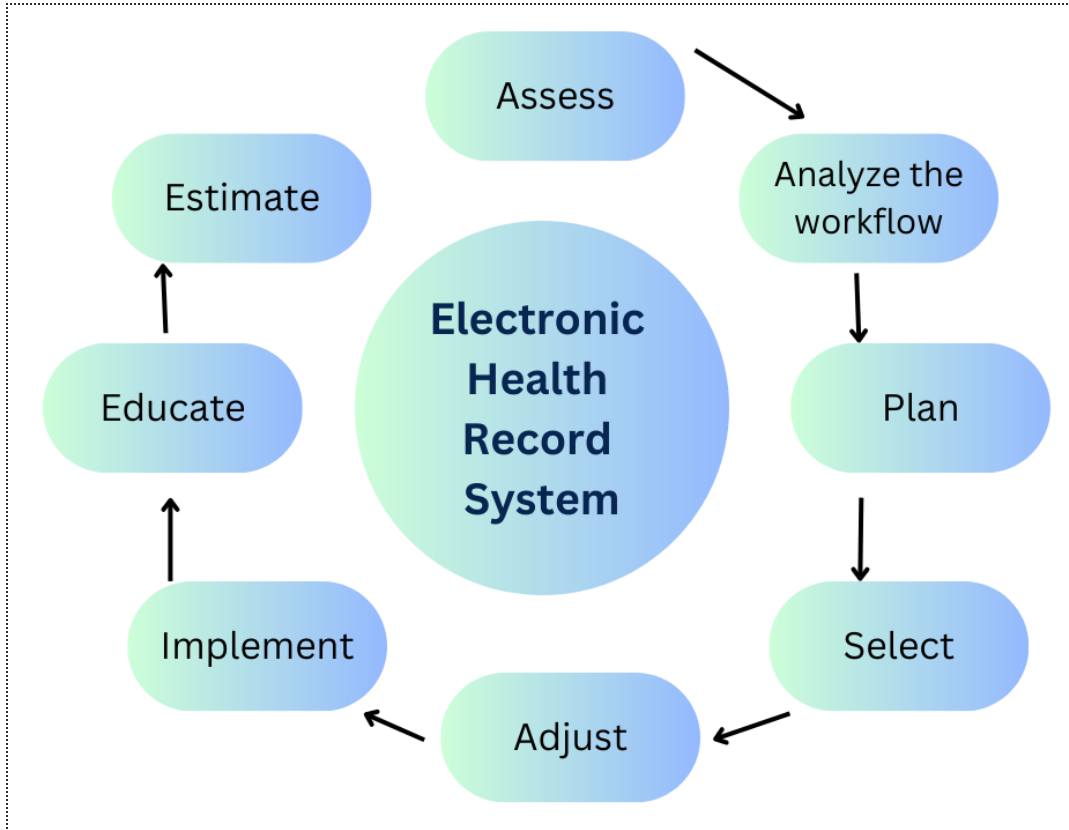


Develop a cloud-based electronic health record (EHR) system using FHIR Azure API

In order to store, exchange, and retrieve patient health data from various healthcare systems and applications securely, this article addresses developing a cloud-based electronic health record (EHR) system utilizing the FHIR Azure API. Fast Healthcare Interoperability Resources also known as FHIR is a healthcare data standard with an application programming interface (API) for representing and exchanging electronic health records. A wide range of FHIR resources, such as patient demographics, prescriptions, test results, treatment plans, and other clinical and administrative data, are supported by the EHR system. It provides customized workflows and alerts to enhance patient care and offer a user-friendly interface for healthcare providers to monitor and manage patient data. To preserve patient privacy and guarantee regulatory compliance, the EHR includes strong security and compliance features.

What is EHR?

An Electronic Health Record digitized representation of a patient's medical history that includes details about their health-related issues, diagnosis, treatments, prescriptions, and lab results. Healthcare practitioners utilize EHRs to manage patient data, communicate with other healthcare professionals engaged in a patient's care, and save patient data.



