Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Максимов Алексей Борисович
Должность: директор департамента по образовательной политике
Дата подписания: 16.07.2025 15:47:15
Уникальный программный ключ:
8db180d1a3f02ac9e60521a5672742735c18b1d6

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение
высшего образования
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
Факультет «Информационные технологии»

УТВЕРЖДЕНО
Декан факультета
Информационных технологий
_____/ Д.Г. Демидов /
«_27_»_____февраля_____2025 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

«Процессы DevOps

Направление подготовки
09.03.03 Прикладная информатика

Профиль
« Корпоративные решения на платформе 1С»

Квалификация
Бакалавр

Формы обучения
очная

Москва, 2025 г.

Разработчик(и):

к.э.н., доцент  / С.В.Суворов /

Согласовано:

Заведующий кафедрой «Прикладная информатика»,
к.э.н., доцент  / С.В.Суворов/

Содержание

1. Цели, задачи и планируемые результаты обучения по дисциплине 4
2. Место дисциплины в структуре образовательной программы 4
3. Структура и содержание дисциплины 5
3.1. Виды учебной работы и трудоемкость (по формам обучения) 5
3.2. Тематический план изучения дисциплины (по формам обучения) 5
3.3. Содержание дисциплины 6
3.4. Тематика семинарских/практических и лабораторных занятий 8
3.5. Тематика курсовых проектов (курсовых работ) 9
4. Учебно-методическое и информационное обеспечение 10
4.1. Нормативные документы и ГОСТы 10
4.2. Основная литература 10
4.3. Дополнительная литература 10
4.4. Электронные образовательные ресурсы 10
4.5. Лицензионное и свободно распространяемое программное обеспечение 10
4.6. Современные профессиональные базы данных и информационные справочные системы 10
5. Материально-техническое обеспечение 10
6. Методические рекомендации 11
6.1. Методические рекомендации для преподавателя по организации обучения 11
6.2. Методические указания для обучающихся по освоению дисциплины 11
7. Фонд оценочных средств 11
7.1. Методы контроля и оценивания результатов обучения 12
7.2. Шкала и критерии оценивания результатов обучения 12
7.3. Оценочные средства 13

1. Цели, задачи и планируемые результаты обучения по дисциплине

Целью дисциплины “Процессы DevOps” является обучение студентов принципам и практикам DevOps, включая непрерывную интеграцию, непрерывную доставку, автоматизацию процессов, инфраструктуру как код и мониторинг в реальном времени. Студенты изучают основы работы с инструментами DevOps, такими как Docker и Kubernetes, а также учатся принципам работы с облачными сервисами. Практическая направленность дисциплины заключается в том, чтобы студенты могли применять полученные знания в реальных проектах по разработке и поддержке программного обеспечения, используя методологию DevOps.

К основным **задачам** освоения дисциплины следует отнести:

- Изучение принципов и практик DevOps, включая непрерывную интеграцию, непрерывную доставку, автоматизацию процессов, инфраструктуру как код и мониторинг в реальном времени.
- Изучение и практическое применение Docker и Kubernetes, включая создание Docker образов, работу с Docker Compose, сетями и томами в Docker, использование Docker Hub и Docker Swarm, а также развертывание приложений и управление ресурсами в Kubernetes.
- Изучение и практическое применение инструментов для автоматизации с помощью Python и CI/CD, включая работу с файлами, директориями, сетью, базами данных и веб-фреймворками, а также непрерывную интеграцию, непрерывную доставку и развертывание, автоматизированное тестирование, мониторинг и логирование.
- Изучение и практическое применение GitLab и Gitea для DevOps, включая управление репозиториями кода, настройку CI/CD и интеграцию с другими инструментами DevOps.
- Изучение принципов работы с облачными технологиями и оркестраторами, такими как Yandex Cloud, AWS, Google Cloud и Azure, включая использование облачных сервисов для автоматизации DevOps и обеспечения безопасности, а также управление ресурсами, обновление приложений, мониторинг и логирование, управление доступом и работу с хранилищами данных в оркестраторах.

