Content

[Introduction 1](#_Toc365478349)

[Prerequisites 2](#_Toc365478350)

[Website Metrics 2](#_Toc365478351)

[View Metrics 2](#_Toc365478352)

[Configure Metrics to Monitor 3](#_Toc365478353)

[Web Site Logs (CLI) 4](#_Toc365478354)

[Configure Web Site Logs 4](#_Toc365478355)

[Stream Logs 5](#_Toc365478356)

[Download Logs 5](#_Toc365478357)

[Web Site Scaling 5](#_Toc365478358)

[Create a Web Site with Node.JS 7](#_Toc365478359)

[Create Node.JS Site locally 7](#_Toc365478360)

[Create the Web Site with Local Git Deployment 8](#_Toc365478361)

[Set Your Credentials 8](#_Toc365478362)

[Deploy your Node.JS Site Using Git 9](#_Toc365478363)

[Test Your Azure Site 9](#_Toc365478364)

[Create PHP Site locally 10](#_Toc365478365)

[Create the Web Site with Local Git Deployment 10](#_Toc365478366)

[Deploy your PHP Site Using Git 10](#_Toc365478367)

[Test Your Azure Site 11](#_Toc365478368)

[Create a Web Site with Python 11](#_Toc365478369)

[Create Python Site locally 11](#_Toc365478370)

[Create the Web Site with Local Git Deployment 12](#_Toc365478371)

[Deploy your Python Site Using Git 12](#_Toc365478372)

[Test Your Azure Site 12](#_Toc365478373)

[Setup Database Connection (CLI) 13](#_Toc365478374)

[Create Database Server 13](#_Toc365478375)

[Create Database 13](#_Toc365478376)

[Connect Website to Database 14](#_Toc365478377)

# Introduction

The purpose of this tutorial is to show the basics of a Windows Azure Web Site:

* How to collect metrics
* How to scale
* How to setup connections to databases
* How to add handler mappings to run Python, node.js and PHP

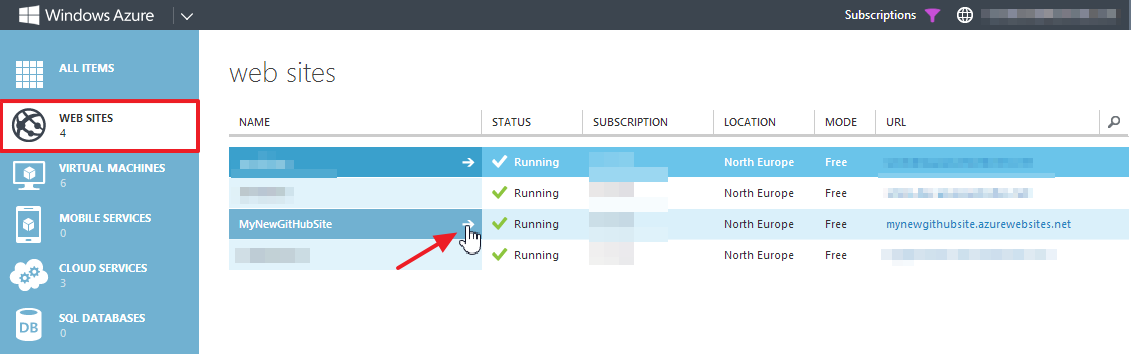
## Prerequisites

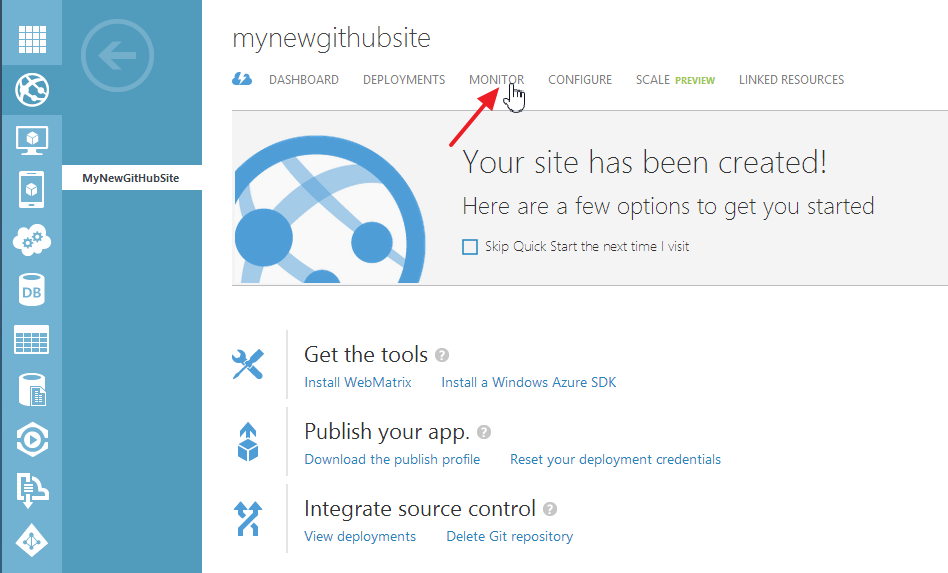
To complete this tutorial, you need an Azure account as well as a configured CLI (see previous tutorial).

