ВІДЗИВ НАУКОВОГО КЕРІВНИКА НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ

Студента Непші Ігора Олександровича
представлену на здобуття освітнього ступеня бакалавр
зі спеціальності 123 «Комп'ютерна інженерія»

1. Робота виконана на тему „Проект хмарної інфраструктури IT-компанії".

2. Актуальність роботи зумовлена потребою у розробці та впровадженні у практику діяльності організації сучасних хмарних рішень.

3. Основний зміст пояснювальної записки роботи викладено на 70 сторінках тексту, записка містить 3 таблиці, 40 ілюстрацій, додатки. Список використаних літературних джерел налічує 23 найменування. Невід'ємною складовою роботи є розгорнута на базі EC2 мережна інфраструктура компанії.

4. Пояснювальна записка складається з анотації, вступу, трьох розділів основного тексту, висновків, списку використаної літератури, додатків.

Розділ 1 присвячений аналізу принципів, моделей, хмарних провайдерів, протоколів та технологій на яких можна побудувати сучасну інфраструктуру IT-компанії. Проведено огляд та аналіз основних вимог до хмарної інфраструктури, а також проведено порівняння основних провайдерів хмарних послуг.

Розділ 2 присвячений питанням проектування хмарної інфраструктури, вибору оптимальних інструментів для розгортання та конфігурації її елементів. У розділі наведено обґрунтування вибору інструментарію реалізації. Також у розділі наведено опис мережевого середовища в якому працює розроблена інфраструктура.

Розділ 3 присвячений безпосередньо процесу розгортання та конфігурування інфраструктури. У ньому наведено етапи налагодження, опис конфігураційних файлів. Також розділ містить результати перевірки коректності та працездатності створеної інфраструктури.

5. Висновки містять сумарні результати проведеної роботи.

6. Список використаних літературних джерел має достатній для даного виду робіт обсяг, є правильно оформленим та повністю опрацьованим у ході виконання роботи.

7. Додатки містять лістинги конфігураційних файлів налагоджених відповідних програмних компонентів

8. Таблиці та ілюстрації оформлені якісно, є інформативними та наведені у відповідних місцях роботи.

9. Якість оформлення роботи є високою.

10. Виявлені на етапі попереднього захисту недоліки оформлення пояснювальної записки та презентації усунені. Рекомендації стосовно покращення проекту мережної інфраструкутри враховані.

11. Ставлення студента до виконання випускної кваліфікаційної роботи – відповідальне.

12. У цілому при написанні випускної кваліфікаційної роботи студент Непша І.О. виявив здібності до оволодіння знаннями, уміннями і навичками з таких дисциплін, як «Архітектура комп'тера», «Операційні системи», «Комп'ютерні мережі», «Захист інформації в інформаційно-комунікаційних системах», «Адміністрування комп'ютерних систем та мереж». Також студент Непша І.О. показав вміння підібрати необхідну літературу та інші інформаційні джерела, проаналізувати їх, виконати постановку задачі на розробку, розробити проект мережної інфраструктури на основі хмарних технологій, обрати відповідний інструментарій, створити закінчений цілісний продукт (у вигляді змодельованого проекту) і оформити свої розробки та напрацювання як випускну кваліфікаційну роботу.

Випускна кваліфікаційна робота студента Непша І.О. написана на високому рівні та може бути допущена до захисту в Екзаменаційній комісії.

Пропонована оцінка – «відмінно».

Науковий керівник
к.т.н., завідувач кафедри КІ та КБ — А.А. Єфіменко
16 червня 2020 року

РЕЦЕНЗІЯ
на кваліфікаційну роботу
студента 4 курсу групи КІ-2 спеціальності 123 «Комп'ютерна інженерія»
факультету інформаційно-комп'ютерних технологій
Державного університету «Житомирська політехніка»
Непші Ігора Олександровича
на тему: „Проект хмарної інфраструктури ІТ-компанії"

Кваліфікаційна робота студента Непші І.О. „Проект хмарної інфраструктури ІТ-компанії" присвячена актуальній тематиці – розробці та впровадженні у практику інфраструктури ІТ-компанії, яка базується на хмарних рішеннях.

Робота містить реалізований на базі Amazon Web Services проект хмарної інфраструктури ІТ-компанії та пояснювальну записку наступного складу: вступ, три розділи основного тексту, висновки, список використаної літератури, додатки.

Перший розділ присвячений аналізу принципів, моделей, хмарних провайдерів, протоколів та технологій, на яких можна побудувати сучасну хмарну інфраструктуру ІТ-компанії. Проведено огляд та аналіз основних вимог до хмарної інфраструктури, а також проведено порівняння основних провайдерів хмарних послуг. Другий розділ присвячений питанням проектування хмарної інфраструктури, вибору оптимальних інструментів для розгортання та конфігурації. Розділ також містить обґрунтування інструментарію та та опис проекту мережевого середовища, в якому працює інфраструктура. Третій розділ присвячений безпосередньо процесу розгортання та конфігурування інфраструктури, у розділі приведено опис процесу та конфігураційних файлів, а також перевірено коректність та працездатність створеної інфраструктури..

Зміст роботи відповідає поставленому завданню. Кваліфікаційна робота виконана на високому рівні. Тема роботи розкрита повністю. Студентом продемонстровані ґрунтовні фахові компетентності, що отримані як під час навчання, так і у процесі виконання роботи. Матеріал пояснювальної записки викладено логічно, науково і технічно грамотно. Ілюстративний матеріал (рисунки, таблиці), що наведений у роботі, оформлений якісно і наведений у відповідних місцях. Оформлення роботи відповідає вимогам стандартів. Проект інфраструктури реалізовано на базі хмарного провайдера Amazon Web Services. Його функціонал відповідає заявленому.

Позитивною стороною роботи є її практична спрямованість. Розроблені у роботі рішення можуть бути застосовані на практиці при розгортанні мережної хмарної інфраструктури компанії. Недоліки роботи: недостатньо глибоко проаналізовані конкурентні хмарні рішення, у пояснювальній записці роботи наявні деякі відхилення від вимог щодо оформлення (наприклад, переліки). Зазначені недоліки не впливають на високу якість роботи.

У цілому кваліфікаційна робота студента Непші І.О. „Проект хмарної інфраструктури ІТ-компанії" відповідає вимогам стандарту рівня вищої освіти «бакалавр» спеціальності 123 «Комп'ютерна інженерія» і рекомендується до захисту в ЕК.

Пропонована оцінка роботи – «відмінно».

Рецензент,
Д.п.н., доц., проф. каф. ІПЗ Т.А.Вакалюк
16 червня 2020 р.

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп’ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему:
«Проект хмарної інфраструктури ІТ-компанії»

Виконав студент 4-го курсу, групи КІ-2
спеціальності 123 «Комп’ютерна інженерія»
_____ Непша І. О.
Керівник, завідувач кафедри КІ та КБ, к.т.н.
_____ Єфіменко А. А.
Рецензент, професор кафедри ІПЗ, д.пед.н.,
доцент
_____ Вакалюк Т. А.

Житомир–2020

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА КІБЕРБЕЗПЕКИ
СПЕЦІАЛЬНІСТЬ 123 «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерної інженерії
та кібербезпеки
Єфіменко А. А.
«06» квітня 2020 р.

ЗАВДАННЯ
на випускну кваліфікаційну роботу

Студента **Непші Ігора Олександровича**
Тема роботи: **Проект хмарної інфраструктури IT-компанії**
Затверджена Наказом університету від **«06» квітня 2020 р. № 98с**
Термін здачі студентом закінченої роботи - **1 червня 2020 р.**
Вихідні дані для роботи:

1. Фізична топологія мережі організації
2. Віртуальні машини EC2 на Amazon Web Services
3. Операційні систмеи Windows Server 2016, Linux
4. Веб-сервери Nginx, IIS
5. Системи моніторингу Prometheus та Grafana

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Єфіменко А. А.	6.04.2020	6.04.2020
2	Єфіменко А. А.	20.04.2020	20.04.2020
3	Єфіменко А. А.	11.05.2020	11.05.2020

Керівник _____ Єфіменко А. А.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі	6 квітня 2020 – 7 квітня 2020	Виконано
2	Пошук, огляд та аналіз аналогічних рішень	8 квітня 2020 – 11 квітня 2020	Виконано
3	Формулювання технічного завдання	12 квітня 2020 – 16 квітня 2020	Виконано
4	Проектування структури	25 квітня 2020 – 30 квітня 2020	Виконано
6	Розгортання серверів в хмарі	1 травня 2020 – 2 травня 2020	Виконано
7	Розгортання та налагодження сервісів	3 травня 2020 – 13 травня 2020	Виконано
8	Розробка прав та політик доступу	13 травня 2020 – 15 травня 2020	Виконано
9	Розгортання та налаштування системи моніторингу	16 травня 2020 – 25 травня 2020	Виконано
10	Тестування роботи сервісів	26 травня 2020 – 27 травня 2020	Виконано
11	Аналіз результатів тестування	28 травня 2020 – 29 травня 20202	Виконано
12	Написання пояснювальної записки	30 травня 2020 – 3 червня 2020	Виконано
13	Підготовка презентації та виступу	4 червня 2020 – 5 червня 2020	Виконано
	Отрмимання відгуку та рецензії на кваліфікаційну роботу	5 червня 2020 – 9 червня 2020	Виконано
14	Захист	20 червня 2020	Виконано

Студент _____ Непша І.О

Керівник _____ Єфіменко А.А

РЕФЕРАТ

Кваліфікаційна робота складається з проекту хмарної інфраструктури IT-компанії та пояснювальної записки до нього. Пояснювальна записки да роботи містить 70 сторінок, 40 ілюстрацій та 3 таблиць.

Метою роботи є розробка проекту хмарної інфраструктури IT-компанії на базі Amazon Web Services.

В роботі визначено основні завдання на розробку інфраструктури, яка задовольнить потребам сучасних IT-компаній. Також в роботі проаналізовано основні інструменти та підходи до побудови хмарної інфраструктури, описано сферу використання, обґрунтовано вибір хмарного провайдера та програмних компонентів. Розроблено, сформовано та налаштовано проект хмарної інфраструктури. Наведено сценарій розгортання та конфігурування серверів, загальна топологія, опис серверів та мережевого середовища, а також озроблено політики безпек та доступу. Також наводиться результат тестування коректності та роботоздатності інфраструктури

КЛЮЧОВІ СЛОВА: AWS, ІНФРАСТРУКТУРА, NGINX, WINDOWS SERVER, LINUX, DOCKER, МЕРЕЖА.

РЕФЕРАТ

Кваліфікаційна робота складається з проекту хмарної інфраструктури IT-компанії та пояснювальної записки до нього. Пояснювальна записки да роботи містить 70 сторінок, 40 ілюстрацій та 3 таблиць.

Метою роботи є розробка проекту хмарної інфраструктури IT-компанії на базі Amazon Web Services.

В роботі визначено основні завдання на розробку інфраструктури, яка задовольнить потребам сучасних IT-компаній. Також в роботі проаналізовано основні інструменти та підходи до побудови хмарної інфраструктури, описано сферу використання, обґрунтовано вибір хмарного провайдера та програмних компонентів. Розроблено, сформовано та налаштовано проект хмарної інфраструктури. Наведено сценарій розгортання та конфігурування серверів, загальна топологія, опис серверів та мережевого середовища, а також озроблено політики безпек та доступу. Також наводиться результат тестування коректності та роботоздатності інфраструктури

КЛЮЧОВІ СЛОВА: AWS, ІНФРАСТРУКТУРА, NGINX, WINDOWS SERVER, LINUX, DOCKER, МЕРЕЖА.

ЗМІСТ

РЕФЕРАТ **4**
ЗМІСТ **5**
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ **6**
ВСТУП **7**
РОЗДІЛ 1. Огляд та аналіз інфрастрактурних рішень для IT компаній .. **9**
1.1. Огляд та аналіз вимог до інфраструктури 9
1.2. Хмарна інфрастурктура 13
1.3. Порівняння хмарних провайдерів 18
РОЗДІЛ 2. ПРОЕКТ ХМАРНОЇ ІНФРАСТРУТУРИ КОМПАНІЇ **29**
2.1. Проектування серверної частини 29
2.2. Проект груп та політик доступу 35
2.3. Розробка системи моніторингу 39
РОЗДІЛ 3. РОЗГОРТАННЯ, ІНТЕГРАЦІЯ ТА ПЕРЕВІРКА КОРЕКТНОСТІ РОБОТИ ІНФРАСТРУКТУРИ **46**
3.1. Розгортання та перевірка доступності серверів 46
3.2. Встановлення та конфігурування основних компонентів Windows Server 49
3.3. Конфігурація, інтеграція та тестування серверу на базі Linux 57
3.4. Аналіз результатів 65
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ **69**

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AWS – Amazon Web Services

GCP – Google Cloud Platform

ОС – операційна система

SSH – Secure Shell

OSI – Open Systems Interconnection

TCP – Transmission Control Protocol

AD – Active Directory

IP – Internet Protocol

LAN – Local Area Network

MAC – Media Access Control

IaC – Infrastructure-as-Code

Paas - Platform as a Service

SaaS - Software as a Service

NAT – Network Address Translation

DHCP – Dynamic Host Configuration Protocol

ЦОД – Центр обробки даних

ПК – Персональный комп’ютер

ПЗ – Програмне забезпечення

CPU – Central Processing Unit

RAM – Random Access Memory

CSP – Cloud Service Provider

ВСТУП

Актуальність теми. В сучасному світі інформаційні технології тісно переплітаються з усіми сферами життя людини. Бізнес та підприємства в десятки та сотні разів оптимізують свою діяльність за рахунок побудови власної IT інфраструктури. Наприклад, торгівля, виробництво і навіть сфера надання послуг значно розширили свою діяльність за рахунок виходу в інтернет, створення власних сайтів або веб-додадків, які дозволяють людям переглядати потрібні їм товари або послуги та замовляти їх, не виходячи з дому.

Але для того, щоб усе це працювало і усі були задоволені потрібно правильно спроектувати та побудувати інфраструктуру, яка б могла задовільнити потреби бізнесу. Потрібно правильно подібрати та сконфігурувати параметри серверів, на яких будуть працювати сайти, веб-додатки та інші сервіси потрібні для коректної та комфортної роботи. Раніше усі компанії купували реальне залізо для свої серверів та розгортали свої сервіси там, але такий підхід має ряд недоліків.

З появою хмарних провайдерів, багато компаній які стрімко ростуть і розвивиються мігрували свою інфраструктуру в хмару. Особоливо це важливо та корисно для компаній які не можуть дозволити собі чекати поки придбається та налаштується реальне залізо. Хмарні обчислення надають механізми практично моментального розгортання, налаштування та маштабування.

Актуальність обраної теми полягає в тому, що на прикладі побудованої в хмарі інфраструктури можна побачити та оцінити основні переваги та недоліки такого підходу.

Метою кваліфікаційної роботи є проектування архітектури, розробка алгоритмів роботи та реалізація хмарної інфраструктури в хмарі.

Встановлена мета обумовлює наступні завдання:

- аналіз основних підходів до побудови інфраструктури
- вибір засобів реалізації системи;
- розробка сучасної хмарної інфраструктури з можливістю подальшого розширення;

– реалізація хмарною інфраструктури.

Об’єктом дослідження є процес розгортання та налаштування хмарних серверів та сервісів для задоволення задач бізнесу

Предметом дослідження є методи і засоби реалізація оптимальної інфраструктури.

В процесі виконання випускної роботи було розроблено розгорнуто інфраструктру, використано послуги хмарного провайдера AWS, теорію нереляційних баз даних на прикладі MongoDB, технологію контейнеризації на базі Docker, інструмент керування користувачами та каталогами Active Directory

РОЗДІЛ 1. Огляд та аналіз інфрастрактурних рішень для ІТ компаній

1.1. Огляд та аналіз вимог до інфраструктури

Інформаційний обмін в сучасному світі є невід’ємною складовою нашого життя. На сьогоднішній день неможливо уявити собі роботу популярної, успішної компанії, без результативного використання інформаційних технологій. Інформаційний світ стає з кожним днем все більше і вдосконалюється зі стрімкою швидкістю. Технології, що знаходяться на етапі розробки ще вчора, сьогодні - невід'ємний засіб ведення бізнесу. Завдяки цьому, побудова високоякісної інформаційної інфраструктури бізнесу, є однією із первинних задач. У сучасних реаліях успішність і прибутковість бізнесу, ефективність роботи компанії, її конкурентні переваги багато в чому визначається саме рівнем розвитку, стабільністю та безпекою ІТ-інфраструктури. Сьогодні вже неможливо уявити собі сучасний офіс, успішного бізнесу без застосування інформаційних технологій. По суті, це один з найважливіших активів компанії. Будь-якій організації, будь то скромна майстерня або велика ІТ компанія, усім потрібні можливі способи зберігання і обробка особистих даних, управління роботою співробітників, а також фінансами та аналітичним обліком.

Найголовніше, що ІТ-інфраструктура повинна задовольняти потребам бізнесу компанії. Для того, щоб організувати дійсно надійну, високопродуктивну і масштабовану ІТ-інфраструктуру, потрібно володіти великою кількістю кваліфікованих фахівців, а також чималим досвідом побудови ІТ-інфраструктур.

Причому постійний розвиток бізнесу, наприклад розширення філіальної мережі, вихід на нові ринки, ускладнення процесів, тягне за собою безперервне ускладнення програмно-апаратного комплексу компанії. Тому необхідність раціонального управління ІТ, особливо для великих структур, виходить на перший план.

Важливою особливістю сучасних великих компаній є оперативність, точність і робота з величезною кількістю інформації, яку необхідно зберігати і захищати. IT-інфраструктура таких організацій, що складається з великої кількості компонентів - додатків, комунікаційних сервісів, інфраструктури серверів, АРМ і систем зберігання даних, системи інформаційної безпеки, мережевий і інженерної інфраструктури, - вимагає системного підходу до управління.

Ключовими завданнями є грамотний вибір програмних та апаратних рішень, адекватне управління всіма системами компанії, забезпечення безперебійності і безпеки роботи. Традиційно вимоги різних компаній до IT-інфраструктурі залежать від особливостей бізнесу. Наприклад, для великих торгових підприємств критично важлива можливість швидко нарощувати роздрібну мережу: відкривати нові магазини і склади і оперативно інтегрувати їх в загальну IT-систему. У зв'язку з цим для IT-інфраструктури торгових мереж характерна масштабність і велика територіальна розподіленість.

В загальному можна сказати, що сучасна IT-інфраструктура повинна відповідати наступним вимогам:

- Безперебійна робота
- Маштабованість
- Захищеність
- Прозорість та керованість
- Адекватна вартість володіння та підтримки

Безперебійна робота передбачає, в першу чергу те що інфраструктура не повинна заважати бізнес-процесам. Від сучасних великих компаній потрібна робота в режимі 24×7 навіть короткочасні збої призводять до простою критичних бізнес-процесів, до великих фінансових і репутаційних втрат. Збої в роботі відбуваються з двох основних причин: вихід з ладу обладнання та різні помилки користувачів або програмні. Проблему з обладнанням можна вирішити дублюванням або об'єднанням декількох екземплярів пристроїв в кластер. Тоді

при виході з ладу одного з них решта будуть виконувати його роботу. Зокрема, механізм віртуалізації дозволяє ще більше розширити можливості дублювання. Для захисту від помилок користувачів і збоїв в програмного забезпечення компанії створюють систему резервного копіювання і відновлення, яка дозволяє відновлювати актуальні дані в короткий термін. Ймовірні проблеми і збої краще запобігати або зупиняти на ранніх стадіях, ніж виправляти. Для цих цілей у великих компаніях використовуються системи проактивного моніторингу, які повідомляють адміністраторам про проблеми в IT-інфраструктурі раніше, ніж ці проблеми призведуть до неприємних наслідків і зупинці бізнес-процесів.

Маштабованість дозволяю розширюватись без неприємних наслідків та потреби перебудови всієї інфраструктури. Для поліпшення масштабованості застосовуються уніфіковані рішення - від однакових конфігурацій операційних систем і програмного забезпечення до шаблонів і скриптів для розгортання інфраструктури цілих філій (наприклад, нових банківських відділень або роздрібних магазинів). Чим більше процесів в IT-інфраструктурі автоматизовано, тим, як правило, краще її масштабованість і керованість. Уніфіковані рішення також сприяють підвищенню швидкості змін IT у відповідь на розвиток бізнесу. Швидше один раз поміняти загальний шаблон, ніж вносити зміни в кожен екземпляр. Найпростішим прикладом такого інструменту для тиражування рішень можуть служити групові політики. Одного разу змінивши налаштування політики на центральному сервері, ми звільняємося від необхідності міняти їх на кожному сервері або робочої станції.

Захищеність усієї інфраструктури є однією з найважливіших, можна навіть сказати фундаментальною задачею. Оскільки, навіть маючи ідеально налаштовану та відладжену інфрастркутуру, можна втратити усе через слабке місце в системі безпеки. Чим складніше інфраструктура, тим більше різних систем вона включає, тим вона вразливіше. Обсяги даних і складність бізнес-процесів безпосередньо впливають на архітектуру системи захисту інформації. За інформацією, яка обробляється в банківських системах, стоять реальні гроші. Очевидно, що доступ до таких даних може привести до серйозних збитків. Ця

особливість різко розширює коло потенційних зловмисників і підвищує вимоги до безпеки. Система захисту інформації в банках обов'язково включає управління правами доступу, шифрування каналів зв'язку, тотальний моніторинг, ведення докладних журналів. У меншій мірі це питання стосується торгових мереж, але і тут важливість фінансової та персональної інформації, а також збільшення обсягів електронних платежів виставляють практично такі ж вимоги до безпеки, як і в банківському секторі.

Прозорість та керованість досягається за рахунок збору даних про інформаційну систему компанії. Це можуть бути звіти про роботу обладнання і програмного забезпечення, відомості про зміни в архітектурі, облік ліцензій - все, що дозволяє прогнозувати поведінку IT при будь-яких змінах. Чим більше інформації про процеси і стан IT-інфраструктури враховується, тим точніше ми можемо передбачити його поведінку і, отже, тим ефективніше можемо нею керувати. Але просто отримати інформацію на якийсь певний момент часу недостатньо. Необхідна історія зміни станів, історія подій, що відбувалися в IT. Отримати таку інформацію без постійного моніторингу інфраструктури неможливо. Якщо кожну подію не тільки фіксувати, а й описувати, оцінювати, це дозволить отримати більш повний прогноз на випадок виникнення проблеми в майбутньому, а також скоротити часові і фінансові витрати на її рішення.

