Azure 101

Practice Development Unit

Technical Enablement

Student Guide

May 2017

© 2017 Microsoft Corporation. All rights reserved. This document is confidential and proprietary to Microsoft. Internal use only. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

Some examples are for illustration only and are fictitious. No real association is intended or inferred.

## Contents

[Introduction 1](#_Toc481680052)

[Pre-requisites 2](#_Toc481680053)

[Accessing and customising the portal 3](#_Toc481680054)

[Documentation 6](#_Toc481680055)

# Introduction

The Azure 101 session is intended as an introductory training for technical learners who have had little to no experience with Azure public cloud.

The aim is to familiarise the student with some of the most commonly used IaaS and PaaS services in Azure, and the portal and CLI interfaces available to drive them.

In terms of orientation, the trainer may use slides to help give an overview of some of the other services available within Azure, and the training will also make the student aware of some of the documentation and training resources available to them as they continue to explore.

## Content

The below sections are PowerPoint content unless specified otherwise

1. Azure Intro
   1. Cloud drivers and key Azure principles
   2. Scale and compliancy
   3. Rate of change, services and open source
2. Infrastructure Services
   1. Compute options
   2. Networking fundamentals
   3. **LAB: Portal familiarisation and customisation, resource groups, vNets and subnets, documentation resources**
   4. Storage and RBAC principles
   5. **LAB: Windows and Linux VMs, customising NSGs, defining Availability Sets**
3. Lunch
4. Application & Platform Services
   1. Responsibilities: Traditional v IaaS v PaaS v SaaS
   2. Web Apps, Mobile Apps, etc
   3. Functions and Logic Apps
5. DevOps & Automation
   1. CI/CD
   2. Toolchains
   3. Monitoring and Insight
6. **LAB: Deploying to Web Apps from a local Docker repository**
7. Data and Analytics
   1. Hadoop, HDinsight and Data Lake
   2. SQL DB, Data Warehouse and Managed Instance
   3. MySQL, PostgreSQL and CosmosDB
   4. Cortana Intelligence Suite, Cognitive Services, Machine Learning and IoT Suite
8. **LAB: Using Logic Apps with the Twitter API**
9. Self-training and options for future enablement sessions

# Pre-requisites

## Overview

All students must complete the required pre-requisites for the training prior to attendance, to maximise the time and value of the session itself.

There is a companion document, *Azure Student Prereqs Guide*, that provides additional guidance to meet the pre-requisites.

## Requirements

* An active Microsoft Azure subscription

## Recommended

* A Twitter account

## Optional

CLI 2.0 and Git are only required for users who wish to use those tools for the Web Apps lab. The student guide also includes instructions for using the Azure portal to achieve the same end result.

* Standard users
  + Cloud Shell within the Azure Portal includes both CLI 2.0 and Git commands
* Power users
  + Windows 10 Linux Subsystem (lxss)
  + Azure CLI 2.0 installed
  + Add Git package to lxss

# Accessing and customising the portal

## Azure Resource Manager Portal

The main Azure portal is <http://portal.azure.com>.

Login using the account for your Azure subscription. Your account information is at the top right, including password change, and viewing permissions and your bill.

Next to that is the Help + Support, for accessing the help or opening up a support ticket. Click on the Help + Support icon and then

* launch the guided tour
* see what’s new
* check the keyboard shortcuts

The cog icon is the Portal Settings. You can filter when you have multiple subscriptions, change the language and certain portal characteristics.

* Change the theme

Next to the Azure Cloud Shell is the Notifications section, for status updates, billing updates and to show deployment activity.

## Dashboard Customisation

The Azure portal enables you to have multiple dashboards and to customise those dashboards. You can also share dashboards with other AAD users or groups within the subscription, leveraging the role based access control (RBAC) to control who has access.

* Create a new dashboard, and name it Azure 101
* From the Tile Gallery’s General area, pull across
  + All Resources Reconfigure (using the ellipsis (…)) to change to 4x4 tiles
  + Clock Reconfigure to 2x1, 24 hour, and London time
  + Quickstart Tutorials
  + Markdown Change title to Azure 101, subtitle to Useful Links, and replace the Content with the markdown below and then resize to fit

### Documentation

\* [Products](https://azure.microsoft.com/en-us/services) and [Pricing]( https://azure.microsoft.com/en-us/pricing/)

\* [Azure Docs](<https://docs.microsoft.com/en-us/azure/>) and [Subscription Limits](<https://docs.microsoft.com/en-us/azure/azure-subscription-service-limits>)

