МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
К Г Э У **«КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ»**
(ФГБОУ ВО «КГЭУ»)

УТВЕРЖДАЮ
Ректор
_____ Э.Ю. Абдуллазянов
« 5 » мая 2023 г.

Дополнительная профессиональная программа
(программа профессиональной переподготовки)

DevOps-инженер
(наименование программы)

дополнительное профессиональное образование
(подвид дополнительного образования)

Казань 2023 г.

Дополнительную профессиональную программу (программу профессиональной переподготовки) разработал:

Руководитель программы «DevOps-инженер»,
директор института цифровых
технологий и экономики,
кандидат технических наук, доцент Э.И. Беляев

Дополнительная профессиональная программа (программа профессиональной переподготовки) рассмотрена и одобрена на заседании рабочей группы проекта «Цифровая кафедра»:

Руководитель проекта «Цифровая кафедра»,
доцент кафедры ИТИС,
кандидат экономических наук, доцент Г.Р. Сибаева

Дополнительная профессиональная программа (программа профессиональной переподготовки) рассмотрена на методическом совете Института цифровых технологий и экономики протокол № 6 от 25 апреля 2023 г.

Согласовано:

Директор института дополнительного
профессионального образования,
доктор технических наук, профессор В.К. Ильин

Эксперты:

Рецензирование дополнительной профессиональной программы (программы профессиональной переподготовки) провели:

Девятков Владимир Васильевич - директор ООО «Элина-Компьютер»;

Алимова Элина Сергеевна - руководитель проектов ООО «Профцифра»;

Кузнецов Дмитрий Андреевич – ведущий разработчик ООО «ИНКОР».

РЕЦЕНЗИЯ
на дополнительную профессиональную программу
(программу профессиональной переподготовки)
«DevOps-инженер»,
разработанную
ФГБОУ ВО «Казанский государственный энергетический университет»

Дополнительная профессиональная программа «DevOps-инженер» разработана в рамках федерального проекта «Развитие кадрового потенциала ИТ-отрасли» национальной программы «Цифровая экономика Российской Федерации», является необходимой ступенью совершенствования инженерной подготовки и чрезвычайно актуальна.

Дополнительная профессиональная программа, представленная на рецензию, включает в себя необходимые компоненты - календарный учебный график, учебный- тематический план, рабочие программы и оценочные материалы. Материально-техническое обеспечение учебного процесса по программе полностью соответствует современным требованиям.

Целью данной программы является подготовка специалистов (DevOps-инженеров), владеющих современными технологиями в области ИТ, способных синхронизировать все этапы создания программного продукта, включая его разработку и тестирование, а также автоматизацию развертывания и сопровождения за счёт применения DevOps-технологии.

В качестве сильных сторон рецензируемой программы переподготовки следует отметить актуальность, привлечение для реализации опытного профессорско- преподавательского состава, а также ведущих представителей индустрии. Дополнительная профессиональная программа «DevOps-инженер» в полной мере отвечает современному уровню развития информационно-коммуникационных технологий.

Директор ООО «Элина – Компьютер»
доктор экономических наук В.В. Девятков
раб. тел. 8(843)292-38-67
vladimir@elina-computer.ru

РЕЦЕНЗИЯ
на дополнительную профессиональную программу
(программу профессиональной переподготовки)
«DevOps-инженер»

Представленная дополнительная профессиональная программа (программа профессиональной переподготовки) разработана в рамках федерального проекта «Развитие кадрового потенциала И'Г-отрасли» национальной программы «Цифровая экономика Российской Федерации». Целью программы профессиональной переподготовки «DevOps-инженер» является представление программной инженерии в виде целостного изложения, освещая концепцию процесса, различные методологии разработки программного обеспечения, отличие программной инженерии от других отраслей. В рамках программы профессиональной переподготовки «DevOps-инженер» рассматриваются основные подходы к организации командной разработки программного обеспечения, современные технологии разработки программного обеспечения, процессы командной разработки ПО, анализируются формальные и гибкие технологии разработки ПО, способы обеспечения качества программных продуктов и мотивации членов команды разработки ПО

Рецензируемая программа профессиональной переподготовки регламентирует цели, ожидаемые результаты, условия и технологии реализации образовательного процесса, оценку качества подготовки выпускника. Содержание включает в себя: характеристику квалификации, связанной с новым видом профессиональной деятельности и трудовыми функциями в соответствии с профессиональным стандартом, учебный план, рабочие программы дисциплин и практики, календарный учебный график и методические материалы. Материально-техническое обеспечение учебного процесса полностью соответствует современным требованиям.

Дополнительная профессиональная программа (программа профессиональной переподготовки) «DevOps-инженер» в полной мере соответствует современному уровню развития информационно-коммуникационных технологий.

Руководитель проектов
ООО «Профцифра»
8 (965) 581-68-65
e.alimova@profcifra.ru

подпись, МП

Алимова Элина Сергеевна

РЕЦЕНЗИЯ

на дополнительную профессиональную программу
(программу профессиональной переподготовки)
«DevOps-инженер»,
разработанную
ФГБОУ ВО «Казанский государственный энергетический университет»

Дополнительная профессиональная программа «DevOps-инженер» разработана в рамках федерального проекта «Развитие кадрового потенциала ИТ-отрасли» национальной программы «Цифровая экономика Российской Федерации».

Дополнительная профессиональная программа, представленная на рецензию, включает в себя календарный учебный график, учебный- тематический план, рабочие программы и оценочные материалы. Материально-техническое обеспечение учебного процесса полностью соответствует современным требованиям.

Целью программы профессиональной переподготовки «DevOps-инженер» является формирование дополнительных компетенций:

- 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;
- по использованию программных инструментов DevOps: Docker, Jenkins, Ansible, Kubernets и Prometheus.

В качестве сильных сторон рецензируемой программы следует отметить актуальность, привлечение для реализации опытного профессорско- преподавательского состава, а также ведущих представителей работодателя. Дополнительная профессиональная программа «DevOps-инженер» в полной мере отвечает современному уровню развития информационных технологий в энергетической отрасли.

Рецензент

Ведущий разработчик
ООО «ИНКОР» **Д.А. Кузнецов**
номер телефона (раб.): +7 (843) 526-40-08
e-mail: dev@in-korp.ru

I. Общие положения

1. Дополнительная профессиональная программа (программа профессиональной переподготовки) ИТ-профиля «DevOps-инженер» (далее - Программа) разработана в соответствии с нормами Федерального закона РФ от 29 декабря 2012 года № 273-ФЗ «Об образовании в Российской Федерации», с учетом требований приказа Минобрнауки России от 1 июля 2013 г. № 499 «Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным профессиональным программам», с изменениями, внесенными приказом Минобрнауки России от 15 ноября 2013 г. № 1244 «О внесении изменений в Порядок организации и осуществления образовательной деятельности по дополнительным профессиональным программам, утвержденный приказом Министерства образования и науки Российской Федерации от 1 июля 2013 г. № 499», приказа Министерства образования и науки РФ от 23 августа 2017 г. № 816 «Об утверждении Порядка применения организациями, осуществляющими образовательную деятельность, электронного обучения, дистанционных образовательных технологий при реализации образовательных программ»; паспорта федерального проекта «Развитие кадрового потенциала ИТ-отрасли» национальной программы «Цифровая экономика Российской Федерации»; постановления Правительства Российской Федерации от 13 мая 2021 г. № 729 «О мерах по реализации программы стратегического лидерства «Приоритет-2030» (в редакции постановления Правительства Российской Федерации от 14 марта 2022 г. № 357 «О внесении изменений в постановление Правительства Российской Федерации от 13 мая 2021 г. № 729»); приказа Министерства цифрового развития, связи и массовых коммуникаций Российской Федерации от 28 февраля 2022 г. № 143 «Об утверждении методик расчета показателей федеральных проектов национальной программы «Цифровая экономика Российской Федерации» и признании утратившими силу некоторых приказов Министерства цифрового развития, связи и массовых коммуникаций Российской Федерации об

утверждении методик расчета показателей федеральных проектов национальной программы «Цифровая экономика Российской Федерации» (далее - приказ Министерства цифрового развития, связи и массовых коммуникаций Российской Федерации № 143); федеральных государственных образовательных стандартов высшего образования по направлениям подготовки 09.03.01 «Информатика и вычислительная техника» (уровень бакалавриата), утвержденного приказом Минобрнауки России от 19 сентября 2017 г. № 929 и 09.03.03 «Прикладная информатика» (уровень бакалавриата), утвержденного приказом Минобрнауки России от 19 сентября 2017 г. № 922, (далее вместе - ФГОС ВО)), а также профессиональных стандартов «Специалист по интеграции прикладных решений», утвержденного приказом Министерства труда и социальной защиты РФ от 05.09.2017 № 658н и «Руководитель разработки программного обеспечения», утвержденного приказом Министерства труда и социальной защиты РФ от 20.07.2022 № 423н.

2. Профессиональная переподготовка заинтересованных лиц (далее - Слушатели), осуществляемая в соответствии с Программой (далее - Подготовка), имеющей отраслевую направленность «Информационно-коммуникационные технологии», проводится в ФГБОУ ВО «Казанский государственный энергетический университет» (далее - ФГБОУ ВО КГЭУ) в соответствии с учебным планом в очной форме обучения.

3. Разделы, включенные в учебный план Программы, используются для последующей разработки календарного учебного графика, учебно-тематического плана, рабочей программы, оценочных и методических материалов. Перечисленные документы разрабатываются ФГБОУ ВО КГЭУ самостоятельно, с учетом актуальных положений законодательства об образовании, законодательства в области информационных технологий и смежных областей знаний ФГОС ВО и профессиональных стандартов 06.041 «Специалист по интеграции прикладных решений», 06.017 «Руководство разработкой компьютерного программного обеспечения»

4. Программа регламентирует требования к профессиональной

переподготовке в области разработки компьютерного программного обеспечения и интеграции приложений информационных систем и облачных сервисов.

Срок освоения Программы составляет 9 месяцев, 400 академических часов.

К освоению Программы в рамках проекта допускаются лица:

- получающие высшее образование по очной (очно-заочной) форме, лица, освоившие основную профессиональную образовательную программу (далее - ОПОП ВО) бакалавриата - в объеме не менее первого курса (бакалавры 2-го курса), ОПОП ВО специалитета - не менее первого и второго курсов (специалисты 3-го курса).

5. Область профессиональной деятельности – 06 связь, информационные и коммуникационные технологии.

II. Цель

6. Целью ДПП ПП является формирование у слушателей, обучающихся по специальностям и направлениям подготовки, отнесенным к ИТ-сфере дополнительных компетенций в области методологии DevOps, для активного взаимодействия специалистов по разработке со специалистами по информационно-технологическому обслуживанию и взаимной интеграции их рабочих процессов для обеспечения качества продукта, по использованию программных инструментов DevOps: Docker, Jenkins, Ansible, Kubernets и Prometheus.

III. Характеристика новой квалификации и связанных с ней видов профессиональной деятельности, трудовых функций и (или) уровней квалификации

7. Виды профессиональной деятельности, трудовая функция, указанные в профессиональном стандарте по соответствующей должности инженера по интеграции прикладных решений, представлены в таблице 1:

Таблица 1

Характеристика новой квалификации, связанной с видом профессиональной деятельности и трудовыми функциями в соответствии с профессиональными стандартами «Специалист по интеграции прикладных решений», «Руководитель разработки программного обеспечения»

Область профессиональной деятельности	Тип задач профессиональной деятельности	Код и наименование профессиональной компетенции	Трудовые действия	Трудовая функция	Обобщенная трудовая функция	Вид профессиональной деятельности
06 Связь, информационные и коммуникационные технологии	проектный	ПК-6 Применяет методологию и принципы непрерывной разработки, интеграции и развертывания ПО	Определение возможности достижения соответствия интеграционного решения первоначальным требованиям заказчика. Подготовка фрагментов технического задания на создание (модификацию) интеграционного решения. Оценка и согласование объемов работ и сроков их выполнения. Управление версиями программного обеспечения в соответствии с регламентом и выбранной системой управления версиями	Инженерно-технологическая поддержка процесса согласования требований к интеграционному решению	Выполнение работ по созданию (модификации) и сопровождению интеграционных решений	Интеграция приложений информационных систем и облачных сервисов
06 Связь, информационные и коммуникационные технологии	проектный	ПК-27 Разворачивает и настраивает инструменты непрерывной разработки ИС	Развертывание и настройка выбранной интеграционной платформы в соответствии с техническими спецификациями на интеграционное решение. Подключение интеграционного решения к компонентам внешней среды. Проверка работоспособности интеграционного решения	Конфигурирование интеграционного решения на базе интеграционной платформы	Выполнение работ по созданию (модификации) и сопровождению интеграционных решений	Интеграция приложений информационных систем и облачных сервисов
06 Связь, информационные и коммуникационные технологии	проектный	ПК-29 Разрабатывает программное обеспечение	Редактирование программного кода. Инициирование разработки проектной и технической документации на компьютерное программное обеспечение. Контроль и оценка качества разработанной проектной и технической документации на компьютерное программное обеспечение	Руководство разработкой программного кода	Руководство процессами разработки компьютерного программного обеспечения	Руководство разработкой компьютерного программного обеспечения

Таблица 2

Характеристика новой и развиваемой цифровой компетенции в ИТ-сфере, связанной с уровнем формирования и развития в результате освоения Программы «DevOps-инженер»

Наименование сферы	**Код и наименование профессиональной компетенции**	**Примерный набор инструментов для освоения и применения компетенций**	**МИНИМАЛЬНЫЙ ИСХОДНЫЙ УРОВЕНЬ РАЗВИТИЯ КОМПЕТЕНЦИЙ** Способность не проявляется/ проявляется в степени, недостаточной для отнесения к 1 уровню сформированности компетенции	**БАЗОВЫЙ УРОВЕНЬ РАЗВИТИЯ КОМПЕТЕНЦИЙ** Способность проявляется под внешним контролем / при внешней постановке задачи/ обучающийся пользуется готовыми, рекомендованными продуктами
Стандарты и методики в ИТ	ПК-6 Применяет методологию и принципы непрерывной разработки, интеграции и развертывания ПО	DevOps, CI/CD, DQ DevOps, Scaled Agile Framework (SAFe), Git, Bamboo / GitLab (CI/CD инструменты), docker, Prometheus, DevOps, Grafana, Ansible	Не применяет методологию и принципы непрерывной разработки, интеграции и развертывания ПО	Применяет методологию и принципы непрерывной разработки, интеграции и развертывания ПО под контролем в составе команды
Прикладные программные комплексы и системы	ПК-27 Разворачивает и настраивает инструменты непрерывной разработки ИС	Jenkins, TeamCity, GoCD, Git, GitFlic, DQ Devops, Ansible, Prometheus, TeamCity, docker, Kubernetes, Bamboo / GitLab (CI/CD инструменты)	не применяет	Разворачивает инструменты непрерывной разработки, настраивает репозиторий и линейный pipeline под внешним контролем
Средства программной разработки	ПК-29 Разрабатывает программное обеспечение	Python, JavaScript, C#, PHP	Решает учебные задачи по программированию, руководствуясь подробной пошаговой задачей под контролем опытных наставников	Участвует в разработке ПО, применяет языки программирования для решения простых с технической точки зрения задач, руководствуясь общей постановкой задач под контролем опытных специалистов

IV. Характеристика новых и развиваемых цифровых компетенций, формирующихся в результате освоения программы

8. В ходе освоения Программы Слушателем приобретаются следующие профессиональные компетенции:

ПК-6 Применяет методологию и принципы непрерывной разработки, интеграции и развертывания ПО;

ПК-27 Разворачивает и настраивает инструменты непрерывной разработки ИС.

В ходе освоения Программы Слушателем совершенствуются следующие профессиональные компетенции:

ПК-29 Разрабатывает программное обеспечение.

V. Планируемые результаты обучения по ДПП ПП

9. Результатами подготовки слушателей по Программе является получение компетенции, необходимой для выполнения нового вида профессиональной деятельности в области разработки программного обеспечения и определения архитектурных и реализационных решений по интеграции приложений информационных систем и облачных сервисов; приобретение новой квалификации – Инженер по интеграции прикладных решений.

10. В результате освоения Программы слушатель должен:

ПК-6 Применяет методологию и принципы непрерывной разработки, интеграции и развертывания ПО:

знать:

– программные средства и платформы инфраструктуры информационных технологий организаций;

– современные инструментальные средства DevOps при ведении профессиональной деятельности;

уметь:

– вырабатывать варианты реализации требований заказчика к интеграционному решению;

иметь навыки:

– определения возможности достижения соответствия интеграционного решения первоначальным требованиям заказчика;

– управления версиями программного обеспечения в соответствии с регламентом и выбранной системой управления версиями;

ПК-27 Разворачивает и настраивает инструменты непрерывной разработки ИС:

знать:

– современные стандарты информационного взаимодействия систем;

– методы и средства сборки и интеграции программных модулей, сервисов и компонент;

– современные модели и методы анализа предметной области;

– системы виртуализации и контейнеризации;

– сценарии развертывания конфигураций ИС;

– системы оркестрации с использованием Kubernets;

уметь:

– выполнять процедуры развертывания и настройки выбранной интеграционной платформы;

– выполнять процедуры сборки программных модулей, сервисов и компонент интеграционного решения в соответствии с техническим заданием;

– настраивать рабочее окружение, подготавливать и запускать Docker-контейнеры;

– управлять конфигурацией с использованием Ansible;

иметь навыки:

– развертывания и настройки выбранной интеграционной платформы в соответствии с техническими спецификациями на интеграционное решение;

– сборки программных модулей, сервисов и компонент интеграционного

решения на базе выбранной интеграционной платформы в соответствии с техническими спецификациями;

ПК-29 Разрабатывает программное обеспечение:

знать:

– выбранный язык программирования, особенности программирования на этом языке;

– компоненты программно-технических архитектур, существующие приложения и интерфейсы взаимодействия с ними;

– современные модели и методы анализа предметной области;

– синтаксис выбранного языка программирования, особенности программирования на этом языке, стандартные библиотеки языка программирования;

– методологии разработки программного обеспечения;

– особенности стека технологий, применяемого в проекте, при организации разработки программного обеспечения;

уметь:

– применять стандартные возможности выбранной среды программирования для редактирования программного кода;

– использовать возможности имеющейся технической и/или программной архитектуры;

– тестировать результаты собственной работы;

иметь навыки:

– разработки и редактирования программного кода;

– оценки качества и эффективности программного кода.

VI. Организационно-педагогические условия реализации ДПП ПП

11. Реализация Программы должна обеспечить получение компетенции, необходимой для выполнения нового вида профессиональной деятельности в области разработки программного обеспечения и определения

архитектурных и реализационных решений по интеграции приложений информационных систем и облачных сервисов.

12. Учебный процесс организуется с применением дистанционных образовательных технологий, инновационных технологий и методик обучения, способных обеспечить получение слушателями знаний, умений и навыков в области – 06 связь, информационные и коммуникационные технологии.

13. Реализация Программы обеспечивается научно-педагогическими кадрами ФГБОУ ВО КГЭУ, допустимо привлечение к образовательному процессу высококвалифицированных специалистов ИТ-сферы и/или дополнительного профессионального образования в части, касающейся профессиональных компетенций в области создания алгоритмов и программ, пригодных для практического применения, с обязательным участием представителей профильных организаций-работодателей. Возможно привлечение региональных руководителей цифровой трансформации (отраслевых ведомственных и/или корпоративных) к проведению итоговой аттестации, привлечение работников организаций реального сектора экономики субъектов Российской Федерации.

VII. Учебный план ДПП ПП

14. Объем Программы составляет 400 часов.

15. Учебный план Программы определяет перечень, последовательность, общую трудоемкость разделов и формы контроля знаний.

Учебный план ДПП ПП «DevOps - инженер»

Наименование дисциплин		Общая трудоемкость, час.	По учебному плану с использованием дистанционных образовательных технологий, час.						СРС, час.	Промежуточная аттестация	
			Аудиторные занятия, час.			Дистанционные занятия, час.					
			всего	из них		всего	из них			Зачет с оценкой	Экзамен
				лекции	лаб. раб		лекции.	лаб. раб			
1		2	3	4	5	6	7	8	9	10	11
1.	Командные коммуникации и системы управления задачами	30	20	4	16				8		2
2.	Системы контроля версий	30	20	4	16				8		2
3.	Организация высокой доступности БД	30	20	4	16				8		2
	Итого	**90**	**60**	**12**	**48**				**24**		**6**
4.	Использование Python для DevOps	38	28	4	24				8		2
5.	Администрирование Linux	38	28	4	24				8		2
	Итого	**76**	**56**	**8**	**48**				**16**		**4**
6.	Системы виртуализации и контейнеризации (Docker)	40	28	4	24				10		2
7.	Системы непрерывной интеграции (Jenkins)	36	24	4	20				10		2
8.	Системы управления конфигурацией. Инфраструктура как код (Ansible)	34	22	2	20				10		2
9.	Системы оркестрации (Kubernets)	38	28	4	24				8		2
10	Проектный практикум	40	32	-	32				8	✔	
11	Практика / Стажировка	36							36	✔	
	Итого	**224**	**134**	**14**	**120**				**82**		**8**
12	**Итоговая аттестация (Демонстрационный экзамен)**								**6**		4
	ВСЕГО	**400**	**250**	**34**	**216**				**128**		**22**

VIII. Календарный учебный график

16. Календарный учебный график представляет собой график учебного процесса, устанавливающий последовательность и продолжительность обучения и итоговой аттестации по учебным дням.

Календарный учебный график ДПП ПП «DevOps-инженер» на 2023-2024 учебный год

Месяц / День недели	Сентябрь					Октябрь					Ноябрь				Декабрь				Январь					Февраль				Март				Апрель					Май				Июнь			
Пн		4	11	18	25	2	9	16	23	30	6	13	20	27	4	11	18	25	1	8	15	22	29	5	12	19	26	4	11	18	25	1	8	15	22	29	6	13	20	27	3	10	17	24
Вт		5	12	19	26	3	10	17	24	31	7	14	21	28	5	12	19	26	2	9	16	23	30	6	13	20	27	5	12	19	26	2	9	16	23	30	7	14	21	28	4	11	18	25
Ср		6	13	20	27	4	11	18	25	1	8	15	22	29	6	13	20	27	3	10	17	24	31	7	14	21	28	6	13	20	27	3	10	17	24	1	8	15	22	29	5	12	19	26
Чт		7	14	21	28	5	12	19	26	2	9	16	23	30	7	14	21	28	4	11	18	25	1	8	15	22	29	7	14	21	28	4	11	18	25	2	9	16	23	30	6	13	20	27
Пт	1	8	15	22	29	6	13	20	27	3	10	17	24	1	8	15	22	29	5	12	19	26	2	9	16	23	1	8	15	22	29	5	12	19	26	3	10	17	24	31	7	14	21	28
Сб	2	9	16	23	30	7	14	21	28	4	11	18	25	2	9	16	23	30	6	13	20	27	3	10	17	24	2	9	16	23	30	6	13	20	27	4	11	18	25	1	8	15	22	29
Вс	3	10	17	24	1	8	15	22	29	5	12	19	26	3	10	17	24	31	7	14	21	28	4	11	18	25	3	10	17	24	31	7	14	21	28	5	12	19	26	2	9	16	23	30
Неделя / Учебная дисциплина, практика	1		2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
Командные коммуникации и системы управления задачами					4	4	4	4	4																																			
Системы контроля версий					4	4	4	4	4																																			
Организация высокой доступности БД												4	4	4	2	2	2	2																									
Использование Python для DevOps										4	4	4	2	2	4	2	4	2																									
Администрирование Linux					4	4	4	4	4	4	4																																	
Системы виртуализации и контейнеризации (Docker)										2	2	2	4	4	4	6	4																											
Системы непрерывной интеграции (Jenkins)																								2	4		4	4	4	2	4													
Системы управления конфигурацией. Инфраструктура как код (Ansible)																								4	2	4	2		2	4	4													
Системы оркестрации (Kubernets)																								4	4	4	4	4	4	4														
Проектный практикум										2	2	2	2	2	2	2	2							2	2	2	2	2	2	2	2													
Практика / Стажировка																																П	П	П	П									
Итого аудиторных часов					12	12	12	12	12	12	12	12	12	12	12	12	12	4						12	12	10	12	10	12	12	10													

Пояснения к графику:
01.09.2023 - 23.09.2023 - комплексная оценка (входной ассесмент);
01.04.2024 - 27.04.2024 - практика/стажировка;
06.05.2024 - 25.05.2024 - комплексная оценка (итоговый ассесмент);
27.05.2024 - 22.05.2024 - итоговая аттестация

IX. Рабочая программа учебных дисциплин

17. Рабочие программы содержат перечень разделов и тем, а также рассматриваемых в них вопросов с учетом их трудоемкости.

