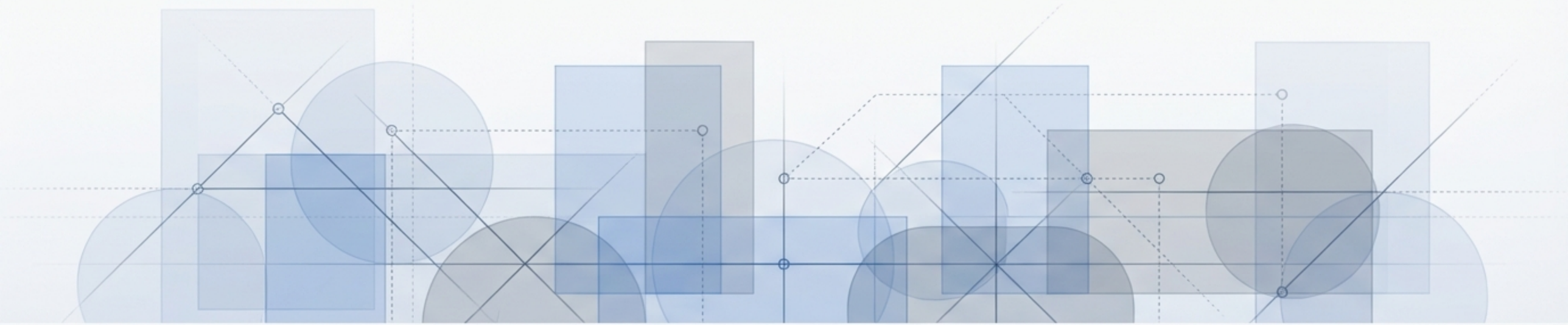
Your Implementation Guide

Deploying the Corporate Intelligence Hub



A Step-by-Step Guide for a Successful Go-Live

Your Implementation Journey

Phase 1: Laying the Foundation

Deploy cloud infrastructure and prepare the database.

Phase 2: Deploying the Core Engine

Install the backend API and data ingestion worker.

