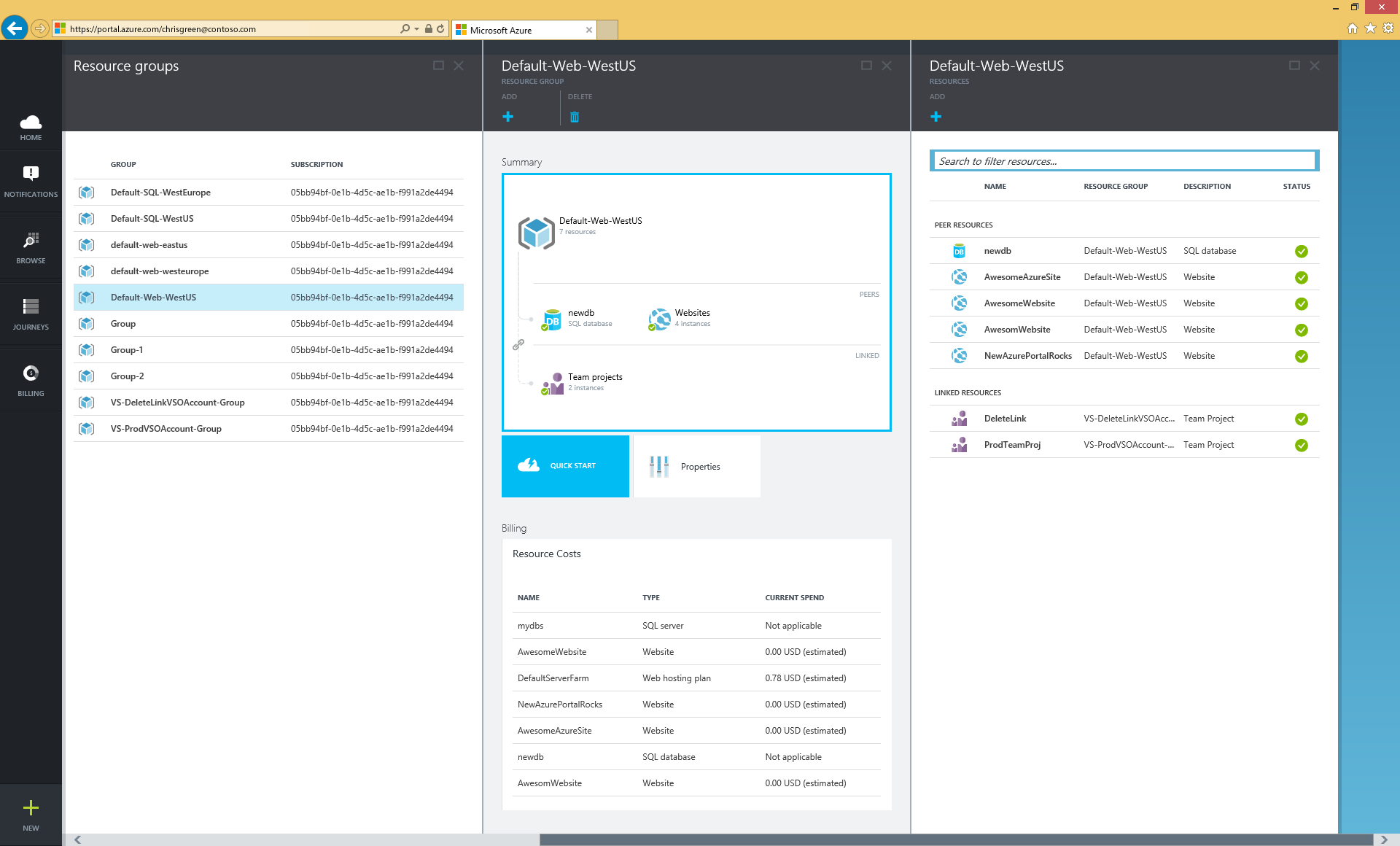
Весной 2014 года Microsoft анонсировала новую версию портала в стадии preview, и в ней появилась Resource Group.

