Rutendo chapfika

ST10205167 || Part 1

Contents

[Part A 2](#_Toc144130894)

[Part B 9](#_Toc144130895)

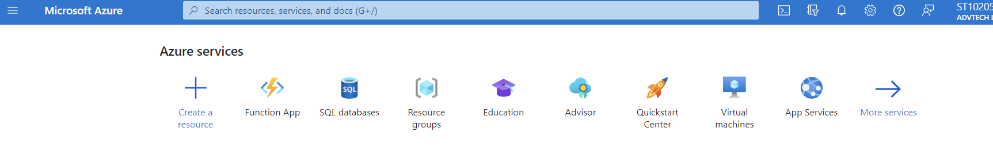
[References 10](#_Toc144130896)

# Part A

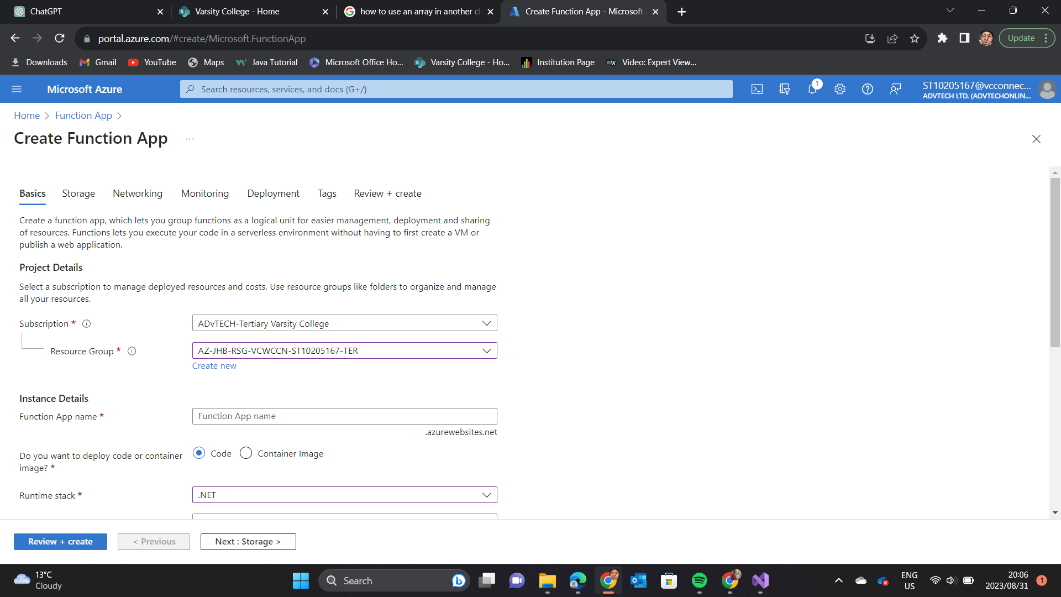
|  |  |  |  |
| --- | --- | --- | --- |
| **Traditional On-Premises** | | **Morden Cloud** | |
| **On-premises**  **definition** | **On-premises**  **example** | **Cloud definition** | **Cloud example** |
| **Monolithic:**  A classic, self-contained software application architecture known as a monolithic on-premises software system functions within the physical infrastructure of an organisation, with all components tightly integrated into a single unit. For code to be executed or compiled and for software to run under a monolithic architecture, each component and all of its related components must be present. (Rahul Awati, 2023) | **Monolithic:**  Take a monolithic SaaS e-commerce application. It might include a web server, a load balancer, a service that serves up product images from a catalogue, an ordering platform, a payment option, and a delivery element. Given their broad application, monolithic tools typically have enormous code bases. The entire platform may need to be compiled and tested in order to make a simple modification to a single function, which is contrary to the agile methodology that most developers use today. (talend, 2023) | **Decomposed:**  Decomposed clouds are sophisticated solutions that have been broken down into evaluable functional primitives or components. This method aids in evaluating the advantages of each component in the context of multi-cloud or hybrid cloud configurations. (Linthicum, 2023) | **Decomposed:**  Consider reworking the company's CRM for the cloud and breaking it down into smaller applications dedicated to specific functions that can be moved over the process by process method rather than taking a lift-and-shift approach that replicates an internal application in a cloud environment. (Linthicum, 2023) |
| **Designed for**  **predictable**  **scalability:**  In order to effectively use software or apps, adequate speed is also required. To address this expanding burden, cloud scalability is used. Where the constant deployment of resources is necessary to handle the workload statically, scalability is frequently used. Elasticity is used to meet the organization's dynamic needs, whereas scalability is used to meet its static needs. (madhav\_mohan, 2023) | **Designed for**  **predictable**  **scalability:**  Consider that you are the owner of a business whose database size was initially small but has since grown as your business has grown. In this situation, all you need to do is ask your cloud service provider to increase your database's capacity in order to handle a heavy workload. (madhav\_mohan, 2023) | **Designed for elastic**  **scale:**  This is a function of cloud elasticity. It is the capacity to automatically increase or decrease resources in response to demand. It is dynamic and reacts to current developments. With this approach, the actual cloud service process and real-time cloud expenditures are more tightly aligned. (Beschokov, 2023) | **Designed for elastic**  **scale:**  AWS Elastic Beanstalk [6] is a contemporary cloud service that perfectly demonstrates elastic scaling. It controls the amount of space needed to execute web applications automatically. It scales up when demand increases by adding additional resources, and it scales down when demand declines to save expenses. Due to the assurance of ideal performance and cost-effectiveness, it serves as a prime illustration of cloud elasticity in action. (Beschokov, 2023) |
| **Relational Database:**  A database management system (RDBMS) that is installed and used locally, such as in a company's data centre, is referred to as an on-premises relational database. It runs on the company's own hardware and servers rather than being hosted in a cloud environment. (Lutkevich & Biscobing, 2023) | **Relational Database:**  Consider a sizable organisation that keeps track of its staff data in an on-site relational database. In this case, the company installs an RDBMS within its own data centre, such as Oracle Database or Microsoft SQL Server. This on-premises RDBMS stores employee data in tables, including names, roles, salaries, and contact information. Due to its on-premises location, the business has complete control over security measures like firewalls, access controls, and encryption to ensure that critical employee data is safeguarded. Employees and authorised personnel can use secure network connections to access and update the database from within the corporate network. (Lutkevich & Biscobing, 2023) | **Polyglot persistence:**  The practise of utilising many data storage systems within a single application or system is known as "polyglot persistence." Instead of relying on a single, universal storage solution, it entails choosing the best database or storage technology for each distinct component or type of data within the application. (Brunskill, 2023) | **Polyglot persistence:**  Think of a cutting-edge cloud-based e-commerce platform. In order to handle the rapidly expanding user data, this platform may leverage polyglot persistence, where user profiles and authentication information are kept in a highly scalable NoSQL database like MongoDB. A relational database like PostgreSQL is used to store product data because of its excellent consistency and structured querying capabilities. Product data requires complex querying and transactions. A distributed log storage system like Apache Kafka is used to store logs of user activity and clickstream data for real-time processing and analytics. For low-latency access, session management data is stored in an in-memory database like Redis. (Brunskill, 2023) |
