

УТВЕРЖДЕНА

Решением Ученого совета
АНО ВО «Центральный университет»
«07» марта 2024 г.
Протокол №1

**Рабочая программа дисциплины (модуля)
«Промышленная разработка»**

Направление подготовки: 02.04.01 Математика и компьютерные науки

Направленность (профиль) подготовки: Backend-разработка

Квалификация (степень) выпускника: магистр

Форма обучения: очная

Срок освоения программы: 2 года

Год набора: 2024

**Москва
2024**

Содержание

1. Краткая характеристика дисциплины (модуля).....	3
2. Перечень планируемых результатов обучения	4
3. Тематический план	6
4. Содержание дисциплины (модуля)	6
5. Учебно-методическое обеспечение	8
6. Материально-техническое обеспечение	8
7. Методические и оценочные материалы.....	10

1. Краткая характеристика дисциплины (модуля)

Рабочая программа дисциплины (модуля) «Промышленная разработка» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования – магистратура по специальности 02.04.01 Математика и компьютерные науки, профиль Backend-разработка, утвержденный приказом Министерства науки и высшего образования Российской Федерации № 810 от 23.08.2017 года.

Изучение дисциплины (модуля) «Промышленная разработка» имеет важное значение для будущих специалистов в области информатики и программной инженерии, поскольку оно позволяет им получить практические навыки и знания, необходимые для разработки сложных программных систем, и подготовить их к успешной карьере в этой области.

Изучение промышленной разработки также имеет значение для повышения качества и эффективности разработки программных систем, поскольку оно позволяет студентам понять основные принципы и методы промышленной разработки, что может привести к созданию более надежных, масштабируемых и поддерживаемых систем.

Место дисциплины (модуля) в структуре образовательной программы

Настоящая дисциплина (модуль) включена в учебный план по программе подготовки магистратуры по направлению 02.04.01 Математика и компьютерные науки, профиль Backend-разработка и входит в обязательную часть Блока 1.

Дисциплина (модуль) изучается на 1 курсе во 2 семестре.

Цель изучения дисциплины (модуля): овладение студентами современными методами и технологиями разработки программных систем, позволяющими создавать сложные и масштабируемые приложения, соответствующие требованиям промышленной разработки.

Задачи изучения дисциплины (модуля):

- формирование знаний о видах облачных ресурсов и различий между способами доставки и развертывания ПО;
- формирование знаний основных гарантий обеспечения надежной инфраструктуры;
- формирование знаний категоризации ресурсов для разворачивания конвейеризируемой инфраструктуры;
- формирование умения работать с инструментами контейнеризации, собирать образы и запускать контейнеры с различными параметрами и использовать сложные сценарии, такие как мультитадийные сборки;
- формирование умения применять инструменты для развертывания мутабельной и иммутабельной инфраструктуры;
- формирование умения анализировать отчеты лучших devops-практик и применять их для улучшения процессов разработки;
- формирование навыка выгружать артефакты сборки проектов, включая образы Docker и результаты тестирования, и хранить секреты в системах непрерывной интеграции;
- формирование навыка настраивать окружение разработчика простого и среднего уровня.

2. Перечень планируемых результатов обучения

Компетенции, формируемые в результате освоения дисциплины (модуля) при проведении учебных занятий в форме контактной работы обучающихся с педагогическими работниками Университета и в форме самостоятельной работы обучающихся:

Компетенция	Содержание компетенции	Индикатор компетенции	Перечень планируемых результатов обучения по дисциплине (модулю)
УК-2.	Способен управлять проектом на всех этапах его жизненного цикла	УК-2.1.	Знает основные методологии управления проектами, ключевые этапы жизненного цикла проекта и инструменты для планирования и контроля
		УК-2.2.	Умеет разрабатывать проектную документацию, устанавливать цели и задачи проекта, а также эффективно распределять ресурсы и управлять рисками на всех этапах его реализации
		УК-2.3.	Имеет практический опыт в управлении реальными проектами, включая планирование, исполнение и завершение, а также в оценке результатов и проведении анализа успешности проекта
ОПК-2.	Способен создавать и исследовать новые математические модели в естественных науках, совершенствовать и разрабатывать концепции, теории и методы	ОПК-2.1.	Знает основные математические модели и методы, используемые в естественных науках, включая статистическое моделирование, дифференциальные уравнения и численные методы, а также современные подходы к исследованию и анализу данных
		ОПК-2.2.	Умеет разрабатывать и адаптировать математические модели для решения конкретных проблем в естественных науках, проводить их анализ и верификацию, а также интерпретировать полученные результаты в контексте научных исследований
		ОПК-2.3.	Имеет практический опыт создания и исследования математических моделей в рамках научных проектов или исследований, включая участие в публикациях, конференциях или коллаборациях, где были разработаны и апробированы новые концепции и методы

