

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:

(подпись руководителя)
М.П. К.С. Улезко

Студент группы 013801

(подпись студента) К.Г. Хоменок
Руководитель практики от БГУИР
Шнейдеров Е.Н. – канд.техн.наук,
доцент

Минск 2024

СОДЕРЖАНИЕ

Введение.....	3
1 План-проспект дипломного проекта.....	4
2 Анализ исходных данных и постановка задач на дипломное проектирование	6
2.1 Анализ исходных данных к дипломному проекту	6
2.2 Обзор существующих облачных провайдеров по теме дипломного проекта.....	8
2.3 Обоснование и описание выбора облачного провайдера	16
2.4. Постановка задач на дипломное проектирование	18
3 Описание и проектирование облачной инфраструктуры	20
3.1 Terraform «инфраструктура как код»	20
3.2 Описание и обоснование используемых распределенных веб-сервисов.....	23
3.3 Контейнеризация и оркестрация с помощью Docker и Docker Compose.....	37
3.4 Описание и обоснование использования CI/CD	42
3.5 Проектирование облачной инфраструктуры.....	44
4 Практическая реализация облачной инфраструктуры для веб-сервиса.....	47
4.1 Обзор разворачиваемого веб-сервиса и используемых библиотек.....	47
5 Оценка количественных показателей функционирования программного средства	48
5.1 Оценка временных показателей программного средства	48
5.2 Оценка ресурсных показателей программного средства.....	48
5.3 Оценка показателей надёжности программного средства.....	48
Заключение	49
Список используемых источников.....	50

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 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 разработанной системы.

Объектом данного дипломного проекта является развертывание распределенного веб-сервиса на платформе Amazon Web Services (AWS). Предметом исследования являются DevOps-технологии и практики, включая Continuous Integration (CI) и Continuous Deployment (CD), а также использование инструмента инфраструктуры как кода Terraform для автоматизации процессов развертывания и управления инфраструктурой сервиса.

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	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 разворачиваемому веб-сервису; анализ существующих облачных провайдеров по теме дипломного проектирования; обоснование и описание выбора облачного провайдера; постановка задач на дипломное проектирование.	27.03.2024 – 31.03.2024
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	Обзор разворачиваемого веб-сервиса, реализация инфраструктуры в виде кода для облачного окружения, настройка непрерывной интеграции и доставки (CI/CD), описание технологий разворачивания распределенного Web-сервиса с использованием Terraform	09.04.2024 – 15.04.2024

Продолжение таблицы 1.1

Этап	Описание	Сроки выполнения
Оценка количественных показателей функционирования разворачиваемого веб-сервиса	Оценка временных показателей; оценка ресурсных показателей.	22.04.2024 – 04.05.2024
Разработка графического материала	Выполнение чертежа «IDEF0 диаграмма декомпозиции», выполнение плаката «UML диаграмма развертывания веб-сервиса», выполнение плаката «Структура манифеста Terraform», выполнение плаката «Схема инфраструктуры AWS», выполнение плаката «Графический интерфейс веб-сервиса», выполнение чертежа «Схема алгоритма развертывания инфраструктуры»	16.04.2024 – 14.05.2024
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

Анализ исходных данных к дипломному проекту представляет собой фундаментальный этап в разработке проекта, направленного на создание и поддержку распределенных веб-сервисов для платформы Amazon Web Services (AWS) с использованием DevOps технологий и инструментов автоматизации.

В соответствии с утвержденным приказом университета от 19.03.2024 № 595-с, тема проекта определена как «DevOps технологии поддержки распределенных Web сервисов для AWS с использованием Terraform». Этот проект предполагает разработку инфраструктуры и развертывания веб-сервиса для поддержки распределенных веб-сервисов на платформе AWS с использованием современных методов DevOps.

Исходные данные к дипломному проекту описывают систему распределенных веб-сервисов, предназначенных для платформы Amazon Web Services (AWS). Цель данной системы заключается в обеспечении поддержки распределенных веб-сервисов на платформе AWS с применением DevOps технологий и инструментов автоматизации, а также в оптимизации процессов развертывания и масштабирования.

В соответствии с функциональными требованиями, веб-сервис должен предоставлять возможность регистрации и аутентификации пользователей, наличие личного кабинета, реляционной базы данных, логирования действий пользователей и поддержки нескольких ролей. Инфраструктурная часть включает использование облачных виртуальных серверов (AWS EC2), облачных сетевых функций, Terraform для реализации «инфраструктуры как код», Docker и Docker Compose для контейнеризации приложений, а также применение облачных технологий контейнеризации и CI/CD.

Требования к графическому интерфейсу предполагают его соответствие принципам инженерного дизайна и современным подходам к проектированию. Программное окружение описывается как Amazon Web Services (VPC, Route Table, Internet Gateway, Public Subnet, Security Group, EC2, RDS, S3, Elastic IP, IAM Policy, IAM Role, CloudWatch), Terraform, GitHub Actions. Все подключаемые библиотеки должны иметь некоммерческую лицензию.

Проектирование системы должно соответствовать документам, включая общие требования к дипломным проектам, стандарты ISO/IEC по качеству

систем и программного обеспечения, а также нормативные документы по разработке программного обеспечения и созданию документации пользователя.

Исходные данные содержат подробное описание системы, ее назначение, требования к функциональности, графическому интерфейсу, программному окружению, а также указания по соответствующим стандартам и нормативам, которые должны учитываться при проектировании и разработке данного проекта.

Для анализа исходных данных к дипломному проекту также необходимо изучить особенности распределенных веб-сервисов Amazon Web Services (AWS). Распределенные веб-сервисы являются ключевым элементом сетевых приложений, способствующим эффективной обработке запросов от множества пользователей. Однако, для оптимальной реализации таких сервисов необходимо учитывать ряд особенностей, характерных для инфраструктуры облачных вычислений, таких как AWS.

Первоначально, эффективное проектирование и развертывание распределенных веб-сервисов на AWS требует глубокого понимания архитектурных принципов, присущих облачным вычислениям. Это включает в себя управление ресурсами, автомасштабирование, отказоустойчивость и обеспечение безопасности данных.

Далее, адаптация инфраструктуры распределенных веб-сервисов под AWS включает в себя использование соответствующих сервисов и инструментов, предоставляемых платформой. Например, для обеспечения высокой доступности и отказоустойчивости можно использовать сервисы, такие как Amazon Elastic Compute Cloud (EC2) для развертывания веб-серверов в различных регионах, а Amazon Relational Database Service (RDS) для хранения данных.

Наконец, обеспечение безопасности является критическим аспектом при развертывании распределенных веб-сервисов на AWS. Это включает в себя использование средств аутентификации, авторизации, а также механизмов шифрования данных, предоставляемых AWS Identity and Access Management (IAM).

Изучение особенностей распределенных веб-сервисов для AWS требует комплексного подхода, объединяющего понимание принципов облачных вычислений с практическим применением соответствующих сервисов и инструментов, предоставляемых платформой AWS.

Функциональные требования к разворачиваемому веб-сервису, ориентированному на платформу WordPress, включают следующие аспекты:

1 Регистрация и авторизация пользователей: веб-сервис должен обеспечивать возможность регистрации новых пользователей и

аутентификации уже зарегистрированных. Это включает в себя формы для ввода учетных данных пользователей, проверку их достоверности и управление процессом аутентификации.

2 Наличие личного кабинета: веб-сервис должен предоставлять зарегистрированным пользователям доступ к личному кабинету, где они могут просматривать и управлять своими персональными данными, настройками профиля и прочей информацией, связанной с их аккаунтом.

