



**Mobiliya Fleet Management**

**User Guide**



**Revision History**

Version No.	1.0
Authorized By	Sagar Shah

© 2018 Mobiliya Technologies

Strictly Private and Confidential. No part of this document should be reproduced or distributed without the prior permission of Mobiliya Technologies.

## Table of Contents

<b>1. Introduction.....</b>	<b>4</b>
1.1. About this Guide.....	4
1.2. About Mobiliya Fleet Management Solution.....	4
<b>2. System Architecture.....</b>	<b>5</b>
2.1. Mobile App Components.....	5
2.2. Cloud Application Components.....	7
<b>3. Intended Audience.....</b>	<b>8</b>
<b>4. Deploy Resources with ARMTemplate on azure.....</b>	<b>9</b>
<b>5. Deployment of Web App services on azure.....</b>	<b>13</b>
<b>6. User Instructions.....</b>	<b>17</b>
1. Web Portal.....	17
1.1 Setup and Configuration changes.....	17
1.2 Super Admin.....	18
1.3 Tenant Admin.....	19
2. Android Application.....	21

## 1. Introduction

### 1.1. About this Guide

The purpose of this user guide is to assist developers in understanding and setting up the Mobiliya Fleet Management solution. It is a step-by-step walkthrough the setup process & usage guidelines of the solution.

### 1.2. About Mobiliya Fleet Management Solution

The complete solution consists of following components,

- Vehicle (Truck/Car)
- OBD/J1939 Dongle
- Mobile Application
- Azure Cloud Application

Commercial Vehicles supporting J1939 protocol will be supported by the solution. Cars supporting OBD Protocol will be supported by the solution.

A user will connect Dongle to vehicle which will retrieve vehicle diagnostic information and forward this information to Mobile application over Bluetooth. Mobile application later on will forward this information to cloud. Cloud application further performs detailed analysis of a given data and provides different reports to stakeholder/user.

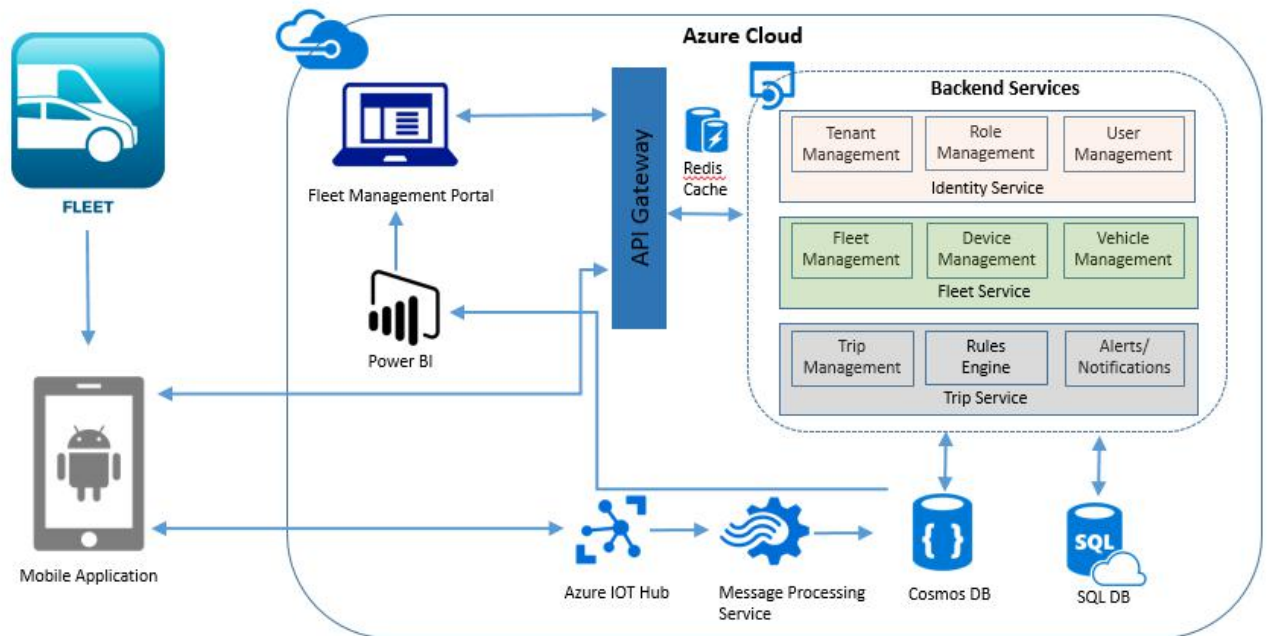
There are 4 types of user actively involved in this system.

- Super Admin (A person who has complete access to system.)
- Tenant Admin
- Fleet Admin
- Driver

## 2. System Architecture

The architecture comprises of two main components,

- Mobile Application
- Cloud Application



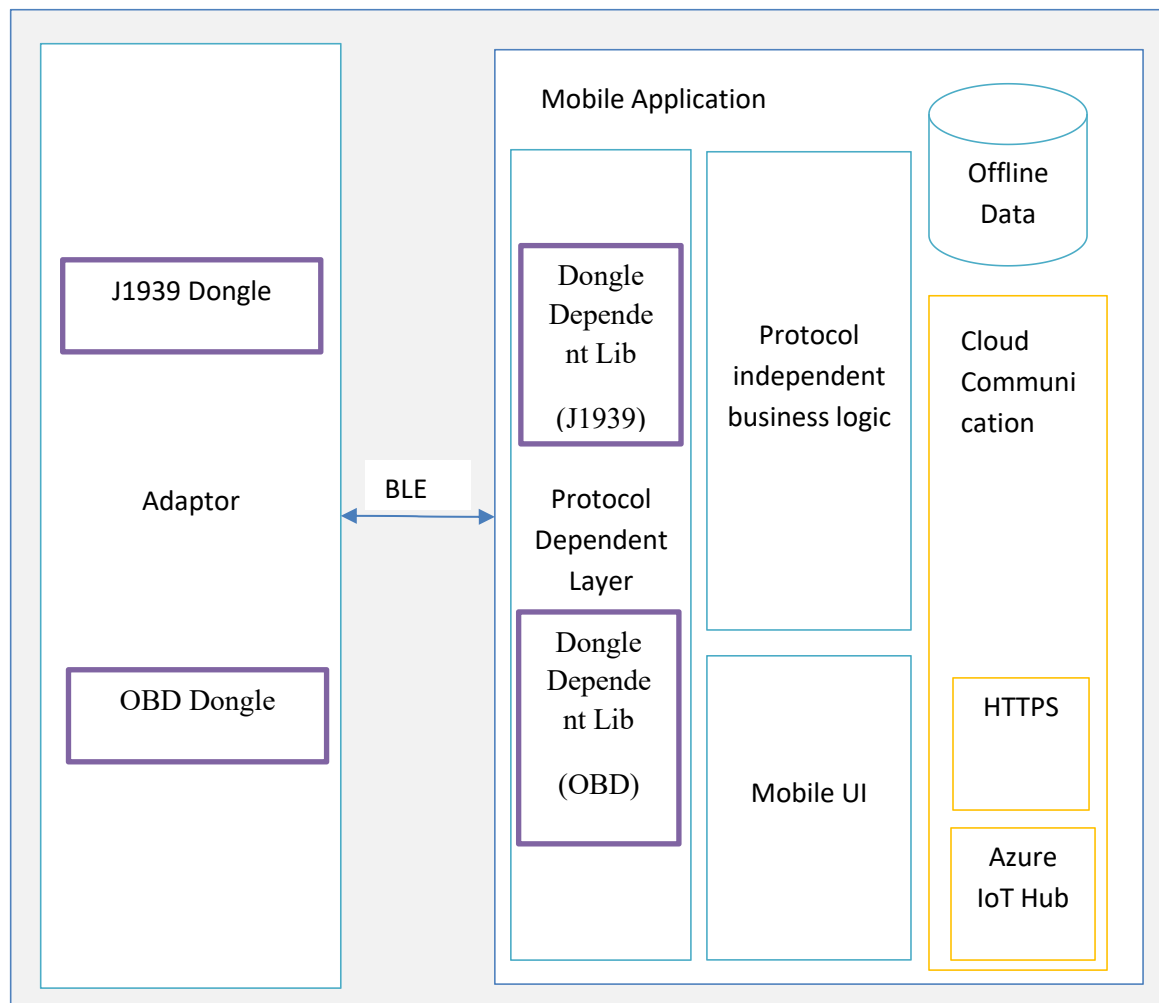
### 2.1. Mobile App Components

Mobile application shall be developed using Android Native Platform. This mobile application shall support Multi-Layer architecture for easy protocol integration.

