Azure Services Setup Guide

Medical Document Chatbot - Step-by-Step Implementation

Version: 1.0
Date: October 15, 2025
Region: Central India (Primary), South India (Backup)
Estimated Setup Time: 8-12 hours

Table of Contents

---

1. Pre-Setup Requirements

1.1 Prerequisites Checklist

- [ ] Azure account with active subscription
- [ ] Valid credit card or payment method
- [ ] Azure CLI installed on your machine
- [ ] Basic understanding of cloud services
- [ ] Text editor (VS Code recommended)
- [ ] Minimum budget: $100-150/month

1.2 Install Required Tools

Install Azure CLI

Windows:

```
# Download and run installer
Invoke-WebRequest -Uri https://aka.ms/installazurecliwindows -OutFile .\AzureCLI.msi
Start-Process msiexec.exe -Wait -ArgumentList '/I AzureCLI.msi /quiet'
```

macOS:

```
brew update && brew install azure-cli
```

Linux:

```
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
```

Verify Installation

```
az --version
az login
```

1.3 Set Default Subscription

```
# List all subscriptions
az account list --output table

# Set default subscription
az account set --subscription "YOUR_SUBSCRIPTION_ID"

# Verify
az account show
```

---

2. Azure Account & Subscription Setup

2.1 Create Azure Account

1. Go to https://azure.microsoft.com/free (https://azure.microsoft.com/free)
2. Click "Start free" or "Pay as you go"
3. Sign in with Microsoft account or create new
4. Complete verification (phone, credit card)
5. Choose subscription type:
    - **Free Trial:** $200 credit for 30 days
    - **Pay-As-You-Go:** For production use

2.2 Enable Required Resource Providers

```
# Enable AI services
az provider register --namespace Microsoft.CognitiveServices
az provider register --namespace Microsoft.MachineLearningServices

# Enable compute services
az provider register --namespace Microsoft.Web
az provider register --namespace Microsoft.ContainerInstance

# Enable storage
az provider register --namespace Microsoft.Storage
az provider register --namespace Microsoft.Sql
az provider register --namespace Microsoft.DocumentDB

# Enable monitoring
az provider register --namespace Microsoft.Insights
az provider register --namespace Microsoft.OperationalInsights

# Verify registration status
az provider list --query "[?registrationState=='Registered'].namespace" --output table
```

2.3 Request Quota Increases (if needed)

For Azure OpenAI and AI services:

1. Go to Azure Portal → Subscriptions → Usage + quotas
2. Search for "OpenAI" or "Cognitive Services"
3. Request increase if needed (especially for GPT-4)

---

3. Resource Group & Networking

3.1 Create Resource Group

```
# Variables
RESOURCE_GROUP="medical-chatbot-rg"
LOCATION="centralindia"
TAGS="Environment=Production Project=MedicalChatbot"

# Create resource group
az group create \
  --name $RESOURCE_GROUP \
  --location $LOCATION \
  --tags $TAGS

# Verify
az group show --name $RESOURCE_GROUP
```

3.2 Create Virtual Network (VNet)

```
# Create VNet
az network vnet create \
  --resource-group $RESOURCE_GROUP \
  --name medical-chatbot-vnet \
  --address-prefix 10.0.0.0/16 \
  --subnet-name default-subnet \
  --subnet-prefix 10.0.1.0/24

# Create additional subnets
az network vnet subnet create \
  --resource-group $RESOURCE_GROUP \
  --vnet-name medical-chatbot-vnet \
  --name app-service-subnet \
  --address-prefix 10.0.2.0/24

az network vnet subnet create \
  --resource-group $RESOURCE_GROUP \
  --vnet-name medical-chatbot-vnet \
  --name ai-services-subnet \
  --address-prefix 10.0.3.0/24
```

3.3 Create Network Security Group (NSG)

```
# Create NSG
az network nsg create \
  --resource-group $RESOURCE_GROUP \
  --name medical-chatbot-nsg

# Add security rules
az network nsg rule create \
  --resource-group $RESOURCE_GROUP \
  --nsg-name medical-chatbot-nsg \
  --name AllowHTTPS \
  --priority 100 \
  --source-address-prefixes '*' \
  --destination-port-ranges 443 \
  --access Allow \
  --protocol Tcp

az network nsg rule create \
  --resource-group $RESOURCE_GROUP \
  --nsg-name medical-chatbot-nsg \
  --name AllowHTTP \
  --priority 110 \
  --source-address-prefixes '*' \
  --destination-port-ranges 80 \
  --access Allow \
  --protocol Tcp
```

---

4. Security Foundation - Azure Key Vault

4.1 Create Key Vault

```
# Variables
KEYVAULT_NAME="medical-chatbot-kv-$(openssl rand -hex 4)"

# Create Key Vault
az keyvault create \
  --name $KEYVAULT_NAME \
  --resource-group $RESOURCE_GROUP \
  --location $LOCATION \
  --enable-purge-protection true \
  --enable-soft-delete true \
  --retention-days 90

# Enable access for your account
USER_OBJECT_ID=$(az ad signed-in-user show --query id -o tsv)

az keyvault set-policy \
  --name $KEYVAULT_NAME \
  --object-id $USER_OBJECT_ID \
  --secret-permissions get list set delete
```