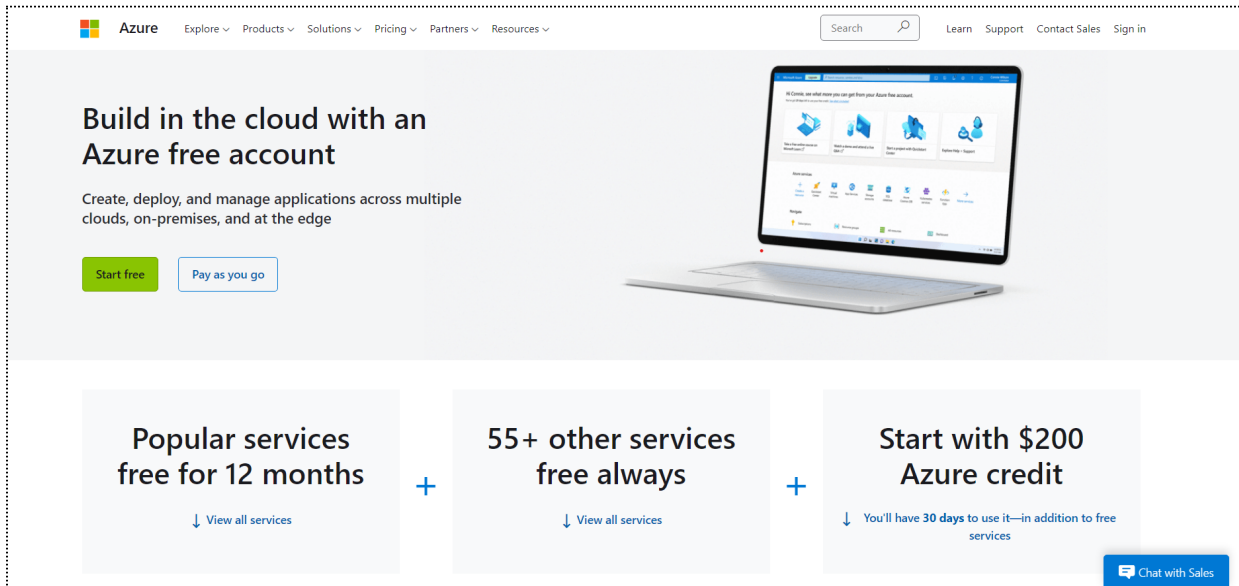
Priority initiative

Electronic health records (EHRs) are a vital component of contemporary healthcare. The purpose of electronic health records is to offer a thorough, accurate, and current record of a patient's medical history, diagnosis and treatment plans. Due to the lack of resources, internet access, and technology, as well as the presence of language difficulties and privacy issues, maintaining medical records in distant locations can be difficult. We require EHRs because of various reasons. Some of them are as follows:

- Scarcity of technology
- Sporadic internet connectivity
- Restricted resources
- Linguistic obstacles
- Privacy concerns

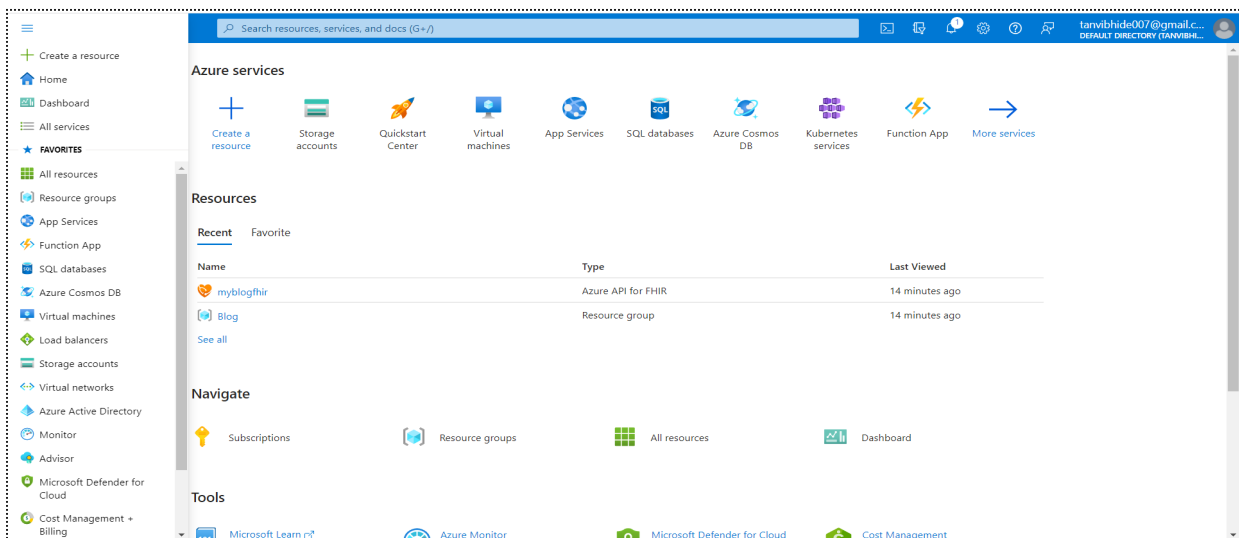
Healthcare Data Management using Azure

1. Create an Azure account: To start using Azure, you first need to create an account on the Azure website (<https://azure.microsoft.com>).



The screenshot shows the Azure website's landing page for free accounts. At the top, there is a navigation bar with the Azure logo and links for Explore, Products, Solutions, Pricing, Partners, and Resources. A search bar and links for Learn, Support, Contact Sales, and Sign in are also present. The main heading reads "Build in the cloud with an Azure free account". Below this, it says "Create, deploy, and manage applications across multiple clouds, on-premises, and at the edge". There are two buttons: "Start free" and "Pay as you go". To the right, a laptop displays the Azure portal interface. Below the main heading, three boxes highlight benefits: "Popular services free for 12 months", "55+ other services free always", and "Start with \$200 Azure credit". Each box has a "View all services" link. A "Chat with Sales" button is in the bottom right corner.