# Website Metrics

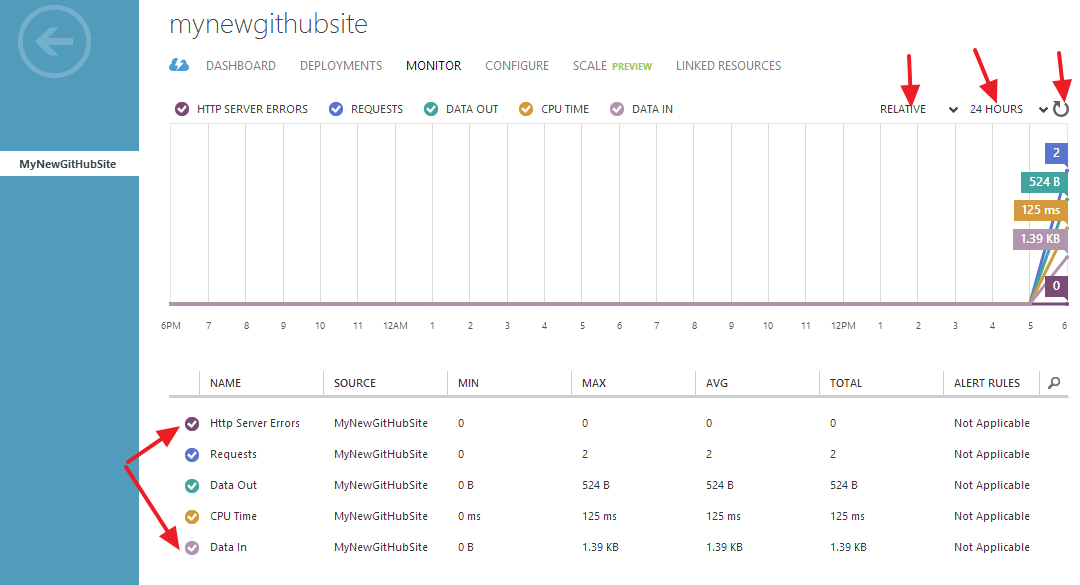
## View Metrics

Login to the Azure dashboard at <https://manage.windowsazure.com>. On the left-hand bar, choose “Web Sites”. Locate the web site you want to collect metrics for and click on the arrow next to its name:



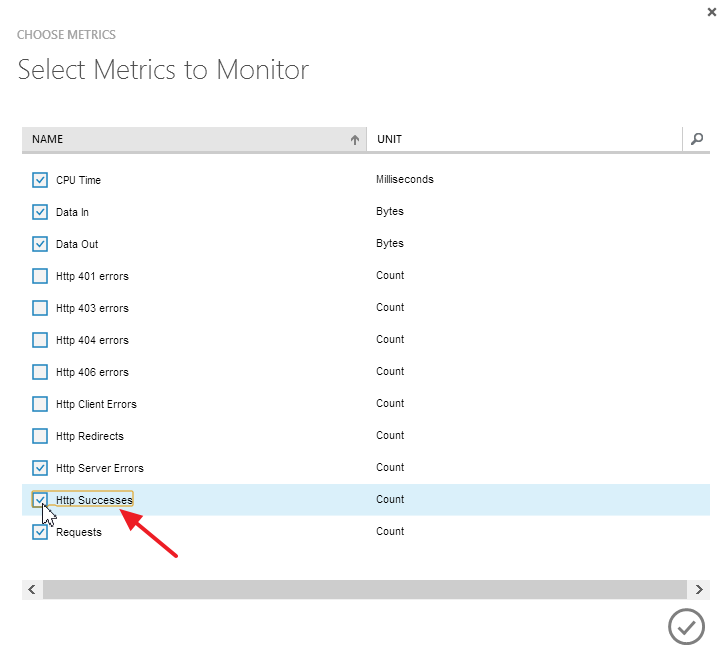
You will see the Quick Start screen or the dashboard. Click on “Monitor” in the menu at the top:

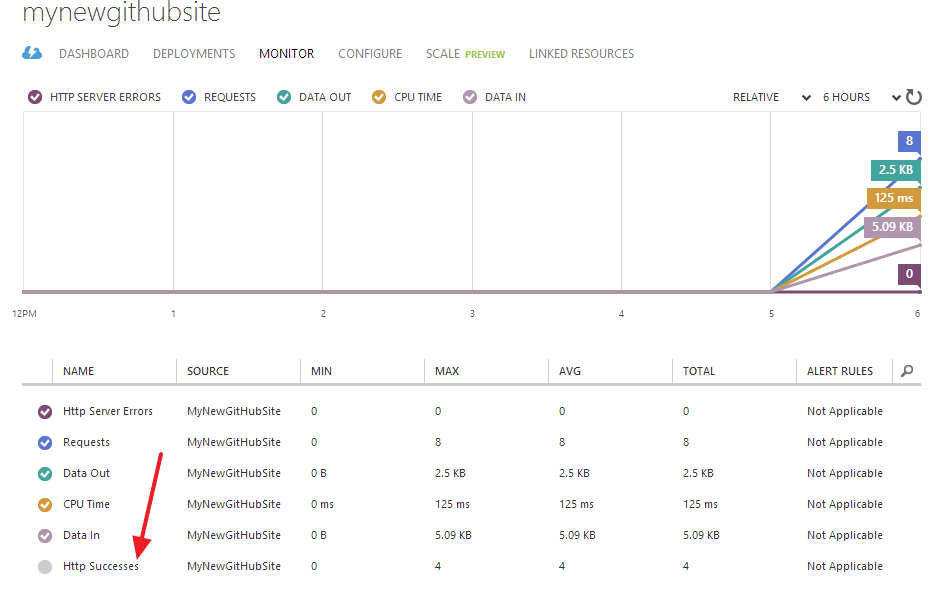
You will see a graph of the various metrics of your website. On this screen, you can:

* Disable and enable the display of various metrics by clicking the checkmark next to the metrics name
* Change the scale from relative to absolute
* Change the time frame displayed
* Refresh the metric

## Configure Metrics to Monitor

You can change the metrics to monitor from this screen as well. In the bar at the bottom, select the “Add Metrics” button:

Select the metrics you want to monitor, or deselect the metrics you want to disable:

After confirming your selection, the metrics displayed are updated according to the new configuration:

# Web Site Logs (CLI)

## Configure Web Site Logs

Assuming the azure CLI is installed and in path, open a console and type the following command:

azure site log set -w -e -f [site]

This will enable web server logging (-w), detailed error messages (-e) and failed requests tracing (-f). If you want to disable any of these options, use the appropriate flag in uppercase (-W, -E, -F).

You can also enable or disable these options with the “Configure” option in the Azure dashboard.

## Stream Logs

With the the following command:

azure site log tail [site]

you can stream new logs onto your local computer. When a new log entry is created (e.g. “not found” 404 error), the log entry will be streamed onto your computer

## Download Logs

Execute the command:

azure site log download [site]

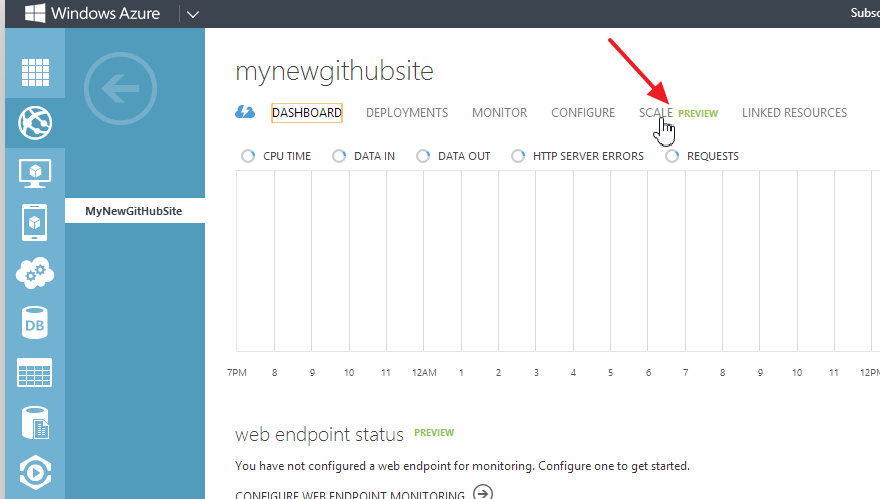
This will download a file called diagnostics.zip that is stored in your current working directory. It contains logs from the webserver as well as other logs as for example Git update logs.

