Office 365 Timesheet

Getting Started Guide

Contents

[1 Overview 1](#_Toc535865217)

[1.1 Prerequisites 1](#_Toc535865218)

[2 Setup Azure App Service 1](#_Toc535865219)

[2.1 Azure App Registration 1](#_Toc535865220)

[2.2 Application Insights 3](#_Toc535865221)

[2.3 Setting Time Zone 4](#_Toc535865222)

[3 Setting up Azure Function App 4](#_Toc535865223)

[3 Registering Timesheet App 5](#_Toc535865224)

[3.1.1 Registering App 5](#_Toc535865225)

[3.1.2 Configuring your Application Registration 6](#_Toc535865226)

[4 User Setup 9](#_Toc535865227)

[4.1 Setup Hierarchy 9](#_Toc535865228)

[4.2 Create Groups 10](#_Toc535865229)

[5 SharePoint Configuration 11](#_Toc535865230)

[5.1 Configuring SharePoint 11](#_Toc535865231)

[5.2 Configuring Users in SharePoint 12](#_Toc535865232)

[6 Adding Custom Tile to App Launcher 12](#_Toc535865233)

[6.1 Creating Custom Tile 12](#_Toc535865234)

[6.2 Promote the tile to the Home Tab 13](#_Toc535865235)

[7 Deploy Solution 14](#_Toc535865236)

[7.1 Update Settings 14](#_Toc535865237)

[7.2 Update Notification Messages 17](#_Toc535865238)

[7.2.1 Email Templates 17](#_Toc535865239)

[7.2.2 Portal Notifications 18](#_Toc535865240)

[7.3 Update Help section 18](#_Toc535865241)

[7.3.1 Feedback/Support 18](#_Toc535865242)

[7.3.2 About page 18](#_Toc535865243)

[7.4 Publish Azure Functions 18](#_Toc535865244)

[7.4.1 Configuring HTTP Functions 19](#_Toc535865245)

[7.4.2 Configured time triggered functions 20](#_Toc535865246)

[7.5 Local Test and Debug 20](#_Toc535865247)

[7.6 Publish to Azure 21](#_Toc535865248)

[7.6.1 Set Deployment Scope 21](#_Toc535865249)

[7.7 Accept Application Consent as Admin 23](#_Toc535865250)

[8 Setting up Email Notification Schedule 23](#_Toc535865251)

[8.1 User List for reminder emails 23](#_Toc535865252)

[8.2 Schedule-based email notifications using Microsoft Flow 23](#_Toc535865253)

[9 Configuring Cleanup Settings 25](#_Toc535865254)

[9.1 Setting Retention Limits 25](#_Toc535865255)

[9.2 Protecting Key Files 28](#_Toc535865256)

[10 User Experience Overview 30](#_Toc535865257)

[10.1 User Personas 30](#_Toc535865258)

[10.2 Process Workflow 31](#_Toc535865259)

[10.3 Timer Functionality 32](#_Toc535865260)

[10.4 Computation Logic 32](#_Toc535865261)

[10.4.1 Email 33](#_Toc535865262)

[10.4.2 Calendar Entries 33](#_Toc535865263)

[10.5 Role-Based Views 34](#_Toc535865264)

[10.6 User Screens 34](#_Toc535865265)

[11 Extensibility 35](#_Toc535865266)

[12 Troubleshooting 35](#_Toc535865267)

[13 Known Issues 36](#_Toc535865268)

# Overview

The purpose of this document is to provide guidance to an ISV partner who is familiar with Office 365, Microsoft Azure and web development to setup and deploy Office 365 Timesheet solution.

## Prerequisites

To setup the solution, make sure that you have:

* Office 365 Tenant (Business or Enterprise SKU)
* Microsoft Azure subscription
* Visual Studio 2017 (Enterprise, recommended)
* Ensure that you have global admin access on the Office 365 tenant where the solution is being deployed and proceed as follows.

# Setup Azure App Service

Make sure that you have owner access on the Microsoft Azure subscription where the web application is planned to be deployed and made available to the users.

## Azure App Registration

First step is to create the web application where users can access the Timesheet application. Make sure that you have Owner access on the Azure subscription before proceeding.

Sign into [Microsoft Azure portal](https://portal.azure.com/).

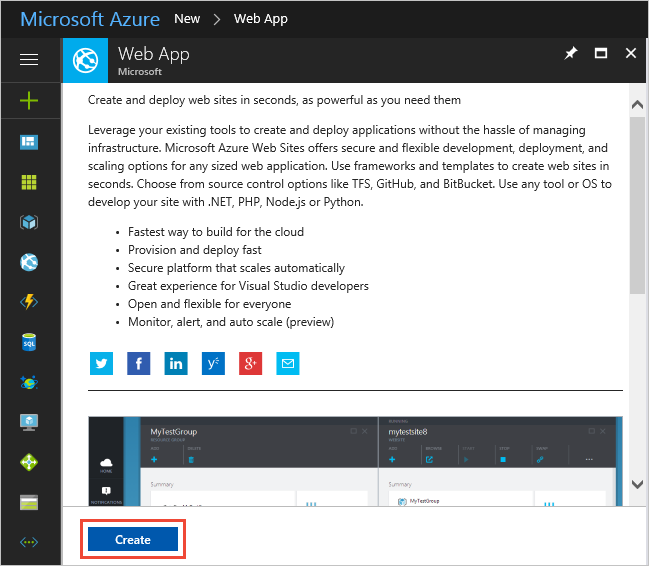


Choose the **+** icon in the left navigation bar, then choose **Web App**.

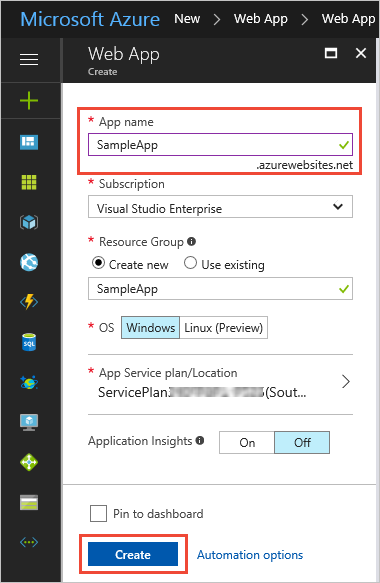


If you don't see **Web App** in the list, use the search box to find it.

At the bottom of the introduction page, choose **Create**.

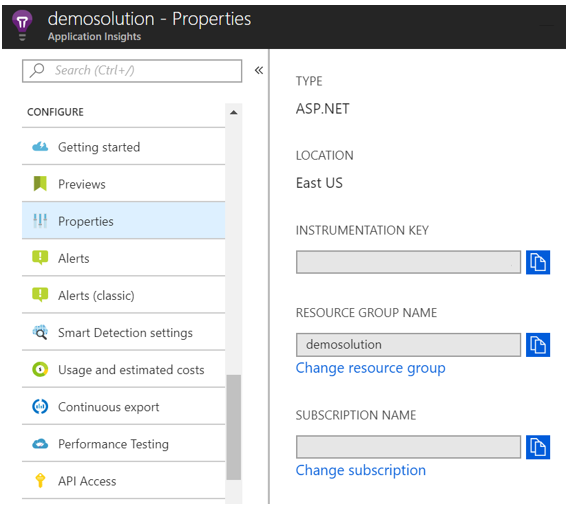


Enter a name for the new web app. You'll see a green checkmark when the name is unique. Then choose **Create**.



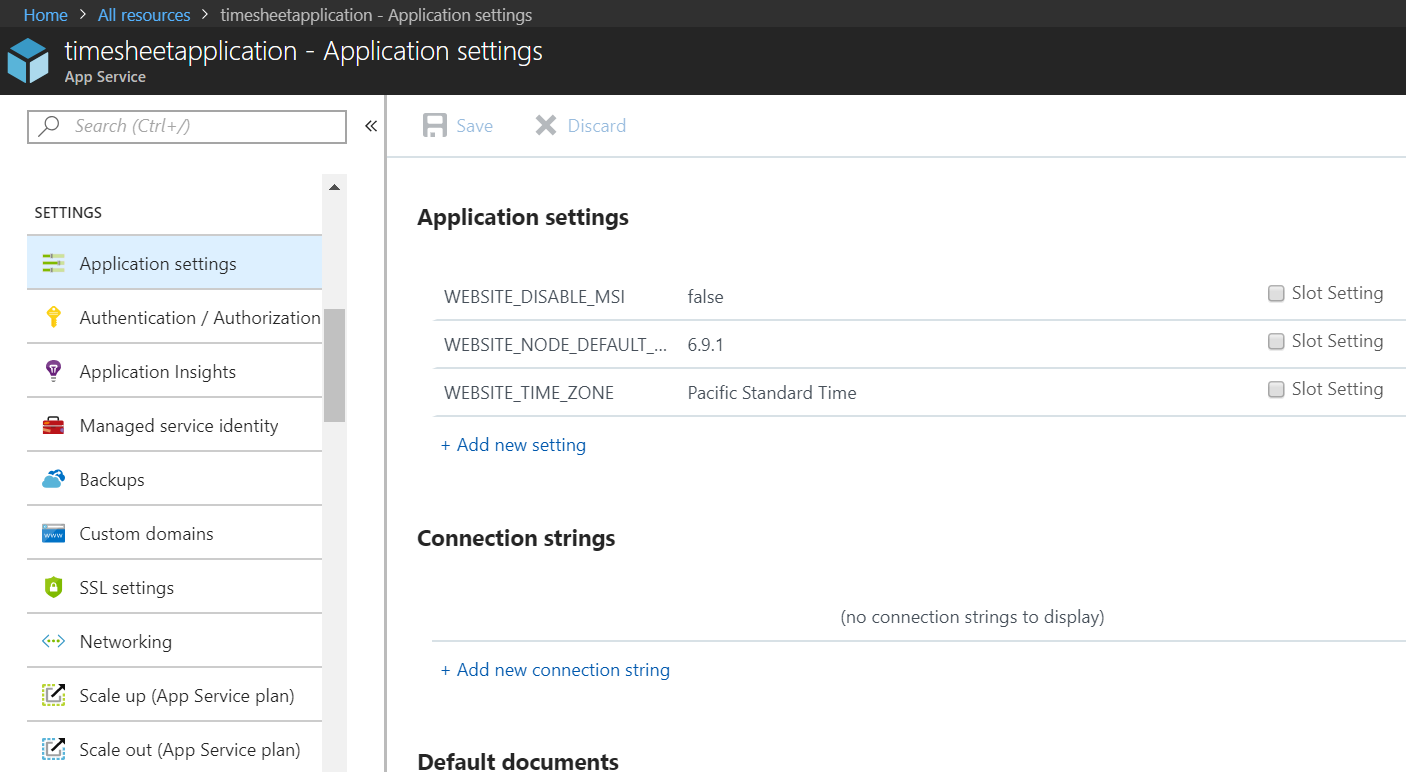
## Application Insights

In case you would like to monitor the live web application, at least initially, it is a good idea to leverage Azure Application Insights. This can be configured from the Azure Portal, Application Insights tab. Take care to note the Instrumentation ID.



## Setting Time Zone

The computation logic in the current version of Timesheet is set based on the *TimeZone* value configured in app settings. It is ideal to configure the app service to run on the specific time zone.

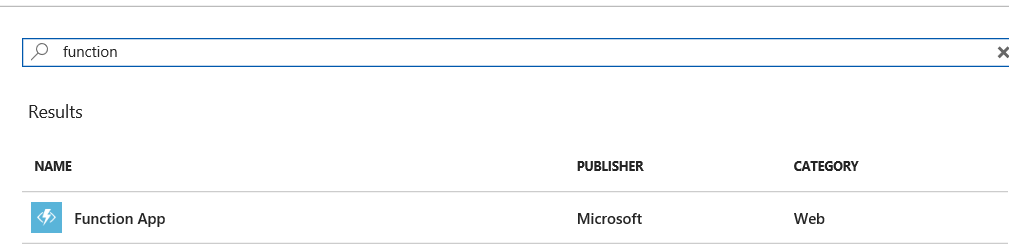


To set the time zone, go to the Application Settings of the app service in the Azure Portal and add a custom setting, WEBSITE\_TIME\_ZONE. Set the name of the time zone as defined in the Windows Registry under *HKLM\Software\Microsoft\Windows Nt\CurrentVersion\Time Zones\* (also listed [here](https://support.microsoft.com/en-us/help/973627/microsoft-time-zone-index-values)). For UTC, set this to ‘GMT Standard Time’.

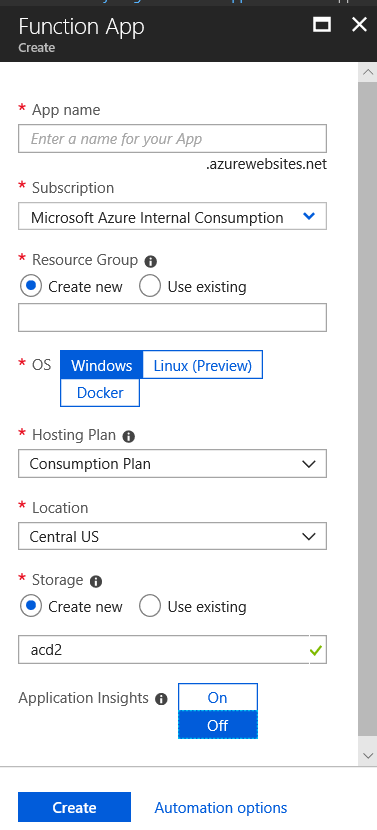
## Setting up Azure Function App

Next we need to set up the Azure Functions that the Timesheet application will be using. Make sure that you have Owner access on the Azure subscription before proceeding.

