

Создание сервисов на облаке и их настройка

Создать свой сервис достаточно легко, для этого нужно нажать **New** в панели управления **Azure** и выбрать **Mobile App** из списка сервисов.

App name - укажите уникальное название своему сервису (в нашем случае **myserviceexample**)

Resource Group необходимо выбрать существующий или указать новый. В рамках группы ресурсов (resource groups) можно создавать нужные группы приложений, сервисов и баз данных.

Желательно поставить галочку **Pin to dashboard** чтобы на главной странице появилась ссылка на наш сервис.

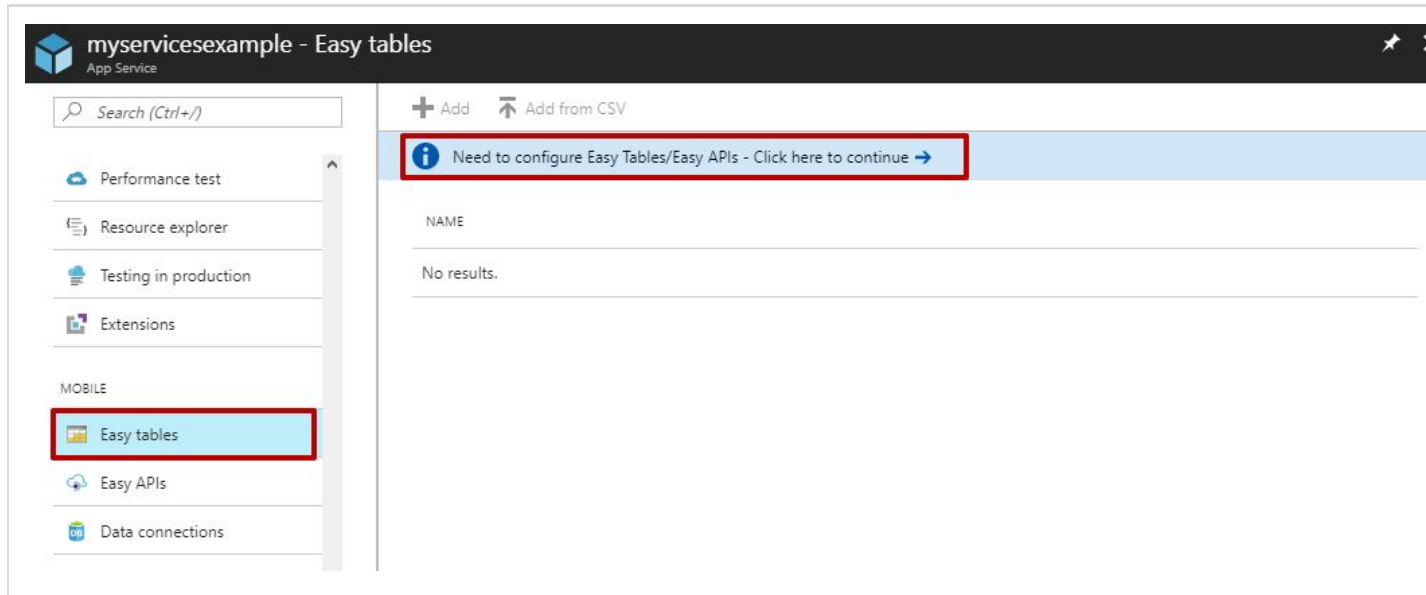
Нажимаем **Create**

Сохранение игрового прогресса из Unity в облако.



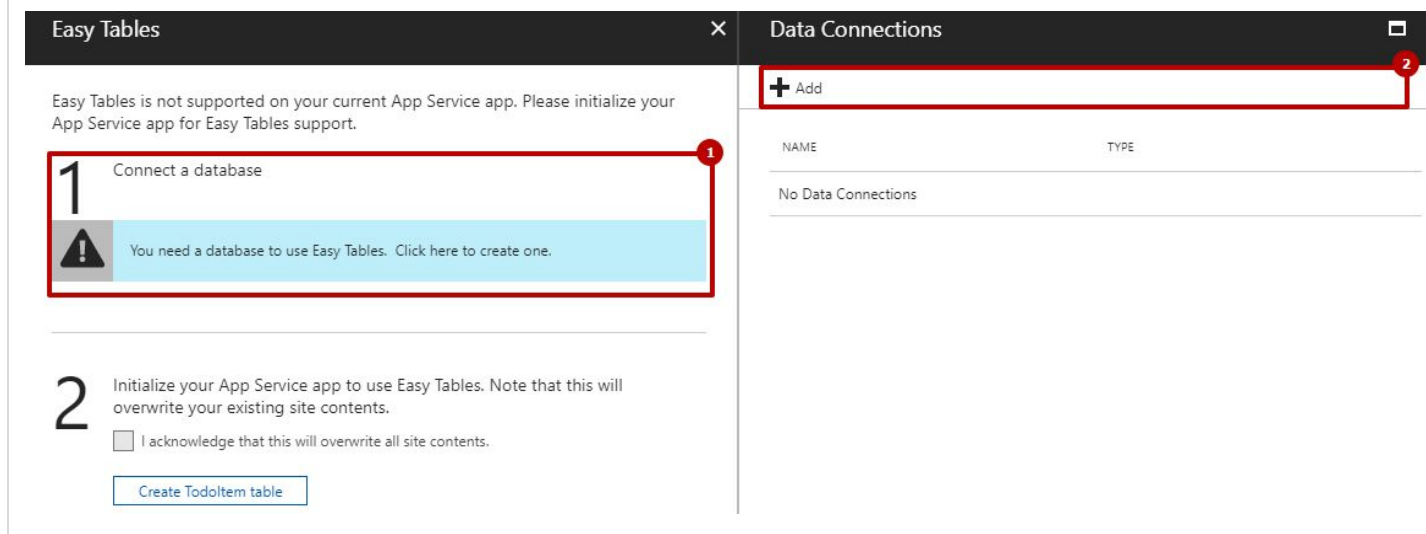
<p><input checked="" type="checkbox"/> Pin to dashboard 6</p> <p>Create 7 Automation options</p>	
<p>The screenshot shows the Azure portal dashboard. At the top, there's a 'Pin to dashboard' checkbox with a red circle containing the number 6. Below it, the 'Create' button is highlighted with a red circle and the number 7, followed by 'Automation options'. The main dashboard area has a 'No resources to display' message on the left. In the center, there's a 'Quickstart tutorials' section with tiles for 'Windows Virtual Machines' and 'Linux Virtual Machines'. On the right, there's a 'Deploying Mobile App' tile, which is highlighted with a red rectangle. A notification bubble at the top right says 'Deployment in progress...' with a close button and the time '12:56 AM'.</p>	<p>После нажатия кнопки Create начнется процесс развертывания сервиса на облаке.</p> <p>После окончания этого процесса сервис будет готов к использованию.</p>

Сохранение игрового прогресса из Unity в облако.



Для хранения данных нашей игры мы будем использовать **Easy Tables**. Для начала нам нужно быстро создать MS SQL базу данных в которой мы будем хранить наши таблицы.

