Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»

Номер зачетной книжки _____
Преддипломная практика зачтена с оценкой
_____ (_____)
(цифрой) (прописью)
_____
(подпись руководителя практики от БГУИР)
_____._____.2024

ОТЧЕТ
по преддипломной практике
Место прохождения практики: ООО «СТЭКЛЭВЭЛ ГРУПП»
Сроки прохождения практики: с 25.03.2024 по 21.04.2024

Руководитель практики от предприятия:
_____ К.С. Улезко
(подпись руководителя)
М.П.

Студент группы 013801
_____ К.Г. Хоменок
(подпись студента)
Руководитель практики от БГУИР
Шнейдеров Е.Н. − канд.техн.наук, доцент

Минск 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современная технологическая эпоха требует эффективных и гибких инструментов для развертывания, управления и масштабирования веб-сервисов. В контексте этой потребности возникает актуальность исследования и разработки DevOps технологий для поддержки распределенных веб-сервисов на платформе AWS с использованием Terraform.

С увеличением объемов данных, скорости разработки и требований к безопасности, становится необходимым создание интегрированных и автоматизированных процессов управления инфраструктурой. DevOps методология предлагает подход, направленный на сближение разработки и эксплуатации, что позволяет сократить время развертывания проектов и улучшить их качество.

Целью данного дипломного проекта является разработка системы поддержки распределенных веб-сервисов на платформе AWS с применением DevOps технологий и инструментов автоматизации, основанных на Terraform.

Для достижения поставленной цели необходимо решить следующие задачи:

– изучение особенностей распределенных веб-сервисов для AWS;

– анализ требований к функциональности системы и выбор соответствующих технологий;

– разработка и реализация инфраструктурной части системы с использованием Terraform;

– интеграция DevOps практик для оптимизации процессов развертывания и масштабирования веб-сервисов;

– тестирование и оптимизация разработанной системы.

Объектом исследования дипломного проекта являются DevOps технологии и методы поддержки распределенных веб-сервисов. Предметом исследования является разработка программного обеспечения для автоматизации процессов управления инфраструктурой на платформе AWS с использованием Terraform.

Данный дипломный проект также призван внести вклад в развитие современных информационных технологий, обеспечивая устойчивость и надежность работы распределенных веб-сервисов. Разработка и применение DevOps технологий на платформе AWS с использованием Terraform позволит оптимизировать процессы управления инфраструктурой, обеспечивая более быстрое развертывание и масштабирование проектов.

1 ПЛАН-ПРОСПЕКТ ДИПЛОМНОГО ПРОЕКТА

Содержание плана-проспекта дипломного проекта представлено в таблице 1.1.

Таблица 1.1 – План-проспект дипломного проекта

Этап	Описание	Сроки выполнения
Введение	Обоснование актуальности темы дипломного проекта; формулировка цели и задач дипломного проектирования; выделение объекта и предмета исследования дипломного проектирования.	26.03.2024 – 27.03.2024
Анализ исходных данных и постановка задач на дипломное проектирование	Изучение особенностей распределенных веб-сервисов для AWS; определение функциональных требований к системе; анализ DevOps технологий и инструментов автоматизации; выбор Terraform в качестве инструмента для управления инфраструктурой как кодом.	27.03.2024 – 31.03.2024
Описание и проектирование облачной инфраструктуры	Описание и проектирование облачной инфраструктуры: Terraform «инфраструктура как код», описание и обоснование используемых распределенных веб-сервисов, контейнеризация и оркестрация с помощью Docker и Docker Compose, описание и обоснование использования CI/CD, проектирование и разработка облачной инфраструктуры.	01.04.2024 – 08.04.2024
Практическая реализация облачной инфраструктуры для веб-сервиса	Практическая реализация облачной инфраструктуры: обзор разворачиваемого веб-сервиса и используемых библиотек, реализация инфраструктуры в виде кода для облачного окружения.	09.04.2024 – 22.04.2024

Продолжение таблицы 1.1

Этап	Описание	Сроки выполнения
Разработка графического материала	Выполнение чертежа «IDEF0 диаграмма декомпозиции»; выполнение плаката «Диаграмма развертывания веб-сервиса»; выполнение плаката «Структура манифеста Terraform», выполнение плаката «UML диаграмма вариантов использования», выполнение плаката «Графический интерфейс веб-сервиса», выполнение чертежа «Схема алгоритма развертывания инфраструктуры»	16.04.2024 – 14.05.2024
Оценка количественных показателей функционирования разворачиваемого веб-сервиса	Оценка временных показателей; оценка ресурсных показателей.	22.04.2024 – 10.05.2024
Экономическое обоснование применения DevOps-технологий поддержки распределенных Web-сервисов	Выполнение задания по экономическому обоснованию применения DevOps-технологий поддержки распределенных Web-сервисов для AWS с использованием Terraform.	22.04.2024 – 04.05.2024
Оформление отчёта дипломного проекта	Составление введения, заключения, списка используемой литературы, приложений, ведомости дипломного проекта.	02.05.2024 – 20.05.2024

Таким образом, дата окончания работы над дипломным проектом должна быть не позднее 27.05.2024.

2 АНАЛИЗ ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧ НА ДИПЛОМНОЕ ПРОЕКТИРОВАНИЕ

2.1 Анализ исходных данных к дипломному проекту

Исходные данные проекта «DevOps технологии поддержки распределенных Web сервисов для AWS с использованием Terraform» представляют собой четкий набор требований и ограничений, определенных приказом университета. Анализ этих данных необходим для формирования стратегии проектирования, разработки и внедрения системы.

Один из пунктов описывает требования к программному окружению, необходимому для реализации проекта. Важно отметить, что Amazon Web Services (AWS) является основным провайдером облачных услуг. Для сетевых функций и хранения контейнеризированных сервисов используются соответственно AWS VPC, ECR и ECS. Использование Terraform как инструмента для определения инфраструктуры как кода обеспечивает автоматизацию и управление инфраструктурой. Для разработки веб-приложения используются Next.js и React, а для управления состоянием веб-приложения – Drizzle. Для обеспечения безопасности и аутентификации некоторых сервисов, таких как CloudWatch, будет использоваться IAM Policy и IAM Role от AWS. Важно отметить, что все используемые библиотеки должны иметь необязывающую (некоммерческую) лицензию в соответствии с требованиями университета.

Помимо этого, исходные данные к дипломному проекту также определяют функциональные требования к системе. Важно обеспечить функциональность регистрации и аутентификации пользователей с использованием различных методов, таких как почтовый ящик и пароль, учетная запись Google и учетная запись Discord. Требуется реализация управления настройками безопасности личного кабинета пользователей, включая управление активными сессиями и устройствами. Кроме того, необходимо предоставить пользовательский интерфейс для редактирования профиля и поиска доступных онлайн-досок для рисования, а также функционал создания и управления организациями. Инфраструктурная часть должна обеспечивать масштабируемость и надежность системы, а использование CI/CD – автоматизацию процесса развертывания и обновления проекта веб-приложения.

DevOps технологии поддержки распределенных веб-сервисов на базе AWS представляют собой стандартные методы и инструменты, используемые для достижения заданных целей в развертывании и управлении

программными средствами. Эта тема требует разработки подробной инструкции по развертыванию конкретного веб-приложения на AWS с учетом его программного окружения и подключаемых библиотек. Для этого планируется использование широкого спектра распределенных веб-сервисов, таких как VPC, Route Table, Internet Gateway, Public Subnet, Security Group, EC2, ECR, ECS, Elastic IP, IAM Policy, IAM Role и CloudWatch.

В процессе выполнения проекта также будут задействованы облачные виртуальные серверы (AWS EC2), облачные сетевые функции, Terraform в качестве инструмента для описания инфраструктуры как кода, Docker и Docker Compose для контейнеризации и оркестрации, а также CI/CD для автоматизации процесса разработки и развертывания.

Анализ исходных данных позволяет увидеть, что выбранный набор технологий и инструментов позволит обеспечить высокую степень автоматизации, масштабируемость и надежность системы. Использование облачных сервисов AWS обеспечивает гибкость и эффективность в управлении ресурсами, а DevOps методология способствует интеграции разработки и операций для достижения более быстрой и надежной поставки веб-сервиса.

Таким образом, проект будет направлен на создание современной и масштабируемой инфраструктуры для поддержки распределенных веб-сервисов на платформе AWS с использованием передовых DevOps практик, инструментов и технологий.

2.2 Обзор существующих облачных провайдеров по теме дипломного проекта

В современном мире облачные технологии становятся все более востребованными для разработки и развертывания различных типов программных средств. Существует множество облачных провайдеров, предлагающих разнообразные услуги и инструменты для работы с облачными ресурсами. В данном обзоре мы сосредоточимся на основных облачных провайдерах, которые пользуются широкой популярностью во всем мире.

Рассмотрение основных облачных провайдеров позволит получить обзор их ключевых характеристик, услуг и инструментов, что поможет принять решение при выборе провайдера для своих проектов.

Среди основных облачных провайдеров выделяются Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), Alibaba Cloud, Yandex Cloud, IBM Cloud и Oracle Cloud (рисунок 2.1). Каждый из этих провайдеров предоставляет уникальные возможности и сервисы, обеспечивая гибкость, масштабируемость и надежность для различных типов проектов.

Рисунок 2.1 – Популярные облачные провайдеры [1]

В топ 5 облачных провайдеров входит AWS, Microsoft Azure, Google Cloud Platform (GCP), Alibaba и IBM Cloud [2].

Далее необходимо рассмотреть каждого облачного провайдера и их преимущества в контексте дипломного проекта, ориентированного на поддержку распределенных веб-сервисов с использованием DevOps технологий.

2.2.1 Amazon Web Services как облачный провайдер

Сперва необходимо рассмотреть гиганта, который занимает первое место в области облачных провайдеров, – AWS.

Amazon Web Services (AWS) – лидер в области облачных вычислений и входит в топ-5 облачных провайдеров мира, занимая заслуженное первое место. AWS предоставляет обширный набор облачных услуг и инструментов, в том числе тех, которые существенно способствуют применению методологии DevOps (рисунок 2.2).

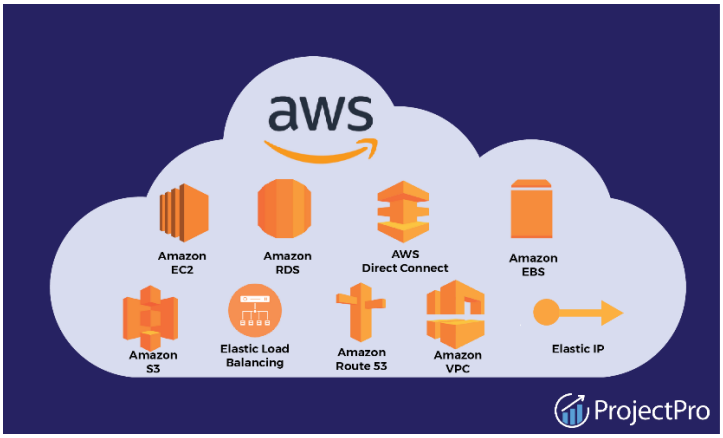

Рисунок 2.2 – Облачный провайдер AWS [3]

1 Вычислительные ресурсы: AWS предоставляет широкий спектр виртуальных машин (Amazon EC2) с различными типами экземпляров, а также управляемые контейнерные службы, такие как Amazon Elastic Container Service (ECS) и Amazon Elastic Kubernetes Service (EKS).

2 Хранилище данных: AWS предлагает различные сервисы хранения данных, такие как Amazon S3 для хранения неструктурированных данных, Amazon RDS для реляционных баз данных, Amazon DynamoDB для NoSQL баз данных и Amazon Redshift для аналитических данных.

3 Сетевые ресурсы: AWS предоставляет инструменты для создания и управления виртуальными сетями, балансировки нагрузки, контентной доставки, а также виртуальных частных сетей (Amazon VPC) для изоляции ресурсов.

4 Управление ресурсами и мониторинг: AWS предоставляет инструменты для автоматизации развертывания и управления инфраструктурой, такие как AWS CloudFormation, а также сервисы мониторинга и журналирования, такие как Amazon CloudWatch и AWS CloudTrail.

5 Инструменты разработки: AWS предлагает набор инструментов для разработчиков, включая AWS CodeCommit (сервис хранения кода), AWS CodeBuild (сервис для сборки кода), AWS CodeDeploy (сервис для автоматизации развертывания) и AWS CodePipeline (сервис для создания непрерывной интеграции и доставки).

6 Интеграция с открытыми источниками: AWS поддерживает множество открытых технологий и инструментов, таких как Docker, Kubernetes, Jenkins, Git и другие, что делает его привлекательным выбором для компаний, работающих в среде DevOps.

7 Безопасность: AWS обеспечивает широкий спектр инструментов и служб для обеспечения безопасности программных средств и данных, включая механизмы аутентификации, управления доступом, шифрования данных и т. д.

Это лишь небольшой обзор возможностей AWS в контексте DevOps. Платформа постоянно расширяется и обновляется, предлагая новые инструменты и сервисы для улучшения процессов разработки, развертывания и управления различными типами программных средств и веб-приложений.

Помимо этого, AWS предоставляет широкий спектр других сервисов, которые могут быть полезны для разработки и поддержки различных типов программных средств в контексте DevOps. Все это в сочетании с высокой производительностью, масштабируемостью и безопасностью делают AWS популярным и уважаемым выбором среди разработчиков и инженеров по

всему миру. С их помощью компании могут реализовывать DevOps практики с высокой степенью эффективности и надежности.

В целом, Amazon Web Services (AWS) предоставляет разработчикам широкий спектр инструментов и услуг для создания, развертывания и управления программных средств в облаке с использованием DevOps практик. Это делает AWS одним из наиболее популярных и предпочтительных облачных провайдеров для различных типов проектов, включая дипломные проекты, ориентированные на разработку и поддержку распределенных веб-сервисов. Возможность использования инфраструктуры как кода, широкий набор сервисов DevOps и высокие стандарты безопасности делают AWS привлекательным выбором для разработчиков, стремящихся к эффективной и надежной реализации своих проектов в облаке.

2.2.2 Microsoft Azure как облачный провайдер

Заголовочная «About» страница, которая рассказывает о том, что такое Microsoft Azure объясняет, – «Облачная платформа Azure включает более 200 продуктов и облачных служб, которые помогут в создании новых решений для сегодняшних и будущих задач. Создавайте и запускайте приложения и управляйте ими в нескольких облаках, локально и в пограничной среде, используя удобные для вас инструменты и платформы» (рисунок 2.3) [4].

Рисунок 2.3 – Облачный провайдер Microsoft Azure

Некоторые из предоставляемых услуг облачного провайдера Microsoft Azure:

1 Вычислительные ресурсы: Azure предлагает виртуальные машины (VM) различных конфигураций, контейнерные службы (Azure Kubernetes Service), а также высокопроизводительные вычисления для обработки данных (Azure Batch).

2 Хранилище данных: Azure предоставляет различные типы хранилищ данных, такие как Azure Blob Storage (для хранения неструктурированных данных), Azure SQL Database (для реляционных баз данных), Azure Cosmos DB (для глобально распределенных баз данных), а также множество других.