To proceed, click on Create New and choose Function App:



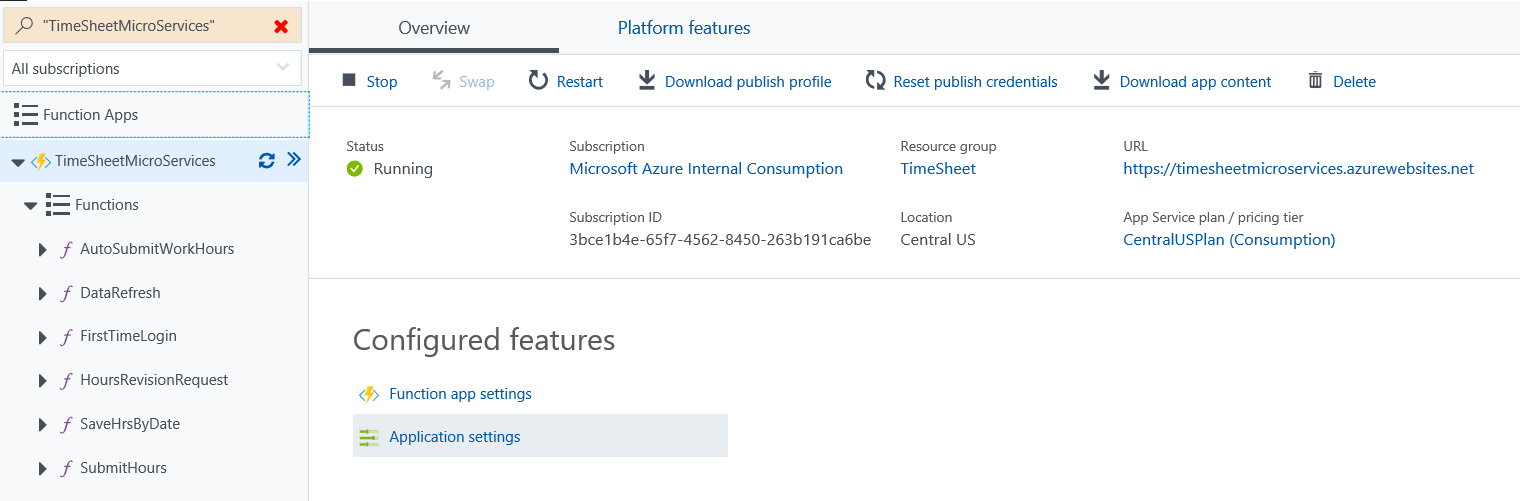
Provide the required details, including a unique name:



Click Create to create the Function App.

Next, we need to set the time zone.

The computation logic in the azure functions is set based on the TimeZone value configured in app settings. It is ideal to configure the app service to run on the specific time zone.



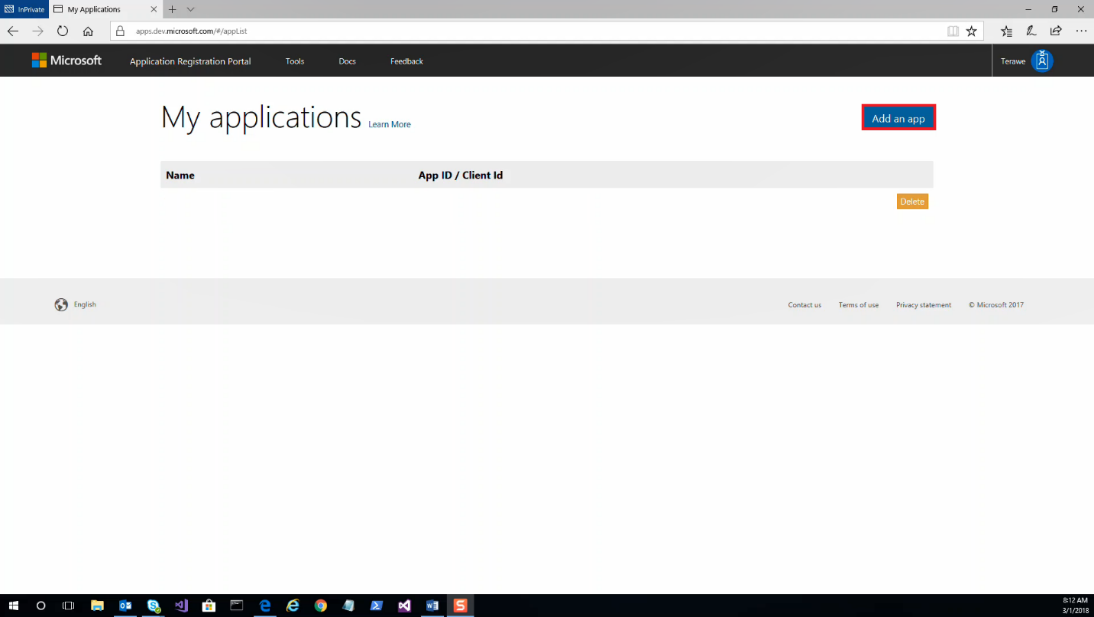
To set the time zone, go to the Application Settings of the Azure Functions in the Azure Portal and add a custom setting, WEBSITE\_TIME\_ZONE. Set the name of the time zone to a value as defined in the Windows Registry under HKLM\Software\Microsoft\Windows Nt\CurrentVersion\Time Zones, based on where the solution is planned to be deployed and used. For UTC, set this to ‘GMT Standard Time’.

# Registering Timesheet App

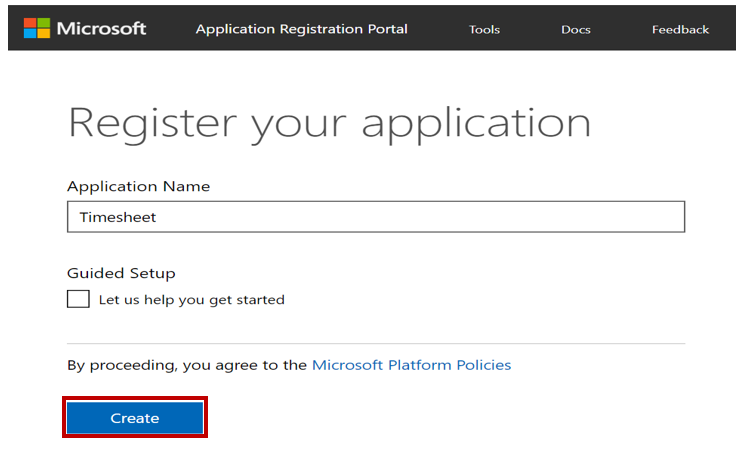
## Registering App

First, we need to register “Timesheet” app to enable users in the organization to access it and to facilitate communication between the dashboard and the solution implementation.

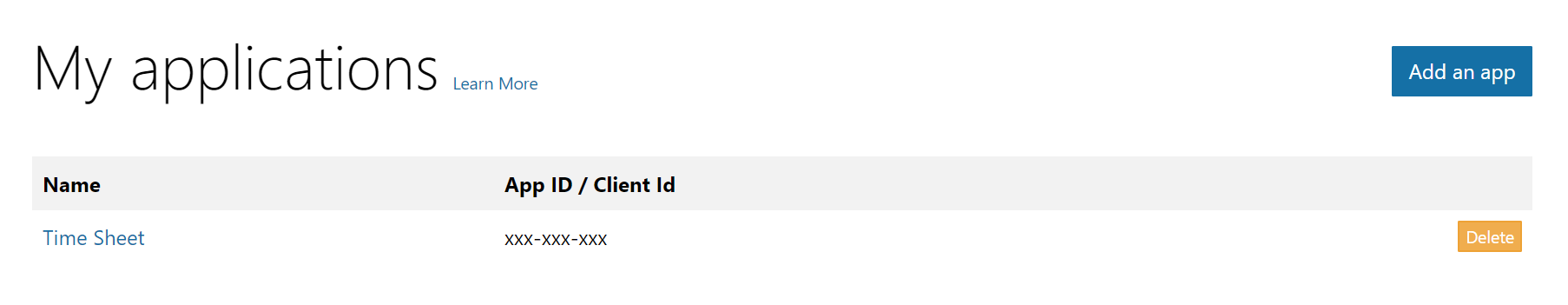
1. To access Application Registration Portal, go to https://apps.dev.microsoft.com and login using your admin account
2. Click on **Add an app**



1. Give an appropriate name, say **'Timesheet'** and Click **Create**. **Note the Application Id**



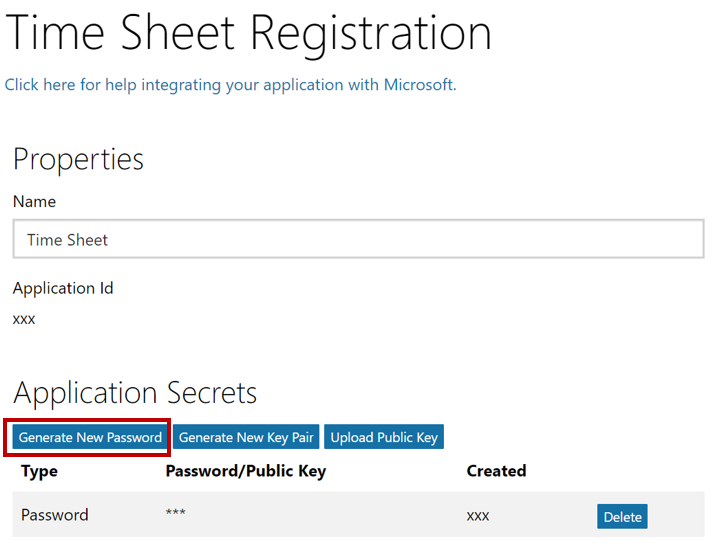
1. Once it is registered you will see it listed as shown below.



## Configuring your Application Registration

Now let’s configure Timesheet Azure Registration using the following details below:

1. Property Name: Enter the name of your application if needed
2. Application ID: Validate that the right application ID is shown
3. Application Secret: Generate a new application secrete that will be use and ensure you capture it for use later. Example:



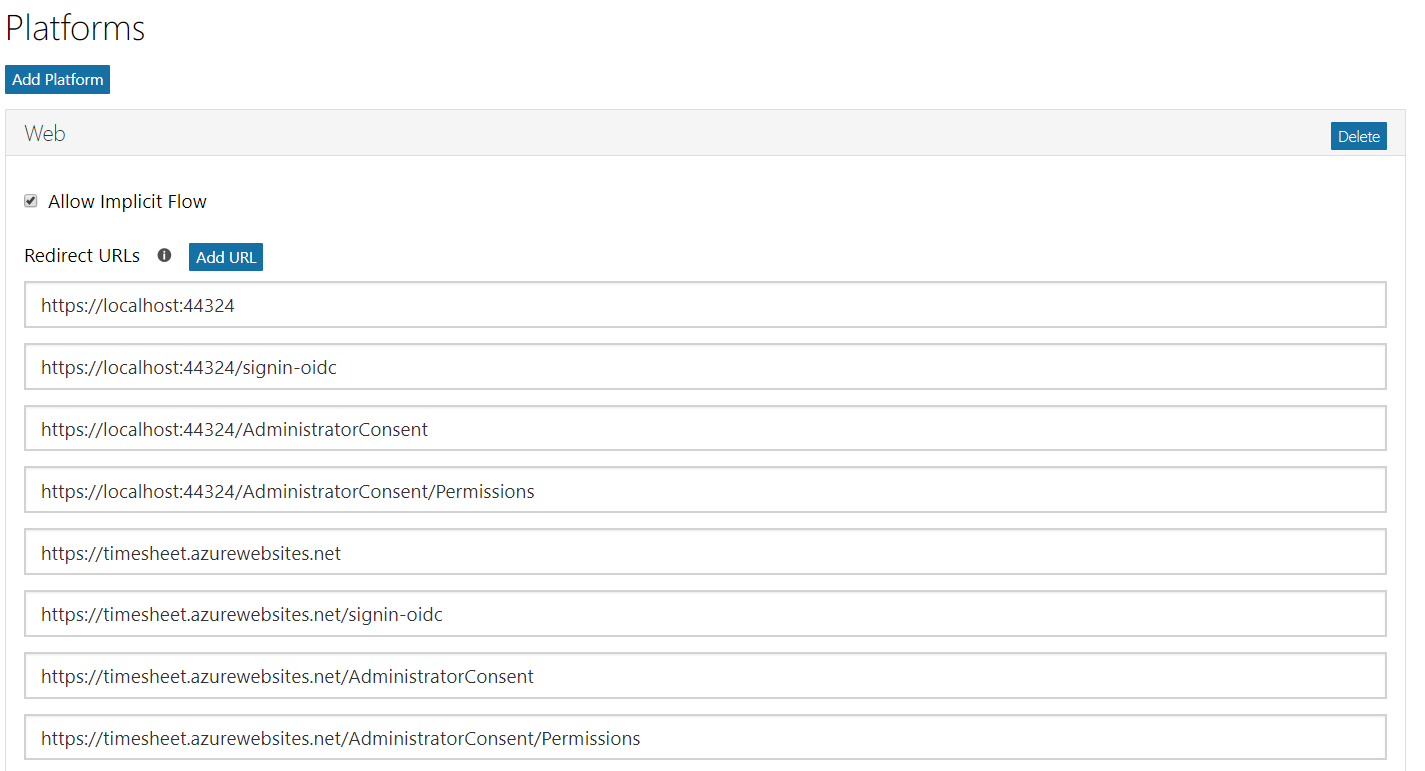
Ensure you note the Password\Public Key which will be needed.