До нее группировка различных ресурсов по логическому принципу, по сути, отсутствовала. Были отдельные базы данных, отдельные сайты, отдельно storage, каждый на своей вкладке. Осознать, как и какие сущности связаны между собой, к каким приложениям относятся было занятием не всегда возможным. Как правило, решалось на уровне именования сущностей по определенному шаблону типа: ApplicationName1\_web\_1\_Prod, ApplicationName2\_db\_2\_Test. Но это не решение проблемы, т.к. надо было просмотреть все типы ресурсов, чтобы составить общую картину.

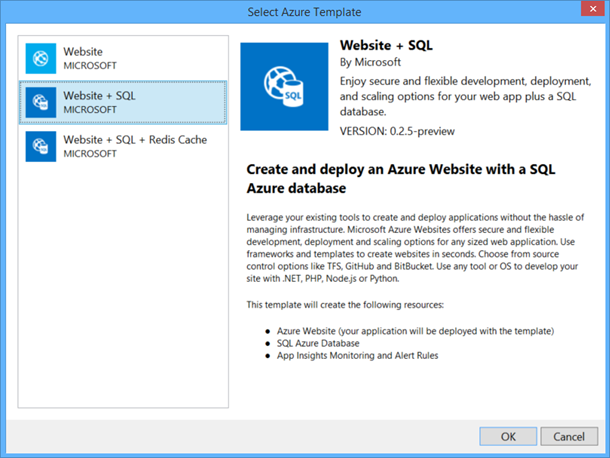
**Resource Group - это логическая группировка ресурсов. (баз данных, веб серверов и т.п.)**

****

Но это было уже более полугода назад. С конца осени 2014 года, когда вышла Azure SDK 2.5, Resource Group стала использоваться не только для логической группировки, но на ее основе стало возможным публиковать все части приложения из visual studio в пару кликов.

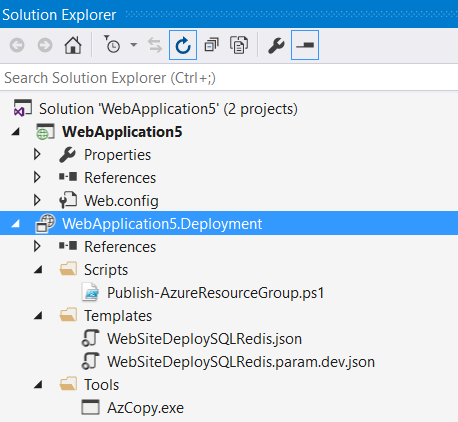
Создается json описание ресурсов в группе, добавить к этому файл трансформации (чтобы иметь возможность замены значений параметров по необходимости), и, по нажатии на кнопку публикации, мы можем раскидать все наши ресурсы по нужным местам.

Есть несколько шаблонов, на основании которых создаются deploy конфигурации.



На мой взгляд, нужно было создать самую простую конфигурацию, вообще без Application Insight. На ее основе было бы куда проще разобраться с тем, что находится внутри. Но, к сожалению, даже в базовом шаблоне WebSite Application Insight уже добавлен. Я его скрыл, чтобы было проще рассказывать о внутренней структуре.

