

Source: C# Corner ([www.c-sharpcorner.com](http://www.c-sharpcorner.com))

PRINT

---

## Article

---



### KUDU Console In Azure

By [Abdul Rasheed Feroz Khan](#) on **Jul 24 2016**

#### Introduction

This article will help you understand what KUDU console is in Azure and how to work with KUDU.

#### KUDU Console

KUDU Console is a debugging service for Azure platform which allows you to explore your web app and surf the bugs present on it, like deployment logs, memory dump, and uploading files to your web app, and adding JSON endpoints to your web apps, etc.

To access the KUDU console of a Web App, you should be the admin for that particular Web App. Using your Azure login credentials, you can access the KUDU console of your Web App, by entering <https://#####.scm.azureWeb Apps.net>

**Note:** ##### is the name of your Web App.

#### Developer Requirements

1. Windows Azure account ([Click here](#) to get a temporary Azure account for free)
2. Visual Studio 2015 – optional (If you don't have a Web App hosted on your account)
3. Web Browser

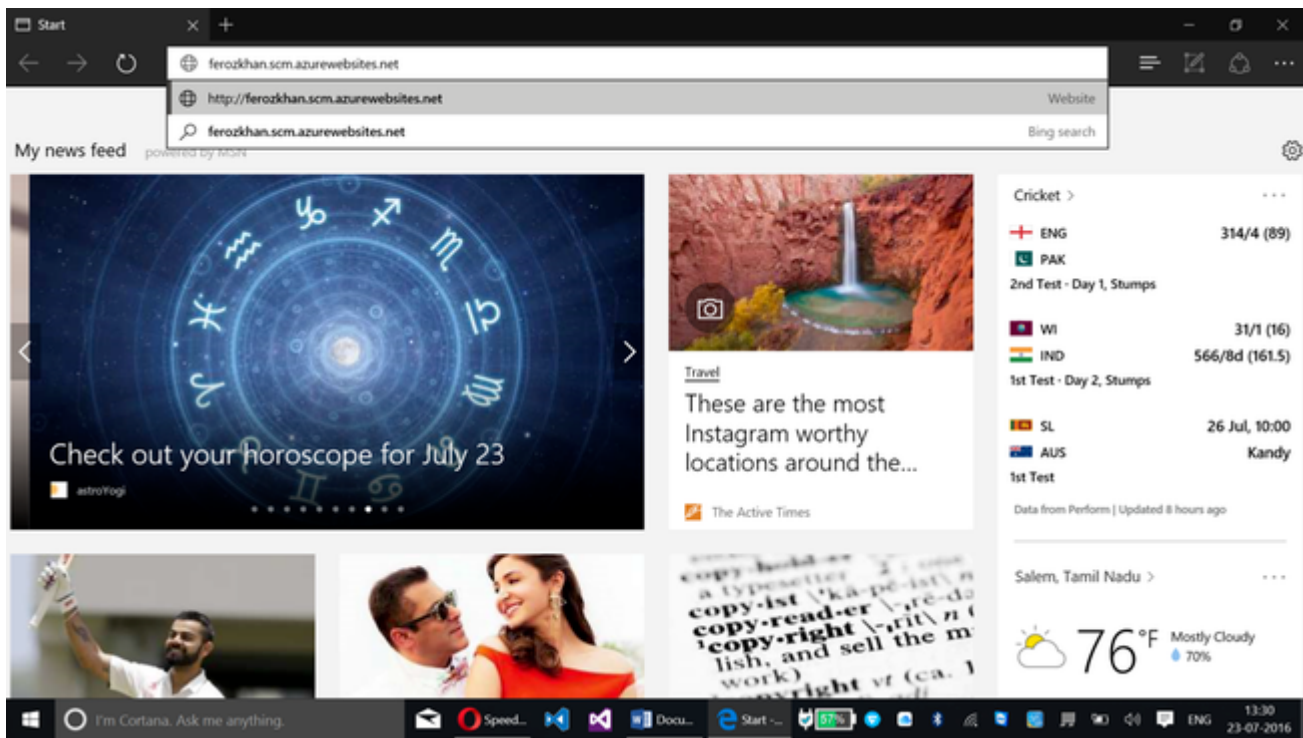
[Click here](#) to learn how to host a Web App using Visual Studio.

Follow the below steps to access the KUDU console of your Web App.