\* [Architecture](https://docs.microsoft.com/en-us/azure/index#pivot=architecture)

\* [Learning Paths](https://azure.microsoft.com/en-us/documentation/learning-paths/)

\* Interactive [Azure Services](http://azureplatform.azurewebsites.net/en-us/) Overview

\* Azure [Resource Explorer](<https://resources.azure.com/>) and [Storage Explorer](<http://storageexplorer.com/>)

\* Azure [Price Calculator](<https://azure.microsoft.com/en-us/pricing/calculator/>) and [TCO Calculator](https://www.tco.microsoft.com/)

### Partner Portals

\* [MPN Portal](http://partner.microsoft.com/)

\* [Cloud Solution Partner (CSP)](https://partner.microsoft.com/en-us/cloud-solution-provider)

\* [Azure Account Portal](http://account.windowsazure.com/Subscriptions)

\* [Partner Support](https://partner.microsoft.com/en-GB/support), including [Advanced Support for Partners (ASP)](https://partner.microsoft.com/en-US/Support/advanced-cloud-support?advancedcloudsupport) and [Visual Studio Subscriptions](https://support.microsoft.com/kb/4013871/?tpqid=800-000036)

\* [Cloud Practice Playbooks](https://partner.microsoft.com/en-US/campaigns/cloud-practice-playbooks) and [Partner Concierge](https://aka.ms/ukconcierge) for marketing

### Training Resources

\* [OpenEdx MOOC Courses](https://openedx.microsoft.com/)

\* [Channel9](https://channel9.msdn.com/)

\* [Microsoft Virtual Academy](https://mva.microsoft.com/product-training/microsoft-azure)

\* [Partner University](https://partner.microsoft.com/en-gb/training), including [Partner Upskill](https://aka.ms/mpnukupskill) and [Cloud and Proud](https://www.microsoft.com/uk/partner/cloudandproud/)

\* Prep guides for [70-533](https://mva.microsoft.com/en-US/training-courses/certification-exam-overview-70-533-implementing-microsoft-azure-infrastructure-solutions-17405) and [70-534](https://mva.microsoft.com/en-us/training-courses/certification-exam-overview-70-534-architecting-microsoft-azure-solutions-17406)

### Keeping Up To Date

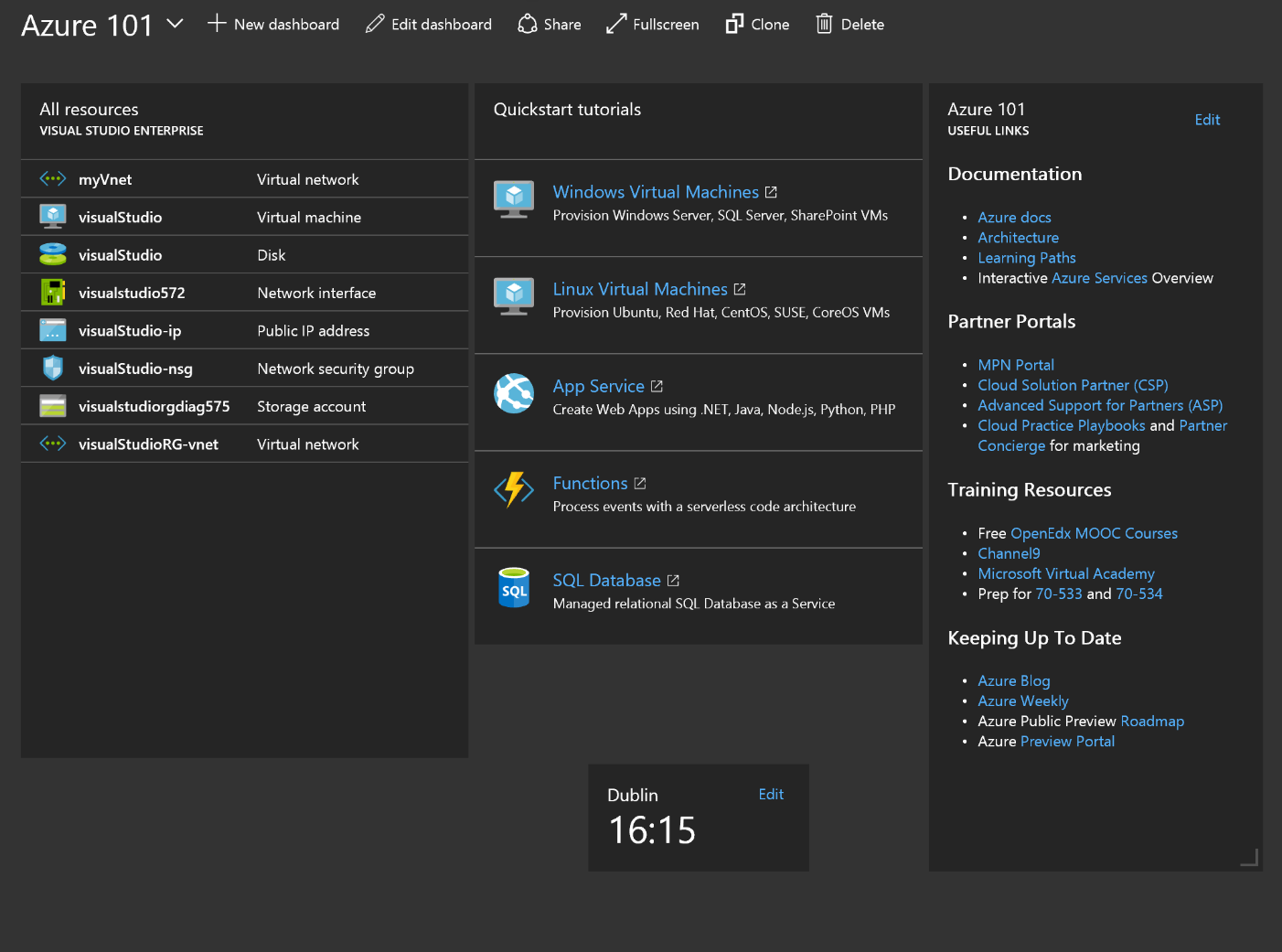
\* [Azure Blog](https://azure.microsoft.com/en-gb/blog/)

