Лабораторна робота №3

ЗНАЙОМСТВО З ІНСТРУМЕНТАМИ УПРАВЛІННЯ В AZURE: AZURE PORTAL, AZURE CLOUD SHELL, BASH, POWERSHELL, ARM TEMPLATES

Мета заняття: ознайомитися з основними інструментами управління в Azure, а саме навчитися працювати з Azure Portal для створення та налаштування ресурсів; опанувати Azure Cloud Shell як інтегроване середовище для керування хмарними сервісами; відпрацювати використання Bash та PowerShell для автоматизації адміністративних завдань; навчитися створювати та застосовувати ARM Templates для опису інфраструктури методом Infrastructure as a Code.

Теоретичні відомості

Загальні відомості про інструменти взаємодії користувачів Azure з хмарою

Перед початом огляду інструментів, що дозволяють створювати, оновлювати, керувати, видаляти та в цілому керувати життєвим циклом ресурсів в Azure, варто розгляти як працює Azure Resource Manager.

Azure Resource Manager – це сервіс для розгортання та управління в Azure. Він забезпечує рівень управління, який допомагає створювати, оновлювати та видаляти ресурси у середовищі Azure. Після того, як ресурс було створено, Azure Resource Manager дозволяє використовувати функції управління, такі як контроль доступу (RBAC), блокування (locks) та теги для захисту й організації ресурсів.

Коли користувач Azure відправляє будь-який запит в Azure, використовуючи Azure API, CLI-інструменти або SDK, то саме Azure Resource Manager опрацьовує його. Користувач проходить процес автентифікації, Azure Resource Manager перевіряє, чи користувач авторизований до виконання тієї чи іншої дії і відправляє далі запит на

відповідний сервіс Azure. Оскільки в кінцевому результаті запити обробляються через один API, то не важливо який інструмент було використано, бо в результаті користувач отримає консистентний та послідовний результат. На рис. 7 зображено як Azure Resource Manager пов'язаний у процесі взаємодії користувача з хмарою з допомогою різних інструментів.

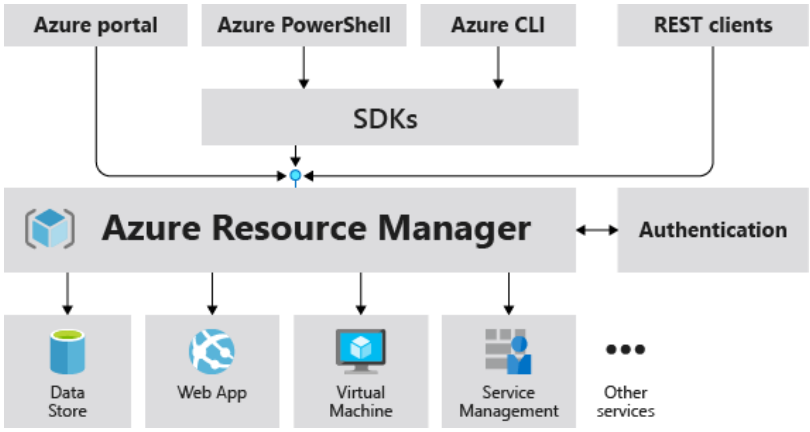


Рис. 7. Azure Resource Manager у процесі керування життєвим циклом ресурсів

Оскільки Azure Resource Manager є дуже важливим компонентом Azure, то його архітектура була побудована із забезпеченням відмовостійкості та безперервної доступності. Запити, які відправляються на Resource manager та control plane:

- розподіляються між регіонами. Resource Manager має окремий інстанс у кожному регіоні Azure, що означає, що у випадку, якщо щось стається з інстансом в одному регіоні, це не впливає на доступність сервісів в тому регіонів. Це саме стосується й інших служб Azure. Хоч сам Resource Manager і розроділений між регіонами, є і ресурси, які мають залежності з певними регіонами. В свою чергу це означає, що запити до Resource Manager можуть бути успішними, але на рівні служби вони можуть не виконатись у випадку збоїв;
- розподіляються між Availability Zones у регіонах, де вони доступні. Цей розподіл гарантує, що у випадку недоступності однієї з Availability

Zones Resource Manager зможе використати іншу Availability Zone або регіон;

- не залежить від одного логічного дата-центру;
- залишається доступним під час обслуговування інфрасктруктури.

Ця відмовостійкість застосовуюється і до сервісів, що отримують запити через Resource Manager. Azure Key Vault – один з сервісів, що використовує перевагу даної консистентності.

Завдяки використанню глобального endpoint management.azure.com, користувачі мають змогу користуватись розподілом трафіку на основі DNS та автоматичною маршрутизацією трафіку до найближчого або справного регіону, що водночас і підвищує відмовостійкість та продуктивність.

Одночасні спроби взаємодії з одним і тим же ресурсом може призвести до несподіваних результатів. У такому випадку Resource Manager виявляє конфлікт та дозволяє успішно виконати лише першу операцію та блокує решту й повертає помилку. Дане рішення гарантує, що оновлення ресурсу є остаточним та надійним, а користувачі можуть бути впевнені у відсутності інконситентності чи втраті даних.

Наприклад, якщо два запити намагаються одночасно оновити один і той же ресурс, то перший запит виконається успішно, а другий отримає у відповідь 409 помилку HTTP. Після отримання цієї помилки є змога спробувати виконати другий запит на оновлення, якщо це доцільно.

Resource Provider – це служба, що дозволяє взаємодіяти з ресурсами певного виду. Наприклад, поширеним Resource Provider є Microsoft.Compute, який дозволяє взаємодіяти з віртуальними машинами, а також іншими обчислювальними ресурсами. Microsoft.Storage є прикладом ще одного популярного Resource Provider, який відповідає за роботу зі службами зберігання даних. Більшість популярних Resource Providers зареєстровані за замовчуванням на рівні Subscription, проте у деяких випадках для роботи з певними ресурсами їх потрібно зареєструвати

самостійно. Щоб побачити які Resource Providers зареєстровані, необхідно на рівні Subscription перейти у Settings > Resource providers (див. рис.8). Також отримати інформацію про зареєстровані Resource Providers або зареєструвати необхідні, можна за допомогою інших інструментів (Azure CLI, Azure PowerShell, API та SDK).

Resource provider визначає ресурси, які користувач може створювати. Ім'я типу ресурсу має наступний формат: {resource-provider}/{resource-type}. Наприклад, тип ресурсу для Azure Key Vault – Microsoft.KeyVault/vaults.

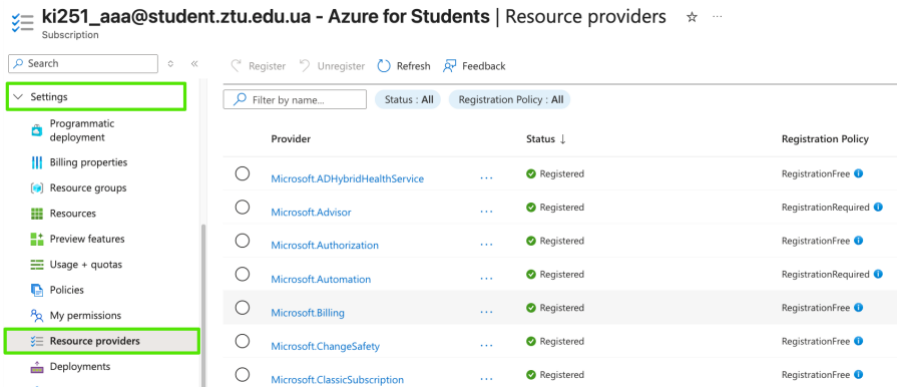


Рис. 8. Перелік Resource providers на рівні Subscriptions

Extension resources – це тип ресурсів, що розширює функціонал інших. До таких ресурсів можна віднести призначення RBAC ролі (Microsoft.Authorization/roleAssignments) або створення resource lock (Microsoft.Authorization/locks).

Azure Portal – це веб-консоль, яка дозволяє створювати та керувати всіма ресурсами Azure. За допомогою Azure Portal можна керувати Azure Subscription за допомогою графічного інтерфейсу. В Azure Portal можна створювати, керувати та контролювати все: від простих вебдодатків до складних хмарних інформаційних систем. Наприклад, можна створити нову базу даних, збільшити кількість обчислювальних ресурсів та контролювати місячні витрати. Також Azure Portal дозволяє переглянути

всі створені ресурси та скористатись майстром для створення нових ресурсів.