ПК-3.	Способен решать задачи профессиональной деятельности, формулировать результат, увидеть следствия полученного результата	ПК-3.1.	Знает основные принципы и методы решения задач профессиональной деятельности, а также способы формулирования и представления результатов, включая анализ последствий и их значимость в контексте проекта
		ПК-3.2.	Умеет применять математические и компьютерные методы для решения конкретных задач, формулировать четкие и обоснованные результаты, а также анализировать их последствия для дальнейших действий и решений
		ПК-3.3.	Имеет практический опыт в решении профессиональных задач, включая участие в проектах, где были получены результаты и проанализированы их следствия, что способствовало принятию обоснованных решений
ПК-6.	Способен разрабатывать программное обеспечение для решения прикладных задач в сфере профессиональной деятельности	ПК-6.1.	Знает основные языки программирования, методы разработки программного обеспечения, а также принципы проектирования и архитектуры программных систем, применяемых в конкретной предметной области
		ПК-6.2.	Умеет анализировать прикладные задачи, разрабатывать алгоритмы и реализовывать их в виде программного обеспечения, используя современные инструменты и технологии, а также проводить тестирование и отладку созданных решений
		ПК-6.3.	Имеет практический опыт разработки программного обеспечения в рамках реальных проектов, включая участие в командах, где были успешно реализованы решения для конкретных прикладных задач в сфере профессиональной деятельности

3. Тематический план

№ п/п	Наименование раздела дисциплины (модуля)	Трудоемкость, академические часы				ТКУ (текущий контроль успеваемости)
		Очная форма				
		Аудиторная работа		Контроль	Самостоятельная работа	
Лекции	Семинары (практические занятия)					
1	Devops: основные практики	2	2		20	Домашнее задание Подготовка к семинару
2	Тестирование ПО	4	4		20	Домашнее задание Подготовка к семинару
3	Основы контейнеризации	4	4		20	Домашнее задание Подготовка к семинару
4	Продвинутые темы контейнеризации	4	4		22	Домашнее задание Подготовка к семинару
5	Непрерывная интеграция и доставка	4	4		22	Домашнее задание Подготовка к семинару
6	Непрерывное развёртывание	4	4		20	Домашнее задание Подготовка к семинару
7	Kubernetes	4	4		20	Домашнее задание Контрольная работа
8	Observability	4	4		20	Домашнее задание
	<i>Зачет с оценкой</i>			4		
	Итого:	30	30	4	164	
	Объем дисциплины (модуля) (в ак. ч.)	228				
	Объем дисциплины (модуля) (в зач. ед.)	6				

4. Содержание дисциплины (модуля)

№п/п	Наименование раздела дисциплины (модуля)	Содержание дисциплины (модуля) по темам
1	Devops: основные практики	Истории развития devops. Понятия devops и DevOps — основные отличия. Культура и методы devops. Три принципа devops. Паттерны и антипаттерны devops. Отчёты State of Devops и деление по основным практикам devops
2	Тестирование ПО	Принципы тестирования. Психология тестирования. Пирамида тестирования: базис, цели, объекты тестирования. Уровни тестирования: модульное тестирование, интеграционное тестирование, системное тестирование, приёмочное тестирование. Связь V-модели и пирамиды тестирования. Методы динамического тестирования. Инструменты тестирования: JUnit. Инструменты Mock-тестирования: Mockito, Hamcrest. Эмуляция браузера в тестировании: Selenium. Перехват запросов: Postman, Charles. Разработка через тестирование. Артефакты автоматического тестирования: JUnit XML, Cobertura Code Coverage

3	Основы контейнеризации	Linux Namespaces. Отличия виртуализации от контейнеризации. Инструменты контейнеризации: Docker, Podman. Понятия образа и Volume, их отличия. Запуск контейнеров: параметры запуска, проброс портов, запуск в интерактивном режиме. Docker и томы. Виды томов: bind, mount, tmpfs. Преимущества использования tmpfs-томов. Образы: OverlayFS и слои. Виды слоёв. Основные команды работы со слоями. Сборки multi-stage. Загрузка образов в Registry, устройство Registry. Способы авторизации в Registry. Примеры Docker Registry: GitLab Container Registry, DockerHub. <i>Linux Namespaces</i>
4	Продвинутые темы контейнеризации	Docker Compose: сервисы, интеграция сервисов, Healthcheck, порядок запуска сервисов. Безопасность в Docker: Docker in Docker, Linux Capabilities. Kaniko как безопасный инструмент сборки образов. Исследование образов и контейнеров на безопасность
5	Непрерывная интеграция и доставка	Концепции непрерывной интеграции. Понятия стадий, задач. Передача артефактов между стадиями и задачами. Матрица сборки проекта. Примеры систем непрерывной интеграции — GitLab CI, GitHub Actions. Выгрузка артефактов сборки проектов: образов Docker, результатов тестирования, уязвимости сборки проектов. Хранение секретов в системах непрерывной интеграции. Агенты для интеграции сервисов
6	Непрерывное развёртывание	Отличие систем управления конфигурации от систем обеспечения облачной инфраструктуры. Виды облачных ресурсов. Настройка облачной инфраструктуры посредством инструмента Terraform. Понятие иммутабельной инфраструктуры. Виды развёртываний ПО. Интеграция систем доставки и систем развёртывания
7	Kubernetes	Отличия оркестраторов от систем контейнеризации. Основные понятия в Kubernetes. Как разворачивать кластер в Kubernetes при помощи MiniKube. Фундамент Kubernetes: containerd, etcd. Основные абстракции в Kubernetes-кластере по уровню развёртывания сервисов: Pod, Service — типы развёртывания сервисов, Ingress. Основные абстракции в Kubernetes-кластере по отказоустойчивости: ReplicaSet, Deployment. Основные абстракции в Kubernetes-кластере по сохранению состояния: PersistentVolumeClaim, StatefulSet, Volumes — эфемерные хранилища. Хранение секретов в Kubernetes-кластере. Подключение внешних хранилищ секретов. Управляющие ресурсы в Kubernetes-кластере: Job, CronJob, DaemonSet. Управление безопасностью в Kubernetes-кластере: RBAC, ServiceAccount, Network Policy. Настройка TLS
8	Observability	Понятие отслеживаемости. Основные гарантии обеспечения: SLA, SLO, SLI. Метрики для отслеживания стабильности инфраструктуры. Виды отслеживаемости: мониторинг, логирование, трассировка. Виды систем сбора метрик и логирования. Отличие стеков ELK (ElasticSearch, LogStash, Kibana) от стека LGTM (Loki, Grafana, Tempo, Mimir). Интеграция инфраструктуры отслеживания в Kubernetes-кластере: Prometheus Service Discovery