3 Наличие реляционной базы данных: для хранения информации о пользователях, их профилях, а также других сопутствующих данных (например, контента сайта) необходимо использовать реляционную базу данных. Это обеспечит структурированное хранение и эффективное управление данными.

4 Наличие логирования действий пользователей: веб-сервис должен осуществлять запись (логирование) действий пользователей, таких как вход в систему, изменение данных профиля, публикация контента и другие события, имеющие значение для безопасности и аналитики.

5 Наличие нескольких ролей: веб-сервис должен поддерживать механизм управления доступом на основе ролей пользователей. Это означает, что различным категориям пользователей должны быть назначены соответствующие роли с определенными правами доступа к функциональности сайта.

Проанализировав исходные данные к проекту, можно сделать вывод о необходимости разработки и поддержки распределенных веб-сервисов для AWS с использованием DevOps технологий. Требования к функциональности включают регистрацию и аутентификацию пользователей, наличие личного кабинета, реляционной базы данных, логирования действий пользователей и управления ролями. Эффективная реализация проекта требует учета особенностей облачных вычислений, выбора соответствующих сервисов AWS и соблюдения современных методов решения развертывания веб-сервиса.

2.2 Обзор существующих облачных провайдеров по теме дипломного проекта

Анализ ключевых облачных провайдеров позволяет выявить характеристики, предлагаемые услуги и инструменты, что способствует принятию обоснованного решения при выборе провайдера для реализации проектов.

Среди ведущих облачных провайдеров выделяются Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), Alibaba Cloud и Oracle Cloud. Каждый из них обладает уникальными особенностями,

предлагая широкий спектр сервисов для обеспечения гибкости, масштабируемости и надежности в различных проектах (рисунок 2.1).



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 следует более детальное рассмотрение каждого облачного провайдера и его преимуществ в контексте дипломного проекта, ориентированного на поддержку распределенных веб-сервисов с использованием DevOps технологий.

2.2.1 Amazon Web Services как облачный провайдер

Первым облачным провайдером, который заслуживает внимания, является Amazon Web Services (AWS) – гигант в сфере облачных вычислений и один из лидеров среди топ-5 облачных провайдеров мира (рисунок 2.2).



Рисунок 2.2 – Облачный провайдер AWS [3]

AWS предлагает широкий спектр облачных услуг и инструментов, включая те, которые значительно облегчают применение методологии DevOps:

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.

Amazon Web Services (AWS) была запущена в 2006 году и стала одним из ведущих облачных провайдеров в мире. Она предоставляет 105 зон доступности в 33 географических регионах с планами на расширение в 18 новых зон доступности и 6 новых регионов. Это обеспечивает высокую доступность и надежность для бизнес-приложений.

AWS предлагает широкий спектр сервисов (более 210). С долей рынка облачных услуг в 33% (по состоянию 2021 года), AWS является лидером в этой

области, что делает предпочтительным выбором AWS для многих организаций. Amazon EC2 предоставляет мощные виртуальные машины с возможностью масштабирования, а Amazon VPC позволяет создавать изолированные сети в облаке [4].

Среди клиентов AWS такие крупные компании, как Netflix, BMW и Samsung. Ценовая политика AWS включает гибкую ценовую модель, включая оплату за час, резервированные экземпляры и бесплатный уровень услуг.

Чистый объем продаж Amazon Web Services (AWS) вырос с примерно 8 миллиардов долларов в 2015 году до более чем 17 миллиардов долларов в 2017 году, достиг 35 миллиардов долларов к 2019 году и в настоящее время составляет 96,8 миллиарда долларов в год. Этот ошеломляющий рост был обусловлен расширением AWS с 32 зон доступности в 2015 году до 52 зон доступности в 2017 году и до 105 зон доступности в настоящее время. Параллельно компания запустила тысячи новых сервисов AWS, что также способствовало ее росту [6].

С учетом высокой производительности, масштабируемости и безопасности AWS остается одним из наиболее популярных облачных провайдеров среди разработчиков и инженеров по всему миру. Все эти факторы делают AWS предпочтительным выбором для различных типов проектов, включая дипломные работы, ориентированные на разработку и поддержку распределенных веб-сервисов. Возможность использования инфраструктуры как кода, широкий спектр сервисов DevOps и высокие стандарты безопасности делают AWS привлекательным решением для разработчиков, стремящихся к успешной 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». Логотип представлен на рисунке 2.3 [5].



Рисунок 2.3 – Облачный провайдер Microsoft Azure

Microsoft Azure предоставляет разнообразные услуги для обеспечения вычислительных ресурсов, хранилища данных, сетевых ресурсов, управления ресурсами и мониторинга, инструментов разработки, интеграции с открытыми источниками и обеспечения безопасности. В числе предоставляемых услуг – виртуальные машины различных конфигураций, хранилище данных Azure Blob Storage, Azure SQL Database, Azure Cosmos DB, виртуальные сети, службы балансировки нагрузки, а также инструменты разработки, такие как Azure DevOps Services, Azure DevTest Labs и Azure Repos. Безопасность также является одним из приоритетов Azure, что подтверждается инвестициями в сумме более 1 миллиарда долларов США за последний год. Обновления и расширения платформы продолжаются, включая разработку новых инструментов и услуг для облегчения процесса разработки, развертывания и управления веб-сервисами.

Облачный провайдер Microsoft Azure был запущен в 2010 году и стал одним из ведущих в мире. Azure предоставляет доступ к более чем 64 регионам по всему миру и более чем 126 зонам доступности, обеспечивая высокую доступность и надежность для бизнес-приложений. Клиенты могут выбирать из различных географических локаций в зависимости от своих потребностей.

Azure предлагает более 200 облачных сервисов, включая вычисления (виртуальные машины, контейнеры и серверы приложений), хранение данных (Azure Blob Storage, Azure SQL Database, Azure Cosmos DB), искусственный интеллект (Azure Machine Learning, Azure Cognitive Services), сетевые решения (виртуальные сети, балансировщики нагрузки, VPN-шлюзы), аналитику и большие данные (Azure Data Lake, Azure HDInsight).

С долей рынка облачных услуг в 20%, Azure является важным игроком на рынке, набирая популярность среди предприятий и разработчиков. В Azure используются виртуальные машины, аналогичные Amazon EC2, для развертывания приложений и обработки данных. Azure Virtual Network (VNet) позволяет создавать изолированные сети в облаке и управлять трафиком.

Среди клиентов Azure такие крупные компании, как Adobe, BMW и General Electric. Ценовая политика Azure основана на оплате использованных ресурсов за час.

По всему миру расположено более 300 физических центров обработки данных Microsoft Azure, в которых размещены компьютерные серверы, каждый из которых оснащен независимым питанием, охлаждением и сетями. Компания соединяет инфраструктуру дата-центров более чем 175 000 миль волоконно-оптических линий связи в 140 странах.

Доходы Microsoft от интеллектуального облака, включающего в себя доходы от Azure, других облачных сервисов и серверных продуктов, в последнем квартале достигли 25,9 млрд долларов, что на 19 % больше, чем в прошлом году. Таким образом, в годовом исчислении доход Microsoft от Intelligent Cloud составляет 103,5 миллиарда долларов. Однако Microsoft не раскрывает информацию о доходах от Azure, что означает, что Azure – это лишь часть общего дохода от Intelligent Cloud.

2.2.3 Google Cloud Platform как облачный провайдер

Google Cloud Platform (GCP) представляет собой мощный облачный провайдер, который предоставляет разнообразные услуги и инструменты для компаний, занимающихся разработкой и поддержкой программного обеспечения в контексте DevOps. Одним из ключевых преимуществ GCP является его гибкая и масштабируемая вычислительная инфраструктура, которая позволяет эффективно управлять вычислительными ресурсами и разворачивать программное обеспечение в облаке с использованием виртуальных машин и контейнерных сред, таких как Kubernetes Engine. Логотип GCP представлен на рисунке 2.4.



