Advanced Developers Conference
Development for Professionals!

Speziell zu Software-Testing

14.-15. Februar 2011, München

Testen in der Cloud

Altbekanntes oder neue Herausforderungen?

Veranstalter: ppedv

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
- Rainer Stropek
  - Developer, Speaker, Trainer
  - MVP for Windows Azure
  - rainer@timecockpit.com
  - twitter FOLLOW ME @rstropek

time cockpit

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

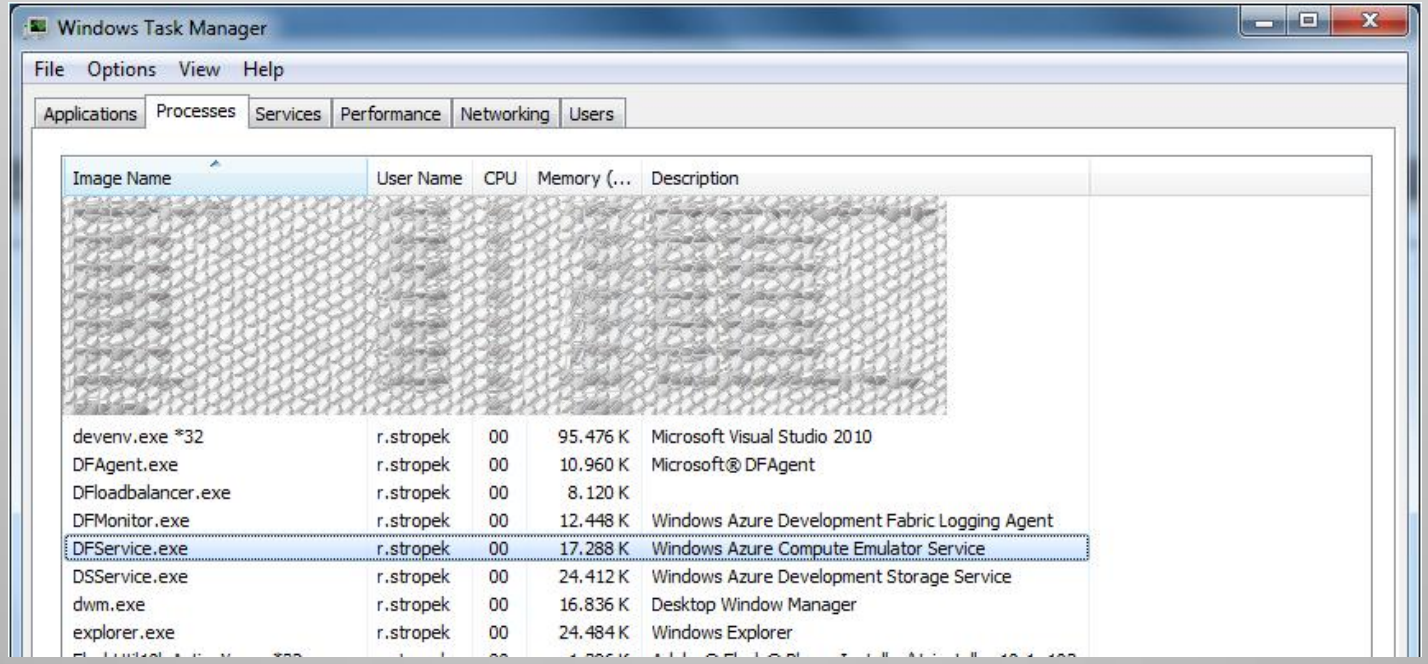
- *Windows Azure Compute Emulator* aka DevFabric
  - Part of Windows Azure SDK → free
- Simulates Windows Azure during development process
  - For debugging purposes
  - To lower costs
  - For offline scenarios
- DevFabric ≠ Windows Azure
  - Can access all locally installed resources
  - Might not be 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



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

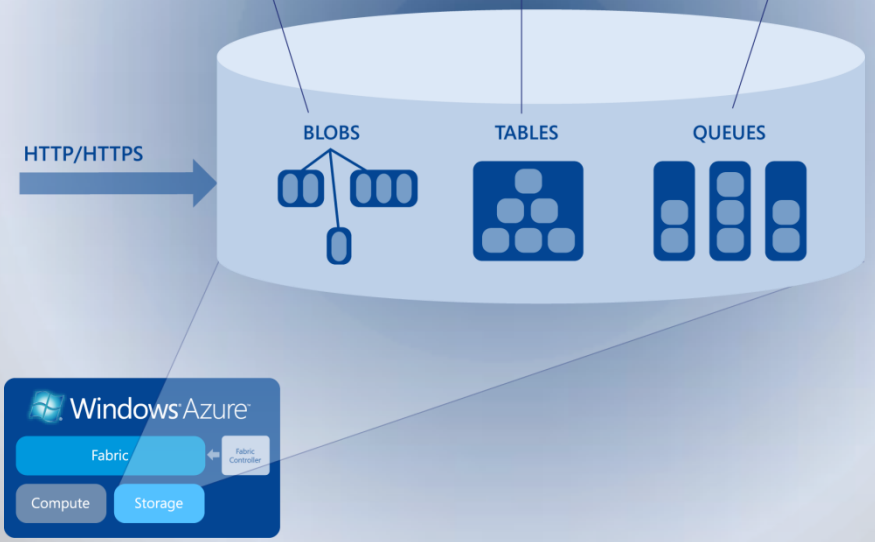
- Encapsulate logic that has to be unit tested into separate class libraries → testing as usual
- Include DevFabric in integration tests using CSPack/CSRun
- Build applications and services that can be run inside and outside of Azure

Storage in Azure

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

Tables: simply structured data, accessed using WCF Data Services

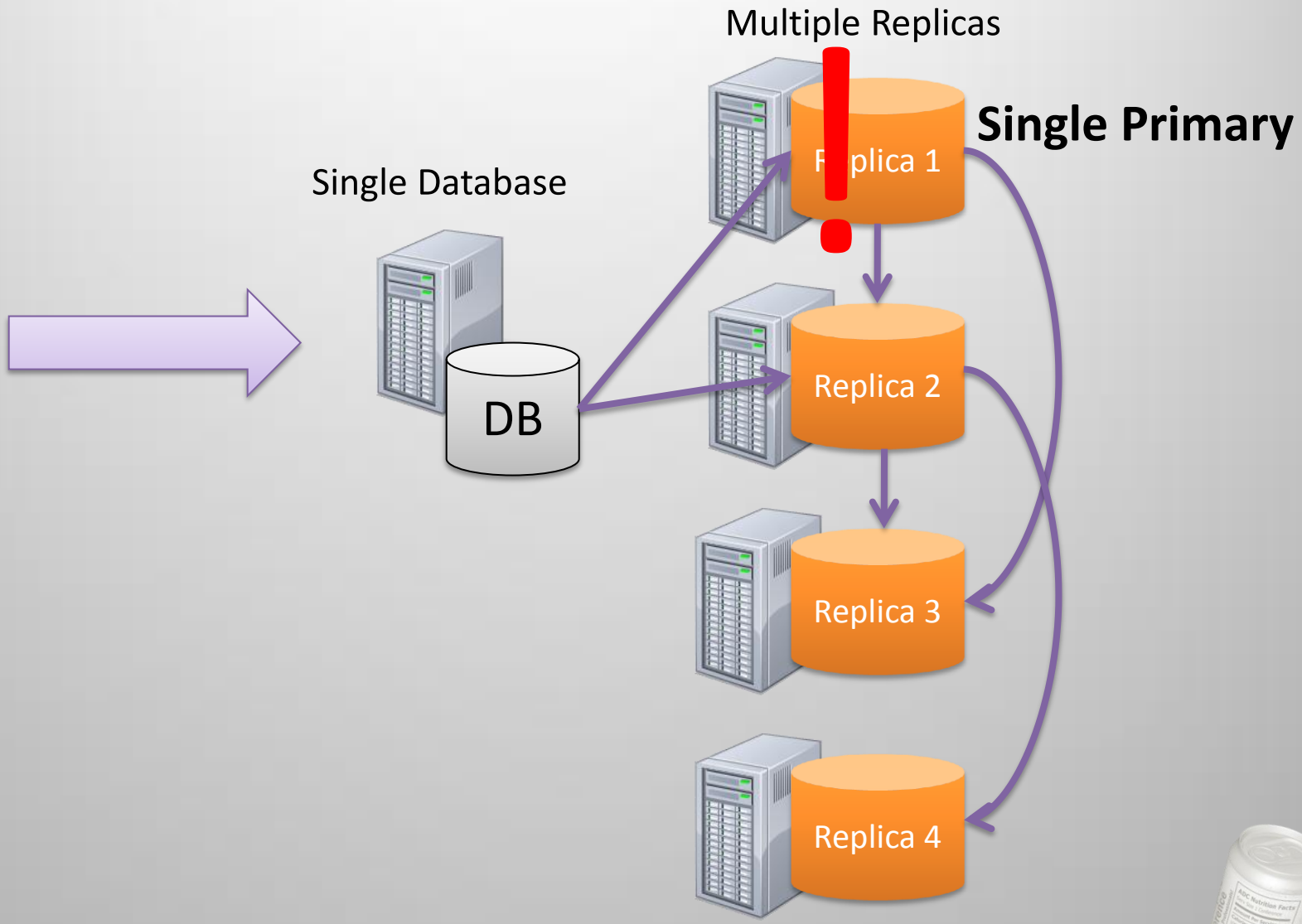
Queues: serially accessed messages or requests, allowing web-roles and worker-roles to interact



