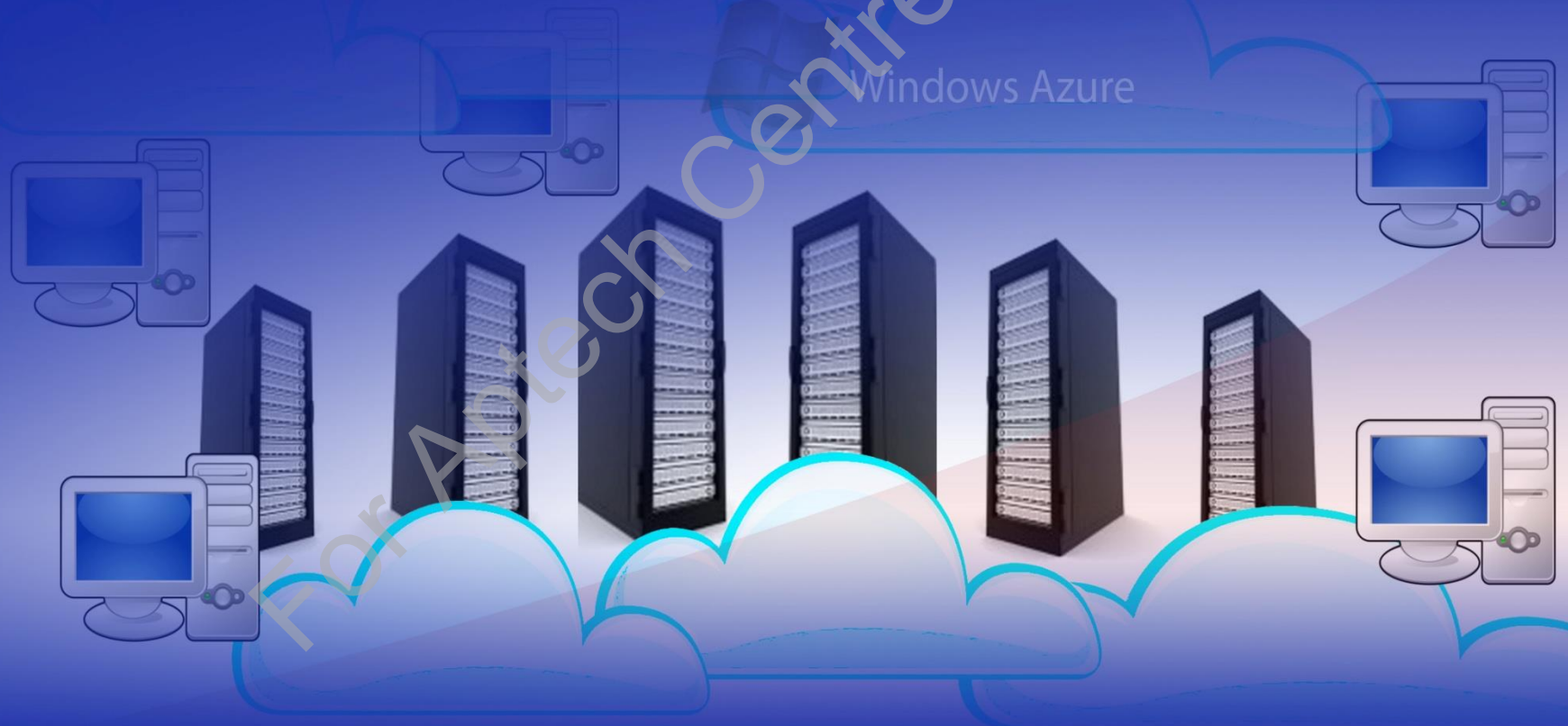
Enterprise Application Development Using Windows Azure and Web Services

Session 15

Continuous Deployment



Learning Objectives

- Describe an overview of Git and TFS
- Explain the process of using Git in Azure
- Define and describe TFS
- Describe Azure diagnostics

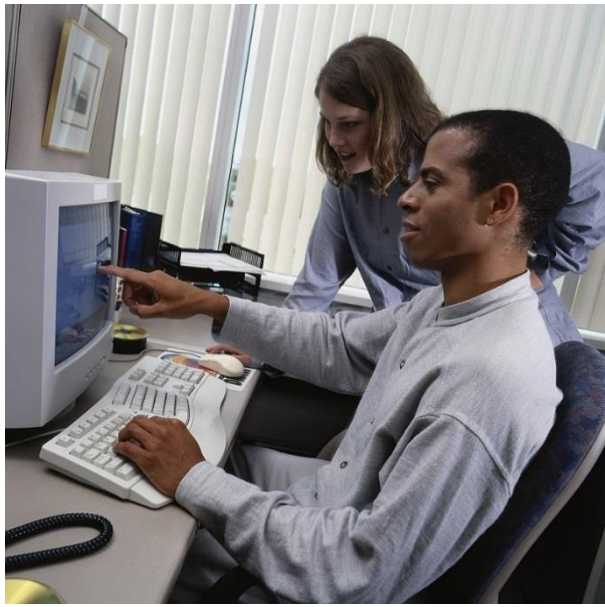
Introduction 1-4

- **Software development projects**:
  - Involves more than one developer working in a team.
  - Number of developers who access and manipulate the files at the same time, it can result in chaos and confusion. To avoid this, version control is used.

- **Version control**:
  - Is also know as source control.
  - Involves managing changes to the assets of a system or project such as programs, documents, reports, and so on.
  - Enables to organize and control the changes or revisions made.