Слід мати на увазі, що основні витрати великої компанії йдуть не на покупку або створення інфраструктури, а на її підтримку, яка має на увазі постійне коректування під поточні потреби бізнесу. Для сучасних великих компаній, що використовують складні бізнес-процеси, зміни самі по собі не дешеві. А помножені на масштаб, вони часто складають астрономічні суми. Тому вимога розумної вартості володіння IT-інфраструктурою стає особливо важливим.

Виходить, що в IT-інфраструктурі підприємства представлений не просто набір IT-рішень, які випадково зібрані в одному місці. Це велика інтегрована система, яка забезпечує роботу організації в цілому. Як і будь-яку з систем, таку систему потрібно правильно проектувати і надалі - експлуатувати. У той же час

вона настільки складна, унікальна і нестандартна, що проводити з нею роботу, як з будь-якою іншою автоматизованою системою, не представляється можливим. А такий вид систем впливає безліч факторів, тому необхідно не пускати на самоплив розвиток IT-інфраструктури, а розробляти спеціальну IT-стратегію. Вона являє собою систему різних пріоритетів, а також планів, правил, які дають можливість домогтися адекватності інфраструктури інформаційних технологій потребам всього бізнесу

1.2. Хмарна інфрастурктура

Хмарні технології (або хмарні обчислення, cloud computing) - технології розподіленої обробки цифрових даних, за допомогою яких комп'ютерні ресурси надаються інтернет-користувачеві як онлайн-сервіс. Програми запускаються і видають результати роботи в вікні web-браузера на локальному ПК. При цьому всі необхідні для роботи програми та їх дані знаходяться на віддаленому інтернет-сервері і тимчасово кешуються на клієнтській стороні: на ПК, ігрових приставках, ноутбуках, смартфонах. Перевага технології в тому, що користувач має доступ до власних даних, але не повинен піклуватися про інфраструктуру, операційну систему та програмне забезпечення, з яким він працює. Слово «хмара» - це метафора, що уособлює складну інфраструктуру, що приховує за собою всі технічні деталі. Технології хмарних обчислень націлені на вирішення наступних завдань:

- Зручна робота з файлами на кількох гаджетах: їх редагування і обробка без перенесення з одного пристрою на інший, без необхідності піклуватися про сумісність програмного забезпечення
- Рішення проблеми обмеженого об'єму жорсткого диска комп'ютера або флеш-карти
- Питання ліцензованого програмного забезпечення

– Можливість одночасної роботи над одним документом кільком людям

Одним з перших кроків до концептуалізації хмарних обчислень вважаються поява CRM-системи Salesforce.com, що надається за підпискою в вигляді веб-сайту (1999) і початок надання послуг з доступу до обчислювальних ресурсів через Інтернет книгарнею Amazon.com (2002) Розвиток сервісів Amazon, фактично перетворилася завдяки цим послугам в технологічну компанію, привело до формулювання ідеї обчислювальної еластичності і запуску в серпні 2006 року проекту під назвою Elastic Computing Cloud (Amazon EC2). Практично одночасно із запуском ECC терміни cloud і cloud computing прозвучали в одному з виступів глави Google Еріка Шмідта, починаючи з цього часу зустрічаються численні згадки хмарних обчислень в ЗМІ, в публікаціях фахівців з інформаційних технологій, в науково дослідної середовищі.

На початку 2008 року OpenNebula NASA, в рамках проекту, що фінансується Європейською комісією RESERVOIR, стала першим програмним забезпеченням з відкритим вихідним кодом для розгортання приватних і гібридних хмар. У квітні 2008 року Google випустив Google App Engine в бета-версії. До середини 2008 року Gartner побачив можливість за допомогою хмарних обчислень «сформувати відносини між споживачами IT-послуг, тими, хто використовує IT-послуги, і тими, хто їх продає», і зазначив, що «організації переходять від належних компанії апаратних і програмних активів до використання сервісно-орієнтованих моделей », так що« прогнозований перехід до обчислень призведе до різкого зростання IT-продукції в деяких областях і значного скорочення в інших областях »

Запуск у 2009 році додатків Google Apps відзначається як наступний важливий крок до популяризації і осмислення хмарних обчислень. У 2009-2011 роках були сформульовані кілька важливих узагальнень уявлень про хмарні обчислення, зокрема, висунута модель приватних хмарних обчислень, актуальна для застосування всередині організацій, виділені різні моделі обслуговування (SaaS, PaaS, IaaS, які будуть описані детальніше далі)

З кожним роком бізнес очікує від ІТ все більшої швидкості виведення і надання для себе нових внутрішніх і зовнішніх сервісів. Візьмемо простий приклад. Десять років тому виділення для нового співробітника робочого місця могло займати цілий тиждень (установка ПК, створення облікового запису, створення поштової скриньки та інше), А зараз це займе 15 хвилин. Грамотний системний адміністратор скаже: «Я легко побудую інфраструктуру, яка буде вирішувати таке завдання і без хмарних послуг» і, безумовно, матиме рацію. Але виникає питання: скільки часу піде на реалізацію, скільки завдань буде відкладено, поки системні адміністратори займаються реалізацією такого проекту?

Щойно був описани проект, цінність якого цілком очевидна, але візьмемо інший приклад. Компанія приймає рішення відкрити нову гілку свого бізнесу. Буде вона успішною або через рік буде закрита - ніхто не знає, але ІТ-ресурси під неї потрібні зараз. Точно так само ІТ-служба може почати будувати інфраструктуру у себе, проводячи закупівлю обладнання, його установку, настройку, перевірку, щоденне обслуговування, або використовувати хмара провайдера як інструмент і зайнятися вирішенням конкретної бізнес-завдання. Ну а про те, що в разі провалу проекту від ресурсів із хмари можна просто відмовитися, а з закупленим "залізом" щось треба робити. Хмарні обчислення передбачають наявність великого числа підключених через мережу комп'ютерів, які фізично можуть розміщуватися в будь-якій точці земної кулі. Постачальники послуг великою мірою покладаються на віртуалізацію при наданні послуг хмарних обчислень. Хмарні обчислення допомагають скоротити операційні витрати за рахунок більш ефективного використання ресурсів,а також вирішують різні завдання управління даними, забезпечуючи:

- повний доступ до даних організації в будь-який час;
- оптимізацію ІТ-інфраструктури за рахунок передплати тільки на необхідні послуги;
- виключення або зниження необхідності розгортання і підтримки обладнання на власних платформах;

– скорочення витрат на обладнання і електроенергію, зменшення вимог до матеріальної частини та потреби в навчанні персоналу;
– оперативне реагування на зростаючі вимоги до обсягу даних.

Хмарні обчислення з моделлю оплати «в міру використання» дозволяють організаціям ставити до обчислення и зберігання як до комунальної послуги и не інвестуваті в інфраструктуру. В результате капіталовкладення превращаются в експлуатаційні витрати.

Умовно всі види хмарних послуг можна розділити на три типи:

– Infrastructure as a Service (інфраструктура як послуга)
– Platform as a Service (платформа як послуга);
– Software as a Service (програмне забезпечення як послуга)

IaaS, PaaS або SaaS - це моделі надання хмарних сервісів. Те, як вони співвідносяться один з одним, часто зображують у вигляді піраміди з різним рівнем контролю інформації. Вершина - це кінцевий користувач, який працює з особистими даними, «загорнутими» у вигляді програми або сервісу з зручним інтерфейсом. Програма або сервіс розгортаються на якійсь технологічній платформі, це другий рівень піраміди. Нарешті, її основа - це інфраструктура: віртуальні сервери, обчислювальні потужності, накопичувачі і канали зв'язку.



Рис. 1.1 Структура та зони відповідальності хмарних сервісів

SaaS (Software-as-a-Service). Ця хмарна модель - найпоширеніша. Програми та сервіси розробляє і обслуговує провайдер, розміщує їх в хмарі і пропонує кінцевому користувачеві через браузер або додаток на його ПК. Клієнт лише вносить абонплату (або користується сервісом безкоштовно), оновленням і технічною підтримкою програм займається провайдер. SaaS-сервіси можуть надавати місце для зберігання файлів (Dropbox), офісний пакет документів для роботи (Google Doc, Microsoft Office 365), допомагати організовувати фотографії (Flickr) або спілкуватися з іншими людьми (Facebook). Основний клієнт SaaS-сервісів - звичайний користувач.

PaaS (Platform-as-a-Service). В цьому випадку хмарний провайдер надає доступ до операційних систем, засобів розробки і тестування, системам управління базами даних. Провайдер контролює не тільки сервери, системи зберігання даних і обчислювальні потужності, але також пропонує користувачеві на вибір певні платформи і засоби управління ними. Приклади PaaS: Google App Engine, IBM Bluemix, Microsoft Azure, VMWare Cloud Foundry. Користувачі PaaS-сервісів - це розробники ПЗ.

IaaS (Infrastructure-as-a-Service). При цій моделі споживач отримує інформаційно-технологічні ресурси - віртуальні сервери з певною обчислювальною потужністю та обсягами пам'яті. Всім «залізом» займається провайдер. Він встановлює на нього ПЗ для створення віртуальних машин, але не займається установкою і підтримкою ПЗ користувача. Провайдер контролює тільки фізичну та віртуальну інфраструктуру. Приклади IaaS: IBM Softlayer, Hetzner Cloud, Microsoft Azure, Amazon EC2, GigaCloud. Клієнти IaaS - це системні адміністратори компаній.

З точки зору кінцевого користувача SaaS - найбільш зрозуміла і зручна хмарна модель. Часто простіше і ефективніше використовувати готовий SaaS-сервіс, який вже відповідає певним вимогам. Але готові рішення не завжди існують, і в такому випадку моделі PaaS і IaaS - незамінні.

1.3. Порівняння хмарних провайдерів

В останні роки можливість використання хмарних інфраструктур все міцніше закріплюється в свідомості власників корпоративних бізнес-рішень (IT-рішень). Навіть якщо підприємство не робить великих ставок на обчислення в публічних хмарах, використання хмарних рішень в тій чи іншій формі є невід'ємною частиною стратегії і планів розвитку інфраструктури більшості підприємств в силу доступності, економічної ефективності та достатнього рівня надійності. Для підприємств, які збираються вступити в цю подорож, стають актуальними такі питання: «Яку хмарну платформу мені вибрати?», «Яка хмарна платформа буде найбільш економічно ефективна для моїх завдань?» і «Як спланувати міграцію сервісів в хмару?»

Далі буде проведено порівняння основни інфраструктурних сервісів найпопулярніших хмарних провайдерів, таких як Amazon Web Services (AWS), Microsoft Azure і Google Cloud Platform (GCP). На додаток до основної інфраструктури кожен постачальник хмарних послуг надає свої унікальні пропрієтарні пропозиції в NoSQL, Big Data, Analytics, ML і інших подібних областях, які можуть впливати на вибір хмари для підприємства в залежності від його конкретних бізнес-вимог і технічних вимог. Але перш за все потібно виділити критерії об’єктивної оцінки сервісів надання послуг.

На жаль, в рамках даного аналізу навряд чи вийде розповісти про всі тонкощі і нюанси, але можна бути впевненим - після завершення порівняння можна буде дати відповіді на наступні питання:

– на що звертати увагу при виборі постачальника?
– які критерії порівняння використовувати?
– чи варто запитувати підтверджуючі документи і багато іншого?

Почнемо з чогось базового. Як ми уявляємо собі дата-центр? Чистота, високий рівень дисципліни, доступ за перепустками і багато-багато обладнання. Саме на нього і варто звернути увагу в першу чергу. При виборі сервіс-провайдера приділити час та перевірити залізо по чек-листу: Моделі обладнання.

Застаріле залізо - гарантія регулярних збоїв і зниження продуктивності. Надійний сервіс-провайдер регулярно модернізує апаратний комплекс і не допустить появи старого заліза. Клас обладнання. Виробники, прагнучи зайняти вигідну позицію на ринку, випускають різні лінійки обладнання - від корпоративних (enterprise) рішень до середніх (midrange) і дешевих (low-end). Логічно, що орієнтуватися варто на провайдера з корпоративними-рішеннями.

При оренді віртуальної інфраструктури на базовому рівні необхідно визначити потреби в наступних ресурсах:

– тактова частота CPU (в GHz);
– обсяг RAM (в Gb);
– обсяг необхідної дискової пам'яті (в Gb)

Однак слід звернути увагу, що на роботу сервісу або програми в хмарі впливає не тільки виділений обсяг дискового простору, але і продуктивність дискової підсистеми. Як правило, другий фактор відіграє важливішу роль. Навіть якщо ви запросите у провайдера конкретний тип дисків, ви не отримаєте 100% гарантію продуктивності, адже на неї також впливає навантаження від "сусідів" по хмарі.

Об'єктивно оцінити продуктивність можна за допомогою двох показників:

– IOPs - кількість операцій введення-виведення;
– Latency - рівень затримок при зверненні

Вкрай важливо, щоб провайдер був готовий відкрито обговорювати використовувані дискові масиви і був готовий надати вам відповідні можливості управління IOPs. Однак, якщо постачальник послуг гарантує вам ці метрики, немає необхідності глибоко занурюватися в деталі.

При виборі IaaS-провайдера обов'язково потрібно впевнитися у надійності самої платформи, на якій розміщена хмарна платформа. Багато провайдерів проводять невеликі екскурсії або продемонструють свою платформу за запитом хорошому постачальнику нема чого приховувати. Однак цього не завжди достатньо. Як правило, будь-якого клієнта хотілося б, щоб його додатки

розміщувалися в дата-центрі, рівень якої підтверджений профільними спеціалізуються на цьому компаніями

Один зі стовпів високої доступності платформи - відладженні процеси супроводу віртуальної інфраструктури. У список цих процесів входить:

- управління інцидентами;
- обробка запитів на обслуговування;
- управління змінами;
- опціонально - управління знаннями співробітників підтримки

Процес обробки інцидентів повинен бути максимально формалізований, що досягається використанням єдиної системи обліку запитів користувачів і повідомлень від системи моніторингу. Такий підхід дозволяє працювати з інцидентами в міру їх надходження, врахувати всі до єдиного повідомлення та інформацію і, як наслідок, знизити вплив на системи і час їх відновлення.

Один з атрибутів грамотно вибудуваного процесу управління змінами - єдиний регламент внесення змін в інфраструктуру. Це включає в себе:

- ретельне планування будь-яких змін;
- тестування;
- наявність плану відкату внесених змін на випадок, якщо щось піде не так;
- мінімізацію ризиків порушення SLA

Як правило, клієнт, у якого з'явився запит або виникла проблема, в першу чергу "стукає" саме до працівника технічної підтримки. Однак недостатньо просто посадити інженерів на телефон - сервіси корпоративного рівня повинні в обов'язковому порядку супроводжуватися високим рівнем обслуговування. Щоб його забезпечити, служба підтримки повинна відповідати певним критеріям:

- обслуговування в режимі 24/7/365;
- кілька каналів зв'язку: телефон, пошта, система звернень на сайті.
- дотримання заявлених параметрів обслуговування:
- час реакції;
- максимальний термін рішення типового запиту;

– максимальний термін рішення інциденту.
– обов'язкове і своєчасне інформування про проведення робіт та усунення інцидентів;
– допомога у вирішенні завдань, що виходять за рамки послуги.

Замовників не завжди цікавить використовуваний провайдером гіпервізор, однак з певних причин рекомендуємо звернути на це увагу:

– Гіпервізор можуть відрізнятися функціональністю, що безпосередньо впливає на продуктивність віртуальних машин і рівень доступності сервісів;
– різні Гіпервізор можуть підтримувати різні ОС;
– рівень захищеності з точки зору ІБ і вразливостей теж може відрізнятися.

Зараз основними гіпервізорами є VMware ESXi, Hyper-V від Microsoft, Citrix Xen і KVM від Red Hat. Однак сьогодні є беззаперечним лідером ринку віртуалізації - компанія VMware, яка фактично і створила його з нуля. Одне з його основних переваг - високий рівень безпеки.

Перед укладенням договору на оренду віртуальної інфраструктури необхідно не тільки по можливості провести тестування - як правило, провайдери безкоштовно надають ресурси для цих цілей - а й ознайомитися з SLA, угодою про рівень послуг, що надаються. За якими критеріями можна перевірити SLA:

– всім типовим запитам повинні відповідати параметри часу рішення, зафіксовані в SLA; для нетипових запитів повинен бути зазначений фіксований відрізок часу, за який має бути визначено час вирішення.
– за порушення SLA провайдер послуг повинен нести фінансову відповідальність.

Після визначення деяких критеріїв оцінки та порівняння хмарних провайдерів можна приступати до вибору найоптимальнішого варіанту під конкретні задачі.

Першим, та найбільшим «гравцем» на ринку надання хмарних послуг виступає Amazon Web Services, який залишає за собою право волдіти більшістю (51%) усіх хмарних сервісів у світі.

AWS також є провайдером загальнодоступних хмар з найширшим спектром продуктів, опцій для обчислень і зберігання даних, а також послуг, які клієнт може передати під управління. AWS Marketplace також є найбільшою торговим майданчиком для додатків і пристроїв сторонніх виробників. Товари та послуги на такому ринку і сам AWS часто оновлюються, щоб постійно додавати функції продуктів для своїх клієнтів. Крім основних послуг, вони також випускають нетрадиційні сервіси, такі як SnowMobile (пристрій передачі даних з автомобіля), RoboMaker (фреймворк для робототехніки) і «наземна станція як послуга» (для керованої завантаження даних з супутника). Це підтримує інтерес клієнтів і може відкрити цілі галузі.

AWS - відмінний вибір як для стартапів, так і для підприємств. AWS надає широкий спектр послуг, які можуть використовувати клієнти - від невеликих web сервісів до масштабних міграцій центрів обробки даних в хмару. Щоб допомогти клієнтам всіх форм і розмірів почати роботу на платформі, AWS випустила нібито нішеві сервіси, такі як RoboMaker, з одного боку, і одночасно створила такі сервіси, як LightSail (віртуальний приватний сервер), щоб допомогти навіть самим маленьким робочим навантаженням з одним сервером.

На даний момент AWS також пропонує найвищі на ринку варіанти здійснення обчислень і засоби зберігання даних. Широкий діапазон типів віртуальних машин (136 типів віртуальних машин в 26 родинах віртуальних машин) дозволяє замовникам запускати практично всі - від невеликих web-систем до найбільших робочих навантажень HPC (High Performance Computing) і SAP(System Analysis and Program Development).

AWS сподівається, що різні розміри віртуальних машин будуть відповідати вашим вимогам. Тому він не підтримує створення призначених для користувача розмірів віртуальних машин (vCPU, RAM). На відміну від інших хмарних провайдерів (CSP), AWS надає тільки певний набір сімейств

віртуальних машин, з графічними процесорами. AWS не дозволяє підключати графічні процесори до будь-якого або всіх типів віртуальних машин в своєму портфелі

Хмарне сховище даних поставляється AWS з різними варіантами, такими як динамічна зміна розміру, різні типи дисків (традиційний і SSD). На відміну від інших CSP, AWS не обмежує кількість операцій введення-виведення в секунду (IOPS) за розміром розділу. Ви можете отримати більше IOPS за додаткову плату навіть для невеликих дисків.

В області надання як сервіс обслуговуваниъ реляційних баз даних AWS підтримує керовані бази даних для MySQL, PostgreSQL, MariaDB, Oracle і MS SQL в рамках своєї пропозиції RDS. Крім того, AWS мають свої власні бази даних, сумісні з MySQL і PostgreSQL, які можуть похвалитися продуктивністю, подібною Oracle, але доступні за невелику плату. AWS вкладають в це значні кошти, а також анонсували версії баз даних Multi-Master і Serverless. Для баз даних NoSQL AWS випускає свій продукт DynamoDB вже близько п'яти років. Це розвиток їх бази даних SimpleDB. AWS є прихильником і надає ряд спеціалізованих NoSQL СУБД. До них відносяться DynamoDB (ключ-значення), Neptune (граф) і Elasticache (кешування для даних ключ-значення).

За останнє десятиліття AWS поліпшив свій портфель мережевих послуг. Почалося з VPC (Virtual Private Cloud) і пов'язаних із мережею, таких як групи безпеки, мережеві списки контролю доступу і шлюзи доступу. У той час користувачам все ще доводилося налаштовувати свої власні сервери трансляції мережевих адрес (NAT), файрволи та подібні вузли. AWS прислухалася до відгуків клієнтів і поступово додавала все необхідне в якості надаваних послуг мережі в свій портфель. Тепер AWS надає керований NAT-шлюз, VPN-шлюз, Transit-шлюз, шлюз прямого доступу і т.д. Нещодавно AWS також оголосили про надання послуги VPN-клієнта. Це усуває необхідність для клієнтів розгортати сервери OpenVPN для управління доступом до хмарним віртуальним машинам.

Також важливим фактором, для вибору AWS є наявність безкоштовного,

пробного облікового запису, яких надає змогу отримати доступ до більшості функція, звичайно, з певними обмеженнями. Це хороший варіант для навчання та розгортання тестових проектів, щоб виявити плюси та мінуси даної платформи.

Наступним гігантом в сфері надання хмарних сервісів виступає платформа Azure від усім відомої компанії Microsoft. Microsoft прийшов пізніше з помітним відставанням від AWS в наданні публічних хмарних сервісів і спочатку зосередився на пропозиціях SaaS і PaaS (програмне забезпечення як послуга і платформа як послуга), оскільки її сильні сторони лежать як в корпоративному, так і в споживчому ПЗ. Спочатку Microsoft орієнтувалася на служби PaaS для Azure і були сфокусовані на існуючій базі розробників якы використовують технології Microsoft. Згодом Microsoft розширила свою увагу і на сервіси Linux, і на надання IaaS (інфраструктура як послуга). Це також відображено в ребрендинг Windows Azure на Microsoft Azure, і на Microsoft Loves Linux. Згодом Microsoft також зробила Azure більш зручною і вбудувала підтримку API для різних сервісів. Однак, незважаючи на широкий спектр послуг, Microsoft істотно відстає від AWS в корпоративному впровадженні. Великі підприємства, в яких вже існують відносини з Microsoft, залишаються значною частиною користувальницької бази, хоча в Azure спостерігається стійке зростання річного доходу. Azure - це зріла хмарна платформа з широким набором функцій, яка може бути кращою платформою для клієнтів, які якимось чином вже використовують продукти Microsoft. Хоча Azure підтримує ряд сервісів на основі продуктів з відкритим вихідним кодом, портфоліо Microsoft в хмарі - це те, що виділяє його серед конкурентів.