3 Сетевые ресурсы: Azure предлагает широкий спектр сетевых решений, включая виртуальные сети (Virtual Network), службы балансировки нагрузки, контентную доставку, VPN и многое другое.

4 Управление ресурсами и мониторинг: С помощью Azure DevOps и Azure Monitor вы можете автоматизировать развертывание и управление веб-сервисами, а также отслеживать их производительность и доступность.

5 Инструменты разработки: Azure предлагает широкий спектр инструментов для разработчиков, включая Azure DevOps Services (для управления жизненным циклом разработки ПО), Azure DevTest Labs (для создания тестовых сред), Azure Repos (для хранения кода), Azure Artifacts (для управления пакетами) и многое другое.

6 Интеграция с открытыми источниками: Azure поддерживает множество открытых технологий и инструментов, таких как Docker, Kubernetes, Jenkins, Git и другие, что делает его привлекательным выбором для компаний, работающих в среде DevOps.

7 Безопасность: Azure обеспечивает широкий спектр инструментов для обеспечения безопасности ваших веб-сервисов и данных, включая механизмы аутентификации, управление доступом, мониторинг безопасности и т. д.

Это лишь небольшой обзор возможностей Azure в контексте DevOps. Платформа постоянно обновляется и расширяется, предлагая новые инструменты и услуги для облегчения процесса разработки, развертывания и управления веб-сервисами. Статистика облачного провайдера сообщает о том, что было инвестировано в безопасность более 1 млрд. долларов США (USD) за последний год (2023) для защиты клиентов от киберугроз.

2.2.3 Google Cloud Platform как облачный провайдер

Google Cloud Platform (GCP) представляет собой мощный облачный провайдер, который предоставляет широкий спектр услуг и инструментов для компаний, занимающихся DevOps. Одной из ключевых возможностей GCP является его гибкая и масштабируемая вычислительная инфраструктура. С помощью виртуальных машин и управляемых контейнерных сред, таких как Kubernetes Engine, разработчики могут эффективно управлять вычислительными ресурсами и развертывать программные средства в облаке.

Кроме того, GCP предлагает обширный набор сервисов для хранения данных. От Cloud Storage для объектов до Cloud SQL для реляционных баз данных и Firestore для NoSQL хранилищ, разработчики имеют доступ к разнообразным решениям для хранения и обработки данных.

Важным аспектом в сфере DevOps является эффективное управление и мониторинг ресурсов. GCP предоставляет инструменты автоматизации

процессов развертывания и управления инфраструктурой, такие как Cloud Build и Stackdriver, которые позволяют разработчикам легко контролировать производительность и доступность своих проектов.

Инструменты разработки на GCP, такие как Cloud Source Repositories и интеграция с Jenkins, обеспечивают командам DevOps средства для эффективной совместной работы над кодом и автоматизации процессов разработки и развертывания.

Кроме того, GCP активно поддерживает открытые технологии и инструменты, что обеспечивает гибкость в интеграции с существующими DevOps процессами и инфраструктурой.

Безопасность также является приоритетом для GCP, и платформа предоставляет обширные инструменты для обеспечения безопасности данных и управления доступом к ресурсам, включая механизмы аутентификации, управления доступом и шифрование данных.

В целом, GCP предоставляет компаниям в области DevOps все необходимые инструменты и сервисы для эффективного разработки, развертывания и управления проектов в облаке.

2.2.4 Alibaba Cloud как облачный провайдер

Alibaba Cloud, как облачный провайдер, играет ключевую роль в сфере DevOps, предоставляя комплексные решения для ускорения процессов разработки и обеспечения непрерывной доставки программных средств. С помощью своих сервисов и инструментов, Alibaba Cloud способствует созданию эффективной и безопасной среды для DevOps, что позволяет компаниям быстро адаптироваться к изменениям и оптимизировать рабочие процессы.

В основе подхода Alibaba Cloud к DevOps лежит решение проблем эффективности и безопасности, связанных с непрерывным развертыванием облачных приложений. Сервисы Alibaba Cloud обеспечивают гибкую разработку разработанных решений, мониторинг всей системы и усиленные меры безопасности в мультиоблачных средах. Это достигается за счет использования таких функций, как высокоскоростная репликация образов, подписывание образов с помощью Key Management Service (KMS) и автоматическое развертывание разработанных программных средств при обновлении образов.

Alibaba Cloud предлагает уникальные возможности для многокластерного развертывания, позволяя использовать ArgoCD для развертывания разработанных решений в нескольких кластерах. Это включает в себя возможность использования YAML или HELM для развертывания,

просмотра топологии развертывания и мониторинга состояния проектов с помощью Application Center. Такой подход способствует оптимизации и ускорению процессов разработки и развертывания программных средств.

Для улучшения процесса доставки разработанных проектов Alibaba Cloud использует цепочку доставки, которая включает в себя функции, такие как кросс-региональная репликация образов, аутентификация образов на основе подписей и автоматическое сканирование безопасности образов. Это помогает обеспечить стабильную и безопасную доставку бизнес-приложений.

Alibaba Cloud Service Mesh, совместимый с Istio, поддерживает кластеры на краю сети и зарегистрированные кластеры. Он предоставляет возможности управления трафиком между сервисами, включая управление трафиком между кластерами в разных регионах и отказоустойчивость. Service Mesh позволяет маршрутизировать и разделять трафик между сервисами, обеспечивать безопасность коммуникации между сервисами и наблюдать за поведением сервисов в сетях.

Поддержка OpenTelemetry в Alibaba Cloud позволяет автоматически собирать логи без необходимости программирования. Это дает возможность просматривать топологию вызовов микросервисов, связывать данные трассировки с логами и настраивать панели мониторинга для быстрого выявления корневых причин и анализа узких мест производительности распределенных веб-сервисов.

Microservices Engine (MSE) от Alibaba Cloud является однооконной платформой для микросервисов, предоставляющей полностью управляемые центры регистрации и конфигурации, шлюз API и возможности управления микросервисами. MSE позволяет легко построить собственную систему микросервисов, обеспечивая гибкость и удобство в управлении микросервисами.

В заключение, Alibaba Cloud предоставляет мощные инструменты и сервисы для DevOps, которые помогают компаниям достигать высокого качества и эффективности в доставке бизнес-приложений. С помощью продвинутых технологий и интеграции с другими продуктами Alibaba Cloud, DevOps-команды могут реализовывать сложные проекты с уверенностью в стабильности и безопасности своих решений [5][6][7].

2.2.5 Yandex Cloud как облачный провайдер

YandexCloud, как облачный провайдер, представляет собой мощную платформу для DevOps-специалистов, предлагая широкий спектр услуг и инструментов для автоматизации и оптимизации процессов разработки, тестирования и развертывания различных разработанных решений.

Основываясь на принципах непрерывной интеграции и доставки (CI/CD), YandexCloud обеспечивает эффективное управление инфраструктурой и ресурсами, что является ключевым для достижения гибкости и скорости в современной разработке программного обеспечения.

В рамках DevOps-подхода YandexCloud предлагает интегрированные решения для мониторинга и логирования, что позволяет командам быстро реагировать на изменения в системе и поддерживать высокий уровень производительности сервисов. Инженеры YandexCloud занимаются не только релизами и мониторингом, но и тестированием, архитектурой, а также написанием кода для доработки или исправления возникающих проблем [8].

YandexCloud активно использует практики SRE (Site Reliability Engineering), что позволяет повысить надежность и стабильность сервисов. SRE-инженеры YandexCloud работают над созданием долгосрочных решений, влияющих на развитие продукта и обеспечивающих его работоспособность, надежность и производительность.

Сервисы YandexCloud включают в себя управление доступами и ресурсами, что является важной частью безопасности и контроля в облачной среде. Это позволяет DevOps-командам настраивать права доступа к ресурсам и сервисам в соответствии с требованиями безопасности и политиками компании [9].

Документация YandexCloud предоставляет подробную информацию о доступных сервисах, их возможностях и способах активации. Это обеспечивает прозрачность и понимание инструментов, доступных для DevOps-команд, и способствует эффективному использованию облачной платформы [10].

YandexCloud поддерживает различные технологии контейнеризации и оркестрации, такие как Kubernetes, что позволяет автоматизировать развертывание и масштабирование решений. Это обеспечивает гибкость и удобство в управлении микросервисами и их взаимодействием.

Интеграция с современными инструментами разработки и системами контроля версий, такими как Git, упрощает процесс разработки и сотрудничества в командах, а также способствует непрерывному улучшению и обновлению продуктов.

YandexCloud предоставляет возможности для анализа данных и машинного обучения, что позволяет DevOps-командам использовать передовые алгоритмы для оптимизации процессов, предсказания проблем и автоматизации рутинных задач.

В заключение, YandexCloud, как облачный провайдер в сфере DevOps, предлагает комплексный набор инструментов и сервисов, которые помогают командам достигать высокой эффективности и качества в разработке и

эксплуатации программного обеспечения, обеспечивая при этом безопасность и стабильность облачной среды. Это делает YandexCloud надежным партнером для компаний, стремящихся к инновациям и постоянному технологическому прогрессу.

2.2.6 IBM Cloud как облачный провайдер

IBMCloud представляет собой интегрированную платформу, предназначенную для удовлетворения потребностей DevOps-команд в быстрой и надежной разработке, тестировании и развертывании различных разработанных решений. С помощью обширного набора инструментов и сервисов IBMCloud обеспечивает эффективное управление кодом, автоматизацию процессов и непрерывную доставку, что является неотъемлемой частью современных DevSecOps практик.

В основе DevOps-подхода IBMCloud лежит создание культуры непрерывного обучения и экспериментирования, что позволяет командам быстро реагировать на обратную связь и данные, а также часто выпускать новые версии продуктов. Это способствует ускорению процесса разработки и повышению качества программного обеспечения [11].

IBMCloud предлагает DevSecOps инструменты, которые помогают командам достигать согласованности среды от тестирования до производства и во всем гибридном облаке. Это включает в себя использование облачных схематик для управления ресурсами облака и инструментов непрерывной доставки, основанных на Tekton, для автоматизации процессов сборки, тестирования и развертывания.

IBMCloud поддерживает DevOps-культуру, которая объединяет бизнес, разработку и операционные команды, способствуя улучшению качества веб-приложений, безопасности и стабильности через частые релизы. Это также помогает снизить затраты за счет повышения эффективности и уменьшения простоев.

С помощью IBM Cloud Continuous Delivery, команды могут управлять исходным кодом с помощью репозиториев, основанных на GitLab Community Edition, и отслеживать работу и проблемы. Это обеспечивает централизованное управление функциями и конфигурацией проектов, а также контроль и управление ключами шифрования.

IBMCloud предоставляет возможности для запуска контейнеризированных рабочих нагрузок с использованием IBM Cloud Kubernetes Service, что обеспечивает безопасность, инновации с открытым исходным кодом и возможности корпоративного уровня. Кроме того, IBM Cloud Satellite позволяет развертывать и запускать проекты последовательно в

локальных, периферийных вычислительных и публичных облачных средах от любого облачного провайдера.

IBMCloud также предлагает инструменты для мониторинга и устранения неполадок инфраструктуры, облачных сервисов и развернутых на нем проектов, что позволяет DevOps-командам получать полную картину состояния системы и быстро реагировать на возникающие проблемы.

В заключение, IBMCloud является мощным инструментом для DevOps-команд, предоставляя все необходимое для непрерывной интеграции и доставки, управления кодом и ресурсами, а также для обеспечения безопасности и стабильности проектов в облачной среде. Это делает IBMCloud важным ресурсом для компаний, стремящихся к инновациям и постоянному технологическому прогрессу в разработке программного обеспечения.

2.2.7 Oracle Cloud как облачный провайдер

Oracle Cloud представляет собой облачную платформу, которая предлагает решения для DevOps, обеспечивая непрерывную интеграцию и непрерывную доставку (CI/CD) для команд разработчиков, работающих на Oracle Cloud Infrastructure (OCI). Это интегрированная служба, которая обеспечивает согласованность идентификации, безопасности, логирования и других аспектов инфраструктуры Oracle Cloud, а также предварительно настроенные безопасные развертывания в OCI Compute Services.

OCI DevOps гибко интегрируется с существующими рабочими процессами и инструментами, такими как GitHub, GitLab, Jenkins и другие, включая частные и облачные ресурсы. Это позволяет командам разработчиков сосредоточиться на коде и рабочих процессах, не заботясь о серверах, поскольку платформа масштабируется для поддержки параллельных сборок.

Служба Oracle DevOps предоставляет документацию и примеры, архитектуры ссылок и демонстрации, которые помогают разработчикам начать работу и оптимизировать свои процессы CI/CD. Кроме того, Oracle предлагает бесплатную пробную версию своих облачных услуг, что позволяет разработчикам ознакомиться с возможностями OCI DevOps без временных ограничений на выбор услуг.

В контексте DevOps, Oracle Cloud поддерживает различные стратегии развертывания, такие как канареечное развертывание и стратегии blue-green, позволяя администраторам выбирать между различными подходами в зависимости от риска развертывания нового релиза, влияния нового релиза на пользователей и инфраструктурных затрат, необходимых для реализации стратегии.

Oracle Cloud также предлагает инструменты для автоматизации процессов разработки программного обеспечения, такие как Jenkins, Terraform и Grafana, которые можно интегрировать с OCI для снижения затрат по сравнению с другими продуктами инфраструктуры. Эти инструменты помогают автоматизировать и оптимизировать процессы разработки и доставки программного обеспечения, охватывая управление Git и репозиториями, отслеживание проблем, CI/CD конвейеры, процессы DevOps, среды разработки, управление инфраструктурой и многое другое.

Кроме того, Oracle предлагает широкий спектр инструментов, включая Cloud Shell, наборы программного обеспечения для разработки (SDK) и интерфейс командной строки (CLI) для Oracle Cloud Infrastructure. Эти инструменты автоматизируют разработку проектов с возможностями для интегрированных сред разработки (IDE), DevOps и Oracle Database.

В заключение, Oracle Cloud является мощным облачным провайдером в сфере DevOps, предлагая комплексные решения для непрерывной интеграции и доставки. Это позволяет командам разработчиков ускорить процесс разработки и доставки программного обеспечения, оптимизировать рабочие процессы и снизить операционные затраты, используя передовые инструменты и сервисы, предоставляемые Oracle Cloud.

2.3 Обоснование и описание выбора облачного провайдера

При разработке и внедрении системы поддержки распределенных веб-сервисов на платформе AWS с использованием DevOps технологий и инструментов автоматизации, выбор облачного провайдера играет критическую роль в обеспечении надежности, масштабируемости и эффективности развертывания. Среди различных вариантов на рынке облачных услуг, Amazon Web Services (AWS) выделяется как оптимальный выбор, основываясь на ряде фундаментальных преимуществ.

Первоначально стоит отметить, что AWS в настоящее время занимает лидирующее положение среди облачных провайдеров. Это подтверждается не только статистическими данными, но и практическим опытом множества компаний, использующих AWS в качестве основного инфраструктурного решения. Такое доминирование обусловлено комплексным подходом к предоставлению облачных услуг, который включает в себя широкий спектр сервисов и высокую степень надежности.

