Deploying XML configuration files

One of the essential steps in deploying software is configuring it to work in a specific environment. This might mean pointing your application to the right database connection string, or tweaking settings to run in production.

Deployment Manager does two things to make configuring your application easy:

1. Automatically updating `appSettings` and `connectionStrings`
2. Executing configuration file transformations

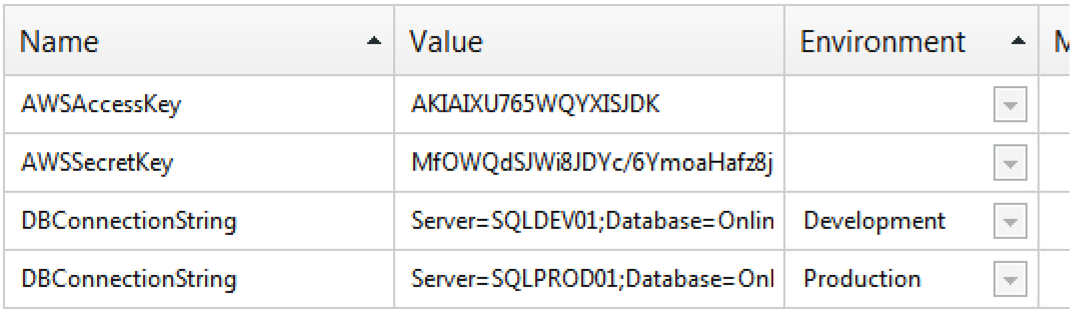
App Settings and Connection Strings

If a variable is defined in the Deployment Manager web interface, and an *appSettings* or *connectionString* record exists for it in any of your .config files, the Agent will automatically replace the value after extracting your package.

For example, suppose you have this configuration file:

```
<configuration>
<appSettings>
<add key="AWSAccessKey" value="testkey"/>
<add key="AWSSecretKey" value="testsecret"/>
</appSettings>
<connectionStrings>
<add name="DBConnectionString" connectionString="Server=(local)\SQLExpress;Database=OnlineStore;Integrated
Security=SSPI" />
</connectionStrings>
</configuration>
```

And you have these variables defined in your Deployment Manager web interface:



Name	Value	Environment
AWSAccessKey	AKIAIXU765WQYXISJDK	
AWSSecretKey	MfOWQdSJWi8JDYc/6YmoaHafz8j	
DBConnectionString	Server=SQLDEV01;Database=Onlin	Development
DBConnectionString	Server=SQLPROD01;Database=Onl	Production

After deploying to an environment named *Production*, Deployment Manager will have updated the file to:

```
<configuration>
<appSettings>
<add key="AWSAccessKey" value="AKIAIXU765WQYXISJDK"/>
<add key="AWSSecretKey" value="MfOWQdSJWi8JDYc/6YmoaHafz8jByBl9aksCoSLB"/>
</appSettings>
<connectionStrings>
<add name="DBConnectionString" connectionString="Server=SQLPROD01;Database=OnlineStore;Integrated
Security=SSPI" />
</connectionStrings>
</configuration>
```

This only works for appSettings and connectionStrings elements.

Configuration File Transformations

After deployment, Deployment Manager will also look for any files that follow the Microsoft web.config transformation process even files that are not web. config files (for example, `YourService.exe.config`).

The configuration transformation files can either be named `*.Release.config`, or `*.<Environment>.config`.

For example, suppose you have the following files in your package:

- Web.config
- Web.Release.config
- Web.Production.config
- Web.Test.config

When deploying to an environment named *Production*, Deployment Manager will execute `Web.Release.config`, followed by `Web.Production.config`.

An example web.config transformation that removes the `<compilation debug="true">` attribute is below:

```
 <?xml version="1.0"?>
<configuration xmlns:xdt="http://schemas.microsoft.com/XML-Document-Transform">
<system.web>
<compilation xdt:Transform="RemoveAttributes(debug)" />
</system.web>
</configuration>
```

The team at AppHarbor created a useful tool to help test configuration file transformations.

PowerShell

If these conventions aren't enough to configure your application, you can always use PowerShell to perform custom configuration tasks. Variables will be passed to your PowerShell script, and PowerShell has rich XML APIs.