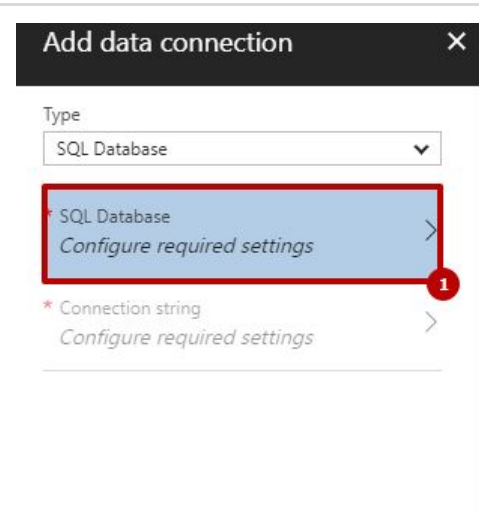
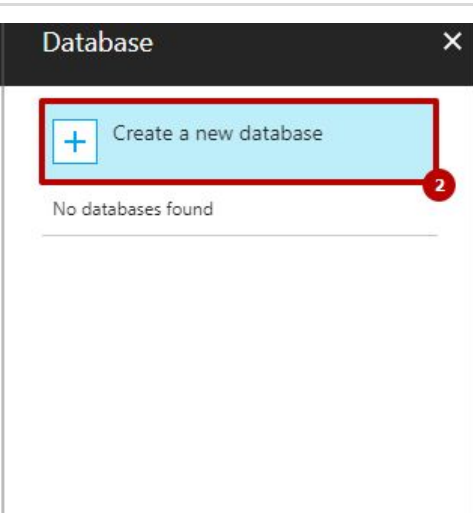
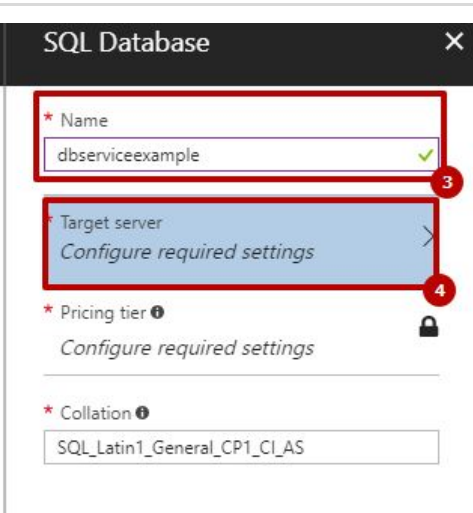
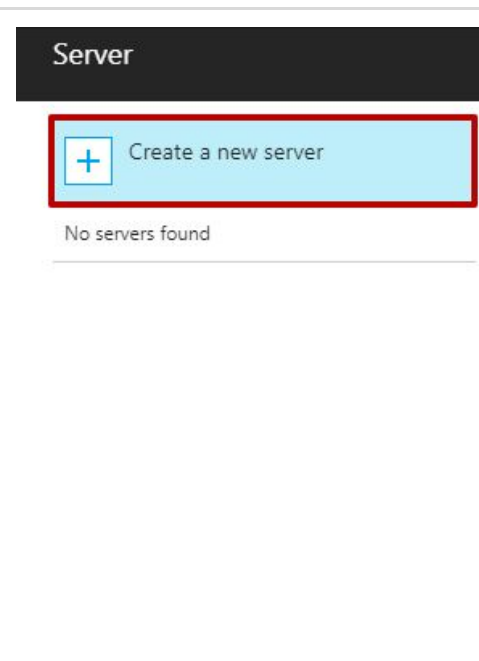
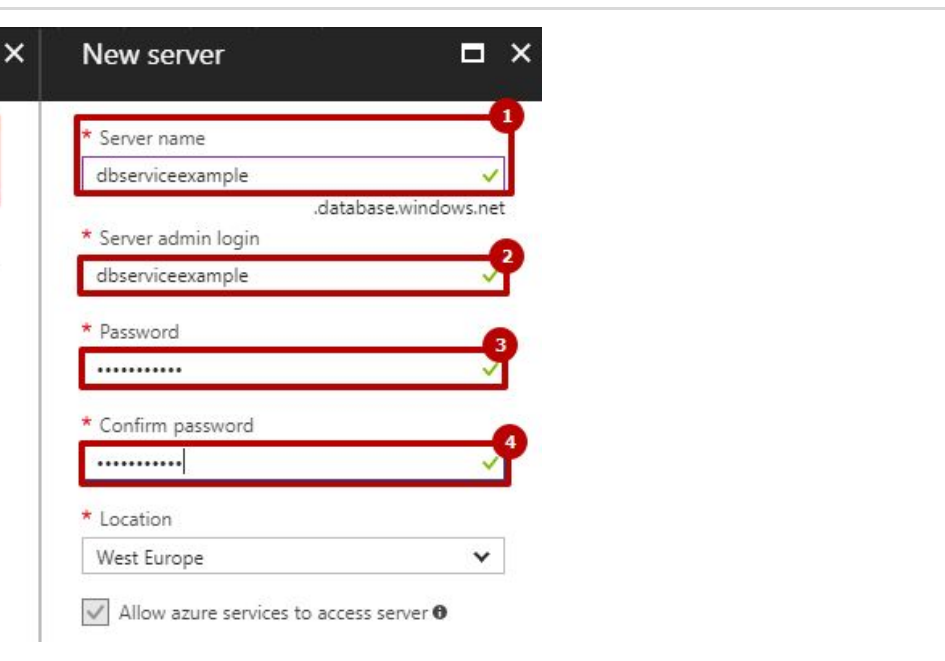
Выбираем в меню слева Easy Tables и потом в поле Click to continue нажимаем стрелочку вправо, как на скриншоте ниже.



Теперь продолжаем настройку наших таблиц, для пункте **1 Connect a database** нажимаем на знак восклицания и в окошке кнопку **Data Connections** кнопку **Add**.



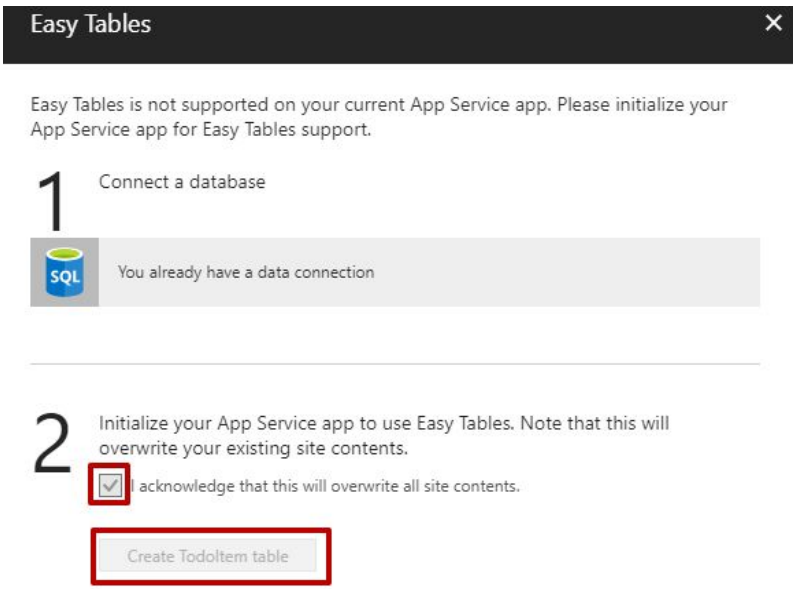
Сохранение игрового прогресса из Unity в облако.