Преимущества AWS в контексте данного проекта очевидны. AWS предоставляет полноценный набор инструментов для развертывания, управления и масштабирования инфраструктуры, необходимой для поддержки распределенных веб-сервисов. Среди ключевых сервисов, которые

обеспечивают успешную реализацию проекта, следует выделить Amazon EC2 для управления виртуальными серверами, Amazon ECS для контейнеризации и оркестрации, а также Terraform для автоматизации процессов развертывания.

Сравнивая AWS с другими ведущими облачными провайдерами, такими как Microsoft Azure, Google Cloud Platform (GCP), Alibaba Cloud, Yandex Cloud, IBM Cloud и Oracle Cloud, можно отметить несколько существенных преимуществ. Во-первых, AWS обладает более широким и взаимосвязанным экосистемным подходом, что облегчает интеграцию различных сервисов и упрощает управление всей инфраструктурой.

Кроме того, AWS обеспечивает более высокую производительность и доступность по сравнению с альтернативными платформами. Это обусловлено глубокими инвестициями в развитие собственной инфраструктуры, включая масштабируемые центры обработки данных и сетевые технологии. Таким образом, AWS способен обеспечить стабильную и быструю работу веб-сервисов даже при высоких нагрузках.

Однако, не только технические аспекты делают AWS предпочтительным выбором для данного проекта. Важным преимуществом является также низкий порог вхождения и интуитивно понятный интерфейс. Это позволяет команде быстро освоить инструменты AWS и эффективно использовать их в рамках проекта, сокращая время на подготовку и обучение.

Более того, учитывая, что знания и опыт по AWS широко распространены среди IT-специалистов, выбор данного провайдера также обеспечивает доступ к большому сообществу экспертов и ресурсам обучения. Это упрощает процесс поддержки и развития проекта, а также обеспечивает доступ к лучшим практикам и решениям.

Таким образом, учитывая все вышеперечисленные факторы, выбор Amazon Web Services (AWS) в качестве облачного провайдера для проекта по разработке и поддержке распределенных веб-сервисов на платформе AWS с использованием DevOps технологий и инструментов автоматизации является логичным и обоснованным решением. Все аспекты, начиная от широкого набора сервисов и высокой производительности, и заканчивая доступностью знаний и ресурсов поддержки, делают AWS оптимальным выбором для успешной реализации проекта.

2.4. Постановка задач на дипломное проектирование

Для дипломного проектирования по теме «DevOps технологии поддержки распределенных Web сервисов для AWS с использованием Terraform» требуется определить функциональные требования к веб-

приложению, а также требования к выполнению инфраструктурной части и тематики дипломного проектирования.

Исходя из анализа исходных данных и существующих программных средств, мы можем сформировать следующие функциональные требования:

– регистрация и авторизация пользователей (почтовый ящик и пароль, учетная запись Google, учетная запись Discord);

– управление настройками безопасности личного кабинета (изменение/добавление почтового ящика, управление активными сессиями/устройствами, подключение учетных записей Google/Discord);

– редактирование профиля пользователя;

– поиск доступных онлайн-досок организации для рисования;

– создание организации;

– приглашение в организацию по электронной почте;

– маркировка онлайн-досок для рисования как избранных;

– функционал онлайн-доски для рисования: выделение областей, нанесение текста, создание стикеров с текстом, создание фигур (квадрат, круг);

– редактирования названия доски,

– отображение списка пользователей в комнате онлайн-доски.

К инфраструктурным требованиям и задачам (тематика дипломного проектирования) относятся:

1 Использование облачных виртуальных серверов (AWS EC2): реализация системы требует создания и управления виртуальными серверами в облаке AWS с использованием EC2 для обеспечения вычислительных ресурсов.

2 Использование облачных сетевых функций: необходимо использовать средства AWS для настройки сетевой инфраструктуры, включая VPC, Route Table, Internet Gateway, Public Subnet и Security Group, для обеспечения безопасного и эффективного обмена данными между сервисами.

3 Использование Terraform как «инфраструктура как код»: для автоматизации процесса развертывания инфраструктуры и управления ресурсами AWS следует применить Terraform, что позволит описывать инфраструктуру в виде кода и поддерживать ее версионность.

4 Использование Docker и системы оркестрации Docker Compose: для упаковки и изоляции приложений следует использовать контейнеризацию с помощью Docker, а также для управления множеством контейнеров и их средой выполнения – Docker Compose.

5 Использование облачных технологий контейнеризации: в качестве платформы для управления контейнерами и оркестрации их работы на

инфраструктуре AWS предлагается использовать AWS Elastic Container Service (ECS) или альтернативные облачные сервисы.

6 Использование CI/CD: для автоматизации процессов непрерывной интеграции и доставки следует использовать соответствующие инструменты, такие как AWS CodePipeline или альтернативные решения.

Таким образом, требуется знание и использование основных сервисов AWS, таких как VPC, Route Table, Internet Gateway, Public Subnet, Security Group, EC2, ECR, ECS, Elastic IP, IAM Policy, IAM Role, CloudWatch; навыки работы с Terraform для описания и автоматизации построения инфраструктуры в виде кода для управления ресурсами AWS; использование Docker и системы оркестрации Docker Compose.

3 ОПИСАНИЕ И ПРОЕКТИРОВАНИЕ ОБЛАЧНОЙ ИНФРАСТРУКТУРЫ

3.1 Terraform «инфраструктура как код»

HashiCorp Terraform – это инструмент инфраструктуры как кода, позволяющий определять облачные и локальные ресурсы в человекочитаемых конфигурационных файлах, которые можно версировать, повторно использовать и совместно использовать. Затем можно использовать последовательный рабочий процесс для обеспечения и управления всей вашей инфраструктурой на протяжении всего ее жизненного цикла. Terraform может управлять низкоуровневыми компонентами, такими как вычислительные ресурсы, ресурсы хранения и сетевые ресурсы, а также высокоуровневыми компонентами, такими как записи DNS и функции SaaS.

Terraform создает и управляет ресурсами на облачных платформах и других сервисах через их интерфейсы прикладного программирования (API). Провайдеры позволяют Terraform работать практически с любой платформой или сервисом с доступным API.

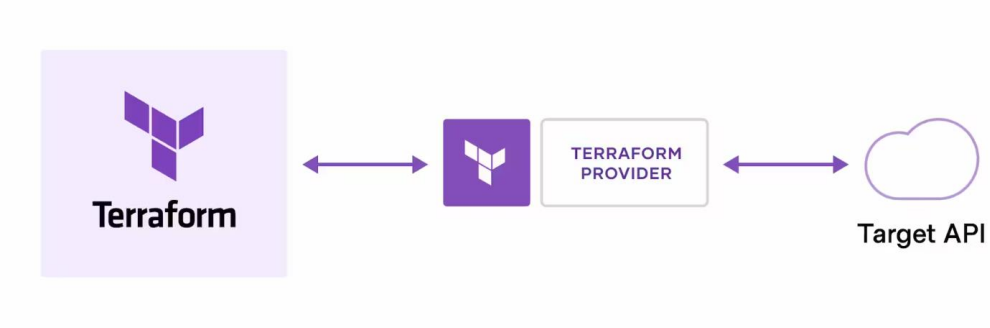

Рисунок 3.1 – Принцип работы Terraform [12]

«Под капотом» работы Terraform или принципа инфраструктуры как код (IaC) лежит процесс, который начинается с загрузки конфигурационных файлов Terraform, описывающих желаемую инфраструктуру, включая ресурсы и их параметры.

1 Анализ конфигурации: Terraform анализирует конфигурационные файлы и создает внутреннее представление желаемой инфраструктуры в виде графа зависимостей между ресурсами. Этот граф описывает порядок, в котором ресурсы должны быть созданы и связаны друг с другом.

2 Инициализация провайдера: Terraform обращается к указанным провайдерам, используя предоставленные в конфигурации ключи доступа и

настройки. Провайдеры – это компоненты, которые взаимодействуют с API конкретного облачного провайдера или другой системы для управления ресурсами.

3 Планирование изменений: Terraform сравнивает текущее состояние инфраструктуры (которое хранится локально в файлах состояния) с желаемым состоянием, описанным в конфигурации. Он определяет, какие ресурсы должны быть добавлены, изменены или удалены, чтобы достичь желаемого состояния, и генерирует план изменений.

4 Применение изменений: после того как план изменений сгенерирован и подтвержден пользователем, Terraform начинает внесение изменений в инфраструктуру, взаимодействуя с API провайдера для создания, изменения или удаления ресурсов. Terraform следит за порядком и зависимостями ресурсов, чтобы убедиться, что изменения применяются в правильном порядке.

5 Обновление состояния: по мере применения изменений Terraform обновляет локальное состояние инфраструктуры, чтобы отразить актуальное состояние. Это позволяет Terraform отслеживать текущее состояние инфраструктуры и использовать его при следующих операциях управления.
Таким образом, Terraform обеспечивает автоматизированное управление инфраструктурой с использованием концепции инфраструктуры как кода, упрощая процесс развертывания и управления облачными ресурсами.

HashiCorp и сообщество Terraform уже написали тысячи провайдеров для управления различными типами ресурсов и сервисов. В реестре Terraform можно найти все общедоступные провайдеры, включая Amazon Web Services (AWS), Azure, Google Cloud Platform (GCP), Kubernetes, Helm, GitHub, Splunk, DataDog и многие другие.

Основной рабочий процесс Terraform состоит из трех этапов:

1 Написание манифеста и (или) write manifest: определение ресурсов, которые могут находиться у нескольких облачных провайдеров и служб. Например, можно создать конфигурацию для развертывания приложения на виртуальных машинах в сети виртуального частного облака (VPC) с группами безопасности и балансировщиком нагрузки.

2 План и (или) plan: Terraform создает план выполнения, описывающий инфраструктуру, которую он будет создавать, обновлять или уничтожать на основе существующей инфраструктуры и вашей конфигурации.

3 Применить и (или) apply: после утверждения Terraform выполняет предложенные операции в правильном порядке, соблюдая все зависимости от ресурсов. Например, если вы обновите свойства VPC и измените количество виртуальных машин в этом VPC, Terraform создаст VPC заново, прежде чем масштабировать виртуальные машины.

Один из удобных функционалов Terraform, – это то, что после написания инфраструктуры как кода можно выполнить некоторую проверку, написав в терминале terraform plan, который в большинстве случаев покажет семантические ошибки, если таковые имеются, или же выведет подробный план построения инфраструктуры, а также её этапы (рисунок 3.2).

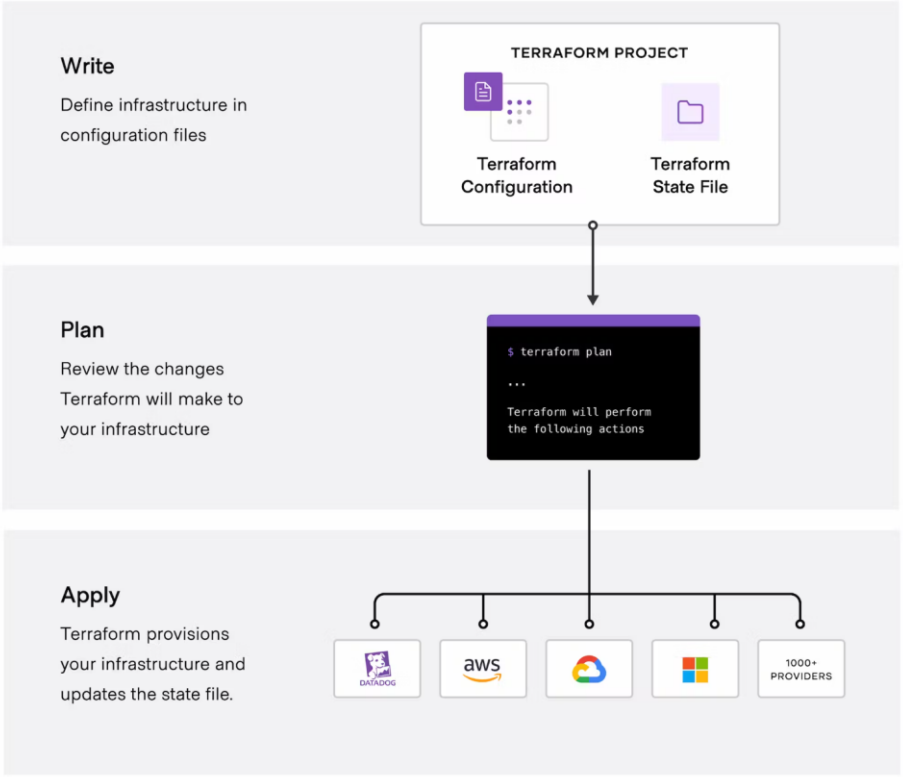

Рисунок 3.2 – Рабочий процесс Terraform [12]

Управление любой инфраструктурой и провайдером. Terraform имеет реестр, в котором можно найти провайдеров для многих платформ и сервисов, которые вы уже используете. Вы также можете написать свои собственные. Terraform использует неизменяемый подход к инфраструктуре, что снижает сложность обновления или модификации ваших сервисов и инфраструктуры.

Отслеживание изменений инфраструктуры. Terraform генерирует план и запрашивает одобрение перед изменением инфраструктуры. Он также отслеживает реальную инфраструктуру в файле состояния, который служит источником истины для вашей среды. Terraform использует файл состояния для определения изменений, которые необходимо внести в инфраструктуру, чтобы она соответствовала вашей конфигурации.

Автоматизация изменения. Файлы конфигурации Terraform (манифесты) являются декларативными, то есть они описывают конечное

состояние инфраструктуры. Отсюда можно сделать вывод, что не нужно писать пошаговые инструкции для создания ресурсов, потому что Terraform управляет основной логикой. Terraform строит граф ресурсов для определения зависимостей между ресурсами и параллельно создает или изменяет независимые ресурсы. Это позволяет Terraform эффективно предоставлять ресурсы.

Стандартизация конфигураций. Terraform поддерживает многократно используемые компоненты конфигурации, называемые модулями, которые определяют настраиваемые коллекции инфраструктуры, экономя время. Таким образом, можно использовать общедоступные модули из реестра Terraform или написать свои собственные.

Совместная работа. Поскольку конфигурация записывается в файл, можно фиксировать манифесты в системе контроля версий (VCS) и использовать Terraform Cloud или другие облачные провайдеры для эффективного управления рабочими процессами Terraform в командах. Terraform Cloud запускает Terraform в последовательной, надежной среде и обеспечивает безопасный доступ к общему состоянию и секретным данным, контроль доступа на основе ролей, частный реестр для обмена модулями и провайдерами и многое другое.

Terraform – это программное обеспечение с открытым исходным кодом, которое позволяет описывать инфраструктуру кодом у специального провайдера, после чего проверять её, масштабировать и изменять.

Таким образом, Terraform, как инструмент (программное обеспечение) написания «инфраструктуры как код», является важным элементом в актуальных технологиях DevOps для того, чтобы эффективно и автоматизировано управлять инфраструктурой разворачиваемого проекта (веб-приложения) в рамках данного дипломного проекта.