5. Учебно-методическое обеспечение

Университет располагает полным набором лицензионного и свободно распространяемого программного обеспечения, включая продукты отечественного производства.

Каждый студент в течение всего периода обучения получает индивидуальный неограниченный доступ к электронно-библиотечной системе и электронной информационно-образовательной среде университета. Эти системы предоставляют возможность доступа к ресурсам из любой точки, где есть подключение к сети Интернет, как на территории университета, так и за его пределами.

Студентам обеспечен удаленный доступ к современным профессиональным базам данных и информационным справочным системам.

Основная литература:

1. Чернышев, С. А. Принципы, паттерны и методологии разработки программного обеспечения : учебник для вузов / С. А. Чернышев. — Москва : Издательство Юрайт, 2025. — 176 с. — (Высшее образование). — ISBN 978-5-534-14383-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/567946>.

Дополнительная литература:

1. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. — СПб.: Питер, 2022. — 1120 с. — ISBN 978-5-4461-1155-8.

6. Материально-техническое обеспечение

Университет располагает материально-технической базой, соответствующей действующим противопожарным правилам и нормам и обеспечивающей проведение всех видов дисциплинарной и междисциплинарной подготовки, практической и научно-исследовательской работ обучающихся, предусмотренных учебным планом.

Помещения, которые представляют собой учебные аудитории для проведения занятий лекционного типа, занятий семинарского (практического) типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, а также помещения для самостоятельной работы и помещения для хранения и профилактического обслуживания учебного оборудования. Помещения укомплектованы специализированной мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории.

Изучение дисциплины (модуля) обеспечивается в учебных аудиториях, оснащенных:

— столами и стульями;

— компьютерной техникой;

— специализированным оборудованием, включая демонстрационное оборудование.

Помещения для самостоятельной работы обучающихся, в том числе приспособленные для использования инвалидами и лицами с ограниченными возможностями здоровья, оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа к в электронную информационно-образовательную среду Университета.

Обучающимся предоставляется доступ (в том числе удаленный) к ресурсам информационно-телекоммуникационной сети «Интернет», электронным ресурсам (в том числе электронным библиотечным системам, современным профессиональным базам данных и информационным справочным системам):

№	Наименование портала (издания, курса, документа)	Ссылка
1.	Научная электронная библиотека elibrary.ru	https://elibrary.ru/defaultx.asp

	библиотека	
2.	База данных для IT-специалистов	https://habr.com
3.	База данных ScienceDirect	https://www.sciencedirect.com
4.	Официальный сайт Министерства науки и высшего образования Российской Федерации	https://minobrnauki.gov.ru/
5.	Федеральный портал «Российское образование»	https://www.edu.ru/
6.	Информационная система "Единое окно доступа к образовательным ресурсам"	http://window.edu.ru/
7.	Единая коллекция цифровых образовательных ресурсов	http://school-collection.edu.ru/
8.	Федеральный центр информационно - образовательных ресурсов	http://fcior.edu.ru/

Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), в том числе комплект лицензионного программного обеспечения, современные профессиональные базы данных и информационные справочные системы:

Наименование ПО	Производство	Лицензионное / свободно распространяемое
Операционные системы:		
Microsoft Imagine (Windows Client, Server)	зарубежное	лицензионное
Браузеры:		
Яндекс.Браузер	отечественное	свободно распространяемое
Google Chrome	зарубежное	свободно распространяемое
Офисные приложения:		
Microsoft Imagine (Visio, OneNote)	зарубежное	лицензионное
TeXstudio	зарубежное	свободно распространяемое
Adobe Acrobat Reader	зарубежное	свободно распространяемое
Программное обеспечение для планирования и учета времени:		
Toggle app	зарубежное	свободно распространяемое
Системы управления проектами:		
Microsoft Imagine (Project)	зарубежное	лицензионное
Системы управления базами данных:		
Microsoft Imagine (SQL Server)	зарубежное	лицензионное
Системы резервного копирования (backup):		
Acronis Backup Advanced for HyperV	зарубежное	лицензионное
Справочно-правовые системы:		
КонсультантПлюс: справочно-правовая система	отечественное	лицензионное
Средства антивирусной защиты:		
Kaspersky Endpoint Security для бизнеса Стандартный Russian Edition	отечественное	лицензионное
Среды разработки:		
Visual Studio Code	зарубежное	свободно распространяемое
Bash (Unix shell)	зарубежное	свободно распространяемое
Anaconda	зарубежное	свободно распространяемое
Robotic Operating System	зарубежное	свободно распространяемое
CopelliaSim	зарубежное	свободно распространяемое
Google Colaboratory	зарубежное	свободно распространяемое
Пакеты программных средств и библиотек:		
AutoPsy	зарубежное	свободно распространяемое

