

Testen in der Cloud

Altbekanntes oder neue
Herausforderungen?



Abstract

Wenn man Marktforschungsunternehmen glaubt, wird der Trend in der IT in den nächsten Jahren Cloud Computing sein. Microsoft bietet mit der Windows Azure Plattform eine Cloud-Umgebung, die sich ideal für Anwendungen auf der Technologie des Konzerns aus Redmond eignet. In dieser Session stellt Rainer Stropek dar, welche neuen Herausforderungen hinsichtlich Testen bei Einbeziehung von Cloud Computing, insbesondere Windows Azure, zu meistern sind. Er zeigt, welche Möglichkeiten zur Testautomatisierung bereit stehen und diskutiert auch die kaufmännischen Aspekte, die sich aus dem Pay-per-use Modell von Cloud Computing für Test- und Entwicklungsumgebungen ergeben.

According to market research organizations like Gartner or IDC cloud computing will be a big trend over the next years. With the Windows Azure Platform Microsoft offers a cloud computing environment that is perfectly suited for applications that are based on technologies like .NET or SQL Server. In this session Rainer Stropek discusses how using the cloud – especially Windows Azure – changes software testing. He shows possibilities for test automation and also covers pricing aspects with regards to test and development environments.



Introduction

- [software architects gmbh](http://www.software-architects.com)
- Rainer Stropek
 - Developer, Speaker, Trainer
 - MVP for Windows Azure
 - rainer@timecockpit.com
 -  [@rstropek](https://twitter.com/rstropek)



<http://www.timecockpit.com>

<http://www.software-architects.com>



Introduction

Testing for the cloud – similarities
and differences



Similarities and Differences

- Similarities
 - Just another .NET application
 - Just another Windows Server
 - Just another IIS
 - Just another SQL Server



Similarities and Differences

- Differences
 - Design, develop and test for **clusters**
 - Handle failures (i.e. **failover**)
 - Design, develop and test for **scalability**
 - Use the **elastic** nature of the cloud



Similarities and Differences

- Side topics
 - Cloud enables new tools for testing
 - E.g. [LoadStorm](#)
 - SLAs become more and more important
 - Who monitors the cloud?



On-Premise Cloud Emulators

DevFabric and DevStorage



DevFabric Introduction

- *Windows Azure Compute Emulator* aka **DevFabric**
 - Part of [Windows Azure SDK](#) → free
- Simulates Windows Azure during development process
 - For debugging purposes
 - To lower costs
 - For offline scenarios
- DevFabric ≠ Windows Azure
 - Can access all locally installed resources
 - Might not be available in the real cloud
 - DevFabric does not mitigate testing in the real cloud



DevFabric Introduction

- Prerequisites
 - [Windows Azure SDK](#) and Azure Tools for VS
 - Visual Studio 2010
 - IIS and SQL Server 2008 R2 (see also [MSDN](#))
- Installation
 - Install SDK and tools
 - Configure DevFabric (see also [MSDN](#))
 - Configure DevStorage (see later)
- DevFabric and DevStorage only support local use
 - Tip: Various articles about how to access DevFabric and DevStorage over the network are available (e.g. [Emmanuel's Blog](#))



Debugging With DevFabric

- Demo DevFabric in Visual Studio
 - F5-Experience

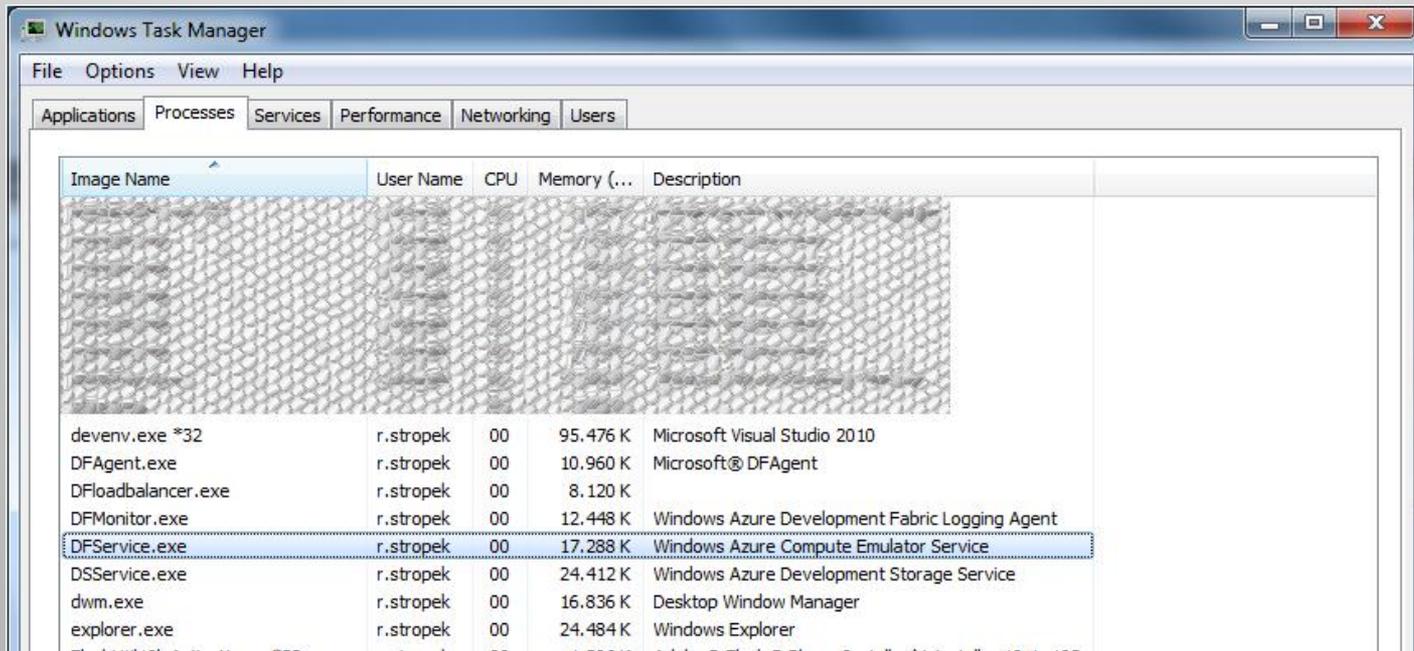


Image Name	User Name	CPU	Memory (...)	Description
devenv.exe *32	r.stropek	00	95,476 K	Microsoft Visual Studio 2010
DFAgent.exe	r.stropek	00	10,960 K	Microsoft® DFAgent
DFloadbalancer.exe	r.stropek	00	8,120 K	
DFMonitor.exe	r.stropek	00	12,448 K	Windows Azure Development Fabric Logging Agent
DFService.exe	r.stropek	00	17,288 K	Windows Azure Compute Emulator Service
DSService.exe	r.stropek	00	24,412 K	Windows Azure Development Storage Service
dwm.exe	r.stropek	00	16,836 K	Desktop Window Manager
explorer.exe	r.stropek	00	24,484 K	Windows Explorer
FltUserHostProcess.exe *32	r.stropek	00	1,888 K	Microsoft® Windows® File System Filter Driver Host Process



Profiling With DevFabric

- Most of today's leading profiler tools (e.g. ANTS Profiler) do not support applications running in DevFabric
 - Example of a working profiler: [YourKit](#) Profiler for .NET (with limitations)
- Tip: Build apps that run with and without DevFabric/Cloud
 - `RoleEnvironment.IsAvailable`



Automating DevFabric

Windows Azure SDK Deployment Tools

- [CSPack.exe](#)
 - Pack binaries for DevFabric or Azure deployment
 - Typically done by Visual Studio
- [CSRun.exe](#)
 - Deploys package to DevFabric and runs it
 - Typically done by Visual Studio
 - Tip: Testers can use CSRun to run an app without Visual Studio and sourcecode
- CSUpload.exe – Uploads VHDs to Azure
- [CSManage.exe](#)
 - Sample that shows how to automate Azure Service Management



DevFabric and Unit Tests

Tips

- Encapsulate logic that has to be unit tested into separate class libraries → testing as usual
- Include DevFabric in integration tests using CSPack/CSRun
- Build applications and services that can be run inside and outside of Azure

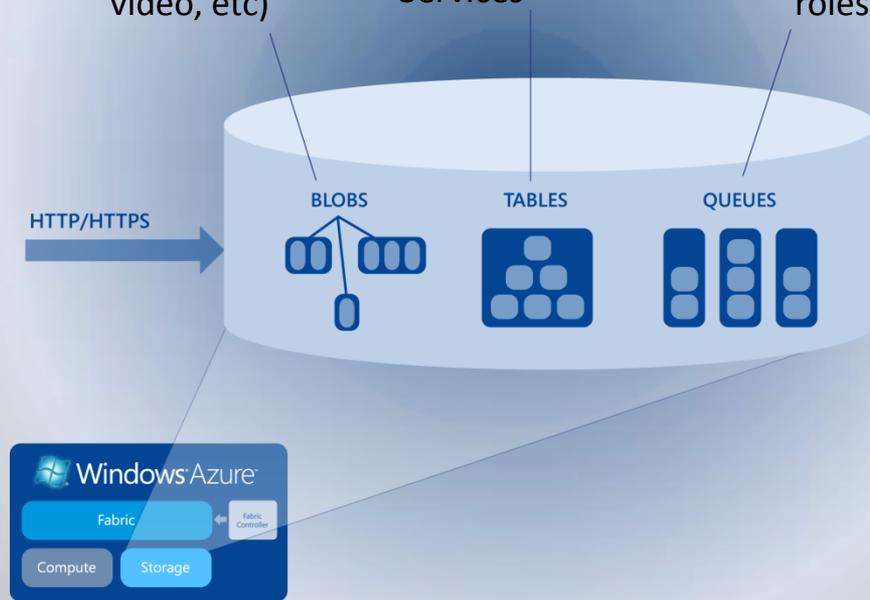


Storage in Azure

Blobs: large, unstructured data (audio, video, etc)

Tables: simply structured data, accessed using WCF Data Services

Queues: serially accessed messages or requests, allowing web-roles and worker-roles to interact