# Web Site Scaling

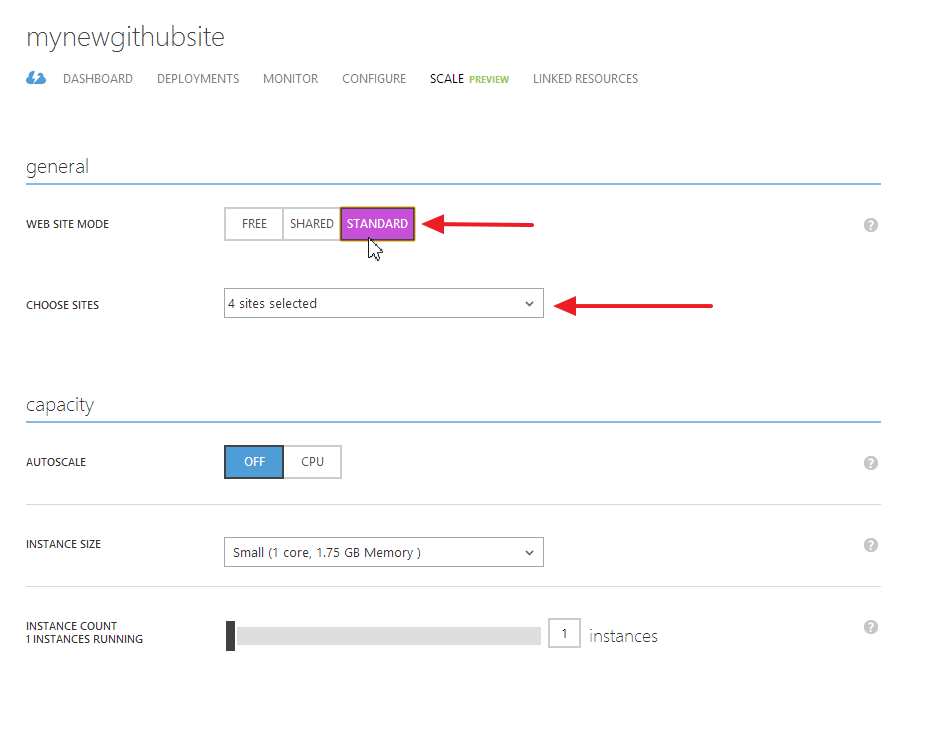
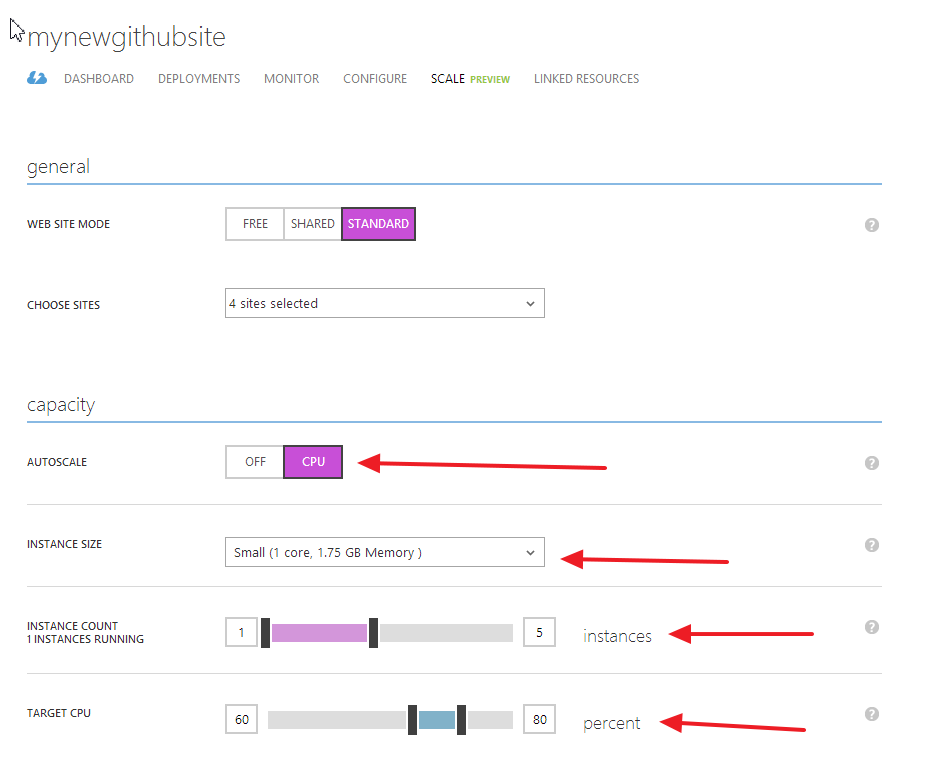
Note: To use this feature, the spending limit for the subscription has to be disabled.