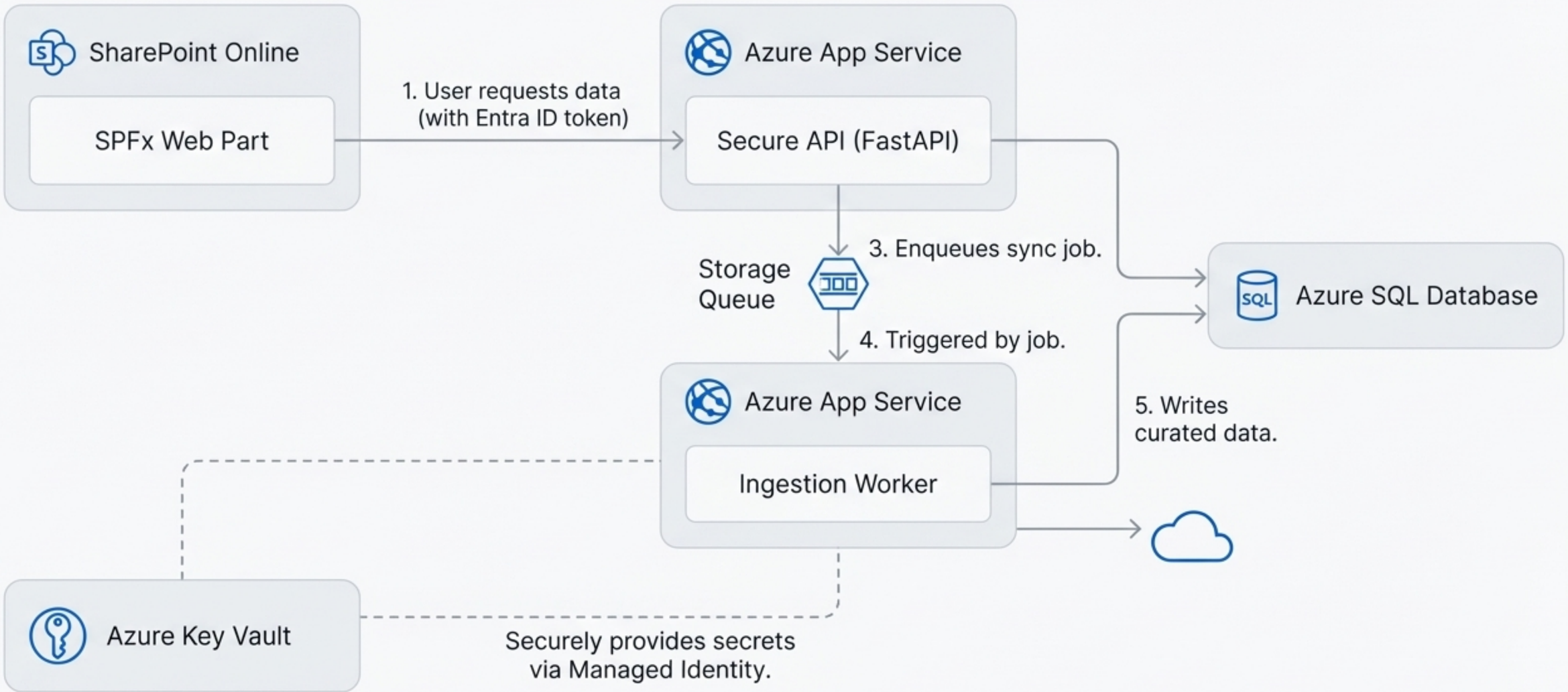
Phase 3: Activating the User Experience

Configure and deploy the SharePoint user interface.

Phase 4: Validation and Go-Live

Perform end-to-end testing and confirm system readiness.

Understanding the Solution Architecture



Inside Your Deployment Package: CIH-Repo-NextStep-v2.zip

This consolidated bundle contains all the code and infrastructure templates you'll need for the entire deployment.

- **`infra/bicep/`:** Azure Infrastructure as Code. Creates and configures all necessary cloud resources.
- **`app/`:** The Core API. The secure FastAPI backend that serves data to the SharePoint UI.
- **`Worker`:** The Data Ingestion Worker. Pulls data from Concur APIs and populates the SQL database.
- **`sql/`:** Database Schema. The DDL scripts to create the required tables in your Azure SQL database.
- **`spfx/cih-spfx/`:** SharePoint UI. The source code for the SharePoint Framework web parts.

Prerequisites for a Smooth Deployment

Please ensure you have the following in place before you begin.

Required Tools & Software

Azure CLI: Installed locally, or access to **Azure Cloud Shell** in the portal.

Code Editor: A tool like VS Code for editing parameter files.

Node.js & Gulp: For the SharePoint SPFx packaging step (can be done by a developer).

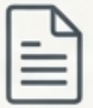
SQL Management Tool: Azure Data Studio, SSMS, or the Azure Portal Query Editor.

Required Permissions & Information

Azure: **Subscription Owner** or a role with permissions to create resource groups, deployments, and role assignments.

SharePoint: **SharePoint Tenant Administrator** role to upload the application package to the App Catalog.

Entra ID: Permissions to grant admin consent for API permissions.



Information: Your Azure **Tenant ID**.

Phase 1: Laying the Foundation

In this section, we will provision the core cloud infrastructure and database schema.

Step 1: Deploy Azure Infrastructure via Bicep

Purpose

This script automates the creation of all required Azure resources (App Service, Key Vault, Storage) and correctly configures their security, including Managed Identity.

Key Actions

1. **Edit Parameters**: Open `infra/bicep/main.parameters.json` and set the values for `prefix`, `resourceGroupName`, and `storageAccountName`.
2. **Run Deployment**: Execute the following command from your terminal:

```
az deployment sub create \
  --location australiaeast \
  --template-file infra/bicep/main.bicep \
  --parameters @infra/bicep/main.parameters.json
```

Best Practice Callout: The `storageAccountName` must be globally unique and contain only lowercase letters and numbers.

Verification Checkpoint

After the deployment succeeds, navigate to the newly created Key Vault in the Azure Portal and manually create the following secrets:

- `concur-client-secret`
- `concur-refresh-token`
- `sql-password`
- `graph-client-secret` (optional)

Step 2: Prepare the Database Schema

Purpose

To create the necessary tables in your SQL database that the worker will write to and the API will read from.

Key Actions

1. **Provision DB**: If you don't have one, create an Azure SQL Database. The Bicep script does *not* create the database itself.
2. **Connect to DB**: Use your preferred tool (Azure Data Studio, SSMS, or the Azure Portal Query Editor) to connect to the database.
3. **Execute Script**: Open and run the DDL script(s) located in the `sql/` folder of the deployment package.

Verification Checkpoint

After the script runs, verify that the following key tables have been created in your database:

✅ `employee`
✅ `credit card fact table`
✅ `expense report fact table`
✅ `import run history table`

Phase 2: Deploying the Core Engine

With the infrastructure in place, we will now deploy the backend API and data ingestion services.

Step 3: Deploy the API and Ingestion Worker

Purpose

To deploy the application code to the App Services provisioned in Phase 1. The API serves the user dashboard, while the Worker handles background data processing.

