# Table of Contents

- Receive alert notifications

## Related

- Azure Resource Manager template functions
- Best practices for Autoscale
- Common metrics for Autoscale
- Webhooks for Autoscale notifications
- Webhooks for alerts on audit logs
- Webhooks for alerts on metrics
- Azure Insights PowerShell quick start samples
- Azure Insights CLI quick start samples

## Resources

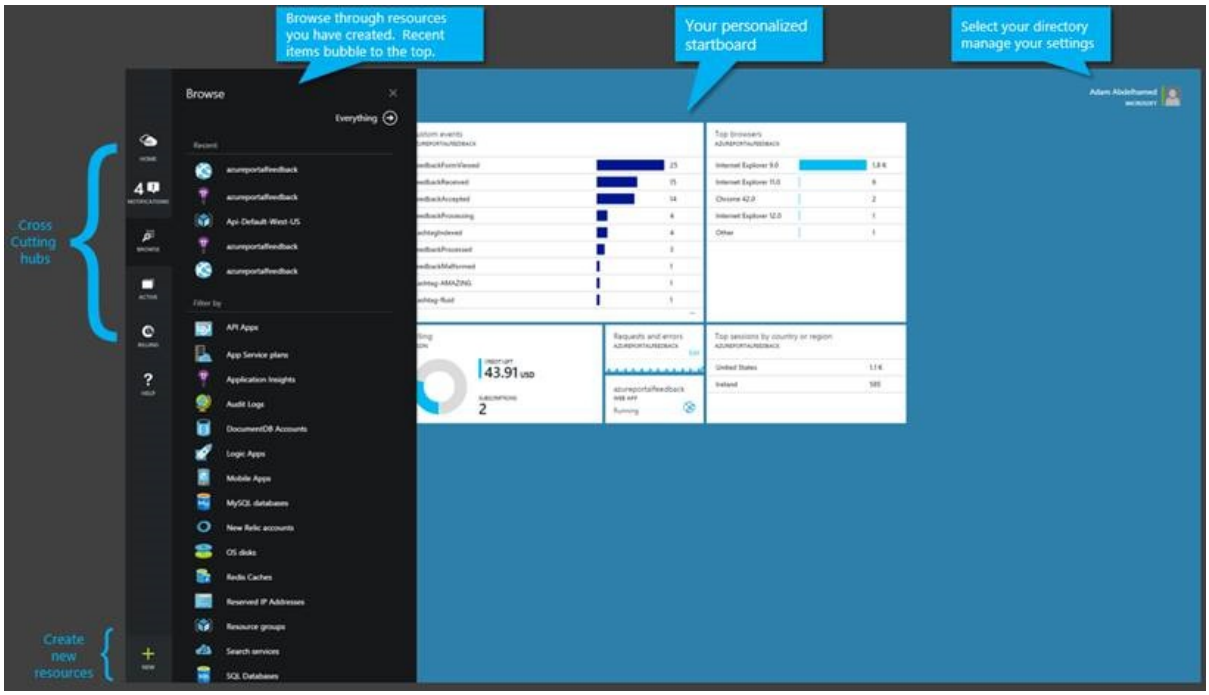
- Azure Roadmap
- MSDN forum
- Pricing calculator
- Stack Overflow

# Microsoft Azure portal overview

12/11/2017 • 3 min to read • Edit Online

The Microsoft Azure portal is a central place where you can provision and manage your Azure resources. This tutorial will familiarize you with the portal and show you how to use some of these key capabilities:

- A **comprehensive marketplace** that lets you browse through thousands of items from Microsoft and other vendors that can be purchased and/or provisioned.
- A **unified and scalable browse experience** that makes it easy to find the resources you care about and perform various management operations.
- **Consistent management pages** (or blades) that let you manage Azure's wide variety of services through a consistent way of exposing settings, actions, billing information, health monitoring and usage data, and much more.
- A **personal experience** that lets you create a customized start screen that shows the information that you want to see whenever you log in. You can also customize any of the management blades that contain tiles.




## Before you get started

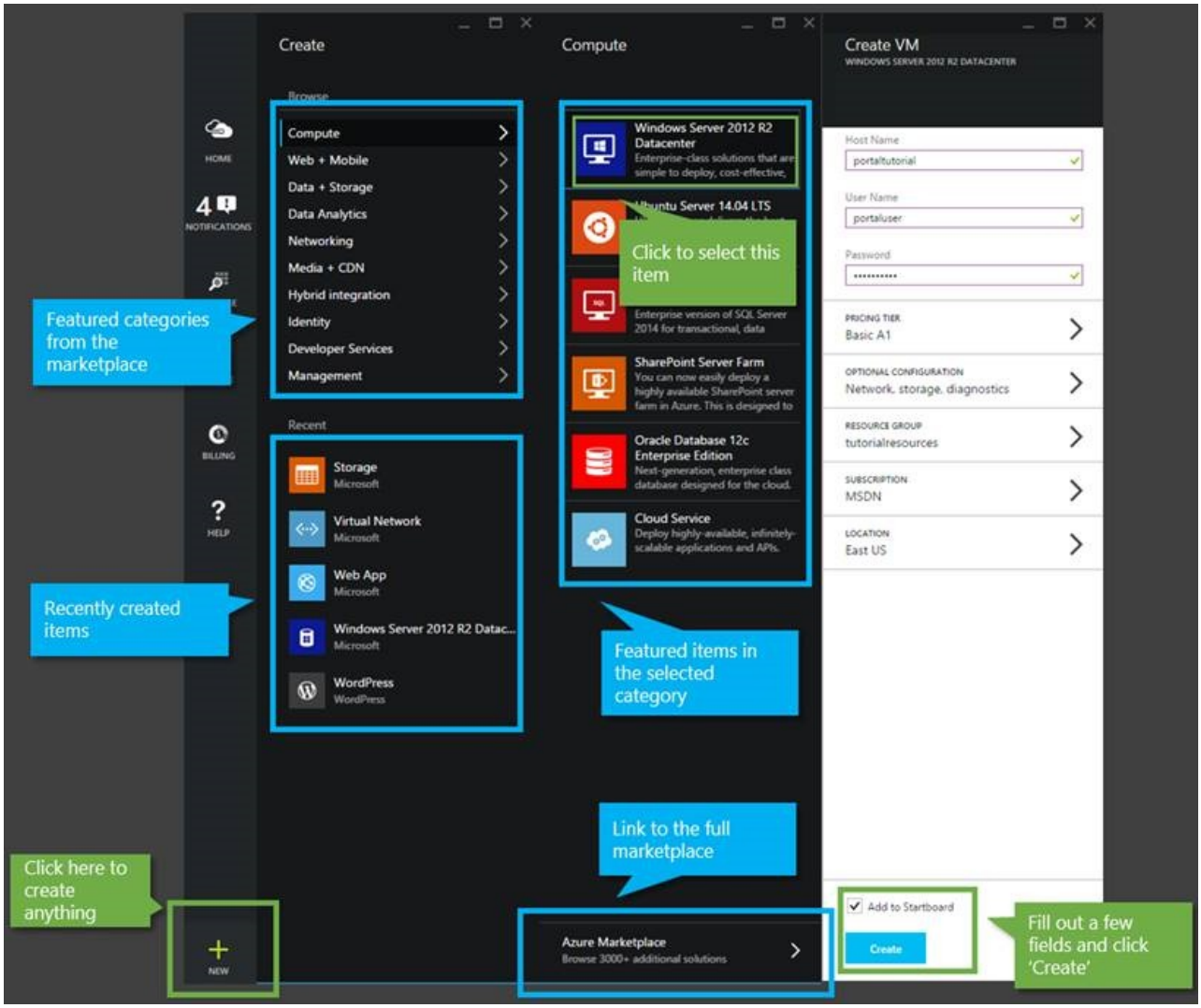
You will need a valid Azure subscription to go through this tutorial. If you don't have one, then sign up for a free trial today. Once you have a subscription, you can access the portal at https://portal.azure.com.

## How to create a resource

Azure has a marketplace with thousands of items that you can create from one place. Let's say you want to create a new Windows Server 2012 VM. The +NEW hub is your entry point into a curated set of featured categories from the marketplace. Each category has a small set of featured items along with a link to the full marketplace that shows all categories and search. To create that new Windows Server 2012 VM, perform the following actions:

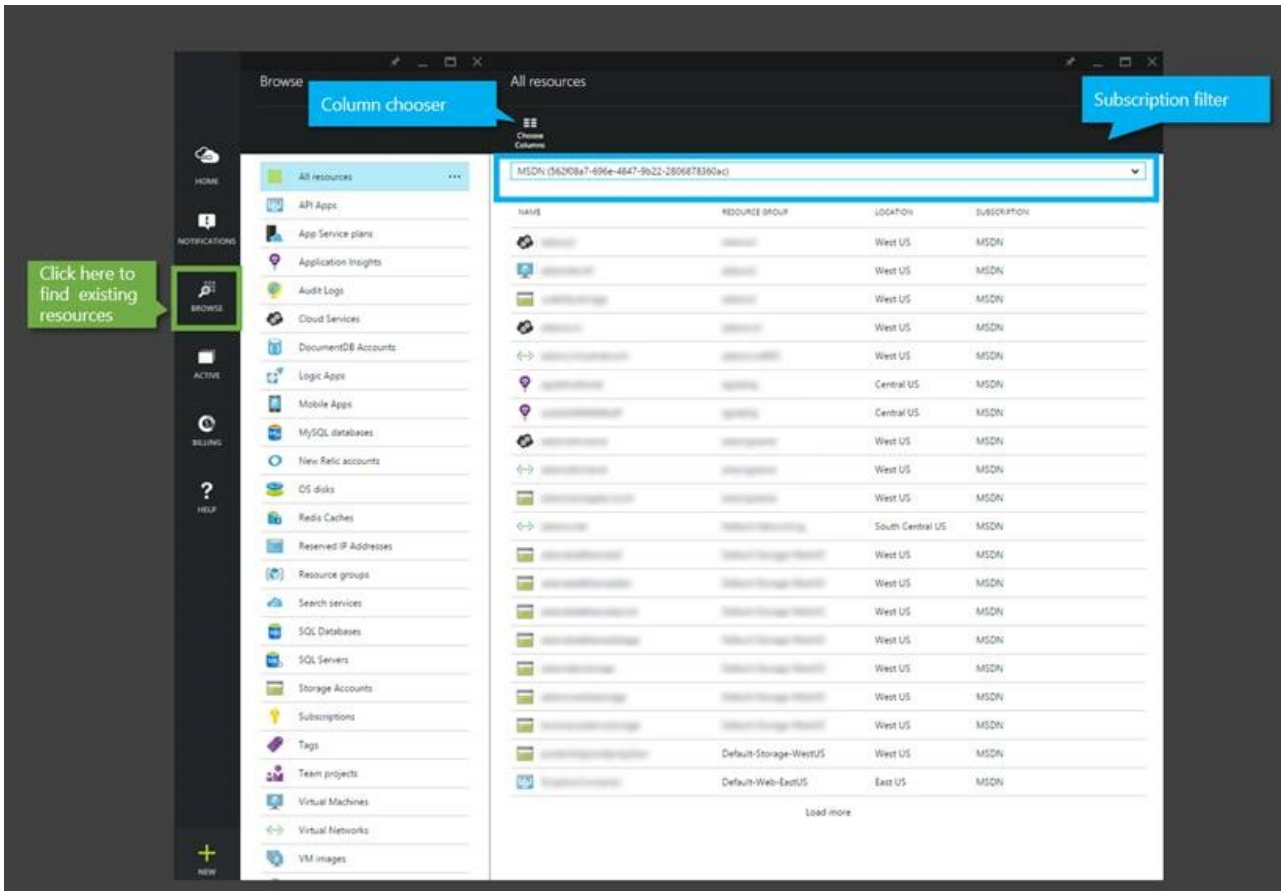
1. Windows Server 2012 is featured, so you can select it from the Compute category.
2. Fill out some basic inputs on a form.
3. Click 'Create' and your VM will begin to provision immediately.

The notifications hub will alert you when your resource has been created and a management blade will open (you can always browse to resources later).




## How to find your resources

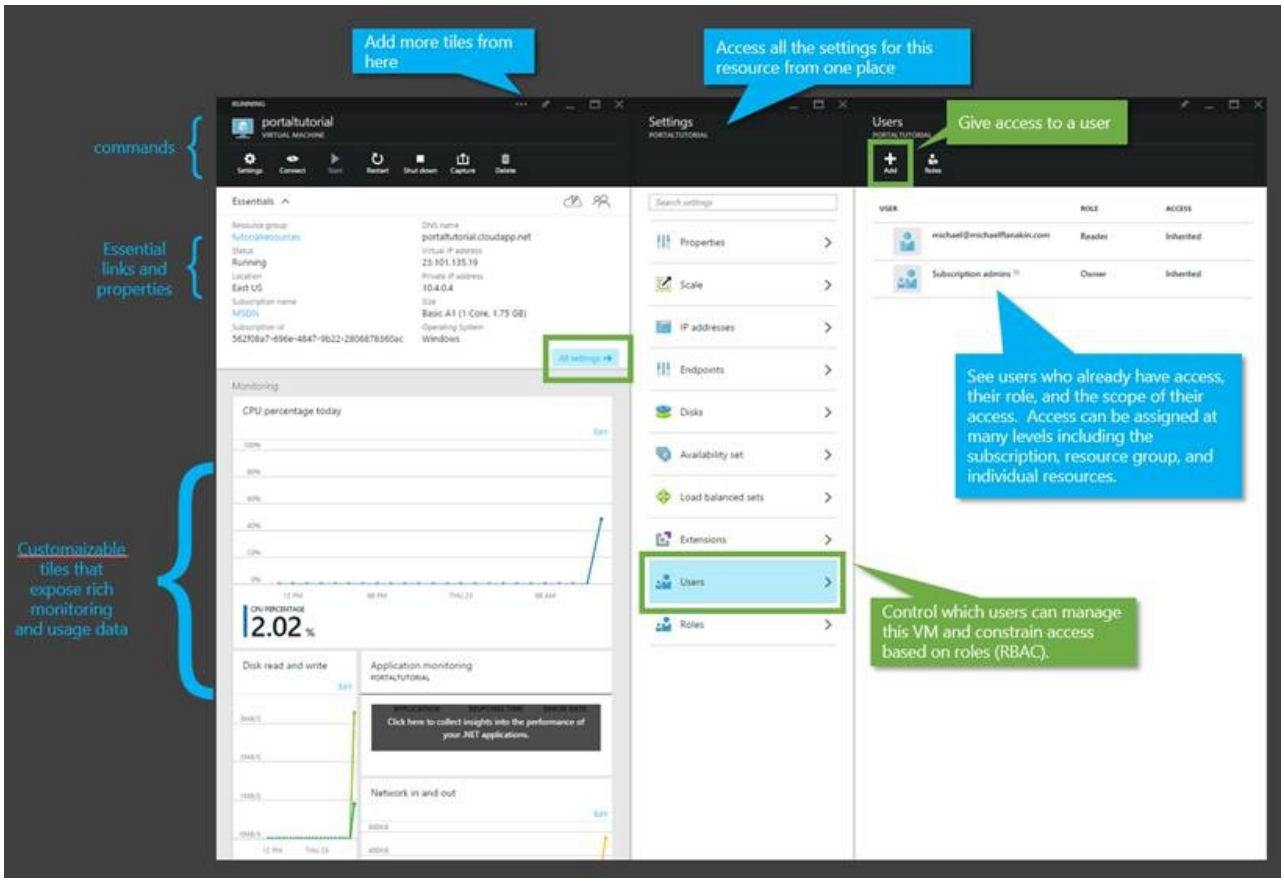
You can always pin frequently accessed resources to your startboard, but you might need to browse to something that you don't frequently access. The browse hub shown below is your way to get to all of your resources. You can filter by subscription, choose/resize columns, and navigate to the management blades by clicking on individual items.




## How to manage and delegate access to a resource

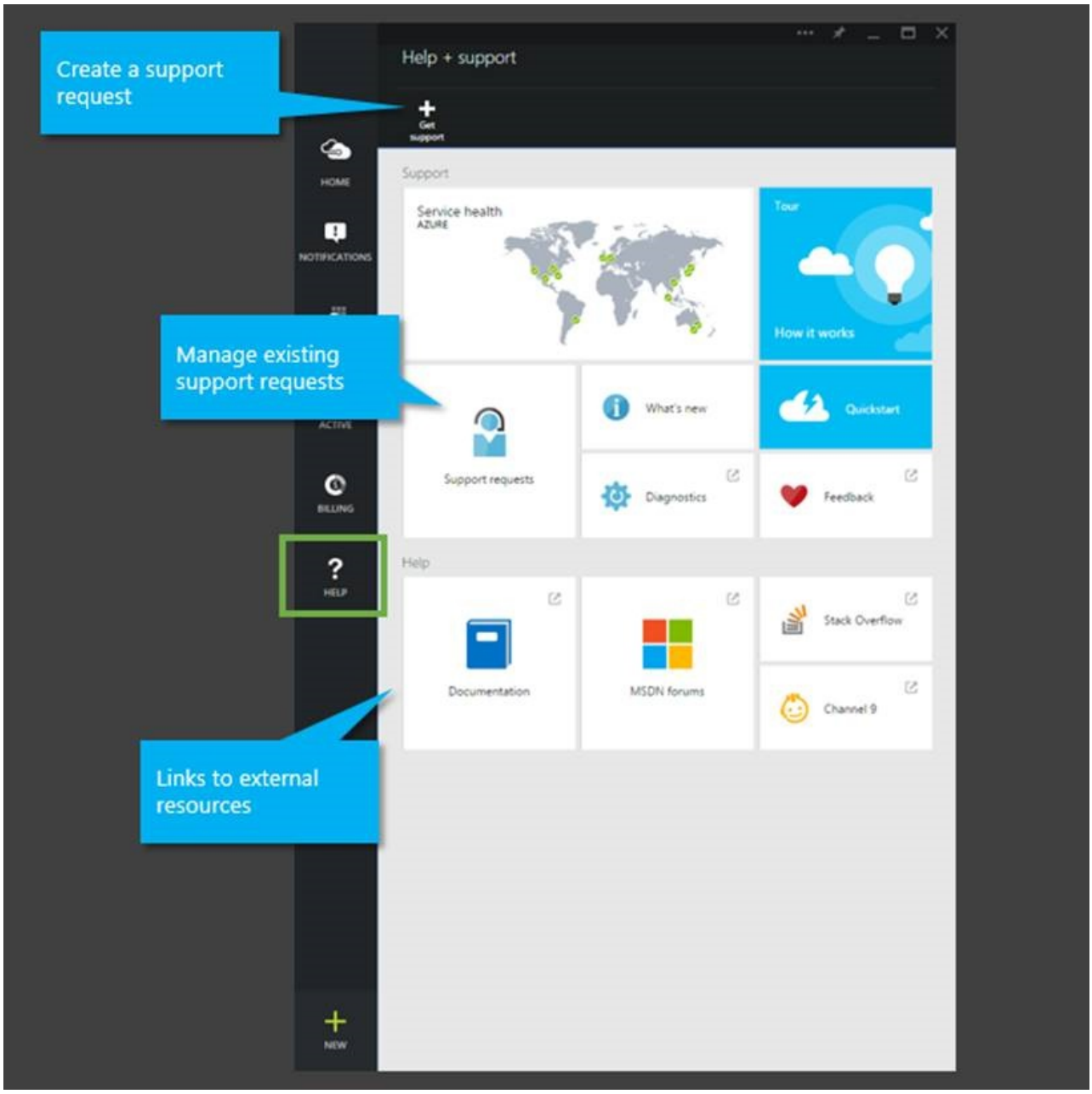
From this blade you can connect to the virtual machine using remote desktop, monitor key performance metrics, control access to this VM using role based access (RBAC), configure the VM, and perform other important management tasks. Delegating access based on role is critical to managing at scale. Click here to learn more about it. To delegate access to a resource, perform the following actions:

1. Browse to your resource.
2. Click ‘All settings’ in the Essentials section.
3. Click ‘Users’ in the settings list.
4. Click ‘Add’ in the command bar.
5. Choose a user and a role.




## How to get help

If you ever have a problem, we're here for you. The portal has a help and support page that can point you in the right direction. Depending on your support plan, you can also create support tickets directly in the portal. After creating a support ticket, you can manage the lifecycle of the ticket from within the portal. You can get to the help and support page by navigating to Browse -> Help + support.




## Summary

Let's review what you learned in this tutorial:

- You learned how to sign up, get a subscription, and browse to the portal
- You got oriented with the portal UI and learned how to create and browse resources
- You learned how to create a resource and browse resources
- You learned about the structure or management blades and how you can consistently manage different types of resources
- You learned how to customize the portal to bring the information you care about to the front and center
- You learned how to control access to resources using role based access (RBAC)
- You learned how to get help and support

The Microsoft Azure portal radically simplifies building and managing your applications in the cloud. Take a look at the management blog to keep up to date as we're constantly listening to feedback and making improvements. ScottGu's blog is another great place to look for all Azure updates.

# Tutorial: DevOps with the Azure Portal

12/11/2017 • 9 min to read • Edit Online

The Azure platform is full of flexible DevOps workflows. In this tutorial, you learn how to leverage the capabilities of the Azure Portal to develop, test, deploy, troubleshoot, monitor, and manage running applications. This tutorial focuses on the following:

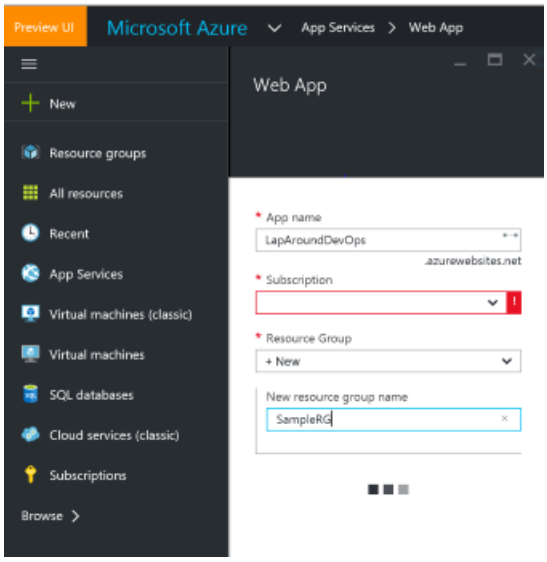
1. Creating a web app and enabling continuous deployment
2. Develop and test an app
3. Monitoring and Troubleshooting an app
4. General application management tasks

## Creating a web app and enabling continuous deployment

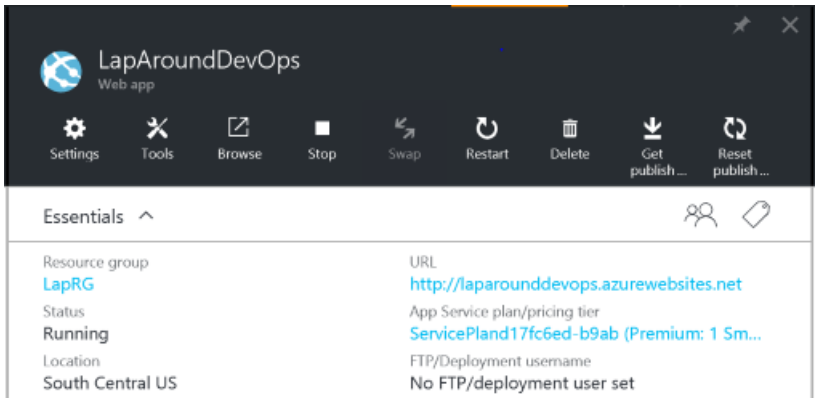
Create a Web app with Azure App Service, which you'll use in the rest of this tutorial. You'll initially enable continuous deployment from your source code repository into our running Azure environment.

1. Sign into the Azure Portal
2. Choose **App Services** > **Add icon** and enter a name, choose your subscription, and create a new resource group to serve as the container for the service.

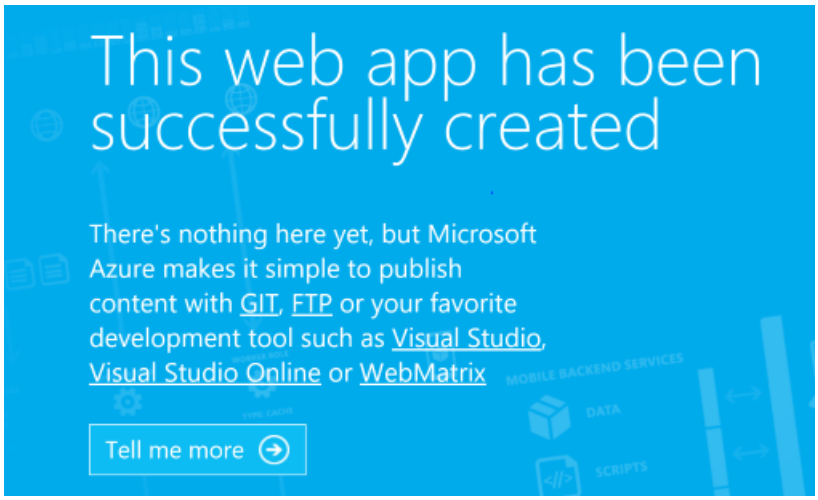
   Resource groups allow you to manage various aspects of the solution such as billing, deployments and monitoring all as a single group via Azure Resource Manager.



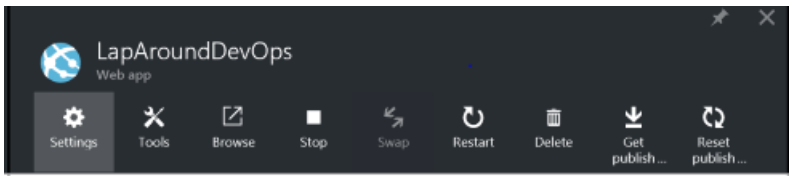

3. After a few moments, your app service is created. Take a few minutes to explore the various menu options for the service in the portal.



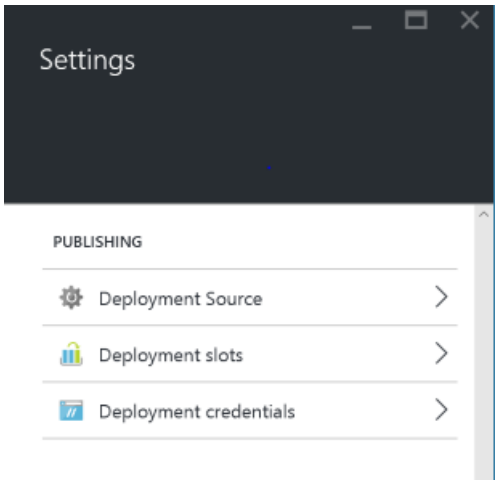

4. Click the URL. Notice the variety of available choices for tools and repositories. You can also use the languages and frameworks of your choice including .NET, Java, and Ruby.



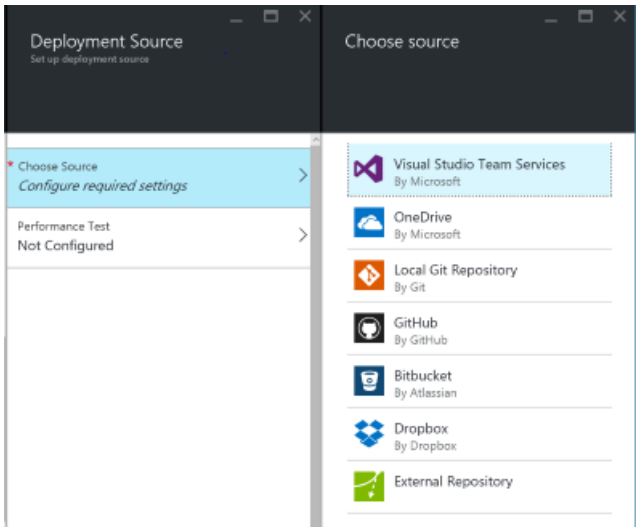

5. The Azure portal makes continuous deployment an easy process that involves only a few simple steps. In the Azure portal, choose settings from the icon for the app service you just created.



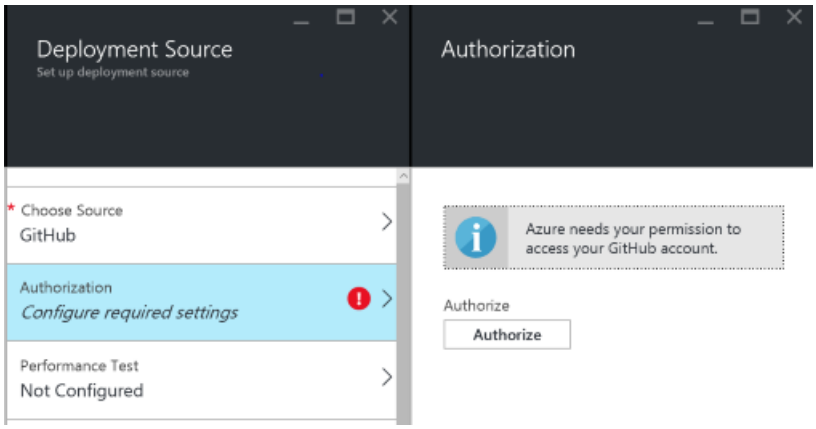

From the blade that opens on the right, scroll to the publishing section.



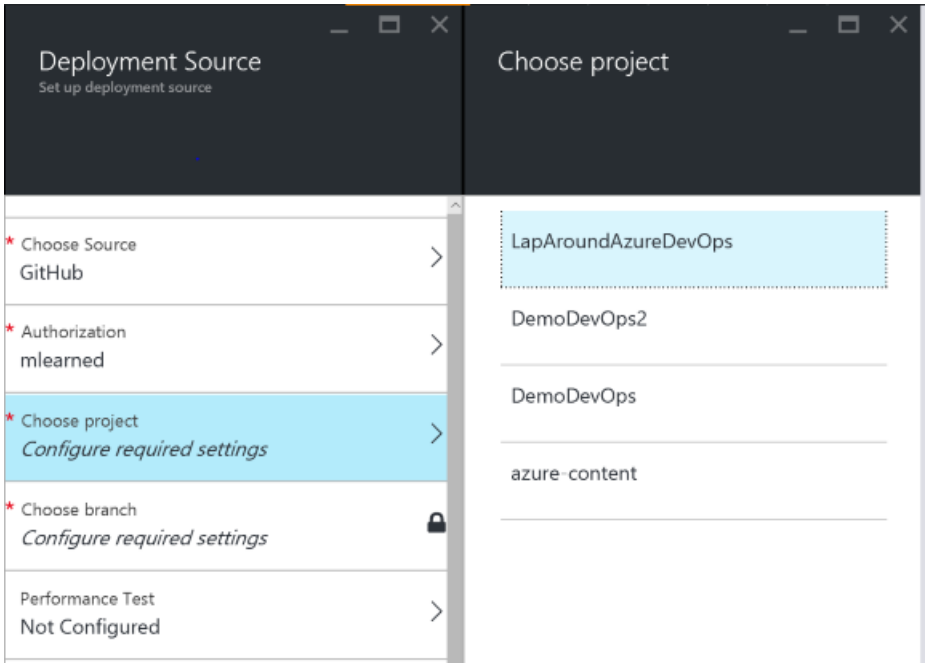

6. Next, configure some settings to enable continuous deployment for the app. Click Deployment Source and then click Choose Source. Notice the variety of options you have for repository sources.




7. For this example choose GitHub. Optionally choose the repository of your choice and setup the authorization credentials.

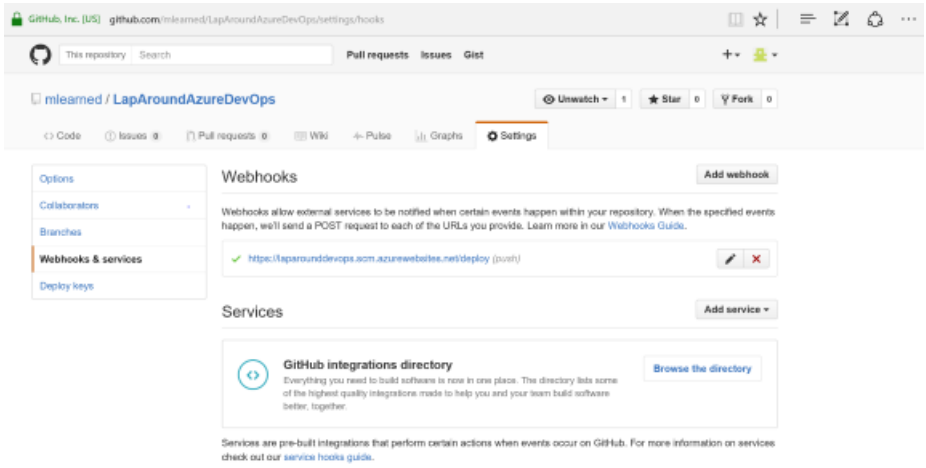



8. After authorization to your repository, you can then choose a project and branch you wish to deploy. There are several fictitious sample examples listed below.

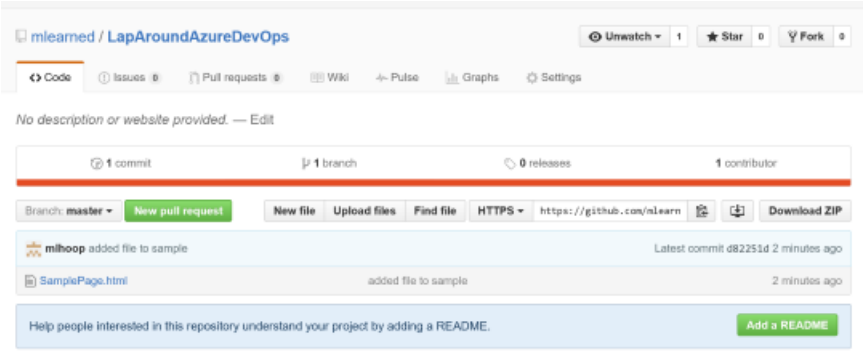



9. Once you choose your project and branch, click ok. You should start to see notifications of a deployment.

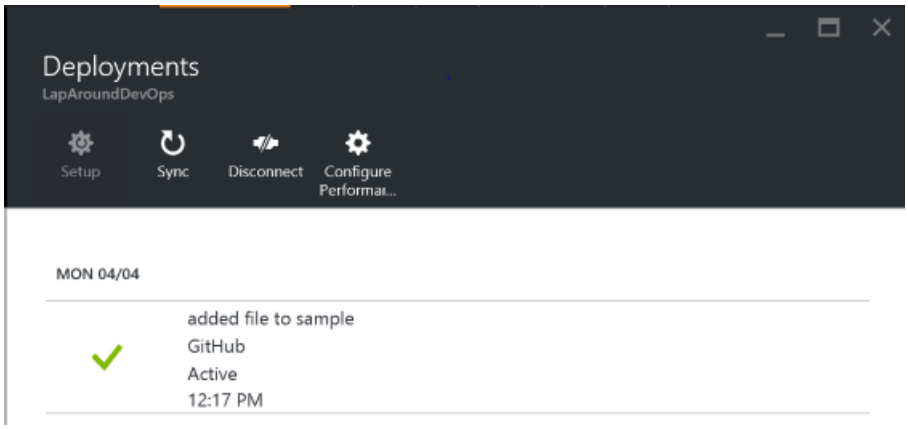

10. Navigate back to GitHub to see the webhook that was created to integrate the source control repo with Azure. The Azure Portal enables integration with GitHub with only a few simple steps.




11. To demonstrate continuous deployment, you quickly add some content to the repository. For a simple example, add a sample text file to a GitHub repo. You are free to use .NET, Ruby, Python, or some other type of application with App Service. Feel free to add a text file, ASP.NET MVC, Java, or Ruby application to the repo of your choice.

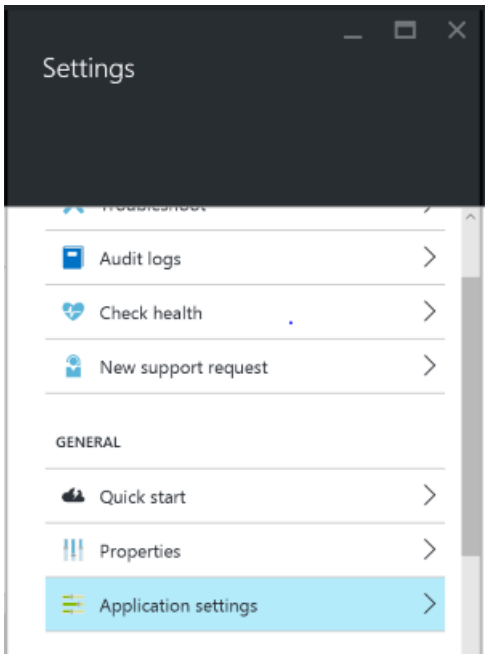



12. After committing changes to your repository, you see a new deployment initiate in the portal notifications area. Click Sync if you do not quickly see changes after committing to your repository.

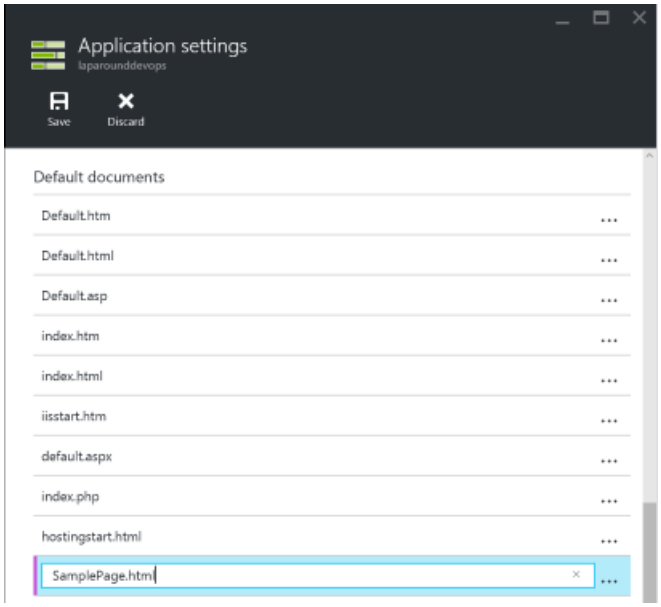



13. At this point, if you try and load the page for the app service, you may receive a 403 error. In this example, it is because there is no typical default document setup for the page such as a file like index.htm or

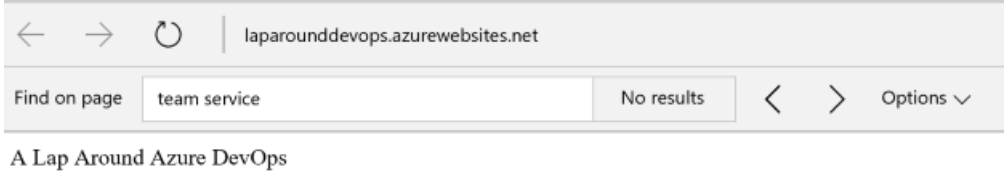
default.html. You can quickly remedy this with the tooling in the Azure Portal. In the Azure Portal choose Settings > Application Settings.




14. A blade opens for application settings. Enter the name of the page "SamplePage.html" and click Save. Take a few minutes to explore the other settings.




15. Optionally refresh your browser URL to ensure you see the expected changes. In this case, there is some simple text now populating the page. Each additional change to the repository would result in a new automatic deployment.

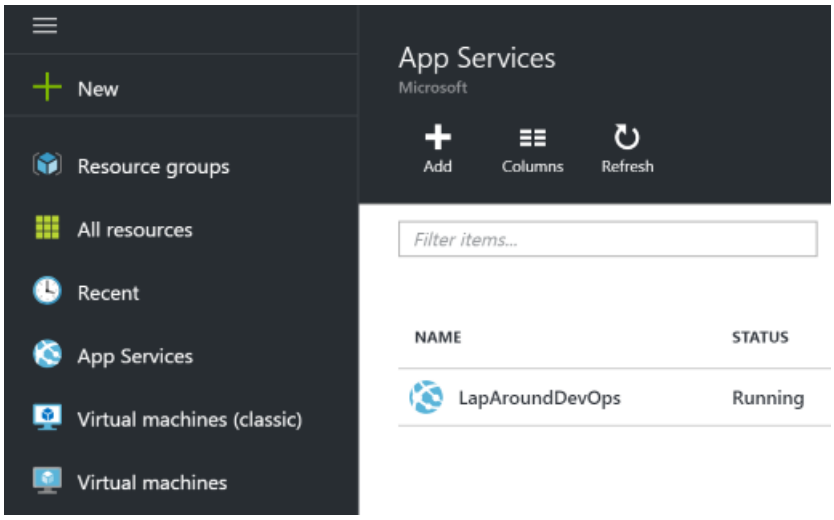



Enabling continuous deployment with the Azure Portal is an easy experience. You can also build more complex release pipelines and use many other techniques with existing source control and continuous integration systems to deploy to Azure, such as leveraging automated build and release management systems.

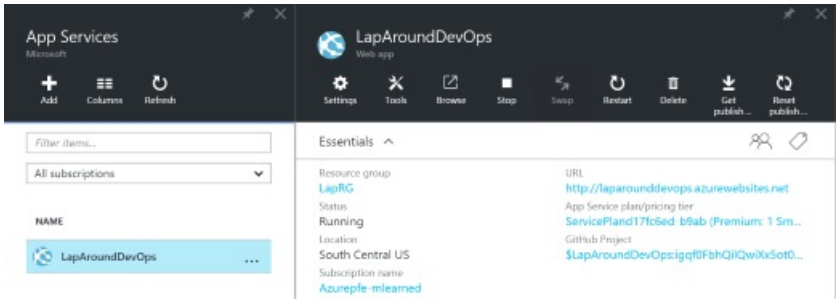
# Develop and test an app

Next, make some changes to the code base and rapidly deploy those changes. You will also setup up some performance testing for the Web app.

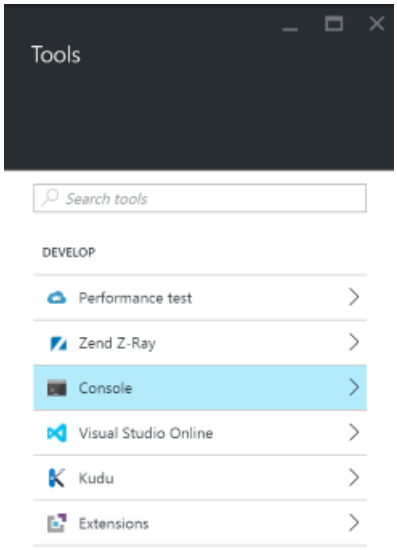
1. In the Azure Portal choose App Services from the navigation pane, and locate your App Service.



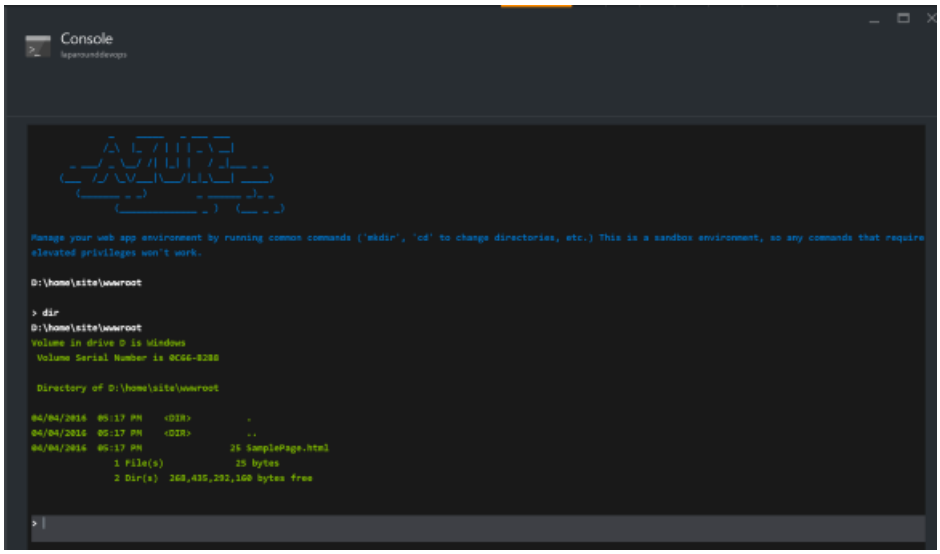

2. Click Tools



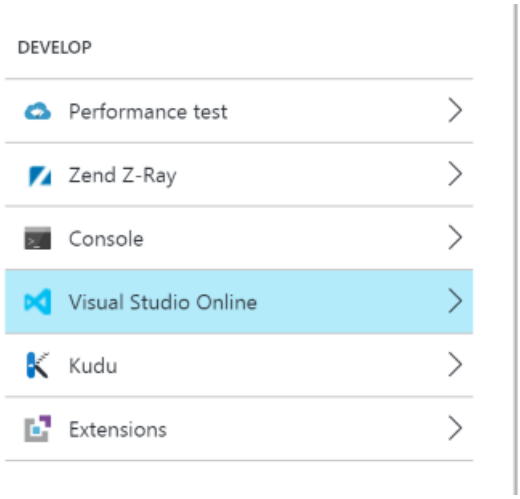

3. Notice the develop category under Tools. There are several useful tools here that allow us to work with apps without leaving the Azure Portal. Click on Console.



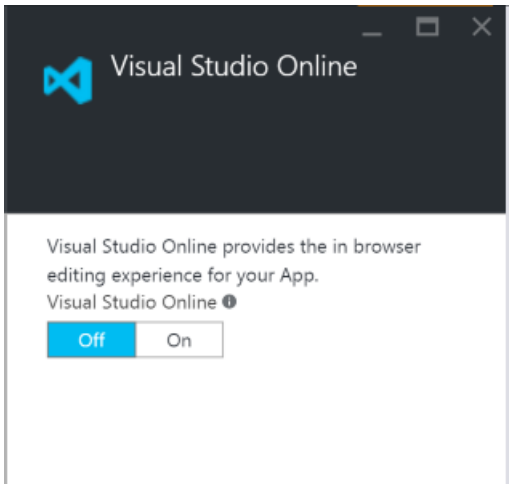

4. In the console window, you can issue live commands for your app. Type the dir command and hit enter. Note that commands requiring elevated privileges do not work.



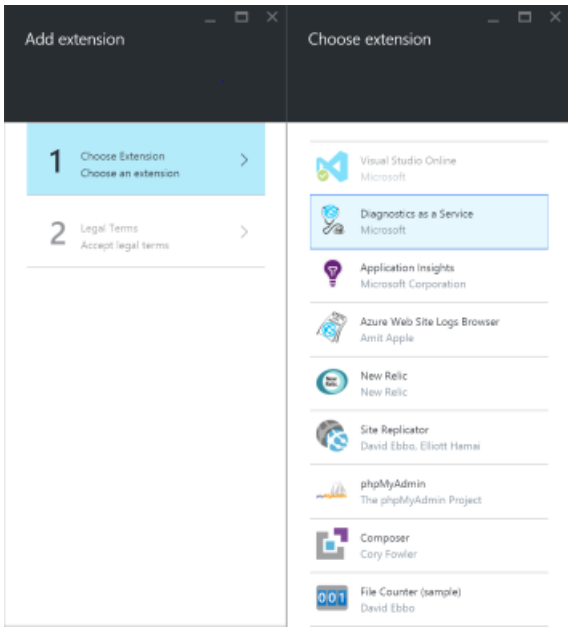

5. Move back to the Develop category and choose Visual Studio Online. Note: Visual Studio Online is now named Visual Studio Team Services.



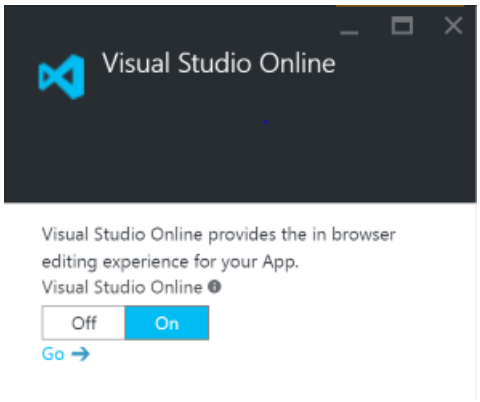

6. Toggle on the in-browser editing experience for your App.



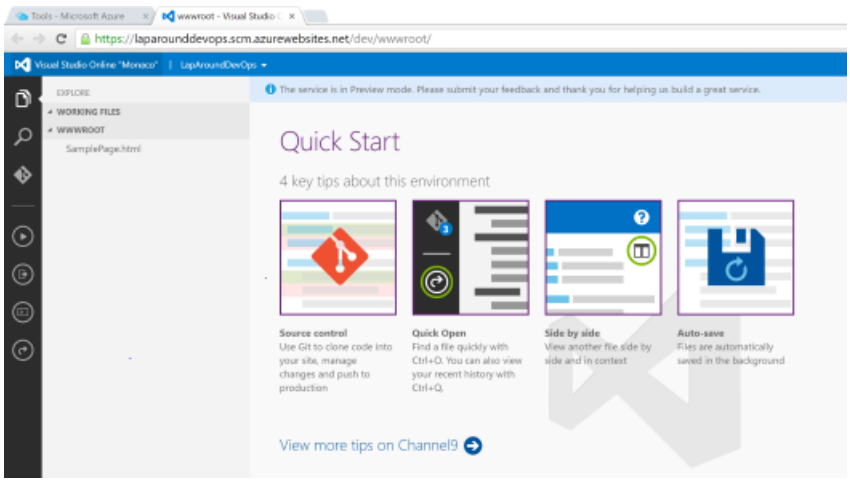

7. A web extension installs for your app. Extensions quickly and easily add functionality to apps in Azure. Notice some of the other extension types available in the screenshot below.



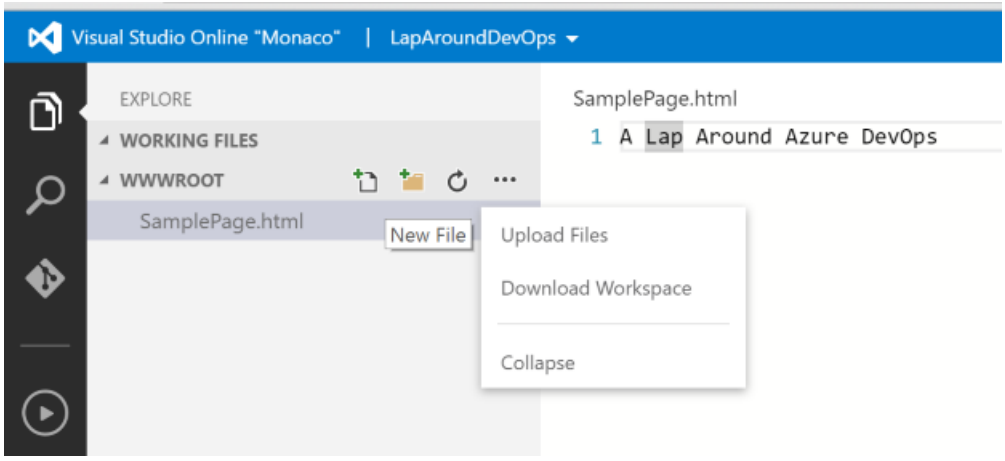

8. Once the Visual Studio Online extension installs, click Go.



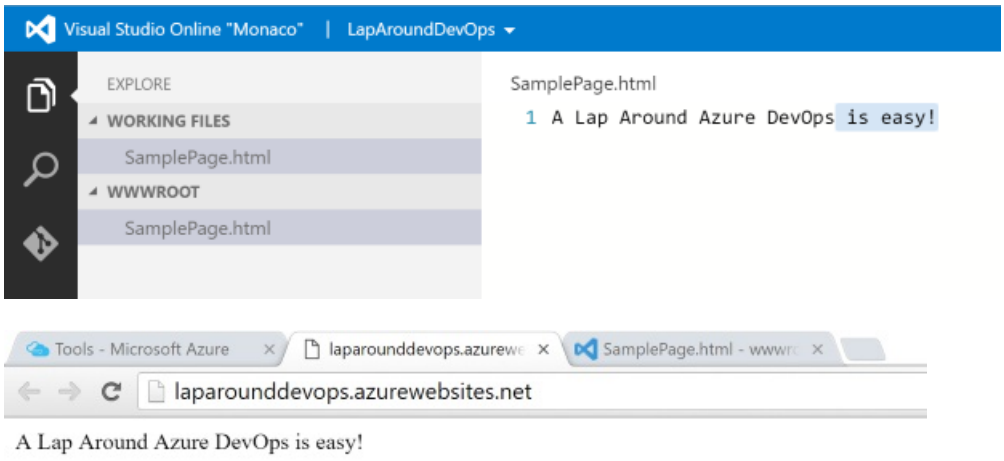

9. A browser tab opens where you see a development IDE directly in the browser. Notice the experience below is in Chrome.



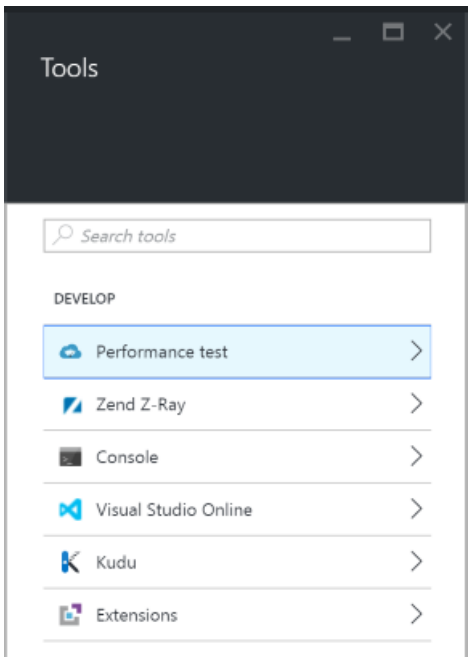

10. You can perform several activities such as edit files, add files and folders, and download content from the live site. Make a quick edit to the SamplePage.html file.



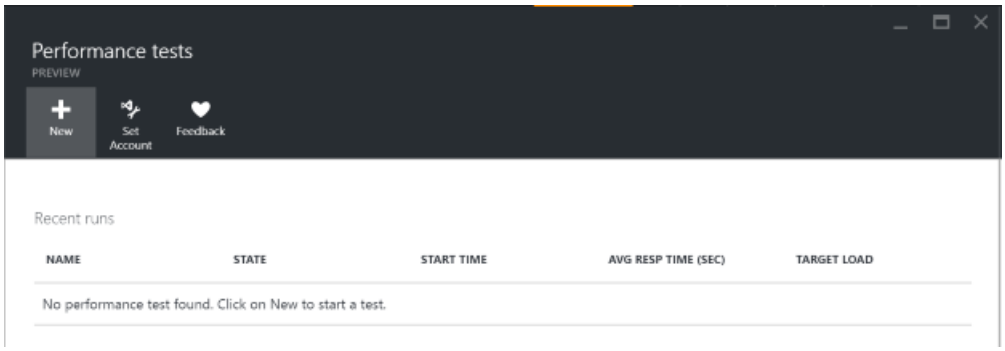

11. In a few moments, the changes are automatically saved. If you navigate back to the page, you can see the changes. Keep in mind live edits like these are most likely not suitable for production environments. However, the tools make it very easy to make quick changes for dev and test environments.



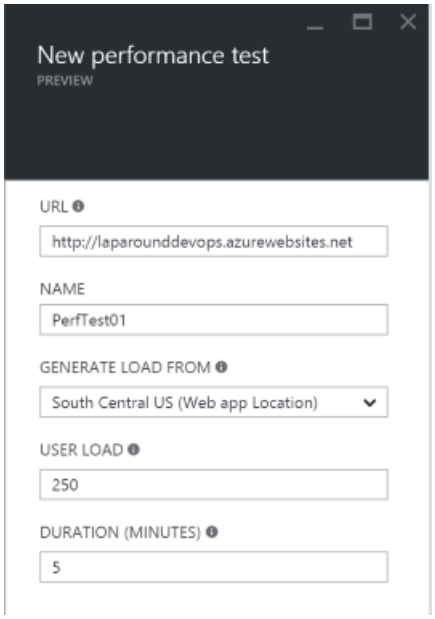

12. Move back to the tools blade and under the Develop category, click on Performance Test.




13. You need to set a team services account. See here for more details: Create a Team Services Account
14. Click on New to create a performance test.

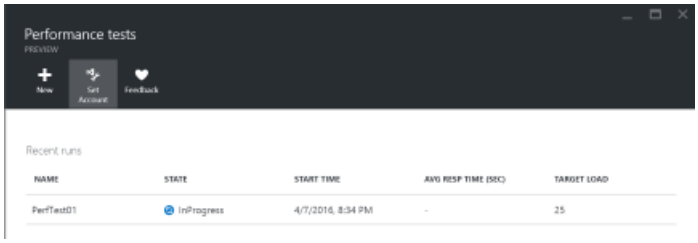



Configure the various values and click Run Test at the bottom of the dialogue to initiate a performance test.

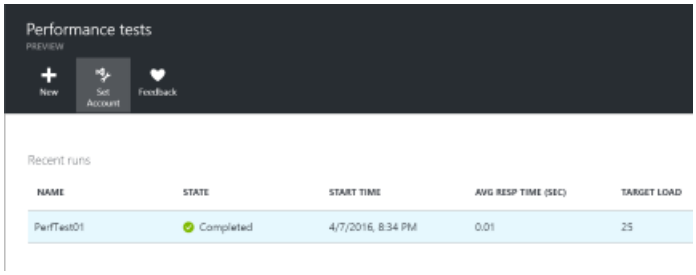




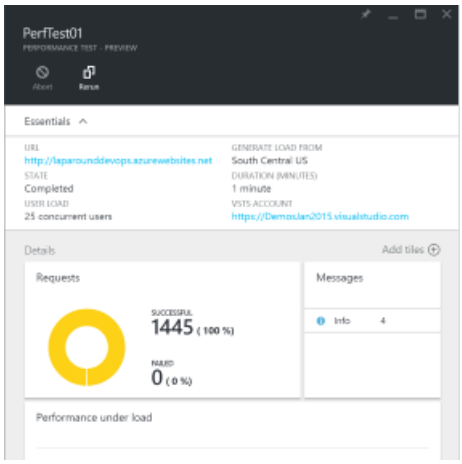
15. Once the test starts running, you can monitor the state.

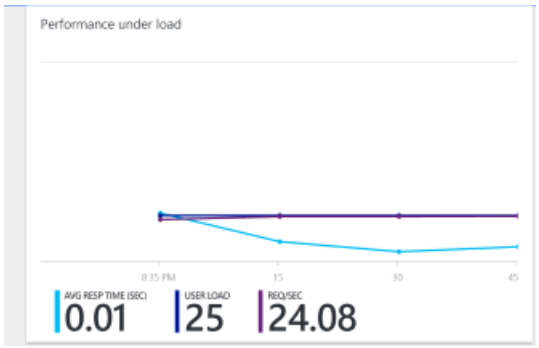


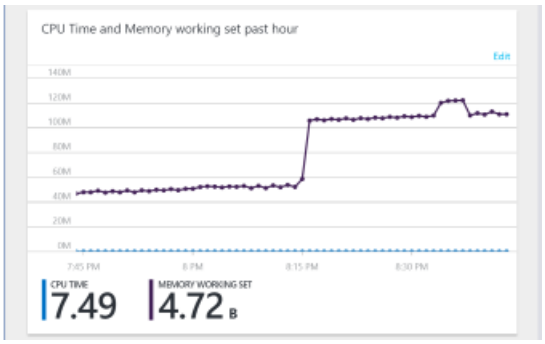

Once the test finishes, clicking on the result shows more details.




16. In this example, you created a small test run, so there is limited data to analyze, but you can see various metrics as well as rerun your test from this view. The Azure Portal makes creating, executing, and analyzing web performance tests an easy process. The screenshots below display the performance data.



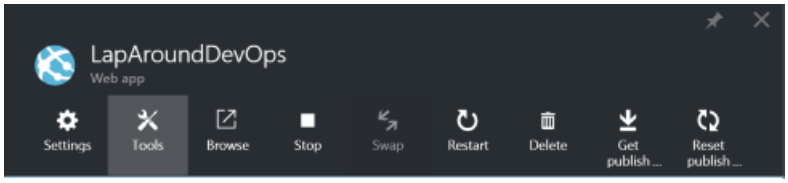



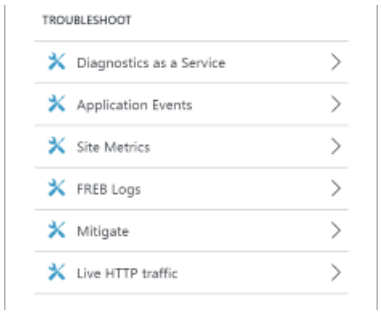

## Monitoring and troubleshooting an app

Azure provides many capabilities for monitoring and troubleshooting running applications.

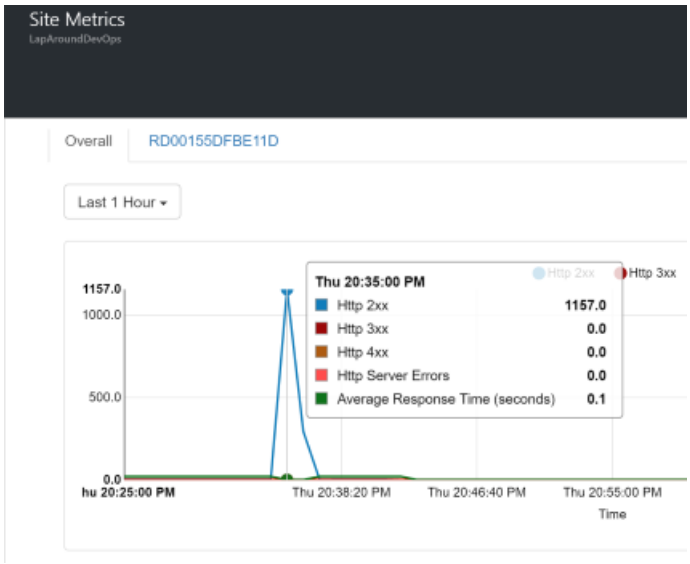
1. In the Azure Portal for our Web app choose Tools.



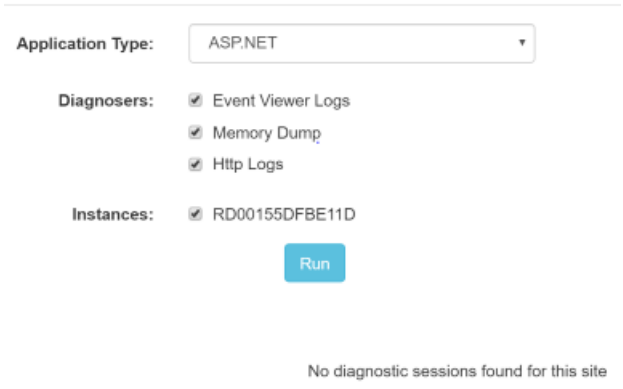

2. Under the Troubleshoot category, notice the various choices for using tools to troubleshoot potential issues with a running app. You can do things like monitor Live HTTP traffic, enable self-healing, view logs, and more.



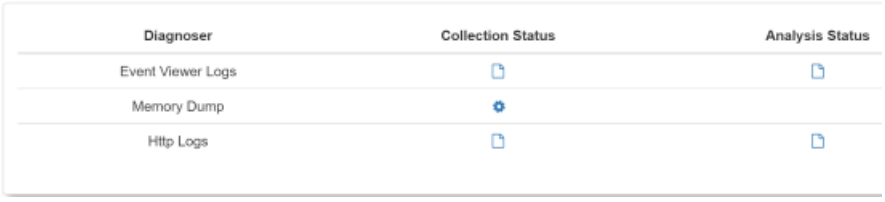

3. Choose Site Metrics to quickly get a view of some HTTP codes.



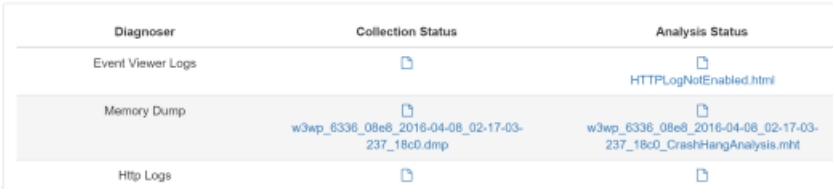

4. Choose Diagnostics as a Service. Choose your application type, then choose Run.



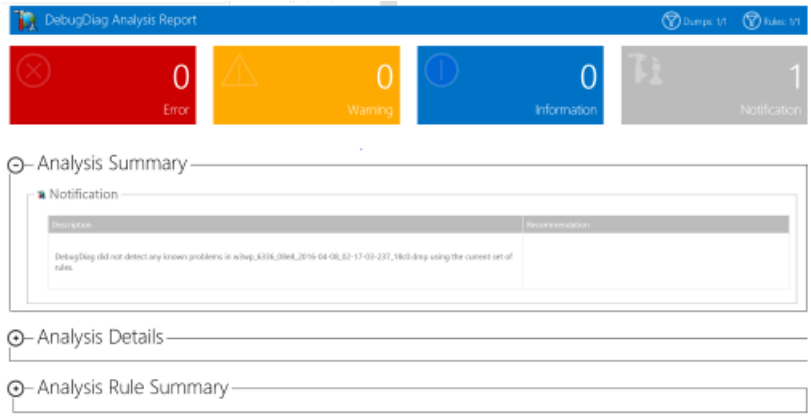

A collection begins.