4.2 Store Initial Secrets (Placeholders)

```
# Store placeholder secrets (you'll update these later)
az keyvault secret set \
  --vault-name $KEYVAULT_NAME \
  --name "openai-api-key" \
  --value "PLACEHOLDER"

az keyvault secret set \
  --vault-name $KEYVAULT_NAME \
  --name "sql-connection-string" \
  --value "PLACEHOLDER"

az keyvault secret set \
  --vault-name $KEYVAULT_NAME \
  --name "storage-connection-string" \
  --value "PLACEHOLDER"
```

□ **Save to Configuration Sheet:**

- Key Vault Name: `$KEYVAULT_NAME`
- Key Vault URL: `https://$KEYVAULT_NAME.vault.azure.net/`

---

5. Storage Services Setup

5.1 Azure Blob Storage

Step 1: Create Storage Account

```
# Variables
STORAGE_ACCOUNT="medchatbot$(openssl rand -hex 4)"

# Create storage account
az storage account create \
  --name $STORAGE_ACCOUNT \
  --resource-group $RESOURCE_GROUP \
  --location $LOCATION \
  --sku Standard_LRS \
  --kind StorageV2 \
  --access-tier Hot \
  --https-only true \
  --min-tls-version TLS1_2

# Enable soft delete
az storage blob service-properties delete-policy update \
  --account-name $STORAGE_ACCOUNT \
  --enable true \
  --days-retained 7
```

Step 2: Create Blob Containers

```
# Get storage account key
STORAGE_KEY=$(az storage account keys list \
  --resource-group $RESOURCE_GROUP \
  --account-name $STORAGE_ACCOUNT \
  --query '[0].value' -o tsv)

# Create containers
az storage container create \
  --name prescription-uploads \
  --account-name $STORAGE_ACCOUNT \
  --account-key $STORAGE_KEY \
  --public-access off

az storage container create \
  --name extracted-data \
  --account-name $STORAGE_ACCOUNT \
  --account-key $STORAGE_KEY \
  --public-access off

az storage container create \
  --name medical-images \
  --account-name $STORAGE_ACCOUNT \
  --account-key $STORAGE_KEY \
  --public-access off
```

Step 3: Configure CORS (for web uploads)

```
az storage cors add \
  --account-name $STORAGE_ACCOUNT \
  --account-key $STORAGE_KEY \
  --services b \
  --methods GET POST PUT \
  --origins '*' \
  --allowed-headers '*' \
  --max-age 3600
```

Step 4: Store Connection String in Key Vault

```
STORAGE_CONNECTION_STRING=$(az storage account show-connection-string \
  --name $STORAGE_ACCOUNT \
  --resource-group $RESOURCE_GROUP \
  --query connectionString -o tsv)

az keyvault secret set \
  --vault-name $KEYVAULT_NAME \
  --name "storage-connection-string" \
  --value "$STORAGE_CONNECTION_STRING"
```

□ **Save to Configuration Sheet:**

- Storage Account Name: `$STORAGE_ACCOUNT`
- Storage Account Key: `$STORAGE_KEY`
- Blob Endpoint: `https://$STORAGE_ACCOUNT.blob.core.windows.net/`

5.2 Azure Redis Cache

```
# Create Redis Cache
REDIS_NAME="medical-chatbot-redis"

az redis create \
  --name $REDIS_NAME \
  --resource-group $RESOURCE_GROUP \
  --location $LOCATION \
  --sku Basic \
  --vm-size c1 \
  --enable-non-ssl-port false

# Get Redis connection info
REDIS_HOST=$(az redis show \
  --name $REDIS_NAME \
  --resource-group $RESOURCE_GROUP \
  --query hostName -o tsv)

REDIS_KEY=$(az redis list-keys \
  --name $REDIS_NAME \
  --resource-group $RESOURCE_GROUP \
  --query primaryKey -o tsv)

# Store in Key Vault
az keyvault secret set \
  --vault-name $KEYVAULT_NAME \
  --name "redis-connection-string" \
  --value "$REDIS_HOST:6380,password=$REDIS_KEY,ssl=True"
```

Save to Configuration Sheet:

- Redis Name: `$REDIS_NAME`
- Redis Host: `$REDIS_HOST`
- Redis Port: `6380` (SSL)
- Redis Key: `$REDIS_KEY`

---

6. Database Services Setup

6.1 Azure SQL Database

Step 1: Create SQL Server

```
# Variables
SQL_SERVER="medical-chatbot-sql-$(openssl rand -hex 4)"
SQL_ADMIN="sqladmin"
SQL_PASSWORD="ComplexP@ssw0rd123!"  # Change this!

# Create SQL Server
az sql server create \
  --name $SQL_SERVER \
  --resource-group $RESOURCE_GROUP \
  --location $LOCATION \
  --admin-user $SQL_ADMIN \
  --admin-password $SQL_PASSWORD

# Configure firewall (allow Azure services)
az sql server firewall-rule create \
  --resource-group $RESOURCE_GROUP \
  --server $SQL_SERVER \
  --name AllowAzureServices \
  --start-ip-address 0.0.0.0 \
  --end-ip-address 0.0.0.0

# Add your IP (for management)
MY_IP=$(curl -s ifconfig.me)
az sql server firewall-rule create \
  --resource-group $RESOURCE_GROUP \
  --server $SQL_SERVER \
  --name AllowMyIP \
  --start-ip-address $MY_IP \
  --end-ip-address $MY_IP
```