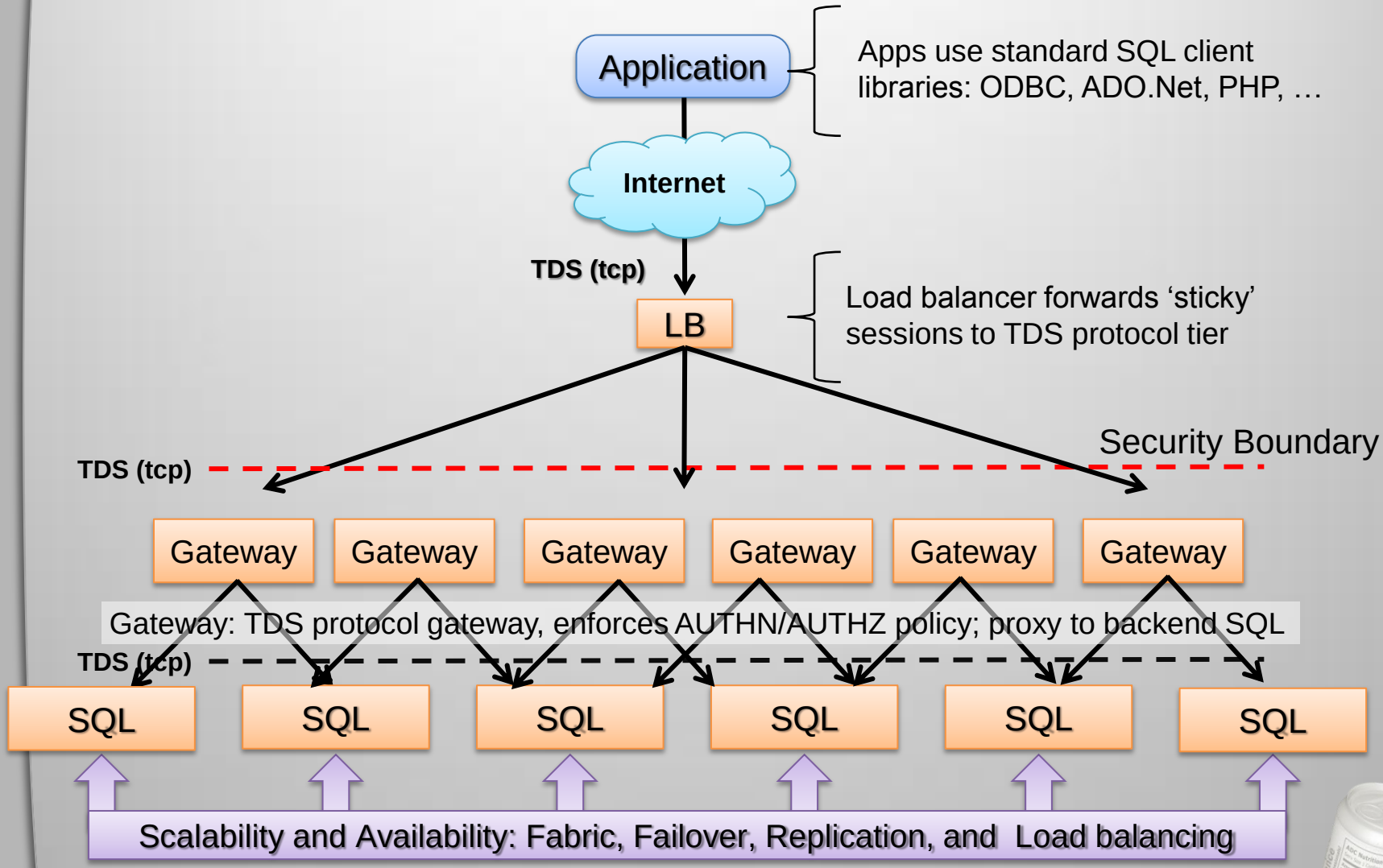
Windows Azure Storage

- *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


Multiple Replicas
Single Primary
Single Database
DB
Replica 1
Replica 2
Replica 3
Replica 4

SQL Azure


Application
Apps use standard SQL client libraries: ODBC, ADO.Net, PHP, …
Internet
TDS (tcp)
LB
Load balancer forwards 'sticky' sessions to TDS protocol tier
Security Boundary
TDS (tcp)
Gateway
Gateway
Gateway
Gateway
Gateway
Gateway
Gateway: TDS protocol gateway, enforces AUTHN/AUTHZ policy; proxy to backend SQL
TDS (tcp)
SQL
SQL
SQL
SQL
SQL
SQL
Scalability and Availability: Fabric, Failover, Replication, and Load balancing

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)

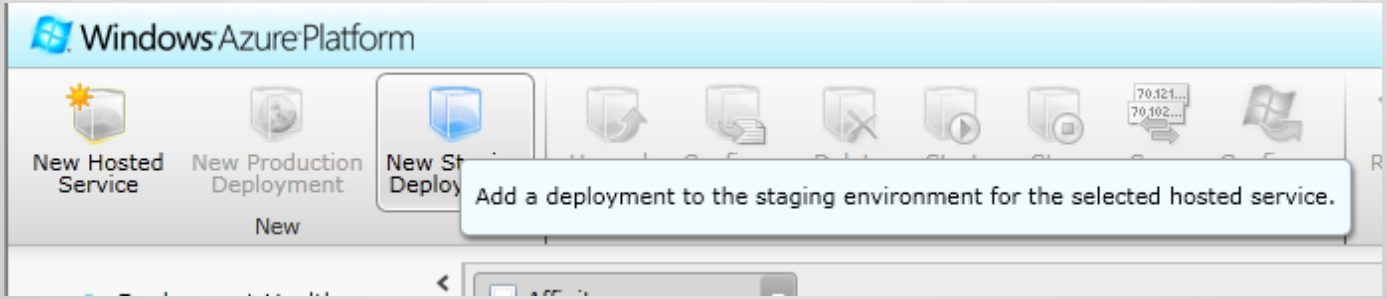
DevStorage

- 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

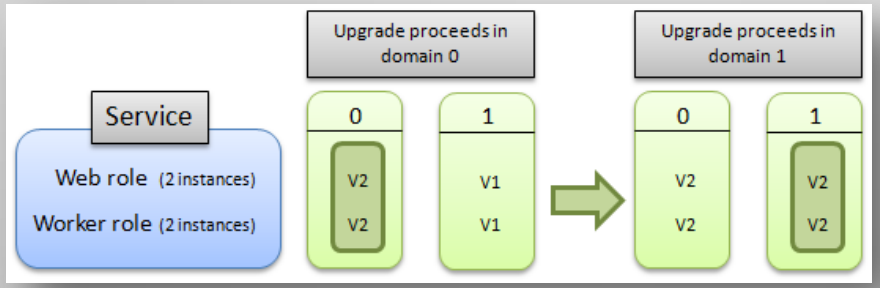
Production and Staging Environments



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

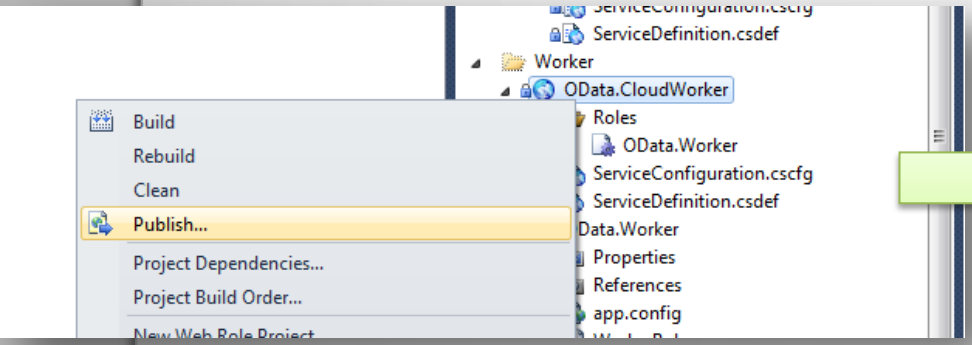
Deployment Types

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

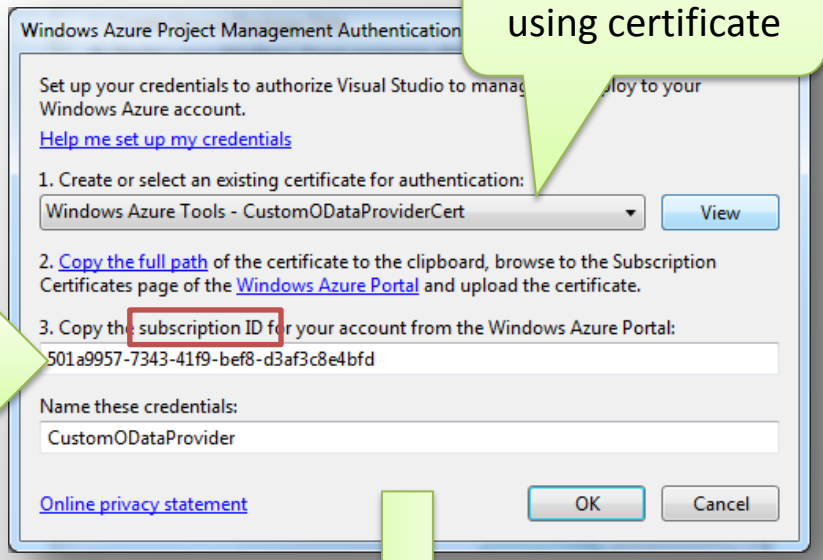


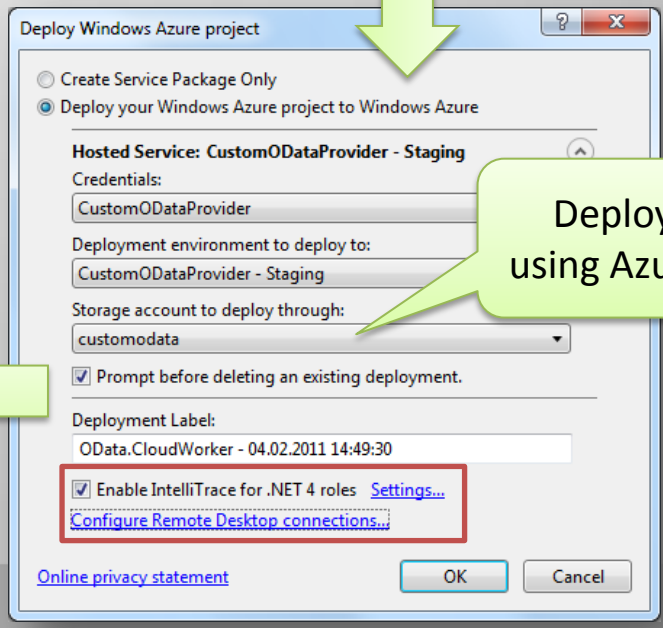
- **VIP Swap** (Virtual IP Swap)
  - Switches Prod ↔ Staging
  - Service model may have changed; endpoints must have stayed the same