\* [Azure Weekly](https://buildazure.com/)

\* [Roadmap](https://azure.microsoft.com/en-gb/roadmap/) and [Updates]( https://azure.microsoft.com/en-gb/updates/)

\* Azure [Preview Portal](http://preview.portal.azure.com)

Once complete, your dashboard should look something like this:



UPDATE THIS PICTURE!

# Documentation

Let’s click through some of the documentation links in the Markdown box:

* Interactive [Azure Services](http://azureplatform.azurewebsites.net/en-us/) Overview
* [Products](https://azure.microsoft.com/en-us/services) and [Pricing](https://azure.microsoft.com/en-us/pricing)
* [Azure Docs](https://docs.microsoft.com/en-us/azure)
* [Architecture](https://docs.microsoft.com/en-us/azure/index#pivot=architecture)
* [Learning Paths](https://azure.microsoft.com/en-us/documentation/learning-paths)

# Create a resource group called Azure101IaaS

* Open the Azure [portal](http://portal.azure.com)
* Either
  + Click on the + New icon (or G+N), search for “Resource Group” and click on it
  + Click on Resource Groups in your favourites and click on Add
  + Click on the More Services > icon, Resource Groups in the General section and then on Add
* Create with
  + Resource Group Name Azure101IaaS
  + Resource Group Location West Europe
* Note deployment notification area
* Refresh the resource groups and click on the Azure101IaaS resource group

# Create a vNet with two subnets

* Add a Virtual Network
  + Click on the +
  + Search on “Virtual Network”
  + Select, then Create
* Name: azure101vNet
* Address space: 10.4.0.0/16
* Subnet name: webSubnet
* Subnet address range: 10.4.1.0/24

Once created, click into it and add the second subnet:

* Select subnets on the blade
* Add dbSubnet (10.4.2.0/24)

Search on information in the portal and on the Azure docs area for:

* Network Security Groups (NSGs)
* GatewaySubnet
* ExpressRoute and Site-to-Site (S2S) VPN Gateways
* Network Virtual Appliances
* User Defined Routes (UDRs) in Route Tables
* vNet Peering
* Region to region S2S VPNs

# Windows virtual machine

Go to the Azure Docs area, and click on Windows Virtual Machines in the Deploy Infrastructure section of the Get Started tab. Choose a five minute quickstart, either portal, PowerShell or CLI.

