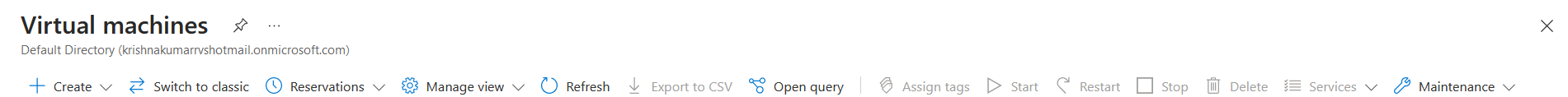
Scenario Implementation based on real time case studies – Assignment

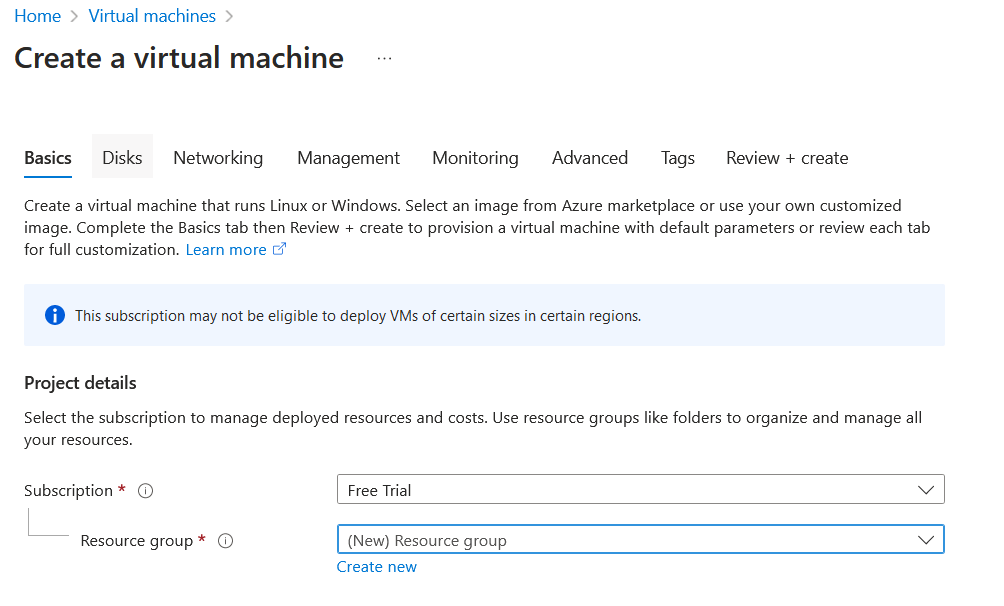
**Mallwart Online Store Azure Infrastructure Enhancement**

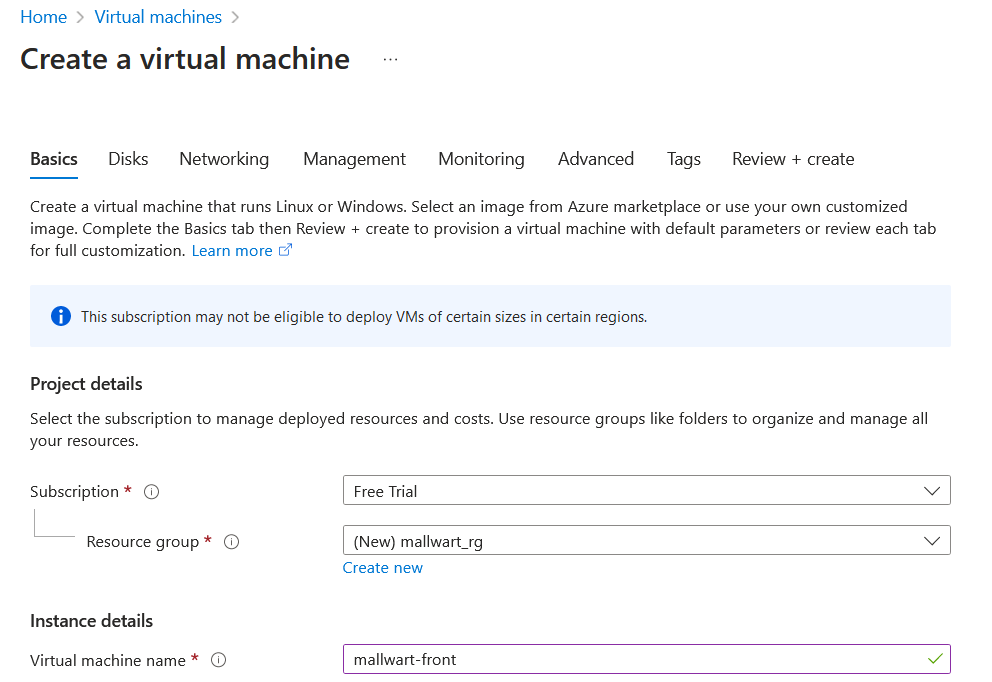
1. **Frontend (Presentation Layer):**
   1. **Create a Virtual Machine for Frontend:**

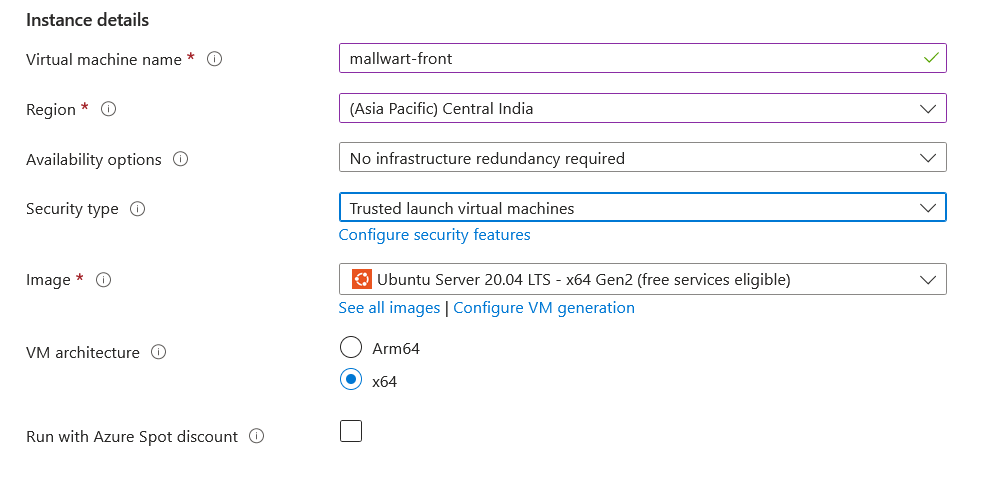
* In the Azure Portal, navigate to "Virtual machines" and click on "Add" to create a new virtual machine.

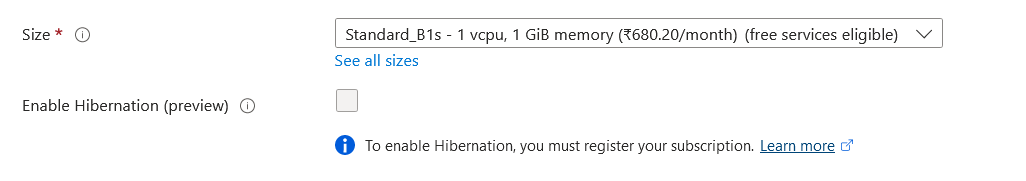


* Choose a subscription, resource group, and region for your VM.

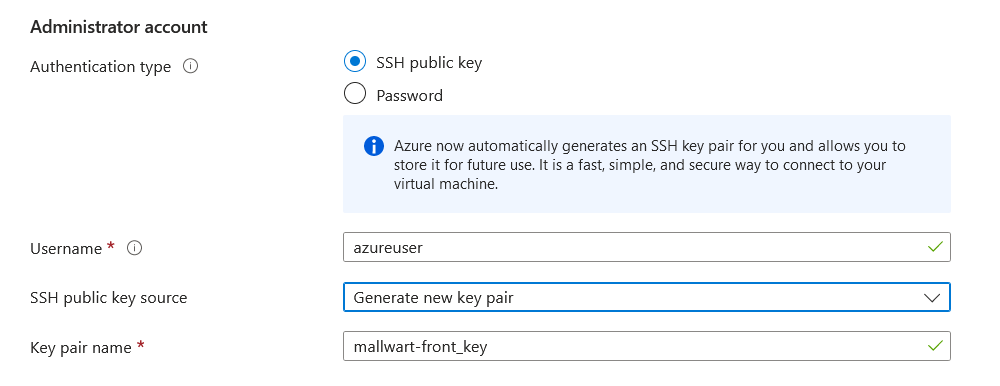
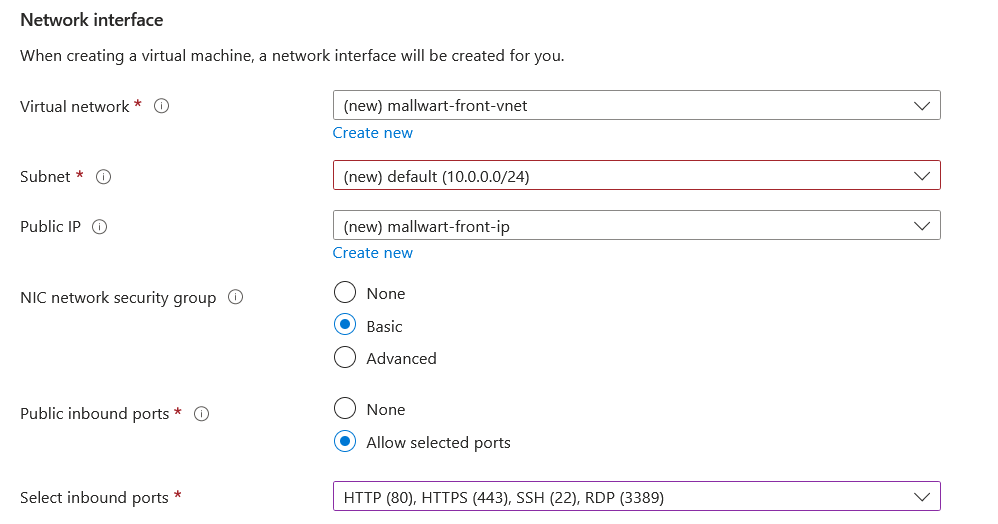




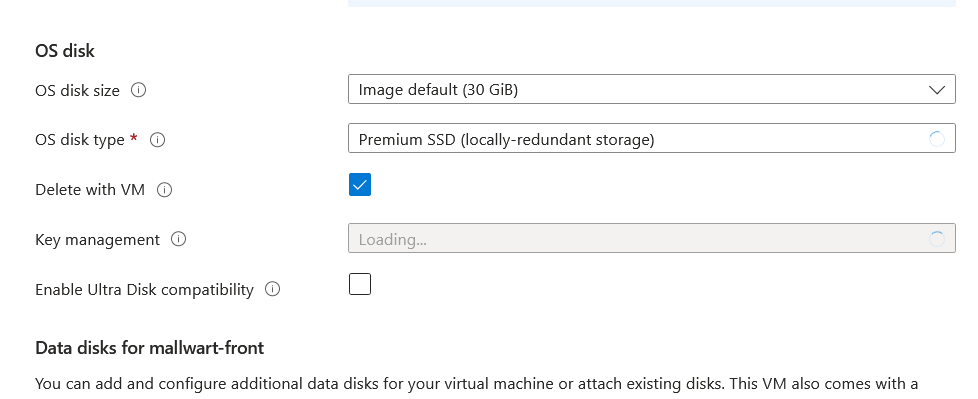
* Enter a name for your VM and select "Ubuntu Server" as the image. 
* Choose a VM size based on your requirements. For a basic frontend, a smaller size may suffice.