1. Platform: In the Platform section select Web Platform option and continue with its configuration
2. Identify the web URLs that will be needed as shown in the example below:
   1. Check 'Allow Implicit Flow'. In the Redirect URLs section, add the URLs that are indicated as allowed for Timesheet web application, once deployed

For example, if Azure web site is created as https://timesheet.azurewebsites.net, take care to add the following:

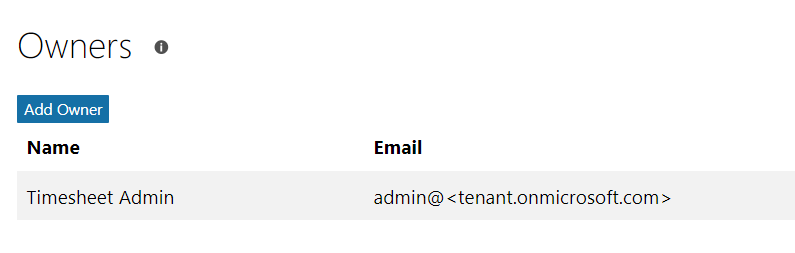
* + - https://timesheet.azurewebsites.net/home
    - https://timesheet.azurewebsites.net/signin-oidc
    - https://timesheet.azurewebsites.net/AdministratorConsent
    - https://timesheet.azurewebsites.net/AdministratorConsent/Permissions

Screen will look something like this:



* 1. Note: Identify your **Home Page** URL, Terms of Service URL and Privacy ULR as needed

1. Identify your Owners: Example below

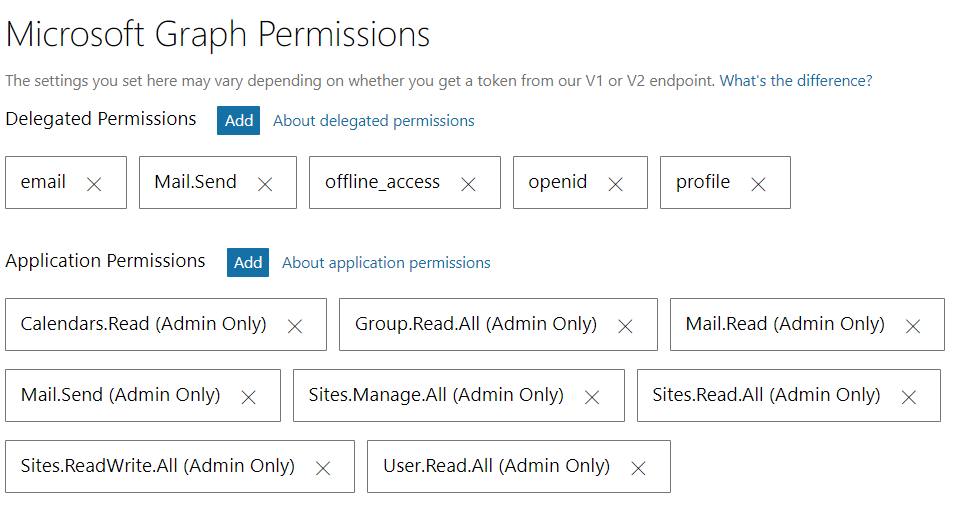


1. Identify Microsoft Graph Permissions as shown below:

Note: In the Microsoft Graph Permissions, set appropriate **Delegated Permissions** and **Application Permissions** by Clicking **Add** and then selecting **permission**.

*See* ***Delegated Permissions*** *and* ***Application Permissions*** *for list of minimum permissions to delegate.*

Example of how what settings we have validate that should be set.



1. Identify your Home Page URL, Terms of Service URL and Privacy URL as needed
2. In the Profile selection, choose a logo that you would like to use and specify the home page URL as the web application URL that you plan to deploy Timesheet to



1. Advanced Options: select Edit Application Manifest to open the manifest. Search for and update the following value as indicated:

"oauth2AllowImplicitFlow": true

Save the changes and go back to the Application settings.

1. Save Settings and you’re done.

**TIP:** To know more about how to leverage Microsoft Graph and samples, go to Microsoft Graph Explorer located at the link below:

* https://developer.microsoft.com/en-us/graph/graph-explorer

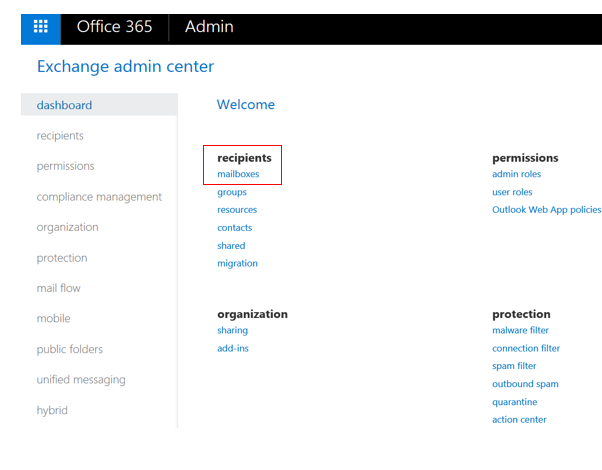
# User Setup

Take care to ensure that organizational hierarchy – Manager relationships – is configured in the tenant to ensure that direct reports and managers can be accurately identified.

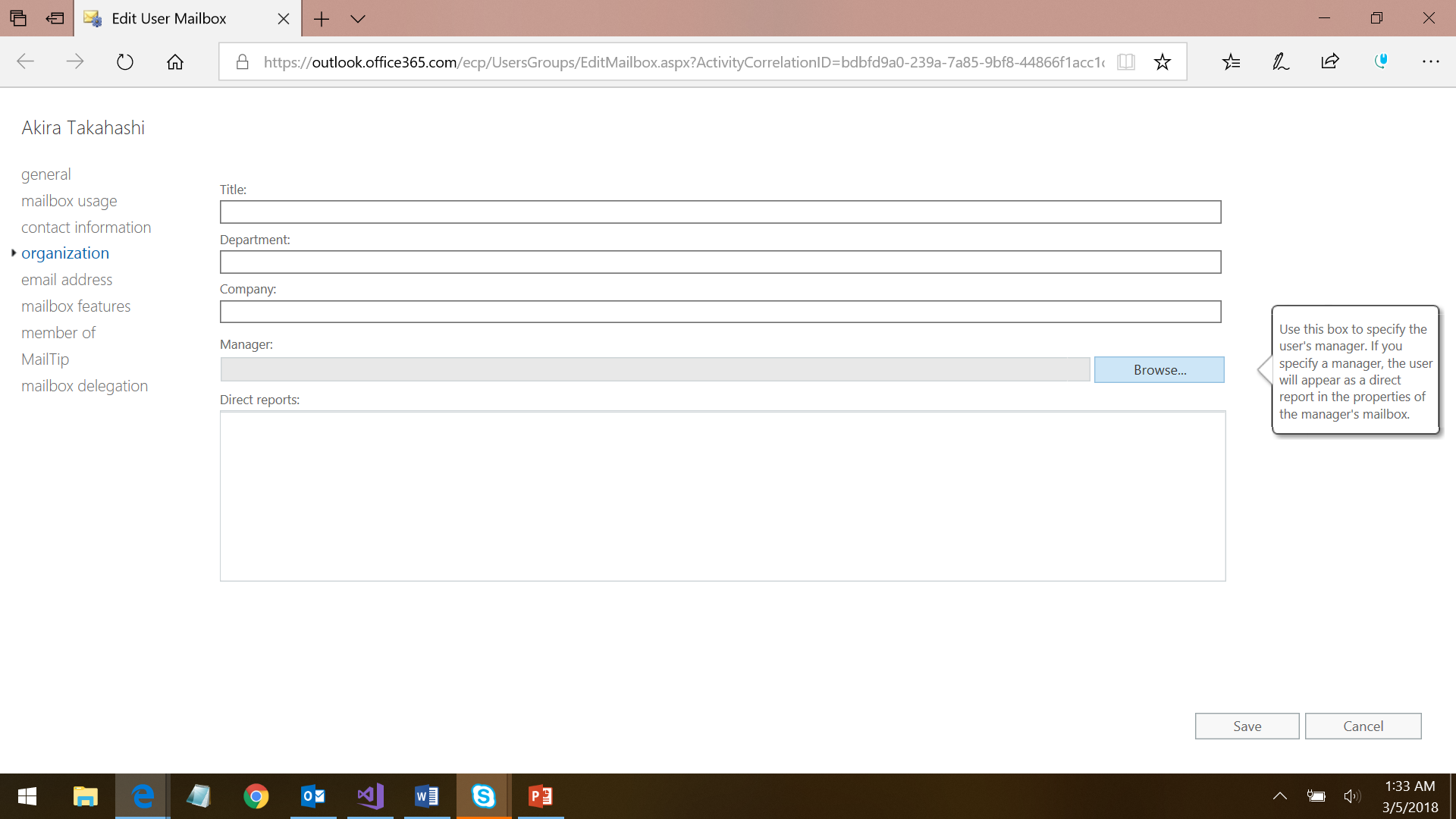
## Setup Hierarchy

Manager for a user is set from the Exchange Admin Center at https://outlook.office365.com/ecp

Select mailboxes to see a list of users with assigned mailboxes in the tenant.



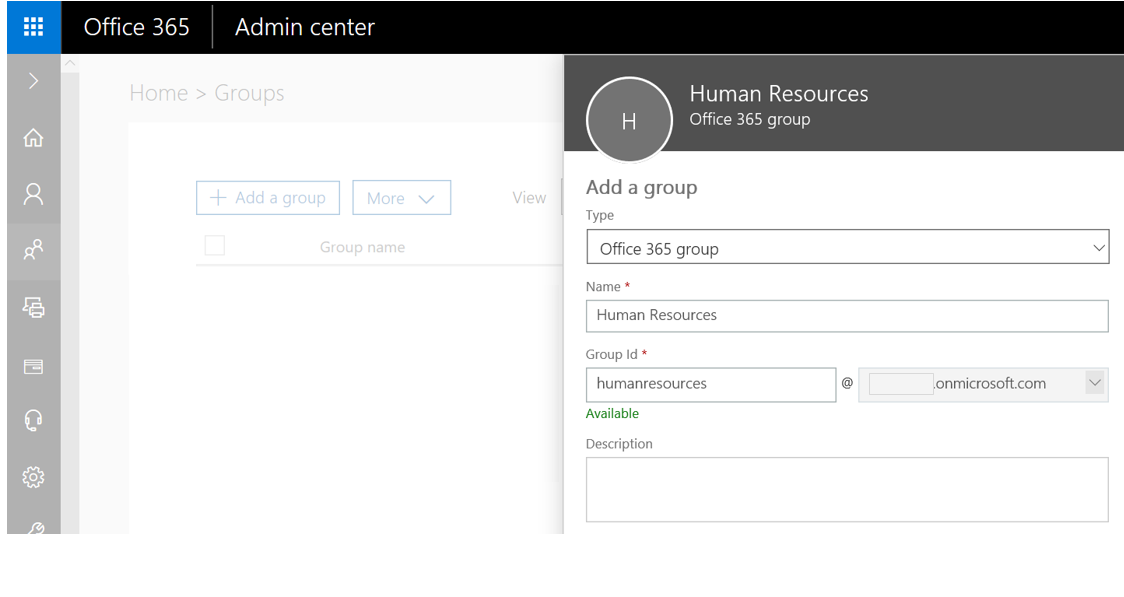
Double-click a user and go to the Organization tab to set the manager for the user:



## Create Groups

To facilitate Timesheet to identify HR users, create a group as follows from Office 365 Admin Portal or from Azure Active Directory settings in Microsoft Azure portal

* **HR Group:** Add users with HR access to a specific group (default: Human Resources)



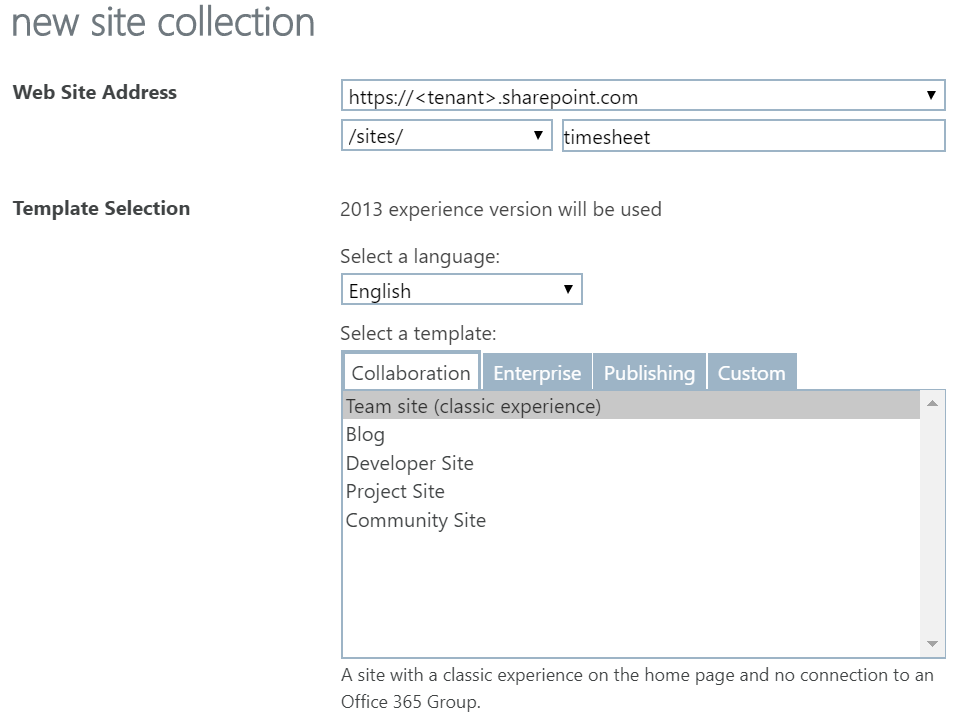
# SharePoint Configuration

In this Section we will configure SharePoint to be used as a repository to store information on user profiles, notifications, work hours and reports.