| **Synchronized processing:**  The coordination and execution of activities or operations in a way that guarantees their synchronised or coordinated occurrence is referred to as synchronous processing. It frequently refers to many processing or threading activities occurring simultaneously in a single computing system. (D., 2023) | **Synchronized processing:**  Multiple client apps may need to access and change a common database at the same time in a database systems situation. Data integrity is protected via synchronised processing techniques like locks and transactions. For instance, synchronised processing avoids conflicts and data corruption when two applications attempt to edit the same database entry. (D., 2023) | **Asynchronized processing:**  Asynchronized processing is a mode of operation that enables more concurrency and responsiveness in applications by executing tasks or actions independently of one another. This method makes it possible for programmes to keep running even when specific processes take a long time, making it a vital idea in the construction of scalable and effective cloud-based systems. (Lutkevich, Asynchronous, 2023) | **Asynchronized processing:**  A cloud-based e-commerce site might be a suitable illustration. When a consumer puts an order, the system might need to carry out a number of operations, including inventory updating, payment processing, and sending an email of confirmation. These tasks are managed separately in an asynchronous paradigm. For instance, the system may acknowledge receiving the order and then queue the updating of the inventory, handling of payments, and sending of emails as distinct asynchronous jobs. As a result, the system can perform each activity in the most effective manner without having to wait for one to finish before beginning the next, ensuring that the consumer receives an order confirmation swiftly. (Lutkevich, Asynchronous, 2023) |
| **Design to avoid failures(MTBF):**  The average period of time between a system's or product's repairable failures is known as the "mean time between failure," or MTBF. A few of the strategies involved in designing to avoid failures and increase MTBF are as follows:  High-quality materials and components should be used to construct the system, Implement redundancy in critical components. Stress Testing run stress testing on the system to find weak spots and potential failure modes. Install sensors and monitoring devices to keep track of the system's health and receive alerts. (nextservicesoftware, 2023) | **Design to avoid failures(MTBF):**  Consider a major hospital that uses an expensive piece of medical equipment—say, an EKG machine—to measure patients' cardiac impulses 16 hours a day, seven days a week. The EKG machine has malfunctioned five times during regular business hours over the past six months (26 weeks), necessitating a four-hour outage each time to identify the problem and rectify it.  26 weeks times 7 days times 16 hours equals 2912 hours, less the downtime of 5 occasions times 4 hours equals 20 hours. So, with 5 failures, our total uptime is 2892 hours. The hospital will purchase high-quality server hardware, install redundant power supplies and cooling systems, perform stress tests to find potential problems, perform routine server maintenance, and use real-time monitoring tools to track server performance and get alerts in case of anomalies in order to maximise MTBF. (nextservicesoftware, 2023) | **Design for failure (MTTR):**  The mean time it takes to repair a system or service after a failure or incident is measured by the Mean Time to Repair (MTTR) metric, which is essential in cloud computing. Services in the cloud are created with the understanding that failures can and will happen. (splunk, 2023) | **Design for failure (MTTR):**  An example of how MTTR is built to fail:  Scenario, On a cloud platform, a well-known e-commerce website is housed. The following procedures are used to provide high availability and a short mean time to repair (MTTR) in the event of failure. To provide redundancy, key website elements like servers and databases are copied across various availability zones. (splunk, 2023) |
| **Occasional large updates:**  Having "occasional large updates" in the context of on-premises infrastructure often refers to infrequent but significant updates to the software or systems running on local servers or within the physical data centre of an organisation. These upgrades frequently include important adjustments, enhancements, or new features. Large-scale modifications to on-premises infrastructure every now and then show that an organisation is managing and improving its IT systems with thought and caution. These modifications aim to significantly enhance the infrastructure while reducing the possibility of disruptions and assuring its stability and security. (Chao, 2014) | **Occasional large updates:**  A medium-sized manufacturing business that runs its activities through an on-site IT infrastructure. On its local servers, this business has a sophisticated Enterprise Resource Planning (ERP) system set up. They employ a technique of "occasional large updates" because of the crucial nature of this system. The way it works is that the corporation rarely updates its ERP system. They instead choose to make significant upgrades once or twice a year. The interruption brought on by frequent updates is lessened by this strategy. The business may make considerable modifications to the ERP system during these major updates. For instance, they could update to a newer version of the programme that offers more sophisticated supply chain management tools. In this case, the company's strategy for "occasional large updates" to its on-premises ERP system ensures that they may take advantage of the newest features and advancements while preserving the consistency and dependability of their business operations. They can manage upgrades using this tactic to reduce disruptions and keep a safe and effective infrastructure. (Chao, 2014) | **Frequent small updates:**  Making frequent, small updates is a software development and deployment strategy used in contemporary cloud computing to improve agility, dependability, and efficiency. Agile development is now being used by development teams to break down more significant software changes into smaller, more manageable steps. It is simpler to design, test, and roll out these smaller