

**Федеральное государственное автономное образовательное
учреждение высшего образования
«Московский физико-технический институт
(национальный исследовательский университет)»**

УТВЕРЖДЕНО

**Директор физтех-школы
прикладной математики и
информатики**

А.М. Райгородский

	Рабочая программа дисциплины (модуля)
по дисциплине:	Основы непрерывной интеграции. Devops
по направлению:	Информатика и вычислительная техника
профиль подготовки:	Прикладная математика и информатика Физтех-школа Прикладной Математики и Информатики кафедра алгоритмов и технологий программирования
курс:	1
квалификация:	магистр

Семестр, формы промежуточной аттестации: 2 (весенний) - Дифференцированный зачет

Аудиторных часов: 60 всего, в том числе:

лекции: 30 час.

семинары: 30 час.

лабораторные занятия: 0 час.

Самостоятельная работа: 30 час.

Всего часов: 90, всего зач. ед.: 2

Количество контрольных работ, заданий: 1

Программу составил: О.Н. Ивченко, заведующий кафедрой

Программа обсуждена на заседании кафедры алгоритмов и технологий программирования 20.02.2025

Аннотация

DevOps — это методология, направленная на автоматизацию процессов сборки, настройки и развёртывания программного обеспечения. Она объединяет разработчиков и специалистов по IT-обслуживанию, способствуя тесному взаимодействию и интеграции их процессов для достижения высокого качества программного продукта.

DevOps внедряется как подход к улучшению гибкости, скорости и надёжности разработки, позволяя быстрее реагировать на изменения и обеспечивать пользователям стабильные обновления с минимальными рисками.

Методология фокусируется на стандартизации окружений разработки с целью быстрого переноса программного обеспечения через стадии жизненного цикла ПО, способствуя быстрому выпуску версий программного продукта. В идеале, системы автоматизации сборки и выпуска должны быть доступны всем разработчикам в любом окружении, и у разработчиков должен быть контроль над окружением разработки, а информационно-технологическая инфраструктура должна становиться более сфокусированной на приложении.

1. Цели и задачи

Цель дисциплины

- формирование у слушателей знаний и навыков по методологии DevOps для активного взаимодействия специалистов по разработке со специалистами по информационно-технологическому обслуживанию и взаимной интеграции их рабочих процессов для обеспечения качества продукта.

Задачи дисциплины

- изучение жизненного цикла (ЖЦ) программного обеспечения;
- изучить роль DevOps-инженера в ЖЦ;
- разобрать программные инструменты DevOps.

2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
УК-3 Способен организовывать и руководить работой команды, вырабатывая командную стратегию для достижения поставленной цели	УК-3.1 Организует и координирует работу участников проекта, способствует конструктивному преодолению возникающих разногласий и конфликтов
	УК-3.2 Учитывает в своей социальной и профессиональной деятельности интересы, особенности поведения и мнения (включая критические) людей, с которыми работает/взаимодействует, в том числе посредством корректировки своих действий
	УК-3.3 Способен предвидеть результаты (последствия) как личных, так и коллективных действий
	УК-3.4 Способен планировать командную работу, распределять поручения членам команды, организовывать обсуждение разных идей и мнений
ОПК-2 Имеет представление об актуальных проблемах науки и техники в области информатики и вычислительной техники, способен на научном языке формулировать профессиональные задачи	ОПК-2.1 Имеет представление о современном состоянии исследований в рамках тематической области своей профессиональной деятельности
	ОПК-2.2 Способен оценивать актуальность исследований в области информатики и вычислительной техники и их практическую значимость
	ОПК-2.3 Владеет профессиональной терминологией, используемой в современной научно-технической литературе, обладает навыками устного и письменного изложения результатов научной деятельности в рамках профессиональной коммуникации

