Contents

Overview of Azure Cloud Shell

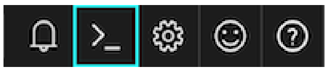
9/3/2019 • 2 minutes to read • Edit Online

Azure Cloud Shell is an interactive, authenticated, browser-accessible shell for managing Azure resources. It provides the flexibility of choosing the shell experience that best suits the way you work, either Bash or PowerShell.

Try from shell.azure.com by clicking below.

Launch Cloud Shell

Try from Azure portal using the Cloud Shell icon.



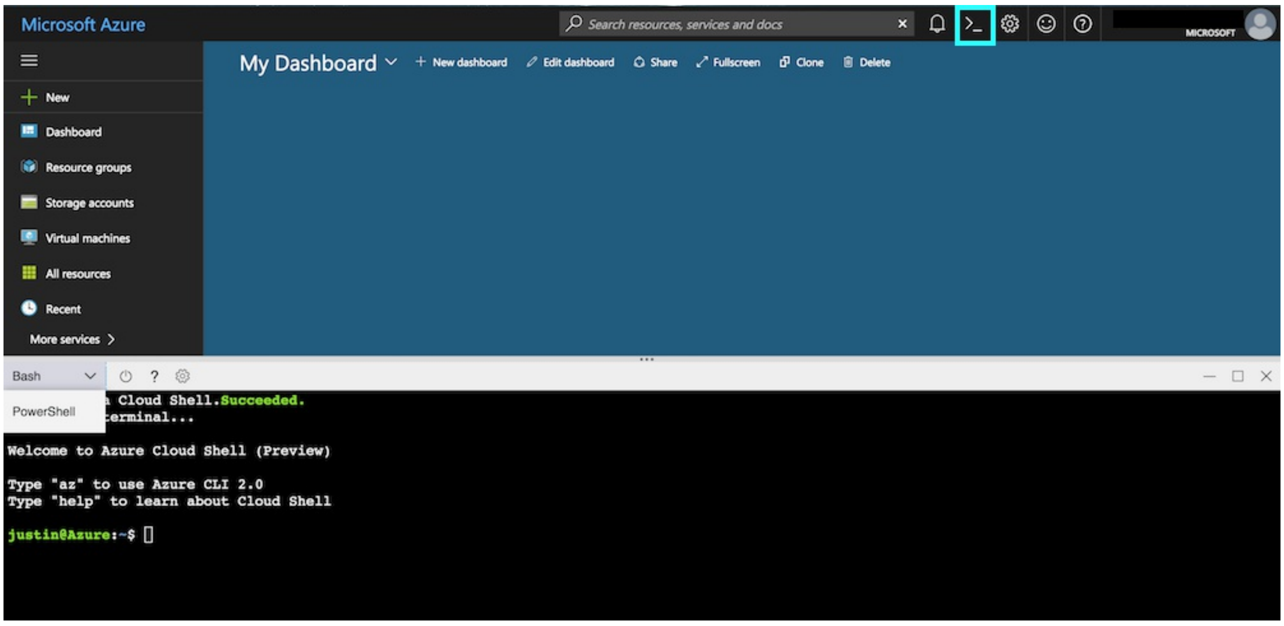
Features

Browser-based shell experience

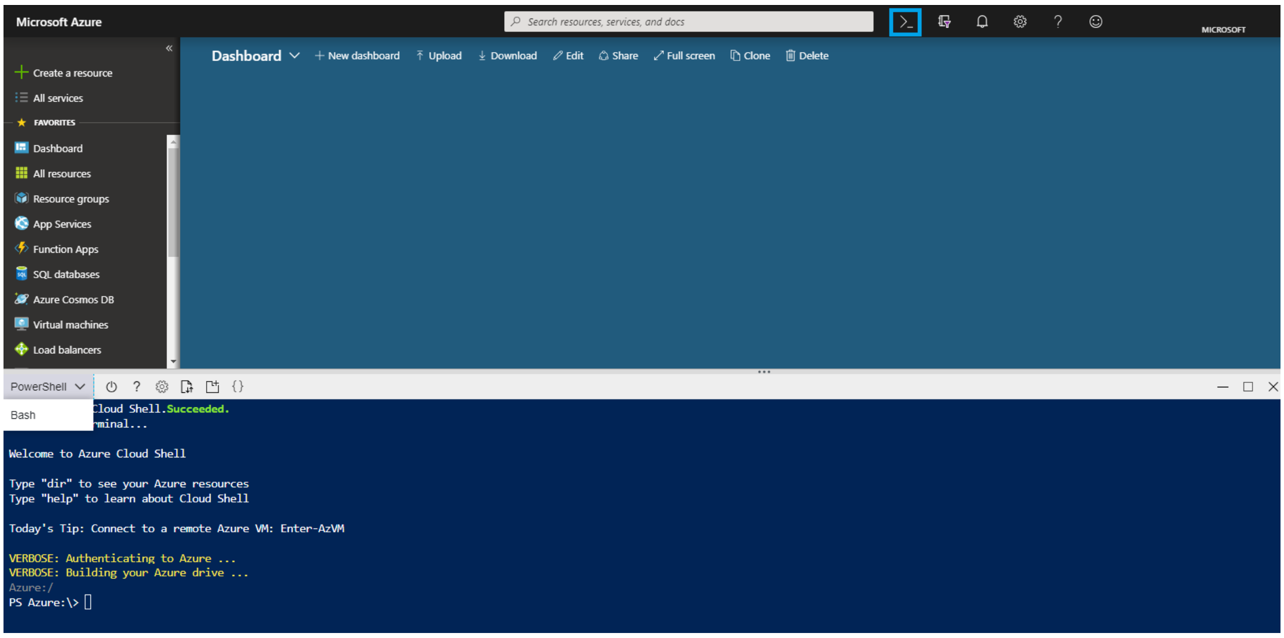
Cloud Shell enables access to a browser-based command-line experience built with Azure management tasks in mind. Leverage Cloud Shell to work untethered from a local machine in a way only the cloud can provide.

Choice of preferred shell experience

Users can choose between Bash or PowerShell from the shell dropdown.



```
 Cloud Shell.Succeeded.
erminal...

Welcome to Azure Cloud Shell (Preview)

Type "az" to use Azure CLI 2.0
Type "help" to learn about Cloud Shell

justin@Azure:~$ 
```



```
Cloud Shell.Succeeded.
rminal...

Welcome to Azure Cloud Shell

Type "dir" to see your Azure resources
Type "help" to learn about Cloud Shell

Today's Tip: Connect to a remote Azure VM: Enter-AzVM

VERBOSE: Authenticating to Azure ...
VERBOSE: Building your Azure drive ...
Azure:/
PS Azure:\> 
```

Authenticated and configured Azure workstation

Cloud Shell is managed by Microsoft so it comes with popular command-line tools and language support. Cloud Shell also securely authenticates automatically for instant access to your resources through the Azure CLI or Azure PowerShell cmdlets.

View the full list of tools installed in Cloud Shell.

Integrated Cloud Shell editor

Cloud Shell offers an integrated graphical text editor based on the open-source Monaco Editor. Simply create and edit configuration files by running `code .` for seamless deployment through Azure CLI or Azure PowerShell.

Learn more about the Cloud Shell editor.

Integrated with docs.microsoft.com

You can use Cloud Shell directly from documentation hosted on docs.microsoft.com. It is integrated in Microsoft Learn, Azure PowerShell and Azure CLI documentation - click on the "Try It" button in a code snippet to open the immersive shell experience.

Multiple access points

Cloud Shell is a flexible tool that can be used from:

- portal.azure.com
- shell.azure.com
- Azure CLI documentation
- Azure PowerShell documentation
- Azure mobile app
- Visual Studio Code Azure Account extension

Connect your Microsoft Azure Files storage

Cloud Shell machines are temporary, but your files are persisted in two ways: through a disk image, and through a mounted file share named `clouddrive`. On first launch, Cloud Shell prompts to create a resource group, storage account, and Azure Files share on your behalf. This is a one-time step and will be automatically attached for all sessions. A single file share can be mapped and will be used by both Bash and PowerShell in Cloud Shell.

Read more to learn how to mount a new or existing storage account or to learn about the persistence mechanisms used in Cloud Shell.

Concepts

- Cloud Shell runs on a temporary host provided on a per-session, per-user basis
- Cloud Shell times out after 20 minutes without interactive activity
- Cloud Shell requires an Azure file share to be mounted
- Cloud Shell uses the same Azure file share for both Bash and PowerShell
- Cloud Shell is assigned one machine per user account
- Cloud Shell persists $HOME using a 5-GB image held in your file share
- Permissions are set as a regular Linux user in Bash

Learn more about features in Bash in Cloud Shell and PowerShell in Cloud Shell.

Pricing

The machine hosting Cloud Shell is free, with a pre-requisite of a mounted Azure Files share. Regular storage costs apply.

Next steps

Bash in Cloud Shell quickstart
PowerShell in Cloud Shell quickstart

Quickstart for Bash in Azure Cloud Shell

10/22/2019 • 2 minutes to read • Edit Online

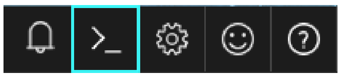
This document details how to use Bash in Azure Cloud Shell in the Azure portal.

NOTE

A PowerShell in Azure Cloud Shell Quickstart is also available.

Start Cloud Shell

1. Launch **Cloud Shell** from the top navigation of the Azure portal.



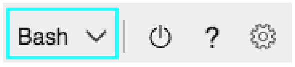
2. Select a subscription to create a storage account and Microsoft Azure Files share.
3. Select "Create storage"

TIP

You are automatically authenticated for Azure CLI in every session.

Select the Bash environment

Check that the environment drop-down from the left-hand side of shell window says `Bash`.



Set your subscription

1. List subscriptions you have access to.

```
az account list
```

2. Set your preferred subscription:

```
az account set --subscription 'my-subscription-name'
```

TIP

Your subscription will be remembered for future sessions using `/home/<user>/.azure/azureProfile.json`.

Create a resource group

Create a new resource group in WestUS named "MyRG".

```
az group create --location westus --name MyRG
```

Create a Linux VM

Create an Ubuntu VM in your new resource group. The Azure CLI will create SSH keys and set up the VM with them.

```
az vm create -n myVM -g MyRG --image UbuntuLTS --generate-ssh-keys
```

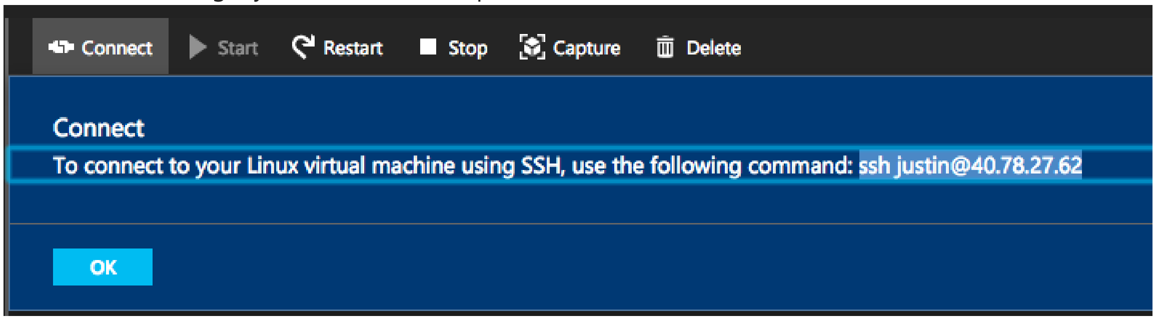
NOTE

Using `--generate-ssh-keys` instructs Azure CLI to create and set up public and private keys in your VM and `$Home` directory. By default keys are placed in Cloud Shell at `/home/<user>/.ssh/id_rsa` and `/home/<user>/.ssh/id_rsa.pub`. Your `.ssh` folder is persisted in your attached file share's 5-GB image used to persist `$Home`.

Your username on this VM will be your username used in Cloud Shell ($User@Azure:).