Архітектура Azure Portal створена для забезпечення відмовостійкості та доступності. Azure Portal розміщений в кожному дата-центрі і така конфігурація робить його стійкими до збоїв окремих дата-центрів і дозволяє уникнути проблем зі швидкодією, оскільки користувачі отримують доступ до Azure Portal, який розміщений географічно найближче до них. Даний ресурс постійно оновлюється і це відбувається без простоїв та непомітно для користувачів. Azure Portal доступний у будь-якому підтримуваному браузері.

За замовчуванням після першого входу користувачі бачать домашню сторінку Azure Portal. На ній зібрані ресурси, які допоможуть максимально ефективно використовувати Azure Subscription. Кнопка «Create» дозволяє швидко знайти необхідний ресурс та створити його у поточній Subscription. Також на домашній сторінці відображаються останні ресурси, що переглядались та посилання на безкоштовні онлайн-курси, документацію та інші корисні ресурси.

Меню Azure Portal та хедер є глобальними елементами та завжди наявні у інтерфейсі Azure Portal. Ці елементи є свого роду оболонкою інтерфейсу користувача, яка пов’язана з кожною окремою службою або функцією. В той же час у хедері знаходяться елементи глобального управління. Робоча панель певного ресурсу чи сервісу також мають службове меню з командами, специфічними до цієї області. На рис. 9 зображений інтерфейс Azure Portal, на якому наявна інформація про віртуальну машину, а в табл. 5 наявне пояснення компонентів елементів меню.

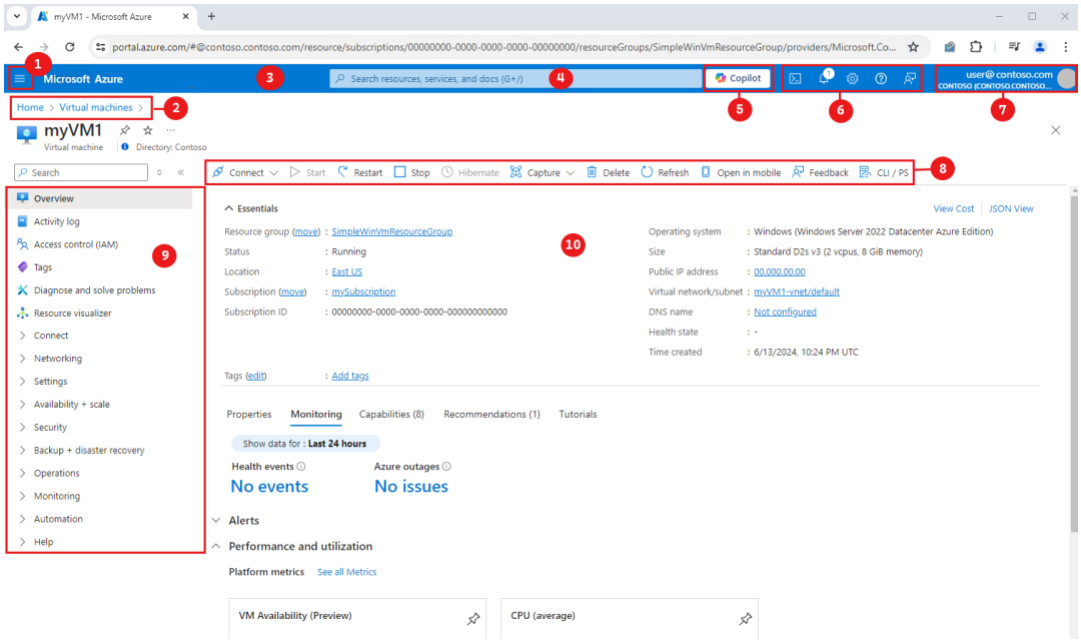


Рис. 9. Інтерфейс Azure Portal

Таблиця 6

Опис елементів інтерфейсу Azure Portal

Номер елементу	Опис
1	**Меню.** Цей глобальний елемент доступний завжди та допомогає користувачеві перемикатись між сервісами Azure. За замовчування воно приховане, поки не клікнути на відповідну піктограму.
2	**Навігаційна стежка.** Використовуйте її для навігації між елементами меню.
3	**Хедер**. Розташований вгорі сторінки та містить в собі елементи глобальні елементи.
4	**Глобальний пошук**. Використовується для швидкого пошуку окремих ресурсів, сервісів або документації. Use the search bar in the page header to quickly find a specific resource, a service, or documentation.
5	**Copilot**. Кнопка для швидкого доступу Microsoft Copilot in Azure.
6	**Глобальні елементи управління**. Це елементи управління такі як Cloud Shell, Notifications, Settings, Support + Troubleshooting і також Feedback.
7	**Ваш обліковий запис**. Дозволяє переглянути інформацію про обліковий запис, перемикатись між директоріями, між обліковими записами, входити та виходити з них.
8	**Командна панель**. Група елементів управління, яка актуальна для поточного ресурсу, що переглядається.
9	**Службове меню.** Бічне меню з командами, які актуальні для поточного ресурсу, що переглядається.
10	**Робоча панель**. Відображає детальну інформацію про відкритий ресурс або службу.

Azure Cloud Shell – це інтерактивний, автентифікований доступний у браузері, термінал для керування ресурсами в Azure. Користувачі Azure Cloud Shell можуть обрати командну оболонку на вибір: Bash або PowerShell.

Cloud Shell запускається в окремому контейнері під управлінням Azure Linux та запускається в якості однієї сесії для одно користувача з тайм-аутом у 20 хвилин, якщо користувач не здійснював ніяких дій.

Для того, щоб створити Azure Cloud Shell, який не доступний за замовчуванням, необхідно натиснути на необхідну піктограму серед глобальних елементів управління (див. рис. 10).

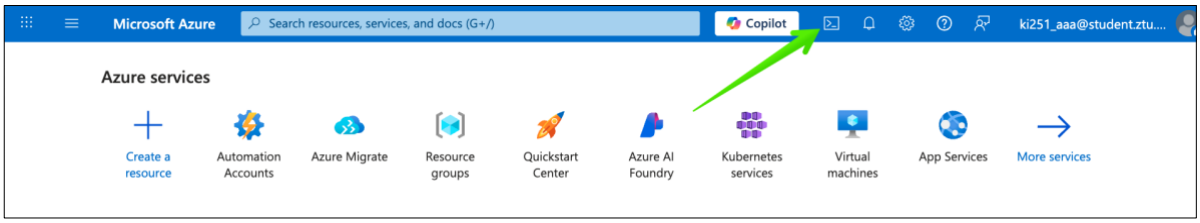


Рис. 10. Піктограма для запуску Azure Cloud Shell

Далі користувач може обрати оболонку на вибір: Bash або PowerShell (див. рис. 11).

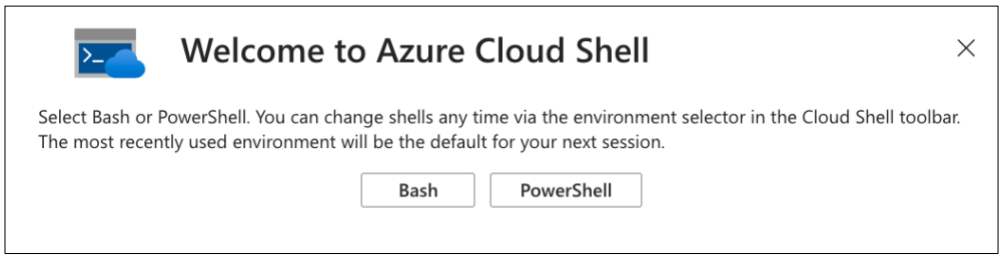


Рис. 11. Вікно вибору оболонки для Azure Cloud Shell

Після вибору будь-якої з оболонок надається можливість вибору використовувати Storage Account, чи ні, а також можливість створити Cloud Shell у віртуальній мережі та мати доступ до ресурсів, розміщених в ній (див. рис. 12).