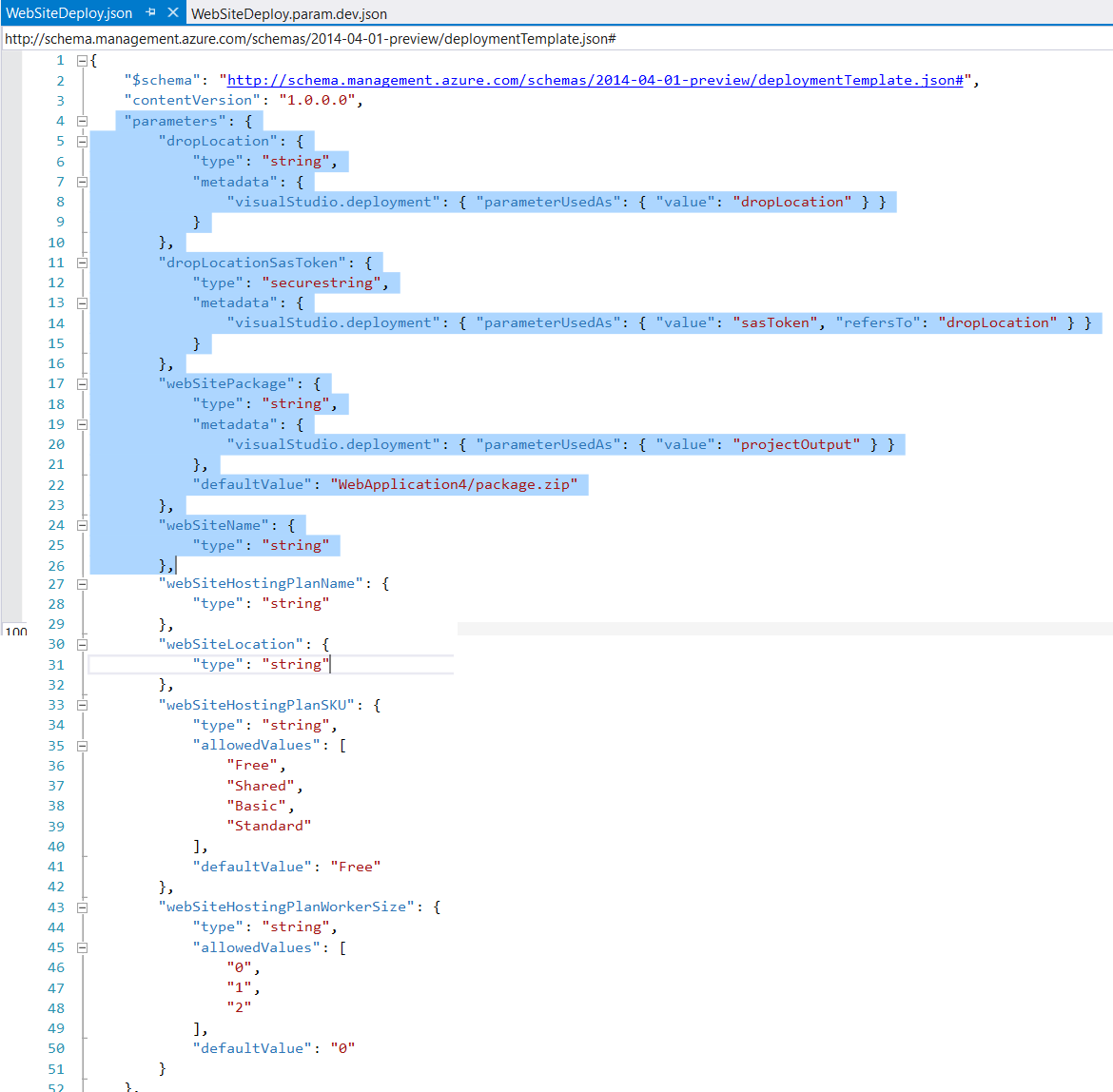
В итоге получится проект



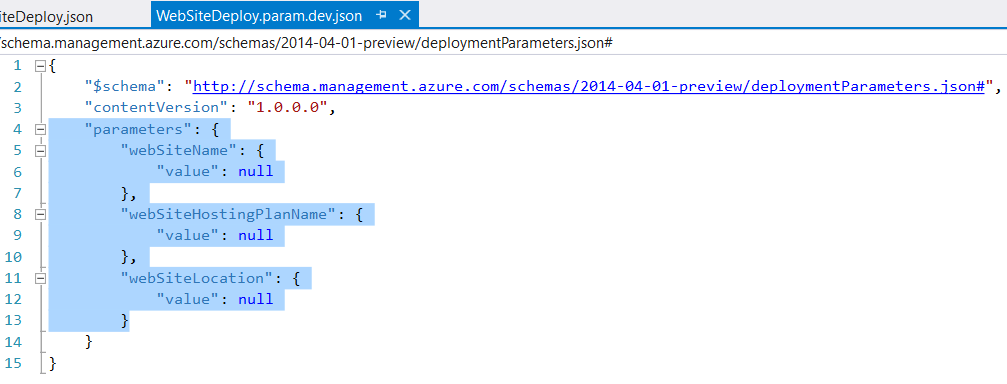
В папке Template лежит все, что нас интересует.