Mobile application communicates with Dongle using Bluetooth. Mobile application logic is divided in following layers.

- Protocol Dependent Layer
- Protocol independent layer i.e. Business Logic Layer.
- Cloud Communication Layer
- Mobile UI
- Internal Offline Storage

Following Diagram shall provide internal details of the architecture.



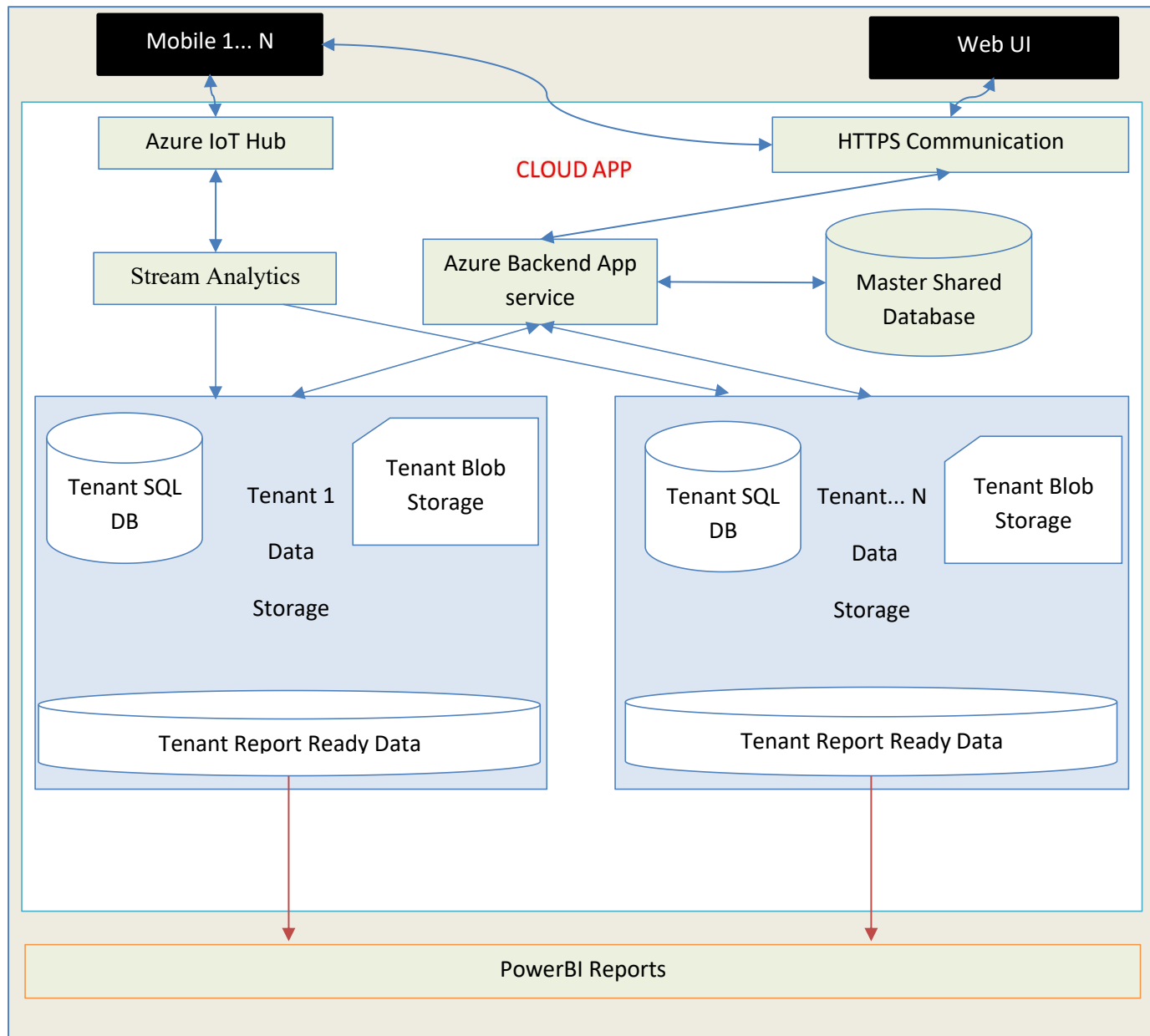
- Dongle will communicate with Protocol dependent layer. This will be a configurable interface to support more than 1 dongle. Protocol dependent layer shall encapsulate protocol/Dongle specific implementation details.
- Protocol Independent business logic is intermediate layer which will communicate with Protocol dependent layer, Local Database, Cloud API and Mobile UI.
- Maintaining local database to track trips and vehicle information. This data will be getting synced with server on Azure.

Mobile application shall be developed using,

- Android Native Platform
- SQLite Database
- Azure IoT client

## 2.2. Cloud Application Components

Cloud application uses shared multi-tenant architecture; where Tenant details and authorization details (roles and privileges) are stored in Master database; however Tenant specific information is stored inside separate Tenant specific databases. Following diagrams provide detailed view of different component used in cloud application.



There are two active users of cloud application.

1. Web application interface
2. Mobile application.

Web application shall use HTTPS based communication channels and invoke different REST APIs provided by Azure App Service (backend services). Azure App Service shall hold the backend business logic part.

Web application (Fleet Management Portal) is also deployed as Azure App Service and shall be implemented using AngularJS and Bootstrap technologies. It will be a responsive web application and will be fitted to most of desktop/laptop resolution.

The Mobile application shall communicate with Backend Azure Service using two different ways,

1. Azure IoT Hub
2. HTTPS REST API

The Azure IoT hub is used to receive dongle/device data along with Mobile GPS location and forward it to backend Azure App service after every one minute (Send duration will be configurable). Azure backend service internally detects tenant and connect with desired tenant and store the tenant specific data in Tenant database.

The Tenant Specific Report Ready Azure SQL/Cosmos data will be consumed by PowerBI reports.

### 3. Intended Audience

This guide is intended for the users who want to use the system. The guide explains different flows for following user roles:

#### *a. Super Admin*

- By default, super admin user is created in system.
- Super admin can manage user accounts.

#### *b. Tenant Admin*

- Tenant admin is added by super admin.
- Each company which registers have one tenant admin who can manage delegate users.
- Tenant admin can also manage fleets, vehicles and able to view reports.

#### *c. Fleet Admin*

- Fleet admin is added by tenant.
- Fleet admin can add, update or delete drivers.
- Fleet admin also be able to remove vehicles from his/her fleet.

#### *d. Driver*

- Drivers can only login to Android app and are not allowed to login on web portal.



## 4. Deploy Resources with ARMTemplate on azure

The following instructions would help the user to deploy bare minimum Azure resources required to get the solution up and running.

### Prerequisites:

The following prerequisites are required to get the Asset Monitoring and Tracking system up and running on a personal account.

1. Active Azure subscription.
2. Link to the ARM(Azure Resource Manager) template that would be used as a script to install the services.(<https://github.com/MobiliyaTechnologies/MobiliyaFleetARMTemplate.git>)
3. Read the Azure Resource Manager Naming Convention(<https://docs.microsoft.com/en-us/azure/architecture/best-practices/naming-conventions>)

A step by step series of examples that tell you how to get a development environment running

After achieving the prerequisites, the next step would be the installation of the ARM script. Following are the steps to do the same.