Обучение по дисциплине «Методология DevOps» направлено на формирование у обучающихся следующих компетенций:

Код и наименование компетенций	Индикаторы достижения компетенции
ПК-6. Способен предотвращать потери и повреждения данных	ИПК-6.1. Знает способы и методы резервного копирования и восстановления данных в проектах игровой компьютерной индустрии
	ИПК-6.2. Умеет выявлять проблемные ситуации; производить резервное копирование и восстановление данных в проектах игровой компьютерной индустрии
	ИПК-6.3. Имеет навыки разработки и применения программного обеспечения для резервного копирования и восстановления данных в продуктах игровой компьютерной индустрии

2. Место дисциплины в структуре образовательной программы

Дисциплина относится к модулю «Проектирование и разработка информационных систем» обязательной части Блока 1. Часть, формируемая участниками образовательных отношенийучебного плана программы бакалавриата.

Основные положения дисциплины должны быть использованы в дальнейшем при изучении следующих дисциплин:

- Тестирование программного обеспечения;
- Информационная безопасность и защита информации;
- Технологии распространения игрового контента;
- Управление программными проектами;
- Учебная практика (проектная);
- Программирование для мобильных устройств;
- Производственная практика (проектно-технологическая);
- Производственная практика (преддипломная);
- Выполнение и защита выпускной квалификационной работы.

3. Структура и содержание дисциплины

Общая трудоемкость дисциплины составляет 3 зачетные единицы (108 часа).

3.1. Виды учебной работы и трудоемкость (по формам обучения)

3.1.1. Очная форма обучения

№ п/п	Вид учебной работы	Количество часов	Семестры
			4
1	**Аудиторные занятия**	**54**	**54**
	В том числе:		
1.1	Лекции	18	18
1.2	Семинарские/практические занятия		
1.3	Лабораторные занятия	36	36
2	**Самостоятельная работа**	**54**	**54**
	В том числе:		
2.1	Подготовка и выполнение лабораторных работ	54	54
3	**Промежуточная аттестация**		
	Зачет/диф.зачет/экзамен	экзамен	
	Итого:	**108**	**108**

3.2. Тематический план изучения дисциплины (по формам обучения)

3.2.1. Очная форма обучения

№ п/п	Разделы/темы дисциплины	Трудоемкость, час					
		Всего	Аудиторная работа				Самостоятельная работа
			Лекции	Лабораторные	Практические	Практическая подготовка	
1.1	Тема 1. «Введение в DevOps и Docker»	2	1				2
1.2	Лабораторная работа 1. «Введение в DevOps и Docker»	3		3			2
2.1	Тема 2. «Работа с Docker»	3	2				2

2.2	Лабораторная работа 2. «Работа с Docker»	4		2			2
3.1	Тема 3. «Введение в Kubernetes (k8s)»	3	2				2
3.2	Лабораторная работа 3. «Введение в Kubernetes (k8s)»	4		2			2
4.1	Тема 4. «Работа с Kubernetes»	2	1				2
4.2	Лабораторная работа 4. «Работа с Kubernetes»	4		2			2
5.1	Тема 5. «Автоматизация с помощью Python»	2	1				2
5.2	Лабораторная работа 5. «Автоматизация с помощью Python»	4		2			2
6.1	Тема 6. «Введение в CI/CD»	4	2				2
6.2	Лабораторная работа 6. «Введение в CI/CD»	4		2			2
7.1	Тема 7. «GitLab, Gitea и DevOps»	4	2				2
7.2	Лабораторная работа 7. «GitLab, Gitea и DevOps»	3		3			2
8.1	Тема 8. «Облачные технологии (Yandex Cloud, AWS, Google Cloud, Azure)»	3	1				2
8.2	Лабораторная работа 8. «Облачные технологии (Yandex Cloud, AWS, Google Cloud, Azure)»	3		2			2
9.1	Тема 9. «Оркестраторы»	4	2				2
9.2	Лабораторная работа 9. «Оркестраторы»	3		2			2
10.1	Тема 10. «Мониторинг с Zabbix»	4	2				2
10.2	Лабораторная работа 10. «Мониторинг с Zabbix»	3		2			2
11.1	Тема 11. «Task трекеры (Jira, Trello, Asana, OpenProject)»	4	2				2
11.2	Лабораторная работа 11. «Task трекеры (Jira, Trello, Asana, OpenProject)»	2		2			2
Итого		**108**	**18**	**36**			**54**