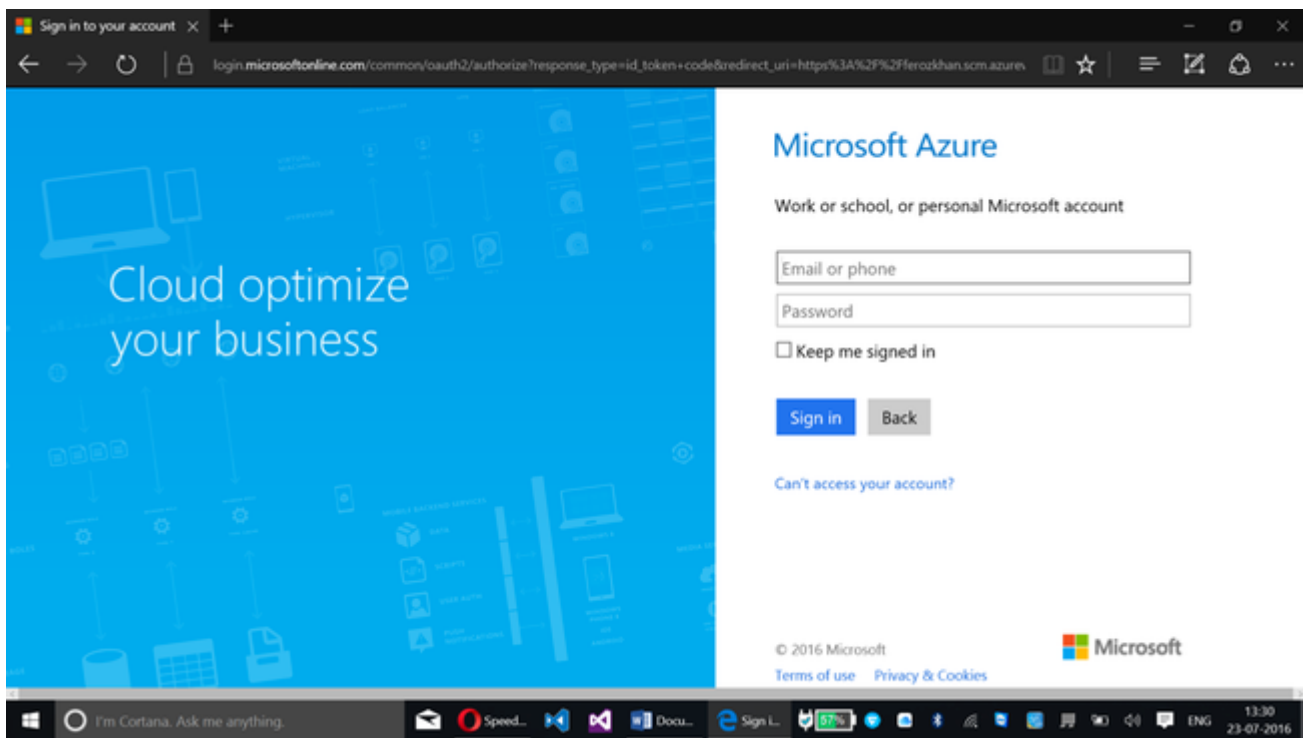
#### **Note:**

- Here, in this demo, I will be working on my own Web App.
- ##### refers to the Web App name.

**Step 1:** Open your web browser and go to <https://www.#####.scm.azurewebsites.net>



**Step 2:** Enter your Azure login credentials over here (Login to the Azure account where your web app has been hosted).



This will take you to the KUDU Troubleshooting Console page of Azure related to your Web App.

The screenshot shows the Kudu Services web interface for the site `ferokhan.scm.azurewebsites.net`. The user is logged in as Ravi Kumar. The **Environment** tab is selected, displaying the following information:

- Build:** 56.50706.2317.0 (1d90b266e8)
- Azure App Service:** 56.0.8598.48 (rd\_websites\_stable.160714-1252)
- Site up time:** 00:00:00:00
- Site folder:** D:\home
- Temp folder:** D:\local\Temp\

Below the environment information, the **REST API** section is visible, with a note: "(works best when using a JSON viewer extension)". It lists several API endpoints:

- App Settings
- Deployments
- Source control info
- Files
- Processes and mini-dumps
- Runtime versions
- Site Extensions: installed | feed
- Web hooks
- WebJobs: all | triggered | continuous
- Functions: list | host.config

### Step 3:

Now go to the Debug Console. Debug Console can be accessed by two ways, either via CMD or via PowerShell. Here, we will be working with PowerShell.

Debug Console --> PowerShell.