ОПК-3 Способен выбирать и (или) разрабатывать подходы к решению типовых и новых задач в области информатики и вычислительной техники, учитывая особенности и ограничения различных методов решения	ОПК-3.1 Способен анализировать задачу, планировать пути решения, предлагать и комбинировать способы решения
	ОПК-3.2 Способен разрабатывать и модернизировать программное и аппаратное обеспечение информационных и автоматизированных систем
	ОПК-3.3 Способен использовать исследовательские методы при решении новых задач, применяя знания из различных областей науки (техники)
	ОПК-3.4 Владеет аналитическими и вычислительными методами решения, понимает и учитывает на практике границы применимости получаемых решений
	ОПК-3.5 Способен адаптировать зарубежные комплексы обработки информации и автоматизированного проектирования к нуждам отечественных предприятий
	ОПК-3.6 Способен самостоятельно приобретать, развивать и применять математические, естественнонаучные, социально-экономические и профессиональные знания для решения нестандартных задач, в том числе в новой или незнакомой среде и в междисциплинарном контексте
	ОПК-3.7 Способен разрабатывать оригинальные алгоритмы и программные средства, в том числе с использованием современных интеллектуальных технологий, для решения профессиональных задач
ПК-2 Понимает и способен применить в научно-исследовательской и прикладной деятельности основные законы естествознания, современный математический аппарат и алгоритмы, современные информационно-коммуникационные технологии	ПК-2.1 Знает основы научно-исследовательской деятельности в области информационных технологий, владеет знанием основ философии и методологии науки; знанием методов научных исследований и навыками их проведения
	ПК-2.2 Умеет применять полученные знания в области фундаментальных научных основ теории информации и решать стандартные задачи в собственной научно-исследовательской деятельности
	ПК-2.3 Имеет практический опыт научно-исследовательской деятельности в области информационно-коммуникационных технологий
	ПК-2.4 Владеет методами и алгоритмами решения задач цифровой обработки сигналов, использования сети Интернет, аннотирования, реферирования, библиографического поиска, опыт работы с научными источниками

3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

- понимание серверной и сетевой инфраструктуры;
- базовые навыки работы с операционными системами;
- базовые знания о базах данных;
- базовые навыки программирования;
- опыт работы с интерфейсами командной строки;
- понимание серверной и сетевой инфраструктуры.

уметь:

- применять нормативно-технические документы (стандарты и регламенты), определяющие требования к проектной и технической документации;
- использовать выбранную среду программирования для разработки процедур интеграции программных модулей.

владеть:

- навыками оценки результатов выполнения назначенных заданий на разработку процедур интеграции, сборку, подключение к внешней среде, проверку работоспособности выпусков программного продукта;
- навыками контроля и оценки качества разработанной проектной и технической документации.

4. Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий

4.1. Разделы дисциплины (модуля) и трудоемкости по видам учебных занятий

№	Тема (раздел) дисциплины	Трудоемкость по видам учебных занятий, включая самостоятельную работу, час.			
		Лекции	Семинары	Лаборат. работы	Самост. работа
1	Гибкие методологии разработки и DevOps	2	2		10
2	Контейнеризация и Docker	2	2		10
3	Создание контейнеров	2	2		10
4	Сети и Хранилища в Docker	2	2		
5	Модели ветвления и концепция CI/CD	2	2		
6	Основные функции Gitlab	2	2		
7	Создание пайплайнов Gitlab CI	2	2		
8	Разработка модульных пайплайнов	2	2		
9	Доставка и развертывание приложений	2	2		
10	Знакомство с Kubernetes	3	3		
11	Механики Контроллеров Kubernetes	3	3		
12	Хранение данных в Kubernetes	3	3		
13	Сетевая подсистема	3	3		
Итого часов		30	30		30
Подготовка к экзамену		0 час.			
Общая трудоёмкость		90 час., 2 зач.ед.			

4.2. Содержание дисциплины (модуля), структурированное по темам (разделам)

Семестр: 2 (Весенний)

1. Гибкие методологии разработки и DevOps

1. Проблематика DevOps

- * Функциональные колодцы.
- * Нисходящая спираль.

2. Гибкие методологии и DevOps культура

- * Взаимодействие в команде.
- * Современные подходы.

Чем обусловлено появление Agile?

- * Три пути DevOps: цикл поставки ценности, петли обратной связи, эксперименты и обучение.

3. Continuous Integration

- * Организация хранения кода.
- * Организация сборки.
- * Организация тестирования.
- * Быстрое получение обратной связи.