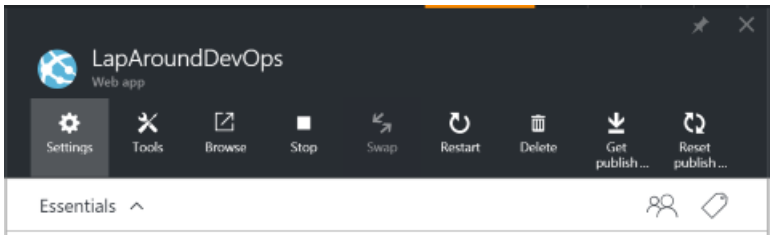

5. You may choose the appropriate log to diagnose potential issues. You need to enable logging to see all of the available data options such as HTTP Logs.



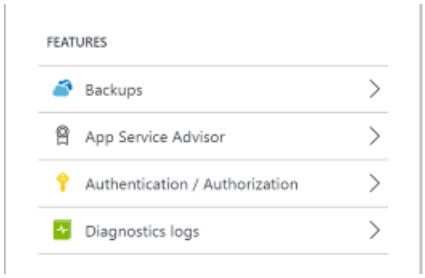
By clicking on the Memory Dump file you can download and analyze a DebugDiag analysis report to help find potential issues.



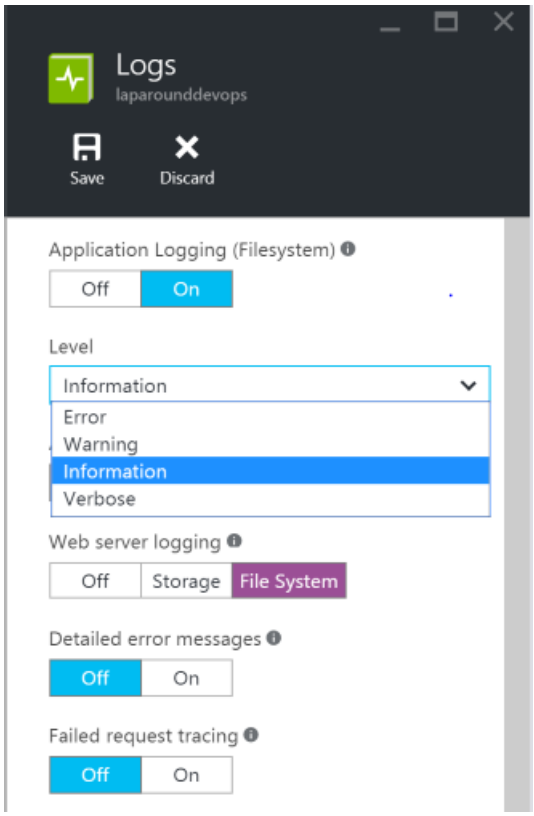

6. To view more data, you need to enable additional logging. In the Azure Portal, navigate to the Web app and choose Settings.



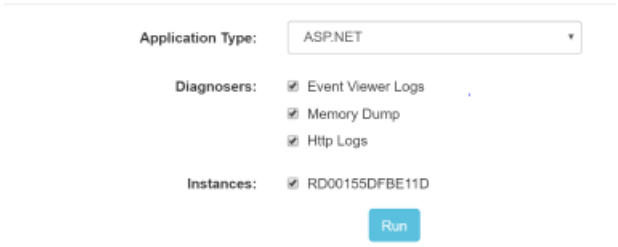

7. Scroll down to the features category, and choose Diagnostic logs.



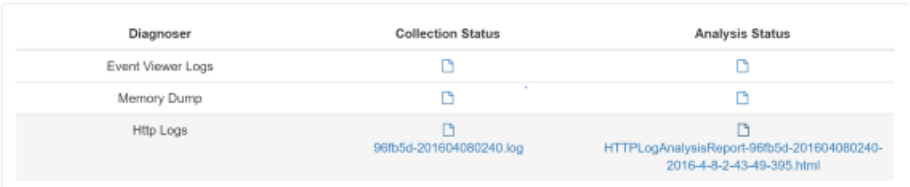

8. Notice the various options for logging. Toggle on Web server logging and click save.




9. Move back to the tools area for the app and choose Diagnostics as a service and click Run to rerun the data collection.

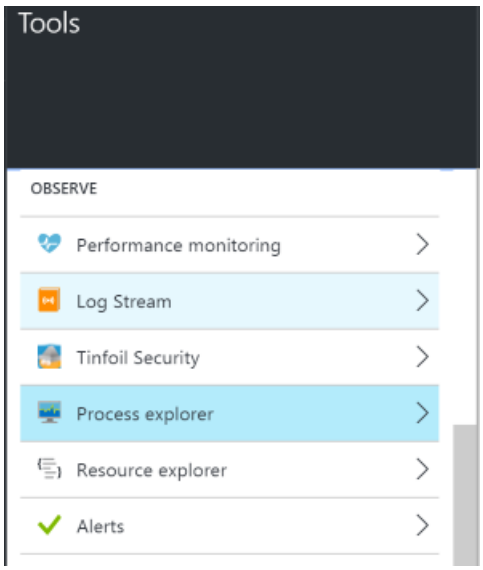



10. With the HTTP logging setting enabled, you now see data collected for HTTP Logs.

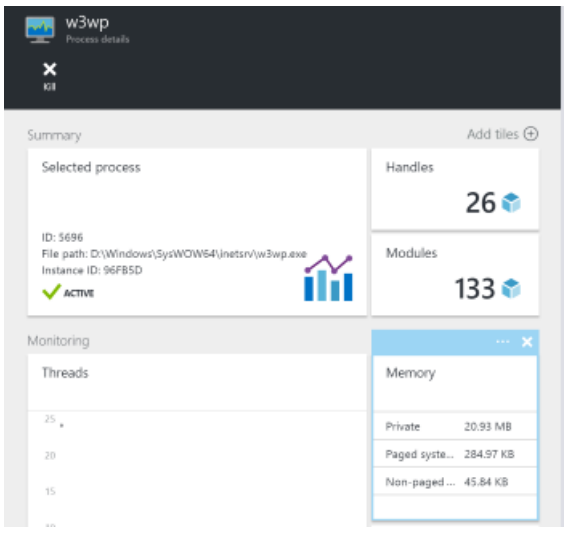



11. By clicking the HTML file log, you produce a rich browser-based report for further investigation.

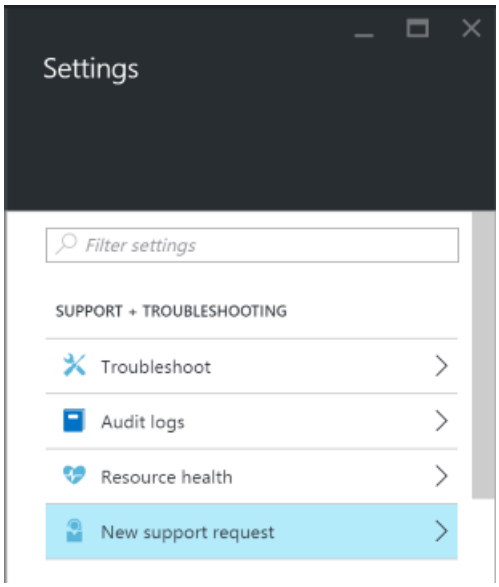
12. Move back to the tools section in the Azure Portal for the app. Scroll to the Tools section and choose Process Explorer.



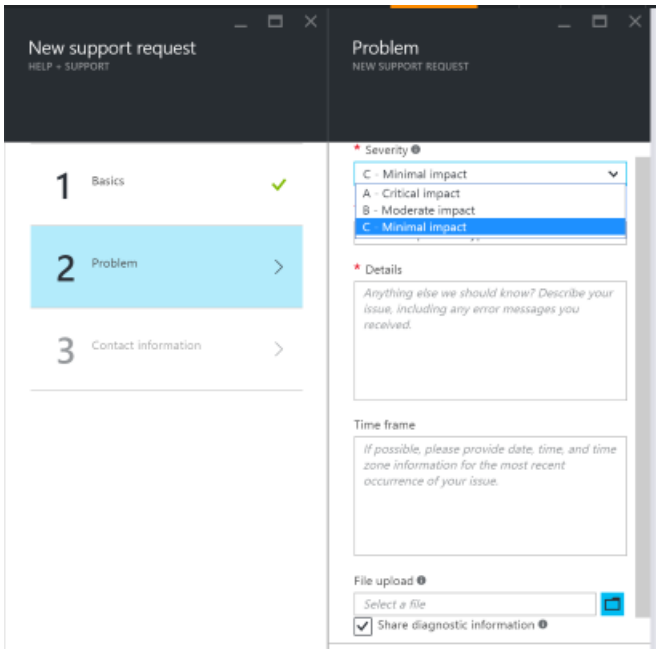

13. By choosing Process Explorer, you can view details about running processes. Notice below you can drill into processes and even kill processes all from the Azure Portal.

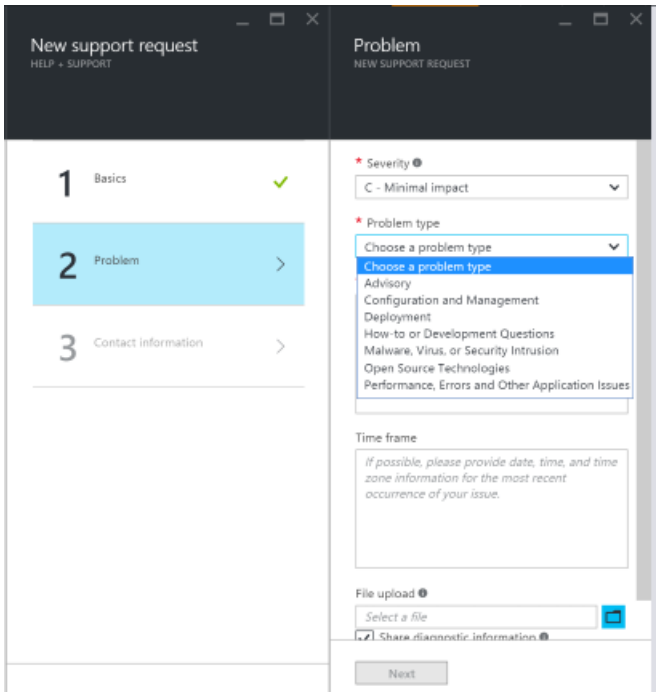




14. Move back to the Settings blade on the left. Click New support request.




15. From the blade on the right, you can fill out details about the issues, enter contact information, and even upload diagnostic data. The Azure Portal enables working with Microsoft support a seamless experience.



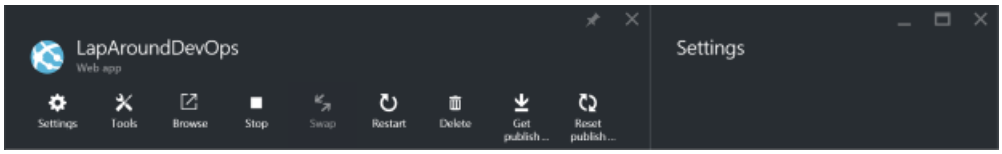



The Azure Portal helps provide powerful and familiar tooling experiences to help monitor and troubleshoot our running applications. You are also able to take action quickly by performing tasks such as recycling processes, enabling and disabling various data collections, and even integrating with Microsoft professional support.

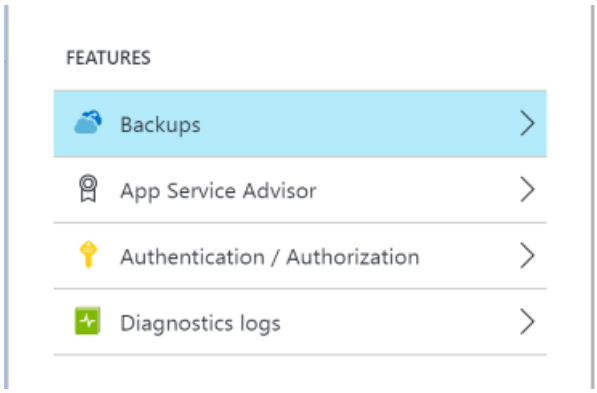
## General Application Management

When managing applications, you often need to perform a broad variety of activities such as configuring backup strategies, implementing and managing identity providers, and configuring Role-based access control. As with the other DevOps experiences, the Azure platform integrates these tasks directly into the portal.

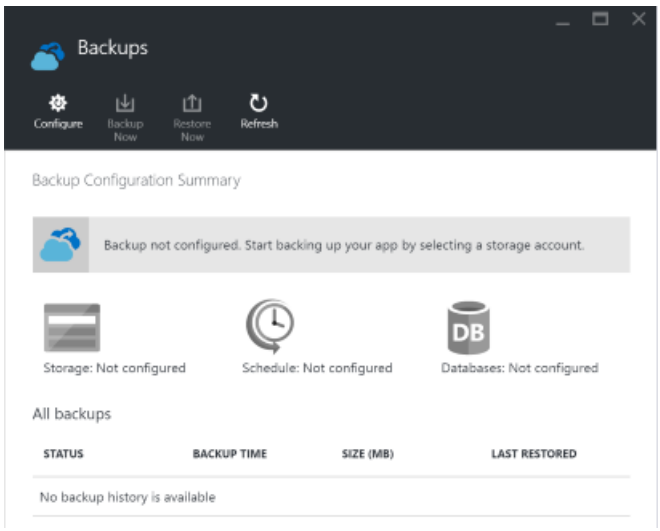
1. To ensure you are keeping the Web App safe from data loss you need to configure backups. Navigate to the Settings area for your Web app.



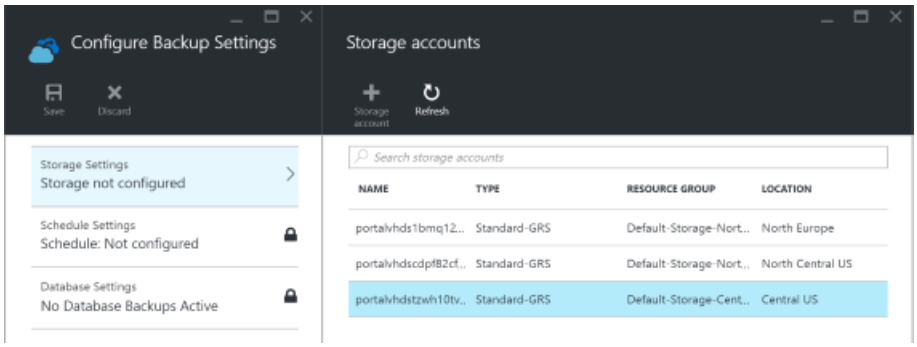

2. In the blade on the right, scroll down to the Features category.



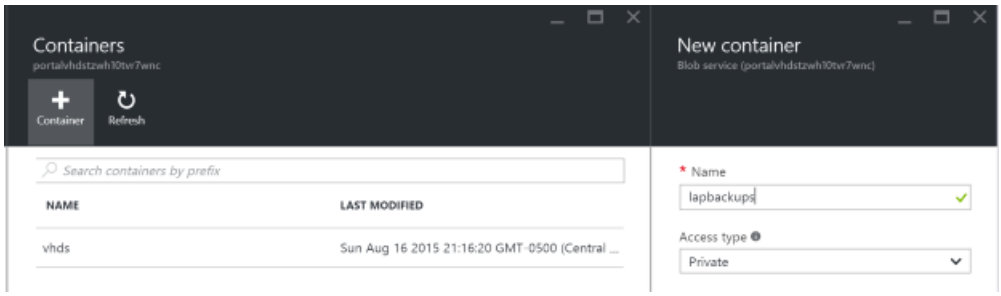

3. Choose Backups; a blade opens on the right.



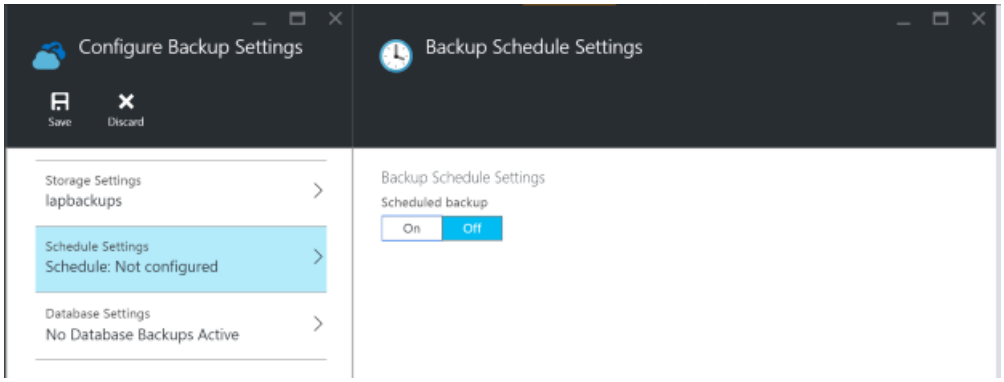

4. Click Configure, choose a storage account from the blade on the right.



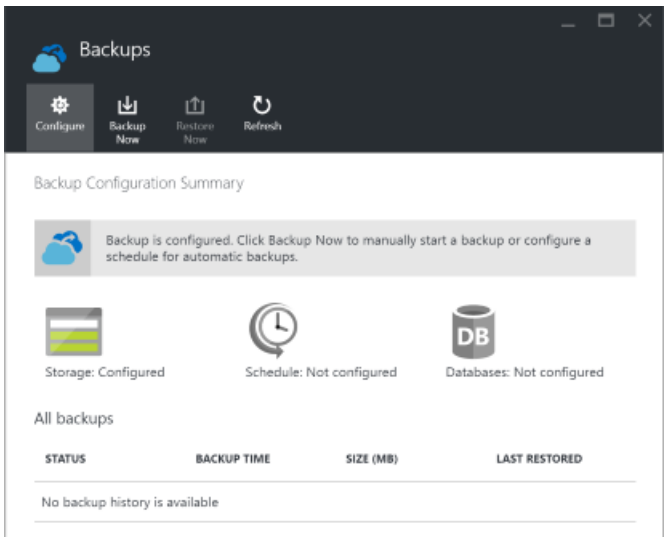

5. Now create and choose a storage container to hold your backups. Click create at the bottom of the blade. Then select the container.



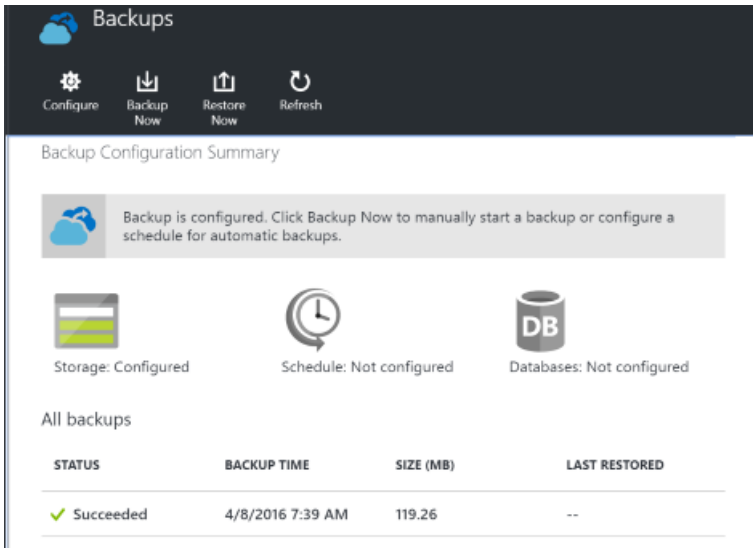

6. Once you have chosen the container, you can configure schedules, as well as setup backups for your databases. For this scenario, click the save icon.



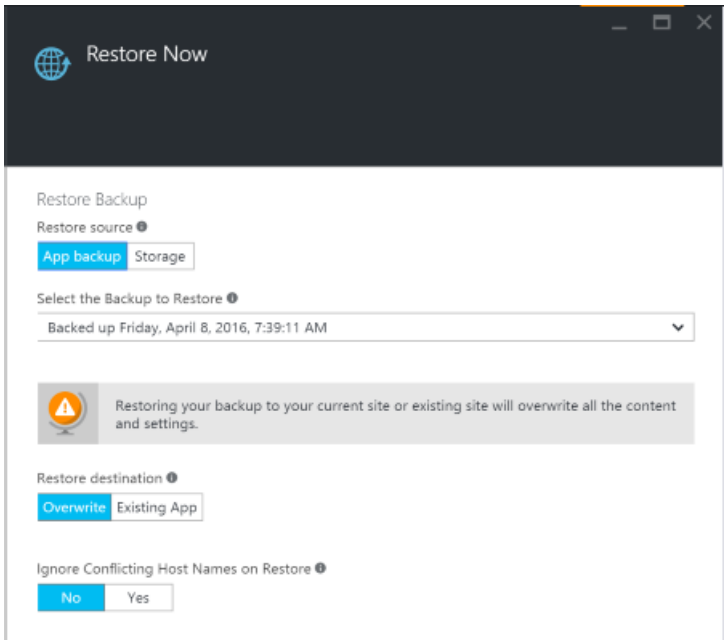

7. After saving, scroll back to the blade on the left for Backups. Click Backup Now to back the application.



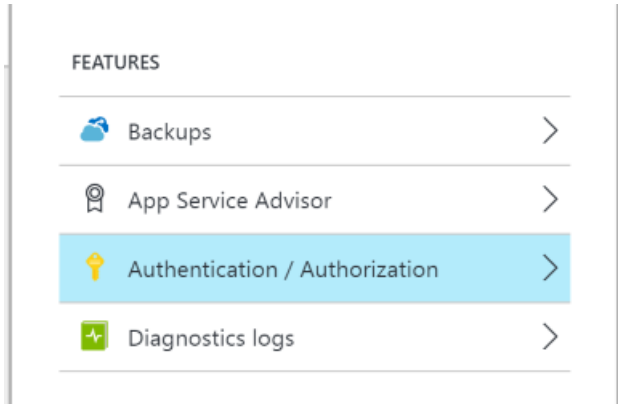

8. In a few moments, you see a backup created. Notice the Restore Now option on the screen-shot below.



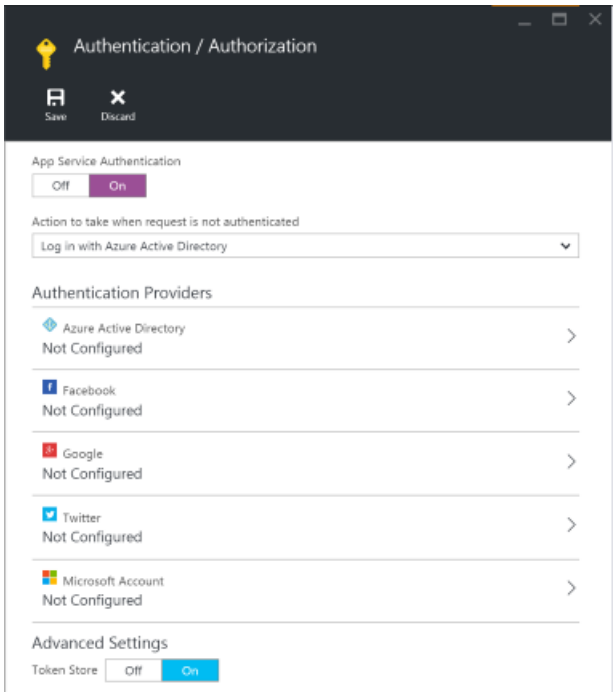

9. Click on Restore Now and examine the options to the blade on the right. You can choose an appropriate backup and easily restore to an earlier state as necessary. The Azure portal has helped us easily enable a simple disaster recovery strategy for the app.



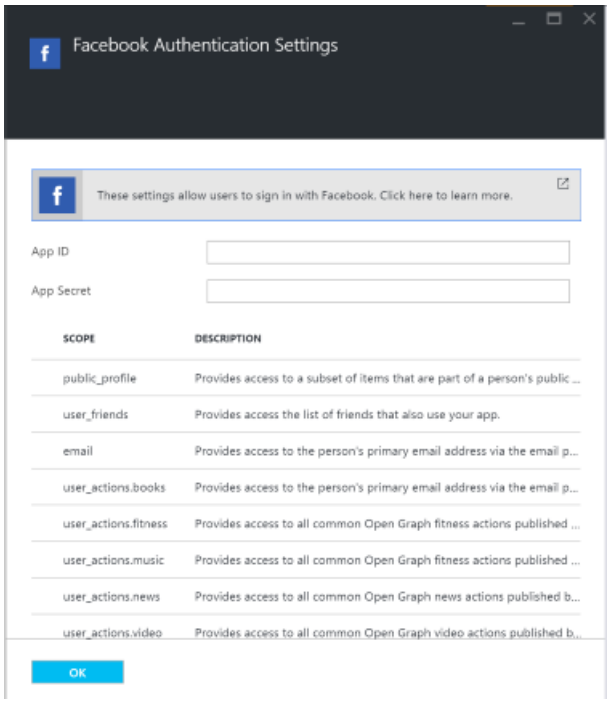

10. Move back to the Settings blade on the left, and under Features and choose Authentication/Authorization.



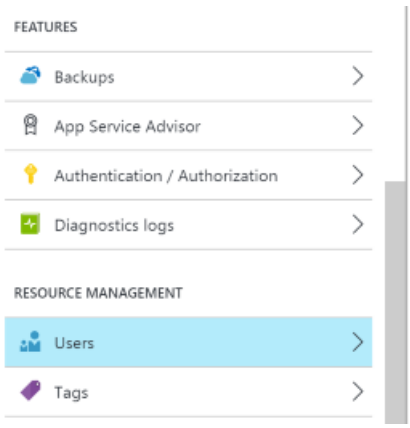

11. In the blade on the right choose App Service Authentication. Notice the variety of options you can configure with popular providers.



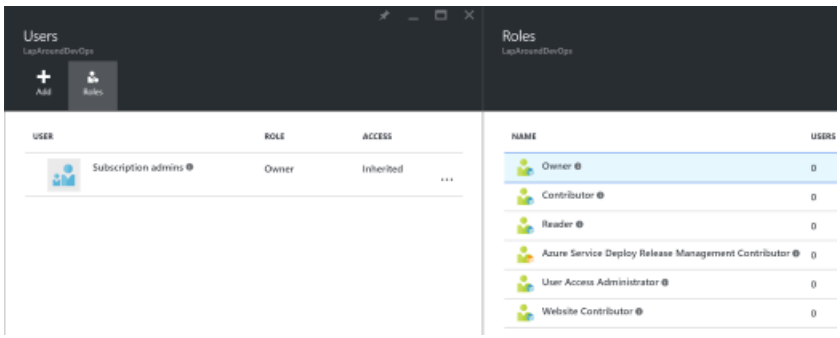

12. Choose the provider of your choice and notice the options for the scope. You can provide an App ID and App Secret and quickly enable Facebook authentication for the app. The Azure Portal enables authentication as a turnkey solution for apps.




13. Move back to the Settings blade and choose Users under the Resource Management category.



14. In the blade on the right examine the various options for adding roles and users. The Azure Portal lets you easily control RBAC (Role-based access control) for the application.




## Summary

This tutorial demonstrated some of the power with the Azure platform by quickly enabling continuous deployment for a web app, performing various development and testing activities, monitoring and troubleshooting a live app, and finally managing key strategies such as disaster recovery, identity, and role-based access control. The Azure platform enables an integrated experience for these DevOps workflows, and you can work efficiently by staying in context for the task at hand.

## Next steps

- Azure Resource Manager is important for enabling DevOps on the Azure platform. To learn more visit Azure Resource Manager overview.
- To learn more about Azure App Service deployment visit Deploy your app to Azure App Service

# Azure Resource Manager overview

11/15/2017 • 15 min to read • Edit Online

The infrastructure for your application is typically made up of many components – maybe a virtual machine, storage account, and virtual network, or a web app, database, database server, and 3rd party services. You do not see these components as separate entities, instead you see them as related and interdependent parts of a single entity. You want to deploy, manage, and monitor them as a group. Azure Resource Manager enables you to work with the resources in your solution as a group. You can deploy, update, or delete all the resources for your solution in a single, coordinated operation. You use a template for deployment and that template can work for different environments such as testing, staging, and production. Resource Manager provides security, auditing, and tagging features to help you manage your resources after deployment.

## Terminology

If you are new to Azure Resource Manager, there are some terms you might not be familiar with.

- **resource** - A manageable item that is available through Azure. Some common resources are a virtual machine, storage account, web app, database, and virtual network, but there are many more.
- **resource group** - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. See Resource groups.
- **resource provider** - A service that supplies the resources you can deploy and manage through Resource Manager. Each resource provider offers operations for working with the resources that are deployed. Some common resource providers are Microsoft.Compute, which supplies the virtual machine resource, Microsoft.Storage, which supplies the storage account resource, and Microsoft.Web, which supplies resources related to web apps. See Resource providers.
- **Resource Manager template** - A JavaScript Object Notation (JSON) file that defines one or more resources to deploy to a resource group. It also defines the dependencies between the deployed resources. The template can be used to deploy the resources consistently and repeatedly. See Template deployment.
- **declarative syntax** - Syntax that lets you state "Here is what I intend to create" without having to write the sequence of programming commands to create it. The Resource Manager template is an example of declarative syntax. In the file, you define the properties for the infrastructure to deploy to Azure.

## The benefits of using Resource Manager

Resource Manager provides several benefits:

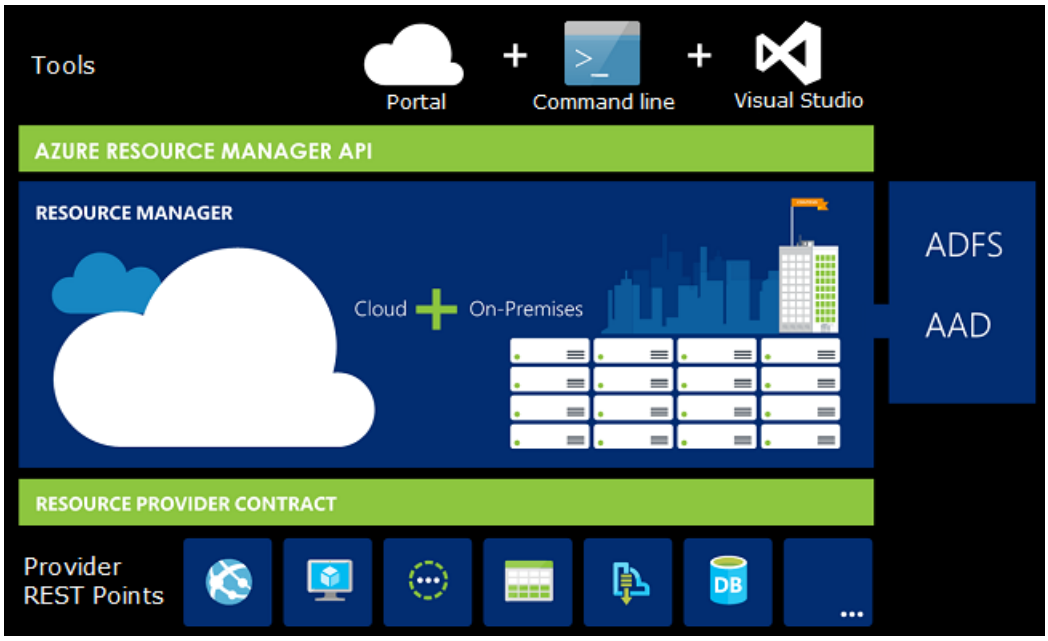
- You can deploy, manage, and monitor all the resources for your solution as a group, rather than handling these resources individually.
- You can repeatedly deploy your solution throughout the development lifecycle and have confidence your resources are deployed in a consistent state.
- You can manage your infrastructure through declarative templates rather than scripts.
- You can define the dependencies between resources so they are deployed in the correct order.
- You can apply access control to all services in your resource group because Role-Based Access Control (RBAC) is natively integrated into the management platform.
- You can apply tags to resources to logically organize all the resources in your subscription.
- You can clarify your organization's billing by viewing costs for a group of resources sharing the same tag.

Resource Manager provides a new way to deploy and manage your solutions. If you used the earlier deployment model and want to learn about the changes, see Understanding Resource Manager deployment and classic deployment.

## Consistent management layer

Resource Manager provides a consistent management layer for the tasks you perform through Azure PowerShell, Azure CLI, Azure portal, REST API, and development tools. All the tools use a common set of operations. You use the tools that work best for you, and can use them interchangeably without confusion.

The following image shows how all the tools interact with the same Azure Resource Manager API. The API passes requests to the Resource Manager service, which authenticates and authorizes the requests. Resource Manager then routes the requests to the appropriate resource providers.




## Guidance

The following suggestions help you take full advantage of Resource Manager when working with your solutions.

1. Define and deploy your infrastructure through the declarative syntax in Resource Manager templates, rather than through imperative commands.
2. Define all deployment and configuration steps in the template. You should have no manual steps for setting up your solution.
3. Run imperative commands to manage your resources, such as to start or stop an app or machine.
4. Arrange resources with the same lifecycle in a resource group. Use tags for all other organizing of resources.

For recommendations about templates, see Best practices for creating Azure Resource Manager templates.

For guidance on how enterprises can use Resource Manager to effectively manage subscriptions, see Azure enterprise scaffold - prescriptive subscription governance.

## Resource groups

There are some important factors to consider when defining your resource group:

1. All the resources in your group should share the same lifecycle. You deploy, update, and delete them together. If one resource, such as a database server, needs to exist on a different deployment cycle it should be in another resource group.
2. Each resource can only exist in one resource group.

3. You can add or remove a resource to a resource group at any time.
4. You can move a resource from one resource group to another group. For more information, see Move resources to new resource group or subscription.
5. A resource group can contain resources that reside in different regions.
6. A resource group can be used to scope access control for administrative actions.
7. A resource can interact with resources in other resource groups. This interaction is common when the two resources are related but do not share the same lifecycle (for example, web apps connecting to a database).

When creating a resource group, you need to provide a location for that resource group. You may be wondering, "Why does a resource group need a location? And, if the resources can have different locations than the resource group, why does the resource group location matter at all?" The resource group stores metadata about the resources. Therefore, when you specify a location for the resource group, you are specifying where that metadata is stored. For compliance reasons, you may need to ensure that your data is stored in a particular region.

## Resource providers

Each resource provider offers a set of resources and operations for working with an Azure service. For example, if you want to store keys and secrets, you work with the **Microsoft.KeyVault** resource provider. This resource provider offers a resource type called **vaults** for creating the key vault.

The name of a resource type is in the format: **{resource-provider}/{resource-type}**. For example, the key vault type is **Microsoft.KeyVault/vaults**.

Before getting started with deploying your resources, you should gain an understanding of the available resource providers. Knowing the names of resource providers and resources helps you define resources you want to deploy to Azure. Also, you need to know the valid locations and API versions for each resource type. For more information, see Resource providers and types.

## Template deployment

With Resource Manager, you can create a template (in JSON format) that defines the infrastructure and configuration of your Azure solution. By using a template, you can repeatedly deploy your solution throughout its lifecycle and have confidence your resources are deployed in a consistent state. When you create a solution from the portal, the solution automatically includes a deployment template. You do not have to create your template from scratch because you can start with the template for your solution and customize it to meet your specific needs. You can retrieve a template for an existing resource group by either exporting the current state of the resource group, or viewing the template used for a particular deployment. Viewing the exported template is a helpful way to learn about the template syntax.

To learn about the format of the template and how you construct it, see Create your first Azure Resource Manager template. To view the JSON syntax for resources types, see Define resources in Azure Resource Manager templates.

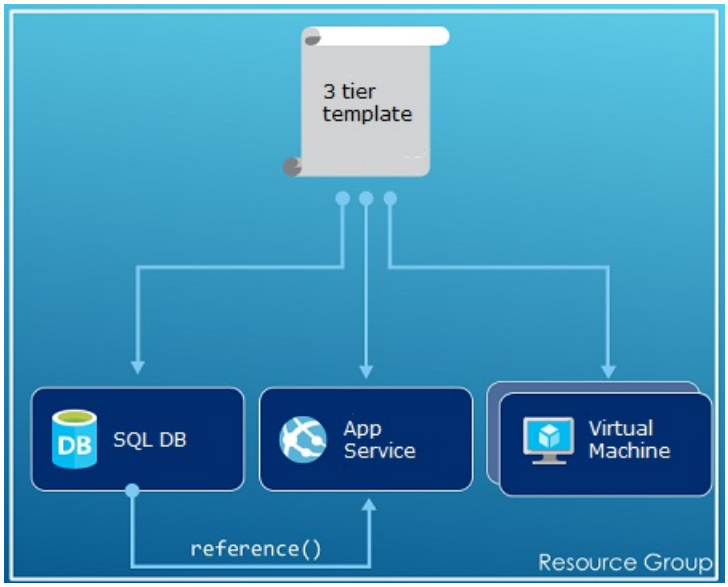
Resource Manager processes the template like any other request (see the image for Consistent management layer). It parses the template and converts its syntax into REST API operations for the appropriate resource providers. For example, when Resource Manager receives a template with the following resource definition:

```
"resources": [
  {
    "apiVersion": "2016-01-01",
    "type": "Microsoft.Storage/storageAccounts",
    "name": "mystorageaccount",
    "location": "westus",
    "sku": {
      "name": "Standard_LRS"
    },
    "kind": "Storage",
    "properties": {
    }
  }
]
```

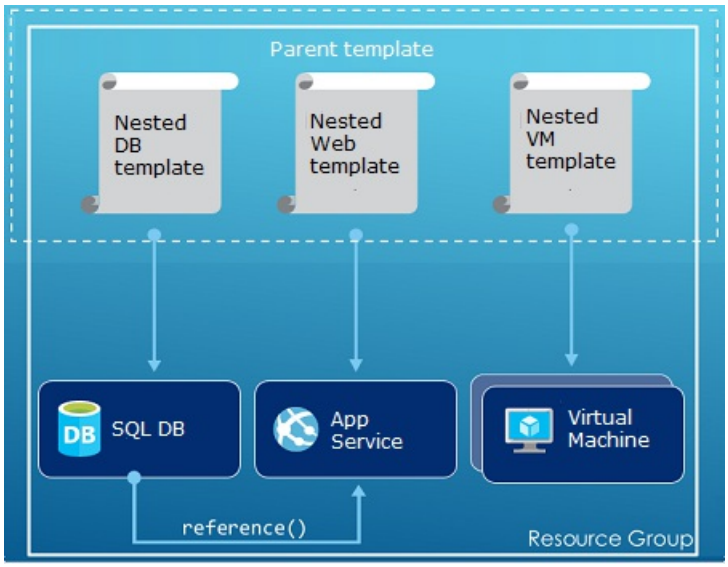
It converts the definition to the following REST API operation, which is sent to the Microsoft.Storage resource provider:

```
PUT
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Micro
soft.Storage/storageAccounts/mystorageaccount?api-version=2016-01-01
REQUEST BODY
{
  "location": "westus",
  "properties": {
  }
  "sku": {
    "name": "Standard_LRS"
  },
  "kind": "Storage"
}
```

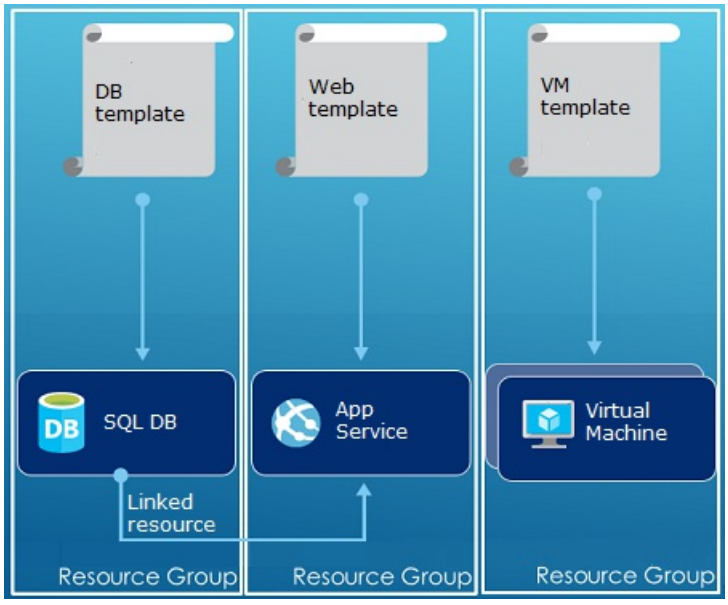
How you define templates and resource groups is entirely up to you and how you want to manage your solution. For example, you can deploy your three tier application through a single template to a single resource group.




But, you do not have to define your entire infrastructure in a single template. Often, it makes sense to divide your deployment requirements into a set of targeted, purpose-specific templates. You can easily reuse these templates for different solutions. To deploy a particular solution, you create a master template that links all the required templates. The following image shows how to deploy a three tier solution through a parent template that includes three nested templates.




If you envision your tiers having separate lifecycles, you can deploy your three tiers to separate resource groups. Notice the resources can still be linked to resources in other resource groups.




For more suggestions about designing your templates, see Patterns for designing Azure Resource Manager templates. For information about nested templates, see Using linked templates with Azure Resource Manager.

Azure Resource Manager analyzes dependencies to ensure resources are created in the correct order. If one resource relies on a value from another resource (such as a virtual machine needing a storage account for disks), you set a dependency. For more information, see Defining dependencies in Azure Resource Manager templates.

You can also use the template for updates to the infrastructure. For example, you can add a resource to your solution and add configuration rules for the resources that are already deployed. If the template specifies creating a resource but that resource already exists, Azure Resource Manager performs an update instead of creating a new asset. Azure Resource Manager updates the existing asset to the same state as it would be as new.

Resource Manager provides extensions for scenarios when you need additional operations such as installing particular software that is not included in the setup. If you are already using a configuration management service, like DSC, Chef or Puppet, you can continue working with that service by using extensions. For information about virtual machine extensions, see About virtual machine extensions and features.

Finally, the template becomes part of the source code for your app. You can check it in to your source code repository and update it as your app evolves. You can edit the template through Visual Studio.

After defining your template, you are ready to deploy the resources to Azure. For the commands to deploy the

resources, see:

- Deploy resources with Resource Manager templates and Azure PowerShell
- Deploy resources with Resource Manager templates and Azure CLI
- Deploy resources with Resource Manager templates and Azure portal
- Deploy resources with Resource Manager templates and Resource Manager REST API

## Tags

Resource Manager provides a tagging feature that enables you to categorize resources according to your requirements for managing or billing. Use tags when you have a complex collection of resource groups and resources, and need to visualize those assets in the way that makes the most sense to you. For example, you could tag resources that serve a similar role in your organization or belong to the same department. Without tags, users in your organization can create multiple resources that may be difficult to later identify and manage. For example, you may wish to delete all the resources for a particular project. If those resources are not tagged for the project, you have to manually find them. Tagging can be an important way for you to reduce unnecessary costs in your subscription.

Resources do not need to reside in the same resource group to share a tag. You can create your own tag taxonomy to ensure that all users in your organization use common tags rather than users inadvertently applying slightly different tags (such as "dept" instead of "department").

The following example shows a tag applied to a virtual machine.

```
"resources": [
  {
    "type": "Microsoft.Compute/virtualMachines",
    "apiVersion": "2015-06-15",
    "name": "SimpleWindowsVM",
    "location": "[resourceGroup().location]",
    "tags": {
        "costCenter": "Finance"
    },
    ...
  }
]
```

To retrieve all the resources with a tag value, use the following PowerShell cmdlet:

```
Find-AzureRmResource -TagName costCenter -TagValue Finance
```

Or, the following Azure CLI 2.0 command:

```
az resource list --tag costCenter=Finance
```

You can also view tagged resources through the Azure portal.

The usage report for your subscription includes tag names and values, which enables you to break out costs by tags. For more information about tags, see Using tags to organize your Azure resources.

## Access control

Resource Manager enables you to control who has access to specific actions for your organization. It natively integrates role-based access control (RBAC) into the management platform and applies that access control to all services in your resource group.

There are two main concepts to understand when working with role-based access control:

- Role definitions - describe a set of permissions and can be used in many assignments.
- Role assignments - associate a definition with an identity (user or group) for a particular scope (subscription, resource group, or resource). The assignment is inherited by lower scopes.

You can add users to pre-defined platform and resource-specific roles. For example, you can take advantage of the pre-defined role called Reader that permits users to view resources but not change them. You add users in your organization that need this type of access to the Reader role and apply the role to the subscription, resource group, or resource.

Azure provides the following four platform roles:

1. Owner - can manage everything, including access
2. Contributor - can manage everything except access
3. Reader - can view everything, but can't make changes
4. User Access Administrator - can manage user access to Azure resources

Azure also provides several resource-specific roles. Some common ones are:

1. Virtual Machine Contributor - can manage virtual machines but not grant access to them, and cannot manage the virtual network or storage account to which they are connected
2. Network Contributor - can manage all network resources, but not grant access to them
3. Storage Account Contributor - Can manage storage accounts, but not grant access to them
4. SQL Server Contributor - Can manage SQL servers and databases, but not their security-related policies
5. Website Contributor - Can manage websites, but not the web plans to which they are connected

For the full list of roles and permitted actions, see RBAC: Built in Roles. For more information about role-based access control, see Azure Role-based Access Control.

In some cases, you want to run code or script that accesses resources, but you do not want to run it under a user's credentials. Instead, you want to create an identity called a service principal for the application and assign the appropriate role for the service principal. Resource Manager enables you to create credentials for the application and programmatically authenticate the application. To learn about creating service principals, see one of following topics:

- Use Azure PowerShell to create a service principal to access resources
- Use Azure CLI to create a service principal to access resources
- Use portal to create Azure Active Directory application and service principal that can access resources

You can also explicitly lock critical resources to prevent users from deleting or modifying them. For more information, see Lock resources with Azure Resource Manager.

## Activity logs

Resource Manager logs all operations that create, modify, or delete a resource. You can use the activity logs to find an error when troubleshooting or to monitor how a user in your organization modified a resource. To see the logs, select **Activity logs** in the **Settings** blade for a resource group. You can filter the logs by many different values including which user initiated the operation. For information about working with the activity logs, see View activity logs to manage Azure resources.

## Customized policies

Resource Manager enables you to create customized policies for managing your resources. The types of policies you create can include diverse scenarios. You can enforce a naming convention on resources, limit which types and

instances of resources can be deployed, or limit which regions can host a type of resource. You can require a tag value on resources to organize billing by departments. You create policies to help reduce costs and maintain consistency in your subscription.

You define policies with JSON and then apply those policies either across your subscription or within a resource group. Policies are different than role-based access control because they are applied to resource types.

The following example shows a policy that ensures tag consistency by specifying that all resources include a costCenter tag.

```
{
  "if": {
    "not" : {
      "field" : "tags",
      "containsKey" : "costCenter"
    }
  },
  "then" : {
    "effect" : "deny"
  }
}
```

There are many more types of policies you can create. For more information, see What is Azure Policy?.

## SDKs

Azure SDKs are available for multiple languages and platforms. Each of these language implementations is available through its ecosystem package manager and GitHub.

Here are our Open Source SDK repositories. We welcome feedback, issues, and pull requests.

- Azure SDK for .NET
- Azure Management Libraries for Java
- Azure SDK for Node.js
- Azure SDK for PHP
- Azure SDK for Python
- Azure SDK for Ruby

For information about using these languages with your resources, see:

- Azure for .NET developers
- Azure for Java developers
- Azure for Node.js developers
- Azure for Python developers

> **NOTE**
> If the SDK doesn't provide the required functionality, you can also call to the Azure REST API directly.

## Next steps

- For a simple introduction to working with templates, see Export an Azure Resource Manager template from existing resources.
- For a more thorough walkthrough of creating a template, see Create your first Azure Resource Manager template.

- To understand the functions you can use in a template, see Template functions
- For information about using Visual Studio with Resource Manager, see Creating and deploying Azure resource groups through Visual Studio.

Here's a video demonstration of this overview:

# Keyboard shortcuts in the Azure portal

12/11/2017 • 1 min to read • Edit Online

This article shows the keyboard shortcuts that work throughout the Azure portal. Individual services may have their own keyboard shortcuts.

## Actions

| TO DO THIS | PRESS |
| --- | --- |
| Create a new resource | G+N |
| Open the 'More services' pane | G+B |
| Search resources | G+/ |
| Search resource menu items | CTRL+/ |
| Move the selected left pane item up | ALT+Shift+Up Arrow |
| Move the selected left pane item down | ALT+Shift+Down Arrow |

## Navigation

| TO DO THIS | PRESS |
| --- | --- |
| Move focus to command bar | G+, |
| Toggle focus between top bar and side bar | G+. |

## Go to

| TO DO THIS | PRESS |
| --- | --- |
| Go to dashboard | G+D |
| Go to all resources | G+A |
| Go to resource groups | G+R |
| Open the left pane item at this position | G+number |

# Supported browsers and devices for the Azure portal

8/25/2017 • 1 min to read • Edit Online

You can run the Azure portal on all modern desktop, tablet devices, and browsers.

### Supported devices

The Azure portal runs great on modern PCs, Macs, and tablets. If you need to manage your Azure resources from a mobile device, try the Azure app that's available for preview on iPhone and Android. For more information, see the blog post Introducing the Azure app public preview.

### Supported browsers

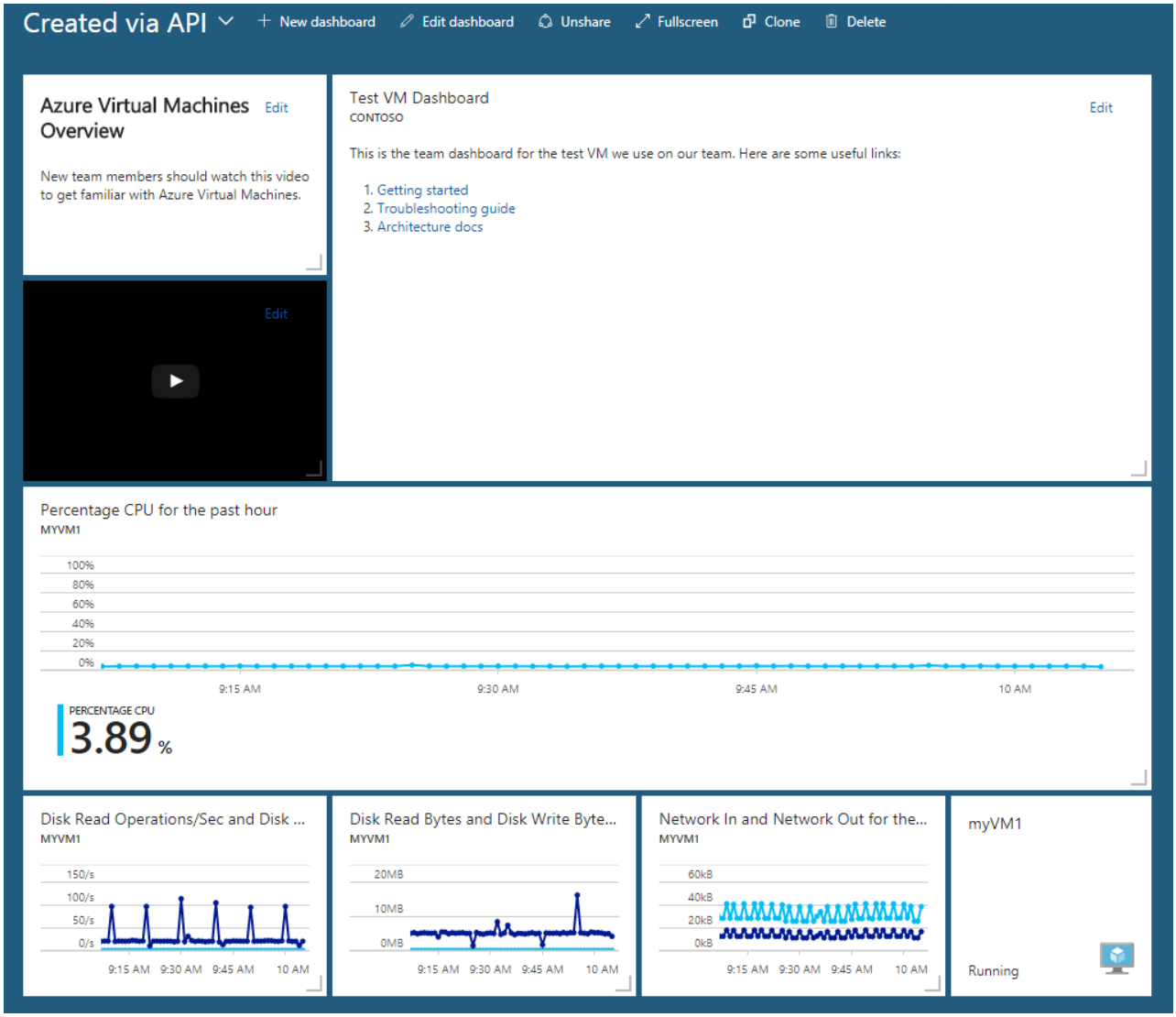
We recommend that you use the most up-to-date browser that's compatible with your operating system. The following browsers are supported:

- Microsoft Edge (latest version)
- Internet Explorer 11
- Safari (latest version, Mac only)
- Chrome (latest version)
- Firefox (latest version)

# The structure of Azure Dashboards

12/11/2017 • 7 min to read • Edit Online

This document walks through the structure of an Azure dashboard, using the following dashboard as an example:




Since shared Azure dashboards are resources, this dashboard can be represented as JSON. The following JSON represents the dashboard visualized above.

```
{
    "properties": {
        "lenses": {
            "0": {
                "order": 0,
                "parts": {
                    "0": {
                        "position": {
                            "x": 0,
                            "y": 0,
                            "rowSpan": 2,
                            "colSpan": 3
                        },
                        "metadata": {
                            "inputs": [],
                            "type": "Extension[azure]/HubsExtension/PartType/MarkdownPart"
```

```
                            "type": "Extension[azure]/HubsExtension/PartType/MarkdownPart",
                            "settings": {
                                "content": {
                                    "settings": {
                                        "content": "## Azure Virtual Machines Overview\r\nNew team members
should watch this video to get familiar with Azure Virtual Machines.",
                                        "title": "",
                                        "subtitle": ""
                                    }
                                }
                            }
                        }
                    },
                    "1": {
                        "position": {
                            "x": 3,
                            "y": 0,
                            "rowSpan": 4,
                            "colSpan": 8
                        },
                        "metadata": {
                            "inputs": [],
                            "type": "Extension[azure]/HubsExtension/PartType/MarkdownPart",
                            "settings": {
                                "content": {
                                    "settings": {
                                        "content": "This is the team dashboard for the test VM we use on our
team. Here are some useful links:\r\n\r\n1. [Getting started](https://www.contoso.com/tsgs)\r\n1.
[Troubleshooting guide](https://www.contoso.com/tsgs)\r\n1. [Architecture docs](https://www.contoso.com/tsgs)",
                                        "title": "Test VM Dashboard",
                                        "subtitle": "Contoso"
                                    }
                                }
                            }
                        }
                    },
                    "2": {
                        "position": {
                            "x": 0,
                            "y": 2,
                            "rowSpan": 2,
                            "colSpan": 3
                        },
                        "metadata": {
                            "inputs": [],
                            "type": "Extension[azure]/HubsExtension/PartType/VideoPart",
                            "settings": {
                                "content": {
                                    "settings": {
                                        "title": "",
                                        "subtitle": "",
                                        "src": "https://www.youtube.com/watch?
v=YcylDIiKaSU&list=PLLasX02E8BPCsnETz0XAMfpLR1LIBqpgs&index=4",
                                        "autoplay": false
                                    }
                                }
                            }
                        }
                    },
                    "3": {
                        "position": {
                            "x": 0,
                            "y": 4,
                            "rowSpan": 3,
                            "colSpan": 11
                        },
                        "metadata": {
                            "inputs": [
                                {
```

```
                                    "name": "queryInputs",
                                    "value": {
                                        "timespan": {
                                            "duration": "PT1H",
                                            "start": null,
                                            "end": null
                                        },
                                        "id": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1",
                                        "chartType": 0,
                                        "metrics": [
                                            {
                                                "name": "Percentage CPU",
                                                "resourceId": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                                            }
                                        ]
                                    }
                                }
                            ],
                            "type": "Extension/Microsoft_Azure_Monitoring/PartType/MetricsChartPart",
                            "settings": {}
                        }
                    },
                    "4": {
                        "position": {
                            "x": 0,
                            "y": 7,
                            "rowSpan": 2,
                            "colSpan": 3
                        },
                        "metadata": {
                            "inputs": [
                                {
                                    "name": "queryInputs",
                                    "value": {
                                        "timespan": {
                                            "duration": "PT1H",
                                            "start": null,
                                            "end": null
                                        },
                                        "id": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1",
                                        "chartType": 0,
                                        "metrics": [
                                            {
                                                "name": "Disk Read Operations/Sec",
                                                "resourceId": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                                            },
                                            {
                                                "name": "Disk Write Operations/Sec",
                                                "resourceId": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                                            }
                                        ]
                                    }
                                }
                            ],
                            "type": "Extension/Microsoft_Azure_Monitoring/PartType/MetricsChartPart",
                            "settings": {}
                        }
                    },
                    "5": {
                        "position": {
                            "x": 3,
                            "y": 7,
                            "rowSpan": 2,
                            "colSpan": 3
```

```
                        },
                        "metadata": {
                            "inputs": [
                                {
                                    "name": "queryInputs",
                                    "value": {
                                        "timespan": {
                                            "duration": "PT1H",
                                            "start": null,
                                            "end": null
                                        },
                                        "id": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1",
                                        "chartType": 0,
                                        "metrics": [
                                            {
                                                "name": "Disk Read Bytes",
                                                "resourceId": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                                            },
                                            {
                                                "name": "Disk Write Bytes",
                                                "resourceId": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                                            }
                                        ]
                                    }
                                }
                            ],
                            "type": "Extension/Microsoft_Azure_Monitoring/PartType/MetricsChartPart",
                            "settings": {}
                        }
                    },
                    "6": {
                        "position": {
                            "x": 6,
                            "y": 7,
                            "rowSpan": 2,
                            "colSpan": 3
                        },
                        "metadata": {
                            "inputs": [
                                {
                                    "name": "queryInputs",
                                    "value": {
                                        "timespan": {
                                            "duration": "PT1H",
                                            "start": null,
                                            "end": null
                                        },
                                        "id": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1",
                                        "chartType": 0,
                                        "metrics": [
                                            {
                                                "name": "Network In",
                                                "resourceId": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                                            },
                                            {
                                                "name": "Network Out",
                                                "resourceId": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                                            }
                                        ]
                                    }
                                }
                            ],
                            "type": "Extension/Microsoft_Azure_Monitoring/PartType/MetricsChartPart",
```

```
                            "settings": {}
                        }
                    },
                    "7": {
                        "position": {
                            "x": 9,
                            "y": 7,
                            "rowSpan": 2,
                            "colSpan": 2
                        },
                        "metadata": {
                            "inputs": [
                                {
                                    "name": "id",
                                    "value": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                                }
                            ],
                            "type": "Extension/Microsoft_Azure_Compute/PartType/VirtualMachinePart",
                            "asset": {
                                "idInputName": "id",
                                "type": "VirtualMachine"
                            },
                            "defaultMenuItemId": "overview"
                        }
                    }
                }
            }
        },
        "metadata": {
            "model": {
                "timeRange": {
                    "value": {
                        "relative": {
                            "duration": 24,
                            "timeUnit": 1
                        }
                    },
                    "type": "MsPortalFx.Composition.Configuration.ValueTypes.TimeRange"
                }
            }
        }
    },
    "id": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/dashboards/providers/Microsoft.Portal/dashboards/aa9786ae-e159-483f-b05f-
1f7f767741a9",
    "name": "aa9786ae-e159-483f-b05f-1f7f767741a9",
    "type": "Microsoft.Portal/dashboards",
    "location": "eastasia",
    "tags": {
        "hidden-title": "Created via API"
    }
}
```

## Common resource properties

Let's break down the relevant sections of the JSON. The top-level properties, **id**, **name**, **type**, **location**, and **tags** properties are shared across all Azure resource types. That is, they don't have much to do with the dashboard's content.

### The id property

The Azure resource id, subject to the naming conventions of Azure resources. When the portal creates a dashboard it generally chooses an id in the form of a guid, but you are free to use any valid name when you create them programmatically.

### The name property

The name is the segment of the resource Id that does not include the subscription, resource type, or resource group information. Essentially, it is the last segment of the resource id.

### The type property

All dashboards are of type **Microsoft.Portal/dashboards**.

### The location property

Unlike other resources, dashboards don't have a runtime component. For dashboards, the location indicates the primary geographic location that stores the dashboard's JSON representation. The value should be one of the location codes that can be fetched using the locations API on the subscriptions resource.

### The tags property

Tags are a common feature of Azure resources that let you organize your resource by arbitrary name value pairs. For dashboards, there is one special tag called **hidden-title**. If your dashboard has this property populated, then it is used as the display name for your dashboard in the portal. Azure resource Ids cannot be renamed, but tags can. This tag gives you a way to have a renamable display name for your dashboard.

```
"tags": { "hidden-title": "Created via API" }
```

### The properties object

The properties object contains two properties, **lenses** and **metadata**. The **lenses** property contains information about the tiles (a.k.a. parts) on the dashboard. The **metadata** property is there for potential future features.

### The lenses property

The **lenses** property contains the dashboard. Note that the lenses object in this example contains a single property called "0". Lenses are a grouping concept that is not currently implemented in dashboards. For now, all of your dashboards have this single property on the lens object, again, called "0".

### The lens object

The object underneath the "0" contains two properties, **order** and **parts**. In the current version of dashboards, **order** is always 0. The **parts** property contains an object that defines the individual parts (a.k.a. tiles) on the dashboard.

The **parts** object contains a property for each part, where the name of the property is a number. This number is not significant.

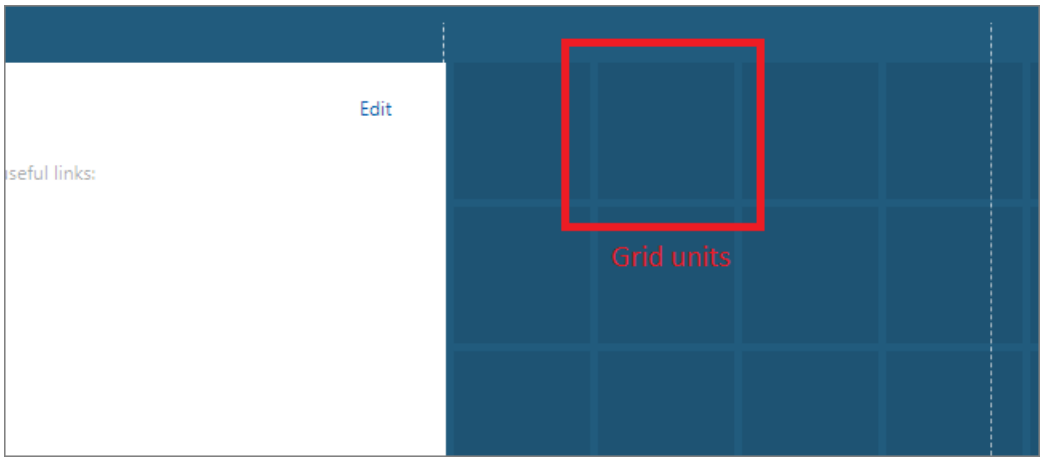
### The part object

Each individual part object has a **position**, and **metadata**.

### The position object

The **position** property contains the size and location information for the part expressed as **x**, **y**, **rowSpan**, and **colSpan**. The values are in terms of grid units. These grid units are visible when the dashboard is in the customize mode as shown here. If you want a tile to have a width of two grid units, a height of one grid unit, and a location in the top left corner of the dashboard then the position obejct looks like this:

```
location: { x: 0, y: 0, rowSpan: 2, colSpan: 1 }
```




### The metadata object

Each part has a metadata property, an object has only one required property called **type**. This string tells the portal which tile to show. Our example dashboard uses these types of tiles:

1. `Extension/Microsoft_Azure_Monitoring/PartType/MetricsChartPart` – Used to show monitoring metrics
2. `Extension[azure]/HubsExtension/PartType/MarkdownPart` – Used to show with text or images with basic formatting for lists, links, etc.
3. `Extension[azure]/HubsExtension/PartType/VideoPart` – Used to show videos from YouTube, Channel9, and any other type of video that works in an html video tag.
4. `Extension/Microsoft_Azure_Compute/PartType/VirtualMachinePart` – Used to show the name and status of an Azure virtual machine.