Рисунок 2.4 – Google Cloud Platform

GCP предлагает широкий набор сервисов для хранения данных, включая Cloud Storage для объектов, Cloud SQL для реляционных баз данных и Firestore для NoSQL хранилищ, что обеспечивает разработчикам разнообразные решения для хранения и обработки данных.

Важным аспектом в области 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.

Сервисы, созданные Google, используют ту же облачную инфраструктуру, которую компания использует для своих внутренних продуктов, таких как Gmail, Google Search, Google Photos и YouTube.

Год основания Google Cloud Platform – 2008 год. С тех пор GCP стал одним из ведущих провайдеров облачных услуг в мире. На сегодняшний день Google Cloud насчитывает 40 регионов и 121 зону доступности. Эти регионы и зоны доступности расположены в США, Северной и Южной Америке, Европе, Азиатско-Тихоокеанском регионе, а также на Ближнем Востоке и в Африке. Клиенты могут выбирать из различных географических локаций в соответствии с их потребностями.

Google Cloud Platform предлагает широкий спектр сервисов, включая вычисления (виртуальные машины, контейнеры, серверы приложений), хранилище данных (Google Cloud Storage, Google Cloud SQL), искусственный интеллект (Google AI Platform, TensorFlow), сеть (Virtual Private Cloud, Google Cloud Load Balancing).

Доля рынка GCP в облачных услугах составляет 9% и продолжает расти. Это делает его значимым игроком на рынке, привлекательным для разработчиков и организаций. В GCP используются виртуальные машины, предоставляющие масштабируемые ресурсы для развертывания приложений и обработки данных. Virtual Private Cloud (VPC) в GCP позволяет создавать изолированные сети и управлять трафиком.

Среди клиентов GCP такие компании, как Snap Inc., Spotify и HSBC. Ценообразование в Google Cloud Platform зависит от использованных ресурсов и предлагает гибкие опции оплаты за использованные ресурсы.

Подразделение Google Cloud компании Alphabet Inc получает доход за счет платы, взимаемой за инфраструктуру, платформу и другие услуги. В последнем квартале выручка Google Cloud составила 9,2 млрд долларов, что на 26 % больше, чем в прошлом году. Таким образом, в годовом исчислении выручка Google Cloud составляет 37 миллиардов долларов.

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.

Подразделение Alibaba Group, занимающееся облачными вычислениями и известное как Alibaba Cloud, является четвертым по величине поставщиком облачных услуг в мире, основным поставщиком облачных услуг в Азиатско-Тихоокеанском регионе и крупнейшим поставщиком облачных услуг в Китае. Через Alibaba Cloud компания предлагает облачные услуги, включая эластичные вычисления, базы данных, хранилища, сетевую виртуализацию, крупномасштабные вычисления, безопасность, управление и услуги приложений, аналитику больших данных и машинное обучение.

В настоящее время Alibaba Cloud работает в 30 регионах и 89 зонах доступности. В континентальном Китае Alibaba является доминирующим поставщиком облачных услуг, имея 15 регионов по всей стране. За пределами материкового Китая Alibaba Cloud работает в США, Европе, Азиатско-Тихоокеанском регионе и на Ближнем Востоке.

Alibaba Group в основном получает доходы от облачных вычислений от корпоративных клиентов в зависимости от продолжительности и использования их услуг. Выручка подразделения Cloud Intelligence Group, в которое входит Alibaba Cloud, за последний квартал составила 3,95 млрд долларов (28 066 млн юаней), что на 3 % больше, чем в прошлом году. Таким образом, в годовом исчислении выручка Alibaba Cloud в настоящее время составляет 15,8 млрд долларов.

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 [7][8].

2.2.5 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.

Облачные сервисы корпорации Oracle включают в себя Oracle Cloud Software-as-a-Service (SaaS) и Oracle Cloud Infrastructure (OCI). В рамках OCI компания является поставщиком облачных услуг, предоставляя инфраструктурные технологии как услугу, включая вычисления, хранение и сетевые сервисы.

В настоящее время Oracle Cloud имеет 48 регионов и 58 зон доступности. Эти регионы и зоны доступности расположены в США, Канаде, Европе, на Ближнем Востоке и в Африке (ЕМЕА), Латинской Америке и Азиатско-Тихоокеанском регионе. Кроме того, Oracle Cloud предлагает правительственные облачные регионы для правительства США, Министерства обороны США (DoD) и правительства Великобритании, а также два суверенных региона для клиентов из стран Европейского союза (ЕС).

Oracle Cloud Infrastructure (OCI) обычно взимает предоплату, которая постепенно снижается по мере потребления услуг OCI клиентом в течение определенного периода времени. В последнем квартале выручка Oracle Cloud IaaS достигла 1,8 млрд долларов, увеличившись на 49% по сравнению с предыдущим годом. Таким образом, в годовом исчислении выручка Oracle Cloud составляет 7,2 млрд долларов.

2.3 Обоснование и описание выбора облачного провайдера

Для обоснования выбора облачного провайдера необходимо провести анализ и сравнительную оценку нескольких провайдеров облачных услуг, включая AWS, в контексте требований и целей дипломного проекта. Этот анализ должен включать в себя составление сравнительных таблиц, в которых будут отображены ключевые параметры и характеристики каждого провайдера.

Сравнительные таблицы позволят объективно оценить преимущества и недостатки каждого провайдера и определить, какой из них лучше соответствует требованиям и целям дипломного проекта. Кроме того, при анализе необходимо учитывать факторы, специфичные для проекта, такие как поддержка DevOps методологии, совместимость с выбранными инструментами разработки и управления инфраструктурой, а также поддержка распределенных веб-сервисов.

После проведения анализа и сравнительной оценки провайдеров необходимо будет принять обоснованное решение о выборе оптимального облачного провайдера для реализации дипломного проекта.

В таблице 2.1 представлена сравнительная таблица облачных провайдеров (AWS, Azure, GCP).

Таблица 2.1 – Сравнительная таблица облачных провайдеров (AWS, Azure, GCP)

	AWS	Azure	GCP
Год запуска	2004	2010	2008
Зоны доступности	105	126	121
Регионы	33	64	40
Сервисы	210+	200+	100+
Облачный рынок	33%	21%	8%
Вычислительный движок	EC2 (Elastic Compute Search)	Virtual Machine	Compute Engine
Сеть	VPC (Virtual Private Cloud)	VNET (Virtual Network)	Cloud Virtual Network
Ценовая политика	по часам	по минутам	по минутам

В таблице 2.2 представлен сравнительный анализ облачных провайдеров (Alibaba Cloud, Oracle Cloud).

Таблица 2.2 – Сравнительная таблица облачных провайдеров (Alibaba Cloud и Oracle Cloud)

	Alibaba Cloud	Oracle Cloud
Год запуска	2009	2016
Зоны доступности	89	58
Регионы	30	48
Сервисы	210+	200+
Облачный рынок	7.7%	2.1%
Вычислительный движок	Alibaba ECS	Compute
Сеть	VPC (Virtual Private Cloud)	VCN (Virtual Cloud Network)
Ценовая политика	постоянная или по мере использования	постоянная или по мере использования

После анализа и сравнительной оценки нескольких ведущих облачных провайдеров, таких как AWS, Azure, GCP, Alibaba Cloud и Oracle Cloud, было

принято обоснованное решение о выборе AWS в качестве оптимального варианта.