4. Сервера Continuous Integration

- * Обзор интерфейса и функционала Gitlab CI.

5. Измерение качества и статическое тестирование безопасности кода

- * Инструменты статического анализа качества кода:

SonarQube, Gitlab Analyzer.

2. Контейнеризация и Docker

Docker - это проект с открытым исходным кодом для автоматизации развертывания приложений в виде переносимых автономных контейнеров, выполняемых в облаке или локальной среде.

1. Контейнеризация

Обзор Linux- и Docker-контейнеров.

Устройство Docker: слои, образы, контейнеры,

Docker-файл, Registry.

Альтернативы Docker (containerd, podman,

LXC, cri-o).

2. Хранение данных: работа с S3, MinIO vs Ceph

3. Хранение и передача чувствительных данных:

Содержание уроков главы:
большой обзор Vault

3. Создание контейнеров

1. Как устроен процесс поставки IT-продукта

2. Системы хранения артефактов: Nexus

3. Continuous Delivery

Визуализация этапов поставки.

Организация процесса поставки. Delivery Pipeline.

Continuous Deployment.

4. Методология Twelve-Factor App

5. Бэкапирование

Организация резервного копирования.

Виды бэкапов.

Ротации бэкапов.

4. Сети и Хранилища в Docker

1. Контейнеризация

Обзор Linux- и Docker-контейнеров.

Устройство Docker: слои, образы, контейнеры,

Docker-файл, Registry.

Альтернативы Docker (containerd, podman,

LXC, cri-o).

2. Хранение данных: работа с S3, MinIO vs Ceph

3. Хранение и передача чувствительных данных:

5. Модели ветвления и концепция CI/CD

Получение кода из системы контроля версий и выполнение сборки.

Настройка инфраструктуры, автоматизированной через подход “инфраструктура как код”.

Копирование кода в целевую среду.

Настройка переменных окружения для целевой среды.

Развертывание компонентов приложения (веб-серверы, API-сервисы, базы данных).

Выполнение дополнительных действий, таких как перезапуск сервисов или вызов сервисов, необходимых для работоспособности новых изменений.

Выполнение тестов и откат изменений окружения в случае провала тестов.

Логирование и отправка оповещений о состоянии поставки.

6. Основные функции Gitlab

Администрирование репозитория. Позволяет хранить и управлять кодом, создавая различные ветки разработки.

Непрерывная интеграция (CI/CD). На платформе можно настраивать пайплайны, чтобы они регулярно проверяли, тестировали и развёртывали код после внесения изменений.

Мониторинг и анализ. Платформа предоставляет средства для отслеживания производительности и выявления узких мест, что позволяет командам быстро реагировать на проблемы.

Управление проектами. На платформе есть готовые инструменты, например, доски задач, расписания, инструменты планирования, которые помогают организовать работу.

Wiki. Команды могут создавать документацию и дополнять её.

Тестирование. GitLab предоставляет все необходимые инструменты для автоматизации тестирования.

Сборка. GitLab упрощает процесс сборки программного обеспечения благодаря встроенным инструментам.

Релиз. На этапе релиза GitLab предоставляет функции, позволяющие управлять версиями ПО и контролировать развёртывание.

Конфигурирование. GitLab предлагает инструменты для управления конфигурацией приложений.

Мониторинг. После развёртывания приложений важно следить за их работоспособностью. GitLab включает в себя функции мониторинга, которые позволяют отслеживать производительность и состояние приложений в реальном времени.

7. Создание пайплайнов Gitlab CI

Автоматизация процессов: GitLab CI/CD автоматизирует процессы сборки, тестирования и развёртывания, что значительно сокращает время на разработку и выпуск новых версий продукта.

Улучшение качества кода: Непрерывное тестирование кода помогает обнаруживать и исправлять ошибки на ранних этапах, что ведет к улучшению общего качества продукта.

Быстрый feedback: Разработчики быстро получают обратную связь о качестве кода и совместимости изменений, что позволяет оперативно вносить необходимые коррективы.

Гибкость и масштабируемость: GitLab CI/CD поддерживает различные среды и конфигурации, предоставляя гибкие настройки для масштабирования проектов.