Azure надає более 151 типів віртуальніх машин и більш ніє 26 сімейств віртуальніх машин, які підтрімують всі - від невеликих web-систем до високонавантажених HPC, Oracle и SAP. Azure має як Windows, так и кілька версій Linux (RHEL, CentOS, SUSE, Ubuntu). У Azure є окреме Сімейство примірніків для роботи з машинним навчання и штучним інтелектом (ML / AI)

Якщо вам потрібні високопродуктивні обчислення, що вимагають до 128 віртуальних процесорів і 3,5Tb оперативної пам'яті, то Azure хороший вибір. Якщо у вас є існуючі ліцензії для операційної системи Windows або MSSQL, то перенесення цих активів в хмару (BYOL) за допомогою програми Microsoft License Mobility Program хороший вибір і можливість. Вартість ліцензій становить значну частину витрат на інфраструктуру і буде основним фактором для клієнтів, які використовують великі розгортання СУБД MSSQL і інших продуктів.

Коли мова заходить про бази даних SQL і NoSQL, в Azure є досить непоганий набір сервісів. Azure забезпечує MSSQL Server і SQL Datawarehouse як сервіс. Azure також надає бази даних MySQL, PostgreSQL та MariaDB. Azure Table являє собою кероване сховище значень ключів, тоді як CosmosDB надає багатомодельну глобально розподілену базу даних NoSQL з декількома моделями узгодженості. Він надає API, сумісний з MongoDB, Cassandra, Gremlin (Graph) і сховищем таблиць Azure. Якщо вам потрібно запустити кілька моделей керованих даних, в тому числі моделі даних документа, графа, ключа-значення, таблиці і сімейства стовпців в одному хмарі, то Cosmos може бути підходящим варіантом. Кеш Azure для Redis доповнює пропозиції керованим кешем.

Хмарна платформа Google (GCP), незважаючи на пізній вихід і має найменшу частку ринку серед постачальників загальнодоступних хмарних обчислень в порівнянні з конкурентами (поточна частка ринку близько 4%), демонструє стійке відсоткове зростання. GCP може похизуватися кількома функціями, які дозволяють йому випереджати своїх конкурентів в певних областях. Ще один аспект в тому, що GCP користується популярністю не тільки у клієнтів, які вже є частиною екосистеми, але також і користувачів хмарних технологій, які прагнуть розширити свій ландшафт на Google в рамках стратегії використання декількох хмарних платформ. Google також почав з PaaS-сервісів, але неухильно розширює свій портфель продуктів.

З точки зору обчислень, Google має найменшу кількість типів віртуальних машин (28 типів по 4 категоріям), але тим не менш, GCP має одну особливість, яка робить ці цифри не такими важливими. Google дозволяє користувачам створювати свої власні нестандартні розміри (процесор, пам'ять), щоб клієнти могли порівнювати розміри своїх хмарних навантажень з їх розмірами на попередньому етапі. GCP також виставляють рахунки клієнтам, грунтуючись на загальному використовуваному процессорном часу і пам'яті, а не на окремих віртуальних машинах. Це зменшує для клієнтів втрати оплаченої невикористаної ємності.

Ще одна унікальна особливість полягає в тому, що GCP дозволяє підключати графічні процесори практично до всіх типів екземплярів віртуальних машин. Це дозволяє перетворити будь-який стандартний або користувальницький екземпляр в віртуальну машину, готову до машинного навчання. Google також є лідером в посекундної тарифікації, що змушує інших постачальників хмарних обчислень брати з них приклад. У порівнянні з нормою погодинної оплати, посекундна тарифікація значно знижує втрату оплаченої потужності. Це дозволяє клієнтам заощадити до 40% коштів у порівнянні зі стандартними розмірами віртуальних машин і виставленням рахунків за годину.

Час запуску віртуальної машини в GCP феноменально вище ніж в інших. Це робить масштабування особливо зручним. GCP також дозволяє динамічно змінювати розміри дисків, щоб вам не доводилося робити акробатику системного адміністрування при заповненні дисків. Продуктивність операцій введення-виведення (IOPS) призначаються на основі розмірів диска і не можуть бути надані окремо. Це може бути проблематично для клієнтів, яким потрібен високий IOPS для невеликого набору даних, і призводить до витрати коштів на непотрібне зберігання.

Для реляційних баз даних GCP забезпечує підтримку СУБД MySQL і PostgreSQL. Для клієнтів, яким потрібна глобально розподілена база даних, яка як і раніше підтримує негайну узгодженість і властивості ACID (Atomicity, Consistency, Isolation, Durability - Атомарність, Узгодженість, Ізольованість і

Довговічність), GCP створила Spanner. Spanner використовує узгоджені алгоритми і атомний годинник для синхронізації транзакцій між вузлами. Ця пропозиція є унікальним для GCP і робить Spanner дуже привабливим для великих корпоративних клієнтів, які пред'являють ці вимоги для свого сховища реляційних даних. Другою СУБД з відкритим вихідним кодом - CockroachDB, заснована на описі Spanner, опублікованій Google. З точки зору NoSQL, у GCP є продукт під назвою BigTable. BigTable - це керована база даних NoSQL з широким стовпцем в петабайтного масштабі, яка використовується Google у власних продуктах, таких як Gmail.

Незважаючи на те, що GCP може не володіти достатньою глибиною можливостей деяких інших постачальників хмарних послуг, в його портфелі є кілька унікальних продуктів, і він є привабливим варіантом, для деяких випадків.

Як детально описано вище, кожна хмара має особливості і переваги, які відповідають конкретним потребам клієнтів. У той час як всі хмарні провайдери будуть продовжувати надавати певні загальні послуги (такі як керована база даних MySQL), кожен буде продовжувати створювати унікальні, диференційовані послуги (наприклад, Aurora, Cosmos, Spanner), які спеціально призначені для вирішення вельми специфічних потреб клієнтів. CSP сподіваються, що це збільшить прихильність клієнтів їхніх послуг і створить бар'єр для переходу до конкурента. З точки зору клієнта, ці сервіси також стануть рушійною силою для прийняття мультихмарної стратегії. Наприклад, клієнт, можливо, захоче використовувати GCP для однієї програми, якому потрібні функції Spanner, в той час як він використовує AWS для своїх служб AI і Azure для певних завдань Windows. Навіть для таких перспективних послуг, як комп'ютерний зір і розпізнавання мови, потреби клієнтів можуть підштовхнути їх до змішування і зіставлення сервісів на хмарних платформах для задоволення вимог їх додатків. Клієнт, швидше за все, буде використовувати одну хмару в якості своєї основної платформи, а служби інших будуть використовуватися для конкретних додатків. Підбиваючи підсумок можна пердставими основні переваги та недоліки даних хмарних провайдерів у виглді таблиці.

Таблиця 1.3

Переваги та недоліки найпопулярніших хмарних провайдерів

Назва провайдера	Переваги	Недоліки
AWS	– Домінуюча позиція на ринку – Підтримка великих організацій – Доступність по усьому світі – Хороша документація – Безкоштовний пробний акаунт – Широкий набір сервісів та функцій	– Складність деяких сервісів
Azure	– Другий найбільший хмарний провайдер – Глубока інтеграція з продуктами Microsoft – Широкий набір – Стрімкий ріст та розвиток	– Деякі проблеми з документацією – Неповні інструменти керування
GCP	– Розроблений спеціально для хмарно орієнтованих підприємств – Прагнення до відкритості та мобільності – Прогресивна цінова політика та система знижок – Направленість на роботу з DevOps підходом	– Пізній вихід на ринок – Обмежений функціонал

Проаналізувавши усі вищеозвучені переваги та недоліки різних хмарних платформ я вирішив зупинитися саме на AWS, оскільки саме вона в найбільшій мірі задовольная усі сучасні потреби бізнесу та надає найширший функціонал та маштабованість

РОЗДІЛ 2. ПРОЕКТ ХМАРНОЇ ІНФРАСТРУТУРИ КОМПАНІЇ

2.1. Проектування серверної частини

В даній роботі була розглянута хмарна інфраструктура невеликої IT компанії, яка складається з двох серверів та сервісів розгорнутих на них. Першим сервером виступає віртуальна машина EC2 з операційної системою Windows Server 2016 на борту. Комп'ютер, на якому встановлена дана операційна система, може виконувати ролі файлового сервера, сервера служби веб-додатків, сервера терміналів, поштового сервера, сервера віддаленого доступу, служби DNS (доменних імен), служби каталогів, сервера потоків мультимедіа та інші. Дуже багато особливостей Windows Server дають можливість співробітникам організації працювати більш продуктивно. Файлові служби та служби друку дозволяють керувати файловими ресурсами. У той же час зберігається безпеку і доступ для користувачів. Служба каталогів в Windows Server називається Active Directory. Завдяки цій службі зберігаються відомості про мережеві об'єкти. Крім того Active Directory дозволяє адміністратору швидко знайти цю інформацію. Ця служба також спрощує роботу з проектування, розгортання та управління каталогами мережі. В оптимальній роботі Active Directory допоможе консоль управління груповими політиками. Windows Server в поєднанні з таким корисним, зручним та багатофункціональним інструментом як служба каталогів Active Directory надають дуже зручний та обширний функціонал для виконання задач по адмініструванню користувачів. Тому доцільно розгорнути AD на даному сервері, також були налаштовані служби DHCP, DNS, IIS з внутрішнім сайтом компанії, RDP доступ до сервера, а також розроблені політики безпеки та доступу, на основі груп безпеки в службі каталогів AD

Другим сервером виступиє віртуальна машина EC2 на базі операційної системи Linux. Оскільки, у плані продуктивності Linux показує себе набагато краще, ніж Windows тому було прийнято рішення використати саме дану операційну систему для розгортання веб-сервера Nginx з розміщеним на ньому

сайтом для роботи з клієнтами, крім того було розгорнуто власний поштови сервер компанії Postfix та базу даних MongoDB, усі сервіси на даній машині були розгорнуті в ізольованому середовищі Docker-контейнерів.

Docker - це програмна платформа для швидкої розробки, тестування і розгортання додатків. Docker упаковує ПО в стандартизовані блоки, які називаються контейнерами. Кожен контейнер включає все необхідне для роботи програми: бібліотеки, системні інструменти, код і середу виконання. Завдяки Docker можна швидко розгортати і масштабувати додатки в будь-якому середовищі і зберігати впевненість в тому, що код буде працювати. Використання Docker на AWS дає розробникам і системним адміністраторам надійний і економічний спосіб збирання, доставки і запуску розподілених додатків будь-якого масштабу. В основі роботи Docker лежить стандартизований спосіб виконання коду. Docker - це операційна система для контейнерів. Подібно до того як віртуальна машина створює віртуальне уявлення апаратного забезпечення сервера (тобто усуває необхідність безпосередньо управляти таким), контейнери створюють віртуальне уявлення серверної операційної системи. Після установки на кожен сервер Docker надає доступ до простих команд, необхідним для збірки, запуску або зупинки контейнерів. Використання Docker дозволяє швидше і ефективніше доставляти або переміщати код, стандартизує виконуються додатками операції і в цілому економить кошти, оптимізуючи використання ресурсів. Завдяки Docker користувачі отримують об'єкт, який з високою надійністю можна запускати на будь-якій платформі. Простий і зрозумілий синтаксис Docker забезпечує повний контроль над виконуваними операціями. Повсюдне впровадження контейнерів має на увазі доступ до різноманітних інструментів і готовим додатків, які можна використовувати з Docker. Дана технологія набула великої популярності в останні роки, більшість передових компаній використовує її в своїх проектах. Знання та досвід роботи з Docker контейнеризації значно підвищу продуктивність та швидкість розробки та стабільності інфраструктури, тому було прийнято рішення використовувати контейнери

Також, використання даної технології надає низку важливих переваг:

- Оптимізація ресурсів віртуальної машини
- Захищеність
- Однаковий спосіб розгортання будь-де
- Изольованість

Таблиця 2.1

Параметри віртуальних хмарних серверів

Назва сервера	Параметри
Linux_prod_server	Операційна система: CentOS
	Процессор: 2 vCPU
	Оперативна пам’ять : 2 Gb
	Дискова підсистема: 30 Gb / - 2 Gb /var – 10 Gb /home – 16 Gb Swap – 2 Gb
	Внутрішня IP-адреса: 192.168.0.7 Зовнішня IP-адреса: 52.53.158.136
Win_prod_server	Операційна система: Windows Server 2016
	Процессор: 2 vCPU
	Оперативна пам’ять : 4 Gb
	Дискова підсистема: 30 Gb C:\\ - 20 Gb D:\\ - 10 Gb
	Внутрішня IP-адреса: 192.168.0.6 Зовнішня IP-адреса: 54.193.170.129

Описаних вище параметрів вистачить для коректної та комфотної роботи усіх сервісів потрібних компанії. Також слід зазначити, що однією з головних переваг роботи з хмарою є простоста горизонтального та вертикального маштабування, тобто якщо настане той час, коли кількість виділених на початку ресурсів стане замало, можна легко додати віртуальних процесорів, оперативної пам’яті чи дискового простору до вже існуючих та працюючих машин, або просто створити нові та залучити їх, також можна зробити кластер віртуальних машин. Кластер забезпечить ще більшу доступність та безпеку даних, оскільки в разі виходу з ладу одного серверу він буде дуже швидко замінений на інший.

Після того як було визначено кількість та параметри серверів потрібно було вибрати спосіб їх розгортання. Звичайно, можна робити усе напряму «руками», але це достатньо довго, не дає бажану гнучкість та змушує адміністратора витрачати свій час на рутину яку можна та потрібно оптимізувати та автоматизувати. Тому було обрано підхід IaC (Infrastructure-as-code).

Модель «Інфраструктура як код (IaC)», яку іноді називають «програмованої інфраструктурою», - це модель, за якою процес налаштування інфраструктури аналогічний процесу програмування. По суті, вона поклала початок усунення кордонів між написанням додатків і створенням середовищ для цих додатків. Додатки можуть містити скрипти, які створюють свої власні віртуальні машини і керують ними. Це основа хмарних обчислень і невід'ємна частина DevOps.

Інфраструктура як код дозволяє керувати віртуальними машинами на програмному рівні. Це виключає необхідність ручного налаштування і оновлень для окремих компонентів обладнання. Інфраструктура стає надзвичайно "еластичною", тобто дуже легко відтворюється і масштабується. Один адміністратор може виконувати розгортання і управління як однієї, так і 1000 машинами, використовуючи один і той же набір коду. Серед гарантованих переваг інфраструктури як коду - швидкість, економічність і зменшення ризику.

Для вирішення задач розгортання хмарної інфраструктури було обрано утиліту Terraform. Terraform - це популярний інструмент компанії Hashicorp, який з'явився в 2014 році. Ця утиліта дозволяє керувати вашою хмарною інфраструктурою в парадигмі Infrastructure as a Code зрозумілій, та легко читаємій декларативній мові. Застосування даного інструменту забезпечує єдиний вид ресурсів і застосування практик роботи з кодом для управління інфраструктурою, які за довгий час вже розроблені співтовариством розробників. Terraform підтримує всі сучасні хмарні платформи, дозволяє безпечно і передбачувано змінювати інфраструктуру. При запуску Terraform читає код і, використовуючи представлені провайдерами хмарного сервісу плагіни, призводить інфраструктуру до описаного стану, здійснюючи необхідні виклики до API.

Мій проект повністю знаходиться в Amazon, розгорнуто на базі AWS-сервісів, і тому я пишу про застосування Terraforma саме в цьому ключі. Окремо зауважу, що він може застосовуватися не тільки для Amazon. Він дозволяє управляти всім, у чого є API. Крім цього можна керувати настройками VPC, IAM-політиками і ролями, або таблицями маршрутизації, сертифікатами, мережевими ACL. Також налаштуваннями web application firewall, S3-бакета, SQS-чергами - всім, що може використовувати наш сервіс в Amazon.

Важливою частиною Terraform є так звані провайдери. Провайдери - це плагіни для роботи з API того чи іншого сервісу. Існує понад 100 різноманітних сервісів з якими може взаємодіяти Terraform. У їх числі провайдери для Amazon, Google, DigitalOcean, VMware Vsphere, Docker, Azure, Github та інші. Серед цих плагінів можна знайти навіть ті які дозволяють керувати правилами Cisco ASA. В рамках одного Terraform проекту можеа використовувати різні провайдери і, відповідно, ресурси різних постачальників послуг або технології. Наприклад, ви можете керувати інфраструктурою в AWS, з зовнішнім DNS - від GoDaddy. А завтра ваша компанія купила стартап який хостився в DO або Azure. І поки ви вирішуєте мігрувати це в AWS чи ні, ви також можете це підтримувати за допомогою того ж інструменту!

Можна виділити 4 основні переваги використання Terraform:

- Terraform - це multiple cloud support утиліта. Тобто, якщо потрібно буде мігрувати інфрастуктуру з AWS, наприклад в GCP чи Azure. То дуже зручно, мати інструмент, який не буде жорстко прив'язаний до якогось конкретного хмарного сервісу.
- Terraform як утиліта знаходиться повністю під вашим контролем. Її можна встановити на ноутбук, на сервер, вона може бути легко вбудована в ваш пайплайн, який може бути зроблений на базі будь-якого інструменту. Такого як, наприклад, GitLab CI чи Jenkins.
- Terraform вміє і добре перевіряє стан вашої інфраструктури. Якщо ви використовуєте Terraform, навіть прогін плану вхолосту з цього ресурсу покаже вам, що є зміни в робочому середовищі. Коли Terraform переглядає ваш код, він паралельно з цим звертається до API хмарного провайдера, отримує від нього стан об'єктів і порівнює: «А зараз там те ж саме, що я робив до цього, про що я пам'ятаю?» Потім порівнює це з кодом, дивиться, що потрібно ще змінити. І, наприклад, якщо в його стан, в його пам'яті, і в вашому коді все однаково, але там є зміни - він вам покаже, і запропонує його повернути до старої версії. На мій погляд, теж дуже хороша властивість.
- І, нарешті, останнє - це модулі. Тобто шаблони, за допомогою яких Terraform дозволяє вам шаблонізувати вашу конфігурацію, створювати деякі темплейти, і тим самим писати менше коду, а також надає можливість його перевикористання

2.2. Проект груп та політик доступу

Оскільки для зручності та гнучкості керування користувачами, їх правами та політиками доступу, було створено домен Active Directory, який надає колосальні можливості для регулювання питань безпеки та доступу.

Групова політика - це інструмент, доступний для адміністраторів, які працюють з архітектурою Active Directory. Він дозволяє централізовано керувати налаштуваннями на клієнтських комп'ютерах і серверах, підключених до домену, а також забезпечує простий спосіб поширення програмного забезпечення. Наприклад, можна забезпечити застосування стандартного набору налаштувань і конфігурацій для груп користувачів або комп'ютерів в домені або за запитом. У всіх компаніях як правило є різні відділи, наприклад відділ системних адміністраторів, розробників, дизайнерів, кожному з відділу необхідний свій стандартний набір програмного забезпечення, їх робочі комп'ютери повинні бути налаштовані під спеціальні завдання і потреби. За допомогою групових політик можна створити набори налаштувань для конкретних груп користувачів в домені. За допомогою Active Directory GPO можна встановити і управляти окремими уніфікованими наборами налаштувань, конкретно для дизайнерів або розробників. Конфігурації для комп'ютерів або користувачів простіше і ефективніше, тому що розташовані в одному місці і не вимагають повтору на кожному комп'ютері

Існує два компонента групових політик - серверний компонент і клієнтський, тобто дана структура відноситься до архітектури "клієнт-сервер".

Серверний компонент - оснастка Microsoft Management Console (MMC), яка використовується для вказівки налаштувань групової політики. MMC може бути використана для створення політик для контролю і управління адміністративними шаблонами і налаштуваннями безпеки (скрипти, встановлення ПЗ та інше). Кожен з них називається розширенням і в свою чергу кожен з них має дочірнє розширення, яке дозволяє додавати нові компоненти або оновлювати існуючі без можливості піддати ризику всю політику.

Клієнтський компонент інтерпретує і застосовує налаштування групової політики для комп'ютерів користувачів або цільовим користувачам. Клієнтські розширення - це компоненти, які запущені на системі користувача і несуть відповідальність за інтерпретацію обробки і застосування в об'єкти групової політики. Для адміністрування GPO використовують Group Policy Management Console (GPMC) і Group Policy Management Editor.

Сценарії використання Active Directory GPO:

- Централізована настройка пакету програм Microsoft Office.
- Централізована настройка управлінням харчуванням комп'ютерів.
- Налаштування веб-браузерів і принтерів.
- Установка і оновлення ПЗ.
- Застосування певних правил в залежності від місця розташування користувача.
- Централізовані налаштування безпеки.
- Перенаправлення каталогів в межах домену.
- Налаштування прав доступу до додатків і системних програм

Для зручності та впорядкованості було створено каталог MainOffice в якому створено підкаталоги для користувачів в різних відділів. Такий підхід дозволяє більш єфективно керувати доступами користувачів, бачити та розуміти якмими правами вони повинні або не повинні володіти. Також це в деяких випадках дозволяє адміністраторам дещо стандартизувати набір прав для користувачів окремих підкаталогів.

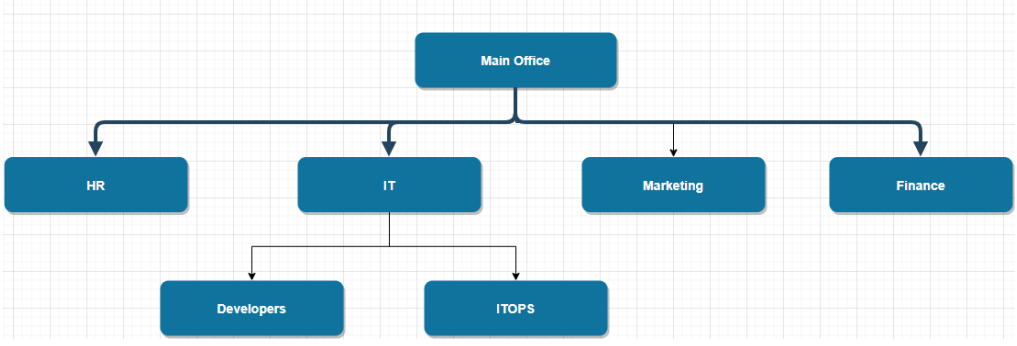


Рис. 2.1. Структура каталогів для зберігання та керування користувачами

Для організації доступу до документації, особистих справ, фінансових звітів та інших внутрішніх файлів було прийнято рішення створити директорію та піддиректорії в ній з усіма потрібними файлами, які зручно розміщені. Доступ до директорій, як і до інших ресурсів надається за помогою відповідних груп безпеки та налаштованими для них політиками.