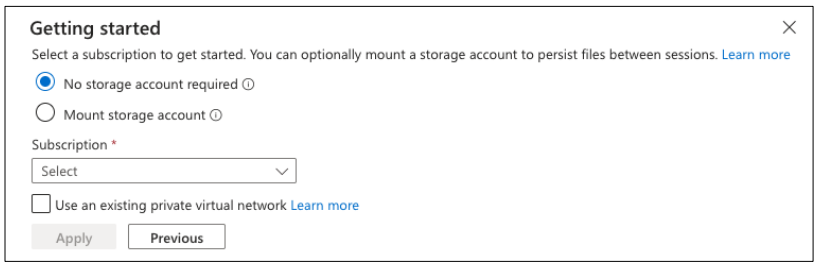


Рис. 12. Вікно налаштування Storage Account та Private Virtual Network

У випадку, якщо користувач обере першу опцію, без використання Storage Account, то по завершенню сесії всі файли, які були створені в середовищі Cloud Shell, перестануть бути доступними. Коли ж користувач обирає опцію використовувати Storage Account, то надається можливість обрати існуючий Storage Account, дозволити асистенту налаштування Storage Account створити його автоматично, використовуючи значення за замовчуванням або ж створити Storage Account самостійно. Важливо зазначити, що у Subscription Azure for Students діють обмеження на регіони, де можуть бути створені ресурси, а можна їх створюва у регіонах *swedencentral*, *norwayeast*, *germanywestcentral*, *switzerlandnorth*, *francecentral*, тому асистент може намагатись створити Storage Account в регіоні який не зазначений в політиці, як дозволений для розгортання ресурсів. Тому необхідно обрати або існуючий Storage Account, що був створений в одному з цих регіонів, та задовільняє вимоги використання Cloud Shell, або створити новий Storage Account.

Після створення Cloud Shell з використанням Storage Account, виконуються наступні дії:

- в заданому Storage Account створюється File Share (див. рис. 13);
- у File Share в директорії .cloudconsole знаходиться .img файл розміром у 5 GiB (див. рис. 14);
- в контейнері Cloud Shell використовується мережевий диск та підключаєтсья по SMB (CIFS) до директорії /usr/csuser/clouddrive (див. рис.15);

- в $HOME створюється симлінк clouddrive -> /usr/csuser/clouddrive (див. рис.16);
- директорія $HOME є точкою монтування пристрою /dev/loop0 (див. рис. 17), який прив’язаний до образу з розширенням img, що розташований у /usr/csuser/clouddrive/.cloudconsole (див. рис. 18).

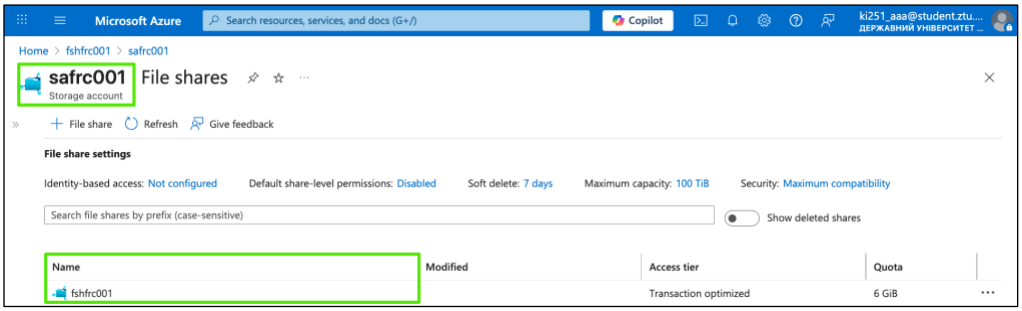


Рис. 13. Створений Azure File Share в Storage Account для Cloud Shell

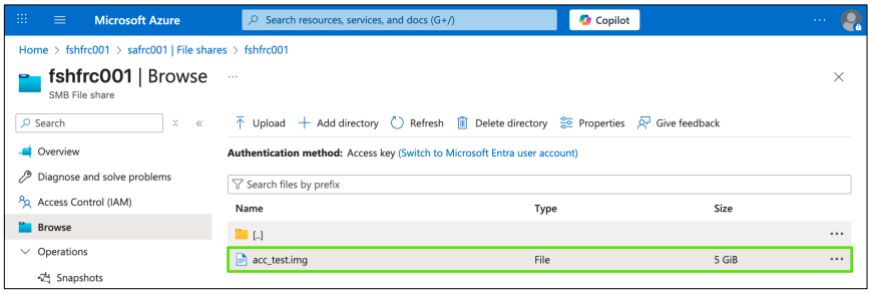


Рис. 14. img-файл розміром 5 GiB у File Share

```
test [ ~ ]$ mount | grep clouddrive
//safrc001.file.core.windows.net/fshfrc001 on /usr/csuser/clouddrive type cifs (rw,nosuid,nodev,relatime,vers=3.1.1,cache=strict,username=safrc001,uid=9527,forceuid,gid=9527,forcegid,addr=20.209.8.48,file_mode=0777,dir_mode=0777,soft,persistenthandles,nounix,serverino,mapposix,noperm,rsize=1048576,wsize=1048576,bsize=1048576,echo_interval=60,actimeo=1,closetimeo=1)
```

Рис. 15. Результат виконання команди mount | grep clouddrive

```
test [ ~ ]$ stat clouddrive
  File: clouddrive -> /usr/csuser/clouddrive
  Size: 22              Blocks: 0          IO Block: 4096   symbolic link
Device: 7,0     Inode: 11          Links: 1
Access: (0777/lrwxrwxrwx)  Uid: ( 9527/    test)   Gid: ( 9527/    test)
Access: 2025-08-25 21:07:00.635754415 +0000
Modify: 2025-08-25 21:06:59.093753805 +0000
Change: 2025-08-25 21:06:59.094753806 +0000
 Birth: 2025-08-25 21:06:59.093753805 +0000
```

Рис. 16. Результат виконання команди stat clouddrive

```
test [ ~ ]$ mount | grep loop0
/dev/loop0 on /home/test type ext2 (rw,nosuid,nodev,relatime)
```

Рис. 17. Результат виконання команди mount | grep loop0

```
test [ ~ ]$ losetup -a
/dev/loop0: [0115]:13835128424026341376 (/usr/csuser/clouddrive/.cloudconsole/acc_test.img)
```

Рис. 18. Результат виконання команди losetup -a

Безпекові питання використання Storage Account в Cloud Shell також варті уваги. Для того, щоб уникнути ситуацій, коли інші користувачі, що мають доступ до Storage Account, змогли отримати доступ до приватних ключів .ssh, що зберігались там. Тож перед тим, як зберігати там домашній директорій Cloud Shell чутливі дані, потрібно перш переконатись, що тільки авторизовані користувачі мають доступ до Storage Account.

Саме монтування для PowerShell працює таким же чином, як і для Bash і файли також зберігаються у відповідних директоріях. Але на відмінну від Bash, у PowerShell є ще одна додаткова особливість – диск Azure:. Перейшовши у нього користувач має змогу переглядати список ресурсів у форматі, схожому до файлової системи (див. рис.19).

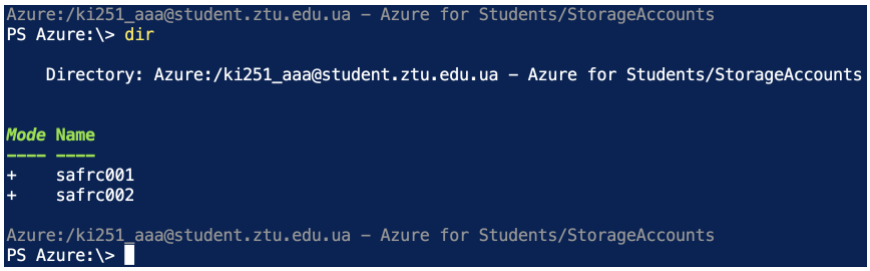


Рис. 19. Перегляд ресурсів з використанням диску Azure:

За замовчуванням сеанси Azure Cloud Shell виконуються в контейнері в мережі Microsoft, яка є окремою від решти ресурсів, які створено в Azure. Команди, що виконуються всередині контейнера, не можуть отримати доступ до ресурсів у приватній віртуальній мережі. Наприклад, користувачі не можуть використовувати Secure Shell (SSH) для підключення з Cloud Shell до віртуальної машини, яка має тільки приватну IP-адресу, або використовувати kubectl для підключення до кластера Kubernetes із заблокованим доступом.