Each type of part has its own configuration. The possible configuration properties are called **inputs**, **settings**, and **asset**.

### The inputs object

The inputs object generally contains information that binds a tile to a resource instance. The virtual machine part in our sample dashboard contains a single input that uses the Azure resource id to express the binding. This resource id format is consistent across all Azure resources.

```
"inputs":
[
    {
        "name": "id",
        "value": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
    }
]
```

The metrics chart part has a single input that expresses the resource to bind to, as well as information about the metric(s) being displayed. Here is the input for the tile that's showing the Network In and Network Out metrics.

```
“inputs”:
[
    {
        "name": "queryInputs",
        "value":
        {
            "timespan":
            {
                "duration": "PT1H",
                "start": null,
                "end": null
           },
            "id": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1",
            "chartType": 0,
            "metrics":
            [
                {
                    "name": "Network In",
                    "resourceId": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                },
                {
                    "name": "Network Out",
                    "resourceId": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                }
            ]
        }
    }
]
```

**The settings object**

The settings object contains the configurable elements of a part. In our sample dashboard, the Markdown part uses settings to store the custom markdown content as well as a configurable title and subtitle.

```
"settings":
{
    "content":
    {
        "settings":
        {
            "content": "This is the team dashboard for the test VM we use on our team. Here are some useful
links:\r\n\r\n1. [Getting started](https://www.contoso.com/tsgs)\r\n1. [Troubleshooting guide]
(https://www.contoso.com/tsgs)\r\n1. [Architecture docs](https://www.contoso.com/tsgs)",
            "title": "Test VM Dashboard",
            "subtitle": "Contoso"
        }
    }
}
```

Similarly, the video tile has its own settings that contain a pointer to the video to play, an autoplay setting, and optional title information.

```
"settings":
{
   "content":
    {
        "settings":
        {
            "title": "",
            "subtitle": "",
            "src": "https://www.youtube.com/watch?
v=YcylDIiKaSU&list=PLLasX02E8BPCsnETz0XAMfpLR1LIBqpgs&index=4",
            "autoplay": false
        }
    }
}
```

**The asset object**

Tiles that are bound to first class manageable portal objects (called assets) have this relationship expressed via the asset object. In our example dashboard, the virtual machine tile contains this asset description. The **idInputName** property tells the portal that the id input contains the unique identifier for the asset, in this case the resource id. Most Azure resource types have assets defined in the portal.

```
"asset": { "idInputName": "id", "type": "VirtualMachine" }
```

# Understand the structure and syntax of Azure Resource Manager templates

12/14/2017 • 5 min to read • Edit Online

This article describes the structure of an Azure Resource Manager template. It presents the different sections of a template and the properties that are available in those sections. The template consists of JSON and expressions that you can use to construct values for your deployment. For a step-by-step tutorial on creating a template, see Create your first Azure Resource Manager template.

## Template format

In its simplest structure, a template contains the following elements:

```
{
    "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "",
    "parameters": {  },
    "variables": {  },
    "resources": [  ],
    "outputs": {  }
}
```

| ELEMENT NAME | REQUIRED | DESCRIPTION |
|---|---|---|
| $schema | Yes | Location of the JSON schema file that describes the version of the template language. Use the URL shown in the preceding example. |
| contentVersion | Yes | Version of the template (such as 1.0.0.0). You can provide any value for this element. When deploying resources using the template, this value can be used to make sure that the right template is being used. |
| parameters | No | Values that are provided when deployment is executed to customize resource deployment. |
| variables | No | Values that are used as JSON fragments in the template to simplify template language expressions. |
| resources | Yes | Resource types that are deployed or updated in a resource group. |
| outputs | No | Values that are returned after deployment. |

Each element contains properties you can set. The following example contains the full syntax for a template:

```
{
    "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "",
    "parameters": {
        "<parameter-name>" : {
            "type" : "<type-of-parameter-value>",
            "defaultValue": "<default-value-of-parameter>",
            "allowedValues": [ "<array-of-allowed-values>" ],
            "minValue": <minimum-value-for-int>,
            "maxValue": <maximum-value-for-int>,
            "minLength": <minimum-length-for-string-or-array>,
            "maxLength": <maximum-length-for-string-or-array-parameters>,
            "metadata": {
                "description": "<description-of-the parameter>"
            }
        }
    },
    "variables": {
        "<variable-name>": "<variable-value>",
        "<variable-object-name>": {
            <variable-complex-type-value>
        },
        "<variable-object-name>": {
            "copy": [
                {
                    "name": "<name-of-array-property>",
                    "count": <number-of-iterations>,
                    "input": {
                        <properties-to-repeat>
                    }
                }
            ]
        },
        "copy": [
            {
                "name": "<variable-array-name>",
                "count": <number-of-iterations>,
                "input": {
                    <properties-to-repeat>
                }
            }
        ]
    },
    "resources": [
      {
          "condition": "<boolean-value-whether-to-deploy>",
          "apiVersion": "<api-version-of-resource>",
          "type": "<resource-provider-namespace/resource-type-name>",
          "name": "<name-of-the-resource>",
          "location": "<location-of-resource>",
          "tags": {
              "<tag-name1>": "<tag-value1>",
              "<tag-name2>": "<tag-value2>"
          },
          "comments": "<your-reference-notes>",
          "copy": {
              "name": "<name-of-copy-loop>",
              "count": "<number-of-iterations>",
              "mode": "<serial-or-parallel>",
              "batchSize": "<number-to-deploy-serially>"
          },
          "dependsOn": [
              "<array-of-related-resource-names>"
          ],
          "properties": {
              "<settings-for-the-resource>",
              "copy": [
                  {
                      "name": ,
```

```
                    "count": ,
                    "input": {}
                }
            ]
        },
        "resources": [
            "<array-of-child-resources>"
        ]
      }
    ],
    "outputs": {
        "<outputName>" : {
            "type" : "<type-of-output-value>",
            "value": "<output-value-expression>"
        }
    }
}
```

This article describes the sections of the template in greater detail.

## Syntax

The basic syntax of the template is JSON. However, expressions and functions extend the JSON values available within the template. Expressions are written within JSON string literals whose first and last characters are the brackets: `[` and `]`, respectively. The value of the expression is evaluated when the template is deployed. While written as a string literal, the result of evaluating the expression can be of a different JSON type, such as an array or integer, depending on the actual expression. To have a literal string start with a bracket `[`, but not have it interpreted as an expression, add an extra bracket to start the string with `[[`.

Typically, you use expressions with functions to perform operations for configuring the deployment. Just like in JavaScript, function calls are formatted as `functionName(arg1,arg2,arg3)`. You reference properties by using the dot and [index] operators.

The following example shows how to use several functions when constructing a value:

```
"variables": {
    "storageName": "[concat(toLower(parameters('storageNamePrefix')), uniqueString(resourceGroup().id))]"
}
```

For the full list of template functions, see Azure Resource Manager template functions.

## Parameters

In the parameters section of the template, you specify which values you can input when deploying the resources. These parameter values enable you to customize the deployment by providing values that are tailored for a particular environment (such as dev, test, and production). You do not have to provide parameters in your template, but without parameters your template would always deploy the same resources with the same names, locations, and properties.

The following example shows a simple parameter definition:

```
"parameters": {
  "siteNamePrefix": {
    "type": "string",
    "metadata": {
      "description": "The name prefix of the web app that you wish to create."
    }
  },
},
```

For information about defining parameters, see Parameters section of Azure Resource Manager templates.

## Variables

In the variables section, you construct values that can be used throughout your template. You do not need to define variables, but they often simplify your template by reducing complex expressions.

The following example shows a simple variable definition:

```
"variables": {
  "webSiteName": "[concat(parameters('siteNamePrefix'), uniqueString(resourceGroup().id))]",
},
```

For information about defining variables, see Variables section of Azure Resource Manager templates.

## Resources

In the resources section, you define the resources that are deployed or updated. This section can get complicated because you must understand the types you are deploying to provide the right values.

```
"resources": [
  {
    "apiVersion": "2016-08-01",
    "name": "[variables('webSiteName')]",
    "type": "Microsoft.Web/sites",
    "location": "[resourceGroup().location]",
    "properties": {
      "serverFarmId": "/subscriptions/<subscription-id>/resourcegroups/<resource-group-
name>/providers/Microsoft.Web/serverFarms/<plan-name>"
    }
  }
],
```

For more information, see Resources section of Azure Resource Manager templates.

## Outputs

In the Outputs section, you specify values that are returned from deployment. For example, you could return the URI to access a deployed resource.

```
"outputs": {
  "newHostName": {
    "type": "string",
    "value": "[reference(variables('webSiteName')).defaultHostName]"
  }
}
```

For more information, see Outputs section of Azure Resource Manager templates.

## Template limits

Limit the size of your template to 1 MB, and each parameter file to 64 KB. The 1-MB limit applies to the final state of the template after it has been expanded with iterative resource definitions, and values for variables and parameters.

You are also limited to:

- 256 parameters
- 256 variables
- 800 resources (including copy count)
- 64 output values
- 24,576 characters in a template expression

You can exceed some template limits by using a nested template. For more information, see Using linked templates when deploying Azure resources. To reduce the number of parameters, variables, or outputs, you can combine several values into an object. For more information, see Objects as parameters.

## Next steps

- To view complete templates for many different types of solutions, see the Azure Quickstart Templates.
- For details about the functions you can use from within a template, see Azure Resource Manager Template Functions.
- To combine multiple templates during deployment, see Using linked templates with Azure Resource Manager.
- You may need to use resources that exist within a different resource group. This scenario is common when working with storage accounts or virtual networks that are shared across multiple resource groups. For more information, see the resourceId function.

# Deploy resources with Resource Manager templates and Azure PowerShell

12/6/2017 • 8 min to read • Edit Online

This article explains how to use Azure PowerShell with Resource Manager templates to deploy your resources to Azure. If you are not familiar with the concepts of deploying and managing your Azure solutions, see Azure Resource Manager overview.

The Resource Manager template you deploy can either be a local file on your machine, or an external file that is located in a repository like GitHub. The template you deploy in this article is available in the Sample template section, or as storage account template in GitHub.

If needed, install the Azure PowerShell module using the instructions found in the Azure PowerShell guide, and then run `Login-AzureRmAccount` to create a connection with Azure. Also, you need to have an SSH public key named `id_rsa.pub` in the .ssh directory of your user profile.

## Deploy a template from your local machine

When deploying resources to Azure, you:

1. Log in to your Azure account
2. Create a resource group that serves as the container for the deployed resources. The name of the resource group can only include alphanumeric characters, periods, underscores, hyphens, and parenthesis. It can be up to 90 characters. It cannot end in a period.
3. Deploy to the resource group the template that defines the resources to create

A template can include parameters that enable you to customize the deployment. For example, you can provide values that are tailored for a particular environment (such as dev, test, and production). The sample template defines a parameter for the storage account SKU.

The following example creates a resource group, and deploys a template from your local machine:

```
Login-AzureRmAccount

Select-AzureRmSubscription -SubscriptionName <yourSubscriptionName>

New-AzureRmResourceGroup -Name ExampleResourceGroup -Location "South Central US"
New-AzureRmResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName ExampleResourceGroup `
  -TemplateFile c:\MyTemplates\storage.json -storageAccountType Standard_GRS
```

The deployment can take a few minutes to complete. When it finishes, you see a message that includes the result:

```
ProvisioningState       : Succeeded
```

## Deploy a template from an external source

Instead of storing Resource Manager templates on your local machine, you may prefer to store them in an external location. You can store templates in a source control repository (such as GitHub). Or, you can store them in an Azure storage account for shared access in your organization.

To deploy an external template, use the **TemplateUri** parameter. Use the URI in the example to deploy the sample template from GitHub.

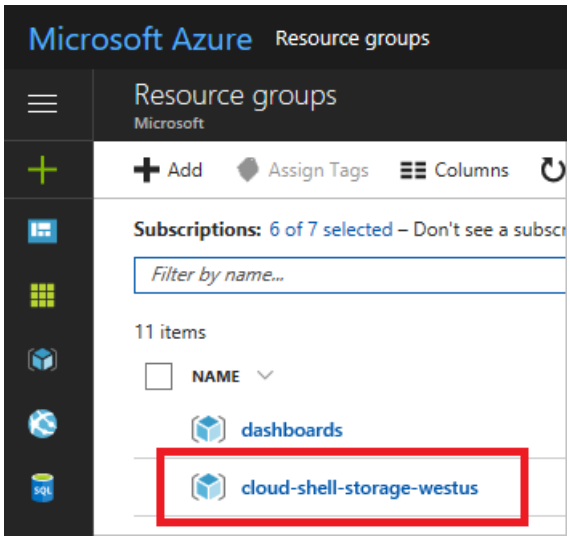
```
New-AzureRmResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName ExampleResourceGroup `
  -TemplateUri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-storage-account-create/azuredeploy.json `
  -storageAccountType Standard_GRS
```

The preceding example requires a publicly accessible URI for the template, which works for most scenarios because your template should not include sensitive data. If you need to specify sensitive data (like an admin password), pass that value as a secure parameter. However, if you do not want your template to be publicly accessible, you can protect it by storing it in a private storage container. For information about deploying a template that requires a shared access signature (SAS) token, see Deploy private template with SAS token.

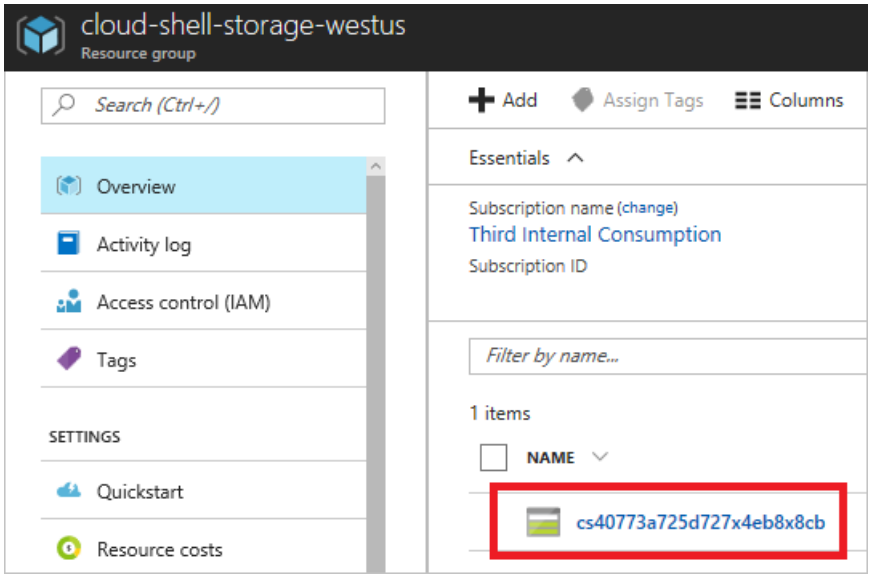
## Deploy template from Cloud Shell

You can use Cloud Shell to deploy your template. However, you must first load your template into the file share for your Cloud Shell. If you have not used Cloud Shell, see Overview of Azure Cloud Shell for information about setting it up.

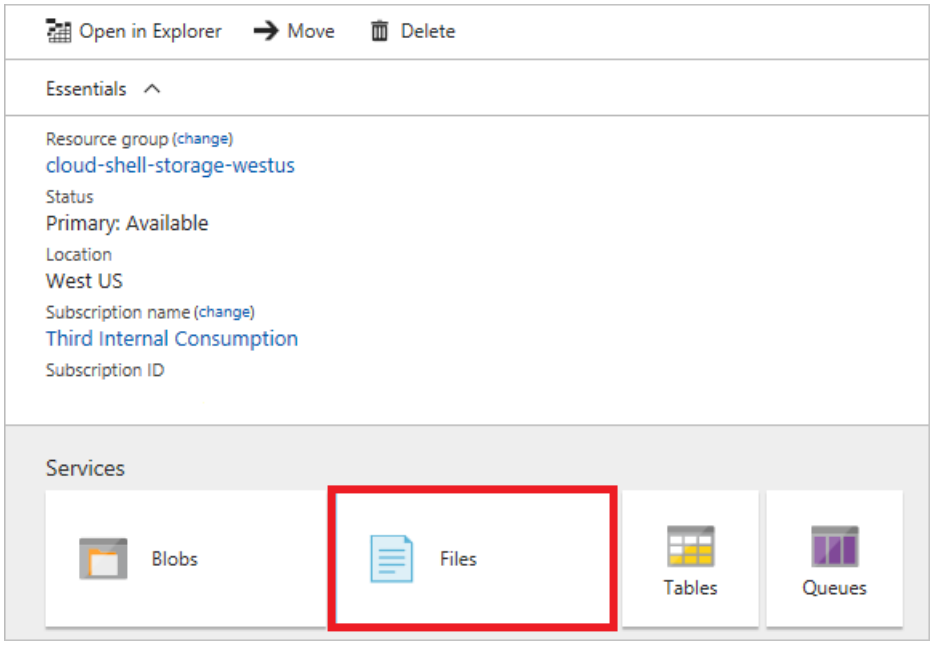
1. Log in to the Azure portal.
2. Select your Cloud Shell resource group. The name pattern is `cloud-shell-storage-<region>`.



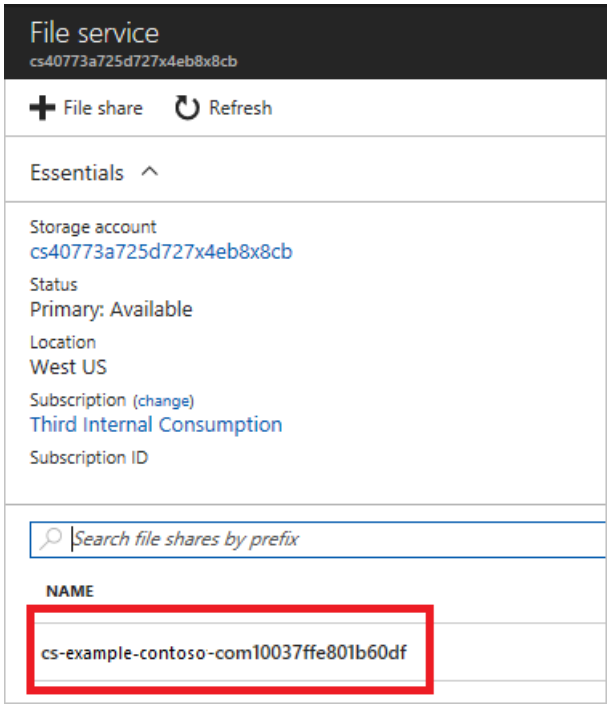

3. Select the storage account for your Cloud Shell.



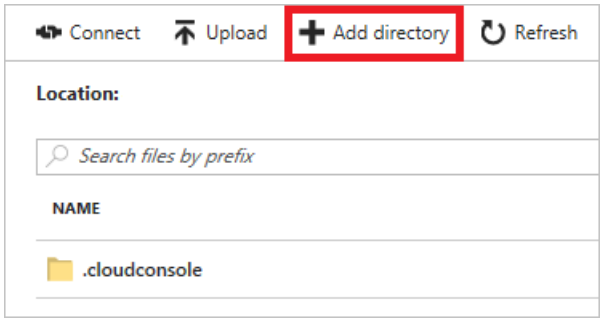
4. Select **Files**.



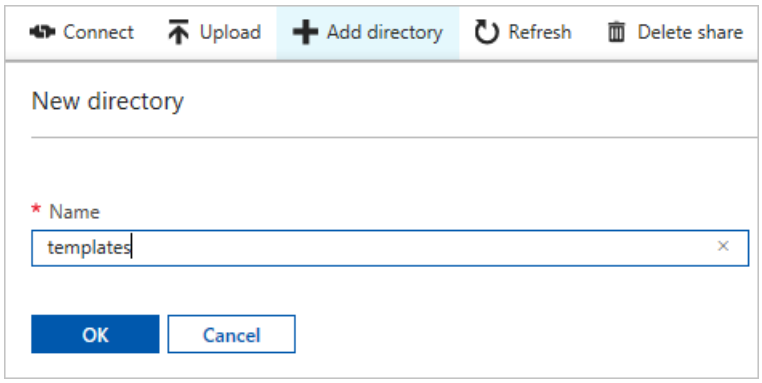

5. Select the file share for Cloud Shell. The name pattern is `cs-<user>-<domain>-com-<uniqueGuid>`.



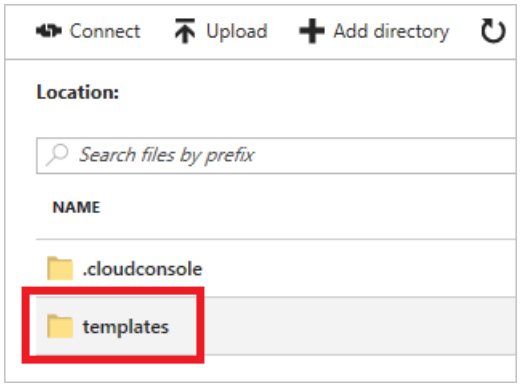

6. Select **Add directory**.



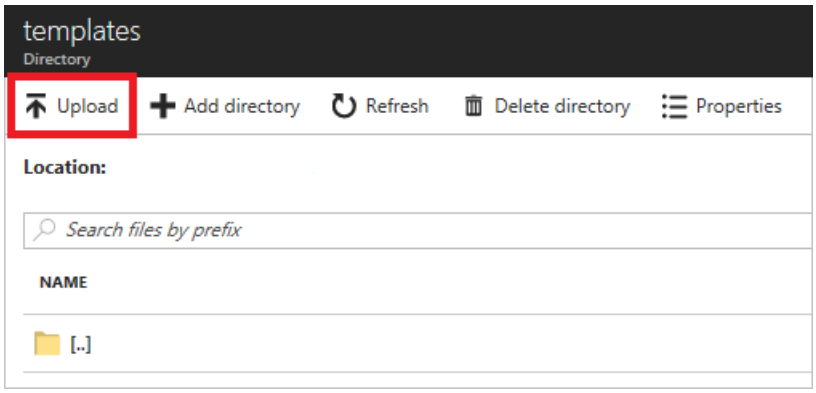

7. Name it **templates**, and select **Okay**.



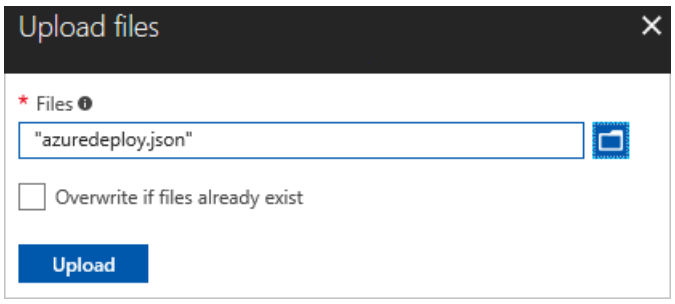

8. Select your new directory.




9. Select **Upload**.




10. Find and upload your template.




11. Open the prompt.

In the Cloud Shell, use the following commands:

```
New-AzureRmResourceGroup -Name ExampleResourceGroup -Location "South Central US"
New-AzureRmResourceGroupDeployment -ResourceGroupName ExampleResourceGroup -TemplateFile
"C:\users\ContainerAdministrator\CloudDrive\templates\azuredeploy.json" -storageAccountType Standard_GRS
```

## Parameter files

Rather than passing parameters as inline values in your script, you may find it easier to use a JSON file that contains the parameter values. The parameter file must be in the following format:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
     "storageAccountType": {
         "value": "Standard_GRS"
     }
  }
}
```

Notice that the parameters section includes a parameter name that matches the parameter defined in your template (storageAccountType). The parameter file contains a value for the parameter. This value is automatically passed to the template during deployment. You can create multiple parameter files for different deployment scenarios, and then pass in the appropriate parameter file.

Copy the preceding example and save it as a file named `storage.parameters.json`.

To pass a local parameter file, use the **TemplateParameterFile** parameter:

```
New-AzureRmResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName ExampleResourceGroup `
  -TemplateFile c:\MyTemplates\storage.json `
  -TemplateParameterFile c:\MyTemplates\storage.parameters.json
```

To pass an external parameter file, use the **TemplateParameterUri** parameter:

```
New-AzureRmResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName ExampleResourceGroup `
  -TemplateUri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-storage-account-
create/azuredeploy.json `
  -TemplateParameterUri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-storage-
account-create/azuredeploy.parameters.json
```

You can use inline parameters and a local parameter file in the same deployment operation. For example, you can specify some values in the local parameter file and add other values inline during deployment. If you provide values for a parameter in both the local parameter file and inline, the inline value takes precedence.

However, when you use an external parameter file, you cannot pass other values either inline or from a local file. When you specify a parameter file in the **TemplateParameterUri** parameter, all inline parameters are ignored. Provide all parameter values in the external file. If your template includes a sensitive value that you cannot include in the parameter file, either add that value to a key vault, or dynamically provide all parameter values inline.

If your template includes a parameter with the same name as one of the parameters in the PowerShell command, PowerShell presents the parameter from your template with the postfix **FromTemplate**. For example, a parameter named **ResourceGroupName** in your template conflicts with the **ResourceGroupName** parameter in the New-AzureRmResourceGroupDeployment cmdlet. You are prompted to provide a value for **ResourceGroupNameFromTemplate**. In general, you should avoid this confusion by not naming parameters with the same name as parameters used for deployment operations.

## Test a template deployment

To test your template and parameter values without actually deploying any resources, use Test-AzureRmResourceGroupDeployment.

```
Test-AzureRmResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName ExampleResourceGroup `
  -TemplateFile c:\MyTemplates\storage.json -storageAccountType Standard_GRS
```

If no errors are detected, the command finishes without a response. If an error is detected, the command returns an error message. For example, attempting to pass an incorrect value for the storage account SKU, returns the following error:

```
Test-AzureRmResourceGroupDeployment -ResourceGroupName testgroup `
  -TemplateFile c:\MyTemplates\storage.json -storageAccountType badSku

Code    : InvalidTemplate
Message : Deployment template validation failed: 'The provided value 'badSku' for the template parameter
'storageAccountType'
          at line '15' and column '24' is not valid. The parameter value is not part of the allowed value(s):
          'Standard_LRS,Standard_ZRS,Standard_GRS,Standard_RAGRS,Premium_LRS'.'.
Details :
```

If your template has a syntax error, the command returns an error indicating it could not parse the template. The message indicates the line number and position of the parsing error.

```
Test-AzureRmResourceGroupDeployment : After parsing a value an unexpected character was encountered:
  ". Path 'variables', line 31, position 3.
```

## Incremental and complete deployments

When deploying your resources, you specify that the deployment is either an incremental update or a complete update. The primary difference between these two modes is how Resource Manager handles existing resources in the resource group that are not in the template:

- In complete mode, Resource Manager **deletes** resources that exist in the resource group but are not specified in the template.
- In incremental mode, Resource Manager **leaves unchanged** resources that exist in the resource group but are not specified in the template.

For both modes, Resource Manager attempts to provision all resources specified in the template. If the resource already exists in the resource group and its settings are unchanged, the operation results in no change. If you change the settings for a resource, the resource is provisioned with those new settings. If you attempt to update the location or type of an existing resource, the deployment fails with an error. Instead, deploy a new resource with the location or type that you need.

By default, Resource Manager uses the incremental mode.

To illustrate the difference between incremental and complete modes, consider the following scenario.

**Existing Resource Group** contains:

- Resource A
- Resource B
- Resource C

**Template** defines:

- Resource A
- Resource B
- Resource D

When deployed in **incremental** mode, the resource group contains:

- Resource A
- Resource B
- Resource C
- Resource D

When deployed in **complete** mode, Resource C is deleted. The resource group contains:

- Resource A
- Resource B
- Resource D

To use complete mode, use the `Mode` parameter:

```
New-AzureRmResourceGroupDeployment -Mode Complete -Name ExampleDeployment `
  -ResourceGroupName ExampleResourceGroup -TemplateFile c:\MyTemplates\storage.json
```

## Sample template

The following template is used for the examples in this article. Copy and save it as a file named storage.json. To understand how to create this template, see Create your first Azure Resource Manager template.

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountType": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_ZRS",
        "Premium_LRS"
      ],
      "metadata": {
        "description": "Storage Account type"
      }
    }
  },
  "variables": {
    "storageAccountName": "[concat(uniquestring(resourceGroup().id), 'standardsa')]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "name": "[variables('storageAccountName')]",
      "apiVersion": "2016-01-01",
      "location": "[resourceGroup().location]",
      "sku": {
          "name": "[parameters('storageAccountType')]"
      },
      "kind": "Storage",
      "properties": {
      }
    }
  ],
  "outputs": {
      "storageAccountName": {
          "type": "string",
          "value": "[variables('storageAccountName')]"
      }
  }
}
```

## Next steps

- The examples in this article deploy resources to a resource group in your default subscription. To use a different subscription, see Manage multiple Azure subscriptions.
- For a complete sample script that deploys a template, see Resource Manager template deployment script.
- To understand how to define parameters in your template, see Understand the structure and syntax of Azure Resource Manager templates.
- For tips on resolving common deployment errors, see Troubleshoot common Azure deployment errors with Azure Resource Manager.
- For information about deploying a template that requires a SAS token, see Deploy private template with SAS token.
- For guidance on how enterprises can use Resource Manager to effectively manage subscriptions, see Azure enterprise scaffold - prescriptive subscription governance.

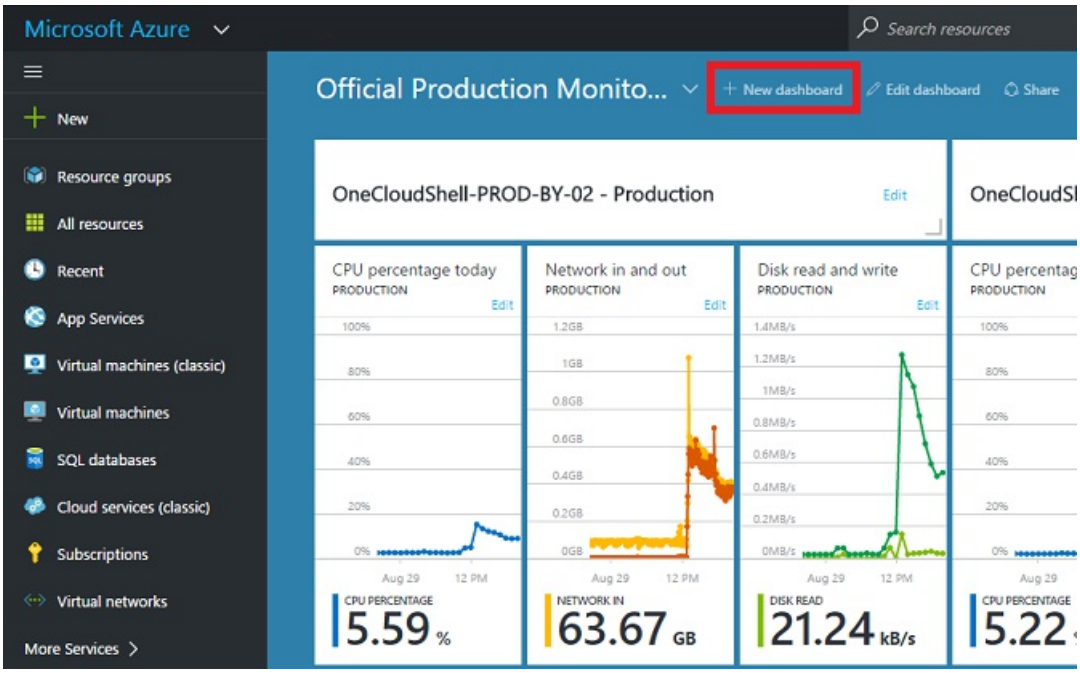
# Create and share dashboards in the Azure portal

12/11/2017 • 4 min to read • Edit Online

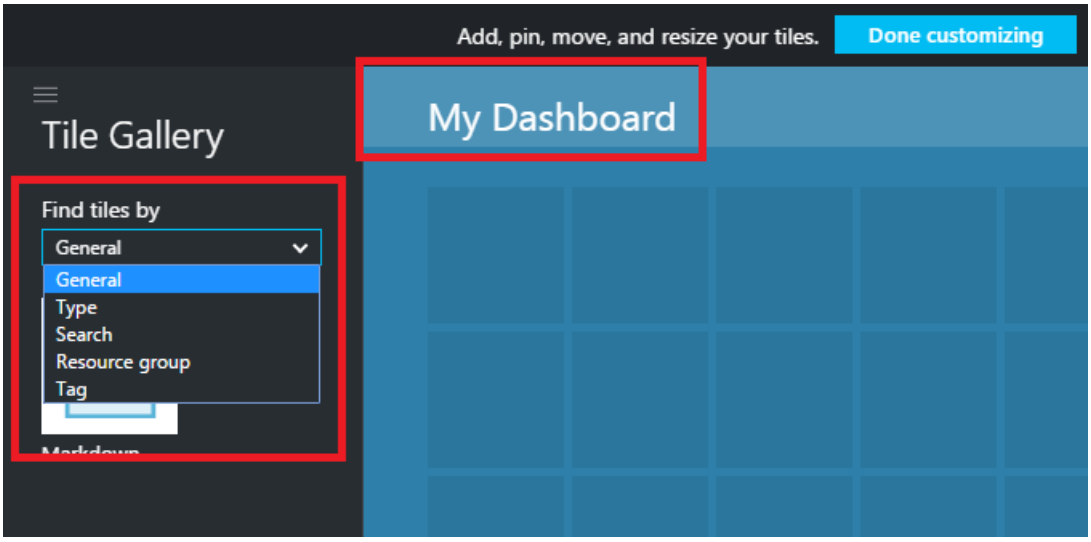
You can create multiple dashboards and share them with others who have access to your Azure subscriptions. This article goes through the basics of creating, editing, publishing, and managing access to dashboards.

## Create a dashboard

To create a dashboard, select the **New dashboard** button next to the current dashboard's name.



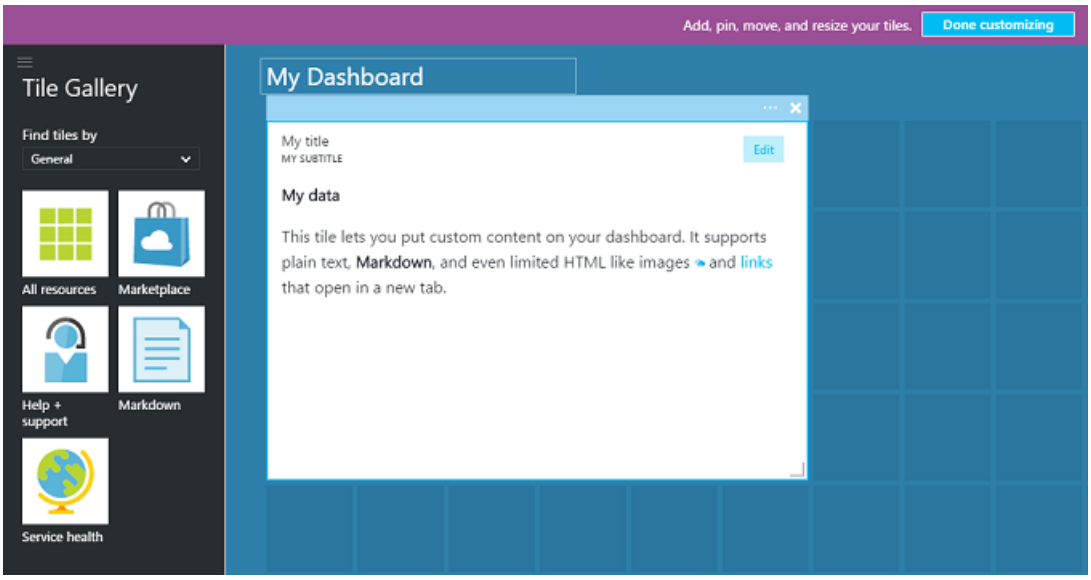

This action creates a new, empty, private dashboard and puts you into customization mode where you can name your dashboard and add or rearrange tiles. When in this mode, the collapsible tile gallery takes over the left navigation menu. The tile gallery lets you find tiles for your Azure resources in various ways: you can browse by resource group, by resource type, by tag, or by searching for your resource by name.




Add tiles by dragging and dropping them onto the dashboard surface wherever you want.

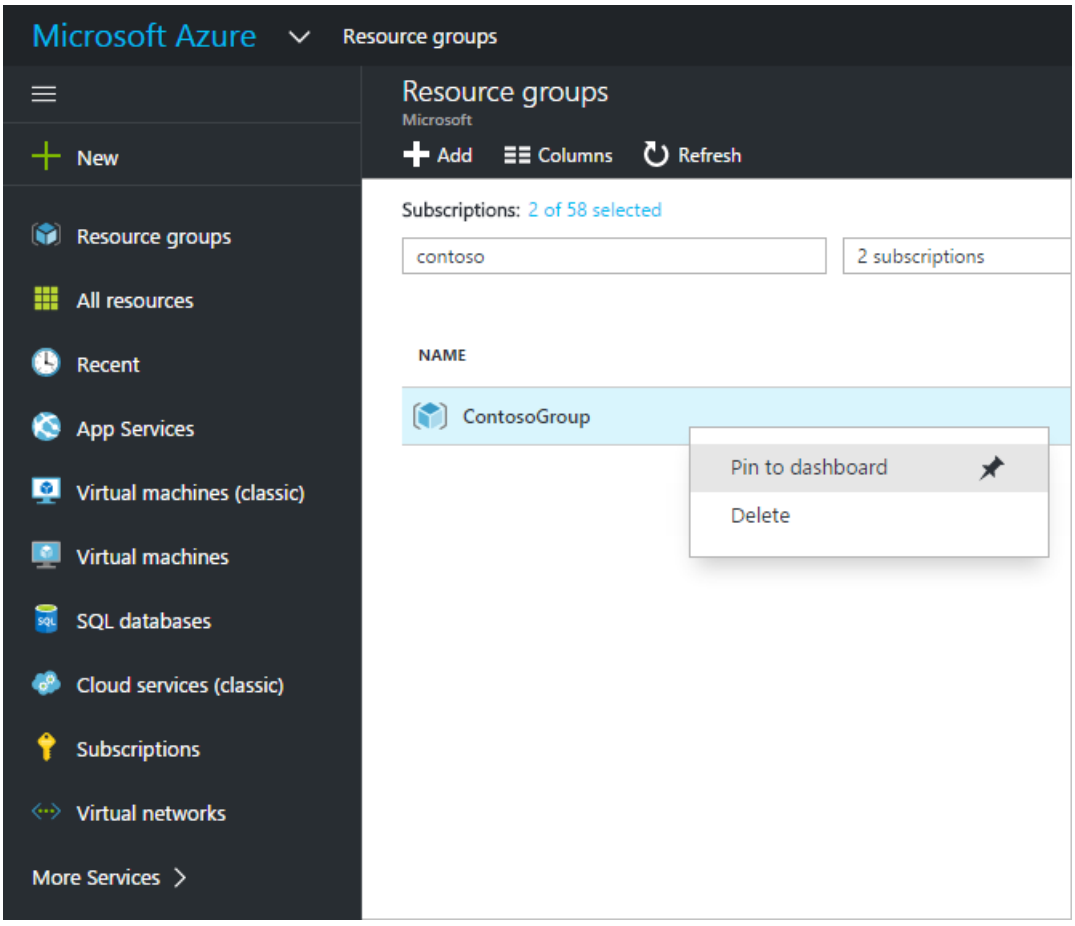
There's a new category called **General** for tiles that are not associated with a particular resource. In this example,

we pin the Markdown tile. You use this tile to add custom content to your dashboard. The tile supports plain text, Markdown syntax, and a limited set of HTML. (For safety, you can't do things like inject `<script>` tags or use certain styling element of CSS that might interfere with the portal.)

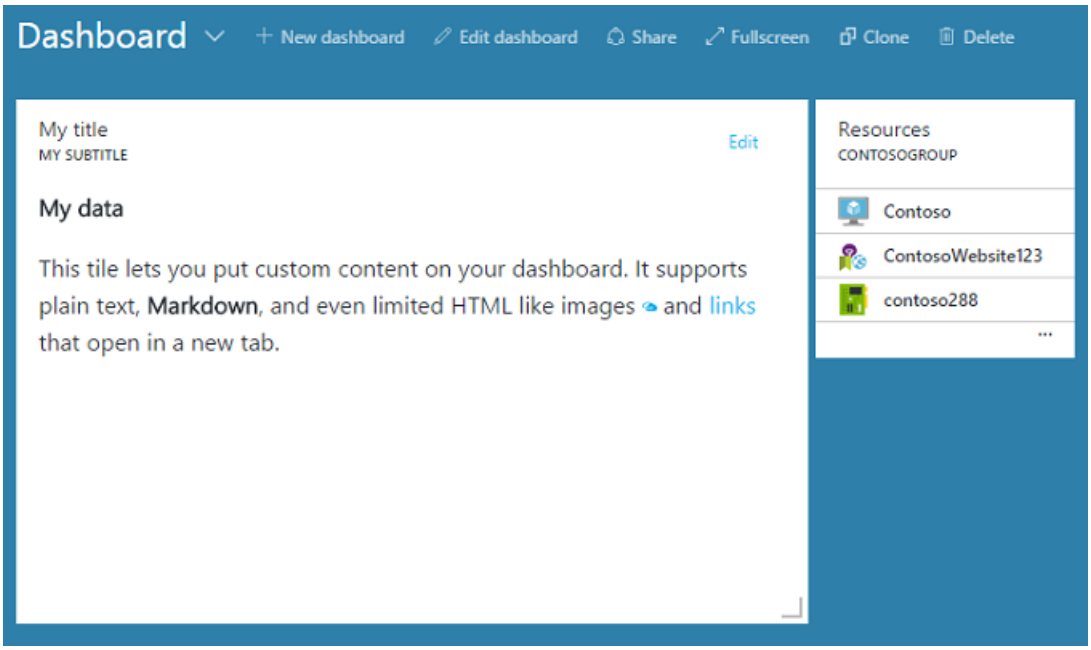



## Edit a dashboard

After creating your dashboard, you can pin tiles from the tile gallery or the tile representation of blades. Let's pin the representation of our resource group. You can either pin when browsing the item, or from the resource group blade. Both approaches result in pinning the tile representation of the resource group.



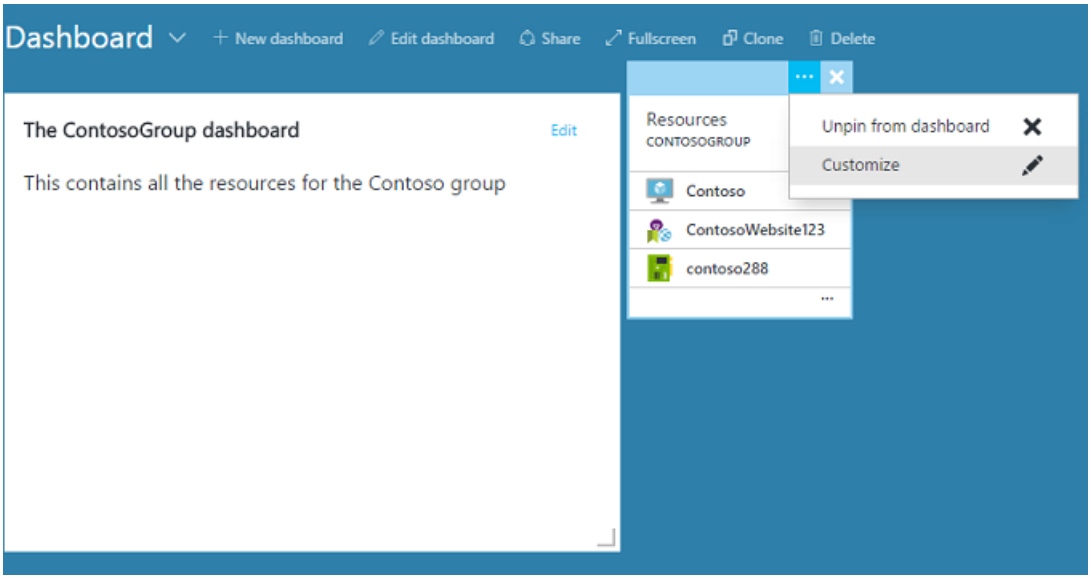

After pinning the item, it appears on your dashboard.



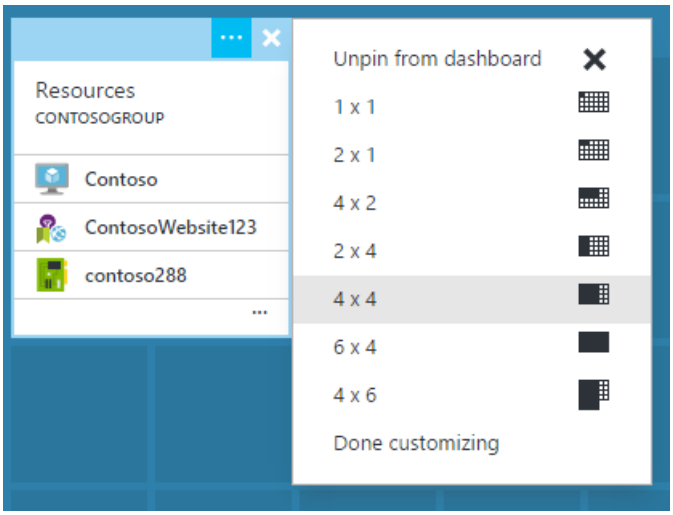

Now that we have a Markdown tile and a resource group pinned to the dashboard, we can resize and rearrange the tiles into a suitable layout.

By hovering and selecting "..." or right-clicking on a tile you can see all the contextual commands for that tile. By default, there are two items:

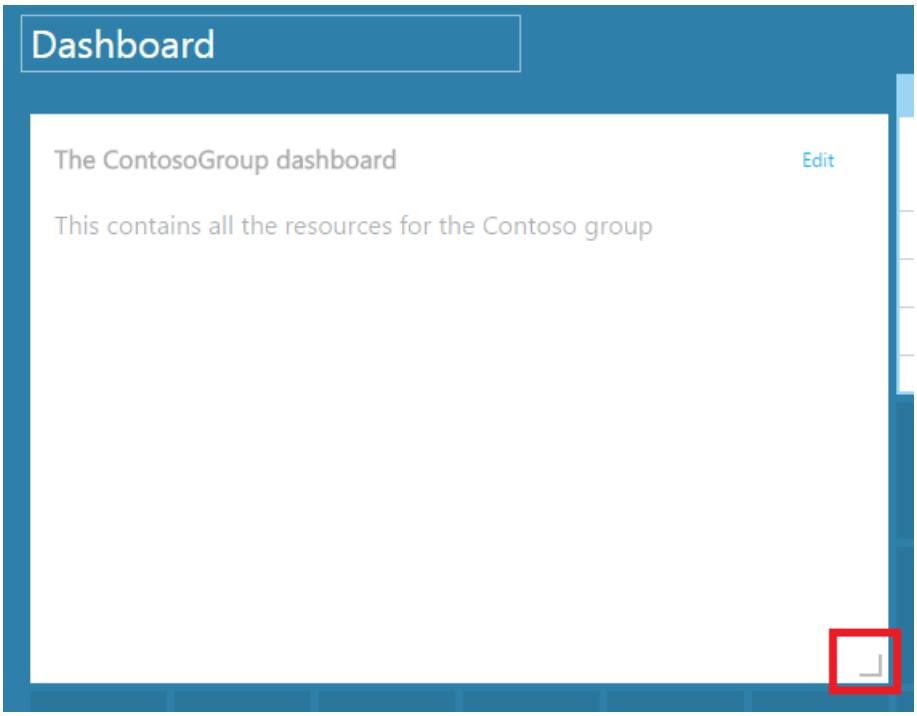
1. **Unpin from dashboard** – removes the tile from the dashboard
2. **Customize** – enters customize mode



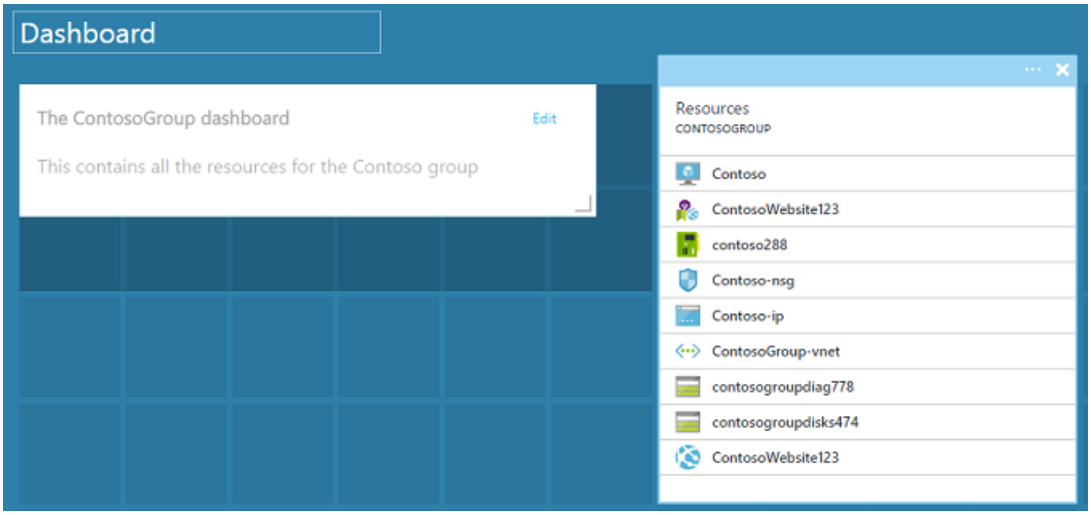

By selecting customize, you can resize and reorder tiles. To resize a tile, select the new size from the contextual menu, as shown in the following image.




Or, if the tile supports any size, you can drag the bottom right-hand corner to the desired size.



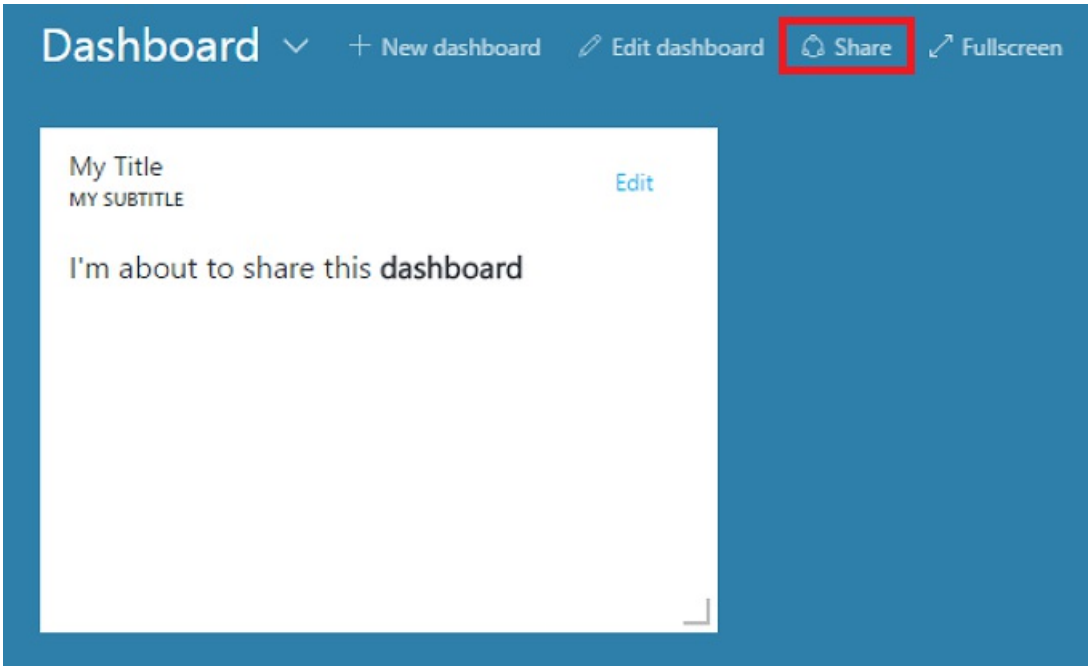

After resizing tiles, view the dashboard.




Once you are finished customizing a dashboard, simply select the **Done customizing** to exit customize mode or right-click and select **Done customizing** from the context menu.

# Publish a dashboard and manage access control

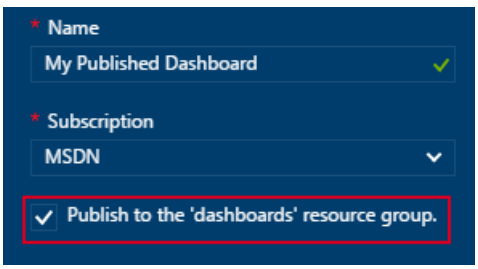
When you create a dashboard, it is private by default, which means you are the only person who can see it. To make it visible to others, use the **Share** button that appears alongside the other dashboard commands.



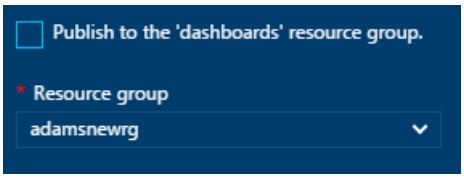

You are asked to choose a subscription and resource group for your dashboard to be published to. To seamlessly integrate dashboards into the ecosystem, we've implemented shared dashboards as Azure resources (so you can't share by typing an email address). Access to the information displayed by most of the tiles in the portal are governed by Azure Role Based Access Control. From an access control perspective, shared dashboards are no different from a virtual machine or a storage account.

Let's say you have an Azure subscription and members of your team have been assigned the roles of **owner**, **contributor**, or **reader** of the subscription. Users who are owners or contributors are able to list, view, create, modify, or delete dashboards within that subscription. Users who are readers are able to list and view dashboards, but cannot modify or delete them. Users with reader access are able to make local edits to a shared dashboard, but are not able to publish those changes back to the server. However, they can make a private copy of the dashboard for their own use. As always, individual tiles on the dashboard enforce their own access control rules based on the resources they correspond to.

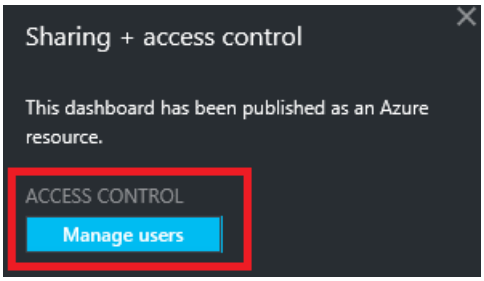
For convenience, the portal's publishing experience guides you towards a pattern where you place dashboards in a resource group called **dashboards**.




You can also choose to publish a dashboard to a particular resource group. The access control for that dashboard matches the access control for the resource group. Users that can manage the resources in that resource group also have access to the dashboards.




After your dashboard is published, the **Sharing + access** control pane will refresh and show you information about the published dashboard, including a link to manage user access to the dashboard. This link launches the standard Role Based Access Control blade used to manage access for any Azure resource. You can always get back to this view by selecting **Share**.



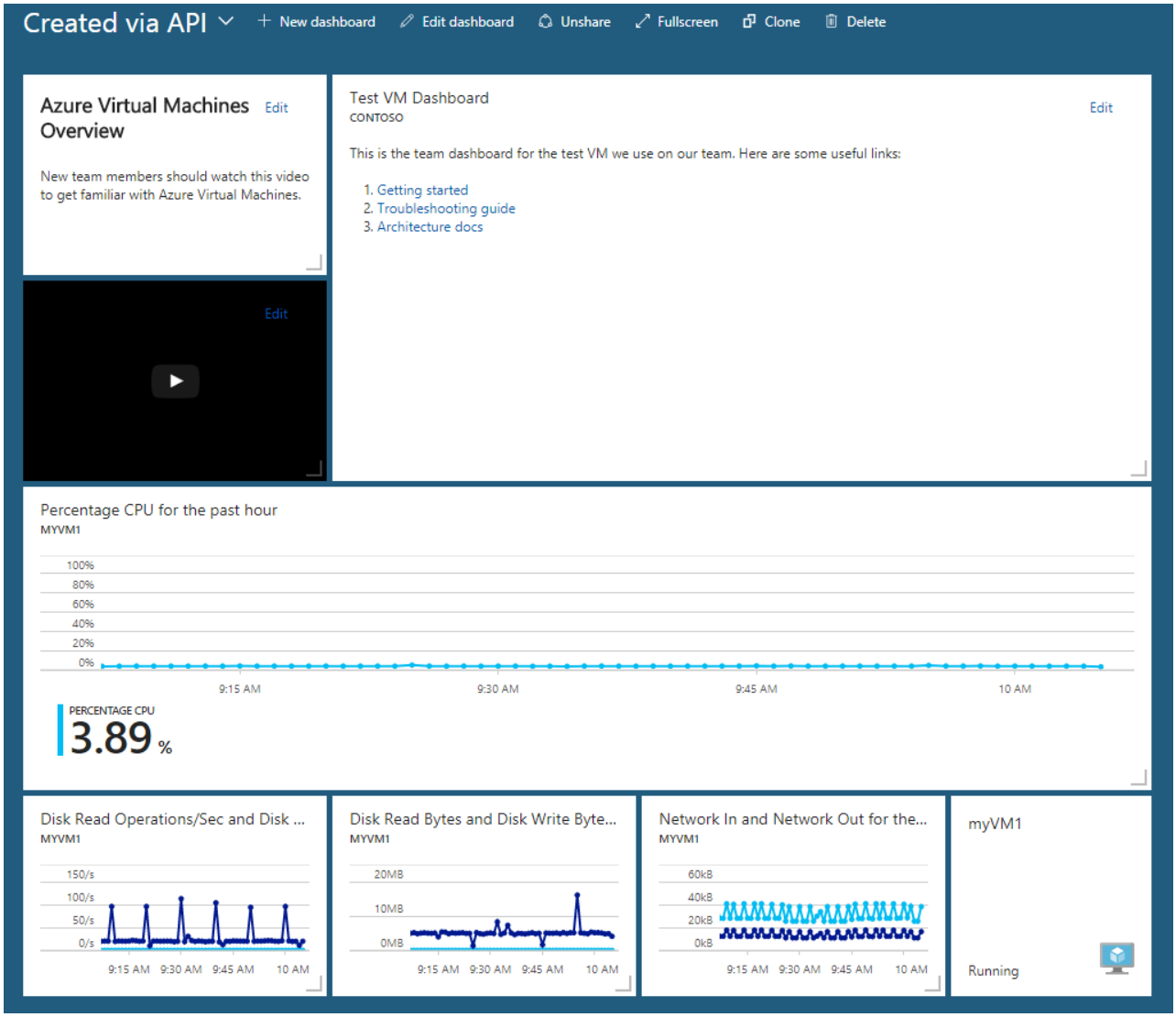

## Next steps

- To manage resources, see Manage Azure resources through portal.
- To deploy resources, see Deploy resources with Resource Manager templates and Azure portal.

# Programmatically create Azure Dashboards

12/11/2017 • 9 min to read • Edit Online

This document walks through the process of programmatically creating and publishing Azure dashboards. The dashboard shown below is referenced throughout the document.




## Overview

Shared dashboards in Azure are resources just like virtual machines and storage accounts. Therefore, they can be managed programmatically via the Azure Resource Manager REST APIs, the Azure CLI, Azure PowerShell commands, and many Azure portal features build on top of these APIs to make resource management easier.

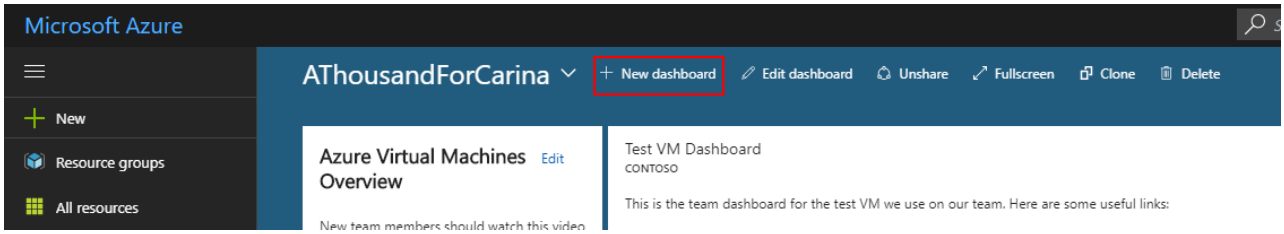
Each of these APIs and tools offers ways to create, list, retrieve, modify, and delete resources. Since dashboards are resources, you can pick your favorite API / tool to use.

Regardless of which tool you use, you need to construct a JSON representation of your dashboard object before you can call any resource creation API. This object contains information about the parts (a.k.a. tiles) on the dashboard. It includes sizes, positions, resources they are bound to, and any user customizations.

The most practical way to build up this JSON document is to use the portal to interactively add and position your tiles. You then export the JSON. Finally, you create a template from the result for later use in scripts, programs, and deployment tools.

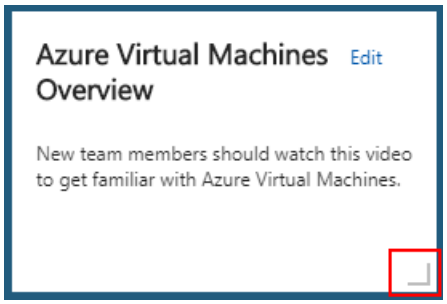
# Create a dashboard

To create a new dashboard, use the New dashboard command on the portal's main screen.

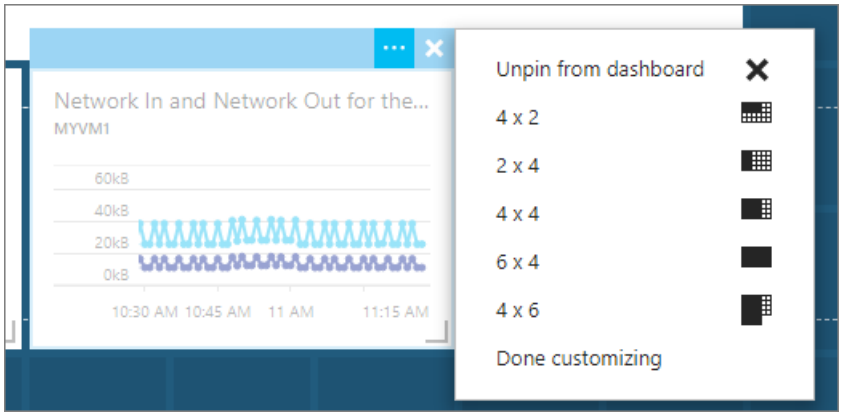



You can then use the tile gallery to find and add tiles. Tiles are added by dragging and dropping them. Some tiles support resizing via a drag handle, while others support fixes sizes that can be seen in their context menu.

**Drag handle**

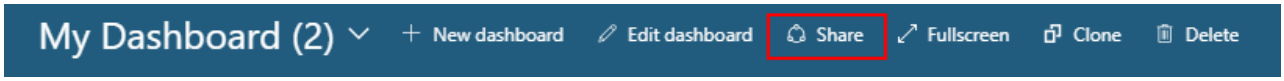



**Fixed sizes via context menu**

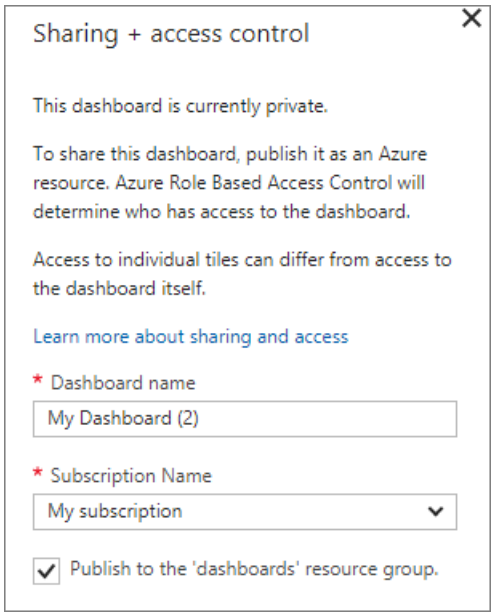



# Share the dashboard

After you have configured the dashboard to your liking the next steps are to publish the dashboard (using the Share command) and then use the resource explorer to fetch the JSON.

