CMAP

Azure Cloud Deployment – Deliverables (PROD)

**v1.0 as of 07.03.2025**

**Document Revision History**

Each time this document is modified, increment the version number appropriately and add a new row to the table below. In the Comments column, document the changes that were made in sufficient detail.

| **Date** | **Author** | **Version** | **Comments** |
| --- | --- | --- | --- |
| 07/03/2025 | Syneren Technologies Corp | 1.0 | Created |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[1. Introduction 1](#_Toc202373182)

[2. Deliverables 1](#_Toc202373183)

[3. Cloud Services Overview 2](#_Toc202373184)

[4. Frontend Deployment Package 3](#_Toc202373185)

[Creating ‘dist’ build for Vite-React 3](#_Toc202373186)

[5. Backend Deployment Package 3](#_Toc202373187)

[5a. Node.js Backend Using Azure App Service 3](#_Toc202373188)

[5b. Python Backend Using Azure App Service and Azure Container Registry (ACR) 4](#_Toc202373189)

[6. Database Setup:s 5](#_Toc202373190)

[Using Azure Cosmos DB with MongoDB API 5](#_Toc202373191)

[Exporting the Database 5](#_Toc202373192)

[7. Key Management: 6](#_Toc202373193)

[Using Azure Key Vault 6](#_Toc202373194)

[8. Final Notes and Best Practices: 7](#_Toc202373195)

# Introduction

This document covers creating deliverables and deployment packages of CMAP frontend and backend, Cosmos DB database.

# Deliverables

| **Component** | **Provided As** | **Notes** |
| --- | --- | --- |
| Frontend | dist.zip | Pre-built Vite static assets |
| Backend (Node.js) | node\_backend.zip | Includes package.json, excludes node\_modules |
| Backend (Python) | python\_backend.tar (Docker image) | No local setup needed |
| Database | db.zip | MongoDB collections as JSON/BSON |

# Cloud Services Overview

| **Component** | **Technology/Service** | **Notes** |
| --- | --- | --- |
| Frontend | Azure Static Web Apps | Hosts Vite static content |
| Backend (Node.js) | Azure App Service (Linux) | ZIP deployment, auto-installs dependencies |
| Backend (Python) | Azure App Service (Linux) + Azure Container Registry (ACR) | CI/CD with Docker build & deploy pipeline |
| Database | Azure Cosmos DB (MongoDB API) | Connection string managed as env variable |
| Document Storage | Azure Blob Storage | For storing application documents |
| Logging | Azure Application Insights | Capture custom logs, traces, and telemetry |
| Key Management | Azure Key Vault | Secure storage of keys, secrets, and connection strings |
| Authentication | App Registrations | Secured Entra ID login |
| Code Environment Changes | None | Changing the URI and other variables in the Code environment |

# Frontend Deployment Package

## Creating ‘dist’ build for Vite-React

**Steps:**

1. **Confirm the web application in dev is working**
   * ‘npm run dev’ is working as expected
   * Integration and Feature testing is done.
2. **Create Build**
   * **Enter the command**: ‘npm run build’
   * **Check for the folder:** ‘dist’
   * **Recent Files are present**: Check if ‘dist’ contains all the latest build files
   * **Local Environment Check**: Test the dist in local environment
3. **Export it as zip**
   * Export the ‘dist’ folder as a zip.

# Backend Deployment Package

## 5a. Node.js Backend Using Azure App Service

**Steps:**

1. **Confirm the web application in dev is working**
   * ‘npm start’ is working as expected
   * Integration and Feature testing is done.
2. **Create Build**
   * **Copy and Paste**: Copy Paste the backend dev code in prod repo/local folder removing all the except the node\_modules
3. **Export it as zip**
   * Export the ‘backend’ folder as a zip.