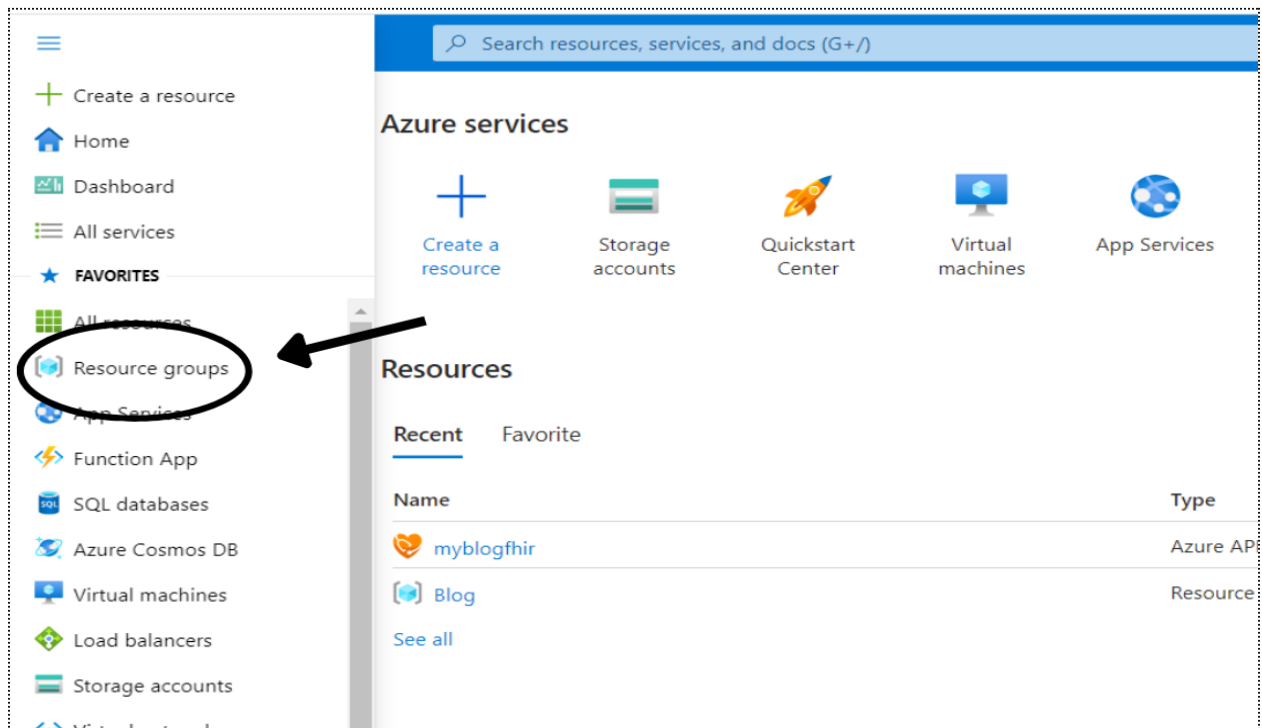
2. Set up a resource group: To create a resource group, go to the Azure portal and click on "Resource groups" in the left-hand menu. Click on "Add" and fill in the required details.



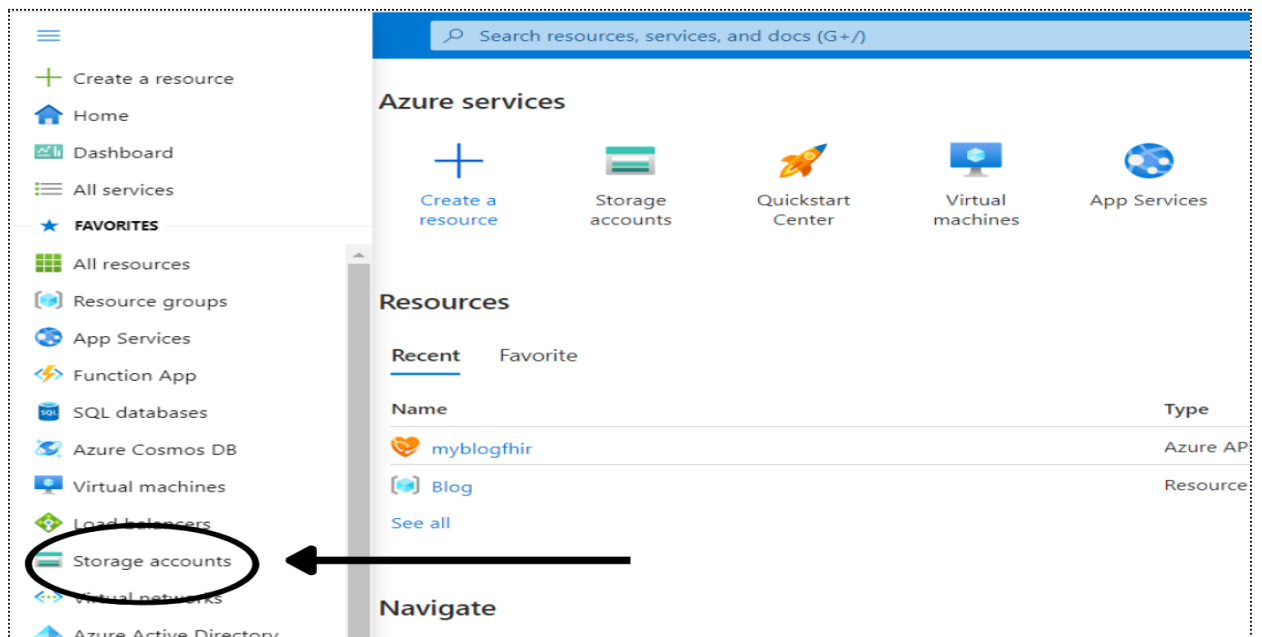
The screenshot shows the Azure portal dashboard. The left-hand navigation menu includes options like "Create a resource", "Home", "Dashboard", "All services", "FAVORITES", "All resources", "Resource groups", "App Services", "Function App", "SQL databases", "Azure Cosmos DB", "Virtual machines", "Load balancers", "Storage accounts", "Virtual networks", "Azure Active Directory", "Monitor", "Advisor", "Microsoft Defender for Cloud", and "Cost Management + Billing". The main content area is titled "Azure services" and features icons for "Create a resource", "Storage accounts", "Quickstart Center", "Virtual machines", "App Services", "SQL databases", "Azure Cosmos DB", "Kubernetes services", "Function App", and "More services". Below this is a "Resources" section with tabs for "Recent" and "Favorite". A table lists recent resources:

Name	Type	Last Viewed
myblogthir	Azure API for FHIR	14 minutes ago
Blog	Resource group	14 minutes ago

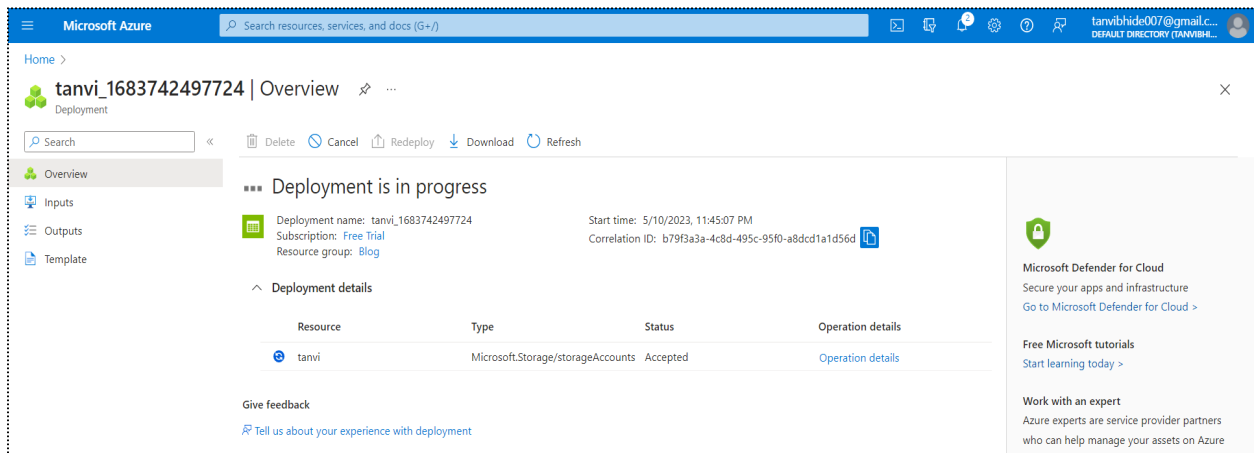
Below the table is a "Navigate" section with icons for "Subscriptions", "Resource groups", "All resources", and "Dashboard". At the bottom, there is a "Tools" section with links to "Microsoft Learn", "Azure Monitor", "Microsoft Defender for Cloud", and "Cost Management".



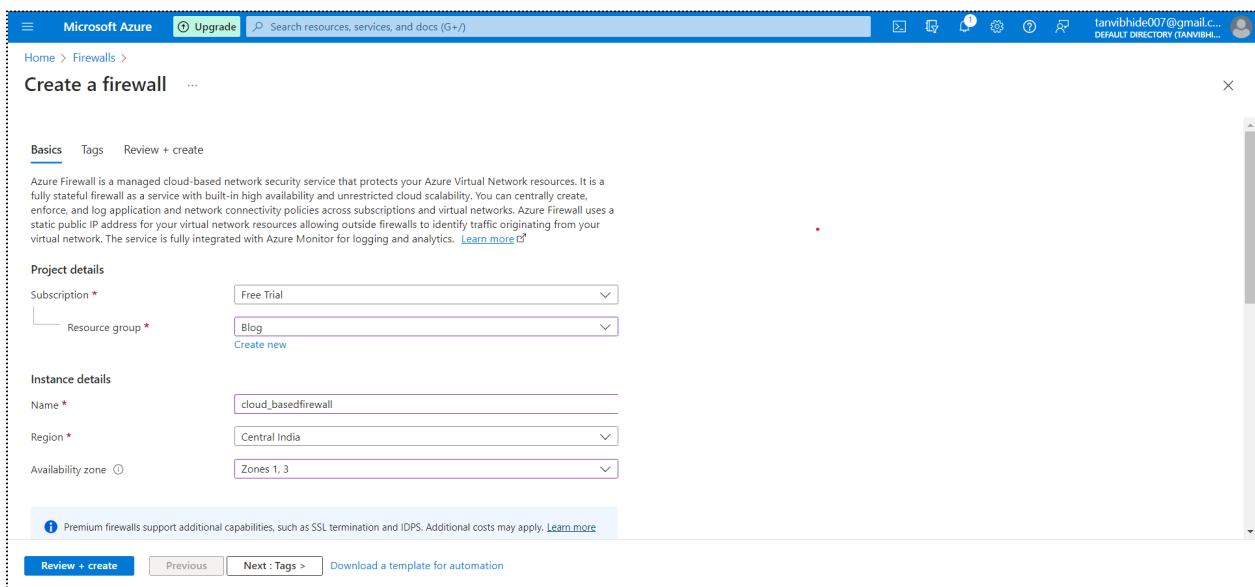
3. Create a storage account: To store patient data, you can create a storage account in Azure. Go to the Azure portal and click on "Storage accounts" in the left-hand menu. Click on "Add" and fill in the required details.