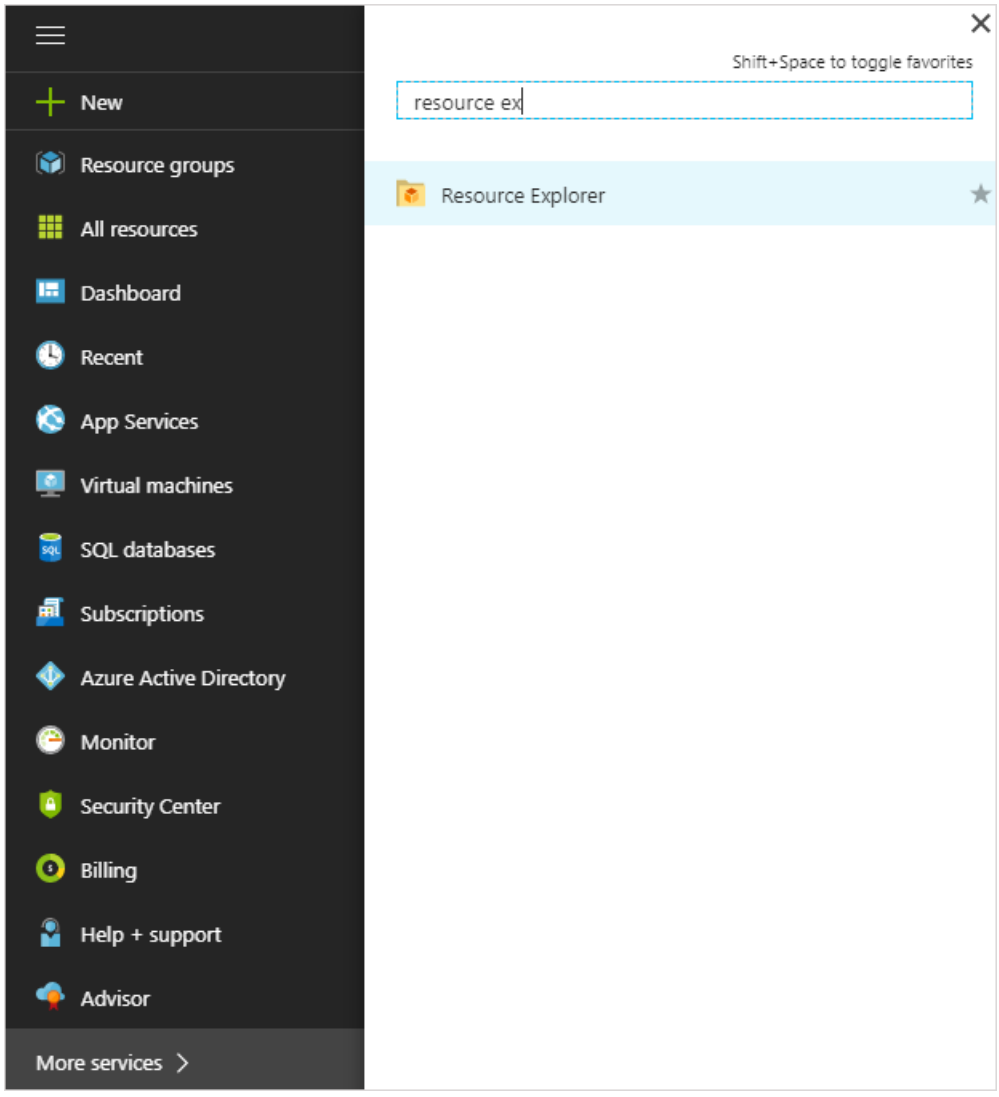



Clicking the Share command shows a dialog that asks you to choose which subscription and resource group to publish to. Keep in mind that you must have write access to the subscription and resource group that you choose.

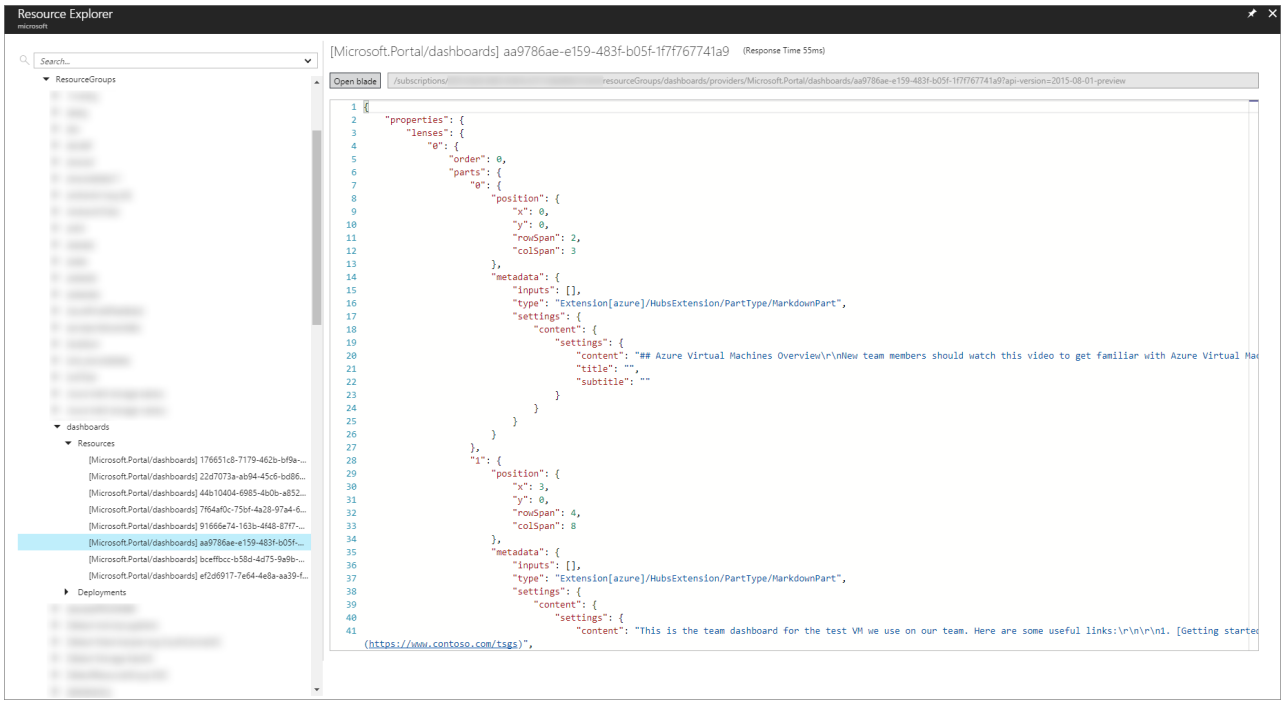



## Fetch the JSON representation of the dashboard

Publishing only takes a few seconds. When it's done, the next step is to go to the Resource Explorer to fetch the JSON.




From the resource explorer, navigate to the subscription and resource group that you chose. Next, click on the newly published dashboard resource to reveal the JSON.




# Create a template from the JSON

The next step is to create a template from this JSON so that it can be reused programmatically with the appropriate resource management APIs, command-line tools, or within the portal.

It is not necessary to fully understand the dashboard JSON structure to create a template. In most cases, you want to preserve the structure and configuration of each tile, and then parameterize the set of Azure resources they're pointing to. Look at your exported JSON dashboard and find all occurrences of Azure resource Ids. Our example dashboard has multiple tiles that all point at a single Azure virtual machine. That's because our dashboard only looks at this single resource. If you search the sample json (included at the end of the document) for "/subscriptions", you find several occurrences of this Id.

```
/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1
```

To publish this dashboard for any virtual machine in the future you need to parameterize every occurrence of this string within the JSON.

There are two flavors of APIs that create resources in Azure. Imperative APIs that create one resource at a time, and a template-based deployment system that can orchestrate the creation of multiple, dependent resources with a single API call. The latter natively supports parameterization and templating so we use it for our example.

# Programmatically create a dashboard from your template using a template deployment

Azure offers the ability to orchestrate the deployment of multiple resources. You create a deployment template that expresses the set of resources to deploy as well as the relationships between them. The JSON format of each resource is the same as if you were creating them one by one. The difference is that the template language adds a few concepts like variables, parameters, basic functions, and more. This extended syntax is only supported in the context of a template deployment and does not work if used with the imperative APIs discussed earlier.

If you're going this route, then parameterization should be done using the template's parameter syntax. You replace all instances of the resource id we found earlier as shown here.

**Example JSON property with hard-coded resource Id**

```
id: "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
```

**Example JSON property converted to a parameterized version based on template parameters**

```
id: "[resourceId(parameters('virtualMachineResourceGroup'), 'Microsoft.Compute/virtualMachines',
parameters('virtualMachineName'))]"
```

You also need to declare some required template metadata and the parameters at the top of the json template like this:

```
{
    "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "virtualMachineName": {
            "type": "string"
        },
        "virtualMachineResourceGroup": {
            "type": "string"
        },
        "dashboardName": {
            "type": "string"
        }
    },
    "variables": {},

    ... rest of template omitted ...
```

**You can see the full, working template at the end of this document.**

Once you have crafted your template you can deploy it using the REST APIs, PowerShell, The Azure CLI, or the portal's template deployment page.

Here are are two versions of our example dashboard JSON. The first is the version that we exported from the portal that was already bound to a resource. The second is the template version that can be programmatically bound to any VM and deployed using Azure Resource Manager.

# JSON representation of our example dashboard (before templating)

This is what you can expect to see if you follow the earlier instructions to fetch the JSON representation of a dashboard that is already deployed. Note the hard-coded resource identifiers that show that this dashboard is pointing at a specific Azure virtual machine.

```
{
    "properties": {
        "lenses": {
            "0": {
                "order": 0,
                "parts": {
                    "0": {
                        "position": {
                            "x": 0,
                            "y": 0,
                            "rowSpan": 2,
                            "colSpan": 3
                        },
                        "metadata": {
                            "inputs": [],
                            "type": "Extension[azure]/HubsExtension/PartType/MarkdownPart",
                            "settings": {
                                "content": {
                                    "settings": {
                                        "content": "## Azure Virtual Machines Overview\r\nNew team members
should watch this video to get familiar with Azure Virtual Machines.",
```

```
                                        "title": "",
                                        "subtitle": ""
                                    }
                                }
                            }
                        }
                    },
                    "1": {
                        "position": {
                            "x": 3,
                            "y": 0,
                            "rowSpan": 4,
                            "colSpan": 8
                        },
                        "metadata": {
                            "inputs": [],
                            "type": "Extension[azure]/HubsExtension/PartType/MarkdownPart",
                            "settings": {
                                "content": {
                                    "settings": {
                                        "content": "This is the team dashboard for the test VM we use on our
team. Here are some useful links:\r\n\r\n1. [Getting started](https://www.contoso.com/tsgs)\r\n1.
[Troubleshooting guide](https://www.contoso.com/tsgs)\r\n1. [Architecture docs](https://www.contoso.com/tsgs)",
                                        "title": "Test VM Dashboard",
                                        "subtitle": "Contoso"
                                    }
                                }
                            }
                        }
                    },
                    "2": {
                        "position": {
                            "x": 0,
                            "y": 2,
                            "rowSpan": 2,
                            "colSpan": 3
                        },
                        "metadata": {
                            "inputs": [],
                            "type": "Extension[azure]/HubsExtension/PartType/VideoPart",
                            "settings": {
                                "content": {
                                    "settings": {
                                        "title": "",
                                        "subtitle": "",
                                        "src": "https://www.youtube.com/watch?
v=YcylDIiKaSU&list=PLLasX02E8BPCsnETz0XAMfpLR1LIBqpgs&index=4",
                                        "autoplay": false
                                    }
                                }
                            }
                        }
                    },
                    "3": {
                        "position": {
                            "x": 0,
                            "y": 4,
                            "rowSpan": 3,
                            "colSpan": 11
                        },
                        "metadata": {
                            "inputs": [
                                {
                                    "name": "queryInputs",
                                    "value": {
                                        "timespan": {
                                            "duration": "PT1H",
                                            "start": null,
                                            "end": null
```

```
                                        "end": null
                                    },
                                    "id": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1",
                                    "chartType": 0,
                                    "metrics": [
                                        {
                                            "name": "Percentage CPU",
                                            "resourceId": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                                        }
                                    ]
                                }
                            }
                        ],
                        "type": "Extension/Microsoft_Azure_Monitoring/PartType/MetricsChartPart",
                        "settings": {}
                    }
                },
                "4": {
                    "position": {
                        "x": 0,
                        "y": 7,
                        "rowSpan": 2,
                        "colSpan": 3
                    },
                    "metadata": {
                        "inputs": [
                            {
                                "name": "queryInputs",
                                "value": {
                                    "timespan": {
                                        "duration": "PT1H",
                                        "start": null,
                                        "end": null
                                    },
                                    "id": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1",
                                    "chartType": 0,
                                    "metrics": [
                                        {
                                            "name": "Disk Read Operations/Sec",
                                            "resourceId": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                                        },
                                        {
                                            "name": "Disk Write Operations/Sec",
                                            "resourceId": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                                        }
                                    ]
                                }
                            }
                        ],
                        "type": "Extension/Microsoft_Azure_Monitoring/PartType/MetricsChartPart",
                        "settings": {}
                    }
                },
                "5": {
                    "position": {
                        "x": 3,
                        "y": 7,
                        "rowSpan": 2,
                        "colSpan": 3
                    },
                    "metadata": {
                        "inputs": [
                            {
                                "name": "queryInputs",
```

```
                                "value": {
                                    "timespan": {
                                        "duration": "PT1H",
                                        "start": null,
                                        "end": null
                                    },
                                    "id": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1",
                                    "chartType": 0,
                                    "metrics": [
                                        {
                                            "name": "Disk Read Bytes",
                                            "resourceId": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                                        },
                                        {
                                            "name": "Disk Write Bytes",
                                            "resourceId": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                                        }
                                    ]
                                }
                            }
                        ],
                        "type": "Extension/Microsoft_Azure_Monitoring/PartType/MetricsChartPart",
                        "settings": {}
                    }
                },
                "6": {
                    "position": {
                        "x": 6,
                        "y": 7,
                        "rowSpan": 2,
                        "colSpan": 3
                    },
                    "metadata": {
                        "inputs": [
                            {
                                "name": "queryInputs",
                                "value": {
                                    "timespan": {
                                        "duration": "PT1H",
                                        "start": null,
                                        "end": null
                                    },
                                    "id": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1",
                                    "chartType": 0,
                                    "metrics": [
                                        {
                                            "name": "Network In",
                                            "resourceId": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                                        },
                                        {
                                            "name": "Network Out",
                                            "resourceId": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                                        }
                                    ]
                                }
                            }
                        ],
                        "type": "Extension/Microsoft_Azure_Monitoring/PartType/MetricsChartPart",
                        "settings": {}
                    }
                },
                "7": {
                    "position": {
```

```
                            "x": 9,
                            "y": 7,
                            "rowSpan": 2,
                            "colSpan": 2
                        },
                        "metadata": {
                            "inputs": [
                                {
                                    "name": "id",
                                    "value": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/contoso/providers/Microsoft.Compute/virtualMachines/myVM1"
                                }
                            ],
                            "type": "Extension/Microsoft_Azure_Compute/PartType/VirtualMachinePart",
                            "asset": {
                                "idInputName": "id",
                                "type": "VirtualMachine"
                            },
                            "defaultMenuItemId": "overview"
                        }
                    }
                }
            }
        },
        "metadata": { }
    },
    "id": "/subscriptions/6531c8c8-df32-4254-d717-
b6e983273e5d/resourceGroups/dashboards/providers/Microsoft.Portal/dashboards/aa9786ae-e159-483f-b05f-
1f7f767741a9",
    "name": "aa9786ae-e159-483f-b05f-1f7f767741a9",
    "type": "Microsoft.Portal/dashboards",
    "location": "eastasia",
    "tags": {
        "hidden-title": "Created via API"
    }
}
```

**Template representation of our example dashboard**

The template version of the dashboard has defined three parameters called **virtualMachineName**, **virtualMachineResourceGroup**, and **dashboardName**. The parameters let you point this dashboard at a different Azure virtual machine every time you deploy. The parameterized ids are highlighted to show that this dashboard can be programmatically configured and deployed to point to any Azure virtual machine. The easiest way to test this feature is to copy the following template and paste it into the Azure portal's template deployment page.

This example deploys a dashboard by itself, but the template language lets you deploy multiple resources, and bundle one or more dashboards along side them.

```
{
    "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "virtualMachineName": {
            "type": "string"
        },
        "virtualMachineResourceGroup": {
            "type": "string"
        },
        "dashboardName": {
            "type": "string"
        }
    },
    "variables": {},
    "resources": [
        {
```

```
{
    "properties": {
        "lenses": {
            "0": {
                "order": 0,
                "parts": {
                    "0": {
                        "position": {
                            "x": 0,
                            "y": 0,
                            "rowSpan": 2,
                            "colSpan": 3
                        },
                        "metadata": {
                            "inputs": [],
                            "type": "Extension[azure]/HubsExtension/PartType/MarkdownPart",
                            "settings": {
                                "content": {
                                    "settings": {
                                        "content": "## Azure Virtual Machines Overview\r\nNew team
members should watch this video to get familiar with Azure Virtual Machines.",
                                        "title": "",
                                        "subtitle": ""
                                    }
                                }
                            }
                        }
                    },
                    "1": {
                        "position": {
                            "x": 3,
                            "y": 0,
                            "rowSpan": 4,
                            "colSpan": 8
                        },
                        "metadata": {
                            "inputs": [],
                            "type": "Extension[azure]/HubsExtension/PartType/MarkdownPart",
                            "settings": {
                                "content": {
                                    "settings": {
                                        "content": "This is the team dashboard for the test VM we use
on our team. Here are some useful links:\r\n\r\n1. [Getting started](https://www.contoso.com/tsgs)\r\n1.
[Troubleshooting guide](https://www.contoso.com/tsgs)\r\n1. [Architecture docs](https://www.contoso.com/tsgs)",
                                        "title": "Test VM Dashboard",
                                        "subtitle": "Contoso"
                                    }
                                }
                            }
                        }
                    },
                    "2": {
                        "position": {
                            "x": 0,
                            "y": 2,
                            "rowSpan": 2,
                            "colSpan": 3
                        },
                        "metadata": {
                            "inputs": [],
                            "type": "Extension[azure]/HubsExtension/PartType/VideoPart",
                            "settings": {
                                "content": {
                                    "settings": {
                                        "title": "",
                                        "subtitle": "",
                                        "src": "https://www.youtube.com/watch?
v=YcylDIiKaSU&list=PLLasX02E8BPCsnETz0XAMfpLR1LIBqpgs&index=4",
                                        "autoplay": false
                                    }
```

```
                                    }
                                }
                            }
                        }
                    },
                    "3": {
                        "position": {
                            "x": 0,
                            "y": 4,
                            "rowSpan": 3,
                            "colSpan": 11
                        },
                        "metadata": {
                            "inputs": [
                                {
                                    "name": "queryInputs",
                                    "value": {
                                        "timespan": {
                                            "duration": "PT1H",
                                            "start": null,
                                            "end": null
                                        },
                                        "id": "[resourceId(parameters('virtualMachineResourceGroup'),
'Microsoft.Compute/virtualMachines', parameters('virtualMachineName'))]",
                                        "chartType": 0,
                                        "metrics": [
                                            {
                                                "name": "Percentage CPU",
                                                "resourceId": "
[resourceId(parameters('virtualMachineResourceGroup'), 'Microsoft.Compute/virtualMachines',
parameters('virtualMachineName'))]"
                                            }
                                        ]
                                    }
                                }
                            ],
                            "type": "Extension/Microsoft_Azure_Monitoring/PartType/MetricsChartPart",
                            "settings": {}
                        }
                    },
                    "4": {
                        "position": {
                            "x": 0,
                            "y": 7,
                            "rowSpan": 2,
                            "colSpan": 3
                        },
                        "metadata": {
                            "inputs": [
                                {
                                    "name": "queryInputs",
                                    "value": {
                                        "timespan": {
                                            "duration": "PT1H",
                                            "start": null,
                                            "end": null
                                        },
                                        "id": "[resourceId(parameters('virtualMachineResourceGroup'),
'Microsoft.Compute/virtualMachines', parameters('virtualMachineName'))]",
                                        "chartType": 0,
                                        "metrics": [
                                            {
                                                "name": "Disk Read Operations/Sec",
                                                "resourceId": "
[resourceId(parameters('virtualMachineResourceGroup'), 'Microsoft.Compute/virtualMachines',
parameters('virtualMachineName'))]"
                                            },
                                            {
                                                "name": "Disk Write Operations/Sec",
```

```
                                                    "resourceId": "
[resourceId(parameters('virtualMachineResourceGroup'), 'Microsoft.Compute/virtualMachines',
parameters('virtualMachineName'))]"
                                                }
                                            ]
                                        }
                                    }
                                ],
                                "type": "Extension/Microsoft_Azure_Monitoring/PartType/MetricsChartPart",
                                "settings": {}
                            }
                        },
                        "5": {
                            "position": {
                                "x": 3,
                                "y": 7,
                                "rowSpan": 2,
                                "colSpan": 3
                            },
                            "metadata": {
                                "inputs": [
                                    {
                                        "name": "queryInputs",
                                        "value": {
                                            "timespan": {
                                                "duration": "PT1H",
                                                "start": null,
                                                "end": null
                                            },
                                            "id": "[resourceId(parameters('virtualMachineResourceGroup'),
'Microsoft.Compute/virtualMachines', parameters('virtualMachineName'))]",
                                            "chartType": 0,
                                            "metrics": [
                                                {
                                                    "name": "Disk Read Bytes",
                                                    "resourceId": "
[resourceId(parameters('virtualMachineResourceGroup'), 'Microsoft.Compute/virtualMachines',
parameters('virtualMachineName'))]"
                                                },
                                                {
                                                    "name": "Disk Write Bytes",
                                                    "resourceId": "
[resourceId(parameters('virtualMachineResourceGroup'), 'Microsoft.Compute/virtualMachines',
parameters('virtualMachineName'))]"
                                                }
                                            ]
                                        }
                                    }
                                ],
                                "type": "Extension/Microsoft_Azure_Monitoring/PartType/MetricsChartPart",
                                "settings": {}
                            }
                        },
                        "6": {
                            "position": {
                                "x": 6,
                                "y": 7,
                                "rowSpan": 2,
                                "colSpan": 3
                            },
                            "metadata": {
                                "inputs": [
                                    {
                                        "name": "queryInputs",
                                        "value": {
                                            "timespan": {
                                                "duration": "PT1H",
                                                "start": null,
                                                "end": null
```

```
                                            },
                                            "id": "[resourceId(parameters('virtualMachineResourceGroup'),
'Microsoft.Compute/virtualMachines', parameters('virtualMachineName'))]",
                                            "chartType": 0,
                                            "metrics": [
                                                {
                                                    "name": "Network In",
                                                    "resourceId": "
[resourceId(parameters('virtualMachineResourceGroup'), 'Microsoft.Compute/virtualMachines',
parameters('virtualMachineName'))]"
                                                },
                                                {
                                                    "name": "Network Out",
                                                    "resourceId": "
[resourceId(parameters('virtualMachineResourceGroup'), 'Microsoft.Compute/virtualMachines',
parameters('virtualMachineName'))]"
                                                }
                                            ]
                                        }
                                    }
                                ],
                                "type": "Extension/Microsoft_Azure_Monitoring/PartType/MetricsChartPart",
                                "settings": {}
                            }
                        },
                        "7": {
                            "position": {
                                "x": 9,
                                "y": 7,
                                "rowSpan": 2,
                                "colSpan": 2
                            },
                            "metadata": {
                                "inputs": [
                                    {
                                        "name": "id",
                                        "value": "[resourceId(parameters('virtualMachineResourceGroup'),
'Microsoft.Compute/virtualMachines', parameters('virtualMachineName'))]"
                                    }
                                ],
                                "type": "Extension/Microsoft_Azure_Compute/PartType/VirtualMachinePart",
                                "asset": {
                                    "idInputName": "id",
                                    "type": "VirtualMachine"
                                },
                                "defaultMenuItemId": "overview"
                            }
                        }
                    }
                }
            }
        },
        "metadata": { },
        "apiVersion": "2015-08-01-preview",
        "type": "Microsoft.Portal/dashboards",
        "name": "[parameters('dashboardName')]",
        "location": "westus",
        "tags": {
            "hidden-title": "[parameters('dashboardName')]"
        }
    }
  ]
}
```

# Turn on high contrast or change the theme in the Azure portal

12/11/2017 • 1 min to read • Edit Online

If you need more contrast or you want to change the color scheme in the Azure portal, go to the portal settings to make the change.

## Turn on high contrast

1. On the top right of the Azure portal, select **Settings**.

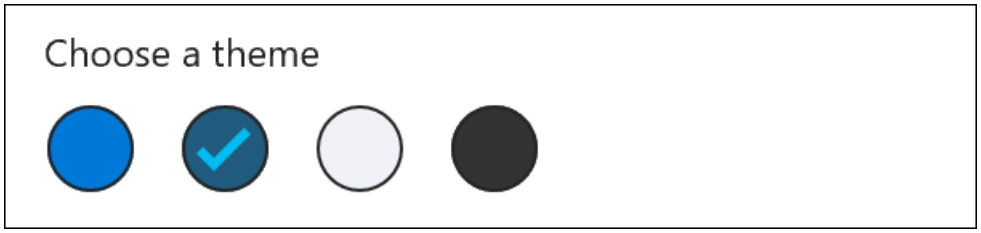
2. Choose **White** or **Black**.


3. Select **Apply**.

## Change theme

1. On the top right of the Azure portal, select **Settings**.

2. Choose a theme.




3. Select **Apply**.

## Next steps

- Keyboard shortcuts in Azure portal
- Supported browsers and devices

# Manage Azure resources through portal

11/15/2017 • 5 min to read • Edit Online

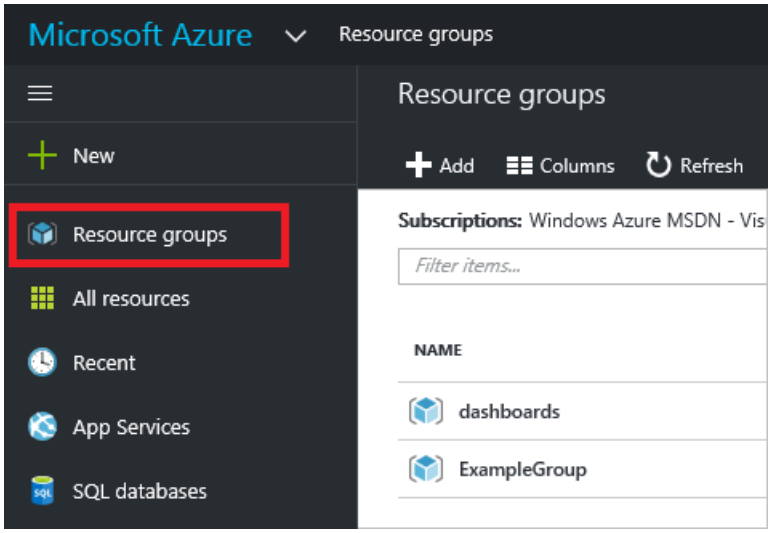
This article shows how to use the Azure portal with Azure Resource Manager to manage your Azure resources. To learn about deploying resources through the portal, see Deploy resources with Resource Manager templates and Azure portal.

## Manage resource groups

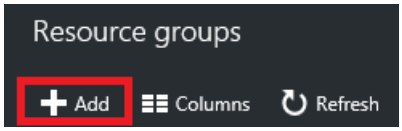
A resource group is a container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. Generally, add resources that share the same lifecycle to the same resource group so you can easily deploy, update, and delete them as a group.

The resource group stores metadata about the resources. Therefore, when you specify a location for the resource group, you are specifying where that metadata is stored. For compliance reasons, you may need to ensure that your data is stored in a particular region.

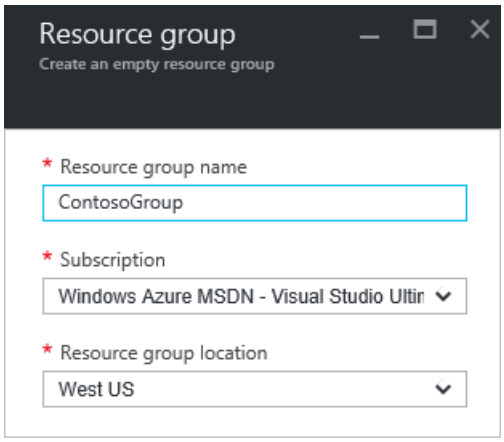
1. To see all the resource groups in your subscription, select **Resource groups**.



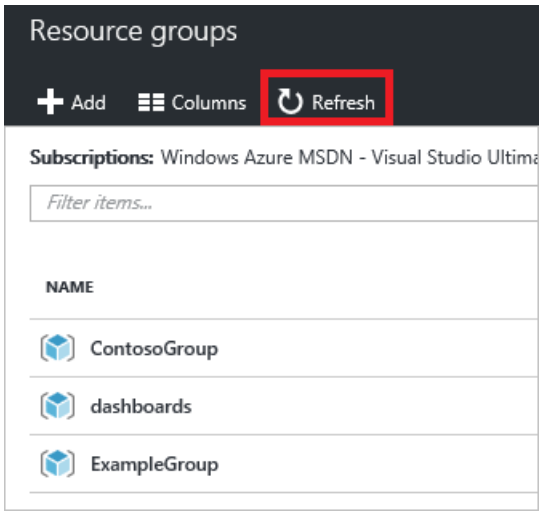

2. To create an empty resource group, select **Add**.



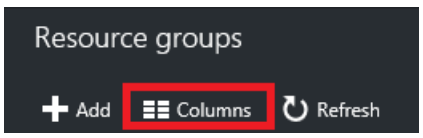

3. Provide a name and location for the new resource group. Select **Create**.



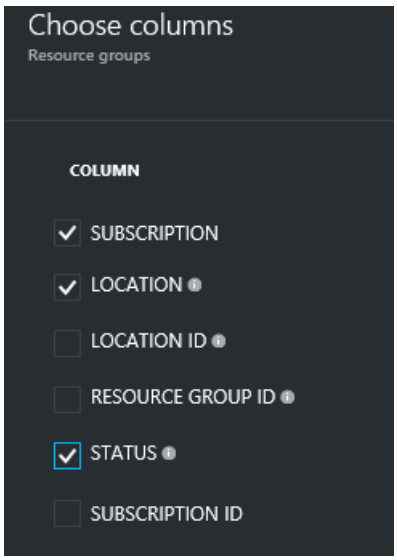

4. You may need to select **Refresh** to see the recently created resource group.



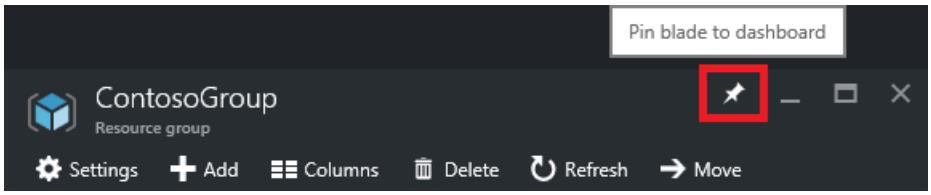

5. To customize the information displayed for your resource groups, select **Columns**.



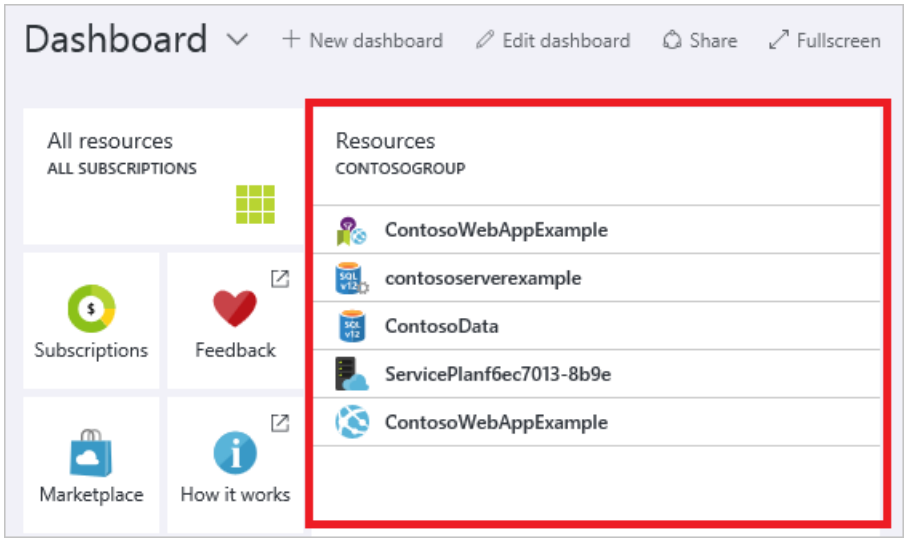

6. Select the columns to add, and then select **Update**.




7. To learn about deploying resources to your new resource group, see Deploy resources with Resource Manager templates and Azure portal.
8. For quick access to a resource group, you can pin the resource group to your dashboard.



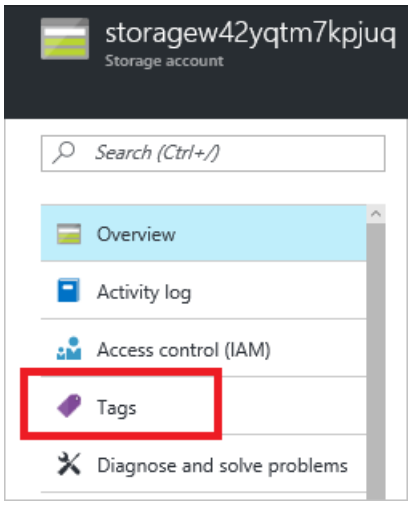

9. The dashboard displays the resource group and its resources. You can select either the resource groups or any of its resources to navigate to the item.



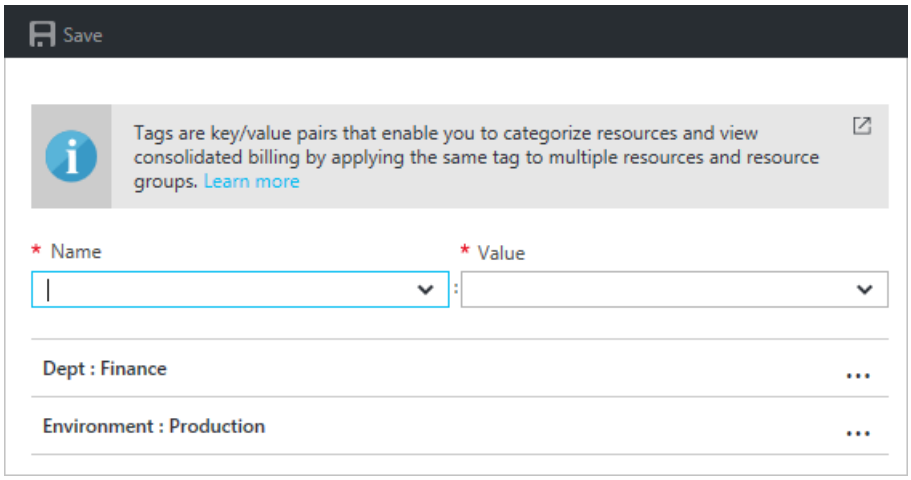

## Tag resources

You can apply tags to resource groups and resources to logically organize your assets. For information about working with tags, see Using tags to organize your Azure resources.

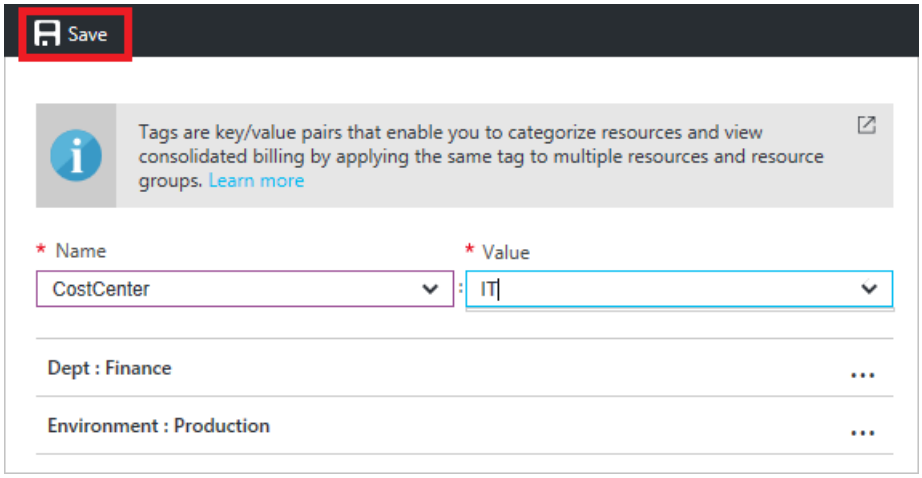
1. To view the tags for a resource or a resource group, select the **Tags** icon.



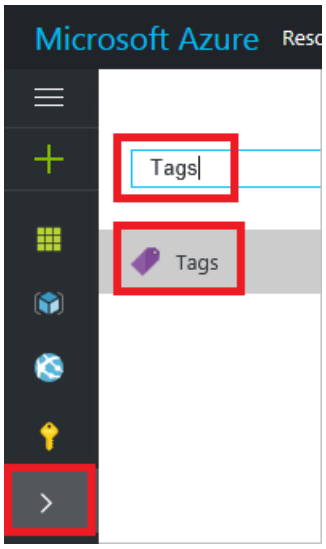

2. You see the existing tags for the resource. If you have not previously applied tags, the list is empty.



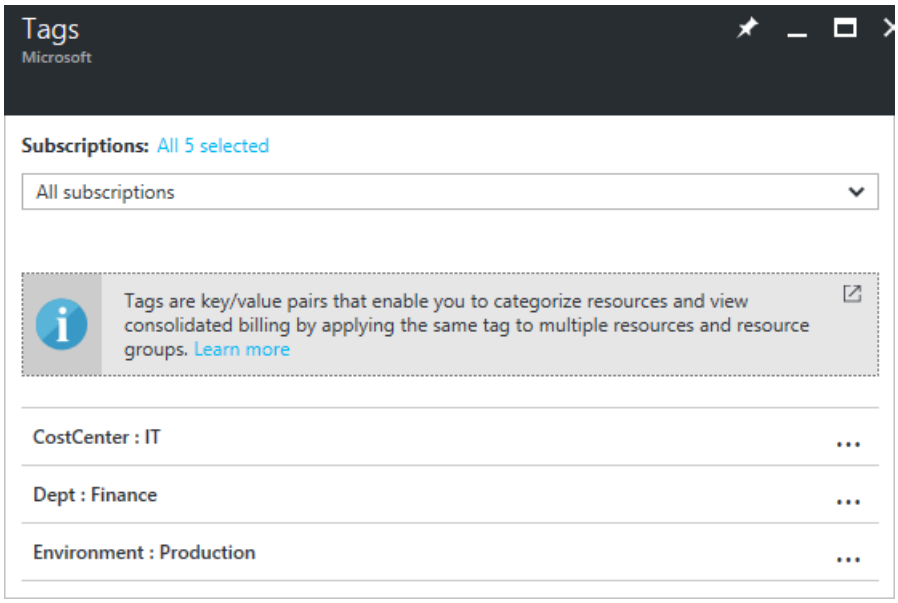

3. To add a tag, type a name and value, or select an existing one from the drop-down menu. Select **Save**.



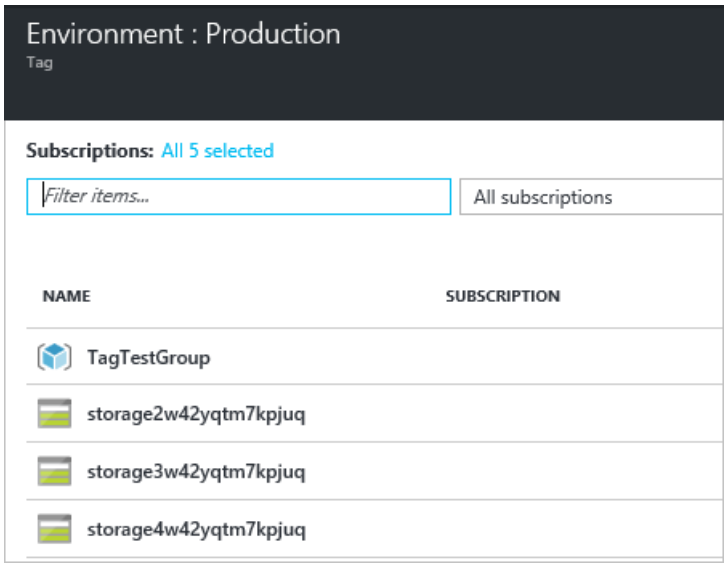

4. To view all the resources that have a tag value, select **>** (**More services**), and enter the word **Tags** into the filter text box. Select **Tags** from the available options.



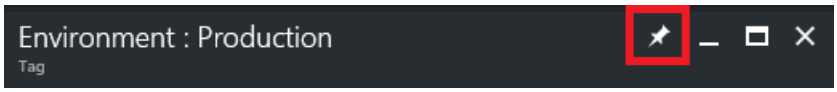

5. You see a summary of the tags in your subscriptions.



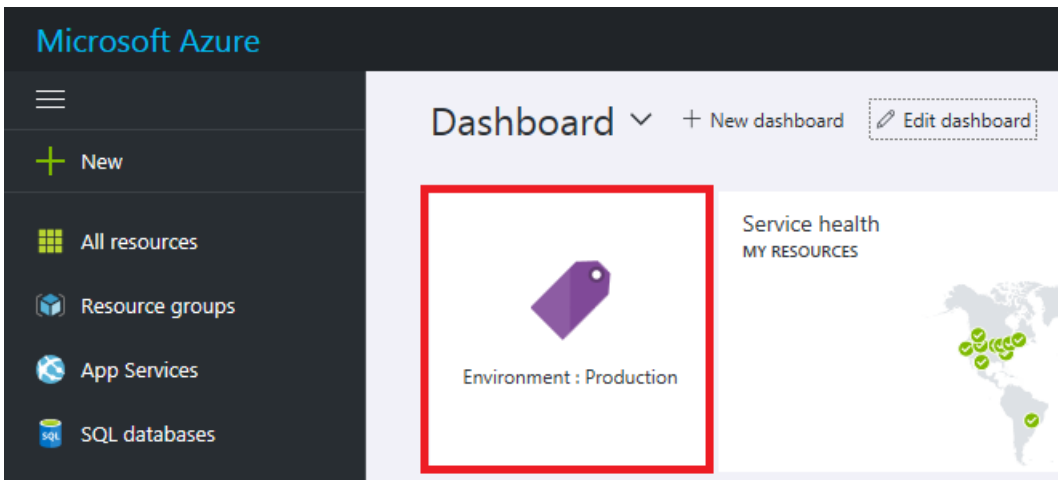

6. Select any of the tags to display the resources and resource groups with that tag.




7. Select **Pin blade to dashboard** for quick access.



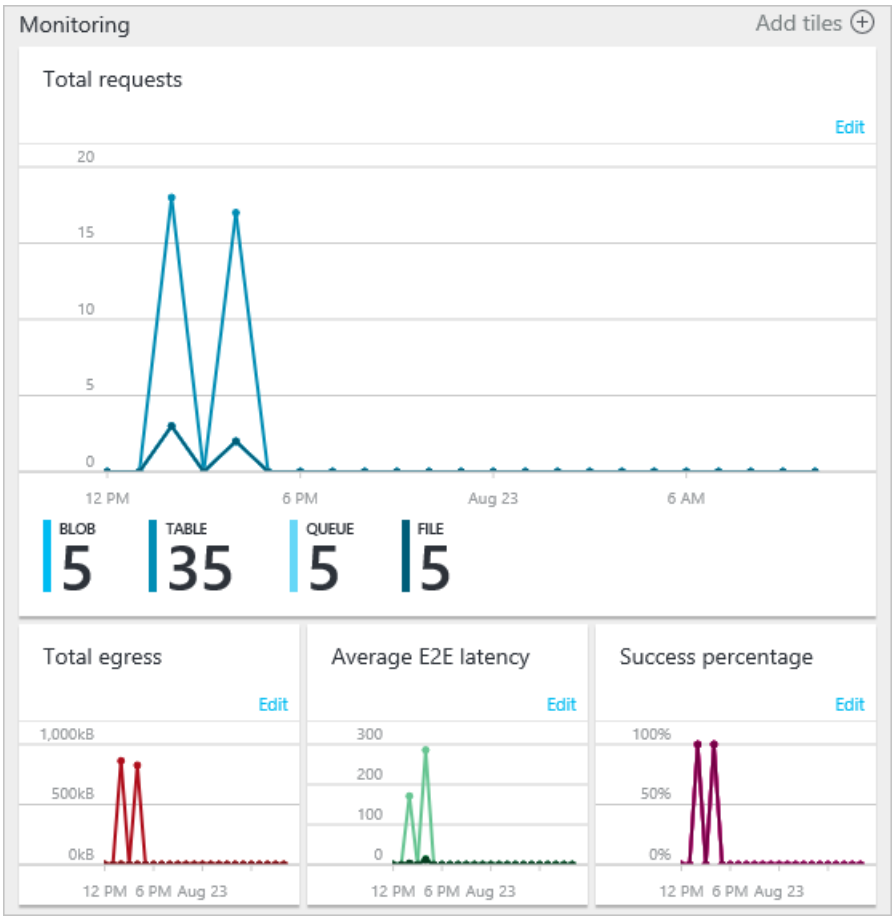

8. You can select the pinned tag from the dashboard to see the resources with that tag.



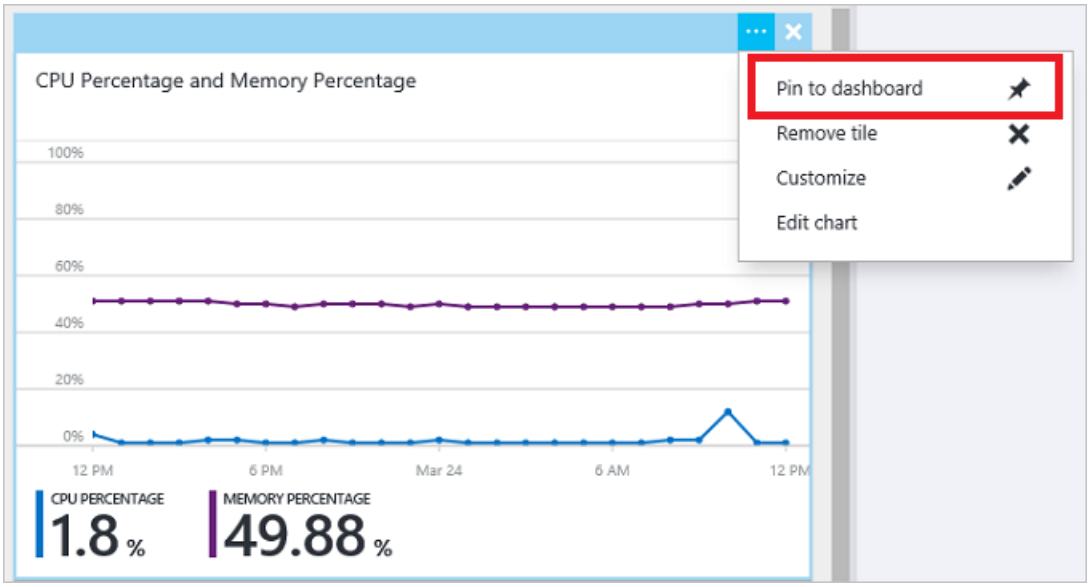

## Monitor resources

When you select a resource, the portal presents default graphs and tables for monitoring that resource type.

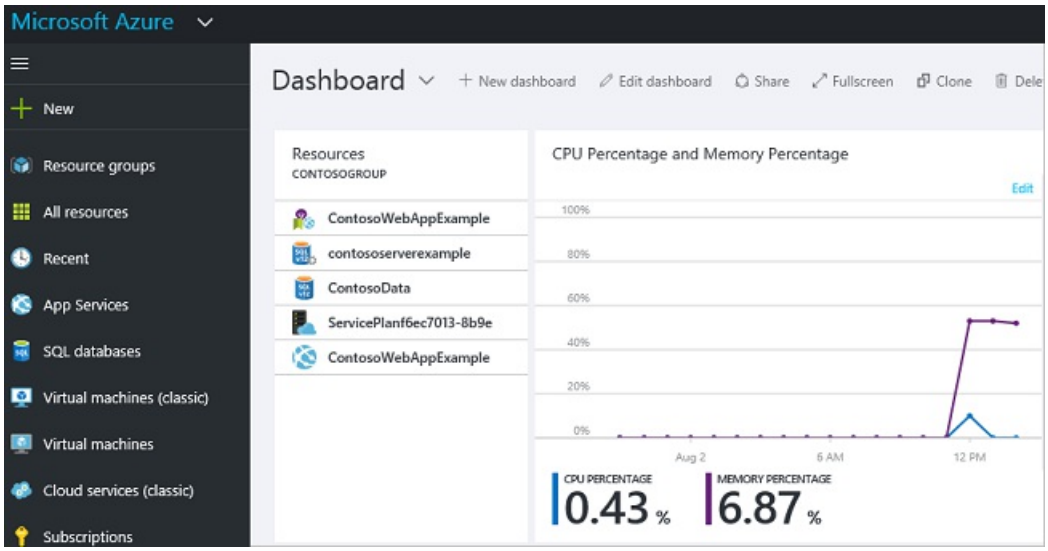
1. Select a resource and notice the **Monitoring** section. It includes graphs that are relevant to the resource type. The following image shows the default monitoring data for a storage account.



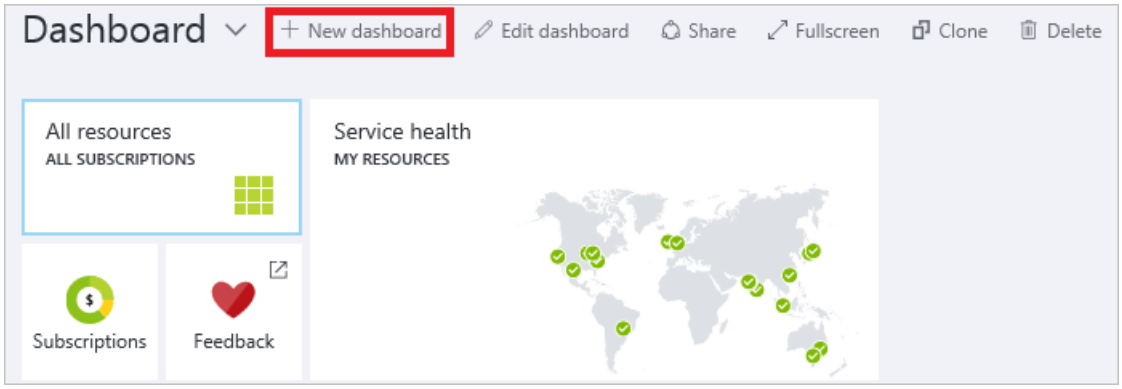

2. You can pin a section to your dashboard by selecting the ellipsis (...) above the section. You can also customize the size the section or remove it completely. The following image shows how to pin, customize, or remove the CPU and Memory section.



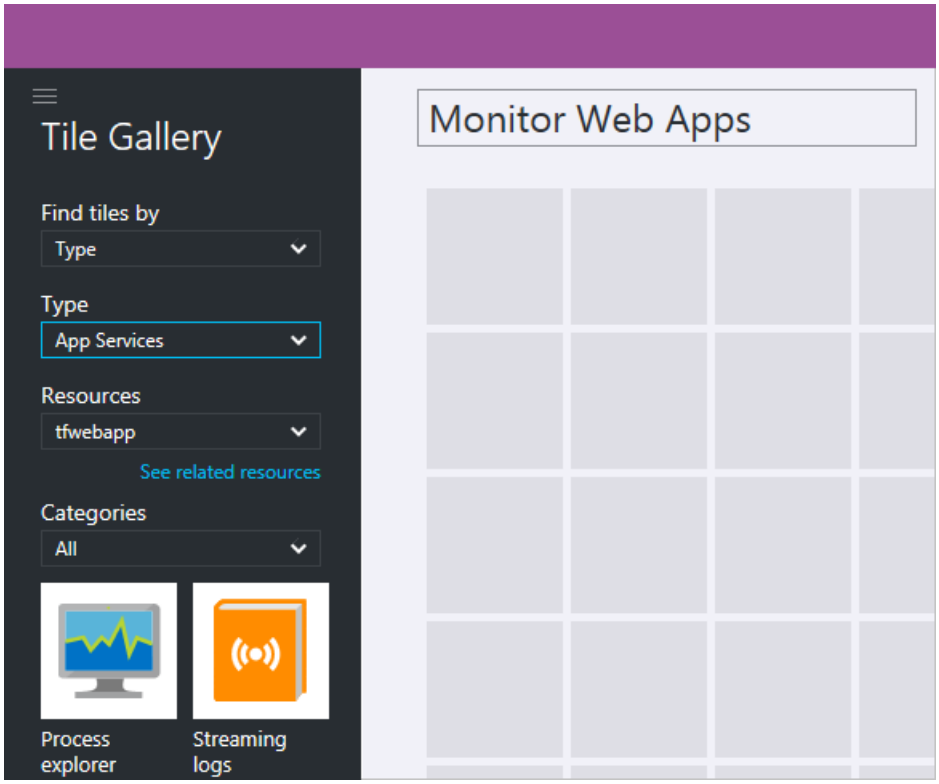

3. After pinning the section to the dashboard, you will see the summary on the dashboard. And, selecting it immediately takes you to more details about the data.




4. To completely customize the data you monitor through the portal, navigate to your default dashboard, and select **New dashboard**.



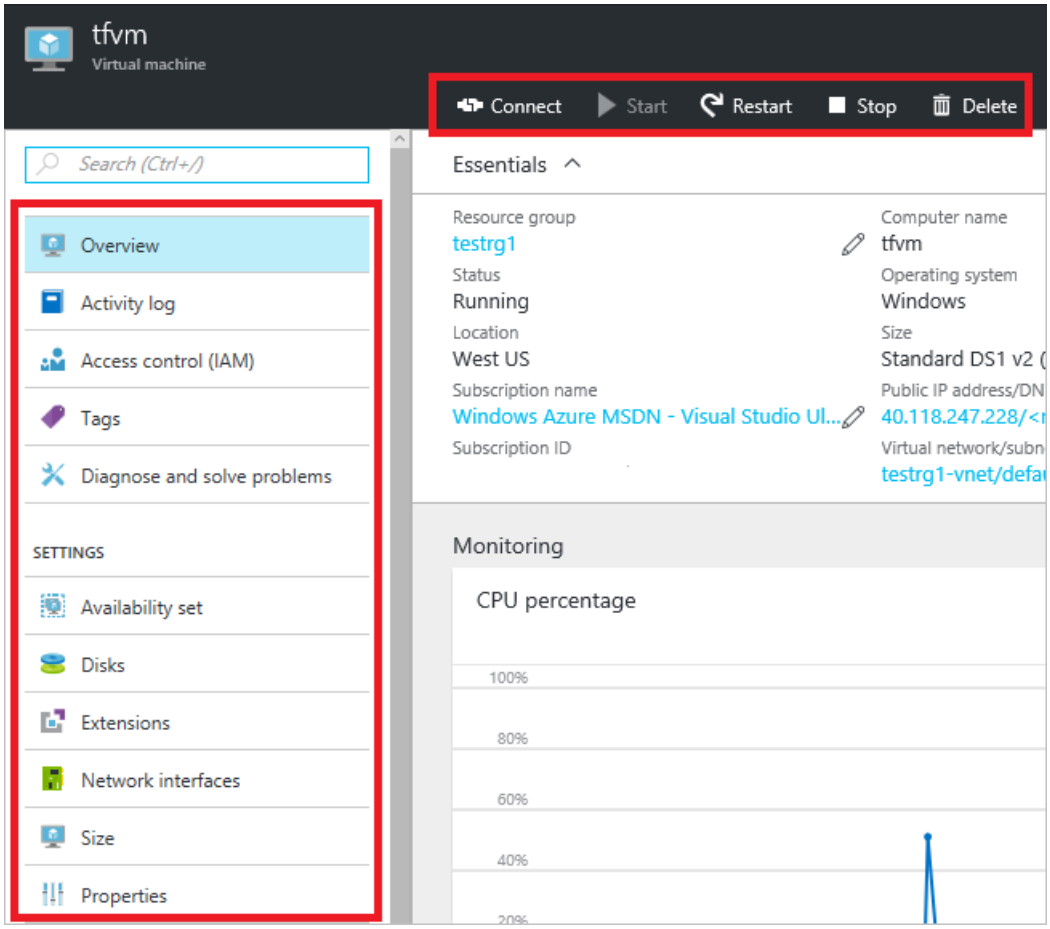

5. Give your new dashboard a name and drag tiles onto the dashboard. The tiles are filtered by different options.




To learn about working with dashboards, see Creating and sharing dashboards in the Azure portal.

## Manage resources

When viewing a resource in the portal, you see the options for managing that particular resource.




From these options, you can perform operations such as starting and stopping a virtual machine, or reconfiguring the properties of the virtual machine.

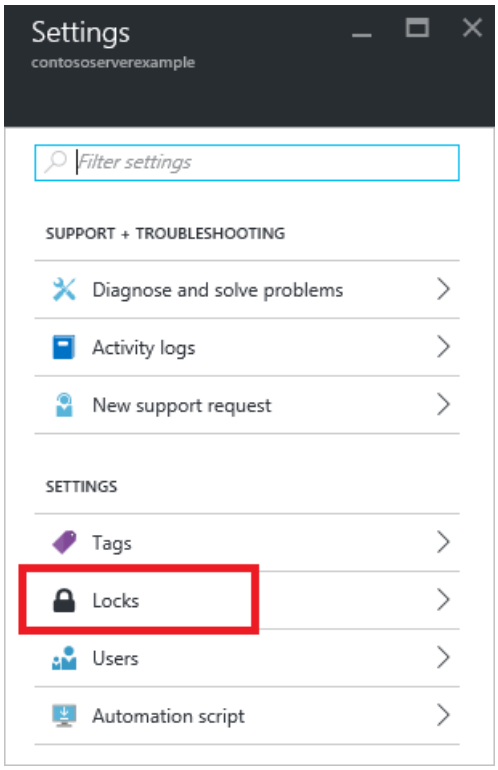
## Move resources

If you need to move resources to another resource group or another subscription, see Move resources to new resource group or subscription.

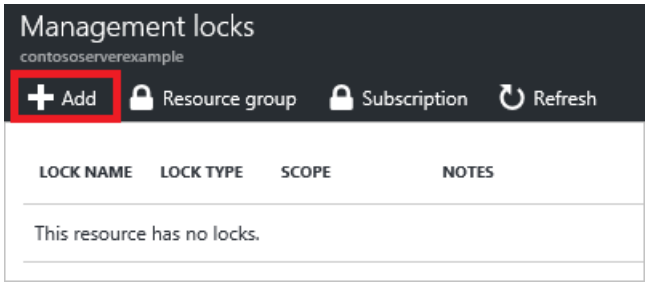
## Lock resources

You can lock a subscription, resource group, or resource to prevent other users in your organization from accidentally deleting or modifying critical resources. For more information, see Lock resources with Azure Resource Manager.

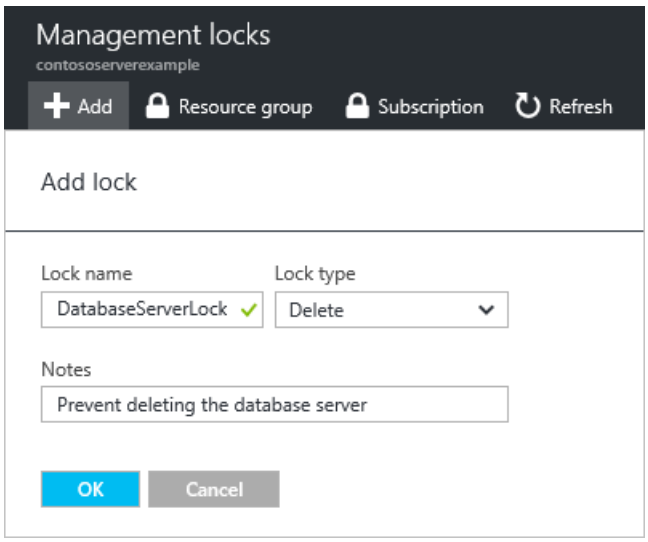
1. In the Settings blade for the resource, resource group, or subscription that you wish to lock, select **Locks**.



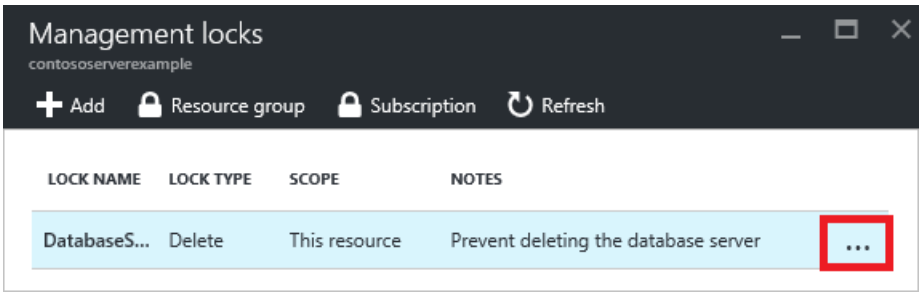

2. To add a lock, select **Add**. If you want to create a lock at a parent level, select the parent. The currently selected resource inherits the lock from the parent. For example, you could lock the resource group to apply a lock to all its resources.