Іменування груп для прозорості та полегчення адміністрування відбувається згідно стандарту, на приклад:

SEC_SRV_ESTUS_WIN_P01_SA – група надає своїм членам адміністраторський доступ до сервера з назвою ESTUS, який знаходиться під керуванням операційної системи Windows. Такий формат іменування надає можливість інтуітивно розуміти, що саме дозволено членам даної групи.

Як слід інтерпретувати назву, щоб отримати з неї максимальну кількість інформації:

1) Першій 3 літери SEC(Security) – означають, що це групи безпеки і вона надає деякі права. Також тут можуть бути літери DIS (Distribution) які вказують на те, що це відповідно групи розсилки. Такі групи зручно створювати для розсилки електронних листів працівникам одного відділу, одного проекту, однієї команди або навіть усім користувачам
2) Наступні 3 літери SRV(Server) – вказують на тип об’єкту до якого надається доступ, в даному випадку це сервер. Також це можуть бути, наприклад DB(Data base) бази даних або APP(Application) якщо це доступ до якогось додатку який синхронізується з доменом Active Directory в якому знаходиться користувач та група.
3) Далі йде просто назва серверу (ESTUS), бази даних (ESTUS-SQL), додатку (ESTUS-WEB-APP) чи ще чогось до чого надається доступ
4) Потім WIN(Windows) вказується операційна система, якщо це сервер. Яка встановлена на сервері, крім WIN(Windows) це такой може бути LNX(Linux), BSD(Free BSD) та інші. Якщо доступ надається до бази даних чи іншого об’єкту для якого операційну систему вказувати не доцільно, то цей параметр пропускається

5) Наступним вказується призначення та номер об’єкта P01(Production) – продакшн сервер вказує на те, що даний об'єкт використовується для роботи на ньому продукта (Найчастіше ПЗ компанії). Крім того це може бути T01 (Test), що вказує на те, що даний об’єкт належить до тестового середовища і використовується для тестування та виявлення багів продукту компанії перед запуском
6) І останні дві літери SA (Server administrator) означають тип доступу. Так SA надає доступ повний доступ до об’єкта, а RO(Read only) надає права без можливості внесення будь-яких змін

Крім того можна керувати доступами до ресурсів та сервісів компанії на рівні провайдера, який надає свої послуги. Так в Amazon сервіс AWS Identity and Access Management (IAM) надає можливості безпечного управління доступом до сервісів та ресурсів AWS. Використовуючи IAM, можна створювати користувачів AWS і групи, керувати ними, а також використовувати дозволи, щоб надавати або забороняти доступ до ресурсів AWS. IAM - це можливість аккаунта AWS, яка надається без додаткової оплати. Плата стягується лише за використання інших сервісів AWS, створених користувачами. Такий підхід надає ряд переваг:

1) Точне керування доступом до ресурсів AWS

   За допомогою IAM користувачі можуть керувати доступом до API сервісів AWS і конкретних ресурсів. IAM також дозволяє додавати особливі умови, при дотриманні яких користувач зможе використовувати AWS, наприклад час доби, вихідний IP-адреса, можливість використання протоколу SSL або необхідність використання пристрою багатофакторної аутентифікації.
2) Аналіз прав доступу

   За допомогою IAM можна аналізувати права доступу до сервісів середовища AWS. Команди по забезпеченню безпеки і адміністратори можуть швидко перевірити, що згідно з нашими правилами публічний і

міжаккаунтний доступ до ресурсів отримають тільки користувачі з відповідними правами. Також можна легко визначити і уточнити свої правила, щоб вони дозволяли доступ тільки до активних сервісів. Завдяки таким функціям буде простіше дотримуватися принцип мінімальних привілеїв

3) Багатофактора аутентифікація MFA(Multi-factor authentication)

   Можна захистити своє середовище AWS за допомогою AWS MFA, безкоштовної можливості забезпечення безпеки, яка доповнює такі дані для доступу, як ім'я користувача і пароль. MFA вимагає від користувачів довести фізичне володіння апаратним токеном MFA або зазначеним мобільним пристроєм MFA шляхом введення дійсного коду MFA

4) Інтеграція

   Сервіс IAM може надавати користувачам і додаткам федеративний доступ до Консолі управління і API сервісів AWS з використанням існуючих систем ідентифікації, таких як Microsoft Active Directory. Для цього можна використовувати будь-яке рішення для управління посвідченнями, яке підтримує стандарт SAML 2.0, або вибрати будь-який з наших зразків федерації (єдиний вхід (SSO) в Консоль управління AWS або федерацію API)

2.3. Розробка системи моніторингу

В реаліях сучасного світу інформаційних технологій жоден проект, сервіс, додаток і взагалі будь-що, що може вийти з ладу не може обійтися без системи моніторингу. Моніторинг IT систем є складовою частиною управління інформаційною інфраструктурою підприємства, що полягає в постійному спостереженні і періодичному аналізі IT об'єктів з відстеженням динаміки Ключовим завданням систем моніторингу IT є отримання, збереження та аналіз інформації про стан підконтрольних елементів IT структури компанії.

Спеціальна програма дозволяє оперативно відреагувати на проблему, що виникла в роботі IT сервісів, а також ефективно запобігати виникненню неполадок.

Бізнес потребує системи для моніторингу IT-інфраструктури, щоб забезпечити запуск і подальшу роботу необхідних йому мережевих систем і сервісів. Однак здійснення моніторингу різних складових частин IT-інфраструктури може стати для вас справжнім головним болем, якщо не вийде для цих цілей вибрати відповідне правильне рішення. Незалежно від масштабів керованої інфраструктури, будь вона невеликого розміру або рівня величезного підприємства, в будь-якому випадку не можна буде обійтися без надійного інструментарію для моніторингу. Навіть якщо ви просто власник персонального сайту, ви все одно маєте потребу в цілодобовому моніторингу доступності вашого ресурсу.

Існує безліч програмних продуктів, як комерційних, так і безкоштовних (з відкритим програмним кодом), які можуть допомогти здійснювати моніторинг IT інфраструктури та повідомляти про будь-які збої в роботі. З огляду на велику кількість пропозицій на ринку, не просто знайти потрібний варіант, який впишеться ціновий діапазон. Хороші новини для багатьох із нас полягають в тому, що на ринку доступні потужні рішення для моніторингу IT-інфраструктури з відкритим програмним кодом. Завдяки суспільству розробників проектів в відкритим програмним кодом, кожен охочий може використовувати потужні, багатофункціональні та гнучкі рішення для свої проектів абсолютно безкоштовно.

З усього різноманіття інструментів моніторингу було обрано зв’язку Prometheus + Grafana, оскільки вони являють собою популярні та стрімко ростучі рішення, які надають широкий набір функцій, який покриває дозволяє вирішувати абсолютну більшість завдань пов’язаних з моніторингом та візуалізацією даних.

Prometheus - популярний проект з відкритим вихідних кодом, велика частина компонентів якого написана на Golang, а частина - на Ruby. Це означає, що у нього є всього один бінарний файл, який потрібно завантажити і запустити разом з компонентами Prometheus. Prometheus повністю сумісний з Docker і доступний на Docker Hub.

Prometheus має центральний компонент, так званий Prometheus Server. Його основне завдання - зберігати і моніторити певні об'єкти. Об'єктом може стати що завгодно: Linux-сервер, сервер Apache, один з процесів, сервер бази даних або будь-який інший компонент системи, яку потрібно контролювати. У термінах Prometheus головна служба моніторингу називається сервером Prometheus, а об'єкти моніторингу - цільовими об'єктами. Як було сказано раніше, цільовим об'єктом може бути один сервер, або цільові об'єкти для перевірки кінцевих точок через HTTP, HTTPS, DNS, TCP і ICMP (* Black-Box Exporter), або проста кінцева точка HTTP, яку видає додаток. Через кінцеву точку HTTP сервер Prometheus перевіряє статус додатка. Кожен елемент цільового об'єкта, який треба хочете моніторити (статус центрального процесора, пам'ять або будь-який інший елемент), називається метрикою. Таким чином, Prometheus збирає через HTTP метрики цільових об'єктів, зберігає їх локально або віддалено і відображає. Сервер Prometheus зчитує цільові об'єкти з інтервалом, який визначаєте на збір метрик, і зберігає їх в базі даних часових рядів. Цільові об'єкти і часовий інтервал зчитування метрик задаються в файлі конфігурації prometheus.yml. Для запиту у бази даних часових рядів Prometheus інформацію про місце зберігання метрик, використовуюється мову запитів PromQL. Іншими словами, за допомогою PromQL йде запит до сервера Prometheus показати статус конкретного цільового об'єкта в даний момент часу і відповідь у вигляді метрики

Prometheus надає клієнтські бібліотеки на декількох мовах програмування, які можна використовувати для забезпечення працездатності програми. Але Prometheus - це не тільки моніторинг додатків. Є опція використання експортерів (exporters) для моніторингу сторонніх систем (таких як сервер Linux, демон MySQL і т.д.). Експортер - частина програмного забезпечення, яка отримує

існуючі метрики від сторонньої системи і експортує їх в формат, зрозумілий серверу Prometheus.

В якості стороннього компонента для візуалізації метрик, що зберігаються в базі даних часових рядів Prometheus, було використано Grafana. Замість того щоб писати запити PromQL безпосередньо на сервер Prometheus, використовується дошка графічного інтерфейсу Grafana для запиту метрик з сервера Prometheus і візуалізації їх на панелі моніторингу Grafan.

Prometheus має компонент управління оповіщенням, званий AlertManager. Він служить для запуска сповіщень через Email, Slack або інші клієнтські повідомлення.

Правила оповіщення визначаються в файлі під назвою alert.rules. Наприклад, якщо сервер Prometheus знайде значення метрики, що перевищує поріг, який визначено в файлі alert.rules, Alert Manager розішле оповіщення.

Prometheus, Grafana і Node Exporter є основою системи моніторингу Prometheus. Для налаштування моніторингу потрібен центральний сервер Prometheus, цільовий об'єкт і шар візуалізації.

Отже з основних переваг Prometheus можна виділити наступні:

- Сучасна архітектура зберігання даних і відносно низька ступінь використання дискової підсистеми
- Оптимізована мова запитів, що дозволяє більш зручно працювати з даними
- Безмежні можливості по створенню і кастомізації дашбордів, їх сортуванню і розташуванню
- Широкі можливості по налаштуванню часових зрізів і застосуванню різних функцій прямо на візуалізації

Grafana - це платформа з відкритим програмним кодом для візуалізації, моніторингу та аналізу даних. Цей інструмент, в поєднанні з Prometheus, - частина нашої двосторонньої системи моніторингу поведінки користувачів і продуктивності системи. Grafana дозволяє користувачам створювати дашборда з панелями, кожна з яких відображає певні показники протягом встановленого

періоду часу. Кожен дашборда універсальний, тому його можна налаштувати для конкретного проекту або з урахуванням будь-яких потреб розробки та / або бізнесу.Найчастіше Grafana використовується в поєднанні з Elasticsearch і InfluxDB, але це програмний засіб підтримує безліч інших джерел даних (Prometheus, MySQL, Postgres і т. д.). Для кожного джерела даних в Grafana передбачений настроюється редактор запитів і спеціальний синтаксис. Головна ціль використання Grafana, отримати добре структуровану інформацію з чітко відслідкованою динамікою у вигляді зрозумілих графіків, шкал навантажень, таблиць та інших.

Основні компоненти Grafana:

- Панель - базовий елемент візуалізації обраних показників. Grafana підтримує панелі з графіками, одиничними статусами, таблицями, тепловими картами кліків і довільним текстом, а також інтеграцію з офіційними і створеними спільнотою плагінами (наприклад, карта світу або годинник) і додатками, які також можна візуалізувати. Можна налаштувати стиль і формат кожної панелі; всі панелі можна перетягувати на нове місце, перебудовувати і змінювати їх розмір.
- Дашборд - набір окремих панелей, розміщених в сітці з набором змінних (наприклад, ім'я сервера, додатка і датчика). Змінюючи змінні, можна перемикати дані, які відображаються на дашборда (наприклад, дані з двох окремих серверів). Усі дашборди можна налаштовувати, а також секціонувати і фрагментувати представлені в них дані відповідно до потреб користувача. У проекті Grafana бере участь велика спільнота розробників коду і користувачів, тому існує великий вибір готових дашбордів для різних типів даних і джерел.
- У дашбордах можна використовувати анотації для відображення певних подій на різних панелях. Анотації додаються налаштованим запитами в Prometheus; на графіку анотація відображається вертикальною червоною лінією. При наведенні курсора на анотацію можна отримати опис події і теги, наприклад, для відстеження відповіді сервера з кодом

помилки 5xx або перезапуску системи. Завдяки цьому можна легко зіставити час, конкретну подію і його наслідки в додатку і дослідити поведінку системи

Дашборды Grafana також деяким чином допомагають в налагодженні додатків. Якщо кінцевий клієнт повідомляє про проблему, Grafana дає можливість відрізнити помилки на стороні клієнта/сервера від реальних помилок або проблем в логіці програми. Можна відстежувати всі веб-запити, ініційовані клієнтом (використовуючи адресу електронної пошти), адміністраторами додатків і самим додатком протягом заданого періоду часу і методом виключення знаходити причну.

Звісно можна проводити корегування файлів конфігурації і виправляти помилки, якщо помічено на дашбордах аномалії в графіках завантаження і продуктивності додатка. Наступний приклад графіка Grafana показує час відгуку на веб-запити протягом певного періоду часу. Для кожного веб-запиту було відстежено максимальни, мінімальний та середній час відгуку. Якщо видно запит, на обробку якого потрібно надто багато часу, масштабується певна частину графіка і досліджується проблема.

Grafana має вбудований механізм оповіщення (наприклад, по електронній пошті або за допомогою повідомлень через Slack) відповідно до певних правил. Ця можливість інуструменту не використовується, оскільки сповіщення будуть доставлятися з Alertmanager Prometheus. Однак деякі проблеми з продуктивністю системи можна побачити тільки після прогону програми, наприклад, незвично довгий час відгуку на веб-запит. Такє повідомлення не буде отримано від Prometheus, але аномалія буде чітко видна на графіку Grafana. Отже, обидва інструменту доповнюють один одного, коли з’являється якась проблема: на високому рівні вона перевіряється в Grafana, щоб зрозуміти, що сталося і чому, а потім копаємо глибше за допомогою Prometheus, використовуючи конкретний ідентифікатор запиту.

На відміну від Prometheus, який використовується як для додатків які тільки розроблюються так і для робочих, Grafana застосовується тільки для використовуваних додатків. Єдиним винятком, коли Grafana використовується для додатка, яке все ще на етапі розробки, є тестування продуктивності. При емулюванні завантаження системи за допомогою JMeter, потім перевіряються дашборди Grafana, щоб побачити, як система реагує.

Крім цілей відстеження продуктивності і налагодження, дашборда Grafana є потужним інструментом для прийняття обґрунтованих бізнес-рішень. При правильному налаштуванні (бажано в тандемі з сервісом Google Analytics) Grafana може візуалізувати аналітику поведінки користувача в системі у вигляді діаграм, гістограм часу і інших графічних елементів. На їх основі, зацікавлені в продукті сторони, можуть приймати рішення про подальше масштабування додатку, додаванні або видаленні деяких функцій і поліпшення циклу взаємодії з клієнтами.

РОЗДІЛ 3. РОЗГОРТАННЯ, ІНТЕГРАЦІЯ ТА ПЕРЕВІРКА КОРЕКТНОСТІ РОБОТИ ІНФРАСТРУКТУРИ

3.1. Розгортання та перевірка доступності серверів

В даному пункті розглянуто створення окремого мережевлшл середовища для роботи серверів, так зване VPC (Virtual private cloud)[1]. Це свого роду локальна мережа з певним пулом адрес, маскою підмережі, власною таблицею маршрутизації, DHCP сервером, шлюзом для доступа в інтернет IGW (Internet gateway) та списками доступую

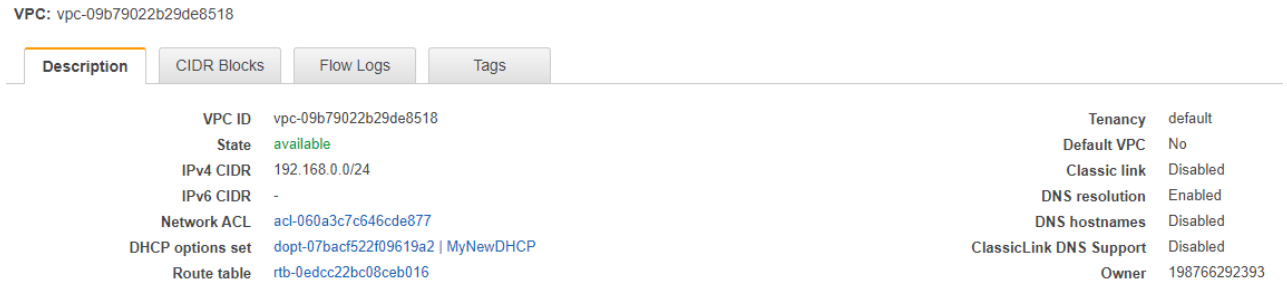


Рис. 3.1. Створення мережевого середовища

Після створення мережевого середовища було проведено розгортання самих серверів, за допомогою утиліти Terraform, в конфігураційному файлі було описано характеристики, зони розташування та імена серверів. Конфігураційний файл буде представлений у додатках під назвою Servers.tf. Варто сказати, що сервери було створені на базі EC2(Elastick cloud computer), які по суті являють собою віртуальні машини з встановленими на них операційними системами та деяким набором програмних продуктів

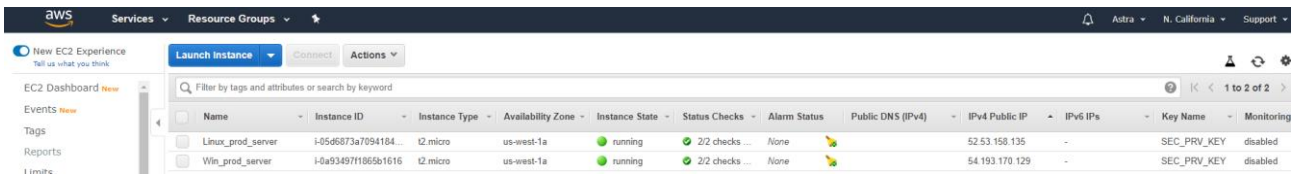


Рис. 3.2. Результат розгортання

Існують «білі» (публічні) та «сірі» (приватні) IP-адреси. Публічні адреси маршрутизуються в глобальній мережі на пряму та є загально доступними для усіх вузлів інтернету. Приватні використовуються для роботи локальних

мережевих середовищ, існуючи тільки в межах цих середовищ, як результат такі адреси можуть повторюватися також ці адреси не доступні з інтернету та не мають доступу до нього напряму. При розгортанні серверам було надано як публічні, для віддаленого доступу, так і приватні IP-адреси для того щоб вони могли могли обмінюватися даними між собою, в рамках локальної мережі. Для передбачуваності та спрощення налаштування серверам було надано статичні приватні IP-адреси.

Після того як сервери були створені, очевидно, до них треба якимось чином отримати доступ. Для машини на базі Windows server 2016 використовується протокол RDP, а для Linux відповідно SSH. При розгортанні серверів створюється приватний ключ, за допомогою якого відбувається аутентифікація при використанні SSH, а також цей ключ використовується для генерація первинного паролю адміністратора для роботи по RDP.



Рис.3.3. Генерація паролю для першого підключення

У більшість програмних засобів, що надають доступ по SSH достатньо у налаштуваннях сессії вказати шлях знаходження приватного ключа і цього буде достатньо для вдалого підключення.