3.3. Содержание дисциплины

1. **Введение в DevOps и Docker**
   - История и эволюция DevOps.
   - Основные принципы и практики DevOps.
   - Введение в контейнеризацию и Docker.
   - Основные понятия Docker: образы, контейнеры, тома, сети.
   - Преимущества использования Docker.
   - Работа с Docker CLI.
   - Docker Hub и Docker Registry.
2. **Работа с Docker**
   - Создание Docker образов: Dockerfile, слои, инструкции.
   - Docker Compose: управление многоконтейнерными приложениями, синтаксис файла docker-compose.yml.

- Сети в Docker: bridge, host, none, overlay.
- Тома в Docker: использование для постоянного хранения данных.
- Docker Hub: использование для хранения и распространения Docker образов.
- Docker Swarm: введение в оркестрацию Docker.

3. **Введение в Kubernetes (k8s)**
   - Обзор Kubernetes: архитектура, основные компоненты (поды, сервисы, ингрессы).
   - Развертывание приложений в Kubernetes: Deployment, ReplicaSet, StatefulSet, DaemonSet.
   - Сервисы в Kubernetes: ClusterIP, NodePort, LoadBalancer, ExternalName.
   - Ингрессы в Kubernetes: использование для маршрутизации трафика в кластере.
   - Конфигурация приложений в Kubernetes: ConfigMap, Secret.
   - Управление ресурсами в Kubernetes: ResourceQuota, LimitRange.
4. **Работа с Kubernetes**
   - Управление ресурсами в Kubernetes: ResourceQuota, LimitRange.
   - Обновление приложений в Kubernetes: RollingUpdate, Recreate.
   - Мониторинг и логирование в Kubernetes: встроенные инструменты, сторонние решения.
   - Управление доступом в Kubernetes: RBAC, ServiceAccount.
   - Работа с хранилищами данных в Kubernetes: Volume, PersistentVolume, PersistentVolumeClaim.
5. **Автоматизация с помощью Python**
   - Использование Python для автоматизации: обзор основных библиотек и инструментов.
   - Автоматизация задач DevOps с помощью Python: написание скриптов для управления инфраструктурой, автоматизация тестирования, автоматизация развертывания.
   - Работа с файлами и директориями в Python.
   - Работа с сетью в Python: запросы HTTP, работа с API.
   - Работа с базами данных в Python: SQLite, MySQL, PostgreSQL.
   - Тестирование кода на Python: unittest, pytest.
   - Работа с веб-фреймворками в Python: Flask, Django.
6. **Введение в CI/CD**
   - Обзор CI/CD: что такое CI/CD, зачем оно нужно, обзор основных инструментов CI/CD.
   - Непрерывная интеграция: принципы, практики, инструменты.
   - Непрерывная доставка и непрерывное развертывание: различия, принципы, практики, инструменты.
   - Автоматизированное тестирование в CI/CD: unit-тесты, интеграционные тесты, E2E-тесты.
   - Мониторинг и логирование в CI/CD: важность, инструменты.
7. **GitLab, Gitea и DevOps**
   - Обзор GitLab и Gitea: установка и настройка, создание групп и пользователей, настройка CI/CD.
   - Возможности GitLab для DevOps: управление репозиториями кода, CI/CD, мониторинг.
   - Работа с Git в GitLab и Gitea: основные операции, ветвление, слияние.
   - Настройка CI/CD в GitLab и Gitea: создание пайплайнов, использование раннеров.
   - Интеграция GitLab и Gitea с другими инструментами DevOps.
8. **Облачные технологии (Yandex Cloud, AWS, Google Cloud, Azure)**