## Configuring SharePoint

Access SharePoint Management portal at https://<tenant>-admin.sharepoint.com

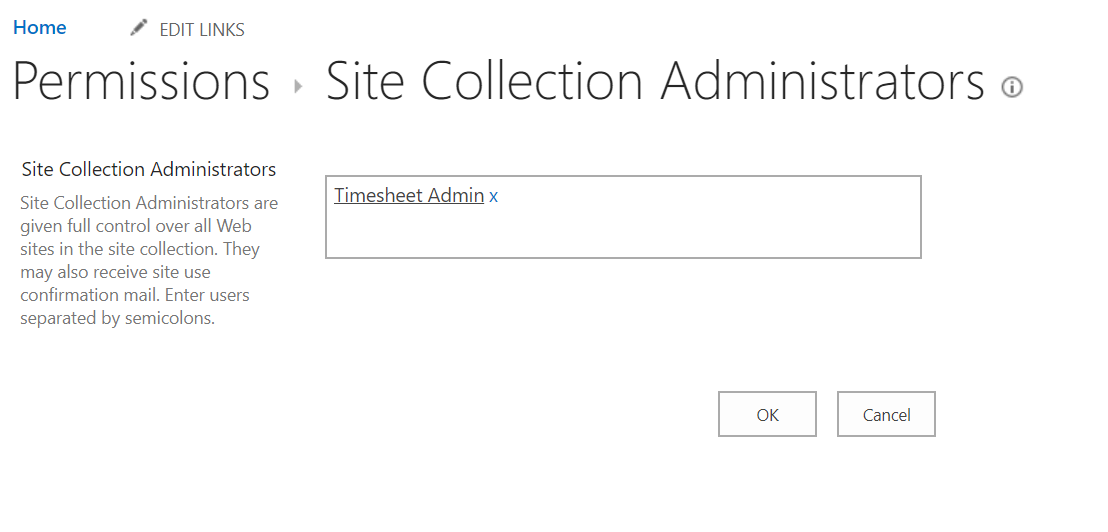
Create a new site, say, Timesheet



Note that the Global Admin who creates the site is automatically added as the Site Collection Admin and owner of the site. Other users with Global Admin rights on the tenant can add or update the site owners.

https://<tenant>.sharepoint.com/sites/timesheet/\_layouts/15/mngsiteadmin.aspx

Example:



## Configuring Users in SharePoint

Timesheet stores data in SharePoint automatically for each user’s time that is being computed in a set of lists. Timesheet runs in the application context, so users are not required to be added to the site to enable update of information in associated lists.

**Note:** Admin should take care to ensure that unauthorized users are never added as members of the site as it contains sensitive user information

# Adding Custom Tile to App Launcher

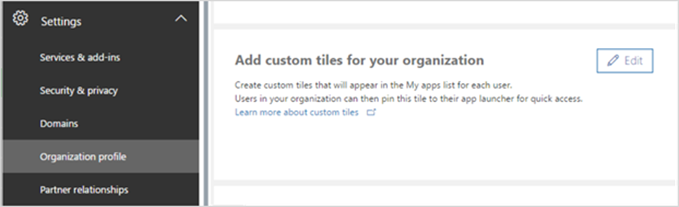
To enable users to quickly access Timesheet, you can add a custom tile to the Office 365 App Launcher, that can be pinned as need to the top navigation bar or to the Home tab.

## Creating Custom Tile

To enable users to quickly access Timesheet, you can add a custom tile to the Office 365 App Launcher, that can then be pinned as need to the top navigation bar or to the Home tab.

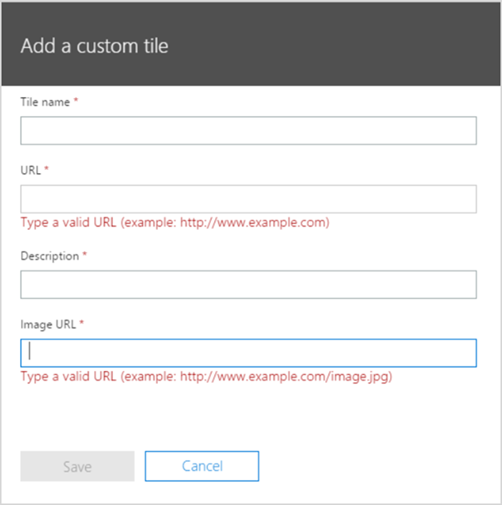
[Sign in to Office 365](https://support.office.com/en-us/article/sign-in-to-office-365-e9eb7d51-5430-4929-91ab-6157c5a050b4) with your work or school account. Select the app launcher icon and choose **Admin**.

In the Office 365 admin center, search for **tiles** or use the left navigation pane by choosing **Settings** > **Organization profile** > **Add custom tiles for your organization**.



**Note:** If you don't see the Custom tiles link, verify you have an Exchange Online mailbox assigned to you and you've successfully signed into your mailbox. Both are required for this feature.

Choose **Add a custom tile**. Enter a **Tile name, URL, Description, Image URL and** choose **Save** to create the custom tile.



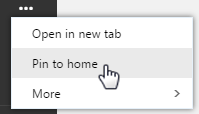
**Tip:** If you’re creating a tile for a SharePoint site, navigate to that site, copy the URL, and paste it here. URL of your default team site looks like this: https://<company\_name>.sharepoint.com

**Tip:** The image should be 60x60 pixels and be available to everyone in your organization. You can, for example, put it in a library on a SharePoint site and generate an [anonymous guest link](https://support.office.com/en-us/article/anonymous-guest-link-80e49744-e30f-44db-8d51-16661b1d4232#__toc371328495) to use as the URL. You may need to first [enable external sharing](https://support.office.com/en-us/article/enable-external-sharing-c8a462eb-0723-4b0b-8d0a-70feafe4be85).

Your custom tile now appears in the app launcher on the **All** tab for you and your users.

## Promote the tile to the Home Tab

Select the app launcher icon and select the **Home** tab. Locate the new tile for your app, select the ellipsis, and choose **Pin to home**.



**Note:** If you don't see the custom tile created in the previous steps, make sure you have an Exchange Online mailbox assigned to you and you've signed into your mailbox at least once. These steps are required for custom tiles in Office 365.

**Important:** Both you and your users need to perform these steps to promote custom tiles from the My apps page to the app launcher.

# Deploy Solution

Get a local copy of the latest version of the source code and open the solution in Visual Studio.

## Update Settings

Open appsettings.json located at .\GoLocal.TimeTracker\GoLocal.TimeTracker.Dashboard and update as follows

**Note:** Properties that are not actively used in Timesheet RC 1.0, but retained from an extensibility standpoint have been marked in grey

|  |  |  |
| --- | --- | --- |
| Property | Description | Example Value |
| **AzureAd** |
| Instance | Public Instance name for AAD | "https://login.microsoftonline.com/" |
| ClientId | ID of the app registered in tenant | "<app\_id>" |
| CallbackPath | Landing page for the app after authentication | "/home" |
| BaseUrl | Return URL for the app after authentication, this should match one of the reply-to URLs specified in the app manifest | "https://<app\_url>.azurewebsites.net" |
| ClientSecret | Client secret for the app registered in tenant | <Noted from the Application Registration portal - apps.dev.microsoft.com> |
| Scopes | Application scope | "openid email profile offline\_access" |
| GraphResourceId | Public end-point for Graph API | "https://graph.microsoft.com/" |
| GraphScopes | Scope required by the application for accessing Microsoft Graph | "Group.Read.All Calendars.Read Mail.Read Mail.Send Sites.Read.All Sites.ReadWrite.All Sites.Manage.All User.Read.All" |
| ApiUrl | API URL for developer debugging | "https://localhost:44376/" |
| TenantId | ID of the tenant where the solution is deployed | "<tenant\_id>" |
| **TimeTracker** |
| DayHours | Specifies number of work hours for the organization at day level with overtime calculated based on the value set. Week and Month level are not supported at this time to facilitate Analytics reports. | "8" |
| WeekHours | "0" |
| MonthHours | "0" |
| FirstDayWeek | Start of work week | "Monday" |
| TimeZone | Time Zone for the application, from values listed [here](https://support.microsoft.com/en-us/help/973627/microsoft-time-zone-index-values) | "Tokyo Standard Time" |
| CutoffDayEmployee | Cut off date for submission of hours by employee to Manager in Cron format | "0 0 3 \* \*" |
| CutoffDayManager | Cut off date for submission of hours by Manager to HR in Cron format | "0 0 10 \* \*" |
| DaysHistory | Number of days for which the data is archived in SharePoint | "360" |
| ReceivedEmailTime | Number of minutes computed for a received email | "3" |
| SentEmailTime | Number of minutes assigned to a sent email | "5" |
| TaskTime | Number of minutes associated with a task based on start date | "30" |
| WorkHoursCacheMinutes | Time in minutes when the work hours cache is refreshed | "2" |
| NotificationsCacheMinutes | Time in minutes when Notifications cache is refreshed | "4" |
| UserProfileCacheMinutes | Time in minutes when User Profile cache is refreshed | "30" |
| ExcludedCategories | Categories that will be excluded from computed hours | "Personal", "Vacation" |
| ExcludedShowAs | Calendar states that will be excluded from computed hours | "Tentative" |
| AllDayCountHours | Time in hours associated with All Day events in calendar | "0" |
| WorkHoursEditable | Enable/Disable editing of work hours by Employee | "true" |
| TeamHoursEditable | Enable/Disable editing of team hours by Manager | "true" |
| EnableTimer | Enable/Disable timer functionality | "false" |
| AdminGroup | Group which identifies employees with Admin access | "TimeTracker Administrator" |
| HrGroup | Group which identifies employees with HR access | "Human Resources" |
| HrNotificationMail | Notification email address for HR group | "<Email address for sending notifications>" |
| HrNotificationAlerts | Indicate if HR should get notifications | "true" |
| SharePointSiteId | SharePoint site where the data repositories are hosted, along with ID | "tenant.sharepoint.com,<site.id>, <site.web.id>" To get this, login to [Graph Explorer](https://developer.microsoft.com/en-us/graph/graph-explorer) and execute the following request query: GET /sites/<url>?$select=id  The response will be in the format: {hostname},{spsite.id},{spweb.id} |
| SharePointSiteName | SharePoint site name | "Timesheet" |
| SharePointGroupId | List used to identify HR users with access to reports. This is not required when an AD group has been configured as detailed [here](#_Toc481056518) | "GroupMembership" |
| SharePointUsersList | List of active users to facilitate scheduled reminder emails | "TimesheetUsers" |
| ReportHoursListPrefix | Prefix for reported hours list in SharePoint | "reporthours\_" |
| TeamHoursListPrefix | Prefix for team hours list in SharePoint | "teamhours\_" |
| WorkHoursListPrefix | Prefix for work hours list in SharePoint | "workhours\_" |
| NotificationsListPrefix | Prefix for notifications list in SharePoint | "notifications\_" |
| SubmitHoursForLastMonth | URL for the Azure Functions obtained from the Azure portal after publishing the Azure Functions as detailed [here](#_Publish_Azure_Functions) | <url> |
| HoursRevisionRequest | <url> |
| SaveHrsByDate | <url> |
| FirstTimeLogin | <url> |
| SubmitTeamHours | <url> |
| GraphRequestUrl | Graph REST API end-point for GET calls | "https://graph.microsoft.com/v1.0/" |
| GraphBetaRequestUrl | End-point for Graph API Beta | "https://graph.microsoft.com/beta/" |
| **ApplicationInsights** |
| InstrumentationKey | If Application Insights is enabled for the site, specify instrumentation key | "<instrumentation\_key>" |

## Update Notification Messages

This section details how the administrator can update the notification messages sent by Timesheet for key events.

## Email Templates

Email templates used for notification emails are included as html files and are shared at .\GoLocal.TimeTracker.Dashboard\wwwroot. The location consists of four files:

* RequestWorkHoursRevision\_template
* SubmitTeamHours\_template
* SubmitWorkHours\_template
* WorkHoursReadyForSubmit\_template

Take care to update them appropriately before deploying the solution. The templates have a placeholder where the direct link to the Timesheet solution can be updated to enable direct one-click access to the solution from within the email notification itself.

## Portal Notifications

Notifications shown in the Notifications page in the Timesheet solution portal can be updated from the resources files located at .\GoLocal.TimeTracker\GoLocal.TimeTracker.Dashboard\Resources\. Note that there is a resource file for each language.

Modify the following strings to update the corresponding notification message:

|  |  |
| --- | --- |
| Event | String to update |
| Employee submits hours | NotSubmitWorkHours1 |
| Manager submits Team Hours | NotSubmitTeamHours |
| Manager requests employee to revise hours | NotRequestWorkHoursRevision1 |
| NotRequestWorkHoursRevision2 |

## Update Help section

Solution consists of a Help section accessible from the ‘**?’** icon on the top navigation bar. Take care to review the content and update appropriately before deploying the solution.