SSH into your Linux VM

1. Search for your VM name in the Azure portal search bar.
2. Click "Connect" to get your VM name and public IP address.



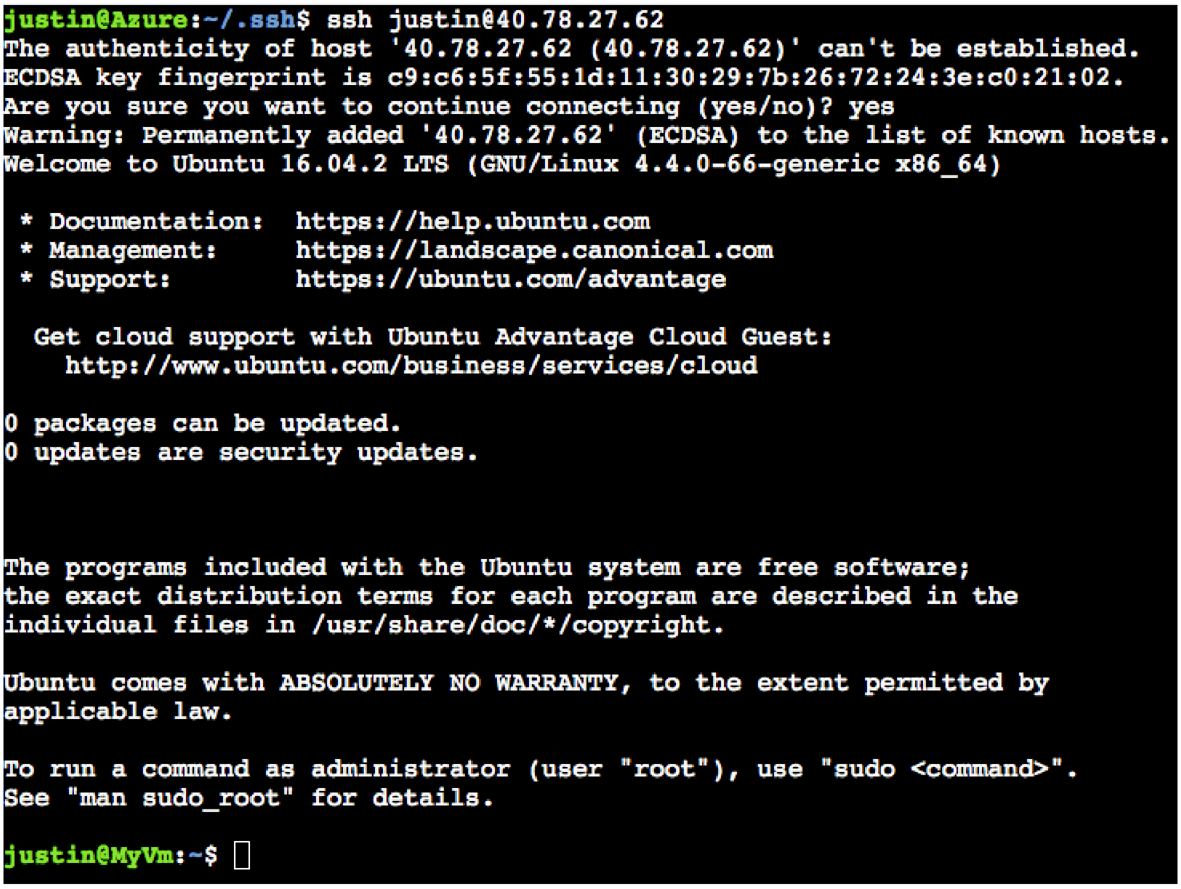
Connect

To connect to your Linux virtual machine using SSH, use the following command: ssh justin@40.78.27.62

3. SSH into your VM with the `ssh` cmd.

```
ssh username@ipaddress
```

Upon establishing the SSH connection, you should see the Ubuntu welcome prompt.



```
justin@Azure:~/.ssh$ ssh justin@40.78.27.62
The authenticity of host '40.78.27.62 (40.78.27.62)' can't be established.
ECDSA key fingerprint is c9:c6:5f:55:1d:11:30:29:7b:26:72:24:3e:c0:21:02.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '40.78.27.62' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-66-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

justin@MyVm:~$
```

Cleaning up

1. Exit your ssh session.

```
exit
```

2. Delete your resource group and any resources within it.

```
az group delete -n MyRG
```

Next steps

Learn about persisting files for Bash in Cloud Shell
Learn about Azure CLI
Learn about Azure Files storage

Quickstart for PowerShell in Azure Cloud Shell

10/22/2019 • 5 minutes to read • Edit Online

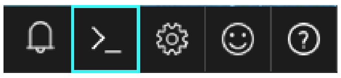
This document details how to use the PowerShell in Cloud Shell in the Azure portal.

NOTE

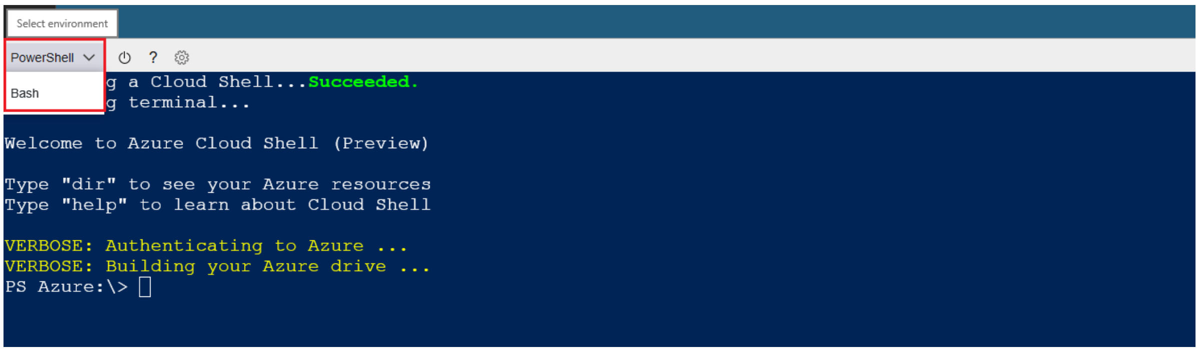
A Bash in Azure Cloud Shell Quickstart is also available.

Start Cloud Shell

1. Click on **Cloud Shell** button from the top navigation bar of the Azure portal



2. Select the PowerShell environment from the drop-down and you will be in Azure drive `(Azure:)`



```
g a Cloud Shell...Succeeded.
g terminal...

Welcome to Azure Cloud Shell (Preview)

Type "dir" to see your Azure resources
Type "help" to learn about Cloud Shell

VERBOSE: Authenticating to Azure ...
VERBOSE: Building your Azure drive ...
PS Azure:\>
```

Run PowerShell commands

Run regular PowerShell commands in the Cloud Shell, such as:

```
PS Azure:\> Get-Date

# Expected Output
Friday, July 27, 2018 7:08:48 AM

PS Azure:\> Get-AzVM -Status

# Expected Output
ResourceGroupName       Name       Location                VmSize  OsType     ProvisioningState  PowerState
-----------------       ----       --------                ------  ------     -----------------  ----------
MyResourceGroup2        Demo        westus         Standard_DS1_v2  Windows    Succeeded           running
MyResourceGroup         MyVM1       eastus            Standard_DS1  Windows    Succeeded           running
MyResourceGroup         MyVM2       eastus   Standard_DS2_v2_Promo  Windows    Succeeded           deallocated
```

Navigate Azure resources

1. List all your subscriptions from `Azure` drive

```
PS Azure:\> dir
```

2. `cd` to your preferred subscription

   ```
   PS Azure:\> cd MySubscriptionName
   PS Azure:\MySubscriptionName>
   ```

3. View all your Azure resources under the current subscription

   Type `dir` to list multiple views of your Azure resources.

   ```
   PS Azure:\MySubscriptionName> dir

       Directory: azure:\MySubscriptionName

   Mode Name
   ---- ----
   +    AllResources
   +    ResourceGroups
   +    StorageAccounts
   +    VirtualMachines
   +    WebApps
   ```

AllResources view

Type `dir` under `AllResources` directory to view your Azure resources.

```
PS Azure:\MySubscriptionName> dir AllResources
```

Explore resource groups

You can go to the `ResourceGroups` directory and inside a specific resource group you can find virtual machines.

```
PS Azure:\MySubscriptionName> cd ResourceGroups\MyResourceGroup1\Microsoft.Compute\virtualMachines

PS Azure:\MySubscriptionName\ResourceGroups\MyResourceGroup1\Microsoft.Compute\virtualMachines> dir


    Directory: Azure:\MySubscriptionName\ResourceGroups\MyResourceGroup1\Microsoft.Compute\virtualMachines


VMName    Location    ProvisioningState VMSize          OS            SKU             OSVersion AdminUserName
NetworkInterfaceName
------    --------    ----------------- ------          --            ---             --------- -------------
--------------------
TestVm1   westus      Succeeded         Standard_DS2_v2 WindowsServer 2016-Datacenter Latest    AdminUser
demo371
TestVm2   westus      Succeeded         Standard_DS1_v2 WindowsServer 2016-Datacenter Latest    AdminUser
demo271
```

> **NOTE**
>
> You may notice that the second time when you type `dir`, the Cloud Shell is able to display the items much faster. This is because the child items are cached in memory for a better user experience. However, you can always use `dir -Force` to get fresh data.

Navigate storage resources

By entering into the `StorageAccounts` directory, you can easily navigate all your storage resources

```
PS Azure:\MySubscriptionName\StorageAccounts\MyStorageAccountName\Files> dir

    Directory: Azure:\MySubscriptionNameStorageAccounts\MyStorageAccountName\Files

Name          ConnectionString
----          ----------------
MyFileShare1  \\MyStorageAccountName.file.core.windows.net\MyFileShare1;AccountName=MyStorageAccountName
AccountKey=<key>
MyFileShare2  \\MyStorageAccountName.file.core.windows.net\MyFileShare2;AccountName=MyStorageAccountName
AccountKey=<key>
MyFileShare3  \\MyStorageAccountName.file.core.windows.net\MyFileShare3;AccountName=MyStorageAccountName
AccountKey=<key>
```

With the connection string, you can use the following command to mount the Azure Files share.

```
net use <DesiredDriveLetter>: \\<MyStorageAccountName>.file.core.windows.net\<MyFileShareName> <AccountKey>
/user:Azure\<MyStorageAccountName>
```

For details, see Mount an Azure Files share and access the share in Windows.

You can also navigate the directories under the Azure Files share as follows:

```
PS Azure:\MySubscriptionName\StorageAccounts\MyStorageAccountName\Files> cd .\MyFileShare1\
PS Azure:\MySubscriptionName\StorageAccounts\MyStorageAccountName\Files\MyFileShare1> dir

Mode  Name
----  ----
+     TestFolder
.     hello.ps1
```

Interact with virtual machines

You can find all your virtual machines under the current subscription via `VirtualMachines` directory.

```
PS Azure:\MySubscriptionName\VirtualMachines> dir

    Directory: Azure:\MySubscriptionName\VirtualMachines


Name       ResourceGroupName  Location  VmSize          OsType            NIC ProvisioningState  PowerState
----       -----------------  --------  ------          ------            --- -----------------  ----------
TestVm1    MyResourceGroup1   westus    Standard_DS2_v2 Windows       my2008r213         Succeeded     stopped
TestVm2    MyResourceGroup1   westus    Standard_DS1_v2 Windows          jpstest         Succeeded deallocated
TestVm10   MyResourceGroup2   eastus    Standard_DS1_v2 Windows           mytest         Succeeded     running
```

Invoke PowerShell script across remote VMs

WARNING

Please refer to Troubleshooting remote management of Azure VMs.