Step 2: Create Databases

```
# Create users database
az sql db create \
  --resource-group $RESOURCE_GROUP \
  --server $SQL_SERVER \
  --name medicalchatbot-users \
  --service-objective S2 \
  --backup-storage-redundancy Local

# Create drugs database
az sql db create \
  --resource-group $RESOURCE_GROUP \
  --server $SQL_SERVER \
  --name medicalchatbot-drugs \
  --service-objective S2 \
  --backup-storage-redundancy Local
```

Step 3: Store Connection String

```
SQL_CONNECTION_STRING="Server=tcp:$SQL_SERVER.database.windows.net,1433;Database=medicalchatbot-users;User ID=$SQL_ADMIN;Password=$S(

az keyvault secret set \
  --vault-name $KEYVAULT_NAME \
  --name "sql-connection-string" \
  --value "$SQL_CONNECTION_STRING"
```

□ Save to Configuration Sheet:

- SQL Server: `$SQL_SERVER.database.windows.net`
- Admin Username: `$SQL_ADMIN`
- Admin Password: `$SQL_PASSWORD`
- Database Names: `medicalchatbot-users`, `medicalchatbot-drugs`

6.2 Azure Cosmos DB

Step 1: Create Cosmos DB Account

```
COSMOS_ACCOUNT="medical-chatbot-cosmos-$(openssl rand -hex 4)"

az cosmosdb create \
  --name $COSMOS_ACCOUNT \
  --resource-group $RESOURCE_GROUP \
  --locations regionName=$LOCATION failoverPriority=0 \
  --default-consistency-level Session \
  --enable-automatic-failover false
```

Step 2: Create Database and Containers

```
# Create database
az cosmosdb sql database create \
  --account-name $COSMOS_ACCOUNT \
  --resource-group $RESOURCE_GROUP \
  --name medicalchatbot

# Create conversations container
az cosmosdb sql container create \
  --account-name $COSMOS_ACCOUNT \
  --resource-group $RESOURCE_GROUP \
  --database-name medicalchatbot \
  --name conversations \
  --partition-key-path /user_id \
  --throughput 400

# Create messages container
az cosmosdb sql container create \
  --account-name $COSMOS_ACCOUNT \
  --resource-group $RESOURCE_GROUP \
  --database-name medicalchatbot \
  --name messages \
  --partition-key-path /conversation_id \
  --throughput 400
```

Step 3: Get Connection String

```
COSMOS_CONNECTION_STRING=$(az cosmosdb keys list \
  --name $COSMOS_ACCOUNT \
  --resource-group $RESOURCE_GROUP \
  --type connection-strings \
  --query "connectionStrings[0].connectionString" -o tsv)

az keyvault secret set \
  --vault-name $KEYVAULT_NAME \
  --name "cosmos-connection-string" \
  --value "$COSMOS_CONNECTION_STRING"
```

□ Save to Configuration Sheet:

- Cosmos Account: `$COSMOS_ACCOUNT`
- Cosmos Endpoint: `https://$COSMOS_ACCOUNT.documents.azure.com:443/`
- Database Name: `medicalchatbot`
- Containers: `conversations`, `messages`

---

7. AI Services Setup

7.1 Azure OpenAI Service

Step 1: Create Azure OpenAI Resource

```
OPENAI_NAME="medical-chatbot-openai"

az cognitiveservices account create \
  --name $OPENAI_NAME \
  --resource-group $RESOURCE_GROUP \
  --location eastus \
  --kind OpenAI \
  --sku S0 \
  --custom-domain $OPENAI_NAME
```

Note: GPT-4 might only be available in specific regions (eastus, swedencentral). Check availability.

Step 2: Deploy GPT-4 Model

```
# Deploy GPT-4 (orchestrator)
az cognitiveservices account deployment create \
  --name $OPENAI_NAME \
  --resource-group $RESOURCE_GROUP \
  --deployment-name gpt-4-deployment \
  --model-name gpt-4 \
  --model-version "0613" \
  --model-format OpenAI \
  --sku-capacity 10 \
  --sku-name Standard

# Deploy text-embedding-ada-002 (for RAG)
az cognitiveservices account deployment create \
  --name $OPENAI_NAME \
  --resource-group $RESOURCE_GROUP \
  --deployment-name embedding-deployment \
  --model-name text-embedding-ada-002 \
  --model-version "2" \
  --model-format OpenAI \
  --sku-capacity 10 \
  --sku-name Standard
```