## Create the VM

* Add a Windows Server 2016 Datacenter server
* Name: myWindowsVM
* VM disk type: SSD
* Resource Group: Existing Azure101IaaS resource group
* Location: West Europe
* Size: DS1\_v2
* Use Managed Disks: Yes

Understand

* Azure Hybrid Use Benefit
* Benefits of Managed Disks vs manual Storage Accounts

## HTTP Web Server

* Add IIS via a terminal services session
* Open port 80/tcp inbound on the NSG
* Test access via a browser
* Stop the VM to deallocate resource

Understand

* Additional VM tutorials available - disks, DSC, images, scale sets, load balancing, backup, monitoring, policies, etc.
* Deallocation vs shutdown in the Remote Desktop console

# Linux virtual machine

## Create your SSH keypair

Run the following commands in your Bash shell:

cd ~

umask 033

ssh-keygen -t rsa -b 2048 -C "richard.cheney@microsoft.com"

ls -Al .ssh

cat .ssh/id\_rsa.pub

The -C is a comment. It is good etiquette to add in your email address so that any key owners are easily identified. You will be prompted for a filename. You can specify an alternative name, or hit enter to use the defaults (recommended). For the passphrase, it is best practice to provide a passphrase with a few memorable but disassociated words. However, for the sake of this exercise you may simply press enter for an empty passphrase, and then you won’t be challenged for it later when using ssh to connect toi the Linux VM.

Copy the contents of the id\_rsa.pub file to the clipboard as you will use this later.

## Create a VM template

Do the following from the portal

* Add an Ubuntu Server 16.04 LTS server
* Name: myLinuxVM
* VM disk type: HDD
* SSH Public Key: Copy and pasted from cat ~/.ssh/id\_rsa.pub
* Resource Group: Existing Azure101IaaS resource group
* Location: West Europe
* Size: D1\_v2
* Subnet: dbSubnet
* High Availability Set: Create myLinuxAS

Do not click on Create straight away. Let’s explore templates a little.

* Click instead on Download template and parameters. Browse the template JSON, the parameters JSON, and the scripts to deploy the template from PowerShell and CLI 2.0.
* Click on Create to submit the job.
* Browse to Services tab on Azure Docs, and then find the Azure Resource Manager tile in the Monitoring + Management area. Discover the [export template](https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-manager-export-template) functionality, as well as the JSON [template structure](https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-authoring-templates).
* Click on Add within the Azure101IaaS Resource Group, and fine Template Deployment.
* Search in the Load a GitHub quickstart template, type “load” and select one of the community templates and accompanying parameters file.
* Search the Azure Quickstart Templates GitHub repo to find the same template.

Use the Bash shell to open a secure shell session

* SSH to the VM’s public IP address: ssh <adminuser>@<public IP address>

# WebApps

## Overview

The PaaS workshop will touch on both Web Apps and Logic Apps. The diagram below gives an overview of what we are doing with a set of html files hosted on GitHub and pushing those into a Web App in Azure.



1. Use git clone to download the sample static HTML site from GitHub to your laptop
2. Login to Azure and create a deployment user
3. Create the Azure101PaaS resource group
4. Create the free tier app service plan
5. Create a web app within the app service plan
6. Configure a Git access point for the web app
7. Define the access point as an upstream git repo called “azure”, and use git push to add the html files

## Brief Instructions

This is a short piece of guidance intended only for users who are comfortable with html, bash and git concepts. If that is not you then please quickly move to the following section, Detailed Instructions!

Follow the Quickstart Tutorial for App Services. The pane on the portal dashboard will take you to the section in Azure Docs. The five minute static HTML 5 tutorial is on the right.

Please make the following changes:

* Call the Resource Group Azure101PaaS
* Do not use the suggested git repo, instead use this one:

git clone https://github.com/richeney/azure101-webapp-html

* You may need to run git init within the azure101-webapp-html directory if you receive errors about the repository
* Do not delete the resource group at the end!

If you hit issues then check the longer set of instructions below first, and if you still have problems then call over the instructor.

## Detailed Instructions

## Select your console

Users that have Windows 10 should have installed the Linux subsystem, and then installed both CLI 2.0 and Git into that subsystem. should open a Command Prompt and then type bash. Type in az login and follow the instructions. This will be your console.

Everyone else may use the Cloud Shell (**>\_**)shown at the top of the Azure Portal. This will create some local storage the first time you use it. There is no need to login to Azure as it is done automatically for you. (Note that we will not be using the permanent storage folder clouddrive as it does not support the file permissions required by git commands, so the clone of the Github repo will be transient.) This will be your console.