- Обзор облачных технологий: что такое облачные технологии, как работать с облачными сервисами Yandex Cloud, AWS, Google Cloud и Azure.
- Введение в Yandex Cloud: основные возможности, управление ресурсами.
- Облачные сервисы для разработки: вычислительные ресурсы, базы данных, хранилища данных, сетевые сервисы, сервисы для разработчиков.
- Использование облачных сервисов для автоматизации DevOps: автоматическое масштабирование, автоматическое развертывание, CI/CD, мониторинг.
- Безопасность в облачных технологиях: основные принципы, практики.

9. **Оркестраторы**
   - Обзор оркестраторов: что такое оркестраторы, зачем они нужны, примеры использования.
   - Работа с оркестраторами: установка и настройка, создание и управление ресурсами.
   - Интеграция оркестраторов с другими инструментами DevOps.
   - Мониторинг и логирование в оркестраторах: встроенные инструменты, сторонние решения.
   - Безопасность в оркестраторах: основные принципы, практики.
10. **Мониторинг с Zabbix**

    - Обзор Zabbix: что такое Zabbix, как его настроить и использовать для мониторинга.
    - Установка и настройка Zabbix: сервер, агенты, веб-интерфейс.
    - Создание элементов данных, триггеров, графиков и дашбордов в Zabbix.
    - Интеграция Zabbix с другими инструментами DevOps.
    - Расширение возможностей Zabbix: использование шаблонов, скриптов, плагинов.

11. **Task трекеры (Jira, Trello, Asana, OpenProject)**

    - Обзор task трекеров: что такое task трекеры, зачем они нужны, примеры использования.
    - Обзор коммерческих и open-source решений: Jira, Trello, Asana, OpenProject.
    - Установка и настройка task трекера: создание проектов, задач, пользователей.
    - Работа с task трекерами: создание и изменение задач, работа с комментариями, прикрепление файлов.
    - Интеграция task трекеров с другими инструментами DevOps.
    - Расширение возможностей task трекеров: использование плагинов, API.

3.4. Тематика семинарских/практических и лабораторных занятий

3.4.1. Лабораторные занятия

Не предусмотрены.

3.4.2. Семинарские/практические занятия

Практическая работа 1. «Основы DevOps и Docker»

В рамках данной работы студенты осваивают процесс установки Docker на локальную машину или виртуальную машину, изучают основные команды Docker CLI, осуществляют создание и запуск Docker контейнеров на основе существующих образов, а также создают собственные Docker образы и загружают их в Docker Hub.

Практическая работа 2. «Работа с Docker»

В ходе выполнения данной работы студенты создают Dockerfile для сборки собственных Docker образов, используют Docker Compose для управления многоконтейнерными приложениями, настраивают сети и тома в Docker для обеспечения связи между контейнерами и сохранения данных.

Практическая работа 3. «Введение в Kubernetes (k8s)»

В рамках данной работы студенты осуществляют установку и настройку среды Kubernetes с использованием Minikube или другого локального решения, создают и управляют подами и сервисами в Kubernetes, работают с ингрессами для маршрутизации трафика, создают конфигурации для управления настройками приложений.

Практическая работа 4. «Работа с Kubernetes»

В ходе выполнения данной работы студенты осуществляют управление ресурсами в Kubernetes с использованием ResourceQuota и LimitRange, обновляют приложения с использованием стратегий RollingUpdate и Recreate, настраивают мониторинг и логирование в Kubernetes, управляют доступом с использованием RBAC и ServiceAccount, работают с хранилищами данных в Kubernetes.