3. Give the lock a name and lock level. Optionally, you can add notes that describe the lock.

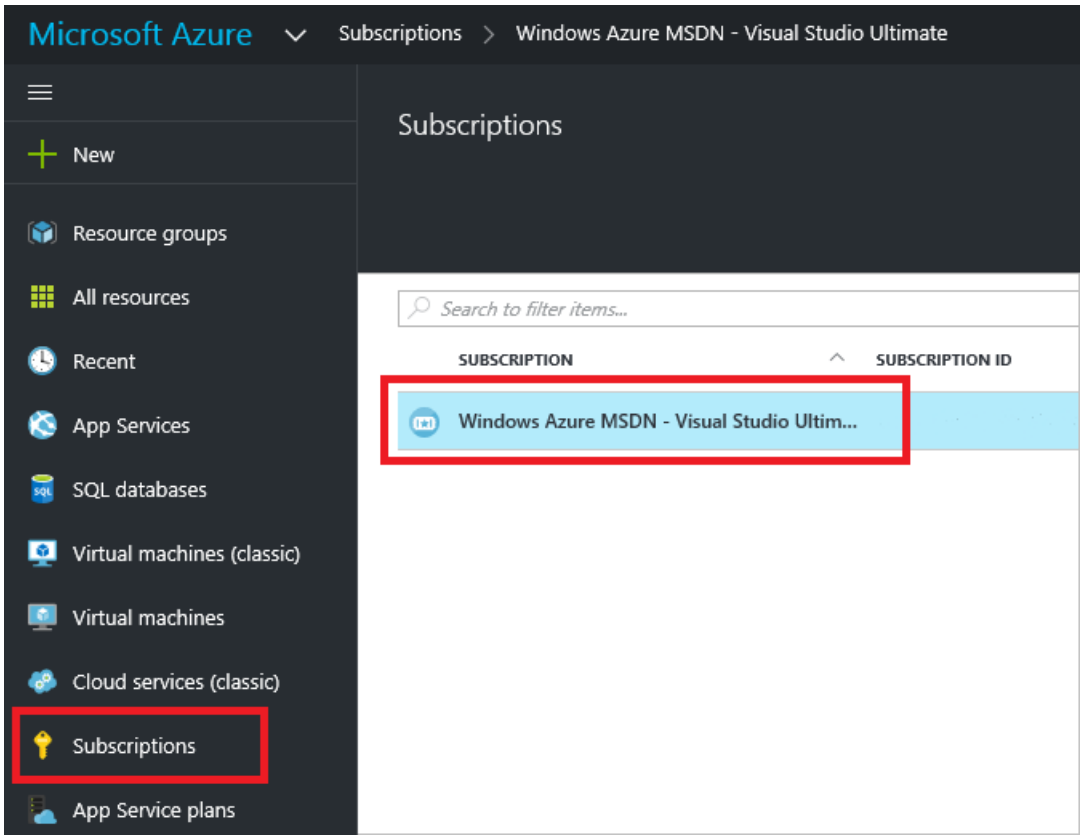



4. To delete the lock, select the ellipsis and **Delete** from the available options.

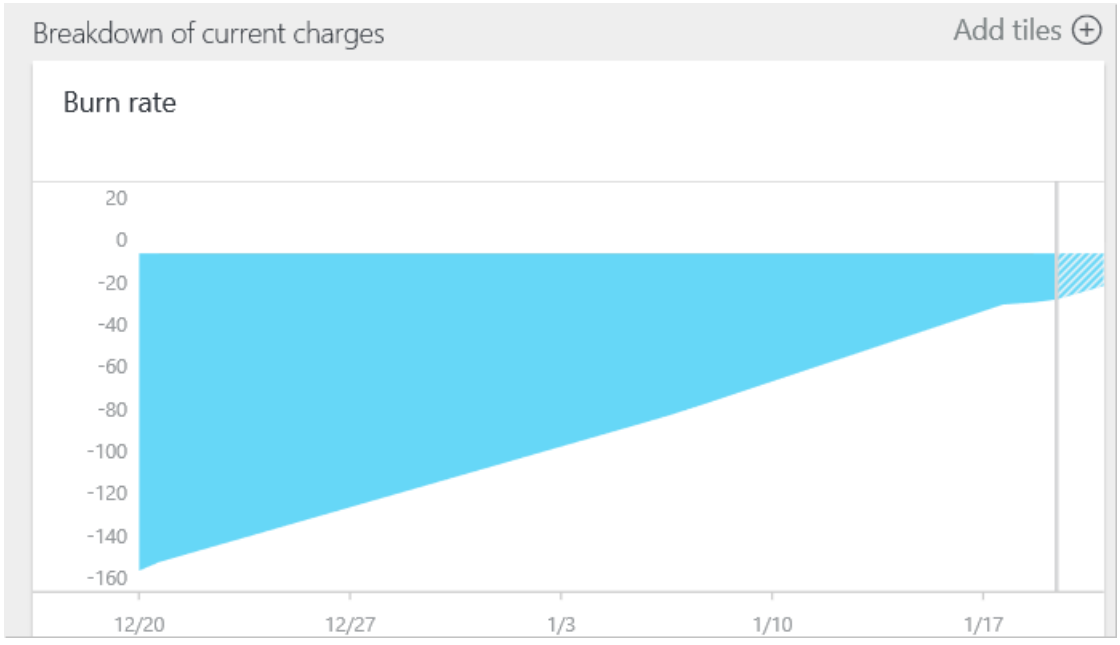



## View your subscription and costs

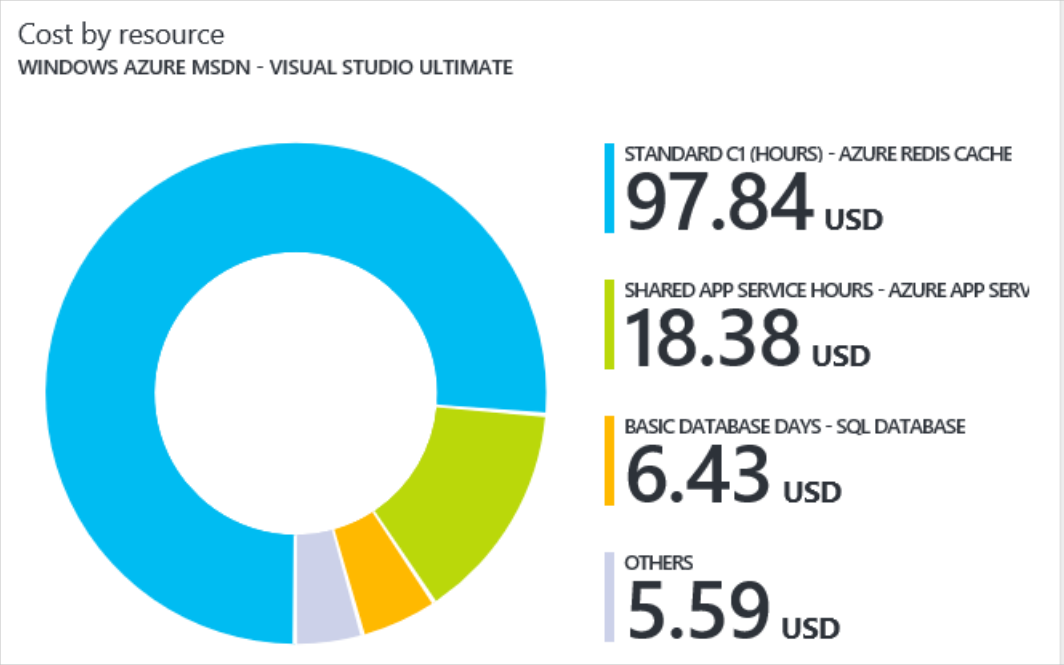
You can view information about your subscription and the rolled-up costs for all your resources. Select **Subscriptions** and the subscription you want to see. You might only have one subscription to select.




You see the burn rate.



And, a breakdown of costs by resource type.




## Export template

After setting up your resource group, you may want to view the Resource Manager template for the resource group. Exporting the template offers two benefits:

1. You can easily automate future deployments of the solution because the template contains all the complete infrastructure.
2. You can become familiar with template syntax by looking at the JavaScript Object Notation (JSON) that represents your solution.

For step-by-step guidance, see Export Azure Resource Manager template from existing resources.

## Delete resource group or resources

Deleting a resource group deletes all the resources contained within it. You can also delete individual resources within a resource group. Use caution when deleting a resource group. That resource group might contain resources that resources in other resource groups depend on.


## Next steps

- To view activity logs, see Audit operations with Resource Manager.
- To view details about a deployment, see View deployment operations.
- To deploy resources through the portal, see Deploy resources with Resource Manager templates and Azure portal.
- To manage access to resources, see Use role assignments to manage access to your Azure subscription resources.
- For guidance on how enterprises can use Resource Manager to effectively manage subscriptions, see Azure enterprise scaffold - prescriptive subscription governance.

# Use Role-Based Access Control to manage access to your Azure subscription resources

12/11/2017 • 2 min to read • Edit Online

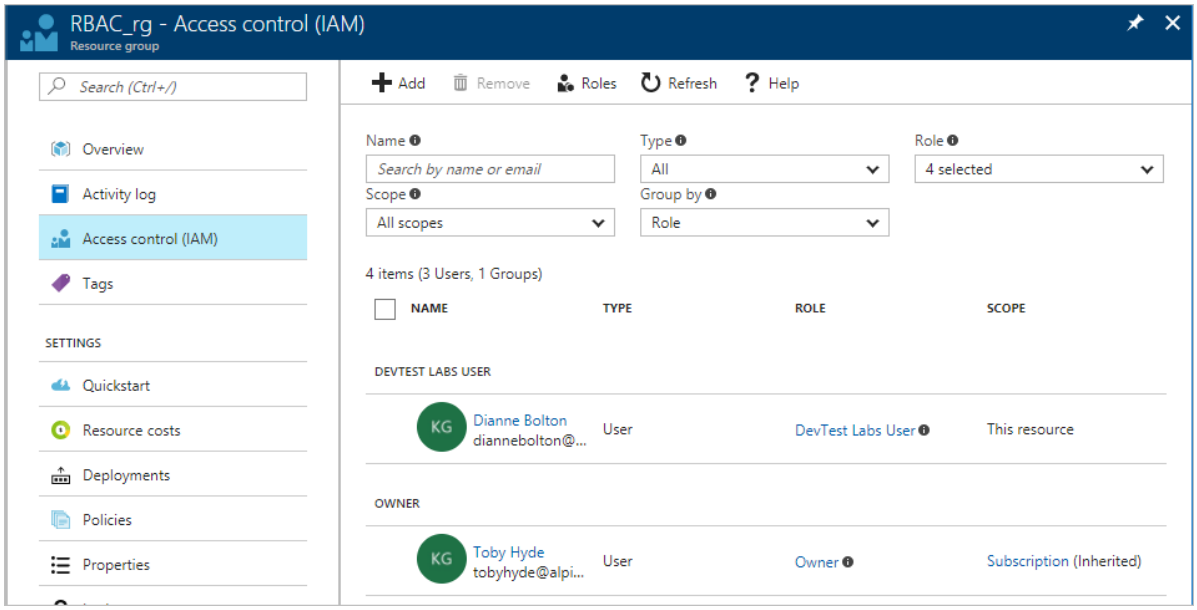
Azure Role-Based Access Control (RBAC) enables fine-grained access management for Azure. Using RBAC, you can grant only the amount of access that users need to perform their jobs. This article helps you get up and running with RBAC in the Azure portal. If you want more details about how RBAC helps you manage access, see What is Role-Based Access Control.

Within each subscription, you can grant up to 2000 role assignments.

## View access

You can see who has access to a resource, resource group, or subscription from its main blade in the Azure portal. For example, we want to see who has access to one of our resource groups:

1. Select **Resource groups** in the navigation bar on the left.
   
   
2. Select the name of the resource group from the **Resource groups** blade.
3. Select **Access control (IAM)** from the left menu.
4. The Access control blade lists all users, groups, and applications that have been granted access to the resource group.



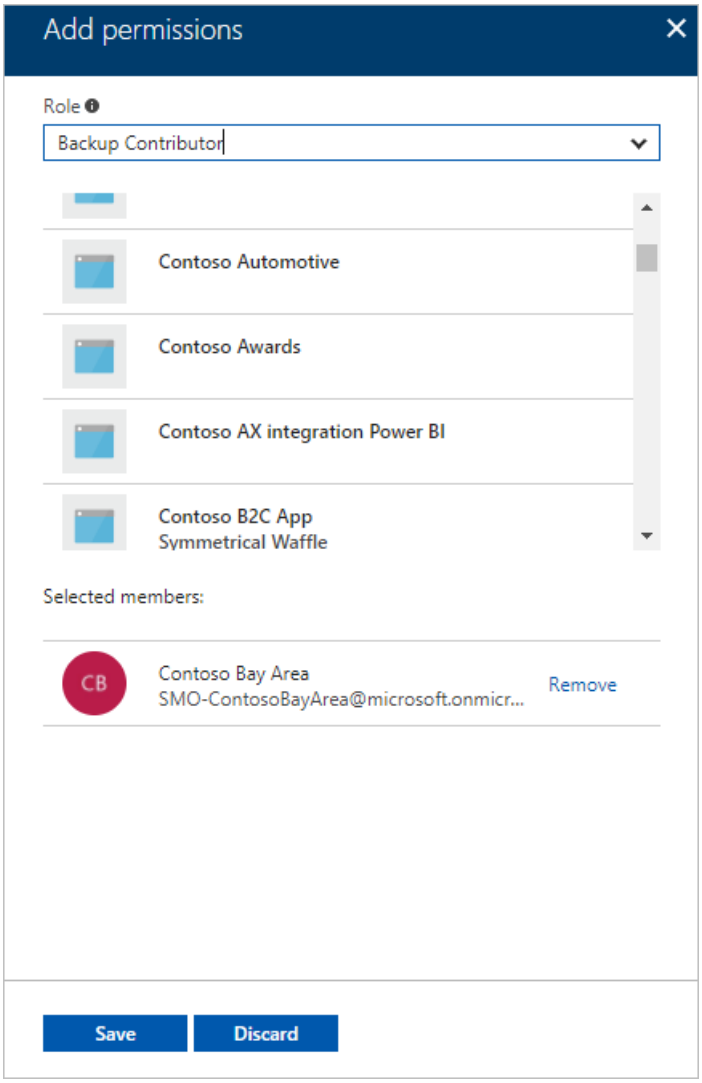

Notice that some roles are scoped to **This resource** while others are **Inherited** it from another scope. Access is either assigned specifically to the resource group or inherited from an assignment to the parent subscription.

> **NOTE**
> Classic subscription admins and co-admins are considered owners of the subscription in the new RBAC model.

## Add Access

You grant access from within the resource, resource group, or subscription that is the scope of the role assignment.

1. Select **Add** on the Access control blade.
2. Select the role that you wish to assign from the **Select a role** blade.
3. Select the user, group, or application in your directory that you wish to grant access to. You can search the directory with display names, email addresses, and object identifiers.



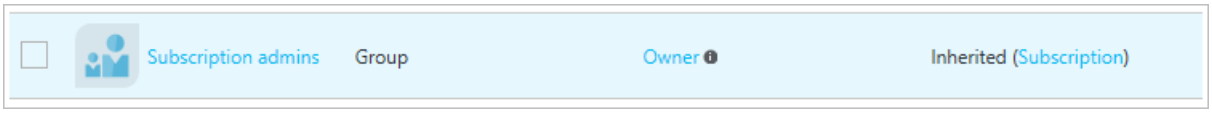

4. Select **OK** to create the assignment. The **Adding user** popup tracks the progress.


After successfully adding a role assignment, it will appear on the **Users** blade.

## Remove Access

1. Hover your cursor over the name of the assignment that you want to remove. A check box appears next to the name.
2. Use the check boxes to select one or more role assignments.
3. Select **Remove**.
4. Select **Yes** to confirm the removal.

Inherited assignments cannot be removed. If you need to remove an inherited assignment, you need to do it at the scope where the role assignment was created. In the **Scope** column, next to **Inherited** there is a link that takes you to the resources where this role was assigned. Go to the resource listed there to remove the role assignment.




## Other tools to manage access

You can assign roles and manage access with Azure RBAC commands in tools other than the Azure portal. Follow the links to learn more about the prerequisites and get started with the Azure RBAC commands.

- Azure PowerShell
- Azure Command-Line Interface
- REST API

## Next Steps

- Create an access change history report
- See the RBAC built-in roles
- Define your own Custom roles in Azure RBAC

# Share Azure dashboards by using Role-Based Access Control

12/11/2017 • 3 min to read • Edit Online

After configuring a dashboard, you can publish it and share it with other users in your organization. You allow others to view your dashboard by using Azure Role-Based Access Control. You assign a user or group of users to a role, and that role defines whether those users can view or modify the published dashboard.

All published dashboards are implemented as Azure resources, which means they exist as manageable items within your subscription and are contained in a resource group. From an access control perspective, dashboards are no different than other resources, such as a virtual machine or a storage account.

> **TIP**
> Individual tiles on the dashboard enforce their own access control requirements based on the resources they display. Therefore, you can design a dashboard that is shared broadly while still protecting the data on individual tiles.

## Understanding access control for dashboards

With Role-Based Access Control (RBAC), you can assign users to roles at three different levels of scope:

- subscription
- resource group
- resource

The permissions you assign are inherited from subscription down to the resource. The published dashboard is a resource. Therefore, you may already have users assigned to roles for the subscription which also work for the published dashboard.

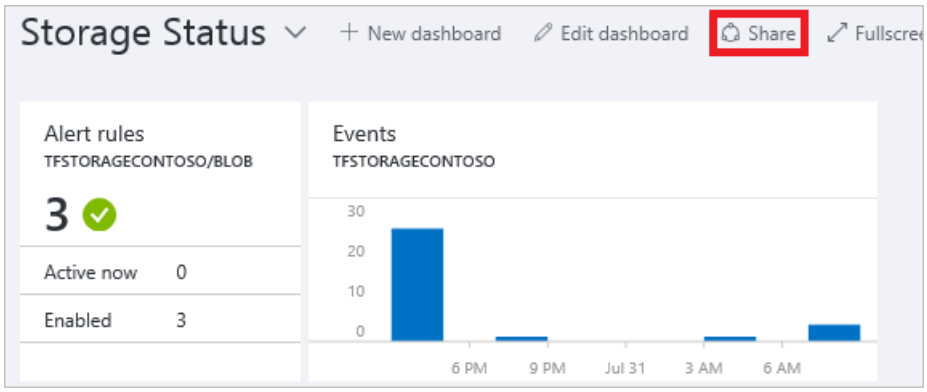
Here is an example. Let's say you have an Azure subscription and various members of your team have been assigned the roles of **owner**, **contributor**, or **reader** for the subscription. Users who are owners or contributors are able to list, view, create, modify, or delete dashboards within the subscription. Users who are readers are able to list and view dashboards, but cannot modify or delete them. Users with reader access are able to make local edits to a published dashboard (such as, when troubleshooting an issue), but are not able to publish those changes back to the server. They will have the option to make a private copy of the dashboard for themselves

However, you could also assign permissions to the resource group that contains several dashboards or to an individual dashboard. For example, you may decide that a group of users should have limited permissions across the subscription but greater access to a particular dashboard. You assign those users to a role for that dashboard.

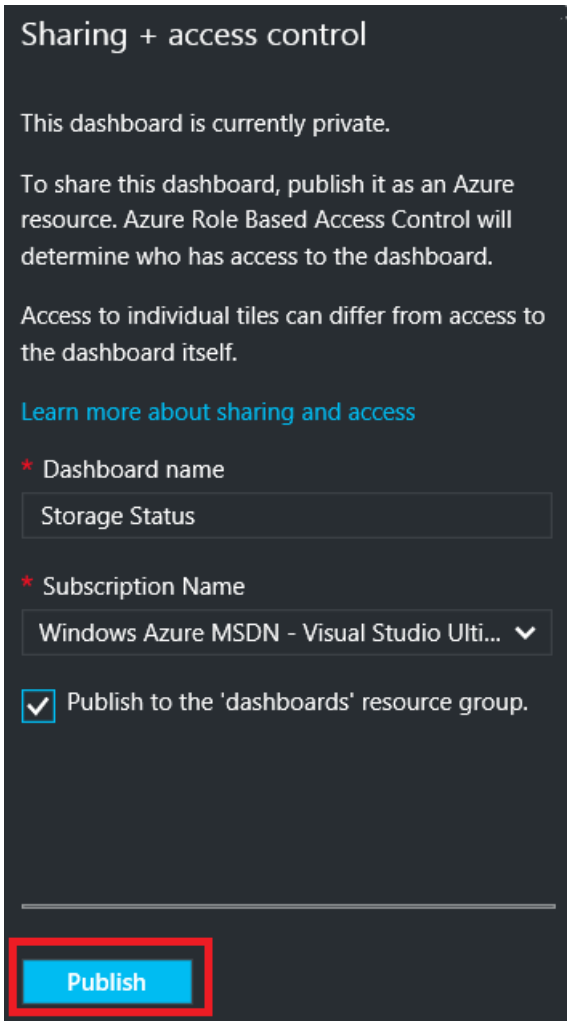
## Publish dashboard

Let's suppose you have finished configuring a dashboard that you want to share with a group of users in your subscription. The steps below depict a customized group called Storage Managers, but you can name your group whatever you would like. For information about creating an Active Directory group and adding users to that group, see Managing groups in Azure Active Directory.

1. In the dashboard, select **Share**.



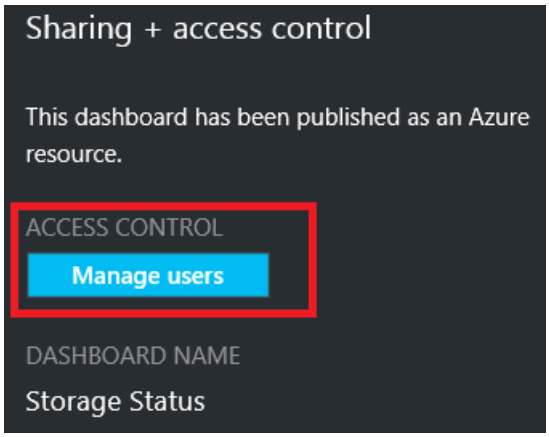

2. Before assigning access, you must publish the dashboard. By default, the dashboard will be published to a resource group named **dashboards**. Select **Publish**.



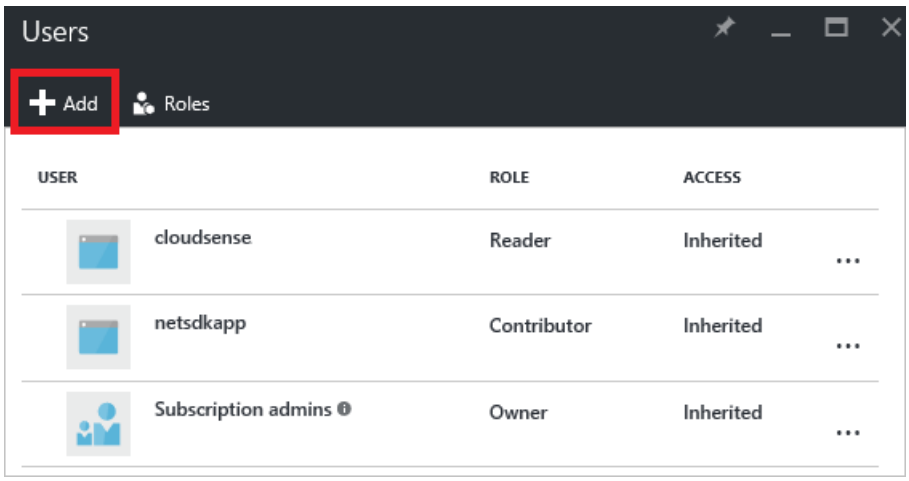

Your dashboard is now published. If the permissions inherited from the subscription are suitable, you do not need to do anything more. Other users in your organization will be able to access and modify the dashboard based on their subscription level role. However, for this tutorial, let's assign a group of users to a role for that dashboard.

## Assign access to a dashboard

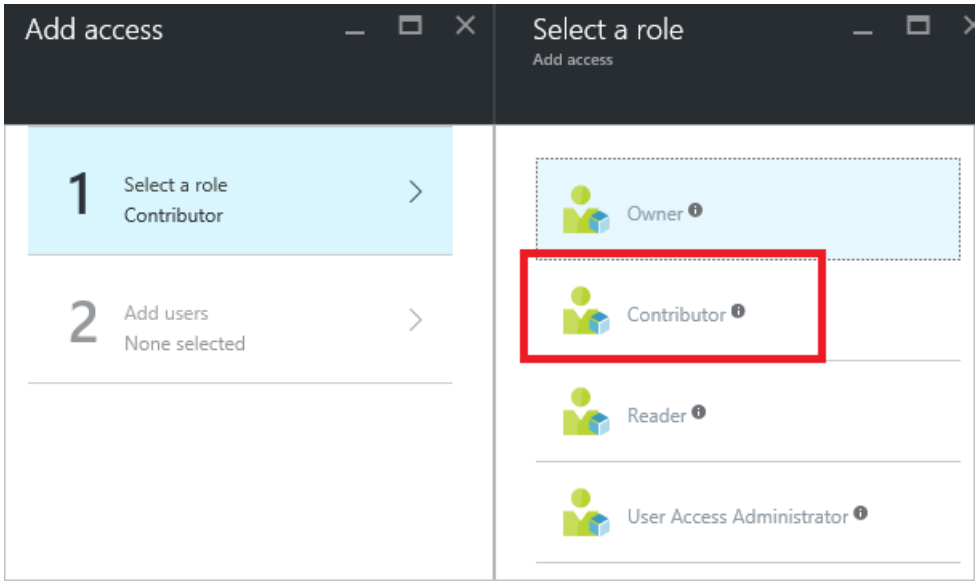
1. After publishing the dashboard, select **Manage users**.



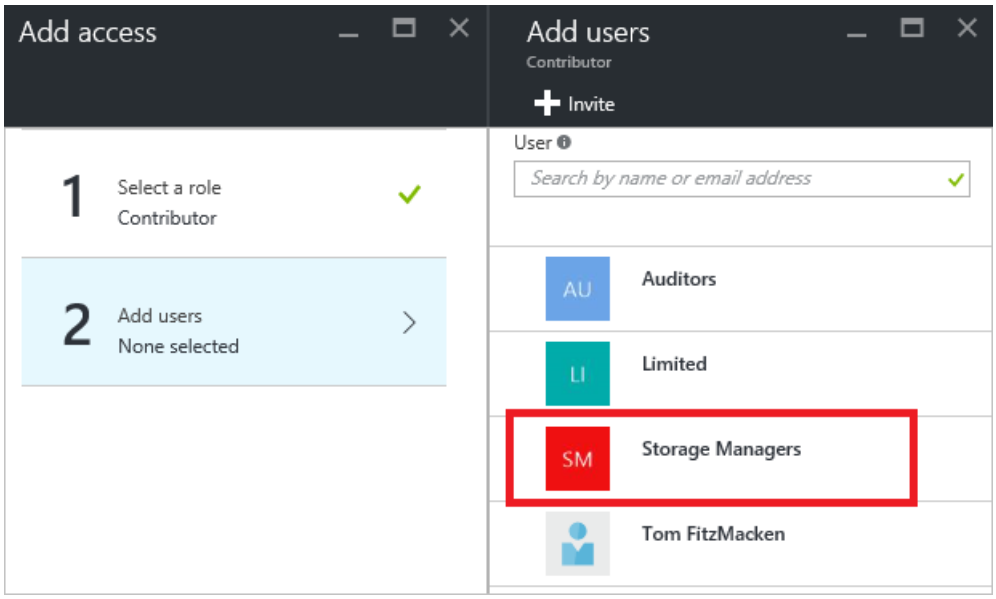

2. You will see a list of existing users that are already assigned a role for this dashboard. Your list of existing users will be different than the image below. Most likely, the assignments are inherited from the subscription. To add a new user or group, select **Add**.



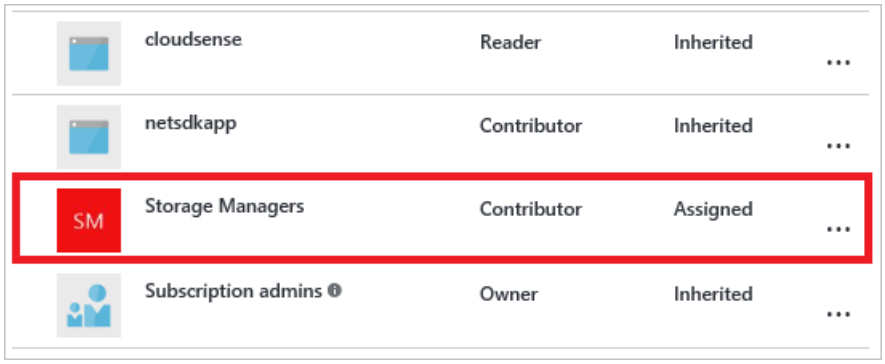

3. Select the role that represents the permissions you would like to grant. For this example, select **Contributor**.




4. Select the user or group that you wish to assign to the role. If you do not see the user or group you are looking for in the list, use the search box. Your list of available groups will depend on the groups you have created in your Active Directory.




5. When you have finished adding users or groups, select **OK**.
6. The new assignment is added to the list of users. Notice that its **Access** is listed as **Assigned** rather than **Inherited**.




## Next steps

- For a list of roles, see RBAC: Built-in roles.
- To learn about managing resources, see Manage Azure resources through portal.

# Use tags to organize your Azure resources

11/15/2017 • 9 min to read • Edit Online

You apply tags to your Azure resources to logically organize them by categories. Each tag consists of a name and a value. For example, you can apply the name "Environment" and the value "Production" to all the resources in production. Without this tag, you might have difficulty identifying whether a resource is intended for development, test, or production. However, "Environment" and "Production" are just examples. You define the names and values that make the most sense for organizing your subscription.

After you apply tags, you can retrieve all the resources in your subscription with that tag name and value. Tags enable you to retrieve related resources that reside in different resource groups. This approach is helpful when you need to organize resources for billing or management.

The following limitations apply to tags:

- Each resource or resource group can have a maximum of 15 tag name/value pairs. This limitation applies only to tags directly applied to the resource group or resource. A resource group can contain many resources that each have 15 tag name/value pairs.
- The tag name is limited to 512 characters, and the tag value is limited to 256 characters. For storage accounts, the tag name is limited to 128 characters, and the tag value is limited to 256 characters.
- Tags applied to the resource group are not inherited by the resources in that resource group.

If you have more than 15 values that you need to associate with a resource, use a JSON string for the tag value. The JSON string can contain many values that are applied to a single tag name. This article shows an example of assigning a JSON string to the tag.

## PowerShell

The examples in this article require version 3.0 or later of Azure PowerShell. If you do not have version 3.0 or later, update your version by using PowerShell Gallery or Web Platform Installer.

To see the existing tags for a *resource group*, use:

```
(Get-AzureRmResourceGroup -Name examplegroup).Tags
```

That script returns the following format:

```
Name                           Value
----                           -----
Dept                           IT
Environment                    Test
```

To see the existing tags for a *resource that has a specified resource ID*, use:

```
(Get-AzureRmResource -ResourceId {resource-id}).Tags
```

Or, to see the existing tags for a *resource that has a specified name and resource group*, use:

```
(Get-AzureRmResource -ResourceName examplevnet -ResourceGroupName examplegroup).Tags
```

To get *resource groups that have a specific tag*, use:

```
(Find-AzureRmResourceGroup -Tag @{ Dept="Finance" }).Name
```

To get *resources that have a specific tag*, use:

```
(Find-AzureRmResource -TagName Dept -TagValue Finance).Name
```

Every time you apply tags to a resource or a resource group, you overwrite the existing tags on that resource or resource group. Therefore, you must use a different approach based on whether the resource or resource group has existing tags.

To add tags to a *resource group without existing tags*, use:

```
Set-AzureRmResourceGroup -Name examplegroup -Tag @{ Dept="IT"; Environment="Test" }
```

To add tags to a *resource group that has existing tags*, retrieve the existing tags, add the new tag, and reapply the tags:

```
$tags = (Get-AzureRmResourceGroup -Name examplegroup).Tags
$tags += @{Status="Approved"}
Set-AzureRmResourceGroup -Tag $tags -Name examplegroup
```

To add tags to a *resource without existing tags*, use:

```
$r = Get-AzureRmResource -ResourceName examplevnet -ResourceGroupName examplegroup
Set-AzureRmResource -Tag @{ Dept="IT"; Environment="Test" } -ResourceId $r.ResourceId -Force
```

To add tags to a *resource that has existing tags*, use:

```
$r = Get-AzureRmResource -ResourceName examplevnet -ResourceGroupName examplegroup
$r.tags += @{Status="Approved"}
Set-AzureRmResource -Tag $r.Tags -ResourceId $r.ResourceId -Force
```

To apply all tags from a resource group to its resources, and *not retain existing tags on the resources*, use the following script:

```
$groups = Get-AzureRmResourceGroup
foreach ($g in $groups)
{
    Find-AzureRmResource -ResourceGroupNameEquals $g.ResourceGroupName | ForEach-Object {Set-AzureRmResource -
ResourceId $_.ResourceId -Tag $g.Tags -Force }
}
```

To apply all tags from a resource group to its resources, and *retain existing tags on resources that are not duplicates*, use the following script:

```
$group = Get-AzureRmResourceGroup "examplegroup"
if ($group.Tags -ne $null) {
    $resources = $group | Find-AzureRmResource
    foreach ($r in $resources)
    {
        $resourcetags = (Get-AzureRmResource -ResourceId $r.ResourceId).Tags
        foreach ($key in $group.Tags.Keys)
        {
            if (($resourcetags) -AND ($resourcetags.ContainsKey($key))) { $resourcetags.Remove($key) }
        }
        $resourcetags += $group.Tags
        Set-AzureRmResource -Tag $resourcetags -ResourceId $r.ResourceId -Force
    }
}
```

To remove all tags, pass an empty hash table:

```
Set-AzureRmResourceGroup -Tag @{} -Name examplegroup
```

## Azure CLI

To see the existing tags for a *resource group*, use:

```
az group show -n examplegroup --query tags
```

That script returns the following format:

```
{
  "Dept"        : "IT",
  "Environment" : "Test"
}
```

Or, to see the existing tags for a *resource that has a specified name, type, and resource group*, use:

```
az resource show -n examplevnet -g examplegroup --resource-type "Microsoft.Network/virtualNetworks" --query
tags
```

When looping through a collection of resources, you might want to show the resource by resource ID. A complete example is shown later in this article. To see the existing tags for a *resource that has a specified resource ID*, use:

```
az resource show --id <resource-id> --query tags
```

To get resource groups that have a specific tag, use `az group list`:

```
az group list --tag Dept=IT
```

To get all the resources that have a particular tag and value, use `az resource list`:

```
az resource list --tag Dept=Finance
```

Every time you apply tags to a resource or a resource group, you overwrite the existing tags on that resource or resource group. Therefore, you must use a different approach based on whether the resource or resource group

has existing tags.

To add tags to a *resource group without existing tags*, use:

```
az group update -n examplegroup --set tags.Environment=Test tags.Dept=IT
```

To add tags to a *resource without existing tags*, use:

```
az resource tag --tags Dept=IT Environment=Test -g examplegroup -n examplevnet --resource-type
"Microsoft.Network/virtualNetworks"
```

To add tags to a resource that already has tags, retrieve the existing tags, reformat that value, and reapply the existing and new tags:

```
jsonrtag=$(az resource show -g examplegroup -n examplevnet --resource-type "Microsoft.Network/virtualNetworks"
--query tags)
rt=$(echo $jsonrtag | tr -d '"{},' | sed 's/: /=/g')
az resource tag --tags $rt Project=Redesign -g examplegroup -n examplevnet --resource-type
"Microsoft.Network/virtualNetworks"
```

To apply all tags from a resource group to its resources, and *not retain existing tags on the resources*, use the following script:

```
groups=$(az group list --query [].name --output tsv)
for rg in $groups
do
  jsontag=$(az group show -n $rg --query tags)
  t=$(echo $jsontag | tr -d '"{},' | sed 's/: /=/g')
  r=$(az resource list -g $rg --query [].id --output tsv)
  for resid in $r
  do
    az resource tag --tags $t --id $resid
  done
done
```

To apply all tags from a resource group to its resources, and *retain existing tags on resources*, use the following script:

```
groups=$(az group list --query [].name --output tsv)
for rg in $groups
do
  jsontag=$(az group show -n $rg --query tags)
  t=$(echo $jsontag | tr -d '"{},' | sed 's/: /=/g')
  r=$(az resource list -g $rg --query [].id --output tsv)
  for resid in $r
  do
    jsonrtag=$(az resource show --id $resid --query tags)
    rt=$(echo $jsonrtag | tr -d '"{},' | sed 's/: /=/g')
    az resource tag --tags $t$rt --id $resid
  done
done
```

# Templates

To tag a resource during deployment, add the `tags` element to the resource you are deploying. Provide the tag name and value.

### Apply a literal value to the tag name

The following example shows a storage account with two tags ( `Dept` and `Environment` ) that are set to literal values:

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "resources": [
    {
      "apiVersion": "2016-01-01",
      "type": "Microsoft.Storage/storageAccounts",
      "name": "[concat('storage', uniqueString(resourceGroup().id))]",
      "location": "[resourceGroup().location]",
      "tags": {
        "Dept": "Finance",
        "Environment": "Production"
      },
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "properties": { }
    }
    ]
}
```

### Apply an object to the tag element

You can define an object parameter that stores several tags, and apply that object to the tag element. Each property in the object becomes a separate tag for the resource. The following example has a parameter named `tagValues` that is applied to the tag element.

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "tagValues": {
      "type": "object",
      "defaultValue": {
        "Dept": "Finance",
        "Environment": "Production"
      }
    }
  },
  "resources": [
    {
      "apiVersion": "2016-01-01",
      "type": "Microsoft.Storage/storageAccounts",
      "name": "[concat('storage', uniqueString(resourceGroup().id))]",
      "location": "[resourceGroup().location]",
      "tags": "[parameters('tagValues')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "properties": {}
    }
  ]
}
```

### Apply a JSON string to the tag name

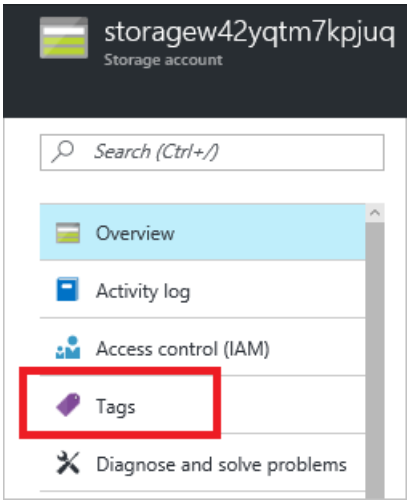
To store many values in a single tag, apply a JSON string that represents the values. The entire JSON string is

stored as one tag that cannot exceed 256 characters. The following example has a single tag named `CostCenter` that contains several values from a JSON string:

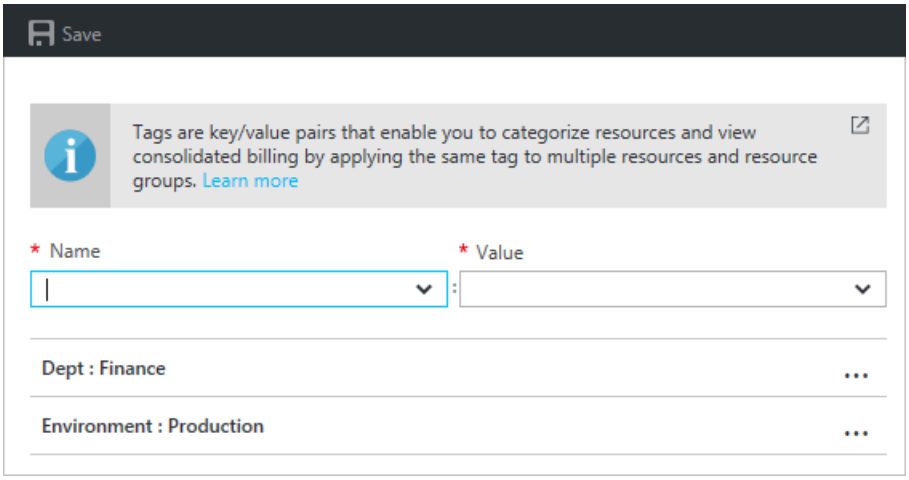
```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "resources": [
    {
      "apiVersion": "2016-01-01",
      "type": "Microsoft.Storage/storageAccounts",
      "name": "[concat('storage', uniqueString(resourceGroup().id))]",
      "location": "[resourceGroup().location]",
      "tags": {
        "CostCenter": "{\"Dept\":\"Finance\",\"Environment\":\"Production\"}"
      },
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "properties": { }
    }
    ]
}
```

## Portal

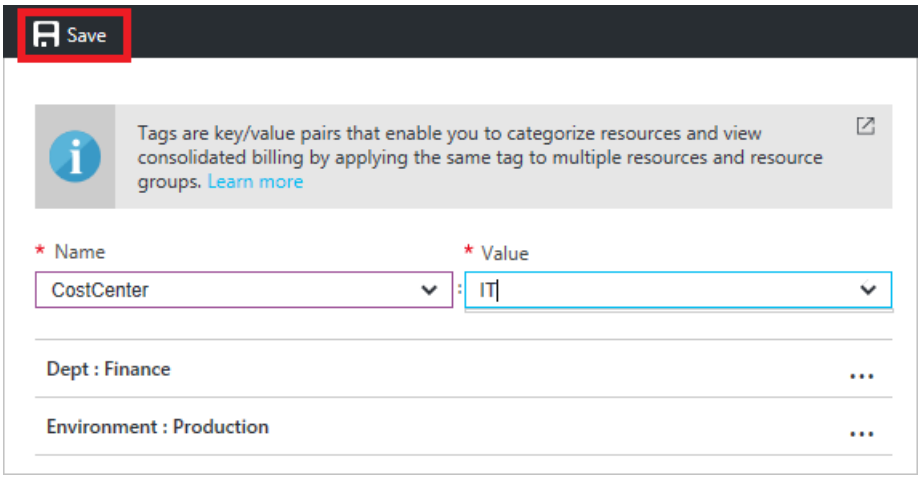
1. To view the tags for a resource or a resource group, select the **Tags** icon.



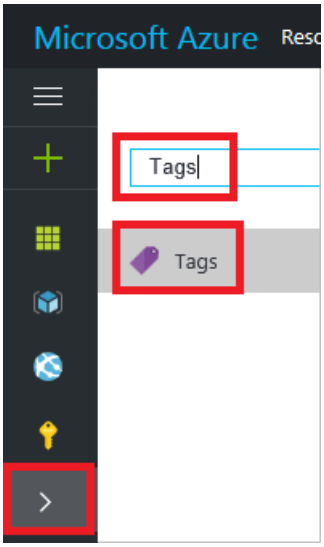

2. You see the existing tags for the resource. If you have not previously applied tags, the list is empty.



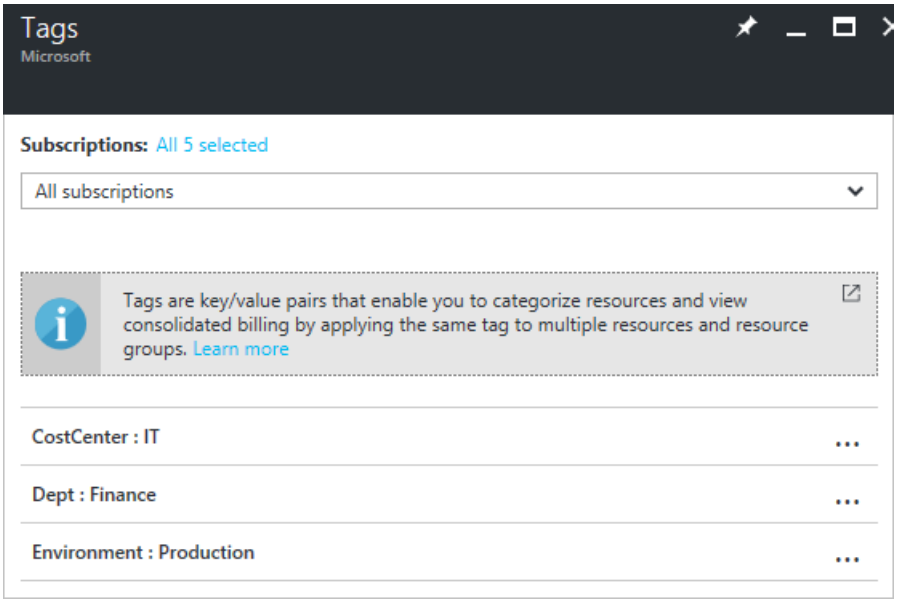

3. To add a tag, type a name and value, or select an existing one from the drop-down menu. Select **Save**.



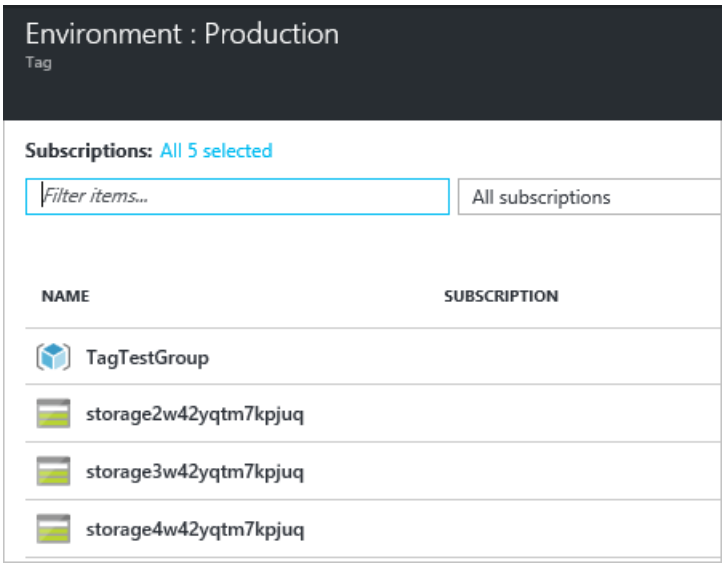

4. To view all the resources that have a tag value, select **>** (**More services**), and enter the word **Tags** into the filter text box. Select **Tags** from the available options.



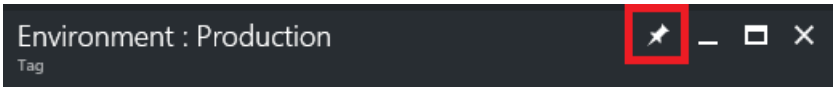

5. You see a summary of the tags in your subscriptions.



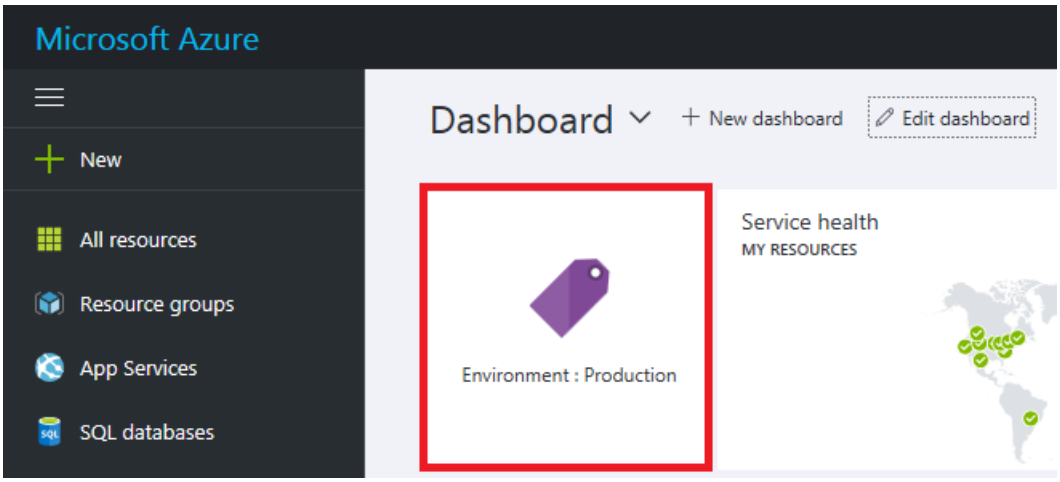

6. Select any of the tags to display the resources and resource groups with that tag.




7. Select **Pin blade to dashboard** for quick access.




8. You can select the pinned tag from the dashboard to see the resources with that tag.




## REST API

The Azure portal and PowerShell both use the Resource Manager REST API behind the scenes. If you need to integrate tagging into another environment, you can get tags by using **GET** on the resource ID and update the set of tags by using a **PATCH** call.

## Tags and billing

You can use tags to group your billing data. For example, if you are running multiple VMs for different organizations, use the tags to group usage by cost center. You can also use tags to categorize costs by runtime environment, such as the billing usage for VMs running in the production environment.

You can retrieve information about tags through the Azure Resource Usage and RateCard APIs or the usage comma-separated values (CSV) file. You download the usage file from the Azure account portal or EA portal. For more information about programmatic access to billing information, see Gain insights into your Microsoft Azure resource consumption. For REST API operations, see Azure Billing REST API Reference.

When you download the usage CSV for services that support tags with billing, the tags appear in the **Tags** column. For more information, see Understand your bill for Microsoft Azure.

| Daily Usage | | | | | | |
|---|---|---|---|---|---|---|
| Usage Date | Meter Category | Unit | Consume | Resource Gr | Instance Id | Tags |
| 5/14/2015 | "Virtual Machines" | "Hours" | 3.999984 | "computeRG | "virtualMachines/catalogVM" | "{"costCenter":"finance", "env":"prod"}" |
| 5/14/2015 | "Virtual Machines" | "Hours" | 3.999984 | "businessRG | "virtualMachines/dataVM" | "{"costCenter":"hr", "env":"test"}" |

## Next steps

- You can apply restrictions and conventions across your subscription by using customized policies. A policy that you define might require that all resources have a value for a particular tag. For more information, see What is Azure Policy?.
- For an introduction to using Azure PowerShell when you're deploying resources, see Using Azure PowerShell with Azure Resource Manager.
- For an introduction to using the Azure CLI when you're deploying resources, see Using the Azure CLI for Mac, Linux, and Windows with Azure Resource Manager.
- For an introduction to using the portal, see Using the Azure portal to manage your Azure resources.
- For guidance on how enterprises can use Resource Manager to effectively manage subscriptions, see Azure enterprise scaffold - prescriptive subscription governance.

# Scale instance count manually or automatically

12/13/2017 • 6 min to read • Edit Online

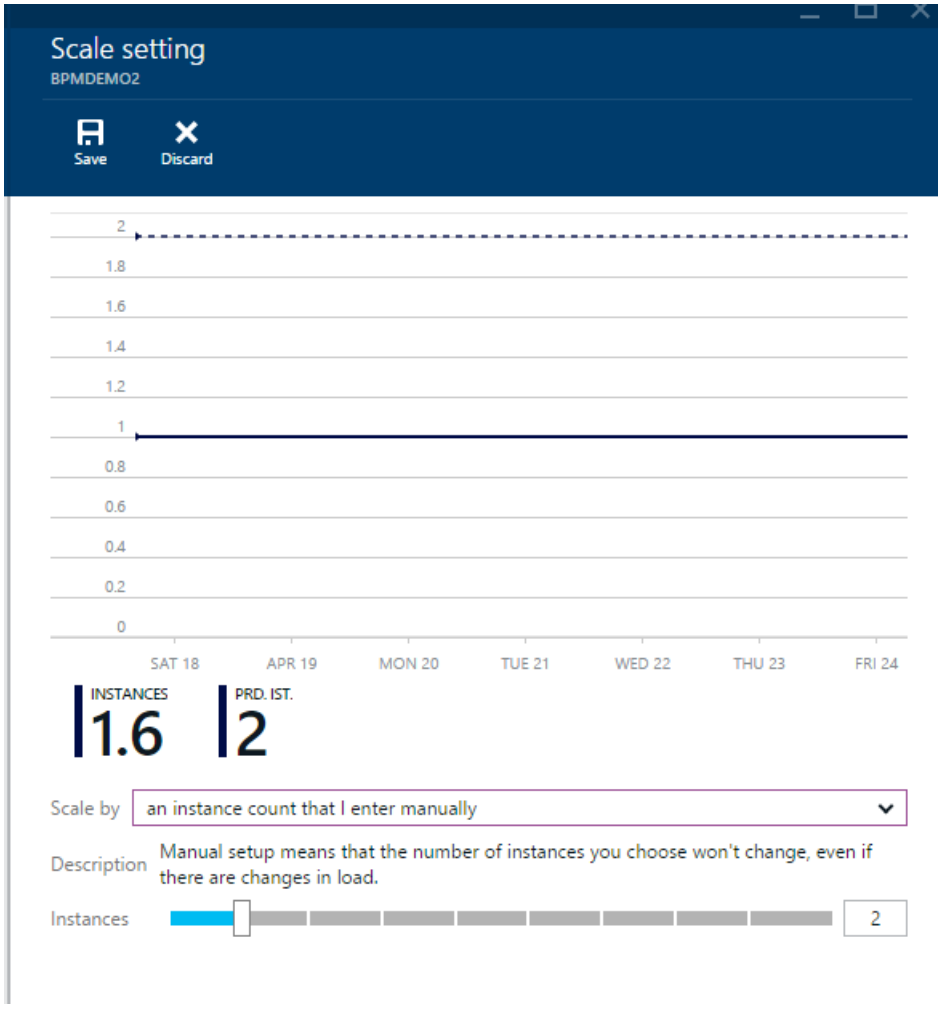
In the Azure Portal, you can manually set the instance count of your service, or, you can set parameters to have it automatically scale based on demand. This is typically referred to as *Scale out* or *Scale in*.

Before scaling based on instance count, you should consider that scaling is affected by **Pricing tier** in addition to instance count. Different pricing tiers can have different numbers cores and memory, and so they will have better performance for the same number of instances (which is *Scale up* or *Scale down*). This article specifically covers *Scale in* and *out*.

You can scale in the portal, and you can also use the REST API or .NET SDK to adjust scale manually or automatically.

## Scaling manually

1. In the Azure Portal, click **Browse**, then navigate to the resource you want to scale, such as an **App Service plan**.
2. Click **Settings > Scale out (App Service plan).**
3. At the top of the **Scale** blade you can see a history of autoscale actions of the service.



> **NOTE**
>
> Only actions that are performed by autoscale will show up in this chart. If you manually adjust the instance count, the change will not be reflected in this chart.

4. You can manually adjust the number **Instances** with slider.
5. Click the **Save** command and you'll be scaled to that number of instances almost immediately.

## Scaling based on a pre-set metric

If you want the number of instances to automatically adjust based on a metric, select the metric you want in the **Scale by** dropdown. For example, for an **App Service plan** you can scale by **CPU Percentage**.

1. When you select a metric you'll get a slider, and/or, text boxes to enter the number of instances you want to scale between:


   Autoscale will never take your service below or above the boundaries that you set, no matter your load.

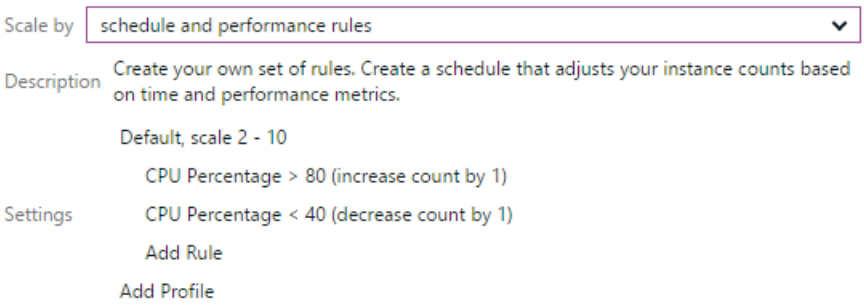
2. Second, you choose the target range for the metric. For example, if you chose **CPU percentage**, you can set a target for the average CPU across all of the instances in your service. A scale out will happen when the average CPU exceeds the maximum you define, likewise, a scale in will happen whenever the average CPU drops below the minimum.
3. Click the **Save** command. Autoscale will check every few minutes to make sure that you are in the instance range and target for your metric. When your service receives additional traffic, you will get more instances without doing anything.

# Scale based on other metrics

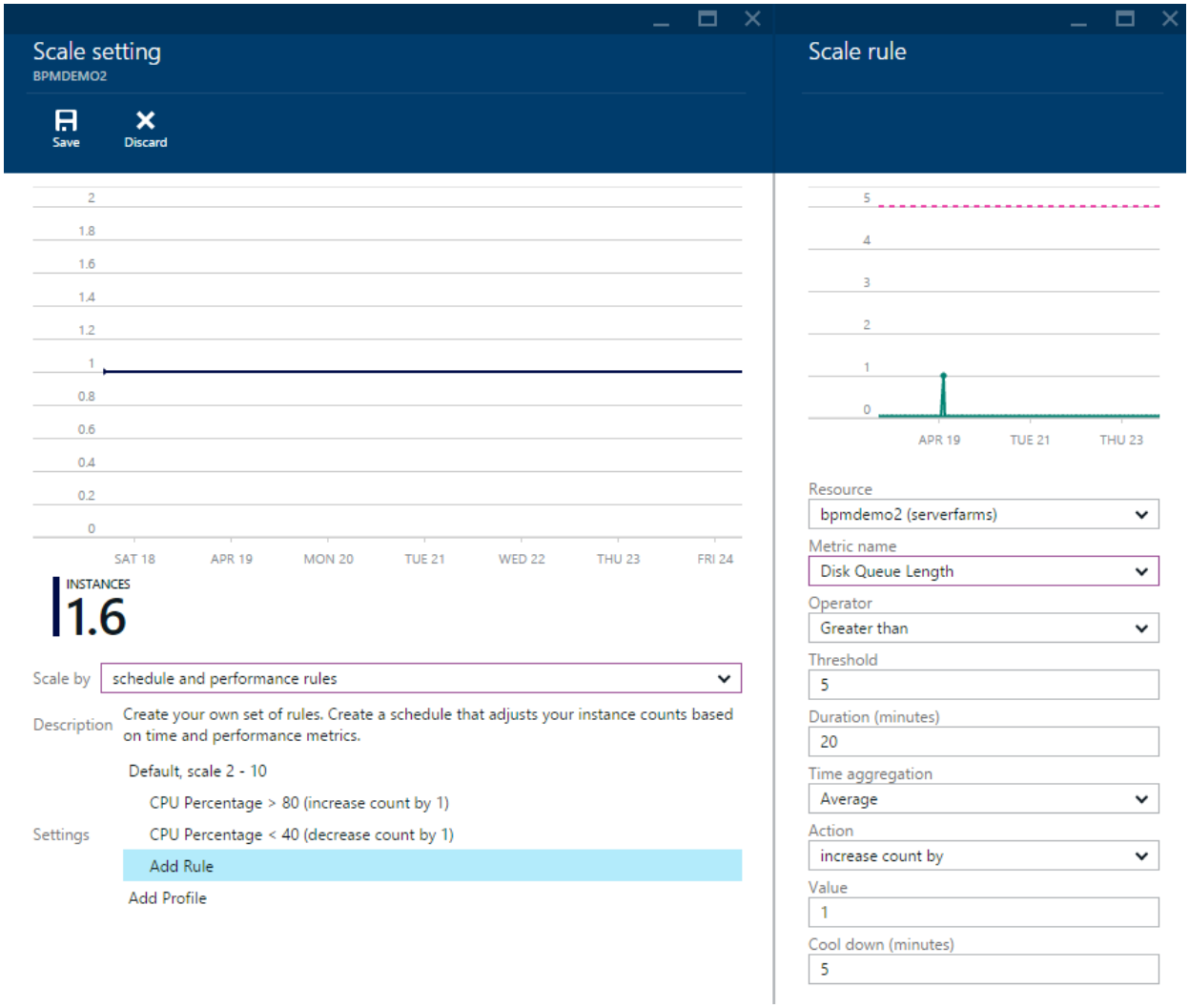
You can scale based on metrics other than the presets that appear in the **Scale by** dropdown, and can even have a complex set of scale out and scale in rules.

### Adding or changing a rule

1. Choose the **schedule and performance rules** in the **Scale by** dropdown:




2. If you previously had autoscale, on you'll see a view of the exact rules that you had.
3. To scale based on another metric click the **Add Rule** row. You can also click one of the existing rows to change from the metric you previously had to the metric you want to scale by.




4. Now you need to select which metric you want to scale by. When choosing a metric there are a couple things to consider:

   - The *resource* the metric comes from. Typically, this will be the same as the resource you are scaling. However, if you want to scale by the depth of a Storage queue, the resource is the queue that you want to scale by.
   - The *metric name* itself.

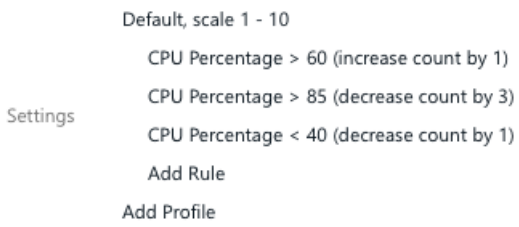
- The *time aggregation* of the metric. This is how the data is combine over the *duration*.

5. After choosing your metric you choose the threshold for the metric, and the operator. For example, you could say **Greater than 80%**.
6. Then choose the action that you want to take. There are a couple different type of actions:
   - Increase or decrease by - this will add or remove the **Value** number of instances you define
   - Increase or decrease percent - this will change the instance count by a percent. For example, you could put 25 in the **Value** field, and if you currently had 8 instances, 2 would be added.
   - Increase or decrease to - this will set the instance count to the **Value** you define.
7. Finally, you can choose cool down - how long this rule should wait after the previous scale action to scale again.
8. After configuring your rule hit **OK**.
9. Once you have configured all of the rules you want, be sure to hit the **Save** command.

**Scaling with multiple steps**

The examples above are pretty basic. However, if you want to be more agressive about scaling up (or down), you can even add multiple scale rules for the same metric. For example, you can define two scale rules on CPU percentage:

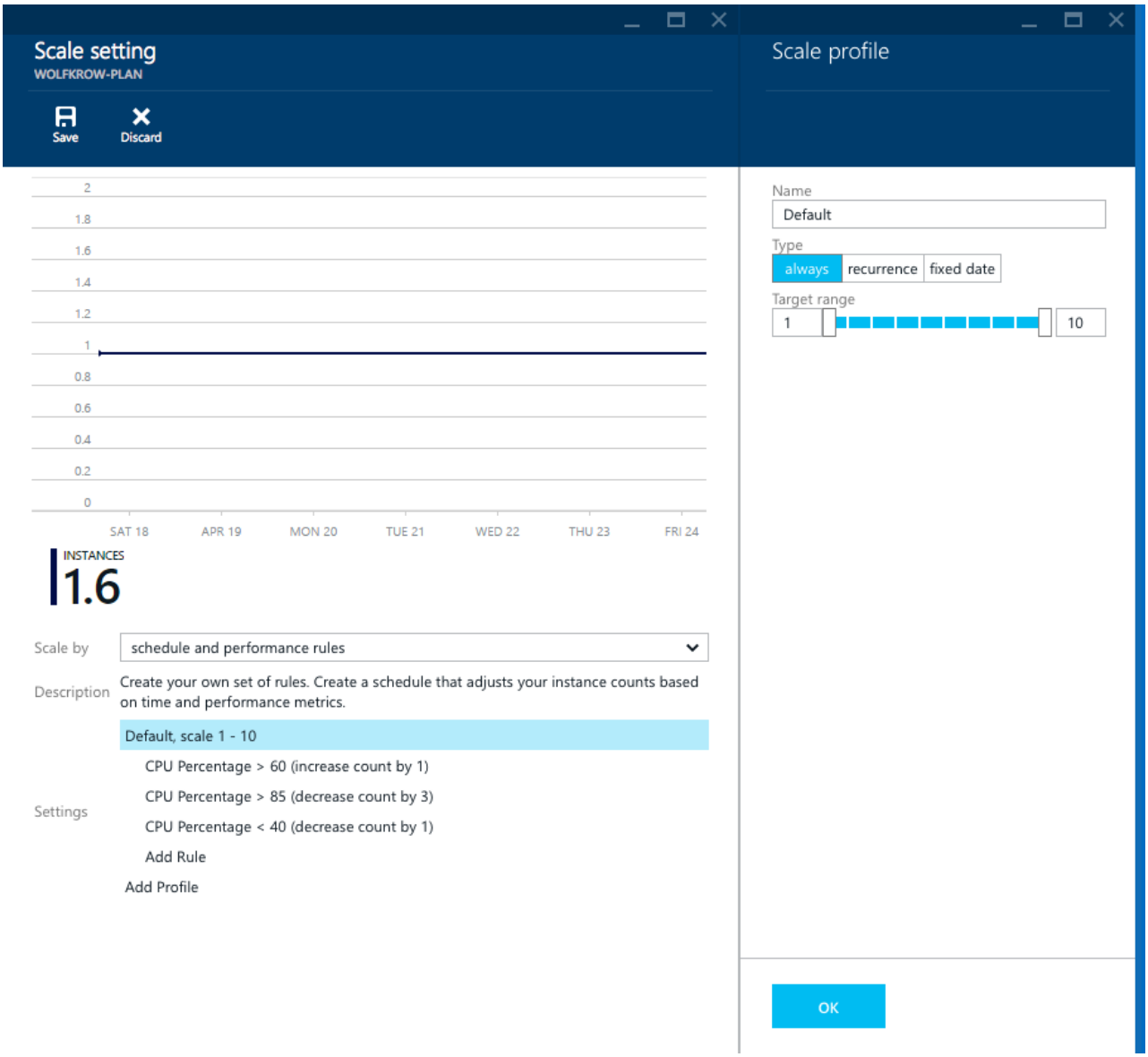
1. Scale out by 1 instance if CPU percentage is above 60%
2. Scale out by 3 instances if CPU percentage is above 85%




With this additional rule, if your load exceeds 85% before a scale action, you will get two additional instances instead of one.