			<p>Для того чтобы создать новое соединение нам нужна база данны. Именно ее созданием мы сейчас займемся. Нажимаем Create a new database затем вводим имя нашей базы данных. Важное поле Target server. По умолчанию у вас сервера не будет, потому дальше мы создадим именно его.</p>
		<p>Нажмите Create a new server для создания сервера.</p> <p>Server name может быть любым уникальным именем, затем идет Server admin login логин администратора (Его нужно не забыть, скопируйте в текстовый файл) и Password пароль для администратора (Его тоже нужно не забыть, скопируйте в текстовый файл) после этого выбираем место размещения сервера и нажимаем Select.</p>	

Сохранение игрового прогресса из Unity в облако.



	<p>После этого нам нужно подключиться к ново созданному MS SQL серверу. В окне Add data connection выбираем Connection String и после этого появится окошко Connection String. Помимо имени порой система просит ввести имя администратора (Его мы только что создавали) и пароль (Его мы тоже только что создавали и я надеюсь не забыли). Нажимаем кнопку Ok.</p>
	<p>Ждем пока появится соединение с базой данных.</p>
	<p>Теперь закончим настройку Easy Tables. Ставим галочку I acknowledge that this will overwrite all site contents и нажимаем кнопку Create TodoItem table.</p>

myserviceexample - Easy tables
App Service

Search (Ctrl+/)

- Performance test
- Resource explorer
- Testing in production
- Extensions

MOBILE

- Easy tables**

+ Add Add from CSV

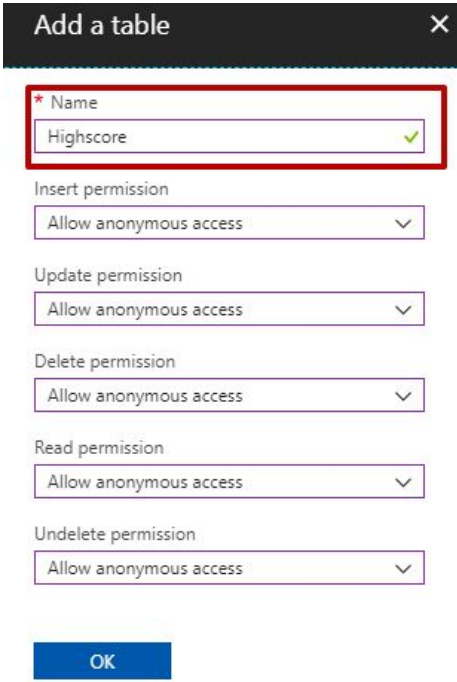
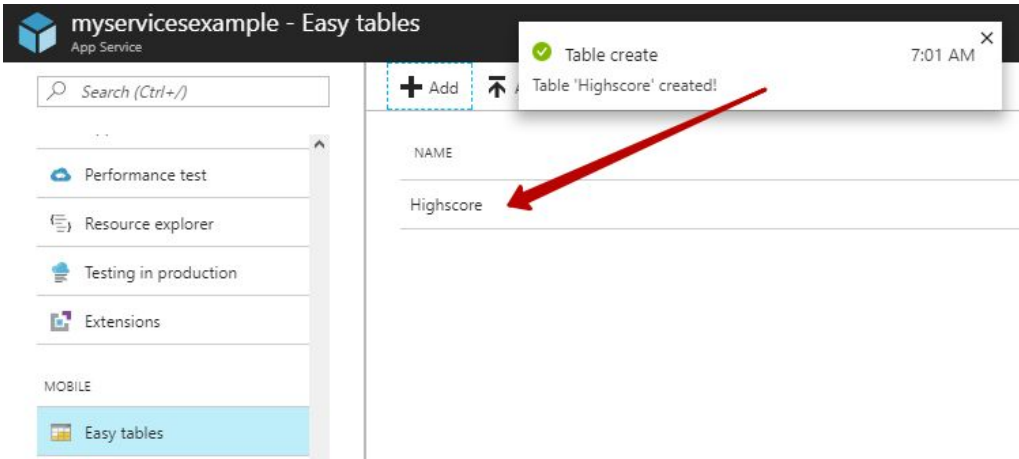
NAME

No results.

Теперь нам необходимо создать таблицу для хранения наших данных. Выбираем снова в меню **Easy tables** и нажимаем кнопку **Add**.

Сохранение игрового прогресса из Unity в облако.



	<p>У вас появится меню Add a table, во избежании путаницы давайте создадим таблицу с именем Highscore. Затем нажимаем кнопку Ok.</p>
	<p>После удачного создания таблицы вы увидите ее в списке.</p>

Сохранение игрового прогресса из Unity в облако.



The screenshot shows the Azure Data Explorer interface. On the left, the 'Highscore (0 records)' table is listed. The 'Schema' pane in the center shows the table's structure with columns: id (String, true), createdAt (Date, true), updatedAt (Date, false), version (Version, false), deleted (Boolean, false), playername (String, false), and score (Number, false). The 'Add a column' dialog is open on the right, showing the 'Column name' field with 'score' entered and the 'Data type' dropdown set to 'Number'. The 'OK' button is visible at the bottom of the dialog.

NAME	TYPE	IS INDEX
id	String	true
createdAt	Date	true
updatedAt	Date	false
version	Version	false
deleted	Boolean	false
playername	String	false
score	Number	false

Теперь нужно сконфигурировать нашу таблицу, добавив туда пару полей которые мы хотим в дальнейшем использовать. Для этого нажимаем **Manage Schema** справа появится окошко **Schema**. В нем можно увидеть стандартные поля, которые по умолчанию были созданы в таблице. И теперь нам необходимо добавить два новых поля. Нажимаем **Add a column**. Первое поле которое мы добавим называется **playername** типа **string** а второе поле **score** типа **number**.

The screenshot shows the Unity 2017.2.0f3 interface. The 'Projects' tab is selected, and the 'New' button is highlighted with a red box. The 'Learn' tab is also visible. The 'On Disk' and 'In the Cloud' sections are shown on the left.

Подключение сервисов к чистому Unity проекту

Теперь займемся подключением нашего сервиса к юнити проекту. Для этого нам необходимо открыть Unity и создать новый проект.

В данном проекте мы использовали версию Unity 2017.2.0f3

Сохранение игрового прогресса из Unity в облако.



Unity 2017.2.0f3

Projects Learn

New Open My Account

Project name

Data Saving Stan

Location

C:\Users\Stanislav\Desktop

3D 2D Add Asset Package

OFF Enable Unity Analytics

Cancel Create project

> Data Saving Stan > Assets

Name	Date modified	Type	Size
AzureServicesForUnity	07.09.2017 8:46	File folder	
AssetStoreTools.meta	07.09.2017 8:46	META File	1 KB
AzureServicesForUnity.meta	07.09.2017 8:46	META File	1 KB

1 - Project name - введите уникальное название вашего нового проекта.