The screenshot shows the Kudu Diagnostic Console for the site `ferokhan.scm.azurewebsites.net/DebugConsole/?shell=powershell`. The user is logged in as Ravi Kumar. The **Debug console** tab is selected, displaying a table of recent sessions:

Icon	Path	Timestamp	Status
LogFiles	LogFiles	22-07-2016 21:46:03	
site	site	26-06-2016 12:38:59	

Below the table, there is a section titled **Kudu Remote Execution Console** with instructions:

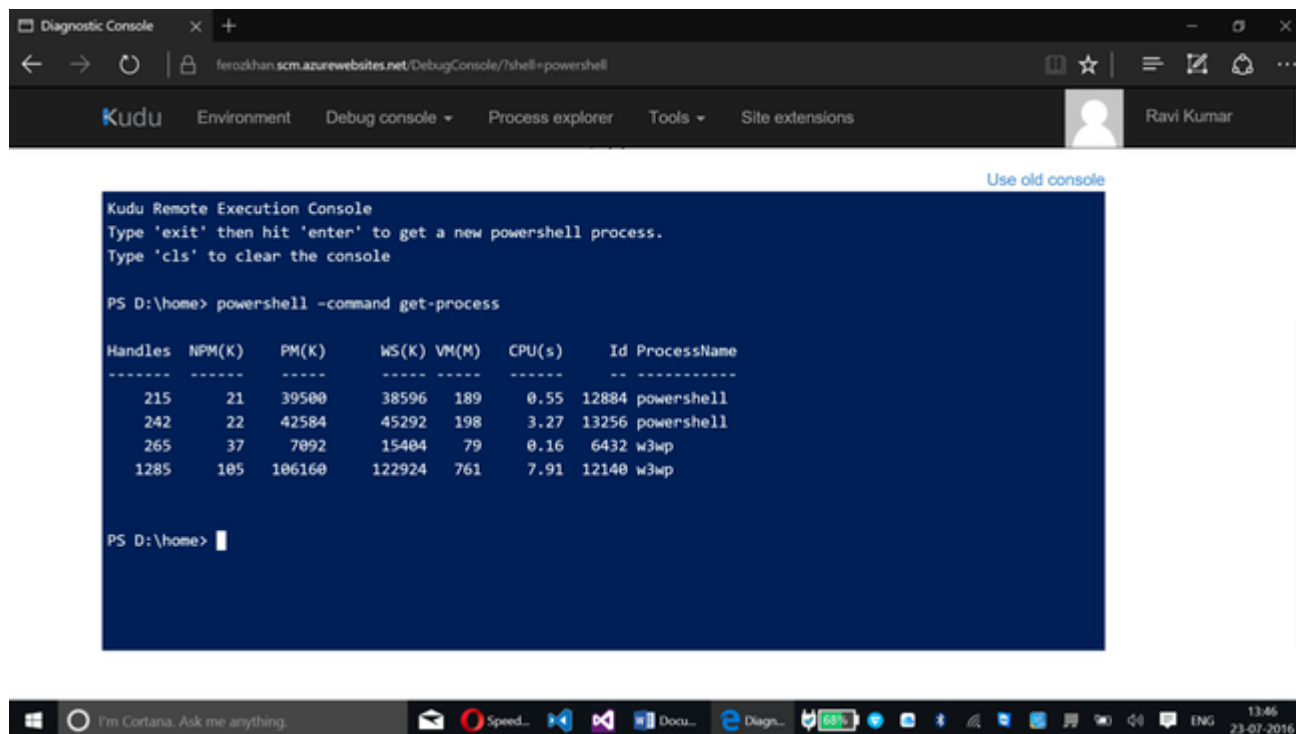
- Type 'exit' then hit 'enter' to get a new powershell process.
- Type 'cls' to clear the console

The console prompt shows `PS D:\home>`. A link "Use old console" is visible in the top right corner of the console area.

This page with PowerShell will appear.

### Step 4:

Enter the command **powershell -command get-process**



The screenshot shows the Kudu Remote Execution Console interface. The browser address bar displays `ferockhan.scm.azurewebsites.net/DebugConsole/?shell=powershell`. The console header includes the Kudu logo, navigation tabs (Environment, Debug console, Process explorer, Tools, Site extensions), and a user profile for Ravi Kumar. The main console area has a dark blue background with white text. It displays instructions: "Type 'exit' then hit 'enter' to get a new powershell process. Type 'cls' to clear the console". Below this, the command `PS D:\home> powershell -command get-process` has been executed. The output is a table of running processes:

Handles	NPM(K)	PM(K)	WS(K)	VM(M)	CPU(s)	Id	ProcessName
215	21	39500	38596	189	0.55	12884	powershell
242	22	42584	45292	198	3.27	13256	powershell
265	37	7892	15404	79	0.16	6432	w3wp
1285	105	106160	122924	761	7.91	12140	w3wp

The console prompt is `PS D:\home>`. At the bottom of the browser window, the Windows taskbar is visible, showing the Start button, Cortana search bar, and several open applications including Speedtest, a music player, and a document editor. The system clock shows 13:46 on 23-07-2016.

Here, you can see the process running on your website. You can find the changes when you give a request at your website, using Id-PID of the W3WP process.

If the changes happens, then the website has a chance of crashing and you need to troubleshoot to find the cause.

---

Thank you for using C# Corner