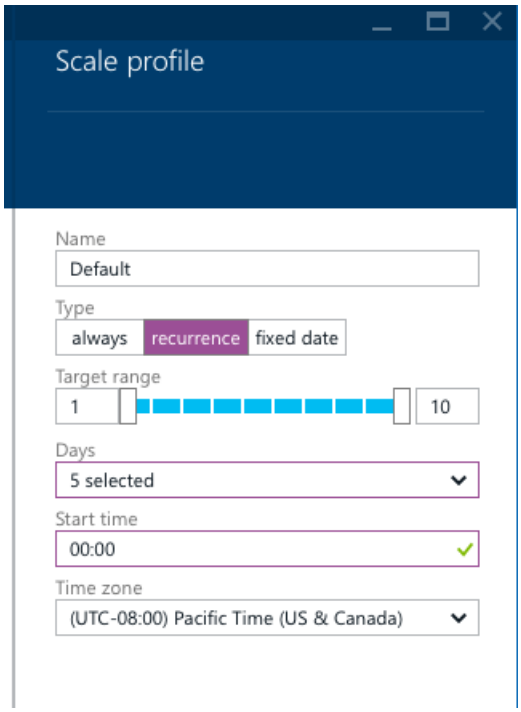
## Scale based on a schedule

By default, when you create a scale rule it will always apply. You can see that when you click on the profile header:

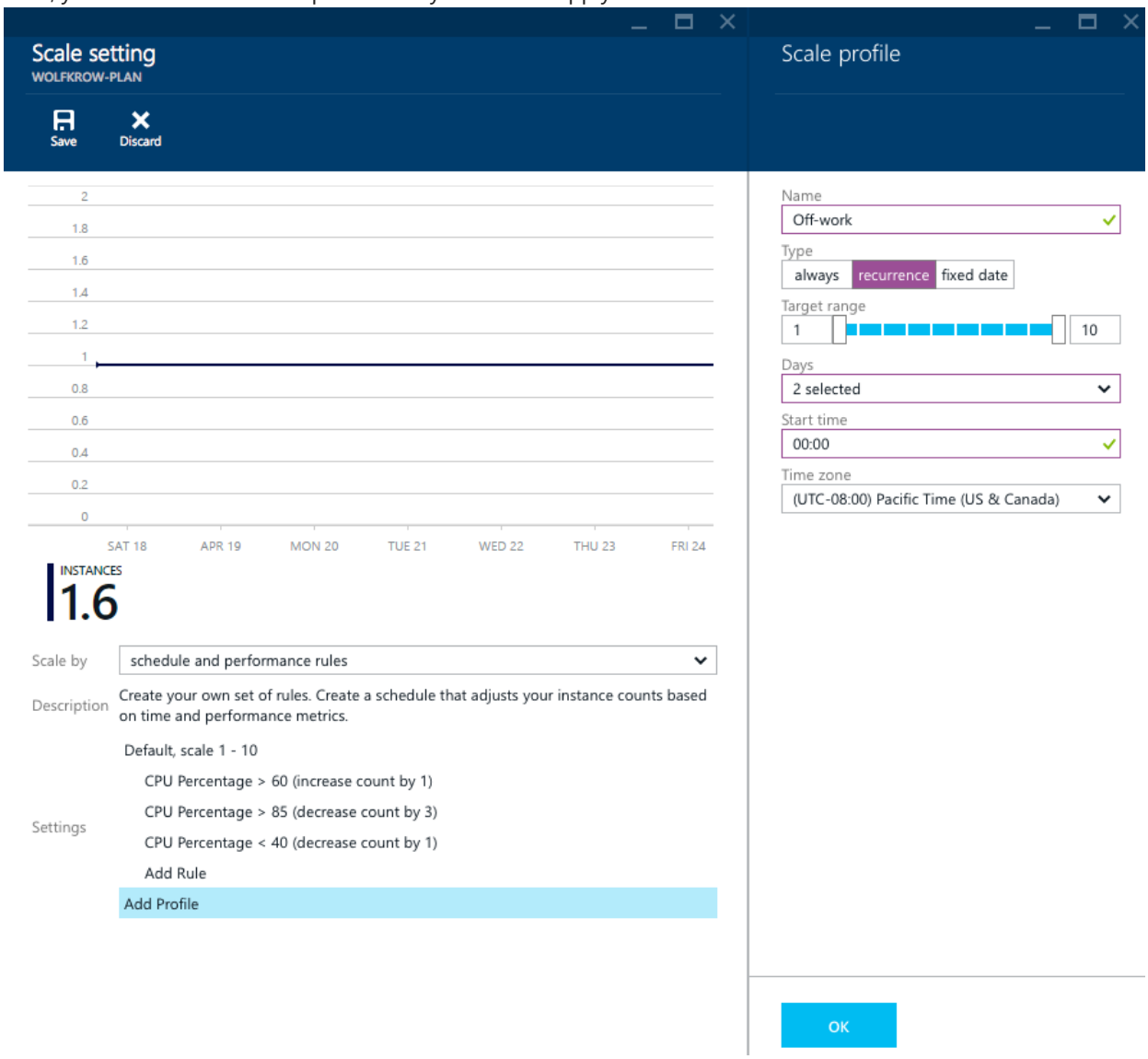



However, you may want to have more agressive scaling during the day, or the week, than on the weekend. You could even shut down your service entirely off working hours.

1. To do this, on the profile you have, select **recurrence** instead of **always,** and choose the times that you want the profile to apply.
2. For example, to have a profile that applies during the week, in the **Days** dropdown uncheck **Saturday** and **Sunday**.
3. To have a profile that applies during the daytime, set the **Start time** to the time of day that you want to start at.




4. Click **OK**.
5. Next, you will need to add the profile that you want to apply at other times. Click the **Add Profile** row.

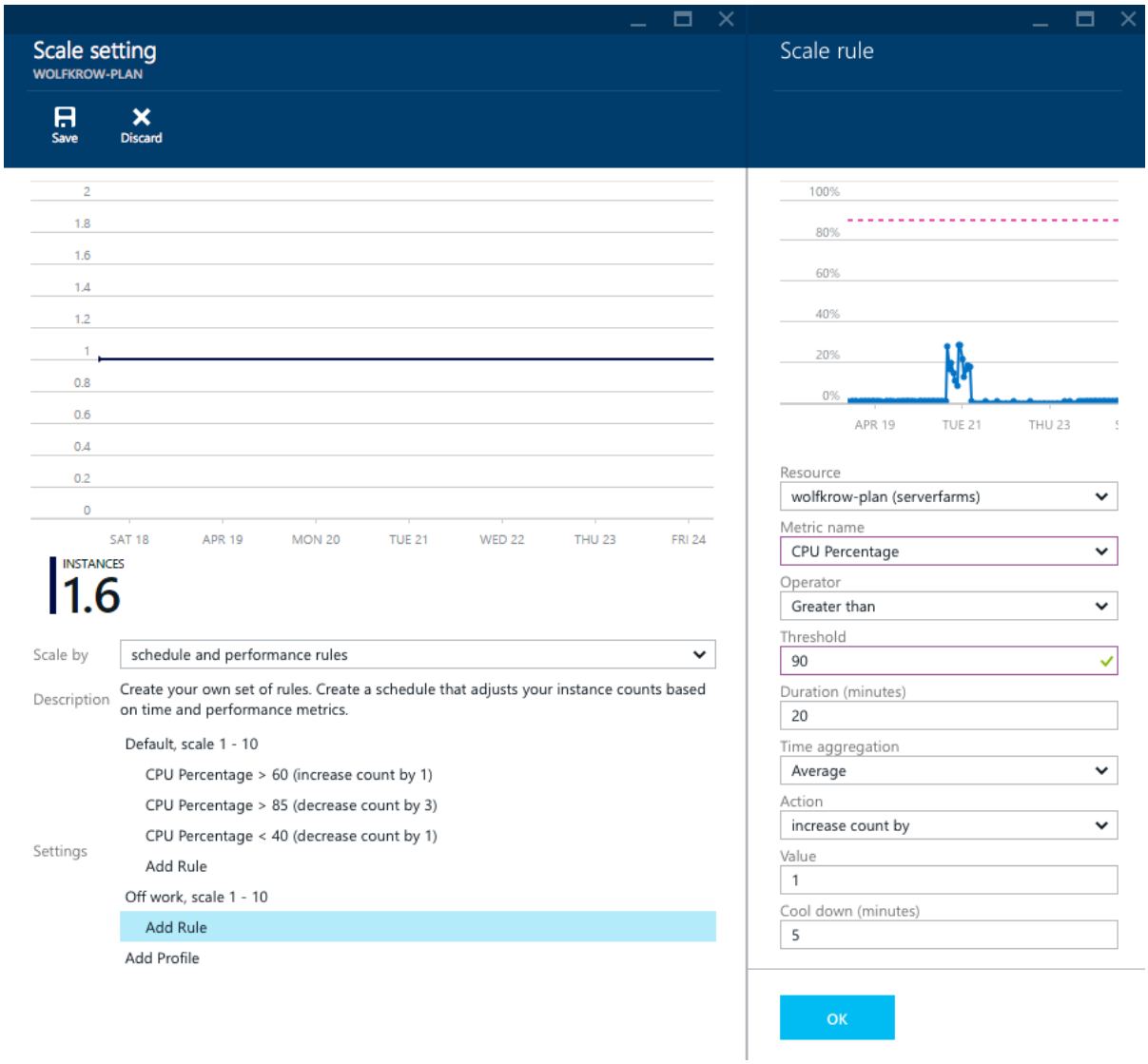



6. Name your new, second, profile, for example you could call it **Off work**.
7. Then select **recurrence** again, and choose the instance count range you want during this time.

8. As with the Default profile, choose the **Days** you want this profile to apply to, and the **Start time** during the day.

> **NOTE**
>
> Autoscale will use the Daylight savings rules for whichever **Time zone** you select. However, during Daylight savings time the UTC offset will show the base Time zone offset, not the Daylight savings UTC offset.

9. Click **OK**.
10. Now, you will need to add whatever rules you want to apply during your second profile. Click **Add Rule**, and then you could construct the same rule you have during the Default profile.




11. Be sure to create both a rule for scale out and scale in, or else during the profile the instance count will only grow (or decrease).
12. Finally, click **Save**.

## Next steps

- Monitor service metrics to make sure your service is available and responsive.
- Enable monitoring and diagnostics to collect detailed high-frequency metrics on your service.
- Receive alert notifications whenever operational events happen or metrics cross a threshold.
- Monitor application performance if you want to understand exactly how your code is performing in the cloud.
- View events and activity log to learn everything that has happened in your service.

- Monitor availability and responsiveness of any web page with Application Insights so you can find out if your page is down.

# Use portal to create an Azure Active Directory application and service principal that can access resources

12/7/2017 • 5 min to read • Edit Online

When you have an application that needs to access or modify resources, you must set up an Azure Active Directory (AD) application and assign the required permissions to it. This approach is preferable to running the app under your own credentials because:

- You can assign permissions to the app identity that are different than your own permissions. Typically, these permissions are restricted to exactly what the app needs to do.
- You do not have to change the app's credentials if your responsibilities change.
- You can use a certificate to automate authentication when executing an unattended script.

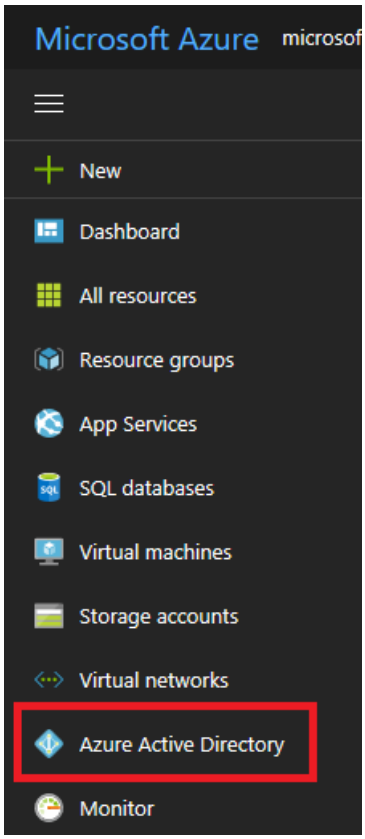
This article shows you how to perform those steps through the portal. It focuses on a single-tenant application where the application is intended to run within only one organization. You typically use single-tenant applications for line-of-business applications that run within your organization.

## Required permissions

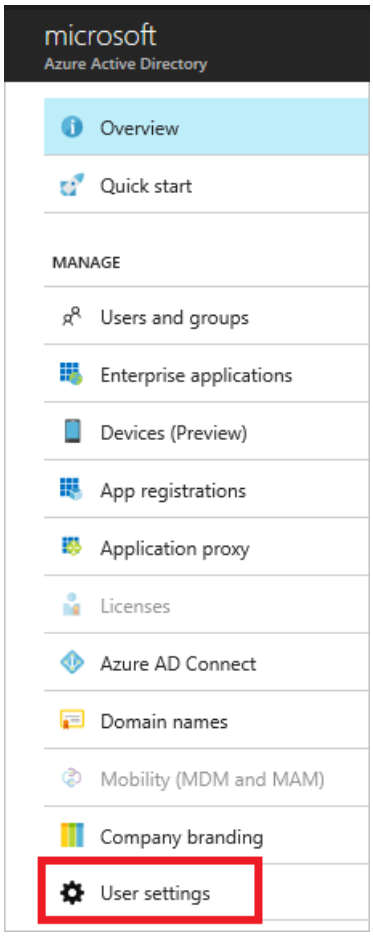
To complete this article, you must have sufficient permissions to register an application with your Azure AD tenant, and assign the application to a role in your Azure subscription. Let's make sure you have the right permissions to perform those steps.

**Check Azure Active Directory permissions**

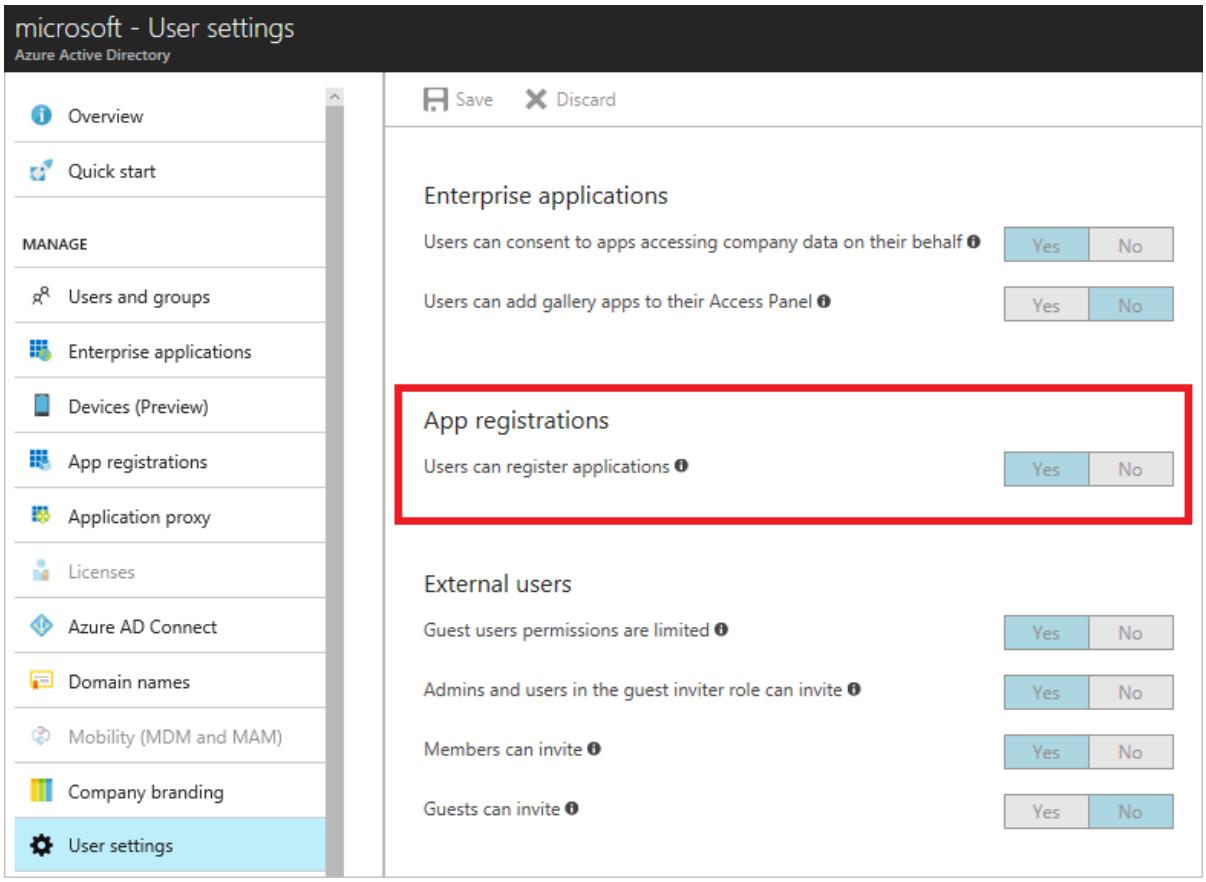
1. Log in to your Azure Account through the Azure portal.
2. Select **Azure Active Directory**.



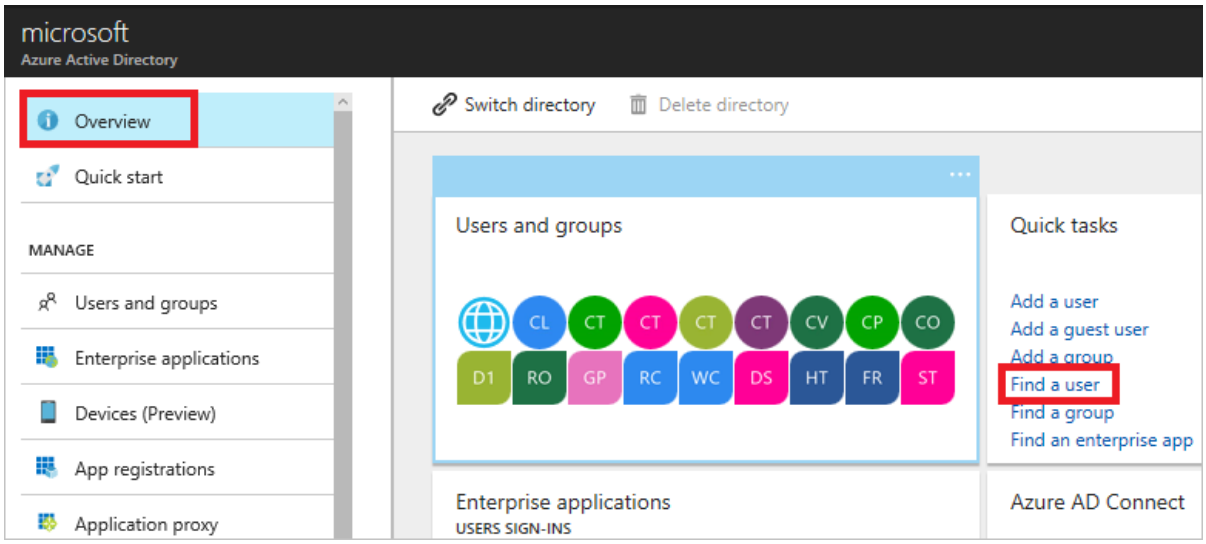

3. In Azure Active Directory, select **User settings**.



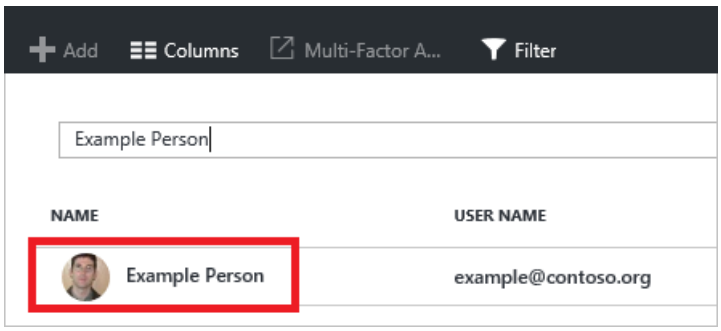

4. Check the **App registrations** setting. If set to **Yes**, non-admin users can register AD apps. This setting means any user in the Azure AD tenant can register an app. You can proceed to Check Azure subscription permissions.



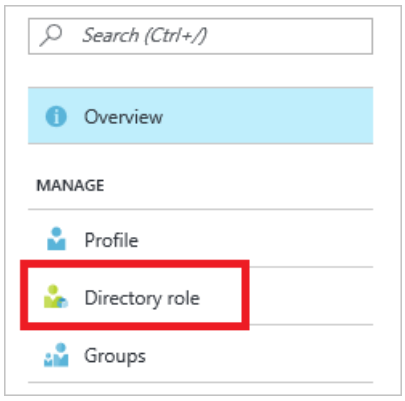

5. If the app registrations setting is set to **No**, only admin users can register apps. Check whether your account is an admin for the Azure AD tenant. Select **Overview** and **Find a user** from Quick tasks.



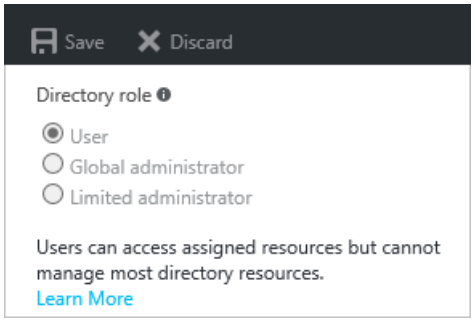

6. Search for your account, and select it when you find it.




7. For your account, select **Directory role**.




8. View your assigned directory role in Azure AD. If your account is assigned to the User role, but the app registration setting (from the preceding steps) is limited to admin users, ask your administrator to either assign you to an administrator role, or to enable users to register apps.



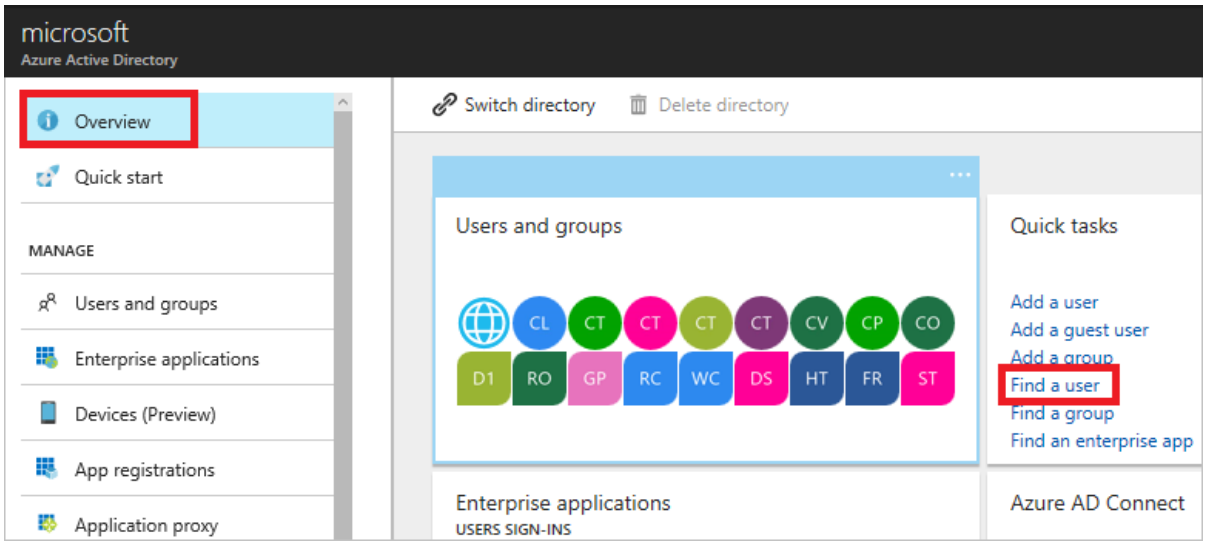

### Check Azure subscription permissions

In your Azure subscription, your account must have `Microsoft.Authorization/*/Write` access to assign an AD app to a role. This action is granted through the Owner role or User Access Administrator role. If your account is assigned to the **Contributor** role, you do not have adequate permission. You receive an error when attempting to assign the service principal to a role.

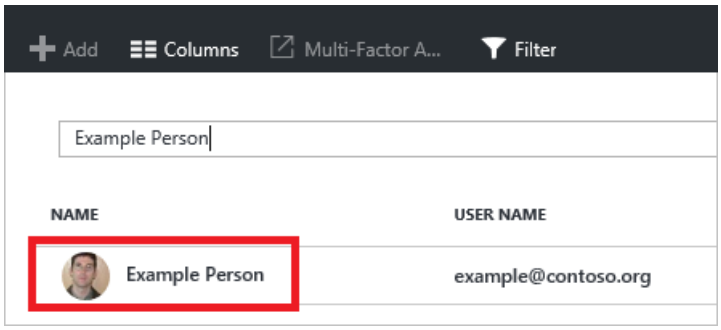
To check your subscription permissions:

1. If you are not already looking at your Azure AD account from the preceding steps, select **Azure Active Directory** from the left pane.

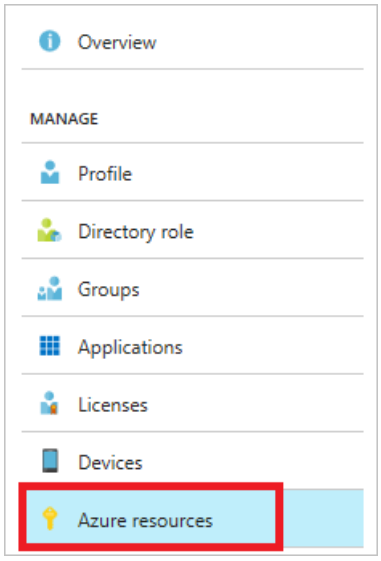
2. Find your Azure AD account. Select **Overview** and **Find a user** from Quick tasks.




3. Search for your account, and select it when you find it.



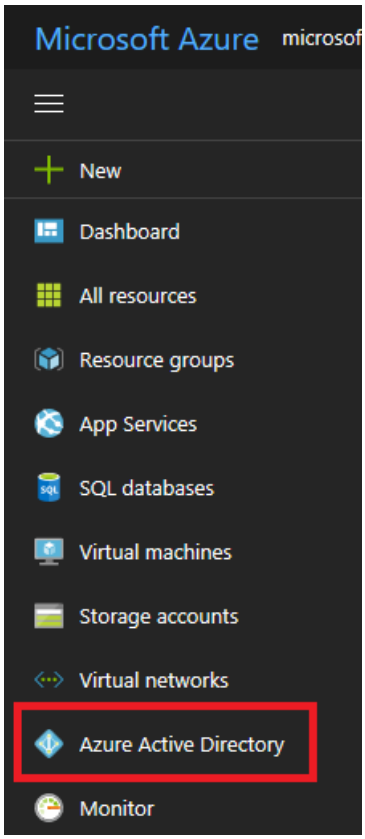

4. Select **Azure resources**.



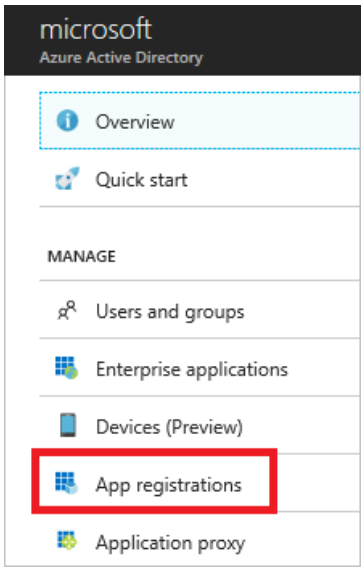

5. View your assigned roles, and determine if you have adequate permissions to assign an AD app to a role. If not, ask your subscription administrator to add you to User Access Administrator role. In the following image, the user is assigned to the Owner role for two subscriptions, which means that user has adequate permissions.


## Create an Azure Active Directory application

1. Log in to your Azure Account through the Azure portal.
2. Select **Azure Active Directory**.

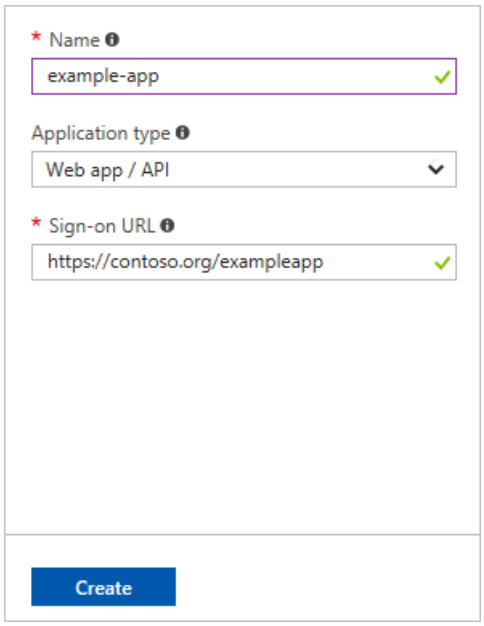



3. Select **App registrations**.




4. Select **New application registration**.


5. Provide a name and URL for the application. Select **Web app / API** for the type of application you want to create. You cannot create credentials for a **Native** application; therefore, that type does not work for an automated application. After setting the values, select **Create**.

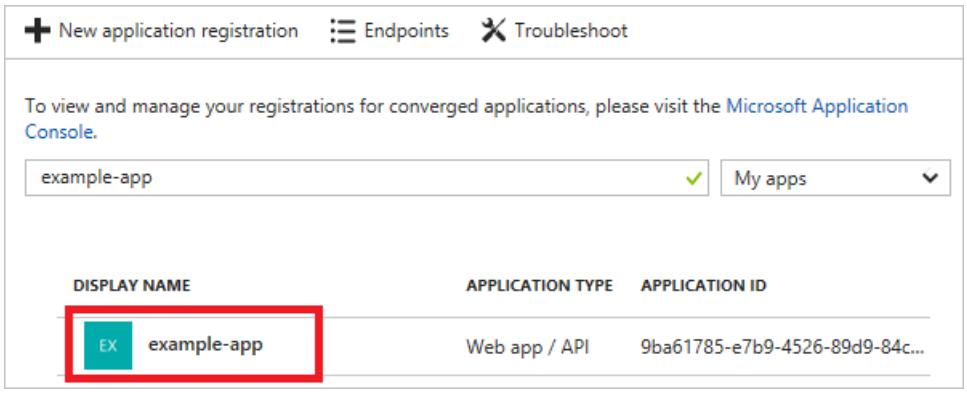



You have created your application.

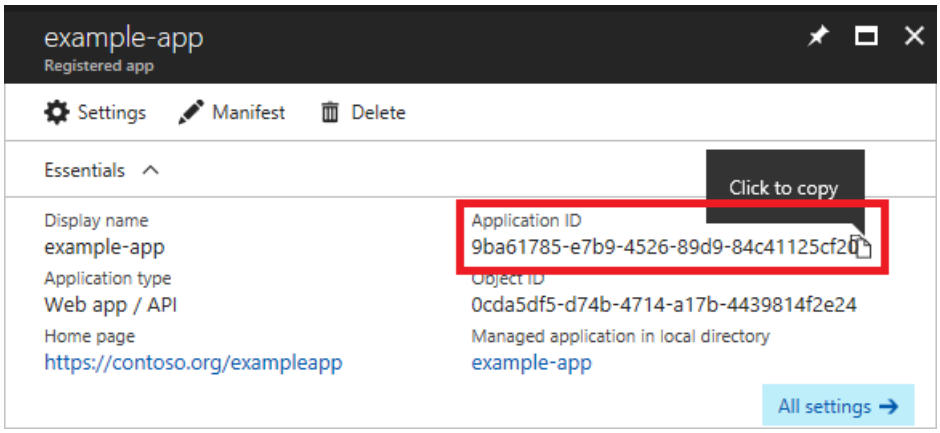
## Get application ID and authentication key

When programmatically logging in, you need the ID for your application and an authentication key. To get those values, use the following steps:

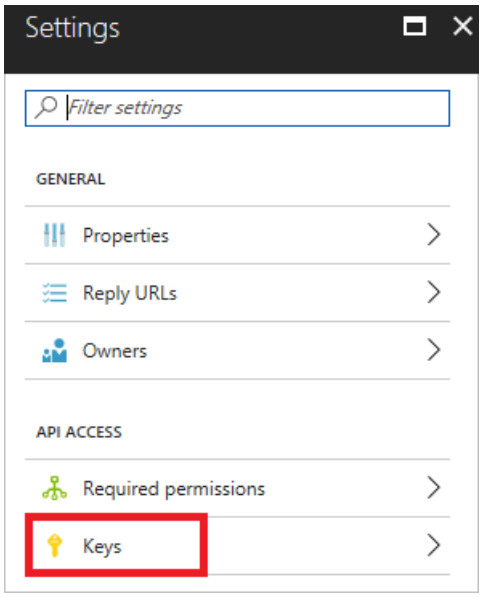
1. From **App registrations** in Azure Active Directory, select your application.



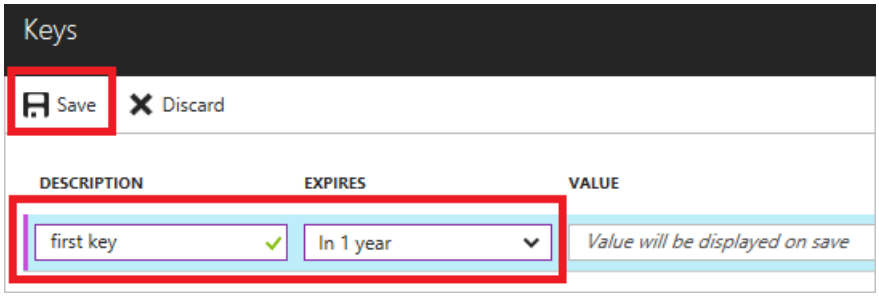

2. Copy the **Application ID** and store it in your application code. Some sample applications refer to this value as the client ID.



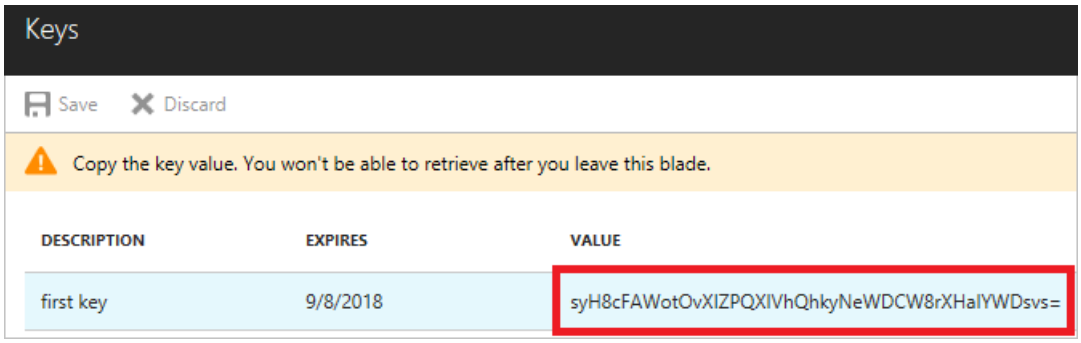

3. To generate an authentication key, select **Keys**.




4. Provide a description of the key, and a duration for the key. When done, select **Save**.



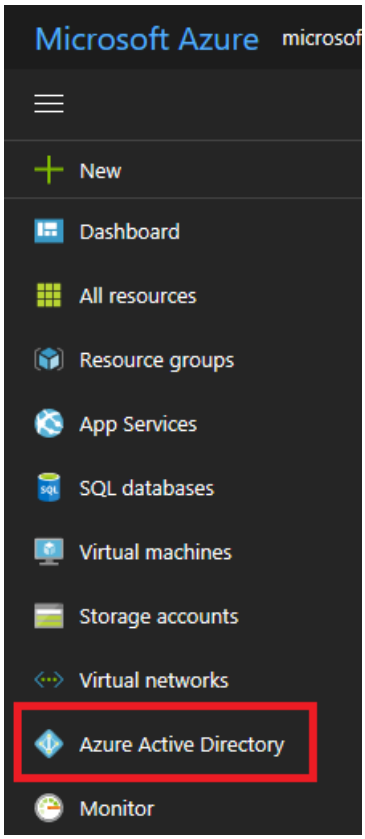

After saving the key, the value of the key is displayed. Copy this value because you are not able to retrieve the key later. You provide the key value with the application ID to log in as the application. Store the key value where your application can retrieve it.



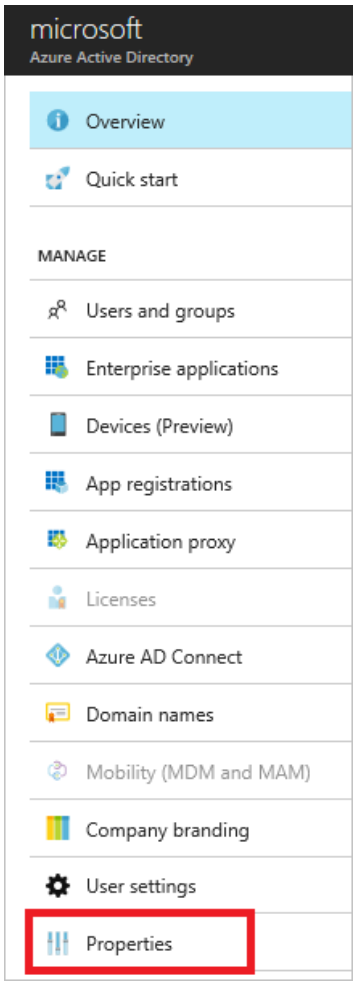

## Get tenant ID

When programmatically logging in, you need to pass the tenant ID with your authentication request.

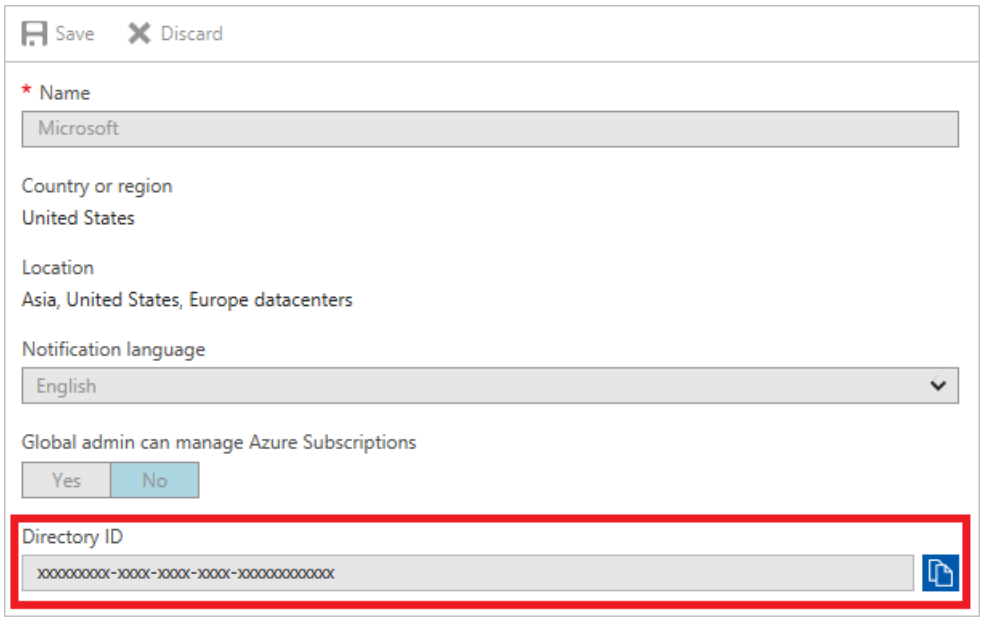
1. Select **Azure Active Directory**.




2. To get the tenant ID, select **Properties** for your Azure AD tenant.




3. Copy the **Directory ID**. This value is your tenant ID.

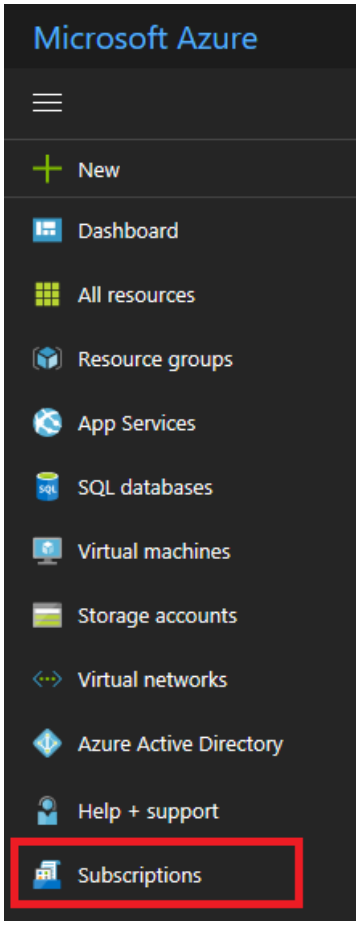



## Assign application to role

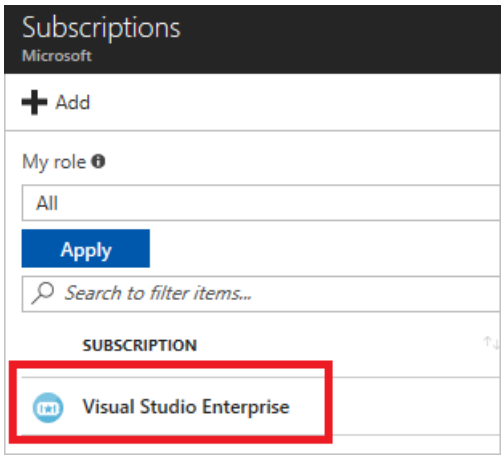
To access resources in your subscription, you must assign the application to a role. Decide which role represents the right permissions for the application. To learn about the available roles, see RBAC: Built in Roles.

You can set the scope at the level of the subscription, resource group, or resource. Permissions are inherited to lower levels of scope. For example, adding an application to the Reader role for a resource group means it can read the resource group and any resources it contains.

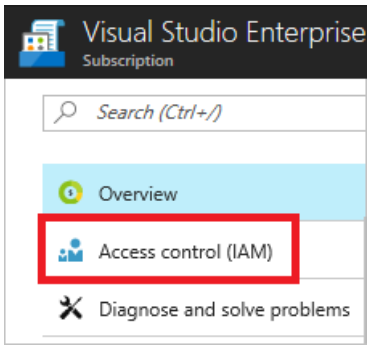
1. Navigate to the level of scope you wish to assign the application to. For example, to assign a role at the subscription scope, select **Subscriptions**. You could instead select a resource group or resource.



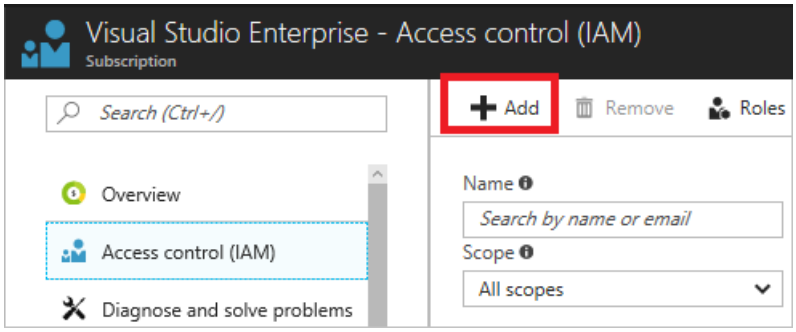

2. Select the particular subscription (resource group or resource) to assign the application to.



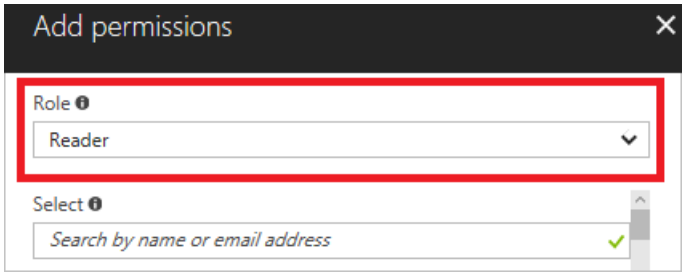

3. Select **Access Control (IAM)**.



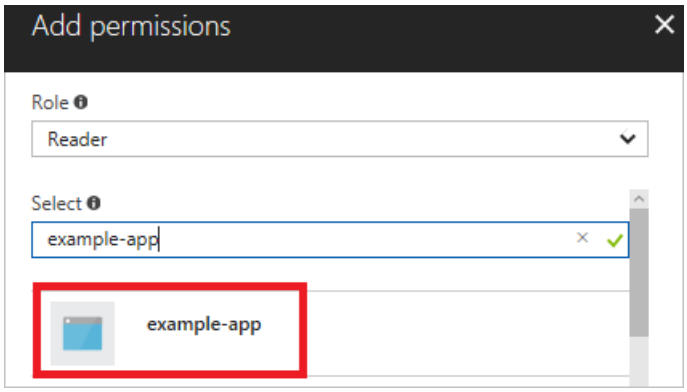

4. Select **Add**.




5. Select the role you wish to assign to the application. The following image shows the **Reader** role.




6. Search for your application, and select it.




7. Select **Save** to finish assigning the role. You see your application in the list of users assigned to a role for that scope.

## Log in as the application

Your application is now set up in Azure Active Directory. You have an ID and key to use for signing in as the application. The application is assigned to a role that gives it certain actions it can perform. For information about logging in as the application through different platforms, see:

- PowerShell
- Azure CLI
- REST
- .NET
- Java
- Node.js
- Python
- Ruby

## Next steps

- To set up a multi-tenant application, see Developer's guide to authorization with the Azure Resource Manager API.
- To learn about specifying security policies, see Azure Role-based Access Control.
- For a list of available actions that can be granted or denied to users, see Azure Resource Manager Resource Provider operations.

# Overview of Metrics in Microsoft Azure

7/18/2017 • 3 min to read • Edit Online

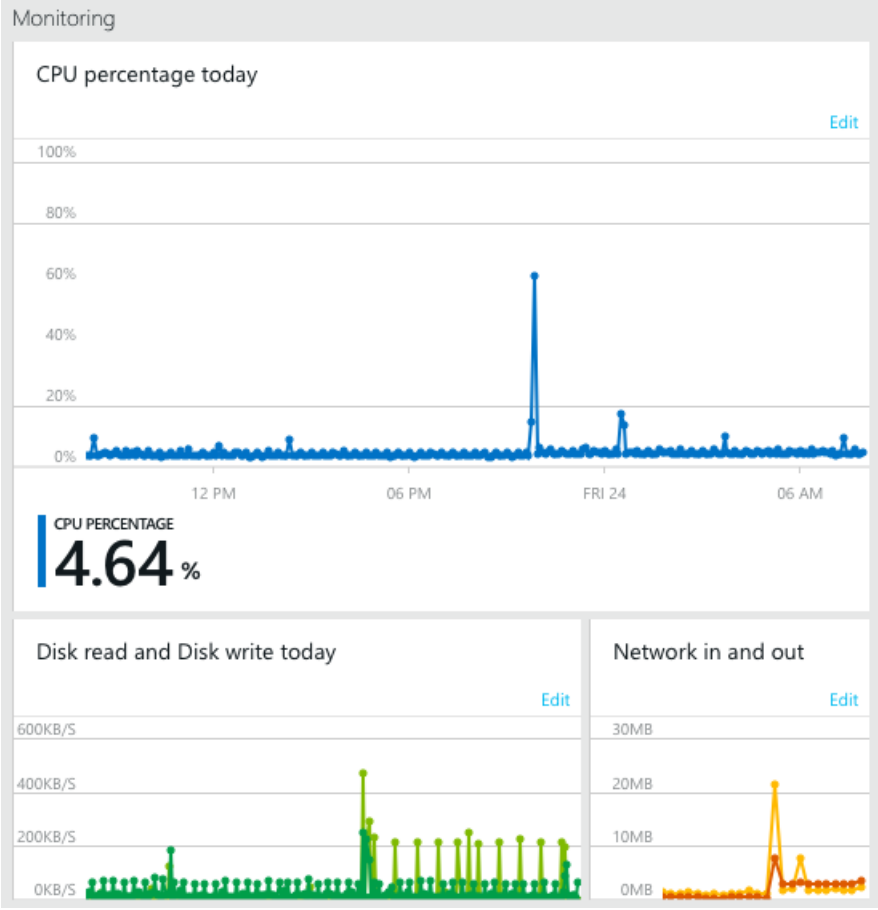
All Azure services track key metrics that allow you to monitor the health, performance, availability and usage of your services. You can view these metrics in the Azure portal, and you can also use the REST API or .NET SDK to access the full set of metrics programmatically.

For some services, you may need to turn on diagnostics in order to see any metrics. For others, such as virtual machines, you will get a basic set of metrics, but need to enable the full set high-frequency metrics. See Enable monitoring and diagnostics to learn more.

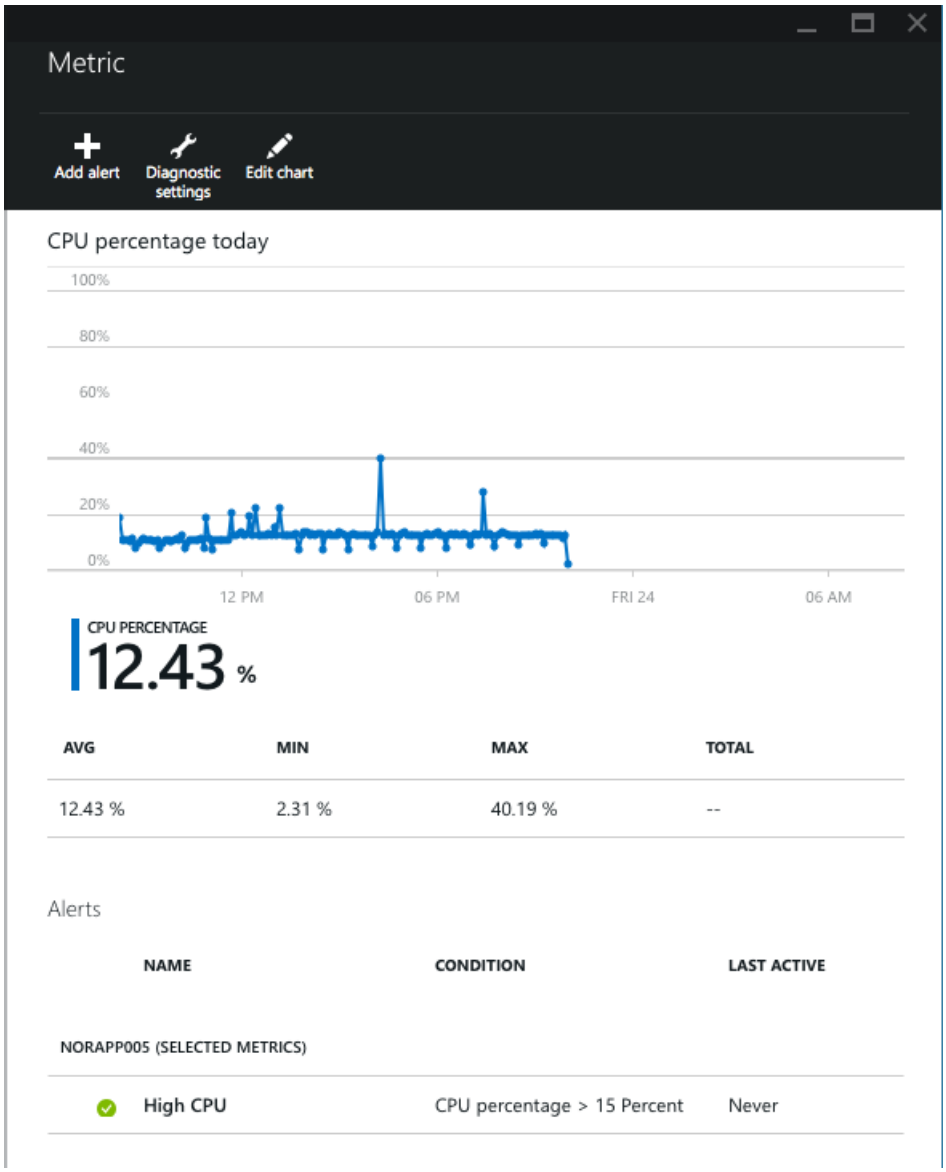
## Using monitoring charts

You can chart any of the metrics them over any time period you choose.

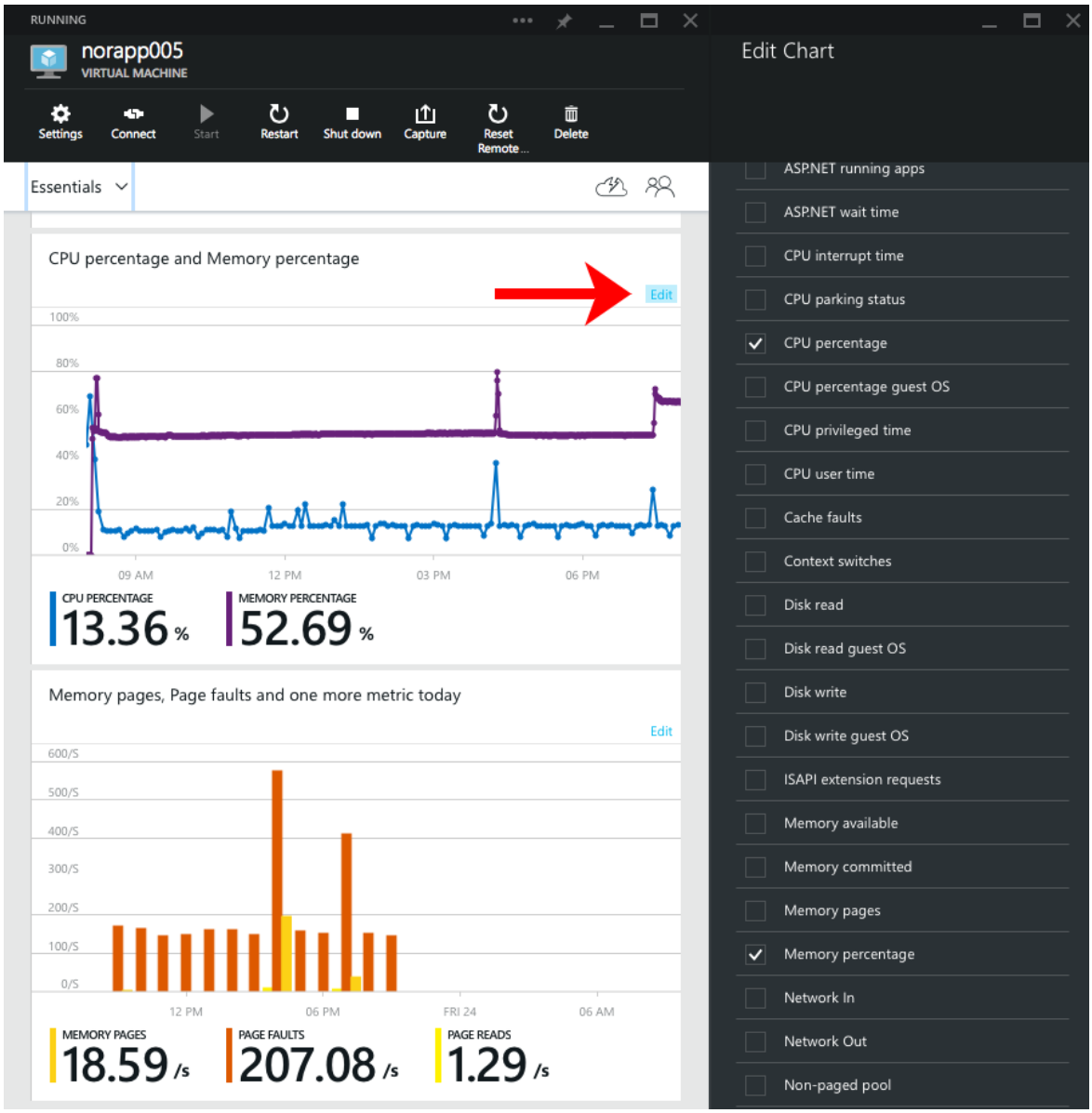
1. In the Azure Portal, click **Browse**, and then a resource you're interested in monitoring.
2. The **Monitoring** section contains the most important metrics for each Azure resource. For example, a web app has **Requests and Errors**, where as a virtual machine would have **CPU percentage** and **Disk read and write**:



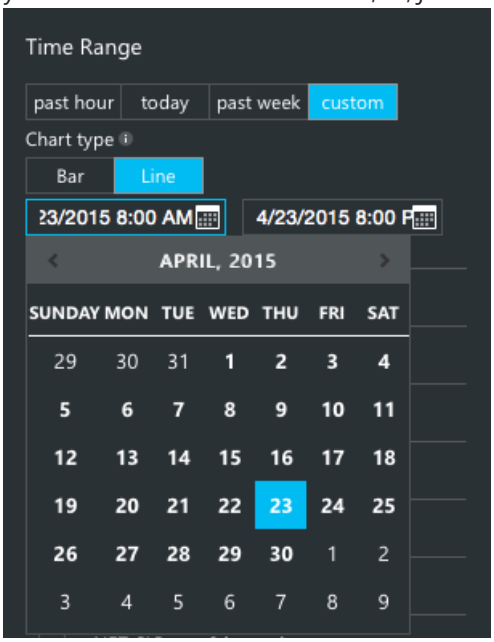

3. Clicking on any chart will show you the **Metric** blade. On the blade, in addition to the graph, is a table that shows you aggregations of the metrics (such as average, minimum and maximum, over the time range you chose). Below that are the alert rules for the resource.




4. To customize the lines that appear, click the **Edit** button on the chart, or, the **Edit chart** command on the Metric blade.
5. On the Edit Query blade you can do three things:
    - Change the time range
    - Switch the appearance between Bar and Line
    - Choose different metics




6. Changing the time range is as easy as selecting a different range (such as **Past Hour**) and clicking **Save** at the bottom of the blade. You can also choose **Custom**, which allows you to choose any period of time over the past 2 weeks. For example, you can see the whole two weeks, or, just 1 hour from yesterday. Type in the text box to enter a different hour.

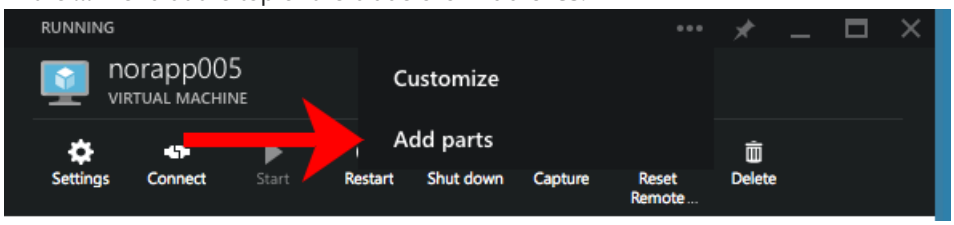



7. Below the time range, you chan choose any number of metrics to show on the chart.
8. When you click Save your changes will be saved for that particular resource. For example, if you have two virtual machines, and you change a chart on one, it will not impact the other.

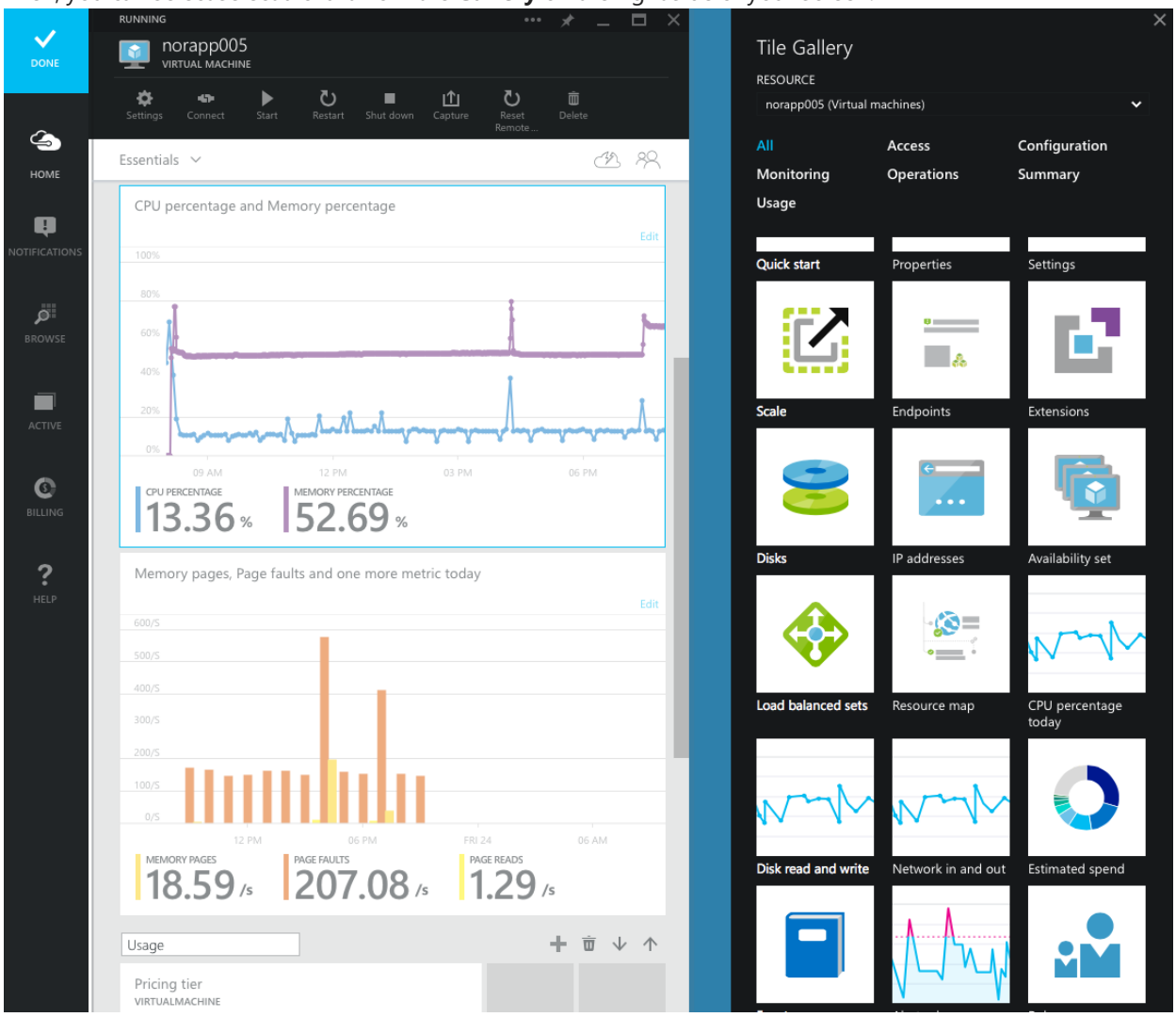
# Creating side-by-side charts

With the powerful customization in the portal you can add as many charts as you want.

1. In the **...** menu at the top of the blade click **Add tiles**:




2. Then, you can select select a chart from the **Gallery** on the right side of your screen:

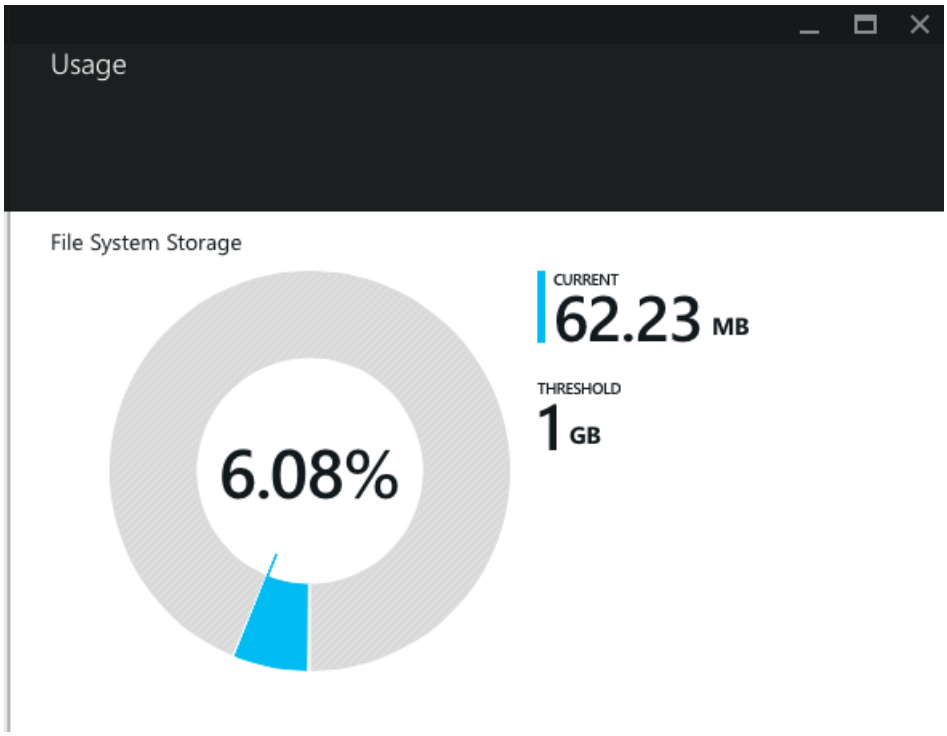



3. If you don't see the metric you want, you can always add one of the preset metrics, and **Edit** the chart to show the metric that you need.

# Monitoring usage quotas

Most metrics show you trends over time, but certain data, like usage quotas, are point-in-time information with a threshold.

You can also see usage quotas on the blade for resources that have quotas:




Like with metrics, you can use the REST API or .NET SDK to access the full set of usage quotas programmatically.

## Next steps

- Receive alert notifications whenever a metric crosses a threshold.
- Enable monitoring and diagnostics to collect detailed high-frequency metrics on your service.
- Scale instance count automatically to make sure your service is available and responsive.
- Monitor application performance if you want to understand exactly how your code is performing in the cloud.
- Use Application Insights for JavaScript apps and web pages to get client analytics about the browsers that visit a web page.
- Monitor availability and responsiveness of any web page with Application Insights so you can find out if your page is down.

4 min to read •

# Monitor availability and responsiveness of any web site

12/15/2017 • 13 min to read • Edit Online

After you've deployed your web app or web site to any server, you can set up tests to monitor its availability and responsiveness. Azure Application Insights sends web requests to your application at regular intervals from points around the world. It alerts you if your application doesn't respond, or responds slowly.

You can set up availability tests for any HTTP or HTTPS endpoint that is accessible from the public internet. You don't have to add anything to the web site you're testing. It doesn't even have to be your site: you could test a REST API service on which you depend.