Одним из основных факторов, подтверждающих выбор AWS, является его лидерство на мировом рынке облачных провайдеров. По данным сравнительной таблицы, AWS занимает второе место по количеству зон доступности, регионов и предоставляемых сервисов. Но приоритетом является то, что AWS имеет значительно большую долю рынка в сравнении с конкурентами, что свидетельствует о высокой степени доверия со стороны пользователей.

Кроме того, AWS предлагает широкий спектр вычислительных движков и сетевых решений, таких как EC2 и VPC, которые хорошо подходят для реализации распределенных веб-сервисов. Ценовая политика AWS, основанная на оплате за использование по часам, также является привлекательным фактором для студентов и начинающих специалистов, позволяя экономить ресурсы при разработке и тестировании проекта.

Важным аспектом выбора AWS является его простота в использовании и доступность документации. Интуитивно понятный интерфейс платформы и обширная база знаний делают процесс разработки и управления инфраструктурой более эффективным и прозрачным.

Таким образом, на основании представленного сравнительного анализа и учета основных факторов, обоснованным решением является выбор облачного провайдера AWS для реализации дипломного проекта. 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

Постановка задач на дипломное проектирование предполагает развертывание веб-сервиса на инфраструктуре облачного провайдера Amazon Web Services (AWS) с использованием распределенных веб-сервисов и инструментов DevOps. Для автоматизации управления инфраструктурой и ее конфигурацией необходимо применять Terraform, – инструмент инфраструктурного кодирования.

Перед началом развертывания веб-сервиса и применения DevOps-технологий необходимо провести анализ задания к дипломному проекту. Данный этап играет ключевую роль в формировании требований и ожиданий, предъявляемых к разрабатываемому дипломному проекту. Анализ задания позволяет выявить основные задачи и цели проекта, определить требования к функциональности и качеству разрабатываемого веб-сервиса, а также установить рамки и ограничения, с которыми необходимо работать. В типовом задании к дипломному проекту прописываются основные этапы работы, ожидаемые результаты и критерии оценки.

Графические материалы и пояснительная записка к дипломному проекту оформляются согласно установленным стандартам и требованиям. Используются соответствующие ГОСТы, регламентирующие оформление документации и графических материалов.

Проведение анализа документации по облачному провайдеру AWS, Terraform и реализации CI/CD позволит ознакомиться с теоретическими и практическими аспектами, необходимыми для успешной реализации дипломного проекта. Это включает изучение основных концепций и функциональных возможностей данных технологий, а также методов их применения в практике.

Следующим этапом является реализация DevOps технологий поддержки распределенных веб-сервисов для AWS с использованием Terraform в соответствии с требованиями, определенными в типовом задании дипломного проекта. Это включает проектирование инфраструктуры, ее описание и последующий запуск (создание) для тестирования или деплоя, а также реализацию DevOps-практики CI/CD с использованием Terraform и GitHub Actions.

Написание пояснительной записки и разработка графического материала являются важными составляющими процесса дипломного проектирования. Пояснительная записка должна содержать описание всех этапов работы, принятых решений и полученных результатов, а также анализ выполненной работы. Графический материал должен наглядно иллюстрировать основные аспекты проекта и поддерживать текст пояснительной записки.

Завершающим этапом, подтверждающим успешное выполнения задания к дипломному проекту, является защита как преддипломной практики, так и самого дипломного проекта.

3 ОПИСАНИЕ И ПРОЕКТИРОВАНИЕ ОБЛАЧНОЙ ИНФРАСТРУКТУРЫ

3.1 Terraform «инфраструктура как код»

HashiCorp Terraform представляет собой инструмент инфраструктуры в формате кода, предназначенный для определения облачных и локальных ресурсов в читаемых человеком конфигурационных файлах. Эти файлы могут быть версионированы, повторно и совместно использоваться. Помимо этого, Terraform использует последовательный рабочий процесс для обеспечения и управления всей инфраструктурой на протяжении ее жизненного цикла. Возможности Terraform включают управление как низкоуровневыми компонентами, такими как вычислительные ресурсы, ресурсы хранения и сетевые ресурсы, так и 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.

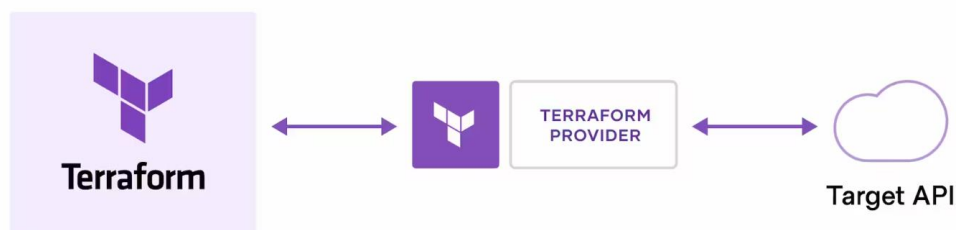


Рисунок 3.1 – Принцип работы Terraform [9]

В основе работы Terraform и принципа инфраструктуры как код лежит процесс, начинающийся с загрузки конфигурационных файлов Terraform. Эти файлы описывают желаемую инфраструктуру, включая ресурсы и их параметры. Этот этап представляет собой начальную точку, с которой Terraform начинает свою работу по созданию, изменению или удалению инфраструктурных ресурсов в соответствии с описанием, предоставленным в конфигурационных файлах. Ниже приведен алгоритм работы Terraform:

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 включает в себя сравнение текущего состояния инфраструктуры, которое сохранено локально в файлах состояния, с желаемым состоянием, описанным в конфигурации. В результате этого процесса определяются ресурсы, которые требуется добавить, изменить или удалить для достижения желаемого состояния. На основе этого сравнения Terraform генерирует план изменений, который содержит необходимые действия для синхронизации текущего и желаемого состояний инфраструктуры.

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.

NashiCorp и сообщество Terraform разработали значительное количество провайдеров для управления разнообразными типами ресурсов и сервисов. В реестре Terraform представлены все общедоступные провайдеры, включая Amazon Web Services (AWS), Azure, Google Cloud Platform (GCP), Kubernetes, Helm, GitHub, Splunk, Docker и многие другие.

Основной рабочий процесс Terraform включает три этапа:

1 Написание манифеста: на этом этапе определяются ресурсы, которые могут присутствовать у различных облачных провайдеров и сервисов. Например, необходимо создать конфигурацию для развертывания приложения на виртуальных машинах в сети виртуального частного облака (VPC) с группами безопасности и балансировщиком нагрузки.

2 Планирование (plan): Terraform генерирует план выполнения, который описывает инфраструктуру, предполагаемую для создания, обновления или уничтожения. Этот план формируется на основе существующего состояния инфраструктуры и конфигурации, предоставленной пользователем.

3 Применение (apply): после подтверждения пользователя Terraform выполняет предложенные операции в правильном порядке, учитывая все зависимости от ресурсов. Например, если происходит обновление свойств VPC и изменение количества виртуальных машин в этом VPC, Terraform создает VPC заново перед масштабированием виртуальных машин. Один из полезных функциональных возможностей Terraform заключается в возможности выполнения проверки после написания инфраструктуры в виде кода с помощью команды `terraform plan` в терминале. Эта команда, как правило, выявляет семантические ошибки, если таковые имеются, и выводит подробный план построения инфраструктуры, включая этапы выполнения (см. рисунок 3.2).