Рабочие программы разрабатываются ФГБОУ ВО КГЭУ с учетом профессиональных стандартов 06.041«Специалист по интеграции прикладных решений» и 06.017 «Руководитель разработки программного обеспечения».

17.1 Наименование дисциплины - Командные коммуникации и системы управления задачами

Лекций – 4 ч, лабораторных -16 ч, сам. раб. - 8

№ п/п	Наименование и краткое содержание	Объем, часов
1.	Системы управления проектами (Jira) Проекты. Задачи. Дорожные карты.	2 л. 8 лаб. 4 сам.
2.	Agile-разработка с помощью Jira. Scrum, Agile, Kanban доски.	2 л. 8 лаб. 4 сам.

17.2 Наименование дисциплины - Системы контроля версий

Лекций – 4 ч, лабораторных -16 ч, сам. раб.-8

№ п/п	Наименование и краткое содержание	Объем, часов
1.	Системы контроля версий Git	2 л.
2.	Сервис GitLab.	1 л. 8 лаб. 4 сам.
3.	CI/CD технология непрерывной интеграции и доставки.	1 л. 8 лаб. 4 сам.

17.3 Наименование дисциплины - Организация высокой доступности БД

Лекций – 8 ч, лабораторных -24 ч, сам. раб.-12

№ п/п	Наименование и краткое содержание	Объем, часов
1.	Раздел 1 Резервное копирование	
1.1	Лек.1 Логическое резервирование Понятие логической резервной копии. Копирование и восстановление отдельных таблиц. Копирование и восстановление баз данных. Копирование и восстановление кластера.	2 л.

№ п/п	Наименование и краткое содержание	Объем, часов
1.2	Лек. 2 Базовая резервная копия. Понятие физической резервной копии. Холодное резервирование. Горячее резервирование	2 л.
1.3	Сам. Архив журнала предзаписи Файловый архив — непрерывная архивация. Потоковый архив - утилита pg_receivewal. Восстановление с использованием архива. Очистка архива.	6 сам.
1.4	Лаб 1. Логическое резервирование	4 лаб.
1.5	Лаб 2. Базовая резервная копия	4 лаб.
1.6	Лаб 3. Архив журнала предзаписи	4 лаб.
2.	Раздел 2 Репликация. Кластерные технологии	
2.1	Лек. 3 Физическая репликация. Переключение на реплику. Возвращение в строй бывшего мастера. Логическая репликация. Публикации и подписки.	2 л
2.2	Сам. Сценарии использования. Использование физической репликации. Использование логической репликации.	3 сам.
2.3	Лаб 4. Физическая репликация	4 лаб.
2.4	Лаб 5. Переключение на реплику	4 лаб.
2.5	Лаб 6. Логическая репликация	4 лаб.
2.6	Лек. 4. Кластерные технологии Ожидания от кластера. Высокая доступность. Отказоустойчивость. Масштабируемость. Согласованность данных. Средства реализации. Решения с реализацией внутри PostgreSQL.	2 л
2.7	Сам. Внешние по отношению к PostgreSQL средства кластеризации: агент, управляющий, консенсус, точка входа. Примеры: Stolon, Patroni, Corosync/Pacemaker.	3 сам.

17.4 Наименование дисциплины - Использование Python для DevOps

Лекций – 8 ч, лабораторных -26 ч, сам. раб.-24

№ п/п	Наименование и краткое содержание	Объем, часов
1	2	3
1	Введение в скрипты Python для DevOps	2 л., 4 лаб., 2 сам.
2	Сценарии Python: даты, классы и коллекции	2 л., 6 лаб., 4 сам.
3	Сценарии Python: файлы, наследование и базы данных	2 л., 8 лаб., 4 сам.
4	DevOps и автоматизация сборки с помощью Python	2 л., 8 лаб., 4 сам.

17.5 Наименование дисциплины - Администрирование Linux

Лекций – 4 ч, лабораторных -24 ч, сам. раб.-10

№ п/п	Наименование и краткое содержание	Объем, часов
1	Российские операционные системы. Особенности. Средства виртуализации. Установка Astra Linux	1 лек. 4 лаб.

№ п/п	Наименование и краткое содержание	Объем, часов
2	Работа в Astra Linux в пользовательском режиме. Авторизация и вход в систему. Режимы работы. Завершение работы. Настройка рабочего пространства пользователя. Работа в офисном пакете LibreOffice. Работа с мультимедийными файлами	1 лек. 4 лаб. 2 сам.
3	Основы работы в Astra Linux.. Основные команды в терминале. Работа с файлами. Планировщик задач. Файловый менеджер Midnight Commander	1 лек. 4 лаб. 2 сам.
4	Базовое администрирование Astra Linux. Управление пользователями. ACL (Access Control List)	1 лек. 4 лаб. 2 сам.
5	Разграничение доступа в Astra Linux. Управление доступом. ACL (Access Control List)	4 лаб. 2 сам.
6	Сетевые службы предприятия. Сетевые сервисы. DHCP. DNS. Web сервер. File сервер. ADL	4 лаб. 2 сам.

17.6 Наименование дисциплины - Системы виртуализации и контейнеризации (Docker)

Лекции-4 ч., 26 ч. лаб. раб., сам. раб. -14 ч.

№ п/п	Наименование и краткое содержание	Объем, часов
1.	Введение контейнеризацию и виртуализацию для облачных вычислительных систем	2 л. 2 сам.
2.	Платформа Docker	2 л. 8 лаб. 4 сам.
3.	Платформы управления многоконтейнерной вычислительной инфраструктурой	8 лаб. 4 сам.
4.	Решения оркестрации в платформе Docker: Docker Machine, Docker Compose, Docker Swarm	10 лаб. 4 сам.

17.7 Наименование дисциплины - Системы непрерывной интеграции (Jenkins)

Лекции -4 ч., лаб. Раб.- 24 ч., сам. Раб. -12 ч.

№ п/п	Наименование и краткое содержание	Объем, часов
1.	Основы непрерывной интеграции и непрерывной поставки (CI/CD)	2 л. 8 лаб. 4 сам.
2.	Предметно-ориентированный язык Jenkins	2 л. 8 лаб. 4 сам.
3.	Управление Jenkins. Deployment. Интеграции	1 л.

		8 лаб. 4 сам.

17.8 Наименование дисциплины - Системы управления конфигурацией. Инфраструктура как код (Ansible)

Лекции-4 ч., лаб. раб.- 26 ч., сам. раб. -12 ч.

№ п/п	Наименование и краткое содержание	Объем, часов
1.	Система управления конфигурациями Ansible	4 л. 8 лаб. 2 сам.
2.	Основы playbooks	1 л. 10 лаб. 8 сам.
3.	Модули ресурсов. Получение структурированного вывода	1 л. 8 лаб. 4 сам.

17.9 Наименование дисциплины - Системы оркестрации (Kubernets)

Лекции-4 ч., лаб. раб.- 24 ч., сам. раб. - 8 ч.

№ п/п	Наименование и краткое содержание	Объем, часов
1.	Общие сведения о составляющих Kubernetes: container runtime, kubernetes control plane, etcd	1 л. 6 лаб. 2 сам.
2.	Общие концепции: pods, deployments/replica sets, jobs. Использование kubectl.	1 л. 6 лаб. 2 сам.
3.	Анатомия Kubermetes: kubelet, kube api server, etcd, kube proxy, core dns. Сервисы, конфигмапы, секреты.	1 л. 6 лаб. 2 сам.
4.	Развертывание кластера при помощи kubeadm. Подключение рабочих узлов	1 л. 6 лаб. 2 сам.

17.10 Проектный практикум

Задание на проект:

1. Подготовить облачную инфраструктуру на базе облачного провайдера.
2. Запустить и сконфигурировать Kubernetes кластер.
3. Установить и настроить систему мониторинга.
4. Настроить и автоматизировать сборку тестового приложения с

использованием Docker-контейнеров.

5. Настроить CI для автоматической сборки и тестирования.
6. Настроить CD для автоматического развёртывания приложения.

17.11 Практика /Стажировка

Целью практики является закрепление знаний и умений в области методологии DevOps для активного взаимодействия специалистов по разработке со специалистами по информационно-технологическому обслуживанию и взаимной интеграции их рабочих процессов для обеспечения качества продукта.

Задачами практики являются:

- закрепление и расширение теоретических знаний и умений, приобретенных в предшествующий период теоретического обучения;
- овладение профессиональными знаниями и умениями в области непрерывной разработки программных продуктов.
- приобретение практического опыта разработки программного обеспечения.

Основные этапы практики:

№	Содержание работ	Часы
1	Подготовительный этап	4
1.1	Прохождение инструктажа по программе практики, формированию комплекта документов, оформлению дневника практики, подготовке и процедуре защиты отчета по практике, выдача индивидуального задания и графика его выполнения	
2	Аналитический	10
2.1	Исследование жизненного цикла ПО. Изучение роли DevOps-инженера в проекте разработки и внедрения ПО	
3	Проектный	16
3.1	Распределение ролей в проектной команде разработки программного обеспечения; анализ требований к программному обеспечению; проектирование программного обеспечения; настройка рабочего окружения, подготовка и запуск Docker-контейнеров; управление конфигурацией с использованием Ansible; применение выбранных языков программирования для написания программного кода; использование выбранной среды программирования и средств системы управления базами данных.	

4	Отчетный этап	6
4.1	Анализ проделанной работы, подготовка отчетной документации, презентации отчета к защите	
4.2	Промежуточная аттестация по практике	
	итого	36

18. Учебно-тематический план Программы определяет тематическое содержание, последовательность разделов и (или) тем и их трудоемкость.

Наименование дисциплин		Количество часов		
		лекции	лаб. раб	СРС
1.	Командные коммуникации и системы управления задачами	4	16	8
2.	Системы контроля версий	4	16	8
3.	Организация высокой доступности БД	4	16	8
4.	Использование Python для DevOps	4	24	8
5.	Администрирование Linux	4	24	8
6.	Системы виртуализации и контейнеризации (Docker)	4	24	10
7.	Системы непрерывной интеграции (Jenkins)	4	20	10
8.	Системы управления конфигурацией. Инфраструктура как код (Ansible)	2	20	10
9.	Системы оркестрации (Kubernets)	4	24	8
10.	Проектный практикум	-	32	8
11.	Практика / Стажировка			36
12.	**Итоговая аттестация (Демонстрационный экзамен)**			**6**

X. Формы аттестации

19. Слушатели, успешно выполнившие все элементы учебного плана, допускаются к итоговой аттестации.

Итоговая аттестация по Программе проводится в форме демонстрационного экзамена.

20. Лицам, успешно освоившим Программу (в области непрерывной разработки высоконагруженных программных систем, необходимых для выполнения нового вида профессиональной деятельности) и прошедшим итоговую аттестацию в рамках проекта «Цифровые кафедры», выдается документ о квалификации: диплом о профессиональной переподготовке.

При освоении ДПП ПП параллельно с получением высшего образования диплом о профессиональной переподготовке выдается не ранее получения соответствующего документа об образовании и о квалификации (за

исключением лиц, имеющих среднее профессиональное или высшее образование).

21. Лицам, не прошедшим итоговую аттестацию или получившим на итоговой аттестации неудовлетворительные результаты, а также лицам, освоившим часть Программы и (или) отчисленным из ФГБОУ ВО КГЭУ, выдается справка об обучении или о периоде обучения по образцу, самостоятельно устанавливаемому ФГБОУ ВО КГЭУ.

XI. Оценочные материалы

22. Контроль знаний, полученных слушателями при освоении разделов (модулей) Программы, осуществляется в следующих формах:

- текущий контроль успеваемости - обеспечивает оценивание хода освоения разделов Программы, проводится в форме кейсовых заданий и тестов;

- промежуточная аттестация - завершает изучение отдельного модуля Программы, проводится в форме экзамена или Зачета с оценкой;

- итоговая аттестация – завершает изучение всей программы.

23. В ходе освоения Программы каждый слушатель выполняет следующие отчетные работы:

23.1 Наименование дисциплины - Командные коммуникации и системы управления задачами

№ п/п	Наименование раздела	Задание	Критерии оценки
1.	Системы управления проектами (Jira) Проекты. Задачи. Дорожные карты.	Лабораторная работа Создание проекта. Управление задачами.	Макс. оценка – 30 баллов. Мин. оценка –20 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания лабораторной работы. Минимальная оценка выставляется, если студент выполнил необходимый минимум.

№ п/п	Наименование раздела	Задание	Критерии оценки
2.	Agile-разработка с помощью Jira. Scrum, Agile, Kanban доски.	Лабораторная работа Применение scrum с помощью Jira Software. Работа с Kanban доской.	Макс. оценка – 30 баллов. Мин. оценка –20 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания лабораторной работы. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
	Промежуточная аттестация	Тестирование	0-59 – неудовлетворительно 60-70 – удовлетворительно 71-85- хорошо 86-100 - отлично

23.2 Наименование дисциплины - Системы контроля версий

№ п/п	Наименование раздела	Задание	Критерии оценки
1.	Системы контроля версий Git	Тест.	0-59 – неудов. 60-70 – удовл. 71-85- хорошо 86-100 - отлично
2.	Сервис GitLab.	Лабораторные работы 1. Создание репозитория на GitLab. Создание SSH-ключа для подключения к репозиторию. Работа с удалённым репозиторием. 2. Командная работа в GitLab. Работа с ветками. Сравнение версий и отмена изменений.	Макс. оценка – 15 баллов. Мин. оценка – 10 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
3.	CI/CD технология непрерывной интеграции и доставки	Лабораторные работы 1. Знакомство с CD (Continuous delivery). 2. Continuous integration с использованием GitLab CI.	Макс. оценка – 15 баллов. Мин. оценка – 10 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил

			необходимый минимум.
	Промежуточная аттестация	Ответы на вопросы билета	0-59 – неудов. 60-70 – удовл. 71-85- хорошо 86-100 - отлично

23.3 Наименование дисциплины - Организация высокой доступности БД

№ п/п	Наименование раздела	Задание	Критерии оценки
1.	Резервное копирование	Практические занятия 1. Логическое резервирование 2. Базовая резервная копия 3. Архив журнала предзаписи	Макс. оценка – 10 баллов. Мин. оценка – 6 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
2.	Репликация. Кластерные технологии.	Практические занятия 4. Физическая репликация 5. Переключение на реплику 6. Логическая репликация	Макс. оценка – 10 баллов. Мин. оценка – 6 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
	Промежуточная аттестация	Ответы на вопросы билета	0-59 – неудов. 60-70 – удовл. 71-85- хорошо 86-100 - отлично

23.4 Использование Python для DevOps

№ п/п	Наименование раздела	Задание	Критерии оценки
1.	Сценарии Python: даты, классы и коллекции	Лабораторная работа. Расчет переменной	0-59 – неудовлетворительно 60-70 – удовлетворительно 71-85- хорошо 86-100 - отлично
2.	Сценарии Python: файлы, наследование и базы данных.	Лабораторная работа. Скрипты поиска локальных репозиторий в текущей директории	0-59 – неудовлетворительно 60-70 – удовлетворительно 71-85- хорошо 86-100 - отлично
3.	DevOps и автоматизация сборки с помощью Python	Лабораторная работа. Pull request	0-59 – неудовлетворительно 60-70 – удовлетворительно 71-85- хорошо 86-100 - отлично
	Промежуточная аттестация	Контрольная работа	Заключается в написании контрольной работы по всем темам курса Макс. оценка – 20 баллов. Мин. оценка – 12 баллов. Максимальную оценку студент получает, если на все задания выполнены правильно. Минимальная оценка выставляется, если задания выполнены частично.

23.5 Наименование дисциплины - Администрирование Linux

№ п/п	Наименование раздела	Задание	Критерии оценки
1.	Российские операционные системы	Лабораторная работа: Средства виртуализации. Установка Astra Linux	Макс. оценка – 10 баллов. Мин. оценка – 6 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент

№ п/п	Наименование раздела	Задание	Критерии оценки
			выполнил необходимый минимум.
2.	Работа в Astra Linux в пользовательском режиме	Лабораторная работа: Авторизация и вход в систему. Режимы работы. Завершение работы. Настройка рабочего пространства пользователя. Работа в офисном пакете LibreOffice. Работа с мультимедийными файлами	Макс. оценка – 10 баллов. Мин. оценка – 6 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
3.	Основы работы в Astra Linux	Лабораторная работа: Основные команды в терминале. Работа с файлами. Планировщик задач. Файловый менеджер Midnight Commander	Макс. оценка – 10 баллов. Мин. оценка – 6 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
4.	Базовое администрирование Astra Linux	Лабораторная работа: Управление пользователями. ACL (Access Control List)	Макс. оценка – 10 баллов. Мин. оценка – 6 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
5.	Разграничение доступа в Astra Linux	Лабораторная работа: Управление доступом. ACL (Access Control List)	Макс. оценка – 10 баллов. Мин. оценка – 6 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
6.	Сетевые службы предприятия	Лабораторная работа: Сетевые сервисы. DHCP. DNS. Web сервер. File сервер. ADL	Макс. оценка – 10 баллов. Мин. оценка – 6 баллов. Максимальная оценка выставляется, если студент

№ п/п	Наименование раздела	Задание	Критерии оценки
			выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
	Промежуточная аттестация	Тестирование	0-59 – неудов. 60-70 – удовл. 71-85- хорошо 86-100 - отлично

23.6 Системы виртуализации и контейнеризации (Docker)

№ п/п	Наименование раздела	Задание	Критерии оценки
1.	Платформа Docker	Создание Docker Swarm кластера	Макс. оценка – 20 баллов. Мин. оценка – 15 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
2.	Платформы управления многоконтейнерной вычислительной инфраструктурой	Создание кластера мониторинга, состоящего из стека микросервисов	Макс. оценка – 20 баллов. Мин. оценка – 15 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
3.	Решения оркестрации в платформе Docker: Docker Machine, Docker Compose, Docker Swarm	Выполнить на лидере Docker Swarm кластера команду и дать письменное описание её функционала	Макс. оценка – 20 баллов. Мин. оценка – 15 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
	Промежуточная аттестация	Контрольная работа Тестирование	Заключается в написании контрольной работы по всем темам курса Макс. оценка – 20 баллов. Мин. оценка – 12 баллов. Максимальную оценку студент получает, если на все задания выполнены правильно. Минимальная оценка выставляется, если задания

№ п/п	Наименование раздела	Задание	Критерии оценки
			выполнены частично. Тестирование 0-59 – неудов. 60-70 – удовл. 71-85- хорошо 86-100 - отлично

23.7 Системы непрерывной интеграции (Jenkins)

№ п/п	Наименование раздела	Задание	Критерии оценки
1.	Основы непрерывной интеграции и непрерывной поставки (CI/CD)	Настройка jenkins	Макс. оценка – 20 баллов. Мин. оценка – 15 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
2.	Предметно-ориентированный язык Jenkins	Запуск ansible-playbook через Freestyle Job	Макс. оценка – 20 баллов. Мин. оценка – 15 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
3.	Управление Jenkins. Deployment. Интеграции	Разработка скрипта на groovy	Макс. оценка – 20 баллов. Мин. оценка – 15 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
	Промежуточная аттестация	Контрольная работа	Заключается в написании контрольной работы по всем темам курса Макс. оценка – 20 баллов. Мин. оценка – 12 баллов. Максимальную оценку студент получает, если на все задания выполнены правильно. Минимальная оценка выставляется, если задания выполнены частично.

23.8 Системы управления конфигурацией. Инфраструктура как код (Ansible)

№ п/п	Наименование раздела	Задание	Критерии оценки
1.	Система управления конфигурациями Ansible	Установка ansible	Макс. оценка – 20 баллов. Мин. оценка – 15 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
2.	Основы playbooks	Запуск playbook на окружении из prod.yml	Макс. оценка – 20 баллов. Мин. оценка – 15 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
3.	Модули ресурсов. Получение структурированного вывода	При помощи ansible-vault расшифруйте все зашифрованные файлы с переменными	Макс. оценка – 20 баллов. Мин. оценка – 15 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
	Промежуточная аттестация	Тестирование	0-59 – неудов. 60-70 – удовл. 71-85- хорошо 86-100 - отлично

23.9 Системы оркестрации (Kubernets)

№ п/п	Наименование раздела	Задание	Критерии оценки
1.	Общие сведения о составляющих Kubernetes: container runtime, kubernetes control plane, etcd	Создайте модуль, содержащий сервер nginx	Макс. оценка – 15 баллов. Мин. оценка – 10 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.

№ п/п	Наименование раздела	Задание	Критерии оценки
2.	Общие концепции: pods, deployments/replica sets, jobs. Использование kubectl.	Создайте метки в Kubernetes	Макс. оценка – 15 баллов. Мин. оценка – 10 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
3.	Анатомия Kubermetes: kubelet, kube api server, etcd, kube proxy, core dns. Сервисы, конфигмапы, секреты.	Использование конфигураций набора реплик	Макс. оценка – 15 баллов. Мин. оценка – 10 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
4.	Развертывание кластера при помощи kubeadm. Подключение рабочих узлов	Создать развертывание для веб-сервера nginx	Макс. оценка – 15 баллов. Мин. оценка – 10 баллов. Максимальная оценка выставляется, если студент выполнил полностью все задания. Минимальная оценка выставляется, если студент выполнил необходимый минимум.
	Промежуточная аттестация	Тестирование	0-59 – неудов. 60-70 – удовл. 71-85- хорошо 86-100 - отлично

23.10 Проектный практикум

1. Подготовить облачную инфраструктуру на базе облачного провайдера.
2. Запустить и сконфигурировать Kubernetes кластер.
3. Установить и настроить систему мониторинга.
4. Настроить и автоматизировать сборку тестового приложения с использованием Docker-контейнеров.
5. Настроить CI для автоматической сборки и тестирования.
6. Настроить CD для автоматического развёртывания приложения.

24. Текущий контроль. Перечень примерных заданий

24.1 Командные коммуникации и системы управления задачами

Тема 1 Системы управления проектами (Jira) Проекты. Задачи. Дорожные карты.

Лабораторная работа 1 Системы управления проектами Jira.

1. Создание и настройка проекта.
2. Создать задачу.
3. Создать подзадачи.
4. Пригласить участников команды.
5. Управление задачами

Тема 2 Agile-разработка с помощью Jira. Scrum, Agile, Kanban доски.

Лабораторная работа 2 Agile-разработка с помощью Jira.

1. Scrum разработка с помощью Jira Software
2. Создать kanban доску с помощью Jira Software
3. Создать эпики в Jira Software
4. Спринты в Jira Software
5. Работа с версиями с помощью Jira Software

24.2 Системы контроля версий

Тема 1 Системы контроля версий Git

Тест

1. Какой тип систем контроля версий сейчас используется чаще всего?

- ○ Системные
- ○ Централизованные
- ○ Локальные
- ◉ Распределенные

2. Выберите действие, которое выполняет команда git

git push	Ответ 1	Отправка изменений на удаленный репозиторий для синхронизации
git add	Ответ 2	Выбор файлов для фиксации изменений
git status	Ответ 3	Просмотр статуса репозитория git
git commit	Ответ 4	Фиксация изменений в репозитории
git init	Ответ 5	Создание репозитория git

3. Какая команда git используется для клонирования удаленного репозитория на локальный компьютер?

- git download
- git checkout
- git clone
- git copy

4. Какая команда git используется для получения изменений с удаленного репозитория на локальный компьютер?

- git push
- git pull
- git sync
- git clone

Тема 2 Сервис GitLab.

Лабораторные работы:

1. Создание репозитория на GitLab. Создание SSH-ключа для подключения к репозиторию. Работа с удалённым репозиторием.

1. Создать учетную запись и рабочую группу
2. Создать репозиторий
3. Загрузить файлы
4. Добавить ключи авторизации.

2. Командная работа в GitLab. Работа с ветками. Сравнение версий и отмена изменений.

1. Добавление файлов в репозиторий git
2. Добавление пользователей в проект
3. Отслеживание изменений в git
4. Фиксация изменений
5. Создние баг-репорта.

Тема 3 CI/CD технология непрерывной интеграции и доставки

1. Знакомство с CD (Continuous delivery).

1. Создаём .gitlab-ci.yml
2. Задаём подготовительные команды
3. Указываем этапы
4. Описываем джобы и задаём команду

2. Continuous integration с использованием GitLab CI

1. Запуск вручную
2. Продолжение при провале
3. Выполнение джобов по условию
4. Запуск по расписанию
5. Серия джобов

24. 3 Организация высокой доступности БД

Тема 1 Резервное копирование

Практические задания:

1. Логическое резервирование

1. На первом сервере создайте несколько баз данных. В них создайте различные объекты (например, таблицы, представления, индексы).
2. Сделайте копию только глобальных объектов кластера с помощью утилиты pg_dumpall.
3. Сделайте копии каждой базы данных кластера с помощью утилиты pg_dump в параллельном режиме.
4. Полностью восстановите кластер на другом сервере, используя созданные резервные копии.
5. Попробуйте подобрать такие данные и параметры команды COPY, чтобы созданную копию таблицы невозможно было загрузить.