## Feedback/Support

This points to an email address that the end-users can reach out to for any technical issues, feedback or assistance. The address can be updated at *Line 119* of *.\GoLocal.TimeTracker\GoLocal.TimeTracker.Dashboard\Views\Shared\\_Layout.cshtml*

## About page

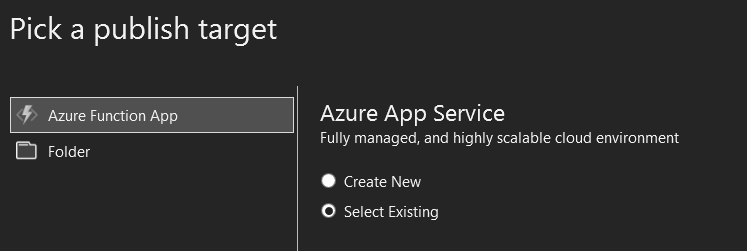
This page contains a basic overview of the Timesheet solution, and can be edited at .\GoLocal.TimeTracker\GoLocal.TimeTracker.Dashboard\wwwroot\about.html

## Publish Azure Functions

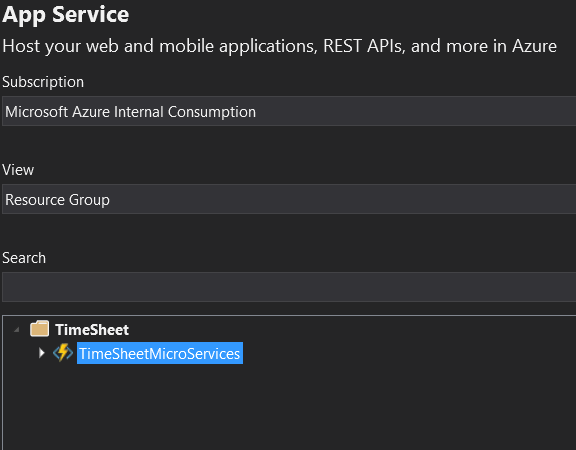
The first key publish step is to publish Azure Functions. Note that the values in the host.json file used in the azure functions are the same as the settings used in the appsettings.json for the web app.

Select the Azure Functions solution and choose Build 🡪 Publish Solution to build and deploy the solution to Azure from the Publish UI.

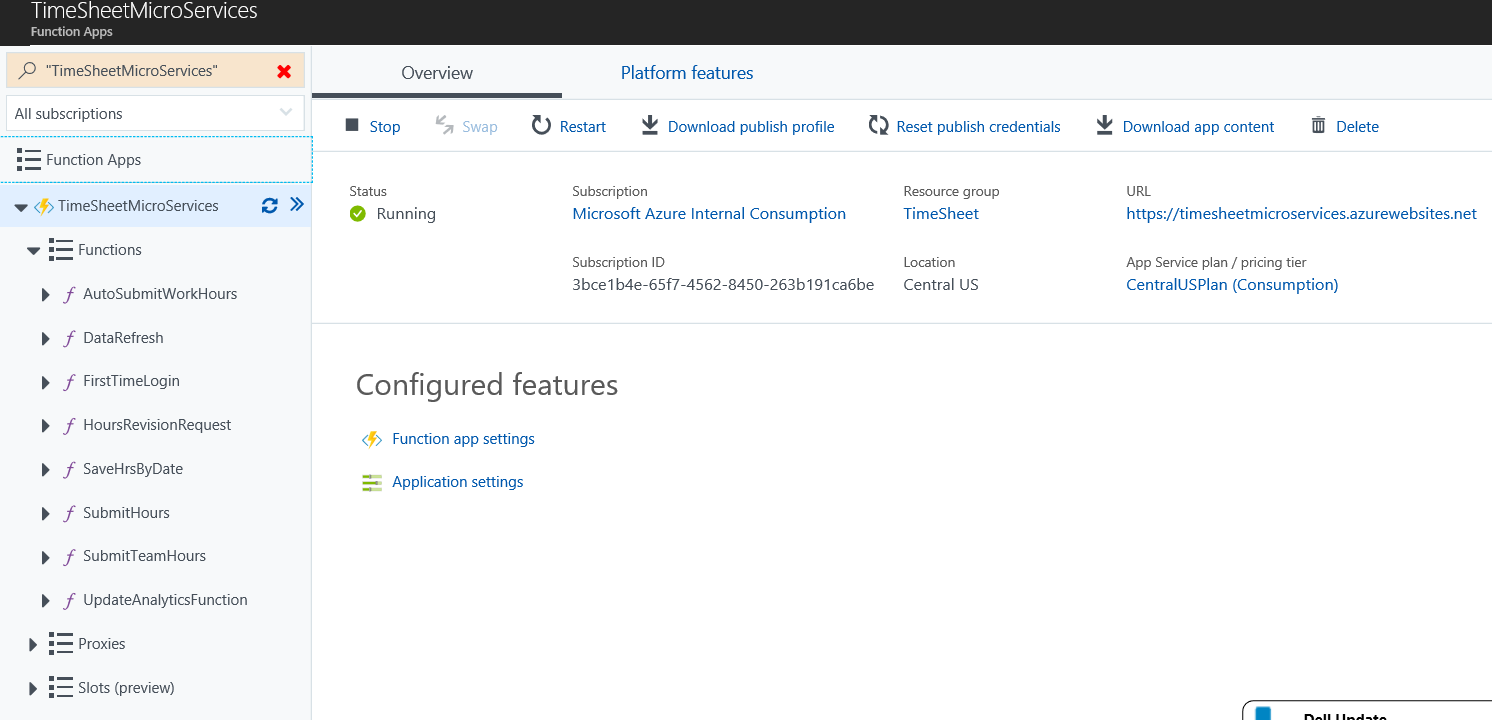
Select Create New Profile option then choose ‘Select Existing’



Click on publish then login with the admin account for the Azure subscription used to host the app service and Azure Functions. Select your resource group.

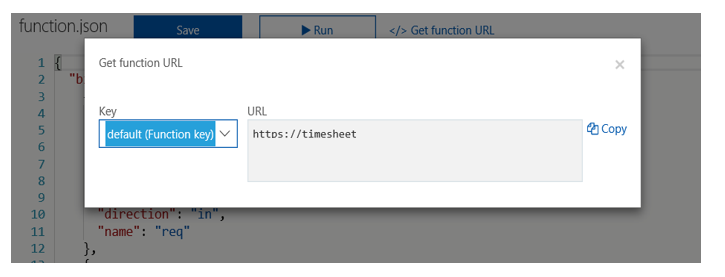


After publishing the Azure Functions log in to the azure portal using the admin account then select the Azure Function application.



## Configuring HTTP Functions

Click on the FirstTimelogin function. Then run the function to make sure it is running.

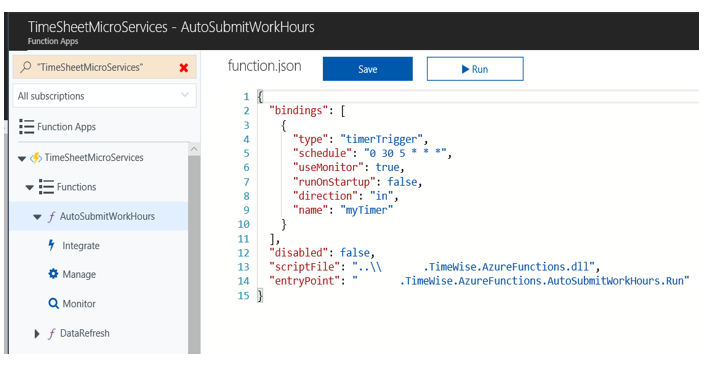


Click on the Get function URL. Copy the URL and save it as this will need to be updated in the appsettings.json file for the Timesheet application.

Repeat the same process and copy the URLs for all the HTTP functions (FirstTimelogin, HoursRevisionRequest, SaveHrsByDate, SubmitHours, SubmitTeamHours).

## Configured time triggered functions

All the time triggered azure functions are set to run every day at 5:30 am by default.



To change the default trigger time, click on any of the time triggered functions (AutoSubmitWorkHours, DataRefresh, UpdateAnalyticsFunction) and update the schedule. The schedule is formatted using the CRON format. For more details on the CRON format refer [here](https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-timer)

## Local Test and Debug

To test the solution locally against the application setup in the tenant, update the appsettings.json appropriately as indicated above, and then set the BaseUrl to a local value such as https://localhost:44324/ Make sure that this is added as a valid reply-to URL in the application as detailed [here](#_Registering_Time_Tracker)

Now run the application in Debug mode from Visual Studio by using the debug controls.

[Debugging in Visual Studio](https://docs.microsoft.com/en-us/visualstudio/debugger/index)

## Publish to Azure

Choose Build 🡪 Publish Solution to build and deploy the solution to Azure from the Publish UI.

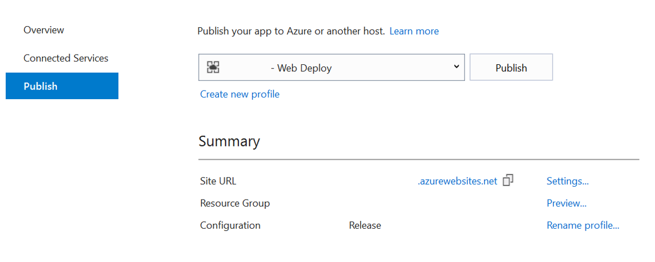
* Select the New Publish profile option and choose Azure App



* Logon using the Azure account that you used to setup the application
* Select the Resource Group where the application is setup and expand, then choose the application



* Click on Publish to deploy the application to the Azure app service

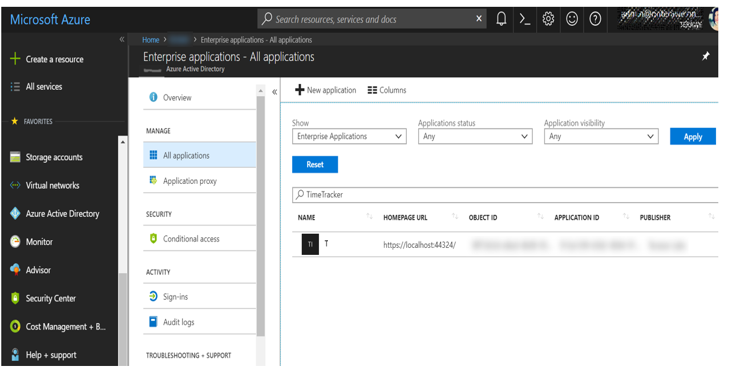


Once publish is completed, the web application is automatically launched on the default browser.

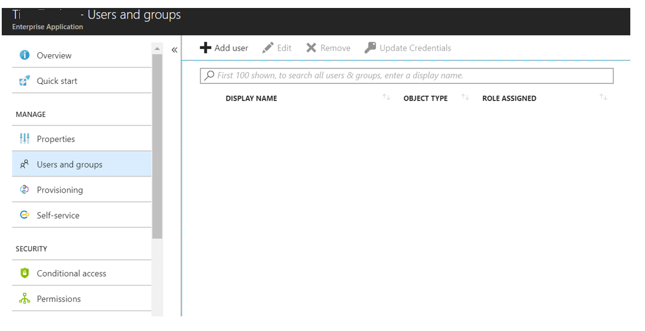
## Set Deployment Scope

To deploy the Timesheet solution to a select set of users in the organization, prior to a global rollout, restrictions can be set in the Azure app as follows:

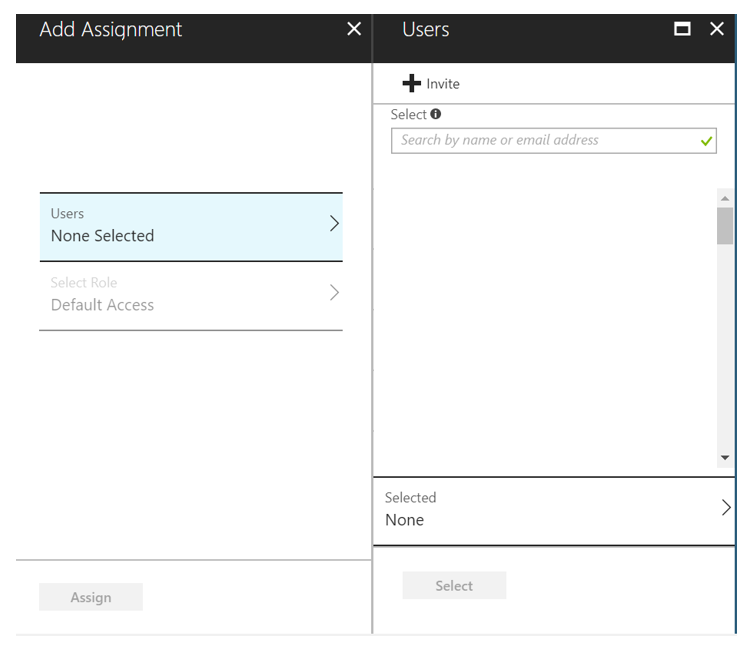
Navigate to the app settings in Azure Portal at Azure Active Directory --> All applications --> Enterprise applications. Select the application configured to be used for the Timesheet solution



Go to Users and Groups



Click on Add User to add Users/Groups who should be given access to the solution. One can also send an invitation to users from this screen.



Following the above steps enables the administrator to do a limited deployment of the Timesheet solution, whereby only a select set of users in the organization can access the solution.

Some scenarios to note here:

* A manager with a subset of users using the solution will see some of the employees as Not Submitted with 0 hours
* Users whose managers are not using the application will be able to submit their hours to the manager, and will get passed on to the HR if auto submit is enabled

When ready, the administrator can remove the restriction by removing access restrictions for the app, which enables all users in the tenant to be able to access the Enterprise application.

## Accept Application Consent as Admin

Once the application is deployed, logon as the admin user at *https://<url>/AdministratorConsent* and click Accept on the list of permissions that show up. This sets application permissions across all users in the tenant.