Щоб забезпечити доступ до приватних ресурсів, є можливість розгорнути Cloud Shell у існуючій віртуальній мережі Azure (див. рис.12), обравши опцію “Use an existing private network” під час створення.

Cloud Shell має ряд попередньо встановлених корисних інструментів, що допоможуть вирішити ті чи інші задачі та можуть стати у нагоді користувачеві. Для перегляду встановлених модулів PowerShell необхідно виконати команду Get-Module -ListAvailable, а в Bash та PowerShell tdnf list для перегляду встановлених TDNF додатків та pip3 list для перегляду встановлених Python додатків.

Серед встановлених додатків наявні наступні:

- інструменти для роботи з Azure:
    - Azure CLI;
    - Azure PowerShell;
    - Az.Tools.Predictor;
    - AzCopy;
    - Service Fabric CLI;
- інші сервіси Microsoft:
    - Office 365 CLI;
    - Exchange Online PowerShell;
    - базовий набір модулівMicrosoft Graph PowerShell:
        - Microsoft.Graph.Applications;
        - Microsoft.Graph.Authentication;
        - Microsoft.Graph.Groups;
        - Microsoft.Graph.Identity.DirectoryManagement;
        - Microsoft.Graph.Identity.Governance;
        - Microsoft.Graph.Identity.SignIns;
        - Microsoft.Graph.Users.Actions;
        - Microsoft.Graph.Users.Functions;
    - MicrosoftPowerBIMgmt PowerShell;
    - Модулі SqlServer PowerShell modules;
- інструменти продуктивності:
    - Linux-інструменти:
        - bash;
        - zsh;
        - sh;
        - tmux;
        - dig;
    - текстові редактори:
        - Cloud Shell editor (code);
        - vim;
        - nano;
        - emacs;
- інструменти для роботи з хмарними сервісами:

    - Docker Desktop;
    - Kubectl;
    - Helm;
    - Cloud Foundry CLI;
    - Terraform;
    - Ansible;
    - Chef InSpec;
    - Puppet Bolt;
    - HashiCorp Packer;
- інструменти для розробників:
    - інструменти для побудови додатків:
        - make;
        - maven;
        - npm;
        - pip;
    - системи контролю версій:
        - Git
        - GitHub CLI;
    - інструменти для роботи з базами даних:
        - MySQL client;
        - PostgreSQL client;
        - sqlcmd Utility;
        - mssql-scripter;
    - інструменти для наступних мов програмування:
        - .NET;
        - PowerShell;
        - Node.js;
        - Java;
        - Python;
        - Ruby;
        - Go

Також є можливість встановити власні інструменти, якщо в Cloud Shell використовується Storage Account та якщо вони не вимагають привілейованого доступу root-користувача. Наприклад, користувачі можуть встановити модулі Python, PowerShell, Node.js або встановити щось за допомогою wget.

Важливо також зазначити, що за замовчуванням при запуску Cloud Shell користувач вже є автентифікованим від імені користувача, з якого Cloud Shell запущено. Для перевірки контексту можна скористатись командлетом

Get-AzSubscription у PowerShell або az account show як у Bash, так і в PowerShell (див. рис.20).

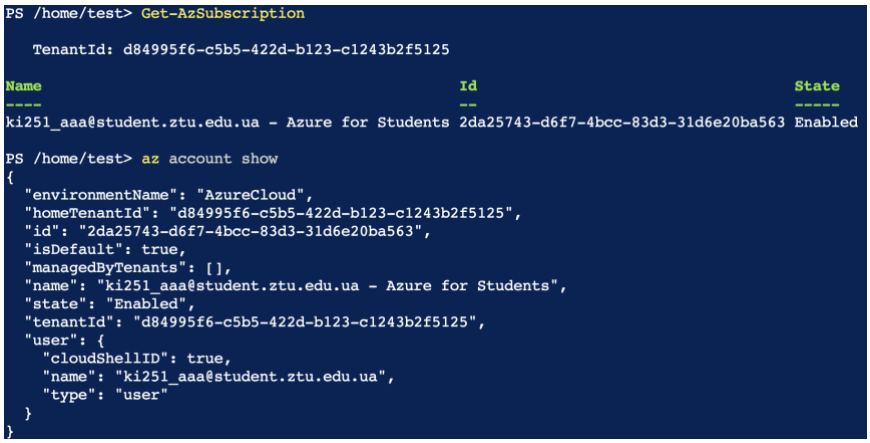


Рис. 20. Перевірка автентифікації в Cloud Shell

Azure CLI – це кросплатформовий інструмент, який спрощує управління ресурсами Azure з командного рядка. Оптимізований для автоматизації та зручності використання, він підтримує інтерактивні сеанси та створення скриптів за допомогою простих команд, які легко інтегруються з моделлю Azure Resource Manager. Користувачі можуть почати використовувати його у браузері за допомогою Azure Cloud Shell або встановити локально.

Інструкція з встановлення Azure CLI доступна на відповідній сторінці документації Microsoft (https://learn.microsoft.com/en-us/cli/azure/install-azure-cli) та містить опції з встановлення для більшості популярних операційних систем (Windows, RHEL/CentOS Stream, SLES/OpenSUSE, Ubuntu/Debian, Azure Linux, MacOS) та запуску в середовищі Docker та Azure Cloud Shell.

Структура команд Azure CLI виглядає наступним чином: reference name - command - parameter - parameter value. Приклад декількох Azure CLI команд:

```
az account set --subscription "my subscription name"
az group create -l westus -n MyResourceGroup
```

А для пошуку необхідних команд можна скористатись відповідними допоміжними командами, типу az vm help або az find vm.

В Azure CLI також наявні глобальні аргументи, які доступні для більшості команд та наведені у таблиці 7.

Таблиця 7

Глобальні аргументи Azure CLI

Аргумент	Опис
--help	Дозволяє переглянути довідку по команді
--output	Зміна виводу результату команди на наступні: json, jsonc, tsv, table, yaml
--query	Фільтрування виводу з використанням JMESPath
--verbose	Вивід деталей виконання команди
--debug	Вивід низькорівневої інформації про виконання REST-запитів для відлагодження
--subscription	Вказання ID або імені Subscription
--only-show-errors	Вимкнення виводу некритичних повідомлень

В Azure CLI наявний так званий interactive mode, який полегшує написання команд завдяками потужному функціоналу автоматичного доповнення команд та пропозицій серед параметрів. Для запуску interactive mode необхідно виконати команду az interactive. Приклад як виглядає interactive mode зображено на рис. 21.

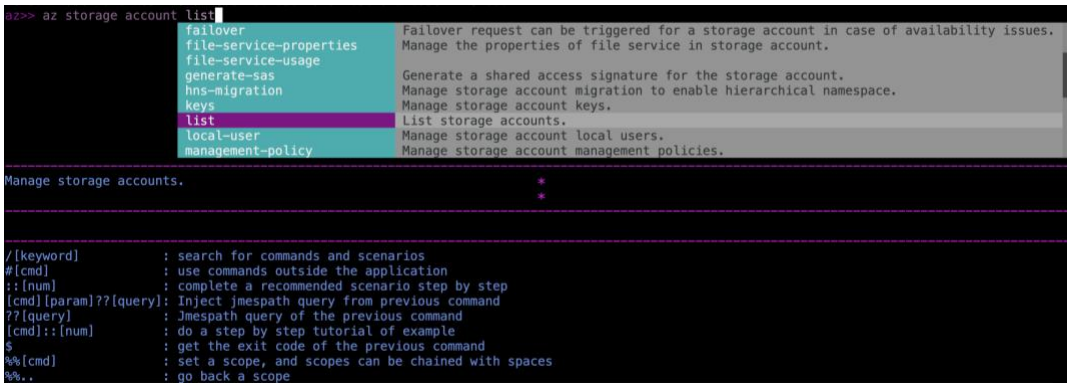


Рис. 21. Azure CLI interactive mode

Схожий функціонал надає доповнення Azure CLI Tools для редактору Visual Studio Code, який доступний за посиланням https://marketplace.visualstudio.com/items?itemName=ms-vscode.azurecli. Для того, щоб Azure CLI Tools почав розпізнавати команди в Azure CLI,

необхідно створити файл з розширенням .azcli. В Azure CLI Tools пропонує наступний функціонал:

- IntelliSense для команд та їх аргументів;
- генерування сніпетів коду з автоматичним вставленням необхідних аргументів;
- виконання поточної команди в інтегрованому терміналі;
- виконання поточної команди та відображення її результату в редакторі;
- відображення документації при наведенні курсору;
- відображення поточної підписки та стандартних налаштувань у рядку стану.