### Step 1

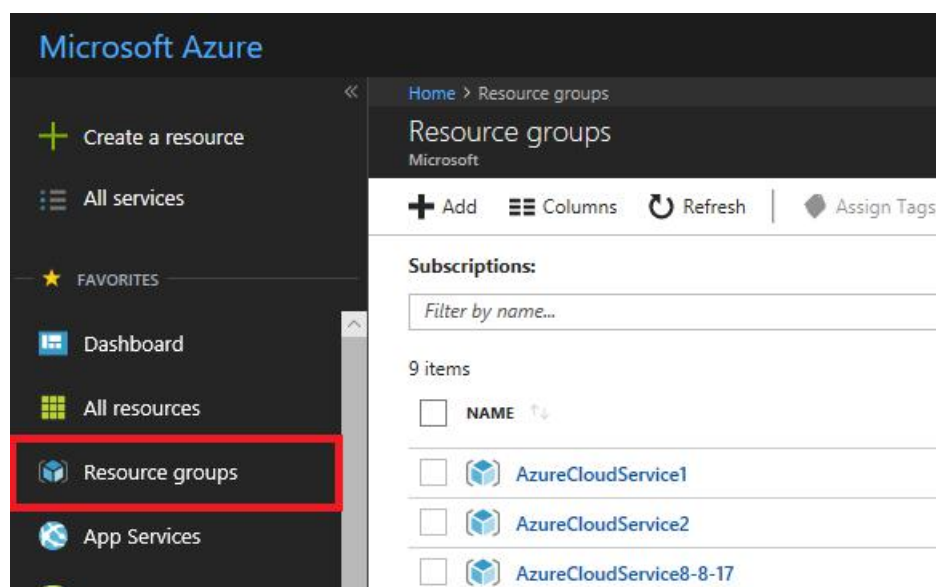
Clone the repository "MobiliyaFleetARMTemplate.git"

(<https://github.com/MobiliyaTechnologies/MobiliyaFleetARMTemplate.git>)

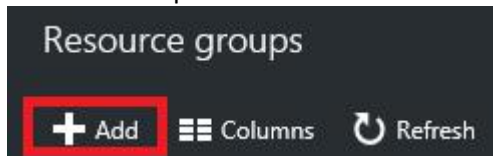
### Step 2

Login to the Azure portal (<https://portal.azure.com>) and select the appropriate subscription if it is not selected by default. Create a resource group that serves as the container for the deployed resources.

To create resource group, select Resource Groups.



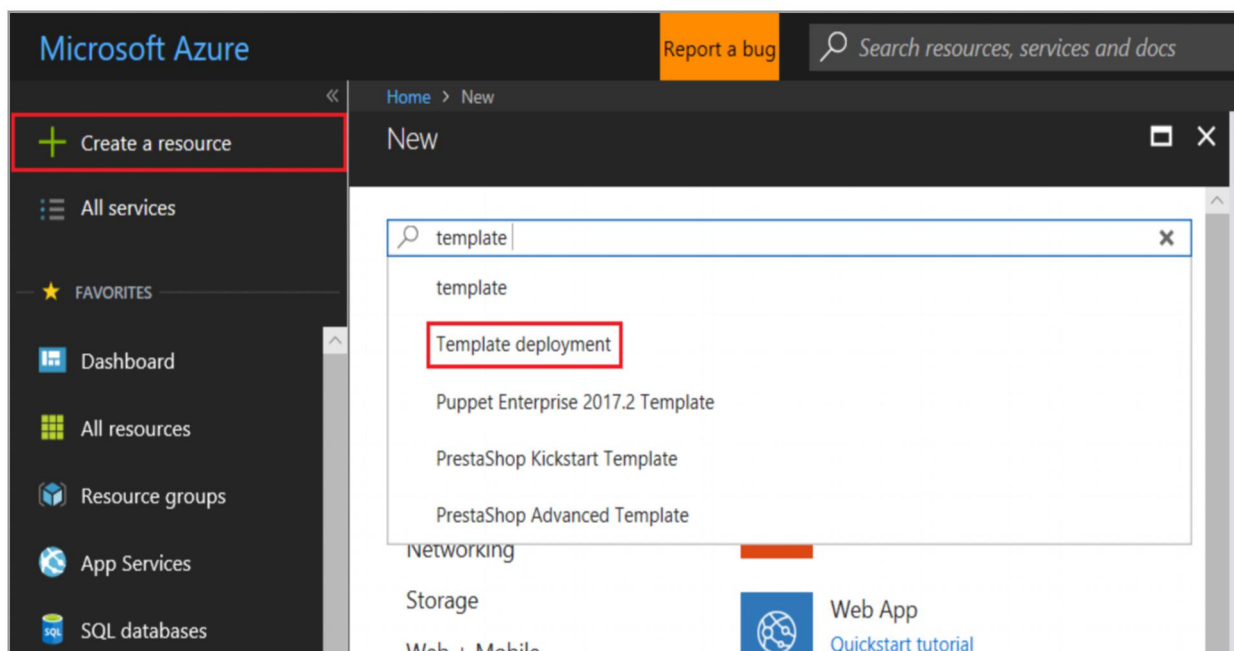
Click on add option



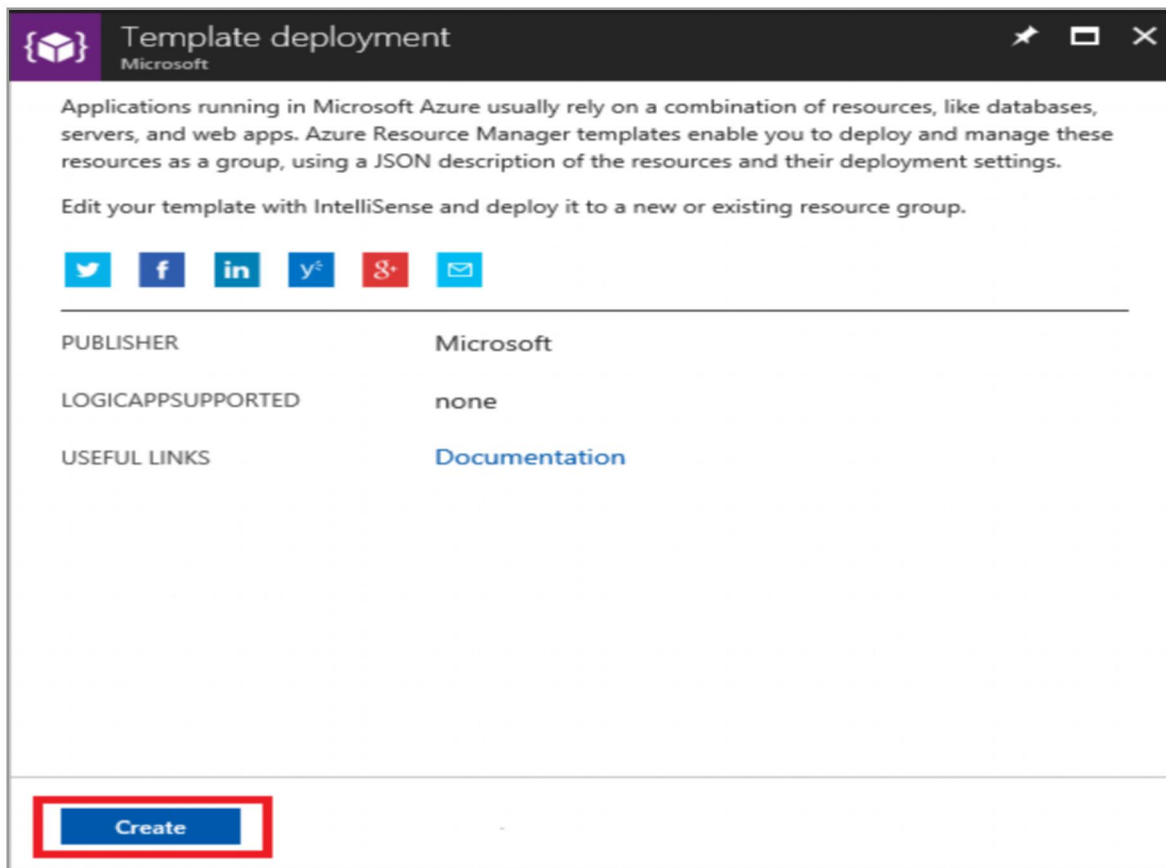
Provide a name and location for the new resource group. Select Create.