### Clone the html files from GitHub

* Type (or copy and paste) the following commands into the console, changing the italicised value for appName to include your name or alias. Note that the appName value will need to be globally unique as it will form the web URL.

git clone <https://github.com/richeney/azure101-webapp-html>

cd azure101-webapp-html

git init

ls -Al

pwd

git config --global credential.helper cache

rg=Azure101PaaS

user=azure101deploy

pwd=azure101p455w0rd

appName=azure101*YourNameHere*

* The first commands copy the html files locally, change to that directory, initialises it for git, and then lists the files. The pwd command prints the working directory so that you know where they are.
* We are then setting git to cache our credentials after the first successful connection to a remote, and also defining some variables to use in the commands in the following sections.
* Once the files have downloaded then double click the index.html file in File Explorer to view the website locally. You should see a couple of pieces of static images and text on the left, and a Twitter timeline on the right. If you are in the Cloud Shell then cat index.html will display the raw html.

### Login in to Azure via CLI 2.0 and create the deployment user

* Click on the App Services in the Quickstart Tutorials

az webapp deployment user set --user-name $user --password $pwd

* You will be prompted to make you username or password more unique if you have chosen one that is too common, although the error is a basic 400 decline HTTP error
* The deployment user is the equivalent of a service account so that Git can authenticate to Azure and deploy to it

### Create the resource group

Within CLI 2.0 create the resource group

az group create --name $rg --location westeurope

### Create the App Service plan

The App Service plans provide the dedicated resource for your apps, and multiple apps can use them. The plans define the region, available instance sizes, scale count and SKU level, i.e. free, shared, basic, standard, premium.

* Create an App Service plan called quickStartPlan on the Free SKU

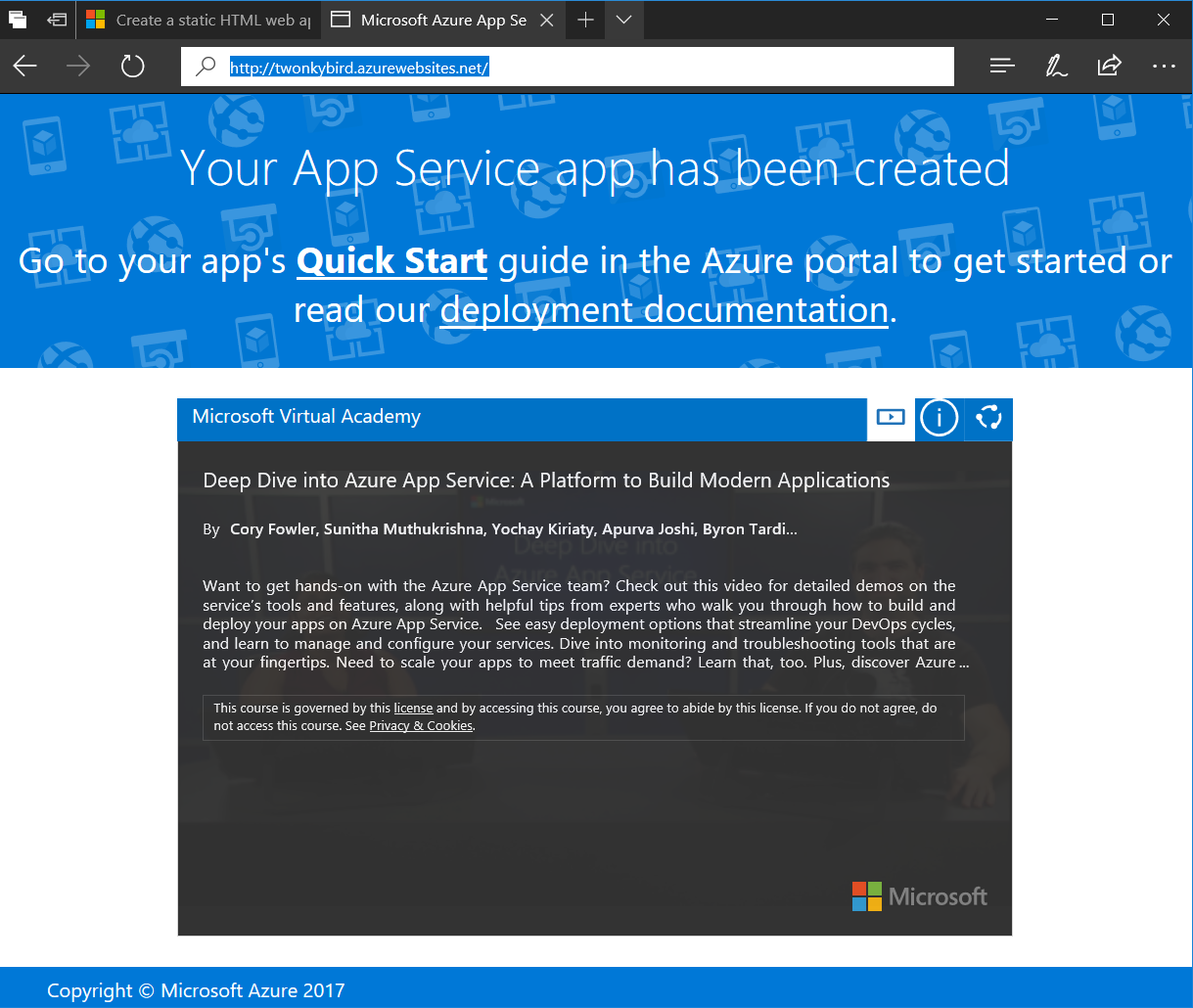
az appservice plan create --name quickStartPlan --resource-group $rg --sku FREE