Приклад використання Azure CLI Tools наведено на рис.22.

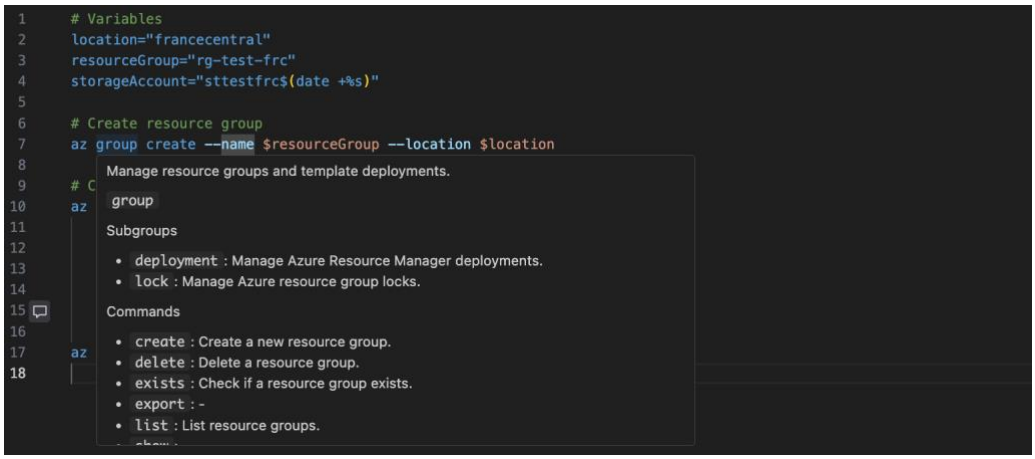


Рис. 22. Приклад застосування Azure CLI Tools у Visual Studio Code

Azure PowerShell – це назва продукту, що об'єднує офіційні модулі Microsoft PowerShell для управління ресурсами Azure. Для його роботи необхідний PowerShell – оболонка командного рядка та мова сценаріїв.

Поточна версія Azure PowerShell – це модуль Az PowerShell. Це рекомендований модуль PowerShell для керування ресурсами Azure за допомогою PowerShell на всіх платформах, включаючи Windows, Linux і macOS. Він містить тисячі команд, які контролюють майже всі аспекти Azure. Модуль Az PowerShell є кросплатформеним. Azure PowerShell рекомендується використовувати з кросплатформеним PowerShell 7 та

вище, хоча він і досі підтримує Windows PowerShell 5.1. Azure PowerShell можжна використовувати в Azure Cloud Shell, локально або в контейнері.

Інструкція для встановлення доступна на відповідній сторінці документації Microsoft (https://learn.microsoft.com/en-us/powershell/azure/install-azure-powershell) та містить опції з встановлення для більшості популярних опереаційних систем (Windows, RHEL/CentOS Stream, SLES/OpenSUSE, Ubuntu/Debian, Azure Linux, MacOS) та запуску в середовищі Docker та Azure Cloud Shell.

Модуль Az PowerShell є модулем-оболонкою для модулів PowerShell, що пов'язані зі службами Azure. Зазвичай один модуль відповідає за окрему службу Azure. Наприклад Az.Network для мережевих служб Azure та Az.Aks для служби Azure Kubernetes Service.

Командлети у модулі Az PowerShell виконують виклики REST до API Azure Resource Manager. Істотні зміни в модулі Az PowerShell обмежуються двома разами на рік. Багато значних змін на рівні API обробляються в командлетах, щоб запобігти їх істотним змінам у самих командлетах, які потенційно могли би змусити користувачів робити зміни у їх скриптах або інших елементах автоматизації.

Модуль Az PowerShell містить командлети для виконання операцій як на рівні управління, так і на рівні даних в Azure. Рівень управління використовується для керування ресурсами у Subscription. Рівень даних використовується для керування можливостями, що надаються ресурсам.

Командлети модулю Az PowerShell повертають об’єкти .NET. Як і будь-який командлет PowerShell, що створює вихідні дані, можна передати пайпом на вхід командлету Get-Member, щоб визначити тип створеного об’єкту. На рис. 23 представлений приклад визначення типу об’єкту.

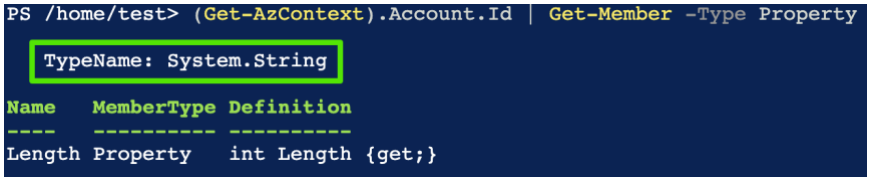


Рис. 23. Визначення типу об'єкту, що повертає командлет Azure PowerShell з використанням командлету Get-Member

Отримати довідку для командлетів Azure PowerShell можна так само, як і для інших вбудованих командлетів PowerShell, використовуючи командлет Get-Help, передавши йому командлет, довідка по якому цікавить користувача. Можна також передати опціональні прапори, типу -Detailed, -Full, -Examples, -Online. Приклад отримання довідки зображений на рис.24 (вивід не є повним).

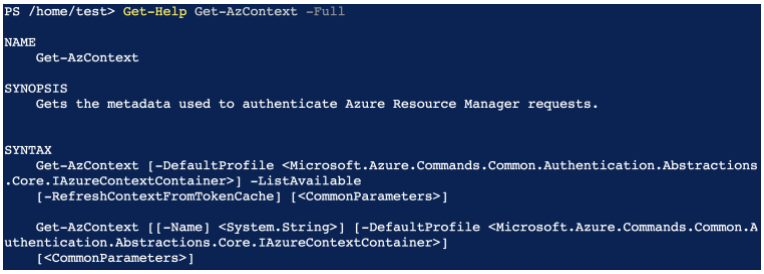


Рис. 24. Використання командлету Get-Help для отримання довідки по командлету Get-AzContext

На відмінну від доповнення для Azure CLI для Visual Studio Code, в магазині доповнень немає виділеного доповнення для Azure PowerShell, проте є доповнення PowerShell, яке також розпізнає модуль Az та допомагає у роботі з ним у редакторі. Приклад роботи автодоповнення Azure PowerShell у Visual Studio Code представлено на рис. 25.

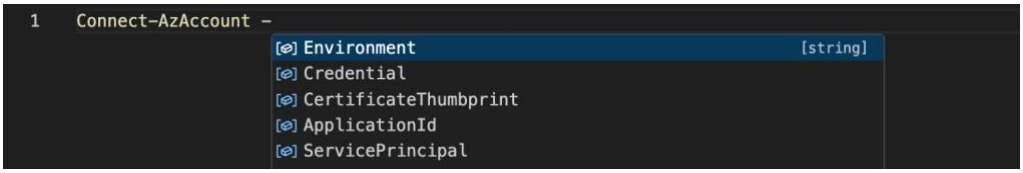


Рис. 25. Автодоповнення командлету в редакторі Visual Studio Code

Власне, перед користувачами Azure може постати логічне запитання: «Тож який інструмент обрати – Azure CLI чи модуль PowerShell?». Направді правильної відповіді для цього запитання немає. Синтаксис Azure

CLI схожий за стилем до скриптів Bash, тому якщо користувач має досвід роботи з Bash та Linux-системами, то використання Azure CLI буде більш нативним. Втім важливо зазначити, що Azure CLI працює і в PowerShell.

В той же час Azure PowerShell є модулем для PowerShell та його використання може бути більш нативним саме для користувачів Windows. Хоча варто зазначити, що PowerShell 7 версії і вище є кросплатформеним і може працювати на операційних системах Linux, Windows та MacOS. В таблиці 8 представлений приклад вирішення одних і тих же задач за допомогою Azure CLI та Azure PowerShell.