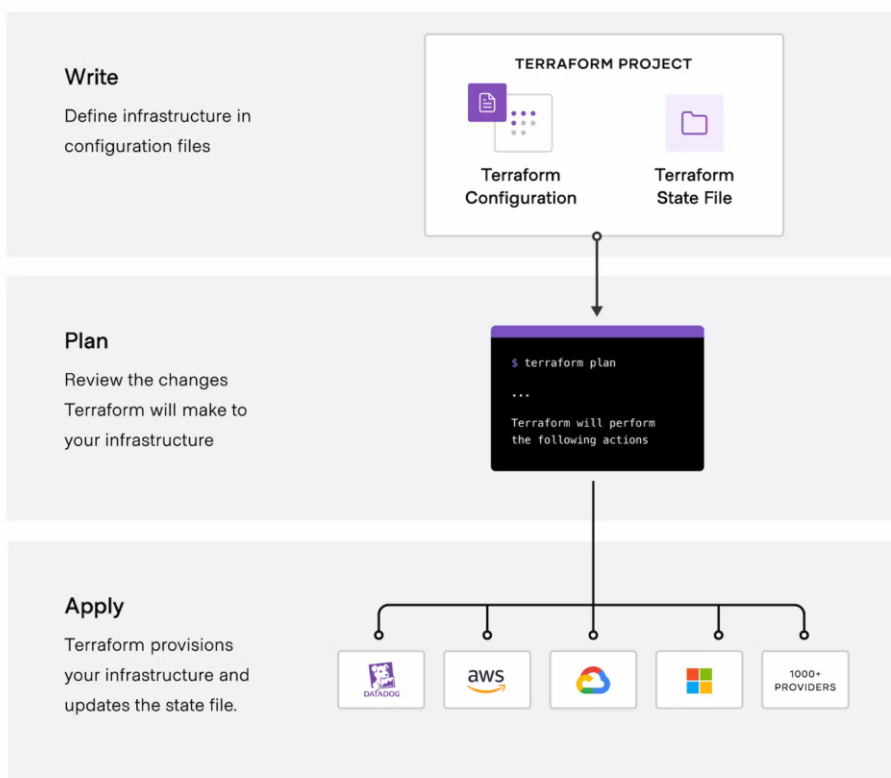


Рисунок 3.2 – Рабочий процесс Terraform [10]

Преимущества использования Terraform:

1 Управление любой инфраструктурой и провайдером: Terraform предоставляет доступ к реестру провайдеров для множества платформ и

сервисов, а также позволяет создавать собственные провайдеры. Использование неизменяемого подхода к инфраструктуре упрощает процесс обновления или модификации сервисов и инфраструктуры.

2 Отслеживание изменений инфраструктуры: Terraform генерирует план изменений и запрашивает подтверждение перед их применением. Файл состояния служит источником истины для среды, а Terraform использует его для определения изменений, необходимых для соответствия конфигурации.

3 Автоматизация изменений: Файлы конфигурации Terraform являются декларативными, что позволяет описывать конечное состояние инфраструктуры без необходимости написания пошаговых инструкций. Terraform эффективно управляет логикой создания и изменения ресурсов, строя граф зависимостей и параллельно обрабатывая независимые ресурсы.

4 Стандартизация конфигураций: Terraform поддерживает модули – это многократно используемые компоненты конфигурации, что позволяет экономить время на создание настраиваемых коллекций инфраструктуры.

5 Совместная работа: Terraform позволяет фиксировать манифесты в системе контроля версий и использовать Terraform Cloud или другие облачные провайдеры для эффективного управления рабочими процессами в командах. Terraform Cloud обеспечивает безопасный доступ к общему состоянию, контроль доступа на основе ролей и другие полезные функции. 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 кода», является важным элементом для того, чтобы эффективно и автоматизировано управлять инфраструктурой разворачиваемого проекта (веб-сервиса) в рамках данного дипломного проекта.

3.2 Описание и обоснование используемых распределенных веб-сервисов

Распределенные веб-сервисы имеют стратегическое значение в сфере построения облачной инфраструктуры, обеспечивая необходимую масштабируемость и надежность при предоставлении онлайн-услуг.

Данная глава направлена на анализ и обоснование выбора распределенных веб-сервисов, включая VPC, Route Table, Internet Gateway, Public Subnet, Security Group, RDS Instance, S3, Elastic IP, IAM Policy, IAM Role и CloudWatch.

В этом контексте следующим этапом является рассмотрение первого из распределенных веб-сервисов – Virtual Private Cloud (VPC).

3.2.1 VPC

Amazon Virtual Private Cloud (VPC) представляет собой сервис, предоставляющий возможность разворачивать ресурсы AWS в логически изолированной виртуальной сети, настраиваемой пользователем. Это обеспечивает полный контроль над окружением виртуальной сети, включая возможность определения собственного диапазона IP-адресов, создание подсетей, а также конфигурацию таблиц маршрутизации и сетевых шлюзов. Для большинства ресурсов в VPC доступны как IPv4, так и IPv6, что обеспечивает безопасный и удобный доступ к ресурсам и приложениям (рисунок 3.3).

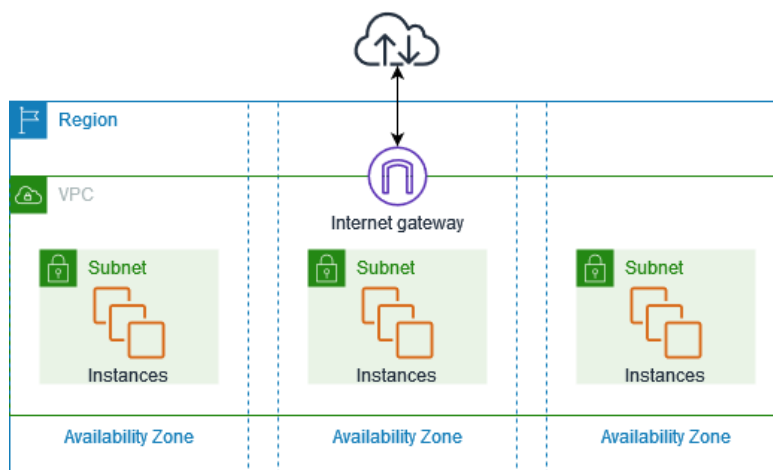


Рисунок 3.3 – Инфраструктура с использованием VPC [11]

В качестве одного из основополагающих сервисов AWS, Amazon VPC обеспечивает удобную настройку параметров сети VPC в соответствии с потребностями пользователя. Путем создания общедоступной подсети для веб-серверов с доступом к Интернету, а также частной подсети для серверных систем, таких как базы данных или серверы приложений, пользователи могут эффективно организовать свою инфраструктуру. Amazon VPC также предлагает использование многоуровневой системы безопасности, включающей в себя группы безопасности и сетевые списки контроля доступа. Эта система обеспечивает контроль доступа к виртуальным машинам Amazon Elastic Compute Cloud (Amazon EC2) в каждой подсети, повышая безопасность и управляемость сетевой инфраструктуры.

Помимо прочего, в рамках виртуальной частной сети (VPC) возможно настройка ряда сервисов, включая журналы потоков, IP Address Manager

(IPAM) для управления IP-адресами, входящую маршрутизацию, Network Access Analyzer для анализа доступа к сети, список контроля доступа к сети, сетевой менеджер, а также Reachability Analyzer - инструмент для анализа статических конфигураций. Кроме того, в рамках VPC предусмотрены группы безопасности и зеркалирование трафика.

Отображение VPC на рисунке 3.3 подчеркивает его значимость как главного и основополагающего распределенного веб-сервиса. VPC выступает в качестве центральной сети инфраструктуры, существенно важной для организации, так как находится выше только по уровню региона (включая доступные зоны и/или зоны доступности).

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.

Добавление VPC в инфраструктуру разворачиваемого веб-сервиса является ключевым шагом для создания надежной и эффективной облачной инфраструктуры.