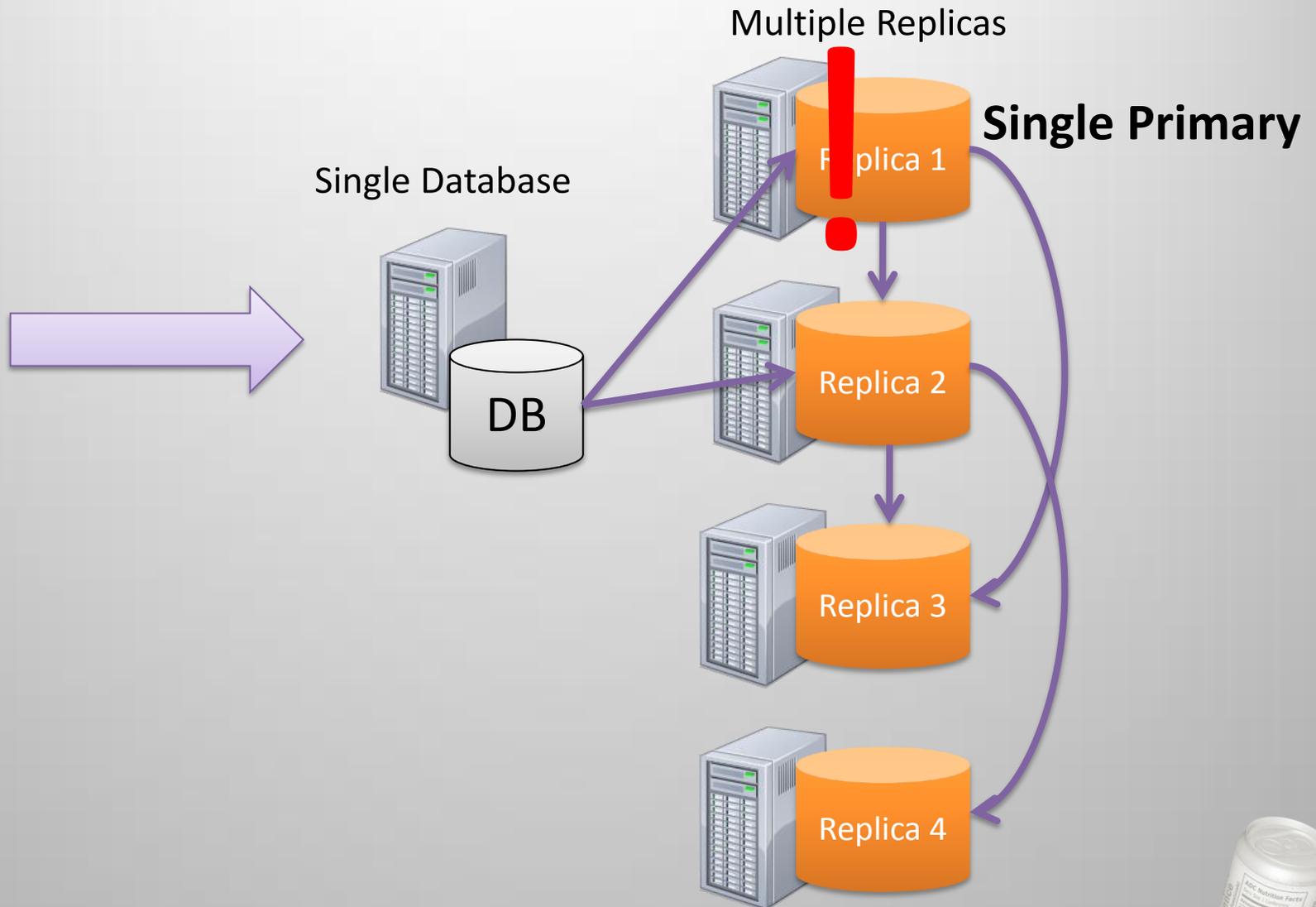


Windows Azure Storage

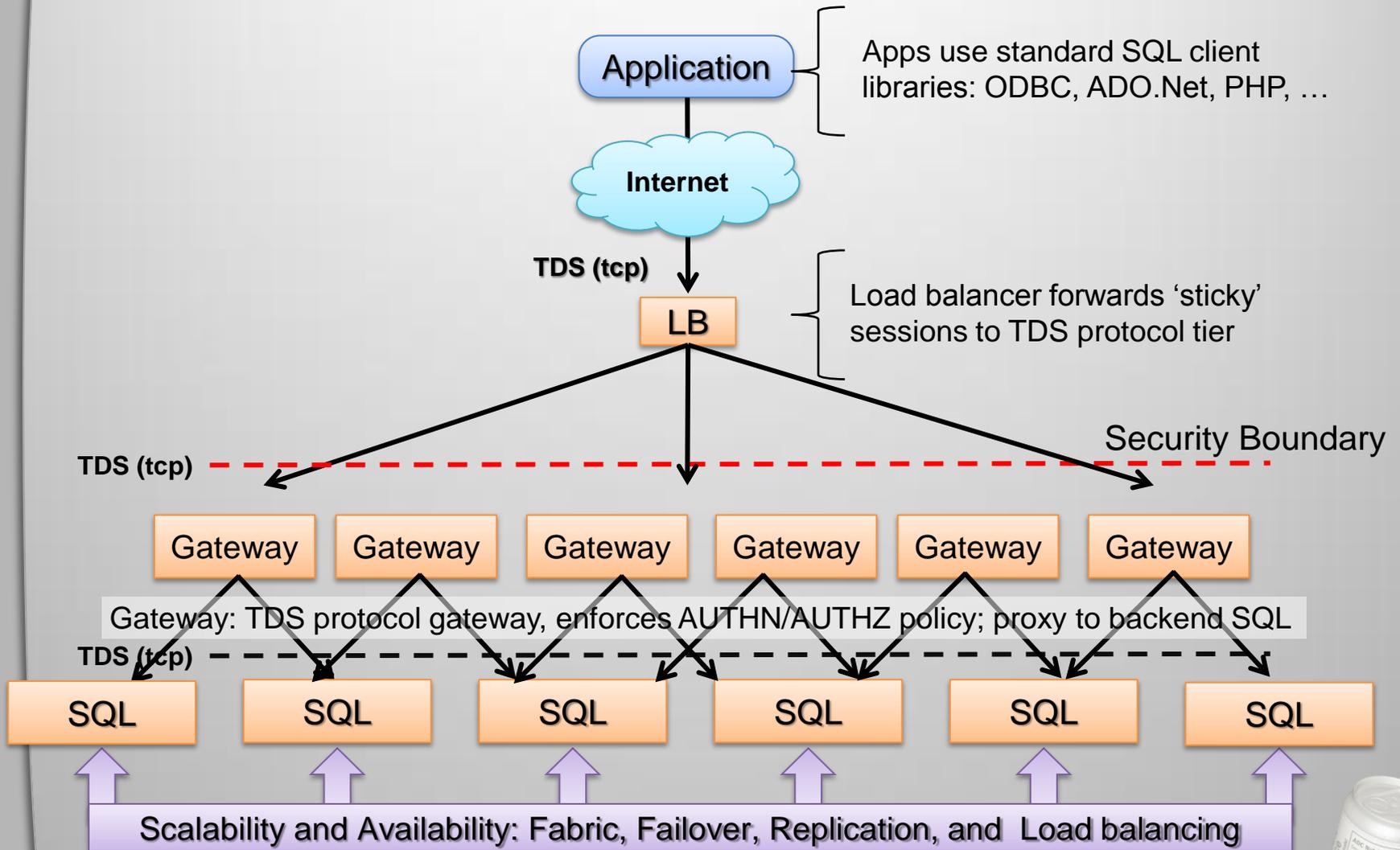
- *Windows Azure Storage Emulator* aka **DevStorage**
 - REST Services on `http://127.0.0.1:1000-10002`
 - `UseDevelopmentStorage=true`
 - Credentials and important tips see [MSDN](#)
- Storage Explorers
 - Visual Studio Server Explorer
 - 3rd party tools (e.g. [Cerebrata](#))



SQL Azure



SQL Azure



SQL Azure

Differences and Limitations

- Features
 - Only RDBMS, no SQL Agent, SSIS, SSRS (already in beta) or SSAS
 - No support for hardware-related features
 - No distributed queries or transactions
- Protocol
 - TDS 7.3 or later
 - No OLE DB support
 - Only TCP/IP protocol without MARS with encryption
- You need Management Studio 2008 R2
- Every table must have a clustered index
- Further details see [MSDN Guidelines and Limitations \(SQL Azure Database\)](#)



DevStorage

- Demo
 - DevStorage and Cloud storage with Cerebrata Cloud Storage Studio and Visual Studio 2010
 - SQL Azure with Management Studio 2008 R2
 - Fiddler with DevStorage and Cloud Storage

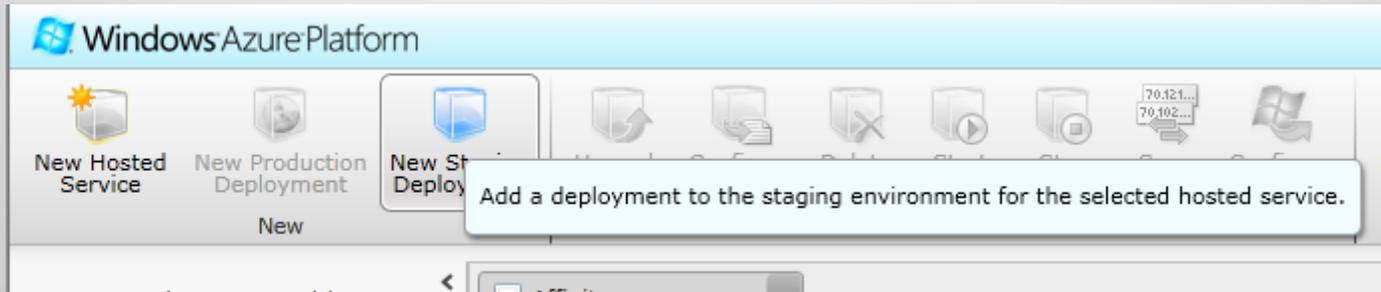


Deployment

Moving your app into the cloud



Production and Staging Environments

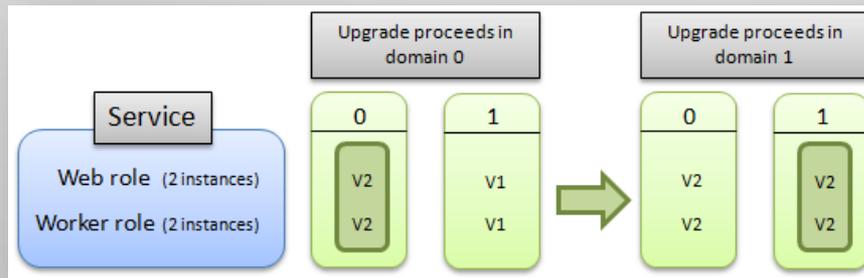


- Production environment
 - `http://<myapp>.cloudapp.net`
- **Staging** environment
 - `http://<guid>.cloudapp.net`
 - Used for testing and preparation of new production version