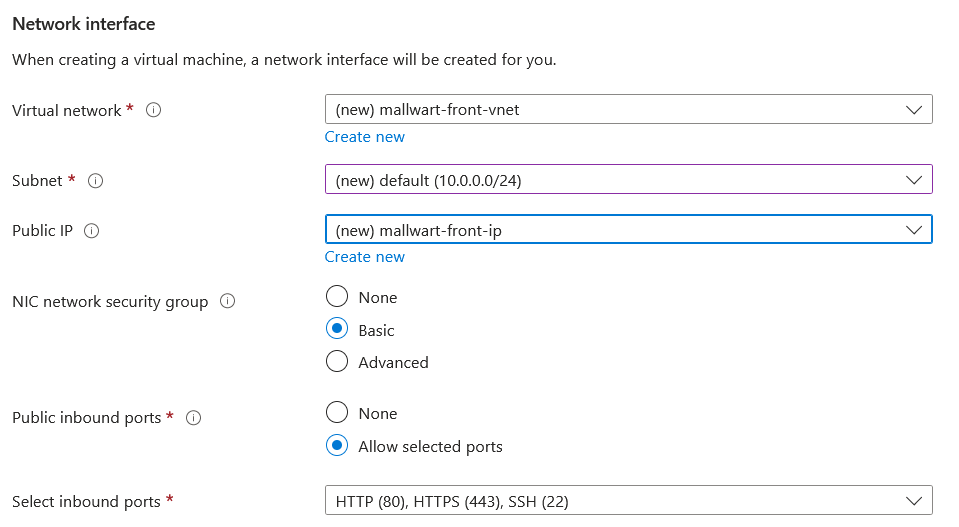
* Set authentication type to SSH public key or password. If using SSH, provide the public key or set a username/password.

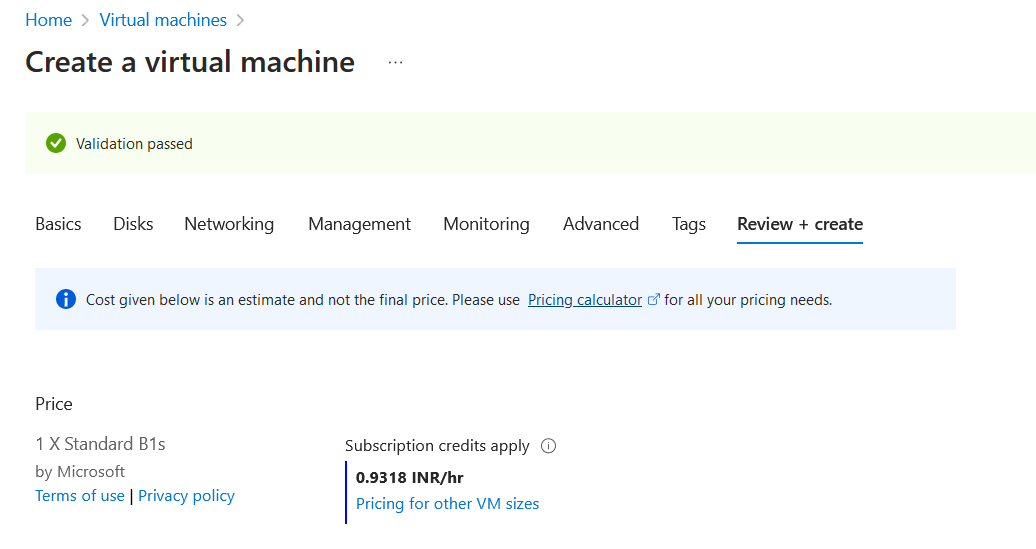
* Click "Next: Disks >" to configure disks, or leave default settings.



* Click "Next: Networking >" to configure networking. Open port 80 (HTTP) and optionally port 443 (HTTPS) for web traffic.

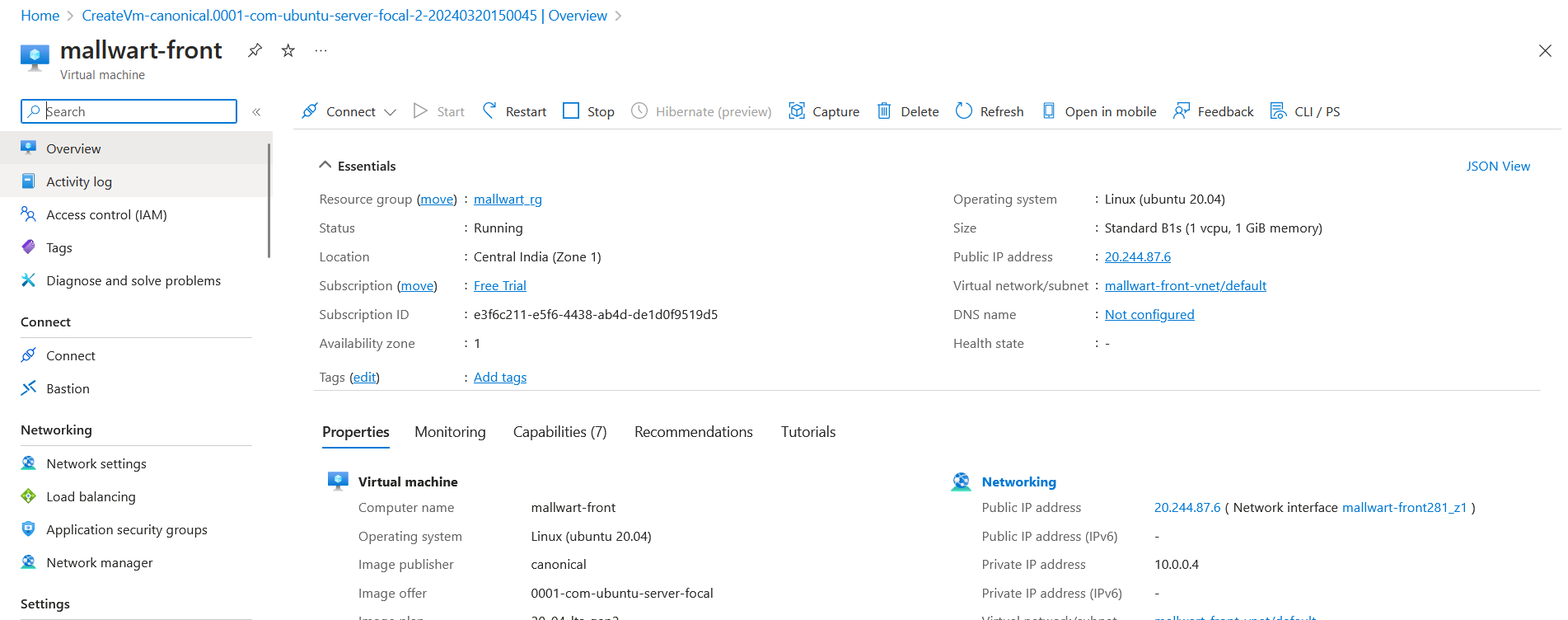


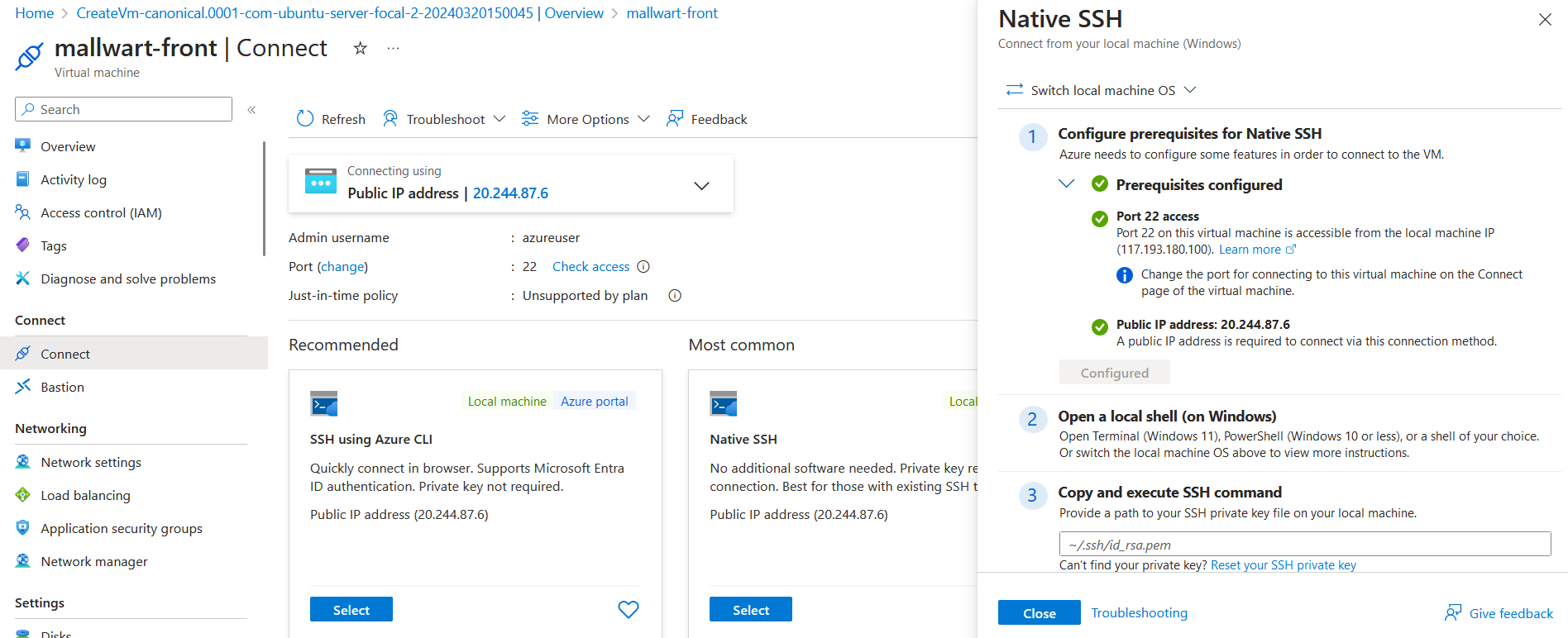
* Click "Next: Management >" to configure management settings, or leave default settings.
* Click "Next: Advanced >" to configure advanced settings, or leave default settings.
* Review the summary and click "Review + create".
* Once validation passes, click "Create" to deploy the VM.