updates. This strategy also includes improved user experience and continuous integration and deployment (CI/CD). In the modern cloud ecosystem, releasing frequent, minor updates is a method to deliver software more effectively, more qualitatively, and in harmony with shifting customer requirements. It's a cornerstone of the agile and DevOps approaches. (Chao, 2014) | **Frequent small updates:**  Consider a business that offers a cloud-based e-commerce platform. They frequently release minor upgrades to keep the platform secure and competitive. To quickly fix vulnerabilities, this strategy entails applying security fixes on a regular basis. constantly updating features in response to customer input and market trends. For instance, streamlining the checkout procedure to enhance client satisfaction. modifying the application's code in a modest way to speed up loading or use less server resources. responding quickly to small faults or issues that users have reported. modifying server capacity to effectively handle varying load. Modern cloud computing is known for its agile approach, which encourages adaptation and user pleasure. (Chao, 2014) |
| **Manual Management:**  In place of depending on cloud-based automated management, manual management refers to the practise of manually controlling and maintaining IT resources, such as servers, storage, and software, within an on-premises (local) data centre. This method requires more physical control of hardware and is less automated. (morefield.com, 2023) | **Manual Management:**  If a business chooses manual management for its server infrastructure in an on-premises data centre, it will need an IT crew in charge of physically maintaining and fixing servers. They have to replace any failed server parts by hand. Patches and upgrades for software are manually applied. Updates must be scheduled ahead of time by IT staff, which may briefly disrupt services. Firewalls and intrusion detection systems are two examples of security solutions that need manual configuration and monitoring. IT managers must manually assign resources like RAM and CPU to various services and apps. Although manual management can take longer and sometimes lacks the scalability and flexibility benefits of cloud-based management, it can provide more direct control over resources. (morefield.com, 2023) | **Automated self-management:**  Modern cloud architecture uses the term automated self-management to describe the ability of cloud systems to autonomously monitor, optimise, and maintain themselves without needing constant human involvement. A crucial component of cloud-native and autonomous computing is this idea. (Chao, 2014) | **Automated self-management:**  A cloud-based e-commerce platform with automatic self-management would be a nice illustration of this. A cloud-based system would be used by this platform to continuously monitor web traffic, CPU use, and memory usage. During a promotion or other time of high demand, the system automatically recognises when traffic is surging. The system automatically creates extra virtual servers to manage the increasing load based on predetermined constraints, such as maintaining a specific threshold of CPU use below 80%. Automated self-management is demonstrated with this real-time autoscaling example. Without human involvement, the cloud infrastructure dynamically modifies itself to meet demand, improving performance and cost-effectiveness. (Chao, 2014) |
| **Snowflake servers:**  Snowflake servers are those whose proposed configurations have considerably changed from their actual ones. Due to their unique settings, these servers can present complications for IT management and DevOps procedures in terms of consistency and maintenance. (Churchman, 2023) | **Snowflake servers:**  Consider a business that has a sizable server infrastructure. Server A, the organization's oldest production server, has been in operation for many years. Its configuration has been manually altered throughout time by many administrators to satisfy particular software specifications and hardware constraints. These modifications were not recorded. Because Server A's configuration drastically deviates from the standardised configuration used for more recent servers (Server B, Server C, etc.), it is now referred to as a snowflake server. It is difficult to manage and update Server A since it necessitates specialised knowledge, and modifications must be gradually implemented to prevent affecting vital services. Snowflake servers, such as Server A, might increase maintenance costs, provide operational risks, and make troubleshooting more challenging. (Churchman, 2023) (Yadav, 2023) | **Immutable infrastructures:**  In a current cloud architecture concept known as immutable infrastructure, server instances or virtual machines (VMs) are never changed after being initially deployed. Any upgrades or modifications are made via generating new instances with the necessary specifications rather than making changes directly to servers. This strategy makes guarantee of consistency and dependability and streamlines deployment procedures. Immutable infrastructure is a useful strategy in contemporary cloud environments where scalability and dependability are essential since it offers advantages including improved security, simpler rollbacks, and predictable deployments. (Bigelow, 2023) (Dadgar, 2023) | **Immutable infrastructures:**  Think about a web application that is stored in the cloud. The application is deployed utilising a specified server image or virtual machine configuration in an immutable infrastructure environment. A new server instance is created with the updated configuration or software version whenever it becomes necessary to upgrade the programme or its dependencies. Traffic is switched progressively or totally to the new instance after testing and confirming that it operates as intended. The old instance is terminated to ensure that no modifications or patches were made to it once all traffic has been transferred to the new instance and it has been verified to be operating as intended. (Bigelow, 2023) (Dadgar, 2023) |