Assuming you have a VM, MyVM1, let's use `Invoke-AzVMCommand` to invoke a PowerShell script block on the remote machine.

```
Enable-AzVMPSRemoting -Name MyVM1 -ResourceGroupname MyResourceGroup
Invoke-AzVMCommand -Name MyVM1 -ResourceGroupName MyResourceGroup -Scriptblock {Get-ComputerInfo} -Credential
(Get-Credential)
```

You can also navigate to the VirtualMachines directory first and run `Invoke-AzVMCommand` as follows.

```
PS Azure:\> cd MySubscriptionName\ResourceGroups\MyResourceGroup\Microsoft.Compute\virtualMachines
PS Azure:\MySubscriptionName\ResourceGroups\MyResourceGroup\Microsoft.Compute\virtualMachines> Get-Item MyVM1
| Invoke-AzVMCommand -Scriptblock {Get-ComputerInfo} -Credential (Get-Credential)

# You will see output similar to the following:

PSComputerName                                          : 65.52.28.207
RunspaceId                                              : 2c2b60da-f9b9-4f42-a282-93316cb06fe1
WindowsBuildLabEx                                       : 14393.1066.amd64fre.rs1_release_sec.170327-1835
WindowsCurrentVersion                                   : 6.3
WindowsEditionId                                        : ServerDatacenter
WindowsInstallationType                                 : Server
WindowsInstallDateFromRegistry                          : 5/18/2017 11:26:08 PM
WindowsProductId                                        : 00376-40000-00000-AA947
WindowsProductName                                      : Windows Server 2016 Datacenter
WindowsRegisteredOrganization                           :
 ...
```

Interactively log on to a remote VM

You can use `Enter-AzVM` to interactively log into a VM running in Azure.

```
PS Azure:\> Enter-AzVM -Name MyVM1 -ResourceGroupName MyResourceGroup -Credential (Get-Credential)
```

You can also navigate to the `VirtualMachines` directory first and run `Enter-AzVM` as follows

```
PS Azure:\MySubscriptionName\ResourceGroups\MyResourceGroup\Microsoft.Compute\virtualMachines> Get-Item MyVM1
| Enter-AzVM -Credential (Get-Credential)
```

Discover WebApps

By entering into the `WebApps` directory, you can easily navigate your web apps resources

```
PS Azure:\MySubscriptionName> dir .\WebApps\

    Directory: Azure:\MySubscriptionName\WebApps

Name            State    ResourceGroup      EnabledHostNames                  Location
----            -----    -------------      ----------------                  --------
mywebapp1       Stopped  MyResourceGroup1   {mywebapp1.azurewebsites.net...   West US
mywebapp2       Running  MyResourceGroup2   {mywebapp2.azurewebsites.net...   West Europe
mywebapp3       Running  MyResourceGroup3   {mywebapp3.azurewebsites.net...   South Central US

# You can use Azure cmdlets to Start/Stop your web apps
PS Azure:\MySubscriptionName\WebApps> Start-AzWebApp -Name mywebapp1 -ResourceGroupName MyResourceGroup1

Name           State    ResourceGroup        EnabledHostNames                   Location
----           -----    -------------        ----------------                   --------
mywebapp1      Running  MyResourceGroup1     {mywebapp1.azurewebsites.net ...   West US

# Refresh the current state with -Force
PS Azure:\MySubscriptionName\WebApps> dir -Force

    Directory: Azure:\MySubscriptionName\WebApps

Name            State    ResourceGroup      EnabledHostNames                  Location
----            -----    -------------      ----------------                  --------
mywebapp1       Running  MyResourceGroup1   {mywebapp1.azurewebsites.net...   West US
mywebapp2       Running  MyResourceGroup2   {mywebapp2.azurewebsites.net...   West Europe
mywebapp3       Running  MyResourceGroup3   {mywebapp3.azurewebsites.net...   South Central US
```

SSH

To authenticate to servers or VMs using SSH, generate the public-private key pair in Cloud Shell and publish the public key to `authorized_keys` on the remote machine, such as `/home/user/.ssh/authorized_keys`.

> **NOTE**
>
> You can create SSH private-public keys using `ssh-keygen` and publish them to `$env:USERPROFILE\.ssh` in Cloud Shell.

Using SSH

Follow instructions here to create a new VM configuration using Azure PowerShell cmdlets. Before calling into `New-AzVM` to kick off the deployment, add SSH public key to the VM configuration. The newly created VM will contain the public key in the `~\.ssh\authorized_keys` location, thereby enabling credential-free SSH session to the VM.

```
# Create VM config object - $vmConfig using instructions on linked page above

# Generate SSH keys in Cloud Shell
ssh-keygen -t rsa -b 2048 -f $HOME\.ssh\id_rsa

# Ensure VM config is updated with SSH keys
$sshPublicKey = Get-Content "$HOME\.ssh\id_rsa.pub"
Add-AzVMSshPublicKey -VM $vmConfig -KeyData $sshPublicKey -Path "/home/azureuser/.ssh/authorized_keys"

# Create a virtual machine
New-AzVM -ResourceGroupName <yourResourceGroup> -Location <vmLocation> -VM $vmConfig

# SSH to the VM
ssh azureuser@MyVM.Domain.Com
```

List available commands

Under `Azure` drive, type `Get-AzCommand` to get context-specific Azure commands.

Alternatively, you can always use `Get-Command *az* -Module Az.*` to find out the available Azure commands.

Install custom modules

You can run `Install-Module` to install modules from the PowerShell Gallery.

Get-Help

Type `Get-Help` to get information about PowerShell in Azure Cloud Shell.

```
Get-Help
```

For a specific command, you can still do `Get-Help` followed by a cmdlet.

```
Get-Help Get-AzVM
```

Use Azure Files to store your data

You can create a script, say `helloworld.ps1`, and save it to your `clouddrive` to use it across shell sessions.

```
cd $HOME\clouddrive
# Create a new file in clouddrive directory
New-Item helloworld.ps1
# Open the new file for editing
code .\helloworld.ps1
# Add the content, such as 'Hello World!'
.\helloworld.ps1
Hello World!
```

Next time when you use PowerShell in Cloud Shell, the `helloworld.ps1` file will exist under the `$HOME\clouddrive` directory that mounts your Azure Files share.

Use custom profile

You can customize your PowerShell environment, by creating PowerShell profile(s) - `profile.ps1` (or `Microsoft.PowerShell_profile.ps1`). Save it under `$profile.CurrentUserAllHosts` (or `$profile.CurrentUserAllHosts`), so that it can be loaded in every PowerShell in Cloud Shell session.

For how to create a profile, refer to About Profiles.

Use Git

To clone a Git repo in the Cloud Shell, you need to create a personal access token and use it as the username. Once you have your token, clone the repository as follows:

```
git clone https://<your-access-token>@github.com/username/repo.git
```

Exit the shell

Type `exit` to terminate the session.

Features & tools for Azure Cloud Shell

8/2/2019 • 2 minutes to read • Edit Online

Azure Cloud Shell is a browser-based shell experience to manage and develop Azure resources.

Cloud Shell offers a browser-accessible, pre-configured shell experience for managing Azure resources without the overhead of installing, versioning, and maintaining a machine yourself.

Cloud Shell provisions machines on a per-request basis and as a result machine state will not persist across sessions. Since Cloud Shell is built for interactive sessions, shells automatically terminate after 20 minutes of shell inactivity.

Azure Cloud Shell runs on `Ubuntu 16.04 LTS`.

Features

Secure automatic authentication

Cloud Shell securely and automatically authenticates account access for the Azure CLI and Azure PowerShell.

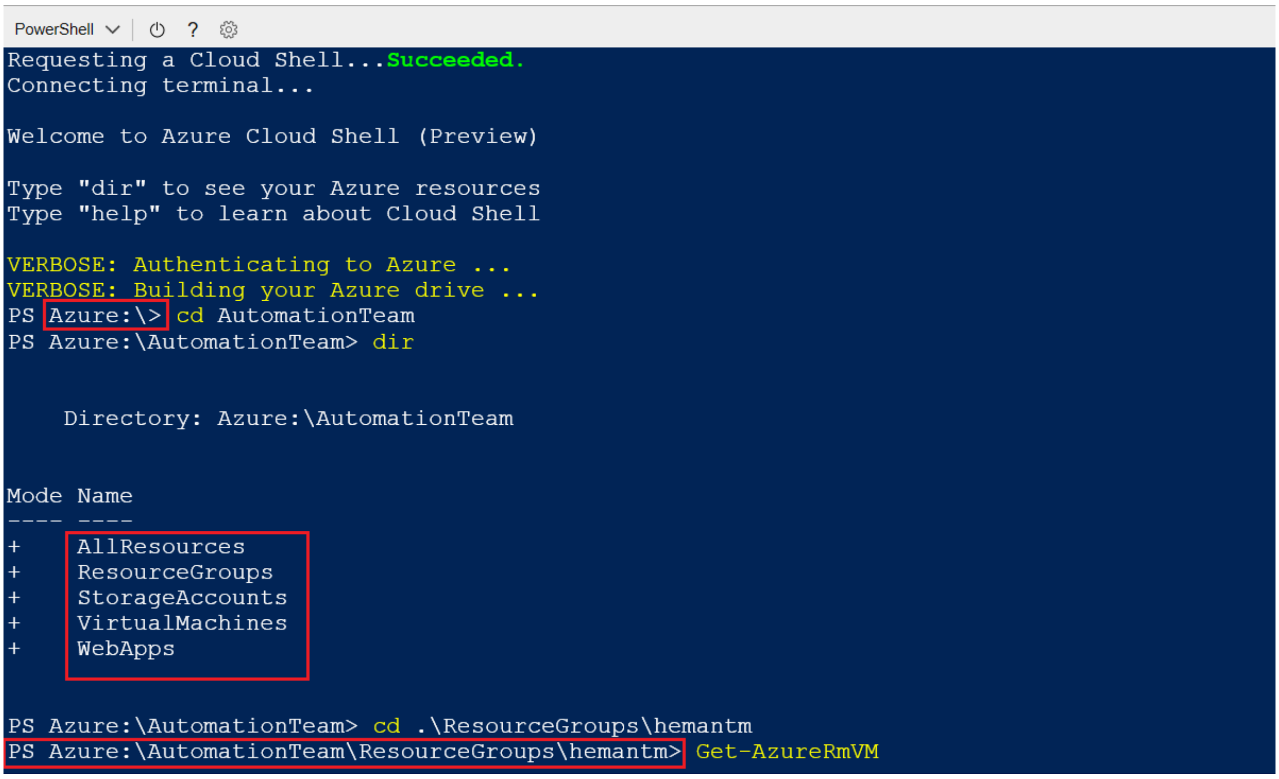
$HOME persistence across sessions

To persist files across sessions, Cloud Shell walks you through attaching an Azure file share on first launch. Once completed, Cloud Shell will automatically attach your storage (mounted as `$HOME\clouddrive`) for all future sessions. Additionally, your `$HOME` directory is persisted as an .img in your Azure File share. Files outside of `$HOME` and machine state are not persisted across sessions. Use best practices when storing secrets such as SSH keys. Services like Azure Key Vault have tutorials for setup.

Learn more about persisting files in Cloud Shell.

Azure drive (Azure:)

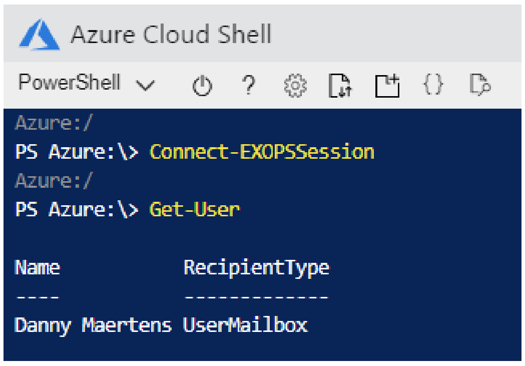
PowerShell in Cloud Shell starts you in Azure drive (`Azure:`). The Azure drive enables easy discovery and navigation of Azure resources such as Compute, Network, Storage etc. similar to filesystem navigation. You can continue to use the familiar Azure PowerShell cmdlets to manage these resources regardless of the drive you are in. Any changes made to the Azure resources, either made directly in Azure portal or through Azure PowerShell cmdlets, are reflected in the Azure drive. You can run `dir -Force` to refresh your resources.



```
Requesting a Cloud Shell...Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell (Preview)

Type "dir" to see your Azure resources
Type "help" to learn about Cloud Shell

VERBOSE: Authenticating to Azure ...
VERBOSE: Building your Azure drive ...
PS Azure:\> cd AutomationTeam
PS Azure:\AutomationTeam> dir


    Directory: Azure:\AutomationTeam


Mode Name
---- ----
+    AllResources
+    ResourceGroups
+    StorageAccounts
+    VirtualMachines
+    WebApps


PS Azure:\AutomationTeam> cd .\ResourceGroups\hemantm
PS Azure:\AutomationTeam\ResourceGroups\hemantm> Get-AzureRmVM
```

