Template Azure Resource Manager com DSC para Instalação do SonarQube no Azure

# O que é o SonarQube?

Débito técnico é um problema crescente para muitos desenvolvedores, a medida que um aplicativo amadurece e código se torna mais estável, que pode se acumular para criar ineficiências e minar a produtividade da equipe para criar novos aperfeiçoamentos. Isso ocorre principalmente porque o código se torna mais complexos e, por consequência, mais difícil de entender. Isso cria obstáculos imprevistos que podem reduzir a capacidade da equipe entregar cada vez mais valor.

SonarQube é uma plataforma é uma plataforma de código aberto que auxilia os desenvolvedores no isolamento e gerenciamento do débito técnico. A plataforma fornece analise dos débitos técnicos e podem fazer parte do processo de compilação. Ao longo dos últimos meses, SonarQube expandiu-se e passou a suportar não só o Java, mas também .Net e JavaScript na análise de código.

Apesar dessas vantagens, a grande barreira para adoção do SonarQube em muitas organizações é o seu processo de configuração, bem como as etapas necessárias para a implantação de uma instância do SonarQube. Essa solução provê um método para acelerar essa adoção utilizando templates de Azure Resource Manager (ARM) e PowerShell DSC (Desired State Configuration) para automatizar a criação dos componentes necessários para suportar a instalação do SonarQube no seu ambiente de desenvolvimento.

# O que é o Azure Resource Manager

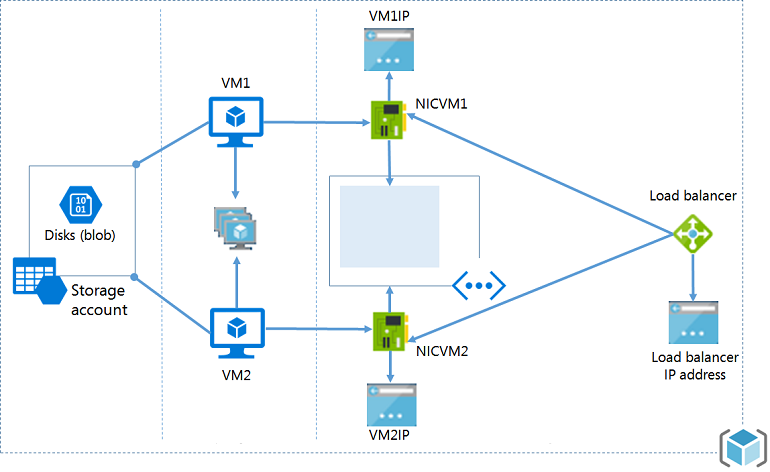
O Azure Resource Manager (ARM – Gerenciador de Recursos Azure) permite o provisionamento de aplicativos usando um template declarativo. Em um único template, é possível implantar múltiplos serviços e suas respectivas dependências. É possível reutilizar o mesmo template para fazer a implantação do mesmo aplicativo durante cada estágio do ciclo de vida do aplicativo.

<https://azure.microsoft.com/en-us/documentation/templates/>

Uma das primeiras questões quando se cria um template é “como começar? ” Uma opção é começar a partir de um template básico, seguindo a estrutura descrita no artigo [Criando uma Template](https://azure.microsoft.com/en-us/documentation/articles/resource-group-authoring-templates/#template-format), adicionando os parâmetros, variáveis e os recursos apropriados. Uma boa alternativa seria começar por dar uma olhada na [galeria quickstart](https://github.com/Azure/azure-quickstart-templates) e procurar cenários similares aos que você está tentando criar. Você pode combinar vários templates ou editar uma já existente para seu cenário específico.

A seguir vamos examinar uma infraestrutura comum:

* Duas máquinas virtuais que usam a mesma conta para armazenamento, estão no mesmo conjunto de disponibilidade, e estão na mesma sub-rede de uma rede virtual.
* Uma única NIC (placa de interface de rede) e um único endereço IP para cada máquina virtual
* Um balanceador de carga de rede com uma regra de balanceamento na porta 80



Este tópico instrui o passo-a-passo de como criar uma template ARM para esta infraestrutura. O template final que você cria é baseado em um template QuickStart chamado “[2 VMs in a Load Balancer and load balancing rules](https://azure.microsoft.com/documentation/templates/201-2-vms-loadbalancer-lbrules/)”.

Mas, como isto é muita coisa para se fazer de uma vez só, vamos primeiro criar uma conta de armazenamento e implantá-la. Depois que você tiver dominado de como criar a conta de armazenamento, você adicionará os outros recursos e reimplantará o template para completar a infraestrutura.

##### **Atenção:**

Você pode usar qualquer tipo de editor quando criar o template. O Visual Studio provê ferramentas que simplificam o desenvolvimento de templates, mas não é necessário ter o Visual Studio para completar este tutorial. Para um tutorial sobre como usar Visual Studio para criar uma implantação de um aplicativo Web e banco de dados SQL, veja o tópico [Creating and deploying Azure resource groups through Visual Studio](https://azure.microsoft.com/en-us/documentation/articles/vs-azure-tools-resource-groups-deployment-projects-create-deploy/).