Deployment in VS

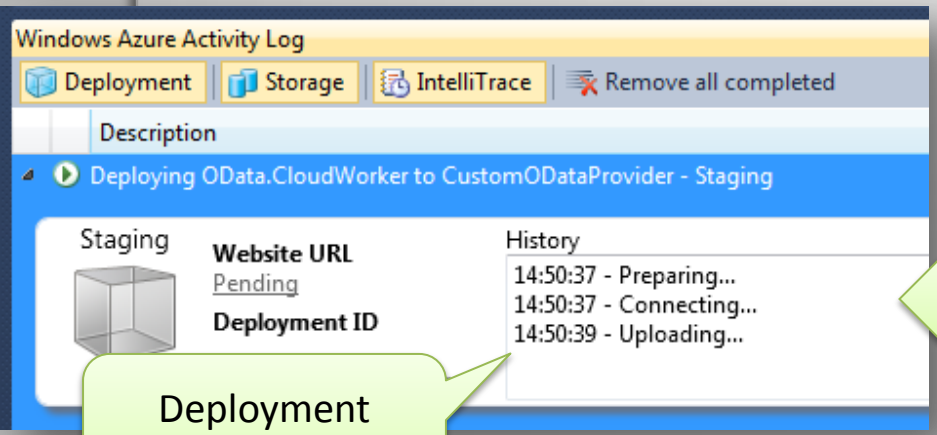


Authentication using certificate



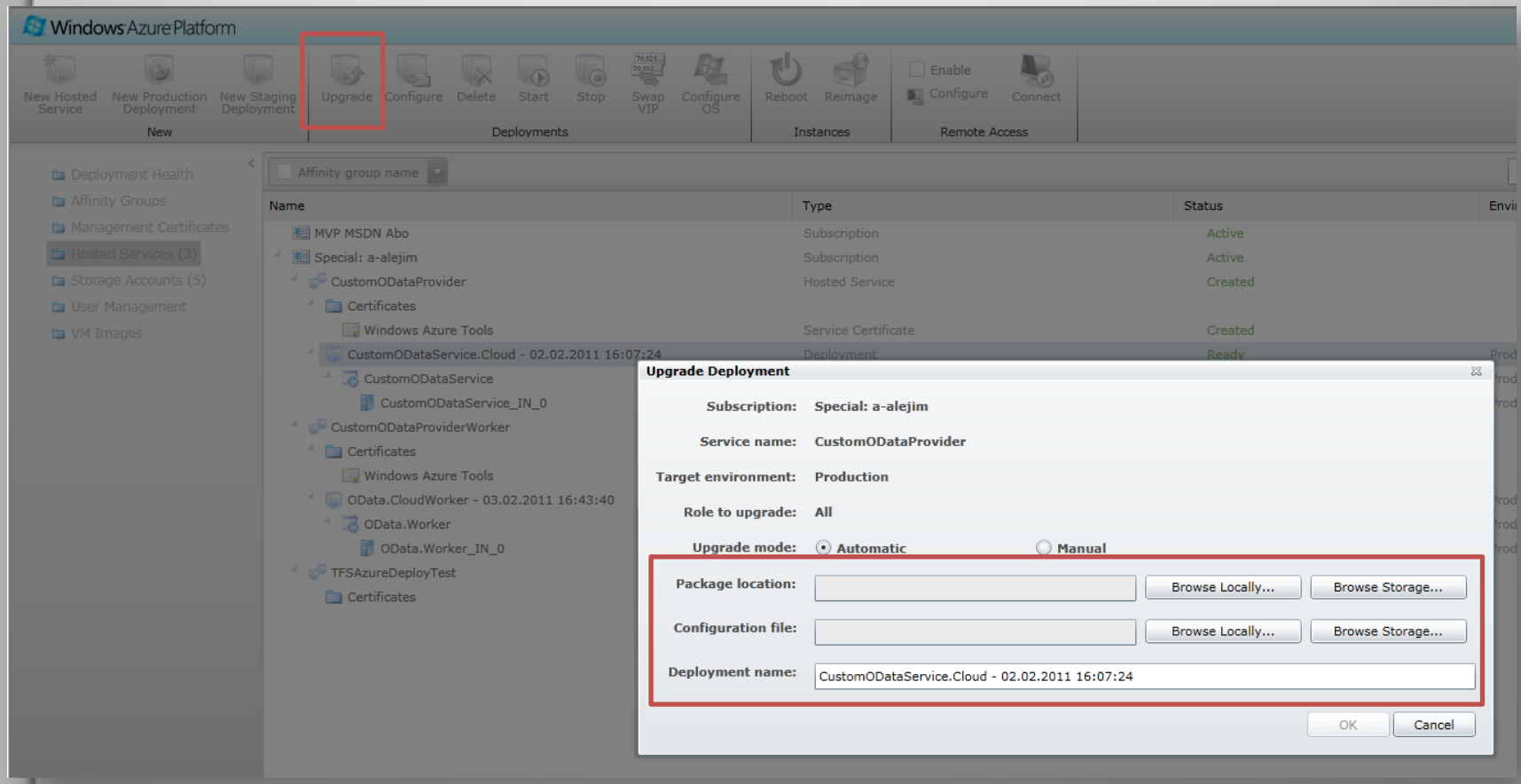


Deployment using Azure Store



Deployment process in VS

Deployment in VS



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

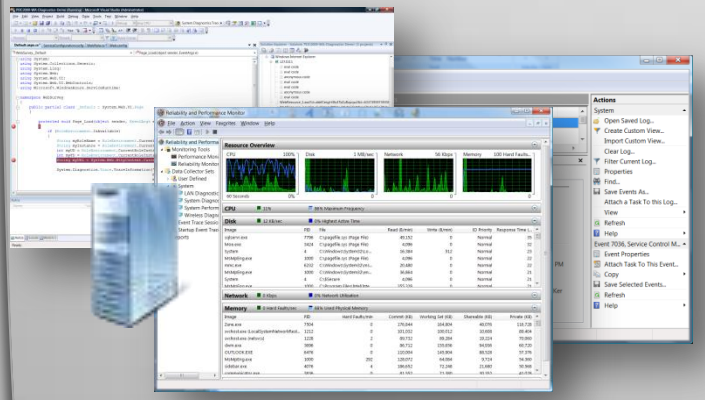
RDP vs. Diagnostics

On-Premise

- Static environment
- Well-known environment
- Single server

Cloud

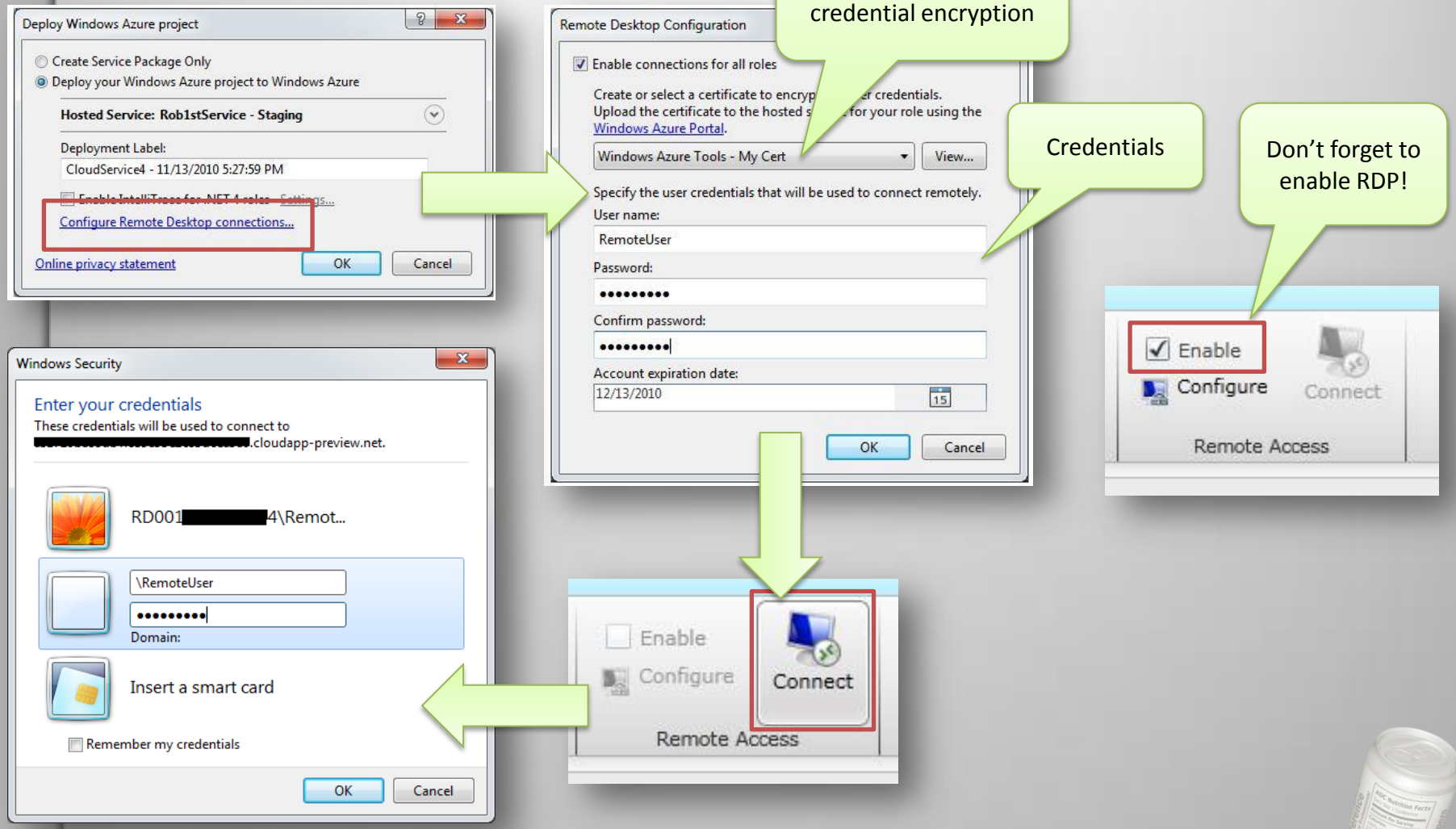
- Dynamic environment
- Multi-instances, elastic
- Many nodes



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)