Есть блок параметров, которые мы можем менять в зависимости от конфигурации.

В нем не хранятся значения, за исключением параметра по умолчанию, только описание имен параметров, типы данных, ну и список возможных значений параметров перечислений.

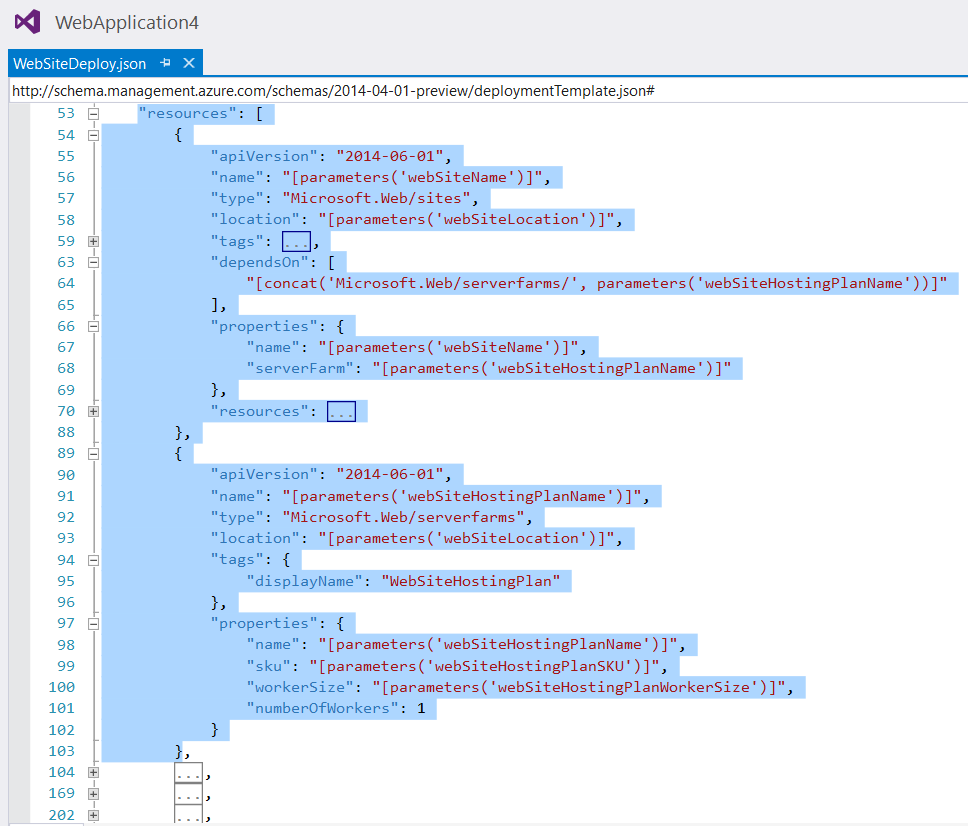


В другом файле, который называется \*.param.dev\* находятся значения этих параметров, которые подставляются при сборке.



\*dev\* - это имя конфигурации, к которой относятся параметры. Таких файлов с конфигурациями можно и нужно создать столько же, во сколько Resource Group мы будем проводить публикацию.

С параметрами разобрались, теперь переходим к определению самих ресурсов.