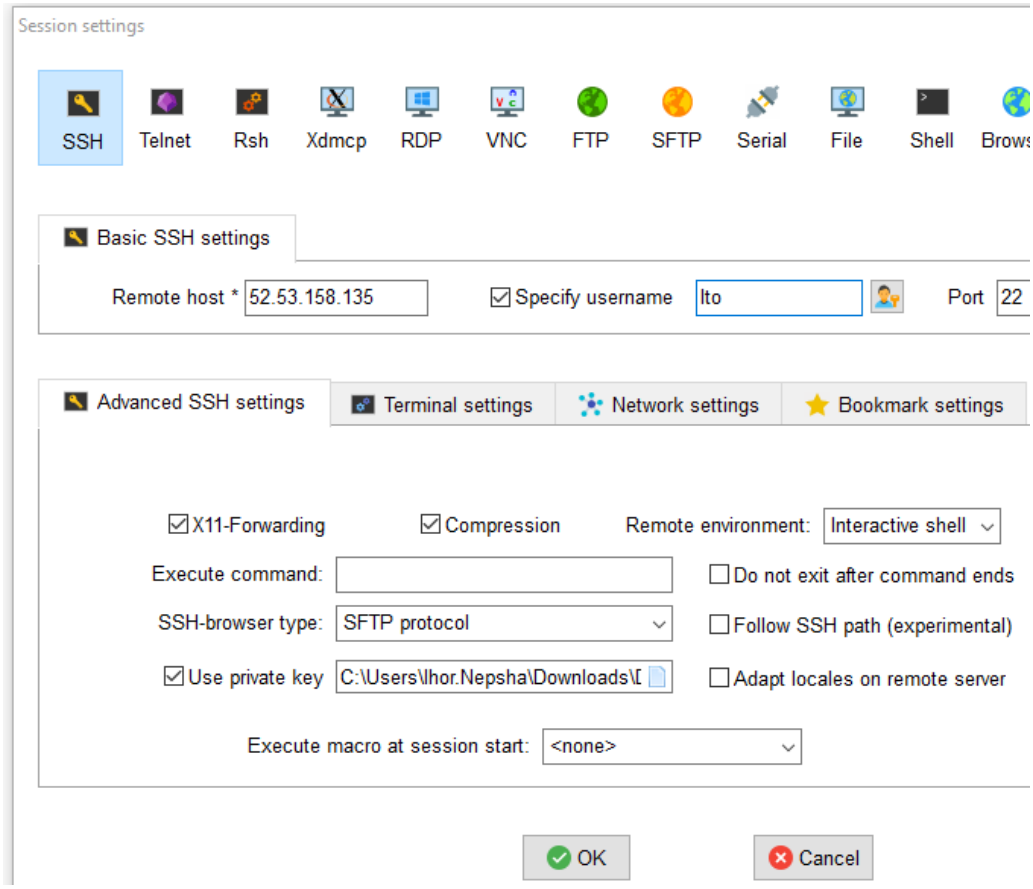


Рис.3.4. Приклад налаштувань сессії для підключення по SSH

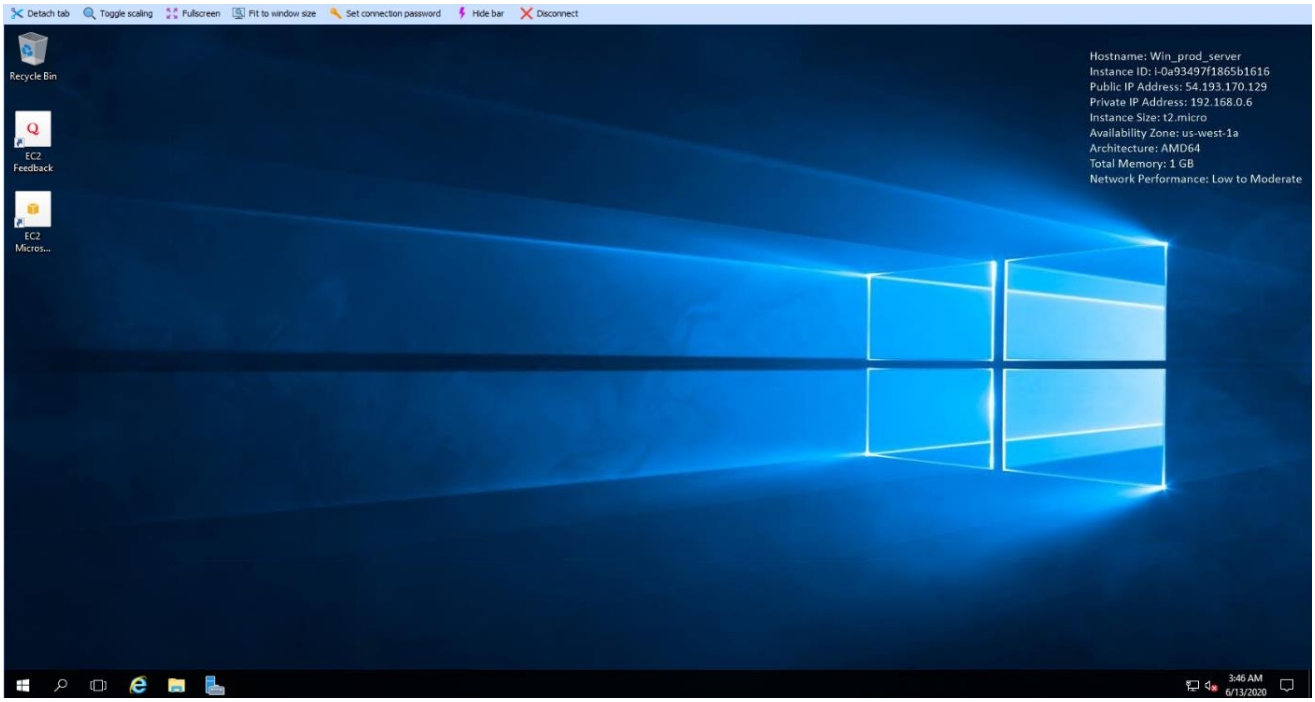


Рис.3.5. Результат першого підключення до сервера з Windows

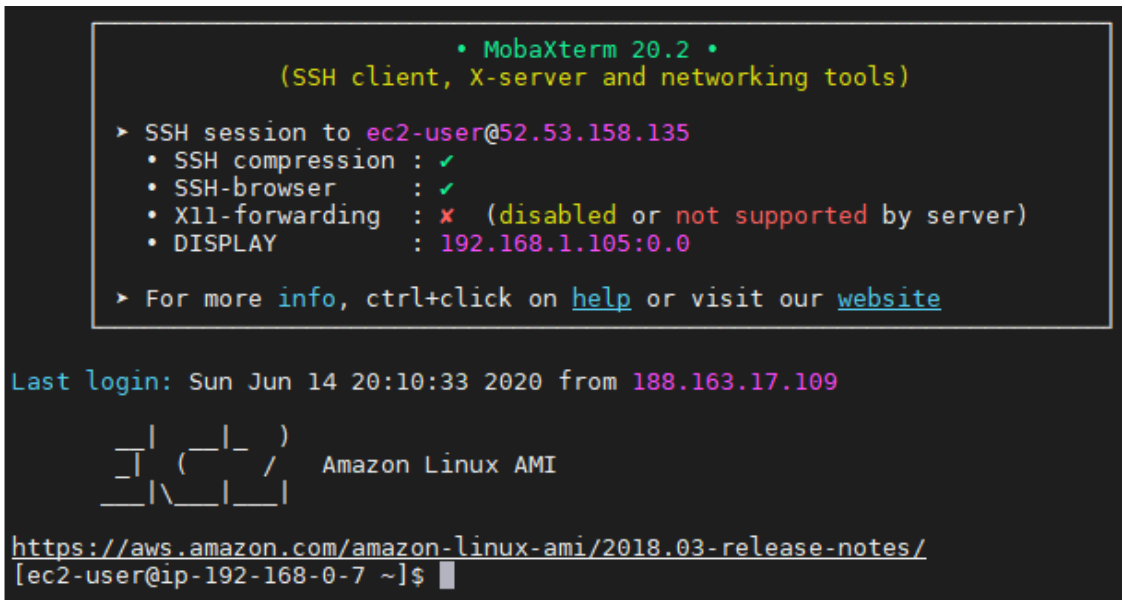
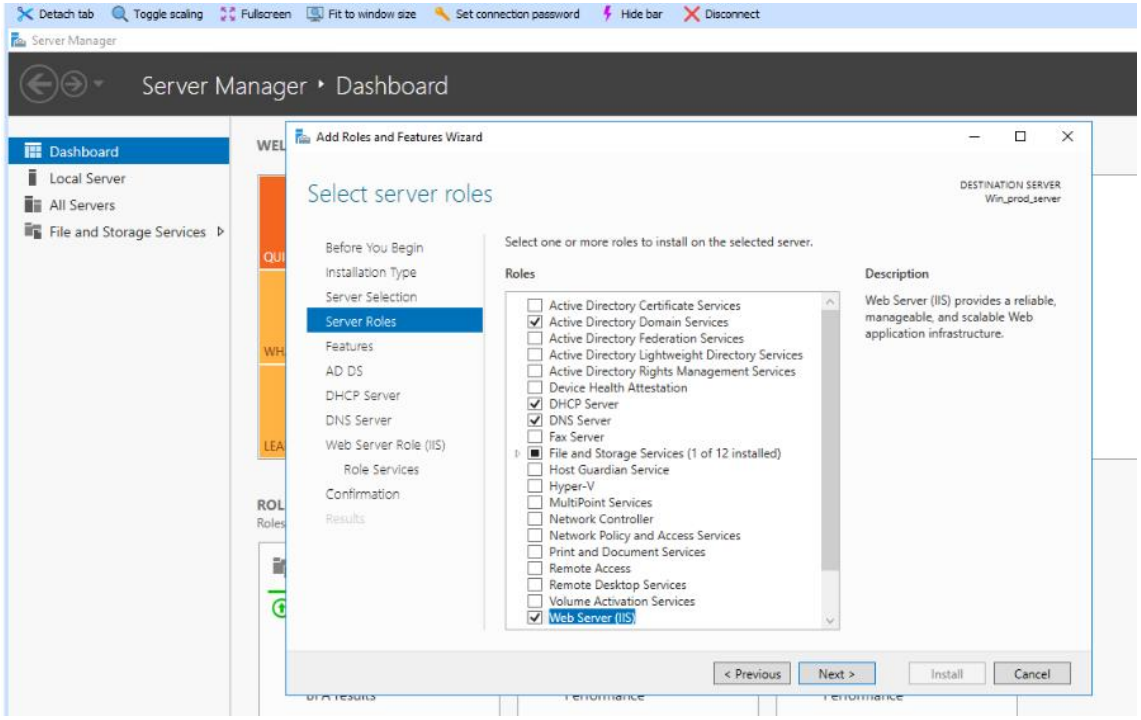


Рис.3.6. Результат першого підключення до сервера з Linux

3.2. Встановлення та конфігурування основних компонентів Windows Server

Операційна система Windows Server 2016 має широкий спектр сервісів та служб, які є життєво необхідними для будь якої інфрастркуктури[3]. В даній роботі на сервері були розгорнуті такі служби як Active Directory, для створення домену та зручності керування усіма об’єктамив ньому, будь то користувачі, компьютери, принтери і т.д. Також було піднято веб сервер IIS на якому розміщено внутрішній сайт компанії, де користувачі можуть переглядати інформацію, яка не повинна бути загальнодоступною. Машина виступає контроллером домену, а також містить у собі DNS та DHCP. Ще однією важливою складовою є служба KDC (Key distribution center), яка за допомогою протоколу Kerberos дає змогу додавати Linux машини у домен Active Directory[6]. Також було розроблено політики безпеки та доступу, створено окремий OU (Organization unit), який буде містити в собі аккаунти користувачів, що розбиті по відповидним підкатегоріям, групи безпеки, яки надаюсь доступ до тих чи інших ресурсів, а ще сервісні аккаунти, які будуть використовуватися для доступу до якихось специфічних сервісів



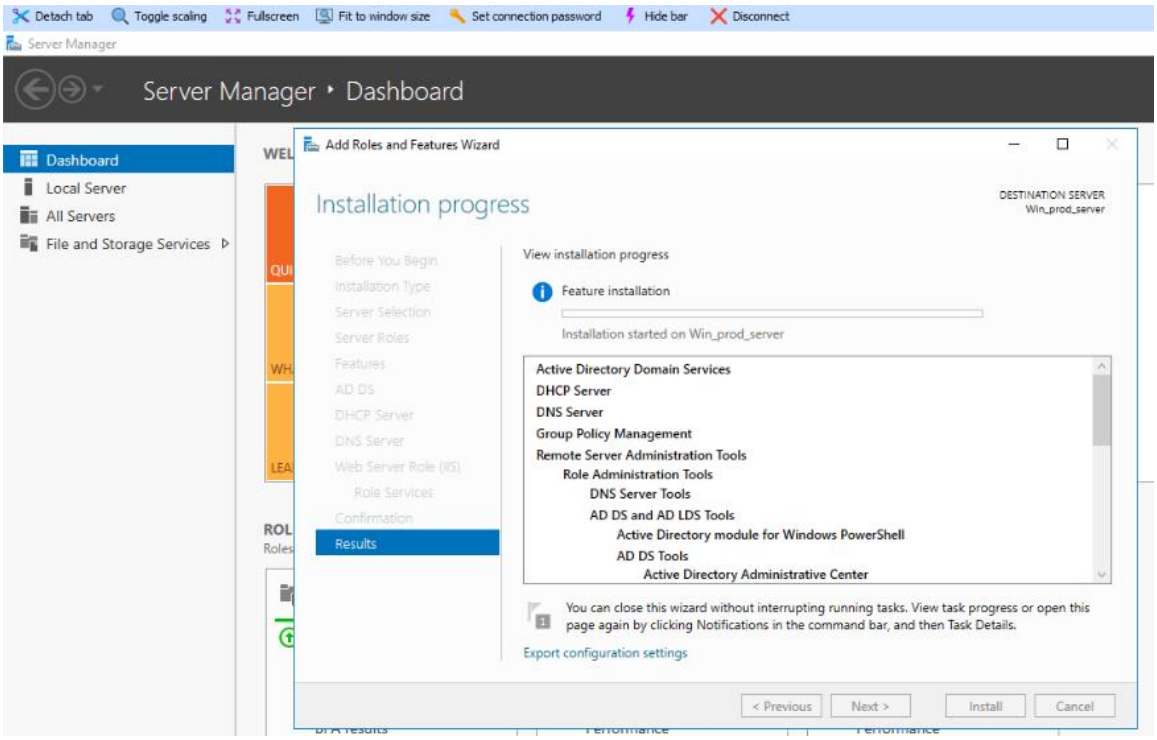
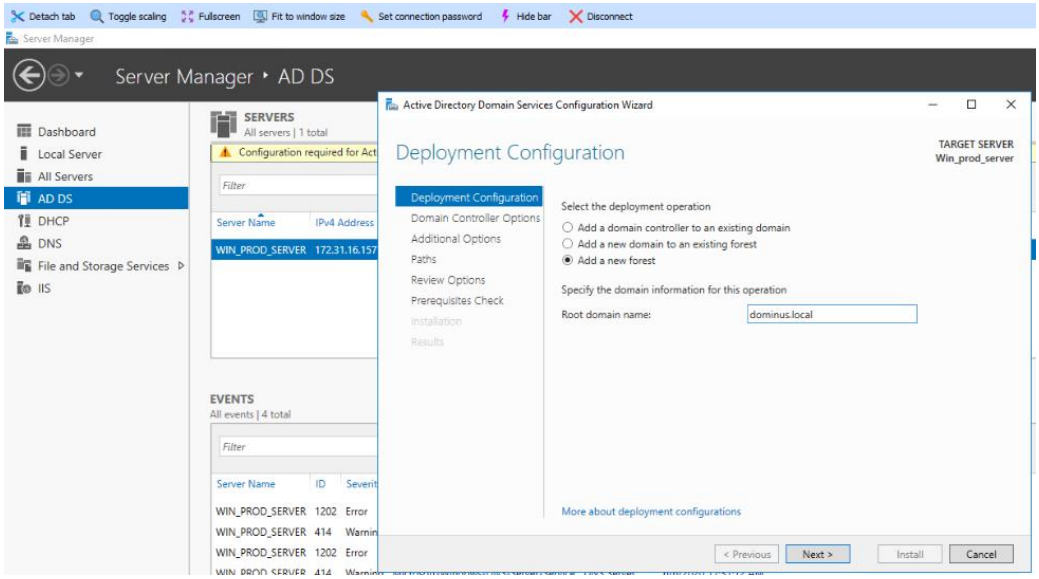
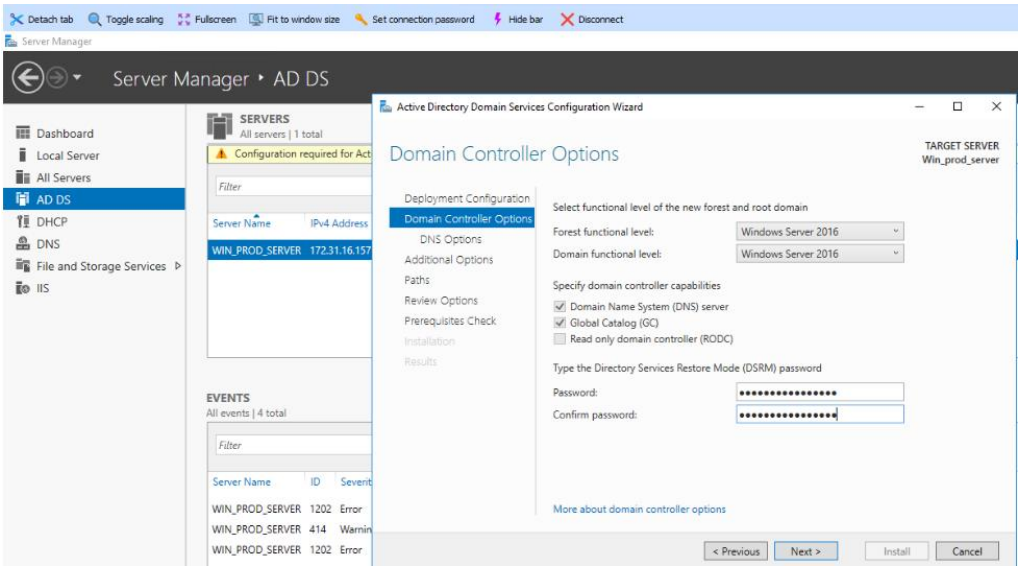
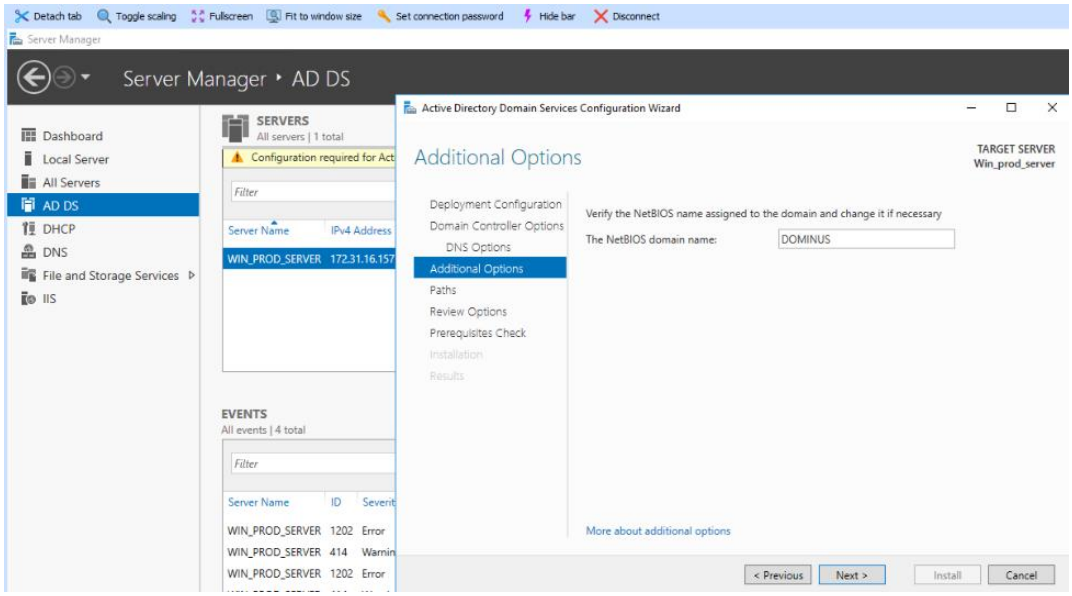


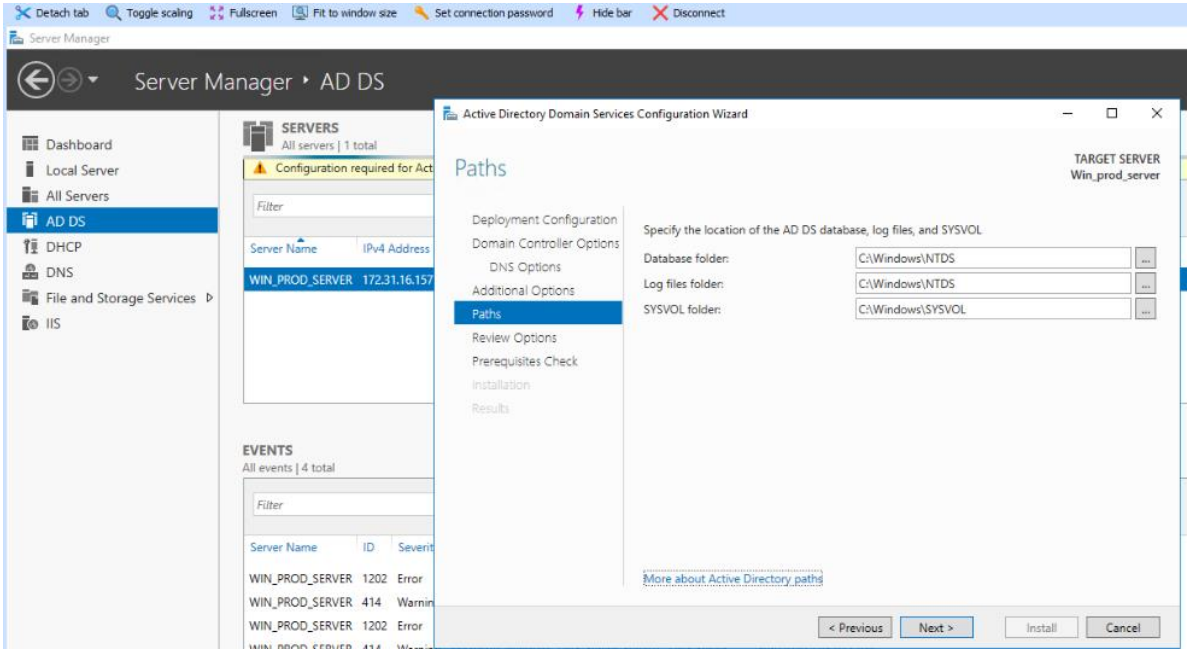
Рис.3.6-3.8. Встановленення необхідних програмних компонентів для Windows Server

Після встановлення необхідних програмних засобів, було проведено їх налаштування


Server Manager ▸ AD DS
Active Directory Domain Services Configuration Wizard
Deployment Configuration
TARGET SERVER
Win_prod_server
Deployment Configuration
Domain Controller Options
Additional Options
Paths
Review Options
Prerequisites Check
Installation
Results
Select the deployment operation
Add a domain controller to an existing domain
Add a new domain to an existing forest
Add a new forest
Specify the domain information for this operation
Root domain name:
dominus.local
More about deployment configurations
< Previous
Next >
Install
Cancel
Dashboard
Local Server
All Servers
AD DS
DHCP
DNS
File and Storage Services
IIS
SERVERS
All servers | 1 total
Configuration required for Act
Filter
Server Name
IPv4 Address
WIN_PROD_SERVER
172.31.16.157
EVENTS
All events | 4 total
Server Name
ID
WIN_PROD_SERVER 1202 Error
WIN_PROD_SERVER 414
WIN_PROD_SERVER 1202 Error


Server Manager ▸ AD DS
Active Directory Domain Services Configuration Wizard
Domain Controller Options
TARGET SERVER
Win_prod_server
Deployment Configuration
Domain Controller Options
DNS Options
Additional Options
Paths
Review Options
Prerequisites Check
Installation
Results
Select functional level of the new forest and root domain
Forest functional level:
Windows Server 2016
Domain functional level:
Windows Server 2016
Specify domain controller capabilities
Domain Name System (DNS) server
Global Catalog (GC)
Read only domain controller (RODC)
Type the Directory Services Restore Mode (DSRM) password
Password:
Confirm password:
More about domain controller options
< Previous
Next >
Install
Cancel
Dashboard
Local Server
All Servers
AD DS
DHCP
DNS
File and Storage Services
IIS
SERVERS
All servers | 1 total
Configuration required for Act
Filter
Server Name
IPv4 Address
WIN_PROD_SERVER
172.31.16.157
EVENTS
All events | 4 total
Server Name
ID
WIN_PROD_SERVER 1202 Error
WIN_PROD_SERVER 414
WIN_PROD_SERVER 1202 Error