Key Actions

1. **Package Code**: Create zip files of the `app/` directory (for the API) and the `Worker` directory.
2. **Deploy via Zip**: Use the Azure CLI or portal tools to "zip deploy" each package to its respective App Service. The worker should be deployed to a separate App Service for best practice.
3. **Configure App Settings**: In the Azure Portal for the API App Service, navigate to Configuration → Application settings and add the following:

```
AAD_TENANT_ID: Your Azure Tenant ID.
AAD_API_AUDIENCE: The Application ID URI of your API's App Registration (e.g., api://<your-api-app-id>).
REQUIRED_AAD_GROUP_IDS (Optional): Comma-separated Group Object IDs for RBAC.
```

✅ Verification Checkpoint

In the App Service, go to Diagnose and solve problems and search for Key Vault configuration. A message indicating that all Key Vault references were successfully resolved is your confirmation that security is working.

How Your Application is Secured

We use modern, best-practice patterns to ensure your data and services are protected.

Passwordless with Managed Identity

The Bicep script automatically configures the App Services with a Managed Identity and grants it `Key Vault Secrets User` access. The applications access secrets without ever needing a stored password or connection string.

No Plain-Text Secrets

Application Settings use Key Vault references (e.g., `@{Microsoft.KeyVault(SecretUri=...)}`). This means sensitive values are never stored in plain text in the application's configuration.

Production-Grade Token Validation

The API enforces that every request has a valid Entra ID bearer token. It performs full JWKS validation to ensure the token is authentic and issued for your application, preventing anonymous access.

Phase 3: Activating the User Experience

Now we will deploy and configure the SharePoint web parts that users will interact with.

Step 4: Deploy the SharePoint User Interface

Purpose

To package and deploy the SharePoint Framework (SPFx) web parts, making the dashboard and search functionality available on your SharePoint site.

The 'Easy Path' for Deployment

1. **Generate Project:** Have a developer run `yo @microsoft/sharepoint` to generate a standard SPFx solution.
2. **Copy Source:** Replace the newly created `src/` folder with the contents of the `spfx/cih-spfx/src` folder from your deployment package.
3. **Build & Package:** Run the following commands in the project folder:

```
npm install
gulp bundle --ship
gulp package-solution --ship
```

4. **Deploy to App Catalog:** A SharePoint Tenant Admin must upload the generated `.sppkg` file (from the `sharepoint/solution` folder) to the tenant's App Catalog and approve the API permissions request.
5. **Add to Page:** Add the "Dashboard" and "Employee Search" web parts to a SharePoint page.
6. **Configure Properties:** In the web part property pane, set the API Base URL (e.g., `https://<your-app>.azurewebsites.net`) and API Resource (your `AAD_API_AUDIENCE` value).

Phase 4: Validation and Go-Live

The final step is to perform an end-to-end test to ensure all components are working together correctly.

Step 5: Perform End-to-End Validation

Purpose

To run a final, comprehensive check of the entire system to confirm a successful deployment.

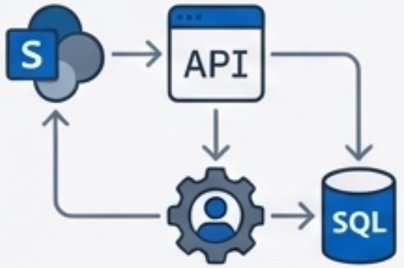
Automated Checks (Recommended)

- Use the CIH-Customer-Validation-Bundle.zip provided as a supporting artifact.
- Run the PowerShell scripts from an admin workstation to automatically verify: API reachability, auth configuration, SQL connectivity, and the existence of queues and Key Vault secrets.

Manual Test Plan

1. **API Health**: Use a REST client with a valid bearer token to call the /health endpoint on your API. You should receive a success message.
2. **Web Part Functionality**: On your SharePoint page, confirm the 'Employee Search' web part returns results when you type in a query.
3. **Data Ingestion**: Manually trigger a data ingestion job.
4. **Dashboard Data**: After the ingestion job completes, confirm that the Dashboard web part populates with data from the Report status and card status endpoints.

Congratulations: Your Corporate Intelligence Hub is Live



You Have Successfully Deployed

- A secure, token-authenticated API backend.
- An automated, queue-based data ingestion worker.
- A fully integrated SharePoint user experience.

Immediate Next Steps

1. Schedule and run your first full data synchronization from Concur.
2. Communicate the availability of the new dashboard to your target business users.
3. Review the default monitoring and alerting rules in Application Insights to ensure operational visibility.