# Part B

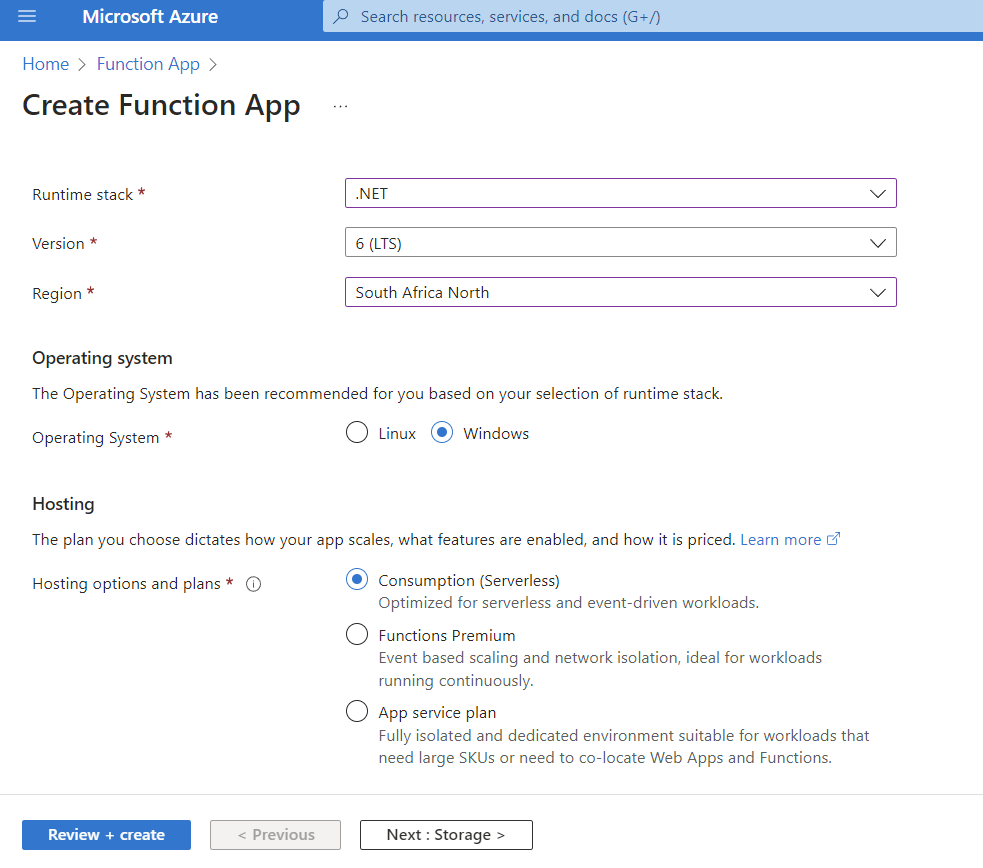
After creating your Azure function and it works, do the following steps to upload your function on Azure.



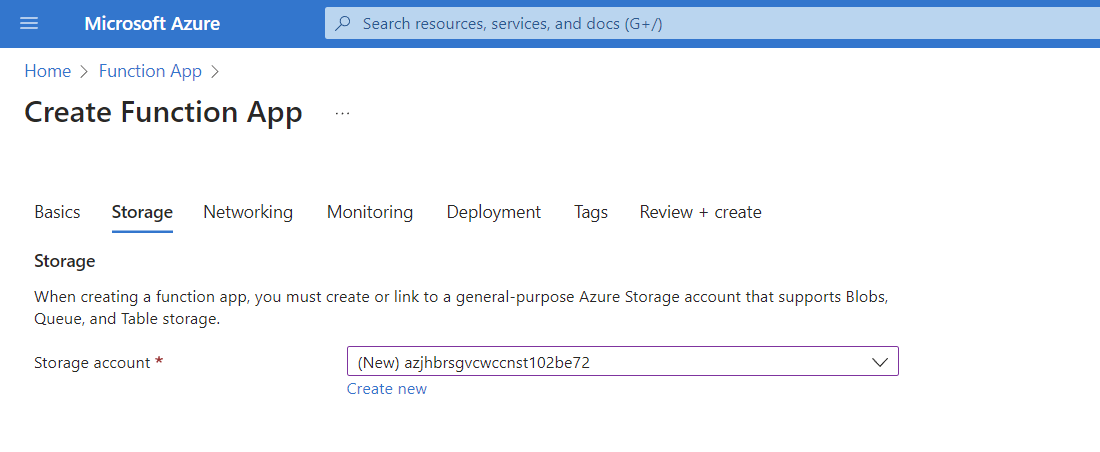
Log onto Azure and select Function App.



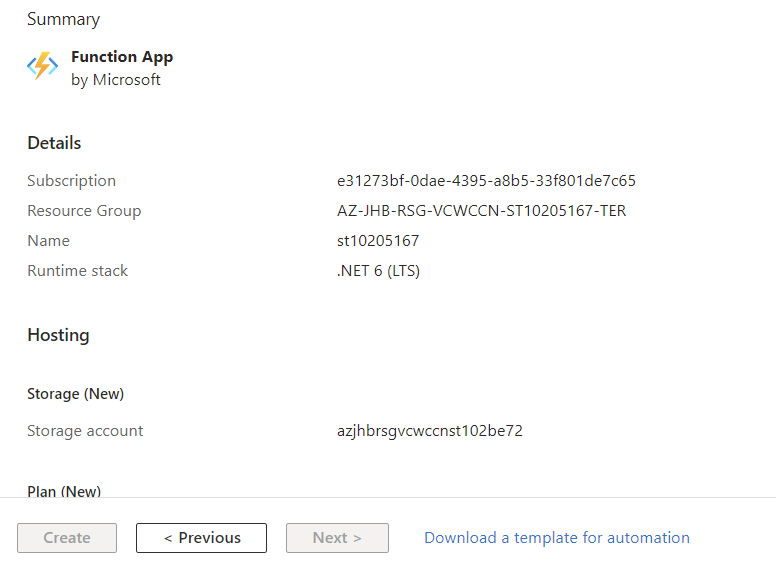
When creating makes sure you choose your resource group allocated to you via your student no.