Azure Diagnostics

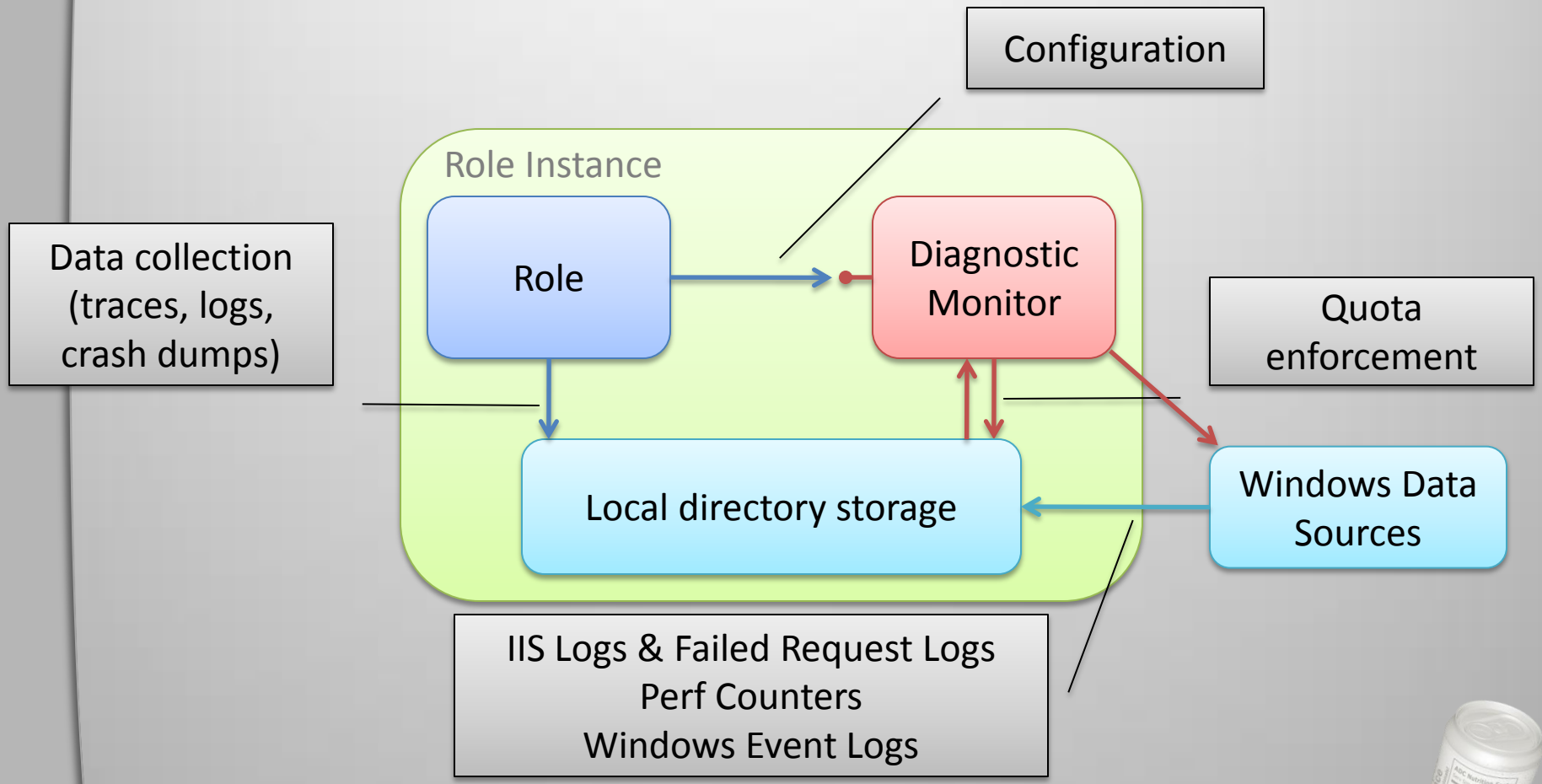


Image Source: Microsoft PDC 09, Session SVC15, Matthew Kerner

Azure Diagnostics

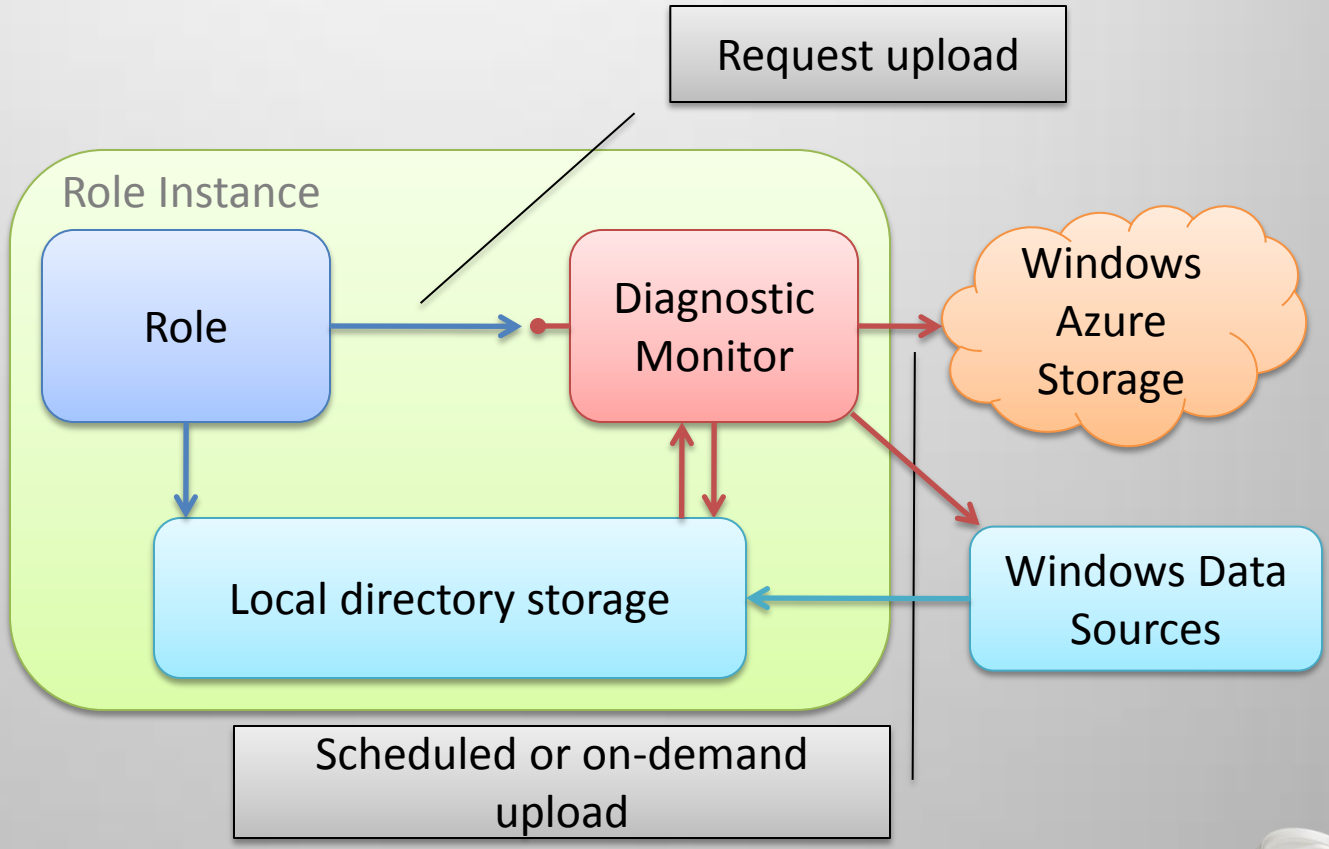


Image Source: Microsoft PDC 09, Session SVC15, Matthew Kerner

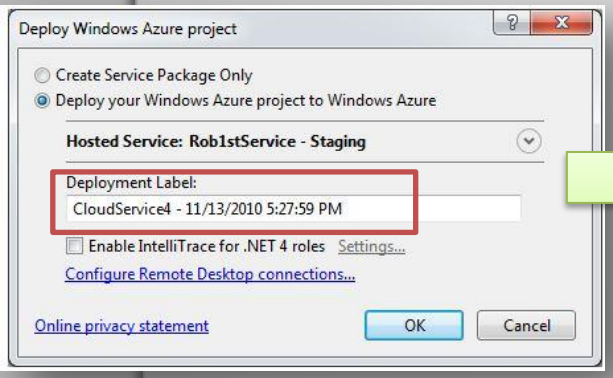
```
public static void ApplyDefaultAzureDiagnosticConfiguration()
{
    // Get default initial configuration.
    var config = DiagnosticMonitor.GetDefaultInitialConfiguration();

    // 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
                                                                          eriod)
{
    config.PerformanceCounters.ScheduledTransferPeriod =
        config.DiagnosticInfrastructureLogs.ScheduledTransferPeriod = scheduleP
}

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

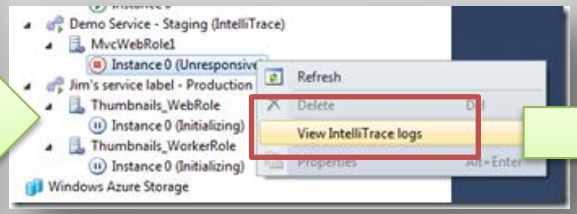
Diagnostics aufsetzen

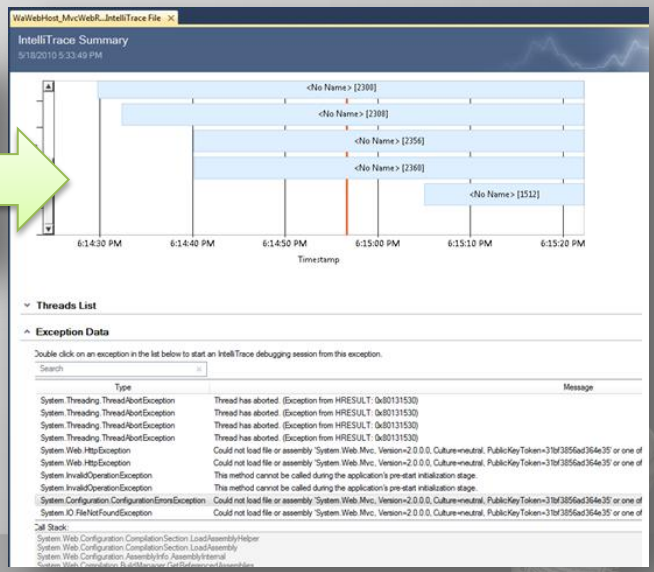
Remote Diagnostics config.

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

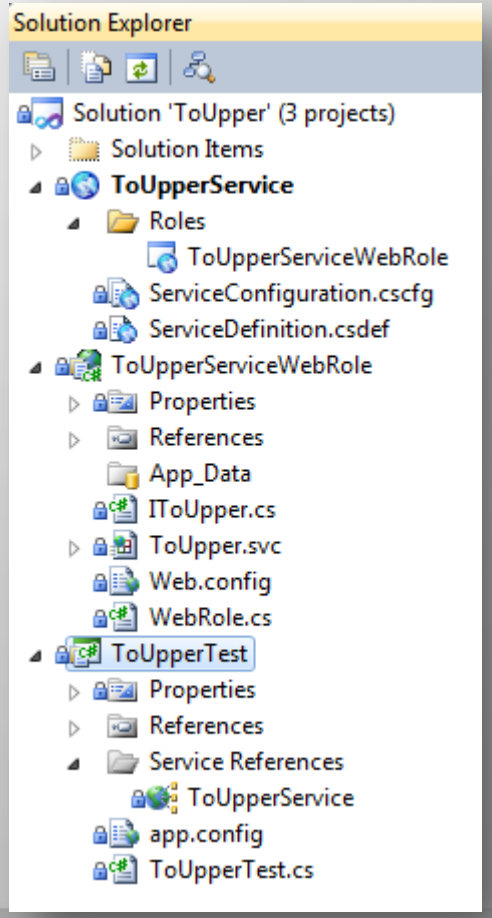
Build Automation with Azure

- 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

Instead of User/Pwd