### Step 3

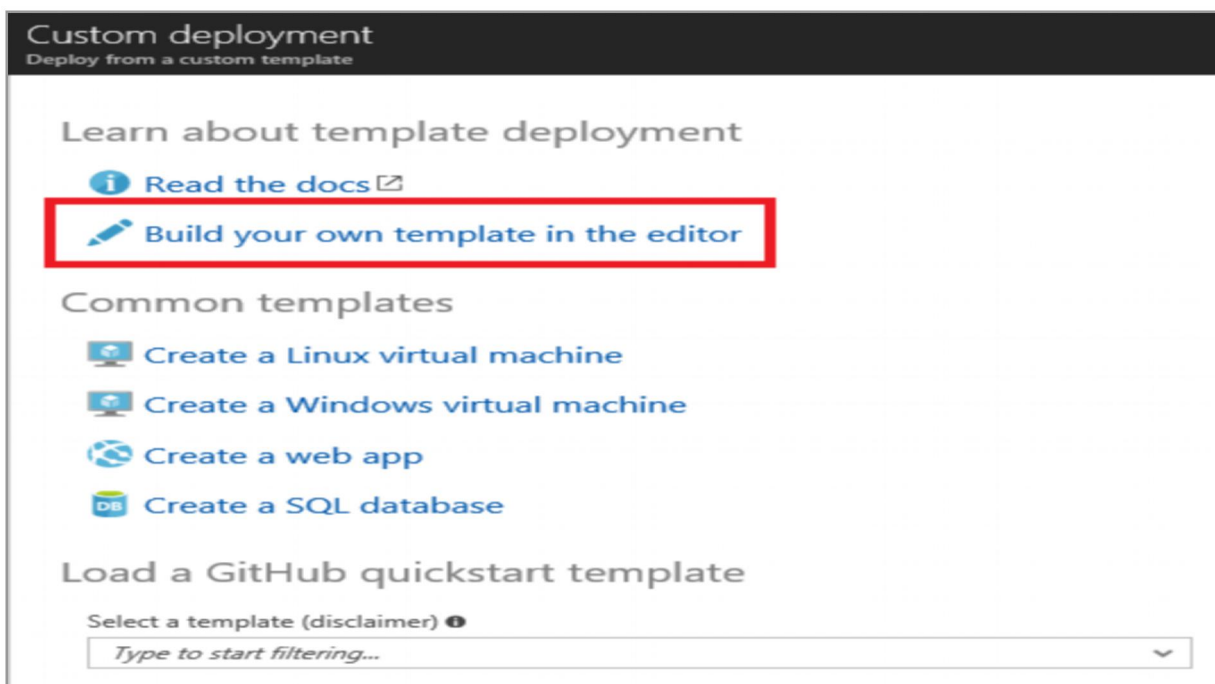
To deploy the template that defines the resources to the resource group ,create a resource and select Template deployment option and load the azuredeploy.json template from your cloned project directory( 'c:\users\..\projects').



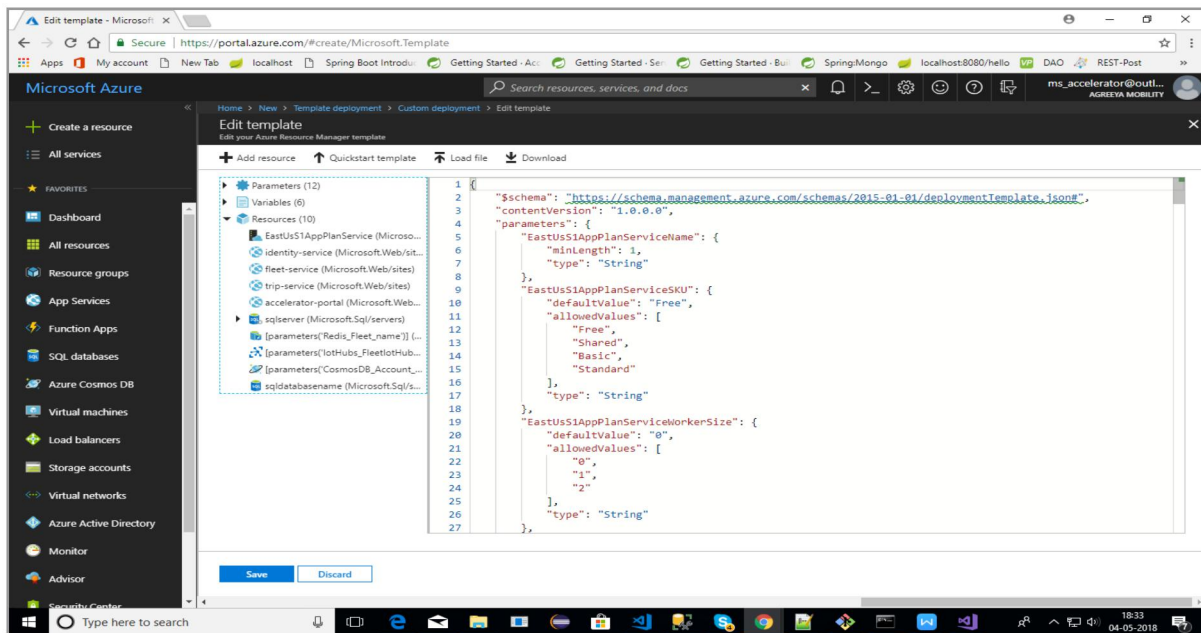
- Select Create



- Select build your own template in the editor.



- Click on load file. Upload deployment template file (azuredeploy.json) from your cloned project.



- Once the file is upload, a form appears which accepts some values before the actual deployment.

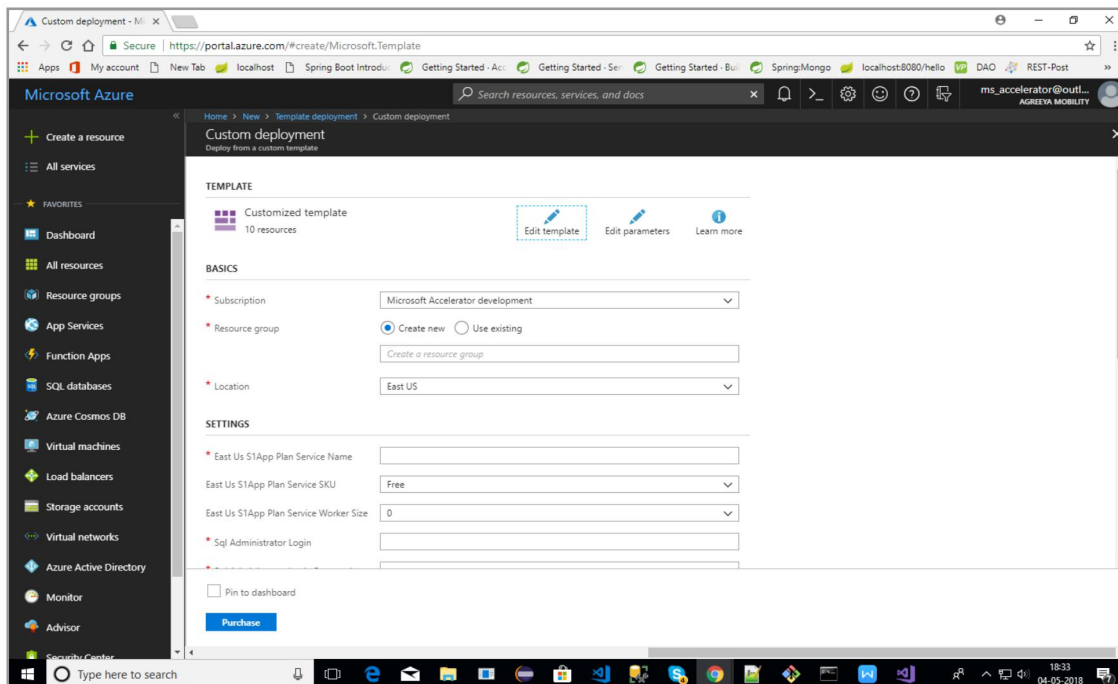
Fill in all the required details using the following guidelines.