Step 3: Get API Key and Endpoint

```
OPENAI_KEY=$(az cognitiveservices account keys list \
  --name $OPENAI_NAME \
  --resource-group $RESOURCE_GROUP \
  --query key1 -o tsv)

OPENAI_ENDPOINT=$(az cognitiveservices account show \
  --name $OPENAI_NAME \
  --resource-group $RESOURCE_GROUP \
  --query properties.endpoint -o tsv)

# Store in Key Vault
az keyvault secret set \
  --vault-name $KEYVAULT_NAME \
  --name "openai-api-key" \
  --value "$OPENAI_KEY"

az keyvault secret set \
  --vault-name $KEYVAULT_NAME \
  --name "openai-endpoint" \
  --value "$OPENAI_ENDPOINT"
```

Save to Configuration Sheet:

- OpenAI Resource: `$OPENAI_NAME`
- OpenAI Endpoint: `$OPENAI_ENDPOINT`

- OpenAI API Key: `$OPENAI_KEY`
- GPT-4 Deployment: `gpt-4-deployment`
- Embedding Deployment: `embedding-deployment`

7.2 Azure Document Intelligence

```
DOC_INTEL_NAME="medical-chatbot-docintel"

az cognitiveservices account create \
  --name $DOC_INTEL_NAME \
  --resource-group $RESOURCE_GROUP \
  --location $LOCATION \
  --kind FormRecognizer \
  --sku S0

# Get keys
DOC_INTEL_KEY=$(az cognitiveservices account keys list \
  --name $DOC_INTEL_NAME \
  --resource-group $RESOURCE_GROUP \
  --query key1 -o tsv)

DOC_INTEL_ENDPOINT=$(az cognitiveservices account show \
  --name $DOC_INTEL_NAME \
  --resource-group $RESOURCE_GROUP \
  --query properties.endpoint -o tsv)

# Store in Key Vault
az keyvault secret set \
  --vault-name $KEYVAULT_NAME \
  --name "document-intelligence-key" \
  --value "$DOC_INTEL_KEY"

az keyvault secret set \
  --vault-name $KEYVAULT_NAME \
  --name "document-intelligence-endpoint" \
  --value "$DOC_INTEL_ENDPOINT"
```

 Save to Configuration Sheet:

- Document Intelligence Name: `$DOC_INTEL_NAME`
- Endpoint: `$DOC_INTEL_ENDPOINT`
- API Key: `$DOC_INTEL_KEY`

7.3 Azure AI Search (Vector Database)

```
AI_SEARCH_NAME="medical-chatbot-search-$(openssl rand -hex 4)"

az search service create \
  --name $AI_SEARCH_NAME \
  --resource-group $RESOURCE_GROUP \
  --location $LOCATION \
  --sku standard \
  --partition-count 1 \
  --replica-count 2

# Get admin key
AI_SEARCH_KEY=$(az search admin-key show \
  --service-name $AI_SEARCH_NAME \
  --resource-group $RESOURCE_GROUP \
  --query primaryKey -o tsv)

# Store in Key Vault
az keyvault secret set \
  --vault-name $KEYVAULT_NAME \
  --name "ai-search-key" \
  --value "$AI_SEARCH_KEY"
```

Save to Configuration Sheet:

- AI Search Name: `$AI_SEARCH_NAME`
- Search Endpoint: `https://$AI_SEARCH_NAME.search.windows.net`
- Admin Key: `$AI_SEARCH_KEY`

7.4 Azure AI Foundry (Model Deployments)

Note: Azure AI Foundry deployments are typically done through Azure ML Studio UI.

Step 1: Create Azure Machine Learning Workspace

```
AML_WORKSPACE="medical-chatbot-aml"

az ml workspace create \
  --name $AML_WORKSPACE \
  --resource-group $RESOURCE_GROUP \
  --location $LOCATION
```

Step 2: Deploy Models via Azure AI Studio

1. Go to https://ai.azure.com (https://ai.azure.com)
2. Sign in with your Azure account
3. Select your workspace: `$AML_WORKSPACE`
4. Navigate to "Model catalog"
5. Deploy the following models:

Model 1: m42-health-llama3-med42-70b

- Click "Deploy" → "Real-time endpoint"
- Name: `med42-llama3-endpoint`
- Instance type: `Standard_NC24ads_A100_v4`
- Instance count: 2
- Click "Deploy"

Model 2: microsoft-biogpt-large

- Click "Deploy" → "Real-time endpoint"
- Name: `biogpt-endpoint`
- Instance type: `Standard_NC12s_v3`
- Instance count: 1
- Click "Deploy"

Model 3: BiomedCLIP-PubMedBERT

- Click "Deploy" → "Real-time endpoint"
- Name: `biomedclip-endpoint`
- Instance type: `Standard_D4s_v3`
- Instance count: 1
- Click "Deploy"

Step 3: Get Endpoint URLs and Keys

After deployment (15-30 minutes), get credentials:

```
# List endpoints
az ml online-endpoint list \
  --workspace-name $AML_WORKSPACE \
  --resource-group $RESOURCE_GROUP
```

Get keys from Azure AI Studio UI → Endpoints → Consume tab

□ Save to Configuration Sheet:

- ML Workspace: `$AML_WORKSPACE`
- Med42 Endpoint: `[Get from portal]`
- BioGPT Endpoint: `[Get from portal]`
- BiomedCLIP Endpoint: `[Get from portal]`
- Endpoint Keys: `[Get from portal]`

---

8. Compute Services Setup

8.1 Azure App Service

Step 1: Create App Service Plan

```
APP_SERVICE_PLAN="medical-chatbot-plan"

az appservice plan create \
  --name $APP_SERVICE_PLAN \
  --resource-group $RESOURCE_GROUP \
  --location $LOCATION \
  --is-linux \
  --sku S1 \
  --number-of-workers 2
```

Step 2: Create Web App

```
WEB_APP_NAME="medical-chatbot-api-$(openssl rand -hex 4)"

az webapp create \
  --name $WEB_APP_NAME \
  --resource-group $RESOURCE_GROUP \
  --plan $APP_SERVICE_PLAN \
  --runtime "PYTHON:3.11"

# Enable HTTPS only
az webapp update \
  --name $WEB_APP_NAME \
  --resource-group $RESOURCE_GROUP \
  --https-only true

# Enable Always On
az webapp config set \
  --name $WEB_APP_NAME \
  --resource-group $RESOURCE_GROUP \
  --always-on true
```

Step 3: Configure App Settings (Environment Variables)

```
az webapp config appsettings set \
  --name $WEB_APP_NAME \
  --resource-group $RESOURCE_GROUP \
  --settings \
    KEYVAULT_URL="https://$KEYVAULT_NAME.vault.azure.net/" \
    ENVIRONMENT="production"
```

Step 4: Enable Managed Identity

```
az webapp identity assign \
  --name $WEB_APP_NAME \
  --resource-group $RESOURCE_GROUP

# Get identity ID
APP_IDENTITY=$(az webapp identity show \
  --name $WEB_APP_NAME \
  --resource-group $RESOURCE_GROUP \
  --query principalId -o tsv)

# Grant Key Vault access
az keyvault set-policy \
  --name $KEYVAULT_NAME \
  --object-id $APP_IDENTITY \
  --secret-permissions get list
```

□ **Save to Configuration Sheet:**

- App Service Name: `$WEB_APP_NAME`
- App URL: `https://$WEB_APP_NAME.azurewebsites.net`
- App Service Plan: `$APP_SERVICE_PLAN`

8.2 Azure Functions

```
FUNCTION_APP_NAME="medical-chatbot-func-$(openssl rand -hex 4)"
FUNCTION_STORAGE="medfunc$(openssl rand -hex 4)"

# Create storage for Functions
az storage account create \
  --name $FUNCTION_STORAGE \
  --resource-group $RESOURCE_GROUP \
  --location $LOCATION \
  --sku Standard_LRS

# Create Function App
az functionapp create \
  --name $FUNCTION_APP_NAME \
  --resource-group $RESOURCE_GROUP \
  --storage-account $FUNCTION_STORAGE \
  --consumption-plan-location $LOCATION \
  --runtime python \
  --runtime-version 3.11 \
  --functions-version 4 \
  --os-type Linux

# Enable managed identity
az functionapp identity assign \
  --name $FUNCTION_APP_NAME \
  --resource-group $RESOURCE_GROUP

FUNC_IDENTITY=$(az functionapp identity show \
  --name $FUNCTION_APP_NAME \
  --resource-group $RESOURCE_GROUP \
  --query principalId -o tsv)

# Grant Key Vault access
az keyvault set-policy \
  --name $KEYVAULT_NAME \
  --object-id $FUNC_IDENTITY \
  --secret-permissions get list
```

 Save to Configuration Sheet:

- Function App Name: `$FUNCTION_APP_NAME`
- Function URL: `https://$FUNCTION_APP_NAME.azurewebsites.net`

---

9. Networking & API Gateway

9.1 Azure CDN

```
CDN_PROFILE="medical-chatbot-cdn"
CDN_ENDPOINT="medical-chatbot-$(openssl rand -hex 4)"

# Create CDN profile
az cdn profile create \
  --name $CDN_PROFILE \
  --resource-group $RESOURCE_GROUP \
  --sku Standard_Microsoft

# Create CDN endpoint
az cdn endpoint create \
  --name $CDN_ENDPOINT \
  --profile-name $CDN_PROFILE \
  --resource-group $RESOURCE_GROUP \
  --origin $WEB_APP_NAME.azurewebsites.net \
  --origin-host-header $WEB_APP_NAME.azurewebsites.net
```

□ Save to Configuration Sheet:

- CDN Profile: `$CDN_PROFILE`
- CDN Endpoint: `https://$CDN_ENDPOINT.azureedge.net`

9.2 Azure Application Gateway + WAF

This requires Azure Portal setup (complex CLI setup)

Portal Steps:

1. Go to Azure Portal → Create a resource → Application Gateway
2. Basics:
   - Name: `medical-chatbot-appgw`
   - Region: `Central India`
   - Tier: `WAF V2`
   - Enable autoscaling: Yes (2-10 instances)
3. Frontends:
   - Frontend IP: Public
   - Create new public IP: `appgw-public-ip`
4. Backends:
   - Add backend pool: `api-backend-pool`
   - Target: Your App Service URL
5. Configuration:
   - Add routing rule
   - Listener: HTTPS (upload SSL cert)
   - Backend target: `api-backend-pool`
6. WAF:
   - Mode: Prevention
   - Rule set: OWASP 3.2
7. Review + Create