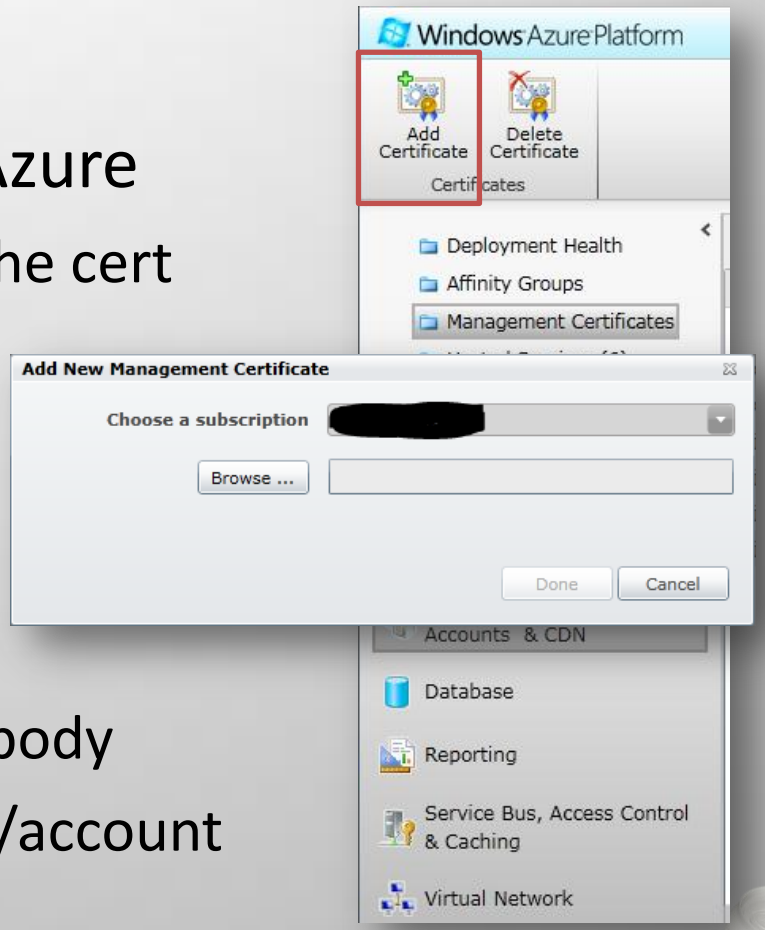


Setting up certificates

- 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 managment
    - 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

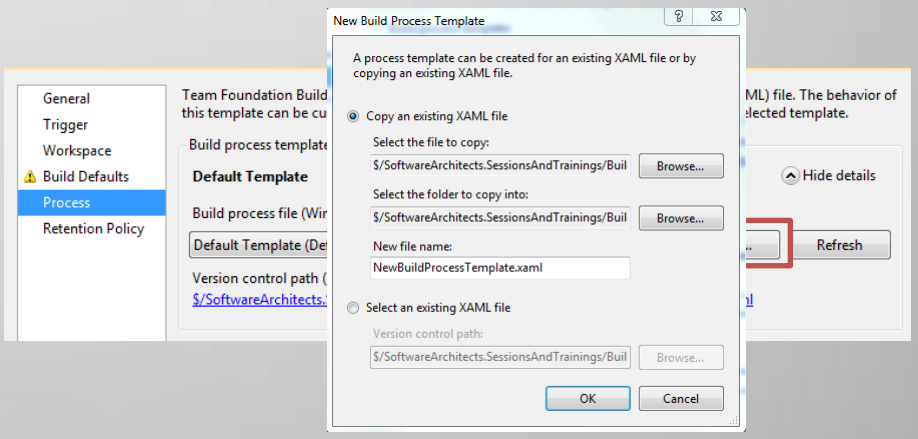
Setting up certificates

- 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

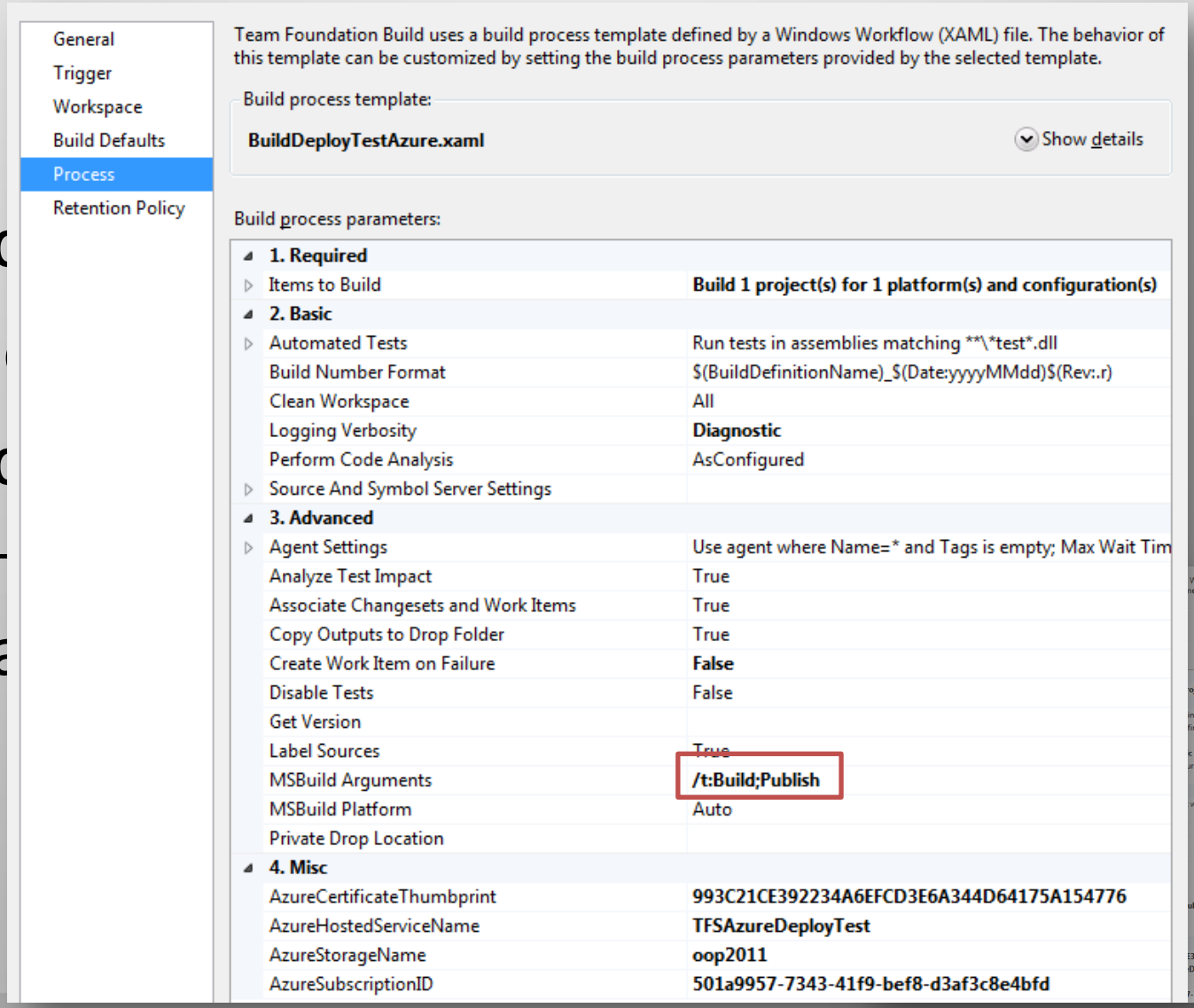
- Goals
  - Create a build process template for customization
  - Build cspkg in Build Process
  - Deploy to Azure using Azure Powershell Cmdlets
  - Run Unit test against newly deployed service
  - Remove Hosted Service
    - Unless you have too much $

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

- Ec
- G
- Ec
  –
- Sa



Build Automation with Azure

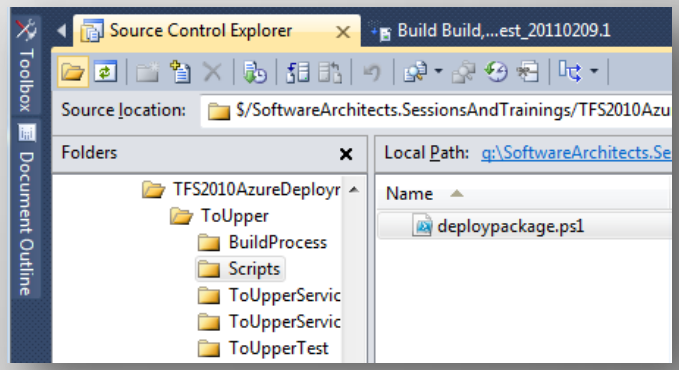
- Goals
  - Create a build process template for customization
  - Build cspkg in Build Process
  - Deploy to Azure using Azure Powershell Cmdlets
  - Run Unit test against newly deployed service
  - Remove Hosted Service
    - Unless you have too much $

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

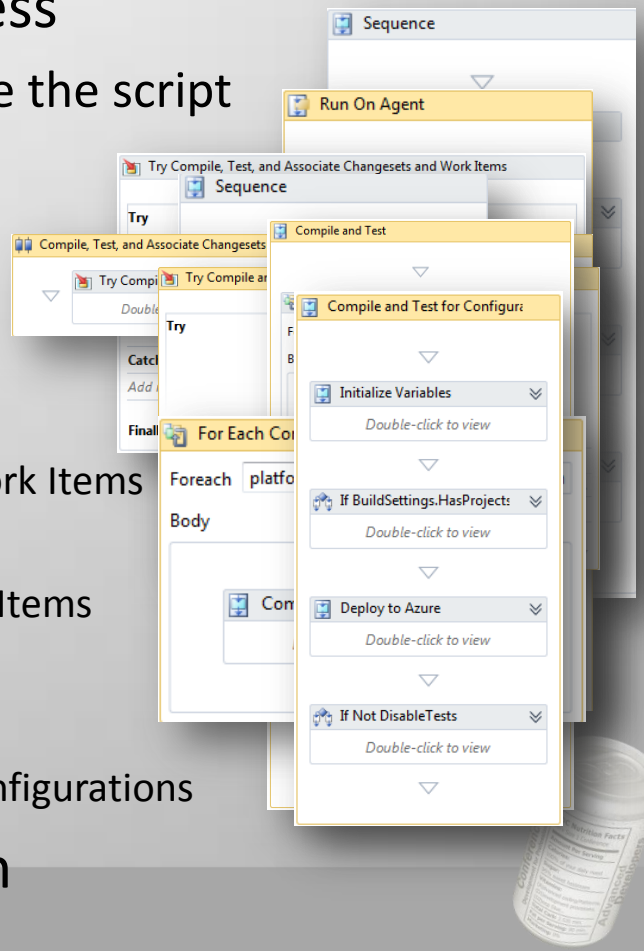


Goal: Deploying to Azure