3.2 Описание и обоснование используемых распределенных веб-сервисов

Распределенные веб-сервисы играют ключевую роль в построении облачной инфраструктуры, обеспечивая масштабируемость и надежность при предоставлении услуг через интернет.

В данной главе необходимо рассмотреть и обосновать выбор используемых распределенных веб-сервисов, – VPC, Route Table, Internet Gateway, Public Subnet, Security Group, EC2, ECR и ECS, Elastic IP, IAM Policy, IAM Role и CloudWatch.

Таким образом, необходимо перейти к рассмотрению первого распределенного веб-сервиса – Virtual Private Cloud (VPC).

3.2.1 VPC

Amazon Virtual Private Cloud (VPC) – это сервис, который дает возможность запускать ресурсы AWS в определяемой пользователем логически изолированной виртуальной сети. Это позволяет полностью контролировать среду виртуальной сети, в том числе выбирать собственный диапазон IP-адресов, создавать подсети, а также настраивать таблицы маршрутизации и сетевые шлюзы. Для большинства ресурсов в VPC можно использовать и IPv4, и IPv6, и таким образом получить безопасный и удобный доступ к ресурсам и приложениям (рисунок 3.2).

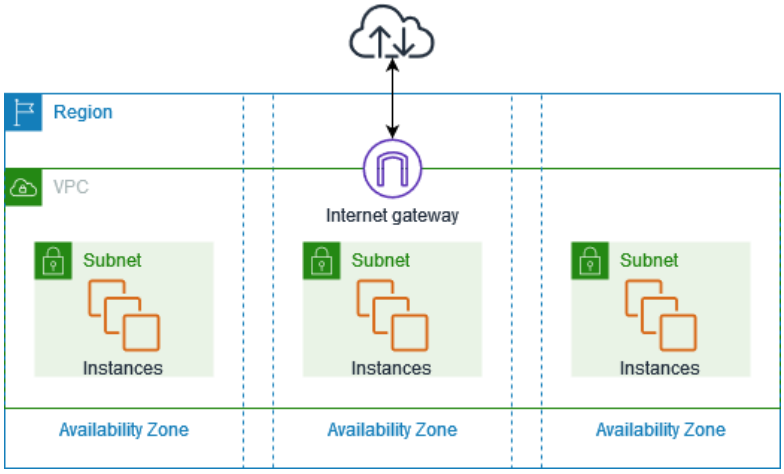

Рисунок 3.3 – Инфраструктура с использованием VPC [13]

Как один из основополагающих сервисов AWS, Amazon VPC упрощает индивидуальную настройку параметров сети VPC. Можно создать общедоступную подсеть для своих веб-серверов с доступом к Интернету. Можно также поместить свои серверные системы, такие как базы данных или серверы приложений, в частную подсеть без доступа к Интернету. Amazon VPC дает возможность использовать многоуровневую систему безопасности, которая состоит из групп безопасности и сетевых списков контроля доступа. Такая система позволяет контролировать доступ к инстансам Эластичного облака вычислений Amazon (Amazon EC2) в каждой подсети.

Иными словами, это сервис, который предоставляет возможность создания виртуальной приватной облачной сети. Данный сервис можно настроить под себя, включая локальный IP-адрес, расположения зоны (availability zone) и мн. др.

Помимо этого, в VPC можно настроить такие сервисы как журналы потоков, IP Address Manager (IPAM), работу с IP-адресами, входящую маршрутизацию, Network Access Analyzer, список контроля доступа к сети,

сетевой менеджер, Reachability Analyzer (инструмент для анализа статических конфигураций), группы безопасности (security groups) и зеркалирование трафика.

Как видно из рисунка 3.3, VPC – это один из главных и основополагающих распределенных веб-сервисов, который представляет собой главную сеть инфраструктуры, выше него находится только регион (доступная зона и (или) availability zone).

Выбор сервиса Amazon Virtual Private Cloud (VPC) обоснован его ключевой ролью в построении безопасной, масштабируемой и надежной облачной инфраструктуры на платформе AWS. VPC позволяет создавать изолированные виртуальные сети с гибкой настройкой параметров, таких как диапазон IP-адресов, подсети и таблицы маршрутизации. Этот сервис обеспечивает контроль доступа к ресурсам и приложениям, что является критическим аспектом в развертывании и управлении распределенными веб-сервисами. Таким образом, включение VPC в рамки проекта обеспечивает основу для построения безопасной и высокопроизводительной облачной инфраструктуры, совместимой с принципами DevOps и поддерживаемой с использованием Terraform.

3.2.2 Route Table

Route Table – это один из сервисов AWS, основанный на AWS VPC, таблица маршрутов содержит набор правил, называемых маршрутами, которые определяют, куда направлять сетевой трафик из подсети или шлюза.

Необходимо перечислить основные понятия для Route Table:

– главная таблица маршрутов (main route table): таблица маршрутов, которая автоматически поставляется с VPC. Она управляет маршрутизацией для всех подсетей, которые явно не связаны с какой-либо другой таблицей маршрутов.

– пользовательская таблица маршрутов (custom route table): таблица маршрутов, которую вы создаете для своего VPC;

– назначение (destination): диапазон IP-адресов, куда должен направляться трафик (CIDR назначения);

– цель (target): шлюз, сетевой интерфейс или соединение, через которое нужно отправить трафик назначения; например, интернет-шлюз;

– ассоциация таблиц маршрутов (route table association): ассоциация между таблицей маршрутов и подсетью, интернет-шлюзом или виртуальным частным шлюзом;

– таблица маршрутов подсети (subnet route table): таблица маршрутов, связанная с подсетью;

– локальный маршрут (local route): маршрут по умолчанию для связи внутри VPC;

– распространение (propagation): если подключен виртуальный частный шлюз к VPC и включено распространение маршрутов, то AWS автоматически добавляет маршруты для VPN-соединения в таблицы маршрутов подсетей;

– таблица маршрутов шлюза (gateway route table): таблица маршрутов, связанная с интернет-шлюзом или виртуальным частным шлюзом;

– ассоциация с границами (edge association): таблица маршрутов, которую вы используете для маршрутизации входящего трафика VPC на устройство, связывание таблицы маршрутов с интернет-шлюзом или виртуальным частным шлюзом и указываете сетевой интерфейс своего устройства в качестве цели для трафика VPC;

– таблица маршрутов транзитного шлюза (transit gateway route table): таблица маршрутов, связанная с транзитным шлюзом.

– таблица маршрутов локального шлюза (local gateway route table): таблица маршрутов, связанная с локальным шлюзом Outposts.

Таким образом, выбор использования распределенного веб-сервиса Route Table на AWS обоснован его ключевой ролью в управлении трафиком между различными сегментами виртуальной сети. Route Table позволяет определять, куда направлять сетевой трафик в зависимости от его назначения, обеспечивая эффективную маршрутизацию внутри облачной инфраструктуры. В контексте DevOps, это средство позволяет настраивать маршруты автоматически и динамически в зависимости от изменений в инфраструктуре, что соответствует принципам непрерывной поставки и автоматизации процессов развертывания. Использование Route Table в проекте обеспечивает гибкость и отказоустойчивость в обработке сетевого трафика между распределенными компонентами приложения, что является необходимым условием для обеспечения высокой доступности и производительности сервисов в облаке. Таким образом, включение Route Table в рамки проекта укрепляет его архитектурную гибкость и соответствие современным практикам DevOps.

3.2.3 Internet Gateway

Интернет-шлюз – это горизонтально масштабируемый, резервный и высокодоступный компонент VPC, обеспечивающий связь между вашим VPC и Интернетом. Он поддерживает трафик IPv4 и IPv6. Он не создает рисков доступности и не ограничивает пропускную способность сетевого трафика.

Интернет-шлюз позволяет ресурсам в публичных подсетях (например, экземплярам EC2) подключаться к Интернету, если ресурс имеет публичный

IPv4-адрес или IPv6-адрес. Аналогично, ресурсы в Интернете могут инициировать подключение к ресурсам в вашей подсети, используя публичный IPv4-адрес или IPv6-адрес. Например, интернет-шлюз позволяет подключиться к экземпляру EC2 в AWS с помощью вашего локального компьютера.

Интернет-шлюз предоставляет цель в таблицах маршрутизации VPC для трафика, маршрутизируемого через Интернет. Для связи по протоколу IPv4 интернет-шлюз также выполняет трансляцию сетевых адресов (NAT). Для связи с использованием IPv6 NAT не нужен, поскольку адреса IPv6 являются общедоступными.

Распределенный веб-сервис Internet Gateway (IGW) на AWS играет ключевую роль в обеспечении доступа к интернету для виртуальных ресурсов внутри Amazon Virtual Private Cloud (VPC). Он представляет собой компонент инфраструктуры, который позволяет виртуальным серверам и сервисам в VPC обмениваться данными с внешним миром, обеспечивая тем самым возможность доступа к внешним ресурсам, веб-сервисам и клиентам через интернет.

В контексте дипломного проекта "DevOps технологии поддержки распределенных Web сервисов для AWS с использованием Terraform", выбор использования Internet Gateway обоснован необходимостью обеспечения связности и доступности веб-приложений и сервисов, развернутых в облаке AWS. Использование IGW позволяет гарантировать высокую доступность внешнего доступа к веб-приложениям, а также обеспечивает масштабируемость и гибкость в управлении сетевым трафиком. Кроме того, Internet Gateway интегрируется с другими сервисами AWS, такими как Route Table и Security Group, что позволяет создавать комплексные сетевые конфигурации с учетом требований безопасности и производительности. Таким образом, использование распределенного веб-сервиса Internet Gateway в рамках проекта является необходимым компонентом для обеспечения полной функциональности и доступности веб-приложений в облачной среде AWS.

3.2.4 Public Subnet

Распределенный веб-сервис Public Subnet (подсеть имеет прямой маршрут к интернет-шлюзу, ресурсы в публичной подсети могут получить доступ к публичному Интернету) на AWS представляет собой сегмент виртуальной сети Amazon Virtual Private Cloud (VPC), который имеет доступ к интернету через Internet Gateway и может использоваться для размещения публичных ресурсов, доступных извне. Public Subnet обеспечивает

возможность хранения и обработки данных в облаке AWS с открытым доступом из интернета, что включает в себя веб-серверы, API-шлюзы, а также другие компоненты, требующие внешнего доступа.

Выбор использования Public Subnet обоснован необходимостью развертывания веб-приложений и сервисов, которые должны быть доступны извне, например, для веб-трафика пользователей или внешних API запросов. Использование Public Subnet позволяет изолировать публичные ресурсы от приватных, обеспечивая тем самым дополнительный уровень безопасности и контроля доступа к важным данным и сервисам.

3.2.5 Security Group

Распределенный веб-сервис Security Group на AWS представляет собой виртуальный брандмауэр, который контролирует трафик входящий и исходящий для экземпляров Amazon EC2 в Amazon Virtual Private Cloud (VPC). Security Group позволяет определять правила доступа на основе IP-адресов, портов и протоколов, обеспечивая тем самым безопасность виртуальных серверов и приложений.

Выбор использования Security Group обоснован необходимостью обеспечения безопасности веб-приложений и сервисов в облаке AWS. Использование Security Group позволяет настраивать правила доступа к виртуальным серверам и сервисам в соответствии с требованиями безопасности проекта, например, блокируя доступ к нежелательным IP-адресам или разрешая доступ только определенным портам и протоколам.

Кроме того, Security Group обеспечивает гибкость и управляемость в управлении правилами безопасности, позволяя администраторам быстро реагировать на изменения в требованиях безопасности и внесение соответствующих правок в конфигурацию. Таким образом, использование распределенного веб-сервиса Security Group в рамках проекта обеспечивает надежную защиту виртуальных серверов и приложений в облачной среде AWS, что является критически важным аспектом для обеспечения безопасности и целостности данных и сервисов.

3.2.6 EC2

Эластичное облако вычислений Amazon (Amazon EC2) предлагает самую широкую и глубокую вычислительную платформу с более чем 750 инстансами и набором новейших процессоров, хранилищ, сетей, операционных систем и моделей покупок, обеспечивая должное соответствие нуждам конкретной рабочей нагрузки. AWS EC2 – первый крупный облачный

провайдер, который поддерживает работу процессоров Intel, AMD и Arm, единственное облако с инстансами EC2 Mac по требованию и с сетью Ethernet 400 Гбит/с. AWS EC2 предлагает лучшее соотношение цены и производительности машинного обучения, а также самую низкую стоимость инстансов логических выводов в облаке. На AWS выполняется больше рабочих нагрузок SAP, высокопроизводительных вычислений (HPC), машинного обучения и Windows, чем в любом другом облаке.

На следующей схеме (рисунок 3.4) показана базовая архитектура экземпляра Amazon EC2, развернутого в виртуальном частном облаке Amazon (VPC). В этом примере экземпляр EC2 находится в зоне доступности в регионе. Экземпляр EC2 защищен с помощью группы безопасности, которая представляет собой виртуальный брандмауэр, контролирующий входящий и исходящий трафик. Закрытый ключ хранится на локальном компьютере, а открытый ключ – на экземпляре. Оба ключа указываются как ключевая пара для подтверждения личности пользователя. В этом сценарии за экземпляром закреплен том Amazon EBS. VPC взаимодействует с Интернетом с помощью интернет-шлюза.

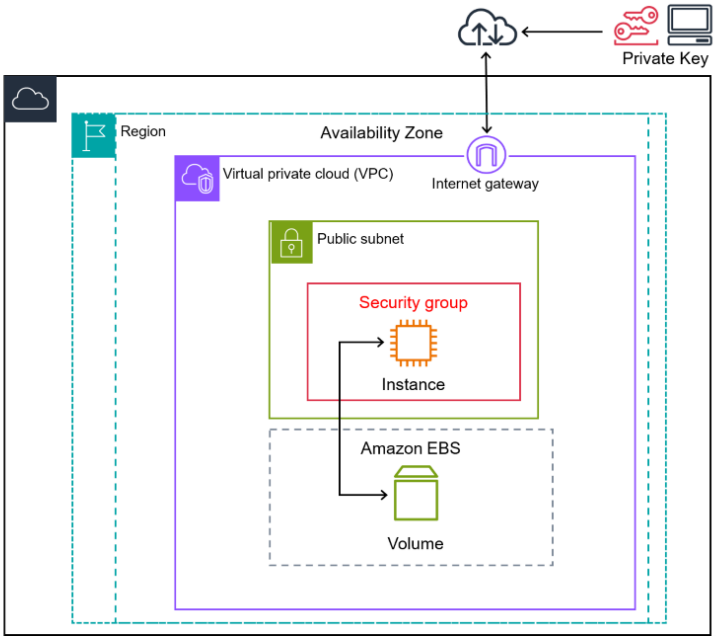

Рисунок 3.4 – Базовая архитектура экземпляра AWS EC2 [14]

Amazon Elastic Compute Cloud (Amazon EC2) предоставляет масштабируемые вычислительные мощности по требованию в облаке Amazon Web Services (AWS). Использование Amazon EC2 позволяет сократить расходы на аппаратное обеспечение и ускорить разработку и развертывание

приложений. С помощью Amazon EC2 можно запустить столько виртуальных серверов, сколько нужно, настроить безопасность и сетевое взаимодействие, а также управлять хранилищем. Помимо этого, можно увеличить мощность (масштабирование) для решения задач, требующих больших вычислений, таких как ежемесячные или ежегодные процессы или скачки посещаемости веб-сайта. При снижении нагрузки можно снова сократить мощность (уменьшить масштаб).