Preferable use small case characters.

**\*\*\*Do Not\*\*** use special symbols.

\* Use the **\*\*i\*\*** image symbol as a guidance for filling in the data.

\* Give special preference to fields marked with **\*\*Has to be unique\*\***.



- \* Once, all the fields have been filled, accept the licensing terms and hit **\*\*Purchase\*\***.
- \* The deployment would take some time; you can have a cup of coffee till then.
- \* Once you get a notification of successful deployment of resources from Azure, you can proceed with the further deployment steps.

## 5. Deployment of Web App services on azure

Follow below steps for deployment of Web App

### Prerequisites:

Download Project from github

URL to the Repository:

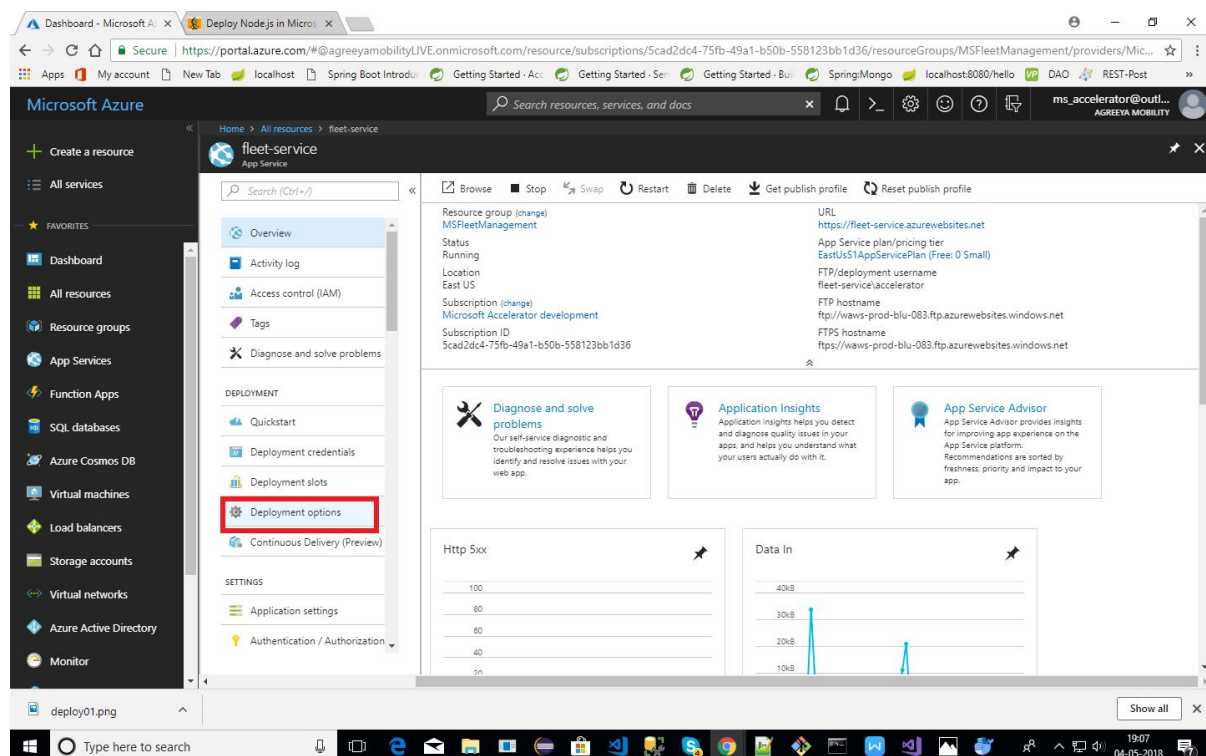
**Identity Service:** <https://github.com/MobiliyaTechnologies/identityService.git>

**Fleet Service:** <https://github.com/MobiliyaTechnologies/FleetService.git>

**Trip Service:** <https://github.com/MobiliyaTechnologies/TripService.git>

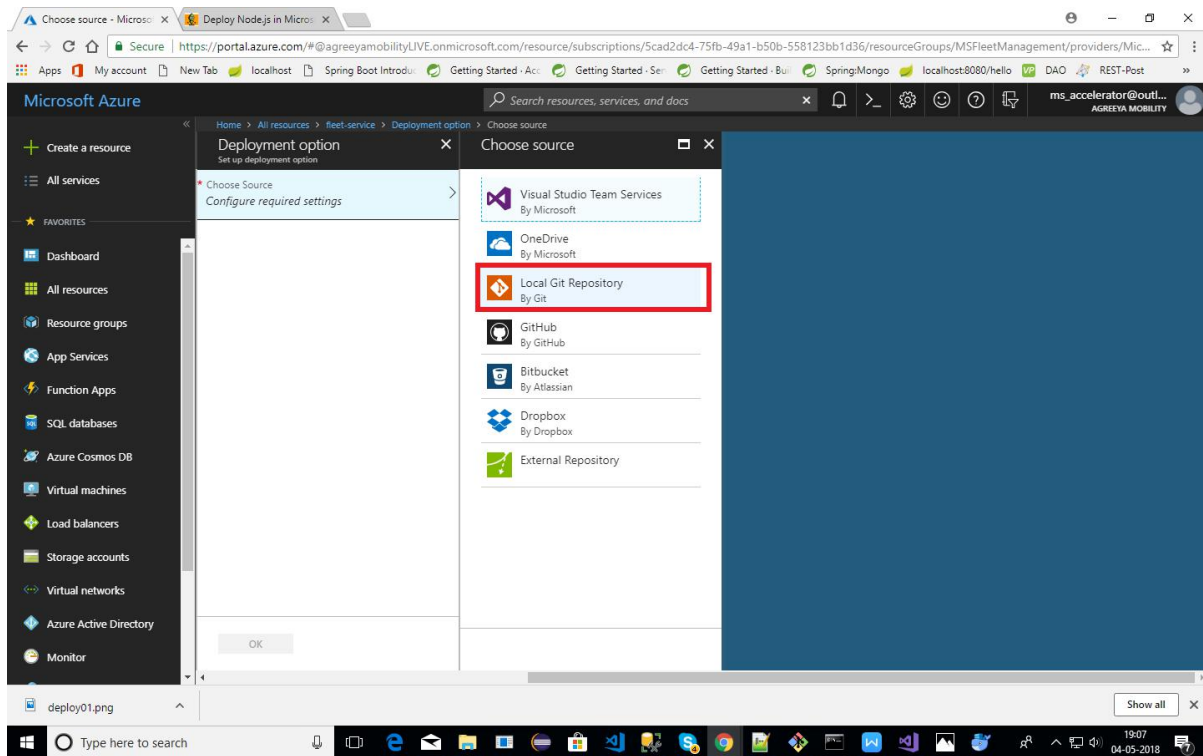
### Step 1

- Once resources are created, click on any app service (Identity/Fleet/Trip) and select Deployment Options.