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.

Основные концепции для таблицы маршрутов включают главную таблицу маршрутов, которая предоставляется автоматически с VPC и управляет маршрутизацией для всех подсетей, не связанных с другими таблицами маршрутов, а также пользовательскую таблицу маршрутов, создаваемую для конкретного VPC.

Каждый маршрут определяется понятиями «назначение» и «цель», где назначение указывает диапазон IP-адресов, куда должен быть направлен трафик (CIDR назначения), а цель определяет шлюз, сетевой интерфейс или соединение, через которые трафик будет направлен, например, интернет-шлюз.

Для связи таблиц маршрутов с подсетями, интернет-шлюзами или виртуальными частными шлюзами используется ассоциация таблиц

маршрутов, а таблица маршрутов подсети связывается непосредственно с определенной подсетью.

Локальный маршрут представляет собой маршрут по умолчанию для внутренней связи в пределах VPC, в то время как распространение маршрутов включает автоматическое добавление маршрутов для VPN-соединения в таблицы маршрутов подсетей при подключении виртуального частного шлюза к VPC.

Таблица маршрутов шлюза связана с интернет-шлюзом или виртуальным частным шлюзом, а ассоциация с границами используется для маршрутизации входящего трафика VPC на конкретное устройство.

Таблица маршрутов транзитного шлюза связана с транзитным шлюзом, тогда как таблица маршрутов локального шлюза связана с локальным шлюзом Outposts.

Использование распределенного веб-сервиса Route Table на AWS обосновано его ключевой ролью в управлении трафиком между различными сегментами виртуальной сети. Route Table предоставляет возможность определения маршрутов для сетевого трафика в зависимости от его назначения, что обеспечивает эффективную маршрутизацию внутри облачной инфраструктуры.

3.2.3 Internet Gateway

Интернет-шлюз представляет собой ключевой компонент горизонтально масштабируемой и высокодоступной виртуальной частной сети (VPC), обеспечивающий устойчивое соединение между VPC и Интернетом. С его помощью возможна передача трафика как по протоколу IPv4, так и IPv6, обеспечивая эффективное взаимодействие ресурсов как в публичных, так и в локальных сетях. Интернет-шлюз действует в качестве посредника для связи между ресурсами, обладающими публичными IP-адресами, и обеспечивает возможность инициирования соединений как с внешних ресурсов, так и из внутренней сети.

Специфическим функционалом интернет-шлюза является обеспечение соответствующих маршрутов в таблицах маршрутизации VPC для трафика, направляемого через Интернет. Кроме того, для обмена данными по протоколу IPv4, Internet Gateway выполняет функцию трансляции сетевых адресов (NAT), что способствует эффективному использованию ресурсов и обеспечивает безопасность сетевого трафика. Однако, в случае использования IPv6, такая функция не требуется, поскольку IPv6-адреса являются общедоступными и не требуют дополнительной трансляции.

Экземпляры EC2 в облаке AWS, а также другие ресурсы, размещенные в публичных подсетях, могут взаимодействовать с Интернетом благодаря наличию публичных IP-адресов и обработке трафика через интернет-шлюз. Этот механизм обеспечивает эффективное управление сетевым трафиком и обеспечивает надежное соединение как для облачных, так и для локальных сред.

Internet Gateway (IGW) в AWS играет ключевую роль в обеспечении доступа к интернету для виртуальных ресурсов внутри Amazon Virtual Private Cloud (VPC). Этот компонент инфраструктуры позволяет виртуальным серверам и сервисам в VPC обмениваться данными с внешним миром, обеспечивая доступ к внешним ресурсам, веб-сервисам и клиентам через интернет.

В рамках дипломного проекта «DevOps технологии поддержки распределенных Web сервисов для AWS с использованием Terraform» использование Internet Gateway обосновано необходимостью обеспечения связности и доступности веб-приложений и сервисов, развернутых в облаке AWS. IGW гарантирует высокую доступность внешнего доступа к веб-приложениям, обеспечивает масштабируемость и гибкость в управлении сетевым трафиком, а также интегрируется с другими сервисами AWS, что позволяет создавать комплексные сетевые конфигурации с учетом требований безопасности и производительности.

3.2.4 Public Subnet

Public Subnet в Amazon Virtual Private Cloud (VPC) играет важную роль в архитектуре облачных приложений, предоставляя среду для размещения публичных ресурсов, которые требуют доступности извне. Этот сегмент сети обеспечивает прямой доступ к интернету через Internet Gateway на AWS, что позволяет веб-серверам, API-шлюзам и другим публичным ресурсам взаимодействовать с внешними клиентами и сервисами.

Использование Public Subnet обосновано необходимостью предоставления доступа к веб-сервису извне, чтобы потенциальный пользователь мог зайти по адресу домена веб-сервиса и получить к нему доступ. Этот подход позволяет изолировать публичные ресурсы от приватных, повышая уровень безопасности и обеспечивая контроль доступа к важным данным и сервисам. Такая конфигурация позволяет эффективно управлять и обеспечивать безопасность веб-инфраструктуры в облаке AWS, что является ключевым аспектом современных распределенных веб-сервисов.

Это включает в себя обслуживание внешних запросов и обмен информацией с внешним миром. При этом изоляция публичных ресурсов от

приватных обеспечивает повышенный уровень безопасности и контроля доступа к важным данным и сервисам.

3.2.5 Private Subnet

Private Subnet в Amazon Virtual Private Cloud (VPC) представляет собой сегмент сети, который не имеет прямого доступа к интернету и используется для размещения приватных ресурсов, таких как базы данных Amazon RDS. Этот компонент обеспечивает изоляцию приватных ресурсов от внешнего интернета, обеспечивая дополнительный уровень безопасности для данных, хранимых и обрабатываемых в облаке AWS.

Выбор использования Private Subnet с Amazon RDS обоснован необходимостью ограничения доступа к базе данных из внешних сетей. Этот подход способствует повышению уровня безопасности данных, хранимых в базе данных, и предотвращает несанкционированный доступ извне. Подключение базы данных к Private Subnet ограничивает доступ к ней только из ресурсов, находящихся в той же сети, что и RDS, обеспечивая дополнительный слой защиты.

При развертывании веб-сервиса на экземпляре EC2, который находится в публичной подсети (Public Subnet), взаимодействие с базой данных RDS, находящейся в Private Subnet, ограничивается. Экземпляр EC2 предоставляет доступ пользователям извне к веб-сервису, поскольку обладает Internet Gateway и принимает внешний трафик. Однако, Private Subnet принимает трафик исключительно из локальной сети, в которой размещен EC2, и не предоставляет доступ извне. Таким образом, доступ к базе данных RDS возможен только из экземпляра EC2, что создает дополнительный уровень безопасности. Учитывая, что доступ к EC2 осуществляется через SSH-ключ, скомпрометировать который является нереальной задачей, обеспечивается высокий уровень безопасности для базы данных RDS.

3.2.6 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.

3.2.7 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.