Amazon EC2 предоставляет следующие возможности высокого уровня:

– инстансы (instances): виртуальные серверы;

– образы машин Amazon (Amazon Machine Images): предварительно настроенные шаблоны для ваших экземпляров, в которых собраны компоненты, необходимые для вашего сервера (включая операционную систему и дополнительное программное обеспечение);

– типы экземпляров (Instances types): различные конфигурации процессора, памяти, хранилища, сетевых мощностей и графического оборудования для ваших экземпляров;

– пары ключей (Key pairs): защищенная информация для входа в систему для ваших экземпляров. AWS хранит открытый ключ, а вы храните закрытый ключ в безопасном месте;

– тома хранилища инстансов (Instance store volumes): тома для хранения временных данных, которые удаляются при остановке, спящем режиме или завершении работы экземпляра;

– тома Amazon EBS: постоянные тома для хранения данных с помощью Amazon Elastic Block Store (Amazon EBS);

– регионы и зоны (Regions and Zones): несколько физических местоположений для ваших ресурсов, таких как экземпляры и тома Amazon EBS;

– группы безопасности (security groups): виртуальный брандмауэр, позволяющий указать протоколы, порты и диапазоны IP-адресов источников, через которые могут подключаться ваши инстансы, а также диапазоны IP-адресов получателей, к которым могут подключаться ваши инстансы.

– эластичные IP-адреса (elastic ip addresses): статические IPv4-адреса для динамических облачных вычислений.

– теги (tags): метаданные, которые можно создавать и назначать ресурсам Amazon EC2;

– виртуальные частные облака (Virtual Private Clouds): виртуальные сети, которые вы можете создать, логически изолированные от остальной части облака AWS.

Виртуальные сети можно создать, чтобы они были логически изолированные от остальной части облака AWS. При желании можно подключить эти виртуальные сети к своей собственной сети.

Выбор использования EC2 в контексте дипломного проекта обоснован необходимостью предоставления инфраструктурных ресурсов для развертывания и запуска веб-приложения и его сервисов. EC2 обеспечивает гибкость в выборе типа и размера инстансов, что позволяет оптимизировать использование ресурсов и обеспечить соответствие требованиям проекта в плане производительности и масштабируемости.

3.2.7 ECR и ECS

Реестр эластичных контейнеров Amazon (Amazon ECR) – это полностью управляемый реестр контейнеров, который предлагает высокопроизводительный хостинг для надежного развертывания образов и артефактов приложений в любом месте (рисунок 3.5).

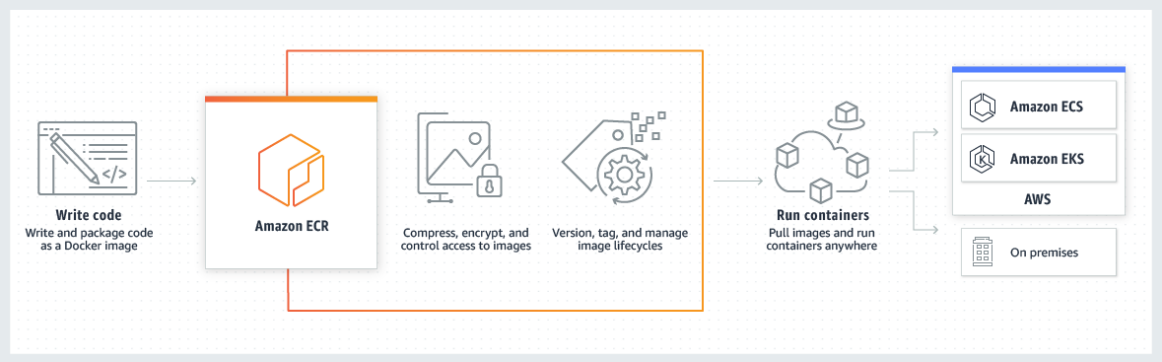

Рисунок 3.5 – Принцип работы AWS ECR [15]

Стандартные примеры использования AWS Elastic Container Registry:

– контроль уязвимостей программного обеспечения;

– оптимизация рабочих нагрузок по развертыванию;

– управление политиками жизненного цикла образов.

Контроль уязвимостей ПО заключается в обеспечении соответствия требований по безопасности образа с помощью тесно интегрированного сервиса Amazon Inspector для контроля уязвимостей, позволяющего автоматизировать оценку уязвимостей и направление запросов на исправление.

Оптимизация рабочих нагрузок по развертыванию заключается в публикации контейнерных приложений с помощью одной команды и легкая интеграция среды с самостоятельным управлением.

Управление политиками жизненного цикла образов заключается в автоматическом сохранении последних образов и отправление их в ненужный архив. Использование правил и тегов для быстрого доступа к образам.

Amazon Elastic Container Service (ECS) – это управляемый сервис контейнеров от Amazon Web Services (AWS), который позволяет легко запускать, управлять и масштабировать контейнеризированные приложения с использованием технологий Docker и Kubernetes. ECS обеспечивает высокую доступность, масштабируемость и безопасность при развертывании приложений в контейнерах (рисунок 3.6).

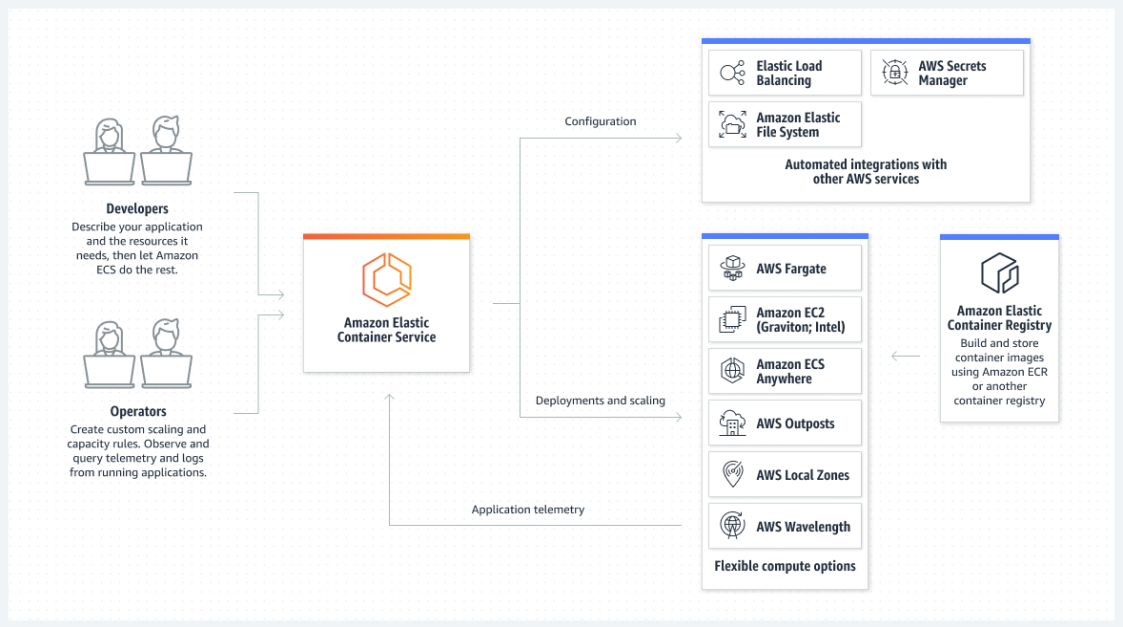

Рисунок 3.6 – Amazon ECS: управление приложениями с гибкими возможностями масштабирования и автоматической интеграцией AWS [16]

На диаграмме показано, как Amazon ECS запускает, отслеживает и масштабирует приложения с помощью гибких вычислительных возможностей с автоматической интеграцией с другими сервисами AWS. Разработчики описывают свои приложения и необходимые ресурсы, а Amazon ECS делает все остальное. Операторы выполняют системные операции, такие как создание пользовательских правил масштабирования и вместимости, а также просматривают и запрашивают данные из журналов приложений и телеметрии. Три раздела отображаются слева направо.

В первом разделе представлена иллюстрация двух разных персонажей, расположенных вертикально друг под другом. Иллюстрация в верхней части раздела называется «Разработчики». В описании говорится: «Опишите свое приложение и необходимые ресурсы, а остальное сделает Amazon ECS».

Приведенная ниже иллюстрация называется «Операторы», а в описании говорится: «Создайте собственные правила масштабирования и вместимости. Просматривайте и запрашивайте данные из журналов приложений и телеметрии». На обеих иллюстрациях есть стрелки, указывающие на второй раздел.

Во втором разделе приведена иллюстрация значка Amazon ECS. Раздел называется «Эластичный контейнерный сервис Amazon». Две стрелки указывают на разные части третьего раздела. Первая стрелка вверху называется «Конфигурация», а вторая стрелка ниже – «Развертывание и масштабирование».

Третий раздел состоит из двух частей. Первая часть сверху расположена возле стрелки от второго раздела, которая называется «Конфигурация». Эта часть называется «Автоматическая интеграция с другими сервисами AWS». Расположенные рядом значки отображают различные сервисы AWS: «Эластичная балансировка нагрузки», «Менеджер секретов AWS» и «Эластичная файловая система Amazon».

Вторая часть третьего раздела расположена возле стрелки от второго раздела, которая называется «Развертывание и масштабирование». Раздел называется «Гибкие вычислительные среды». Расположенные рядом значки отображают различные варианты вычислений AWS: «AWS Fargate», «Amazon EC2 Graviton; Intel», «Amazon ECS Anywhere», «AWS Outposts», «Локальные зоны AWS» и «AWS Wavelength». На соседней иллюстрации справа есть стрелка, указывающая на этот список. Эта иллюстрация называется «Реестр эластичных контейнеров Amazon» и в ней говорится: «Создавайте и храните образы контейнеров с помощью Amazon ECR или других образов контейнеров».

Для того, чтобы использовать ESC достаточно просто описать проект, его зависимости и необходимые ресурсы, а эластичный сервис контейнеров Amazon (Amazon ECS) будет запускать, контролировать, а также масштабировать данный проект, используя гибкие возможности вычислительных ресурсов благодаря автоматической интеграции с другими вспомогательными сервисами AWS. Также можно выполнять системные операции, такие как создание пользовательских правил масштабирования и вместимости, а также просматривайте и запрашивайте данные из журналов приложений и телеметрии.

Таким образом, выбор использования распределенных веб-сервисов ECR и ECS обоснован их ключевой ролью в упаковке, развертывании и управлении контейнеризированными приложениями в облачной среде AWS. Эти сервисы обеспечивают гибкость, масштабируемость и безопасность в развертывании и управлении приложениями, что является важным аспектом в

рамках проекта по разработке и поддержке распределенных веб-сервисов с использованием DevOps практик.

3.2.8 Elastic IP

Эластичный IP-адрес – это статический IPv4-адрес, предназначенный для динамических облачных вычислений. Эластичный IP-адрес выделяется учетной записи AWS и принадлежит до тех пор, пока пользователь его не освободит. Используя Elastic IP-адрес, можно замаскировать отказ экземпляра или программного обеспечения, быстро переназначив адрес на другой экземпляр в аккаунте пользователя. Кроме того, можно указать Elastic IP-адрес в записи DNS для вашего домена, чтобы ваш домен указывал на ваш экземпляр.

Таким образом, Elastic IP-адрес – это публичный IPv4-адрес, доступный из интернета. Если у экземпляра нет публичного IPv4-адреса, то его можно связать с Elastic IP-адрес, чтобы обеспечить связь с Интернетом.

Основные преимущества использования Elastic IP:

1 Стабильность и надежность: EIP предоставляет постоянный IP-адрес, который не изменяется при перезапуске экземпляров EC2 или других виртуальных ресурсов, что обеспечивает стабильный доступ к приложениям.

2 Управление и масштабируемость: EIP позволяет управлять IP-адресами и связывать их с различными экземплярами EC2 и другими ресурсами в облаке AWS. Это обеспечивает гибкость при настройке сетевой инфраструктуры и масштабировании приложений.

3 Бесплатная зона переноса: AWS предоставляет бесплатную зону переноса для EIP, что позволяет безопасно и эффективно переносить IP-адреса между различными экземплярами EC2 и регионами AWS.

4 Обеспечение безопасности: Использование EIP позволяет избежать блокировки IP-адресов в списке антиспам-фильтров или других систем безопасности, так как он остается постоянным и надежным.

Выбор использования EIP обусловлен необходимостью обеспечения постоянной и стабильной точки доступа к веб-сервису, работающим в облаке AWS, что будет обеспечивать стабильную и надежную точку доступа к веб-сервису в облаке AWS, а это является ключевым аспектом для обеспечения непрерывной работы и доступности веб-сервисов.

3.2.9 IAM Policy

С помощью IAM Policy можно управлять доступом в AWS, создавая политики и прикрепляя их к IAM-идентификаторам (пользователям, группам

пользователей или ролям) или ресурсам AWS. Политика – это объект в AWS, который, будучи связанным с идентификатором или ресурсом, определяет их разрешения. AWS оценивает эти политики, когда принципал IAM (пользователь или роль) делает запрос. Разрешения в политиках определяют, будет ли запрос разрешен или отклонен. Большинство политик хранятся в AWS в виде документов JSON. AWS поддерживает шесть типов политик: политики на основе идентификации, политики на основе ресурсов, границы разрешений, организационные SCP, ACL и политики сессий.

Политики IAM определяют разрешения на действие независимо от метода, который вы используете для выполнения операции. Например, если политика разрешает действие GetUser, то пользователь с такой политикой может получить информацию о пользователе из AWS Management Console, AWS CLI или AWS API. При создании пользователя IAM вы можете выбрать, разрешить ли ему консольный или программный доступ. Если доступ к консоли разрешен, пользователь IAM может войти в консоль, используя свои учетные данные. Если разрешен программный доступ, пользователь может использовать ключи доступа для работы с CLI или API.

Распределенный веб-сервис IAM Policy на AWS (Identity and Access Management) играет важную роль в управлении доступом к ресурсам и сервисам облака. Выбор использования IAM Policy обусловлен необходимостью обеспечения безопасного и гибкого управления доступом к различным ресурсам в облаке AWS.

Основные преимущества использования IAM Policy:

1 Гранулированное управление доступом: IAM Policy позволяет создавать гранулированные политики доступа, определяющие, какие пользователи или роли могут выполнять какие операции над определенными ресурсами. Это обеспечивает принцип минимальных привилегий и улучшает безопасность инфраструктуры.

2 Гибкость и масштабируемость: IAM Policy позволяет настраивать политики доступа для различных типов ресурсов, включая EC2, S3, RDS и другие, а также для различных типов действий, таких как чтение, запись, удаление и т. д. Это обеспечивает гибкость в настройке доступа и масштабируемость при добавлении новых ресурсов и пользователей.