```
# 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

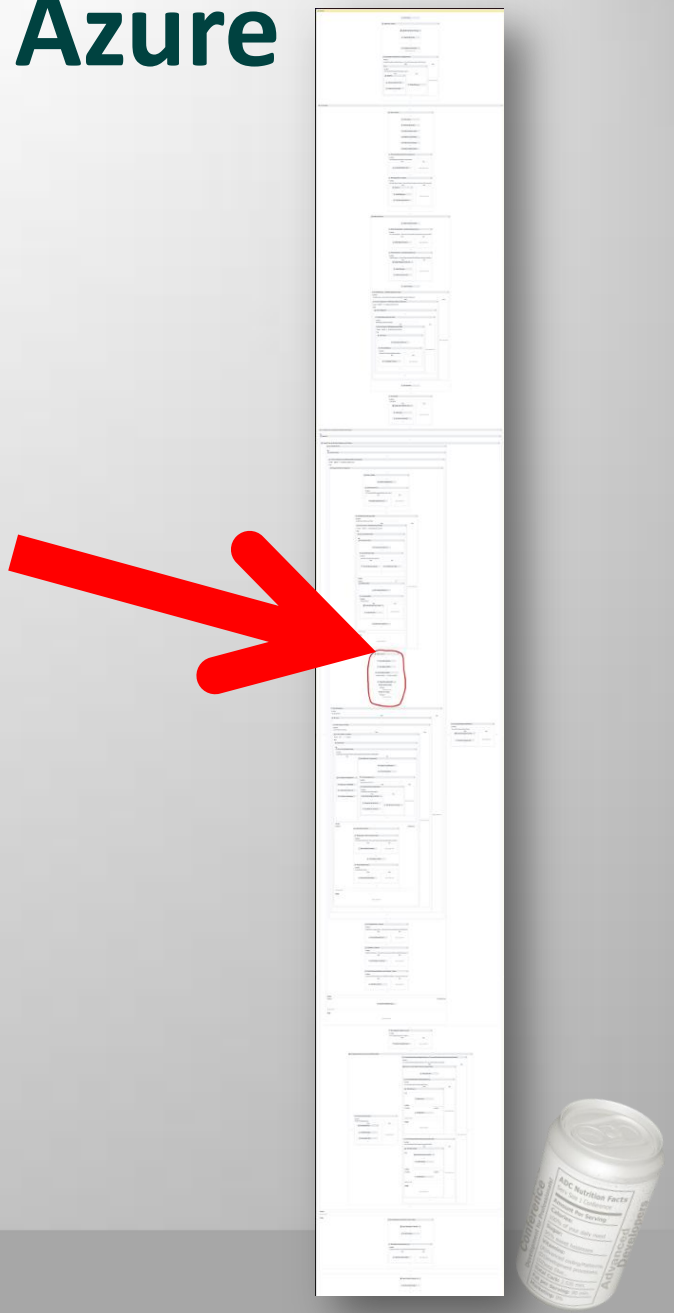
- Executing the script in the Build Process
  - Modify Build Process Template to execute the script
- Open the Build Template
- Navigate to
  - Sequence
  - Run On Agent
  - Try Compile, Test and Associate Changesets and Work Items
  - Sequence
  - Compile, Test, and Associate Changesets and Work Items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