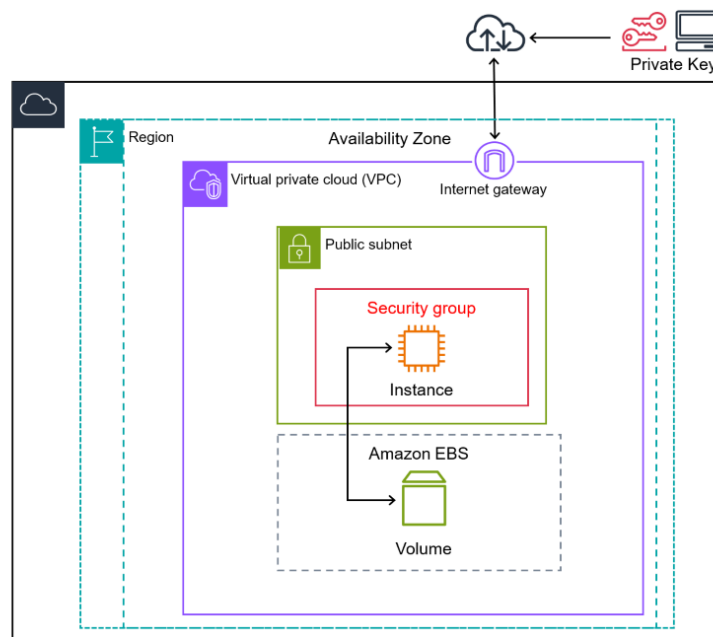


Рисунок 3.4 – Базовая архитектура экземпляра AWS EC2 [12]

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.

3.2.8 S3 Bucket

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).

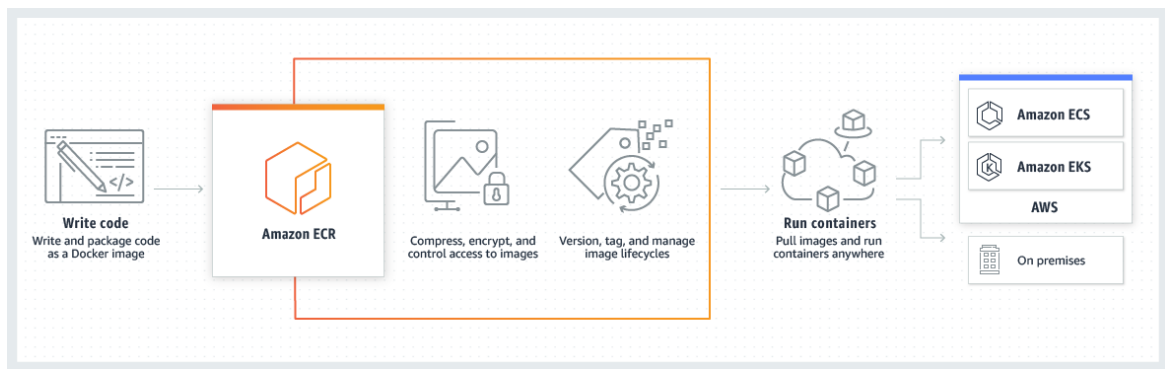


Рисунок 3.5 – Принцип работы AWS ECR [13]

ECS обеспечивает высокую доступность, масштабируемость и безопасность при развертывании приложений в контейнерах (рисунок 3.6).

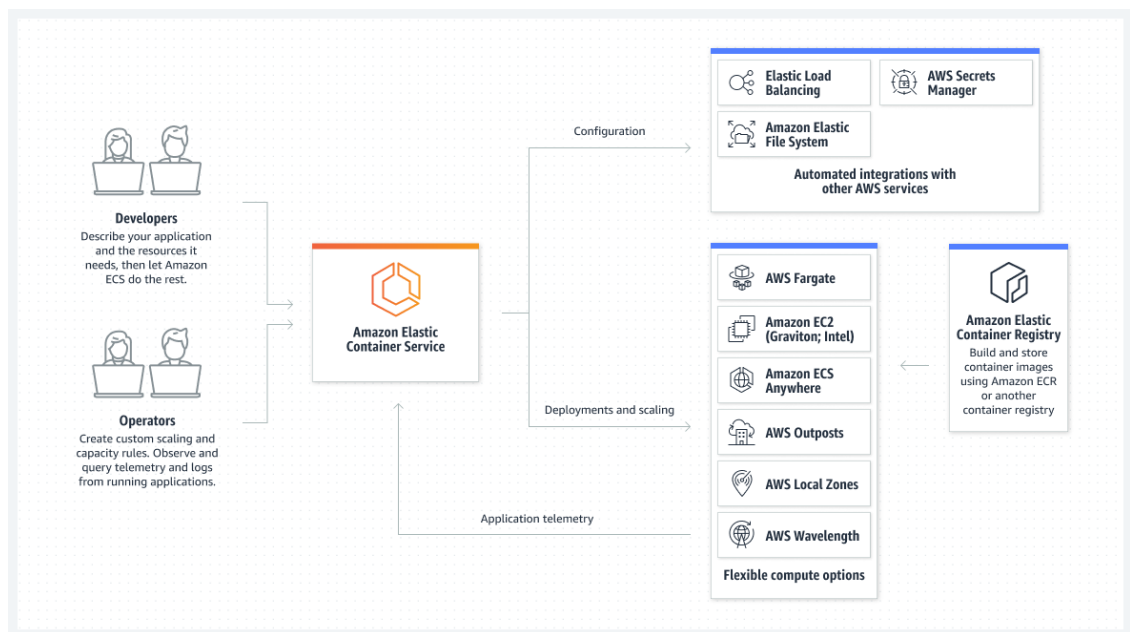


Рисунок 3.6 – Amazon ECS: управление приложениями с гибкими возможностями масштабирования и автоматической интеграцией AWS [14]

3.2.9 RDS

3.2.10 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-адресом, 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.

3.2.11 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.

3.2.12 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.

3.2.13 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).

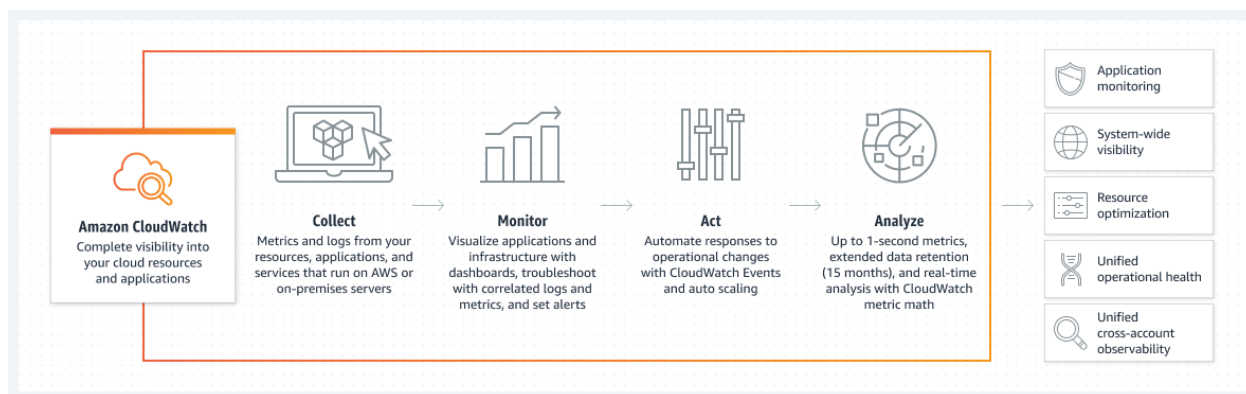


Рисунок 3.7 – Принцип работы Amazon CloudWatch [15]

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 получают образ, получают 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.