Azure offers three scaling options for a web site to handle high load. “Free” and “Shared” are both hosted on shared instances, but in “Shared” mode, you can define more instances to be used for your website.

“Standard” mode offers the greatest flexibility and runs your website on dedicated virtual machines with the possibility to autoscale.

In the Azure dashboard and with your web site selected, choose “Scale” from the menu bar at the top:

On the next screen, choose “Standard” as your scaling option. Under “Choose Sites”, choose the sites you want to scale:

In the capacity section below, select “CPU” as Autoscale option. This will tell Azure to automatically scale your website based on their CPU usage. Choose the minimum and maximum number of instances you want to be running as well as the target CPU usage:

Azure will automatically add instances if the CPU usage if above the given range and remove instances if the usage is below the given range.

C:\Users\Gernot\Documents\projects\elastacloud\msr-courseware\repo\2. Getting Started with Virtual Machines and Websites\screenshots\c - scaling options\04 save scaling.pngClick on the save button at the bottom to enable scaling with your settings:

# Create a Web Site with Node.JS

In this section, you will create a new Node.JS website. It is assumed that git (Windows: http://msysgit.github.com/) and the Azure CLI is installed and the publish settings file is already imported.

## Create Node.JS Site locally

On your local machine, create a new folder of your choice. This folder will be used to store your website.

Open your favorite text editor and copy the following code that will send the string Hello World to the browser. Save the file as **server.js** in the folder you have previously created:

**var** http **=** require**(**'http'**);**

**var** port **=** process**.**env**.**port **||** 1337**;**

http**.**createServer**(function** **(**req**,** res**)** **{**

res**.**writeHead**(**200**,** **{** 'Content-Type'**:** 'text/plain' **});**

res**.**end**(**'Hello World\n'**);**

**}).**listen**(**port**);**

Create a new text file and paste the following code. Save the file as **package.json** in the same folder. This file will tell the Node.JS package manager (npm) about your application structure and could also contain dependencies if required:

**{**

"name"**:** "HelloWorld"**,**

"version"**:** "0.0.1"**,**

"description"**:** ""**,**

"main"**:** "./server.js"**,**

"engines"**:** **{** "node"**:** ">= 0.6.0" **}**

**}**

In the text editor create a new file and paste the following code. Save it as **web.config** in your web site folder.

This configuration indicates that the **server.js** file is a Node.js application that should be handled by the **iisnode** module. iisnode is a native IIS 7.x module that allows hosting of node.js applications in IIS 7.x.

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<configuration>

<system.webServer>

<modules runAllManagedModulesForAllRequests=**"false"** />

<!-- indicates that the server.js file is a node.js application

to be handled by the iisnode module -->

<handlers>

<add name=**"iisnode"** path=**"server.js"** verb=**"\*"** modules=**"iisnode"** />

</handlers>

<rewrite>

<rules>

<clear />

<rule name=**"app"** enabled=**"true"** patternSyntax=**"ECMAScript"** stopProcessing=**"true"**>

<match url=**"server\.js.+"** negate=**"true"** />

<conditions logicalGrouping=**"MatchAll"** trackAllCaptures=**"false"** />

<action type=**"Rewrite"** url=**"server.js"** />

</rule>

</rules>

</rewrite>

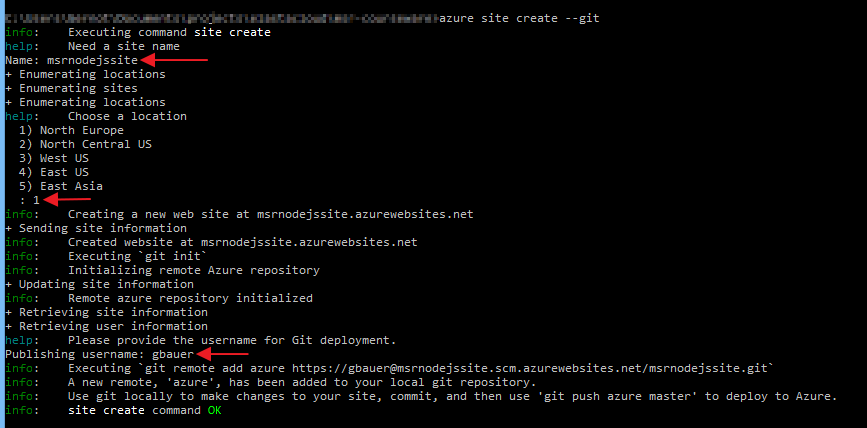
</system.webServer>

</configuration>

## Create the Web Site with Local Git Deployment

Open up a console/shell. Change to your local website directory and create a new site with the following command:

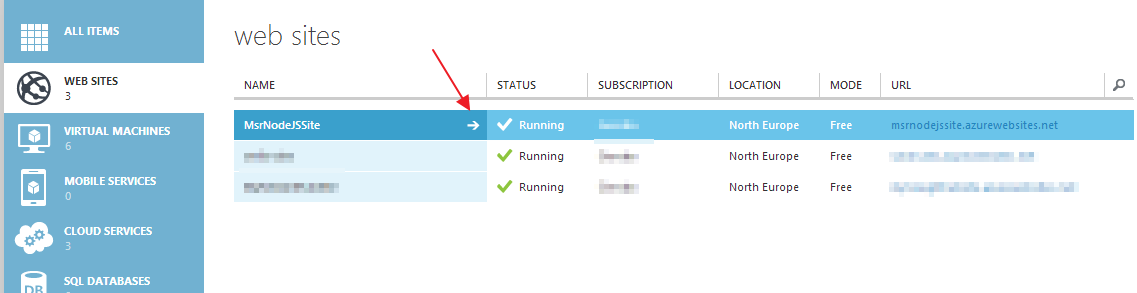
azure site create --git

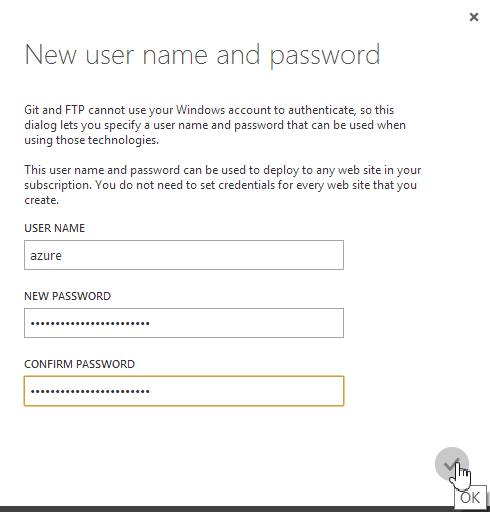
Enter the necessary information when prompted. The output should look like this:

If you already setup deployment credentials for any of your site, use the user name of these credentials. Otherwise, you can set up new credentials in the next step

## Set Your Credentials

Go to <https://manage.windowsazure.com> and under “Web Sites”, select the site you created and click on the arrow:



Select the “Deployments” tab and click on the “Reset your Deployment Credentials” link at the bottom:

Save your credentials and continue.

## Deploy your Node.JS Site Using Git

In your shell, execute the following command:

git add .

git commit -m "initial commit"

git push azure master

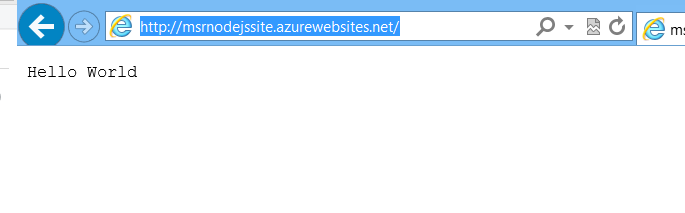
This will commit your files to your local directory and push it to Azure.

## Test Your Azure Site

After pushing your local content to your web site, you should test your web site. The following command will open and browser pointing to your new site:

git site browse

You should see a Hello World message:

Create a Web Site with PHP

In this section, you will create a new PHP website. It is assumed that git (Windows: http://msysgit.github.com/) and the Azure CLI is installed and the publish settings file is already imported.

## Create PHP Site locally

On your local machine, create a new folder of your choice. This folder will be used to store your website.

Open your favorite text editor and copy the following code. Save the file as **index.php** in the folder you have previously created:

<!DOCTYPE html>

<html>

<head>

<title>**Windows Azure Web Sites - Hello World sample!**</title>

</head>

<body>

<h1>**Windows Azure Web Sites - Hello World sample!**</h1>

<?php

// Show all information, defaults to INFO\_ALL

phpinfo();

?>

</body>

</html>

## Create the Web Site with Local Git Deployment

Open up a console/shell. Change to your local website directory and create a new site with the following command:

azure site create --git

Enter the necessary information when prompted.

