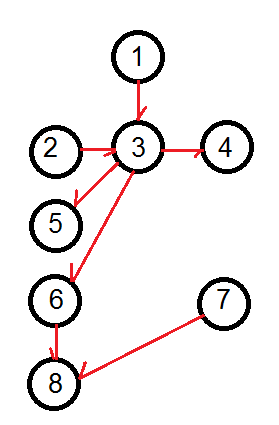
**Задание 3.**

**3.1** Для полученной структуры модулей проверить на содержательном уровне выполнение требований к модульности и информационной закрытости.

После построения структуры модулей, определения внутренних и внешних связей можно сделать вывод о том, что система обладает достаточно модульной архитектурой и информационной закрытостью, т.к. модули внутри сильно связаны, но, при этом, имеют достаточно простой внешний интерфейс. Единственным модулем, который обладает слабой внутренней связностью, является kube-controller-manager, что можно объяснить его объединением в себе других контроллеров разного типа, которые, для простоты были выделены в controller-manager. Внешняя связность модулей невысокая (максимум СЦ=4), что говорит о том, что они относительно независимы.

**3.2** Построить диаграмму размещения модулей по физическим узлам для тех систем, где это возможно, или по изолированно выполняющимся процессам для всех остальных. За основу взять в первом случае требования к платформе и технологии реализации, а во втором случае – оценку взаимовлияния сбоев и критического повышения потребляемых ресурсов, приводящих к взаимной блокировке.



**Задание 4.** Выполнить топологическую сортировку построенных модулей

(На основе остовного дерева из лабораторной работы №2)

1. kubectl

2. kube-controller-manager

3. kube-apiserver

4. kube-scheduler

5. etcd

6. kubelet

7. ingress-controller

8. kube-proxy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Шаг | Текущая | Белые | Серые | Черные |
| 0 | - | 1,2,3,4,5,6,7,8 | - | - |
| 1 | 1 | 2,3,4,5,6,7,8 | 1 | - |
| 2 | 3 | 2,4,5,6,7,8 | 1,3 | - |
| 3 | 4 | 2,5,6,7,8 | 1,3,4 | - |
| 4 | 3 | 2,5,6,7,8 | 1,3 | 4 |
| 5 | 5 | 2,6,7,8 | 1,3,5 | 4 |
| 6 | 3 | 2,6,7,8 | 1,3 | 4,5 |
| 7 | 6 | 2,7,8 | 1,3,6 | 4,5 |
| 8 | 8 | 2,7 | 1,3,6,8 | 4,5 |
| 9 | 6 | 2,7 | 1,3,6 | 4,5,8 |
| 10 | 3 | 2,7 | 1,3 | 6,4,5,8 |
| 11 | 1 | 2,7 | 1 | 3,6,4,5,8 |
| 12 | - | 2,7 | - | 1,3,6,4,5,8 |
| 13 | 2 | 7 | 2 | 1,3,6,4,5,8 |
| 14 | - | 7 | - | 2,1,3,6,4,5,8 |
| 15 | 7 | - | 7 | 2,1,3,6,4,5,8 |
| 16 | - | - | - | 7,2,1,3,6,4,5,8 |

Таким образом:

7. ingress-controller

2. kube-controller-manager

1. kubectl

3. kube-apiserver

6. kubelet

4. kube-scheduler

5. etcd

8. kube-proxy

Вопросы:

* В каком порядке должны быть разработаны модули, чтобы каждый из них после разработки сразу можно было бы протестировать на полную функциональность (то есть появление потом новых модулей не приводило бы к необходимости его перетестирования)

1. kube-apiserver
2. etcd
3. kube-scheduler
4. kube-controller-manager
5. kubectl
6. kubelet
7. kube-proxy
8. ingress-controller

* В каком порядке нужно принимать проектные решения по модулям (какие выбрать технологии, структуры данных, интерфейсы и т.п.), чтобы потом не менять эти решения в процессе проектирования остальных модулей.

Для проектирования самих модулей системы не нужно выбирать технологии или структуры данных. Для обеспечения работы необходим следующий порядок действий:

1. Определение системы, на которой будет работать кластер (например, ubuntu)
2. Определения списка машин в кластере (нод)
3. Подготовка нод, установка докера, bridge-utils и т.д.
4. Собственно установка kubernetes
5. Настройка системы через конфигурационные файлы.