Goal: Deploying to Azure

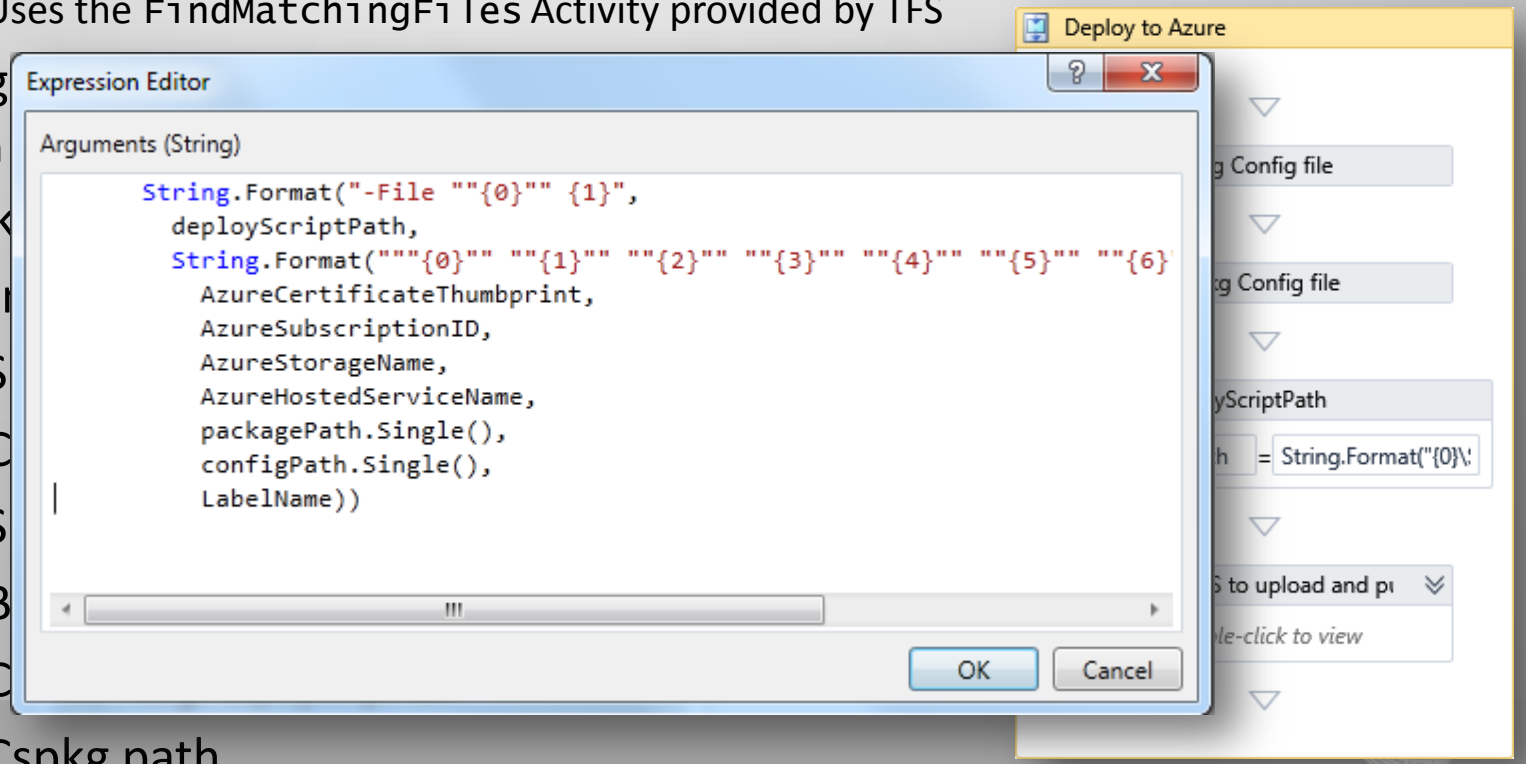
You are Here!

;-)



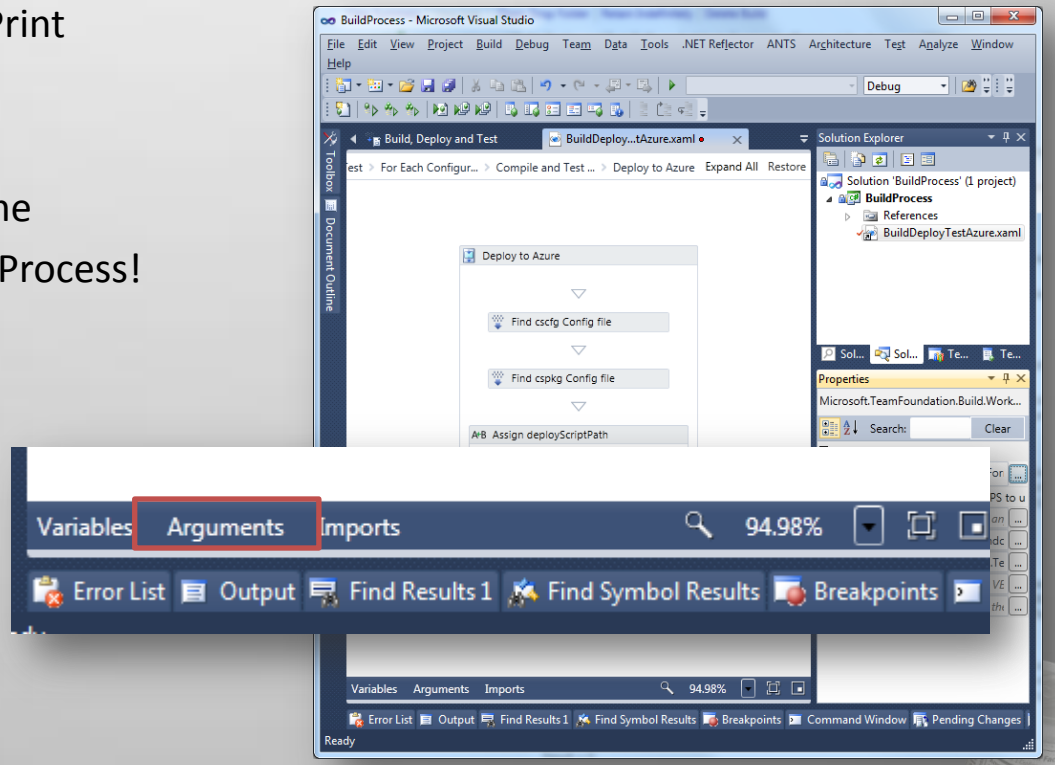
Goal: Deploying to Azure

- This “Deploy to Azure” Sequence
- Finds the cscfg and cspkg files in the Publish directory
  - Uses the `FindMatchingFiles` Activity provided by TFS
- Assig
  - a
- Invok
- Argu
  - S
  - C
  - S
  - B
  - C
  - Cspkg path