## Deploy your PHP Site Using Git

In your shell, execute the following command:

git add .

git commit -m "initial commit"

git push azure master

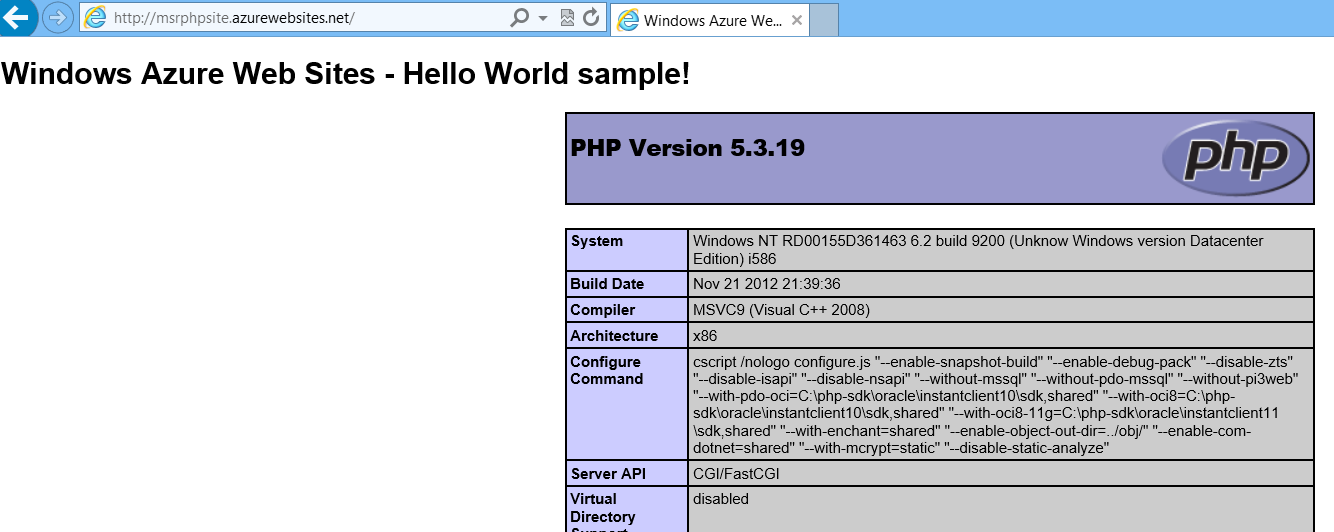
This will commit your files to your local directory and push it to Azure.

## Test Your Azure Site

After pushing your local content to your web site, you should test your web site. The following command will open and browser pointing to your new site:

git site browse

You should see the PHP info page:



# Create a Web Site with Python

In this section, you will create a new Python website. It is assumed that git (Windows: http://msysgit.github.com/) and the Azure CLI is installed and the publish settings file is already imported.

## Create Python Site locally

On your local machine, create a new folder of your choice. Create a subdirectory named “ConfigurePython”.

Open your favorite text editor and copy the following code. Save the file as **ConfigurePython.py** in the ConfigurePython subdirectory you have previously created:

**def** application**(**environ**,** start\_response**):**

status **=** '200 OK'

response\_headers **=** **[(**'Content-type'**,** 'text/plain'**)]**

start\_response**(**status**,** response\_headers**)**

**yield** 'Hello from Windows Azure Websites\n'

*application* is a Python callable, which will serve as the entry point called by a WSGI-compliant server. This callable object accepts 2 positional arguments:

* *environ*: a dictionary with various environment variables
* *start\_response*: a callable provided by the web server for transfer of HTTP status and response header

This handler will return the plain text “Hello from Windows Azure Websites” for every request made to it.

To set the correct handler mapping, create a file named web.config in the web root directory and paste the following content:

<configuration>

<appSettings>

<add key=**"pythonpath"** value=**"D:\home\site\wwwroot\ConfigurePython"** />

<add key=**"WSGI\_HANDLER"** value=**"ConfigurePython.application"** />

</appSettings>

<system.webServer>

<handlers>

<add name=**"PythonHandler"**

verb=**"\*"** path=**"handler.fcgi"**

modules=**"FastCgiModule"**

scriptProcessor=**"D:\Python27\Python.exe|D:\Python27\Scripts\wfastcgi.py"**

resourceType=**"Either"** />

</handlers>

<rewrite>

<rules>

<rule name=**"Configure Python"** stopProcessing=**"true"**>

<match url=**"(.\*)"** ignoreCase=**"false"** />

<conditions>

<add input=**"{REQUEST\_FILENAME}"** matchType=**"IsFile"** negate=**"true"** />

</conditions>

<action type=**"Rewrite"** url=**"handler.fcgi/{R:1}"** appendQueryString=**"true"** />

</rule>

</rules>

</rewrite>

</system.webServer>

</configuration>

In addition, create a file called “handler.fcgi” in the base directory as well. Leave this file empty.