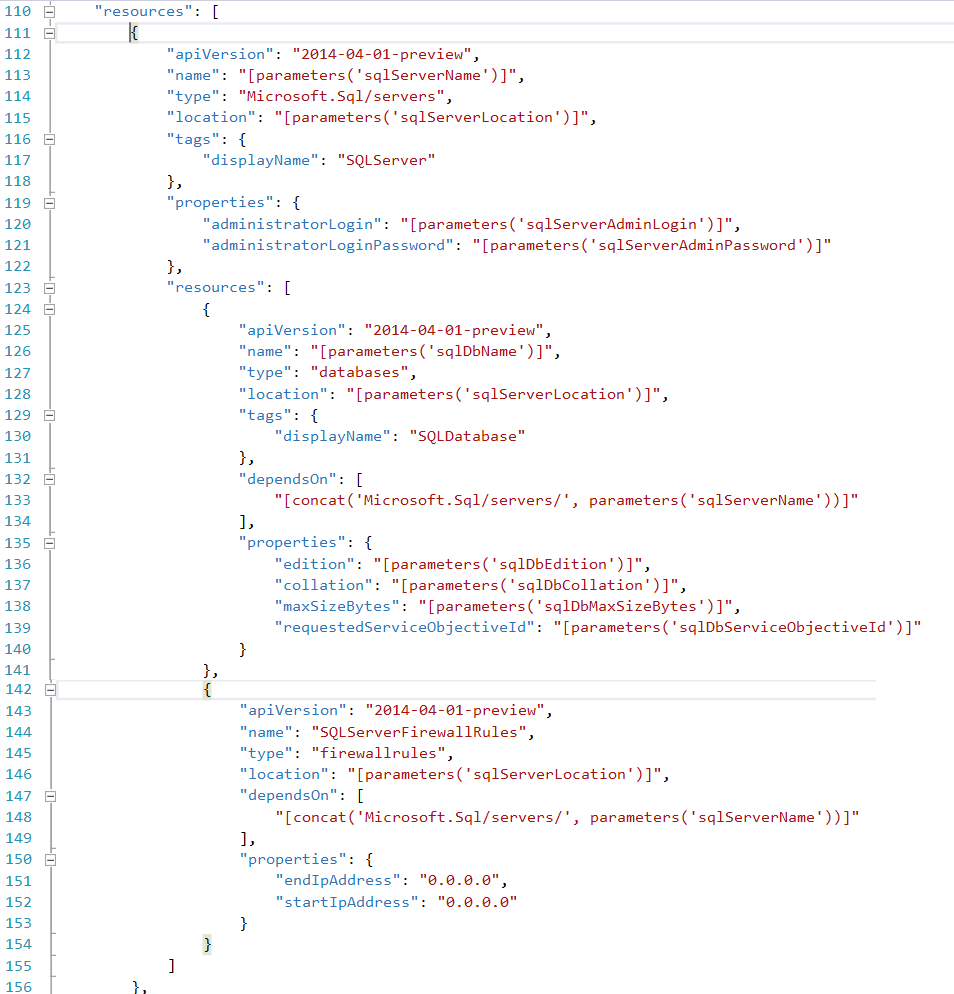
Здесь мы видим 2 ресурса. WebSite и ServerFarm, причем первый зависит от второго. После проведения трансформации вместо “parameters(‘’)” будет подставлено значение из \*param\* файла и файл будет простым Json документом. Значение каждого конкретного поля комментировать смысла нет, они понятны тем, кто пользовался Azure ранее, а для остальных можно прочесть в статье<http://azure.microsoft.com/blog/2014/11/26/azure-resource-manager-2-5-for-visual-studio/>

Я умышленно закрыл раздел resources для WebSite, чтобы было нагляднее.

Внутри ресурса - стандартные MSDEPLOY



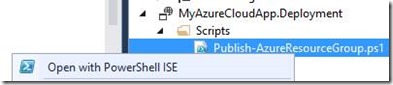
Если вы возьмете шаблон с SQL Server и/или Redis, то у вас просто добавятся 2 секции примерно такого-же формата. У Redis секция вообще внимания не заслуживает, у SQL Server, минимальный набор параметров есть.