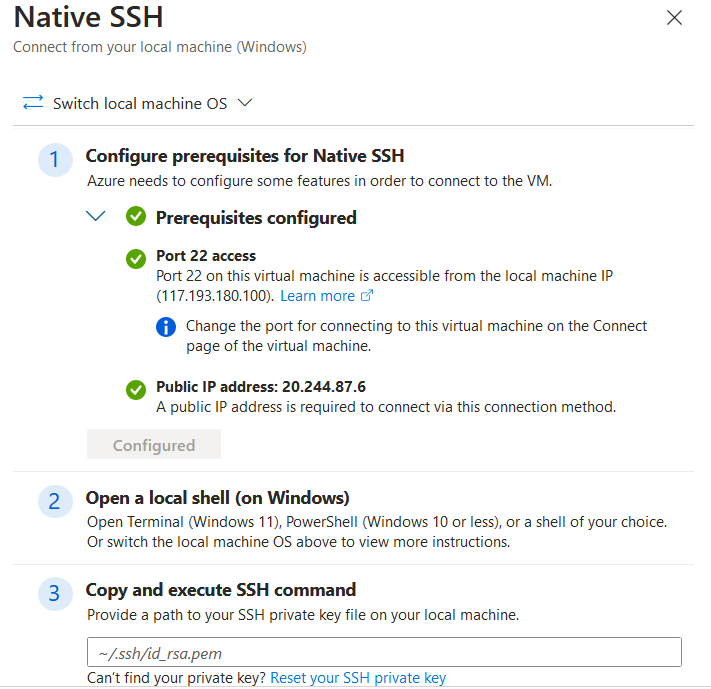


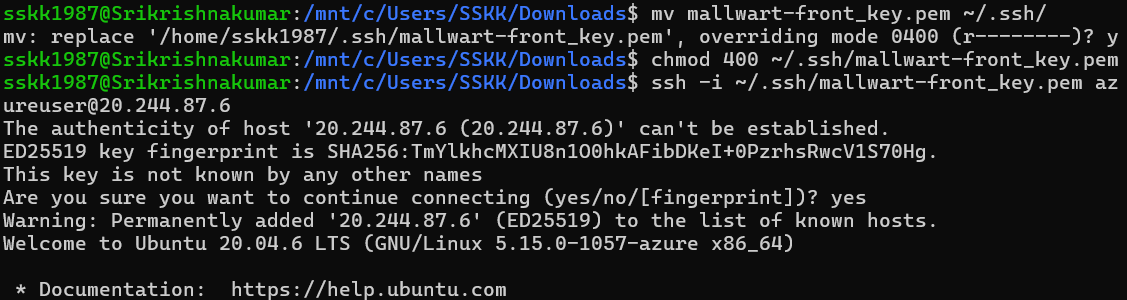
 **Connect to the Virtual Machine:**

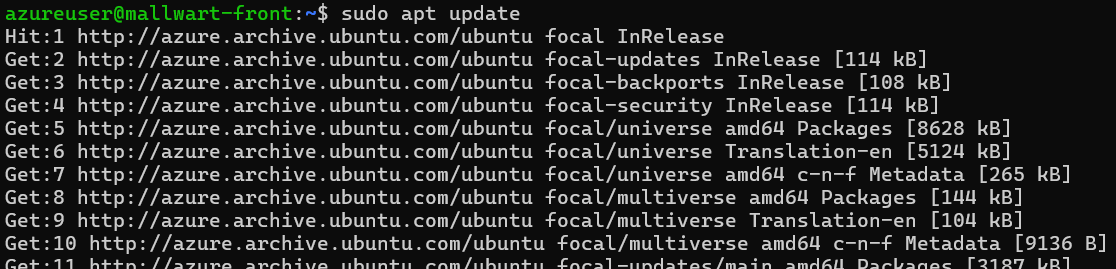
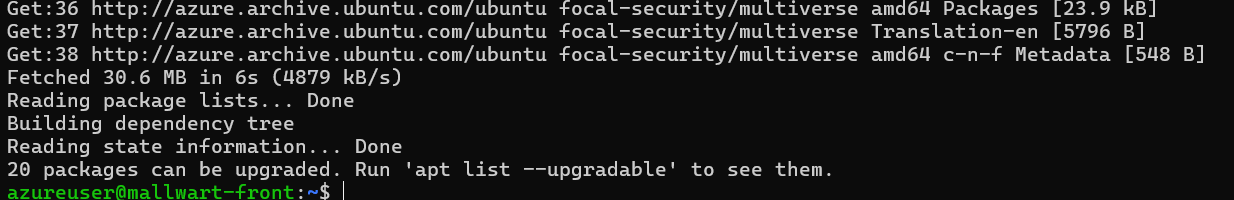
* Once the VM is deployed, go to "Virtual machines" in the Azure Portal and select your VM.

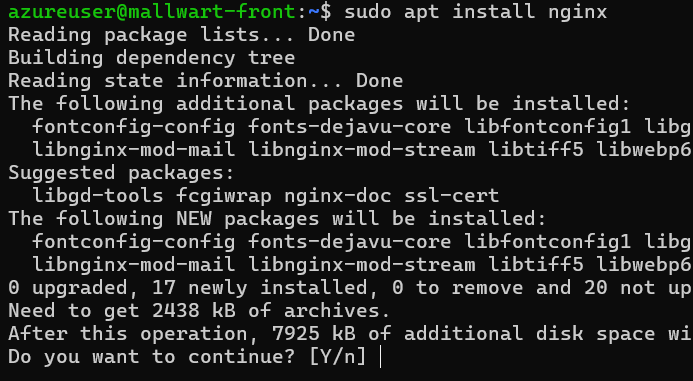


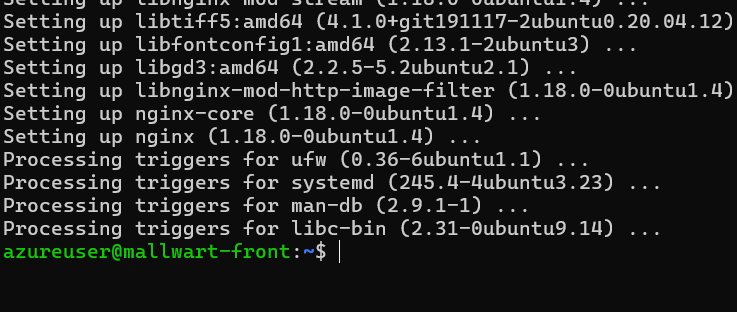
* Under the "Connect" tab, you'll find instructions for connecting to the VM via SSH. 



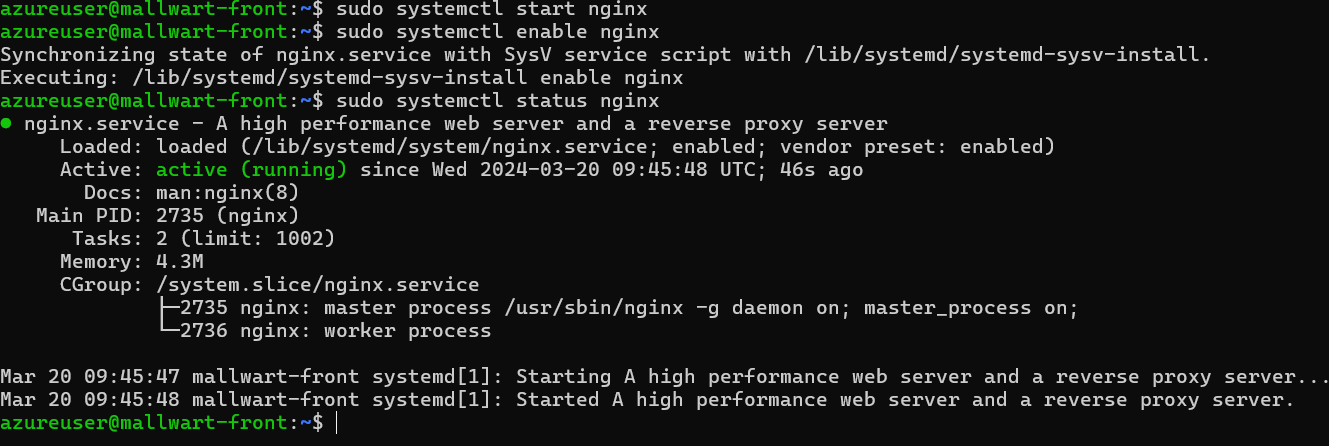
 

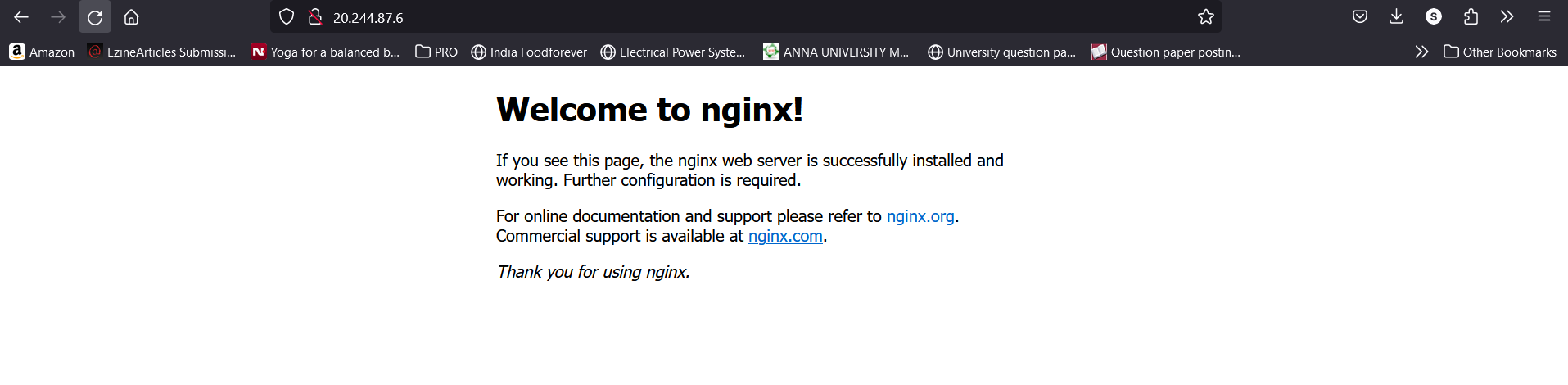
* 1. **Install Web Server Software (Nginx or Apache):**
* Once connected to the VM via SSH, update the package list:  
* Install Nginx: sudo apt install nginx.