Server Manager ▸ AD DS
Active Directory Domain Services Configuration Wizard
Additional Options
TARGET SERVER
Win_prod_server
Deployment Configuration
Domain Controller Options
DNS Options
Additional Options
Paths
Review Options
Prerequisites Check
Installation
Results
Verify the NetBIOS name assigned to the domain and change it if necessary
The NetBIOS domain name:
DOMINUS
More about additional options
< Previous
Next >
Install
Cancel
Dashboard
Local Server
All Servers
AD DS
DHCP
DNS
File and Storage Services
IIS
SERVERS
All servers | 1 total
Configuration required for Act
Filter
Server Name
IPv4 Address
WIN_PROD_SERVER
172.31.16.157
EVENTS
All events | 4 total
Server Name
ID
WIN_PROD_SERVER 1202 Error
WIN_PROD_SERVER 414
WIN_PROD_SERVER 1202 Error



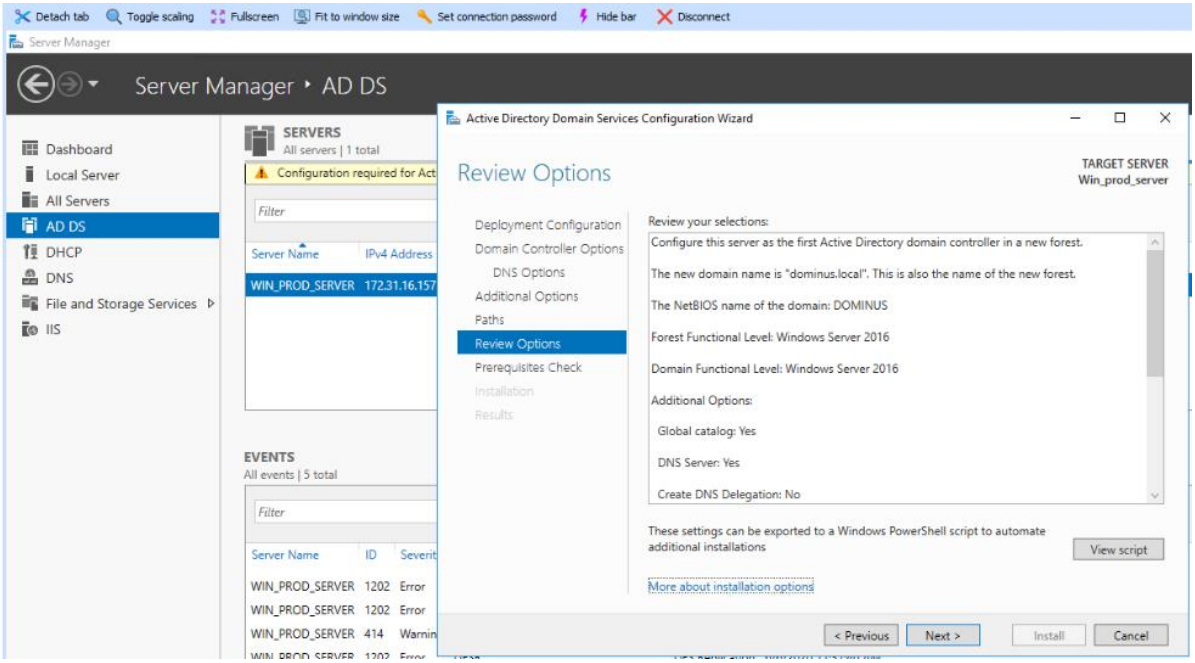


Рис. 3.9.-3.13. Створення домену та ініціалізація контроллера

Після створення домену було каталог Domain Objects, а підкаталоги для користувачів, груп та сервісних акаунтів. В підкаталозі Users створено ще додаткові каталоги, відповідно до їх сфер роботи. Наприклад є підкаталог для розробників (Developers), менеджерів (Managers), фінансового відділу (Finance) та інших. На рисунку 3.14 представлена структура каталогу Domain Objcets

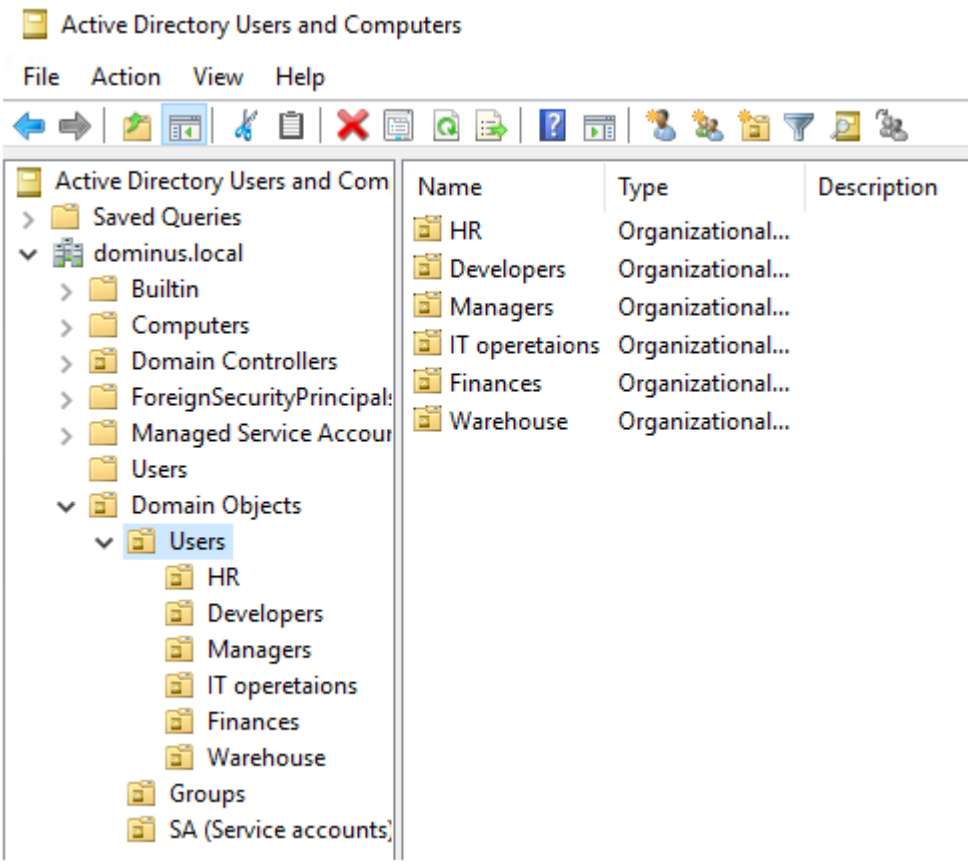


Рис. 3.14. Структура каталогу Domain Objects

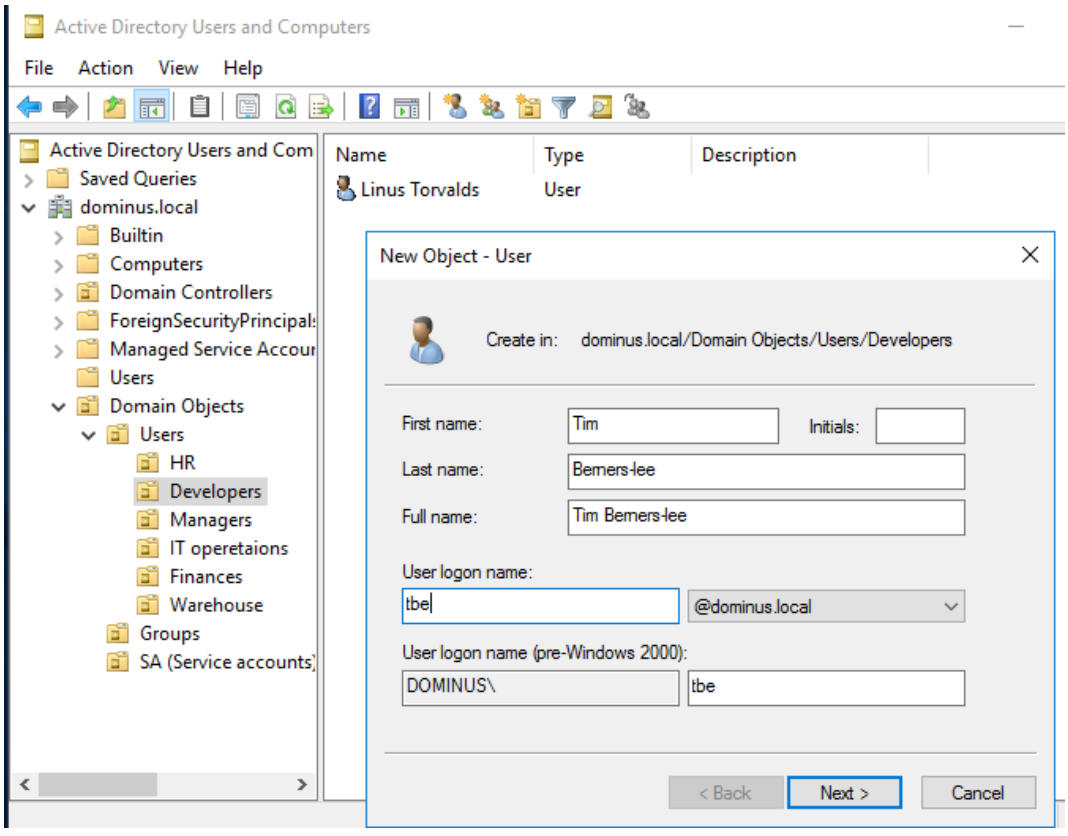


Рис. 3.15. Приклад створення користувача домену

Для зручності логіни усіх користувачів, окрім адміністратора, представляють набір з 3 букв, де перша це буква з якої починається ім’я користувача,а дві інші, перші букви прізвища