Практическая работа 5. «Автоматизация с помощью Python»

В рамках данной работы студенты пишут скрипты на Python для автоматизации различных задач DevOps, таких как управление файлами и директориями, отправка HTTP-запросов, работа с API, работа с базами данных, тестируют код с использованием модулей unittest или pytest.

Практическая работа 6. «Введение в CI/CD»

В ходе выполнения данной работы студенты осуществляют настройку CI/CD с использованием выбранного инструмента, такого как Jenkins, GitLab CI или другого, создают пайплайны CI/CD для автоматической сборки, тестирования и развертывания приложений, настраивают мониторинг и логирование в CI/CD.

Практическая работа 7. «GitLab, Gitea и DevOps»

В рамках данной работы студенты осуществляют установку и настройку GitLab или Gitea, работают с Git для управления версиями кода, настраивают CI/CD в GitLab или Gitea, интегрируются с другими инструментами DevOps.

Практическая работа 8. «Облачные технологии (Yandex Cloud, AWS, Google Cloud, Azure)»

В ходе выполнения данной работы студенты работают с облачными сервисами, включая настройку и использование Yandex Cloud, AWS, Google Cloud и Azure, используют облачные сервисы для автоматизации DevOps, включая автоматическое масштабирование, автоматическое развертывание, CI/CD и мониторинг.

Практическая работа 9. «Оркестраторы»

В рамках данной работы студенты осуществляют установку и настройку оркестратора, такого как Kubernetes, Docker Swarm или другой, создают и управляют ресурсами, интегрируют оркестратор с другими инструментами DevOps.

Практическая работа 10. «Мониторинг с Zabbix»

В ходе выполнения данной работы студенты осуществляют установку и настройку Zabbix, создают элементы данных, триггеры, графики и дашборды в Zabbix, интегрируют Zabbix с другими инструментами DevOps.

Практическая работа 11. «Task трекеры (Jira, Trello, Asana, OpenProject)»

В рамках данной работы студенты осуществляют установку и настройку task трекера, такого как Jira, Trello, Asana или OpenProject, создают проекты и задачи, работают с комментариями и прикрепленными файлами, интегрируют task трекер с другими инструментами DevOps.

3.5. Тематика курсовых проектов (курсовых работ)

Курсовые проекты не предусмотрены.

4. Учебно-методическое и информационное обеспечение

4.1. Нормативные документы и ГОСТы

1. Федеральный закон от 29 декабря 2012 года No 273-ФЗ «Об образовании в Российской Федерации» (с изменениями и дополнениями);
2. Федеральный государственный образовательный стандарт высшего образования - бака-лавриат по направлению подготовки 09.03.02 Информационные системы и технологии, утвер-жденный Приказом Министерства образования и науки РФ от 19 сентября 2017 г. N 929 "Об утверждении федерального... Редакция с изменениями N 1456 от 26.11.2020;
3. Приказ Министерства образования и науки РФ от 05 апреля 2017 г. No 301 «Об утвер-ждении Порядка организации и осуществления образовательной деятельности по образователь-ным программам высшего образования – программам бакалавриата, программам специалитета, программам магистратуры.

4.2. Основная литература

1. ""Философия DevOps. Искусство управления IT" - Дженнифер Дэвис - Питер - 2016 - 416 страниц - ISBN: 978-5-496-02555-3
2. "Docker на практике" - Эйдан Хобсон Сейерс, Иан Милл - ДМК Пресс - 2019 - 516 страниц - ISBN: 978-5-97060-772-5
3. "Continuous delivery. Практика непрерывных апдейтов" - Эберхард Вольф - Питер - 2017 - 320 страниц - ISBN: 978-5-4461-0480-2
4. "Введение в технологии контейнеров и Kubernetes" - Маркелов А. - ДМК Пресс - 2019 - 194 страниц - ISBN: 978-5-97060-775-6
5. Автоматизация рутинных задач с помощью Python" - Свейгарт Эл - Издательство не указано - 2021

4.3. Дополнительная литература

1. "Kubernetes Patterns" - Bilgin Ibryam, Roland Huss - O'Reilly Media, Inc. - 2023 - ISBN: 9781098131685

4.4. Электронные образовательные ресурсы

ЭОР не предусмотрен.