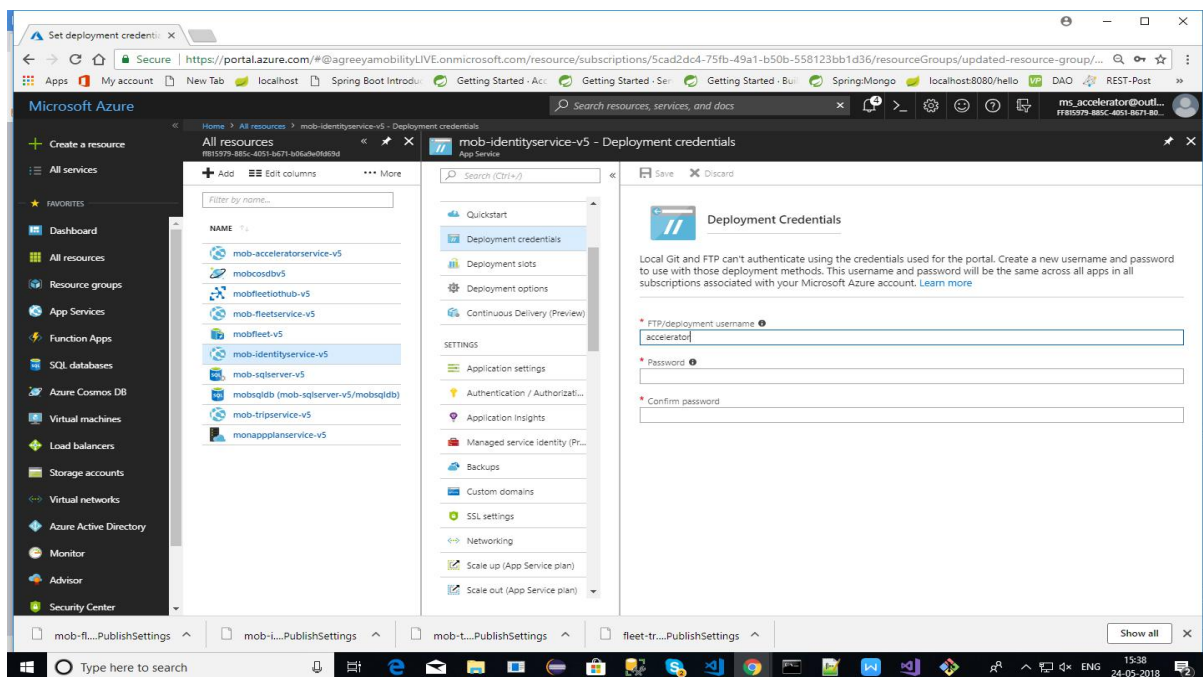
## Step 2

- Click on choose resource and select Local Git Repository option as shown below:



## Step 3

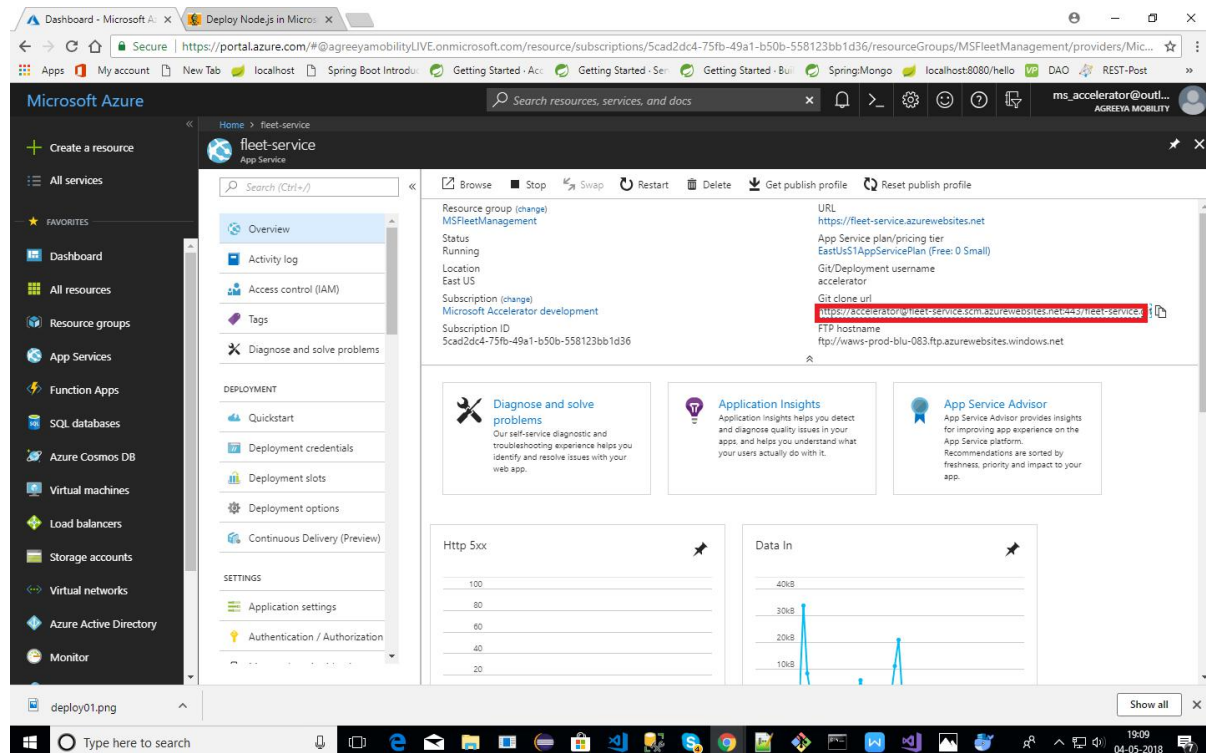
Set Deployment Credentials.





### Step 4

- Copy the git URL and clone the repository.



- Enter the URL for the Git repository, as displayed in the Azure Portal. . Enter your credentials (as explained in step 3)when prompted.

### Step 5

- Commit your files:
  1. On your computer, move the cloned project (IdentityService/FleetService/TripService) to upload to Azure git repository(as explained in step 4) that was created when you cloned the repository.
  2. Open Git Bash.
  3. Change the current working directory to Azure git repository.
  4. Stage the file for commit to Azure git repository using below command.

```
$ git status
```

```
$ git add .
```

5. Commit the files to Azure git repository using below command.

```
$ git commit -m "Add existing files".
```

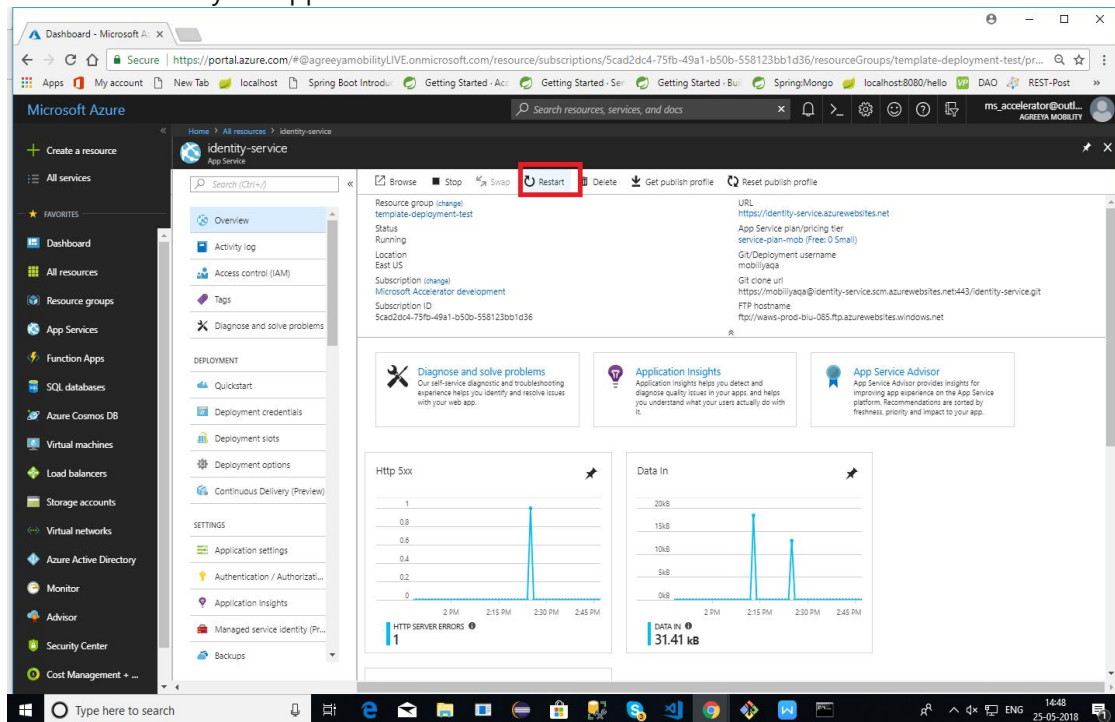
6. Push the changes to Azure git repository using below command.

```
$ git push origin master
```

The application would be deployed.