There are two types of availability tests:

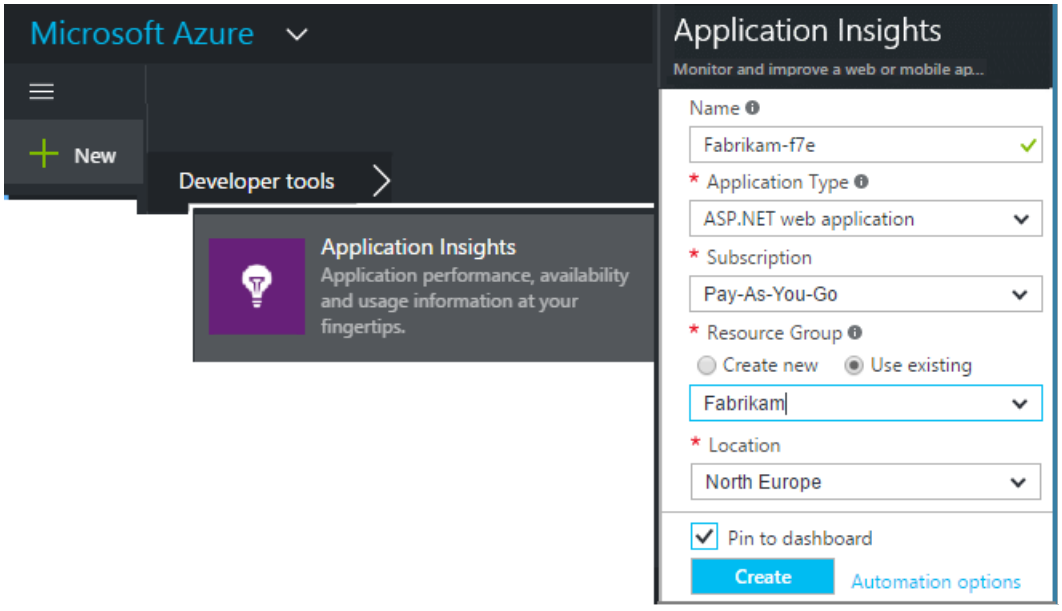
- URL ping test: a simple test that you can create in the Azure portal.
- Multi-step web test: which you create in Visual Studio Enterprise and upload to the portal.

You can create up to 100 availability tests per application resource.

## Open a resource for your availability test reports

**If you have already configured Application Insights** for your web app, open its Application Insights resource in the Azure portal.

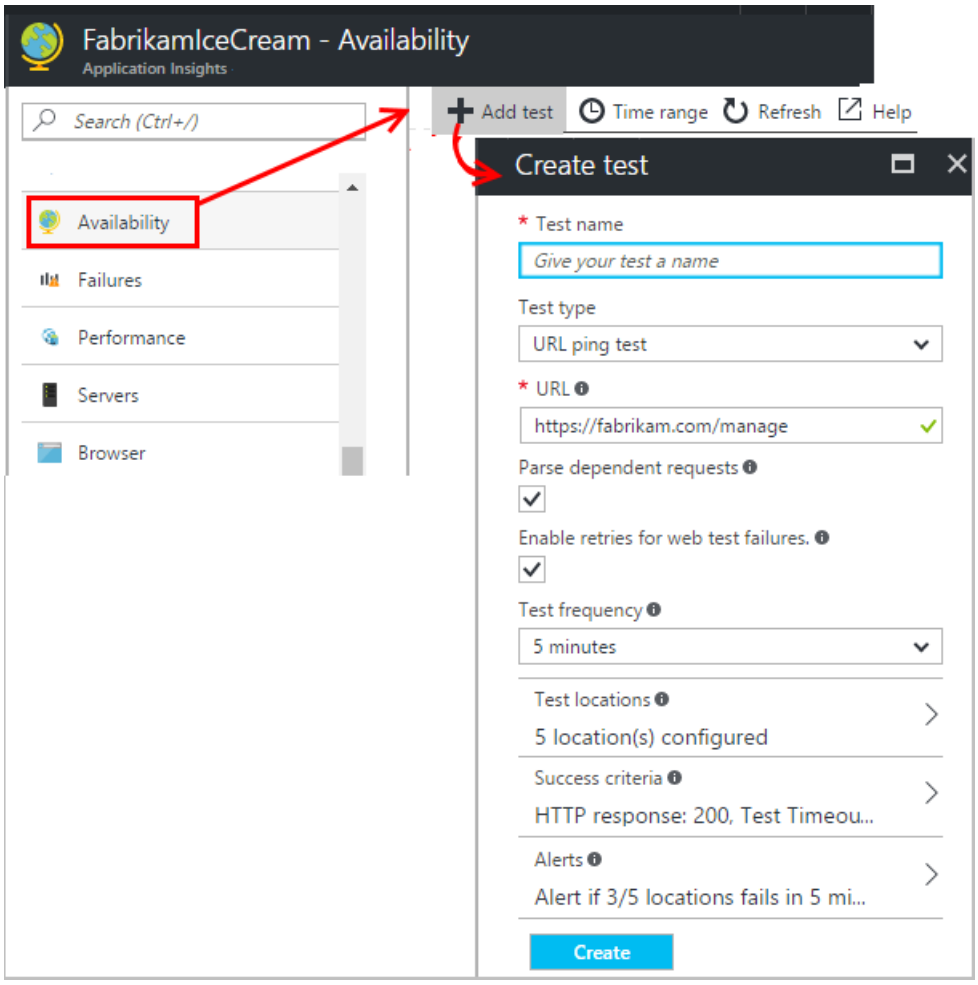
**Or, if you want to see your reports in a new resource,** sign up to Microsoft Azure, go to the Azure portal, and create an Application Insights resource.




Click **All resources** to open the Overview blade for the new resource.

## Create a URL ping test

Open the Availability blade and add a test.




- **The URL** can be any web page you want to test, but it must be visible from the public internet. The URL can include a query string. So, for example, you can exercise your database a little. If the URL resolves to a redirect, we follow it up to 10 redirects.
- **Parse dependent requests**: If this option is checked, the test requests images, scripts, style files, and other files that are part of the web page under test. The recorded response time includes the time taken to get these files. The test fails if all these resources cannot be successfully downloaded within the timeout for the whole test.

  If the option is not checked, the test only requests the file at the URL you specified.

- **Enable retries**: If this option is checked, when the test fails, it is retried after a short interval. A failure is reported only if three successive attempts fail. Subsequent tests are then performed at the usual test frequency. Retry is temporarily suspended until the next success. This rule is applied independently at each test location. We recommend this option. On average, about 80% of failures disappear on retry.
- **Test frequency**: Sets how often the test is run from each test location. With a frequency of five minutes and five test locations, your site is tested on average every minute.
- **Test locations** are the places from where our servers send web requests to your URL. Choose more than one so that you can distinguish problems in your website from network issues. You can select up to 16 locations.
- **Success criteria**:

  **Test timeout**: Decrease this value to be alerted about slow responses. The test is counted as a failure if the responses from your site have not been received within this period. If you selected **Parse dependent requests**, then all the images, style files, scripts, and other dependent resources must have been received within this period.

  **HTTP response**: The returned status code that is counted as a success. 200 is the code that indicates that a normal web page has been returned.

**Content match**: a string, like "Welcome!" We test that an exact case-sensitive match occurs in every response. It must be a plain string, without wildcards. Don't forget that if your page content changes you might have to update it.

- **Alerts** are, by default, sent to you if there are failures in three locations over five minutes. A failure in one location is likely to be a network problem, and not a problem with your site. But you can change the threshold to be more or less sensitive, and you can also change who the emails should be sent to.

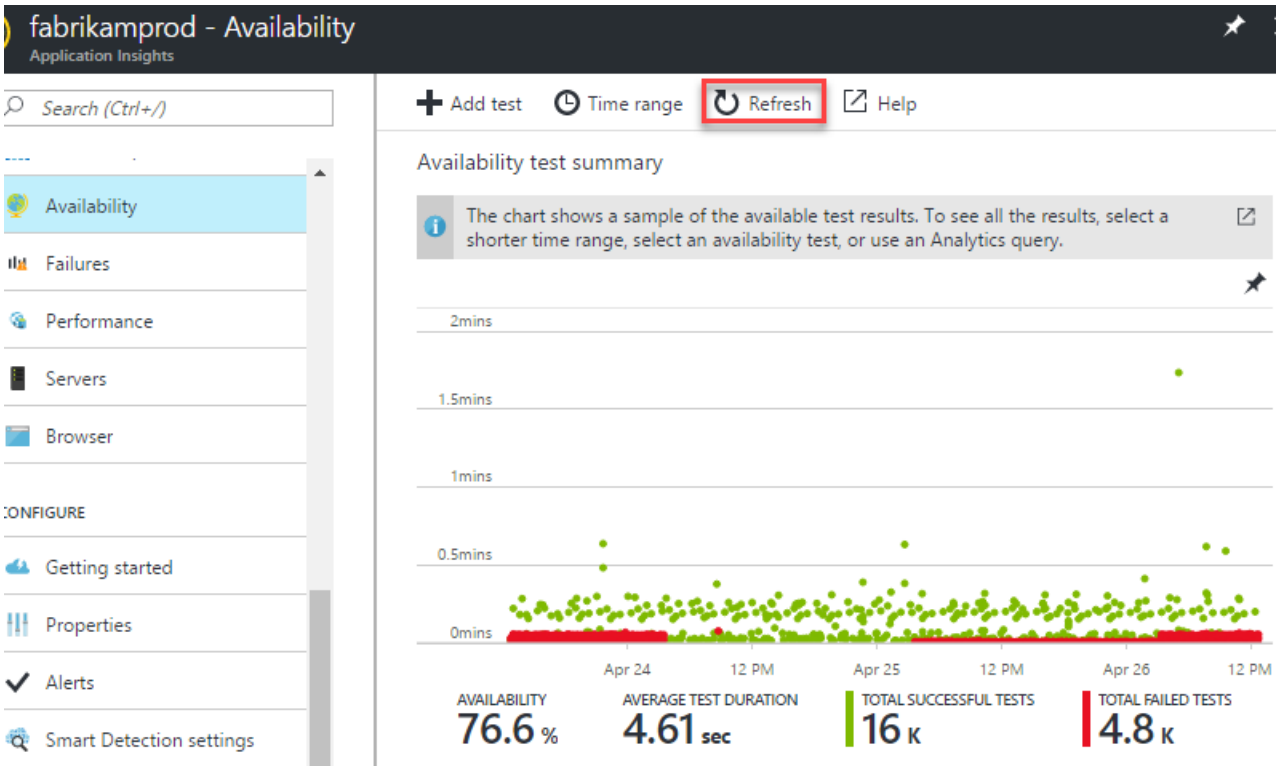
  You can set up a webhook that is called when an alert is raised. (But note that, at present, query parameters are not passed through as Properties.)

**Test more URLs**

Add more tests. For example, In addition to testing your home page, you can make sure your database is running by testing the URL for a search.

## See your availability test results

After a few minutes, click **Refresh** to see test results.




The scatterplot shows samples of the test results that have diagnostic test-step detail in them. The test engine stores diagnostic detail for tests that have failures. For successful tests, diagnostic details are stored for a subset of the executions. Hover over any of the green/red dots to see the test timestamp, test duration, location, and test name. Click through any dot in the scatter plot to see the details of the test result.

Select a particular test, location, or reduce the time period to see more results around the time period of interest. Use Search Explorer to see results from all executions, or use Analytics queries to run custom reports on this data.

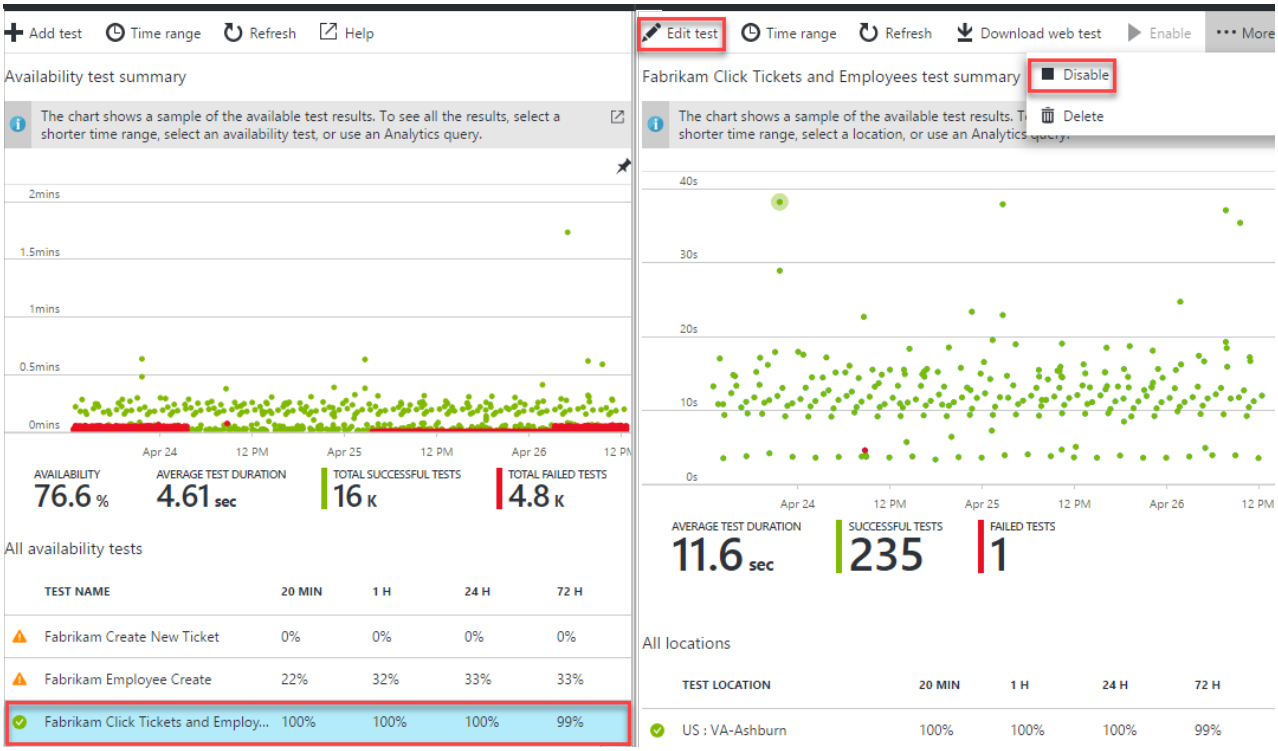
In addition to the raw results, there are two Availability metrics in Metrics Explorer:

1. Availability: Percentage of the tests that were successful, across all test executions.
2. Test Duration: Average test duration across all test executions.

You can apply filters on the test name, location to analyze trends of a particular test and/or location.

## Inspect and edit tests

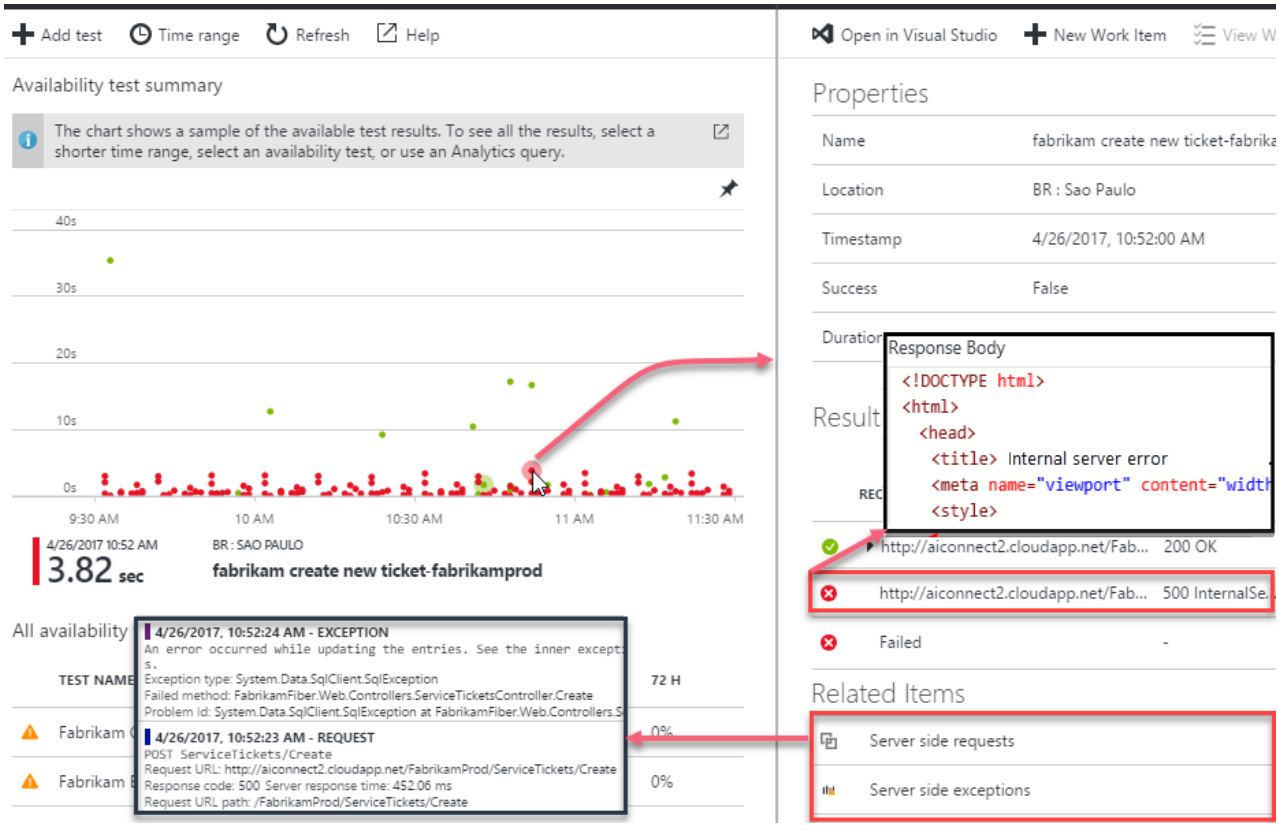
From the summary page, select a specific test. There, you can see its specific results, and edit or temporarily disable it.




You might want to disable availability tests or the alert rules associated with them while you are performing maintenance on your service.

## If you see failures

Click a red dot.




From an availability test result, you can:

- Inspect the response received from your server.

- Diagnose failure with server side telemetry collected while processing the failed request instance.
- Log an issue or work item in Git or VSTS to track the problem. The bug will contain a link to this event.
- Open the web test result in Visual Studio.

*Looks OK but reported as a failure?* See FAQ for ways to reduce noise.

## Multi-step web tests

You can monitor a scenario that involves a sequence of URLs. For example, if you are monitoring a sales website, you can test that adding items to the shopping cart works correctly.

> **NOTE**
> There is a charge for multi-step web tests. Pricing scheme.

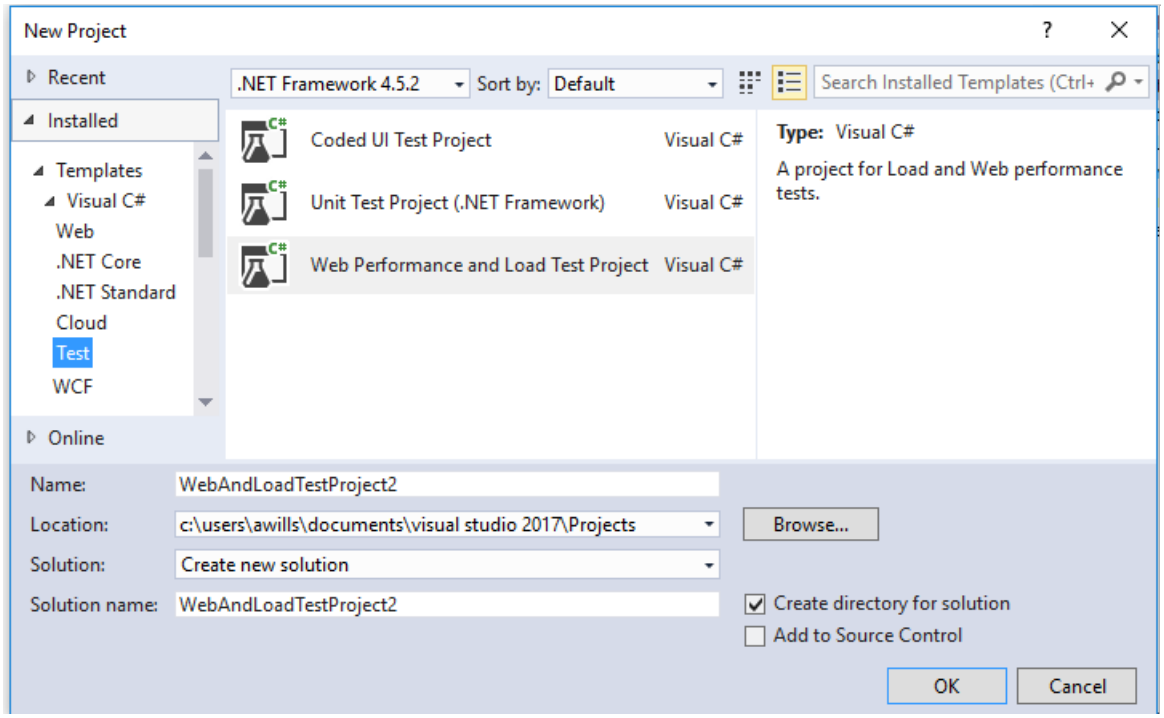
To create a multi-step test, you record the scenario by using Visual Studio Enterprise, and then upload the recording to Application Insights. Application Insights replays the scenario at intervals and verifies the responses.

> **NOTE**
> You can't use coded functions or loops in your tests. The test must be contained completely in the .webtest script. However, you can use standard plugins.

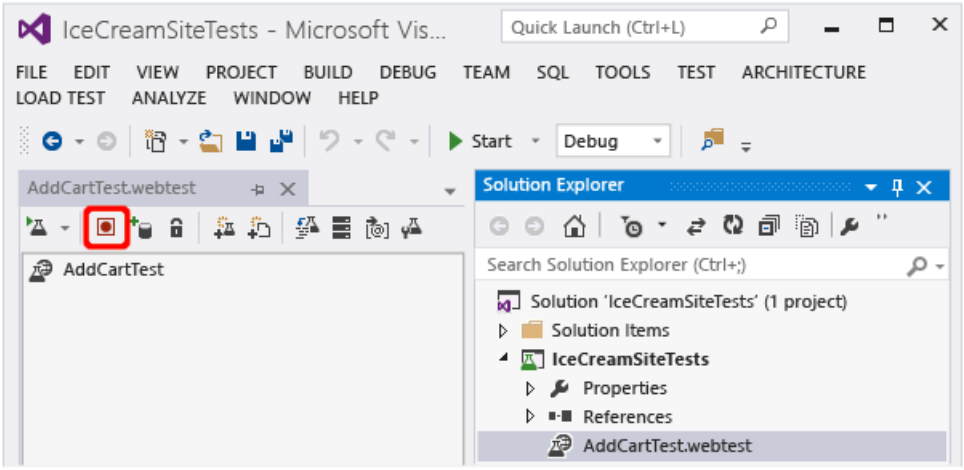
#### 1. Record a scenario

Use Visual Studio Enterprise to record a web session.

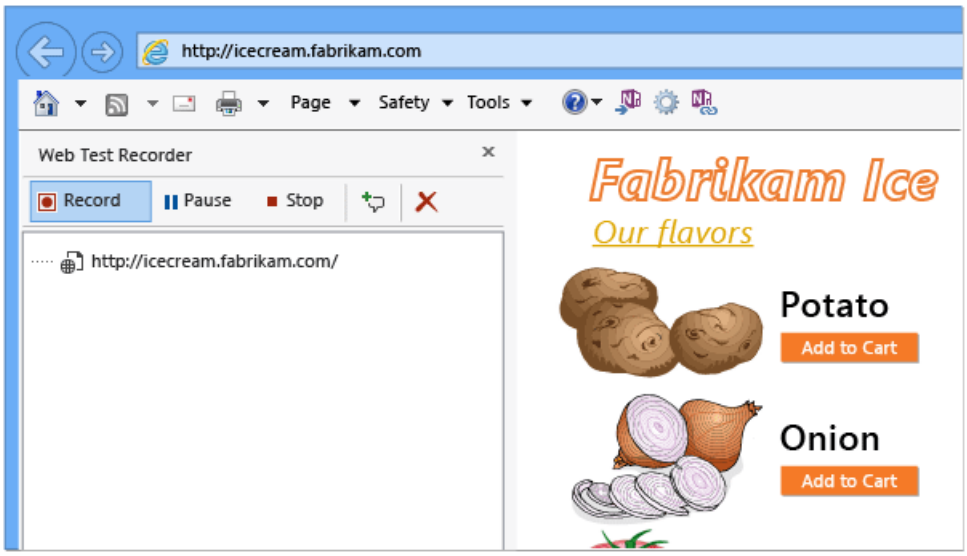
1. Create a Web performance test project.




   - *Don't see the Web Performance and Load Test template?* - Close Visual Studio Enterprise. Open **Visual Studio Installer** to modify your Visual Studio Enterprise installation. Under **Individual Components**, select **Web Performance and load testing tools**.
2. Open the .webtest file and start recording.




3. Do the user actions you want to simulate in your test: open your website, add a product to the cart, and so on. Then stop your test.




Don't make a long scenario. There's a limit of 100 steps and 2 minutes.

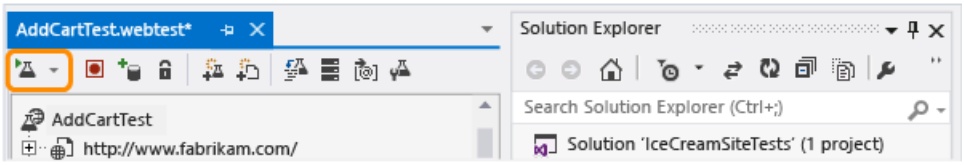
4. Edit the test to:

   - Add validations to check the received text and response codes.
   - Remove any superfluous interactions. You could also remove dependent requests for pictures or to ad or tracking sites.

     Remember that you can only edit the test script - you can't add custom code or call other web tests. Don't insert loops in the test. You can use standard web test plug-ins.

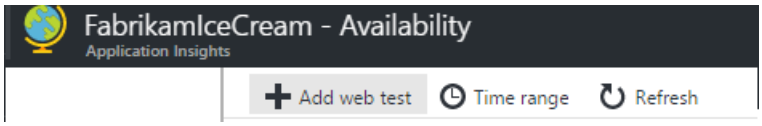
5. Run the test in Visual Studio to make sure it works.

   The web test runner opens a web browser and repeats the actions you recorded. Make sure it works as you expect.

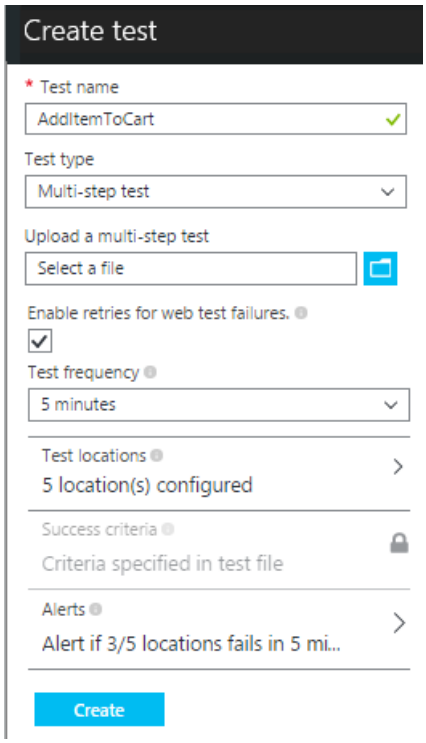



**2. Upload the web test to Application Insights**

1. In the Application Insights portal, create a web test.




2. Select multi-step test, and upload the .webtest file.




Set the test locations, frequency, and alert parameters in the same way as for ping tests.

#### 3. See the results

View your test results and any failures in the same way as single-url tests.

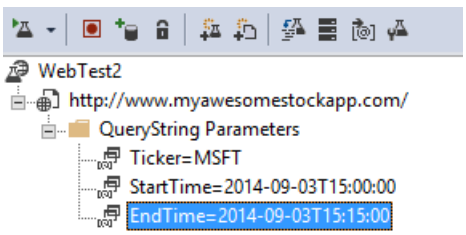
In addition, you can download the test results to view them in Visual Studio.

#### Too many failures?

- A common reason for failure is that the test runs too long. It mustn't run longer than two minutes.
- Don't forget that all the resources of a page must load correctly for the test to succeed, including scripts, style sheets, images, and so forth.
- The web test must be entirely contained in the .webtest script: you can't use coded functions in the test.

### Plugging time and random numbers into your multi-step test

Suppose you're testing a tool that gets time-dependent data such as stocks from an external feed. When you record your web test, you have to use specific times, but you set them as parameters of the test, StartTime and EndTime.

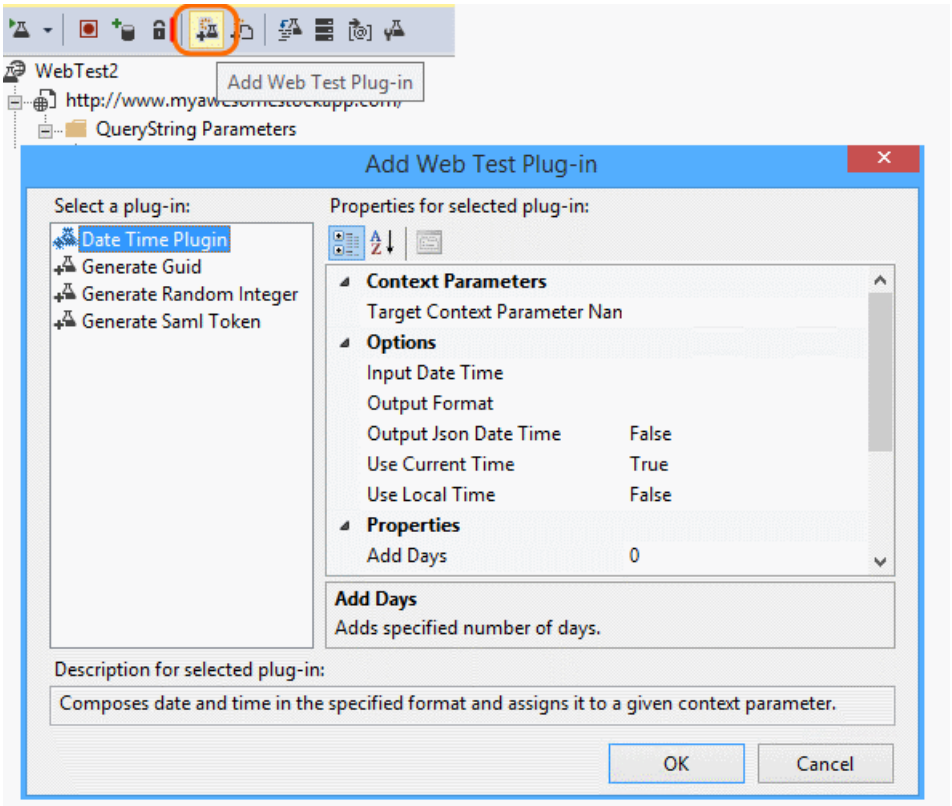



When you run the test, you'd like EndTime always to be the present time, and StartTime should be 15 minutes ago.

Web Test Plug-ins provide the way to do parameterize times.

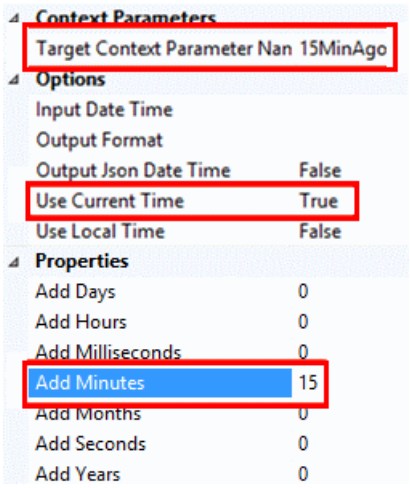
1. Add a web test plug-in for each variable parameter value you want. In the web test toolbar, choose **Add**

**Web Test Plugin**.

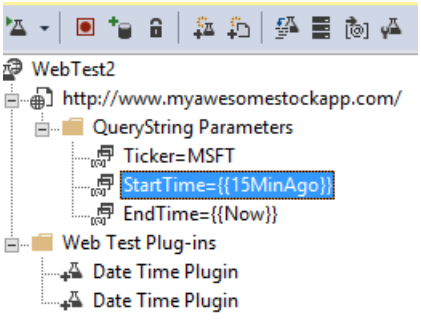



In this example, we use two instances of the Date Time Plug-in. One instance is for "15 minutes ago" and another for "now."

2. Open the properties of each plug-in. Give it a name and set it to use the current time. For one of them, set Add Minutes = -15.




3. In the web test parameters, use {{plug-in name}} to reference a plug-in name.




Now, upload your test to the portal. It uses the dynamic values on every run of the test.

# Dealing with sign-in

If your users sign in to your app, you have various options for simulating sign-in so that you can test pages behind the sign-in. The approach you use depends on the type of security provided by the app.

In all cases, you should create an account in your application just for the purpose of testing. If possible, restrict the permissions of this test account so that there's no possibility of the web tests affecting real users.

### Simple username and password

Record a web test in the usual way. Delete cookies first.

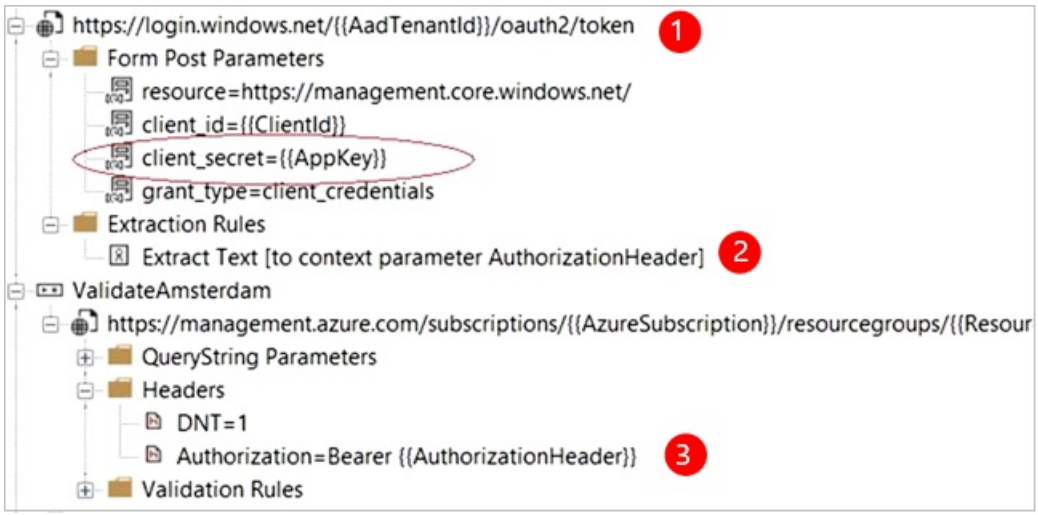
### SAML authentication

Use the SAML plugin that is available for web tests.

### Client secret

If your app has a sign-in route that involves a client secret, use that route. Azure Active Directory (AAD) is an example of a service that provides a client secret sign-in. In AAD, the client secret is the App Key.

Here's a sample web test of an Azure web app using an app key:




1. Get token from AAD using client secret (AppKey).
2. Extract bearer token from response.
3. Call API using bearer token in the authorization header.

Make sure that the web test is an actual client - that is, it has its own app in AAD - and use its clientId + appkey. Your service under test also has its own app in AAD: the appID URI of this app is reflected in the web test in the "resource" field.

### Open Authentication

An example of open authentication is signing in with your Microsoft or Google account. Many apps that use OAuth provide the client secret alternative, so your first tactic should be to investigate that possibility.

If your test must sign in using OAuth, the general approach is:

- Use a tool such as Fiddler to examine the traffic between your web browser, the authentication site, and your app.
- Perform two or more sign-ins using different machines or browsers, or at long intervals (to allow tokens to expire).
- By comparing different sessions, identify the token passed back from the authenticating site, that is then passed to your app server after sign-in.
- Record a web test using Visual Studio.

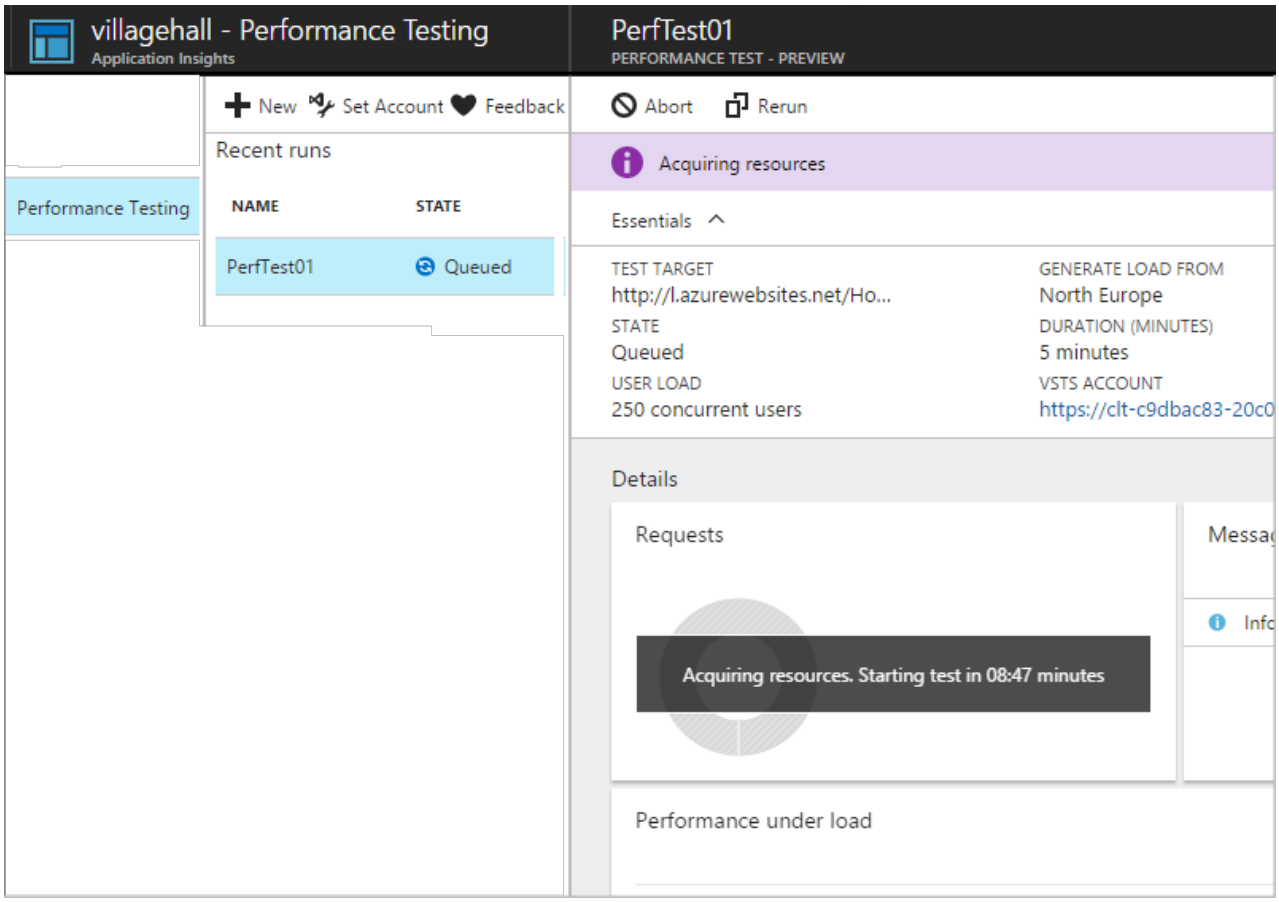
- Parameterize the tokens, setting the parameter when the token is returned from the authenticator, and using it in the query to the site. (Visual Studio attempts to parameterize the test, but does not correctly parameterize the tokens.)

## Performance tests

You can run a load test on your website. Like the availability test, you can send either simple requests or multi-step requests from our points around the world. Unlike an availability test, many requests are sent, simulating multiple simultaneous users.

From the Overview blade, open **Settings**, **Performance Tests**. When you create a test, you are invited to connect to or create a Visual Studio Team Services account.

When the test is complete, you are shown response times and success rates.




**TIP**

To observe the effects of a performance test, use Live Stream and Profiler.

## Automation

- Use PowerShell scripts to set up an availability test automatically.
- Set up a webhook that is called when an alert is raised.

## Questions? Problems?

- *Intermittent test failure with a protocol violation error?*

  The error ("protocol violation..CR must be followed by LF") indicates an issue with the server (or dependencies). This happens when malformed headers are set in the response. It can be caused by load

balancers or CDNs. Specifically, some headers might not be using CRLF to indicate end-of-line, which violates the HTTP specification and therefore fail validation at the .NET WebRequest level. Inspect the response to spot headers which might be in violation.

Note: The URL may not fail on browsers that have a relaxed validation of HTTP headers. See this blog post for a detailed explanation of this issue: http://mehdi.me/a-tale-of-debugging-the-linkedin-api-net-and-http-protocol-violations/

- *Site looks okay but I see test failures?*
  - Check all the images, scripts, style sheets, and any other files loaded by the page. If any of them fails, the test is reported as failed, even if the main html page loads OK. To desensitize the test to such resource failures, simply uncheck the "Parse Dependent Requests" from the test configuration.
  - To reduce odds of noise from transient network blips etc., ensure "Enable retries for test failures" configuration is checked. You can also test from more locations and manage alert rule threshold accordingly to prevent location specific issues causing undue alerts.
- *I don't see any related server side telemetry to diagnose test failures?*

  If you have Application Insights set up for your server-side application, that may be because sampling is in operation.
- *Can I call code from my web test?*

  No. The steps of the test must be in the .webtest file. And you can't call other web tests or use loops. But there are several plug-ins that you might find helpful.
- *Is HTTPS supported?*

  We support TLS 1.1 and TLS 1.2.
- *Is there a difference between "web tests" and "availability tests"?*

  The two terms may be referenced interchangeably. Availability tests is a more generic term that includes the single URL ping tests in addition to the multi-step web tests.
- *I'd like to use availability tests on our internal server that runs behind a firewall.*

  There are two possible solutions:
  - Configure your firewall to permit incoming requests from the IP addresses of our web test agents.
  - Write your own code to periodically test your internal server. Run the code as a background process on a test server behind your firewall. Your test process can send its results to Application Insights by using TrackAvailability() API in the core SDK package. This requires your test server to have outgoing access to the Application Insights ingestion endpoint, but that is a much smaller security risk than the alternative of permitting incoming requests. The results will not appear in the availability web tests blades, but appears as availability results in Analytics, Search, and Metric Explorer.
- *Uploading a multi-step web test fails*

  There's a size limit of 300 K.

  Loops aren't supported.

  References to other web tests aren't supported.

  Data sources aren't supported.
- *My multi-step test doesn't complete*

  There's a limit of 100 requests per test.

The test is stopped if it runs longer than two minutes.

- *How can I run a test with client certificates?*

  We don't support that, sorry.

## Next steps

Search diagnostic logs

Troubleshooting

IP addresses of web test agents

# Monitor Azure web app performance

11/1/2017 • 3 min to read • Edit Online

In the Azure Portal you can set up application performance monitoring for your Azure web apps. Azure Application Insights instruments your app to send telemetry about its activities to the Application Insights service, where it is stored and analyzed. There, metric charts and search tools can be used to help diagnose issues, improve performance, and assess usage.

## Run time or build time

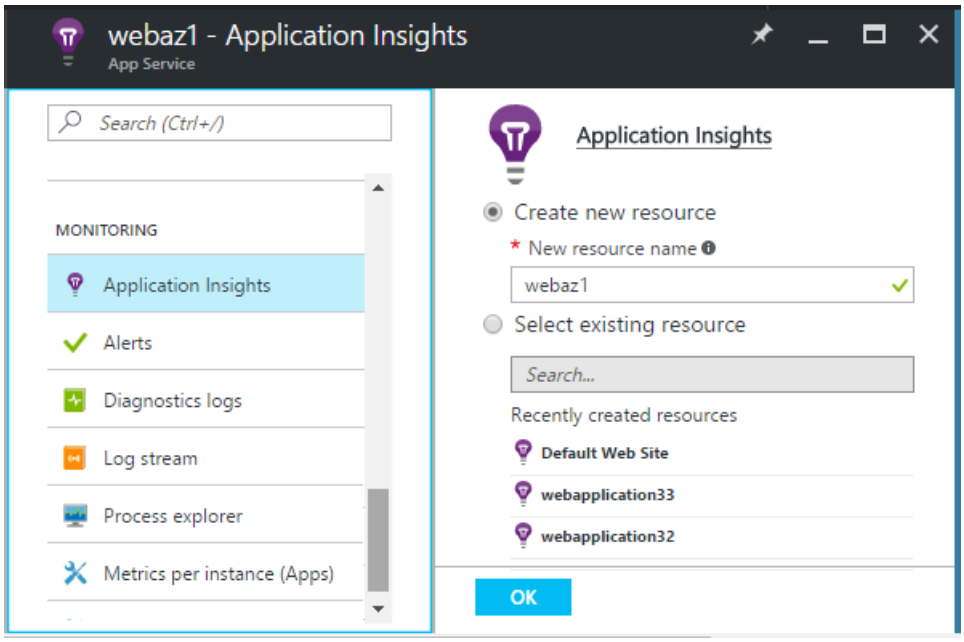
You can configure monitoring by instrumenting the app in either of two ways:

- **Run-time** - You can select a performance monitoring extension when your web app is already live. It isn't necessary to rebuild or re-install your app. You get a standard set of packages that monitor response times, success rates, exceptions, dependencies, and so on.
- **Build time** - You can install a package in your app in development. This option is more versatile. In addition to the same standard packages, you can write code to customize the telemetry or to send your own telemetry. You can log specific activities or record events according to the semantics of your app domain.

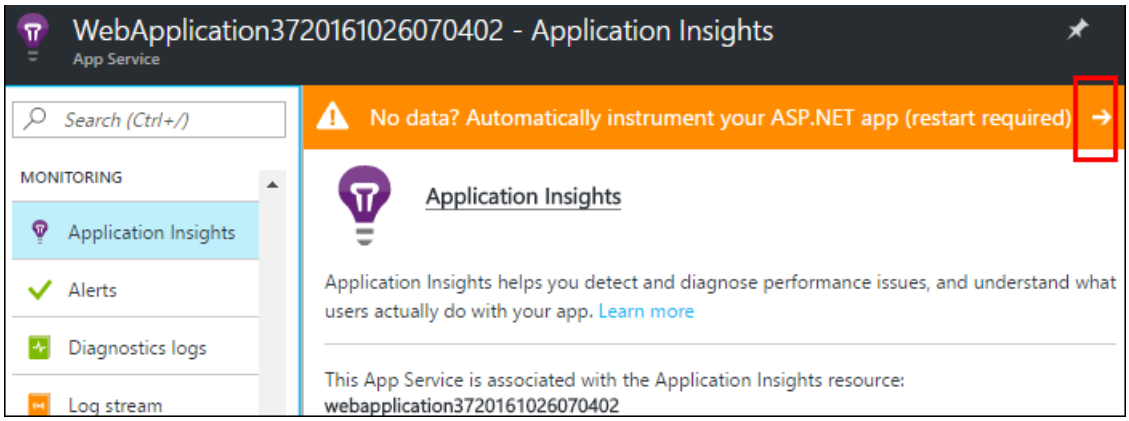
## Run time instrumentation with Application Insights

If you're already running a web app in Azure, you already get some monitoring: request and error rates. Add Application Insights to get more, such as response times, monitoring calls to dependencies, smart detection, and the powerful Log Analytics query language.

1. **Select Application Insights** in the Azure control panel for your web app.




   - Choose to create a new resource, unless you already set up an Application Insights resource for this app by another route.
2. **Instrument your web app** after Application Insights has been installed.




**Enable client side monitoring** for page view and user telemetry.

- Select Settings > Application Settings
- Under App Settings, add a new key value pair:

  Key: `APPINSIGHTS_JAVASCRIPT_ENABLED`

  Value: `true`

- **Save** the settings and **Restart** your app.

3. **Monitor your app**. Explore the data.

Later, you can build the app with Application Insights if you want.

*How do I remove Application Insights, or switch to sending to another resource?*

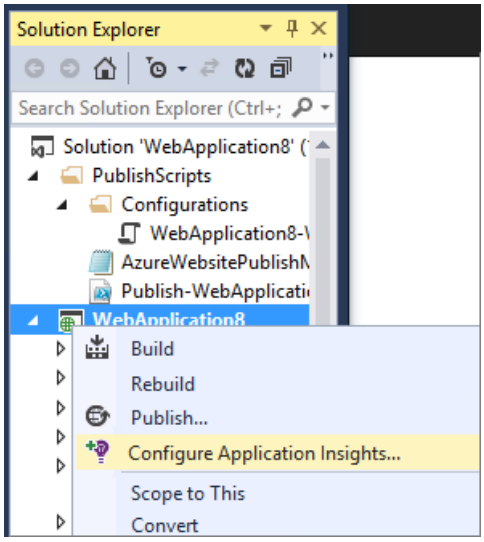
- In Azure, open the web app control blade, and under Development Tools, open **Extensions**. Delete the Application Insights extension. Then under Monitoring, choose Application Insights and create or select the resource you want.

## Build the app with Application Insights

Application Insights can provide more detailed telemetry by installing an SDK into your app. In particular, you can collect trace logs, write custom telemetry, and get more detailed exception reports.

1. **In Visual Studio** (2013 update 2 or later), configure Application Insights for your project.

   Right-click the web project, and select **Add > Application Insights** or **Configure Application Insights**.




   If you're asked to sign in, use the credentials for your Azure account.

The operation has two effects:

a. Creates an Application Insights resource in Azure, where telemetry is stored, analyzed and displayed.
b. Adds the Application Insights NuGet package to your code (if it isn't there already), and configures it to send telemetry to the Azure resource.

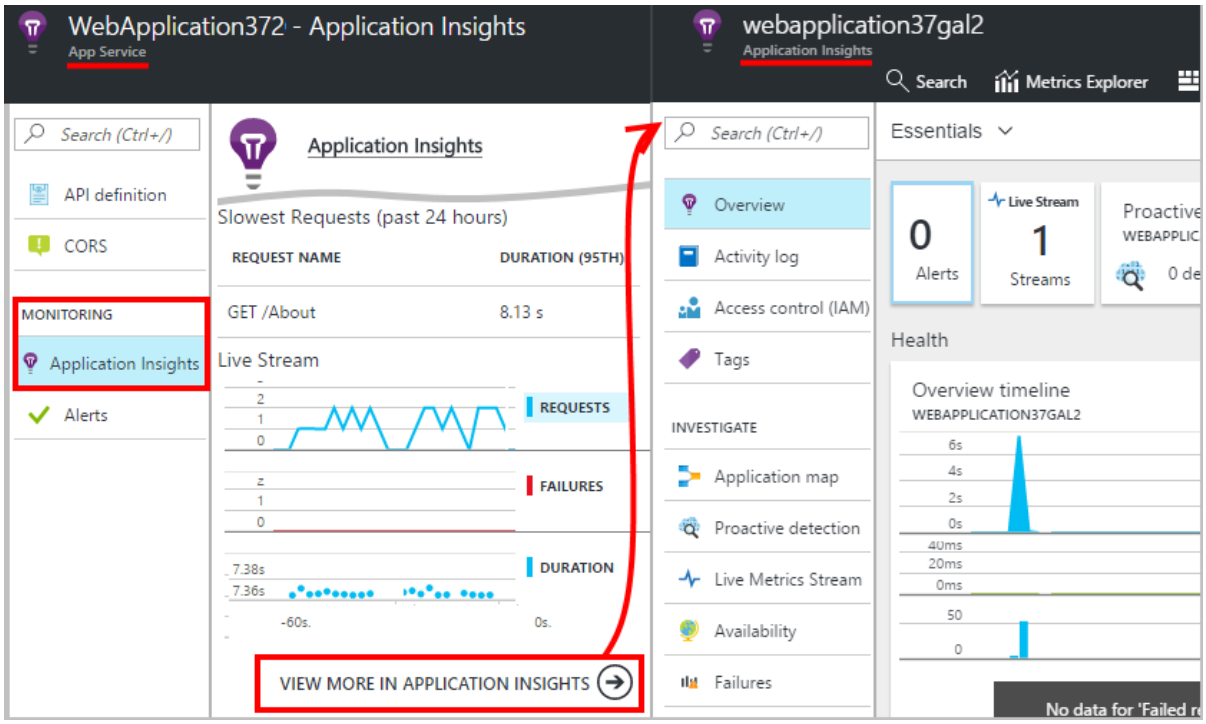
2. **Test the telemetry** by running the app in your development machine (F5).
3. **Publish the app** to Azure in the usual way.

*How do I switch to sending to a different Application Insights resource?*

- In Visual Studio, right-click the project, choose **Configure Application Insights** and choose the resource you want. You get the option to create a new resource. Rebuild and redeploy.

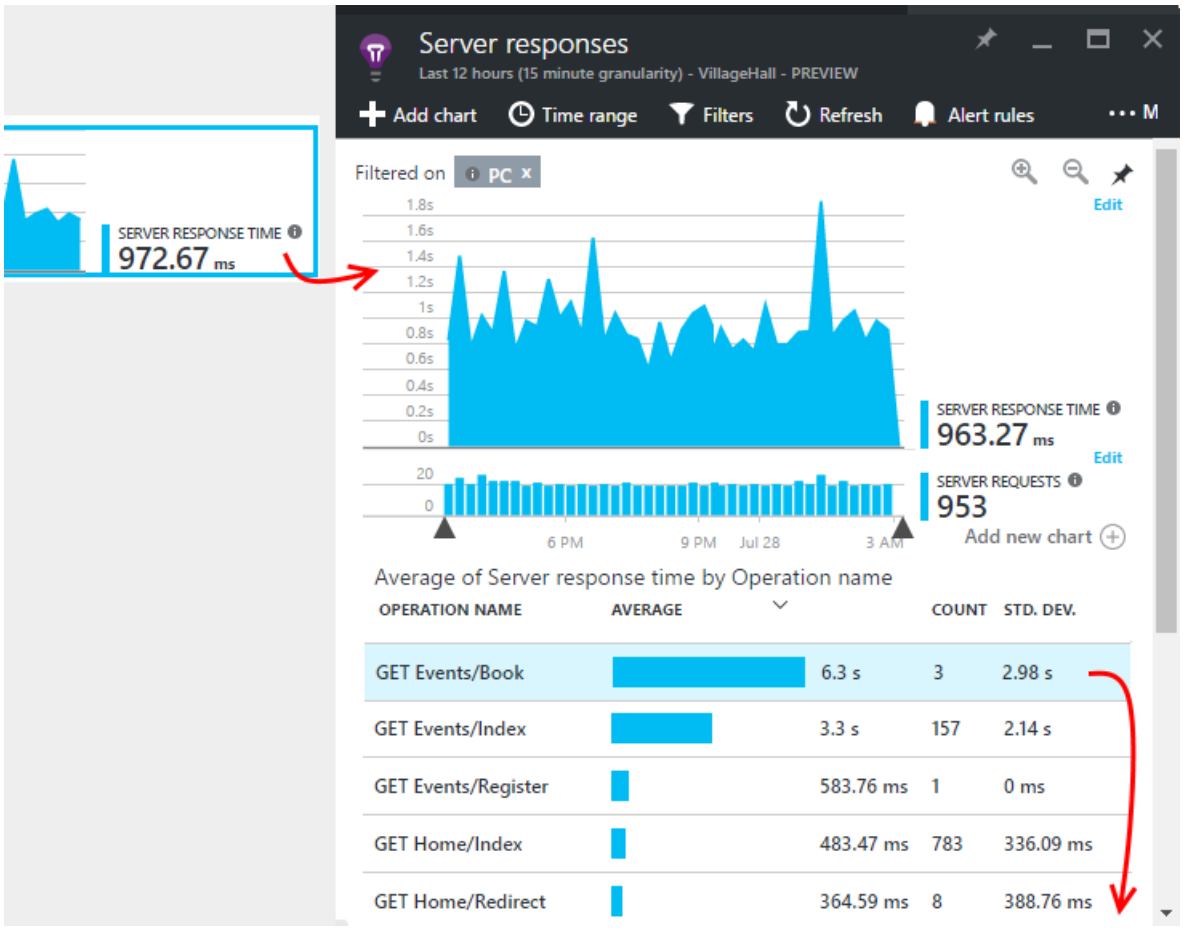
## Explore the data

1. On the Application Insights blade of your web app control panel, you see Live Metrics, which shows requests and failures within a second or two of them occurring. It's very useful display when you're republishing your app - you can see any problems immediately.
2. Click through to the full Application Insights resource.



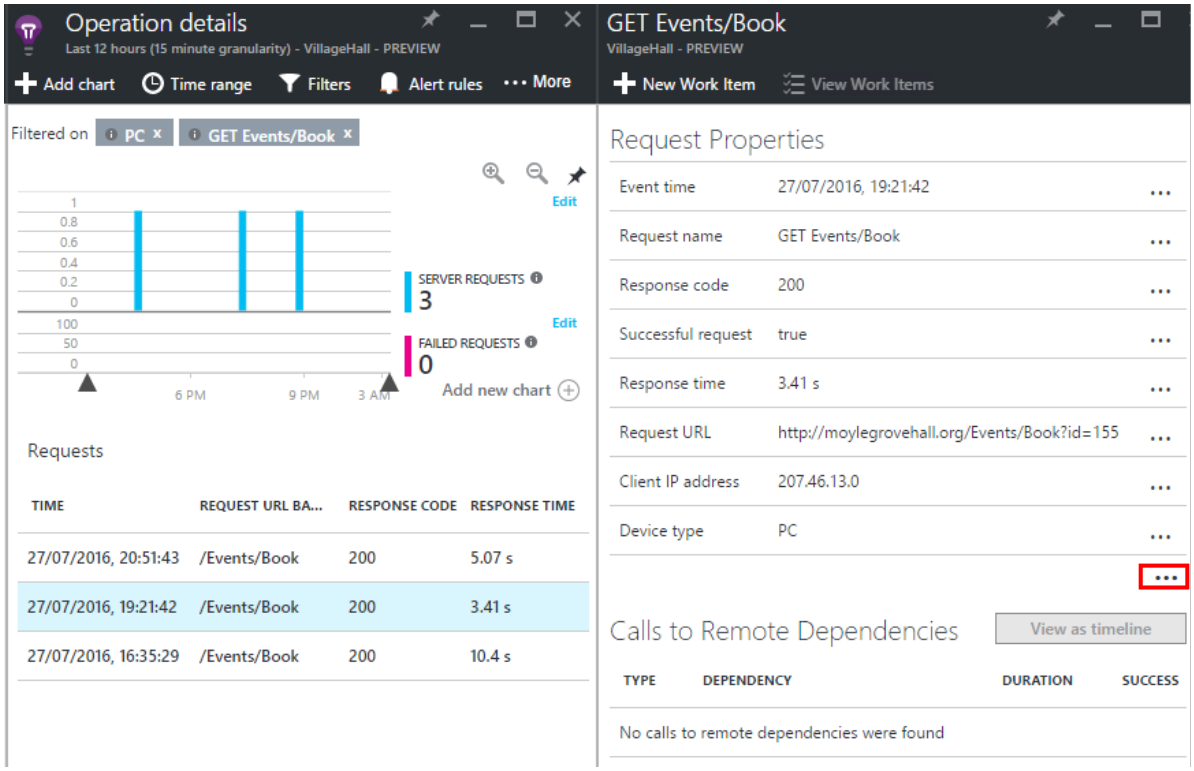

You can also go there either directly from Azure resource navigation.

3. Click through any chart to get more detail:




You can customize metrics blades.

4. Click through further to see individual events and their properties:




Notice the "..." link to open all properties.

You can customize searches.

For more powerful searches over your telemetry, use the Log Analytics query language.

## More telemetry

- Web page load data
- Custom telemetry

## Video

## Next steps

- Run the profiler on your live app.
- Azure Functions - monitor Azure Functions with Application Insights
- Enable Azure diagnostics to be sent to Application Insights.
- Monitor service health metrics to make sure your service is available and responsive.
- Receive alert notifications whenever operational events happen or metrics cross a threshold.
- Use Application Insights for JavaScript apps and web pages to get client telemetry from the browsers that visit a web page.
- Set up Availability web tests to be alerted if your site is down.

1 min to read •

1 min to read •

# Azure Resource Manager template functions

9/19/2017 • 1 min to read • Edit Online

This topic describes all the functions you can use in an Azure Resource Manager template.

You add functions in your templates by enclosing them within brackets: `[` and `]`, respectively. The expression is evaluated during deployment. While written as a string literal, the result of evaluating the expression can be of a different JSON type, such as an array, object, or integer. Just like in JavaScript, function calls are formatted as `functionName(arg1,arg2,arg3)`. You reference properties by using the dot and [index] operators.

A template expression cannot exceed 24,576 characters.

Template functions and their parameters are case-insensitive. For example, Resource Manager resolves **variables('var1')** and **VARIABLES('VAR1')** as the same. When evaluated, unless the function expressly modifies case (such as toUpper or toLower), the function preserves the case. Certain resource types may have case requirements irrespective of how functions are evaluated.

## Array and object functions

Resource Manager provides several functions for working with arrays and objects.

- array
- coalesce
- concat
- contains
- createArray
- empty
- first
- intersection
- json
- last
- length
- min
- max
- range
- skip
- take
- union

## Comparison functions

Resource Manager provides several functions for making comparisons in your templates.

- equals
- less
- lessOrEquals
- greater
- greaterOrEquals

## Deployment value functions

Resource Manager provides the following functions for getting values from sections of the template and values related to the deployment:

- deployment
- parameters
- variables

## Logical functions

Resource Manager provides the following functions for working with logical conditions:

- and
- bool
- if
- not
- or

## Numeric functions

Resource Manager provides the following functions for working with integers:

- add
- copyIndex
- div
- float
- int
- min
- max
- mod
- mul
- sub

## Resource functions

Resource Manager provides the following functions for getting resource values:

- listKeys and list{Value}
- providers
- reference
- resourceGroup
- resourceId
- subscription

## String functions

Resource Manager provides the following functions for working with strings:

- base64
- base64ToJson
- base64ToString

- concat
- contains
- dataUri
- dataUriToString
- empty
- endsWith
- first
- guid
- indexOf
- last
- lastIndexOf
- length
- padLeft
- replace
- skip
- split
- startsWith
- string
- substring
- take
- toLower
- toUpper
- trim
- uniqueString
- uri
- uriComponent
- uriComponentToString

## Next steps

- For a description of the sections in an Azure Resource Manager template, see Authoring Azure Resource Manager templates
- To merge multiple templates, see Using linked templates with Azure Resource Manager
- To iterate a specified number of times when creating a type of resource, see Create multiple instances of resources in Azure Resource Manager
- To see how to deploy the template you have created, see Deploy an application with Azure Resource Manager template

# Best practices for Autoscale

12/13/2017 • 9 min to read • Edit Online

This article teaches best practices to autoscale in Azure. Azure Monitor autoscale applies only to Virtual Machine Scale Sets, Cloud Services, and App Service - Web Apps. Other Azure services use different scaling methods.

## Autoscale concepts

- A resource can have only *one* autoscale setting
- An autoscale setting can have one or more profiles and each profile can have one or more autoscale rules.
- An autoscale setting scales instances horizontally, which is *out* by increasing the instances and *in* by decreasing the number of instances. An autoscale setting has a maximum, minimum, and default value of instances.
- An autoscale job always reads the associated metric to scale by, checking if it has crossed the configured threshold for scale-out or scale-in. You can view a list of metrics that autoscale can scale by at Azure Monitor autoscaling common metrics.
- All thresholds are calculated at an instance level. For example, "scale out by 1 instance when average CPU > 80% when instance count is 2", means scale-out when the average CPU across all instances is greater than 80%.
- All autoscale failures are logged to the Activity Log. You can then configure an activity log alert so that you can be notified via email, SMS, webhook, etc. whenever there is an autoscale failure.
- Similarly, all successful scale actions are posted to the Activity Log. You can then configure an activity log alert so that you can be notified via email, SMS, webhooks, etc. whenever there is a successful autoscale action. You can also configure email or webhook notifications to get notified for successful scale actions via the notifications tab on the autoscale setting.

## Autoscale best practices

Use the following best practices as you use autoscale.

**Ensure the maximum and minimum values are different and have an adequate margin between them**

If you have a setting that has minimum=2, maximum=2 and the current instance count is 2, no scale action can occur. Keep an adequate margin between the maximum and minimum instance counts, which are inclusive. Autoscale always scales between these limits.

**Manual scaling is reset by autoscale min and max**

If you manually update the instance count to a value above or below the maximum, the autoscale engine automatically scales back to the minimum (if below) or the maximum (if above). For example, you set the range between 3 and 6. If you have one running instance, the autoscale engine scales to 3 instances on its next run. Likewise, if you manually set the scale to 8 instances, on the next run autoscale will scale it back to 6 instances on its next run. Manual scaling is very temporary unless you reset the autoscale rules as well.

