МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра математического обеспечения вычислительных комплексов и
информационных систем

# АВТОМАТИЗАЦИЯ КОНТРОЛЯ И МОНИТОРИНГА СОСТОЯНИЯ КЛАСТЕРА KUBERNETES

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 273 группы
направления 02.04.03 Математическое обеспечение и администрирование
информационных систем
факультета компьютерных наук и информационных технологий
Радюшина Вадима Станиславовича

Научный руководитель:
д. ф.-м. н., профессор _____ Д. К. Андрейченко

подпись, дата

Зав. кафедрой:
д. ф.-м. н., профессор _____ Д. К. Андрейченко

подпись, дата

Саратов 2023

# ВВЕДЕНИЕ

**Актуальность темы.** IT-индустрия – одна из наиболее быстроразвивающихся отраслей 21-го века. Начиная примерно с 2015-го года одним из наиболее быстрорастущих направлений являются облачные вычисления. Они позволяют компаниям перенести часть своей инфраструктуры в «облако» и сократить расходы.

Существует множество зарубежных облачных провайдеров от крупных компаний, такие как Microsoft Azure, Amazon Web Services (AWS), Google Cloud Platform (GCP), и решения от отечественных IT-гигантов: Yandex Cloud, MTS Cloud, Cloud.ru (ранее SberCloud) от Сбербанка и другие. Также, с появлением платформы OpenStack, компании могут создавать свою собственную частную облачную инфраструктуру.

В то же время вместе с облачными технологиями растет и популярность платформы управления контейнерами Kubernetes и его дистрибутивов, например OpenShift от Red Hat или Rancher. Облачные провайдеры также предлагают Kubernetes как услугу (KaaS или managed Kubernetes), используя свои собственные варианты (например Elastic Kubernetes Service (EKS) в AWS, Azure Kubernetes Service (AKS), Google Kubernetes Engine (GKE)). Все больше компаний используют Kubernetes в том или ином виде для того, чтобы уменьшить время простоя приложений и упростить процесс их развертывания. При этом, у одной компании может быть более 50 кластеров, и некоторые из них иметь очень большой размер. Так, например, компания OpenAI имеет кластер с 7500 узлами для обучения своих моделей.

Kubernetes представляет собой сложную систему с множеством компонентов, каждый из которых в любой момент времени может выйти из строя. Например, из-за проблем с гипервизором может выключиться один из узлов кластера, что повлечет за собой простой приложений внутри кластера. Рабочие нагрузки, критичные для компании, могут не запуститься после неудачного обновления. Или, что еще хуже, могут выйти из строя системные компоненты, такие как API сервер или внутреннее хранилище etcd, что в

свою очередь повлечет за собой деградацию кластера. Таким образом, актуальной становится проблема контроля и мониторинга его состояния.

Проверка состояния может проводиться инженерами с помощью инструментов командной строки, однако подобное решение не масштабируется и отнимает главный человеческий ресурс – время. В связи с этим возникает потребность в автоматизации контроля и мониторинга состояния кластеров Kubernetes.

Хотя существуют мощные системы мониторинга и оповещения, такие как Prometheus, Zabbix или VictoriaMetrics, а также приложения для визуализации метрик полученных из таких систем, к примеру Grafana, подобные инструменты довольно объемны и не имеют удобного API, а значит данные из них сложно интегрируются, например, с порталом самообслуживания. К тому же, люди, незнакомые с такими системами, могут не понять, что именно происходит с кластером в данный момент.

**Цель магистерской работы** – разработка приложения для автоматизация контроля и мониторинга состояния кластера Kubernetes.

Поставленная цель определила **следующие задачи**:

1. выявить основные компоненты Kubernetes, наиболее критичные для его работы;
2. выбрать инструментарий для реализации приложения;
3. спроектировать архитектуру приложения и выявить места, в которых можно применить параллелизацию.

**Методологические основы** облачных технологий и вычислений, а также системы Kubernetes представлены в работах Фостера, Денниса, Лушка, Сайфана, Биллджина и Роланда.

**Практическая значимость магистерской работы.** Разработано приложение для автоматизация контроля и мониторинга состояния кластера Kubernetes.

**Структура и объём работы**. Магистерская работа состоит из введения, 4 разделов, заключения, списка использованных источников и 2 приложений.

Общий объем работы – 76 страниц, из них 59 страниц – основное содержание, включая 10 рисунков и 1 таблица, список использованных источников информации – 24 наименования.

## КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

**Первый раздел «Облачные технологии»** посвящен основным понятиям и концепциям облачных технологий.

Облачные технологии – модель обеспечения доступа к компьютерным ресурсам, например к вычислительным мощностям и хранилищам данных. Суть этой модели состоит в том, что с ее помощью можно предоставить удаленный доступ к серверам, базам данных, сетям и сервисам.

Облачные вычисления делятся на три типа: частные, публичные и гибридные. Каждый из них имеет разный уровень безопасности и управления владельцем.