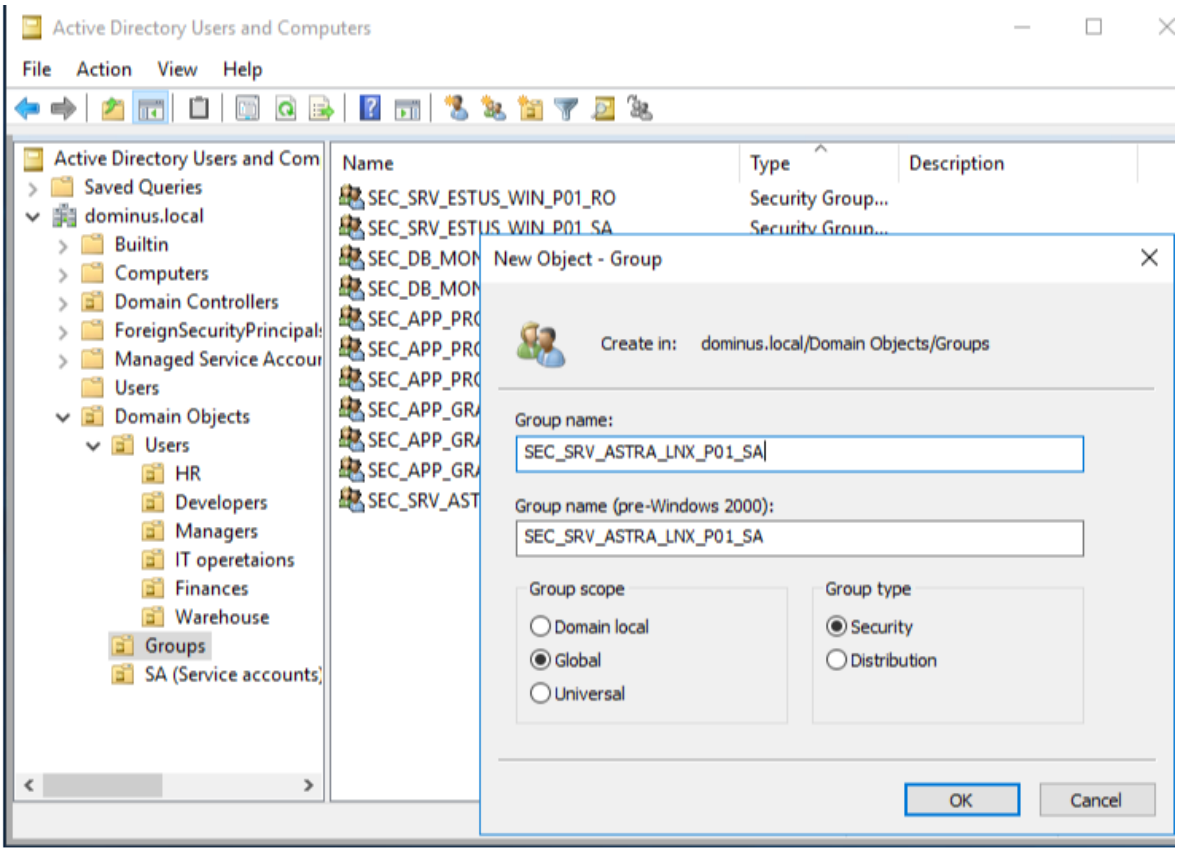


Рис. 3.16. Приклад створення групи безпеки

Далі буде наведено приклад налагодження DHCP серверу

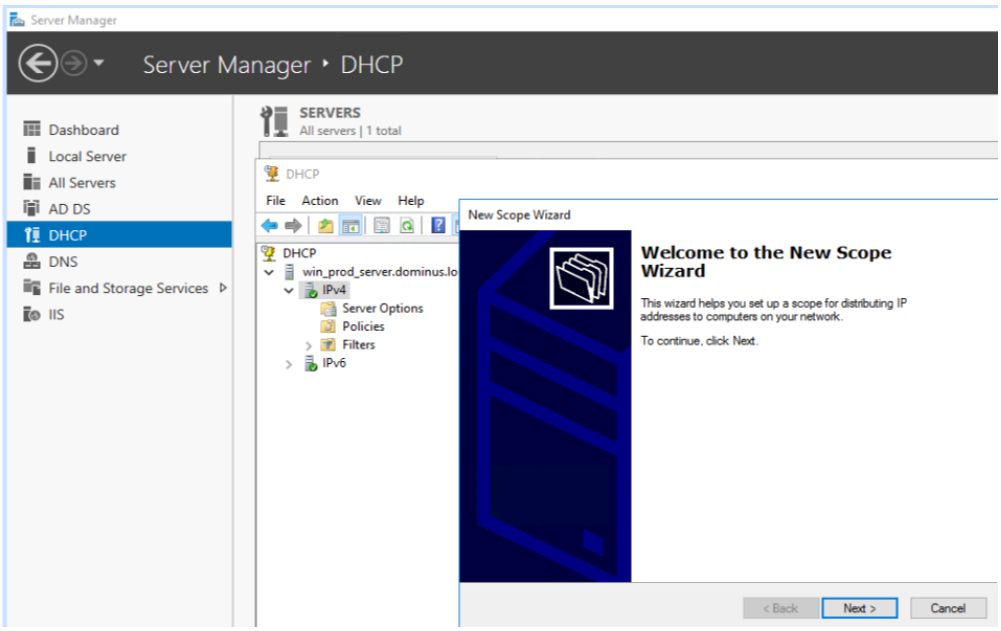


Рис. 3.17. Початок налаштування

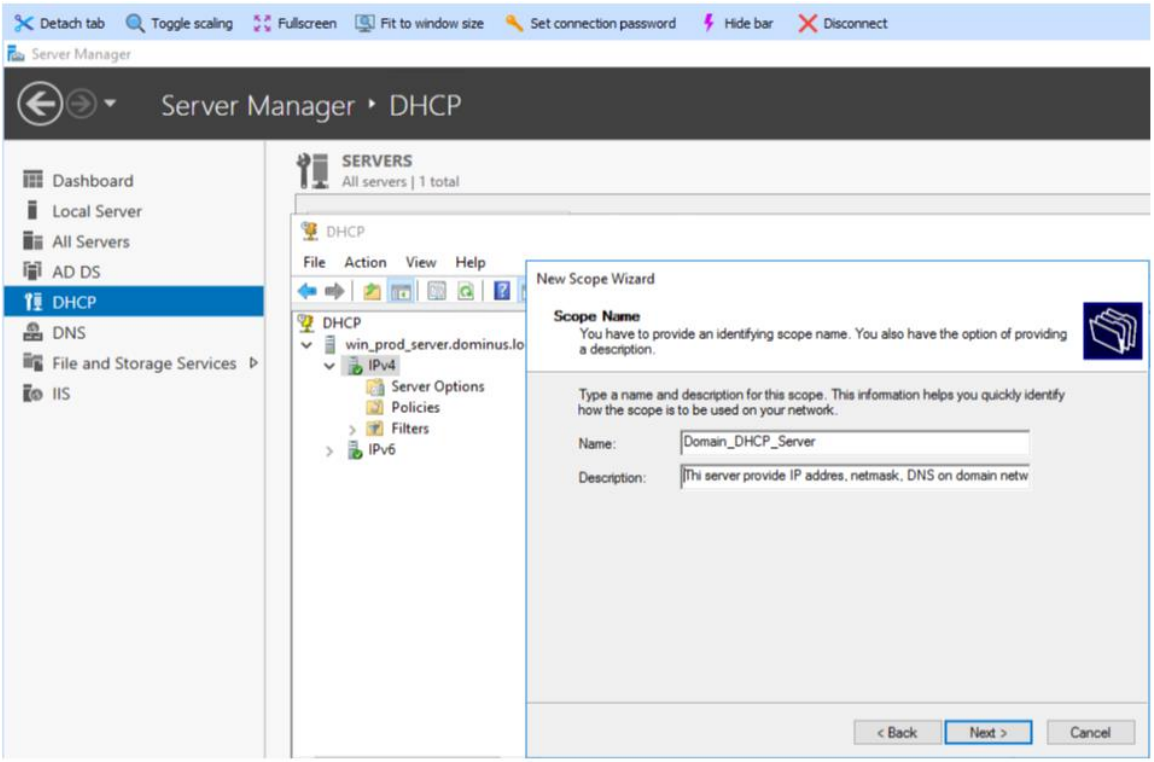


Рис. 3.18. Встановлення імені та опису

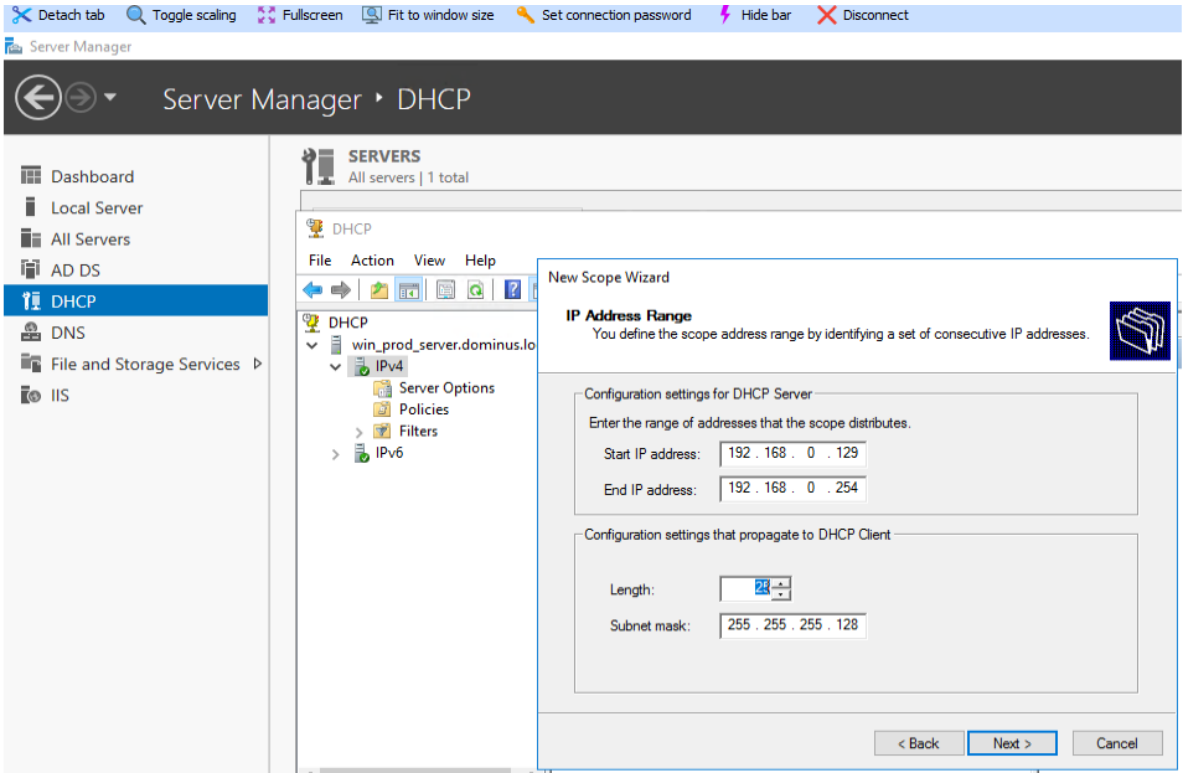


Рис. 3.19. Визначення параметрів DHCP пулу

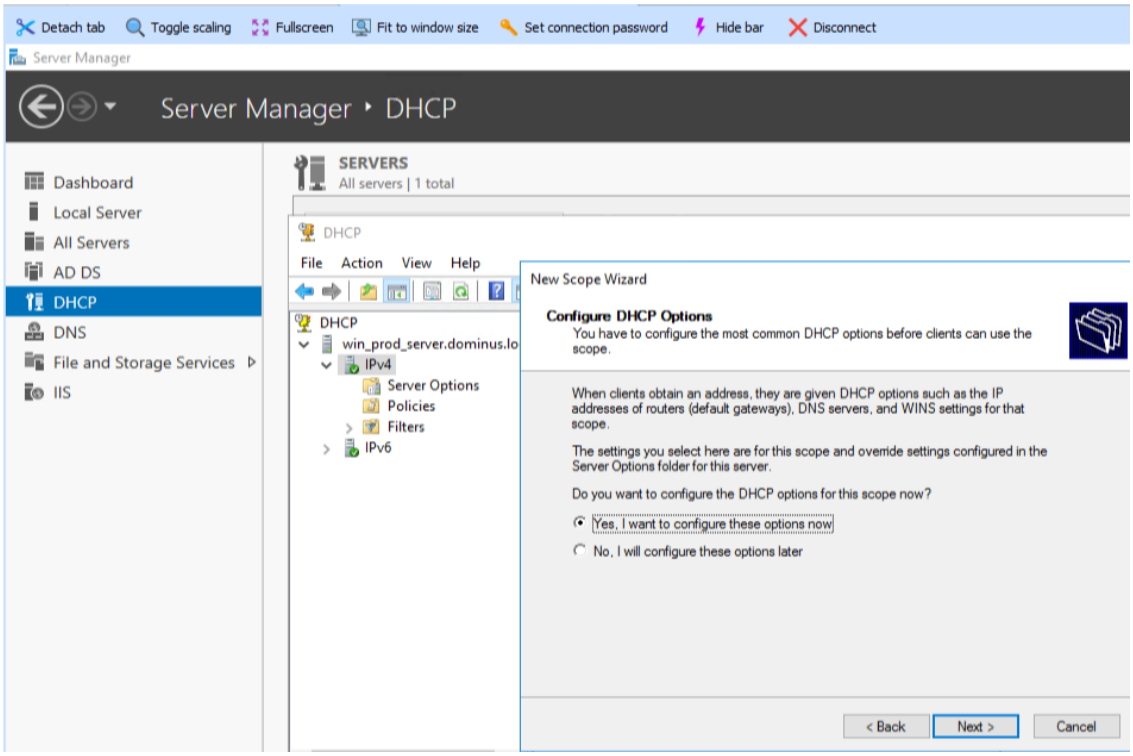


Рис. 3.20. Завершення етапу конфігурації

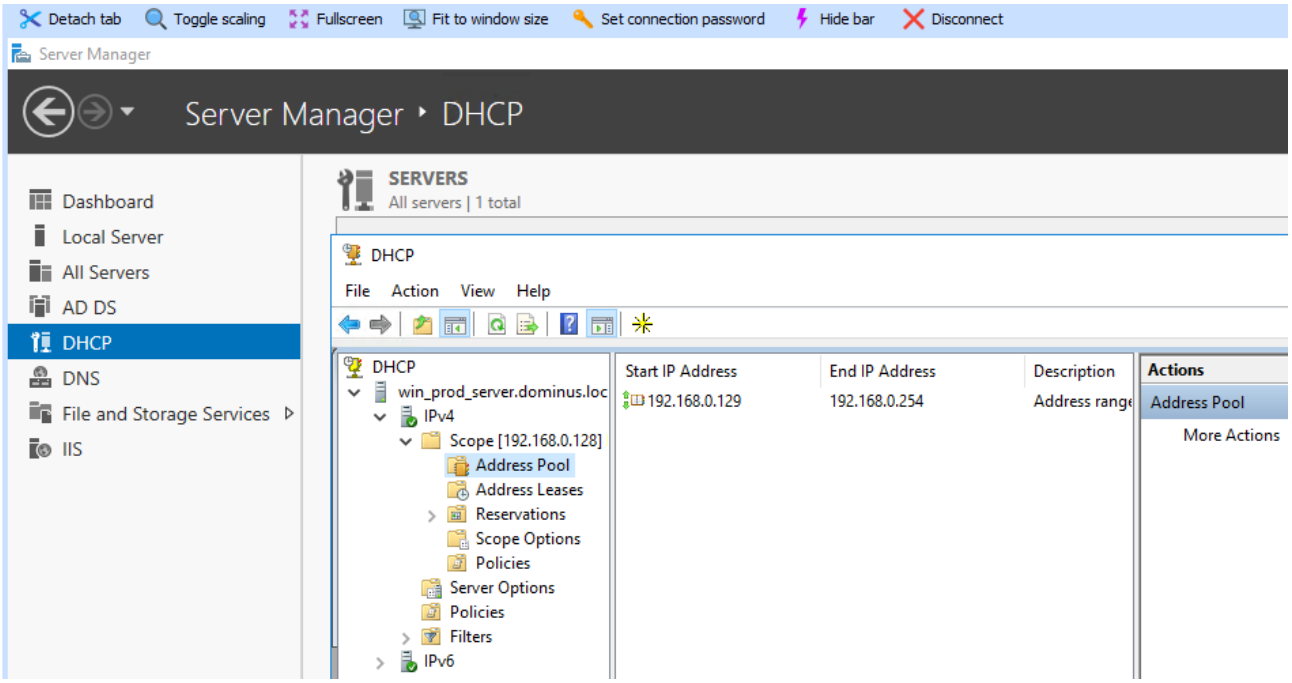


Рис. 3.21. Результат налаштувань

DNS-сервером в домені виступає IP-адреса контроллера домену

3.3. Конфігурація, інтеграція та тестування серверу на базі Linux

Основними сервісами, які були розгорнуті на Linux машині є веб-сервер Nginx, який працює в ізольованому середовищі Docker контейнера[16]. На даному сервері розміщено сайт компанії, також розгорнута база даних MongoDB, яка працює з даними з сайту.

```
[ec2-user@ip-192-168-0-7 ~]$ sudo yum install amazon-linux-extras install docker
Loaded plugins: priorities, update-motd, upgrade-helper
No package amazon-linux-extras available.
No package install available.
Resolving Dependencies
--> Running transaction check
---> Package docker.x86_64 0:19.03.6ce-4.58.amzn1 will be installed
--> Processing Dependency: runc >= 1.0.0 for package: docker-19.03.6ce-4.58.amzn1.x86_64
--> Processing Dependency: containerd >= 1.3.2 for package: docker-19.03.6ce-4.58.amzn1.x86_64
--> Processing Dependency: xfsprogs for package: docker-19.03.6ce-4.58.amzn1.x86_64
--> Processing Dependency: pigz for package: docker-19.03.6ce-4.58.amzn1.x86_64
--> Processing Dependency: libseccomp.so.2()(64bit) for package: docker-19.03.6ce-4.58.amzn1.x86_64
--> Running transaction check
---> Package containerd.x86_64 0:1.3.2-1.3.amzn1 will be installed
---> Package libseccomp.x86_64 0:2.3.1-2.4.amzn1 will be installed
---> Package pigz.x86_64 0:2.3.3-1.6.amzn1 will be installed
---> Package runc.x86_64 0:1.0.0-0.1.20200204.gitdc9208a.1.amzn1 will be installed
---> Package xfsprogs.x86_64 0:4.5.0-18.23.amzn1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                    Arch                    Version
================================================================================
Installing:
 docker                     x86_64                  19.03.6ce-4.58.amzn1
Installing for dependencies:
 containerd                 x86_64                  1.3.2-1.3.amzn1
 libseccomp                 x86_64                  2.3.1-2.4.amzn1
 pigz                       x86_64                  2.3.3-1.6.amzn1
 runc                       x86_64                  1.0.0-0.1.20200204.gitdc9208a.
 xfsprogs                   x86_64                  4.5.0-18.23.amzn1

Transaction Summary
================================================================================
Install  1 Package (+5 Dependent packages)

Total download size: 79 M
Installed size: 273 M
Is this ok [y/d/N]: y
Downloading packages:
(1/6): pigz-2.3.3-1.6.amzn1.x86_64.rpm
(2/6): libseccomp-2.3.1-2.4.amzn1.x86_64.rpm
(3/6): containerd-1.3.2-1.3.amzn1.x86_64.rpm
(4/6): runc-1.0.0-0.1.20200204.gitdc9208a.1.amzn1.x86_64.rpm
(5/6): xfsprogs-4.5.0-18.23.amzn1.x86_64.rpm
(6/6): docker-19.03.6ce-4.58.amzn1.x86_64.rpm
--------------------------------------------------------------------------------
Total
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : libseccomp-2.3.1-2.4.amzn1.x86_64
  Installing : runc-1.0.0-0.1.20200204.gitdc9208a.1.amzn1.x86_64
  Installing : containerd-1.3.2-1.3.amzn1.x86_64
  Installing : pigz-2.3.3-1.6.amzn1.x86_64
  Installing : xfsprogs-4.5.0-18.23.amzn1.x86_64
```

Рис.3.22. Встановлення Docker

```
[ec2-user@ip-192-168-0-7 ~]$ sudo service docker start
Starting cgconfig service:                                 [  OK  ]
Starting docker:        .                                  [  OK  ]
[ec2-user@ip-192-168-0-7 ~]$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:d58e752213a51785838f9eed2b7a498ffa1cb3aa7f946dda11af39286c3db9a9
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

[ec2-user@ip-192-168-0-7 ~]$
```

Рис. 3.23 Перевірка роботи

```
[ec2-user@ip-192-168-0-7 ~]$ sudo yum install nginx
Loaded plugins: priorities, update-motd, upgrade-helper
amzn-main
amzn-updates
Resolving Dependencies
--> Running transaction check
---> Package nginx.x86_64 1:1.16.1-1.37.amzn1 will be installed
--> Processing Dependency: libprofiler.so.0()(64bit) for package:
--> Running transaction check
---> Package gperftools-libs.x86_64 0:2.0-11.5.amzn1 will be inst
--> Processing Dependency: libunwind.so.8()(64bit) for package: g
--> Running transaction check
---> Package libunwind.x86_64 0:1.1-10.8.amzn1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================
 Package                                              Arch
================================================================
Installing:
 nginx                                                x86_64
Installing for dependencies:
 gperftools-libs                                      x86_64
 libunwind                                            x86_64

Transaction Summary
================================================================
Install  1 Package (+2 Dependent packages)

Total download size: 1.2 M
Installed size: 3.0 M
Is this ok [y/d/N]: y
Downloading packages:
(1/3): libunwind-1.1-10.8.amzn1.x86_64.rpm
(2/3): nginx-1.16.1-1.37.amzn1.x86_64.rpm
(3/3): gperftools-libs-2.0-11.5.amzn1.x86_64.rpm
----------------------------------------------------------------
Total
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : libunwind-1.1-10.8.amzn1.x86_64
  Installing : gperftools-libs-2.0-11.5.amzn1.x86_64
  Installing : 1:nginx-1.16.1-1.37.amzn1.x86_64
  Verifying  : libunwind-1.1-10.8.amzn1.x86_64
  Verifying  : gperftools-libs-2.0-11.5.amzn1.x86_64
  Verifying  : 1:nginx-1.16.1-1.37.amzn1.x86_64

Installed:
  nginx.x86_64 1:1.16.1-1.37.amzn1

Dependency Installed:
  gperftools-libs.x86_64 0:2.0-11.5.amzn1

Complete!
[ec2-user@ip-192-168-0-7 ~]$
```

Рис. 3.24. Встановлення Nginx

```
[ec2-user@ip-192-168-0-7 ~]$ sudo docker run --name docker-nginx -p 80:80 -d -v /home/ec2-user/fedcorse:/usr/share/nginx/html nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
8559a31e96f4: Pull complete
8d69e59170f7: Pull complete
3f9f1ec1d262: Pull complete
d1f5ff4f210d: Pull complete
1e22bfa8652e: Pull complete
Digest: sha256:21f32f6c08406306d822a0e6e8b7dc81f53f336570e852e25fbe1e3e3d0d0133
Status: Downloaded newer image for nginx:latest
c70f37c5de1c17d351a7e61696b483b6d2a1e8322abf1dfd9a68d9074440a26c
[ec2-user@ip-192-168-0-7 ~]$
```

Рис. 3.25. Створення Docker контейнеру з Nginx та розміщеним на ньому веб-сайтом

```
[ec2-user@ip-192-168-0-7 ~]$ sudo docker run -d -p 192.168.0.7:27017:27917 --name mongo-exp-project mongo
972f3b7f00c9376bd155dcfb95537317603 3ac8fc88734802b63186f375b6333
[ec2-user@ip-192-168-0-7 ~]$ sudo docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS                                  NAMES
972f3b7f00c9        mongo               "docker-entrypoint.s…"   26 seconds ago      Up 25 seconds               27017/tcp, 192.168.0.7:27017->27917/tcp   mongo-exp-project
dde7a1862037        mongo               "docker-entrypoint.s…"   12 minutes ago      Created                     27017/tcp                              mongodb_noauth
c70f37c5de1c        nginx               "/docker-entrypoint.…"   14 hours ago        Up 14 hours                 0.0.0.0:80->80/tcp                     docker-nginx
17dba6f56282        hello-world         "/hello"                 14 hours ago        Exited (0) 14 hours ago                                            reverent_feynman
[ec2-user@ip-192-168-0-7 ~]$ sudo docker exec -it 972f3b7f00c9 mongo
MongoDB shell version v4.2.7
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("feec9ab8-423e-49c5-9f41-49d25f434119") }
MongoDB server version: 4.2.7
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2020-06-13T18:15:11.425+0000 I  STORAGE  [initandlisten]
2020-06-13T18:15:11.425+0000 I  STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2020-06-13T18:15:11.425+0000 I  STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2020-06-13T18:15:12.171+0000 I  CONTROL  [initandlisten]
2020-06-13T18:15:12.171+0000 I  CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-06-13T18:15:12.172+0000 I  CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2020-06-13T18:15:12.172+0000 I  CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

> use Orders_DB
switched to db Orders_DB
> db.Orders_DB.insert({"name":"John Snow","id": 1, "Delivery Address": "Baker street 222b"})
WriteResult({ "nInserted" : 1 })
> show dbs
Orders_DB  0.000GB
admin      0.000GB
config     0.000GB
local      0.000GB
> db.Orders_DB.find()
{ "_id" : ObjectId("5ee519e933d96211ba420d5d"), "name" : "John Snow", "id" : 1, "Delivery Address" : "Baker street 222b" }
>
```

Рис. 3.26. Розгортання та перевірка роботи бази даних

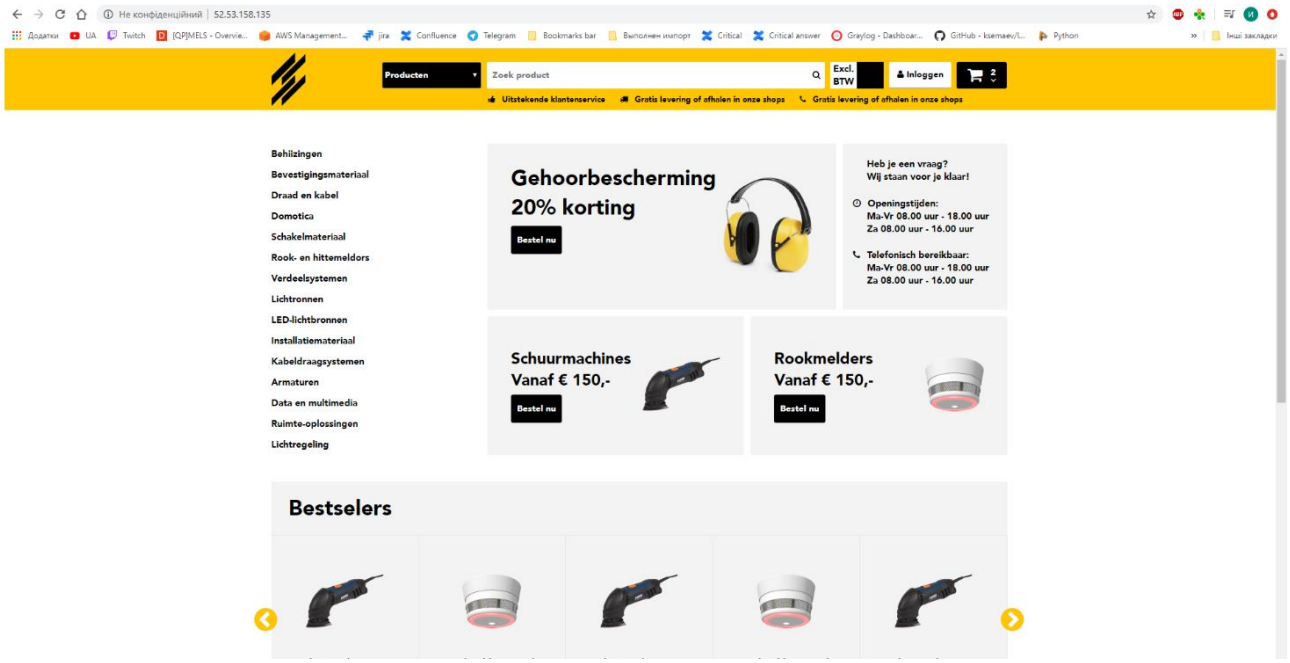


Рис. 3.27. Перевірка доступності веб-сайту з інтеренету

Після розгортання та перевірка працездатності основних сервісів, які базуються на Linux, було вирішено задачу інтеграції з Windows, а саме додано Linux машину в домен Active Directory[16]. Це дає змогу оптимізувати доступ та керування ресурсами, можна додавати права доступа користувачам Windows проходити аутентифікацію на Linux сервері з тими ж самими логіном та паролем, що й на Windows. Для реалізації даної концепції було використано служби SSSD(System Security Services Daemon), яка були налаштовані з допомогою пакету Realmd (Realm discovery), а для аутентифікації було використано протокол Kerberos, оскільки він є стандартом серед протоколів взаємної аутентифікації клієнта та сервера і використовує надійні алгоритми шифруванняр[17].

```
# Configuration snippets may be placed in this directory as well
includedir /etc/krb5.conf.d/

[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log

[libdefaults]
 dns_lookup_realm = false
 ticket_lifetime = 24h
 renew_lifetime = 7d
 forwardable = true
 rdns = false
 pkinit_anchors = FILE:/etc/pki/tls/certs/ca-bundle.crt
 default_realm = DOMINUS.LOCAL
 default_ccache_name = KEYRING:persistent:%{uid}

[realms]
 DOMINUS.LOCAL = {
   kdc = Win_prod_server.dominus.local
   admin_server = Win_prod_server.dominus.local
 }

[domain_realm]
 .dominus.local = DOMINUS.LOCAL
 dominus.local = DOMINUS.LOCAL
```

Рис. 3.28. Редагування конфігураційного фалу для Kerberos

```
[ec2-user@ip-172-31-4-137 etc]$ sudo realm discover Win_prod_server.dominus.local --verbose
 * Resolving: _ldap._tcp.win_prod_server.dominus.local
 * Resolving: win_prod_server.dominus.local
 * Performing LDAP DSE lookup on: 172.31.16.157
 * Successfully discovered: dominus.local
dominus.local
  type: kerberos
  realm-name: DOMINUS.LOCAL
  domain-name: dominus.local
  configured: no
  server-software: active-directory
  client-software: sssd
  required-package: oddjob
  required-package: oddjob-mkhomedir
  required-package: sssd
  required-package: adcli
  required-package: samba-common
[ec2-user@ip-172-31-4-137 etc]$
```

Рис. 3.29. Сканування та пошук домену Active Directory

```
[ec2-user@ip-192-168-0-7 ~]$ sudo realm join -U Administrator@DOMINUS.LOCAL DOMINUS.LOCAL --verbose
 * Resolving: _ldap._tcp.dominus.local
 * Performing LDAP DSE lookup on: 192.168.0.6
 * Successfully discovered: dominus.local
Password for Administrator@DOMINUS.LOCAL:
 * Required files: /usr/sbin/oddjobd, /usr/libexec/oddjob/mkhomedir, /usr/sbin/sssd, /usr/bin/net
 * LANG=C LOGNAME=root /usr/bin/net -s /var/cache/realmd/realmd-smb-conf.MEJ7L0 -U Administrator@DOMINUS.LOCAL ads join dominus.local
Enter Administrator@DOMINUS.LOCAL's password:DNS update failed: NT_STATUS_INVALID_PARAMETER

Using short domain name -- DOMINUS
Joined 'IP-192-168-0-7' to dns domain 'dominus.local'
No DNS domain configured for ip-192-168-0-7. Unable to perform DNS Update.
 * LANG=C LOGNAME=root /usr/bin/net -s /var/cache/realmd/realmd-smb-conf.MEJ7L0 -U Administrator@DOMINUS.LOCAL ads keytab create
Enter Administrator@DOMINUS.LOCAL's password:
 * /sbin/chkconfig sssd on
 * /sbin/service sssd restart
Stopping sssd: cat: /var/run/sssd.pid: No such file or directory
[FAILED]
Starting sssd: [  OK  ]
 * /bin/sh -c /usr/sbin/authconfig --update --enablesssd --enablesssdauth --enablemkhomedir --nostart && /sbin/chkconfig oddjobd on
Starting oddjobd: [  OK  ]
 * Successfully enrolled machine in realm
[ec2-user@ip-192-168-0-7 ~]$
```

Рис. 3.30. Приєднання до домену

Linux машина успішно пройшла аутентифікацію Kerberos та була додана до домену, це можна побачити на рисунку 3.31

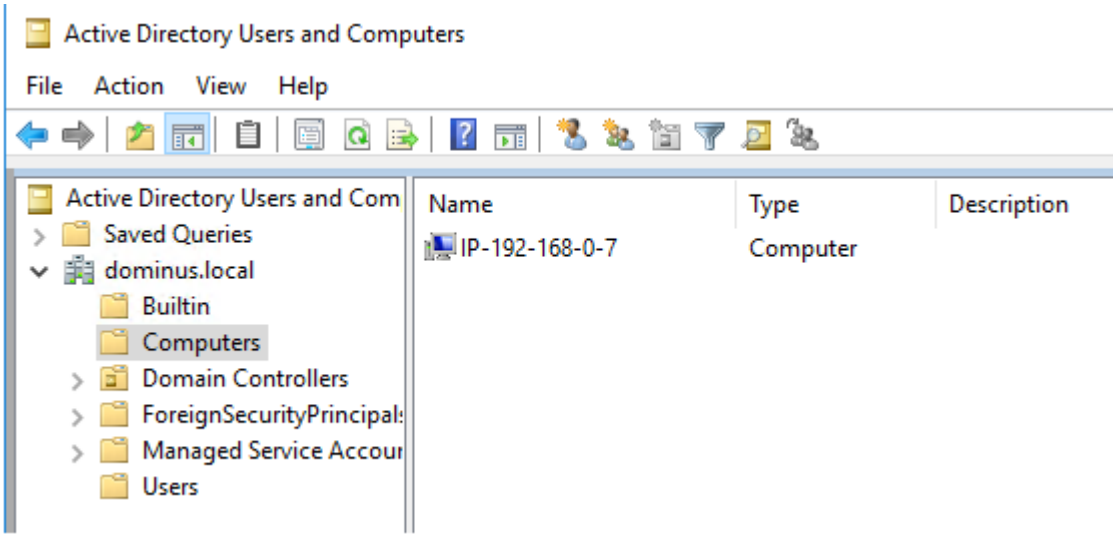


Рис. 3.31. Перевірка успішності приєднання

Після того як машина була приєднана до домену, було налагоджено політики доступу, оскільки політики Windows не працюють на пряму в Linux, потрібно задати їх у Linux,а потім якимось чином синхронізувати. Це було реалізовано наступним чином, на Windows була створена група безпеки SEC_SRV_ASTRA_LNX_PO1, учасники якої повинні мати змогу логінитися на Linux сервері під своїми даними, які вони використовують на Windows, а також мати права суперкористувача (sudo). Ця група була внесена в конфігураційний файл утиліти realm на Linu, яка в свою чергу синхронізую користувачів з Active Directory та їх права в системі Linux.

```
[ec2-user@ip-192-168-0-7 ~]$ sudo realm permit --groups SEC_SRV_ASTRA_LNX_P01_SA@dominus.local
[ec2-user@ip-192-168-0-7 ~]$ sudo realm list
dominus.local
  type: kerberos
  realm-name: DOMINUS.LOCAL
  domain-name: dominus.local
  configured: kerberos-member
  server-software: active-directory
  client-software: sssd
  required-package: oddjob
  required-package: oddjob-mkhomedir
  required-package: sssd
  required-package: adcli
  required-package: samba-common
  login-formats: %U@dominus.local
  login-policy: allow-permitted-logins
  permitted-logins:
  permitted-groups: SEC_SRV_ASTRA_LNX_P01_SA@dominus.local
[ec2-user@ip-192-168-0-7 ~]$
```