Deployment Types

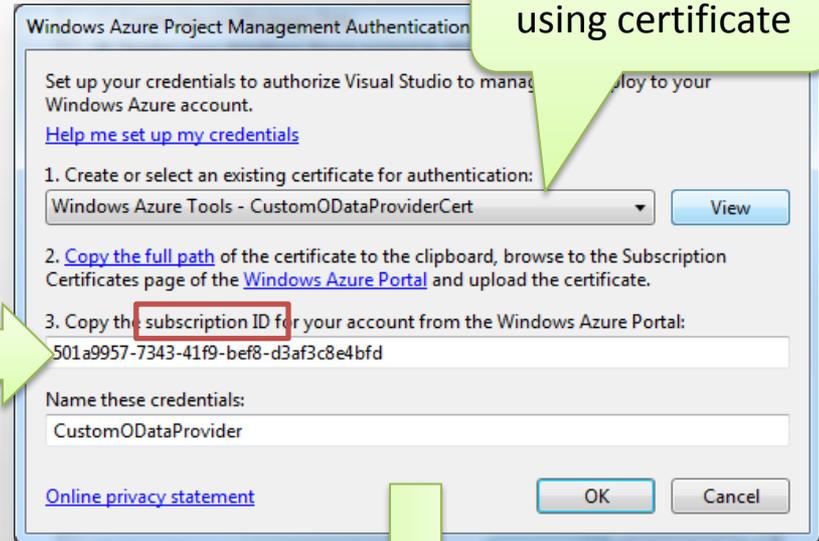
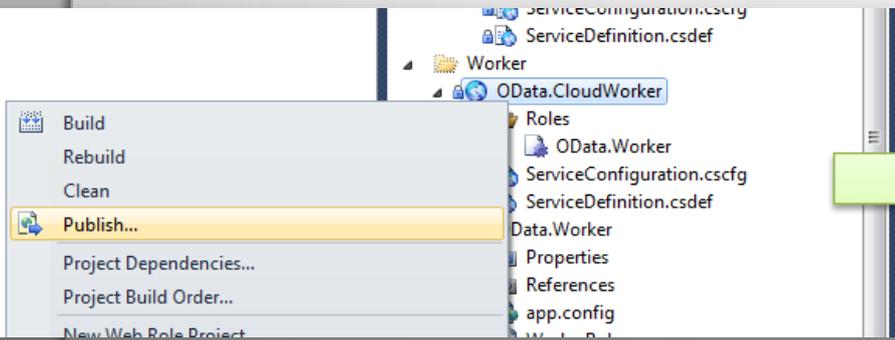
- In-Place update
 - Can be performed on prod and staging
 - Service model must be identical (e.g. same number of roles)



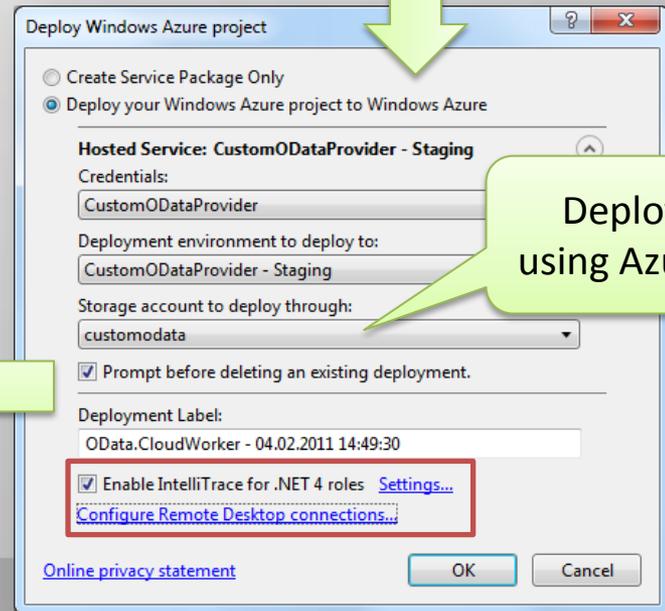
- **VIP Swap** (Virtual IP Swap)
 - Switches Prod ↔ Staging
 - Service model may have changed; endpoints must have stayed the same



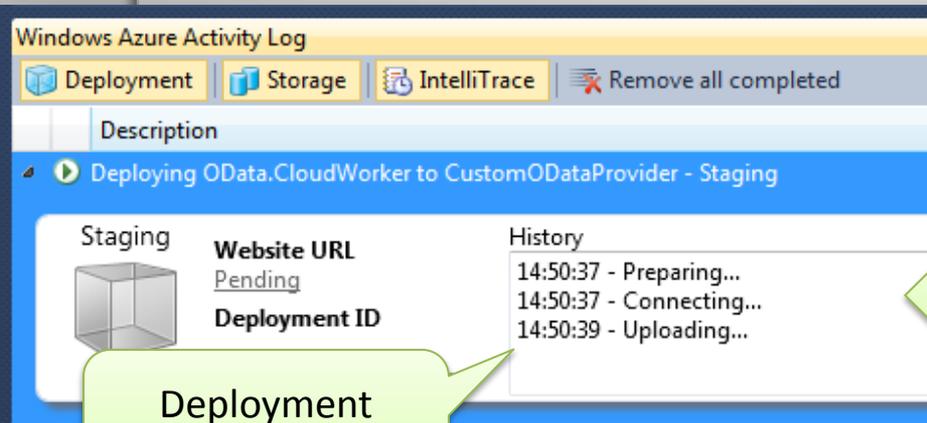
Deployment in VS



Authentication using certificate



Deployment using Azure Store



Deployment process in VS



Deployment in VS

The screenshot shows the Windows Azure Platform management console. The 'Upgrade' button in the top toolbar is highlighted with a red box. An 'Upgrade Deployment' dialog box is open, displaying the following configuration:

Upgrade Deployment

Subscription: Special: a-alejim

Service name: CustomODataProvider

Target environment: Production

Role to upgrade: All

Upgrade mode: Automatic Manual

Package location: Browse Locally... Browse Storage...

Configuration file: Browse Locally... Browse Storage...

Deployment name: CustomODataService.Cloud - 02.02.2011 16:07:24

Buttons: OK, Cancel



VIP Swap



- Prod contains V_x
- Deploy V_{next} to Staging
 - Connect staging to staging data stores
 - Do final QS
 - Connect staging to prod data stores
 - Do final QS and warmup
- Perform VIP Swap, now V_{next} is online
- Stop **and delete** staging



Troubleshooting

Hunting errors in the cloud using
RDP, Diagnostics and IntelliTrace



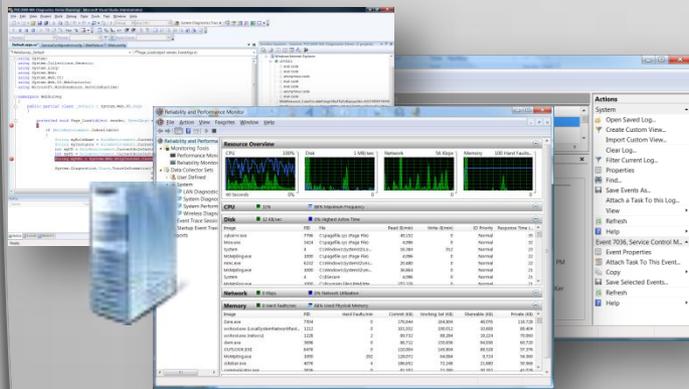
RDP vs. Diagnostics

On-Premise

- Static environment
- Well-known environment
- Single server

Cloud

- Dynamic environment
- Multi-instances, elastic
- Many nodes

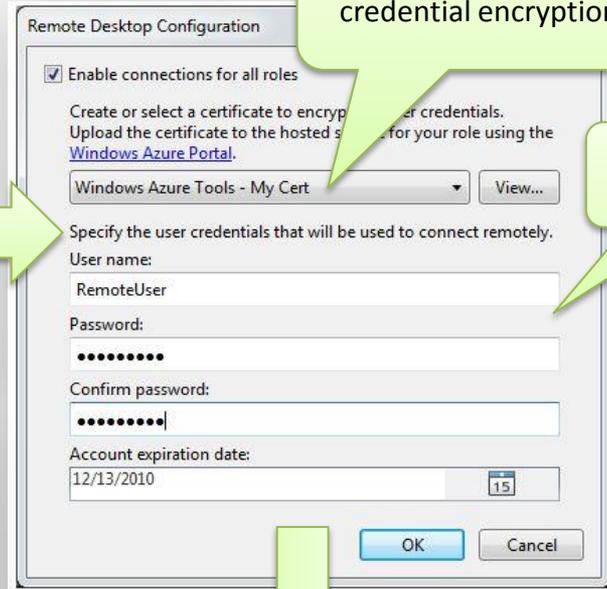
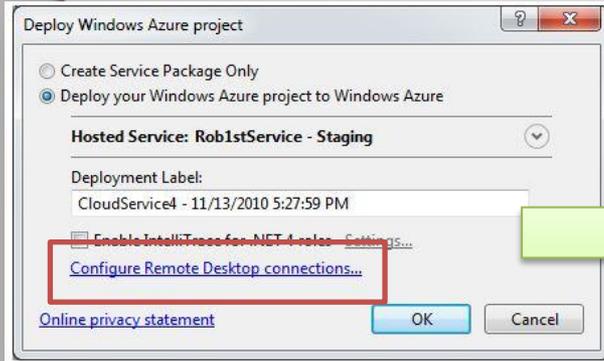