# Setting up Email Notification Schedule

The administrator can enable email notifications to be sent out to employees at a specific date, to review and submit their work hours at the beginning of the next submission period or especially before the auto submit date, in case enabled.

## User List for reminder emails

The automated reminder is limited to active users of the Timesheet, who have used the application at least once, by automatically updating a list of users at first-time login.

The list, with name configured in app settings, consists of the following information:

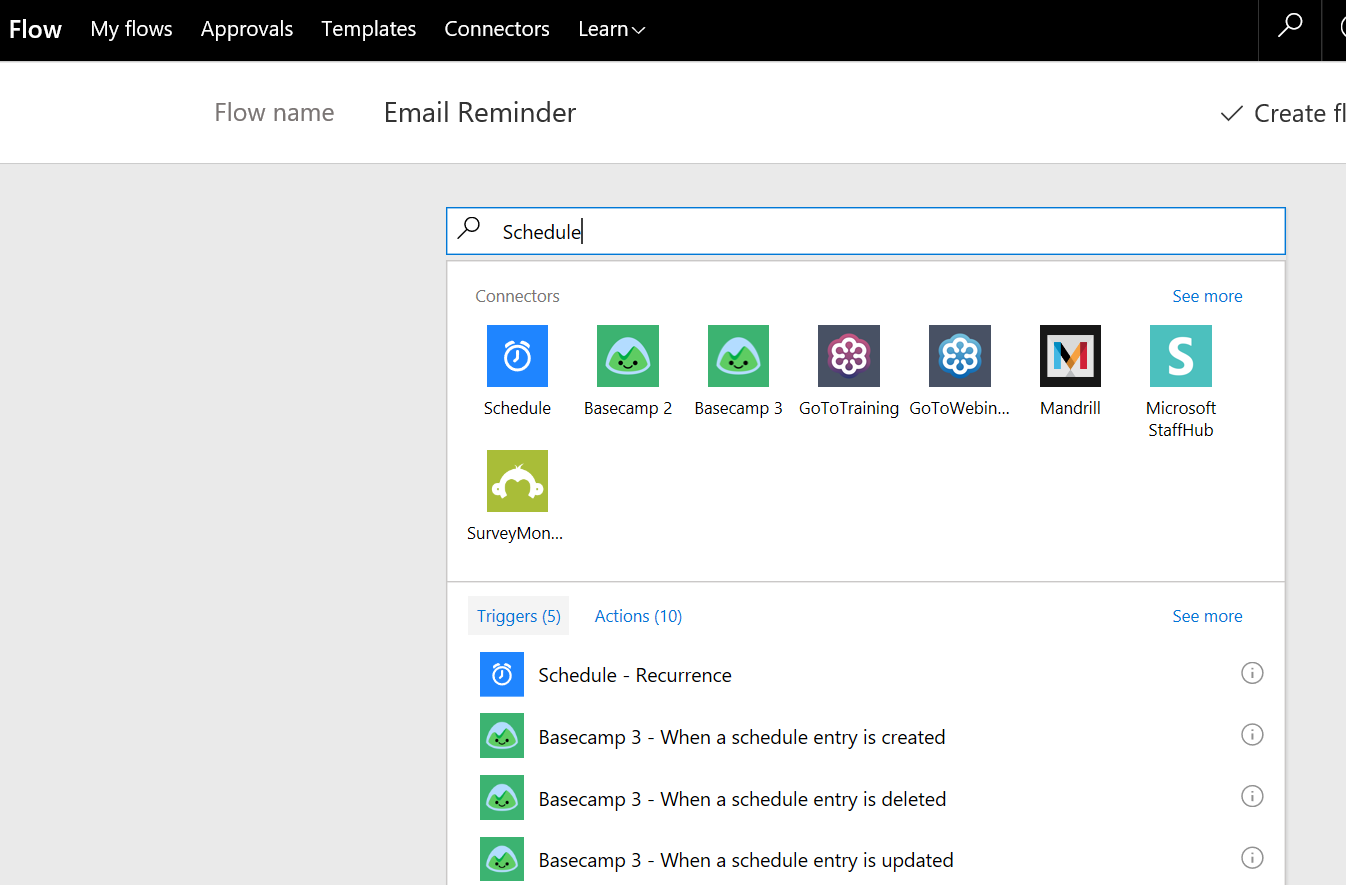
|  |  |
| --- | --- |
| Title | Unique value, say, a number |
| ObjectIdentifier | ID of the logged in user |
| UserMail | Email address of the user, to send notifications |

## Schedule-based email notifications using Microsoft Flow

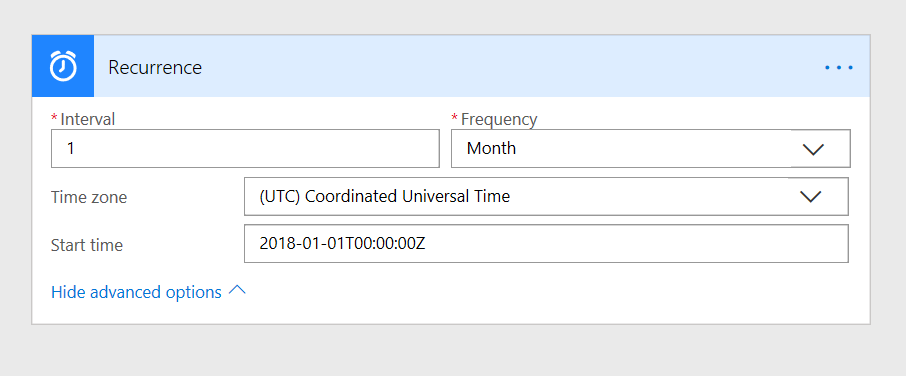
Go to Microsoft Flow – http://flow.microsoft.com

Choose Create Flow from blank template option

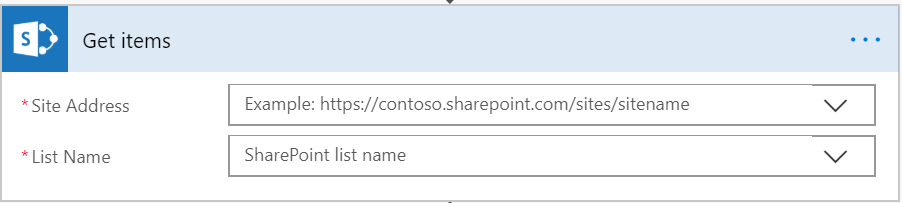
Select Schedule Connector as the first step, and choose Recurrence trigger



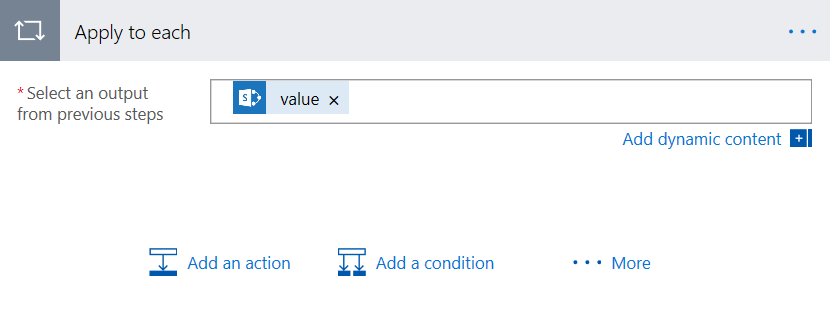
Set appropriate time frame, say, every month:



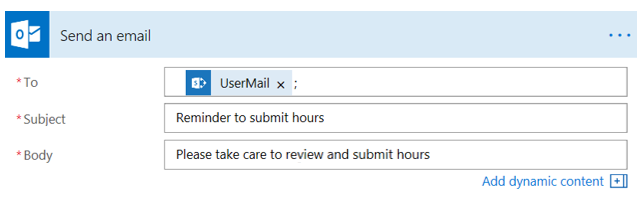
Next, add Action: SharePoint – Get Items



Add an Apply to each on the result set

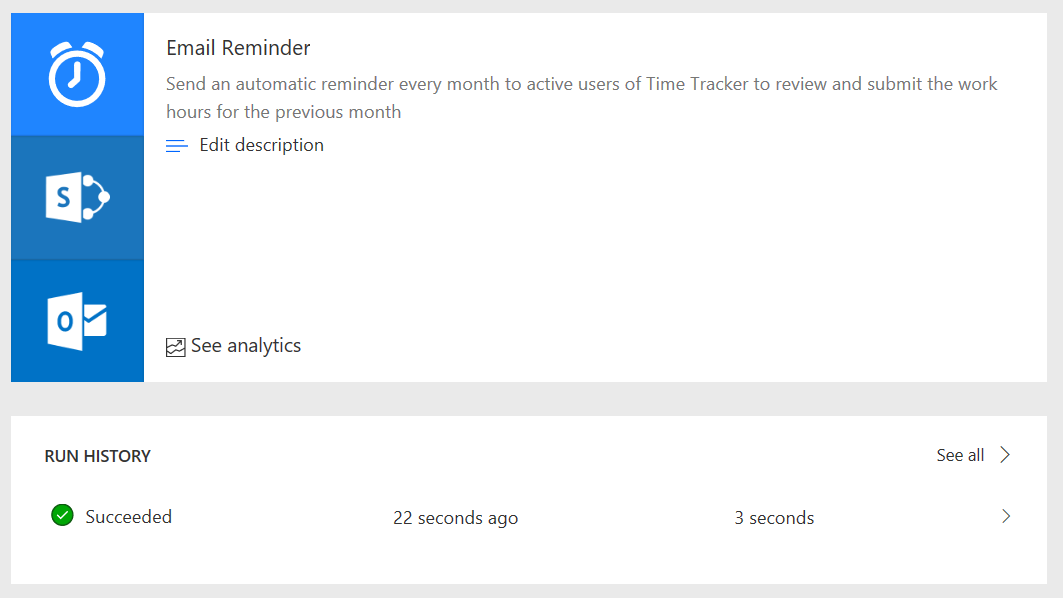


Add new action: Office 365 Outlook - Send an email



Take care to update the Subject and Body in line with the organization policies and locale for the target audience.

Click on Create Flow to save and create the flow



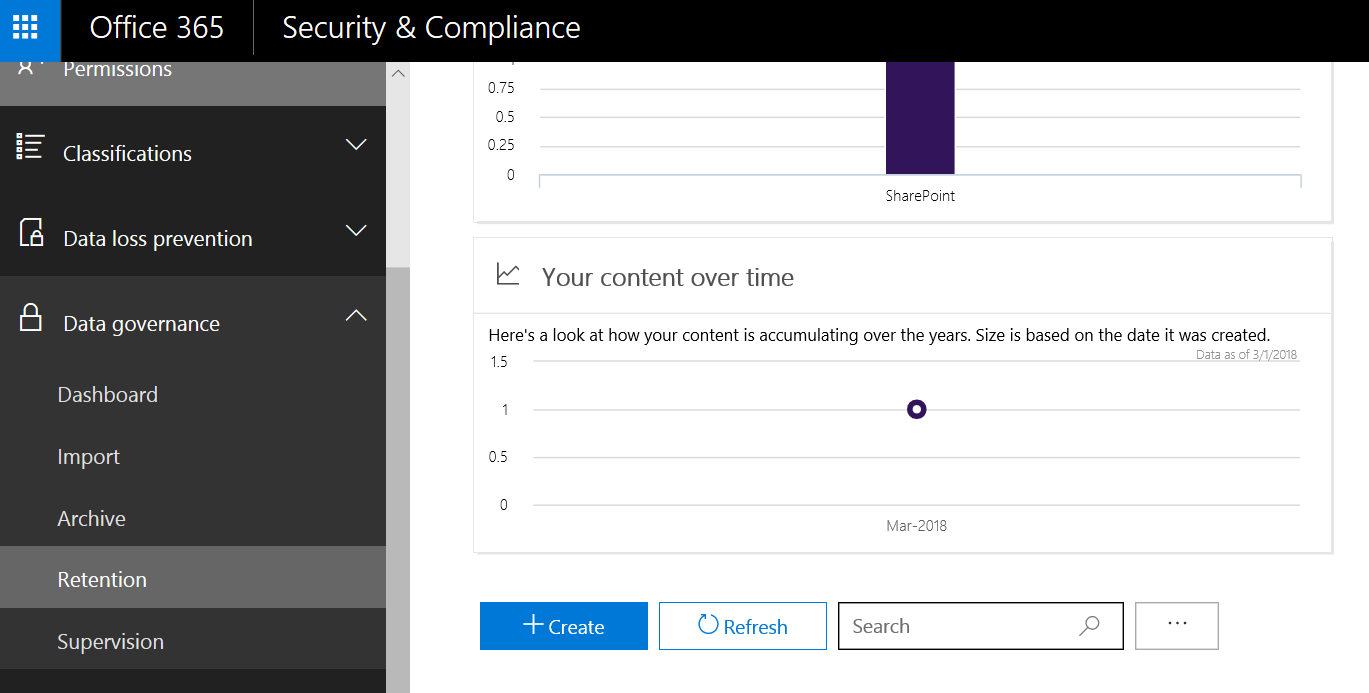
# Configuring Cleanup Settings

SharePoint is the primary storage location for Timesheet, for maintaining repositories for User Profiles, Team Hours, Reports and Work Hours. From a performance standpoint as well as for optimization, it is important to archive or remove files beyond a specific time frame. One can use the powerful retention and archiving capabilities of Office 365 to configure this.