## Step 7

Restart and test your application on the web.



Now every time you push code to azure repo, site will get automatically deployed!

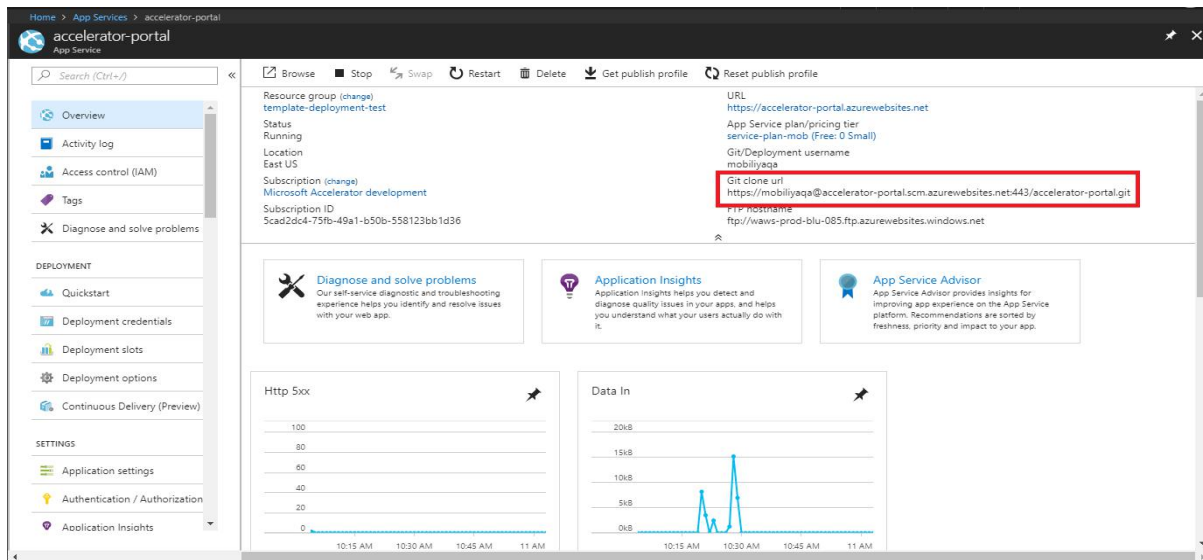


## 6. User Instructions

### 1. Web Portal

#### 1.1 Setup and Configuration changes

1. **Prerequisites:** You need to have Git and Node.js installed.
2. **Clone the repository :** You have to clone the repository "MobiliyaFleetWebPortal".  
(<https://github.com/MobiliyaTechnologies/MobiliyaFleetWebPortal.git>)
3. **Environment/config changes:** This step assumes that you have deployed backend services and those are up and running.  
You need to change configuration URLs in 'environment.ts' file (located at /src/environments/environment.ts), change SERVICE\_URL values.  
USER(line no 5) : set identity service url  
FLEET(line no 6): set fleet service url  
TRIP(line no 7): set trip service url
4. **Build and Run:**
  - i. Install the npm packages described in package.json  
`npm install`
  - ii. You can run the application using below command  
`ng serve`  
Application default runs on 4200 port. Open <http://localhost:4200> in browser.
  - lii. Once application tested locally, you are ready to deploy your application.
  - iv. Build the application using below command(Generates dist folder)  
`ng build`
5. **Deploy:**
  - i. Follow **step 1** and **step 2** from section **5. Deployment of web app services in azure** , to set azure Local Git Repository.
  - ii. Clone the local git repository of azure app service(accelerator-portal) using url as shown in below image.



li. Go to source folder (cloned in step 2). Copy files from **dist** folder into newly cloned repository in step i.

lii. You need to push your code using below commands.

A. *git add .*

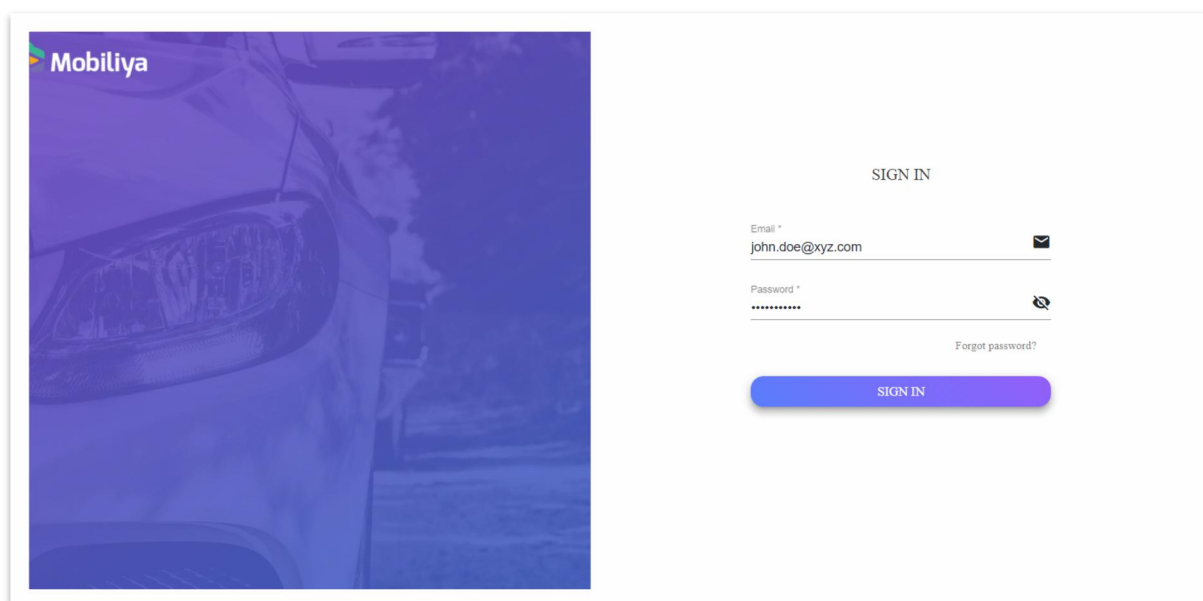
B. *git commit -m "message-string"*

C. *git push origin master*

## 1.2 Super Admin

### Login

Super admin account is created by default. You need to open accelerator portal URL in your browser. And enter your username and password. And click on Sign In button.



### Add User

On login super admin is directed to user-management page. Super admin can add delegate Users. You can click on add User icon. And add tenant admin.

The screenshot shows the 'Add New User' form in the Mobiliya Fleet Management interface. On the left, there is a sidebar with 'Users', 'Devices', and 'Reports' sections. The 'All users' list is visible, showing various user roles like Tenant Admin, Fleet admin, and Driver. The main form area is titled 'Add New User' and contains fields for Name, Email Id, Role Name, Mobile Number, and Company Name. The 'Add' button is highlighted in blue.

Name *	Email Id *	Role Name *
Tenant Admin	tenant.admin@test.com	Tenant admin

Mobile Number *	Company Name *
9576954654	Test

### Devices

Super admin can add, edit or delete devices. Devices are nothing but dongle which is associated with vehicle.

### Reports

Super admin can view reports. Reports are generated in powerbi. Reports are based on vehicle that is selected. Reports include maximum of one week data where date can be selected.

## 1.3 Tenant Admin

### Login

Tenant Admin can login with the credentials used during sign up. He needs to open accelerator portal URL in your browser. And enter your username and password. And click on Sign In button.