Interactive Disassembler (IDA)	зарубежное	свободно распространяемое
Системы управления библиографической информацией:		
Zotero	зарубежное	свободно распространяемое
Сервисы и службы:		
Bind	зарубежное	свободно распространяемое
Docker	зарубежное	свободно распространяемое

7. Методические и оценочные материалы

Методические указания для обучающихся по освоению дисциплины (модуля)

В процессе изучения дисциплины (модуля) «Промышленная разработка» в рамках текущего контроля успеваемости используются такие виды учебной работы, как лекции, семинары, контрольная работа, домашние задания, а также различные виды самостоятельной работы обучающихся по заданию преподавателя, направленные на развитие навыков профессиональной лексики, закрепление практических профессиональных компетенций, поощрение инициатив.

Лекция – систематическое, последовательное, монологическое изложение преподавателем учебного материала, как правило, теоретического характера.

В процессе лекций рекомендуется вести конспект лекций: кратко и схематично фиксировать основные идеи, выводы и обобщения лекции; выделять важные мысли, ключевые слова и термины. Необходимо отметить вопросы или материалы, которые вызывают затруднения, и попытаться найти ответы в рекомендованной литературе. Если разобраться в материале не удастся, следует сформулировать вопрос и задать его преподавателю на консультации или во время семинарского (практического) занятия.

Участие в семинаре (аудиторная работа) – активная работа студента на семинаре, его ответы на вопросы преподавателя и участие в дискуссии.

Для успешного участия в семинаре студентам рекомендуется заранее ознакомиться с темой обсуждения, прочитать необходимые материалы и подготовить вопросы. Важно активно слушать и вовлекаться в дискуссию, высказывая свои мнения и аргументируя их. При ответах на вопросы преподавателя стоит быть уверенным, четким и логичным, опираясь на изученный материал. Также полезно поддерживать диалог с однокурсниками, чтобы обогатить обсуждение и расширить свои знания.

Домашнее задание – набор задач по темам недели.

При работе над домашними заданиями важно внимательно ознакомиться с требованиями и сроками выполнения. Рекомендуется разбивать задания на этапы, чтобы избежать перегрузки и лучше усвоить материал. Использовать различные источники информации, включая учебники и онлайн-ресурсы, для более глубокого понимания темы.

Контрольная работа – письменная работа с набором задач, которые нужно решить за ограниченное время.

Цель контрольной работы - получить специальные знания по одной или нескольким темам дисциплины (модуля) и продемонстрировать навыки их практического применения.

Самостоятельная работа – работа студентов, направленная на углубленное изучение отдельных тем и вопросов учебной дисциплины (модуля).

В процессе самостоятельной работы студенты взаимодействуют с рекомендованными материалами при минимальном участии преподавателя. Задачи студента включают работу с конспектами лекций (обработка текста), повторное изучение учебных материалов планов и тезисов ответов, изучение дополнительных тем, выполнение учебно-исследовательских заданий и другое.

Система оценивания результатов обучения по дисциплине (модулю)

Критерии получения уровня и оценивания сформированности компетенций по дисциплине (модулю) «Промышленная разработка»

Оценивание уровня учебных достижений обучающихся по дисциплине (модулю) осуществляется в виде текущего контроля успеваемости и промежуточной аттестации.

Промежуточная аттестация по дисциплине (модулю) осуществляется в форме *зачета с оценкой*, при этом проводится оценка компетенций, сформированных по дисциплине.

Для оценивания текущего контроля успеваемости и промежуточной аттестации используется десятибалльная шкала оценивания, которая соотносится с традиционной пятибалльной шкалой следующим образом:

Десятибалльная оценка	Пятибалльная оценка	Оценка за зачет	Общая характеристика результата обучения по дисциплине (модулю)
10	Отлично	Зачтено	Студент полностью владеет знаниями, изложенными в рабочей программе, и глубоко осмысляет дисциплину (модуль). Он самостоятельно и логически последовательно отвечает на все вопросы, акцентируя внимание на наиболее важном. Умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя ключевые моменты и устанавливая причинно-следственные связи. Четко формулирует ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты дисциплины (модуля) с практическими задачами.
9	Отлично	Зачтено	
8	Отлично	Зачтено	
7	Хорошо	Зачтено	Студент обладает знаниями предмета почти в полном объеме рабочей программы и самостоятельно, логически последовательно и всесторонне отвечает на все вопросы, акцентируя внимание на наиболее значимых моментах. Он умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя его ключевые аспекты и устанавливая причинно-следственные связи. Формулирует свои ответы, уверенно
6	Хорошо	Зачтено	