RDP vs. Diagnostics

- Remote access via RDP
 - During development
 - Troubleshooting for specific instance (e.g. memory or CPU consumption)
- Diagnostics
 - Permanent
 - Long-term statistics
 - Monitor health of complete system



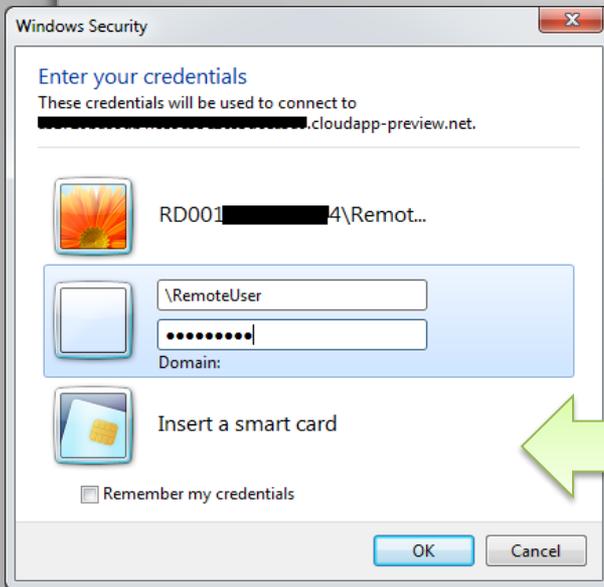
Remote Connection (RDP)



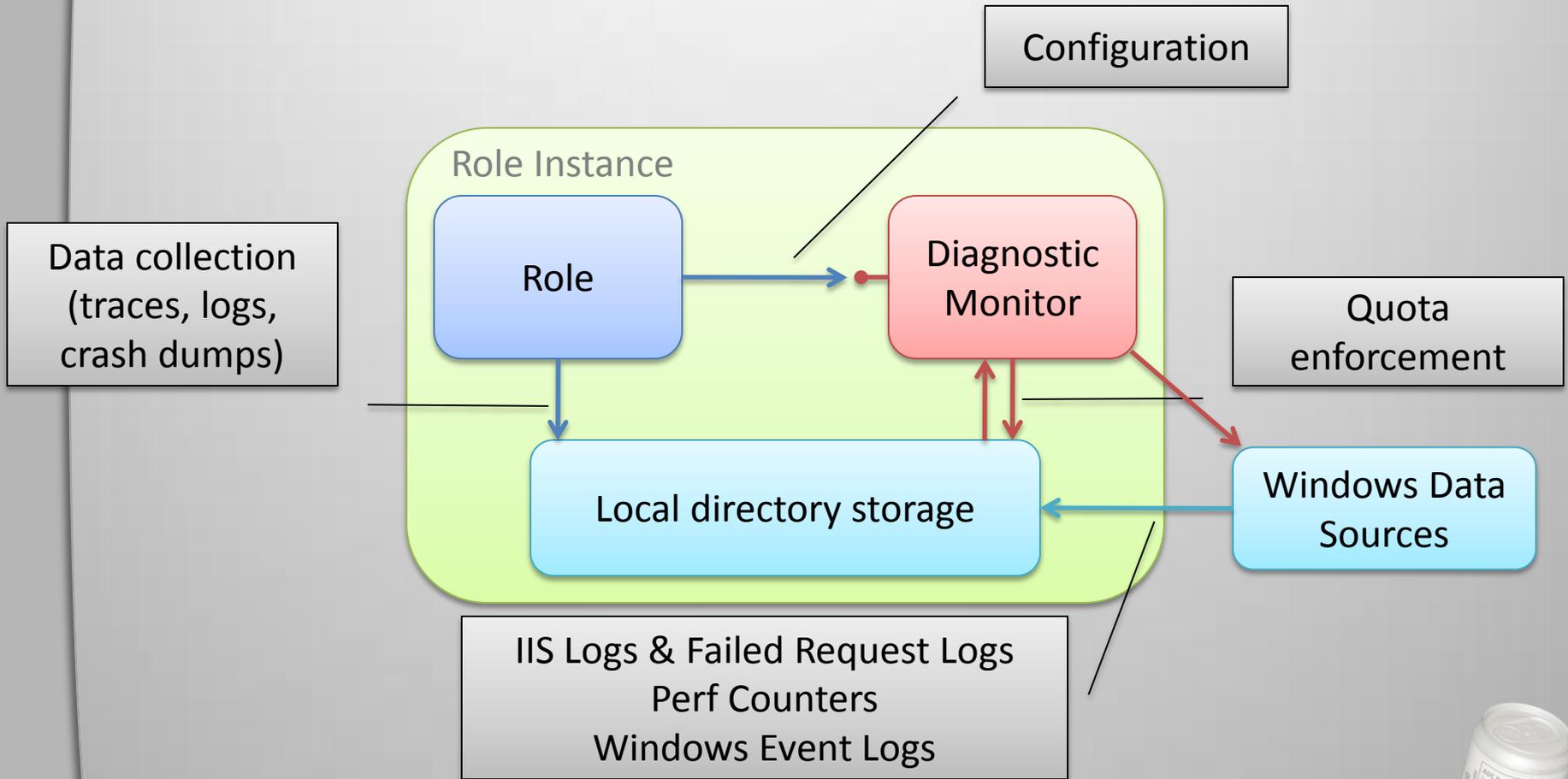
Certificate for credential encryption

Credentials

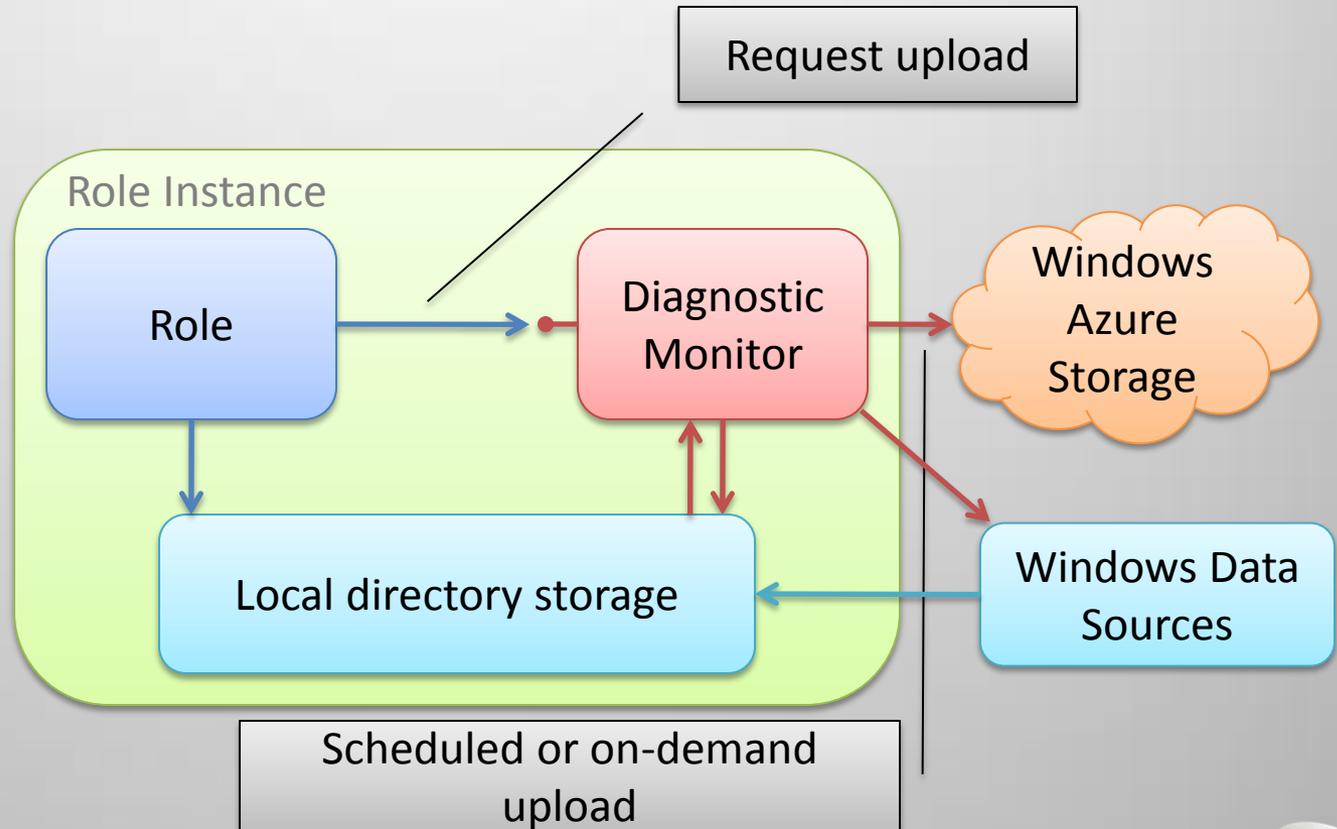
Don't forget to enable RDP!