A storage account named 'tanvi' is created in the resource group 'Blog'.

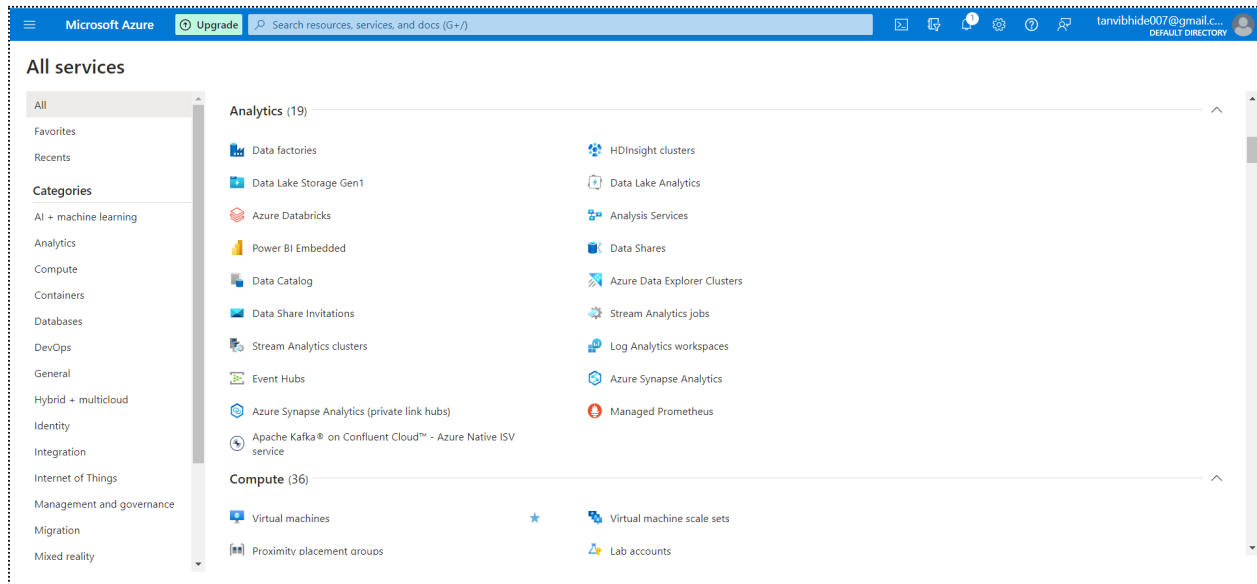


4. Set up security: To ensure that patient data is secure, you can set up security measures such as firewalls, virtual networks, and access controls. You can do this in the Azure portal under the "Security + Compliance" tab.



5. Integrate with EHR systems: To access patient data from EHR (Electronic Health Record) systems, you can use Azure API for FHIR. For this you need to deploy Azure API for FHIR.

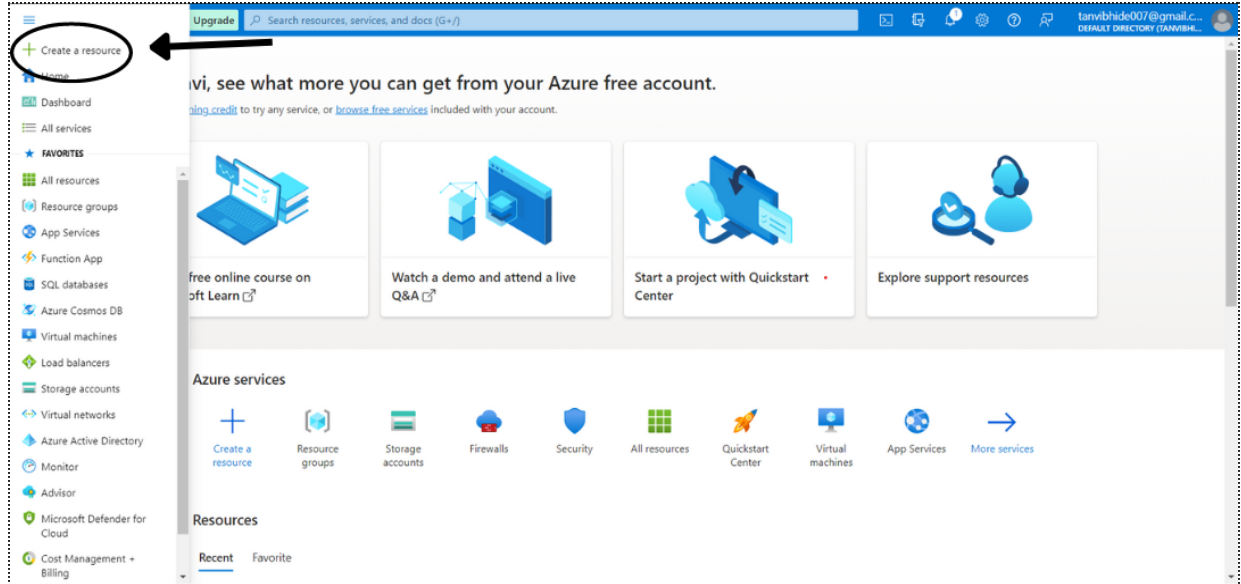
6. Analyze patient data: You can use Azure's analytics services such as Azure Data Factory and Azure Machine Learning to analyze patient data and gain insights into patient health. These services can be set up in the Azure portal under the "Analytics" tab.



