**Get up and running with the Fabrikam sample app**

1. In Azure, create App Insights instance, RG, etc.
2. Connect OMS to App Insights. Settings -> Data -> Application Insights. Pick appropriate subscription and instance.
3. Get the Fabrikam VS 2015 source from http://fabrikam.codeplex.com/SourceControl/latest
   1. Open in VS 2017. Update Nugets, fix any issues.
4. Create an Azure SQL DB database: FabrikamFiber
   1. For local->cloud debugging, add client IP to server firewall.
   2. Optionally, enable Auditing & Threat Detection.
   3. Optionally, configure Automatic tuning.
   4. Configure Diagnostics including to Log Analytics (OMS) workspace.
5. Update web.config and replace the connection strings set to LocalDb with the appropriate Azure SQL DB connection string.
   1. Server=tcp:pz17-fab-sql.database.windows.net,1433;Initial Catalog=FabrikamFiber;Persist Security Info=False;User ID=fabrikam;Password=F@brikam2017;MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=True;Connection Timeout=30;
6. Debug the solution in Visual Studio to ensure table creation in database = app is now minimally functional.
   1. Note: you may need to install IIS Express or IIS. You may need to repair existing IIS Express.
   2. Note: make sure the database user account you specify in the connection string has create table permission. Maybe add to db\_owner for demo.
7. Create an ASP in the same RG as the Azure SQL DB. Choose to also create an App Insights instance in the same RG.
   1. After creation, confirm that the web app is already configured for this App Insights instance but shows as offline since we have not yet published the app.
8. In Visual Studio, select the web app. Project menu -> Add Application Insights Telemetry
   1. Sign in as needed
   2. Select the existing (just created) AI resource
   3. Click Register
   4. This is build-time AI configuration
   5. Enable System.Diagnostics.Trace collection as prompted
   6. Verify App Insights 100% configured
   7. Verify App Insights added to /Views/Shared/\_Layout.cshtml for browser telemetry (also see https://docs.microsoft.com/azure/application-insights/app-insights-javascript)
   8. Verify ApplicationInsights.config file exists in solution. Verify build action/copy correct. Open and show instrumentation key; point out this would need to be changed for different AppI instances and that it could be moved to code if it should be changed programmatically for different environments.
9. Enable perf counters in IIS Express in the App Insights config file if IIS Express will be used for local debugging.
10. Debug the app locally and explore the Application Insights functionality in Visual Studio.
11. Include the snapshot debugger in the app. See https://docs.microsoft.com/azure/application-insights/app-insights-snapshot-debugger
    1. Requires TelemetryClient.TrackException(ex)
    2. Requires role in App Insights - will be prompted when trying to access snapshot in portal
    3. Requires Visual Studio Enterprise
12. Deploy the app to the previously created ASP. Verify functionality.
13. Create an Azure dashboard for this RG. Pin App Insights charts to it.
14. Connect App Insights instance to VSTS project so that problems can quickly have work items created for them.