2. Базовая резервная копия

1. В первом кластере создайте табличное пространство и базу данных с таблицей в этом пространстве.
2. Сделайте базовую резервную копию кластера с помощью pg_basebackup в формате tar со сжатием.
3. Разверните второй кластер из этой резервной копии. Табличное пространство разместите в другом каталоге, изменив файл tablespace_mapping.
4. Запустите второй сервер и проверьте его работоспособность.
5. Удалите базу данных и табличное пространство в обоих кластерах.

3. 3. Архив журнала предзаписи

1. На первом сервере создайте базу данных и в ней таблицу с какими-нибудь данными.
2. Настройте непрерывное архивирование.
3. Сделайте базовую резервную копию кластера с помощью pg_basebackup, без файлов журнала.
4. Вставьте еще несколько строк в таблицу и убедитесь, что текущий сегмент WAL попал в архив.
5. Восстановите второй сервер из резервной копии, указав в параметрах одну только команду восстановления. Проверьте, что в таблице восстановились все строки.
6. Остановите второй сервер и восстановите его повторно из той же резервной копии, на этот раз указав целевую точку восстановления immediate. Проверьте таблицу.

Тема 2 Репликация. Кластерные технологии.

Практические задания:

4. Физическая репликация

1. Настройте физическую потоковую репликацию между двумя серверами в синхронном режиме, без обратной связи.
2. Проверьте работу репликации. Убедитесь, что при остановленной реплике фиксация не завершается.
3. Воспроизведите ситуацию, при которой запрос на реплике прерывается из-за очистки версий строк на мастере.
4. Запретите откладывать применение конфликтующих изменений; проверьте, что запрос отменяется сразу.
5. Включите обратную связь и убедитесь, что очистка на мастере не прерывает выполнение запроса.
6. Остановите реплику и убедитесь, что слот препятствует очистке. Удалите слот, чтобы очистка возобновилась.

5. Переключение на реплику

1. Выполните необходимую настройку мастера и реплики для потоковой репликации с использованием слота, без непрерывного архивирования.
2. Имитируйте сбой основного сервера и переключитесь на реплику.
3. Верните в строй бывший основной сервер, выполнив резервную копию с нового мастера и настроив необходимые параметры.

   Убедитесь, что репликация работает и использует слот.
4. Переключитесь на новую реплику, чтобы бывший мастер снова стал основным сервером.

6. Логическая репликация

1. Создайте две базы данных на одном сервере.
   В первой базе данных создайте таблицу с первичным ключом и добавьте в нее несколько строк.
2. Перенесите определение созданной таблицы во вторую базу данных с помощью логической резервной копии.
3. Настройте логическую репликацию таблицы из первой базы данных во вторую.
4. Проверьте работу репликации.
5. Удалите подписку.

24. 4 Использование Python для DevOps

Лабораторная работа 1.

```python
#!/usr/bin/env python3
a = 1
b = '2'
c = a + b
```

```
* Какое значение будет присвоено переменной c?
* Как получить для переменной c значение 12?
* Как получить для переменной c значение 3?
```

- Какое значение будет присвоено переменной c? - Интерпретатор выдаст ошибку - попытка сложения целочисленного значения со строковым.
- Как получить для переменной c значение 12?

```
#!/usr/bin/env python3
a = 1
b = '2'
c = str(a) + b # Если допустим строковый результат
c = (a + int(b)) * int(b) * int(b) # Если допустим только целочисленный результат
```

- Как получить для переменной c значение 3?

```
#!/usr/bin/env python3
a = 1
b = '2'
c = a + int(b)
```

Лабораторная работа 2.

Мы устроились на работу в компанию, где раньше уже был DevOps Engineer. Он написал скрипт, позволяющий узнать, какие файлы модифицированы в репозитории, относительно локальных изменений. Этим скриптом недовольно начальство, потому что в его выводе есть не все изменённые файлы, а также непонятен полный путь к директории, где они находятся. Как можно доработать скрипт ниже, чтобы он исполнял требования вашего руководителя?

```python
#!/usr/bin/env python3

import os
```

```
bash_command = ["cd ~/netology/sysadm-homeworks", "git status"]
result_os = os.popen(' && '.join(bash_command)).read()
is_change = False
for result in result_os.split('\n'):
    if result.find('modified') != -1:
        prepare_result = result.replace('\tmodified:   ', '')
        print(prepare_result)
        break

```
```
#!/usr/bin/env python3
import os

basedir = "~/netology/sysadm-homeworks"
bash_command = [f"cd {basedir}", "git status "]
result_os = os.popen(' && '.join(bash_command)).read()
for result in result_os.split('\n'):
    if result.find('modified') != -1:
        prepare_result = result.replace('modified:', basedir)
        print(prepare_result)
% python3 test.py
    ~/netology/sysadm-homeworks   README.md
```

Лабораторная работа 3.

Доработать скрипт выше так, чтобы он мог проверять не только локальный репозиторий в текущей директории, а также умел воспринимать путь к репозиторию, который мы передаём как входной параметр. Мы точно знаем, что начальство коварное и будет проверять работу этого скрипта в директориях, которые не являются локальными репозиториями.

```
#!/usr/bin/env python3

import os
import sys

basedir = ""
try:
    basedir = sys.argv[1]
except:
    print("Incorrect repository path")

if basedir != "":
        bash_command = [f"cd {basedir}",  "git status "]
        result_os1 = os.listdir(basedir);

        if result_os1.__contains__(".git"):
                result_os = os.popen(' && '.join(bash_command)).read()
                for result in result_os.split('\n'):
                    if result.find('modified') != -1:
                        prepare_result = result.replace('modified:', basedir)
                        print(prepare_result)
        else:
                print("There is no git repository on the entered path")
% python3 test2.py Documents/DevOps/git/devops-netology
    Documents/DevOps/git/devops-netology   README.md
% python3 test2.py Documents/DevOps/
There is no git repository on the entered path
```

Лабораторная работа 4.

Наша команда разрабатывает несколько веб-сервисов, доступных по http. Мы точно знаем, что на их стенде нет никакой балансировки, кластеризации, за DNS прячется конкретный IP сервера, где установлен сервис. Проблема в том, что отдел, занимающийся нашей инфраструктурой очень часто меняет нам сервера, поэтому IP меняются примерно раз в неделю, при этом сервисы сохраняют за собой DNS имена. Это бы совсем никого не беспокоило, если бы несколько раз сервера не уезжали в такой сегмент сети нашей компании, который недоступен для разработчиков. Мы хотим написать скрипт, который опрашивает веб-сервисы, получает их IP, выводит информацию в стандартный вывод в виде: <URL сервиса> - <его IP>. Также, должна быть реализована возможность проверки текущего IP сервиса с его IP из предыдущей проверки. Если проверка будет провалена - оповестить об этом в стандартный вывод сообщением: [ERROR] <URL сервиса> IP mismatch: <старый IP> <Новый IP>. Будем считать, что наша разработка реализовала сервисы: drive.google.com, mail.google.com, google.com.

```
#!/usr/bin/env python3

import socket
from string import whitespace

hosts = ["drive.google.com", "mail.google.com", "google.com"]
fileList = []

with open('host_test.log') as file:
    for f in file:
        fileList.append(f)

with open('host_test.log', 'w+') as file:
    for i in hosts:
        result = socket.gethostbyname(i)
        added = 0
        for y in fileList:
            inList = y.find(" {}".format(i))
            if (inList != -1):
                ipstr=y.replace('\n', '').split("  ")[1].translate({None:
whitespace})
                if (ipstr == result):
                    print(" {}  {}\n".format(i, result))
                    file.write(" {}  {}\n".format(i, result))
                    added = 1
                    break
                else:
                    print("[ERROR] {} IP mismatch: {}  {}\n".format(i, ipstr,
result))
                    file.write("[ERROR] {} IP mismatch: {}  {}\n".format(i, ipstr,
result))
                    added = 1
                    break
        if (added == 0):
            print(" {}  {}\n".format(i, result))
            file.write(" {}  {}\n".format(i, result))
% python3 host_test.py
 drive.google.com  142.250.185.78
```

```
 mail.google.com  142.250.186.69

 google.com  142.250.184.238

 % python3 host_test.py
 drive.google.com  142.250.185.78

[ERROR] mail.google.com IP mismatch: 142.250.186.69  142.250.186.101

 google.com  142.250.184.238

 % python3 host_test.py
 drive.google.com  142.250.185.78

 mail.google.com  142.250.186.101

 google.com  142.250.184.238
```

Лабораторная работа 5.

Так получилось, что мы очень часто вносим правки в конфигурацию своей системы прямо на сервере. Но так как вся наша команда разработки держит файлы конфигурации в github и пользуется gitflow, то нам приходится каждый раз переносить архив с нашими изменениями с сервера на наш локальный компьютер, формировать новую ветку, коммитить в неё изменения, создавать pull request (PR) и только после выполнения Merge мы наконец можем официально подтвердить, что новая конфигурация применена. Мы хотим максимально автоматизировать всю цепочку действий. Для этого нам нужно написать скрипт, который будет в директории с локальным репозиторием обращаться по API к github, создавать PR для вливания текущей выбранной ветки в master с сообщением, которое мы вписываем в первый параметр при обращении к ру-файлу (сообщение не может быть пустым). При желании, можно добавить к указанному функционалу создание новой ветки, commit и push в неё изменений конфигурации. С директорией локального репозитория можно делать всё, что угодно. Также, принимаем во внимание, что Merge Conflict у нас отсутствуют и их точно не будет при push, как в свою ветку, так и при слиянии в master. Важно получить конечный результат с созданным PR, в котором применяются наши изменения.

Ваш скрипт:

```
#!/usr/bin/env python3

import os
import subprocess
import sys
import time
import requests
import json
import re
from datetime import datetime
```

```
def git_exec(command):
    print(command)
    if command.find("git commit") >= 0:
        command_splitted = ["git", "commit", "-m"]
        command_splitted.append(command.split('git commit -m ')[1])
    else:
        command_splitted = command.split()
    command_e = subprocess.Popen(command_splitted, stdout=subprocess.PIPE,
                                 stderr=subprocess.STDOUT, cwd=resolved_path,
text=True)
    e = command_e.communicate()[0].split('\n')[0]
    if e.find('fatal:') >= 0:
        print(
            f'В папке {resolved_path} нет git репозитория. Поищите в другой папке.')
        exit()
    return e


token = ""
if token == "":
    print(f"""
\t!!! Задайте токе в теле скрипта, переменная "token" !!!

\thttps://docs.github.com/en/authentication/keeping-your-account-and-data-
secure/creating-a-personal-access-token
""")
    exit()

try:
    if sys.argv[1]:
        message = " ".join(sys.argv[1:])
except IndexError:
    print(
        f"Введите сообщение для реквеста, например:\n\n\t{sys.argv[0]} поправил
конфиг, была ошибка синтаксиса\n")
    exit()

path = "./"
resolved_path = os.path.normpath(os.path.abspath(
    os.path.expanduser(os.path.expandvars(path))))

try:
    origin_push_url = git_exec("git remote get-url --push origin")
except FileNotFoundError:
    print(
        f'Не могу найти папку {path}'
    )
    exit()

if origin_push_url.find('fatal:') >= 0:
    print(
        f'В папке {resolved_path} нет git репозитория. Поищите в другой папке.')
    exit()

gh_acc, gh_repo = re.split('git@github.com:|/|.git', origin_push_url)[1:3]

repo_url = f'https://api.github.com/repos/{gh_acc}/{gh_repo}'

headers = {"Authorization": f"token {token}",
           "Accept": "application/vnd.github.v3+json"}

git_status = subprocess.Popen(["git", "status", "--porcelain"],
stdout=subprocess.PIPE,
```

```
                                stderr=subprocess.STDOUT, cwd=resolved_path,
text=True).communicate()[0].split('\n')

cur_time = datetime.now()
branch_name = f"""{datetime.strftime(cur_time, "%Y-%m-%d_%H%M%S")}-config-local-
edit"""
date_commit_text = datetime.strftime(cur_time, "%Y-%m-%d %H:%M:%S")

# exit()
if len(git_status) > 1 or git_status[0] != '':
    git_exec(f"git checkout -b {branch_name}")
    git_exec(f"git add .")
    git_exec(f"git commit -m 'config local edit at {date_commit_text}'")
    git_exec(f"git push --set-upstream origin {branch_name}")
    r = requests.get(f"{repo_url}/branches/{branch_name}", headers=headers)
    git_exec(f"git checkout main")
    while r.status_code >= 300:
        r = requests.get(f"{repo_url}/branches/{branch_name}", headers=headers)
        print(f'Репозитой пока не создан. Ответ GitHub: {r}, {r.content}')
        time.sleep(1)
    payload = {"title": branch_name, "body": message,
               "head": branch_name, "base": "main"}
    r = requests.post(f"{repo_url}/pulls", headers=headers,
                      data=json.dumps(payload))
    if r.status_code >= 300:
        print(
            f"Что-то пошло не так! Ответ GitHub API на создание Pull Request:
{r}\n\n{r.content}\n")
        exit()
    else:
        print(f'Ответ GitHub на создание Pull Request: {r}')
        git_exec(f"git branch -D {branch_name}")
    pull_req_merge_url = f"{r.json()['url']}/merge"
    payload = {"commit_title": f"MERGED {branch_name} into main"}
    r = requests.put(pull_req_merge_url, headers=headers,
                     data=json.dumps(payload))
    if r.status_code >= 300:
        print(
            f"Что-то пошло не так! Ответ GitHub API на мерж Pull Request:
{r}\n\n{r.content}\n")
        exit()
    else:
        print(f'Ответ GitHub на мерж Pull Request: {r}')
        git_exec(f"git push origin -d {branch_name}")
        print(f'\nЗагружаем изменения main:\n')
        os.popen(f"cd {resolved_path} && git pull").read()
        print(f'\n')
```

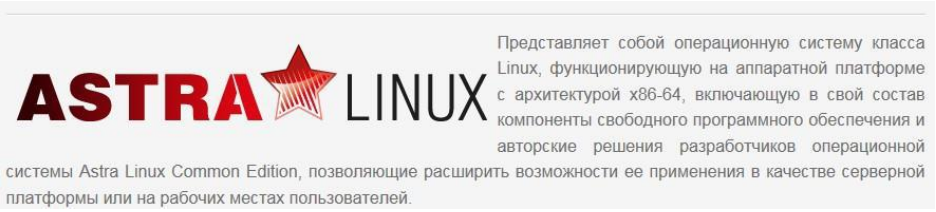
24. 5 Администрирование Linux

Тема 1 Установка Astra Linux.

Лабораторная работа:

1. Установка операционной системы

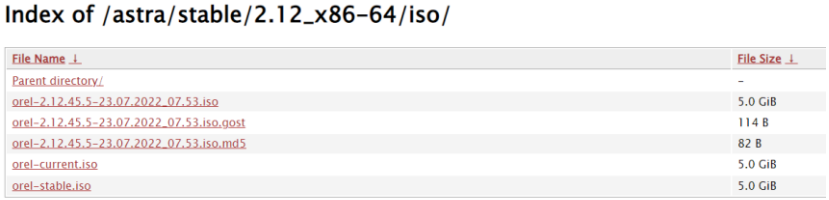
Устанавливаемая операционная система Astra Linux Common Edition.



Представляет собой операционную систему класса Linux, функционирующую на аппаратной платформе с архитектурой x86-64, включающую в свой состав компоненты свободного программного обеспечения и авторские решения разработчиков операционной системы Astra Linux Common Edition, позволяющие расширить возможности ее применения в качестве серверной платформы или на рабочих местах пользователей.

2. Скачиваем образ Операционной системы AstraLinux

Версия 2.12. релиз "Орел"
Скачиваем образ установочного диска orel-stable.iso на FTP-сервере
https://dl.astralinux.ru/astra/stable/2.12_x86-64/iso/



3. Последовательность установки.

1.Предварительные действия по созданию виртуальной машины установки.

Внешний вид окон может отличаться, в зависимости от версии ОС
Запустить диспетчер Hyper-V. Выбрать создание виртуальной машины.
Имя виртуальной машины Astra-Linux-Orel, расположение по умолчанию (*если не указана преподавателем другая информация*).
Память 1024 (или больше). Тип используемой сети – частная (название сети задает преподаватель).
Виртуальный жесткий диск создается максимальной емкости, имя диска Astra-Linux-Orel.vhd, расположение размещения по умолчанию (*если иная информация не указана преподавателем*).
Расположение ISO образа для установки операционной системы указывает преподаватель.
После выполнения установки следует запустить на выполнение виртуальною машину.
Для снятия копий экрана при работе с виртуальной машиной следует использовать пункт меню Буфер обмена --> Снять экран (или комбинацию горячих клавиш Ctrl + C).
Изображение копируется в буфер обмена компьютера, затем его можно вставить в отчет (или сразу в документ редактора Word или в графический редактор, выполнить в нем обработку, а затем вставить в отчет).

Установка Astra Linux

Шаг 1. Запуск установщика Сразу же после перезагрузки появится меню выбора способа установки. Выберите «Графическая установка ОС».

Шаг2. Лицензия На первом шаге установщика вам нужно принять лицензионное соглашение разработчиков.

Шаг 3. Переключение раскладки Выберите клавишу, с помощью которой будет переключаться раскладка.

Шаг 4. Начало установки Дождитесь загрузки всех необходимых компонентов.

Шаг 5. Имя компьютера

Введите имя компьютера, оно будет использоваться для обнаружения компьютера в локальной сети. (В качестве имени введите свою фамилию)

Шаг 6. Имя пользователя

Введите имя пользователя или логин который будете использовать для входа в систему. Имя учетной записи администратора операционной системы начинается с маленькой латинской буквы. Может состоять из латинских букв и цифр. (В качестве имени введите свою фамилию и дату установки)

Шаг 7. Пароль пользователя

На этом шаге нужно несколько раз ввести пароль для нового пользователя.

Шаг 8. Часовой пояс

Из предлагаемого программой установки Astra Linux выберите часовой пояс. Соответствующее время станет в дальнейшем использоваться операционной системой.

Шаг 9. Разметка диска

Дальше нам нужно выполнить разметку диска, вы можете выбрать автоматический вариант, и тогда система создаст нужные разделы сама. Но в нашей работе мы рассмотрим настройку разметку вручную. Поэтому выберите «Вручную».

Затем выберите нужный жесткий диск:

Если диск был пуст, подтвердите создание новой таблицы разделов и снова выберите нужный диск.

Шаг 10. Корневой раздел

Нажмите «Создать новый раздел»:

Выберите размер раздела. Минимальный размер 5 Гб, но рекомендуется выделить не менее 20-30 Гб, чтобы вам было достаточно для установки всех программ.

Затем «Начало», чтобы расположить раздел в начале пространства:

Осталось изменить настройки раздела, по умолчанию используется файловая система ext4 и для первого раздела установщик сделает точку монтирования «/» корень.

Опуститесь на «Настройка раздела закончена» и нажмите «Продолжить»:

Шаг 11. Домашний раздел

Домашний раздел создается аналогичным образом. Сначала выберите свободное пространство:

Затем «Создать новый раздел»:

Здесь точка монтирования должна быть /home:

Шаг 12. Раздел подкачки

Подкачка используется, когда в системе не хватает оперативной памяти. Неиспользуемое содержимое ОЗУ сбрасывается на диск в этот раздел. Размер желательно выбрать таким же, как и размер ОЗУ, а файловая система «Раздел подкачки», точку монтирования устанавливать не нужно.

Шаг 13. Завершение разметки

Для продолжения выберите «Завершить разметку и записать изменения на диск»:

Затем подтвердите правильность разметки:

Шаг 14. Установка

Дождитесь пока установка Astra Linux завершиться:

Шаг 15-. Выбор ядра ОС

Выберите версию ядра операционной системы для установки. В нашем случае рекомендуется выбрать Linux - 5.15 - Generic.

Шаг 16. Выбор программного обеспечения

Отметьте программное обеспечение, которое нужно установить вместе с системой:

Затем выберите дополнительные функции, если это необходимо:

Шаг 17. Загрузка и установка ПО

Дождитесь, пока завершиться скачивание и установка выбранных программ. Это самый долгий этап:

Шаг 18. Дополнительные опции

Если нужно, вы можете указать дополнительные опции, например, выключить автоматическую настройку сети.

Шаг 19. Установка загрузчика

Укажите нужно ли устанавливать загрузчик Grub. Если на вашем компьютере установлен только один Linux дистрибутив, то установка загрузчика обязательна.

Шаг 20. Перезагрузка

Установка Astra Linux на жесткий диск завершена и теперь вы можете перезагрузить компьютер, чтобы пользоваться новой системой.

Шаг 21. Загрузчик

Выберите пункт, подчеркнутый по умолчанию в меню загрузчика Grub:

Затем дождитесь завершения загрузки системы:

Шаг 22. Вход

Введите логин и пароль, заданные во время установки системы:

Шаг 23. Готово

Если все было выполнено правильно, вы увидите рабочий стол Astra Linux. Все готово, и теперь вы можете пользоваться своей системой

Стартовая настройка AstraLinux

Обновление системы:

Проведем обновление операционной системы Astra Linux. Для этого зайдем, в Терминал и выполним комманды:

- sudo apt update (Обновление списка пакетов);
- sudo apt full-upgrade (Данная команда выполняет обновление пакетов, а также удаляет или устанавливает новые пакеты, если это потребуется для разрешения зависимостей.);
- sudo apt autoclean (очищает систему от deb пакетов, которые больше не нужны, рекомендуется делать периодически);
- sudo apt autoremove (удаляет не удаленные зависимости от уже удаленных пакетов)

Тема 2 Работа в Astra Linux в пользовательском режиме.

Лабораторная работа:

Скачайте образ операционной системы (Astra_Linux_2.1)

Первого пользователя в системе вы создали при установке (логин: student, пароль: Asdf1234). В этом задании Вам нужно создать несколько новых пользователей, еще несколько групп в системе и добавить пользователей в группы.

Для создания пользователей и групп можно пользоваться любыми изученными средствами.

1. Создайте две группы: **analysts** с заданным "вручную" GID **444** и **managers** с паролем **qwerty123**.
2. Далее создайте трех обычных пользователей: **samuel** с корневым каталогом в **/etc/samuel**, **lily**, сразу при создании включив его в **managers** и **harry** с заданным "вручную" UID **1110**.
3. Добавьте пользователя **harry** в группу **managers** и **analysts**, а пользователя **samuel** в группу **managers**.
4. Назначьте пароль пользователю **samuel** пароль **my_password**.
5. Назначьте пользователю **lily** права администратора (добавить в группу sudo).
6. Отправьте на проверку файлы **/etc/passwd** и **/etc/group**.
7. Удалите созданных пользователей.
8. Смените тему fly на Зеленую

9. Измените системный шрифт по умолчанию на Noto Serif Tamil (тип – Bold, кегль 15, подчеркнутый)

10. Поставьте в качестве обоев для первого рабочего стола любую понравившуюся картинку (можно скачать из Интернета)

11. На панели быстрого запуска поместите ярлык браузера Mozilla Firefox

12. Откройте текстовый редактор LibreOffice Writer.

13. Перед вами окно создания нового текстового документа. Скопируйте любой понравившийся вам текст (но не менее 4 абзацев) и вставьте в документ. Либо наберите текст самостоятельно.

14. В самом начале отдельным абзацем напишите какой-либо заголовок текста (1 предложение. 1 – 6 слов).

15. Определите тип Заглавие для данного заголовка.

16. Выделите Заглавие жирным подчеркнутым шрифтом. Установите гарнитуру PT Serif, кегль 20.

17. Цвет шрифта первого абзаца текста установите розовый. Выделите абзац курсивом. Выровняйте по ширине страницы.

18. Для второго абзаца текста установите гарнитуру – Cousine, основной отступ слева сдвиньте на значение 1, первую строку – на значение 2. Исправьте всю орфографию в данном абзаце (если встречаются имена собственные – нажмите Пропустить, чтобы ни одно слово не осталось подчеркнутым)

19. Для третьего абзаца текста оформите выравнивание текста по правому краю. Сделайте текст зачеркнутым, голубого цвета, гарнитуры Arimo, кегль 8.

20. Четвертый абзац отформатируйте как список с маркерами «галочка». Сделайте выделение текста в нем сиреневым маркером. Превратите половину абзаца в верхний индекс.

21. Сохраните полученный документ. Имя документа - Работа_Writer. Отправьте свою работу для проверки преподавателю.