3 Интеграция с другими сервисами AWS: IAM Policy интегрируется с другими сервисами AWS, такими как S3, EC2, RDS и другими, обеспечивая возможность управления доступом к различным ресурсам и сервисам из единого интерфейса.

4 Мониторинг и аудит доступа: IAM Policy позволяет отслеживать и анализировать действия пользователей и ролей в рамках вашей учетной записи

AWS, обеспечивая возможность мониторинга и аудита доступа для обеспечения безопасности и соответствия требованиям.

Использование IAM Policy в контексте дипломного проекта обеспечивает необходимый уровень безопасности и контроля доступа к ресурсам и сервисам облака AWS. Помимо этого, использование IAM Policy необходимо для реализации распределенного веб-сервиса CloudWatch. Это позволяет эффективно управлять пользователями, ролями и политиками безопасности, обеспечивая соблюдение правил доступа и минимизацию рисков безопасности.

3.2.10 IAM Role

IAM Role – это идентификатор IAM, который можно создать в учетной записи и который имеет определенные разрешения. Роль IAM похожа на IAM User (создание пользователей) в том смысле, что это идентификатор AWS с политиками разрешений, которые определяют, что идентификатор может и чего не может делать в AWS. Однако вместо того, чтобы быть уникально связанной с одним человеком, роль предназначена для того, чтобы ее мог взять на себя любой, кому она нужна. Кроме того, роль не имеет стандартных долгосрочных учетных данных, таких как пароль или ключи доступа, связанных с ней. Вместо этого, когда принимается роль пользователем, она предоставляет вам временные учетные данные безопасности для сеанса работы с ролью.

Роли можно использовать для делегирования доступа пользователям, приложениям или службам, которые обычно не имеют доступа к ресурсам AWS. Например, можно предоставить пользователям учетной записи AWS доступ к ресурсам, которые они обычно не имеют, или предоставить пользователям одной учетной записи AWS доступ к ресурсам другой учетной записи. Или можно разрешить мобильному приложению использовать ресурсы AWS, но не нужно встраивать ключи AWS в приложение (где их сложно обновлять и где пользователи могут их извлечь). Иногда необходимо просто предоставить доступ к AWS пользователям, у которых уже есть идентификаторы, определенные вне AWS, например в корпоративном каталоге. Или можно предоставить доступ к своей учетной записи третьим лицам, чтобы они могли провести аудит ваших ресурсов.

Выбор использования распределенного веб-сервиса IAM Role на AWS обусловлен потребностью в безопасном и гибком управлении доступом к ресурсам облака, особенно в среде с множеством различных сервисов. IAM Role предоставляет временные учетные записи с ограниченными

привилегиями, которые могут использоваться временно или автоматически для выполнения определенных задач или операций.

3.2.11 CloudWatch

Amazon CloudWatch – это служба, которая отслеживает работу приложений, реагирует на изменения производительности, оптимизирует использование ресурсов и предоставляет информацию о состоянии операционной системы. Собирая данные по всем ресурсам AWS, CloudWatch обеспечивает видимость производительности всей системы и позволяет пользователям устанавливать сигналы тревоги, автоматически реагировать на изменения и получать единое представление о работоспособности системы (рисунок 3.7).

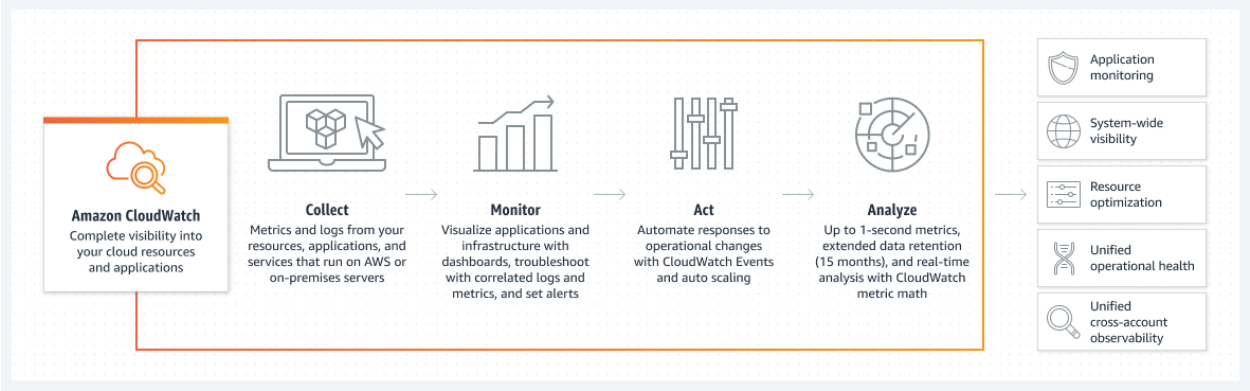

Рисунок 3.7 – Принцип работы Amazon CloudWatch [17]

Amazon CloudWatch собирает и визуализирует журналы, метрики и данные о событиях в реальном времени в виде автоматизированных панелей, чтобы упростить обслуживание инфраструктуры и приложений.

Выбор использования распределенного веб-сервиса CloudWatch на AWS обусловлен потребностью в мониторинге, анализе и управлении ресурсами облака для обеспечения их надежной и эффективной работы. CloudWatch предоставляет широкий спектр инструментов для сбора, отображения и анализа метрик, журналов и событий, что позволяет оперативно реагировать на изменения и проблемы в инфраструктуре.

Основные причины выбора CloudWatch:

1 Мониторинг и анализ метрик: CloudWatch позволяет собирать и отображать метрики по всем ресурсам облака, включая вычислительные ресурсы, хранилище данных, сетевые ресурсы и многие другие. Это позволяет

оперативно отслеживать производительность и использование ресурсов, выявлять узкие места и оптимизировать работу инфраструктуры.

2 Мониторинг журналов и событий: CloudWatch позволяет собирать, анализировать и мониторить журналы и события с различных ресурсов, что позволяет выявлять и анализировать проблемы, ошибки и нештатные ситуации. Это помогает оперативно реагировать на проблемы и обеспечивать надежную работу приложений и сервисов.

3 Управление и оптимизация ресурсов: CloudWatch предоставляет инструменты для анализа использования ресурсов и оптимизации их работы. Например, автоматическое масштабирование группы EC2-инстансов или использование правил мониторинга для запуска действий при определенных событиях помогает эффективно использовать ресурсы и сокращать затраты.

4 Интеграция с другими сервисами AWS: CloudWatch интегрируется с другими сервисами AWS, что позволяет использовать его в сочетании с другими инструментами для автоматизации процессов мониторинга и управления ресурсами. Например, события CloudWatch могут запускать Lambda-функции для автоматического реагирования на изменения в инфраструктуре.

Таким образом, использование распределенного веб-сервиса CloudWatch на AWS обеспечивает надежный и эффективный мониторинг, анализ и управление ресурсами облака, что делает его важным компонентом для обеспечения безопасности, надежности и эффективности работы в облачной среде.

3.3 Контейнеризация и оркестрация с помощью Docker и Docker Compose

Docker – это открытая платформа для разработки, доставки и запуска приложений. Docker позволяет отделить приложения от инфраструктуры, чтобы быстро поставлять программное обеспечение. С помощью Docker вы можете управлять инфраструктурой так же, как и приложениями. Воспользовавшись методологиями Docker для доставки, тестирования и развертывания кода, вы сможете значительно сократить время между написанием кода и его запуском в производство.

Docker позволяет упаковывать и запускать приложения в слабо изолированной среде, называемой контейнером. Изоляция и безопасность позволяют запускать множество контейнеров одновременно на одном хосте. Контейнеры легковесны и содержат все необходимое для работы приложения, поэтому вам не нужно полагаться на то, что установлено на хосте. Помимо этого, можно делиться контейнерами во время работы и быть уверенным, что

все, кто получат образ, получат тот же контейнер, который работает точно так же.

Docker предоставляет инструменты и платформу для управления жизненным циклом контейнеров:

– разработайте свое приложение и его вспомогательные компоненты с помощью контейнеров;

– контейнер становится единицей для распространения и тестирования вашего приложения.

Когда вы будете готовы, разверните приложение в производственной среде в виде контейнера или оркестрированной службы. Это работает одинаково независимо от того, является ли ваша производственная среда локальным центром обработки данных, облачным провайдером или гибридом этих двух систем.

Docker оптимизирует жизненный цикл разработки, позволяя разработчикам работать в стандартизированных средах с использованием локальных контейнеров, в которых хранятся ваши приложения и сервисы. Контейнеры отлично подходят для рабочих процессов непрерывной интеграции и непрерывной доставки (CI/CD).

Рассмотрим следующий пример:

1 Разработчики пишут код локально и делятся своей работой с коллегами с помощью контейнеров Docker.

2 С помощью Docker они переносят свои приложения в тестовую среду и запускают автоматические и ручные тесты.

3 Когда разработчики находят ошибки, они могут исправить их в среде разработки и перенести в тестовую среду для проверки и подтверждения.

4 Когда тестирование завершено, доставить исправление клиенту достаточно просто – выложить обновленный образ в производственную среду.

Выполнение большего количества рабочих нагрузок на одном и том же оборудовании. Docker – легкий и быстрый, а также представляет собой жизнеспособную и экономически эффективную альтернативу виртуальным машинам на базе гипервизора, что позволяет использовать больше серверных мощностей для достижения бизнес-целей. Docker идеально подходит для сред с высокой плотностью размещения, а также для малых и средних развертываний, где вам нужно делать больше с меньшими ресурсами.

Архитектура Docker (рисунок 3.8). Docker использует клиент-серверную архитектуру. Клиент Docker общается с демоном Docker, который выполняет всю работу по созданию, запуску и распространению контейнеров Docker. Клиент Docker и демон могут работать в одной системе, или вы можете подключить клиента Docker к удаленному демону Docker. Клиент Docker и демон взаимодействуют с помощью REST API, через сокеты UNIX или

сетевой интерфейс. Еще один клиент Docker - Docker Compose, позволяющий работать с приложениями, состоящими из набора контейнеров.

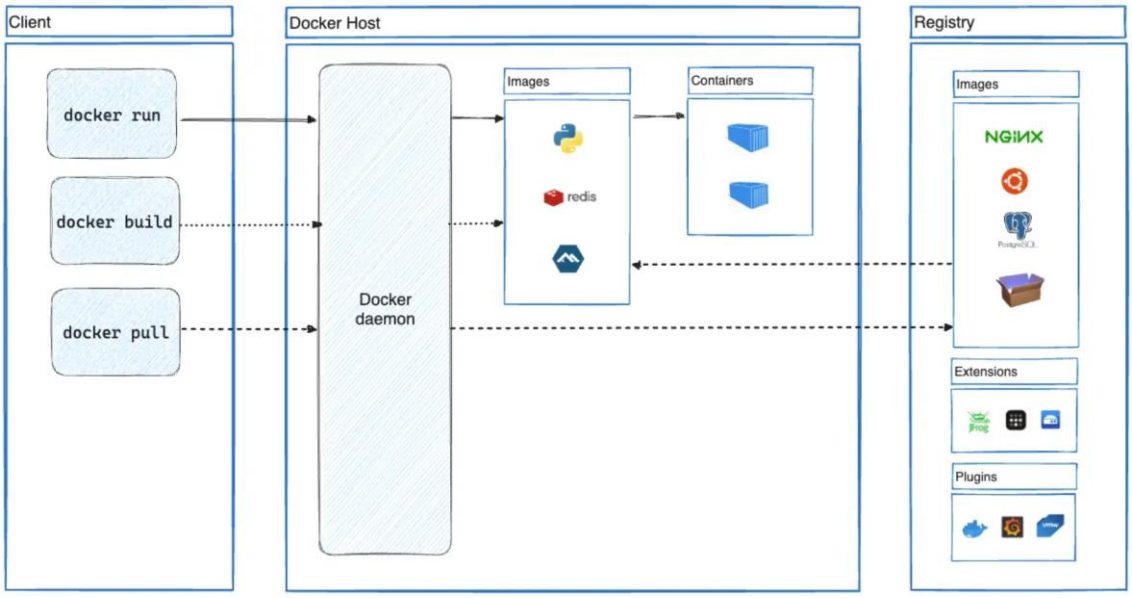

Рисунок 3.8 – Архитектура Docker [18]

Демон Docker (dockerd) прослушивает запросы API Docker и управляет объектами Docker, такими как образы, контейнеры, сети и тома. Демон также может взаимодействовать с другими демонами для управления службами Docker.

Клиент Docker (docker) – это основной способ взаимодействия многих пользователей Docker с Docker. Когда вы используете такие команды, как docker run, клиент отправляет эти команды dockerd, который их выполняет. Команда docker использует API Docker. Клиент Docker может взаимодействовать с несколькими демонами.

Образ (Image) – это шаблон, доступный только для чтения, с инструкциями по созданию контейнера Docker. Часто образ основан на другом образе с некоторыми дополнительными настройками. Например, можно создать образ, основанный на образе ubuntu, но устанавливающий веб-сервер Apache и статический веб-сайт, а также детали конфигурации, необходимые для работы этого статического веб-сайта.

Помимо этого, можно создавать собственные образы или использовать только те, которые созданы другими и опубликованы в реестре. Чтобы создать свой собственный образ, необходимо создать Dockerfile с простым синтаксисом для определения шагов, необходимых для создания образа и его запуска. Почти каждая инструкция в Dockerfile создает слой в образе, который имеет какой-либо объем (RUN, COPY, ADD, CMD).

Когда происходит изменение Dockerfile и пересобирание образа, пересобираются только те слои, которые изменились. Именно это делает образы такими легкими, маленькими и быстрыми по сравнению с другими технологиями виртуализации.

Контейнер – это запускаемый экземпляр образа. Можно создавать, запускать, останавливать, перемещать или удалять контейнер с помощью Docker API или CLI. Контейнер можно подключить к одной или нескольким сетям, присоединить к нему хранилище или даже создать новый образ на основе его текущего состояния.

По умолчанию контейнер относительно хорошо изолирован от других контейнеров и хост-машины. Можно управлять тем, насколько изолированы сеть, хранилище и другие базовые подсистемы контейнера от других контейнеров или от хост-машины.

Контейнер определяется его образом, а также любыми параметрами конфигурации, которые вы предоставляете ему при создании или запуске. Когда контейнер удаляется, все изменения его состояния, не сохраненные в постоянном хранилище, исчезают.

Docker написан на языке программирования Go и использует ряд возможностей ядра Linux для обеспечения своей функциональности. Docker использует технологию пространств имен для создания изолированного рабочего пространства, называемого контейнером. Когда вы запускаете контейнер, Docker создает набор пространств имен для этого контейнера.

Эти пространства имен обеспечивают уровень изоляции. Каждый аспект контейнера выполняется в отдельном пространстве имен, и доступ к нему ограничен этим пространством.

Docker Compose – это инструмент для определения и запуска многоконтейнерных приложений. Это ключ к оптимизации и повышению эффективности разработки и развертывания.

Compose упрощает управление всем стеком приложений, позволяя легко управлять сервисами, сетями и томами в едином и понятном конфигурационном файле YAML. Затем с помощью одной команды вы создаете и запускаете все службы из своего конфигурационного файла.