**Always use a scale-out and scale-in rule combination that performs an increase and decrease**

If you use only one part of the combination, autoscale scale-in that single out, or in, until the maximum, or minimum, is reached.

**Choose the appropriate statistic for your diagnostics metric**

For diagnostics metrics, you can choose among *Average*, *Minimum*, *Maximum* and *Total* as a metric to scale by. The most common statistic is *Average*.

**Choose the thresholds carefully for all metric types**

We recommend carefully choosing different thresholds for scale-out and scale-in based on practical situations.

We *do not recommend* autoscale settings like the examples below with the same or very similar threshold values for out and in conditions:

- Increase instances by 1 count when Thread Count <= 600
- Decrease instances by 1 count when Thread Count >= 600

Let's look at an example of what can lead to a behavior that may seem confusing. Consider the following sequence.

1. Assume there are 2 instances to begin with and then the average number of threads per instance grows to 625.
2. Autoscale scales out adding a 3rd instance.
3. Next, assume that the average thread count across instance falls to 575.
4. Before scaling down, autoscale tries to estimate what the final state will be if it scaled in. For example, 575 x 3 (current instance count) = 1,725 / 2 (final number of instances when scaled down) = 862.5 threads. This means autoscale would have to immediately scale-out again even after it scaled in, if the average thread count remains the same or even falls only a small amount. However, if it scaled up again, the whole process would repeat, leading to an infinite loop.
5. To avoid this situation (termed "flapping"), autoscale does not scale down at all. Instead, it skips and reevaluates the condition again the next time the service's job executes. This could confuse many people because autoscale wouldn't appear to work when the average thread count was 575.

Estimation during a scale-in is intended to avoid "flapping" situations, where scale-in and scale-out actions continually go back and forth. Keep this behavior in mind when you choose the same thresholds for scale-out and in.

We recommend choosing an adequate margin between the scale-out and in thresholds. As an example, consider the following better rule combination.

- Increase instances by 1 count when CPU% >= 80
- Decrease instances by 1 count when CPU% <= 60

In this case

1. Assume there are 2 instances to start with.
2. If the average CPU% across instances goes to 80, autoscale scales out adding a third instance.
3. Now assume that over time the CPU% falls to 60.
4. Autoscale's scale-in rule estimates the final state if it were to scale-in. For example, 60 x 3 (current instance count) = 180 / 2 (final number of instances when scaled down) = 90. So autoscale does not scale-in because it would have to scale-out again immediately. Instead, it skips scaling down.
5. The next time autoscale checks, the CPU continues to fall to 50. It estimates again - 50 x 3 instance = 150 / 2 instances = 75, which is below the scale-out threshold of 80, so it scales in successfully to 2 instances.

**Considerations for scaling threshold values for special metrics**

For special metrics such as Storage or Service Bus Queue length metric, the threshold is the average number of messages available per current number of instances. Carefully choose the threshold value for this metric.

Let's illustrate it with an example to ensure you understand the behavior better.

- Increase instances by 1 count when Storage Queue message count >= 50
- Decrease instances by 1 count when Storage Queue message count <= 10

Consider the following sequence:

1. There are 2 storage queue instances.
2. Messages keep coming and when you review the storage queue, the total count reads 50. You might assume

that autoscale should start a scale-out action. However, note that it is still 50/2 = 25 messages per instance. So, scale-out does not occur. For the first scale-out to happen, the total message count in the storage queue should be 100.

3. Next, assume that the total message count reaches 100.
4. A 3rd storage queue instance is added due to a scale-out action. The next scale-out action will not happen until the total message count in the queue reaches 150 because 150/3 = 50.
5. Now the number of messages in the queue gets smaller. With 3 instances, the first scale-in action happens when the total messages in all queues add up to 30 because 30/3 = 10 messages per instance, which is the scale-in threshold.

**Considerations for scaling when multiple profiles are configured in an autoscale setting**

In an autoscale setting, you can choose a default profile, which is always applied without any dependency on schedule or time, or you can choose a recurring profile or a profile for a fixed period with a date and time range.

When autoscale service processes them, it always checks in the following order:

1. Fixed Date profile
2. Recurring profile
3. Default ("Always") profile

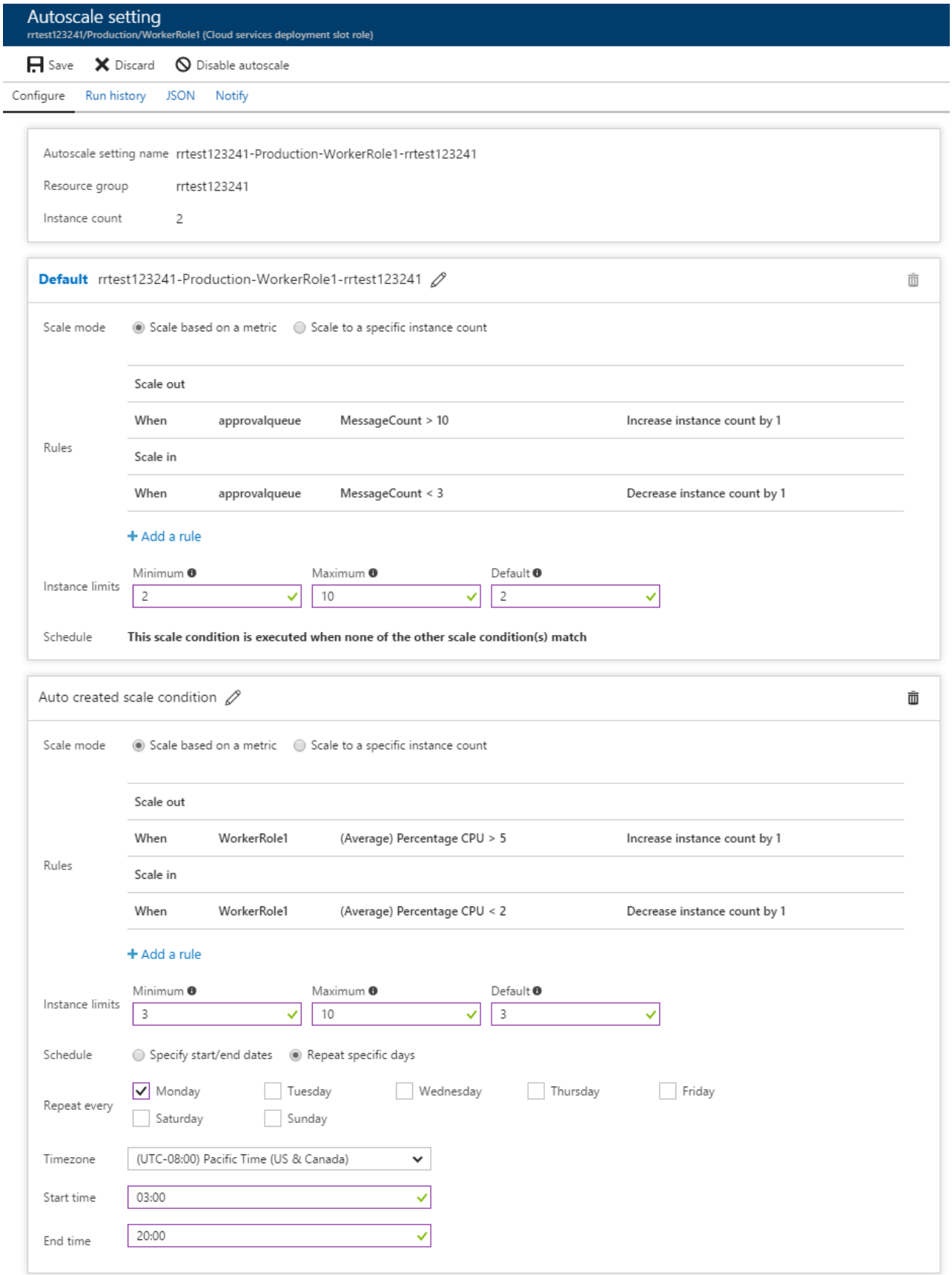
If a profile condition is met, autoscale does not check the next profile condition below it. Autoscale only processes one profile at a time. This means if you want to also include a processing condition from a lower-tier profile, you must include those rules as well in the current profile.

Let's review this using an example:

The image below shows an autoscale setting with a default profile of minimum instances = 2 and maximum instances = 10. In this example, rules are configured to scale-out when the message count in the queue is greater than 10 and scale-in when the message count in the queue is less than 3. So now the resource can scale between 2 and 10 instances.

In addition, there is a recurring profile set for Monday. It is set for minimum instances = 3 and maximum instances = 10. This means on Monday, the first time autoscale checks for this condition, if the instance count is 2, it scales to the new minimum of 3. As long as autoscale continues to find this profile condition matched (Monday), it only processes the CPU-based scale-out/in rules configured for this profile. At this time, it does not check for the queue length. However, if you also want the queue length condition to be checked, you should include those rules from the default profile as well in your Monday profile.

Similarly, when autoscale switches back to the default profile, it first checks if the minimum and maximum conditions are met. If the number of instances at the time is 12, it scales in to 10, the maximum allowed for the default profile.




**Considerations for scaling when multiple rules are configured in a profile**

There are cases where you may have to set multiple rules in a profile. The following set of autoscale rules are used by services use when multiple rules are set.

On *scale out*, autoscale runs if any rule is met. On *scale-in*, autoscale require all rules to be met.

To illustrate, assume that you have the following 4 autoscale rules:

- If CPU < 30 %, scale-in by 1

- If Memory < 50%, scale-in by 1
- If CPU > 75%, scale-out by 1
- If Memory > 75%, scale-out by 1

Then the follow occurs:

- If CPU is 76% and Memory is 50%, we scale-out.
- If CPU is 50% and Memory is 76% we scale-out.

On the other hand, if CPU is 25% and memory is 51% autoscale does **not** scale-in. In order to scale-in, CPU must be 29% and Memory 49%.

**Always select a safe default instance count**

The default instance count is important autoscale scales your service to that count when metrics are not available. Therefore, select a default instance count that's safe for your workloads.

**Configure autoscale notifications**

Autoscale will post to the Activity Log if any of the following conditions occur:

- Autoscale issues a scale operation
- Autoscale service successfully completes a scale action
- Autoscale service fails to take a scale action.
- Metrics are not available for autoscale service to make a scale decision.
- Metrics are available (recovery) again to make a scale decision.

You can also use an Activity Log alert to monitor the health of the autoscale engine. Here are examples to create an Activity Log Alert to monitor all autoscale engine operations on your subscription or to create an Activity Log Alert to monitor all failed autoscale scale in/scale out operations on your subscription.

In addition to using activity log alerts, you can also configure email or webhook notifications to get notified for successful scale actions via the notifications tab on the autoscale setting.

## Next Steps

- Create an Activity Log Alert to monitor all autoscale engine operations on your subscription.
- Create an Activity Log Alert to monitor all failed autoscale scale in/scale out operations on your subscription

# Azure Monitor autoscaling common metrics

7/18/2017 • 5 min to read • Edit Online

Azure Monitor autoscaling allows you to scale the number of running instances up or down, based on telemetry data (metrics). This document describes common metrics that you might want to use. In the Azure portal for Cloud Services and Server Farms, you can choose the metric of the resource to scale by. However, you can also choose any metric from a different resource to scale by.

Azure Monitor autoscale applies only to Virtual Machine Scale Sets, Cloud Services, and App Service - Web Apps. Other Azure services use different scaling methods.

## Compute metrics for Resource Manager-based VMs

By default, Resource Manager-based Virtual Machines and Virtual Machine Scale Sets emit basic (host-level) metrics. In addition, when you configure diagnostics data collection for an Azure VM and VMSS, the Azure diagnostic extension also emits guest-OS performance counters (commonly known as "guest-OS metrics"). You use all these metrics in autoscale rules.

You can use the `Get MetricDefinitions` API/PoSH/CLI to view the metrics available for your VMSS resource.

If you're using VM scale sets and you don't see a particular metric listed, then it is likely *disabled* in your diagnostics extension.

If a particular metric is not being sampled or transferred at the frequency you want, you can update the diagnostics configuration.

If either preceding case is true, then review Use PowerShell to enable Azure Diagnostics in a virtual machine running Windows about PowerShell to configure and update your Azure VM Diagnostics extension to enable the metric. That article also includes a sample diagnostics configuration file.

#### Host metrics for Resource Manager-based Windows and Linux VMs

The following host-level metrics are emitted by default for Azure VM and VMSS in both Windows and Linux instances. These metrics describe your Azure VM, but are collected from the Azure VM host rather than via agent installed on the guest VM. You may use these metrics in autoscaling rules.

- Host metrics for Resource Manager-based Windows and Linux VMs
- Host metrics for Resource Manager-based Windows and Linux VM Scale Sets

#### Guest OS metrics Resource Manager-based Windows VMs

When you create a VM in Azure, diagnostics is enabled by using the Diagnostics extension. The diagnostics extension emits a set of metrics taken from inside of the VM. This means you can autoscale off of metrics that are not emitted by default.

You can generate a list of the metrics by using the following command in PowerShell.

```
Get-AzureRmMetricDefinition -ResourceId <resource_id> | Format-Table -Property Name,Unit
```

You can create an alert for the following metrics:

| METRIC NAME | UNIT |
| --- | --- |
| \Processor(_Total)% Processor Time | Percent |
| \Processor(_Total)% Privileged Time | Percent |
| \Processor(_Total)% User Time | Percent |
| \Processor Information(_Total)\Processor Frequency | Count |
| \System\Processes | Count |
| \Process(_Total)\Thread Count | Count |
| \Process(_Total)\Handle Count | Count |
| \Memory% Committed Bytes In Use | Percent |
| \Memory\Available Bytes | Bytes |
| \Memory\Committed Bytes | Bytes |
| \Memory\Commit Limit | Bytes |
| \Memory\Pool Paged Bytes | Bytes |
| \Memory\Pool Nonpaged Bytes | Bytes |
| \PhysicalDisk(_Total)% Disk Time | Percent |
| \PhysicalDisk(_Total)% Disk Read Time | Percent |
| \PhysicalDisk(_Total)% Disk Write Time | Percent |
| \PhysicalDisk(_Total)\Disk Transfers/sec | CountPerSecond |
| \PhysicalDisk(_Total)\Disk Reads/sec | CountPerSecond |
| \PhysicalDisk(_Total)\Disk Writes/sec | CountPerSecond |
| \PhysicalDisk(_Total)\Disk Bytes/sec | BytesPerSecond |
| \PhysicalDisk(_Total)\Disk Read Bytes/sec | BytesPerSecond |
| \PhysicalDisk(_Total)\Disk Write Bytes/sec | BytesPerSecond |
| \PhysicalDisk(_Total)\Avg. Disk Queue Length | Count |
| \PhysicalDisk(_Total)\Avg. Disk Read Queue Length | Count |
| \PhysicalDisk(_Total)\Avg. Disk Write Queue Length | Count |

| METRIC NAME | UNIT |
| --- | --- |
| \LogicalDisk(_Total)% Free Space | Percent |
| \LogicalDisk(_Total)\Free Megabytes | Count |

### Guest OS metrics Linux VMs

When you create a VM in Azure, diagnostics is enabled by default by using Diagnostics extension.

You can generate a list of the metrics by using the following command in PowerShell.

```
Get-AzureRmMetricDefinition -ResourceId <resource_id> | Format-Table -Property Name,Unit
```

You can create an alert for the following metrics:

| METRIC NAME | UNIT |
| --- | --- |
| \Memory\AvailableMemory | Bytes |
| \Memory\PercentAvailableMemory | Percent |
| \Memory\UsedMemory | Bytes |
| \Memory\PercentUsedMemory | Percent |
| \Memory\PercentUsedByCache | Percent |
| \Memory\PagesPerSec | CountPerSecond |
| \Memory\PagesReadPerSec | CountPerSecond |
| \Memory\PagesWrittenPerSec | CountPerSecond |
| \Memory\AvailableSwap | Bytes |
| \Memory\PercentAvailableSwap | Percent |
| \Memory\UsedSwap | Bytes |
| \Memory\PercentUsedSwap | Percent |
| \Processor\PercentIdleTime | Percent |
| \Processor\PercentUserTime | Percent |
| \Processor\PercentNiceTime | Percent |
| \Processor\PercentPrivilegedTime | Percent |
| \Processor\PercentInterruptTime | Percent |
| \Processor\PercentDPCTime | Percent |

| METRIC NAME | UNIT |
| --- | --- |
| \Processor\PercentProcessorTime | Percent |
| \Processor\PercentIOWaitTime | Percent |
| \PhysicalDisk\BytesPerSecond | BytesPerSecond |
| \PhysicalDisk\ReadBytesPerSecond | BytesPerSecond |
| \PhysicalDisk\WriteBytesPerSecond | BytesPerSecond |
| \PhysicalDisk\TransfersPerSecond | CountPerSecond |
| \PhysicalDisk\ReadsPerSecond | CountPerSecond |
| \PhysicalDisk\WritesPerSecond | CountPerSecond |
| \PhysicalDisk\AverageReadTime | Seconds |
| \PhysicalDisk\AverageWriteTime | Seconds |
| \PhysicalDisk\AverageTransferTime | Seconds |
| \PhysicalDisk\AverageDiskQueueLength | Count |
| \NetworkInterface\BytesTransmitted | Bytes |
| \NetworkInterface\BytesReceived | Bytes |
| \NetworkInterface\PacketsTransmitted | Count |
| \NetworkInterface\PacketsReceived | Count |
| \NetworkInterface\BytesTotal | Bytes |
| \NetworkInterface\TotalRxErrors | Count |
| \NetworkInterface\TotalTxErrors | Count |
| \NetworkInterface\TotalCollisions | Count |

## Commonly used Web (Server Farm) metrics

You can also perform autoscale based on common web server metrics such as the Http queue length. It's metric name is **HttpQueueLength**. The following section lists available server farm (Web Apps) metrics.

### Web Apps metrics

You can generate a list of the Web Apps metrics by using the following command in PowerShell.

```
Get-AzureRmMetricDefinition -ResourceId <resource_id> | Format-Table -Property Name,Unit
```

You can alert on or scale by these metrics.

| METRIC NAME | UNIT |
|---|---|
| CpuPercentage | Percent |
| MemoryPercentage | Percent |
| DiskQueueLength | Count |
| HttpQueueLength | Count |
| BytesReceived | Bytes |
| BytesSent | Bytes |

## Commonly used Storage metrics

You can scale by Storage queue length, which is the number of messages in the storage queue. Storage queue length is a special metric and the threshold is the number of messages per instance. For example, if there are two instances and if the threshold is set to 100, scaling occurs when the total number of messages in the queue is 200. That can be 100 messages per instance, 120 and 80, or any other combination that adds up to 200 or more.

Configure this setting in the Azure portal in the **Settings** blade. For VM scale sets, you can update the Autoscale setting in the Resource Manager template to use *metricName* as *ApproximateMessageCount* and pass the ID of the storage queue as *metricResourceUri*.

For example, with a Classic Storage Account the autoscale setting metricTrigger would include:

```
"metricName": "ApproximateMessageCount",
 "metricNamespace": "",
 "metricResourceUri":
"/subscriptions/SUBSCRIPTION_ID/resourceGroups/RES_GROUP_NAME/providers/Microsoft.ClassicStorage/storageAccoun
ts/STORAGE_ACCOUNT_NAME/services/queue/queues/QUEUE_NAME"
```

For a (non-classic) storage account, the metricTrigger would include:

```
"metricName": "ApproximateMessageCount",
"metricNamespace": "",
"metricResourceUri":
"/subscriptions/SUBSCRIPTION_ID/resourceGroups/RES_GROUP_NAME/providers/Microsoft.Storage/storageAccounts/STOR
AGE_ACCOUNT_NAME/services/queue/queues/QUEUE_NAME"
```

## Commonly used Service Bus metrics

You can scale by Service Bus queue length, which is the number of messages in the Service Bus queue. Service Bus queue length is a special metric and the threshold is the number of messages per instance. For example, if there are two instances and if the threshold is set to 100, scaling occurs when the total number of messages in the queue is 200. That can be 100 messages per instance, 120 and 80, or any other combination that adds up to 200 or more.

For VM scale sets, you can update the Autoscale setting in the Resource Manager template to use *metricName* as *ApproximateMessageCount* and pass the ID of the storage queue as *metricResourceUri*.

```
"metricName": "MessageCount",
 "metricNamespace": "",
"metricResourceUri":
"/subscriptions/SUBSCRIPTION_ID/resourceGroups/RES_GROUP_NAME/providers/Microsoft.ServiceBus/namespaces/SB_NAM
ESPACE/queues/QUEUE_NAME"
```

**NOTE**

For Service Bus, the resource group concept does not exist but Azure Resource Manager creates a default resource group per region. The resource group is usually in the 'Default-ServiceBus-[region]' format. For example, 'Default-ServiceBus-EastUS', 'Default-ServiceBus-WestUS', 'Default-ServiceBus-AustraliaEast' etc.

# Use autoscale actions to send email and webhook alert notifications in Azure Monitor

7/18/2017 • 3 min to read • Edit Online

This article shows you how set up triggers so that you can call specific web URLs or send emails based on autoscale actions in Azure.

## Webhooks

Webhooks allow you to route the Azure alert notifications to other systems for post-processing or custom notifications. For example, routing the alert to services that can handle an incoming web request to send SMS, log bugs, notify a team using chat or messaging services, etc. The webhook URI must be a valid HTTP or HTTPS endpoint.

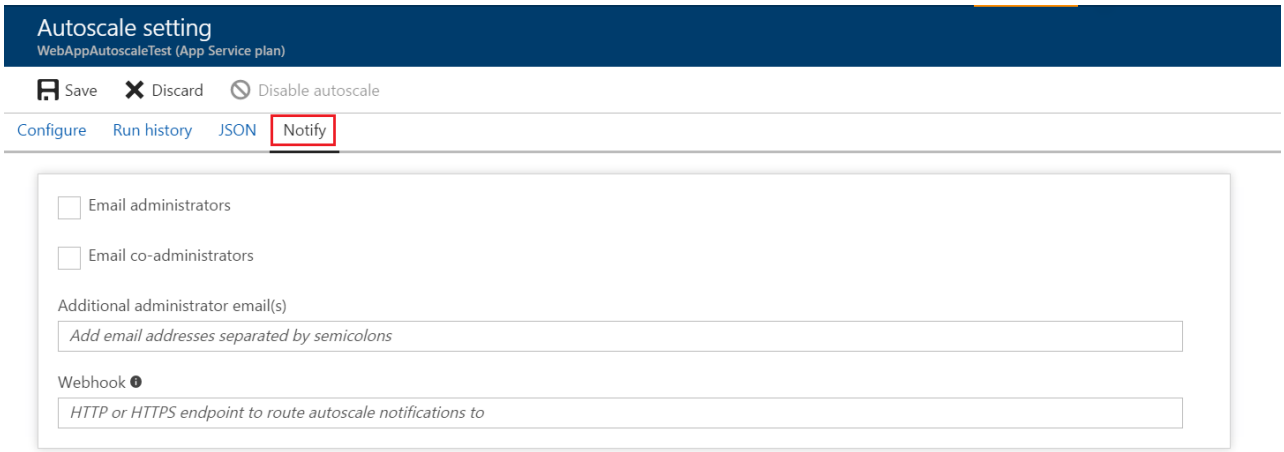
## Email

Email can be sent to any valid email address. Administrators and co-administrators of the subscription where the rule is running will also be notified.

## Cloud Services and Web Apps

You can opt-in from the Azure portal for Cloud Services and Server Farms (Web Apps).

- Choose the **scale by** metric.




## Virtual Machine scale sets

For newer Virtual Machines created with Resource Manager (Virtual Machine scale sets), you can configure this using REST API, Resource Manager templates, PowerShell, and CLI. A portal interface is not yet available. When using the REST API or Resource Manager template, include the notifications element with the following options.

```
"notifications": [
      {
        "operation": "Scale",
        "email": {
          "sendToSubscriptionAdministrator": false,
          "sendToSubscriptionCoAdministrators": false,
          "customEmails": [
              "user1@mycompany.com",
              "user2@mycompany.com"
              ]
        },
        "webhooks": [
          {
            "serviceUri": "https://foo.webhook.example.com?token=abcd1234",
            "properties": {
              "optional_key1": "optional_value1",
              "optional_key2": "optional_value2"
            }
          }
        ]
      }
    ]
```

| FIELD | MANDATORY? | DESCRIPTION |
|---|---|---|
| operation | yes | value must be "Scale" |
| sendToSubscriptionAdministrator | yes | value must be "true" or "false" |
| sendToSubscriptionCoAdministrators | yes | value must be "true" or "false" |
| customEmails | yes | value can be null [] or string array of emails |
| webhooks | yes | value can be null or valid Uri |
| serviceUri | yes | a valid https Uri |
| properties | yes | value must be empty {} or can contain key-value pairs |

## Authentication in webhooks

The webhook can authenticate using token-based authentication, where you save the webhook URI with a token ID as a query parameter. For example, https://mysamplealert/webcallback?tokenid=sometokenid&someparameter=somevalue

## Autoscale notification webhook payload schema

When the autoscale notification is generated, the following metadata is included in the webhook payload:

```
{
        "version": "1.0",
        "status": "Activated",
        "operation": "Scale In",
        "context": {
                "timestamp": "2016-03-11T07:31:04.5834118Z",
                "id":
"/subscriptions/s1/resourceGroups/rg1/providers/microsoft.insights/autoscalesettings/myautoscaleSetting",
                "name": "myautoscaleSetting",
                "details": "Autoscale successfully started scale operation for resource 'MyCSRole' from
capacity '3' to capacity '2'",
                "subscriptionId": "s1",
                "resourceGroupName": "rg1",
                "resourceName": "MyCSRole",
                "resourceType": "microsoft.classiccompute/domainnames/slots/roles",
                "resourceId":
"/subscriptions/s1/resourceGroups/rg1/providers/microsoft.classicCompute/domainNames/myCloudService/slots/Produ
ction/roles/MyCSRole",
                "portalLink":
"https://portal.azure.com/#resource/subscriptions/s1/resourceGroups/rg1/providers/microsoft.classicCompute/doma
inNames/myCloudService",
                "oldCapacity": "3",
                "newCapacity": "2"
        },
        "properties": {
                "key1": "value1",
                "key2": "value2"
        }
}
```

| FIELD | MANDATORY? | DESCRIPTION |
|---|---|---|
| status | yes | The status that indicates that an autoscale action was generated |
| operation | yes | For an increase of instances, it will be "Scale Out" and for a decrease in instances, it will be "Scale In" |
| context | yes | The autoscale action context |
| timestamp | yes | Time stamp when the autoscale action was triggered |
| id | Yes | Resource Manager ID of the autoscale setting |
| name | Yes | The name of the autoscale setting |
| details | Yes | Explanation of the action that the autoscale service took and the change in the instance count |
| subscriptionId | Yes | Subscription ID of the target resource that is being scaled |
| resourceGroupName | Yes | Resource Group name of the target resource that is being scaled |

| FIELD | MANDATORY? | DESCRIPTION |
| --- | --- | --- |
| resourceName | Yes | Name of the target resource that is being scaled |
| resourceType | Yes | The three supported values: "microsoft.classiccompute/domainnames/slots/roles" - Cloud Service roles, "microsoft.compute/virtualmachinescalesets" - Virtual Machine Scale Sets, and "Microsoft.Web/serverfarms" - Web App |
| resourceId | Yes | Resource Manager ID of the target resource that is being scaled |
| portalLink | Yes | Azure portal link to the summary page of the target resource |
| oldCapacity | Yes | The current (old) instance count when Autoscale took a scale action |
| newCapacity | Yes | The new instance count that Autoscale scaled the resource to |
| Properties | No | Optional. Set of <Key, Value> pairs (for example, Dictionary <String, String>). The properties field is optional. In a custom user interface or Logic app based workflow, you can enter keys and values that can be passed using the payload. An alternate way to pass custom properties back to the outgoing webhook call is to use the webhook URI itself (as query parameters) |

# Call a webhook on Azure Activity Log alerts

8/10/2017 • 3 min to read • Edit Online

Webhooks allow you to route an Azure alert notification to other systems for post-processing or custom actions. You can use a webhook on an alert to route it to services that send SMS, log bugs, notify a team via chat/messaging services, or do any number of other actions. This article describes how to set a webhook to be called when an Azure Activity Log alert fires. It also shows what the payload for the HTTP POST to a webhook looks like. For information on the setup and schema for an Azure metric alert, see this page instead. You can also set up an Activity Log alert to send email when activated.

> **NOTE**
> This feature is currently in preview and will be removed at some point in the future.

You can set up an Activity Log alert using the Azure PowerShell Cmdlets, Cross-Platform CLI, or Azure Monitor REST API. Currently, you cannot set one up using the Azure portal.

## Authenticating the webhook

The webhook can authenticate using either of these methods:

1. **Token-based authorization** - The webhook URI is saved with a token ID, for example,
   `https://mysamplealert/webcallback?tokenid=sometokenid&someparameter=somevalue`
2. **Basic authorization** - The webhook URI is saved with a username and password, for example,
   `https://userid:password@mysamplealert/webcallback?someparamater=somevalue&foo=bar`

## Payload schema

The POST operation contains the following JSON payload and schema for all Activity Log-based alerts. This schema is similar to the one used by metric-based alerts.

```
{
        "status": "Activated",
        "context": {
                "resourceProviderName": "Microsoft.Web",
                "event": {
                        "$type":
"Microsoft.WindowsAzure.Management.Monitoring.Automation.Notifications.GenericNotifications.Datacontracts.Insta
nceEventContext, Microsoft.WindowsAzure.Management.Mon.Automation",
                        "authorization": {
                                "action": "Microsoft.Web/sites/start/action",
                                "scope":
"/subscriptions/s1/resourcegroups/rg1/providers/Microsoft.Web/sites/leoalerttest"
                        },
                        "eventDataId": "327caaca-08d7-41b1-86d8-27d0a7adb92d",
                        "category": "Administrative",
                        "caller": "myname@mycompany.com",
                        "httpRequest": {
                                "clientRequestId": "f58cead8-c9ed-43af-8710-55e64def208d",
                                "clientIpAddress": "104.43.166.155",
                                "method": "POST"
                        },
                        "status": "Succeeded",
                        "subStatus": "OK",
                        "level": "Informational",
                        "correlationId": "4a40beaa-6a63-4d92-85c4-923a25abb590",
                        "eventDescription": "",
                        "operationName": "Microsoft.Web/sites/start/action",
                        "operationId": "4a40beaa-6a63-4d92-85c4-923a25abb590",
                        "properties": {
                                "$type":
"Microsoft.WindowsAzure.Management.Common.Storage.CasePreservedDictionary,
Microsoft.WindowsAzure.Management.Common.Storage",
                                "statusCode": "OK",
                                "serviceRequestId": "f7716681-496a-4f5c-8d14-d564bcf54714"
                        }
                },
                "timestamp": "Friday, March 11, 2016 9:13:23 PM",
                "id":
"/subscriptions/s1/resourceGroups/rg1/providers/microsoft.insights/alertrules/alertonevent2",
                "name": "alertonevent2",
                "description": "test alert on event start",
                "conditionType": "Event",
                "subscriptionId": "s1",
                "resourceId":
"/subscriptions/s1/resourcegroups/rg1/providers/Microsoft.Web/sites/leoalerttest",
                "resourceGroupName": "rg1"
        },
        "properties": {
                "key1": "value1",
                "key2": "value2"
        }
}
```


| ELEMENT NAME | DESCRIPTION |
| --- | --- |
| status | Used for metric alerts. Always set to "activated" for Activity Log alerts. |
| context | Context of the event. |
| resourceProviderName | The resource provider of the impacted resource. |
| conditionType | Always "Event." |

| ELEMENT NAME | DESCRIPTION |
| --- | --- |
| name | Name of the alert rule. |
| id | Resource ID of the alert. |
| description | Alert description as set during creation of the alert. |
| subscriptionId | Azure Subscription ID. |
| timestamp | Time at which the event was generated by the Azure service that processed the request. |
| resourceId | Resource ID of the impacted resource. |
| resourceGroupName | Name of the resource group for the impacted resource |
| properties | Set of `<Key, Value>` pairs (i.e. `Dictionary<String, String>`) that includes details about the event. |
| event | Element containing metadata about the event. |
| authorization | The RBAC properties of the event. These usually include the "action", "role" and the "scope." |
| category | Category of the event. Supported values include: Administrative, Alert, Security, ServiceHealth, Recommendation. |
| caller | Email address of the user who performed the operation, UPN claim, or SPN claim based on availability. Can be null for certain system calls. |
| correlationId | Usually a GUID in string format. Events with correlationId belong to the same larger action and usually share a correlationId. |
| eventDescription | Static text description of the event. |
| eventDataId | Unique identifier for the event. |
| eventSource | Name of the Azure service or infrastructure that generated the event. |
| httpRequest | Usually includes the "clientRequestId", "clientIpAddress" and "method" (HTTP method e.g. PUT). |
| level | One of the following values: "Critical", "Error", "Warning", "Informational" and "Verbose." |
| operationId | Usually a GUID shared among the events corresponding to single operation. |
| operationName | Name of the operation. |

| ELEMENT NAME | DESCRIPTION |
| --- | --- |
| properties | Properties of the event. |
| status | String. Status of the operation. Common values include: "Started", "In Progress", "Succeeded", "Failed", "Active", "Resolved". |
| subStatus | Usually includes the HTTP status code of the corresponding REST call. It might also include other strings describing a substatus. Common substatus values include: OK (HTTP Status Code: 200), Created (HTTP Status Code: 201), Accepted (HTTP Status Code: 202), No Content (HTTP Status Code: 204), Bad Request (HTTP Status Code: 400), Not Found (HTTP Status Code: 404), Conflict (HTTP Status Code: 409), Internal Server Error (HTTP Status Code: 500), Service Unavailable (HTTP Status Code: 503), Gateway Timeout (HTTP Status Code: 504) |

## Next steps

- Learn more about the Activity Log
- Execute Azure Automation scripts (Runbooks) on Azure alerts
- Use Logic App to send an SMS via Twilio from an Azure alert. This example is for metric alerts, but could be modified to work with an Activity Log alert.
- Use Logic App to send a Slack message from an Azure alert. This example is for metric alerts, but could be modified to work with an Activity Log alert.
- Use Logic App to send a message to an Azure Queue from an Azure alert. This example is for metric alerts, but could be modified to work with an Activity Log alert.

# Configure a webhook on an Azure metric alert

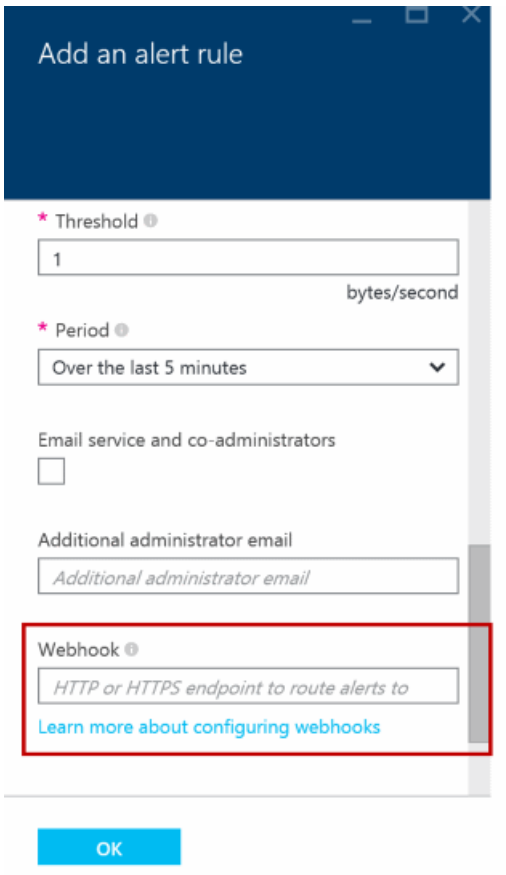
8/10/2017 • 3 min to read • Edit Online

Webhooks allow you to route an Azure alert notification to other systems for post-processing or custom actions. You can use a webhook on an alert to route it to services that send SMS, log bugs, notify a team via chat/messaging services, or do any number of other actions. This article describes how to set a webhook on an Azure metric alert and what the payload for the HTTP POST to a webhook looks like. For information on the setup and schema for an Azure Activity Log alert (alert on events), see this page instead.

Azure alerts HTTP POST the alert contents in JSON format, schema defined below, to a webhook URI that you provide when creating the alert. This URI must be a valid HTTP or HTTPS endpoint. Azure posts one entry per request when an alert is activated.

## Configuring webhooks via the portal

You can add or update the webhook URI in the Create/Update Alerts screen in the portal.




You can also configure an alert to post to a webhook URI using the Azure PowerShell Cmdlets, Cross-Platform CLI, or Azure Monitor REST API.

## Authenticating the webhook

The webhook can authenticate using token-based authorization. The webhook URI is saved with a token ID, eg.

```
https://mysamplealert/webcallback?tokenid=sometokenid&someparameter=somevalue
```

## Payload schema

The POST operation contains the following JSON payload and schema for all metric-based alerts.

```
{
"status": "Activated",
"context": {
            "timestamp": "2015-08-14T22:26:41.9975398Z",
            "id": "/subscriptions/s1/resourceGroups/useast/providers/microsoft.insights/alertrules/ruleName1",
            "name": "ruleName1",
            "description": "some description",
            "conditionType": "Metric",
            "condition": {
                        "metricName": "Requests",
                        "metricUnit": "Count",
                        "metricValue": "10",
                        "threshold": "10",
                        "windowSize": "15",
                        "timeAggregation": "Average",
                        "operator": "GreaterThanOrEqual"
                },
            "subscriptionId": "s1",
            "resourceGroupName": "useast",
            "resourceName": "mysite1",
            "resourceType": "microsoft.foo/sites",
            "resourceId": "/subscriptions/s1/resourceGroups/useast/providers/microsoft.foo/sites/mysite1",
            "resourceRegion": "centralus",
            "portalLink":
"https://portal.azure.com/#resource/subscriptions/s1/resourceGroups/useast/providers/microsoft.foo/sites/mysit
e1"
},
"properties": {
              "key1": "value1",
              "key2": "value2"
              }
}
```

| FIELD | MANDATORY | FIXED SET OF VALUES | NOTES |
|---|---|---|---|
| status | Y | “Activated”, “Resolved” | Status for the alert based off of the conditions you have set. |
| context | Y | | The alert context. |
| timestamp | Y | | The time at which the alert was triggered. |
| id | Y | | Every alert rule has a unique id. |
| name | Y | | The alert name. |
| description | Y | | Description of the alert. |
| conditionType | Y | “Metric”, “Event” | Two types of alerts are supported. One based on a metric condition and the other based on an event in the Activity Log. Use this value to check if the alert is based on metric or event. |

| FIELD | MANDATORY | FIXED SET OF VALUES | NOTES |
|---|---|---|---|
| condition | Y | | The specific fields to check for based on the conditionType. |
| metricName | for Metric alerts | | The name of the metric that defines what the rule monitors. |
| metricUnit | for Metric alerts | "Bytes", "BytesPerSecond", "Count", "CountPerSecond", "Percent", "Seconds" | The unit allowed in the metric. Allowed values are listed here. |
| metricValue | for Metric alerts | | The actual value of the metric that caused the alert. |
| threshold | for Metric alerts | | The threshold value at which the alert is activated. |
| windowSize | for Metric alerts | | The period of time that is used to monitor alert activity based on the threshold. Must be between 5 minutes and 1 day. ISO 8601 duration format. |
| timeAggregation | for Metric alerts | "Average", "Last", "Maximum", "Minimum", "None", "Total" | How the data that is collected should be combined over time. The default value is Average. Allowed values are listed here. |
| operator | for Metric alerts | | The operator used to compare the current metric data to the set threshold. |
| subscriptionId | Y | | Azure subscription ID. |
| resourceGroupName | Y | | Name of the resource group for the impacted resource. |
| resourceName | Y | | Resource name of the impacted resource. |
| resourceType | Y | | Resource type of the impacted resource. |
| resourceId | Y | | Resource ID of the impacted resource. |
| resourceRegion | Y | | Region or location of the impacted resource. |
| portalLink | Y | | Direct link to the portal resource summary page. |

| FIELD | MANDATORY | FIXED SET OF VALUES | NOTES |
| --- | --- | --- | --- |
| properties | N | Optional | Set of `<Key, Value>` pairs (i.e. `Dictionary<String, String>`) that includes details about the event. The properties field is optional. In a custom UI or Logic app-based workflow, users can enter key/values that can be passed via the payload. The alternate way to pass custom properties back to the webhook is via the webhook uri itself (as query parameters) |

**NOTE**

The properties field can only be set using the Azure Monitor REST API.

## Next steps

- Learn more about Azure alerts and webhooks in the video Integrate Azure Alerts with PagerDuty
- Execute Azure Automation scripts (Runbooks) on Azure alerts
- Use Logic App to send an SMS via Twilio from an Azure alert
- Use Logic App to send a Slack message from an Azure alert
- Use Logic App to send a message to an Azure Queue from an Azure alert

# Azure Monitor PowerShell quick start samples

11/20/2017 • 8 min to read • Edit Online

This article shows you sample PowerShell commands to help you access Azure Monitor features. Azure Monitor allows you to autoScale Cloud Services, Virtual Machines, and Web Apps. It also allows you to send alert notifications or call web URLs based on values of configured telemetry data.

> **NOTE**
> Azure Monitor is the new name for what was called "Azure Insights" until Sept 25th, 2016. However, the namespaces and thus the following commands still contain the the word "insights."

## Set up PowerShell

If you haven't already, set up PowerShell to run on your computer. For more information, see How to Install and Configure PowerShell.

## Examples in this article

The examples in the article illustrate how you can use Azure Monitor cmdlets. You can also review the entire list of Azure Monitor PowerShell cmdlets at Azure Monitor (Insights) Cmdlets.

## Sign in and use subscriptions

First, log in to your Azure subscription.

```
Login-AzureRmAccount
```

You'll see a sign in screen. Once you sign in your Account, TenantID, and default Subscription ID are displayed. All the Azure cmdlets work in the context of your default subscription. To view the list of subscriptions you have access to, use the following command:

```
Get-AzureRmSubscription
```

To change your working context to a different subscription, use the following command:

```
Set-AzureRmContext -SubscriptionId <subscriptionid>
```

## Retrieve Activity log for a subscription

Use the `Get-AzureRmLog` cmdlet. The following are some common examples.

Get log entries from this time/date to present:

```
Get-AzureRmLog -StartTime 2016-03-01T10:30
```

Get log entries between a time/date range:

```
Get-AzureRmLog -StartTime 2015-01-01T10:30 -EndTime 2015-01-01T11:30
```

Get log entries from a specific resource group:

```
Get-AzureRmLog -ResourceGroup 'myrg1'
```

Get log entries from a specific resource provider between a time/date range:

```
Get-AzureRmLog -ResourceProvider 'Microsoft.Web' -StartTime 2015-01-01T10:30 -EndTime 2015-01-01T11:30
```

Get all log entries with a specific caller:

```
Get-AzureRmLog -Caller 'myname@company.com'
```

The following command retrieves the last 1000 events from the activity log:

```
Get-AzureRmLog -MaxEvents 1000
```

`Get-AzureRmLog` supports many other parameters. See the `Get-AzureRmLog` reference for more information.

> **NOTE**
>
> `Get-AzureRmLog` only provides 15 days of history. Using the **-MaxEvents** parameter allows you to query the last N events, beyond 15 days. To access events older than 15 days, use the REST API or SDK (C# sample using the SDK). If you do not include **StartTime**, then the default value is **EndTime** minus one hour. If you do not include **EndTime**, then the default value is current time. All times are in UTC.

## Retrieve alerts history

To view all alert events, you can query the Azure Resource Manager logs using the following examples.

```
Get-AzureRmLog -Caller "Microsoft.Insights/alertRules" -DetailedOutput -StartTime 2015-03-01
```

To view the history for a specific alert rule, you can use the `Get-AzureRmAlertHistory` cmdlet, passing in the resource ID of the alert rule.

```
Get-AzureRmAlertHistory -ResourceId
/subscriptions/s1/resourceGroups/rg1/providers/microsoft.insights/alertrules/myalert -StartTime 2016-03-1 -
Status Activated
```

The `Get-AzureRmAlertHistory` cmdlet supports various parameters. More information, see Get-AlertHistory.

## Retrieve information on alert rules

All of the following commands act on a Resource Group named "montest".

View all the properties of the alert rule:

```
Get-AzureRmAlertRule -Name simpletestCPU -ResourceGroup montest -DetailedOutput
```

Retrieve all alerts on a resource group:

```
Get-AzureRmAlertRule -ResourceGroup montest
```

Retrieve all alert rules set for a target resource. For example, all alert rules set on a VM.

```
Get-AzureRmAlertRule -ResourceGroup montest -TargetResourceId
/subscriptions/s1/resourceGroups/montest/providers/Microsoft.Compute/virtualMachines/testconfig
```

`Get-AzureRmAlertRule` supports other parameters. See Get-AlertRule for more information.

## Create metric alerts

You can use the `Add-AlertRule` cmdlet to create, update, or disable an alert rule.

You can create email and webhook properties using `New-AzureRmAlertRuleEmail` and `New-AzureRmAlertRuleWebhook`, respectively. In the Alert rule cmdlet, assign these properties as actions to the **Actions** property of the Alert Rule.

The following table describes the parameters and values used to create an alert using a metric.

| PARAMETER | VALUE |
| --- | --- |
| Name | simpletestdiskwrite |
| Location of this alert rule | East US |
| ResourceGroup | montest |
| TargetResourceId | /subscriptions/s1/resourceGroups/montest/providers/Microsoft.Compute/virtualMachines/testconfig |
| MetricName of the alert that is created | \PhysicalDisk(_Total)\Disk Writes/sec. See the `Get-MetricDefinitions` cmdlet about how to retrieve the exact metric names |
| operator | GreaterThan |
| Threshold value (count/sec in for this metric) | 1 |
| WindowSize (hh:mm:ss format) | 00:05:00 |
| aggregator (statistic of the metric, which uses Average count, in this case) | Average |
| custom emails (string array) | 'foo@example.com','bar@example.com' |
| send email to owners, contributors and readers | -SendToServiceOwners |

Create an Email action

```
$actionEmail = New-AzureRmAlertRuleEmail -CustomEmail myname@company.com
```

Create a Webhook action

```
$actionWebhook = New-AzureRmAlertRuleWebhook -ServiceUri https://example.com?token=mytoken
```

Create the alert rule on the CPU% metric on a classic VM

```
Add-AzureRmMetricAlertRule -Name vmcpu_gt_1 -Location "East US" -ResourceGroup myrg1 -TargetResourceId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.ClassicCompute/virtualMachines/my_vm1 -MetricName
"Percentage CPU" -Operator GreaterThan -Threshold 1 -WindowSize 00:05:00 -TimeAggregationOperator Average -
Actions $actionEmail, $actionWebhook -Description "alert on CPU > 1%"
```

Retrieve the alert rule

```
Get-AzureRmAlertRule -Name vmcpu_gt_1 -ResourceGroup myrg1 -DetailedOutput
```

The Add alert cmdlet also updates the rule if an alert rule already exists for the given properties. To disable an alert rule, include the parameter **-DisableRule**.

## Get a list of available metrics for alerts

You can use the `Get-AzureRmMetricDefinition` cmdlet to view the list of all metrics for a specific resource.

```
Get-AzureRmMetricDefinition -ResourceId <resource_id>
```

The following example generates a table with the metric Name and the Unit for it.

```
Get-AzureRmMetricDefinition -ResourceId <resource_id> | Format-Table -Property Name,Unit
```

A full list of available options for `Get-AzureRmMetricDefinition` is available at Get-MetricDefinitions.

## Create and manage AutoScale settings

A resource (a Web app, VM, Cloud Service, or Virtual Machine Scale Set) can have only one autoscale setting configured for it. However, each autoscale setting can have multiple profiles. For example, one for a performance-based scale profile and a second one for a schedule-based profile. Each profile can have multiple rules configured on it. For more information about Autoscale, see How to Autoscale an Application.

Here are the steps to use:

1. Create rule(s).
2. Create profile(s) mapping the rules that you created previously to the profiles.
3. Optional: Create notifications for autoscale by configuring webhook and email properties.
4. Create an autoscale setting with a name on the target resource by mapping the profiles and notifications that you created in the previous steps.

The following examples show you how you can create an Autoscale setting for a Virtual Machine Scale Set for a Windows operating system based by using the CPU utilization metric.

First, create a rule to scale out, with an instance count increase.

```
$rule1 = New-AzureRmAutoscaleRule -MetricName "Percentage CPU" -MetricResourceId
/subscriptions/s1/resourceGroups/big2/providers/Microsoft.Compute/virtualMachineScaleSets/big2 -Operator
GreaterThan -MetricStatistic Average -Threshold 60 -TimeGrain 00:01:00 -TimeWindow 00:10:00 -
ScaleActionCooldown 00:10:00 -ScaleActionDirection Increase -ScaleActionValue 1
```

Next, create a rule to scale in, with an instance count decrease.

```
$rule2 = New-AzureRmAutoscaleRule -MetricName "Percentage CPU" -MetricResourceId
/subscriptions/s1/resourceGroups/big2/providers/Microsoft.Compute/virtualMachineScaleSets/big2 -Operator
GreaterThan -MetricStatistic Average -Threshold 30 -TimeGrain 00:01:00 -TimeWindow 00:10:00 -
ScaleActionCooldown 00:10:00 -ScaleActionDirection Decrease -ScaleActionValue 1
```

Then, create a profile for the rules.

```
$profile1 = New-AzureRmAutoscaleProfile -DefaultCapacity 2 -MaximumCapacity 10 -MinimumCapacity 2 -Rules
$rule1,$rule2 -Name "My_Profile"
```

Create a webhook property.

```
$webhook_scale = New-AzureRmAutoscaleWebhook -ServiceUri "https://example.com?mytoken=mytokenvalue"
```

Create the notification property for the autoscale setting, including email and the webhook that you created previously.

```
$notification1= New-AzureRmAutoscaleNotification -CustomEmails ashwink@microsoft.com -
SendEmailToSubscriptionAdministrators SendEmailToSubscriptionCoAdministrators -Webhooks $webhook_scale
```

Finally, create the autoscale setting to add the profile that you created previously.

```
Add-AzureRmAutoscaleSetting -Location "East US" -Name "MyScaleVMSSSetting" -ResourceGroup big2 -
TargetResourceId /subscriptions/s1/resourceGroups/big2/providers/Microsoft.Compute/virtualMachineScaleSets/big2
-AutoscaleProfiles $profile1 -Notifications $notification1
```

For more information about managing Autoscale settings, see Get-AutoscaleSetting.

# Autoscale history

The following example shows you how you can view recent autoscale and alert events. Use the activity log search to view the autoscale history.

```
Get-AzureRmLog -Caller "Microsoft.Insights/autoscaleSettings" -DetailedOutput -StartTime 2015-03-01
```

You can use the `Get-AzureRmAutoScaleHistory` cmdlet to retrieve AutoScale history.

```
Get-AzureRmAutoScaleHistory -ResourceId
/subscriptions/s1/resourceGroups/myrg1/providers/microsoft.insights/autoscalesettings/myScaleSetting -StartTime
2016-03-15 -DetailedOutput
```

For more information, see Get-AutoscaleHistory.

**View details for an autoscale setting**

You can use the `Get-Autoscalesetting` cmdlet to retrieve more information about the autoscale setting.

The following example shows details about all autoscale settings in the resource group 'myrg1'.

```
Get-AzureRmAutoscalesetting -ResourceGroup myrg1 -DetailedOutput
```

The following example shows details about all autoscale settings in the resource group 'myrg1' and specifically the autoscale setting named 'MyScaleVMSSSetting'.

```
Get-AzureRmAutoscalesetting -ResourceGroup myrg1 -Name MyScaleVMSSSetting -DetailedOutput
```

#### Remove an autoscale setting

You can use the `Remove-Autoscalesetting` cmdlet to delete an autoscale setting.

```
Remove-AzureRmAutoscalesetting -ResourceGroup myrg1 -Name MyScaleVMSSSetting
```

## Manage log profiles for activity log

You can create a *log profile* and export data from your activity log to a storage account and you can configure data retention for it. Optionally, you can also stream the data to your Event Hub. This feature is currently in Preview and you can only create one log profile per subscription. You can use the following cmdlets with your current subscription to create and manage log profiles. You can also choose a particular subscription. Although PowerShell defaults to the current subscription, you can always change that using `Set-AzureRmContext`. You can configure activity log to route data to any storage account or Event Hub within that subscription. Data is written as blob files in JSON format.

#### Get a log profile

To fetch your existing log profiles, use the `Get-AzureRmLogProfile` cmdlet.

#### Add a log profile without data retention

```
Add-AzureRmLogProfile -Name my_log_profile_s1 -StorageAccountId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Storage/storageAccounts/my_storage -Locations
global,westus,eastus,northeurope,westeurope,eastasia,southeastasia,japaneast,japanwest,northcentralus,southcent
ralus,eastus2,centralus,australiaeast,australiasoutheast,brazilsouth,centralindia,southindia,westindia
```

#### Remove a log profile

```
Remove-AzureRmLogProfile -name my_log_profile_s1
```

#### Add a log profile with data retention

You can specify the **-RetentionInDays** property with the number of days, as a positive integer, where the data is retained.

```
Add-AzureRmLogProfile -Name my_log_profile_s1 -StorageAccountId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Storage/storageAccounts/my_storage -Locations
global,westus,eastus,northeurope,westeurope,eastasia,southeastasia,japaneast,japanwest,northcentralus,southcent
ralus,eastus2,centralus,australiaeast,australiasoutheast,brazilsouth,centralindia,southindia,westindia -
RetentionInDays 90
```

#### Add log profile with retention and EventHub

In addition to routing your data to storage account, you can also stream it to an Event Hub. In this preview release the storage account configuration is mandatory but Event Hub configuration is optional.

```
Add-AzureRmLogProfile -Name my_log_profile_s1 -StorageAccountId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Storage/storageAccounts/my_storage -serviceBusRuleId
/subscriptions/s1/resourceGroups/Default-ServiceBus-
EastUS/providers/Microsoft.ServiceBus/namespaces/mytestSB/authorizationrules/RootManageSharedAccessKey -
Locations
global,westus,eastus,northeurope,westeurope,eastasia,southeastasia,japaneast,japanwest,northcentralus,southcent
ralus,eastus2,centralus,australiaeast,australiasoutheast,brazilsouth,centralindia,southindia,westindia -
RetentionInDays 90
```

# Configure diagnostics logs

Many Azure services provide additional logs and telemetry that can do one or more of the following:

- be configured to save data in your Azure Storage account
- sent to Event Hubs
- sent to an OMS Log Analytics workspace.

The operation can only be performed at a resource level. The storage account or event hub should be present in the same region as the target resource where the diagnostics setting is configured.

**Get diagnostic setting**

```
Get-AzureRmDiagnosticSetting -ResourceId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Logic/workflows/andy0315logicapp
```

Disable diagnostic setting

```
Set-AzureRmDiagnosticSetting -ResourceId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Logic/workflows/andy0315logicapp -StorageAccountId
/subscriptions/s1/resourceGroups/Default-Storage-
WestUS/providers/Microsoft.Storage/storageAccounts/mystorageaccount -Enable $false
```

Enable diagnostic setting without retention

```
Set-AzureRmDiagnosticSetting -ResourceId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Logic/workflows/andy0315logicapp -StorageAccountId
/subscriptions/s1/resourceGroups/Default-Storage-
WestUS/providers/Microsoft.Storage/storageAccounts/mystorageaccount -Enable $true
```

Enable diagnostic setting with retention

```
Set-AzureRmDiagnosticSetting -ResourceId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Logic/workflows/andy0315logicapp -StorageAccountId
/subscriptions/s1/resourceGroups/Default-Storage-
WestUS/providers/Microsoft.Storage/storageAccounts/mystorageaccount -Enable $true -RetentionEnabled $true -
RetentionInDays 90
```

Enable diagnostic setting with retention for a specific log category

```
Set-AzureRmDiagnosticSetting -ResourceId /subscriptions/s1/resourceGroups/insights-
integration/providers/Microsoft.Network/networkSecurityGroups/viruela1 -StorageAccountId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Storage/storageAccounts/saktestst orage -Categories
NetworkSecurityGroupEvent -Enable $true -RetentionEnabled $true -RetentionInDays 90
```

Enable diagnostic setting for Event Hubs

```
Set-AzureRmDiagnosticSetting -ResourceId /subscriptions/s1/resourceGroups/insights-
integration/providers/Microsoft.Network/networkSecurityGroups/viruela1 -serviceBusRuleId
/subscriptions/s1/resourceGroups/Default-ServiceBus-
EastUS/providers/Microsoft.ServiceBus/namespaces/mytestSB/authorizationrules/RootManageSharedAccessKey -Enable
$true
```

Enable diagnostic setting for Log Analytics (OMS)

```
Set-AzureRmDiagnosticSetting -ResourceId /subscriptions/s1/resourceGroups/insights-
integration/providers/Microsoft.Network/networkSecurityGroups/viruela1 -WorkspaceId
/subscriptions/s1/resourceGroups/insights-
integration/providers/providers/microsoft.operationalinsights/workspaces/myWorkspace -Enabled $true
```

Note that the WorkspaceId property takes the *resource ID* of the workspace. You can obtain the resource ID of your Log Analytics workspace using the following command:

```
(Get-AzureRmOperationalInsightsWorkspace).ResourceId
```

These commands can be combined to send data to multiple destinations.

# Azure Monitor Cross-platform CLI 1.0 quick start samples

8/10/2017 • 3 min to read • Edit Online

This article shows you sample command-line interface (CLI) commands to help you access Azure Monitor features. Azure Monitor allows you to AutoScale Cloud Services, Virtual Machines, and Web Apps and to send alert notifications or call web URLs based on values of configured telemetry data.

> **NOTE**
> Azure Monitor is the new name for what was called "Azure Insights" until Sept 25th, 2016. However, the namespaces and thus the commands below still contain the "insights".

## Prerequisites

If you haven't already installed the Azure CLI, see Install the Azure CLI. If you're unfamiliar with Azure CLI, you can read more about it at Use the Azure CLI for Mac, Linux, and Windows with Azure Resource Manager.

In Windows, install npm from the Node.js website. After you complete the installation, using CMD.exe with Run As Administrator privileges, execute the following from the folder where npm is installed:

```
npm install azure-cli --global
```

Next, navigate to any folder/location you want and type at the command-line:

```
azure help
```

## Log in to Azure

The first step is to login to your Azure account.

```
azure login
```

After running this command, you have to sign in via the instructions on the screen. Afterward, you see your Account, TenantId, and default Subscription Id. All commands work in the context of your default subscription.

To list the details of your current subscription, use the following command.

```
azure account show
```

To change working context to a different subscription, use the following command.

```
azure account set "subscription ID or subscription name"
```

To use Azure Resource Manager and Azure Monitor commands, you need to be in Azure Resource Manager mode.

```
azure config mode arm
```

To view a list of all supported Azure Monitor commands, perform the following.

```
azure insights
```

# View activity log for a subscription

To view a list of activity log events, perform the following.

```
azure insights logs list [options]
```

Try the following to view all available options.

```
azure insights logs list -help
```

Here is an example to list logs by a resourceGroup

```
azure insights logs list --resourceGroup "myrg1"
```

Example to list logs by caller

```
azure insights logs list --caller "myname@company.com"
```

Example to list logs by caller on a resource type, within a start and end date

```
azure insights logs list --resourceProvider "Microsoft.Web" --caller "myname@company.com" --startTime 2016-03-
08T00:00:00Z --endTime 2016-03-16T00:00:00Z
```

# Work with alerts

You can use the information in the section to work with alerts.

**Get alert rules in a resource group**

```
azure insights alerts rule list abhingrgtest123
azure insights alerts rule list abhingrgtest123 --ruleName andy0323
```

**Create a metric alert rule**

```
azure insights alerts actions email create --customEmails foo@microsoft.com
azure insights alerts actions webhook create https://someuri.com
azure insights alerts rule metric set andy0323 eastus abhingrgtest123 PT5M GreaterThan 2
/subscriptions/df602c9c-7aa0-407d-a6fb-eb20c8bd1192/resourceGroups/Default-Web-
EastUS/providers/Microsoft.Web/serverfarms/Default1 BytesReceived Total
```

**Create webtest alert rule**

```
azure insights alerts rule webtest set leowebtestr1-webtestr1 eastus Default-Web-WestUS PT5M 1 GSMT_AvRaw
/subscriptions/b67f7fec-69fc-4974-9099-a26bd6ffeda3/resourcegroups/Default-Web-
WestUS/providers/microsoft.insights/webtests/leowebtestr1-webtestr1
```

#### Delete an alert rule

```
azure insights alerts rule delete abhingrgtest123 andy0323
```

## Log profiles

Use the information in this section to work with log profiles.

#### Get a log profile

```
azure insights logprofile list
azure insights logprofile get -n default
```

#### Add a log profile without retention

```
azure insights logprofile add --name default --storageId /subscriptions/1a66ce04-b633-4a0b-b2bc-
a912ec8986a6/resourceGroups/insights-
integration/providers/Microsoft.Storage/storageAccounts/insightsintegration7777 --locations
global,westus,eastus,northeurope,westeurope,eastasia,southeastasia,japaneast,japanwest,northcentralus,southcent
ralus,eastus2,centralus,australiaeast,australiasoutheast,brazilsouth,centralindia,southindia,westindia
```

#### Remove a log profile

```
azure insights logprofile delete --name default
```

#### Add a log profile with retention

```
azure insights logprofile add --name default --storageId /subscriptions/1a66ce04-b633-4a0b-b2bc-
a912ec8986a6/resourceGroups/insights-
integration/providers/Microsoft.Storage/storageAccounts/insightsintegration7777 --locations
global,westus,eastus,northeurope,westeurope,eastasia,southeastasia,japaneast,japanwest,northcentralus,southcent
ralus,eastus2,centralus,australiaeast,australiasoutheast,brazilsouth,centralindia,southindia,westindia --
retentionInDays 90
```

#### Add a log profile with retention and EventHub

```
azure insights logprofile add --name default --storageId /subscriptions/1a66ce04-b633-4a0b-b2bc-
a912ec8986a6/resourceGroups/insights-
integration/providers/Microsoft.Storage/storageAccounts/insightsintegration7777 --serviceBusRuleId
/subscriptions/1a66ce04-b633-4a0b-b2bc-a912ec8986a6/resourceGroups/Default-ServiceBus-
EastUS/providers/Microsoft.ServiceBus/namespaces/testshoeboxeastus/authorizationrules/RootManageSharedAccessKey
--locations
global,westus,eastus,northeurope,westeurope,eastasia,southeastasia,japaneast,japanwest,northcentralus,southcent
ralus,eastus2,centralus,australiaeast,australiasoutheast,brazilsouth,centralindia,southindia,westindia --
retentionInDays 90
```

## Diagnostics

Use the information in this section to work with diagnostic settings.

#### Get a diagnostic setting

```
azure insights diagnostic get --resourceId /subscriptions/df602c9c-7aa0-407d-a6fb-
eb20c8bd1192/resourceGroups/andyrg/providers/Microsoft.Logic/workflows/andy0315logicapp
```

#### Disable a diagnostic setting

```
azure insights diagnostic set --resourceId /subscriptions/df602c9c-7aa0-407d-a6fb-
eb20c8bd1192/resourceGroups/andyrg/providers/Microsoft.Logic/workflows/andy0315logicapp --storageId
/subscriptions/df602c9c-7aa0-407d-a6fb-eb20c8bd1192/resourceGroups/Default-Storage-
WestUS/providers/Microsoft.Storage/storageAccounts/shibanitesting --enabled false
```

#### Enable a diagnostic setting without retention

```
azure insights diagnostic set --resourceId /subscriptions/df602c9c-7aa0-407d-a6fb-
eb20c8bd1192/resourceGroups/andyrg/providers/Microsoft.Logic/workflows/andy0315logicapp --storageId
/subscriptions/df602c9c-7aa0-407d-a6fb-eb20c8bd1192/resourceGroups/Default-Storage-
WestUS/providers/Microsoft.Storage/storageAccounts/shibanitesting --enabled true
```

## Autoscale

Use the information in this section to work with autoscale settings. You need to modify these examples.

#### Get autoscale settings for a resource group

```
azure insights autoscale setting list montest2
```

#### Get autoscale settings by name in a resource group

```
azure insights autoscale setting list montest2 -n setting2
```

#### Set auotoscale settings

```
azure insights autoscale setting set montest2 -n setting2 --settingSpec
```