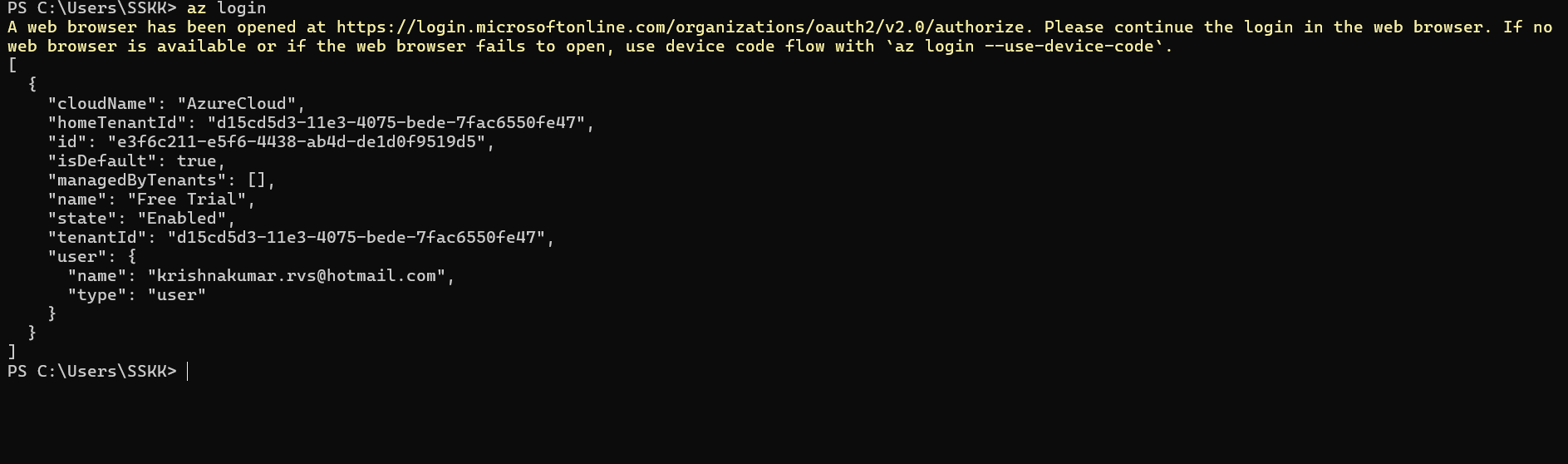
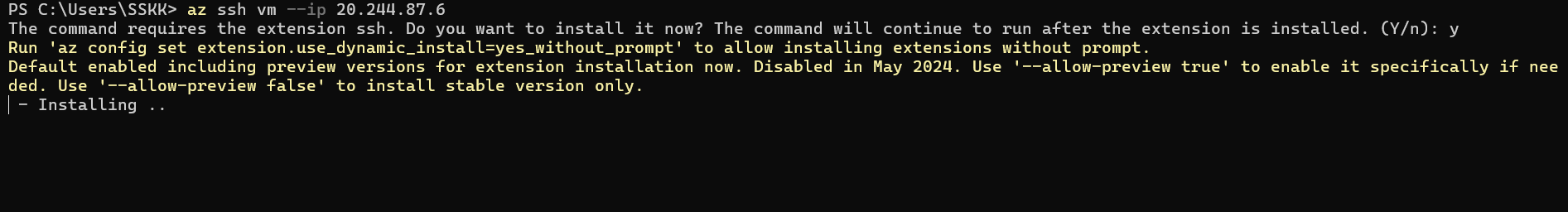
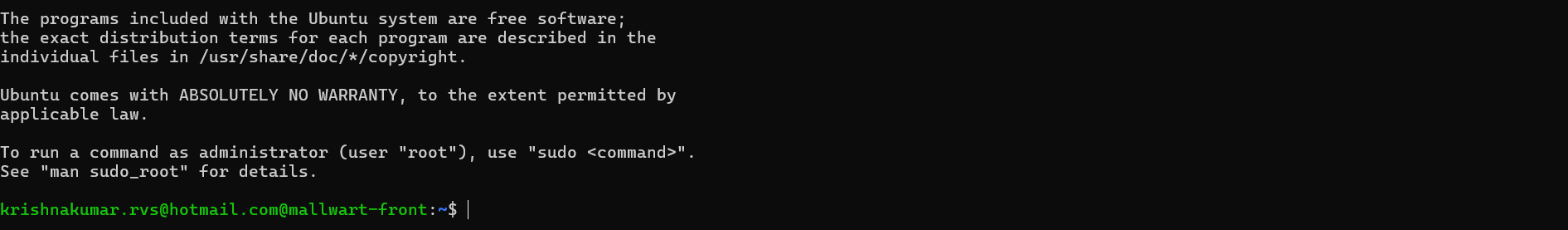
* Start the Nginx service: sudo systemctl start nginx.
* Enable Nginx to start on boot: sudo systemctl enable nginx.
* Verify that Nginx is running: sudo systemctl status nginx.



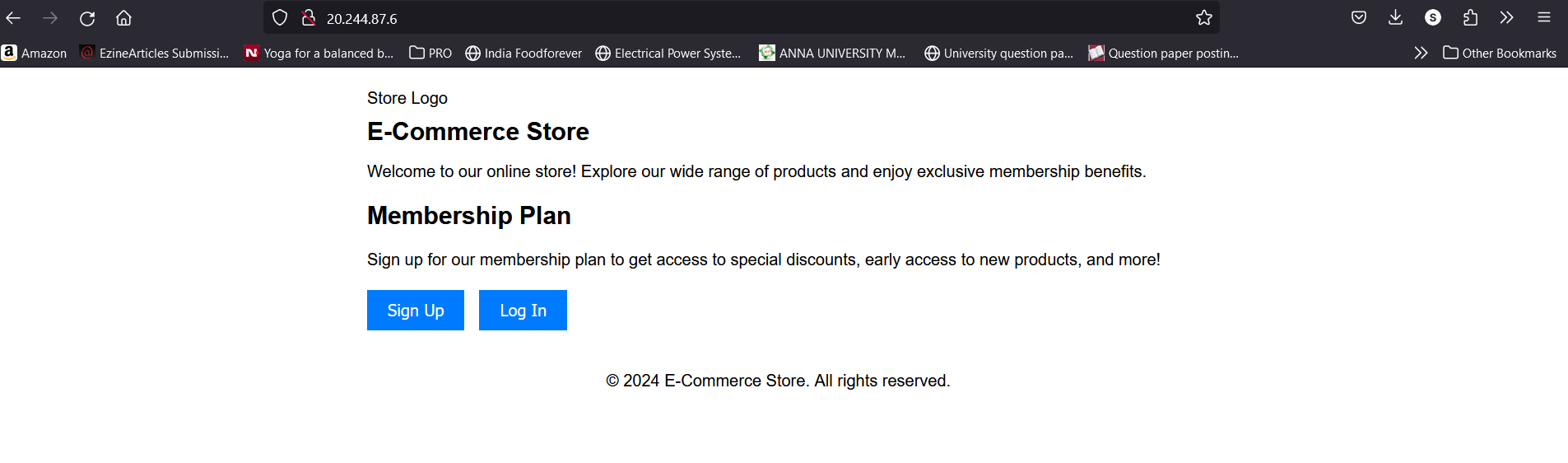


1.3 **Upload Frontend Files:**

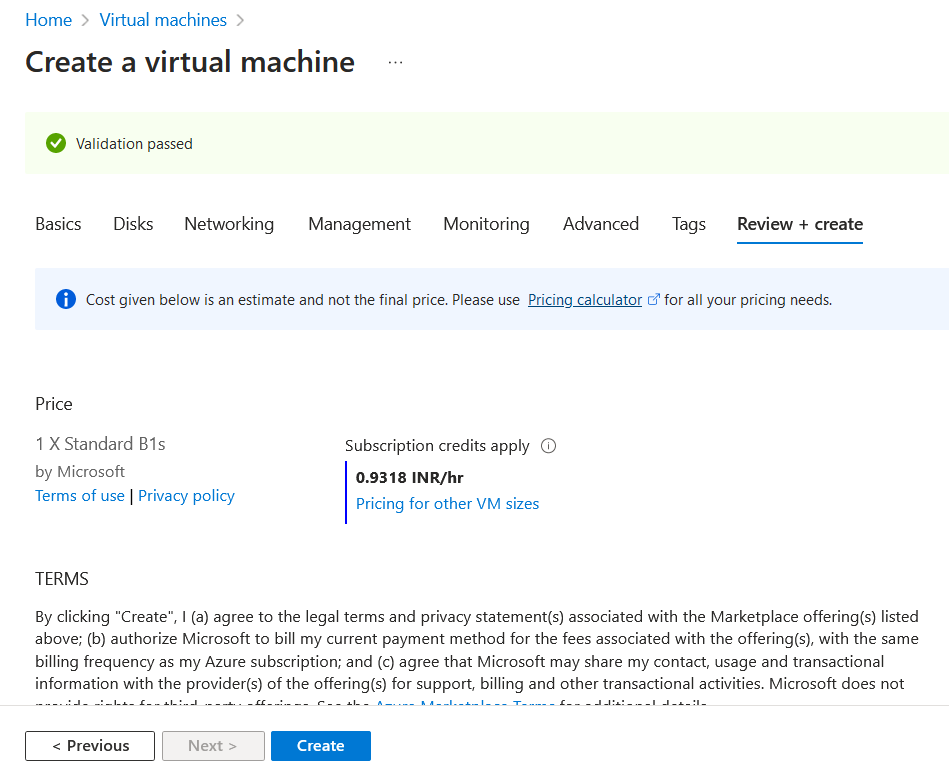
* Create a directory to store your frontend files, such as /var/www/html: sudo mkdir -p /var/www/html.

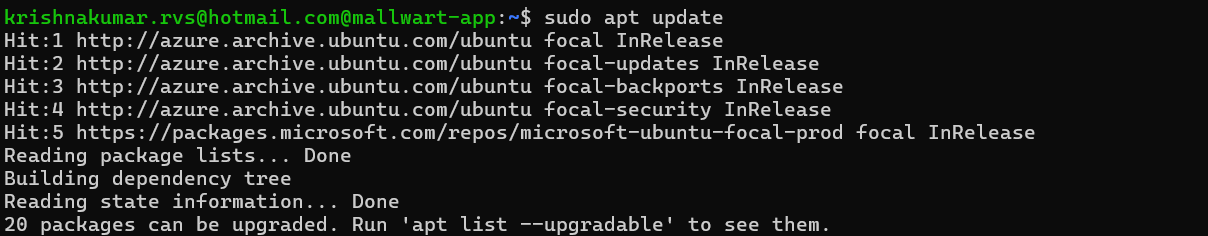
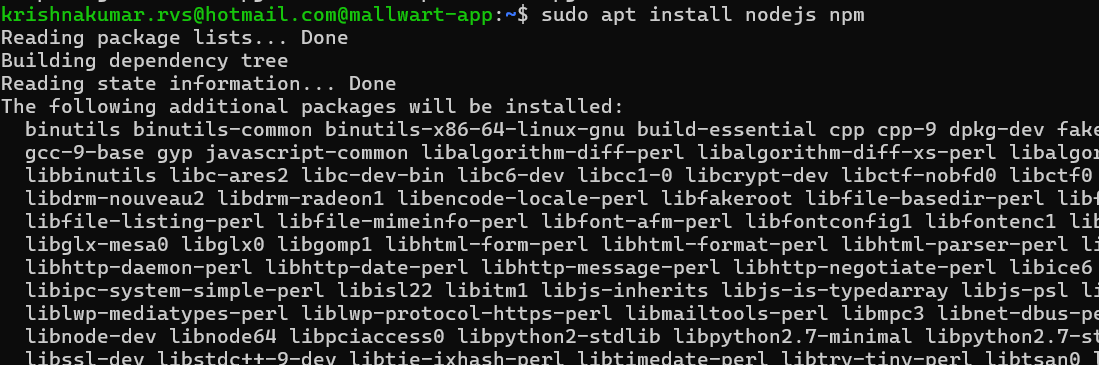
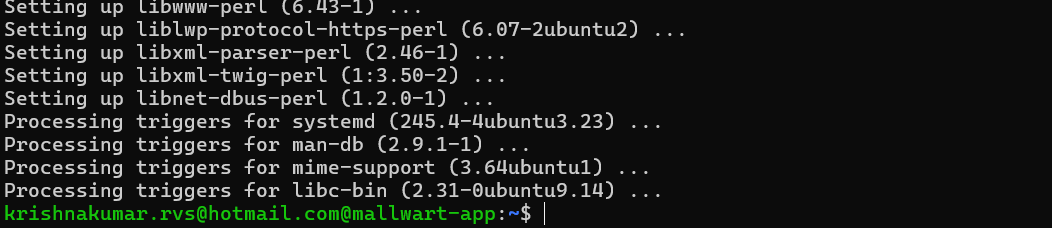
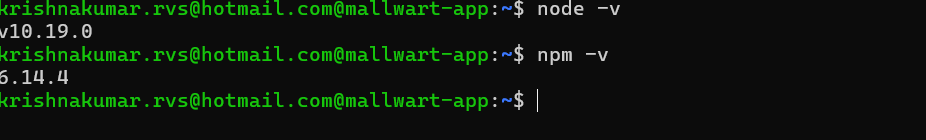
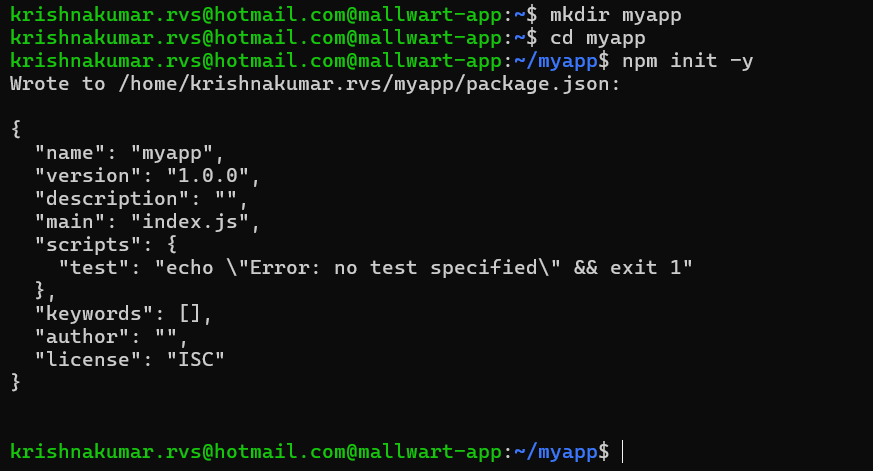
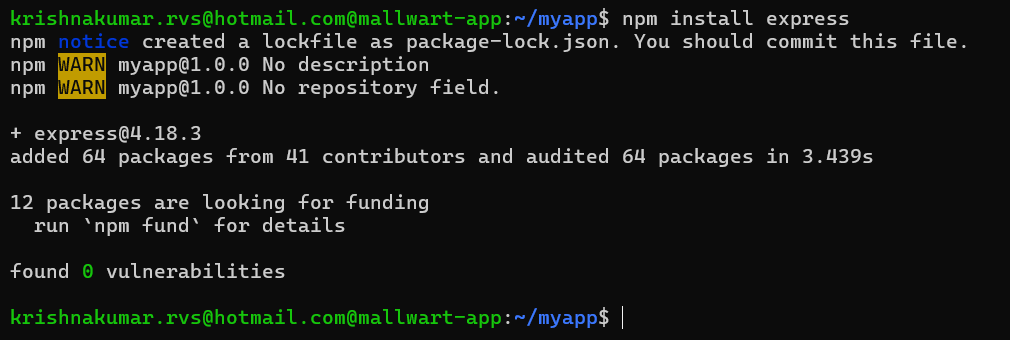
  