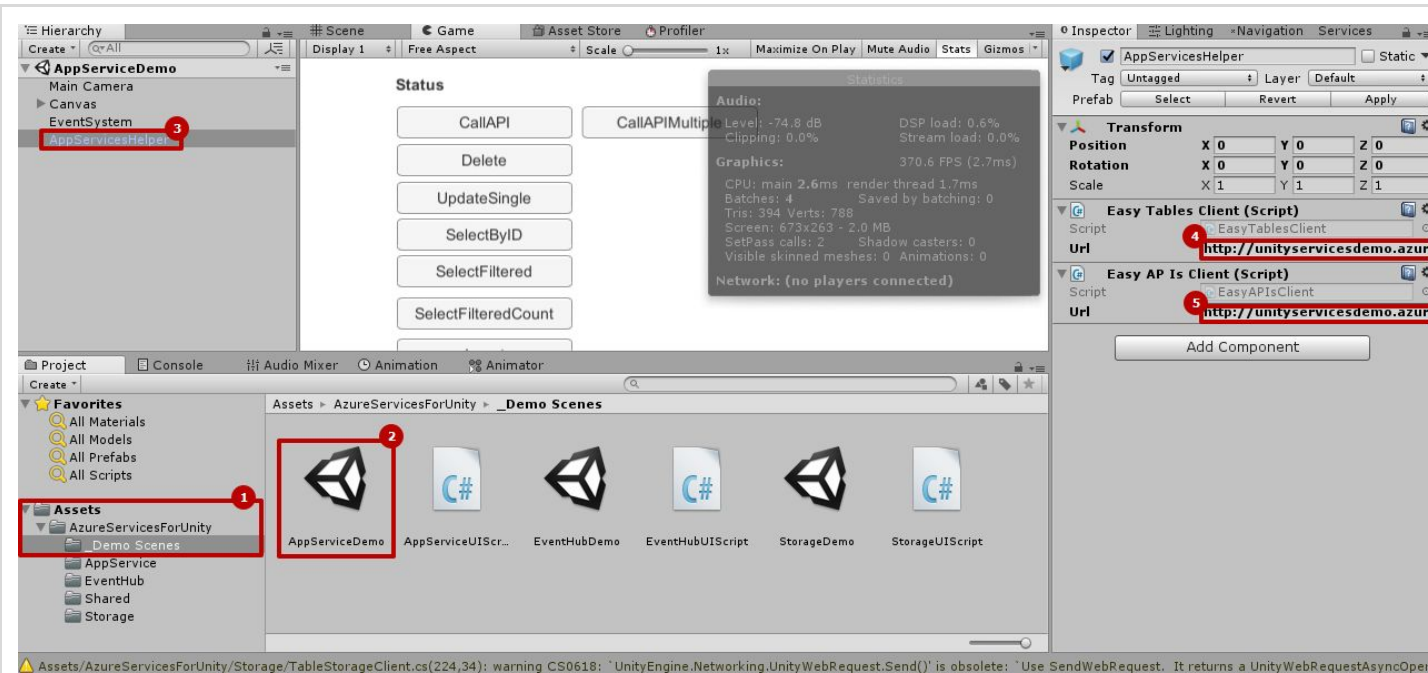
2 - Location - укажите директорию где будет храниться новый проект.

3 - Create project нажмите чтобы запустить создание нового проекта.

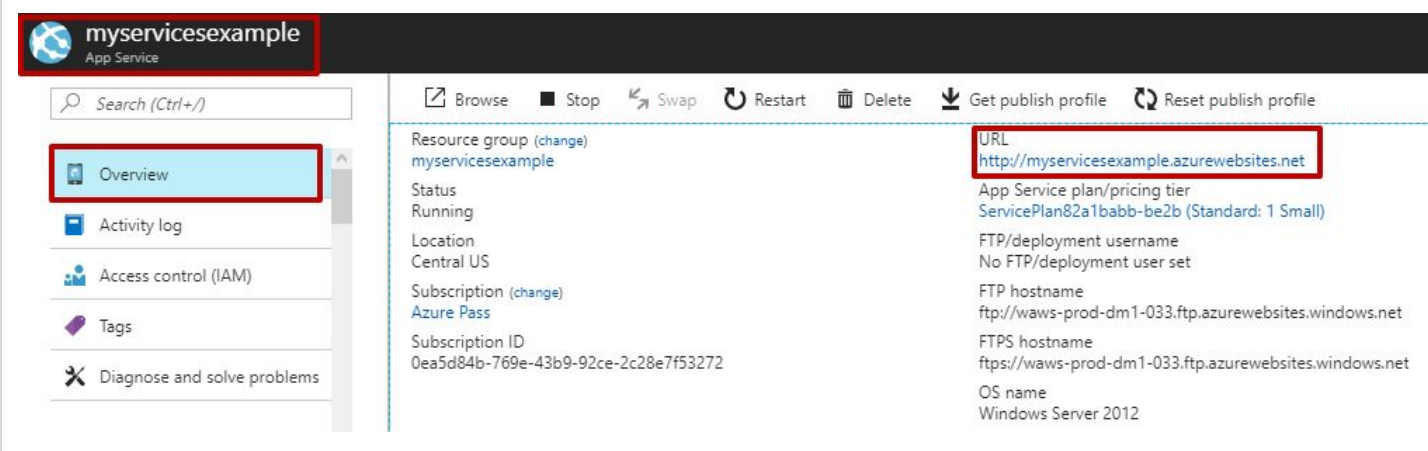
Теперь нам необходимо Git скачать ассет для работы с сервисами Azure в Unity, ссылка на ассет:
<https://github.com/dgkanatsios/AzureServicesForUnity>

Файлы из директории **Assets** из архива необходимо перенести в директорию **Assets** созданного вашего нового Unity проекта.

Сохранение игрового прогресса из Unity в облако.



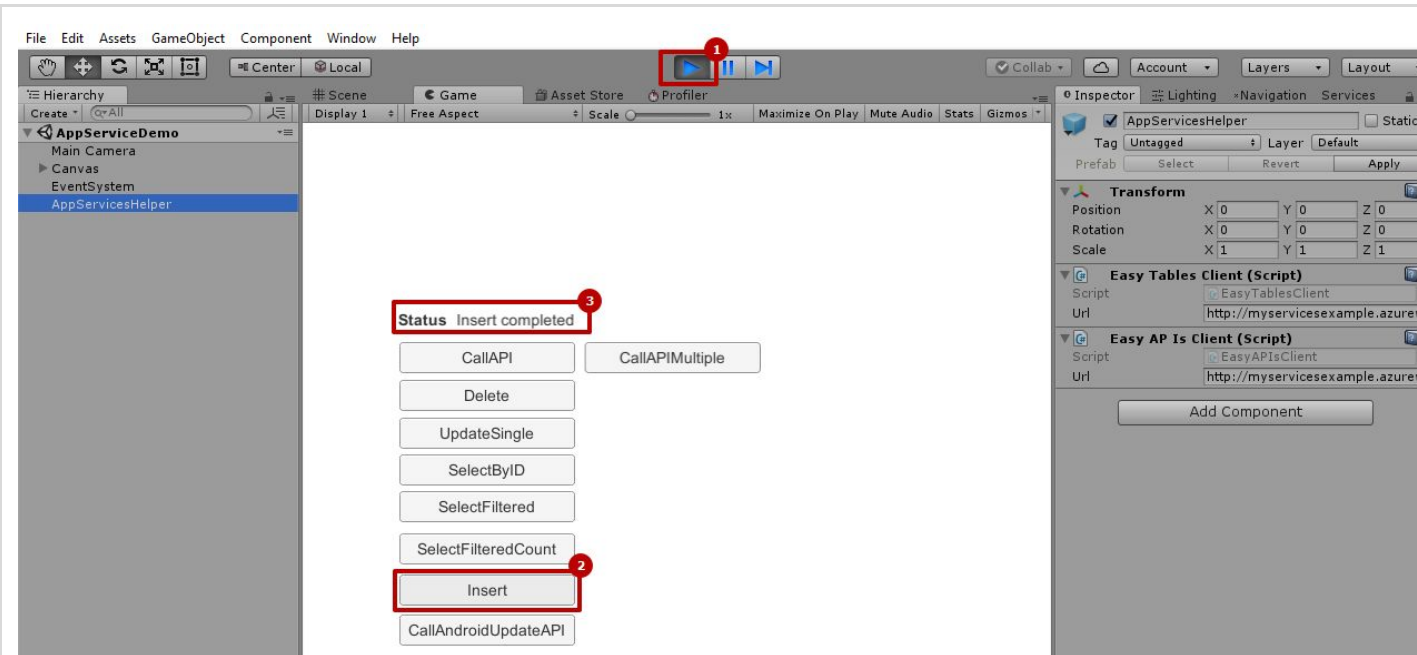
После добавления нами ассета в проект, его необходимо настроить. Открываем первую демо сцену с названием **"AppServiceDemo"**, как показано на скриншоте, затем выбираем GameObject с названием **"AppServiceManager"** и в инспекторе видим два скрипта для настроек. Нас интересует скрипт **EasyTablesClient**, в него вставляем путь к нашему сервису, который вы сможете взять из Azure, как показано ниже.