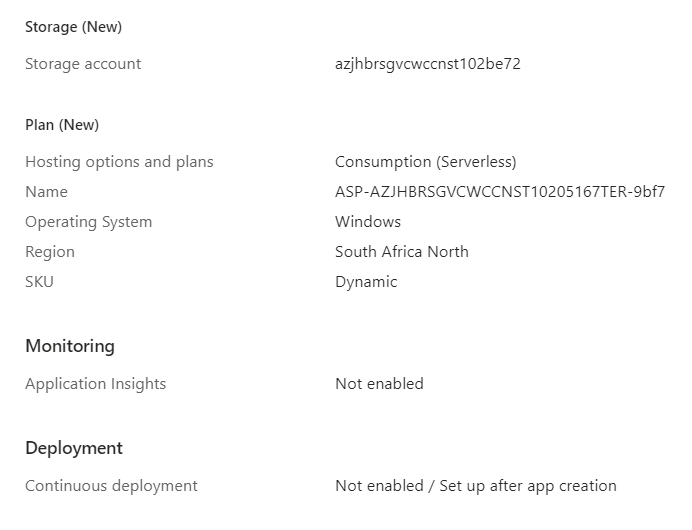


Make sure to select the .net version that matches your visual studio. Region select South Africa North. Go next when done

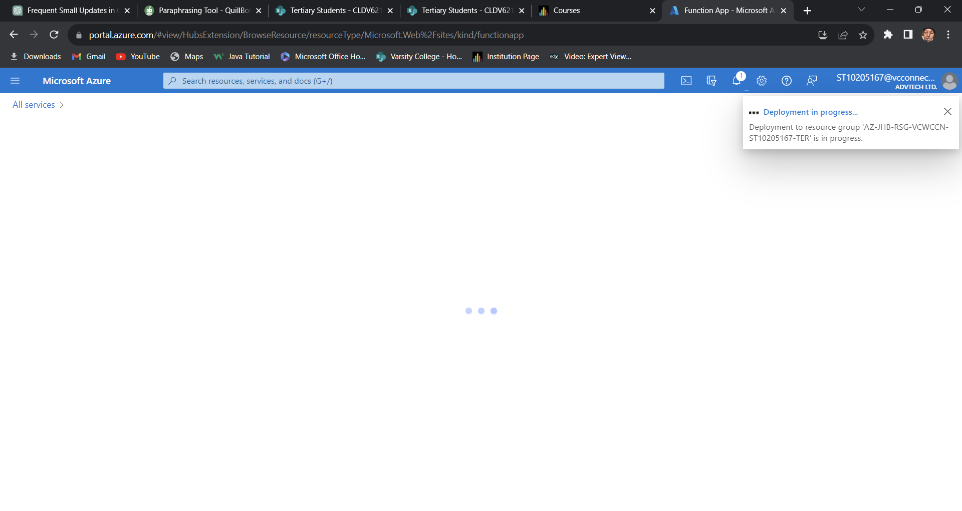


Select the new storage account.