Рис. 3.32. Додання групи, що дає доступ до Linux машини

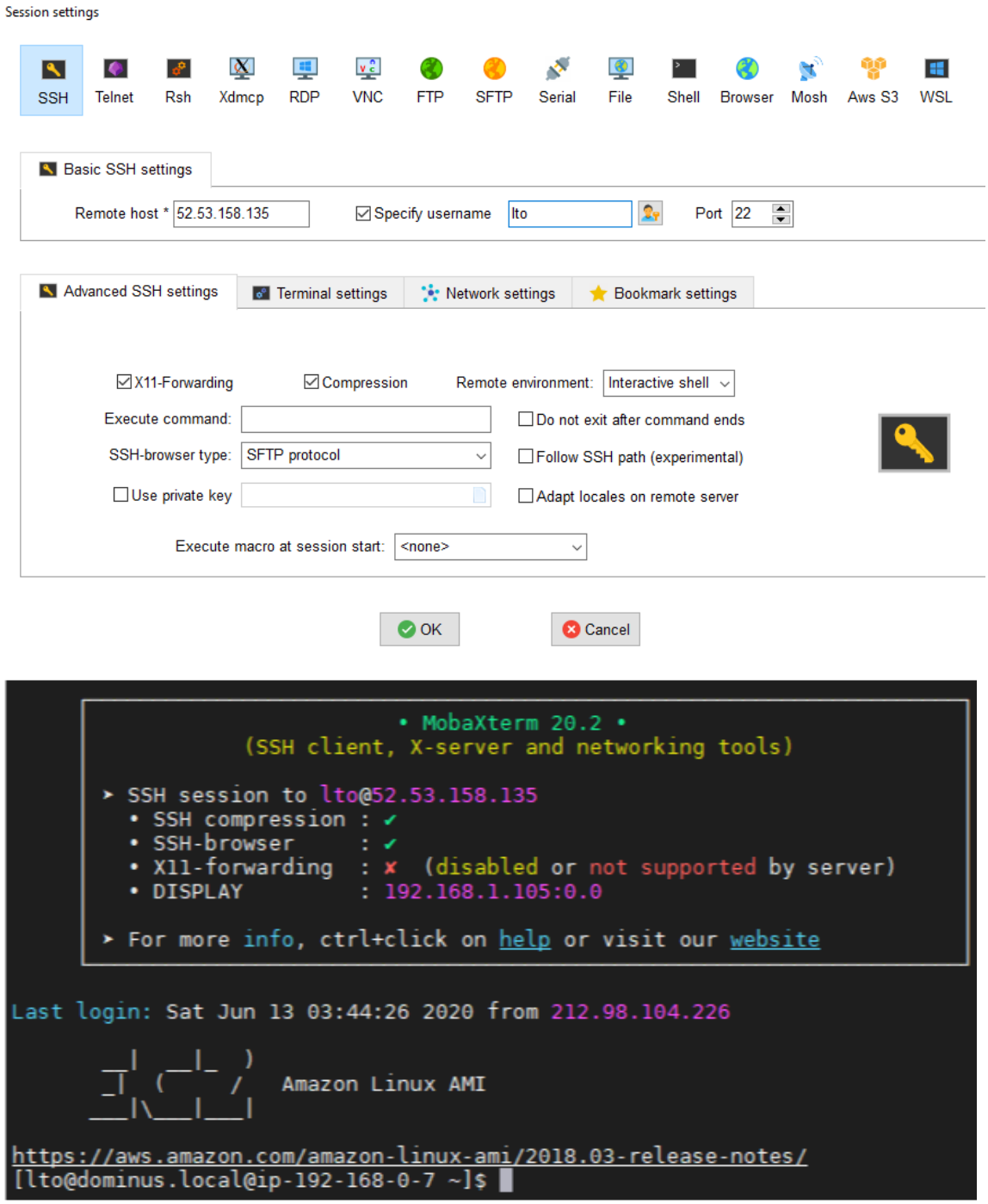
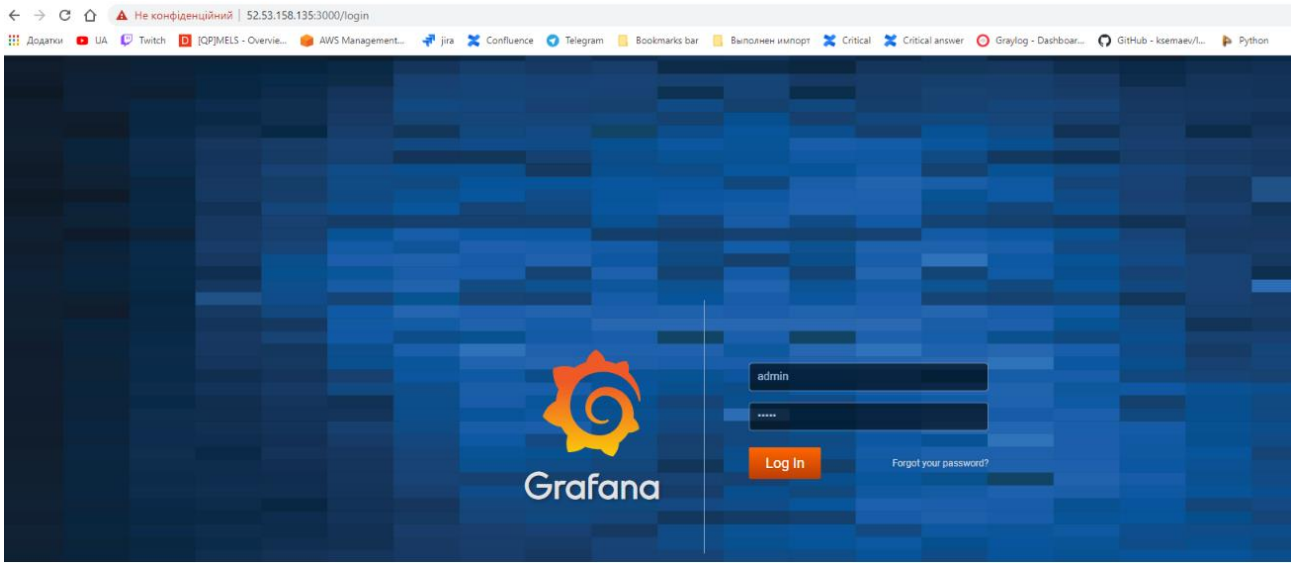


Рис. 3.33 – 3.34. Підключення по Linux сервера з даними користувача Windows

Підключення виконано успішно, синхронізація працює коректно. Користувач, який доданий у відповідну групу безпеки має доступ до Linux сервера.

Після налаштування та перевірки основних сервісів, а також вдалох інтеграції з Windows було налаштовано нелевичку систему моніторингу за допомогою стеку Prometheus + Grafana

```
[ec2-user@ip-192-168-0-7 swarmprom]$ sudo docker stack deploy -c docker-compose.yml mon
Creating network mon_net
Creating config mon_caddy_config
Creating config mon_dockerd_config
Creating config mon_node_rules
Creating config mon_task_rules
Creating service mon_dockerd-exporter
Creating service mon_cadvisor
Creating service mon_grafana
Creating service mon_alertmanager
Creating service mon_node-exporter
Creating service mon_prometheus
Creating service mon_caddy
[ec2-user@ip-192-168-0-7 swarmprom]$ sudo docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE
yuzovj0hzzqy        mon_alertmanager    replicated          1/1                 stefanprodan/swarmprom-alertmanager:v0.14.0
traenmgecszd        mon_caddy           replicated          1/1                 stefanprodan/caddy:latest
s9tmw4tyjvqk        mon_cadvisor        global              0/1                 google/cadvisor:latest
yw2cwxdu0tx8        mon_dockerd-exporter global              1/1                 stefanprodan/caddy:latest
cj2yf4d6hcpk        mon_grafana         replicated          1/1                 stefanprodan/swarmprom-grafana:5.3.4
k4bpxw4t5yhw        mon_node-exporter   global              0/1                 stefanprodan/swarmprom-node-exporter:v0.16.0
olwmv3t1w0ja        mon_prometheus      replicated          1/1                 stefanprodan/swarmprom-prometheus:v2.5.0
```



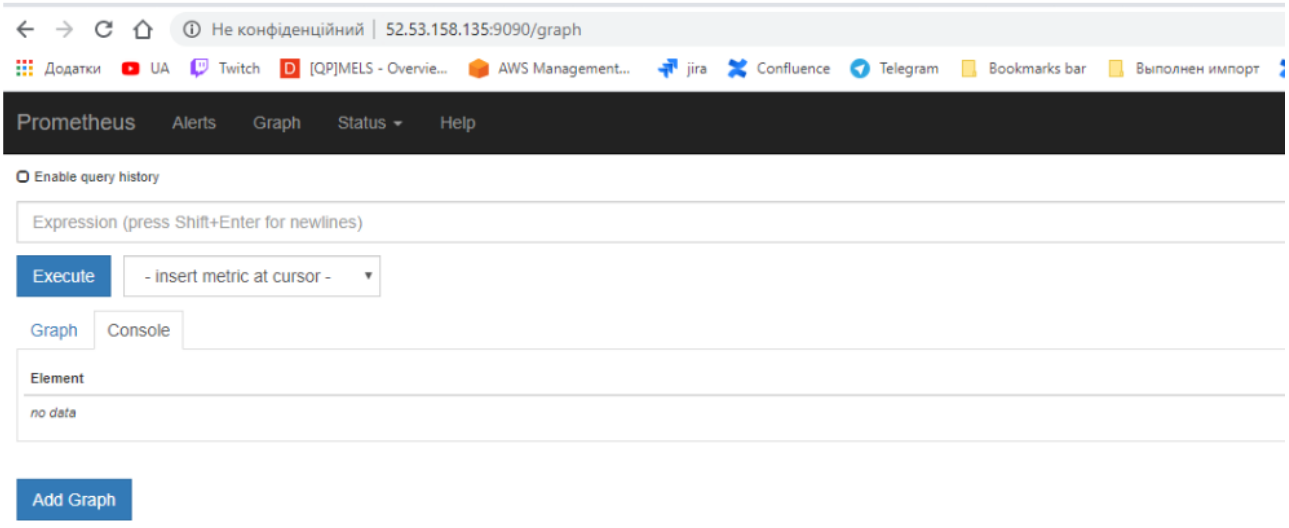


Рис.3.35 – 3.37 Налаштування моніторингу



Рис. 3.38. Приклад дашборду для моніторингу

3.4. Аналіз результатів

Як і планувалося, хмарна інфраструктура була успішно розгорнута на Amazon Web Services. Сервери є доступними та готовими до роботи. Windows Server 2016 успішно справляється з усіма покладеними на нього задчами. Сконфігурований DHCP сервер коректно розповсюджує мережеву інформацію вузлам, DNS сервер активно та правильно декодує символьні адреси в логічні. Домен Active Directory також працює коректно, користувачі успішно авторизуються під своїми обліковими записами, групові політки вдало застосоувються та дають потрібні права користувачам, яких в них додають.

Машина на базі Linux також вдало досягає поставлених перед нею цілей. Веб-сайт компанії, розміщений на веб-сервері Nginx доступний для пористувачів мережі інтернет. База даних MongoDB справляється з запитами клієнтів сайту. Інтеграція з Windows також пройшла максимально успішно, було перевірено можливість віддаленого доступу до Linux серверу, користувачів відповідної групи в домені. Користувачі успішно авторизувалися під своїми обліковими записами створеними у Windows. Очевидно, що моніторинг також є важливою частиною будь-якої інфраструктури, адже вчасне виявлення проблеми дозволяє запобігати критичних інцидентів, які можуть підставити під удар всю систему.

Одним з найважливіших факторів оцінки будь-якої системи є собівартість

її розгортання та підтримки. Собівартість інфраструктуру розробленої в даній роботі – одна з найважливіших її переваг, оскільки не секрет, що будь-який бізнес прагне мінімізувати свої розходи, та максимізувати доходи. Фактично, для розгортання подібної інфраструктури потрібно платити лише за оренду віртуальних машин Amazon і зарплату спеціалістам, які будуть займатися розгортанням, конфігуруванням, підтримкою, моніторингом, розширення та вдосконалення інфраструктури. Усі інші програмні компоненти, що були використані при побудові інфраструктури – опенсортс, тобто їх програмний код відкритий та безкоштовний, кожен бажаючий може використовувати його в своїх цілях абсолютно безкоштовно. Тим не менш це зовсім не означає, що такі рішеня гірше тих за які потрібно платити, просто це кардинально різні підходи розробників до розповсюдження власного програмного забезпечення.

Звичайно, розроблена інфраструктура не є ідеальною і її можна розширювати та модернізувати. Особисто я бачу три основні напрямки можливого розвитку, перший, більш глибока система моніторингу, тобто налаштування її таким чином, щоб можна було моніторити та відслідковувати буквально усе від навантаження на процесору сервера до часу за який у користувачів перший раз завантажується сторінка веб-сайту компанії. Другий напрямок, це горизонтальне та вертикальне розширення інфраструктури, тобто оренда більш потужних віртуальних машин, з більшою кількістю оперативної пам'яті та вищою частотою процесора, а також збільшення кількості цих машин. І третій напрямок, створення кластеру серверів, для забезпечення більою відмовостійкості, щоб при виході з ладу одного серверу, його одразу заміняв аналогічний інший.

ВИСНОВКИ

В даній кваліфікаційній роботі було проаналізовано основні типи та підходи організації хмарних інфраструктур сучасних IT-компаній, визначено основопложні принципи побудови таких інфраструктур, розглянуто переваги та недоліки кожного підходу. Сформавано власне бачення на реалізацію такої інфраструктури та власне її побудова

Розлянуто та проаналізовано переваги та недоліки, 3 основних провайдерів хмарних послуг. Дослідження кожного з них показали, що оптимальним буде використовувати Amazon Web Services, оскільки має найбільш розвинену эко-систему та інтеграцію між власними сервісами, також слід зазначити, що даний провайдер займає перше місце в світі по популярності і більша частина провідних компаній світу, користується усією різноманітністю його послуг та сервісів.

Були розглянуті інструменти для максимально зручного і швидкого розгортання та конфігурації. Спроектовано мережеве середовище в якому була побудована інфраструктура. Сформовані мінімальні системні вимоги до даної інфраструктури.

Проаналізовані програмні компоненти які виступають основою інфраструктури, такі як веб-сервери Nginx для Linux та IIS для Windows, база даних MongoDB, система контейнеризації Docker, служба каталогів Active Directory та деякі інші. Розроблена система прав та досьупа в домені Acvtive Directory, розділення користувачів по каталогам відповідно їх сфер діяльності, створено групи безпеки для доступу до внутрішніх ресурсів, серверів, баз даних тощо.

Були описані налаштування наступні налаштування на серверах: встановлення, конфігурація та розгортання сайту на базі веб-серверу Nginx, встановлення та конфігурація DHCP та DNS серверів, розгортання бази даних MongoDB, створення та налаштування домену Active Directory, інтеграція Linux серверу в домен, налаштування базової системи моніторингу та візуалізації Prometheus + Grafana. Після проведення усіх налаштувань, було проведено

тестування коректності роботи серверів та сервісів на них, а також сформовано висновки щодо модернізації та можливостей покращення інфраструктури.

Реалізована інфраструктура є повністю робочим прикладом реалізації реальної інфраструктури IT-компанії, звичайно у дещо зменшеному маштабі. Конфігурації можуть бути застосовані для налаштувань інфраструктури реальних IT-компаній, без яких-небудь обмежень. Макет може виористовуватися як приклад для дослідження побудови інфрастрктур в хмарному середовищі Amazon Web Services

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. DevOps on AWS : Learn to set up your infrastructure on cloud [Електронний ресурс] – Режим доступу до ресурсу: https://www.eduonix.com/devops-on-aws-learn-to-set-up-your-infrastructure-on-cloud

2. AWS in Motion [Електронний ресурс] – Режим доступу до ресурсу: https://www.manning.com/livevideo/aws-in-motion?a_aid=hackrio

3. Моримото Рэнд. Microsoft Windows Server 2016. Полное руководсто / Рэнд Моритимо. – М: Русское издательство, 2018. – 1456 с.

4. Томас Орин. Администрирование Windows Server 2012. Учебный курс Microsoft / Орин Томас, Йен Маклин. – Киев: Диалектика, 2014. – 464 с.

5. Минаси Марк. Windows Server 2012 R2. Полное руководство. Том 2: Дистанционное администрирование, установка среды с несколькими доменами, виртуализация, мониторинг и обслуживание сервера / Марк Минаси. – Харьков: Вильямс, 2009. – 464 с.

6. Kerberos – A ‘Hello World’ example on CentOS/RHEL 7 [Електронний ресурс] – Режим доступу до ресурсу: https://codingbee.net/rhce/rhce-kerberos

7. Krause J. Windows Server 2016 Cookbook, 2nd Edition / J. Krause – Seattle: Packt, 2016. – 494 с.

8. Бэнкер Кайл. MongoDB в действии / Кайл Бэнкер. – СПБ: ДМК пресс, 2017. – 450 с.

9. Немет Эви. Unix и Linux. Руководство системного администратора / Эви Немет, Гарт Снайдер.. – М: Диалектика 2020. – 442 с.

10. Руководство по MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/nosql/mongodb/

11. Керниган Брайан У. Unix. Программное окружение / Брайан У. Керниган, Роб Пайк.. – СПБ: Символ-Плюс. 2003 – 414 с.

12. Лин С. Администрирование Microsoft Windows Server 2012 / С. Лин. – Минск: Ковчег. 2014 – 304 с.

13. Лимончелли Томас А. Системное и сетевое администрирование. Практическое руководство / Томас А. Лимончелли, Кристина Хоган, Страта Чейлап. Харьков: Вильямс. 2009 – 944 с

14. Граннеманю Скотт. Linux. Необходимый код и команды. Карманный справочник / Скотт Граннеманю. – М: Диалектика. 2010 – 416 с

15. Nginx в Docker-контейнере в Ubuntu 14.04 [Електронний ресурс] – Режим доступу до ресурсу: https://www.codeflow.site/ru/article/how-to-run-nginx-in-a-docker-container-on-ubuntu-14-04

16. Ubuntu в домене Windows AD [Електронний ресурс] – Режим доступу до ресурсу: https://jtprog.ru/ubuntu-in-domen-windows/

17. Подключение Debian GNU/Linux 8.6 к домену Active Directory с помощью SSSD и realmd [Електронний ресурс] – Режим доступу до ресурсу: https://blog.it-kb.ru/2016/10/15/join-debian-gnu-linux-8-6-to-active-directory-domain-with-sssd-and-realmd-for-authentication-and-configure-ad-domain-security-group-authorization-for-sudo-and-ssh-with-putty-sso/

18. Ввод компьютера в домен Windows [Електронний ресурс] – Режим доступу до ресурсу: https://help.ubuntu.ru/wiki/

19. Протокол Kerberos [Електронний ресурс] – Режим доступу до ресурсу: http://ypn.ru/427/kerberos-protocol/

20. Kerberos [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Kerberos

21. Grafana как еще один инструмент для технического мониторинга [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/southbridge/blog/431122/

22. Мониторинг с помощью Prometheus [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/southbridge/prometheus-monitoring-ba8fbda6e83

23. Установка системы мониторинга Prometheus [Електронний ресурс] – Режим доступу до ресурсу: https://voxlink.ru/kb/linux/ustanovka-sistemy-monitoringa-prometheus/

ДОДАТКИ

Додаток А

Лістинг конфігураційних файлів

Servers.tf

```
provider  “aws” {
    access_key = “**********”
    secret_key = “********************”
    region        = “ us-west-1”
    }
    resource "aws_vpc" "aws_vpc" {
        cidr_block  = "${var.vpc_cidr}"
        instance_tenancy = "default"
        enable_dns_support  = "true"
        enable_dns_hostnames = "false"
        assign_generated_ipv6_cidr_block    = "false"
        enable_classiclink  = "false"
        tags {
            Name  = "MyVPC"
        }
    }
resource “aws-instance” “Win_prod_server” {
    ami = “ami-04e59c05167ea7bd5”
    instance_type = “t2.micro”
}
resource “aws-instance” “Win_prod_server” {
    ami = “ami-04e59c05167ea7bd5”
    instance_type = “t2.micro”
}
```

Kerberos.conf

```
# Configuration snippets may be placed in this directory as well
includedir /etc/krb5.conf.d/

[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log

[libdefaults]
 dns_lookup_realm = false
 ticket_lifetime = 24h
 renew_lifetime = 7d
 forwardable = true
 rdns = false
 pkinit_anchors = FILE:/etc/pki/tls/certs/ca-bundle.crt
 default_realm = DOMINUS.LOCAL
 default_ccache_name = KEYRING:persistent:%{uid}

[realms]
 DOMINUS.LOCAL = {
  kdc = Win_prod_server.dominus.local
  admin_server = Win_prod_server.dominus.local
 }
[domain_realm]
 .dominus.local = DOMINUS.LOCAL
 dominus.local = DOMINUS.LOCAL
```