Compose работает во всех средах: продакшн (production), стейджинг (staging), разработка (dev), тестирование (testing), а также в рабочих процессах CI. В нем также есть команды для управления всем жизненным циклом вашего приложения:

– запуск, остановка и восстановление служб;

– просмотр состояния запущенных служб;

– потоковый вывод журнала запущенных служб;

– выполнение разовой команды для службы.

Docker Compose опирается на конфигурационный файл YAML, который обычно называется compose.yaml или docker-compose.yml.

Файл compose.yaml соответствует правилам определения многоконтейнерных приложений, изложенным в спецификации Compose.

Модель Docker Compose. Вычислительные компоненты приложения определяются как сервисы. Сервис – это абстрактная концепция, реализуемая на платформах путем запуска одного и того же образа контейнера и его конфигурации один или несколько раз.

Сервисы взаимодействуют друг с другом через сети. В спецификации Compose сеть – это абстракция возможностей платформы для создания IP-маршрута между контейнерами внутри сервисов, соединенных между собой.

Сервисы хранят и обмениваются постоянными данными в томах. Спецификация описывает такие постоянные данные как высокоуровневое монтирование файловой системы с глобальными опциями.

Некоторые сервисы требуют конфигурационных данных, которые зависят от времени выполнения или платформы. Для этого в спецификации определена специальная концепция configs. С точки зрения контейнера сервисов, конфигурации можно сравнить с томами, поскольку они представляют собой файлы, монтируемые в контейнер. Однако фактическое определение включает в себя отдельные ресурсы платформы и сервисы, которые абстрагируются этим типом.

Секрет – это особый вид конфигурационных данных для конфиденциальных данных, которые не должны быть открыты без учета соображений безопасности. Секреты предоставляются сервисам в виде файлов, монтируемых в их контейнеры, но ресурсы платформы для предоставления конфиденциальных данных достаточно специфичны, чтобы заслужить отдельное понятие и определение в спецификации Compose.

Проект – это отдельное развертывание спецификации приложения на платформе. Имя проекта, задаваемое с помощью атрибута name верхнего уровня, используется для объединения ресурсов в группы и изоляции их от других приложений или других установок того же приложения, специфицированного Compose, с различными параметрами. Если вы создаете ресурсы на платформе, вы должны префиксировать имена ресурсов проектом и установить метку com.docker.compose.project.

Compose предлагает возможность задать пользовательское имя проекта и переопределить его, чтобы один и тот же файл compose.yaml можно было развернуть дважды на одной и той же инфраструктуре без изменений, просто передав другое имя.

Использование инструмента контейнеризации Docker и его оркестрации с помощью Docker Compose, а также дальнейшее использование Amazon

Elastic Container Registry (ECR) и Amazon Elastic Container Service (ECS) являются важным компонентом облачной инфраструктуры по нескольким причинам.

Во-первых, Docker обеспечивает стандартизацию и изоляцию приложений, позволяя упаковывать приложения и все их зависимости в контейнеры, что делает их переносимыми и независимыми от окружения. Это позволяет значительно упростить процесс развертывания и масштабирования приложений, а также обеспечивает консистентность в различных средах, включая разработку, тестирование и производство.

Во-вторых, Docker Compose предоставляет удобный способ определения и управления многоконтейнерными приложениями с помощью простого файла конфигурации. Это позволяет легко описывать связи и зависимости между сервисами, определять сетевые параметры, тома данных и переменные окружения, что делает развертывание и управление приложениями более эффективным и прозрачным.

Третье, Amazon ECR предоставляет управляемый сервис для хранения, управления и развертывания контейнеров Docker, обеспечивая безопасное и надежное хранение образов контейнеров и интеграцию с другими сервисами AWS.

Четвертое, Amazon ECS предоставляет высокопроизводительный и масштабируемый сервис для управления контейнерами Docker в облачной среде AWS. Он позволяет запускать, управлять и масштабировать контейнеры на основе заданий и кластеров, автоматизируя процесс развертывания и управления инфраструктурой.

Таким образом, использование Docker, Docker Compose, ECR и ECS вместе обеспечивает полный цикл разработки, развертывания и управления многоконтейнерными приложениями в облачной среде, обеспечивая высокую производительность, масштабируемость и надежность приложений.

3.4 Описание и обоснование использования CI/CD

Непрерывная интеграция (Continuous Integration, CI) и непрерывная поставка (Continuous Delivery, CD) представляют собой культуру, набор принципов и практик, которые позволяют разработчикам чаще и надежнее развертывать изменения программного обеспечения.

CI/CD – это одна из DevOps-практик. Она также относится и к agile-практикам: автоматизация развертывания позволяет разработчикам сосредоточиться на реализации бизнес-требований, на качестве кода и безопасности.

Непрерывная интеграция (Continuous Integration, CI) и непрерывная доставка (Continuous Delivery, CD) являются подмножествами в рамках более крупного зонтика DevOps - проще говоря, это два основных процесса, которые обеспечивают методологию DevOps. Будучи двумя сторонами одной медали, они работают вместе, чтобы устранить сложности, связанные с постоянными инновациями. Необходимо разобрать эти два процесса (рисунок 3.9).

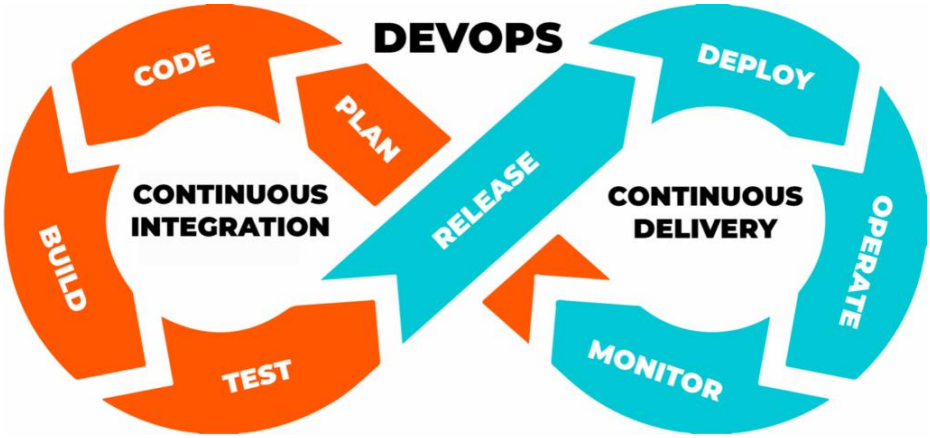

Рисунок 3.9 – DevOps и CI/CD [19]

Непрерывная интеграция (Continuous integration, CI) – это процесс разработки, заключающийся в автоматической сборке и выполнении модульных тестов после внесения изменений в исходный код. CI требует от команд разработчиков ежедневно по несколько раз интегрировать изменения кода в общий репозиторий исходного кода.

Основная цель непрерывной интеграции – создать последовательный, постоянный метод автоматической сборки и тестирования приложений, гарантирующий, что изменения, внесенные одним разработчиком, пригодны для использования во всей кодовой базе. Благодаря непрерывной интеграции разработчики могут решить проблемы, с которыми они сталкиваются при написании, интеграции, тестировании и доставке программных приложений конечным пользователям.

Непрерывная доставка (CD) – это продолжение непрерывной интеграции. Это процесс, в котором команды DevOps разрабатывают и поставляют полные части программного обеспечения в репозиторий – например, GitHub или реестр контейнеров – короткими, контролируемыми циклами. Непрерывная доставка делает релизы регулярными и предсказуемыми событиями для сотрудников DevOps и незаметными для конечных пользователей.

Еще одна цель непрерывной доставки – постоянно поддерживать код в состоянии, пригодном для развертывания, чтобы обновления могли выходить в любой момент без особых проблем. Программисты работают в течение одного-двухнедельного спринта, а не месяцами разрабатывают обновление. Таким образом, обновления могут появляться в программах каждые несколько недель, а не в одном большом годовом цикле выпуска. С помощью автоматизированного программного обеспечения код будет развернут на всех серверах, их приостановят, выведут код, убедятся, что он приземлился правильно, а затем снова включат и все это без проблем для пользователей.

CI можно рассматривать как первый шаг, а CD – как второй для создания и развертывания кода. CI – это скорее подготовка кода к выпуску (сборка/тестирование), а CD – собственно выпуск кода (выпуск/развертывание).

Таким образом, в контексте данного дипломного проекта, применение непрерывной интеграции и непрерывной поставки (CI/CD) обосновано сразу несколькими аспектами.

Во-первых, развертывание и обновление изменений веб-приложения с использованием распределенных веб-сервисов на платформе AWS требует некоторой степени автоматизации и контроля процесса. CI/CD позволяет автоматизировать сборку, тестирование и развертывания программного обеспечения, обеспечивая быстрое и надежное внедрение изменений в инфраструктуру. Это особенно важно для эффективного управления проектом.

Во-вторых, в контексте DevOps методологии, внедрение CI/CD способствует ускорению цикла разработки, улучшению качества и уменьшению рисков. Оно позволяет разработчикам чаще и безопаснее вносить изменения в код, а также проводить тестирование и развертывание в автоматизированном режиме. Это особенно важно для проектов, где высокая скорость разработки и поставки необходима для конкурентного преимущества, именно поэтому это является технологией в мире DevOps.

Частота использования CI/CD зависит от конкретного проекта и его требований. Однако, в современной разработке программного обеспечения, CI/CD является стандартной практикой, которая широко применяется в индустрии. Это подтверждается популярностью инструментов CI/CD, таких как Jenkins, CircleCI, GitLab CI/CD и других, а также активным обсуждением этой темы в сообществе разработчиков и специалистов по DevOps.

Исходя из описания, изучения и анализа практики DevOps – CI/CD, её применение оправдано как с технической, так и с методологической точек зрения, и соответствует современным требованиям и практикам разработки программного обеспечения.

3.5 Проектирование облачной инфраструктуры

Для проектирования облачной инфраструктуры необходимо предварительно изобразить инфраструктуру (рисунок 3.10).

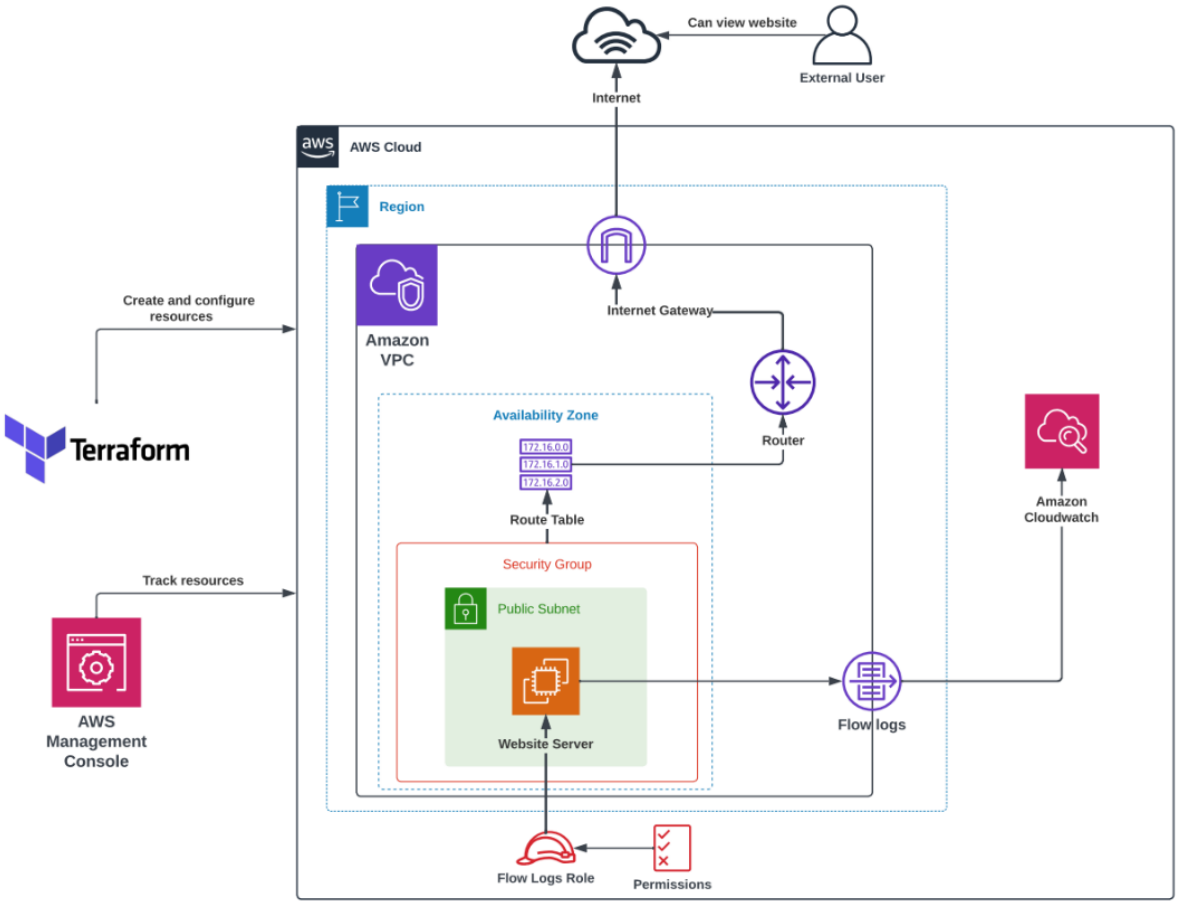

Рисунок 3.10 – Облачная инфраструктура

Облачная инфраструктура, основанная на AWS Cloud, представляет собой сложную систему, построенную для поддержки распределенных веб-сервисов. Внутри AWS Cloud располагается Region, который является выбором региона доступности. Одним из ключевых элементов этой инфраструктуры является Amazon VPC, который обеспечивает изоляцию и безопасность ресурсов. Внутри VPC используются доступные зоны доступности (availability zones), где развернуты экземпляры EC2, выступающие в качестве хостов для веб-сервисов.

Для обеспечения безопасности сети внутри каждой доступной зоны доступности используются Security Group и Public Subnet. Соединение между ними осуществляется через Route Table и Router, а трафик маршрутизируется

через Internet Gateway, обеспечивая доступ к веб-сервисам извне. Благодаря Route Table, Router, Internet Gateway и Public Subnet внешний пользователь сможет получить доступ к веб-сервису извне.

Контроль доступа и мониторинг безопасности обеспечиваются через использование IAM Policy (Permissions) и Flow Logs Role, интегрированных с EC2 Instance. Данные мониторинга передаются в Amazon Cloudwatch для последующего анализа и мониторинга.

AWS Management Console играет важную роль в управлении всей инфраструктурой, предоставляя возможность отслеживать и управлять ресурсами. Однако основным инструментом управления инфраструктурой является Terraform. Terraform позволяет создавать, конфигурировать и управлять всеми ресурсами в инфраструктуре, используя подход «инфраструктура как код».

