# Full Azure Migration Checklist - RotoWrite

Complete migration checklist for moving RotoWrite from Vercel/Supabase to Azure App Service with Azure Database for PostgreSQL, replacing Supabase Auth with Azure AD B2C or NextAuth.js.

---

## Phase 1: Azure Infrastructure Setup

### 1.1 Azure Database for PostgreSQL Setup

- [ ] Create Azure Database for PostgreSQL Flexible Server
- [ ] Choose appropriate SKU (recommend: Standard_B2s or higher for production)
- [ ] Configure firewall rules to allow Azure services
- [ ] Enable pgvector extension: CREATE EXTENSION IF NOT EXISTS vector;
- [ ] Create database user and set permissions
- [ ] Configure backup retention (7-35 days recommended)
- [ ] Set up connection string and store securely

### 1.2 Azure App Service Setup

- [ ] Create Azure App Service Plan (Linux, recommended: Basic B1 or higher)
- [ ] Create Web App resource
- [ ] Configure Node.js runtime (v20 LTS)
- [ ] Set up deployment from GitHub/GitLab
- [ ] Configure custom domain (optional)
- [ ] Enable HTTPS/SSL certificate
- [ ] Configure Application Insights for monitoring

### 1.3 Azure AD B2C Setup (Alternative: NextAuth.js with Azure AD)

- [ ] Create Azure AD B2C tenant (or use existing Azure AD)
- [ ] Register application in Azure AD
- [ ] Configure redirect URIs:
  - https://your-app.azurewebsites.net/api/auth/callback/azure
  - https://your-app.azurewebsites.net/api/auth/callback
- [ ] Create client secret
- [ ] Configure API permissions (User.Read, email, profile, openid)
- [ ] Set up user flows (sign-in, sign-up, password reset)
- [ ] Test authentication flow in Azure portal

---

## Phase 2: Database Migration

### 2.1 Schema Migration

- [ ] Export current Supabase schema using pg_dump --schema-only
- [ ] Remove Supabase-specific references:
  - Remove REFERENCES auth.users(id) from users table

  - Change users.id from UUID to VARCHAR or keep UUID (decide on ID strategy)

- [ ] Update schema to work without Supabase Auth:
  - Modify users table to be standalone (no auth.users reference)
  - Keep all other tables unchanged
- [ ] Create migration scripts for Azure PostgreSQL
- [ ] Test schema creation on Azure PostgreSQL

### 2.2 Data Migration

- [ ] Export data from Supabase: pg_dump --data-only
- [ ] Map Supabase auth.users data to new users table structure
- [ ] Create data transformation script:
  - Map auth.users.id → users.id
  - Map auth.users.email → users.email
  - Preserve user metadata (full_name, avatar_url, role)
- [ ] Export all application tables (writer_models, projects, briefs, etc.)
- [ ] Import data to Azure PostgreSQL
- [ ] Verify data integrity:
  - Count records in each table
  - Check foreign key relationships
  - Verify vector embeddings are intact

### 2.3 Row Level Security Migration

- [ ] Review all RLS policies in supabase/migrations/00002_row_level_security.sql
- [ ] Convert RLS policies to application-level authorization:
  - Update lib/auth.ts functions
  - Add authorization checks in API routes
  - Implement middleware-based route protection
- [ ] Test authorization logic thoroughly

---

## Phase 3: Authentication System Replacement

### 3.1 Choose Authentication Solution

#### Option A: Azure AD B2C (Recommended for enterprise)

- [ ] Install @azure/msal-browser and @azure/msal-node
- [ ] Create MSAL configuration
- [ ] Implement sign-in flow
- [ ] Implement sign-out flow
- [ ] Handle token refresh

#### Option B: NextAuth.js with Azure AD Provider (Easier migration)

- [ ] Install next-auth package

- [ ] Configure Azure AD provider
- [ ] Set up NextAuth API route: /app/api/auth/[...nextauth]/route.ts
- [ ] Configure session strategy (JWT recommended)