Manage Exchange Online

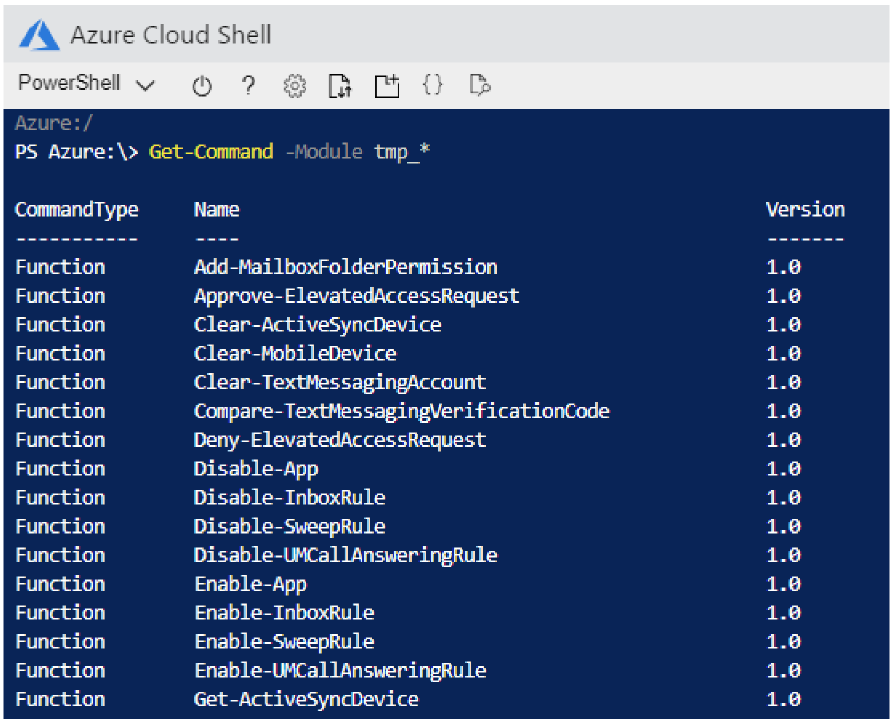
PowerShell in Cloud Shell contains a private build of the Exchange Online module. Run `Connect-EXOPSSession` to get your Exchange cmdlets.



```
Azure:/
PS Azure:\> Connect-EXOPSSession
Azure:/
PS Azure:\> Get-User

Name           RecipientType
----           -------------
Danny Maertens UserMailbox
```

Run `Get-Command -Module tmp_*`

> **NOTE**
>
> The module name should begin with `tmp_`, if you have installed modules with the same prefix, their cmdlets will also be surfaced.



```
Azure:/
PS Azure:\> Get-Command -Module tmp_*

CommandType     Name                                               Version
-----------     ----                                               -------
Function        Add-MailboxFolderPermission                        1.0
Function        Approve-ElevatedAccessRequest                      1.0
Function        Clear-ActiveSyncDevice                             1.0
Function        Clear-MobileDevice                                 1.0
Function        Clear-TextMessagingAccount                         1.0
Function        Compare-TextMessagingVerificationCode              1.0
Function        Deny-ElevatedAccessRequest                         1.0
Function        Disable-App                                        1.0
Function        Disable-InboxRule                                  1.0
Function        Disable-SweepRule                                  1.0
Function        Disable-UMCallAnsweringRule                        1.0
Function        Enable-App                                         1.0
Function        Enable-InboxRule                                   1.0
Function        Enable-SweepRule                                   1.0
Function        Enable-UMCallAnsweringRule                         1.0
Function        Get-ActiveSyncDevice                               1.0
```

Deep integration with open-source tooling

Cloud Shell includes pre-configured authentication for open-source tools such as Terraform, Ansible, and Chef InSpec. Try it out from the example walkthroughs.

Tools

CATEGORY	NAME
Linux tools	bash zsh sh tmux dig
Azure tools	Azure CLI and Azure classic CLI AzCopy Azure Functions CLI Service Fabric CLI Batch Shipyard blobxfer
Text editors	code (Cloud Shell editor) vim nano emacs
Source control	git
Build tools	make maven npm pip

CATEGORY	NAME
Containers	Docker Machine Kubectl Helm DC/OS CLI
Databases	MySQL client PostgreSql client sqlcmd Utility mssql-scripter
Other	iPython Client Cloud Foundry CLI Terraform Ansible Chef InSpec Puppet Bolt HashiCorp Packer

Language support

LANGUAGE	VERSION
.NET Core	2.0.0
Go	1.9
Java	1.8
Node.js	8.9.4
PowerShell	6.2.0
Python	2.7 and 3.5 (default)

Next steps

Bash in Cloud Shell Quickstart
PowerShell in Cloud Shell Quickstart
Learn about Azure CLI
Learn about Azure PowerShell

Deploy with Terraform from Bash in Azure Cloud Shell

8/2/2019 • 3 minutes to read • Edit Online

This article walks you through creating a resource group with the Terraform AzureRM provider.

Hashicorp Terraform is an open source tool that codifies APIs into declarative configuration files that can be shared amongst team members to be edited, reviewed, and versioned. The Microsoft AzureRM provider is used to interact with resources supported by Azure Resource Manager via the AzureRM APIs.

Automatic authentication

Terraform is installed in Bash in Cloud Shell by default. Additionally, Cloud Shell automatically authenticates your default Azure CLI subscription to deploy resources through the Terraform Azure modules.

Terraform uses the default Azure CLI subscription that is set. To update default subscriptions, run:

```
az account set --subscription mySubscriptionName
```

Walkthrough

Launch Bash in Cloud Shell

1. Launch Cloud Shell from your preferred location
2. Verify your preferred subscription is set

```
az account show
```

Create a Terraform template

Create a new Terraform template named main.tf with your preferred text editor.

```
vim main.tf
```

Copy/paste the following code into Cloud Shell.

```
resource "azurerm_resource_group" "myterraformgroup" {
    name = "myRgName"
    location = "West US"
}
```

Save your file and exit your text editor.

Terraform init

Begin by running `terraform init`.

```
justin@Azure:~$ terraform init

Initializing provider plugins...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.azurerm: version = "~> 0.2"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

The terraform init command is used to initialize a working directory containing Terraform configuration files. The `terraform init` command is the first command that should be run after writing a new Terraform configuration or cloning an existing one from version control. It is safe to run this command multiple times.

Terraform plan

Preview the resources to be created by the Terraform template with `terraform plan`.

```
justin@Azure:~$ terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  + azurerm_resource_group.demo
      id:       <computed>
      location: "westus"
      name:     "myRGName"
      tags.%:   <computed>


Plan: 1 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------

Note: You didn't specify an "-out" parameter to save this plan, so Terraform
can't guarantee that exactly these actions will be performed if
"terraform apply" is subsequently run.
```

The terraform plan command is used to create an execution plan. Terraform performs a refresh, unless explicitly disabled, and then determines what actions are necessary to achieve the desired state specified in the configuration files. The plan can be saved using -out, and then provided to terraform apply to ensure only the pre-planned actions are executed.

Terraform apply

Provision the Azure resources with `terraform apply`.

```
justin@Azure:~$ terraform apply
azurerm_resource_group.demo: Creating...
  location: "" => "westus"
  name:     "" => "myRGName"
  tags.%:   "" => "<computed>"
azurerm_resource_group.demo: Creation complete after 0s (ID:
/subscriptions/mySubIDmysub/resourceGroups/myRGName)

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

The terraform apply command is used to apply the changes required to reach the desired state of the configuration.

Verify deployment with Azure CLI

Run `az group show -n myRgName` to verify the resource has succeeded provisioning.

```
az group show -n myRgName
```

Clean up with terraform destroy

Clean up the resource group created with the Terraform destroy command to clean up Terraform-created infrastructure.

```
justin@Azure:~$ terraform destroy
azurerm_resource_group.demo: Refreshing state... (ID: /subscriptions/mySubID/resourceGroups/myRGName)

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  - azurerm_resource_group.demo


Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

azurerm_resource_group.demo: Destroying... (ID: /subscriptions/mySubID/resourceGroups/myRGName)
azurerm_resource_group.demo: Still destroying... (ID: /subscriptions/mySubID/resourceGroups/myRGName, 10s
elapsed)
azurerm_resource_group.demo: Still destroying... (ID: /subscriptions/mySubID/resourceGroups/myRGName, 20s
elapsed)
azurerm_resource_group.demo: Still destroying... (ID: /subscriptions/mySubID/resourceGroups/myRGName, 30s
elapsed)
azurerm_resource_group.demo: Still destroying... (ID: /subscriptions/mySubID/resourceGroups/myRGName, 40s
elapsed)
azurerm_resource_group.demo: Destruction complete after 45s

Destroy complete! Resources: 1 destroyed.
```

You have successfully created an Azure resource through Terraform. Visit next steps to continue learning about

Cloud Shell.

Next steps

Learn about the Terraform Azure provider

Bash in Cloud Shell quickstart

PowerShell in Azure Cloud Shell for Windows users

10/22/2018 • 2 minutes to read • Edit Online

In May 2018, changes were announced to PowerShell in Azure Cloud Shell. The PowerShell experience in Azure Cloud Shell now runs PowerShell Core 6 in a Linux environment. With this change, there may be some differences in the PowerShell experience in Cloud Shell compared to what is expected in a Windows PowerShell experience.

File system case sensitivity

The file system is case-insensitive in Windows, whereas on Linux, the file system is case-sensitive. Previously `file.txt` and `FILE.txt` were considered to be the same file, but now they are considered to be different files. Proper casing must be used while `tab-completing` in the file system. PowerShell specific experiences, such as `tab-completing` cmdlet names, parameters, and values, are not case-sensitive.

Windows PowerShell aliases vs Linux utilities

Some existing PowerShell aliases have the same names as built-in Linux commands, such as `cat`, `ls`, `sort`, `sleep`, etc. In PowerShell Core 6, aliases that collide with built-in Linux commands have been removed. Below are the common aliases that have been removed as well as their equivalent commands:

REMOVED ALIAS	EQUIVALENT COMMAND
`cat`	`Get-Content`
`curl`	`Invoke-WebRequest`
`diff`	`Compare-Object`
`ls`	`dir` `Get-ChildItem`
`mv`	`Move-Item`
`rm`	`Remove-Item`
`sleep`	`Start-Sleep`
`sort`	`Sort-Object`
`wget`	`Invoke-WebRequest`

Persisting $HOME

Earlier users could only persist scripts and other files in their Cloud Drive. Now, the user's $HOME directory is also persisted across sessions.

PowerShell profile

By default, a user's PowerShell profile is not created. To create your profile, create a `PowerShell` directory under `$HOME/.config`.

```
mkdir (Split-Path $profile.CurrentUserAllHosts)
```

Under `$HOME/.config/PowerShell`, you can create your profile files - `profile.ps1` and/or `Microsoft.PowerShell_profile.ps1`.

What's new in PowerShell Core 6

For more information about what is new in PowerShell Core 6, reference the PowerShell docs and the Getting Started with PowerShell Core blog post.

Persist files in Azure Cloud Shell

11/21/2019 • 6 minutes to read • Edit Online

Cloud Shell utilizes Azure File storage to persist files across sessions. On initial start, Cloud Shell prompts you to associate a new or existing file share to persist files across sessions.

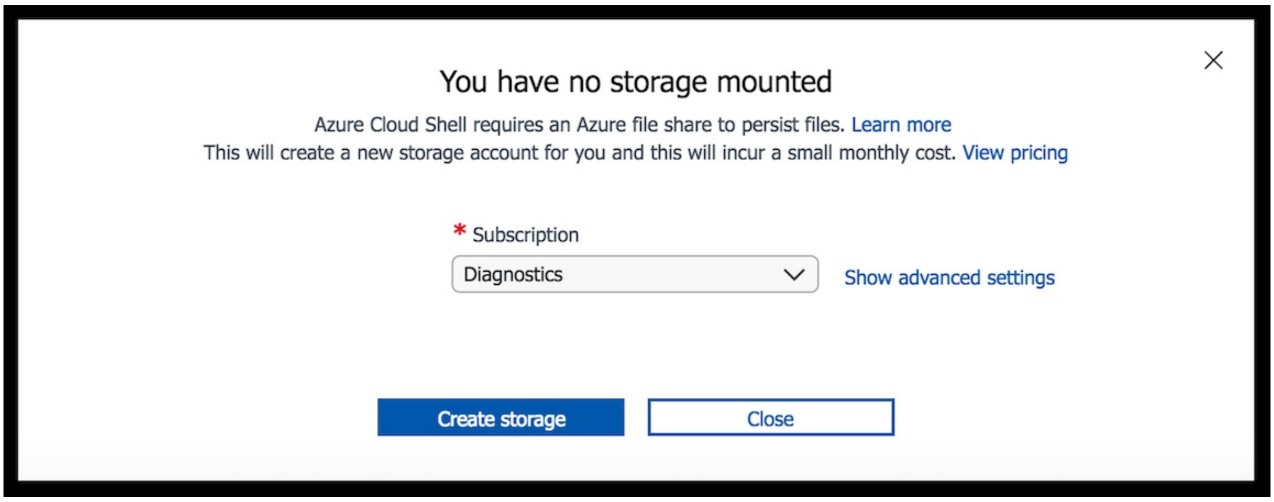
NOTE

Bash and PowerShell share the same file share. Only one file share can be associated with automatic mounting in Cloud Shell.