Introduction 2-4

- Version control systems:
  - Have one main repository for all the project files.
    - A software repository is a location where software packages or files are maintained and retrieved when required.
  - Enables development team members to check in and check out files.
  - Automatically monitors which users changed the files, during what date/time they were changed, and what changes were made.



Introduction 3-4

Inserting comments:

- There is also an option to insert comments for each change made so that other team members can view the history and retrieve a desired file based on the comments.

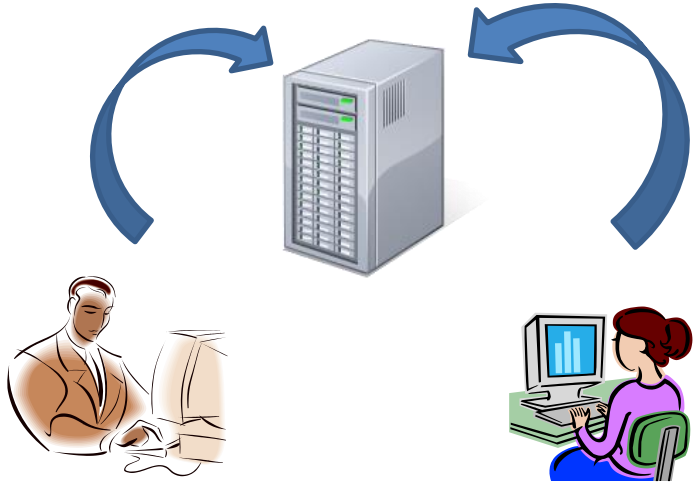
- **A software repository:**

  - It is a location where software packages or files are maintained and retrieved when required.

Introduction 4-4

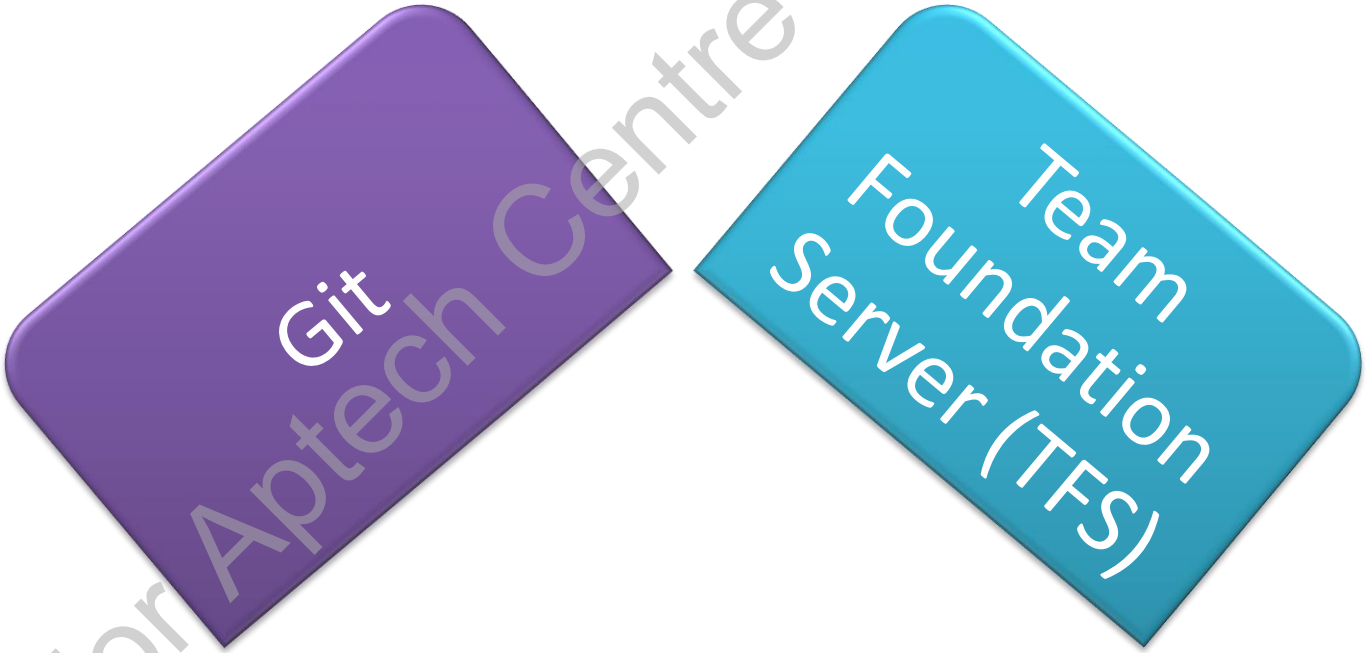
- Some version control systems also enable merging changes to the same file.

- The system will merge both sets of modifications, resulting in a new file when two or more developers work locally on the same file at the same time or they push the files into the main repository.



