# Prerequisites for Gradle Build

## On-premises TFS Server

* + TFS 2013
  + TFS Build 2013
  + Code should be checked into GIT Source Control

## VS Online / TFS Service

* + Hosted Build Controller cannot be used. Hence configure your on-premises Build Server to connect to VSOnline.
  + Code should be checked into GIT Source Control

## Client Machine

* + Team Explorer 2013

## Server hosting Team Foundation Build Agent

* + Java to be installed.
    - JDK version 🡪 1.5 or higher
    - JAVA\_HOME environment variable to be set
  + Gradle version 🡪 1.9 or higher
  + For ‘gradle’ command:
    - Gradle to be installed
    - GRADLE\_HOME environment variable to be set
  + For ‘gradlew’ command:
    - Needs Internet connection, esp. for first time run – Gradle’s characteristic

# Prerequisites for Deploying Build Binaries to Azure Storage

## Server hosting Team Foundation Build Agent

* + AzCopy to be installed.
    - AZCOPY environment variable to be set to the path where AzCopy.exe is available
    - AZCOPY 2.0.0.0 or higher

## Azure Storage

* + Storage Account should be created
  + Access Key to be provided to Build Template
  + Container should be created

**NOTE:** If any of the above mentioned Environment variables are set newly, then the TFS Build Service should be restarted. Else TFS Build cannot resolve the Environment variables

# Prerequisites for Downloading from Azure Storage to Azure VMs

## Server hosting Team Foundation Build Agent

* + **For Linux VM:**
    - OpenSSH or any tool that provides an SSH client for Windows
    - Path of ssh.exe to be added to environment variable %PATH% on Build Agent.
    - SSH private key (PPK file) to be saved on the File system.
      * If there are multiple Build Agents configured, you can choose to store the private key in a shared drive to which all Build Agents have access to.
      * This location should be provided in the Azure VM Config XML.
      * See <http://www.windowsazure.com/en-us/manage/linux/how-to-guides/ssh-into-linux/> to know how to create .PPK file.
  + **For Windows VM:**
    - Windows Azure PowerShell
    - WinRM Config for Client as follows:
      * winrm quickconfig
      * winrm set winrm/config/client '@{AllowUnencrypted="true"}' # Not Required for HTTPS connection
      * winrm set winrm/config/client/auth '@{Basic="true"}'
      * winrm set winrm/config/client '@{TrustedHosts*="<DNS of VMs comma separated>*"}'
  + The following will have to be present on Build Server File System for the Build Template to work:
    - Include the Assembly
    - Azure VM Config XML
      * Store on the Build Agent or a path that the Build Agent has access to
      * This path should be provided to the Build Template
    - Windows VM – Download Script
    - Linux VM – Download Script

## Client Machine

* + Latest .NET Framework
    - This will include MSBuild.exe which should be used to create the .dll file for the ‘DeploymentToAzureVM.csproj’
    - Execute the command:

MSBuild.exe DeploymentToAzureVM.csproj

This will create the .dll in the bin\Debug folder

* + Windows Azure Powershell
    - Create endpoints from the Windows PowerShell window using the Azure cmdlets
    - **For Linux VM:**

If the VMs are behind load balancers, then create a new endpoint with a new public port, which maps to the loadbalanced endpoint’s SSH private port. Ex:

Get-Azurevm -ServiceName $serviceName -Name $hostName

| Add-AzureEndpoint -Name "directSsh" -Protocol tcp

-LocalPort $sshPrivatePort

-PublicPort $newPublicPort

| Update-AzureVM

* + - **For Windows VM:**

If the VMs are behind load balancers, then create a new endpoint with a new public port, which maps to the loadbalanced endpoint’s private port. Ex:

Get-Azurevm -ServiceName $serviceName -Name $hostName

| Add-AzureEndpoint -Name "winRM" -Protocol tcp

-LocalPort $winRMListenerPort

-PublicPort $newPublicPort

 | Update-AzureVM

## Azure Linux VM

* + SSH server to be running
  + Standalone SSH endpoint needs to be created for every VM, which is not behind the load balancer. You can create it on the Azure Management portal / via Azure powershell cmdlet.
  + If the VM is behind a load balancer, then a mapping needs to be created between the public and private ports. See the ‘Client Machine’ section on how to create this mapping.

## Azure Windows VM

* + WinRM Config as follows (The following should be executed on Windows Azure Elevated Powershell / Command Prompt window):
    - winrm quickconfig
    - winrm set winrm/config/service '@{AllowUnencrypted="true"}' # Not Required for HTTPS connection
    - winrm set winrm/config/service/auth '@{Basic="true"}'
    - winrm set winrm/config/service/auth '@{CredSSP="True"}' #Not Required
  + Change the Remote Management Exception on Windows Firewall for Remote IPs. You can do this from Advanced Windows Firewall snap-in.
  + Standalone WinRM endpoint needs to be created for every VM, which is not behind the load balancer. The private port of this endpoint should be the same as the WinRM listener port.
  + If the VMs are behind a load balancer, then Standalone WinRM endpoints need to be created for every VM. Each VM will have a different public port. You can create it on the Windows Azure Mangement portal or from the Windows Azure Powershell CLI. For details on how to use the Azure Powershell CLI, see the ‘Client Machine’ section.