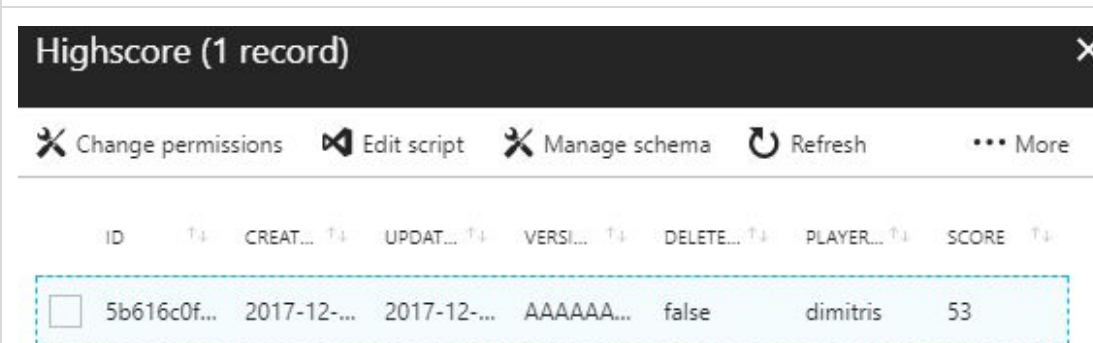
Путь к сервису вы можете взять обратившись к созданному ранее сервису **App Service** на вкладке **Overview**.

Важно: В данный момент мы работаем с демонстрационным проектом. Когда вы будите работать в своем игровом проекте с **Easy Tables** не забудьте на сцену поместить скрипт **EasyTablesClient**.

Сохранение игрового прогресса из Unity в облако.

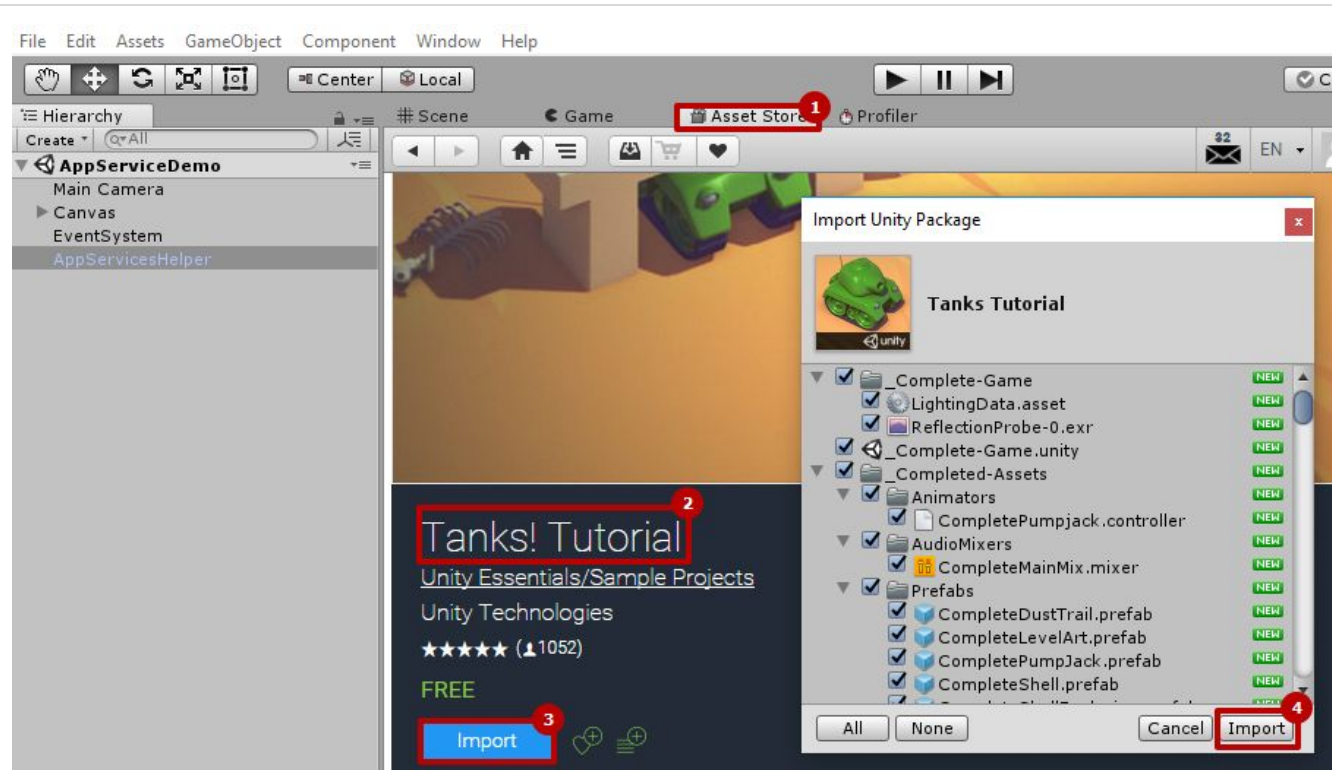


Давайте теперь запустим нашу сцену и посмотрим как она работает. После запуска нам необходимо нажать кнопку **Insert** и подождать пока статус загрузки не примет значение **Insert completed**.

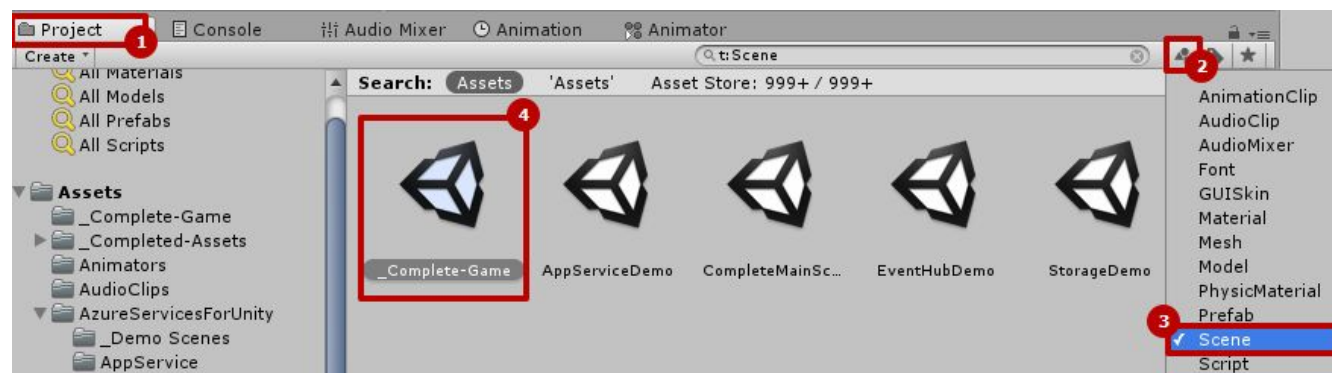


На облаке в нашей таблице **Highscore** должна будет появиться новая запись.