Version Control Software Tools

- Following are two popular version control software tools:



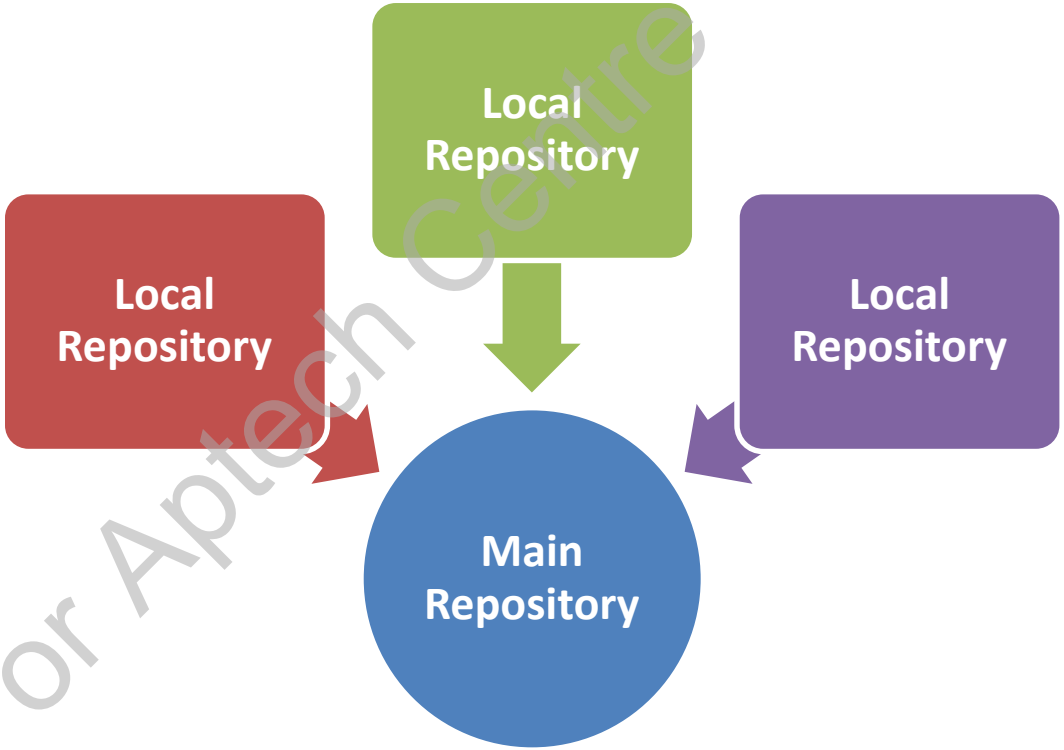
Git 1-5

Git

- Is an open source distributed version control system.
- Allows each developer in the team to have a copy of the local source repository and work even when there is no connectivity.
- Allows the developers to perform version control operations such as viewing or maintaining history and comparing different versions of a file without a network connection.
- Provides developers a flexible workflow for creating repositories to allow the developers to save code.
- Enables code from the local repositories to be synchronized with the software repository.

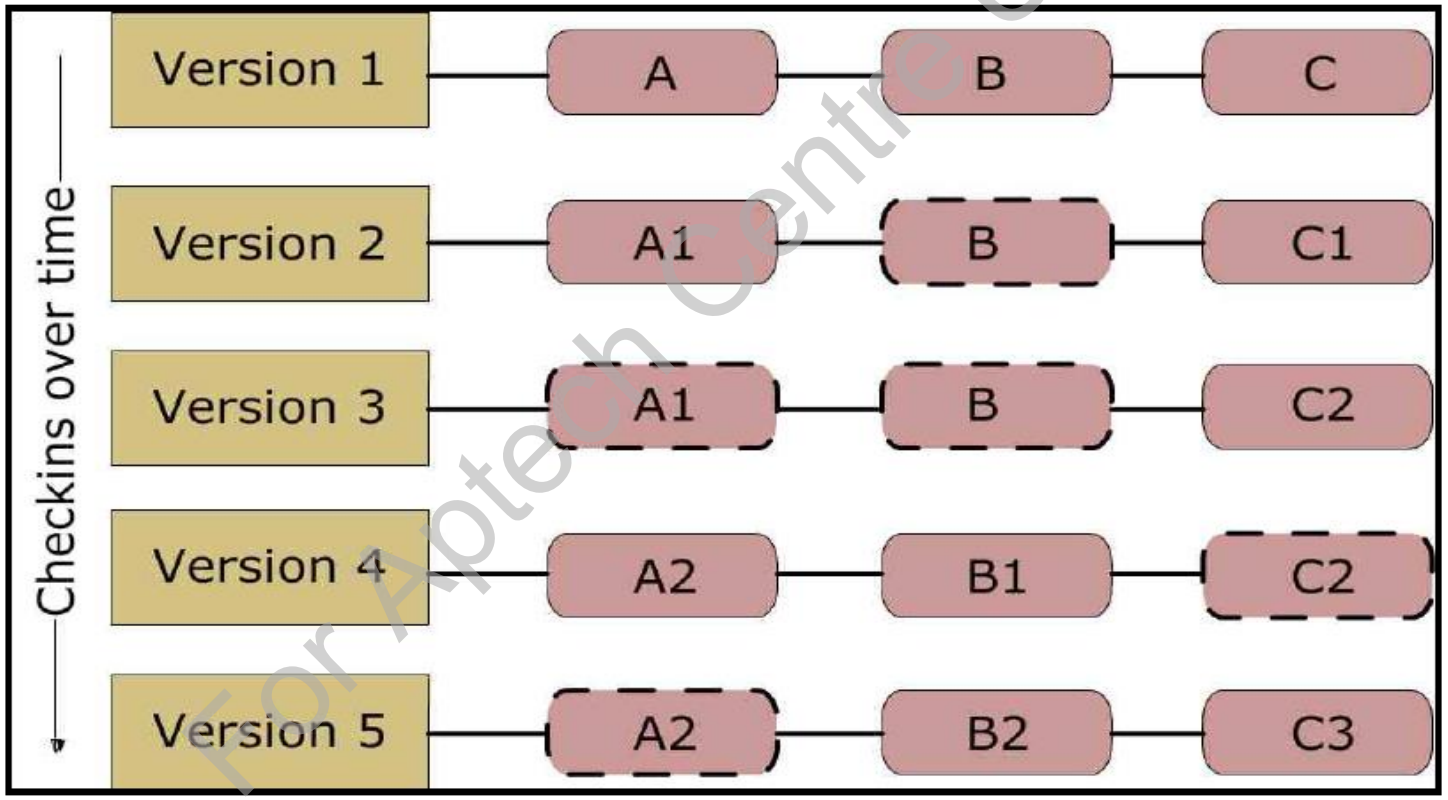
Git 2-5

- Following figure shows repository structure in the Git framework:



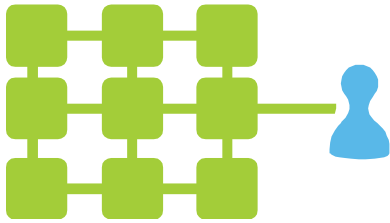
Git 3-5

- Git is also helpful where code is distributed across many repositories as shown in the following figure:



Git 4-5

- **Branch**:
  - Means a parallel version of the software repository.
  - Each developer can publish, merge, or dispose of the branch.
  - Is stored within the repository, but does not affect the primary or master branch, thus enabling one to continue working without affecting the live version.
  - When the changes to be made are completed, your branch can be merged back into the master branch to publish the changes.
  - The default branch in Git is called **master**.



Git 5-5

- **Git**:
  - Can be used with:

  - Provides a powerful Distributed Version Control System (DVCS) feature that allows each developer to work on a local copy of an application.
  - Needs separate tools such as Visual Studio to simplify work as its user interface is complicated.

TFS

- **TFS:**
  - Is a Microsoft product that covers the entire ALM including:

    - Source code management through Team Foundation Version Control (TFVC) or Git
    - Reporting
    - Requirements management
    - Project management (for both agile software development and waterfall teams)
    - Automated builds
    - Lab management
    - Testing
    - Release management capabilities

Difference between Git and TFS 1-2

- Following is the key difference between Git and TFS:

Git	TFS
• Is a distributed version control tool.	• Is a centralized version control tool.

Difference between Git and TFS 2-2

- Following are the two types of Version Control Systems:

Centralized Version Control System	Distributed Version Control System
• A single server acts as the code repository. • In this approach, all operations take place on the server. • The operations need a connection to the server. • As a developer, you check out a working copy, which is a snapshot of the code at a given point in time.	• There is no central repository. • You define a **master** repository and perform a clone operation on it. • The developer clones the repository (that is, makes a duplicate copy of it) on the local machine. • This clone contains all the data in the repository. • Once this is done, the developer can work offline, and work from anywhere.

Installing Git 1-2

- Steps to install Git for Windows are:

Step 1
- Go to the http://msysgit.github.io/ or http://git-scm.com/download/win and download Git for Windows.

Step 2
- Click the downloaded executable file to begin the installation process. Figure shows the installation wizard.

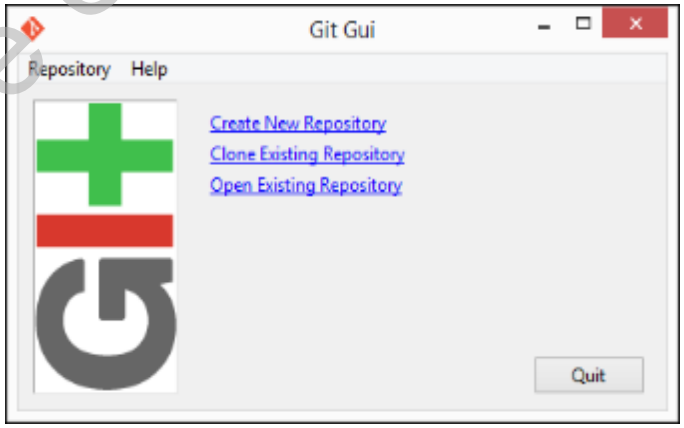
Step 3
- Follow the step-by-step procedure as shown in the installation wizard.

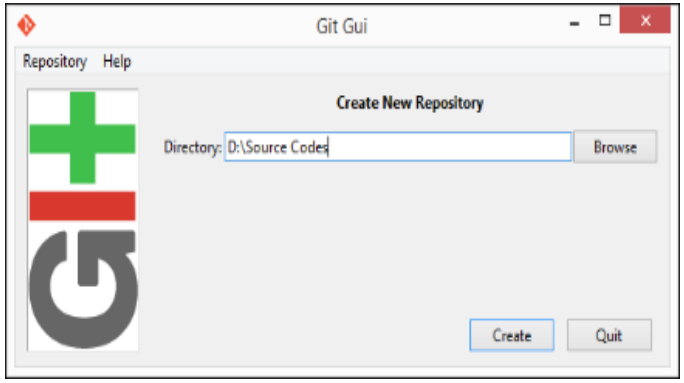
Step 4
- Click **Finish** to exit the setup.

Installing Git 2-2

- Steps to create a local repository are:

Step 1	Go to the folder containing your software project.
Step 2	Right-click and select **GitGui** from the shortcut menu. The GitGui dialog box is displayed as shown in figure.
Step 3	Click **Create New Repository** and specify the folder name as shown in figure. The repository will be created.





Publishing from Git to Azure Web Sites

- **Azure Web Sites**:
  - Can host applications created in any of the one supported programming languages or frameworks such as ASP.NET, PHP, and so on.
  - Provides support for continuous deployment through source control tools such as Git and TFS.

- Publishing content from Git to Azure Web Sites involves a series of steps:
  - Create a local repository as described earlier.
  - Add a Web page to the local repository.
  - Enable the Web site repository.
  - Deploy the project.