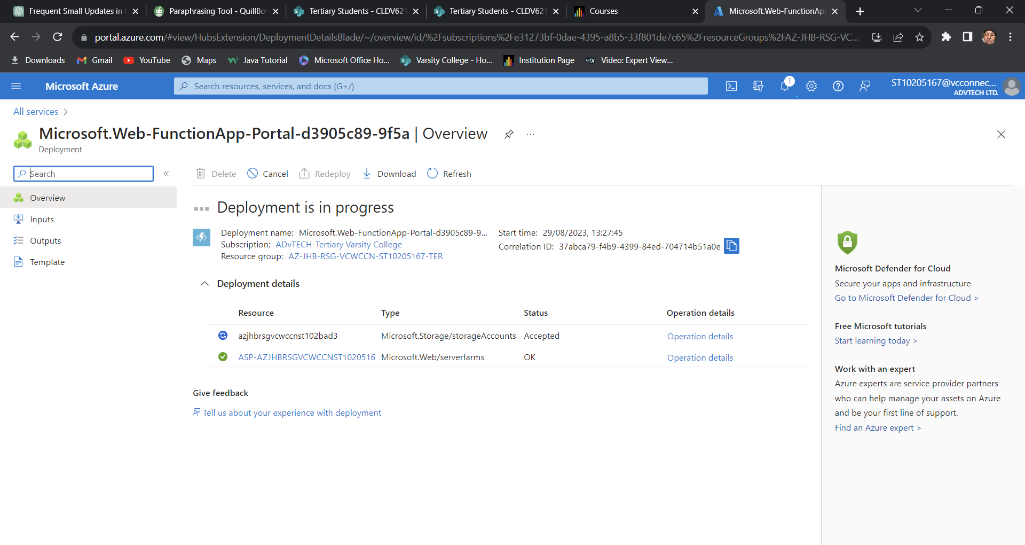




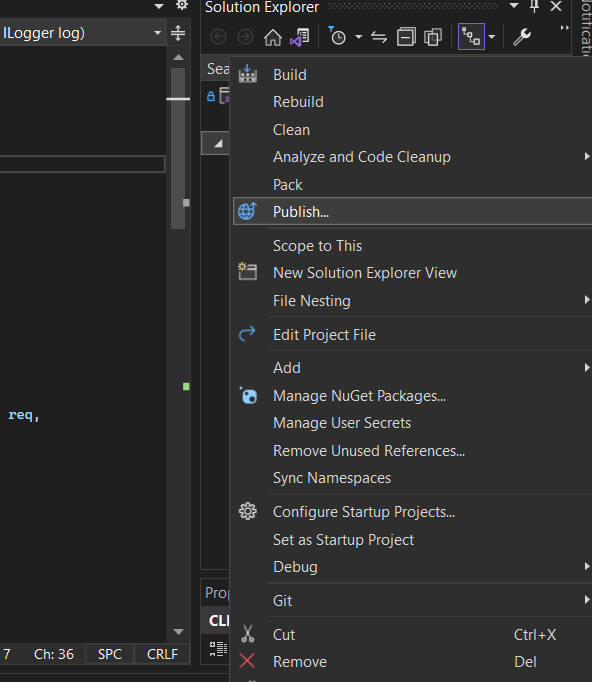
Review the information and go next.



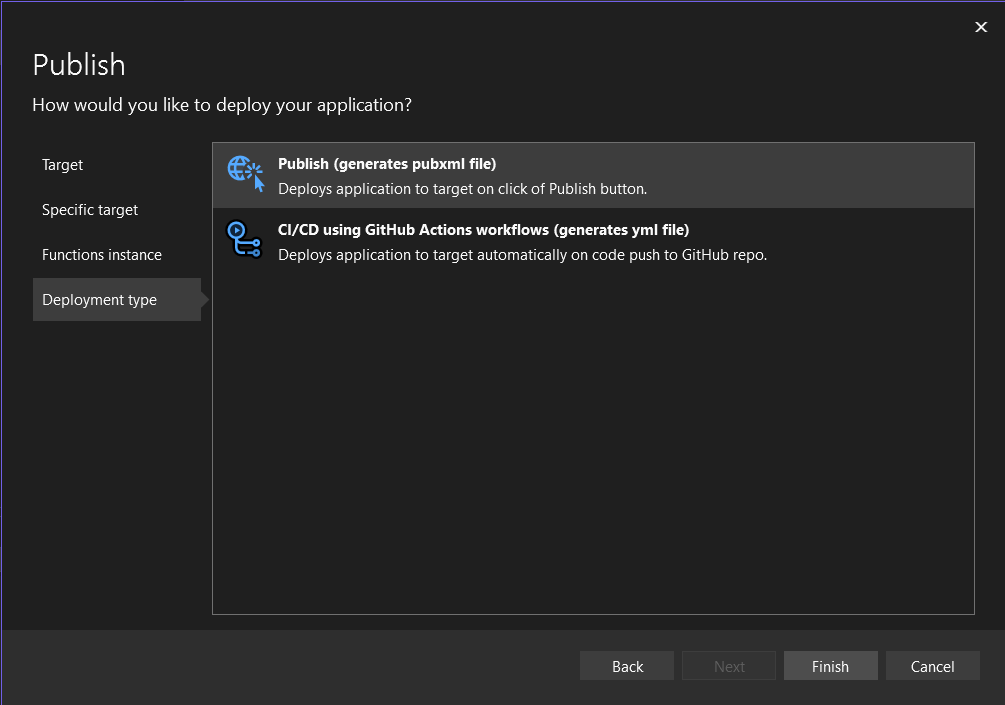
After reviewing start deploying the function and the following should pop up on your screen.



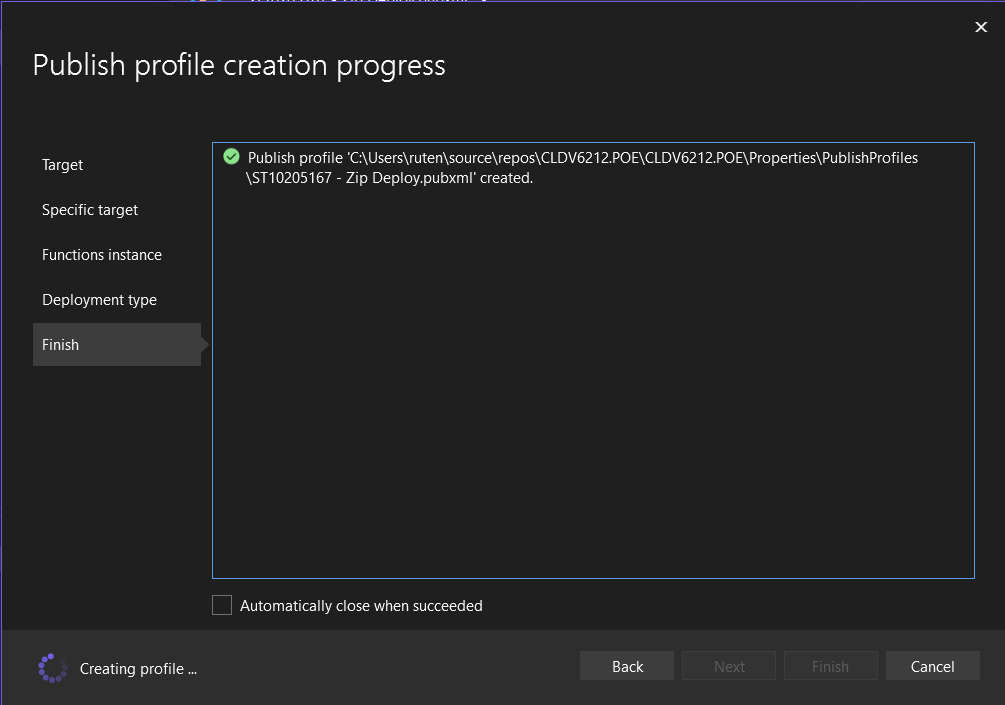
After deploying is successful open you function app on visual studio.