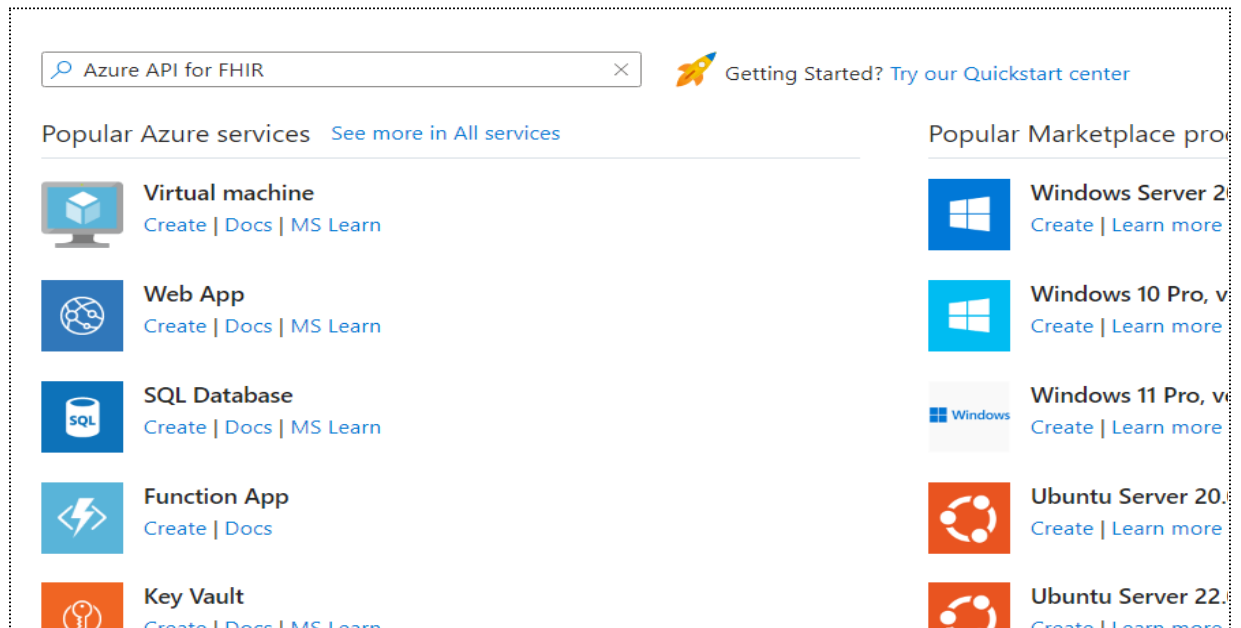
7. Manage resources: Using Azure's monitoring and management capabilities, you can manage resources once your healthcare data management system has been configured there. Utilizing these tools, you may monitor consumption, performance, expenses, and adjustments will be made as necessary.