Десятибалльная оценка	Пятибалльная оценка	Оценка за зачет	Общая характеристика результата обучения по дисциплине (модулю)
			интерпретирует результаты анализов и других исследований, а также решает сложные ситуационные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты предмета с практическими задачами.
5	Удовлетворительно	Зачтено	Студент обладает базовыми знаниями по дисциплине (модулю), но испытывает трудности при самостоятельных ответах и использует неточные формулировки. В ходе ответов он допускает ошибки, касающиеся сути вопросов. Студент способен решать только самые простые задачи и владеет лишь минимальным набором методов исследования.
4	Удовлетворительно	Зачтено	
3	Не сдан	Не зачтено	Студент не овладел обязательным минимумом знаний по предмету и не может ответить на вопросы, даже если преподаватель задает дополнительные наводящие вопросы.
2	Не сдан	Не зачтено	
1	Не сдан	Не зачтено	

Дисциплина (модуль) «Промышленная разработка» оценивается следующим образом:

Активность	Вес	Описание
Домашние задания	70%	За каждое из заданий можно набрать 10 баллов
Аудиторная работа	15%	На каждом семинаре студент может заработать баллы за интересные вопросы, работу на семинаре и выполнение заданий
Контрольная работа	15%	Контрольная работа, на которой оценивается процент правильных ответов и конвертируется в количество набранных баллов (так, за 100% правильных ответов студент получает 10 баллов)

Формула расчёта итоговой оценки по дисциплине (модулю) «Промышленная разработка»: « $0,7 \times$ среднее за домашние задания + $0,15 \times$ среднее за аудиторную работу + $0,15 \times$ контрольная работа».

Текущий контроль успеваемости обучающихся по дисциплине (модулю)

Примерные вопросы для подготовки к семинарам

Тестирование ПО

1. Какие основные принципы тестирования ПО?
2. Какова роль психологии тестирования в процессе разработки?
3. Что такое пирамида тестирования и какие уровни она включает?
4. В чем разница между модульным и интеграционным тестированием?
5. Каковы цели системного тестирования?
6. Как V-модель соотносится с пирамидой тестирования?
7. Какие методы динамического тестирования вы знаете?

8. Как работает инструмент JUnit для тестирования ПО?
9. Что такое Mockito и как он используется в тестировании?
10. Чем Hamcrest отличается от Mockito?
11. Какова роль Selenium в тестировании веб-приложений?
12. Как использовать Postman для перехвата запросов?
13. В чем преимущества использования Charles для тестирования?
14. Что такое разработка через тестирование (TDD)?
15. Какие артефакты создаются при автоматическом тестировании с помощью JUnit?
16. Как Cobertura Code Coverage помогает в тестировании?
17. Каковы основные подходы к тестированию API?
18. Какие типы тестов являются наиболее важными для веб-приложений?
19. Каковы основные проблемы, с которыми сталкиваются тестировщики?
20. Какие метрики используются для оценки качества тестирования?

Основы контейнеризации

1. Что такое Linux Namespaces и как они работают?
2. В чем разница между виртуализацией и контейнеризацией?
3. Какие инструменты контейнеризации вы знаете?
4. Каковы основные функции Docker?
5. Что такое Podman и чем он отличается от Docker?
6. Что такое образ контейнера и как он создается?
7. Какова роль Volume в контейнеризации?
8. В чем разница между bind и mount томами?
9. Каковы преимущества использования tmpfs-томов?
10. Что такое OverlayFS и как он используется в Docker?
11. Какие виды слоев существуют в образах контейнеров?
12. Каковы основные команды для работы со слоями в Docker?
13. Что такое сборки multi-stage и в чем их преимущества?
14. Как загружать образы в Registry?
15. Какие типы Registry существуют для Docker?
16. Как происходит авторизация в Docker Registry?
17. Приведите примеры Docker Registry.
18. Как можно управлять ресурсами контейнеров?
19. Каковы основные проблемы, с которыми сталкиваются при работе с контейнерами?
20. Как обеспечивается безопасность контейнеров?

Продвинутые темы контейнеризации

1. Что такое Docker Compose и как он используется?
2. Как организуются сервисы в Docker Compose?
3. Что такое Healthcheck и как его настроить в Docker?
4. Каков порядок запуска сервисов в Docker Compose?
5. Какие меры безопасности применяются в Docker?
6. Что такое Docker in Docker и как он используется?
7. Как Linux Capabilities влияют на безопасность контейнеров?
8. Что такое Kaniko и как он помогает в сборке образов?
9. Как исследовать образы и контейнеры на безопасность?
10. Какие инструменты можно использовать для анализа безопасности контейнеров?
11. В чем преимущества использования Docker Compose для микросервисов?
12. Как управлять конфигурацией сервисов в Docker Compose?
13. Какие лучшие практики безопасности следует учитывать при работе с Docker?
14. Каковы основные ошибки, которые могут возникнуть при использовании Docker?
15. Как тестировать безопасность контейнеров перед развертыванием?
16. Каковы преимущества использования многослойной архитектуры в контейнерах?
17. Как обеспечивается мониторинг контейнеров?