Create new storage

When you use basic settings and select only a subscription, Cloud Shell creates three resources on your behalf in the supported region that's nearest to you:

- Resource group: `cloud-shell-storage-<region>`
- Storage account: `cs<uniqueGuid>`
- File share: `cs-<user>-<domain>-com-<uniqueGuid>`



You have no storage mounted

Azure Cloud Shell requires an Azure file share to persist files. Learn more
This will create a new storage account for you and this will incur a small monthly cost. View pricing

* Subscription: Diagnostics

The file share mounts as `clouddrive` in your `$Home` directory. This is a one-time action, and the file share mounts automatically in subsequent sessions.

The file share also contains a 5-GB image that is created for you which automatically persists data in your `$Home` directory. This applies for both Bash and PowerShell.

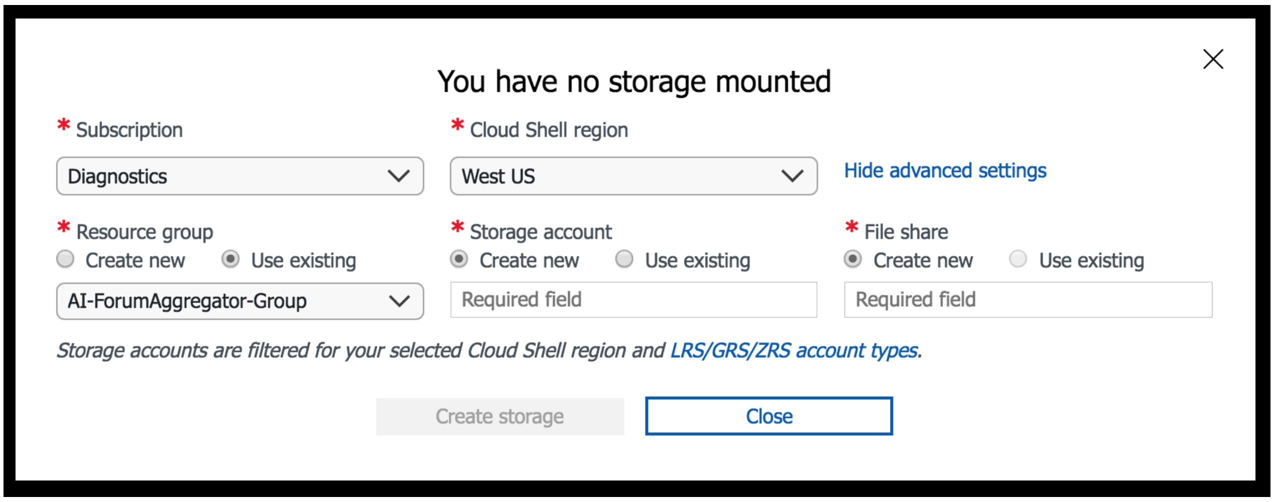
Use existing resources

By using the advanced option, you can associate existing resources. When selecting a Cloud Shell region you must select a backing storage account co-located in the same region. For example, if your assigned region is West US then you must associate a file share that resides within West US as well.

When the storage setup prompt appears, select **Show advanced settings** to view additional options. The populated storage options filter for locally redundant storage (LRS), geo-redundant storage (GRS), and zone-redundant storage (ZRS) accounts.

NOTE

Using GRS or ZRS storage accounts are recommended for additional resiliency for your backing file share. Which type of redundancy depends on your goals and price preference. Learn more about replication options for Azure Storage accounts.


You have no storage mounted

* Subscription: Diagnostics

* Cloud Shell region: West US

Hide advanced settings

* Resource group: Create new [✓ Use existing] — AI-ForumAggregator-Group

* Storage account: [✓ Create new] Use existing — Required field

* File share: [✓ Create new] Use existing — Required field

Storage accounts are filtered for your selected Cloud Shell region and LRS/GRS/ZRS account types.

Securing storage access

For security, each user should provision their own storage account. For role-based access control (RBAC), users must have contributor access or above at the storage account level.

Cloud Shell uses an Azure File Share in a storage account, inside a specified subscription. Due to inherited permissions, users with sufficient access rights to the subscription will be able to access all the storage accounts, and file shares contained in the subscription.

Users should lock down access to their files by setting the permissions at the storage account or the subscription level.

Supported storage regions

Associated Azure storage accounts must reside in the same region as the Cloud Shell machine that you're mounting them to. To find your current region you may run `env` in Bash and locate the variable `ACC_LOCATION`. File shares receive a 5-GB image created for you to persist your `$Home` directory.

Cloud Shell machines exist in the following regions:

AREA	REGION
Americas	East US, South Central US, West US
Europe	North Europe, West Europe
Asia Pacific	India Central, Southeast Asia

Restrict resource creation with an Azure resource policy

Storage accounts that you create in Cloud Shell are tagged with `ms-resource-usage:azure-cloud-shell`. If you want to disallow users from creating storage accounts in Cloud Shell, create an Azure resource policy for tags that are triggered by this specific tag.

How Cloud Shell storage works

Cloud Shell persists files through both of the following methods:

- Creating a disk image of your `$Home` directory to persist all contents within the directory. The disk image is saved in your specified file share as `acc_<User>.img` at `fileshare.storage.windows.net/fileshare/.cloudconsole/acc_<User>.img`, and it automatically syncs changes.
- Mounting your specified file share as `clouddrive` in your `$Home` directory for direct file-share interaction. `/Home/<User>/clouddrive` is mapped to `fileshare.storage.windows.net/fileshare`.

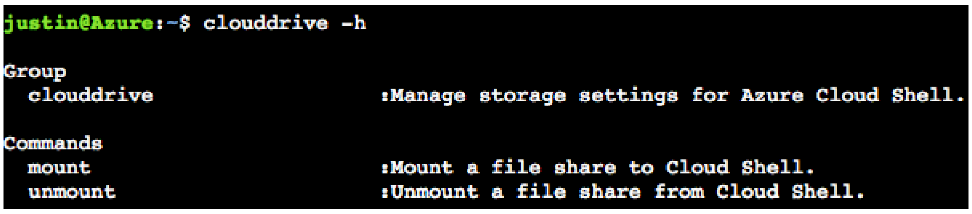
NOTE

All files in your `$Home` directory, such as SSH keys, are persisted in your user disk image, which is stored in your mounted file share. Apply best practices when you persist information in your `$Home` directory and mounted file share.

clouddrive commands

Use the `clouddrive` command

In Cloud Shell, you can run a command called `clouddrive`, which enables you to manually update the file share that is mounted to Cloud Shell.



```
justin@Azure:~$ clouddrive -h

Group
  clouddrive                    :Manage storage settings for Azure Cloud Shell.

Commands
  mount                         :Mount a file share to Cloud Shell.
  unmount                       :Unmount a file share from Cloud Shell.
```

List `clouddrive`

To discover which file share is mounted as `clouddrive`, run the `df` command.

The file path to clouddrive shows your storage account name and file share in the URL. For example, `//storageaccountname.file.core.windows.net/filesharename`

```
justin@Azure:~$ df
Filesystem                                          1K-blocks   Used  Available Use% Mounted on
overlay                                             29711408 5577940   24117084  19% /
tmpfs                                                 986716       0     986716   0% /dev
tmpfs                                                 986716       0     986716   0% /sys/fs/cgroup
/dev/sda1                                           29711408 5577940   24117084  19% /etc/hosts
shm                                                    65536       0      65536   0% /dev/shm
//mystoragename.file.core.windows.net/fileshareName 5368709120    64 5368709056   1% /home/justin/clouddrive
justin@Azure:~$
```

Mount a new clouddrive

Prerequisites for manual mounting

You can update the file share that's associated with Cloud Shell by using the `clouddrive mount` command.

If you mount an existing file share, the storage accounts must be located in your select Cloud Shell region. Retrieve the location by running `env` and checking the `ACC_LOCATION`.

The `clouddrive mount` command

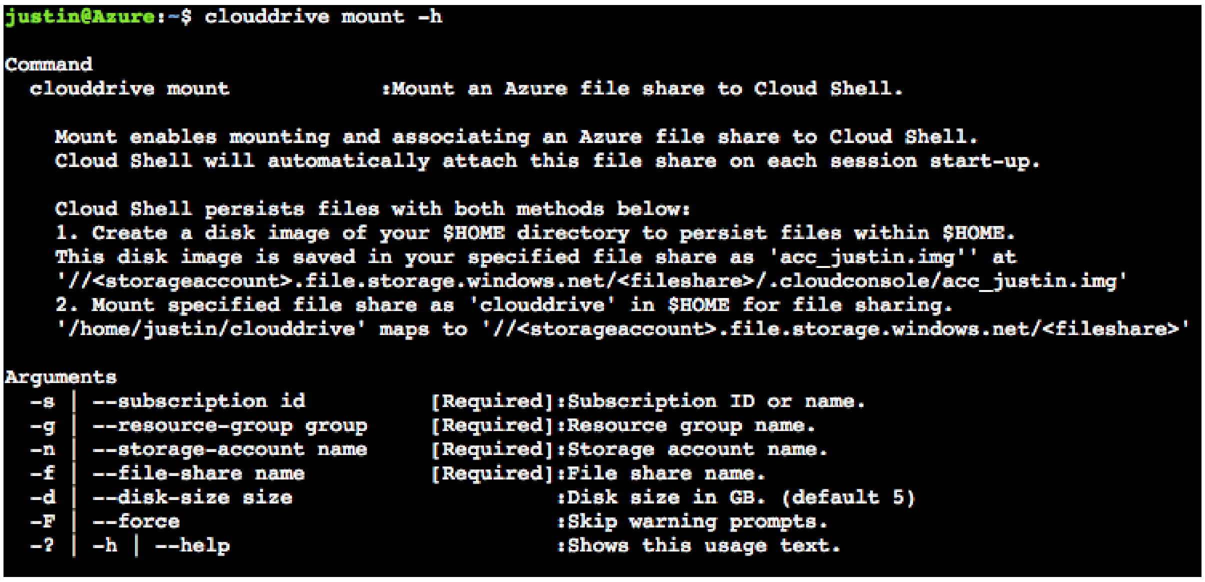
NOTE

If you're mounting a new file share, a new user image is created for your `$Home` directory. Your previous `$Home` image is kept in your previous file share.

Run the `clouddrive mount` command with the following parameters:

```
clouddrive mount -s mySubscription -g myRG -n storageAccountName -f fileShareName
```

To view more details, run `clouddrive mount -h`, as shown here:



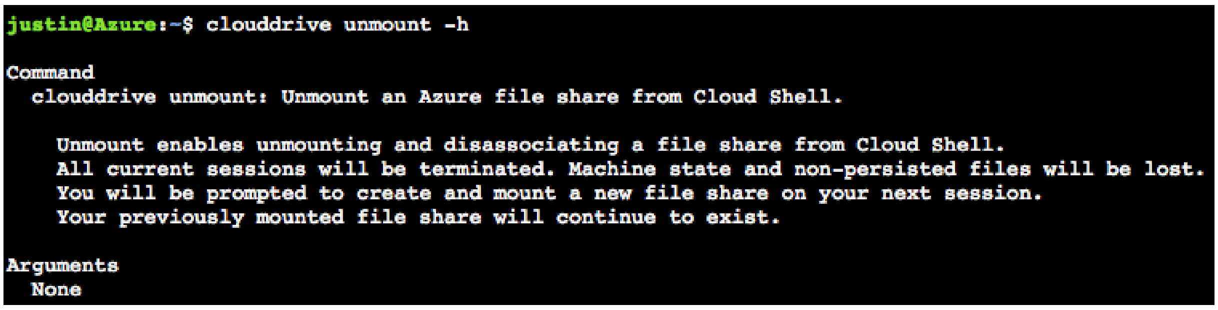
```
justin@Azure:~$ clouddrive mount -h

Command
  clouddrive mount            :Mount an Azure file share to Cloud Shell.

    Mount enables mounting and associating an Azure file share to Cloud Shell.
    Cloud Shell will automatically attach this file share on each session start-up.

    Cloud Shell persists files with both methods below:
    1. Create a disk image of your $HOME directory to persist files within $HOME.
    This disk image is saved in your specified file share as 'acc_justin.img'' at
    '//<storageaccount>.file.storage.windows.net/<fileshare>/.cloudconsole/acc_justin.img'
    2. Mount specified file share as 'clouddrive' in $HOME for file sharing.
    '/home/justin/clouddrive' maps to '//<storageaccount>.file.storage.windows.net/<fileshare>'

Arguments
  -s | --subscription id        [Required]:Subscription ID or name.
  -g | --resource-group group   [Required]:Resource group name.
  -n | --storage-account name   [Required]:Storage account name.
  -f | --file-share name        [Required]:File share name.
  -d | --disk-size size                   :Disk size in GB. (default 5)
  -F | --force                            :Skip warning prompts.
  -? | -h | --help                        :Shows this usage text.
```

Unmount clouddrive

You can unmount a file share that's mounted to Cloud Shell at any time. Since Cloud Shell requires a mounted file share to be used, you will be prompted to create and mount another file share on the next session.