22. Используя полученные знания о мультимедиа утилитах, нужно открыть содержимое предложенного файла и ввести ответ, «спрятанный» в контенте. Нужный файл располагается по пути /home/student на предложенной виртуальной машине (имя пользователя: student, пароль: Asdf1234).

23. Необходимо определить:

24. Предлагаемую высоту изображения 4.png при печати в мм (только целую часть числа)

25. Значение частоты дискретизации в Гц для файла 7.flac

Тема 3 Основы работы в Astra Linux.

Лабораторная работа:

1. Перейдите в корневой каталог.
2. Создайте пользователя студент выполнив команду adduser student. Войдите в систему под этим пользователем выполнив su student.
3. Скопируйте файл passwd из каталога /etc в домашнюю директорию пользователя student. Задание необходимо выполнить, оставаясь в корневом каталоге.
4. Создайте символическую ссылку symlink на скопированный файл passwd
5. Создайте две жестких ссылки hardlink и hardlink1 на скопированный файл passwd
6. Проверьте что ссылки работают командой cat
7. Удалите оригинал /home/student/passwd
8. Создайте каталог в домашней директории с именем Task2. В подкаталоге Task2 создайте каталог с именем classics.
9. Создайте файл file1 в каталоге classics.
10. Удалите каталог classics вместе с его содержимым.
11. Посмотрите текущее время и дату на вашей системе.
12. Очистите окно терминала

13. Посмотрите, в какой директории вы сейчас находитесь?

14. Перейдите в Вашу домашнюю директорию

15. Посмотрите содержимое вашего рабочего каталога

16. Посмотрите все файлы в текущей директории, включая скрытые

17. Перейдите из текущей директории в директорию /etc

18. Перейдите в домашний каталог и выполните команду ls -a -R > filelog

19. Прикрепите файл /home/student/.bash_history и /home/student/filelog.

20. Выполнить задание: Каждую минуту записывать текущую дату в файл:

21. Авторизуйтесь под пользователем root

22. В домашнем каталоге пользователя /root создайте файл с названием cron.

23. Внутри файла напишите задание для планировщика crontab, которое каждую минуту будет записывать в файл /root/date.cron текущую дату и время, при этом предыдущие записи удаляться не должны.

24. Добавьте созданный файл в планировщик: crontab /root/cron

25. Проверьте добавилось ли задание по расписанию и работает ли оно.

26. Прикрепите файл /var/spool/cron/crontabs/root.

Тема 4 Базовое администрирование Astra Linux

Лабораторная работа:

1. Скачайте образ виртуальной машины (AstraLinuxSE).

2. Разверните ее и войдите в систему (имя пользователя: student; пароль: Asdf1234)

3. Авторизуйтесь под пользователем root. Выполните команды.

```
adduser admin1
usermod -G astra-admin admin1
```

4. Авторизуйтесь под пользователем admin1. Добавьте трех новых пользователей с соответствующими домашними директориями: student7,

student8, student9. Задайте пароли для каждого из них, используйте команду adduser.

5. Создайте группу course и добавьте в нее всех трех пользователей.

6. Для пользователя student7 выставите ограничение: срок действия пароля 5 месяцев(150 дней) и предупреждение об окончании срока действия пароля 9 дней.

7. Заблокируйте пользователя student8, заблокировав его пароль. Проверьте, что блокировка подействовала.

8. Войдите в систему под пользователем student9, находясь в этом же терминале.

9. Войдите в систему под пользователем root. Разблокируйте пользователя student8. Проверьте, что блокировка снята.

10. Войдите в систему под пользователем admin Выполните команду

```
chage -l student7 > /home/admin1/student7
```

11. Прикрепите файлы из домашней директории

12. admin1: bash_history, student7

13. войдите в систему (имя пользователя: student; пароль: Asdf1234)

14. Добавьте новый сетевой интерфейс в виртуальную машину, или измените существующий.

15. Затем проверьте вывод команды ifconfig, отредактируйте файлы конфигураций.

16. В файле interfaces назначьте данному интерфейсу адрес 192.168.1.5 с маской 255.255.255.0.

17. Перезапустите службу и перезапустите интерфейс.

18. Проверьте доступность вашего сетевого интерфейса

19. Посмотрите текущую таблицу маршрутизации

20. Проверьте что получилось командой ifconfig:

21. Настройте виртуальный сетевой интерфейс, назначьте ему адрес 196.168.1.135, маску 255.255.255.0, broadcast 192.168.1.255.

22. Перед тем, как добавить маршруты посмотрите таблицу маршрутизации.

23. Для локального интерфейса выполняем команду:

```
route add -net 127.0.0.0 netmask 255.0.0.0 lo
```

24. Снова посмотрите таблицу маршрутизации.

25. Выполните команды:

```
ifconfig > /home/task
route >> /home/task
```

26. Прикрепите файл /home/student/.bash_history и /home/task.

Тема 5 Разграничение доступа в Astra Linux

Лабораторная работа:

1. Скачайте образ виртуальной машины (AstraLinuxSE).
2. Разверните ее и войдите в систему (имя пользователя: student; пароль: Asdf1234)
3. Авторизуйтесь под root. Создайте двух пользователей (user1 и user2) и задайте их пароли. Зарегистрируйтесь в первой консоли как user1.
4. С помощью Ctrl+Alt+F2 (Alt+F2) откройте второй текстовый терминал и зарегистрируйтесь как user
5. Аналогично откройте третий текстовый терминал и зарегистрируйтесь в нем с правами суперпользователя.
6. Нажатием Ctrl+Alt+F1 (Alt+Fl) вернитесь в первую консоль. Теперь, переключая консоль, вы можете работать с объектами операционной системы от имени двух разных пользователей и администратора системы.
7. С правами user1 попробуйте войти в каталог /root. С помощью команды ls -la /просмотрите список основных каталогов и посмотрите, каких прав доступа вам недостает для входа в каждый из каталогов.
8. Переключитесь в консоль администратора и создайте два новых временных каталога mkdir -m 777 /home/temp1 и mkdir -m 1777 /home/temp2.

Проверьте права доступа к каталогам /home/user 1 и /home/user2: они должны быть установлены в 755. Вернитесь в консоль user1.

9. Создайте в домашнем каталоге пользователя /home/user1 четыре каталога с именами: qul, qu2, qu3, qu4. При создании каталогов объявите следующие права доступа к ним: (qul - 777, qu2 - 404, qu3 - 1333, qu4 - 505). С помощью команды ls /home/user1 убедитесь в том, что каталоги созданы. В каждом из каталогов создайте по три текстовых файла с именами (jan, feb, mar), (apr, may, jun), (jul, aug, sep), (oct, nov, dec). В каждый файл запишите календарь на определенный месяц текущего года. Например, команда cal 1 2010 > jan создает в текущем каталоге файл jan и записывает в него календарь на январь 2010 года.

10. Измените нужные права доступа в "недоступные" каталоги qu2, qu4 и создайте там указанные файлы. После этого верните каталогам прежние права доступа.

11. Прочитайте содержимое одного из файлов в "темном" каталоге.

12. Перейдите во 2-ю консоль и с правами пользователя user2 войдите в каталог /home/user1/qul. Создайте в каталоге /home/user2 новый файл quart1 путем конкатенации нескольких имеющихся (cat jan feb mar >/home/user2/quart1). С помощью команды file определите тип созданного файла. Попробуйте вывести его на экран командой cat.

13. Установите права доступа 077 на созданный файл quart1. Вновь попробуйте прочесть его.

14. Установите для файла quatr1 права на доступ 4700.

15. Перейдите в консоль администратора и передайте право владения на файлы и aug пользователю user2 (команда chown user2 aug). Поочередно из консолей user1 и user2 проверьте, как изменились права владения файлами после его передачи.

16. Авторизуйтесь под admin1 Создайте каталог /foo

17. Дайте доступ к этой папке двум пользователям: user1 и user2. Первому полные права, а другому только на чтение.

18. Посмотрите ACL права доступа для папки foo

19. Удалите ACL для пользователя user1

20. Удалите полностью ACL список для /foo

21. Создайте новый файл file2 в вашей рабочей директории

22. Посмотрите ACL для файла file2.

23. Измените права доступа на чтение запись и выполнение для группы файла file2.

24. Посмотрите права на file2 с помощью команды ls -l а также посмотрите ACL для этого файла.

25. Добавьте в ACL группу group1 для файла file2. Установите только права на чтение и выполнение для данной группы.

26. Добавьте в ACL пользователя user2 для файла file2. Добавьте для этого пользователя доступ по исполнению файла.

27. Под пользователем user1 создайте папку dir1 в домашнем каталоге, и уберите все права для всех остальных.

28. Добавьте ACL по умолчанию (default ACL) на каталог dir1 для пользователя user2 с правами rw:

29. Создайте файл file1 в папке dir1 и посмотрите ACL.

30. Создайте file1 в домашнем каталоге и уберите все права для всех остальных.

31. Добавьте пользователя user2 к списку доступа для file1 с полными правами.

32. Проверьте доступ к файлу от пользователя user2 и admin1.

33. У администратора не должно быть доступа к файлу, а для пользователя user2 полный доступ.

34. Под admin1 выполните:

sudo getfacl /home/user1/dir1 > acl.file

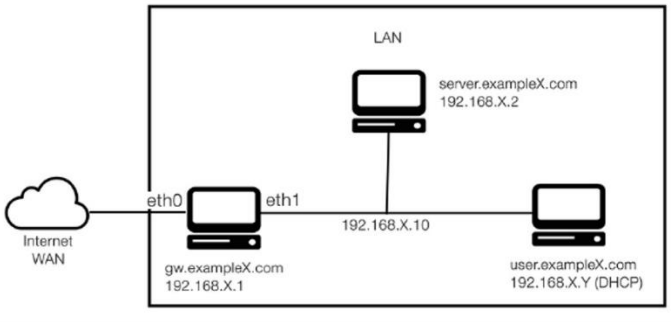
sudo getfacl /home/user1/dir1/file1 >> acl.file

Тема 6 Разграничение доступа в Astra Linux

Лабораторная работа:

4. Скачайте образы виртуальных машин (AstraLinuxSE, AstraLinuxSE_server).

5. Для доступа в системы воспользуйтесь учетными данными (имя пользователя: student; пароль: Asdf1234)



6. Настройте сервер gw в соответствии с конфигурацией, указанной выше. Номер Вашей сети обозначатся X = 5

- Ваша сеть: 192.168.5.0/24
- Домен: example5.com
- IP адрес для gw: 192.168.5.1
- Маска сети: 255.255.255.0
- Имя хоста для gw: gw.example5.com

7. По аналогии настройте server:

- IP адрес для server: 192.168.5.2
- Маска сети: 255.255.255.0
- Имя хоста для server: server.example5.com

8. На клиентской машине:

- Настроить получение адреса на интерфейсе по dhcp
- Имя хоста для user: user.example5.com

9. При необходимости перезагрузите виртуальную машину и отключите wicd

10. Текущая система будет использована как gw.example5.com. Добавьте еще один сетевой интерфейс. Для этого на вашем ПК необходимо создать виртуальный коммутатор или сетевой мост для внутренней коммутации.

11. Разверните еще 2 виртуальные машины AstralinuxSE_server и и AstralinuxSE

12. На gw выполните команды:

```
sudo ifconfig > gw.file
ping –c 3 192.168.5.2 >> gw.file
ping –c 3 ya.ru >> gw.file
hostname >> gw.file
```

13. На server выполните команды**:**

```
sudo ifconfig > server.file
ping –c 3 192.168.5.1 >> server.file
hostname >> server.file
```

14. На клиентской машине user выполните:

```
hostname > user.file
```

15. Если нет доступа к Winscp ssh выполните:

```
systemctl enable ssh.service
sudo init 6
```

16. Прикрепите файлы gw.file, server.file и user.file

17. Используя виртуальные машины, развернутые в предыдущем задании настройте DHCP сервер для вашей внутренней сети в соответствии с конфигурацией, указанной ниже. DHCP-сервер должен быть настроен на компьютере gw.example1.com

18. Установите DHCP сервер (пакет dhcp3-server при помощи пакетного менеджера в системе) и отредактируйте файл /etc/default/isc-dhcp-server

19. Перейдите в каталог /etc/dhcp/, откройте файл dhcpd.conf вашим текстовым редактором, установите следующие параметры.

20. Стандартная конфигурация сервера DHCP

```
ddns-update-style none;
log-facility local7;
subnet 192.168.1.0 netmask 255.255.255.0
{
default-lease-time 600;
max-lease-time 7200;
range 192.168.1.151 192.168.1.200;
option routers 192.168.1.1;
option domain-name "exaample1.com";
option domain-name-servers 192.168.1.2;
}
```

21. Проверьте, правильно ли был изменен конфигурационный файл

22. Перезапустите службу isc-dhcp-server.

23. Проверьте, получил ли адрес компьютер пользователя user (выше указанными командами)

24. Мониторинг выданных адресов можно определить следующим образом:

```
tail -f /var/lib/dhcp/dhcpd.leases
```

25. на gw выполните

```
tail -n 23 /var/lib/dhcp/dhcpd.leases > dhcp.file
```

26. на user выполните:

```
sudo ifconfig > dhcp.user
ping -c 3 192.168.1.1 >> dhcp.user
hostname >> dhcp.user
```

27. Используя виртуальные машины развернутые в предыдущем задании настройте первичный DNS-сервер на комьютере **server** для Вашего домена example1.com в соответствии с конфигурацией, указанной ниже. Проверьте работу DNS-сервера с компьютера server и с клиентской виртуальной машины User.

28. На компьютерах server.example1.com и gw.example1.com: Откройте терминал,и войдите с правами суперпользователя.

29. Установите пакет bind9 при помощи пакетного менеджера в системе (при необходимости надо примонтировать установочный диск astra linux).

30. Укажите основной dns-сервер в файле /etc/resolv.conf

31. На компьютере server.example1.com (проверьте настройки сети для сервера)

32. Настройка мастер (первичного) сервера зоны example1.com (на компьютере (server.example1.com)

33. Перейдите в каталог /etc/bind/, создайте файл вашей зоны (например, example1.com при помощи вашего текстового редактора, установите следующие параметры.

```
$TTL   3h
example1.com. SOA ns root.server (
2019051901 ; Серийный номер зоны (образуется датой)
10800 ; Обновлять данные каждые три часа
900 ; Попытка повторения через 15 минут
```

```
604800 ; Истечение актуальности через неделю
86400 ; Минимальный срок жизни 1 день
)
NS    ns
;     A     192.168.1.2
;     MX    1 server
;     MX    2 gw
ns    A     192.168.1.2
server  A    192.168.1.2
gw    A     192.168.1.1
;nfs  CNAME    server
;samba  CNAME    server
;ftp  CNAME     server
;www  CNAME      server
;user1  CNAME    server
;smtp  CNAME     server
;imap  CNAME     server
;pop3  CNAME     server
;ntp  CNAME     gw
```

34. Выполните проверку конфигурационного файла.

35. Добавьте в файл/etc/bind/named.conf.local запись:

```
zone "example1.com" {
type master;
file "/etc/bind/example1.com";
};
```

36. Перезапустите службу bind.

37. На компьютере gw.example1.com (проверьте настройки сети для сервера):

38. На компьютере gw.exapmle1.com настойте вторичный сервер зоны dns. Отредактируйте файл /etc/bind/named.conf.local

```
...
zone "example1.com" {
type slave;
file "/var/cache/bind/example1.com";
masters { 192.168.1.2; };
};
```

39. Отредактируйте файл /var/cache/bind/example1.com

```
$TTL 3h
example1.com. SOA ns root.server 1 1d 12h 1w 3h
NS ns
NS gw.example1.com.
```

40. Выполнить

```
rndc reload
```

41. Проверьте на клиентской машине user соединение с server и gate.

42. Выполните команду на user.example1.com:

```
ping -c 3 gw > dns.file
ping 0c 3 server >> dns.file
```

43. Используя виртуальные машины развернутые в предыдущем задании на кампьютере server установите пакет apache2

44. Установите пакет для проверки аутентификации libapache2-mod-auth-pam

45. Проверьте, что в файле /etc/apache2/ports.conf содержатся следующие строки

```
NameVirtualHost *:80
Listen 80
```

46. Создайте директорию /var/www/html/test и дайте пользователю www-data доступ к ней.

47. В каталоге /etc/apache2/sites-available должны находиться файлы с настройками виртуальных хостов и как минимум один из них должен быть разрешен к использованию командой:

```
a2ensite config_filename
```

Минимальное содержимое таких файлов с конфигурациями виртуальных хостов выглядит следующим образом:

```
<VirtualHost *:80>
ServerAdmin webmaster@localhost
ServerName server.domain.name
DocumentRoot /var/www/html/
<Directory /var/www/html/test>
Options Indexes FollowSymLinks MultiViews
AllowOverride None
</Directory>
ErrorLog /var/log/apache2/error.log
LogLevel warn
CustomLog /var/log/apache2/access.log combined
</VirtualHost>
```

48. После окончания правки конфигурационных файлов необходимо перезапустить сервер apache

49. В конфигурационных файлах виртуальных хостов web-сервера Apache2 укажите:

```
AuthType Basic
AuthName "private area"
AuthBasicProvider PAM
AuthPAMService apache
Require valid-user
```

50. В файл /etc/pam.d/apache2 добавьте

```
auth required pam_unix.so
```

```
account required pam_unix.so
```

51. Для корректного функционирования авторизации через PAM пользователю, от которого работает web-сервер (по умолчанию — www-data), необходимо выдать права на чтение информации из БД пользователей и сведений о мандатных метках:

```
usermod -a -G shadow www-data
setfacl -d -m u:www-data:r /etc/parsec/macdb
setfacl -R -m u:www-data:r /etc/parsec/macdb
setfacl -m u:www-data:rx /etc/parsec/macdb
```

52. Включите PAM авторизацию:

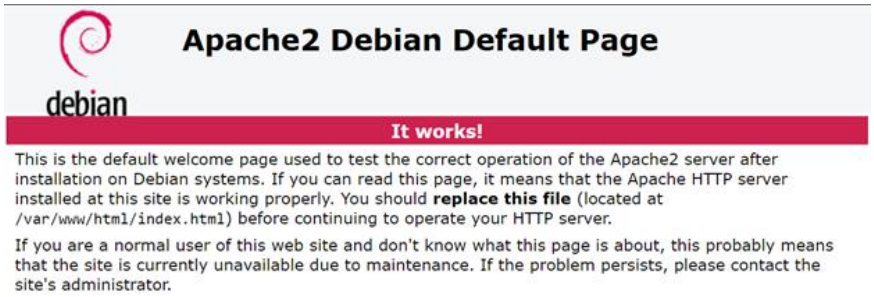
53. Создайте пользователя webuser под которым будете заходить на сайт.

54. Выполните команду настройки минимальный и максимальный наборы мандатных категорий:

55. Скопируйте файл index.html в каталог, который создали.

56. На клиенте в браузере перейдите на сайт server.exampleX.com, а затем на server.exampleX.com.test

57. После того как авторизуетесь и увидите страницу вида:



58. Выполните команду на сервере:

```
#tail -5 /var/log/apache2/access.log > /home/weblog
```

59. Прикрепите файл /home/weblog

24. 6 Системы виртуализации и контейнеризации (Docker)

Лабораторная работа 1
Дайте письменные ответы на следующие вопросы:

- В чём отличие режимов работы сервисов в Docker Swarm кластере: replication и global?

В режиме replicated приложение запускается в том количестве экземпляров, какое укажет пользователь. При этом на отдельной
ноде может быть как несколько экземпляров приложения, так и не быть совсем.

В режиме global приложение запускается обязательно на каждой ноде и в единственном экземпляре.

- Какой алгоритм выбора лидера используется в Docker Swarm кластере?

Raft.

- Протокол решает проблему согласованности - чтобы все manager-ноды имели одинаковое представление о состоянии кластера
- Для отказоустойчивой работы должно быть не менее трёх manager-нод.
- Количество нод обязательно должно быть нечётным, но лучше не более 7 (рекомендация из документации Docker).
- Среди manager-нод выбирается лидер, его задача гарантировать согласованность.
- Лидер отправляет keepalive-пакеты с заданной периодичностью в пределах 150-300мс. Если пакеты не пришли, менеджеры начинают выборы нового лидера.
- Если кластер разбит, нечётное количество нод должно гарантировать, что кластер останется консистентным,
т.к. факт изменения состояния считается совершенным, если его отразило большинство нод. Если разбить кластер пополам,
нечётное число гарантирует, что в какой-то части кластера будеть большинство нод.

- Что такое Overlay Network?

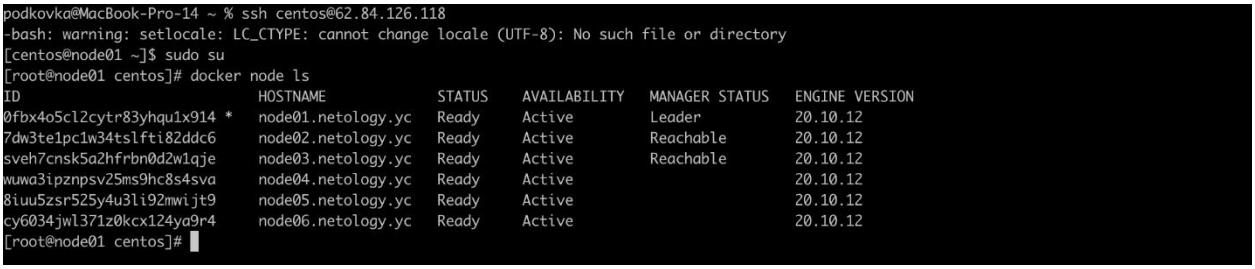
L2 VPN сеть для связи демонов Docker между собой. В основе используется технология vxlan

Лабораторная работа 2

Создать ваш первый Docker Swarm кластер в Яндекс.Облаке

Для получения зачета, вам необходимо предоставить скриншот из терминала (консоли), с выводом команды:

```
docker node ls
```



Лабораторная работа 3

Создать ваш первый, готовый к боевой эксплуатации кластер мониторинга, состоящий из стека микросервисов.

Для получения зачета, вам необходимо предоставить скриншот из терминала (консоли), с выводом команды:

```
docker service ls
```

```
[root@node01 centos]# docker service ls
ID             NAME                                MODE         REPLICAS   IMAGE                                          PORTS
ti55r7evyx86   swarm_monitoring_alertmanager       replicated   1/1        stefanprodan/swarmprom-alertmanager:v0.14.0
p7gekt51hjoc   swarm_monitoring_caddy              replicated   1/1        stefanprodan/caddy:latest                      *:3000->3000/tcp, *:9090->9090/tcp, *:9093-9094->9093-9094/tcp
ixtceqngyqwv   swarm_monitoring_cadvisor           global       6/6        google/cadvisor:latest
715xjm8diecs   swarm_monitoring_dockerd-exporter   global       6/6        stefanprodan/caddy:latest
v8v3vfxh6ogl   swarm_monitoring_grafana            replicated   1/1        stefanprodan/swarmprom-grafana:5.3.4
3uynrlu3sipq   swarm_monitoring_node-exporter      global       6/6        stefanprodan/swarmprom-node-exporter:v0.16.0
g2ivfy4tkss7   swarm_monitoring_prometheus         replicated   1/1        stefanprodan/swarmprom-prometheus:v2.5.0
miiwn1djce41   swarm_monitoring_unsee              replicated   1/1        cloudflare/unsee:v0.8.0
[root@node01 centos]#
```

Лабораторная работа 4

Выполнить на лидере Docker Swarm кластера команду (указанную ниже) и дать письменное описание её функционала, что она делает и зачем она нужна:

```
# см.документацию: https://docs.docker.com/engine/swarm/swarm_manager_locking/
docker swarm update --autolock=true
```

```
[root@node03 centos]# service docker restart
Redirecting to /bin/systemctl restart docker.service
[root@node03 centos]# docker node ls
Error response from daemon: Swarm is encrypted and needs to be unlocked before it can be used. Please use "docker swarm unlock" to unlock it.
[root@node03 centos]# docker swarm unlock
Please enter unlock key:
[root@node03 centos]# docker node ls
ID                            HOSTNAME              STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
1zs279voao1yxomzw1vsum8ia     node01.netology.yc    Ready     Active         Leader           20.10.12
md26qdduqxfix6pv4n4aqshjz     node02.netology.yc    Ready     Active         Reachable        20.10.12
i8zhu0yoe0xz4tmatkvfbc1o9 *   node03.netology.yc    Ready     Active         Reachable        20.10.12
1mtbfw14mjl30jlhn3nw8rbsa     node04.netology.yc    Ready     Active                          20.10.12
lwhdxw7dfhj4bjsybfyfn7h2u     node05.netology.yc    Ready     Active                          20.10.12
geg5gq8yfa6h38roc81qa272l     node06.netology.yc    Ready     Active                          20.10.12
[root@node03 centos]#
```

--autolock=true обязывает вводить ключ разблокировки на ноде, чтобы она могла заново присоединиться к кластеру, если была перезапущена. Ввод ключа позволит расшифровать лог Raft и загрузить все "секреты" в память ноды (логины, пароли, TLS ключи, SSH ключи и т.д.)
Для защиты кластера от несанкционированного доступа к файлам ноды. Например, получив жесткий диск сервера или образ диска VM с нодой, не получить доступ к кластеру и нодам без ключа.