### 3.2 Update Authentication Code

- [ ] Replace lib/supabase/client.ts:
    - Remove Supabase client creation
    - Create new auth client (MSAL or NextAuth)
- [ ] Replace lib/supabase/server.ts:
    - Remove Supabase server client
    - Create server-side auth helper
- [ ] Update lib/auth.ts:
    - Replace getCurrentUser() to use new auth system
    - Update isAdmin() function
    - Update checkPermission() function
- [ ] Update lib/auth-microsoft.ts:
    - Replace Supabase OAuth with direct Azure AD flow
    - Or remove if using NextAuth.js (handles it automatically)

### 3.3 Update Middleware

- [ ] Update middleware.ts:
    - Remove Supabase session refresh logic
    - Add new auth session validation
    - Update protected route logic
- [ ] Update lib/supabase/middleware.ts or create new auth middleware
- [ ] Test middleware on all routes

### 3.4 Update API Routes (26 files to update)

- [ ] Update all API routes in app/api/:
    - Replace createClient() from Supabase
    - Replace supabase.auth.getUser() with new auth check
    - Replace supabase.from() queries with direct PostgreSQL queries
- [ ] Create PostgreSQL connection helper: lib/db.ts
- [ ] Update authentication checks in:
    - app/api/generate/route.ts
    - app/api/admin/**/*.ts (all admin routes)
    - app/api/auth/**/*.ts (auth routes)
    - app/api/research/**/*.ts
    - app/api/seo/**/*.ts
    - app/api/writer-models/**/*.ts

    - All other API routes

### 3.5 Update Client Components

- [ ] Update login page: app/login/page.tsx
- [ ] Update auth callback: app/api/auth/callback/route.ts
- [ ] Update sign-out: app/api/auth/signout/route.ts
- [ ] Update components using Supabase client:
    - Search for createClient() usage in components
    - Replace with new auth client

---

## Phase 4: Database Access Layer

### 4.1 Create Database Connection

- [ ] Install PostgreSQL client: pg or postgres package
- [ ] Create lib/db.ts:
    - Connection pool setup
    - Query helper functions
    - Transaction support
- [ ] Configure connection pooling
- [ ] Set up environment variables for database connection

### 4.2 Create Database Query Helpers

- [ ] Create query helpers for each table:
    - lib/db/users.ts - User queries
    - lib/db/projects.ts - Project queries
    - lib/db/writer-models.ts - Writer model queries
    - lib/db/briefs.ts - Brief queries
    - lib/db/training-content.ts - Training content queries
- [ ] Implement vector similarity search function
- [ ] Add error handling and logging

### 4.3 Update All Database Queries

- [ ] Replace all supabase.from() calls with direct SQL queries
- [ ] Update components using Supabase queries
- [ ] Test all database operations

---

## Phase 5: Environment Variables & Configuration

### 5.1 Update Environment Variables

- [ ] Remove Supabase variables:
    - NEXT_PUBLIC_SUPABASE_URL

  - NEXT_PUBLIC_SUPABASE_ANON_KEY
  - SUPABASE_SERVICE_ROLE_KEY
- [ ] Add Azure Database variables:
  - DATABASE_URL (connection string)
  - DATABASE_HOST
  - DATABASE_PORT
  - DATABASE_NAME
  - DATABASE_USER
  - DATABASE_PASSWORD
- [ ] Add Azure AD variables:
  - AZURE_AD_CLIENT_ID
  - AZURE_AD_CLIENT_SECRET
  - AZURE_AD_TENANT_ID
  - AZURE_AD_REDIRECT_URI
- [ ] Keep existing AI service keys (Grok, OpenAI, Tavily)
- [ ] Update NEXT_PUBLIC_APP_URL to Azure domain

### 5.2 Configure Azure App Service

- [ ] Add all environment variables in Azure Portal
- [ ] Configure Application Settings
- [ ] Set up connection strings
- [ ] Configure app settings for production

---

## Phase 6: Deployment & Testing

### 6.1 Build Configuration

- [ ] Update next.config.ts if needed for Azure deployment
- [ ] Create web.config for Azure App Service (if needed)
- [ ] Test build locally: npm run build
- [ ] Verify build output