Adding a Web Page to the Local Repository 1-2

- Following are the steps to add a Web page to the repository:

Step 1

Create an HTML file named **home.html** in the root of the Git repository (for example, the **Source Codes** repository created earlier).

Step 2

Add the following text in the HTML file and save it:

```
<html>
Hi, a demo usingGit.
</html>
```

This is now a Web page.

Step 3

Right-click the repository folder, **Source Codes**, and click **Git Bash**. The Git Bash emulation is displayed which allows you to run Git from the command line.

Adding a Web Page to the Local Repository 2-2

Step 4

Type the following command to add the home.html file to the repository:

```
git add home.html
```

The file will be added but the repository is still not committed.

Step 5

Commit the change to the repository using the following command:

```
git commit -m "Addedhome.html to the local repository"
```

The file will be successfully committed into the repository.

Enabling the Web Site Repository

- Consider that you have created and hosted a site named **sampledemoweb** on **Azure Web Sites**.
- Steps to enable a Git repository for a Web site by using the Azure portal are:

Step 1	• Login to the Azure portal.
Step 2	• Select **WEB SITES** on the left of the page and select the Web site **sampledemoweb** as shown in figure.
Step 3	• Select the **DASHBOARD** tab.
Step 4	• Click **Set up deployment from source control** in the **quick glance** section to enable a repository.
Step 5	• Select **LocalGit** and click **Next**. You may be asked to create user credentials for connecting to the repository in the future.

Deploying and Troubleshooting 1-3

- After you create and enable a repository, the next task is to deploy your project.
- Steps used to publish the Web site to Azure Web Sites using Local Git are:

Step 1

Launch the Bash emulation window and type the following command:

```
git remote add azuresite
https://aptechuser@sampledemoweb.scm.azurewebsites.net:443/sampledemoweb.git
```

The purpose of the remote command is to add a named reference or alias to a remote repository.

Here, you have specified 'azuresite' as a reference for your Azure Web Site repository.

The remote Web repository is also called **remote** for short.

Deploying and Troubleshooting 2-3

Step 2

Type the following command to push the repository contents from the local repository to the 'azuresite' remote:

```
git push azuresite master
```

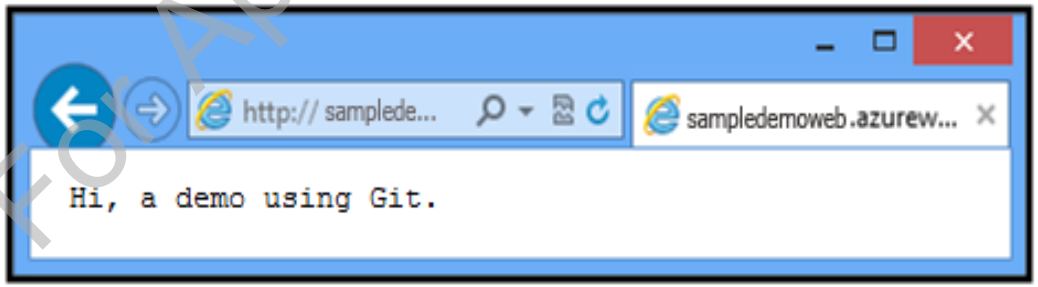
The contents of the local repository will be pushed to the portal.

Step 3

Click the **BROWSE** link at the bottom of the portal to confirm that the deployment of **home.html** was successful.

The page displays the text that you had added into **home.html**.

Following figure shows the Web page added from the local Git repository:



Deploying and Troubleshooting 3-3

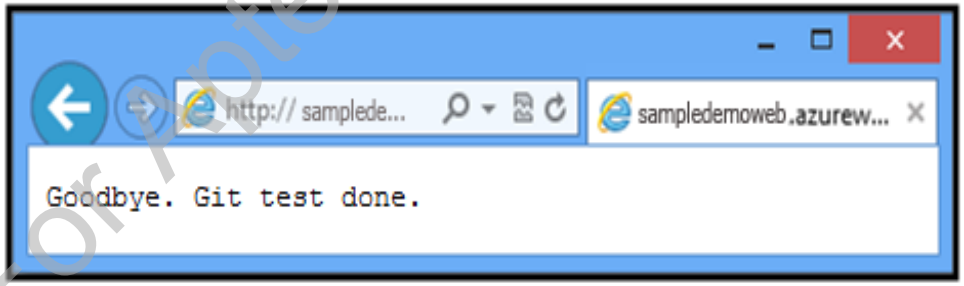
Step 4

Edit the contents of the HTML file, **home.html**, to add 'Goodbye. Git test done' and save the file.

To add and commit the changes, launch the Bash emulation window and type the commands as shown:

```
git add index.html
git commit -m "Farewell"
git push azuresite master
```

Press **Ctrl**+**F5** and refresh the browser displaying the portal. The page shows the changes made to the HTML file as shown in the following figure:



Troubleshooting

- Table lists the common errors or problems faced by developers while publishing an Azure Web site using Git:

Symptom	Cause	Solution
Trouble in resolving host 'hostname'	This error occurs if incorrect address information is entered when creating the 'azuresite' remote.	• To list all remotes with the associated URL, use the `gitremote -v` command. • Check if the correct URL for 'azuresite' remote is entered. • Use the correct URL to remove and recreate this remote.
No refs in common and none specified. Specify a branch such as `master`.	This error occurs when a branch performing a Git push operation is not specified and the push default value used by Git is not set.	Repeat the push operation by specifying the master branch. For example: `git push azuresite master`
`srcrefspec [branchname]` does not match any.	This error occurs when a branch other than master on the 'azuresite' remote is pushed.	Repeat the push operation by specifying the master branch. For example: `git push azuresite master`

Using TFS 1-3

- Developers often use TFS for source control, bug tracking, requirements gathering and managing the complete lifecycle of software development.