18. Какие существуют методы управления сетевыми настройками в Docker?
19. Как реализовать автоматическое масштабирование контейнеров?
20. Каковы основные тренды в области контейнеризации на сегодняшний день?

Примерные домашние задания

Домашнее задание “Docker Compose Advanced”.

Для выполнения домашнего задания по теме “Docker Compose Advanced” ты можешь выбрать один из двух языков программирования (Python/Java):

1. Docker Compose Advanced. Java
2. Docker Compose Advanced. Python

Что нужно сделать:

- Перейти по выбранной ссылке и выполнить задания
- Добавь SSH-ссылку на репозиторий с выполненным заданием в систему для отслеживания прогресса.

- Приложи эту же SSH-ссылку в меню akhcheck и отправь на проверку

Важно: Задание проходит автоматическую проверку, поэтому убедись, что код оформлен корректно и все тесты успешно выполняются.

Docker Compose Advanced. Java

Описание задания

В этом задании Вам предстоит собрать конфигурацию Docker Compose под приложение на Java.

Для этого Вам необходимо будет сделать скопировать в свой репозитории содержимое ветки `task-docker-compose` в репозитории.

Требования к сборке

- База Данных - MySQL, версия 5.7
- Версия Java - 8
- Файл `docker-compose.yml` должен быть в папке `docker-compose-template`.

Вам необходимо будет настроить подключение папки `CREATE.sql` из папки со скриптами во время создания контейнера.

Создайте `init container`, который будет заполнять данные в базу.

Подсказка: проект Веб собран при помощи Spring Boot.

Еще одна подсказка. Веб-приложение не стартует без базы данных. Вы должны найти способ создать базу данных перед запуском приложения.

Требования к запуску

- Приложение должно быть запущено на порту 8080
- Для заказов необходимо отображение:
- История заказов -> status **MOVING**
- Далее будет запущено приложение

Процедура сдачи

• Создайте Merge Request из ветки `task-docker-compose` в ветку `main` или в ветку `master`.

- Добавьте в ревьюеры проверяющего задание

Максимальный балл: 10

Docker Compose Advanced. Python

Описание задания

Преамбула

Коллега на работе разрабатывал проект на работе. Однако, он уволился, и компания перешла в контейнеризированную среду.

Ваша цель - запустить проект. Вам даны права Reporter, поэтому вам необходимо ветку `task-docker-compose` в Ваш репозиторий.

Что от вас ждет начальство?

- (2 балла) Docker-образ приложения `backend` и запуск контейнера с `backend` через `docker compose` (2 балла)
- (1/3 балла) Запуск в `docker compose` Docker Container-а с базой данных. Если приложение `backend` будет запускаться без внешней базы (можно подключить SQLite), то будет добавлен 1 балл. Если будет запускаться приложение с базой данных Postgres, то будет ставиться 3 балла за эту часть задания. **Важно:** данные в базе данных должны храниться постоянно (в отдельном Volume-е)
- (3 балла) Запуск сервиса `frontend` с использованием конфигурации NGINX. Из внешней сети контейнера сервис должен быть доступен на порту 8189.
 - (2 балла) ставится за успешный запрос к API через сервис `frontend`
 - (1 балл) ставится за успешный проброс стилевых файлов (этим должен заниматься Frontend). При успешном подключении вы увидите большие буквы по одному из URL.
- (3 балла) Непоседливый разработчик забыл добавить тестовые данные в приложение. Для этого необходимо создать `init container`, который перед запуском сервисов `backend` и `frontend` будет заполнять базу данных 2 тестовыми пользователями:

```
users = [  
    User(login='pavel', email='a@gmail.com'),  
    User(login='yura', email='b@gmail.com')  
]
```

 - Инициализация базы данных должна стартовать перед запуском `backend`
 - Frontend должен стартовать после успешного запуска сервиса `backend` (в ином случае, `nginx` не запустится)

Процедура сдачи задания

- Скопируйте содержимое ветки `task-docker-compose` репозитория в репозиторий, в котором выполняете задания курса.
- В ветке `task-docker-compose` в папке `docker-compose-template` создайте файл `docker-compose.yml`, в котором составьте корректную конфигурацию проектов.
- Добавьте в репозиторий необходимые файлы для запуска проекта (например, нужны будут Dockerfile-ы для образов, которые собираетесь использовать для сборки проекта)
- Перед тем, как отправлять задание на проверку, убедитесь, что проект запускается.

Шаги по запуску проекта:

```
docker compose down -v  
docker compose up
```

Если нет команды `docker compose`, то можно заменить команду на `docker-compose`. Если у преподавателей не запустится проект, то, увы, задание не будет зачтено. Важно сделать запуск "под ключ".

Электронный документ

Полезные ссылки

- <https://fastapi.tiangolo.com/ru/deployment/manually/> - запуск приложения FastAPI.

Максимальный балл: 10

Домашнее задание: Практика по Ansible

Что нужно сделать:

- Добавьте SSH-ссылку на репозиторий с выполненным заданием в систему для отслеживания прогресса.
- Приложите эту же SSH-ссылку в меню akhcheck и отправьте на проверку.
- **Важно:** Задание проходит автоматическую проверку, поэтому убедитесь, что код оформлен корректно и все тесты успешно выполняются.