### 6.2 Deployment

- [ ] Connect GitHub repository to Azure App Service
- [ ] Configure deployment branch (main/master)
- [ ] Set up build command: npm run build
- [ ] Configure start command: npm start
- [ ] Deploy initial version
- [ ] Monitor deployment logs

### 6.3 Testing Checklist

- [ ] Test user registration/sign-up
- [ ] Test Microsoft SSO login
- [ ] Test user logout

- [ ] Test protected routes (dashboard, admin)
- [ ] Test API endpoints:
    - Content generation
    - Writer model training
    - Project creation
    - Brief creation
    - SEO analysis
    - Research features
- [ ] Test database operations:
    - Create/read/update/delete operations
    - Vector similarity search
    - Foreign key relationships
- [ ] Test admin functions
- [ ] Test user permissions/roles

---

## Phase 7: Monitoring & Optimization

### 7.1 Monitoring Setup

- [ ] Configure Application Insights
- [ ] Set up alerts for errors
- [ ] Monitor database performance
- [ ] Track API response times
- [ ] Monitor authentication failures

### 7.2 Performance Optimization

- [ ] Configure database connection pooling
- [ ] Set up Redis cache (optional, for session storage)
- [ ] Optimize database queries
- [ ] Configure CDN for static assets (Azure CDN)
- [ ] Set up database indexes (verify existing ones)

### 7.3 Security Hardening

- [ ] Enable HTTPS only
- [ ] Configure CORS properly
- [ ] Review and update security headers
- [ ] Set up database firewall rules
- [ ] Rotate secrets and keys
- [ ] Enable Azure AD conditional access policies

---

## Phase 8: Documentation & Rollback Plan

### 8.1 Documentation

- [ ] Update DEPLOYMENT.md with Azure instructions

- [ ] Document new authentication flow
- [ ] Update environment variable documentation
- [ ] Create runbook for common issues
- [ ] Document database connection details

### 8.2 Rollback Plan

- [ ] Keep Supabase instance running during migration
- [ ] Document rollback steps
- [ ] Test rollback procedure
- [ ] Set up database backup before migration
- [ ] Keep Vercel deployment as backup

---

## Estimated Timeline

- **Phase 1**: 2-3 days (Infrastructure setup)
- **Phase 2**: 3-5 days (Database migration)
- **Phase 3**: 5-7 days (Authentication replacement)
- **Phase 4**: 3-4 days (Database access layer)
- **Phase 5**: 1 day (Configuration)
- **Phase 6**: 3-5 days (Deployment & testing)
- **Phase 7**: 2-3 days (Monitoring & optimization)
- **Phase 8**: 1 day (Documentation)

**Total Estimated Time**: 20-30 days

---

## Critical Dependencies

1. **Database Migration**: Must be completed before authentication changes
2. **Authentication**: Must be working before API routes can be updated
3. **Database Access Layer**: Required before updating components
4. **Testing**: Should be done incrementally, not all at once

---

## Risk Mitigation

- Run migration in staging environment first
- Keep Supabase running in parallel during transition
- Implement feature flags for gradual rollout
- Have rollback plan ready
- Test thoroughly at each phase before proceeding

---

## Task Dependencies

1. **Azure Infrastructure Setup** → Database Schema Migration
2. **Database Schema Migration** → Database Data Migration
3. **Azure Infrastructure Setup** → Authentication System Replacement
4. **Authentication System Replacement** → Update Authentication Code

5. **Database Data Migration** → Create Database Access Layer
6. **Update Authentication Code** → Update API Routes
7. **Create Database Access Layer** → Update API Routes
8. **Update Authentication Code** → Update Client Components
9. **Create Database Access Layer** → Update Client Components
10. **Update API Routes** → Deploy and Test
11. **Update Client Components** → Deploy and Test
12. **Configure Environment** → Deploy and Test
13. **Deploy and Test** → Monitoring Setup
14. **Deploy and Test** → Documentation