Лабораторная работа 5

Создать собственный образ операционной системы с помощью Packer.

Для получения зачета, вам необходимо предоставить:

Скриншот страницы, как на слайде из презентации (слайд 37).

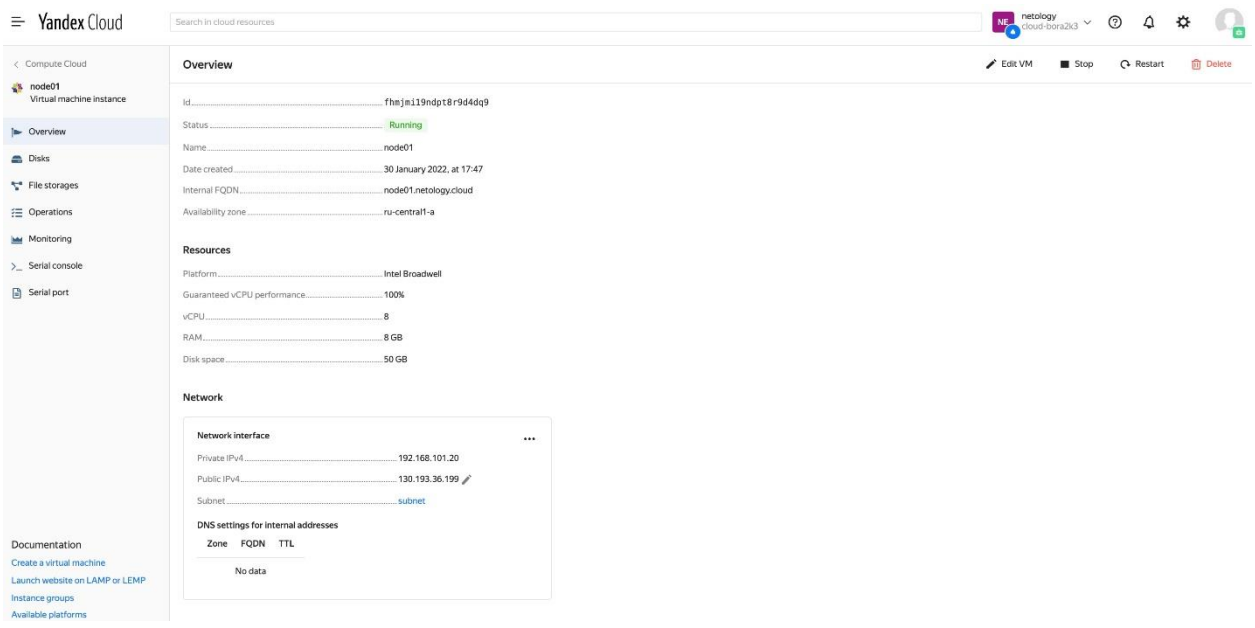
```
    yandex:
    yandex: Complete!
==> yandex: Stopping instance...
==> yandex: Deleting instance...
    yandex: Instance has been deleted!
==> yandex: Creating image: centos-7-base
==> yandex: Waiting for image to complete...
==> yandex: Success image create...
==> yandex: Destroying boot disk...
    yandex: Disk has been deleted!
Build 'yandex' finished after 2 minutes 8 seconds.

==> Wait completed after 2 minutes 8 seconds

==> Builds finished. The artifacts of successful builds are:
--> yandex: A disk image was created: centos-7-base (id: fd8dlq6iuec7m7n5t8oh) with family name centos
podkovka@MacBook-Pro-14 packer % yc compute image list
+----------------------+---------------+--------+----------------------+--------+
|          ID          |     NAME      | FAMILY |     PRODUCT IDS      | STATUS |
+----------------------+---------------+--------+----------------------+--------+
| fd8dlq6iuec7m7n5t8oh | centos-7-base | centos | f2e6u62hbpkah20ftmhi | READY  |
+----------------------+---------------+--------+----------------------+--------+
```

Лабораторная работа 6

Создать вашу первую виртуальную машину в Яндекс.Облаке.

Для получения зачета, вам необходимо предоставить:

Скриншот страницы свойств созданной ВМ:

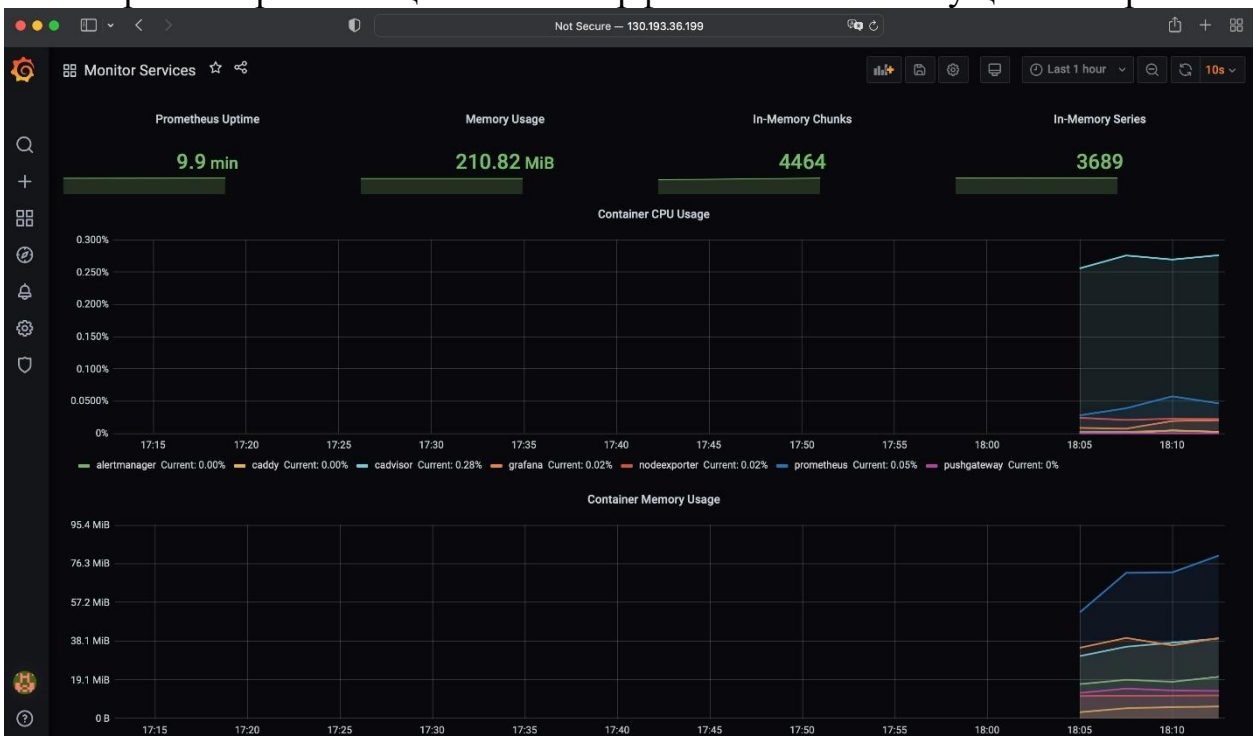


Лабораторная работа 7

Создать ваш первый готовый к боевой эксплуатации компонент мониторинга, состоящий из стека микросервисов.

Для получения зачета, вам необходимо предоставить:

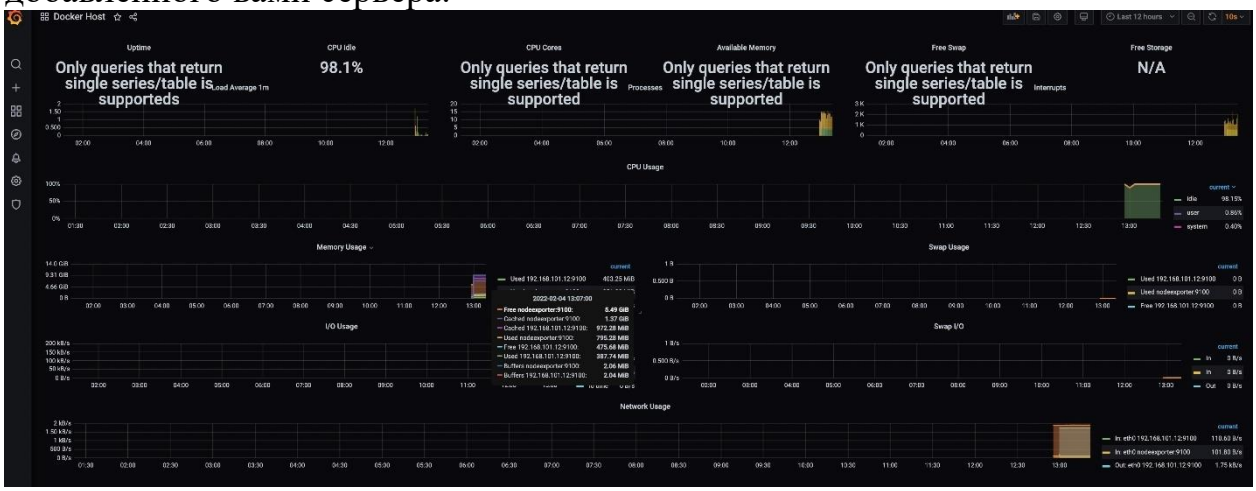
Скриншот работающего веб-интерфейса Grafana с текущими метриками:



Создать вторую ВМ и подключить её к мониторингу развёрнутому на первом сервере.

Для получения зачета, вам необходимо предоставить:

Скриншот из Grafana, на котором будут отображаться метрики добавленного вами сервера.



Создаем node02.tf по аналогии с node01.tf, добавив ip

```
  network_interface {
    ....
    ....
    ip_address = "192.168.101.12"
  }
```

Добавляем в inventory

```
....
[monitoring:children]
machines

[machines]
node02.netology.cloud ansible_host=-----ip_node2-----
```

Создаем exporters/docker-compose.exporters.yml по аналогии с node01
Создаем docker-compose.yml с секциями nodeexporter и cadvisor
Создаем add_nodes.yml по аналогии с provision.yml
Добавляем node02 - ansible-playbook add_nodes.yml
В Grafana на дашбордах в Metrics-Legend добавляем {{instance}}

24. 7 Системы непрерывной интеграции (Jenkins)

Лабораторная работа 1

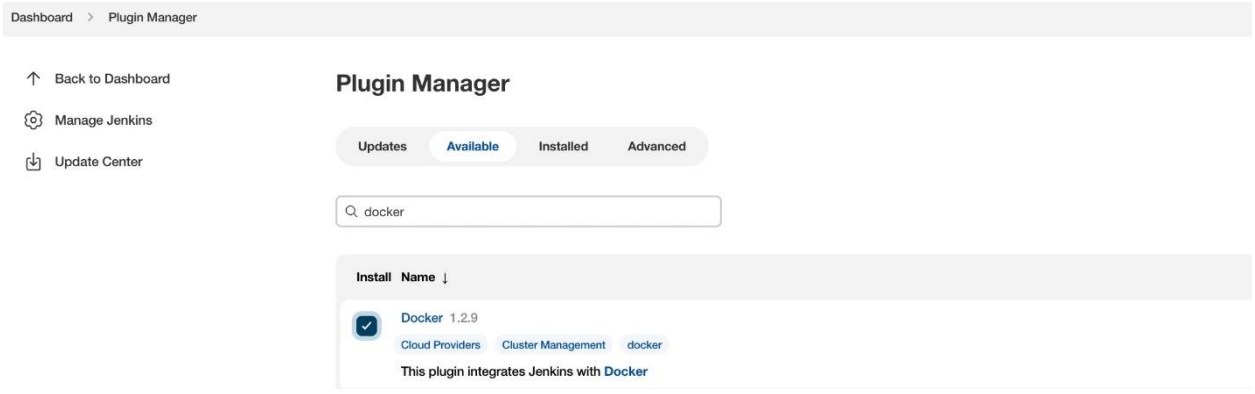
1. Установить jenkins по любой из инструкций

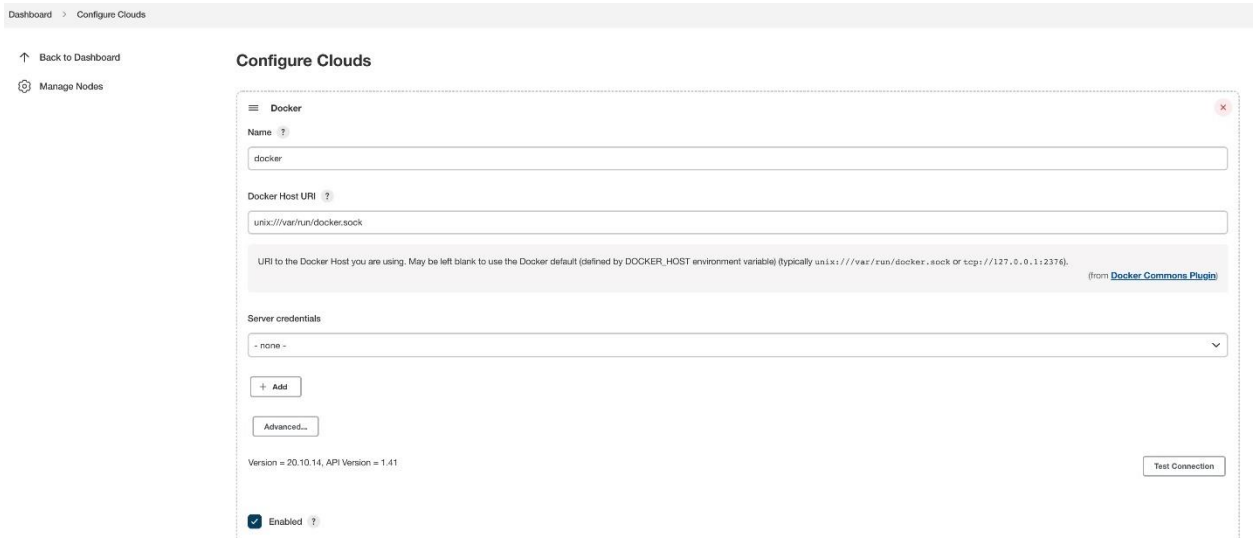
```
version: '3.8'
services:
  jenkins:
    image: jenkins/jenkins:latest-jdk11
    privileged: true
    user: root
    ports:
      - 8080:8080
      - 50000:50000
    container_name: jenkins
    volumes:
      - $HOME/jenkins_compose/jenkins_configuration:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
```

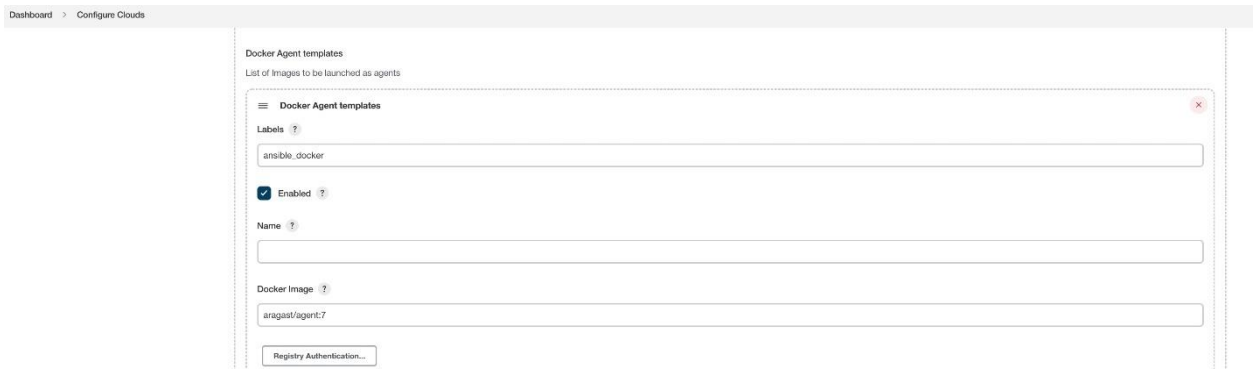
2. Запустить и проверить работоспособность

```
$ docker-compose up -d
$ docker-compose ps
NAME                COMMAND                  SERVICE             STATUS
PORTS
jenkins             "/sbin/tini -- /usr/…"   jenkins             running
0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp, :::8080->8080/tcp, :::50000-
>50000/tcp
$ docker exec jenkins cat /var/jenkins_home/secrets/initialAdminPassword
f27d88xxxxxxxxxxxxxxxxxxxx4e2e75
```

3. Сделать первоначальную настройку
4. Настроить под свои нужды
5. Поднять отдельный cloud



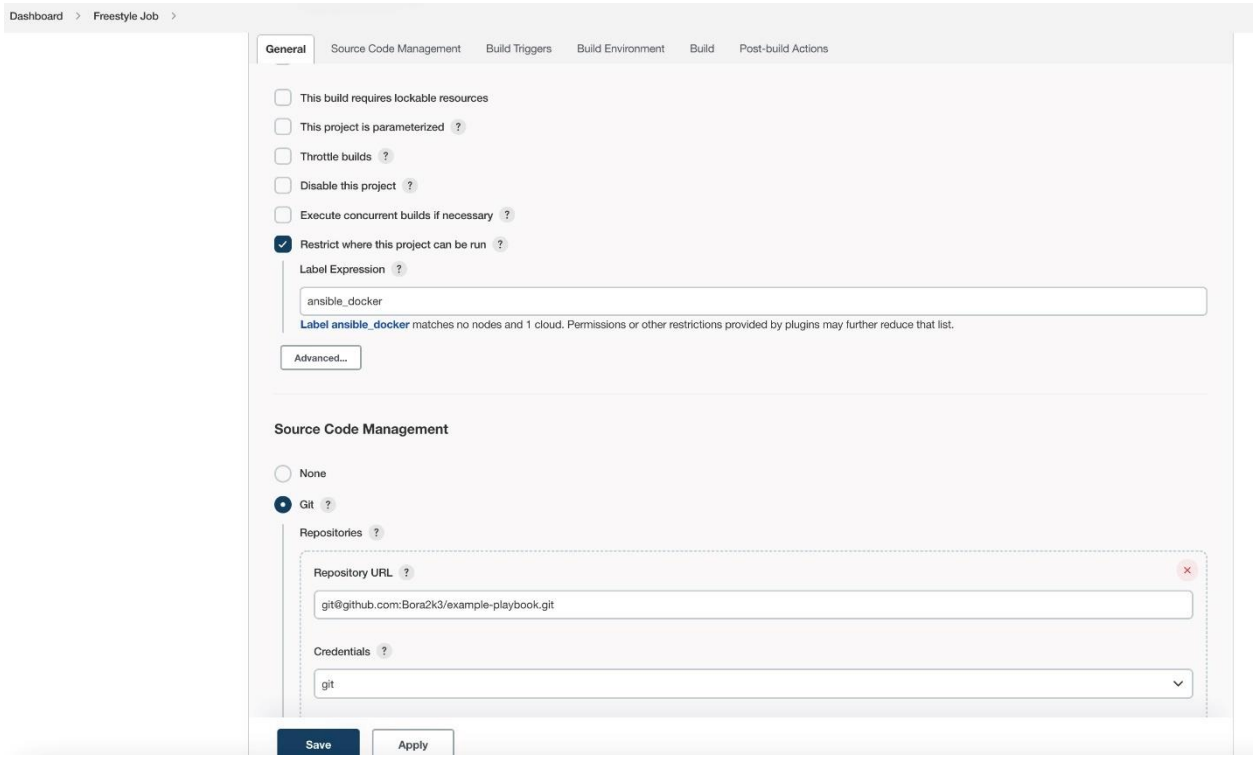


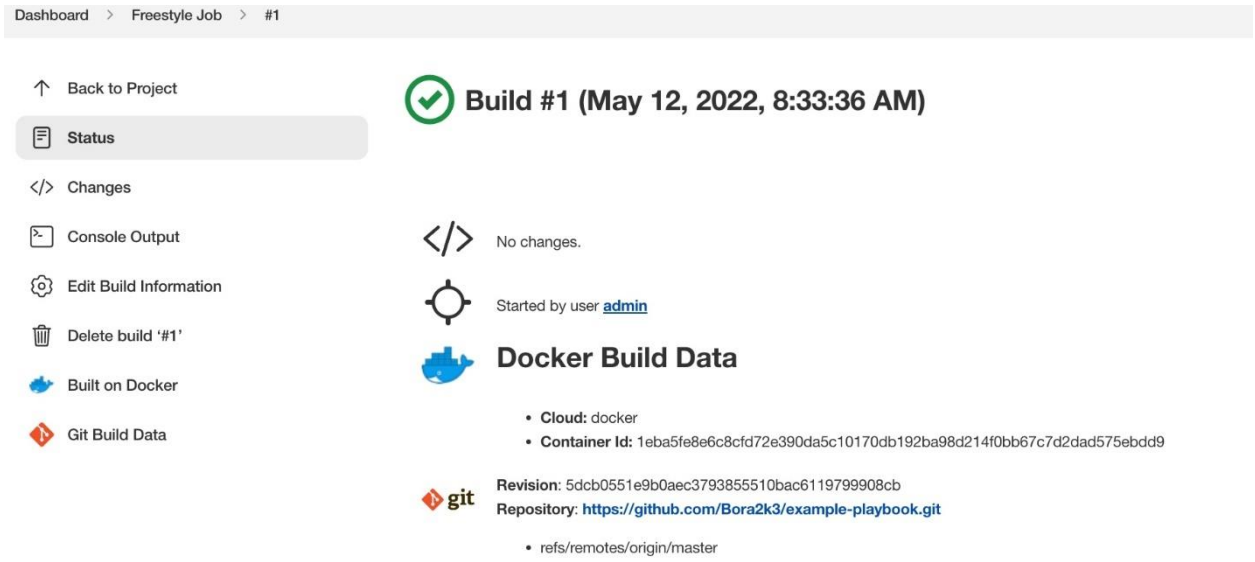


6. Для динамических агентов можно использовать образ
7. Обязательный параметр: поставить label для динамических агентов: ansible docker
8. Сделать форк репозитория с playbook

Лабораторная работа 2

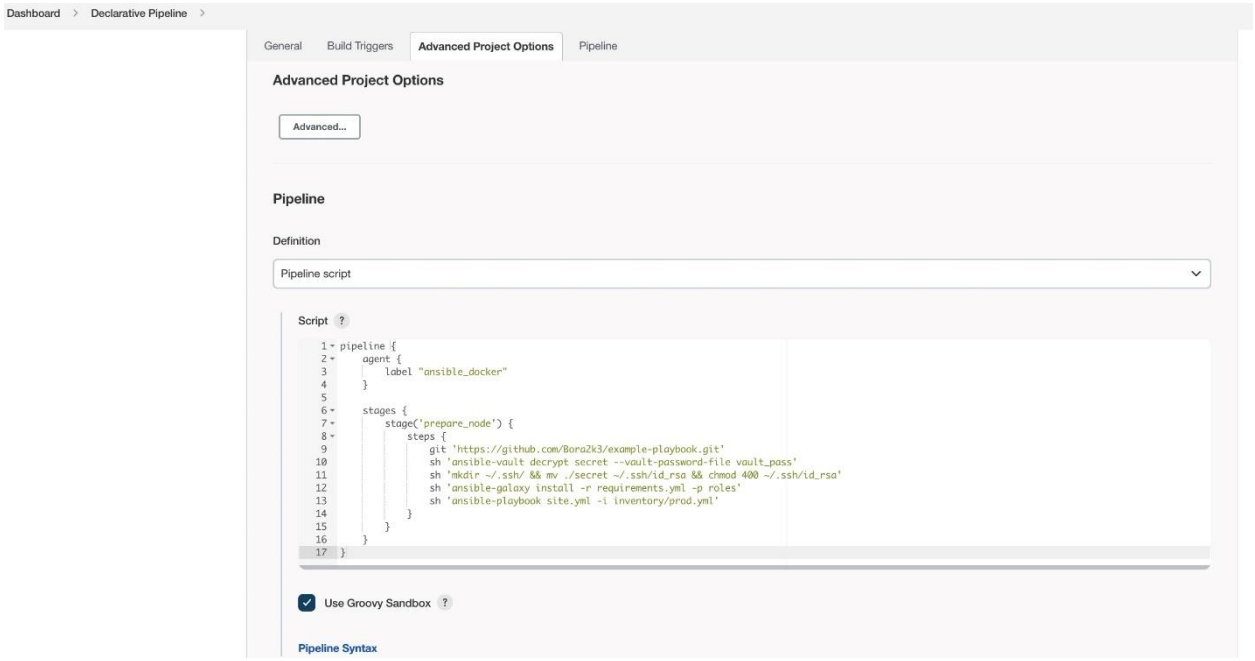
1. Сделать Freestyle Job, который будет запускать ansible-playbook из форка репозитория

