# Microsoft Cloud Adoption Plan (Phase 1)

## Define Strategy

### Motivations and drivers

We want to have a more flexible IT setup, and be able to create and manage resources ourselves.

We want price transparency, and be able to take IT decisions based on the actual cost.

We want to be able to run anywhere. (Plug and play)

Innovate and modernize current application (Excel), to fit into a cloud setup.

### Business outcomes

What are the expected business outcomes from adopting the cloud?

#### High Priority:

Agility outcomes: Time-to-market

Scalability

#### Mid Priority:

Agility outcomes: Cycle time is a Six Sigma term that refers to the duration from the start to finish of a function.

Performance outcomes:

Reliability (Trade collectors uptime)

Performance (Be able to schedule large data integrations and calculations outside business peak hours).

### Business justification

We have a full responsibility and availability of our IT resources.

### First adoption project

List of excel sheet based on data from Azure SQL databases and output from R code.

### Key stakeholders

*Kell Pradsgaard Nielsen, Simon Brejnebjerg Jensen, Lars Normann Lund, Siamak Shamaee, Rami Jibreel, Kasper Kørup Trosborg and Gustav Sigen Ploug-Sørensen.*

### Cloud Strategy Team

Kell Pradsgaard Nielsen, Simon Brejnebjerg Jensen and Lars Normann Lund

## Plan

### Digital estate

|  |  |  |  |
| --- | --- | --- | --- |
| **Application/WorkloadBusiness** | **Business Unit** | **Priority (high, mid, low)** | **Proposed Rationalization** |
| Trayport MarketTrades | Energy Commodity | high | Rearchitect |
| R code running from edanalytics02 | MarketAnalytic | high | Rehost/Refactor |
| PointConnect | MarketAnalytic | high | Rebuild |
| MKOnline | MarketAnalytic | high | Rebuild |

# Azure Roadmap

## Database structure

The databases are all Azure SQL databases and they are all in the same SQL elastic pool. For now, the Pool got 400 eDTU with a standard configuration, and each database are allowed to use a maximum of 300 DTUs. Currently there are 20 databases in the Pool including 6 test databases. All incoming data integration goes through the bulk databases, and then uses external tables in the other databases to run the merge statement. This solution does not perform well with very large datasets, but for now we control the amount of incoming data in the integration applications.

The Database setup consist of 5 layers:

**1st layer** - Holds all bulk tables. No data should be stored here. We should only have write access directly in this layer.

**2nd layer** - Holds data that is widely used across the databases. UTC to CET table, holiday calendar along with mapping and meta data for the databases etc.

**3rd layer** - Is the external data layer. Here we store external data from various sources (PC, WS, EM, Tesla, Trayport etc.). All data in the same format.

**4th layer** - Is the working layer. Here each of the departments has their own database that may be structured however they like. This layer is used for working with the data, not for storing or showing to the end user.

**5th layer** - Is the model output layer. Here we store everything needed for models in operation. E.g. the Price Model, Einsman Model etc.

Bulk table and procedure naming:

* Table: Layer \_ Target \_ Source \_ ScriptName.
* Procedure: Layer \_ Import\_So Table
  + Example: Scraper that collects data from [**https://tennet.nl**](https://tennet.nl) and writes data to **3rd** layer **External** database is called **NL**T**ennetGrabber**.R
    - Bulk table: 3\_External\_Tennet\_NLTennetGrabber
    - Procedure: 3\_ImportExternal\_Tennet\_NLTennetGrabber

### Review our current Azure SQL setup

We need to figure out, if this is the best approach for using Azure SQL Server. We wanted to go serverless, and not just copy our current setup on to Azure. Because we are not able to do cross database queries, we decided to use External Tables, to access other databases. This comes with some performance issues, but we are aware of the possible problems, and try to handle the data when we integrate into the databases, and when we query the databases.

We need to figure out, if we are missing out on something. Will this setup preform when we scale up (Increase the amount of data, speeding up the data integration and adding more data sources)?

We should be more selective about the data we save. We do not need to save all data just because it is available for us. If we can collect the data later on, we should leave it at the source.

### Azure Database Access

Currently we use SQL Server Authentication to access the databases, but we could use Azure AD. If we use Azure AD, we do not have control of AD groups. This could slow the development process.

Should we use Azure active directory? Should we use our own domain, to avoid delays for IT-Operation?

### Schedule job on Azure SQL Database

We will use Azure functions or Azure Flows for scheduling job on the Azure SQL server.