Уменьшение рисков: Автоматизированное развёртывание в различные среды с использованием стратегий, таких как canary deployments и blue-green deployment, помогает минимизировать риски при выходе новых версий.

Экономия ресурсов: Минимизация ручных операций не только сокращает время на разработку и деплой, но и экономит ресурсы команды, позволяя разработчикам сосредоточиться на задачах высокого приоритета.

8. Разработка модульных пайплайнов

Фиксация и контроль версий. После написания исходного кода разработчики передают его в систему контроля версий, такую как Git. Это гарантирует, что каждое изменение в базе кода отслеживается и записывается. При необходимости изменения можно отменить, что способствует поддержанию чистой и организованной среды.

Сборка и компиляция. Pipeline извлекает последнюю версию кода из системы контроля версий и компилирует его в развёртываемый артефакт, например, двоичный файл или образ контейнера. Для сборки обычно используются такие инструменты, как Maven, Gradle или Docker.

Автоматизированное тестирование. Включает проведение комплекса тестов (модульных, интеграционных, сквозных), что обеспечивает высокое качество программного обеспечения.

Анализ качества. Проводится параллельно с автоматизированным тестированием. Используются инструменты статического анализа кода (SonarQube или ESLint) для оценки на соответствие стандартам кодирования, выявления потенциальных проблем кода и внедрения лучших практик.

Упаковка артефактов. После всех тестов и проверок качества артефакт упаковывается в формат, подходящий для развертывания, например, в виде образа контейнера или ZIP-файла.

Развертывание. Этот этап включает развертывание в целевой среде, которая подходит для дальнейшего тестирования или производства. Обычно используются инструменты, такие как Kubernetes и Jenkins.

Мониторинг и обратная связь заключительный этап обратной связи, который помогает команде выявить проблемы или узкие места в ПО и быстро решить их или внести изменения.

9. Доставка и развертывание приложений

- 1 коммит
- 2 сборка
- 3 тестирование
- 4 развертывание

10. Знакомство с Kubernetes

Модуль 1. Установка и настройка kubernetes кластера

Модуль 2. Объекты kubernetes, helm, cert manager

Модуль 3. Создание CI/CD в kubernetes

Модуль 4. Дополнительные материалы

11. Механики Контроллеров Kubernetes

Deployment. Kubernetes — наиболее распространенного подхода к развертыванию рабочей нагрузки в Kubernetes.

StatefulSet. Этот контроллер обеспечивает хранилище с отслеживанием состояния (stateful storage) для stateful приложений.

CronJob. Этот контроллер запускает задания — компоненты рабочей нагрузки Kubernetes, которые выполняют определенные задачи — в соответствии с установленным расписанием.

12. Хранение данных в Kubernetes

Создание yaml файла с pv-nfs.yaml с описанием Persistence Volume на основе NFS.

13. Сетевая подсистема

Принципы работы с сетевой подсистемой и настройка системы.

5. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю)

Компьютеры либо ноутбуки с доступом в интернет, камерой и микрофоном. Практические занятия (семинары) и самостоятельная работа студентов должна проводиться в специализированных аудиториях с комплектом мультимедийного оборудования и/или доской для записей материалов.

6.Перечень рекомендуемой литературы

Основная литература

1. Введение в программирование , учебное пособие / И. Ю. Баженова, В. А. Сухомлин. — Москва, ИНТУИТ, 2016.— URL: <https://e.lanbook.com/book/100695> (дата обращения: 30.12.2020). - Полный текст (Режим доступа : из сети МФТИ / Удаленный доступ)
2. Программирование на C++ [Электронный ресурс] / Н. Дейл, Ч. Уимз, М. Хедингтон. — М., ДМК Пресс, 2007.— URL: <https://e.lanbook.com/book/1219> (дата обращения: 26.01.2021). - Полный текст (Режим доступа : из сети МФТИ / Удаленный доступ)

Дополнительная литература

7. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)

Не используются

8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень необходимого программного обеспечения и информационных справочных систем (при необходимости)

В качестве программных средств будут использоваться редакторы исходного кода (Vim и другие) или интегрированные среды разработки (IntelliJ IDEA, Visual Studio и другие).