Deploy Azure API for FHIR using Azure portal

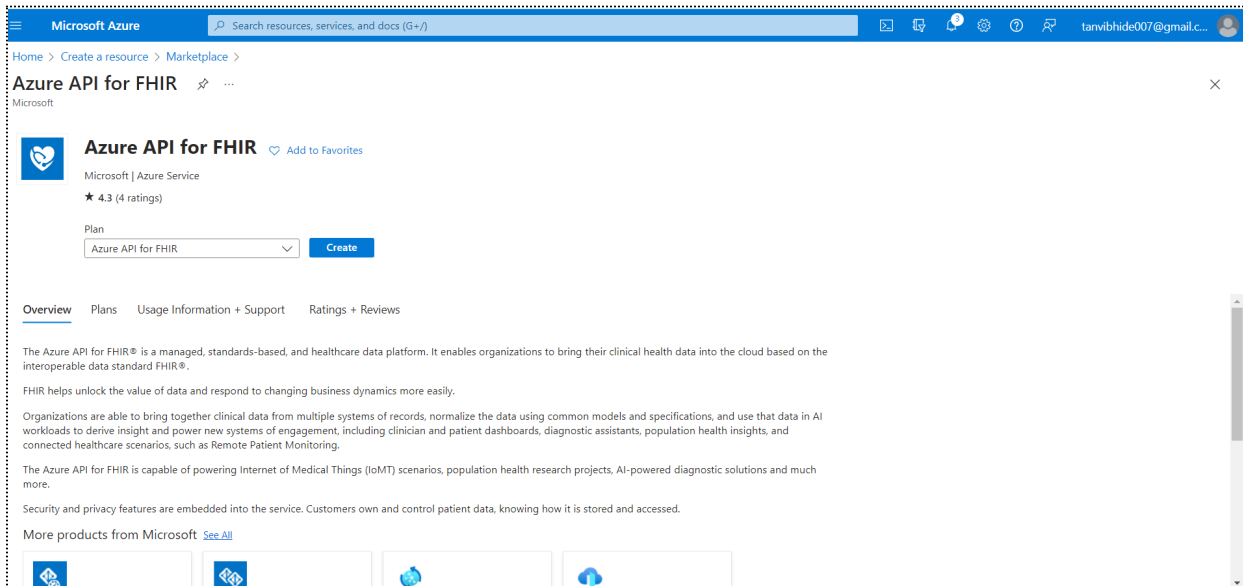
1. Create a new resource



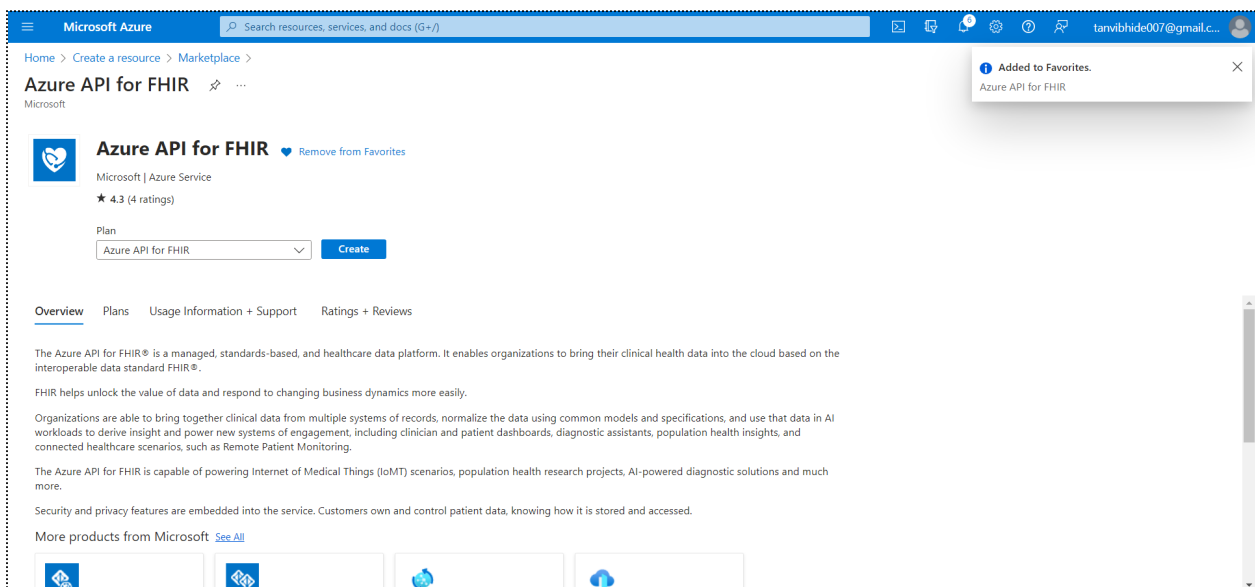
2. Search for Azure API for FHIR



3. Create Azure API for FHIR account



For easier access later, you can add it to your favorites.



4. Enter account details

Microsoft Azure Search resources, services, and docs (G+)

Home > Create a resource > Marketplace > Azure API for FHIR >

Create Azure API for FHIR

* Basics * Networking * Additional settings Tags Review + create

Azure API for FHIR® is a managed, standards-based, and compliant healthcare data platform. It enables organizations to bring their clinical health data into the cloud based on the interoperable data standard FHIR®.

Project details

Subscription * Free Trial

Resource group * (New) Blog
[Create new](#)

Instance details

Account name * myblogfhir.azurehealthcareapis.com

Location * Central India

FHIR version * R4

[Review < create](#) [Next: Networking >](#)

Microsoft Azure Search resources, services, and docs (G+)

Home > Create a resource > Marketplace > Azure API for FHIR >

Create Azure API for FHIR

Validation success.

* Basics * Networking * Additional settings Tags Review + create

Basics

Subscription	Free Trial
Resource group	Blog
Account name	myblogfhir.azurehealthcareapis.com
Location	Central India
FHIR version	R4

Networking

Connectivity method	Public endpoint (all networks)
---------------------	--------------------------------

Additional settings

Authority	https://login.microsoftonline.com/e1b228d4-30d1-4044-97da-24e932abb2d6
Audience	https://myblogfhir.azurehealthcareapis.com
Allowed object IDs	
Provisioned throughput (RU/s)	400

[Create](#) [Previous](#) [Download a template for automation](#)

The screenshot shows the 'Create Azure API for FHIR' configuration page in the Microsoft Azure portal. The page is divided into several sections:

- Authentication:** Authority is set to `https://login.microsoftonline.com/e1b228d4-30d1-4044-97da-24e932...` and Audience is set to `https://myblogfhir.azurehealthcareapis.com`. Allowed object IDs is currently empty.
- SMART on FHIR:** A note explaining that SMART on FHIR is a set of open specifications to integrate partner applications with FHIR servers. A link is provided to learn more about steps to enable SMART capabilities per ONC G(10) certification requirements.
- Database settings:** Provisioned throughput (RU/s) is set to 400. Data encryption is set to Service-managed key (selected) or Customer-managed key (unselected). A link is provided to learn more about using customer-managed keys.

At the bottom of the page, there are three buttons: 'Review < create', 'Previous', and 'Next: Tags >'.

Implementation in VSCode

1. Import necessary libraries

```
import adal
import requests
import json
```

2. Define authentication credentials

```
authority_uri = 'https://login.microsoftonline.com/{tenant_id}'
resource_uri = 'https://{resource_name}.azurehealthcareapis.com'
client_id = '<your_client_id>'
client_secret = '<your_client_secret>'
```

Replace the `{tenant_id}`, `{resource_name}`, `{client_id}` and `{client_secret}` placeholders with your actual authentication credentials

3. Authenticate

```
context = adal.AuthenticationContext(authority_uri)
token = context.acquire_token_with_client_credentials(
    resource_uri,
    client_id,
    client_secret
)['accessToken']
```

Make a GET request to the FHIR API:

```
headers = {'Authorization': 'Bearer ' + token}
url = resource_uri + '/Patient?_count=10'
response = requests.get(url, headers=headers)
```

This sends a GET request to the FHIR API, requesting a list of 10 patient records.