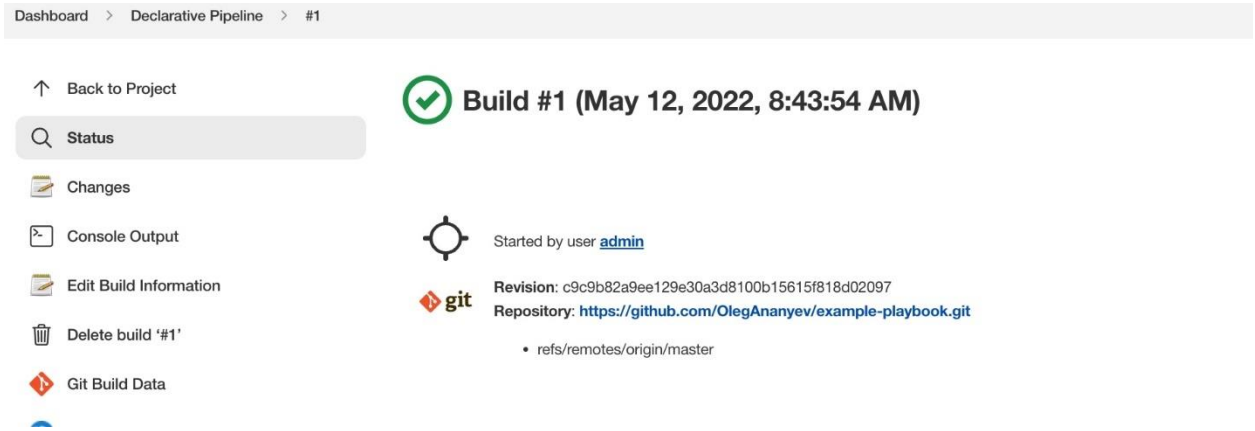





```
PLAY RECAP *********************************************************************
localhost                  : ok=5    changed=4    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0

Finished: SUCCESS
```

2. Сделать Declarative Pipeline, который будет выкачивать репозиторий с playbook и запускать её





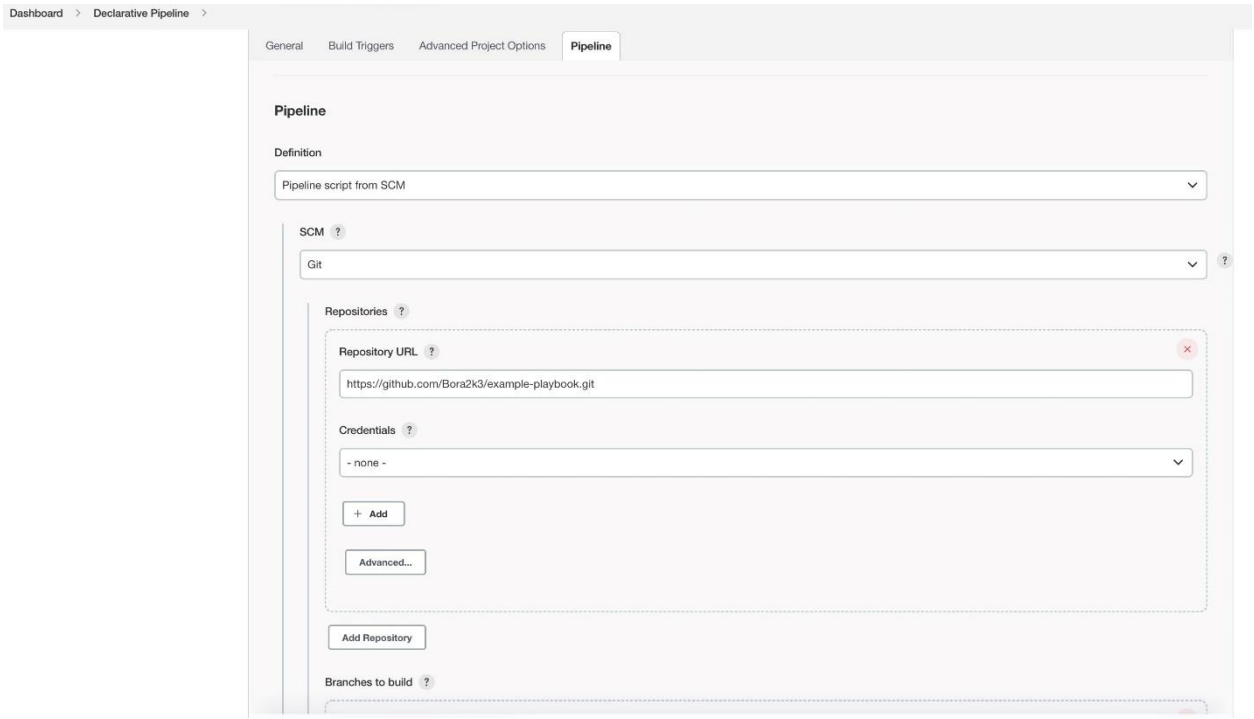

```
PLAY RECAP *********************************************************************
localhost                  : ok=5    changed=4    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0

[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

3. Перенести Declarative Pipeline в репозиторий в файл Jenkinsfile

```
Изменил содержимое Jenkinsfile в репозитории
pipeline {
    agent {
        label "ansible_docker"
    }

    stages {
        stage('prepare_node') {
```

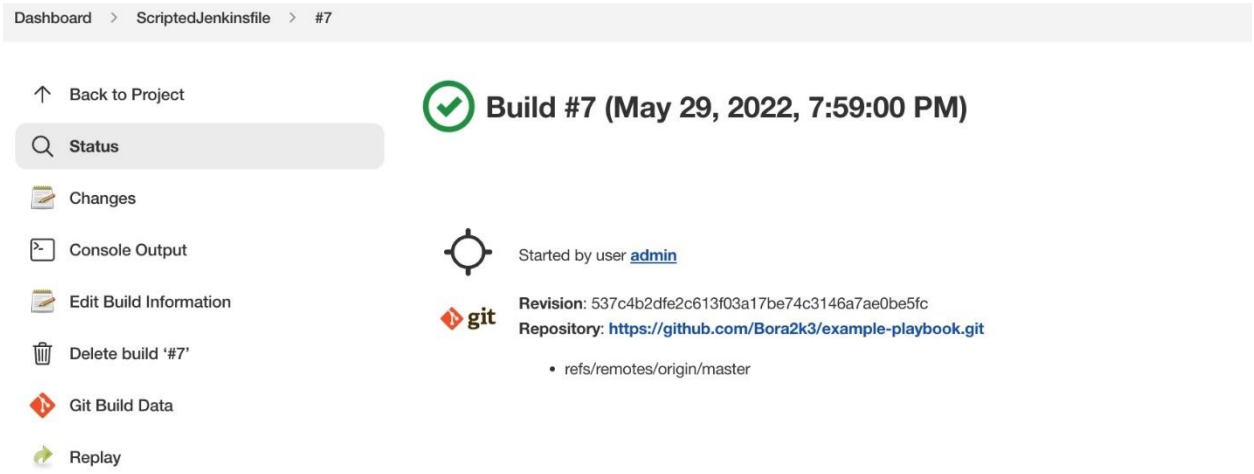
```
            steps {
                git 'https://github.com/Bora2k3/example-playbook.git'
                sh 'ansible-vault decrypt secret --vault-password-file vault_pass'
                sh 'mkdir ~/.ssh/ && mv ./secret ~/.ssh/id_rsa && chmod 400
~/.ssh/id_rsa'
                sh 'ansible-galaxy install -r requirements.yml -p roles'
                sh 'ansible-playbook site.yml -i inventory/prod.yml'
            }
        }
    }
}
```

4. Перенастроить Job на использование Jenkinsfile из репозитория




```
PLAY RECAP *********************************************************************
localhost                  : ok=5    changed=4    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0

[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

5. Создать Scripted Pipeline, наполнить его скриптом из pipeline
6. Заменить credentialsId на свой собственный
7. Проверить работоспособность, исправить ошибки, исправленный Pipeline вложить в репозитрий в файл ScriptedJenkinsfile




```
PLAY RECAP *********************************************************************
localhost                  : ok=5    changed=4    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0

[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Лабораторная работа 3

1. Создать скрипт на groovy, который будет собирать все Job, которые завершились хотя бы раз неуспешно. Добавить скрипт в репозиторий с решеним с названием AllJobFailure.groovy
2. Установить customtools plugin
3. Поднять инстанс с локальным nexus, выложить туда в анонимный доступ .tar.gz с ansible версии 2.9.x
4. Создать джобу, которая будет использовать ansible из customtool
5. Джоба должна просто исполнять команду ansible-version, в ответ прислать лог исполнения джобы

24. 8 Системы управления конфигурацией. Инфраструктура как код (Ansible)

Лабораторная работа 1

1. Установите ansible версии 2.10 или выше.

```
$ ansible --version
ansible [core 2.12.5]
```

2. Создайте свой собственный публичный репозиторий на github с произвольным именем.

```
https://github.com/Bora2k3/08-ansible-01-base
```

3. Скачайте playbook из репозитория с домашним заданием и перенесите его в свой репозиторий.

Лабораторная работа 2

1. Попробуйте запустить playbook на окружении из test.yml, зафиксируйте какое значение имеет факт some_fact для указанного хоста при выполнении playbook.

```
$ ansible-playbook -i inventory/test.yml site.yml

PLAY [Print os facts]
*********************************************************************************
*********************************************************************************
*****************************

TASK [Gathering Facts]
*********************************************************************************
*********************************************************************************
*****************************
ok: [localhost]

TASK [Print OS]
*********************************************************************************
*********************************************************************************
***********************************
ok: [localhost] => {
    "msg": "Ubuntu"
}

TASK [Print fact]
*********************************************************************************
*********************************************************************************
*********************************
ok: [localhost] => {
    "msg": 12
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
***************************************
localhost                  : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

2. Найдите файл с переменными (group_vars) в котором задаётся найденное в первом пункте значение и поменяйте его на 'all default fact'.

```
% cat group_vars/all/examp.yml
---
  some_fact: all default fact
```

3. Воспользуйтесь подготовленным (используется docker) или создайте собственное окружение для проведения дальнейших испытаний.

```
docker-compose.yml

version: '3'
services:
  centos7:
    image: pycontribs/centos:7
    container_name: centos7
    restart: unless-stopped
    entrypoint: "sleep infinity"

  ubuntu:
```

```
    image: pycontribs/ubuntu
    container_name: ubuntu
    restart: unless-stopped
    entrypoint: "sleep infinity"
```

4. Проведите запуск playbook на окружении из prod.yml. Зафиксируйте полученные значения some_fact для каждого из managed host.

```
$ ansible-playbook -i inventory/prod.yml -v site.yml
Using /etc/ansible/ansible.cfg as config file

PLAY [Print os facts]
*********************************************************************************
*********************************************************************************
*****************************

TASK [Gathering Facts]
*********************************************************************************
*********************************************************************************
*****************************
ok: [ubuntu]
ok: [centos7]

TASK [Print OS]
*********************************************************************************
*********************************************************************************
***********************************
ok: [centos7] => {
    "msg": "CentOS"
}
ok: [ubuntu] => {
    "msg": "Ubuntu"
}

TASK [Print fact]
*********************************************************************************
*********************************************************************************
*********************************
ok: [centos7] => {
    "msg": "el"
}
ok: [ubuntu] => {
    "msg": "deb"
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
*************************************
centos7                    : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
ubuntu                     : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

5. Добавьте факты в group_vars каждой из групп хостов так, чтобы для some_fact получились следующие значения: для deb - 'deb default fact', для el - 'el default fact'.

```
$ cat group_vars/deb/examp.yml ;echo ""
---
  some_fact: "deb default fact"
$ cat group_vars/el/examp.yml ;echo ""
---
  some_fact: "el default fact"
```

6. Повторите запуск playbook на окружении prod.yml. Убедитесь, что выдаются корректные значения для всех хостов.

```
$ ansible-playbook -i inventory/prod.yml -v site.yml
Using /etc/ansible/ansible.cfg as config file

PLAY [Print os facts]
*********************************************************************************
*********************************************************************************
*****************************

TASK [Gathering Facts]
*********************************************************************************
*********************************************************************************
*****************************
ok: [ubuntu]
ok: [centos7]

TASK [Print OS]
*********************************************************************************
*********************************************************************************
***********************************
ok: [centos7] => {
    "msg": "CentOS"
}
ok: [ubuntu] => {
    "msg": "Ubuntu"
}

TASK [Print fact]
*********************************************************************************
*********************************************************************************
*********************************
ok: [centos7] => {
    "msg": "el default fact"
}
ok: [ubuntu] => {
    "msg": "deb default fact"
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
*************************************
centos7                    : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
ubuntu                     : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

7. При помощи ansible-vault зашифруйте факты в group_vars/deb и group_vars/el.

```
$ ansible-vault encrypt group_vars/deb/examp.yml
New Vault password:
Confirm New Vault password:
Encryption successful
$ ansible-vault encrypt group_vars/el/examp.yml
New Vault password:
Confirm New Vault password:
Encryption successful
```

8. Запустите playbook на окружении prod.yml. При запуске ansible должен запросить у вас пароль. Убедитесь в работоспособности.

```
$ ansible-playbook -i inventory/prod.yml site.yml
```

```
PLAY [Print os facts]
*********************************************************************************
*********************************************************************************
*****************************
ERROR! Attempting to decrypt but no vault secrets found
$ ansible-playbook -i inventory/prod.yml site.yml --ask-vault-pass
Vault password:

PLAY [Print os facts]
*********************************************************************************
*********************************************************************************
*****************************

TASK [Gathering Facts]
*********************************************************************************
*********************************************************************************
*****************************
ok: [ubuntu]
ok: [centos7]

TASK [Print OS]
*********************************************************************************
*********************************************************************************
***********************************
ok: [centos7] => {
    "msg": "CentOS"
}
ok: [ubuntu] => {
    "msg": "Ubuntu"
}

TASK [Print fact]
*********************************************************************************
*********************************************************************************
***********************************
ok: [centos7] => {
    "msg": "el default fact"
}
ok: [ubuntu] => {
    "msg": "deb default fact"
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
*************************************
centos7                    : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
ubuntu                     : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

9. Посмотрите при помощи ansible-doc список плагинов для подключения. Выберите подходящий для работы на control node.

нужен "local"

10. В prod.yml добавьте новую группу хостов с именем local, в ней разместите localhost с необходимым типом подключения.

```
$ cat inventory/prod.yml ; echo ""
---
  el:
```

```
    hosts:
      centos7:
        ansible_connection: docker
  deb:
    hosts:
      ubuntu:
        ansible_connection: docker
  local:
    hosts:
      localhost:
        ansible_connection: local
```

11. Запустите playbook на окружении prod.yml. При запуске ansible должен запросить у вас пароль. Убедитесь что факты some_fact для каждого из хостов определены из верных group_vars.

```
без создания отдельного group_vars -> получил для local из all
$ ansible-playbook -i inventory/prod.yml site.yml --ask-vault-pass
Vault password:

PLAY [Print os facts]
*********************************************************************************
*********************************************************************************
*****************************

TASK [Gathering Facts]
*********************************************************************************
*********************************************************************************
*****************************
ok: [localhost]
ok: [ubuntu]
ok: [centos7]

TASK [Print OS]
*********************************************************************************
*********************************************************************************
***********************************
ok: [centos7] => {
    "msg": "CentOS"
}
ok: [ubuntu] => {
    "msg": "Ubuntu"
}
ok: [localhost] => {
    "msg": "Ubuntu"
}

TASK [Print fact]
*********************************************************************************
*********************************************************************************
*********************************
ok: [centos7] => {
    "msg": "el default fact"
}
ok: [ubuntu] => {
    "msg": "deb default fact"
}
ok: [localhost] => {
    "msg": "all default fact"
}
```

```
PLAY RECAP
*********************************************************************************
*********************************************************************************
*************************************
centos7                    : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
localhost                  : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
ubuntu                     : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
создал отдельный group-vars для local -> получил для local из local
$ ansible-playbook -i inventory/prod.yml site.yml --ask-vault-pass
Vault password:

PLAY [Print os facts]
*********************************************************************************
*********************************************************************************
*****************************

TASK [Gathering Facts]
*********************************************************************************
*********************************************************************************
*****************************
ok: [localhost]
ok: [ubuntu]
ok: [centos7]

TASK [Print OS]
*********************************************************************************
*********************************************************************************
***********************************
ok: [centos7] => {
    "msg": "CentOS"
}
ok: [ubuntu] => {
    "msg": "Ubuntu"
}
ok: [localhost] => {
    "msg": "Ubuntu"
}

TASK [Print fact]
*********************************************************************************
*********************************************************************************
*********************************
ok: [centos7] => {
    "msg": "el default fact"
}
ok: [ubuntu] => {
    "msg": "deb default fact"
}
ok: [localhost] => {
    "msg": "local default fact"
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
*************************************
centos7                    : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
localhost                  : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

```
ubuntu                     : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

12. Заполните README.md ответами на вопросы. Сделайте git push в ветку master. В ответе отправьте ссылку на ваш открытый репозиторий с изменённым playbook и заполненным README.md.

Лабораторная работа 3

1. При помощи ansible-vault расшифруйте все зашифрованные файлы с переменными.

```
$ ansible-vault decrypt --ask-vault-password group_vars/deb/* group_vars/el/*
```

2. Зашифруйте отдельное значение PaSSw0rd для переменной some_fact паролем netology. Добавьте полученное значение в group_vars/all/exmp.yml.

```
$ ansible-vault encrypt_string "PaSSw0rd"
New Vault password:
Confirm New Vault password:
!vault |
          $ANSIBLE_VAULT;1.1;AES256

38643066646437356365386634383037646562353437373135613063373939353631313936373463

30623739353865633432643630636139373861613135666620a663137613839616237666535663330

65306138323139396633536326534646136333437666262323532363863623035326161393133 62

35643432663164343 40a37656464383134306164313161383638373565363035623564313763326 3
          3434
Encryption successful

$ cat group_vars/all/examp.yml
---
  some_fact: !vault |
          $ANSIBLE_VAULT;1.1;AES256

38643066646437356365386634383037646562353437373135613063373939353631313936373463

30623739353865633432643630636139373861613135666620a663137613839616237666535663330

65306138323139396633536326534646136333437666262323532363863623035326161393133 62

35643432663164343 40a37656464383134306164313161383638373565363035623564313763326 3
          3434
```

3. Запустите playbook, убедитесь, что для нужных хостов применился новый fact.

```
$ rm -rf group_vars/local
$ ansible-playbook -i inventory/prod.yml site.yml --ask-vault-pass
Vault password:

PLAY [Print os facts]
*********************************************************************************
*********************************************************************************
****************************
```

```
TASK [Gathering Facts]
*********************************************************************************
*********************************************************************************
****************************
ok: [localhost]
ok: [ubuntu]
ok: [centos7]

TASK [Print OS]
*********************************************************************************
*********************************************************************************
**********************************
ok: [centos7] => {
    "msg": "CentOS"
}
ok: [ubuntu] => {
    "msg": "Ubuntu"
}
ok: [localhost] => {
    "msg": "Ubuntu"
}

TASK [Print fact]
*********************************************************************************
*********************************************************************************
**********************************
ok: [centos7] => {
    "msg": "el default fact"
}
ok: [ubuntu] => {
    "msg": "deb default fact"
}
ok: [localhost] => {
    "msg": "PaSSw0rd"
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
*************************************
centos7                    : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
localhost                  : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
ubuntu                     : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

4. Добавьте новую группу хостов fedora, самостоятельно придумайте для неё переменную.

```
$ ansible-playbook -i inventory/prod.yml site.yml --ask-vault-pass
Vault password:

PLAY [Print os facts]
*********************************************************************************
*********************************************************************************
****************************

TASK [Gathering Facts]
*********************************************************************************
```

```
*********************************************************************************
****************************
ok: [localhost]
ok: [fedora]
ok: [ubuntu]
ok: [centos7]

TASK [Print OS]
*********************************************************************************
*********************************************************************************
**********************************
ok: [centos7] => {
    "msg": "CentOS"
}
ok: [ubuntu] => {
    "msg": "Ubuntu"
}
ok: [localhost] => {
    "msg": "Ubuntu"
}
ok: [fedora] => {
    "msg": "Fedora"
}

TASK [Print fact]
*********************************************************************************
*********************************************************************************
*********************************
ok: [centos7] => {
    "msg": "el default fact"
}
ok: [ubuntu] => {
    "msg": "deb default fact"
}
ok: [fedora] => {
    "msg": "fed default fact"
}
ok: [localhost] => {
    "msg": "PaSSw0rd"
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
*************************************
centos7                    : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
fedora                     : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
localhost                  : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
ubuntu                     : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

5. Напишите скрипт на bash: автоматизируйте поднятие необходимых контейнеров, запуск ansible-playbook и остановку контейнеров.

```
$ ./script.sh
[+] Running 4/4
 ⠿ Network 08-ansible-01-base_default  Created
0.1s
```

```
 ⠿ Container ubuntu                    Started
1.3s
 ⠿ Container centos7                   Started
1.3s
 ⠿ Container fedora                    Started
1.2s
Vault password:

PLAY [Print os facts]
*********************************************************************************
*********************************************************************************
*****************************

TASK [Gathering Facts]
*********************************************************************************
*********************************************************************************
*****************************
ok: [localhost]
ok: [fedora]
ok: [ubuntu]
ok: [centos7]

TASK [Print OS]
*********************************************************************************
*********************************************************************************
*********************************
ok: [centos7] => {
    "msg": "CentOS"
}
ok: [ubuntu] => {
    "msg": "Ubuntu"
}
ok: [localhost] => {
    "msg": "Ubuntu"
}
ok: [fedora] => {
    "msg": "Fedora"
}

TASK [Print fact]
*********************************************************************************
*********************************************************************************
*********************************
ok: [centos7] => {
    "msg": "el default fact"
}
ok: [ubuntu] => {
    "msg": "deb default fact"
}
ok: [fedora] => {
    "msg": "fed default fact"
}
ok: [localhost] => {
    "msg": "PaSSw0rd"
}

PLAY RECAP
*********************************************************************************
*********************************************************************************
*************************************
centos7                    : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

```
fedora                     : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
localhost                  : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
ubuntu                     : ok=3    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0

[+] Running 4/4
 ⠿ Container fedora                    Removed
1.5s
 ⠿ Container centos7                   Removed
1.5s
 ⠿ Container ubuntu                    Removed
1.5s
 ⠿ Network 08-ansible-01-base_default  Removed
```

6. Все изменения должны быть зафиксированы и отправлены в вашей личный репозиторий.

24.9 Системы оркестрации (Kubernets)

Лабораторная работа 1. Модули

В Kubernetes группа из одного или нескольких контейнеров называется pod. Контейнеры в модуле развертываются вместе и запускаются, останавливаются и реплицируются как группа. Простейшее определение pod описывает развертывание одного контейнера. Например, модуль веб-сервера nginx может быть определен как такой pod-nginx.yaml файл

```
apiVersion: v1
вид: метаданные
Pod:
 имя: пространство имен
  nginx:
 метки:
 запуск: спецификация
nginx:
 контейнеры:
 - имя: nginx
    изображение: nginx: последние
    порты:
 - Контейнерный порт: 80
```

Определение модуля - это объявление желаемого состояния. Желаемое состояние - очень важная концепция в модели Kubernetes. Многие вещи представляют желаемое состояние системы, и ответственность Kubernetes заключается в том, чтобы убедиться, что текущее состояние соответствует желаемому состоянию. Например, когда вы создаете модуль, вы заявляете, что хотите, чтобы контейнеры в нем были запущены. Если контейнеры не запущены (например, сбой программы), Kubernetes продолжит (повторно) создавать их для вас, чтобы привести их в желаемое состояние. Этот процесс продолжается до тех пор, пока модуль не будет удален.