□ Save to Configuration Sheet:

- Application Gateway: `medical-chatbot-appgw`
- Public IP: `[Get from portal]`
- Backend Pool: `api-backend-pool`

9.3 Azure API Management

```
APIM_NAME="medical-chatbot-apim-$(openssl rand -hex 4)"
APIM_EMAIL="admin@yourdomain.com"  # Change this

az apim create \
  --name $APIM_NAME \
  --resource-group $RESOURCE_GROUP \
  --location $LOCATION \
  --publisher-email $APIM_EMAIL \
  --publisher-name "Medical Chatbot" \
  --sku-name Developer \
  --enable-managed-identity true
```

Note: APIM creation takes 30-45 minutes.

Configure API after creation:

1. Go to Azure Portal → API Management → APIs
2. Add API → HTTP
3. Display name: `Medical Chatbot API`
4. Backend URL: `https://$WEB_APP_NAME.azurewebsites.net`
5. Add operations (GET, POST, etc.)
6. Configure policies (rate limiting, JWT validation)

□ Save to Configuration Sheet:

- APIM Name: `$APIM_NAME`
- Gateway URL: `https://$APIM_NAME.azure-api.net`

- Management URL: `https://$APIM_NAME.management.azure-api.net`

---

10. Authentication - Azure AD B2C

10.1 Create Azure AD B2C Tenant

Portal Steps (CLI not supported):

1. Go to Azure Portal → Create a resource → "Azure Active Directory B2C"
2. Create new B2C tenant:
    - Organization name: `MedicalChatbot`
    - Initial domain: `medicalchatbot.onmicrosoft.com`
    - Country: India
3. Click "Create"
4. Switch to new B2C tenant (top-right directory switcher)

10.2 Register Application

1. In B2C tenant → App registrations → New registration
2. Name: `Medical Chatbot Web App`
3. Redirect URI: `https://$WEB_APP_NAME.azurewebsites.net/auth/callback`
4. Click "Register"
5. Note: **Application (client) ID**
6. Certificates & secrets → New client secret
7. Note: **Secret value**

10.3 Create User Flows

1. B2C → User flows → New user flow
2. Select "Sign up and sign in"
3. Name: `B2C_1_signupsignin`
4. Identity providers: Email signup
5. User attributes: Email, Display Name, Phone (optional)
6. Create

10.4 Get Configuration Details

Save to Configuration Sheet:

- B2C Tenant: `medicalchatbot.onmicrosoft.com`
- Tenant ID: `[Get from portal]`
- Application ID: `[Get from step 10.2]`
- Client Secret: `[Get from step 10.2]`
- User Flow: `B2C_1_signupsignin`
- Authority: `https://medicalchatbot.b2clogin.com/medicalchatbot.onmicrosoft.com/B2C_1_signupsignin`

---

11. Monitoring & Logging

11.1 Application Insights

```
APP_INSIGHTS_NAME="medical-chatbot-insights"

# Create Log Analytics Workspace first
LOG_WORKSPACE="medical-chatbot-logs"

az monitor log-analytics workspace create \
  --resource-group $RESOURCE_GROUP \
  --workspace-name $LOG_WORKSPACE \
  --location $LOCATION

WORKSPACE_ID=$(az monitor log-analytics workspace show \
  --resource-group $RESOURCE_GROUP \
  --workspace-name $LOG_WORKSPACE \
  --query id -o tsv)

# Create Application Insights
az monitor app-insights component create \
  --app $APP_INSIGHTS_NAME \
  --location $LOCATION \
  --resource-group $RESOURCE_GROUP \
  --workspace $WORKSPACE_ID

# Get instrumentation key
INSTRUMENTATION_KEY=$(az monitor app-insights component show \
  --app $APP_INSIGHTS_NAME \
  --resource-group $RESOURCE_GROUP \
  --query instrumentationKey -o tsv)

CONNECTION_STRING=$(az monitor app-insights component show \
  --app $APP_INSIGHTS_NAME \
  --resource-group $RESOURCE_GROUP \
  --query connectionString -o tsv)

# Store in Key Vault
az keyvault secret set \
  --vault-name $KEYVAULT_NAME \
  --name "app-insights-key" \
  --value "$INSTRUMENTATION_KEY"

az keyvault secret set \
  --vault-name $KEYVAULT_NAME \
  --name "app-insights-connection-string" \
  --value "$CONNECTION_STRING"
```

Link to App Service

```
az webapp config appsettings set \
  --name $WEB_APP_NAME \
  --resource-group $RESOURCE_GROUP \
  --settings \
    APPLICATIONINSIGHTS_CONNECTION_STRING="$CONNECTION_STRING"
```

Save to Configuration Sheet:

- App Insights Name: `$APP_INSIGHTS_NAME`
- Instrumentation Key: `$INSTRUMENTATION_KEY`
- Connection String: `$CONNECTION_STRING`