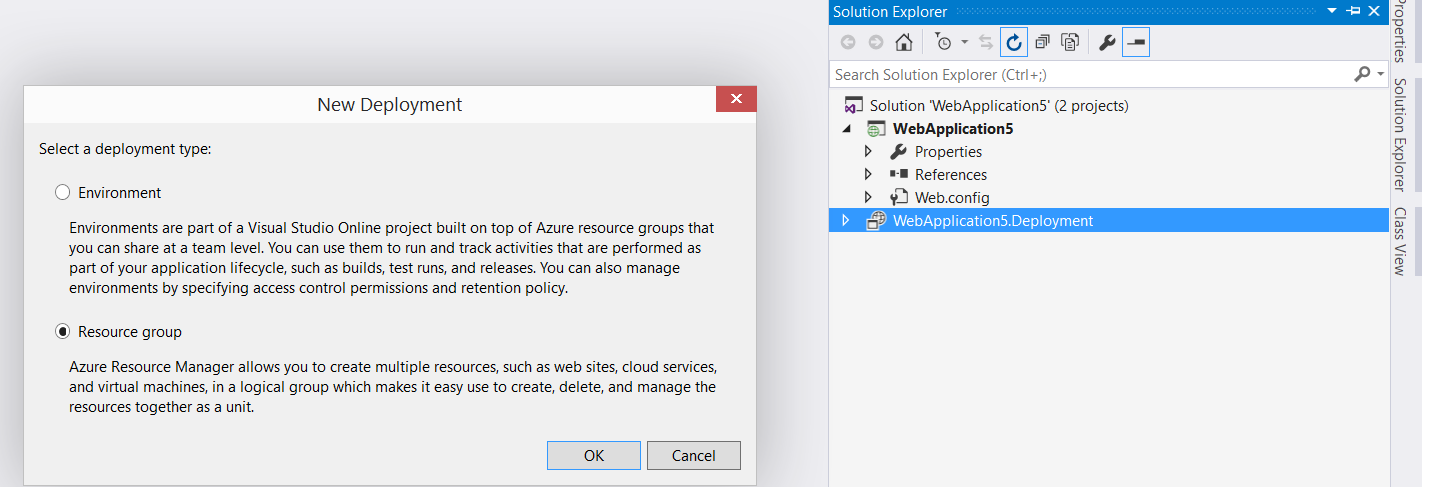
**Процесс публикации**

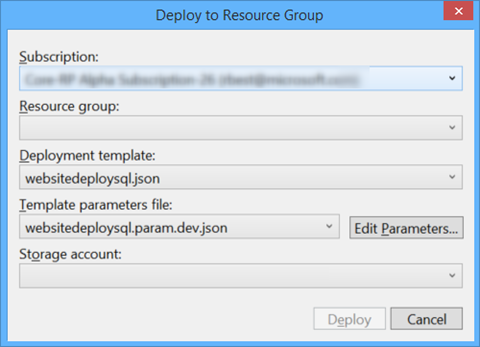
У Вас есть целых 2 способа опубликовать результаты:

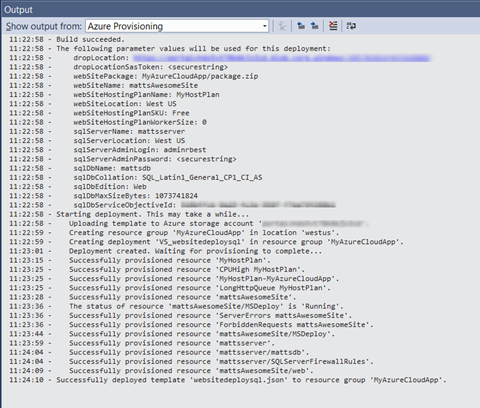
Либо исполняете powershell скрипт, заботливо созданный разработчиками для автоматизации.



Либо нажимаете кнопку deploy на проекте,



выбрав на основании какой подписки Azure и каких параметров проводить публикацию. 