- Following are the two ways to work with TFS:

On-premises

Online
(which is a cloud service hosted by Microsoft)

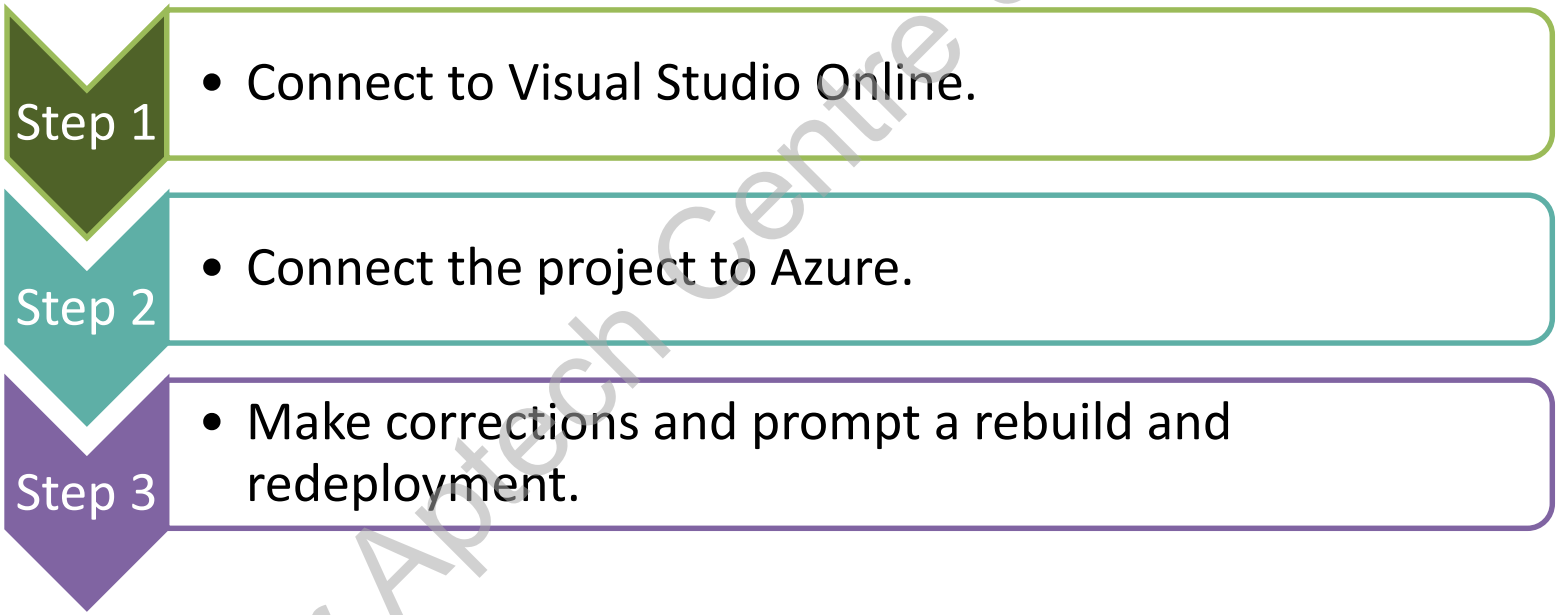
The online version is called Visual Studio Online (VSO).

Using TFS 2-3

- The TFS cloud service is backed by Microsoft's cloud platform, Windows Azure.
- No need to download or install any server or SDKs to work with VSO.
- Log in using your Microsoft Account and start developing.
- You can access the TFS service at the URL: tfs.visualstudio.com.
- It publishes the build for connecting Windows Azure Web site whenever a new check-in is devised.

Using TFS 3-3

- Steps to set up a cloud service, build, and deploy to Azure by using VSO are:



Connecting to Visual Studio Online 1-2

- **Visual Studio Online (VSO)**:
  - Administers everything from cloud-based ALM solutions, issuing tracking to automated builds and load testing, and hosting code repository.
  - Provides the user with the freedom to use applications from anywhere owing to its portability.
  - Can be configured and installed on a single server by enabling users through the cloud infrastructure.

Connecting to Visual Studio Online 2-2

- **System Requirements for linking VSO with Windows Azure**:
  - Visual Studio Online can be linked to Windows Azure. Ensure these basic requirements in order to get it up and running:

Pay-As-You-Go subscription	Microsoft Account
• You cannot link Windows Azure subscriptions to MSDN subscriber benefits. • Hence, you need to create a new Pay-As-You-Go subscription.	• The Microsoft Account that you use for Visual Studio Online must be the Co-Administrator or Service Administrator on the subscription.

  - Then, add or adjust account users in Visual Studio Online.

Linking Visual Studio Online to Windows Azure Subscription 1-2

- Steps to link Visual Studio Online to Windows Azure Subscription:

Step 1

- Logon to the Windows Azure subscription at https://manage.windowsazure.com with the same login as the owner of Visual Studio Online tenant.

Step 2

- Click **Visual Studio Online**.

Step 3

- Click **Create or Link a Visual Studio Online Account** and a new menu will open.

Linking Visual Studio Online to Windows Azure Subscription 2-2

Step 4

- Next, click **Link to Existing**.
- Here, you will have to confirm the Visual Studio Online tenant that is displayed is correct.