* Upload HTML, CSS, and JavaScript files to the /var/www/html directory using any other file transfer method.
* Ensure that your index.html (or main HTML file) is located in the root directory (/var/www/html) so that it can be accessed directly.



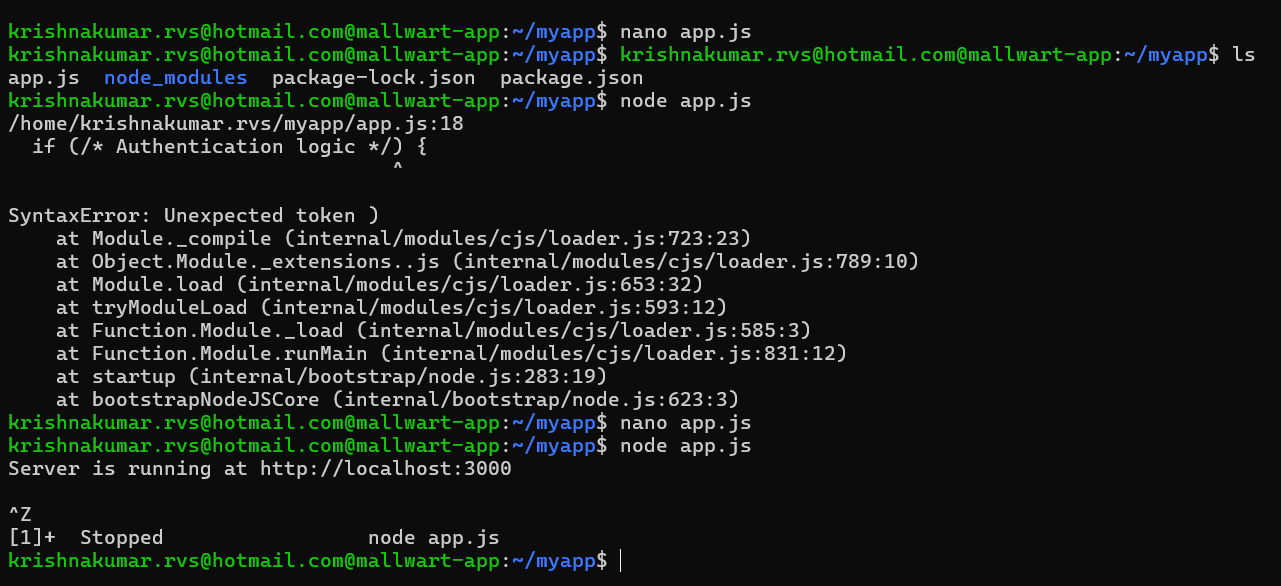
1. **Application Tier (Backend with Authorization):**
   * **Create a Virtual Machine for Application Tier**
   * Create another VM instance for hosting the backend application.
   * Choose appropriate VM size and configuration based on your application's requirements.
   * Select Ubuntu Server as the operating system for consistency with the frontend VM.



* + **Install Backend Framework and Dependencies:**
  + **Install Node.js and npm:**
* SSH into your Ubuntu VM.
* Update the package index: sudo apt update. 
* Install Node.js and npm: sudo apt install nodejs npm.  
* Verify the installation: node -v and npm -v. 
* **Initialize Your Node.js Application:**
* Create a directory for your application: mkdir myapp.
* Navigate to the directory: cd myapp.
* Initialize a new Node.js application: npm init -y. This will create a package.json file. 
* **Install Express.js:**
* Install Express.js as a dependency: npm install express.
* 

**2.3** **Create the Backend Application:**

* Create a new file for your backend application, e.g., app.js.



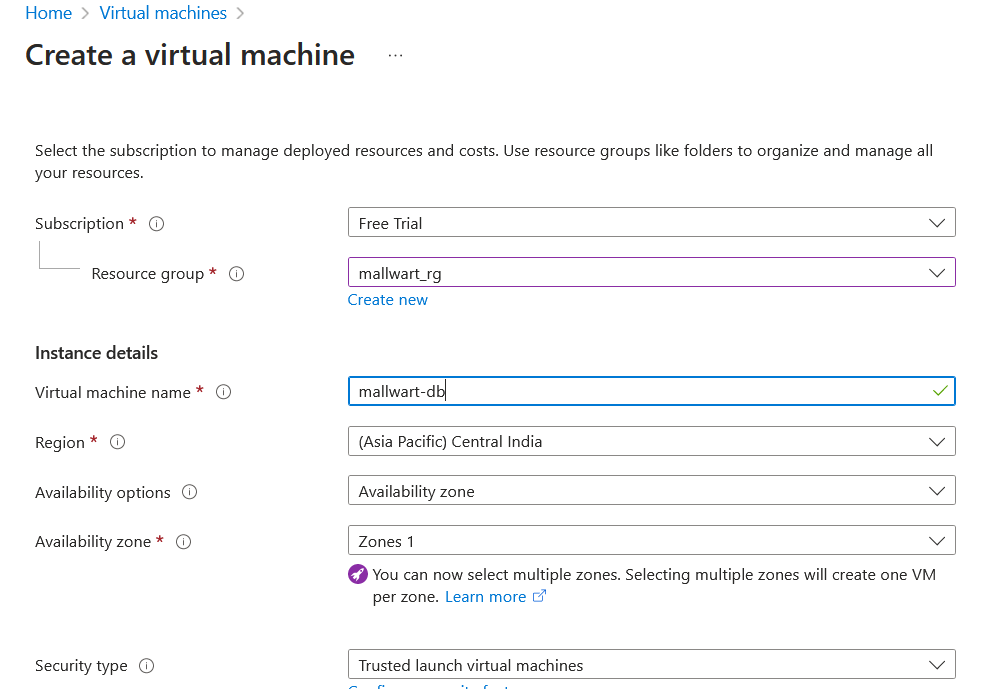
* Implement routes for user authentication and signup using Express.js.