Azure Diagnostics



Azure Diagnostics



```
public static void ApplyDefaultAzureDiagnosticConfiguration()
{
    // Get default initial configuration.
    var config = DiagnosticMonitor.GetDefaultInitialConfiguration();

    // Adding performance counters to the default diagnostic configuration
    ConfigureDiagnostics(config);
    ScheduleTransfer(config, TimeSpan.FromMinutes(1));

    // Start the diagnostic monitor with the modified configuration.
    DiagnosticMonitor.Start("Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString", config);
}
```

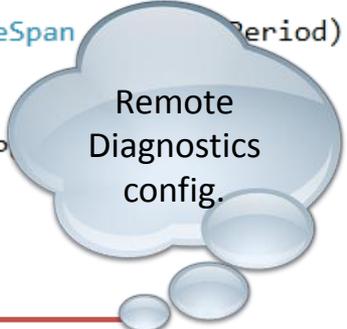
```
public static void ConfigureDiagnostics(DiagnosticMonitorConfiguration config)
{
    config.PerformanceCounters.DataSources.Add(
        new PerformanceCounterConfiguration()
        {
            CounterSpecifier = @"\Processor(_Total)\% Processor Time",
            SampleRate = TimeSpan.FromSeconds(5)
        });
}
```

```
public static void ScheduleTransfer(DiagnosticMonitorConfiguration config, TimeSpan transferPeriod)
{
    config.PerformanceCounters.ScheduledTransferPeriod =
        config.DiagnosticInfrastructureLogs.ScheduledTransferPeriod = transferPeriod;
}
```

```
public static DeploymentDiagnosticManager GetRemoteDiagnosticsManager()
{
    return CloudAccountDiagnosticMonitorExtensions.CreateDeploymentDiagnosticManager(
        CloudStorageAccount.FromConfigurationSetting("StorageConnectionString"),
        ConfigurationManager.AppSettings["DeploymentID"]);
}
```



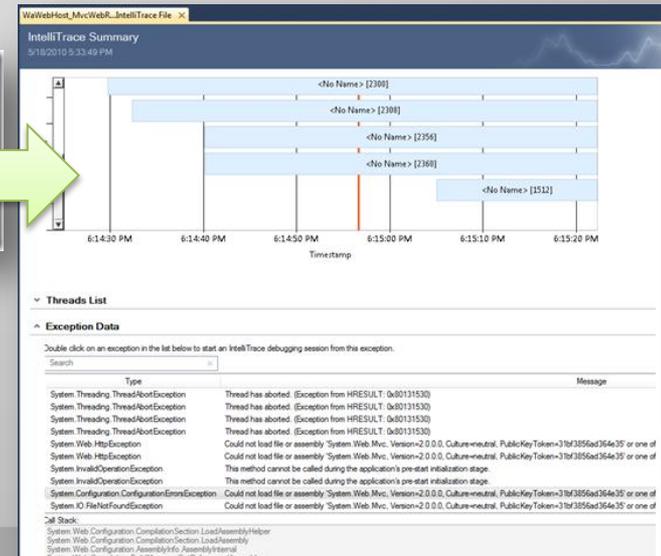
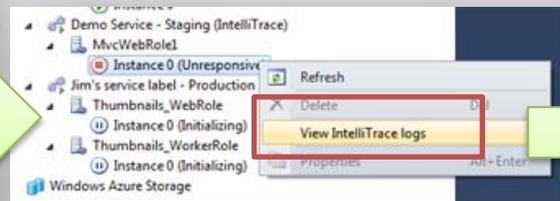
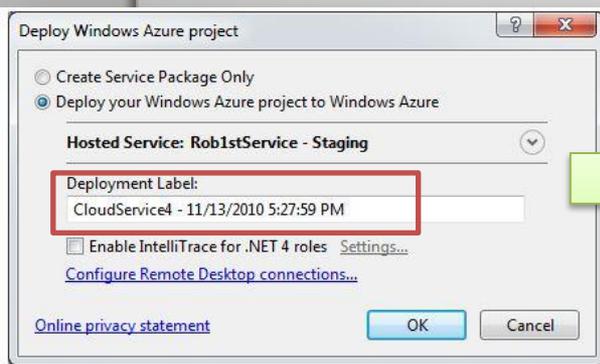
Diagnostics aufsetzen



Remote
Diagnostics
config.

IntelliTrace in Azure

- Collect data about events that happened in Azure
- Open data in VS and see e.g. exceptions, call flow, etc.
- IntelliTrace data is collected in Windows Azure Storage



Build Automation with Azure

- Automate deployment and run unit tests in the cloud



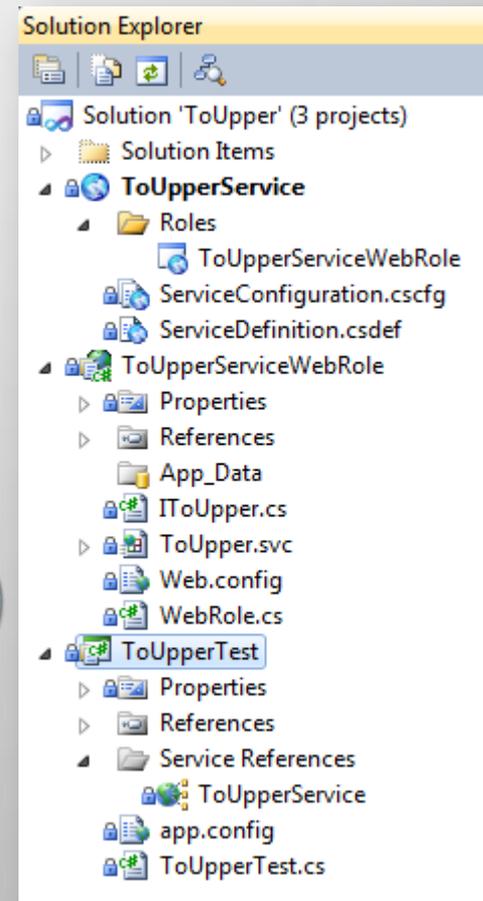
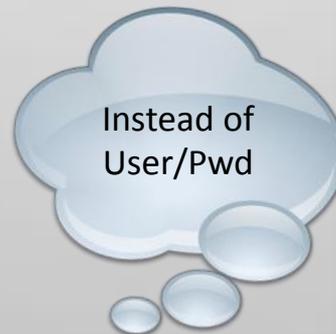
Build Automation with Azure

- Goals
 - Create a build process template for customization
 - Build cspkg in Build Process
 - Deploy to Azure using Azure Powershell Cmdlets
 - Run Unit test against newly deployed service
 - Remove Hosted Service
 - Unless you have too much \$



Pre-requisites

- Working Azure Solution
 - WebRole with Service (ToUpper)
 - Test Assembly
 - Service Reference to Service
- Build Server with
 - Azure Toolkit (tested with 1.3)
 - Powershell
 - Powershell [Azure Cmdlets](#)
- Azure Account
 - Hosted Service for testing
 - Certificate for Buildserver
 - Certificate from Developer Machine



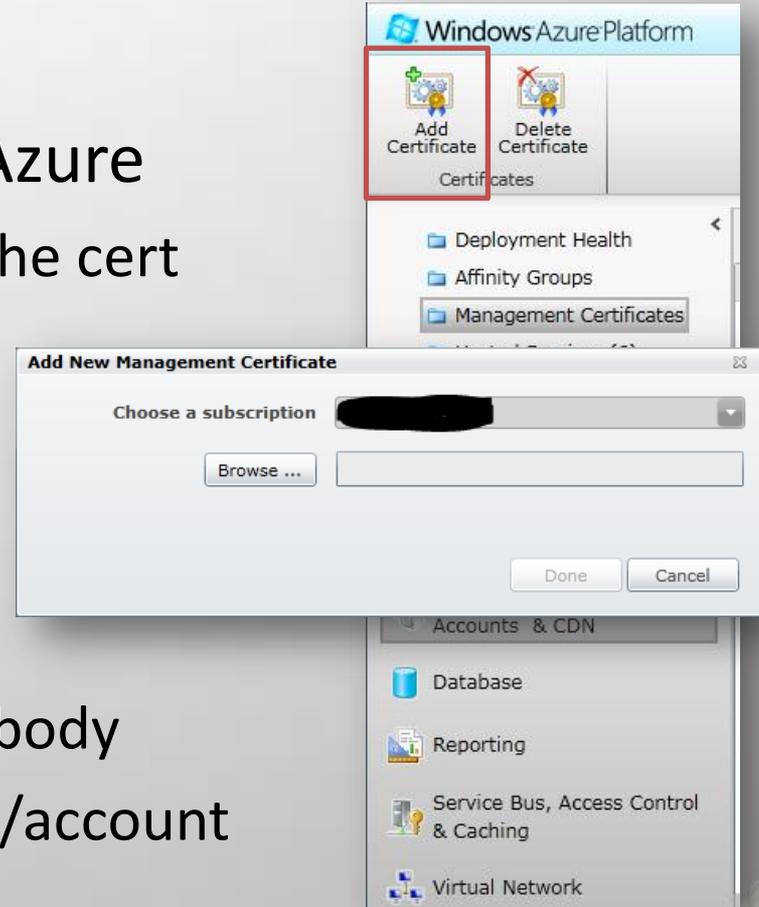
Setting up certificates

- Why certificates?
 - Allows you to store credentials in a secure place
 - No passwords in source code
 - Only way for Azure REST API
- Idea:
 - Every machine allowed to access management
 - Creates and installs a new certificate in windows secure store
 - Upload public key to azure management portal („Management Keys“)
- Login on machine with build account
 - The account that runs the build agent
 - Open visual studio shell
 - Execute a command
 - `makecert -r -pe -a sha1 -n "CN=Build Machine Certificate" -ss My -len 2048 -sp "Microsoft Enhanced RSA and AES Cryptographic Provider" -sy 24 buildmachine.cer`
 - This creates a public key, buildmachine.cer
 - Upload buildmachine.cer to management portal



Setting up certificates

- Upload the certificate to Azure
 - Browse to the folder with the cert
- Sidenotes:
 - A cer file is not critical
 - It is only a public key
 - You don't want to loose it
 - You can distribute it to anybody
 - Authenticates the machine/account