The screenshot shows the Mobiliya login page. On the left, there is a blue-tinted image of a car. On the right, there is a 'SIGN IN' form with fields for Email and Password. The 'Email' field contains 'john.doe@xyz.com'. The 'Password' field is masked with dots. There is a 'Forgot password?' link and a 'SIGN IN' button.

**SIGN IN**

Email \*  
john.doe@xyz.com

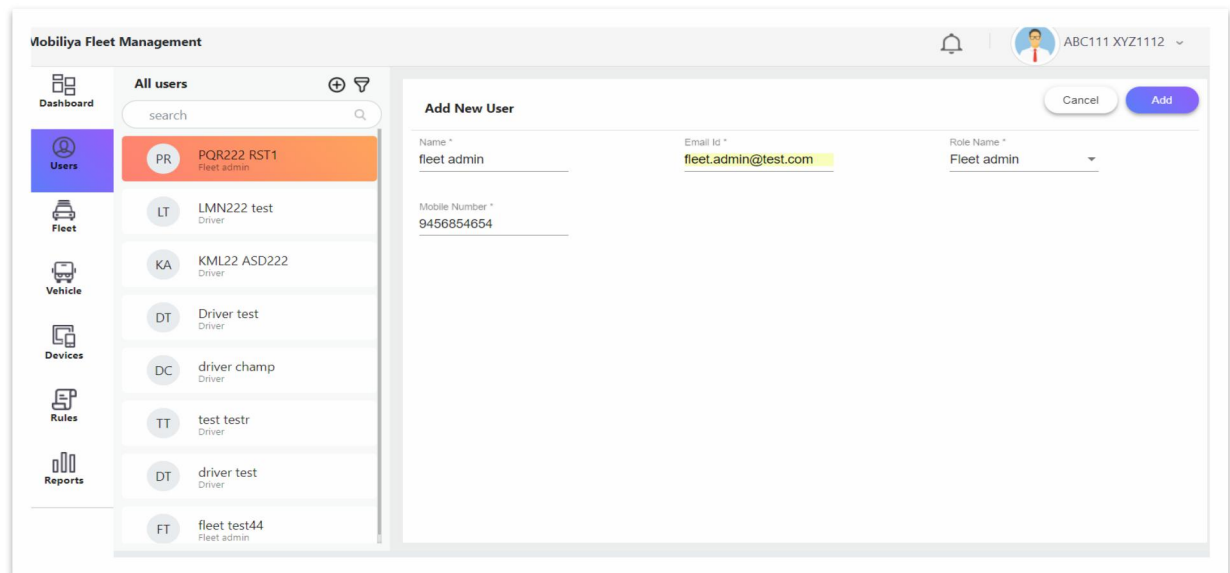
Password \*  
\*\*\*\*\*

[Forgot password?](#)

**SIGN IN**

### Add User

On login tenant admin is directed to dashboard page. Tenant admin can add delegate Users. You can click on add User icon. And add fleet admin.



### Fleet

Tenant admin can add, edit or delete fleets.

### Vehicles

Tenant admin can add, edit or delete vehicles. This section also contains trip information of vehicles.

### Devices

Super admin can add, edit or delete devices. Devices are nothing but dongle which is associated with vehicle.

### Reports

Tenant admin can view reports. Reports are generated in powerbi. Reports are based on vehicle that is selected. Reports include maximum of one week data where date can be selected.

### Rules

Tenant admin can create speeding or Geofence rules and assign it to drivers within a fleet.

## 2. Android Application

### *Configuration changes*

When IOTHub deployment is done then user has to change connection string in the android code file Constants.java, line number 14 "IOT\_CONNSTRING".

When URL for REST call changes, user has to change below strings from Constants.java file

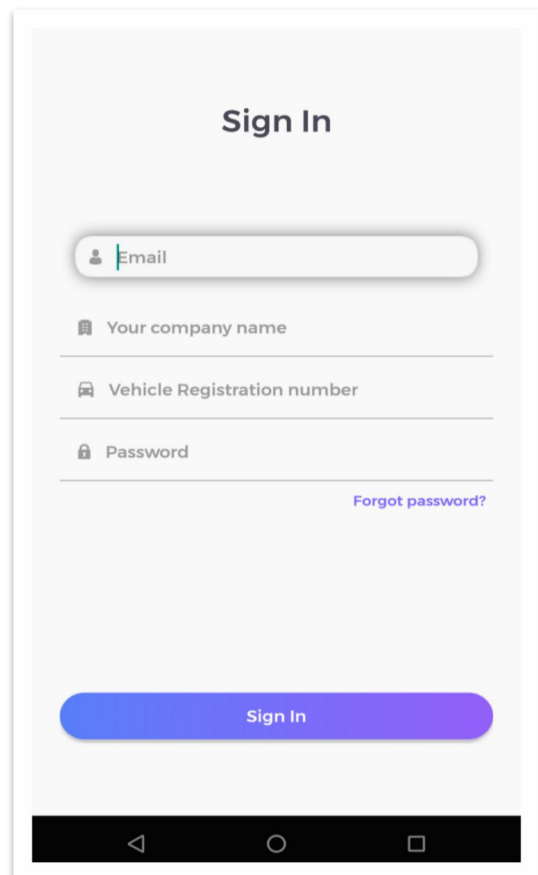
"IDENTITY\_DEV\_URL" Line no 24,

"FLEET\_DEV\_URL", Line no 25,

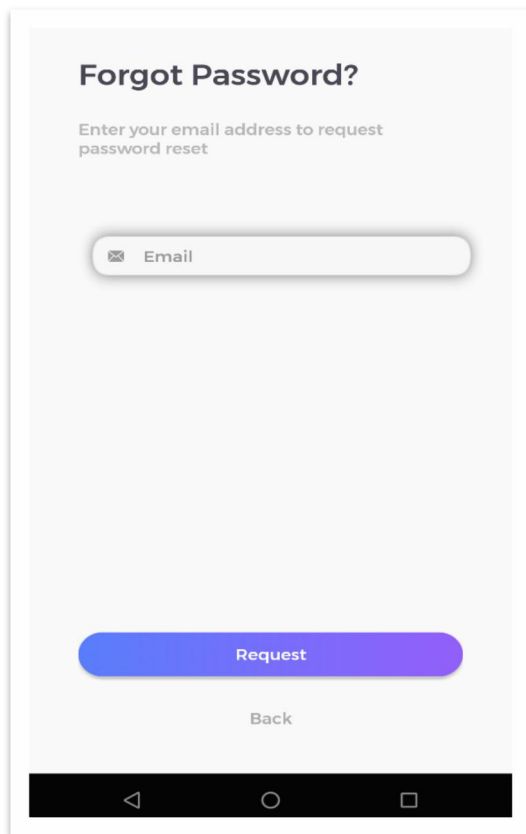
"TRIP\_DEV\_URL" Line no 30

### *Driver Sign In*

User can Sign In from here. User can change a password using Forgot Password option.



The screenshot displays the 'Sign In' screen of the Mobiliya Fleet Android application. The screen has a light gray background. At the top, the title 'Sign In' is centered in a bold, dark gray font. Below the title, there are four input fields, each with a small icon on the left and a label on the right: 1. An email field with a person icon and the label 'Email'. 2. A company name field with a building icon and the label 'Your company name'. 3. A vehicle registration number field with a car icon and the label 'Vehicle Registration number'. 4. A password field with a lock icon and the label 'Password'. To the right of the password field, there is a link labeled 'Forgot password?' in a smaller, purple font. At the bottom of the form area, there is a large, rounded rectangular button with a blue-to-purple gradient, labeled 'Sign In' in white text. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps icons.



### *Forgot Password*

The Forgot Password option can be accessed from the Sign In Screen. User can change password from here in case user cannot recall the existing password.

### *Dashboard*

After successful login and connection with dongle, driver is redirected to dashboard where he can manage trips and view his own ratings, vehicle health.