Описание задания

В этом задании вам необходимо продемонстрировать умение настраивать тестовый стенд.

Шаги выполнения

1. Используйте репозиторий для одного из прошлых заданий
2. Добавьте в задании по CI дополнительный шаг **deliver**, который
3. (Ручная проверка, 3 балла) Настройте Pipeline для сборки образа и Push-a в Registry. Образ в Registry должен иметь tag **staging**. Название Stage - **BuildImage**, название Job - **BuildDebugImage**
4. (Ручная проверка) (3 балла) Создайте ветку **develop**. В ветке **develop** настройте Pipeline в Ansible, который будет выкачивать созданный репозиторий на сервер `node03.hadoop.akhcheck.ru` (доступ был выдан в ходе запуска курса) в ветку **develop**. Репозиторий должен называться `DevopsExam` и находиться по пути: `/home/<your user>/Pipeline`, в репозитории будет выкачана ветка **develop**. Проверяющий будет проверять, что репозиторий создан и в репозитории выбрана конкретная ветка.
5. (Ручная проверка) (4 балла) Настройте установку необходимого окружения:
 - для заданий по Python - настройте виртуальное окружение в Ansible и создание артефактов тестирования для тестовых стендов
 - для заданий по Java - настройте выкачивание Docker-образа для Maven и запуск сервиса.

Максимальный балл: 10

Домашнее задание. Kubernetes - Basics.

Что нужно сделать:

- Добавьте SSH-ссылку на репозиторий с выполненным заданием в систему для отслеживания прогресса.
- Приложите эту же SSH-ссылку в меню akhcheck и отправьте на проверку.
- **Важно:** Задание проходит автоматическую проверку, поэтому убедитесь, что код оформлен корректно и все тесты успешно выполняются.

Описание задания

Создайте настройку для Kubernetes-кластера. В качестве примера можно взять настроенное приложение с `docker compose` с предыдущих заданий

Что должно включаться в сборку:

1. Pod-ы для запуска приложения (3 балла).
2. Настройка сервиса с использованием кластера ClusterIP (3 балла)
3. Конфигурация сервера при помощи Ingress. При этом доступ должен быть как к базе данных, как к Frontend, так и в Backend (4 балла)

Если у вас есть собственный проект, который вы хотели поднять с помощью Kubernetes, то можно использовать его!

Формат сдачи задания

Необходимо прислать ссылку на репозиторий. В репозитории должен детально описан порядок сборки манифестов. Без порядка сборки манифестов задание не будет засчитано!

Максимальный балл: 10

Домашнее задание: Kubernetes - Продвинутое

Что нужно сделать:

- Добавьте SSH-ссылку на репозиторий с выполненным заданием в систему для отслеживания прогресса.

- Приложи эту же SSH-ссылку в меню akhcheck и отправь на проверку.

Важно: Задание проходит автоматическую проверку, поэтому убедись, что код оформлен корректно и все тесты успешно выполняются.

Описание задания

Создайте настройку для своего проекта в Kubernetes-кластере. Продолжаем выполнять задание:

Что должно включаться в сборку:

1. Настройка секретов для Docker Registry и для Basic Auth (1+2 балла).
2. Настройка Job для миграции или CronJob для ежедневных задач (3 балла)
3. Переформатирование базы данных в StatefulSet (2 балла)
4. Запуск базы данных в режиме High Availability (4 балла)

Если у вас есть собственный проект, который вы хотели поднять с помощью Kubernetes, то можно использовать его!

Формат сдачи задания

Необходимо прислать ссылку на репозиторий. В репозитории должен детально описан порядок сборки манифестов. Без порядка сборки манифестов задание не будет засчитано!

Примерные задания для контрольной работы

1. **Сравнительный анализ.** Опишите основные отличия между оркестраторами контейнеров и системами контейнеризации. Приведите примеры каждого типа и объясните, как они взаимодействуют друг с другом.

2. **Основные понятия.** Объясните ключевые термины, связанные с Kubernetes, такие как Pod, Node, Cluster, и Namespace. Какова их роль в управлении контейнерами?

3. **Разворачивание кластера.** Приведите пошаговую инструкцию по разворачиванию кластера Kubernetes с использованием MiniKube. Укажите необходимые команды и настройки.

4. **Фундамент Kubernetes.** Объясните роль containerd и etcd в архитектуре Kubernetes. Как эти компоненты способствуют управлению контейнерами и состоянием кластера?

5. **Абстракции для развертывания сервисов.** Опишите разницу между Pod и Service в Kubernetes. Как используются Ingress и какие преимущества он предоставляет при развертывании приложений?

6. **Отказоустойчивость.** Объясните, как ReplicaSet и Deployment обеспечивают отказоустойчивость в Kubernetes. Приведите примеры их использования в реальных сценариях.

7. **Сохранение состояния.** Что такое PersistentVolumeClaim и как он используется для управления состоянием в Kubernetes? Опишите также, как StatefulSet и Volumes обеспечивают хранение данных.


8. **Хранение секретов.** Опишите процесс хранения секретов в Kubernetes-кластере. Как можно подключить внешние хранилища секретов, такие как HashiCorp Vault?