Step 5

- Pick the type of subscription for Visual Studio Online services.
- At this stage, the account can be unlinked or relinked to a different subscription.

Step 6

- Click **Link Account**. Windows Azure will show a confirmation whether it is linking the Visual Studio Online tenant.

Performing Project Check-in 1-2

- Steps to perform a project check-in are:

Step 1	• Launch Visual Studio Online.
Step 2	• Open an existing solution to be deployed or else, create a new one.
Step 3	• Using Solution Explorer pane, open the shortcut menu for the solution.
Step 4	• Click **Add Solution to Source Control**. The **Add Solution to Source Control** dialog box is displayed. • You can choose to accept the defaults or customize various options.

Performing Project Check-in 2-2

Step 5

- Select **OK** when done.
- After it finishes adding, the Solution Explorer displays source control icons.

Step 6

- To perform the check in process, open the shortcut menu for the solution once again.
- This time you will see a new option, **Check In**.
- Click the **CheckIn** option on the shortcut menu.

Step 7

- Then, in the **Pending Changes** area of the Team Explorer, type a comment for check In and click **Check In**.
- While checking in, observe the options that have been selected.
- If certain changes are not included, click **Include All links**.

Connecting to Windows Azure 1-2

The project will be successfully checked in after following the steps:

- You can now connect the team project to Azure after you have a VSO project ready with some source code in it.
- The basic steps to be followed are as follows:

Step 1

- Go to the Azure portal and choose Website or cloud service.
- You might need to create a new one by choosing '+' or **Add** icon which is situated at the bottom left.
- Then, you will have to choose the Cloud Service or Website and then click **Quick Create**.

Step 2

- Click **Set up publishing with Visual Studio Online**.

Step 3

- Type the name of the Visual Studio Online account in the text box.

Connecting to Windows Azure 2-2

Step 4	Step 5	Step 6
• Click **Authorize Now** link. At this stage, you may need to sign in to authorize.	• When the OAuth pop-up dialog box appears, select **Accept** for authorizing Azure to configure the team project in VSO.	• Once authorization is successful, you will see a drop-down list that has names of all the Visual Studio Online team projects. • Choose the appropriate team project that you click the wizard's checkmark button. • Once the project is connected, instructions to cross-check the changes will appear. • The next time you login to Visual Studio Online, it will build and deploy the project to Azure.

Rebuilding and Redeploying 1-3

- Steps to activate a rebuild and redeploy of the project are:

Step 1

- Click the **Source Control Explorer** link in the Team Explorer pane of Visual Studio Online.

Step 2

- Browse through, select, and open your solution file.

Step 3

- Choose any file, make changes to it, and save the file.

Rebuilding and Redeploying 2-3

Step 4

- In Team Explorer, click **Pending Changes** and type an appropriate comment for the changes made.

Step 5

- Select **Check In**.

Step 6

- Return to the Team Explorer home page using the **Home** button.

Step 7

- Choose the **Builds** link to view the builds in progress. The Team Explorer shows that a build has been triggered for your check-in.

Rebuilding and Redeploying 3-3

Step 8

- Select the name and double-click to open it. This file will provide a detailed information of the progress of the build.
- You can view the build definition that was created when you link TFS to Azure.

Step 9

- Select the shortcut menu to view the build definition and select **Edit Build Definition**.
- In the **Trigger** tab, you can view the build definition.
- The definition has been set to build whenever you check-in.
- In the **Process** tab, you will be able to view the deployment environment. Here, you will notice the name of the cloud service.

Azure Diagnostics 1-8

- A Windows Azure hosted service may often consist of several instances of roles.
- These instances may run 24 hours in a remote Windows Azure datacenter.
- It is essential to monitor these instances non-intrusively in order to detect failure and take suitable measures.
- To do this, diagnostics is often performed for Azure hosted services and applications.

Azure Diagnostics 2-8

Windows Azure Diagnostics

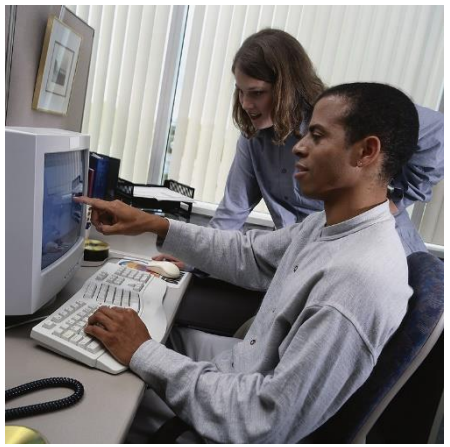
- Enables developers to collect and analyze diagnostic data from a worker role or Web role running in Azure.

The Windows Azure Diagnostics Library

- Is built into Windows Azure SDK for .NET.
- Developers can configure Diagnostics either before deployment or at runtime within Visual Studio 2013 using the Azure SDK.

Azure Diagnostics 3-8

- Through Visual Studio, developers:
  - Can customize the diagnostics data that is collected for a role that runs in Azure.
  - Need to change diagnostics settings in Visual Studio by changing the configuration file (**diagnostics.wadcfg**) so that when they next deploy their cloud service, the new settings are automatically reflected.