* You should see output JSON when the command succeeds

### Create the Web App

* Create the web app. The name for the web app must be globally unique as it forms part of the FQDN. You will be prompted to change it if it already exists.

az webapp create --name $appName --resource-group $rg --plan quickStartPlan

* Again, there will be output JSON once the command succeeds
* Open your web browser and navigate to http://<appName>.azurewebsites.net. (You will also find the link in the new web app in the Azure portal.)
  1. 

### Create the Git deployment access point

* Create the deployment point

deployuri=$(az webapp deployment source config-local-git --name $appName --resource-group $rg --query url --output tsv)

echo $deployuri

* The first command create an https endpoint similar to below, and then saves that value into the $deployuri variable. The second prints the variable to screen, e.g.

https://<username>@<appName>.scm.azurewebsites.net/<appName>.git

* Notice the scm section of the URL - this is the Kudu address and we’ll come back to that later

### Add a Git remote called “azure” and then push to the Web App

* Create the remote, calling it azure

git remote add azure $deployuri

(Typing git remote afterwards will list the new azure remote as well as the origin remote, i.e. github.)

* Push the local html repo up to the azure remote

git push azure master

* + (Note that master is the name of the Git branch that we are pushing up to Azure)
  + Refresh the web page and confirm that it has changed

### Change the html and push it up again

* Edit the index.html to change the Twitter account to your own. You can use nano index.html, or vi index.html for those familiar with using terminal editors. Or you may go into the web app in the portal and use the App Service Editor in the blade and edit the index.html directly in the browser.
* Push the changes up to the azure remote

git commit -a -m "Description of the change"

git push azure master

* Refresh the web page and see if it has been changed

## WebApp Ecosystem

* Browse the blade in the portal, checking out deployment slots, scale up and out, App Service Editor and Application Insights
* Click on your website link. Access Kudu by inserting scm before azurewebsites.net, i.e. https://<website>.scm.azurewebsites.net/. More and more functionality is now surfaced in the portal, but there is still some additional extensibility here.

## CLI 2.0

* Browse around the az commands. Run az by itself to show the base commands. CLI 2.0 has autocomplete for commands, for switches and for parameter values. The --help gives contextual help dependant on where you are.
* Try the various outputs, e.g.
  + az group list
  + az group list --output jsonc
  + az group list --output tsv
  + az group list --output table

You can define your default output type using az configure. Different outputs make sense in different situations, and give flexibility to integrate with bash and python scripts.

* Let’s pull out some single key:value pairs using JMESPATH queries:

az webapp deployment user show

az webapp deployment user show --query “publishingUserName” --output tsv

* The following selects the right web app from a list, and then gets the id for it:

az webapp list --query "[?name == \`$appName\`].id" --output tsv

* Or we can create a custom table

az webapp list --query "[].{name:name, state:state, web:defaultHostName, region:location}" --output table

# Logic Apps

In this section we will use logic apps to poll for tweets from the presenter, and automatically send that content out from your own Twitter account.

* In the Azure101PaaS resource group, click Add and find Logic Apps
* Create a new Logic App in West Europe
* Click into it and select Twitter for the new tweet trigger
  + Sign in to Twitter
  + Search on tweets from @RichCheneyAzure with the hashtag #azure101
  + Set frequency to every two minutes
* Add a condition for the username to be RichCheneyAzure
* Click on new action, search on tweet and pick Post a Tweet, and in the Tweet Text dialog box select Tweet Text
* The trainer will take a photo and tweet using the #azure101 hashtag
* The trigger history should eventually show a status other than skipped
* Refreshing your webpage should show the updated twitter feed