### Azure SQL Database backup

We need to figure out how to restore databases from backup. The backup should be made separately for the databases. Some databases only need backup daily, and some should have backup every hour. We should define how difficult it is to recreate the data. This should be applied for all solutions.

## Data integration applications

Application could be programmed in R, C# .NET Core, C# .NET framework or python. We want to deploy the code using:

* Azure Functions
* Durable Functions
* Docker Container
* Virtual Machines

### Azure functions

Azure Functions should be deployed using Consumption or Premium plan. This will have a maximum duration at 10 minutes. By using Single responsibility principle and Durable Functions, we can handle longer running code. We need to code Azure Functions in C# .NET Core and python.

### Durable Functions

Should be used when we need to orchestrate multiple functions or need to run the same function multiple times. We should be able to run most of our C# code in Durable Functions or Azure Functions.

### Docker Container

We want to run R code on Azure using Docker Container. It can be deployed and build in Azure Container Registry and Container Instance will be used to schedule and run the containers.

We have looked into Kubernetes, but find it to difficult to get started with for now. But later on, we should explorer the benefits by using Kubernetes with Docker.

### Virtual Machines

We will use virtual machines when we cannot run the code using Azure Functions, Containers or other serverless/cloud solutions.

## Logging

For logging we could use Log Analytics or Azure Files to store a log file. To notify about error we should be able to set this up in Log Analytics. This solution could be very expensive, and we need to log only the absolute necessary messages to Log analytics. We could write more comprehensive log files to Azure files.

## How to present the data for then traders

We need to find an alternative for our Excel sheet. We should still be using the Excel sheets, but should not store data in the sheets, and the amount of data going into the sheets should be reduced. The solution should be hosted in the cloud, and with the right security, be accessible anywhere. The trader should be able to trigger calculation from the presentation layer, and run calculations on Azure SQL Server, R code in Docker Containers or Azure functions. Preferably all methods. The trader should be able to pass parameters for the calculations.

The result from these calculations should be shown in the presentation layer and for different users at the same time. The users should get the calculation by refreshing the screen or the data would be streamed to the screens.

## File share on Azure

File-share in Azure/SharePoint?

## Resources Backup

We should create backup for all resources and use source control for all resources deployed to Azure.

We should apply backup for all the resources, and should be able recreate them from the backup.

# **Azure Roadmap (Phase 1)**

## Step (1) Proof of concept (1 Week)

Tasks:

* + - Deploy R and C# Docker Container to Azure - Where are we going to deploy Docker containers?
    - How are we going to show the data for the traders, and how are the traders putting in new input?
    - (Talk to Cloud Solution Architect and data specialist)

## Step (2) Review SQL Setup (2 Weeks)

Tasks:

* + - Review our current Azure SQL setup.
    - Azure Database Access - Create a document with best practice and apply the concept on our current setup.
    - How do we control user access to Azure SQL database?
    - Should we use Azure active directory? Should we use our own domain, to avoid delays for IT-Operation?
    - Create database backup at test how to get things up running.

## Step (3) Clean Up (2 Weeks)

Tasks:

* + - Organize Azure resources and apply correct naming and security. (Management groups, Subscriptions, Resource groups and )
    - Azure Portal - All user should use two-factor authentication - Apply this as default.

## Step (4) Logging and Storage (1 Week)

Tasks:

* + - Application logging in Azure. (Log Analytics, Azure Files or etc.)
    - Should we use common storage
    - File-share in Azure/SharePoint?

## Step (5) Containers and functions (1 Week)

Tasks:

* + - Create a List of excel sheet we should move to the cloud in Phase 1. Add to the list all resources these sheets uses.
    - Create R containers to create data the Excel sheets in Phase 1
    - Move Trayport Market Trade Collector (C# - .Net Core) to a container
    - Deploy all containers
    - Make sure all functions are running

## Step (6) Reports (2 Weeks)

Tasks:

* Create reports matching all Excel sheets. Use the presentation tool for Step (1)

## Step (7) Uptime and backup (1 Week)

Tasks:

* What is the expected uptime for the database, containers, functions and reports etc.
* Setup backup for all recourses
* Rebuild the recourses from backup

## Step (8) Implement the new cloud solution (2 Weeks)

Tasks:

* + - Create access for all users
    - Apply logging for all solutions

## Step (9) Cost analysis (1 Week)

Tasks:

* Analyze cost