4.5. Лицензионное и свободно распространяемое программное обеспечение

1. Microsoft Visual Studio: интегрированная среда разработки (IDE), которая поддерживает различные языки программирования и позволяет разрабатывать кроссплатформенные приложения.
2. Visual Studio Code: легковесный и расширяемый текстовый редактор, позволяющий разрабатывать кроссплатформенные приложения на различных языках программирования.
3. VirtualBox: это мощный и гибкий инструмент виртуализации, который позволяет пользователям запускать несколько операционных систем на одном компьютере, обеспечивая кросс-платформенную совместимость и эффективное использование ресурсов.

4.6. Современные профессиональные базы данных и информационные справочные системы

Не предусмотрены.

5. Материально-техническое обеспечение

Методика преподавания дисциплины «Методология DevOps» предусматривает использование онлайн-курса в системе дистанционного обучения Университета, групповых и индивидуальных консультаций обучающихся, аудиторных занятий в сочетании с внеаудиторной работой с целью формирования и развития профессиональных навыков обучающихся.

Практические работы по дисциплине «Методология DevOps» осуществляются в форме самостоятельной проработки теоретического материала обучающимися; выполнения практического задания; защиты преподавателю лабораторной работы (знание теоретического материала и выполнение практического задания по теме лабораторной работы).

6. Методические рекомендации

6.1. Методические рекомендации для преподавателя по организации обучения

Методика преподавания дисциплины «Методология DevOps» предусматривает использование онлайн-курсов в системе дистанционного обучения, проведение групповых и индивидуальных консультаций, а также аудиторных занятий в сочетании с внеаудиторной работой для формирования и развития профессиональных навыков студентов.

Практические занятия по дисциплине «Методология DevOps» включают самостоятельную проработку теоретического материала, выполнение практического задания и защиту лабораторной работы перед преподавателем, включая проверку знания теоретического материала и успешное выполнение задания по теме работы.

6.2. Методические указания для обучающихся по освоению дисциплины

Изучение дисциплины осуществляется в соответствии с учебным планом.

На занятиях осуществляется закрепление полученных, в том числе и в процессе самостоятельной работы, знаний. Особое внимание обращается на умение применять полученные знания на практике, в том числе при решении реальных задач, отличающихся от проработанных.

В процессе самостоятельной работы студенты закрепляют и углубляют знания, полученные во время аудиторных занятий, самостоятельно знакомятся с теоретическим материалом, выполняют лабораторные работы, готовятся к текущему контролю и промежуточной аттестации.

Текущий контроль осуществляется на аудиторных занятиях в виде защиты лабораторных работ. Критериями оценки результатов являются:

- уровень освоения теоретического материала;
- уровень владения практическими навыками (в виде вопросов по процессу выполнения лабораторных работ);
- умения обучающегося использовать теоретические знания при выполнении практических задач (в виде дополнительных заданий);
- сформированность компетенций;
- оформление материала в соответствии с требованиями.

Промежуточный контроль осуществляется на экзамене в форме тестирования в системе дистанционного обучения Университета, включающего вопросы на знание практической части.

7. Фонд оценочных средств

7.1. Методы контроля и оценивания результатов обучения

В процессе обучения используются следующие оценочные формы самостоятельной работы студентов, оценочные средства текущего контроля успеваемости и промежуточных аттестаций: **экзамен**.

7.2. Шкала и критерии оценивания результатов обучения

К промежуточной аттестации допускаются только студенты, выполнившие все виды учебной работы, предусмотренные рабочей программой по дисциплине «Методология DevOps».