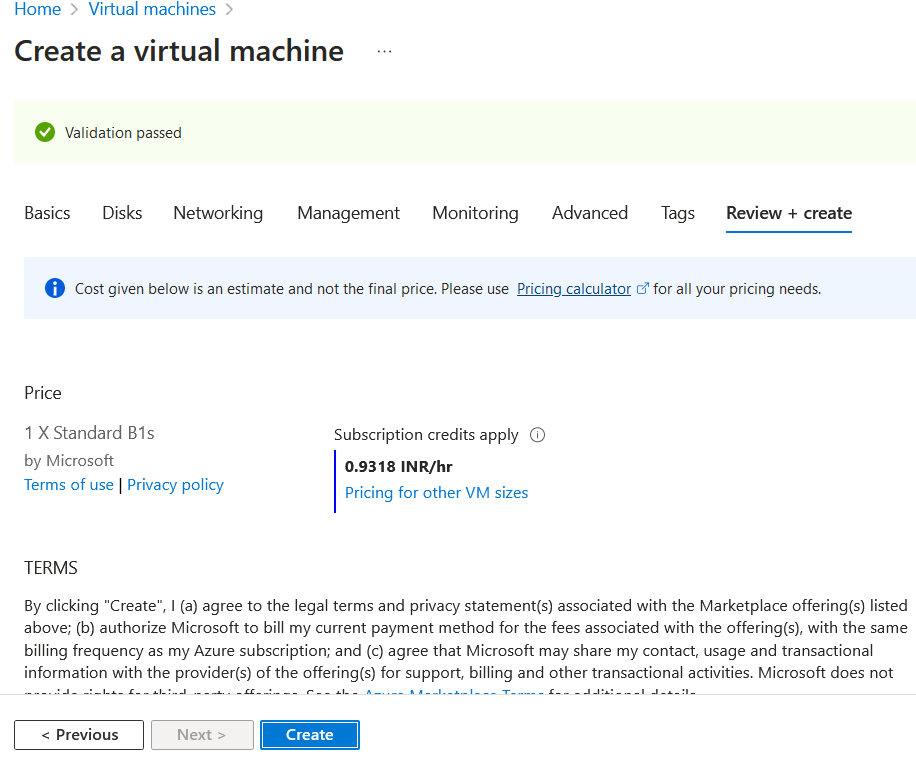
1. **Data tier**

**3.1 Provision MySQL VM**

**Create a New Virtual Machine:**

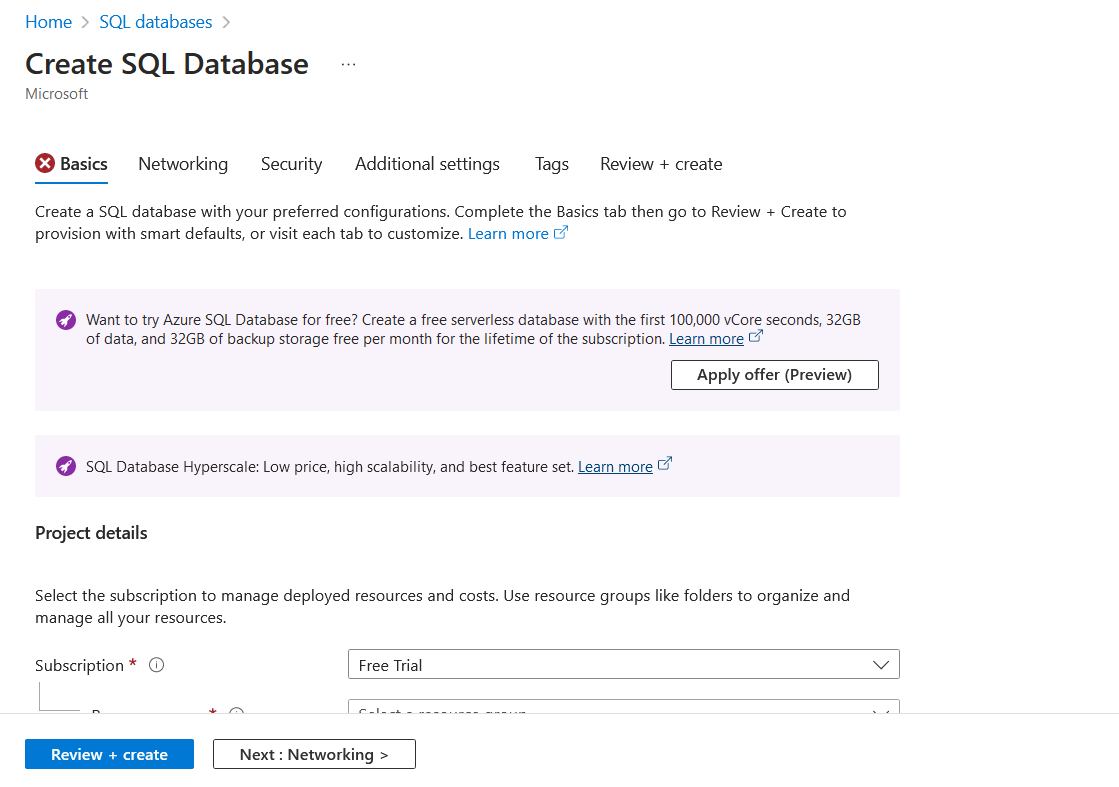
* 1. Provision a new virtual machine in your Azure environment
  2. Choose an appropriate operating system (e.g., Ubuntu, CentOS) for hosting MySQL.
  3. Follow the provider's instructions to set up the VM, including selecting hardware specifications, networking configuration, and SSH access.

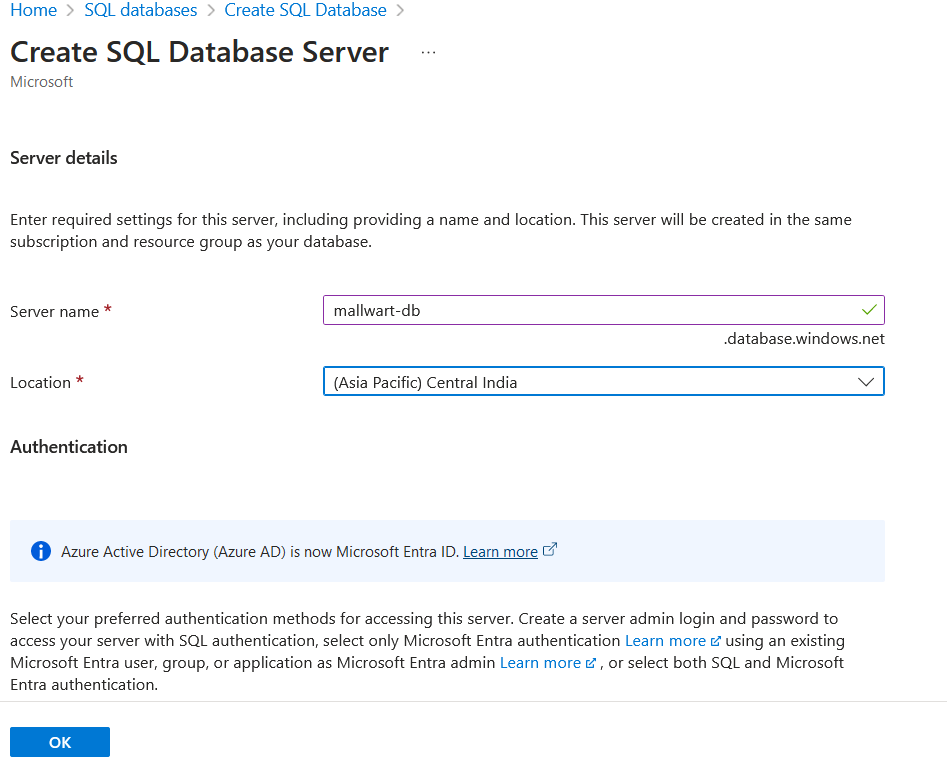


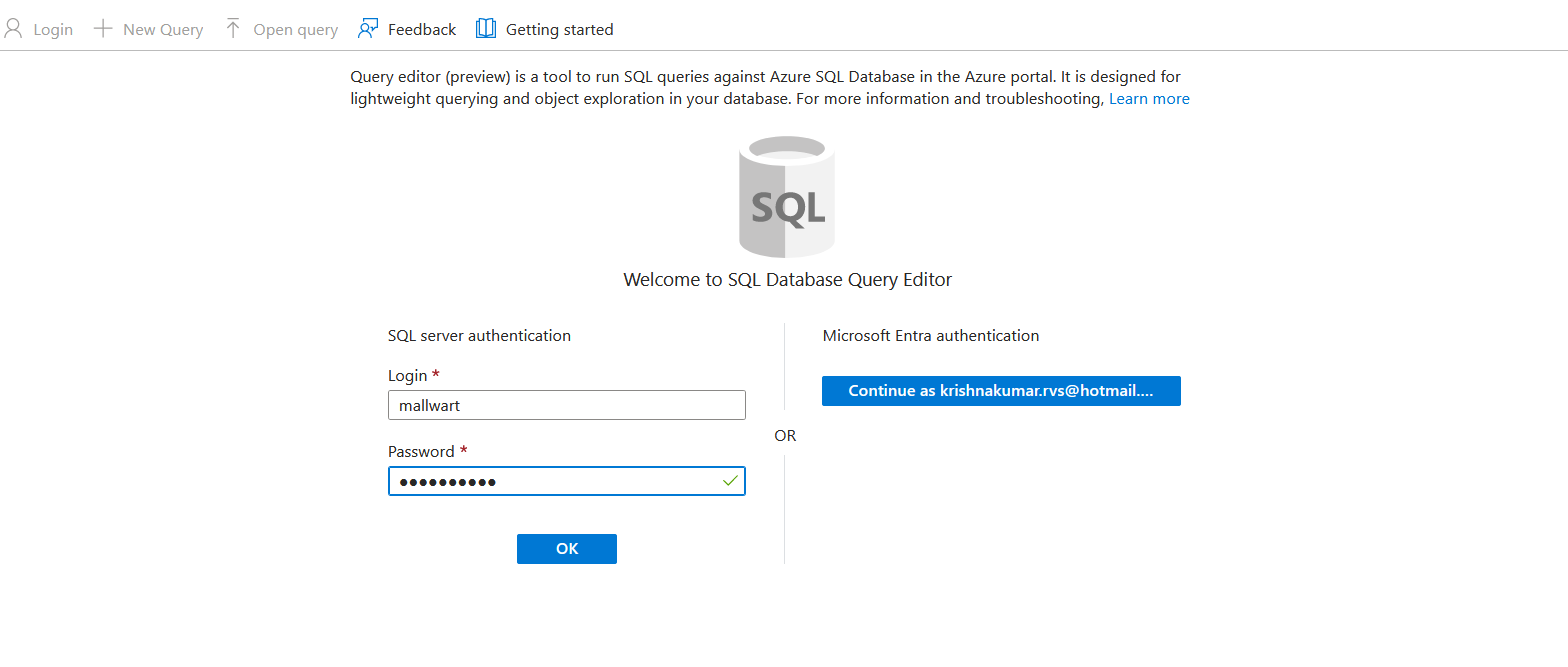
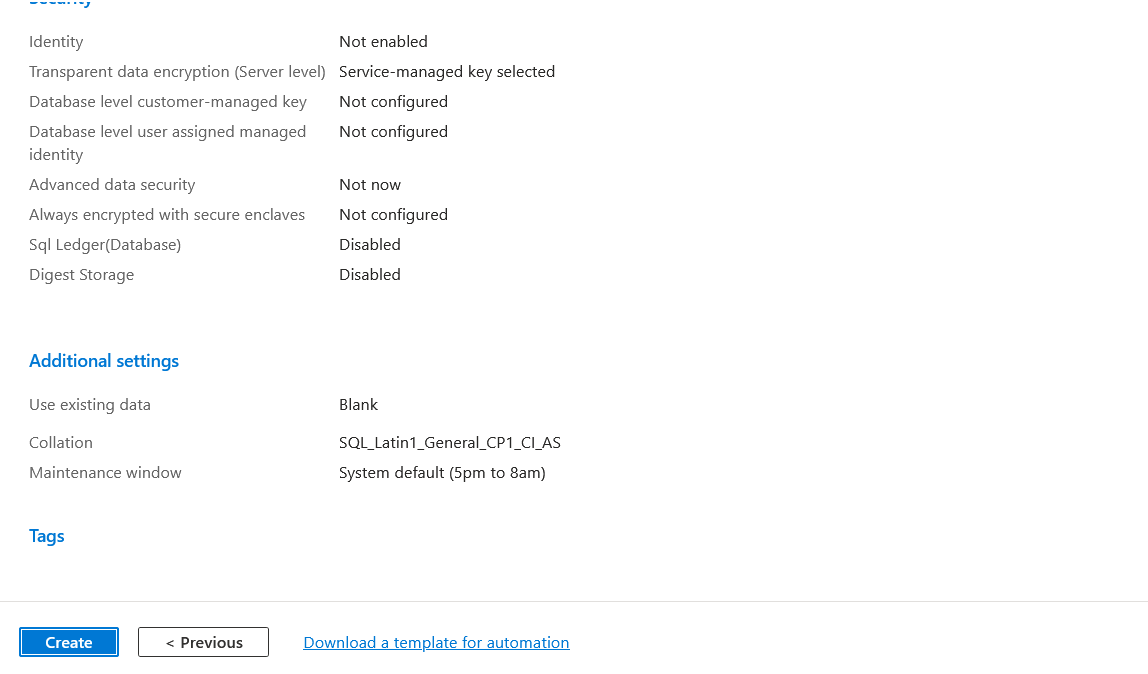
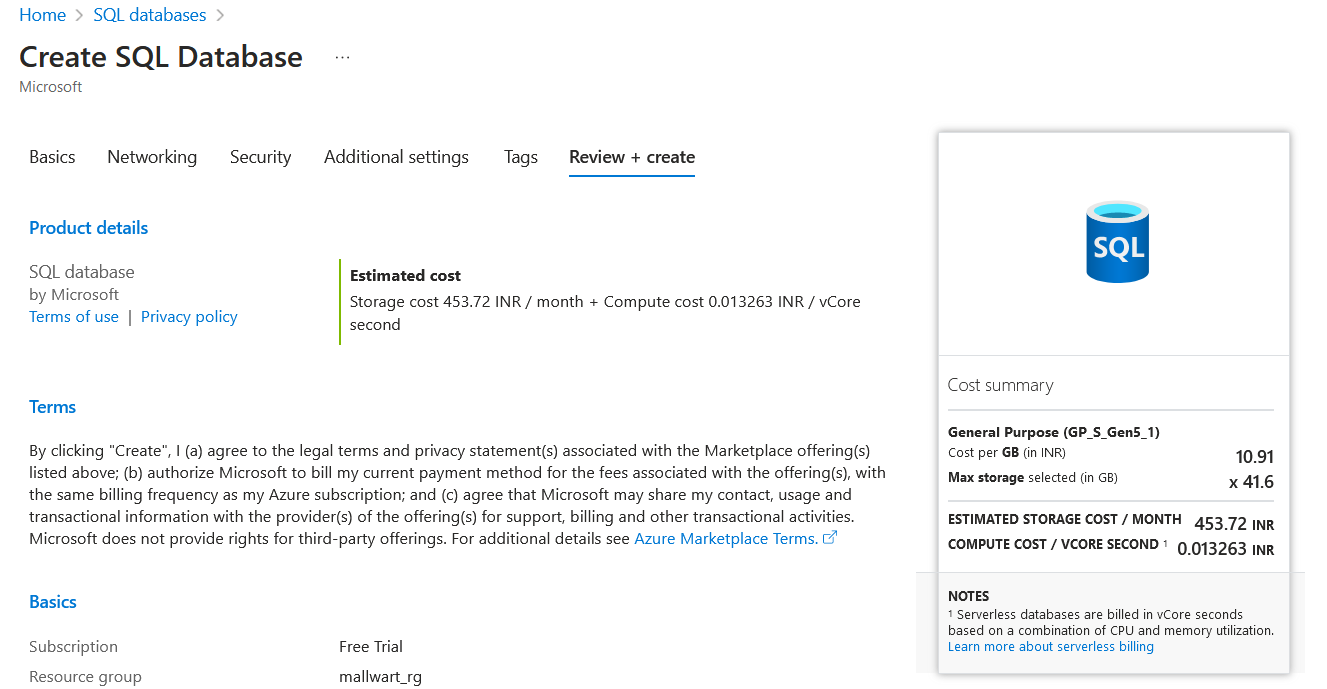