## 5b. Python Backend Using Azure App Service and Azure Container Registry (ACR)

**Steps:**

1. **Login to Azure CLI**
   * az login
2. **Create Resource Group (if not created already)**
   * az group create --name cmap-python-backend-rg --location eastus
   * Free tier is available in eastus, centralus, and a few other regions.
3. **Create Azure Container Registry (Free)**

az acr create \

--resource-group cmap-python-backend-rg \

--name cmapacrbackend \

--sku Basic \

--admin-enabled true

* + --sku Basic is the smallest, and free up to a few GB/month.

1. **Build the Image Remotely (no local docker)**
   * az acr build --registry cmapacrbackend --image python-ocr:latest
   * Run from the folder where your Dockerfile is.
2. **Create a blob container (if not already)**

az storage container create \

--account-name <your-storage-account-name> \

--name <your-container-name> \

--public-access blob

1. **Export the image from ACR to Blob Storage**

az acr export \

--name cmapacrbackend \

--image python-ocr:latest \

--storage-account <your-storage-account-name> \

--storage-container <your-container-name> \

--resource-group <your-resource-group> \

--output-path python-ocr-exported.tar.gz \

--force

* + This will export the image as a .tar.gz into your blob storage.

1. **Generate a download link (SAS URL)**

az storage blob generate-sas \

--account-name <your-storage-account-name> \

--container-name <your-container-name> \

--name python-ocr-exported.tar.gz \

--permissions r \

--expiry 2025-12-31T23:59:00Z \

--output tsv

* + Then prepend with the blob URL:

https://<your-storage-account-name>.blob.core.windows.net/<your-container-name>/python-ocr-exported.tar.gz?<SAS\_TOKEN>

* + Use that link to share or download.

# Database Setup:s

## Using Azure Cosmos DB with MongoDB API

### Exporting the Database

#### Option A: Via MongoDB Compass (For Initial Setup)

**Steps:**

1. Connect to the DB using MongoDB Compas
2. Import User
   * Click on user collection and select export all as json.
3. Import Systems
   * Click on systems collection and select export all as json.
4. Continue or Exit
   * You can continue doing same for all collections if you need and zip them together.
   * Download MongoDB Tools to use this command to export the whole new db in bson if it works:

mongodump --db cmap your\_db\_dump\_parent\_folder

#### Option B: Via CMAP Web Application

**Steps:**

1. Login to CMAP via Global Admin
2. Click on Import/Export DB button
   * Click on Export tab
   * Click on ‘Load Collections’
   * Choose some or select all
   * Keep the exported db name as cmap or your choice
   * Hit Export.
3. A zip db file will be downloaded.

# Key Management:

## Using Azure Key Vault

Azure Key Vault securely stores secrets, keys, and certificates used by your applications.

**Steps:**

1. **Create Azure Key Vault**
   * Azure Portal → **Create a resource** → Search **Key Vault**
   * Enter name, subscription, resource group, region
2. **Add Secrets / Keys**
   * In your Key Vault → **Secrets** → **+ Generate/Import**
   * Add secrets such as:
     + Cosmos DB connection string
     + Storage account keys or SAS tokens
     + Any other sensitive credentials
3. **Access Key Vault from App Services**
   * Grant your App Service managed identity **Get** permissions on the Key Vault secrets via Access Policies
   * In your application, use Azure SDK or environment variables referencing Key Vault references (e.g., via @Microsoft.KeyVault syntax in App Settings)
4. **Use Key Vault references in App Service**
   * In App Service Configuration, set values like

@Microsoft.KeyVault(SecretUri=https://<your-vault-name>.vault.azure.net/secrets/<secret-name>/<secret-version>)

* + This allows your app to retrieve secrets securely at runtime without hardcoding them

# Final Notes and Best Practices:

* **Managed Identities**: Use **Azure Managed Identity** to grant access to Key Vault and other services without requiring credentials in your code.
* **Logging Enablement**: Always enable **App Service diagnostics logging** alongside Application Insights for holistic observability.
* **Scalability**: Azure services like Static Web Apps, Cosmos DB, and Blob Storage scale seamlessly, supporting future growth in users and data.