7.2.1. Критерии оценки ответа на экзамене

(формирование компетенций — ПК-6)

«Отлично»:

Выполнены все виды учебной работы, предусмотренные учебным планом. Обучающийся выполнил и защитил практические работы по офисным приложениям со средним баллом от 4,5 до 5. Итоговое тестирование выполнено на 85 — 100%. Обучающийся демонстрирует прочные теоретические знания, практические навыки, владеет терминами, делает аргументированные выводы и обобщения, приводит примеры, оперирует приобретенными знаниями, умениями, навыками, применяет их в ситуациях повышенной сложности. При этом могут быть допущены незначительные ошибки, неточности, которые обучающийся может исправить самостоятельно.

«Хорошо»:

Выполнены все виды учебной работы, предусмотренные учебным планом. Обучающийся выполнил и защитил практические работы по офисным приложениям со средним баллом от 4 до 4,5. Итоговое тестирование выполнено на 70 — 84%. Обучающийся демонстрирует достаточные теоретические знания, практические навыки, владеет терминами, делает аргументированные выводы и обобщения, приводит примеры, оперирует приобретенными знаниями, умениями, навыками. При этом могут быть допущены незначительные ошибки, неточности, затруднения при аналитических операциях, переносе знаний и умений на новые, нестандартные ситуации, которые обучающийся может исправить при незначительной коррекции преподавателем.

«Удовлетворительно»:

Выполнены все виды учебной работы, предусмотренные учебным планом. Обучающийся выполнил и защитил практические работы по офисным приложениям со средним баллом ниже 4. Итоговое тестирование выполнено на 55 — 69%. Обучающийся демонстрирует неполное соответствие теоретических знаний, практических навыков, владеет терминами, делает аргументированные выводы и обобщения, приводит примеры, оперирует приобретенными знаниями, умениями, навыками. При этом могут быть допущены ошибки, неточности, затруднения при аналитических операциях, переносе знаний и умений на новые, нестандартные ситуации, которые обучающийся может исправить при коррекции преподавателем.

«Неудовлетворительно»:

Не выполнен один или более видов учебной работы, предусмотренных учебным планом. Обучающийся не выполнил одно или более заданий текущего и промежуточного контроля. Итоговое тестирование выполнено на 0 — 54%. Обучающийся демонстрирует незнание теоретических основ предмета, отсутствие практических навыков, не умеет делать аргументированные выводы и приводить примеры, не владеет терминами, проявляет отсутствие логичности и последовательности изложения, делает ошибки, которые не может исправить даже при коррекции преподавателем, отказывается отвечать на дополнительные вопросы, допускает значительные ошибки, испытывает значительные затруднения при оперировании знаниями и умениями при их переносе на новые ситуации.

7.2.2. Критерии оценки работы обучающегося на лабораторных занятиях:

(формирование компетенций — ПК-6)

«5» (отлично): выполнены все практические задания, предусмотренные работами, обучающийся четко и без ошибок ответил на все контрольные вопросы, проявил творческий подход при выполнении заданий, смог выполнить дополнительные задания.

«4» (хорошо): выполнены все практические задания, предусмотренные работами, обучающийся с корректирующими замечаниями преподавателя ответил на все контрольные вопросы, проявил творческий подход при выполнении заданий, смог частично выполнить дополнительные задания.

«3» (удовлетворительно): выполнены все практические задания, предусмотренные работами, с замечаниями преподавателя; обучающийся ответил на все контрольные вопросы с замечаниями, дополнительные задания выполнены с замечаниями.

«2» (неудовлетворительно): обучающийся не выполнил или выполнил неправильно практические задания, предусмотренные работами, обучающийся ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы, дополнительные задания выполнены неверно или не выполнены.