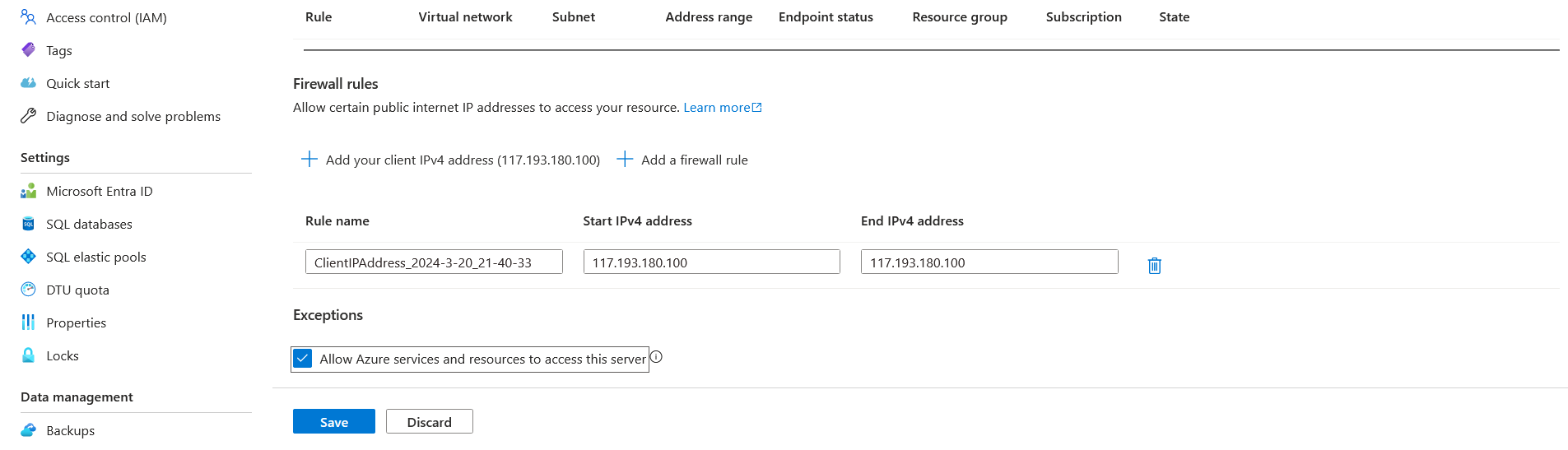
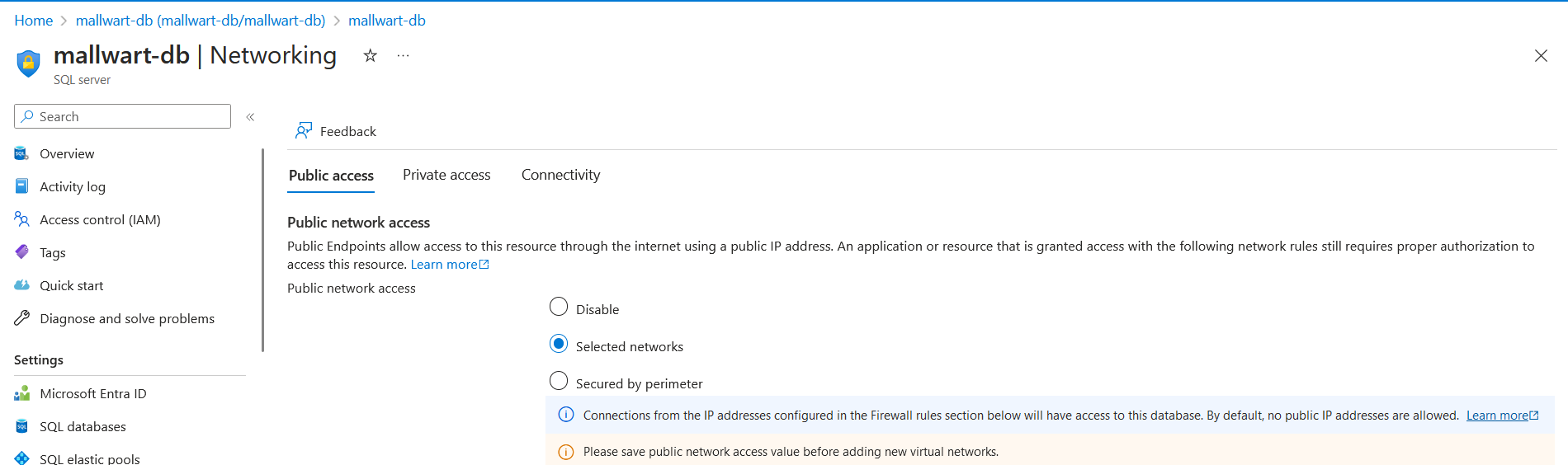
**3.2 Install MySQL Server**

* **Install MySQL Server:**
  + Once the VM is set up and accessible, install MySQL Server on the VM.
  + Use the package manager specific to your operating system to install MySQL. For example, on Ubuntu, you can use apt:







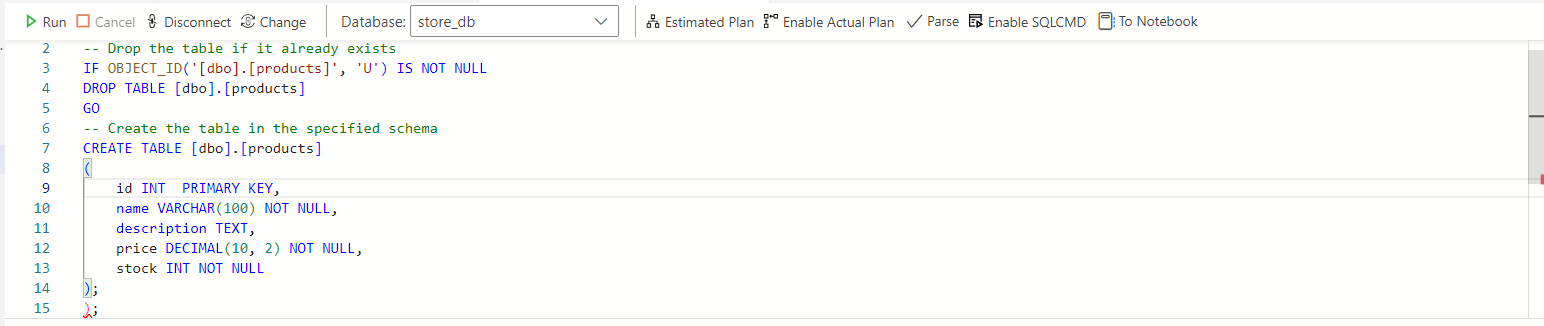
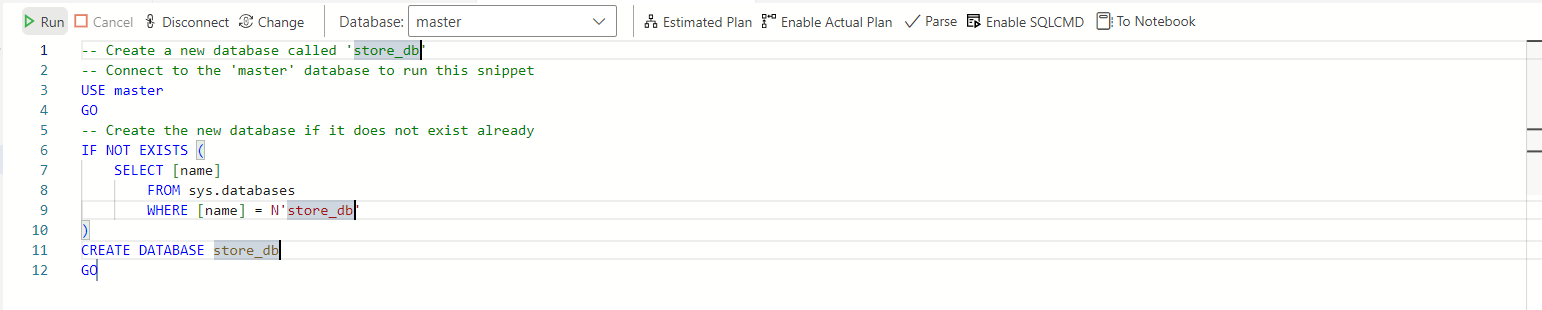


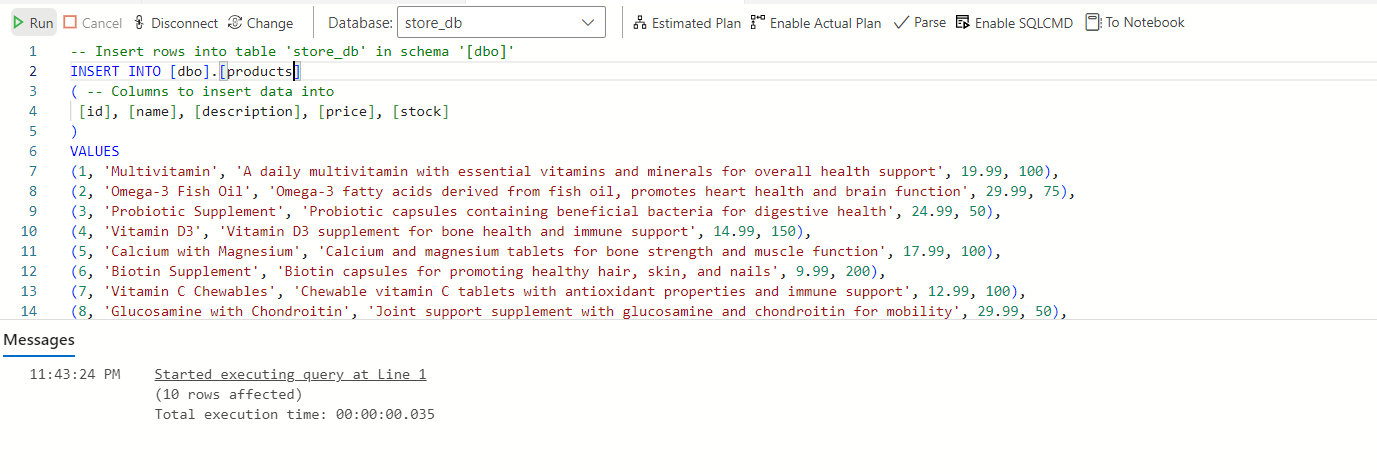
* **Configure MySQL**
  + Configure MySQL settings such as root password, database charset, and other options as per your requirements.
  + Edit the MySQL configuration file (my.cnf) to adjust settings if necessary.