1. Run `clouddrive unmount`.
2. Acknowledge and confirm prompts.

Your file share will continue to exist unless you delete it manually. Cloud Shell will no longer search for this file share on subsequent sessions. To view more details, run `clouddrive unmount -h`, as shown here:



```
justin@Azure:~$ clouddrive unmount -h

Command
  clouddrive unmount: Unmount an Azure file share from Cloud Shell.

    Unmount enables unmounting and disassociating a file share from Cloud Shell.
    All current sessions will be terminated. Machine state and non-persisted files will be lost.
    You will be prompted to create and mount a new file share on your next session.
    Your previously mounted file share will continue to exist.

Arguments
  None
```

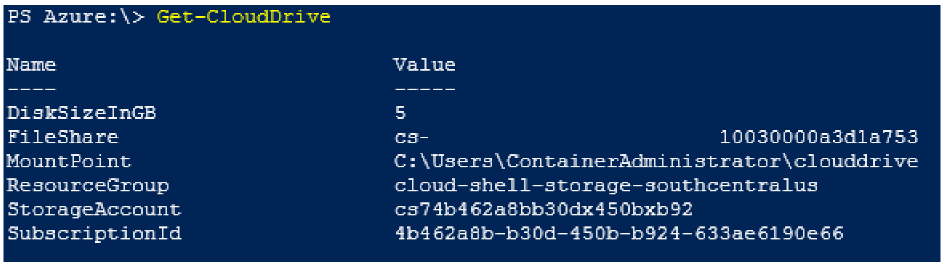
WARNING

Although running this command will not delete any resources, manually deleting a resource group, storage account, or file share that's mapped to Cloud Shell erases your `$Home` directory disk image and any files in your file share. This action cannot be undone.

PowerShell-specific commands

List `clouddrive` Azure file shares

The `Get-CloudDrive` cmdlet retrieves the Azure file share information currently mounted by the `clouddrive` in the Cloud Shell.

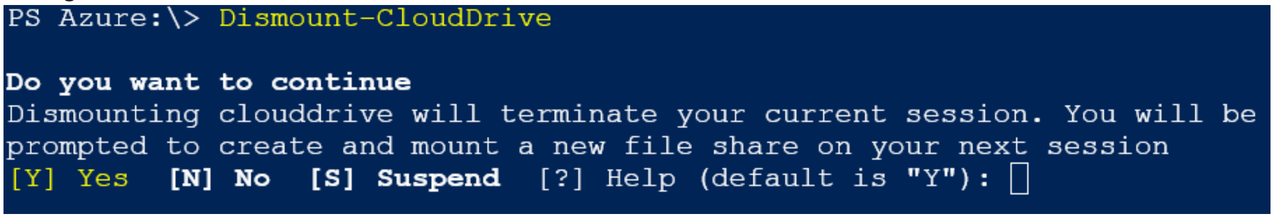


```
PS Azure:\> Get-CloudDrive

Name                          Value
----                          -----
DiskSizeInGB                  5
FileShare                     cs-                      10030000a3d1a753
MountPoint                    C:\Users\ContainerAdministrator\clouddrive
ResourceGroup                 cloud-shell-storage-southcentralus
StorageAccount                cs74b462a8bb30dx450bxb92
SubscriptionId                4b462a8b-b30d-450b-b924-633ae6190e66
```

Unmount `clouddrive`

You can unmount an Azure file share that's mounted to Cloud Shell at any time. If the Azure file share has been removed, you will be prompted to create and mount a new Azure file share at the next session.

The `Dismount-CloudDrive` cmdlet unmounts an Azure file share from the current storage account. Dismounting the `clouddrive` terminates the current session. The user will be prompted to create and mount a new Azure file share during the next session.

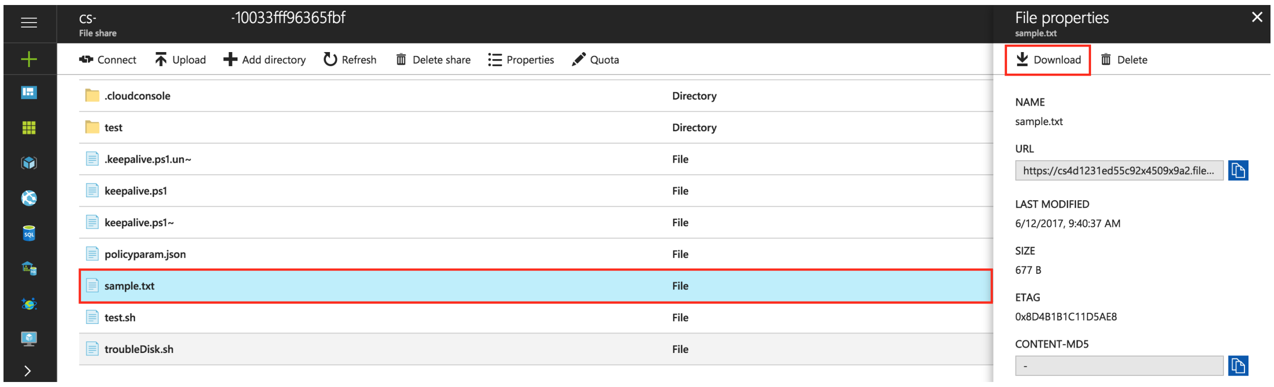


```
PS Azure:\> Dismount-CloudDrive

Do you want to continue
Dismounting clouddrive will terminate your current session. You will be
prompted to create and mount a new file share on your next session
[Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"):
```

Transfer local files to Cloud Shell

The `clouddrive` directory syncs with the Azure portal storage blade. Use this blade to transfer local files to or from your file share. Updating files from within Cloud Shell is reflected in the file storage GUI when you refresh the blade.

Download files



cs-   -10033fff96365fbf
File share

Connect Upload Add directory Refresh Delete share Properties Quota

.cloudconsole	Directory
test	Directory
.keepalive.ps1.un~	File
keepalive.ps1	File
keepalive.ps1~	File
policyparam.json	File
sample.txt	File
test.sh	File
troubleDisk.sh	File

File properties
sample.txt

Download Delete

NAME
sample.txt

URL
https://cs4d1231ed55c92x4509x9a2.file...

LAST MODIFIED
6/12/2017, 9:40:37 AM

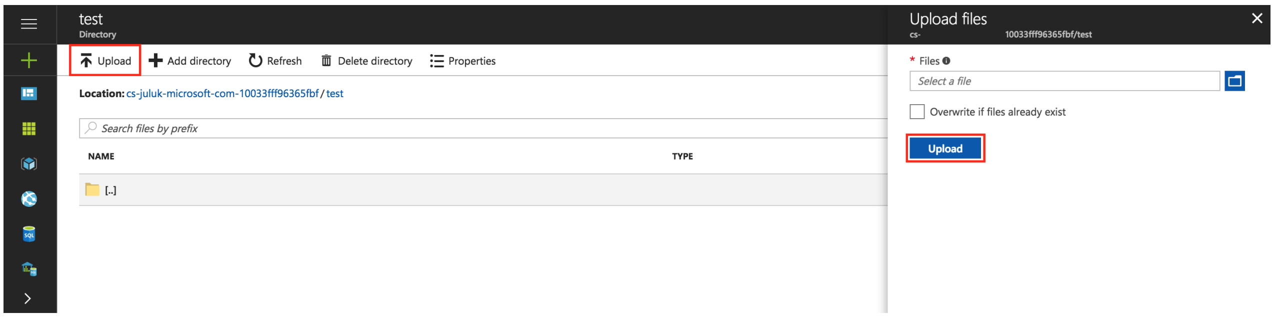
SIZE
677 B

ETAG
0x8D4B1B1C11D5AE8

CONTENT-MD5
-

1. In the Azure portal, go to the mounted file share.
2. Select the target file.
3. Select the **Download** button.

Upload files



1. Go to your mounted file share.
2. Select the **Upload** button.
3. Select the file or files that you want to upload.
4. Confirm the upload.

You should now see the files that are accessible in your `clouddrive` directory in Cloud Shell.

Note: If you need to define a function in a file and call it from the PowerShell cmdlets, then the dot operator must be included. For example: . .\MyFunctions.ps1

Next steps

Cloud Shell Quickstart
Learn about Microsoft Azure Files storage
Learn about storage tags

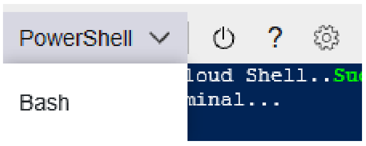
Using the Azure Cloud Shell window

9/10/2019 • 2 minutes to read • Edit Online

This document explains how to use the Cloud Shell window.

Swap between Bash and PowerShell environments

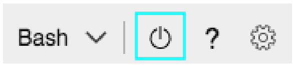
Use the environment selector in the Cloud Shell toolbar to swap between Bash and PowerShell environments.



```
loud Shell..Su
minal...
```

Restart Cloud Shell

Click the restart icon in the Cloud Shell toolbar to reset machine state.

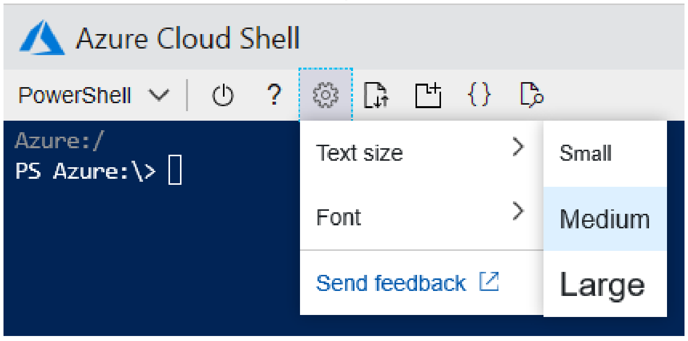


WARNING

Restarting Cloud Shell will reset machine state and any files not persisted by your Azure file share will be lost.

Change the text size

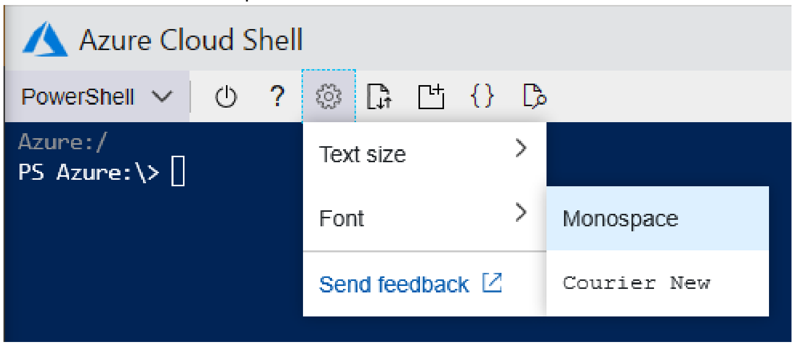
Click the settings icon on the top left of the window, then hover over the "Text size" option and select your desired text size. Your selection will be persisted across sessions.



```
Azure:/
PS Azure:\>
```

Change the font

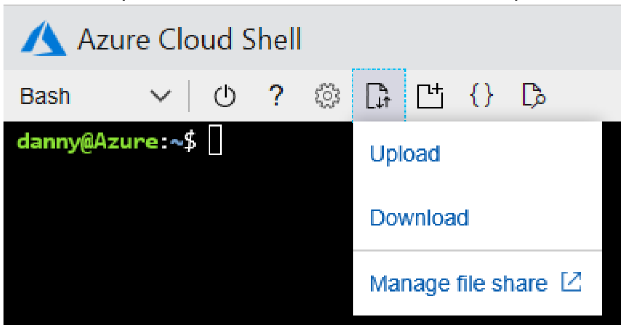
Click the settings icon on the top left of the window, then hover over the "Font" option and select your desired font. Your selection will be persisted across sessions.



```
Azure:/
PS Azure:\>
```

Upload and download files

Click the upload/download files icon on the top left of the window, then select upload or download.