11.2 Azure Monitor Alerts

```
# Create action group for notifications
az monitor action-group create \
  --name "medical-chatbot-alerts" \
  --resource-group $RESOURCE_GROUP \
  --short-name "MedAlert" \
  --email-receiver name=admin email=admin@yourdomain.com

# Create alert for high error rate
az monitor metrics alert create \
  --name "High Error Rate" \
  --resource-group $RESOURCE_GROUP \
  --scopes "/subscriptions/$(az account show --query id -o tsv)/resourceGroups/$RESOURCE_GROUP/providers/Microsoft.Web/sites/$WEB_API
  --condition "avg exceptions/count > 10" \
  --window-size 5m \
  --evaluation-frequency 1m \
  --action medical-chatbot-alerts
```

11.3 Communication Services (for SMS/Email)

```
COMM_SERVICE_NAME="medical-chatbot-comm"

az communication create \
  --name $COMM_SERVICE_NAME \
  --resource-group $RESOURCE_GROUP \
  --location global \
  --data-location UnitedStates

# Get connection string
COMM_CONNECTION_STRING=$(az communication list-key \
  --name $COMM_SERVICE_NAME \
  --resource-group $RESOURCE_GROUP \
  --query primaryConnectionString -o tsv)

az keyvault secret set \
  --vault-name $KEYVAULT_NAME \
  --name "communication-services-key" \
  --value "$COMM_CONNECTION_STRING"
```

□ **Save to Configuration Sheet:**

- Communication Service: `$COMM_SERVICE_NAME`
- Connection String: `$COMM_CONNECTION_STRING`

---

12. Testing & Validation

12.1 Test Storage Access

```
# Upload test file to Blob Storage
echo "Test prescription" > test.txt
az storage blob upload \
  --account-name $STORAGE_ACCOUNT \
  --container-name prescription-uploads \
  --name test.txt \
  --file test.txt \
  --account-key $STORAGE_KEY

# Verify
az storage blob list \
  --account-name $STORAGE_ACCOUNT \
  --container-name prescription-uploads \
  --account-key $STORAGE_KEY \
  --output table
```

12.2 Test Database Connection

```
# Install SQL command-line tool
# For macOS: brew install msodbcsql17 mssql-tools

# Connect to SQL Database
sqlcmd -S $SQL_SERVER.database.windows.net \
  -d medicalchatbot-users \
  -U $SQL_ADMIN \
  -P $SQL_PASSWORD \
  -Q "SELECT @@VERSION;"
```

12.3 Test OpenAI Deployment

```
# Using curl
curl -X POST \
  "$OPENAI_ENDPOINT/openai/deployments/gpt-4-deployment/chat/completions?api-version=2024-02-01" \
  -H "Content-Type: application/json" \
  -H "api-key: $OPENAI_KEY" \
  -d '{
    "messages": [
      {"role": "system", "content": "You are a helpful assistant."},
      {"role": "user", "content": "Say hello!"}
    ]
  }'
```

12.4 Test Key Vault Access

```
# Retrieve a secret
az keyvault secret show \
  --vault-name $KEYVAULT_NAME \
  --name "openai-api-key" \
  --query value -o tsv
```

12.5 Verify All Services are Running

```
# Check resource group resources
az resource list \
  --resource-group $RESOURCE_GROUP \
  --output table

# Check App Service status
az webapp show \
  --name $WEB_APP_NAME \
  --resource-group $RESOURCE_GROUP \
  --query state -o tsv

# Check Function App status
az functionapp show \
  --name $FUNCTION_APP_NAME \
  --resource-group $RESOURCE_GROUP \
  --query state -o tsv
```

---

13. Cost Optimization

13.1 Set Up Budget Alerts

```
# Create budget (example: $500/month)
az consumption budget create \
  --budget-name "medical-chatbot-monthly" \
  --amount 500 \
  --time-grain Monthly \
  --start-date "2025-11-01" \
  --end-date "2026-12-31" \
  --resource-group $RESOURCE_GROUP \
  --notifications \
    name=actual-80 \
    enabled=true \
    operator=GreaterThan \
    threshold=80 \
    contact-emails="admin@yourdomain.com"
```

13.2 Enable Auto-Shutdown for Dev Resources

```
# For App Service (scale down after hours)
az webapp config set \
  --name $WEB_APP_NAME \
  --resource-group $RESOURCE_GROUP \
  --auto-heal-enabled true
```

13.3 Use Reserved Instances (for production)

For long-term (1-3 years), purchase reserved capacity:

- App Service: Save up to 55%
- SQL Database: Save up to 80%
- Cosmos DB: Save up to 65%

Portal: Reservations → Purchase

---

14. Troubleshooting

14.1 Common Issues

Issue: Azure OpenAI quota exceeded

Solution:

```
# Request quota increase
# Go to Portal → Subscriptions → Usage + quotas → Search "OpenAI" → Request increase
```

Issue: App Service deployment fails

Solution:

```
# Check logs
az webapp log tail \
  --name $WEB_APP_NAME \
  --resource-group $RESOURCE_GROUP

# Restart app
az webapp restart \
  --name $WEB_APP_NAME \
  --resource-group $RESOURCE_GROUP
```