Также выделяются три основных типа облачных сервисов: программное обеспечение как услуга (SaaS), платформа как услуга (PaaS) и инфраструктура как услуга (IaaS).

ПО как услуга (SaaS) – модель предоставления доступа к программному обеспечению, в которой приложения размещаются на серверах поставщика облачных решений. Приложения работают с помощью сети Интернет. Пользователь ПО не тратит средства на создание и поддержание собственной вычислительной инфраструктуры. Вместо этого он платит подписку за услугу, стоимость которой пропорциональна объемам использования ресурсов.

Платформа как услуга (PaaS) – модель, в которой предоставляется доступ к инструментам, необходимым для создания и контроля приложений. У пользователей данной модели нет необходимости в трате ресурсов для создания и поддержки собственной инфраструктуры. Они получают доступ к инфраструктуре и промежуточному программному обеспечению поставщика облачных решений через Интернет (например с помощью веб-браузера).

Инфраструктура как услуга (IaaS) – модель, в которой обеспечивается доступ к инфраструктурным компонентам с помощью сети Интернет по требованию. Ее главное преимущество заключается в том, что они размещаются у поставщика облачных решений. Эти компоненты обеспечивают вычисления, а также хранение данных и пропускную способность сети. Все это для того, чтобы пользователи имели возможность запускать свои приложения в облаке. В данной модели за установку, последующую настройку и обеспечение безопасности, а также обслуживание ПО как правило отвечает заказчик. Компоненты инфраструктуры, то есть ее вычислительные ресурсы, размещаются в любом типе облака (частном, публичном или гибридном).

**Второй раздел «Платформа управления контейнерами Kubernetes»** посвящен системе Kubernetes.

Kubernetes – открытое программное обеспечение для оркестрации контейнеризированных приложений. Включает в себя возможность автоматизации развертывания, масштабирования и координации. Название происходит от греческого языка, на котором оно означает «рулевой». Изначально был разработан компанией Google для своих внутренних нужд. Затем, в 2014 году был опубликован исходный код проекта, а в 2015 система была передана под управление фонду Cloud Native Computing Foundation.

С помощью Kubernetes может быть автоматизировано:

Обнаружение служб и балансировка нагрузки – в случае необходимости Kubernetes может предоставить доступ к контейнеру, с помощью DNS-имени или IP-адреса. Если трафик к контейнеру резко вырастет, он может поднять дополнительный экземпляр контейнера, чтобы сбалансировать нагрузку и распределить сетевой трафик.

Оркестрация хранилища – Kubernetes позволяет автоматически монтировать в контейнер определенную систему хранения.

Развертывание и откат – с помощью Kubernetes можно описать желаемое состояние контейнеров, благодаря чему он может изменить

фактическое состояние. Например, создание новых контейнеров для развертывания, а также удаление существующих и перенос всех их ресурсов в новый контейнер может быть полностью автоматизировано.

Эффективное распределение ресурсов – Kubernetes имеет кластер узлов, на которых он запускает контейнезированные приложения. Каждое такое приложение должно иметь информацию о том, сколько процессорного времени и памяти ему требуется. Kubernetes будет размещать контейнеры на узлах таким образом, чтобы максимально эффективно использовать ресурсы.

Самовосстановление – контейнеры внутри Kubernetes перезапускаются при выходе из строя. Не прошедшие проверку работоспособности контейнеры заменяются или уничтожаются, на них не приходит трафик, пока они не будут готовы к работе.

Управление конфигурацией и секретами – с помощью Kubernetes можно хранить и управлять конфигурацией и конфиденциальной информацией, например настройками приложений, паролями или SSH ключами. Добавлять и обновлять их можно не собирая образы контейнеров заново и не раскрывая секреты.

В рамках своей экосистемы Kubernetes содержит ряд абстракций, которые представляют состояние системы: развернутые контейнеризованные приложения и рабочие нагрузки, связанные с ними сетевые и дисковые ресурсы и другую информацию о том, что делает кластер. Эти абстракции представлены объектами в API Kubernetes.

Основные объекты Kubernetes включают в себя:

Pod (под) – базовая единица Kubernetes. Включает в себя один или несколько контейнеров, которым гарантирован запуск на одном узле.

Service (служба) – способ предоставления сетевого доступа к приложению, представленного в виде одного или нескольких подов.

Volume (том) – ресурс для хранения, а также совместного использования данных из контейнеров, запущенных на одном узле.

Namespace (пространство имен) – объект, необходимый для механизма изоляции группы объектов внутри одного кластера.

Kubernetes также содержит абстракции более высокого уровня, которые управляются контроллерами, которые, в свою очередь, создают базовые объекты и предоставляют дополнительные функции. Среди них:

ReplicaSet – набор из нескольких реплик одного пода, которые запущены одновременно.

Deployment – объект, служащий для декларативного обновления подов. Deployment создает ReplicaSet, который уже запускает необходимое количество подов.