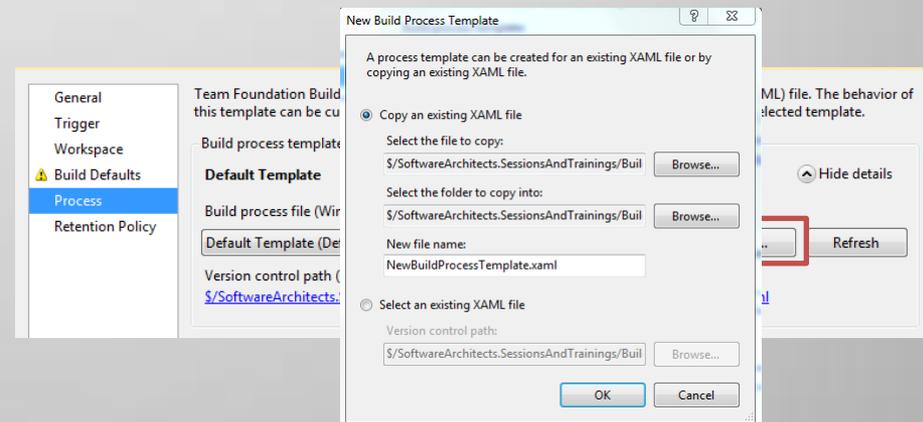
Goal: Create a custom build template

- Create a new build definition
 - Team Explorer -> Builds -> New Build Definition
 - Choose a build Controller

– Process

- Show details
- Choose „New“
- Select a new name

- New process templated created in
 - „\$TeamProject\BuildProcessTemplates\



Goal: Create a custom build template

- Create a new blank Solution
- Add created Process Template XAML to it.
 - Better to edit Process Template within a solution
 - Especially true if creating custom activities
- We have our build definition for now
 - Will customize it later to do azure deployment



Build Automation with Azure

- Goals
 - Create a build process template for customization ✓
 - Build cspkg in Build Process
 - Deploy to Azure using Azure Powershell Cmdlets
 - Run Unit test against newly deployed service
 - Remove Hosted Service
 - Unless you have too much \$



Goal: Building cspkg in Build Process

- `Microsoft.CloudService.targets` provides a „Publish“ target
 - used by default for cloud projects
 - call the target additionally to the normal build
- Creates the cspkg in the `Publish` folder
 - Is automatically copied to Drop location



Goal: Building cspkg in Build

General
Trigger
Workspace
Build Defaults
Process
Retention Policy

Team Foundation Build uses a build process template defined by a Windows Workflow (XAML) file. The behavior of this template can be customized by setting the build process parameters provided by the selected template.

Build process template: **BuildDeployTestAzure.xaml** [Show details](#)

Build process parameters:

1. Required	
Items to Build	Build 1 project(s) for 1 platform(s) and configuration(s)
2. Basic	
Automated Tests	Run tests in assemblies matching <code>***test*.dll</code>
Build Number Format	<code>\$(BuildDefinitionName)_\$(Date:yyyyMMdd)\$(Rev.:r)</code>
Clean Workspace	All
Logging Verbosity	Diagnostic
Perform Code Analysis	AsConfigured
Source And Symbol Server Settings	
3. Advanced	
Agent Settings	Use agent where Name=* and Tags is empty; Max Wait Tim
Analyze Test Impact	True
Associate Changesets and Work Items	True
Copy Outputs to Drop Folder	True
Create Work Item on Failure	False
Disable Tests	False
Get Version	
Label Sources	True
MSBuild Arguments	/t:Build;Publish
MSBuild Platform	Auto
Private Drop Location	
4. Misc	
AzureCertificateThumbprint	993C21CE392234A6EFCD3E6A344D64175A154776
AzureHostedServiceName	TFSAzureDeployTest
AzureStorageName	oop2011
AzureSubscriptionID	501a9957-7343-41f9-bef8-d3af3c8e4bfd

Windows Workflow (XAML) file. The behavior of parameters provided by the selected template.

[Show details](#)

Build 1 project(s) for 1 platform(s) and configuration(s)

Run tests in assemblies matching `***test*.dll`

`$(BuildDefinitionName)_$(Date:yyyyMMdd)$(Rev.:r)`

All

Diagnostic

AsConfigured

Use agent where Name=* and Tags is empty; Max Wait Tim

True

True

True

False

False

True

Auto

993C21CE392234A6EFCD3E6A344D64175A154776

TFSAzureDeployTest

oop2011

501a9957-7343-41f9-bef8-d3af3c8e4bfd

- Ec
- G
- Ec
- Sa

Build Automation with Azure

- Goals
 - Create a build process template for customization ✓
 - Build cspkg in Build Process ✓
 - Deploy to Azure using Azure Powershell Cmdlets
 - Run Unit test against newly deployed service
 - Remove Hosted Service
 - Unless you have too much \$



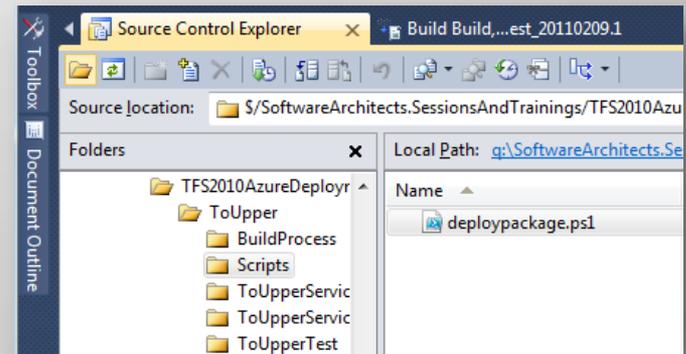
Goal: Deploying to Azure

- Powershell Cmdlets
 - Provide scriptable access to Azure Management
 - A wrapper for the Windows Azure REST API
 - No magic calls
 - You could use whatever to call the REST Service
 - Cmdlets provide some helpers
 - Uploading to blob store and creating a deployment is a single call
 - Free to use, on MS Code Gallery



Goal: Deploying to Azure

- We created a PS script that
 - Creates a new deployment in staging
 - Sets deployment to running
 - Swaps with Production
 - Waits till role is „Ready“
- Waiting till „Ready“ is crucial
 - Follow-up unit tests would fail
- That script is checked in
 - Checked-out during the build process and therefore executable



Goal: Deploying to Azure

```
# certificatethumb subscriptionId servicename package config
$certTP = $args[0]
$cert = Get-Item cert:\CurrentUser\My\$certTP
$sub = $args[1]
$storageAccount = $args[2]
$serviceName = $args[3]
$package = $args[4]
$config = $args[5]
$label = $args[6]
Add-PSSnapin AzureManagementToolsSnapIn

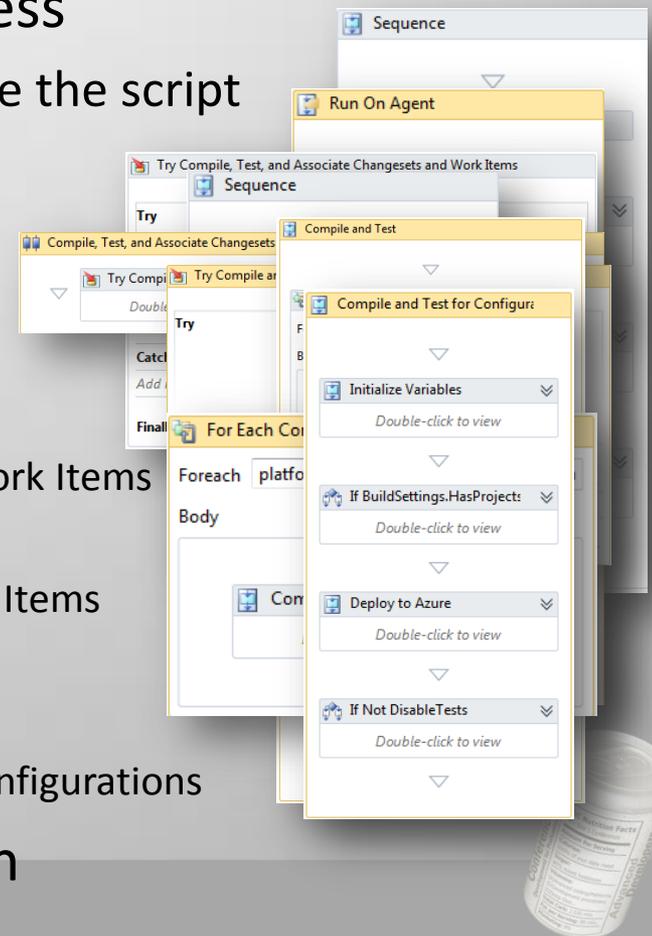
New-Deployment -serviceName $serviceName -storageserviceName $storageAccount -subscriptionId $sub -
certificate $cert -slot 'Staging' -package $package -configuration $config -label $label | Get-OperationStatus -
waitToComplete

Get-HostedService $serviceName -Certificate $cert -SubscriptionId $sub |Get-Deployment -Slot 'Staging' |Set-
DeploymentStatus 'Running' |Get-OperationStatus -waitToComplete
Get-Deployment staging -subscriptionId $sub -certificate $cert -serviceName $serviceName | Move-Deployment | Get-
OperationStatus -waitToComplete
Get-HostedService $serviceName -Certificate $cert -SubscriptionId $sub |Get-Deployment -Slot 'Staging' |Set-
DeploymentStatus 'Suspended' |Get-OperationStatus -waitToComplete
Get-HostedService $serviceName -Certificate $cert -SubscriptionId $sub |Get-Deployment -Slot 'Staging' |Remove-
Deployment | Get-OperationStatus -waitToComplete
Get-HostedService $serviceName -Certificate $cert -SubscriptionId $sub |Get-Deployment -Slot 'Production' |Set-
DeploymentStatus 'Running' |Get-OperationStatus -waitToComplete

$ready = $False
while(!$ready)
{
    $d = Get-HostedService $serviceName -Certificate $cert -SubscriptionId $sub |Get-Deployment -Slot 'Production'
    $ready = ($d.RoleInstanceList[0].InstanceStatus -eq "Ready") -and ($d.Label -eq $label)
}
}
```

Goal: Deploying to Azure

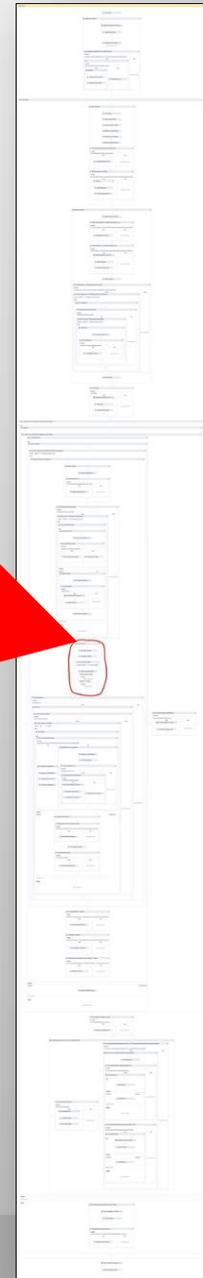
- Executing the script in the Build Process
 - Modify Build Process Template to execute the script
- Open the Build Template
- Navigate to
 - Sequence
 - Run On Agent
 - Try Compile, Test and Associate Changesets and Work Items
 - Sequence
 - Compile, Test, and Associate Changesets and Work Items
 - Try Compile and Test
 - Compile and Test
 - For Each Configuration in BuildSettings.PlatformConfigurations
 - Compile and Test for Configuration



Goal: Deploying to Azure

You are Here!