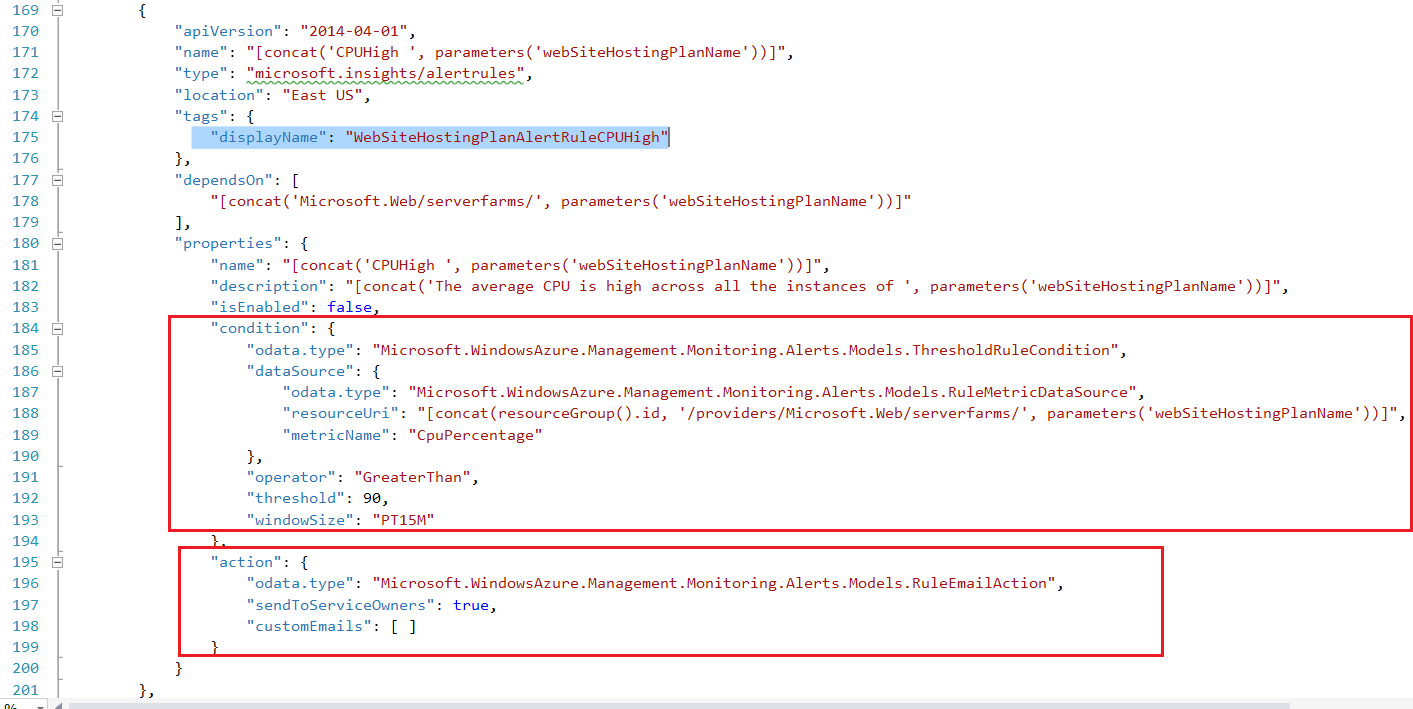
За процессом работы мы можем наблюдать в консоли Output 

А сам результат увидим уже на портале.

**Application Insight Rules**

Тем, кто использует Application Insight, стоит обратить внимание на описание этих правил в документе, которые при деплое будут опубликованы вместе с проектом.