4. Parse the response

```
if response.status_code == 200:
    data = response.json()
    for patient in data['entry']:
        name = patient['resource']['name'][0]
        given_name = name.get('given', [''])[0]
        family_name = name.get('family', '')
        print(f"Patient Name: {given_name} {family_name}")
else:
    print(f"Request failed with status code: {response.status_code}")
```

This checks if the response status code is 200 (i.e., if the request was successful), and if so, parses the JSON response and prints out the name of each patient in the list. In order to receive an access token, we must first login with Azure AD using our credentials (tenant ID, client ID, and client secret). In order to get patient data for a

given patient ID, we next utilize this access token to issue a GET request to the FHIR API.

```
import adal
import requests

# Azure AD credentials
tenant_id = '<your tenant ID>'
client_id = '<your client ID>'
client_secret = '<your client secret>'
# FHIR API endpoint
fhir_url = '<your FHIR API endpoint>'
patient_id = '<desired patient ID>'
# Azure AD authentication
context= adal.AuthenticationContext('https://login.microsoftonline.com/' +
tenant_id)
token=context.acquire_token_with_client_credentials(fhir_url, client_id,
client_secret)
# Get patient data
headers = {'Authorization': 'Bearer ' + token['accessToken'],
'Content-Type': 'application/json'}
response = requests.get(fhir_url + '/Patient/' + patient_id,
headers=headers)
# Print patient data
print(response.json())
```

A tenant is an instance of Azure AD that represents an organization.

An application that requires access to resources or services from Azure AD or other identity providers is referred to as a client. A web application, a mobile application, a desktop application, or even another service or API can serve as the client.

In order to request access to resources or services, a client must first register with Azure AD. Once registered, the client is given a special client ID and client secret that are used to authenticate and authorize the client.

Once we have the patient data in JSON format, we can use the FHIR client library to parse the data and extract the desired information (in this case, patient name, gender,

and birthdate). The FHIR client library provides a convenient way to work with FHIR resources in Python and handles the mapping of JSON data to Python objects.

```
from fhirclient import client
import json
# Assume patient_data is the JSON data returned from a GET request to the
# FHIR API
# Create a FHIR client instance
fhir_client = client.FHIRClient(
    server_url='http://my-fhir-server-url.com/fhir',
    use_format_param=True,
    verify_ssl=False,
)
# Use the FHIR client to parse the patient data
parsed_data = fhir_client.resource('Patient', json.loads(patient_data))
# Extract patient name, gender, and birthdate
patient_name = parsed_data.name[0].given[0] + ' ' +
    parsed_data.name[0].family
patient_gender = parsed_data.gender
patient_birthdate = parsed_data.birthDate.isostring
# Print out the extracted information
print('Patient Name: ' + patient_name)
print('Patient Gender: ' + patient_gender)
print('Patient Birthdate: ' + patient_birthdate)
```

In this illustration, we'll suppose that `patient_data` contains the JSON information that the FHIR API returned in response to a GET request. The patient name, gender, and birthday are extracted from the data using the FHIR client library. Finally, we print the information that was extracted.

Advantages of EHR

Electronic health records (EHRs) have a number of advantages in the healthcare industry, some of which are as follows:

- **Efficiency gains:** By giving healthcare professionals instant access to patient data including medical histories, test findings, and treatment plans, EHRs may streamline workflow and save time.
- **Improved care coordination:** EHRs make it simple for various healthcare professionals to communicate patient data, which can assist to guarantee that everyone engaged in a patient's treatment is on the same page.
- **Savings:** EHRs can assist in lowering the expenses related to paper-based records, such as printing, storing, and retrieving.
- **Enhancing patient outcomes:** EHRs can give healthcare professionals more thorough and accurate patient information, enabling improved decision-making and eventually improving patient outcomes.

Business Benefits

Using Azure FHIR API for EHR has various business advantages, including:

- **Efficiency gain:** Azure FHIR API makes it easier to store and retrieve patient health data, which can help healthcare practitioners work more effectively.
- **Improved patient care:** Azure FHIR API can enhance the standard of care given to patients by enabling rapid and simple access to a patient's medical history and health data.
- **Savings:** By offering a scalable, cloud-based solution that is simple to integrate with other healthcare systems, using Azure FHIR API can help lower the cost of operating and maintaining EHR systems.
- **Increased interoperability:** The Azure FHIR API makes use of a standardized data model to encourage interoperability between various healthcare systems and

organizations, which can improve care coordination and improve patient outcomes.

- Data security: Azure FHIR API offers strong security features, such as encryption, access controls, and compliance with legal standards like HIPAA, to safeguard patient health data.

In conclusion, electronic health records (EHR) have revolutionized the healthcare sector by giving professionals access to patient health data from a single location at any time. EHR system adoption provides a number of advantages, including better patient care, more effective healthcare delivery, fewer medical mistakes, and higher patient participation. Moreover, by using the Azure FHIR API, EHR systems have improved and made healthcare analytics possible, which may aid healthcare companies in making wise decisions.