Таким образом, проектирование облачной инфраструктуры для поддержки распределенных веб-сервисов на AWS с использованием Terraform – это сложный и многогранный процесс, охватывающий широкий спектр аспектов. Начиная с разработки сетевой архитектуры, где необходимо учитывать доступность зон и безопасность ресурсов, и заканчивая выбором и настройкой инструментов управления и мониторинга. Этот комплексный подход обеспечивает не только эффективное функционирование веб-сервисов, но и их масштабируемость, что является ключевым для успешного предоставления услуг в облачной среде.

4 ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ОБЛАЧНОЙ ИНФРАСТРУКТУРЫ ДЛЯ ВЕБ-СЕРВИСА

4.1 Обзор разворачиваемого веб-сервиса и используемых библиотек

Это развертываемое веб-приложение предназначено для управления распределенными веб-сервисами на платформе AWS с использованием DevOps технологий.

Это веб-приложение предоставляет возможность совместной работы над рисунками и контентом в режиме реального времени. Пользователи могут создавать онлайн-доски для рисования, добавлять различные элементы, такие как текст и стикеры, а также маркировать доски как избранные для быстрого доступа. Помимо этого, приложение предлагает различные методы аутентификации, включая почту, Google и Discord, для удобства пользователей. Функционал управления пользователями включает в себя приглашение пользователей по электронной почте и назначение им ролей в организации (например, Admin или Member). Пользователи также могут управлять своим профилем, выходить из учетной записи и отслеживать активные сессии. Кроме того, в реальном времени отображается список пользователей, работающих над определенной доской, обеспечивая прозрачность и синхронизацию в работе.

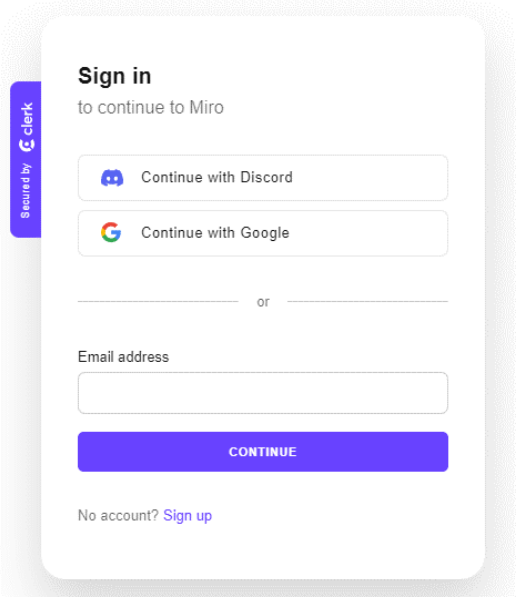

Рисунок 4.1 – Авторизация разворачиваемого веб-приложения

Авторизация, аутентификация и регистрация пользователей в веб-сервисе реализованы с помощью подключаемого инструмента (библиотеки)

Clerk, а хранение данных в Convex, который подключается в JWT Templates в панели управления Clerk.

Clerk – это инструмент для управления аутентификацией и авторизацией в веб-приложениях. Он предоставляет готовые решения для регистрации, входа пользователей и управления их учетными записями. Вместе с Convex, который используется для хранения данных, Clerk обеспечивает безопасность и удобство функций авторизации и аутентификации.

Используя Clerk, пользователи могут:

1 Регистрироваться и входить в систему с помощью электронной почты и пароля, а также через учетные записи Discord и Google.

2 Создавать организации и управлять ими, приглашая других пользователей присоединиться к ним.

3 Редактировать свой профиль и управлять своими настройками, включая изменение пароля.

4 Отслеживать активные сеансы и управлять ими, например, выходить из всех сеансов одновременно.

Благодаря этому функционалу пользователи могут безопасно и удобно взаимодействовать с приложением, а администраторы могут эффективно управлять доступом и настройками пользователей.

Для того, чтобы изначально настроить проект под использование convex необходимо иметь установленный node, после чего написать в терминале:

```
npm install convex
npx convex dev
```

После чего необходимо выбрать «a new project» и задать имя проекту, далее проект должен появится в панели управления Convex, если имеются заданные переменные окружения в виде ключей для Convex. Также необходимо установить Clerk – npm install @clerk/nextjs, npm install @clerk/clerk-react.

Далее необходимо настроить authMiddleware(). Помощник authMiddleware() интегрирует аутентификацию Clerk в веб-приложение Next.js с помощью промежуточного ПО. authMiddleware() совместим с маршрутизаторами App и Pages.

Для этого необходимо создать файл middleware.ts. Его содержимое должно быть следующим:

```
import { authMiddleware } from "@clerk/nextjs";

export default authMiddleware();
```

```
export const config = {
  matcher:          ["/((?!.+\\.[\\w]+$|_next).*)",          "/",
"/(api|trpc)(.*)"],
};
```

Далее необходимо создать в панели управления Clerk JWT Template для Convex, чтобы он мог хранить данные. Имя обязательно должно совпадать с «convex». После чего необходимо создать файл convex/auth.config.js, содержимое которого:

```
export default {
  providers: [
    {
      domain: "https://your-issuer-url.clerk.accounts.dev/",
      applicationID: "convex",
    },
  ]
};
```

Значение domain необходимо взять в панели управления Clerk, в строке Issuer, который примерно выглядит так:

```
https://XXXXXXXXXXXXX.clerk.accounts.dev
```

После завершения разработки веб-приложения необходимо провести деплой всех изменений. Для этого используется команда npx convex dev в терминале. Эта команда инициирует процесс деплоя и обеспечивает развертывание всех обновлений.

Для реализации функционала отслеживания изменений в режиме реального времени был использован инструмент Liveblocks.io. Для начала работы с ним, необходимо установить соответствующие зависимости с помощью команд:

```
npm install @liveblocks/client
npm install @liveblocks/react
```

Затем создается новое приложение с помощью команды:

```
npx create-liveblocks-app@latest --init --framework react
```

Данная команда создает начальную структуру приложения и интегрирует Liveblocks.io.

После установки зависимостей и создания приложения необходимо настроить конфигурационный файл liveblocks.config.ts, в котором указывается публичный ключ API для работы с функционалом Liveblocks.io.

Для безопасности эти ключи не должны быть явно указаны в коде приложения. Вместо этого, они хранятся в специальном файле .env.local, который обеспечивает безопасное хранение конфиденциальной информации.

Содержимое файла .env.local:

```
CONVEX_DEPLOYMENT=dev:XXXX
NEXT_PUBLIC_CONVEX_URL=https://XXXX.convex.cloud
NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY=pk_test_XXXX
CLERK_SECRET_KEY=sk_test_XXXX
NEXT_PUBLIC_LIVEBLOCKS_PUBLIC_KEY=pk_dev_XXXX
NEXT_PUBLIC_LIVEBLOCKS_SECRET_KEY=sk_dev_XXXX
```

В коде, где необходимы данные переменные, достаточно просто ссылаться следующим образом:

```
process.env.<имя_переменной>!
```

Файл .env.local содержит конфиденциальные данные, такие как ключи API и URL-адреса, необходимые для безопасной работы с различными сервисами веб-приложения. В этом файле хранятся следующие переменные:

1 CONVEX_DEPLOYMENT: указывает на развертку Convex для деплоя изменений веб-приложения.

2 NEXT_PUBLIC_CONVEX_URL: предоставляет URL-адрес для доступа к Convex, используемый в публичной части веб-приложения.

3 NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY: обеспечивает публичный ключ Clerk для взаимодействия с сервисом аутентификации и авторизации.

4 CLERK_SECRET_KEY: представляет собой секретный ключ Clerk для обеспечения безопасности операций аутентификации и авторизации.

5 NEXT_PUBLIC_LIVEBLOCKS_PUBLIC_KEY: содержит публичный ключ Liveblocks для работы с функционалом отслеживания изменений в реальном времени.

6 NEXT_PUBLIC_LIVEBLOCKS_SECRET_KEY: хранит секретный ключ Liveblocks, который обеспечивает безопасное взаимодействие с сервисом.

Эти переменные обеспечивают безопасную работу веб-приложения, так как они не явно указаны в коде и хранятся в защищенном файле .env.local, который не подлежит версионированию и не доступен извне.

Помимо этого, для удобства разворачиваемого приложения, можно использовать Docker. Dockerfile выглядит следующим образом:

```
FROM node AS build

WORKDIR /app

COPY package*.json ./

RUN npm install --legacy-peer-deps

COPY . .

RUN npm run build
RUN npm install convex

EXPOSE 3000

CMD ["npm", "run", "dev"]
```

Использование инструмента оркестрации Docker Compose даже для единственного сервиса представляет собой хорошую практику в разработке веб-приложений. Несмотря на то, что в текущей конфигурации присутствует только один сервис, Docker Compose обеспечивает удобство при запуске контейнеров и управлении ими. Docker Compose выглядит следующим образом:

```
version: '3.8'

services:
  app:
    build:
      context: .
    container_name: miro
    restart: on-failure
    ports:
      - "3000:3000"
    env_file:
      - .env.local
```

Теперь для того, чтобы запустить проект достаточно просто написать docker-compose up -d --build.

Первый запуск docker compose и сборка проекта, включая скачивание образа node заняло 79.9 секунд и размер образа составляет 2.84 G.

Теперь необходимо оптимизировать размер образа собираемого веб-приложения, для этого используем DevOps best practices (лучшие практики) при собирании образа:

1 Использование многоэтапной сборки (Multi-stage build): это позволяет уменьшить размер итогового образа, оставив только необходимые файлы и зависимости для работы приложения. В данном случае мы можем использовать один этап для сборки приложения и другой для запуска его.

2 Уменьшение числа слоев образа: чем меньше слоев в образе, тем меньше его размер. Можно объединить несколько команд COPY и RUN в одну для уменьшения числа промежуточных слоев.

3 Очистка кэша npm: после установки зависимостей рекомендуется очистить кэш npm для уменьшения размера образа. Это можно сделать, добавив npm cache clean --force в конце инструкции RUN для установки зависимостей.

Таким образом, итоговый Dockerfile выглядит следующим образом:

```
FROM node AS build

WORKDIR /app

COPY package*.json ./

RUN npm install --legacy-peer-deps

COPY . .

RUN npm run build && \
    npm cache clean --force

FROM node AS final

WORKDIR /app

COPY --from=build /app .

EXPOSE 3000

CMD ["npm", "run", "dev"]
```

Благодаря использованию best practices при сборке образа веб-сервиса, удалось добиться значительного снижения его размера до 2.01 ГБ. Это на 0.83 ГБ меньше, чем предыдущая версия образа. Уменьшение размера веб-сервиса

способствует оптимизации использования ресурсов и улучшении производительности при его развертывании и запуске в облачной среде.

4.2 Реализация инфраструктуры в виде кода

5 ОЦЕНКА КОЛИЧЕСТВЕННЫХ ПОКАЗАТЕЛЕЙ ФУНКЦИОНИРОВАНИЯ ПРОГРАММНОГО СРЕДСТВА

5.1 Оценка временных показателей программного средства

Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text.

5.2 Оценка ресурсных показателей программного средства

Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text.

5.3 Оценка показателей надёжности программного средства

Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text. Some text.

ЗАКЛЮЧЕНИЕ

Text.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

[1] Top 10 Cloud Provider Comparison 2023 [Электронный ресурс]. – Режим доступа : https://dev.to/dkechag/cloud-vm-performance-value-comparison-2023-perl-more-1kpp. – Дата доступа : 27.03.2024.

[2] 11 Top Cloud Service Providers Globally In 2024 [Электронный ресурс]. – Режим доступа : https://www.cloudzero.com/blog/cloud-service-providers/. – Дата доступа : 27.03.2024.

[3] Top 10 AWS Services for Data Engineering Projects [Электронный ресурс]. – Режим доступа : https://www.projectpro.io/article/aws-services-for-data-engineering/644. – Дата доступа : 27.03.2024.

[4] What is Azure – Microsoft Cloud Services | Microsoft Azure [Электронный ресурс]. – Режим доступа : https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure. – Дата доступа : 27.03.2024.

[5] DevOps – Alibaba Cloud [Электронный ресурс]. – Режим доступа : https://www.alibabacloud.com/ru/solutions/container/devops?_p_lc=1. – Дата доступа : 27.03.2024.

[6] Alibaba Cloud DevOps Pipeline (Flow): Enterprise Continuous Delivery Tool [Электронный ресурс]. – Режим доступа : https://www.alibabacloud.com/en/product/apsara-deveops/flow?_p_lc=1. – Дата доступа : 28.03.2024.

[7] DevOps Capability Improvement Model – Alibaba DevOps Practice Part 26 [Электронный ресурс]. – Режим доступа : https://www.alibabacloud.com/blog/devops-capability-improvement-model---alibaba-devops-practice-part-26_598658. – Дата доступа : 28.09.2024.

[8] SRE vs DevOps: что делают и чем отличаются специалисты | Yandex Cloud [Электронный ресурс]. – Режим доступа : https://yandex.cloud/ru/blog/posts/2023/01/devops-and-sre. – Дата доступа : 28.03.2024.

[9] Сам себе DevOps: как разобраться с доступами в Yandex Cloud / Хабр [Электронный ресурс]. – Режим доступа : https://habr.com/ru/companies/yandex_cloud_and_infra/articles/760252/. – Дата доступа : 28.03.2024.

[10] Yandex Cloud Services [Электронный ресурс]. – Режим доступа : https://yandex.cloud/en/docs/overview/concepts/services. – Дата доступа : 29.03.2024.

[11] IBM Cloud DevOps [Электронный ресурс]. – Режим доступа : https://cloud.ibm.com/docs/ContinuousDelivery?topic=ContinuousDelivery-devops_intro. – Дата доступа : 29.03.2024.

[12] What is Terraform | Terraform | HashiCorp Developer [Электронный ресурс]. – Режим доступа : https://developer.hashicorp.com/terraform/intro. – Дата доступа : 01.04.2024.

[13] Логически изолированное виртуальное частное облако – Цены на Amazon VPC – Amazon Web Services [Электронный ресурс]. – Режим доступа : https://aws.amazon.com/ru/vpc/features/. – Дата доступа : 02.04.2024.

[14] What is Amazon EC2? – Amazon Elastic Compute Cloud [Электронный ресурс]. – Режим доступа : https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html. – Дата доступа : 02.04.2024.

[15] Container Registry – Amazon Elastic Container Registry (Amazon ECR) – AWS [Электронный ресурс]. – Режим доступа : https://aws.amazon.com/ecr/?nc1=h_ls. – Дата доступа : 02.04.2024.

[16] Fully Managed Container Solution – Amazon Elastic Container Service (Amazon ECS) – Amazon Web Services [Электронный ресурс]. – Режим доступа : https://aws.amazon.com/ecs/?nc1=h_ls. – Дата доступа : 03.04.2024.

[17] APM Tool – Amazon CloudWatch – AWS [Электронный ресурс]. – Режим доступа : https://aws.amazon.com/cloudwatch/. – Дата доступа : 03.04.2024.

[18] Docker overview | Docker Docs [Электронный ресурс]. – Режим доступа : https://docs.docker.com/get-started/overview/. – Дата доступа : 03.04.2024.

[19] What's the Difference Between CI/CD and DevOps? [Электронный ресурс]. – Режим доступа : https://www.navisite.com/blog/insights/ci-cd-vs-devops/. – Дата доступа : 04.04.2024.