Сохранение игрового прогресса из Unity в облако.

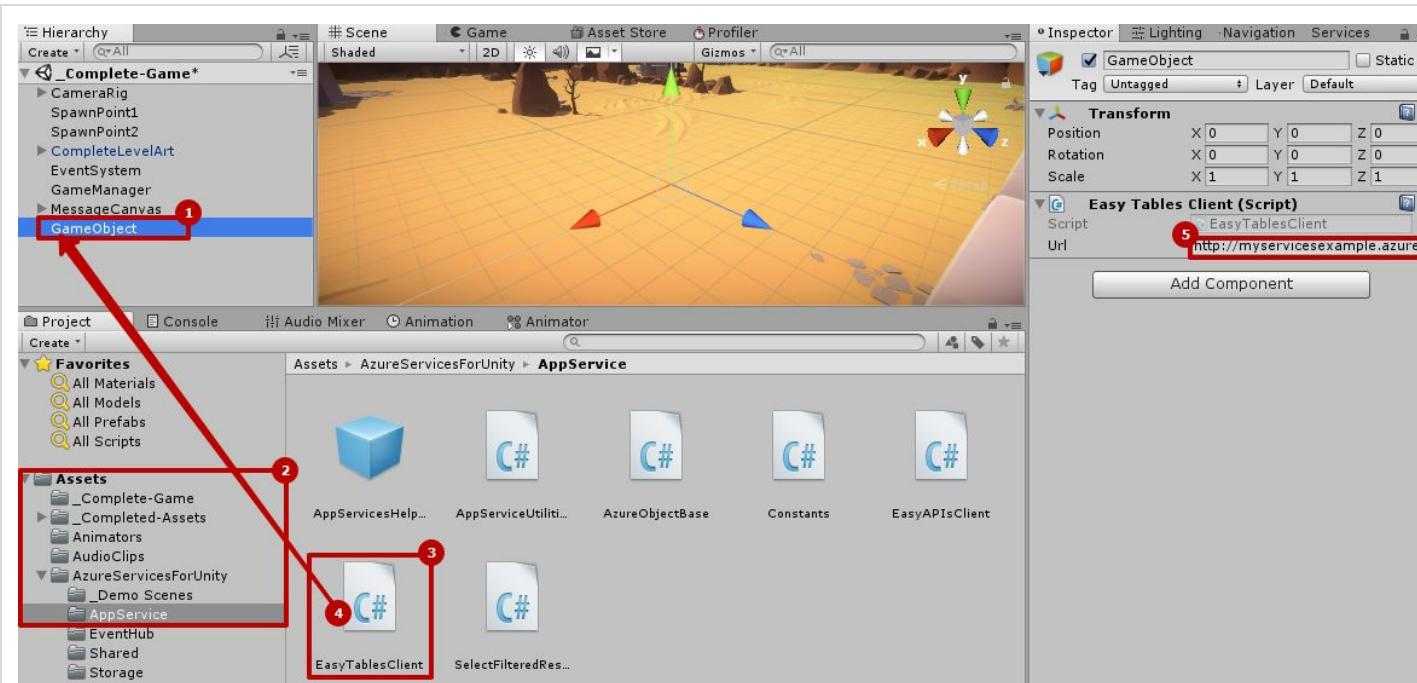


Отличный пример официальный бесплатный tutorial unity под названием **Tanks! Tutorial** который мы скачаем с **Unity Asset Store**. На скриншоте ниже показана последовательность действий, чтобы загрузить данную игру в наш проект. Сначала мы открываем окно **Asset Store** в нем в поиске ищем **Tanks! Tutorial**, у вас будет кнопка **Download** сначала, когда ассет скачается то появится кнопка **Import**. Затем нажимаете еще одну кнопку **Import** и подождите когда все загрузится в ваш проект.



Давайте откроем главную сцену нашего проекта, которая называется **"_Complete-Game"** Для этого Вам необходимо в окне **Project** открыть фильтр и выбрать **Scenes**.

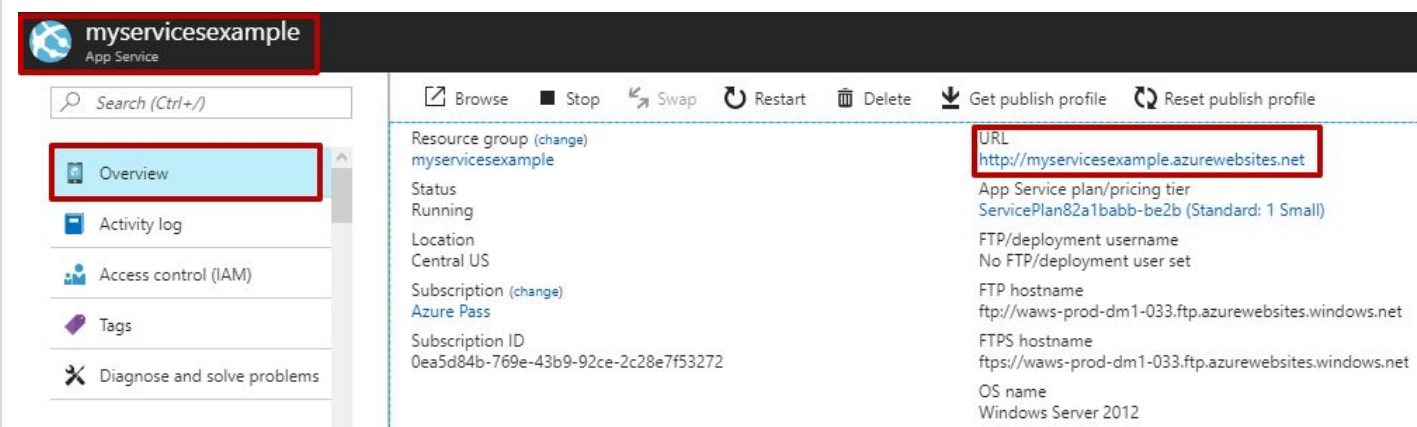
Сохранение игрового прогресса из Unity в облако.



Для того чтобы наш скрипт сохранения данных в Azure работал, создадим пустой **GameObject** нажав правой кнопкой мыши на пустой области окна **Hierarchy** в левой части **Unity**

Самое время переместить на него скрипт **EasyTablesClient** (вы сможете его быстро найти по аналогии поиска сцены, указав Scripts в качестве фильтра, либо найти его в дериктории **Assets \ AzureServicesForUnity \ AppService**)

И теперь вставим в строку URI нашего скрипта адрес нашего сервиса.



Адрес нашего сервиса не изменился и вы можете его скопировать по аналогии с предыдущим примером.