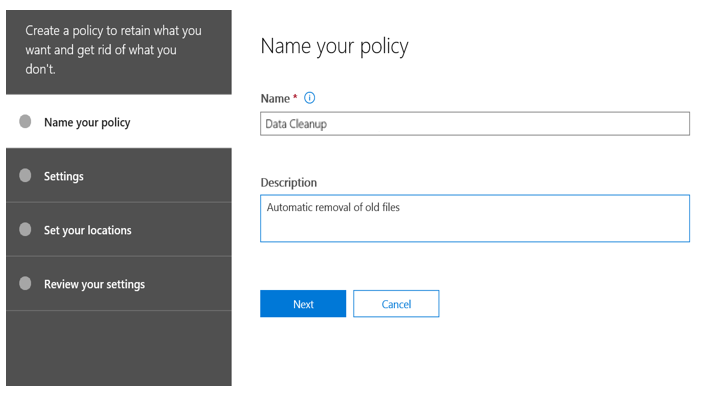
## Setting Retention Limits

Open Office 365 Security and Compliance Center at https://protection.office.com

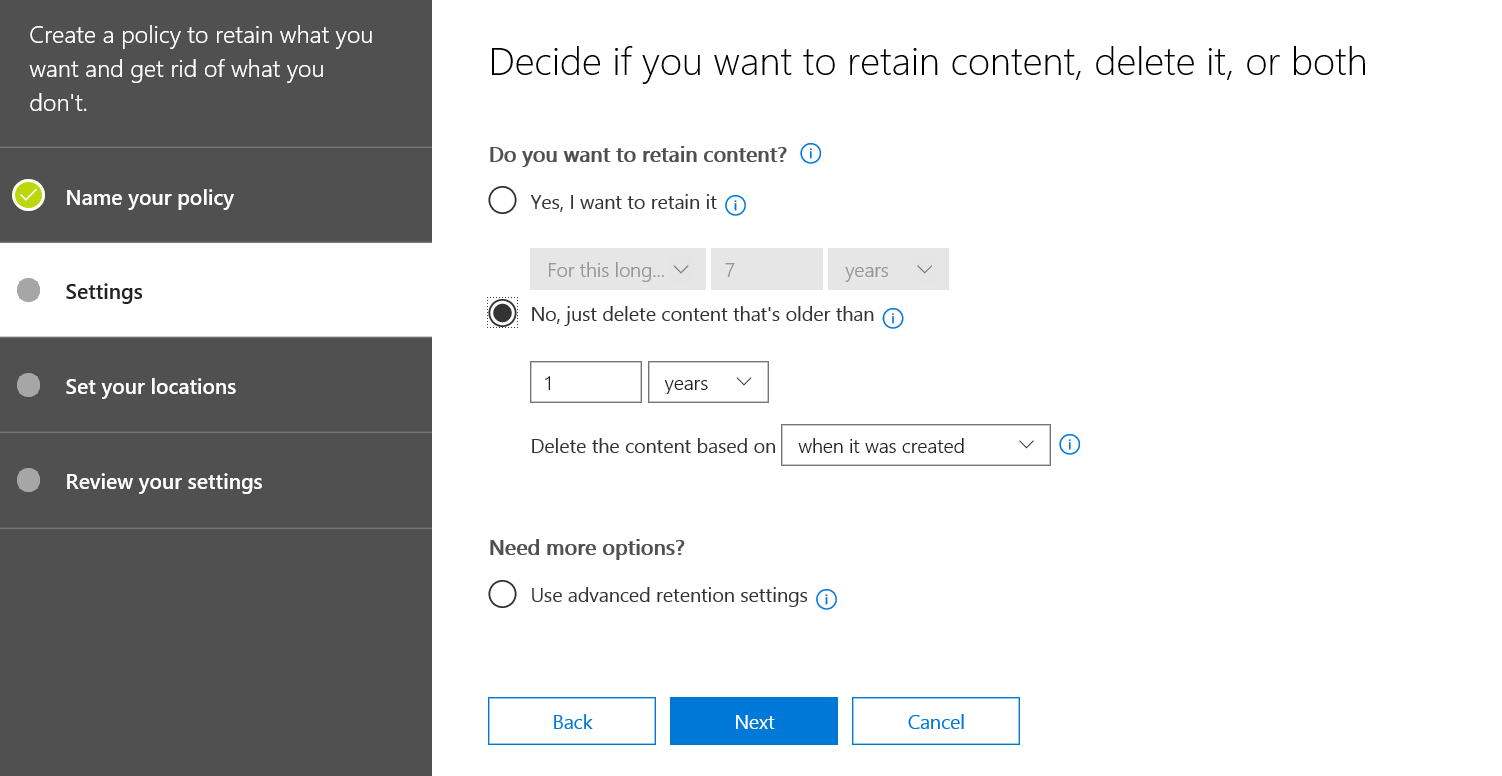
Choose Data Governance --> Retention and click 'Create'



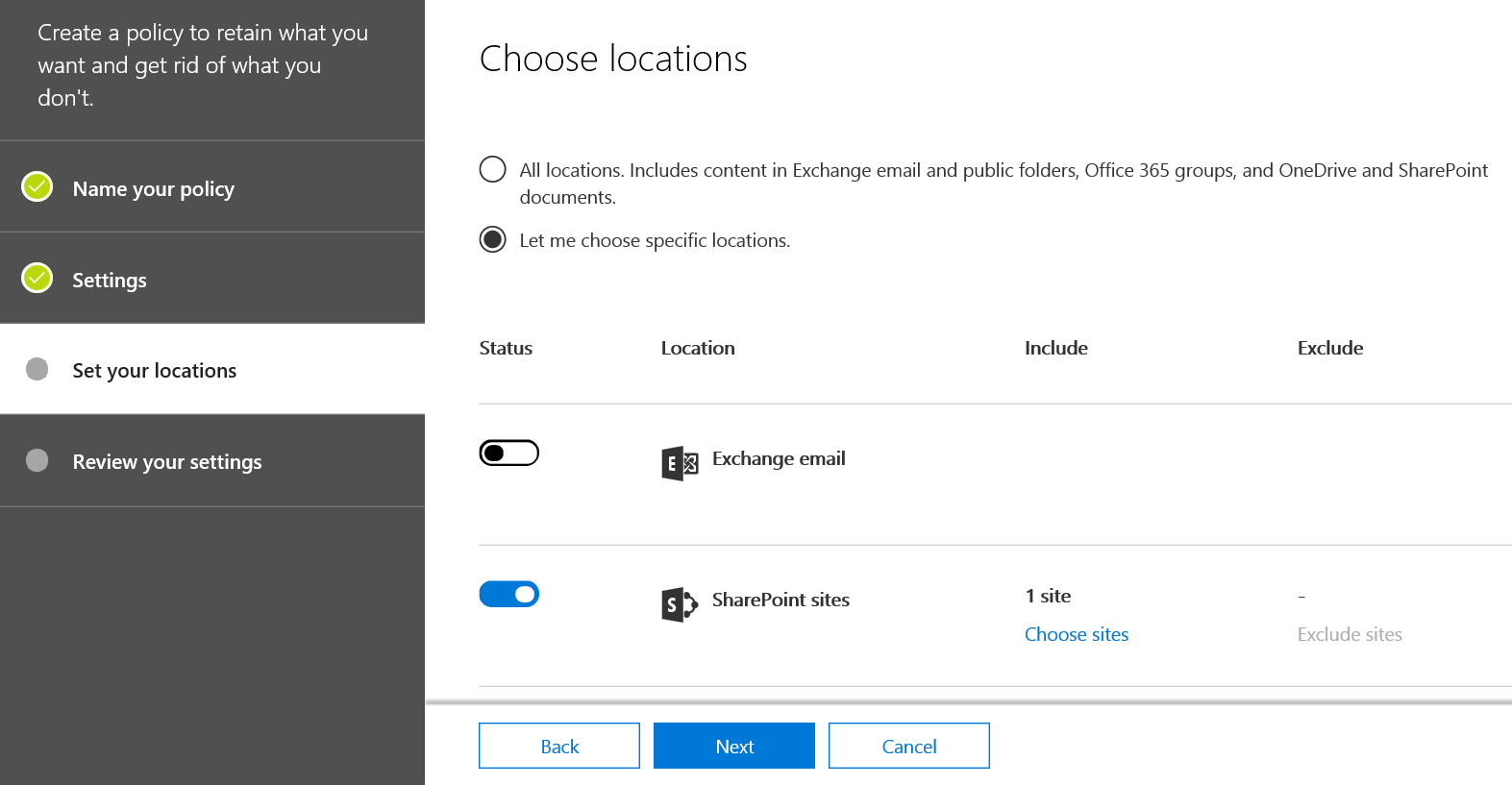
Give an appropriate Name and Description



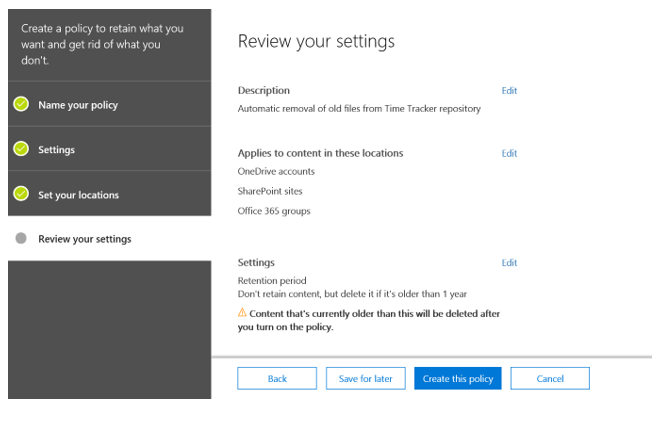
Opt to delete content older than 1 year (or based on organization policy)



Select SharePoint sites, and add Timesheet site to the list



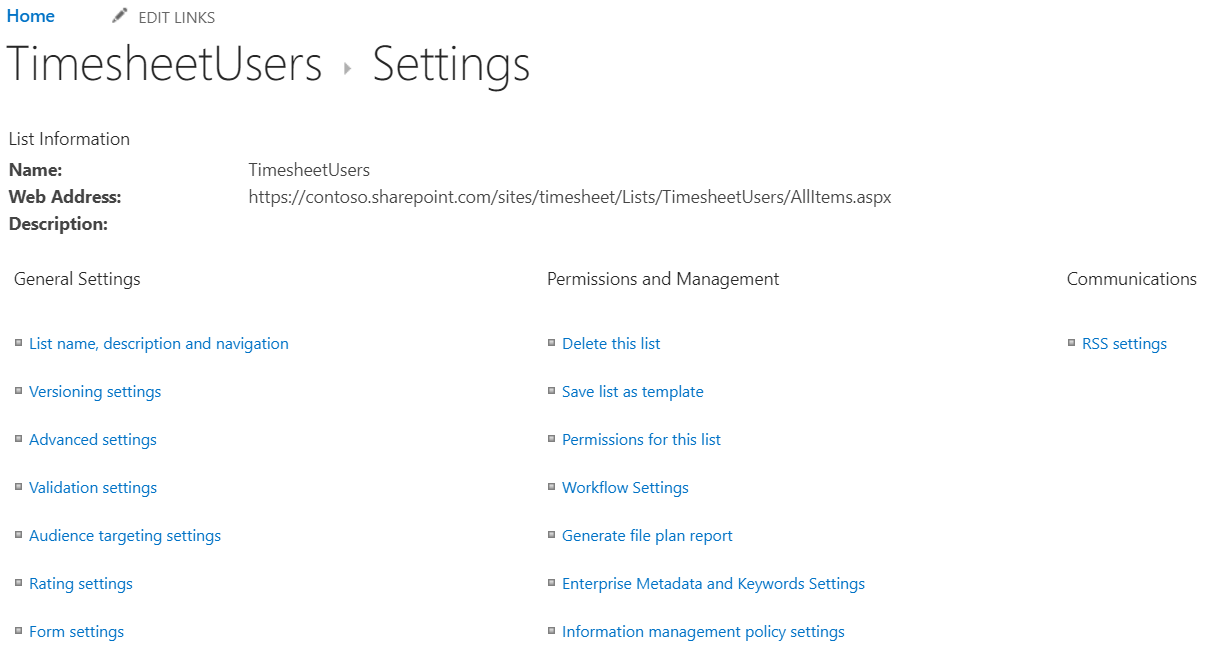
Review settings and click on Create



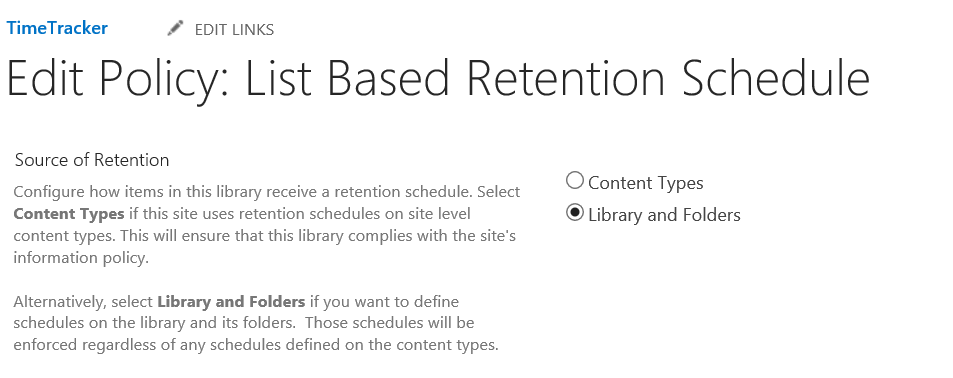
## Protecting Key Files

Lists used for maintaining user email addresses (used to track active users to facilitate automated schedule-based reminder emails) is key to be excluded from the cleanup policy set for the site. To enable this, we can leverage the Retention policy at the List Settings level in SharePoint which will override the site-level settings.