Таблиця 8

Порівняння команд Azure CLI та Azure PowerShell

Command	Azure CLI	Azure PowerShell
Create Resource Group	az group create --name <ResourceGroupName> --location eastus	New-AzResourceGroup -Name <ResourceGroupName> -Location eastus
Create Azure Virtual Machine	az vm create --resource-group myResourceGroup --name myVM --image UbuntuLTS --admin-username azureuser --admin-password '<Password>'	New-AzVM -ResourceGroupName <ResourceGroupName> -Name myVM -Image UbuntuLTS -Credential (Get-Credential)
Create Azure Storage Account	az storage account create --name <StorageAccountName> --resource-group <ResourceGroupName> --location eastus --sku Standard_LRS --kind StorageV2	New-AzStorageAccount -Name <StorageAccountName> -ResourceGroupName <ResourceGroupName> -Location eastus -SkuName Standard_LRS -Kind StorageV2

ARM templates – це декларативні JSON-файли, у яких визначається інфраструктура та конфігурація для розгортання в Azure. Вони дозволяють описати ресурси, залежності між ними, параметри й вихідні дані, а потім відтворювати ці розгортання багаторазово з одним результатом.

Основні характеристики ARM templates:

- **декларативний синтаксис**: ARM templates дають змогу декларативно створювати та розгортати всю інфраструктуру Azure. Наприклад, можна розгорнути не лише віртуальні машини, а й мережеву інфраструктуру, системи зберігання даних та будь-які інші ресурси, що необхідні.
- **повторювані результати**: інфраструктура може багаторазово розгортатися протягом усього життєвого циклу розробки з гарантією узгодженості. Шаблони є ідемпотентними, тобто повторне розгортання одного й того ж шаблону призводить до отримання тих самих типів ресурсів у тому самому стані. Користувач може розробити один шаблон, який описує бажаний стан, замість створення багатьох окремих шаблонів для оновлень. Наприклад, файл для створення облікового запису зберігання: якщо під час розгортання такий ресурс із вказаними властивостями вже існує, змін не відбувається;
- **оркестрація**: немає потреби керувати складнощами порядку виконання операцій. Resource Manager самостійно організовує розгортання взаємозалежних ресурсів у правильній послідовності. Якщо можливо, ресурси розгортаються паралельно, що прискорює процес у порівнянні з послідовними розгортаннями. Для розгортання достатньо однієї команди, а не кількох імперативних;
- **модульні файли**: шаблони можна розбивати на менші, багаторазово використовувані компоненти та зв’язувати їх під час розгортання. Дозволяється також вкладати один шаблон в інший;
- **створення будь-якого ресурсу Azure**: нові сервіси та функції Azure можна використовувати відразу ж у шаблонах. Щойно ресурсний провайдер представляє новий ресурс, його вже можна розгорнути через ARM templates, не чекаючи оновлення інструментів чи модулів;
- **розширюваність**: за допомогою deployment scripts користувач може додавати PowerShell або Bash-скрипти до шаблонів. Це розширює можливості налаштування ресурсів під час розгортання. Скрипт може

бути включений у сам шаблон або зберігатися зовнішньо й підключатися за посиланням. Таким чином, можна виконати повне налаштування середовища в межах одного ARM template;

- **тестування**: для перевірки відповідності шаблону рекомендованим практикам застосовується ARM Template Tool Kit (arm-ttk). Це PowerShell-скрипт, доступний на GitHub, який спрощує навчання роботі з мовою шаблонів.
- **попередній перегляд змін**: операція *what-if* дає змогу переглянути зміни перед розгортанням. Вона показує, які ресурси буде створено, оновлено або видалено, а також які властивості ресурсів буде змінено. What-if враховує поточний стан середовища, що знімає необхідність окремо керувати станом;
- **вбудована перевірка**: шаблон розгортається лише після успішної перевірки. Resource Manager перевіряє коректність перед запуском розгортання, щоб зменшити ризик зупинки у «напівготовому» стані;
- **відстежувані розгортання**: в Azure Portal можна переглянути історію розгортань і деталі кожного шаблону. Відображаються шаблон, використані параметри та вихідні значення. Інші сервіси *infrastructure as code* у Portal не відстежуються;
- **Policy as code**: Azure Policy реалізує підхід policy as code для автоматизації governance. Якщо застосовуються політики, то під час розгортання шаблонів автоматично виконується remediation для невідповідних ресурсів;
- **Deployment Blueprints**: користувач може використовувати готові Blueprints від Microsoft для дотримання регуляторних і безпекових вимог. Blueprints містять заздалегідь підготовлені шаблони для типових архітектур;
- **інтеграція з CI/CD**: шаблони інтегруються в процеси безперервної інтеграції та розгортання. Це дає змогу автоматизувати release-

пайплайни для швидкого й надійного оновлення застосунків та інфраструктури. Використовуючи Azure DevOps і Resource Manager template task, можна задіяти Azure Pipelines для безперервної збірки та розгортання ARM-проєктів;

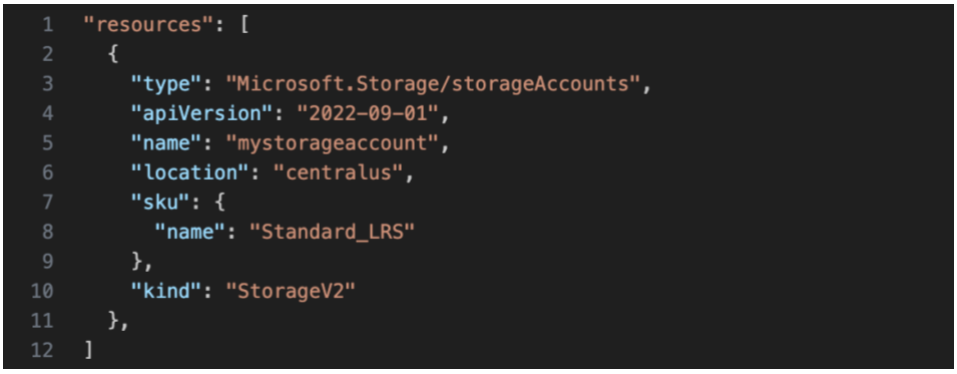
- **експортований код**: для наявної Resource Group можна отримати шаблон, експортувавши її поточний стан або переглянувши шаблон конкретного розгортання. Це корисний спосіб навчитися синтаксису ARM templates.
- **інструменти для створення**: ARM templates можна створювати у Visual Studio Code з розширенням Template Tools. Доступні IntelliSense, підсвітка синтаксису, вбудована довідка та інші функції. Також підтримується Visual Studio.

У шаблоні можна використовувати template expressions, які розширюють можливості JSON. Такі вирази застосовують функції, що надає Resource Manager та їх вичерпний список доступний за посиланням https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/template-functions.

ARM template містить такі секції:

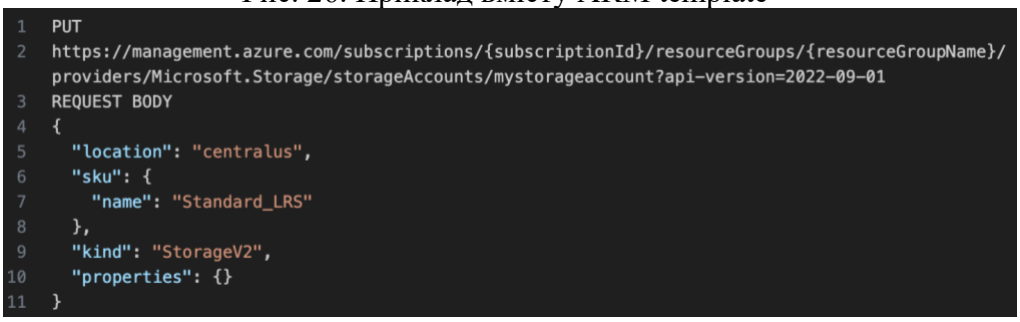
- **Parameters** – значення, що задаються під час розгортання. Вони дають змогу користувачу налаштовувати один і той самий шаблон для різних середовищ;
- **Variables** – значення, які багаторазово використовуються в межах шаблону. Їх можна формувати на основі параметрів;
- **User-defined functions** – користувацькі функції, які спрощують роботу з шаблоном;
- **Resources** – опис ресурсів, що підлягають розгортанню;
- **Outputs** – значення, що повертаються з розгорнутих ресурсів.