Сохранение игрового прогресса из Unity в облако.



```
GameManager.cs*
Data Saving Stan
Complete.GameManager
m_NumRoundsToWin

1 using System.Collections;
2 using UnityEngine;
3 using UnityEngine.SceneManagement;
4 using UnityEngine.UI;
5 using AzureServicesForUnity.AppService;
6

27 private void Start()
28 {
29     // Create the delays so they only have to be made once.
30     m_StartWait = new WaitForSeconds (m_StartDelay);
31     m_EndWait = new WaitForSeconds (m_EndDelay);
32
33     SpawnAllTanks();
34     SetCameraTargets();
35
36     // Once the tanks have been created and the camera is using them as targets, start the game.
37     StartCoroutine (GameLoop ());
38
39     //get the authentication token somehow...
40     //e.g. for facebook, check the Unity Facebook SDK at https://developers.facebook.com/docs/unity
41     EasyTablesClient.Instance.AuthenticationToken = "";
42
43
44 }
```

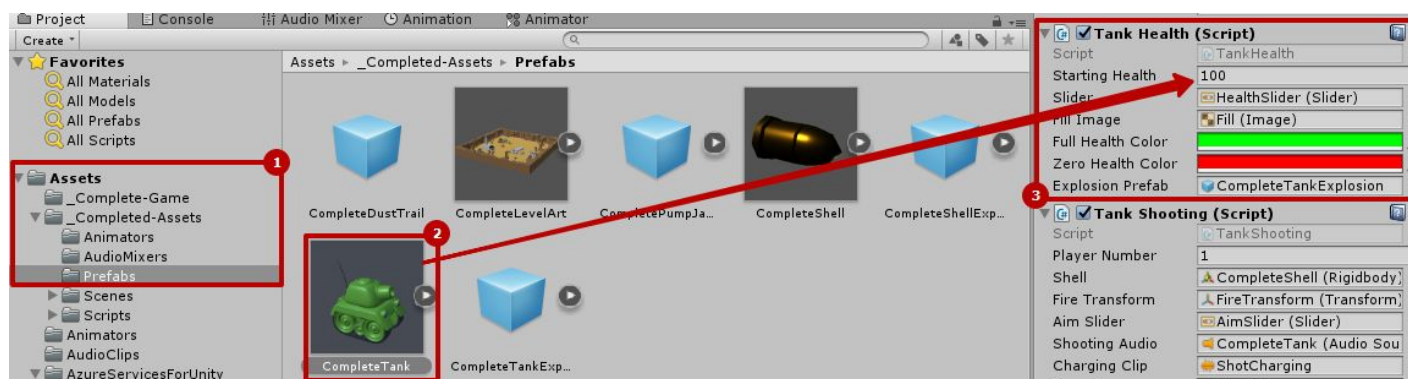
Давайте теперь перейдем в скрипт **GameManager** (он находится в директории **Assets \ _Completed-Assets \ Scripts \ Managers**) и добавим туда пару строк связанных с авторизацией.

Укажем пространство имен **using AzureServicesForUnity.AppService;**

В текущем примере это поле мы оставим пустым.

Код авторизации:

```
//get the authentication token somehow...
//e.g. for facebook, check the Unity Facebook SDK at
https://developers.facebook.com/docs/unity
EasyTablesClient.Instance.AuthenticationToken = "";
```



В нашем примере использования **EasyTables** мы будем в них писать победителя текущего раунда. На самом деле так можно записывать любые важные игровые данные такие как сохранения, настройки, данные профиля игрока. Для того чтобы понять какой из танков выиграл, его здоровье должно быть ноль, исходя из этого мы будем модифицировать скрипт **TankHealth**.

Сохранение игрового прогресса из Unity в облако.



```
TankHealth.cs | GameManager.cs
Data Saving Stan | Complete.TankHealth | m_StartingHealth

1 using UnityEngine;
2 using UnityEngine.UI;
3 using AzureServicesForUnity.AppService;
4 using AzureServicesForUnity.Shared;
5

74 private void OnDeath ()
75 {
76     // Set the flag so that this function is only called once.
77     m_Dead = true;
78
79     // Move the instantiated explosion prefab to the tank's position and turn it on.
80     m_ExplosionParticles.transform.position = transform.position;
81     m_ExplosionParticles.gameObject.SetActive (true);
82
83     // Play the particle system of the tank exploding.
84     m_ExplosionParticles.Play ();
85
86     // Play the tank explosion sound effect.
87     m_ExplosionAudio.Play();
88
89     // Turn the tank off.
90     gameObject.SetActive (false);
91
92     Send();
93 }

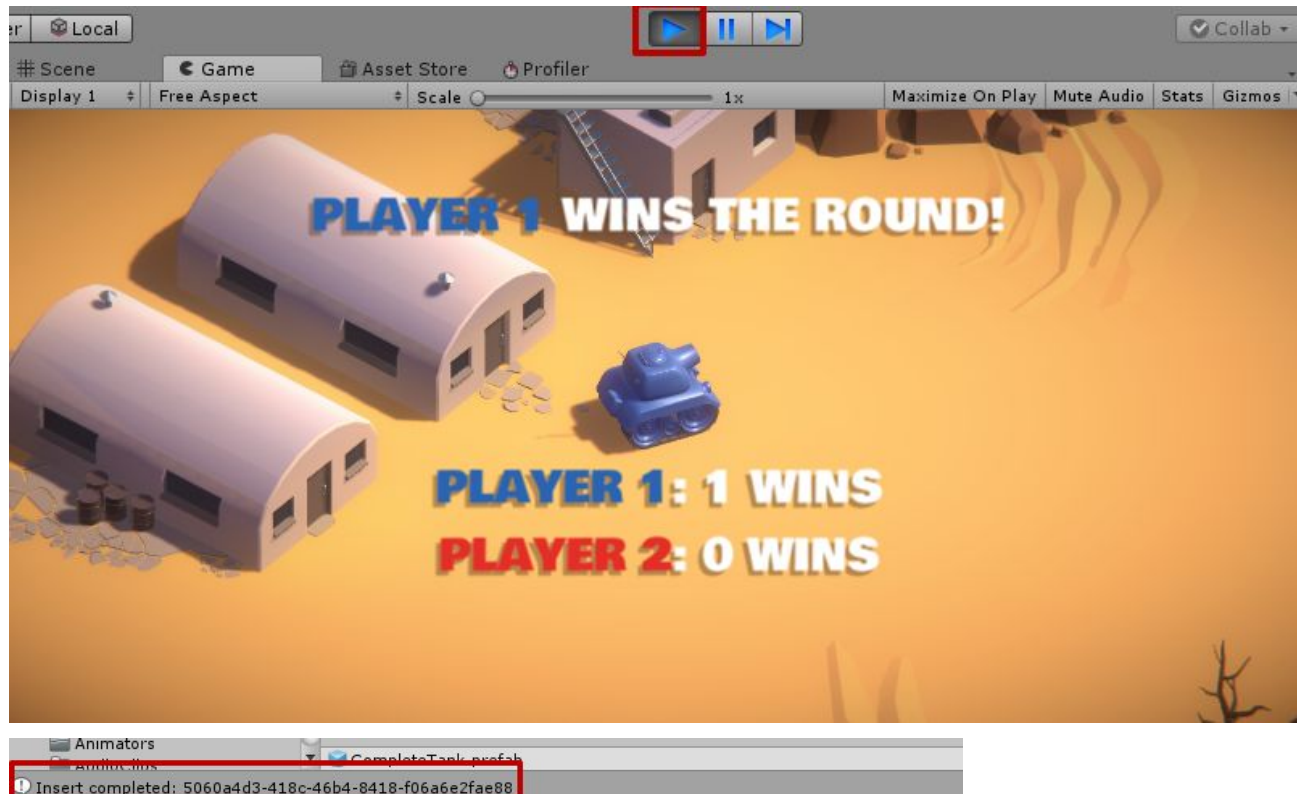
95 public void Send()
96 {
97     Highscore score = new Highscore();
98     int playerNumber = 1;
99     if (GetComponent<TankMovement>().m_PlayerNumber == 1)
100     {
101         playerNumber = 2;
102         score.playername = "Player" + playerNumber;
103         score.score = 1;
104         EasyTablesClient.Instance.Insert(score, insertResponse =>
105         {
106             if (insertResponse.Status == CallBackResult.Success)
107             {
108                 string result = "Insert completed: " + insertResponse.Result.id;
109                 Debug.Log(result);
110             }
111             else
112             {
113                 Debug.Log("Status: " + insertResponse.Status);
114             }
115         });
116         Debug.Log("Sended " + score.playername);
117     }
118 }
```