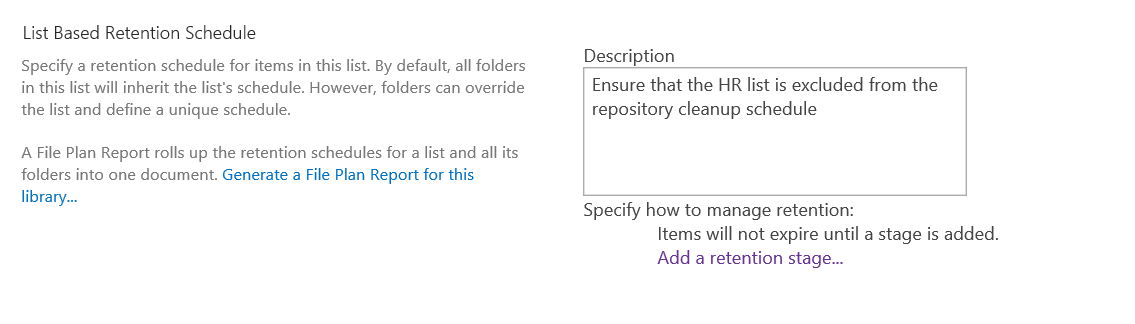
Open the List, TimesheetUsers (as defined in app settings), and choose List Settings from the top menu



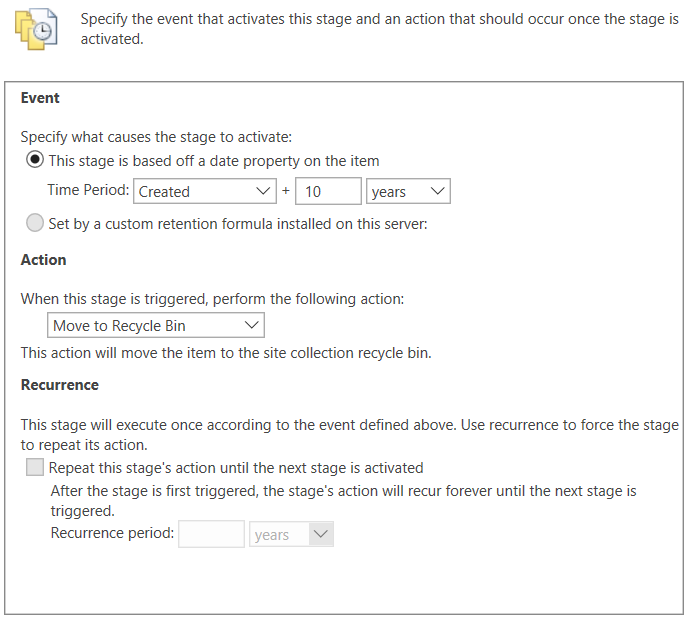
Select Information Management Policy Settings and set Source of Retention as ‘Library and Folders’



Under the List Based Retention schedule, give an appropriate description



Add retention stage, setting the time limit based on organization policies, say, 10 years



Click on Ok to enable the policy, repeat the same process for other lists, based on policy.

# User Experience Overview

This section details at a high level, how different personas interact with the solution.

## User Personas

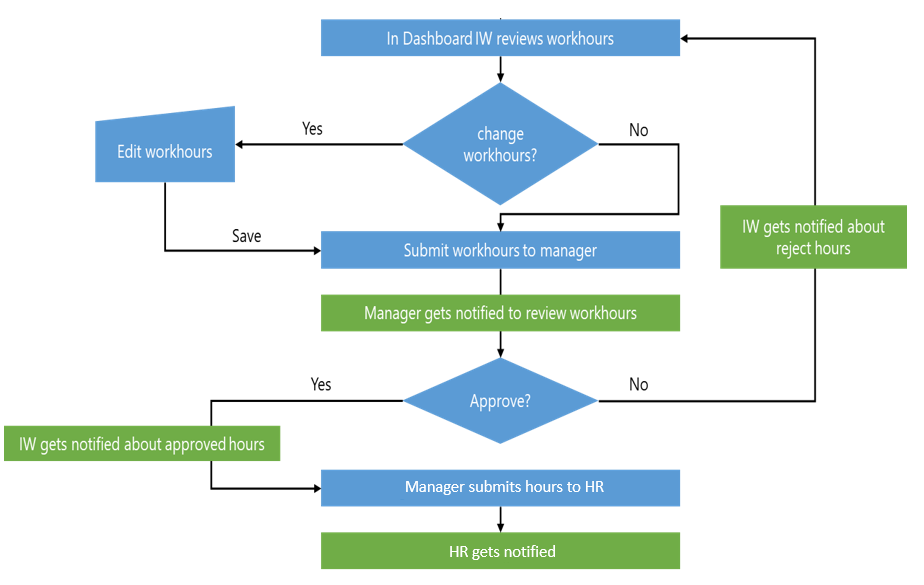
There are four key personas accessing the application:

|  |  |
| --- | --- |
| **User** | Individual Worker (IW) with no direct reports who can review work hours, adjust them as needed and submit to Manager for approval |
| **Manager** | User with direct reports who can review the hours submitted. Manager can then either submit the hours reported by the Team to the HR, or send a notification back to specific workers to adjust and resubmit their hours |
| **HR** | HR can see the hours reported by the workers and submitted following approval by the managers, and generate weekly/monthly reports for further processing |
| **Admin** | Admin can manage the overall settings for the Timesheet |

In addition, an organization can also have some other classes of users such as desk less workers, workers with no or limited activity on Office 365 or temporary workers. They are out of scope for the application.

## Process Workflow

The overall flow for the solution is as follows:



1. IW accesses the dashboard via the custom tile in the Office 365 apps dashboard
   * Administrator can set up automated email reminders for employees to review and submit their work hours by following the steps detailed in [this section](#_Setting_up_Email)
2. IW review the work hours and if needed makes adjustments, if not, just submits the work hours to the Manager
   * If an adjustment is needed, the IW clicks on the “edit” functionality to make a change, once it is done the total work hours gets updated accordingly
   * The edit functionality also allows the IW to add a comment to explain the reason of the time change
   * If Auto Submit is not enabled, for a first-time user, the hours for the 3 most recent months where there are non-zero work hours is automatically submitted. For example, a user submits hours for the first time in March, and there are some hours that have not been submitted in January and February, the hours get submitted for January, February and March.
3. Manager gets notified that work hours are ready to be reviewed, he can click the notification to go to the dashboard or can go manually as stated above
4. Manager reviews the hours and if everything is fine he clicks on submit in order to send to HR otherwise he clicks on a “revise” button for a specific IW (employee) to send the hours back to IW or he can click on adjust in order to make changes to the total hours of that IW. Note that any change or reject will create a notification for the IW
   * Once the IW resubmits the hours, manager will review and approve or sent back for revision which will generate a notification for the IW. Once review is done he will submit the hours to HR
5. HR will get notified according to the notification settings when time entries are ready and have been approved by manager
6. At this point HR can generate the reports or export the information

**Auto Submit:** If Auto Submit is enabled, configurable in App Settings, the hours get auto-submitted for all employees who are yet to submit the hours. Notifications get sent out automatically to managers and HR in this case.

## Timer Functionality

If EnableTimer is set to True in application settings, timer functionality is enabled in the Timesheet application. This enables two additional capabilities:

* Start/Stop button in the dashboard page
  + User can click on start button to log the start time and stop button to record the stop time. Cumulative time recorded this way is the total Timer hours for the day
* Timer Hours row in Edit Hours page
  + In addition to Calculated Hours and Adjusted Hours, an additional row called Timer Hours is also available in the Edit Hours page

Timer Hours is a sub-set of the total calculated hours for the day, filtered based on time during which the user had indicated as working (time between a start event and a stop event). This is further split to:

* Timer Email Hours
  + Email Hours based on emails sent/received when timer was on
* Timer Calendar Hours
  + Calendar events during the time that timer was on
* Timer Other Hours
  + Difference between total timer hours and the sum of timer email and timer calendar hours
  + For example, a calendar event which ended after a stop event will be counted in Other Hours

When Timer functionality is enabled, the timer hours are taken as default throughout the application instead of calculated hours. This can be overridden with Adjusted Hours at the time of submission.

## Computation Logic

This section gives an overview of how the work hours are computed for an employee.

**IMPORTANT NOTE**

All computations in Timesheet are currently based on the time zone configured, at the organization level, in app settings. Support for different custom locales within an organization is in consideration for a future release.

Note that when Timer functionality is enabled, the default hours taken will be the Timer Hours, which will be the total Hours filtered based on when the Timer was on.

Work Hours take into account the following type of activities/items:

## Email

Application queries the Exchange Server directly to fetch the number of emails sent by the user and emails received, and read, by the user. It does not access the content of the email.

Time associated with sent or received emails can be configured in app settings. The default values are set to 5 minutes for Sent Items and 3 minutes for Read items, which can be overridden in app settings. Administrator can specify categories to be excluded in app settings and emails categorized accordingly will be excluded from computing the work hours.

An employee can make use of Outlook rules, for instance, to automatically categorize mails that should not be counted.

## Calendar Entries

All calendar entries are reviewed by the application and processed to compute work hours in line with the following logic:

Meetings and Appointments that are in the calendar are considered for work hours’ computation

Exclusion categories apply for Calendar entries as well, the same way as for emails

Administrator can also configure additional exclusion settings based on the ‘Show as’ status for the calendar entry

|  |  |  |
| --- | --- | --- |
| Category | Is Counted? | Remarks |
| Meeting | | |
| **Status** | | |
| Scheduled by user | Y |  |
| Accepted | Y |  |
| Rejected | N | Once rejected, the event is removed from the user's calendar |
| Cancelled | Y | Once cancelled, the event is marked as Free. Free Events can be excluded using app settings |
| **Show As** | | |
| Busy | Y |  |
| Tentative | N | Default app settings file has this included in the Exclude list |
| Free | Y |  |
| Out of Office | Y |  |
| Working elsewhere | Y |  |
| All Day Meeting | N | Number of hours associated with All Day Events can be configured in app settings, default set to 0 |
| Appointments | | |
| **Show As** | | |
| Busy | Y |  |
| Tentative | N | Default app settings file has this included in the Exclude list |
| Free | Y |  |
| Out of Office | Y |  |
| Working elsewhere | Y |  |

**All Day Events:** Number of hours associated with All Day Events can be configured in app settings, default set to 0

## Role-Based Views

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User Type | | Dashboard | Notifications | Report Hours | Analytics | My Team Hours |
| Individual Worker | | ✓ | ✓ |  |  |  |
| Manager | | ✓ | ✓ |  |  | ✓ |
| HR | With Direct Reports | ✓ | ✓ | ✓ | ✓ | ✓ |
| No Direct Reports | ✓ | ✓ | ✓ | ✓ |  |

## User Screens

|  |  |
| --- | --- |
| **Dashboard** | * Review automatically computed hours and make adjustments as needed, using the ‘Edit Hours’ screen * Submit the hours to Manager * Hours cannot be edited or resubmitted after submission unless sent back for revision by the Manager |
| **Notifications** | * Notifications received by the user |
| **Report Hours** | * Monthly Hours reports that can be printed or exported to CSV by HR |
| **Analytics** | * Graph view of the % of employees who have reported overtime on any specific day in a month * This is based on employees who submitted hours in a specific month with overtime that the manager has approved |

# Extensibility

Timesheet is designed to be extensible to support potential new requirements for partners and customers across multiple industry verticals and regions.

Please refer to the **Timesheet Extensibility Guide for Partners** for details on different extensibility points and how to go about some top use cases in this regard.

# Troubleshooting

|  |  |
| --- | --- |
| **Error** | **Recommended Solution** |
| “Reply-to address does not match one defined for the application” | Confirm that the reply-to URLs specified for the application in the Application Registration Portal exactly matches the reply-to address specified in the error message |
| Dashboard shows 500 error after first time successful load on Azure | Scale up the web app to add more memory to see if it resolves the issue |
| Calculated hours do not match with what is observed in the mailbox | Note that the application uses the time zone configured in app settings for all computation. Refer [this section](#_Setting_Time_Zone) for how to set the app service to run in the context of the time zone to reduce offset |
| Developer Mode Shows when a user tries to log in using any browser | Developer Mode Shows when a user tries to log in: If you try to log into the site and noticed that the site is indicating that it is in Developer mode and unable to log in or does not render correctly, one tip is to verify that all users that need access are members in SharePoint. If they are not this will prevent the application for showing their time data stored in SharePoint. |
| User gets an error about not getting email address after logging in | This is because Graph explorer API is not able to read user details. Check the consent permissions by logging into Azure portal as admin user. If permissions are not granted click on ‘Grant consent’ button. |

# Known Issues

This section lists some of the key known issues:

* Auto Submit
  + Notifications are not sent for auto-submitted work hours
  + A manager should have logged in at least once to the Timesheet application for his/her employees’ work hours to be considered for auto-submit
* Calculated Hours
  + Some offset is possible between the time that an employee has taken an action and the time that gets reflected in Timesheet due to the fact that computation happens in the time zone configured in the app setting, depending on the time zone difference between the employee and the configured time zone
  + Calendar events that span across two days are counted based on when the event started. For example, a meeting that started at 10 PM and ended at 3 AM the next day is counted as 5 hours on the first day
  + Overlapping meetings will get counted separately as different events
* Login may take longer the first time as all the calculations are completed, but will be faster subsequently as the records start getting cached
* Events such as status change and notifications may be delayed in some cases based on the cache settings configured for the application
* Timesheet is supported on Microsoft Edge and Google Chrome.