Azure Architecture (Services + CI/CD) – Job Portal

1. Application Stack Overview

Frontend

- **React** (located in `client/` folder)
- Builds into static files served to users
- Uses Axios to call backend APIs
- Rendered as a Single Page Application (SPA)

Backend

- **Node.js + Express** (located in `server/` folder)
- Exposes REST API endpoints:
  - `GET /api/jobs`
  - `GET /api/jobs/:id`
  - `GET /api/health`
- Handles filtering, pagination, and future business logic

Database (future integration)

- Will use **Azure SQL Database** or **Cosmos DB**
- Replaces current mock data
- Stores jobs, users, favorites, etc.

API Communication

- Frontend → Axios → Backend API
- Backend returns JSON responses consumed by React

Health Endpoint

- `GET /api/health` returns `{ status: "OK" }`
- Used by Azure Application Insights to check uptime and system health

---

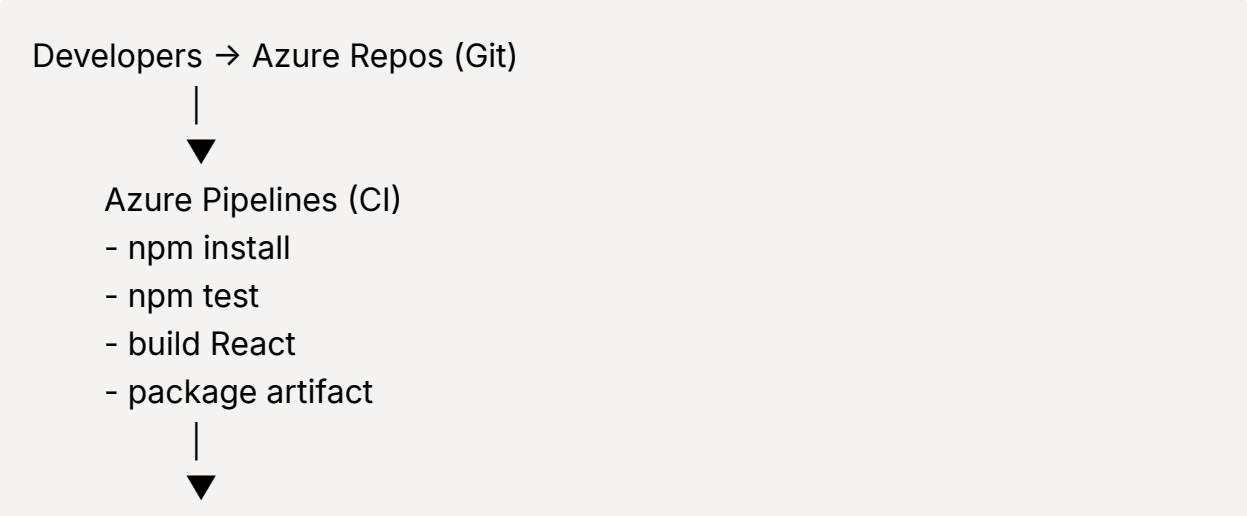
2. Azure Services Overview

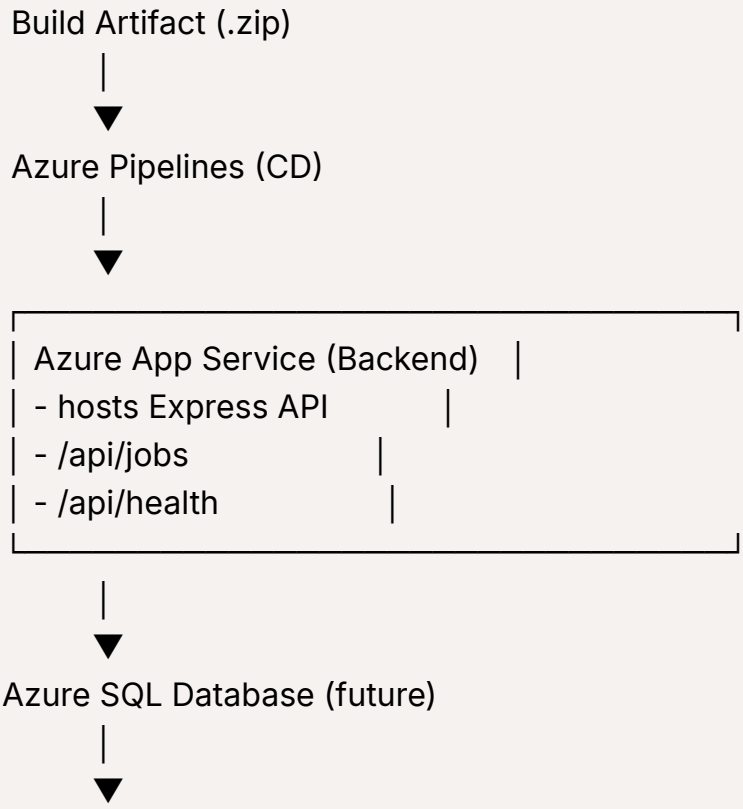
Goal	Azure Service	Why It Matters
Host your Node backend	**Azure App Service (Web App)**	Runs your Express server 24/7, handles traffic, scaling, SSL, etc.
Host your React frontend	**Azure Static Web Apps** *or* App Service	Hosts the built React frontend, serves static files globally.
Store job listings/data	**Azure SQL Database** (or Cosmos DB)	Cloud database for scalable data persistence.
Automate builds & deploys	**Azure DevOps Pipelines**	CI/CD engine for building, testing, and deploying automatically.
Manage code & branches	**Azure Repos**	Git hosting with PRs, branches, and version control.
Monitor performance	**Application Insights**	Tracks health, errors, logs, and response times; runs availability tests.

This satisfies the assignment requirement of **using at least 3 Azure services**.

---

3. Azure Architecture Diagram (High-Level)





4. CI/CD Pipeline Definition

CI (Continuous Integration)

Triggered on every push to the repository (e.g., `dev` or feature branches).

Azure Pipelines will automatically:

1. Install backend dependencies

```
npm install
```

2. Install frontend dependencies

```
cd client && npm install
```

3. Run backend tests

```
npm test
```

4. Build the frontend

```
cd client && npm run build
```

5. Package the application into a deployable artifact

Outcome: Stops bugs early, ensures code quality before deployment.

CD (Continuous Deployment)

Triggered after a successful CI build.

Azure Pipelines:

1. Uses a **Service Connection** (secure Azure credential link)
2. Publishes backend to **Azure App Service**
3. Publishes frontend build to **Static Web Apps** or App Service
4. Updates environment variables automatically
5. Can deploy to:
    - **Development** environment (auto)
    - **Production** environment (with approval)

Outcome: Code goes live automatically without manual deployment.

---

5. Monitoring & Health Tracking

Application Insights monitors your live app

It automatically collects:

- Request success/failure
- Response time
- CPU/memory usage
- Exceptions
- Dependency failures

Availability Tests

You configure Application Insights to call:

```
https://<yourapp>.azurewebsites.net/api/health
```