МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Университет «Дубна»

Институт системного анализа и управления
Кафедра системного анализа и управления

ДНЕВНИК
производственной практики

Научно-исследовательская работа

Студент _____Гришин Сергей Алексеевич_____ /_____/
Ф.И.О. подпись

Группа № 5015, 1 курс магистратуры, очная форма обучения

Направление подготовки: 27.04.03 Системный анализ и управление

Место прохождения практики: Государственный университет «Дубна»

Руководитель практики:
от университета _____доц. Белов М.А._____
должность, Ф.И.О.

Сроки прохождения практики с 06.02.2023 по 04.06.2023

Дубна, 2023 г.

Дата	Выполняемая работа	Кол-во часов	Отметка о выполнении	Подпись непосредственного руководителя по месту прохождения практики
В течение семестра		*96*	*Выполнено*	
В течение семестра		80	*Выполнено*	
В течение семестра		112	*Выполнено*	
		288		

Руководитель практики:

от университета _____доц. Белов М.А._____ /_____/
должность, Ф.И.О. подпись

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Университет «Дубна»

Кафедра системного анализа и управления

«Утверждаю»
Заведующий кафедрой
_____ /проф. Черемисина Е.Н./

Отчет по практике
Научно-исследовательская работа

Учебно-практическая система непрерывной интеграции и контейнеризации для разработки информационных систем управления большими данными
тема

Студент-практикант_____Гришин Сергей Алексеевич_____

Группа студента __5015___

Место прохождения практики Государственный университет «Дубна»

Руководитель от кафедры_____доц. Белов М.А._____

Рекомендуемая оценка _____
(оценка) (подпись руководителя от университета)

Дата представления отчета «____» _____ 20___.

Студент-практикант _____
(подпись)

Содержание

Актуальность

На данный момент процесс разработки ПО требует большого количества автоматизации. Использование систем непрерывной интеграции вместе с системами управления версиями и контейнерными технологиями позволяет оптимальным образом организовать процесс создания версий ПО

Для данного процесса была создана аббревиатура CI/CD (continuous integration/continuous delivery). Благодаря использованию CI/CD появляется возможность рано и часто тестировать новые версии ПО, что позволяет выявить несоответствие запросам целевой аудитории. CI/CD позволяет свести к нулю ручной труд при выполнении рутинных операций: сборки и предоставления новых версий ПО. Все эти операции производятся с помощью специальных инструментов Jenkins, GitLab CI и др.

За счёт автоматизации процесс CI/CD решает ряд важных проблем:

- Улучшает качество кода,
- Устраняет разобщённость между разработкой и операционной деятельностью,
- Повышает скорость внедрения нового функционала,
- Позволяет быстро исправлять баги в ПО,
- Отсеивает некоторый процент ошибок на пути к выпуску версии ПО.

Решая перечисленные проблемы, CI/CD нормализует работу и коммуникацию внутри команды разработки. Наличие CI/CD делает процесс разработки и выпуска ПО более компактным и эффективным. Пока компьютеры выполняют рутину, разработчики могут сосредоточиться на решении бизнес-задач и проведении экспериментов.

При проведении экспериментов особо важным является возможность точно воссоздать условия в, которых они были проведены. При разработке ПО, данный вопрос проявляется со стороны поддерживания большого количество платформ. Крайне удобным средством поддержания большого количества разнородных платформ является использование виртуализации пространства выполнения ПО (контейнерные технологии).

Первые контейнерные технологи появились еще в 1982, а с их развитием, конечному пользователю, как и разработчику ПО все меньше приходится заботиться о правильной настройке всех зависимостей, как операционной системы, так и библиотек программирования. Наиболее сильно это проявляется в наши дни, когда ПО работает с высокотехнологичными хранилищами данных, и основывает свою архитектуру на микросервисах.

Так же свое неотъемлемое место контейнерные технологии нашли в уже описанном выше CI/CD, где они играют роль легковесных тестовых окружений, которые крайне быстро создаются и не требуют отчистки, в отличие от использования реальных тяжеловесных ОС.

Данный плюс крайне сильно проявляется при работе с системами больших данных, которые зачастую требуют долго и скрупулезной настройки, а в случае их выхода из строя повторения всего процесса установки и настройки.