Під час розгортання шаблону Resource Manager перетворює його у запити REST API. Наприклад, якщо Resource Manager отримує значення, як на рис 26, на REST API запит, як на рис. 27.



```
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "apiVersion": "2022-09-01",
    "name": "mystorageaccount",
    "location": "centralus",
    "sku": {
      "name": "Standard_LRS"
    },
    "kind": "StorageV2"
  },
]
```

Рис. 26. Приклад вмісту ARM template



```
PUT
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/
providers/Microsoft.Storage/storageAccounts/mystorageaccount?api-version=2022-09-01
REQUEST BODY
{
  "location": "centralus",
  "sku": {
    "name": "Standard_LRS"
  },
  "kind": "StorageV2",
  "properties": {}
}
```

Рис. 27. Конвертований у REST API запит

Для створення ARM template рекомендується використовувати Visual Studio Code та доповнення Azure Resource Manager tools extension, яке доступне за посиланням https://marketplace.visualstudio.com/items?itemName=msazurermtools.azurerm-vscode-tools.

Далі створюється новий файл із назвою azuredeploy.json та відкривається у Visual Studio Code. У редакторі коду достатньо ввести arm, після чого з’являться готові Azure Resource Manager сніпети для швидкого створення ARM template. Необхідно обрати arm!, щоб створити шаблон, призначений для розгортання у межах Azure resource group (див рис. 28).

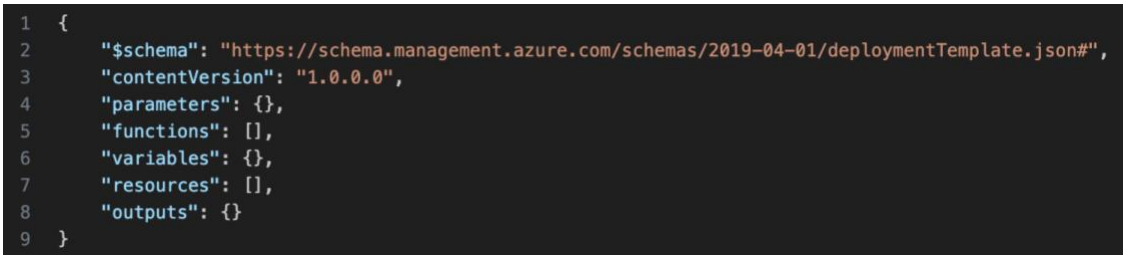


Рис. 28. Створений за шаблоном стартовий ARM template

Наступним кроком буде додавання ресурсу у шаблон. У блоці resources потрібно ввести storage та обрати сніпет arm-storage. Ця дія додає ресурс Storage Account до шаблону та після модифікацій значень за замовчуванням можна отримати наступний опис специфікації Storage Account (див. рис.29).

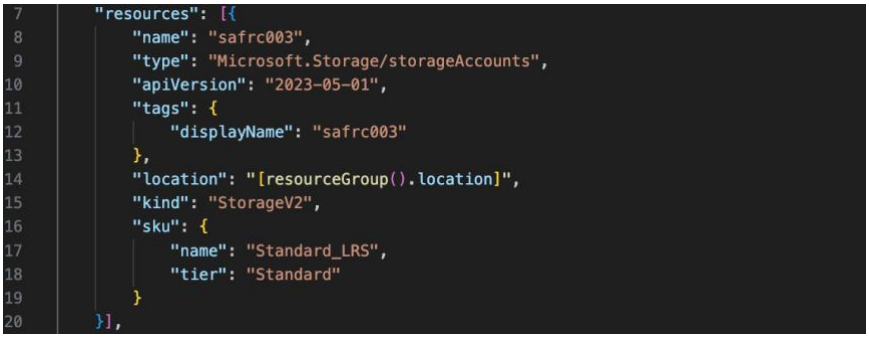


Рис. 29. Вказані параметри для розгортання Storage Account

Однією з переваг використання доповнення є валідація значень, що вводяться. При спробі ввести некоректне значення, типу SKU Tier, розширення згенерує попередження про недопустиме значення.

Наступним кроком буде додавання файлу з параметрами. Файл параметрів дозволяє зберігати значення, специфічні для певного середовища (наприклад, test або production), та передавати їх під час розгортання.

У Visual Studio Code достатньо клацнути правою кнопкою на шаблоні та обрати Select/Create Parameter File. Потім – New > All Parameters, вказати ім’я та розташування файлу.

Новий файл параметрів буде прив’язаний до шаблону, і розширення виконуватиме валідацію обох файлів одночасно.

Наприклад, якщо у файлі параметрів для storageAccountName ввести значення з двох символів, з’явиться помилка, адже воно не відповідає вимогам minLength. Після виправлення на коректне значення помилка зникає. Приклад додавання параметру зображено на рис. 30.

```
4     "parameters": {"storageAccountName": {
5         "type": "string",
6         "metadata": {
7             "description": "Storage Account Name"
8         },
9         "minLength": 3,
10        "maxLength": 24
```

Рис. 30. Додавання параметру storageAccountName

Далі потрібно вказати місця, де буде використовуватись створений параметр (див. рис. 31).

```
16    "resources": [
17        {
18            "name": "[parameters('storageAccountName')]",
19            "type": "Microsoft.Storage/storageAccounts",
20            "apiVersion": "2023-05-01",
21            "tags": {
22                "displayName": "[parameters('storageAccountName')]"
23            },
24            "location": "[resourceGroup().location]",
25            "kind": "StorageV2",
26            "sku": {
27                "name": "Standard_LRS",
28                "tier": "Standard"
29            }
30        }
31    ],
```

Рис. 31. Використання створеного параметру

Після цього вказується файл з параметрами та заповнюється його значення (див. рис.32).

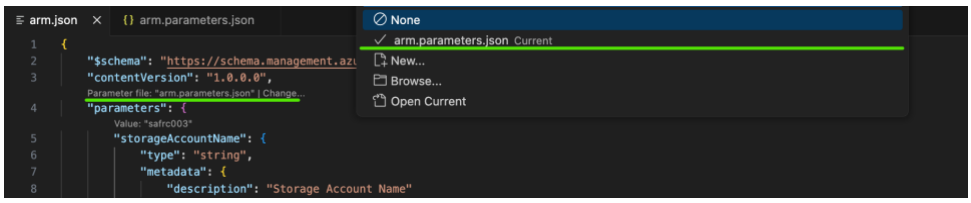


Рис. 32. Вказання файлу з параметрами

Наступним кроком можна вказати значення для попередньо створеного параметру, що буде використовуватись (див. рис. 33).

Рис. 33. Вказання значення параметру storageAccountName

Переконавшись що файли готові до застосування та розгортання ресурсів, виконується наступна команда, представлена на рис. 34.

Рис. 34. Команда для розгортання ресурсу з застосуванням ARM template

Після виконання даної команди можна переконатись у її успішному виконанні за виводом у інтерфейсі командного рядку, який не містить помилок (див. рис.35).

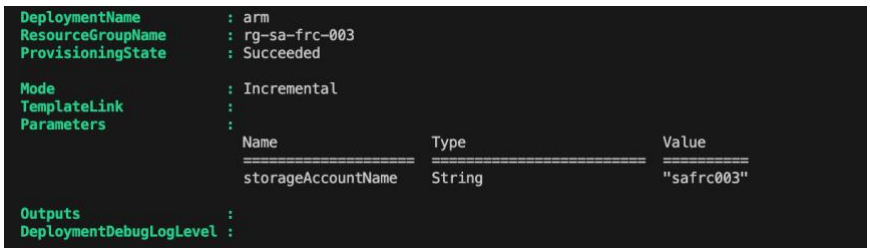


Рис. 35. Успішно створене розгортання Storage Account з використанням ARM template

Завдання на лабораторну роботу

1. На Azure Portal запустити Azure Cloud Shell та під час створення обрати опцію "Mount storage account" та "I want to create a storage account". Далі в процесі створення задати наступні значення:
   - Resource group, яка буде називатись rg-cs-sa-xxx-yyy-zzz, де xxx – назва регіону за варіантом, згідно даних табл.9, а yyy – ініціали студенти, zzz – номер варіанту (наприклад, для 15 варіанту значення буде 015);
   - Region – значення за варіантом, згідно даних табл. 9;

- Storage account name – stcloudshellxxxyyyzzz, де xxx – назва регіону за варіантом, згідно даних табл.9, yyy – ініціали студента, а zzz – номер варіанту;

- File share – fscloudshellxxxyyyzzz, де xxx – назва регіону за варіантом, згідно даних табл.9, yyy – ініціали студента, а zzz – номер варіанту. До звіту додати результат виконаного завдання, зі скріншотами, де можна побачити запущений Cloud Shell та створений Azure Storage Account.