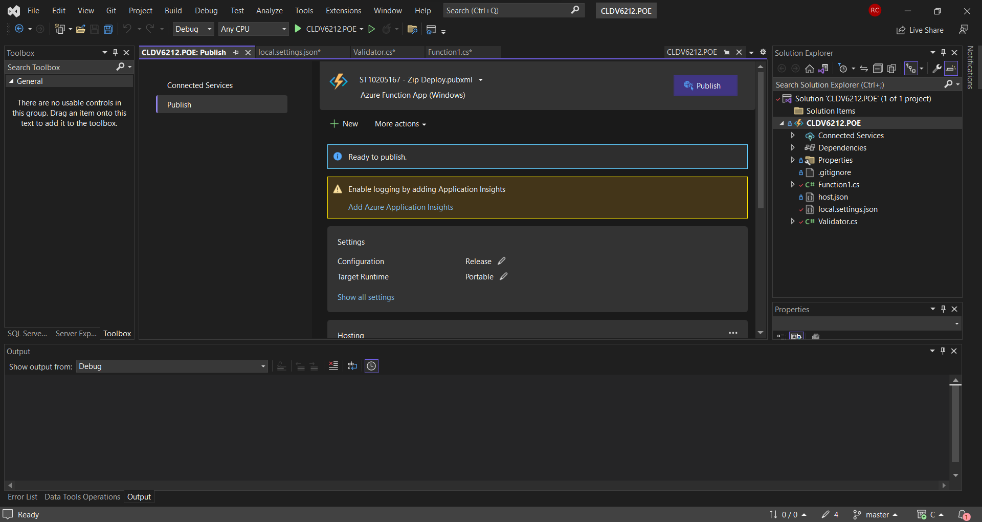


Go to your solutions explorer and right click on the function app and select publish.

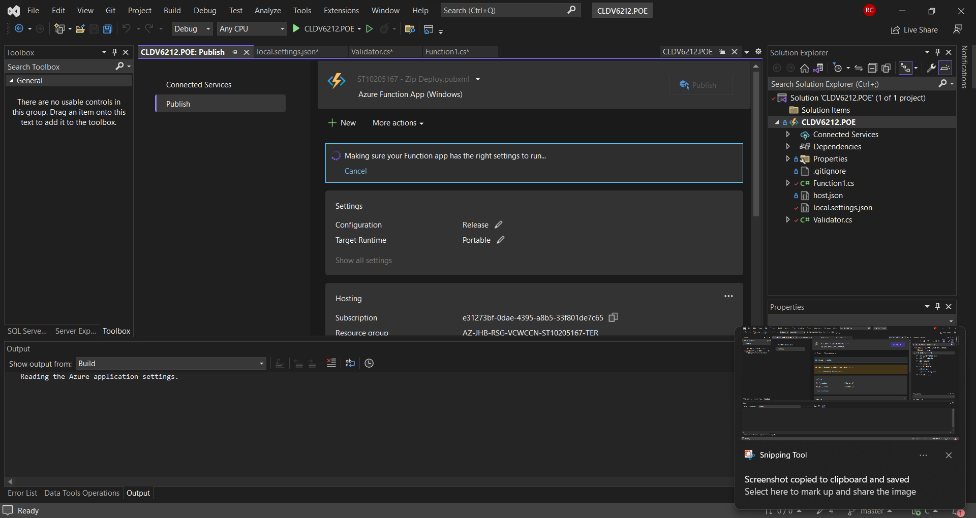


Select public publish as per above.