Двойной клик по названию скрипта **TankHealth** в поле **Script** позволит открыть его в редакторе. В методе **OnDeath ()** добавим вызов метода **Send()**. Этот метод работает по принципу **Insert**, после того как танк умирает он создает в базе данных новую запись, где пишет имя победителя и просто единичку в score.

```
public void Send()
{
    Highscore score = new Highscore();
    int playerNumber = 1;
    if (GetComponent<TankMovement>().m_PlayerNumber == 1)
    {
        playerNumber = 2;
        score.playername = "Player" + playerNumber;
        score.score = 1;
        EasyTablesClient.Instance.Insert(score, insertResponse
=>
    {
        if (insertResponse.Status == CallBackResult.Success)
        {
            string result = "Insert completed: " +
insertResponse.Result.id;
            Debug.Log(result);
        }
        else
            Debug.Log("Status: " + insertResponse.Status);
    });
    Debug.Log("Sended " + score.playername);
}
```

На скриншоте ниже показано как должен выглядеть модифицированный скрипт **TankHealth**.

Сохранение игрового прогресса из Unity в облако.



После всех изменений не забываем сохранить скрипт. Затем идем в **Unity** и запускаем игру.

После того как один из танков победит другой в нашей базе данных в облаке должна будет появиться новая запись с победителем.

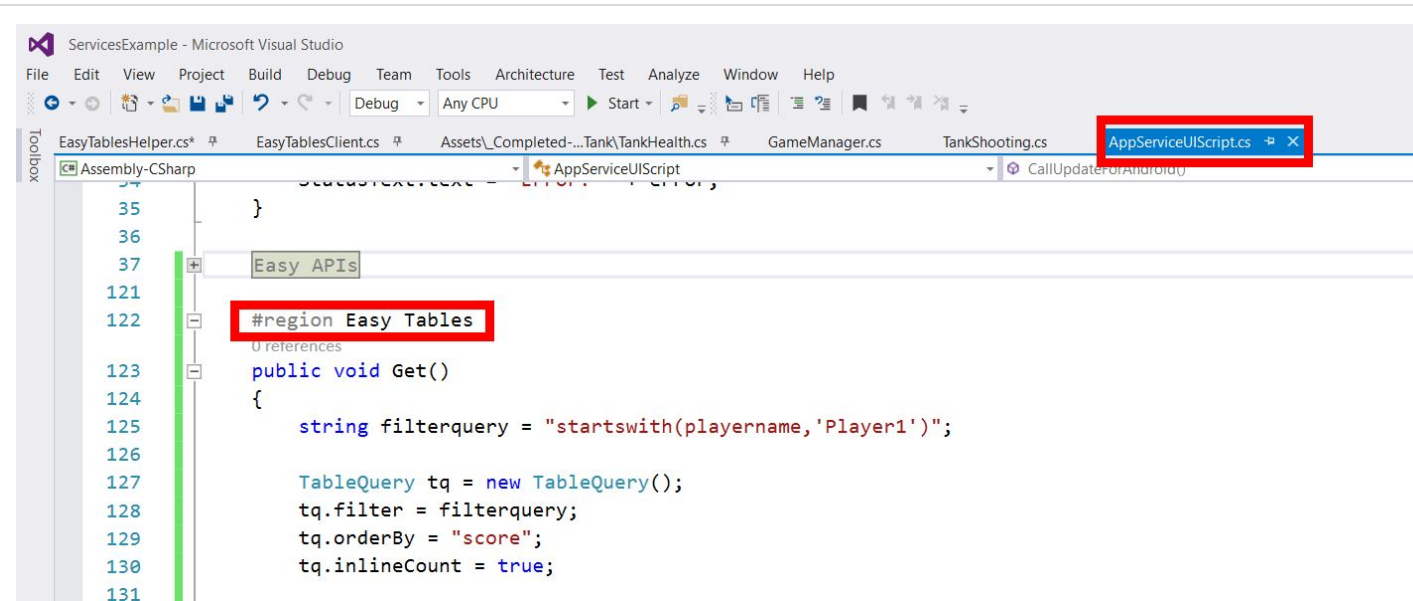
Unity отобразит статус **Insert completed** в нижнем углу консоли.

Highscore (2 records)							
<div>✕ Change permissions Edit script ✕ Manage schema Refresh ... More</div>							
ID	CREAT...	UPDAT...	VERS...	DELETE...	PLAYER...	SCORE	
5b616c0f...	2017-12-...	2017-12-...	AAAAAA...	false	dimitris	53	
<input checked="" type="checkbox"/> 639f984b...	2017-12-...	2017-12-...	AAAAAA...	false	Player1	1	

Теперь зайдите в облако на **EasyTables** в таблице **Highscore** должна появиться запись игрока победителя.

Ниже мы еще рассмотрим пример как можно получить данные о конкретном игроке а точнее как можно легко сделать метод **Select**.

Сохранение игрового прогресса из Unity в облако.



Если открыть скрипт **AppServiceScript** то в нем нужно найти регион **Easy Tables**, это список методов для работы с **Easy Tables Api**.

Insert - метод добавляет в таблицу новую запись

Select (Query) - Метод для получения записей из таблицы с помощью запроса.

SelectByID - метод с помощью которого можно получить запись по ее Id

Update - Можно модифицировать данные в таблице.

DeleteByID - Метод удаления поля в базе данных по его Id

<https://github.com/rio900/unityazureservices>

Ниже написано описание доступных методов работы с **EasyTables**, которые могут пригодиться в вашей игре.

Ниже приведен пример метода для получения всех записей для первого игрока. Значение query можно редактировать получая из базы данных специфические выборки.

```
public void Get()
{
    string filterquery = "startswith(playername,'Player1')";

    TableQuery tq = new TableQuery();
    tq.filter = filterquery;
    tq.orderBy = "score";
    tq.inlineCount = true;

    EasyTablesClient.Instance.SelectFiltered<Highscore>(tq, x
=>
    {
        if (x.Status == CallBackResult.Success)
        {
            foreach (var item in x.Result.results)
            {
                if (Globals.DebugFlag)
                Debug.Log(string.Format("ID is {0},score is {1},name is {2}",
                    item.id, item.score, item.playername));
                StatusText.text = string.Format("Brought {0} rows
out of {1}",
                    x.Result.results.Count(), x.Result.count);
            }
        }
        else
            ShowError(x.Exception.Message);
    });
    StatusText.text = "Loading...";
}
```