;-)



Goal: Deploying to Azure

- This “Deploy to Azure” Sequence
- Finds the cscfg and cspkg files in the Publish directory
 - Uses the FindMatchingFiles Activity provided by TFS

- Assign

- a

- Invoke

- Arguments

- S

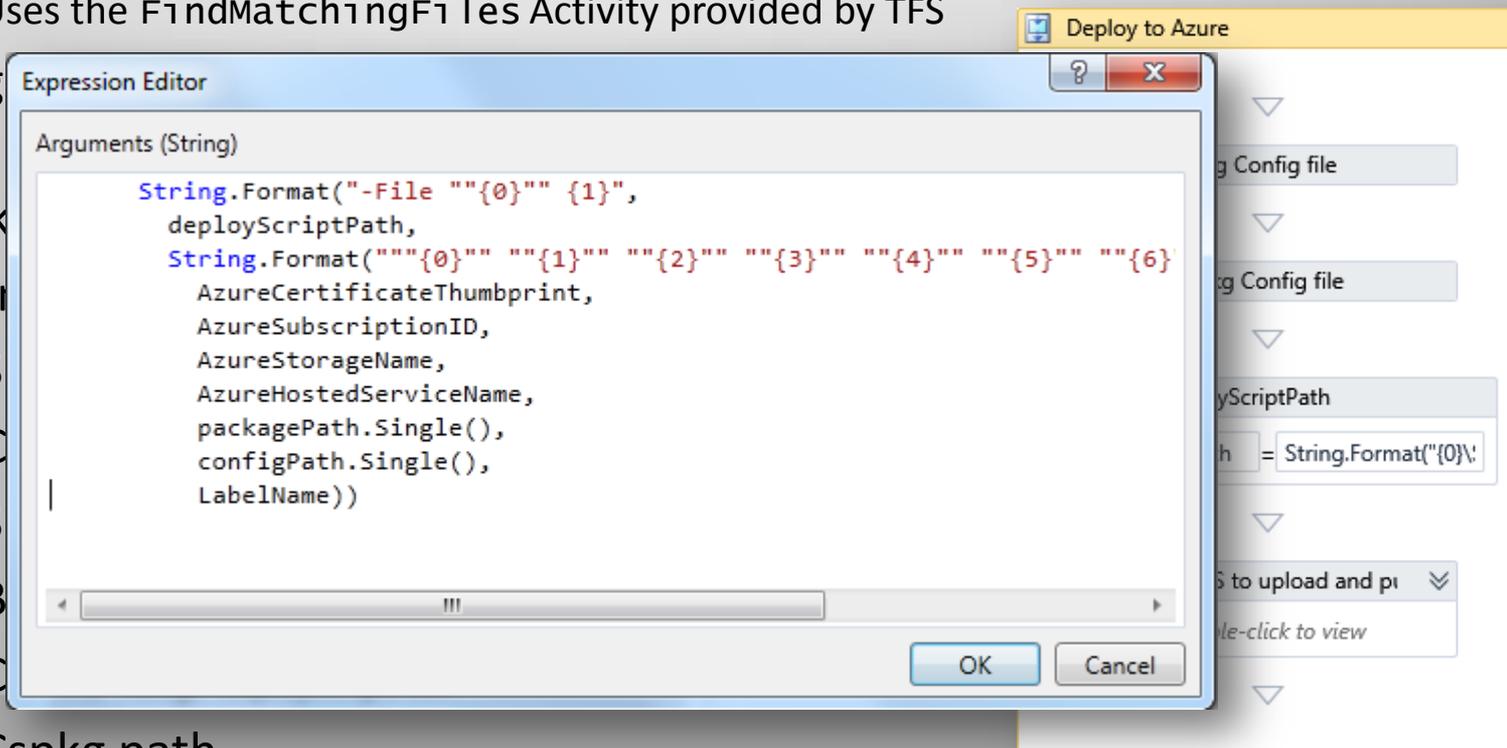
- C

- S

- B

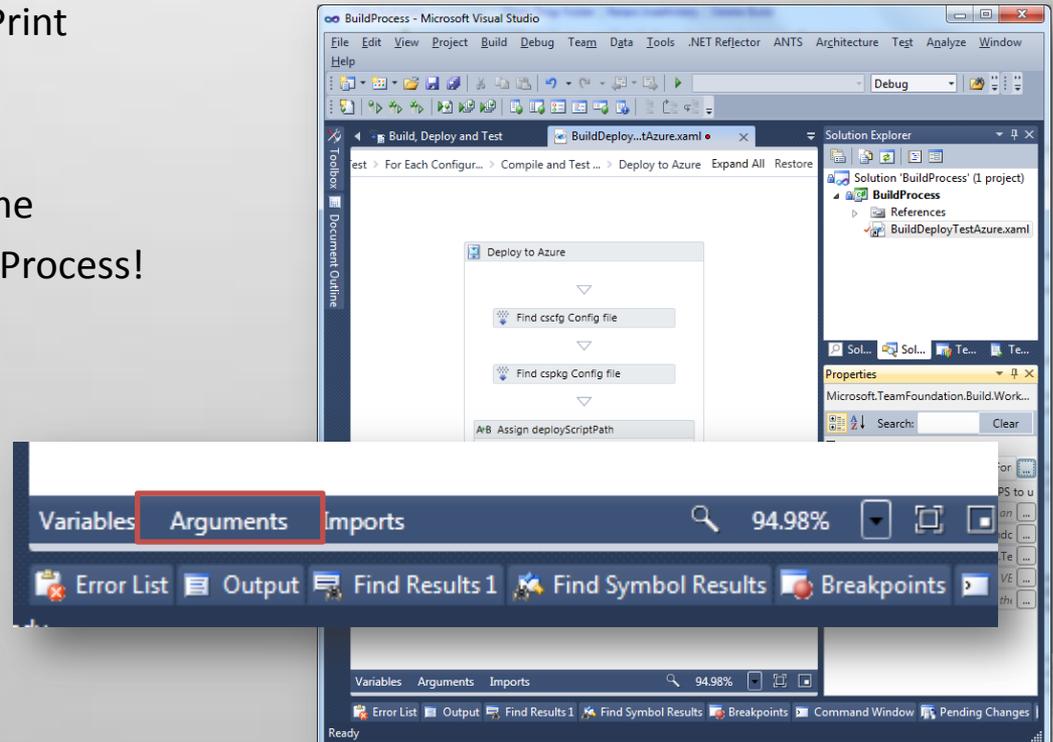
- C

- Cspkg path



Goal: Deploying to Azure

- Where does it the following information from?
 - AzureCertificateThumbPrint
 - AzureSubscriptionID
 - AzureStorageName
 - AzureHostedServiceName
- Arguments to the WorkFlow Process!



Goal: Deploying to Azure

- Create Arguments

Name	Direction	Argument type	Default value
PrivateDropLocation	In	String	<i>Enter a VB expression</i>
Verbosity	In	BuildVerbosity	Microsoft.TeamFoundation.Build.Workflow.BuildVerbosity.I
Metadata	Property	ProcessParameterMetadataCo	(Collection) ...
SupportedReasons	Property	BuildReason	All ▼
AzureCertificateThumbprint	In	String	<i>Enter a VB expression</i>
AzureSubscriptionID	In	String	<i>Enter a VB expression</i>
AzureHostedServiceName	In	String	<i>Enter a VB expression</i>
AzureStorageName	In	String	<i>Enter a VB expression</i>

Create Argument

Variables Arguments Imports 94.98%



Goal: Deploying to Azure

- Edit the Build Process and fill in the blanks
- Where From ?
 - Azure Portal!
 - AzureStorageName – SomeBlobStorage name (you might have to create one first)
 - AzureHostedServiceName – A newly created Hosted service (without any deployment)

The image shows a screenshot of the Visual Studio interface. The main window displays the 'Build Process' for a project named 'BuildDeployTestAzure.xaml'. The 'Build process parameters' section is expanded to show the 'Advanced' tab. Several parameters are highlighted with red boxes:

- AzureCertificateThumbprint**: [Redacted]
- AzureHostedServiceName**: [Redacted]
- AzureStorageName**: [Redacted]
- AzureSubscriptionID**: [Redacted]

The 'Properties' window on the right shows the following details:

- Name**: CN=Windows Azure Tools
- Thumbprint**: [Redacted]
- Valid from**: 2/2/2011 10:02:19 AM UTC
- Valid to**: 2/2/2012 10:02:19 AM UTC
- Issued by**: CN=Windows Azure Tools
- Subscription ID**: [Redacted]

Blue lines connect the redacted values in the 'Build process parameters' window to the corresponding values in the 'Properties' window.