9. **Управляющие ресурсы.** Объясните, что такое Job, CronJob и DaemonSet в Kubernetes. В каких случаях их следует использовать и какие задачи они решают?

10. **Управление безопасностью.** Как реализуется управление безопасностью в Kubernetes с помощью RBAC, ServiceAccount и Network Policy? Приведите примеры их настройки и использования.

Задания для промежуточной аттестации по дисциплине (модулю)

№ п/п	Задание	Ответ	Компетенция
1.	Осуществляя управление проектов на стадии оценки назовите концепцию разработки, которая соединяет разработчиков и специалистов операционного обслуживания. Название напишите английскими буквами.	Devops/ devops	УК-2
2.	При планировании проекта определите методологию разработки, которую можно использовать, если точно известны все сроки реализации проекта от начала и до конца, процессы строго регламентированы и рутинны. Ответ напишите в виде одного слова на русском или английском языке.	waterfall/водопад/водопадная / Водопад / Водопадная / Waterfall	УК-2
3.	При планировании следующей итерации проекта назовите мероприятие в методологии Scrum, в которой проводится рефлексия относительно завершенного спринта. Ответ напишите в виде одного слова на русском или английском языке.	Ретроспектив a/retrospective /ретроспектив a/ Retrospective	УК-2
4.	Представьте, что вы являетесь руководителем проекта. Вам необходимо провести обучение сотрудников по теме Q&A. Для этого отсортируйте уровни тестирования ПО по степени уменьшения автоматизированности 1) Компонентное/Unit 2) Приёмочное 3) Интеграционное 4) Системное В ответ запишите последовательность цифр, пример: 1234	3421	УК-2
5.	На этапе планирования вам необходимо вам необходимо выбрать методологии, в которых количество итераций фиксировано. Выберите все ответы: 1) Kanban 2) Инкрементная 3) Scrum	24/42	УК-2

	4) Итеративная		
6.	Назовите вид абстракции в Kubernetes, которая гарантирует наличие Pod-ов в фиксированном количестве копий.	ReplicaSet/replicaset/replicaset	ПК-3
7.	Выберите систему управления конфигурациями, в которой обязательно наличие Агента для хранения конфигурации: 1) SaltStack 2) Chef 3) Ansible 4) Puppet	3	ПК-3
8.	Пользователь Петя в Terraform настроил 10 серверов. Теперь Петя указал, что ему необходимо настроить 40 серверов (поменял конфигурацию с 10 серверов на 40 серверов). При этом конфигурация сервера была полностью изменена. Какое количество серверов будет создано?	40	ОПК-2
9.	Укажите количество Job в следующем Pipeline-е Gitlab CI: 	9	ПК-3
10.	Укажите вид абстракции, который позволяет хранить секретные данные в Kubenertes.	secret/Secret	ПК-3
11.	Укажите тип метрики в Prometheus, которая позволяет отслеживать неубывающий тип метрики (запрещено уменьшать значение метрики).	Counter/counter/count	ПК-6
12.	Укажите количество Runner-ов, на которых может запускаться Job с тегами <code>cpu</code> и <code>dind</code> , если в распоряжении имеются следующие Runner-ы: 1. Worker Dind с тегами <code>common</code> , <code>cpu</code> и <code>dind</code> 2. GPU-worker с тегами <code>gpu</code> и <code>dind</code> 3. Worker с тегами <code>common</code> и <code>cpu</code> 4. ARM worker с тегами <code>cpu</code> , <code>arm64</code> 5. Deploy Worker с тегами <code>common</code> , <code>cpu</code>	1	ПК-6
13.	Назовите свойство системы управления конфигурацией, которое указывает на неизменность результата применения операции после определенного количества повторений.	идемпотентность/Идемпотентность/idempotency/Idempotency	ОПК-2
14.	Конфигурация метрики Graphite имеет следующую конфигурацию retention: <code>60s:1d,5m:30d,1h:3y</code> Укажите количество точек, которое хранится в метрике. Считаем, что в 1 календарном году 365 дней.	36360	ОПК-2
15.	Перед вами следующий Dockerfile. Вычислите размер полученного образа. FROM ubuntu	92	ОПК-2

	<p>RUN apt-get update && apt-get install -y nginx RUN rm -rf /var/lib/apt/lists/* Образ ubuntu занимает 30 МБ, запуск команды apt-get update - добавляет 42 МБ. Команда apt-get install -y nginx - 20 МБ. Команда rm -rf /var/lib/apt/lists/* освобождает 25 МБ. Ответ укажите в МБ.</p>		
16.	Укажите тип отчета в Gitlab CI, который выгружает результаты тестирования.	junit/JUnit/junit-xml/JUnit-xml	ОПК-2
17.	Укажите полное название образа ubuntu:24.04 при отправке образа в Docker Registry.	docker.io/library/ubuntu:24.04	ПК-6
18.	Женя написал в своем Dockerfile строку "FROM CU-Exam". У образа CU-Exam есть теги v0.1 и v0.3 и v0.5. Версия v0.3 также имеет тег latest. Вопрос: какой образ будет скачан. Укажите номер версии.	v0.5/CU-Exam:v0.5	ПК-6