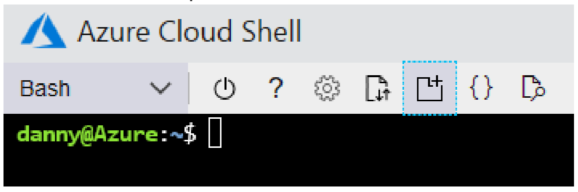
```
danny@Azure:~$
```

- For uploading files, use the pop-up to browse to the file on your local computer, select the desired file, and click the "Open" button. The file will be uploaded into the `/home/user` directory.
- For downloading file, enter the fully qualified file path into the pop-up window (i.e., basically a path under the `/home/user` directory which shows up by default), and select the "Download" button.

> **NOTE**
>
> Files and file paths are case sensitive in Cloud Shell. Double check your casing in your file path.

Open another Cloud Shell window

Cloud Shell enables multiple concurrent sessions across browser tabs by allowing each session to exist as a separate process. If exiting a session, be sure to exit from each session window as each process runs independently although they run on the same machine.
Click the open new session icon on the top left of the window. A new tab will open with another session connected



```
danny@Azure:~$
```

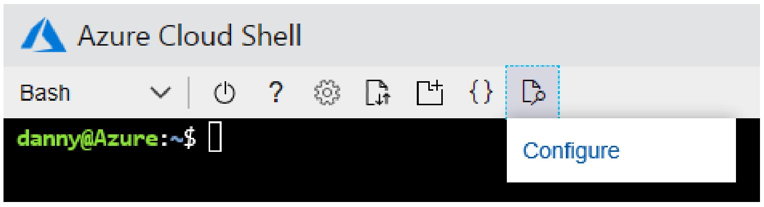
to the existing container.

Cloud Shell editor

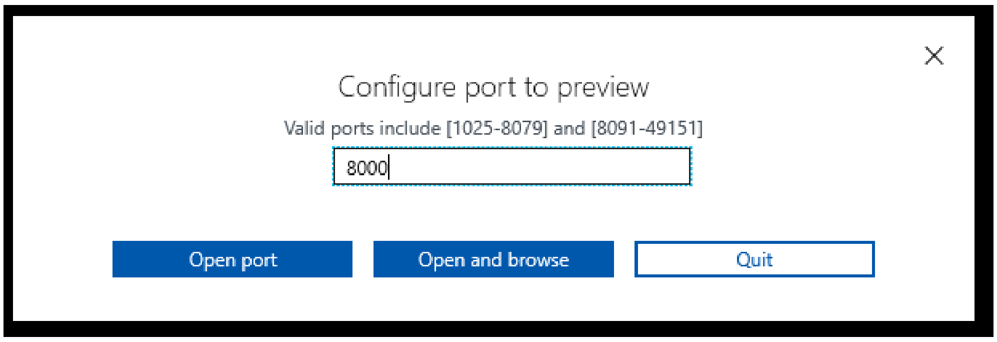
- Refer to the Using the Azure Cloud Shell editor page.

Web preview

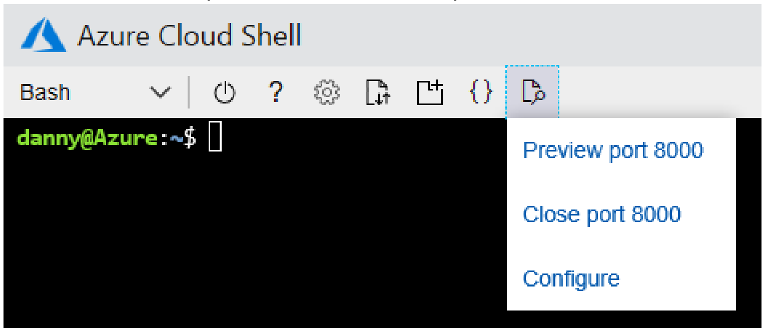
Click the web preview icon on the top left of the window, select "Configure", specify the desired port to open. Select either "Open port" to only open the port, or "Open and browse" to open the port and preview the port in a new tab.



```
danny@Azure:~$
```


Configure port to preview
Valid ports include [1025-8079] and [8091-49151]
8000
Open port
Open and browse
Quit

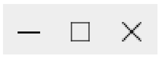
Click the web preview icon on the top left of the window, select "Preview port ..." to preview an open port in a new tab. Click the web preview icon on the top left of the window, select "Close port ..." to close the open port.



```
danny@Azure:~$
```

Minimize & maximize Cloud Shell window

Click the minimize icon on the top right of the window to hide it. Click the Cloud Shell icon again to unhide. Click the maximize icon to set window to max height. To restore window to previous size, click restore.



Copy and paste

- Windows: `Ctrl-c` to copy is supported but use `Shift-insert` to paste.
  - FireFox/IE may not support clipboard permissions properly.
- Mac OS: `Cmd-c` to copy and `Cmd-v` to paste.

Resize Cloud Shell window

Click and drag the top edge of the toolbar up or down to resize the Cloud Shell window.

Scrolling text display

Scroll with your mouse or touchpad to move terminal text.

Exit command

Running `exit` terminates the active session. This behavior occurs by default after 20 minutes without interaction.

Next steps

Bash in Cloud Shell Quickstart
PowerShell in Cloud Shell Quickstart

Use managed identities for Azure resources in Azure Cloud Shell

10/14/2019 • 2 minutes to read • Edit Online

Azure Cloud Shell supports authorization with managed identities for Azure resources. Utilize this to retrieve access tokens to securely communicate with Azure services.

About managed identities for Azure resources

A common challenge when building cloud applications is how to securely manage the credentials that need to be in your code for authenticating to cloud services. In Cloud Shell you may need to authenticate retrieval from Key Vault for a credential that a script may need.

Managed identities for Azure resources makes solving this problem simpler by giving Azure services an automatically managed identity in Azure Active Directory (Azure AD). You can use this identity to authenticate to any service that supports Azure AD authentication, including Key Vault, without having any credentials in your code.

Acquire access token in Cloud Shell

Execute the following commands to set your MSI access token as an environment variable, `access_token`.

```
response=$(curl http://localhost:50342/oauth2/token --data "resource=https://management.azure.com/" -H
Metadata:true -s)
access_token=$(echo $response | python -c 'import sys, json; print (json.load(sys.stdin)["access_token"])')
echo The MSI access token is $access_token
```

Handling token expiration

The local MSI subsystem caches tokens. Therefore, you can call it as often as you like, and an on-the-wire call to Azure AD results only if:

- a cache miss occurs due to no token in the cache
- the token is expired

If you cache the token in your code, you should be prepared to handle scenarios where the resource indicates that the token is expired.

To handle token errors, visit the MSI page about curling MSI access tokens.

Next steps

Learn more about MSI
Acquiring access tokens from MSI VMs

Embed Azure Cloud Shell

3/14/2019 • 2 minutes to read • Edit Online

Embedding Cloud Shell enables developers and content writers to directly open Cloud Shell from a dedicated URL, shell.azure.com. This immediately brings the full power of Cloud Shell's authentication, tooling, and up-to-date Azure CLI/Azure PowerShell tools to your users.

Regular sized button

Large sized button

How-to

Integrate Cloud Shell's launch button into markdown files by copying the following:

```
[![Launch Cloud Shell](https://shell.azure.com/images/launchcloudshell.png "Launch Cloud Shell")]
(https://shell.azure.com)
```

The HTML to embed a pop-up Cloud Shell is below:

```
<a style="cursor:pointer" onclick='javascript:window.open("https://shell.azure.com", "_blank",
"toolbar=no,scrollbars=yes,resizable=yes,menubar=no,location=no,status=no")'><img alt="Launch Azure Cloud
Shell" src="https://shell.azure.com/images/launchcloudshell.png" /></a>
```

Customize experience

Set a specific shell experience by augmenting your URL.

EXPERIENCE	URL
Most recently used shell	shell.azure.com
Bash	shell.azure.com/bash
PowerShell	shell.azure.com/powershell

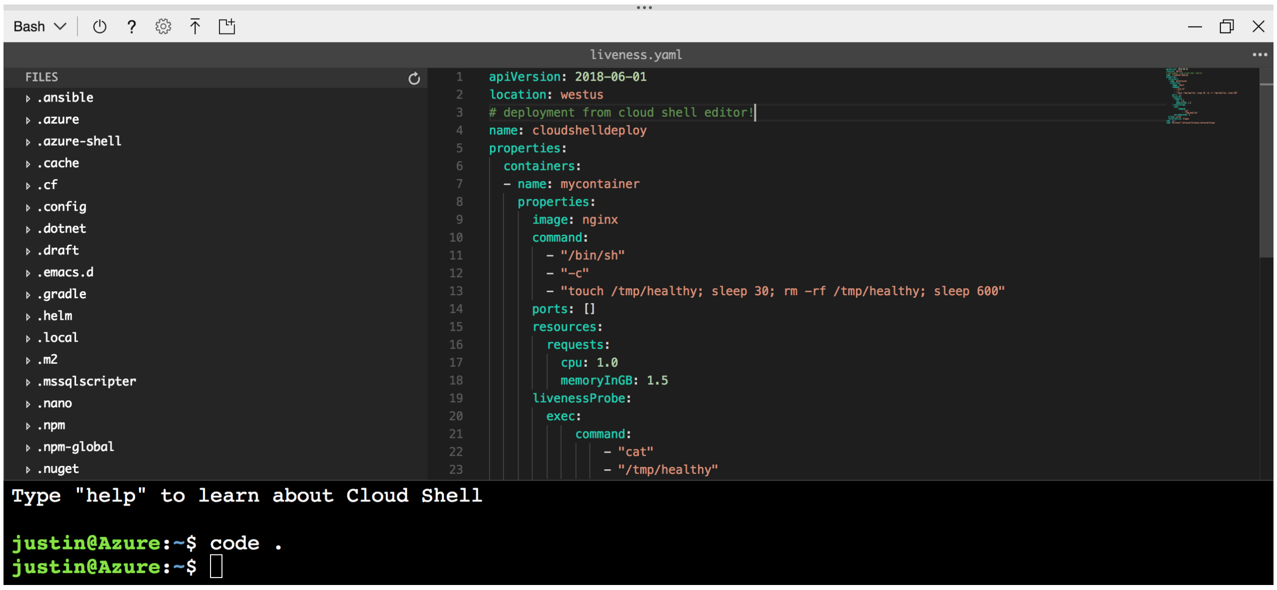
Next steps

Bash in Cloud Shell quickstart
PowerShell in Cloud Shell quickstart

Using the Azure Cloud Shell editor

3/1/2019 • 2 minutes to read • Edit Online

Azure Cloud Shell includes an integrated file editor built from the open-source Monaco Editor. The Cloud Shell editor supports features such as language highlighting, the command palette, and a file explorer.



```yaml
apiVersion: 2018-06-01
location: westus
# deployment from cloud shell editor!
name: cloudshelldeploy
properties:
  containers:
  - name: mycontainer
    properties:
      image: nginx
      command:
        - "/bin/sh"
        - "-c"
        - "touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600"
      ports: []
      resources:
        requests:
          cpu: 1.0
          memoryInGB: 1.5
      livenessProbe:
        exec:
            command:
                - "cat"
                - "/tmp/healthy"
```

```
Type "help" to learn about Cloud Shell

justin@Azure:~$ code .
justin@Azure:~$ 
```

Opening the editor

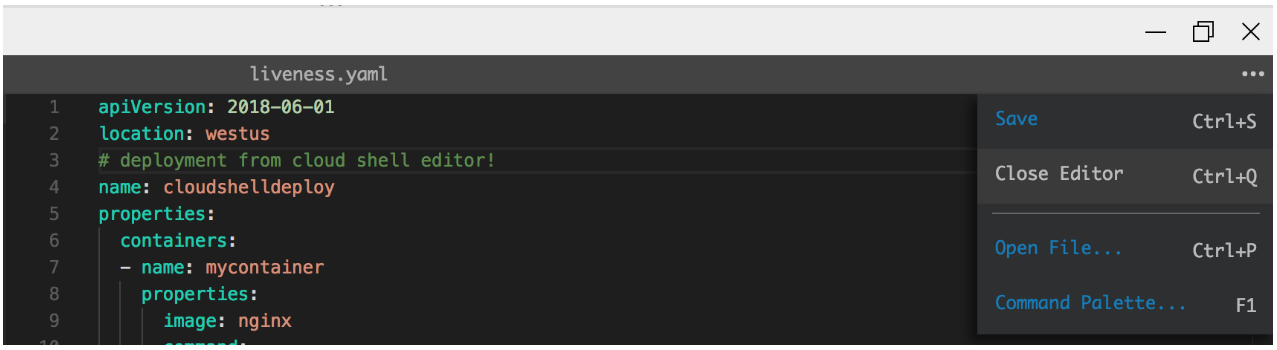
For simple file creation and editing, launch the editor by running `code .` in the Cloud Shell terminal. This action opens the editor with your active working directory set in the terminal.