В больших распределённых системах данные распределены по большому количеству машин. Если данные физически находятся на одном сервере, а обрабатываются на другом – расходы на передачу данных могут превысить расходы на саму обработку. Поэтому одним из важнейших принципов проектирования решений для работы с такими объемами данных является принцип локальности данных, при котором по возможности данные обрабатываются на той же машине, на которой хранятся. Все современные средства работы с большими данными так или иначе следуют этому принципу.

Создание учебно-практической системы такого характера предоставит возможность быстрого развертывания среды разработки, которая сможет поддерживать требуемые для проекта операционные системы и языки программирования.

Задачи

В рамках выполнения данной работы планируется провести следующие деятельности, нацеленные на создание прототипа общей структуры проекта и выявление наиболее подходящего набора технологий:

- Анализ целевой аудитории.
- Анализ существующих систем контроля версиями ПО,
- Анализ существующих средств управления на основе кода,
- Выявление возможностей взаимодействия проанализированных систем между собой,
- Анализ пригодности существующих систем оркестрации контейнерами для процесса развертывания решения,
- Разработка архитектуры проекта.

В частности, планируется к рассмотрению следующие решения:

- Docker,
- Kubernetes,
- Terraform,
- GitLab,
- GitFlic,
- Bitbucket,
- VMware Tanzu,
- VMware vCenter,
- OpenVZ.

В результате проделанной работ планируется получить функционирующий прототип программно-технологического CI/CD решения, имеющего возможность быстрого развертывания и настройки под требуемые задачи.

Анализ пригодности существующих систем оркестрации контейнерами для процесса развертывания решения

Реализация работы планирует использование контейнерных технологий как основного инструмента. Сейчас существуетмножество систем для управления контейнерами, большинство из них основывают свою работу на универсальном формате Open Container Initiative (OCI), что в свою очередь расширяет перечень возможных инструментов реализации [1].

Наиболее популярными решениями на данный момент являются:

- Docker.
- Kubernetes
- CRI-O
- Containerd

Kubernetes в свою очередь работает с контейнерами не на прямую, а используя вышеупомянутые технологии Docker, CRI-O, Containerd [1]. Так же данная технология обладает высокой надежностью и возможностью масштабирования, но при этом требует не тривиальной настройки. Использование этой технологии в данном проекте не является целесообразным так как, главной целью является упрощение процесса создания среды разработки, что исключает использование сложных в найстройке решений. Так же использовании столь мощного иструмента в данно проекте будет являтся не целесообразным.