7.3. Оценочные средства

7.3.1 Экзаменационные вопросы

1. **Вопрос по Docker**: Как Docker использует слои для создания образов? Какие преимущества это дает?
    - A) Docker использует слои для создания образов, чтобы упростить обновление и распределение образов. **(+)**
    - B) Docker использует слои для создания образов, чтобы увеличить безопасность контейнеров.
    - C) Docker использует слои для создания образов, чтобы увеличить скорость выполнения контейнеров.
    - D) Docker использует слои для создания образов, чтобы уменьшить размер контейнеров.
2. **Вопрос по Kubernetes**: Как Kubernetes использует сервисы для обеспечения сетевого доступа к подам? Объясните различия между ClusterIP, NodePort, LoadBalancer и ExternalName.
    - A) Kubernetes использует сервисы для обеспечения сетевого доступа к подам. ClusterIP - это внутренний IP-адрес, NodePort открывает порт на всех узлах, LoadBalancer использует внешний балансировщик нагрузки, а ExternalName предоставляет абстракцию для внешнего имени. **(+)**
    - B) Kubernetes использует сервисы для обеспечения сетевого доступа к подам. ClusterIP, NodePort, LoadBalancer и ExternalName - это различные типы подов в Kubernetes.
    - C) Kubernetes использует сервисы для обеспечения сетевого доступа к подам. ClusterIP, NodePort, LoadBalancer и ExternalName - это различные типы сетей в Kubernetes.
    - D) Kubernetes использует сервисы для обеспечения сетевого доступа к подам. ClusterIP, NodePort, LoadBalancer и ExternalName - это различные типы хранилищ данных в Kubernetes.
3. **Вопрос по Python для автоматизации**: Как вы бы использовали Python для автоматизации задач DevOps? Приведите пример скрипта Python, который вы бы использовали для управления инфраструктурой или автоматизации тестирования.
    - A) Python можно использовать для автоматизации задач DevOps, например, для написания скриптов для управления инфраструктурой или автоматизации тестирования. Например, можно написать скрипт Python,

который автоматически создает и удаляет виртуальные машины в облаке. **(+)**
    - B) Python можно использовать для автоматизации задач DevOps, например, для создания пользовательских интерфейсов для приложений.
    - C) Python можно использовать для автоматизации задач DevOps, например, для разработки мобильных приложений.
    - D) Python можно использовать для автоматизации задач DevOps, например, для анализа больших данных.
4. **Вопрос по CI/CD**: Что такое CI/CD и какие преимущества оно предоставляет командам разработки? Объясните разницу между непрерывной интеграцией, непрерывной доставкой и непрерывным развертыванием.
    - A) CI/CD - это методологии, которые позволяют командам разработки чаще и надежнее выпускать изменения кода. Непрерывная интеграция означает автоматическую сборку и тестирование кода, непрерывная доставка означает автоматическую сборку, тестирование и развертывание кода в стадии предпродажного тестирования, а непрерывное развертывание означает автоматическую сборку, тестирование и развертывание кода в продакшн. **(+)**
    - B) CI/CD - это языки программирования, используемые для разработки веб-приложений.
    - C) CI/CD - это базы данных, используемые для хранения информации о коде и его изменениях.
    - D) CI/CD - это облачные платформы, предоставляющие вычислительные ресурсы для разработки.
5. **Вопрос по облачным технологиям**: Что такое облачные технологии и какие преимущества они предоставляют? Как вы бы использовали облачные сервисы, такие как Yandex Cloud, AWS, Google Cloud или Azure, для автоматизации DevOps?
    - A) Облачные технологии - это услуги, которые предоставляют вычислительные ресурсы и хранилище данных через интернет. Они могут быть использованы для автоматизации DevOps, например, для автоматического масштабирования, автоматического развертывания, CI/CD и мониторинга. **(+)**
    - B) Облачные технологии - это программное обеспечение для создания и управления виртуальными машинами.
    - C) Облачные технологии - это инструменты для мониторинга и логирования приложений.
    - D) Облачные технологии - это сетевые протоколы для передачи данных.