9. Методические указания для обучающихся по освоению дисциплины (модуля)

Успешное освоение курса требует напряжённой самостоятельной работы студента. В программе курса приведено минимально необходимое время для работы студента над темой.

Самостоятельная работа включает в себя:

- проработку учебного материала (по конспектам лекций, учебной и научной литературе), подготовку ответов на вопросы, предназначенных для самостоятельного изучения, доказательство отдельных утверждений, свойств;
- выполнение индивидуальных домашних заданий и итогового проекта.

Промежуточный контроль знаний проводится в виде индивидуальных домашних работ в формате проектной работы.

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению: Информатика и вычислительная техника
профиль подготовки: Прикладная математика и информатика
Физтех-школа Прикладной Математики и Информатики
кафедра алгоритмов и технологий программирования
курс: 1
квалификация: магистр

Семестр, формы промежуточной аттестации: 2 (весенний) - Дифференцированный зачет

Разработчик: О.Н. Ивченко, заведующий кафедрой

1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
УК-3 Способен организовывать и руководить работой команды, вырабатывая командную стратегию для достижения поставленной цели	УК-3.1 Организует и координирует работу участников проекта, способствует конструктивному преодолению возникающих разногласий и конфликтов
	УК-3.2 Учитывает в своей социальной и профессиональной деятельности интересы, особенности поведения и мнения (включая критические) людей, с которыми работает/взаимодействует, в том числе посредством корректировки своих действий
	УК-3.3 Способен предвидеть результаты (последствия) как личных, так и коллективных действий
	УК-3.4 Способен планировать командную работу, распределять поручения членам команды, организовывать обсуждение разных идей и мнений
ОПК-2 Имеет представление об актуальных проблемах науки и техники в области информатики и вычислительной техники, способен на научном языке формулировать профессиональные задачи	ОПК-2.1 Имеет представление о современном состоянии исследований в рамках тематической области своей профессиональной деятельности
	ОПК-2.2 Способен оценивать актуальность исследований в области информатики и вычислительной техники и их практическую значимость
	ОПК-2.3 Владеет профессиональной терминологией, используемой в современной научно-технической литературе, обладает навыками устного и письменного изложения результатов научной деятельности в рамках профессиональной коммуникации
ОПК-3 Способен выбирать и (или) разрабатывать подходы к решению типовых и новых задач в области информатики и вычислительной техники, учитывая особенности и ограничения различных методов решения	ОПК-3.1 Способен анализировать задачу, планировать пути решения, предлагать и комбинировать способы решения
	ОПК-3.2 Способен разрабатывать и модернизировать программное и аппаратное обеспечение информационных и автоматизированных систем
	ОПК-3.3 Способен использовать исследовательские методы при решении новых задач, применяя знания из различных областей науки (техники)
	ОПК-3.4 Владеет аналитическими и вычислительными методами решения, понимает и учитывает на практике границы применимости получаемых решений
	ОПК-3.5 Способен адаптировать зарубежные комплексы обработки информации и автоматизированного проектирования к нуждам отечественных предприятий
	ОПК-3.6 Способен самостоятельно приобретать, развивать и применять математические, естественнонаучные, социально-экономические и профессиональные знания для решения нестандартных задач, в том числе в новой или незнакомой среде и в междисциплинарном контексте
	ОПК-3.7 Способен разрабатывать оригинальные алгоритмы и программные средства, в том числе с использованием современных интеллектуальных технологий, для решения профессиональных задач
	ПК-2.1 Знает основы научно-исследовательской деятельности в области информационных технологий, владеет знанием основ философии и методологии науки; знанием методов научных исследований и навыками их проведения

ПК-2 Понимает и способен применить в научно-исследовательской и прикладной деятельности основные законы естествознания, современный математический аппарат и алгоритмы, современные информационно-коммуникационные технологии	ПК-2.2 Умеет применять полученные знания в области фундаментальных научных основ теории информации и решать стандартные задачи в собственной научно-исследовательской деятельности
	ПК-2.3 Имеет практический опыт научно-исследовательской деятельности в области информационно-коммуникационных технологий
	ПК-2.4 Владеет методами и алгоритмами решения задач цифровой обработки сигналов, использования сети Интернет, аннотирования, реферирования, библиографического поиска, опыт работы с научными источниками

2. Показатели оценивания компетенций

В результате изучения дисциплины «Основы непрерывной интеграции. Devops» обучающийся должен:

знать:

- понимание серверной и сетевой инфраструктуры;
- базовые навыки работы с операционными системами;
- базовые знания о базах данных;
- базовые навыки программирования;
- опыт работы с интерфейсами командной строки;
- понимание серверной и сетевой инфраструктуры.