Создайте модуль, содержащий сервер nginx

```
kubectl create -f pod-nginx.yaml
```

Перечислите все модули:

```
kubectl get pods -o wide

NAME      READY     STATUS    RESTARTS   AGE       IP            NODE
nginx     1/1       Running   0          8m        172.30.21.2   kuben03
```

Опишите модуль:

```
kubectl describe pod nginx

Name:           nginx
Namespace:      default
Node:           kuben03/10.10.10.83
Start Time:     Wed, 05 Apr 2017 11:17:28 +0200
Labels:         run=nginx
Status:         Running
IP:             172.30.21.2
Controllers:    <none>
Containers:
  nginx:
    Container ID:
docker://a35dafd66ac03f28ce4213373eaea56a547288389ea5c901e27df73593aa5949
    Image:              nginx:latest
    Image ID:           docker-pullable://docker.io/nginx
    Port:               80/TCP
    State:              Running
      Started:          Wed, 05 Apr 2017 11:17:37 +0200
    Ready:              True
    Restart Count:      0
    Volume Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-92z22 (ro)
    Environment Variables:      <none>
Conditions:
  Type          Status
  Initialized   True
  Ready         True
  PodScheduled  True
Volumes:
  default-token-92z22:
    Type:       Secret (a volume populated by a Secret)
    SecretName: default-token-92z22
QoS Class:      BestEffort
Tolerations:    <none>
Events:
...
```

Удалить модуль:

```
kubectl delete pod nginx
```

Модуль может находиться в одной из следующих фаз:

Ожидание: сервер API создал ресурс pod и сохранил его в etcd, но модуль еще не был запланирован, и изображения контейнеров не были извлечены из реестра.

Запуск: модуль запланирован на узел, и все контейнеры были созданы kubelet.

Успешно: все контейнеры в модуле успешно завершили работу и не будут перезапущены.

Сбой: все контейнеры в модуле завершились, и по крайней мере один контейнер завершился сбоем.

Неизвестно: Серверу API не удалось запросить состояние модуля, обычно из-за ошибки при обмене данными с kubelet.

Лабораторная работа 2. Ярлыки

В Kubernetes метки представляют собой систему для организации объектов в группы. Метки - это пары ключ-значение, которые прикреплены к каждому объекту. Селекторы меток могут быть переданы вместе с запросом на сервер apiserver для получения списка объектов, которые соответствуют этому селектору меток.

Чтобы добавить метку в модуль, добавьте раздел labels в разделе metadata в определении модуля:

```
apiVersion: v1
вид: метаданные
Pod:
 метки:
 запустить: nginx
...
```

Чтобы пометить работающий модуль

```
kubectl label pod nginx type=webserver
```

Для составления списка модулей на основе меток

```
kubectl get pods -l type=webserver

NAME      READY     STATUS    RESTARTS   AGE
nginx     1/1       Running   0          21m
```

Метки могут быть применены не только к модулям, но и к другим объектам Kuberntes, таким как узлы. Например, мы хотим маркировать рабочие узлы на основе их положения в центре обработки данных

```
kubectl label node kuben01 rack=rack01

kubectl get nodes -l rack=rack01

NAME      STATUS    AGE
kuben01   Ready     2d

kubectl label node kuben02 rack=rack01

kubectl get nodes -l rack=rack01

NAME      STATUS    AGE
kuben01   Ready     2d
kuben02   Ready     2d
```

Метки также используются в качестве селектора для служб и развертываний.

Лабораторная работа 3. Наборы реплик

Набор реплик гарантирует, что одновременно выполняется указанное количество копий pod. Другими словами, набор реплик гарантирует, что модуль или однородный набор модулей всегда готов и доступен. Если модулей слишком много, некоторые из них будут уничтожены. Если их слишком мало, начнется больше. В отличие от созданных вручную модулей,

модули, поддерживаемые набором реплик, автоматически заменяются, если они выходят из строя, удаляются или завершаются.

Конфигурация набора реплик состоит из:

- Желаемое количество реплик
- Определение модуля
- Селектор для привязки управляемого модуля

Селектор - это метка, присвоенная модулям, которые управляются набором реплик. Метки включены в определение модуля, который создается набором реплик. Набор реплик использует селектор, чтобы определить, сколько экземпляров модуля pod уже запущено, чтобы при необходимости настроить.

В nginx-rs.yaml файле определите набор реплик с помощью replica 1 для нашего модуля nginx.

```
apiVersion: extensions/v1beta1
вид:
 ReplicaSet-ов -реплик:
 метки:
 выполнить: пространство имен
  nginx:
 название:
 nginx-rs:
 реплики: 3
  селектора:
 Соответствующие метки:
 запустить: шаблон
  nginx:
 метаданные:
 метки:
 запуск: спецификация
    nginx:
 контейнеры:
 - изображение: nginx:1.12
        imagePullPolicy: Всегда
        имя: nginx
        порты:
 - Контейнерный порт: 80
          протокол: TCP
      Политика перезапуска: всегда
```

Создайте набор реплик

```
kubectl create -f nginx-rs.yaml
```

Перечислите и опишите набор реплик

```
kubectl get rs
NAME      DESIRED   CURRENT   READY     AGE
nginx     3         3         3         1m

kubectl describe rs nginx
Name:           nginx
Namespace:      default
Image(s):       nginx:latest
Selector:       run=nginx
Labels:         run=nginx
Replicas:       3 current / 3 desired
Pods Status:    3 Running / 0 Waiting / 0 Succeeded / 0 Failed
No volumes.
```

Набор реплик позволяет легко увеличивать или уменьшать количество реплик вручную или с помощью агента управления автоматическим масштабированием, просто обновляя поле replicas. Например, уменьшите количество реплик до нуля, чтобы удалить все модули, управляемые данным набором реплик

```
kubectl scale rs nginx --replicas=0

kubectl get rs nginx
NAME      DESIRED   CURRENT   READY     AGE
nginx     0         0         0         7m
```

Масштабируйте набор реплик для создания новых модулей

```
kubectl scale rs nginx --replicas=9

kubectl get rs nginx
NAME      DESIRED   CURRENT   READY     AGE
nginx     9         9         0         9m
```

Также в случае сбоя узла набор реплик заботится о сохранении того же количества модулей, планируя запуск контейнеров на отказавшем узле для остальных узлов в кластере.

Чтобы удалить набор реплик

```
kubectl delete rs/nginx
```

Удаление набора реплик удаляет все модули, управляемые этой репликой. Но, поскольку модули, созданные контроллером, на самом деле не являются неотъемлемой частью набора репликации, а только управляются им, мы можем удалить только набор репликации и оставить модули запущенными.

```
kubectl delete rs/nginx --cascade=false
```

Теперь нет ничего, что управляло бы модулями, но мы всегда можем создать новый набор репликации с соответствующим выбором меток и снова сделать их управляемыми.

Один набор реплик может одновременно сопоставлять модули с меткой env=production и модули с меткой env=dev. Также мы можем сопоставить все модули, которые содержат метку с ключом run, независимо от его фактического значения, действуя как нечто подобное run=*.

```
...
селектор:
 Выражения соответствия:
 - ключ: выполнить
       оператор: В
       значениях:
 - nginx
           web
```

Лабораторная работа 3. Развертывания

Развертывание предоставляет декларативные обновления для модулей и реплик. Объект развертывания определяет стратегию перехода между развертываниями одного и того же приложения.

В основном существует два способа обновления приложения:

- сначала удалите все существующие модули, а затем запустите новые или
- запустите новые и, как только они будут запущены, удалите старые

Последнее можно сделать с помощью двух разных подходов:

- добавьте все новые модули, а затем удалите все старые сразу
- добавляйте новые модули по времени, а затем удаляйте старые один за другим

Чтобы создать развертывание для нашего веб-сервера nginx, отредактируйте nginx-deploy.yaml файл как apiVersion: приложения /

```
v1: метаданные
развертывания:
 метки:
 имя: пространство имен
  nginx:
спецификация:
 Минимальные секунды: 10
  переходных предельных секунд: 300
  ревизийисторический предел: 3
  реплики: 6
  селектора:
 Соответствующие метки:
 запуск: стратегия
  nginx:
 Обновление:
 Максимальный всплеск: 1
      maxUnavailable: 1
    тип: RollingUpdate
  шаблон:
 метаданные:
 метки:
 запуск: спецификация
    nginx:
 контейнеры:
 - изображение: nginx: 1.12
        имя:
        nginx:
 - Контейнерный порт: 80
          протокол: TCP
        проверка готовности:
 HttpGet:
 путь: /
            порт: 80
            схема: HTTP
          Начальные задержки секунд: 30
          Время ожидания секунд: 10
          секунд периода: 5
          Порог успеха: 3
          Порог сбоя: 1
      Политика перезапуска: Всегда
```

и создайте развертывание

```
kubectl create -f nginx-deploy.yaml
deployment "nginx" created
```

При развертывании создаются следующие объекты

```
kubectl get all -l run=nginx -o wide

NAME           DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES      S
```

```
deploy/nginx   3         3         3            3           37s   nginx
nginx:1.12   run=nginx

NAME                  DESIRED   CURRENT   READY   AGE    CONTAINERS   IMAGES .
SELECTOR
rs/nginx-698d6b8c9f   3         3         3       37s    nginx        nginx:1.12
pod-template

NAME                          READY     STATUS    RESTARTS   AGE       IP
NODE
po/nginx-698d6b8c9f-cj9n6   1/1       Running   0          37s       10.38.4.200
kubew04
po/nginx-698d6b8c9f-sr6fh   1/1       Running   0          37s       10.38.5.137
kubew05
po/nginx-698d6b8c9f-vpsm4   1/1       Running   0          37s       10.38.3.125
kubew03
```

В соответствии с определениями, приведенными в приведенном выше файле, существует развертывание, три модуля и набор реплик.

Развертывание определяет стратегию для модулей обновлений

```
  СТРАТЕГИИ:
 Обновление:
 Максимальный всплеск: 1
      maxUnavailable: 1
    тип: RollingUpdate
```

В приведенном выше фрагменте мы устанавливаем стратегию обновления как непрерывное обновление. В течение срока службы приложения некоторые модули необходимо обновлять, например, из-за изменения изображения. Стратегия последовательного обновления удаляет некоторые старые модули, одновременно добавляя новые, сохраняя приложение доступным во время процесса и гарантируя, что не будет недостатка в обработке запросов пользователя. Это стратегия по умолчанию.

Верхний и нижний предел для количества модулей выше или ниже желаемого количества реплик настраиваются с помощью maxSurge и maxUnavailable параметров. Первый определяет, сколько экземпляров pod существует сверх требуемого количества реплик, настроенного при развертывании. По умолчанию значение равно 1, что означает, что может быть не более одного экземпляра pod больше, чем желаемое количество. Например, если желаемое количество реплик установлено равным 3, во время обновления никогда не будет запущено более 4 экземпляров pod одновременно.

Второй параметр определяет, сколько модулей может быть недоступно ниже требуемого количества реплик во время обновления. Также по умолчанию используется значение 1, что означает, что количество доступных экземпляров pod никогда не должно быть меньше 1, чем желаемое количество реплик. Например, если желаемое количество реплик установлено равным 3, всегда будет доступно как минимум 2 экземпляра pod для обслуживания запросов в течение всего развертывания.

Например, для обновления модулей с помощью другой версии образа nginx

```
kubectl set image deploy nginx nginx=nginx:1.13
```

Проверьте статус развертывания, пока это происходит

```
kubectl rollout status deploy nginx
```

Если вы хотите приостановить развертывание

```
kubectl rollout pause deploy nginx
```

Возобновите его, чтобы завершить

```
kubectl rollout resume deploy nginx
```

Теперь появился новый набор реплик, который теперь контролирует модули. Этот набор реплик управляет новыми модулями, имеющими изображение nginx:1.13. Старый набор реплик все еще существует и может быть использован в случае понижения версии.

Чтобы понизить версию, отмените развертывание

```
kubectl rollout undo deploy nginx
```

Это приведет к возвращению развертывания к предыдущему состоянию.

При создании новых модулей стратегия перехода ожидает, когда модули станут готовы. Если новые модули никогда не будут готовы, время ожидания развертывания истечет и приведет к сбою развертывания.

Стратегия развертывания использует проверку готовности, чтобы определить, готов ли новый модуль к использованию. Если проверка готовности завершается неудачей, развертывание останавливается. minReadySeconds Свойство указывает, как долго новый созданный модуль должен быть готов, прежде чем модуль будет считаться доступным. Пока модуль не станет доступным, процесс обновления продолжаться не будет. Как только проверка готовности завершится успешно и модуль станет доступным, процесс обновления может быть продолжен. С правильно настроенным зондом готовности и надлежащими minReadySeconds настройками kubernetes не позволяет нам развертывать версию образа с ошибками.

Например, давайте улучшим развертывание до версии образа с ошибками. Эта новая версия образа не работает должным образом, и, следовательно, процесс обновления завершится ошибкой при обновлении первого модуля.

Обновите изображение

kubectl set image deploy nginx nginx=kalise/nginx:buggy

Проверьте статус развертывания

kubectl rollout status deploy nginx

Waiting for rollout to finish: 1 out of 6 new replicas have been updated...

Из-за проверки готовности процесс обновления заблокирован для первого модуля, в котором запущена версия с ошибками. Из-за зависания развертывания мы не можем получить полностью нерабочее приложение. По умолчанию, после того, как развертывание не может добиться какого-либо прогресса в течение 600 секунд, оно считается неудачным

```
kubectl rollout status deploy nginx
Waiting for rollout to finish: 1 out of 6 new replicas have been updated...
error: deployment "nginx" exceeded its progress deadline
```

Этот таймер управляется progressDeadlineSeconds свойством, установленным в дескрипторе развертывания.

Поскольку процесс обновления никогда не продолжится, единственное, что нужно сделать сейчас, это прервать развертывание

```
kubectl rollout undo deployment nginx
```

Развертывание, может быть увеличено и уменьшено

```
kubectl scale deploy nginx --replicas=6
deployment "nginx" scaled

kubectl get deploy nginx
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx     6         6         6            3           11m
```

При развертывании модули всегда управляются набором реплик. Однако, поскольку набор реплик контролируется развертыванием, если мы попытаемся масштабировать набор реплик вместо развертывания, развертывание будет иметь приоритет, и количество модулей будет сообщено о количестве, запрошенном при развертывании.

Например, попробуйте увеличить набор реплик из предыдущего примера до 10 реплик

kubectl scale rs nginx-698d6b8c9f --replicas=10

мы видим, что количество модулей увеличено до 10 в соответствии с запросом на масштабирование набора реплик до 10 pod. Через несколько секунд развертывание получит приоритет и удалит все новые модули, созданные путем масштабирования набора реплик, поскольку желаемый stae, указанный при развертывании, равен 6 модулям.

24.10 Проектный практикум

Что необходимо для сдачи проекта:

1. Репозиторий с конфигурационными файлами Terraform и готовность продемонстрировать создание всех ресурсов с нуля.
2. Пример pull request с комментариями созданными atlantis'ом или снимки экрана из Terraform Cloud.
3. Репозиторий с конфигурацией ansible, если был выбран способ создания Kubernetes кластера при помощи ansible.
4. Репозиторий с Dockerfile тестового приложения и ссылка на собранный docker image.
5. Репозиторий с конфигурацией Kubernetes кластера.
6. Ссылка на тестовое приложение и веб интерфейс Grafana с данными доступа.

   Все репозитории рекомендуется хранить на одном ресурсе (github, gitlab)

24.11 Практика /Стажировка

Оценочные средства

Текущий контроль успеваемости осуществляется в течение периода прохождения практики, включает контроль самостоятельной работы обучающихся в письменной форме.

Промежуточная аттестация по практике осуществляется в форме зачета с оценкой, которая проводится, в форме публичной защиты отчета по практике.

Отчет по практике является основным документом, характеризующим работу обучающегося во время практики. Отчет составляется в соответствии с программой практики и в общем виде содержит следующие разделы:

Готовый отчет формируется в следующем порядке:

1. титульный лист отчета ;
2. задание на практику;
3. оглавление отчета;
4. текст отчета (по разделам);
5. приложения.

25. Промежуточная аттестация. Перечень примерных заданий

25.1 Наименование дисциплины - Командные коммуникации и системы управления задачами

Вопросы к тесту:

1. Что такое JIRA Issue?
2. Компоненты JIRA
3. Экран JIRA
4. Как создать проблему в JIRA?
5. Подзадача
6. Что такое Agile?
7. Манифест и принципы Agile
8. Преимущества и недостатки Agile
9. JIRA Agile
10. Создание проблемы в Agile
11. Как создать эпик в Agile?
12. Отчеты в JIRA
13. Канбан

25.2 Наименование дисциплины - Системы контроля версий

Перечень вопросов:

1. Какой тип систем контроля версий сейчас используется чаще всего?
2. Выберите действие, которое выполняет команда git?
3. Какая команда git используется для клонирования удаленного репозитория на локальный компьютер?
4. Для чего используется ключ SSH на gitlab?
5. Какая команда git используется для получения изменений с удаленного репозитория на локальный компьютер?
6. Настройка continuous integration в gitlab
7. Как создать баг-репорта в gitlab?

25.3 Наименование дисциплины - Организация высокой доступности БД

Перечень вопросов

1. Понятие логической резервной копии
2. Копирование и восстановление отдельных таблиц
3. Копирование и восстановление баз данных
4. Копирование и восстановление кластера
5. Понятие логической резервной копии
6. Копирование и восстановление отдельных таблиц
7. Копирование и восстановление баз данных
8. Копирование и восстановление кластера
9. Понятие логической резервной копии
10. Копирование и восстановление отдельных таблиц
11. Копирование и восстановление баз данных
12. Копирование и восстановление кластера
13. Понятие логической резервной копии
14. Копирование и восстановление отдельных таблиц
15. Копирование и восстановление баз данных
16. Копирование и восстановление кластера
17. Понятие физической резервной копии
18. Холодное резервирование
19. Горячее резервирование
20. Файловый архив — непрерывная архивация
21. Потоковый архив — утилита pg_receivewal
22. Восстановление с использованием архива
23. Очистка архива
24. Задачи репликации
25. Схема работы физической репликации
26. Способы доставки журнальных записей
27. Особенности и ограничения использования реплики
28. Синхронная и асинхронная репликация
29. Мониторинг репликации
30. Возможные проблемы и способы их решения
31. Переключение на реплику
32. Возвращение в строй бывшего мастера
33. Особенности, связанные с файловым архивом
34. Отличия логической репликации от физической
35. Публикации и подписки
36. Логическое декодирование и слоты логической репликации
37. Конфликты и их разрешение
38. Выполнение триггеров на подписчике
39. Использование физической репликации
40. Использование логической репликации
41. Ожидания от кластера

42. Средства реализации
43. Решения с реализацией внутри PostgreSQL
44. Решения с внешними системами управления

25.4 Наименование дисциплины - Использование Python для DevOps

Задание на контрольную работу:

Имеется приложение hello-world написанное на Python + FastAPI, которое отдает строку "Hello World <окружение>!".

Задачи:

1. Завести публичный репозиторий на GitHub или аналогах, в который скопировать весь проект из директории devops и продолжать работу в нем
2. Завернуть это приложение в Dockerfile, в котором
   - Установятся зависимости через poetry (установка через poetry)
   - Будет запускаться приложение через web-сервер gunicorn (gunicorn.sh)
3. Написать helm chart. в котором
   - Configmap для переменных окружения (ENVIRONMENT)
     a. Переменная ENVIRONMENT должна иметь значения: dev, stage, prod в зависимости от окружения
   - Secret для секретных переменных окружения (ENVIRONMENT_FROM_SECRET)
     b. Секреты должны шифроваться любым способом доступным для k8s (например: helm-secrets)
     c. Секрет ENVIRONMENT_FROM_SECRET должен иметь значения: secret_dev, secret_stage, secret_prod в зависимости от окружения
   - Deployment, который будет
     d. Запускать 2 реплики приложения
     e. Работать на 8000 порте
     f. Прокидывать в контейнер ConfigMap и Secret
     g. Иметь readiness и liveness probes по эндпоинту /healthcheck
   - Service типа ClusterIP
   - Ingress для nginx сервера, который должен сконфигурирован для хостов
     h. dev - localhost
     i. stage - farforstaging.ru
     j. prod - farfor.ru
   - CertManager в котором
     k. Issuer для выписывания сертификатов acme-letsencrypt через http01.ingess
     l. Certificate - соответствующий сертификат связанный с Issuer для доменов окружений stage и prod
   - HorizontalPodAutoscaler, который будет скейлить реплики от 2 до 4
   - _helpers.tpl, в котором определить шаблоны для

m. selectorlabels, в котором определить селекторные метки app и release
n. labels включающий в себя selectorLabels, в котором определить общие метки chart и version + selectorLabels и использовать шаблоны в нужных местах в чарте

– helmfile для трех (dev, stage, prod) окружений
– Максимально использовать общие переменные и переиспользовать переменные окружений по необходимости
– Написать полную пошаговую инструкцию по сборке приложения и запуску через minikube

Приложение должно:

1. Собираться через Dockerfile
2. Запускаться через minikube
3. Работать по адресу http://0.0.0.0:8000/ и отображать значение переменной ENVIRONMENT в ответе
4. По адресу http://0.0.0.0:8000/docs отображать значение переменной ENVIRONMENT_FROM_SECRET в заголовке страницы

25.5 Наименование дисциплины Администрирование Linux

Примеры тестов

Какая команда позволяет просмотреть справочную информацию или руководства?

- man
- which
- what
- mal

Какой командой следует запустить скрипт script1.sh, находящийся в каталоге /etc/dir1/, если работа пользователя в данный момент происходит в /home/Liza/dir2?

- ./scrip1.sh
- /etc/dir1/script1.sh
- ../../../../script1.sh
- run script1.sh

Теоретические вопросы:

1. Какой дистрибутив Linux отечественного производства полностью совместим с Windows?

2. Возможен ли интерактивный вход в систему суперпользователя root по умолчанию после установки?

3. В какую группу по умолчанию включается создаваемый при установке ОС Astra Linux пользователь?

4. Какой механизм позволяет созданному при установке ОС Astra Linux пользователю проводить настройку системы, требующую привилегий root?

5. Какой тип сессии необходимо установить для загрузки стандартного рабочего стола ОС необходимо при графическом входе в ОС?

6. Какую команду следует использовать для завершения сессии в консольном режиме?

7. Описать, что означают термины: файл, каталог.

8. Написать регулярное выражение для поиска всех файлов в системе размером более 500 МБ

9. Подсчитать количество строк, в которых содержится пользователь user в файле /etc/group

10. Какая файловая система используется в Astra Linux?

11. В какую группу включается создаваемый при установке операционной системы пользователь?

12. Как проверить соединение компьютера с другими устройствами в сети?

13. С помощью какой команды можно добавить нового пользователя в систему?

14. Пользователь был создан с использованием команды $ useradd student. В какой директории окажется student после того, как войдет в систему?

15. С помощью какой команды можно посмотреть наличие и настройки сетевых интерфейсов?

16. Что необходимо указать для настройки интерфейса сетевой платы?

17. Какие параметры имеет каждый зарегистрированный пользователь?

18. Какие параметры необходимо указать в настройках сетевого интерфейса при статической адресации?

19. Как назначить права на чтение и исполнение для файла file.conf для всех остальных?

20. Измените права доступа на чтение запись и выполнение для группы файла file2?

21. Что является результатом выполнения данной команды chmod 755 file?

23. Что означает данная запись rw-r-xr-- ?

24. Как сделать file1 исполняемым?

25.6 Системы виртуализации и контейнеризации (Docker)

Задание на контрольную работу:

Перед выполнением необходимо запустить следующие команды:

```
# Остановка всех контейнеров
docker kill $(docker ps -a -q)
# Удаление всех контейнеров
docker rm $(docker ps -a -q)
# Удаление всех неиспользуемых сущностей (используемых в docker)
docker system prune -a
```

1. Написать программу на python3, которая будет выводить в консоль текущее время (каждые 5 секунд). Создать образ с данной программой и запустить контейнер. В результате, вы сможете увидеть в логах контейнера

текущее время. Дополнительно ввести проверку архитектуры аппаратного обеспечения, при использовании X86 - выводить сообщение об используемой архитектуре и время каждые 10 секунд, при x64 - каждые 7 секунд, при ARM - каждые 3 секунды. Дополнительно ввести в программу возможность вывода времени в виде временного штампа.

Дополнительная информация:

- В Dockerfile указать следующее `CMD`:
```
# Обязательно с флагом -u
CMD [ "python3", "-u", "main.py"]
```
- Используемый образ: `python:3.8-slim-buster`
- Команда для запуска контейнера:
```
# Флаг --rm автоматически удалит контейнер,
# если тот остановится
docker run -d --rm <название образа>
```
- команда для проверки логов:
```
docker logs <id контейнера>
# для получения id контецнера, используйте команду
# docker ps -a
```

2. Добавить в ПО значение величины задержки отображения (не каждые 5 секунд, а, например, каждые 10 секунд)

Дополнительная информация:

- Для передачи значения использовать `ENV` переменную с названием `TIME_SLEEP`;
- Для передачи значения переменной при запуске контейнера воспользоваться командой:
```
docker run -d --rm --env TIME_SLEEP=10 <название образа>
```

Перечень тестовых вопросов:

1. В чем 5 сходств между Docker и виртуальной машиной?
2. Чем Docker отличается от виртуальной машины?
3. В чем разница между контейнерной сетью и сетью виртуальных машин?
4. Можно ли запустить несколько процессов внутри контейнера Docker?
5. Работает ли Docker в Linux, macOS и Windows?
6. Что такое DockerHub?
7. Что такое Dockerfile?
8. Чем Dockerfile отличается от Docker Compose?
9. Могу ли я использовать JSON вместо YAML для своего файла Docker Compose?