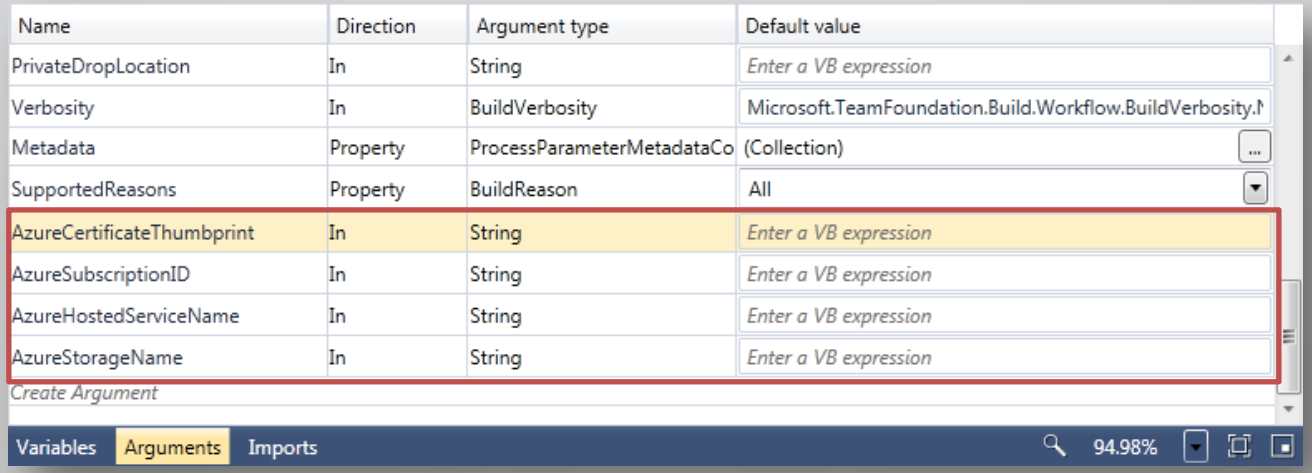
Goal: Deploying to Azure

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



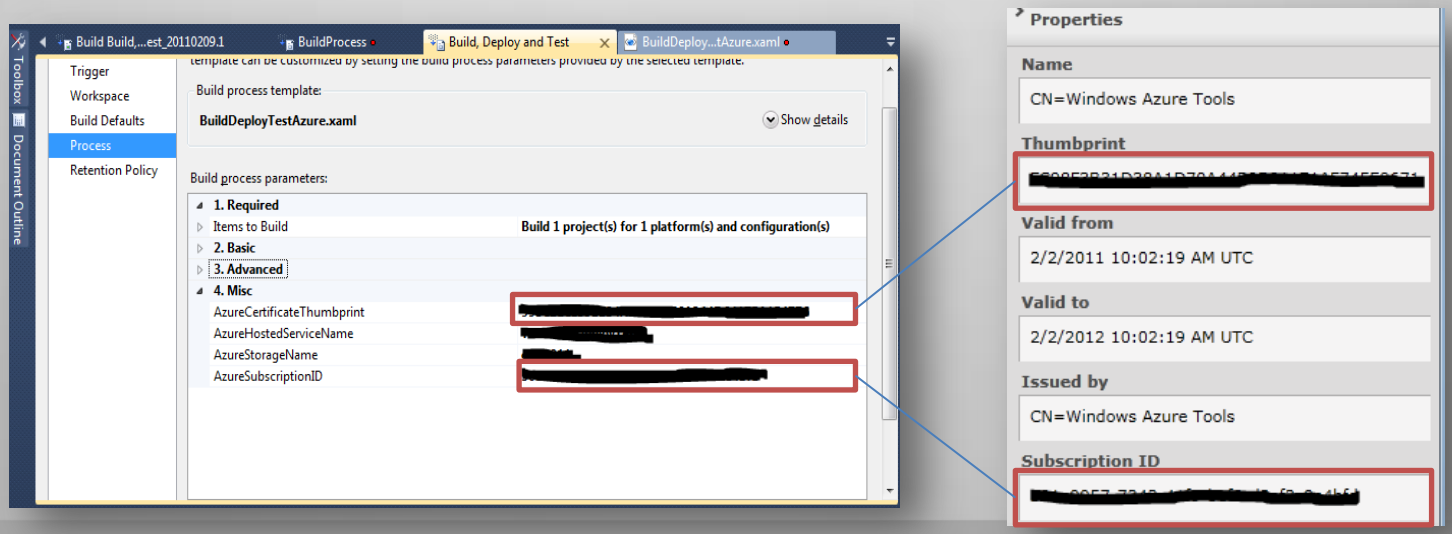
Goal: Deploying to Azure

- Create Arguments



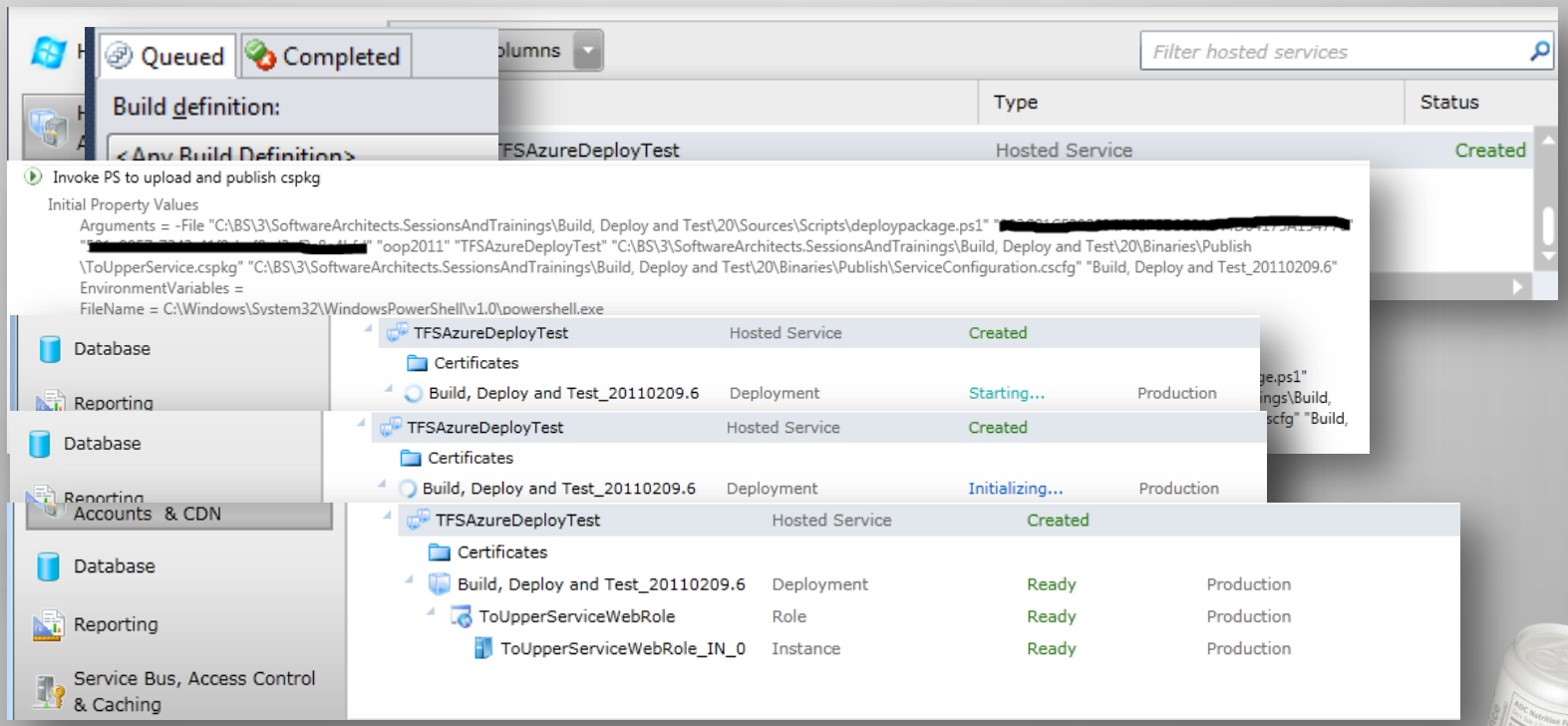
Goal: Deploying to Azure

- 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)



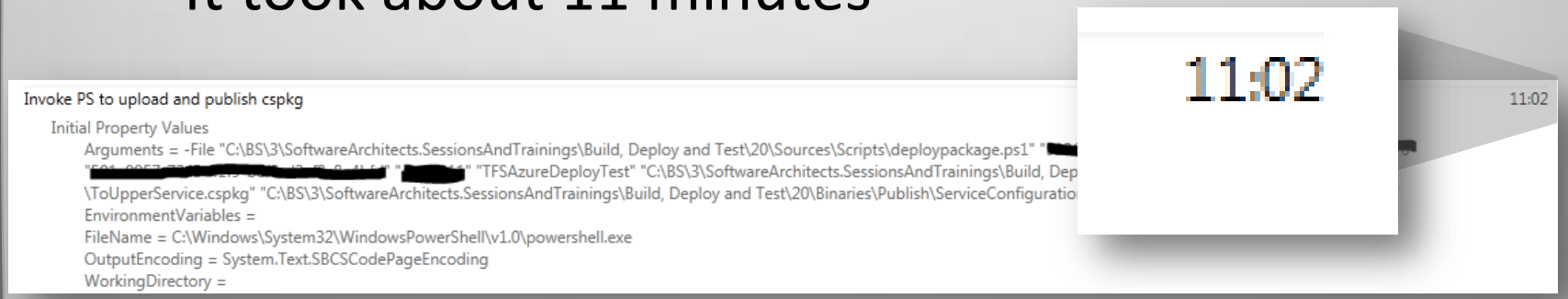
Goal: Deploying to Azure

- Let‘s try it! – Queue a new build



Goal: Deploying to Azure

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



Build Automation with Azure

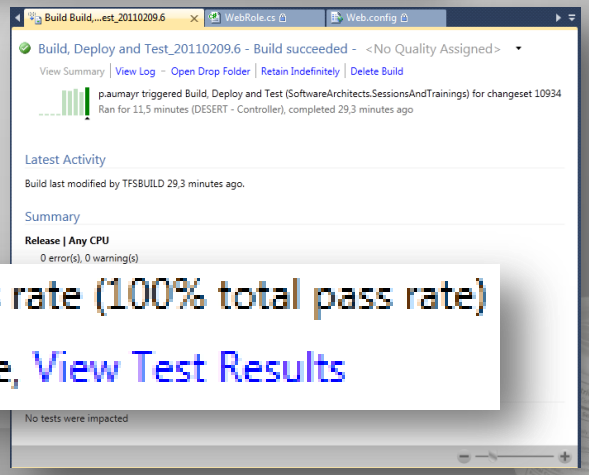
- Goals
  - Create a build process template for customization
  - Build cspkg in Build Process
  - Deploy to Azure using Azure Powershell Cmdlets
  - Run Unit test against newly deployed service
  - Remove Hosted Service
    - Unless you have too much $

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)

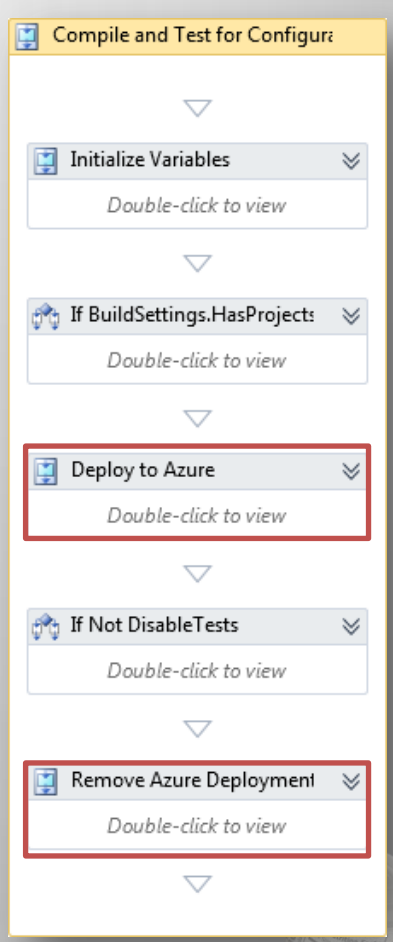
1/1 test(s) passed, 0 failed, 0 inconclusive, View Test Results

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?

Veranstalter: ppedv

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!

Veranstalter: ppedv

Advanced Developers Conference
Development for Professionals!

Speziell zu Software-Testing

14.-15. Februar 2011, München

Vielen Dank!

Rainer Stropek

Veranstalter: ppedv