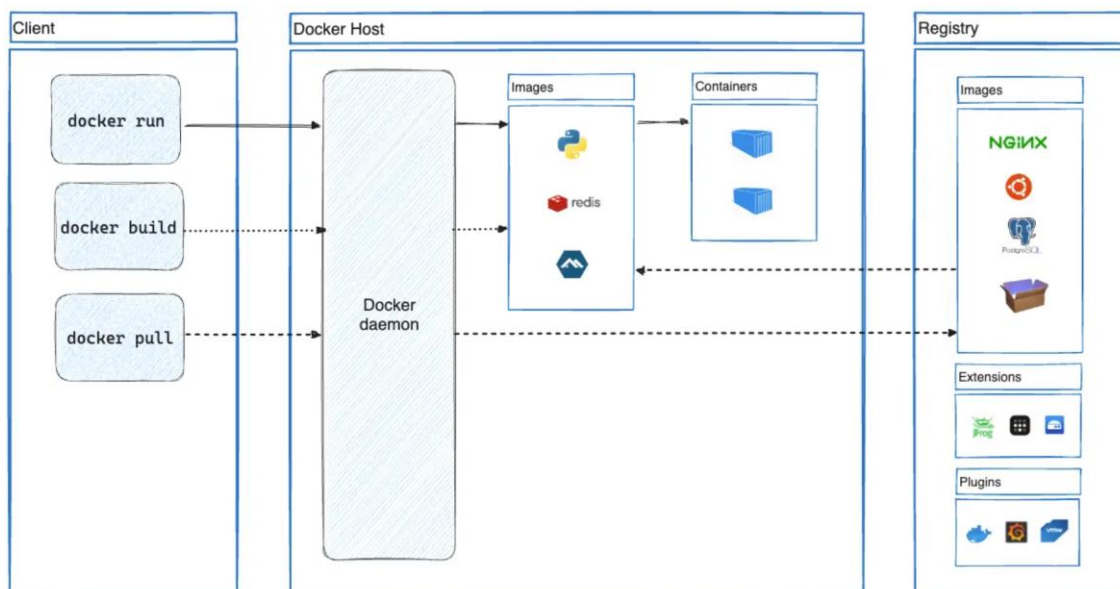


Рисунок 3.8 – Архитектура Docker [16]

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 пересобираение 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 поставка (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).

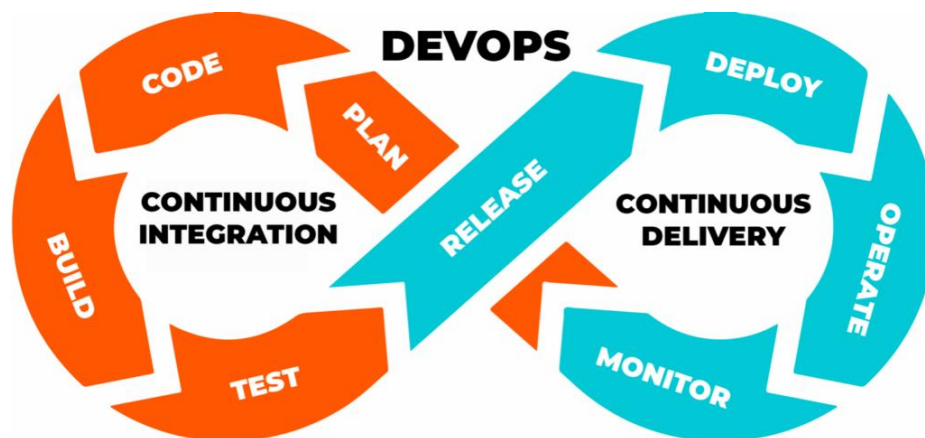


Рисунок 3.9 – DevOps и CI/CD [17]

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, контролируемые 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 включают 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).

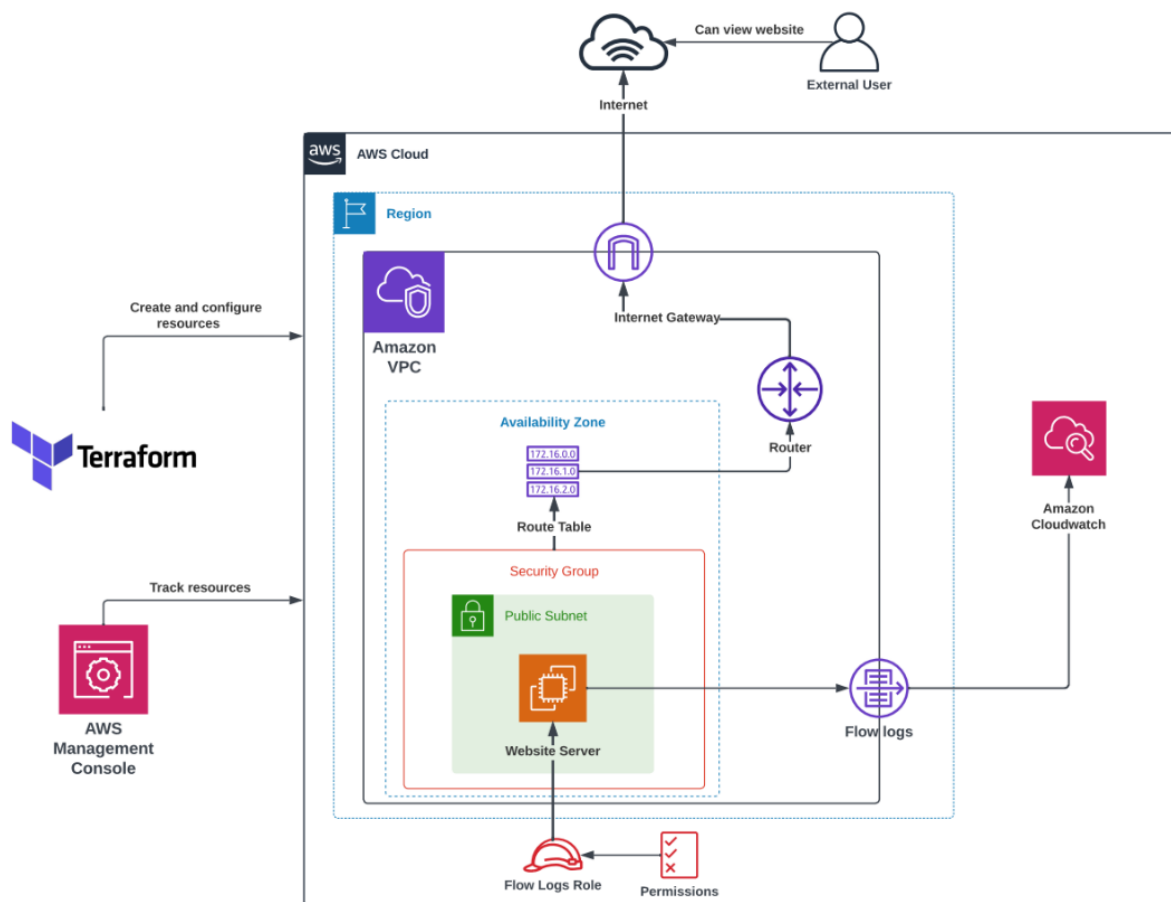


Рисунок 3.10 – Облачная инфраструктура

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

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] Top Cloud Service Providers in 2021: AWS, Microsoft Azure and Google Cloud Platform | CloudThat [Электронный ресурс]. – Режим доступа : <https://www.cloudthat.com/resources/blog/top-cloud-service-providers-in-2021-aws-microsoft-azure-and-google-cloud-platform>. – Дата доступа : 27.03.2024.
- [5] 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.
- [6] Top 10 Cloud Service Providers Globally in 2024 - Dgtl Infra [Электронный ресурс]. – Режим доступа : <https://dgtlinfra.com/top-cloud-service-providers/>. – Дата доступа : 27.03.2024.
- [7] 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.
- [8] 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.
- [9] What is Terraform | Terraform | HashiCorp Developer [Электронный ресурс]. – Режим доступа : <https://developer.hashicorp.com/terraform/intro>. – Дата доступа : 01.04.2024.
- [11] 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.
- [12] 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.

[13] 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.

[14] 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.

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

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

[17] 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.