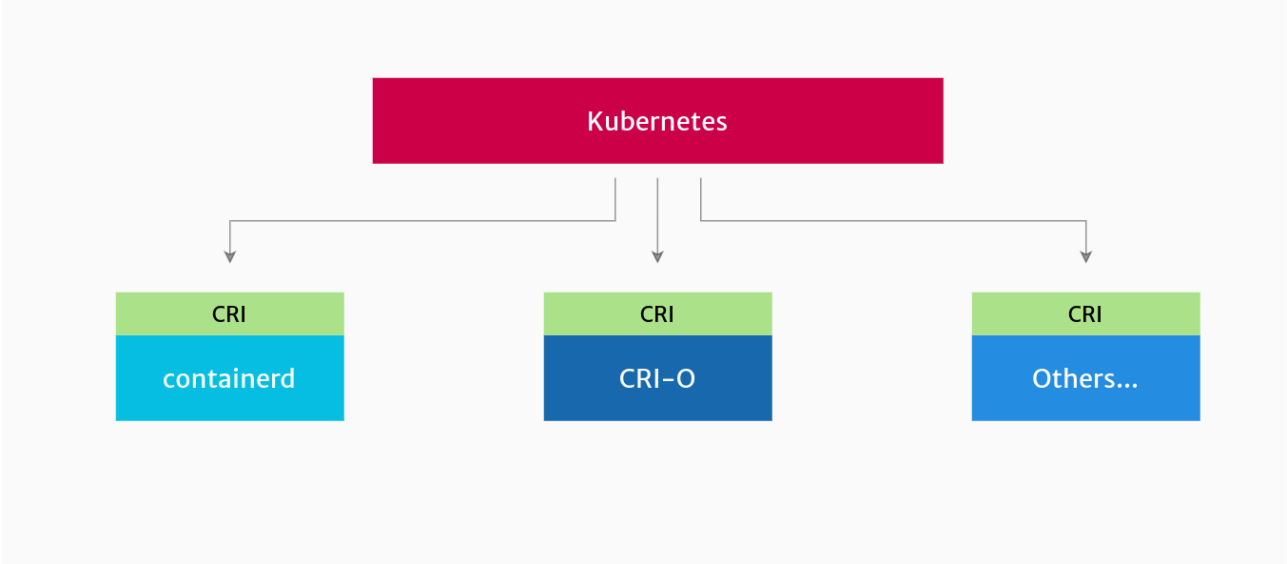

Рисунок 1Схема работы Kubernetes с инструментами управления контейнерами

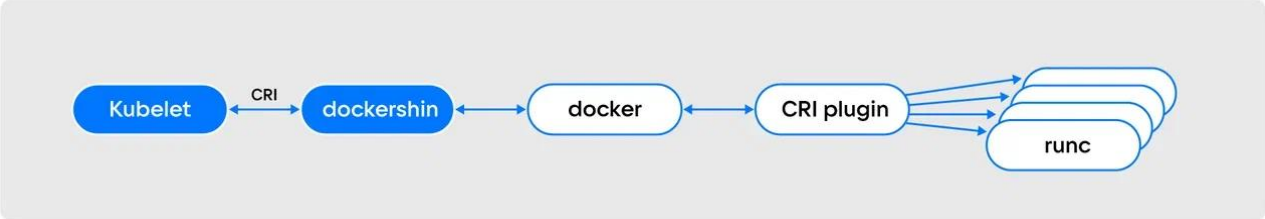

Рисунок 2Схема работы Kubernetes Docker

Рисунок 3Схема работы Kubernetes с CRI-O

Containerd в свою очередь является более низкоуровневым ответвлением проекта Docker и на данный момент сам Docker использует его в качестве API для работы с runcкоторый в свою очередь отвечает за выполнение контейнеров в операционной системе.

CRI-O на данный момент используется в такх больших компаниях как VK. И нативно поддерживается Kubernetes, в отличие от Docker, для использования которого Kubernetes использует dockershim [2]. CRI-O, в отличие от containerd, имеет ограниченный набор функций и внутренних компонентов. Это уменьшает потенциальную поверхность атаки и снижает уязвимость компонента [2].

На данный момент Docker преобладет в open source проектах, а так же широ используется его инструмент для каскадного запуска Docker-compose, который так же имеет простой и понятный cli и простую систему создания конфигурации, основанную на yaml файлах. Данные качетсва являются решеющими для создания прототипа системы, с максимально низким порогом вхождения, что в свою очередь послужило поводом остановиться на этом наборе технологий (Docker в связке с Docker-compose).

Анализ существующих систем контроля версиями ПО

Использованание систем управления версиями является неотъемлемой частью процесса разработки ПО. Существует огроное количество решений выполняющих данную задачу. При анализе существующих решений особое внимание уделялось простоте развертывания и конфигурирования систем, а так же наличие контейнеризрованного способа установки.

GitFlic является Российским аналогом GitHub, который написан на Java. Система не предоставляет готовго контейнерного решения, а ее устанвка является не тривиальным процессом, который требует большого количетсва зависимостей. Перечисленный минусы исключают использование данного инструмента при реализации прототипа проекта.

Gitlab предоставляет все требуемые для реализации прототипа проекта возможности включая:

- Наличие образа контейнера
- Открытый исходный код
- Полный набор инструментов CI/CD
- Является корпоративным стандартом

Bitbucket существует как исключительно облачное решение с возможность приобретения платных планов использовани, являеттся более медленным по сравнению с Gitlab. Так же Gitlab обладает большей функциональностью «из коробки».

Исходя из вышеперечисленного для реализации прототипа проекта был выбран Gitlab, с уаором на использование его в связке с Docker и Docker-compose.

Вывод

Список литературы

1. Различия между Docker, containerd, CRI-O и runc [Электронный ресурс] — Режим доступа: https://habr.com/ru/companies/domclick/articles/566224/, свободный (дата обращения 16.05.2023).
2. Жизнь после Docker: как команда VK Cloud переходила на CRI-O [Электронный ресурс] — Режим доступа: https://habr.com/ru/companies/vk/articles/707312/, свободный (дата обращения 30.05.2023).