2. В середовищі Azure Cloud Shell обрати Bash та використовуючи Azure CLI, створити Resource group, що буде називатись rg-data-sa-xxx-yyy-zzz, де xxx – назва регіону за варіантом, згідно даних табл.9, а yyy – ініціали студенти, zzz – номер варіанту; всередині даної Resource group створити Azure Storage Account, який буде називатись stdataxxxyyyzzz та розміщений у регіоні згідно даних табл. 9. До звіту додати використані команди Azure CLI та скріншот створеного Storage Account, показавши, що він створений у заданій Resource group.
3. В середовищі Azure Cloud Shell обрати PowerShell та використовуючи Azure PowerShell створити Resource group, яка буде називатись rg-vnet-xxx-yyy, де xxx – назва регіону за варіантом, згідно даних табл.9, а yyy – ініціали студенти, zzz – номер варіанту; всередині даної Resource group створити Azure Virtual Network, яка буде називатись vnetxxxyyyzzz, де xxx – назва регіону за варіантом, згідно даних табл.9, а yyy – ініціали студенти, zzz – номер варіанту та розміщена у регіоні згідно даних табл. 9. До звіту додати використані командлети Azure PowerShell та скріншот створеної Azure Virtual Network, показавши, що її було створено у заданій Resource group.
4. Створити два ARM templates – один для розгортання Resource group, яка буде називатись rg-kv-xxx-zzz-yyy, де xxx – назва регіону за варіантом, згідно даних табл.9, а yyy – ініціали студенти, zzz – номер варіанту і другий для створення Azure Key Vault всередині цієї Resource group,

який буде називатись kvxxxyyyzzz, де xxx – назва регіону за варіантом, згідно даних табл.9, а yyy – ініціали студенти, zzz – номер варіанту та розміщена у регіоні згідно даних табл. 9. ARM templates мають бути параметризовані та містити правила, які відповідають правилами іменування ресурсів в Azure. Для значень параметрів має бути створені Parameter files. Запустити розгортання ARM templates, використовуючи Azure CLI або Azure PowerShell (що саме використовувати – зазначено у табл. 9). До звіту додати вміст ARM templates, Parameter files, команди для запуску розгортання Resource group та Azure Key Vault, скріншоти результатів виконання розгортань у Azure CLI чи Azure PowerShell, скріншоти, що демонструють, що ресурси були створені.

5. Використовуючи Azure Portal, видалити усі створені впродовж виконання попередніх завдань ресурси. Надати скріншоти, що демонструють успішне видалення усіх ресурсів.

Таблиця 9

Таблиця з даними для завдань 1-4

№ .	1 завдання (регіон)	2 завдання (регіон)	3 завдання (регіон, мережа, підмережа)	4 завдання (регіон)
1	swedencentral (sdc)	norwayeast (nwe)	germanywestcentral (gwc), VNet: 10.1.1.0/24; Subnet: 10.1.1.0/25	switzerlandnorth (szn)
2	norwayeast (nwe)	germanywestcentral (gwc)	switzerlandnorth (szn), VNet: 172.16.2.0/24; Subnet: 172.16.2.128/25	francecentral (frc)
3	germanywestcentral (gwc)	switzerlandnorth (szn)	francecentral (frc), VNet: 192.168.3.0/24; Subnet: 192.168.3.0/26	swedencentral (sdc)
4	switzerlandnorth (szn)	francecentral (frc)	swedencentral (sdc), VNet: 10.1.4.0/24; Subnet: 10.1.4.0/25	norwayeast (nwe)

Продовження таблиці 9

5	francecentral (frc)	swedencentral (sdc)	norwayeast (nwe), VNet: 172.16.5.0/24; Subnet: 172.16.5.128/25	germanywestcentral (gwc)
6	swedencentral (sdc)	norwayeast (nwe)	germanywestcentral (gwc), VNet: 192.168.6.0/24; Subnet: 192.168.6.0/26	switzerlandnorth (szn)
7	norwayeast (nwe)	germanywestcentral (gwc)	switzerlandnorth (szn), VNet: 10.1.7.0/24; Subnet: 10.1.7.0/25	francecentral (frc)
8	germanywestcentral (gwc)	switzerlandnorth (szn)	francecentral (frc), VNet: 172.16.8.0/24; Subnet: 172.16.8.128/25	swedencentral (sdc)
9	switzerlandnorth (szn)	francecentral (frc)	swedencentral (sdc), VNet: 192.168.9.0/24; Subnet: 192.168.9.0/26	norwayeast (nwe)
10	francecentral (frc)	swedencentral (sdc)	norwayeast (nwe), VNet: 10.1.10.0/24; Subnet: 10.1.10.0/25	germanywestcentral (gwc)
11	swedencentral (sdc)	norwayeast (nwe)	germanywestcentral (gwc), VNet: 172.16.11.0/24; Subnet: 172.16.11.128/25	switzerlandnorth (szn)
12	norwayeast (nwe)	germanywestcentral (gwc)	switzerlandnorth (szn), VNet: 192.168.12.0/24; Subnet: 192.168.12.0/26	francecentral (frc)
13	germanywestcentral (gwc)	switzerlandnorth (szn)	francecentral (frc), VNet: 10.1.13.0/24; Subnet: 10.1.13.0/25	swedencentral (sdc)
14	switzerlandnorth (szn)	francecentral (frc)	swedencentral (sdc), VNet: 172.16.14.0/24; Subnet: 172.16.14.128/25	norwayeast (nwe)

Продовження таблиці 9

15	francecentral (frc)	swedencentral (sdc)	norwayeast (nwe), VNet: 192.168.15.0/24; Subnet: 192.168.15.0/26	germanywestcentral (gwc)
16	swedencentral (sdc)	norwayeast (nwe)	germanywestcentral (gwc), VNet: 10.1.16.0/24; Subnet: 10.1.16.0/25	switzerlandnorth (szn)
17	norwayeast (nwe)	germanywestcentral (gwc)	switzerlandnorth (szn), VNet: 172.16.17.0/24; Subnet: 172.16.17.128/25	francecentral (frc)
18	germanywestcentral (gwc)	switzerlandnorth (szn)	francecentral (frc), VNet: 192.168.18.0/24; Subnet: 192.168.18.0/26	swedencentral (sdc)
19	switzerlandnorth (szn)	francecentral (frc)	swedencentral (sdc), VNet: 10.1.19.0/24; Subnet: 10.1.19.0/25	norwayeast (nwe)
20	francecentral (frc)	swedencentral (sdc)	norwayeast (nwe), VNet: 172.16.20.0/24; Subnet: 172.16.20.128/25	germanywestcentral (gwc)

Контрольні питання

1. Що таке Azure Resource Manager і яку роль він відіграє у життєвому циклі ресурсів в Azure?
2. Які функції управління ресурсами надає ARM після створення ресурсу (назвіть не менше двох)?
3. Як ARM обробляє запити користувача, що надсилаються через Azure CLI, PowerShell чи Portal?
4. Що таке Resource Provider? Наведіть приклади двох найбільш поширених.
5. Чим відрізняється extension resource від звичайного ресурсу? Наведіть приклад.
6. Які основні можливості надає Azure Portal для управління Subscription?
7. Для чого потрібен Azure Cloud Shell і які оболонки в ньому можна обрати?
8. Яка роль Storage Account у роботі Cloud Shell? Що відбувається з файлами, якщо його не використовувати?
9. Наведіть приклад команди створення Resource Group в Azure CLI та в Azure PowerShell.
10. Що таке ARM template, які основні його секції та переваги використання?