Issue: Key Vault access denied

Solution:

```
# Verify permissions
az keyvault show \
  --name $KEYVAULT_NAME \
  --query properties.accessPolicies

# Re-grant access
az keyvault set-policy \
  --name $KEYVAULT_NAME \
  --object-id $APP_IDENTITY \
  --secret-permissions get list
```

Issue: AI model deployment fails

Solution:

- Check region availability for specific models
- Verify compute quota in subscription
- Try different instance types
- Contact Azure Support for model access

14.2 Diagnostic Commands

```
# Check all resource states
az resource list \
  --resource-group $RESOURCE_GROUP \
  --query "[].{Name:name, Type:type, Location:location, State:properties.provisioningState}" \
  --output table

# View activity log
az monitor activity-log list \
  --resource-group $RESOURCE_GROUP \
  --offset 1h \
  --query "[].{Time:eventTimestamp, Operation:operationName.localizedValue, Status:status.localizedValue}" \
  --output table

# Check current costs
az consumption usage list \
  --start-date 2025-10-01 \
  --end-date 2025-10-15 \
  --query "[].{Date:usageStart, Cost:pretaxCost}" \
  --output table
```

14.3 Support Resources

- **Azure Documentation:** https://docs.microsoft.com/azure (https://docs.microsoft.com/azure)
- **Azure Status:** https://status.azure.com (https://status.azure.com)
- **Azure Support:** https://azure.microsoft.com/support (https://azure.microsoft.com/support)
- **Community Forums:** https://docs.microsoft.com/answers (https://docs.microsoft.com/answers)
- **Stack Overflow:** Tag `azure`

---

15. Next Steps

15.1 Post-Setup Checklist

- [ ] All services created and running
- [ ] All secrets stored in Key Vault
- [ ] Configuration sheet filled out
- [ ] Test connections verified
- [ ] Monitoring alerts configured
- [ ] Budget alerts set up
- [ ] Backup strategy implemented
- [ ] Documentation updated

15.2 Development Setup

1. Clone your project repository
2. Copy `.env.template` from Configuration Sheet
3. Fill in values from Key Vault
4. Install dependencies
5. Run local development server
6. Test API endpoints
7. Deploy to staging
8. Run integration tests
9. Deploy to production

15.3 Production Readiness

Before going live:

- [ ] SSL certificates configured
- [ ] Custom domain mapped
- [ ] WAF rules tested
- [ ] Rate limiting configured
- [ ] Data backup verified
- [ ] Disaster recovery tested
- [ ] Load testing completed
- [ ] Security audit passed
- [ ] Compliance checklist completed

Appendix A: Quick Reference Commands

Get All Resource Names

```
# Save this script as get-resources.sh
#!/bin/bash

echo "Resource Group: $RESOURCE_GROUP"
echo "Key Vault: $KEYVAULT_NAME"
echo "Storage Account: $STORAGE_ACCOUNT"
echo "SQL Server: $SQL_SERVER.database.windows.net"
echo "Cosmos DB: $COSMOS_ACCOUNT"
echo "OpenAI: $OPENAI_NAME"
echo "App Service: $WEB_APP_NAME.azurewebsites.net"
echo "Function App: $FUNCTION_APP_NAME.azurewebsites.net"
echo "AI Search: $AI_SEARCH_NAME"
echo "Redis: $REDIS_NAME"
```

Retrieve All Secrets

```
# List all secrets in Key Vault
az keyvault secret list \
  --vault-name $KEYVAULT_NAME \
  --query "[].{Name:name}" \
  --output table

# Get specific secret
az keyvault secret show \
  --vault-name $KEYVAULT_NAME \
  --name "openai-api-key" \
  --query value -o tsv
```

Clean Up (Delete All Resources)

```
# WARNING: This deletes everything!
az group delete \
  --name $RESOURCE_GROUP \
  --yes \
  --no-wait
```

Appendix B: Estimated Costs

Service	Tier	Monthly Cost (USD)
Azure OpenAI (GPT-4)	Standard	$50-100
Azure AI Foundry	GPU Compute	$100-200
Document Intelligence	S0	$10-20
AI Search	Standard S1	$250
App Service	S1 × 2	$150
Azure Functions	Consumption	$10-20
SQL Database	S2 × 2	$300
Cosmos DB	Autoscale	$25-50
Blob Storage	Standard	$20-50
Redis Cache	Basic C1	$17
Key Vault	Standard	$5
Application Insights	Pay-as-you-go	$20-50
API Management	Developer	$50
CDN	Standard	$10-20
Communication Services	Pay-as-you-go	$10-30
Total Estimated		**$1,027-1,327/month**

Note: Costs vary based on usage. Enable auto-scaling and monitoring to optimize.

Document Information

Version: 1.0

Last Updated: October 15, 2025

Author: Medical Chatbot Team

Contact: support@medicalchatbot.com

Next Review Date: November 15, 2025

End of Setup Guide

For configuration tracking, see: `Azure_Configuration_Master_Sheet.md`