StatefulSet – объект, необходимый для развертывания приложений с состоянием. Похож на Deployment за исключением того, что создает поды напрямую, и запускает их по очереди, а не все сразу.

DaemonSet – набор из нескольких реплик пода, каждый из которых запущен на всех узлах кластера (или на подмножестве узлов, если указан селектор меток).

Job – объект для создания одного или несколько подов, которые должны успешно завершиться.

**Третий раздел «Язык программирования Go»** посвящен языку программирования Go.

Go (также известен как Golang) – компилируемый многопоточный язык программирования, разработанный внутри компании Google. Является «родным» для Kubernetes.

Разрабатывался как язык программирования, главной целью которого было создание высокоэффективных программ, которые будут работать на распределенных системах и многоядерных процессорах.

Go создавался в расчёте на то, что программы, написанные на нем, будут компилироваться в объектный код и исполняться на процессоре без промежуточного ПО, например виртуальной машины. В связи с этим одним

из критериев выбора некоторых архитектурных решений была возможность обеспечить максимально быструю скорость компиляции.

Синтаксис Go содержит некоторые конструкции, которые направлены на то, чтобы код оставался кратким и удобно читаемым. Точка с запятой так же как и в C обозначают окончание операторов, но не является обязательной, так как компилятор сам проставит ее где необходимо. Методы могут возвращать несколько значений, а результат функции в паре со значением ошибки является общепринятым подходом к обработке исключений. Имеется синтаксис для инициализации параметров структуры по именам и для инициализации хеш-таблиц и срезов.

В Go существует ряд встроенных типов, в том числе численные, логический и строчный.

Вместо наследования существуют два механизма. Первый это встраивание, которое можно рассматривать как композицию. Второй – интерфейсы. Они представляют собой класс типов и обеспечивают полиморфизм во время выполнения. Объект, имеющий тип интерфейса, в то же время имеет и другой тип, который реализует данный интерфейс.

В Go есть встроенные средства для создания параллельных программ. Параллелизм в данном случае относится не только к использованию нескольких ядер центрального процессора для различных задач, но и к асинхронному выполнению медленных операций, таких как получения данных из базы данных или по сети.

Основной примитив параллелизма в Go это горутина, легковесный процесс. Вызов функции с использованием ключевого слова **go** перед ним запускает функцию в новой горутине.

**Четвертый раздел «Реализация приложения»** посвящен реализации приложения для автоматизации контроля и мониторинга состояния кластера Kubernetes.

Приложение проверяет узлы кластера Kubernetes, рабочие нагрузки, доступность сервера API Kubernetes и статус базы данных etcd. Эти проверки

не зависят друг от друга, а следовательно могут выполняться параллельно. Также независимо друг от друга происходит проверка рабочих нагрузок в разных пространствах имен.

После всех проверок, будет выбран «цвет» облака: зеленый, желтый или красный. Вся информация о состоянии кластера, а также полезная нагрузка в виде времени последней проверки будет собрана в JSON-документ и доступна в таком виде по HTTP.

Для обеспечения минимальных сетевых задержек приложение должно работать в кластере Kubernetes.

## ЗАКЛЮЧЕНИЕ

Облачные технологии и Kubernetes, как их неотъемлемая часть, крепко внедрились в современную ИТ-инфраструктуру. В настоящее время почти каждая компания имеет по крайней мере один кластер Kubernetes или его дистрибутив, для развертывания своих приложений.

Цель работы, которая заключалась в создании приложения для автоматизации контроля и мониторинга состояния кластера Kubernetes, была успешно выполнена.

Также в ходе данной работы были полностью выполнены поставленные задачи:

- определены наиболее важные компоненты Kubernetes;
- определены инструменты для реализации приложения;
- спроектирована архитектура приложения и выявлены места, в которых была применена параллелизация.

**Отдельные части магистерской работы были опубликованы в студенческом научном журнале:**

Радюшин В. С. Устранение неполадок etcd split-brain в кластере Kubernetes // Студенческий: электрон. научн. журн. — 2023. — No 16(228).

Радюшин, В.С. Проверка состояния кластера Kubernetes // Студенческий: электрон. научн. журн. — 2023. — No 20(232).

**Основные источники информации:**

1. Foster, I., Gannon, D. Cloud Computing for Science and Engineering. — The MIT Press, 2017. — 392 с.
2. OpenStack Cloud Computing Cookbook: Over 100 practical recipes to help you build and operate OpenStack cloud computing, storage, networking, and automation / K. Jackson, C. Bunch, E. Sigler, Denton J. — 4 изд. — Packt Publishing, 2018. — 398 с.
3. Лукша, М. Kubernetes в действии. — Москва : ДМК Пресс, 2019. — 672 с.
4. Сайфан, Д. Осваиваем Kubernetes. Оркестрация контейнерных архитектур. — Санкт Петербург : Питер, 2019. — 400 с.
5. Билджин, И., Роланд, Х. Паттерны Kubernetes. Шаблоны разработки собственных облачных приложений. — Санкт Петербург : Питер, 2020. — 320 с.