Azure Diagnostics 4-8

- Steps to configure diagnostics in Visual Studio 2013 are:

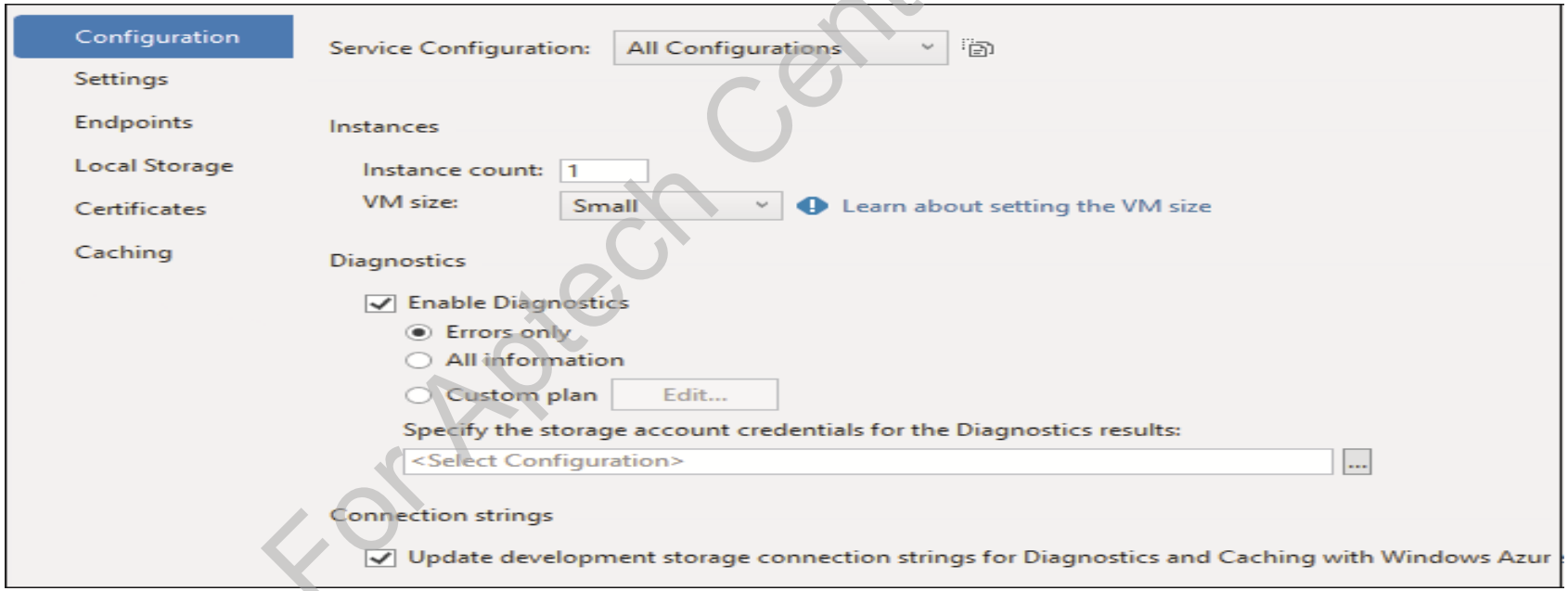
Step 1

- Click **Properties on the shortcut menu of the role** and select the **Configuration** tab.

Azure Diagnostics 5-8

Step 2

- In the **Diagnostics** section, select the **Enable Diagnostics** check box if it is not already selected as shown in figure.



Azure Diagnostics 6-8

Step 3

- Click **Custom plan** button to customize the settings and then, click **Edit**. This displays the **Diagnostics configuration** dialog box containing tabs for each source of diagnostic data.

Step 4

- Set the log level to one of the following values (in order from least information to most): **Critical**, **Error**, **Warning**, **Information**, or **Verbose**.

Step 5

- Type a value for the buffer size and transfer period for application logs.
- Applications generate application logs using the `System.Diagnostics` API.
- In order to generate data in these logs from your application code, you need to add a reference to `System.Diagnostics.dll`, and use one of the static methods defined in the Trace class.

Azure Diagnostics 7-8

Step 6

- Select the **Event logs** tab and then, select the check boxes for the types of events that you want to track.
- The categories that are displayed correspond to various filters in the Windows Event Viewer.
- Again, set the log level to one of the following values (in order from least information to most): **Critical**, **Error**, **Warning**, **Information**, or **Verbose**.

Step 7

- Click the **Performance counters** tab and select the check boxes for the performance counters that you want to track.
- To track a performance counter that is missing in the list, enter it by using the suggested format and click **Add**.

Step 8

- Click the **Infrastructure logs** tab and specify the settings that you want. The logs indicate settings pertaining to the infrastructure of Azure Diagnostics.

Azure Diagnostics 8-8

Step 9

- Finally, click the **Log directories** tab, specify the data collected from log directories for IIS requests and crash dumps, and then click **OK** to close the dialog box.

Step 10

- You need to re-deploy your cloud service and test it.
- There are two ways in which one can view the diagnostics data, namely, a report generated by Visual Studio 2013 or through tables in the storage account.

Summary 1-2

- Git is a distributed version control system that allows each developer to have a copy of the local source repository and work without connectivity.
- TFS is a Microsoft product that covers the entire Application Lifecycle Management and provides users with the support for distributing source control.
- The key difference between Git and TFS is that TFS is a centralized version control tool, whereas Git is a distributed version control tool.
- Publishing content from Git to Azure Websites involves tasks such as, adding a Web page to the local repository, enabling the Website repository, and deploying the project.

Summary 2-2

- The online version of TFS is called Visual Studio Online (VSO).
- Visual Studio Online administers everything from cloud-based ALM solution, issuing tracking to automated builds and load testing and hosting code repositories.
- Windows Azure Diagnostics enables developers to collect and analyze diagnostic data from a worker role or Web role running in Azure.
- The Windows Azure Diagnostics library is built into Windows Azure SDK for .NET.