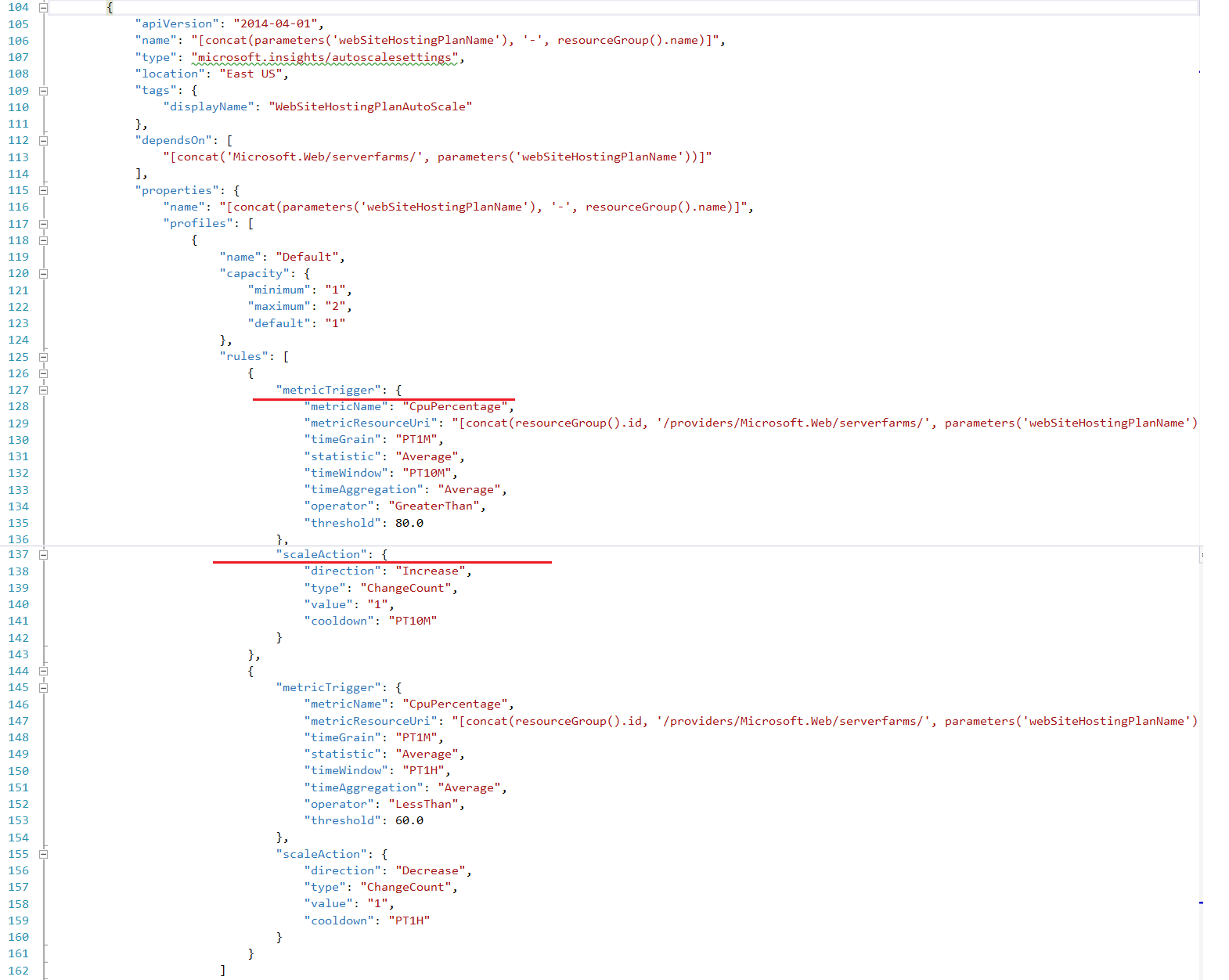
Большая часть правил выглядит примерно одинаково. Вот пример правила, по которому если загрузка процессора выше 90%, то отправляется нотификация. Правило состоит из 2 основных частей: Condition - условие, которое должно выполниться, и Action- действие, которое выполнится по условию.



В примерах почти все правила такие. Есть нотификация при увеличении длины очереди запросов к IIS, появлении множества 400\* ошибок (есть вероятность, что вас пытаются сломать).

Есть еще один тип, который я бы выделил уже потому, что он структурно не похож на вышеописанные. Это AutoScaling - автомасштабирование.

Он сложнее, ведь у нас есть правило на увеличение числа машин и на уменьшение в зависимости от текущего значения нагрузки на CPU.



**Нюансы**

Понятно, что при таком деплое мы не полностью развернем работоспособные приложения с нуля, как минимум, в базе нет данных.

Зато у нас появилась автоматизация процесса создания ресурсов, которую мы дальше можем улучшать по своему усмотрению, а наши действия стали более повторимыми “repeatable”.

**Ссылки**

* Анонс SDK 2.5 <http://azure.microsoft.com/blog/2014/11/12/announcing-azure-sdk-2-5-for-net-and-visual-studio-2015-preview/> <http://msdn.microsoft.com/en-us/library/azure/dn873976.aspx>
* Preview версия тулзов к студии <http://azure.microsoft.com/blog/2014/08/11/azure-resource-manager-tools-preview/>
* Релизная версия тулзов к студии <http://azure.microsoft.com/blog/2014/11/26/azure-resource-manager-2-5-for-visual-studio/>
* Resource groups <http://azure.microsoft.com/en-us/documentation/articles/azure-preview-portal-using-resource-groups/>