уметь:

- применять нормативно-технические документы (стандарты и регламенты), определяющие требования к проектной и технической документации;
- использовать выбранную среду программирования для разработки процедур интеграции программных модулей.

владеть:

- навыками оценки результатов выполнения назначенных заданий на разработку процедур интеграции, сборку, подключение к внешней среде, проверку работоспособности выпусков программного продукта;
- навыками контроля и оценки качества разработанной проектной и технической документации.

3. Перечень типовых (примерных) вопросов, заданий, тем для подготовки к текущему контролю

Перечень примерных (типовых) вопросов к текущему контролю:

1. Что такое DevOps?
2. Жизненный цикл ПО
3. DevOps-инженер – роль в проекте разработки и внедрения ПО
4. Обзор систем виртуализации и контейнеризации
5. Введение в экосистему контейнеров на основе Docker
6. Настройка рабочего окружения, подготовка и запуск Docker-контейнеров
7. Структура и содержание профилей информационных систем
8. Методологические основы проектирования информационных систем
9. Методология структурного анализа и проектирования информационных систем SADT
10. Основные понятия нотации IDEF0

4. Перечень типовых (примерных) вопросов и тем для проведения промежуточной аттестации обучающихся

Перечень вопросов к дифференцированному зачету:

1. Жизненный цикл ПО.
2. DevOps-инженер – роль в проекте разработки и внедрения ПО.

3. Экосистема контейнеров на основе Docker.
4. Автоматизация разработки, тестирования и доставки ПО с использованием Jenkins.
5. Основы управления конфигурацией с использованием Ansible.
6. Основные встроенные модули Ansible.
7. Взаимодействие Ansible с Docker.
8. Основы оркестрации с использованием Kubernetes.
9. Микросервисная архитектура.
10. Взаимодействие Kubernetes с Docker.
11. Особенности сбора метрик с микросервисов и Docker-контейнеров.

Пример билета:

Билет №1

1. Жизненный цикл ПО.
2. DevOps-инженер – роль в проекте разработки и внедрения ПО.

Критерии оценивания

отлично

10: всесторонние, систематизированные, глубокие знания учебной программы дисциплины и умение уверенно применять их на практике при решении конкретных задач, свободное и правильное обоснование принятых решений;

9: систематизированные, глубокие знания учебной программы дисциплины и умение уверенно применять их на практике при решении конкретных задач, правильное обоснование принятых решений;

8: глубокие знания учебной программы дисциплины и умение применять их на практике при решении конкретных задач, правильное обоснование принятых решений;

хорошо

7: твердо знает материал, грамотно и по существу излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач некоторые неточности;

6: знает материал, грамотно излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач некоторые неточности;

5: знает основной материал, грамотно излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач неточности;

удовлетворительно

4: фрагментарный, разрозненный характер знаний, недостаточно правильные формулировки базовых понятий, нарушения логической последовательности в изложении программного материала, но при этом он владеет основными разделами учебной программы, необходимыми для дальнейшего обучения и может применять полученные знания по образцу в стандартной ситуации;

3: характер знаний достаточен для дальнейшего обучения и может применять полученные знания по образцу в стандартной ситуации;

неудовлетворительно

2: не знает большей части основного содержания учебной программы дисциплины, допускает грубые ошибки в формулировках основных понятий дисциплины и не умеет правильно использовать полученные знания при решении типовых практических задач.

1: не знает формулировок основных понятий дисциплины и не умеет использовать полученные знания при решении типовых практических задач.

5. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

Дифференцированный зачет выполняется в формате устного ответа на билет. На подготовку дается 30 минут.