Goal: Deploying to Azure

- Let's try it! – Queue a new build

The screenshot displays the TFS interface with a build definition named 'TFSAzureDeployTest' in a 'Created' state. A detailed view of the build definition is overlaid, showing the following information:

- Build definition:** TFSAzureDeployTest
- Type:** Hosted Service
- Status:** Created
- Invoke PS to upload and publish cspkg**
- Initial Property Values**
- Arguments:** -File "C:\BS\3\SoftwareArchitects.SessionsAndTrainings\Build, Deploy and Test\20\Sources\Scripts\deploypackage.ps1" "C:\BS\3\SoftwareArchitects.SessionsAndTrainings\Build, Deploy and Test\20\Binaries\Publish\ToUpperService.cspkg" "C:\BS\3\SoftwareArchitects.SessionsAndTrainings\Build, Deploy and Test\20\Binaries\Publish\ServiceConfiguration.cscfg" "Build, Deploy and Test_20110209.6"
- EnvironmentVariables:**
- FileName:** C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

The interface also shows a tree view of the build definition's components:

- TFSAzureDeployTest (Hosted Service) - Created
 - Certificates
 - Build, Deploy and Test_20110209.6 (Deployment) - Starting... (Production)
- TFSAzureDeployTest (Hosted Service) - Created
 - Certificates
 - Build, Deploy and Test_20110209.6 (Deployment) - Initializing... (Production)
- TFSAzureDeployTest (Hosted Service) - Created
 - Certificates
 - Build, Deploy and Test_20110209.6 (Deployment) - Ready (Production)
 - ToUpperServiceWebRole (Role) - Ready (Production)
 - ToUpperServiceWebRole_IN_0 (Instance) - Ready (Production)



Goal: Deploying to Azure

- Did the PS script wait till ready? – Yes
 - It took about 11 minutes

11:02

11:02

Invoke PS to upload and publish cspkg

Initial Property Values

```
Arguments = -File "C:\BS\3\SoftwareArchitects.SessionsAndTrainings\Build, Deploy and Test\20\Sources\Scripts\deploypackage.ps1" "
"504-0957-22-1215-01-18-0-11-5" " " "TFSAzureDeployTest" "C:\BS\3\SoftwareArchitects.SessionsAndTrainings\Build, Dep
\ToUpperService.cspkg" "C:\BS\3\SoftwareArchitects.SessionsAndTrainings\Build, Deploy and Test\20\Binaries\Publish\ServiceConfiguration
EnvironmentVariables =
FileName = C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
OutputEncoding = System.Text.SBCSCodePageEncoding
WorkingDirectory =
```



Build Automation with Azure

- Goals
 - Create a build process template for customization ✓
 - Build cspkg in Build Process ✓
 - Deploy to Azure using Azure Powershell Cmdlets ✓
 - Run Unit test against newly deployed service
 - Remove Hosted Service
 - Unless you have too much \$



Goal: Run Unit test against newly deployed service

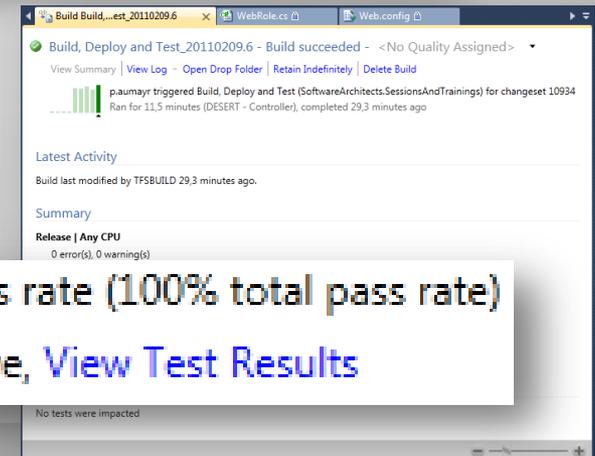
- Unit test currently runs against a local service
- Change endpoint in test configuration
 - Service name you chose(!)

```
<endpoint address="http://TFSAzureDeployTest.cloudapp.net/ToUpper.svc" binding="basicHttpBinding"
→ -bindingConfiguration="BasicHttpBinding_IToUpper" contract="ToUpperService.IToUpper"
→ name="BasicHttpBinding_IToUpper_Azure" -/>
```

- Start a new build
- Open build

▼ 1 test run(s) completed - 100% average pass rate (100% total pass rate)

1/1 test(s) passed, 0 failed, 0 inconclusive, [View Test Results](#)



The screenshot shows a TFS build system interface. At the top, it displays the build name "Build Build, ..._est_20110209.6" and the status "Build succeeded - <No Quality Assigned>". Below this, there are links for "View Summary", "View Log", "Open Drop Folder", "Retain Indefinitely", and "Delete Build". A progress bar indicates the build is complete. The "Latest Activity" section shows that the build was last modified by "TFSBUILD" 29.3 minutes ago. The "Summary" section shows "Release | Any CPU" with "0 error(s), 0 warning(s)". A test run summary is displayed, showing "1 test run(s) completed - 100% average pass rate (100% total pass rate)" and "1/1 test(s) passed, 0 failed, 0 inconclusive, View Test Results". At the bottom, it states "No tests were impacted".

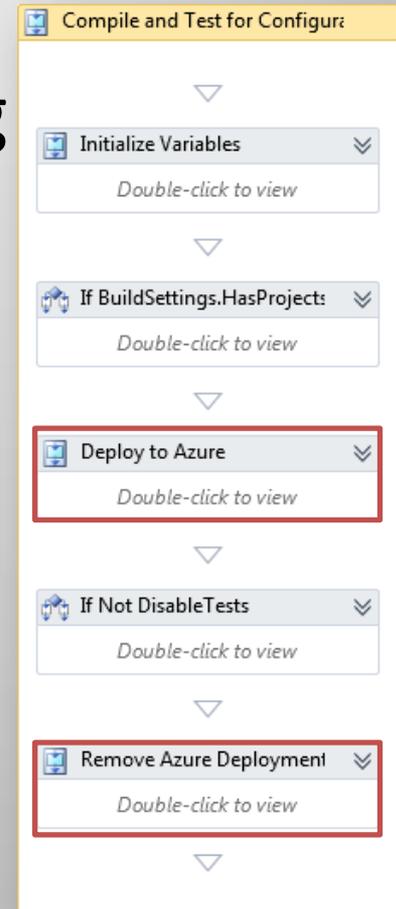
Build Automation with Azure

- Goals
 - Create a build process template for customization ✓
 - Build cspkg in Build Process ✓
 - Deploy to Azure using Azure Powershell Cmdlets ✓
 - Run Unit test against newly deployed service ✓
 - Remove Deployment
 - Unless you have too much \$



Goal: Remove Deployment

- After testing, service is still running
- We created a second PS script
 - Suspends productive deployment
 - Removes the deployment
 - Executed after unit tests



Goal: Remove Deployment

```
# certificatethumb subscriptionId servicename
$certTP = $args[0]
$cert = Get-Item cert:\CurrentUser\My\$certTP
$sub = $args[1]
$serviceName = $args[2]
Add-PSSnapin AzureManagementToolsSnapIn

Get-HostedService $serviceName -Certificate $cert -SubscriptionId $sub |
    Get-Deployment -Slot 'Production' |
    Set-DeploymentStatus 'Suspended' |
    Get-OperationStatus -waitToComplete

Remove-Deployment -Slot 'Production' -ServiceName $serviceName -
SubscriptionId $sub -Certificate $cert |
    Get-OperationStatus -waitToComplete
```

Build Automation with Azure

- Goals
 - Create a build process template for customization ✓
 - Build cspkg in Build Process ✓
 - Deploy to Azure using Azure Powershell Cmdlets ✓
 - Run Unit test against newly deployed service ✓
 - Remove Deployment ✓
 - Unless you have too much \$ (See next section)



Build Automation with Azure -- Summary

- Use Powershell Cmdlets for automation
 - Very handy, no custom Activities
- Can be used for more
 - Storage creation
- Make sure your azure role is „Ready“
 - Followup activities might depend on it
- Finally, cleaning up after one-self saves money



Costs

**Pay only those test resources that
you really need**



Windows Azure Pricing



COMPUTE

- ▶ Virtual Machine instances
- ▶ Load balancers, routers, etc.
- ▶ Relational DB instances
- ▶ Automated service management
 - Fabric controller operations
 - Load balancer programming

PRICE

- ▶ \$0.12 / hour per size unit



STORAGE

- ▶ Blob Storage
- ▶ Table Storage
- ▶ Multiple replicas

PRICE

- ▶ \$0.15 / GB stored / month
- ▶ Storage transactions: \$0.01 / 10k



BANDWIDTH

- ▶ Ingress/Egress (to/from internet only)

PRICE

- ▶ Bandwidth: \$0.10 IN; \$0.15 OUT; / GB

SQL Azure

- ▶ Easy to use
- ▶ Reliable
- ▶ Compatible with what you have

PRICE

- ▶ 1GB db : \$9.99/month
- ▶ 5 GB db: \$49.95/month *
- ▶ 10 GB db : \$99.99/month
- ▶ 50 GB db: \$499.95/month *
- ▶ Data transfers = \$0.10 in / \$0.15 out / GB

* Starting June 28, 2010



Windows Azure Pricing Advantages

- Get a production-like test environment for very little money
 - Compute and storage cluster
- Keep test environment online only as long as you need it
 - Tip: Think about keeping test data in the cloud
- VIP Swap to put new releases into production



**Advanced
Developers
Conference**
Development for Professionals!

Speziell zu Software-Testing

14.-15. Februar 2011, München

FRAGEN?



Wir sehen uns wieder!

**Advanced
Developers
Conference**
Development for Professionals!

High-Level-Konferenz speziell zu C++

05. – 06. Mai 2011, direkt am Chiemsee
cpp.ADC2011.de

 **ppedv Training**
★★★★★
einfach ausgezeichnet!

Trainings und Events der ppedv

Mehr als 100 verschiedene Trainings
auf Microsoft-Technologien spezialisiert
11 Standorte in D & AT
Maßgeschneiderte Trainings
direkt bei Ihnen vor Ort!

www.ppedv.de



**Advanced
Developers
Conference**
Development for Professionals!

Speziell zu Software-Testing

14.-15. Februar 2011, München

Hat Ihnen mein Vortrag gefallen?
Ich freue mich auf Ihr Feedback!



Vielen Dank!

Rainer Stropek