## Criando o template ARM

Um template é um arquivo JSON que define todos os recursos que você vai implantar. Também permite a definição de parâmetros que são especificados durante a implantação, variáveis que são construídas a partir de outros valores e expressões, e resultados da implantação.

Vamos começar com o template mais simples:

Copy to clipboardCopy

{

"$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",

"contentVersion": "1.0.0.0",

"parameters": { },

"variables": { },

"resources": [ ],

"outputs": { }

}

Grave este arquivo com o nome de **azuredeploy.json** (o template pode ter qualquer nome, só necessita ser um arquivo do tipo json).

## Crie uma conta de armazenamento

Dentro da seção de **resources**, adicione um objeto que define uma conta de armazenamento, como mostrado abaixo:

Copy to clipboardCopy

"resources": [

{

"type": "Microsoft.Storage/storageAccounts",

"name": "[parameters('storageAccountName')]",

"apiVersion": "2015-06-15",

"location": "[resourceGroup().location]",

"properties": {

"accountType": "Standard\_LRS"

}

}

]

Você pode estar se perguntando de onde vêm estas propriedades e valores. As propriedades **type**, **name**, **apiVersion**, e **location** são elementos padrão que estão disponíveis para todos os tipos de recurso. Você pode aprender mais sobre estes elementos comuns no artigo [Resources](https://azure.microsoft.com/en-us/documentation/articles/resource-group-authoring-templates/#resources). **name** é atribuído a um valor de parâmetro que passado durante uma implantação, e **location** é a localização usada pelo grupo de recursos. Examinaremos como determinar **type** e **apiVersion** nas seções abaixo.

A seção de **properties** contém todas as propriedades que são únicas para determinado tipo de recurso. Os valores que você especifica nesta seção correspondem aos usados na operação PUT da REST API para a criação daquele tipo de recurso. Quando criado uma conta de armazenamento, você deve fornecer um **accountType**. Note que, no artigo [REST API for creating a Storage account](https://msdn.microsoft.com/library/azure/mt163564.aspx), a seção de propriedades da operação REST também contém uma propriedade **accountType**, e os valores permitidos estão documentados. Neste exemplo, o tipo de conta está configurado para **Standard\_LRS**, mas é possível especificar algum outro valor ou permitir a usuários que passem o tipo de conta como parâmetro.

Agora vamos voltar para a sessão **parameters**, e ver como se define o nome da conta de armazenamento. Para maiores informações sobre o uso de parâmetros, veja o artigo [Parameters](https://azure.microsoft.com/en-us/documentation/articles/resource-group-authoring-templates/#parameters).

Copy to clipboardCopy

"parameters" : {

"storageAccountName": {

"type": "string",

"metadata": {

"description": "Storage Account Name"

}

}

}

Aqui você definiu o parâmetro do tipo string que será o nome da conta de armazenamento. O valor deste parâmetro será usado durante a implantação de um template.

## Implantando uma template

Nós temos um template completo para criar uma conta de armazenamento, que foi salva com o nome de **azuredeploy.json**:

Copy to clipboardCopy

{

"$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",

"contentVersion": "1.0.0.0",

"parameters" : {

"storageAccountName": {

"type": "string",

"metadata": {

"description": "Storage Account Name"

}

}

},

"resources": [

{

"type": "Microsoft.Storage/storageAccounts",

"name": "[parameters('storageAccountName')]",

"apiVersion": "2015-06-15",

"location": "[resourceGroup().location]",

"properties": {

"accountType": "Standard\_LRS"

}

}

]

}

Há várias maneiras de implantar um template, como você pode verificar no artigo [Resource Deployment](https://azure.microsoft.com/en-us/documentation/articles/resource-group-template-deploy/). Para implantar um template com o Azure PowerShell, use:

# **Utilizando o Template SonarQube ARM**

# **Projeto Visual Studio:**

A solução é primeiramente feita a partir do template de projeto C# Azure Resource Group. Esse é composto primeiramente de dois arquivos:

\*.json: É a configuração do template.

\*.parameters.json: É uma listagem do dicionário dos parâmetros a ser aproveitado durante a implementação para customizar a solução.

O arquivo Azure Deploy Json identifica vários pontos-chaves para aprofundar a configuração deste modelo:

* Variáveis: Contém uma lista das variáveis que são aproveitados para a configuração deste template.
* Recursos: Contém todos os recursos que são implementados como parte dessa solução ARM. Isso inclui, a máquina virtual, a conta de armazenamento, e o NIC.

NOTA: mantenha nomes pequenos, há limite de 24 caracteres para contas de armazenamento, e essa concatenação irá adicionar um guid, de modo a manter os nomes de contas de armazenamento pequenos.

# **Template:**

Esse template contém vários artefatos.

Atualmente os principais componentes deste modelo são os seguintes itens:

* **An SQL Azure Server with a SQL Azure Database:** Essa plataforma como serviço fornece suporte para o banco de dados exigido pelo SonarQube.
* **An IaaS(Infrastructure as a Service) Virtual Machine with jdk8:** Esse servidor fornece o suporte necessário para a aplicação SonarQube. E com o SonarQube instalado, disponibiliza a configuração para o apontamento do banco de dados do SQL Azure acima identificado.