This configuration:

* Has python scripts in the path “ConfigurePython”
* Rewrites all urls to this path. In case you want to have resources (e.g. CSS and images) not running through Python, you can change the rewrite rules to incorporate these options.

## Create the Web Site with Local Git Deployment

Open up a console/shell. Change to your local website directory and create a new site with the following command:

azure site create --git

Enter the necessary information when prompted.

## Deploy your Python Site Using Git

In your shell, execute the following command:

git add .

git commit -m "initial commit"

git push azure master

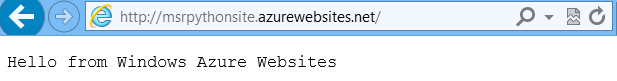
This will commit your files to your local directory and push it to Azure.

## Test Your Azure Site

After pushing your local content to your web site, you should test your web site. The following command will open and browser pointing to your new site:

git site browse

You should see the output of the Python script above:



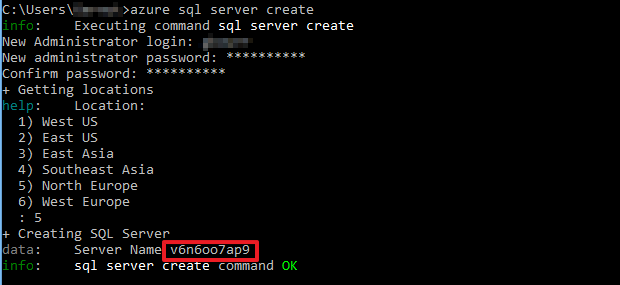
# Setup Database Connection (CLI)

## Create Database Server

Open a command prompt and enter the following command:

azure sql server create

This will create an Azure SQL Server where you can create databases. Set the login name, password and region when prompted. The region should be near your website. The output looks like this:



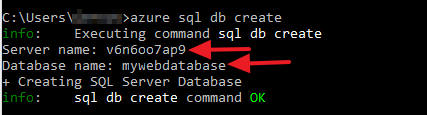
Remember the server name as you will need it in the next step.

## Create Database

The next step is to create a database. You can do this with the command

azure sql db create

You will be prompted for the SQL server as well as the database name to use. Use the server name from the previous step as server name. Use descriptive name for the database name:



## Connect Website to Database

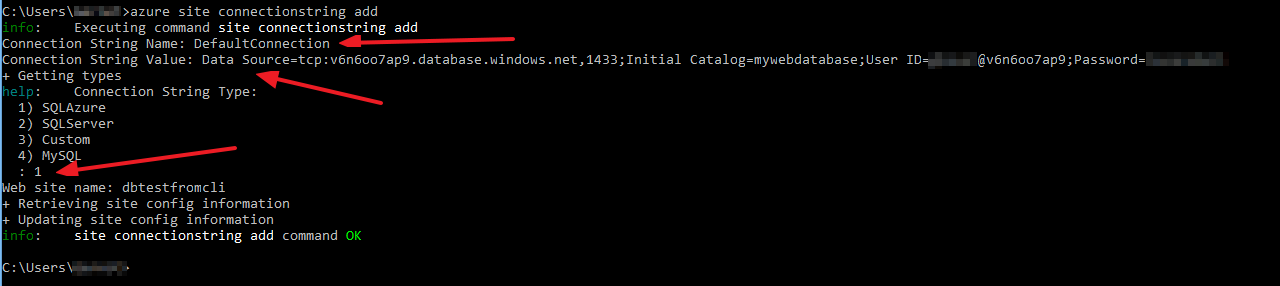
In the final step, you have to set the connection string for the database connection in your webapp. The connection string is of the form:

Data Source=tcp:[db\_server].database.windows.net,1433;Initial Catalog=[db\_name];User ID=[admin\_login]@[db\_server];Password=[password]

Replace the placeholders with your configuration. In the command prompt, enter:

azure site connectionstring add

This will prompt you for an (arbitrary) connection string name, the connection string itself and the type of the database connection (SQLAzure):

Now the connection string for the site is configured.

For .NET sites, these connection strings will be injected into your .NET configuration connectionStrings settings at runtime, overriding existing entries where the key equals the linked database name. For PHP and Node sites these settings will be available as environment variables at runtime, prefixed with the connection type. The environment variable prefixes are as follows:

* SQL Server: SQLCONNSTR\_
* MySQL: MYSQLCONNSTR\_
* SQL Database: SQLAZURECONNSTR\_
* Custom: CUSTOMCONNSTR\_

If you named your connection string “DefaultConnection”, it can be accessed through the environment variable “SQLAZURECONNSTR\_DefaultConnection”.