To directly open a file for quick editing, run `code <filename>` to open the editor without the file explorer.

To open the editor via UI button, click the `{}` editor icon from the toolbar. This will open the editor and default the file explorer to the `/home/<user>` directory.

Closing the editor

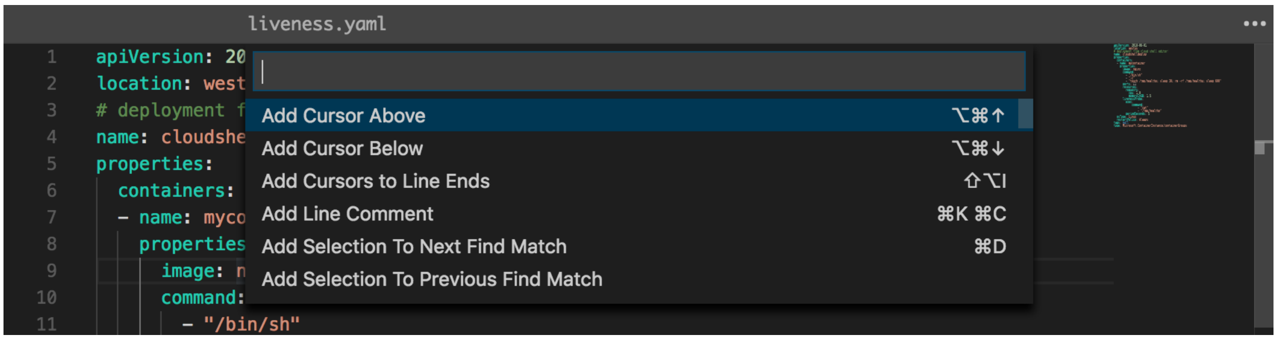
To close the editor, open the `...` action panel in the top right of the editor and select `Close editor`.



```yaml
apiVersion: 2018-06-01
location: westus
# deployment from cloud shell editor!
name: cloudshelldeploy
properties:
  containers:
  - name: mycontainer
    properties:
      image: nginx
```

Command palette

To launch the command palette, use the `F1` key when focus is set on the editor. Opening the command palette can also be done through the action panel.



```
liveness.yaml
apiVersion: 20
location: west
# deployment f
name: cloudshe
properties:
  containers:
  - name: myco
    properties
      image: n
      command:
      - "/bin/sh"
```

Add Cursor Above ⌥⌘↑
Add Cursor Below ⌥⌘↓
Add Cursors to Line Ends ⇧⌥I
Add Line Comment ⌘K ⌘C
Add Selection To Next Find Match ⌘D
Add Selection To Previous Find Match

Contributing to the Monaco Editor

Language highlight support in the Cloud Shell editor is supported through upstream functionality in the Monaco Editor's use of Monarch syntax definitions. To learn how to make contributions, read the Monaco contributor guide.

Next steps

Try the quickstart for Bash in Cloud Shell View the full list of integrated Cloud Shell tools

Troubleshooting & Limitations of Azure Cloud Shell

11/12/2019 • 7 minutes to read • Edit Online

Known resolutions for troubleshooting issues in Azure Cloud Shell include:

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

General troubleshooting

Early timeouts in FireFox

- **Details**: Cloud Shell utilizes an open websocket to pass input/output to your browser. FireFox has preset policies that can close the websocket prematurely causing early timeouts in Cloud Shell.
- **Resolution**: Open FireFox and navigate to "about:config" in the URL box. Search for "network.websocket.timeout.ping.request" and change the value from 0 to 10.

Disabling Cloud Shell in a locked down network environment

- **Details**: Administrators may wish to disable access to Cloud Shell for their users. Cloud Shell utilizes access to the `ux.console.azure.com` domain, which can be denied, stopping any access to Cloud Shell's entrypoints including portal.azure.com, shell.azure.com, Visual Studio Code Azure Account extension, and docs.microsoft.com.
- **Resolution**: Restrict access to `ux.console.azure.com` via network settings to your environment. The Cloud Shell icon will still exist in portal.azure.com, but will not successfully connect to the service.

Storage Dialog - Error: 403 RequestDisallowedByPolicy

- **Details**: When creating a storage account through Cloud Shell, it is unsuccessful due to an Azure policy placed by your admin. Error message will include:
  `The resource action 'Microsoft.Storage/storageAccounts/write' is disallowed by one or more policies.`
- **Resolution**: Contact your Azure administrator to remove or update the Azure policy denying storage creation.

Storage Dialog - Error: 400 DisallowedOperation

- **Details**: When using an Azure Active Directory subscription, you cannot create storage.
- **Resolution**: Use an Azure subscription capable of creating storage resources. Azure AD subscriptions are not able to create Azure resources.

Terminal output - Error: Failed to connect terminal: websocket cannot be established. Press `Enter` to reconnect.

- **Details**: Cloud Shell requires the ability to establish a websocket connection to Cloud Shell infrastructure.
- **Resolution**: Check you have configured your network settings to enable sending https requests and websocket requests to domains at *.console.azure.com.

Set your Cloud Shell connection to support using TLS 1.2

- **Details**: To define the version of TLS for your connection to Cloud Shell, you must set browser-specific settings.
- **Resolution**: Navigate to the security settings of your browser and select the checkbox next to "Use TLS 1.2".

Bash troubleshooting

Cannot run the docker daemon

- **Details**: Cloud Shell utilizes a container to host your shell environment, as a result running the daemon is disallowed.
- **Resolution**: Utilize docker-machine, which is installed by default, to manage docker containers from a remote Docker host.

PowerShell troubleshooting

GUI applications are not supported

- **Details**: If a user launches a GUI application, the prompt does not return. For example, when one clone a private GitHub repo that has two factor authentication enabled, a dialog box is displayed for completing the two factor authentication.
- **Resolution**: Close and reopen the shell.

Troubleshooting remote management of Azure VMs

> **NOTE**
>
> Azure VMs must have a Public facing IP address.

- **Details**: Due to the default Windows Firewall settings for WinRM the user may see the following error:
  ```
  Ensure the WinRM service is running. Remote Desktop into the VM for the first time and ensure it can be discovered.
  ```
- **Resolution**: Run `Enable-AzVMPSRemoting` to enable all aspects of PowerShell remoting on the target machine.

`dir` does not update the result in Azure drive

- **Details**: By default, to optimize for user experience, the results of `dir` is cached in Azure drive.
- **Resolution**: After you create, update or remove an Azure resource, run `dir -force` to update the results in the Azure drive.

General limitations

Azure Cloud Shell has the following known limitations:

Quota limitations

Azure Cloud Shell has a limit of 20 concurrent users per tenant per region. If you try to open more simultaneous sessions than the limit, you will see an "Tenant User Over Quota" error. If you have a legitimate need to have more sessions open than this (for example for training sessions), contact support in advance of your anticipated usage to request a quota increase.

Cloud Shell is provided as a free service and is designed to be used to configure your Azure environment, not as a general purpose computing platform. Excessive automated usage may be considered in breach to the Azure Terms of Service and could lead to Cloud Shell access being blocked.

System state and persistence

The machine that provides your Cloud Shell session is temporary, and it is recycled after your session is inactive for 20 minutes. Cloud Shell requires an Azure file share to be mounted. As a result, your subscription must be able to set up storage resources to access Cloud Shell. Other considerations include:

- With mounted storage, only modifications within the `clouddrive` directory are persisted. In Bash, your `$HOME` directory is also persisted.
- Azure file shares can be mounted only from within your assigned region.

- In Bash, run `env` to find your region set as `ACC_LOCATION`.
- Azure Files supports only locally redundant storage and geo-redundant storage accounts.

Browser support

Cloud Shell supports the latest versions of following browsers:

- Microsoft Edge
- Microsoft Internet Explorer
- Google Chrome
- Mozilla Firefox
- Apple Safari
  - Safari in private mode is not supported.

Copy and paste

- Windows: `Ctrl-c` to copy is supported but use `Shift-insert` to paste.
  - FireFox/IE may not support clipboard permissions properly.
- Mac OS: `Cmd-c` to copy and `Cmd-v` to paste.

Usage limits

Cloud Shell is intended for interactive use cases. As a result, any long-running non-interactive sessions are ended without warning.

User permissions

Permissions are set as regular users without sudo access. Any installation outside your `$Home` directory is not persisted.

Bash limitations

Editing .bashrc

Take caution when editing .bashrc, doing so can cause unexpected errors in Cloud Shell.

PowerShell limitations

Preview version of AzureAD module

Currently, `AzureAD.Standard.Preview`, a preview version of .NET Standard-based, module is available. This module provides the same functionality as `AzureAD`.

`SqlServer` module functionality

The `SqlServer` module included in Cloud Shell has only prerelease support for PowerShell Core. In particular, `Invoke-SqlCmd` is not available yet.

Default file location when created from Azure drive

Using PowerShell cmdlets, users cannot create files under the Azure drive. When users create new files using other tools, such as vim or nano, the files are saved to the `$HOME` by default.

Tab completion can throw PSReadline exception

If the user's PSReadline EditMode is set to Emacs, the user tries to display all possibilities via tab completion, and the window size is too small to display all the possibilities, PSReadline will throw unhandled exception.

Large gap after displaying progress bar

If a command or user action displays a progress bar, such a tab completing while in the `Azure:` drive, then it is possible that the cursor is not set properly and a gap appears where the progress bar was previously.

Random characters appear inline

The cursor position sequence codes, for example `5;13R`, can appear in the user input. The characters can be manually removed.

Personal data in Cloud Shell

Azure Cloud Shell takes your personal data seriously, the data captured and stored by the Azure Cloud Shell service are used to provide defaults for your experience such as your most recently used shell, preferred font size, preferred font type, and file share details that back cloud drive. Should you wish to export or delete this data, use the following instructions.

> **NOTE**
>
> This article provides steps for how to delete personal data from the device or service and can be used to support your obligations under the GDPR. If you're looking for general info about GDPR, see the GDPR section of the Service Trust portal.

Export

In order to **export** the user settings Cloud Shell saves for you such as preferred shell, font size, and font type run the following commands.

1. Launch Cloud Shell
2. Run the following commands in Bash or PowerShell:

Bash:

```
token="Bearer $(curl http://localhost:50342/oauth2/token --data "resource=https://management.azure.com/" -H
Metadata:true -s | jq -r ".access_token")"
curl https://management.azure.com/providers/Microsoft.Portal/usersettings/cloudconsole?api-version=2017-12-01-
preview -H Authorization:"$token" -s | jq
```

PowerShell:

```
$token= ((Invoke-WebRequest -Uri "$env:MSI_ENDPOINT`?resource=https://management.core.windows.net/" -Headers
@{Metadata='true'}).content |  ConvertFrom-Json).access_token
((Invoke-WebRequest -Uri https://management.azure.com/providers/Microsoft.Portal/usersettings/cloudconsole?api-
version=2017-12-01-preview -Headers @{Authorization = "Bearer $token"}).Content | ConvertFrom-Json).properties
| Format-List
```

Delete

In order to **delete** your user settings Cloud Shell saves for you such as preferred shell, font size, and font type run the following commands. The next time you start Cloud Shell you will be asked to onboard a file share again.

> **NOTE**
>
> If you delete your user settings, the actual Azure Files share will not be deleted. Go to your Azure Files to complete that action.

1. Launch Cloud Shell
2. Run the following commands in Bash or PowerShell:

Bash:

```
token="Bearer $(curl http://localhost:50342/oauth2/token --data "resource=https://management.azure.com/" -H
Metadata:true -s | jq -r ".access_token")"
curl -X DELETE https://management.azure.com/providers/Microsoft.Portal/usersettings/cloudconsole?api-
version=2017-12-01-preview -H Authorization:"$token"
```

PowerShell:

```
$token= ((Invoke-WebRequest -Uri "$env:MSI_ENDPOINT`?resource=https://management.core.windows.net/" -Headers
@{Metadata='true'}).content |  ConvertFrom-Json).access_token
Invoke-WebRequest -Method Delete -Uri
https://management.azure.com/providers/Microsoft.Portal/usersettings/cloudconsole?api-version=2017-12-01-
preview -Headers @{Authorization = "Bearer $token"}
```

Azure Government limitations

Azure Cloud Shell in Azure Government is only accessible through the Azure portal.