**Step 4: Create Database and Tables**

1. **Create Database and Tables:**
   * Connect to the MySQL server and create the database and tables as described in the previous instructions.

Database created in the name of store\_db and a table is inserted in the name of products

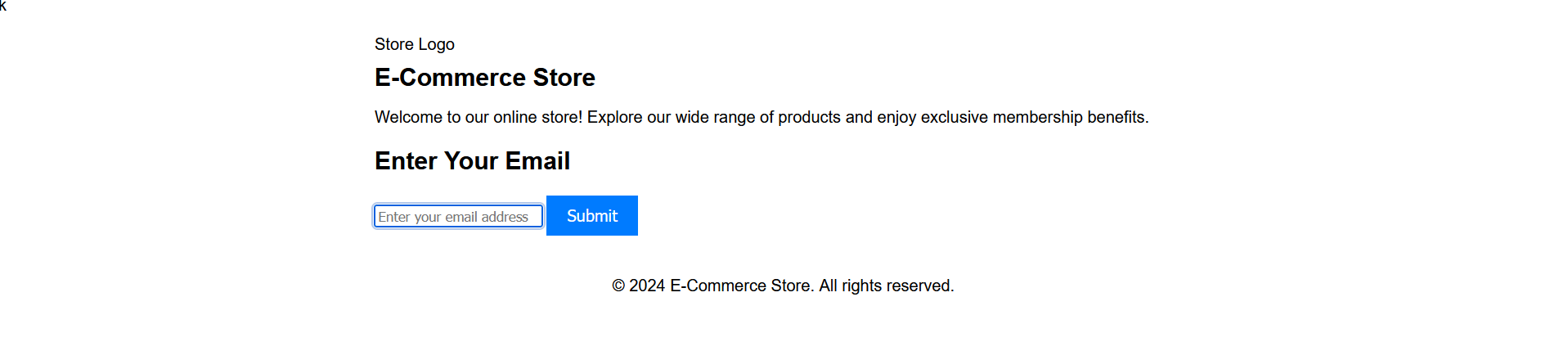




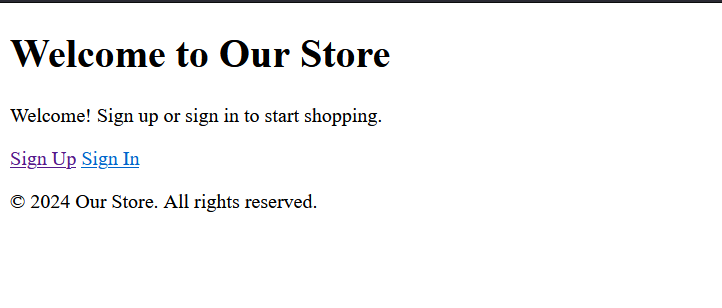


1. **Product demo**

4.1 Landing page (Frontend)

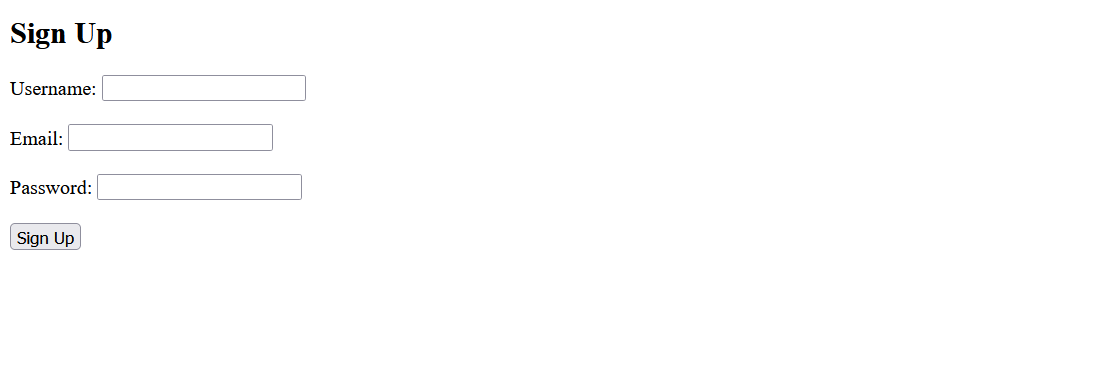


* Modified Landing page (Frontend)



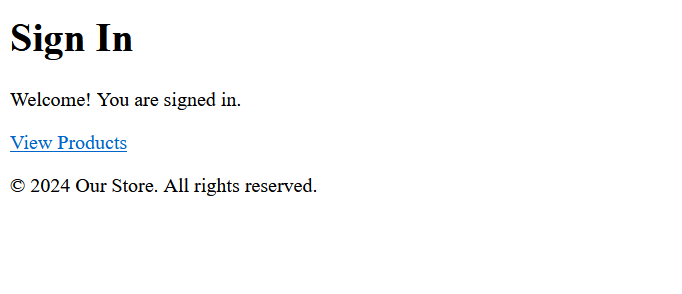
4.2 Authorization

* Signup page

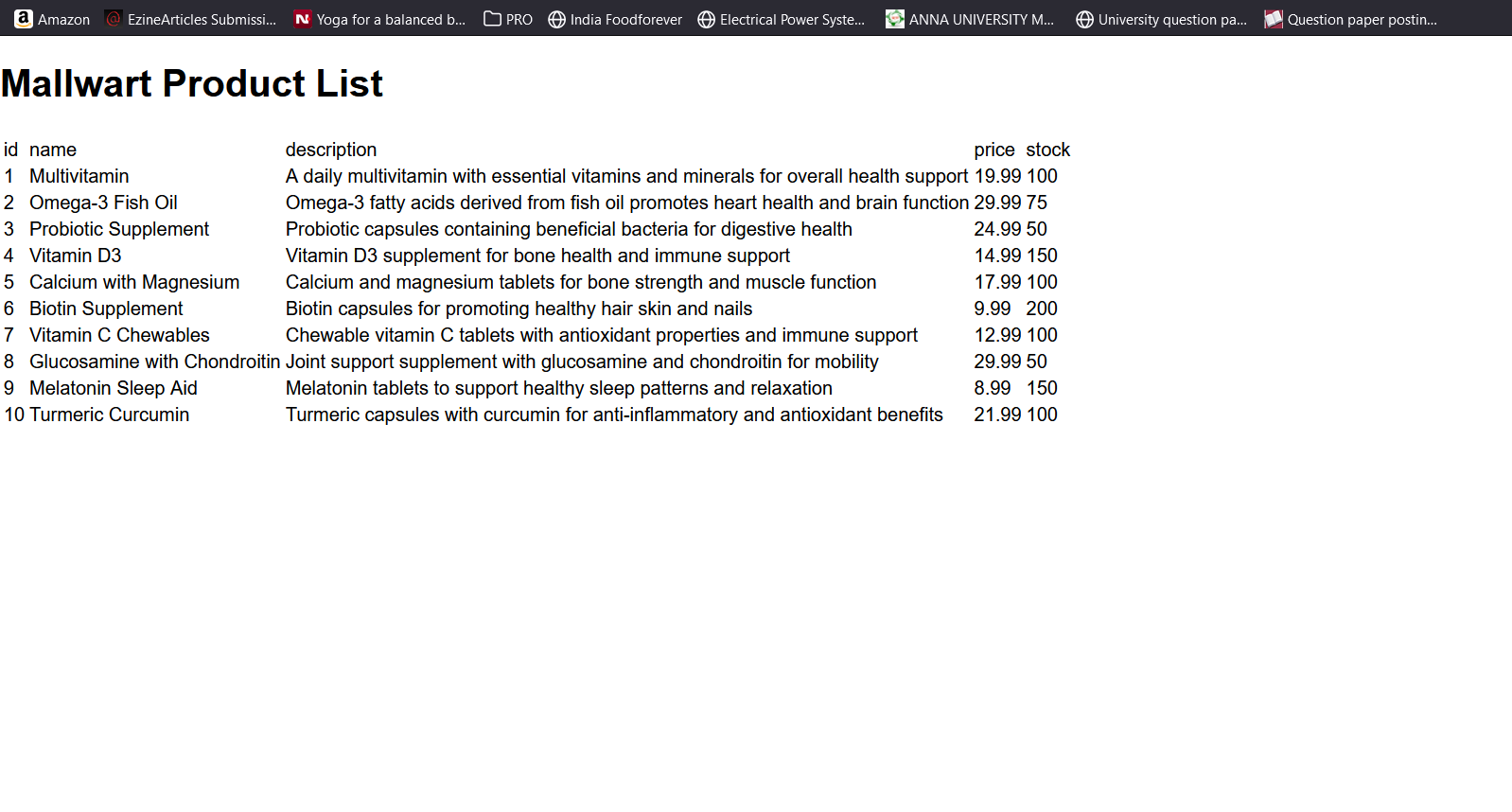




* Signin page



4.3 View Products (**Access Database from Application Tier)**



Implementing a three-tier architecture in Azure requires careful planning and consideration of best practices to ensure scalability, reliability, and security. Here are some tips for best practices and enhanced workflow:

1. **Use Azure Resource Manager (ARM) Templates:** ARM templates allow you to define your infrastructure as code, making it easier to deploy and manage your resources consistently. Define templates for each tier (presentation, application, data) and use parameterization for flexibility.
2. **Separation of Concerns:** Clearly define the responsibilities of each tier. Keep presentation layer focused on user interface, application layer on business logic, and data layer on data storage and retrieval. Use Azure services like Azure App Service, Azure SQL Database, and Azure Storage for each tier respectively.
3. **Scalability:** Design your architecture for scalability to handle varying workloads. Use Azure services like Azure Load Balancer, Azure Application Gateway, and Azure Autoscale to distribute traffic and scale resources dynamically based on demand.
4. **High Availability:** Ensure high availability of your application by distributing resources across multiple Azure regions or Availability Zones. Use Azure Traffic Manager for global load balancing and failover.
5. **Security:** Implement security best practices at each tier. Use Azure Active Directory for authentication and authorization. Encrypt data at rest and in transit using Azure Key Vault and Azure Storage Service Encryption. Implement network security groups (NSGs) to control traffic flow.
6. **Monitoring and Logging:** Monitor the health and performance of your application using Azure Monitor. Set up alerts for key metrics and configure logging to capture diagnostic information. Use Azure Application Insights for application performance monitoring and troubleshooting.
7. **DevOps Practices:** Adopt DevOps practices for automated deployment, continuous integration, and continuous delivery (CI/CD). Use Azure DevOps or GitHub Actions for source control, build, and release pipelines. Implement infrastructure as code (IaC) with tools like Azure Resource Manager (ARM) templates or Terraform.
8. **Testing:** Implement comprehensive testing strategies including unit testing, integration testing, and end-to-end testing. Use Azure DevTest Labs for creating test environments and running automated tests.
9. **Documentation:** Maintain comprehensive documentation for your architecture, including design decisions, deployment instructions, and operational procedures. Use Azure Architecture Center and Azure documentation for guidance and best practices.
10. **Cost Management:** Optimize costs by rightsizing resources, leveraging reserved instances, and using Azure Cost Management for monitoring and optimizing spending. Implement tagging and resource grouping for better cost visibility and management.

By following these best practices and implementing enhanced workflow processes, you can build a robust and scalable three-tier architecture in Azure that meets your business requirements and delivers optimal performance and reliability.