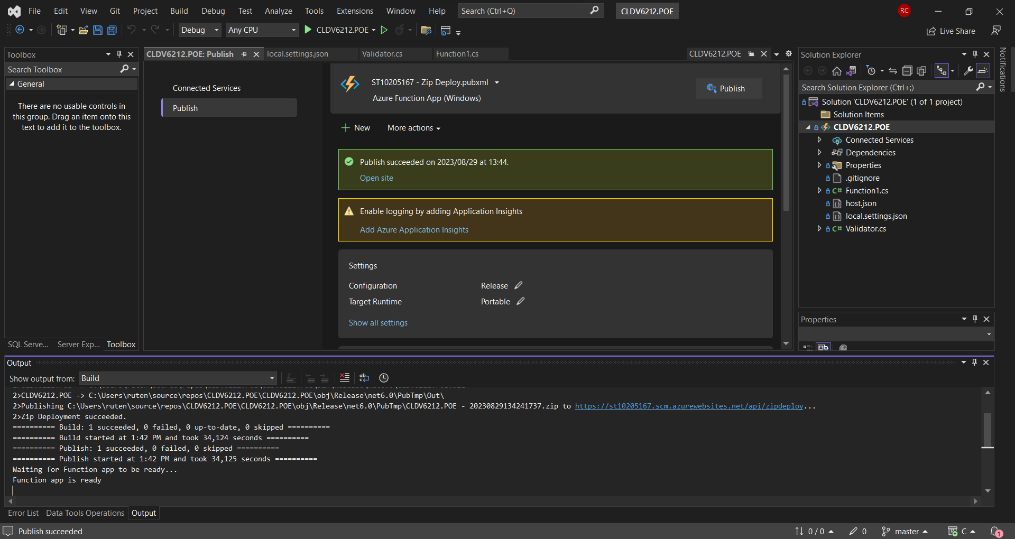




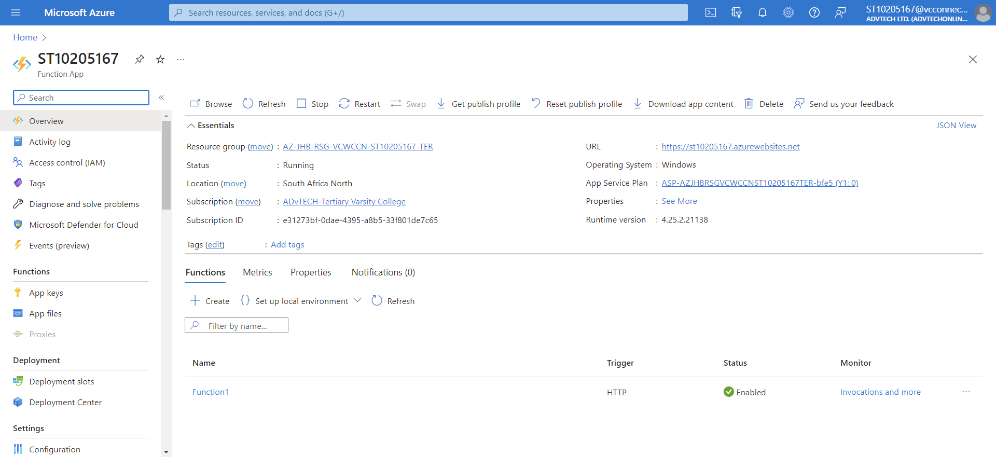
After selecting publish you should see the following UI/UX and select publish.



Publishing the application



You will get the following message when publishing is successful.



Upon completion of publishing your application logon to azure and select your Azure Fuction App that you are creating.

A screenshot of a computer

Description automatically generated

Open the app and click on get URL to use the app.

[https://st10205167.azurewebsites.net/api/id/{ID}](https://st10205167.azurewebsites.net/api/id/%7bID%7d)?

Link of my function above!

# References

Beschokov, M. (2023, August 26). *Cloud Elasticity*. Retrieved from wallarm.com: https://www.wallarm.com/what/cloud-elasticity

Bigelow, S. J. (2023, August 28). *immutable infrastructure*. Retrieved from techtarget.com: https://www.techtarget.com/searchitoperations/definition/immutable-infrastructure#:~:text=Immutable%20infrastructure%20is%20an%20approach,each%20time%20any%20change%20occurs.

Brunskill, V.-L. (2023, August 26). *Polyglot persistence*. Retrieved from techtarget.com: https://www.techtarget.com/searchapparchitecture/definition/polyglot-persistence

Chao, L. (2014). *Cloud Database Development and Management .* New York: Taylor & Francis Group.

Churchman, M. (2023, August 28). *Snowflake Configurations and DevOps Automation*. Retrieved from sumologic.com: https://www.sumologic.com/blog/snowflake-configurations-and-devops-automation/

D., A. (2023, August 26). *Process Synchronization in Operating Systems: Definition & Mechanisms*. Retrieved from study.com: https://study.com/academy/lesson/process-synchronization-in-operating-systems-definition-mechanisms.html

Dadgar, A. (2023, August 2023). *What is Mutable vs. Immutable Infrastructure?* Retrieved from hashicorp: https://www.hashicorp.com/resources/what-is-mutable-vs-immutable-infrastructure

Linthicum, D. (2023, August 24). *Multicloud architecture decomposition simplified*. Retrieved from infoworld.com: https://www.infoworld.com/article/3610423/multicloud-architecture-decomposition-simplified.html

Lutkevich, B. (2023, August 27). *Asynchronous*. Retrieved from techtarget.com: https://www.techtarget.com/searchnetworking/definition/asynchronous

Lutkevich, B., & Biscobing, J. (2023, August 26). *Defintion of a relational database* . Retrieved from techtarget.com: https://www.techtarget.com/searchdatamanagement/definition/relational-database

madhav\_mohan. (2023, August 26). *Scalability and Elasticity in Cloud Computing*. Retrieved from geeksforgeeks.org: https://www.geeksforgeeks.org/scalability-and-elasticity-in-cloud-computing/

morefield.com. (2023, August 28). *ON-PREMISES VS. CLOUD*. Retrieved from morefield.com: https://morefield.com/blog/on-premises-vs-cloud/

nextservicesoftware. (2023, August 27). *Mean Time Between Failures (MTBF): How to Calculate & Increase*. Retrieved from nextservicesoftware.com: https://nextservicesoftware.com/news/mean-time-between-failures-mtbf

Rahul Awati, I. W. (2023, August 24). *monolithic architecture*. Retrieved from techtarget.com: https://www.techtarget.com/whatis/definition/monolithic-architecture

splunk. (2023, August 2023). *What Is Mean Time to Repair (MTTR)?* Retrieved from splunk.com: https://www.splunk.com/en\_us/data-insider/what-is-mean-time-to-repair.html

talend. (2023, August 24). *Monolithic vs. Microservices: a guide to application architecture* . Retrieved from talend.com: https://www.talend.com/resources/monolithic-architecture/

Yadav, G. (2023, August 28). *What are the snowflake servers?* Retrieved from learnsteps.com: https://www.learnsteps.com/what-are-snowflake-servers/