10. Какое максимальное количество контейнеров можно запустить на хосте?

11. Можно ли иметь собственный частный реестр Docker?

12. Упаковывает ли контейнер Docker всю ОС?

13. Опишите, сколько способов доступно для настройки демона Docker?

14. На что ссылается CNM? Каковы его компоненты?

15. Какие существуют типы сетевых драйверов Docker?

16. Какие функции возможны только в Docker Enterprise Edition по сравнению с Docker Community Edition?

17. Чем сеть Docker Bridge отличается от традиционного моста Linux?

18. Как создать определяемую пользователем сеть Bridge?

19. Как удалить определяемую пользователем сеть Bridge?

20. Как подключить контейнер Docker к определяемой пользователем сети моста?

21. Поддерживает ли Docker IPv6?

22. Поддерживает ли формат файла Docker Compose протокол IPv6?

23. Чем оверлейная сеть отличается от мостовой сети?

24. Как отключить сетевой стек в контейнере?

25. Как создать сеть MacVLAN для контейнера Docker?

26. Можно ли исключить использование IP-адреса в сети MacVLAN?

27. Потеряю ли я свои данные при выходе из контейнера?

28. Поддерживает ли Docker Enterprise Edition Kubernetes?

29. Что такое Docker Swarm?

30. Что такое --memory-swapфлаг?

31. Можете ли вы объяснить различные типы монтирования томов, доступные в Docker?

32. Как обмениваться данными между DockerHost?

33. Как сделать резервную копию, восстановить или перенести тома данных в контейнере Docker?

34. Как настроить автоматические сборки на DockerHub

35. Как настроить драйвер ведения журнала по умолчанию в Docker?

36. Почему моим службам требуется 10 секунд для повторного создания или остановки?

37. Как запустить несколько копий файла Compose на одном хосте?

38. В чем разница между up, run и start в Docker Compose?

39. Что такое доверенный реестр Docker?

40. Как объявить переменные среды по умолчанию в Docker Compose?

41. Можете ли вы перечислить способы совместного использования конфигураций Compose между файлами и проектами в Docker Compose?

42. Какова роль файла .dockerignore?

43. Какова цель команды EXPOSE в Dockerfile?

44. Чем инструкция ENTRYPOINT в Dockerfile отличается от инструкции RUN?

45. Почему сборка кеша в Docker так важна?

46. Зачем нужен мониторинг Docker?

47. Разница между контейнерами Windows и контейнерами Hyper-V

48. В чем основное различие между Swarm и Kubernetes?

49. Можно ли запустить Kubernetes на платформе Docker EE 2.0?

50. Можно ли использовать Docker Compose для создания кластера Swarm/Kubernetes?

51. Для чего предназначена команда «docker stack deploy»?

52. Перечислите основные компоненты Docker EE 2.0?

53. Объясните концепцию HA в режиме Swarm?

25.7 Системы непрерывной интеграции (Jenkins)

Задание на контрольную работу:

1. Создайте образ контейнера с установленным Jenkins с помощью файла Docker.

2. Когда мы запускаем этот образ, сервис Jenkins автоматически запускается в контейнере.

3. Создайте цепочку заданий Job 1, Job 2, Job 3 и Job 4, используя подключаемый модуль конвейера сборки в Jenkins.

4. Задача 1. Автоматически извлекать репозиторий GitHub, когда разработчики отправляют код на GitHub.

5. Задача 2: просматривая код или программный файл, Jenkins должен автоматически запускать соответствующий языковой интерпретатор установленного контейнера изображения для развертывания кода (например: если код написан на PHP, то Jenkins запускает контейнер с уже установленным PHP).

6. Задача 3: проверьте свое приложение, работает оно или нет.

7. Задача 4: если приложение не работает, отправьте электронное письмо разработчику с сообщениями об ошибках.

8. Создайте дополнительное задание (задание 5) для мониторинга: если контейнер, в котором запущено приложение, по какой-либо причине не работает, то это задание должно автоматически запустить контейнер снова.

25.8 Системы управления конфигурацией. Инфраструктура как код (Ansible)

Вопросы к тестированию:

1) Что такое Ansible?

2) Какая польза от Ansible?

3) Что такое Ansible Galaxy?

4) Что такое непрерывная доставка?

5) Как получить доступ к переменным среды оболочки в Ansible?

6) Какой код нужно написать для доступа к имени переменной?

7) Объясните, как можно отключить cowsay?

8) Объясните, как вы можете копировать файл рекурсивно на целевой хост?

9) Как вы можете отправить изменения в Документацию в Ansible?

10) Каков наилучший способ сделать контент повторно используемым / распространяемым?

11) Что такое Ansible Tower?

12) Какой метод проверки инвентарных переменных, определенных для хоста?

13) Укажите разницу между именем переменной и переменными среды.

14) Что такое специальные команды?

15) Объясните Ansible факты

16) Как вы видите все переменные для хоста?

17) Объясните модули в ansible

18) Когда следует тестировать пьесы и роли?

25.9 Системы оркестрации (Kubernets)

Вопросы к тестированию:

1. Какой командой можно повысить привилегии своего пользователя?

- chown
- chmod
- sudo
- useradd

2. Как посмотреть сетевые интерфейсы системы?

- ifconfig / ip a
- ps a / pstree
- df -h
- nslookup

3. Какой командой посмотреть запущенные процессы в системе Linux?

- ls
- ps
- pwd
- find

4. Какой командой скопировать файл?

- file
- rm
- cp
- locate

5. Каким набором команд можно выяснить сетевую доступность инстанса?

- cp / mv
- ps / pstree/ kill
- df / dd / ionice
- ping / telnet

6. Как посмотреть количество оперативной памяти?

- free
- df
- history
- nano

7. Как собрать образ Docker?

- docker run
- docker build
- docker ps
- docker commit

8. Как называется инструкция сборки образа Docker?

- Docker Image
- Docker Registry
- Dockerfile
- Docker Daemon

9. Какой командой можно отправить образ Docker в Registry?

- docker push

- docker pull
- docker run
- docker exec

10. Какой командой можно посмотреть все контейнеры Docker?

- docker ps -a
- docker ps
- docker images
- docker images -a

11. Как зайти в консоль сервера?

- Набрать адрес сервера в браузере http://адрес_сервера
- Зайти к нему по ftp ftp://адрес_сервера
- Зайти на сервер по SSH ssh адрес_сервера
- Нужен обязательно физический доступ к серверу, по сети это сделать нельзя.

26. Итоговая аттестация

Итоговая аттестация проводится в виде демонстрационного экзамена, который представляет собой два этапа: разработку проекта на дисциплине «проектный практикум» и защиту проекта на итоговой аттестации. В ходе защиты демонстрируется реализация разработанного программного обеспечения и процесс его сборки.

Задание на проект:

7. Подготовить облачную инфраструктуру на базе облачного провайдера.
8. Запустить и сконфигурировать Kubernetes кластер.
9. Установить и настроить систему мониторинга.
10. Настроить и автоматизировать сборку тестового приложения с использованием Docker-контейнеров.
11. Настроить CI для автоматической сборки и тестирования.
12. Настроить CD для автоматического развёртывания приложения.

Этапы выполнения:

1. Создание облачной инфраструктуры

Для начала необходимо подготовить облачную инфраструктуру при помощи Terraform.

Особенности выполнения:

– бюджет купона ограничен, что следует иметь в виду при проектировании инфраструктуры и использовании ресурсов;

– следует использовать последнюю стабильную версию terraform.

Предварительная подготовка к установке и запуску Kubernetes кластера.

1. Создайте сервисный аккаунт, который будет в дальнейшем использоваться Terraform для работы с инфраструктурой с необходимыми и достаточными правами. Не стоит использовать права суперпользователя

2. Подготовьте backend для Terraform: а. Рекомендуемый вариант: Terraform Cloud; б. Альтернативный вариант: S3 bucket.

3. Настройте workspaces:

– Рекомендуемый вариант: создайте два workspace: *stage* и *prod*. В случае выбора этого варианта все последующие шаги должны учитывать факт существования нескольких workspace.

– Альтернативный вариант: используйте один workspace, назвав его *stage*. Пожалуйста, не используйте workspace, создаваемый Terraform-ом по-умолчанию (*default*).

4. Создайте VPC с подсетями в разных зонах доступности.

5. Убедитесь, что теперь вы можете выполнить команды terraform destroy и terraform apply без дополнительных ручных действий.

6. В случае использования Terraform Cloud в качестве backend убедитесь, что применение изменений успешно проходит, используя web-интерфейс Terraform cloud.

Ожидаемые результаты:

1. Terraform сконфигурирован и создание инфраструктуры посредством Terraform возможно без дополнительных ручных действий.

2. Полученная конфигурация инфраструктуры является предварительной, поэтому в ходе дальнейшего выполнения задания возможны изменения.

2. Создание Kubernetes кластера

На этом этапе необходимо создать Kubernetes кластер на базе предварительно созданной инфраструктуры. Требуется обеспечить доступ к ресурсам из Интернета.

Это можно сделать двумя способами:

1. Рекомендуемый вариант: самостоятельная установка Kubernetes кластера.

– При помощи Terraform подготовить как минимум 3 виртуальных машины Compute Cloud для создания Kubernetes-кластера. Тип виртуальной машины следует выбрать самостоятельно с учётом требовании к производительности и стоимости. Если в дальнейшем поймете, что необходимо сменить тип инстанса, используйте Terraform для внесения изменений

– Подготовить ansible конфигурации, можно воспользоваться, например Kubespray

– Задеплоить Kubernetes на подготовленные ранее инстансы, в случае нехватки каких-либо ресурсов вы всегда можете создать их при помощи Terraform.

2. С помощью terraform resource для kubernetes создать региональный мастер kubernetes с размещением нод в разных 3 подсетях. С помощью terraform resource для kubernetes node group.

Ожидаемый результат:

1. Работоспособный Kubernetes кластер.
2. В файле ~/.kube/config находятся данные для доступа к кластеру.
3. Команда kubectl get pods --all-namespaces отрабатывает без ошибок.

3. Создание тестового приложения

Для перехода к следующему этапу необходимо подготовить тестовое приложение, эмулирующее основное приложение разрабатываемое вашей компанией.

Способ подготовки:

1. Рекомендуемый вариант:

– Создайте отдельный git репозиторий с простым nginx конфигом, который будет отдавать статические данные.

– Подготовьте Dockerfile для создания образа приложения.

2. Альтернативный вариант: Используйте любой другой код, главное, чтобы был самостоятельно создан Dockerfile.

Ожидаемый результат:

1. Git репозиторий с тестовым приложением и Dockerfile.

2. Регистр с собранным docker image. В качестве регистра может быть DockerHub, созданный также с помощью terraform.

4. Подготовка системы мониторинга и деплой приложения

Уже должны быть готовы конфигурации для автоматического создания облачной инфраструктуры и поднятия Kubernetes кластера. Теперь необходимо подготовить конфигурационные файлы для настройки нашего Kubernetes кластера.

Цель:

1. Задеплоить в кластер prometheus, grafana, alertmanager, экспортер основных метрик Kubernetes.
2. Задеплоить тестовое приложение, например, nginx сервер отдающий статическую страницу.

Рекомендуемый способ выполнения:

1. Воспользовать пакетом kube-prometheus, который уже включает в себя Kubernetes оператор для grafana, prometheus, alertmanager и node_exporter. При желании можете собрать все эти приложения отдельно.
2. Для организации конфигурации использовать qbec, основанный на jsonnet. Обратите внимание на имеющиеся функции для интеграции helm конфигов и helm charts
3. Если на первом этапе вы не воспользовались Terraform Cloud, то задеплойте в кластер atlantis для отслеживания изменений инфраструктуры.

Ожидаемый результат:

1. Git репозиторий с конфигурационными файлами для настройки Kubernetes.
2. Http доступ к web интерфейсу grafana.
3. Дашборды в grafana отображающие состояние Kubernetes кластера.
4. Http доступ к тестовому приложению.

5. Установка и настройка CI/CD

Осталось настроить ci/cd систему для автоматической сборки docker image и деплоя приложения при изменении кода.

Цель:

1. Автоматическая сборка docker образа при коммите в репозиторий с тестовым приложением.

2. Автоматический деплой нового docker образа.

Можно использовать teamcity, jenkins либо gitlab ci

Ожидаемый результат:

3. Интерфейс ci/cd сервиса доступен по http.
4. При любом коммите в репозитории с тестовым приложением происходит сборка и отправка в регистр Docker образа.
5. При создании тега (например, v1.0.0) происходит сборка и отправка с соответствующим label в регистр, а также деплой соответствующего Docker образа в кластер Kubernetes.

Что необходимо для сдачи задания:

7. Репозиторий с конфигурационными файлами Terraform и готовность продемонстрировать создание всех ресурсов с нуля.
8. Пример pull request с комментариями созданными atlantis'ом или снимки экрана из Terraform Cloud.
9. Репозиторий с конфигурацией ansible, если был выбран способ создания Kubernetes кластера при помощи ansible.
10. Репозиторий с Dockerfile тестового приложения и ссылка на собранный docker image.
11. Репозиторий с конфигурацией Kubernetes кластера.
12. Ссылка на тестовое приложение и веб интерфейс Grafana с данными доступа.
13. Все репозитории рекомендуется хранить на одном ресурсе (github, gitlab)

XII. Материально-техническое и учебно-методическое обеспечение программы

Учебная аудитория № В-103.

Оснащение: 180 посадочных мест, доска аудиторная, акустическая система, проектор, усилитель-микшер для систем громкой связи, экран, микрофон, миникомпьютер, монитор, подключение к сети «Интернет», доступ в электронную информационно-образовательную среду.

Учебные аудитории № В-600 а, б.

Оснащение: 60 посадочных мест, компьютер (60шт.), проектор (2 шт.), экран (2 шт.) веб-камера (12 шт.), подключение к сети "Интернет", доступ в электронную информационно-образовательную среду

Учебные аудитории № Г-323-325.

Оснащение: 62 посадочных мест, компьютеры (62 штуки), подключение к сети «Интернет», доступ в электронную информационно-образовательную среду.

Программное обеспечение

1. Мой Офис - https://myoffice.ru/products/education/
2. PDF Commander - https://pdf-editor.su/thanks-downloading.php
3. Yandex Browser- https://browser.yandex.ru/?&banerid=0500000134
4. Astra Linux Common Edition -https://astralinux.axoft.ru/selectproduct?tag=1128
5. EvaProject- https://www.evateam.ru/registration/
6. GitLab/GitHub- https://github.com/
7. Visual Studio 2019 Comm – https://visualstudio.microsoft.com/ru/downloads/
8. ElephantSQL - PostgreSQL as a Service - https://www.elephantsql.com/
9. PostgreSQL для Windows - https://postgrespro.ru/windows
10. Jenkins- https://www.jenkins.io/download/
11. TeamCity- https://www.jetbrains.com/ru-ru/teamcity/download/
12. ОС Red Hat, Debian, CentOS- https://www.centos.org/download/

XIII. Список литературы

1. Кознов, Д. В.; Введение в программную инженерию: учебное пособие.; Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, Москва, Саратов; 2020; http://www.iprbookshop.ru/89428.html.

2. Baarsen J. van. GitLab Cookbook. - Packt Publishing, 2014. - 172 с.

3. Sagar R. Jira Quick Start Guide: Manage your projects efficiently using the all-new Jira. - Packt Publishing, 2019. - 162 с.

4. Kuruvilla J. JIRA Development Cookbook. - 3rd. - Packt Publishing, 2016. - 598 с.

5. Проектирование информационных систем [Электронный ресурс]: учебное пособие / В. И. Грекул, Г. Н. Денищенко, Н. Л. Коровкина. – 2-е изд. - Электрон. текстовые дан. - М.: Национальный открытый университет

«ИНТУИТ», 2016. - 570 с. - Режим доступа: https://e.lanbook.com/book/100391.

6. Моргунов, Е. П. М79 PostgreSQL. Основы языка SQL: учеб. пособие [Электронный ресурс] / Е. П. Моргунов; под ред. Е. В. Рогова, П. В. Лузанова. - СПб.: БХВ-Петербург, 2018. - 336 с.: ил. URL: https://edu.postgrespro.ru/sql_primer.pdf .

7. Новиков Б. А. Основы технологий баз данных: учеб. пособие [Электронный ресурс] / Б. А. Новиков, Е. А. Горшкова, Н. Г. Графеева; под ред. Е. В. Рогова. - 2-е изд. - М.: ДМК Пресс, 2020. - 582 с. URL: https://edu.postgrespro.ru/dbtech.pdf.

8. Руководство по PostgreSQL [Электронный ресурс] // Metanit.com: [сайт]. [2022]. URL: https://metanit.com/sql/postgresql/

9. Орлов С.А. Теория и практика языков программирования [Электронный ресурс]: учебник / С. А. Орлов. - 2-е изд. - Электрон. текстовые дан. - СПб.: Питер, 2017. - 688 с. - Режим доступа: https://ibooks.ru/reading.php?productid=355466.

10. Плас, Дж. Вандер. Python для сложных задач: наука о данных и машинное обучение [Электронный ресурс] / Дж. Вандер Плас. - Электрон. текстовые дан. - СПб.: Питер, 2018. - 576 с. - Режим доступа: https://ibooks.ru/reading.php?productid=356721.

11. Лутц М. Программирование на Python, том I, 4-е издание. Пер. С англ. – СПб.: Символ-Плюс, 2014. – 992 с.

12. Лутц М. Программирование на Python, том II, 4-е издание. Пер. С англ. – СПб.: Символ-Плюс, 2014. – 992 с.

13. Дэвис, К. Шаблоны проектирования для облачной среды: руководство / К. Дэвис; перевод с английского Д. А. Беликова. - Москва: ДМК Пресс, 2020. - 388 с. - ISBN 978-5-97060-807-4. - Текст: электронный // Лань: электронно-библиотечная система https://e.lanbook.com/book/140593

14. Лукша, М. Kubernetes в действии / М. Лукша ; перевод с английского А. В. Логунов. - Москва: ДМК Пресс, 2019. - 672 с. - ISBN 978-5-97060-657-5. - Текст: электронный https://e.lanbook.com/book/131688

15. Маркелов, А. А. Введение в технологию контейнеров и Kubernetes / А. А. Маркелов. - Москва: ДМК Пресс, 2019. - 194 с. - ISBN 978-5-97060-775-6. - Текст: электронный // Лань: электронно-библиотечная система. https://e.lanbook.com/book/131702

16. Кочер, П. С. Микросервисы и контейнеры Docker: руководство / П. С. Кочер ; перевод с английского А. Н. Киселева. - Москва: ДМК Пресс, 2019. - 240 с. - ISBN 978-5-97060-739-8. - Текст: электронный // Лань: электронно-библиотечная система. https://e.lanbook.com/book/123710

17. Партыка Т.Л. Операционные системы, среды и оболочки / Т.Л. Партыка, И.И. Попов. - Москва: Форум, 2017. - 560 с. - URL: https://ibooks.ru/bookshelf/361454/reading.

18. Гунько А.В. Системное программирование в среде Linux: учебное пособие / А.В. Гунько. - Новосибирск: Новосибирский государственный технический университет, 2020. - 235 с. - URL: https://ibooks.ru/bookshelf/372316/reading.

19. Вавренюк А.Б. Операционные системы. Основы UNIX / А.Б. Вавренюк, О.К. Курышева, С.В. Кутепов. - Москва: Инфра-М, 2021. - 160 с. - URL: https://ibooks.ru/bookshelf/360745/reading.

20. Фуско Дж. Linux. Руководство программиста. - Санкт-Петербург: Питер, 2021. - 448 с. - URL: https://ibooks.ru/bookshelf/377961/reading.

21. Кетов Д. В. Внутреннее устройство Linux. - 2-е изд., перераб. и доп. / Д.В. Кетов. - Санкт-Петербург: БХВ-Петербург, 2021. - 400 с. - URL: https://ibooks.ru/bookshelf/380032/reading (дата обращения: 29.03.2023).

22. Ной Гифт. Python и DevOps: Ключ к автоматизации Linux. - (Серия «Бестселлеры O’Reilly»). - Санкт-Петербург: Питер, 2022. - 544 с. - URL: https://ibooks.ru/bookshelf/379920/reading (дата обращения: 29.03.2023).

Электронные и интернет-ресурсы

№ п/п	Наименование электронных и интернет-ресурсов	Ссылка
1	ИНТУИТ. Национальный Открытый Университет.	https://www.intuit.ru/
2	Agile-манифест разработки программного обеспечения	https://agilemanifesto.org/iso/ru/manifesto.html
3	Репозиторий образов Docker	Docker.io / docker/containerd 1.6+
4	Репозиторий образов Kubernetes	k8s.gcr.io
5	Электронный университет КГЭУ - виртуальная образовательная среда.	https://lms.kgeu.ru/course/view.php?id=2663
6	Предустановленные образы виртуальных машин	http://osboxes.org

П Р О Т О К О Л

заседания Президиума по рассмотрению дополнительных профессиональных программ (программ профессиональной переподготовки) ИТ-профиля, реализуемых на «цифровых кафедрах» в рамках федерального проекта «Развитие кадрового потенциала ИТ-отрасли» национальной программы «Цифровая экономика Российской Федерации» (далее – Президиум)

г. Москва

11 июля 2023 г. № 1

Председательствовал: - директор Департамента развития цифровых компетенций и образования Т.Н. Трубникова

Присутствовали:

от Министерства цифрового развития, связи и массовых коммуникаций Российской Федерации

Казанцева Анастасия Юрьевна - заместитель директора Департамента развития цифровых компетенций и образования

от Министерства науки и высшего образования Российской Федерации

Гришкин Виталий Викторович - директор Департамента координации деятельности образовательных организаций

Богоносов Константин Александрович - заместитель директора Департамента координации деятельности образовательных организаций

от АНО «Цифровая экономика»

Горячкина Юлия Викторовна - директор по направлению «Кадры для цифровой экономики» АНО «Цифровая экономика»

От ФГАНУ «Социоцентр»

Келлер Андрей Владимирович	-	и.о. директора ФГАНУ «Социоцентр»

От АНО ВО «Университет Иннополис»

Бариев Искандер Ильгизарович	-	первый проректор – заместитель директора АНО ВО «Университет Иннополис»

Приняло участие 7 членов Президиума из 8, кворум имеется.

О согласовании дополнительных профессиональных программ (программ профессиональной переподготовки) ИТ-профиля, реализуемых на «цифровых кафедрах» в рамках федерального проекта «Развитие кадрового потенциала ИТ-отрасли» национальной программы «Цифровая экономика Российской Федерации», прошедших оценку экспертов АНО «Цифровая экономика» и получивших статус «соответствует» на первом этапе экспертизы
(Т.Н. Трубникова, А.Ю. Казанцева, В.В. Гришкин, К.А. Богоносов, Ю.В. Горячкина, А.В. Келлер, И.И. Бариев)

Согласовать перечень дополнительных профессиональных программ (программ профессиональной переподготовки) ИТ-профиля, для реализации на «цифровых кафедрах» в рамках федерального проекта «Развитие кадрового потенциала ИТ-отрасли» национальной программы «Цифровая экономика Российской Федерации», согласно приложению.

Голосовали:
«ЗА» – 7 голосов,
«ПРОТИВ» – 0 голосов,
«Воздержался» – 0 голосов.
Решение принято.

Директор Департамента развития цифровых компетенций и образования — Т.Н. Трубникова

Директор по направлению «Кадры для цифровой экономики» АНО «Цифровая экономика», секретарь Президиума — Ю.В. Горячкина

27	114	DevOps-инженер	ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ"
28	980	Цифровизация процессов коммерциализации и инноваций	ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"
29	979	Цифровая трансформация нефтегазового комплекса	ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"
30	975	Экономика и управление на предприятиях нефтегазодобывающей и горнодобывающей промышленности в условиях цифровизации	ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"
31	915	Цифровая гуманитаристика	ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "КАЗАНСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ"
32	914	Цифровая аналитика и принятие решений на основе данных	ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "КАЗАНСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ"
33	910	Digital дизайн	ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "КАЗАНСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ"
34	867	Разработка программных роботов	ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "УФИМСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"
35	1046	Технологии искусственного интеллекта и анализа данных	ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ ИМЕНИ В.И. ВЕРНАДСКОГО"
36	1010	Дизайн СМИ и брендинг	ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)"